19CSE303-Embedded Systems
2022 Offline CSE Batch

Evaluation Policy

****L-T-P-C: 3-0-3-4**

Assessment- 65:35	Internal -65	External -35
Mid Term Exam	20	
Continuous Assessment Theory (CAT)	15	
Continuous Assessment Lab (CAL)	30	
End Semester Exam		35

- **Continuous Evaluation Theory -15%**
  - 3 Assignment + Class Note Submission– 7%
  - 2 Class Test - 8% (written exam)
- **Continuous Evaluation Lab -30%**
  - Group Project – 10%
  - Lab sheets –5% (based on submission)
  - 2 Internal Exam + Viva - 15% (P1-5%+P2-10%)

End Semester Exam-35%

Text Book(s)

1. Furber SB. ARM system-on-chip architecture. pearson Education; 2000.
2. Martin T. The Insider's guide to the Philips ARM7-based microcontrollers. Coventry, Hitex, UK, Ltd. 2005.

Reference(s)

1. Valvano JW. Embedded Systems: Introduction to ARM Cortex-M Microcontrollers.
2. Jonathan W. Valvano; 2016. Valvano JW. Embedded microcomputer systems: real time interfacing. Cengage Learning; 2012
3. https://courses.edx.org/courses/course-v1:UTAustinX+UT.6.10x+1T2017/course/
4. https://courses.edx.org/courses/course-v1:UTAustinX+UT.6.20x+2T2018/course/

Definition

- **Embed**:
  – ***to enclose closely in a surrounding mass.***
- **System:** implicitly a ***controlling*** system.

➢ **System:** Anything which accepts input, processes it and presents the output in the required format can be a system.

Embedded systems (ES) = **information processing systems embedded into a larger product**

It is more than a Computer, It is a “Complete System”

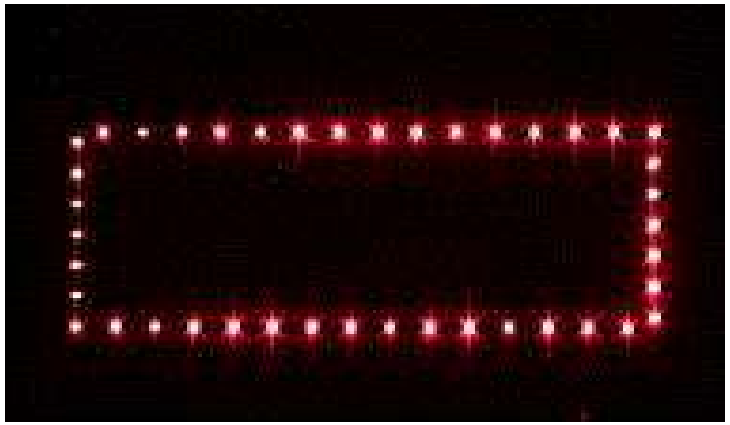
What is an Embedded System?

- “An embedded system can be defined as those control systems which are designed either by microprocessor or microcontroller for a specific tasks. ” OR
- “An embedded system is some combination of computer hardware & software, either fixed in capability or programmable, that is specifically designed for a particular kind of application device. ” OR
- “An embedded system is the one that has computer hardware with software embedded in it as one of its most important components.”

A “short list” of embedded systems

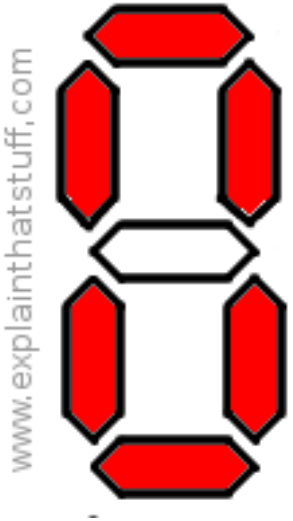
Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems
Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR’s, DVD players
Video game consoles
Video phones
Washers and dryers

Embedded System Daily Applications



Moving message display

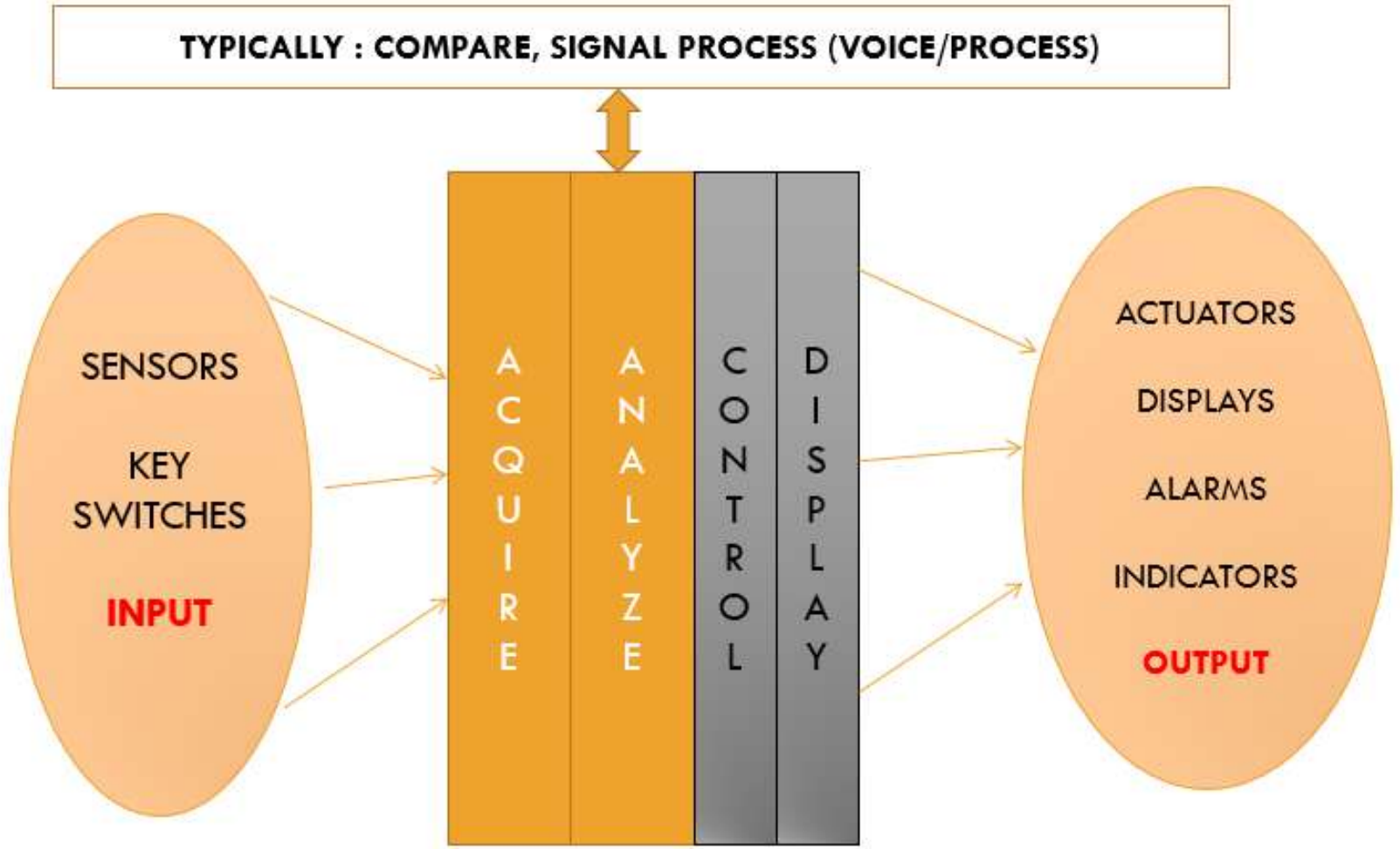
Digital clock



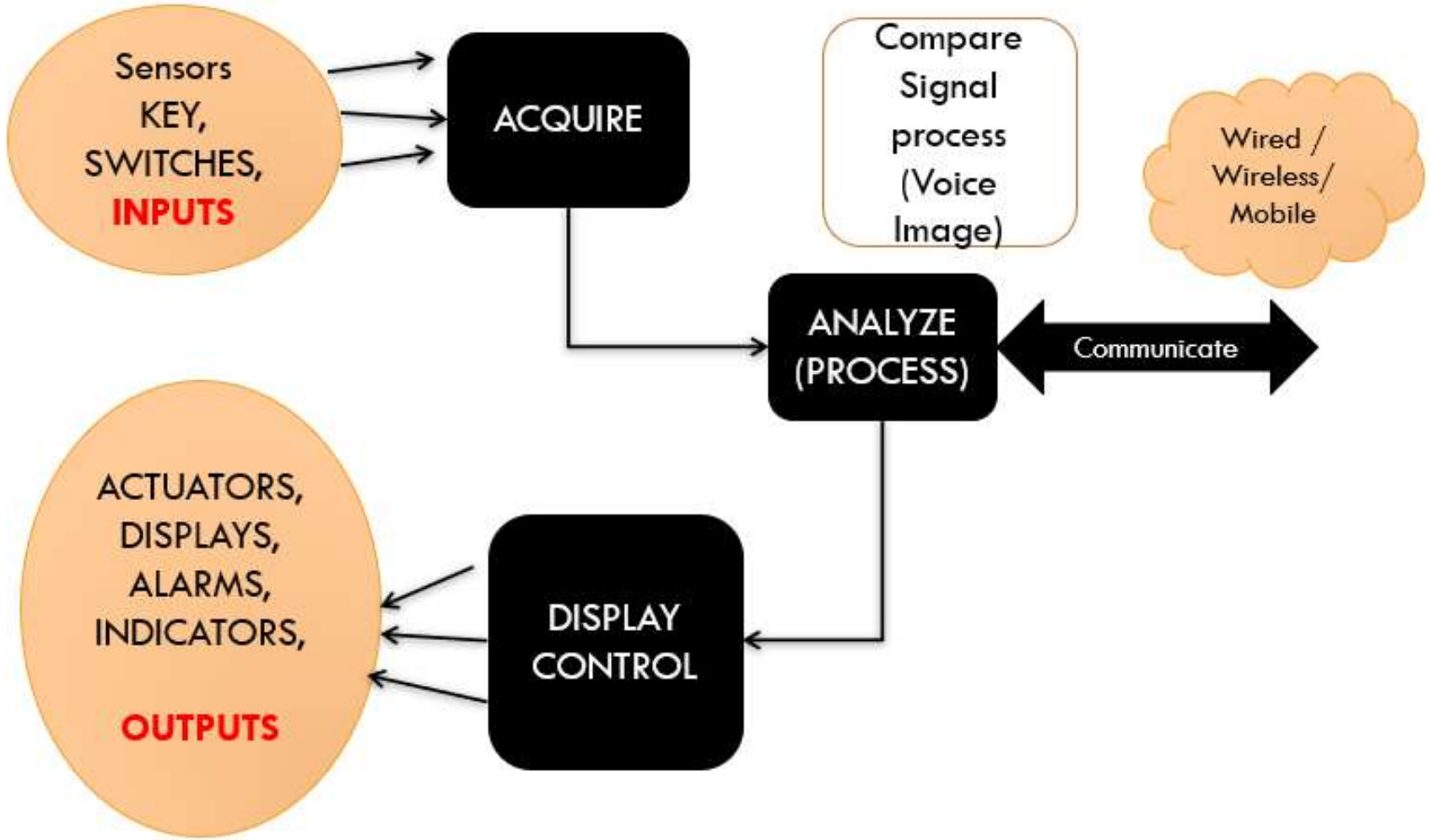
7 segment display

Traffic Light

A Diagrammatic representation of Embedded Systems Architecture – Presented here!



A Diagrammatic representation of Embedded Systems Architecture – Presented here!



Essential Components of Embedded Systems

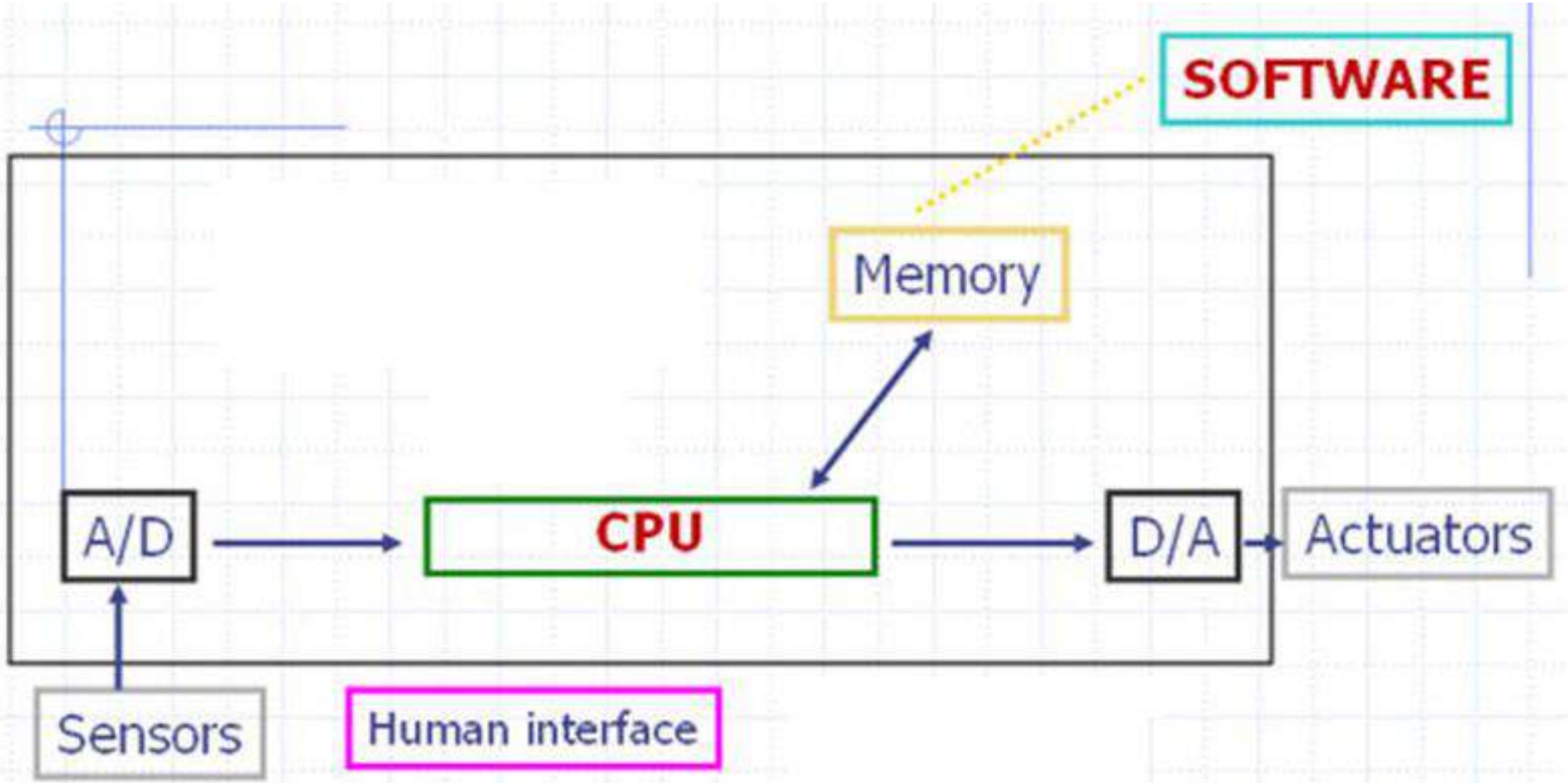
- **The ES has three major components and they are classified as follows:**
  - Hardware
  - Software
  - RTOS (Real Time OS)

- **Hardware:**
  - Microprocessor / Microcontroller
  - Sensors
  - Converters (A/D and D/A)
  - Actuators
  - Memory (On-chip and Off chip)
  - Communication path with the interacting environment

- **Software:**
  - Application software that can perform a series of task.

