Course Name: Micro Processors and Micro Controllers

Topic Name: 8086 Microprocessor

Dr. Appalabathula Venkatesh M.Tech, Ph.D
Assistant Professor,
Department of EEE, ANITS

Cos

Course Objectives:

At the end of the course, students will be able to

- **CO1** (BL-2) Describe the architecture and various addressing modes of a typical 8085 microprocessor
- **CO2** (BL-4) Classify different memory devices to Discuss the interfacing between memory and 8085 microprocessor
- **CO3** (BL-3) Describe the architecture of a typical 8086 microprocessor to illustrate the general bus operations
- **CO4** (BL-3) Describe the various peripheral devices and show how the peripherals (8259,8251 & 8253) are interfaced with Microprocessor.
- **CO5** (BL-4) Use the architecture of 8051 microcontroller and illustrate how 8051 is interfaced with advanced applications.

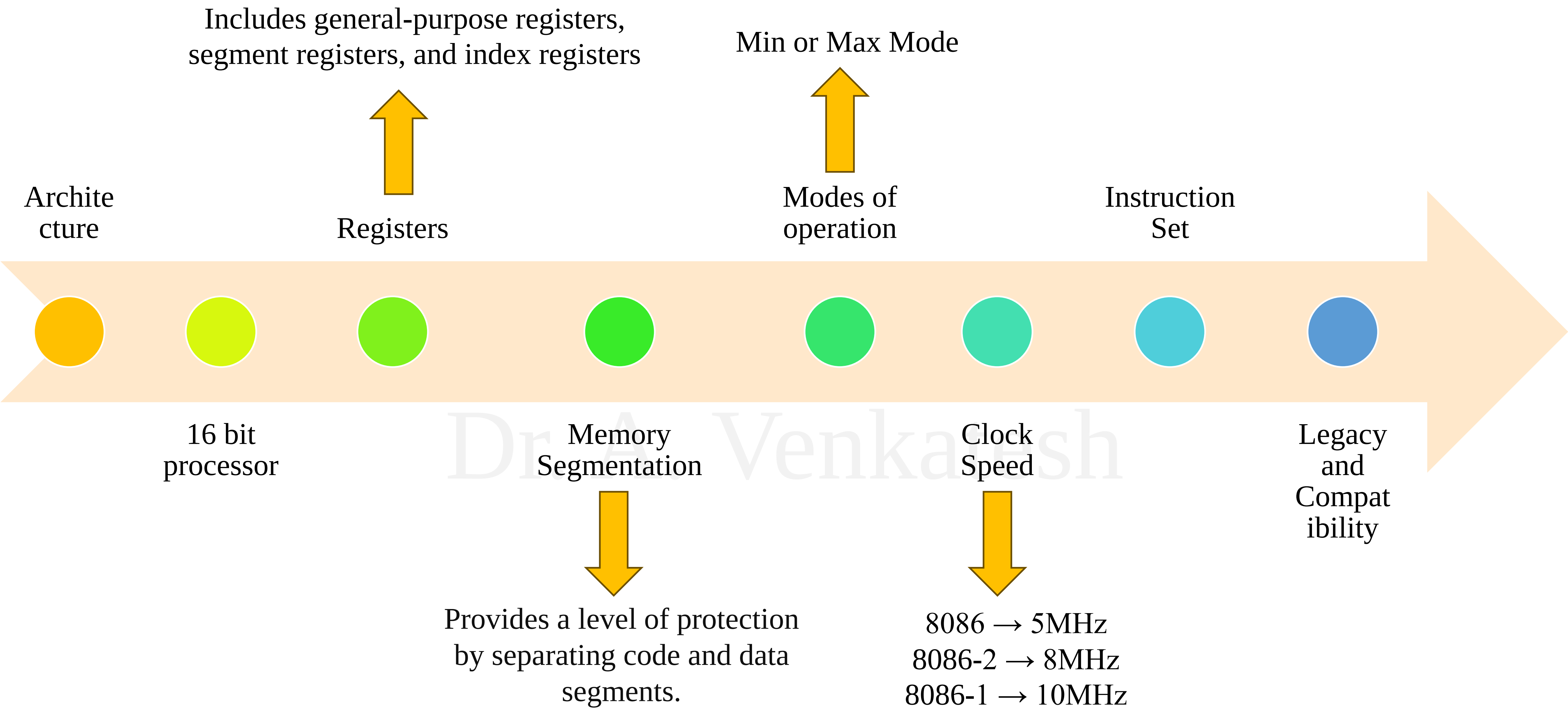
Syllabus

UNIT 3

8086 Microprocessor:

Introduction to microprocessor 8086, Internal Architecture and pin diagram, Register organization, Memory segmentation, Types of interrupt, External Memory addressing.

Introduction to 8086 Microprocessor



Architecture

16 bit processor

Registers

Includes general-purpose registers, segment registers, and index registers

Memory Segmentation

Provides a level of protection by separating code and data segments.

Modes of operation

Min or Max Mode

Clock Speed

8086 → 5MHz
8086-2 → 8MHz
8086-1 → 10MHz

Instruction Set

Legacy and Compatibility

Comparison between 8085 and 8086

Parameter	8085	8086
Size	**8 bit** Microprocessor	**16 bit** Microprocessor
Address lines	**16 bit** address bus	**20 bit** address bus
Data bus	**8 bit** data bus	**16 bit** data bus
No. of flag registers	**5 Flag** registers	**9 Flag** registers
Memory segmentation	It **doesn't support** memory segmentation	It **supports** memory segmentation.
Min or Max modes	It **doesn't** have modes.	It **has Min or Max modes** of operation.
Cost	**Low**	**High**
Clock Speed	Microprocessor clock frequency is **3MHz**	Microprocessor clock frequency is varies as **5MHz, 8MHz and 10MHz** for different versions.
No. of transistors used	Less. **Approx. 6500**	More, **Approx. 29000**
Pipelining	It doesn't supports.	It supports pipelining
Processor based on	**Accumulator** based	**General purpose register** based
Memory capacity	It can access **upto 64KB**	It can access **upto 1MB**
No of processors used	**One**	**Multiple**(Additional processors can be employed based on the application)
No of I/O addressed	**2^8=256 I/O's**	**2^16=65,536 I/O's**
Instruction Queue	It **doesn't** have any instruction Queue	It have an instruction queue

Internal Architecture of 8086

1.Bus Interface Unit (BIU):

1. Instruction Queue
2. Address Calculation Unit (ACU):Effective Address Calculation
3. Queue Control Unit: Queue Management
4. Segmentation Unit: Segment Registers

2. Execution Unit (EU):

1. ALU (Arithmetic Logic Unit)
2. General-Purpose Registers
3. Pointer and Index Registers
4. Flag Register
5. Timing and Control Unit:

