AACS3064
Computer Systems Architecture

Chapter 3: Addressing Data in Memory & Segment

Chapter Overview

1) Data Storage Sizes
2) Data Addressing
3) Segmented Memory Management
4) Program Execution Registers

1. Data Storage Sizes

1. Data Storage Sizes

Data Storage Sizes

NAME	LENGTH
NIBBLE	4-bit
BYTE	8-bit
WORD	2-byte (16-bit)
DOUBLEWORD	4-byte (32-bit)
QUADWORD	8-byte (64-bit)
PARAGRAPH	16-byte (128-bit)

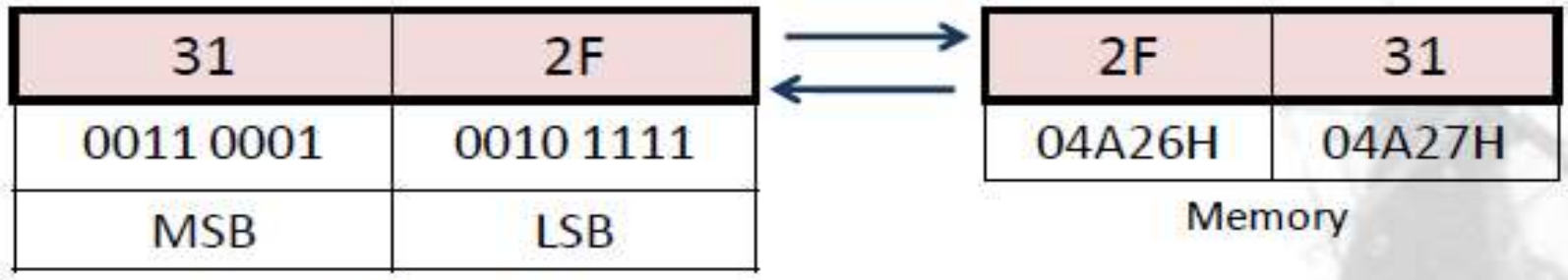
2. Data Addressing

2. Data Addressing

Data Addressing – Little Endian Order

- LSB is stored at the first memory address.
- Reverse-byte sequence
  - 2F (low order byte) → low memory address
  - 31 (high order byte) → high memory address

E.g.

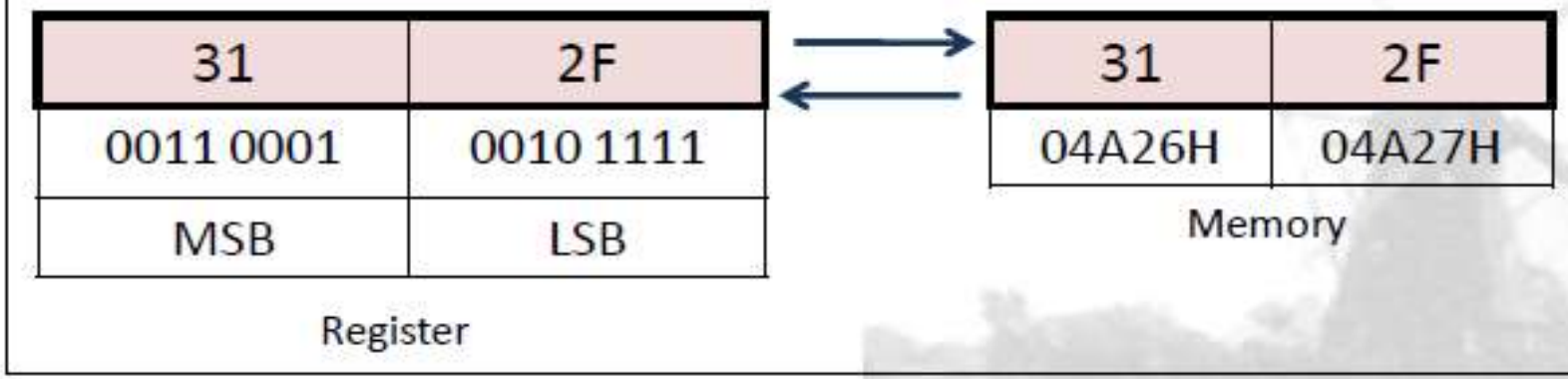


2. Data Addressing (Continued)

Data Addressing – Big Endian Order

- MSB is stored at the first memory address.
- Normal Sequence
  - 2F (low order byte) → high memory address
  - 31 (high order byte) → low memory address

E.g.



