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.

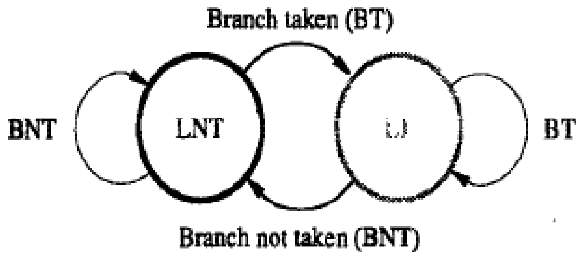
The branch prediction decision is always the same every time a given instruction is executed is called static branch prediction. The approach in which the branch prediction decision may change depending on execution history is called dynamic branch prediction.

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.

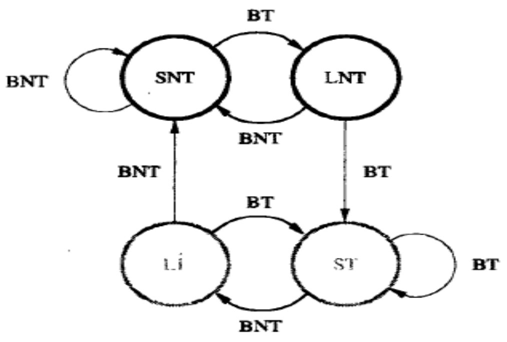
The four states are:

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.

<u>**11 MARKS**</u>

1. Explain about Single bus organization of the Data path inside a processor:

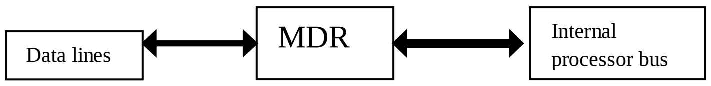
To execute an instruction, the processor has to perform the following steps:

- 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.

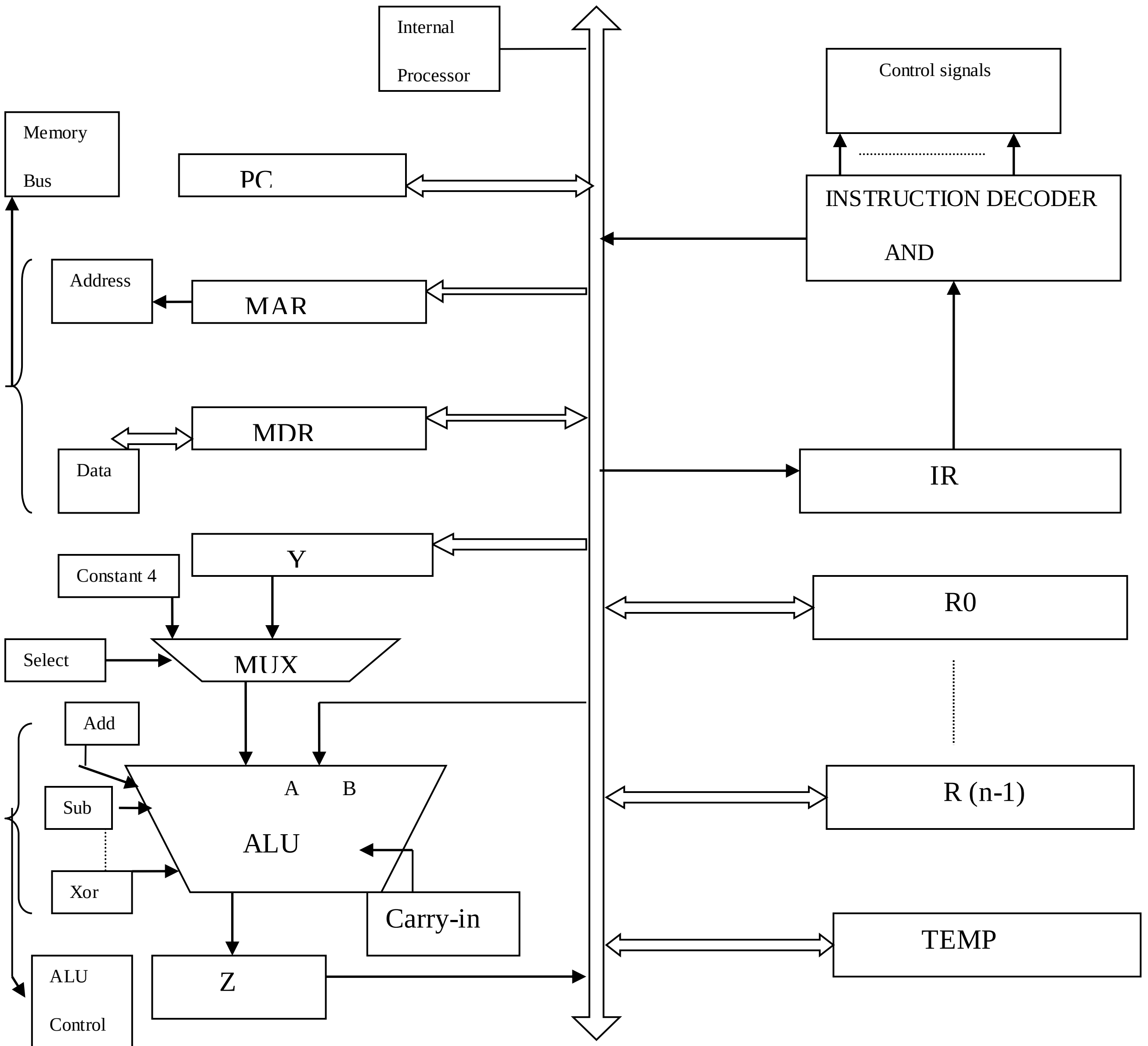
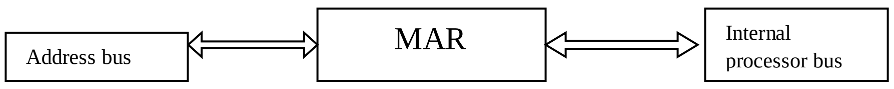


Fig: Single-Bus Organization Of The Data Path Inside A Processor

- 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.

