Session 20-21

5(a) What is n-bit processor? Distinguish between 8085 and 8086.
[1+2 Marks]

What is an n-bit processor? *(1 mark)*

An **n-bit processor** is a microprocessor that can process **n bits of data at a time**.
It means the **size of its data bus, registers, and ALU** are all **n bits wide**.
👉 Example: 8085 is an **8-bit processor**, and 8086 is a **16-bit processor**.

Difference between 8085 and 8086 *(2 marks)*

Feature	8085	8086
Type	8-bit microprocessor	16-bit microprocessor
Data Bus Width	8-bit	16-bit
Address Bus Width	16-bit (64 KB memory)	20-bit (1 MB memory)
Registers	8-bit general purpose	16-bit general purpose
Clock Speed	Up to 3 MHz	Up to 10 MHz
Architecture	Single memory (no segmentation)	Segmented memory (Code, Data, Stack, Extra)

5(b)

i) "Computer works on 0 and 1". Explain this statement.
ii) Describe IN and OUTS instruction with respect to 8086.
[2+2 Marks]

Here's the clear **4-mark answer (2+2)** 👇

---

i) "Computer works on 0 and 1" – Explanation *(2 marks)*

A **computer works on binary system**, which uses only **two digits: 0 and 1**.

- These digits represent the **OFF (0)** and **ON (1)** states of electronic switches (transistors) inside the computer.
- All data — numbers, characters, and instructions — are **converted into binary form** before processing.
- The **CPU** performs all operations using these binary signals.

👉 **Hence, computers understand and process only 0s and 1s (binary language).**

---

ii) IN and OUT instructions (8086) *(2 marks)*

Instruction	Description	Example
IN	Used to **read data** from an **input port** into the **accumulator (AL or AX)**.	`IN AL, 60H` → Reads data from port 60H into AL.
OUT	Used to **send data** from the **accumulator (AL or AX)** to an **output port**.	`OUT 80H, AL` → Sends data from AL to port 80H.

Summary:

- `IN` → Input from port → Accumulator
- `OUT` → Output from accumulator → Port

5(c) Define peripheral. How can we determine any device as peripheral or not? Are DMA and coprocessors peripheral?
[3 Marks]

Definition of Peripheral: *(1 mark)*

A **peripheral** is any **external device connected to the computer** that **sends data to** or **receives data from** the CPU.
Examples: Keyboard, Mouse, Printer, Hard Disk, etc.

How to determine if a device is peripheral: *(1 mark)*

A device is considered a **peripheral** if it is **not part of the CPU or main memory**, but is **connected externally** to perform input, output, or storage functions.
👉 In short:
If it **communicates with the CPU through I/O ports or buses**, it is a **peripheral**.

Are DMA and Coprocessors peripheral? *(1 mark)*

- **DMA (Direct Memory Access)** controller is considered a **peripheral**, because it is an **external hardware device** that transfers data between memory and I/O devices without CPU intervention.
- **Coprocessor** (like math coprocessor 8087) is **a peripheral**, because it **works directly with the CPU** to perform specialized processing tasks.

☑ **Final answer:**
DMA → Peripheral
Coprocessor → peripheral

5(d)i) Suppose you would like to transfer data from your disk drive to a flash drive by using DMA controller. Explain whole procedure in details to complete the activities.
ii) Explain handshaking. Describe `MOV CS:[AX], DI` instructions with respect to 8086.
[1.5+2.5 Marks]

---

i) DMA transfer: disk → flash drive — full procedure (1.5 marks)

(Assuming a classic DMA controller + disk controller; steps shown in logical order)

1. **Allocate a buffer in main memory**
   CPU reserves a contiguous buffer and records its physical address (segment:offset) and

byte count. On ISA-style DMA, the buffer must respect page boundaries and use DMA page registers.

2. **Program the DMA controller**
    - Select a free DMA channel.
    - Load the DMA base address (offset) and page register (high bits) for the buffer.
    - Load the transfer byte count (number of bytes − 1).
    - Set the DMA mode to "read from device to memory" (peripheral → memory) and enable the channel.
3. **Prepare the disk controller (source)**
    - Program the disk/IDE/SCSI controller with the disk-sector(s) to read and give it the destination buffer info if required (older controllers only need to be told to read; DMA controller holds address).
    - Issue the read command to the disk controller.
4. **Device requests DMA when data ready**
    - When the disk controller has data ready it asserts **DRQ** (device request). The DMA controller will then begin bus arbitration.
5. **Bus arbitration / grant**
    - DMA asserts **HRQ** (Hold Request) to CPU.
    - CPU finishes current bus cycle, then raises **HLDA** (Hold Acknowledge). CPU releases address/data buses.
    - DMA controller takes control of the bus.
6. **Actual data transfer (DMA cycles)**
    - DMA performs appropriate memory cycles (byte or word transfers) to move data from the disk controller to memory buffer. It may assert **DACK** to the device for each transfer.
    - DMA decrements count and increments address automatically.
7. **Completion**
    - When count reaches zero (terminal count) DMA signals completion (TC) and/or the disk controller may issue its IRQ (e.g., IRQ14 for primary IDE).
    - DMA lowers HRQ; CPU lowers HLDA and regains bus control.
8. **Post-transfer tasks by CPU**
    - CPU checks status from disk controller to confirm successful read.
    - CPU initiates write from memory buffer to flash drive — either by programming DMA again for device-to-device transfer (device-to-device often uses memory as intermediary) or by issuing I/O commands to the flash controller (could use DMA again where flash is the target).
    - Update file system / close buffers, handle errors if reported.