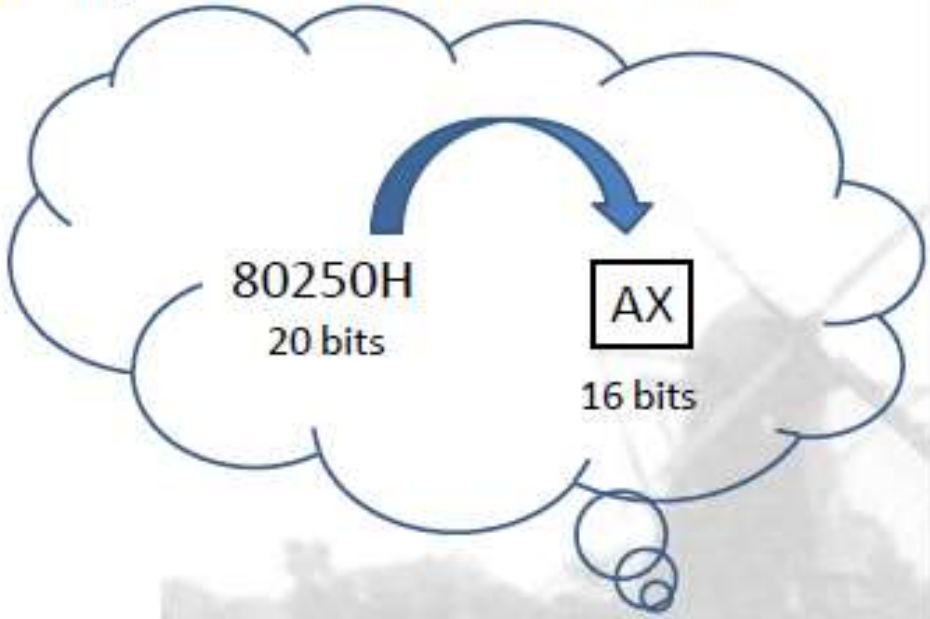
3. Segmented Memory Management

3. Segmented Memory Management

Segments and Addressing

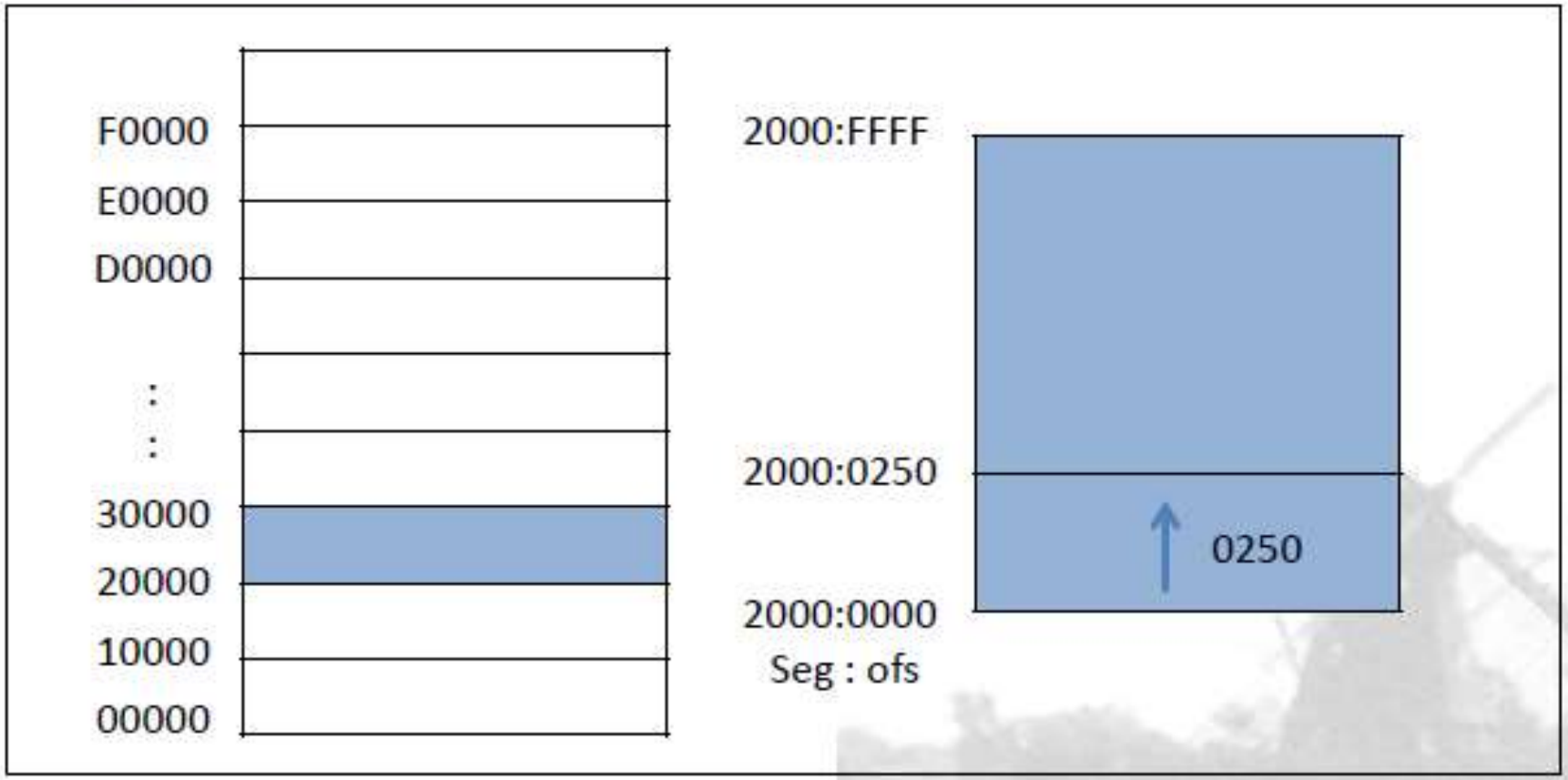
- **Real-address Mode :**
  x86 processor can access 1, 048, 576 bytes of memory (1 MByte) using 20 bit addresses in the range 0 to FFFFF hexadecimal.

- **Segmented memory**
  All of the memory is divided into 64KByte units called segment.



