SPPU 2024 PATTERN

COMPUTER ORGANIZATION AND MICROPROCESSOR

JOIN WHATSAPP CHANNEL

SUGGESTION :

BEFORE STARTING TO READ THE NOTES, PURCHASE OUR MOST IMPORTANT QUESTIONS PDF FOR ONLY ₹40. MESSAGE ME ON INSTAGRAM:

https://www.instagram.com/abhyasmitra_sppu

AND SCORE EASILY 60+ MARKS IN YOUR UNIVERSITY EXAM.

ALL THE BEST!

Table of Contents

UNIT I Computer Evolution and Performance

SYLLABUS — UNIT I *(06 Hours)*

A Brief History of Computers, Von Neumann Architecture, Harvard Architecture, Designing for Performance, Evolution of Intel processor architecture - 4 bit to 64 bit, performance assessment. A top level view of Computer function and interconnection - Computer Components, Interconnection structure, bus interconnection. Computer Arithmetic - The Arithmetic and Logic Unit, addition and subtraction of signed numbers, design of adder and fast adder, carry look ahead addition, multiplication of positive numbers, signed operand multiplication, Booth's algorithm for multiplying binary integers.

1.1 Brief History of Computers (Generations)

Computers have grown in five generations. Remember the main thing used in each generation.

Generation	Years	Main Component	Example
1st	1945-55	Vacuum Tubes	ENIAC
2nd	1955-65	Transistors	IBM 1401
3rd	1965-75	Integrated Circuits (IC)	IBM 360
4th	1975-Present	Microprocessor (VLSI)	Intel 4004 onwards
5th	Present-Future	AI, Parallel Processing	Modern AI systems

Exam Tip: Generation question is very common as 2 marks. Just remember component name + year.

1.2 Von Neumann Architecture

This is the basic design used by almost all computers today. Proposed by John Von Neumann.

Main Idea

- Program (instructions) and Data are stored together in the SAME memory.
- CPU fetches instruction, then executes it, one at a time (sequential).
- This is also called Stored Program Concept.

Main Parts

1. CPU (Control Unit + ALU)
2. Memory Unit (stores both data and program)
3. Input/Output Unit
4. System Bus (connects all parts)

> ***Note:*** *Von Neumann = ONE memory for both data and instructions = ONE bus, so CPU cannot fetch data and instruction at the same time. This is called Von Neumann Bottleneck.*

1.3 Harvard Architecture

Harvard architecture solves the Von Neumann bottleneck.

- Uses SEPARATE memory for data and instructions.
- Separate buses for data and instructions, so CPU can fetch both at the SAME time.
- Faster than Von Neumann because no waiting.
- Used in: DSP processors, microcontrollers.

Von Neumann vs Harvard (Important Table)

Point	Von Neumann	Harvard
Memory	Single, shared	Separate for data & instruction
Bus	Single bus	Separate buses
Speed	Slower (bottleneck)	Faster
Design	Simple, cheap	Complex, costly
Used in	General PCs	DSP, microcontrollers

Exam Tip: This comparison table is a guaranteed 4 to 8 mark question. Learn this table by heart.

1.4 Designing for Performance

Performance of a computer depends on improving speed at every level. Main techniques used by designers:

1. Increasing clock speed of processor.
2. Increasing number of cores (parallelism).
3. Pipelining - overlapping execution of multiple instructions.
4. Improving cache memory - reduces gap between fast CPU and slow main memory.
5. Power and energy efficiency along with speed (balance needed).

> ***Note:*** *There is a 'processor-memory gap' problem - CPU speed increases faster than memory speed. Cache memory is the main solution for this gap.*

1.5 Evolution of Intel Processor (4-bit to 64-bit)

This is a very important table. Remember bit size + name + year roughly.

Processor	Bit Size	Year (approx)	Key Point
Intel 4004	4-bit	1971	First microprocessor
Intel 8008	8-bit	1972	First 8-bit processor
Intel 8085	8-bit	1976	Popular 8-bit, single power supply
Intel 8086	16-bit	1978	First 16-bit, our syllabus processor
Intel 80286	16-bit	1982	Added protected mode
Intel 80386	32-bit	1985	First 32-bit processor
Intel 80486	32-bit	1989	Built-in cache & math co-processor
Pentium series	32-bit	1993 onwards	Superscalar architecture
Itanium / Core series	64-bit	2001 onwards	64-bit, multi-core

Exam Tip: If asked 'Evolution of Intel Processor', write this table in order. Examiner checks the sequence 4-bit to 64-bit.

1.6 Performance Assessment (How Performance is Measured)

Performance is measured using these parameters:

- **Clock Speed (MHz/GHz) -** how many cycles CPU does per second.
- **CPI (Cycles per Instruction) -** average cycles needed for one instruction.
- **MIPS (Million Instructions Per Second) -** instructions executed per second.
- **FLOPS -** floating point operations per second (used for scientific computers).
- **Benchmarks -** standard test programs (like SPEC) to compare different computers fairly.

Important Formula:

CPU Time = Instruction Count × CPI × Clock Cycle Time

1.7 Computer Components and Interconnection

Basic components of a computer (also called Computer Function):

1. CPU (Central Processing Unit) - controls and processes data.
2. Main Memory - stores data and program currently in use.
3. I/O Modules - move data between computer and outside world.
4. System Interconnection (Bus) - connects CPU, Memory, I/O together.

