Chinmay Sir Topics & Questions

# [ques](https://docs.google.com/presentation/d/1XTF8uHa-LQL6jSblhkkLIyn8sKIao5Mo/edit?usp=sharing)

1. What is n-bit processor?

A microcontroller or processor is called 'N-bit' because controllers and processors have a fundamental data width. Each register might be N bits, each instruction might be N bits, the data bus might be N bits, the memory might be addressed with N bits.

1. describe the relationship among hardware, software and firmware.

<https://theintactone.com/2019/10/12/cf-u4-topic-2-relationship-between-hardware-and-software/>

1. Mention the function of 8086 queue.

In 8086, a 6-byte instruction queue is presented at the Bus Interface Unit (BIU). It is used to pre fetch and store at the maximum of 6 bytes of instruction code from the memory. Due to this, overlapping instruction fetch with instruction implementation increases the processing speed.

The processor executes a program by fetching the instructions from memory and executing them. Usually the processor execution speed is much faster than the memory access speed. Instruction queue is used to prefetch the next instructions in a separate buffer while the processor is executing the current instruction.

In 8086, a 6-byte instruction queue is presented at the Bus Interface Unit (BIU). It is used to prefetch and store at the maximum of 6 bytes of instruction code from the memory. Due to this, overlapping instruction fetch with instruction execution increases the processing speed.

The execution unit (EU) is supposed to decode or execute an instruction. When EU is busy in decoding and executing an instruction, the BIU fetches up to six instruction bytes for the next instructions. These bytes are called as the pre-fetched bytes and they are stored in a first in first out (FIFO) register set, which is called as a queue.

The fetching of opcodes well in advance, prior to their need for execution increases the overall efficiency of the processor boosting its speed. The processor no longer has to wait for the memory access operations for the subsequent instruction opcode to complete.

Instruction queue is 6 bytes so that it can store the longest instruction.

8086 is the 1st processor to support Instruction Queue.

1. 8086 is a 16-bit microprocessor, but its memory mapping is 20 bits. How it is possible?

8086 is a 16 bit microprocessor that means its arithmetic logic unit can execute 16 bit multiple instruction at a time and it's internal registers most of the instructions are 16 bits.

But 8086 has 20 bit address bus means it can address any one of 2^20 = 1 MB memory locations.

The BIU (Bus Interface Unit) of the CPU consists of four segment register. Upper 16 bits of the starting address for that segment.

In BIU, pointer registers also called offset consists the distance from the base address to the next instruction byte to be fetched.

To make from 16 bit to 20 bit memory address the BIU automatically inserts zeroes for the lowest 4 bits of the segment base address, then the IP is added to the code segment(CS). Then the 16 bit offset is added to the segment. Thus 20 bit memory mapping is possible.

For example,

If the CS register contains 348AH, the starting address for the code segment is 348A0H (Here 0 is appended)

