DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

III SEMESTER

COMPUTER ORGANIZATION AND ARCHITECTURE

SYLLABUS

UNIT – I

BASIC STRUCTURES OF COMPUTER: Functional Units, Multiprocessors and Multi computers, Memory Locations and Addresses, Memory operations, Instructions and Instruction Sequencing, Addressing modes, Assembly Language, Basic Input/output operations, Stacks and Queues, Subroutines, Shift and rotate Instructions, Byte-Sorting program.

UNIT – II

The IA-32 Pentium Example: Registers and Addressing, IA-32 Instructions, IA-32 Assembly Language, Program Flow Control, Logic and Shift/Rotate Instructions, I/O Operations, Subroutines, Other Instructions, Program Examples.

UNIT – III

INPUT/OUTPUT ORGANIZATION: Accessing I/O Devices, Interrupts, Interrupt Hardware, Enabling and Disabling Interrupts, Handling Multiple Devices, Controlling Device Requests, Exceptions, Use Of Interrupts in Operating Systems, Pentium Interrupt Structure, Direct Memory Access, Busses, Interface Circuits, Standard I/O Interfaces.

UNIT – IV

THE MEMORY SYSTEM: Some Basic Concepts, Semiconductor RAM Memories, Read-Only Memories, Speed, Size, and Cost, Cache Memories, Performance Considerations, Virtual memories, Memory Management requirements, Secondary Storage.

UNIT – V

BASIC PROCESSING UNIT : Some Fundamental Concepts, Execution Of a Complete Instruction, Multiple-Bus Organization, Hardwired Control, Microprogrammed Control, PIPELINING: Basic Concepts, Data Hazards, Instruction Hazards, Influence On Instructions Sets, Datapath and Control Considerations, Superscalar Operations, Performance Considerations

TEXT BOOKS

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, fifth edition, TataMcGraw Hill Education, 2011.

REFERENCES

1. John P. Hayes, “Computer Architecture and Organization”, Third edition, Tata McGraw Hill, 2013
2. William Stallings, “Computer organization and Architecture – Designing for performance”, 9th edition, Pearson education, 2012
3. Computer System Architecture – M.Moris Mano, IIIrd Edition, PHI / Pearson, 2006.

UNIT I

BASIC STRUCTURES OF COMPUTER: Functional Units, Multiprocessors and Multi computers, Memory Locations and Addresses, Memory operations, Instructions and Instruction Sequencing, Addressing modes, Assembly Language, Basic Input/output operations, Stacks and Queues, Subroutines, Shift and rotate Instructions, Byte-Sorting program.

2 Marks

1. What is Computer Organization?

The organization of the computer is defined by its internal registers, the timing and control structure, and the set of instructions that it uses.

2. What is meant by Computer Architecture?

- It is concerned with the structure and behaviour of the computer.
- It includes the information formats, the instruction set and techniques for addressing memory.

3. What is difference between Computer Architecture and Computer Organization? (Apr 12)

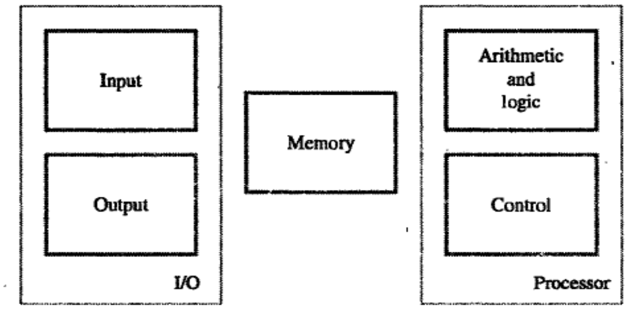
S.No	Computer Architecture	Computer Organization
1.	It refers to the attributes that have a direct impact on the logical execution of the program.	It refers to the operational units and their interconnections that realize the architectural specifications
2.	Architectural attributes includes the Instruction set, data types, no of bits used to represent the data, I/O mechanisms.	Organizational attributes include those h/w details such as control signals, interfaces b/w the computer memory & I/O peripherals

4. List the various functional units of a computer

A computer consists of 5 main parts.

- Input
- Memory
- Arithmetic and logic
- Output
- Control Units

5. Draw the structure of functional unit.



6. Write about Input unit

Computers accept coded information through input units, which read the data. Whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted over a cable to either the memory or the processor.

7. Give some example of input devices

- Joysticks
- Trackballs
- Mouses
- Microphones (Capture audio input and it is sampled & it is converted into digital codes for storage and processing).

8. What is the use of memory unit and write its types

It stores the programs and data.

There are 2 types of storage classes

- Primary
- Secondary

9. What is primary storage?

It is a fast memory that operates at electronic speeds. Programs must be stored in the memory while they are being executed. The memory contains large no of semiconductor storage cells. Each cell carries 1 bit of information. The Cells are processed in a group of fixed size called Words.

10. Write the types of memory

There are 3 types of memory.They are

- RAM(Random Access Memory)
- Cache memory
- Main Memory

11. Write about RAM

Memory in which any location can be reached in short and fixed amount of time after specifying its address is called RAM. Time required to access 1 word is called Memory Access Time

12. Write about Cache Memory

The small,fast,RAM units are called Cache. They are tightly coupled with processor to achieve high performance.

13. What is the function of ALU?

Most of the computer operations (arithmetic & logic) are performed in ALU. The data required for the operation is brought by the processor and the operation is performed by the ALU.

14. What are basic operations of a computer memory?

The basic operations of the memory are READ and WRITE.
READ – read the data from input device to memory.
WRITE – writes data to the output device.

15. Write about Output Unit

Its function is to send the processed results to the outside world. Eg.Printer Printers are capable of printing 10000 lines per minute but its speed is comparatively slower than the processor

16. Write about control unit

The information received from the input unit is either stored in the computer's memory for later reference or immediately used by the arithmetic and logic circuitry to perform the desired operations. The processing steps are determined by a program stored in the memory. Finally the results are sent back to the outside world through the output unit. All of these actions are coordinated by the control unit.

17. What is multiprocessor?

Large computer systems may contain a number of processor units, in which case they are called multiprocessor. These systems either execute a number of different application tasks in parallel or they execute subtasks of a single large task in parallel.

18. What is meant by multi computer?

Multi computer is an interconnected group of complete computers to achieve high total computation power. The term generally refers to an architecture in which each processor has its own memory rather than multiple processors with a shared memory. Something similar to parallel computing

19. Write the difference between multi computer and multiprocessor

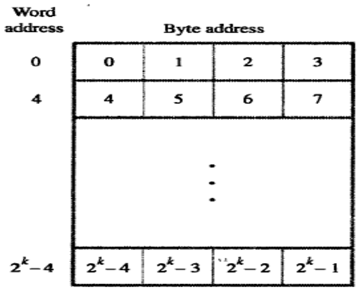
S.No	Multicomputer	Multiprocessors
1	A computer made up of several computers. similar to parallel computing	A multiprocessor system is simply a computer that has more than one CPU on its motherboard
2	Distributed computing deals with hardware and software systems containing more than one processing element, multiple programs, running under a loosely or tightly controlled regime.	Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system
3	multicomputer have one physical address space per CPU	Multiprocessors have a single physical address space (memory) shared by all the CPUs
4	It can run faster	A multiprocessor would run slower, because it would be in ONE computer.
5	A multi-computer is multiple computers, each of which can have multiple processors. Used for true parallel processing	A multi-processor is a single system with multiple CPU's

20. What is byte addressability?

There are three basic information quantities to deal with: the bit, byte, and word .a byte is always 8 bits, but the word length typically ranges from 16 to 64 bits. It is impractical to assign n distinct addresses to individual bit locations in the memory. the most practical assignment is to have successive addresses refer to successive byte locations in the memory .this the assignment used in modern computers, called byte-addressable memory. Byte locations have addresses 0,1,2…….,thus if the word length of the machine is 32 bits ,successive words are located at addresses 0,4,8,….,with each word consisting of four bytes**.**

21. What is big –endian?

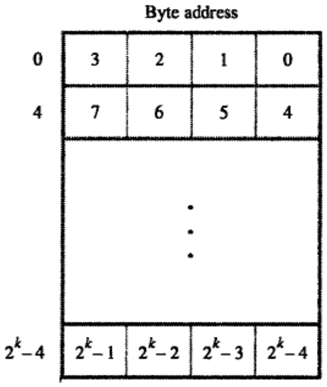
Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in computer memory. The name big-endian is used when lower byte addresses are used for the more significant bytes (the leftmost bytes) of the word.



(a) Big-endian assignment

22. What is little-endian?

The name little0endian is used for the opposite ordering, where the lower byte addresses are used for the less significant bytes (the rightmost bytes)of the word. The words “more significant” and “less significant” are used in relation to the weights (powers of 2) assigned to bits when the word represents a number.



(b) Little-endian assignment

23. What is an instruction code?

An instruction code is a group of bits that instruct the computer to perform a specific operation. It is usually divided into parts, each having its own particular interpretation. The most basic part of an instruction code is its operation part.

24. What is an operation code?

The operation code of an instruction is a group of bits that define such operations as add, subtract, multiply, shift, and complement.

The number of bits required for the operation code of an instruction depends on the total number of operations available in the computer. The operation code must consist of at least n bits for a given 2^n (or less) distinct operations

25. What is an instruction format? (Apr 13)

The format of the instruction consists of two parts. The first part specifies the operation to be performed and the second specifies an address. The memory address tells the control where to find an operand in memory. This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.

Opcode

The operation itself is usually represented by a code called the opcode (for OPeration CODE)

Operands

All the other parts of an instruction are called operands.

Addresses

Some of the operands may be the actual addresses of data in memory.

Example

ADD AX,[102]

- The opcode is ADD
- There are two operands, AX and [102]

- This instruction contains one address - 102

Instructions are often categorized by the number of operands and addresses they contain. The above is a 2 operand 1 address instruction.

26. Write about types of computer instructions

A computer must have instructions capable of performing 4 types of operations:

- Data transfers between the memory and the processor registers
- Arithmetic and logic operations on data
- Program sequencing and control
- I/O transfers

27. What is zero –address instruction?

An instruction that contains no address fields; operand sources and destination are both implicit. It may for example enable stack processing: a zero-address instruction implies that the absolute address of the operand is held in a special register that is automatically incremented (or decremented) to point to the location of the top of the stack.

Instruction format:

Opcode

Example: ADD

The above instruction consists of an operation code only. It has no address field. The operation has the effect of popping the two top numbers from the stack, adding the numbers and pushing the sum into the stack. Here all operands are performed within stack. To evaluate arithmetic expressions, they must be first converted into reverse polish notation. The operand at address X is pushed on to the top of the stack. Automatically the stack pointer is incremented.

Example:

```
X=(A + B) + (C + D)
PUSH A ; TOS <- A top of the stack
PUSH B ; T0S <- B
ADD    ;    TOS <- A + B
PUSH C ; TOS <- C
PUSH D ; TOS <- D
ADD    ;    TOS <- C + D
ADD    ;    TOS <- (C + D) + (A + B)
POP X  ; M [X] <- TOS
```

28. What is one –address instruction?

One address instructions computers needs one address field. An implied accumulator (AC) register is used for all data manipulation. Here all the operations are carried out between the accumulator register and a memory operand.

Instruction format:

Opcode	Operand

ADD X

It denotes the operation,

AC <-- AC+ M[X]

Where

AC = Accumulator register

M[X] = Memory operand at address X.

Example:

X=(A + B)x(C + D)

LOAD A ; AC <- M [A]

ADD B ; AC <- AC + M[B]

STORE T ; M [T] <- AC

LOAD C ; AC <- M [C]

ADD D ; AC <- AC + M[D]

MUL T ; AC <- ACxM[T]

STORE X ; M[X] <- AC

29. What is two –address instruction?

In this format each address field specify either a processor register or a memory word i.e., they contain two address fields.

Instruction format:

Opcode	Source	Destination

ADD R1, R2

It denotes the operation,

R1 <-- R1 + R2

Here the destination register is the same as any one of the source registers.

Mov R1, R2

It denotes the operation,

R1 <-- R2

It transfers the content of register R2 to register R1.

ADD R1, X

It denotes the operation,

R1 <-- R1 + M[X]

Where R1 = Processor register

M[X] = Memory operand at address X.

Example:

X=(A + B)-(C + D)

MOV R1,A ; R1 <- M[A]

ADD R1,B ; R1 <- R1 + M [B]

MOV R2,C ; R2 <- M [C]

ADD R2, D ; R2<-R2 + M[D]

SUB R1,R2 ; R1 <- R1 - R2

MOV X,R1 ; M[X] <- R1

30. What is three –address instruction?

Three address instructions computer needs three register address fields. The register address field may be a processor register or a memory operand. Cyber 170 computer needs three address instructions .

Instruction format:

Opcode	Source1	Source2	Destinatin

ADD R1, R2, R3

The above instruction denotes the operations,

R1 <-- R2 + R3 (arithmetic addition)

Where

R2, R3 = Source registers

R1 = Destination register

Example:

X = (A + B) * (C + D) :

```
ADD   R1, A, B       /*  R1  ¬  M[A]  +  M[B]  */
ADD   R2, C, D       /*  R2  ¬  M[C]  +  M[D]  */
MUL   X, R1, R2      /*  M[X] ¬ R1 * R2        */
```

Example of computer using this type of instructions Cyber 170.

31. Write about basic instruction types. (Apr 13)

S.No	**Basic Instruction Types**: **Instruction Type**	**Syntax**	**Example**	**Description**
1	Three Address	Operation Source1,Source2,Destination	Add A,B,C	Add values of variable A ,B & place the result into c.
2	Two Address	Operation Source,Destination	Add A,B	Add the values of A,B & place the result into B.
3	One Address	Operation Operand	Add B	Content of accumulator add with content of B.
4	Zero address	Operation	POP	Content of the top of the stack will be moved to accumulator

32. What is Addressing Mode?

It specifies a rule for interpreting or modifying the address field of the instruction (before the operand is actually referenced)

33. What is the need of Variety of Addressing Modes?

- To give programming flexibility to the user
- To use the bits in the address field of the instruction efficiently

34. What are the different types of addressing modes available?

The different types of addressing modes are:

1. Immediate addressing mode
2. Register addressing mode
3. Direct or absolute addressing mode
4. Indirect addressing mode
5. Indexed addressing mode
6. Base with index
7. Base with index and offset
8. Relative addressing mode
9. Auto increment
10. Auto decrement

35. What is implied addressing mode?

Address of the operands are specified implicitly in the definition of the instruction

- No need to specify address in the instruction
- EA = AC, or EA = Stack[SP]
- Examples from Basic Computer

CLA, CME, INP

36. What is Immediate Addressing Mode?

Instead of specifying the address of the operand, operand itself is specified

- No need to specify address in the instruction
- However, operand itself needs to be specified
- Sometimes, require more bits than the address
- Fast to acquire an operand

Operand = address field

➢ e.g. ADD 5

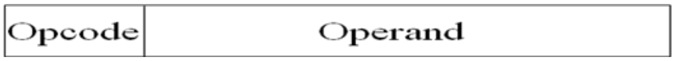
o Add 5 to contents of accumulator

o 5 is operand

➢ No memory reference to fetch data

➢ Limited range

Instruction

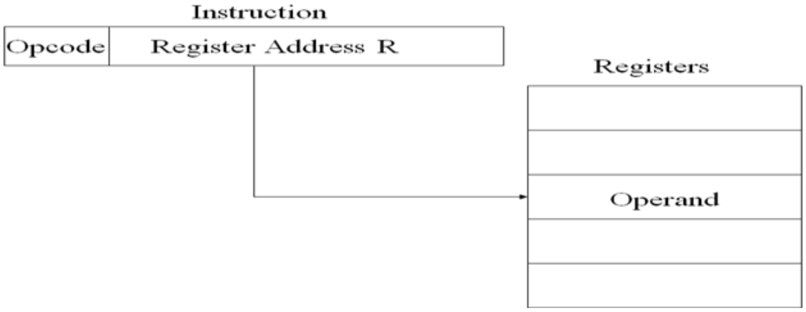


Opcode	Operand

37. What is Register Addressing Mode?

Address specified in the instruction is the register address

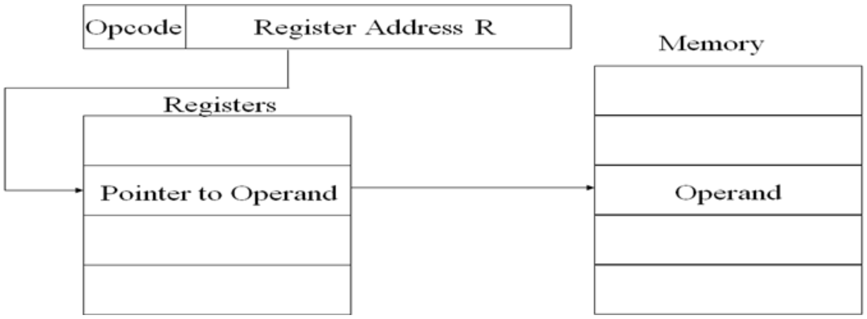
- Designated operand need to be in a register
- Shorter address than the memory address
- Saving address field in the instruction
- Faster to acquire an operand than the memory addressing
- EA = IR(R) (IR(R): Register field of IR)



38. What is Register Indirect Addressing Mode?

Instruction specifies a register which contains the memory address of the operand

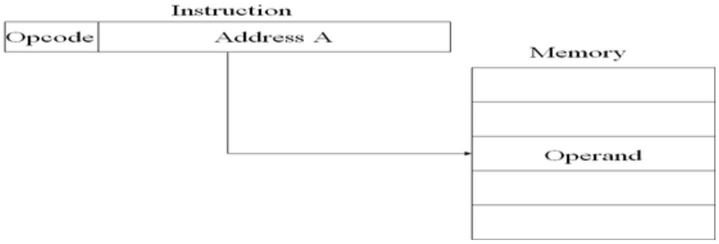
- ➢ Saving instruction bits since register address is shorter than the memory address
- ➢ Slower to acquire an operand than both the register addressing or memory addressing
- ➢ EA = [IR(R)] ([x]: Content of x)



39. What is Direct Addressing Mode?

Instruction specifies the memory address which can be used directly to access the memory

- Faster than the other memory addressing modes
- Too many bits are needed to specify the address for a large physical memory space
- EA = IR(addr) (IR(addr): address field of IR)



- ➢ **e.g. ADD A**

Add contents of cell A to accumulator

Look in memory at address A for operand

- ➢ Single memory reference to access data
- ➢ No additional calculations to work out effective address
- ➢ Limited address space

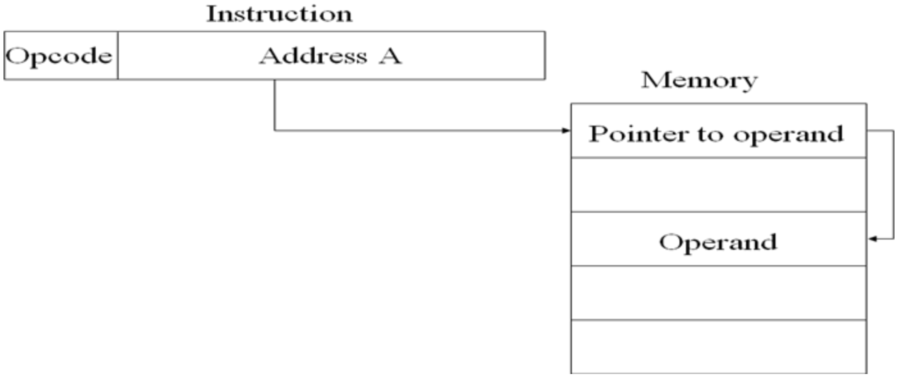
40. What is Indirect Addressing Mode?

The address field of an instruction specifies the address of a memory location that contains theaddress of the operand

- When the abbreviated address is used large physical memory can be addressed with a relatively small number of bits

- Slow to acquire an operand because of an additional memory access

- EA = M[IR(address)]



41. What is an effective address?

The effective address can be defined as address of the operand in a computation type instruction or the target address in a branch-type instruction.

42. What is a Program Counter? (Apr 13)

The program counter (PC) has 12 bits and it holds the address of the next instruction to be read from memory after the current instruction is executed. The PC goes through a counting sequence and causes the computer to read sequential instructions previously stored in memory.

To read an instruction, the content of PC is taken as the address for memory and a memory read cycle is initiated. PC is then incremented by one, so it holds the address of the next instruction in sequence.

A branch instruction calls for a transfer to a nonconsecutive instruction in the program. The address part of a branch instruction is transferred to PC to become the address of the next instruction.

43. What is an Instruction Register?

The instruction read from memory is placed in the **instruction register (IR).**

44. What is Data Register?

Data register holds the operand read from memory.

The computer needs processor registers for manipulating data.

45. What is Address Register?

Address register holds register for holding a memory address.

Memory **address register (AR)** has 12 bits since this is the width of a memory address.

46. What are the different types of Registers used for input and output?

Two registers are used for input and output.

Input register (INPR) receives an 8-bit character from an input device.

Output register (OUTR) holds an 8-bit character for an output device.

47. What is Relative Addressing Mode?

The Address fields of an instruction specifies the part of the address (abbreviated address) which can be used along with a designated register to calculate the address of the operand

- Address field of the instruction is short
- Large physical memory can be accessed with a small number of address bits
- EA = f(IR(address), R), R is sometimes implied

48. What are the different types of Relative Addressing Mode?

3 different Relative Addressing Modes depending on R

* PC Relative Addressing Mode (R = PC)

- EA = PC + IR(address)

* Indexed Addressing Mode (R = IX, where IX: Index Register)

- EA = IX + IR(address)

* Base Register Addressing Mode

(R = BAR, where BAR: Base Address Register)

- EA = BAR + IR(address)

49. What is an assembler?

Programming language processor that translates an assembly language program (the source program) to the machine language program (the object program) executable by a computer.

50. What are assembler directives?

In addition to providing a mechanism for representing instructions in a program ,the assembly language allows the programmer to specify other information needed to translate the source program into the object program. Suppose that the name SUM is used to represent the value 200 .this fact may be conveyed to the assembler program through a statement such as

SUM EQU 200

The above statement informs the assembler that the name SUM should be replaced by the value 200wherever it appears in the program, such statements, called assembler directives(or commands)

51. Write about program-controlled I/O (Nov 12)

The difference in speed between processor and I/O devices creates the need for mechanisms to synchronize the transfer of data between them. There will be several I/O devices connected to the processor; the processor checks the ``status'' input port periodically, under program control by the I/O handling procedure. If an I/O device requires service, it will signal this need by altering its input to the ``status'' port. When the I/O control program detects that this has occurred (by reading the status port) then the appropriate operation will be performed on the I/O device which requested the service.

52. List the advantages of program-controlled I/O

Program-controlled I/O has a number of advantages:

- All control is directly under the control of the program, so changes can be readily implemented.
- The order in which devices are serviced is determined by the program, this order is not necessarily fixed but can be altered by the program, as necessary. This means that the ``priority'' of a device can

be varied under program control. (The ``priority" of a determines which of a set of devices which are simultaneously ready for servicing will actually be serviced first).

- It is relatively easy to add or delete devices.

53. What is pushdown stack?

A stack is a list of data elements, usually words or bytes with the accessing restriction that elements can be added or removed at one end of the list only. This end is called the top of the stack, and the other end is called the bottom. The structure is sometimes referred to as a pushdown stack.

54. Write short note on subroutine

It is necessary to perform a particular subtask many times on different data values. Such a subtask is usually called a subroutine.

55. What it is the use of link register?

The simplest subroutine linkage method is to save the return address in a specific location, which may be a register dedicated to this function. Such a register is called the link register.

56. What is parameter passing and how it is accomplished?

The exchange of information between a calling program and a subroutine is referred to as parameter passing. It may be accomplished in several ways. The parameter may be placed in registers or memory location, where they can be accessed by the subroutine. Alternatively, the parameters may be placed on the processor stack used for saving the return address.

57. Write about shift instruction

There are many applications that require the bits of an operand to be shifted right or left some specified number of bit positions. There are two types of shifts

- ➢ Logical shift left (LshiftL)
- ➢ Logical shift right (LshiftR)
- ➢ Arithmetic shift right

58. Write about condition code register or status register.

Result of various operation for user by subsequent conditional branch instruction is accomplished by recording the required information in individual bits often called **Condition code Flags.** These flags are usually grouped together in a special processor register called the **condition code register or status register.**

59. Write some commonly used flags

- ➢ **N(Negative)**→set to 1 if the result is –ve ,otherwise cleared to 0.
- ➢ **Z(Zero)**→ set to 1 if the result is 0 ,otherwise cleared to 0.
- ➢ **V(Overflow)**→ set to 1 if arithmetic overflow occurs ,otherwise cleared to 0.
- ➢ **C(Carry)**set to 1 if carry and results from the operation ,otherwise cleared to 0

60. What is Subroutines?

In a given program, it is often necessary to perform a particular task many times on different data values. It is prudent to implement this task as a block of instructions that is executed each time the task has to be performed. Such a block of instructions is usually called a subroutine.

61. What is subroutine linkage?

The way in which a computer makes it possible to call and return from subroutines is referred to as its subroutine linkage method. The simplest subroutine linkage method is to save the return address in a specific location, which may be a register dedicated to this function. Such a register is called the link register. When the subroutine completes its task, the Return instruction returns to the calling program by branching indirectly through the link register.

62. Write about parameter passing

When calling a subroutine, a program must provide to the subroutine the parameters, that is, the operands or their addresses, to be used in the computation. Later, the subroutine returns other parameters, which are the results of the computation. This exchange of information between a calling program and a subroutine is referred to as parameter passing.

63. Write about Logical Shifts instruction

Two logical shift instructions are needed, one for shifting left (LShiftL) and another for shifting right (LShiftR). These instructions shift an operand over a number of bit positions specified in a count operand contained in the instruction. The general form of a Logicalshift-left instruction is

LShiftL R*i*, R*j*, count

which shifts the contents of register R*j* left by a number of bit positions given by the count operand, and places the result in register R*i*, without changing the contents of R*j*.

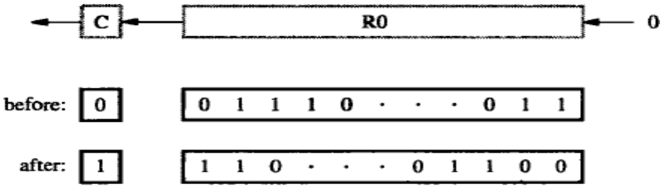
The count operand may be given as an immediate operand, or it may be contained in a processor register.

64. Write about the Logical-shift-left instruction

The instruction

LShiftL #2,R0

The above instruction shifts the contents of register R0 left by two bit positions is shown below.



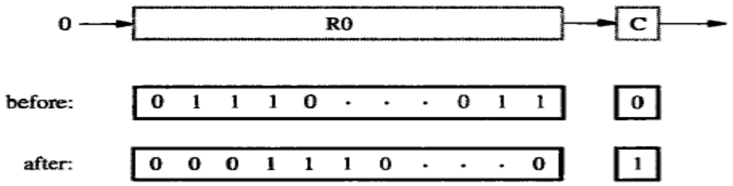
(a) Logical shift left LShiftL #2,R0

65. Write about the Logical-shift-right instruction

The instruction

LShiftR #2,R0

The above instruction shifts the contents of register R0 right by two bit positions is shown below.



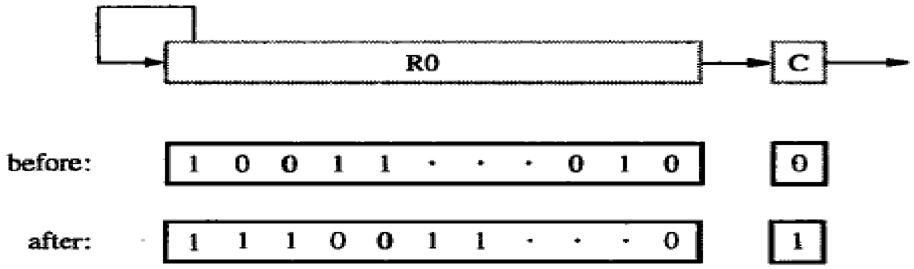
(b) Logical shift right LShiftR #2,R0

66. Write about Arithmetic Shifts instruction

In an arithmetic shift, the bit pattern being shifted is interpreted as a signed number. When a 2's-complement binary number shifted, a number one bit position to the left is equivalent to multiplying it by 2, and shifting it to the right is equivalent to dividing it by 2. Of course, overflow might occur on shifting left, and the remainder is lost when shifting right. Another important observation is that on a right shift the sign bit must be repeated as the fill-in bit for the vacated position as a requirement of the 2's-complement representation for numbers. This requirement when shifting right distinguishes arithmetic shifts from logical shifts in which the fill-in bit is always 0. Otherwise, the two types of shifts are the same. An example of an Arithmetic shift- right instruction, AShiftR, The Arithmetic-shift-left is exactly the same as the Logical-shift-left. For example ,

AShiftR #2,R0

The above instruction shifts the contents of register R0 right by two bit positions and insert's 1 in the vacated positions is shown below.



(c) Arithmetic shift right AShiftR #2,R0

67. Write about Rotate instruction

In the shift operations, the bits shifted out of the operand are lost, except for the last bit shifted out which is retained in the Carry flag C. For situations where it is desirable to preserve all of the bits, rotate instructions may be used instead. These are instructions that move the bits shifted out of one end of the operand into the other end. Two versions of both the Rotate-left and Rotate-right instructions are often provided.

In one version, the bits of the operand are simply rotated. In the other version, the rotation includes the C flag. When the C flag is not included in the rotation, it still retains the last bit shifted out of the end of the register.

68. List out the various rotate instruction

In addition to the shift instructions, there are also four rotate instructions:

- RotateL (Rotate left without the carry flag CF)
- RotateR (Rotate right without the carry flag CF)

- RotateLC (Rotate left including the carry flag CF)
- RotateRC (Rotate right including the carry flag CF)

The OP codes RotateL, RotateLC, RotateR, and RotateRC, denote the instructions that perform the rotate operations.

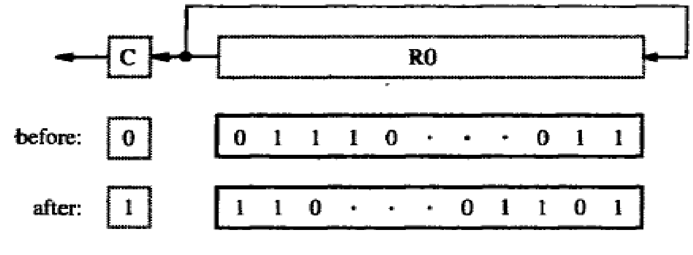
69. Write about RotateL instruction

This instruction rotates the content of the specified register on left by the specified number of bit position which is specified in the count variable.

For example,the instruction

RotateL #2,R0

Rotates the content of R0 to left by 2 bit positions but not through the carry is shown below.



(a) Rotate left without carry RotateL #2,R0

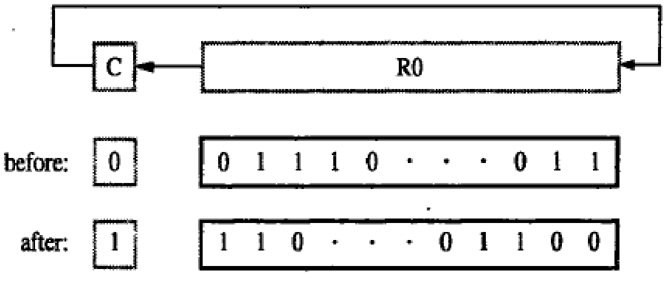
70. Write about RotateLC instruction

This instruction rotates the content of the specified register on left by the specified number of bit position through carry; the number of shift position will be specified in the count variable.

For example,the instruction

RotateLC #2,R0

Rotates the content of R0 to left by 2 bit positions through the carry is shown below.



(b) Rotate left with carry RotateLC #2,R0

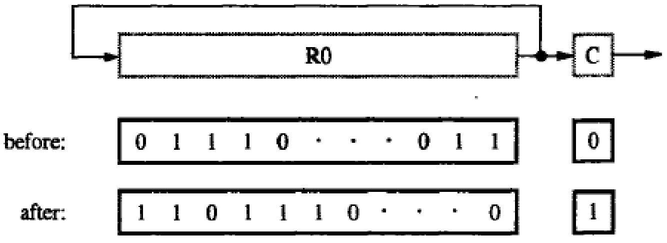
71. Write about RotateR instruction

This instruction rotates the content of the specified register on right by the specified number of bit position which is specified in the count variable.

For example,the instruction

RotateR #2,R0

Rotates the content of R0 to right by 2 bit positions but not through the carry is shown below.



(c) Rotate right without carry RotateR #2,R0

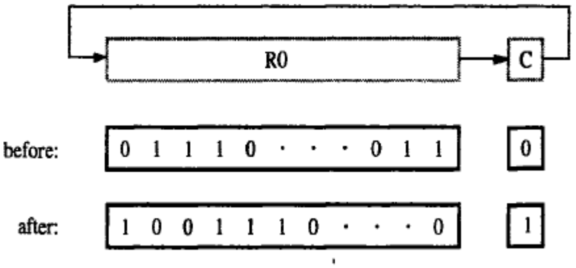
72. Write about RotateRC instruction

This instruction rotates the content of the specified register on right by the specified number of bit position through carry, the number of shift position will be specified in the count variable.

For example, the instruction

RotateRC #2,R0

Rotates the content of R0 to right by 2 bit a position through the carry is shown below.

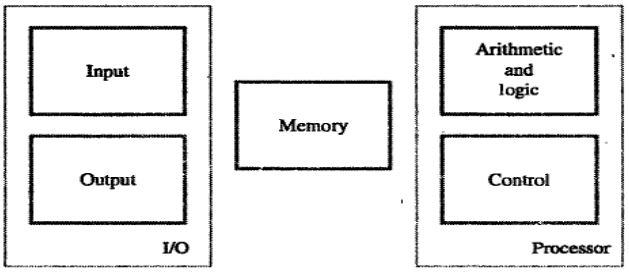


(d) Rotate right with carry RotateRC #2,R0

11 Marks

1. Write in detail about Functional units of a computer.

Functional units of a Computer



A computer consists of 5 main parts.

- Input
- Memory
- Arithmetic and logic
- Output
- Control Units

Input unit accepts coded information from human operators, from electromechanical devices such as keyboards, or from other computers over digital communication lines. The information received is either stored in the computer’s memory for later reference or immediately used by the arithmetic and logic circuitry to perform the desired operations.

The processing steps are determined by a program stored in the memory. Finally the results are sent back to the outside world through the output unit. All of these actions are coordinated by the control unit. The list of instructions that performs a task is called a program. Usually the program is stored in the memory.

The processor then fetches the instruction that makes up the program from the memory one after another and performs the desire operations.

Input Unit:

- Computers accept coded information through input units, which read the data.
- Whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted over a cable to either the memory or the processor.
- Some input devices are
  - ✓ Joysticks
  - ✓ Trackballs
  - ✓ Mouses
  - ✓ Microphones (Capture audio input and it is sampled & it is converted into digital codes for storage and processing).

Memory Unit:

It stores the programs and data.
There are 2 types of storage classes

- ✓ Primary
- ✓ Secondary

Primary Storage:

- It is a fast memory that operates at electronic speeds.
- Programs must be stored in the memory while they are being executed.
- The memory contains large no of semiconductor storage cells.
- Each cell carries one bit of information.
- The cells are processed in a group of fixed size called words. To provide easy access to any word in a memory, a distinct address is associated with each word location.
- Addresses are numbers that identify successive locations.
- The number of bits in each word is called the word length.
- The word length ranges from 16 to 64 bits.
- There are 3 types of memory. They are
  - Random Access Memory
  - Cache memory
  - Main Memory

RAM:

- Memory in which any location can be reached in short and fixed amount of time after specifying its address is called RAM. Time required to access 1 word is called Memory Access Time.

Cache Memory:

- The small,fast,RAM units are called Cache. They are tightly coupled with processor to achieve high performance.

Main Memory:

- The largest and the slowest unit is called the main memory.

ALU:

- Most computer operations are executed in ALU.
- Consider a example, Suppose 2 numbers located in memory are to be added. They are brought into the processor and the actual addition is carried out by the ALU. The sum may then be stored in the memory or retained in the processor for immediate use. Access time to registers is faster than access time to the fastest cache unit in memory.

Output Unit:

- Its function is to send the processed results to the outside world. eg.Printer Printers are capable of printing 10000 lines per minute but its speed is comparatively slower than the processor.

Control Unit:

- The operations of Input unit, output unit, ALU are co-ordinate by the control unit. The control unit is the Nerve centre that sends control signals to other units and senses their states. Data transfers between the processor and the memory are also controlled by the control unit through timing signals.
- The operation of computers are,
  - ✓ The computer accepts information in the form of programs and data through an input unit and stores it in the memory
  - ✓ Information stored in the memory is fetched, under program control into an arithmetic and logic unit, where it is processed.
  - ✓ Processed information leaves the computer through an output unit.
  - ✓ All activities inside the machine are directed by the control unit.

2. Write short notes on multi processor and multi computer

Multi computer:

- Multi computer is an interconnected group of complete computers to achieve high total computation power.. The term generally refers to an architecture in which each processor has its own memory rather than multiple processors with a shared memory. Something similar to parallel computing.
- Distributed computing deals with hardware and software systems containing more than one processing element or storage element, concurrent processes, or multiple programs, running under a loosely or tightly controlled regime.
- A multicomputer may be considered to be either a loosely coupled NUMA computer or a tightly coupled cluster. Multicomputer is commonly used when strong computer power is required in an environment with restricted physical space or electrical power.
- Common suppliers include Mercury Computer Systems, CSPI, and SKY Computers. Common uses include 3D medical imaging devices and mobile radar.

- In distributed computing a program is split up into parts that run simultaneously on multiple computers communicating over a network. Distributed computing is a form of parallel computing, but parallel computing is most commonly used to describe program parts running simultaneously on multiple processors in the same computer.
- Both types of processing require dividing a program into parts that can run simultaneously, but distributed programs often must deal with heterogeneous environments, network links of varying latencies, and unpredictable failures in the network or the computers

Multi processor:

- Large computer systems may contain a number of processor units, in which case they are called multiprocessor. These systems either execute a number of different application tasks in parallel or they execute subtasks of a single large task in parallel
- Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them. There are many variations on this basic theme, and the definition of multiprocessing can vary with context, mostly as a function of how CPUs are defined (multiple cores on one die, multiple chips in one package, multiple packages in one system unit, etc.).

3. Write in detail about memory location and addresses

- Number and character operands, as well as instructions are stored in the memory of a computer. The memory consist of many millions of storage cells, each of which can store a bit of information having the value 0 or 1.because a single bit represents a very small amount of information, bits are seldom handled individually. The usual approach is to deal with them in groups of fixed size.
- The memory is organized so that a group of n bits can be stored or retrieved in a single basic operation. Each group of n bits is referred to as a word of information, and n is called the word length. The memory of a computer can be represented as a collection of words shown below.

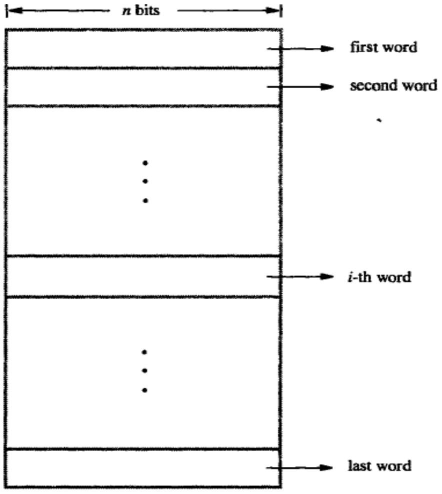


Figure 2.5 Memory words.

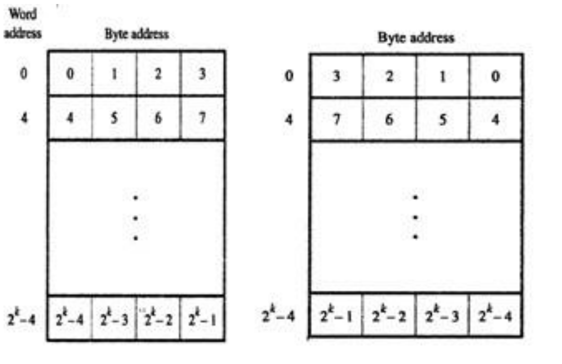
- Accessing the memory to store or retrieve a single item of information ,either a word or a byte, requires distinct names or addresses for each item location.

Byte addressability

There are three basic information quantities to deal with: the bit, byte, and word .a byte is always 8 bits, but the word length typically ranges from 16 to 64 bits. It is impractical to assign n distinct addresses to individual bit locations in the memory. the most practical assignment is to have successive addresses refer to successive byte locations in the memory .this the assignment used in modern computers, called byte-addressable memory. Byte locations have addresses 0,1,2…….,thus if the word length of the machine is 32 bits ,successive words are located at addresses 0,4,8,….,with each word consisting of four bytes**.**

Big –endian and Little-endian assignments

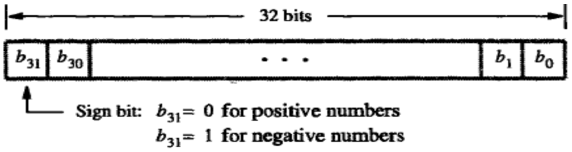
- Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in computer memory. The name big-endian is used when lower byte addresses are used for the more significant bytes (the leftmost bytes)of the word.
- The name little0endian is used for the opposite ordering, where the lower byte addresses are used for the less significant bytes (the rightmost bytes)of the word. The words “more significant” and “less significant” are used in relation to the weights (powers of 2) assigned to bits when the word represents a number. Both little-endian and big-endian assignments are used in commercial machines.
- In both cases ,byte addresses 0,4,8,….are taken as the addresses of successive words in the memory and are the addresses used when specifying memory read and write operations for words.



(a) Big-endian assignment

(b) Little-endian assignment

- In addition to specifying the address ordering of bytes within a word, it is also necessary to specify the labeling of bits within a byte or a word. The most common convention is shown below


32 bits
b_{31}
b_{30}
. . .
b_1
b_0
Sign bit: $b_{31}=$ 0 for positive numbers
$b_{31}=$ 1 for negative numbers

(a) A signed integer

- It is the most natural ordering for the encoding of numerical data. The same ordering is also used for labeling bits within a byte, that is b_7,b_6,…..b_0,from left to right.

Word alignment

In the case of a 32-bit word length, natural word boundaries occur at addresses 0,4,8…..the word locations have aligned addresses. In general, words are said to be aligned in memory if they begin at a byte address that is multiple of the number of bytes in a word. The number of bytes in a word is power of 2.hence,if the word length is 16(2 bytes),aligned words begin at byte addresses 0,2,4,…..and for a word length of 64(23 bytes),aligned words begin at byte addresses 0,8,16,….

Accessing numbers, characters, and character strings

A number usually occupies one word. It can be accessed in the memory by specifying its word address. Similarly, individual characters can be accessed by their byte address. In many applications, it is necessary to handle character strings of variable length .the beginning of the string is indicated by giving the address of the byte containing its first character. Successive byte locations contain successive characters of the string.

There are two ways to indicate the length of the string .a special control character with the meaning “end of the string “cam be used as the last character in the string, or a sequence memory word location or processor register can contain a number indicating the length of the string in bytes.

4. Write short notes on memory operations

The general- purpose computers use a set of instructions called a program to process data. A computer executes the program to create output data from input data. Both program instructions and data operands are stored in memory. Two basic operations requires in memory access Load operation (Read or Fetch)- Contents of specified memory location are read by processor. Store operation (Write)- Data from the processor is stored in specified memory location.

The load operation transfers a copy of the contents of a specific memory location to the processor. The memory contents remain unchanged. To start a load operation the processor sends the address of the desired location to the memory and requests that its contents be read. The memory reads the data stored at this address and sends them to the processor.

The store operation transfers an item of information from the processor to a specific memory location, destroying the former contents of that location. The processor sends the address of the desired location to the memory, together with the data to be written into that location.

5. Explain instructions and instruction sequencing

A computer must have instruction capable of performing the following operations. They are,

- Data transfer between memory and processor register.
- Arithmetic and logical operations on data.
- Program sequencing and control.
- I/O transfer.

Register Transfer Notation:

The possible locations in which transfer of information occurs are,

- Memory Location
- Processor register
- Registers in I/O sub-system.

Location	**Hardware Binary Address**	**Example**	**Description**
Memory	LOC,PLACE,A,VAR2	R1 ← [LOC]	The contents of memory location are transferred to. the processor register
Processor	R0,R1,….	[R3]←[R1]+[R2]	Add the contents of register R1 &R2 and places .their sum into register R3.It is .called Register Transfer Notation.
I/O registers	DATAIN,DATAOUT		Provides Status information

Assembly Language Notation:

It is another type of notation to represent machine instructions and programs.

Assembly Language Format	**Description**
Move LOC,R1	Transfers the contents of memory location to the processor register R1.
Add R1,R2,R3	Add the contents of register R1 & R2 and places their sum into register R3

Basic Instruction Types:

The operation of adding two numbers os a fundamental capability in any computer. The statement C=A+B

In a high level language program is a command to the computer to add the current values of the two variables called A and B,and to assign the sum to a third variable C.when the program containing this statement is compiled ,the three variables,A,B,C,are assigned to distinct location in the memory. The contents of these locations represents the values of the 3 variables. Hence ,the above high –level language statement requires the action

$$C \leftarrow [A] + [B]$$

The following table shows the basic instruction types:

Instruction Type	**Syntax**	**Example**	**Description**
Three Address	Operation Source1,Source2,Destination	Add A,B,C	Add values of variable A ,B & place the result into c.
Two Address	Operation Source,Destination	Add A,B	Add the values of A,B & place the result into B.
One Address	Operation Operand	Add B	Content of accumulator add with content of B.

Instruction Execution and Straight–line Sequencing:

Instruction Execution:

There are 2 phases for Instruction Execution. They are,

- Instruction Fetch
- Instruction Execution

Instruction Fetch:

The instruction is fetched from the memory location whose address is in PC.This is placed in IR.

Instruction Execution:

Instruction in IR is examined to determine whose operation is to be performed.

Program execution Steps:

- To begin executing a program, the address of first instruction must be placed in PC.
- The processor control circuits use the information in the PC to fetch & execute instructions one at a time in the order of increasing order.
- This is called Straight line sequencing. During the execution of each instruction, the PC is incremented by 4 to point the address of next instruction. Thus arte the move instruction at location I + 8 is executed ,the PC contains the value i + 12 ,which is the address of the first instruction of the next program segment.

- Executing a given instruction is a two-phase procedure. In the first phase, called instruction fetch, the instruction is fetched from the memory location whose address is in the PC.this instruction is placed in the instruction register (IR)in the processor.
- At the start of second phase ,called instruction execute, the instruction in IR is examined to determine which operation is to be performed.
- The specified operation is the performed by the processor.

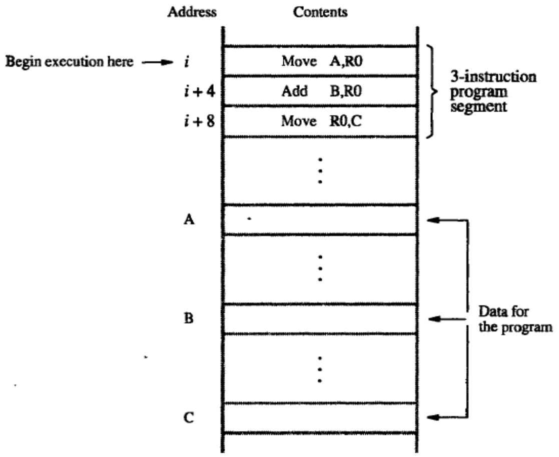


Fig. Program for C←[A]+[B]

Branching:

- Consider a task of adding a list of n numbers.The Address of the memory locations containing the n numbers are symbolically given as NUM$_1$,NUM$_2$…..NUM$_n$ and a separate Add instruction is used to add each number to the contents of register R0. After all the numbers have been added,the result is placed in memory location SUM.

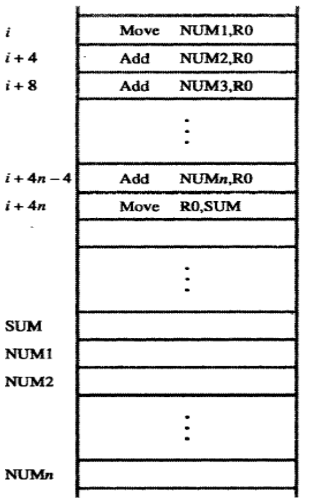


Fig. Straight line program for adding n numbers

Using loop to add 'n' numbers:

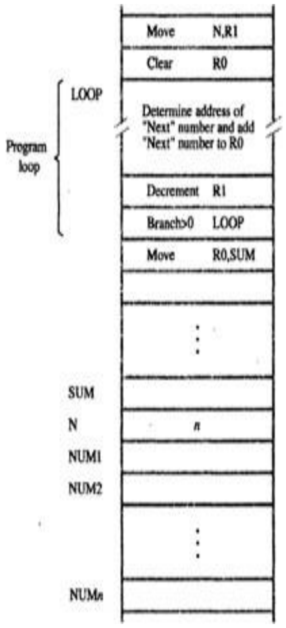
- Number of entries in the list „n‟ is stored in memory location M.Register R1 is used as a counter to determine the number of times the loop is executed.
- Content locations M are loaded into register R1 at the beginning of the program.
- It starts at location Loop and ends at the instruction. Branch>0.During each pass, the address of the next list entry is determined and the entry is fetched and added to R0.

Decrement R1

- It reduces the contents of R1 by 1 each time through the loop.

Brach > 0 Loop

- A conditional branch instruction causes a branch only if a specified condition is satisfied.



Conditional Codes:

- Result of various operation for user by subsequent conditional branch instruction is accomplished by recording the required information in individual bits often called **Condition code Flags.**
- These flags are usually grouped together in a special processor register called the **condition code register or status register.**
- Individual condition code flags are set to 1 or cleared to 0,depending on the outcome of the operation performed.

Commonly used flags:

- **N(Negative)**→set to 1 if the result is –ve ,otherwise cleared to 0.
- **Z(Zero)**→ set to 1 if the result is 0 ,otherwise cleared to 0.
- **V(Overflow)**→ set to 1 if arithmetic overflow occurs ,otherwise cleared to 0.
- **C(Carry)**set to 1 if carry and results from the operation ,otherwise cleared to 0

The N and Z flags indicate whether the result of arithmetic of logic operation is negative or zero. The N and Z flags may also be affected by instructions that transfer data, such as Move, Load or Store. This makes it possible for a later conditional branch instruction to cause a branch based on the sign and value of the operand that was moved.

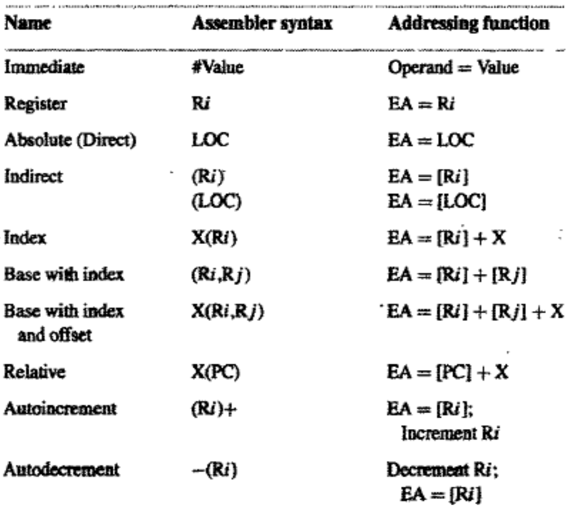
6. Write in detail about addressing modes (Apr 13)

The different ways in which the location of an operand is specified in an instruction is called as Addressing mode.

Generic Addressing Modes are :

- Immediate mode
- Register mode
- Absolute mode

- Indirect mode
- Index mode
- Base with index
- Base with index and offset
- Relative mode
- Auto-increment mode
- Auto-decrement mode



Name	Assembler syntax	Addressing function
Immediate	#Value	Operand = Value
Register	R*i*	EA = R*i*
Absolute (Direct)	LOC	EA = LOC

Indirect	(R*i*)
	(LOC)
		EA = [R*i*]
		EA = [LOC]

Index	X(R*i*)	EA = [R*i*] + X
Base with index	(R*i*,R*j*)	EA = [R*i*] + [R*j*]
Base with index and offset	X(R*i*,R*j*)	EA = [R*i*] + [R*j*] + X
Relative	X(PC)	EA = [PC] + X
Autoincrement	(R*i*)+	EA = [R*i*]; Increment R*i*
Autodecrement	−(R*i*)	Decrement R*i*; EA = [R*i*]

Implementation of Variables and Constants:

Variables:

In assembly language, a variable is represented by allocating a register or a memory location to hold its value. The value can be changed as needed using the appropriate instructions. There are 2 accessing modes to access the variables. They are

- Register Mode
- Absolute Mode

Register Mode:

The operand is the contents of the processor register. The name(address) of the register is given in the instruction.

Absolute Mode (Direct Mode):

The operand is in memory location. The address of this location is given explicitly in the instruction. In some assembly languages, this mode is called Direct.

Eg: MOVE LOC,R2

The above instruction uses the register and absolute mode. The processor register is the temporary storage where the data in the register are accessed using register mode. The absolute mode can represent global variables in the program.

Mode	**Assembler Syntax**	**Addressing Function**
Register mode	Ri	EA=Ri
Absolute mode	LOC	EA=LOC

Where **EA**-Effective Address

Constants:

Address and data constants can be represented in assembly language using Immediate Mode.

Immediate Mode.

The operand is given explicitly in the instruction.

Eg: Move 200 $_{\text{immediate}}$, R0

It places the value 200 in the register R0.The immediate mode used to specify the value of source operand. In assembly language, the immediate subscript is not appropriate so # symbol is used. It can be re-written as Move #200,R0

Assembly Syntax:	**Addressing Function**
Immediate #value	Operand =value

Indirection and Pointers:

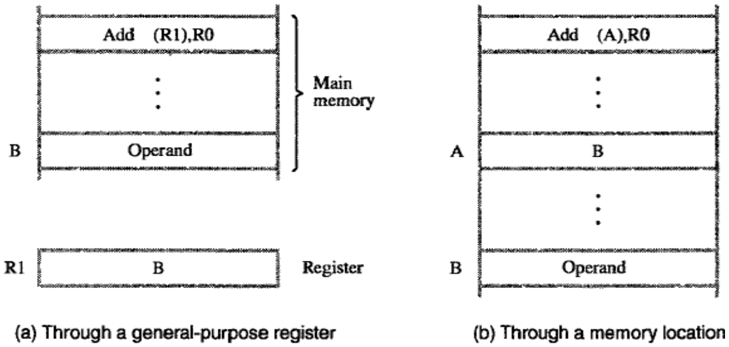
Instruction does not give the operand or its address explicitly. Instead it provides information from which the new address of the operand can be determined. This address is called effective Address(EA) of the operand

Indirect Mode:

The effective address of the operand is the contents of a register . We denote the indirection by the name of the register or new address given in the instruction. Address of an operand(B) is stored into R1 register. If we want this operand, we can get it through register R1(indirection). The register or new location that contains the address of an operand is called the **pointer.**

Mode	**Assembler syntax**	**Addressing function**
Indirect	Ri , LOC	EA=[Ri] or EA=[LOC]

Example:



(a) Through a general-purpose register

(b) Through a memory location

Figure 2.11 Indirect addressing.

To execute the Add instruction fig-a ,the processor uses the value B which is in register R1,as the EA of the operand .it requests a read operation from the memory to read the contents of the location B.the value read is the desired operand, which the processor adds to the contents of register R0.indirect addressing through a memory location is also possible as shown in fig-b. in this case ,the processor first reads the contents of memory location A, then requests a second read operation using the value B as an address to obtain the operand. The register or memory location that contains the address of an operand is called a pointer.

Indexing and Arrays:

Index Mode:

The effective address of an operand is generated by adding a constant value to the contents of a register.

The constant value uses either special purpose or general purpose register. The index mode is indicated symbolically as,

$$\mathbf{X(R_i)}$$

Where

X – denotes the constant value contained in the instruction

R$_i$ – It is the name of the register involved.

The Effective Address of the operand is,

EA=X + [Ri]

The index register R1 contains the address of a new location and the value of X defines an offset(also called a displacement).

To find operand,

First go to Reg R1 (using address)-read the content from R1-1000

Add the content 1000 with offset 20 get the result.

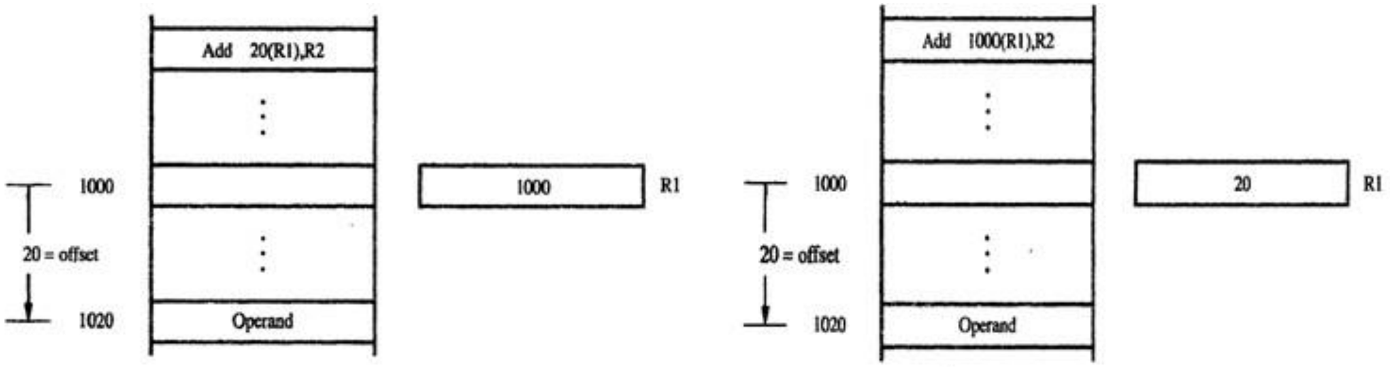
1000+20=1020

Here the constant X refers to the new address and the contents of index register define the offset to the operand. The sum of two values is given explicitly in the instruction and the other is stored in register.

Eg: Add 20(R1) , R2 (or) EA=>1000+20=1020

Index Mode	**Assembler Syntax**	**Addressing Function**
Index	X(Ri)	EA=[Ri]+X
Base with Index	(Ri,Rj)	EA=[Ri]+[Rj]
Base with Index and offset	X(Ri,Rj)	EA=[Ri]+[Rj] +X

Example



(a) Offset is given as a constant

(b) Offset is in the index register

The above figure illustrates two ways of using the index mode .in fig –a ,the index register R1.contains the address of a memory location ,and the value X defined an offset(also called a displacement)from this address to the location where the operand is found.

An alternative in fig-b illustrates that the constant X corresponds a memory address, and the contents of the index register define the offset to the operand. In either case, the effective address is the sum of two values one is given explicitly in the instruction, and the other is stored in a register.

Relative Addressing:

It is same as index mode. The difference is, instead of general purpose register, here we can use program counter(PC).

Relative Mode:

The Effective Address is determined by the Index mode using the PC in place of the general purpose register This mode can be used to access the data operand. But its most common use is to specify the target address in branch instruction.Eg. Branch>0 Loop It causes the program execution to goto the branch target location. It is identified by the name loop if the branch condition is satisfied.

Mode	Assembler syntax	Addressing function
Relative	X(PC)	EA=[PC]+X

Additional Modes:

There are two additional modes. They are

- Auto-increment mode
- Auto-decrement mode

These two modes are useful for accessing data items in successive locations in the memory.

Auto-increment mode:

The Effective Address of the operand is the contents of a register in the instruction. After accessing the operand, the contents of this register is automatically incremented to point to the next item in the list.

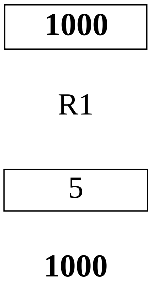
The auto increment mode is denoted by putting the specified register in parentheses to show that the contents of the register are used as the effective address ,followed by a plus sign to indicate that these contents are to be incremented after the operand is accessed. Thus the autoincrement mode is written as shown below

Mode	Assembler syntax	Addressing function
Auto-increment	(Ri)+	EA=[Ri]; Increment Ri

Example

Assume the content of R1 is 1000 and the location 1000 contains the value 5 ,which is nothing but the operand. Then the instruction

(R1)+



1000

R1

5

1000

After accessing the location 1000, the content of R1 will be incremented by 1 for byte –sixed operands,2 for 16-bit operands ,and 4 for 32-bit operands. In this case suppose R1 is pointing integer value ,then R1 is incremented by 2.then R1 contains 1002.

Auto-decrement mode:

The Effective Address of the operand is the contents of a register in the instruction. After accessing the operand, the contents of this register is automatically decremented to point to the next item in the list.

The auto decrement mode is denoted by putting the specified register in parentheses ,preceded by a minus sign to indicate that the contents of the register are to be decremented before being used as the effective address. Thus the autodecrement mode is written as shown below

Mode	Assembler syntax	Addressing function
Auto-decrement	-(Ri)	EA=[Ri]; Decrement Ri

Example

Assume the content of R1 is 1002 and the location 1002 contains the value 15 ,which is nothing but the operand. Then the instruction

-(R1)

1002

R1

15

1000

Before accessing the operand, the content of R1 will be decremented by 1 for byte –sixed operands,2 for 16-bit operands ,and 4 for 32-bit operands. In this case suppose R1 is pointing integer value ,then R1 is decremented by 2.then R1 contains 1000,then this instruction access the value 15.

7. Write short notes assembly language and assembler directives

Assembly language:

Machine instructions are represented by patterns 0's and 1's.such patterns are awkward to deal with then preparing programs. Therefore symbolic names are used to represent the patterns, such as Move, Add, Increment and Branch .when writing programs for a specific computer, such words are replaced by acronym called mnemonics .such as MOV,ADD,INC ,and BR.A completes set of such symbolic names and rules for their use constitute a programming language, generally referred to as an assembly language.

Example instruction:

ADD #5,R3

Adds 5 to the contents of register R3 and puts the result back into register R3.

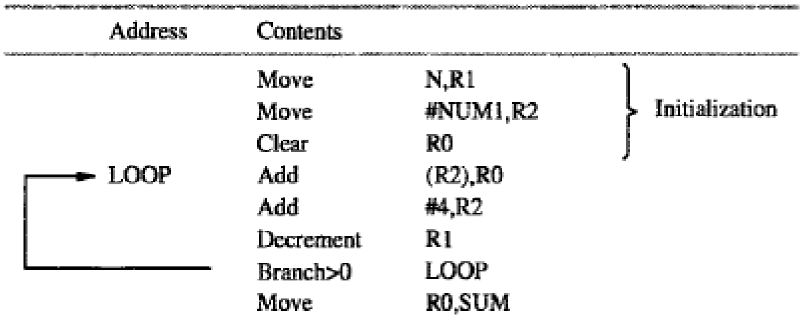
Assembler directives:

In addition to providing a mechanism for representing instructions in a program, the assembly language allows the programmer to specify other information needed to translate the source program into the object program. Suppose that the name SUM is used to represent the value 200 .this fact may be conveyed to the assembler program through a statement such as

SUM EQU 200

The above statement informs the assembler that the name SUM should be replaced by the value 200wherever it appears in the program, such statements, called assembler directives(or commands)

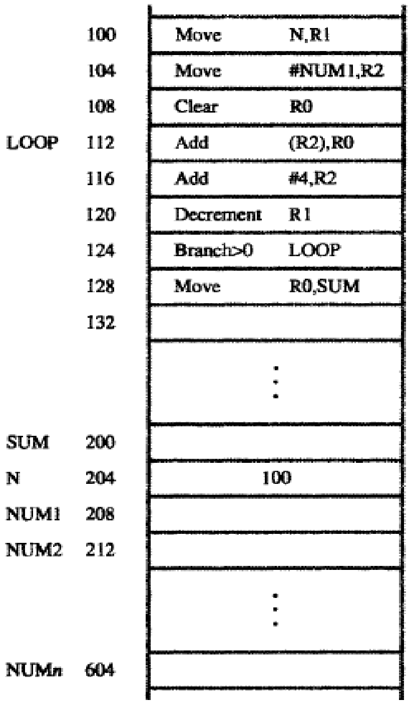
Example



```
Address      Contents

             Move          N,R1         }
             Move          #NUM1,R2     } Initialization
             Clear         R0           }
LOOP         Add           (R2),R0
             Add           #4,R2
             Decrement     R1
             Branch>0      LOOP
             Move          R0,SUM
```

In order to run this program on a computer, it is necessary to write its source code in the required assembly language ,specifying all the information needed to generate the corresponding object program. And then the object program is loaded into main memory shown below.



	100	Move N,R1
	104	Move #NUM1,R2
	108	Clear R0
LOOP	112	Add (R2),R0
	116	Add #4,R2
	120	Decrement R1
	124	Branch>0 LOOP
	128	Move R0,SUM
	132	
		⋮
SUM	200	
N	204	100
NUM1	208	
NUM2	212	
		⋮
NUM*n*	604	

The below figure shows the assembly language representation for the above program. The assembler directive EQU ,informs the assembler about the value of SUM. The second assembler directive ORIGIN ,tells the assembler program where in the memory to place the data block that follows. In this case, the location specified has the address 204 Since this location is to be loaded with the value 100 ,a DATAWORD directive is used to inform the assembler of this requirement. It states that the data value 100 is to be placed in the memory word at address 204.

Any statement that results in instructions or data being placed in a memory location may be given a memory address label.the label is assigned a value equal to the address of that location .because the DATAWORD statement is given the label N .the name N is assigned the value 204.whenever N is encountered in the rest of the program ,it is replace with the value 204.using N as a label in this manner is equivalent to using the assembler directive

N EQU 204

The RESERVE assembler directive declares that a memory block of 400 bytes is to be reserved for data, and that the name NUM1 is to be associated with address 208.the second ORIGIN directive specifies that the instructions of the object program are to be loaded in the memory starting at address 100.the END directive ,which tells the assembler that this is the end of the source program text.

8. Explain with example about Shift and Rotate Instructions

There are many applications that require the bits of an operand to be shifted right or left some specified number of bit positions. The details of how the shifts are performed depend on whether the operand is a signed number or some more general binary-coded information. For general operands, we use a logical shift. For a signed number, we use an arithmetic shift, which preserves the sign of the number.

Logical Shifts

Two logical shift instructions are needed, one for shifting left (LShiftL) and another for shifting right (LShiftR). These instructions shift an operand over a number of bit positions specified in a count operand contained in the instruction. The general form of a Logicalshift-left instruction is

LShiftL R*i*, R*j*, count

which shifts the contents of register R*j* left by a number of bit positions given by the count operand, and places the result in register R*i*, without changing the contents of R*j*.