Notes / practical points: ensure buffer alignment/page rules for the DMA channel, handle cache coherency (flush/invalidate CPU caches if present), and use interrupts (IRQ) to notify CPU for completion and error handling.

---

ii) Handshaking and `MOV CS:[AX], DI` in 8086 (2.5 marks)

Handshaking — short explanation (about 1 mark)

Handshaking is the exchange of control/status signals between two devices to coordinate data transfer so both sides know when data is ready and when it has been accepted.
Common forms:

- **Simple ACK/READY**: Sender asserts DATA and STROBE; receiver responds with ACK when it has read the data.
- **DMA handshaking**: device asserts **DRQ**; DMA controller uses **DACK**, **HRQ/HLDA** to gain bus control and transfer data.
- **Handshake guarantees** correct timing, prevents data loss, and allows devices of different speeds to communicate.

`MOV CS:[AX], DI` — explanation (about 1.5 marks)

- **What it does:** Writes the 16-bit contents of register **DI** into memory at the address given by the segment `CS` and offset `AX`.
- **Address calculation:** Physical address = `(CS << 4) + AX`. The instruction stores the word in memory at that linear address (two bytes: low byte at offset, high byte at offset+1).
- **Important clarifications:**
  - This **does not change** the CS register — it only *uses* CS as the segment base for the memory operand (CS: is a segment override).
  - Syntax meaning: destination is `CS:[AX]` (memory), source is `DI` (register). So data flows **DI → memory**.
  - This addressing mode (`[AX]`) is valid; using CS as segment override is allowed for memory operands.
  - Example: if `CS=2000h`, `AX=0010h`, `DI=1234h`, then memory at physical `20000h + 0010h` receives the word `34h` (low) and `12h` (high).

If you want, I can give a short timing diagram of DMA handshaking (DRQ → HRQ → HLDA → DACK → transfer → TC) or a one-line example showing registers and bytes for the `MOV` example.

6(a) Distinguish between microprocessor and microcontroller. Give the features of 8051.
[2+1 Marks]

📊 Microprocessor vs Microcontroller

Topic	Microprocessor (µP)	Microcontroller (µC)
Definition	A CPU on a single chip — needs external memory and I/O devices to work.	A complete computer system on a single chip (CPU + Memory + I/O + Timers).
Examples	Intel 8086, Pentium, Core i7	Intel 8051, PIC, Arduino (ATmega328), ARM Cortex-M
Memory	Requires external RAM/ROM.	Has built-in (internal) RAM and ROM.
I/O Ports	Must be connected through external chips.	Built-in I/O ports are available.
Working Type	Used for complex calculations, heavy tasks, and general-purpose computer design.	Performs specific control-oriented tasks; used in embedded systems.
Cost & Power	Expensive and consumes more power.	Cheaper and consumes less power.
Speed	Generally much higher (in GHz).	Comparatively lower (in MHz).
Applications	Used in PCs, laptops, and servers.	Used in washing machines, microwave ovens, cars, and IoT devices.

✨ এক কথায়

- **Microprocessor** হলো কম্পিউটারের *মস্তিষ্ক* 🧠 (general-purpose computing)।
- **Microcontroller** হলো *ছোট্ট অল-ইন-ওয়ান চিপ* 🛠️ (specific কাজের জন্য)।

Features of 8051 Microcontroller *(1 Mark)*

- 8-bit CPU
- 4 KB on-chip ROM
- 128 bytes on-chip RAM
- 32 I/O pins (4 ports × 8 bits)
- Two 16-bit timers/counters
- Full duplex serial communication
- 64 KB external memory support
- 6 interrupt sources

6(b) Shortly describe 82C55 PPI. How can it be programmed?
[2+2 Marks]

82C55 PPI – Short Description *(2 Marks)*

The **82C55 (Programmable Peripheral Interface)** is an **I/O device** used to interface peripherals (like keyboards, displays, printers) with a microprocessor.
It provides **24 programmable I/O lines**, divided into **three 8-bit ports**:

- **Port A** – 8-bit
- **Port B** – 8-bit
- **Port C** – 8-bit (can be split into two 4-bit ports: C upper & C lower)

Main Features:

- Each port can be programmed as **input or output**.
- Can operate in **three modes**:
  - **Mode 0:** Simple I/O (no handshake)
  - **Mode 1:** Handshake I/O (synchronized data transfer)
  - **Mode 2:** Bidirectional data transfer (Port A only)

Programming of 82C55 *(2 Marks)*

The 82C55 is programmed by writing a **control word** to its **control register**.