3. Segmented Memory Management(Continued)

Segmented Memory Map



3. Segmented Memory Management(Continued)

Segments and Addressing

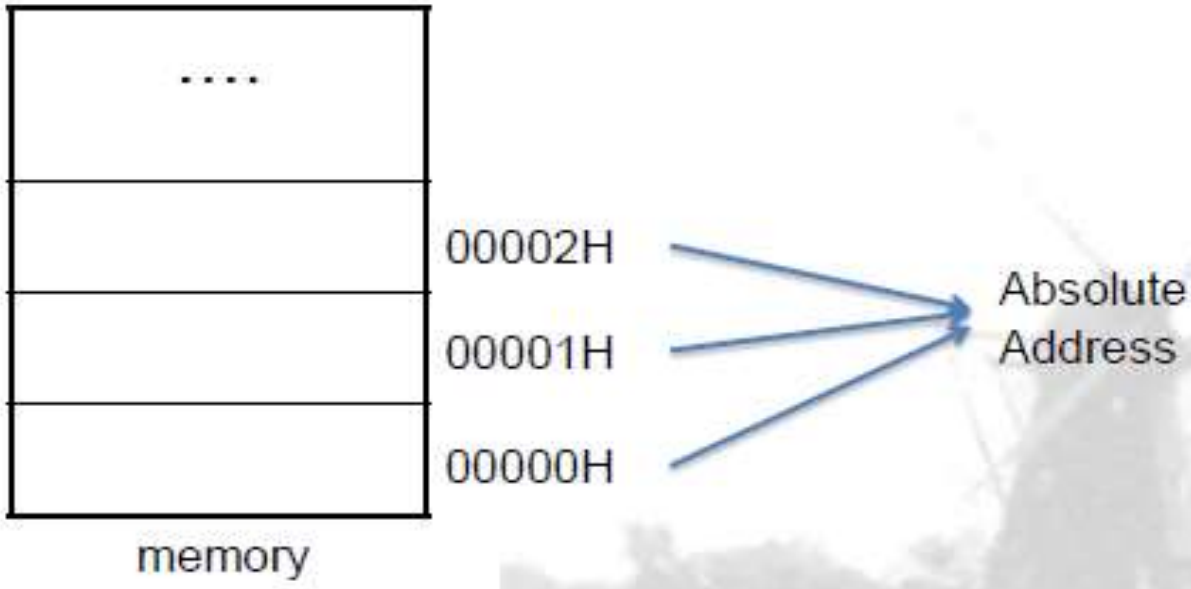
- Addressing schemes
  How memory address is referred.

- 2 methods:
  - (i) absolute address (physical address)
  - (ii) segment : offset address (logical address)

3. Segmented Memory Management(Continued)

Absolute Addressing

- Is a 20-bit value that directly references a specific location in memory.
- ***E.g.*** 04A26H



3. Segmented Memory Management(Continued)

Segments : Offset Address

- Combines the starting address of a segment with an offset value.
- Consists of 2 portions:
  - ✓ Segment address
  - ✓ Offset address

} Real address

3. Segmented Memory Management(Continued)

Segments Address

- Segment address is stored in segment register without last digit.

  E.g.

  038E0H → 038EH

  binary form:

  0000 0011 1000 1110 [0000]

- Effectively, the 20-bit address is stored in the 16-bit segment register

3. Segmented Memory Management(Continued)

Offset Address

- The Distance in bytes from the segment address to another location within the segment.
- Offset address (16-bit) ranges from 0000H (0D) to FFFFH (65,535D)
- Each segment can be up to 64KB in size

3. Segmented Memory Management(Continued)

20-bit Linear Address Calculation

- To obtain actual or absolute address of memory location from segment : offset address, the processors involves:

  (i) convert 16-bit segment address into 20-bit address

  (ii) add the offset address

Physical Address = (Logical Address * 10H) + Offset Address

3. Segmented Memory Management(Continued)

20-bit Linear Address Calculation

- Suppose a variable's hexadecimal segment-offset address is

08F1:0100

08F1 x 10H = 08F10H

Adjusted Segment Value :	0 8 F 1 0
Add the offset :	0 1 0 0
Absolute address:	0 9 0 1 0

3. Segmented Memory Management(Continued)

Segments

- Special areas of memory containing the code, data and stack information.
- OS use them to keep track of locations of individual program segments.
- 3 types :
  - ✓ **Code Segment**
  - ✓ **Data Segment**
  - ✓ **Stack Segment**

3. Segmented Memory Management(Continued)

Segments

Code Segment
Hold the machine instructions

Data Segment
Hold program's defined data and constants

Stack Segment
Hold local function variables and function parameters.

