ASSEMBLY LANGUAGE PROGRAM BEGINNING

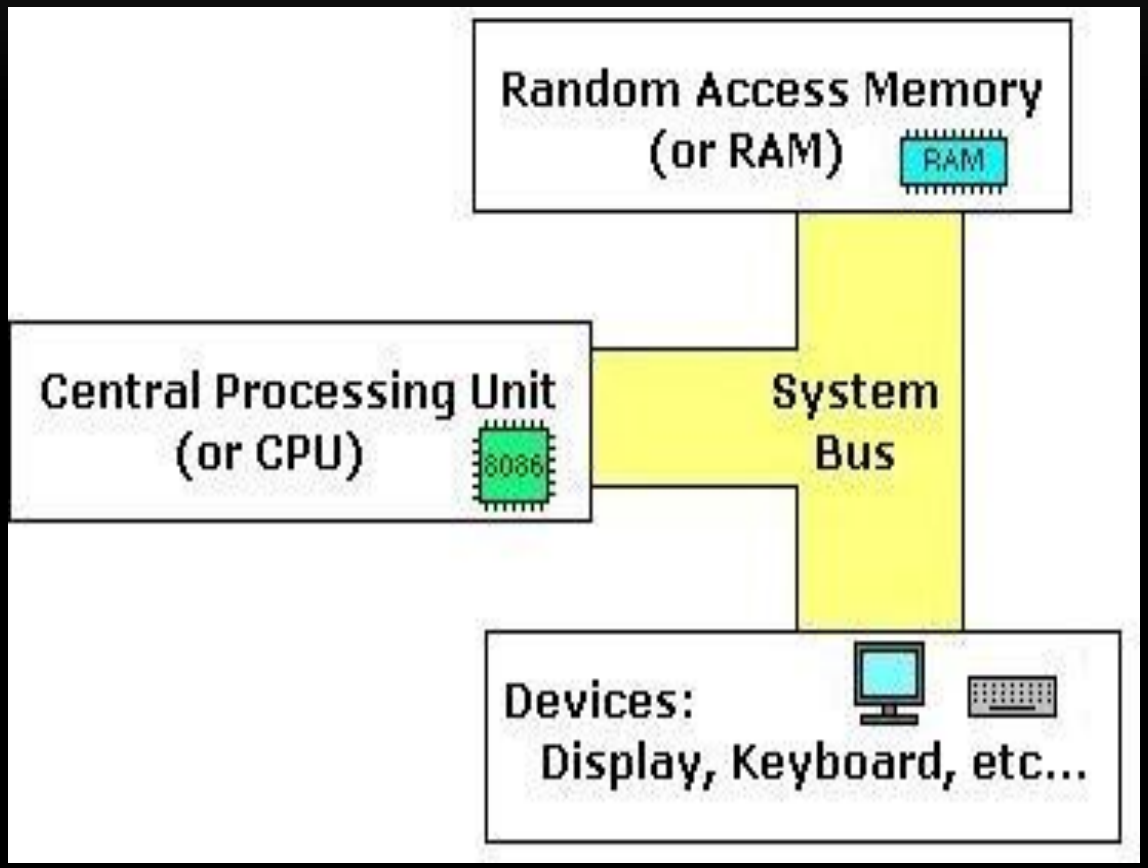
Course Teacher

Shaila Rahman

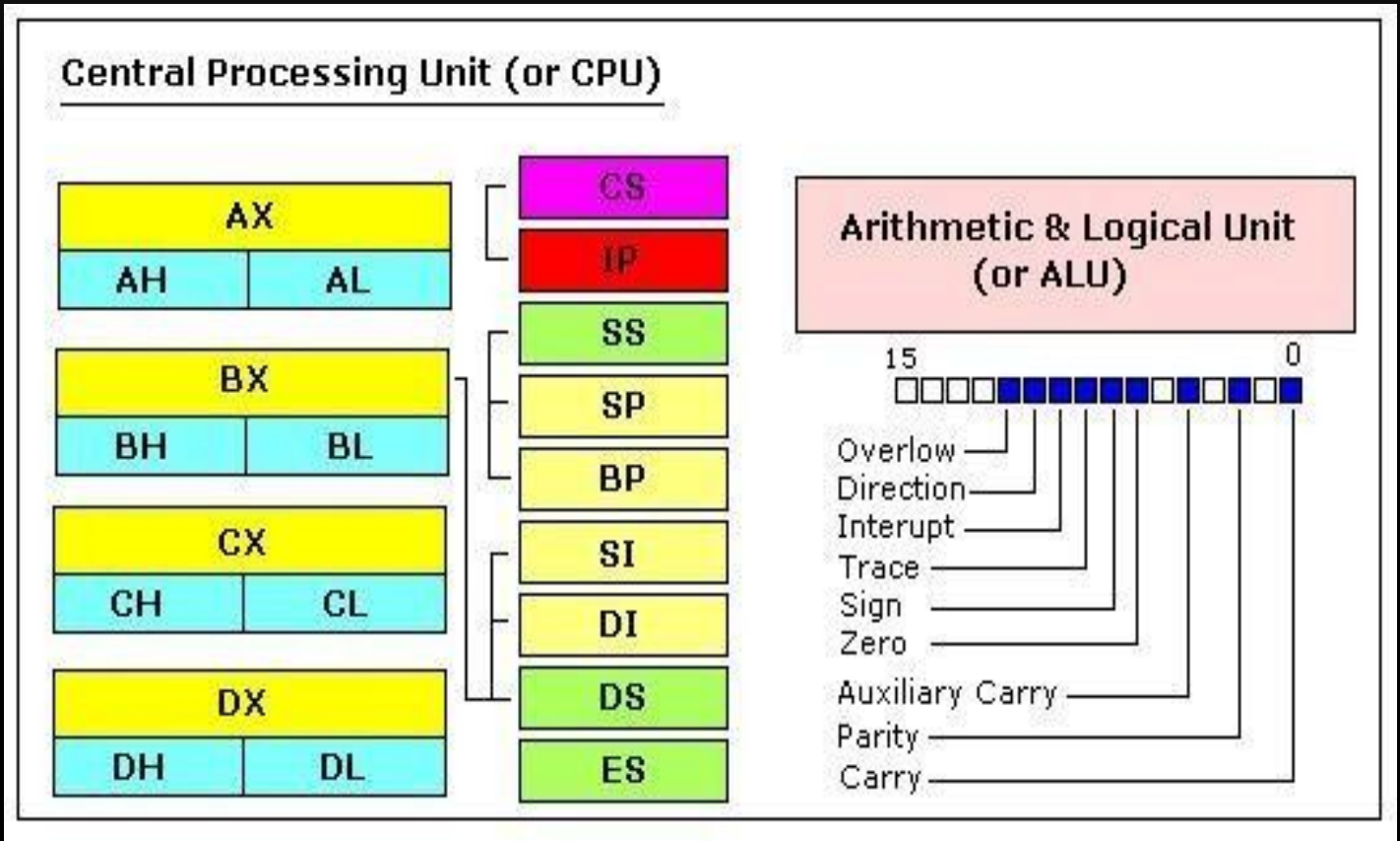
LESSON PLAN

- Assembly program structure
- First assembly program with EMU8086
- Related concepts with the first program:
  - Loading program
  - Boot process
  - Handling the stack

MAIN COMPUTING SYSTEM STRUCTURE



RECALLING MAIN CONCEPTS



DIRECTIVE MODEL

The model indicates the size of code and data segment.

- Syntax

.MODEL SMALL/ COMPACT/MEDIUM/ LARGE

SMALL	**code - one segment** **data - one segment**
MEDIUM	**code more than one segment** **data one segment**
COMPACT	**code one segment** **data more than one segment**
LARGE	**code more than one segment** **data more than one segment**
	One segment size is 64 KB

SEGMENT DIRECTIVES

8086 uses Segment: special areas defined to contain CODE, DATA and STACK

Assembly program consists of three parts, or segments according to this concept.

STACK SEGMENT

Stack segment is used to store temporary information actually the contents of registers for interrupted section of program, mainly return address.

- The stack segment begins with *directive* .STACK
- Syntax

.STACK size

Size in no. of bytes.

