Mansoura University
Faculty of Computers and Information
Department of Computer Science
First Semester: 2020-2021

[CS214P] Assembly Language

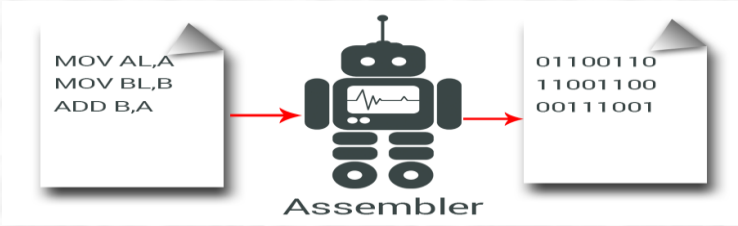
Grade: Third Year (Computer Science)

Sara El-Metwally, Ph.D.

Faculty of Computers and Information,

Mansoura University,

Egypt.



Computer Science Department
Faculty of Computers and Information
Mansoura University

Assembly Language

"Instruction Addressing and Execution"

Sara El-Metwally, Ph.D.
Faculty of Computers and Information,
Mansoura University, Egypt.

Email: sarah_almetwally4@mans.edu.eg
sara.elmetwally.2007@gmail.com

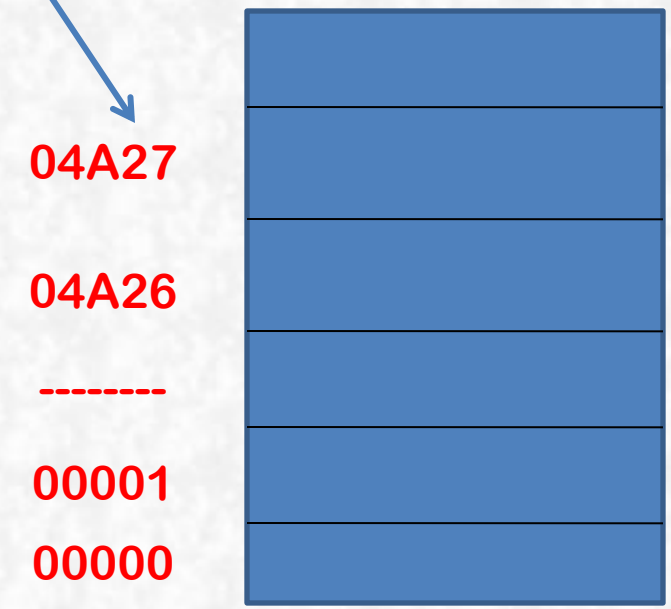
Addressing Data in Memory

- **An absolute address**: 20-bit value that directly references a specific location in memory.

- **A segment : offset address**, combines the starting address of a segment with an offset value.

