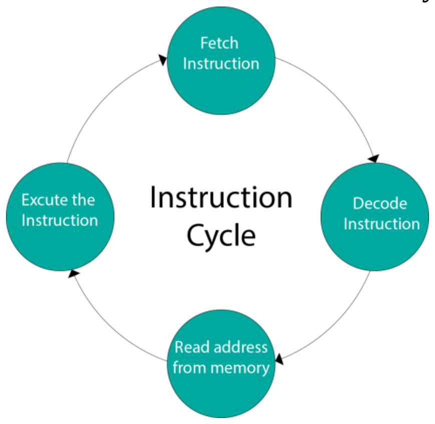
COMPUTER ORGANIZATION AND ARCHITECTURE(CS T36)
SECOND YEAR / THIRD SEMESTER (2022-2023)
ANSWER KEY

PART A – (10*2=20)

1. **Draw the flow of the instruction cycle.(CT1)**



2. **Define stack.(CT1)**
The stack is **a list of data words**. It uses the Last In First Out (LIFO) access method which is the most popular access method in most of the CPU. A register is used to store the address of the topmost element of the stack which is known as Stack pointer (SP).
3. **What is Subroutine? (CT1)**
Subroutines are **small blocks of code in a modular program designed to perform a particular task**. Since a subroutine is in itself a small program, it can contain any of the sequence , selection and iteration constructs.
4. **Write down the stages of Instruction Execution.(CT1)**

In a basic computer, each instruction cycle consists of the following phases:

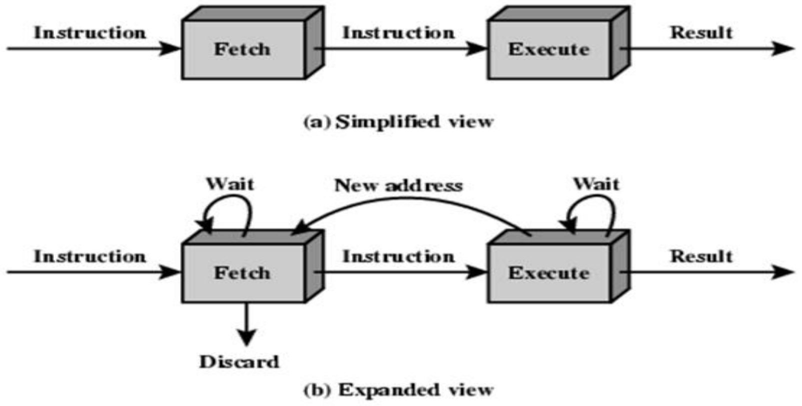
1. Fetch instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory.
4. Execute the instruction.

5. **Why does the DMA priority over CPU when both request memory transfer?(CT2)**
The CPU isinvolved only at the beginning and end of the transfer. **When the CPU wishes to read or write a block a data, it issues a command to the DMA channel bysending read/write operation, address of I/O, number of words to be read or written**. Hence DMA have priorityover the CPU when both request a memory transfer.
6. **List out the types of interrupts.(CT2)**
Interrupts have two types:
Hardware interrupt and
Software interrupts.

7. **What is virtual memory?(CT2)**
Virtual memory is **a common technique used in a computer's operating system (OS)**. Virtual memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily transferring data from random access memory (RAM) to disk storage.

8. **Give the formula to calculate the average memory access time.**
Average Memory Access Time = **Hit ratio * Cache Memory Access Time + (1 – Hit ratio) * Time** required to access a block of main memory.

9. **Draw the hardware organization of the two-stage pipeline.(M)**



10. **List the types of data hazards.**

There are three situations in which a data hazard can occur:

Read after write (RAW), a true dependency.

Write after read (WAR), anti-dependency.

Write after write (WAW), an output dependency.

PART B – (11*5=55)

UNIT I

11. Define addressing modes. Give the details of different addressing modes. (CT1)

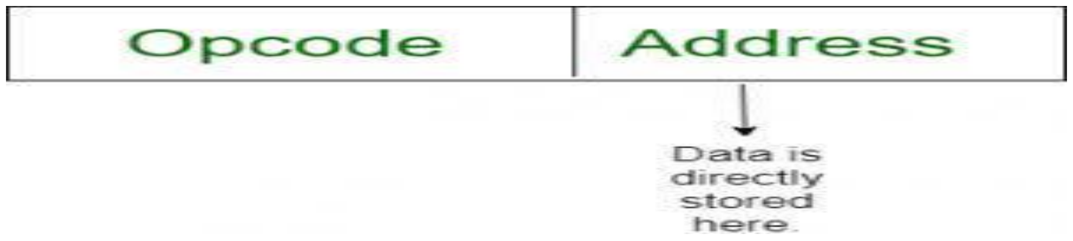
Addressing Modes– The term addressing modes refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

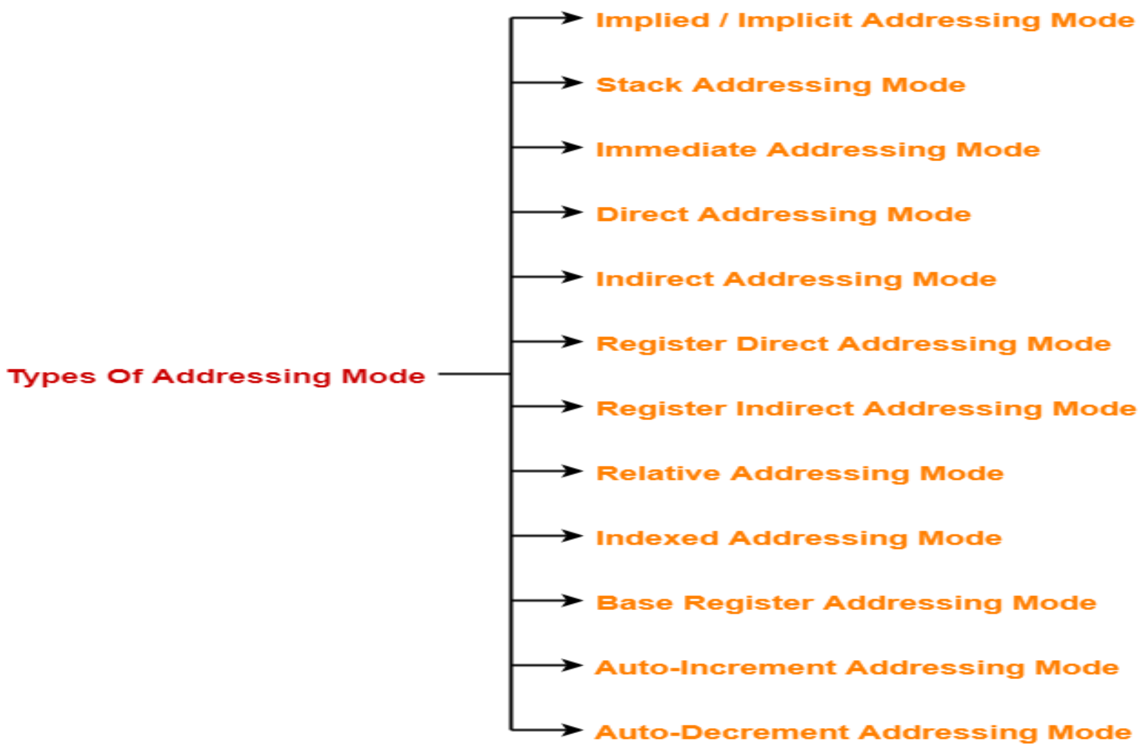
Addressing modes for 8086 instructions are divided into two categories:

1) Addressing modes for data

2) Addressing modes for branch

An assembly language program instruction consists of two parts




Types Of Addressing Mode
Implied / Implicit Addressing Mode
Stack Addressing Mode
Immediate Addressing Mode
Direct Addressing Mode
Indirect Addressing Mode
Register Direct Addressing Mode
Register Indirect Addressing Mode
Relative Addressing Mode
Indexed Addressing Mode
Base Register Addressing Mode
Auto-Increment Addressing Mode
Auto-Decrement Addressing Mode

OR

12. List the different types if instructions and explain Shift and rotate instructions.(CT1)

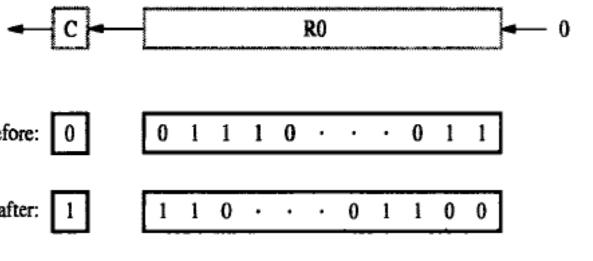
Shift and Rotate Instructions

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

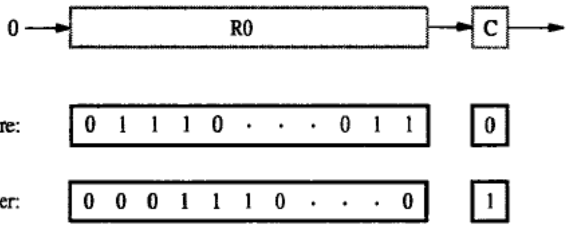
The details of how the shifts are performed depend on whether the operand is a signed number or some more general binary coded information. For general operand we use a logical shift , for number we use a arithmetic shift which preserves the sign of the number.

Logical Shift:

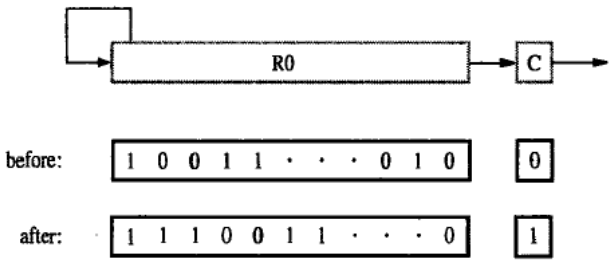
Two logical shift instructions are needed , one for shifting left(LShiftL) and another for shifting right (LShiftR).



(a) Logical shift left LShiftL #2,R0



(b) Logical shift right LShiftR #2,R0



(c) Arithmetic shift right AShiftR #2,R0

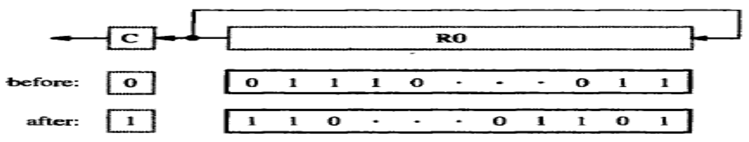
```
Move        #LOC,R0       R0 points to data.
MoveByte    (R0)+,R1      Load first byte into R1.
LShiftL     #4,R1         Shift left by 4 bit positions.
MoveByte    (R0),R2       Load second byte into R2.
And         #$F,R2        Eliminate high-order bits.
Or          R1,R2         Concatenate the BCD digits.
MoveByte    R2,PACKED     Store the result.
```

Arithmetic Shifts

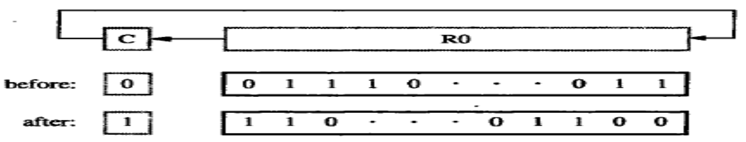
- Shifts may occur right and left.
- The requirement on right shifting distinguishes arthimetic shifts from logical in which the fill- in bits is always zero.
- Right shift – AShiftR
- Left shift – AShiftL
- The arithmetic left shift is exactly same as the logical left shift.

Rotate operation

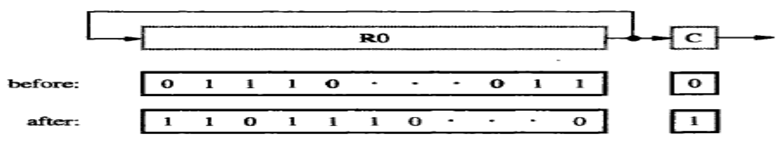
- In the shift operations, the bit shifted out of the operand are lost, except for the last bit shifted out which is retain in the carry flag C.
- To preserve all bits, a set of rotate instructions can be used.
- Two versions of both the left and the right rotate instruction are usually provided.
- The mnemonics Rotate L,Rotate LC, Rotate R, and Rotate Rc, denote the instruction that the rotate operations.



