BACS1024 INTRODUCTION TO COMPUTER SYSTEMS

Chapter 4: Addressing Data in Memory and Segment

0. Overview

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

1. Data Storage Size

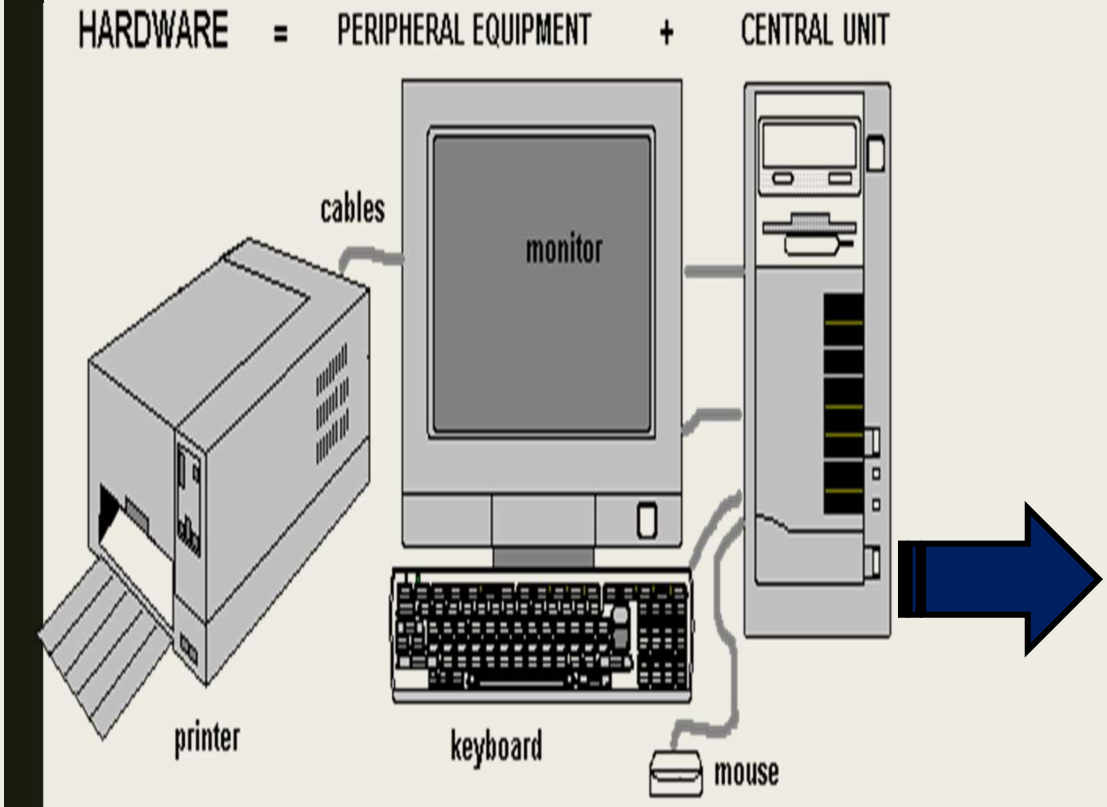
1. Data Storage Size

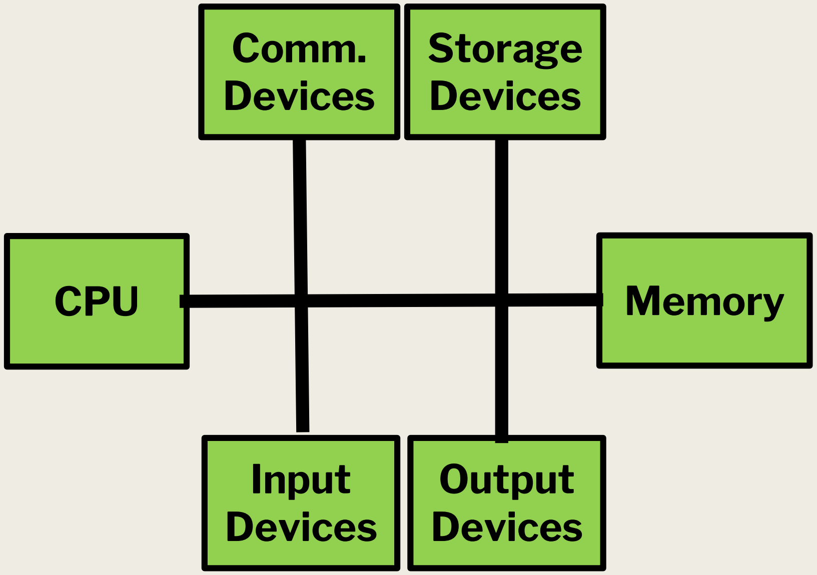
- The smallest & fundamental unit in computer is measure by bit.
- Data stored in memory in byte basis.
- There are other sizes available to facilitate data storage.