If IP is 4214H, then the 20 bit physical address of memory is

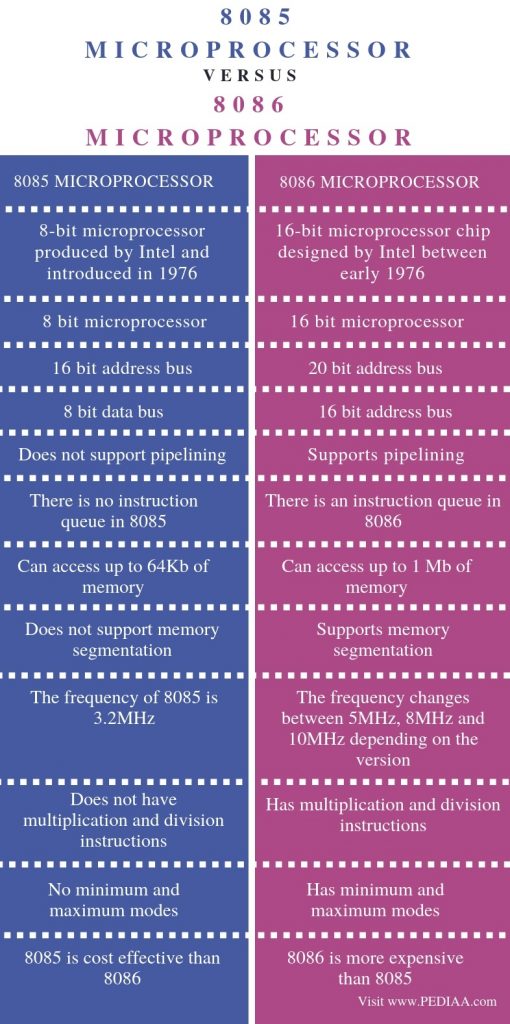
CS = 348A0H

IP = 4214H

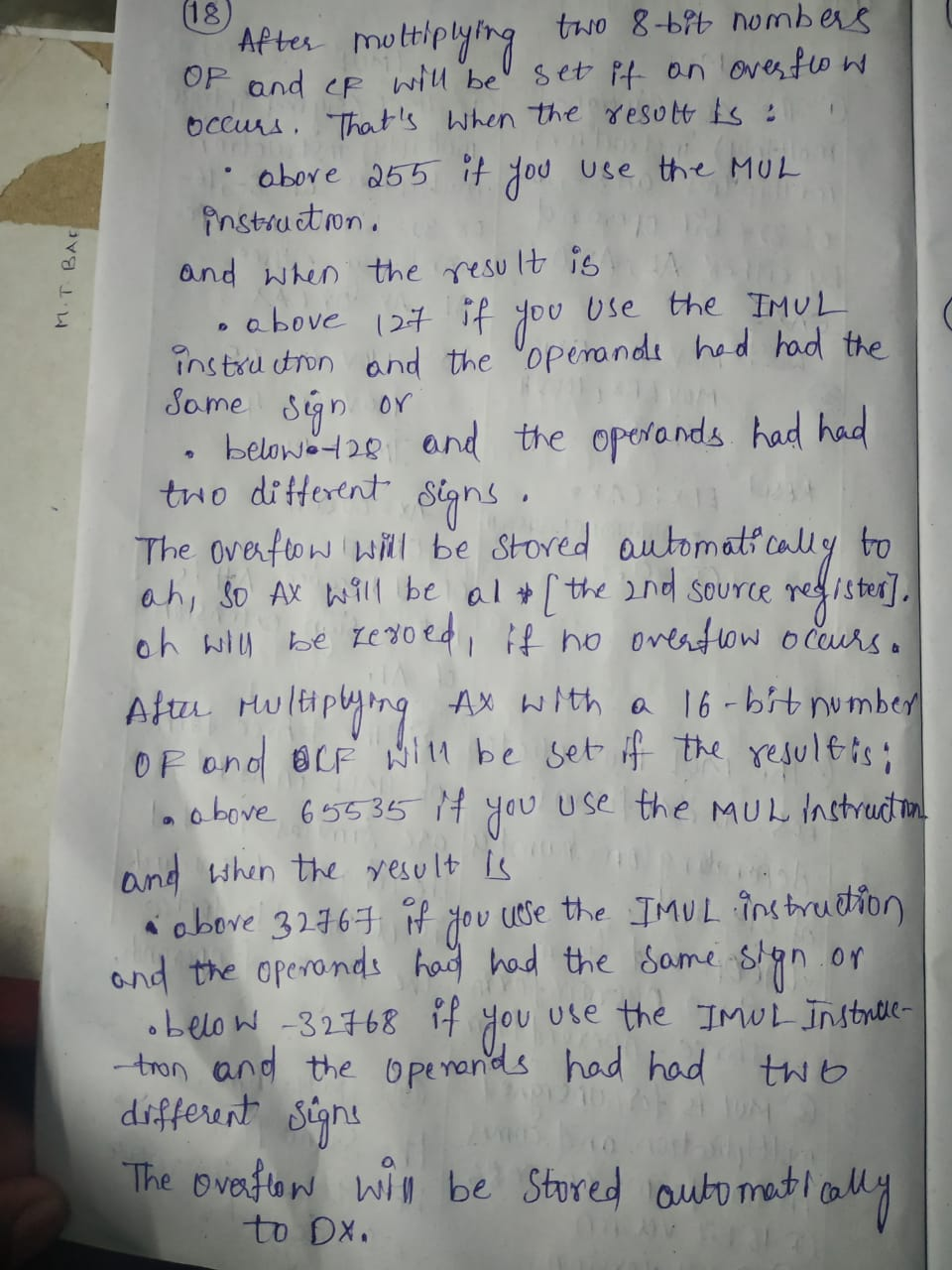
-------------------------------------------

Physical address = 38AB4H

This is why 8086 memory mapping is 20 bits.



1. Give the flag status of the flag register after performing this multiplication = 3B\*A



# [82C55 PPI](https://docs.google.com/presentation/d/1XTF8uHa-LQL6jSblhkkLIyn8sKIao5Mo/edit?usp=sharing)

**6. How can the two groups [A, B] of ports of 82C55 be programmed?**

**Answer:**  
The 82C55 PPI has **two groups** of ports — **Group A** (Port A and upper Port C) and **Group B** (Port B and lower Port C).  
Each group can be **programmed independently** in any of the available modes:

* **Mode 0:** Simple input/output
* **Mode 1:** Input/output with handshaking
* **Mode 2:** Bidirectional data transfer (only for Port A)

The control word register determines how each port operates.

**7. The continuous motor is familiar to us, but the stepper is a digital motor. Why it is digital? How it can be operated through 82C55 PPI?**

**Answer:**  
A **stepper motor** is called a **digital motor** because it moves in **discrete steps** rather than continuous rotation — each pulse moves it by a fixed angle.  
Using **82C55 PPI**, we can operate a stepper motor by:

* Sending **binary patterns** to the motor’s control lines through **Port A or Port B**.
* Each output pattern energizes the coils sequentially to make the motor rotate **step-by-step**.

Thus, digital signals from the PPI control the motor’s position and direction precisely.