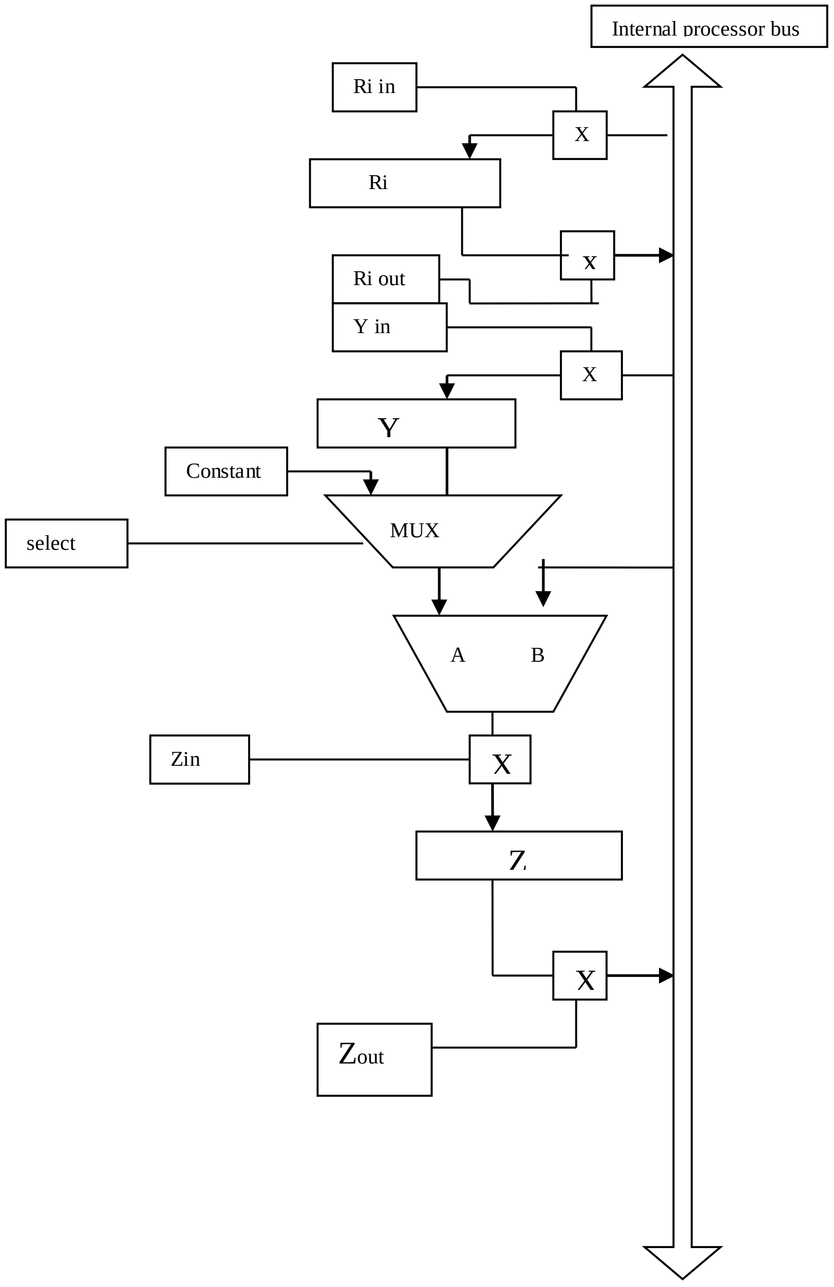


Fig: Register Transfer

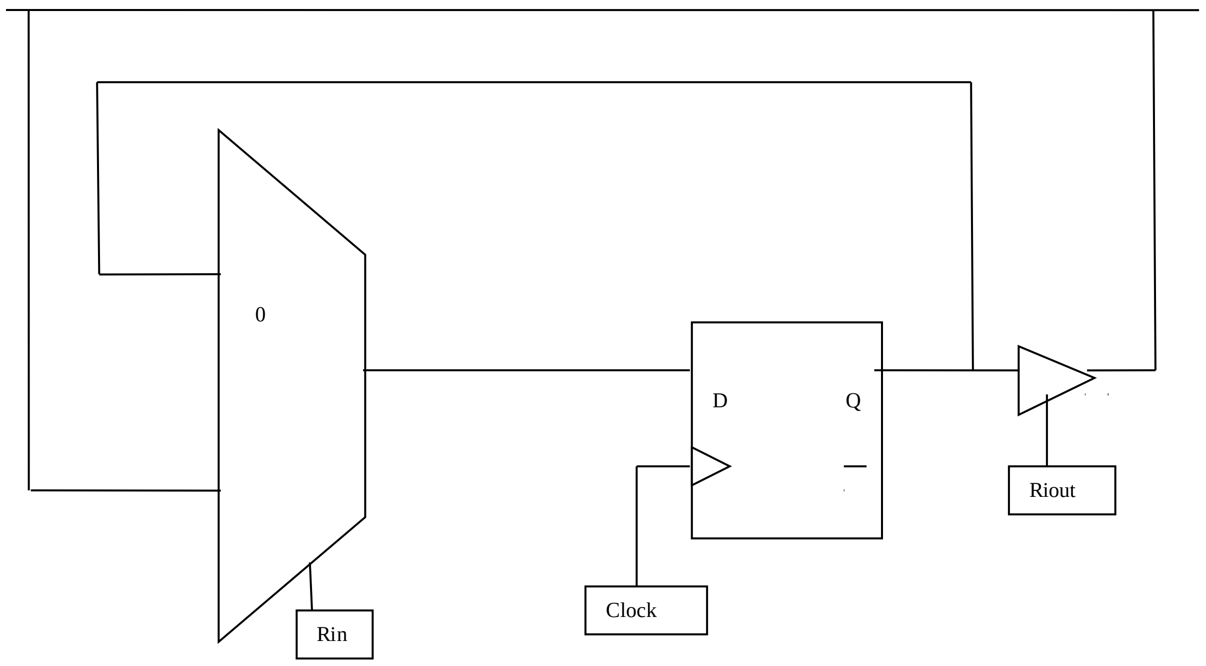


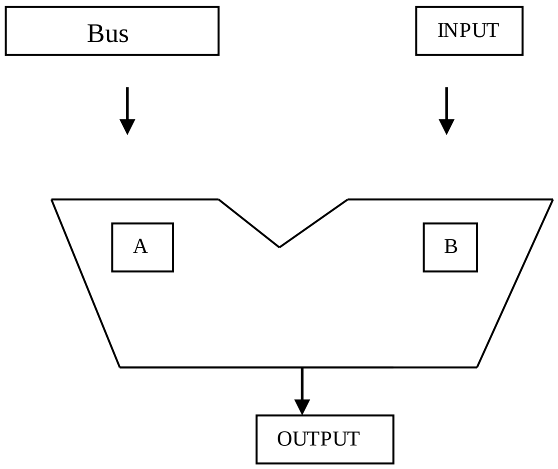
Fig: Input And Output Gating For One Register Bit

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.

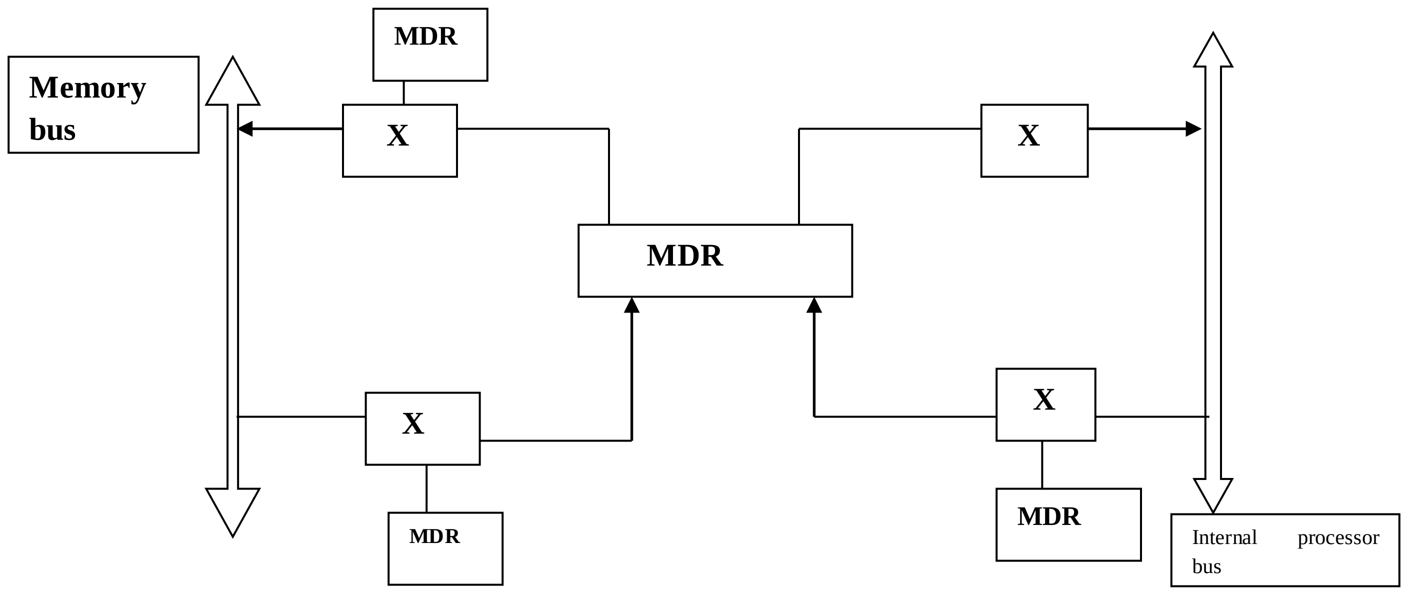


Fig: Connection And Control Signals For Register MDR

✓ 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

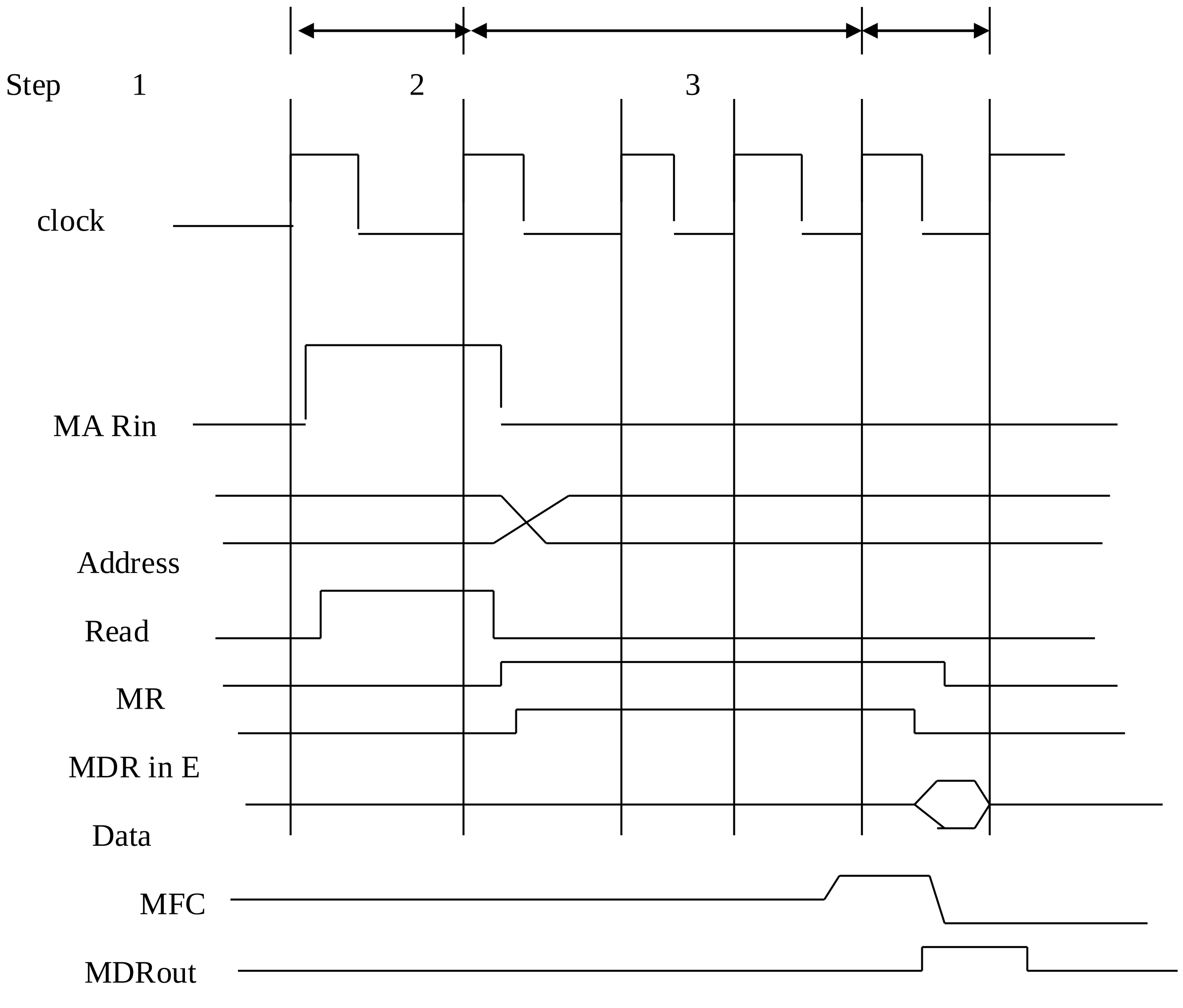


Fig: Timing Of A Memory Read Operation

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.