Storage size	Length (in bits)	Length (in bytes)
Bit	1	-
Byte	8	$2^0 = 1$
Word	16	$2^1 = 2$
Doubleword	32	$2^2 = 4$
Quadword	64	$2^3 = 8$
Paragraph	128	$2^4 = 16$

2. Data Addressing

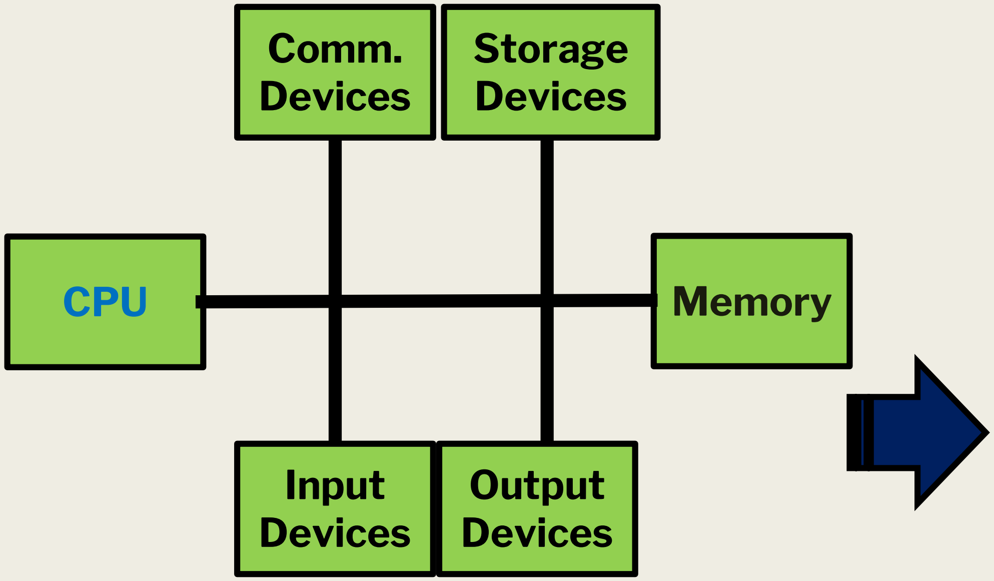
2. Data Addressing





Note: Comm. Devices = Communication Devices

2. Data Addressing



Comm. Devices

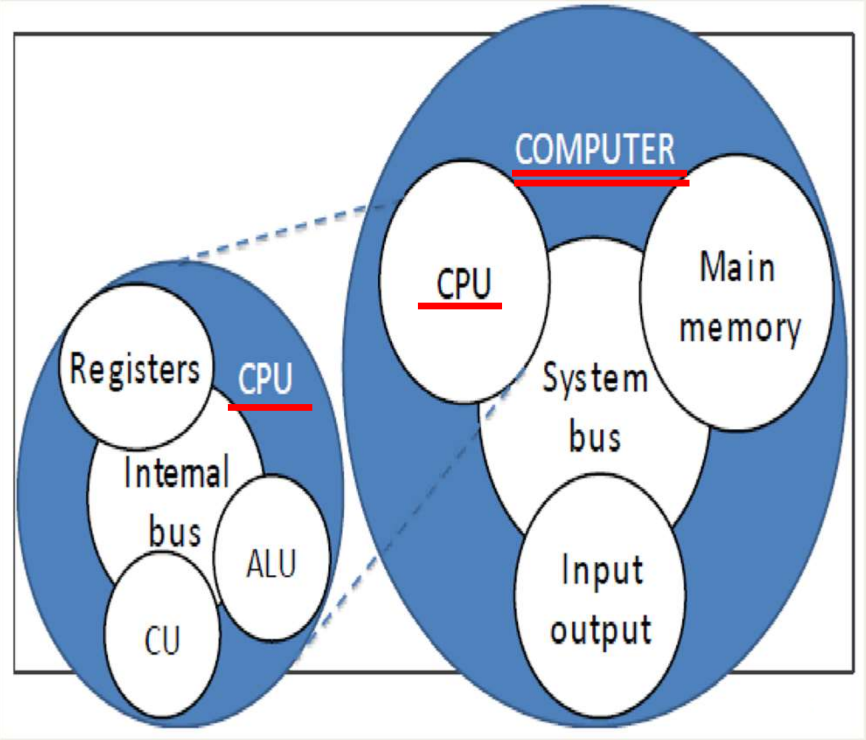
Storage Devices

CPU

Memory

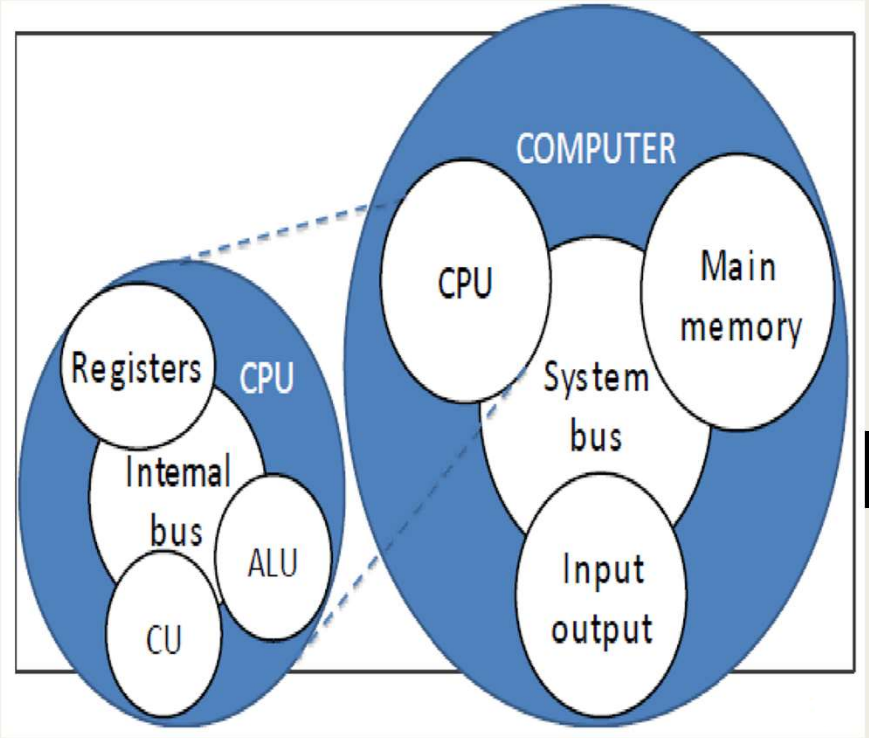
Input Devices

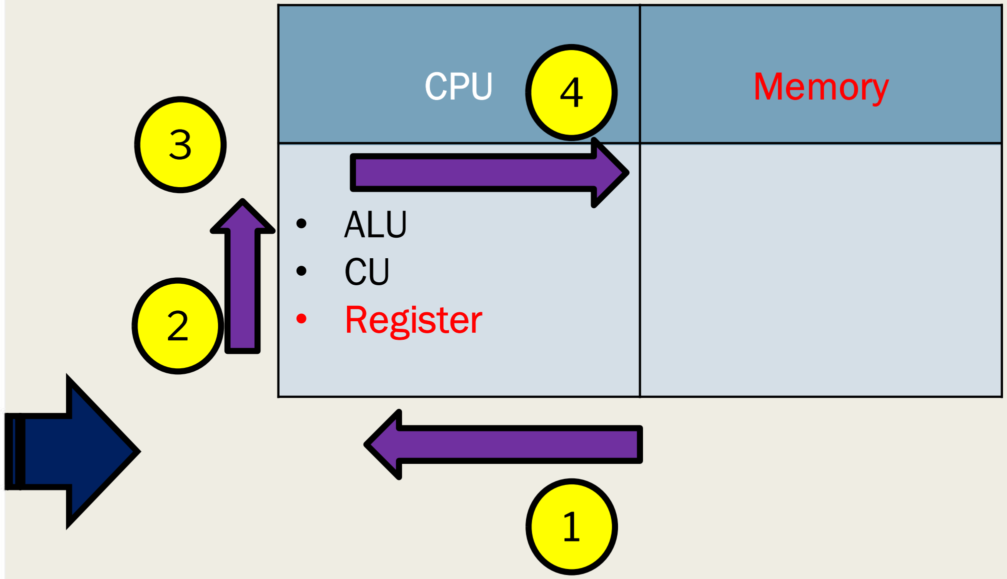
Output Devices



- Note: Comm. Devices = Communication Devices

2. Data Addressing