- **RTOS:**
  - Defines the way the system works
  - Supervises the application software
  - Provides a mechanism to let the processor to run a process as per scheduling (Process scheduling)
  - Perform Context switching between the processes

Block Diagram

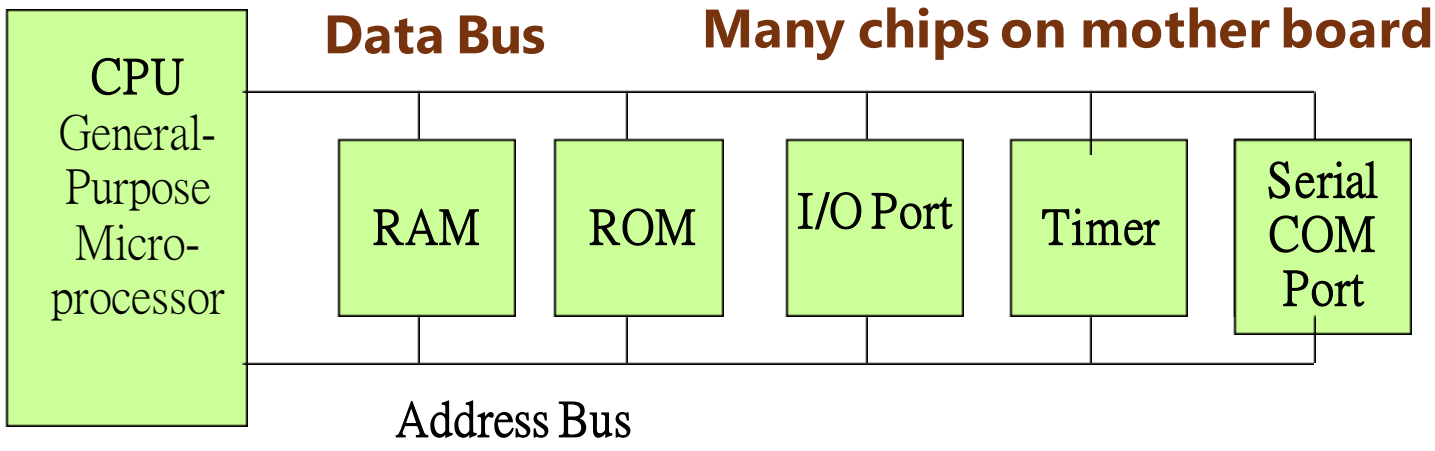


Important Units of microprocessor/controller

- CPU: Central Processing Unit
- I/O: Input /Output
- Bus: Address bus & Data bus
- Memory: RAM & ROM
- Timer
- Interrupt

Microprocessor

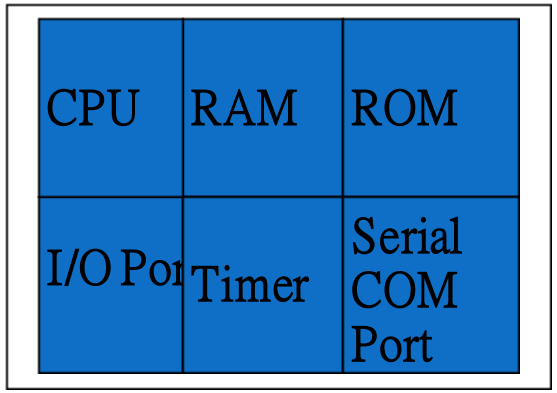
- General-purpose microprocessor
- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself
- Example--Intel's x86: 8086,8088,80386,80486, Pentium



General-Purpose Microprocessor System

Microcontroller

- A smaller computer.
- On-chip RAM, ROM, I/O ports...
- Example:- Motorola's 6811, Intel's 8051 and PIC 16X



A single chip

Microcontroller

Microprocessor v/s Microcontroller

Microprocessor

- CPU is stand-alone, RAM, ROM, I/O, timer are separate
- Designer can decide on the amount of ROM, RAM and I/O ports.
- Expensive
- General-purpose
- Examples:-
  - 8085,8086 microprocessors
  - Motorola6800,
  - Intel's 8086, etc.

Microcontroller

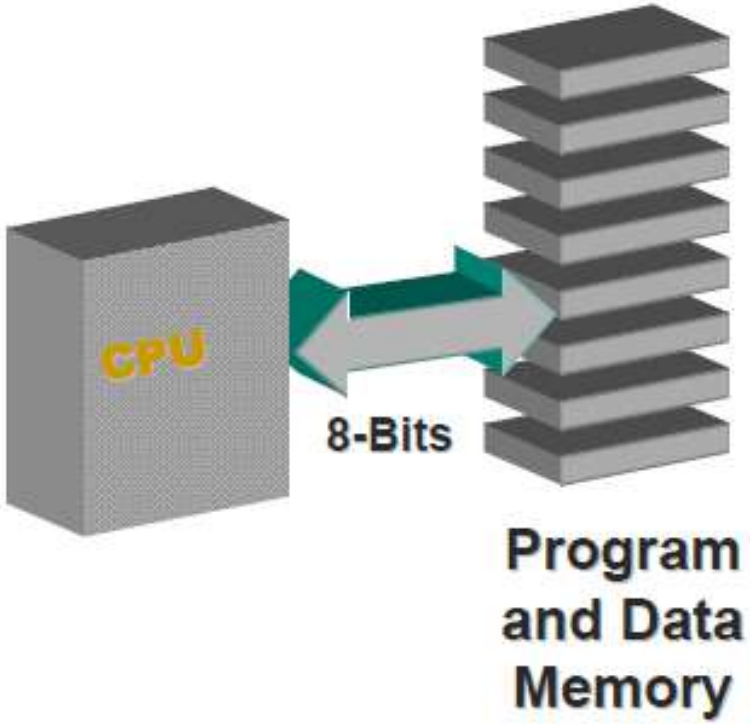
- CPU, RAM, ROM, I/O and timer are all on a single chip
- Fixed amount of on-chip ROM, RAM, I/O ports
- For applications in which cost, power and space are critical
- Single-purpose
- Examples:-
  - 8051,
  - PIC mc,
  - Motorola
  - MC's, Phillips, etc.

Main Architecture

- Embedded processors are constructed into 2 main architecture
  - Von Neumann
  - Harvard

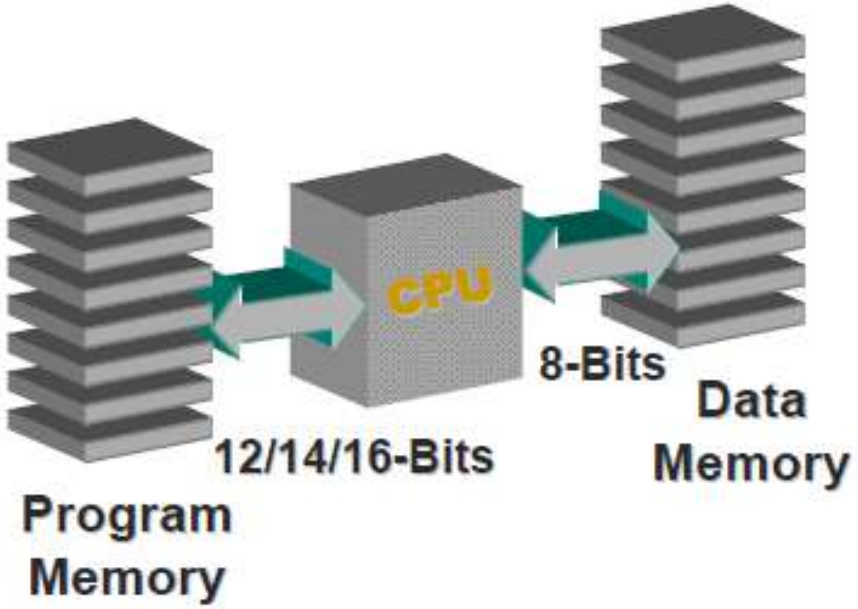
Von Neumann Architecture

- Fetches instructions and data from same memory
- Limits operating bandwidth



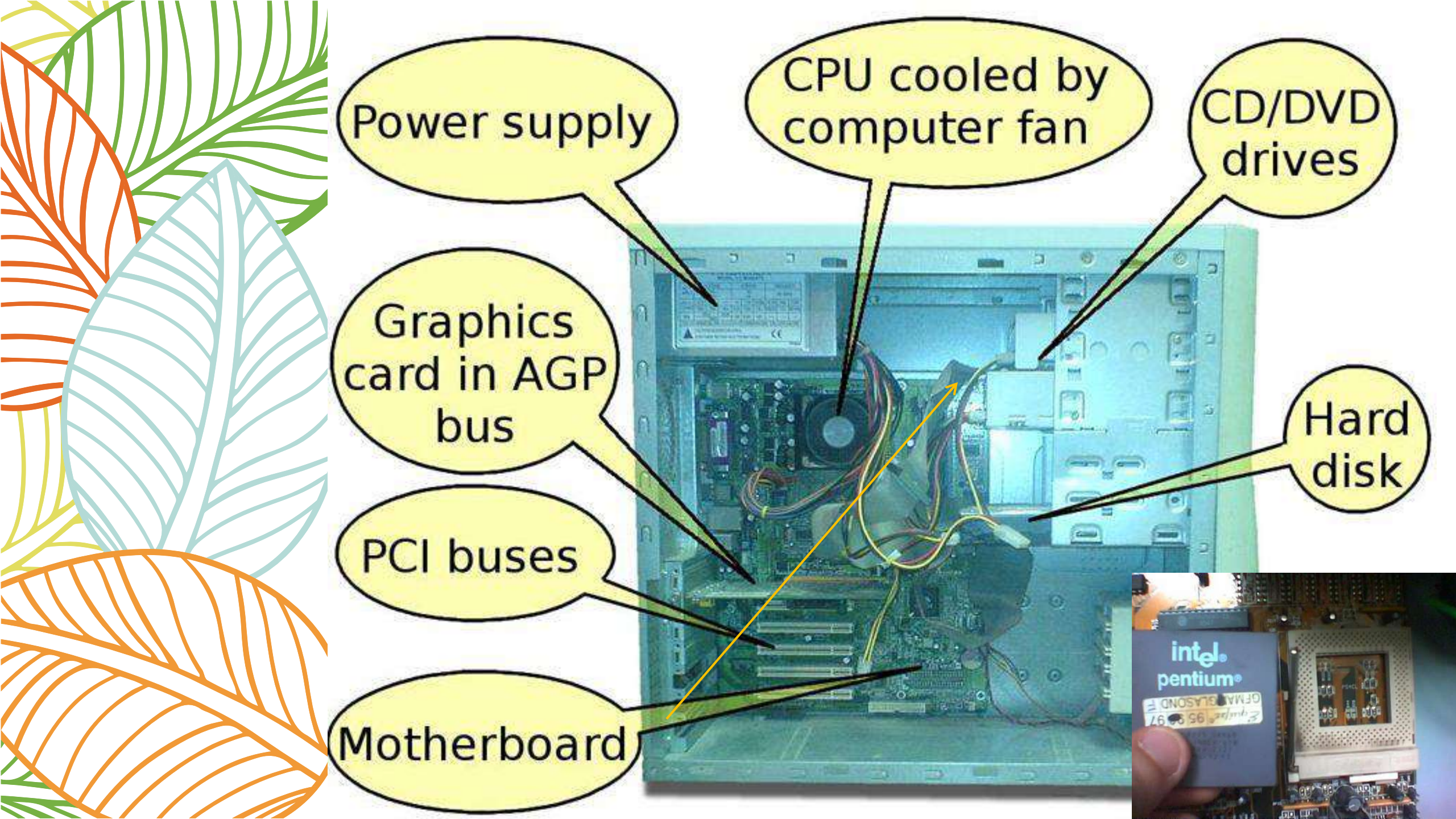
Harvard Architecture

- Two separate memory spaces for instruction and data
- Increases throughput
- Different program and data bus widths are possible



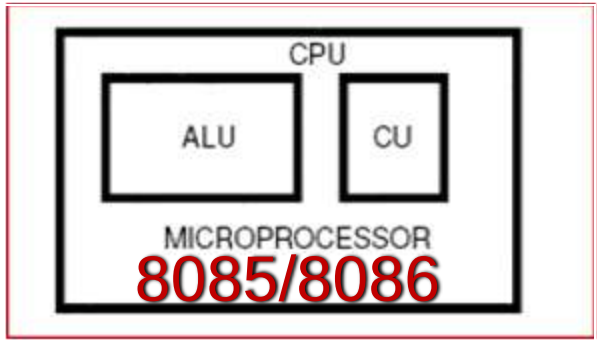
Thank You

Architecture of Microprocessors
Case Study: 8085


Power supply
CPU cooled by computer fan
CD/DVD drives
Graphics card in AGP bus
Hard disk
PCI buses
Motherboard
intel
pentium

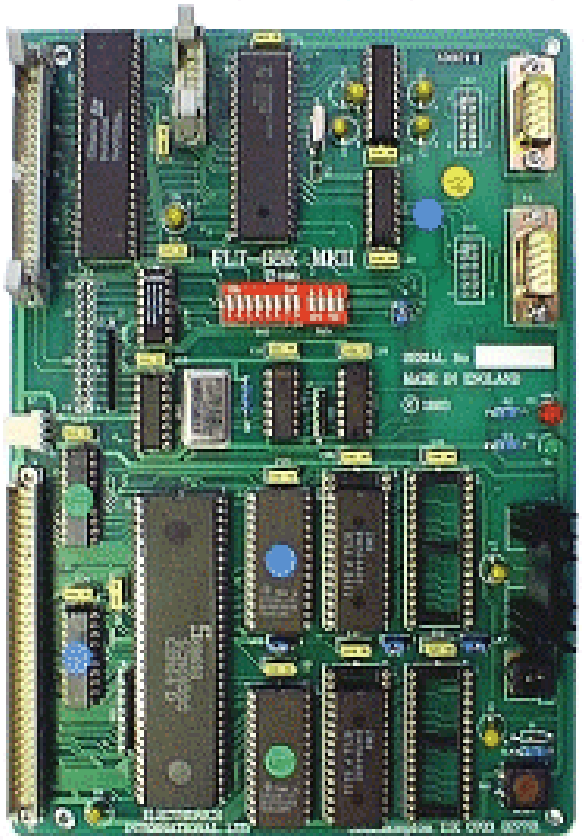
8251 (USART)

8255 (Parallel interface)



8257 (DMA Controller)

8254 (Timer)

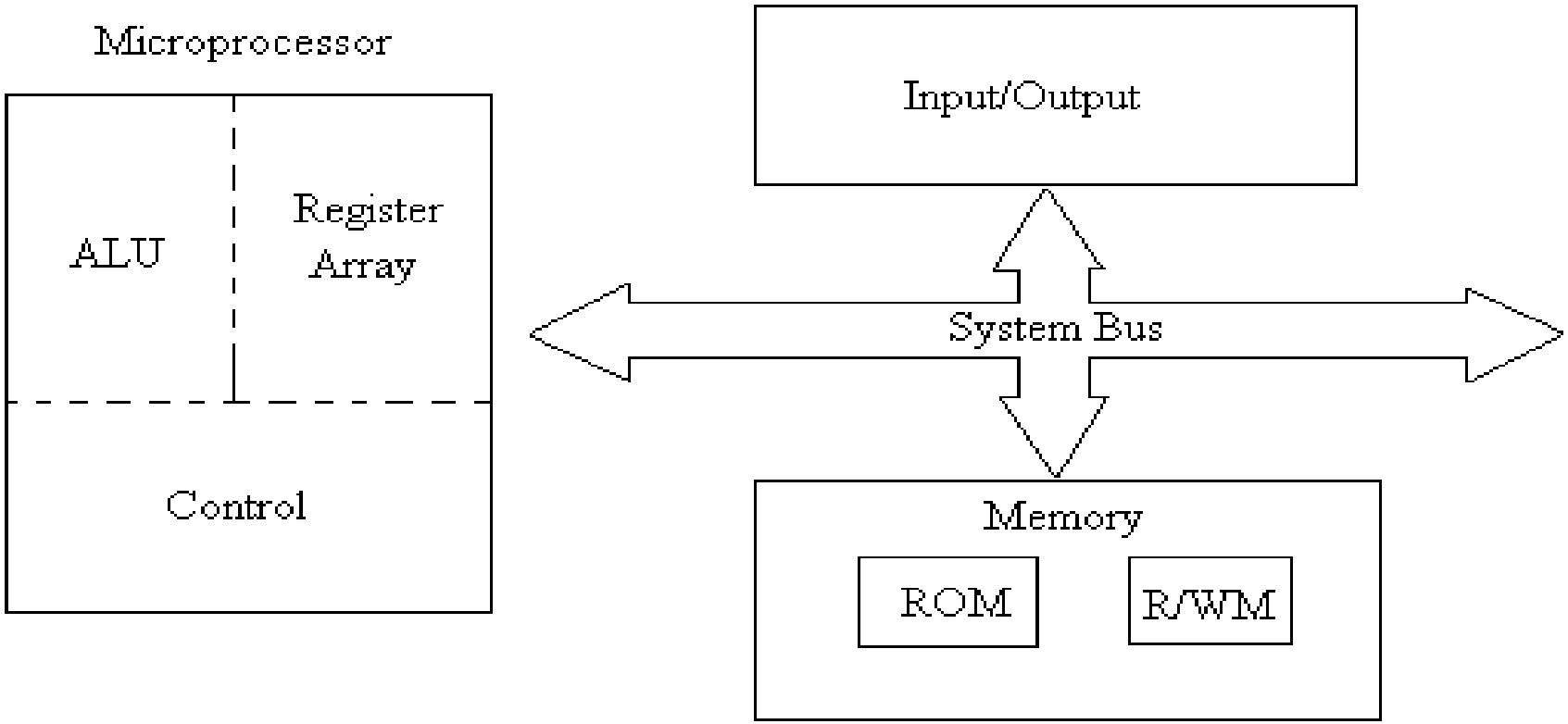


MICROPROCESSOR Based Development Board

What is a Microprocessor Based System?

