

8086

MICROPROCESSOR

Prepared By

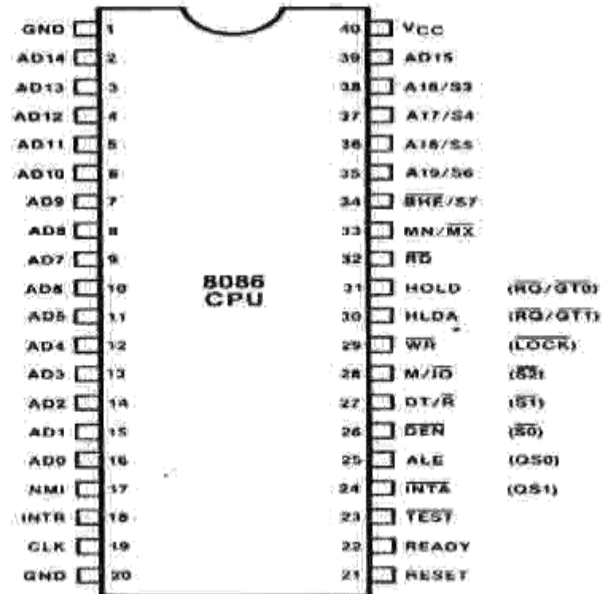
Dr. D. Jayakumar, M.Tech., Ph.D.,

Associate Professor, Dept Of ECE

Kuppam Engineering College, Kuppam

8086 Microprocessor

Common Signals		
Name	Function	Type
AD15-AD0	Address/Data Bus	Bidirectional, 3-State
A19/S6-A16/S3	Address/Status	Output, 3-State
BHE/S7	Bus High Enable/Status	Output, 3-State
MN/MX	Minimum/Maximum Mode Control	Input
RD	Read Control	Output, 3-State
TEST	Wait On Test Control	Input
READY	Wait State Control	Input
RESET	System Reset	Input
NMI	Non-Maskable Interrupt Request	Input
INTR	Interrupt Request	Input
CLK	System Clock	Input
VCC	+5V	Input
GND	Ground	
Minimum Mode Signals (MN/MX = VCC)		
Name	Function	Type
HOLD	Hold Request	Input
HLDA	Hold Acknowledge	Output
WR	Write Control	Output, 3-State
M/IO	Memory/IO Control	Output, 3-State
DT/R	Data Transmit/Receive	Output, 3-State
EN	Data Enable	Output, 3-State
ALE	Address Latch Enable	Output
INTA	Interrupt Acknowledge	Output
Maximum Mode Signals (MN/MX = GND)		
Name	Function	Type
RD/GT1, 0	Request/Grant Bus Access Control	Bidirectional
LOCK	Bus Priority Lock Control	Output, 3-State
S2-S0	Bus Cycle Status	Output, 3-State
QS1, QS0	Instruction Queue Status	Output

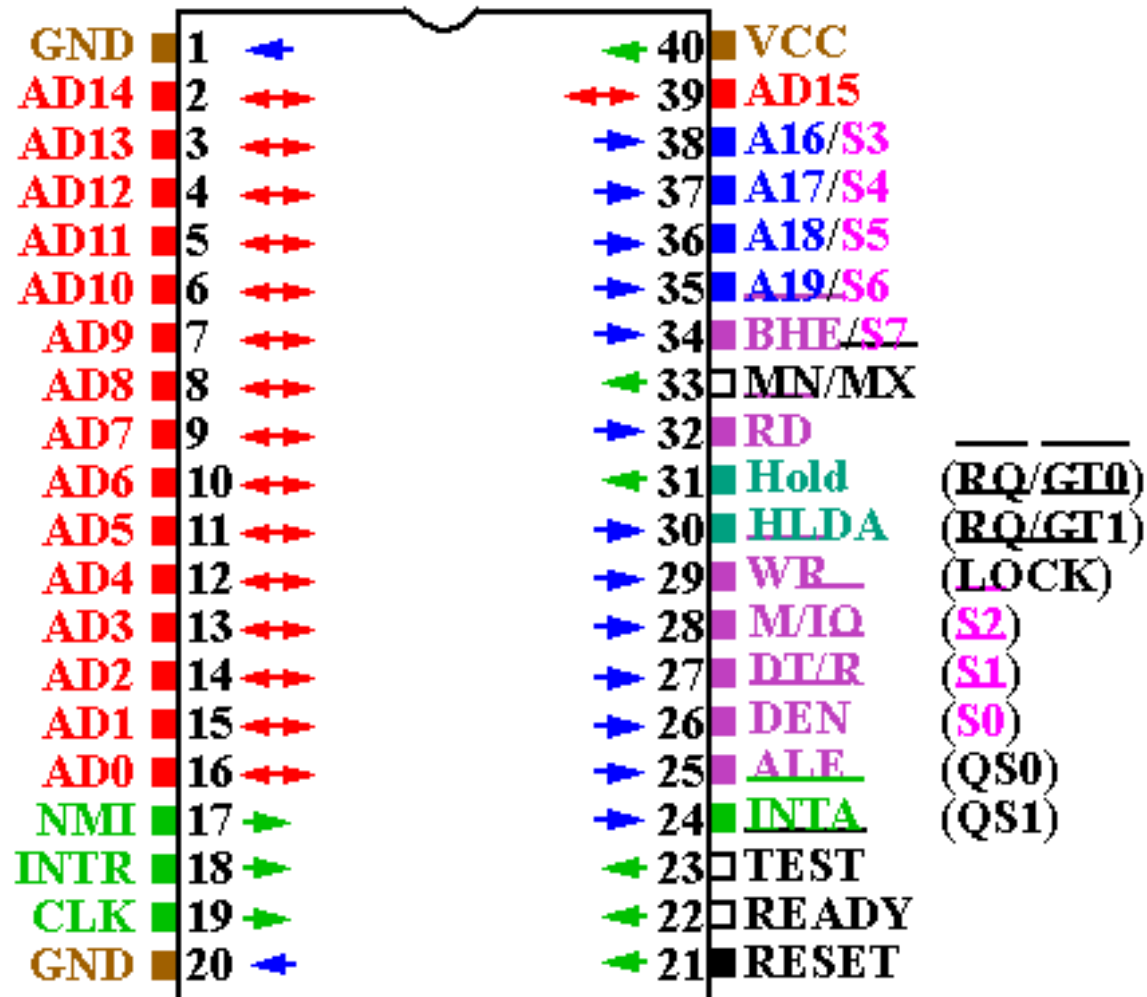


MAXIMUM MODE PIN FUNCTIONS (e.g., LOCK) ARE SHOWN IN PARENTHESES

Figure 4-1. 8086 Pin Definitions

8086 CPU

MIN MODE (MAX MODE)