CPU	Memory
- ALU - CU - Register	

- **1. Fetch** : Read from memory
- **2. Decode** : Translate
- **3. Execute** : Process
- **4. Store** : Write to memory

2. Data Addressing

- **How data stored in memory?**
  - ❑ Computer memory consists of a sequence of storage cells.
  - ❑ Each cell is identified in hardware and software by its memory addresses of 20 bit basis.
  - ❑ For a memory with ***n*** number of storage cells in memory, its addresses are enumerated from ***0*** to ***n-1***, to store a consecutive sequence of bytes that represents a simple data value.

2. Data Addressing

- **How data stored in memory?**
  - ❑ The "value" of a digit is determined its value as a single digit and also by the position it holds in the complete number, its "significance".
  - ❑ These positions can be mapped to memory mainly in two ways:
    - ❖ increasing numeric significance with increasing memory addresses (or increasing time), known as ***little-endian***, and
    - ❖ decreasing numeric significance with increasing memory addresses (or increasing time), known as ***big-endian***

2. Data Addressing

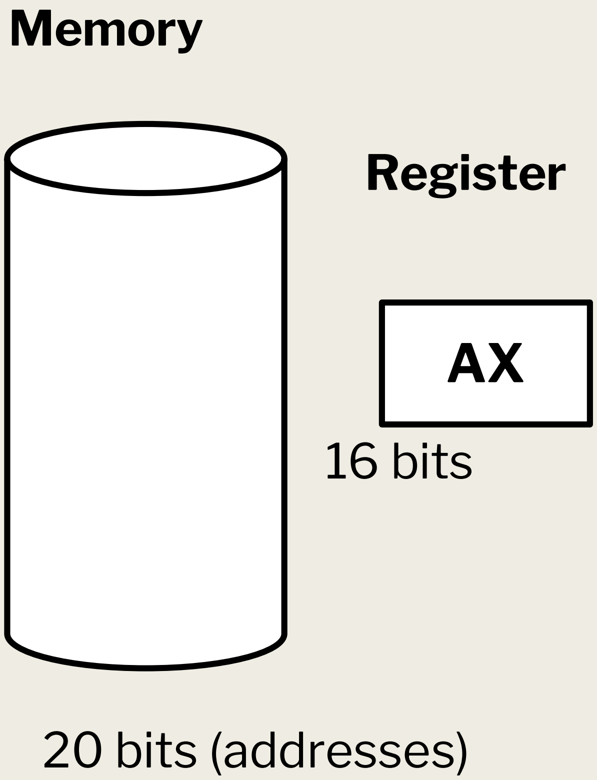
How data stored in memory?