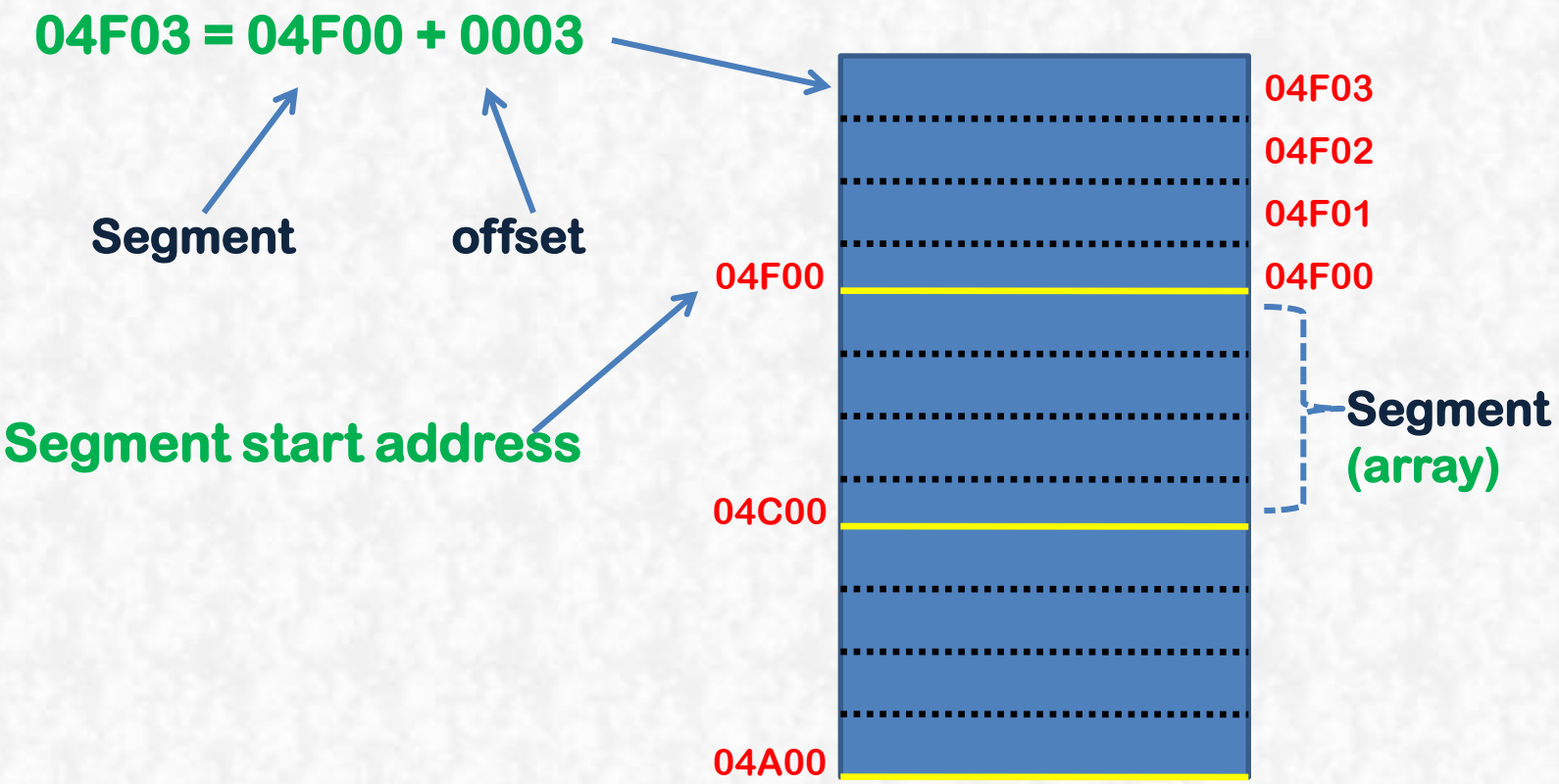
Addressing Data in Memory

An absolute address

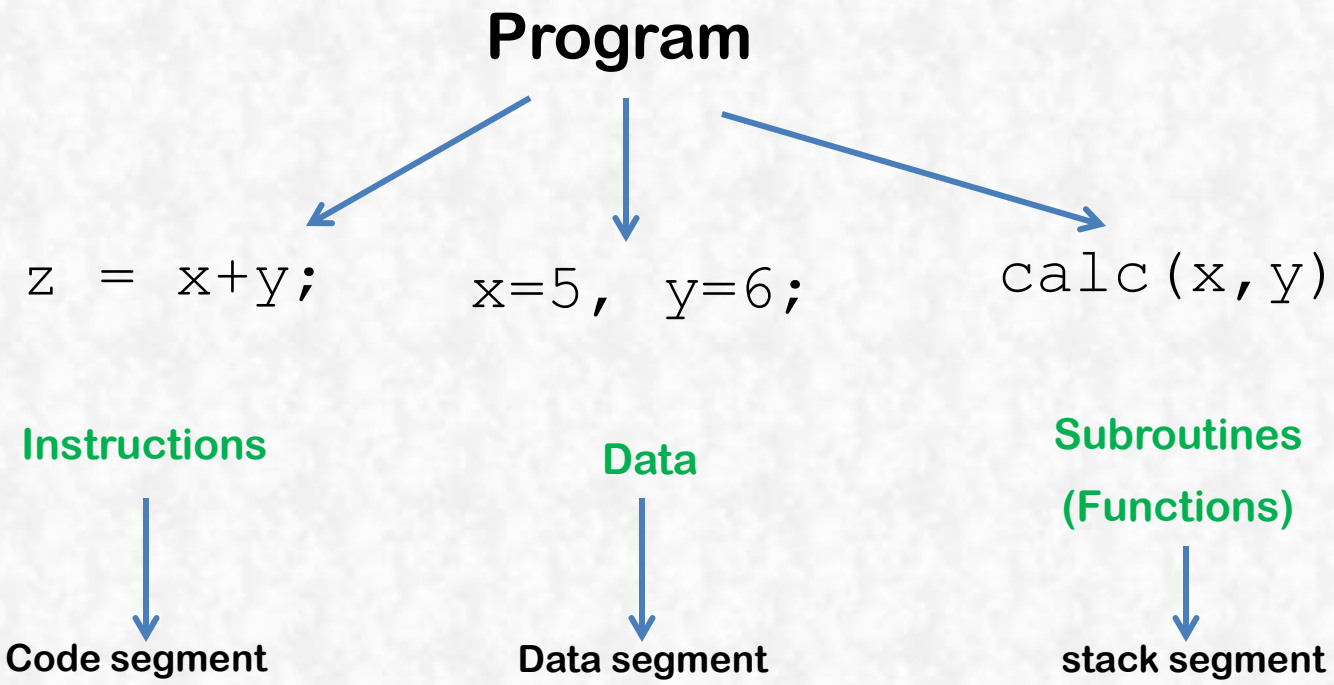


Memory
(Stack)

