1. What is computer architecture? What are the two different views of computer architecture?

ans: The science and art of designing hardware/software interface and designing, selecting and interconnecting hardware components to create a computing system that meets functionality requirements, performance, energy consumption, cost and other specific goals.
The two different views of computer architecture are:

i) Instruction set Architecture
ii) Computer organization

Instruction set architecture (ISA): what the computer does (logical view)
Computer organization: how the ISA is implemented (physical view)

2. A processor has dynamic instruction count of 150 million and requires an average of 3.4 clock cycles to execute an instruction. If the execution time of the program is 1.5 seconds, then what is the clock rate of processor?

ans: Here,

Instruction, $N = 150$ million $= 150 \times 10^6$ instruction
Clock cycle, $S = 3.4$ cycle/instruction
Execution time, $T = 1.5$ second
Clock Rate, $R = ?$

We know,

$$T = \frac{N \times S}{R}$$

$$\Rightarrow R = \frac{N \times S}{T}$$

$$\Rightarrow R = \frac{150 \times 10^6 \times 3.4}{1.5}$$

$$\therefore R = 34 \times 10^7 \text{ cycles/second}$$

(ans:)

3. What is addressing mode? Write a machine language program to find the multiplication of 10 integers numbers of an array using indirect addressing mode. Assume that the word length is 32 bits and the memory is byte addressable. The addresses of the memory locations containing the 10 numbers are symbolically given as NUM1, NUM2 ... NUM10 and after the multiplication, the result is placed in memory location RESULT.

ans: The different ways in which the location of an operand is specified in an instruction are referred to as addressing mode.

Program:

```
        MOVE      #10, R1
        MOVE      #NUM1, R2
        MOVE      #1, R0
LOOP    MUL       (R2), R0
        ADD       #16, R2
        Decrement R1
        Branch>0  LOOP
        MOVE      R0, RESULT
```

4. Write necessary machine instructions to evaluate the following statement : A = (A+2)* B - C *(D+200) using one address instruction format.

ans:

```
1. LOAD   A    ; AC ← M[A]
2. ADD    #2   ; AC ← AC +2
3. MUL    B    ; AC ← AC * M[B]
4. STORE  T    ; M[T] ← AC
```

```
5. LOAD    D      ;  AC ← M[D]
6. ADD     #200   ;  AC ← AC + 200
7. MUL     C      ;  AC ← AC * M[C]
8. SUB     T      ;  AC ← M[T] - AC
9. STORE   A      ;  M[A] ← AC
```

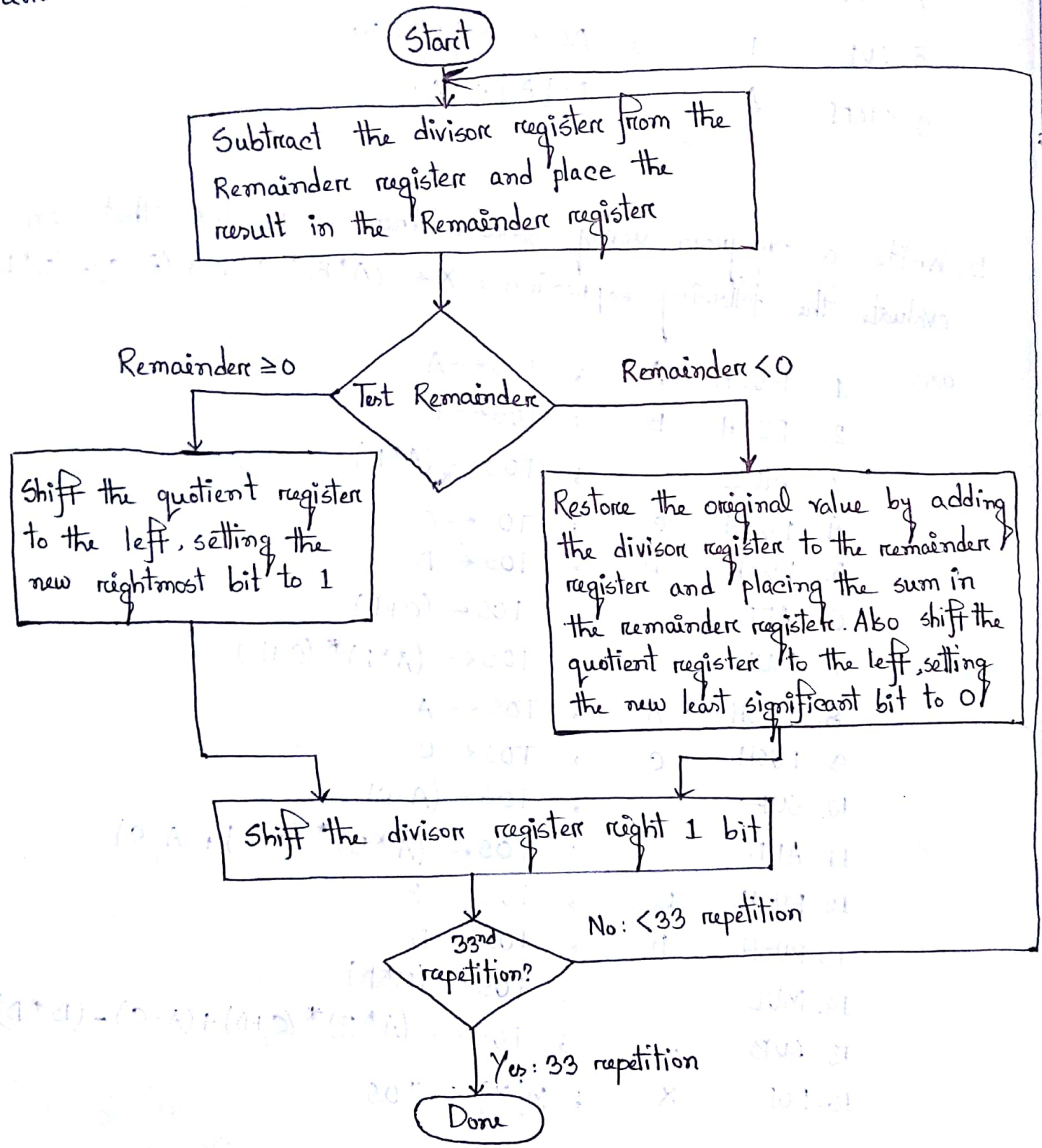
5. Write a program using zero-address instruction that can evaluate the following expression : $X \leftarrow (A*B)*(C+D)+(A-C)-(B*D)$

ans:

```
1.  PUSH   A   ;  TOS ← A
2.  PUSH   B   ;  TOS ← B
3.  MUL        ;  TOS ← (A*B)
4.  PUSH   C   ;  TOS ← C
5.  PUSH   D   ;  TOS ← D
6.  ADD        ;  TOS ← (C+D)
7.  MUL        ;  TOS ← (A*B)*(C+D)
8.  PUSH   A   ;  TOS ← A
9.  PUSH   C   ;  TOS ← C
10. SUB        ;  TOS ← (A-C)
11. ADD        ;  TOS ← (A*B)*(C+D)+(A-C)
12. PUSH   B   ;  TOS ← B
13. PUSH   D   ;  TOS ← D
14. MUL        ;  TOS ← (B*D)
15. SUB        ;  TOS ← (A*B)*(C+D)+(A-C)-(B*D)
16. POP    X   ;  M[X] ← TOS
```

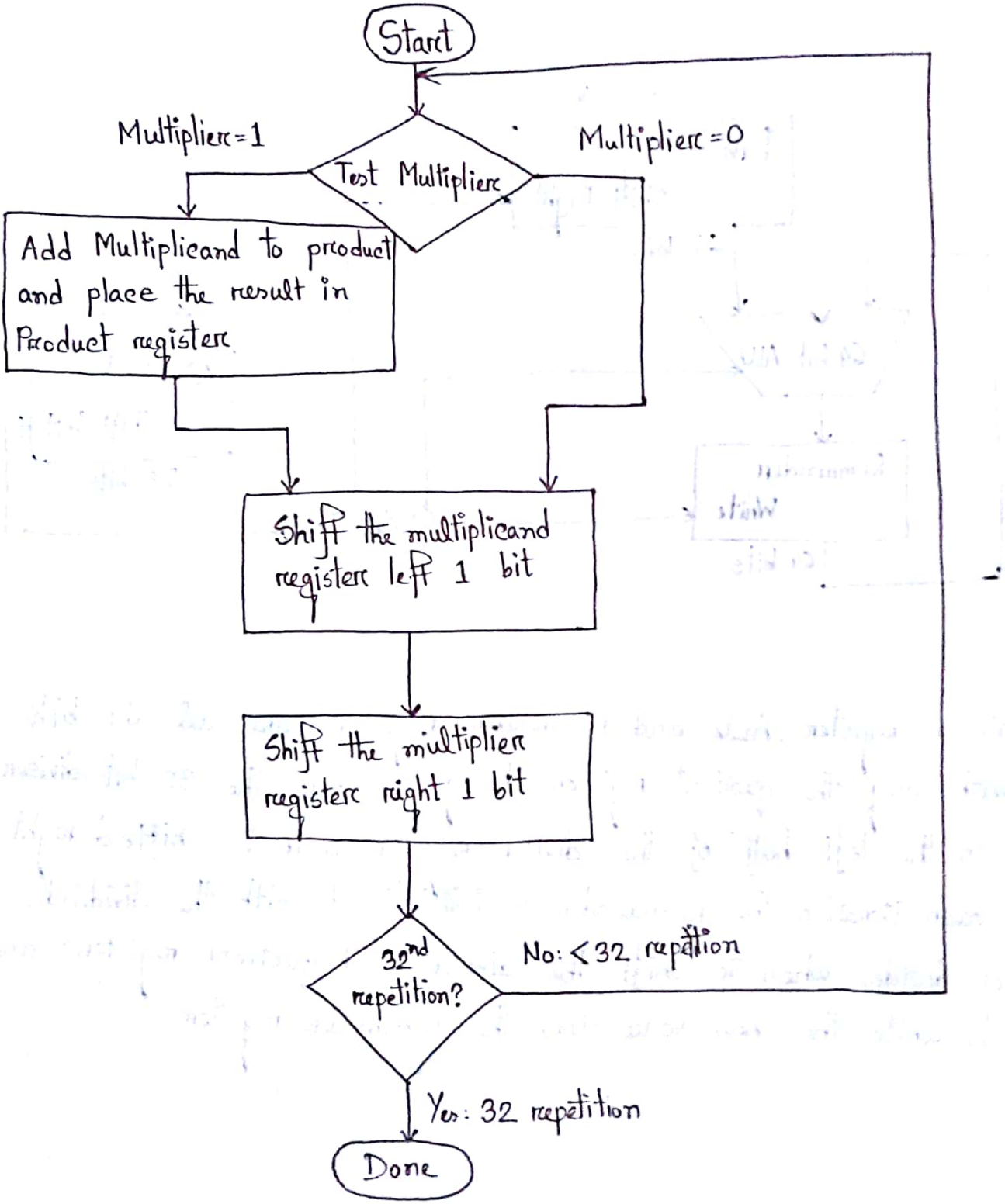
6. Draw the flowchart of sequential division algorithm for unsigned numbers.

ans:



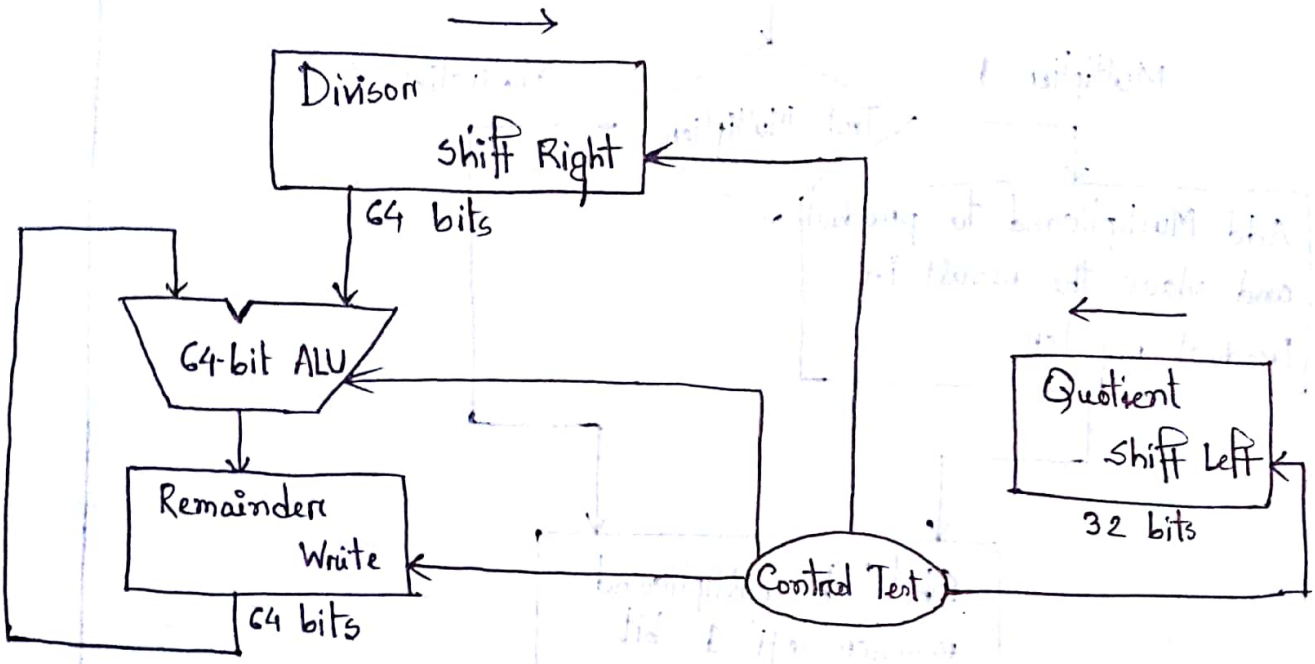
7. Draw the flowchart of sequential multiplication algorithm.

ans:



8. Design a sequential divisor circuit for unsigned numbers. Explain it's various components and operations.

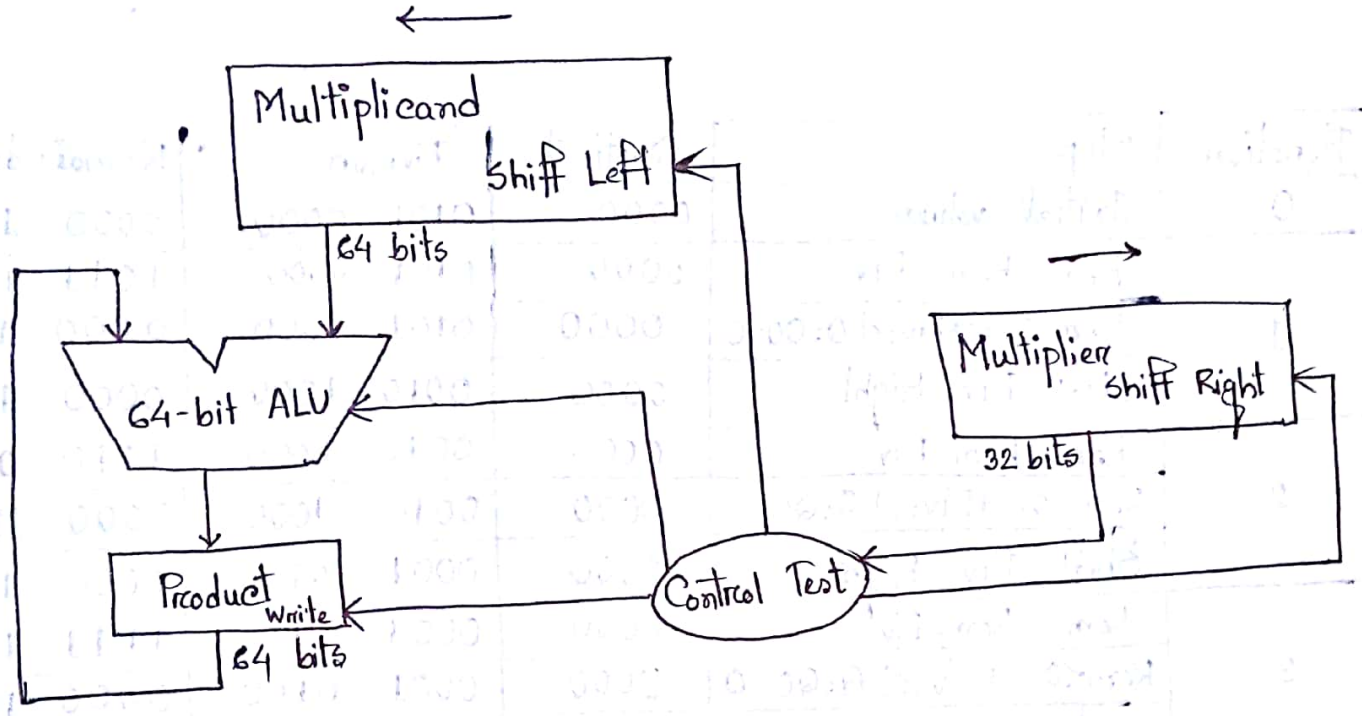
ans:



The divisor register, ALU and remainder register are all 64 bits wide, with only the quotient register being 32 bits. The 32-bit divisor starts in the left half of the divisor register and is shifted right 1 bit each iteration. The remainder is initialized with the dividend. Control decides when to shift the divisor and quotient registers and when to write the new value into the remainder register.

9. Design a sequential multiplier circuit for unsigned numbers. Explain it's various components and operations.

ans:



The multiplicand register, ALU and product register are all 64 bits wide, with only the multiplier register containing 32 bits. The 32 bit multiplicand starts in the right half of the multiplicand register and is shifted left 1 bit on each step. The multiplier is shifted in the opposite direction in each step. The algorithm starts with the product initialized to 0. Control decides when to shift the multiplicand and multiplier registers and when to write new values into the product register.

10. Divide $(1111)_2$ by $(0101)_2$ using the sequential division algorithm

ans:

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0101 0000	0000 1111
1	Rem = Rem - Div	0000	0101 0000	1011 1111
	Rem < 0 ⇒ +Div; sl Q; Q0=0	0000	0101 0000	0000 1111
	Shift Div Right	0000	0010 1000	0000 1111
2	Rem = Rem - Div	0000	0010 1000	1110 0111
	Rem < 0 ⇒ +Div; sl Q; Q0=0	0000	0010 1000	0000 1111
	Shift Div Right	0000	0001 0100	0000 1111
3	Rem = Rem - Div	0000	0001 0100	1111 1011
	Rem < 0 ⇒ +Div; sl Q; Q0=0	0000	0001 0100	0000 1111
	Shift Div Right	0000	0000 1010	0000 1111
4	Rem = Rem - Div	0000	0000 1010	0000 0101
	Rem > 0 ⇒ sl Q, Q0=1	0001	0000 1010	0000 0101
	Shift Div Right	0001	0000 0101	0000 0101
5	Rem = Rem - Div	0001	0000 0101	0000 0000
	Rem > 0 ⇒ sl Q, Q0=1	0011	0000 0101	0000 0000
	Shift Div Right	0011	0000 0010	0000 0000

$\therefore (1111)_2 \div (0101)_2 = (0011)_2$

(ans:)

11. Multiply $(0101)_2$ by $(0011)_2$ using the sequential Multiplication Algorithm

ans:

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	0011	0000 0101	0000 0000
1	1: Prod = Prod + Meand	0011	0000 0101	0000 0101
	Shift Left Multiplicand	0011	0000 1010	0000 0101
	Shift Right Multiplier	0001	0000 1010	0000 0101
2	1 : Prod = Prod + Meand	0001	0000 1010	0000 1111
	Shift Left Multiplicand	0001	0001 0100	0000 1111
	Shift Right Multiplier	0000	0001 0100	0000 1111
3	0 : no operation	0000	0001 0100	0000 1111
	Shift Left Multiplicand	0000	0010 1000	0000 1111
	Shift Right Multiplier	0000	0010 1000	0000 1111
4	0 : no operation	0000	0010 1000	0000 1111
	Shift Left Multiplicand	0000	0101 0000	0000 1111
	Shift Right Multiplier	0000	0101 0000	0000 1111

$$\therefore (0101)_2 \times (0011)_2 = (1111)_2$$

(ans:)

12. What are the tasks of a computer architect?

ans: a. determine which attributes are important for a new computer

b. design a computer to maximize performance and energy efficiency while staying within cost, power and availability constraints. The tasks has many aspects:

- → instruction set design
- → functional organization
- → logic design
- → implementation which encompass
  - integrated circuit design
  - packaging
  - power and cooling

c. optimizing the design

13. What is a computer? What are the basic functions performed by a computer?

ans: A computer is a electronic calculating machine that accepts digitized input information, processes the information according to a list of internally stored instructions and produces the resulting output information. Functions performed by a computer are:

- accepting information to be processed as input
- storing a list of instructions to process the information
- processing the information according to the list of instructions
- providing the results of the processing as output

14. What is BUS? What are the advantages and disadvantages of single bus architecture? Draw the basic block diagram of the single bus architecture?

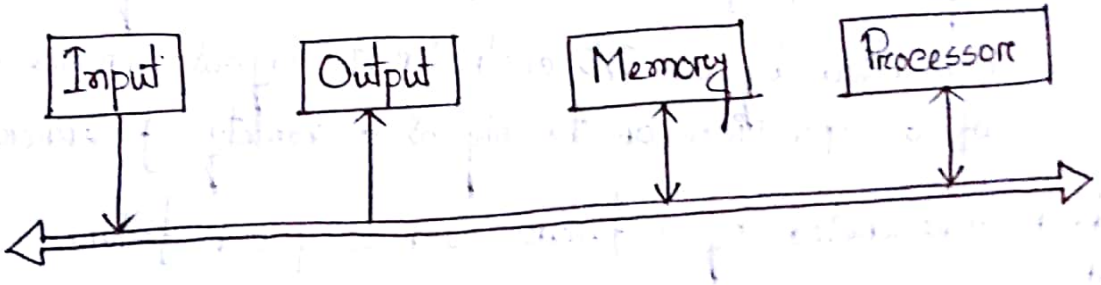
ans: A group of lines that serves a connecting path for several devices is called a bus.

Advantages of single bus architecture:

i) Low cost
ii) Compatibility
iii) Used widely

Disadvantages of single bus architecture:

i) Low speed
ii) Jumpers and DIP switches
iii) Becoming out-dated



15. Write down the procedure of interrupt operation. What are the four classes of interrupt?

ans: Normal execution of programs may be interrupted if some device requires urgent servicing to deal with the situation immediately, the normal execution of the current program must be interrupted.