CPU Internal Structure

- Control Unit (CU) - directs operation of CPU.
- Arithmetic Logic Unit (ALU) - performs computer's data processing.
- Registers - small, fast storage inside CPU.
- CPU Interconnection - links CU, ALU and registers.

1.8 Bus Interconnection

A bus is a shared communication path connecting two or more devices.

Types of Lines in a Bus

Bus Type	Function
Data Bus	Carries actual data between CPU, memory, I/O (bidirectional)
Address Bus	Carries memory address to be accessed (unidirectional, from CPU)
Control Bus	Carries control signals like Read, Write, Clock, Interrupt

> ***Note:*** *Width of Address Bus decides maximum memory that CPU can access. Example: 20-bit address bus = 2^20 = 1 MB memory (used in 8086).*

Bus Design Issues

- Bus width (more width = more data/address at once)

- Type of transfer (Read, Write, Read-Modify-Write)
- Timing (Synchronous - common clock, Asynchronous - no common clock)
- Bus arbitration (deciding which device uses bus when many devices want it)

1.9 Computer Arithmetic

1.9.1 Arithmetic and Logic Unit (ALU)

ALU is the part of CPU that does all calculations (add, subtract, AND, OR, etc.) It takes input from registers, processes, and gives output back to registers.

1.9.2 Addition and Subtraction of Signed Numbers

Computers use 2's Complement method for signed numbers because it makes subtraction same as addition.

- To find 2's complement: invert all bits (1's complement) and add 1.
- Addition: simply add both numbers in 2's complement form.
- Subtraction: take 2's complement of the number to be subtracted, then add.
- Overflow occurs when result does not fit in the given number of bits (carry into sign bit differs from carry out).

> ***Note:*** *Rule for Overflow: If two numbers of SAME sign are added and result has DIFFERENT sign, overflow has occurred.*

1.9.3 Design of Adder

Half Adder:

- Adds 2 single bits. Has 2 inputs (A, B) and 2 outputs (Sum, Carry).
- Sum = A XOR B, Carry = A AND B

Full Adder:

- Adds 3 bits (A, B and Carry-in). Used to build multi-bit adders.
- Sum = A XOR B XOR Cin
- Carry-out = AB + BCin + ACin

Ripple Carry Adder:

- Made by connecting many full adders in series.
- Carry of one stage 'ripples' to next stage - this makes it SLOW for large bits.

1.9.4 Fast Adder / Carry Look Ahead Adder (CLA)

Ripple carry adder is slow because each stage waits for carry from previous stage. Carry Look Ahead Adder solves this.

- It calculates all carries IN ADVANCE using extra logic circuits, before final sum is calculated.
- Two important terms:
  - Carry Generate (Gi) = Ai AND Bi (this stage generates a carry on its own)

    - Carry Propagate (Pi) = Ai XOR Bi (this stage passes carry forward)
- Formula: Carry-out (Ci+1) = Gi + (Pi . Ci)
- Because carries are calculated in parallel (not one by one), CLA is much FASTER than ripple carry adder.

Exam Tip: CLA adder is a popular 4-6 mark theory question - just remember Generate, Propagate, and that it removes the carry delay problem.

1.9.5 Multiplication of Positive (Unsigned) Numbers

Done like normal pen-paper multiplication but in binary, using shift and add method:

1. Check LSB of multiplier.
2. If bit is 1, add multiplicand to partial product.
3. If bit is 0, add 0.
4. Shift partial product right by 1 bit.
5. Repeat for all bits of multiplier.

1.9.6 Signed Operand Multiplication

When numbers can be negative, we cannot directly use the simple shift-add method, because sign bit creates problems. Solution: use Booth's Algorithm.

1.9.7 Booth's Algorithm (VERY IMPORTANT)

Booth's Algorithm is used to multiply two signed binary numbers (in 2's complement form) using ADD, SUBTRACT, and SHIFT operations.

Steps of Booth's Algorithm

1. Take Multiplicand (M), Multiplier (Q), and one extra bit Q-1 = 0 initially. Set Accumulator (A) = 0.
2. Check last 2 bits: current Q0 and Q-1.
3. If Q0 Q-1 = 00 or 11 → only do Arithmetic Right Shift (ASR) of A, Q, Q-1 together.
4. If Q0 Q-1 = 01 → A = A + M, then do ASR.
5. If Q0 Q-1 = 10 → A = A - M (i.e. add 2's complement of M), then do ASR.
6. Repeat this process for total number of bits in multiplier (n times).
7. Final Answer = combination of A and Q registers (this is the product).

Q0	Q-1	Operation
0	0	Shift only (ASR)
0	1	A = A + M, then shift
1	0	A = A - M, then shift
1	1	Shift only (ASR)

> ***Note:*** *Booth's algorithm reduces number of additions when there are continuous 1's or 0's in the multiplier, making multiplication faster.*

Exam Tip: Booth's Algorithm NUMERICAL is asked almost every year for 8-10 marks. Practice at least 2-3 numericals before exam. Steps to always show: Initial values table, then row by row A, Q, Q-1 and operation performed, then final answer.