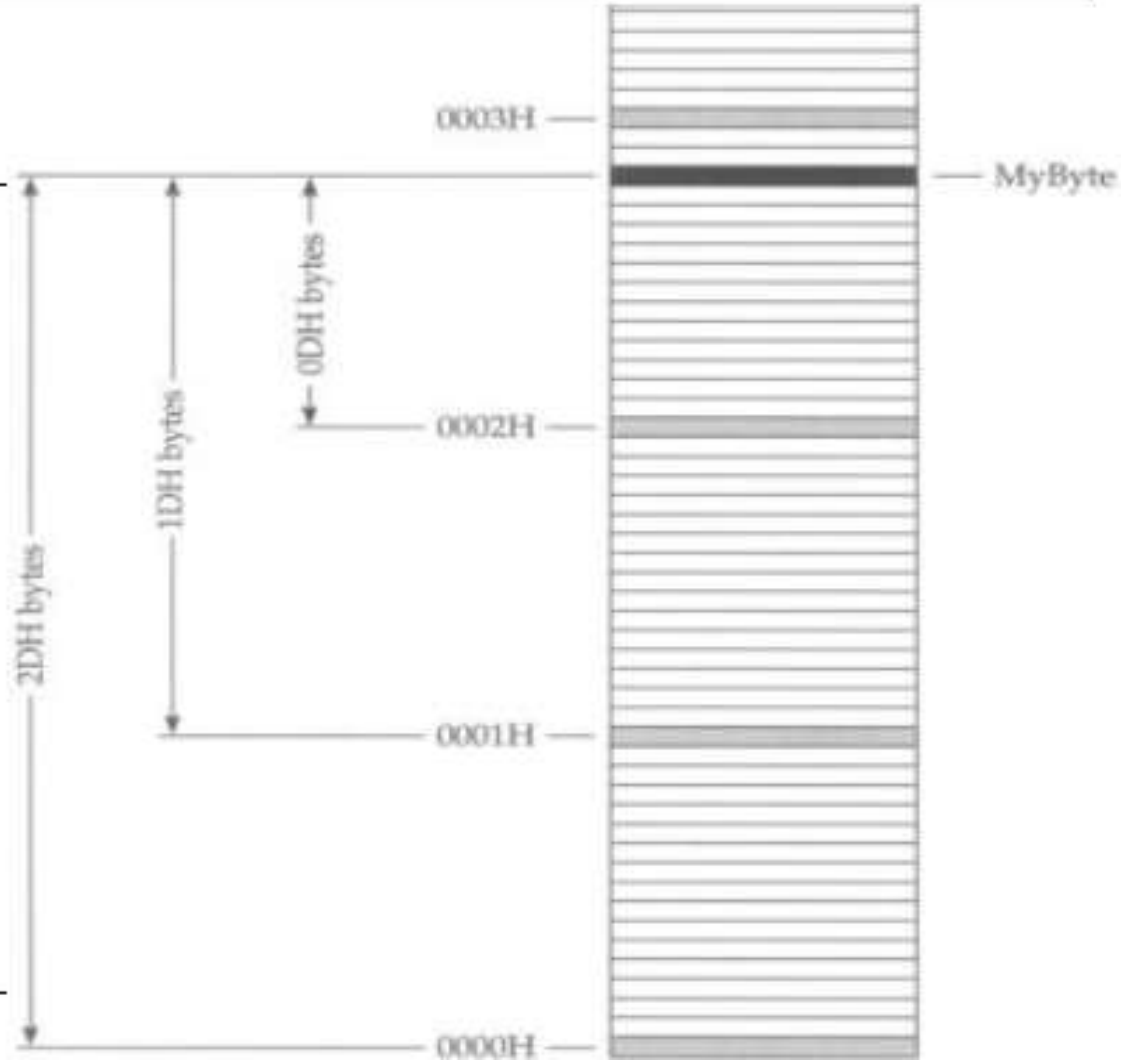
3. Segmented Memory Management(Continued)

Segments

Segments can overlap.

E.g.
Address of a byte data `MyByte` is given as 0001:001D.

possible addresses :
0000H : 002DH
0002H : 000DH
0001H : 001DH



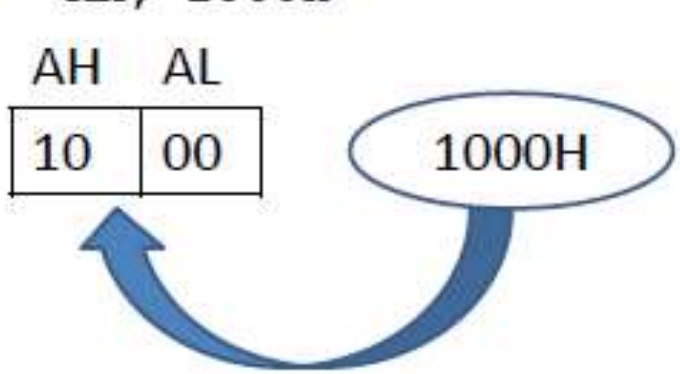
3. Segmented Memory Management(Continued)

Initializing data segment register

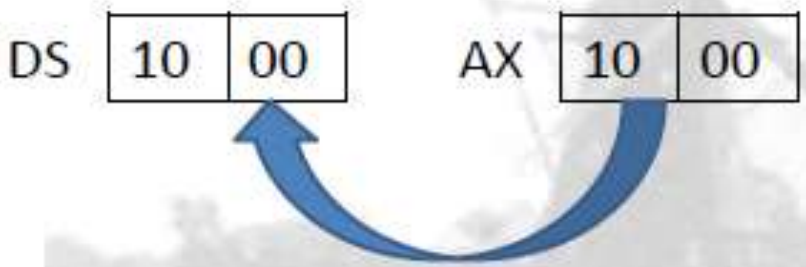
- Numerical (immediate) values cannot be moved directly into the segment register.
- It is a 2-step process:

```
MOV    AX, 1000H
MOV    DS, AX
```

```
MOV    AX, 1000H
```




```
MOV    DS, AX
```

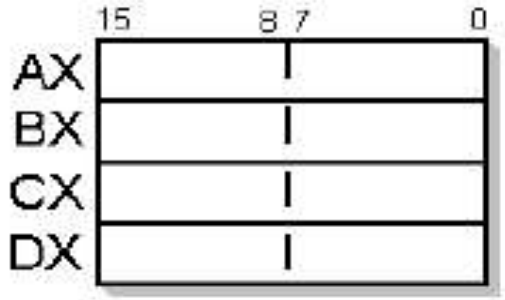


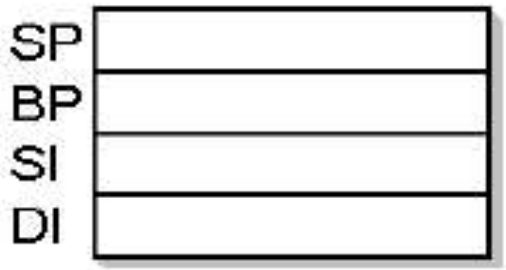
4. Program Execution Registers

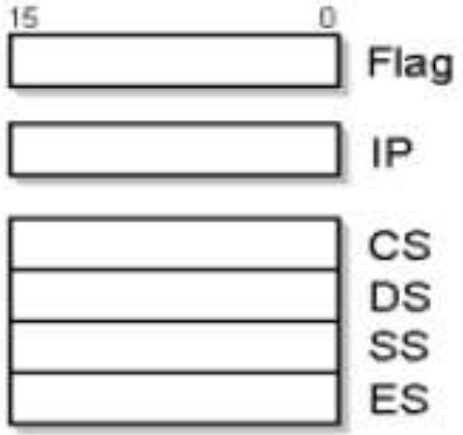
4. Program Execution Registers

Registers

- Named high speed storage locations inside the CPU.
- Use to store information temporarily.
- In 8086 :
  - All registers are 16-bit registers.
  - The general purpose registers can be accessed as either 8-bit or 16-bit registers.