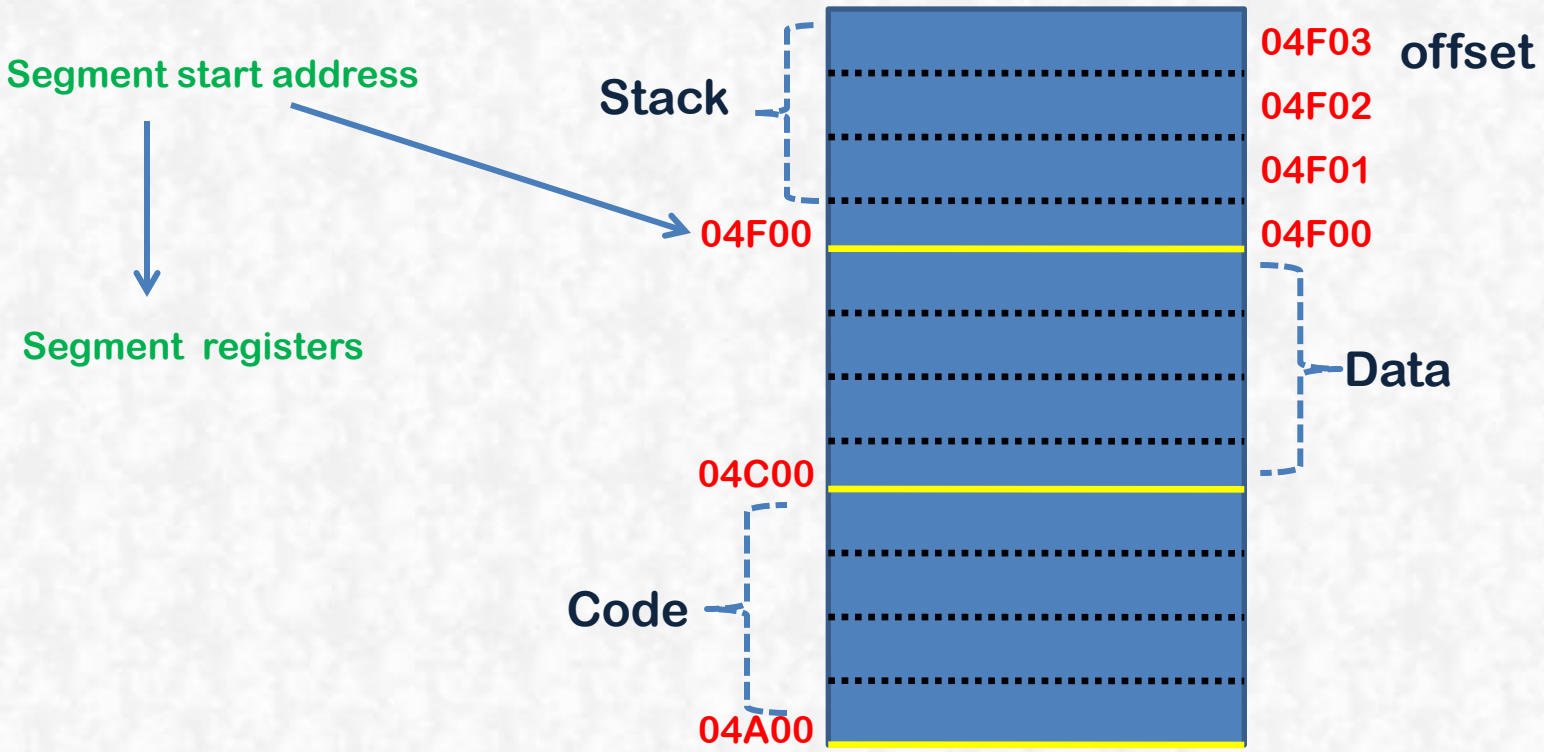
Addressing Data in Memory



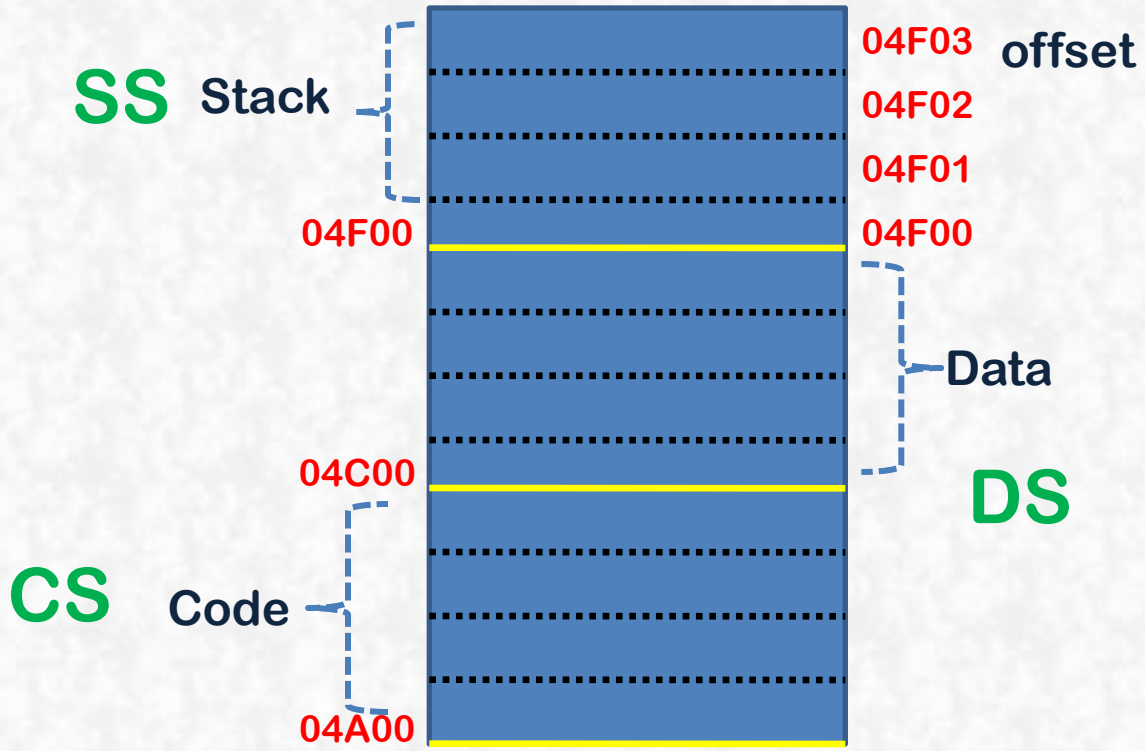
Segments and Addressing



Segments and Addressing



Segments and Addressing



Segments and Addressing

- A segment begins on a paragraph boundary, which is an address divisible by decimal 16, or hex 10 (i.e. always the rightmost hex digit is zero).
- It is unnecessary to store the zero digit in segment register.
- 038E0 H = 038E H= 038E[0] H.
- All memory locations within a segment are relative to the segment starting address.

Segments and Addressing

Consider DS= 038E(0)H, offset = 0032H What is the actual memory address used by the processor?

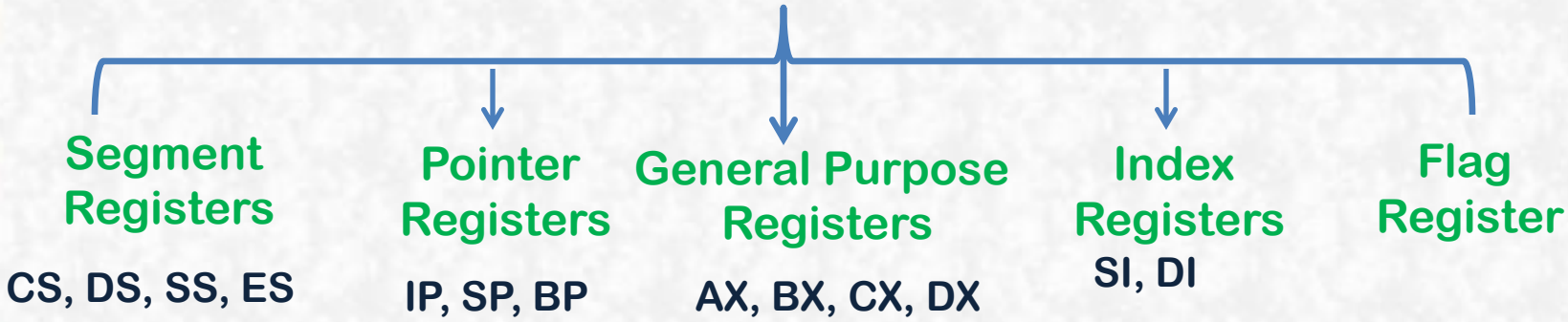
$$\begin{array}{rll} & 038E0H & \text{Data segment start address} \\ + & 0032\,H & \text{Offset} \\ \hline & 03912\,H & \text{Actual Address} \end{array}$$