8086 Features

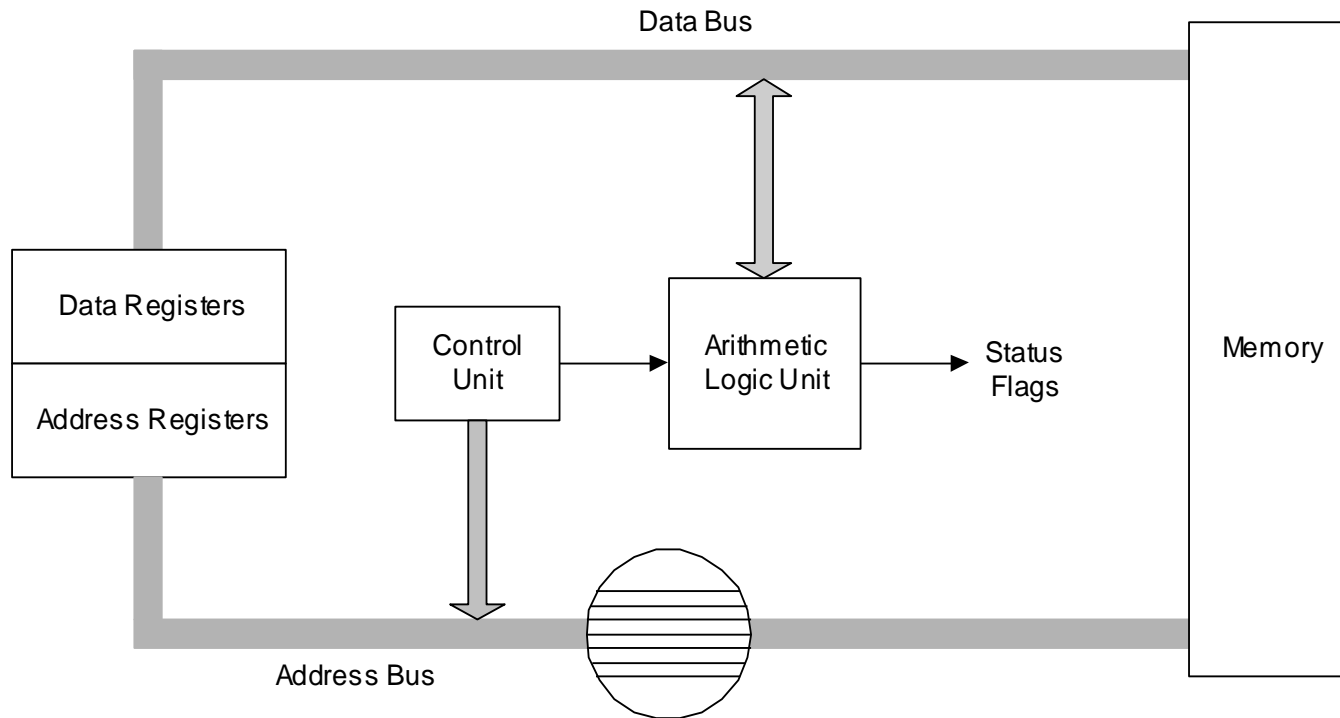
- **16-bit Arithmetic Logic Unit**
- **16-bit data bus (8088 has 8-bit data bus)**
- **20-bit address bus - $2^{20} = 1,048,576 = 1 \text{ meg}$**

The address refers to a byte in memory. In the 8088, these bytes come in on the 8-bit data bus. In the 8086, bytes at even addresses come in on the low half of the data bus (bits 0-7) and bytes at odd addresses come in on the upper half of the data bus (bits 8-15).

The 8086 can read a 16-bit word at an even address in one operation and at an odd address in two operations. The 8088 needs two operations in either case.

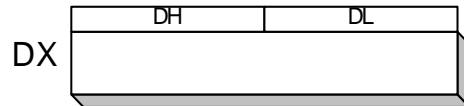
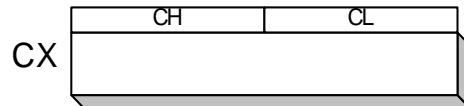
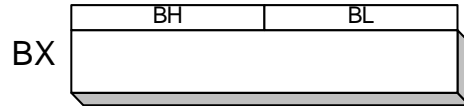
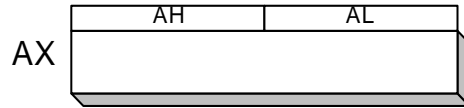
The least significant byte of a word on an 8086 family microprocessor is at the lower address.

Simplified CPU Design



Intel 16-bit Registers

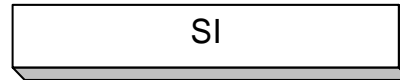
General Purpose



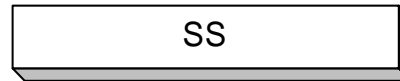
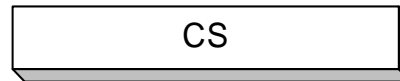
Status and Control



Index



Segment



8086 Architecture

- The 8086 has two parts, the Bus Interface Unit (BIU) and the Execution Unit (EU).
- The BIU fetches instructions, reads and writes data, and computes the 20-bit address.
- The EU decodes and executes the instructions using the 16-bit ALU.
- The BIU contains the following registers:

IP - the Instruction Pointer

CS - the Code Segment Register

DS - the Data Segment Register

SS - the Stack Segment Register

ES - the Extra Segment Register

The BIU fetches instructions using the CS and IP, written CS:IP, to construct the 20-bit address. Data is fetched using a segment register (usually the DS) and an effective address (EA) computed by the EU depending on the addressing mode.

The EU contains the following 16-bit registers:

AX - the Accumulator

BX - the Base Register

CX - the Count Register

DX - the Data Register

SP - the Stack Pointer \ defaults to stack segment

BP - the Base Pointer /

SI - the Source Index Register

DI - the Destination Register

These are referred to as general-purpose registers, although, as seen by their names, they often have a special-purpose use for some instructions.

