Microprocessors & Interfacing

INSTRUCTION SET

BITS Pilani

Dr. Gargi Prabhu
Department of CS & IS

STOS

- Stores AL, AX, or EAX at the extra segment memory location addressed by the DI register.

E.g.

STOSB	ES:[DI] = AL; DI = DI ± 1
STOSW	ES:[DI] = AX; DI = DI ± 2

- The STOSB (stores a byte) instruction stores the byte in AL at the extra segment memory location addressed by DI.
- The STOSW (stores a word) instruction stores AX in the extra segment memory location addressed by DI.

Example

MOV AL, 10

MOV DI, 2010

STOB

```
-d 2010
0859:2010  10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:2020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:2030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:2040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
```

Example

- Suppose that the STOSW instruction is used to clear an area of memory called Buffer using a count called Count and the program is to function call Clear Buffer in the environment using the inline assembler.

```
void ClearBuffer (int count, short* buffer)
{
    _asm{
          push edi              ;save registers
          push es
          push ds
          mov  ax,0
          mov  ecx, count
          mov  edi, buffer
          pop  es               ;load ES with DS
          rep  stosw            ;clear Buffer
          pop  es               ;restore registers
          pop  edi
    }
}
```

The repeat prefix (REP) is added to any string data transfer instruction, except the LODS instruction. The REP prefix causes CX to decrement by 1 each time the string instruction executes. After CX decrements, the string instruction repeats. If CX reaches a value of 0, the instruction terminates and the program continues with the next sequential instruction.

ADD Instruction

- Adding two operands and storing the result in the destination operand.

Syntax: ADD destination, source

E.g.

ADD AL,BL	AL = AL + BL
ADD CX,DI	CX = CX + DI
ADD EBP,EAX	EBP = EBP + EAX
ADD CL,44H	CL = CL + 44H
ADD BX,245FH	BX = BX + 245FH
ADD EDX,12345H	EDX = EDX + 12345H

ADD Instruction

E.g.

ADD [BX],AL	AL adds to the byte contents of the data segment memory location addressed by BX with the sum stored in the same memory location
ADD CL,[BP]	The byte contents of the stack segment memory location addressed by BP add to CL with the sum stored in CL
ADD AL,[EBX]	The byte contents of the data segment memory location addressed by EBX add to AL with the sum stored in AL
ADD BX,[SI+2]	The word contents of the data segment memory location addressed by SI + 2 add to BX with the sum stored in BX
ADD CL,TEMP	The byte contents of data segment memory location TEMP add to CL with the sum stored in CL
ADD BX,TEMP[DI]	The word contents of the data segment memory location addressed by TEMP + DI add to BX with the sum stored in BX
ADD [BX+D],DL	DL adds to the byte contents of the data segment memory location addressed by BX + DI with the sum stored in the same memory location
ADD BYTE PTR [DI],3	A 3 adds to the byte contents of the data segment memory location addressed by DI with the sum stored in the same location

Result of Arithmetic Operation

- Whenever arithmetic and logic instructions execute, the contents of the flag register change.
- Contents of the interrupt, trap, and other flags do not change due to arithmetic and logic instructions.
- Only the flags located in the rightmost 8 bits of the flag register and the overflow flag change.

E.g.

MOV DL,12H

ADD DL,33H

Z = 0 (result not zero)

C = 0 (no carry)

A = 0 (no half-carry)

S = 0 (result positive)

P = 0 (odd parity)

O = 0 (no overflow)

Example of Conditional Flag Registers

```
 0011 0100 1101 1100
+0000 0111 0010 1110
 0011 1100 0000 1010

CF = 0
PF = 1
AF = 1
ZF = 0
SF = 0
OF = 0
```

- **Carry flag (CF)-** carry out of MSB
- **Parity flag (PF) -**set to 1 if low-order 8 bits (low order byte) contain even number of 1's
- **Auxiliary carry flag (AF)** - carry out of bit 3
- **Zero flag (ZF)** - set to 1 if result is 0; set to 0 if result is nonzero
- **Sign flag (SF)** - MSB of result
- **Overflow flag (OF)** - set if carry in to MSB is not equal to carry out from MSB)

Memory-to-Register Addition

- An application requires memory data to be added to the AL register

E.g.

```
MOV DI,OFFSET NUMB          ;address NUMB
MOV AL,0                    ;clear sum
ADD AL,[DI]                 ;add NUMB
ADD AL,[DI+1]               ;add NUMB+1
```