Segments and Addressing

Give me one segment: offset value corresponding to this actual address 28F30H?

$$\begin{array}{r} 28F00H \\ +\ 0030\ H \\ \hline 28F30\ H \end{array} \qquad \begin{array}{r} 28F30H \\ +\ 0000\ H \\ \hline 28F30\ H \end{array}$$

Registers


Segment Registers
CS, DS, SS, ES
Pointer Registers
IP, SP, BP
General Purpose Registers
AX, BX, CX, DX
Index Registers
SI, DI
Flag Register

Registers

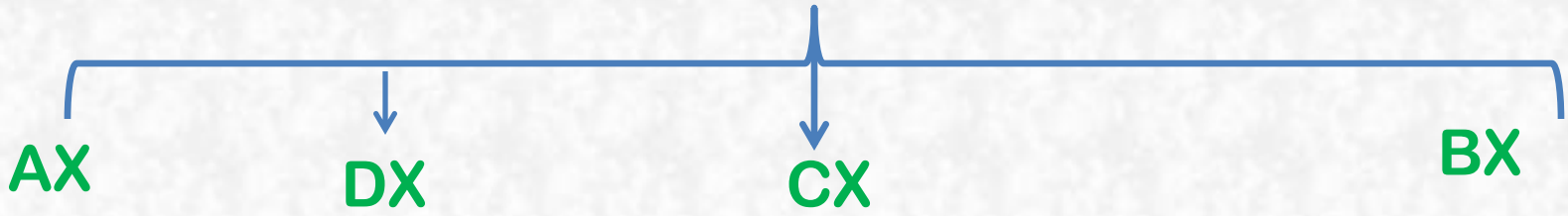
IP: offset address of the next instruction that is to execute.

The address used by the processor to locate the instruction will be ..... CS:IP

SP: offset address

The address used by the processor to locate the stack data will be ..... SS:SP SS:BP