UNIT II Memory Management

SYLLABUS — UNIT II *(06 Hours)*

Characteristics of Memory System, The memory hierarchy. Cache Memory - Cache memory principles, Elements of cache design - cache address, size, and mapping functions. Replacement algorithms, write policy, line size, number of cache, one level and two level cache. Performance characteristics of two level cache - locality & operations. Internal Memory - semiconductor main memory, advanced DRAM organization. External Memory - Hard Disk organization. RAID - level 1 to level 6.

2.1 Characteristics of Memory System

Every memory device can be classified by these characteristics:

Characteristic	Meaning
Location	Inside CPU, Internal (main memory) or External (disk)
Capacity	Total amount of data it can store
Unit of Transfer	Number of bits read/written at a time
Access Method	Sequential, Direct, Random, or Associative
Performance	Access time, Cycle time, Transfer rate
Physical Type	Semiconductor (RAM/ROM), Magnetic (disk), Optical (CD)
Permanence	Volatile (loses data on power off) or Non-Volatile

2.2 Memory Hierarchy

No single memory can be fast, large, AND cheap at the same time. So memory is arranged in levels - fastest & smallest on top, slowest & biggest at the bottom.

- Registers (fastest, smallest, costliest) - top
- Cache Memory (L1, L2, L3)
- Main Memory (RAM)
- Secondary/External Memory (Hard Disk, SSD)
- Magnetic Tape / Cloud Storage (slowest, cheapest, largest) - bottom

> ***Note:*** *Golden Rule of Memory Hierarchy: As we go DOWN → Speed decreases, Cost per bit decreases, Capacity increases.*

Exam Tip: 'Draw and explain Memory Hierarchy' is a guaranteed question. Just draw the pyramid diagram with these 5 levels and mention the golden rule.

2.3 Cache Memory Principles

Cache is a very small, very fast memory placed BETWEEN CPU and Main Memory. It stores copies of frequently used data so CPU does not have to go to slow main memory every time.

Cache works on the Principle of Locality:

- **Temporal Locality -** if data is used now, it will likely be used again SOON.
- **Spatial Locality -** if a memory location is used, NEARBY locations will likely be used soon.

How Cache Works (Hit and Miss)

- Cache Hit: CPU finds the required data in cache → fast access.
- Cache Miss: data not found in cache → CPU must fetch from main memory (slow), and this data is then stored in cache for future use.

Hit Ratio = (Number of Hits) / (Total Memory Accesses)

2.4 Elements of Cache Design

2.4.1 Cache Addresses and Size

- Cache size is much smaller than main memory but gives big performance improvement.
- Bigger cache = more hits, but also more cost and slightly more access time.

2.4.2 Mapping Functions (VERY IMPORTANT)

Mapping decides WHERE a main memory block will be placed inside the cache. There are 3 types:

Type	How it works	Advantage	Disadvantage
Direct Mapping	Each memory block maps to exactly ONE fixed cache line	Simple, cheap, fast	More misses (conflict problem)
Associative Mapping	Memory block can go to ANY cache line	Flexible, less misses	Needs complex search (costly hardware)
Set-Associative Mapping	Cache divided into sets; block maps to a SET, can go anywhere inside that set	Best balance of speed & flexibility	More complex than direct mapping

Exam Tip: Draw a simple diagram for each mapping type showing memory blocks going into cache lines/sets. This is asked very frequently for 8 marks.

2.4.3 Replacement Algorithms

Used in Associative and Set-Associative mapping when cache is full and a new block must replace an old one.

- **LRU (Least Recently Used) -** removes the block not used for the longest time. Most popular and best performing.
- **FIFO (First In First Out) -** removes the oldest block that came in first.
- **LFU (Least Frequently Used) -** removes block used least number of times.
- **Random -** removes any block randomly. Simple but not efficient.

2.4.4 Write Policy

Decides how cache handles a WRITE operation (when CPU updates data):

- **Write Through -** data is written to BOTH cache and main memory at the same time. Safe but slower.

- **Write Back -** data is written ONLY to cache first; main memory is updated later (only when block is replaced). Faster but riskier if power fails.

2.4.5 Line Size and Number of Caches

- Line Size: amount of data moved between cache and main memory in one go. Bigger line size helps spatial locality but wastes space if data not used.
- One Level Cache: single cache between CPU and main memory.
- Two (Multiple) Level Cache: L1 cache (very small, very fast, inside CPU) and L2 cache (bigger, slightly slower, outside CPU core) work together. L1 checked first, then L2, then main memory.

2.5 Performance of Two-Level Cache

Two level cache improves performance because:

- L1 handles most frequent/fast requests (very low access time).
- L2 backs up L1, reducing the number of times CPU needs to go to slow main memory.
- Overall hit ratio increases, average access time decreases.
- Works because of Locality of Reference (temporal + spatial) explained above.

2.6 Internal Memory

2.6.1 Semiconductor Main Memory Types

Type	Full Form / Meaning	Key Feature
RAM	Random Access Memory	Volatile, read/write, used as main memory
DRAM	Dynamic RAM	Uses capacitor, needs refreshing, cheaper, used as main RAM
SRAM	Static RAM	Uses transistors (flip-flop), no refresh needed, faster, costly - used in cache
ROM	Read Only Memory	Non-volatile, stores fixed data/firmware
PROM/EPROM/EEPROM	Programmable variants of ROM	Can be programmed once or multiple times

Exam Tip: SRAM vs DRAM comparison is a classic question. Remember: SRAM = Static = cache = fast = costly. DRAM = Dynamic = main memory = needs refresh = cheap.