The count operand may be given as an immediate operand, or it may be contained in a processor register. Need to specify the bit values brought into the vacated positions at the right end of the destination operand, and to determine what happens to the bits shifted out of the left end. Vacated positions are filled with zeros.

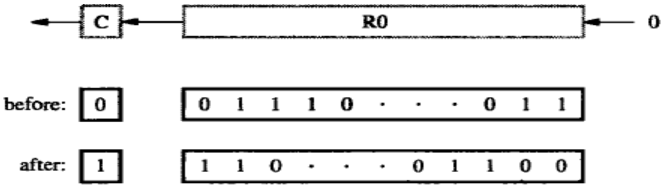
In computers that do not use condition code flags, the bits shifted out are simply dropped. In computers that use condition code flags, these bits are passed through the Carry flag, C, and then dropped. Involving the C flag in shifts is useful in performing arithmetic operations on large numbers that occupy more than one word.

The Logical-shift-left:

The instruction

LShiftL #2,R0

The above instruction shifts the contents of register R0 left by two bit positions is shown below.



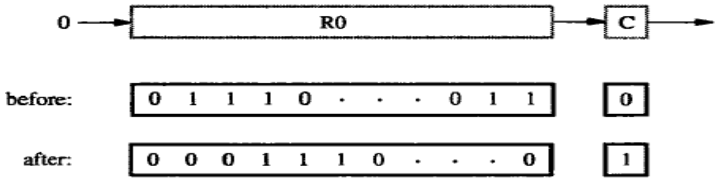
(a) Logical shift left LShiftL #2,R0

The Logical-shift-right:

The instruction

LShiftR #2,R0

The above instruction shifts the contents of register R0 right by two bit positions is shown below.



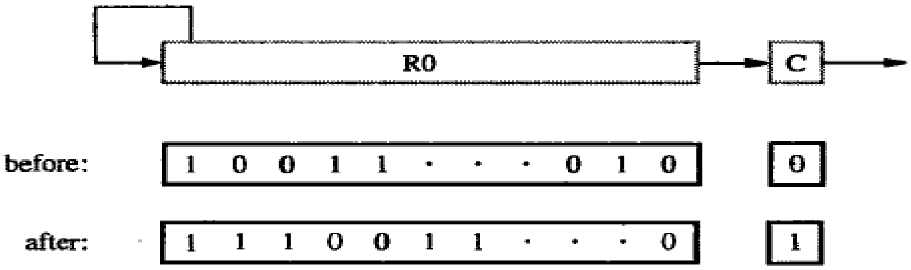
(b) Logical shift right LShiftR #2,R0

Arithmetic Shifts:

In an arithmetic shift, the bit pattern being shifted is interpreted as a signed number. When a 2's-complement binary number shifted, a number one bit position to the left is equivalent to multiplying it by 2, and shifting it to the right is equivalent to dividing it by 2. Of course, overflow might occur on shifting left, and the remainder is lost when shifting right. Another important observation is that on a right shift the sign bit must be repeated as the fill-in bit for the vacated position as a requirement of the 2's-complement representation for numbers. This requirement when shifting right distinguishes arithmetic shifts from logical shifts in which the fill-in bit is always 0. Otherwise, the two types of shifts are the same. An example of an Arithmeticshift- right instruction, AShiftR, The Arithmetic-shift-left is exactly the same as the Logical-shift-left.For example ,

AShiftR #2,R0

The above instruction shifts the contents of register R0 right by two bit positions and insert's 1 in the vacated positions is shown below.



(c) Arithmetic shift right AShiftR #2,R0

Rotate instruction

In the shift operations, the bits shifted out of the operand are lost, except for the last bit shifted out which is retained in the Carry flag C. For situations where it is desirable to preserve all of the bits, rotate instructions may be used instead. These are instructions that move the bits shifted out of one end of the operand into the other end. Two versions of both the Rotate-left and Rotate-right instructions are often provided.

In one version, the bits of the operand are simply rotated. In the other version, the rotation includes the C flag. When the C flag is not included in the rotation, it still retains the last bit shifted out of the end of the register.

In addition to the shift instructions, there are also four rotate instructions:

- RotateL (Rotate left without the carry flag CF)
- RotateR (Rotate right without the carry flag CF)
- RotateLC (Rotate left including the carry flag CF)
- RotateRC (Rotate right including the carry flag CF)

The OP codes RotateL, RotateLC, RotateR, and RotateRC, denote the instructions that perform the rotate operations.

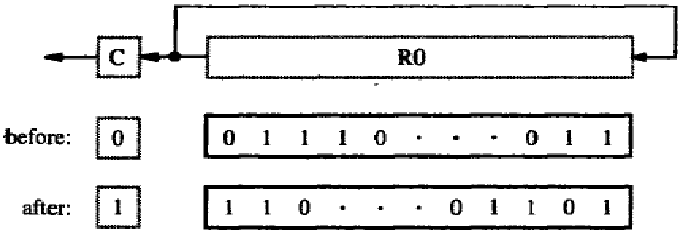
RotateL (Rotate left without the carry flag CF)

This instruction rotate the content of the specified register on left by the specified number of bit position which is specified in the count variable.

For example, the instruction

RotateL #2,R0

Rotates the content of R0 to left by 2 bit positions but not through the carry is shown below.



(a) Rotate left without carry RotateL #2,R0

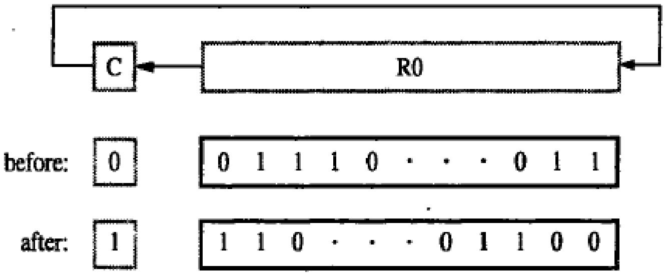
RotateLC (Rotate left including the carry flag CF)

This instruction rotate the content of the specified register on left by the specified number of bit position through carry, the number of shift position will be specified in the count variable.

For example, the instruction

RotateLC #2,R0

Rotates the content of R0 to left by 2 bit positions through the carry is shown below.



(b) Rotate left with carry RotateLC #2,R0

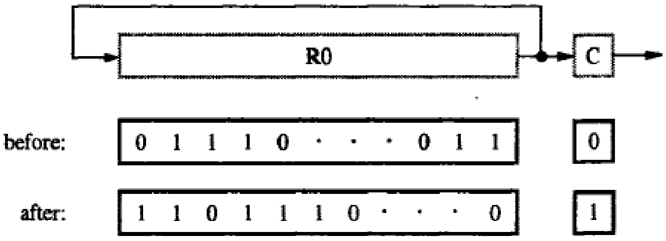
RotateR (Rotate right without the carry flag CF)

This instruction rotate the content of the specified register on right by the specified number of bit position which is specified in the count variable.

For example, the instruction

RotateR #2,R0

Rotates the content of R0 to right by 2 bit positions but not through the carry is shown below.



(c) Rotate right without carry RotateR #2,R0

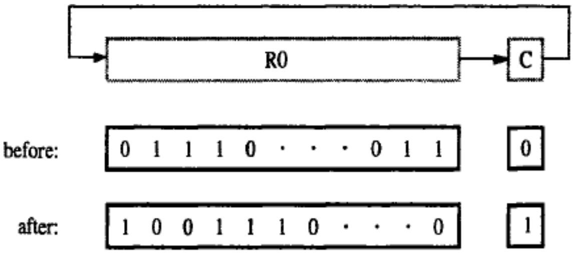
RotateRC (Rotate right including the carry flag CF)

This instruction rotate the content of the specified register on right by the specified number of bit position through carry, the number of shift position will be specified in the count variable.

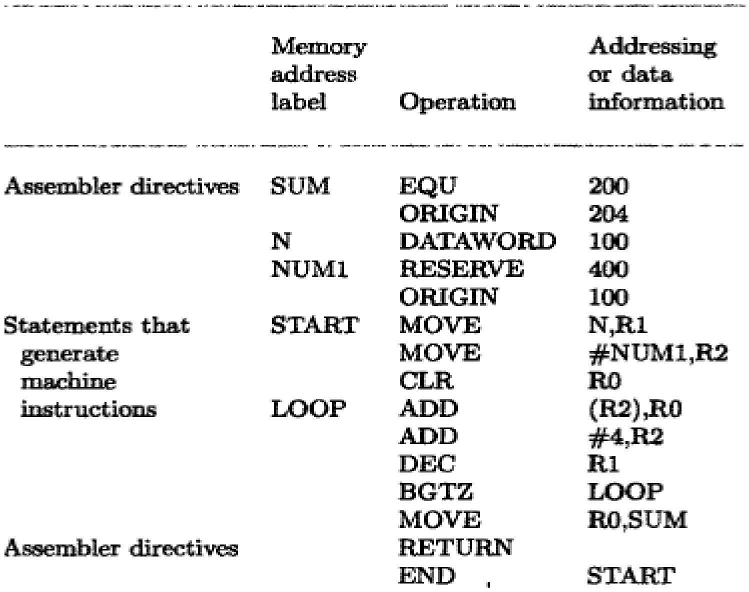
For example, the instruction

RotateRC #2,R0

Rotates the content of R0 to right by 2 bit a position through the carry is shown below.



(d) Rotate right with carry RotateRC #2,R0



	Memory address label	Operation	Addressing or data information
Assembler directives	SUM	EQU	200
		ORIGIN	204
	N	DATAWORD	100
	NUM1	RESERVE	400
		ORIGIN	100
Statements that generate machine instructions	START	MOVE	N,R1
		MOVE	#NUM1,R2
		CLR	R0
	LOOP	ADD	(R2),R0
		ADD	#4,R2
		DEC	R1
		BGTZ	LOOP
		MOVE	R0,SUM
Assembler directives		RETURN	
		END	START

9. Describe about subroutines

Subroutines

In a given program, it is often necessary to perform a particular task many times on different data values. It is prudent to implement this task as a block of instructions that is executed each time the task has to be performed. Such a block of instructions is usually called a *subroutine*.

For example, a subroutine may evaluate a mathematical function, or it may sort a list of values into increasing or decreasing order. It is possible to reproduce the block of instructions that constitute a subroutine at every place where it is needed in the program. However, to save space, only one copy of this block is placed in the memory, and any program that requires the use of the subroutine simply branches to its starting location. When a program branches to a subroutine we say that it is *calling* the subroutine. The instruction thatperforms this branch operation is named a Call instruction.

After a subroutine has been executed, the calling program must resume execution, continuing immediately after the instruction that called the subroutine. The subroutine is said to *return* to the program that called it, and it does so by executing a Return instruction. Since the subroutine may be called from different places in a calling program, provision must be made for returning to the appropriate location. The location where the calling program resumes execution is the location pointed to by the updated program counter (PC) while the Call instruction is being executed. Hence, the contents of the PC must be saved by the Call instruction to enable correct return to the calling program. The way in which a computer makes it possible to call and return from subroutines is referred to as its *subroutine linkage* method. The simplest subroutine linkage method is to save the return address in a specific location, which may be a register dedicated to this function. Such a register is called the *link register*. When the subroutine completes its task, the Return instruction returns to the calling program by branching indirectly through the link register.

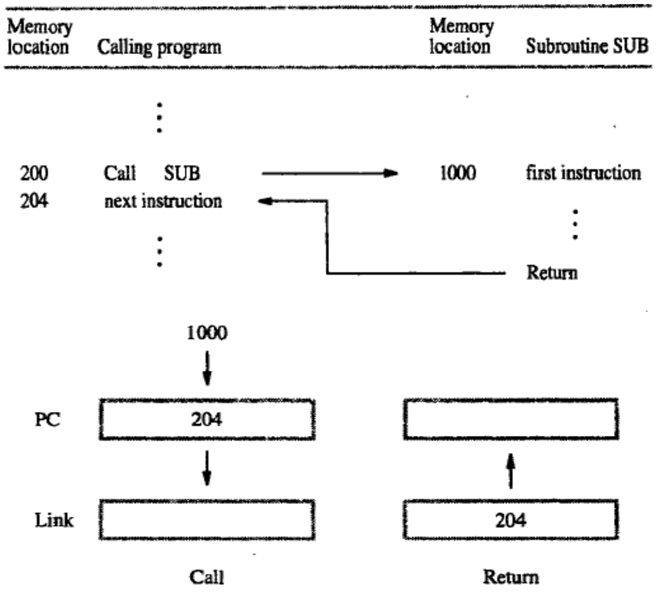
The Call instruction is just a special branch instruction that performs the following operations:

- Store the contents of the PC in the link register
- Branch to the target address specified by the Call instruction

The Return instruction is a special branch instruction that performs the operation

- Branch to the address contained in the link register

The below figure illustrates how the PC and the link register are affected by the Call and Return Instructions.



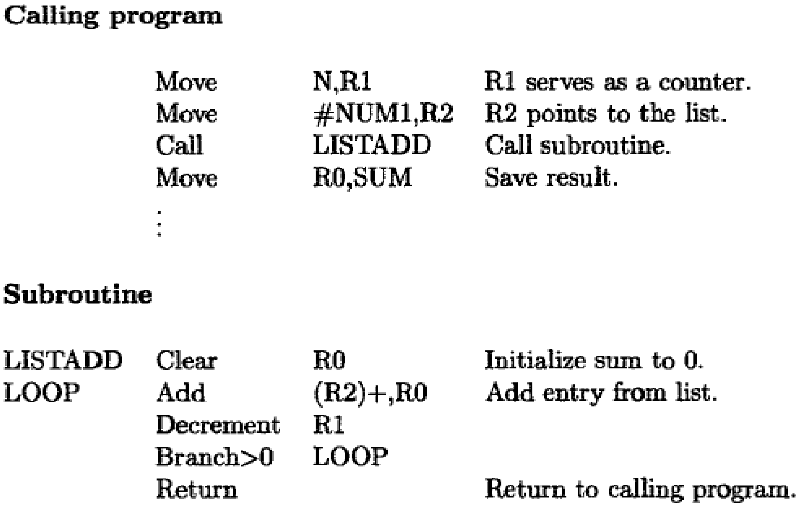
Subroutine Nesting and the Processor Stack

A common programming practice, called *subroutine nesting*, is to have one subroutine call another. In this case, the return address of the second call is also stored in the link register, overwriting its previous contents.

Hence, it is essential to save the contents of the link register in some other location before calling another subroutine. Otherwise, the return address of the first subroutine will be lost. Subroutine nesting can be carried out to any depth. Eventually, the last subroutine called completes its computations and returns to the subroutine that called it. The return address needed for this first return is the last one generated in the nested call sequence. That is, return addresses are generated and used in a last-in–first-out order. This suggests that the return addresses associated with subroutine calls should be pushed onto the processor stack.

Parameter Passing

When calling a subroutine, a program must provide to the subroutine the parameters, that is, the operands or their addresses, to be used in the computation. Later, the subroutine returns other parameters, which are the results of the computation. This exchange of information between a calling program and a subroutine is referred to as *parameter passing*. Parameter passing may be accomplished in several ways. The parameters may be placed in registers or in memory locations, where they can be accessed by the subroutine. Alternatively, the parameters may be placed on the processor stack. Passing parameters through processor registers is straightforward and efficient.



```
Calling program

          Move        N,R1        R1 serves as a counter.
          Move        #NUM1,R2    R2 points to the list.
          Call        LISTADD     Call subroutine.
          Move        R0,SUM      Save result.
          ⋮

Subroutine

LISTADD   Clear       R0          Initialize sum to 0.
LOOP      Add         (R2)+,R0    Add entry from list.
          Decrement   R1
          Branch>0    LOOP
          Return                  Return to calling program.
```

The above figure shows how the program for adding a list of numbers can be implemented as a subroutine, LISTADD, with the parameters passed through registers.

- The size of the list, *n*, contained in memory location N, and the address, NUM1, of the first number, are passed through registers R2 and R4.
- The sum computed by the subroutine is passed back to the calling program through register R3. The first four instructions in the above figure constitute the relevant part of the calling program.
- The first two instructions load *n* and NUM1 into This instruction also saves the return address (i.e., the address of the Store instruction in the calling program) in the link register.
- The subroutine computes the sum and places it in R3. After the Return instruction is executed by the subroutine, the sum in R3 is stored in memory location SUM by the calling program.
- In addition to registers R2, R3, and R4, which are used for parameter passing, the subroutine also uses R5. Since R5 may be used in the calling program, its contents are saved by pushing them onto the processor stack upon entry to the subroutine and restored before returning to the calling program.

If many parameters are involved, there may not be enough general-purpose registers available for passing them to the subroutine. The processor stack provides a convenient and flexible mechanism for passing an arbitrary number of parameters.

LISTADD, which uses the processor stack for parameter passing.

- The address of the first number in the list and the number of entries are pushed onto the processor stack pointed to by register SP.
- The subroutine is then called. The computed sum is placed on the stack before the return to the calling program.

Assume that before the subroutine is called, the top of the stack is at level 1.

- The calling program pushes the address NUM1 and the value *n* onto the stack and calls subroutine LISTADD. The top of the stack is now at level 2.
- The subroutine uses four registers while it is being executed. Since these registers may contain valid data that belong to the calling program, their contents should be saved at the beginning of the subroutine by pushing them onto the stack. The top of the stack is now at level 3.
- The subroutine accesses the parameters *n* and NUM1 from the stack using indexed addressing with offset values relative to the new top of the stack (level 3). Note that it does not change the stack pointer because valid data items are still at the top of the stack.
- The value *n* is loaded into R2 as the initial value of the count and the address NUM1 is loaded into R4, which is used as a pointer to scan the list entries.
- At the end of the computation, register R3 contains the sum. Before the subroutine returns to the calling program, the contents of R3 are inserted into the stack, replacing the parameter NUM1, which is no longer needed.
- Then the contents of the four registers used by the subroutine are restored from the stack. Also, the stack pointer is incremented to point to the top of the stack that existed when the subroutine was called, namely the parameter *n* at level 2.
- After the subroutine returns, the calling program stores the result in location SUM and lowers the top of the stack to its original level by incrementing the SP by 8.

Observe that for subroutine LISTADD in Figure, we did not use a pair of instructions
Subtract SP, SP, #4
Store R*j*, (SP)
to push the contents of each register on the stack. Since we have to save four registers, this would require eight instructions. We needed only five instructions by adjusting SP immediately to point to the top of stack that will be in effect once all four registers are saved. Then, we used the Index mode to store the contents of registers. We used the same optimization when restoring the registers before returning from the subroutine.

10. Write short notes on Stacks & Queues

Stack

A *stack* is a list of data elements, usually words, with the accessing restriction that elements can be added or removed at one end of the list only. This end is called the top of the stack, and the other end is called the bottom. The structure is sometimes referred to as a *pushdown* stack. Imagine a pile of trays in a cafeteria; customers pick up new trays from the top of the pile, and clean trays are added to the pile by placing them onto the top of the pile.

Another descriptive phrase, *last-in–first-out* (LIFO) stack, is also used to describe this type of storage mechanism; the last data item placed on the stack is the first one removed when retrieval begins. The terms *push* and *pop* are used to describe placing a new item on the stack and removing the top item from the stack, respectively.

In modern computers, a stack is implemented by using a portion of the main memory for this purpose. One processor register, called the *stack pointer* (SP), is used to point to a particular stack structure called the *processor stack.*

Data can be stored in a stack with successive elements occupying successive memory locations. Assume that the first element is placed in location BOTTOM, and when new elements are pushed onto the stack, they are placed in successively lower address locations.

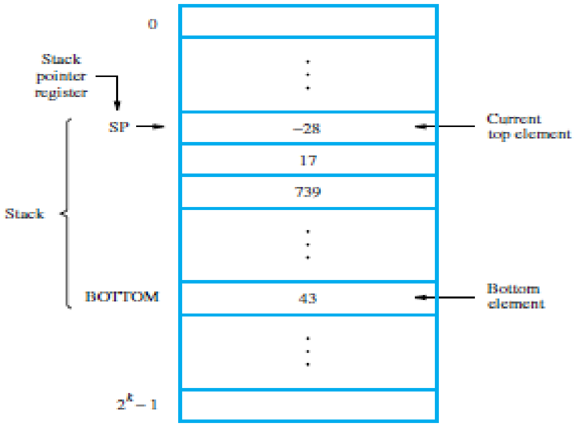
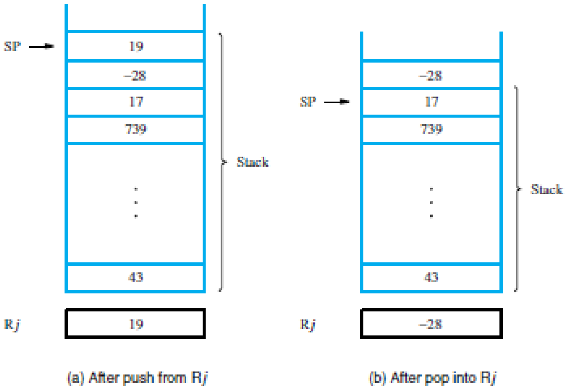


Fig. A stack of words in memory

The stack contains numerical values, with 43 at the bottom and −28 at the top. The stack pointer, SP, is used to keep track of the address of the element of the stack that is at the top at any given time. If we assume a byte-addressable memory with a 32-bit word length, the push operation can be implemented as

Subtract SP, SP, #4
Store R*j*, (SP)

where the Subtract instruction subtracts 4 from the contents of SP and places the result in SP. Assuming that the new item to be pushed on the stack is in processor register R*j*, the Store instruction will place this value on the stack. These two instructions copy the word from R*j* onto the top of the stack, decrementing the stack pointer by 4 before the store (push) operation. The pop operation can be implemented as Load R*j*, (SP)
Add SP, SP, #4
These two instructions load (pop) the top value from the stack into register R*j* and then increment the stack pointer by 4 so that it points to the new top element.



(a) After push from R*j*

(b) After pop into R*j*

Fig. Effect of stack operations

Queue

Another useful data structure that is similar to stack is called a queue. Data are stores and retrieved from a queue on a first-in-first-out (FIFO) basis. Thus new data are added at the back (high-address end) and retrieved from the front (low-address end) of the queue.

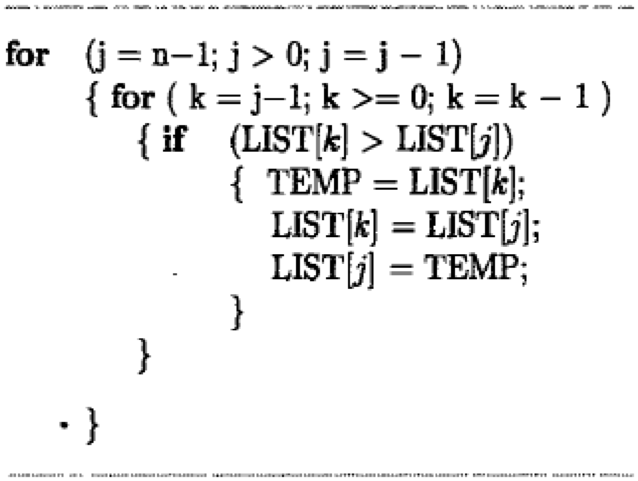
There are two important differences in how a stack and a queue are implemented.

- One end of the stack is fixed (bottom), while other end rises and falls as data are pushed and popped.
- Both ends of the queue move to higher addresses as data are added at the back and removed from the front. So two pointers are needed to keep track of the two ends of the queue.
- Another difference between a stack and a queue is that, without further control, a queue would continuously move through the memory of a computer in the direction of higher addresses.
- One way to limit the queue to a fixed region in memory is to use a circular buffer.

11. Explain with example the byte sorting program

Consider a program for sorting a list of bytes stored in memory into ascending alphabetic order. Assume that the lists consists of n bytes, not necessarily distinct, and that each byte contains the ASCII code for a character from the set of letters A through Z. when an ASCII character is stored in a byte location, it is customary to set the most significant bit position to 0. Using this code, we can sort a list of characters alphabetically by sorting their codes in increasing numerical order, considering them positive numbers.

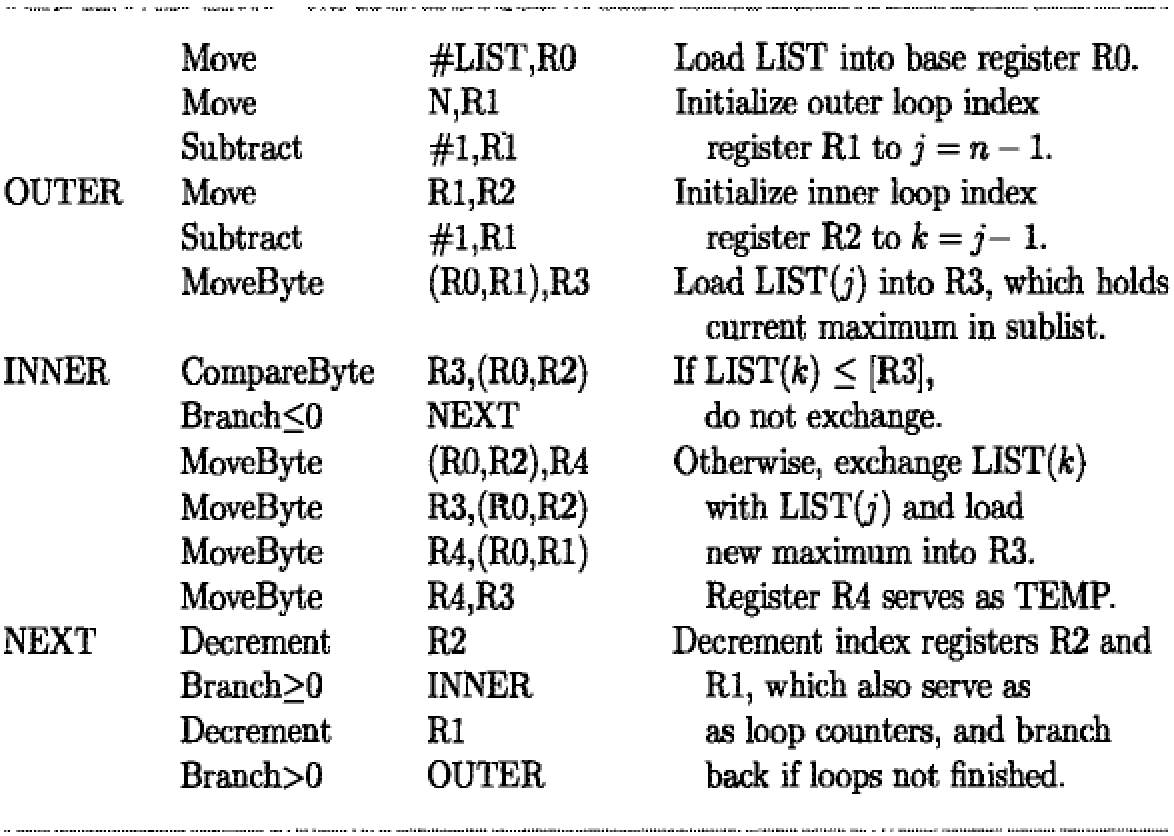
Let the list be stored in memory locations LIST, through LIST+n-1, and let n be a 32-bit value stored at address N. We sort using a straight-selection sort algorithm. The largest number is found and placed at the end of the list in location LIST+n-1. Then the largest number in the remaining sublist of n-1 numbers is placed at the end of the sublist in location LIST+n-2. The procedure is repeated until the list is sorted.



```
for   (j = n–1; j > 0; j = j – 1)
      { for ( k = j–1; k >= 0; k = k – 1 )
          { if    (LIST[k] > LIST[j])
                  {  TEMP = LIST[k];
                     LIST[k] = LIST[j];
                     LIST[j] = TEMP;
                  }
          }

    • }
```

(a) C-language program for sorting



```
          Move          #LIST,R0      Load LIST into base register R0.
          Move          N,R1          Initialize outer loop index
          Subtract      #1,R1           register R1 to j = n – 1.
OUTER     Move          R1,R2         Initialize inner loop index
          Subtract      #1,R1           register R2 to k = j– 1.
          MoveByte      (R0,R1),R3    Load LIST(j) into R3, which holds
                                        current maximum in sublist.
INNER     CompareByte   R3,(R0,R2)    If LIST(k) ≤ [R3],
          Branch≤0      NEXT            do not exchange.
          MoveByte      (R0,R2),R4    Otherwise, exchange LIST(k)
          MoveByte      R3,(R0,R2)      with LIST(j) and load
          MoveByte      R4,(R0,R1)      new maximum into R3.
          MoveByte      R4,R3           Register R4 serves as TEMP.
NEXT      Decrement     R2            Decrement index registers R2 and
          Branch≥0      INNER           R1, which also serve as
          Decrement     R1              as loop counters, and branch
          Branch>0      OUTER           back if loops not finished.
```

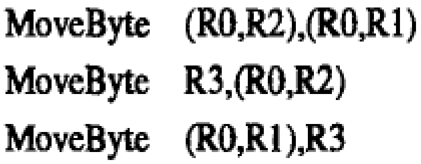
(b) Assembly language program for sorting

Control flow is handled differently in the two programs for purposes of efficiency in the assembly language program.

Using the if-then control statement in the C-language program causes the three-line then clause to exchange LIST(k) and LIST(j) if LIST(k)>LIST(j).

In the assembly language program, a branch is taken around the four-instruction exchange code if LIST(k)<=LIST(j).

If the machine instruction set allows a move operation from one memory location directly to another memory location, then the four-instruction exchange code in the inner loop can be replaced by the three-instruction sequence.



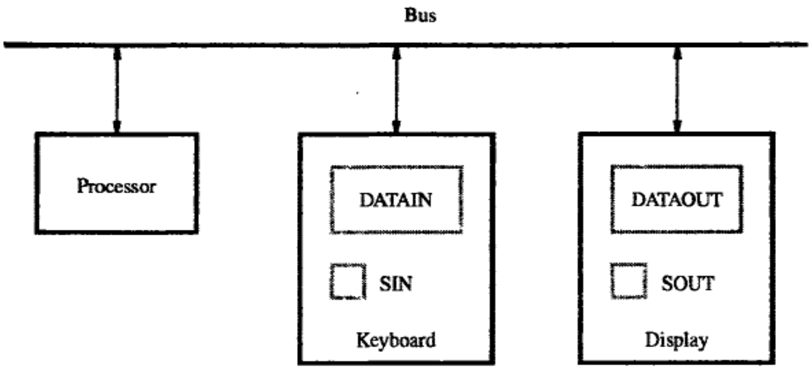
```
MoveByte    (R0,R2),(R0,R1)
MoveByte    R3,(R0,R2)
MoveByte    (R0,R1),R3
```

The above program works correctly only if the list has at least two elements because the check for loop termination is done at the end of each loop. Hence, there is at least one pass through the loop, regardless of the value of n.

12. Explain Basic input /output operation

Consider a task that reads in character input from a keyboard and produces character output on a display screen. A simple way of performing such I/O reads is to use a method known as program controlled I/O.The rate of data transfer from the keyboard to a computer is limited by the typing speed of the user, which is unlikely to exceed a few characters per second. The rate of output transfers from the computer to the display is much higher. It is determined by the rate at which characters can be transmitted over the link between the computer and the display device, typically several thousand characters per second. however, this is still much slower than the speed of a processor that can execute many millions of instructions per second. The difference between the processor and I/O devices creates the need for mechanisms to synchronize the transfer of data between them.

A solution to this problem is ,the processor sends the first character and then waits for a signal from the display that the character has been received. It then sends the second character and so on. Input is sent from the keyboard in a similar way ,the processor waits for a signal indicating that a character key has been stuck and that its code is available in some buffer register associated with the keyboard. Then the processor proceeds to read that code. The keyboard and the display are separate devices as shown below



The action of striking a key on the keyboard does not automatically cause the corresponding character to be displayed on the screen. One block of instructions in the I/O program transfers the character into the processor, and another associated block of instruction causes the character to be displayed.

Consider the problem of moving a character code from the keyboard to the processor, striking a key stores the corresponding character cede in a 8-bit buffer register associated with the keyboard.

This register called DATAIN .To inform the processor that a valid character is in DATAIN ,a status control flag ,SIN ,is set to 1.A program monitors SIN , and when SIN is set to 1,the processor reads the contents of DATAIN.When the character is transferred to the processor, SIN is automatically cleared to 0.If a second character is entered at the keyboard, SIN is again set to 1 and the process repeats.

An analogous process takes place when characters are transferred from the processor to the display. A buffer register, DATAOUT ,and a status control flag, SOUT ,are used for this transfer .When SOUT equals 1,the display is ready to receive a character. Under program control, the processor monitors SOUT ,and when SOUT is set to 1,the processor transfers a character code to DATAOUT..The transfer of a character to DATAOUT clears SOUT to 0;when the display device is ready to receive a second character, OUT is again set to 1.The buffer registers DATAIN and DATAOUT and the status flags SIN and SOUT are part of circuitry commonly known as a device interface. The circuitry for each device is connected to the processor via a bus is shown above.

UNIT II

The IA-32 Pentium Example: Registers and Addressing, IA-32 Instructions, IA-32 Assembly Language, Program Flow Control, Logic and Shift/Rotate Instructions, I/O Operations, Subroutines, Other Instructions, Program Examples.

2 Marks

1. Write about IA-32

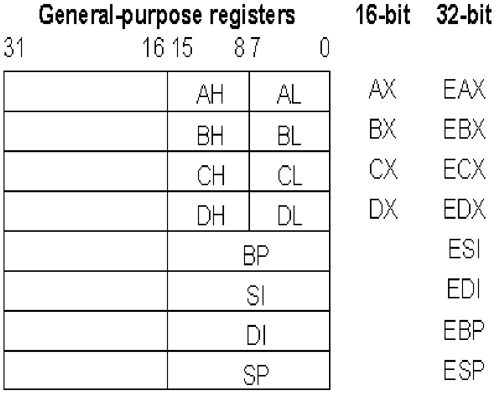
The Intel architecture (IA) processors operate with 32-bit memory address and 32-bit data operands. They are referred to as IA-32 processors, and the most recent Pentium series.

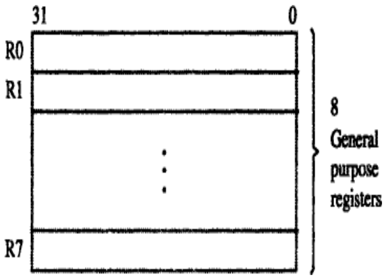
The first IA-32 processor was 80386 then 80486, Pentium, Pentium Pro, Pentium II, Pentium III, Pentium 4 has been implemented. Theses processors has increasing level of performance, achieved through a number of architectural and microelectronic technology improvements. The latest members of the family have specialized instructions for handling multimedia information and for vector data processing.

2. List out the various register of IA- 32 Register structure

- General purpose registers
- floating point registers
- segment registers
- instruction pointer
- status registers

3. Write about General purpose registers

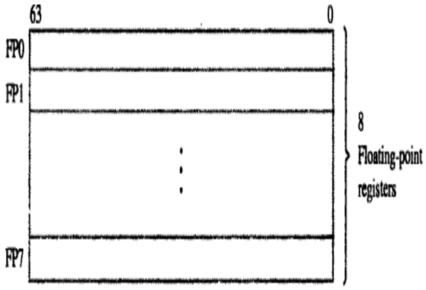




The above diagram shows the general purpose registers. The eight 32-bit registers labeled R0 through R7 are general purpose registers that can be used to bold either data operands or addressing information.

4. Write about floating point registers

There are eight floating point registers for holding double word or quad word (64 bits) floating point data operands. The floating point registers have an extension field to provide a total length of 80 bits, the extra bits are used for increased accuracy while floating point numbers are operated on in the processor.



5. What is Segment registers

There are six segment registers that hold 16-bit segment selectors. A segment selector is a special pointer that identifies a segment in memory. The six segment registers are:

- CS: code segment register
- SS: stack segment register
- DS, ES, FS, GS: data segment registers

Code segment (CS)

The code segment holds the instructions of a program.

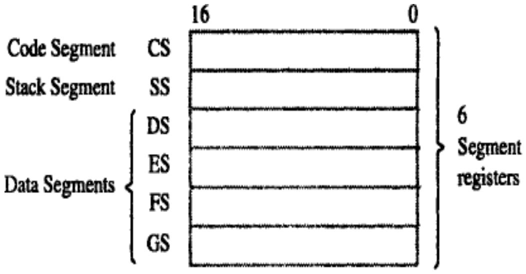
Stack segment (SS)

The stack segment contains the processor stack.

Data segment (DS, RD, FS, and GS)

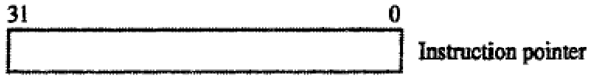
Four data segments are provided for holding data operands. These four data segment registers provide programs with flexible and efficient ways to access data.

The six segment registers below contain selector values that are used in locating these segments in the memory address space.

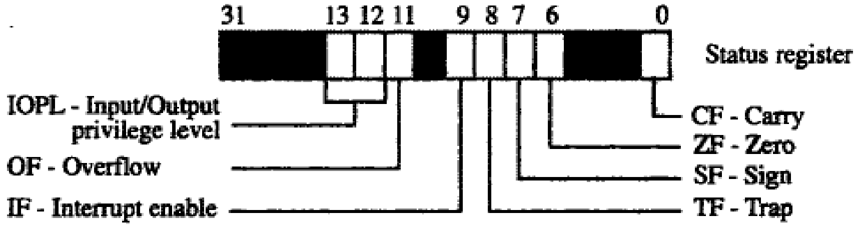


6. Define Instruction pointer

The **EIP** register is the 32-bit instruction pointer. The EIP register (or instruction pointer) can also be called "program counter." It contains the offset in the current code segment for the next instruction to be executed. It is advanced from one instruction boundary to the next in straight-line code or it is moved ahead or backwards by a number of instructions when executing JMP,CALL, RET instructions.



7. Write about status registers



The status register holds the condition code flags (CF,ZF,SF,OF).These flags contain information about the results of arithmetic operations.

8. What is EFLAGS Register

The 32-bit EFLAGS register contains a group of status flags, a control flag, and a group of system flags. When it is used in the conditional control instructions look at the condition code bits (in the EFLAGS register) to make a decision on whether to take the jump or not.

9. Write about ESP register and EBP register

The **ESP** register is the 32-bit stack pointer, used to manage push and pop operations.

The **EBP** register is the 32-bit base pointer. In connection with the ESP, the EBP is used to manage the stack frame, that part of the stack used to communicate with subprograms and store local variables.

10. Write about the Index Registers

The **ESI** and **EDI** registers are used as source and destination addresses for string and array operations.

The ESI "**Extended Source Index**" and EDI "**Extended Destination Index**" facilitate high–speed memory transfers.

11. List the various addressing modes of IA-32

The addressing mode of IA-32 includes:

1. Immediate mode
2. Direct mode
3. Register mode
4. Register indirect mode
5. Base with displacement mode
6. Index with displacement mode
7. Base with index mode
8. Base with index and displacement mode

12. What is immediate mode? Give example

The operand is contained in the instruction. It is a signed 8-bit or 32-bit number, with the length being specified by a bit in the OP code of the instruction. Thus bit is 0 for the short version and 1 for the long version.

Instruction format:

Opcode	Register	Value

Example:

MOV EAX, 25

The above instruction moves the decimal value 25 into the EAX register. A number given in this form using the digits 0 through 9 is assumed to be in decimal notation .the suffix B and H are used to specify binary and hexadecimal numbers, respectively .For example the instruction,

MOV EAX, 3FAOOH

Moves the hex number 3FA00 into EAX.

13. What is direct mode? Give example

The memory address of the operand is given by a 32-bit value in the instruction

Instruction format:

Opcode	Register	Location

Example:

MOV EAX, LOCATION

The above instruction uses the direct addressing mode to move the doubleword at the memory location specified by the address label LOCATION into register EAX.This assumes that the LOCATION has been defined as an address label for a memory location in the data declaration. If LOCATION represents the address 1000 then this instruction moves the doubleword at 1000 into EAX.

In the IA-32 assembly language square brackets can be used to explicitly indicate the direct addressing mode as in the instruction.

MOV EAX, [LOCATION]

14. What is Register mode? Give example

The operand is contained in one of the general purpose register specified in the instruction.

Instruction format:

Opcode	Dest.Register	Src.Register

Example:

MOV EAX, ECX

Both operands use register mode. The contents of register **ECX** are copied to register EAX.

Before execution of the above instruction, the contents of ECX and EAX are:

ECX
50

EAX
00

After execution

ECX
50

EAX
50

15. What is Register indirect mode? Give example

The memory address of the operand is contained in one of the eight general purpose register specified in the instruction.

Instruction format:

Opcode	**Register**	**[Register]**

Example:

MOV EAX,[EBX]

The above instruction moves the contents of LOCATION specified by the register EBX into the register EAX.

Before execution of the above instruction, the contents of EAX and EBX are:

EAX
00

EBX
1000

20

1000

After execution

EAX
20

EBX
1000

20

1000

16. What is Base with displacement mode? Give example

An 8-bit or 32-bit signed displacement and one of the eight general purpose register to be used as a base register are specified in the instruction. The effective address of the operand in the sum of the contents of the base register and the displacement.

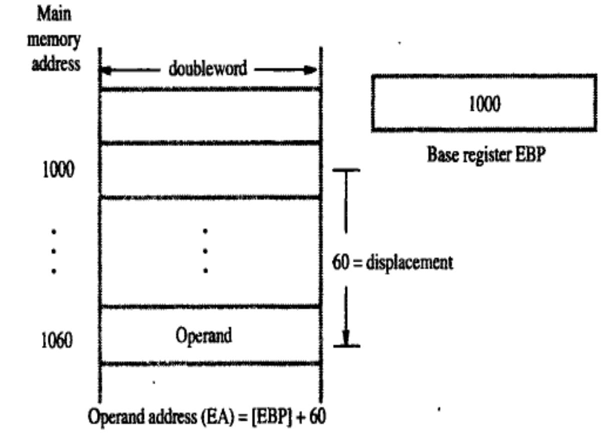
Instruction format:

Opcode	**Dest.Register**	**[Register]+Displacement**

Example:

MOV EAX, [EBP + 60]

The second operand uses base displacement mode. The instruction contains a constant. That constant is added to the contents of register EBP to form an effective address. The contents of memory at the effective address are copied into register EAX.



(a) Base with displacement mode, expressed as [EBP + 60]

17. What is Index with displacement mode? Give example

A 32-bit signed displacement, one of the eight general purpose register to be used as an index register, and a scale factor of 1,2,4 or 8 are specified in the instruction. To obtain the effective address of the operand, the contents of the index register are multiplied by the scale factor and then added to the displacement.i.e

Offset = (Index * Scale) + displacement

Instruction format:

Opcode	Dest.Register	[Register] * Scale factor +Displacement

Example:

MOV AL,[EBP * 4 + 10]

18. What is Base with index mode? Give example

Two of the eight general purpose register and a scale factor of 1,2,4 or 8 are specified in the instruction. The register is used as base and index register and the effective address of the operand is calculated as follows: the contents of the index register are multiplied by the scale factor and added to the contents of the base register. i.e

Offset = Base + (Index * Scale)

Instruction format:

Opcode	Dest.Register	[Register1] + [Register2] * Scale factor

Example:

MOV EAX, [ESP+ESI*4]

The contents of registers ESI is multiplied with the scale factor 4 and then the content of ESP is added to form an effective address. The contents of memory at the effective address are copied into register EAX

19. What is Base with index and displacement mode? Give example

An 8 bit or 32 –bit signed displacement two of the eight general purpose registers and a scale factor of 1,2,4 or 8 are specified in the instruction. The register is used as base and index register and the effective address of the operand is calculated as follows: the contents of the index register are multiplied by the scale factor and then added to the contents of the base register and the displacement.

An effective address is computed by:

Offset = Base + (Index * Scale) + displacement

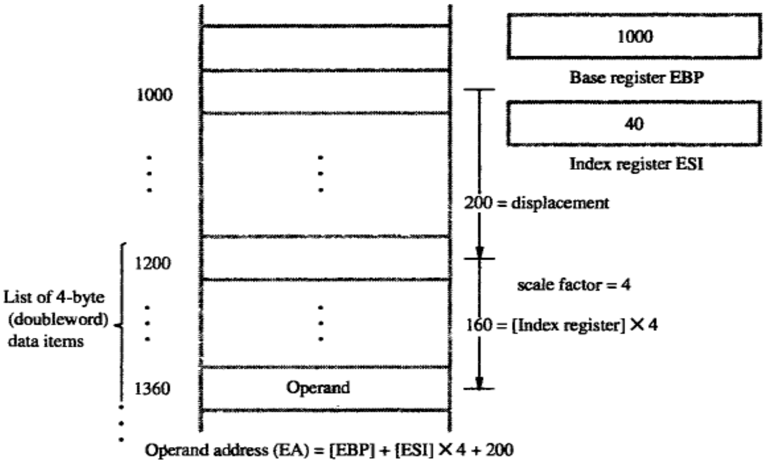
Instruction format:

Opcode	**Dest.Register**	**[Register1] + [Register2] * Scale factor + Displacement**

Example:

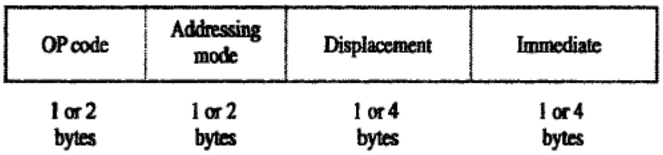
MOV EAX, [EBP+ESI*4 + 200]

The contents of registers ESI is multiplied with the scale factor 4 and then the content of EBP and displacement are added to form an effective address. The contents of memory at the effective address are copied into register EAX


1000
Base register EBP
1000
40
Index register ESI
200 = displacement
1200
List of 4-byte (doubleword) data items
scale factor = 4
160 = [Index register] X 4
1360
Operand
Operand address (EA) = [EBP] + [ESI] X 4 + 200

(b) Base with displacement and index mode, expressed as [EBP + ESI * 4 + 200]

20.Write about IA-32 Instruction



OP code	Addressing mode	Displacement	Immediate
1 or 2 bytes	1 or 2 bytes	1 or 4 bytes	1 or 4 bytes

The instructions are variable in length ,ranging from one byte to 12 bytes ,consisting of up to four fields .the OP-code field consists of one or two bytes, with most instructions requiring only one byte, the addressing mode information is contained in one or two bytes immediately following the OP code. For instructions that involve the use of only one register in generating the effective address of an operand ,only one byte is needed in the addressing mode field. Two bytes are needed for encoding the last two addressing modes. These modes use two registers to generate the effective address of a memory operand.

21. What is One byte instruction?

Registers can be incremented or decremented by instructions, that occupy one byte. Examples are

INC EDI

And

DEC ECX

In which the general purpose register EDI and ECX are specified by 3-bit codes in the single OP-code byte.

22. What is immediate mode encoding?

The OP-code specifies when the immediate addressing mode is used. For example the instruction

MOV EAX, 820

Is encoded into 5 bytes .a one –byte OP code specifies the move operation, the fact that a 32-bit immediate operand is used and the name of the destination register. The OP code byte is directly followed by the 4-byte immediate value of 820.when an 8-bit immediate operand is used, as in the instruction

MOV DL, 5

Only two bytes are needed to encode the instruction

23. In which situation a program flow control changes.

There are two main ways in which the flow of executing instructions varies from straight –line sequencing, calls to subroutines and returns from them break straight line sequencing. Also, branch instructions, either conditional or unconditional, can cause a break. The branch instructions are called jumps.

24. Write about Conditional jump instruction

The conditional Jump instructions test the four condition code flags in the status register. The instruction

JG LABEL

is an example of a conditional Jump instruction. The condition is *greater-than* as indicated by the G suffix in the OP code.

25. Write about Unconditional Jump Instruction

An unconditional Jump instruction, JMP, causes a branch to the instruction at the target address. In addition to using short (one-byte) or long (four-byte) relative signed offsets to determine the target address, as is done in conditional Jump instructions, the JMP instruction also allows the use of other addressing modes.

26. What is the use of Compare Instructions?

It is often necessary to make conditional jumps in a program based on the results of comparing two numbers. The compare instruction

CMP dst,src

performs the operation

$$dst \leftarrow [src]$$

and sets the condition code flags based on the result obtained. Neither of the operands is changed .the first operand is always compared to the second. For example, the compare instruction by a conditional jump that is based on the “greater than” condition, then the jump will take to the target address if the destination operand is greater than the source operand.

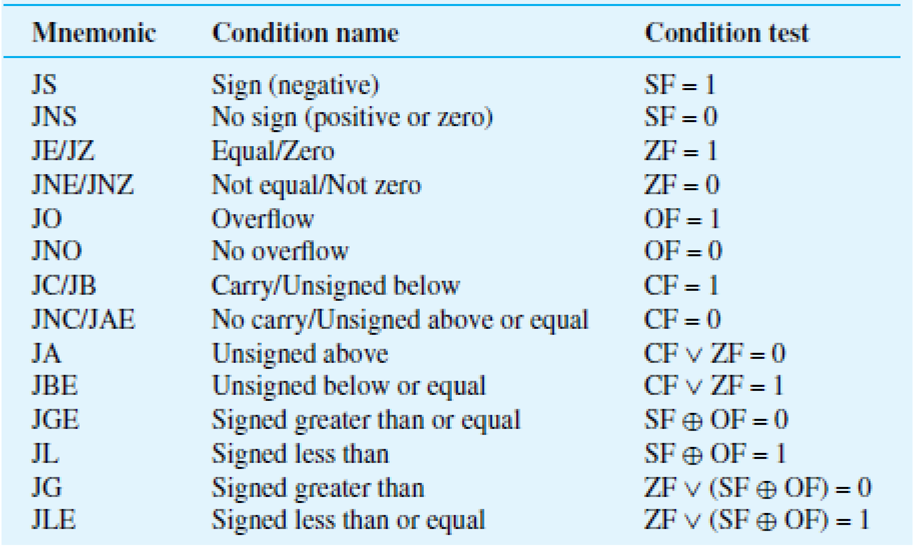
27. What is Logical Shift instruction?

An operand can be shifted right or left, using either logical or arithmetic shifts,by a number of bit positions determined by a specified count. The format of the shift instruction is

OPcode dst , count

Where the destination operand to be shifted is specified by the general addressing modes and the count is given either as an 8-bit immediate value or is contained in the 8-bit register CL.

28. List out the various jump instructions of IA-32



Mnemonic	Condition name	Condition test
JS	Sign (negative)	SF = 1
JNS	No sign (positive or zero)	SF = 0
JE/JZ	Equal/Zero	ZF = 1
JNE/JNZ	Not equal/Not zero	ZF = 0
JO	Overflow	OF = 1
JNO	No overflow	OF = 0
JC/JB	Carry/Unsigned below	CF = 1
JNC/JAE	No carry/Unsigned above or equal	CF = 0
JA	Unsigned above	$CF \vee ZF = 0$
JBE	Unsigned below or equal	$CF \vee ZF = 1$
JGE	Signed greater than or equal	$SF \oplus OF = 0$
JL	Signed less than	$SF \oplus OF = 1$
JG	Signed greater than	$ZF \vee (SF \oplus OF) = 0$
JLE	Signed less than or equal	$ZF \vee (SF \oplus OF) = 1$

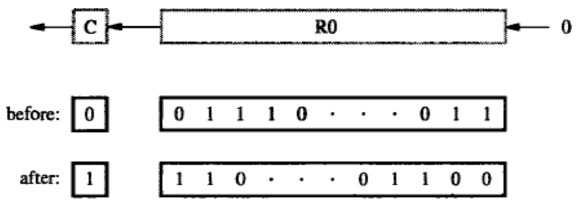
29. List out the various logical shift instruction

There are four shift instructions:

- SHL (Shift left logical)
- SHR (Shift right logical)
- SAL (Shift left arithmetic; operation is identical to SHL)
- SAR (Shift right arithmetic)

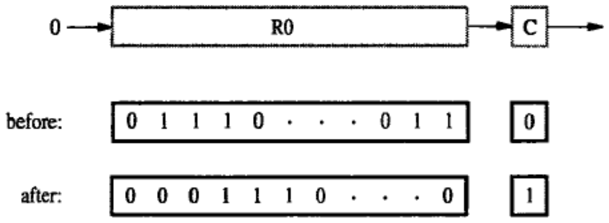
30. What is SHL instruction?

The SHL (shift left) instruction performs a logical left shift on the destination operand, filling the lowest bit with 0.For example, **SHL R0, #2**



31. What is SHR instruction?

The SHR (shift right) instruction performs a logical right shift on the destination operand. The highest bit position is filled with a zero. For example, **SHR R0,#2**



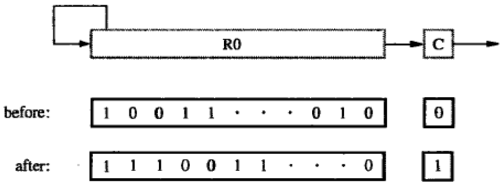
32. What is SAL and SAR instruction

SAL:

The operation of SAL is identical to SHL.

SAR:

SAR (shift arithmetic right) performs a right arithmetic shift on the destination operand. For example, **SAR R0,#2**



before: 1 0 0 1 1 · · · 0 1 0 | 0

after: 1 1 1 0 0 1 1 · · · 0 | 1

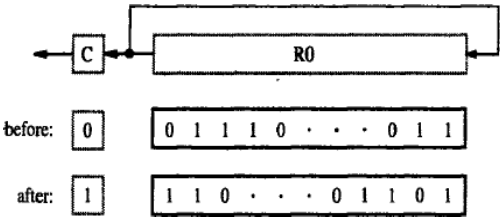
33. List out the various rotate instruction

In addition to the shift instructions, there are also four rotate instructions:

- ROL (Rotate left without the carry flag CF)
- ROR (Rotate right without the carry flag CF)
- RCL (Rotate left including the carry flag CF)
- RCR (Rotate right including the carry flag CF)

34. What is ROL instruction?

- ROL (rotate) shifts each bit to the left
- The highest bit is copied into both the Carry flag and into the lowest bit
- No bits are lost
- For example , **ROL R0,#2**

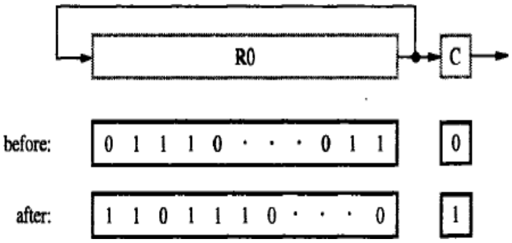


before: 0 | 0 1 1 1 0 · · · 0 1 1

after: 1 | 1 1 0 · · · 0 1 1 0 1

35. What is ROR Instruction?

- ROR (rotate right) shifts each bit to the right
- The lowest bit is copied into both the Carry flag and into the highest bit
- No bits are lost
- For example , **ROR R0,#2**

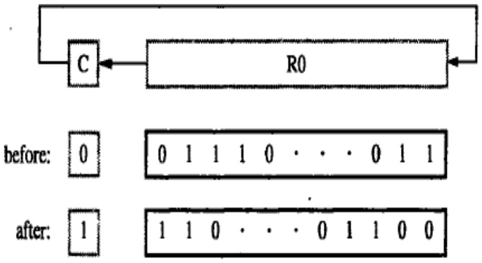


before: 0 1 1 1 0 · · · 0 1 1 | 0

after: 1 1 0 1 1 1 0 · · · 0 | 1

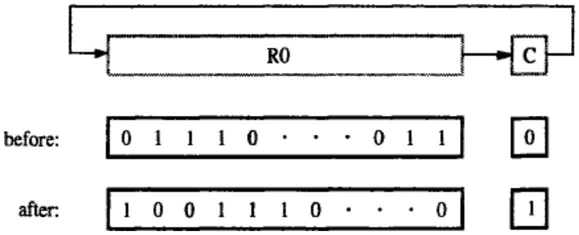
36. What is RCL instruction?

- RCL (rotate carry left) shifts each bit to the left
- Copies the Carry flag to the least significant bit
- Copies the most significant bit to the Carry flag
- For example , **RCL R0,#2**



37. What is RCR instruction?

- RCR (rotate carry right) shifts each bit to the right
- Copies the Carry flag to the most significant bit
- Copies the least significant bit to the Carry flag
- For example , **ROR R0,#2**



38. What is memory mapped I/O?

- Instead of having special methods for accessing the values to be read or written, just get them from memory or put them into memory.
- The device is connected directly to certain main memory locations.

39. Why use memory mapped I/O?

- Makes programming simpler.
- Do not have special commands to access I/O devices.
  - Just use lw and sw.
- Takes some memory locations
  - Very few compared to the size of main memory.

40. What is Isolated I/O?

The IA-32 instructions set also have two instructions, with OP codes IN and OUT, that are used only for I/O purposes. The addresses issued by these instructions are in an address space that is separate from the memory address space used by the other instructions. This arrangement is called isolated I/O .

41. Write short notes on block transfers

The IA-32 architecture also has two block transfer I/O instructions REPINS and REPOUTS .they transfer a block lf data serially, one item at a time, between the memory and an I/O device. The S suffix in the

OP codes stands for string, and the REP prefix stands for “repeat the item by item transfer until the complex block has been transferred”. The instructions themselves do not specify the parameters needed to describe the transfer. These parameters are specified implicitly by processor registers DX,EDI and ECX as follows:

DX contains a 16-bit I/O device address

EDI contains a 32-bit address for the beginning of a block in memory

ECX contains the number of data items to be transferred.

A suffix B or D in the OP-code mnemonic indicates that the item size is either of byte or doubleword length. Thus REPINSB is a byte –block transfer, and REPINSD is a doubleword –block transfer.

42. How block transfer instruction operates?

The block transfer instruction operates as follows: After each data item is transferred, the index register EDI is incremented by 1 or 4 depending on the size of the data items, and the ECX register is decremented by 1.The transfers are repeated until the contents of the counter register ECX have been decremented to 0.The effect of these single instruction is equivalent to a program loop that uses register ECX as the loop counter

43. Write short notes on Subroutines

In the IA-32 architecture, register ESP is used as the stack pointer. It points to the current top element (TOS) in the processor stack. The stack grows toward lower numbered addresses. The width of the stack is 32 bits, that is, all stack entries are double words.

There are two instructions for pushing and popping individual elements onto and off the stack. The instruction

PUSH src

decrements ESP by 4, and then stores the doubleword at location src into the memory location pointed to by ESP. The instruction

POP dst

reverses this process by retrieving the TOS doubleword from the location pointed to by ESP,storing it at location dst, and then incrementing ESP by 4. These instructions implicitly use ESP as the stack pointer. The source and destination operands are specified using the IA-32 addressing modes.

44. List out the other instructions that are available in IA-32

The other instructions available in IA-32 are:

1. Multiplication instruction:
2. Division instruction:
3. Multimedia Extension (MMX) instructions
4. Vector (SIMD) Floating-Point Operations

45. What is Multiplication instruction?

The signed integer multiplication instruction, IMUL, performs 32-bit multiplication. Depending on the form of the instruction that is used, the destination may be implicit and the 64-bit product may be truncated to 32 bits.

One form of this instruction is

IMUL src

which implicitly uses the EAX register as the multiplicand. The multiplier specified by src can be in a register or in the memory. The full 64-bit product is placed in registers EDX (high-order half) and EAX (low-order half).

46. Write about Division instruction:

The integer divide instruction, IDIV, operates on a 64-bit dividend and a 32-bit divisor to generate a 32-bit quotient and a 32-bit remainder. The format of the instruction is

IDIV src

The source operand is the divisor. The 64-bit dividend is formed by the contents of register EDX (high-order half) and register EAX (low-order half). After performing the division, the quotient is placed in EAX and the remainder is placed in EDX.

47. What is MMX instruction?

The IA-32 instruction set has a number of instructions that operate in parallel on such data packed into 64-bit quadwords.(A quadword contains 8 bytes or four 16-bit words).These instructions are called *multimedia extension* (MMX) instructions.

The operands for MMX instructions can be in the memory, or in the eight floating-point registers. Thus, these registers serve a dual purpose. They can hold either floating-point numbers or MMX operands. When used by MMX instructions, the registers are referred to as MM0 through MM7.

48. Write about Vector (SIMD) Floating-Point Operations

A set of instructions that are used to perform arithmetic operations on small group of floating point numbers is provided.SIMD (single-instruction-multiple-data) instructions are useful for vector and matrix calculations in scientific applications.

In Intel terminology, these instructions are called streaming SIMD extension (SSE) instructions. They handle packed 128-bit double quadwords, each consisting of four 32-bit floating-point numbers. Eight additional 128-bit registers, XMM0 to XMM7, are available for holding these operands.

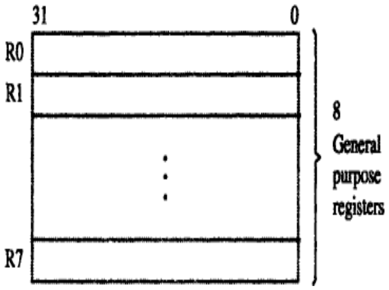
11 Marks

1. Write in detail about IA-32 registers.

IA- 32 Register structure contains the following

- General purpose registers
- floating point registers
- segment registers
- instruction pointer
- status registers

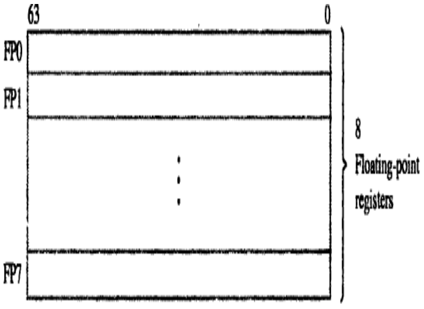
General purpose registers:



The above diagram shows the general purpose registers. The eight 32-bit registers labeled R0 through R7 are general purpose registers that can be used to bold either data operands or addressing information.

Floating point registers:

There are eight floating point registers for holding doubleword or quadword (64 bits) floating point data operands. The floating point registers have an extension field to provide a total length of 80 bits, the extra bits are used for increased accuracy while floating point numbers are operated on in the processor.



Segment registers:

IA-32 architectures are based on a memory model that associates different areas of the memory, called segments, with different usages. There are three segments,

Code segment (CS)

The code segment holds the instructions of a program.

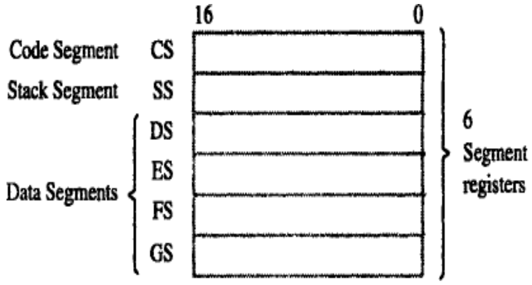
Stack segment (SS)

The stack segment contains the processor stack

Data segment (DS,RD,FS,GS)

Four data segments are provided for holding data operands.

The six segment registers below contain selector values that are used in locating these segments in the memory address space.

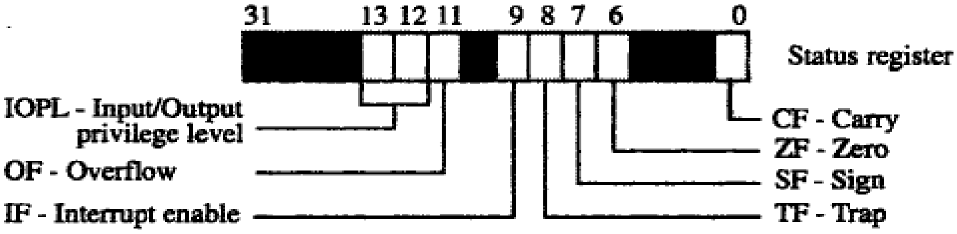


Instruction pointer:

It is a 32-bit register. It serves as the program counter and contains the address of the next instruction to be executed

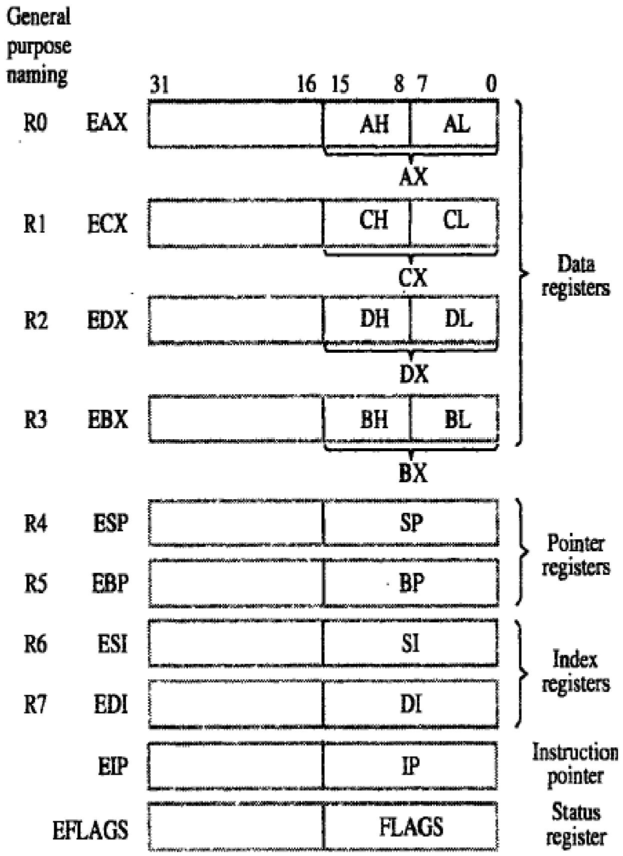
31
0
Instruction pointer

Status registers:



The status register holds the condition code flags (CF, ZF, SF, OF).These flags contain information about the results of arithmetic operations.

Compatibility of the IA-32 register structure:



The eight general purpose registers are grouped into 3 different types

1. Data register
2. Pointer register
3. Index register

The data register used for holding operands ,and the pointer and index registers for holding address and address indices used to determine the effective address of a memory operand.

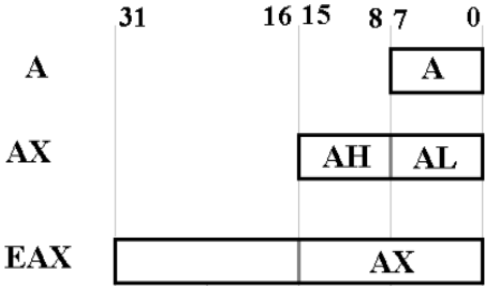
The above figure shows the compatibility of the IA-32 register structure with earlier Intel processor register structures.