General Purpose Registers



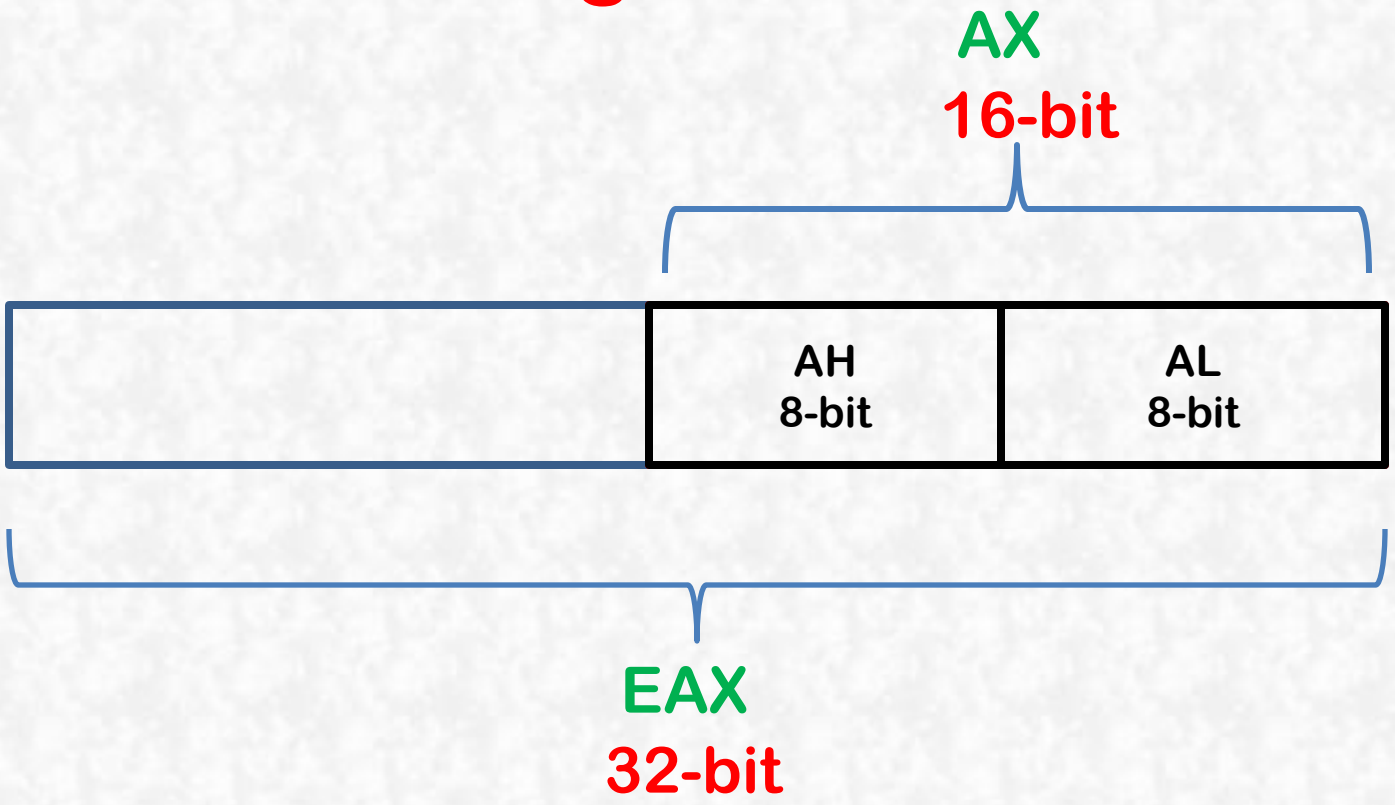
- I/O
- Arithmetic

- I/O
- MUL, DIV of large numbers
  AX:DX

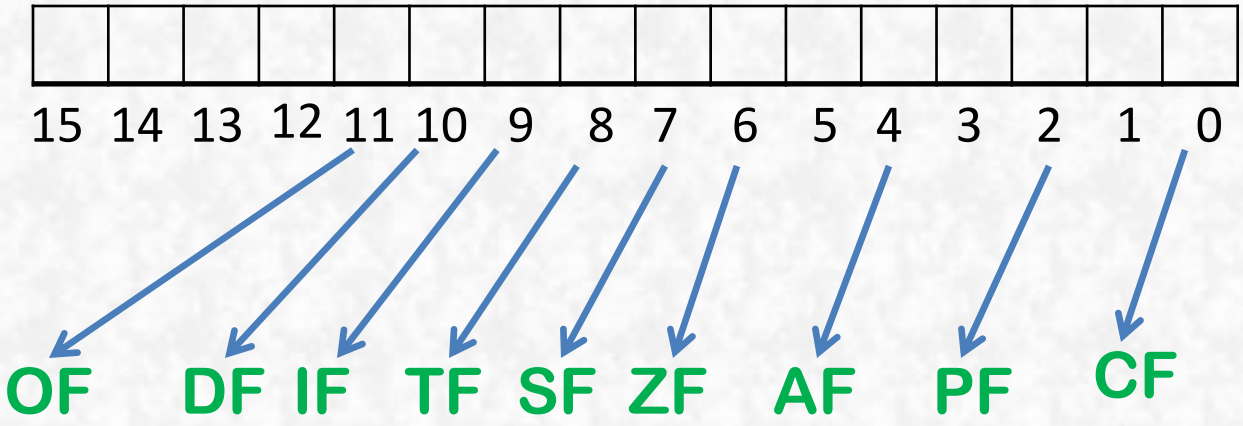
- Value that controls the number of times a loop is repeated
- Value to shift bits left or right.

- the only general-purpose register which may be used for indirect addressing MOV [BX], AX
- Computation
- String addressing and indexing

Registers

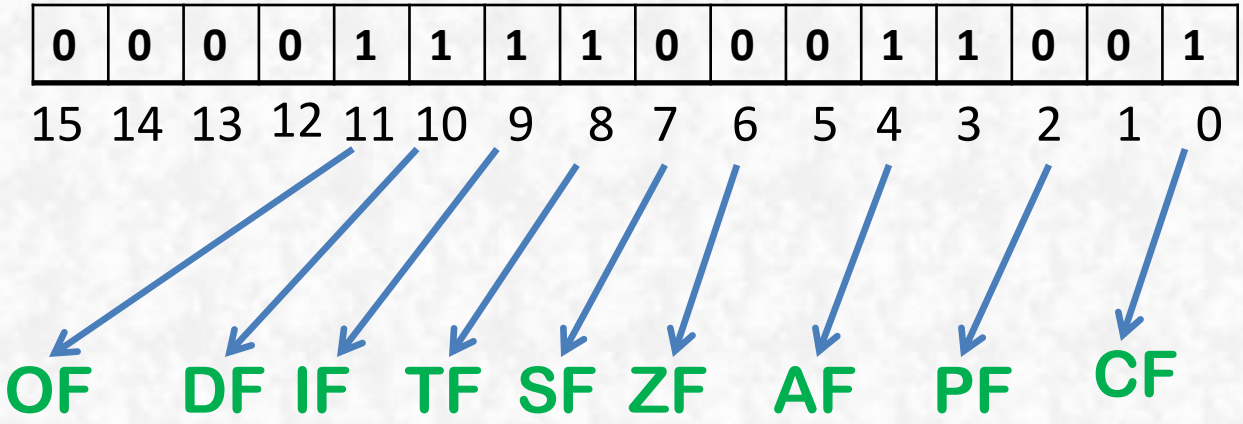


Flag Register



Flag Register

- A flag register has a value 0F19H, write the current status of your system.



Features of Operating System

BIOS Boot Process

Turning on the power

→ Processor reset state

→ CS = FFFF[0] H
IP =0000

↓

FFFF0H, entry point to BIOS in ROM

↓

- Interrupt Vector Table
- BIOS data area

↓

BOIS checks disk for the system files and accesses the boot strap loader.

← the boot strap loader loads system files from the disk into memory and transfer control to them.

BIOS Boot Process

Turning on the power

Processor reset state

CS = FFFF[0] H
IP =0000

FFFF0H, entry point to BIOS in ROM

- ✓ Clear memory locations to zero.
- ✓ Perform parity check of memory.

•Interrupt Vector Table
•BIOS data area

BOIS checks disk for the system files and accesses the boot strap loader.

the boot strap loader loads system files from the disk into memory and transfer control to them.

BIOS Boot Process

Turning on the power

Processor reset state

CS = FFFF[0] H
IP =0000

FFFF0H, entry point to BIOS in ROM

- ✓ First instruction to be executed FFFF0H.
- ✓ BIOS contains set of routines to provide device support (checks various ports to identify and initialize devices, services for I/O)

- •Interrupt Vector Table
- •BIOS data area

BOIS checks disk for the system files and accesses the boot strap loader.

the boot strap loader loads system files from the disk into memory and transfer control to them.