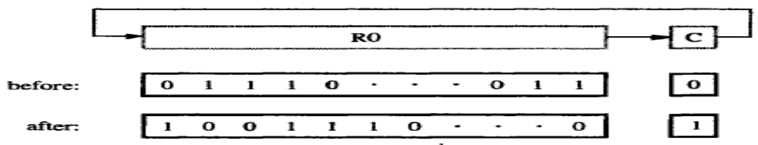
(a) Rotate left without carry RotateL #2,R0



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



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



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

UNIT II

13. Give a detailed note on Subroutine with example ALP program.(CT1)

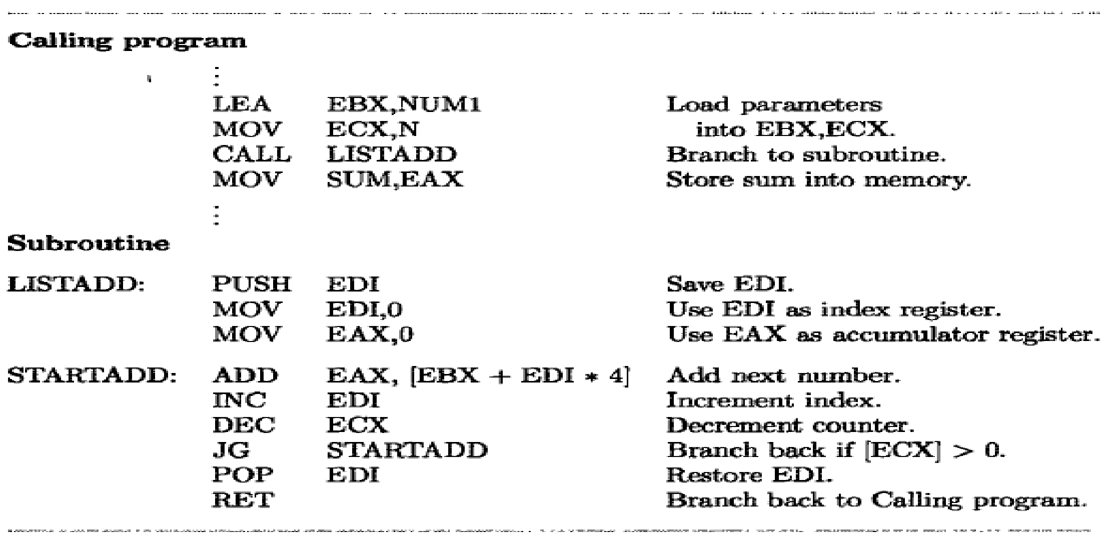
The processor stack data structure is convenient for handling entry to and return from subroutines. In the IA-32 architecture, register ESP is used as the stack pointer, which points to the current top element (TOS) in the processor stack.

There are four instructions for pushing elements onto the stack and for popping them off. The instructions are,

```
PUSH      src

POP       dst

PUSHAD

POPAD
```

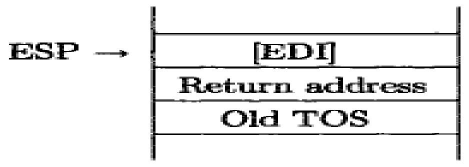
The subroutine is called by the instruction

```
CALL      LISTADD
```



```
Calling program
            :
            LEA    EBX,NUM1                Load parameters
            MOV    ECX,N                     into EBX,ECX.
            CALL   LISTADD                 Branch to subroutine.
            MOV    SUM,EAX                 Store sum into memory.
            :
Subroutine
LISTADD:    PUSH   EDI                     Save EDI.
            MOV    EDI,0                   Use EDI as index register.
            MOV    EAX,0                   Use EAX as accumulator register.

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

(a) Calling program and subroutine



(b) Stack contents after saving EDI in subroutine

OR

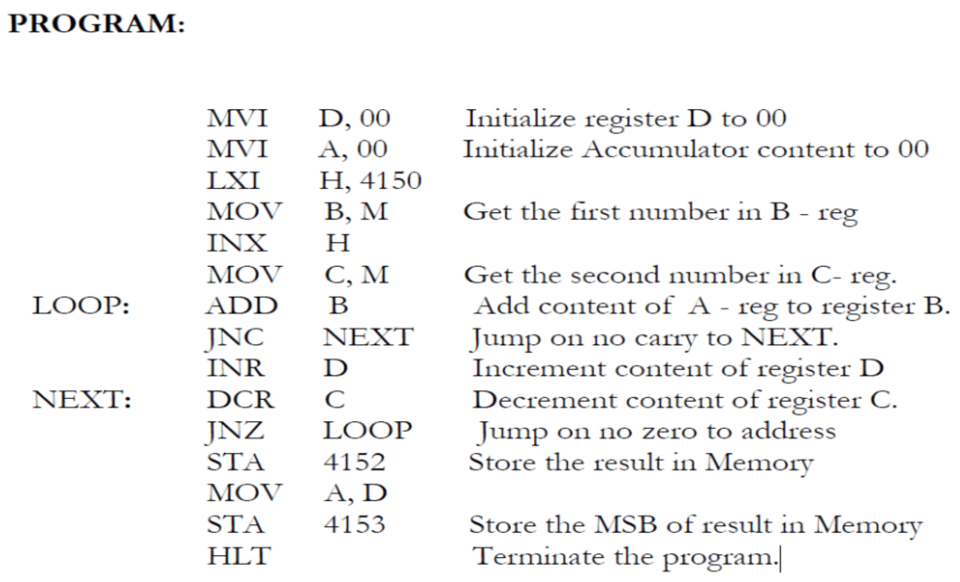
14. Write ALP program for Multiplication of two 8-bit numbers.
MULTIPLY AND DIVIDE INSRTUCTIONS

The instructions for integer multiply and divide are,

IMUL	REG,src
IMUL	src
IDIV	src(source operand)

Algorithm to Multiply Two 8 Bit Numbers

Step I : Initialize the data segment.
Step II : Get the first number in AL register.
Step III : Get the second number in BL register.
Step IV : Multiply the two numbers.
Step V : Display the result.
Step VI : Stop


PROGRAM:

```
          MVI   D, 00     Initialize register D to 00
          MVI   A, 00     Initialize Accumulator content to 00
          LXI   H, 4150
          MOV   B, M      Get the first number in B - reg
          INX   H
          MOV   C, M      Get the second number in C- reg.
LOOP:     ADD   B         Add content of  A - reg to register B.
          JNC   NEXT      Jump on no carry to NEXT.
          INR   D         Increment content of register D
NEXT:     DCR   C         Decrement content of register C.
          JNZ   LOOP      Jump on no zero to address
          STA   4152      Store the result in Memory
          MOV   A, D
          STA   4153      Store the MSB of result in Memory
          HLT             Terminate the program.
```

UNIT III

15. Explain various data transfer modes used in DMA. (CT2,M)

As we have seen earlier, the two commonly used mechanisms for implementing I/O operations are:

- Interrupts and
- Direct memory access

INTERRUPTS:

Synchronization is achieved by having the I/O device send a special signal over the bus whenever it is ready for a data transfer operation.

DIRECT MEMORY ACCESS:

Basically for high speed I/O devices, the device interface transfer data directly to or from the memory without informing the processor. When interrupts are used, additional overhead involved with saving and restoring the program counter state information. To transfer large blocks of data at high speed, an alternative approach is used. A special control unit will allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor.

DMA controller is a control circuit that performs DMA transfers, is a part of the I/O device interface. It performs functions that normally be carried out by the processor.

DMA controller must increment the memory address and keep track of the number of transfers. The operations of DMA controller must be under the control of a program executed by the processor.

To initiate the transfer of block of words, the processor sends the starting address, the number of words in the block and the direction of the transfer.

- Three registers in a DMA interface are:
    - Starting address
    - Word count
    - Status and control flag

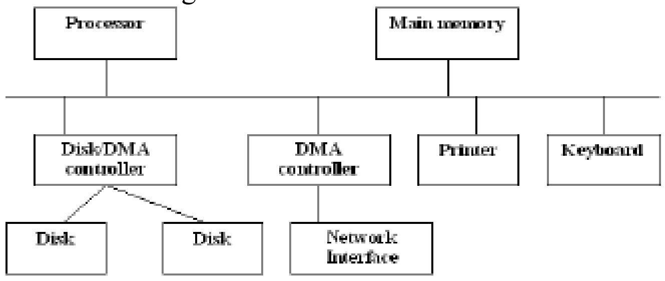


Fig 3.3 Use of DMA controllers in a computer system

BUS ARBITRATION

The device that is allowed to initiate data transfers on the bus at any given time is called the bus master. Arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it. The two approaches are centralized and distributed arbitrations.

In centralized, a single bus arbiter performs the required arbitration whereas in distributed, all device participate in the selection of the next bus master. The bus arbiter may be the processor or a separate unit connected to the bus. The processors normally the bus master unless it grants bus mastership to one of the DMAcontrollers. A simple arrangement for bus arbitration using daisy chain and distributed arbitration scheme.

OR

16. Discuss the action carried out by the processor after the occurrence of an interrupt.(CT2,M)