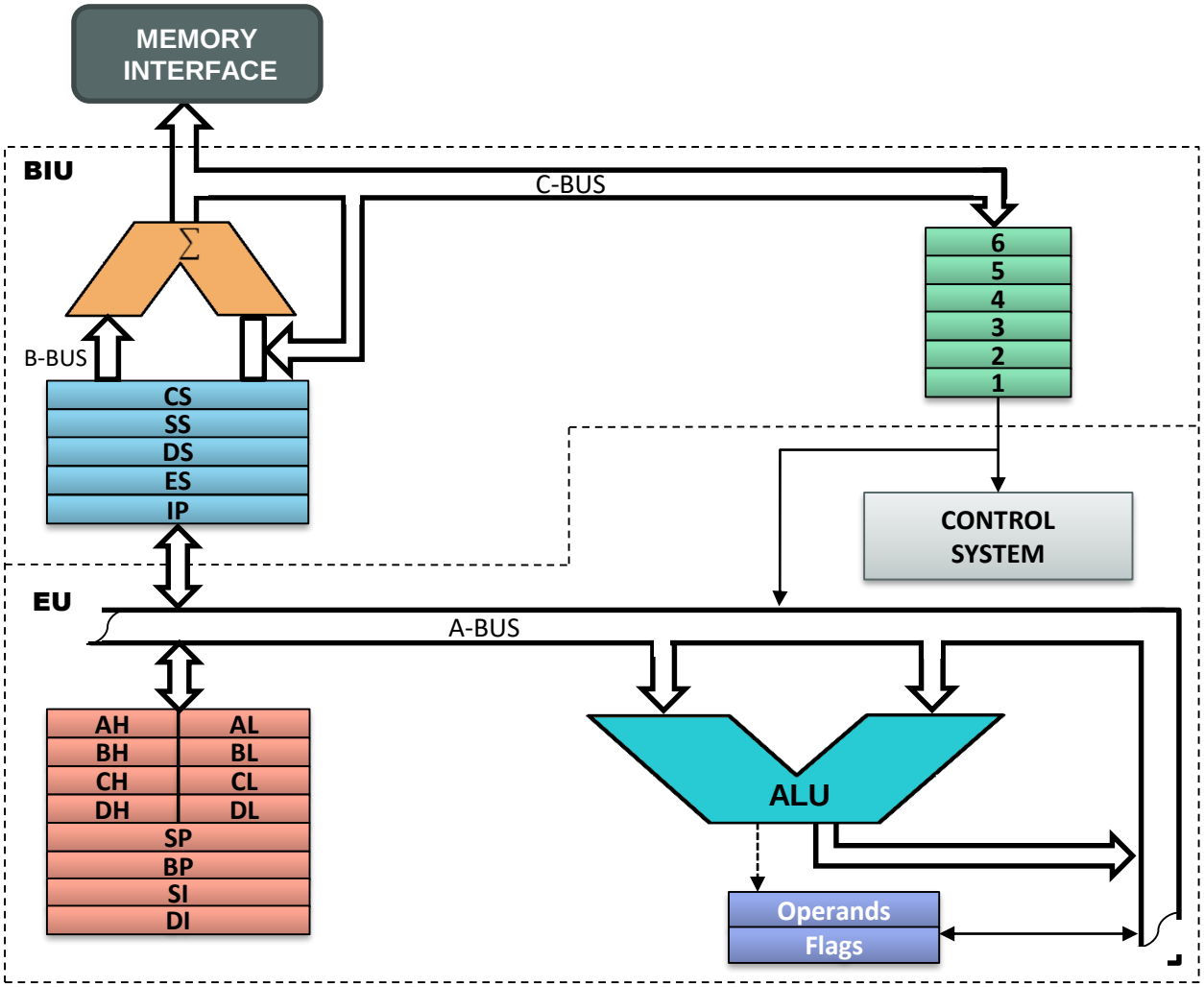
Array Addition

- Memory arrays are sequential lists of data.
- Suppose that an array of data (ARRAY) contains 10 bytes, numbered from element 0 through element 9.

E.g.

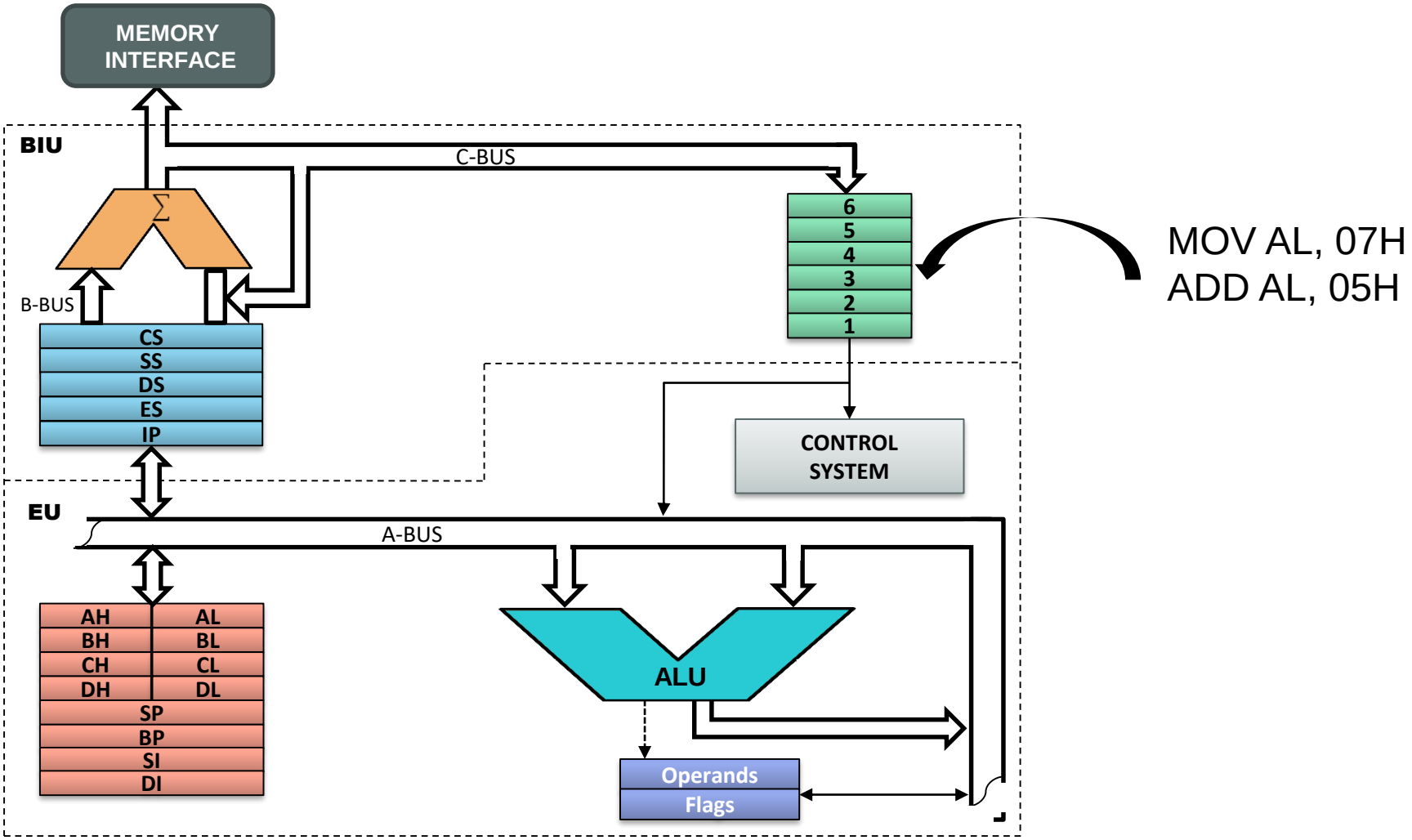
```
MOV AL,0                ;clear sum
MOV SI,3                ;address element 3
ADD AL,ARRAY[SI]        ;add element 3
ADD AL,ARRAY[SI+2]      ;add element 5
ADD AL,ARRAY[SI+4]      ;add element 7
```