   Instruction Decoder

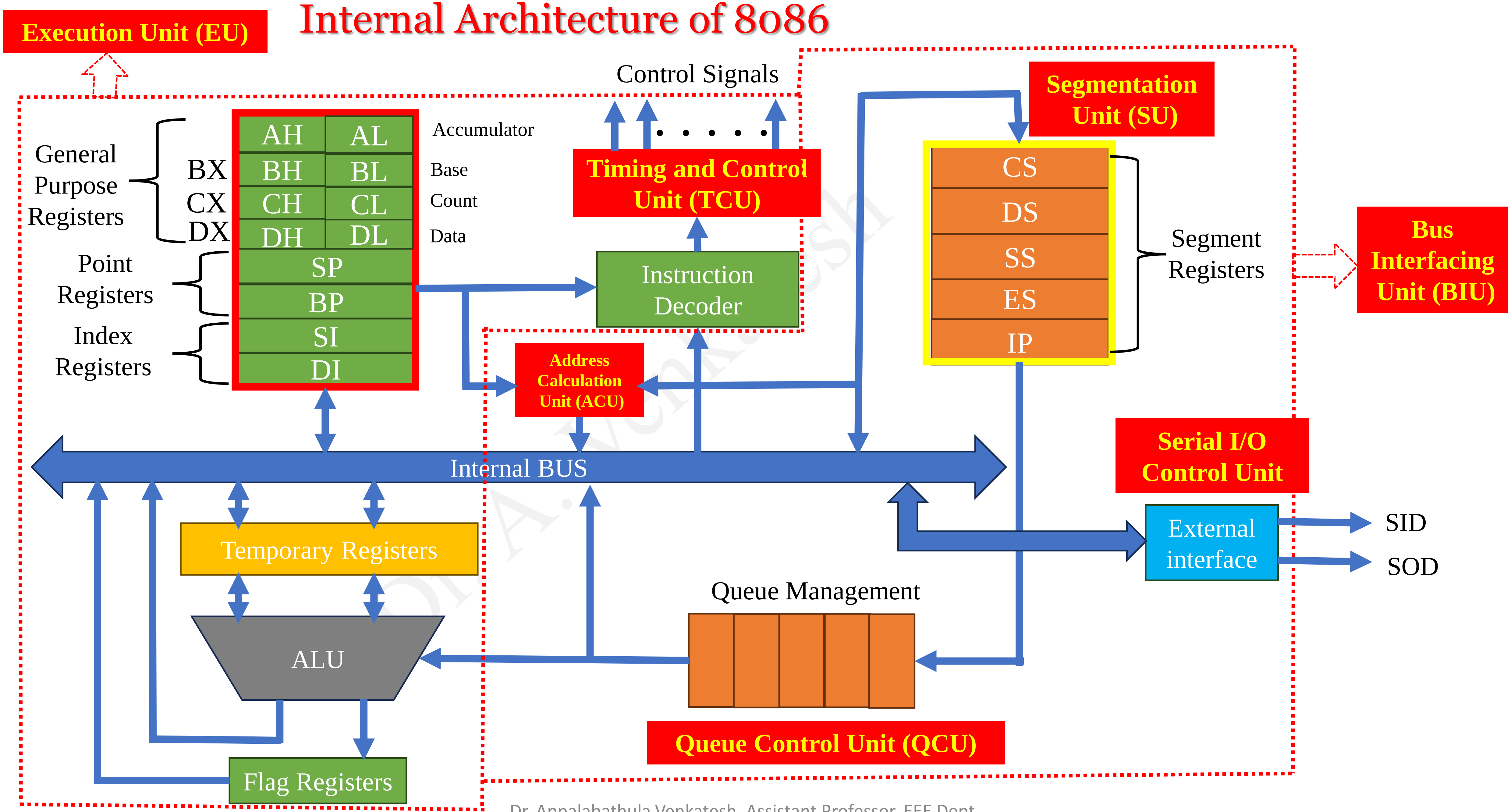
   Control Signals

   Clock Generator

6. Serial I/O Control Unit:

   External Interface

Internal Architecture of 8086



Internal Architecture of 8086

Bus Interfacing Unit (BIU)

Instruction Queue: The BIU fetches instructions from memory and stores them in a prefetch queue. **This queue helps to improve instruction execution speed by allowing the processor to fetch the next instruction while the current one is being executed**.

Segment Register: The BIU manages the segment registers (CS, DS, SS, ES), which are used in the memory segmentation scheme.

Execution Unit (EU)

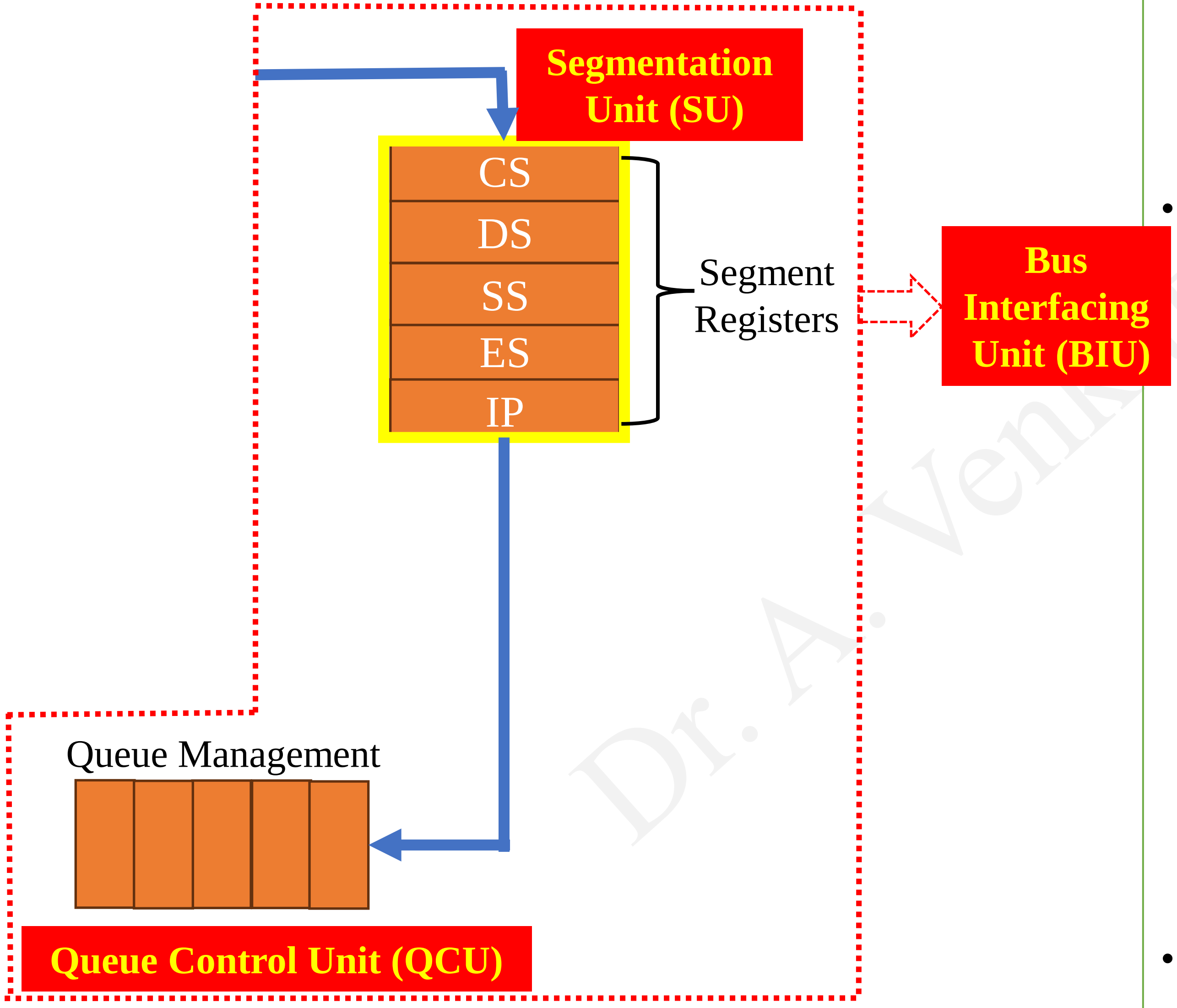
ALU (Arithmetic Logic Unit): Performs arithmetic and logical operations. The ALU is responsible for carrying out the actual data processing operations specified by the instructions.