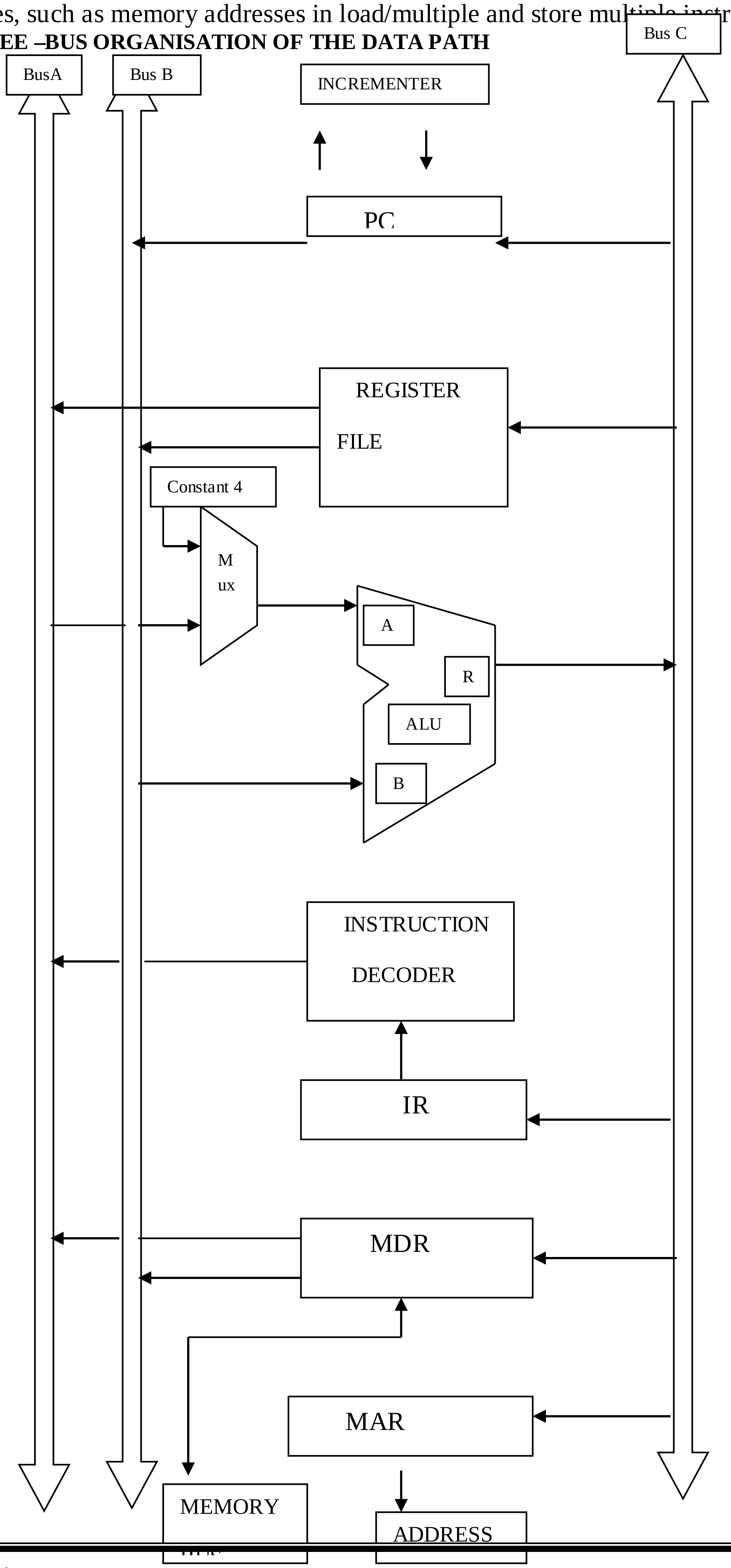
FEATURES:

➢ 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

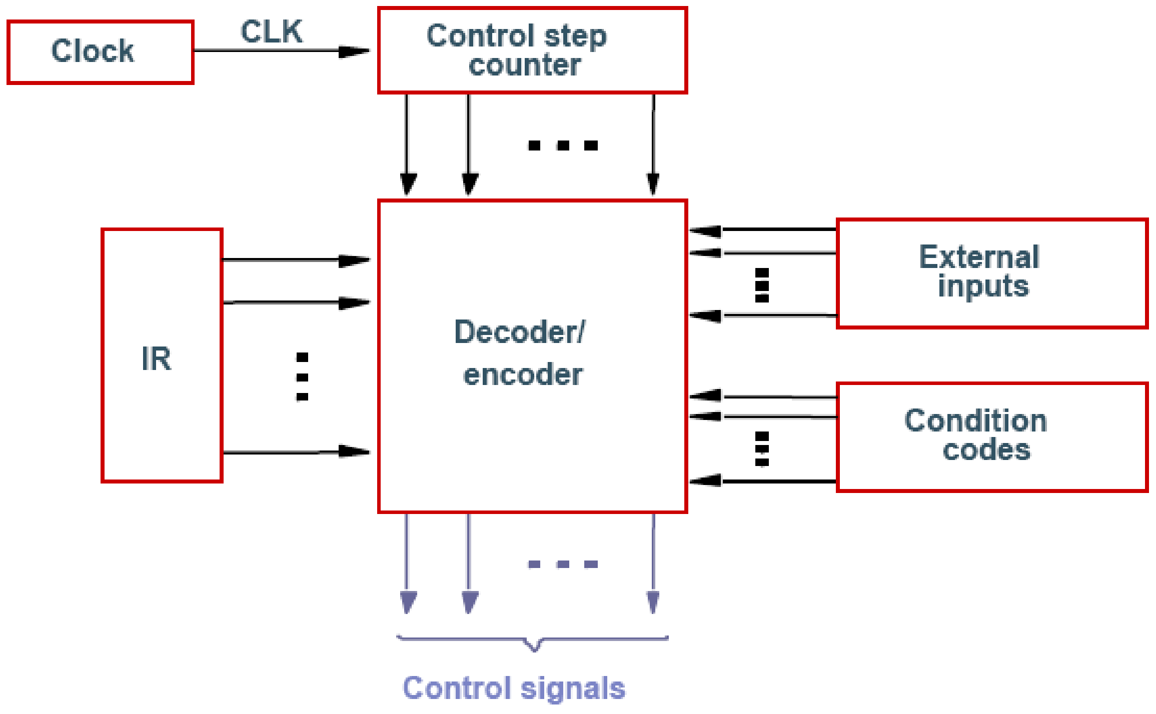


Fig. Control Unit Organization

- 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.