Steps to program:

1. **Select control register** using address lines (A1, A0 = 11).
2. **Send control word** from CPU to the control register.
3. Control word defines:
   - Which ports are input/output.
   - Which mode (0, 1, or 2) each group will operate in.

Example:

```
MOV AL, 82H   ; Control word (set Port A input, Port B output, etc.)
OUT 83H, AL   ; Send to control register of 82C55
```

☑ **In short:**
82C55 provides **24 I/O lines** programmable in different modes, and it is **programmed using a control word** written to its control register.

🧩 82C55 PPI – সংক্ষিপ্ত বিবরণ *(২ মার্ক)*

82C55 হলো একটি **Programmable Peripheral Interface (PPI)** চিপ, যা মাইক্রোপ্রসেসরের সাথে বিভিন্ন **ইনপুট-আউটপুট ডিভাইস** (যেমন কী-বোর্ড, ডিসপ্লে, প্রিন্টার ইত্যাদি) সংযোগ করতে ব্যবহৃত হয়।

- এতে মোট **২৪টি I/O লাইন** আছে, যা ৩টি ৮-বিট পোর্টে বিভক্ত –

  - **Port A** → 8-bit
  - **Port B** → 8-bit
  - **Port C** → 8-bit (এটি দুই ভাগে ভাগ করা যায় – C Upper ও C Lower)

মূল বৈশিষ্ট্য:

- প্রতিটি পোর্টকে **input বা output** হিসেবে সেট করা যায়।
- এটি **৩টি মোডে কাজ করে:**
  - **Mode 0:** Simple I/O (সাধারণ ইনপুট/আউটপুট, কোনো handshake নেই)
  - **Mode 1:** Handshake I/O (সিঙ্ক্রোনাইজড ডাটা ট্রান্সফার)
  - **Mode 2:** Bidirectional I/O (Port A এর জন্য দুই দিকের ডাটা ট্রান্সফার)

⚙️ 82C55 কিভাবে প্রোগ্রাম করা হয় *(২ মার্ক)*

82C55-কে প্রোগ্রাম করা হয় একটি **Control Word** পাঠিয়ে, যা এর **Control Register**-এ লেখা হয়।

প্রোগ্রামিং ধাপ:

1. **Address lines (A1, A0)** দিয়ে Control Register নির্বাচন করা হয় (A1A0 = 11)।
2. CPU থেকে **Control Word** পাঠানো হয় Control Register-এ।
3. এই Control Word-এর মাধ্যমে নির্ধারণ করা হয় –
   - কোন পোর্ট **input** আর কোনটা **output** হবে।
   - প্রতিটি পোর্ট কোন **mode (0, 1, 2)** তে কাজ করবে।

উদাহরণ:

```
MOV AL, 82H      ; Control word সেট করা হচ্ছে
OUT 83H, AL      ; Control register-এ পাঠানো হলো
```

6(c) Why is 8087 coprocessor used? How does the coprocessor interface with the main processor?
[1.5+1.5 Marks]

Why the 8087 Coprocessor is Used *(1.5 marks)*

The **Intel 8087** is a **math coprocessor** (Numeric Data Processor) designed to work with the **8086/8088** CPU.
Its purpose is to **speed up complex arithmetic operations**, especially:

- Floating-point (real-number) calculations
- Exponential, logarithmic, trigonometric, and square-root functions

☑ **In short:** The 8087 relieves the main CPU from time-consuming mathematical tasks, giving much **faster execution of scientific and engineering computations.**

How the Coprocessor Interfaces with the Main Processor *(1.5 marks)*

- The 8087 is **physically connected** to the same **address, data, and control buses** as the 8086.
- The 8086 and 8087 **work in parallel**:
  - 8086 fetches and decodes all instructions.
  - When it detects an **ESC (escape)** instruction (for the 8087), it passes the instruction to the coprocessor.
  - Both processors then execute simultaneously—the 8087 performs numeric processing while 8086 continues with other instructions.
- They are synchronized by **status and control signals** (like `BUSY`, `READY`, `TEST`).

☑ **Summary:**
8087 shares the system bus with **8086**, executes numeric instructions in parallel, and uses **ESC** opcodes plus control signals for coordination.