General-Purpose Registers: The 8086 has several general-purpose registers, including AX, BX, CX, DX, SI, DI, BP, and SP. These registers are used for **data manipulation and storage during program execution.**

Pointer and Index Registers: Includes registers like SI (source index), DI (destination index), BP (base pointer), and SP (stack pointer), which are used for specific purposes **like string operations and addressing**.

Flag Register: Contains flags that indicate the **status of operations**, such as the zero flag (ZF), sign flag (SF), carry flag (CF), and others. These flags are **crucial for conditional branching and decision-making during program execution**

Internal Architecture of 8086



- **Instruction Queue: When EU executes instructions, the BIU gets 6-bytes of the next instruction and stores them in the instruction queue** and this process is known as instruction pre fetch. This process increases the speed of the processor.

- **Segment Registers:** A segment register contains the **addresses of instructions and data in memory** which are used by the processor to access memory locations. It points to the starting address of a memory segment currently being used. There are 4 segment registers in 8086 as given below:

  - **Code Segment Register (CS):** Holds **instruction codes** of a program.
  - **Data Segment Register (DS):** The **data, variables and constants** given in the program are held.
  - **Stack Segment Register (SS):** holds **addresses and data of subroutines.** It also holds the **contents of registers and memory locations** given in PUSH instruction.
  - **Extra Segment Register (ES):** holds the **destination addresses of some data of certain string instructions**.

- **Instruction Pointer (IP):** It acts as a **program counter**. It indicates to the **address of the next instruction** to be executed.

Internal Architecture of 8086

Bus Interfacing Unit (BIU)

Instruction Queue: The BIU fetches instructions from memory and stores them in a prefetch queue. **This queue helps to improve instruction execution speed by allowing the processor to fetch the next instruction while the current one is being executed**.

Segment Register: The BIU manages the segment registers (CS, DS, SS, ES), which are used in the memory segmentation scheme.

Execution Unit (EU)

ALU (Arithmetic Logic Unit): Performs arithmetic and logical operations. The ALU is responsible for carrying out the actual data processing operations specified by the instructions.

General-Purpose Registers: The 8086 has several general-purpose registers, including AX, BX, CX, DX, SI, DI, BP, and SP. These registers are used for **data manipulation and storage during program execution.**

Pointer and Index Registers: Includes registers like SI (source index), DI (destination index), BP (base pointer), and SP (stack pointer), which are used for specific purposes **like string operations and addressing**.

Flag Register: Contains flags that indicate the **status of operations**, such as the zero flag (ZF), sign flag (SF), carry flag (CF), and others. These flags are **crucial for conditional branching and decision-making during program execution**

Internal Architecture of 8086

1. **Carry Flag (CF) :** In case of addition this flag is set if there is **a carry out of the MSB**. The carry flag also serves as a borrow flag for subtraction. In case of subtraction it is set when borrow is needed.
2. **Parity Flag(PF):** It is set to 1 if result of byte operation or lower byte of the word operation contain **an even number of ones**; otherwise it is zero.
3. **Auxiliary Carry Flag (AF):** This flag is set if there is an overflow out of bit 3 i.e., **carry from lower nibble to higher nibble** (D3 bit to D0 bit). This flag is used for BCD operations and it is not available for the programmer.
4. **Zero Flag (ZF):** The zero flag; sets if the **result of operation in ALU is zero** and flag resets if the result is nonzero. The zero flag is also set if a certain register content becomes zero following an increment or decrement operation of that register.
5. **Sign Flag(SF):** After the execution of arithmetic or logical operations. If the MSB of the result is 1, the sign bit is set. **Sign bit 1 indicates the result is negative; otherwise it is positive**.
6. **Overflow Flag (OF):** This flag is set **if result is out of range**. For addition this flag is set when there is a carry into the MSB and no carry out of the MSB or vice-versa. For subtraction, it is set when the MSB needs a borrow and there is no borrow from the MSB, or vice-versa.

The three remaining flags are used to control certain operations of the processor.

1. **Trap Flag (TF):** One way to **debug a program is to run the program one instruction at a time and see the contents of used registers and memory variables after execution of every instruction**. This process is called "**single stepping**" through a program. Trap flag is used for single stepping through a program. If set, a trap is executed after execution of each instruction, i.e., interrupt service routine is executed which displays various registers and memory variable contents on the display after execution of each instruction. Thus programmer can easily trace and correct errors in the program.
2. **Interrupt Flag (IF):** It is used to allow/prohibit the interruption of a program. If set, **a certain type of interrupt (a maskable interrupt) can be recognized by the 8086**; otherwise, these interrupts are ignored.
3. **Direction Flag (DF):** It is used with string instructions. If DF= O, **the string is processed from its beginning with the first element having the lowest address.** Otherwise, the string is processed from the high address towards the low address.

Internal Architecture of 8086

Segmentation Unit (SU)

Segment Registers: As mentioned earlier, the segment registers (CS, DS, SS, ES) are managed by the Bus Interface Unit but are used **for addressing memory locations** in the Segmentation Unit.

Address Calculation Unit (ACU)

Effective Address Calculation: The ACU is responsible for calculating the **effective address of operands in memory**, **taking into account the memory segmentation scheme**.

Control Unit (CU)

Instruction Decoder: **Interprets the fetched instructions and generates control signals** for the various functional units of the processor.

Control Signals: The Control Unit generates **control signals that coordinate the operations of the BIU, EU, and other units.** These signals **ensure** that the **instructions are executed in the correct sequence.**

Internal Architecture of 8086

Queue Control Unit (QCU)

Queue Management: Coordinates the movement of instructions between the **instruction queue and the EU**. This helps in the **efficient execution of instructions**.

Clock Generator

Clock Distribution: Generates the clock signals necessary **for the synchronization of all internal operations**.

Serial I/O Control Unit

External Interface:

Address Bus, Data Bus, and Control Bus: These buses facilitate **communication** between **the microprocessor and external devices, including memory and I/O devices**.