The Intel’s original 8-bit processors, the data registers were called A,B,C and D.In Intel 16-bit processors, these registers were labeled AX,BX,CX and DX .the high and low-order bytes in each register are identified by suffixes H and L .For example, the two bytes in register AX are referred to as AH and AL.

In IA-32 processors, the prefix E is used to identify the corresponding “extended “32-bit register :EAX,EBX,ECX and EDX.The E –prefix labeling is also used for the other 32-bit registers shown in the above figure. They are the extended versions of the corresponding 16-bit register used in earlier processors. The IA-32 processor state can be switched dynamically between 32-bit operation and 16-bit operation during program execution on an instruction by instruction basis by the use of instruction prefix bytes.

EAX register:

This is the general–purpose register used for arithmetic and logical operations. This division is seen also in the EBX, ECX, and EDX registers; the code can reference BX, BH, CX, CL, etc. This register has an implied role in both multiplication and division.



EBX register:

This can be used as a general–purpose register, but was originally designed to be the base register, holding the address of the base of a data structure. The easiest example of such a data structure is a singly dimensioned array.

ECX register:

This can be used as a general–purpose register, but it is often used in its special role as a counter register for loops or bit shifting operations.

EDX register:

This can be used as a general–purpose register. It also plays a special part in executing integer multiplication and division. For multiplication, DX or EDX store the more significant bits of the product. The 16–bit implementation of multiplication uses AX to hold one of the integers to be multiplied and uses the register pair DX:AX to hold the 32-bit product.

The 32–bit implementation of multiplication uses EAX to hold one of the integers to be multiplied and uses the register pair EDX:EAX to hold the 64-bit product.

The Instruction Pointer(EIP)

The **EIP** register is the 32-bit instruction pointer.The EIP register (or instruction pointer) can also be called "program counter." It contains the offset in the current code segment for the next instruction to be executed. It is advanced from one instruction boundary to the next in straight-line code or it is moved ahead or backwards by a number of instructions when executing JMP,CALL, RET instructions.

The Index Registers

The **ESI** and **EDI** registers are used as source and destination addresses for string and array operations.

The ESI "**Extended Source Index**" and EDI "**Extended Destination Index**" facilitate high–speed memory transfers.

The Other Pointers register's are:

1.ESP register:

The **ESP** register is the 32-bit stack pointer, used to manage push and pop operations.

2.EBP Register:

The **EBP** register is the 32-bit base pointer. In connection with the ESP, the EBP is used to manage the stack frame, that part of the stack used to communicate with subprograms and store local variables.

3.EFLAGS Register:

The **EFLAGS** register holds a collection of at most 32 Boolean flags. The flags are divided into two broad categories: **control flags** and **status flags**.

2. Explain IA-32 Addressing modes

The IA-32 architecture has a large and flexible set of addressing modes. They are designed to access individual data items or data items that are members of an ordered list begins at a specified memory address. There are also several addressing modes that provide mere flexibility in accessing data operands in the memory.

The addressing mode of IA-32 includes:

1. Immediate mode
2. Direct mode
3. Register mode
4. Register indirect mode
5. Base with displacement mode
6. Index with displacement mode
7. Base with index mode
8. Base with index and displacement mode

Immediate mode

The operand is contained in the instruction. It is a signed 8-bit or 32-bit number, with the length being specified by a bit in the OP code of the instruction. Thus bit is 0 for the short version and 1 for the long version.

Instruction format:

Opcode	Register	Value

Example:

MOV EAX,25

The above instruction moves the decimal value 25 into the EAX register. A number given in this form using the digits 0 through 9 is assumed to be in decimal notation .the suffix B and H are used to specify binary and hexadecimal numbers, respectively .For example the instruction,

MOV EAX,3FAOOH

Moves the hex number 3FA00 into EAX.

Direct mode

The memory address of the operand is given by a 32-bit value in the instruction

Instruction format:

Opcode	Register	Location

Example:

MOV EAX,LOCATION

The above instruction uses the direct addressing mode to move the doubleword at the memory location specified by the address label LOCATION into register EAX.This assumes that the LOCATION has been defined as an address label for a memory location in the data declaration. If LOCATION represents the address 1000 then this instruction moves the doubleword at 1000 into EAX.

In the IA-32 assembly language square brackets can be used to explicitly indicate the direct addressing mode as in the instruction.

MOV EAX,[LOCATION]

Register mode

The operand is contained in one of the general purpose register specified in the instruction.

Instruction format:

Opcode	Dest.Register	Src.Register

Example:

MOV EAX, ECX

Both operands use register mode. The contents of register **ECX** is copied to register EAX.

Before execution of the above instruction, the contents of ECX and EAX are:

ECX
50

EAX
00

After execution

ECX
50

EAX
50

Register indirect mode

The memory address of the operand is contained in one of the eight general purpose register specified in the instruction.

Instruction format:

Opcode	Register	[Register]

Example:

MOV EAX,[EBX]

The above instruction moves the contents of LOCATION specified by the register EBX into the register EAX.

Before execution of the above instruction, the contents of EAX and EBX are:

EAX
00

EBX
1000

20

1000

After execution

EAX

20

EBX

1000

20

1000

Base with displacement mode

An 8-bit or 32-bit signed displacement and one of the eight general purpose register to be used as a base register are specified in the instruction. The effective address of the operand in the sum of the contents of the base register and the displacement.

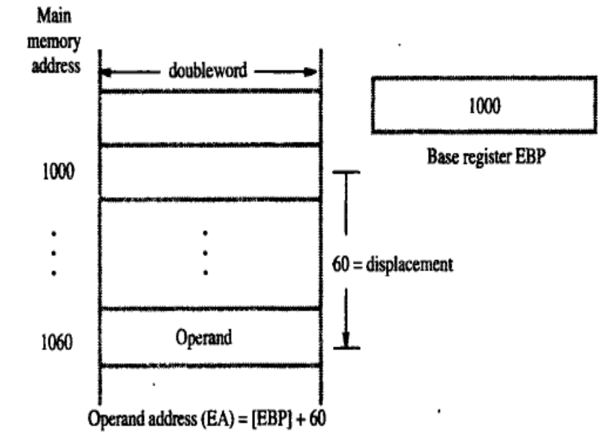
Instruction format:

Opcode	**Dest.Register**	**[Register]+Displacement**

Example:

MOV EAX, [EBP + 60]

The second operand uses base displacement mode. The instruction contains a constant. That constant is added to the contents of register EBP to form an effective address. The contents of memory at the effective address are copied into register EAX.



(a) Base with displacement mode, expressed as [EBP + 60]

Index with displacement mode

A 32-bit signed displacement, one of the eight general purpose register to be used as an index register ,and a scale factor of 1,2,4 or 8 are specified in the instruction. To obtain the effective address of the operand, the contents of the index register are multiplied by the scale factor and then added to the displacement.i.e

Offset = (Index * Scale) + displacement

Instruction format:

Opcode	**Dest.Register**	**[Register] * Scale factor +Displacement**

Example:

MOV AL,[EBP * 4 + 10]

Base with index mode

Two of the eight general purpose register and a scale factor of 1,2,4 or 8 are specified in the instruction. The registers are used as base and index register and the effective address of the operand is calculated as follows: the contents of the index register are multiplied by the scale factor and added to the contents of the base register.i.e

Offset = Base + (Index * Scale)

Instruction format:

Opcode	Dest.Register	[Register1] + [Register2] * Scale factor

Example:

MOV EAX, [ESP+ESI*4]

The contents of registers ESI is multiplied with the scale factor 4 and then the content of ESP is added to form an effective address. The contents of memory at the effective address are copied into register EAX

Base with index and displacement mode

An 8 bit or 32 –bit signed displacement two of the eight general purpose registers and a scale factor of 1,2,4 or 8 are specified in the instruction. The register is used as base and index register and the effective address of the operand is calculated as follows: the contents of the index register are multiplied by the scale factor and then added to the contents of the base register and the displacement.

An effective address is computed by:

Offset = Base + (Index * Scale) + displacement

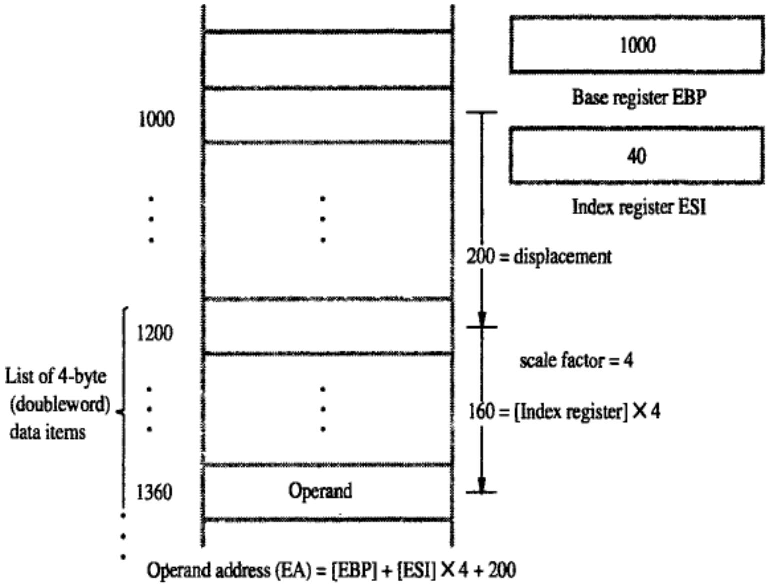
Instruction format:

Opcode	Dest.Register	[Register1] + [Register2] * Scale factor + Displacement

Example:

MOV EAX, [EBP+ESI*4 + 200]

The contents of registers ESI is multiplied with the scale factor 4 and then the content of EBP and displacement are added to form an effective address. The contents of memory at the effective address are copied into register EAX


1000
Base register EBP
40
Index register ESI
1000
200 = displacement
1200
scale factor = 4
List of 4-byte (doubleword) data items
160 = [Index register] X 4
1360
Operand
Operand address (EA) = [EBP] + [ESI] X 4 + 200

(b) Base with displacement and index mode, expressed as [EBP + ESI * 4 + 200]

The below table shows the various addressing modes of IA-32 processor:

S.No	Name	Assembler syntax	Addressing function
1	Immediate	Value	Operand = Valur
2	Direct	Location	EA = Location
3	Register	Reg	EA = Reg that is Operand=Reg
4	Register indirect	[Reg]	EA = [Reg]
5	Base with displacement	[Reg + Disp]	EA = [Reg] + Disp
6	Index with displacement	[Reg * S + Disp]	EA = [Reg] * S + Disp
7	Base with index	[Reg1 + Reg2 * S]	EA = [Reg1] + [Reg2] * S
8	Base with index and displacement	[Reg1 + Reg2 * S + Disp]	EA = [Reg1] + [Reg2] * S + Disp

Where,

Value	an 8 or 32 – bit signed number
Location	a 32-bit address
Reg,Reg1,Reg2	one of the general purpose registers EAX,EBX,ECX,EDX,ESP,EBP,ESI,EDI,with the exception that ESP cannot be used as an index register.
Disp	an 8 or 32 – bit signed number except that in the index with displacement mode it can only be 32 bits.
S	a scale factor of 1,2,4 or 8

3. Explain A-32 Instructions

IA-32 Instruction format

OP code	Addressing mode	Displacement	Immediate
1 or 2 bytes	1 or 2 bytes	1 or 4 bytes	1 or 4 bytes

The instructions are variable in length ,ranging from one byte to 12 bytes ,consisting of up to four fields .the OP-code field consists of one or two bytes, with most instructions requiring only one byte, the addressing mode information is contained in one or two bytes immediately following the OP code. For instructions that involve the use of only one register in generating the effective address of an operand ,only one byte is needed in the addressing mode field. Two bytes are needed for encoding the last two addressing modes. These modes use two registers to generate the effective address of a memory operand.

If a displacement value is needed in computing an effective address for a memory operand, it is encoded into either one or four bytes in a field that immediately follows the addressing mode field. If one of the operand is an immediate value, then it is placed in the last field of and instruction and it occupies either one or four bytes.

IA-32 instructions have either one or two operands. In the two –operand case, only one of the operands can be in the memory. The other must be in a process register. In addition to the usual instructions for moving data between the memory and the processor register, and for performing arithmetic operations, the instruction set includes a number of different logical and shift/rotate operations on data. Byte string instructions are included for nonnumeric data processing. Push and Pop operations for manipulating the processor stack are directly supported in the instruction set.

The instruction

ADD dst,src

Performs the operation

And $\text{dst} \leftarrow [\text{dst}]+[\text{src}]$

Performs the operation **MOV dst,src**

$$\text{dst} \leftarrow [\text{src}]$$

One byte instructions

Registers can be incremented or decremented by instructions,that occupy one boe byte.examples are

INC EDI

And

DEC ECX

In which the general purpose register EDI and ECX are specified by 3-bit codes in the single OP-code byte.

Immediate mode encoding

The OP-code specifies when the immediate addressing mode is used. For example the instruction

MOV EAX,820

Is encoded into 5 bytes .a one –byte OP code specifies the move operation, the fact that a 32-bit immediate operand is used and the name of the destination register. The OP code byte is directly followed by the 4-byte immediate value of 820.when an 8-bit immediate operand is used, as in the instruction

MOV DL,5

Only two bytes are needed to encode the instruction.

Addressing mode and displacement fields

As a general; rule, one operand of a two-operand instruction must be in a register .the other operand can also be in a register, or it can be in the memory. There are two exceptions where both operands can be in the memory. The first is the case where the source operand is an immediate operand, and the destination operand is in the memory. The second is the case of instruction for Push and Pop operations on the processor stack. The stack is located in the stack segment of memory, and it is possible to push a memory operand onto the stack or to pop an operand from the stack into the memory.

When both operands are in registers, only one addressing mode byte is needed. For example ,the instruction

ADD EAX,EDX

is encoded into two bytes. The first byte contains the OP code and the other byte is an addressing mode byte that specifies the two registers. The instruction

MOV ECX,N

is encoded in 6 bytes: one for the OP code, one for the addressing mode byte that specifies both the Direct mode and the destination register ECX,and four bytes for the address of memory location N.

The instruction

ADD EAX,[EBX+EDI*4]

requires two addressing mode bytes because two registers are used to generate the effective address of the source operand. The scale factor of 4 is also included in the second of these two bytes. Thus, the instruction requires a total of 3 bytes, including the OP-code byte.

In the encoding of two-operand instructions, the specification of the register operand and the memory operand are placed in a fixed order, with the register operand always, being specified first. In order to distinguish between the instructions

MOV EAX,LOCATION

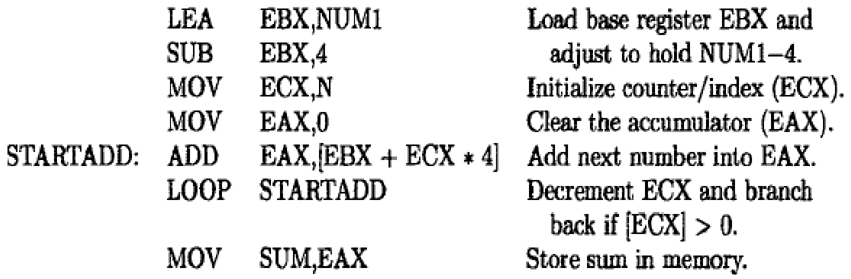
Which loads the contents of memory location LOCATION into register EAX ,and the instruction

MOV LOCATION,EAX

Which stores the contents of EAX into LOCATION ,the OP-code byte contains a bit called the direction bit. This bit indicates which operand is the source.

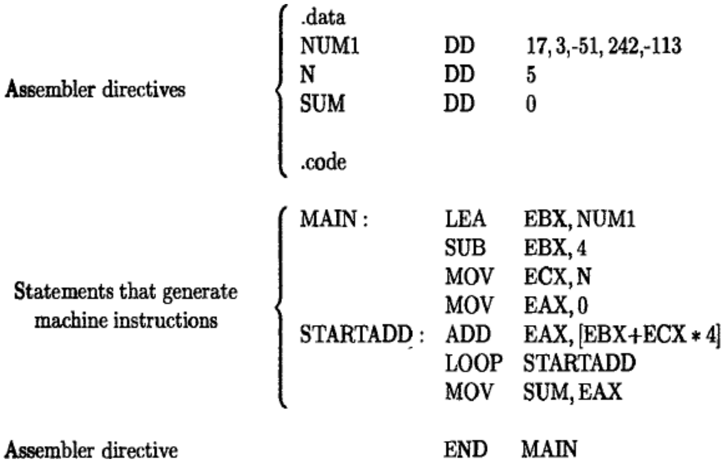
4. Write about IA-32 assembly language.

Basic aspects of the IA-32 assembly language for specifying OP code, addressing modes and instruction address labels are shown in the below program:



```
           LEA    EBX,NUM1             Load base register EBX and
           SUB    EBX,4                  adjust to hold NUM1−4.
           MOV    ECX,N                Initialize counter/index (ECX).
           MOV    EAX,0                Clear the accumulator (EAX).
STARTADD:  ADD    EAX,[EBX + ECX * 4]  Add next number into EAX.
           LOOP   STARTADD             Decrement ECX and branch
                                         back if [ECX] > 0.
           MOV    SUM,EAX              Store sum in memory.
```

The assembler directives are needed to define the data area of a program and to define the correspondence between symbolic names for data locations and the actual physical address values. A complete assembly language program is shown below:



```
                          .data
                          NUM1        DD     17,3,-51,242,-113
Assembler directives      N           DD     5
                          SUM         DD     0

                          .code

                          MAIN :      LEA    EBX, NUM1
                                      SUB    EBX, 4
Statements that generate              MOV    ECX, N
machine instructions                  MOV    EAX, 0
                          STARTADD :  ADD    EAX, [EBX+ECX * 4]
                                      LOOP   STARTADD
                                      MOV    SUM, EAX

Assembler directive                   END    MAIN
```

The data and code assembler directives define the beginning og the data and code sections of the program .In the data section ,the DD directives allocate 4-byte doublewords initialized to the decimal values 17,3,-51,242 and -113.the next two doubleword locations ,initialized to 5 and 0 ,are given the address labels N and SUM.

The 3 symbolic names declared in the data section are used in the addressing modes of the instruction in the code section. The MAIN label is used to specify the location where instruction execution is to begin, and this label is used in the END assembler directive that terminates the text file for the program.

5. Write in detail about IA-32 program flow control

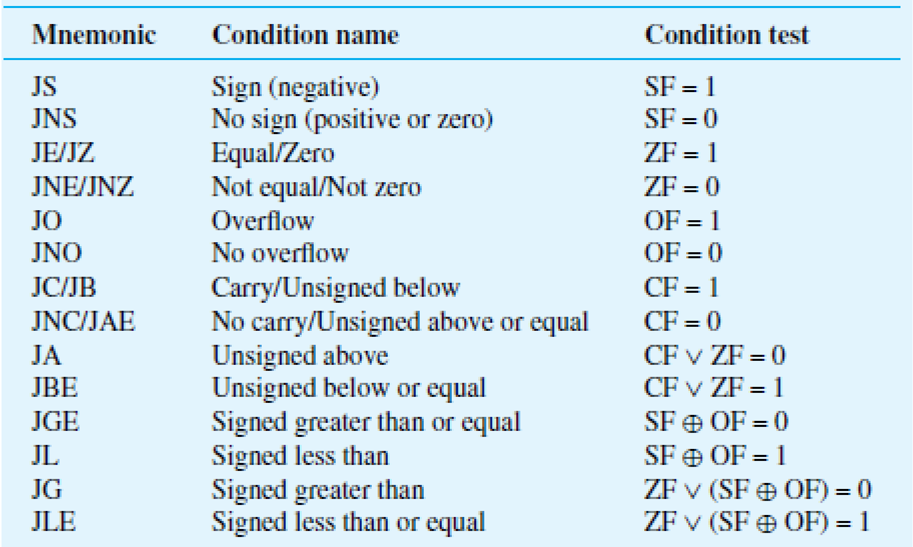
There are two main ways in which the flow of executing instructions varies from straight –line sequencing, calls to subroutines and returns from them break straight line sequencing. Also ,branch instructions, either conditional or unconditional, can cause a break. The branch instructions are called jumps.

Conditional jumps and condition code flags

The conditional Jump instructions test the four condition code flags in the status register. The instruction

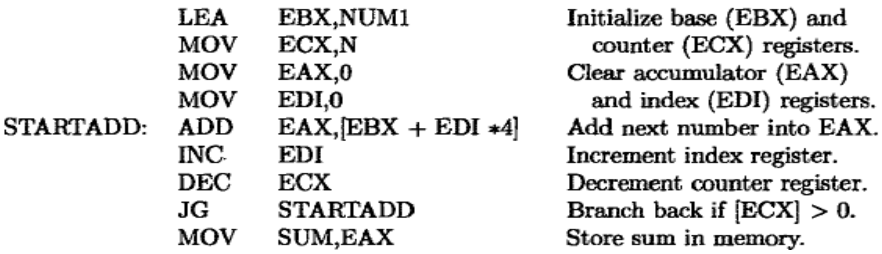
JG LABEL

is an example of a conditional Jump instruction. The condition is *greater-than* as indicated by the G suffix in the OP code. The below table summarizes the conditional Jump instructions and the corresponding combinations of the condition code flags that are tested. The Jump instructions that test the sign flag (SF) are used when the operands of a preceding arithmetic or comparison instruction are signed numbers. For example, the JG instruction tests for the greater-than condition when signed numbers are involved, and it considers the SF flag. When unsigned numbers are involved, the JA (jump-above) instruction tests for the greater than condition without considering the SF flag.



Mnemonic	Condition name	Condition test
JS	Sign (negative)	SF = 1
JNS	No sign (positive or zero)	SF = 0
JE/JZ	Equal/Zero	ZF = 1
JNE/JNZ	Not equal/Not zero	ZF = 0
JO	Overflow	OF = 1
JNO	No overflow	OF = 0
JC/JB	Carry/Unsigned below	CF = 1
JNC/JAE	No carry/Unsigned above or equal	CF = 0
JA	Unsigned above	$CF \vee ZF = 0$
JBE	Unsigned below or equal	$CF \vee ZF = 1$
JGE	Signed greater than or equal	$SF \oplus OF = 0$
JL	Signed less than	$SF \oplus OF = 1$
JG	Signed greater than	$ZF \vee (SF \oplus OF) = 0$
JLE	Signed less than or equal	$ZF \vee (SF \oplus OF) = 1$

For example, program for adding numbers



```
           LEA    EBX,NUM1            Initialize base (EBX) and
           MOV    ECX,N                 counter (ECX) registers.
           MOV    EAX,0               Clear accumulator (EAX)
           MOV    EDI,0                 and index (EDI) registers.
STARTADD:  ADD    EAX,[EBX + EDI *4]  Add next number into EAX.
           INC    EDI                 Increment index register.
           DEC    ECX                 Decrement counter register.
           JG     STARTADD            Branch back if [ECX] > 0.
           MOV    SUM,EAX             Store sum in memory.
```

the instruction

JG STARTADD

is an example of jump instruction. The condition is “Greater than 0”is tested ,if it is true then the control transfers to STARTADD otherwise control transferred to the next statement which is coming after JG STARTADD.

Unconditional Jump Instruction

An unconditional Jump instruction, JMP, causes a branch to the instruction at the target address. In addition to using short (one-byte) or long (four-byte) relative signed offsets to determine the target address, as is done in conditional Jump instructions, the JMP instruction also allows the use of other addressing modes. This flexibility in generating the target address can be very useful. Consider the Case statement that is found in many high-level languages. It is used to perform one of a number of alternative computations at some point in a program. Each of these alternatives is referred to as a case. Suppose that for each case, a routine is defined to perform the corresponding computation. Suppose also that the 4-byte starting addresses of the routines are stored in a table in the memory, beginning at a location labeled JUMPTABLE. The cases are numbered with indices 0, 1, 2, ... At execution time, the index of the selected case is loaded into index register ESI. A jump to the routine for the selected case is performed by executing the instruction

JMP [JUMPTABLE + ESI * 4]

which uses the Index with displacement addressing mode.

Compare instructions

It is often necessary to make conditional jumps in a program based on the results of comparing two numbers. The compare instruction

CMP dst,src

performs the operation

$$dst \leftarrow [src]$$

and sets the condition code flags based on the result obtained. Neither of the operands is changed .the first operand is always compared to the second. For example, the compare instruction by a conditional jump that is based on the “greater than” condition, then the jump will take to the target address if the destination operand is greater than the source operand.

6. Write in detail about logic and shift/rotate instructions

Logic operations:

The IA-32 architecture has instructions that perform the logic operations AND, OR, and XOR. The operation is performed bitwise on two operands, and the result is placed in the destination location.

For example, suppose register EAX contains the hexadecimal pattern 0000FFFF and register EBX contains the pattern 02FA62CA. The instruction

AND EBX, EAX

clears the left half of EBX to all zeroes, and leaves the right half unchanged. The result in EBX will be 000062CA.There is also a NOT instruction which generates the logical complement of all bits of the operand, that is, it changes all 1s to 0s and all 0s to 1s.

Shift instruction:

An operand can be shifted right or left ,using either logical or arithmetic shifts,by a number of bit positions determined by a specified count. The format of the shift instruction is

OPcode dst , count

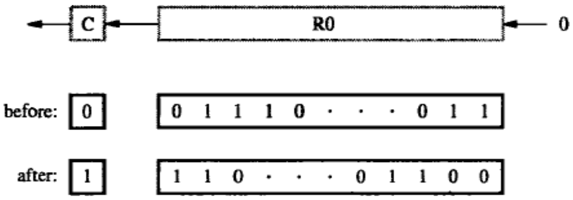
Where the destination operand to be shifted is specified by the general addressing modes and the count is given either as an 8-bit immediate value or is contained in the 8-bit register CL.

There are four shift instructions:

- SHL (Shift left logical)
- SHR (Shift right logical)
- SAL (Shift left arithmetic; operation is identical to SHL)
- SAR (Shift right arithmetic)

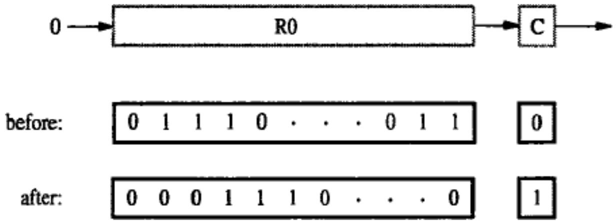
1. SHL (Shift left logical)

The SHL (shift left) instruction performs a logical left shift on the destination operand, filling the lowest bit with 0.For example, **SHL R0,#2**



2. SHR (Shift right logical)

The SHR (shift right) instruction performs a logical right shift on the destination operand. The highest bit position is filled with a zero. For example, **SHR R0,#2**

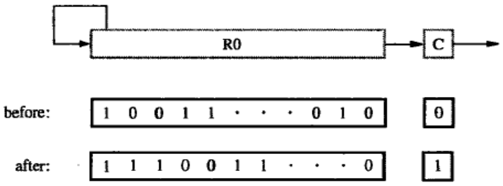


3. SAL (Shift left arithmetic)

The operation of SAL is identical to SHL.

4. SAR (Shift right arithmetic)

SAR (shift arithmetic right) performs a right arithmetic shift on the destination operand. For example, **SAR R0,#2**



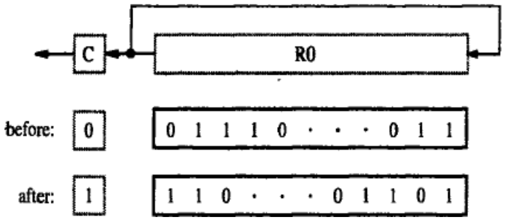
Rotate instruction:

In addition to the shift instructions, there are also four rotate instructions:

- ROL (Rotate left without the carry flag CF)
- ROR (Rotate right without the carry flag CF)
- RCL (Rotate left including the carry flag CF)
- RCR (Rotate right including the carry flag CF)

1. ROL (Rotate left without the carry flag CF)

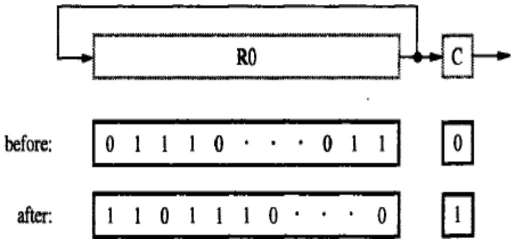
- ROL (rotate) shifts each bit to the left
- The highest bit is copied into both the Carry flag and into the lowest bit
- No bits are lost
- For example , **ROL R0,#2**



2. ROR (Rotate right without the carry flag CF)

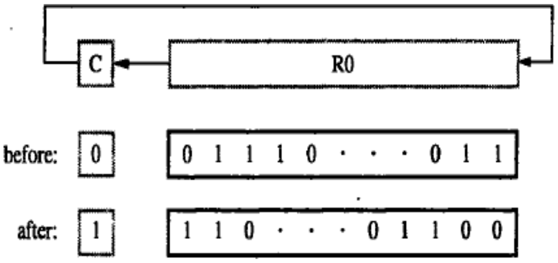
- ROR (rotate right) shifts each bit to the right

- The lowest bit is copied into both the Carry flag and into the highest bit
- No bits are lost
- For example , **ROR R0,#2**



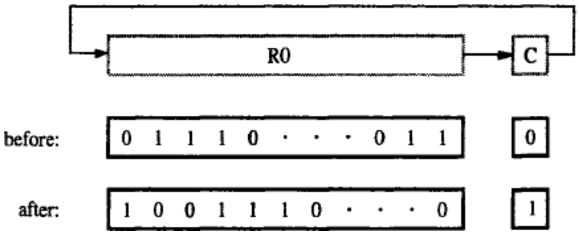
3. RCL (Rotate left including the carry flag CF)

- RCL (rotate carry left) shifts each bit to the left
- Copies the Carry flag to the least significant bit
- Copies the most significant bit to the Carry flag
- For example , **RCL R0,#2**



4. RCR (Rotate right including the carry flag CF)

- RCR (rotate carry right) shifts each bit to the right
- Copies the Carry flag to the most significant bit
- Copies the least significant bit to the Carry flag
- For example , **ROR R0,#2**



The rotate instructions require the count argument to be either an 8-bit immediate value or the 8-bit contents of register CL.

7. Explain I/O Operation in IA-32

Memory mapped I/O

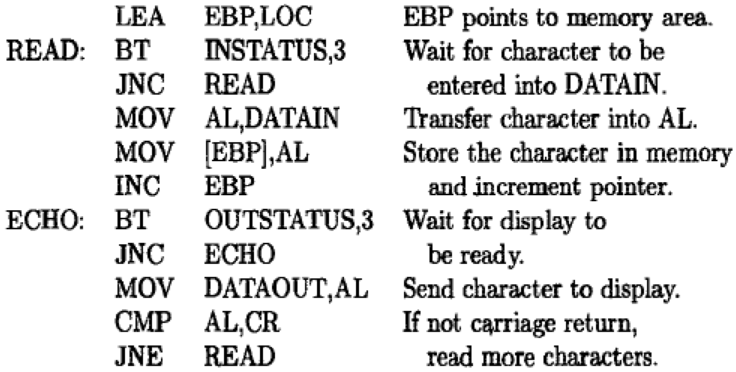
The IA-32 Move instruction can be used to transfer directives to I/O devices ,and to transfer data and status information to and from devices. For example, suppose that keyboard and display devices have their synchronization flags SIN and SOUT stored in bit 3 of device status registers INSTATUS and OUTSTATUS, respectively. Using program controlled I/O; a byte can be read from the keyboard buffer register DATAIN into register AL using the wait loop

```
READWAIT:     BT INSTATUS,3
              INC READWAIT
              MOV ALDATAIN
```

The instruction BT is a bit-test instruction .the value in the destination bit position specified by the source operand is loaded into the carry flag CF .the conditional jump JNC causes a jump to READWAIT if CF = 0. Similarly, an output operation to send a byte from register AL to the display buffer register DATAOUT is performed by

```
WRITEWAIT:    BT OUTSTATUS,3
              INC WRITEWAIT
              MOV DATAOUT,AL
```

An IA-32 program that reads one line of character from a keyboard, stores them in memory starting at address LOC ,and echo's them back out to the display shown below



```
        LEA   EBP,LOC         EBP points to memory area.
READ:   BT    INSTATUS,3      Wait for character to be
        JNC   READ              entered into DATAIN.
        MOV   AL,DATAIN       Transfer character into AL.
        MOV   [EBP],AL        Store the character in memory
        INC   EBP               and increment pointer.
ECHO:   BT    OUTSTATUS,3     Wait for display to
        JNC   ECHO              be ready.
        MOV   DATAOUT,AL      Send character to display.
        CMP   AL,CR           If not carriage return,
        JNE   READ              read more characters.
```

Isolated I/O

The IA-32 instructions set also have two instructions, with OP codes IN and OUT, that are used only for I/O purposes. The addresses issued by these instructions are in an address space that is separate from the memory address space used by the other instructions. This arrangement is called isolated I/O to distinguish it from memory –mapped I/O in which the addressable locations in I/O devices are in the same address space as memory locations. The same address and data lines on Intel processor chips are used for both address spaces. An output control line is used to indicate which address space is reference by the current instruction.

The 16-bit addresses are used in the byte-addressable I/O address space. There are 8-bit and 32-bit I/O device operand location that hold data ,status, and command information. The first 256 addresses can be specified directly using an 8-bit field in the In and Out instructions. The format for the input instruction using this mode is

IN REG ,DEVADDR

Where the destination REG must be register AL or EAX,denoting an 8-bit or a 32-bit operand transfer respectively. The last field in the instruction contains the 8-bit device address DEVADDR.the corresponding output instruction is

OUT DEVADDR,REG

Since the address space is byte addressable, a keyboard device that can send an 8-bit ASCII character to the processor could have its data buffer register at byte address DEVADDR and its 8-bit starts register at address

DEVADDR + 1

The full 16-bit I/O address spans 64K locations; it can be referenced through the DX register using the input instruction

IN REG,DX

Where, as before,REG must be AL or EAX.The 16-bit device address is contained in the DX register ,which is low order 16 bits of the EDX register, and the width of the data transfer is determined by the size of the REG operand. The corresponding output instruction os

OUT DX,REG

Block transfers

In addition to the instruction IN and OUT that transfer a single item of information between the processor and an I/O device ,the IA-32 architecture also has two block transfer I/O instructions REPINS and REPOUTS .they transfer a block lf data serially, one item at a time, between the memory and an I/O device. The S suffix in the OP codes stands for string, and the REP prefix stands for “repeat the item by item transfer until the complex block has been transferred”. The instructions themselves do not specify the parameters needed to describe the transfer. These parameters are specified implicitly by processor registers DX,EDI and ECX as follows:

DX contains a 16-bit I/O device address

EDI contains a 32-bit address for the beginning of a block in memory

ECX contains the number of data items to be transferred.

A suffix B or D in the OP-code mnemonic indicates that the item size is either of byte or doubleword length. Thus REPINSB is a byte –block transfer, and REPINSD is a doubleword –block transfer.

The block transfer instruction operates as follows :After each data item is transferred ,the index register EDI is incremented by 1 or 4 depending on the size of the data items, and the ECX register is decremented by 1.The transfers are repeated until the contents of the counter register ECX have been decremented to 0.The effect of these single instruction is equivalent to a program loop that uses register ECX as the loop counter.

As an example, suppose that a block lf 128 doublewords is to be transferred from a disk storage device into the memory. Assume that the addressable data buffer register in the disk device contains a doubleword data item, and has the I/O address MEMBLOCK.The counter register ECX has to be initialized to 128.The instruction sequence

```
LEA        EDI,MEMBLOCK
MOV        DX,DISKDATA
MOV        ECX,128
REPINSD
```

can be used to accomplish the transfer. This assumes that MEMBLOCK has been defined as an address label, and DISKDATA has been defined by an EQU assembler directive to represent the 16-bit address of the device data buffer register.

8. Write in detail about Subroutines in IA-32Subroutines:

In the IA-32 architecture, register ESP is used as the stack pointer. It points to the current top element (TOS) in the processor stack. The stack grows toward lower numbered addresses. The width of the stack is 32 bits, that is, all stack entries are doublewords.

There are two instructions for pushing and popping individual elements onto and off the stack. The instruction

PUSH src

decrements ESP by 4, and then stores the doubleword at location src into the memory location pointed to by ESP. The instruction

POP dst

reverses this process by retrieving the TOS doubleword from the location pointed to by ESP,storing it at location dst, and then incrementing ESP by 4. These instructions implicitly use ESP as the stack pointer. The source and destination operands are specified using the IA-32 addressing modes.

There are also two more instructions that push or pop the contents of multiple registers. The instruction

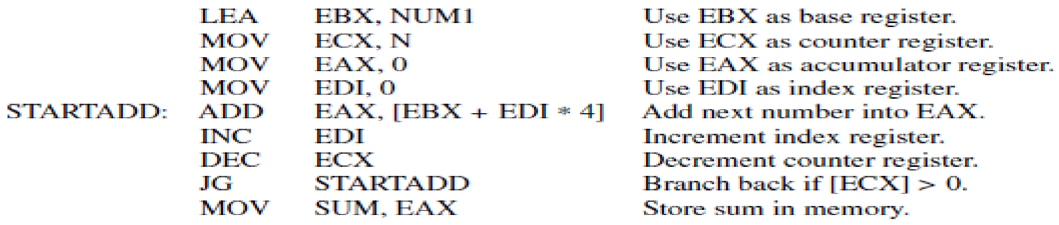
PUSHAD

pushes the contents of all eight general-purpose registers EAX through EDI onto the stack,and the instruction

POPAD

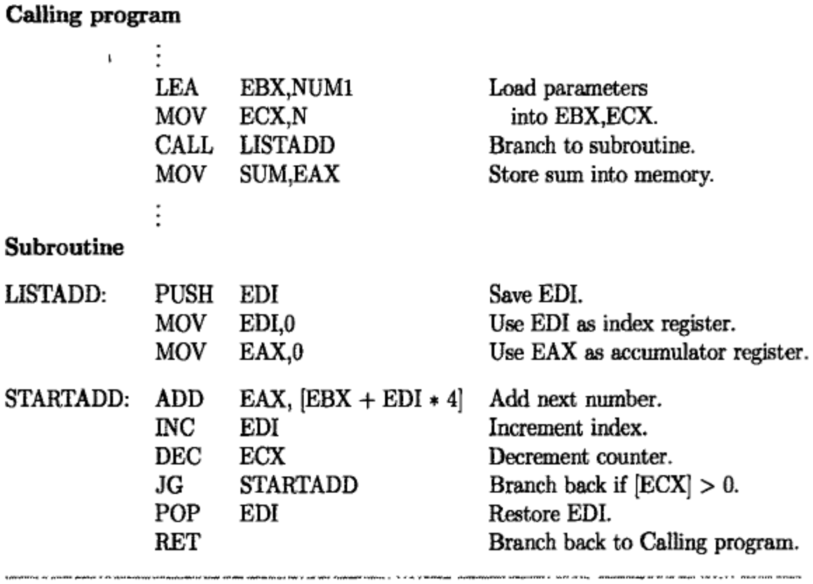
pops them off in the reverse order. When POPAD reaches the old stored value of ESP, it discards those four bytes without loading them into ESP and continues to pop the remaining values into their respective registers. These two instructions are used to efficiently save and restore the contents of all registers as part of implementing subroutines.

For example, The list-addition program shown below



```
           LEA   EBX, NUM1          Use EBX as base register.
           MOV   ECX, N             Use ECX as counter register.
           MOV   EAX, 0             Use EAX as accumulator register.
           MOV   EDI, 0             Use EDI as index register.
STARTADD:  ADD   EAX, [EBX + EDI * 4]  Add next number into EAX.
           INC   EDI                Increment index register.
           DEC   ECX                Decrement counter register.
           JG    STARTADD           Branch back if [ECX] > 0.
           MOV   SUM, EAX           Store sum in memory.
```

can be written as a subroutine as shown below:



```
Calling program
           :
           LEA   EBX,NUM1              Load parameters
           MOV   ECX,N                   into EBX,ECX.
           CALL  LISTADD               Branch to subroutine.
           MOV   SUM,EAX               Store sum into memory.
           :
Subroutine

LISTADD:   PUSH  EDI                   Save EDI.
           MOV   EDI,0                 Use EDI as index register.
           MOV   EAX,0                 Use EAX as accumulator register.

STARTADD:  ADD   EAX, [EBX + EDI * 4]  Add next number.
           INC   EDI                   Increment index.
           DEC   ECX                   Decrement counter.
           JG    STARTADD              Branch back if [ECX] > 0.
           POP   EDI                   Restore EDI.
           RET                         Branch back to Calling program.
```

(a) Calling program and subroutine

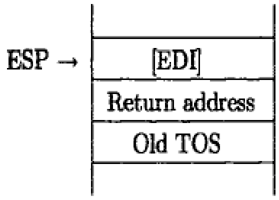
Parameters are passed through registers. Memory address NUM1 of the first number in the list is loaded into register EBX by the calling program.

The number of entries in the list, contained in memory location N, is loaded into register ECX. The calling program expects to get the final sum passed back to it in register EAX. Thus, registers EBX,ECX, and EAX are used for passing parameters.

Register EDI is used by the subroutine as an index register in performing the addition, so its contents have to be saved and restored in the subroutine by PUSH and POP instructions.
The subroutine is called by the instruction

CALL LISTADD

which first pushes the return address onto the stack and then jumps to LISTADD. The return address is the address of the MOV instruction that immediately follows the CALL instruction. The subroutine saves the contents of register EDI on the stack.



(b) Stack contents after saving EDI in subroutine

The above figure shows the stack contents at this point. After executing the loop, the saved contents of register EDI are restored. The instruction RET returns execution control to the calling program by popping the TOS element into the Instruction Pointer (register EIP).

9. Write about the other instructions that are available in IA-32

The other instructions available in IA-32 are:

5. Multiplication instruction:
6. Division instruction:
7. Multimedia Extension (MMX) instructions
8. Vector (SIMD) Floating-Point Operations

1. Multiplication instruction:

The signed integer multiplication instruction, IMUL, performs 32-bit multiplication. Depending on the form of the instruction that is used, the destination may be implicit and the 64-bit product may be truncated to 32 bits.

One form of this instruction is

IMUL src

which implicitly uses the EAX register as the multiplicand. The multiplier specified by src can be in a register or in the memory. The full 64-bit product is placed in registers EDX (high-order half) and EAX (low-order half).

A second form of this instruction is

IMUL REG, src

The destination operand, REG, must be one of the eight general-purpose registers. The source operand can be in a register or in the memory. The product is truncated to 32 bits before it is placed in the destination register REG.

For both forms, the CF and OF flags are set if there are any 1s (including sign bits) in the high-order half of the 64-bit product. Otherwise, the CF and OF flags are cleared. The other flags are undefined.

2. Division instruction:

The integer divide instruction, IDIV, operates on a 64-bit dividend and a 32-bit divisor to generate a 32-bit quotient and a 32-bit remainder. The format of the instruction is

IDIV src

The source operand is the divisor. The 64-bit dividend is formed by the contents of register EDX (high-order half) and register EAX (low-order half). After performing the division, the quotient is placed in EAX and the remainder is placed in EDX.

All of the condition code flags are undefined. Division by zero causes an exception. If the dividend value is represented by 32 bits, it must first be placed in EAX, and then sign-extended to the required 64-bit operand size in registers EAX and EDX. This is done by the instruction CDQ (convert doubleword to uadword), which has no operands because the source and destination are implicitly registers EAX and EDX, respectively

3. Multimedia Extension (MMX) instructions

A two-dimensional graphic or video image can be represented by a large array of sampled image points, called *pixels*. The color and brightness of each point can be encoded into an 8-bit data item. Processing of such data has two main characteristics.

The first is that manipulations of individual pixels often involve very simple arithmetic or logic operations.

The second is that very high computational performance is needed for some real-time display applications. The same characteristics apply to sampled audio signals or speech processing, where a sequence of signed numbers represents samples of a continuous analog
signal taken at periodic intervals.

In such applications, processing efficiency is achieved if the individual data items, which are usually bytes or 16-bit words, are packed into small groups whose elements can
be processed in parallel.

The IA-32 instruction set has a number of instructions that operate in parallel on such data packed into 64-bit quadwords.(A quadword contains 8 bytes or four 16-bit words).These instructions are called *multimedia extension* (MMX) instructions.

The operands for MMX instructions can be in the memory, or in the eight floating-point registers. Thus, these registers serve a dual purpose. They can hold either floating-point numbers or MMX operands. When used by MMX instructions, the registers are referred to as MM0 through MM7.

The MOVE instruction is provided for transferring 64-bit quadword operands between the memory and the MMX registers. The instruction

PADDB MMi,src

Adds the corresponding bytes of two 8-byte operands individually and places eight sums in the destination register. The source can be in the memory or in an MMX register but the destination must be an MMX register. The instruction

PADDB MM2, [EBX]

adds eight corresponding bytes of the quadwords in register MM2 and in the memory location pointed to by register EBX. The eight sums are computed in parallel. The results are placed in register MM2.

4. Vector (SIMD) Floating-Point Operations

A set of instructions that are used to perform arithmetic operations on small group of floating point numbers is provided.SIMD (single-instruction-multiple-data) instructions are useful for vector and matrix calculations in scientific applications.

In Intel terminology, these instructions are called *streaming SIMD extension* (SSE) instructions. They handle packed 128-bit double quadwords, each consisting of four 32-bit floating-point numbers. Eight additional 128-bit registers, XMM0 to XMM7, are available for holding these operands.

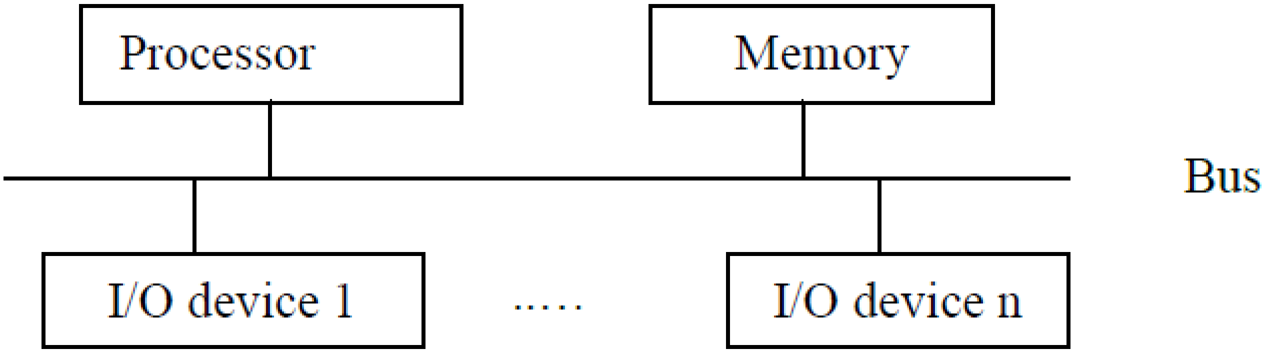
Add and multiply are two of the basic instructions are provided in this group .they operate on the four corresponding pairs of 32-bit operands in the 128-bit compound source and destination operands and place the four individual results in the 128-bit destination location.

UNIT – III

INPUT/OUTPUT ORGANIZATION: Accessing I/O Devices, Interrupts, Interrupt Hardware, Enabling and Disabling Interrupts, Handling Multiple Devices, Controlling Device Requests, Exceptions, Use Of Interrupts in Operating Systems, Pentium Interrupt Structure, Direct Memory Access, Busses, Interface Circuits, Standard I/O Interfaces.

2 Marks

1. Give the organization of single bus structure?



2. What is memory mapped I/O?

With Memory mapped I/O, any machine instruction that can access memory can be used for transfer data to or from an I/O device.

3. What is program controlled I/O?

In program controlled I/O, the processor repeatedly checks a status flag to achieve the required synchronization between the processor and an input and output device.

4. What are the various mechanisms for implementing I/O operations?

1. Program controlled I/O
2. Interrupts
3. Memory mapped I/O
4. DMA

5. Define ISR

ISR is nothing but interrupt service routine. It can handle the execution of the interrupts and it responses to an interrupt request.

6. What constitute the device’s interface circuit?

The address decoder, the data and status register and the control circuitry required to co-ordinate I/O transfers constitute the device"s interface circuit.

7. What is interrupt service routine?

The routine executed in response to an interrupt request is called as interrupt service routine. In short , it is called as ISR.

8. What is the purpose of interrupt acknowledgement signal?

The interrupt acknowledgement signal is used by the processor to inform the device that is request has been recognized so that it may remove its interrupt request signal.

9. Define interrupt latency?

The delay between the time an interrupt request is received and the start of execution of the interrupt service routine is called interrupt latency.

10. What is real time processing?

The concept of interrupts is used in many control applications where processing of certain routines must be accurately timed relative to external events. This type of application is called as real time processing?

11. What are Special gates used for driving INTR line?

The special gates used for driving INTR line are,

1.Open collector

2.Open drain

12. When is an interrupt line said to be edge – triggered?

An interrupt line is said to be edge triggered if the interrupt handling circuit responds only to the leading edge of the signal.

13. Give a typical scenario assuming that interrupts are enabled?

- The device raises an interrupt request.
- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits in the bus.
- The device is informed that its request has been recognized and in response, it activates the interrupt request signal.
- The action requested by the Interrupt is performed by the ISR.
- Interrupts are enabled and execution of the interrupted program is resumed.

14. What are vectored interrupts?

To reduce the time involved in the polling process, a device requesting an interrupt may identify itself directly to the processor. Then the processor can immediately start executing the corresponding ISR. The term vectored interrupts refer to all interrupt handling schemes based on this approach.

15. What is interrupt vector?

Interrupt vector is the starting address of the interrupt service routine stored in the location pointed by the interrupting device.

16. What are privileged instructions?

Privileged instructions are the instructions which are executed only while the processor is running in the supervisor mode.

17. What is privilege exception?

An attempt to execute a privileged instruction while in the user mode leads to a special type of interrupt called a privilege exception.

18. What are the two independent mechanisms for controlling interrupt requests?

To control interrupt requests ,the mechanisms used are ,

1. At the device end , an interrupt enable bit in a control register determines whether the device is allowed to generate an interrupt request.

2. At the processor end, either an interrupt enable bit in the PS register or a priority structure determines whether a given interrupt request will be accepted.

19. What are exceptions? Give an Example?

An except is a term often used to refer to any event that cause an interruption.

Eg. I/O interrupts.

20. What is a debugger?

A debugger is a program used by system software which helps the programmer finds errors in a program.

21. What are the two facilities provided by a debugger?

The facilities provided by a debugger are.

- 1.Trace
- 2.Breakpoints.

22. What does an exception occur when the processor is in trace mode?

When the processor is operating in the trace mode, an exception occurs after execution of every instruction, using the debugging program as the exception service routine. The trace exception is disabled during the execution of debugging program.

23. What are the uses of interrupts in OS?

The uses of interrupts in OS are,

1. To assign priorities
2. Switch from one user program to another.
3. Implementing security.
4. Protection features.
5. Co-ordinate I/O activities.

24. What is the process?

A program together with any information that describes current state of execution is regarded by the OS as an entity called process.

25. Define Multitasking?

Multitasking is a mode of operation in which a processor executes several user programs at the same time.

26. What is time slicing?

Time slicing is a common OS technique that makes multitasking possible. With this technique, each program runs for a short time period called as a time slice, then another program runs for its time slice and so on. The period is determined by continuously running hardware clock, which generates an interrupt every second.

27. What are the three states of a process?

A process can be in one of the three possible states.

1.Running,

2.Runnable.

3.Blocked.

28. Differentiate a process in running and runnable state?

The running state means that the program is currently being executed .The process is runnable if the program is ready for execution but is waiting to be selected by the scheduler.

29. What is program state?

A program state is state which includes register contents, program counter and the program status word.

30. What is DMA? Write the advantages of DMA.

A special control unit that may be provided to allow transfer of a block of data directly between an external device and the main memory without continuous intervention by the processor. This approach is called Direct memory access.

- Computer system performance is improved by direct transfer of data between memory and I/O devices, bypassing the CPU.
- CPU is free to perform operations that do not use system buses.

31. What is the purpose of a DMA controller?

The DMA controller performs the functions that would normally be carried out by the processor when accessing the main memory.

32. What is cycle stealing?

Cycle stealing is an interweaving technique used by DMA controller to steal the memory cycles from the processor.

33. What is a block or burst mode?

The DMA controller may be given exclusive to the main memory to transfer a block of data without interruption. This is known as block or burst mode.

34. What is bus master?

The device that is allowed to initiate data transfers on the bus at any given time is called bus master.

35. What is bus arbitration?

Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.

36. Name the two approaches to bus arbitration.

The approaches to bus arbitration are,

1. Centralized arbitration
2. Distributed arbitration.

37. What do you mean by distributed arbitration?

Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.

38. What is the purpose of a bus protocol?

A bus protocol is the set of rules that governs the behavior of various devices connected to the bus.

39. Define master?

Master is a device that initiates data transfer by issuing read or write commands on the bus. Master is also called as initiator.

39. What is a slave?

The device addressed by the master is called as slave. Slave can also called as target.

40. What is a synchronous bus?

In synchronous bus, all devices derive timing information from the common clock line. Equally spaced pulses on this define equal time intervals.

41. What is an asynchronous bus?

In asynchronous bus, controlling data transfer on the bus is based on the use of handshake between the master and slave.

42. Define full handshake. List the two advantages of a full handshake.

An asynchronous logic uses hardware handshaking to exchange information on the bus. Here the clock line is eliminated by two control signals master ready and slave ready.

The advantages of a full handshake are,

1. Highest degree of flexibility is provided.
2. Highest degree of reliability is provided.

43. What are the main advantages of asynchronous bus?

The main advantage of an asynchronous bus is that the handshake process eliminates the need for synchronization of the sender and the receiver blocks, thus simplifying timing design.

44. What is a port?

The side opposite to bus signals in an I/O interface consists of data path with its associated controls to transfer data between the interface and the I/O device. This side called a port.

45. What is the difference between serial port and parallel port?

1. A parallel port transfers data in the form of a number of bits typically 8 or 16 simultaneously to or from the device. A serial port transmits and receives data one bit at a time.

2. Parallel format is suitable for devices that are physically close to the computer. Serial format is much more convenient and cost-effective where longer cables are needed.

46. What is a bridge?

A bridge is an interconnection circuit between two buses. It translates the signals and protocols of one bus into those of the other.

47. What is transaction?

A complete transfer operation on the bus , involving an address and a burst of data is called a transaction.

48. Define SCSI? Or. What is SCSI?

SCSI stands for **Small Computer System Interface**. SCSI refers to the standard bus which is defined by ANSI (American National Standard Institute). under designation X3.131.

SCSI bus the several options. It may be,

- **Narrow bus** (It has 8 data lines transfer 1 byte at a time.)
- **Wide bus (i**t has 16 data lines & transfer 2 byte at a time.)
- **Single-Ended Transmission (**Each signal uses separate wire.)
- **HVD (High Voltage Differential) (** it was a 5v (TTL Cells))
- **LVD (Low Voltage Differential) (**it uses 3.3v)

49. What are the different categories of SCSI bus signals?

SCSI bus signals are classified as,

1.Data signal

2.Phase signal

3.Information signal

4.Handshake.

5.Direction of transfer.

50. What are the objectives of USB?

The objectives of USB are as follows,

1.Provide a simple ,low cost and easy to use interconnection system.

2.Enhance user convenience through a „plug and play‟ mode operation.

51. What is isochronous? (or) What does isochronous data stream means?

An isochronous data stream means that the successive events are separated by equal periods of time.

52. What is hub?

A hub is the intermediate control point between the host and the I/O device.

53. Define serial port.

A serial port transmits and receives data one word bit at a time.

54. What is meant by bus arbitration?

It is the process by which the next device to become the bus master is selected and the bus mastership is transferred to it.

Types:

There are 2 approaches to bus arbitration. They are,

- Centralized arbitration
- Distributed arbitration

55. Name and give purpose of widely used bus standards?

- **PCI** defines an expansion bus on the motherboard.
- **SCSI** and **USB** are used for connecting additional devices both inside and outside the computer box.
- **SCSI** bus is a high speed parallel bus intended for devices such as disk and video display.
- **USB** uses a serial transmission to suit the needs of equipment ranging from keyboard to game control to internal connection

56. What is distributed arbitration?

Distributed Arbitration:

It means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process.

57. What is interrupt?

- When a program enters a wait loop, it will repeatedly check the device status. During this period, the processor will not perform any function.
- The Interrupt request line will send a hardware signal called the interrupt signal to the processor.
- On receiving this signal, the processor will perform the useful function during the waiting period.

58. Define DMA? (or) why do we need DMA?

To allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor, a special control unit is provided. This approach is called Direct access memory or DMA.

59. How interrupt request from multiple devices can be handled?

An I/O Device Request An Interrupt By Activating A Bus Line Called Interrupt Request

- SINGLE LEVEL
- MULTI LEVEL

60. What are the components of I/O interface?

. An I/O interface consists of the circuitry required to connect an I/O device to a computer bus. On one side of the interface are the bus signals for address, data and control. On the other side is a data path with its associated controls to transfer data between the interface and the I/O device. This side is called a port. This may be either a serial port or parallel port.

61. Mention the advantage of USB bus.

USB helps to add many devices to a computer system at any time without opening the computer box.

62. What are the i/o data transfer method using memory busses?

Three methods used for data transfer between I/O devices and CPU

- program i/o or polling
- interrupt driven i/o
- direct memory access

63. Distinguish between isolated and memory mapped I/O? (Nov 12)

The isolated I/O method

- Isolates memory and I/O addresses so that memory address values are not affected by interface address assignment since each has its own address space.
- This is the case in computers that employ only one set of read and write signals and do not distinguish between memory and I/O addresses.
- The assigned addresses for interface registers cannot be used for memory words, which reduce the memory address (range available).

The memory mapped I/O

- uses the same address space for both memory and I/O.
- The computer treats an interface register as being part of the memory system.
- In memory mapped I/O organization, there are no specific inputs or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words.

64. Explain handshaking with respect to data transfer.

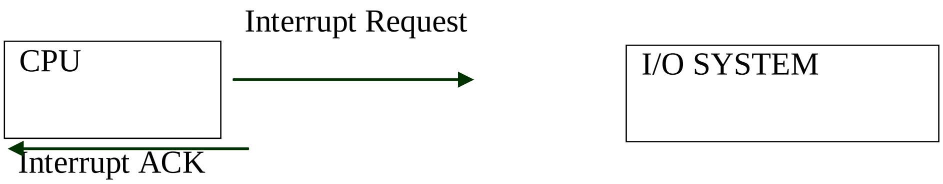
Some signals are exchanged between the processor and processor and peripheral prior to the data transfer. This is called handshake signals. Data transfer on the on the bus is based on the use of handshake between the master and the slave

65. What are the functions of a typical I/O interface?

When the CPU issues an I/O command. The command, the command contains address of the device, according to the address, the device is selected. The CPU reads data through buffer from I/P device or writes data through latch to output device addressed

66. How does the processor handle an interrupt request?

The processor instruct an I/O device interface to activate the interrupt-request line as soon as it is ready for a data transfer, then it performs other useful work. It transfer the request to interrupt service routine, which handles the task.



67. Specify the different I/O transfer mechanisms available.

They are two different I/O transfer

- Direct access memory or DMA.
- Bus arbitration

To allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor, a special control unit is used called DMA.

Bus arbitration is the process by which the next device to become the bus master is selected and the bus mastership is transferred to it.

Types:

There are 2 approaches to bus arbitration. They are,

- Centralized arbitration
- Distributed arbitration

68. What are priority groups?

When interrupt requests arrive from two (or) more devices simultaneously, the processor has to decide which request should be serviced first, and which one should be delayed. Devices are organized in groups and each group is connected at priority level

69. What are the operating system routines of a keyboard driver?

Clear keybuf: clears the keyboard buffer

Install keyboard: installs the allegro keyboards interrupt controller

70. How is the interrupt request from a device handled?

When the device interrupt the processor, CPU suspends the current program execution and branches into an **Interrupt Service Routine** (ISR).After execution of ISR, the CPU return back to the interrupted program

71. Why does the DMA gets priority over CPU when both request memory Transfer? (or) state the important of DMA in networks

- A DMA controller connects high speed network to the computer bus
- The disk controller which controls two disks, also has DMA capability and provides two DMA channels

72. Why does DMA have priority over the CPU when both request a memory transfer?

The data transfer is monitored by DMA controller which is known as DMA channel. The CPU is involved only at the beginning and end of the transfer.

When the CPU wishes to read or write a block a data, it issues a command to the DMA channel by sending read/write operation, address of I/O, number of words to be read or written. Hence DMA have priority over the CPU when both request a memory transfer.

73. What is the advantage of using interrupt initiated data transfer over transfer under program control without interrupt?

In the interrupt initiated data transfer, the processor identifies the request and transfer the control Interrupt Service Routine(ISR) to perform the task and its resumes back with the useful task where as, the processor has to waste its time by performing all the task.

74. What is the difference between subroutine and interrupt service routine.

Subroutine: subroutine or the subprogram is the routine which could be called by another subroutine or main routine under program control

Interrupt Service Routine (ISR): ISR is called automatically on the occurrence of an interrupt which is predefined

75. What factors influences the bus design decision?

A bus is a set of lines (wires) designed to transfer all bits of a word from a specified source to a specified destination. Protocol defines certain set of rules that give on the behavior of various devices connected to the bus.

76. Write the advantages of USB.

1. Simple connectivity:
2. Simple cables:
3. One interface for many devices:
4. Automatic configuration:
5. No user setting:

11 Marks

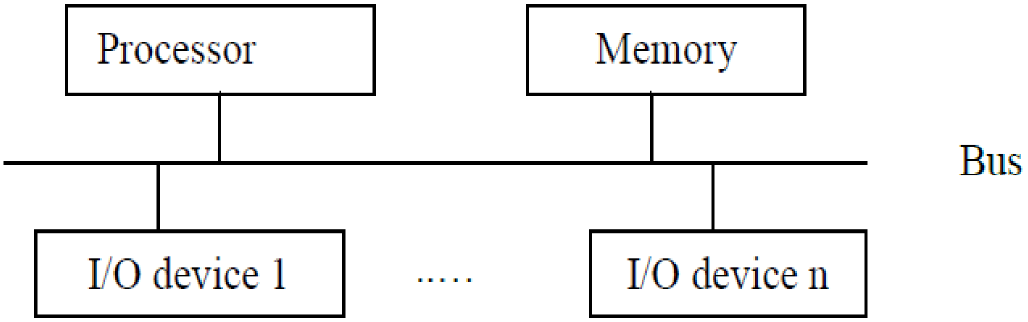
1. Explain about Programmed I/O And Memory Mapped I/O. (Nov 12)

A simple arrangement to connect I/O devices to a computer is to use a single bus structure. It consists of three sets of lines to carry

- Address
- Data
- Control Signals.

- When the processor places a particular address on address lines, the devices that recognize this address responds to the command issued on the control lines.
- The processor request either a read or write operation and the requested data are transferred over the data lines.
- When I/O devices & memory share the same address space, the arrangement is called memory mapped I/O.

Single Bus Structure



Eg:-
Move DATAIN, Ro - Reads the data from DATAIN then into processor register Ro.
Move Ro, DATAOUT - Send the contents of register Ro to location DATAOUT.
DATAIN - Input buffer associated with keyboard.
DATAOUT - Output data buffer of a display unit / printer.

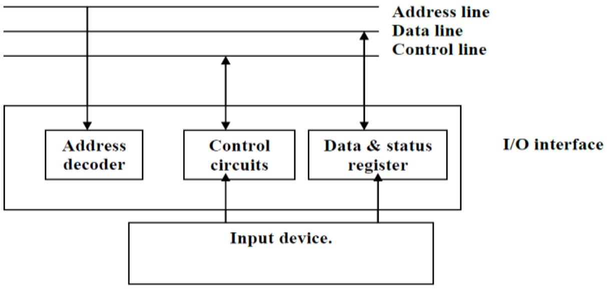


Fig: I/O Interface for an Input Device

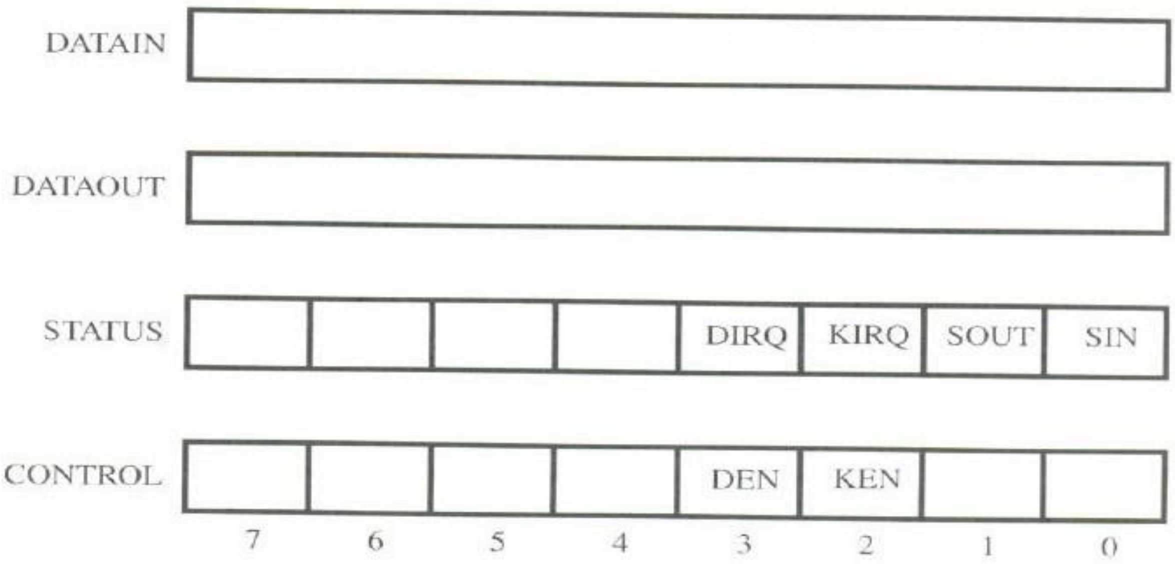
Address Decoder:
It enables the device to recognize its address when the address appears on address lines.

Data register - It holds the data being transferred to or from the processor.
Status register - It contains infn/. Relevant to the operation of the I/O devices.
The address decoder, data & status registers and the control circuitry required to co-ordinate I/O transfers constitute the device‟ s I/F circuit.

For an input device, SIN status flag in used SIN = 1, when a character is entered at the keyboard.

For an output device, SOUT status flag is used SIN = 0, once the char is read by processor.

Eg



DIR Q - Interrupt Request for display.
KIR Q -Interrupt Request for keyboard.
KEN - keyboard enable.
DEN - Display Enable.
SIN, SOUT - status flags.
The data from the keyboard are made available in the DATAIN register & the data sent to the display are stored in DATAOUT register.