The AX, BX, CX, and DX registers can be considered as two 8-bit registers, a High byte and a Low byte. This allows byte operations and compatibility with the previous generation of 8-bit processors, the 8080 and 8085. 8085 source code could be translated in 8086 code and assembled. The 8-bit registers are:

AX --> AH,AL

BX --> BH,BL

CX --> CH,CL

DX --> DH,DL

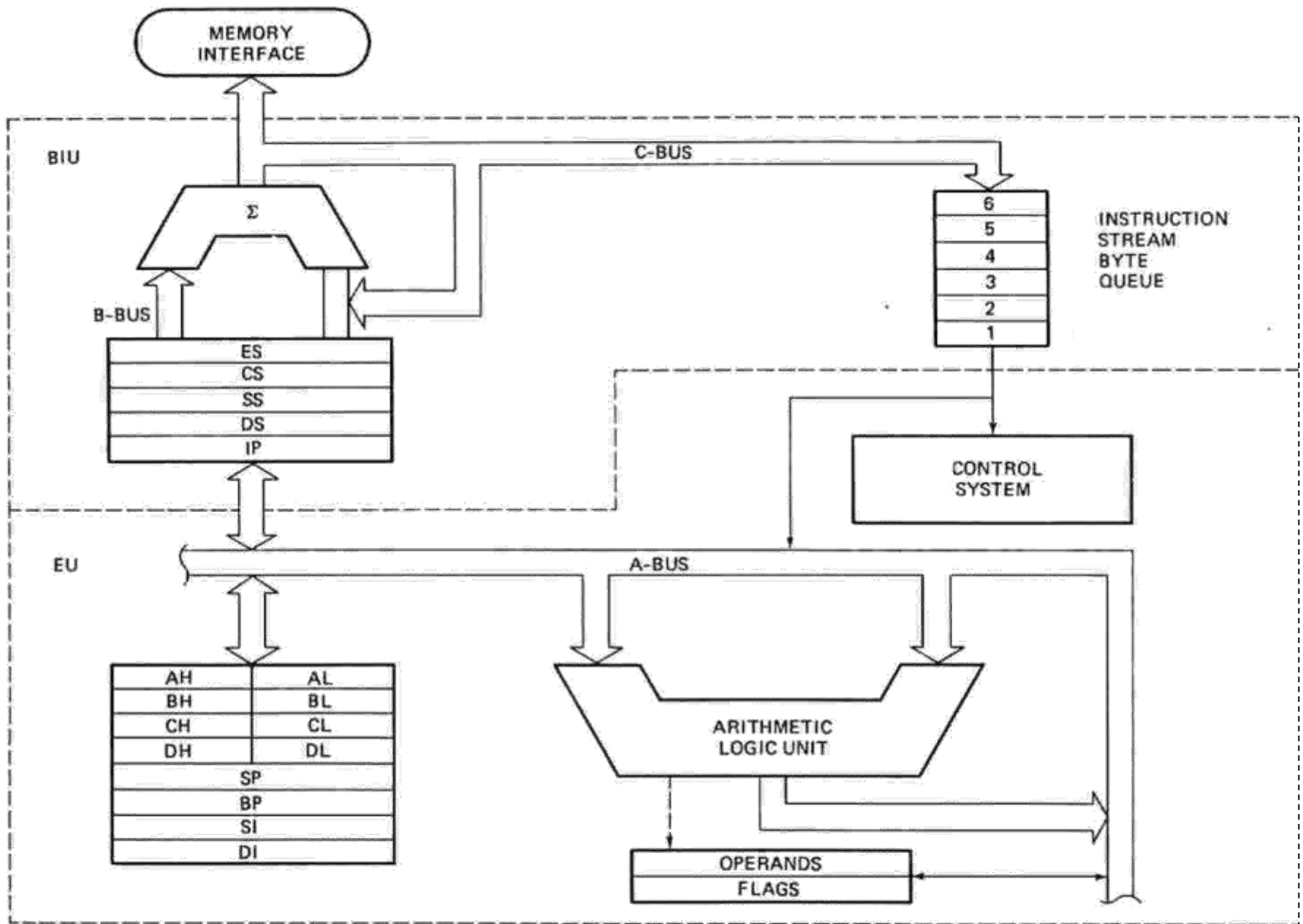


FIGURE 2-7 8086 internal block diagram. (Intel Corp.)

Registers

<u>Class</u>	<u>Register</u>	<u>Purpose</u>
Data:	AX,BX	“general” purpose
	CX	string and loop ops only
	DX	mult/div and I/O only
Address:	SP	stack pointer
	BP	base pointer (can also use BX)
	SI,DI	index registers
Segment:	CS	code segment
	SS	stack segment
	DS	data segment
	ES	extra segment
Control:	IP	instruction pointer (lower 16 bit of PC)
	FLAGS	C, Z, N, B, P, V and 3 control bits

8086 Programmer's Model

BIU registers
(20 bit adder)

ES
CS
SS
DS
IP

Extra Segment
Code Segment
Stack Segment
Data Segment
Instruction Pointer

EU registers

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL
	SP	
	BP	
	SI	
	DI	
	FLAGS	

Accumulator
Base Register
Count Register
Data Register
Stack Pointer
Base Pointer
Source Index Register
Destination Index Register

8086/88 internal registers 16 bits (2 bytes each)

AX	AH	AL	Accumulator	} Data group
BX	BH	BL	Base	
CX	CH	CL	Count	
DX	DH	DL	Data	

AX, BX, CX and DX are two bytes wide and each byte can be accessed separately

SP	Stack pointer	} Pointer and index group
BP	Base pointer	
SI	Source index	
DI	Destination index	
IP	Instruction pointer	

These registers are used as memory pointers.

Flags _H	Flags _L	Status and control flags
--------------------	--------------------	--------------------------

Flags will be discussed later

ES	Extra	} Segment group
CS	Code	
DS	Data	
SS	Stack	

Segment registers are used as base address for a segment in the 1 M byte of memory

The 8086/8088 Microprocessors: Registers

- Registers

- Registers are in the CPU and are referred to by specific names
- Data registers
 - Hold data for an operation to be performed
 - There are 4 data registers (AX, BX, CX, DX)
- Address registers
 - Hold the address of an instruction or data element
 - Segment registers (CS, DS, ES, SS)
 - Pointer registers (SP, BP, IP)
 - Index registers (SI, DI)
- Status register
 - Keeps the current status of the processor
 - On an IBM PC the status register is called the FLAGS register
- In total there are fourteen 16-bit registers in an 8086/8088

Data Registers: AX, BX, CX, DX

- Instructions execute faster if the data is in a register
- AX, BX, CX, DX are the data registers
- Low and High bytes of the data registers can be accessed separately
 - AH, BH, CH, DH are the high bytes
 - AL, BL, CL, and DL are the low bytes
- Data Registers are general purpose registers but they also perform special functions
- AX
 - Accumulator Register
 - Preferred register to use in arithmetic, logic and data transfer instructions because it generates the shortest Machine Language Code
 - Must be used in multiplication and division operations
 - Must also be used in I/O operations

- **BX**
 - Base Register
 - Also serves as an address register
- **CX**
 - Count register
 - Used as a loop counter
 - Used in shift and rotate operations
- **DX**
 - Data register
 - Used in multiplication and division
 - Also used in I/O operations

Pointer and Index Registers

- Contain the offset addresses of memory locations
- Can also be used in arithmetic and other operations
- **SP: Stack pointer**
 - Used with SS to access the stack segment
- **BP: Base Pointer**
 - Primarily used to access data on the stack
 - Can be used to access data in other segments
- **SI: Source Index register**
 - is required for some string operations
 - When string operations are performed, the SI register points to memory locations in the data segment which is addressed by the DS register. Thus, SI is associated with the DS in string operations.