BIOS Boot Process

Turning on the power

Processor reset state

CS = FFFF[0] H
IP =0000

FFFF0H, entry point to BIOS in ROM

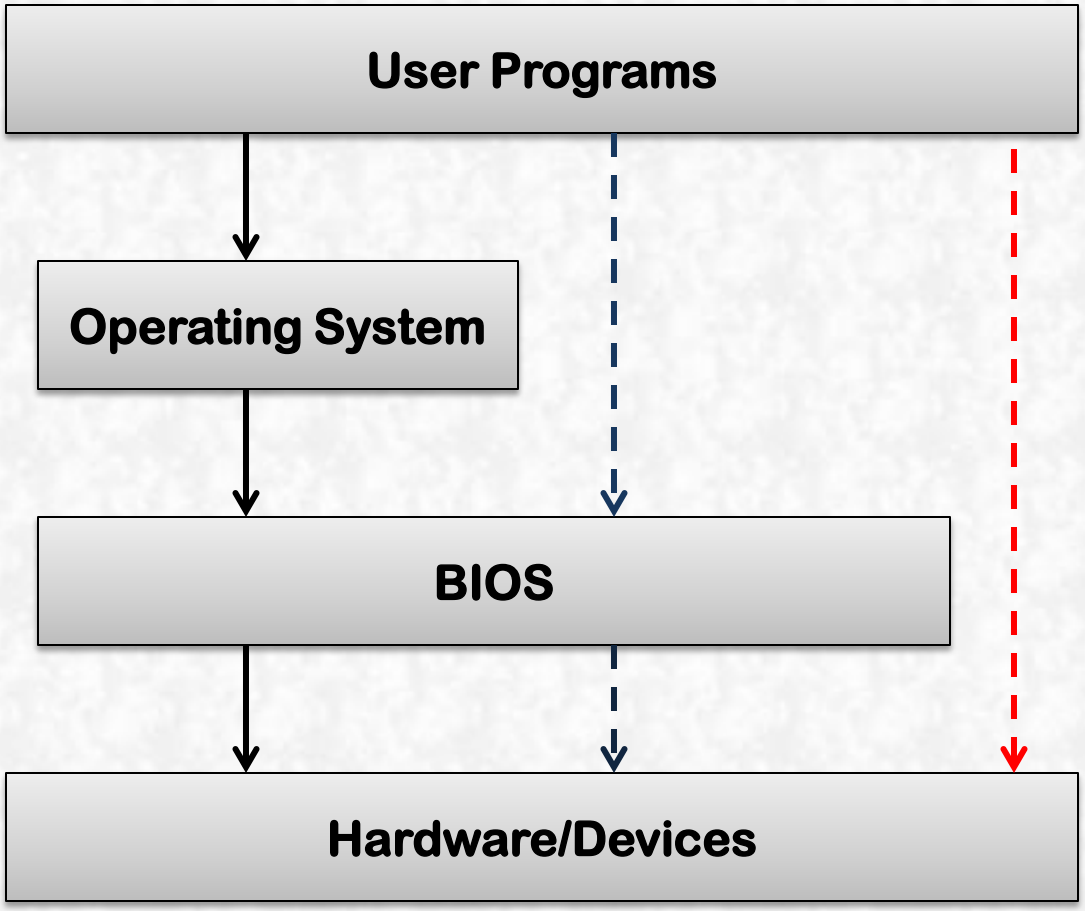
- ✓ IVT, begins at location 0 and contains 256 4-byte addresses in the form (segment: offset) for handling interrupts.
- ✓ BIOS data area begins at location 40[0], status of attached devices.

- •Interrupt Vector Table
- •BIOS data area

BOIS checks disk for the system files and accesses the boot strap loader.

the boot strap loader loads system files from the disk into memory and transfer control to them.

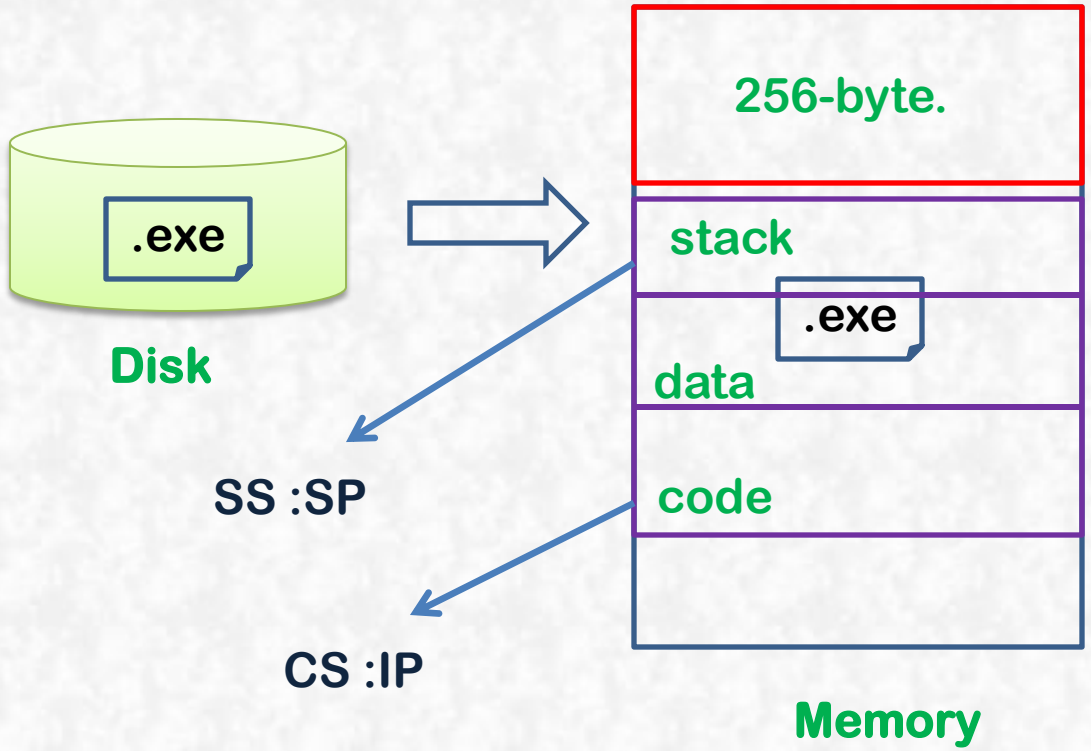
I/O Interface



Types of executable programs

- .COM consists of one segment that contains code, data, and stack.
- .COM could be used as small utility program or a resident program in memory.
- .EXE consists of separate segments for code, data, and stack.

