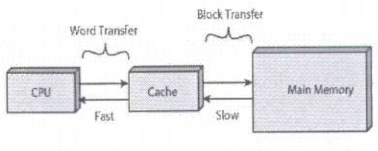
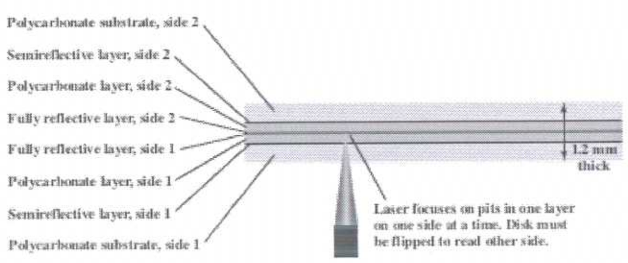
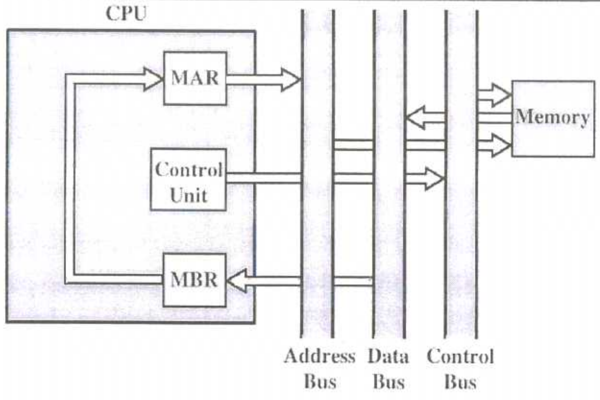
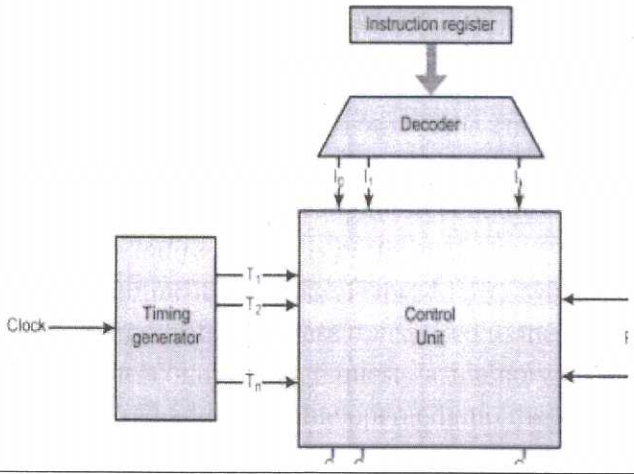
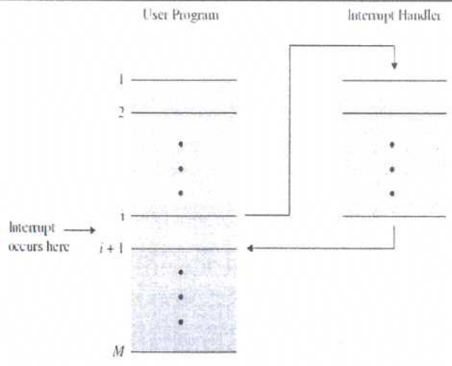
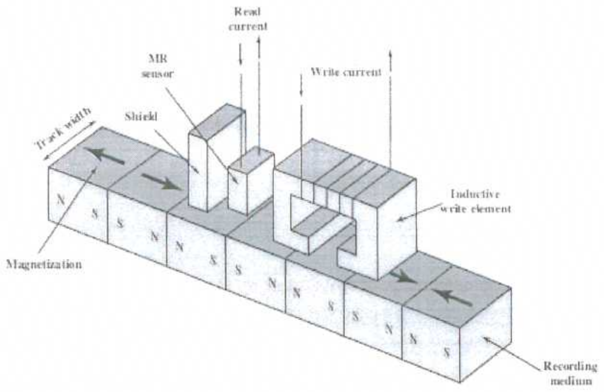
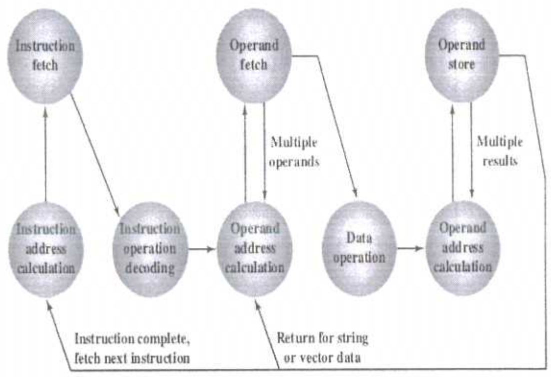
Revision: 2015 Course code: 3131 Course Title: Computer Architecture				
Qst. No	**Scoring Indicator**	**Split up score**	**Sub total**	**To ta l**
I 1)	Part A Program counter (PC) holds the address of the instruction to be fetched next. The Instruction Register (IR) holds the instruction currently being executed.	1+1	2	2
I 2)	The processor permits an I/O module to read from or write to memory, so that the I/O-memory transfer can occur without the processor interaction.This operation is known as Direct memory access (DMA)	2	2	2
I 3)	*Program Status Word (PSW)* contains condition codes plus other status information.	2	2	2
I 4)	Execution of a program consists of sequential execution of instructions. Each instruction is executed during an instruction cycle made up of shorter sub cycles(example fetch, indirect, execute, interrupt). The performance of each sub cycle involves one or more shorter operations, that is,*micro-operations.*	2	2	2
I 5)	A bus that connects major computer components (processor,memory, I/O) is called a **system bus**. It is a combination of address, data and control lines.	2	2	2
II 1)	Part B t1: MAR ← (IR(address)) t2: MBR ← Memory t3: R1 ← (R1) + (MBR) We begin with the IR containing the ADD instruction. In the first step, the address portion of the IR is loaded into the MAR. Then the referenced memory location is read. Finally, the contents of R1 and MBR are added by the ALU.	3 steps	3x2	6
2)	Small amount of fast memory. Sits between normal main memory and CPU. May be located on CPU chip or module.	Figure Explanation	2 4	6