Program:
WAIT K Move # Line, Ro
Test Bit #0, STATUS
Branch = 0 WAIT K
Move DATAIN, R1
WAIT D Test Bit #1, STATUS
Branch = 0 WAIT D
Move R1, DATAOUT
Move R1, (Ro)+

```
Compare #OD, R1
Branch = 0 WAIT K
Move #DOA, DATAOUT
Call PROCESS
```

Explanation:

- This program, reads a line of characters from the keyboard & stores it in a memory buffer starting at locations LINE.
- Then it calls the subroutine “PROCESS” to process the input line.
- As each character is read, it is echoed back to the display.
- Register Ro is used as a updated using Auto – increment mode so that successive characters are stored in successive memory location.
- Each character is checked to see if there is carriage return (CR), char, which has the ASCII code 0D(hex).
- If it is, a line feed character (on) is sent to more the cursor one line down on the display & subroutine PROCESS is called. Otherwise, the program loops back to wait for another character from the keyboard.

Program Controlled I/O

- Here the processor repeatedly checks a status flag to achieve the required synchronization between Processor & I/O device.(ie) the processor polls the device.
- There are 2 mechanisms to handle I/o operations. They are,
- □ Interrupt,-
- □ DMA (Synchronization is achieved by having I/O device send special over the bus where is ready for data transfer operation)

DMA:

- Synchronization is achieved by having I/O device send special over the bus where is ready for data transfer operation)
- It is a technique used for high speed I/O device.
- Here, the input device transfers data directly to or from the memory without continuous involvement by the processor.

2. Explain about Interrupts With Neat Diagram.(Nov 12)

OVERVIEW:

- Interrupt Hardware
- Enabling and Disabling Interrupts
- Handling Multiple Devices
  - ❖ Vectored Interrupts
  - ❖ Interrupts Nesting
  - ❖ Interrupt Priority
- Controlling Device Requests

- Exceptions

INTERRUPTS:

- An interrupt is an event inside a computer system requiring some urgent action by the CPU. In response to the interrupt, the CPU suspends the current program execution and branches in to an Interrupt Service Routine(ISR).
- The ISR is a program that services the interrupt by taking appropriate actions. After the execution of ISR, the CPU returns back to the interrupted program.
- On returning from ISR, the CPU resumes the old interrupted program as if nothing has taken place.
- This means the CPU should continue from the place when interruption has occurred and the CPU status at that time should be the same.
- For this purpose, the condition of the CPU should be saved before taking up ISR. Before returning from ISR, this saved status should be loaded back into the CPU.

- The processor can perform some useful task while waiting for I/O device to become ready.

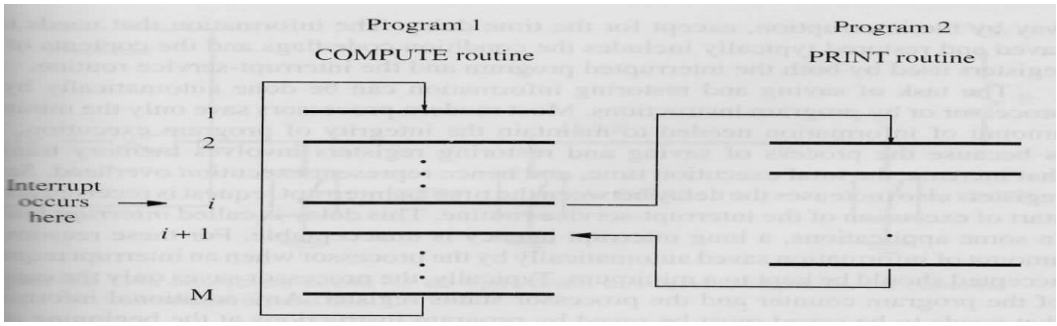


Fig: Interrupt Service Routine

- To allow this to happen, the I/O devices can be allowed to alert the processor when it becomes ready.
- If can be done by sending a hardware signal called interrupt to the processor. Interrupt Request Line Is used for this purpose.
- The process is no longer to continuously check the status of external devices, it can use the waiting period other useful functions.
- Interrupts eliminates waiting period. The routine serviced in response to an interrupt request is called interrupt service routine. The processor acknowledges the interrupt by interrupt acknowledgement signal.
- The interrupt service routine is very similar to sub routine.
- The task of saving and restoring information can be done automatically by processor or by program instruction.
- Saving and restoring registers requires memory transfers are ISR overhead.
- The time between when an interrupt request is received and the start of execution of ISR is called interrupt latency.

- The routine executed in response to an interrupt request is called the interrupt service routine, which is the PRINT routine is our example.
- Interrupts bear considerable resemblance to subroutine calls.
- Assume that an interrupt request arrives during execution of instruction I in figure.
- The processor first completes execution of instruction I, and then it leads the program counter with address of the first instruction of the interrupt service routine.

INTERRUPT HARDWARE:

- An I/O device requests can interrupt by activating a bus line called interrupt request. Computers can request an interrupt. The interrupts are classified as:
-
    - **Single level interrupt**
    - **Multi Level interrupt**

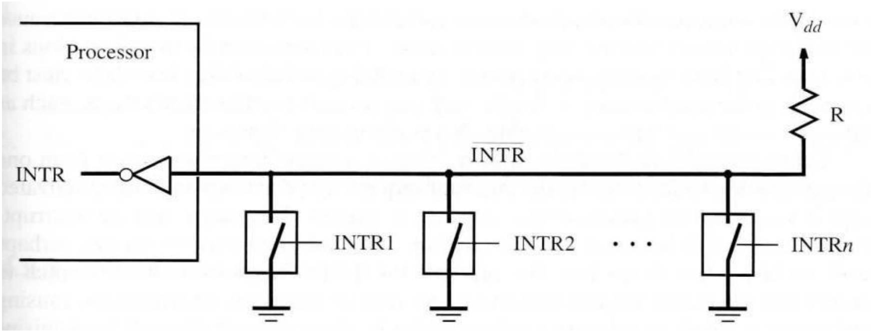


Fig: Common Interrupt Request Line

- A single interrupt request line may be used to serve an I/O device as shown in figure

- All devices are connected to interrupt request line via switches to ground. The device to raise interrupt closes the switch.

- If all interrupt request lines of I/O devices are inactive, then interrupt request line will be equal to V_{dd} . when a device attempts to raise an interrupt by closing the switch, the voltage drops to 0, causing the interrupt request signal INTR, received by the processor to go 1, INTR is logical OR (or) the interrupt request from individual devices.

$$\mathbf{INTR = INTR_1 + INTR_2 + \ldots.. + INTR_n}$$

ENABLING AND DISABLING INTERRUPTS:

- The arrival of an interrupt request from an external device cause the processor to suspend the execution of one program and start the execution of another.
- A simple way is to provide machine instructions, such as interrupt-enable and interrupt-disable that perform this function.
- Let us consider in detail the interrupt handling by single interrupt request line. When a device activates the interrupt request signal, it keeps this signal activate until it is serviced by processor.

- This ensures that this active request signal does not lead to successive interruptions, causing to enter an infinite loop.

This can be handled by the following ways.

1. The first possibility is to have the processor ignore the interrupt request line until the ISR is completed. This can be done using interrupt disable as the first instruction in ISR and interrupt enable as the last instruction.

2. The second option is processor while saving the contents of program counter (pc) and programs status Register(PS) on the stack, it also performs an equivalent of executing an interrupt disable instruction (interrupt enable bit =0), while after completing ISR H is enabled.

The sequence of events in handling interrupts can be summarized as follows.

1. The device raises an interrupt request.
2. The processor interrupts the program currently being executed.
3. Interrupts are disabled by interrupt disable instruction or by setting interrupt enable bit to 0.
4. The device is informed that is interrupt request has been recognized by interrupt acknowledgement.
5. The ISR Is serviced.
6. Interrupts are enabled and execution of the interrupted program is resumed.

HANDLING MULTIPLE DEVICES:

Consider the situation where number of devices capable of initiating interrupts is connected to processor. The devices are operationally independent.

- Polling Scheme
- Vectored interrupts
- Interrupt nesting
- Interrupt priority
- Daisy chain
- Priority Groups

Polling Scheme:

- It is a simplest scheme to identify the interrupting device. The device that raises the interrupt will set one of the bit d (IRQ) in status register t01.
- The processor will poll the devices to find which raised an interrupt.

Disadvantage

- In this technique is ti9me spent in interrogating the IRQ bits of all devices.

Vectored interrupts:

- To reduce time involved in polling process, a device requesting an interrupt may identify itself directly to the processor.
- The code is used to identify the device and it may represent the starting address of the interrupt-service routine for that device.
- If the interrupt produces a call to a predetermined memory location, which is starting address of ISR, then that address is called vectored address and such interrupt are called vectored interrupts.

Interrupt Nesting:

- For some device, a long delay in responding to an interrupts request may lead to error in the operation of computer.
- Such interrupts are acknowledged and serviced even though processor is executing an interrupt service routine (ISR) for another device.
- A system of interrupts that allow an interrupt service routine to be interrupted is known as nested interrupts.

Interrupt Priority:

- When interrupt request arrives from two or more devices simultaneously, the processor has to decide which request should be serviced first. And which one one should be delayed.

- The processor takes this decision with the help of interrupt priorities.

- The processor accepts interrupt request having highest priority.

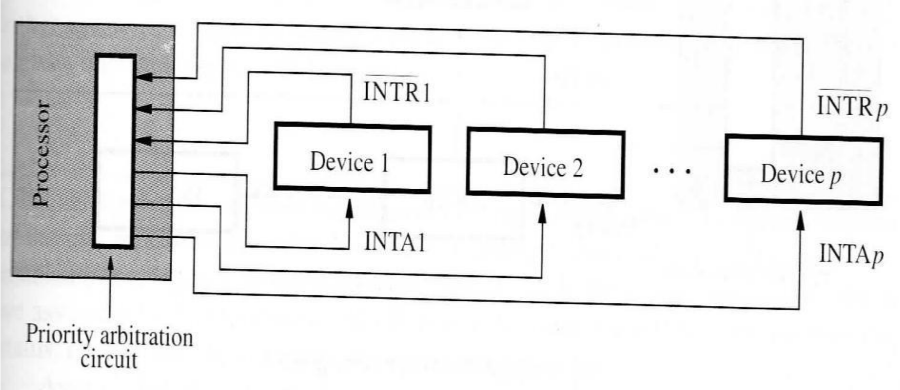

Processor
$\overline{\text{INTR}}1$
$\overline{\text{INTR}}p$
Device 1
Device 2
. . .
Device p
INTA1
INTAp
Priority arbitration circuit

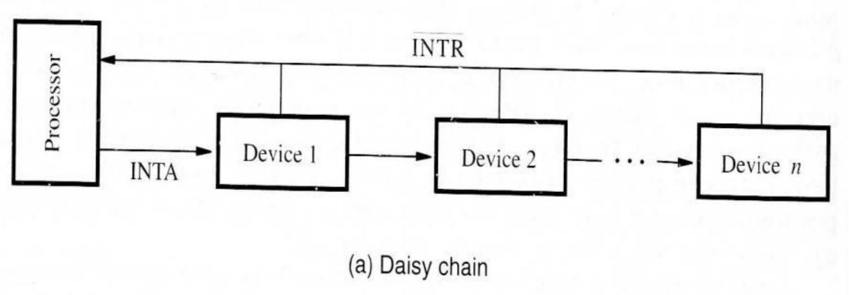
Fig: Handling Priority Interrupts

Daisy chain:

Consider the problem of simultaneous request from two or more devices. The processor has to decide which request to be serviced first.

Priority is determined by the order in which devices are polled. In daisy chain scheme, interrupt request line INTR is common to all devices.

The interrupt acknowledgement INTA, propagates serially through the devices.

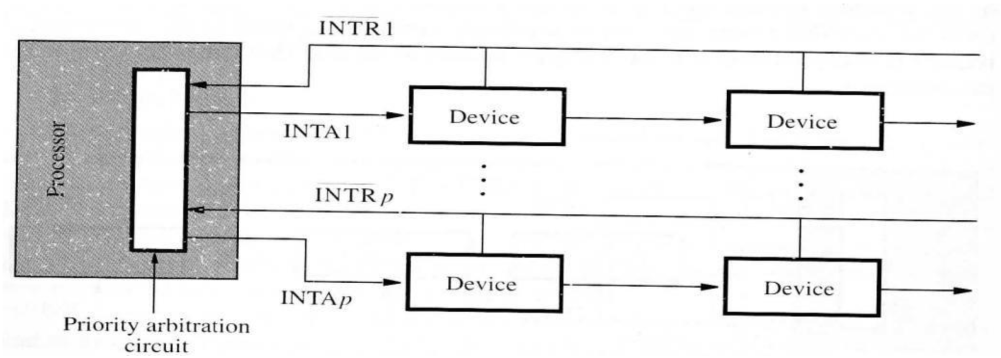


(a) Daisy chain

- When several devices raise an interrupt request, the processor responds by setting INTA line to 1. The signal is received by device 1.
- Device 1 passes the signal on to device 2 only if it does not require any service. If device 1 has a pending request for interrupt, it block S the INTA signal and proceeds to put is device identifying code on the data lines.
- Therefore, in daisy chain arrangement, the device that is closest to the processor has the highest priority.

Priority groups:

- In this scheme, devices are organized in groups, and each group is connected at a different priority level within a group, the devices are connected in a daisy chain.



$\overline{\text{INTR}}$ 1
Processor
INTA1
Device
Device
$\overline{\text{INTR}}$ *p*
INTA*p*
Device
Device
Priority arbitration circuit

Fig. Arrangement of priority groups

Controlling device request:

I/O device generate an interrupt request when it is ready for data transfer, while handing interrupt requests it is requested to ensure that interrupts request are generated only by those I/O devices that are being used by a given program.

There are two mechanism used to control device request.

- One is at the device end
- And the other at processor end

- At the device end, an interrupt enable bit in the control and status register. The programmer is allowed to set or reset this interrupt-enable bit.

- It the processor end, either an interrupt-enable in the program status register (PS) or a priority structure determines whether the given request will be accepted.

Exceptions:

An interrupt is an event that causes the execution of one program to be suspended and begins execution of another program. Exception is referred to any event that causes an interruption. I/O interrupt is an example of exception.

- **Faults**
  - Faults are exceptions that are detected and serviced before the execution of the faulting instruction.
  - Example: Page fault in virtual memory system.
- **Traps**
  - Traps are exceptions that are reported immediately after the execution of the instructions which causes the problem.
  - Example: op-code field of instruction may not correspond to any level instruction.
- **Aborts:**
  - These are exceptions which do not permit the precise location of the instructions causing the exception to be determined. They report severe error-hardware error.

3. Explain the use of interrupts in operating system

Use Of Interrupts In Operating System:

The operating system is responsible for coordinating all activities within a computer. It makes extensive use of interrupts to perform input/output operations and communicate with and control the execution o user programs.

The interrupt mechanism enables the operating system to assign priorities, switch from one user program to another, implement security and protection features and coordinate input/output activities.

Multitasking is a mode of operation in which a processor executes several user programs at the same time. Each program runs for a short period called a time slice, then another program runs for its time slice and so on.

Operating system routines:

a) OS Initialization, services and scheduler

OSINIT	set interrupt vectors.
	Time-Slice-Clock-SHEDULER
	Software interrupt-OSSERVICES

Keyboard interrupt-IO Data.

OSSERVICES Examine stack to determine requested operation call appropriate routine.

SCHEDULER Save program state.
Select a runnable process.
Restore saved context to of new process.
Push new values for PS and PC on stack.
Return from interrupt.

b) Input/output routines

IOINIT Set process status to Blocked.
Initialize memory Buffer address pointer and counter.
Call device driven to initialize device an enable interrupts in the device interface.

IODATA Poll devices to determine source of interrupts call appropriate driver.
If END = 1, then set process status to Runnable Return from interrupt.

c) Keyboard Driver.

KBDINIT Enable interrupt.
Return from subroutine.

KBDDATA Check device status.
If ready, then transfer character.
If character = CR, then{ set END = 1;
Disable interrupts}
Else set END=0
Return from subroutine.

At the time the operating system is started, an initialization routine, called OSINIT is executed. OSINIT loads the starting address of a routine called SCHEDULER in the interrupt vector corresponding to the timer interrupt. Hence at the end of each time slice, the timer interrupt causes this routine to be executed.
The current state of execution of program is called a process.
A process can be in one of the three states.

Running -the program is currently being executed.
Runable – The program is ready for execution but is waiting to be selected by the scheduler.
Blocked – the program is not to be ready to resume execution for some reason. It may be waiting for completion of an input/output operation.

- Assume program A is in the Running state during a given time slice. At the end of the time slice, the times interrupt the execution of this program and start the execution of SCHEDULER. The information saved, which is called the program state, include register content, the program, counter, and the processor status word.

- Then, SCHEDULER selects for execution some other program B, which was suspended earlier and is in the Runnable state.

- Suppose that program a need to read an input line from the keyboard. It request input/output service from the operating system. It uses the stack or the processor register to pass information to the OS describing the required operation, the input/output device and the address of a buffer in the program data read where the line should be placed. Then it executes a software interrupt instruction.

- The interrupt vector for the instruction points to the OSSERVICES routine. This routine examines the information on the stack and initiates the requested operation by calling IOINIT, which is responsible for starting input/output operations.

- The IOINIT routine seats the process associated with program A into the Blocked state. This routine carries the initiating address pointers and byte counter, and then calls KBDINIT, which performs any initialization operations needed by the device.

- KBDINIT enables interrupts in the interface circuit by setting the appropriate bit in its control register and then it returns to IONINIT, which returns to OSSERVICES. The keyboard is now ready to participate in a data transfer operations. It will generate an interrupt request whenever a key is pressed. This interrupt points to an OS routine called IODATA.

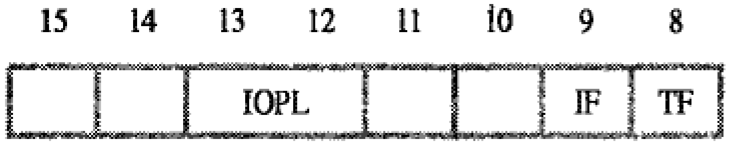
IODATA begins by polling these devices to determine the one requesting service. Then, it calls the device driver KBDDATA, which will transfer one character of data. If the character is a carriage Return, it will set END=1, to inform IODATA that the requested input/output operation has been completed.

4. Explain in detail about the Pentium interrupt structure.

The IA-32 architecture of which are examples of Pentium processors, uses two interrupt request lines, a non-maskable interrupt (NMI) and a maskable interrupt, also called user interrupt request, INTR.
Interrupt requests on NMI are always accepted by the processor.
Requests on INTR are accepted only if they have a higher privilege level than the program currently executing.

INTR requests can also be enabled or disabled by setting an interrupt-enable bit in the processor status register.
In addition to external interrupts, there are many events that arise during program execution that can cause an exception. These include invalid opcodes, division errors, overflow etc. The occurrence of any of these events causes the processor to branch to an interrupt-service routine. Each interrupt is assigned a vector number. In case of INTR, vector number is sent by the I/O device over the bus when the interrupt request is acknowledged. For all other exceptions, the vector number is pressigned. Based on the vector number, the processor determines the starting address of the interrupt-service routine from a table called the Interrupt Descriptor Table.

A companion chip to the Pentium processor is called the Advanced Programmable Interrupt Controller (APIC). Various devices are connected to the processor through this chip. The interrupt controller implements a priority structure among different devices and sends an appropriate vector number to the processor for each device.

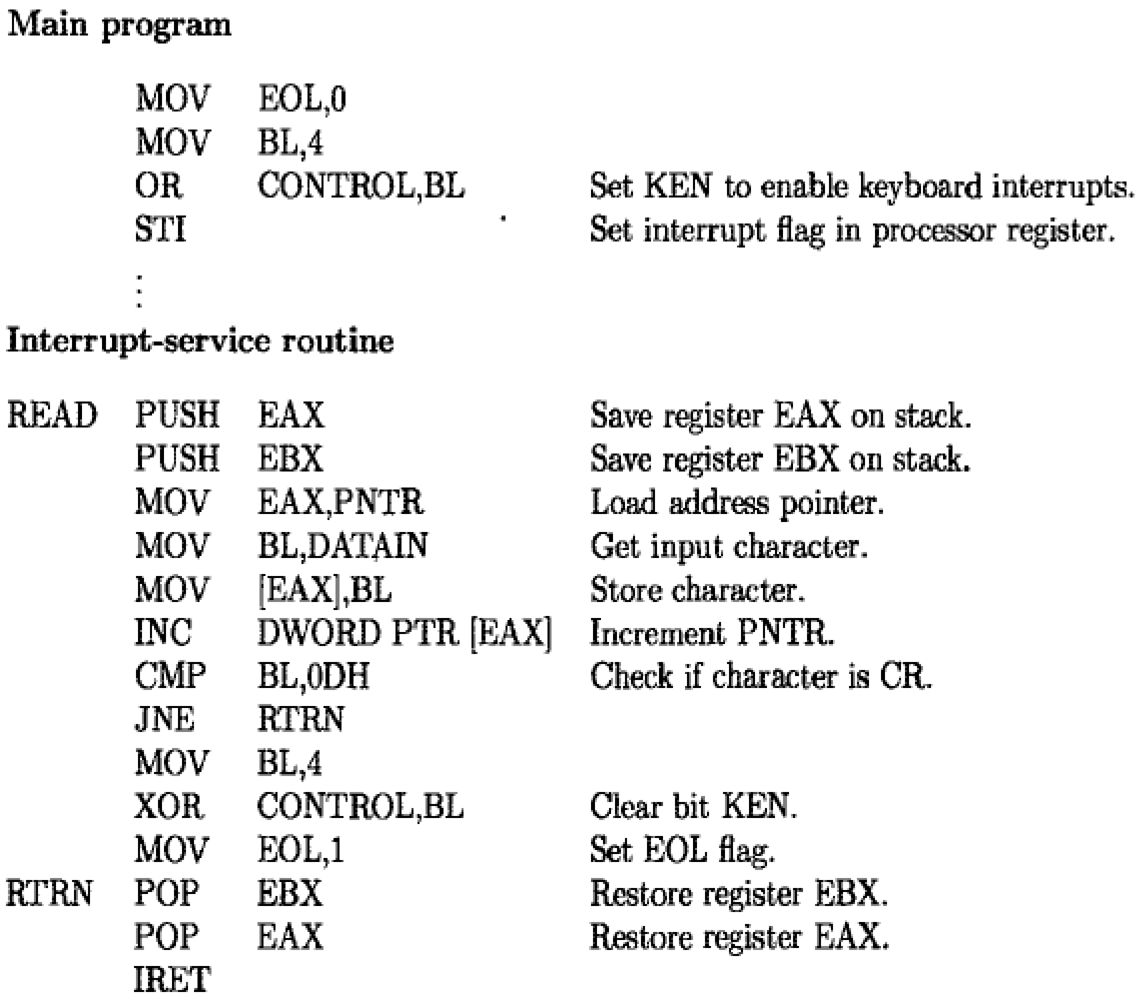


15	14	13	12	11	10	9	8
		IOPL				IF	TF

The processor status register, called EFLAGS in Intel is shown above. It shows 8 to 15 bits of this register, which contain the Interrupt Enable Flag (IF), the Trap Flag (TF) and the I/O Privilege Level (IOPL). When IF=1, INTR interrupts are accepted. The Trap flag enables trace interrupts after every instruction.

When an interrupt request is received or when an exception occurs, the processor takes the following actions:

1. It pushes the processor status register, the current segment register (CS), and the instruction pointer (EIP) onto the processor stack pointed to by the processor stack pointer, ESP.
2. In the case of an exception resulting from an abnormal execution condition, it pushes a code on the stack describing the cause of the exception.
3. It clears the corresponding interrupt-enable flag, if appropriate, so that further interrupts from the same source are disabled
4. It fetches the starting address of the interrupt request, by transferring input or output data, the interrupt-service routine returns to the interrupted program using a return from interrupt instruction, IRET. This instruction pops EIP,CS and the processor status register from the stack into the corresponding registers, thus restoring the processor state.



```
Main program

        MOV     EOL,0
        MOV     BL,4
        OR      CONTROL,BL               Set KEN to enable keyboard interrupts.
        STI                              Set interrupt flag in processor register.
        :

Interrupt-service routine

READ    PUSH    EAX                      Save register EAX on stack.
        PUSH    EBX                      Save register EBX on stack.
        MOV     EAX,PNTR                 Load address pointer.
        MOV     BL,DATAIN                Get input character.
        MOV     [EAX],BL                 Store character.
        INC     DWORD PTR [EAX]          Increment PNTR.
        CMP     BL,0DH                   Check if character is CR.
        JNE     RTRN
        MOV     BL,4
        XOR     CONTROL,BL               Clear bit KEN.
        MOV     EOL,1                    Set EOL flag.
RTRN    POP     EBX                      Restore register EBX.
        POP     EAX                      Restore register EAX.
        IRET
```

Fig. An interrupt servicing routine to read one line from a keyboard using interrupts on IA-32 processors

5. What is DMA? Explain in detail. (Nov 12)

DIRECT MEMORY ACCESS

A special control unit may be provided to allow the transfer of large block of data at high speed directly between the external device and main memory, without continuous intervention by the processor. This approach is called DMA.

DMA transfers are performed by a control circuit called the DMA Controller.

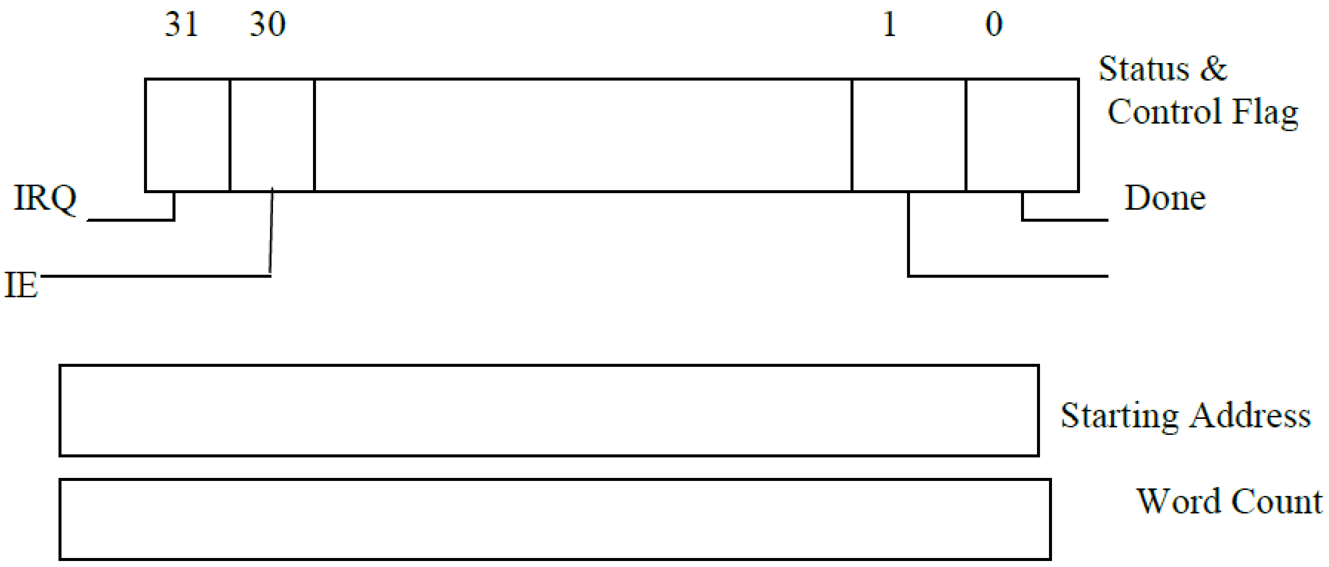
To initiate the transfer of a block of words, the processor sends,

- Starting address
- Number of words in the block
- Direction of transfer.

- When a block of data is transferred , the DMA controller increment the memory address for successive words and keep track of number of words and it also informs the processor by raising an interrupt signal.
- While DMA control is taking place, the program requested the transfer cannot continue and the processor can be used to execute another program.

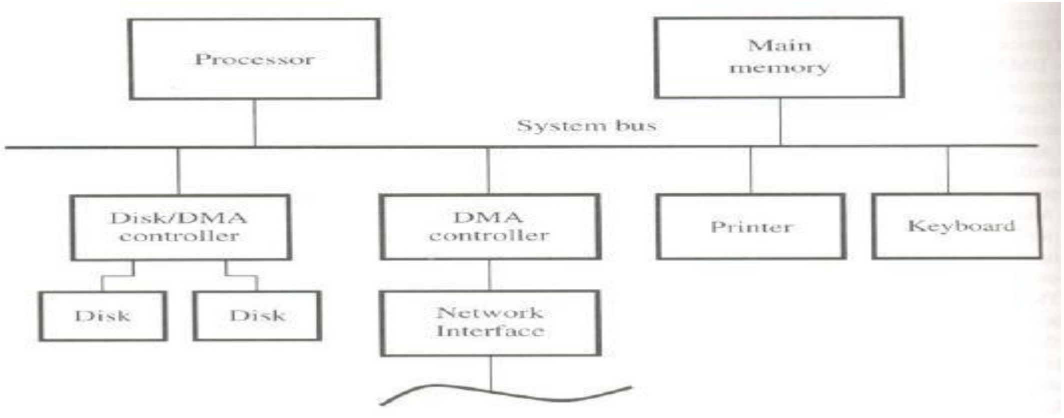
- After DMA transfer is completed, the processor returns to the program that requested the transfer.

Fig: Registers in a DMA Interface



R/W - determines the direction of transfer.
When
R/W =1, DMA controller read data from memory to I/O device.
R/W =0, DMA controller perform write operation.
Done Flag=1, the controller has completed transferring a block of data and is ready to receive another command.
IE=1, it causes the controller to raise an interrupt (interrupt Enabled) after it has completed transferring the block of data.
IRQ=1, it indicates that the controller has requested an interrupt.

Fig: Use of DMA controllers in a computer system



- A DMA controller connects a high speed network to the computer bus . The disk controller two disks, also has DMA capability and it provides two DMA channels.

- To start a DMA transfer of a block of data from main memory to one of the disks, the program write s the address and the word count inf. into the registers of the corresponding channel of the disk controller.
- When DMA transfer is completed, it will be recorded in status and control registers of the DMA channel (ie) Done bit=IRQ=IE=1.

Cycle Stealing:

- Requests by DMA devices for using the bus are having higher priority than processor requests.
- Top priority is given to high speed peripherals such as ,
- Disk
- High speed Network Interface and Graphics display device.
- Since the processor originates most memory access cycles, the DMA controller can be said to steal the memory cycles from the processor. This interviewing technique is called Cycle stealing.

Burst Mode:

The DMA controller may be given exclusive access to the main memory to transfer a block of data without interruption. This is known as Burst/Block Mode

Bus Master:

The device that is allowed to initiate data transfers on the bus at any given time is called the bus master.

Bus Arbitration:

It is the process by which the next device to become the bus master is selected and the bus mastership is transferred to it.

Types:

There are 2 approaches to bus arbitration. They are,

- Centralized arbitration (A single bus arbiter performs arbitration)
- Distributed arbitration (all devices participate in the selection of next bus master).

Centralized Arbitration:

- Here the processor is the bus master and it may grants bus mastership to one of its DMA controller.
- A DMA controller indicates that it needs to become the bus master by activating the Bus Request line (BR) which is an open drain line.
- The signal on BR is the logical OR of the bus request from all devices connected to it.
- When BR is activated the processor activates the Bus Grant Signal (BGI) and indicated the DMA controller that they may use the bus when it becomes free.
- This signal is connected to all devices using a daisy chain arrangement.
- If DMA requests the bus, it blocks the propagation of Grant Signal to other devices and it indicates to all devices that it is using the bus by activating open collector line, Bus Busy (BBSY).

Fig: A simple arrangement for bus arbitration using a daisy chain

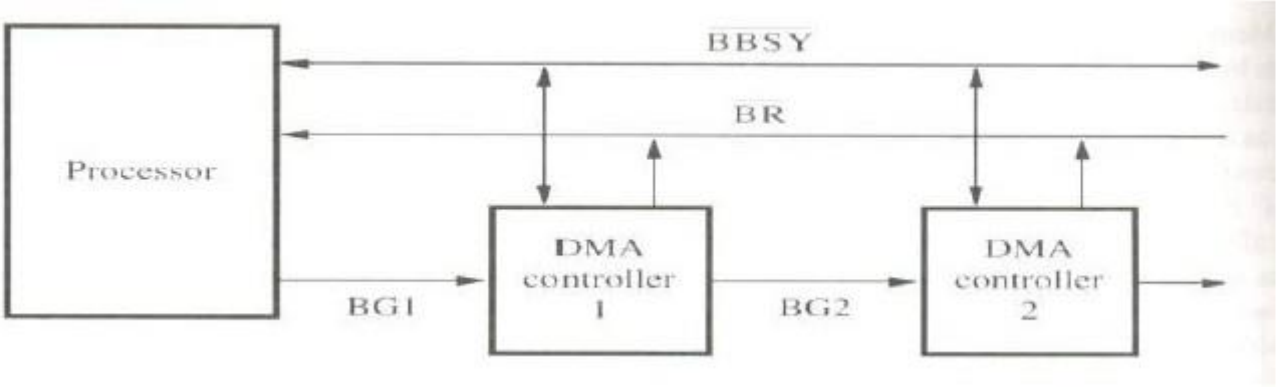
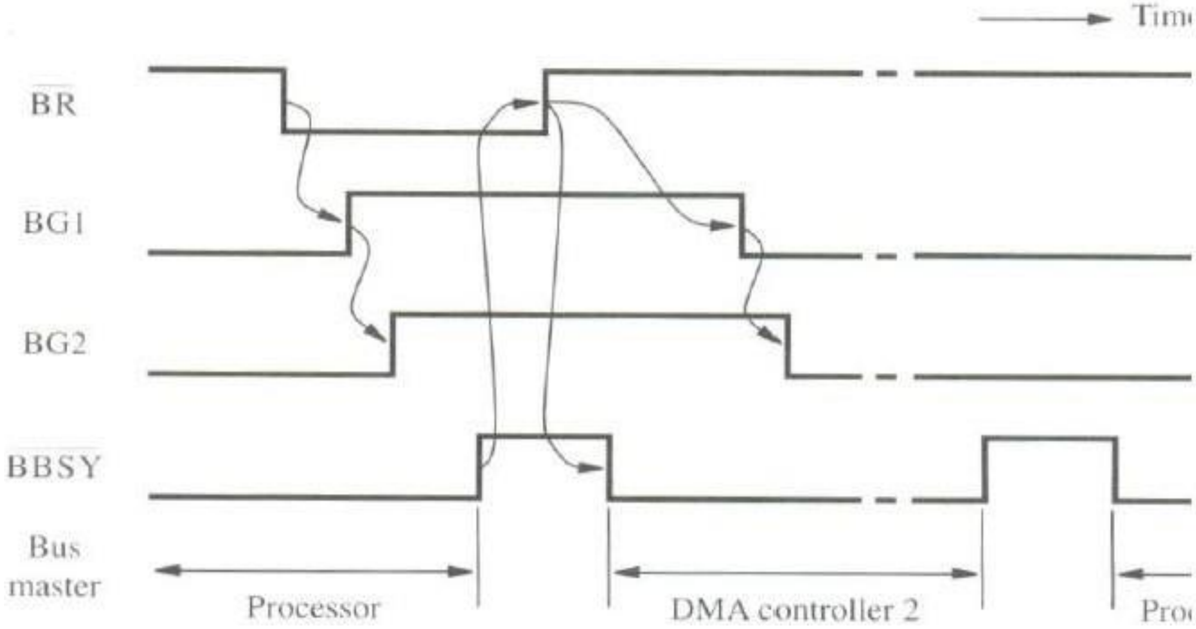


Fig: Sequence of signals during transfer of bus mastership for the devices



The timing diagram shows the sequence of events for the devices connected to the processor is shown. DMA controller 2 requests and acquires bus mastership and later releases the bus.

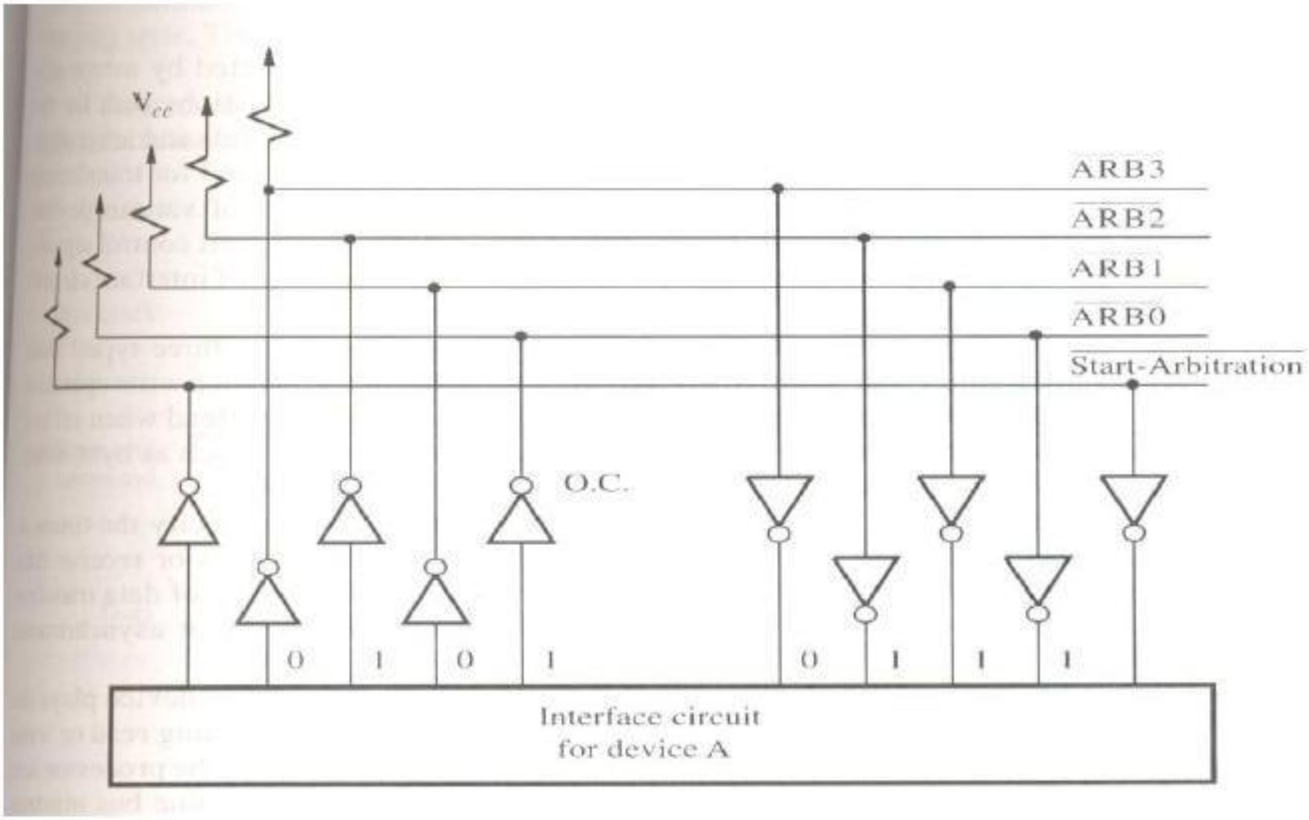
During its tenture as bus master, it may perform one or more data transfer.

After it releases the bus, the processor resources bus mastership

Distributed Arbitration:

It means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process.

Fig: A distributed arbitration scheme



- Each device on the bus is assigned a 4 bit id.
- When one or more devices request the bus, they assert the Start-Arbitration signal & place their 4 bit ID number on four open collector lines, ARB0 to ARB3.
- A winner is selected as a result of the interaction among the signals transmitted over these lines.
- The net outcome is that the code on the four lines represents the request that has the highest ID number.
- The drivers are of open collector type. Hence, if the i/p to one driver is equal to 1, the i/p to another driver connected to the same bus line is equal to "0”(ie. bus is in low-voltage state).

Eg:

Assume two devices A & B have their ID 5 (0101), 6(0110) and their code is 0111.

Each devices compares the pattern on the arbitration line to its own ID starting from MSB.

If it detects a difference at any bit position, it disables the drivers at that bit position. It does this by placing “0” at the i/p of these drivers.

In our eg. “A” detects a difference in line ARB1, hence it disables the drivers on lines ARB1 & ARB0.

This causes the pattern on the arbitration line to change to 0110 which means that “B” has won the contention.

6. Explain about Buses in detail.

Buses

A bus protocol is the set of rules that govern the behavior of various devices connected to the bus ie, when to place information in the bus, assert control signals etc.

The bus lines used for transferring data is grouped into 3 types. They are,

- Address line

- Data line
- Control line.

Control signals - Specifies that whether read / write operation has to be performed. It also carries timing information / (i.e) they specify the time at which the processor & I/O devices place the data on the bus & receive the data from the bus. During data transfer operation, one device plays the role of a “Master”.

Master **-** device initiates the data transfer by issuing read / write command on the bus. Hence it is also called as Initiator. The device addressed by the master is called as Slave / Target.

Types of Buses:

There are 2 types of buses. They are,

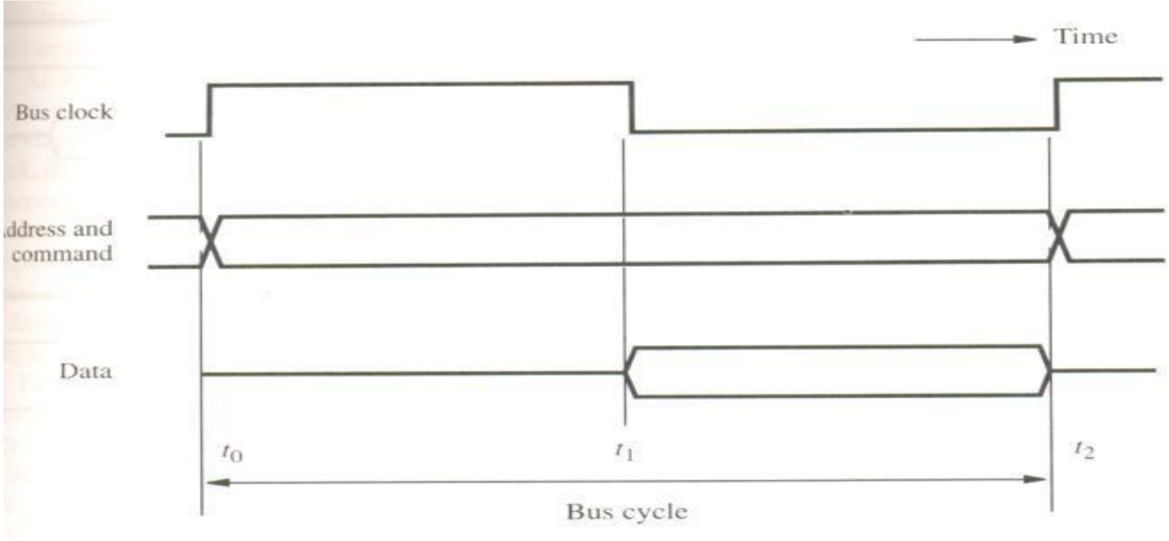
- Synchronous Bus
- Asynchronous Bus

Synchronous Bus:-

In synchronous bus, all devices derive timing information from a common clock line. Equally spaced pulses on this line define equal time. During a bus cycle, one data transfer on take place. The crossing points indicate the tone at which the patterns change.

A signal line in an indeterminate / high impedance state is represented by an intermediate half way between the low to high signal levels.

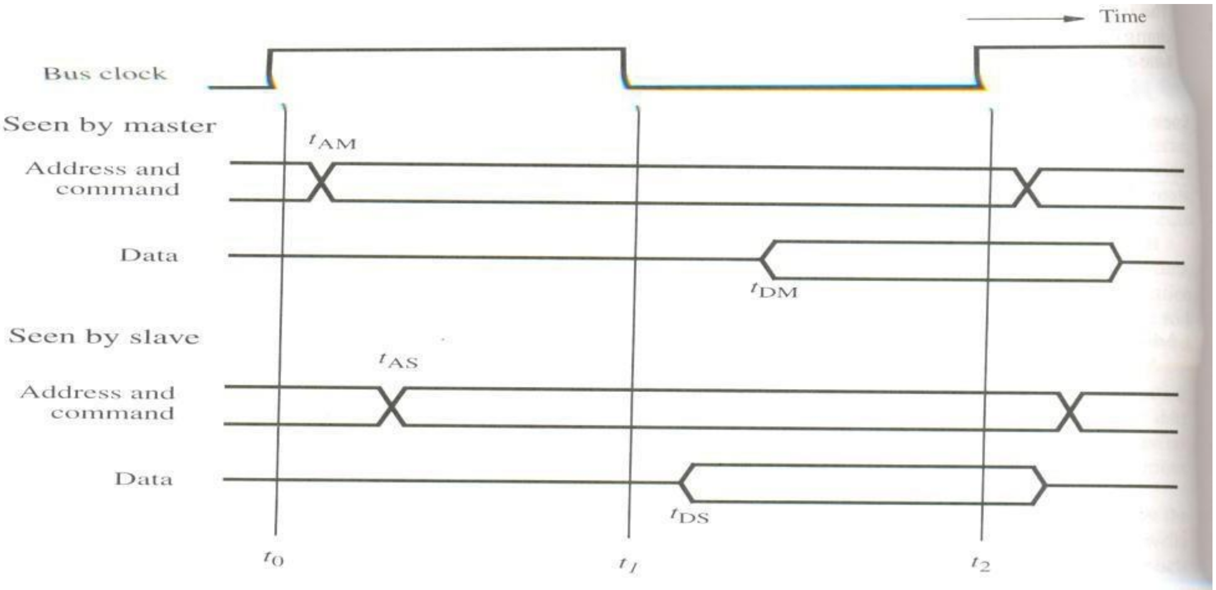
Fig: Timing of an input transfer of a synchronous bus.



- At time to, the master places the device address on the address lines & sends an appropriate command on the control lines.
- In this case, the command will indicate an input operation & specify the length of the operand to be read.
- The clock pulse width t1 – t0 must be longer than the maximum delay between devices connected to the bus.

- The clock pulse width should be long to allow the devices to decode the address & control signals so that the addressed device can respond at time t1.
- The slaves take no action or place any data on the bus before t1.

Fig: A detailed timing diagram for the input transfer



- The picture shows two views of the signal except the clock.
- One view shows the signal seen by the master & the other is seen by the salve.
- The master sends the address & command signals on the rising edge at the beginning of clock period (t0). These signals do not actually appear on the bus until tam.
- Sometimes later, at tAS the signals reach the slave.
- The slave decodes the address & at t1, it sends the requested data.
- At t2, the master loads the data into its i/p buffer.
- Hence the period t2, tDM is the setup time for the masters i/p buffer.
- The data must be continued to be valid after t2, for a period equal to the hold time of that buffers.

Demerits:

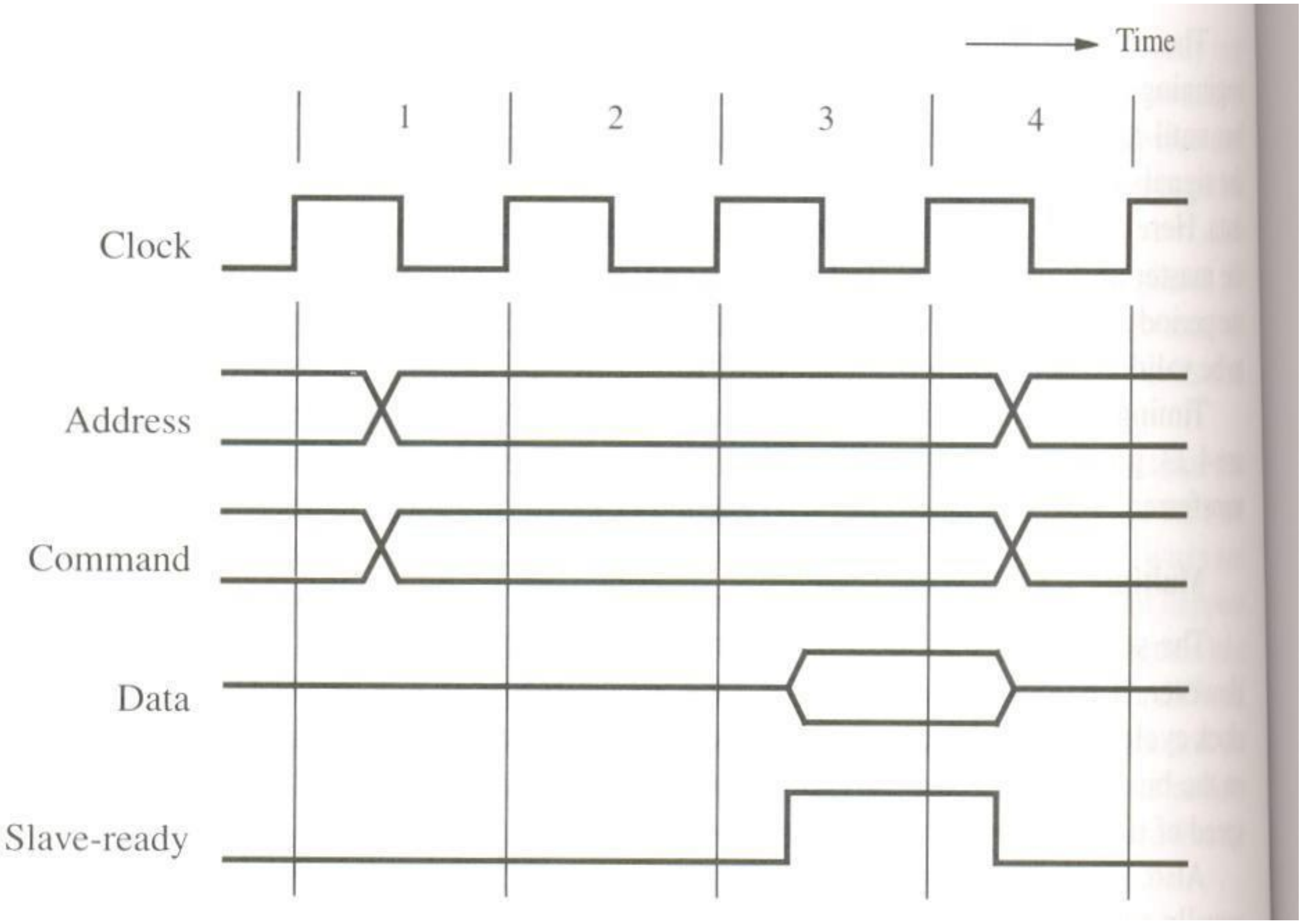
- The device does not respond.
- The error will not be detected.

Multiple Cycle Transfer:-

- During, clock cycle1, the master sends address & cmd infn/. On the bus requesting a read operation.
- The slave receives this information & decodes it.
- At the active edge of the clock (ie) the beginning of clock cycel2, it makes accession to respond immediately.
- The data become ready & are placed in the bus at clock cycle3.
- At the same times, the slave asserts a control signal called slave-ready.

- The master which has been waiting for this signal, strobes, the data to its i/p buffer at the end of clock cycle3.
- The bus transfer operation is now complete & the master sends a new address to start a new transfer in clock cycle4.
- The slave-ready signal is an acknowledgement form the slave to the master confirming that valid data has been sent.

Fig:An input transfer using multiple clock cycles



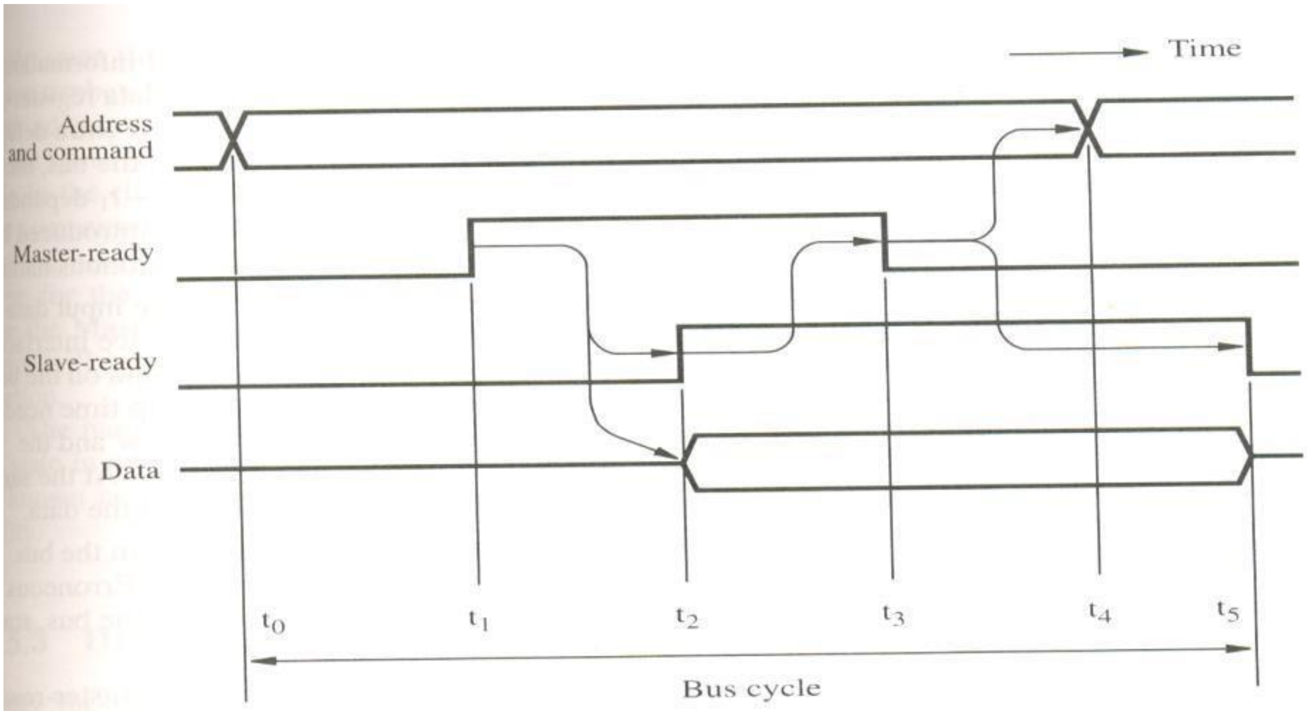
Asynchronous Bus:-

An alternate scheme for controlling data transfer on. The bus is based on the use of handshake between Master & the Slave. The common clock is replaced by two timing control lines.

They are

□ Master–ready
□ Slave ready.

Fig:Handshake control of data transfer during an input operation



The handshake protocol proceed as follows :

- At **t0** The master places the address and command information on the bus and all devices on the bus begin to decode the information
- At **t1** The master sets the Master ready line to 1 to inform the I/O devices that the address and command information is ready.
- The delay t1 – t0 is intended to allow for any skew that may occurs on the bus.
- The skew occurs when two signals simultaneously transmitted from one source arrive at the destination at different time.
- Thus to guarantee that the Master ready signal does not arrive at any device a head of the address and command information the delay t1 – t0 should be larger than the maximum possible bus skew.

- At **t2** The selected slave having decoded the address and command information performs the required i/p operation by placing the data from its data register on the data lines. At the same time, it sets the "slave – Ready" signal to 1.
- At **t3** The slave ready signal arrives at the master indicating that the i/p data are available on the bus.
- At **t4** The master removes the address and command information on the bus. The delay between t3 and t4 is again intended to allow for bus skew. Errorneous addressing may take place if the address, as seen by some device on the bus, starts to change while the master – ready signal is still equal to 1.
- At **t5** When the device interface receives the 1 to 0 tranitions of the Master – ready signal. Iremoves the data and the slave – ready signal from the bus. This completes the i/p transfer.

- In this diagram, the master place the output data on the data lines and at the same time it transmits the address and command information.
- The selected slave strobes the data to its o/p buffer when it receives the Master-ready signal and it indicates this by setting the slave – ready signal to 1.
- At time t0 to t1 and from t3 to t4, the Master compensates for bus.

- A change of state is one signal is followed by a change is the other signal. Hence this scheme is called as **Full Handshake**.
- It provides the higher degree of flexibility and reliability.

7. What are interface circuits? Explain

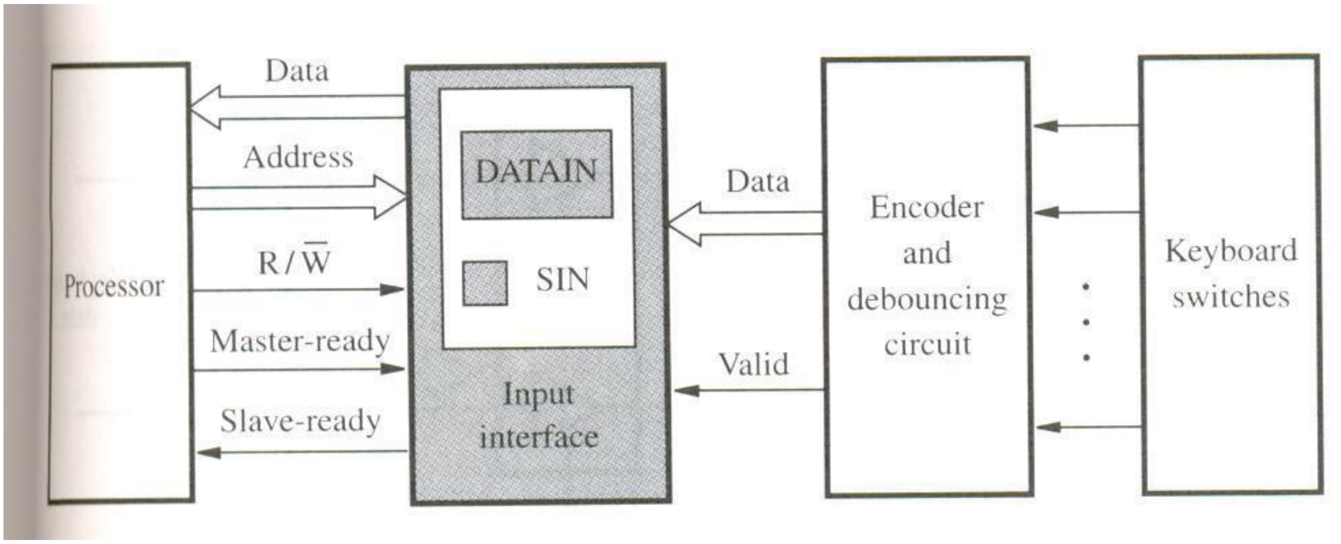
An I/O interface consists of the circuitry required to connect an I/O device to a computer bus. On one side of the interface are the bus signals for address, data and control. On the other side is a data path with its associated controls to transfer data between the interface and the I/O device. This side is called a port. This may be either a serial port or parallel port.

An I/O interface does the following

1. Provides a storage buffer for at least one word of data
2. Contains status flags that can be accessed by the processor to determine whether the buffer is full or empty.
3. Contains address-decoding circuitry to determine when it is being addressed by the processor.
4. Generates the appropriate timing signals required by the bus control scheme.
5. Performs any format conversion that may be necessary to transfer data between the bus and the I/O device, such as parallel -serial conversion in case of serial port.

Parallel port

A parallel port transfers data in the form of a number of bits, typically 8 or 16, simultaneously to or from the device. The connection between the device and the computer uses a multiple-pin connector and a cable with as many wires, typically arranged in a flat configuration. This arrangement is suitable for devices that are physically close to the computer.

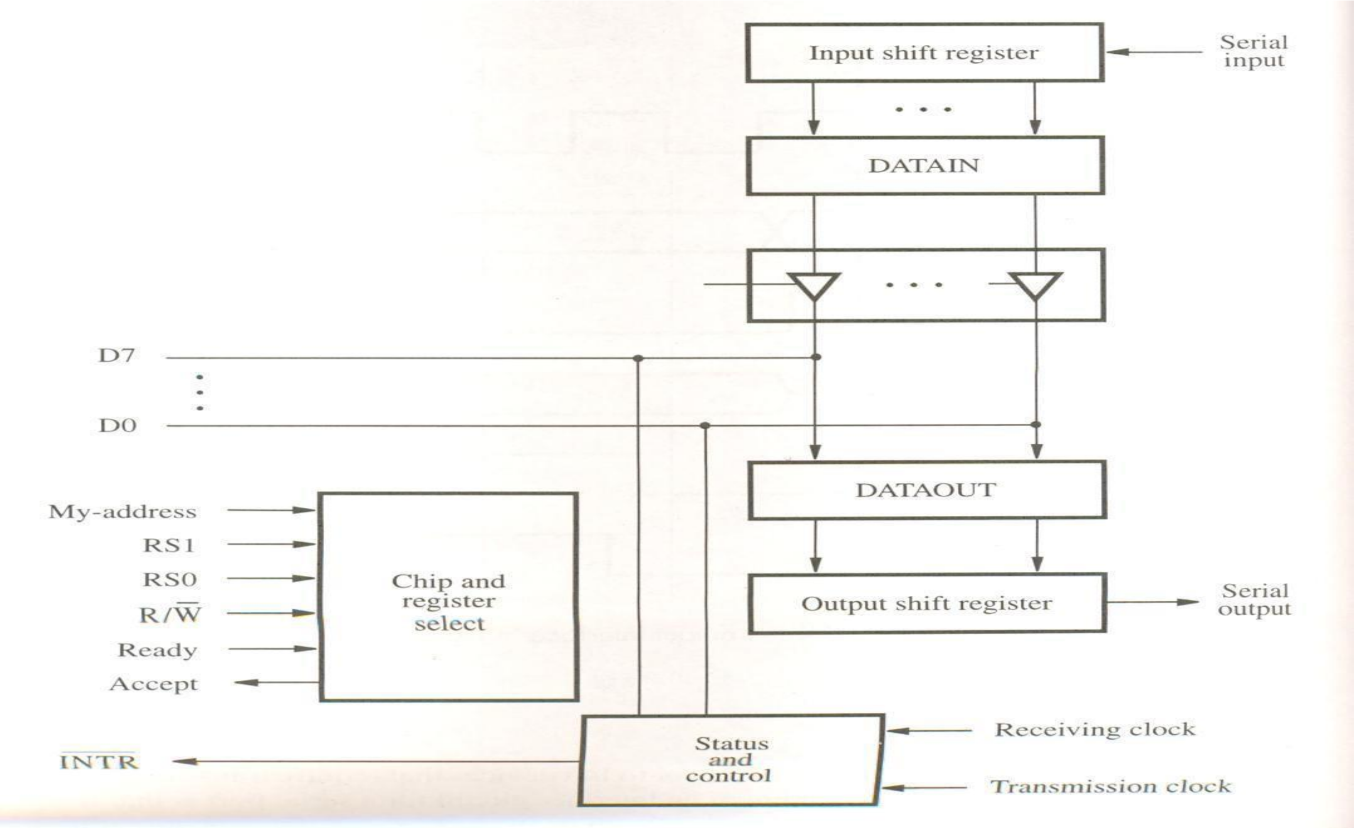


- The output of the encoder consists of the bits that represent the encoded character and one signal called valid,which indicates the key is pressed.
- The information is sent to the interface circuits,which contains a data register,DATAIN and a status flag SIN.

- When a key is pressed, the Valid signal changes from 0 to1,causing the ASCII code to be loaded into DATAIN and SIN set to 1.
- The status flag SIN set to 0 when the processor reads the contents of the DATAIN register.
- The interface circuit is connected to the asynchronous bus on which transfers are controlled using the Handshake signals Master ready and Slave-ready.

Serial port

A serial port transmits and receives data one bit at a time. The serial format is much more convenient and cost-effective where longer cables are needed. A serial port is used to connect the processor to I/O devices that require transmission of data one bit at a time. The key feature of an interface circuit for a serial port is that it is capable of communicating in a bit-serial fashion on the device side and in a parallel fashion on the bus side.



8. Explain in detail about PCI bus.

PCI: (Peripheral Component Inter Connect)

- PCI is developed as a low cost bus that is truly processor independent.
- It supports high speed disk, graphics and video devices.
- PCI has plug and play capability for connecting I/O devices.
- To connect new devices, the user simply connects the device interface board to the bus.

Data Transfer:

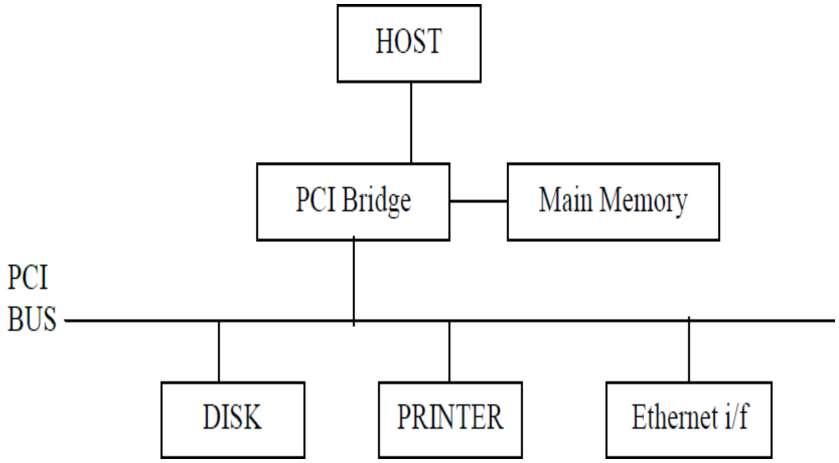
- The data are transferred between cache and main memory is the bursts of several words and they are stored in successive memory locations.

- When the processor specifies an address and request a „read‟ operation from memory, the memory responds by sending a sequence of data words starting at that address.
- During write operation, the processor sends the address followed by sequence of data words to be written in successive memory locations.
- PCI supports read and write operation.
- A read / write operation involving a single word is treated as a burst of length one.
- PCI has three address spaces. They are

    - Memory address space
    - I/O address space
    - Configuration address space

- I/O address space → It is intended for use with processor
- Configuration space → It is intended to give PCI, its plug and play capability.
- PCI Bridge provides a separate physical connection to main memory.
- The master maintains the address information on the bus until data transfer is completed.
- At any time, only one device acts as bus master.
- A master is called an initiator in PCI which is either processor or DMA.
- The addressed device that responds to read and write commands is called a target.
- A complete transfer operation on the bus, involving an address and bust of data is called a transaction.