There are many situations where other tasks can be performed while waiting for an I/O device to become ready. A hardware signal called an Interrupt will alert the processor when an I/O device becomes ready. Interrupt-request line is usually dedicated for this purpose.

COMPUTE and PRINT routines.

One must therefore know the difference between Interrupt Vs Subroutine.

Interrupt latency is concerned with saving information in registers will increase the delay between the time an interrupt request is received and the start of execution of the interrupt-service routine.

INTERRUPT HARDWARE

Most computers have several I/O devices that can request an interrupt. A single interrupt request line may be used to serve n devices.

ENABLING AND DISABLING INTERRUPTS

All computers fundamentally should be able to enable and disable interruptions as desired. Again reconsider the COMPUTE and PRINT example.

When a device activates the interrupt-request signal, it keeps this signal activated until it learns that the processor has accepted its request.

When interrupts are enabled, the following is a typical scenario:

- The device raises an interrupt request.
- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits in the processor status register (PS).
- The device is informed that its request has been recognized and deactivates the interrupt request signal.
- The action requested by the interrupt is performed by the interrupt-service routine.
- Interrupts are enabled and execution of the interrupted program is resumed.\

HANDLING MULTIPLE DEVICES

While handling multiple devices, the issues concerned are:

- How can the processor recognize the device requesting an interrupt?
- How can the processor obtain the starting address of the appropriate routine?
- Should a device be allowed to interrupt the processor while another interrupt is being serviced?
- How should two or more simultaneous interrupt requests be handled?

VECTORED INTERRUPTS

INTERRUPT NESTING

SIMULTANEOUS REQUESTS

UNIT IV

17. Explain the virtual memory translation and TLB with the necessary diagram.(CT2)

- Virtual memory is an architectural solution to increase the effective size of the memory system.
- Recall that the addressable memory space depends on the number of address bits in a computer.
  - For example, if a computer issues 32-bit addresses, the addressable memory space is 4G bytes.

Virtual memory organization

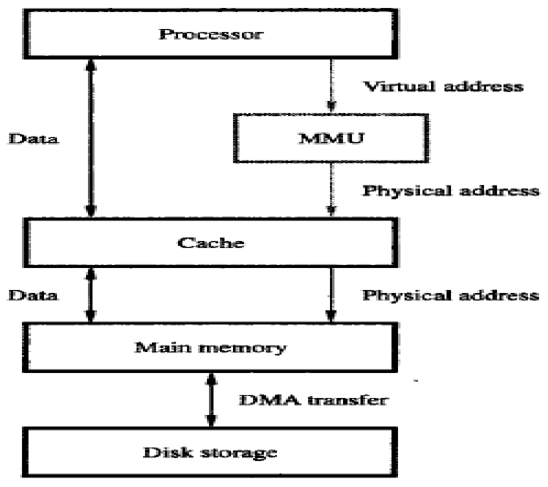


Fig 4.13 virtual memory organization

- Memory management unit (MMU) translates virtual addresses into physical addresses.

Address translation

.

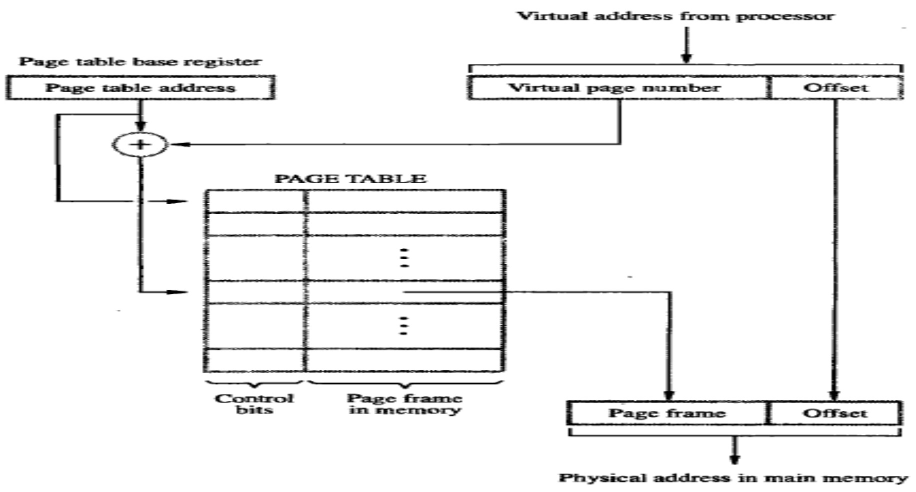


Fig 4.14 virtual memory address translation

<u>Associative-mapped TLB</u>

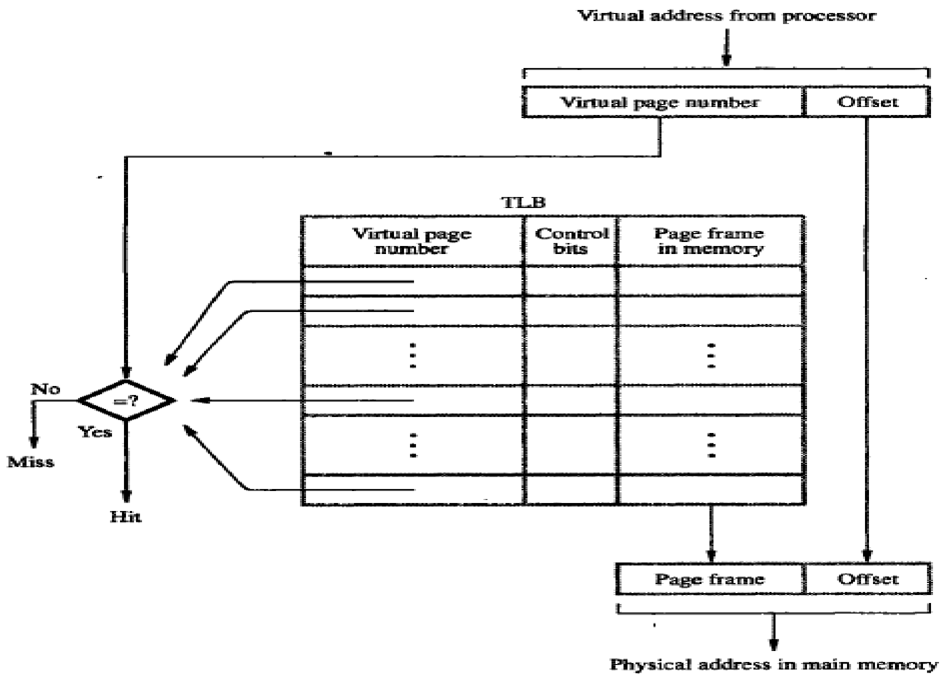


Fig 4.15 Use of an associative – mapped TLB

OR

18. Discuss any six ways of improving the cache performance.(M)

- A key design objective of a computer system is to achieve the best possible performance at the lowest possible cost.
    - Price/performance ratio is a common measure of success.
- Performance of a processor depends on:
    - How fast machine instructions can be brought into the processor for execution.