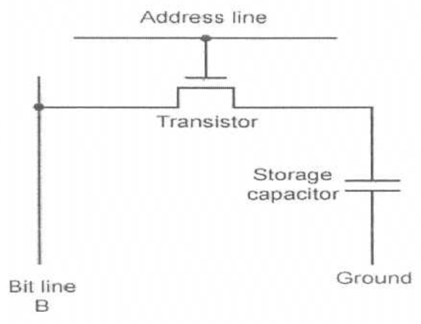
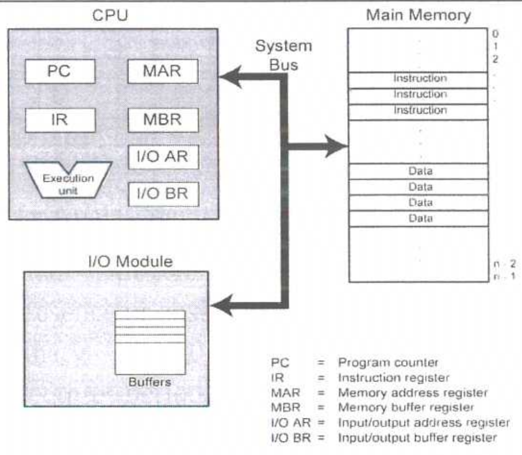
	Main memory consists of a number of fixed length blocks. Each cache line also includes control bits to indicate whether the line has been modified after loaded into the cache. When CPU requests contents of memory location check cache for this data. If present, get from cache (fast) If not present, read required block from main memory to cache. Then deliver from cache to CPU.  Word Transfer Block Transfer CPU Cache Main Memory Fast Slow (a) Single cache			
3)	**DVD**  Polycarbonate substrate, side 2 Semireflective layer, side 2 Polycarbonate layer, side 2 Fully reflective layer, side 2 Fully reflective layer, side 1 Polycarbonate layer, side 1 Semireflective layer, side 1 Polycarbonate substrate, side 1 1.2 mm thick Laser focuses on pits in one layer on one side at a time. Disk must be flipped to read other side. (b) DVD-ROM, double-sided, dual-layer—Capacity 17 GB Digital Video Disk Used to indicate a player for movies Only plays video disks Digital Versatile Disk Used to indicate a computer drive Will read computer disks and play video disks Multi-layer Very high capacity (4.7G per layer) Full length movie on single disk The DVD's greater capacity is due to three differences from CDs Bits are packed more closely on a DVD	Definition Explanation Example	2 2 2	6

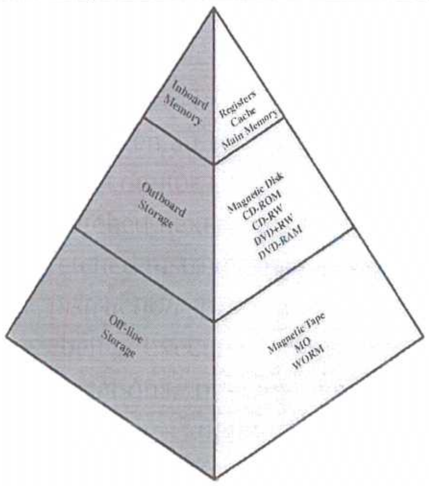
	The DVD employs a second layer of pits and lands on top of the first layer The DVD-ROM can be two sided, whereas data are recorded on only one side of a CD.			
4)	 CPU MAR Control Unit MBR Memory Address Bus Data Bus Control Bus Figure 11.8 Data Flow, Indirect Cycle Once the fetch cycle is over, the control unit examines the contents of the IR to determine if it contains an operand specifier using indirect addressing. If so, an *indirect cycle is performed.* The rightmost N bits of the MBR, which contain the address reference, are transferred to the MAR. Then the control unit requests a memory read, to get the desired address of the operand into the MBR.	Figure Explanation	3 3	6
5)	 Instruction register Decoder I_0 I_1 I_k Clock Timing generator T_1 T_2 T_n Control Unit	Figure Explanation	3 3	6