Fig: Use of a PCI bus in a Computer system



Data Transfer Signals on PCI Bus:

Name Function

CLK - 33 MHZ / 66 MHZ clock

FRAME # - Sent by the indicator to indicate the duration of transaction

AD - 32 address / data line

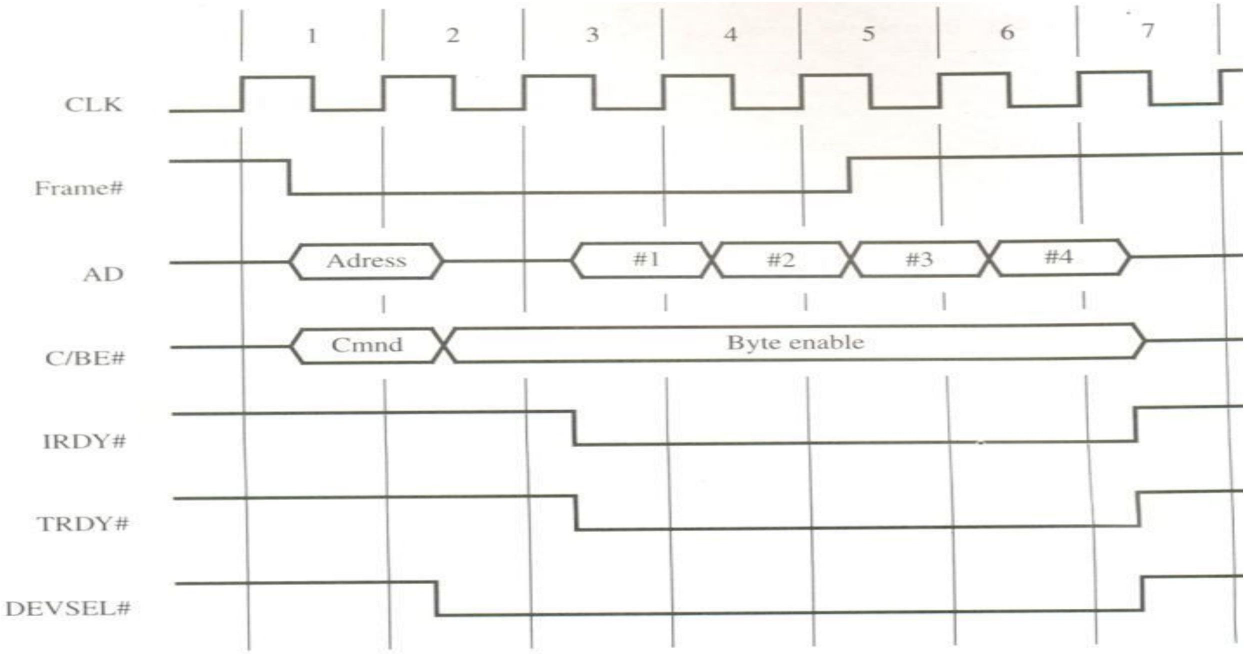
C/BE # - 4 command / byte Enable Lines

IRDY, TRDYA - Initiator Ready, Target Ready Signals

DEVSEL # - A response from the device indicating that it has recognized its address and is ready for data transfer transaction.

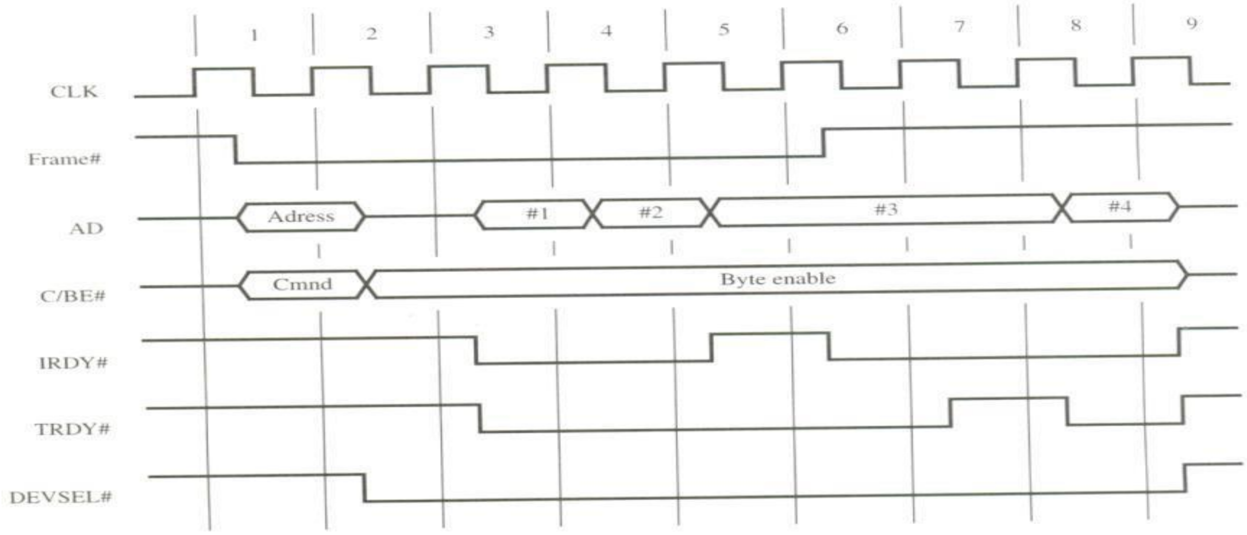
IDSEL # - Initialization Device Select Individual word transfers are called phases.

Fig: Read operation an PCI Bus



- In Clock cycle1, the processor asserts FRAME # to indicate the beginning of a transaction; it sends the address on AD lines and command on C/BE # Lines.
- Clock cycle2 is used to turn the AD Bus lines around ; the processor ; The processor removes the address and disconnects its drives from AD lines.
- The selected target enable its drivers on AD lines and fetches the requested data to be placed on the bus.
- It asserts DEVSEL # and maintains it in asserted state until the end of the transaction.
- C/BE # is used to send a bus command in clock cycle and it is used for different purpose during the rest of the transaction.
- During clock cycle 3, the initiator asserts IRDY #, to indicate that it is ready to receive data.
- If the target has data ready to send then it asserts TRDY #. In our eg, the target sends 3 more words of data in clock cycle 4 to 6.
- The indicator uses FRAME # to indicate the duration of the burst, since it read 4 words, the initiator negates FRAME # during clock cycle 5.
- After sending the 4th word, the target disconnects its drivers and negates DEVSEL # during clock cycle 7.

Fig: A read operation showing the role of IRDY# / TRY#



- It indicates the pause in the middle of the transaction.
- The first and words are transferred and the target sends the 3rd word in cycle 5.
- But the indicator is not able to receive it. Hence it negates IRDY#.
- In response the target maintains 3rd data on AD line until IRDY is asserted again.
- In cycle 6, the indicator asserts IRDY. But the target is not ready to transfer the fourth word immediately; hence it negates TRDY in cycle 7. Hence it sends the 4th word and asserts TRDY# at cycle 8.

Device Configuration:

- The PCI has a configuration ROM memory that stores information about that device.
- The configuration ROM‟ s of all devices are accessible in the configuration address space.
- The initialization s/w read these ROM‟ s whenever the S/M is powered up or reset
- In each case, it determines whether the device is a printer, keyboard, Ethernet interface or disk controller.
- Devices are assigned address during initialization process and each device has an w/p signal called IDSEL # (Initialization device select) which has 21 address lines (AD) (AD to AD31).
- During configuration operation, the address is applied to AD i/p of the device and the corresponding AD line is set to and all other lines are set to 0.

AD11 - AD31 **Upper address line**

A00 - A10 **Lower address line** → Specify the type of the operation and to access the content of device configuration ROM.

The configuration software scans all 21 locations. PCI bus has interrupt request lines. Each device may requests an address in the I/O space or memory space

Electrical Characteristics:

- The connectors can be plugged only in compatible motherboards PCI bus can operate with either 5 – 33V power supply.
- The motherboard can operate with signaling system.

9. Explain in detail about SCSI bus.

SCSI Bus:- (Small Computer System Interface)

SCSI refers to the standard bus which is defined by ANSI (American National Standard Institute).

SCSI bus the several options. It may be,

Narrow bus - It has 8 data lines & transfers 1 byte at a time.
Wide bus - It has 16 data lines & transfer 2 byte at a time.
Single-Ended Transmission - Each signal uses separate wire.
HVD (High Voltage Differential) - It was 5v (TTL cells)
LVD (Low Voltage Differential) - It uses 3.3v

Because of these various options, SCSI connector may have 50, 68 or 80 pins. The data transfer rate ranges from 5MB/s to 160MB/s 320Mb/s, 640MB/s.

The transfer rate depends on,

- Length of the cable
- Number of devices connected.
- To achieve high transfer rat, the bus length should be 1.6m for SE signaling and 12m for LVD signaling.
- The SCSI bus us connected to the processor bus through the SCSI controller.
- The data are stored on a disk in blocks called sectors.
- Each sector contains several hundreds of bytes. These data will not be stored in contiguous memory location.
- SCSI protocol is designed to retrieve the data in the first sector or any other selected sectors.
- Using SCSI protocol, the burst of data are transferred at high speed.
- The controller connected to SCSI bus is of 2 types. They are,
    - Initiator
    - Target

Initiator:

It has the ability to select a particular target & to send commands specifying the operation to be performed.

They are the controllers on the processor side.

Target:

The disk controller operates as a target.

It carries out the commands it receive from the initiator. The initiator establishes a logical connection with the intended target.

Steps:

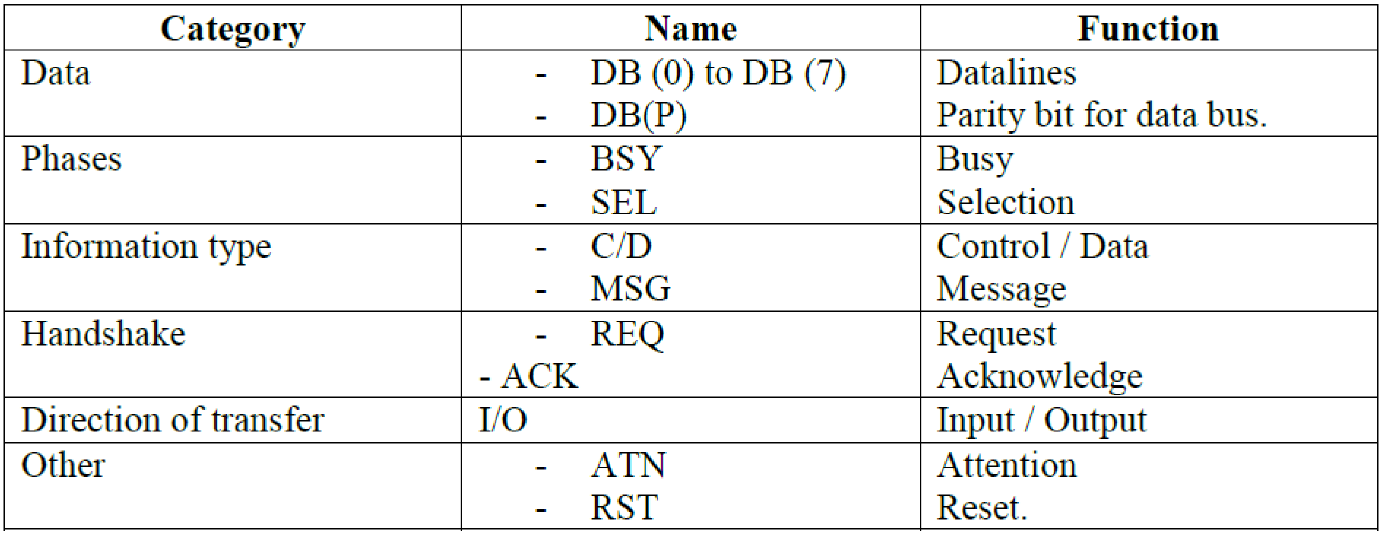
- Consider the disk read operation, it has the following sequence of events.
- The SCSI controller acting as initiator, contends process, it selects the target controller & hands over control of the bus to it.
- The target starts an output operation, in response to this the initiator sends a command specifying the required read operation.

- The target that it needs to perform a disk seek operation, sends a message to the initiator indicating that it will temporarily suspends the connection between them.
- Then it releases the bus.
- The target controller sends a command to disk drive to move the read head to the first sector involved in the requested read in a data buffer. When it is ready to begin transferring data to initiator, the target requests control of the bus. After it wins arbitration, it reselects the initiator controller, thus restoring the suspended connection.
- The target transfers the controls of the data buffer to the initiator & then suspends the connection again. Data are transferred either 8 (or) 16 bits in parallel depending on the width of the bus.
- The target controller sends a command to the disk drive to perform another seek operation. Then it transfers the contents of second disk sector to the initiator. At the end of this transfer, the logical connection b/w the two controller is terminated.
- As the initiator controller receives the data, if stores them into main memory using DMA approach.
- The SCSI controller sends an interrupt to the processor to inform it that the requested operation has been completed.

Bus Signals:-

- The bus has no address lines.
- Instead, it has data lines to identify the bus controllers involved in the selection / reselection / arbitration process.
- For narrow bus, there are 8 possible controllers numbered from 0 to 7.
- For a wide bus, there are 16 controllers.
- Once a connection is established b/w two controllers, these is no further need for addressing & the datalines are used to carry the data.

SCSI bus signals:



Category	Name	Function
Data	- DB (0) to DB (7) - DB(P)	Datalines Parity bit for data bus.
Phases	- BSY - SEL	Busy Selection
Information type	- C/D - MSG	Control / Data Message
Handshake	- REQ - ACK	Request Acknowledge
Direction of transfer	I/O	Input / Output
Other	- ATN - RST	Attention Reset.

- All signal names are proceeded by minus sign.
- This indicates that the signals are active or that the dataline is equal to 1, when they are in the low voltage state.

Phases in SCSI Bus:-

The phases in SCSI bus operation are,

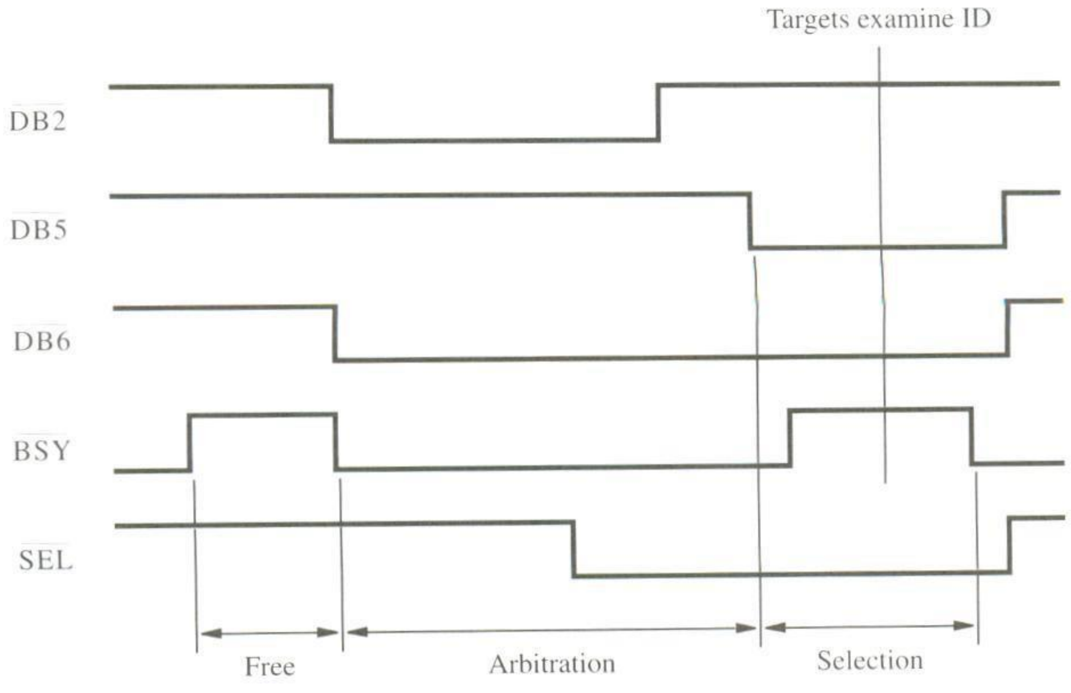
□ Arbitration
□ Selection
□ Information transfer
□ Reselection

Arbitration:-

- When the –BSY signal is in inactive state, the bus will he free & any controller can request the use of the bus.
- Since each controller may generate requests at the same time, SCSI uses distributed arbitration scheme.
- Each controller on the bus is assigned a fixed priority with controller 7 having the highest priority.
- When –BSY becomes active, all controllers that are requesting the bus examines the data lines & determine whether the highest priority device is requesting the bus at the same time.
- The controller using the highest numbered line realizes that it has won the arbitration process.
- At that time, all other controllers disconnect from the bus & wait for –BSY to become inactive again.

Fig:Arbitration and selection on the SCSI bus.Device 6 wins arbitration and select device 2

Selection:



Selection:
Here Device wons arbitration and it asserts –BSY and –DB6 signals.

The Select Target Controller responds by asserting –BSY.

This informs that the connection that it requested is established.

Reselection:
The connection between the two controllers has been reestablished,with the target in control the bus as required for data transfer to proceed.

10. What is USB? Explain in detail.

USB – Universal Serial Bus
USB supports 3 speed of operation. They are,

- Low speed (1.5Mb/s)
- Full speed (12mb/s)
- High speed (480mb/s)

The USB has been designed to meet the key objectives. They are,

□ It provide a simple, low cost & easy to use interconnection s/m that overcomes the difficulties due to the limited number of I/O ports available on a computer.
□ It accommodate a wide range of data transfer characteristics for I/O devices including telephone & Internet connections.
□ Enhance user convenience through**'Plug & Play'** mode of operation.

Port Limitation:-

- Normally the system has a few limited ports.
- To add new ports, the user must open the computer box to gain access to the internal expansion bus & install a new interface card.
- The user may also need to know to configure the device & the s/w.

Merits of USB:-
USB helps to add many devices to a computer system at any time without opening the computer box.
Device Characteristics:-

- The kinds of devices that may be connected to a cptr cover a wide range of functionality.
- The speed, volume & timing constrains associated with data transfer to & from devices varies significantly.

Eg:1 Keyboard □ Since the event of pressing a key is not synchronized to any other event in a computer system, the data generated by keyboard are called asynchronous.
The data generated from keyboard depends upon the speed of the human operator which is about 100bytes/sec.
Eg:2 Microphone attached in a cptr s/m internally / externally

The sound picked up by the microphone produces an analog electric signal, which must be converted into digital form before it can be handled by the cptr.

This is accomplished by sampling the analog signal periodically.

The sampling process yields a continuous stream of digitized samples that arrive at regular intervals, synchronized with the sampling clock. Such a stream is called isochronous (ie) successive events are separated by equal period of time.

If the sampling rate in „S‟ samples/sec then the maximum frequency captured by sampling process is s/2.

A standard rate for digital sound is 44.1 KHz.

Requirements for sampled Voice:-

It is important to maintain precise time (delay) in the sampling & replay process.

A high degree of jitter (Variability in sampling time) is unacceptable.

Eg-3:Data transfer for Image & Video:-

The transfer of images & video require higher bandwidth.

The bandwidth is the total data transfer capacity of a communication channel.

To maintain high picture quality, The image should be represented by about 160kb, & it is transmitted 30 times per second for a total bandwidth if 44MB/s.

Plug & Play:-

The main objective of USB is that it provides a plug & play capability.

The plug & play feature enhances the connection of new device at any time, while the system is operation.

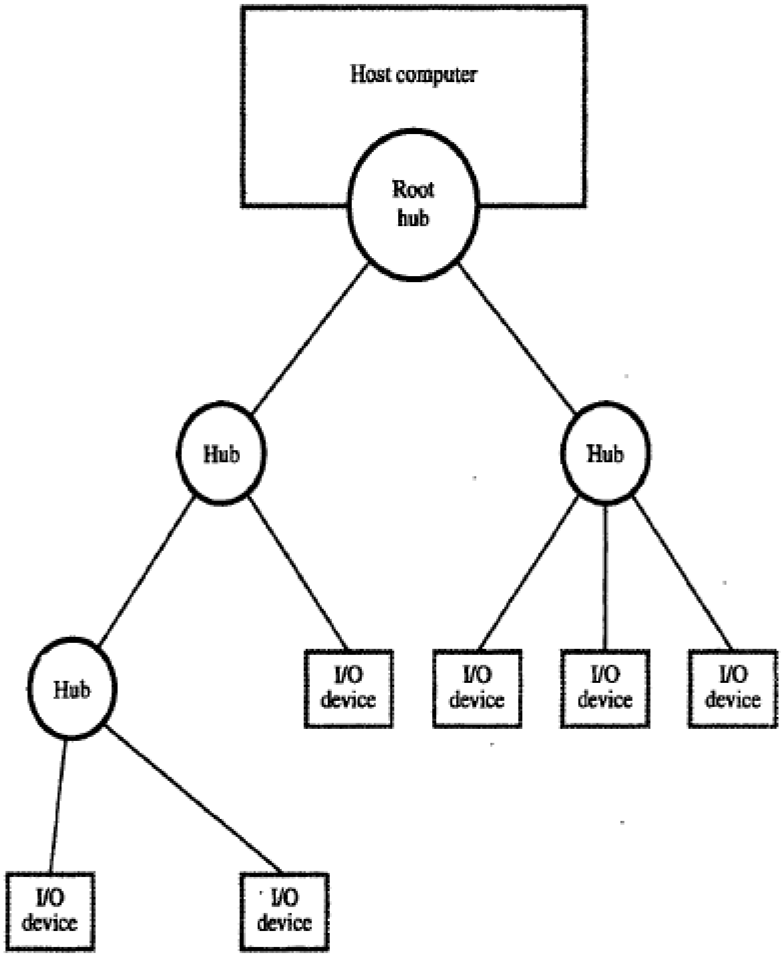
The system should,

□ Detect the existence of the new device automatically.
□ Identify the appropriate device driver s/w.
□ Establish the appropriate addresses.
□ Establish the logical connection for communication.

USB Architecture:-

- USB has a serial bus format which satisfies the low-cost & flexibility requirements.
- Clock & data information are encoded together & transmitted as a single signal.
- There are no limitations on clock frequency or distance arising form data skew, & hence it is possible to provide a high data transfer bandwidth by using a high clock frequency.
- To accommodate a large no/. of devices that can be added / removed at any time, the USB has the tree structure.

Fig:USB Tree Structure



- Each node of the tree has a device called hub, which acts as an intermediate control point b/w host & I/O devices.
- At the root of the tree, the „root hub‟ connects the entire tree to the host computer.
- The leaves of the tree are the I/O devices being served (eg. Keyboard, speaker, digital TV), which are called functions in USB terminology.
- The tree structure enables many devices to be connected while using only simple point-point serial links.
- Each hub has a port where devices maybe connected, including other hubs.
- In normal operation, a hub copies a message that it receives from its upstream connection to all its downstream ports. A message sent by host computer is broadcast to all I/O devices, but only the addressed device will respond to that message.

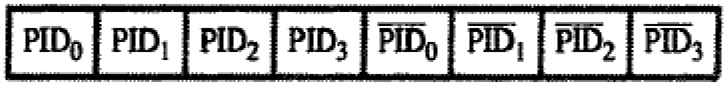
Addressing

- A device usually has several addressable locations to enable the software to send and receive control and status information and to transfer data.
- When a USB is connected to a host computer, its root hub is attached to the processor bus, where it appears as a single device.
- The host software communicates with individual devices attached to the USB by sending packets of information, which the root hub forwards to the appropriate device in the USB tree.
- Each device on the USB, whether it is a hub or an I/O device, is assigned a 7-bit address. This address is local to the USB tree and is not related in any way to the addresses used on the processor bus.

- A hub may have any number of devices and addresses are assigned arbitrarily.
- When a device is first connected to a hub or when it is powered on, it has the address 0.
- Locations in the device to or from which data transfer can take place, such as status, control and data registers are called endpoints. They are identified by a 4-bit number.

USB protocols

- The information transferred on the USB can be divided into two broad categories: control and data.
- Control packets perform tasks such as addressing a device to initiate data transfer, acknowledging that data have been received correctly, or indicating an error.
- Data packets carry information that is delivered to a device.
- A packet consists of one or more fields containing different kinds of information. The first field of any packet is called the packet identifier, PID, which identifies the type of that packet.
- There are 4 bits of information, but they are transmitted twice. The first time they are sent with their true values, and the second time with each bit complemented.

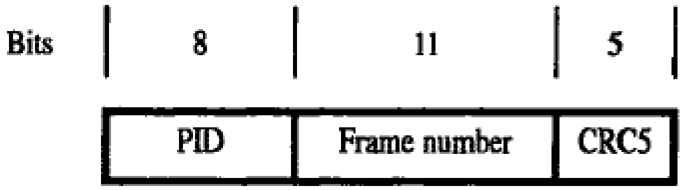


(a) Packet identifier field

Isochronous Traffic on USB

One of the key objectives of the USB is to support the transfer of isochronous data, such as sampled voice, in a simple way. Devices that generate or receive isochronous data require a time reference to control the sampling process. To provide this reference, transmission over the USB is divided into frames of equal length. A frame is 1 ms long for low- and full-speed data. The root hub generates a Start Of Frame control packet (SOF) precisely once every 1 ms to mark the beginning of a new frame.

The arrival of an SOF packet at any device constitutes a regular clock signal that the device can use for its own purposes. The main requirement for isochronous traffic is consistent timing.



(a) SOF Packet

Isochronous data are allowed only on full-speed and high-sped links. For high-speed links, SOF packet is repeated eight times at equal intervals within the 1 ms frame to create eight microframes of 125 µs each.

Electrical Characteristics

The cables used for USB connections consist of four wires.

Two used to carry power, +5v and Ground.

UNIT – IV

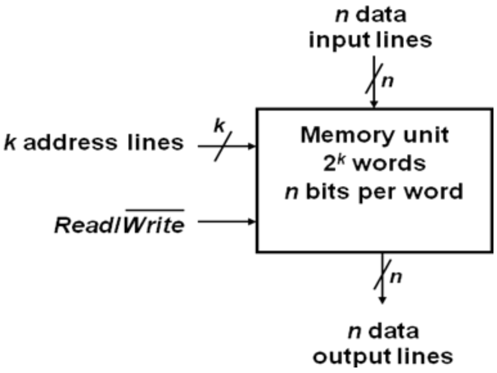
THE MEMORY SYSTEM: Some Basic Concepts, Semiconductor RAM Memories, Read-Only Memories, Speed, Size, and Cost, Cache Memories, Performance Considerations, Virtual memories, Memory Management requirements, Secondary Storage.

2 MARKS

1. **What is memory system?**

Every Computer contains several types of devices to store the instructions and data for its operation. These storage devices plus the algorithm implements by hardware and software needed to manage the stored information from the memory system of computer.

2. **Draw the block diagram for memory unit.**



3. **Define addressing scheme?**

Addressing scheme is a scheme used in any computer to determine and maximum size of the memory.

4. **What are the two registers involved in data transfer between the memory and the processor?**

The registers used to transfer data are,

MAR (Memory address register)

MDR (Memory data register).

5. **Give the classification of memory (Apr 11)**

a. CPU Register

b. Main memory

c. Secondary Memory

d. Cache.

6. **Define Static Memories and Dynamic Memories.**

Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memories. In Dynamic Memories such cells do not retain their state indefinitely.

7. What is read access time?

A basic performance measure is the average time to read a fixed amount of information for instance, one word from the memory. This parameter is called the read access time.

8. Define RAM

In storage location can be accessed in any order and access time is independent of the location being accessed, the memory is termed as random access memory

9. What are ROM & PROMs?

Memories whose content cannot be altered online if they can be altered at all are read only memories.

Semi conductor ROMs whose contents can be changed offline-with some difficulties is called PROMs.

10. Difference between static RAM and Dynamic RAM.

S.no	STATIC RAM	DYNAMIC RAM
1.	They are fast	They are slow
2.	They are very expensive	They are less expensive
3.	They require several transistors	They require less no several transistors
4.	They retain their state indefinitely	They do not retain their state indefinitely

11. Differentiate asynchronous DRAM with synchronous DRAM?

S.no.	asynchronous DRAM	synchronous DRAM
1.	The timing of the memory device is controlled asynchronously	The timing of the memory device controlled synchronously.
2	There is specialized memory controlled circuit that provides the necessary control information	The memory operations are synchronized with a clock signals.
3	Separate refresh circuit is not used	It uses the separate refresh circuit.

12. List the differences between SRAM AND DRAM?

SRAM: Static random access memory. It tends to be faster and they require no refreshing

DRAM: Dynamic random access memory. Data is stored in the form of charges. So continuous refreshing is needed.

13. What is volatile memory?

A memory is volatile if the loss of power destroys the stored information. Information can be stored indefinitely in a volatile memory by providing battery backup or other means to maintain a continuous supply of power.

14. What are the types of memory? / What are the categories of memories?

- SRAM (Satic Random Access Memory)
- DRAM (Dynamic Random Access Memory)

15. What is flash memory?

A recent semiconductor technology called flash memory of a same non-volatility as a PROM, but it can be done a bit at a time.

16. What is cache memory? (Apr 11)

Memory word are stored in cache data memory and are grouped into small pages called cache blocks or line. The contents of the caches data memory are thus copies of a set of main memory blocks.

17. Mention two system organizations for caches.

Two system organizations for caches are

- look aside
- look through

18. What is RAMBUS memory?

The key feature of Rambus technology is a fast signaling method used to transfer information between chip using narrow bus.

19. What is write-through protocol?

For write operation, the cache location and the main memory location are updated simultaneously

20. Give the difference between EEPROM and Flash memory?

The primary difference between EEPROM and flash memory is that flash restricts writes to multiple kilobytes blocks, increasing the memory capacity per chip by reducing area of control.

21. Differences between cache memory and virtual memory

- In caches, replacement is primarily controlled by the hardware. In VM, replacement is primarily controlled by the os.
- The Number of bits in the address determines the size of VM, where as cache size is independent of the address size.
- But there is only one class of cache.

22. Uses of Virtual Memory.

Protection: VM is often used to protect one program from others in the system

Base and Bounds: this method allows relocation. User processes cannot be allowed to change these registers, but the OS must be able to do so on a process switch.

23. What is meant by Interleaved Memory Or What is memory interleaving?

Banks of memory are often one word wide , so bus width need not be changed to access memory. However several independent areas of memory can be accessed simultaneously by using interleaved memory.

24. What is write back protocol?

In this scheme, only the block in cache is modified. The main memory when the block must be replaces in the cache. This requires the use of a dirty bit to keep track of blocks, that have been modified.

25. What is virtual memory and what are the benefits of virtual memory?

Virtual memory is a computer system technique which gives an application program the impression that it has contiguous working memory (an address space), while in fact it may be physically fragmented and may even overflow on to disk storage.

Benefits

- Programs can be larger than physical memory
- Entire program need not be in memory

26. Give the features of a ROM cell

- ROM is Read only memory
- It is usually non-volatile memory meaning that when you turn power off to the electronic device the ROM memory retains its contents.
- This is typically found on computer mother boards were start-up instructions are installed.
- ROM is permanent memory, so it never loses what is stored in it.

27. Define Locality of Reference.

Many applications continually reference large amounts of data. Often, the link to this data is slow, as in the cases of primary or secondary memory references, database queries, or network requests. This leads to poor performance in the application. To improve application efficiency by exploiting the principle of Locality of Reference also Known as Caching

28. What is Translation Look aside Buffer? Or what is TLB?

A translation look aside buffer (TLB) is a cache that memory management hardware uses to improve virtual address translation speed. All current desktop, notebook, and server processors use a TLB to map virtual and physical address spaces, and it is nearly always present in any hardware which utilizes virtual memory.

29. Define data transfer rate.

The data transfer rate (DTR) is the amount of digital data that is moved from one place to another in a given time.

30. Define memory access time?

The time required to access one word is called the memory access time. Or It is the time that elapses between the initiation of an operation and the completion of that operation.

31. Define memory cycle time?

It is the minimum time delay required between the initiations of two successive memory operations.

Eg. The time between two successive read operations.

32. When is a memory unit called as RAM?

A memory unit is called as RAM if any location can be accessed for a read or writes operation in some fixed amount of time that is independent of the location's address.

33. What is MMU?

MMU is the Memory Management Unit. It is a special memory control circuit used for implementing the mapping of the virtual address space onto the physical memory.

- Programs can be larger than physical memory
- Entire program need not be in memory

34. Define memory cell?

A memory cell is capable of storing one bit of information. It is usually organized in the form of an array.

35. What is a word line?

In a memory cell, all the cells of a row are connected to a common line called as word line.

36. What are the Characteristics of semiconductor RAM memories?

- ➢ They are available in a wide range of speeds.
- ➢ Their cycle time range from 100ns to less than 10ns.
- ➢ They replaced the expensive magnetic core memories.
- ➢ They are used for implementing memories.

37. Why SRAMs are said to be volatile?

Because of their contents are lost when power is interrupted. So SRAMs are said to be volatile.

38. What are the Characteristics of SRAMs?

- ➢ SRAMs are fast.
- ➢ They are volatile.
- ➢ They are of high cost.
- ➢ Less density.

39. What are the Characteristics of DRAMs?

- ➢ Low cost.
- ➢ High density.
- ➢ Refresh circuitry is needed.

40. Define Refresh Circuit?

It is a circuit which ensures that the contents of a DRAM are maintained when each row of cells are accessed periodically.

41. Define Memory Latency?

It is used to refer to the amount of time it takes to transfer a word of data to or from the memory.

42. What are asynchronous DRAMs?

In asynchronous DRAMs, the timing of the memory device is controlled asynchronously. A specialized memory controller circuit provides the necessary control signals RAS and CAS that govern the timing. The processor must take into account the delay in the response of the memory. such memories are asynchronous DRAMs .

43. What are synchronous DRAMs?

Synchronous DRAMs are those whose operation is directly synchronized with a clock signal.

44. Define Bandwidth?

When transferring blocks of data, it is of interest to know how much time is needed to transfer an entire block. Since blocks can be variable in size it is useful to define a performance measure in terms of number of bits or bytes that can be transferred in one second. This measure is often referred to as the memory bandwidth.

45. What is double data rate SDRAMs? Or what is DDR SDRAM?

Double data rates SDRAMs are those which can transfer data on both edges of the clock and their bandwidth is essentially doubled for long burst transfers.

46. Differentiate static RAM and dynamic RAM?

Static RAM

- They are fast
- They are very expensive
- They retain their state indefinitely.
- They require several transistors
- Low density

Dynamic RAM

- They are slow
- They are less expensive
- They don’t retain their state indefinitely
- They require less no transistors.
- High density

47. What is Ram Bus technology?

The key feature of Ram bus technology is a fast signaling method used to transfer information between chips. Instead of using signals that have voltage levels of either 0 or Vsupply to represent the logic values, the signals consist of much smaller voltage swings around a reference voltage , vref. Small voltage swings make it possible to have short transition times, which allows for a high speed of transmission.

48. What are RDRAMs?

RDRAMs are Rambus DRAMs. Rambus requires specially designed memory chips. These chips use cell arrays based on the standard DRAM technology. Multiple banks of cell arrays are used to access more than one word at a time. Circuitry needed to interface to the Rambus channel is included on the chip. Such chips are known as RDRAMs.

49. What are the special features of Direct RDRAMs?

- It is a two channel Rambus.
- It has 18 data lines intended to transfer two bytes of data at a time.
- There are no separate address lines.

50. What are the disadvantages of EPROM?

The chip must be physically removed from the circuit for reprogramming and its entire contents are erased by the ultraviolet light.

51. Differentiate flash devices and EEPROM devices.

Flash devices

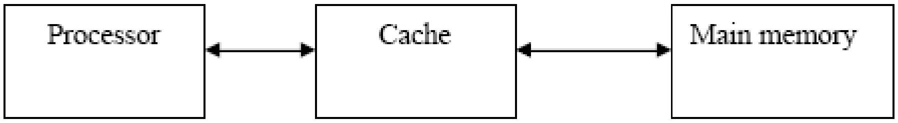
- It is possible to read the contents of a single cell, but it is only possible to write an entire block of cells.
- Greater density which leads to higher capacity.
- Lower cost per bit.
- Consumes less power in their operation and makes it more attractive for use in portable equipments that is battery driven.

EEPROM devices

- It is possible to read and write the contents of a single cell.
- Relatively more cost
- Consumes more power.

52. What is cache memory?

It is a small, fast memory that is inserted between the larger, slower main memory and the processor. It reduces the memory access time.



53. Define flash memory?

It is an approach similar to EEPROM technology. A flash cell is based on a single transistor controlled by trapped charge just like an EEPROM cell.

54. What are the two aspects of locality of reference? Define them.

Two aspects of locality of reference are temporal aspect and spatial aspect. Temporal aspect is that a recently executed instruction is likely to be executed again very soon.

The spatial aspect is that instructions in close proximity to a recently executed instruction are also to be executed soon.

55. Define cache line.

Cache block is used to refer to a set of contiguous address locations of some size. Cache block is also referred to as cache line.

56. What are the two ways in which the system using cache can proceed for a write operation?

- Write through protocol technique.
- Write-back or copy back protocol technique.

57. What is write-through protocol?

For a write operation using write through protocol during write hit: the cache location and the main memory location are updated simultaneously.

For a write miss, the information is written directly to the main memory.

58. What is write-back or copy back protocol?

For a write operation using this protocol during **write hit:** the technique is to update only the cache location and to mark it as updated with an associated flag bit, often called the dirty or modified bit. The main memory location of the word is updated later, when the block containing this marked word is to be removed from the cache to make room for a new block.

For a write miss: the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

59. What are the mapping technique? Discuss the different mapping techniques used in cache memory

- Direct mapping
- Associative mapping
- Set Associative mapping

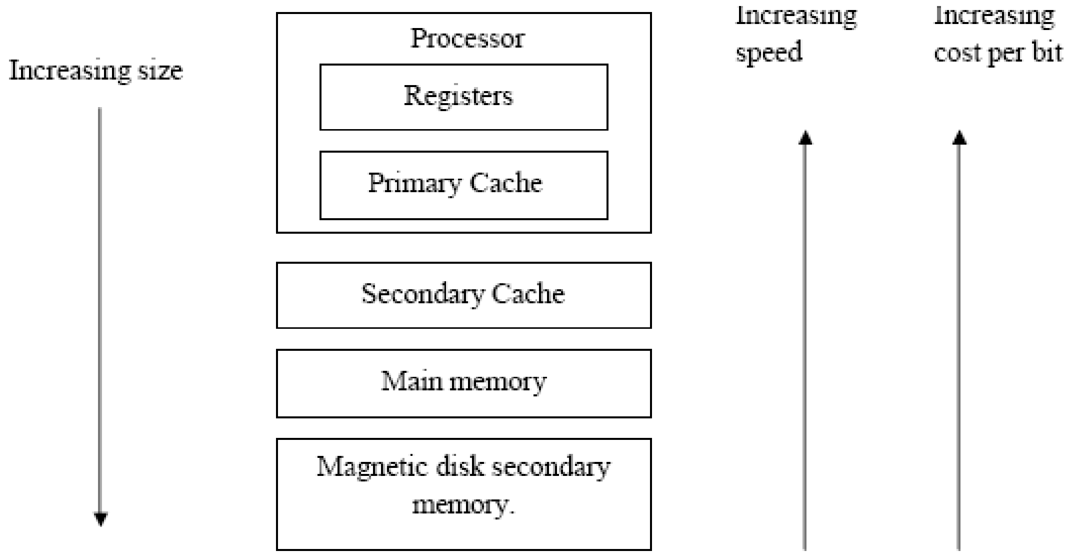
60. What is a hit?

A successful access to data in cache memory is called hit.

61. Define hit rate?

The number of hits stated as a fraction of all attempted access.

62. Describe the memory hierarchy?



63. Define miss rate?

It is the number of misses stated as a fraction of attempted accesses.

64. Define access time for magnetic disks?

The sum of seek time and rotational delay is called as access time for disks. Seek time is the time required to move the read/write head to the proper track. Rotational delay or latency is the amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector passes under the read/write head.

65. What is the formula for calculating the average access time experienced by the processor?

t_{ave}=hc +(1-h)M

Where,

h =Hit rate

M=miss penalty

C=Time to access information in the cache.

66. What is the formula for calculating the average access time experienced by the processor in a system with two levels of caches?

tave =h1c1(1-h1)h2c2+(1-h1)(1-h2)M

where,

h1=hit rate in L1 cache

h2=hit rate in L2 cache

C1=Time to access information in the L1 cache.

C2=Time to access information in the L2 cache.

67. What are pages?

All programs and data are composed of fixed length units called pages. each consists of blocks of words that occupies contiguous locations in main memory.

68. What is replacement algorithm?

When the cache is full and a memory word that is not in the cache is referenced, the cache control hardware must decide which block should be removed to create space for the new block that contains the reference word .The collection of rules for making this decision constitutes the replacement algorithm.

69. What is meant by internal and external fragmentation?

Fragmentation occurs in a dynamic memory allocation system when many of the free blocks are too small to satisfy any request.

External Fragmentation: External Fragmentation happens when a dynamic memory allocation algorithm allocates some memory and a small piece is left over that cannot be effectively used. If too much external fragmentation occurs, the amount of usable memory is drastically reduced. Total memory space exists to satisfy a request, but it is not contiguous.

Internal Fragmentation: Internal fragmentation is the space wasted inside of allocated memory blocks because of restriction on the allowed sizes of allocated blocks. Allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

70. What is content addressable memory and what are the advantages of this memory?

Content-addressable memory, also referred to as associative memory or abbreviated CAM, is a mechanism for storing information that can be retrieved based on its content, not its storage location.

Advantage

It is typically used for high-speed storage and retrieval of fixed content, such as documents stored for compliance with government regulations.

71. Define interleaving

Cell array can be organized in two banks. Each bank can be accessed separately. So the consecutive words of given block are stored in different banks. It is known as interleaving of words. It increases the transfer rate.

72. Define SIMM and DIMM

SIMM –> Single In – line memory modules.

DIMM-> Dual In – line memory modules.

SIMM means single in line memory module and DIMM means dual in line memory module. These two large memories are created by using the DRAM. This module is an assembly of several memory chips on a separate small board that plugs vertically into a single socket on the mother board.

73.What is Memory controller?

A memory controller is a circuit which is interposed between the processor and the dynamic memory. It is used for performing multiplexing of address bits.

74. Draw backs present in the DRAM.

All dynamic memories have to be refreshed and it does not have a refreshing capability. So the memory controller has to provide all the information needed to control the refreshing operation. This increases the over head of the controller circuit.

75. How many memory chips are needed to construct 2M*16 memory system using 512K * 8 static memory chips?

4096/512 =16 chips 16 memory chips are needed to construct 2M*16 memory system using 512K * 8 static memory chips

76. How is disk access time calculated?

Disk access time can be calculated by adding up the 'seek time' and the average 'latency time'

- **Seek time**: the time needed for the read/write arm to look for the desired track
- **Latency time** (also called rotational delay): the time it takes for the desired sector on the track to spin around to the read/write arm (since the piece of data required might be on the other side of the disk relative to the read/write at the moment).
- **Transfer time**: the time a hard disk drive needs to read and transmit one block of data
- Disk access time in hard disk drives is measured in milliseconds
- Even though this might seem fast, CPUs are still able to calculate much faster than this, causing hard disk drives to be slow in comparison

Disk access time = seek time + latency time + transfer time

77. What is the use of EEPROM?

- EEPROM stands for Electrically Erasable Programmable Read-Only Memory
- It is a type of non-volatile memory used in computers and other electronic devices to store small amounts of data that must be saved when power is removed
- e.g., calibration tables or device configuration.

78. State the hardware needed to implement the LRU in replacement algorithm.

- The hardware is equipped with a counter (typically 64 bits). After each instruction the counter is incremented.
- In addition, each page table entry has a field large enough to accommodate the counter. Every time the page is referenced the value from the counter is copied to the page table field. When a page fault occurs the operating system inspects all the page table entries and selects the page with the lowest counter. This is the page that is evicted as it has not been referenced for the longest time.

79. What is DDR SDRAM?

- Double data rate synchronous dynamic random-access memory (DDR SDRAM) is a class of memory integrated circuits used in computers.
- The SDRAM transfer data on the both edges of the clock, their bandwidth is essentially doubled for long burst transfers. Such devices are known as double-data-rate SDRAMs.

80. An address space is specified by 24 bits and the corresponding memory space by 16 bits: How many words are there in the virtual memory and in the main memory?OR

An address space is specified by 24 bits & the corresponding memory space is 16
bits.

a) How many words are there in address space?

b) How many words are there in memory space?

c) If a page has 2k words, how many pages & blocks are in the system?

Solution:-

a) Address space = 24 bits

224 = 24.220 = 16M words

b) Memory space: 16 bits

216 = 64k words

c) page consists of 2k words

Number of pages in add space = 16M/2K = 8000

Number of blocks = 64k/2k = 32 blocks

81. What is data stripping?

In a write system a single large file is stored in several separate disk units by breaking the file up in to a number of small pieces and stored these pieces on different disk.

82. Define hit ratio (Nov 13)

Hit rate=No of hits/no of bus cycles *100%

83. Define the terms hit, miss and ratio with respect to cache

Cache is a small high-speed memory. Stores data from some frequently used addresses (of main memory).

Cache hit: Data found in cache. Results in data transfer at maximum speed.

Cache miss: Data not found in cache. Processor loads data from M and copies into cache. This results in extra delay, called miss penalty.

Hit ratio = percentage of memory accesses satisfied by the cache.

Miss ratio = 1-hit ratio

84. Which type of memory provides backup storage? (Nov 12)

External storage provides data back-up. It includes floppy disks, tapes.

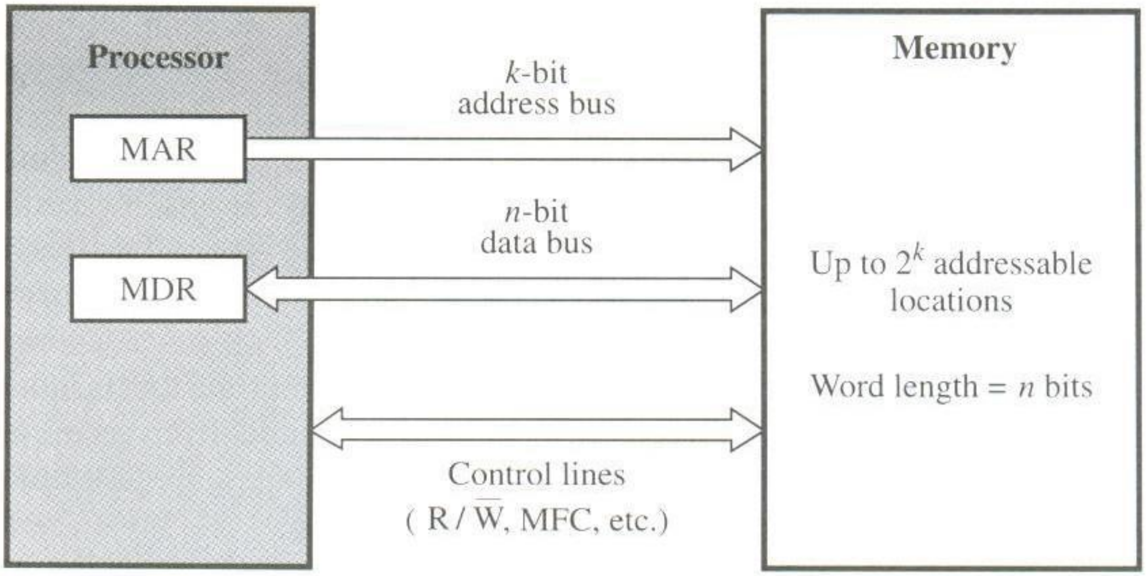
11 MARKS

1. Describe in detail the concepts of memory.

The maximum size of the memory that can be used in any computer is determined by the addressing scheme.

Address	**Memory Locations**
16 Bit	216 = 64 K
32 Bit	232 = 4G (Giga)
40 Bit	240 = IT (Tera)

Fig: Connection of Memory to Processor:



If MAR is k bits long and MDR is n bits long, then the memory may contain upto 2K addressable locations and the n-bits of data are transferred between the memory and processor. This transfer takes place over the processor bus.

The processor bus has,

- Address Line
- Data Line
- Control Line (R/W, MFC – Memory Function Completed)

➢ The control line is used for co-ordinating data transfer.

- The processor reads the data from the memory by loading the address of the required memory location into MAR and setting the R/W line to 1.
- The memory responds by placing the data from the addressed location onto the data lines and confirms this action by asserting MFC signal.
- Upon receipt of MFC signal, the processor loads the data onto the data lines into MDR register.
- The processor writes the data into the memory location by loading the address of this location into MAR and loading the data into MDR sets the R/W line to 0.

Memory Access Time → It is the time that elapses between the intiation of an Operation and the completion of that operation.
Memory Cycle Time → It is the minimum time delay that required between the initiation of the two successive memory operations.

In RAM, if any location that can be accessed for a Read/Write operation in fixed amount of time, it is independent of the location‟ s address.

Cache Memory:

- It is a small, fast memory that is inserted between the larger slower main memory and the processor.
- It holds the currently active segments of a pgm and their data.

Virtual memory:

- The address generated by the processor does not directly specify the physical locations in the memory.
- The address generated by the processor is referred to as a virtual / logical address.
- The virtual address space is mapped onto the physical memory where data are actually stored.
- The mapping function is implemented by a special memory control circuit is often called the memory management unit.
- Only the active portion of the address space is mapped into locations in the physical memory.
- The remaining virtual addresses are mapped onto the bulk storage devices used, which are usually magnetic disk.
- As the active portion of the virtual address space changes during program execution, the memory management unit changes the mapping function and transfers the data between disk and memory.
- Thus, during every memory cycle, an address processing mechanism determines whether the addressed in function is in the physical memory unit.
- If it is, then the proper word is accessed and execution proceeds.
- If it is not, a page of words containing the desired word is transferred from disk to memory.

- This page displaces some page in the memory that is currently inactive.

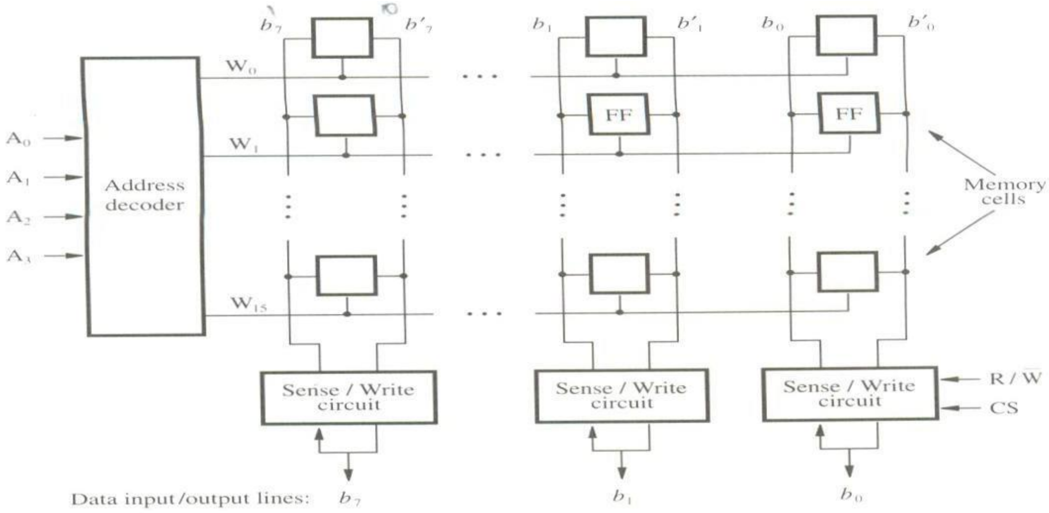
2. Describe in detail about Semiconductor RAM memories (Apr 11)

Semi-Conductor memories are available is a wide range of speeds. Their cycle time ranges from 100ns to 10ns

Internal Organization Of Memory Chips

- Memory cells are usually organized in the form of array, in which each cell is capable of storing one bit of information.
- Each row of cells constitute a memory word and all cells of a row are connected to a common line called as word line.
- The cells in each column are connected to Sense / Write circuit by two bit lines.
- The Sense / Write circuits are connected to data input or output lines of the chip.
- During a write operation, the sense / write circuit receive input information and store it in the cells of the selected word.

Fig: Organization of bit cells in a memory chip



- The data input and data output of each senses / write ckt are connected to a single bidirectional data line that can be connected to a data bus of the cptr.

R / W - Specifies the required operation.

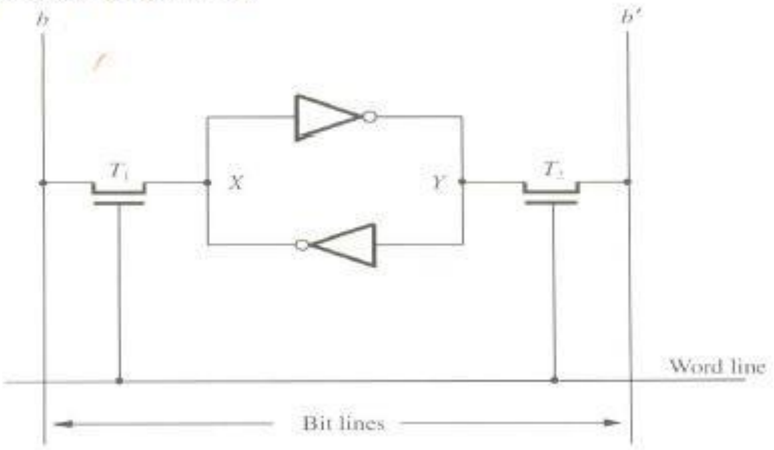
CS - **Chip Select** input selects a given chip in the multi-chip memory system

Bit Organization	**Requirement of external connection for address, data and control lines**
128 (16x8)	14
(1024) 128x8(1k)	19

Static Memories:

Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memory.

Fig:Static RAM cell



- Two inverters are cross connected to form a batch
- The batch is connected to two bit lines by transistors T1 and T2.
- These transistors act as switches that can be opened / closed under the control of the word line.
- When the wordline is at ground level, the transistors are turned off and the latch retain its state.

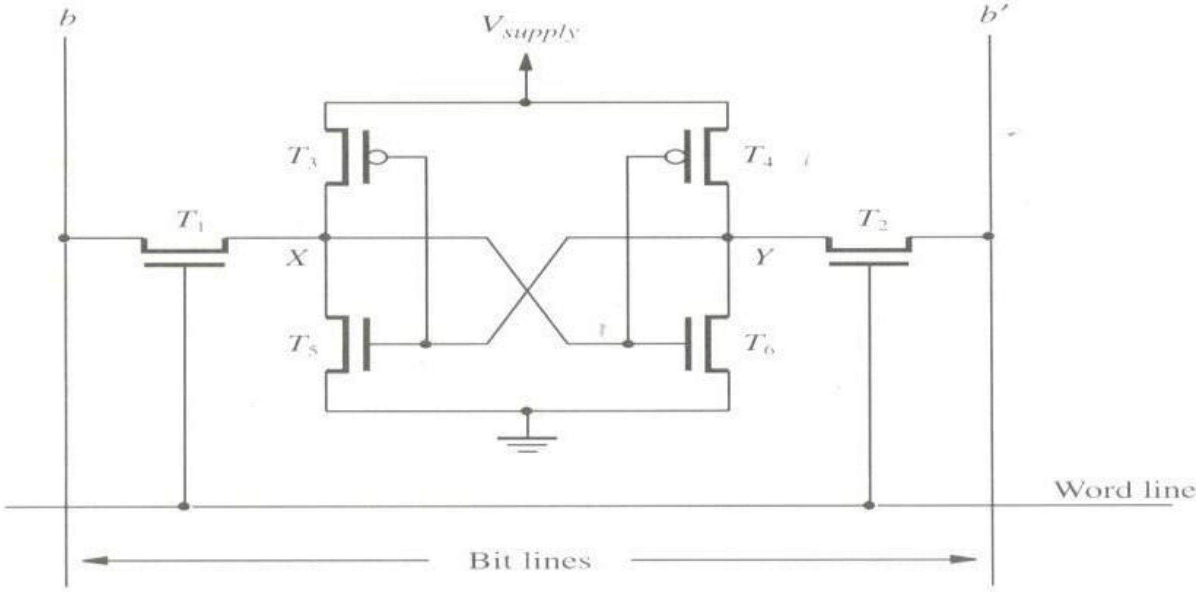
Read Operation:

- In order to read the state of the SRAM cell, the word line is activated to close switches T1 and T2.
- If the cell is in state 1, the signal on bit line b is high and the signal on the bit line b is low.Thus b and b are complement of each other.
- Sense / write circuit at the end of the bit line monitors the state of b and b‟ and set the output accordingly.

Write Operation:

- The state of the cell is set by placing the appropriate value on bit line b and its complement on b and then activating the word line. This forces the cell into the corresponding state.
- The required signal on the bit lines are generated by Sense / Write circuit.

Fig:CMOS cell (Complementary Metal oxide Semi Conductor):



- Transistor pairs (T3, T5) and (T4, T6) form the inverters in the latch.
- In state 1, the voltage at point X is high by having T5, T6 on and T4, T5 are OFF.
- Thus T1, and T2 returned ON (Closed), bit line b and b will have high and low signals respectively.
- The CMOS requires 5V (in older version) or 3.3.V (in new version) of power supply voltage.
- The continuous power is needed for the cell to retain its state

Merit :

It has low power consumption because the current flows in the cell only when the cell is being activated accessed.

Static RAM‟ s can be accessed quickly. It access time is few nanoseconds.

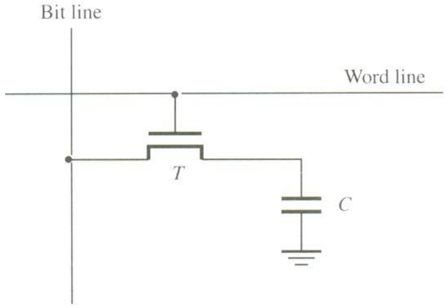
Demerit:

SRAM‟ s are said to be volatile memories because their contents are lost when the power is interrupted.

Asynchronous DRAMS:-

- Less expensive RAM‟ s can be implemented if simplex calls are used such cells cannot retain their state indefinitely. Hence they are called Dynamic RAM’s (DRAM).
- The information stored in a dynamic memory cell in the form of a charge on a capacitor and this charge can be maintained only for tens of Milliseconds.
- The contents must be periodically refreshed by restoring by restoring this capacitor charge to its full value.

Fig:A single transistor dynamic Memory cell

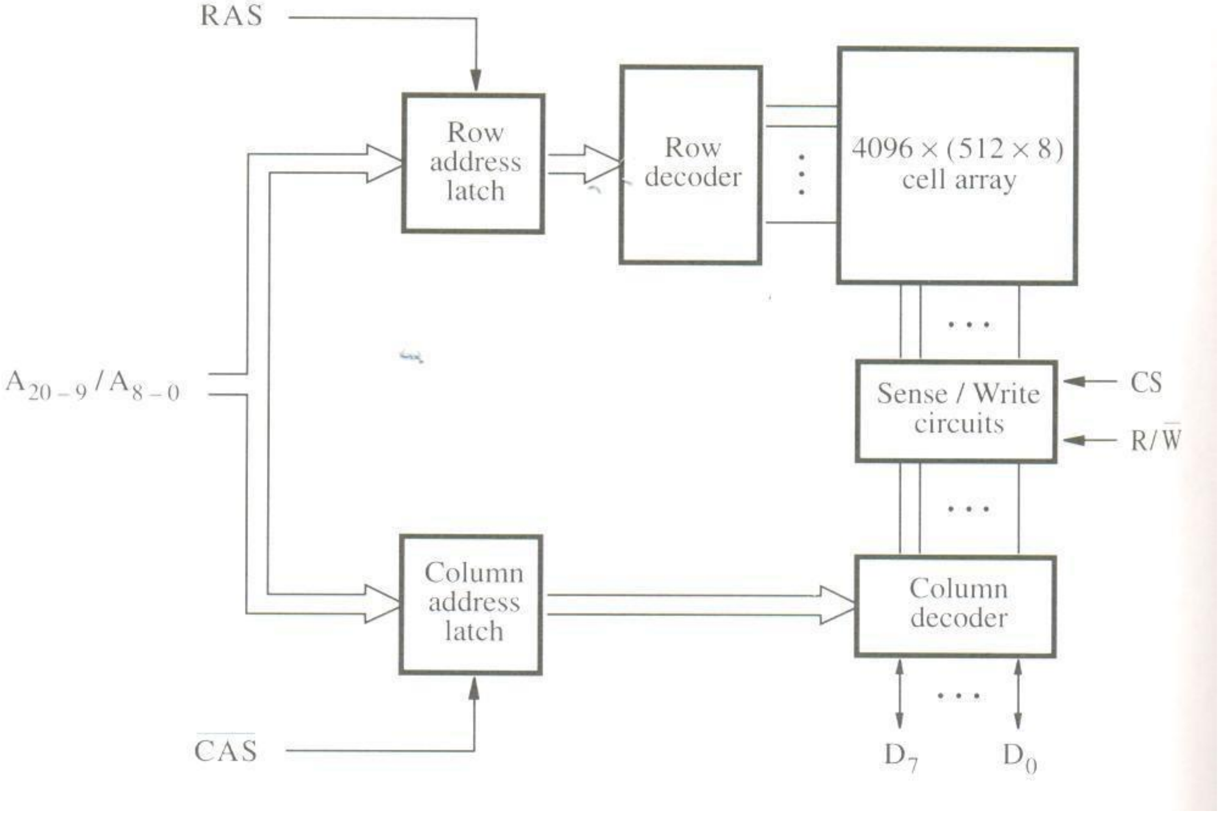


- In order to store information in the cell, the transistor T is turned „on‟ & the appropriate voltage is applied to the bit line, which charges the capacitor.
- After the transistor is turned off, the capacitor begins to discharge which is caused by the capacitor‟ s own leakage resistance.
- Hence the information stored in the cell can be retrieved correctly before the threshold value of the capacitor drops down.

- During a read operation, the transistor is turned „on‟ & a sense amplifier connected to the bit line detects whether the charge on the capacitor is above the threshold value.
  - If charge on capacitor > threshold value -> Bit line will have logic value „1‟ .

- If charge on capacitor < threshold value -> Bit line will set to logic value „0‟ .

Fig:Internal organization of a 2M X 8 dynamic Memory chip.



DESCRIPTION:

- The 4 bit cells in each row are divided into 512 groups of 8.
- 21 bit address is needed to access a byte in the memory(12 bit□ To select a row,9 bit□ Specify the group of 8 bits in the selected row).

- A8-0 □ Row address of a byte.
- A20-9 □ Column address of a byte.
- During Read/ Write operation ,the row address is applied first. It is loaded into the row address latch in response to a signal pulse on Row Address Strobe(RAS) input of the chip.
- When a Read operation is initiated, all cells on the selected row are read and refreshed.
- Shortly after the row address is loaded,the column address is applied to the address pins & loaded into Column Address Strobe(CAS).
- The information in this latch is decoded and the appropriate group of 8 Sense/Write circuits are selected.
- R/W =1(read operation)□ The output values of the selected circuits are transferred to the data lines D0 - D7.

- R/W =0(write operation)□ The information on D0- D7 are transferred to the selected circuits.

- RAS and CAS are active low so that they cause the latching of address when they change from high to low. This is because they are indicated by RAS & CAS.

- To ensure that the contents of a DRAM „s are maintained, each row of cells must be accessed periodically.
- Refresh operation usually perform this function automatically.
- A specialized memory controller circuit provides the necessary control signals RAS & CAS, that govern the timing.
- The processor must take into account the delay in the response of the memory. Such memories are referred to as Asynchronous DRAM's**.**

Fast Page Mode:

- Transferring the bytes in sequential order is achieved by applying the consecutive sequence of column address under the control of successive CAS signals.
- This scheme allows transferring a block of data at a faster rate. The block of transfer capability is called as Fast Page Mode.

Synchronous DRAM:

- Here the operations e directly synchronized with clock signal.
- The address and data connections are buffered by means of registers.
- The output of each sense amplifier is connected to a latch.
- A Read operation causes the contents of all cells in the selected row to be loaded in these latches
- Data held in the latches that correspond to the selected columns are transferred into the data output register, thus becoming available on the data output pins.

.