<table>
<tr><th>Big endian order</th><th>Little endian order</th></tr>
<tr><td colspan="2"><strong>Little</strong> and <strong>big endian</strong> are two ways of storing multibyte data-types (int, float, etc)</td></tr>
<tr>
<td>
• Normal-byte sequence
• E.g.: 1024_{16}
❑ First byte of binary representation of the multibyte data-type is stored first
<u>Register</u>
<table><tr><td>10_{16}</td><td>24_{16}</td></tr><tr><td>MSB</td><td>LSB</td></tr></table>
<u>Memory</u>
<table><tr><td>10_{16}</td><td>24_{16}</td></tr><tr><td>00000H</td><td>00001H</td></tr></table>
</td>
<td>
• Reversed-byte sequence
• E.g.: 1024_{16}
❑ Last byte of binary representation of the multibyte data-type is stored first
<u>Register</u>
<table><tr><td>10_{16}</td><td>24_{16}</td></tr><tr><td>MSB</td><td>LSB</td></tr></table>
<u>Memory</u>
<table><tr><td>24_{16}</td><td>10_{16}</td></tr><tr><td>00000H</td><td>00001H</td></tr></table>
</td>
</tr>
<tr>
<td><u>Advantage</u>
• Easier for human to read (Left to right).</td>
<td><u>Advantage</u>
• Some computer operations may be simpler and faster to perform.</td>
</tr>
</table>

3. Segmented Memory Management

3. Segmented Memory Management

- **Segments & Addressing**
  - ❑ Real-address mode
    - ❖ The x86 processor can access 1, 048, 576 bytes (1MB) of memory using 20 bit addresses in the range of 00000H to FFFFFH.
  - ❑ Segmented memory
    - ❖ All of the memory is divided into 64KByte units called segment.



Memory

Register

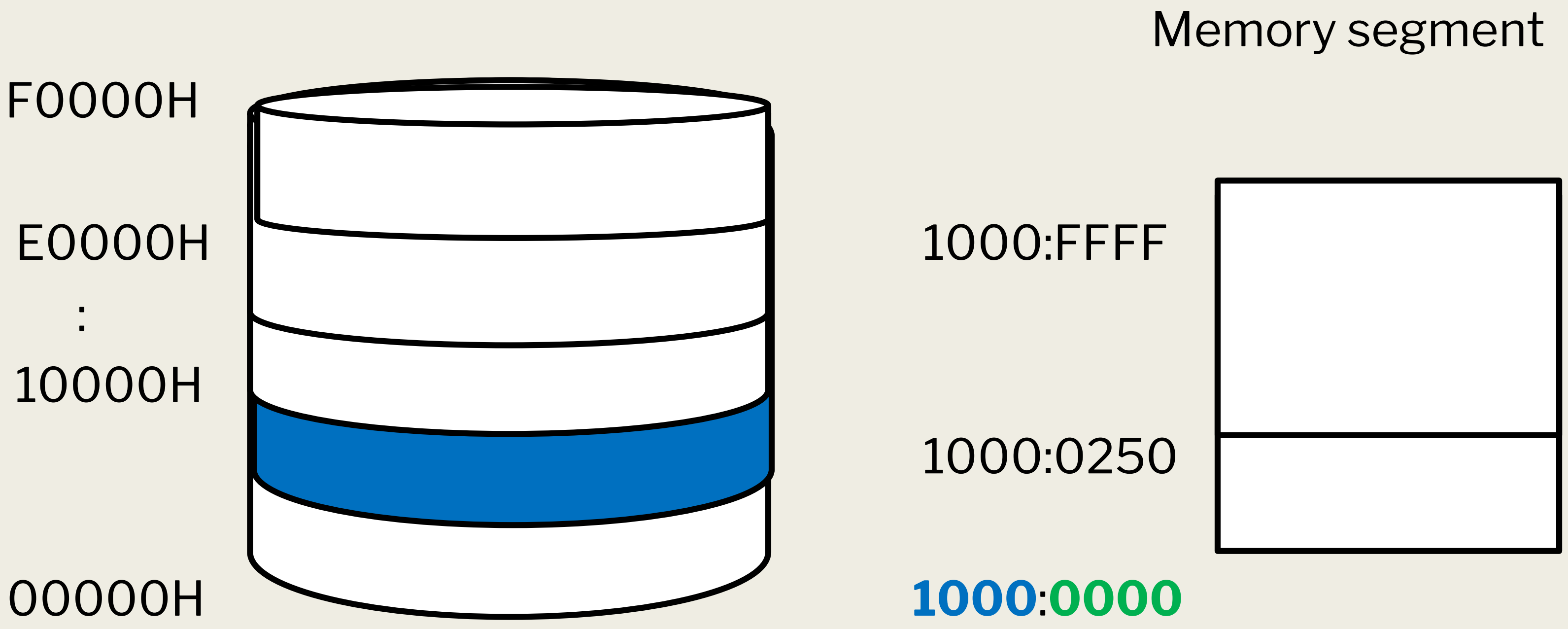
AX

16 bits

20 bits (addresses)