6(d)i) "Different CPU model or brand have different operational capability". Explain this statement.
ii) Give the benefits of condition codes.
[3+1 Marks]

Here's a short and clear **answer (2 + 2 marks)** 👇

---

i) "Different CPU model or brand have different operational capability." *(2 Marks)*

This means that **each CPU model or brand is designed with different architecture, instruction set, and performance level** — so their operations and capabilities are not the same.

Explanation:

- CPUs differ in **word size** (8-bit, 16-bit, 32-bit, 64-bit).
- They have different **instruction sets** (e.g., x86, ARM, RISC-V).
- Processing speed (clock frequency), number of cores, and built-in features (like floating-point unit, cache size, pipelining, etc.) vary.
- Therefore, a program written for one CPU type may not run directly on another brand or model.

☑ **Example:**
Intel Core i7 and ARM Cortex-M both are CPUs but their instruction sets and processing capabilities are totally different.

---

ii) Benefits of Condition Codes *(2 Marks)*

Condition codes (or flag bits) are special bits in the processor's **status register** that show the result of arithmetic or logical operations.

Benefits:

1. They help in **decision-making** — e.g., branch or jump based on Zero (Z), Carry (C), Sign (S), or Overflow (O) flags.
2. Simplify **program control flow** without recalculating results.
3. Improve **program efficiency** — conditional instructions can quickly decide what to do next.
4. Used for **loop control**, **comparisons**, and **error checking** in programs.

☑ **In short:**
Condition codes make it easier and faster for the CPU to take logical decisions based on previous operation results.

Session 19-20

5. (a)(i) "CPU actually works on binary digits" - Justify this statement.
(ii) Enlist the major evolution in computational era with its key technology.
(2 + 2 Marks)

i) "Computer works on 0 and 1" – Explanation *(2 marks)*

A **computer works on binary system**, which uses only **two digits: 0 and 1**.

- These digits represent the **OFF (0)** and **ON (1)** states of electronic switches (transistors) inside the computer.
- All data — numbers, characters, and instructions — are **converted into binary form** before processing.
- The **CPU** performs all operations using these binary signals.

👉 **Hence, computers understand and process only 0s and 1s (binary language).**

Major Evolution in Computational Era with Key Technology

Era / Generation	Key Technology
First Generation (1940–1956)	Vacuum Tubes
Second Generation (1956–1963)	Transistors
Third Generation (1964–1971)	Integrated Circuits (ICs)
Fourth Generation (1971–Present)	Microprocessors
Fifth Generation (Present & Beyond)	Artificial Intelligence (AI), Parallel Processing, Quantum Computing

☑ **In short:** Each generation is marked by a **major technological advancement** that improved speed, size, and efficiency of computers.

(b) Explain the learning outcome from this course.
(3 Marks)(Write Yourself)

(c) Give advantages and disadvantages of flags in CPU. Give the flag status of flag register after performing the following operation:

```
ABCD
xA
```

(2 + 2 Marks)

Advantages and Disadvantages of Flags in CPU *(2 Marks)*

Advantages:

1. Flags indicate the **status of the last operation**, helping in **decision-making** (e.g., jump if zero).
2. They enable **efficient control flow** in programs without recomputation.
3. Useful in **error detection** (overflow, carry, parity, etc.).

Disadvantages:

1. Flags occupy **special bits in the status register**, increasing CPU complexity.
2. Misinterpretation of flags can **cause program errors** if not handled correctly.
3. Not all operations automatically update flags, requiring **extra instructions** sometimes.

Flag Status after Operation *(2 Marks)*

Operation:

```
ABCD
× A
```

Step by step (hex multiplication):

1. Convert to decimal:
   - `ABCD` = 43981 (decimal)
   - `A` = 10 (decimal)
2. Multiply: `43981 × 10 = 439810`
3. Check 16-bit limit:
   - Maximum 16-bit value = `FFFF` = 65535
   - Result 439810 > 65535 → **Overflow occurs**

Flag Status:

Flag	Status	Reason
Carry (C)	1	Multiplication result exceeds 16 bits

Flag	Status	Reason
Auxiliary Carry (AC)	0	Not used in MUL
Zero (Z)	0	Result ≠ 0
Sign (S)	1	Most significant bit (MSB) of result in 16-bit register is 1
Overflow (O)	1	Result exceeds 16-bit range
Parity (P)	0	Even/odd number of 1s in low byte of result

(d)(i) Distinguish between coprocessor and peripheral.
(ii) Mention the features of 80287.
(1.5 + 1.5 Marks)

---

i) Difference between Coprocessor and Peripheral *(1.5 Marks)*

Feature	Coprocessor	Peripheral
Definition	A specialized processor that works **along with the main CPU** to perform specific tasks (e.g., math calculations).	An external device that communicates with the CPU to perform **input/output or storage functions**.
Integration	Works in **parallel with the CPU**; tightly coupled.	Connected via I/O ports or buses; loosely coupled.
Example	Intel 8087 (math coprocessor)	Keyboard, Printer, Hard Disk, DMA controller

☑ **In short:** Coprocessor **assists CPU in computation**, peripheral **extends I/O functionality**.