Procedure of interrupt operation:

- the device raises an interrupt signal

- the processor provides the requested service by executing an appropriate interrupt-service routine
- the state of the processor is first saved before servicing the interrupt. Normally the contents of the PC, the general registers and some control information are stored in memory
- when the interrupt-service routine is completed, the state of the processor is restored so that the interrupted program may continue

Classes of interrupts:

- Program : Generated by some condition that occurs as a result of an instruction execution such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction or reference outside a user's allowed memory space
- Timer : Generated by a timer within the processor. This allows a operating system to perform certain functions on a regular basis
- I/O : Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions
- Hardware failure : Generated by a failure such as power failure:

16. What is operating system? What are the tasks of OS routines?

ans: Operating system is a large program or actually a collection of routines that is used to control a sharing of and interaction among various computer units as they perform application programs. The OS routines perform the tasks required to assign computer resource to individual application programs. These tasks include:

- assigning memory and magnetic disk space to program & data files
- moving data between memory and disk units
- handling I/O operations

17. What do you understand by Multiprocessors and multi-computer systems?

ans: Multiprocessor Computer:

→ good for executing several different application tasks in parallel

→ good for executing subtasks of a single large task in parallel

→ all processors have access to all of the memory - shared-memory multiprocessor. Example: some commercial server computers using two/four processors.

→ cost - processors, memory units, complex interconnection networks

Multicomputers:

→ each computer only have access to it's own memory

→ example: a network of computers, such as a LAN (local area network), WAN (wide area network), MAN (metropolitan area network) etc.

→ exchange message via a communication network - message-passing multicomputers

18. What is system software? What are the functions of system software?

ans: System software is a collection of programs that are executed as needed to perform functions such as

→ Receiving and interpreting user commands

→ Running standard application programs such as word processors etc or games

→ Managing the storage and retrieval of files in secondary storage devices

→ Controlling I/O units to receive input information and produce output results

19. Write a program that can evaluate the following expression using one address, two address and three address instruction schemes:

$$D \leftarrow A^2 + B^2 + 2AB$$

ans: One address instruction:

```
1. LOAD   A    ; AC ← M[A]
2. MUL    A    ; AC ← AC * M[A]
3. STORE  T    ; M[T] ← AC
4. LOAD   B    ; AC ← M[B]
5. MUL    B    ; AC ← AC * M[B]
6. ADD    T    ; AC ← AC + M[T]
7. STORE  T    ; M[T] ← AC
8. LOAD   A    ; AC ← M[A]
9. MUL    B    ; AC ← AC * M[B]
10. MUL   #2   ; AC ← AC * 2
11. ADD   T    ; AC ← AC + M[T]
12. STORE D    ; M[D] ← AC
```

Two address instruction:

```
1. MOV  A, R1    ; R1 ← M[A]
2. MUL  A, R1    ; R1 ← R1 * M[A]
3. MOV  B, R2    ; R2 ← M[B]
4. MUL  B, R2    ; R2 ← R2 * M[B]
5. ADD  R1, R2   ; R2 ← R1+R2
6. MOV  A, R3    ; R3 ← M[A]
7. MUL  B, R3    ; R3 ← R3 * M[B]
8. MUL  #2, R3   ; R3 ← R3*2
9. ADD  R2, R3   ; R3 ← R2+R3
10. MOV R3, D    ; M[D] ← R3
```

Three address instruction:

```
1. MUL  A, A, R1      ; R1 ← M[A] * M[A]
2. MUL  B, B, R2      ; R2 ← M[B] * M[B]
3. ADD  R1, R2, R3    ; R3 ← R1+R2
4. MUL  A, B, R4      ; R4 ← M[A] * M[B]
5. MUL  R4, #2, R5    ; R5 ← R4 * 2
6. ADD  R3, R5, D     ; M[D] ← R3+R5
```

20. Draw and briefly describe the internal organization of the memory chips with memory cells.

ans: Memory cells are usually organized in the form of an array, in which each cell is capable of storing one bit of information. A possible organization is illustrated in the figure where each row of cells constitutes a memory word and all cells of a row are connected to a common line referred to as the word line which

is driven by the address decoder on the chip. The cells in each column are connected to a Sense/Write circuit by two bit lines. The Sense/Write circuits are connected to the data input/output lines of the chip. During a Read operation, these circuits sense, or read, the information stored in the cells selected by a word line and transmit this information to the output data lines. During a write operation, the Sense/Write circuits receive input information and store it in the cells of the selected word.

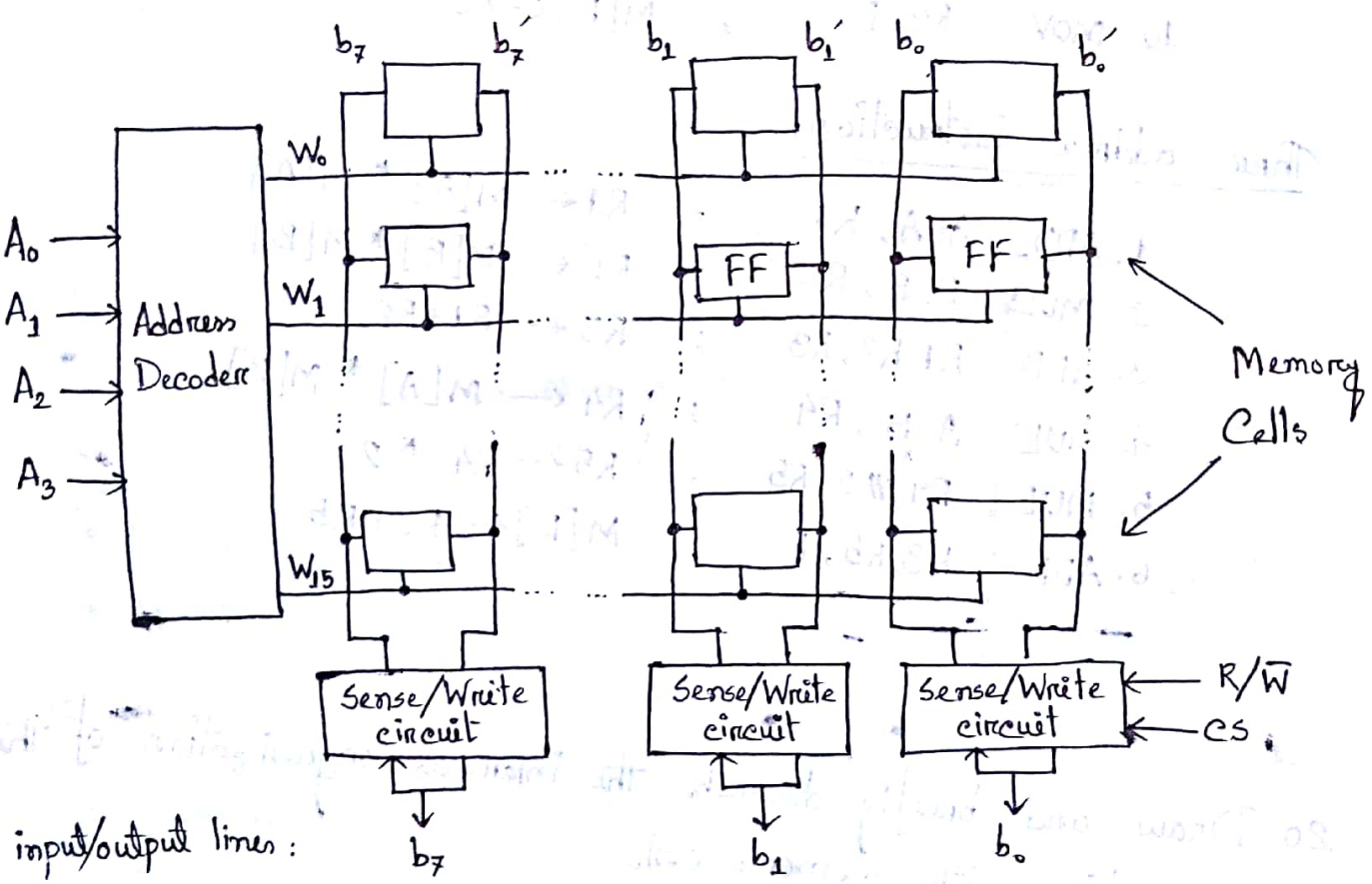


Figure: internal organization of memory chips size=128 bits (16×8)

The memory circuit in figure consists of 16 words of 8 bits each. This is referred to as a 16×8 organization. The data input and the data output of each Sense/Write circuit are connected to a single bidirectional

data line that can be connected to the data bus of a computer. Two control lines, R/$\overline{W}$ and CS, are provided in addition to address and data lines. The R/$\overline{W}$ input specifies the required operation and the CS (Chip Select) input selects a given chip in a multichip memory system. The memory circuit stores 128 bits and requires 14 external connections for address (4), data (8) and control (power supply, ground connection) lines.

21. What do you understand by locality of reference? State the two principles of locality of reference.

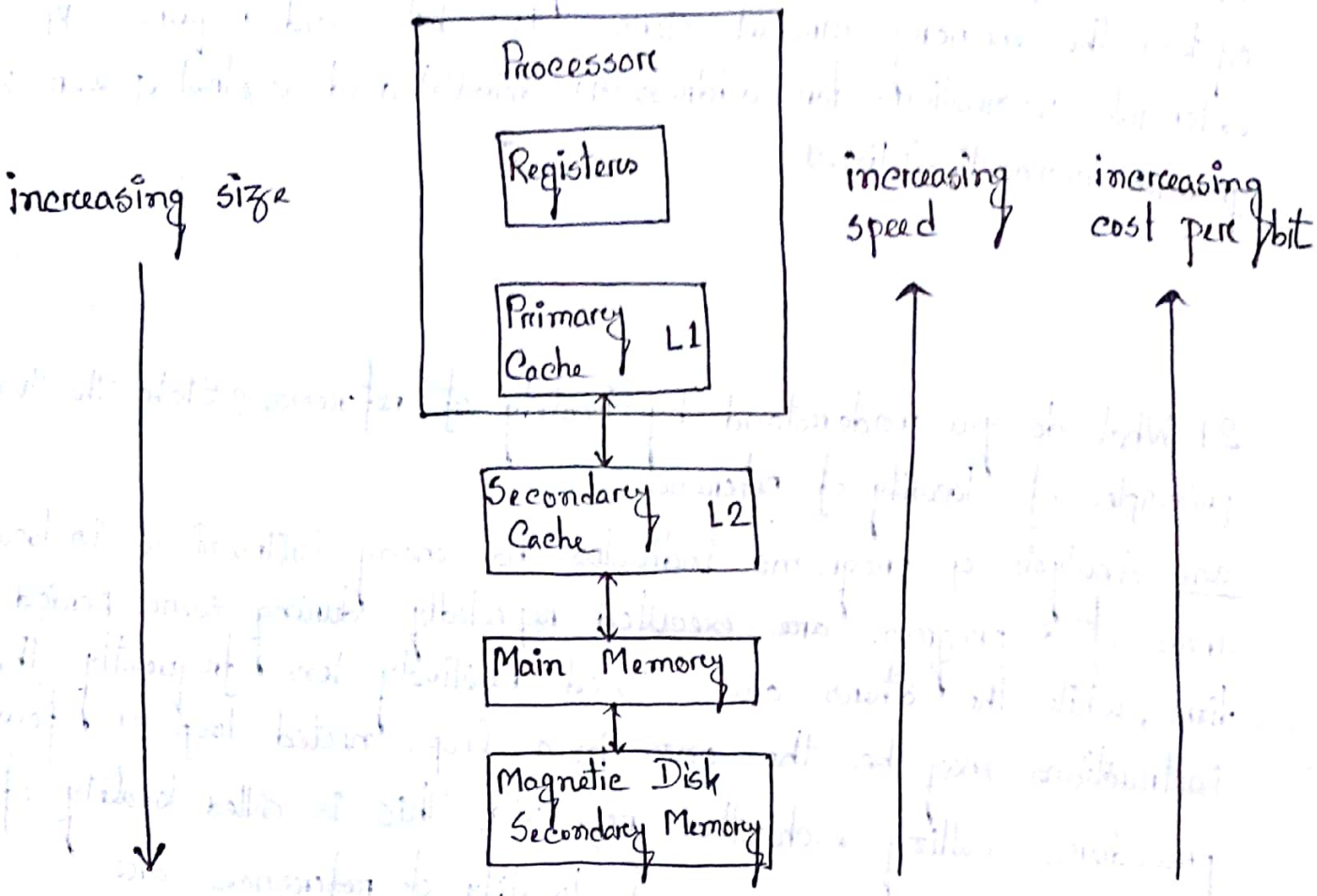
ans: Analysis of programs indicates that many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently. These instructions may be the ones in a loop, nested loop or few procedures calling each-other repeatedly. This is called locality of reference. The two principles of locality of reference are:

i) Temporal locality of reference: Whenever an instruction or data is needed for the first time, it should be brought into a cache. It will be used again repeatedly.

ii) Spatial locality of reference: Instead of fetching just one item from the main memory to the cache at a time, several items that have addresses adjacent to the item being fetched may be useful. The term "block" refers to a set of contiguous addresses locations of some size.

22. Draw the schematic diagram of the memory hierarchy and briefly explain the different levels of the memory hierarchy.

ans:



→ fastest access is to the data held in processor registers. Registers are at the top of the memory hierarchy

→ relatively small amount of memory that can be implemented on the processor chip. This is processor cache.

→ two levels of cache: Level 1 (L1): cache is on the processor chip
Level 2 (L2): cache is between the main memory & processor

→ next level is main memory, implemented as SIMMs. Much larger but much slower than cache memory

→ next level is magnetic disks. Huge amount of inexpensive storage

→ speed of memory access is critical, the idea is to bring instructions and data that will be used in the near future as close to the processor as possible

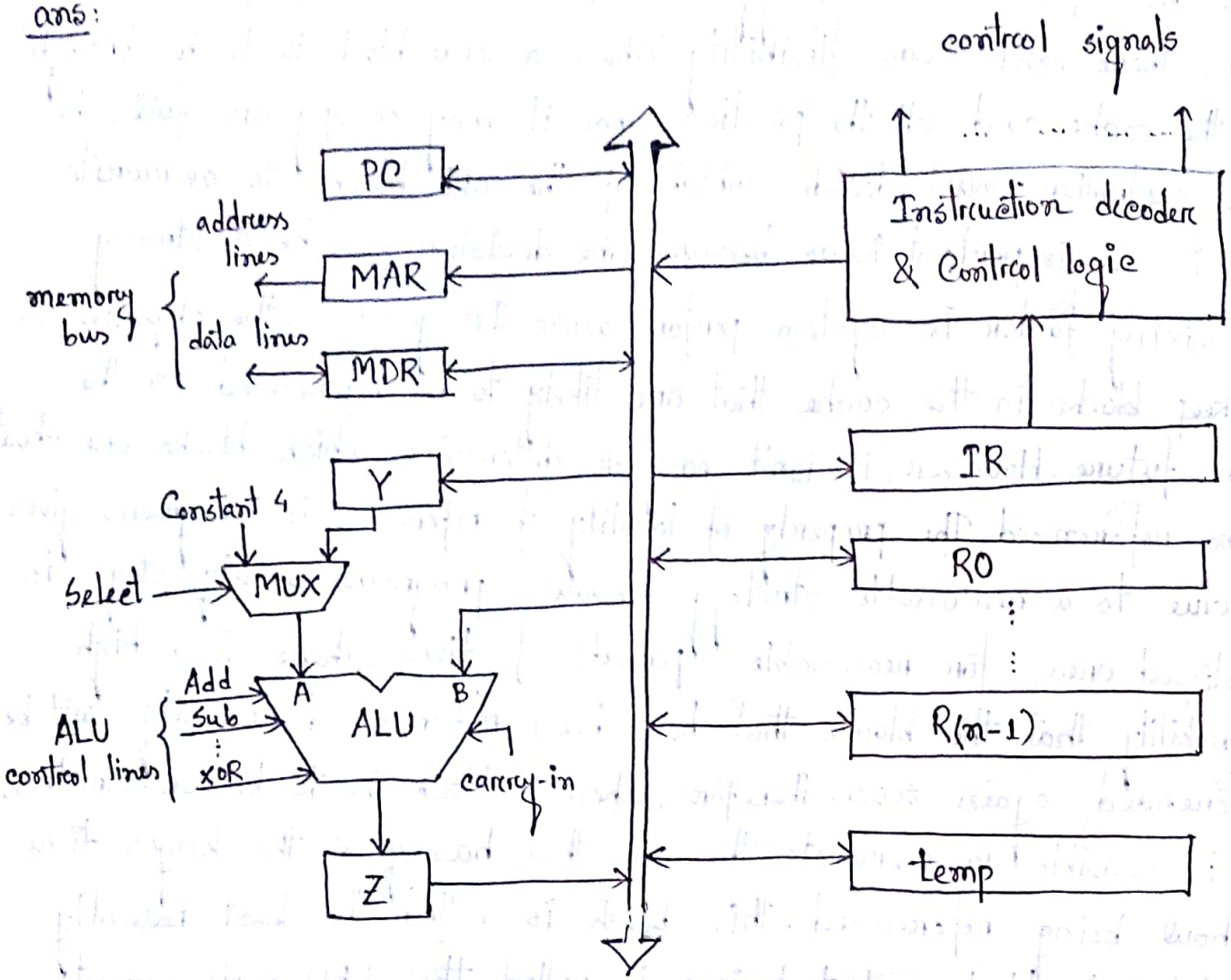
23. Briefly describe the LRU replacement algorithm.

ans: In a direct-mapped cache, the position of each block is predetermined hence no replacement strategy exists. In associative and set-associative caches there exists some flexibility. When a new block is to be brought into the cache and all the positions that it may occupy are full, the cache controller must decide which of the old blocks to overwrite. This is an important issue because the decision can be a strong determining factor in system performance. In general, the objective is to keep blocks in the cache that are likely to be referenced in the near future. However, it isn't easy to determine which blocks are about to be referenced. The property of locality of reference in programs gives a clue to a reasonable strategy. Because programs usually stay in localized areas for reasonable periods of time, there is a high probability that the blocks that have been referenced recently will be referenced again soon. Therefore, when a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced. This block is called the least recently used (LRU) block, and technique is called the LRU replacement algorithm.

LRU algorithm has been used extensively. It provides poore performance in some cases. Performance of the LRU algorithm may be improved by introducing a small amount of randomness into which block might be replaced.

24. Draw the basic single bus organization of the data path inside a processor, label it's various components and briefly explain their functions.

ans:



→ ALU, control unit and all the registers are connected via a single common bus

→ bus is internal to the processor and shouldn't be confused with the external bus that connects the processor to the memory and I/O devices

→ data lines of the external memory bus are connected to the internal processor bus via MDR. Register MDR has two inputs and two outputs. Data may be loaded to (from) MDR from (to) internal processor bus or external memory bus.

→ address lines of the external memory bus are connected to the internal processor bus via MAR. MAR receives input from the internal processor bus. MAR provides output to external memory bus.

→ instruction decoder and control logic block or control unit issues signals to control the operations of all units inside the processor and for interacting with the memory bus. Control signals depend on the instruction loaded in the instruction register (IR)

→ outputs from the control logic block are connected to control lines of the memory bus; ALU, to determine which operation is to be performed; select input of the multiplexer MUX to select between Register Y and Constant 4; control lines of the registers, to select the registers

→ registers Y, Z and temp are used by the processor for temporary storage during execution of some instructions.

→ registers R0 to R(n-1) are used to store data generated by one instruction for later use by another instruction

→ data is stored in R0 through R(n-1) after the execution of an instruction

→ multiplexer MUX selects either the output of register Y or a constant 4, depending upon the control input select

→ Constant 4 is used to increment the value of the PC

25. Write the sequence of control steps for the single bus structure for the instruction: MUL (R4), R3

ans:

1. PC_{out}, MAR_{in}, Read, Select4, MUL, Z_{in}
2. Z_{out}, PC_{in}, Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. $R4_{out}$, MAR_{in}, Read

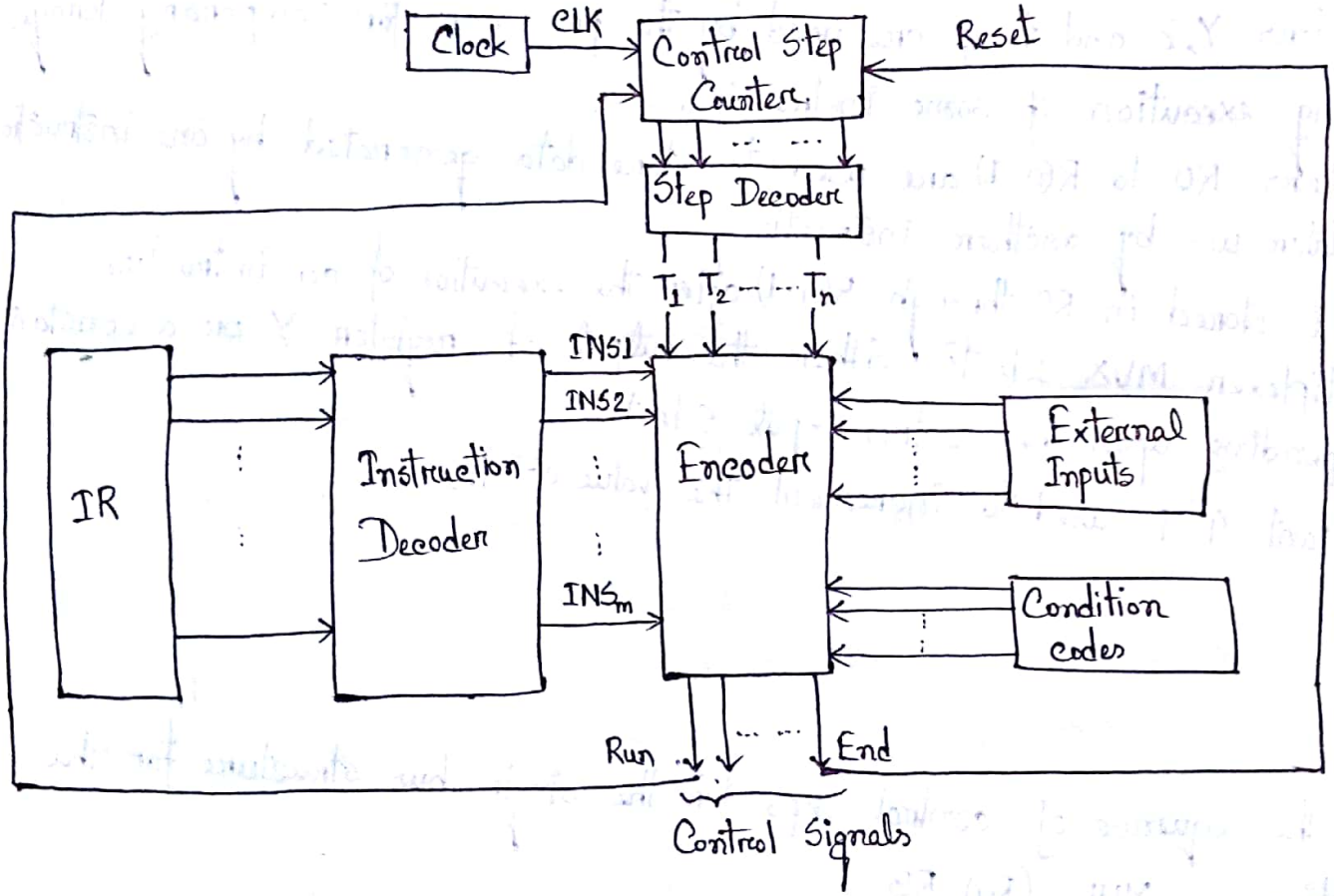
5. $R3_{out}$, Y_{in} , WMFC

6. MDR_{out} , Select Y, MUL, Z_{in}

7. Z_{out} , $R3_{in}$, End

26. Draw the basic block diagram of a hardwired processor control unit.

ans:



27. Identify the difference between single bus and three bus architecture

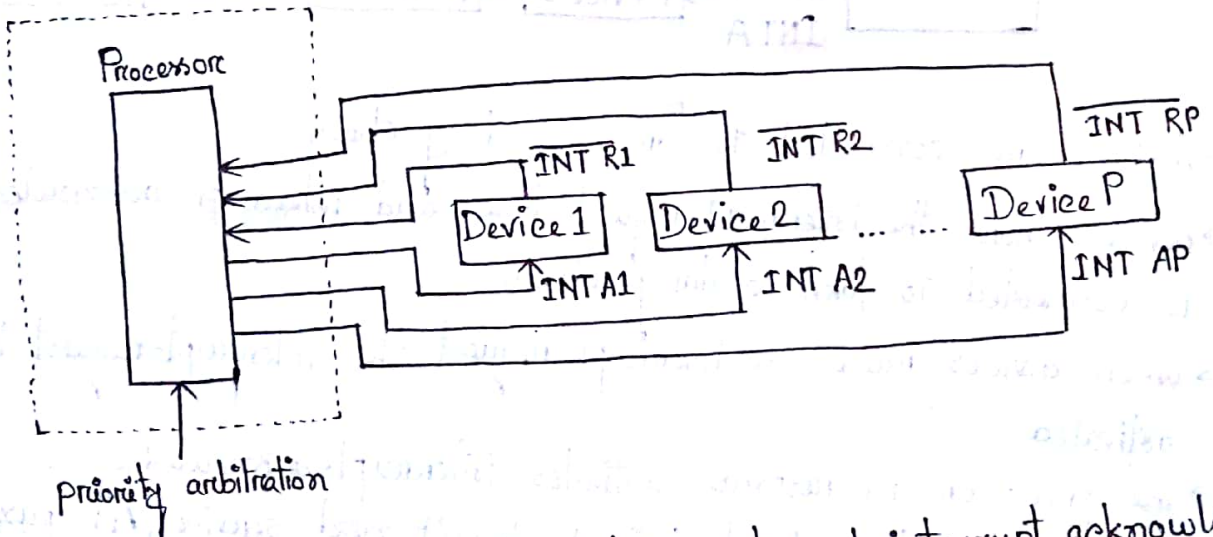
ans: i) Using single bus, a system can transfer only one data at a time whereas using three bus, a system can transfer multiple data at a time.

ii) three bus structure's can perform better than one bus structure.
iii) single bus structure's cost is cheaper than multiple bus structure
iv) single bus leads to propagation delay but three bus does not.

28. What are the different techniques to handle simultaneous interrupt requests? Explain each one of them.

ans: If more than one device interrupt simultaneously, the device with highest priority gets the service. Four schemes to handle simultaneous interrupt requests are:
i) Priority structured scheme
ii) Polling scheme
iii) Daisy chain scheme
iv) Prioritized daisy chain scheme

Priority Structured Scheme:



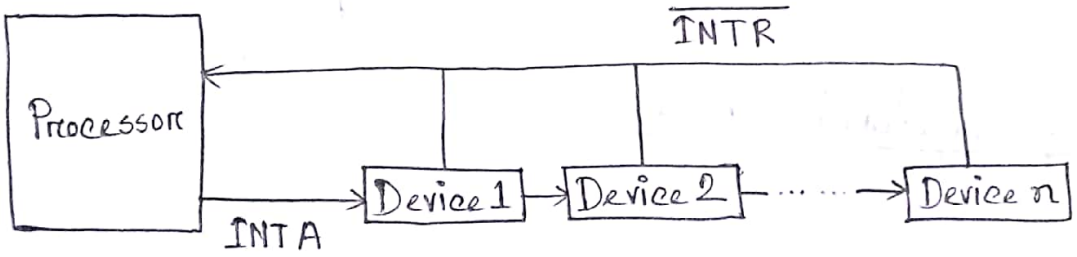
→ each device has a separate interrupt request and interrupt acknowledge line
→ each interrupt request line is assigned a different priority level
→ interrupt requests received over these lines are sent to a priority arbitration circuit in the processor

→ if the interrupt request has a higher priority level than the priority of the processor, then the request is accepted.

Polling Scheme:

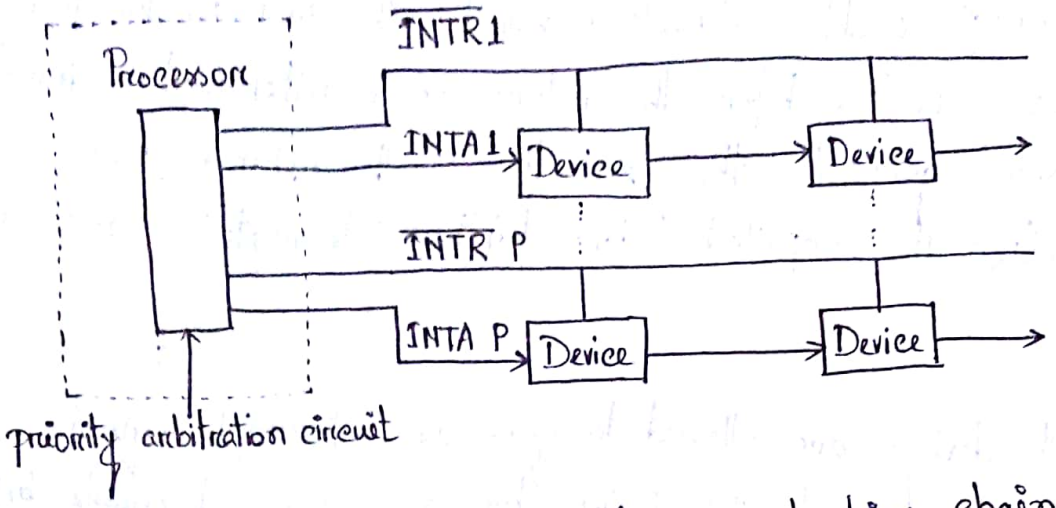
→ if the processor uses a polling mechanism to poll the status registers of I/O devices to determine which device is requesting an interrupt
→ in this case the priority is determined by the order in which the devices are polled
→ The first device with status bit set to 1 is the device whose interrupt request is accepted

Daisy chain Scheme:



→ devices are connected to form a daisy chain
→ devices share the interrupt-request line and interrupt acknowledge line is connected to form a daisy chain
→ when devices raise an interrupt request, the interrupt-request line is activated
→ the processor in response activates interrupt-acknowledge
→ received by device 1, if device 1 doesn't need service, it passes the signal to device 2
→ priority: device that is electrically closest to the processor has the highest priority (Priority = Electronic Proximity of the device to the processor)

Prioritized Daisy Chain Scheme:



→ a combination of priority structure and daisy chain schemes
→ devices are organized into groups
→ each group is assigned a different priority level
→ all the devices within a single group share an interrupt-request line and are connected to form a daisy chain. Inside a group, priority is determined by electronic proximity to the processor

29. What is the role of Interrupt Enable (IE) and Interrupt Disable (ID) instructions? How are they used by ISR? What's the role of the IE bit inside the I/O device interface and inside the CPU?

ans: The role of Interrupt Enable (IE) and Interrupt Disable (ID) instructions is to avoid interruption by the same or any other device during the execution of an interrupt service routine.

The processor hardware ignore the interrupt-request line until the execution of the first instruction of the interrupt-service routine has been completed. Then by using an Interrupt-disable instruction as the first instruction in

the interrupt-service routine, the programmer can ensure that no further interruptions will occur until an Interrupt enable instruction is executed. Typically, the interrupt enable instruction will be the last instruction in the interrupt service routine before the return from interrupt instruction. The processor must guarantee that execution of the return from interrupt instruction is completed before further interruption can occur

To control which devices are allowed to generate interrupt requests, the interface circuit of each I/O device has an interrupt enable bit. If the interrupt enable (IE) bit in the I/O device interface is set to 1 then the device is allowed to generate an interrupt request.

30. What is the role of cache memory in pipeline?

ans: If instructions are fetched from the main memory, the instruction fetch stage would take as much as ten times greater than the other stage operations inside the processor. However, if instructions are to be fetched from the cache memory which is on the processor chip, the time required to fetch the instruction would be more or less similar to the time required for other basic operations.

31. What is pipelining? "Pipelining doesn't reduce the latency of an instruction rather it improves the throughput" - Explain with an example.

ans: Pipelining is a particularly effective way of organizing concurrent activity in a computer system. Pipelining is commonly known as an assembly line operation. Pipelining increases the CPU

instruction throughput - the number of instructions completed per unit of time. But it doesn't reduce the execution time of an individual instruction. In fact, it usually slightly increases the execution time of each instruction due to overhead in the pipeline control. The increase in instruction throughput means a program runs faster and has lower total execution time. Considering a non-pipelined machine with 6 execution stages of lengths 50ns, 50 ns, 60 ns, 60ns, 50 ns, 50 ns.

∴ Instruction latency = (50 + 50 + 60 + 60 + 50 + 50) ns = 320 ns

∴ Time to execute 100 instructions = 100 × 320 = 32000 ns

If we use pipelining on that machine, considering the clock skew adds 5ns of overhead to each execution stage.

∴ Length of pipelined stage = MAX(length of unpipelined stage) + overhead = 60 + 5 = 65ns

∴ Instruction latency = 65 ns

∴ Time to execute 100 instructions = 65 × 6 × 1 + 65 × 1 × 99 = 390 + 6435 = 6825 ns

So, execution time before pipelining = 32000 ns

execution time after pipelining = 6825 ns

∴ speedup for 100 instructions = $\frac{32000}{6825}$ = 4.69

It can be said that, pipelining doesn't reduce the latency of an instruction rather it improves the throughput.