- **Microcomputer is a microprocessor based system.**
- **Includes :**
  - **Microprocessor**
  - **Input/Output**
  - **Memory**
- **Components are connected through a communication path called a bus**
- **How we can differentiate microcomputer and a microprocessor?**
  - uP is one component of the microcomputer
  - The CPU functions of the microcomputer are performed by the uP
- **The term peripherals is used for I/O devices.**

uP based System Architecture with Bus Architecture



What is a microprocessor?

- **Microprocessor :-**
  - multipurpose,
  - programmable,
  - clock driven,
  - register-based electronic device
  - that reads binary instruction from a storage device called memory,
  - accepts binary data as input and
  - process data according to the instructions, and
  - provides results as output.
- Manufactured by using LSI or VLSI
- Microprocessor operation is similar to human brain
- **How we can differentiate CPU & uP ?**
- CPU is implemented on one or more circuit boards to perform the computing functions
- uP is similar to CPU
- uP - CPU on a single chip

Main 3 segments of uP

- **ALU :-** This area of uP will perform various computing functions such as arithmetic operations (Addition & Subtraction) and logic operations (AND/OR/EXOR)

- **Register Array :-** This area of uP consists of various register; B,C,D,E,H & L. Store data temporarily during execution of a program and are accessible to the user through instruction

- **Control Unit :-** Provides necessary timing and control signals to all the operations. Controls the flow of data b/w uP and memory and peripherals.

Memory

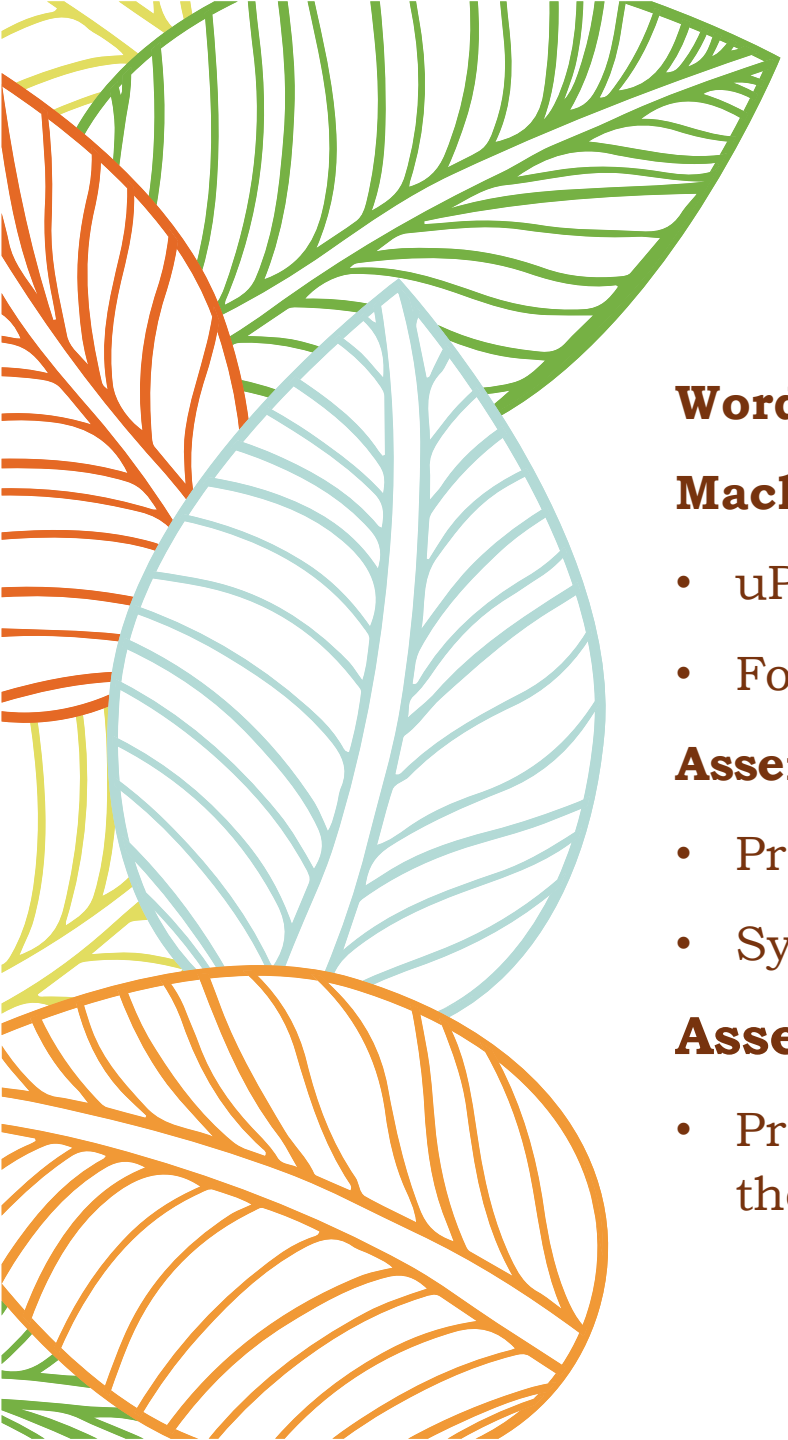
- Memory stores binary information as instruction and data
- 2 sections
  - **Read only Memory (ROM)**
    - Program in ROM can only be read, not altered.
    - Store programs that don't need alteration
  - **Read/Write Memory (RAM)**
    - User Memory
    - Stores user program & data
    - Information stored can be easily read and altered
- **Consider the monitor program, generally stored in ROM**
  - This program interprets the information entered through a keyboard and provides equivalent binary to the uP.
  - The monitor programs monitors the Hex Key and stores data in R/W memory

Input/ Output & System Bus

- **Input/Output :-**
- Also known as peripherals
- Communicating with the outside world through I/O
- Inputs devices :- keyboard, switches, ADC
    - Transfer binary information from outside world to the uP
- Output devices:- LED, CRT, Video screen (monitor), XY Plotter, Magnetic Tape, DAC
    - Transfer data from uP to the outside world
- **System Bus:-**
- Communication path between the uP and peripherals
- Bus - Group of wires to carry bits
- All peripherals share the same bus
- uP communicates with only one peripheral at a time
- Timing is provided by the CU of the uP.

How does a uP works?

- **Fetch, Decode, Execute**
  - uP fetches the first instruction from the memory
  - Decodes it
  - Execute the instruction
- Uses the system bus to fetch the instruction and data from the memory in the entire process
- Uses registers to store data temporarily
- Performs the computing function in the ALU
- Sends out the result in binary using the same bus lines to the o/p.



Terminology

Word :- No: of bits uP recognizes and process at a time

Machine Language:-

- uP communicates and operates in 0's & 1's
- For communication, give the instruction in binary language

Assembly language

- Programmer writes the program in words
- Symbolic code for each instruction, called mnemonics

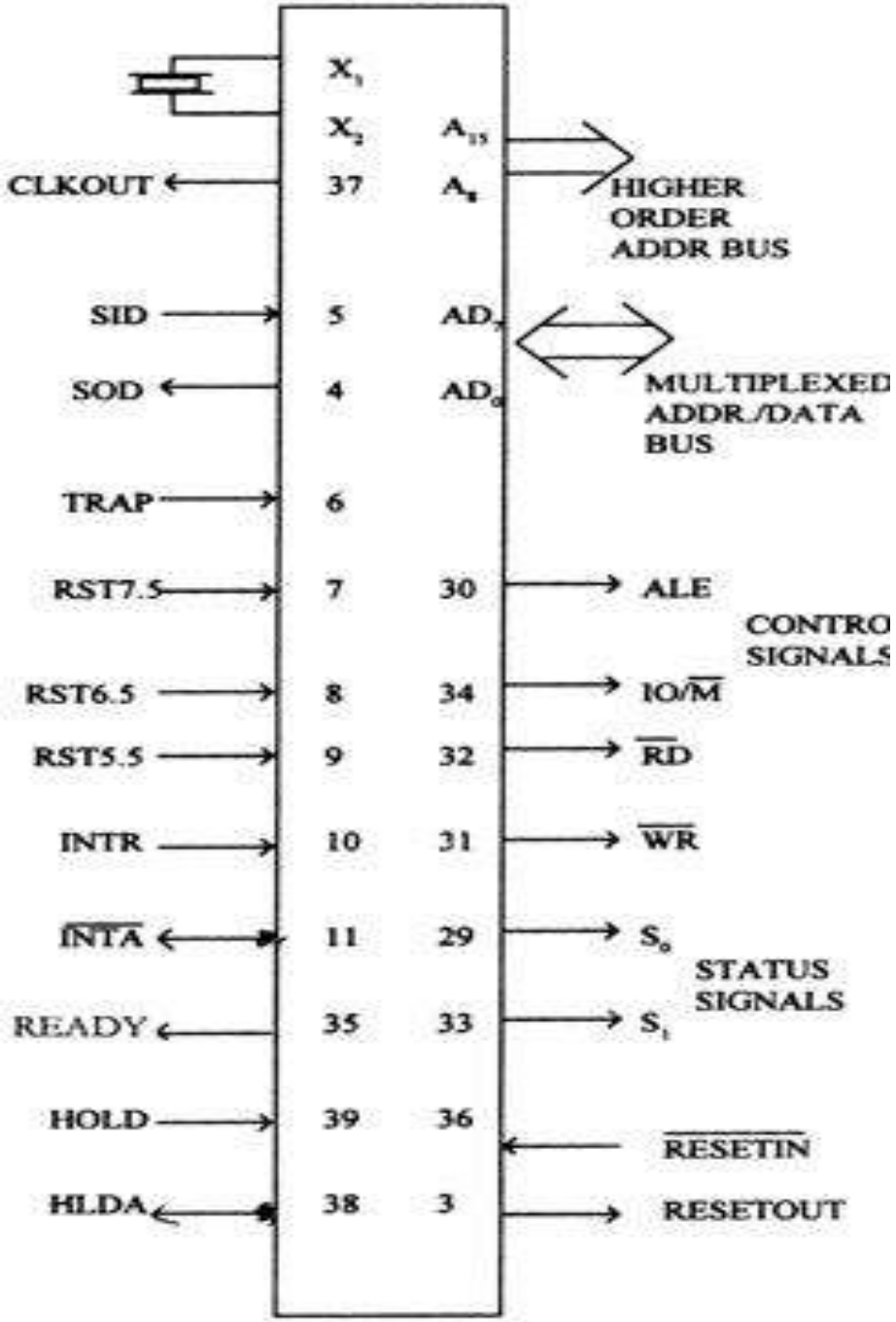
Assembler :-

- Program that translates the mnemonics entered from the keyboard into the corresponding binary machine codes of the uP

INTEL 8085-Features

- It is an 8 bit general purpose microprocessor.
- It is a single chip N-MOS device with 40 pins.
- Capable for addressing 64K of memory
- It has multiplexed address and data bus.(AD0-AD7).
- It works on 5 Volt dc power supply.
- The maximum clock frequency is 3 MHz - (8085A-5MHz)
- It provides 74 instructions with 5 different addressing modes.

- **Signals can be classified into 6 groups**:
  - Address Bus
  - Data Bus
  - Control and Status signals
  - Power supply and frequency signals
  - Externally initiated signals
  - Serial I/O ports



- **Address Bus**
  - 16 address lines
  - A15 – A8 & AD7 – AD0
  - A15 – A8 are unidirectional, called high order address
  - AD7 – AD0 for dual purpose, they carry both address bits and data bits

- **Multiplexed Address/Data Bus**
  - AD7-AD0 are bidirectional
  - Low order address bus or data bus
  - Earlier part of cycle, acts as low order address bus
  - Later part, acts as data bus

- **Control & Status Signal**
  - 2 control signals (RD & WR)
  - 3 Status signals (IO/M, S1 & S0)
  - 1 Special Signal (ALE)

- **Externally Initiated Signals**
  - Includes 5 Interrupt signals
    - INTR, INTA, RST 7.5, RST 6.5, RST 5.5, TRAP
  - 3 DMA (Direct memory Access) Controller signals
    - HOLD, HLDA, READY
  - RESET IN
  - RESET OUT

- **Power Supply and Clock Frequency**
  - Vcc - +5V power supply
  - Vss – Ground Reference
  - X1and X2 are the inputs from the crystal or clock generating circuit.
  - The frequency is internally divided by 2.
  - So, to run the microprocessor at 3 MHz, a clock running at 6 MHz should be connected to the X1 and X2 pins. (Crystal of 6MHz frequency should be connected)
  - CLK OUT – Clock Output
  - An output clock pin to drive the clock of the rest of the system.