---

ii) Features of 80287 *(1.5 Marks)*

- Math coprocessor designed for **80286 CPU**
- Performs **floating-point arithmetic operations** (addition, subtraction, multiplication, division, square root, trigonometric functions)
- **Registers:** 8 × 80-bit floating-point registers (stack-based)
- Can execute **ESC (escape) instructions** from main CPU in parallel
- Reduces CPU workload for **complex numeric computations**
- Supports **IEEE 754 standard** for floating-point operations

☑ **In short:** 80287 is an advanced math coprocessor for 80286, handling all floating-point and complex numeric calculations efficiently.

---

6. (a) Explain the responsibilities of segment register in protected mode memory addressing. If DS = 01051H in a protected mode system, which entry, table, and requested privilege level are selected?
(4 Marks)

(a) Responsibilities of Segment Registers in Protected Mode

In **protected mode** (like in 80286/80386+):

1. **Segment registers** (CS, DS, SS, ES, FS, GS) **do not directly hold physical addresses**.
2. Instead, they **hold a 16-bit selector**, which refers to a **descriptor in a descriptor table**.
   - **GDT (Global Descriptor Table)** – for global segments.
   - **LDT (Local Descriptor Table)** – for process-specific local segments.
3. The **descriptor** contains:
   - Base address of the segment (32-bit in 386+)
   - Segment limit (size)
   - Access rights / privilege level (DPL – Descriptor Privilege Level)
4. **Responsibilities:**
   - Map logical addresses (selector + offset) to **linear/physical addresses**.
   - Enforce **access rights and privilege checks**.
   - Ensure **memory protection**, preventing programs from accessing unauthorized memory.

Given Example: DS = 01051H

- DS = **0105H** (16-bit selector).
- **Selector structure:**

Bits	Meaning
15–3	Index (entry number in GDT or LDT)
2	Table Indicator (TI) – 0 = GDT, 1 = LDT
1–0	Requested Privilege Level (RPL)

1. Convert 0105H to binary: `0000 0001 0000 0101`
2. Extract **fields**:
   - **RPL (bits 1–0)** = `01` → Requested Privilege Level = 1
   - **TI (bit 2)** = `0` → Table = **GDT**
   - **Index (bits 15–3)** = `0000 0001 0000 0` → Entry number = **32** (binary `0000 0001 0000 0` = 32 decimal)

☑ **Answer:**

- **Descriptor Table:** GDT
- **Entry Number:** 32
- **Requested Privilege Level:** 1

(b)(i) Why is accumulator called so?
(ii) Enlist the differences between 8086 and 8088 microprocessors.
(1.5 + 1.5 Marks)

(i) Why is the Accumulator called so? *(1.5 Marks)*

- The **accumulator (register A or AX)** is called so because it **"accumulates" the results of arithmetic and logic operations** performed by the CPU.
- Most instructions like **ADD, SUB, AND, OR, MUL, DIV** either take the accumulator as a **source, destination, or both**.
- It is the **primary working register** in the CPU for calculations and temporary storage during operations.

(ii) Differences between 8086 and 8088 Microprocessors *(1.5 Marks)*

Feature	8086	8088
Data Bus Width	16-bit	8-bit
Address Bus Width	20-bit	20-bit
Instruction Fetch	Fetches 16 bits per cycle	Fetches 8 bits per cycle (slower instruction fetch)
Performance	Faster due to 16-bit bus	Slower due to narrower bus
Application	Systems needing higher performance	Systems with cheaper 8-bit memory and peripheral support

☑ **In short:** 8086 is **16-bit** and faster; 8088 is **8-bit bus** and slower but compatible with 8-bit hardware.

(c)(i) Explain handshaking with respect to CPU.
(ii) Describe addressing modes of DSP56300. Explain the trigger of DMA.
(1.5 + 2.5 Marks)

(d)(i) How does 82C55 can be programmed?
(ii) Distinguish among the various modes of operation of 82C55.
(1.5 + 1.5 Marks)

(i) How can 82C55 be programmed? *(1.5 Marks)*

The **Intel 82C55 (Programmable Peripheral Interface)** is programmed by **writing a control word** into its **Control Register**.

- This control word defines how ports **A, B, and C** will operate.
- Each port can be set as **input or output**, and can operate in **different modes** (Mode 0, 1, or 2).
- The control word is sent by the **CPU** through the **data bus** to the control register of 82C55.