- **DI: Destination Index register**
 - is also required for some string operations.
 - When string operations are performed, the DI register points to memory locations in the data segment which is addressed by the ES register. Thus, DI is associated with the ES in string operations.
- The SI and the DI registers may also be used to access data stored in arrays

Segment Registers - CS, DS, SS and ES

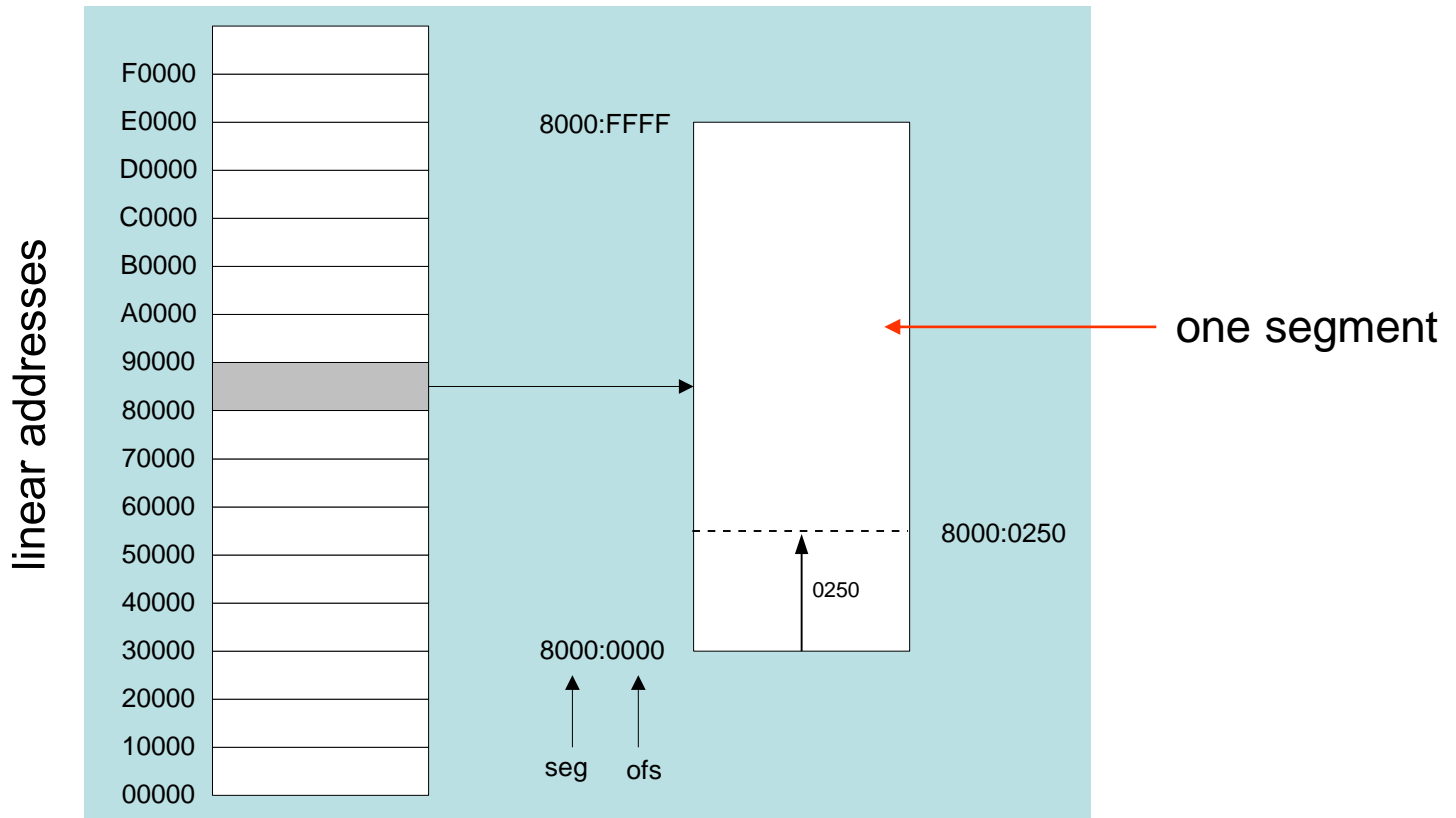
- Are Address registers
- Store the memory addresses of instructions and data
- **Memory Organization**
 - Each byte in memory has a 20 bit address starting with 0 to $2^{20}-1$ or 1 meg of addressable memory
 - Addresses are expressed as 5 hex digits from 00000 - FFFFF
 - Problem: But 20 bit addresses are **TOO BIG** to fit in 16 bit registers!
 - Solution: Memory Segment
 - Block of 64K (65,536) consecutive memory bytes
 - A segment number is a 16 bit number
 - Segment numbers range from 0000 to FFFF
 - Within a segment, a particular memory location is specified with an offset
 - An offset also ranges from 0000 to FFFF

Segmented Memory Architecture (*real mode*)

- A **segment** addresses 64K of memory
- A **segment register** contains the starting location of a segment
 - the absolute location of a segment can be obtained by appending a hexadecimal zero
- An **offset** is the distance from the beginning of a segment to a particular instruction or variable

Segmented Memory

Segmented memory addressing: absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset

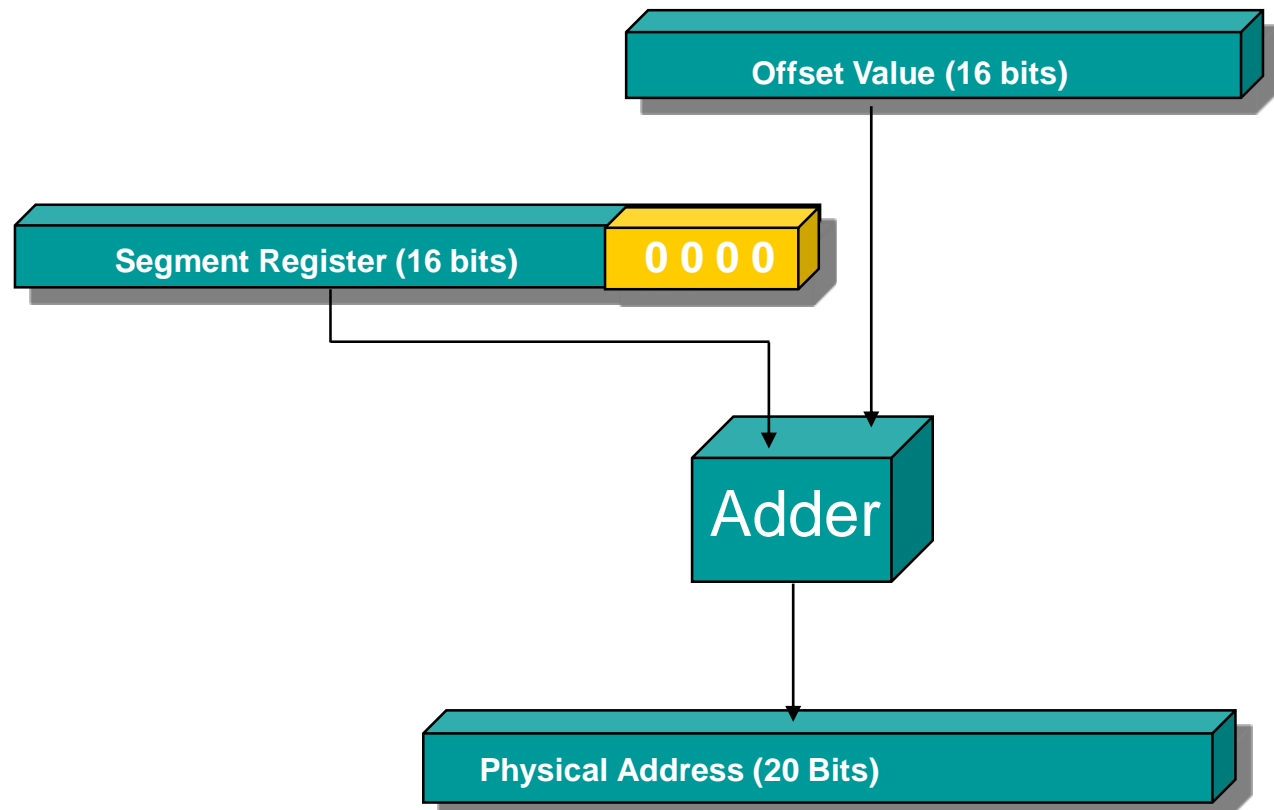


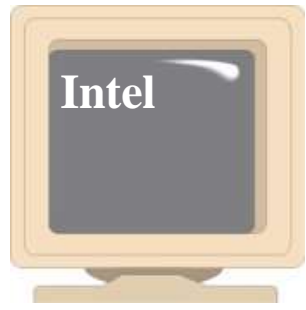


Intel

Memory Address Generation

- The BIU has a dedicated adder for determining physical memory addresses

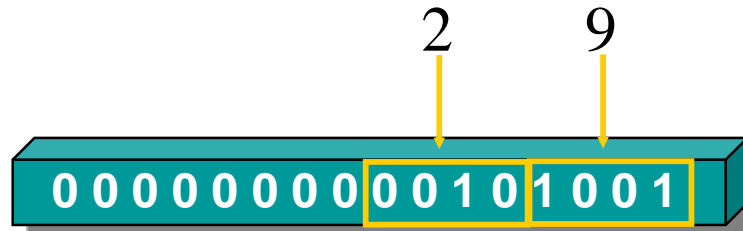




Example Address Calculation

- If the data segment starts at location 1000h and a data reference contains the address 29h where is the actual data?

Offset:



Segment:



Address:



Segment:Offset Address

- Logical Address is specified as `segment:offset`
- Physical address is obtained by shifting the segment address 4 bits to the left and adding the offset address
- Thus the physical address of the logical address A4FB:4872 is

$$\begin{array}{r} A4FB0 \\ + \underline{4872} \\ \hline A9822 \end{array}$$

Example

•If DS=7FA2H and the offset is 438EH

a) Calculate the physical address

$$7FA20 + 438E = 83DAE$$

b) calculate the lower range

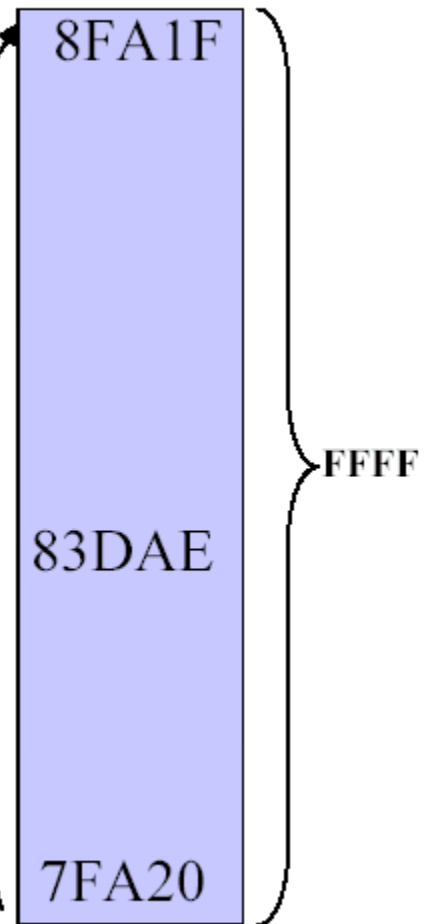
$$7FA20 + 0000 = 7FA20$$

c) Calculate the upper range of the data segment

$$7FA20 + FFFF = 8FA1F$$

d) Show the logical Address

7FA2:438E



Your turn . . .

What linear address corresponds to the segment/offset address 028F:0030?

$$028F0 + 0030 = 02920$$

Always use hexadecimal notation for addresses.