Fig:Synchronous DRAM

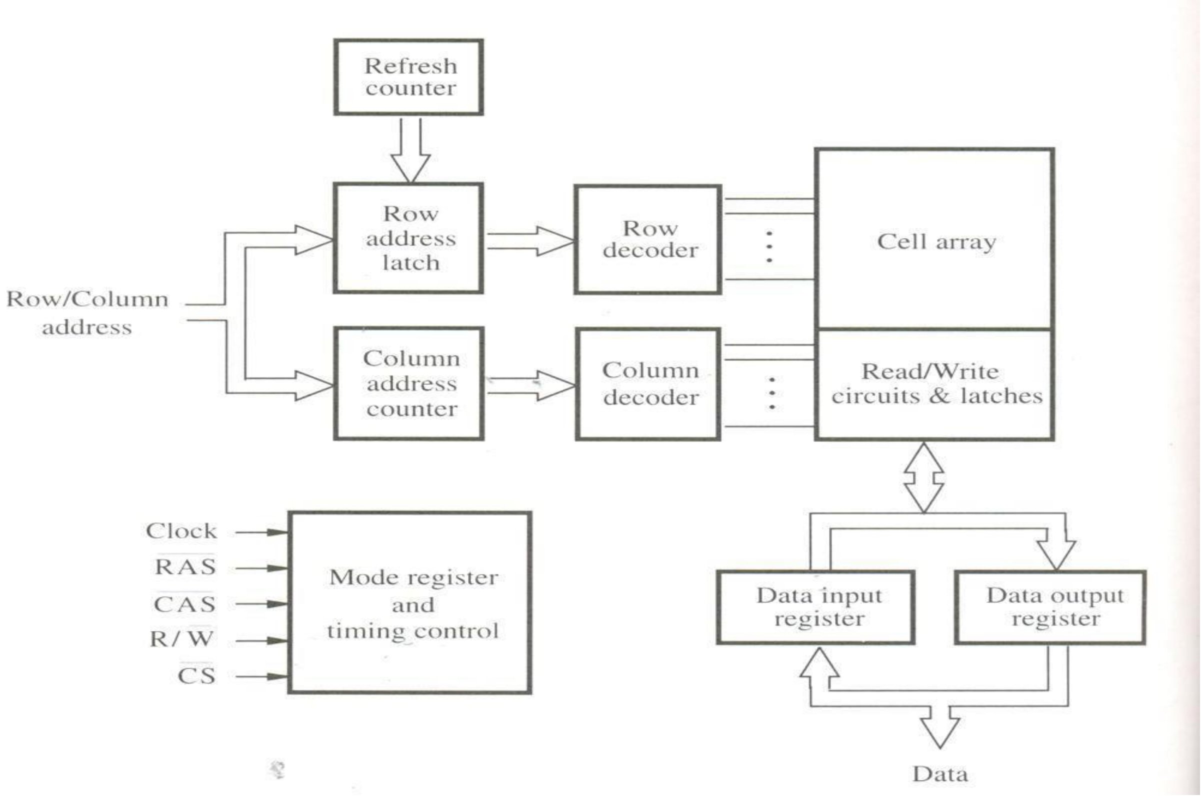
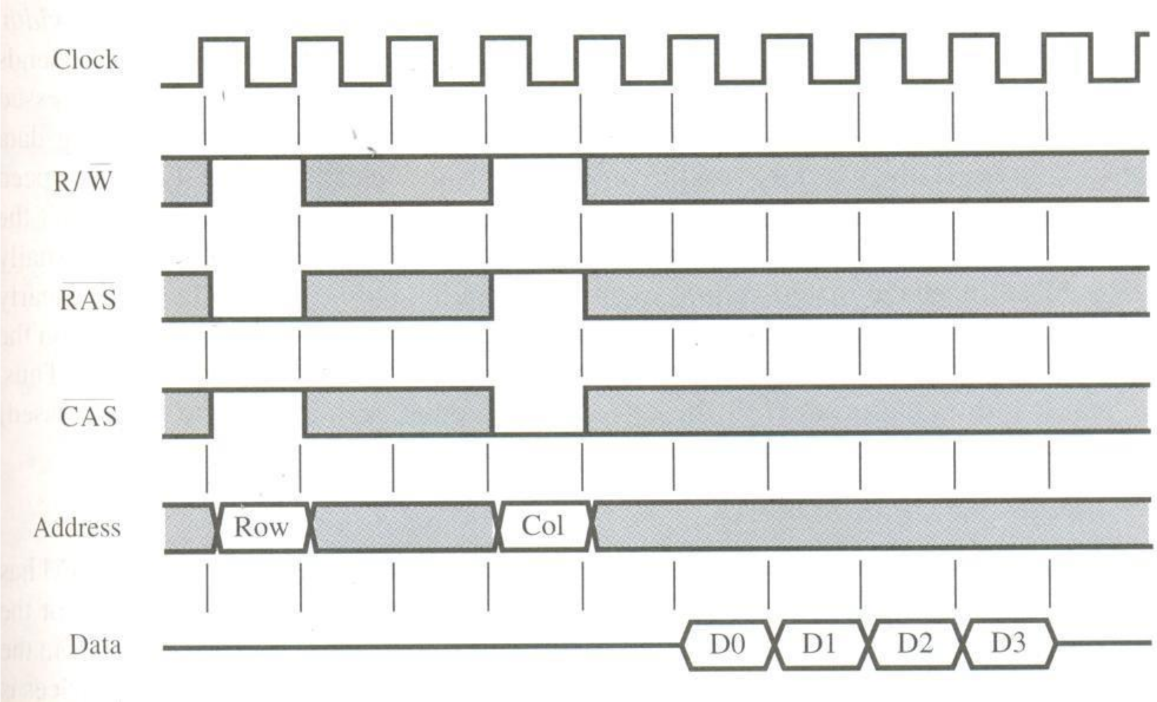


Fig:Timing Diagram Burst Read of Length 4 in an SDRAM



- First ,the row address is latched under control of RAS signal.
- The memory typically takes 2 or 3 clock cycles to activate the selected row.
- Then the column address is latched under the control of CAS signal.
- After a delay of one clock cycle,the first set of data bits is placed on the data lines.
- The SDRAM automatically increments the column address to access the next 3 sets of bits in the selected row, which are placed on the data lines in the next 3 clock cycles.

Latency & Bandwidth:

A good indication of performance is given by two parameters.They are,

- Latency
- Bandwidth

Latency:

It refers to the amount of time it takes to transfer a word of data to or from the memory.

For a transfer of single word,the latency provides the complete indication of memory performance.

For a block transfer,the latency denote the time it takes to transfer the first word of data.

Bandwidth:

It is defined as the number of bits or bytes that can be transferred in one second

Bandwidth mainly depends upon the speed of access to the stored data & on the number of bits that can be accessed in parallel.

Double Data Rate SDRAM(DDR-SDRAM):

- The standard SDRAM performs all actions on the rising edge of the clock signal.
- The double data rate SDRAM transfer data on both the edges(loading edge, trailing edge).

- The Bandwidth of DDR-SDRAM is doubled for long burst transfer.
- To make it possible to access the data at high rate , the cell array is organized into two banks.
- Each bank can be accessed separately.
- Consecutive words of a given block are stored in different banks.
- Such interleaving of words allows simultaneous access to two words that are transferred on successive edge of the clock.

Larger Memories:

Dynamic Memory System:

- The physical implementation is done in the form of Memory Modules.
- If a large memory is built by placing DRAM chips directly on the main system printed circuit board that contains the processor ,often referred to as Motherboard;it will occupy large amount of space on the board.
- These packaging consideration have led to the development of larger memory units known as SIMM‟ s & DIMM‟ s .

- SIMM-Single Inline memory Module
- DIMM-Dual Inline memory Module
- SIMM & DIMM consists of several memory chips on a separate small board that plugs vertically into single socket on the motherboard.

MEMORY SYSTEM CONSIDERATION:

To reduce the number of pins, the dynamic memory chips use multiplexed address inputs.

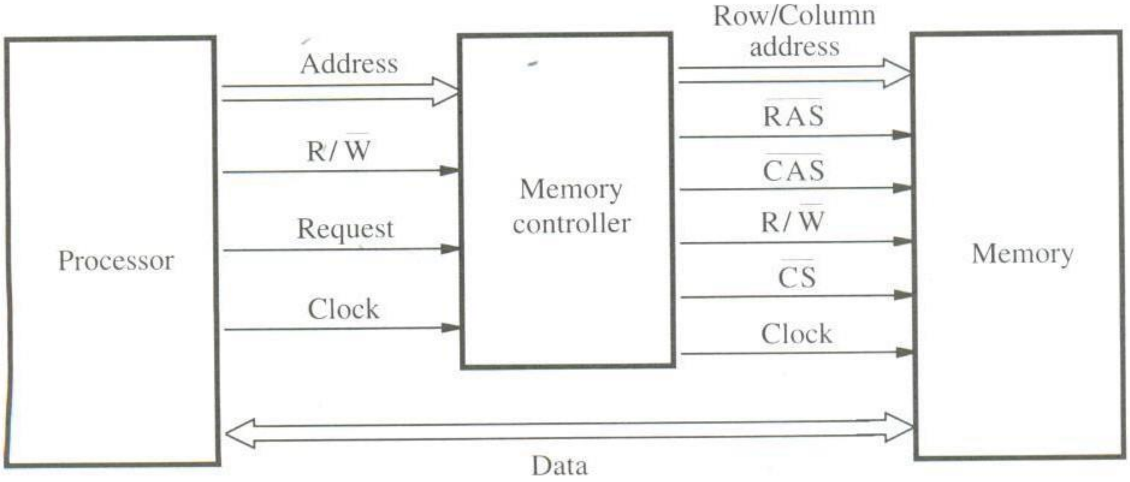
The address is divided into two parts.They are,

High Order Address Bit(Select a row in cell array & it is provided first and latched into memory chips under the control of RAS signal).

Low Order Address Bit(Selects a column and they are provided on same address pins and latched using CAS signals).

The Multiplexing of address bit is usually done by Memory Controller Circuit.

Fig:Use of Memory Controller



- The Controller accepts a complete address & R/W signal from the processor, under the control of a Request signal which indicates that a memory access operation is needed.
- The Controller then forwards the row & column portions of the address to the memory and generates RAS &CAS signals.
- It also sends R/W &CS signals to the memory.
- The CS signal is usually active low, hence it is shown as CS.

Refresh Overhead:

All dynamic memories have to be refreshed.

In DRAM ,the period for refreshing all rows is 16ms whereas 64ms in SDRAM.

Eg:Given a cell array of 8K(8192).
Clock cycle=4
Clock Rate=133MHZ
No of cycles to refresh all rows =8192*4
=32,768
Time needed to refresh all rows=32768/133*106
=246*10-6 sec
=0.246sec
Refresh Overhead =0.246/64
Refresh Overhead =0.0038

Rambus Memory:

- The usage of wide bus is expensive.
- Rambus developed the implementation of narrow bus.
- Rambus technology is a fast signaling method used to transfer information between chips.
- Instead of using signals that have voltage levels of either 0 or Vsupply to represent the logical values, the signals consists of much smaller voltage swings around a reference voltage Vref.

- .The reference Voltage is about 2V and the two logical values are represented by 0.3V swings above and below Vref..
- This type of signaling is generally is known as Differential Signalling.
- Rambus provides a complete specification for the design of communication links(Special Interface circuits) called as Rambus Channel.
- Rambus memory has a clock frequency of 400MHZ.
- The data are transmitted on both the edges of the clock so that the effective data transfer rate is 800MHZ.
- The circuitry needed to interface to the Rambus channel is included on the chip.Such chips are known as Rambus DRAM‟ s(RDRAM).
- Rambus channel has,

    - 9 Data lines(1-8 Transfer the data,9th line Parity checking).
    - Control line
    - Power line

A two channel rambus has 18 data lines which has no separate address lines.It is also called as

Direct RDRAM's.

Communication between processor or some other device that can serves as a master and RDRAM modules are serves as slaves ,is carried out by means of packets transmitted on the data lines.

There are 3 types of packets.They are,

□ Request
□ Acknowledge
□ Data

3. Describe in detail about Read-Only Memories

READ ONLY MEMORY:

Both SRAM and DRAM chips are volatile,which means that they lose the stored information if power is turned off.

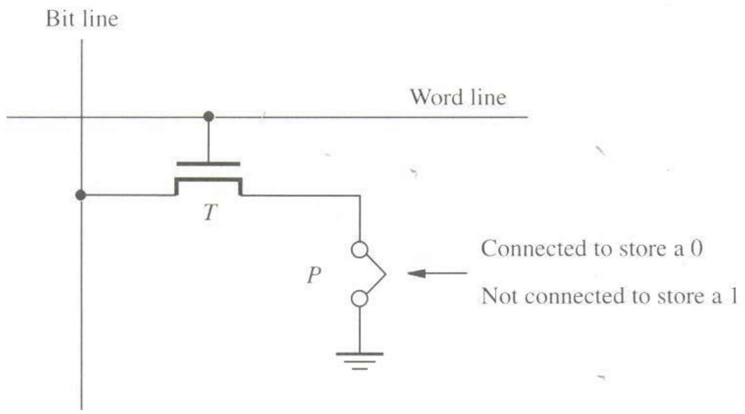
Many application requires Non-volatile memory (which retain the stored information if power is turned off).

Eg:Operating System software has to be loaded from disk to memory which requires the program that boots the Operating System ie. It requires non-volatile memory.

Non-volatile memory is used in embedded system.

Since the normal operation involves only reading of stored data ,a memory of this type is called ROM

Fig:ROM cell



At Logic value ‘0’ □ Transistor(T) is connected to the ground point(P).
Transistor switch is closed & voltage on bitline nearly drops to zero.
At Logic value ‘1’ □ Transistor switch is open.
The bitline remains at high voltage.
To read the state of the cell,the word line is activated.
A Sense circuit at the end of the bitline generates the proper output value.

Types of ROM:

Different types of non-volatile memory are,

- PROM
- EPROM
- EEPROM
- Flash Memory

PROM:-Programmable ROM:

- PROM allows the data to be loaded by the user.
- Programmability is achieved by inserting a „fuse‟ at point P in a ROM cell.
- Before it is programmed, the memory contains all 0‟ s
- The user can insert 1‟ s at the required location by burning out the fuse at these locations using high-current pulse.
- This process is irreversible.

Merit:

- It provides flexibility.
- It is faster.
- It is less expensive because they can be programmed directly by the user.

EPROM:-Erasable reprogrammable ROM:

- EPROM allows the stored data to be erased and new data to be loaded.
- In an EPROM cell, a connection to ground is always made at „P‟ and a special transistor is used, which has the ability to function either as a normal transistor or as a disabled transistor that is always turned „off‟ .

- This transistor can be programmed to behave as a permanently open switch, by injecting charge into it that becomes trapped inside.
- Erasure requires dissipating the charges trapped in the transistor of memory cells. This can be done by exposing the chip to ultra-violet light, so that EPROM chips are mounted in packages that have transparent windows.

Merits:

- It provides flexibility during the development phase of digital system.
- It is capable of retaining the stored information for a long time.

Demerits:

- The chip must be physically removed from the circuit for reprogramming and its entire contents are erased by UV light.

EEPROM:-Electrically Erasable ROM:

Merits:

- It can be both programmed and erased electrically.
- It allows the erasing of all cell contents selectively.

Demerits:

- It requires different voltage for erasing ,writing and reading the stored data.

Flash Memory:

- In EEPROM, it is possible to read & write the contents of a single cell.
- In Flash device, it is possible to read the contents of a single cell but it is only possible to write the entire contents of a block.
- Prior to writing,the previous contents of the block are erased.
- Eg.In MP3 player,the flash memory stores the data that represents sound.
- Single flash chips cannot provide sufficient storage capacity for embedded system application.
- There are 2 methods for implementing larger memory modules consisting of number of chips.They are,
    - Flash Cards
    - Flash Drives.

Merits:

- Flash drives have greater density which leads to higher capacity & low cost per bit.
- It requires single power supply voltage & consumes less power in their operation.

Flash Cards:

- One way of constructing larger module is to mount flash chips on a small card.
- Such flash card have standard interface.

- The card is simply plugged into a conveniently accessible slot.
- Its memory size are of 8,32,64MB.
- Eg:A minute of music can be stored in 1MB of memory. Hence 64MB flash cards can store an hour of music.

Flash Drives:

- Larger flash memory module can be developed by replacing the hard disk drive.
- The flash drives are designed to fully emulate the hard disk.
- The flash drives are solid state electronic devices that have no movable parts.

Merits:

- They have shorter seek and access time which results in faster response.
- They have low power consumption which makes them attractive for battery driven application.
- They are insensitive to vibration.

Demerit:

- The capacity of flash drive (<1GB) is less than hard disk(>1GB).
- It leads to higher cost perbit.
- Flash memory will deteriorate after it has been written a number of times(typically atleast 1 million times.)

4. What is cache memory? Describe in detail. (Apr 13)

CACHE MEMORIES

The effectiveness of cache mechanism is based on the property of „**Locality of reference'.**

Locality of Reference:

Many instructions in the localized areas of the program are executed repeatedly during some time period and remainder of the program is accessed relatively infrequently.

It manifests itself in 2 ways.They are,

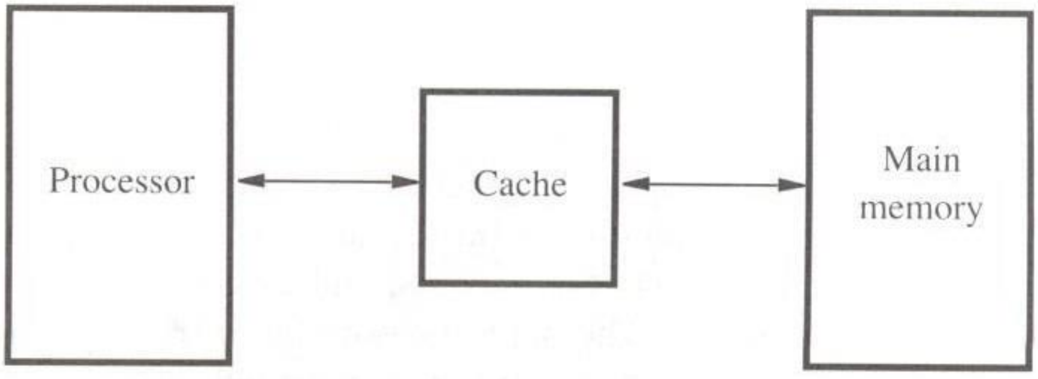
Temporal(The recently executed instruction are likely to be executed again very soon.)

Spatial(The instructions in close proximity to recently executed instruction are also likely to be executed soon.)

If the active segment of the program is placed in cache memory, then the total execution time can be reduced significantly. The term Block refers to the set of contiguous address locations of some size.

The cache line is used to refer to the cache block.

Fig:Use of Cache Memory



- The Cache memory stores a reasonable number of blocks at a given time but this number is small compared to the total number of blocks available in Main Memory.
- The correspondence between main memory block and the block in cache memory is specified by a mapping function.
- The Cache control hardware decide that which block should be removed to create space for the new block that contains the referenced word.
- The collection of rule for making this decision is called the replacement algorithm.
- The cache control circuit determines whether the requested word currently exists in the cache.
- If it exists, then Read/Write operation will take place on appropriate cache location. In this case Read/Write hit will occur.
- In a Read operation, the memory will not involve.

The write operation is proceed in 2 ways.They are,

- Write-through protocol
- Write-back protocol

Write-through protocol:
Here the cache location and the main memory locations are updated simultaneously.

Write-back protocol:

- This technique is to update only the cache location and to mark it as with associated flag bit called dirty/modified bit.
- The word in the main memory will be updated later,when the block containing this marked word is to be removed from the cache to make room for a new block.
- If the requested word currently not exists in the cache during read operation,then read miss will occur.
- To overcome the read miss Load –through / Early restart protocol is used.

Read Miss:
The block of words that contains the requested word is copied from the main memory into cache.

Load –through:

- After the entire block is loaded into cache,the particular word requested is forwarded to the processor.
- If the requested word not exists in the cache during write operation,then Write Miss will occur.
- If Write through protocol is used,the information is written directly into main memory.
- If Write back protocol is used then block containing the addressed word is first brought intothe cache and then the desired word in the cache is over-written with the new information.

5. Explain different types of mapping functions in cache memory (Apr 11)

Mapping Function:

Direct Mapping:

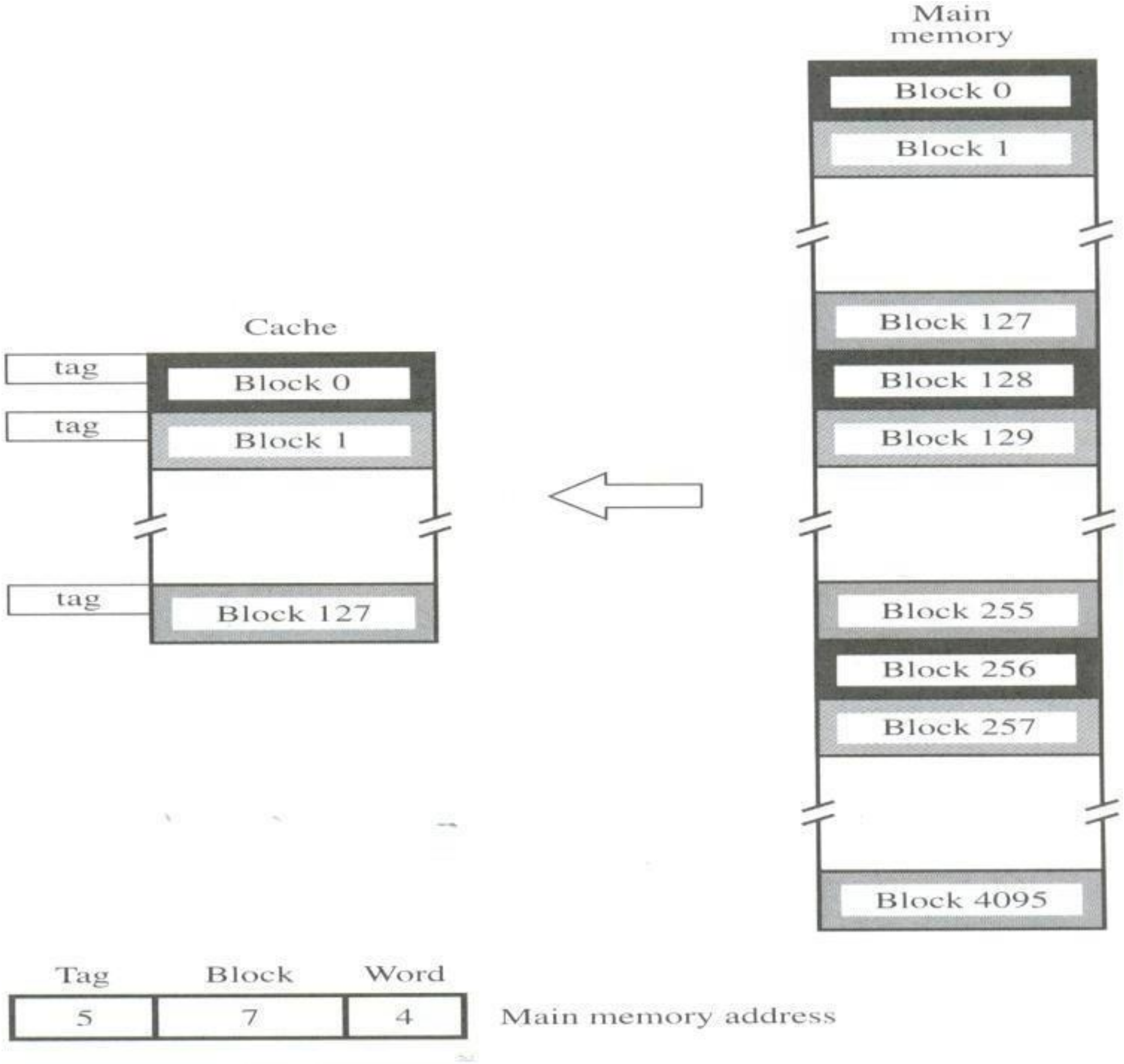
- It is the simplest technique in which block j of the main memory maps onto block „j‟ modulo 128 of the cache.

- Thus whenever one of the main memory blocks 0,128,256 is loaded in the cache,it is stored in block 0.
- Block 1,129,257 are stored in cache block 1 and so on.

The contention may arise when,

- When the cache is full
- When more than one memory block is mapped onto a given cache block position.
- The contention is resolved by allowing the new blocks to overwrite the currently resident block.
- Placement of block in the cache is determined from memory address.

Fig: Direct Mapped Cache



The memory address is divided into 3 fields.They are,

Low Order 4 bit field(word)□ Selects one of 16 words in a block.

7 bit cache block field□ When new block enters cache,7 bit determines the cache position in which this block must be stored.

5 bit Tag field□ The high order 5 bits of the memory address of the block is stored in 5 tag bits associated with its location in the cache.

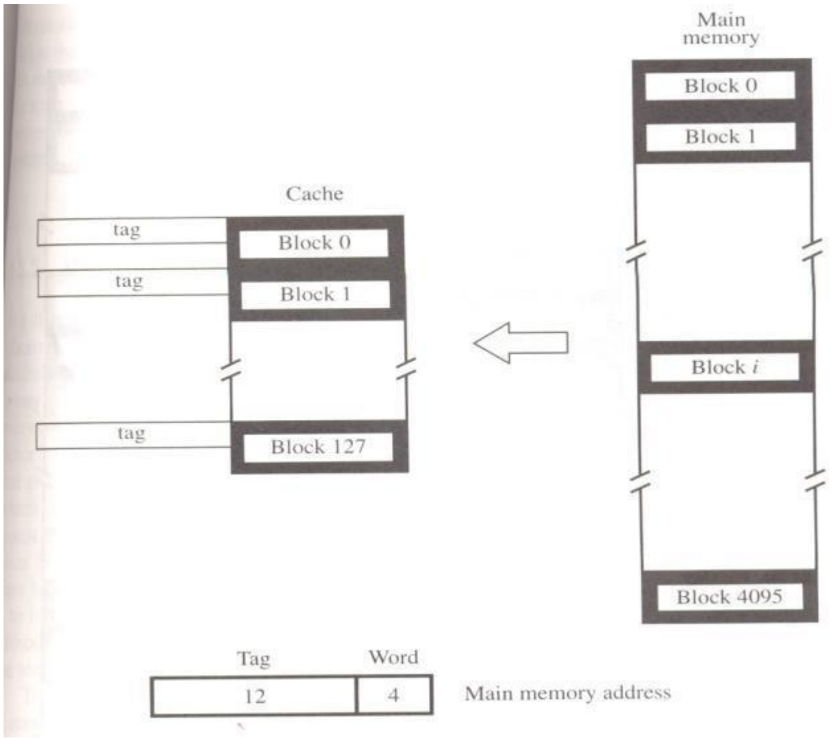
As execution proceeds, the high order 5 bits of the address is compared with tag bits associated with that cache location. If they match,then the desired word is in that block of the cache. If there is no match,then the block containing the required word must be first read from the main memory and loaded into the cache.

Merit:
It is easy to implement.

Demerit:
It is not very flexible.

Associative Mapping:
In this method, the main memory block can be placed into any cache block position.

Fig:Associative Mapped Cache.



- 12 tag bits will identify a memory block when it is resolved in the cache.
- The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is persent.This is called associative mapping.
- **I**t gives complete freedom in choosing the cache location.
- A new block that has to be brought into the cache has to replace(eject)an existing block if the cache is full.
- In this method,the memory has to determine whether a given block is in the cache.
- A search of this kind is called an associative Search.

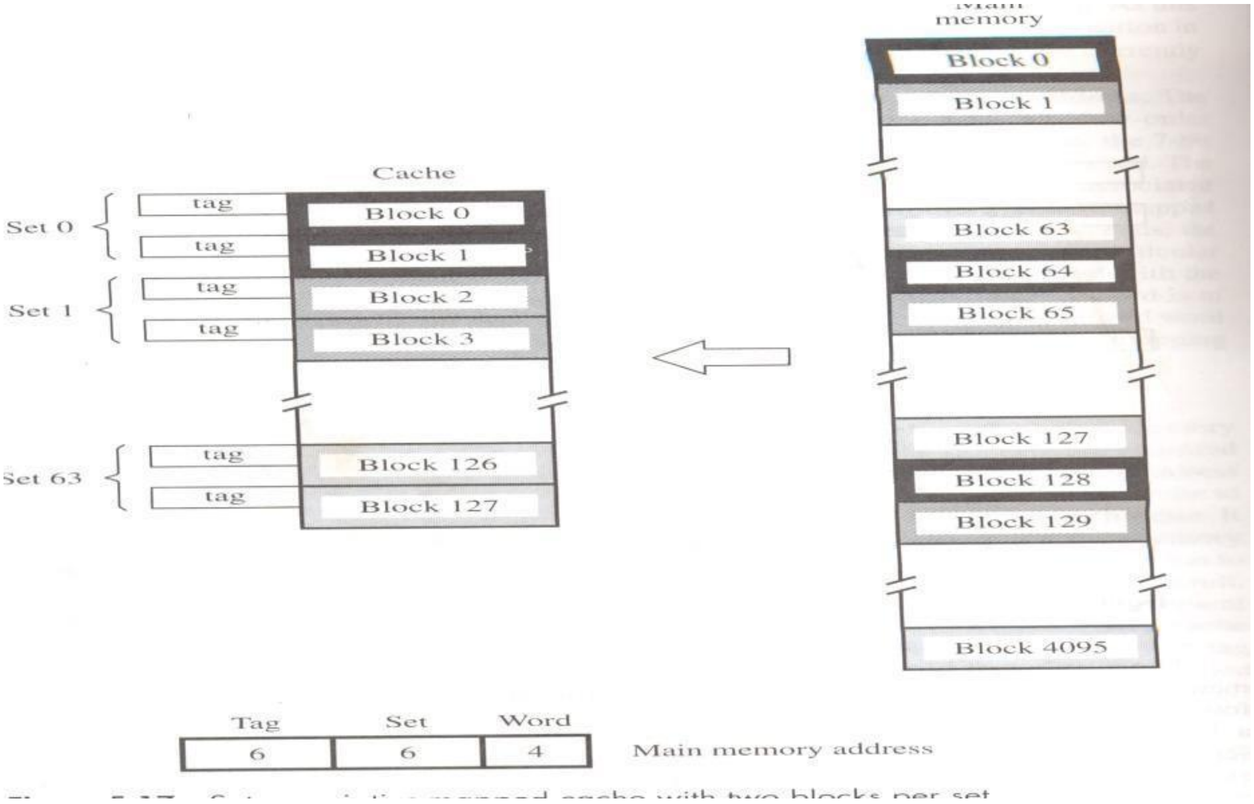
Merit:
It is more flexible than direct mapping technique.

Demerit:
Its cost is high.

Set-Associative Mapping:

- It is the combination of direct and associative mapping.
- The blocks of the cache are grouped into sets and the mapping allows a block of the main memory to reside in any block of the specified set.
- In this case,the cache has two blocks per set,so the memory blocks 0,64,128……..4032 maps into cache set „0‟ and they can occupy either of the two block position within the set.
- 6 bit set field□ Determines which set of cache contains the desired block .
- 6 bit tag field□ The tag field of the address is compared to thetags of the two blocks of
- the set to clock if the desired block is present.

Fig: Set-Associative Mapping:



No of blocks per set	**no of set field**
2	6
3	5
8	4
128	no set field

- The cache which contains 1 block per set is called direct Mapping.
- A cache that has „k‟ blocks per set is called as „k-way set associative cache‟ .
- Each block contains a control bit called a valid bit.
- The Valid bit indicates that whether the block contains valid data.
- The dirty bit indicates that whether the block has been modified during its cache residency.

- Valid bit=0□ When power is initially applied to system
- Valid bit =1□ When the block is loaded from main memory at first time.
- If the main memory block is updated by a source & if the block in the source is already exists in the cache,then the valid bit will be cleared to „0‟ .
- If Processor & DMA uses the same copies of data then it is called as the Cache Coherence Problem.

Merit:

The Contention problem of direct mapping is solved by having few choices for block placement.

The hardware cost is decreased by reducing the size of associative search.

Replacement Algorithm:

- In direct mapping, the position of each block is pre-determined and there is no need of replacement strategy.
- In associative & set associative method,the block position is not pre-determined;ie..when the cache is full and if new blocks are brought into the cache, then the cache controller must decide which of the old blocks has to be replaced.
- Therefore,when a block is to be over-written,it is sensible to over-write the one that has gone the longest time without being referenced.This block is called Least recently Used(LRU) block & the technique is called LRU algorithm.
- The cache controller track the references to all blocks with the help of block counter.

Eg:

Consider 4 blocks/set in set associative cache,
2 bit counter can be used for each block.

When a **‘hit’** occurs,then block counter=0;The counter with values originally lower than the referenced one are incremented by 1 & all others remain unchanged.

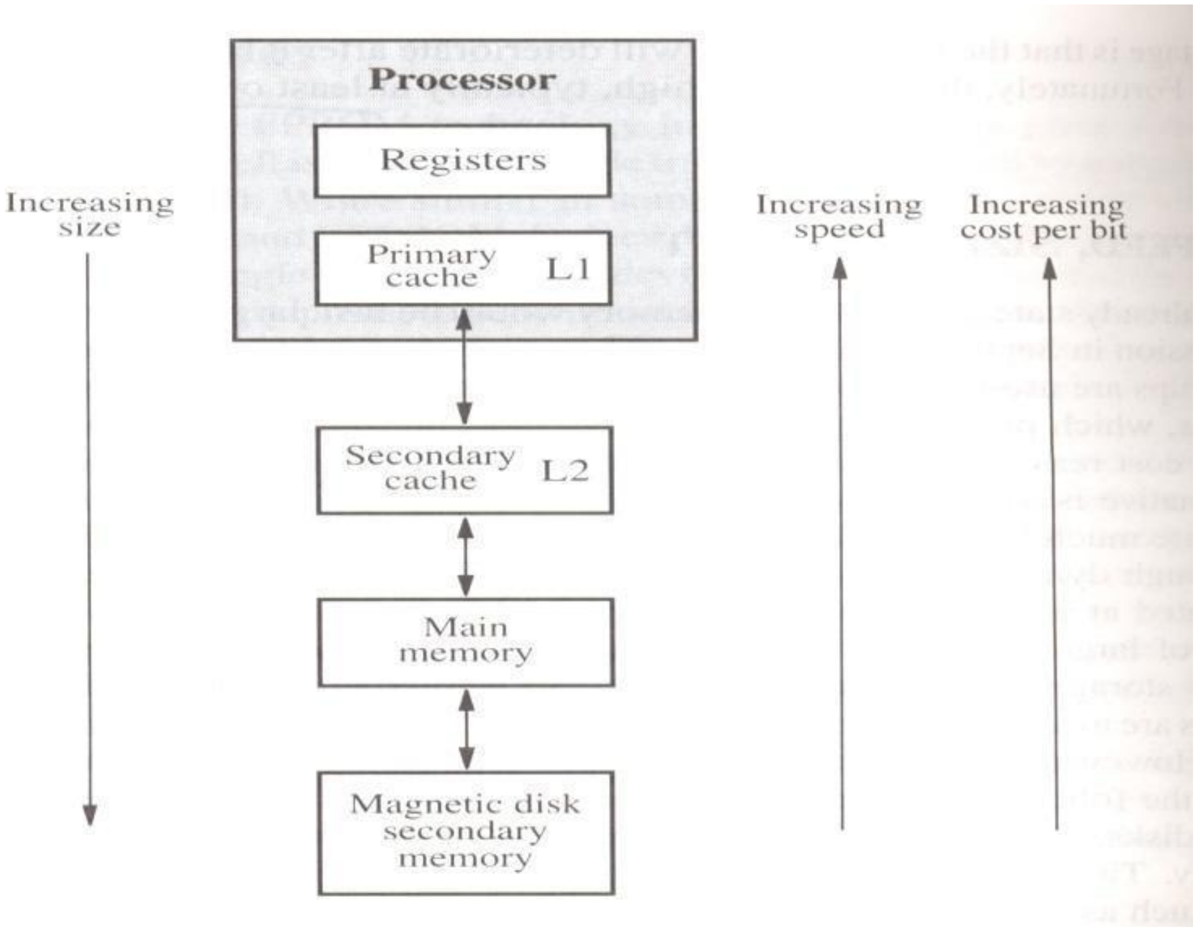
When a **‘miss’** occurs & if the set is full,the blocks with the counter value 3 is removed,the new block is put in its place & its counter is set to „0‟ and other block counters are incremented by 1.

Merit:

The performance of LRU algorithm is improved by randomness in deciding which block is to be over-written.

6. Write about memory hierarchy.

Fig:Memory Hierarchy



Characteristics	SRAM	DRAM	Magnetis Disk
Speed	Very Fast	Slower	Much slower than DRAM
Size	Large	Small	Small
Cost	Expensive	Less Expensive	Low price

Magnetic Disk:

A huge amount of cost effective storage can be provided by magnetic disk;The main memory can be built with DRAM which leaves SRAM‟ s to be used in smaller units where speed is of essence.

Memory	Speed	Size	Cost
Registers	Very high	Lower	Very Lower
Primary cache	High	Lower	Low
Secondary cache	Low	Low	Low
Main memory	Lower than Seconadry cache	High	High
Secondary Memory	Very low	Very High	Very High

Types of Cache Memory:

The Cache memory is of 2 types.They are,

Primary /Processor Cache(Level1 or L1 cache)

Secondary Cache(Level2 or L2 cache)

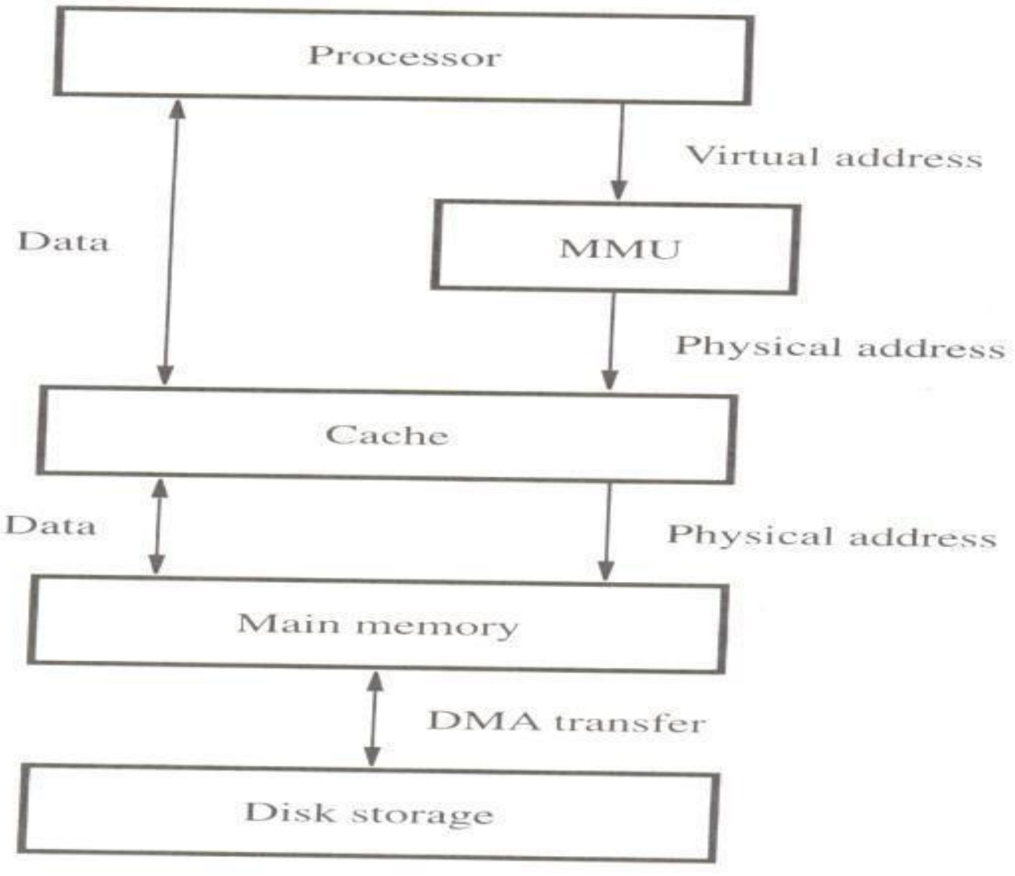
Primary Cache --- It is always located on the processor chip.
Secondary Cache ---It is placed between the primary cache and the rest of the memory.
The main memory is implemented using the dynamic components(SIMM,RIMM,DIMM).
The access time for main memory is about 10 times longer than the access time for L1 cache.

7. Explain in detail the concept of Virtual Memory (Apr 13)

VIRTUAL MEMORY:

- Techniques that automatically move program and data blocks into the physical main memory when they are required for execution is called the Virtual Memory**.**
- The binary address that the processor issues either for instruction or data are called the virtual / Logical address.
- The virtual address is translated into physical address by a combination of hardware and software components.This kind of address translation is done by MMU(Memory Management Unit).
- When the desired data are in the main memory ,these data are fetched /accessed immediately.
- If the data are not in the main memory,the MMU causes the Operating system to bring the data into memory from the disk.
- Transfer of data between disk and main memory is performed using DMA scheme.

Fig:Virtual Memory Organisation



Address Translation:

- In address translation,all programs and data are composed of fixed length units called Pages.
- The Page consists of a block of words that occupy contiguous locations in the main memory.
- The pages are commonly range from 2K to 16K bytes in length.
- The cache bridge speed up the gap between main memory and secondary storage and it is implemented in software techniques.
- Each virtual address generated by the processor contains virtual Page number(Low order bit) and offset(High order bit)
- Virtual Page number+ Offset Specifies the location of a particular byte (or word) within a page.

Page Table:
It contains the information about the main memory address where the page is stored & the current status of the page.

Page Frame:
An area in the main memory that holds one page is called the page frame.

Page Table Base Register:
It contains the starting address of the page table.

Virtual Page Number+Page Table Base register -- Gives the address of the corresponding entry in the page table.ie)it gives the starting address of the page if that page currently resides in memory.

Control Bits in Page Table:
The Control bits specifies the status of the page while it is in main memory.

Function:

- The control bit indicates the validity of the page ie)it checks whether the page is actually loaded in the main memory.
- It also indicates that whether the page has been modified during its residency in the memory;this information is needed to determine whether the page should be written back to the disk before it is removed from the main memory to make room for another page.

- The Page table information is used by MMU for every read & write access.
- The Page table is placed in the main memory but a copy of the small portion of the page table is located within MMU.
- This small portion or small cache is called Translation LookAside Buffer(TLB).
- This portion consists of the page table enteries that corresponds to the most recently accessed pages and also contains the virtual address of the entry.

Fig:Virtual Memory Address Translation

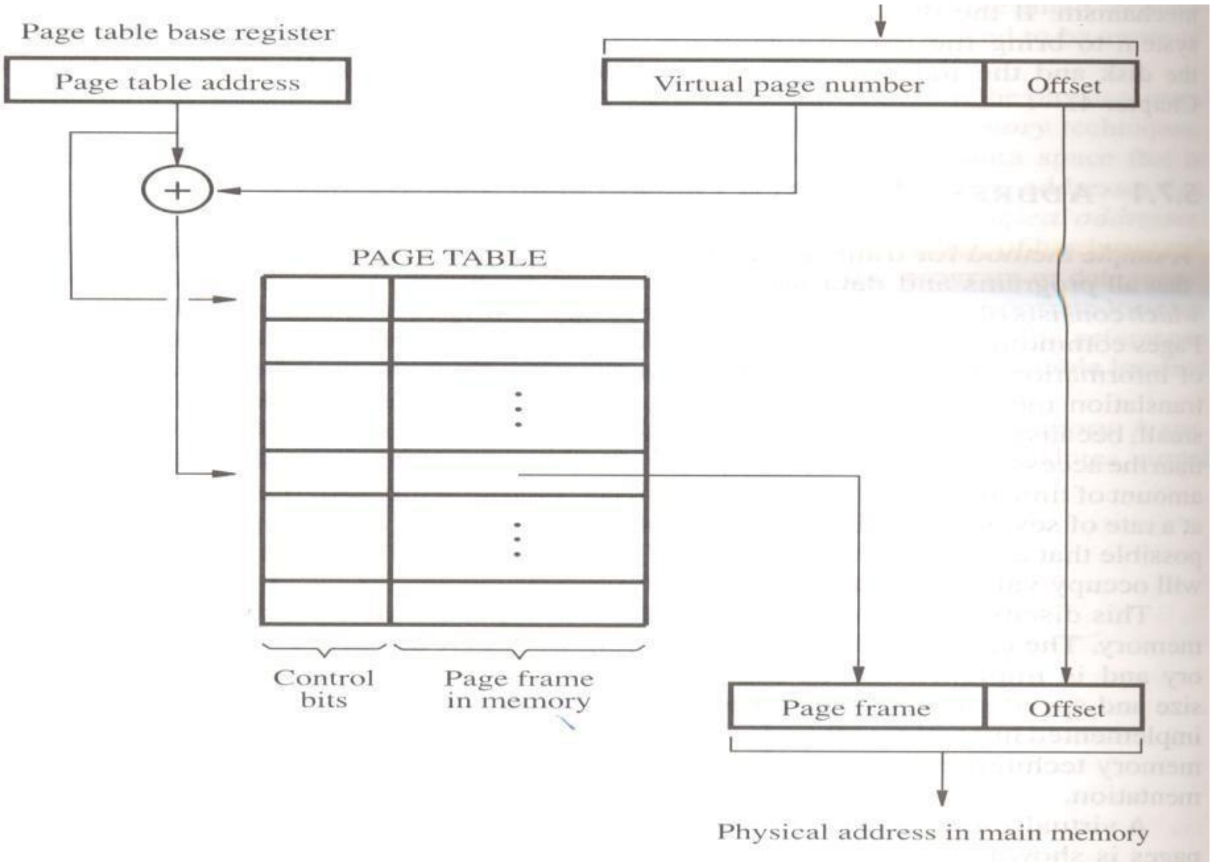
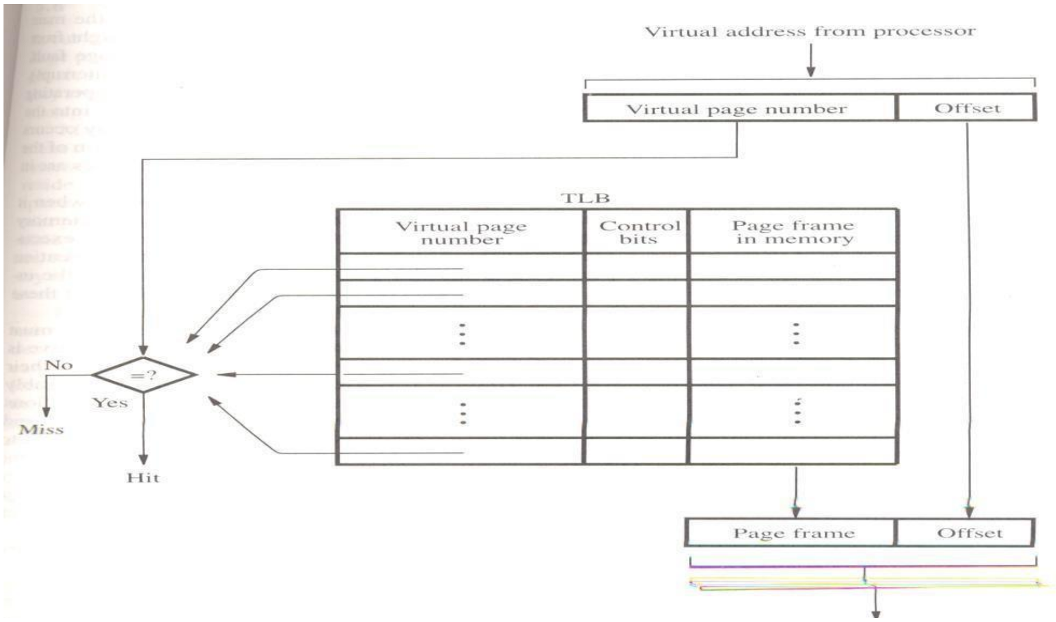


Fig:Use of Associative Mapped TLB



- When the operating system changes the contents of page table ,the control bit in TLB will invalidate the corresponding entry in the TLB.
- Given a virtual address,the MMU looks in TLB for the referenced page.
- If the page table entry for this page is found in TLB,the physical address is obtained immediately.
- If there is a miss in TLB,then the required entry is obtained from the page table in the main memory & TLB is updated.
- When a program generates an access request to a page that is not in the main memory ,then Page Fault will occur.
- The whole page must be broght from disk into memry before an access can proceed.
- When it detects a page fault,the MMU asks the operating system to generate an interrupt.
- The operating System suspend the execution of the task that caused the page fault and begin execution of another task whose pages are in main memory because the long delay occurs while page transfer takes place.
- When the task resumes,either the interrupted instruction must continue from the point of interruption or the instruction must be restarted.
- If a new page is brought from the disk when the main memory is full,it must replace one of the resident pages.In that case,it uses LRU algorithm which removes the least referenced Page.
- A modified page has to be written back to the disk before it is removed from the main memory. In that case,write –through protocol is used.

8. Discuss in detail about the secondary storage devices

SECONDARY STORAGE:

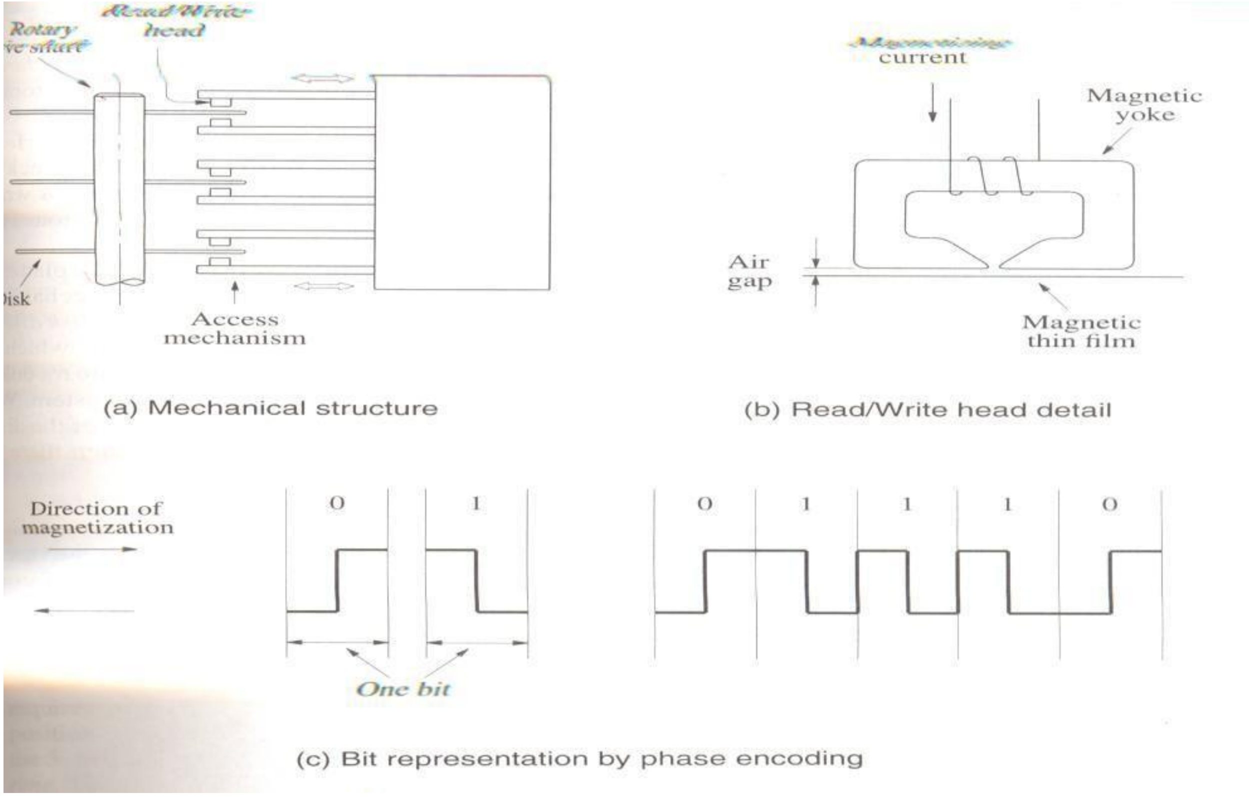
- The Semi-conductor memories donot provide all the storage capability.
- The Secondary storage devices provide larger storage requirements.
- Some of the Secondary Storage devices are,
  - Magnetic Disk
  - Optical Disk
  - Magnetic Tapes.

Magnetic Disk:

- Magnetic Disk system consists o one or more disk mounted on a common spindle.
- A thin magnetic film is deposited on each disk, usually on both sides.
- The disk are placed in a rotary drive so that the magnetized surfaces move in close proximity to read /write heads.
- Each head consists of magnetic yoke & magnetizing coil.
- Digital information can be stored on the magnetic film by applying the current pulse of suitable polarity to the magnetizing coil.
- Only changes in the magnetic field under the head can be sensed during the Read operation.

- Therefore if the binary states 0 & 1 are represented by two opposite states of magnetization, a voltage is induced in the head only at 0-1 and at 1-0 transition in the bit stream.
- A consecutive (long string) of 0‟ s & 1‟ s are determined by using the clock which is mainly used for synchronization.
- Phase Encoding or Manchester Encoding is the technique to combine the clocking information with data.
- The Manchester Encoding describes that how the self-clocking scheme is implemented.

Fig:Mechanical Structure



(a) Mechanical structure

(b) Read/Write head detail

(c) Bit representation by phase encoding

- The Read/Write heads must be maintained at a very small distance from the moving disk surfaces in order to achieve high bit densities.
- When the disk are moving at their steady state, the air pressure develops between the disk surfaces & the head & it forces the head away from the surface.
- The flexible spring connection between head and its arm mounting permits the head to fly at the desired distance away from the surface.

Wanchester Technology:

- Read/Write heads are placed in a sealed, air –filtered enclosure called the Wanchester Technology.

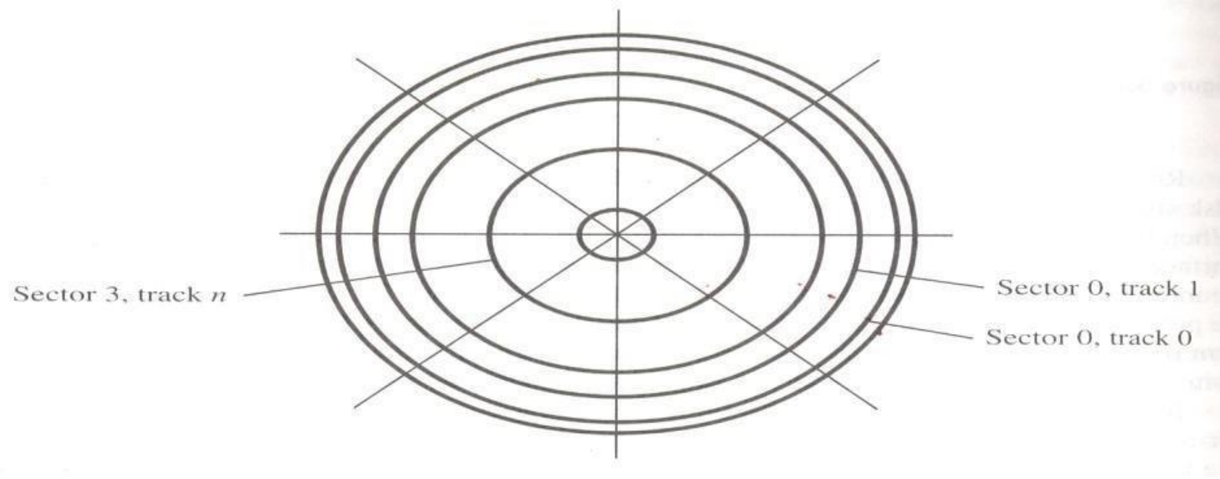
- In such units, the read/write heads can operate closure to magnetic track surfaces because the dust particles which are a problem in unsealed assemblies are absent.

Merits:

- It have a larger capacity for a given physical size.
- The data intensity is high because the storage medium is not exposed to contaminating elements.
- The read/write heads of a disk system are movable.
- The disk system has 3 parts.They are,

Disk Platter(Usually called Disk)
Disk Drive(spins the disk & moves Read/write heads)
Disk Controller(controls the operation of the system.)

Fig:Organizing & Accessing the data on disk



- Each surface is divided into concentric tracks.
- Each track is divided into sectors.
- The set of corresponding tracks on all surfaces of a stack of disk form a logical cylinder.
- The data are accessed by specifying the surface number,track number and the sector number.
- The Read/Write operation start at sector boundaries.
- Data bits are stored serially on each track.
- Each sector usually contains 512 bytes.
- Sector header -> contains identification information.
- It helps to find the desired sector on the selected track.
- ECC (Error checking code)- used to detect and correct errors.
- An unformatted disk has no information on its tracks.
- The formatting process divides the disk physically into tracks and sectors and this process may discover some defective sectors on all tracks.
- The disk controller keeps a record of such defects.
- The disk is divided into logical partitions. They are,
    - ✓ Primary partition
    - ✓ Secondary partition
- In the diag, Each track has same number of sectors.
- So all tracks have same storage capacity.
- Thus the stored information is packed more densely on inner track than on outer track.

Access time

There are 2 components involved in the time delay between receiving an address and the beginning of the actual data transfer. They are,

- ✓ Seek time
- ✓ Rotational delay / Latency

Seek time – Time required to move the read/write head to the proper track.

Latency – The amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector passes under the read/write head.

Seek time + Latency = Disk access time

Typical disk

One inch disk- weight=1 ounce, size -> comparable to match book

Capacity -> 1GB

Inch disk has the following parameter

Recording surface=20

Tracks=15000 tracks/surface

Sectors=400.

Each sector stores 512 bytes of data

Capacity of formatted disk=20x15000x400x512=60x109 =60GB

Seek time=3ms

Platter rotation=10000 rev/min

Latency=3ms

Internet transfer rate=34MB/s

Data Buffer / cache

- A disk drive that incorporates the required SCSI circuit is referred as SCSI drive.
- The SCSI can transfer data at higher rate than the disk tracks.
- An efficient method to deal with the possible difference in transfer rate between disk and SCSI bus is accomplished by including a data buffer.
- This buffer is a semiconductor memory.
- The data buffer can also provide cache mechanism for the disk (ie) when a read request arrives at the disk, then controller first check if the data is available in the cache(buffer).
- If the data is available in the cache, it can be accessed and placed on SCSI bus . If it is not available then the data will be retrieved from the disk.

Disk Controller

The disk controller acts as interface between disk drive and system bus.

The disk controller uses DMA scheme to transfer data between disk and main memory.

When the OS initiates the transfer by issuing Read/Write request, the controllers register will load the following information. They are,

Main memory address(address of first main memory location of the block of words involved in the transfer)

Disk address(The location of the sector containing the beginning of the desired block of words)

9. Discuss the following

(i) Interleaving

(ii) Hit rate and Miss penalty

(iii) Pre-fetching

IMPROVING CACHE PERFORMANCE:

- Two key factors in the commercial success of a computer are performance and cost. The objective is the best possible Performance at the lowest cost.
- The challenge in design alternative is to improve the performance without increasing the cost. A common measure of success is the price/performance ratio.
- The memory hierarchy shows the best price/performance ratio. Each level of hierarchy plays an important role. The speed and efficiency of data transfer between various level of hierarchy are having great significance. Both is not possible if, both the slow and the fast units are accessed in the same manner, but can be achieved by the parallelism in the organization of the slower unit. An effective way to introduce parallelism is to use an "interleaved organization"

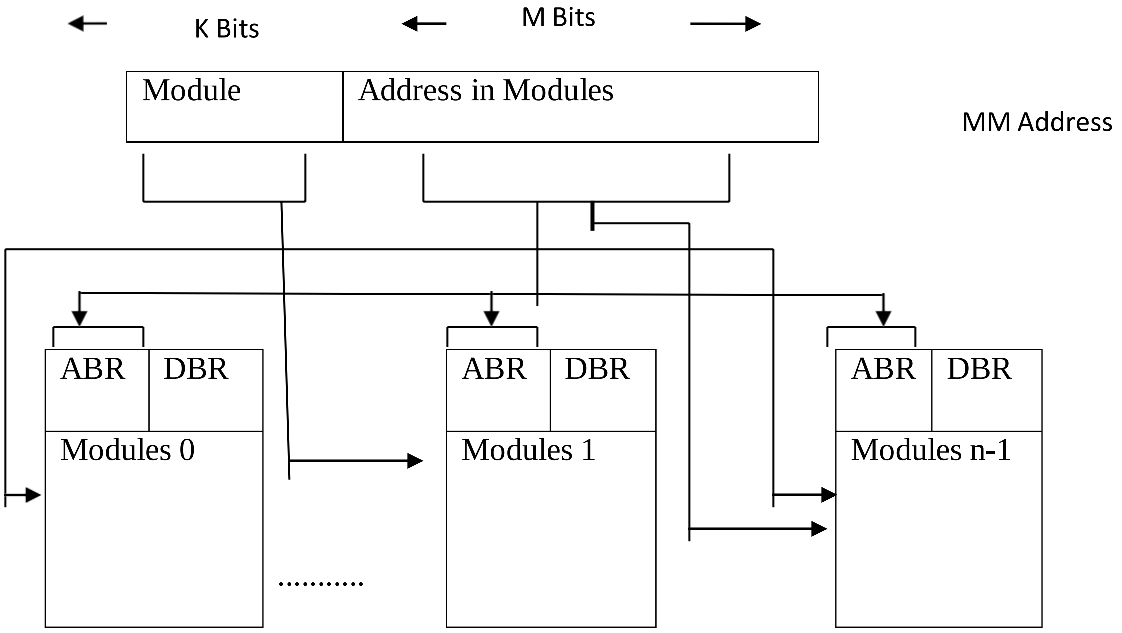
INTERLEAVING

- If the main memory of computer is structured as a collection of physical separate modules , each with its own address buffer register(ABR) and data buffer register(DBR), memory address operations may proceed in more than one module at the same time.
- How individual address are distributed over the modules is critical. Two methods of address layout are there.
- In the first case, the memory address generated by the processor is decoded as shown in fig. the higher-order k bits name one of n modules, and the lower-order m bits name a particular word in that module.
- When consecutive locations are accessed , when a block of data is transferred to a cache, only one module is involved. At the same time, devices with direct memory access(DMA) ability may be accessing information in other memory modules.

The second way to address the modules is shown in fig. It is called memory interleaving. The low-order k bits of the memory address select a module, and the higher-order m bits name a location within that module. In this way , consecutive addresses are located in successive modules. This results in faster access to a block of data and higher average utilization of the memory system as a whole.

Interleaving is used to within SDRAM chips to improve the speed of accessing successive word of data.

Fig(a). Consecutive words in a modules



Fig(b). Consecutive words in Consecutive a modules

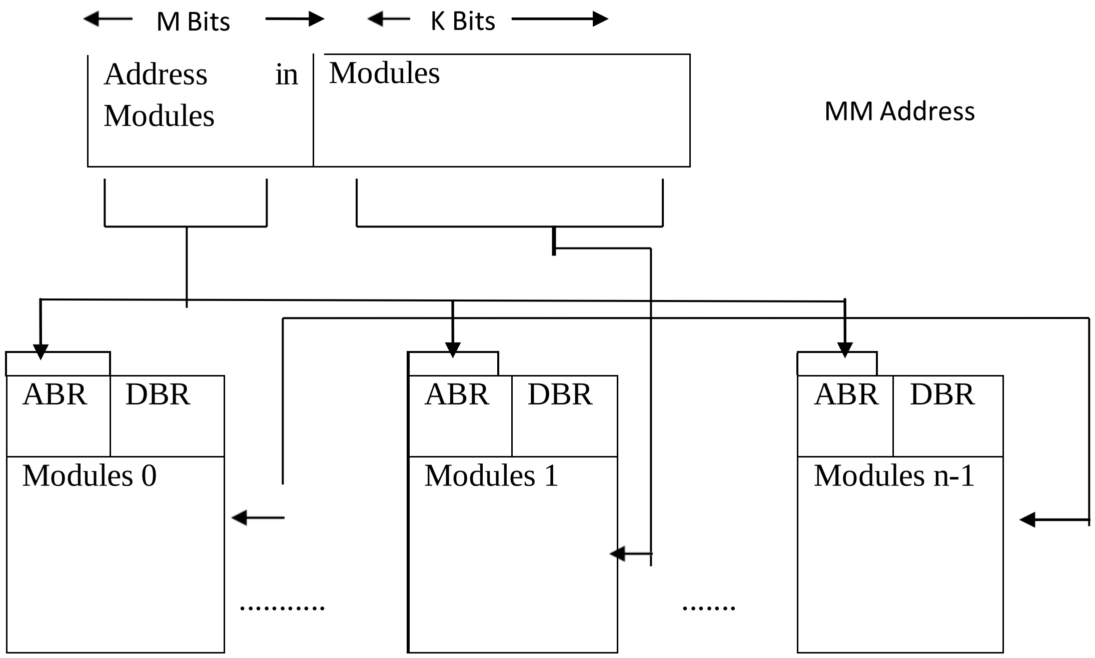


Fig : Addressing multiple-module memory sytems

HIT RATE AND MISS PENALTY

- The effective implementation of the memory hierarchy is the success rate in accessing information at various levels of hierarchy.

- The successful access to data in cache is called a hit. The number of hits are stated as a fraction of all attempted accesses is call the hit rate, and the misses rate is the number of misses stated as a fraction of attempted accesses.

- High hit rates are essential for high-performance computers , well over 0.9.

- Performance is affected by the miss. The extra time needed to bring the desired information into the catch is called the miss penalty. This penalty make the processor to stall.

- In general, the miss penalty is the time needed to bring a block of data from a slower unit to a faster unit. The miss penalty is reduced, if the efficient mechanisms are implemented.

- The performance of a computer is affected positively by increased hit rate and negatively by increased miss penalty, the block sizes that are neither very small nor large give the best results.

- In practice, block sizes in the range of 16 to 128 bytes have been the most popular choices.

PREFETCHING

- The new data are brought into the cache when they are first needed. A read miss occurs , and the desired data are loaded from the main memory. The processor has to pause until the new data arrive.

- A special prefetch instruction may be provided in the instruction set of the processor. Executing this instruction causes the addressed data to the loaded into the cache, in the case of read miss.

- Prefetch instructions can be inserted into a program either programmer or by the compiler.

- However the overall effect of the software prefetching on performance is positive and it is supported by machine instructions of many processors.

- Prefetching can also done through hardware . this involves adding circuitry thet attempts to discover a pattern in memory references and then prefetches data according to that.

- Example:

- Intel’s Pentium 4 processor has facilitates for prefetching information into caches using both software and hardware approaches. There are special prefetch instructions that can be included in programs to bring a block of data into the desired level of cache.

CACHES ON THE PROCESSOR CHIP

- The space on the processor chip is needed for many other functions, this limits the size of the cache that can be accommodated.

- All high-performance processor chips include some form of cache. Some manufacturers have chosen to implement two separate caches, one for instructions and another for data.

- Example: 68404, Pentium 3 and Pentium 4 processors.

- Example: ARM710T Processor.

- A combined cache provides better bit rate and it offers greater flexibility in mapping new information into the cache.

DISADVANTAGES:

- Increase parallelism comes at the expense of more complex circuitry.
- In high-performance processors two levels oh caches are normally used. The L1 caches(s) is on the processor chip. The L2 cache, which is much larger, may be implemented externally.
- If both L1 and L2 caches are used, the L1 cache should be designed to do fast access. A practical way to speed up access to the cache is to access more than one word simultaneously and let the processor use them one at a time.
- The average access time experienced by the processor in a system with two levels of caches is
- $T_{ave}=h_1C_1+(1-h_1)h_2C_2+(1-h_1)(1-h_2)M$

Where h_1 is the hit rate in the L_1 cache

h_2 is the hit rate in the L_2 cache

C_1 is the time to access information in the L_1 cache

C_2 is the time to access information in the L_2 cache

M is the time to access information in the main memory

Other Enhancements:

Several other possibilities exist for enhancing performance.

Write buffer

- When a write-through protocol is user, each write operation results in writing new value into the memory.
- If the processor must wait for the memory function to be completed , the it slowed down by all write requests.
- It is not necessary for the processor to wait for the write request to be completed.
- To improve performance , a write buffer can be included for temporary storage of write requests.
- The processor [places each write request into this buffer and continues execution if the next instruction.
- The write requests stored in the write buffer are sent to the main memory when it is not having any reading operation.
- The write buffer may hold a number of write requests.

LOOKUP-FREE CACHE

- A cache that can support multiple outstanding misses is called lookup-free. Since it can service only one miss at a time, it must include circuitry that keeps track of all outstanding misses. They may be done with special registers that hold the pertinent information about these misses.