Working of 8086 with an example

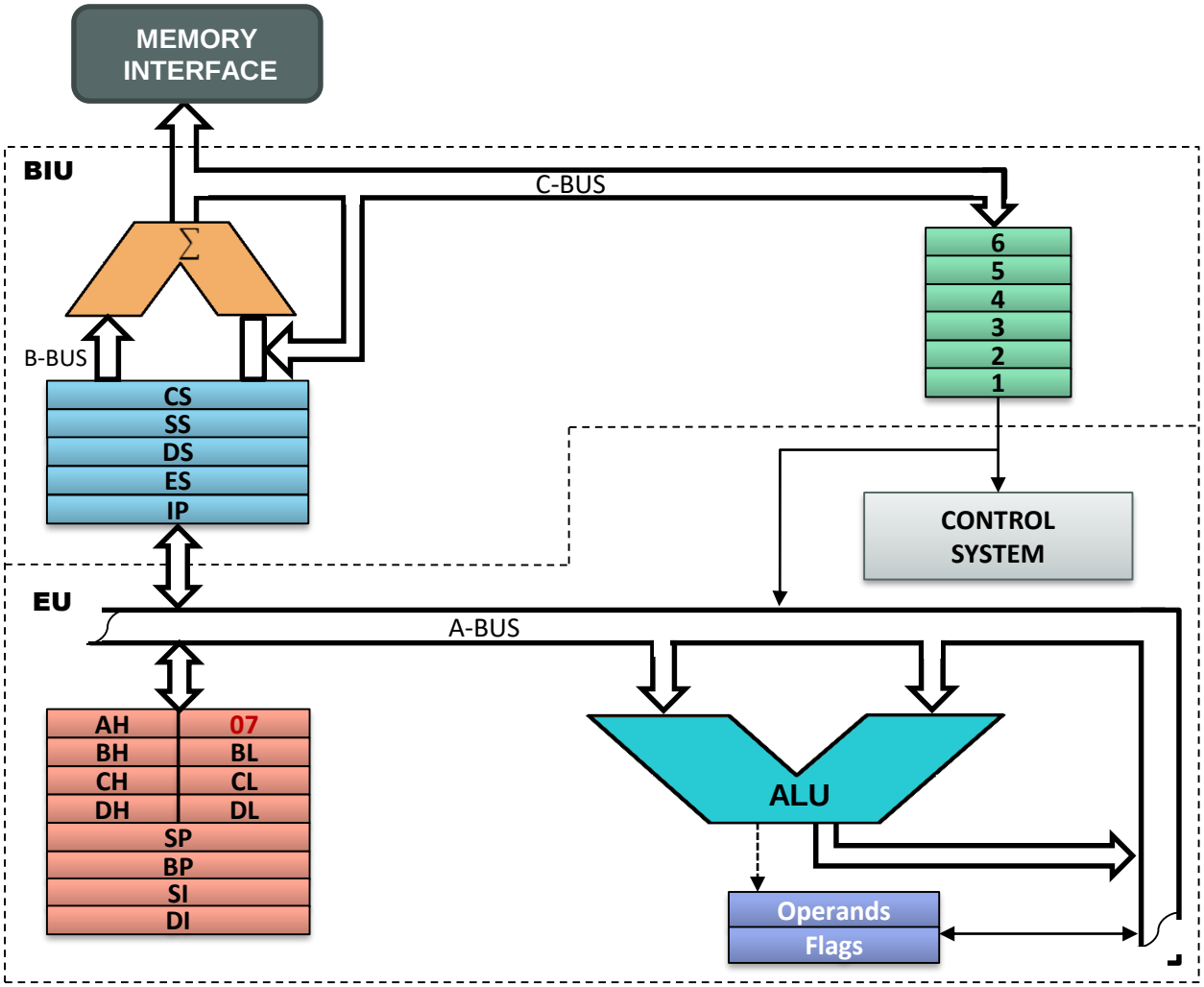


MOV AL, 07H
ADD AL, 05H

Working of 8086 with an example



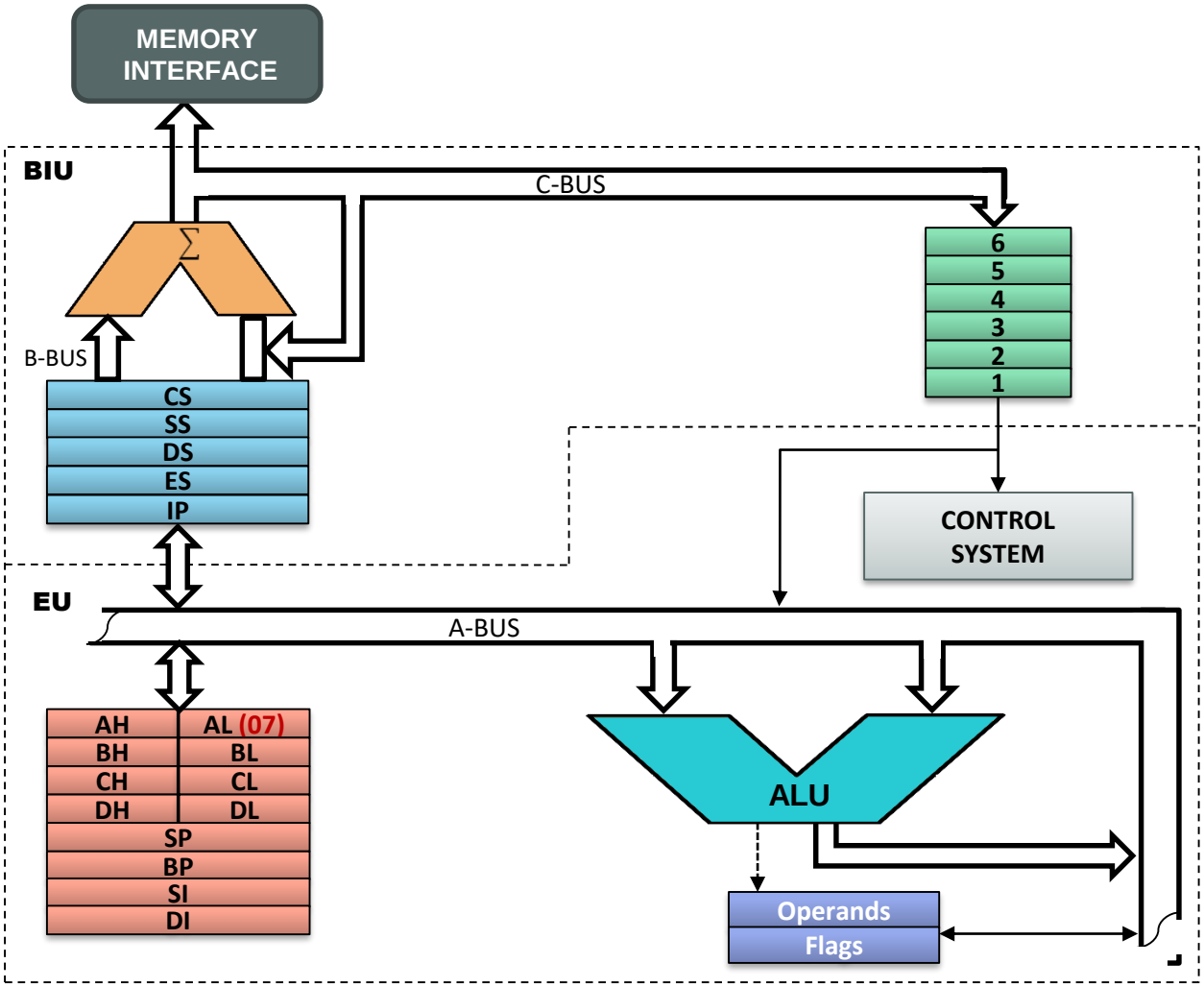
Working of 8086 with an example