32. Suppose device A has ID 9 and device B has ID 5. If device A and device B simultaneously request the bus then who will be the bus master according to the distributed bus arbitration algorithm?

ans:

Device A ID = 9 ⇒ 1001
Device B ID = 5 ⇒ 0101 $\longrightarrow$ 1101 $\longrightarrow$ 1000 $\longrightarrow$ 1000 $\longrightarrow$ A wins

Process:

→ device A transmits the pattern 1001 on the arbitration lines
→ device B transmits the pattern 0101 on the arbitration lines
→ pattern 1101 appears on the arbitration lines which is the logical OR of 9,5.
→ each device compares the pattern that appears on the arbitration lines to it's own ID, starting with MSB
→ if it detects a difference, it transmit's 0's on the arbitration lines for that and all lower bit positions
→ device A compare it's ID 9 with a pattern 1001 to 1101
→ detecting difference, it transmits a pattern 1000 on the arbitration lines
→ the pattern that appears on the arbitration lines is the logical OR of 1000 and 0000, which is 1000.
→ this pattern is same as the device ID of A and hence A has won the arbitration procedure and becomes the bus master

33. What is the difference between an interrupt service routine (ISR) and a subroutine?

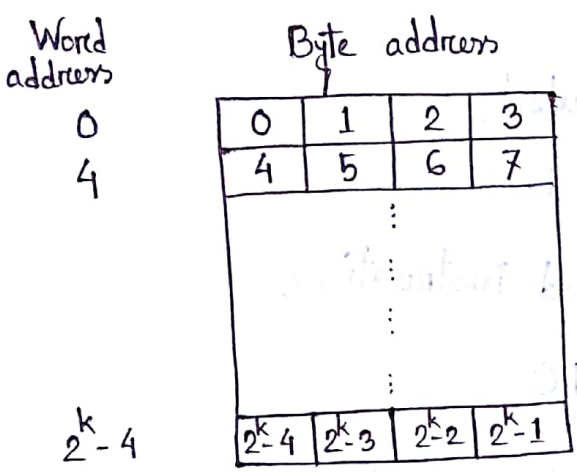
ans: Routine executed in response to an interrupt request is called the interrupt service routine (ISR). Subroutine is a portion of code within a large program which performs a specific task and is relatively independent of the remaining code. Interrupt Service Routines are to handle hardware interrupts. These routines aren't independent threads but more like signals. ISR is called if any thread is suspended by an interrupt. Subroutine runs when it is called. ISR runs whenever a certain signal occurs (signal can be generated by software or hardware).

34. With suitable figures, define big endian and little endian address assignment.

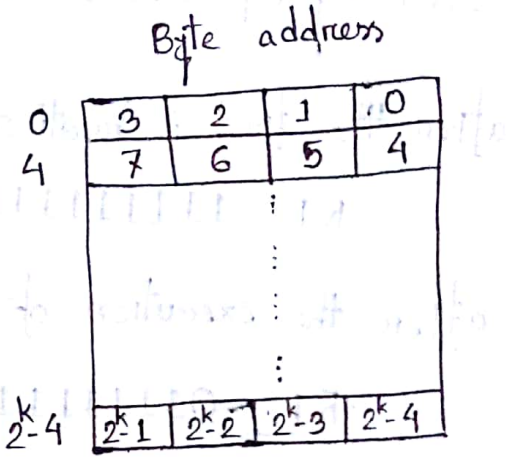
ans: There are two ways that byte addresses can be assigned across words:

i) Big-endian: big-endian is used when lower byte addresses are used for the more significant bytes (the leftmost bytes) of the word

ii) Little-endian: little-endian is used for the opposite ordering where the lower byte addresses are used for the less significant bytes (the rightmost bytes) of the word.



Big endian assignment



Little endian assignment

The word "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 big-endian and little-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.

35. What will be the content of register R1 after the following instructions are executed. Assume, R1 to be a 16 bit register and 2's complement representation is used for the negative numbers.

```
MOV       #-25, R1
RotateRC  #4, R1
AshiftR   #3, R1
```

ans: Here,

2's complement of -25 : 1000000000011001

∴ 1's complement : 1000000000011000

∴ Binary representation : 1111111111100111

when the first instruction is executed,

R1 = 1111111111100111

after the execution of the second instruction,

R1 = 0111111111111110

after the execution of the third instruction,

R1 = 1110111111111111

36. Assume that propagation delays along the bus and through the ALU are 2ns and 4ns respectively. The setup time for the registers is 0.2ns and the hold time is 0.3ns. What is the minimum clock period needed?

ans: Here, Minimum clock period = propagation delay + setup time + hold time

= 2ns + 4ns + 0.2ns + 0.3ns

= 6.5ns

37. What is condition code register? How condition codes or status flags set/Reset? Explain with an example.

ans: The processor keeps track of information about the results of various operations for use by subsequent conditional branch instructions. This 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. Four commonly used flags are:

N (negative)/S (sign) : set to 1 if the result is negative otherwise cleared to 0
Z (zero) : set to 1 if the result is zero otherwise cleared to 0
V (overflow) : set to 1 if arithmetic overflow occurs otherwise cleared to 0
C (carry) : set to 1 if a carry-out results from the operation otherwise cleared to 0

For example: A : 11110000 = (-16)
B : 00010100 = (20)

$\therefore A-B = -36$

$\therefore A = 11110000$

$\therefore +(-B) = 11101100$

$1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0$

$\downarrow\ \ \downarrow$

C S

Here, S=1 because the result is negative, Z=0 because the result is not zero, C=1 because there's a carry out from the operation, V=0 because no overflow occured though the carry-in and the carry-out is same.

N/S	Z	C	V
1	0	1	0

38. How does the following instruction work between main memory and processor : i) ADD R1, A

ii) ADD A, B

iii) ADD (R1), R2

iv) ADD (A), R1

v) ADD (A), (B)

vi) ADD (R1), (A)

Solution : i) ADD R1, A

steps:

1. The instruction address is stored in PC
2. The address of instruction (ADD) will go to the memory address register (MAR) from PC
3. Control unit (CU) will send the read signal
4. Control unit (CU) will send Wait for Memory Function (WMFC) to complete signal
5. The instruction will go to the memory data register (MDR) from memory
6. Memory will send the Memory Function Complete (MFC) signal
7. The instruction will then go from MDR to IR
8. The instruction fetch cycle will then complete and the execution cycle will st
9. The value of R1 will go to the ALU
10. The address location of A will send to the Memory Address Register (M
11. The Control unit (CU) will send the read signal
12. The Control unit (CU) will send WMFC signal
13. The value of A will go to the MDR
14. Memory will send the MFC signal
15. The value of A will go to the ALU from MDR
16. Control Unit (CU) will send the ADD signal
17. The addition will occur
18. The result of A will go to MDR
19. The Control unit will send the WRITE signal

20. Control unit will send the WMFC signal
21. The value of A will be stored in the memory
22. Memory will send the MFC signal

ii) ADD A, B

steps:

1. Address of the instruction (ADD) will go to PC
2. The address of instruction (ADD) will go to the MAR from PC
3. Control Unit (CU) will send the read signal
4. Control Unit (CU) will send the WMFC signal
5. The instruction will go to the MDR from memory
6. Memory will send the MFC signal
7. The instruction will then go from MDR to IR
8. The address location of A will send to the MAR
9. Control Unit (CU) will send read signal
10. Control Unit (CU) will send WMFC signal
11. The value of A will go to MDR from memory
12. Memory will send the MFC signal
13. The value of A will go to the ALU from MDR
14. The address location of B will send to the MAR
15. Control Unit (CU) will send read signal
16. Control Unit (CU) will send WMFC signal
17. The value of B will go to MDR from memory
18. Memory will send the MFC signal
19. The value of B will go to the ALU from MDR
20. Control Unit will send ADD signal
21. Addition will occur
22. The result of B will go to the MDR from ALU
23. The Control Unit (CU) will send WRITE signal

24. Control Unit (CU) will send WMFC signal
25. The value of B will be stored in the memory from MDR
26. Memory will send the MFC signal

iii) ADD (R1), R2

Steps:

1. Address of the instruction (ADD) will go to PC
2. Address of instruction (ADD) will go to the MAR from PC
3. Control Unit (CU) will send read signal
4. Control Unit (CU) will send WMFC signal
5. The instruction will go to the MDR from memory
6. Memory will send the MFC signal
7. The instruction will go to IR from MDR
8. R1 register has the address of R1 value. So, this address will send back to MAR
9. Control Unit (CU) will send read signal
10. Control Unit (CU) will send WMFC signal
11. The value of address R1 will go to MDR from memory
12. Memory will send MFC signal
13. The value of R1 will go to ALU from MDR
14. The value of R2 will go to ALU
15. Control Unit (CU) will send ADD signal
16. The addition will occur
17. The value of R2 will be stored in R2 register from ALU

iv) ADD (A), R1

1. Address of the instruction (ADD) will go to PC
2. Address of instruction (ADD) will go to the MAR from PC
3. Control Unit (CU) will send read signal
4. Control Unit (CU) will send WMFC signal

5. The instruction will go to MDR from memory
6. Memory will send MFC signal
7. The instruction will then go to IR from MDR
8. The address location of A(i) will send to the MAR
9. Control Unit (CU) will send read signal
10. Control Unit (CU) will send WMFC signal
11. The value of address location A(i+1) will be sent to MDR from memory
12. Memory will send MFC signal
13. The address location of A will send to MAR from MDR
14. Control Unit (CU) will send read signal
15. Control Unit (CU) will send WMFC signal
16. The value of A will be sent to MDR from memory
17. Memory will send MFC signal
18. The value of A will go to ALU from MDR
19. The value of R1 will go to ALU
20. Control Unit (CU) will send ADD signal
21. Addition will occur
22. The value of R1 will be stored in R1 register from ALU

v) ADD (A), (B)

1. Address of the instruction (ADD) will go to PC
2. Address of instruction (ADD) will go to the MAR from PC
3. Control Unit (CU) will send read signal
4. Control Unit (CU) will send WMFC signal
5. The instruction will go to MDR from memory
6. Memory will send MFC signal
7. The instruction will then go to IR from MDR
8. Address location of A(i) will send to the MAR
9. Control unit will send read signal then control unit will send WMFC signal
10. The value of A which has the location address of A(i+1) will be sent to MDR
11. Memory will send MFC signal
12. Address location of A will send to MAR from MDR
13. Control unit will send read signal then control unit will send WMFC signal
14. The value of A will be sent to MDR from memory; memory will send MFC signal

15. The value of A will go to ALU from MDR
16. Address that holds the address of B as a value or data will go to MAR
17. Control Unit (CU) will send read signal
18. Control Unit (CU) will send WMFC signal
19. The value of B which has the location address of actual value of B will go to MDR
20. Memory will send MFC signal
21. The address of B will go to MAR from MDR
22. Control Unit will send read signal
23. Control Unit will send WMFC signal
24. The value of B will go to MDR from memory
25. Memory will send MFC signal
26. The value of B will go to ALU from MDR
27. Control Unit will send ADD signal
28. Addition will occur
29. The value of B will go to MDR from ALU
30. Control Unit will send Write signal
31. Control Unit will send WMFC signal
32. The value of B will go to memory from MDR
33. Memory will send MFC signal

vi) ADD (R1), (A)