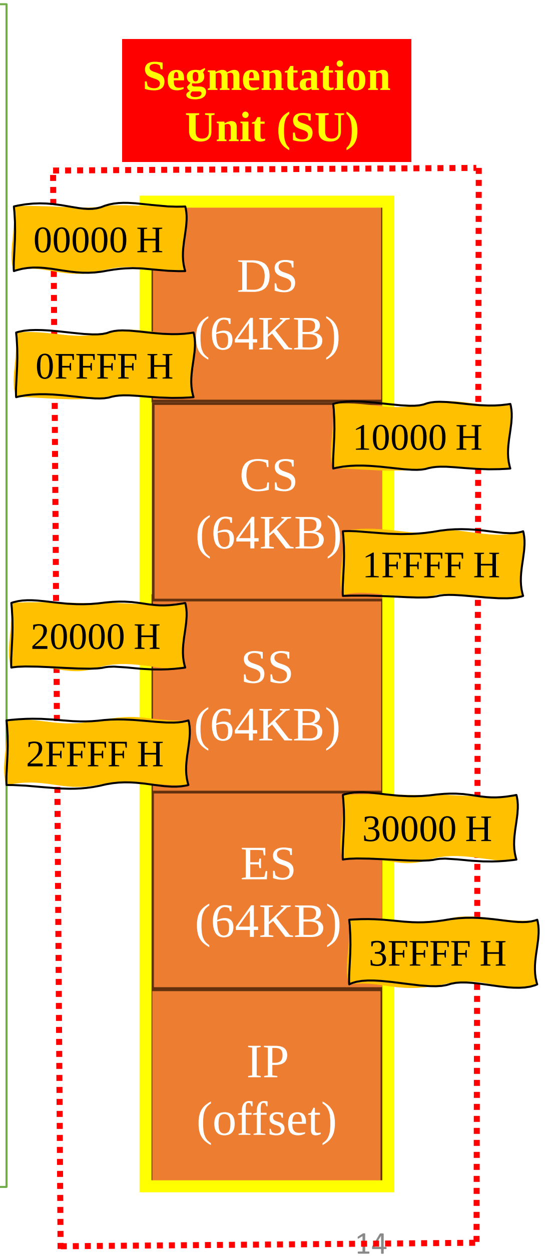
Memory Segmentation

Segmented Addressing: In the 8086 architecture, memory is divided into segments, and each segment is 64 KB in size. Instead of using a single 16-bit address to access the entire memory space (as in a flat memory model), the 8086 uses a segmented addressing scheme. The combination of a 16-bit segment address and a 16-bit offset address points to a specific location in memory.

1.The physical address (A) is calculated as follows: A=Segment×10H+Offset
2.This segmentation allows the use of more than 64 KB of memory, which was a significant advantage at the time.

Segment Registers: The 8086 has four segment registers: CS (Code Segment), DS (Data Segment), SS (Stack Segment), and ES (Extra Segment). Each register holds a 16-bit value representing the base address of its associated segment. When combined with an offset, these registers allow the processor to access different parts of the memory.

1. **CS (Code Segment):** Points to the segment where the currently executing program's instructions are located.
2. **DS (Data Segment):** Typically points to the segment where data elements, such as variables, are stored.
3. **SS (Stack Segment):** Points to the segment used for the program stack.
4. **ES (Extra Segment):** An additional segment register that can be used for various purposes.

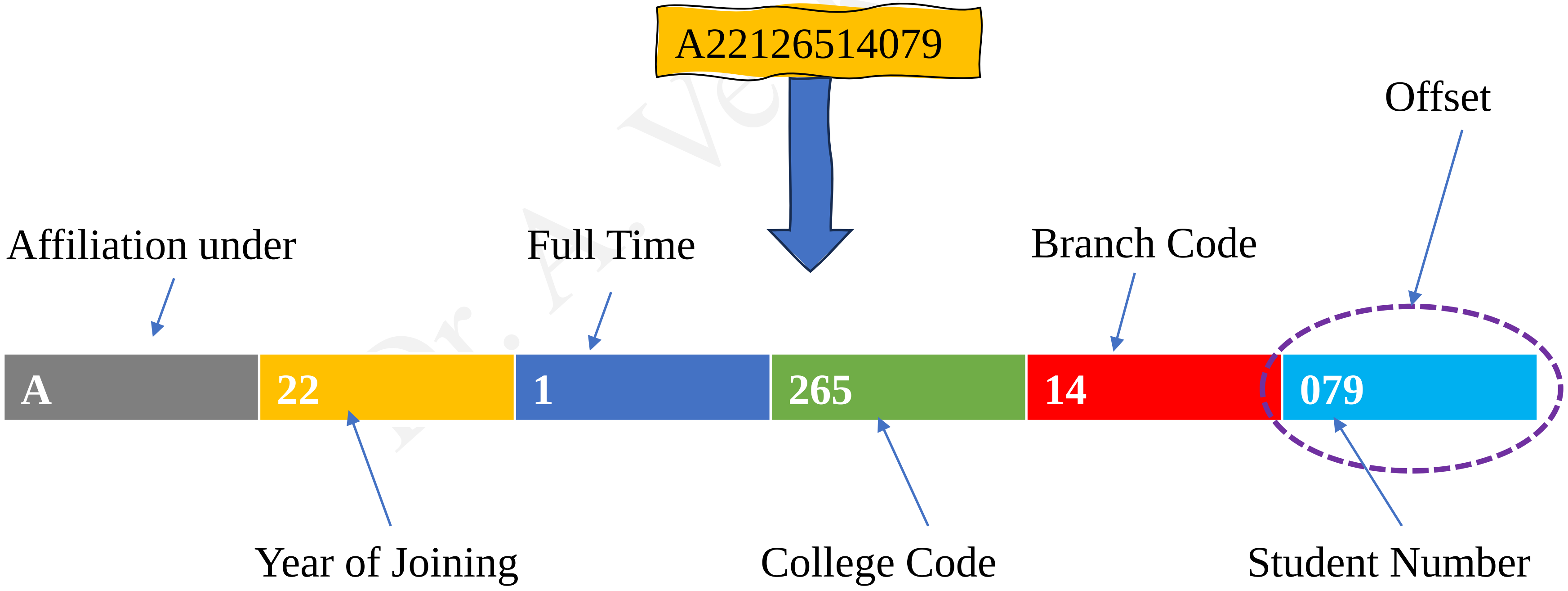


Physical Address Calculation

For Memory segmentation in 8086 → The physical address (A) = Segment×10H+Offset

Example

The physical address (A) = A22126514×1000+079



A22126514079

Affiliation under

Full Time

Branch Code

Offset

A	22	1	265	14	079

Year of Joining

College Code

Student Number

Physical Address Calculation

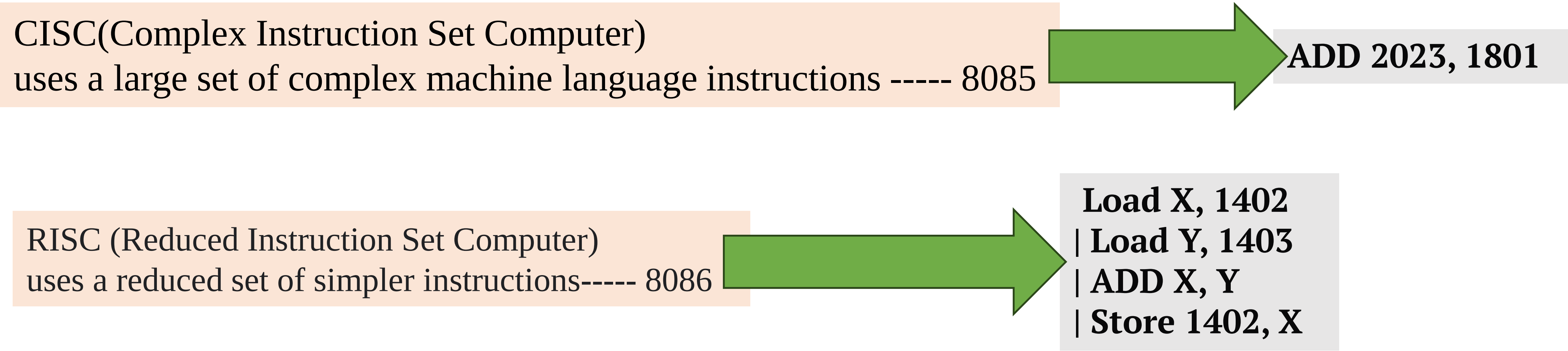
The physical address (A) = Segment×16+Offset

The value of Code Segment (CS) Register is 1012H and the value of different offsets is as follows:
BX: 2023H , IP: 0112H , DI: 5206H
Calculate the physical address of the memory location pointed by the CS register.