	The key inputs are the instruction register, the clock, flags, and control bus signals. In the case of the flags and control bus signals, each individual bit typically has some meaning. The control unit makes use of the op code and will perform different actions (issue a different combination of control signals) for different instructions. To simplify the control unit logic, there should be a unique logic input for each opcode. This function can be performed by a decoder, which takes an encoded input and produces a single output. In general, a decoder will have n binary inputs and 2n binary outputs. Each of the 2n different input patterns will activate a single unique output. The clock portion of the control unit issues a repetitive sequence of pulses. This is useful for measuring the duration of micro-operations. Essentially, the period of the clock pulses must be long enough to allow the propagation of signals along data paths and through processor circuitry. However, as we have seen, the control unit emits different control signals at different time units within a single instruction cycle. Thus, we would like a counter as input to the control unit, with a different control signal being used for T1, T2, and so forth. At the end of an instruction cycle, the control unit must feed back to the counter to reinitialize it at T1.			
6)	 User Program Interrupt Handler 1 2 i Interrupt occurs here i + 1 M Processor checks for interrupt indicated by an interrupt signal. If no interrupt pending, fetch next instruction If interrupt pending: Suspend execution of current program Save context*(information about the program)* Set PC to start address of interrupt handler routine. Process interrupt. Restore context and continue interrupted program.	Figure Explanation	2 4	6

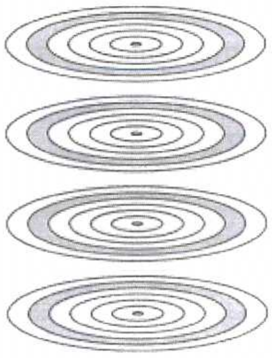
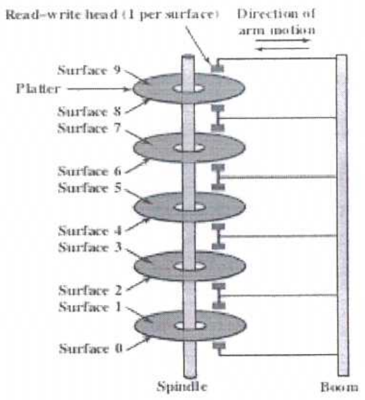
7)	 Read current MR sensor Write current Shield Track width Inductive write element Magnetization Recording medium **Read and Write Mechanisms** Data is Recorded and retrieved from the disk via conductive coil called a head. May be single read/write head or separate heads for read and write **Write mechanism** Electricity flowing through a coil produces a magnetic field. Electric pulses are sent to the write head, and the resulting magnetic patterns are recorded on the surface of the disk.Different patterns are recorded for positive and negative currents. **Traditional Read Mechanism** Magnetic field moving relative to coil produces electric current in the coil. When the surface of the disk passes under the head, it generates a current of the same polarity as the one already recorded. Coil is the same for read and write process **Contemporary Read Mechanism** Separate read head, close to write head.The read head consists of Partially shielded magneto resistive (MR) sensor. Electrical resistance depends on direction of magnetic field. By passing a current through the MR sensor, resistance changes are detected as voltage signals.	Read Write	2x3	6

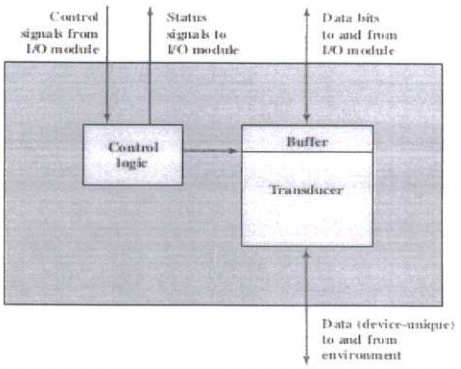
III a)	<u>Part C</u> <u>Instruction cycle state diagram</u>  Instruction fetch Operand fetch Operand store Multiple operands Multiple results Instruction address calculation Instruction operation decoding Operand address calculation Data operation Operand address calculation Instruction complete, fetch next instruction Return for string or vector data Figure 3.6 Instruction Cycle State Diagram Instruction address calculation (iac): Determine the address of the next instruction to be executed. Instruction fetch (if): Read instruction from its memory location into the processor. Instruction operation decoding (iod): Analyze instruction to determine type of operation to be performed and operand(s) to be used. Operand address calculation (oac): If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand. Operand fetch (of): Fetch the operand from memory or read it in from I/O. Data operation (do): Perform the operation indicated in the instruction. Operand store (os): Write the result into memory or out to I/O.	Figure Explanation	4 5	9