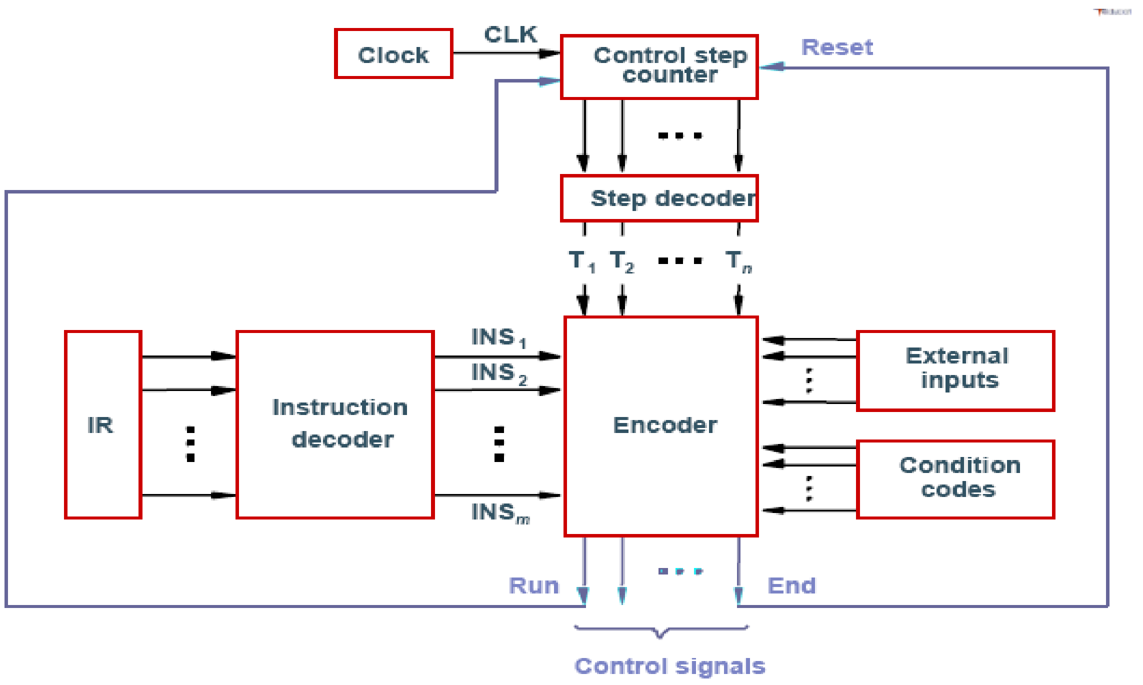
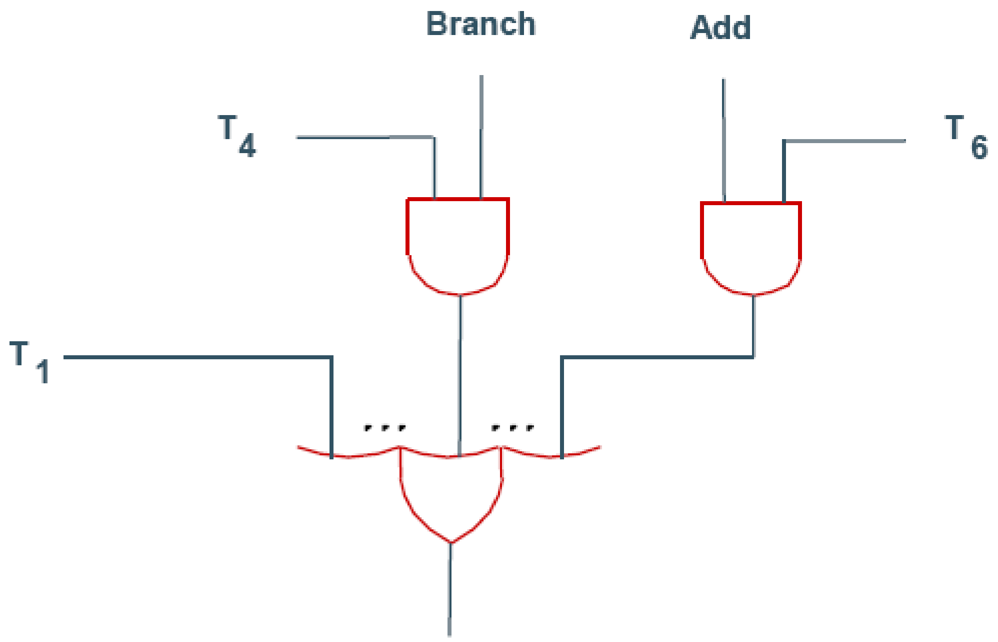


Fig. Separation of decoding and encoding functions

Let us see how the encoder generates signal for single bus processor organization shown in figure



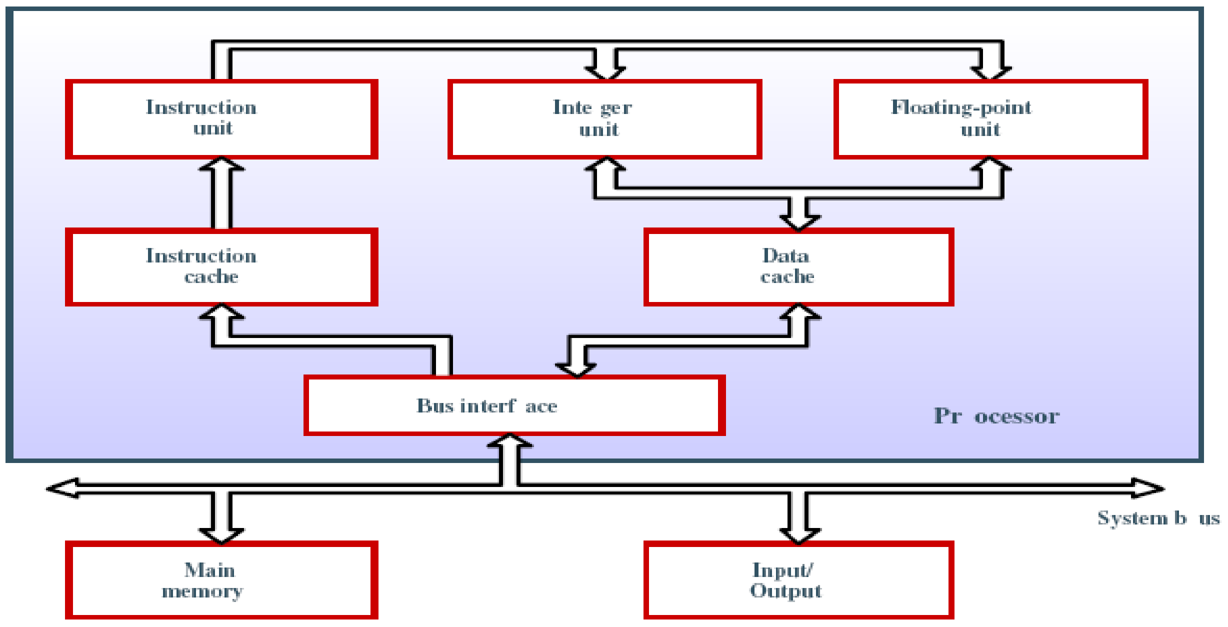
- Z_{in},The encoder circuit implements the following logic functions to generate Z_{in}.The logic function for Z_{in} is derived from the control sequence in figure.

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	WMFC	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.

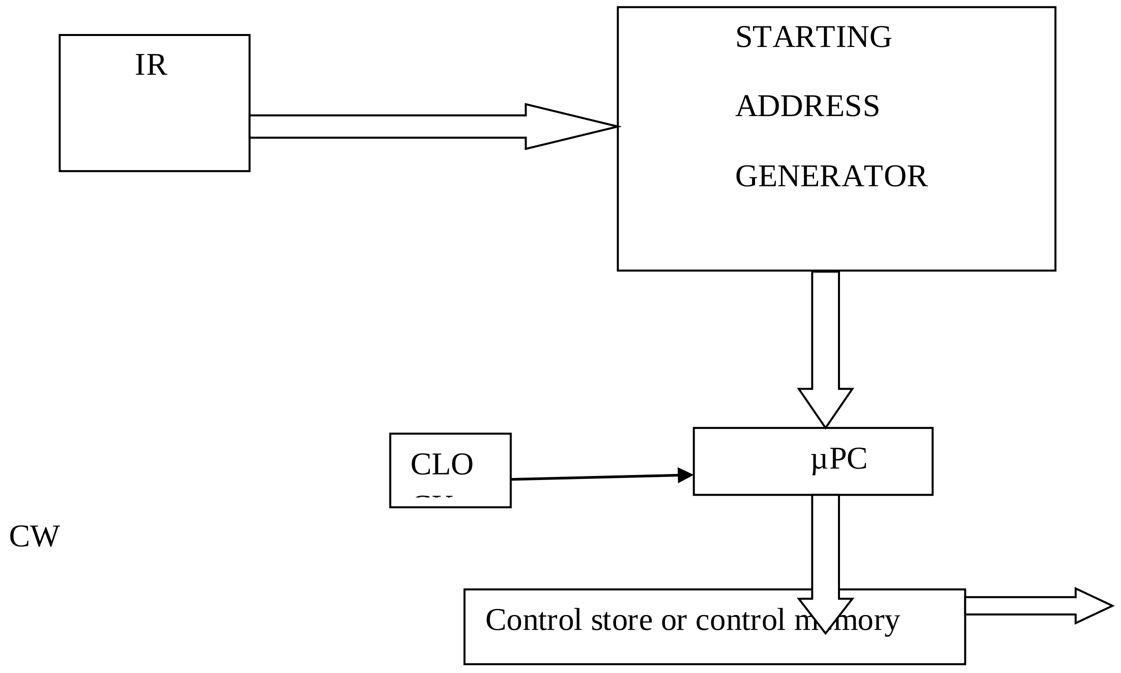


Fig: Basic Organization Of A Micro programmed Control Unit

- 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

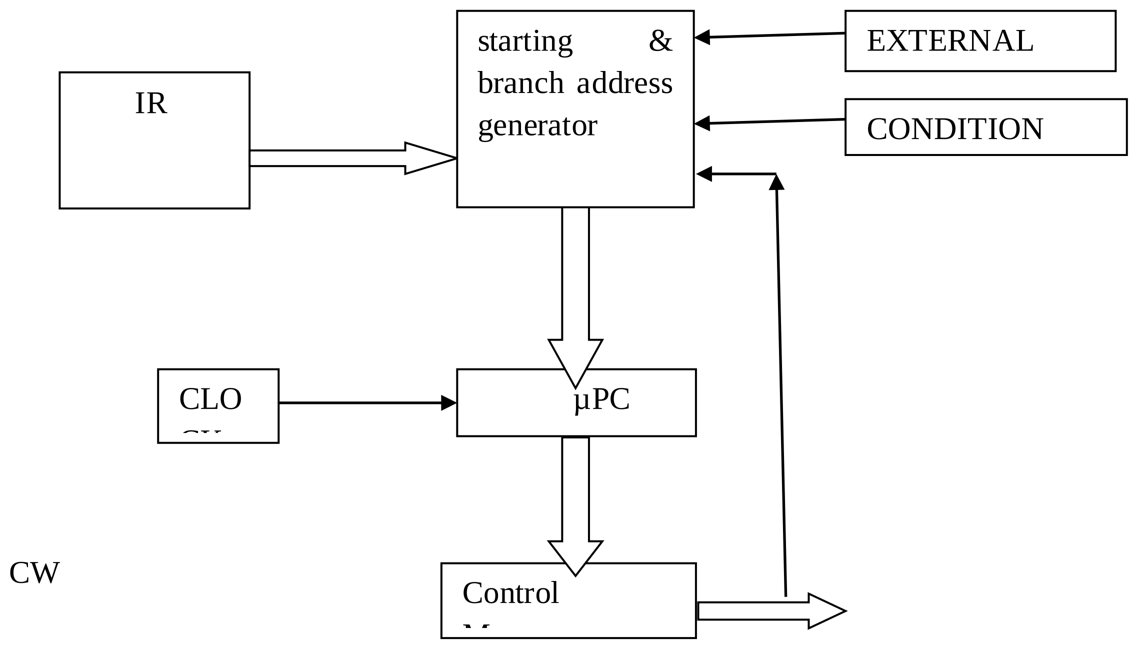


Fig:A Organization Of Control Unit To Allow Conditional Branching In The Micro program

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

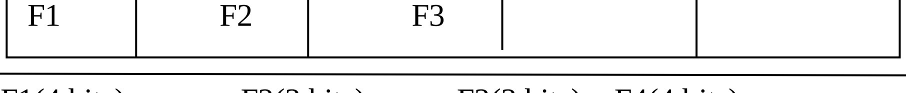
2) Many signals are mutually exclusive(e.g) only one function of ALU can be activated at a time.