The physical address (A) = Base address of CS×10H+Offset

The physical address (A) = 1012×10H+ 0112H = 10232H

Introduction to RISC and CISC



CISC(Complex Instruction Set Computer)
uses a large set of complex machine language instructions ----- 8085

ADD 2023, 1801

RISC (Reduced Instruction Set Computer)
uses a reduced set of simpler instructions----- 8086

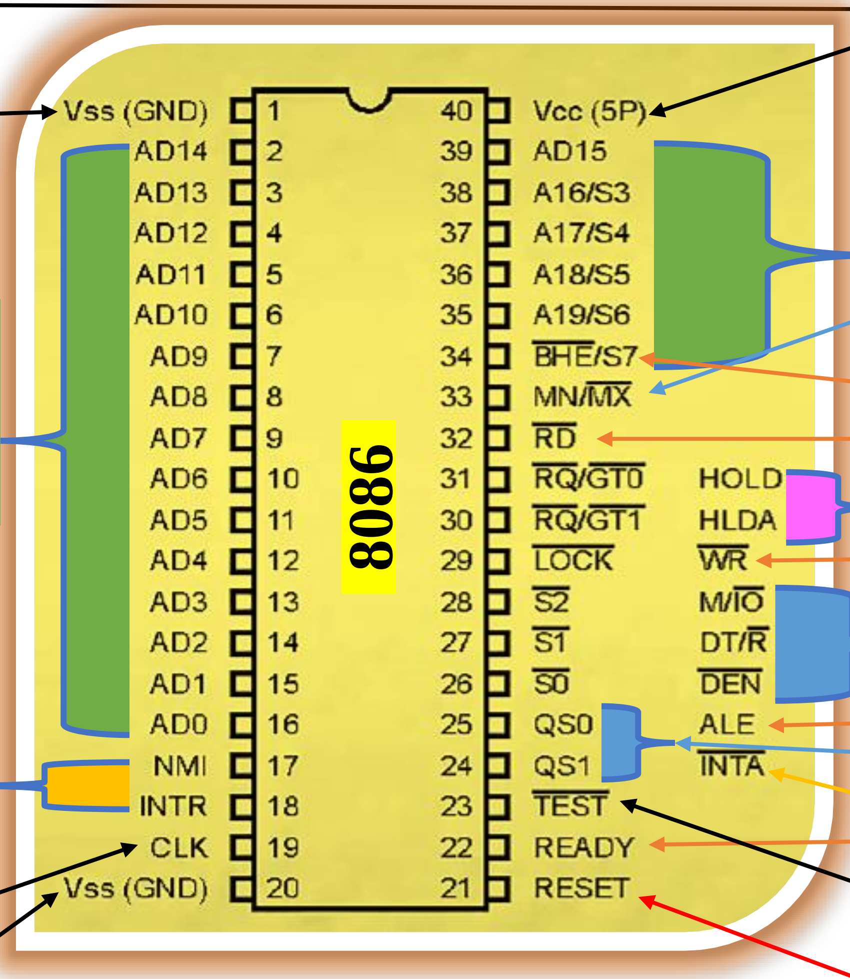
Load X, 1402
| Load Y, 1403
| ADD X, Y
| Store 1402, X

The CSIC architecture processes complex instructions that require several clock cycles for execution.	The RISC architecture executes simple yet optimized instructions in a single clock cycle.
On average, it takes two to five clock cycles per instruction (CPI).	It processes instructions at an average speed of 1.5 clock cycles per instruction (CPI).

Differences between RISC and CISC

RISC	CISC
RISC is **Reduced Instruction Cycle.**	CISC is **Complex Instruction Cycle.**
Instruction decoding is simpler than in CISC.	The task of decoding instructions is quite complex.
Focus on software	Focus on hardware
Uses only Hardwired control unit	Uses both hardwired and microprogrammed control unit
Transistors are used for more registers	Transistors are used for storing complex Instructions
Fixed sized instructions	**Variable sized instructions**
Can perform **only Register to Register Arithmetic** operations	Can perform **REG to REG or REG to MEM or MEM to MEM**
Requires **more number of registers**	Requires **less number of registers**
Code size is large	**Code size is small**
An instruction executed in a **single clock cycle**	Instruction takes **more than one clock cycle**
An **instruction fit in one word**.	Instructions are **larger than the size of one word**
Simple and limited addressing modes.	**Complex and more addressing modes.**
The **number of instructions are less** as compared to CISC.	The **number of instructions are more** as compared to RISC.
It consumes the **low power.**	It consumes **more/high power.**
RISC is highly **pipelined.**	CISC is **less pipelined.**
RISC required **more RAM.**	CISC required **less RAM.**
Here, **Addressing modes are less.**	Here, **Addressing modes are more.**

Pin description of 8086 microprocessor



Ground

Power Supply

Address/Status bus A_{16} - A_{19}

Address/Data bus
AD_0 - AD_7→Lower order byte data
AD_8- AD_{15}→Higher order byte data

Modes of operation.
High → Min Mode; Low → Max Mode

Control Signals BHE,RD, WR,ALE,READY

Direct Memory Access Signals

Status Signals(Memory& I/O control)
S_0,S_1, S_2

Interrupt Signals

Status Signals(Instruction Que control)
QS_0,QS_1

Clock frequency signal

Interrupt Signals

Idle

Reset Signal

Ground

$DT/\overline{R}$→Data Transmit/Receive
$\overline{DEN}$→Data Enable
$RQ/\overline{GT}$→Request/Grant
$\overline{BHE}$→Bus High Enable

DEN→It is used to enable Transreceiver 8286. The transreceiver is a device used to separate data from the address/data bus.

$\overline{LOCK}$ → It indicates to the other processors not to ask the CPU to leave the system bus

Status Control Signals

Status of instruction queue

- These signals provide the status of instruction queue

QS_0	**QS_1**	**Status**
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

Memory & I/O control signals

- These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals.

S_2	**S_1**	**S_0**	**Status**
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

Status Control Signals

S_3 and S_4 show that which segment is currently accessed by the microprocessor among the **four segments** present in it.

S_3	S_4	Status
0	0	Extra Segment
0	1	Stack Segment
1	0	Code Segment
1	1	Data Segment

BHE	S_7	Status
0	0	All 16bits will be accessed
0	1	Lower Byte will be accessed
1	0	Upper Byte will be accessed
1	1	Idle

S_5, when enabled, shows the presence of an interrupts in the microprocessor. So, basically, it serves as an **interrupt flag**. The signal at **S_6** shows the status of the bus master for the current operation. More simply we can say, whether the 8086 is the **bus master or any other proficient device is acting as the bus master.**

BHE' / S_7 – ***Pin number 34*** – **BHE** is an acronym for Bus High Enable. The combination of the BHE signal and **S_7** status informs about the **existence of the data on the bus**. Also, different combinations show whether the bus is containing overall 16 bit, upper byte or lower byte of the data.