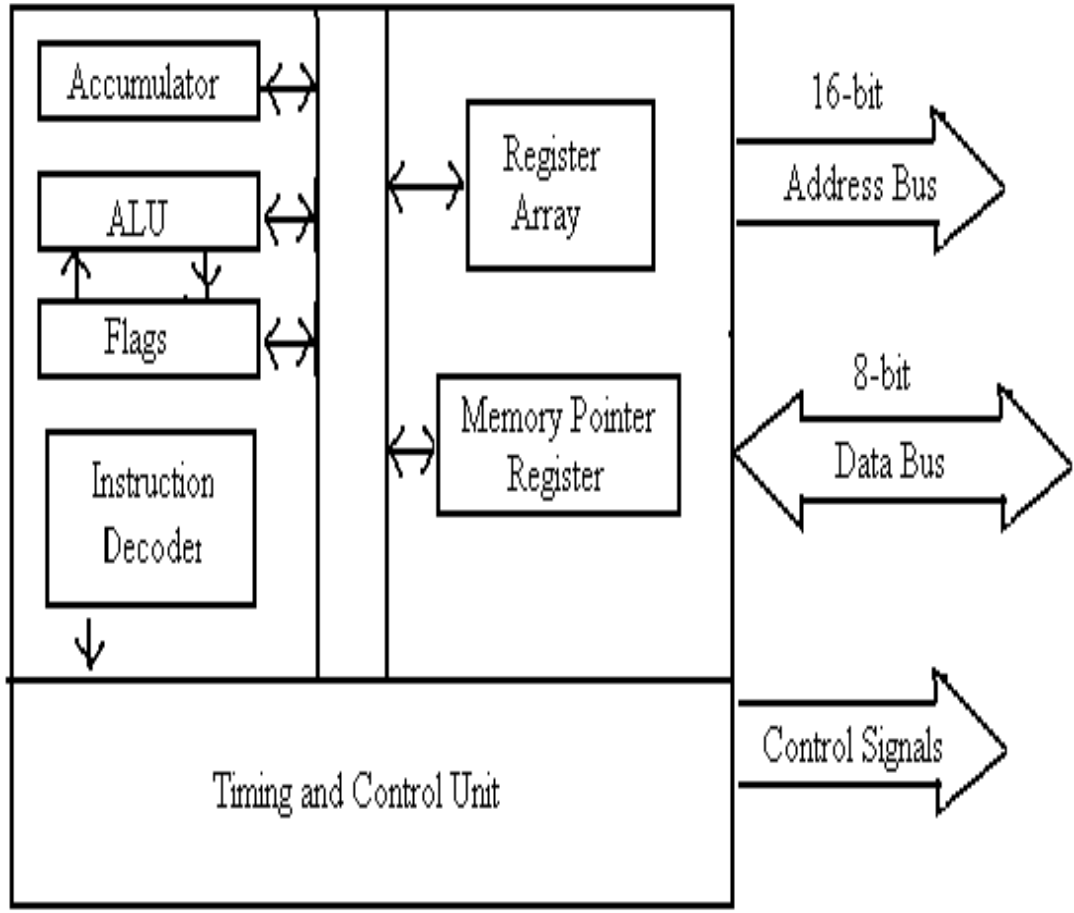
- **Serial I/O Ports**
  - SID – Serial Input Transmission
  - SOD – Serial Output Transmission

8085 Models

- Hardware Model (physical electronics components)
- Programming Model (information needed to write program)

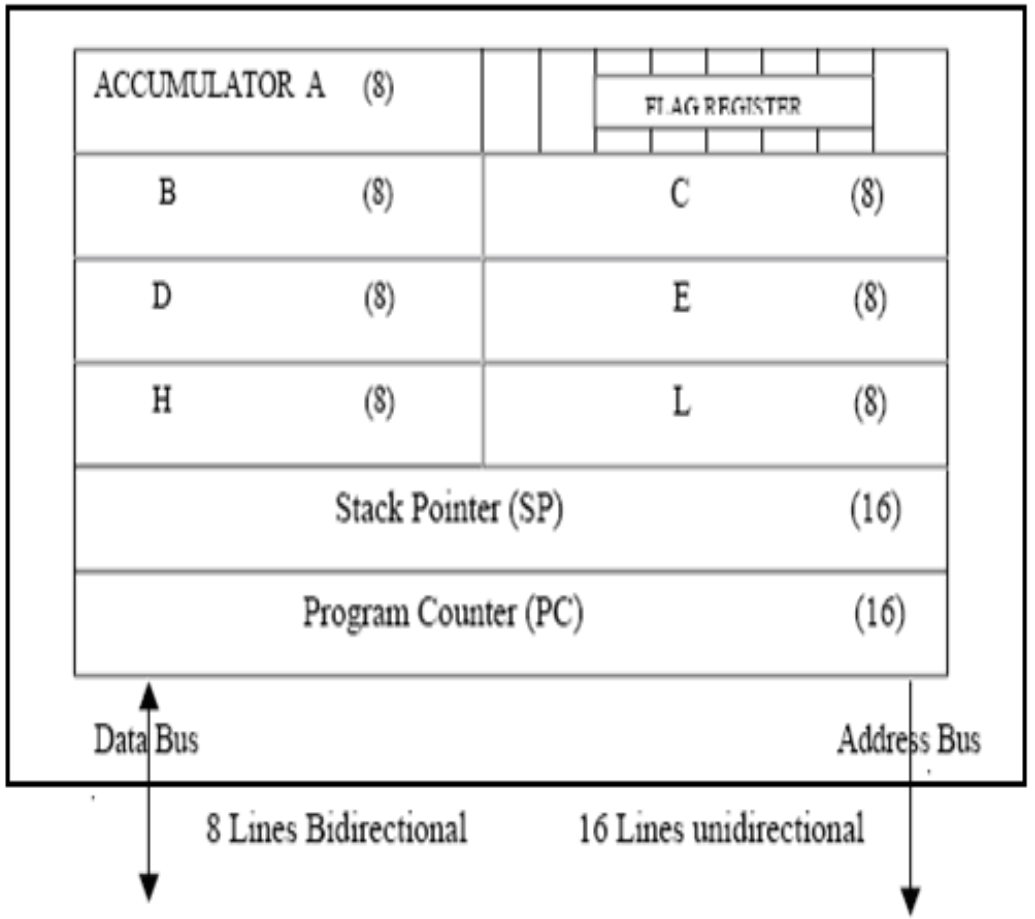
8085 Hardware Model

- **2 major segments**
  - **Segment 1**
    - ALU
    - 8-bit Accumulator
    - Instruction Decoder
    - Flags
  - **Segment 2**
    - 8-bit & 16 bit Register



8085 Programming Model

- Includes some segments of ALU and registers
- Includes information critical for writing assembly language
- 6 GPR (General Purpose Registers)
- 1 Accumulator
- 1 Flag register
- 16-bit Program Counter
- 16-bit Stack Pointer



Program Counter

- 16–bit register hold memory address
- PC is used to sequence the execution of the instruction
- Function of PC is to point to the memory address from which the next byte is to be fetched
- When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location

Stack Pointer

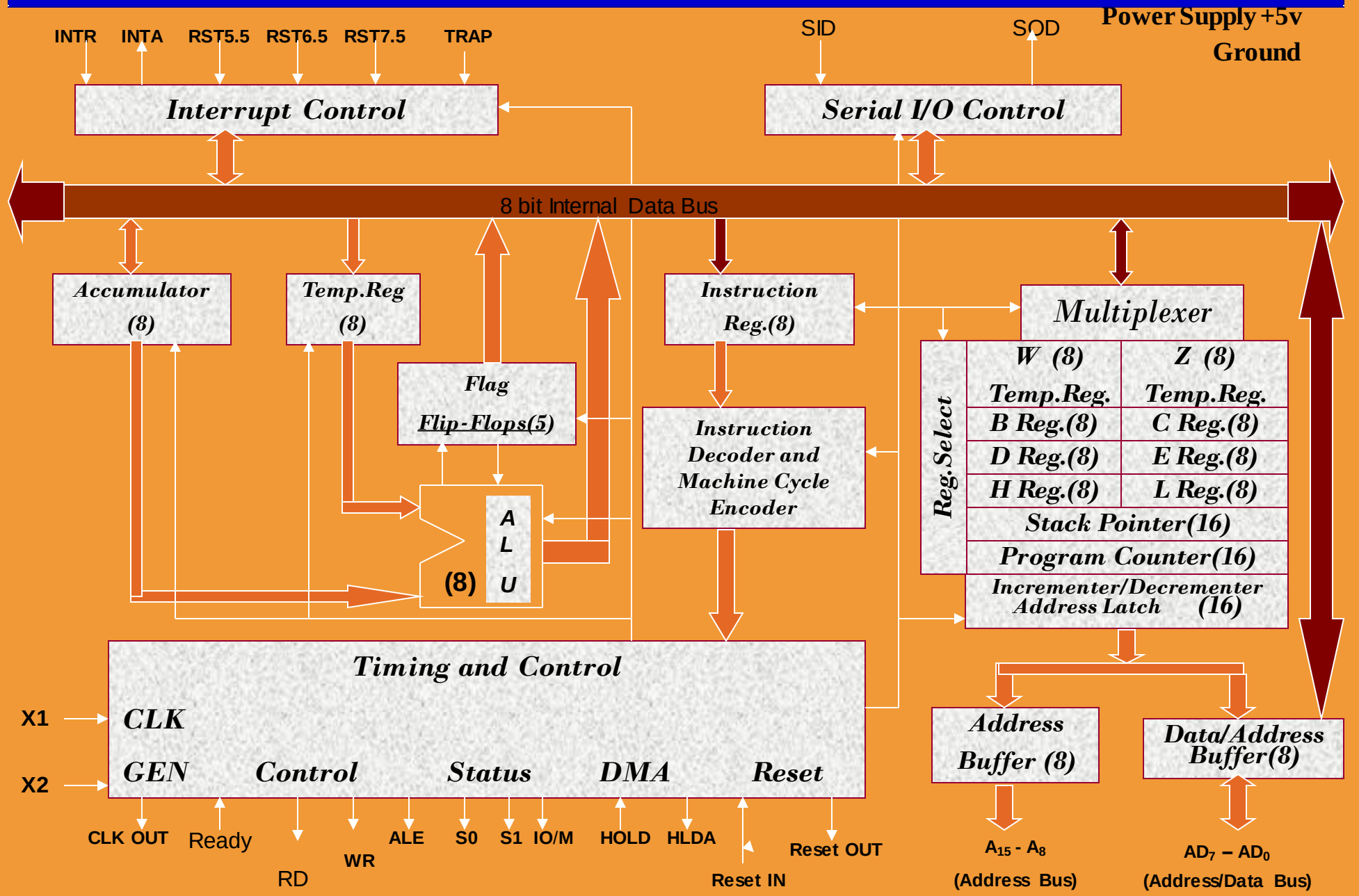
- 16-bit register used as a memory pointer
- It points to a memory location in R/W memory, called the stack
- The beginning of the stack is defined by loading 16-bit address in the stack pointer.
- The stack is the sequence of memory locations defined by the programmer.
- The stack is used to save the content of a register during the execution of the program.

8085 Architecture

- ALU
- Timing & Control Unit
- Instruction Register and Decoder
- Register Array
- Interrupt Control
- Serial I/O Control

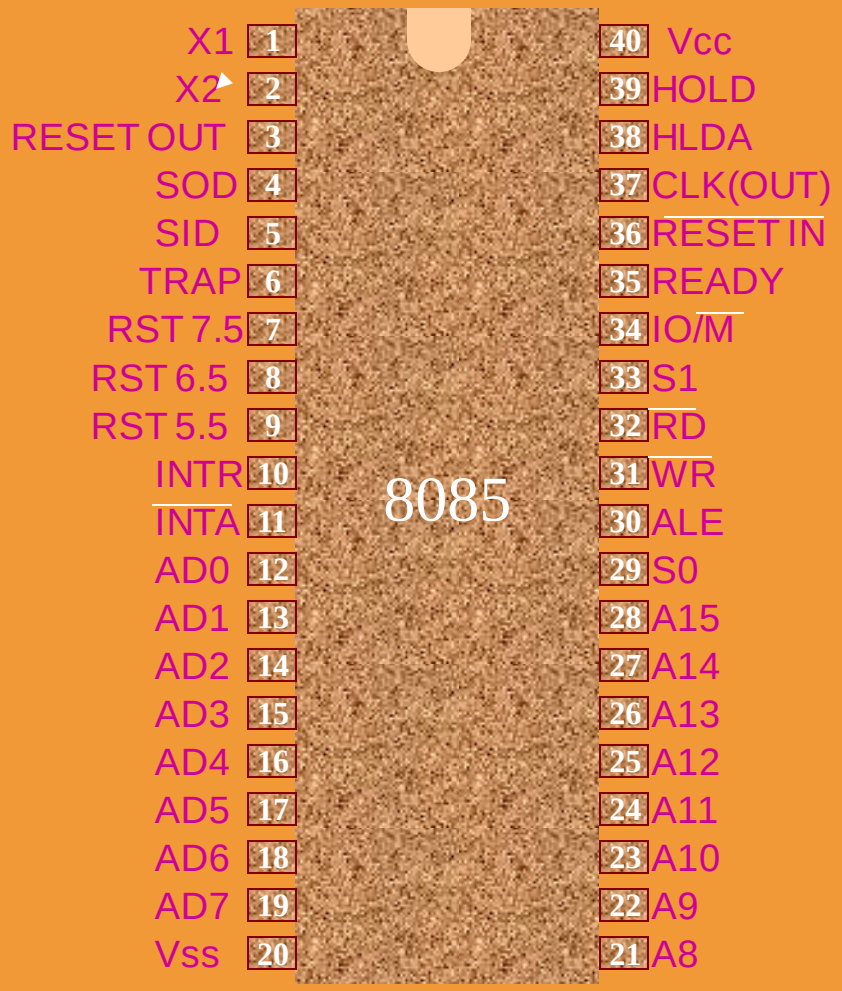
8085 ARCHITECTURE

Aum Amriteswaryai Namah:



Aum Amriteswaryai Namah:

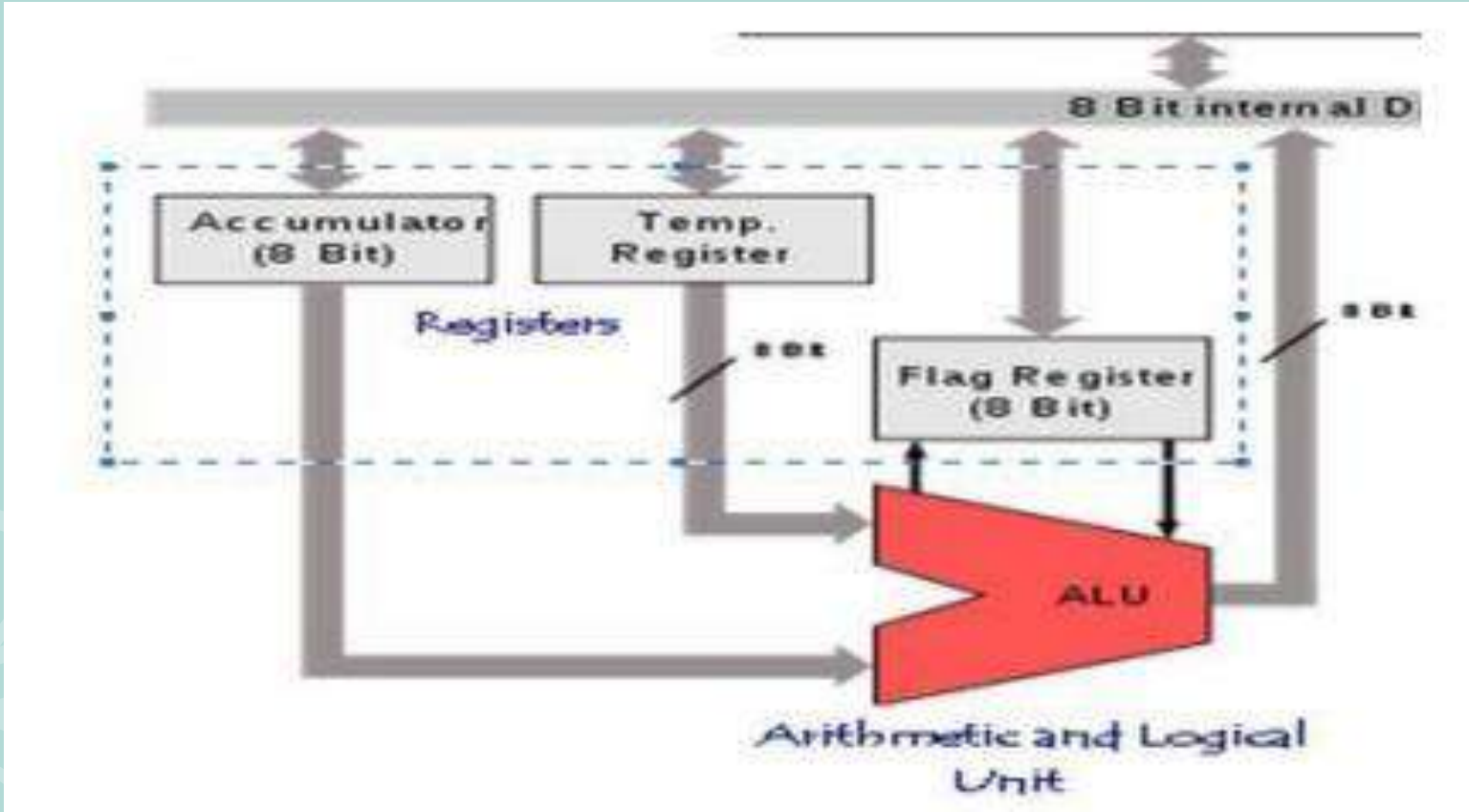
PIN DIAGRAM



S1	S0	
O	O	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

ALU (ARITHMETIC AND LOGICAL UNIT)

- It is referred to be as ALU which acts as a backbone for any arithmetic and logical operation as addition, subtraction, division, multiplication, AND, OR, NOT etc.
- Once the operation is performed, the result will be stored in accumulator, which is again a register.