MOV AL, 07H

ADD AL, 05H

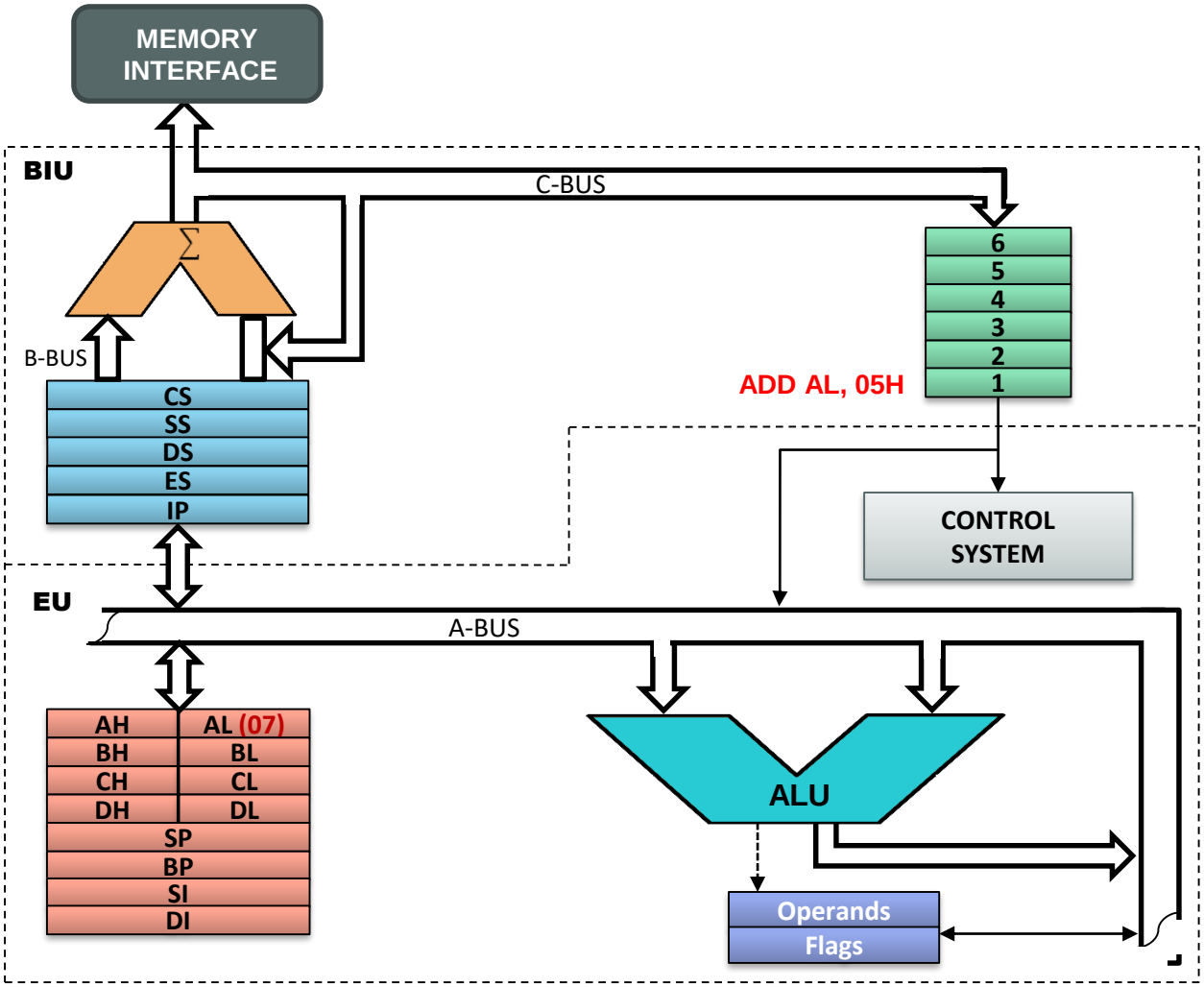
Working of 8086 with an example



MOV AL, 07H

ADD AL, 05H

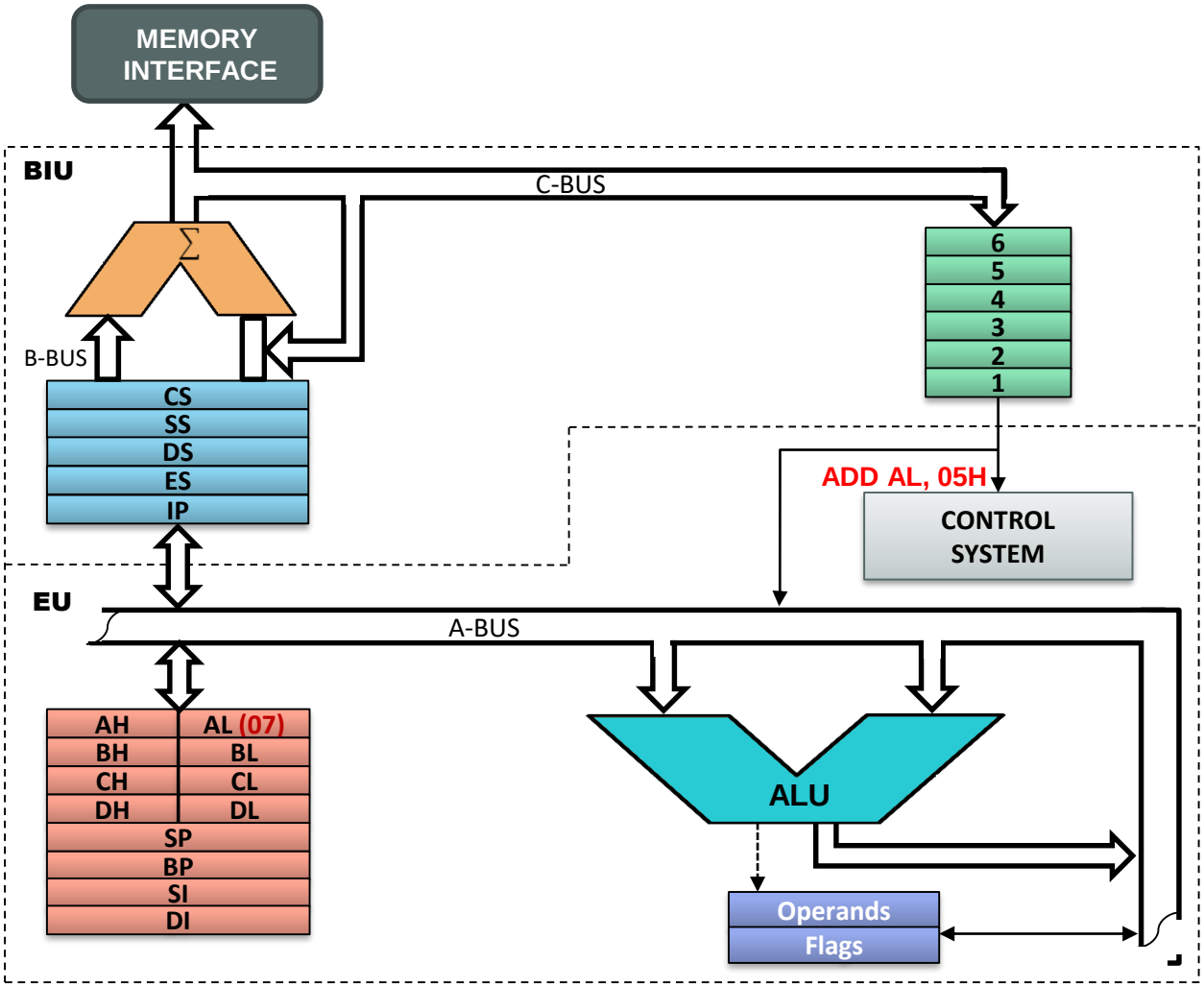
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