Modes of operation-MinMode

Minimum Mode Pins

The following 8 pins function descriptions are for the 8086 in minimum mode; MN/ MX = 1.

$\overline{\boldsymbol{INTA}}$

It is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and T4 of each interrupt acknowledge cycle.

ALE

ALE is provided by the processor to latch the address into the 8282/8283 address latch. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

$\overline{\boldsymbol{DEN}}$

It is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. DEN is active LOW during each memory and IO access. It will be low beginning with T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. It floats to tri-state off during local bus "hold acknowledge".

$\boldsymbol{DT/\overline{R}}$

It stands for Data Transmit/Receive signal. It decides the direction of data flow through the transreceiver. In minimum mode, 8286/8287 transceiver is used for the data bus and used to control the direction of data flow through the transceiver. This signal floats to tri-state off during local bus "hold acknowledge".

$\mathbf{M}/\overline{\boldsymbol{IO}}$

This pin is used to distinguish a memory access or an I/O access. When this pin is Low, it accesses I/O and when high it access memory. $\mathrm{M}/\overline{IO}$ becomes valid in the T4 state preceding a bus cycle and remains valid until the final T4 of the cycle. M/IO floats to 3 - state OFF during local bus "hold acknowledge".

$\overline{\boldsymbol{WR}}$

Depending on the state of the signal, it indicates that the processor is performing a memory write or IO write cycle. is active for T2, T3 and Tw of any write cycle. It is active LOW, and floats to 3-state OFF during local bus "hold acknowledge ".

HOLD & HLDA

Hold indicates that another master is requesting a local bus "HOLD". To be acknowledged, HOLD must be active HIGH. The processor receiving the "HOLD " request will issue HLDA (HIGH) as an acknowledgement in the middle of the T1-clock cycle. After "HOLD" is detected as being Low, the processor will lower the HLDA and when the processor needs to run another cycle, it will again drive the local bus and control lines.

Modes of operation-Max Mode

Maximum Mode Pins

The following 8 pins function descriptions are for the 8086 in maximum mode; MN/ MX = 0.

QS_1, QS_0

These signals provide the status of instruction queue

QS_0	QS_1	Status
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

S_2, S_1, S_0

These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals.

S_2	S_1	S_0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

$\overline{LOCK}$

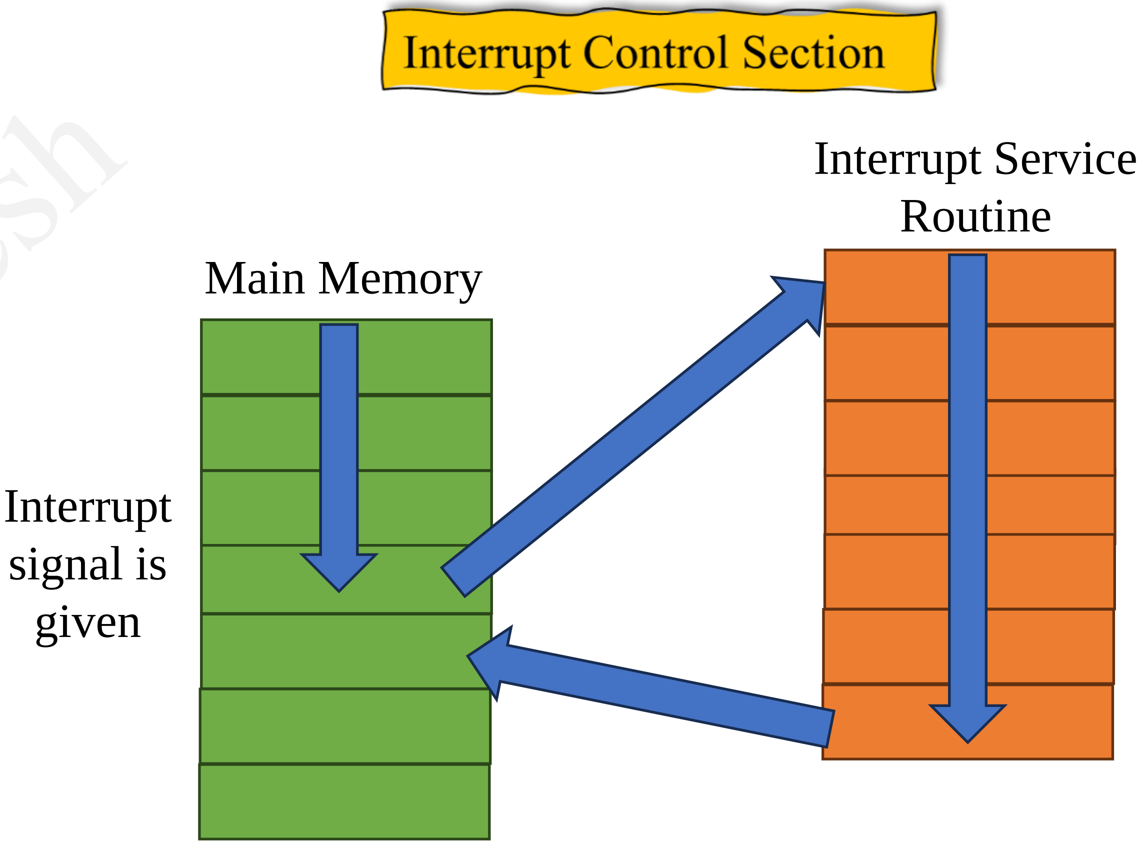
When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.

$RQ/\overline{GT}1$and $RQ/\overline{GT}0$

These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment.
$RQ/\overline{GT}0$ has a higher priority than $RQ/\overline{GT}1$.

What Exactly means Interrupt Control Section

- As the name suggests this control **interrupts a process**.
- Consider that a microprocessor is executing the main program. Now whenever the interrupt signal is enabled or requested the microprocessor shifts the control from main program to process the incoming request and after the completion of request, the control goes back to the main program.
- For example an Input/output device may send an interrupt signal to notify that the data is ready for input.
- The microprocessor temporarily stops the execution of main program and transfers control to specific special routine known as "**Interrupt Service Routine"(ISR).**
- After ISR control is transferred back to main program.
- Figure below shows the idea of communication between microprocessor and device with the help of interrupt


Interrupt Control Section
Interrupt Service Routine
Main Memory
Interrupt signal is given

Sequence of executions in 8086 when an interrupt signal enabled

When 8086 MP encounters an interrupt

1.The value of the flag register is pushed into the stack.

It means that first, the value of SP (Stack Pointer) is decremented by two then the value of the flag register is pushed to the memory address of the stack segment.