Program loader



Program segment prefix, PSP

- Store program state
- Get command line arguments.

Loads the address of PSP in DS and ES.

Transfers control to the program for execution, beginning with first instruction at code segment with offset 0.

The Stack

- Saves return address when a program calls a subroutine.
- Saves data that will pass to a subroutine.
- Saves the current status of system registers, so it can be used for other calculations.
- SS:SP
- It stores data at the highest location in the segment and stores data downward through memory.

The Stack

- Portion of the stack that is reserved for a particular routine is called stack frame.
- PUSH
- POP
- Suppose you need to push the contents of AX=026B, and BX=04E3 and SP=36.

The Stack

AX=026B
BX=04E3

offset	Stack frame
34	0000
32	0000
30	0000
2E	0000

SP=36

PUSH AX

Decrements SP by 2
Stores contents of AX

AX=026B
BX=04E3

SP=36

offset	Stack frame
34	0000
32	0000
30	0000
2E	0000

PUSH AX

Decrements SP by 2
Stores contents of AX

AX=026B
BX=04E3

offset	Stack frame
34	6B02
32	0000
30	0000
2E	0000

SP=36

SP=34

PUSH BX

Decrements SP by 2
Stores contents of BX

AX=026B
BX=04E3

offset	Stack frame	
		SP=36
34	6B02	SP=34
32	E304	SP=32
30	0000	
2E	0000	

POP BX

restores contents of BX
Increments SP by 2

offset	Stack frame	
		SP=36
34	6B02	SP=34
32	E304	SP=32
30	0000	
2E	0000	

POP BX

restores contents of BX
Increments SP by 2

offset	Stack frame
34	6B02
32	E304
30	0000
2E	0000

SP=36

SP=34

BX=04E3

POP AX

restores contents of AX
Increments SP by 2

offset	Stack frame	SP=36
34	6B02	AX=026B
32	E304	BX=04E3
30	0000	
2E	0000	

What is meaning of SP=0?

Instruction Execution and Addressing

- An assembly language programmer writes a program in symbolic code and uses the assembler to translate it into machine code as a .COM or .EXE program.
- For a program execution, the system loads only the machine code into memory.
- Every instruction consists of at least one operation such as move, add, or return.
- Depending on the operation, an instruction may have one or more operands that reference the data.

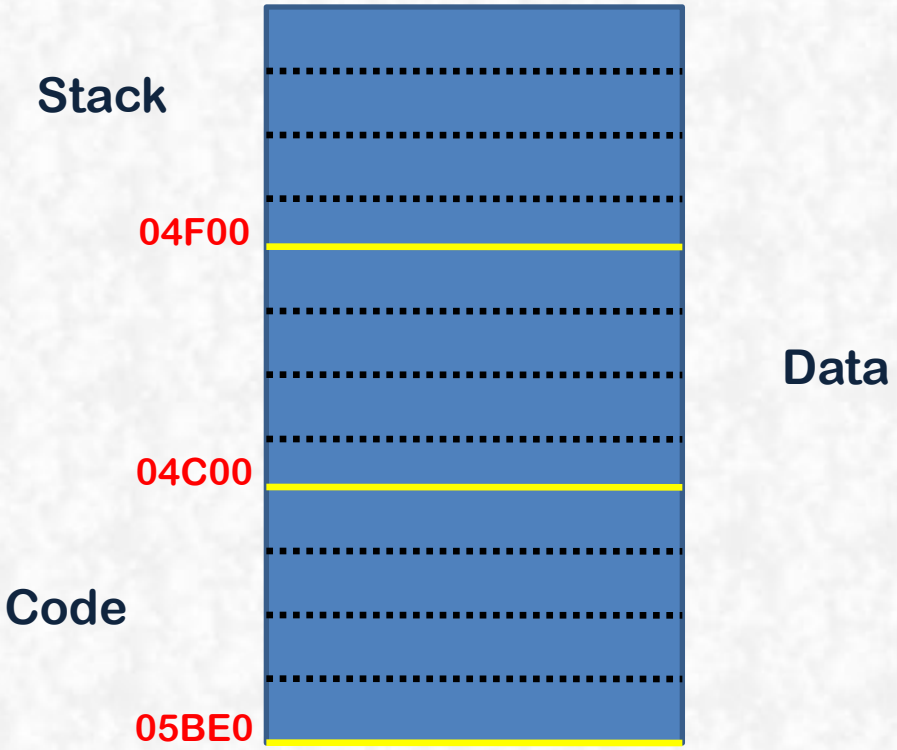
Instruction Execution and Addressing

- The basic steps the processor takes in executing an instruction: fetch, decode and execute.
- The fetch, decode and execute operations can be overlapped.

Notes

Assume the program loader determines to load .EXE program in memory beginning at location 05BE0H

- ✓ Program loader initializes CS=05BE, IP=0000
- ✓ If word size =8 bit, Instruction size = 2 bytes, the processor will increment IP by 2
- ✓ The address of the next instruction will be executed is : 05BE2