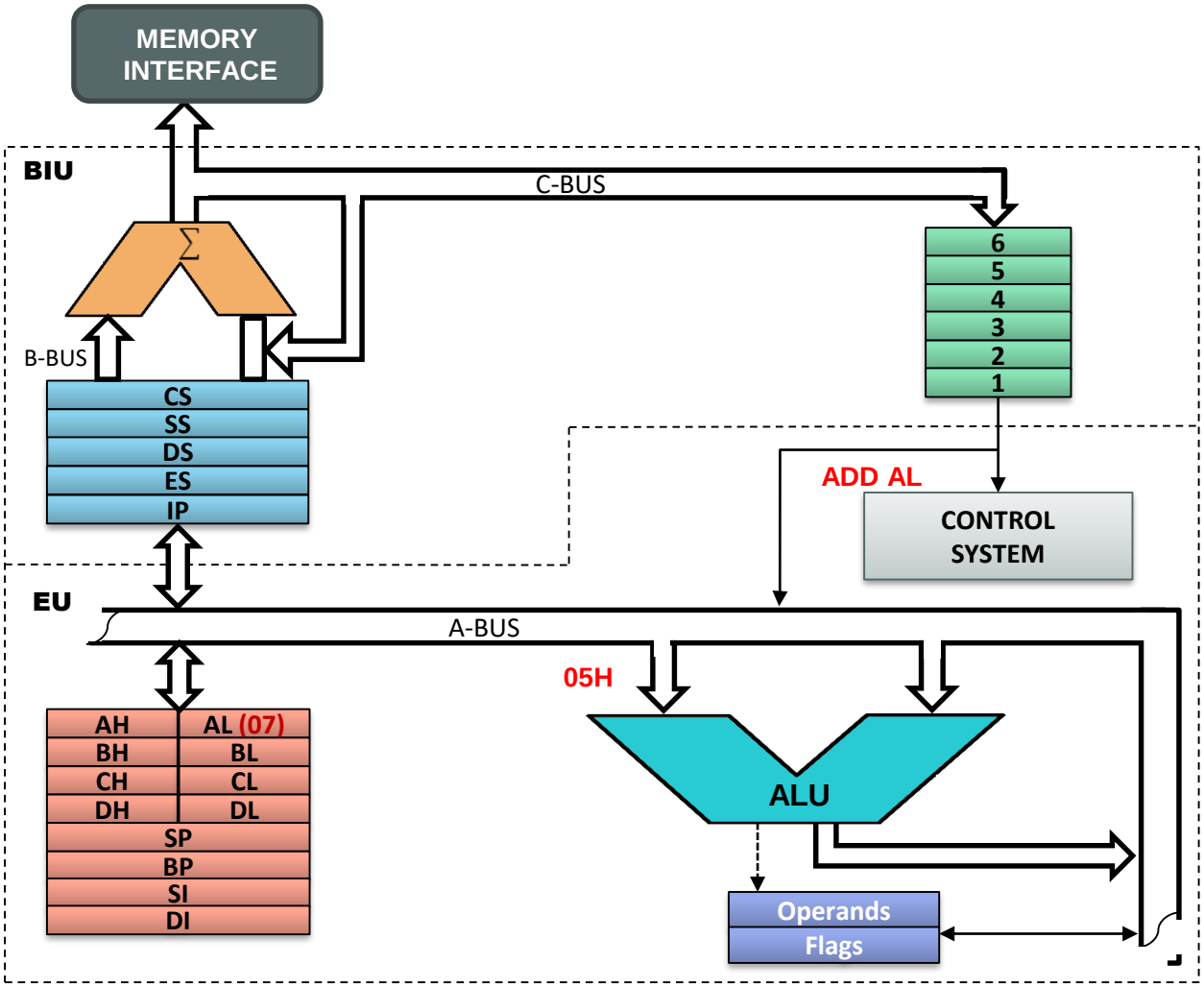
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