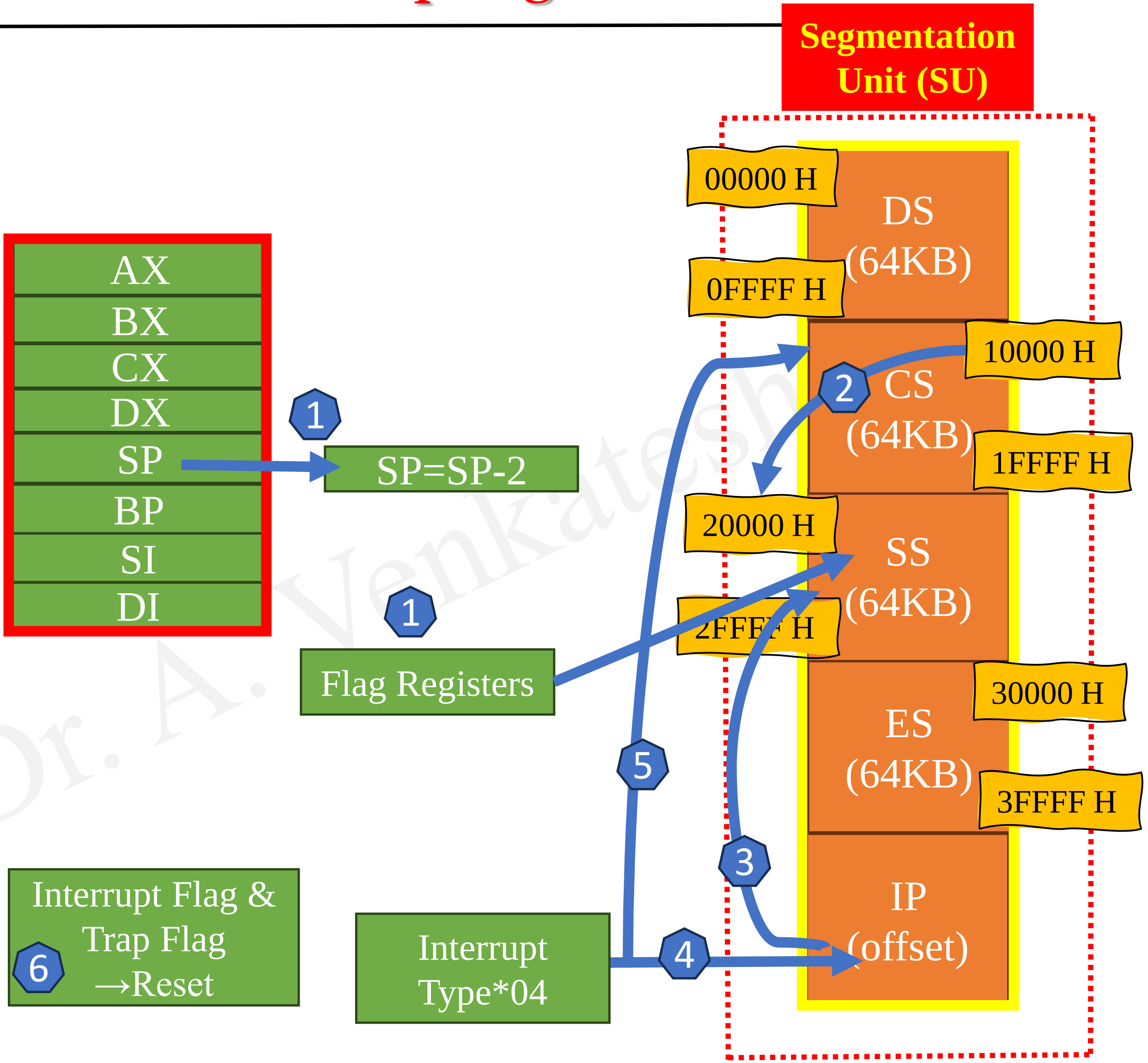
1.The value of starting memory address of CS (Code Segment) is pushed into the stack.

2.The value of IP (Instruction Pointer) is pushed into the stack.

3.IP is loaded from word location (Interrupt type) * 04.

4.CS is loaded from the following word location.

5.Interrupt, and Trap flags are reset to 0.

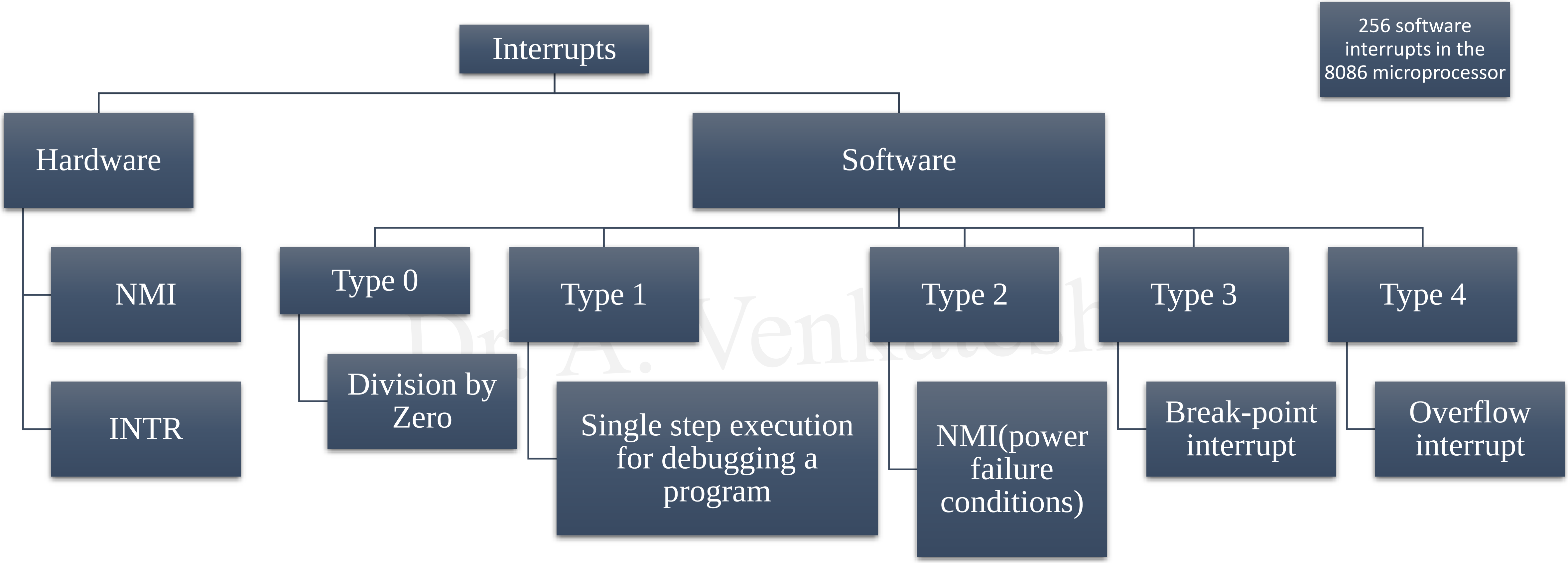


Types of Interrupts in 8086

Hardware interrupt:
Caused by any peripheral device by sending a signal through a specified pin to the microprocessor

Software Interrupts:
These are instructions inserted within the program to generate interrupts

256 software interrupts in the 8086 microprocessor



Hardware Interrupts

Hardware Interrupts

- ***NMI (Non-Maskable Interrupt):*** It is a single pin non-maskable hardware interrupt that **cannot be disabled**. It is the **highest priority interrupt in the 8086** microprocessor. After its execution, this interrupt **generates a TYPE 2 interrupt**.

IP is loaded from word location 00008 H, and CS is loaded from the word location 0000A H.

- *INTR (Interrupt Request):* It provides a single interrupt request and is **activated by the I/O port**. This interrupt can be **masked or delayed**. It is a **level-triggered** interrupt.

It can receive any interrupt type, so the value of IP and CS will change on the interrupt type received.