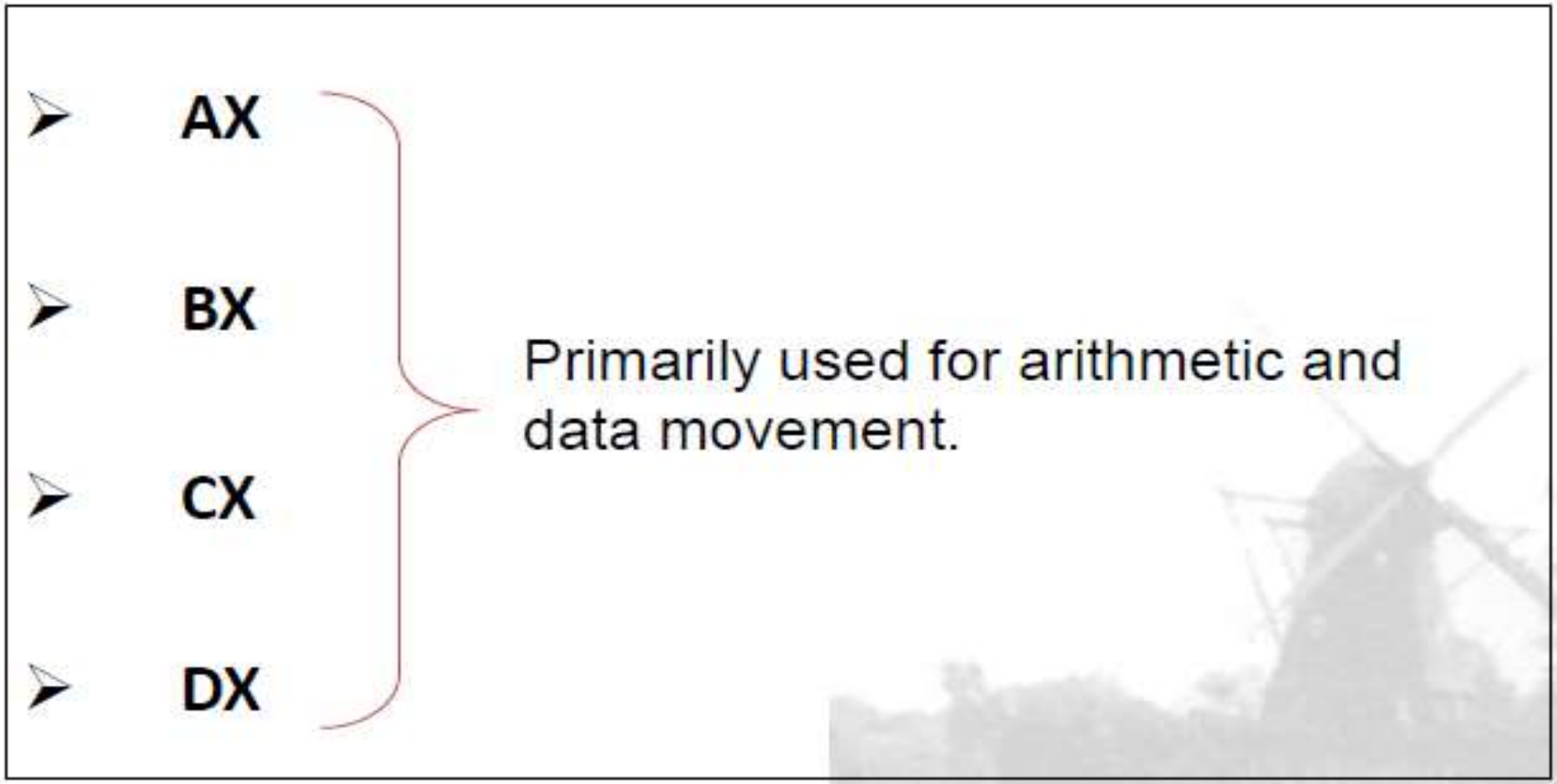
4. Program Execution Registers (Continued)

Registers of 8086 Microprocessor

Category	Bits	Register Names
General	16	AX, BX, CX, DX
	8	AH, AL, BH, BL, CH, CL, DH, DL
Pointer	16	SP, BP, IP
Index	16	SI, DI
Segment	16	CS, DS, SS, ES
Flag	16	CF, AF, ZF, TF, DF, SF, OF, PF