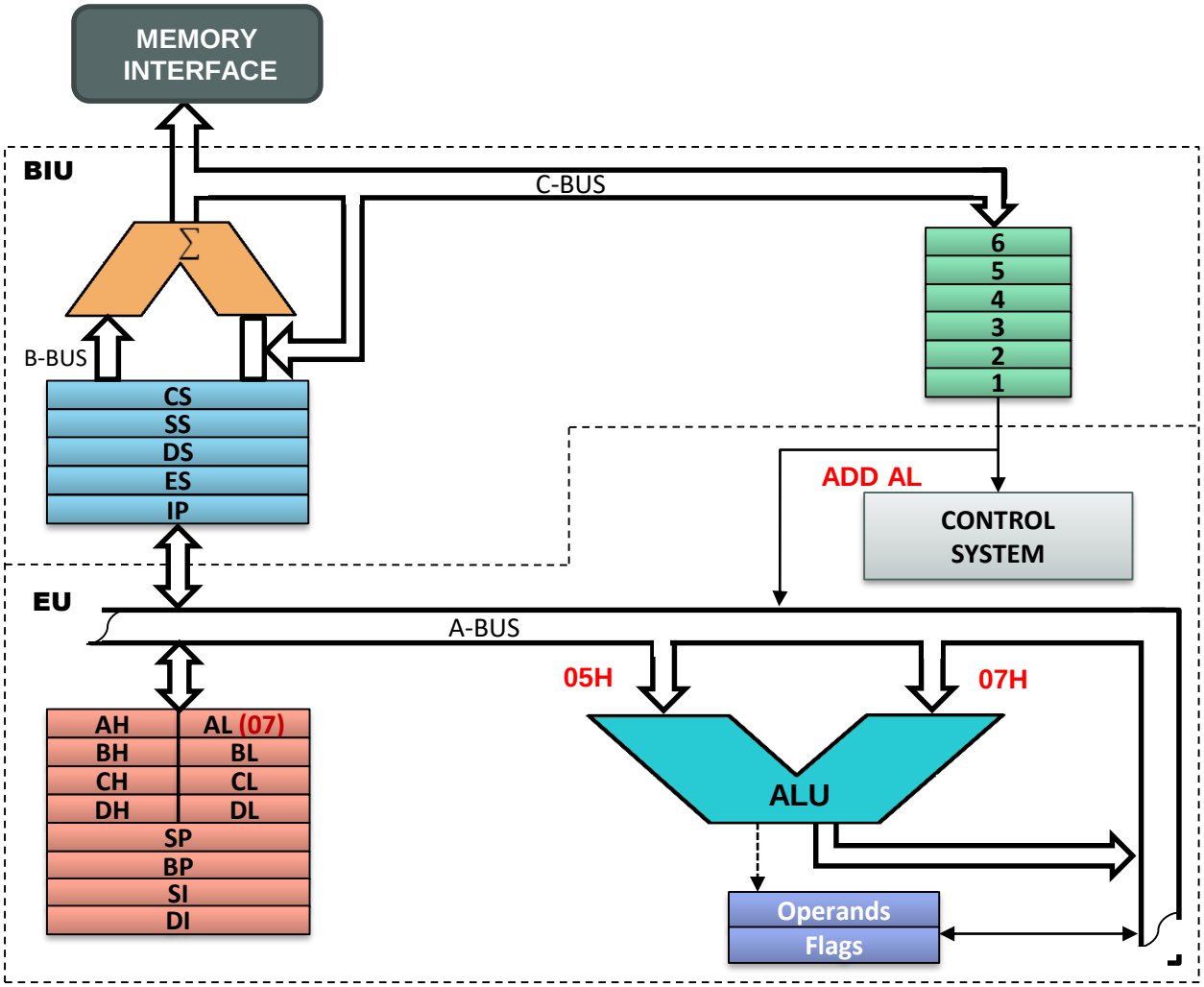
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

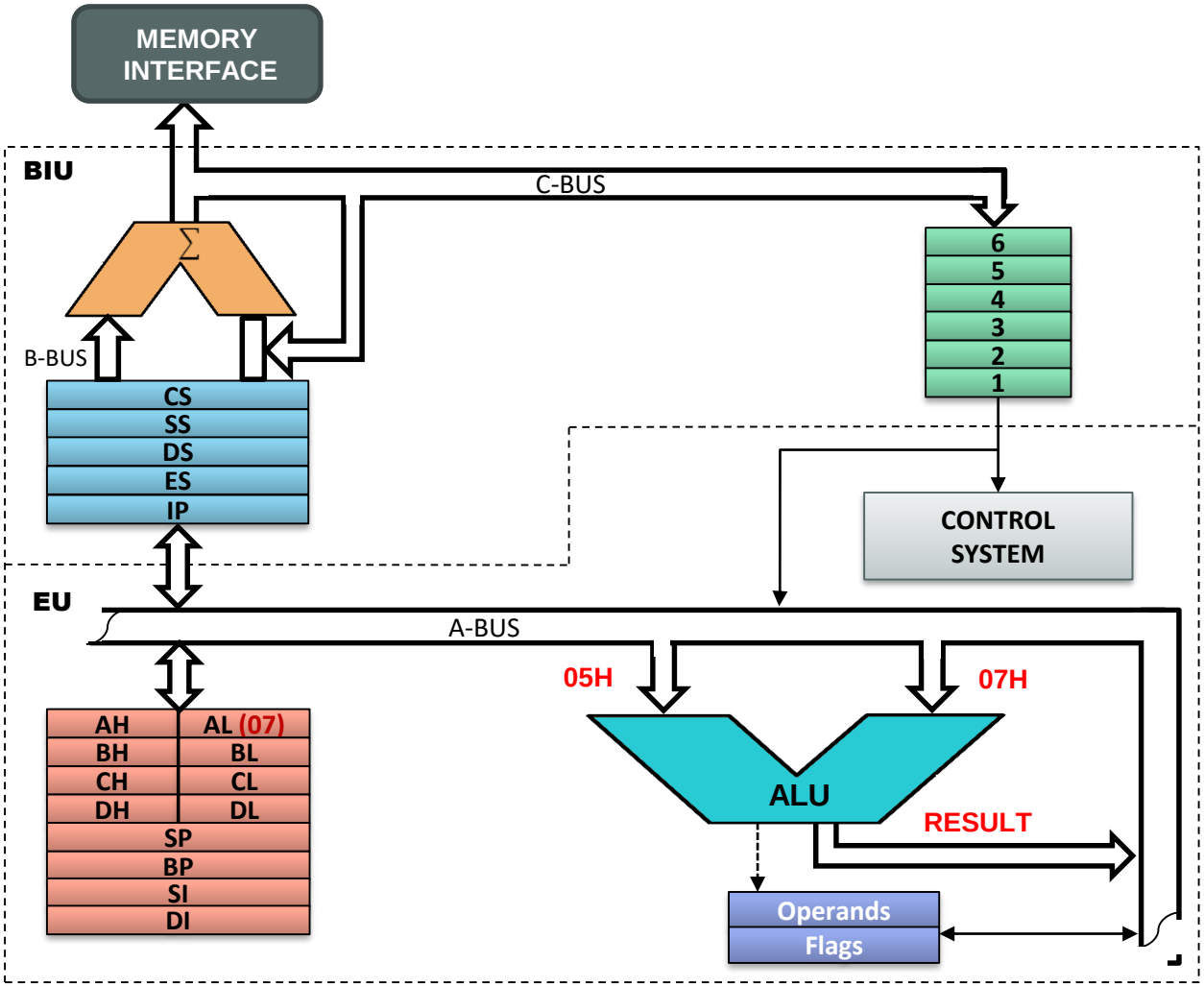
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