Sequence of operations for Hardware Interrupts

NMI (Non-Maskable Interrupt):

- **Completes the current instruction** that is in progress.
- Pushes the **Flag register values on to the stack**.
- Pushes the **CS** (code segment) value and **IP** (instruction pointer) value of the **return address** on to **the stack**.
- **IP** is **loaded from** the contents of the word location **00008H**.
- **CS** is **loaded from** the contents of the **next word location 0000AH**.
- Interrupt flag and trap flag are reset to 0.

INTR

- First completes the current instruction.
- Activates **INTA output and receives the interrupt type, say X**.
- **Flag register** value, **CS** value of the return address and **IP value** of **the return address** are pushed on to the **stack**.
- **IP** value is loaded from the **contents of word location X × 4**
- **CS is loaded from** the contents of the **next word** location.
- Interrupt flag and trap flag is reset to 0.

Software Interrupts

Software Interrupts

- These are instructions inserted within the program to generate interrupts.
- There are 256 software interrupts in the 8086 microprocessor.
- The instructions are of the format INT type, where the type ranges from 00 to FF.
- The starting address ranges from 00000 H to 003FF H. These are 2-byte instructions.
- IP is loaded from type * 04 H, and CS is loaded from the following address given by (type * 04) + 02 H.

Sequence of operations for Software Interrupts

Break point instruction

- **Flag register value** is pushed on to the **stack.**
- CS value of the return address and IP value of the return address are pushed on to the stack.
- **IP is loaded from the contents of the word location 3×4 = 0000CH**
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

INTO - Interrupt on overflow instruction

- Flag register values are pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- **IP is loaded from the contents of word location 4×4 = 00010H**
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0

External Memory Addressing

- Addressing modes are the methods used to specify the data that an instruction intends to operate on.

Specifies Given Data

- An immediate data
- An Address

Specifies Given operand

- A Register
- A Register Pair

External Memory Addressing

External Memory Addressing

Register

Both the **operands** are **registers**

Example:

MOV AX,BX

MOV AX,DX

ADD AL,BL

Immediate

The source operand is a 8 bit or 16 bit data. Destination operand can never be immediate data**.**

Example:

MOV AX,2000

MOV CL, 0AH

ADD AL,45

AND AX,0000

Displacement/Direct Addressing Mode

The effective address is **directly given in** the **instruction as** displacement

MOV AX, [DISP]

MOV CS,AX.

Register Indirect

The effective address is **in SI, DI or BX.**

Example:

Physical Address = Segment Address + Effective Address

MOV AX, [DI]

MOV AL, [BX]

MOV AX, [SI]

External Memory Addressing

External Memory Addressing

Based Index Mode

The **effective address** is **sum of base register and index register**

MOV AL, [BP+SI]

MOV AX,[BX+DI]

Indexed Mode

The effective address is **sum of index register and displacement.**

MOV AX, [SI+2000]

MOV AL, [DI+3000]

Based Mode

The effective address is the **sum of base register and displacement**

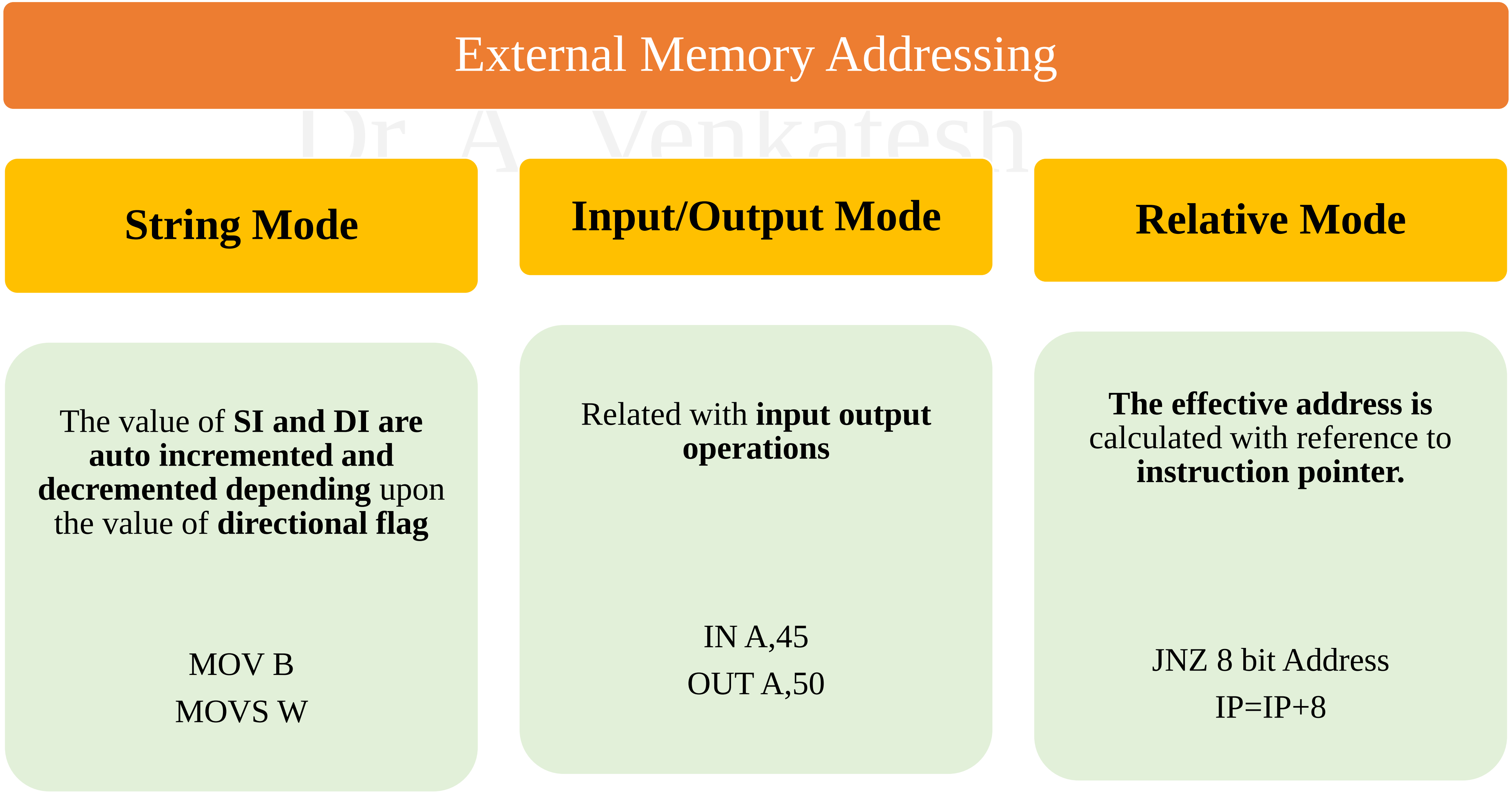
MOV AL, [BP+0100]

Based Indexed displacement Mode

The effective address is the **sum of index register, base register and displacement**

MOV AL, [SI+BP+2000]

External Memory Addressing



External Memory Addressing

String Mode

The value of **SI and DI are auto incremented and decremented depending** upon the value of **directional flag**

MOV B

MOVS W

Input/Output Mode

Related with **input output operations**

IN A,45

OUT A,50

Relative Mode

The effective address is calculated with reference to **instruction pointer.**

JNZ 8 bit Address

IP=IP+8