4. Program Execution Registers (Continued)

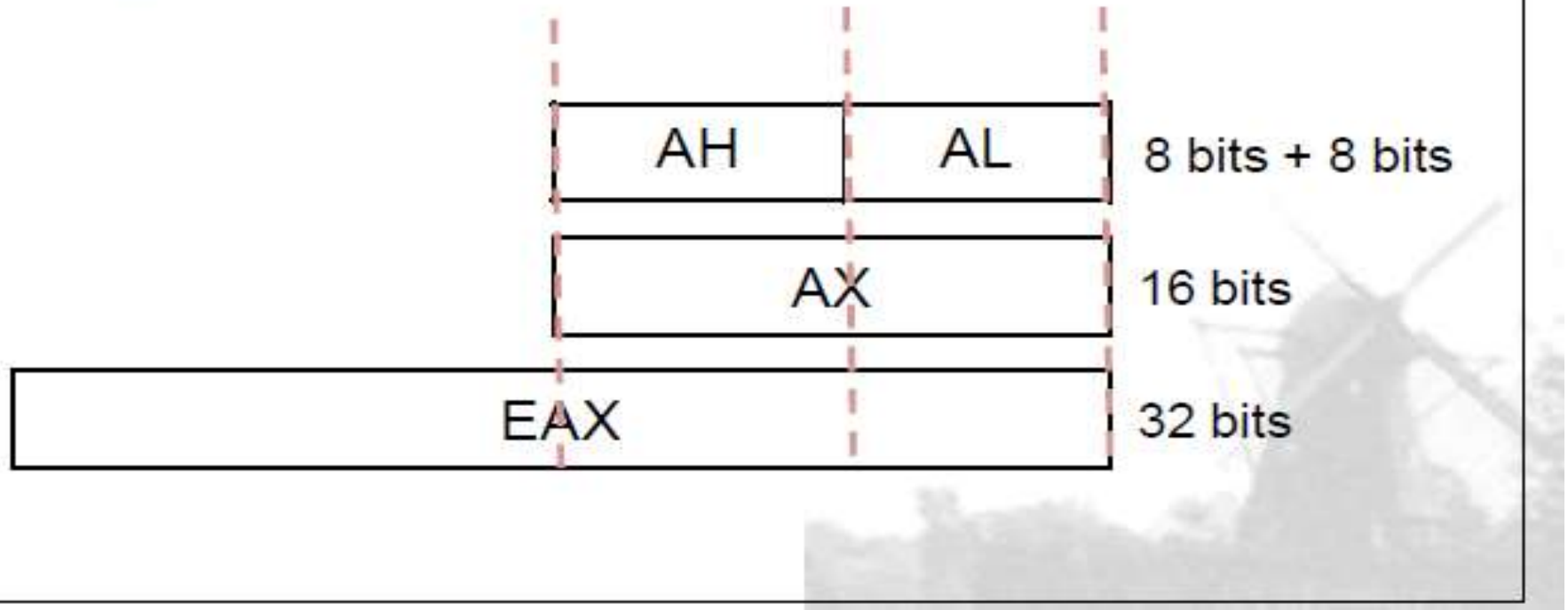
General Purpose Registers



4. Program Execution Registers (Continued)

General Purpose Registers

- Can be addressed as 8-bit registers.
- E.g.



4. Program Execution Registers (Continued)

General Purpose Registers

E.g. :

AX	
0011 0000	0011 1001

AX =

AH =

AL =

4. Program Execution Registers (Continued)

General Purpose Registers

- Specialized Uses :
  - Different registers are used for different functions.
  - The first letter of each general purpose register indicates its use.

  - ✓ **AX** : Accumulator register
  - ✓ **BX** : Base register
  - ✓ **CX** : Count register
  - ✓ **DX** : Data register

4. Program Execution Registers (Continued)

General Purpose Registers

- ✓ **AX** :
  - Used for operations involving input /output and most arithmetic.
  - E.g. AX is automatically used by multiplication and division instructions.

- ✓ **BX** :
  - The only general-purpose register that can be used as an index to extend addressing.
  - Used for computations.
  - Can also be combined with DI and SI as a base register for special addressing.

4. Program Execution Registers (Continued)

General Purpose Registers

- ✓ **CX :**
  - Contain a value to control the number of times a loop is repeated.
  - Contain a value to shift bits left or right.
  - For computations

- ✓ **DX :**
  - Input / Output Operations
  - Multiply and divide operations that involve large values.

4. Program Execution Registers (Continued)

Index Registers

- Used for indexed addressing
  - Hold offset address and associated with other base register.
  - ✓ **SI** : Source index Register
  - ✓ **DI** : Destination Index Register

4. Program Execution Registers (Continued)

Index Registers

- ✓ **SI** :
  - Required for some string (character) handling operations.
  - Associated with DS register.
- ✓ **DI** :
  - Required for some string operations.
  - Associated with ES register.

4. Program Execution Registers (Continued)

Pointer Registers

- ✓ **IP** : Instruction pointer register
  CS:IP
- ✓ **SP** : Stack pointer register
  SS:SP
- ✓ **BP** : Base pointer register
  SS:BP

Hold the offset address and associate with other base register

4. Program Execution Registers (Continued)

Pointer Registers

✓ **IP** :

- Contains the offset address of the next instruction that is to execute.
- Indicates the current instruction within the currently executing code segment.
- E.g.

Segment address in CS		39B40H
Plus offset address in IP	+	514H
Address of the next instruction		3A054H

4. Program Execution Registers (Continued)

Pointer Registers

✓ **SP** :

- SS:SP refers to the current word being processed in the stack.
- E.g.

Segment address in SS	4BB30H
Plus offset in SP	+ 412H
Address in stack	4BF42H

4. Program Execution Registers (Continued)

Pointer Registers

- ✓ **BP** :
  - Facilitates referencing parameters, which are data and addresses that a program passes via the stack.
  - SS:BP
  - Can also be combined with DI and SI as a base register for special addressing.