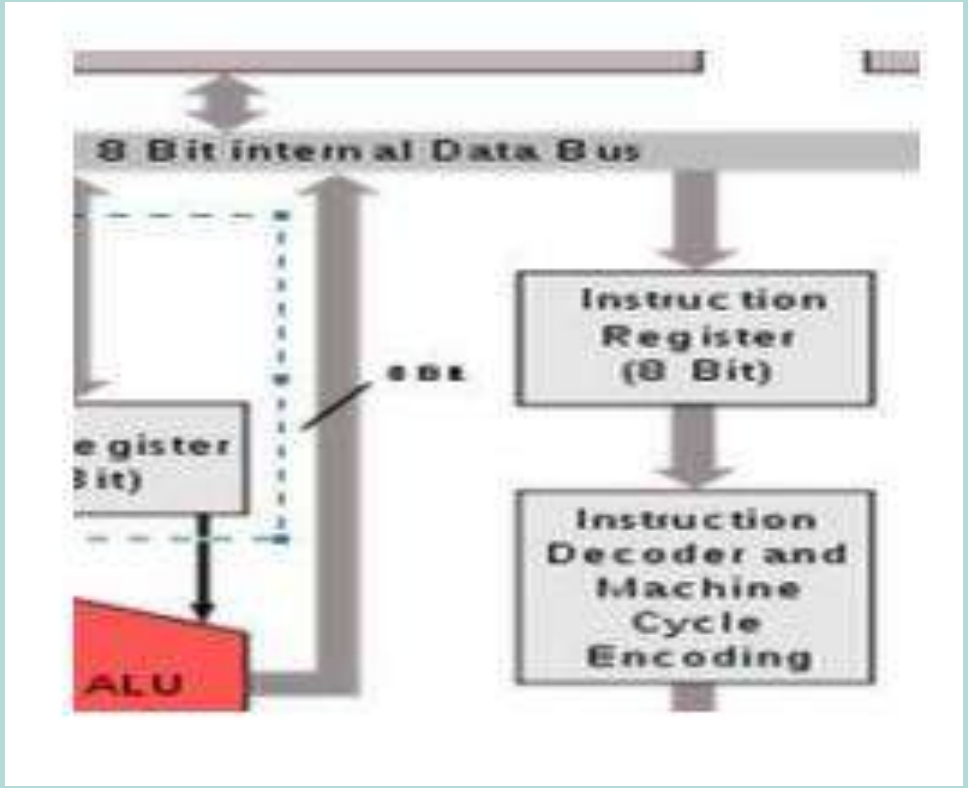


Timing and Control Unit

- Synchronizes all the uP operations with the clock
- Generates control signal necessary for the communication between uP and peripherals
- Two pins HOLD and HLDA used for DMA(Direct Memory Access). DMA controller sends a request by making Bus request control line high. CPU has received the HOLD request. HLDA will set high and in the next clock cycle after processing the data HLDA become low after HOLD signal is removed.

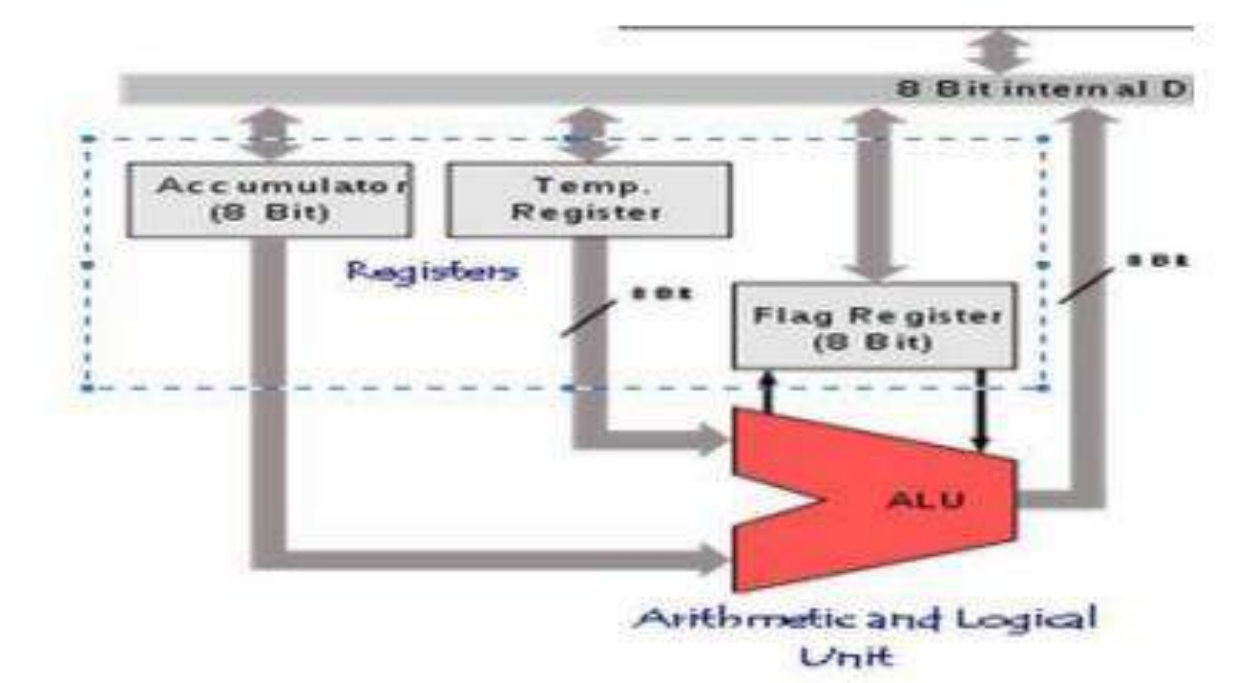
Instruction Register and Decoder

- Instruction register is a temporary storage area for the current instruction before it is being executed.
- Instruction fetched from memory is loaded into IR
- Decoder decodes the instruction & establishes the sequence of events to follow
- Not programmable & cannot accessed through any instruction
- Decoded instruction will be then moved to the next stage

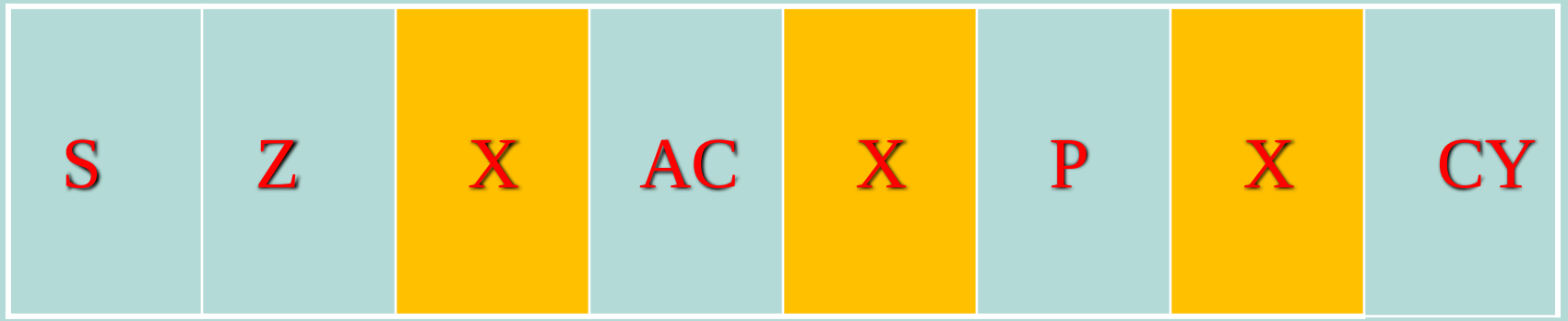


ACCUMULATOR

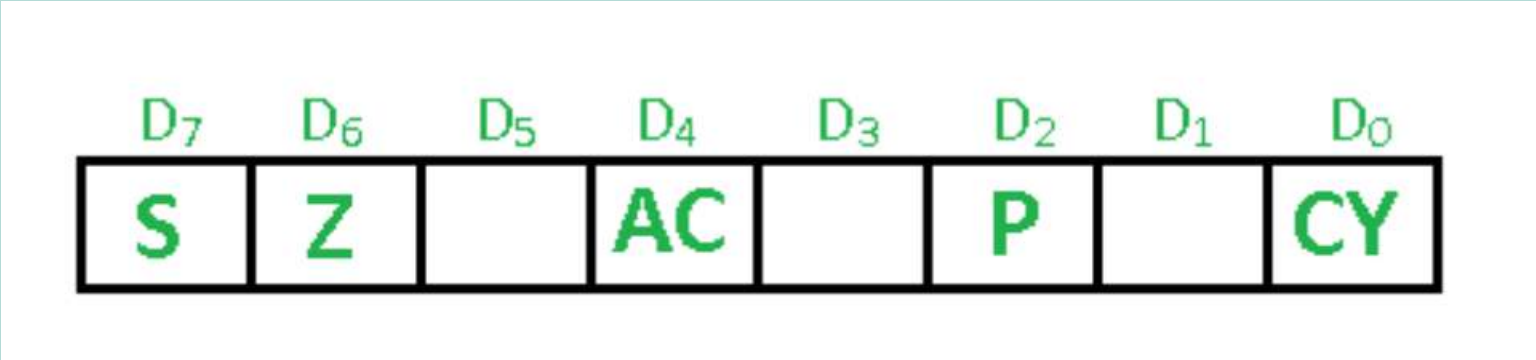
- Most important component of the entire Mu-P. 8 bit register.
- It is inevitable in all the arithmetic and logical operations since the result is going to be stored in this register.
- It is represented by the character A.
- When there is a new 8 bit data entering, the previously stored data will get automatically over written.
- Any operation that is happening will happen through Accumulator register only.
- Without Accumulator, Nothing would happen



Program Status Word Register (PSW) - Flag Register



Flag Register



S – Sign Flag, if S = 0 then the number is positive if S= 1 then the number is negative

Z – Zero Flag - if the ALU operation result becomes 0 then Z set to 1, if Z=0 then the operation will continue

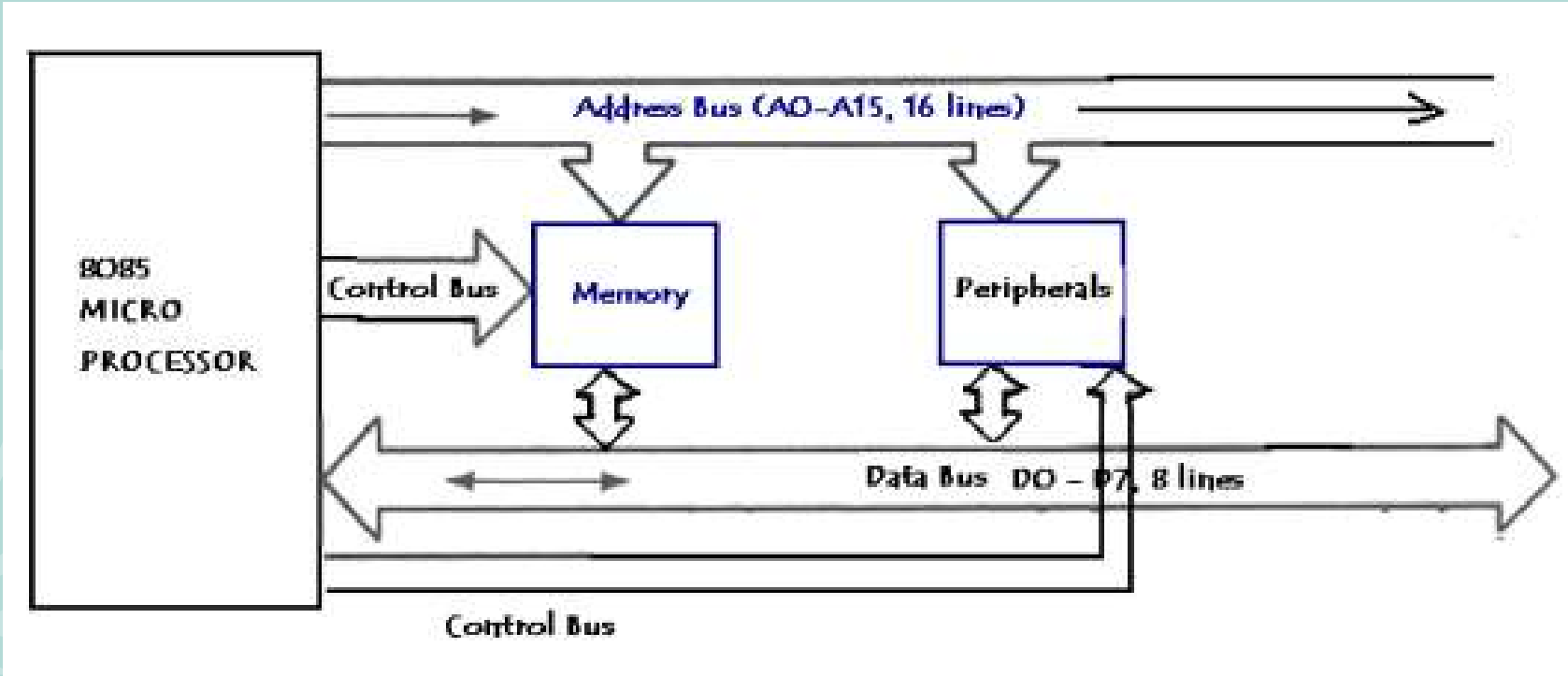
AC – Auxiliary Carry Flag – to detect if there was a carry from 3rd bit to 4th bit during addition or subtraction – if AC = 1 carry is present else AC = 0

P – Parity Flag – if P = 1 then the accumulator is having even number of One's (even parity) of else P= 0 (odd parity)

C – Carry Flag – If C = 0 then no carry else C=1 detect a carry after the addition or subtraction process

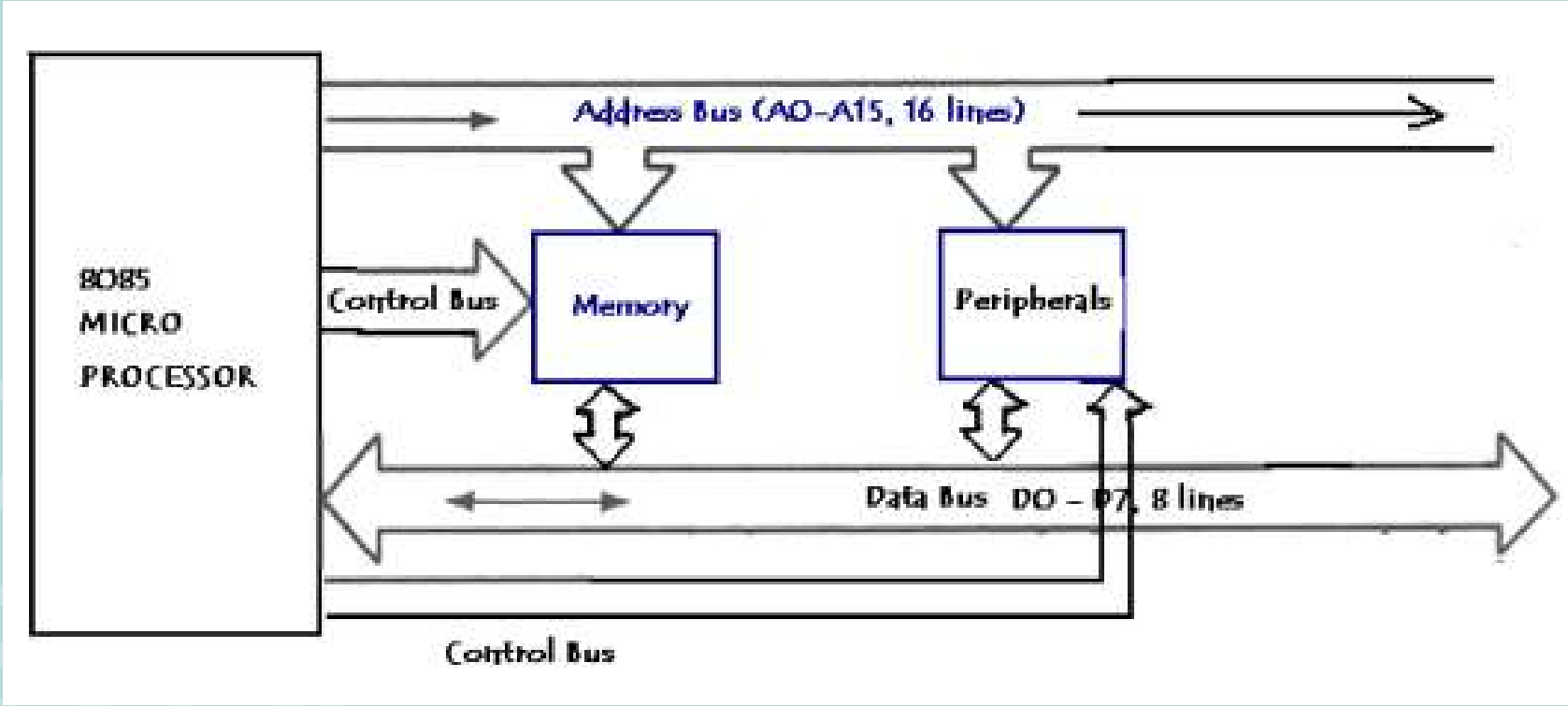
BUS STRUCTURE IN 8085

- What a bus is?
- A bus is as simple as the bus that people travel with. What is it meant for?
- It takes people and freight from a place to another.
- Same is the case here with microprocessor.
- Microprocessor often needs to send lot of control signals, data etc. to the peripherals and devices connected to it.
- All those are carried via the bus. In short it is the medium.