2.6.2 Advanced DRAM Organization

- **SDRAM (Synchronous DRAM) -** works synchronized with system clock, faster than normal DRAM.
- **DDR SDRAM (Double Data Rate) -** transfers data on both rising and falling edge of clock, doubling speed (DDR2, DDR3, DDR4, DDR5 are improved versions).
- **RDRAM (Rambus DRAM) -** uses high speed bus for very fast data transfer.
- **CDRAM (Cache DRAM) -** combines small SRAM cache with DRAM on same chip.

2.7 External Memory - Hard Disk Organization

A hard disk is made of round metal/glass plates called platters, coated with magnetic material.

Term	Meaning
Platter	Circular disk where data is stored magnetically
Track	A circular ring on the platter surface
Sector	Small division of a track, smallest storage unit
Cylinder	Set of tracks at the same position on all platters stacked together
Read/Write Head	Moves over platter to read or write data
Seek Time	Time taken to move head to correct track
Rotational Latency	Time taken for disk to rotate to the correct sector
Access Time	Seek Time + Rotational Latency + Transfer Time

2.8 RAID Levels (Level 1 to Level 6)

RAID = Redundant Array of Independent Disks. Multiple disks are combined to work as one unit, to improve speed and/or reliability.

RAID Level	Technique	Main Benefit
RAID 0	Striping (data split across disks, no redundancy)	Fastest speed, but NO fault tolerance
RAID 1	Mirroring (exact copy on 2 disks)	Full data redundancy, very safe
RAID 2	Striping with Hamming code for error correction	Error correction (rarely used today)
RAID 3	Striping with 1 dedicated parity disk (byte level)	Good for large sequential data
RAID 4	Striping with 1 dedicated parity disk (block level)	Better than RAID 3 for small data requests
RAID 5	Striping with parity distributed across ALL disks	Most popular, good balance of speed + safety
RAID 6	Like RAID 5 but with DOUBLE distributed parity	Can survive failure of 2 disks at once

> ***Note:*** *Memory trick: RAID 0 = sPeed (no protection). RAID 1 = Mirror (copy). RAID 5 = most commonly used in real servers (single parity). RAID 6 = RAID 5 + extra safety (double parity).*

Exam Tip: RAID levels table is asked for 8-10 marks almost every exam. Learn RAID 0, 1, 5, 6 very well — these are most important. 2,3,4 just need one line each.

UNIT III Introduction to 8086 Microprocessor

SYLLABUS — UNIT III *(06 Hours)*

8086 Architecture: Introduction to 16 bit microprocessor, Architecture and Pin diagram of 8086, Programmers model of 8086 (Registers). Addressing modes of 8086: Immediate Addressing, Register Addressing, Direct Addressing, Indirect Addressing, Indexed Addressing, Based Addressing, Based Indexed Addressing. Instruction set of 8086: Data Movement Instructions, Arithmetic Instructions, Logic Instructions, Control Transfer Instructions, String Instructions, Input/Output Instructions, Flag Control Instructions, Process Control Instructions, Other Instructions.

3.1 Introduction to 8086 (16-bit Microprocessor)

8086 is a 16-bit microprocessor made by Intel in 1978.

Feature	Value
Data Bus	16-bit
Address Bus	20-bit
Maximum Memory	2^20 = 1 MB (1,048,576 locations)
Clock Speed	5 MHz, 8 MHz, 10 MHz versions
Number of Pins	40 pins (DIP package)
Registers	14 registers, all 16-bit

Exam Tip: 8086 features table is a direct 2-4 mark question. Memorize: 16-bit data bus, 20-bit address bus, 1 MB memory.

3.2 Architecture of 8086

8086 internal architecture is divided into TWO independent units that work together. This is the most important diagram of this unit.

3.2.1 Bus Interface Unit (BIU)

Job of BIU: handles all communication with outside world like memory and I/O ports.

- Calculates physical address (using segment registers).
- Fetches instructions from memory and sends to instruction queue.
- Contains: Segment Registers, Instruction Pointer (IP), and Instruction Queue.

3.2.2 Execution Unit (EU)

Job of EU: decodes and executes the instructions.

- Contains ALU, Control Unit (CU), General Purpose Registers, and Flag Register.
- Tells BIU where to fetch/write data from memory.

> ***Note:** Because BIU fetches NEXT instruction while EU executes CURRENT instruction, this overlapping is called Pipelining, and it makes 8086 faster than older processors.*

Exam Tip: 'Draw and explain architecture of 8086' is the MOST asked question in this subject (10 marks). Draw the block diagram clearly showing BIU and EU separately with all registers labelled.