4. Program Execution Registers (Continued)

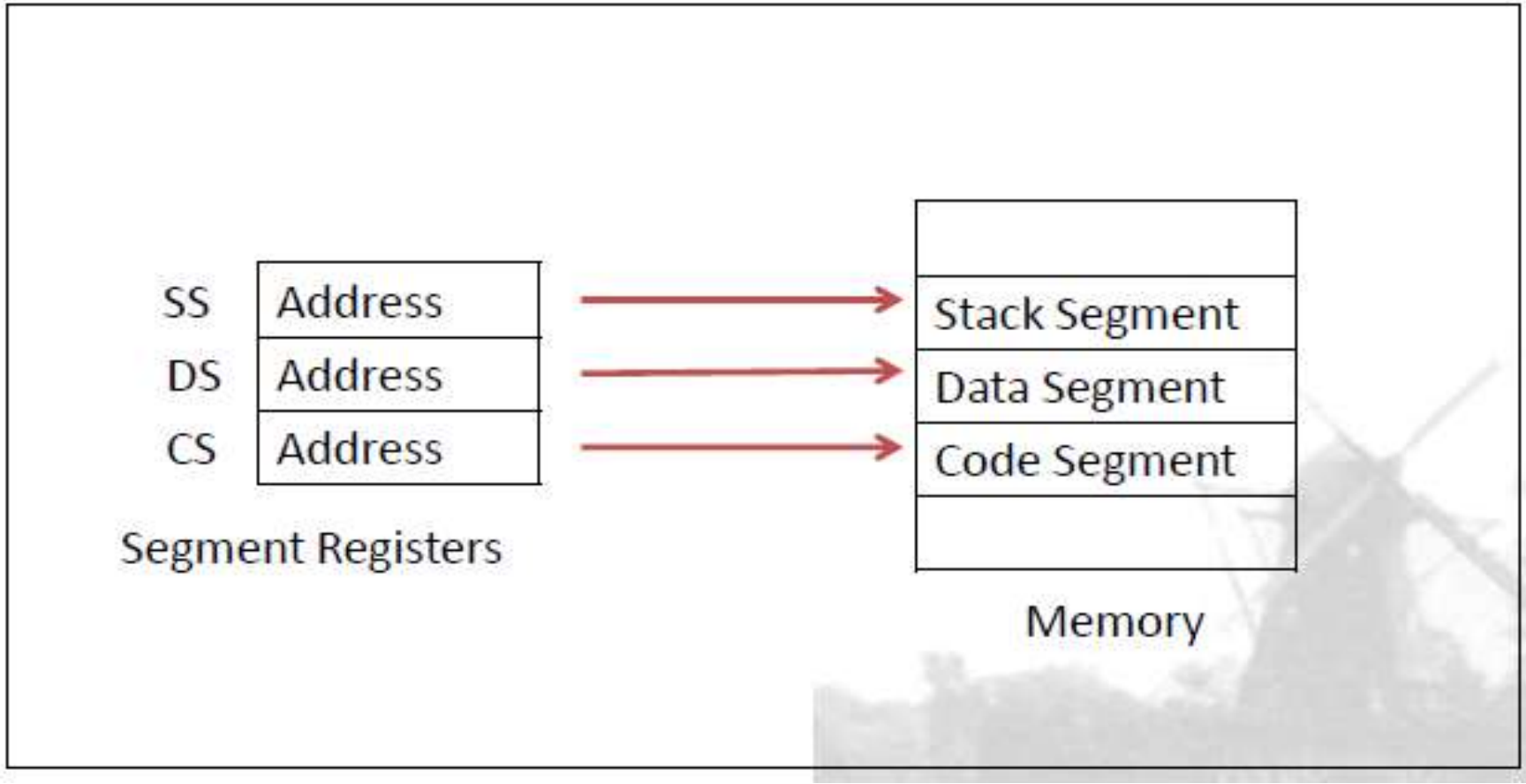
Segment Registers

- Addressing an area of memory known as the *current segment*.
- Store the starting address of particular segment.

  - ✓ **CS** : Code segment register
  - ✓ **DS** : Data segment register
  - ✓ **SS** : Stack segment register
  - ✓ **ES** : Extra segment register

4. Program Execution Registers (Continued)

Segment Registers



4. Program Execution Registers (Continued)

Segment Registers

✓ **CS :**

- Contains starting address of a program's code segment.
- (CS) address + IP offset value = address of instruction to be fetched for execution.

✓ **DS :**

- Contains starting address of a program's data segment.
- Instructions use this address to locate data.
- (DS) address + instruction offset value = reference to a specific byte location in the data segment