10. Explain about memory management requirements

1. Virtual memory gives us an illusion that deals with only one large program. But that is not the case. If all of the programs do not fit into the available memory space, parts of it are swapped to and from the disk into the main memory. These are done with the help of management routine that resides as a part of the operating system.
2. There are two spaces available in the memory. They are namely
    - System space
    - User space
3. The operating system routines reside in system space.
4. The user application programs reside in user space.
5. There may be a number of user spaces, one for each user.These are arrnaged by providing a separate page table for each user program. The MMU uses a page table base register to determine the address of the table to be used in the translation process
6. By changing the contents of the page table base register, the oprating system can switch from one space to another. Thus the physical main memory is shared by the active pages of the system space and several user spaces.
7. But these two spaces must not be overlapped as its leads to serious threats. Hence a notion of promotion must be addressed. This kind of protection can be provided in several ways. The basic form of protection utilizes two kinds of states namely
    - Supervisor state
    - User state
8. The processor is always placed in the supervisor state while executing the oprating system routines and placed in the user state while executing the user programs
9. There are some machine instructions known as privileged instructions cannot be executed in the user state. These can be executed only in the supervisor state thereby a user program is prevented from accessing the page tables of other user spaces of the system space.

10. It is sometimes needed for one application program to have access to certain pages belonging to another program. The operating system can arrange this by having these pages to appear in both spaces. The shared pages will therefore have entries in two different page tables.
11. The control bits in each table entry can be set to control the access privileges granted to each program. For Example, one program may be allowed to read and write a give page, while the other program may be given only read access.

UNIT – V

BASIC PROCESSING UNIT: Some Fundamental Concepts, Execution of a Complete Instruction, Multiple-Bus Organization, Hardwired Control, Micro programmed Control,

PIPELINING: Basic Concepts, Data Hazards, Instruction Hazards, Influence on Instructions Sets, Data path and Control Considerations, Superscalar Operations, Performance Considerations

2 MARKS

1. What do you mean by micro-operation?

To perform fetch, decode and execute cycles the processor unit has to perform set of operations called micro-operation.

2. Define Processor.

It executes machine instructions and coordinates the activities of other units. It is also called as instruction set processor or central processing unit (CPU).

3. What is Data path?

The data registers, ALU and the interconnecting bus are referred to as data path.

4. What is meant by program counter?

It is a processor register mainly used for execution. It stores the address of the next instruction to be executed. After fetching an instruction the content of the PC are updated to point to the next instruction in the sequence.

5. Define IR?

IR is an instruction register. To execute an instruction the processor fetches the contents of the memory location pointed by the PC. The contents of this location are interpreted as an instruction to be executed. They are loaded into the IR.

6. What is micro program? (Apr 13)

A sequence of one or more micro operations designed to control specific operation, such as addition, multiplication is called a micro program.

7. What do you mean by hardwired control unit?

In the hardwired control, the control units use fixed logic circuits to interpret instructions and generate control signals from them.

8. Define microinstruction?

It is to assign one bit position to each control signal required in the CPU. However, This scheme has one serious drawback –assigning individual bits to each control signal results in long micro instructions, because the number of required signal is usually large. Moreover, only few bits are used in any given instruction .The solution of this problem is to group the control signals.

9. List the two techniques used for grouping of control signals

1. Control signals: IN and OUT signals
2. Gating signals: Read, write, clear A, set carry in, continue operation etc.

10. Write down the steps to execute an instruction.

- Fetch the contents of the memory location pointed by the PC and store that content into instruction registers. IR← ╔ PC ╗
- Increment the PC value by 4 to point out the next instruction in the program. PC←[PC]+4
- Carry out the actions specified by the instruction in the IR.

11. Define fetch step.

To perform the execution of the instruction we have to fetch the content from the memory and store that content into the processor register IR. This is known as fetching phase.

12. What is meant by execution phase?

Carry out the actions specified by the instruction in the instruction in the instruction register is known as execution instruction

13. Define MAR, MDR?

MAR means memory address register and MDR means memory data registers. These two are the processor registers that can be used in memory read and write operations.

14. Define register transfer and list out the signals used to do it.

As instruction execution involves a sequence of steps in which data are transferred from one register to another register. Two control signals are used to place the contents of the registers on the bus or to load the data on the bus into the registers. The signals are Ri in, Ri out.

15. Write down the control sequence for Move (R1), R2.

The control sequence is:

R1 out, MAR in Read

MDRoutE, WMFC

MDRout, R2 in,

16. Write down the steps to transfer the content of register R1 to register R4.

- Enable the output of register R1 by setting R1 out to 1. This places the contents of R1 on the processor bus.

- Enable the input of register R4 by setting R4 into 1. This loads data from processor bus into register R4.

17. Define multiphase clocking.

In some processor data transfers may use both the rising and falling edges of the clock. Two or more clock signals are needed to guarantee proper transfer of data. This is known as multiphase clocking.

18. Define MFC signal.

To accommodate the validity in response time, the processor waits until it receives an indication that the requested Read operation has been completed. A control signal MFC (Memory Function Complete) is used for this purpose.

19. Write down the steps to execute Add (R3), R1 instruction.

Fetch the instruction

Fetch the first operand

Perform the addition

Load the result into R1.

20. Define register file.

In multi bus architecture all the general purpose registers are called combined into a single clock called as register file.

21. Define interrupt?

CPU supervises the other system components via special control lines. Whenever the CPU receives the signals from the IO device (i.e.) interrupt signals, it suspends the current execution of the program and performs the interrupt request. After process the interrupt request, CPU transfers from supervisor mode to user mode.

22. Define instruction cycle. (Nov 12)

The sequence of operations involved in processing an instruction is called as an instruction cycle. It is divided into two phases: 1.fetch cycle 2. Execution cycle. The instruction is obtained from main memory during the fetch cycle. The execution cycle includes decoding the instruction, fetching any required operands, and performing the operation specified by the instructions opcode.

23. Define Hardwired control?

The circuit is design with the useful goals of minimizing the number of components used and maximizing the speed of operation. Once the unit is constructed, the only way implement changes in control unit behaviors are by redesigning the entire unit. Such a circuit is called hardwired control design.

24. What is the difference between hardwired control and micro' programmed control memory?

Hardwired Control: Implementation of hardwired is using sequential circuits and flip flops. If any change is to be done then the whole design is to be modified.

Micro Program Control:_Micro program is based on microinstruction. If the change is *to* be design then part of program is *to* be modified.

25. What is the difference between horizontal microinstructions and vertical microinstructions

Horizontal Micro Instruction: Ability to express a high degree of parallelism. The length of format is long. Little encoding of control information.

Vertical Micro Instruction: The length of the format is short Limited ability *to* express parallel micro operations. Considerable encoding of control information

26. Define multi-cycle?

ALU processes each m bit slice in K consecutive clock cycles is termed as multi-cycle.

27. Explain load-store architecture?

The program fragment that uses only the "load and store instruction to access memory is called load and store architecture. It is common to allow other instruction to specify operands in memory.

28. How do you measure the speed of a pipeline?

Pipeline speedup s(m) = T(1)1T(m) ,

M-Stage ...

T(m)- The execution time for same target workload on an m-stage pipeline

T(1) - The execute on time for same target workload on a non pipelined processor

29. How do you calculate the performance of the pipeline?

Pipeline's performance *I* cost ratio PCR = *f/k*

f - Clock frequency k - Hardware cost

30. Define Hit ratio.

The performance of cache memory is frequently measured in terms of a quantity called hit ratio. Let N1 and N2 denote the number of references to M1 and M2respectively in the block address stream.The block hit ratio H is defined by H=N1/N1+N2

31. What is the difference between macro and microinstructions?

Macro Instruction: Assign symbolic name to sequence of instructions I

Micro Instruction: Specify low-level micro operations.

32. Explain coprocessor function?

Coprocessor is a separate instruction set processor (ie) closely coupled to the CPU and whose instruction and registers direct extensions of the CPU'S.

33. What is control word?

It is a word whose individual bits represent the various control signals. Control sequence of an instruction defines a unique combination of 1’s and 0’s in the control word.

A sequence of CW’s corresponding to the control sequence of a machine Instruction constitutes the micro routine for that instruction.

34. Define control store.

The micro routines for all instructions in the instructions set of a computer are stored in a special memory called the control store. To read the control words sequentially from the control store, a micro program counter is used.

35. List out the situations that not increment the micro Pc value.

- When a new instruction is loaded into the IR, the micro PC is loaded with the starting address of the micro routine for that instruction.
- When a branch instruction is encountered and the branch condition is satisfied the micro Pc is loaded with the micro Pc is loaded with the branch target address.
- When an End instruction is encountered micro Pc is loaded with the address of the first CW in the micro routine for the instruction fetch cycle.

36. What is the drawback present in micro instruction s representation and how can we eliminate it?

Assigning individual bits to each control signal results in long micro instruction s because the number of required signals is usually large. Moreover only a few bits are set 1. So the available bit space is poorly used. We can overcome this draw back by grouping the relevant control signals.

37. Define vertical organization.

Highly encoded scheme groups more number of instruction s into a single group. So minimum number of groups is enough to represent instruction set. This is known as vertical organization.

38. What is meant by horizontal organization?

Minimally encoded scheme groups minimum number of instruction s into single group. So we need more group to represent the instruction set. This is known as vertical organization.

39. Define bit OR ing technique.

By using this technique we can modify the branch address. It use an Or gate to change the least significant bit of the specified instruction’s address to1, if the addressing mode is used.

40. Why it is need of pre fetch instruction?

One drawback of micro programmed control is the slower operation because of the time it takes to fetch instruction s from the control store. Faster operation is achieved if the next instruction is pre fetched while three current one is being executed.

41. Define emulation.

Programs written in the machine language of M2 can be run on computer M1 that is M1 emulate M2. Emulation allows us to replace absolute equipments.

42. What is meant by micro programmed control?

In some processor the control signals are generated by a program similar to machine language programs. This is known as micro programmed control.

43. Comparison between Hardwired and Micro programmed control control

Attribute	Hardwired control	Micro programmed control
Speed	Fast	Slow
Ability to handle large	Somewhat difficult	Easier
Design process	Somewhat complicated	Orderly and systematic
Applications	Mostly RISC microprocessors	Mainframe ,some microprocessors

44. Write the register transfer sequence for storing a word in memory.

Writing word into a memory location, the derived address is loaded into MAR. Then the data can be written are loaded into MDR and a write command is issued.

Hence executing the instruction MOV R2, (R1) requires the **following sequence**

R1out, MAR in

R2out MDR in, write

MDR out E, WMFC.

The processor remains in step3 until the memory operation is completed and as MFC response is received.

45. What is a micro program sequencer?

If each machine instruction is implemented by a microinstruction using a bit for each control word, a micro-program counter is sufficient to control sequencing.

Advantage:

Writing micro program is fairly simple because standard software techniques can be used.

Disadvantage:

Two major disadvantages exist.

Large number of microinstructions and large control store

Execution time is larger.

Consider a more complicated example of a complex machine instruction

ADD src,Rdst which adds the source operand to the contents of the destination register and result will be stored in the destination register.

Assume that source operand can be specified in the following addressing modes

register 2) autoincrement 3)autodecrement 4)indexed as well as the indirect forms of these four modes

46. What are the sequences of operations involved in processing an instruction constitutes an instruction cycle?

The sequence of operations involved in processing an instruction constitutes an instruction cycle, which can be subdivided into 3 major phases:

1. Fetch cycle

2. Decode cycle

3. Execute cycle

47. What are advantage and disadvantage of hardwired control and Micro programmed control?

Advantages of Micro programmed control

It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.

Control functions are implemented in software rather than hardware.

Disadvantages

A micro programmed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.

48. What is the address sequencing capabilities required in control memory?

Each microinstruction should explicitly or implicitly specify the next micro instruction to be used; such address sequencing capabilities are required in the control memory

49. In what ways Width and Height of the control memory can be reduced?

To reduce the number of pins, the dynamic memory chips use multiplexed address inputs. The address is divided into two parts.

High-order address bits-select a row in the cell array

Low-order address bits-select a column in the cell array

A typical processor issues all bits of an address at the same time

50. List the advantages of Multi-bus organization.

Compared to single-bus architecture, the using of multiple-bus architecture have a great advantage in speed and of course, will affect performance also. Instead of using single-bus architecture,

It is more convenient to use multiple-bus architecture. Using multiple-bus architecture will make each device to connect to own bus, which means that each device will have its own bus.

51. What are the inputs for Hardwired control?

Step	Action	Comments
1	PCout,MARin,read,select 4,add,Zin	Load pc in MAR Issue read request to memory. select 4 to Y.do the add operation result stored in Z [PC<-PC+4->word size)]
2	Zout,Pcin,Vin,WMFC	Load pc with next address(INR).wait until memory responds
3	MDRout,IRin	Load 1nstr from MDR to IR
4	R3out,MARin,read	Read the first operand pointed to by R2(from memory)
5	R1out,Yin,WMFC	Enable Riout to transfer the second operand to yin
6	MDRout,select Y, Add, in	Add the operand Y&R1and store the result in z.
7	Zout,R1in,End	Transfer the result back to resulted R

52. Under what situations the micro program counter is not incremented after new instruction is fetched from micro program memory?

There is a situation arises when the control unit is required to check the status of the condition codes or external inputs to choose between alternative course of action.

53. What are the relative merits of horizontal and vertical micro instruction format?

Table shows the comparison between horizontal and vertical organization.

S.No	Horizontal	Vertical
1.	Long formats	Short formats
2.	Ability to express a high degree of parallelism	Limited ability to express parallel micro operations
3.	Little encoding of the control information	Considerable encoding of the control information
4.	Useful when higher operating speed is desired	Slower operating speeds

54. Define Pipelining.

Pipelining increases instruction throughput by performing multiple operations at the same time (in parallel), but does not reduce instruction latency (the time to complete a single instruction from start to finish) as it still must go through all steps.

55. Explain the role of cache memory in Pipelining?

- Each pipeline stage is expected to complete in one clock cycle.
- The clock period should be long enough to let the slowest pipeline stage to complete.
- Faster stages can only wait for the slowest one to complete.
- Since main memory is very slow compared to the execution, if each instruction needs to be fetched from main memory, pipeline is almost useless.
- Fortunately, we have cache.

56. What is hazard?

Any condition that causes the pipeline to stall is called a hazard.

57. What is data hazard?

A data hazard is any condition in which either the source or destination operands of an instruction are not available at the time expected in the pipeline.

58. What is instruction hazard?

The pipeline may be stalled because of a delay in the availability of an instruction which results in cache miss. These hazards are called control hazards or instruction hazards.

59. What is structural hazard?

Structural hazard arises in a situation when two instructions require use of a given hardware resource at the same time.

60. What is branch penalty?

The time lost as result of a branch instruction is often referred to as branch penalty. The branch penalty is one clock cycle. For a longer pipeline, the branch penalty may be higher.

61. What is meant by dispatch unit?

Dispatch unit is a separate unit, which takes instructions from the front of the queue and sends them to the execution unit. The dispatch unit also performs the decoding function.

62. Define the terms branch delay slot and delayed branching.

The location following the branch instruction is called a branch delay slot. There maybe more than one branch delay slot, depending on the time it takes to execute a branch instruction.

A technique called delayed branching can minimize the penalty incurred as a result of conditional branch instructions. The instructions in delay slots are always fetched. If there are no useful instructions, these are filled with NOP instructions. That is branching takes place one instruction later than where the branch instruction appears in the instruction sequence in the memory, hence “delayed branch”.

63. What is speculative execution?

Speculative execution means that instructions are executed before the processor is certain that they are in the correct execution sequence. It must be noted that no processor registers or memory locations are updated until it is confirmed that these instructions should indeed be executed.

64. Define static and dynamic branch prediction.

A decision on which way to predict the result of the branch may be made in hardware by observing whether the target address of the branch is lower than or higher than the address of the branch instruction.

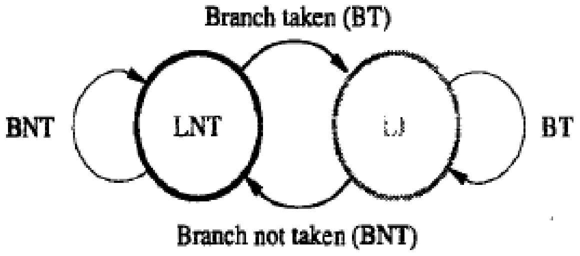
The branch prediction decision is always the same every time a given instruction is executed is called static branch prediction. The approach in which the branch prediction decision may change depending on execution history is called dynamic branch prediction.

65. What is a 2-state algorithm?

The branch prediction algorithms are to reduce the probability of making a wrong decision, to avoid fetching instructions that eventually have to be discarded. The algorithm may be described by the two-state machine. The two states are:

LT: Branch is likely to be taken

LNT: Branch is likely not to be taken



66. Represent a 4-state machine algorithm.

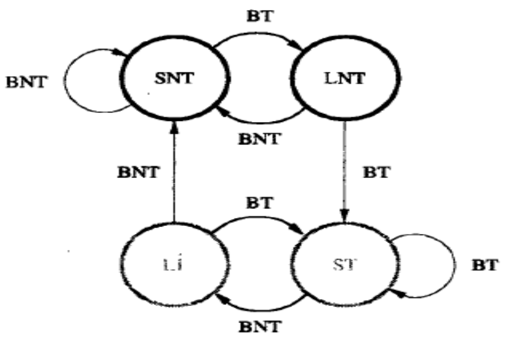
The four states are:

ST: Strongly likely to be taken

LT: Likely to be taken

LNT: Likely not to be taken

SNT Strongly likely not to be taken



67. What is misprediction?

The state information used in dynamic branch prediction may be recorded in a look-up table. It is possible for two branch instructions to share the same table entry. This leads to branch being mispredicted, but it does not cause an error in execution. Misprediction only introduces a small delay in execution time.

68. State the advantage and disadvantage of complex addressing modes.

Complex addressing modes involve several accesses to memory that do not necessarily lead to faster execution.

The main advantage is that they reduce the number of instructions needed to perform a given task and thereby reduce the program space needed in the main memory.

The disadvantage is that their long execution times cause the pipeline to stall, thus reducing its effectiveness. They require more complex hardware to decode and execute them.

69. State the features of addressing modes used in modern processors.

- Access to an operand does not require more than one access to the memory.

- Only load and store instructions access memory operands.
- The addressing modes used do not have side effects.

The addressing modes that have these features are register, register indirect and index.

70. List the way condition codes are to be handled.

1. To provide flexibility in reordering instructions, the condition-code flags should be affected by as few instructions as possible.

2. The compiler should be able to specify in which instructions of a program the condition codes are affected and in which they are not.

71. What are superscalar processors?

A processor can be equipped with multiple processing units to handle several instructions in parallel in each processing stage. Hence, several instructions can start execution in the same clock cycle, and the processor is said to use multiple-issue. Such processors are capable of achieving an instruction execution throughput of more than one instruction per cycle. They are known as superscalar processors.

72. What is deadlock?

A deadlock is a situation that can arise when two units, A and B, use a shared resource. Suppose that unit B cannot complete its task until unit A completes its task. At the same time, unit B has been assigned a resource that unit A needs. If this happens, neither unit can complete its task.

73. How to prevent a deadlock that arises during instruction execution?

If instructions are dispatched out of order, a deadlock can arise. To prevent deadlocks, the dispatcher must take many factors into account. Issuing instructions out of order increases the complexity of the dispatch unit. Hence, most processor use only in-order dispatching.

74. How to indicate the performance? What tool is used?

A useful performance indicator is the Instruction throughput, which is the number of instructions executed per second. For sequential execution, the throughput P_s is given by,

$$P_s = R/S$$

Any time a pipeline is stalled, the instruction throughput is reduced. The performance is highly influenced by factors such as branch and cache miss penalties.

11 MARKS

1. Explain about Single bus organization of the Data path inside a processor:

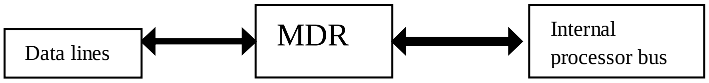
To execute an instruction, the processor has to perform the following steps:

- Fetch the contents of the memory location pointed to by the pc. The contents of this location are interpreted as an instruction to be executed.
- Hence, they are loaded in to the IR. Symbolically, this can be written as
  - IR <- [[PC]]
  - Assuming that the memory is byte addressable increment the contents of the pc by 4 that is
  - PC <-[PC] +4

Carry out the actions specified by the instruction in the IR. The first two steps are termed as fetch step and the 3rd step is known as execution phase

The fig shows an organization of the data path inside a processor with a single bus. The bus is internal to the processor that connects the processor to the memory and IO devices

- The data and address lines of the external memory are connected to the internal processor bus via the memory data register (MDR) and the memory address register (MAR). The registers MDR has two inputs and two outputs.



Data may be loaded into MDR either from the memory bus or from the internal process bus. The data stored in MDR may be placed on either bus.

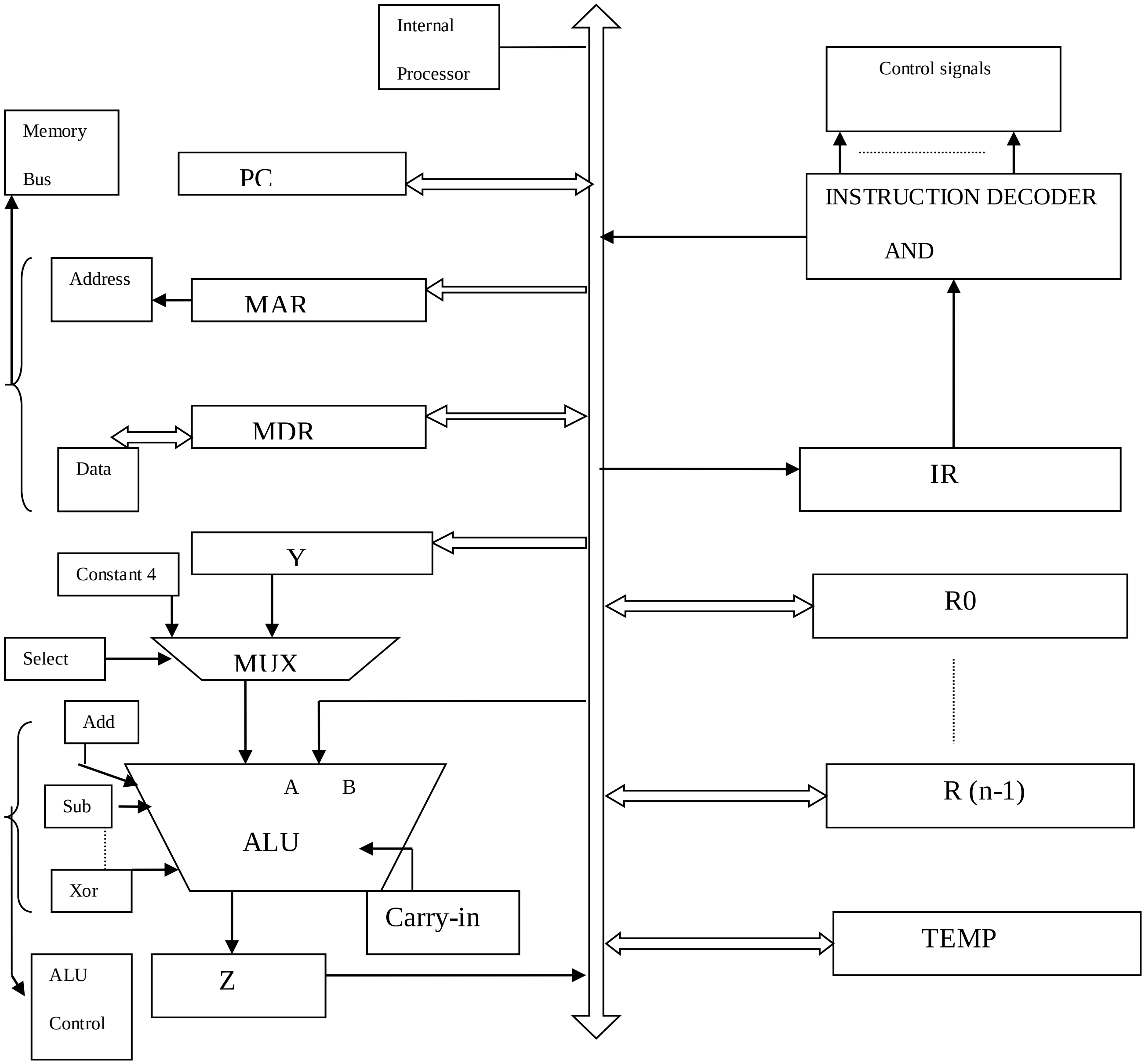
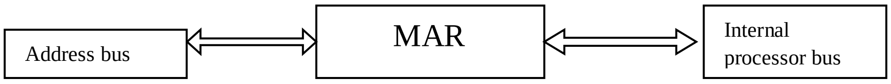


Fig: Single-Bus Organization Of The Data Path Inside A Processor

- The input of MAR is connected to the internal bus and its output is connected to the external bus.



- The control lines of the memory bus are connected to the instruction decoder and controls logic block. This unit is responsible for issuing the signals that control the operation of all units inside the processor and for interacting with the memory bus.
- The registers R0 through R (n-1) in the general purpose register the number of register and the use of processor may vary from one processor to another.

- Some register are dedicated as special purpose register such as index register, stack pointers, accumulator etc. for example in the diagram y, z, temp are used by the programmer for any instruction.
- The multiplexer MUX selects either the output of register Y or a constant value to be provided as input A of the AW.The constant 4 is used to increment the contents of the program counter.
- As instruction execution progresses, data are transferred from one register to other of the ten passing through the ALU to perform some arithmetic or logical operation.
- The instruction decoder and control logic unit is responsible for implementing the actions specified by the instruction loaded in the IR register.
- The registers, ALU and inter connecting bus are collectively referred to as the data path.
- The sequence of execution of instruction is as follows:
  - ✓ Transfer a word of data from one processor register to another or to the ALU.
  - ✓ Perform arithmetic or a logic operation and store the result in a processor register
  - ✓ Fetch the contents of a given memory location and load them into a processor register
  - ✓ Store a word of data from a processor register into a given memory location.

Some of the operations performed while executing an instruction are:

1. Register transfer
2. Arithmetic or logic operation
3. Fetching a word from memory
4. Storing a word in memory

Register transfer:

- ➢ Instruction execution will be faster if the operands are stored in registers. During execution of an instruction data may be transferred from one register to another.
- ➢ The registers are connected to the bus via switches controlled by the signals Riin and Riout, when Riin is set to 1, the data on the bus are loaded into Ri are placed on the bus.

The data transfer from R1 to R2 is done as follows:

- Enable R1out by setting it to 1. The content of R1 is now available on the processor internal bus.
- Enable R2in by setting H to 1. This loads data from the processor bus in to R2.

- ➢ Depending on the operations to be done, the control signals associated with the registers are asserted at the start of the clock cycle, as the flip-flops that the resistors are edge triggered.

- When edge triggered flip-flops are not used, two or more clock signals may be needed to guarantee proper transfer of data. This is called multiphase clocking.

The instruction MOV R1, R2 is done as follows.

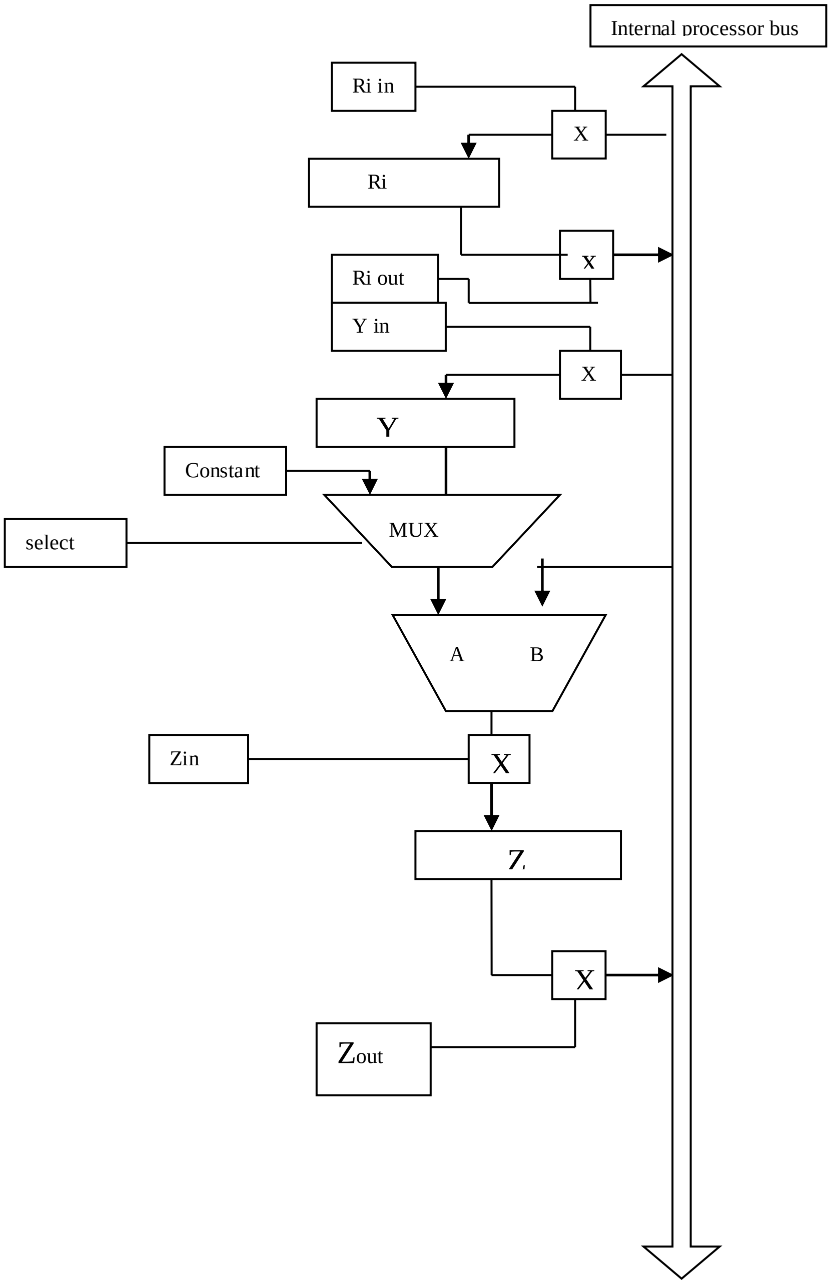


Fig: Register Transfer

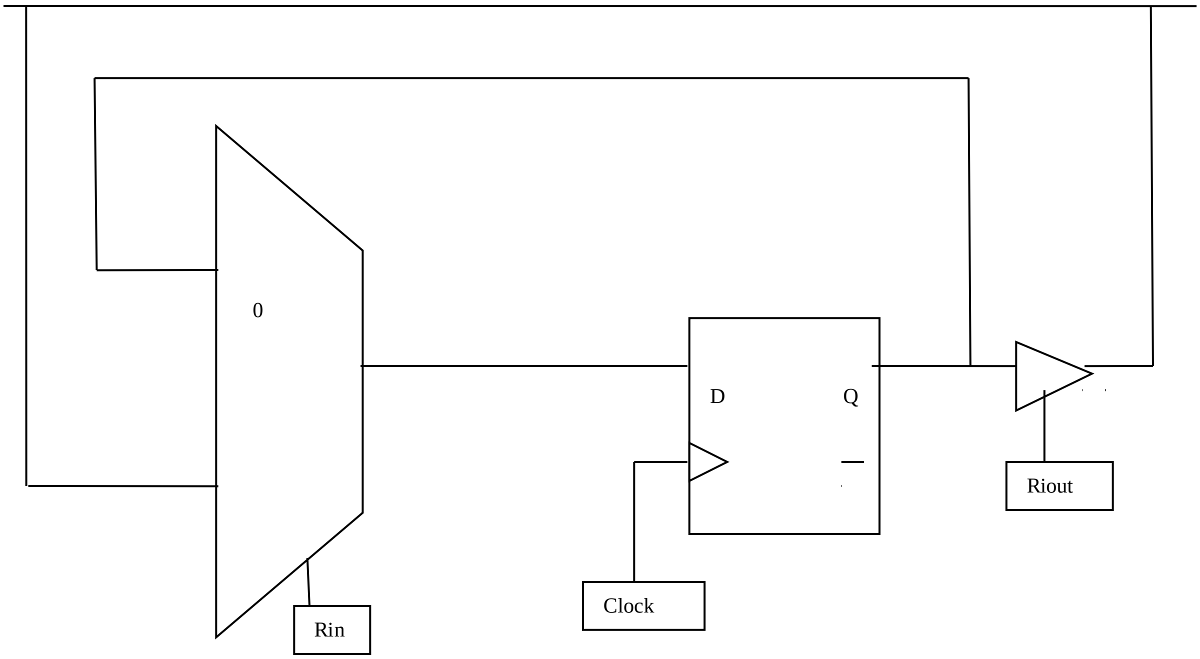


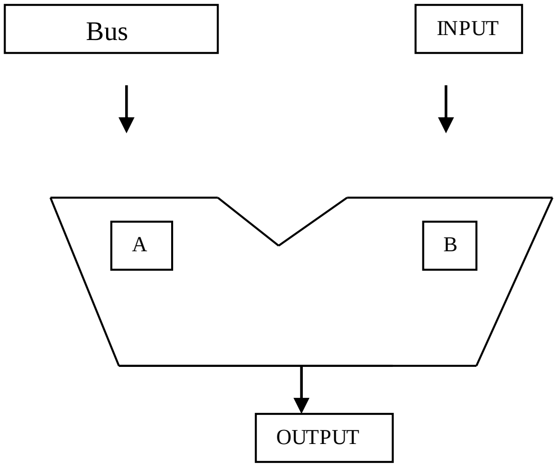
Fig: Input And Output Gating For One Register Bit

Performing an arithmetic or logic operation:

The arithmetic and logic unit does the computations. During computations, the data may be taken from the registers or from the memory via the bus. For e.g.: as shown in fig the ALU takes two inputs, one from the MUX, which selects either the constant 4 or one of the registers and other input from the bus. The result is then stored in a temporary register z which is then transferred to either any one of the general purpose registers R0 to Rn, or to the memory via the bus. Hence the sequence of operation to add R1 and R2 and then store the result in register R3 is

R3=R1+R2 is

1. *R1 out, Y in*
2. *R2 out, select Y, add, Z in [Selects Y is control signal to the MUX to select the register content through Y register]*
3. *Z out, R3 in. [At any point of time.]*



Fetching a word from memory:

X	Y	Z	
0	0	0	ADD
0	0	1	SUB
0	1	0	DIVIDE
0	1	1	MULTIPLY
1	0	0	AND
1	0	1	OR
1	1	0	NOT
1	1	1	XOR

- ✓ To fetch a word from memory, the processor has to specify the address of the memory location where this information is stored and request a read operation.
- ✓ The processor transfers the required address to MAR, where o/p is connected to the address lines of the memory bus.
- ✓ The requested data from the memory is stored in registers MDR, form where they are transferred to other registers in the processor.

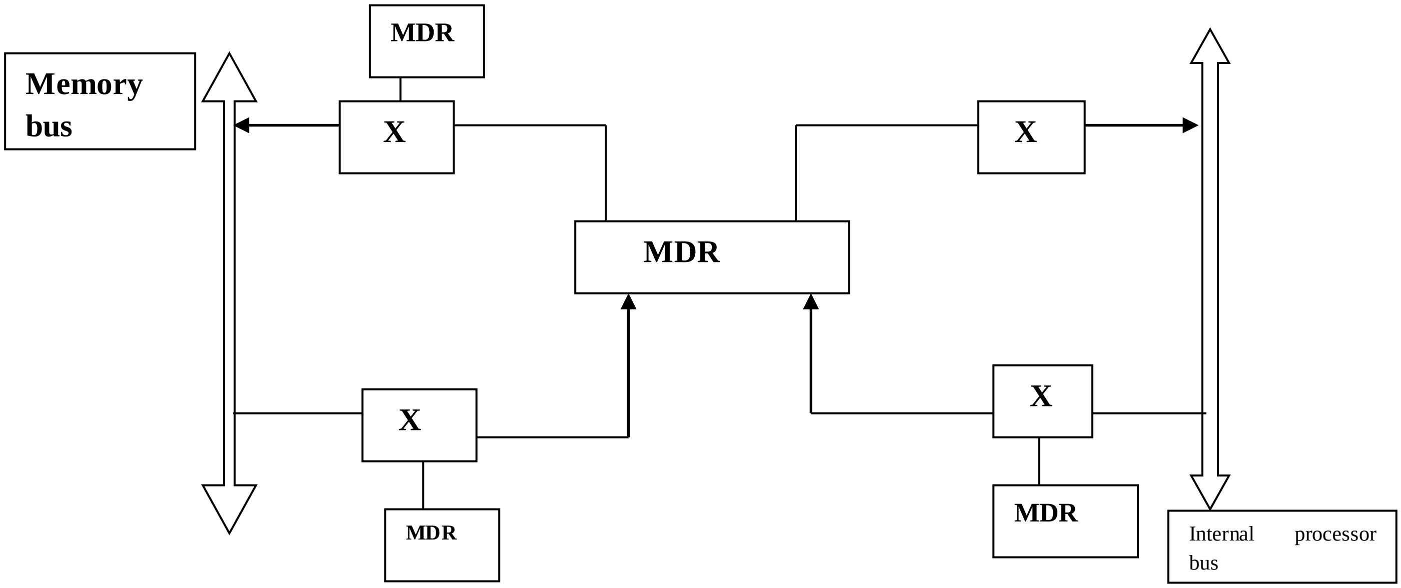


Fig: Connection And Control Signals For Register MDR

✓ It has four control signals:

- MDR in and MDR outE control the connection to the internal bus
- MDR inE and MDR outE control the connection to the external bus
- During memory read and write operation, the timing of internal. Processor operations must be coordinated with the response of the addressed device on the memory bus.
- The processor completes one internal data transfer in one clock cycles.

Example:

To perform read operation, consider the instruction move R1, R2. This instruction execution is given as follows.

1. MAR<-[R1]
2. Start a read operation on the memory bus.
3. Wait for the control signal called memory function completed (MFC)
4. Load MDR from the memory bus.

5. R2<-MDR.

The control signals is being activated as follows for the above instruction

1. R1 Out, MAR, read.
2. MDRin E, WMFC (wait for memory function complete)
3. MDRout, R2 in

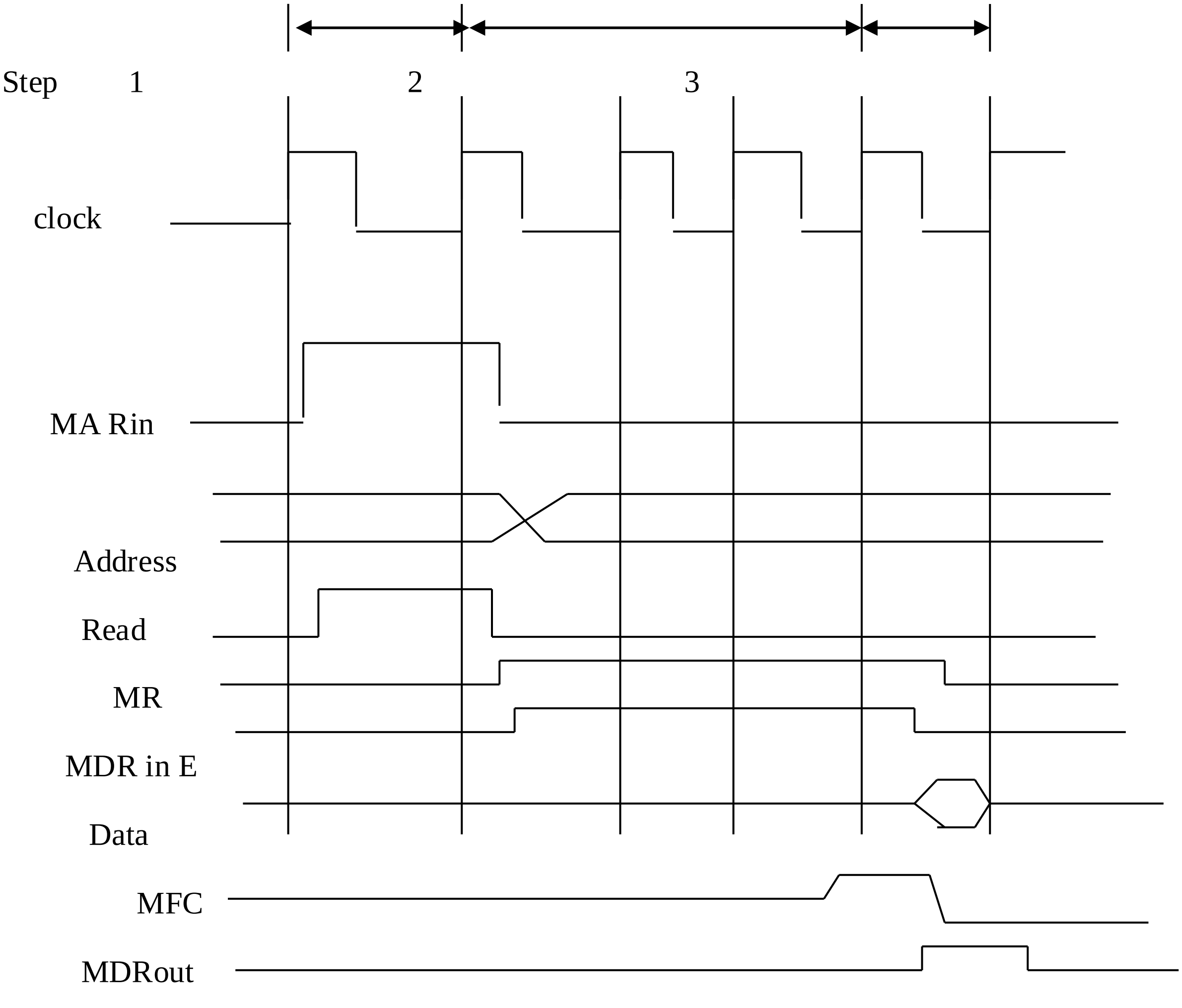


Fig: Timing Of A Memory Read Operation

Storing a word in memory:

Writing word into a memory location, the derived address is loaded into MAR. Then the data can be written are loaded into MDR and a write command is issued.

Hence executing the instruction MOV R2, (R1) requires the following sequence

1. R1out, MAR in
2. R2out MDR in, write
3. MDR out E, NMFC.

The processor remains in step3 until the memory operation is completed and as MFC response is received.

2. List and explain the steps involved in the execution of a complete execution.

Consider the instruction which adds the content of memory location pointed to by R3 to Register R1.

Actions:

The following are the actions while executing an instruction

i) Fetch the instruction

ii) Fetch the first operand

iii) Perform the addition

iv) Load the result into R1

The control sequence for execution of the above instruction:

Step	action	Comments
1	PCout,MARin,read,select 4,add,Zin	Load pc in MAR Issue read request to memory. select 4 to Y.do the add operation result stored in Z [PC<-PC+4->word size)]
2	Zout,Pcin,Vin,WMFC	Load pc with next address(INR).wait until memory responds
3	MDRout,IRin	Load 1nstr from MDR to IR
4	R3out,MARin,read	Read the first operand pointed to by R2(from memory)
5	R1out,Yin,WMFC	Enable Riout to transfer the second operand to yin
6	MDRout,select Y, Add, in	Add the operand Y&R1and store the result in z.
7	Zout,R1in,End	Transfer the result back

		to resulted R

STEP	ACTION	COMMENTS
1	PC out, MAR in,read,select 4,add, Zin	Increment PC
2	Z out,PC in, Y in, WMFC	Update PC and wait for memory
3	MDR out, IR in	Instruction fetch in IR
4	Offset-in-IR out add, Z in	Target address calculation pc+offset given in branch instruction
5	Zout, PC in, end	Load pc with new address(target address)

The steps 1 to 3 constitute the instruction fetch phase:

- The contents of PC is loaded intoMAR and read request is sent.
- Select signal is used to select the constant 4
- The value is added to the operand and the result is stored in register Z.
- The updated value is moved from register Z back into the PC in step 2.
- The word fetched from the memory is loaded into the IR.
- The steps 4 to 7 constitute the instruction decoding phase:
- The contents of R3 are transferred to the MAR in step4 and memory read operation is initiated.
- When read operation is completed, the memory operand is available in register MDR and the addition operation is performed in step 6.
- The sum is stored in register Z and transferred to R1 in step 7.
- The End signal causes a new instruction fetch cycle to begin by returning to step 1.

Branch instruction:

A branch instruction replaces the contents of the PC with the branch target address. This address is usually obtained by adding an offset X, which is given in branch instruction.

There are two types of branch

1. Unconditional branch
2. Conditional branch

Control sequence for an unconditional branch instruction:

- The steps 1 to 3 constitute the fetch phase and it ends when the instructions is loaded into the IR in step3.
- Since the value of updated PC is already available in register Y, the offset X is gated onto the bus in step 4.
- The result which is the branch target address is loaded into the PC in step5.
- For the conditional branch:
- The ststus of the condition codes must be checked before loading a new value into thw PC.
- The step 4 in the above control sequence must be replaced with Offset-field.
- Thus if N=0 the processor returns to step 1 immediately after step 4.
- If N=1 step 5 is performed to execute branch instruction.

3. Explain multiple bus organization in detail. OR Explain the execution of a three operand instruction using multiple bus organization

Multiple bus organization

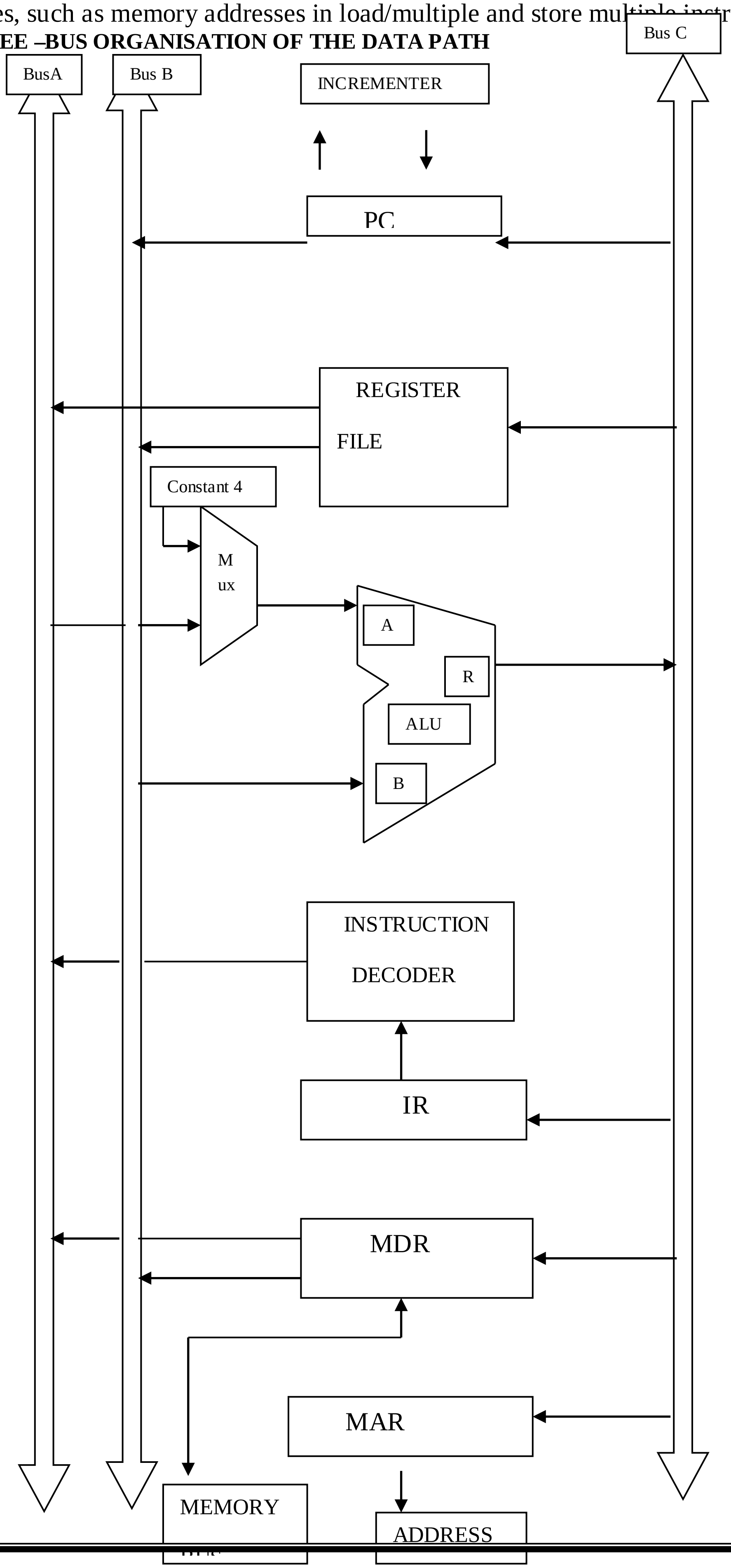
- It shows the three bus structure used to connect the registers and the ALU of a processor. Using 3-bus structure we can reduce the number of steps needed.
- All the general purpose registers are combined into a single block called the register file. The register file is said to have 3 ports.
- There are 2 ports allowing the contents of 2 different registers to be accessed simultaneously and have their contents placed on the buses A and B .The third port allows the data on bus C to the loaded into a third register during the same clock cycle.

FEATURES:

- ➢ Buses A and B are used to transfer the source operand to the A and B input of the ALU ,Where an arithmetic or logic operation may be performed. The result is transferred to the destination over bus C. If needed, the ALU may simply pass one of its 2 input operands unmodified to bus C. We will call the ALU control signals for such an operation R = A or R = B.
- ➢ A second feature is the introduction of the incriminator unit, Which is used to increment the PC by A. Using the incriminator eliminates the need to add 4 to the PC using the main ALU as was done in the figure.

The source of the constant 4 at the ALU input multiplier is still useful. It can be used to increment other addresses, such as memory addresses in load/multiple and store multiple instruction.

Fig:THREE –BUS ORGANISATION OF THE DATA PATH



Consider the 3 operand instruction

ADD R4,R5,R6.

The control sequence for executing this instruction is

STEP	ACTION
1	PC_{out}, R = B, MAR_{in}, Read, IncPC
2	WMF_c
3	MDR_{out} B,R = B,IR_{in}
4	$R4_{out}$,R5 $_{out}$ B,Select A,ADD,$R6_{in}$,END

STEP 1: The contents of PC are passes through the ALU , using R = B control signal, and loaded into the MAR to start a memory read operation. At the same time the PC is incremented by 4.The value is loaded into MAR is the original contents of the PC. The incremented value is loaded into the PC at the end of the clock cycle and will not effect the contents of the MAR.

STEP 2:The processor waits for MFC and loads the data received into MDR, then transfers them into IR.

STEP 3: Finally the execution phase of the instructs requires only one control step to complete. By introducing more paths for data transfer a significant reduction in the number of clock cycles needed to execute an instruction is achieved.

4. Explain the various design methods of hardwired control unit.

Or Describe the control unit organization with a separate Encoder and Decoder functions in a hardwired control

Hardwired control:

To execute instruction, the processor must have some means of generating the control signals needed in proper sequence. The 2 categories are

(i) Hardwired control

(ii) Micro programmed control

Hardwired control:

- A Hardwired control unit consists of combinational circuits to generate various control signals. It shows an overall block diagram of the hardwired control unit

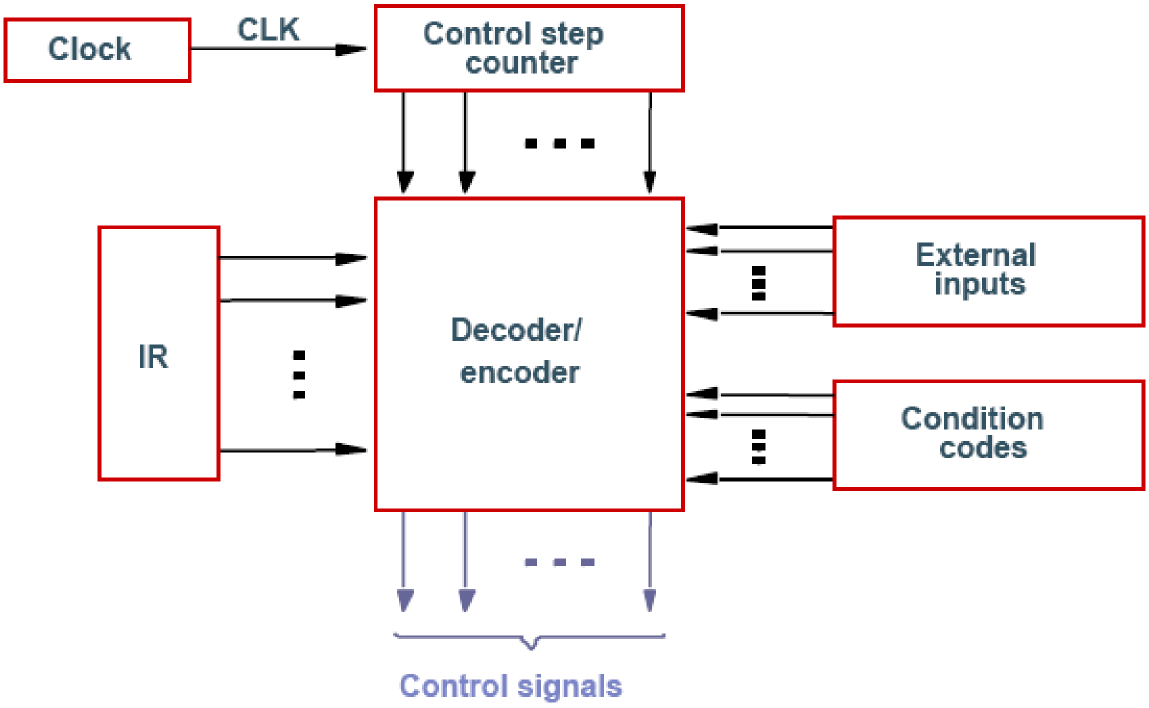


Fig. Control Unit Organization

- The opcode is the main input to the control unit. It is an input to the opcode decoder.
- The decoder/encoder block is a combinational circuit that generates the required control output, depending on the state of its inputs. Consider the sequence of control signals for the instruction ADD(R3), R1

1. PC_{out},MAR_{in},Read,Select 4,Add,Z_{in}

2. Z_{out},PC_{in},Y_{in},WMF_{c}

3. MDR_{out},IR_{in}

4. $R3_{out}$,MAR_{in},Read

5. $R1_{out}$,Y_{in},WMF_{c}

6. MDR_{out},Select Y,Add,Z_{in}

7. Zo_{ut},$R1_{in}$,End

Each step in the sequence is completed in one clock period. A counter may be used to keep track of the control steps, are shown in the figure. Each state or count of this counter corresponds to one control step. The required control signals are determined by the following instruction.

1. Contents of the control step counter

2. Contents of the instruction register

3. Contents of the condition code flags

4. External input signals such as MFC and interrupt requests.

The figure shows the separation of the decoding and encoding functions.

- The step decoder function provides a separate signal line for each step, or time slot, in the control sequence.
- Similarly, the output of the instruction decoder consists of a separate line for each machine instruction. For any instruction loaded in the IR, one of the outlines INS, through INS_{in}, is set to 1, and all other lines are set to 0..
- The input signals to the encoder block in figure are combined to generate the individual control signals Y_{in}, PC_{out}, Add, End and so on.

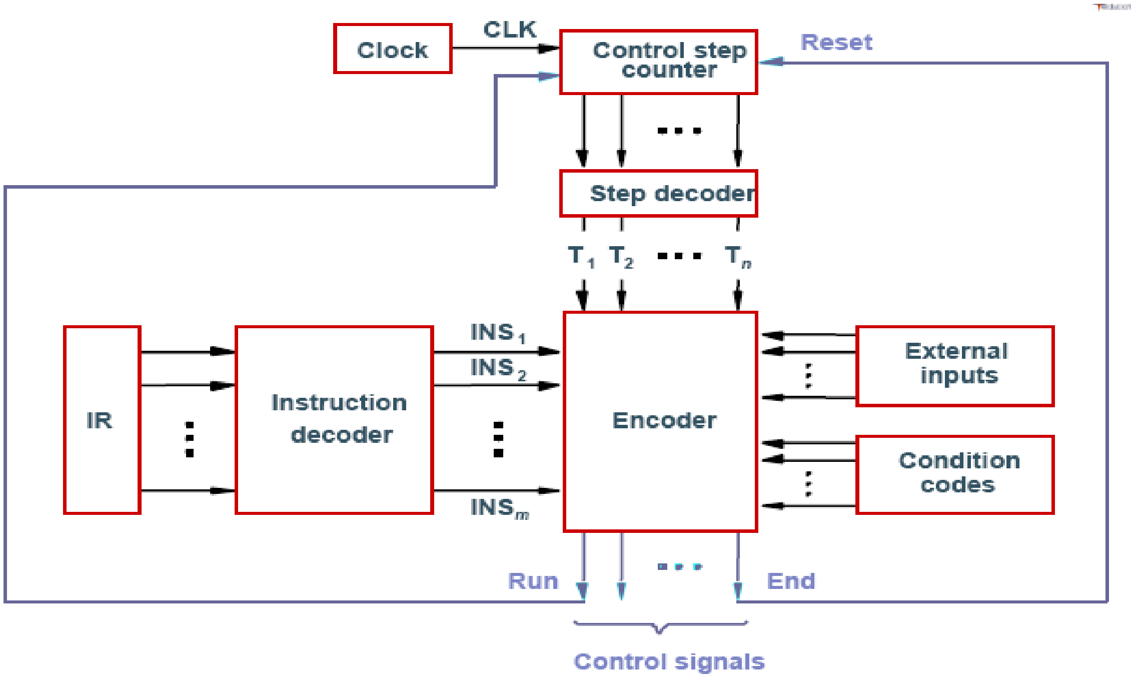
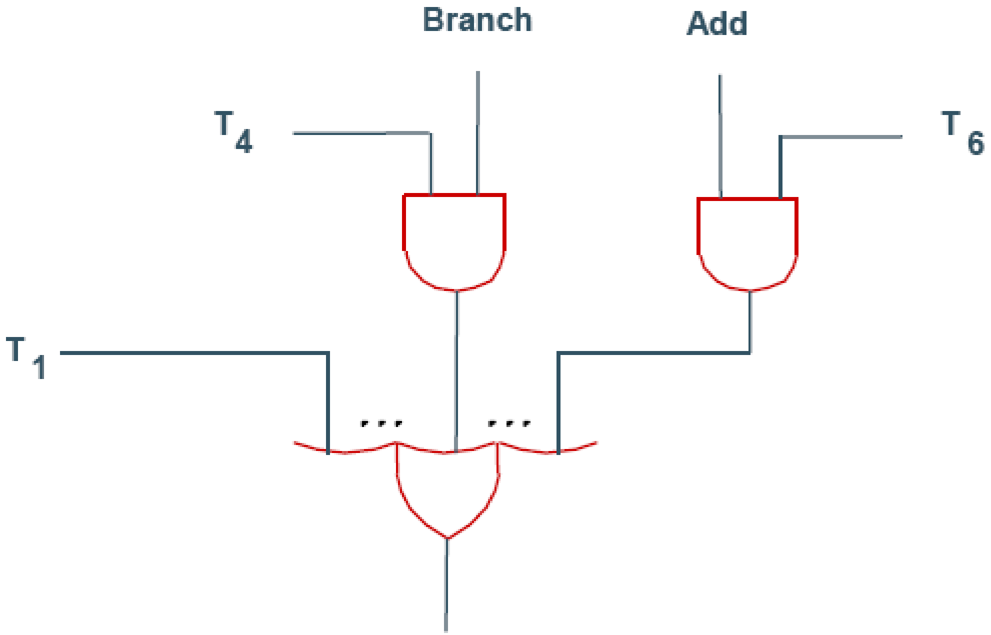


Fig. Separation of decoding and encoding functions

Let us see how the encoder generates signal for single bus processor organization shown in figure



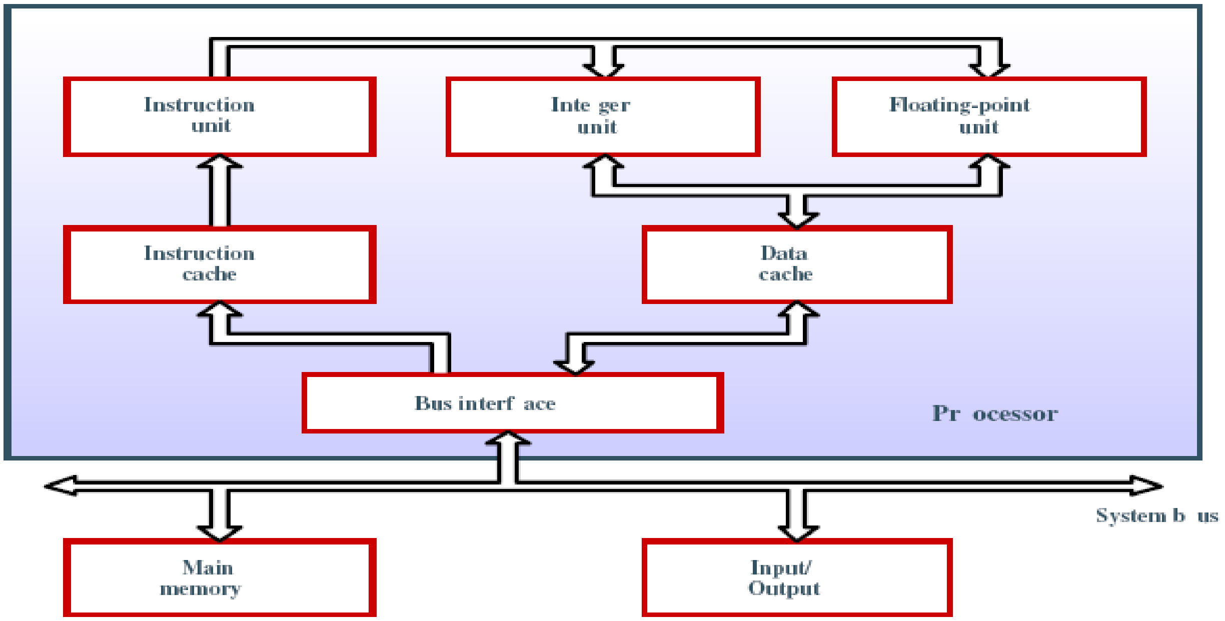
- Z_{in},The encoder circuit implements the following logic functions to generate Z_{in}.The logic function for Z_{in} is derived from the control sequence in figure.

The end signal starts a new instruction fetch cycle by resetting the control step counter to its starting value.

- Another control signal is called RUN is set to 1,RUN causes the counter to be implemented by one at the end of every clock cycle. When RUN is equal to 0, the counter stops counting. This is needed whenever the WMF_c signal is issued, to cause the processor to wait for the reply from the memory.
- The sequence of operation carried out by machine is determined by the wiring of logic elements, hence the name hard-wired.
- The controller that uses this approach can operate at high speed. However, it has little flexibility, and the complexity of the instruction set it can implement is limited.

5. Draw and explain the block diagram of a complete processor

A Complete Processor



- The processor consists of instruction unit, integer unit, floating point unit, instruction cache, data cache, bus interface unit, main memory module and input/output module.

- The instruction unit fetches instruction from the instruction cache or from the main memory. The complete processor provides 2 processing units floating point and integer unit.

- These 2 units gets data from data cache.The processor provides bus interface unit to control the interface of processor to system bus, main memory module and input/output modules.

Advantages of hardwired control unit:

- A hardwired control unit works fast.

- The combinational circuits generates the control signals based on the input signal status. As soon as the required input signal combination takes place, immediately the output is generated by a combinational circuit.

Disadvantages of hardwired control unit:

- If the CPU has the large number of control points, the control unit design becomes very complex. It is tedious to design the pulse distributor circuitry since several gates input and output have to be kept tracks of during designing.

- The design does not give any flexibility. If any modification is required, it is extremely difficult to make the correction. Design modification becomes necessary under the following situations:

1. There is a design mistake in the original gates.

2. A new feature is to be added to an existing design.

3. A new hardware component of higher speed is available, which will improve the performance.

6. Explain micro programmed control unit. What are the advantages and disadvantage of it? OR With a neat diagram explain the internal organization of a processor. (Apr 11)

Micro programmed Control:

Microprogramming is a modern concept used to designing a control unit. It can be used for designing control logic for any digital system. Some common applications of input/output controllers such as disk controller, peripheral devices such as printer and hard disk drive.

- The microinstructions are executed when the corresponding control signals are made active. Each instruction needs a specific set of micro operation in an order .
- Micro programmed control, in which control signals are generated by a program similar to machine language programs. First, we introduce some common terms.

Control memory: The control signals needed for an instruction and the time sequence can be stored in the memory.

Control Word: CW is a word whose individual bits represent various control signals. Each of the control steps in the control sequence of an instruction defines a unique

combination of 1s and 0s in the CW. The CW corresponding to the 7 steps of fig.

Micro instruction	PC in	PC out	MAR in	read	MDR out	IR in	Y In	Select	add	Z in	Z out	R1 out	R1 in	R3 out	WM FC	end
1	0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
4	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0
6	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

ig: Example –Micro Instruction

- If the bit is 1, the corresponding control signal is activated. If the bit is 0, the control signal is not activated.
- When a control memory word is fetched from control memory some bits may be 1 and others 0.The control signals corresponding to '1' bits are generated. Thus, multiple micro operations are executed simultaneously.

Control Store:

A special memory in which the microinstructions for all instructions in the instruction set of a computer are stored in a control memory. It is also called as a control memory

Micro routine: A sequence of control words corresponding to the control sequence of a machine instruction constitutes the micro routine.

Micro instruction: Each control memory word is known as a microinstruction.

Basic organization of micro programmers control unit is shown fig.

- To read the control words sequentially from the control memory, a microprogram counter(MPC) is used. Every time a new instruction is loaded in the IR, the output of the block labelled ;starting address generator' is loaded into the MPC. 24
- The MPC is automatically incremented by the clock, causing successive microinstructions to be read from the control memory. Hence the control signals are delivered to various parts to the processor in the control sequence.