Notes

Assume that the next instruction to be executed is: `mov AL, [0016]`

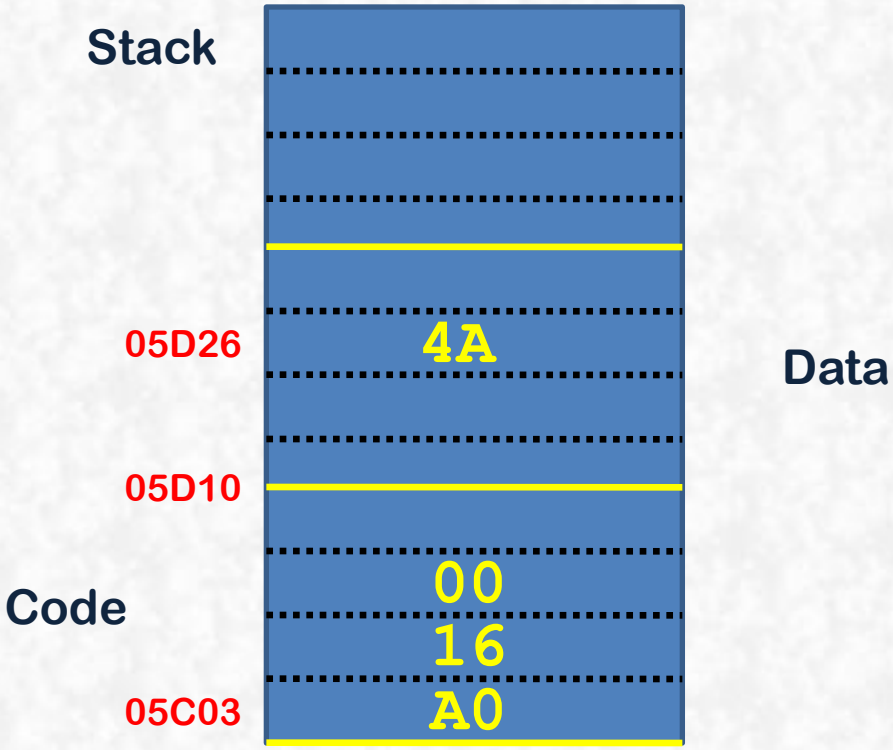
`mov AL, [0016]`

`A01600`

- ✓ How the program accesses data?

1. Initialize DS= 05D1H, offset =0016
2. The address will be 05D26H

AX		4A

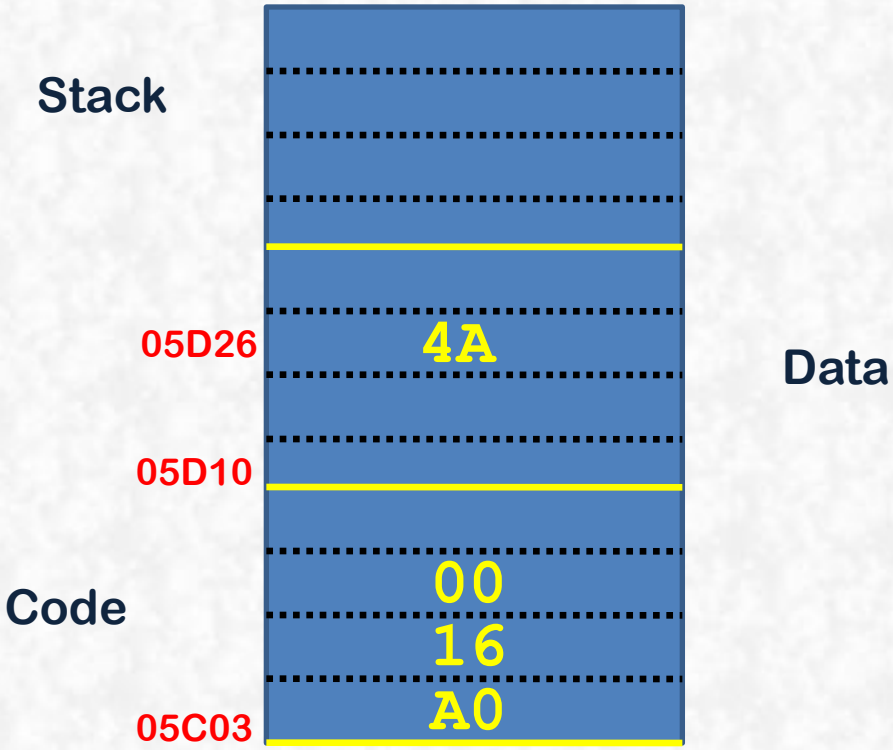


Notes

Assume that the next instruction to be executed is: `mov AL, [0016]`

Q What is the address of the next instruction to be executed? IP register will increment by ..

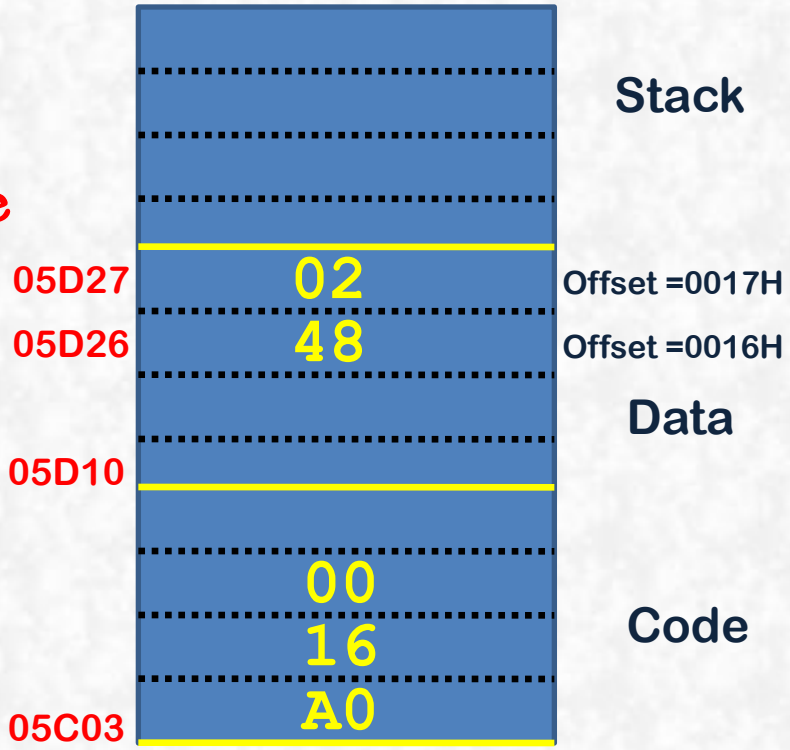
1. 05C06
2. 3



Notes

Assume that the next instruction to be executed is: `mov [0016],AX`

An Instruction may access more than one byte at a time.



Instruction Operands

- An Instruction may have zero, one, two, or three operands.

```
WORDX DW 0
……..
MOV CX, WORDX
MOV CX, 25
MOV CX, BX
MOV CX, [BX]
```