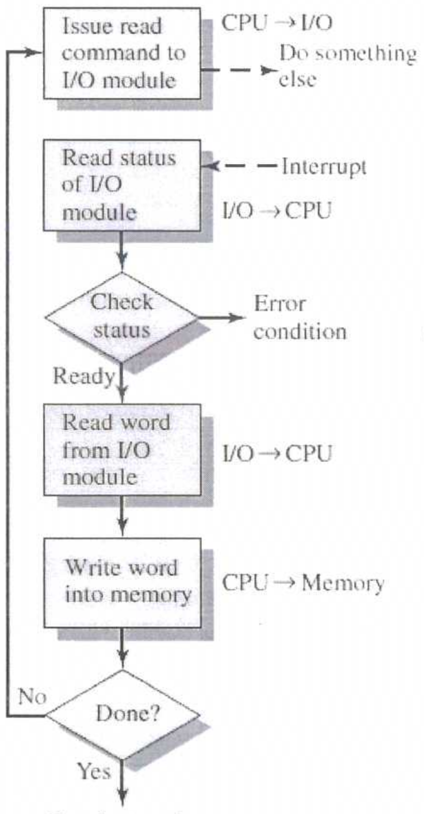
b)	 Address line Transistor Storage capacitor Bit line B Ground An Analog device Made with cells that store data as charge on capacitors Presence or absence of charge in a capacitor is interpreted as a binary 1 or 0 Requires periodic charge refreshing to maintain data storage The term *dynamic* refers to the tendency of the stored charge to leak away, even with power continuously applied	Figure Explanation	3 3	6
IV a)	 CPU PC MAR IR MBR I/O AR Execution unit I/O BR System Bus Main Memory 0 1 2 Instruction Instruction Instruction Data Data Data Data n - 2 n - 1 I/O Module Buffers PC = Program counter IR = Instruction register MAR = Memory address register MBR = Memory buffer register I/O AR = Input/output address register I/O BR = Input/output buffer register Figure 3.2 Computer Components: Top-Level View The CPU exchanges data with memory. ▪ For this purpose, it uses two internal registers A memory address register (MAR), which specifies the address in memory for the next read or write A memory buffer register (MBR), which contains the data to be written into memory or receives the data read from memory. An I/O address register (I/OAR) specifies a particular I/O device.	Figure Explanation	4 5	9

	An I/O buffer (I/OBR) register is used for the exchange of data between an I/O module and the CPU Program counter (PC) holds the address of the instruction to be fetched next The fetched instruction is loaded into instruction register (IR). An *instruction register* (IR) holds the *instruction* currently being executed.A memory module consists of a set of locations, having address in a sequential manner. Each location contains a binary number that can be an instruction or data. An I/O module transfers data from external devices to CPU and memory, and vice versa.			
b)	 Inboard Memory Registers Cache Main Memory Outboard Storage Magnetic Disk CD-ROM CD-RW DVD+RW DVD-RAM Off-line Storage Magnetic Tape MO WORM ▪ Registers —In CPU Internal or Main memory —May include one or more levels of cache —"RAM" External memory —Secondary Storage devices (hard disk, CD, DVD etc) As one goes down the hierarchy, the following occur: Decreasing cost per bit Increasing capacity Increasing access time	Figure Explanation	3 3	6

V a)	**Physical Characteristics of Magnetic Disk** 1. **Head motion** Fixed/Movable Head Disk One read write head per track Heads mounted on fixed ridged arm Movable head One read write head per side Mounted on a movable arm 2. **Disk portability** Nonremovable disk • Removable disk Can be removed from drive and replaced with another disk Provides unlimited storage capacity Easy data transfer between systems Eg : Floppy disks Non removable disk Permanently mounted in the drive 3. **Sides** single double For most disks, the magnetizable coating is applied to both sides of the platter, which is called double sided. Some less expensive disk systems use single-sided disks. 4. **Platters** Single or multiple platter	Explanation 4 points	4x1. 5	6

	One head per side Heads are joined and aligned Aligned tracks on each platter form cylinders reduces head movement Increases speed (transfer rate) 5. **Head Mechanism** Nonremovable disk Contact (floppy) Removable disk Fixed gap Aerodynamic gap (Winchester) **Tracks and Cylinders**   Read-write head (1 per surface) Direction of arm motion Surface 9 Platter Surface 8 Surface 7 Surface 6 Surface 5 Surface 4 Surface 3 Surface 2 Surface 1 Surface 0 Spindle Boom			

b)	 Control signals from I/O module Status signals to I/O module Data bits to and from I/O module Control logic Buffer Transducer Data (device-unique) to and from environment The interface to the I/O module is in the form of control, data, and status signals. Control signals determine the function that the device will perform, such as send data to the I/O module (INPUT or READ), accept data from the I/O module (OUTPUT or WRITE), report status,or perform some control function particular to the device (e.g., position a disk head). Data are in the form of a set of bits to be sent to or received from the I/O module. Status signals indicate the state of the device. Examples are READY/NOT-READY to show whether the device is ready for data transfer. Control logic associated with the device controls the device's operation in response to direction from the I/O module. The transducer converts data from electrical to other forms of energy during output and from other forms to electrical during input. Typically, a buffer is associated with the transducer to temporarily hold data being transferred between the I/O module and the external environment; a buffer size of 8 to 16 bits is common.	Lisiting phase Explanation	2 3.5x 2	9
VI a)	The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The processor, while waiting, must repeatedly interrogate the	Figure Explanation	3 3	6

	status of the I/O module. As a result, the level of the performance of the entire system is degraded. An alternative is for the processor to issue an I/O command to a module and then go on to do some other useful work. The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor. The processor then executes the data transfer, as before, and then resumes its former processing.  Issue read command to I/O module CPU → I/O Do something else Read status of I/O module Interrupt I/O → CPU Check status Error condition Ready Read word from I/O module I/O → CPU Write word into memory CPU → Memory Done? No Yes Next instruction (b) Interrupt-Driven I/O			
b)	**RAID** Redundant Array of Independent Disks The RAID scheme consists of seven levels, zero through six.	Concept and listing Each level	2 7x1	9