☑ **In short:**
82C55 is programmed by sending a **control word** that defines port directions and operating modes.

(ii) Modes of Operation of 82C55 *(1.5 Marks)*

Mode	Name	Description
Mode 0	Simple Input/Output	Basic data transfer without handshaking; each port works independently.
Mode 1	Strobed Input/Output	Data transfer with **handshaking signals** using Port C lines for control.
Mode 2	Bidirectional Bus I/O	**Port A** used for **two-way communication** with handshaking (mainly for microprocessor-to-microprocessor data transfer).

☑ **In short:**
82C55 works in **Mode 0 (simple I/O)**, **Mode 1 (strobed I/O)**, and **Mode 2 (bidirectional I/O)**.

Session: 18-19

---

1. (a) (i)How can you determine the microprocessor as 8-bit or 16-bit or 32-bit or 64-bit?
(2 Marks)

(ii) Distinguish between 8085 and 8086.(2)

(i) How to determine a microprocessor as 8-bit, 16-bit, 32-bit, or 64-bit? *(2 Marks)*

A microprocessor is classified as **8-bit, 16-bit, 32-bit, or 64-bit** based on the **size of the data bus** or **internal register width** — that is, the number of bits it can **process or transfer in one operation**.

☑ **Explanation:**

- If the CPU can process **8 bits at a time**, it is an **8-bit processor** (e.g., 8085).
- If it can process **16 bits at a time**, it is a **16-bit processor** (e.g., 8086).
- If it can process **32 bits**, it is a **32-bit processor** (e.g., Intel 80386).
- If it can process **64 bits**, it is a **64-bit processor** (e.g., AMD Ryzen, Intel Core i7).

(b)(i) Mention the role of segment register during protected mode operation.
(2 Marks)

Role of Segment Register during Protected Mode Operation *(2 Marks)*

In **protected mode**, the **segment registers (CS, DS, SS, ES, FS, GS)** no longer hold direct physical addresses — instead, they hold **selectors** that point to entries in **descriptor tables** (like GDT or LDT).

☑ **Main Roles:**

1. **Address Translation:**
   - The value in a segment register is used as an **index (selector)** to locate a **segment descriptor** in the **Global Descriptor Table (GDT)** or **Local Descriptor Table (LDT)**.
   - This descriptor contains the **base address**, **limit**, and **access rights** of the segment.
2. **Memory Protection:**
   - Each segment descriptor defines access privileges (read/write/execute) and the range of addresses that a program can use — preventing one program from accessing another's memory area.

👉 **In short:**
In protected mode, **segment registers provide logical-to-physical address translation** and **ensure memory protection** by using descriptor tables.

(ii) Distinguish microcontroller and microprocessor.
(2.5 Marks)

(c)(i) Shortly describe 82C55 PPI with its operational modes.
(2 Marks)

(ii) Why is memory decoding necessary in computer systems?
(1.5 Marks)

Why Memory Decoding is Necessary in Computer Systems *(1.5 Marks)*

Memory decoding is necessary to **identify and select the correct memory location or device** when the CPU accesses memory.

☑ **Explanation:**

- The CPU sends an **address** on the address bus.
- The **memory decoder** (made of logic gates or decoder ICs) interprets this address and **activates the specific chip or location** corresponding to that address.
- Without decoding, multiple devices might respond to the same address, causing **data conflict or wrong access**.

☑ **In short:**
Memory decoding ensures that **only one memory device or location is selected at a time**, allowing the CPU to access data **accurately and efficiently**.

(d) Criticize the statement "More registers integration produce faster CPU".
(2 Marks)

Criticism of the Statement:

"More registers integration produce faster CPU." *(2 Marks)*

☑ **Explanation:**
While having **more registers** can reduce memory access and **improve performance**, it **does not always guarantee a faster CPU**.

Reasons / Criticism:

1. **Diminishing Returns:**
   After a certain point, adding more registers gives **little performance gain**, because most programs don't use all of them efficiently.

2. **Increased Complexity:**
   More registers make **instruction decoding and context switching** more complex, which can **slow down** execution.
3. **Other Factors Matter:**
   CPU speed also depends on **clock frequency, cache size, pipeline depth, and architecture efficiency**, not just the number of registers.

☑ **In short:**
Having more registers can help performance **to some extent**, but **CPU speed depends on the overall architecture**, not only on register count.