3.3 Pin Diagram of 8086 (Important Pins)

8086 has 40 pins. Important pins to remember:

Pin/Signal	Meaning
AD0-AD15	Address/Data bus (time multiplexed/shared)
A16-A19	Address lines (upper address bits)
RD	Read control signal
WR (or /WR)	Write control signal
READY	Tells CPU that memory/IO is ready for data transfer
RESET	Resets the processor
CLK	Clock input signal
INTR / INTA	Interrupt Request / Interrupt Acknowledge
MN/MX	Selects Minimum or Maximum mode of operation

3.4 Programmer's Model of 8086 (Registers)

8086 has 14 registers, all 16-bit. They are divided into groups:

3.4.1 General Purpose Registers

Register	Full Form / Use	Can split into
AX	Accumulator - used for arithmetic, I/O	AH (high byte), AL (low byte)
BX	Base register - used to hold address	BH, BL
CX	Count register - used as counter in loops	CH, CL
DX	Data register - used in multiply/divide, I/O	DH, DL

3.4.2 Segment Registers (4 registers, each 16-bit)

- **CS (Code Segment) -** holds base address of code/program.
- **DS (Data Segment) -** holds base address of data.
- **SS (Stack Segment) -** holds base address of stack.
- **ES (Extra Segment) -** extra segment, used for string operations.

3.4.3 Pointer and Index Registers

- **IP (Instruction Pointer) -** holds offset address of next instruction to execute.
- **SP (Stack Pointer) -** holds offset address of top of stack.
- **BP (Base Pointer) -** used to access data in stack.
- **SI (Source Index) -** used in string/array operations, holds source offset.
- **DI (Destination Index) -** used in string/array operations, holds destination offset.

3.4.4 Flag Register (9 flags used)

Flag	Name	Meaning
CF	Carry Flag	Set when there is a carry/borrow out of MSB
ZF	Zero Flag	Set when result of operation is zero
SF	Sign Flag	Set when result is negative
OF	Overflow Flag	Set when signed result overflows
PF	Parity Flag	Set when result has even number of 1 bits
AF	Auxiliary Carry Flag	Set on carry from lower nibble (used in BCD)
TF	Trap Flag	Used for single-step debugging
IF	Interrupt Flag	Set/clear to enable/disable interrupts
DF	Direction Flag	Decides direction for string operations

Exam Tip: 'Draw register set of 8086' and 'Explain flag register' are both common 8-mark questions.

3.5 Physical Address Calculation

8086 has 16-bit registers but 20-bit address bus, so address is calculated as:

Physical Address = (Segment Register value × 10H) + Offset

Example: if CS = 2000H and IP = 0500H, then Physical Address = 20000H + 0500H = 20500H.

3.6 Addressing Modes of 8086

Addressing mode means HOW the operand (data) is specified/located in an instruction. 8086 has these types:

Addressing Mode	Description	Example
Immediate	Data value is given directly in instruction	MOV AL, 05H
Register	Data is in a register	MOV AX, BX
Direct	Memory address is given directly	MOV AX, [1500H]
Register Indirect	Address of data is held inside a register	MOV AX, [BX]
Indexed	Address = Index register + displacement	MOV AX, [SI+04H]
Based	Address = Base register + displacement	MOV AX, [BX+04H]
Based Indexed	Address = Base register + Index register + displacement	MOV AX, [BX+SI+04H]

Exam Tip: All 7 addressing modes with one example each is a 100% guaranteed question (8 marks). Learn the examples exactly, examiners check the syntax.

3.7 Instruction Set of 8086

Instructions are grouped into categories. Learn at least 3-4 examples from each group.

3.7.1 Data Movement Instructions

Instruction	Use
MOV	Move/copy data from source to destination
PUSH / POP	Push data to stack / pop data from stack
IN / OUT	Move data between I/O port and accumulator
XCHG	Exchange data between two locations
LEA	Load effective address into a register

3.7.2 Arithmetic Instructions

Instruction	Use
ADD / SUB	Addition / Subtraction
MUL / IMUL	Multiplication (unsigned / signed)
DIV / IDIV	Division (unsigned / signed)
INC / DEC	Increment by 1 / Decrement by 1
CMP	Compare two values (sets flags, no result stored)

3.7.3 Logic Instructions

Instruction	Use
AND, OR, XOR, NOT	Standard logical bitwise operations
TEST	Performs AND for testing bits, but does not store result
SHL / SHR	Shift bits Left / Right (logical shift)
ROL / ROR	Rotate bits Left / Right

3.7.4 Control Transfer Instructions

Instruction	Use
JMP	Unconditional jump to another location
JZ, JNZ, JC, JNC etc.	Conditional jumps (based on flag status)
CALL / RET	Call a procedure/subroutine and return from it
LOOP	Repeat a set of instructions until counter (CX) becomes zero

3.7.5 String Instructions

Instruction	Use
MOVS	Move string data (byte/word) from source to destination
CMPS	Compare two strings
SCAS	Scan a string for a particular value
LODS / STOS	Load string into AL/AX / Store AL/AX into string
REP	Repeat the string instruction CX number of times

3.7.6 Input/Output Instructions

- IN - reads data from an input port into accumulator (AL/AX).

- OUT - sends data from accumulator (AL/AX) to an output port.

3.7.7 Flag Control Instructions

Instruction	Use
STC / CLC	Set Carry flag / Clear Carry flag
STI / CLI	Set Interrupt flag (enable) / Clear Interrupt flag (disable)
STD / CLD	Set Direction flag / Clear Direction flag
CMC	Complement (toggle) the Carry flag

3.7.8 Process Control Instructions

- HLT - Halts (stops) the processor until reset or interrupt.
- WAIT - Makes processor wait until a particular condition occurs.
- NOP - No Operation, does nothing, used for timing delay.
- ESC / LOCK - Used for coprocessor handling and bus locking respectively.