**8. Briefly describe 82C55 PPI.**

**Answer:**  
The **Intel 82C55 Programmable Peripheral Interface (PPI)** is a general-purpose **I/O device** that interfaces a microprocessor with external devices like keyboard, display, ADC/DAC, or motor.  
It has **three 8-bit ports (Port A, B, and C)**, which can be configured as **input or output** in different **modes (0, 1, 2)** using a control word.  
It provides **flexible interfacing** between CPU and peripherals without complex hardware.

**9. Shortly describe 82C55 PPI. How it can be programmed?**

**Answer:**  
The **82C55 PPI** provides **24 I/O pins** divided into **Port A, B, and C**.  
It is **programmable** through a **control word** written into its control register.  
This control word sets:

* Input/output direction of each port
* Operating mode (Mode 0, 1, or 2)

Programming is done by sending a proper **8-bit control word** from the CPU to define how each port will function.

**10. Explain mode 1 operation and mode 2 operations of 82C55.**

**Answer:**

**Mode 1 – Input/Output with Handshaking:**

* Provides **strobed I/O** using handshake signals for synchronized data transfer.
* Each group uses part of **Port C** for control signals.
* Ensures data is transferred only when both devices are ready.
* Example: Used for interfacing with printer or keyboard.

**Mode 2 – Bidirectional Data Transfer:**

* Available **only for Port A (Group A)**.
* Allows **two-way communication** using handshake lines from **Port C**.
* Data can flow **both to and from** the peripheral using proper synchronization.
* Example: Used for data exchange between two microprocessors.

# [Coprocessor](https://docs.google.com/presentation/d/1oFHXHe8BAQs2_tgtxNZbexudpGxhTKz3/edit?usp=sharing)

**1. Why coprocessor is used in computer systems?**

**Answer:**  
A **coprocessor** is used to **increase the performance** of a computer by handling **specialized tasks** like arithmetic, floating-point operations, or signal processing.  
It offloads these tasks from the main CPU, allowing faster and more efficient processing.

**2. What is the difference between coprocessors and peripheral devices?**

**Answer:**

| **Coprocessor** | **Peripheral Device** |
| --- | --- |
| Works closely with the CPU to perform computation tasks. | Performs input/output or external communication. |
| Shares memory and instruction control with the main CPU. | Controlled by I/O commands, not part of CPU operations. |
| Example: 8087 Math Coprocessor. | Example: Keyboard, printer, disk drive. |

**3. How can the coprocessor interface with the main processor?**

**Answer:**  
The coprocessor interfaces with the main processor through a **bus interface** using:

* **Shared system buses (data, address, control).**
* **Special control signals** like TEST, BUSY, and READY.  
  It synchronizes operations using these signals and executes **coprocessor instructions** embedded in the main program.

**4. Briefly describe a numeric coprocessor.**

**Answer:**  
A **numeric coprocessor** (or **math coprocessor**) is a hardware unit that performs **floating-point arithmetic**, **logarithmic**, and **trigonometric** calculations faster than the CPU.  
Example: **Intel 8087**, used with the 8086 processor, executes complex math functions using an **internal stack-based architecture**.

**5. Why coprocessor is used in computer systems? How does the coprocessor interface with the main processor?**

**Answer:**  
A coprocessor is used to **enhance processing speed** for **mathematical or logical tasks** that are difficult for the CPU.  
It interfaces with the main processor through **shared buses** and **control signals**, executing special coprocessor opcodes that the CPU recognizes but passes to the coprocessor for execution.

**6. “All coprocessors are peripherals, but all peripherals are not coprocessors.” Explain this statement.**

**Answer:**  
All **coprocessors** are **peripheral devices** because they connect externally to the CPU and assist it.  
However, not all peripherals (like printers, keyboards, or monitors) perform computation — hence, they are **not coprocessors**.  
Only devices that **enhance CPU computation** qualify as coprocessors.

**7. Explain about 8087 coprocessors.**

**Answer:**  
The **Intel 8087** is a **numeric coprocessor** designed to work with the **8086/8088** CPU.  
It performs **floating-point arithmetic**, **exponential**, **logarithmic**, and **trigonometric** operations.  
It uses a **stack of 8 registers (ST0–ST7)** and operates **in parallel** with the CPU using a **synchronized bus interface**.

**8. Distinguish between 8086 and 8087 coprocessors.**

**Answer:**

| **8086** | **8087** |
| --- | --- |
| General-purpose microprocessor. | Numeric (math) coprocessor. |
| Executes integer and control operations. | Executes floating-point and complex math. |
| Can work independently. | Works alongside 8086; depends on it for instruction flow. |
| ALU handles integer arithmetic. | Dedicated FPU (Floating Point Unit). |

**9. Mention 6 processors with their corresponding coprocessors.**

**Answer:**

| **Main Processor** | **Coprocessor** |
| --- | --- |
| 8086 | 8087 |
| 80286 | 80287 |
| 80386 | 80387 |
| 80486 | Built-in FPU (no external coprocessor) |
| Pentium | Integrated FPU |
| ARM Cortex-M | Floating Point Unit (FPU) |

.