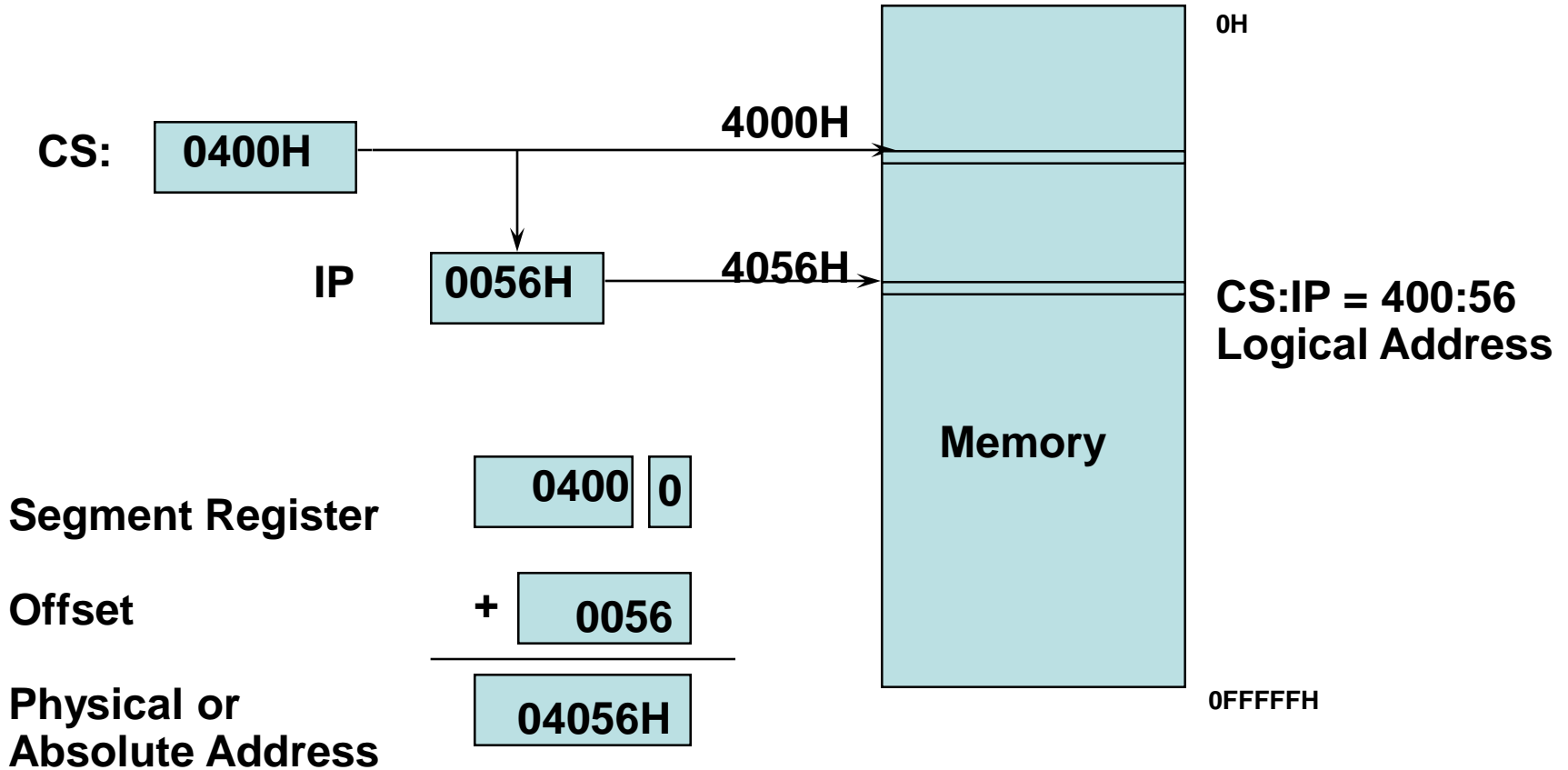
Your turn . . .

What segment addresses correspond to the linear address 28F30h?

Many different segment-offset addresses can produce the linear address 28F30h. For example:

28F0:0030, 28F3:0000, 28B0:0430, . . .

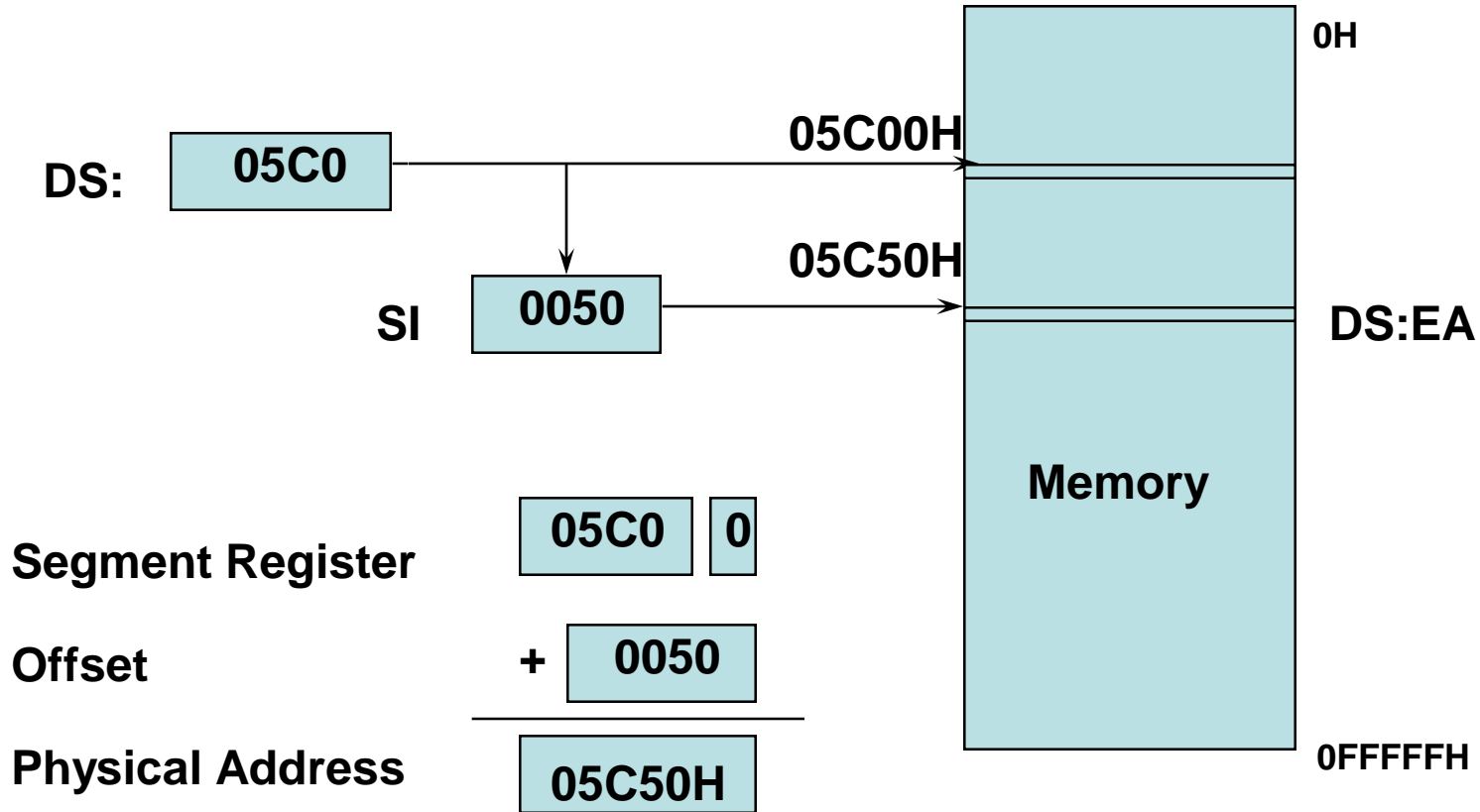
The Code Segment



The offset is the distance in bytes from the start of the segment. The offset is given by the IP for the Code Segment. Instructions are always fetched with using the CS register.