3.7.9 Other Instructions

- INT - Generates a software interrupt.
- IRET - Returns from an interrupt service routine.
- XLAT - Used for table lookup/translation.

Exam Tip: You don't need to memorize every single instruction. Remember at least 4-5 instructions per category with their use, that is enough for most questions.

UNIT IV Memory Organization and Interrupts

SYLLABUS — UNIT IV *(06 Hours)*

Memory Organization: Segmentation, logical to physical address translation, even and odd memory banks, Read write cycle timing diagrams, Address mapping and decoding, I/O: memory mapped I/O & I/O Mapped I/O. Interrupts: Interrupt Control & status registers, Interrupt Vector Table (IVT), ISR, Hardware and software Interrupts, 8259 (Programmable Interrupt Controller): Features, Block Diagram, Control & Status registers.

4.1 Segmentation in 8086

8086 memory (1 MB) is divided into logical parts called segments. This is called Segmentation.

- Each segment can be maximum 64 KB in size.
- 4 types of segments: Code Segment (CS), Data Segment (DS), Stack Segment (SS), Extra Segment (ES).
- Segments can overlap each other in memory.

Why segmentation is used:

- To organize program, data, and stack separately.
- To allow relocation of program in memory easily.
- To support 1 MB addressing using only 16-bit registers.

4.2 Logical to Physical Address Translation

Programmer/program uses Logical Address (Segment : Offset), but actual memory uses Physical Address (20-bit). Conversion formula:

Physical Address = (Segment Register × 10H) + Offset Address

Example: Logical Address is 3000H : 0400H

- Physical Address = (3000H × 10H) + 0400H = 30000H + 0400H = 30400H

Exam Tip: This numerical (logical to physical address conversion) is asked very frequently for 2-4 marks. Always multiply segment by 10H (means shift left by 1 hex digit) then add offset.

4.3 Even and Odd Memory Banks

Since 8086 has a 16-bit (2 byte) data bus, memory is divided into TWO banks to allow transferring 2 bytes at once.

- **Even Bank (Low Bank) -** connected to data lines D0-D7, holds EVEN addressed bytes (0, 2, 4...). Enabled by A0 = 0.
- **Odd Bank (High Bank) -** connected to data lines D8-D15, holds ODD addressed bytes (1, 3, 5...). Enabled by signal BHE (Bus High Enable) = 0.

> ***Note:*** *If accessing a WORD (2 bytes) that starts at an EVEN address, both banks work together and data is read in ONE bus cycle (fast). If word starts at an ODD address, it needs TWO bus cycles (slower) - this is called an unaligned access.*

4.4 Read/Write Cycle Timing Diagram (Concept)

A bus cycle is the time taken to complete one read or write operation. Basic steps in a typical 8086 bus cycle (4 T-states: T1, T2, T3, T4):

1. T1: Address is sent out on the address bus by CPU.
2. T2: Control signals (RD/WR) are activated; data bus prepares for transfer.
3. T3: Actual data transfer happens (Read or Write).
4. T4: Bus cycle completes; signals are de-activated.

- If memory/device is slow, WAIT states are inserted between T3 and T4 using READY signal.

4.5 Address Mapping and Decoding

Address decoding means selecting the correct memory chip/IO device for a given address using extra hardware (decoders like 74LS138).

- Higher order address bits are used to select a particular chip (chip select logic).
- Lower order address bits are used to select a particular location inside that chip.
- This is needed because memory is built using multiple smaller chips combined together.

4.6 Memory Mapped I/O vs I/O Mapped I/O

Point	Memory Mapped I/O	I/O Mapped I/O
Address space	Shares SAME address space as memory	SEPARATE address space (only 64KB, 16-bit)
Instructions used	Normal memory instructions (MOV)	Special instructions (IN, OUT)
Address pins used	Full 20-bit address used	Only 16-bit address (A0-A15) used
Speed	Can be slightly slower (shares with memory)	Faster and dedicated for I/O
Address range	Reduces available memory for programs	Does not affect memory address space

Exam Tip: Memory Mapped I/O vs I/O Mapped I/O is asked as a direct comparison question for 4-8 marks.

4.7 Interrupts - Basic Concept

An interrupt is a signal that tells the CPU to STOP its current work temporarily, handle something urgent, and then RESUME the original work.

Steps when Interrupt occurs

1. CPU receives interrupt signal.
2. CPU finishes current instruction, saves current state (Flags, IP, CS) onto the stack.

3. CPU finds the address of the Interrupt Service Routine (ISR) using the Interrupt Vector Table.
4. CPU jumps to and executes the ISR (the special program that handles this interrupt).
5. After ISR finishes, IRET instruction is used to return back and restore the original state.

4.7.1 Interrupt Vector Table (IVT)

- IVT is a table stored in the first 1KB of memory (00000H to 003FFH).
- It contains addresses (CS:IP) of all 256 possible Interrupt Service Routines.
- Each entry in IVT takes 4 bytes (2 bytes for IP, 2 bytes for CS).
- Address of IVT entry for interrupt number N = N × 4.