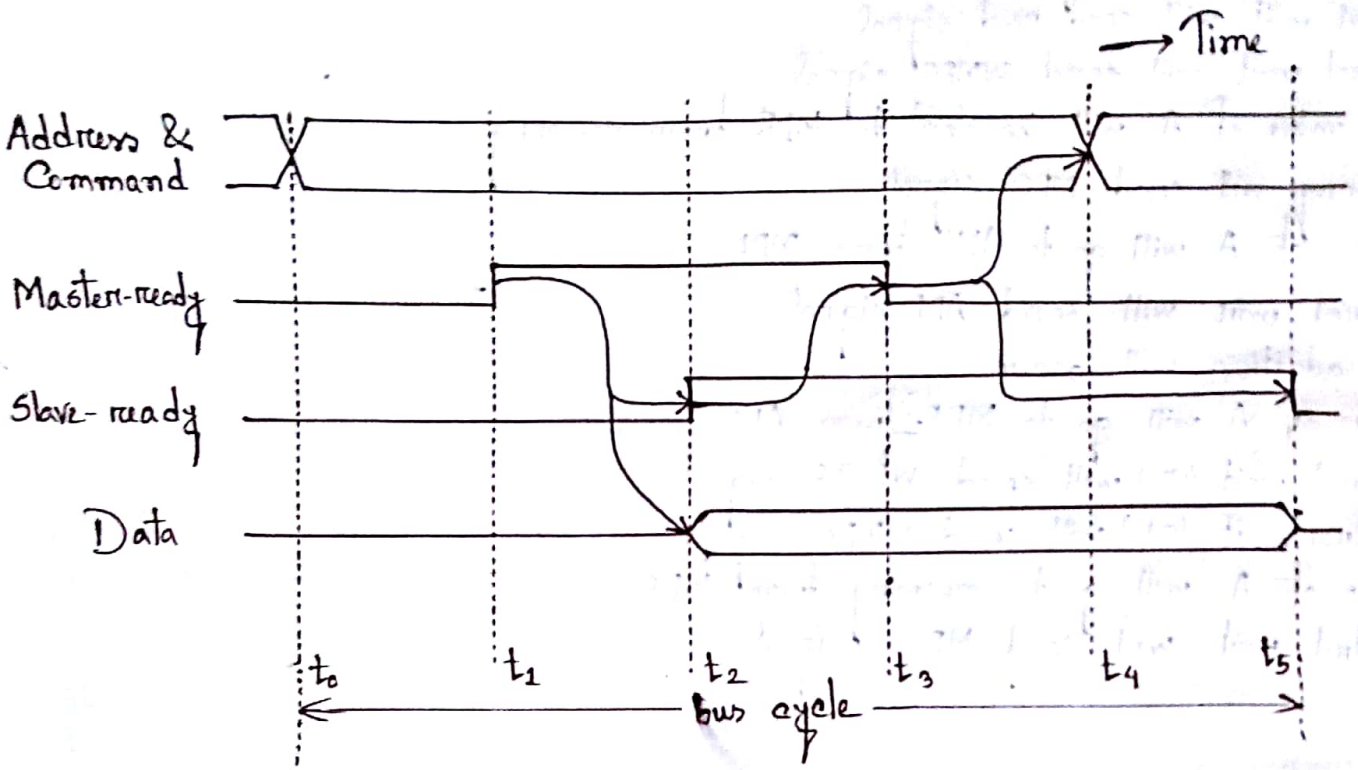
1. Address of the instruction (ADD) will go to PC
2. Address of instruction (ADD) will go to the MAR from PC
3. Control Unit (CU) will send read signal
4. Control Unit (CU) will send WMFC signal
5. The instruction will go to MDR from memory
6. Memory will send MFC signal
7. The instruction will then go to IR from MDR
8. Address of R1 will be sent to MAR
9. Control unit will send read signal
10. Control unit will send WMFC signal
11. The value of R1 will be sent to MDR from memory
12. Memory will send MFC signal
13. The value of R1 will be sent to ALU from MDR
14. Address that holds the address of A as a value will be sent to MAR
15. Control unit will send read signal
16. Control unit will send WMFC signal
17. Value of A that holds the address of actual value of A will go to MDR

18. Memory will send MFC signal
19. Address of A will go to MAR from MDR
20. Control unit will send read signal
21. Control unit will send WMFC signal
22. The value of A will be sent to MDR from memory
23. Memory will send MFC signal
24. Value of A will go to ALU from MDR
25. Control unit will send ADD signal
26. The addition will occur
27. Value of A will go to MDR from ALU
28. Control unit (CU) will send WRITE signal
29. Control unit (CU) will send WMFC signal
30. Value of A will go to memory from MDR
31. Control unit will send MFC signal

39. Describe the data transfer procedure using Handshake protocol in asynchronous bus?

ans: Data transfer using the handshake protocol:

→ Master places the address and command information on the bus.
→ Asserts the master-ready signal to indicate to the slaves that address and command info has been placed on the bus
→ All devices on the bus decode the address
→ Address slave performs the required operation and informs the processor it has done so by asserting the slave-ready-signal
→ Master removes all signals from the bus once slave-ready is asserted
→ if the operation is a read operation, master also strobes the data into its input buffer



Here,

t_0 : master places the address and command information on the bus

t_1 : master asserts the master-ready-signal. Signal is asserted at t_1 instead of t_0

t_2 : addressed slave places the data on bus and asserts slave-ready-signal

t_3 : slave-ready-signal arrives at the master

t_4 : master strobes data and removes the address and command information

t_5 : slave receives the transition of the master-ready signal from 1 to 0. It removes the data and the slave-ready-signal from the bus

40. What is hazard? Explain the three different types of hazard with necessary examples.

ans: Any condition that causes a pipeline to stall is called a Hazard. The three types of hazards are:

i) Data hazard
ii) Instruction (control) hazard
iii) Structural hazard

Data hazard: Any condition in which either the source or the destination operands of an instruction aren't available at the time expected in the pipeline. So, some operation has to be delayed and the pipeline stalls. 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..

A ← 3+R1	A ← 5 x C
B ← 4 x A	B ← 20+C
Hazard Occurs	No Hazard

When two operations depend on each other, they must be executed sequentially in the correct order. For example:

```
MUL   R2, R3, R4    → R4 = R2*R3
ADD   R5, R4, R6    → R6 = R4+R5
```

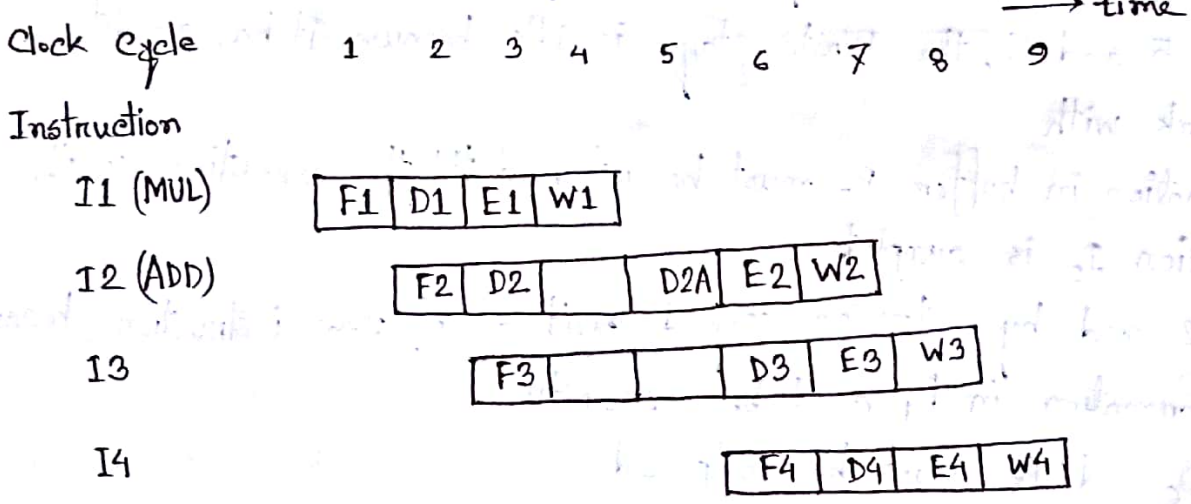
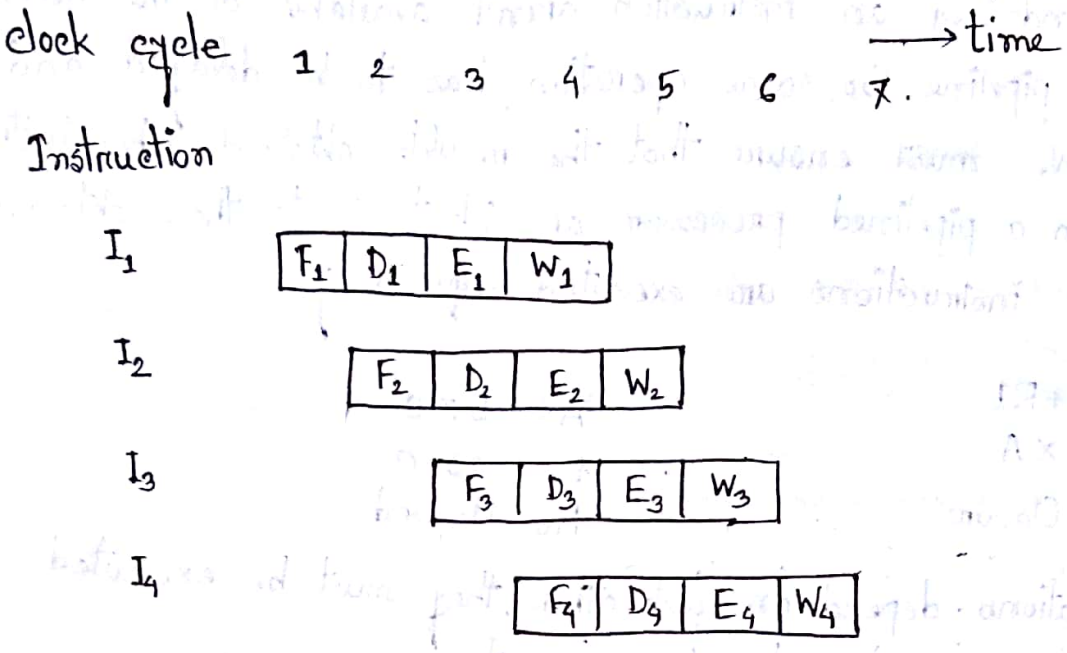


Figure: pipeline stalled by data dependency between D2 and W1

The description how data hazard occurs is given below:

→ execution of the instruction occurs in the E stage of the pipeline
→ execution of most arithmetic and logic operations would take only one clock cycle
→ however some operations such as division would take more time to complete



→ For example, the operation specified in instruction I_2 takes three cycles to complete from cycle 4 to cycle 6

→ cycles 5 and 6, the Write stage is idle because it has no data to work with

→ Information in buffer B_2 must be retained till the execution of the instruction I_2 is complete.