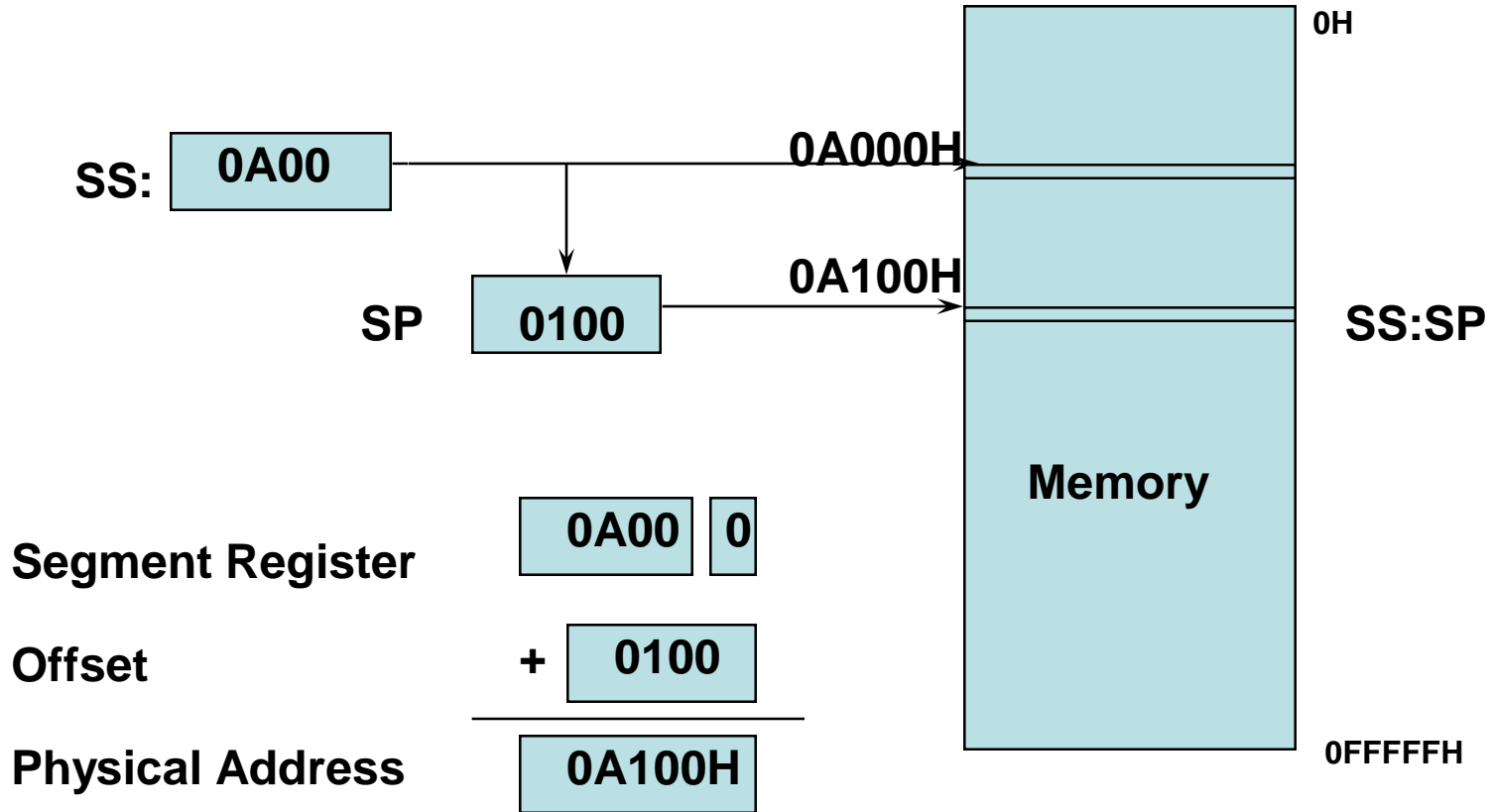
The physical address is also called the absolute address.

The Data Segment



Data is usually fetched with respect to the DS register.
The effective address (EA) is the offset.
The EA depends on the addressing mode.

The Stack Segment



The offset is given by the SP register.
The stack is always referenced with respect to the stack segment register.
The stack grows toward decreasing memory locations.
The SP points to the last or top item on the stack.

PUSH - pre-decrement the SP
POP - post-increment the SP

Flags



*Bits marked X are undefined.

Overflow

Direction

Interrupt enable

Trap

Sign

Zero

Auxiliary flag

Parity flag

Carry flag

6 are status flags
3 are control flag

Flag Register

- Conditional flags:
 - They are set according to some results of arithmetic operation. You do not need to alter the value yourself.
- Control flags:
 - Used to control some operations of the MPU. These flags are to be set by you in order to achieve some specific purposes.

Flag					O	D	I	T	S	Z		A		P		C
Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- CF (carry) Contains carry from leftmost bit following arithmetic, also contains last bit from a shift or rotate operation.

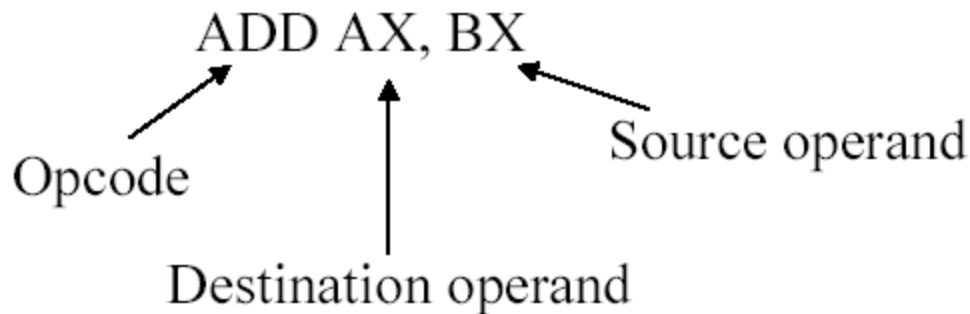
Flag Register

- OF (overflow) Indicates overflow of the leftmost bit during arithmetic.
- DF (direction) Indicates left or right for moving or comparing string data.
- IF (interrupt) Indicates whether external interrupts are being processed or ignored.
- TF (trap) Permits operation of the processor in single step mode.

- SF (sign) Contains the resulting sign of an arithmetic operation (1=negative)
- ZF (zero) Indicates when the result of arithmetic or a comparison is zero. (1=yes)
- AF (auxiliary carry) Contains carry out of bit 3 into bit 4 for specialized arithmetic.
- PF (parity) Indicates the number of 1 bits that result from an operation.

Software


- The sequence of commands used to tell a microcomputer what to do is called a program,
- Each command in a program is called an instruction
- 8088 understands and performs operations for 117 basic instructions
- The native language of the IBM PC is the machine language of the 8088
- A program written in machine language is referred to as machine code
- In 8088 assembly language, each of the operations is described by alphanumeric symbols instead of just 0s or 1s.