4.7.2 ISR (Interrupt Service Routine)

ISR is a small program/subroutine written to handle a specific interrupt. It always ends with IRET instruction to return control to main program.

4.8 Hardware vs Software Interrupts

Point	Hardware Interrupt	Software Interrupt
Source	Generated by external hardware devices (keyboard, timer)	Generated by INT instruction in the program itself
Pins used	Uses INTR / NMI pins	No external pin needed
Priority	Can be maskable (INTR) or non-maskable (NMI)	Always executed, cannot be ignored
Example	Keyboard press, Timer tick	INT 21H (DOS function call)

- **Maskable Interrupt (INTR) -** can be disabled/ignored using CLI instruction (Interrupt Flag = 0).
- **Non-Maskable Interrupt (NMI) -** CANNOT be disabled, always serviced. Used for critical events like power failure.

4.9 Interrupt Control and Status Registers

The Flag Register's IF (Interrupt Flag) bit acts as the interrupt control:

- IF = 1 (set using STI): interrupts are ENABLED.
- IF = 0 (cleared using CLI): maskable interrupts are DISABLED.
- TF (Trap Flag) is used for single-step interrupt during debugging.

4.10 8259 Programmable Interrupt Controller (PIC)

8086 can directly handle very few hardware interrupts. The 8259 PIC chip is added to manage MULTIPLE hardware interrupts efficiently.

4.10.1 Features of 8259

- Can manage 8 interrupt inputs (IR0 to IR7) on a single chip.

- Can be cascaded (connected together) to handle up to 64 interrupts using multiple 8259 chips.
- Priority of each interrupt can be programmed (set by software).
- Can mask (disable) individual interrupt lines selectively.

4.10.2 Block Diagram of 8259 (Main Parts)

- **IRR (Interrupt Request Register) -** stores all interrupt requests that are waiting to be serviced.
- **ISR (In-Service Register) -** stores which interrupt is CURRENTLY being serviced.
- **IMR (Interrupt Mask Register) -** stores which interrupts are masked (disabled).
- **Priority Resolver -** decides which interrupt should be serviced first when many requests arrive together.
- **Control Logic -** manages overall working and communicates with CPU using INT and INTA signals.

4.10.3 Control and Status Registers of 8259

- **ICW (Initialization Command Words) -** 4 registers (ICW1-ICW4) used to SET UP/initialize the 8259 before use.
- **OCW (Operation Command Words) -** 3 registers (OCW1-OCW3) used DURING operation, to mask interrupts, set priority mode, etc.

Exam Tip: 8259 PIC block diagram with IRR, ISR, IMR explained is a common 8-10 mark question. Draw the block diagram showing connection to 8086 (INT, INTA pins).

UNIT V Parallel Organization

SYLLABUS — UNIT V *(06 Hours)*

Multiprocessors, Clusters, Flynn's Taxonomy for Multiple Processor Organizations, Closely and Loosely Coupled Multiprocessors Systems, Symmetric Multiprocessor (SMP) Organization, UMA, NUMA. RISC: Instruction execution characteristics, use of large register file, compiler-based register optimization, RISC architecture and pipelining. RISC Vs CISC.

5.1 Multiprocessors

A multiprocessor system has TWO or more CPUs (processors) working together, sharing memory and I/O, to increase speed and reliability.

- Multiple processors can execute different instructions/programs at the SAME time (true parallel processing).
- Used for: faster computation, better reliability (if one processor fails, others continue), load sharing.

5.2 Clusters

A cluster is a group of complete, independent computers (each with its own memory and OS) connected together through a network, working as ONE single system.

- Unlike multiprocessors, each computer in a cluster has its OWN memory (not shared).
- Used for: high availability, load balancing, very large scale computing (e.g. web servers, supercomputers).

Point	Multiprocessor	Cluster
Memory	Shared among processors	Each node has separate memory
Connection	Common bus/interconnect inside one system	Network connecting separate computers
Cost	Costly (special hardware)	Cheaper (uses normal computers)

5.3 Flynn's Taxonomy (VERY IMPORTANT)

Flynn classified computer architectures into 4 types based on number of Instruction streams and Data streams.

Type	Full Form	Meaning	Example
SISD	Single Instruction Single Data	One instruction works on one data at a time (traditional computer)	Old uniprocessor PCs
SIMD	Single Instruction Multiple Data	One instruction works on MANY data items at once	GPU, Vector processors
MISD	Multiple Instruction Single Data	Many instructions work on the SAME single data (rare, mostly theoretical)	Fault-tolerant systems

Type	Full Form	Meaning	Example
MIMD	Multiple Instruction Multiple Data	Many instructions work on many different data, fully independent	Multiprocessors, Multi-core CPUs

Exam Tip: Flynn's Taxonomy with all 4 types and one example each is asked almost every exam for 8 marks. This is one of the MOST important topics in Unit V.

5.4 Closely Coupled vs Loosely Coupled Multiprocessors

Point	Closely Coupled (Tightly Coupled)	Loosely Coupled
Memory	Processors SHARE common main memory	Each processor has its OWN local memory
Communication	Through shared memory (fast)	Through message passing over network (slower)
Example	SMP systems, Multicore CPU	Clusters, Distributed systems
Speed	Faster communication between processors	Slower, but easier to scale to many nodes

5.5 Symmetric Multiprocessor (SMP) Organization

In SMP, two or more SIMILAR processors share the SAME main memory and I/O, and are controlled by ONE single Operating System.