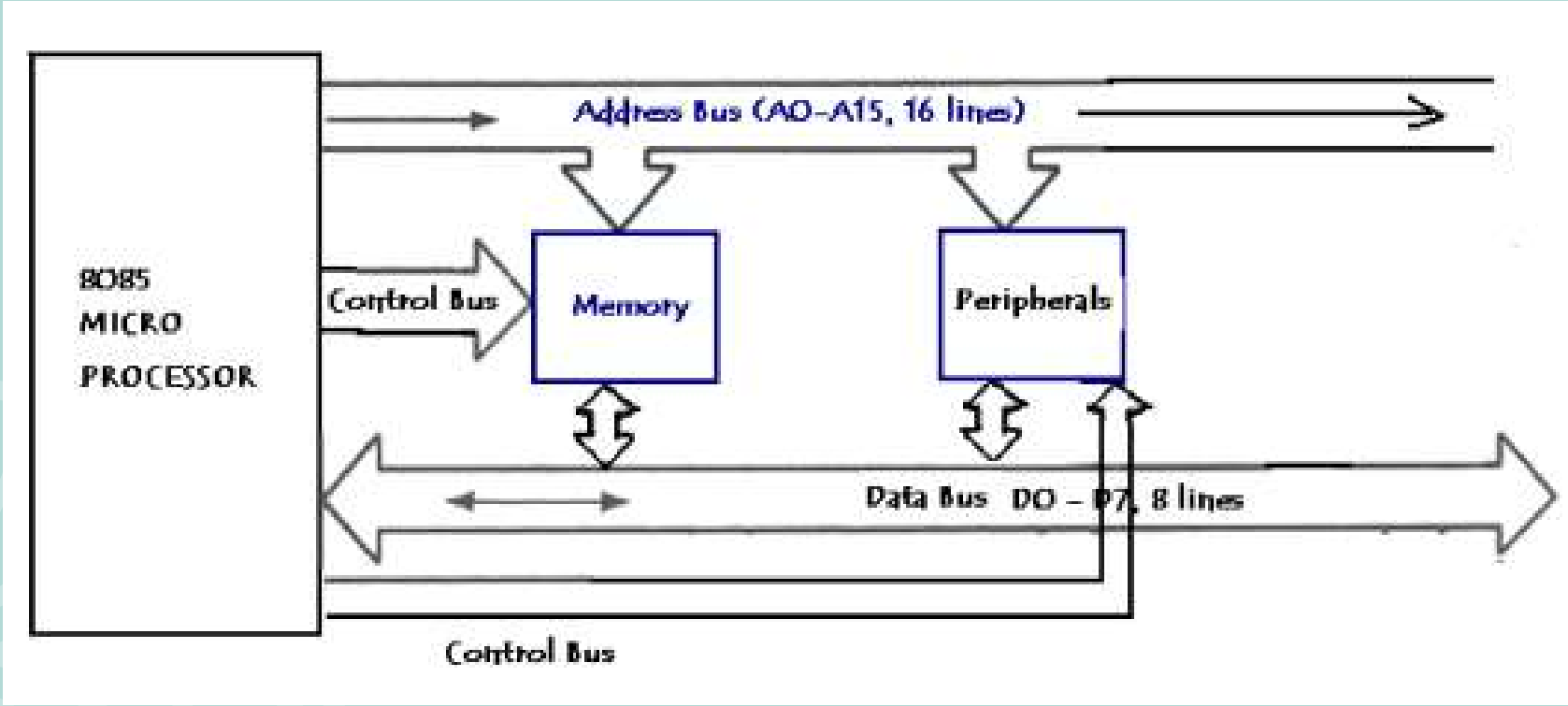
Address Bus

- Address bus
- As the name indicates it is used to carry the address.
- 8085 has 16 address lines which means it can have 2^16 = 65536bytes memory locations.
- The address bus will be mainly used to recognize a memory location or a connected peripheral.
- Postman basically delivers the letters using address. Likewise it is mandatory to have the identity for the memory locations and it is referred to be as an address.
- Address bus is always unidirectional. The communication happens from microprocessor to the peripherals



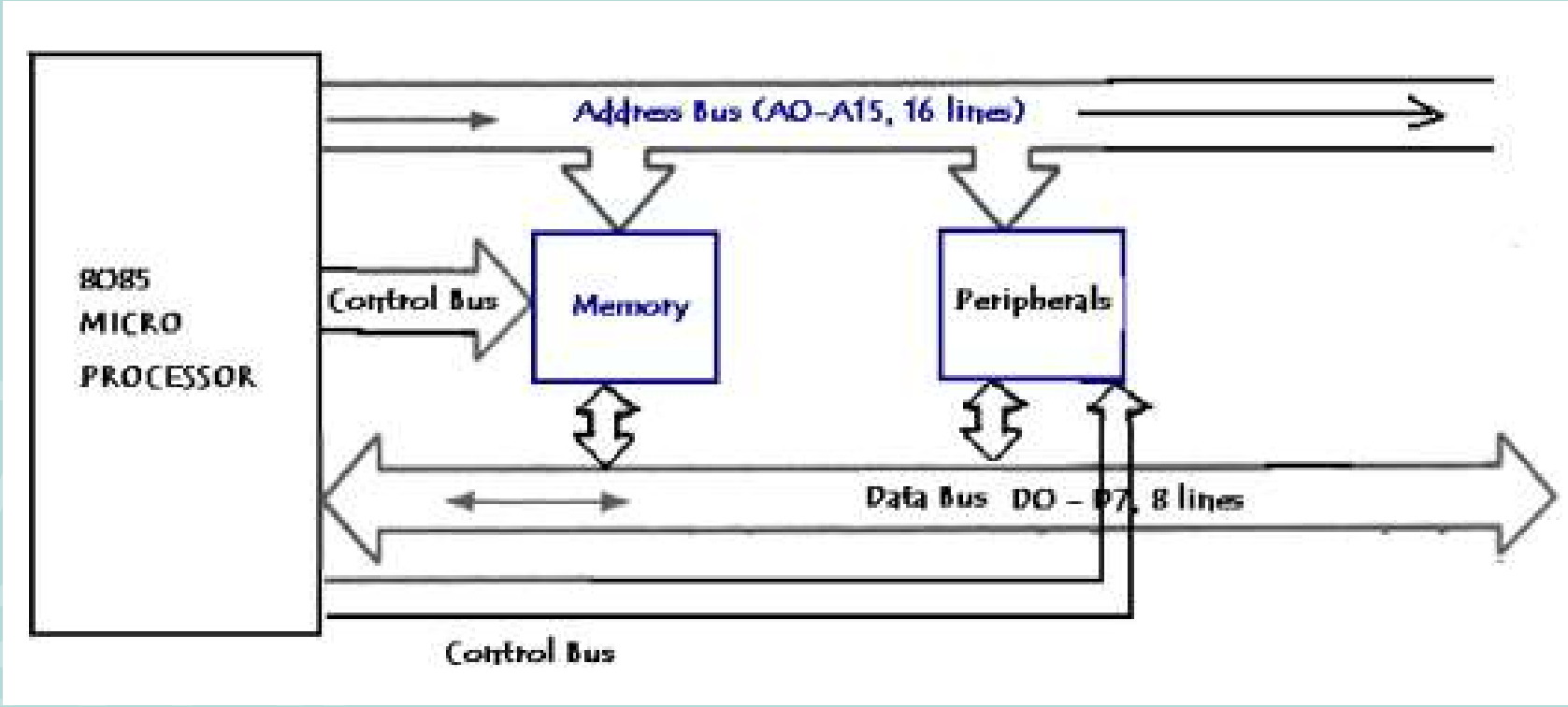
Data Bus

- Data bus
- This is bidirectional wires which carries data from to Microprocessor.
- Any information that gets in or goes out of Microprocessor is through the data bus only.
- There are 8 data lines from D0 to D7 as and this is the reason why we call it an 8 bit processor.
- Data bus is used to carry the instructions, results of the operations etc. to the peripherals and memory unit.
- When we quote that it is an 8 bit processor, it implies that the large data chunk has to be broken into smaller ones of up to 255 (o to 255, 28)



Control Bus

- Control bus
- All the controlling actions are carried out through these lines.
- It can be unidirectional or bidirectional.
- The control signals will intimate the microprocessor on where to read or where to write.
- Many control signals are available with 8085.
- All of them are discussed in detail when the pin description of 8085 is handled. Reader has to wait till then.



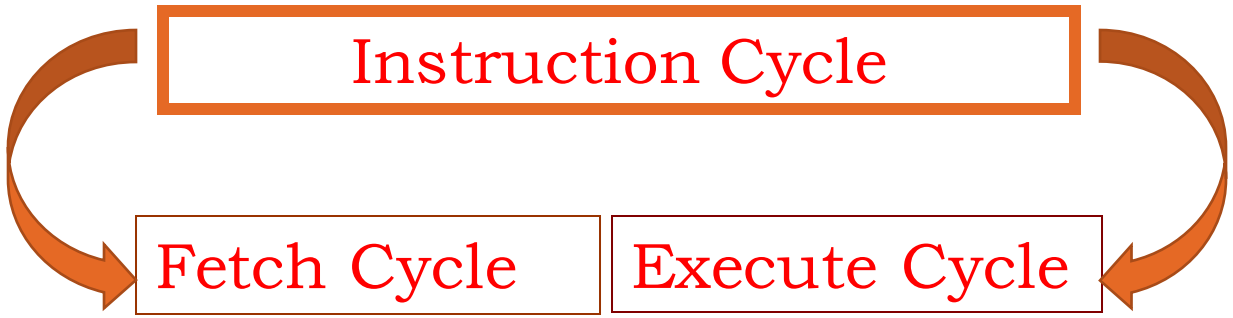
Processor Cycles

Instruction Cycle:

- The sequence of operations that a processor has to carry out while executing the instruction is called instruction cycle.

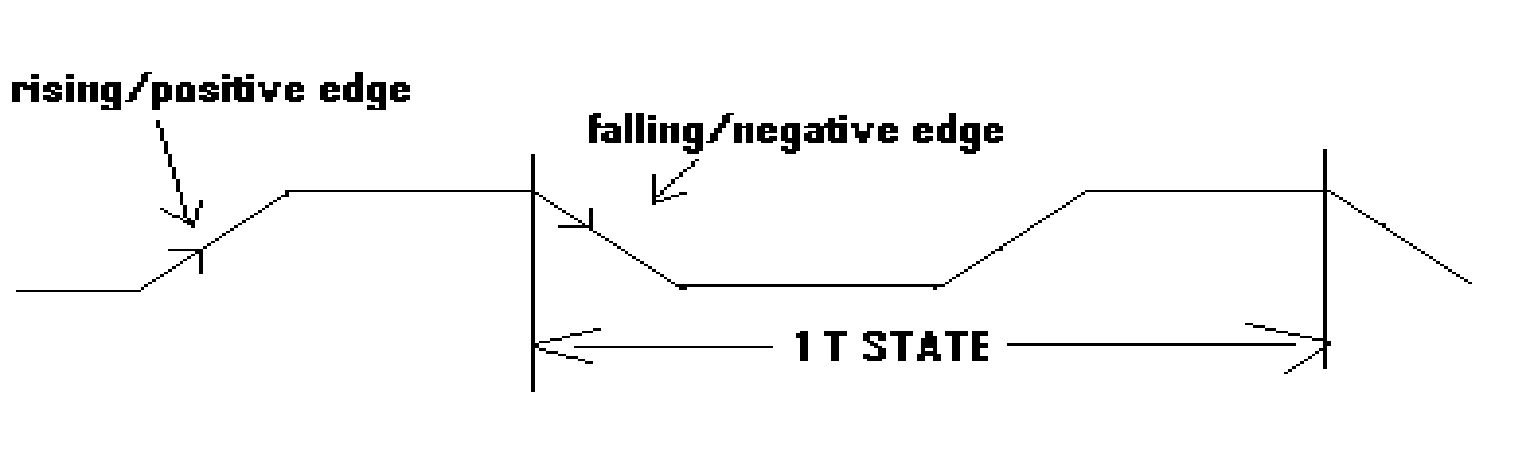
Machine Cycle:

- Each instruction cycle of a processor consist of a number of basic operations called machine cycles or processor cycles.



T States

- The time required by the processor to complete an operation is called machine cycle and is expressed in T - States.
- One T – state is equal to the time period of the internal clock signal of the processor.
- The T- state starts at the falling edge of a clock pulse and to the falling edge of the next clock pulse.



Thank You

How UP executes an instruction?

ASSEMBLY INSTRUCTION:

```
LDA 4200
MVI B,78H
ADD B
STA 4500H
HLT
```

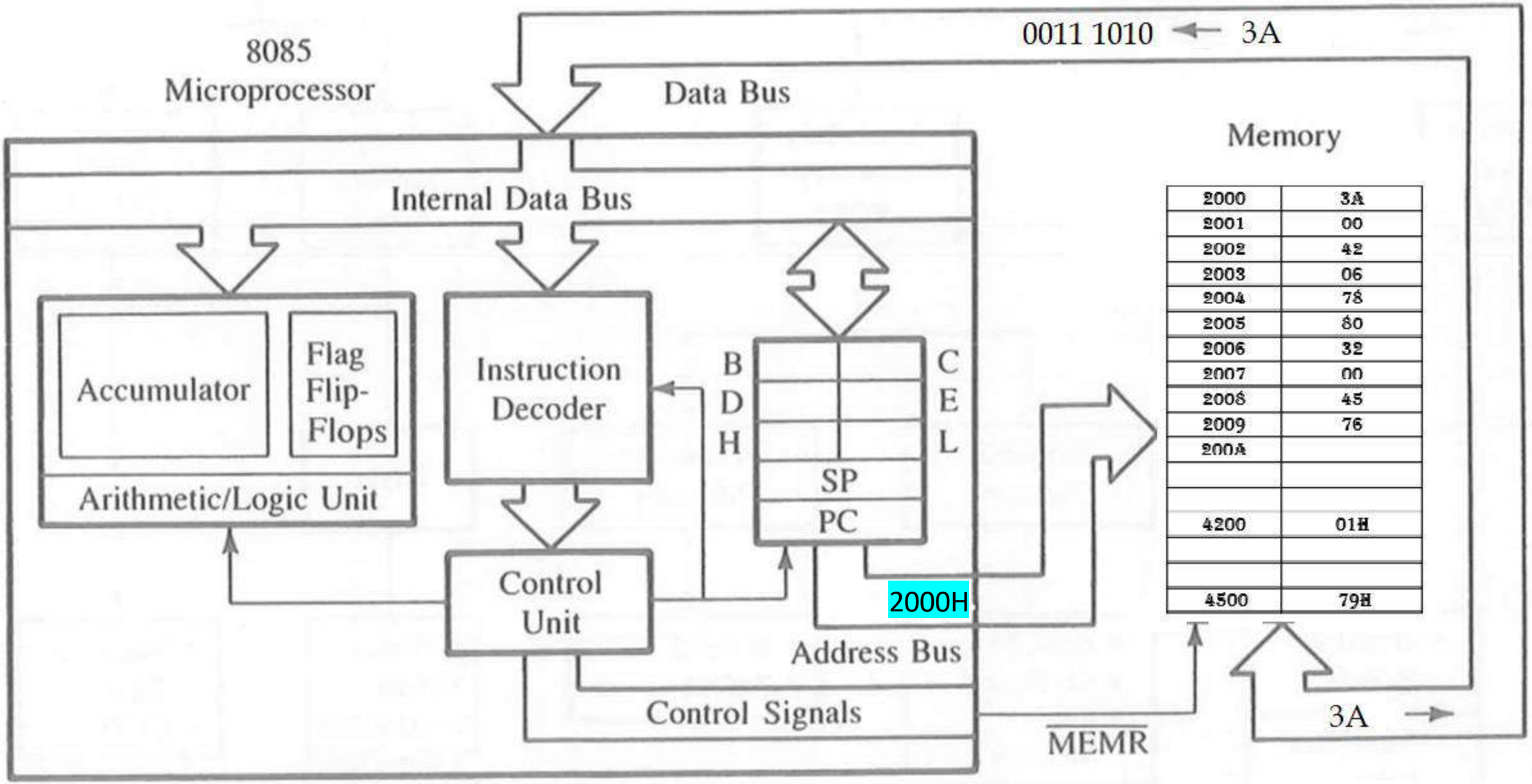
MACHINE CODE:

2000	3A	LDA 4200
2001	00	
2002	42	
2003	06	MVI B,78H
2004	78	
2005	80	ADD B
2006	32	STA 4500H
2007	00	
2008	45	
2009	76	HLT
200A		

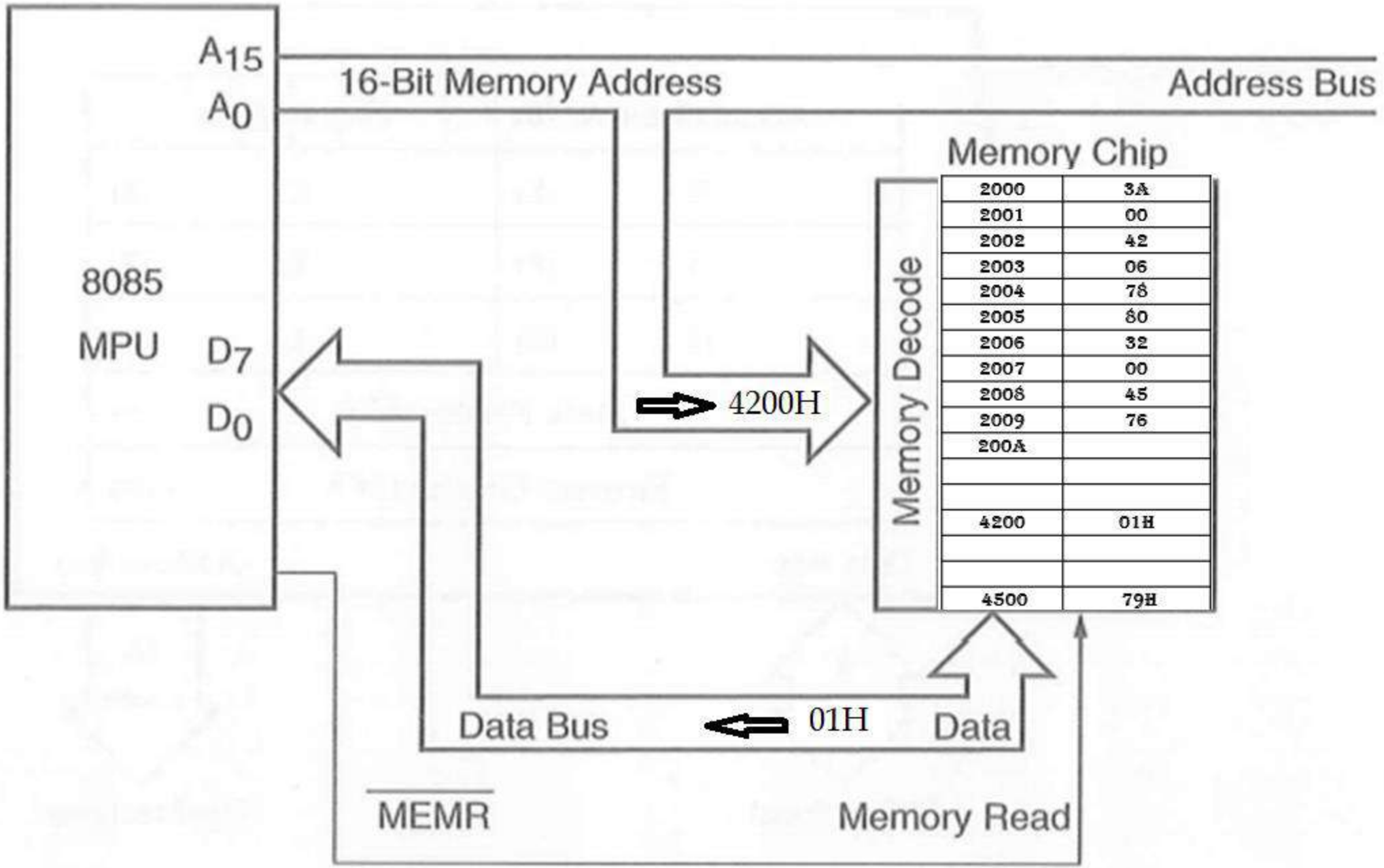
MEMORY

2000	3A
2001	00
2002	42
2003	06
2004	78
2005	80
2006	32
2007	00
2008	45
2009	76
200A	
4200	01H
4500	79H

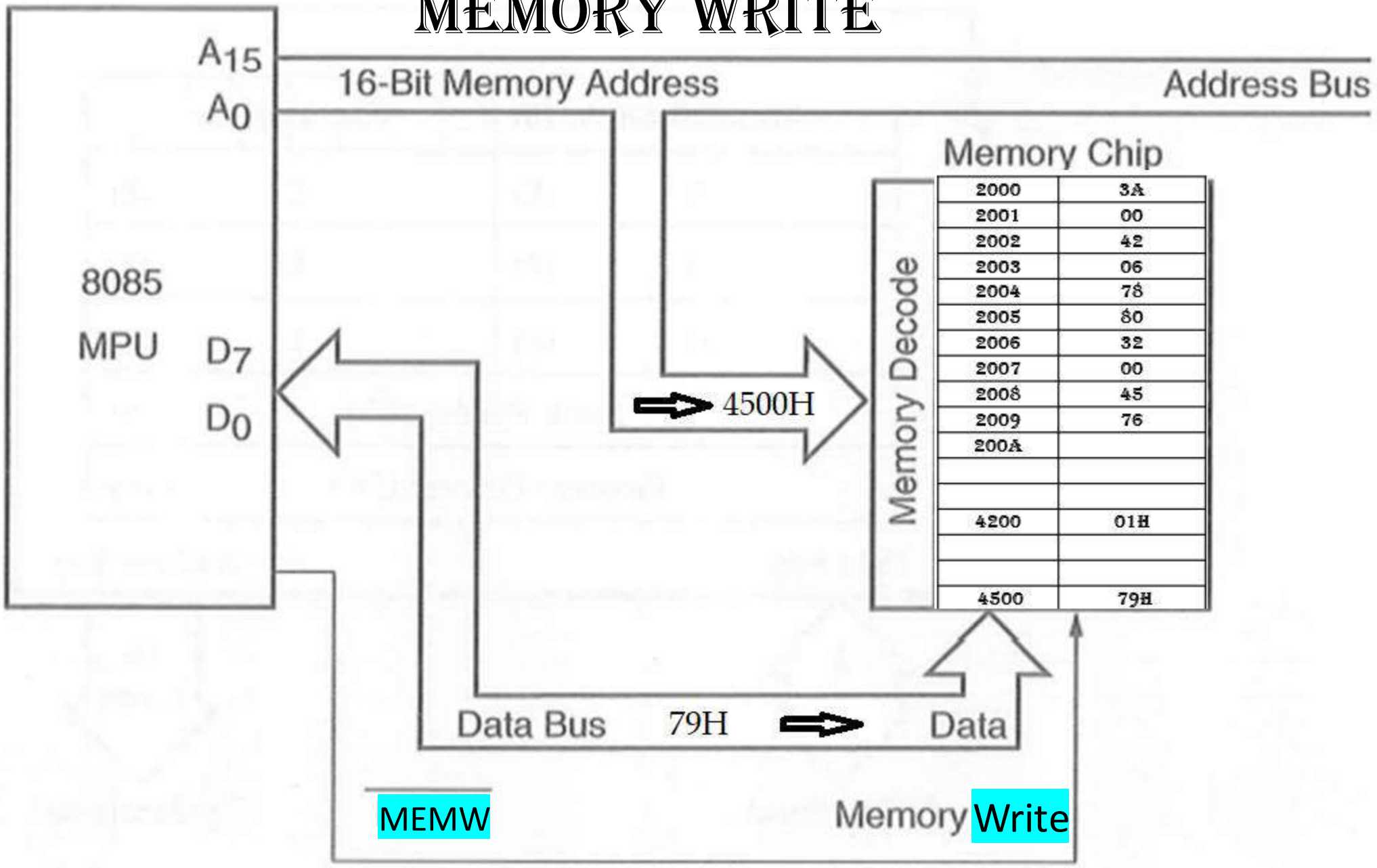
OPCODE FETCH



MEMORY READ


A15
A0
16-Bit Memory Address
Address Bus
Memory Chip
8085
MPU
D7
D0
4200H
Memory Decode
2000 3A
2001 00
2002 42
2003 06
2004 78
2005 80
2006 32
2007 00
2008 45
2009 76
200A
4200 01H
4500 79H
Data Bus
01H
Data
MEMR
Memory Read

MEMORY WRITE



8085 Instruction set & addressing mode

MICROPROCESSORS HISTORY

TABLE 1.1
Intel Microprocessors: Historical Perspective

Processor	Year of Introduction	Number of Transistors	Initial Clock Speed	Address Bus	Data Bus	Addressable Memory
4004	1971	2,300	108 kHz	10-bit	4-bit	640 bytes
8008	1972	3,500	200 kHz	14-bit	8-bit	16 K
8080	1974	6,000	2 MHz	16-bit	8-bit	64 K
8085	1976	6,500	5 MHz	16-bit	8-bit	64 K
8086	1978	29,000	5 MHz	20-bit	16-bit	1 M
8088	1979	29,000	5 MHz	20-bit	8-bit*	1 M
80286	1982	134,000	8 MHz	24-bit	16-bit	16 M
80386	1985	275,000	16 MHz	32-bit	32-bit	4 G
80486	1989	1.2 M	25 MHz	32-bit	32-bit	4 G
Pentium	1993	3.1 M	60 MHz	32-bit	32/64-bit	4 G
Pentium Pro	1995	5.5 M	150 MHz	36-bit	32/64-bit	64 G
Pentium II	1997	8.8 M	233 MHz	36-bit	64-bit	64 G
Pentium III	1999	9.5 M	650 MHz	36-bit	64-bit	64 G
Pentium 4	2000	42 M	1.4 GHz	36-bit	64-bit	64 G

*External 8-bit and internal 16-bit data bus

POINTS TO REMEMBER . . .

- Microprocessor uses buses for connecting different components & peripherals.
- There are three types of buses being used. They are 1. Data bus 2. Address bus and 3. Control bus.
- 8085 has 8 bit data lines and so why it is referred to be as a microprocessor.
- 8085 has 16 address lines.
- ALU (Arithmetical and Logical unit) is one of the most important components of a microprocessor which performs all arithmetical and logical operations.
- There are 6 general purpose registers:- **B,C,D,E,H & L**
- Accumulator, Program counter, Stack pointer are all referred to be as special function registers. They are all meant for a specific purpose.

CONTD.,

- Stack pointer is a register which points to the temporary storage area within the available memory.
- Program counter is a 16 bit register that will hold the address of the next instruction to be executed.
- 8085 has flag register which helps in getting the status of the arithmetic operations.
- The operating frequency of 8085 is 3 MHz.
- Interrupt and Polling are the options available for handling multiple processes.
- Interrupt is the best option comparing polling.
- TRAP carries a very high priority and it cannot be stopped by any command or any other mask. It is of the highest priority among all the other interrupts.

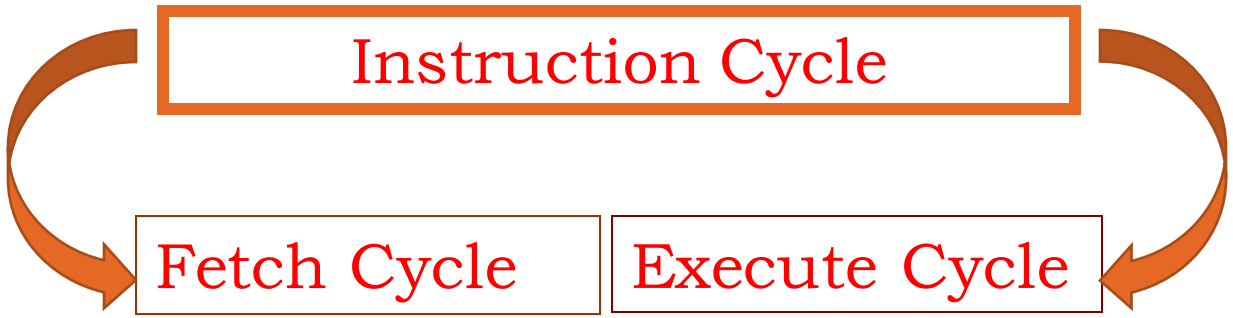
Processor Cycles

Instruction Cycle:

- The sequence of operations that a processor has to carry out while executing the instruction is called instruction cycle.

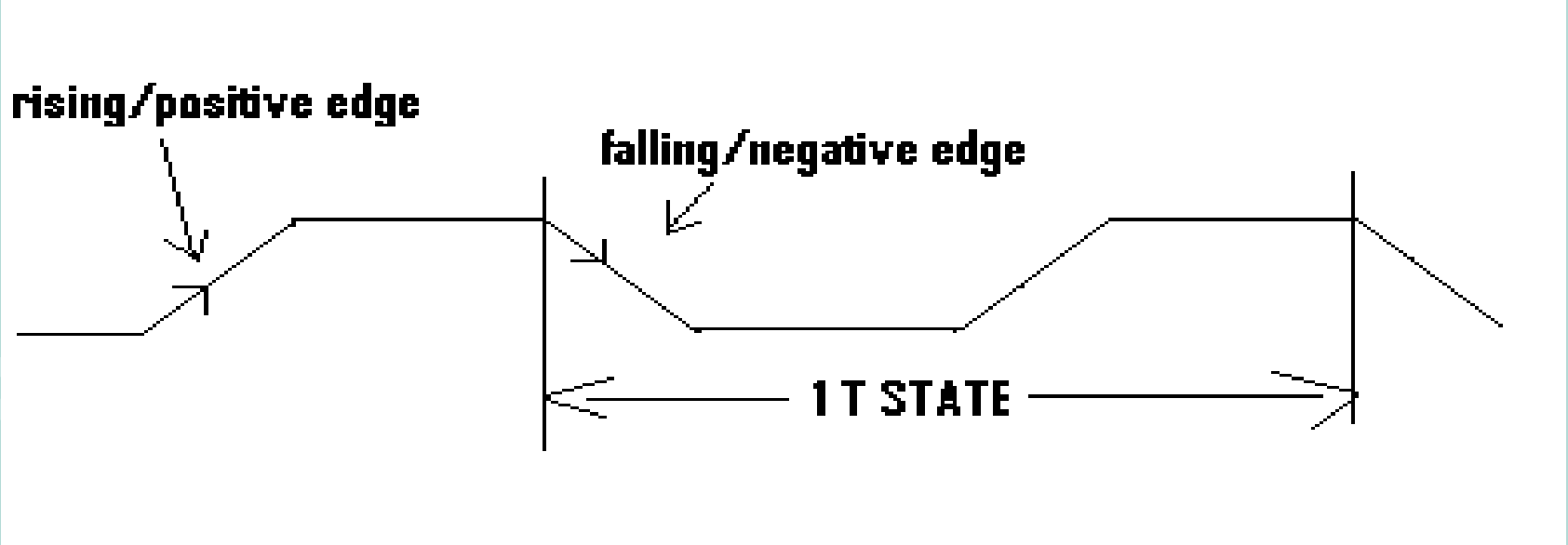
Machine Cycle:

- Each instruction cycle of a processor consist of a number of basic operations called machine cycles or processor cycles.