These levels do not have a hierarchical relationship.

RAID is a set of physical disk drives viewed by the operating system as a single logical drive

Data distributed across physical drives

Redundant disk capacity is used to store parity information, which is used to recover data in case of a disk failure

RAID 0

No redundancy

Data striped across all disks

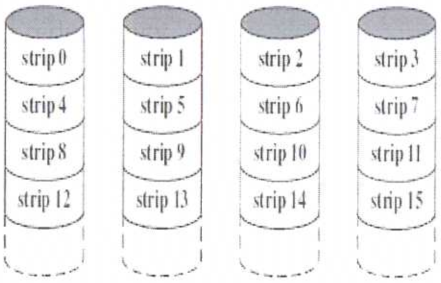
Round Robin striping

Increase speed

Multiple data requests probably not on same disk

Disks seek in parallel

The data are striped across the available disks.



(a) RAID 0 (Nonredundant)

RAID 1

In RAID 1, redundancy is achieved by duplicating all the data.

Data is striped across disks

2 copies of each stripe on separate disks (mirrored disks)

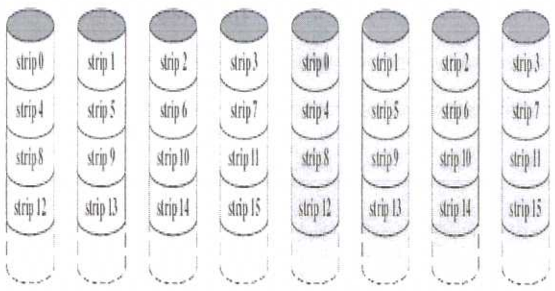
A read request can be serviced by either of the two disks that contains the requested data

A write request requires that both corresponding strips be updated, but this can be done in parallel.

Recovery

When a drive fails, the data may still be accessed from the second drive

Disadvantage of RAID 1 is the cost; it requires twice the disk space.



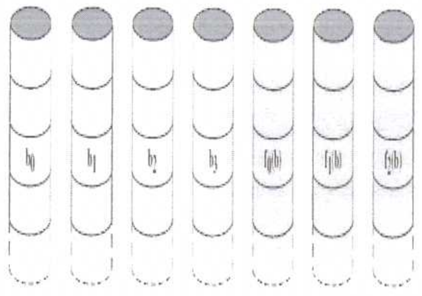
(b) RAID 1 (Mirrored)

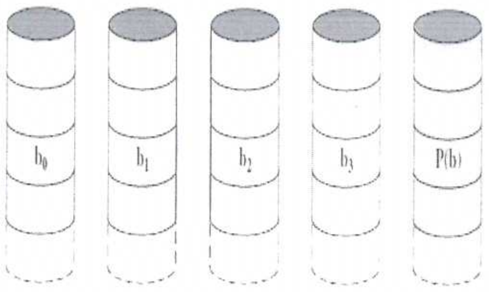
RAID 2

With RAID2, an error-correcting code is calculated across corresponding bits on each data disk

The bits of the code are stored in the corresponding bit positions on multiple parity disks.

Usually, a Hamming code is used which is able to correct single-bit errors and detect double-bit errors.

	Lots of redundancy Expensive Not used  b_0 b_1 b_2 b_3 $f_0(b)$ $f_1(b)$ $f_2(b)$ (c) RAID 2 (Redundancy through Hamming code) **RAID 3** Similar to RAID 2 Only one redundant disk parallel data access, with data distributed in small strips. Instead of an error-correcting code, a simple parity bit is computed for the set of individual bits in the same position on all of the data disks. In the event of a drive failure, the parity drive is accessed and data is reconstructed from the remaining devices. Data on failed drive can be reconstructed from surviving data and parity info Very high transfer rates			



(d) RAID 3 (Bit-interleaved parity)

RAID 4

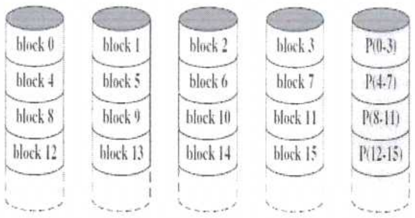
Each disk operates independently so that separate I/O requests can be satisfied in parallel

Good for high I/O request rate

Large stripes

Bit by bit parity calculated across stripes on each disk

Parity stored on parity disk



(e) RAID 4 (Block-level parity)