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(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.

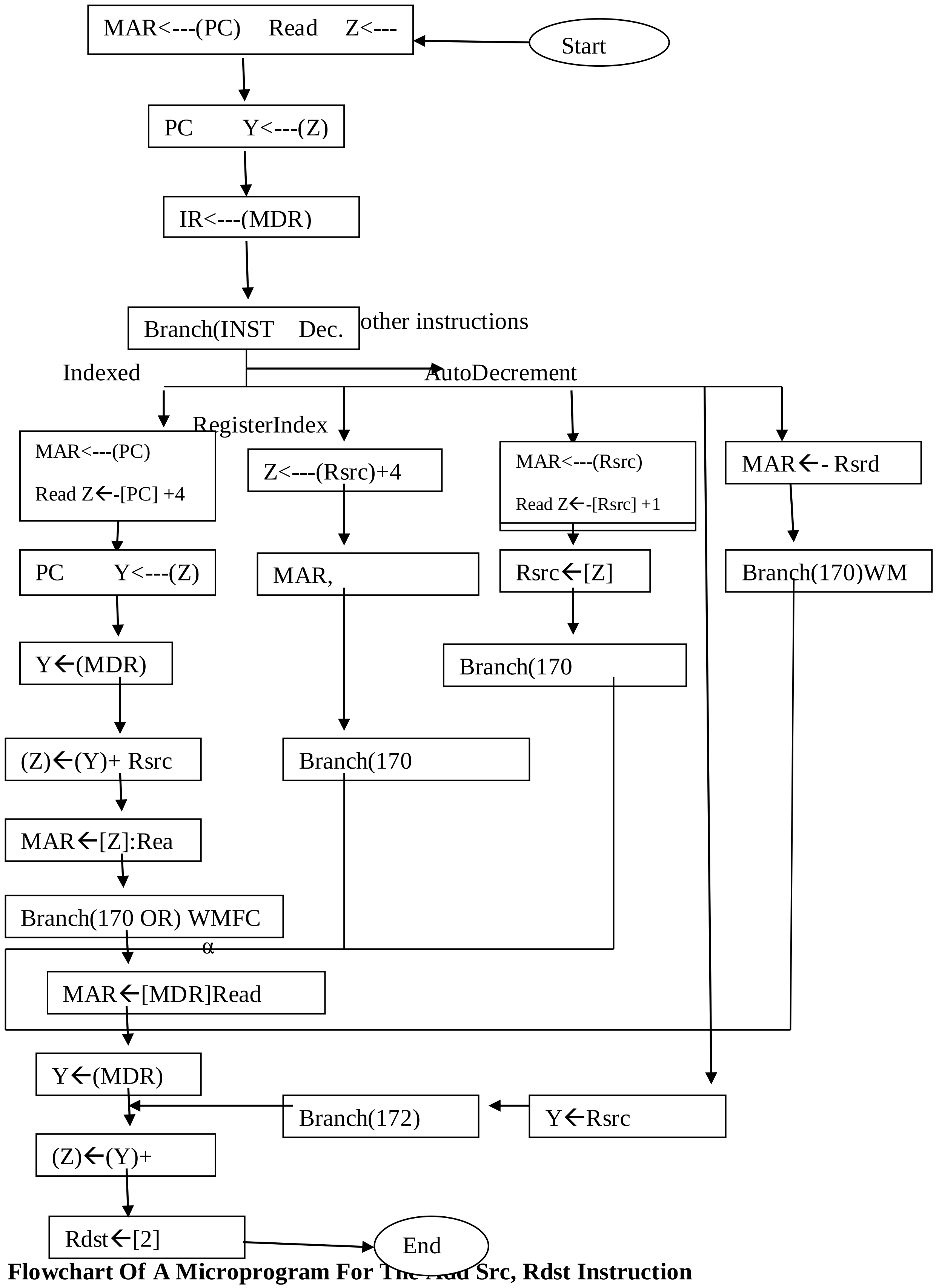


Fig: Flowchart Of A Microprogram For The Add Src, Rdst Instruction

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.

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.

- 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.

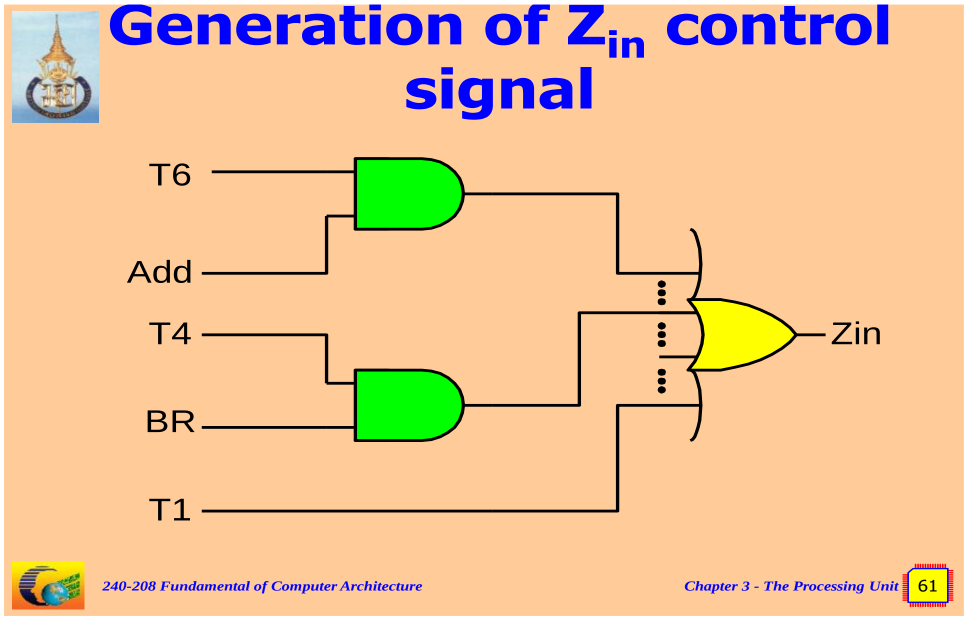
Techniques of grouping of control signals:

The grouping of control signals can be done either by using technique called vertical organisation or by using technique called horizontal organisation.

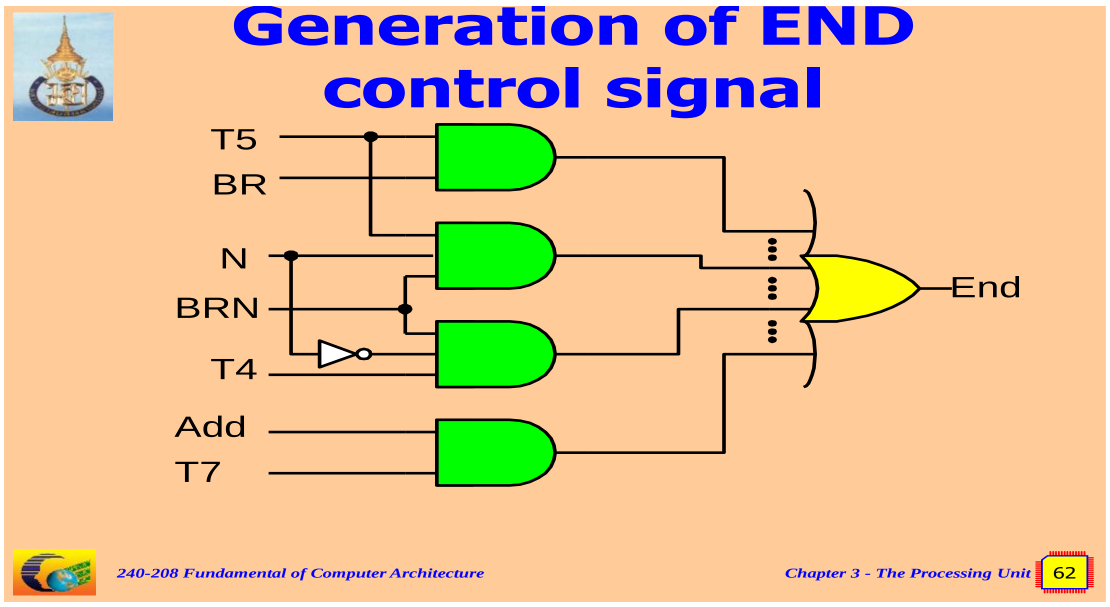
- Highly encoded scheme that use impact codes to specify only a small number of control function in each microinstruction are referred to as a vertical organization.
- On the other hand, minimally encoded scheme, in which resources can be controlled with a single instruction, is called a horizontal organization.