    - How fast the instructions can be executed.

Interleaving

- Divides the memory system into a number of memory modules. Each module has its own address buffer register (ABR) and data buffer register (DBR).
- Arranges addressing so that successive words in the address space are placed in different modules.
- When requests for memory access involve consecutive addresses, the access will be to different modules.
- Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.

Methods of address layouts

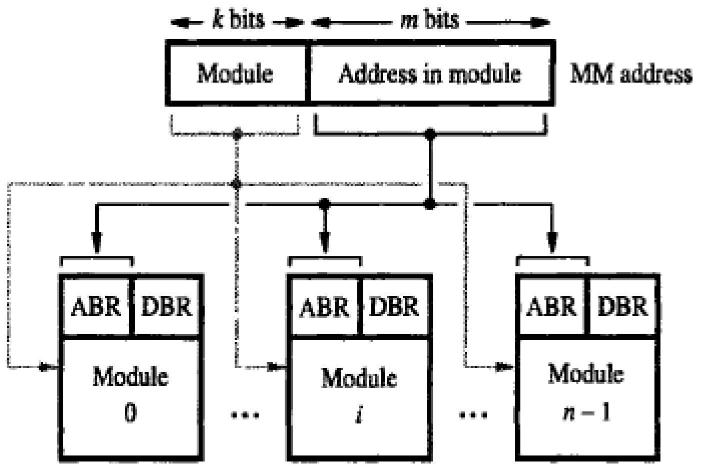


Fig 4.11 Consecutive words in a module

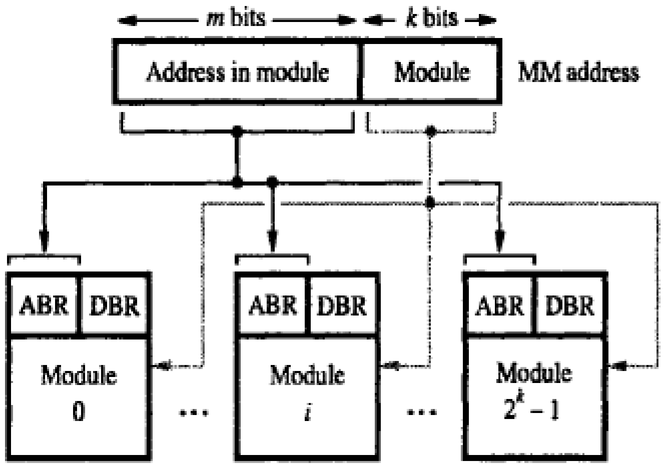


Fig 4.12 consecutive words in consecutive modules

Hit Rate and Miss Penalty

- Hit rate
- Miss penalty
- Hit rate can be improved by increasing block size, while keeping cache size constant
- Block sizes that are neither very small nor very large give best results.
- Miss penalty can be reduced if load-through approach is used when loading new blocks into cache.

Caches on the processor chip

- In high performance processors 2 levels of caches are normally used.
- Avg access time in a system with 2 levels of caches is

$T_{ave} = h1c1+(1-h1)h2c2+(1-h1)(1-h2)M$

Other Performance Enhancements

Write buffer

- Write-through
- Write-back:

Prefetching

- Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.

Lockup-Free Cache

- Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.
- A cache of this type is said to be “locked” while it services a miss.
- Cache structure which supports multiple outstanding misses is called a lockup free cache.

UNIT V

19. Explain with a neat diagram, the basic organization of a micro programmed control unit. (M)

Every instruction in a processor is implemented by a sequence of one or more sets of concurrent micro operations.

The control signals to be activated at any time are specified by a microinstruction, which is fetched from CM.

A sequence of one or more micro operations designed to control specific operation, such as addition, multiplication is called a micro program.

The micro programs for all instructions are stored in the control memo

The micro programmed control unit,

- control memory

- control address register

- Micro instruction register

- Micro program sequencer

The components of control unit work together as follows:

 the control address register holds the address of the next microinstruction to be read.

 When address is available in control address register, the sequencer issues READ command to the control memory.

 After issue of READ command, the word from the addressed location is read into the microinstruction register.

 Now the content of the micro instruction register generates control signals and next address information for the sequencer.

 The sequencer loads a new address into the control address register based on the next address information.

Advantages of Micro programmed control

Disadvantages

Microinstruction

A simple way to structure microinstructions is to assign one bit position to each control signal required in the CPU.

Grouping of control signals

Grouping technique is used to reduce the number of bits in the microinstruction.