RAID 5

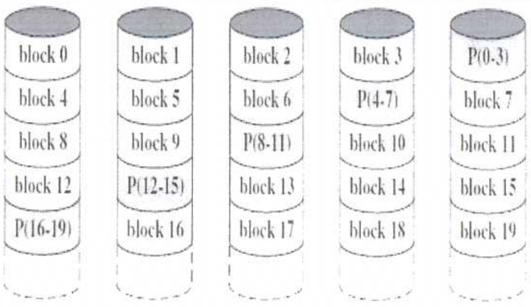
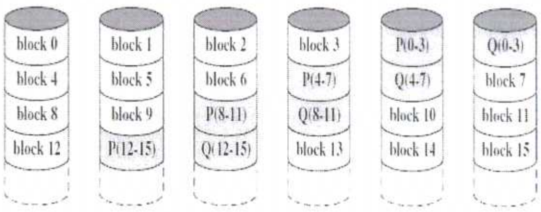
→ Like RAID 4

Parity striped across all disks

Round robin allocation for parity stripe

Avoids RAID 4 bottleneck at parity disk

Commonly used in network servers

	 block 0 block 4 block 8 block 12 P(16-19) block 1 block 5 block 9 P(12-15) block 16 block 2 block 6 P(8-11) block 13 block 17 block 3 P(4-7) block 10 block 14 block 18 P(0-3) block 7 block 11 block 15 block 19 (f) RAID 5 (Block-level distributed parity) **RAID 6** → In the RAID 6 scheme, two different parity calculations are carried out and stored in separate blocks on different disks. → User requirement of N disks needs N+2 → High data availability o Three disks need to fail for data loss o write process difficult because each write affects two parity blocks  block 0 block 4 block 8 block 12 block 1 block 5 block 9 P(12-15) block 2 block 6 P(8-11) Q(12-15) block 3 P(4-7) Q(8-11) block 13 P(0-3) Q(4-7) block 10 block 14 Q(0-3) block 7 block 11 block 15 (g) RAID 6 (Dual redundancy)			
VII a)	**User-Visible Registers** A user-visible register is one that is referenced by machine language that the processor executes. We can characterize these in the following categories: General purpose Registers Data Registers Address Registers	Definition & Listing Explanation	2 4x1	6

	Condition Code **1. General purpose Registers** General-purpose registers can be assigned to a variety of functions by the programmer. Any general-purpose register can contain the operand for any opcode. **2. Data Registers** Data registers are used to hold data **3. Address Registers** Used to hold addresses **4. Condition codes** Condition codes are bits set by the processor hardware as the result of operations An arithmetic operation may produce a positive, negative, zero, or overflow result. In addition to the result itself being stored in a register or memory, a condition code is also set.			
b)	**1. Instruction Pipelining** Processors make use of instruction pipelining to speed up execution. Pipelining involves breaking up the instruction cycle into a number of separate stages that occur in sequence, such as fetch instruction, decode instruction, determine operand addresses, fetch operands, execute instruction, and write operand result. Instructions move through these stages, as on an assembly line, so that each stage can be working on a different instruction at the same time. Instruction pipelining improves the overall system performance. Instruction pipelining is similar to the use of an assembly line in a manufacturing plant. This process is also referred to as *pipelining, because, as in a pipeline, new inputs are accepted at one end* before previously accepted inputs appear as outputs at the other end.	Listing Explanation	2 4	6

Two Stage Instruction Pipeline

1. Instruction processing is subdivided into two stages: **fetch instruction** and **execute instruction**.
   There are times during the execution of an instruction when main memory is not being accessed.
   This time could be used to fetch the next instruction in parallel with the execution of the current one.
   The pipeline has two independent stages.
   The first stage fetches an instruction and buffers it.
   When the second stage is free, the first stage passes it the buffered instruction.

   While the second stage is executing the instruction, the first stage takes advantage of any unused memory cycles to fetch and buffer the next instruction. This is called ***instruction prefetch*** *or* ***fetch overlap***.

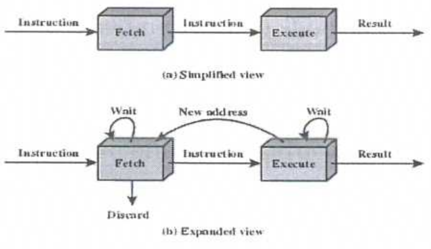


Figure 12.9 Two-Stage Instruction Pipeline

Six-stage Instruction Pipeline

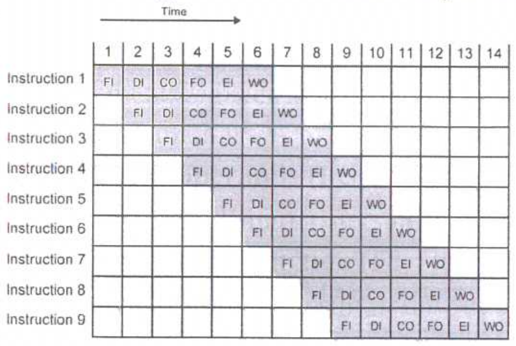
Instruction Execution is decomposed into the following stages

- **Fetch instruction (FI):** Read the next expected instruction into a buffer.
- **Decode instruction (DI):** Determine the opcode and the operand specifiers.
- **Calculate operands (CO):** Calculate the effective address of each source operand. This may involve displacement, register indirect, indirect, or other forms of address calculation.
- **Fetch operands (FO):** Fetch each operand from memory. Operands in registers need not be fetched.
- **Execute instruction (EI):** Perform the indicated operation and store the result, if any, in the specified destination operand location.