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
T6
Add
T4
BR
T1
Zin
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
T5
BR
N
BRN
T4
Add
T7
End
240-208 Fundamental of Computer Architecture
Chapter 3 - The Processing Unit
62

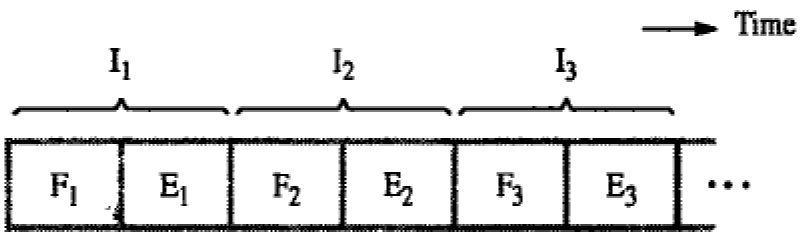
10. Explain the concepts of Pipelining.

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.

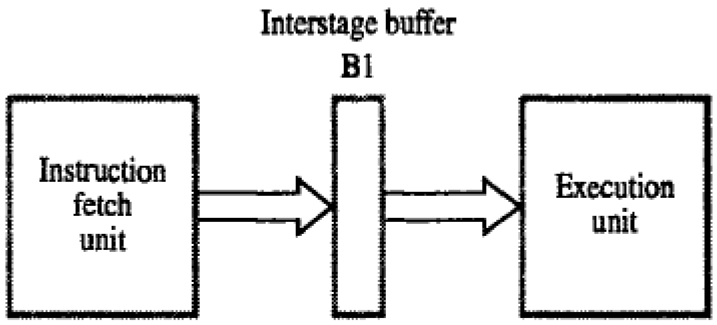


Hardware organization

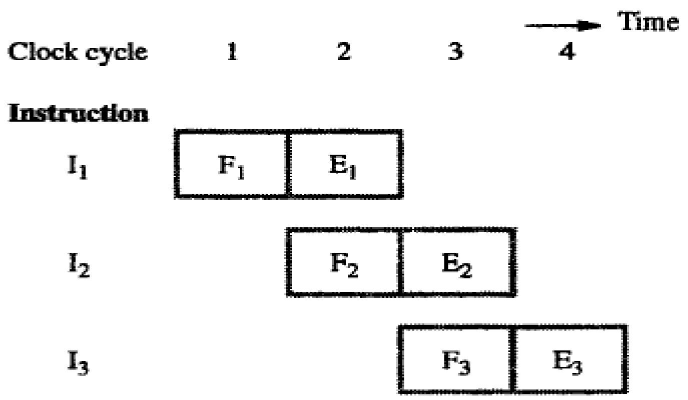
An inter-stage storage buffer, B1, is needed to hold the information being passed from one stage to the next.

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.

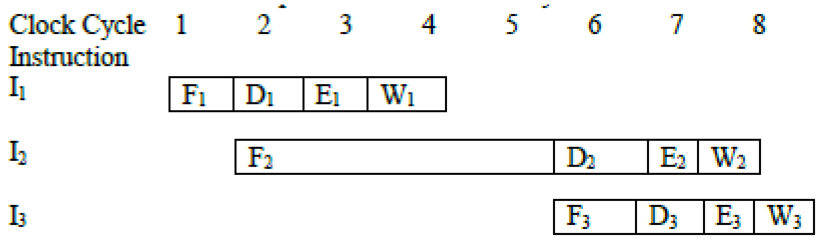
11. Write short notes on Pipeline performance

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.

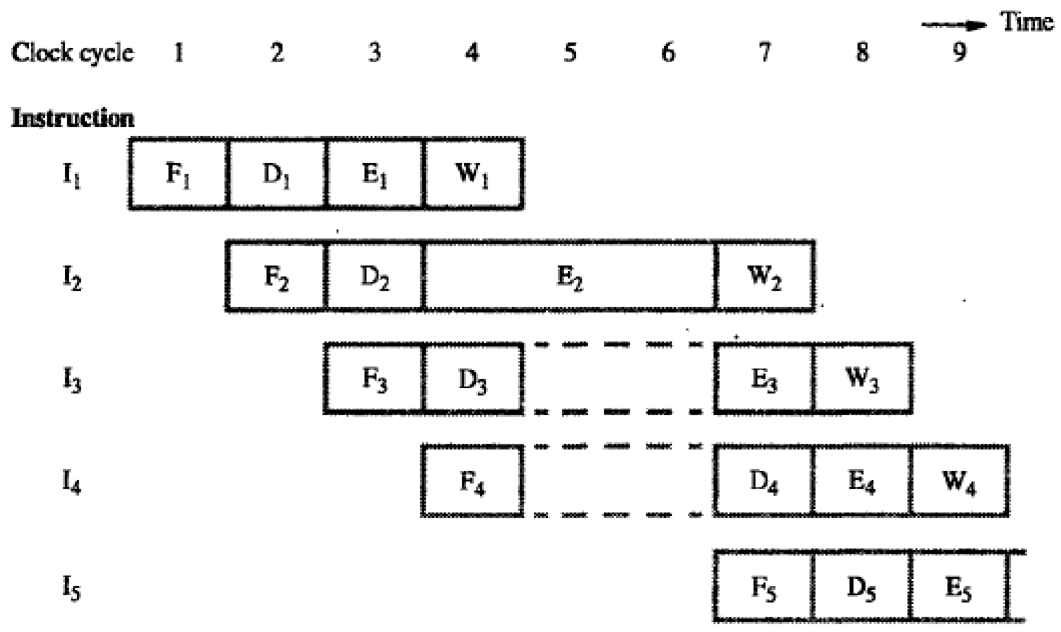
12. Define hazard. What are types of hazards?

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.

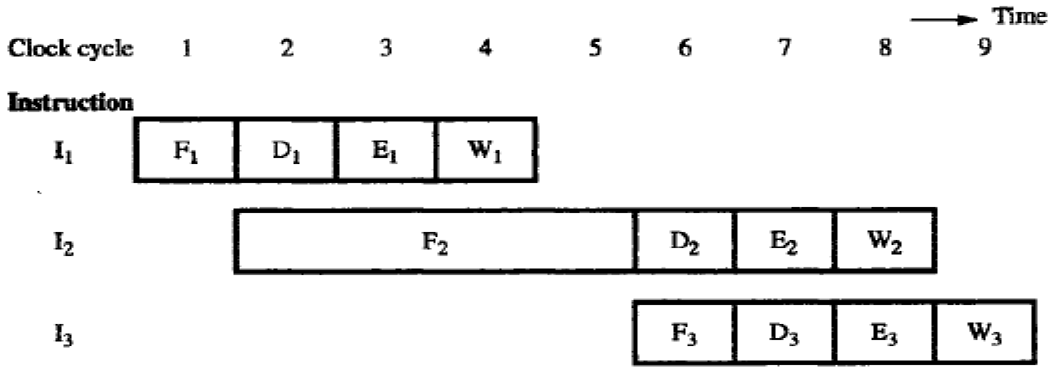
Data Hazard

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.

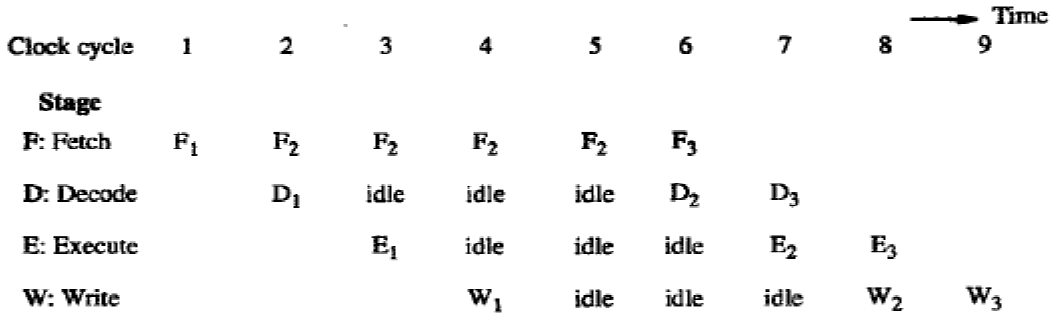


Control Hazards

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

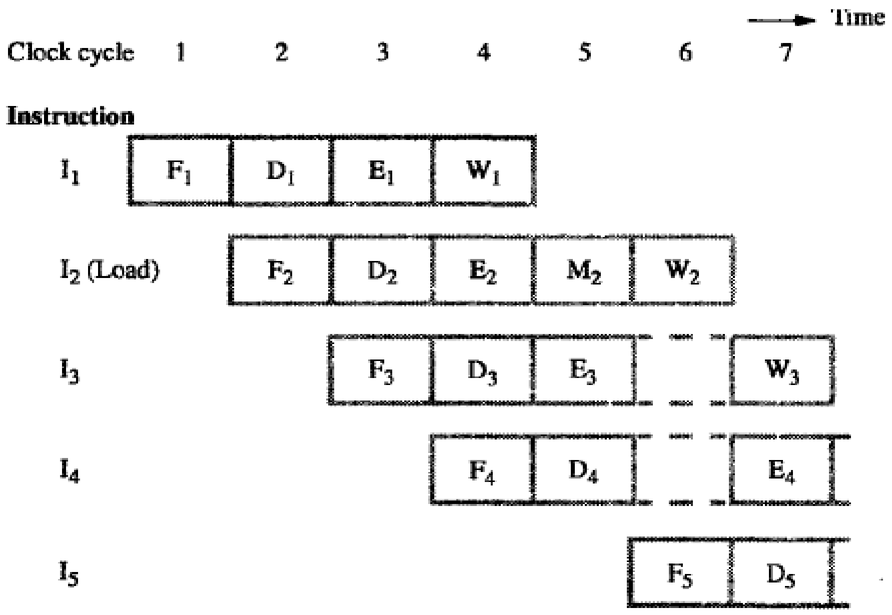
Pipeline stall caused by a cache miss in F2.