T States

- The time taken by the processor to execute a machine cycle is expressed in T - States.
- One T – state is equal to the time period of the internal clock signal of the processor.
- The T- state starts at the falling edge of a clock.



8085 – HOW AN INSTRUCTION WILL LOOK LIKE?

- **What is an instruction?**
  - It is a command that can make the microprocessor perform a specific function.
  - All instructions are collectively grouped as instruction set.
  - An Instruction is basically a combination of two parts one is **operation code** and the next is **operands**.

❑Operation code is also called as opcode.

- ❑ The **opcode** will be used to identify what kind of operation needs to be performed.
- ❑ **Operands** can be one or two based on the instruction.
- ❑ **Operands** will be used to spot the source and destination. An operand can be any of the following:

1. Registers (like Accumulator, B register and so on)
2. Immediate value (Say 10H, 010101B)
3. An address, i.e. memory location. (Say 30H)
4. Any of the supported input or output ports.

ADDRESSING MODES

8085 supports five types of addressing:

The addressing modes are

1. **Immediate Addressing**
2. **Direct Addressing**
3. **Register Addressing**
4. **Register Indirect Addressing**
5. **Implicit Addressing**

Mode	Description	Example
Immediate addressing mode	Simplest mode, where the instruction itself will have the information to be moved or to be processed. The data to be processed can be moved to any register or memory location.	• **MVI A,05H** – 05H is the data to be moved to Accumulator with MVI instruction which is abbreviation of Move Immediate. Similarly immediate addressing mode can be used in arithmetic operations as well. Simple example would be: • ADI 01H, where 1 is added to the content already available in the accumulator.
Direct addressing mode	In this mode one of the operands will be the address itself where the information is available	• **LDA 2000H** – Where, 2000H is the address of the source and LDA is the instruction which is expanded as load accumulator. The contents available in the memory location 2000H will be stored in accumulator once this instruction is executed. • **STA 2000H** – This is yet another instruction which falls in the category of direct addressing more. This instruction prompts the microprocessor to store the content of the accumulator into the specified address location 2000H.

Indirect addressing mode	In the previous mode, address was specified right away, but here the place where the address is available will be specified.	**LDAX D** - Load the accumulator with the contents of the memory location whose address is stored in the register pair DE
Register addressing mode	Here the instruction will have the names of the register in which the data is available.	**MOV Rd, Rs** – **Rs** is the source register and Rd is the destination register. Data will be moved from source to destination with this instruction. One simple instance would be MOV C, D where the content of D is moved to the register C. Meanwhile the same addressing mode is applicable for arithmetic operations. **ADD C** will add the content of register C to accumulator and will store the result again in accumulator.
Implicit addressing mode	The instruction itself will have the data to be processed and there will not be any other operands.	CMC is one of such instructions which fall in this category of implicit addressing mode. CMC is the abbreviation of Complement Carry.

Addressing modes supported in 8085 microprocessor

8085 – INSTRUCTION SET.

246 instructions are available in 8085.

The instruction set can be classified as follows:

1. Data transfer instructions
2. Arithmetic instructions
3. Logical instructions
4. Branching instructions and
5. Control instructions.

1. Data transfer instructions – it is used for moving data from source to destination. The content of source will not be modified after the data transfer is complete. It will remain the same.

MOV Rd, Rs **MOV Rd, M** **MOV M, Rs**	This instruction copies the contents of the source register into the destination register and as mentioned already, it would not alter the content of the destination register. There are three instructions supported for this category: 1. From a register to register move. 2. From memory location to a register move. 3. From a register to memory. Point to note is the memory address will be hold or pointed by the HL register pair. Example: **MOV B,C** **MOV B,M** **MOV M,C** And another thing to note is no flags will be affected due to these instructions.

MVI Rd, Data **MVI M, Data** ★	This is an immediate data operation where the data is moved to the destination register mentioned. Example: **MVI Rd, 76H** Also the immediate data can be moved to a memory location specified by HL registers. Example: **MVI M, 76H** No flags will be affected due to the usage of these instructions.
LXI <Register_Pair> , <16_bit_data> ★	Immediate addressing mode can be realized through this instruction. The 16 bit data is moved to the register pair mentioned as one of the operands. Example: **LXI H, 2545H**

POP Register_Pair	It is exactly opposite to PUSH. It will extract the top two memory spots from the stack and will put it on to the register pair mentioned. Example : **POP B**
OUT <8 bit address>	If the programmer wishes to export the data in the accumulator to be moved to the external world, it can be done only through ports. This can be accomplished through OUT instruction which will output the content of the accumulator to the mentioned port address. Example: **OUT 54H**
IN <8 bit address>	This is simply opposite to the previous command. Here the data can be input from the port to the accumulator. Example: **IN 40H**

LDA <16_bit_address> 	Expanded as load accumulator and it is very straight here in understanding. The memory location pointed by a 16 bit address will be copied to the accumulator. The source will remain unaltered. The flags will remain unaltered. Example: **LDA 2004H**
LDAX <Register_Pair>	The content pointed out by the memory address which is stored in the register pair will be moved to the accumulator. This is the best example for the indirect addressing mode. No flags will be altered because of this. And the source content will remain unaltered. Example: **LDAX H**

STAX <Register_Pair> 	This will store the contents of the accumulator to the address pointed by the register pair and this is an awesome example for the reader to understand the indirect addressing mode. There will not be any alteration in the source i.e. the accumulator. Example: **STAX H**
LHLD <16_bit_address> 	Instruction is expanded as Load H and L. This is one of the classical examples for direct addressing mode. The contents of the memory location pointed by 16 bit address will copied into the L and the next memory location's content will be copied to H register. The content of the source will not be altered and no flags will be affected as an impact of this instruction. Example: **LHLD 2000H**
SHLD <16_bit_address> 	This is exactly in reverse to the previous instruction. Here the contents of the H and L registers will be stored in the 2 consecutive memory locations as pointed by the 16_bit_address. . Source remains unaltered and the flags are so as well. Example: **SHLD 2000H**

SPHL	Used to copy the content of H and L register pair into the stack pointer. Contents of the source remain unaltered. Example: **SPHL**
XCHG 	Expanded as exchange. This instruction will help in exchanging the contents of HL with DE register pair. No flags will be affected. Example: **XCHG**
XTHL	This is also an instruction related to exchanging the data. But, this time the content of HL register pair will be exchanged with the contents of top of the stack. Example: **XTHL**

2. ARITHMETIC INSTRUCTIONS

ADD R **ADD M** 	The contents from the register or memory locations will be added to the content of the accumulator and the result will be stored back in the accumulator. Also the flags will be affected and status will be reflected in the flags. Add with carry options are also available and can be used for cases where carry operations are required. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Example : **ADD B** **ADD M**
ADI 	Expanded as Add Immediate. It will add the immediate data mentioned as operand with the content of the accumulator. All the flags could get affected and might be modified based on the execution results. Example: **ADI 54H**
SUB R **SUB M** 	The contents from the register or memory locations will be subtracted from the contents of the accumulator, and the result will be stored back in the accumulator. Also the flags will be affected and status will be reflected in the flags. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Example : **SUB B, SUB M**

SUI	This is subtract immediate. It subtracts the immediate data from the contents of the accumulator and the result will be stored back in the accumulator. The flags will be modified based on the result of the operation. Example: **SUI**
INR R **INR M** 	Used for incrementing the register's content or address's content by 1. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. No flags will be affected in this case. Example : **INR B, INR M**

DCR M **DCR R** 	Used for decrementing the content of the address or register by 1. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. No flags will be affected in this case. Example : **DCR B, DCR M**
ACI 	Expanded as add immediate to the accumulator with carry. The 8 bit data and the Carry flag will be added to the contents of the accumulator. Then the result will be stored back in the accumulator. The flags will be changed based on the result of the operation. Example: **ACI 51H**
SBI 	Expanded as subtract immediate from the accumulator with borrow. The 8 bit data and the borrow flag will be subtracted from the contents of accumulator and the result is stored back in the accumulator. The flags will be changed based on the result of the operation. Example: **SBI 51H**
DAD **Register Pair**	This is used for 16 bit addition. Contents of the register pair specified will be added to the content of HL register pair and result will be stored in the HL register. Based on the result the CY flag might be altered. Example: **DAD B**

INX **Register Pair** 	This is used for incrementing the content of register pair by one. The result will be retained in the same place. Example: **INX H**
DCX **Register Pair** 	This is used for decrementing the content of register pair by one. The result will be retained in the same place. Example: **DCX H**
DAA	Expanded as Decimal Adjust Accumulator. It is used in the conversion of binary to BCD. Flags will be modified and that change will reflect the result. Example: **DAA**

ADC R **ADC M** 	Add register / memory to the accumulator with carry is abbreviated as ADC. Here the contents of register or memory and the Carry flag will be added to the contents of the accumulator. Result will be stored in the accumulator. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Flags will be modified based on the result of the operation. Example: **ADC R** **ADC M**
SBB R **SBB M** 	Same as previous instruction, but here subtraction is done. Here, the contents of the register or memory location and the borrow flag are subtracted from the contents of the accumulator and result will e placed in the accumulator. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. All flags are modified to reflect the result of the subtraction. Example: **SBB R** **SBB M**

3. LOGICAL INSTRUCTIONS

- Points to be noted before learning/reading on logical instructions are summarized as follows:
- One of the operands must be stored in the accumulator without which it would not be possible to use the logical instructions.
- Any addressing mode can be used with the logical instructions. There is no restriction in this regard.
- All the instructions that perform logical operations are categorized under LOGICAL instruction set.

CMP R **CMP M**	Expanded as Compare. The contents of the register or the memory location (R or M) will be compared with the contents of the accumulator. Both the contents of register and accumulator will be retained after the operation. Example: **CMP C** The result of the operation compare can be known through the flags. Following are the cases: ◆ if (A) < (register/memory): carry flag is set ◆ if (A) = (register/memory): zero flag is set ◆ if (A) > (register/memory): carry and zero flags are reset
CPI – 8 bit	This instruction is expanded as Compare Immediate. It compares the immediate operand (8 bit data) with the content of the accumulator. The contents of accumulator and 8 bit data which is being compared will remain unchanged. Example: **CPI 90H** The result of the operation compare can be known through the flags. Following are the cases: ◆ if (A) < data: carry flag is set ◆ if (A) = data: zero flag is set ◆ if (A) > data: carry and zero flags are reset

ANA R **ANA M**	This will logically AND the content of the register or the memory location / register with the content of accumulator and result will be stored in accumulator. Flags will be affected based on the result. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Example: **ANA B** The flags S, Z, P in the flag registers will be modified based on the result of the operation. CY is reset. AC is set.
ANI	Expanded as AND Immediate. It will logically AND the contents with the contents in the accumulator. Result will be stored in the accumulator. Example: **ANI 01H** Flags S,Z and P will be modified based on the result of the operation. Also CY flag is reset. AC flag is set.
CMA	CMA is expanded as Complement Accumulator. It will complement the contents of the accumulator and result will be stored back in accumulator. No flags will be affected because of this instruction. Example: **CMA**
CMC	CMC is expanded as Complement Carry flag. It will not affect any other flags. Example: **CMC**

STC	Set Carry is abbreviated as STC. It sets the Carry flag and it will not have impact on any other flags. Example: **STC**
XRA A **XRA M**	This will Exclusively OR (EXOR) the content of the register or the memory location / register with the content of accumulator and result will be stored in accumulator. Flags will be affected based on the result. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Flags S,Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **XRA B** **XRA M**

ORA A **ORA M**	This will logically OR the content of the register or the memory location / register with the content of accumulator and result will be stored in accumulator. Flags will be affected based on the result. If the operand is an address instead of a register, then it will be mentioned with the help of registers HL. Flags S, Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **ORA B** **ORA M**
XRI	XRI is expanded as XOR immediate. The contents of the accumulator will be XORed with immediate 8 bit data and the result will be placed in the accumulator. Flags S,Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **XRI 10H**
ORI	ORI is expanded as OR immediate. The contents of the accumulator will be ORed with immediate 8 bit data and the result will be placed in the accumulator. Flags S,Z and P will be modified based on the result of the operation. CY and AC flags will be reset. Example: **ORI 10H**

RAL 	Expanded as Rotate accumulator left through carry. Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: **RAL**.
RAR 	Expanded as Rotate Accumulator Right through carry. Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: **RAR**.

RLC 	Rotate Accumulator Left. Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: **RLC.**
RRC 	Expanded as Rotate Accumulator to the right. Each bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RRC

4. Branching Instructions

Used to alter the normal flow and shift the control to the different address/area.

JMP 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. This is an unconditional jump. The next set of instructions which are to be dealt are examples for conditional execution. Example: **JMP 2000H**
Jump conditionally. There are many conditional instructions and all of them are summarized below with a brief note.	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the flag register. Example: **JZ 2034H or JM 2000H**

JC	**Jump on Carry**	**CY = 1**
JNC	**Jump on no Carry**	**CY = 0**
JP	**Jump on positive**	**S = 0**
JM	**Jump on minus**	**S = 1**
JZ	**Jump on zero**	**Z = 1**
JNZ	**Jump on no zero**	**Z = 0**
JPE	**Jump on parity even**	**P = 1**
JPO	**Jump on parity odd**	**P = 0**

Table 2.5 Jump instructions

Conditional jump instructions are also available and they are listed in the table 2.6 as follows with a brief note on what each instruction does.

CB	**Call on Carry**	**CY = 1**
CNC	**Call on no Carry**	**CY = 0**
CP	**Call on positive**	**S = 0**
CM	**Call on minus**	**S = 1**
CZ	**Call on zero**	**Z = 1**
CNZ	**Call on no zero**	**Z = 0**
CPE	**Call on parity even**	**P = 1**
CPO	**Call on parity odd**	**P = 0**

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the flag register.

Example: **CP 2000H, CZ 2300H**

<table>
<tr>
<td>Unconditional return from the subroutine
RET</td>
<td>The program sequence will be transferred from the subroutine to the calling program. This will be done unconditionally.
Example: RET</td>
</tr>
<tr>
<td>Conditional return instructions are supported as well in 8085. They are listed in the table 2.7 with a brief note.
<table>
<tr><td>RC</td><td>Return on Carry</td><td>CY = 1</td></tr>
<tr><td>RNC</td><td>Return on no Carry</td><td>CY = 0</td></tr>
<tr><td>RP</td><td>Return on positive</td><td>S = 0</td></tr>
<tr><td>RM</td><td>Return on minus</td><td>S = 1</td></tr>
<tr><td>RZ</td><td>Return on zero</td><td>Z = 1</td></tr>
<tr><td>RNZ</td><td>Return on no zero</td><td>Z = 0</td></tr>
<tr><td>RPE</td><td>Return on parity even</td><td>P = 1</td></tr>
<tr><td>RPO</td><td>Return on parity odd</td><td>P = 0</td></tr>
</table>
</td>
<td>The program sequence will get transferred from the subroutine to main calling program based on the flag status as shown in the table 2.7.
Example: RC</td>
</tr>
</table>

RST (0-7)

These instructions can be used to transfer the control of the program to the one of the 8 locations mapped to RST 0 to 7 as follows as shown in table 2.8

Instruction	Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

5. Control instructions

Microprocessor is controlled by these control instructions

NOP	It is expanded as No – Operation. The name itself reveals the purpose of the command. The instruction will be fetched and then decoded. As a result of this, PC will hold the address of the next instruction to be executed. In short, this instruction will not affect or modify anything. **Point to be noted is, there is no other operands needed for this instruction.** Example: **NOP**
HLT	Expanded as Halt. The current instruction will be executed and further execution of the program will be halted. There will be need for an interrupt or reset to come out of the HLT mode. Example: **HLT**
DI and EI	Expanded as Enable Interrupts and Disable Interrupts. If interrupts have to be ignore, then DI should be executed which will make the processor to ignore any interrupt request except TRAP. Just to remember, TRAP is a non maskable interrupt. No flags will be affected. Example: **DI** To enable the interrupts, EI has to be used to re-enable the interrupts. No flags will be affected. Example: **EI**
RIM	Expanded as Read Interrupt Mask. This helps in reading the status of RST 7.5, 6.5, 5.5 and Serial data input bit.

Thank You