---

2. (a)(i) Compare PROM, EPROM, and EEPROM.**(1.5 Marks)**

🧩 Difference between PROM, EPROM and EEPROM

বিষয়	PROM	EPROM	EEPROM
Full Form	Programmable Read Only Memory	Erasable Programmable Read Only Memory	Electrically Erasable Programmable Read Only Memory
Programming Method	একবার প্রোগ্রাম করা যায় (fuse burning)	প্রোগ্রাম করা যায় EPROM programmer দিয়ে	প্রোগ্রাম ও মুছে ফেলা যায় বৈদ্যুতিকভাবে (electrically)
Erasing Method	Erase করা যায় না	UV light (ultraviolet light) দ্বারা erase করা হয়	Electrical signal দিয়েই erase করা যায়
Reusability	একবার ব্যবহারের পর আর ব্যবহার করা যায় না	বারবার erase ও reprogram করা যায়	সহজেই বারবার erase ও reprogram করা যায়
Erase Time	– (not erasable)	প্রায় 15–20 মিনিট UV আলোতে	খুব দ্রুত (মাইক্রোসেকেন্ড লেভেল)
Erase Method Location	–	চিপটিকে ডিভাইস থেকে খুলে UV আলোতে দিতে হয়	ডিভাইসের ভিতরেই (in-circuit) erase করা যায়
Volatility	Non-volatile (power off হলেও ডেটা থাকে)	Non-volatile	Non-volatile
Cost	সস্তা	মাঝারি	তুলনামূলকভাবে বেশি

বিষয়	PROM	EPROM	EEPROM
Speed	দ্রুত পড়া যায়	পড়া দ্রুত, লেখা ধীর	পড়া ও লেখা দুটোই দ্রুত
Common Use	স্থায়ী firmware বা configuration ডেটা	পুরনো BIOS chip বা test firmware	আধুনিক BIOS, Flash memory, pen drive, SSD ইত্যাদি

(ii) Give the evolution of microprocessor from mechanical era to present (with important advancements).
(2.5 Marks)

Evolution of Microprocessor

Era / Generation	Time Period	Key Technology / Advancement
Mechanical Era	Before 1940	Mechanical computing devices like **abacus, Pascaline**; calculation done manually or mechanically.
First Generation	1940–1956	**Vacuum tubes** used in ENIAC; large, expensive, high power consumption.
Second Generation	1956–1963	**Transistors** replaced vacuum tubes; smaller, faster, and more reliable.
Third Generation	1964–1971	**Integrated Circuits (ICs)**; multiple transistors on a single chip, reducing size and cost.
Fourth Generation	1971–Present	**Microprocessors** (single-chip CPUs like 8085, 8086); made computers smaller, faster, and more affordable.
Fifth Generation / Modern Era	Present & Beyond	**Advanced microprocessors, RISC/CISC architectures, multicore CPUs, AI processors**; focus on **parallel processing, low power, high performance, and embedded systems**.

☑ **In short:**
The evolution of microprocessors shows the transition from **mechanical devices → vacuum tubes → transistors → ICs → single-chip microprocessors → modern multicore and AI processors**, with each step **reducing size, increasing speed, and improving efficiency**.

(b)(i) How does the DMA speed up CPU performance?**(2 Marks)**
(ii) Distinguish between SRAM and DRAM.**(2 Marks)**

🧩 Difference between SRAM and DRAM

বিষয়	SRAM (Static RAM)	DRAM (Dynamic RAM)
Full Form	Static Random Access Memory	Dynamic Random Access Memory
Data Storage Method	Data সংরক্ষণ হয় flip-flop সার্কিটের মাধ্যমে	Data সংরক্ষণ হয় capacitor-এর চার্জ আকারে
Refresh Requirement	Refresh দরকার হয় না	প্রতি কয়েক মিলিসেকেন্ড পরপর refresh দরকার হয়
Speed	খুব দ্রুত (fast access time)	তুলনামূলকভাবে ধীর
Cost	দাম বেশি	দাম কম
Density (Storage capacity)	কম (small size, কম bit প্রতি chip)	বেশি (বড় মেমরি সাইজ তৈরি করা যায়)
Power Consumption	বেশি (কারণ flip-flop সার্কিট সবসময় power নেয়)	কম (capacitor-based design)
Used As	Cache memory (CPU cache, registers)	Main memory (RAM modules)
Complexity of Circuit	জটিল সার্কিট (6 transistor per bit)	সহজ সার্কিট (1 transistor + 1 capacitor per bit)
Volatility	Volatile (power off হলে ডেটা হারায়)	Volatile (power off হলে ডেটা হারায়)
Access Time	কম (দ্রুত)	বেশি (ধীর)

(c)(i) Why is stepper motor so called?
(1.5 Marks)
(ii) "All coprocessors are peripherals, but all peripherals are not coprocessors" - Explain this statement.
(2.5 Marks)

ব্যাখ্যা:

1. **Coprocessor হলো একটি Peripheral:**
   - Coprocessor হলো একটি বিশেষ ধরনের **peripheral device**, কারণ এটি মেইন প্রসেসরের সাথে **যোগাযোগ করে**।