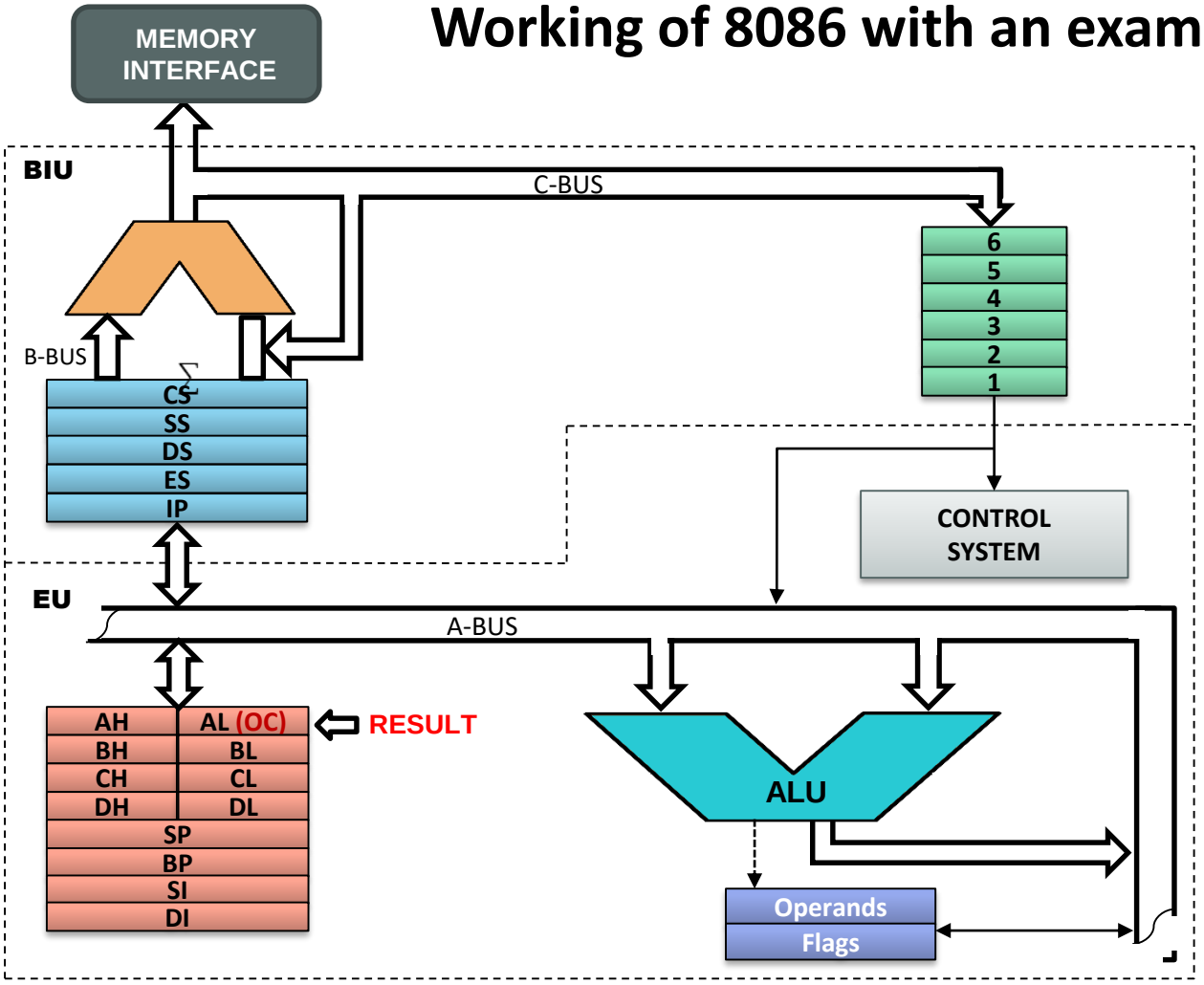
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

```
-t=104
AX=0007 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0106 NV UP EI NG NZ AC PO NC
0859:0106 0405              ADD     AL,05
-t
AX=000C BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0108 NV UP EI PL NZ NA PE NC
0859:0108 0000              ADD     [BX+SI],AL                  DS:0000=E6
```

INC Instruction

- Increment addition (INC) adds 1 to a register or a memory location, except a segment register.

E.g.

INC BL	BL = BL + 1
INC SP	SP = SP + 1
INC EAX	EAX = EAX + 1
INC BYTE PTR[BX]	Adds 1 to the byte contents of the data segment memory location addressed by BX
INC WORD PTR[SI]	Adds 1 to the word contents of the data segment memory location addressed by SI
INC DWORD PTR[ECX]	Adds 1 to the doubleword contents of the data segment memory location addressed by ECX
INC DATA1	Adds 1 to the contents of data segment memory location DATA1
INC RCX	Adds 1 to RCX (64-bit mode)

Indirect Memory Increments

- INC BYTE PTR [DI] instruction clearly indicates byte-sized memory data

E.g.

```
MOV AL, byte ptr [BX] ;   Load the byte at the address in BX into AL
ADD byte ptr [SI], 5 ;    Add 5 to the byte at the address in SI
```

- INC WORD PTR [DI] instruction unquestionably indicates a word-sized memory data
- INC DWORD PTR [DI] instruction indicates doubleword-sized data.

Example 1

```
-e 2010
0859:2010  00.10   00.20   00.30   00.40   00.50   00.60   00.
-a 100
0859:0100 mov bx, 2010
0859:0103 mov al, byte ptr[bx]
0859:0105 mov cx, word ptr[bx]
0859:0107
```

```
-t=100
AX=000C BX=2010 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0103 NV UP EI NG NZ AC PO NC
0859:0103 8A07              MOV     AL,[BX]                         DS:2010=10
-t
AX=0010 BX=2010 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0105 NV UP EI NG NZ AC PO NC
0859:0105 8B0F              MOV     CX,[BX]                       DS:2010=2010
-t
AX=0010 BX=2010 CX=2010 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0107 NV UP EI NG NZ AC PO NC
0859:0107 050000            ADD     AX,0000
-
```

Example 2

```
-e 2010
0859:2010  00.10   00.20   00.30   00.40   00.50   00.60   00.
-a 100
0859:0100 mov bx,2010
0859:0103 mov al, byte ptr[BX]
0859:0105 inc byte ptx[BX]
0859:0107 mov ah, byte ptr[BX]
0859:0109 mov cx, word ptr[BX]
0859:010B inc word ptr[BX]
0859:010D mov dx, word ptr[BX]
0859:010F
```