.STACK 100h ; allocates 100 bytes for stack segment

STACK

- The word is from data structure
- Last In, First Out (LIFO) mechanism
- STACK in OS has three main functions:
  - Contains return address
  - Data
  - Content of present registers

STACK

- PUSH
  - Decrease SP by 2 and store a value there
- POP
  - Return a value from stack and increase SP by 2

DATA SEGMENT

The data segment begins with *directive* .DATA

- Syntax

```
.DATA  ; logical definition of data segment
```

Memory variables will be declared in this section.

DATA TYPES

There are four types of data -

BYTES (8-bit) : This can be defined as DB = Define Byte
WORD (16-bit): This can be defined as DW = Define Word
DOUBLE WORD (32-bit): This can be defined as DD = Define Double Word
QUAD WORD (64-bit) : This can be defined as DQ = Define Quad Word
Ten Bytes (80-bit) : This can be defined as DT = Define Ten Bytes

Syntax :
Variable_ Name type value
8086 works with byte and word operation.
i.e. DB or DW applicable.

EXAMPLES

- B DW 3478h
- C DB ‘D’
- D DW 10h,20h,30h
- Msg DB “THIS IS A MESSAGE$”
- A EQU 12345h
- Num DB ‘ ? ’

DEFINING CONSTANTS

To assign a name to a constant, we can use the EQU (equates)

pseudo-op. The Syntax is

EQU (Equates)

Name EQU Constant

CODE SEGMENT

The code segment begins with the *directives*

- Syntax

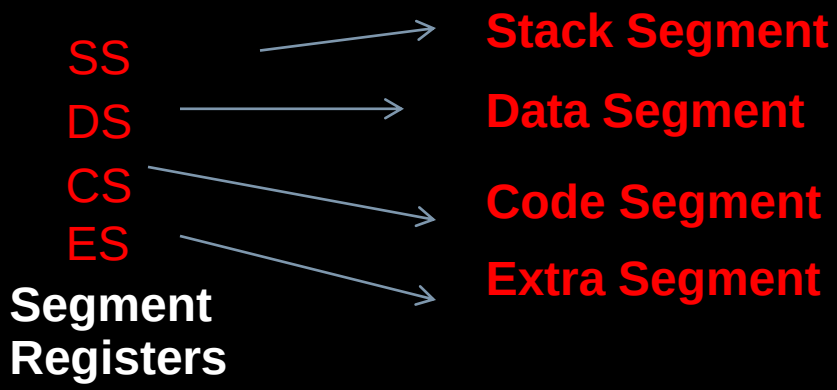
```
.CODE    ; logical position of code segment
                starts from here
```

Instructions goes after this statement .

INSTRUCTION EXECUTION AND ADDRESSING

- Executing an instruction include
  - Fetch the next instruction, put to a queue (QUEUE: FIFO vs. STACK LIFO)
  - Decode the instruction
  - Execute the instruction

SEGMENT REGISTERS



PROCEDURES

Inside a code segment, instructions are organized as procedures. The simplest procedure definition is-

- Syntax

name PROC

;body of the procedure

RET

name ENDP

where name is user given and the keyword PROC for procedure and ENDP for end of procedure.

AN EXAMPLE OF A CODE SEGMENT DEFINITION

```
.CODE
MAIN PROC
;main procedure instructions go here
MAIN ENDP
;other procedures go here. Any procedure needs RET statement but not needed in main.
```

A PROGRAM STRUCTURE

```
.MODEL SMALL        ; BOTH Code and data size= 64KB
. STACK IOOH        ; stack size 100 bytes
.DATA               ; Starting data segment
;data definitions go here
.CODE               ; starting code segment
MAIN PROC
;instructions go here
MAIN ENDP
;other procedures go here
END MAIN
```

MOV/ADD/SUB/XCHG (SWAP)

- Syntax

Opcode Destination, Source

Possible combination

- Reg-reg
- Reg-mem
- Mem-reg
- Mem-Immd
- Reg-Immd

INC/DEC/ NEG/ MUL/DIV

Syntax

- Opcode operand
- Example

INC AX

NEG AX ; 2’s complement of AX