4. Program Execution Registers (Continued)

Segment Registers

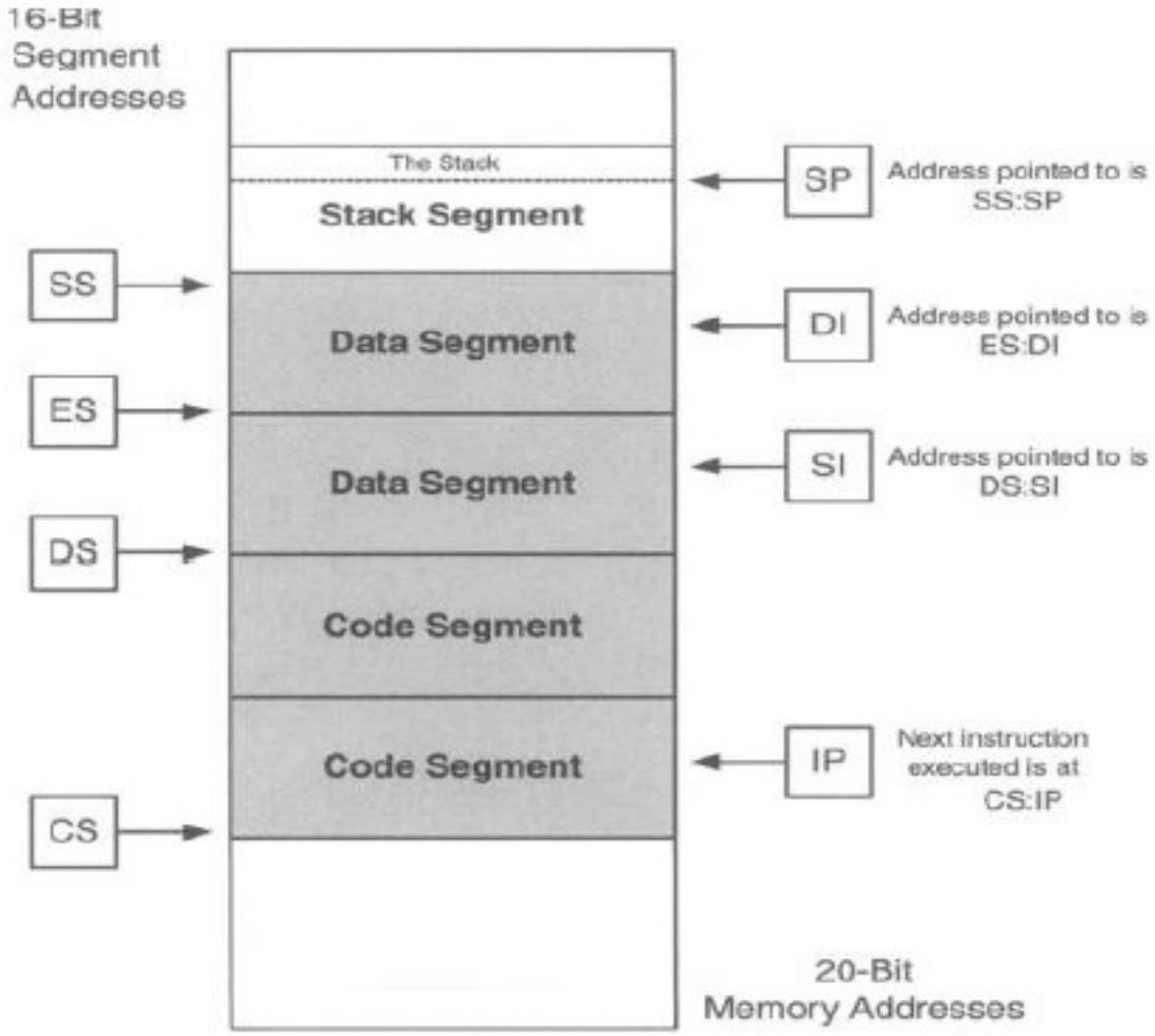
✓ **SS** :

- Permits the implementation of a stack in memory, which a program uses for temporary storage of addresses and data.
- (SS) Address + SP offset value = the current word in the stack being addressed.

✓ **ES** :

- Used by some string operations to handle memory addressing.
- Must initialize it with an appropriate segment address if required by a program.

4. Program Execution Registers (Continued)



4. Program Execution Registers (Continued)

Flag Registers

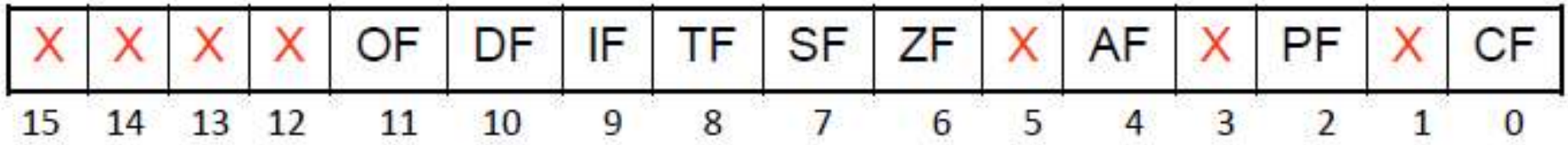
- Indicate the status of various activities.
- Many instructions involving comparisons and arithmetic change the status of the flags, which some instructions may test to determine subsequent action.

```
AX=0000  BX=0000  CX=0000  DX=0000
DS=17AD  ES=17AD  SS=17AD  CS=17AD

         SP=FFEE  BP=0000  SI=0000  DI=0000
         IP=0100   NV UP EI PL NZ NA PO NC
```

4. Program Execution Registers (Continued)

Flag Registers



CF : Carry Flag
AF : Auxiliary carry Flag
SF : Sign Flag

IF : Interrupt enable flag
DF : Direction flag
TF : Single step trap flag

PF : Parity Flag
ZF : Zero Flag
OF : Overflow Flag

4. Program Execution Registers (Continued)

Conditional or status Flags

- Reflect the outcome of arithmetic and logical operations

- **CF** : Indicates a carry after an arithmetic operations.
  Set when the result of an unsigned arithmetic operation is too large to fit into the destination.

- **OF** : Indicates overflow of a msb following arithmetic.
  Set when the result of a signed arithmetic operation is too large or too small to fir into the destination.

4. Program Execution Registers (Continued)

Conditional or status Flags

- **ZF** : Indicates that the result of an arithmetic or logic operation is zero.
- **SF** : Indicates arithmetic sign of the result after an arithmetic operation.
- **PF** : Set if the result contains an even number of 1 bits.
  Even number of bits (even parity)
  Odd number (odd parity).
  Used for error checking.

4. Program Execution Registers (Continued)

Conditional or status Flags

- **AF :**

  Is set when an arithmetic operation causes a carry from bit3 to bit 4 in an 8-bit operand.

  Important for BCD addition and subtraction;

  Only used for DAA and DAS instructions to adjust the value of AL after a BCD addition (subtraction).

4. Program Execution Registers (Continued)

Control Flags

- Control certain CPU operations.
- Programs can set or reset individual bits to control the operation.

- **TF** : Used for single stepping through a program.
- **IF** : Indicates that all external interrupts, are to be processed or ignored.
- **DF** : Determines left or right direction for moving or comparing string data.

Chapter Review

1) Data Storage Sizes
2) Data Addressing
3) Segmented Memory Management
4) Program Execution Registers