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

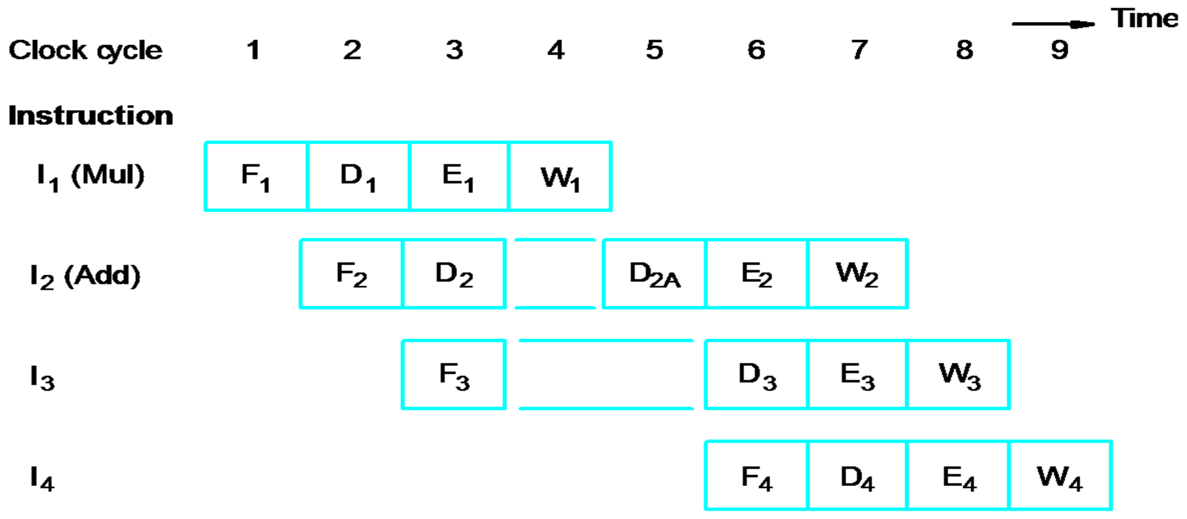


Fig. Pipeline stalled by data dependency between D2 and W1

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.

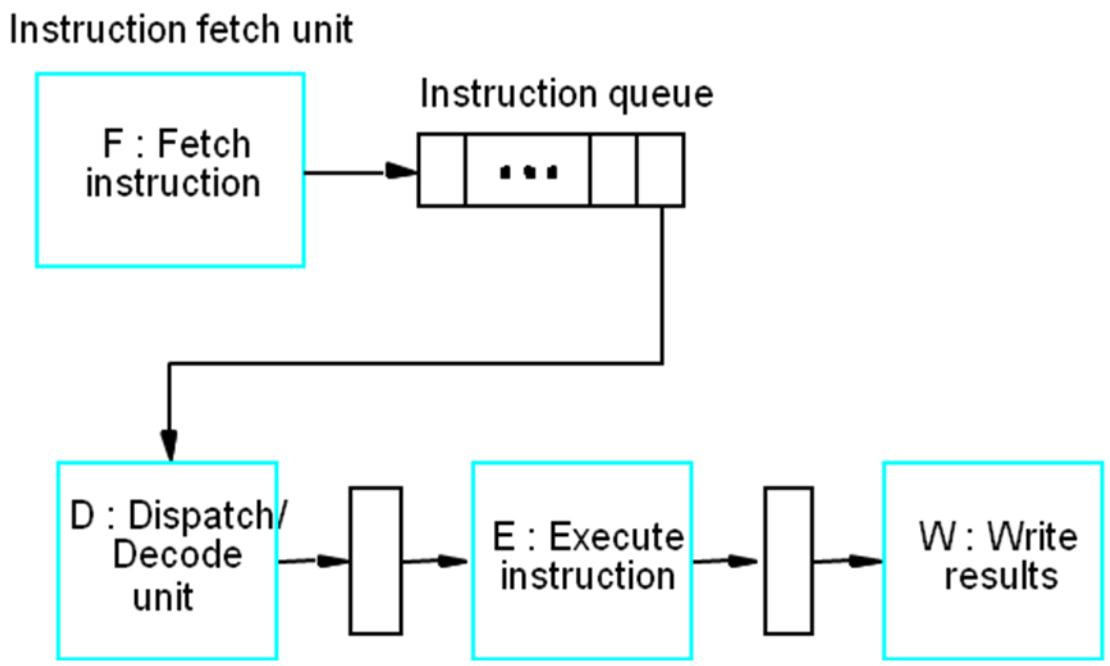


Fig. Use of instruction queue in hardware organization

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

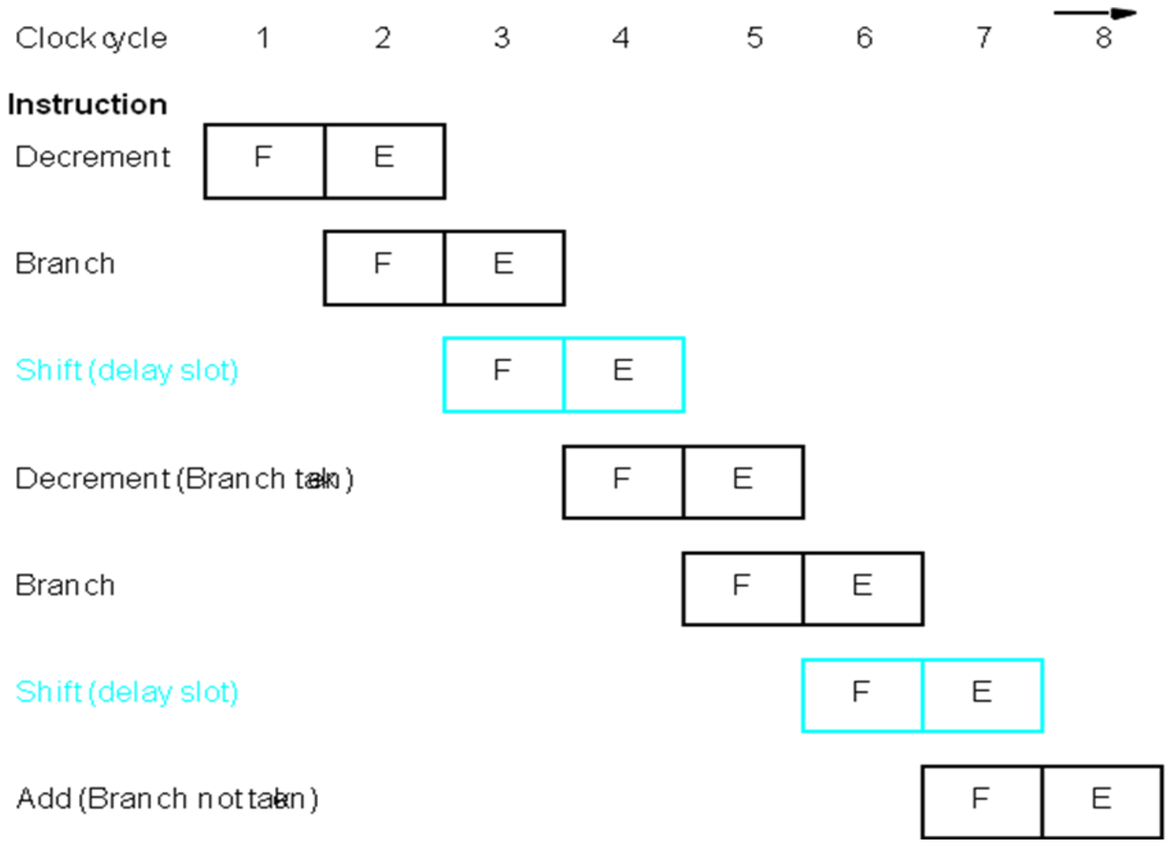


Fig. Execution timing showing the delay slot being filled during the last two passes through the loop