- **Write operand (WO):** Store the result in memory.

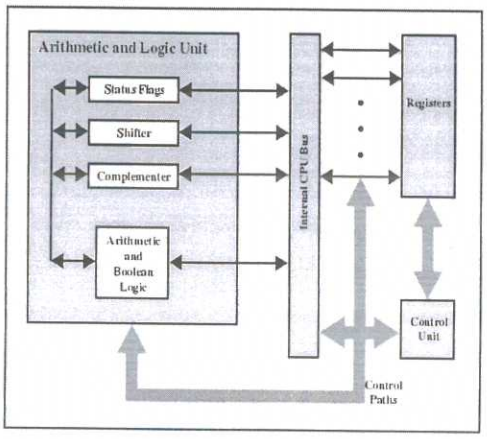
Instruction execution consists of 6 stages.

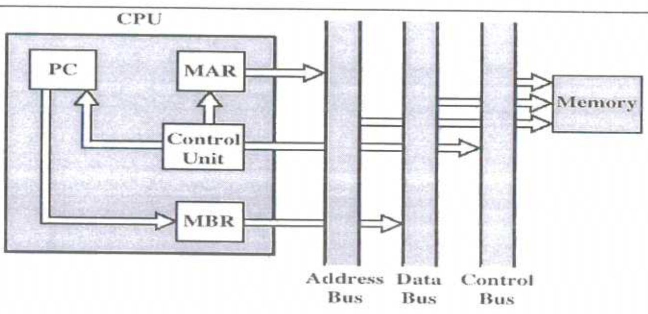
All these stages can be executed in parallel.

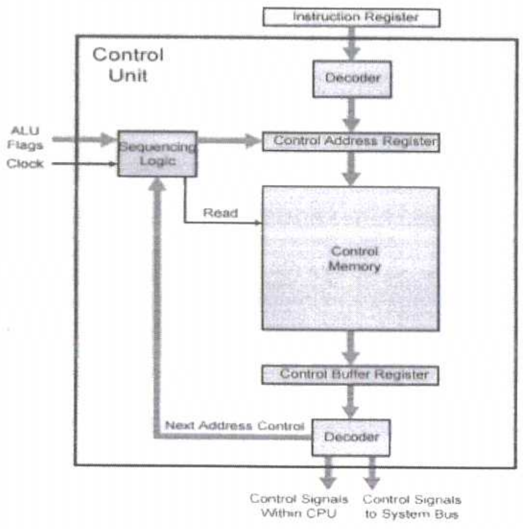


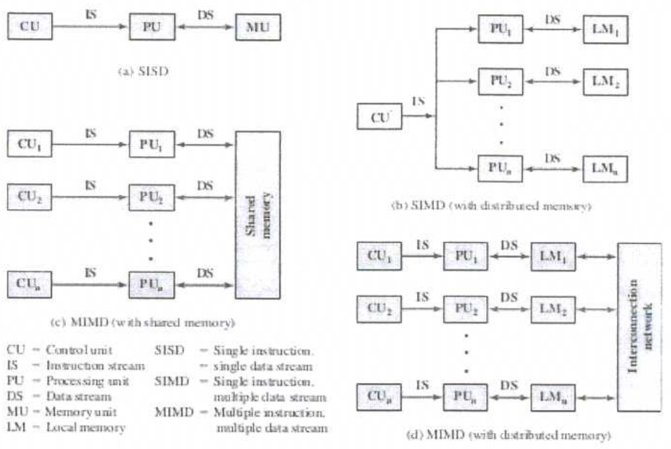
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

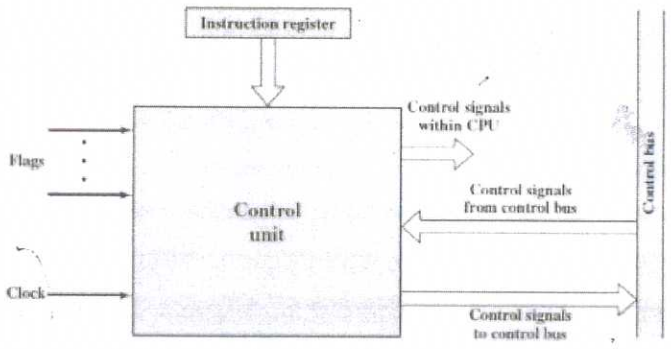
Figure 12.10 Timing Diagram for Instruction Pipeline Operation