3. Segmented Memory Management

- **Segmented Memory Map**



Memory segment

F0000H

E0000H

:

10000H

00000H

1000:FFFF

1000:0250

1000:0000

- **Segment:Offsets→ Segment address = 1000h**

  → Offset address = 0000h

3. Segmented Memory Management

- **Segmented Memory**
  - ❑ **Addressing scheme:** How memory address is referred?
    - ❖ Absolute address (physical address)
      - ✓ Uses a 20 bit value that directly references a specific memory location
    - ❖ Segment:offset address (logical address)
      - ✓ Combines the starting address of a segment with an offset value

3. Segmented Memory Management

- **Segmented Memory**
  - ❑ **Segment address** is stored un segment register without last digit.
  - ❑ E.g.: 038E0H → 038EH

    (absolute address) → Segment address
  - ❑ Effectively, the 20-bit address is stored n the 16-bit segment register
  - ❑ **Offset address** is the distance in bytes from the segment address to another location within the segment.
  - ❑ Offset address ranges from 0000H (0_{10}) to FFFFH ($65{,}535_{10}$)
  - ❑ Each segment can be up to 64KB in size

3. Segmented Memory Management

- **20-bit Linier Address Calculation**
  - ❑ To obtain actual / absolute address of memory location from segment:offset address, the processors involves:

    Step 1: Convert 16-bit segment address into 20-bit address

    Step 2: Add the offset address
  - ❑ Absolute address = (16-bit segment address x 10H) + Offset address
  - ❑ E.g.: Given segment and offset address at **08F1:0100**

    Step 1: 08F1H x 10H = 08F10H

    Step 2: $\underline{+\ \ 0100H}$

    09010H

3. Segmented Memory Management

- **Segment**
  - Memory segment are the special areas of memory containing code, data and stack information.
  - The operating system keep tracks of the locations of individual program segment based on segments.
  - There are 3 key types of memory segment:
    - Code segment : Hold machine instructions
    - Data segment : Hold programs' defined data & constants
    - Stack segment : Hold local function variables & parameters

3. Segmented Memory Management

- **Segment**
  - ❑ To initialize data segment register, there are 2 steps

```
MOV AX, @DATA
MOV DS, AX
```

  - ❑ (Note: The immediate value cannot be moved directly into segment register)

4. Program Execution Registers

4. Program Execution Registers

- **Registers**
  - ❑ Registers are defined as high speed storage located inside the CPU.
  - ❑ It is used to store data 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

- **Registers**
  - ❑ Categories of registers

Categories	A.k.a	Functions	Bits	registers
General purpose register	Data registers	Handle data movement & arithmetic computation	16	AX, BX, CX, DX
			8	AH, AL, BH, BL, CH, CL, DH, DL
Address registers	Segment:offset registers	Handle addressing	16	CS, DS, ES, SS
				IP, BP, SP, SI, DI
Status registers	Flag register	Indicate computer status	1	OF, DF, IF, TF, SF, ZF, AF, PF, CF

4. Program Execution Registers

- **General Purpose Registers**

Registers		Description
AX	Accumulator register	• Used for operations involving input / output & most arithmetic
BX	Base register	• Used as an index to extend addressing & computation • Also used as DI & SI as a base register for special addressing
CX	Count register	• Used to control the no. of times a LOOP instruction is repeated • Also support computations
DX	Data register	• Input / output operations • Multiplication & division operations that involve large value

4. Program Execution Registers

- **Address Registers – Segment Registers**

Registers		Description
CS	Code segment register	• Hold the start address of code segment
DS	Data segment register	• Hold the start address of data segment
ES	Extra segment register	• Hold the start address of extra segment
SS	Stack segment register	• Hold the start address of stack segment