Gating signals: IN and OUT signals

Control signals: Read, Write, clear A, Set carry in, continue operation, end, etc.

ALU signals: Add, Sub, etc;

There are 46 signals and hence each microinstruction will have 46 bits.

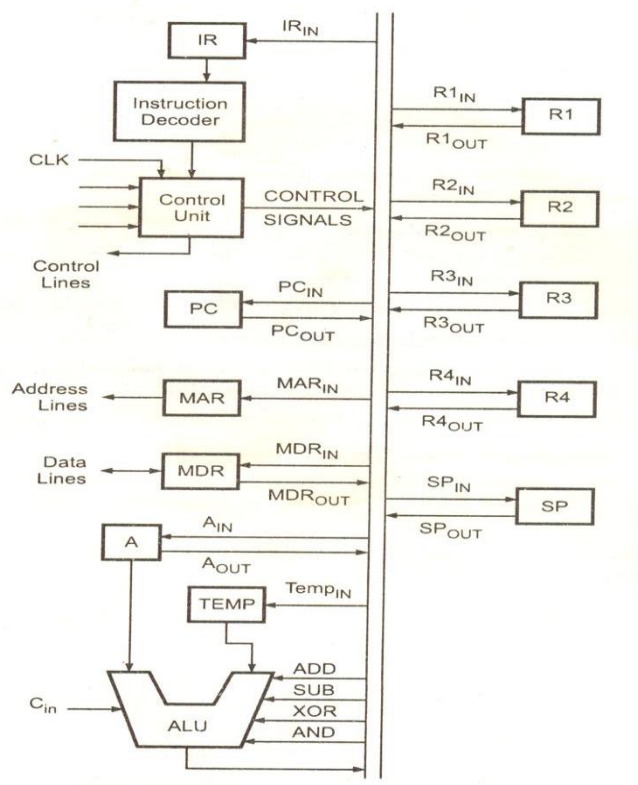
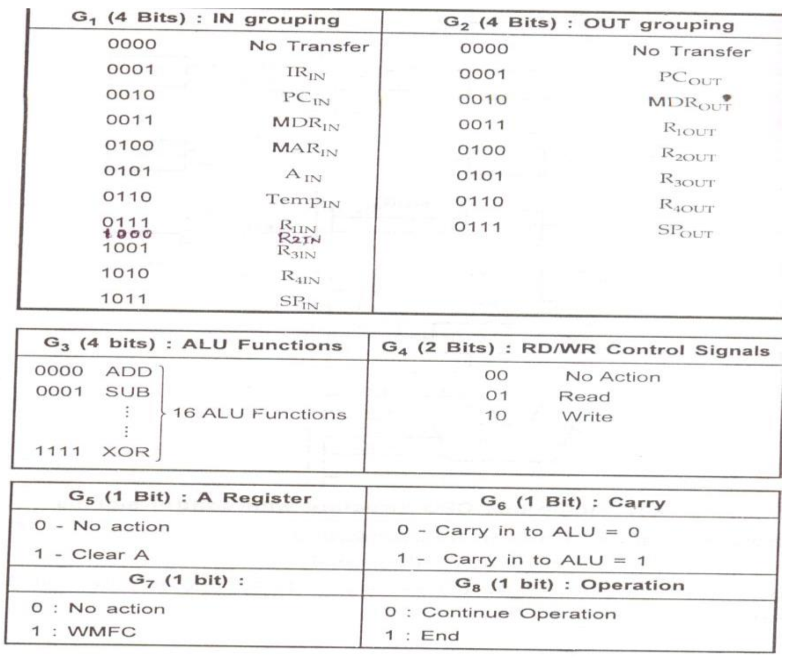


Fig. 3.16 Single bus CPU structure with control signals

46 control signals can be grouped in 7 different groups.



G$_1$ (4 Bits) : IN grouping		G$_2$ (4 Bits) : OUT grouping	
0000	No Transfer	0000	No Transfer
0001	IR$_{IN}$	0001	PC$_{OUT}$
0010	PC$_{IN}$	0010	MDR$_{OUT}$
0011	MDR$_{IN}$	0011	R$_{1OUT}$
0100	MAR$_{IN}$	0100	R$_{2OUT}$
0101	A$_{IN}$	0101	R$_{3OUT}$
0110	Temp$_{IN}$	0110	R$_{4OUT}$
0111	R$_{1IN}$	0111	SP$_{OUT}$
1000	R$_{2IN}$		
1001	R$_{3IN}$		
1010	R$_{4IN}$		
1011	SP$_{IN}$		

G$_3$ (4 bits) : ALU Functions			G$_4$ (2 Bits) : RD/WR Control Signals	
0000	ADD	16 ALU Functions	00	No Action
0001	SUB		01	Read
⋮			10	Write
1111	XOR			

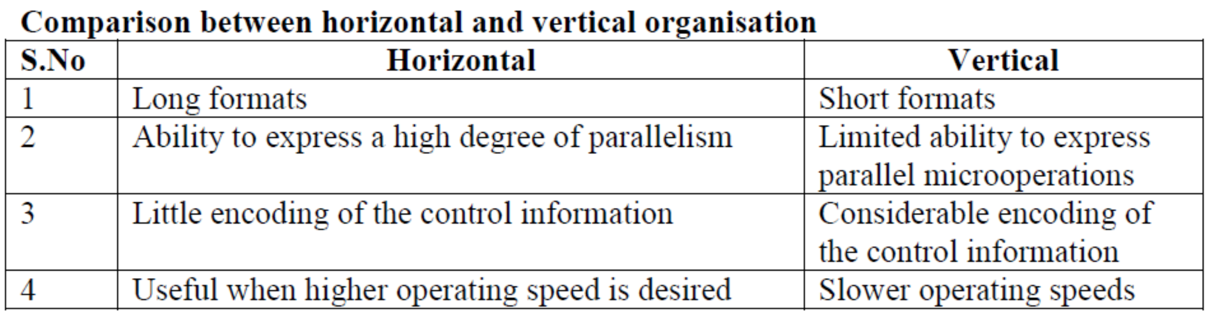
G$_5$ (1 Bit) : A Register	G$_6$ (1 Bit) : Carry
0 - No action	0 - Carry in to ALU = 0
1 - Clear A	1 - Carry in to ALU = 1
G$_7$ (1 bit) :	**G$_8$ (1 bit) : Operation**
0 : No action	0 : Continue Operation
1 : WMFC	1 : End

Vertical organization

Highly encoded scheme that can be compact codes to specify only a small number of control functions in each microinstruction are referred to as a vertical organization.

Horizontal organization

The minimally encoded scheme, in which resources can be controlled with a single instruction is called a horizontal organization.



Comparison between horizontal and vertical organisation

S.No	Horizontal	Vertical
1	Long formats	Short formats
2	Ability to express a high degree of parallelism	Limited ability to express parallel microoperations
3	Little encoding of the control information	Considerable encoding of the control information
4	Useful when higher operating speed is desired	Slower operating speeds

Micro program sequencing

The task of micro program sequencing is done by micro program sequencer.

2 important factors must be considered while designing the micro program sequencer:

a) The size of the microinstruction

b) 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, microinstruction can be executed in less time resulting better throughout.

During execution of a micro program the address of the next microinstruction to be executed has 3 sources:

i. Determined by instruction register

ii. Next sequential address

iii. Branch

Microinstructions can be shared using microinstruction branching.

Let us assume that the source operand can be specified in the following addressing modes:

a) Indexed

b) Auto increment

c) Auto decrement

d) Register indirect

e) Register direct

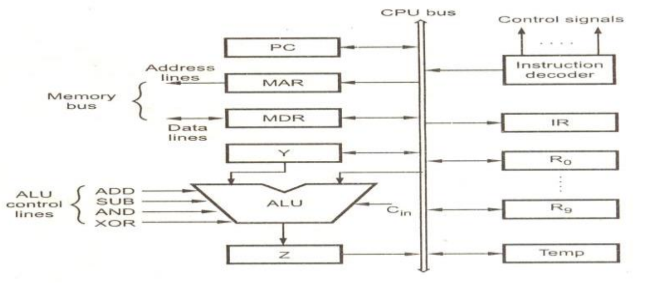


Fig. 3.17 CPU structure

Techniques for modification or generation of branch addresses

i. Bit-O-Ring

The branch address is determined by O-Ring particular bit or bits with the current address of microinstruction.

Eg: If the current address is 170 and branch address is 172 then the branch address can be generated by O-ring 02(bit 1), with the current address.