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

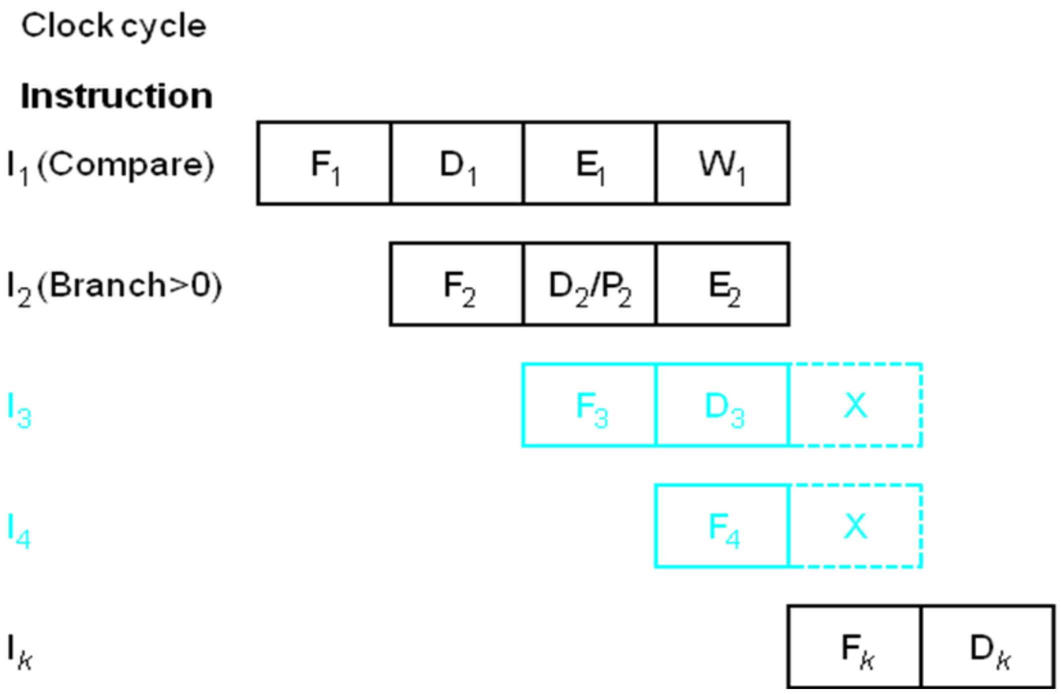


Fig. Timing when a branch decision has been incorrectly predicted as not taken.

- 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

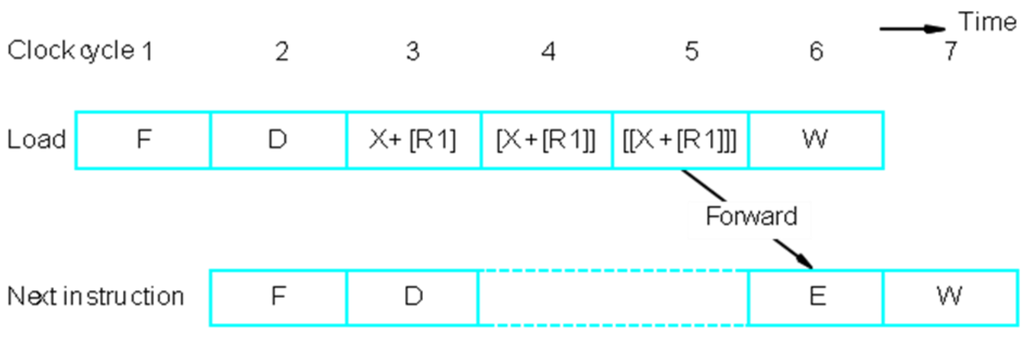
Addressing modes should provide the means for accessing a variety of data structures simply and efficiently. Useful addressing modes include index, indirect, auto-increment, and auto-decrement. In choosing the addressing modes to be implemented in a pipelined processor, we must consider the effect of 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



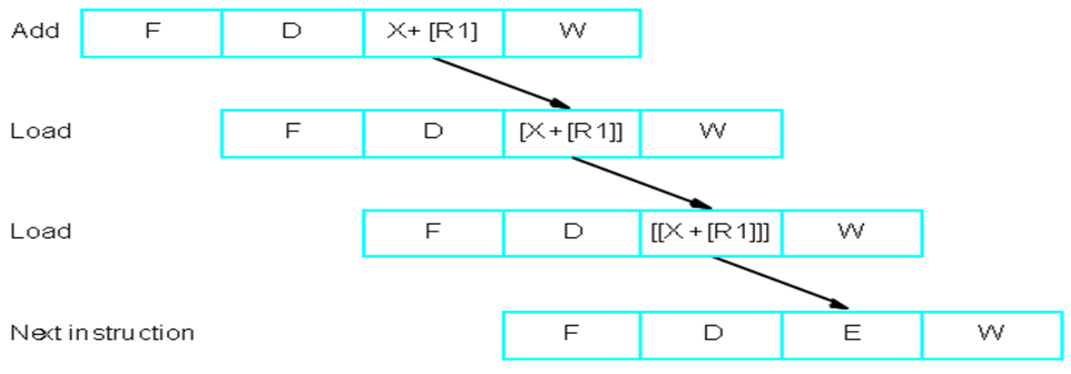
(a) Complex addressing mode

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

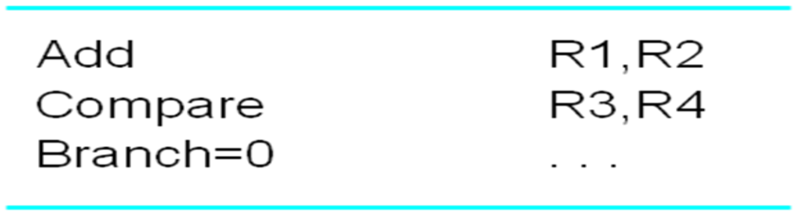
Conclusion: complex addressing modes are not suitable for pipelined execution.

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

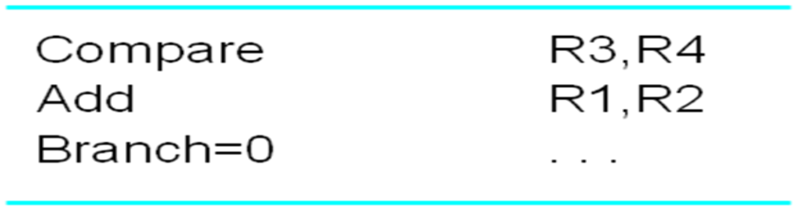
16. What are condition codes? Explain.

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

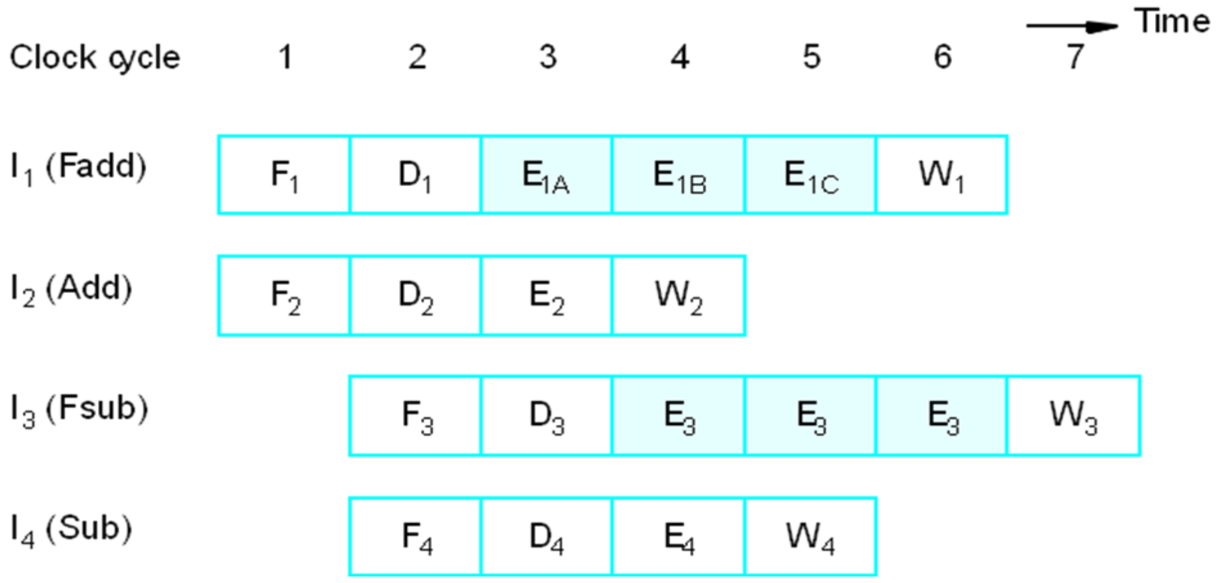
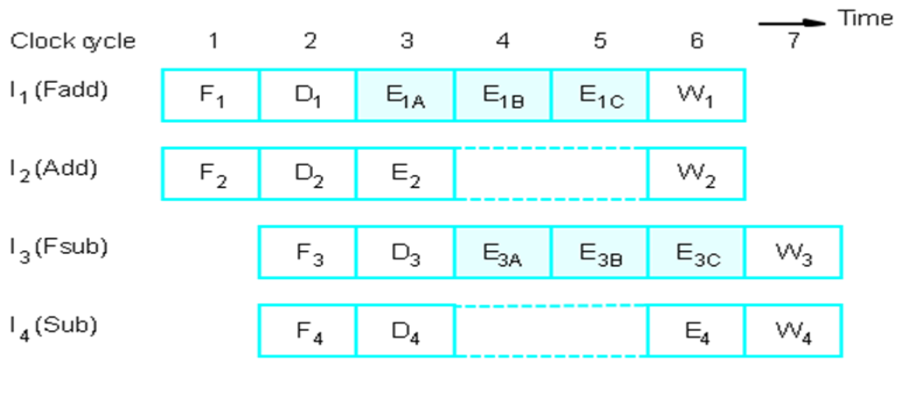


Fig.An example of instruction execution flow in the processor assuming no hazards are encountered.

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,)