Instructions

LABEL: INSTRUCTION ; COMMENT

Address identifier



Does not generate any machine code



Ex. START: MOV AX,BX ; copy BX into AX

• Addressing modes

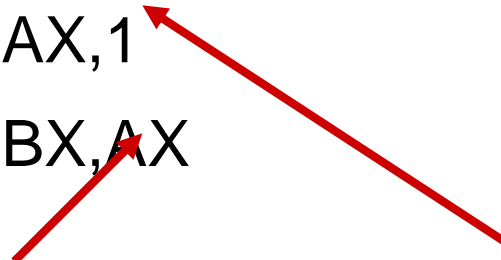
- **Register** and **immediate** modes we have already seen

MOV AX,1

MOV BX,AX

register

immediate



3F03 - 80x86 assembler

- Typical addressing modes
 - Absolute address mode

MOV AX,[0200]



value stored in memory location DS:0200

3F03 - 80x86 assembler

- Typical addressing modes
 - Register indirect

MOV AX,[BX]



value stored at address contained in DS:BX

3F03 - 80x86 assembler

- Typical addressing modes
 - Displacement

```
MOV DI,4
```

```
MOV AX,[0200+DI]
```

value stored at DS:0204



3F03 - 80x86 assembler

- Typical addressing modes
 - Indexed

```
MOV BX,0200
```

```
MOV DI,4
```

```
MOV AX,[BX+DI]
```

value stored at DS:0204



3F03 - 80x86 assembler

- Typical addressing modes
 - Memory indirect

```
MOV DI,0204
```

```
MOV BX,[DI]
```

```
MOV AX,[BX]
```

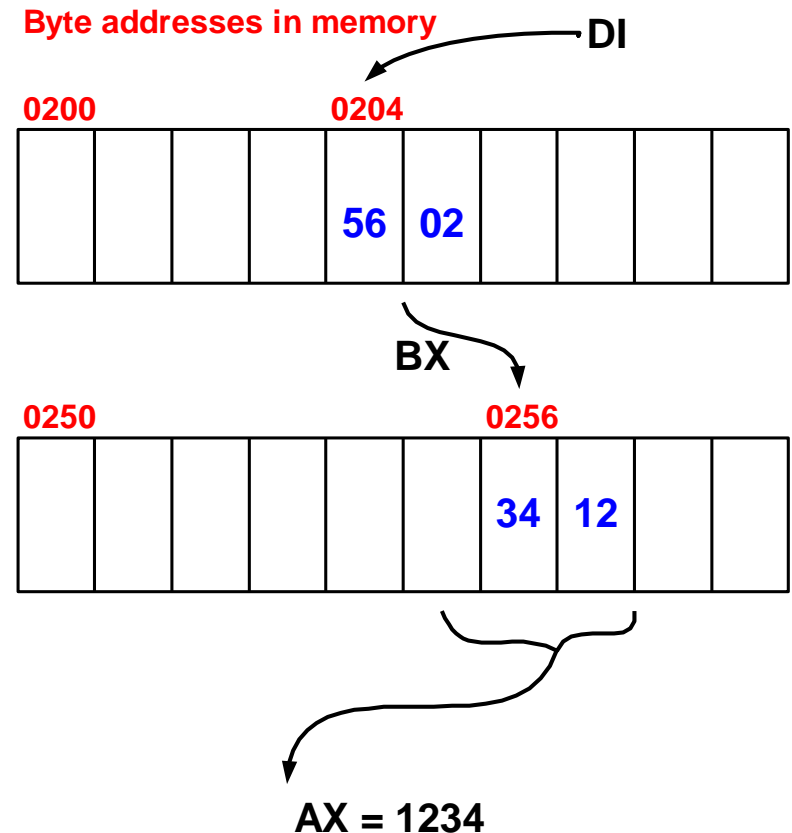
If DS:0204 contains 0256,
then AX will contain
whatever is stored at
DS:0256

3F03 - 80x86 assembler

- Typical addressing modes
 - Memory indirect

```
MOV DI,0204
MOV BX,[DI]
MOV AX,[BX]
```

If DS:0204 contains 0256,
then AX will contain
whatever is stored at
DS:0256



8086 in Maximum Mode

The IBM PC is a maximum mode 8088 system. When an 8086/8088 is used in the maximum mode (MN/MX pin grounded) it requires the use of an 8288 Bus Controller. The system can support multiple processors on the system bus by the use of an 8289 Bus Arbiter.

The following signals now come from the 8288: ALE, DT/R', DEN, and INTA'.

The M/IO', RD', and WR' signals are replaced by:

MRDC' - memory read command

MWTC' - memory write command

IORC' - I/O read command

IOWC' - I/O write command

AMWC' - Advanced memory write command

AIOWC' - Advanced I/O write command

The advanced commands become active earlier in the cycle to give devices an earlier indication of a write operation.

8086 Maximum Mode

When in the maximum mode, the 8086/8088 has 3 status lines that are connected to the 8288 and provide it with the information it needs to generate the system bus signals. The information provided by the status bits is as follows.

S2'	S1'	S0'	<u>operation</u>	<u>signal</u>
0	0	0	Interrupt Acknowledge	INTA'
0	0	1	Read I/O port	IORC'
0	1	0	Write I/O port	IOWC', AIOWC'
0	1	1	Halt	none
1	0	0	Instruction Fetch	MRDC'
1	0	1	Read Memory	MRDC'
1	1	0	Write Memory	MWTC', AMWC'
1	1	1	Passive	none

Direct Memory Access - DMA

DMA allows data to go between memory and a peripheral, such as a disk drive, without going through the cpu.

The DMA controller takes over the address bus, data bus, and control bus. The 8237A DMA Controller is a commonly used device and is in the IBM PC.

Figure 11-4 is a simplified block diagram showing the use of a DMA controller. For example, to read a disk file the following operations occur.

1. Send a series of commands to the disk controller to find and read a block of data.
2. When the controller has the first byte of the block, it sends a DMA request DREQ to the DMA controller.
3. If that input of the DMA controller is unmasked, the DMA controller sends a hold-request HQR to the cpu.
4. The cpu responds with a hold-acknowledge HLDA and floats its buses.
5. The DMA controller then takes control of the buses.
6. The DMA controller sends out the memory address and DMA acknowledge DACK0 to the disk controller.
7. The DMA controller asserts the MEMW' and IOR' lines.

Memory

Terminology

Volatile - data is lost when power is turned off.

Nonvolatile - retains data when powered off.

Random Access - all data locations accessed in the same amount of time.

Sequential Access - data accessed in varying amounts of time, e.g., tape.

ROM - Read Only Memory.

RAM - Random Access Memory

By convention, RAM in a PC is really Read/Write Memory and ROM (EPROM) in a PC, although random access memory, is not referred to as RAM.

Examples

VOLATILE

Static RAM

Dynamic RAM

NONVOLATILE

ROM, PROM, EPROM, EEPROM, FLASH

Disk, tape

Magnetic core, magnetic bubble

Interface 8086 to 6116 static RAM

8086