4. Program Execution Registers

- **Address Registers – Offset Registers**

Registers		Description
SI	Source index	• Support string (character) handling operations. • Associated with DS register
DI	Destination index	• Support string (character) handling operations. • Associated with ES register
IP	Instruction pointer	• Holds the address of next instruction that is to execute • Associated with CS register
BP	Base pointer	• Support parameter referencing via the stack • Associated with SS register
SP	Stack pointer	• Holds the address of current word being processed in the stack. • Associated with SS register

4. Program Execution Registers

- **Address Registers – Segment:Offset Registers**

Registers		Description
CS	IP	• Provides the address of instruction to be fetched for execution
DS	SI	• Provides the reference to a specific byte location in the data segment
ES	DI	• Used by some string operations to handle memory addressing
SS	SP	• Provides the current words in the stack being addressed

4. Program Execution Registers

- **Address Registers – Status Registers**

Registers		Description
OF	Overflow flag	• Indicates overflow of msb after an arithmetic operations. • Set when the result of a signed arithmetic operation is too large / too small to fit into the destination
DF	Direction flag	• Determines left / right direction for moving / comparing string data
IF	Interrupt flag	• Indicates that all external interrupts, are to be processed / ignored
TF	Trap frag	• Used for single stepping through a program
SF	Sign flag	• Indicates arithmetic sign of the result after an arithmetic operation

4. Program Execution Registers

- **Address Registers – Status Registers**

Registers		Description
ZF	Zero flag	• Indicates that the result of an arithmetic / logical operation is zero
AF	Auxiliary flag	• Set when an arithmetic operation causes a carry from bit3 to bit4 in an 8-bit operand
PF	Parity flag	• Support error checking • Set when the result contain an even number of 1 bit
CF	Carry flag	• Indicates a carry after an arithmetic operations. • Set when the result of an unsigned arithmetic operation is too large to fit into the destination

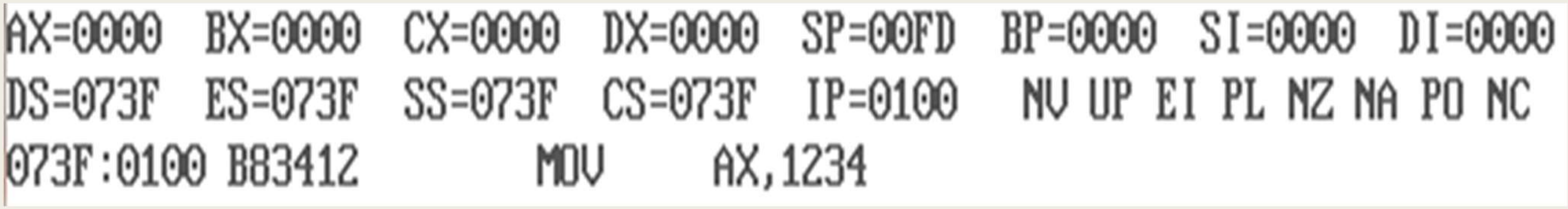
4. Program Execution Registers

- **Address Registers – Status Registers**

Flag name	SET (1)	Clear (0)
OF	OV	NV
DF	DN	UP
IF	EI	DI
SF	NG	PL
ZF	ZE	NZ
AF	AC	NA
PF	PE	PO
CF	CY	NC

4. Program Execution Registers

- **Registers**



```
AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 B83412        MOV     AX,1234
```

Categories	Bits	registers
General purpose register	16	AX, BX, CX, DX
	8	AH, AL, BH, BL, CH, CL, DH, DL
Address registers	16	CS, DS, ES, SS
		IP, BP, SP, SI, DI
Status registers	1	OF, DF, IF, TF, SF, ZF, AF, PF, CF

Chapter Review

Chapter Review

1. Data Storage Sizes

- ❑ Byte
- ❑ Word
- ❑ Doubleword
- ❑ Quadword
- ❑ Paragraph

2. Data Addressing

- ❑ Absolute address
- ❑ Segment offset address

3. Segmented Memory Management

- ❑ Segment
- ❑ Offset

4. Program Execution Registers

- ❑ General purpose registers
- ❑ Address registers
- ❑ Status registers