VIII a)	 Arithmetic and Logic Unit Status Flags Shifter Complementer Arithmetic and Boolean Logic Internal CPU Bus Registers Control Unit Control Paths Figure 12.2 Internal Structure of the CPU The ALU does the actual computation or processing of data.The control unit controls the movement of data and instructions into and out of the processor and controls the operation of the ALU.The registers is used for internal storage.The data transfer and logic control paths includes an element labeled internal processor bus.This element is needed to transfer data between the various registers and the ALU.CPU must have some working space (temporary storage) called registers.	Explanation of each	4x2	8
b)	**Data Flow (Interrupt)** The current contents of the PC must be saved so that the processor can resume normal activity after the interrupt. Thus, the contents of the PC are transferred to the MBR to be written into memory. The special memory location reserved for this purpose is loaded into the MAR from the control unit. The PC is loaded with the address of the interrupt routine.	Figure Explanation	3 3	6

	 CPU PC MAR Control Unit MBR Memory Address Bus Data Bus Control Bus Figure 11.9 Data Flow, Interrupt Cycle			
IX a)	The control memory contains a program that describes the behavior of the control unit. The set of microinstructions is stored in the control memory. The control address register contains the address of the next microinstruction to be read. When a microinstruction is read from the control memory, it is transferred to a control buffer register. The left-hand portion of that register connects to the control lines emanating from the control unit. Thus, reading a microinstruction from the control memory is the same as executing that microinstruction. The third element shown in the figure is a sequencing unit that loads the control address register and issues a read command. The control unit functions as follows: 1. To execute an instruction, the sequencing logic unit issues a READ command to the control memory. 2. The word whose address is specified in the control address register is read into the control buffer register. 3. The content of the control buffer register generates control signals and next address information for the sequencing logic unit. 4. The sequencing logic unit loads a new address into the control address register based on the next-address information from the control buffer register and the ALU flags.	Figure Explanation	3 6	9

	 Instruction Register Control Unit Decoder ALU Flags Clock Sequencing Logic Control Address Register Read Control Memory Control Buffer Register Next Address Control Decoder Control Signals Within CPU Control Signals to System Bus			
b)	1. Address of next instruction is in PC 2. Address (MAR) is placed on address bus 3. Control unit issues READ command 4. Result (data from memory) appears on data bus 5. Data from data bus copied into MBR 6. PC incremented by 1 (in parallel with data fetch from memory) 7. Data (instruction) moved from MBR to IR MBR is now free for further data fetches t_1: MAR ← (PC) t_2: MBR ← Memory PC ← (PC) + I t_3: IR ← (MBR)	3 main steps	3x2	6

Xa)	 CU IS PU DS MU (a) SISD CU1 CU2 CUn PU1 PU2 PUn Shared memory (c) MIMD (with shared memory) CU – Control unit IS – Instruction stream PU – Processing unit DS – Data stream MU – Memory unit LM – Local memory SISD – Single instruction, single data stream SIMD – Single instruction, multiple data stream MIMD – Multiple instruction, multiple data stream LM1 LM2 LMn (b) SIMD (with distributed memory) Interconnection network (d) MIMD (with distributed memory) A taxonomy first introduced by Flynn is still the most common way of categorizing systems with parallel processing capability. Flynn proposed the following categories of computer systems: o Single instruction, single data (SISD) stream: o Single instruction, multiple data (SIMD) stream: o Multiple instruction, single data (MISD) stream: o Multiple instruction, multiple data (MIMD) stream: 1. Single instruction, single data (SISD) stream: A single processor executes a single instruction stream to operate on data stored in a single memory. Eg : Uniprocessors. 2. Single instruction, multiple data (SIMD) stream A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis. Each processing element has an associated data memory, so that each instruction is executed on a different set of data by the different processors. Eg : Vector and array processors.	Figure Explanation	4 4x1.5	10

	3. Multiple instruction, single data (MISD) stream A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence. This structure is not implemented. 4. Multiple instruction, multiple data (MIMD) stream A set of processors simultaneously execute different instruction sequences on different data sets. Eg : SMPs, clusters, and NUMA systems.			
b)	 Instruction register Control signals within CPU Flags Control signals from control bus Control unit Clock Control signals to control bus Control bus For the control unit to perform its function, it must have inputs that allow it to determine the state of the system and outputs that allow it to control the behavior of the system The inputs are: • Clock: This is how the control unit "keeps time." The control unit causes one micro-operation to be performed for each clock pulse. This is also referred to as the processor cycle time. • Instruction register: The opcode and addressing mode of the current instruction are used to determine which micro-operations to perform during the execute cycle. • Flags: These are needed by the control unit to determine the status of the processor and the outcome of previous ALU operations. • Control signals from control bus: The control bus portion of	Figure Explanation	2 3	5

	the system bus provides signals to the control unit. ● The outputs are: • Control signals within the processor: These are two types: those that cause data to be moved from one register to another, and those that activate specific ALU functions. • Control signals to control bus: These are also of two types: control signals to memory, and control signals to the I/O modules. ● Three types of control signals are used: • those that activate an ALU function, • those that activate a data path, • those that are signals on the external system bus or other external interface			