ii. Using condition variables

It is used to modify the contents CM address register directly, thus eliminating whole or in part the need for branch addresses in microinstructions.

Eg: Let the condition variable CY indicate occurrences of CY = 1, and no carry when CY = 0

iii. Wide-Branch Addressing

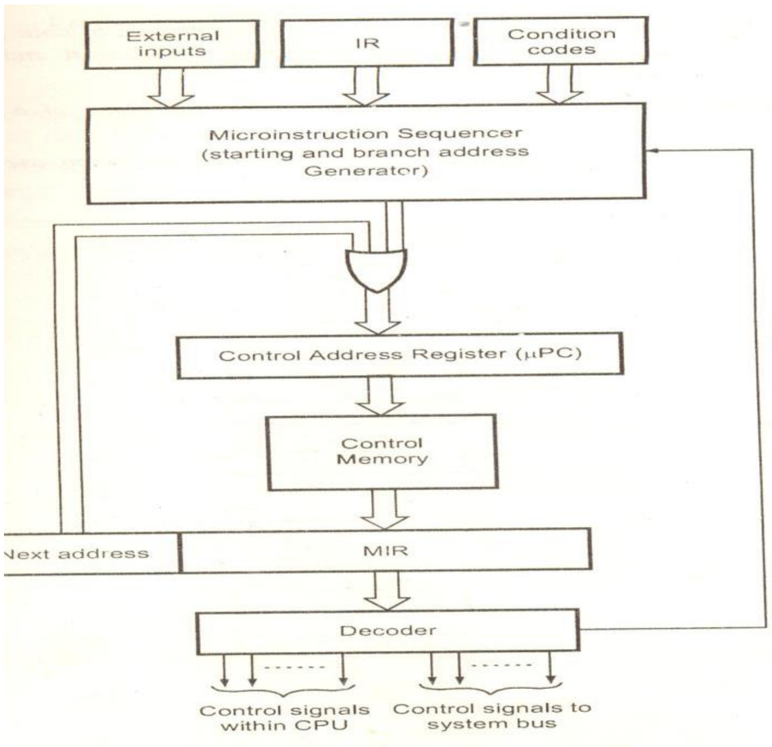
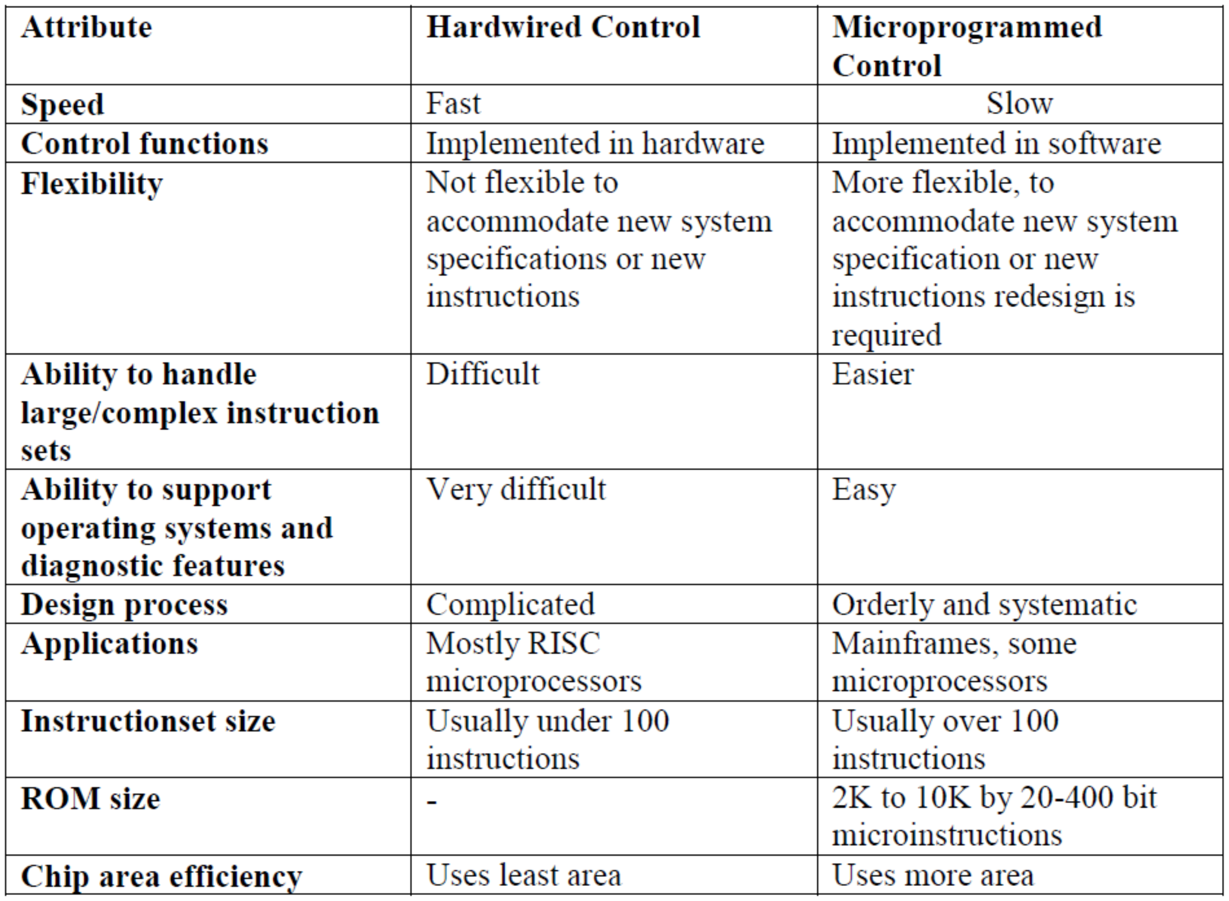

External inputs
IR
Condition codes
Microinstruction Sequencer (starting and branch address Generator)
Control Address Register (μPC)
Control Memory
Next address
MIR
Decoder
Control signals within CPU
Control signals to system bus

Fig. 3.19 Microinstruction sequencing organisation

Comparison between Hardwired and Microprogrammed Control



Attribute	Hardwired Control	Microprogrammed Control
Speed	Fast	Slow
Control functions	Implemented in hardware	Implemented in software
Flexibility	Not flexible to accommodate new system specifications or new instructions	More flexible, to accommodate new system specification or new instructions redesign is required
Ability to handle large/complex instruction sets	Difficult	Easier
Ability to support operating systems and diagnostic features	Very difficult	Easy
Design process	Complicated	Orderly and systematic
Applications	Mostly RISC microprocessors	Mainframes, some microprocessors
Instructionset size	Usually under 100 instructions	Usually over 100 instructions
ROM size	-	2K to 10K by 20-400 bit microinstructions
Chip area efficiency	Uses least area	Uses more area

OR

20. Discuss different types of hazards that occur in a pipeline.

These units must be capable of performing their tasks simultaneously and without interfering with one another.

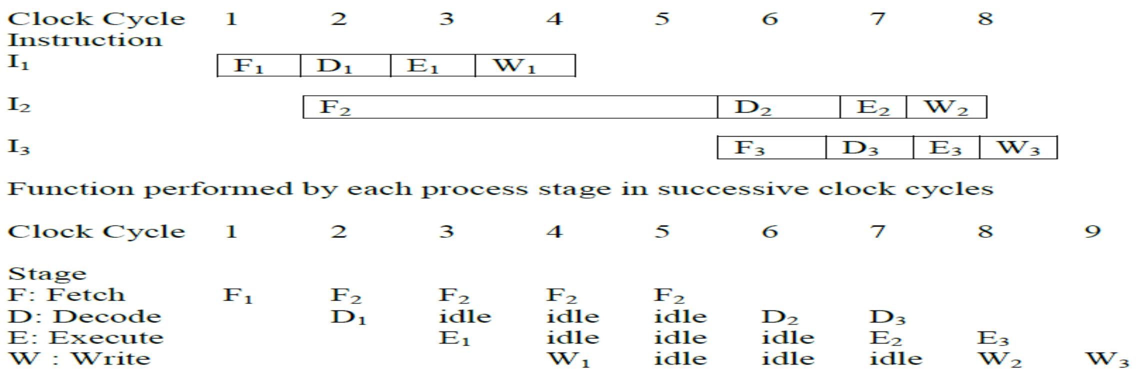
Information is passed from one unit to the next through a storage buffer.

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.

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.

Instruction execution steps in successive clock cycles:



Clock Cycle Instruction	1	2	3	4	5	6	7	8	
I_1	F_1	D_1	E_1	W_1					
I_2		F_2				D_2	E_2	W_2	
I_3						F_3	D_3	E_3	W_3

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

HAZARD

Any location that causes the pipeline to stall is called hazard.

DATA HAZARD

A data hazard is any conditions 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.

Pipeline performance

For a variety of reasons, one of the pipeline stages may not be able to complete its processing task for a given instruction I the time allotted.

Pipe Lining:

1. Basic Concepts:

Pipelining is a particularly effective way of organization concurrent activity in a computer system.

Instruction pipeline:

The fetch, decode and execute cycles for several instructions are performed simultaneously to reduce overall processing time. This process is referred to as instruction pipelining.

Consider 4 stage processes

F □ Fetch: read the instruction from the memory

D□ Decode: Decode the instruction and fetch the some operands

E□ Execute: perform the operation specified by the instruction.

W□ Write: Store the results in the destination location.