→ stage 2 and by extension stage 1 can't accept new instructions because the information in B_1 can't be overcwritten

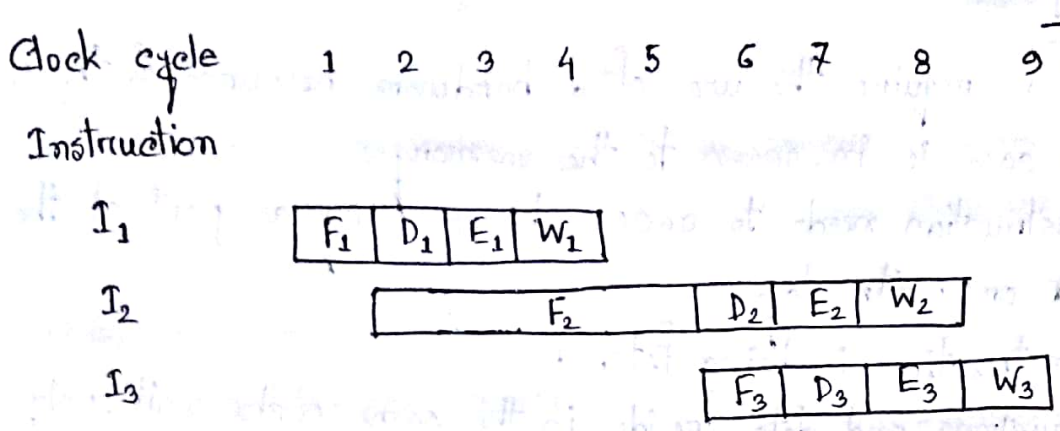
→ steps D_6 and F_5 must be postponed

Thus, the lack of availability of data at the expected time causes data hazard

Control or Instruction hazard:

→ pipeline may be stalled because an instruction isn't available at the expected time

→ For example, while fetching an instruction a cache miss may occur and hence the instruction may have to be fetched from main memory



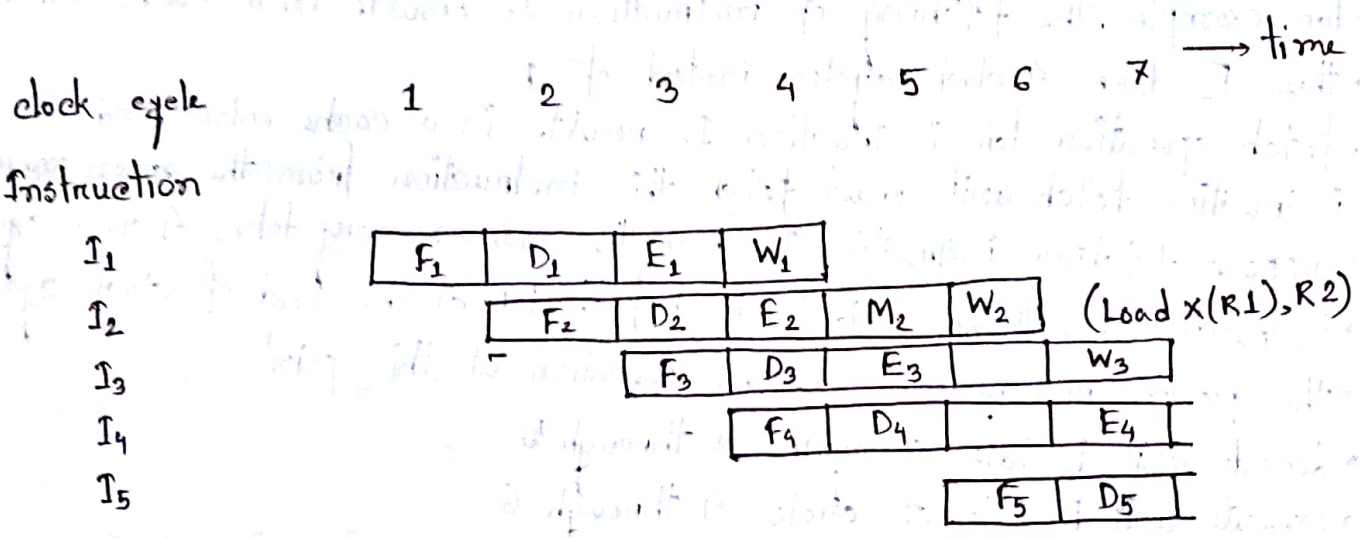
→ Fetching the instruction from the main memory takes much longer than fetching the instruction from the cache
→ thus the fetch cycle of the instruction can't be completed in one cycle
→ for example, the fetching of instruction I_2 results in a cache miss
→ thus F_2 takes 4 clock cycles instead of 1
→ fetch operation for instruction I_2 results in a cache miss and the instruction fetch unit must fetch this instruction from the main memory
→ suppose fetching instruction I_2 from the main memory takes 4 clock cycles
→ instruction I_2 will be available in buffer B1 at the end of clock cycle 5
→ the pipeline resumes its normal operation at this point
→ decode unit is idle at cycles 3 through 5
→ execute unit is idle at cycles 4 through 6
→ write unit is idle at cycles 5 through 7
→ such idle periods are called as stalls or bubbles
→ once created in one of the pipeline stages, a bubble moves downstream unit it reaches the last unit

clock cycle → time	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

Structural Hazard:

→ two instructions require the use of a hardware resource at the same time
→ most common case is in access to the memory:
- one instruction needs to access the memory as part of the execute or write stage
- other instruction is being fetched
- if instructions and data reside in the same cache unit, only one instruction can proceed and the other is delayed

→ many processors have separate data and instruction caches to avoid delays
→ in general, structural hazards can be avoided by providing sufficient resources on the processor chip



→ memory address X+R1 is computed in step E_2 in cycle 4, memory access takes place in cycle 5, operand read from the memory is written into register R2 in cycle 6
→ execution of instruction I_2 takes two clock cycles 4 and 5
→ in cycle 6, both instructions I_2 and I_3 require access to register file
→ pipeline is stalled because the register file can't handle two operations at once.

41. Let a processor works at a clock cycle of 10 seconds, a program having 50 instructions and each instruction has 4 steps on an average

i) What is the clock rate of the machine?

ii) Determine the execution time of the program?

ans: i) The processor takes,

10 seconds to complete 1 cycle

$\therefore$ 1 second to complete $\frac{1}{10}$ cycle

$= 0.1$ cycle

$\therefore$ Clock rate, $R = 0.1$ cycle/second

ii) Here,

instruction, $N = 50$

clock cycle, $S = 4$ cycle/instruction

clock rate, $R = 0.1$ cycle/second

execution time, $T = ?$

We know,

$$T = \frac{N \times S}{R}$$

$$\Rightarrow T = \frac{50 \times 4}{0.1}$$

$\therefore T = 2000$ seconds (ans:)

42. Consider a system whose fetch, decode, execute, write stages require 3ns, 15ns, 40ns and 20ns respectively. There are 5 instructions from I_1 to I_5. Calculate how much time will be required to complete the execution of the instructions?

ans: Here,

Instruction latency $= (3+15+40+20)$ ns $= 78$ ns

$\therefore$ Time to execute 5 instructions $= (78 \times 5)$ ns $= 390$ ns

If we use pipelining on the system, considering the clock skew adds 3ns of overhead to each execution stage.

∴ Length of pipelined stage = MAX (Length of unpipelined stage) + overhead
= MAX (3, 15, 40, 20) + 3
= 40 + 3
= 43

∴ Instruction latency = 43ns
∴ Time to execute 5 instructions = 43×4×1 + 43×1×4 = 172 + 172 = 344ns

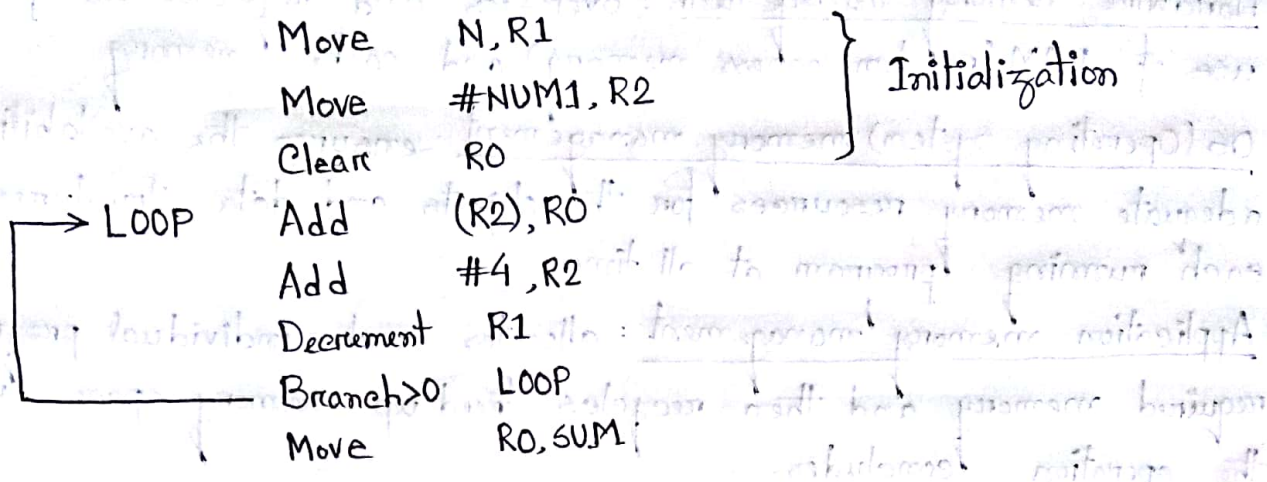
So,
execution time before pipelining = 390 ns
execution time after pipelining = 344ns
(ans:)

43. What is the purpose of branching? Explain with an example.

ans: Branch instructions load a new value into the program counter. As a result, the processor fetches and executes the instruction at this new address, called the branch target, instead of the instruction at the location that follows the branch instruction in sequential address order. A conditional branch instruction causes a branch only if a specified condition is satisfied. If the condition is not satisfied, the PC is incremented in the normal way, and the next instruction in sequential address order is fetched and executed.

Indirect Addressing in a loop to add N numbers using branch:

Assume that the number of entries in the list, N, is stored in memory location N. Register R1 is used as a counter to determine the number of times the loop is executed. Hence, the contents of location N are loaded into register R1 at the beginning. The instruction "Decrement R1"



reduces the contents of R1 by 1 each time through the loop. Execution of the loop is repeated as long as the result of the decrement operation is greater than zero. The instruction "Branch >0 LOOP" is a conditional branch instruction that causes a branch to location LOOP if the result of the immediately preceeding instruction which is the decremented value in register R1, is greater than zero. This means that the loop is repeated as long as there are entries in the list that are yet to be added to R0. At the end of the N^{th} pass through the loop, the decrement instruction produces a value of zero and hence branching doesn't occur. Instead, the MOVE instruction is fetched and executed. It moves the final result from R0 into memory location SUM.

44. Describe the purpose of a Memory Management Unit (MMU)?

ans: A memory management unit (MMU) is a computer hardware component that handles all memory and caching operations associated with the processor. In other words, the MMU is responsible for all aspects of memory management. It is usually integrated into the processor, although in some systems it occupies a separate IC (Integrated Circuit) chip. The work of MMU can be divided into 3 major catagories:

→ Hardwire memory management : oversees and regulates the processor's use of RAM (random access memory) and cache memory

→ OS (Operating System) memory management : ensures the availability of adequate memory resources for the objects and data structures of each running program at all times.

→ Application memory management : allocates each individual program's required memory and then recycles freed-up memory space when the operation concludes.

The main two purpose of MMU are :

→ First to give a process a linear memory space that isn't dependent on the actual location or size of the machine's physical memory. This allow a program to be coded in a standard manner so it can be run along with other programs without having to recompile or relocate the program

→ Second the OS can configure the MMU to provide isolation between programs and even allow over-committing memory and use some disk space as a swap file so that programs larger than the physical memory can be executed.