- All processors have EQUAL access and EQUAL capability (symmetric = equal).
- Any processor can run any task, including OS tasks - no processor is 'master' permanently.
- Operating system schedules processes/threads on whichever processor is free.

Advantages of SMP

- Better performance (parallel execution of tasks).
- High availability - if one processor fails, others keep system running.
- Easy to scale by adding more processors (incremental growth).

5.6 UMA vs NUMA (Memory Access Models)

Point	UMA	NUMA
Full Form	Uniform Memory Access	Non-Uniform Memory Access
Memory Access Time	SAME for all processors, no matter which memory location	DIFFERENT - faster for local memory, slower for remote memory
Memory Organization	Single shared memory pool, equally accessible to all	Memory is divided; each processor has its own LOCAL memory
Scalability	Limited - bus/memory becomes bottleneck with more processors	Better scalability for large number of processors
Used in	Small SMP systems	Large scale multiprocessor systems

Exam Tip: UMA vs NUMA is a frequent comparison question (4-6 marks). Remember: UMA = Uniform = Same time for everyone. NUMA = Non-Uniform = local memory is faster.

5.7 RISC (Reduced Instruction Set Computer)

RISC is a CPU design philosophy that uses a SMALL set of SIMPLE instructions, each taking roughly the SAME (usually 1) clock cycle.

5.7.1 Instruction Execution Characteristics of RISC

- Most instructions execute in ONE clock cycle (very fast and predictable timing).
- Fixed instruction length/format (easy for decoding).
- Only LOAD and STORE instructions can access memory; all other operations work only on registers.
- Few addressing modes (simple design).

5.7.2 Use of Large Register File

- RISC processors have a LARGE number of general-purpose registers (often 32 or more).
- Why: since memory access is slow, more data is kept inside fast registers, reducing need to access memory frequently.
- This directly improves speed because register access is much faster than memory access.

5.7.3 Compiler-Based Register Optimization

Since RISC depends heavily on registers, the COMPILER plays a big role in deciding which variables should be kept in registers for best performance.

- Compiler analyzes the program to find which variables are used most frequently.
- It tries to keep those variables in registers as long as possible (this is called register allocation/optimization).
- Good compiler optimization is critical to get the full speed benefit of RISC architecture.

5.7.4 RISC Architecture and Pipelining

Because RISC instructions are simple, fixed length, and take equal time, they are PERFECT for Pipelining.

- Pipelining means breaking instruction execution into stages (Fetch, Decode, Execute, Memory Access, Write Back) and overlapping multiple instructions at different stages simultaneously.
- Since every RISC instruction behaves uniformly, pipeline stages stay balanced with very few delays (hazards).
- This overlapping greatly increases overall throughput (instructions completed per second).

5.8 RISC vs CISC (VERY IMPORTANT TABLE)

Point	RISC	CISC
Full Form	Reduced Instruction Set Computer	Complex Instruction Set Computer
Instruction Set	Small, simple instructions	Large, complex instructions
Instruction Size	Fixed length	Variable length
Execution Time	Mostly 1 clock cycle per instruction	Multiple clock cycles per instruction

Point	RISC	CISC
Addressing Modes	Few	Many
Memory Access	Only through LOAD/STORE instructions	Many instructions can access memory directly
Registers	Large number of registers	Fewer registers
Pipelining	Very efficient, easy to implement	Difficult due to complex/variable instructions
Compiler Role	Compiler does more work (complex compiler)	Hardware does more work (simpler compiler)
Examples	ARM, MIPS, SPARC	Intel x86, 8086

Exam Tip: RISC vs CISC is THE most important and most repeated question of Unit V (8-10 marks every time). Learn this entire table by heart - examiner usually wants minimum 6-7 points.

Last Minute Revision - Most Important Points

Read this page 1 hour before exam if you have no time left. These are the highest-weightage topics from each unit.

Unit I Quick Points

- Von Neumann (single memory) vs Harvard (separate memory) - table.
- Intel processor evolution: 4004 (4-bit) to Core/Itanium (64-bit).
- Booth's Algorithm steps and numerical - practice once before exam.
- Carry Look Ahead Adder - Generate and Propagate concept.

Unit II Quick Points

- Memory Hierarchy pyramid diagram.
- 3 Cache Mapping techniques - Direct, Associative, Set-Associative.
- RAID levels table - especially RAID 0, 1, 5, 6.
- SRAM vs DRAM comparison.

Unit III Quick Points

- 8086 Architecture diagram - BIU and EU.
- All 7 Addressing Modes with examples.
- Register set: General Purpose, Segment, Pointer/Index, Flag register.
- Physical Address = Segment × 10H + Offset.

Unit IV Quick Points

- Logical to Physical address conversion numerical.
- Memory Mapped I/O vs I/O Mapped I/O table.
- Interrupt Vector Table and steps of interrupt handling.
- 8259 PIC - IRR, ISR, IMR registers and block diagram.

Unit V Quick Points

- Flynn's Taxonomy - SISD, SIMD, MISD, MIMD with examples.
- UMA vs NUMA table.
- RISC vs CISC table - most repeated question, learn fully.

ALL THE BEST FOR YOUR EXAM