- অর্থাৎ, এটি একটি **external বা auxiliary device** যা প্রসেসরের কাজকে সাহায্য করে।
- উদাহরণ: Math Coprocessor, Graphics Coprocessor।
- তাই বলা যায়, **"All coprocessors are peripherals"**, কারণ সব coprocessor মেইন প্রসেসরের সাথে সংযুক্ত এবং তার কাজকে সমর্থন করে।

2. **সব Peripherals Coprocessor নয়:**
   - কিন্তু সব peripherals **computation করতে পারে না**, তারা শুধুমাত্র **I/O কাজের জন্য** ব্যবহৃত হয়।
   - উদাহরণ: Keyboard, Printer, Disk drive।
   - তাই বলা যায়, **"but all peripherals are not coprocessors"**, কারণ প্রতিটি peripheral computation বা special instruction দিতে সক্ষম নয়।

সারসংক্ষেপ:

- **Coprocessor** = Peripheral + Special computation capability
- **Peripheral** = শুধু I/O support (সবাই coprocessor নয়)

(d) "CPU actually works on binary digits "- Explain this statement(2)

---

3. (a) Suppose you would like to transfer data from your disk drive to a flash drive by using DMA controller. Explain the whole procedure in detail to complete the activities.**(2 Marks)**

OUT OF SYLLABUS

(ii) Describe overlapping data movement mechanism of DMA.
(2 Marks)
(b)(i) Describe the responsibility of memory management unit in computer system.
(3 Marks)

(i) Responsibility of Memory Management Unit (MMU) — (3 Marks)

The **Memory Management Unit (MMU)** is a crucial hardware component in a computer system that controls and manages how memory is accessed and used by the CPU.

Main Responsibilities of MMU:

1. **Address Translation (Mapping):**
   - Converts **virtual addresses** generated by the CPU into **physical addresses** in main memory.
   - Enables programs to use virtual memory independent of physical memory layout.
2. **Memory Protection:**
   - Ensures that one process cannot access another process's memory area.
   - Prevents unauthorized read/write access to restricted memory regions.
3. **Paging and Segmentation Management:**
   - Handles **page tables** and **segment registers** to divide memory into smaller, manageable units.
   - Supports efficient use of physical memory through paging or segmentation.
4. **Cache and Memory Control:**
   - Works with cache controllers to manage memory hierarchy for faster data access.

☑ **In short:**
The MMU manages memory **allocation, protection, and translation**, ensuring each process gets a **safe and efficient** view of memory in a **multitasking environment**.

(ii) Calculate the number of page table entries that are needed for following combinations of virtual address size (n) and page size (P):

n	p=2^P	#PTE
16	4K	
64	16K	

Formula:

$$\#PTE = \frac{\text{Virtual Address Space}}{\text{Page Size}}$$

Case 1:

n = 16 bits, Page size = 4 KB

$$\text{Virtual Address Space} = 2^{16} = 64 \text{ KB}$$

$$\text{Page Size} = 4 \text{ KB} = 2^{12}$$

$$\#PTE = \frac{2^{16}}{2^{12}} = 2^4 = 16$$

✅ **Answer:** 16 page table entries

Case 2:

n = 64 bits, Page size = 16 KB

$$\text{Virtual Address Space} = 2^{64}$$

$$\text{Page Size} = 16 \text{ KB} = 2^{14}$$

$$\#PTE = \frac{2^{64}}{2^{14}} = 2^{50}$$

✅ **Answer:** 2^{50} page table entries ≈ **1.1259 × 10^{15}**

(c)(i) Define handshaking.(1)
(ii) Explain page fault handling mechanism in virtual memory system.
(3 Mark)

Page Fault Handling Mechanism in Virtual Memory System (3 Marks)

A **page fault** occurs when a program tries to access a page that is **not currently loaded in main memory** (RAM). The CPU generates a **page-fault interrupt**, and the operating system handles it.

Steps in Page Fault Handling:

1. **Detect Page Fault:**
    - CPU tries to access a virtual address.
    - MMU checks the **page table**.
    - If the page is **not present**, a **page fault interrupt** is raised.
2. **OS Handles the Fault:**
    - The **OS saves the CPU state**.
    - Determines which **page is needed** and whether **a free frame** is available in memory.
3. **Bring Page into Memory:**
    - If no free frame, the OS may use **page replacement** (swap out a page to disk).
    - The **required page is read from secondary storage** (disk) into a free frame in RAM.
4. **Update Page Table:**
    - The page table entry is updated with the **new frame number** and **present bit** set to 1.
5. **Resume Execution:**
    - CPU resumes the **faulting instruction** using the newly loaded page.

☑ **In short:**
Page fault handling allows the system to **extend memory using virtual memory**, ensuring the program can continue even if the required page is not initially in RAM.

(d) "CPU actually works on binary digits" - Explain this statement.
(2 Marks)