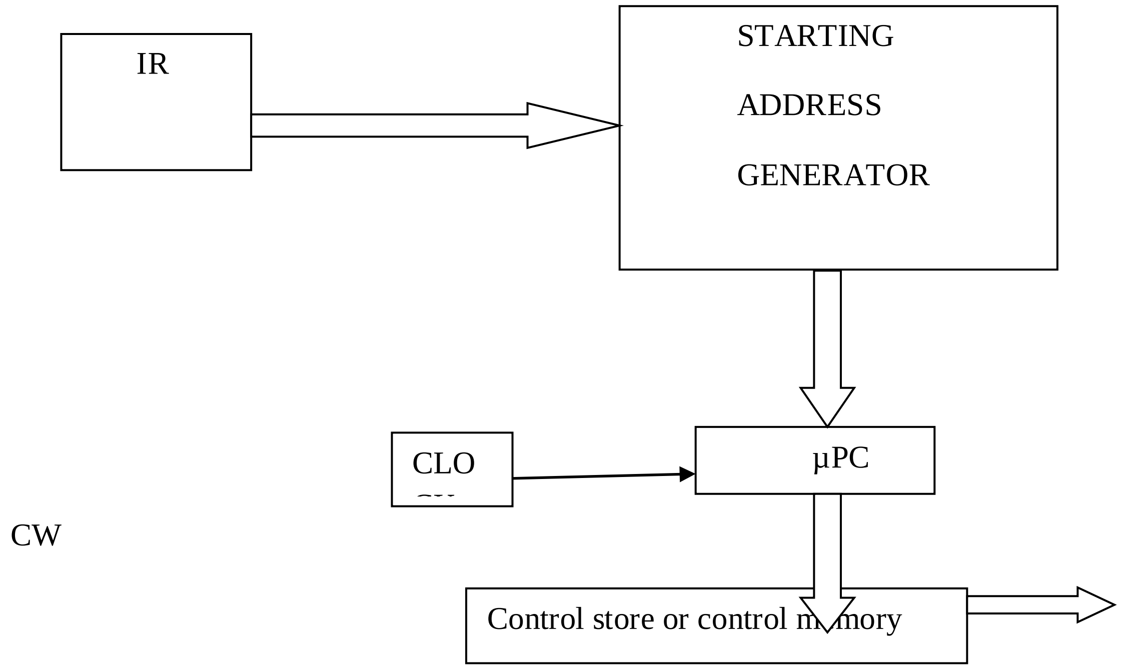


Fig: Basic Organization Of A Micro programmed Control Unit

- There is a situation arises when the control unit is required to check the status of the condition codes or external inputs to choose between alternative course of action. In micro programmed control the alternative approach is to use conditional branch microinstructions.

- In addition to the branch address, these microinstruction specify which of the external inputs, condition codes, or possibly bits of the instruction register should be checked as a condition for branching to take place.

- The instruction Branch < 0 may be implemented by a micro routine . After loading this instruction into IR, a branch microinstruction transfers control to the corresponding micro routine, which is assumed to start at location 25 in the control memory. This address is the starting address generator block in fig.

- The microinstruction at location 25 tests the N bit of the condition codes. If the bit is equal to 0, a branch takes place to location 0 to fetch a new machine instruction.

- Otherwise, the microinstruction at location 26 is executed to put the branch target address into register Z1. The microinstruction in location 27 loads this address into the PC.

- To support micro program branching, the organization of the control, unit should be modified as shown in fig.

The starting address generator block of a fig becomes the starting and branch address generator. This blocks a new address into the MPC when a microinstruction instructs it to do so.

To allow implementation of a conditional branch, inputs to this block consists of the external inputs and condition codes as well as the contents of the instruction register. In this control unit , the MPC is incremented every time a new instruction is fetched from the micro program memory, except in the following situations:

1) When a new instruction is loaded into the IR, the MPC is loaded with the starting address of the micro routine for that instruction.

2) When a branch microinstruction is encounted and the branch condition is satisfied, the MPC is loaded with the branch address

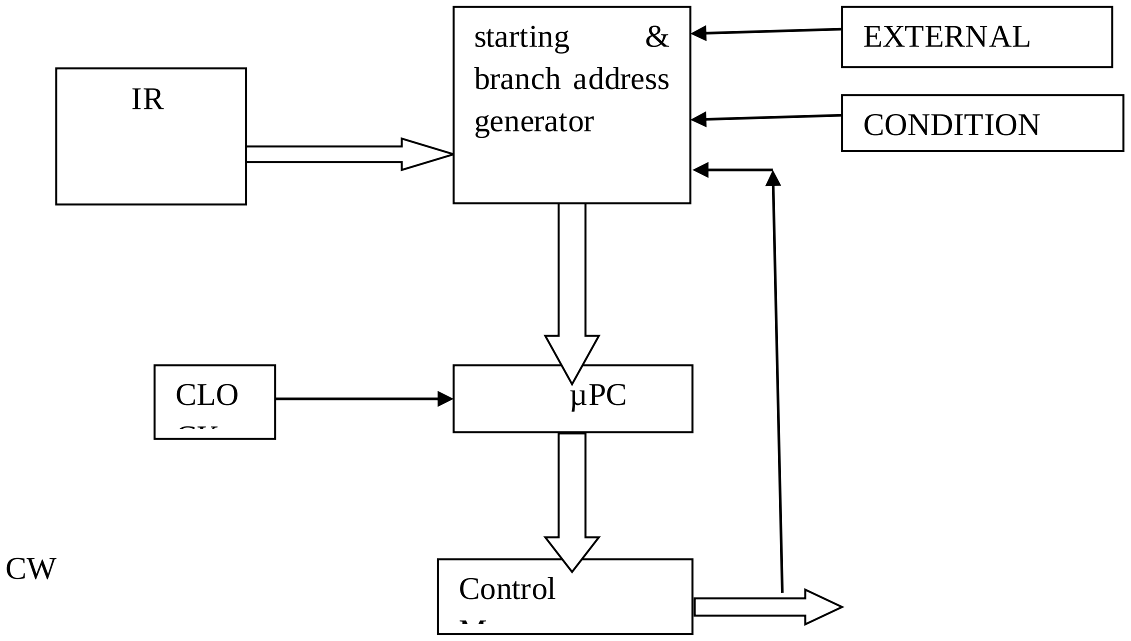


Fig:A Organization Of Control Unit To Allow Conditional Branching In The Micro program

ADDRESS	MICROINSTRUCTION
0	PC_{out}, MAR_{in}, Read, Select 4, Add, Z_{in}
1	Z_{out}, PC_{in}, Y_{in}, WMF_C
2	MDR_{out},IR_{in}
3	Branch to starting address of appropriate instruction
………	
25	If N = 0, then branch to microinstruction 0
26	Offset-field of IR_{out}, Select Y,Add,Z_{in}
27	Z_{out}, PC_{in},End

Fig: Micro routine For The Instruction Branch < 0

MICROINSTRUCTIONS

- A simple way to structure microinstruction is to assign one bit position to each control signal required in the CPU. However, this scheme has one serious drawback assigning individual bits to each control signal results in long microinstructions, because the number required signals is usually large.

- Moreover only a few bits are used in any given instruction. The solution of the problem is solved by grouping the control signals.

Grouping of control signals:

- Control signals can be grouped so that all mutually exclusive signals are placed in the same group. Thus, at most one micro operations per group is specified in any microinstruction. Then it is possible to use a binary coding scheme to represent the signals with a group.

Example:

4 bits suffice to represent 16 available functions in the ALU. Register output control signals can be placed in a group consisting of PC_{out}, $R0_{out}$, $R1_{out}$, $R2_{out}$, $R3_{out}$ and $TEMP_{out}$, Z_{out}, MDR_{out}.

Any one of these can be selected by a unique 4-bit code. Further natural groups can be made for the remaining signals.

Gating signals: IN and OUT signals
Control signals: Read, Write, Clear A, Set carry in WMFC, END etc.

ALU Signals:

- Add, Sub, etc. There are in all 39 signals and hence each microinstruction will have 39 bits. It is not all necessary to use all 39 bits for every microinstruction because by grouping of control signals we minimize number of bits for microinstructions.

Ways to reduce number of bits in microinstruction:

1)Most signals are not needed simultaneously

2) Many signals are mutually exclusive(e.g) only one function of ALU can be activated at a time.

3) A source for data transfers must be unique which means that it should not be possible to get the contents of 2 different registers on to the bus at the same time.

4) Read and Write signals to the memory cannot be activated simultaneously.

The below figure shows an example of a partial format for the microinstructions, in which each group occupies a field large enough to contain the required codes. Most fields much include one inactive code for the case in which no action is required.

Fig:An example of a partial format for field encoded microinstruction

F1	F2	F3		

F1(4 bits)	F2(3 bits)	F3(3 bits)	F4(4 bits)	F5(2 bits)
0000: No transfer	0000:No transfer	0000:add transfer	00:Noaction	00:Noaction
0001:PC_{out}	001:PC_{in}	001:MAR_{in}	0000:ADD	01: Read

0010:MDR_{out}	010:IR_{in}	010 MDR_{in}	0001:SUB	10:Write
0011:Z_{out}	011:Z_{in}	011:TEMPin	.	16 ALU Functions
0100:$R0_{out}$	100:$R0_{in}$			100:Yin .
0101: $R1_{out}$	101: $R1_{in}$		1111 :XOR	
0110: $R2_{out}$	110: $R2_{in}$			
0111: $R3_{out}$	111: $R3_{in}$			
1010:$TEMP_{out}$				
1011:$Offset_{out}$				

F6	F7

F6(1bit)	F7(1bit)	F8(1bit)
0:select Y	0:No action	0:Continue
1:select 4	1:WMF_C	1:End

- ➢ For example, the all zero pattern in F1 indicates that none of the registers that may be specified in this field should have its contents placed on the bus. An inactive code is not needed in all fields.
- ➢ For example, F4 contains 4 bits that specify one of the 16 operations performed in the ALU. Since no space code is included, the ALU is active during the execution of every microinstruction.
- ➢ However its activity is monitored by the rest of the machine through registerZ,which is loaded only when the Z_{in} signal is activated.
- ➢ Grouping control signals into fields requires a little more hardware because decoding circuits must be used to decode the bit patterns of each field into individual control signals.

Techniques of grouping of control signals:

The grouping of control signals can be done either by using technique called vertical organization or by using technique called horizontal organization.

- Highly encoded scheme that use impact codes to specify only a small number of control function in each microinstruction are referred to as a vertical organization.
- On the other hand, minimally encoded scheme, in which resources can be controlled with a single instruction, is called a horizontal organization.

Microinstruction sequencing:

Micro program sequencing is done by micro program sequencer. There are 2 important factors that must be considered while designing the micro program sequencer:

- The size of the microinstruction and
- The address generation time.

The size of the microinstruction should be minimum so that the size of control memory required to store microinstructions is also less. This reduces the cost of control memory with less address generation time, microinstructions can be executed in less terms,, resulting better throughput.

During execution of a micro program the address of the next microinstruction to be executed has 3 resources:

- Determined by instruction register
- Next sequential address
- Branch

- ➢ One of these 3 address sources first occurs once per instruction cycle. The second source is most commonly used. However if we store separate microinstructions for each machine instruction, there will be large number of microinstructions.
- ➢ As a result, large space in control memory is required to store these microinstructions. We want to organise the micro program such that they share as many microinstruction S POSSIBLE.
- ➢ This requires many branch microinstructions, both unconditional and conditional to transfer control among the various microinstructions. Thus it is important to design compact, time efficient techniques for microinstruction branching.

Consider instruction ADD, Ser, Rdst. The instruction adds the source operand to the contents of register Rdst and places the sum in Rdst, the destination register.

- ➢ It shows a flowchart of a micro program for ADD, Ser, Rdst instruction.
- ➢ Each box in the chart corresponds to a microinstruction that controls the transfers and operation indicated within the box. The microinstructions isolated at the address indicated by the actual number above the upper right corner of the box. Each octal digit represents 3 bits.

Consider the point labelled α in the fig. At this point it is necessary to choose between actions required by direct and indirect addressing modes.

- ➢ If the indirect mode is specified in the instruction, then the microinstruction is location 170 is performed to fetch the operand from the memory. If the direct mode is specified, This fetch must be bypassed by branching immediately to location 171.
- ➢ The remaining micro operations required to complete execution of instruction are same and hence shared by the instructions having operand with different addressing modes.

We can say branching allows sharing of microinstructions for different micro programs and it reduces the size of control memory.

Advantages of microprogramming:

- The design of micro program control unit is less complex.
- The micro program is flexible.
- The given CPUs instruction set can be easily modified by changing the micro programs without affecting the data path.
- The debugging and maintenance of a micro programmers CPU is easy.

Disadvantages of microprogramming:

- A micro programmed CPU is slow
- For a small CPU with very limited hardware resources, a micro program CU is expensive as compared to a hardwired control unit.

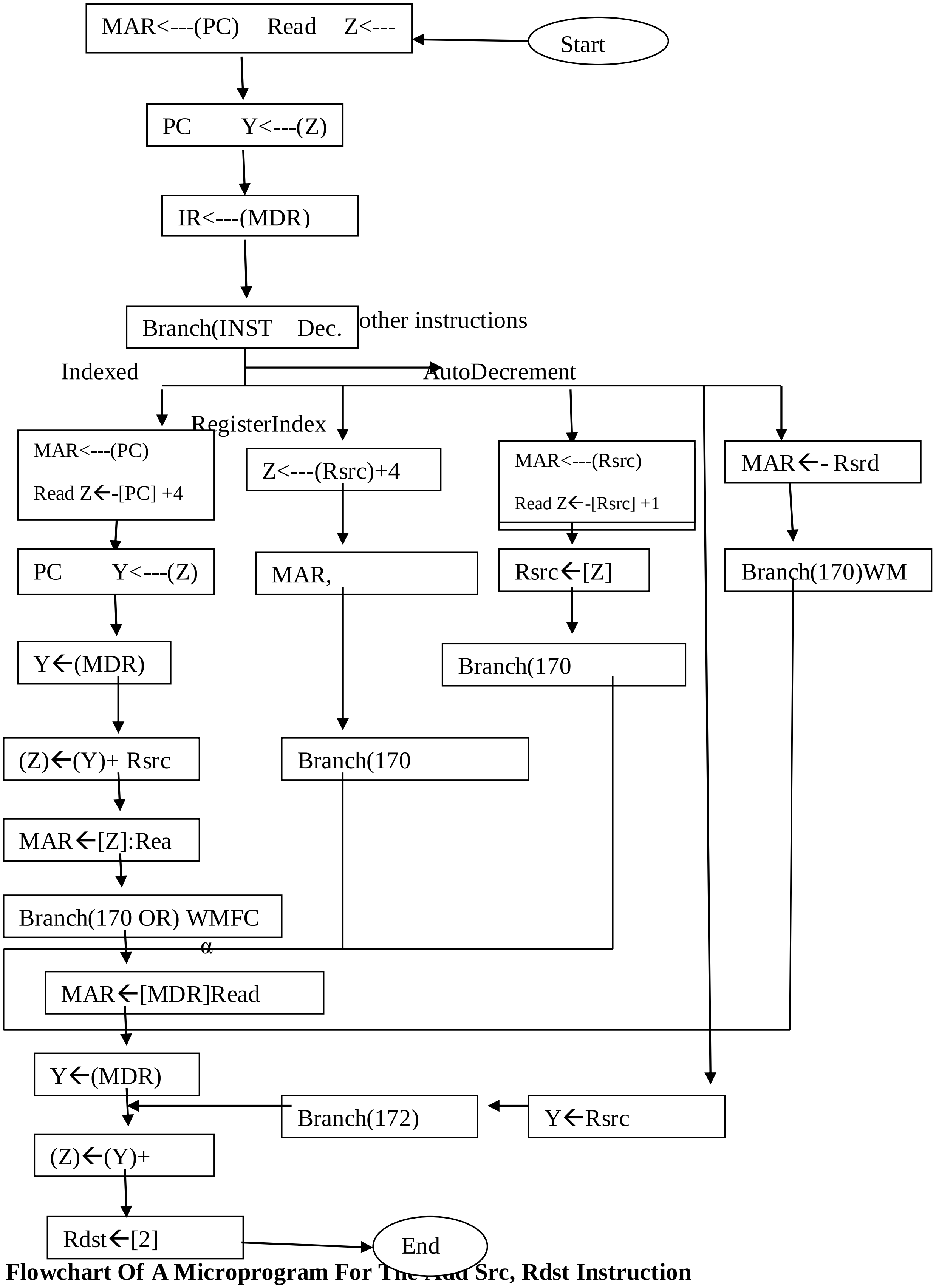


Fig: Flowchart Of A Microprogram For The Add Src, Rdst Instruction

7. Compare hardwired control unit and micro programmed control unit

S.No.	Attribute	Hardwired Control	Micro programmed Control
1.	Speed	Fast	Slow
2.	Flexibility	Implementation hardware	Implemented in software
3.	Ability to handle large/complex instructions	Somewhat difficult	Easier
4.	Design Process	Somewhat complication	Orderly and system active
5.	Ability to support operating system and diagnostic feature	Very difficult	Easy
6.	Applications	Mostly RISC	Mainframes, some microprocessor
7.	Chip area efficiency	Use least area	Use most area

8. Explain how control signals are generated using micro programmed control or

Draw the necessary diagrams and explain the control signal generation using micro programmed control. Or

How the functional field micro instruction is generated? Explain

Microinstructions:

- A simple way to structure microinstruction is to assign one bit position to each control signal required in the CPU. However, this scheme has one serious drawback assigning individual bits to each control signal results in long microinstructions, because the number required signals is usually large.
- Moreover only a few bits are used in any given instruction. The solution of the problem is solved by grouping the control signals.

<u>Grouping of control signals:</u>

- Control signals can be grouped so that all mutually exclusive signals are placed in the same group. Thus, at most one micro operations per group is specified in any microinstruction.
- Then it is possible to use a binary coding scheme to represent the signals with a group.

Example:

4 bits suffice to represent 16 available functions in the ALU. Register output control signals can be placed in a group consisting of PC_{out}, $R0_{out}$, $R1_{out}$, $R2_{out}$, $R3_{out}$ and $TEMP_{out}$, Z_{out}, MDR_{out}.

Any one of these can be selected by a unique 4-bit code. Further natural groups can be made for the remaining signals.

Gating signals: IN and OUT signals

Control signals: Read, Write, Clear A, Set carry in WMFC, END etc.

ALU Signals:

- Add, Sub, etc. There are in all 39 signals and hence each microinstruction will have 39 bits. It is not all necessary to use all 39 bits for every microinstruction because by grouping of control signals we minimize number of bits for microinstructions.

Ways to reduce number of bits in microinstruction:

1)Most signals are not needed simultaneously

2) Many signals are mutually exclusive(e.g) only one function of ALU can be activated at a time.

3) A source for data transfers must be unique which means that it should not be possible to get the contents of 2 different registers on to the bus at the same time.

4) Read and Write signals to the memory cannot be activated simultaneously.

The below figure shows an example of a partial format for the microinstructions, in which each group occupies a field large enough to contain the required codes.

- For example, the all zero pattern in F1 indicates that none of the registers that may be specified in this field should have its contents placed on the bus. An inactive code is not needed in all fields.
- For example, F4 contains 4 bits that specify one of the 16 operations performed in the ALU. Since no space code is included, the ALU is active during the execution of every microinstruction.
- However its activity is monitored by the rest of the machine through register Z, which is loaded only when the Z_{in} signal is activated.

Most fields much include one inactive code for the case in which no action is required.

F1	F2	F3		

F1(4 bits)	F2(3 bits)	F3(3 bits)	F4(4 bits)	F5(2 bits)
0000: No transfer	0000:No transfer	0000:add transfer	00:Noaction	00:Noaction
0001:PC_{out}	001:PC_{in}	001:MAR_{in}	0000:ADD	01: Read

0010:MDR_{out}	010:IR_{in}	010 MDR_{in}	0001:SUB	10:Write
0011:Z_{out}	011:Z_{in}	011:TEMPin	.	16 ALU Functions
0100:$R0_{out}$	100:$R0_{in}$	100:Yin	.	
0101: $R1_{out}$	101: $R1_{in}$		1111 :XOR	
0110: $R2_{out}$	110: $R2_{in}$			
0111: $R3_{out}$	111: $R3_{in}$			
1010:$TEMP_{out}$				
1011:$Offset_{out}$				

F6	F7

F6(1bit)	F7(1bit)	F8(1bit)
0:select Y	0:No action	0:Continue
1:select 4	1:WMF_C	1:End

Fig: An example of a partial format for field encoded microinstruction

.

- Grouping control signals into fields requires a little more hardware because decoding circuits must be used to decode the bit patterns of each field into individual control signals.

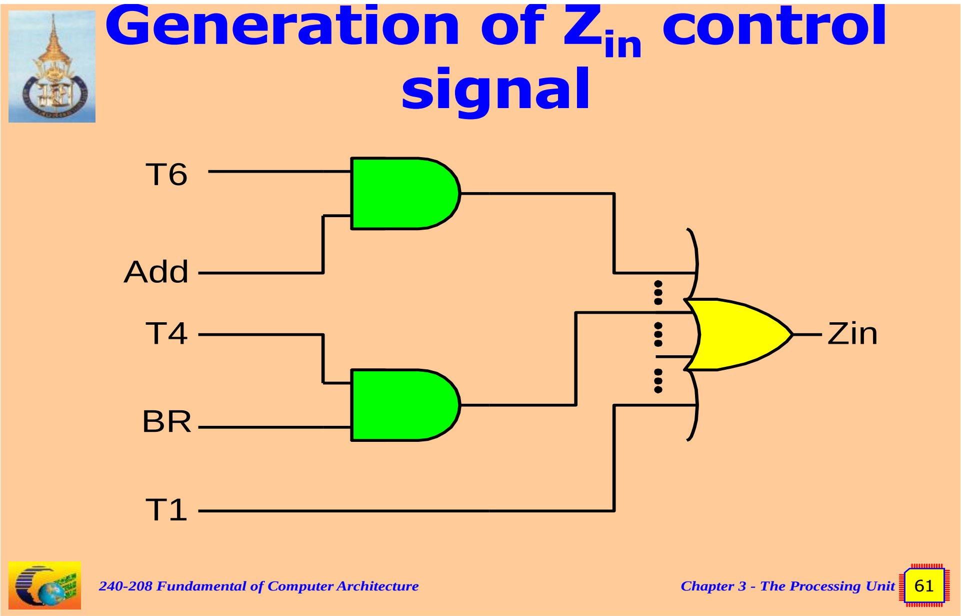
Techniques of grouping of control signals:

The grouping of control signals can be done either by using technique called vertical organisation or by using technique called horizontal organisation.

- Highly encoded scheme that use impact codes to specify only a small number of control function in each microinstruction are referred to as a vertical organization.
- On the other hand, minimally encoded scheme, in which resources can be controlled with a single instruction, is called a horizontal organization.

9. Generate thye logic circuit for the followin g functions and explain

(i). $Z_{in} = T_{1\,+} T_6 \,.\, ADD + T_{4.}BR +$ ....

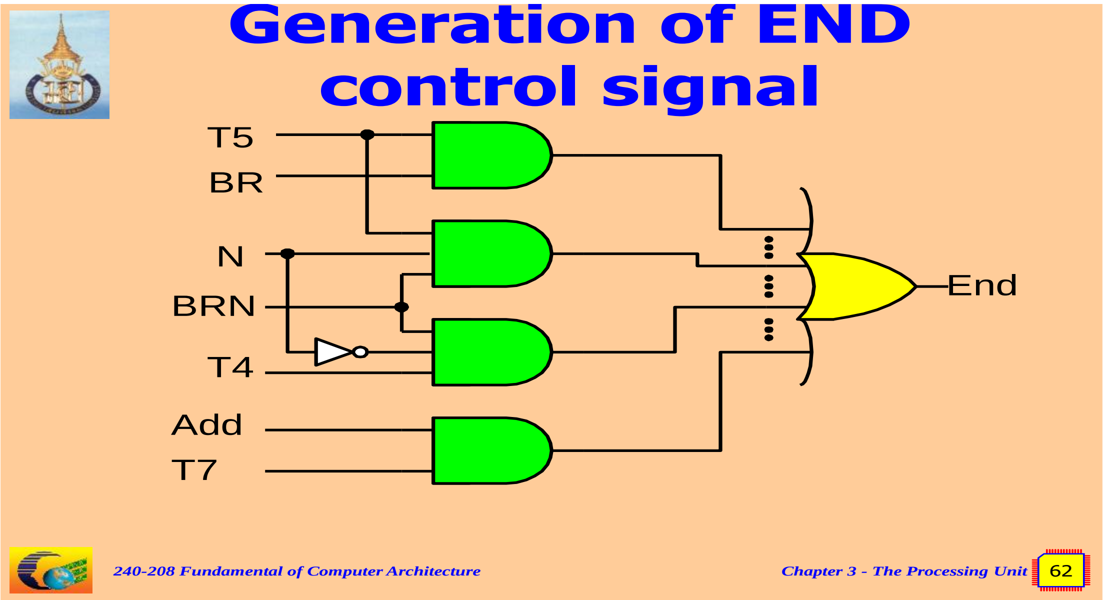


Generation of Z_{in} control signal

240-208 Fundamental of Computer Architecture

Chapter 3 - The Processing Unit 61

(ii).END = T_7. ADD + T_5 .BR+($T_{5.}$ N+ $T_{4.}$N).BRN+............



Generation of END control signal

240-208 Fundamental of Computer Architecture

Chapter 3 - The Processing Unit 62

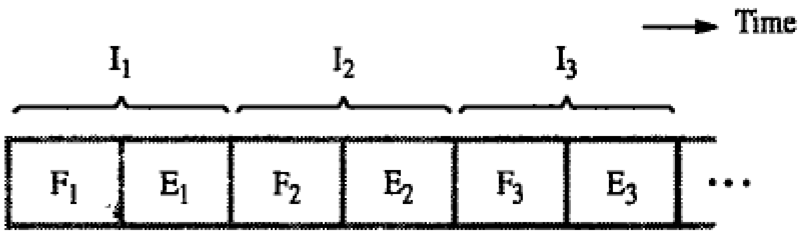
10. Explain the concepts of Pipelining.

Pipelining

It is a particularly effective way of organizing concurrent activity in a computer system.

Sequential execution

The processor executes a program by fetching and executing instructions, one after the other. Execution of a program consists of a sequence of fetch and execute steps.

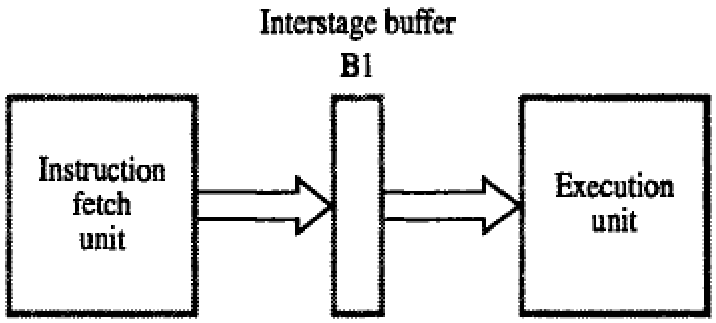


Hardware organization

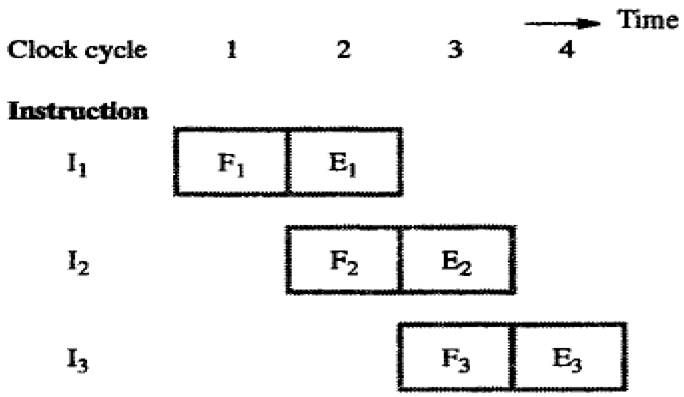
An inter-stage storage buffer, B1, is needed to hold the information being passed from one stage to the next.

New information is loaded into this buffer at the end of each clock cycle.

F Fetch: read the instruction from the memory
D Decode: decode the instruction and fetch the source operand(s)
E Execute: perform the operation specified by the instruction
W Write: store the result in the destination location



Pipelined execution



- In the first clock cycle, the fetch unit fetches an instruction I1 (step F1) and stores it in buffer B1 at the end of the clock cycle.
- In the second clock cycle the instruction fetch unit proceeds with the fetch operation for instruction I2 (step F2).

- Meanwhile, the execution unit performs the operation specified by instruction I1,which is available to it in buffer B1 (step E1).
- By the end of the second clock cycle, the execution of instruction I1 is completed and instruction I2 is available.
- Instruction I2 is stored in B1, replacing I1, which is no longer needed.
- Step E2 is performed by the execution unit during the third clock cycle, while instruction I3 is being fetched by the fetch unit.
- Four instructions are in progress at any given time. So it needs four distinct hardware units.
- These units must be capable of performing their tasks simultaneously and without interfering with one another.
- Information is passed from one unit to the next through a storage buffer.
- During clock cycle 4, the information in the buffers is as follows:
- Buffer B1 holds instruction I3, which was fetched in cycle 3 and is being decoded by the instruction-decoding unit.
- Buffer B2 holds both the source operands for instruction I2 and the specifications of the operation to be performed. This is the information produced by the decoding hardware in cycle 3.
    - The buffer also holds the information needed for the write step of instruction I2(step W2).
    - Even though it is not needed by stage E, this information must be passed on to stage W in the following clock cycle to enable that stage to perform the required write operation.
- Buffer B3 holds the results produced by the execution unit and the destination information for instruction I1.

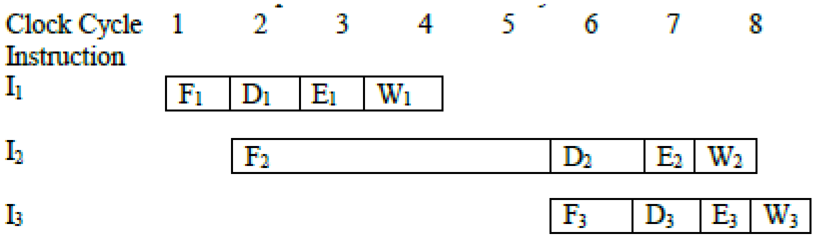
11. Write short notes on Pipeline performance

Pipeline performance

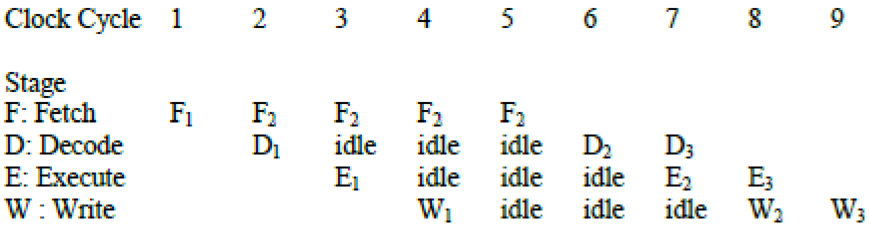
- Pipelining is proportional to the number of pipeline stages.
- For variety of reasons, one of the pipeline stages may not be able to complete its processing task for a given instruction in the time allotted.
- For eg, stage E in the four-stage pipeline is responsible for arithmetic and logic operations and one clock cycle is assigned for this task.
- Although this may be sufficient for most operations some operations such as divide may require more time to complete.
- Instruction I2 requires 3 cycles to complete from cycle 4 through cycle 6.
- Thus in cycles 5 and 6 the write stage must be told to do nothing, because it has no data to work with.
- Meanwhile, the information in buffer B2 must remain intact until the execute stage has completed its operation.
- This means that stage 2 and in turn stage1 are blocked from accepting new instructions because the information in B1 cannot be overwritten.
- Thus steps D4 and F5 must be postponded.
- The pipeline may also be stalled because of a delay in the availability of an instruction.

- For example, this may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory.
- Such hazards are often called control hazards or instruction hazards.

Instruction execution steps in successive clock cycles:



Function performed by each process stage in successive clock cycles



Clock Cycle	1	2	3	4	5	6	7	8	9
Stage									
F: Fetch	F_1	F_2	F_2	F_2	F_2				
D: Decode		D_1	idle	idle	idle	D_2	D_3		
E: Execute			E_1	idle	idle	idle	E_2	E_3	
W : Write				W_1	idle	idle	idle	W_2	W_3

- Instruction I1 is fetched from the cache in cycle1,and its execution proceeds normally.
- Thus in cycles 5 and 6, the write stage must be told to do nothing, because it has no data to work with.
- Meanwhile, the information in buffer B2 must remain intact until the execute stage has completed its operation.
- This means that stage 2 and in turn stage1 are blocked from accepting now instructions because the information in B1 cannot be overwritten.
- Thus steps D4 and F4 must be postponed.
- Pipelined operation is said to have been stacked for two clock cycles.
- Normal pipelined operation resumes in cycle 7.

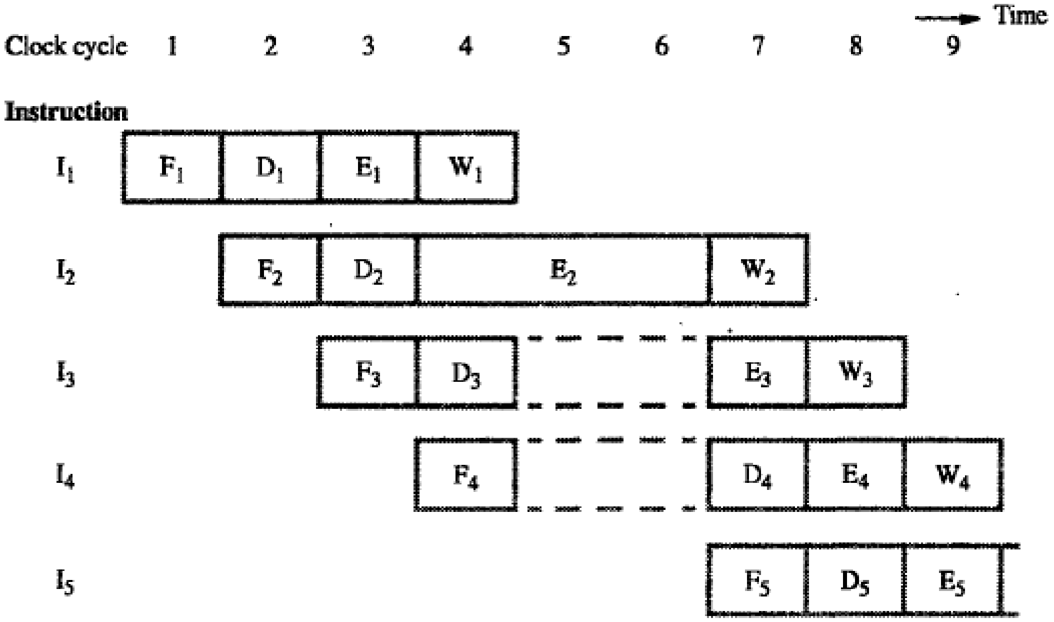
12. Define hazard. What are types of hazards?

Hazard

Any condition that causes the pipeline to stall is called a hazard. Consider the following example where execution takes more than a cycle. Here the pipelined operation is said to have been stalled for two clock cycles.

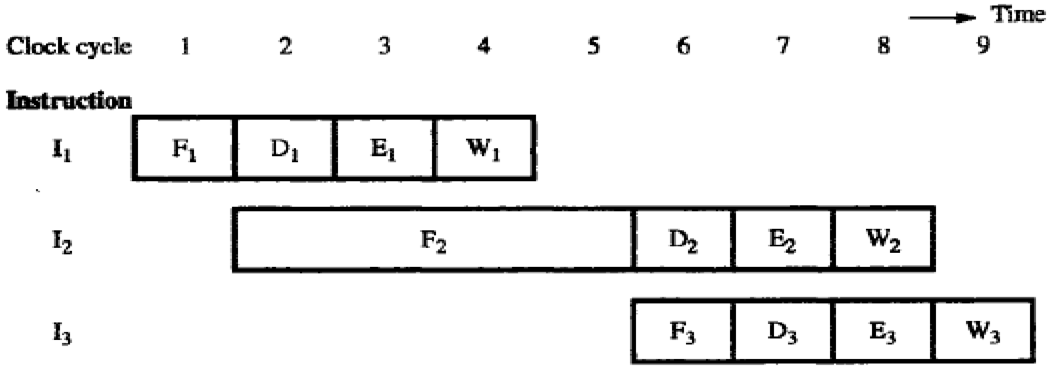
Data Hazard

A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result, some operation has to be delayed, and the pipeline stalls.



Control Hazards

The pipeline may also be stalled because of a delay in the availability of an instruction. For example, this may a result of a miss in the cache, requiring the instruction to be fetched from the main memory. Such hazards are often called control hazards or instruction hazards.



(a) Instruction execution steps in successive clock cycles

Clock cycle	1	2	3	4	5	6	7	8	9
Stage									
F: Fetch	F_1	F_2	F_2	F_2	F_2	F_3			
D: Decode		D_1	idle	idle	idle	D_2	D_3		
E: Execute			E_1	idle	idle	idle	E_2	E_3	
W: Write				W_1	idle	idle	idle	W_2	W_3

(b) Function performed by each processor stage in successive clock cycles

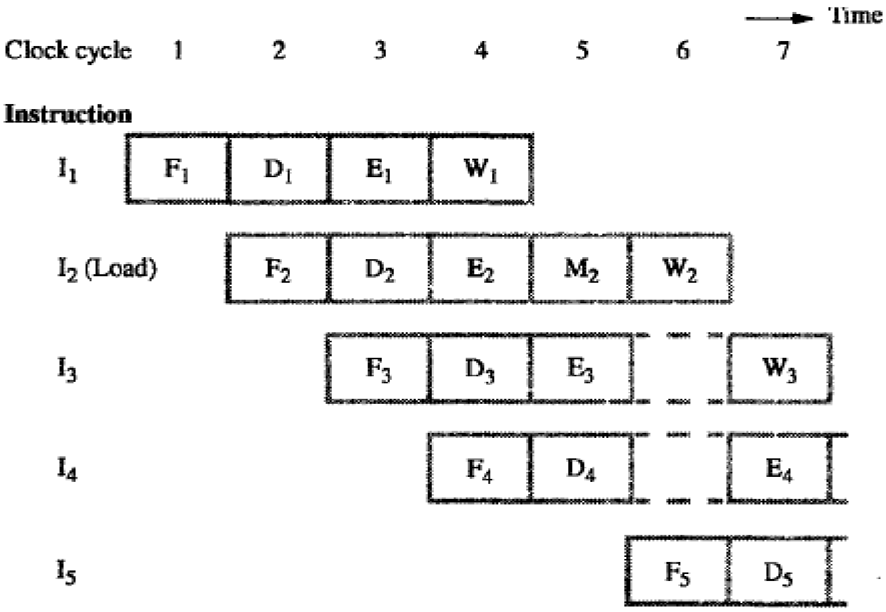
Pipeline stall caused by a cache miss in F2.

Structural Hazards

A third type of hazard that may be encountered in pipelined operation is known as a structural hazard. This is the situation when two instructions require the use of a given hardware resource at the same time. The most common case in which this hazard may arise is in access to memory. Many processors use separate instruction and data caches to avoid this delay.

Example of the structural hazard is shown below.

Load X(R1),R2



13. Explain Data Hazards in detail.

A data hazard is a situation in which the pipeline is stalled because the data to be operated on are delayed for some reason. Consider the program that contains to instructions I1 followed by I2. When this program is executed in a pipeline, the execution of I2 can begin before the execution of I1 is completed. This means that the results generated by I1 may not be available for use by I2.

- We must ensure that the results obtained when instructions are executed in a pipelined processor are identical to those obtained when the same instructions are executed sequentially.
- Hazard occurs

  $A \leftarrow 3 + A$

  $B \leftarrow 4 \times A$

- No hazard

  $A \leftarrow 5 \times C$

  $B \leftarrow 20 + C$

- When two operations depend on each other, they must be executed sequentially in the correct order.

- Another example:

  Mul R2, R3, R4

  Add R5, R4, R6

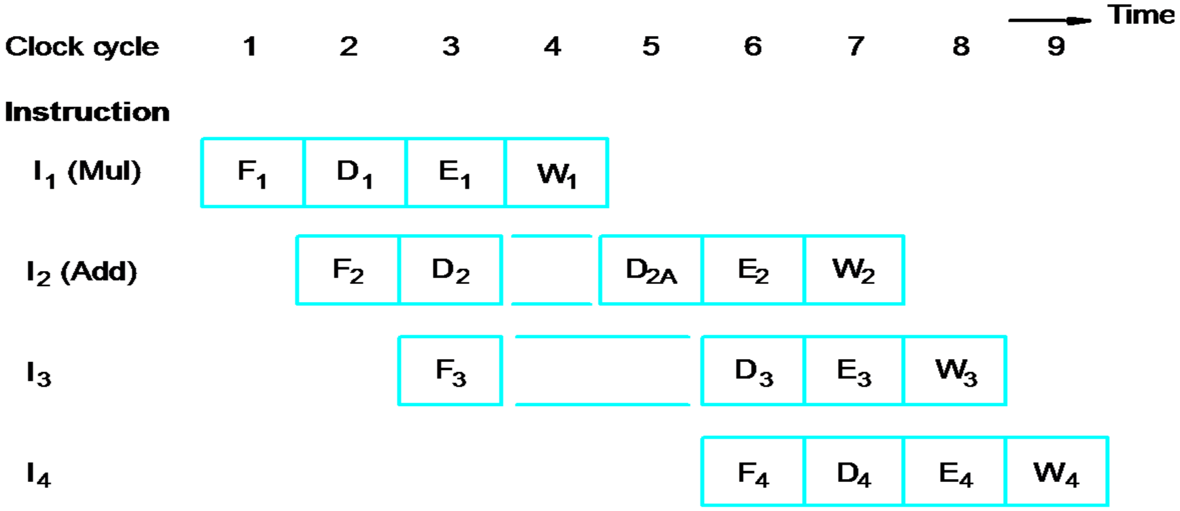


Fig. Pipeline stalled by data dependency between D2 and W1

Operand Forwarding

- Instead of from the register file, the second instruction can get data directly from the output of ALU after the previous instruction is completed.
- A special arrangement needs to be made to "forward" the output of ALU to the input of ALU.

Fig. Operand forwarding in a pipelined processor

Handling Data Hazards in Software

- Let the compiler detect and handle the hazard:

```
I1: Mul  R2, R3, R4
    NOP
    NOP
I2: Add  R5, R4, R6
```

- The compiler can reorder the instructions to perform some useful work during the NOP slots.

Side Effects

- The previous example is explicit and easily detected.
- Sometimes an instruction changes the contents of a register other than the one named as the destination.

- When a location other than one explicitly named in an instruction as a destination operand is affected, the instruction is said to have a side effect. (Example?)
- Example: conditional code flags:

  Add R1, R3

  AddWithCarry R2, R4

- Instructions designed for execution on pipelined hardware should have few side effects.

14. Explain Instruction Hazards in detail. (Apr 12)

The purpose of the instruction fetch unit is to supply the execution units with a steady stream of instructions. Whenever this stream is interrupted, the pipeline stalls, as cache miss. A branch instruction may also cause the pipeline to stall.

Unconditional Branches

Fig. An idle cycle caused by a branch instruction

Branch Timing

Fig. Branch timing

Instruction Queue and Prefetching

Either a cache miss or a branch instruction stalls the pipeline for one or more clock cycles. To reduce the effect of these interruptions, many processors employ sophisticated fetch units that can fetch instructions before they are needed and put them in a queue.

A separate unit, called the dispatch unit, takes instructions from the front of the queue and sends them to the execution unit.

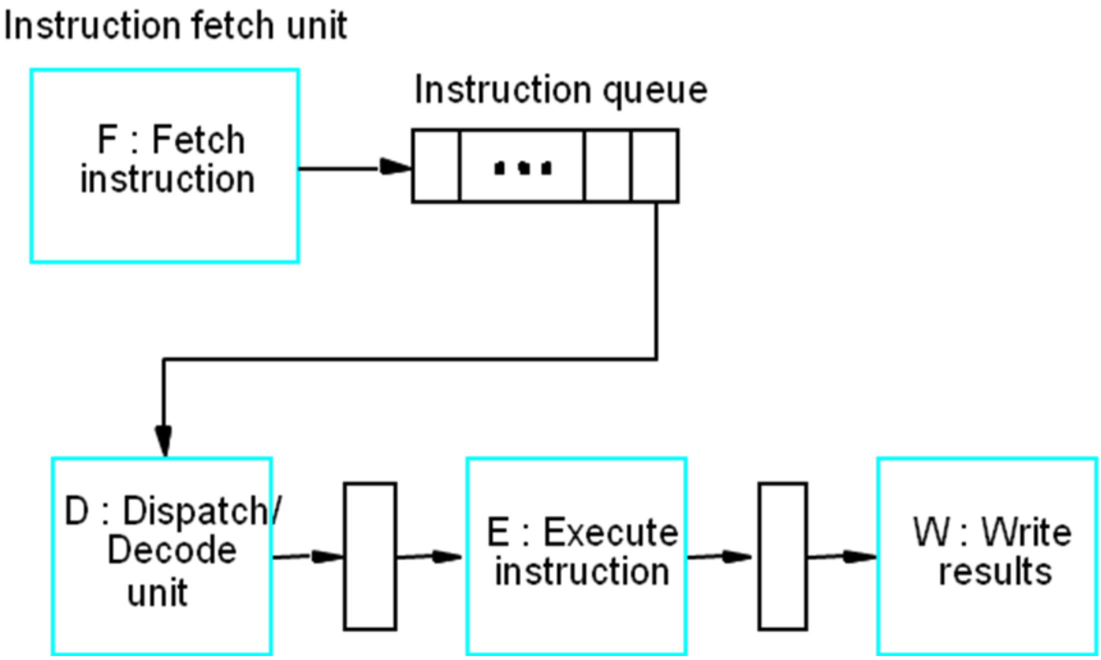


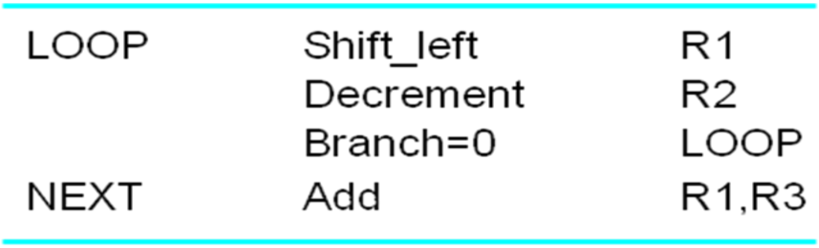
Fig. Use of instruction queue in hardware organization

Conditional Braches

- A conditional branch instruction introduces the added hazard caused by the dependency of the branch condition on the result of a preceding instruction.
- The decision to branch cannot be made until the execution of that instruction has been completed.
- Branch instructions represent about 20% of the dynamic instruction count of most programs.

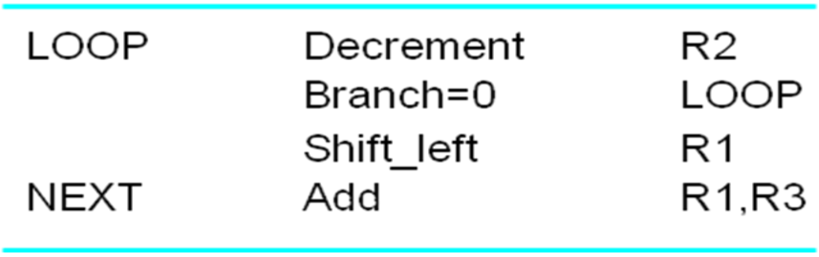
Delayed Branch

- The instructions in the delay slots are always fetched. Therefore, we would like to arrange for them to be fully executed whether or not the branch is taken.
- The objective is to place useful instructions in these slots.
- The effectiveness of the delayed branch approach depends on how often it is possible to reorder instructions.



```
LOOP    Shift_left    R1
        Decrement     R2
        Branch=0      LOOP
NEXT    Add           R1,R3
```

(a) Original program loop



```
LOOP    Decrement     R2
        Branch=0      LOOP
        Shift_left    R1
NEXT    Add           R1,R3
```

Fig. Reordering of instructions for a delayed branch

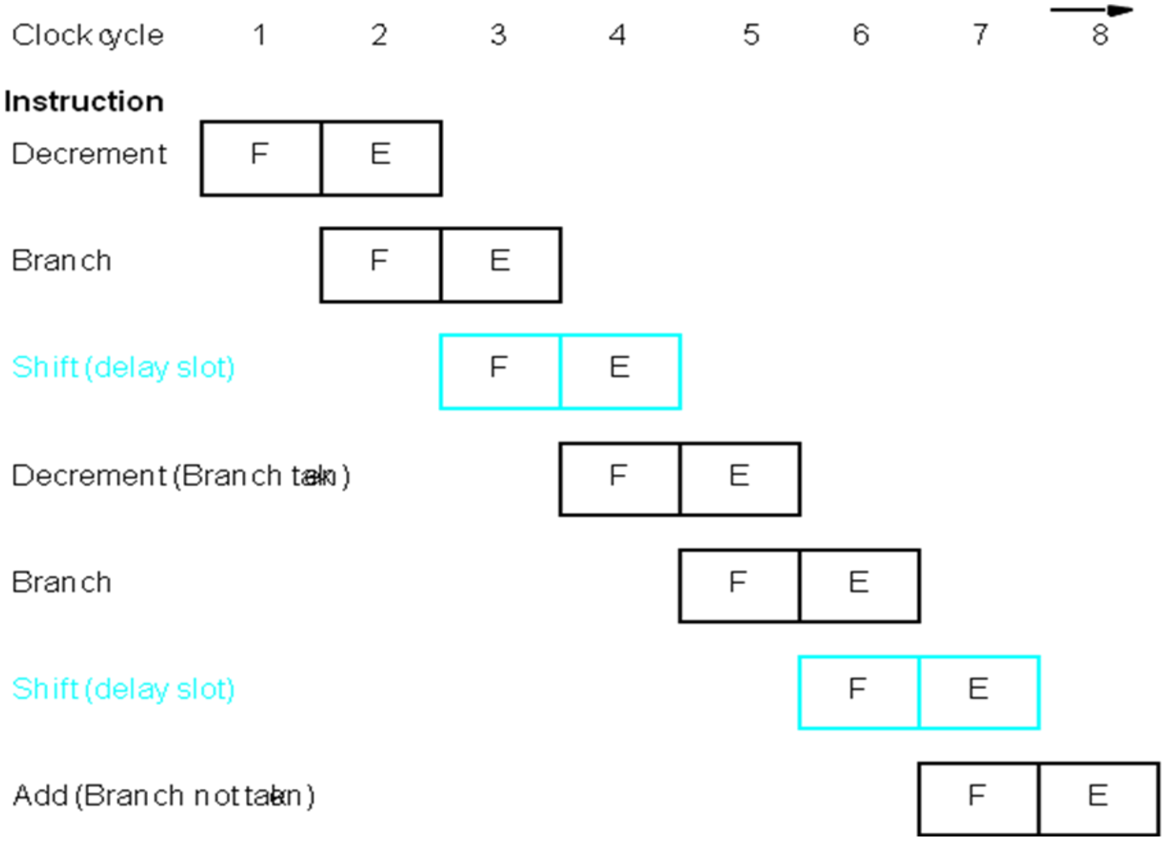


Fig. Execution timing showing the delay slot being filled during the last two passes through the loop

Branch Prediction

Another technique for reducing the branch penalty associated with conditional branches is to attempt to predict whether or not a particular branch will be taken. Until the branch condition is evaluated, instruction execution along the predicted path must be done on a speculative basis.

- Speculative execution means that instructions are executed before the processor is certain that they are in the correct execution sequence.
- Need to be careful so that no processor registers or memory locations are updated until it is confirmed that these instructions should indeed be executed.

Incorrectly Predicted Branch

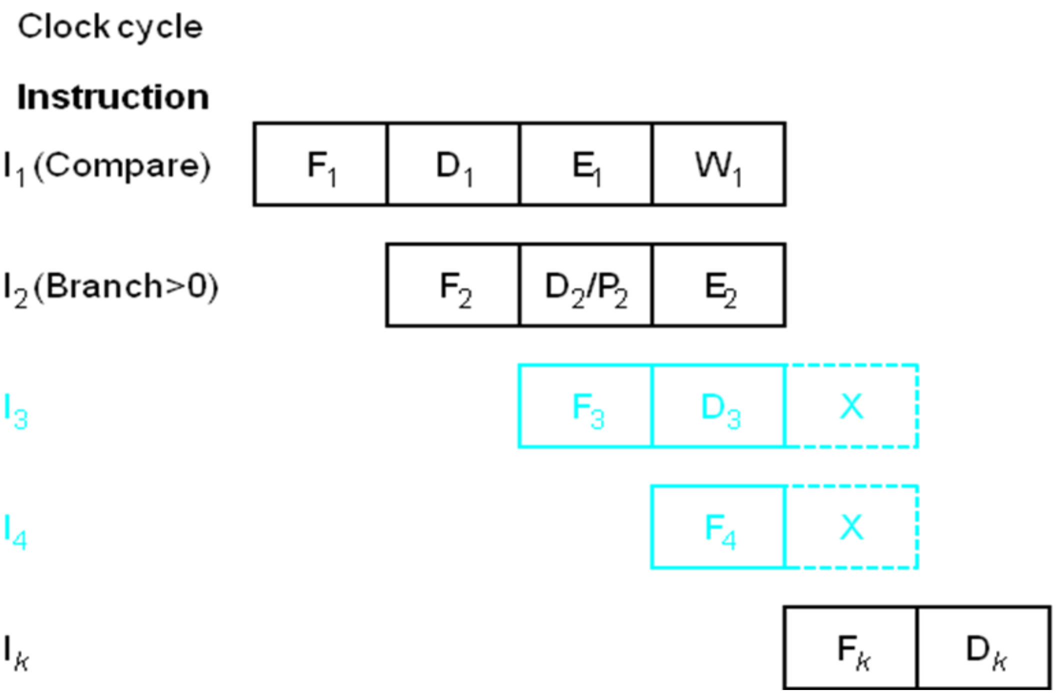


Fig. Timing when a branch decision has been incorrectly predicted as not taken.

- Better performance can be achieved if we arrange for some branch instructions to be predicted as taken and others as not taken.
- Use hardware to observe whether the target address is lower or higher than that of the branch instruction.
- Let compiler include a branch prediction bit.
- So far the branch prediction decision is always the same every time a given instruction is executed – static branch prediction.

15. Write in detail about the influence of Pipelining on Instruction Sets

Influence on Instruction Sets

Some instructions are much better suited to pipeline execution than others. Two key aspects of machine instructions are;

- Addressing modes

- Conditional code flags

Addressing Modes

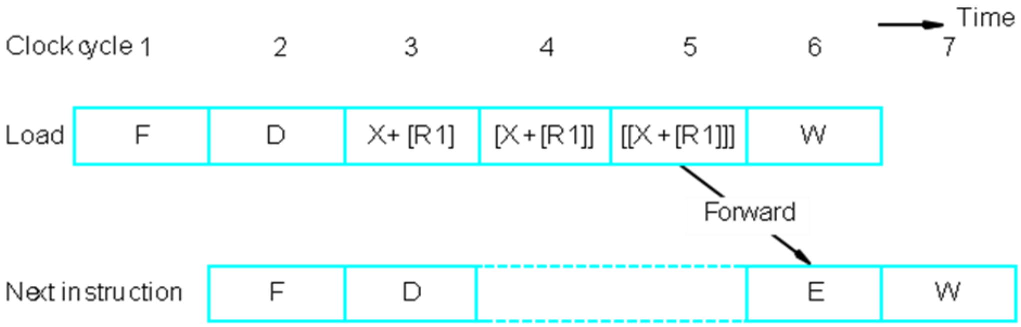
Addressing modes should provide the means for accessing a variety of data structures simply and efficiently. Useful addressing modes include index, indirect, auto-increment, and auto-decrement. In choosing the addressing modes to be implemented in a pipelined processor, we must consider the effect of each addressing mode on instruction flow in the pipeline. Two important considerations are the side effects of modes such as auto-increment and auto-decrement and the extent to which complex addressing modes cause the pipeline to stall. Another important factor is whether a given mode is likely to be used by compilers.

Load X(R1), R2

Fig. Effect of a load instruction on pipeline timing

Complex Addressing Mode

Load (X(R1)), R2



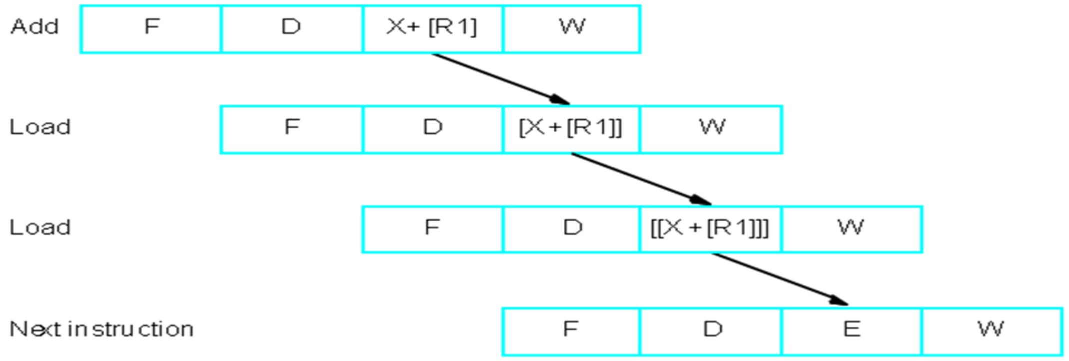
(a) Complex addressing mode

Simple Addressing Mode

Add #X, R1, R2

Load (R2), R2

Load (R2), R2



(b) Simple addressing mode

- In a pipelined processor, complex addressing modes do not necessarily lead to faster execution.

Advantage: reducing the number of instructions / program space

Disadvantage: cause pipeline to stall / more hardware to decode / not convenient for compiler to work with

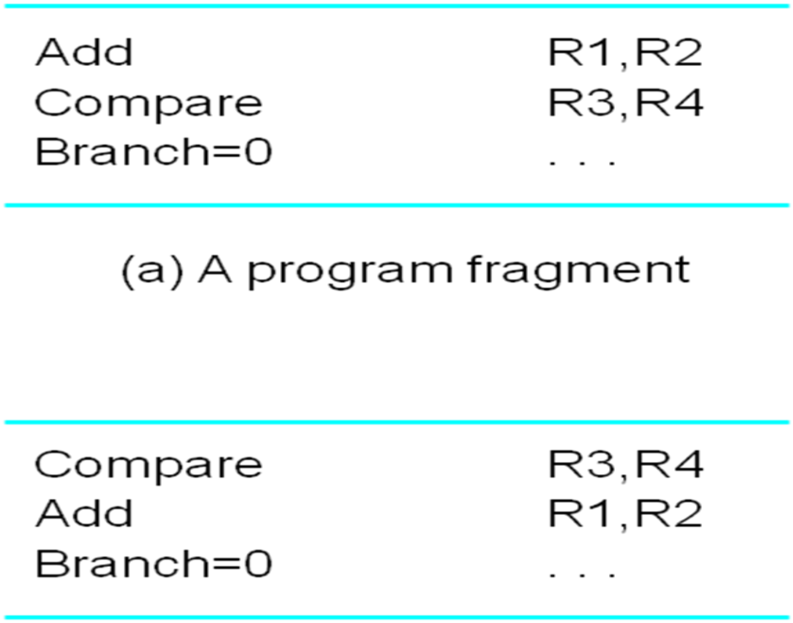
Conclusion: complex addressing modes are not suitable for pipelined execution.

Good addressing modes should have:

- Access to an operand does not require more than one access to the memory
- Only load and store instruction access memory operands

16. What are condition codes? Explain.

In many processors, the condition code flags are stored in the processor status register. They are either set or cleared by many instructions, so that they can be tested by subsequent conditional branch instructions to change the flow of program execution. An optimizing compiler for a pipelined processor attempts to reorder instructions to avoid stalling the pipeline when branches or data dependencies between successive instructions occur. In doing so, the compiler must ensure that reordering does not cause a change in the outcome of a computation. The dependency introduced by the condition-code flags reduces the flexibility available for the compiler to reorder instructions.



```
Add            R1,R2
Compare        R3,R4
Branch=0       . . .
```

(a) A program fragment

```
Compare        R3,R4
Add            R1,R2
Branch=0       . . .
```

(b) Instructions reordered

- Two conclusion:

➢ To provide flexibility in reordering instructions, the condition-code flags should be affected by as few instruction as possible.

➢ The compiler should be able to specify in which instructions of a program the condition codes are affected and in which they are not.

17. What are the considerations needed for using pipelined design?

Original design

Fig. Three bus organization of the data path

Pipelined Design

Separate instruction and data caches

- PC is connected to IMAR

- DMAR

- Separate MDR

- Buffers for ALU

- Instruction queue

- Instruction decoder output

- Reading an instruction from the instruction cache

- Incrementing the PC

- Decoding an instruction

- Reading from or writing into the data cache

- Reading the contents of up to two regs
- Writing into one register in the reg file
- Performing an ALU operation

Fig. Data path modified for pipelined execution with inter-stage buffers at input and output of ALU

18. Explain in detail about Superscalar operation.

Superscalar Operations

Pipelining makes instructions to execute concurrently. Several instructions are present in the pipeline at the same time, but they are in different stages of their execution. While one instruction is performing an ALU operation, another instruction is being decoded and yet another is being fetched from the memory. In the absence of hazards, one instruction enters the pipeline and one instruction completes execution in each clock cycle. This means that the maximum throughput of a pipelined processor is one instruction per clock cycle.

A more aggressive approach is to equip the processor with multiple processing units to handle several instructions in parallel in each processing stage. With this arrangement, several instructions start execution in the same clock cycle, and the processor is said to use multiple-issue. Such processors are capable of achieving an instruction execution throughput of more than one instruction per cycle. They are known as superscalar processors.

Fig. A processor with two execution units

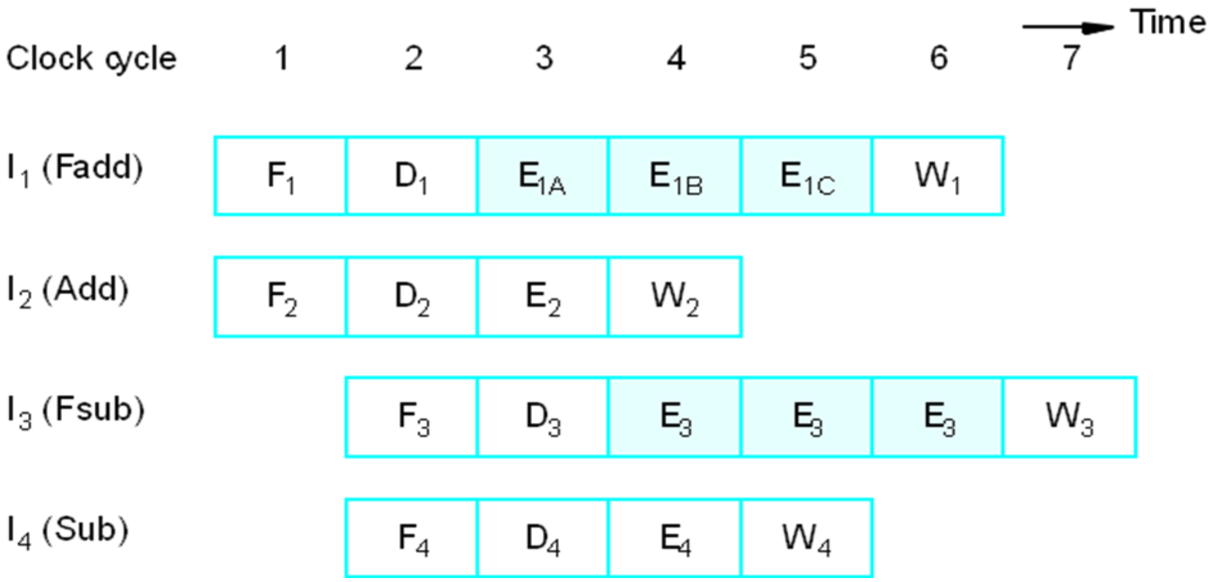
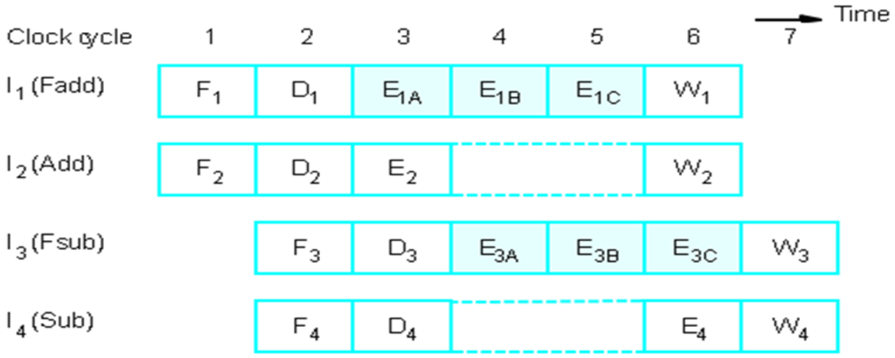


Fig.An example of instruction execution flow in the processor assuming no hazards are encountered.

Out Of Order Execution

Exceptions may be caused by a bus error during an operand fetch or by an illegal operation, such as an attempt to divide by zero. the program counter points to the instruction have been executed to completion. If such a situation is permitted, the processor is said to have imprecise exceptions.

If an exception occurs during an instruction, all subsequent instructions that may have been partially executed are discarded. This is called a precise exception.



(a) Delayed write

Execution Completion

- It is desirable to used out-of-order execution, so that an execution unit is freed to execute other instructions as soon as possible.
- At the same time, instructions must be completed in program order to allow precise exceptions.

The results are written into temporary registers. The content of these registers are later transferred to the permanent registers in correct program order. This step is often called the commitment step because the effect of the instruction cannot be reversed after that point. If an instruction causes an exception, the results of any subsequent instruction that has been executed would still be in temporary registers and can safely be discarded.

When out-of-order execution is allowed, a special control unit is needed to guarantee in-order commitment. This is called the commitment unit. It uses a queue called the reorder buffer to determine which instruction should be committed next.

An instruction is said to have retired only when it is at the head of the queue, all the instructions that were dispatched before it must also have been retired. Hence, instructions may complete execution out of order, but they are retired in program order.

19. List out the performance considerations using pipeline.

- The execution time T of a program that has a dynamic instruction count N is given by:

$$T = \frac{N \times S}{R}$$

where S is the average number of clock cycles it takes to fetch and execute one instruction, and R is the clock rate.

- Instruction throughput is defined as the number of instructions executed per second.

$$P_s = \frac{R}{S}$$

- An *n*-stage pipeline has the potential to increase the throughput by *n* times.
- However, the only real measure of performance is the total execution time of a program.
- Higher instruction throughput will not necessarily lead to higher performance.

Number of Pipeline Stages

- Since an *n*-stage pipeline has the potential to increase the throughput by *n* times, how about we use a 10,000-stage pipeline?
- As the number of stages increase, the probability of the pipeline being stalled increases.
- The inherent delay in the basic operations increases.
- Hardware considerations (area, power, complexity,)