Example 2

```
-e 2010
0859:2010  00.10   00.20   00.30   00.40   00.50   00.60   00.
-a 100
0859:0100 mov bx,2010
0859:0103 mov al, byte ptr[BX]
0859:0105 inc byte ptx[BX]
0859:0107 mov ah, byte ptr[BX]
0859:0109 mov cx, word ptr[BX]
0859:010B inc word ptr[BX]
0859:010D mov dx, word ptr[BX
0859:010F
```

```
-t
AX=0010 BX=2010 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0105 NV UP EI NG NZ NA PE NC
0859:0105 FE07              INC     BYTE PTR [BX]                 DS:2010=10
-t
AX=0010 BX=2010 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0107 NV UP EI PL NZ NA PE NC
0859:0107 8A27              MOV     AH,[BX]                       DS:2010=11
-t
AX=1110 BX=2010 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0109 NV UP EI PL NZ NA PE NC
0859:0109 8B0F              MOV     CX,[BX]                     DS:2010=2011
-t
AX=1110 BX=2010 CX=2011 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=010B NV UP EI PL NZ NA PE NC
0859:010B FF07              INC     WORD PTR [BX]               DS:2010=2011
-t
AX=1110 BX=2010 CX=2011 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=010D NV UP EI PL NZ NA PE NC
0859:010D 8B17              MOV     DX,[BX]                     DS:2010=2012
-t
AX=1110 BX=2010 CX=2011 DX=2012 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=010F NV UP EI PL NZ NA PE NC
0859:010F 0000              ADD     [BX+SI],AL                  DS:2010=12
```

DEC Instruction

- Decrement subtraction (DEC) subtracts 1 from a register or the contents of a memory location

E.g.

DEC BH	BH = BH – 1
DEC CX	CX = CX – 1

ADC (Addition with Carry)

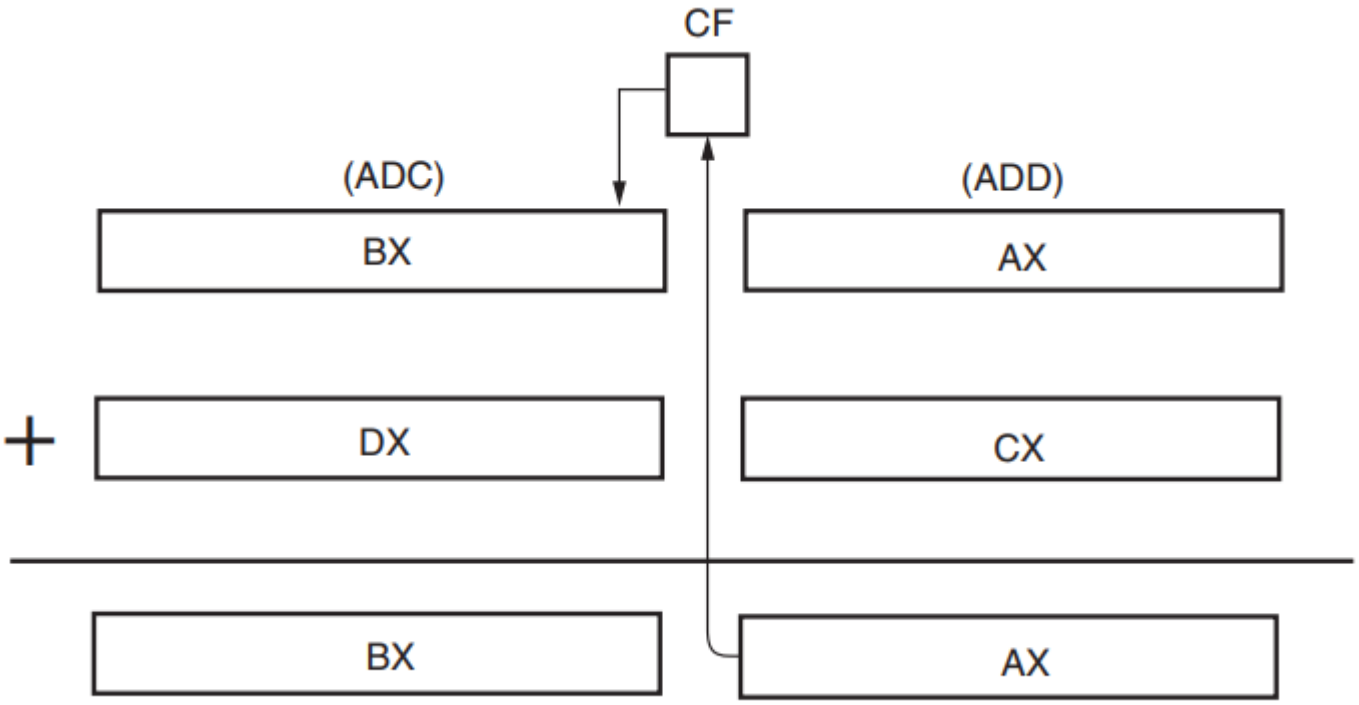
- An addition-with-carry instruction (ADC) adds the bit in the carry flag (C) to the operand data.
- This instruction mainly appears in software that adds numbers that are wider than 16 bits in the 8086–80286

E.g.

ADC AL,AH	AL = AL + AH + carry
ADC CX,BX	CX = CX + BX + carry

ADC (Addition with Carry)

E.g.
ADD AX,CX
ADC BX,DX



CMP (Comparison)

- Subtraction that changes only the flag bits; the destination operands never changes.

E.g.

CMP CL,BL	CL – BL
CMP AX,SP	AX – SP
CMP EBP,ESI	EBP – ESI
CMP RDI,RSI	RDI – RSI (64-bit mode)
CMP AX,2000H	AX – 2000H
CMP R10W,12H	R10 (word portion) – 12H (64-bit mode)
CMP [DI],CH	CH subtracts from the byte contents of the data segment memory location addressed by DI
CMP CL,[BP]	The byte contents of the stack segment memory location addressed by BP subtracts from CL
CMP AH,TEMP	The byte contents of data segment memory location TEMP subtracts from AH

CMP

Zero Flag (ZF): Set if the result of the subtraction is zero.

Sign Flag (SF): Set to the sign bit of the result (the most significant bit of the result).

Overflow Flag (OF): Set if there is a signed overflow in the result.

Carry Flag (CF): Set if there is a borrow (or no borrow) from the most significant bit during subtraction (unsigned overflow).

Program Control Instructions

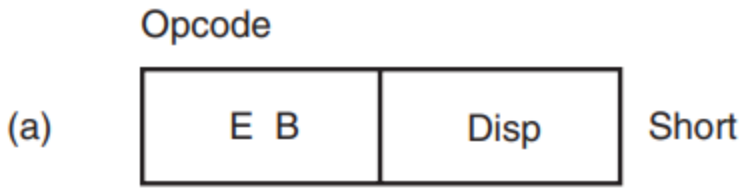
- Direct the flow of a program and allow the flow to change.
- A change in flow often occurs after a decision made with the CMP or TEST instruction is followed by a conditional jump instruction
- jump (JMP), allows the programmer to skip sections of a program and branch to any part of the memory for the next instruction.
- A conditional jump instruction allows the programmer to make decisions based upon numerical tests.
- Results of numerical tests are held in the flag bits, which are then tested by conditional jump instructions

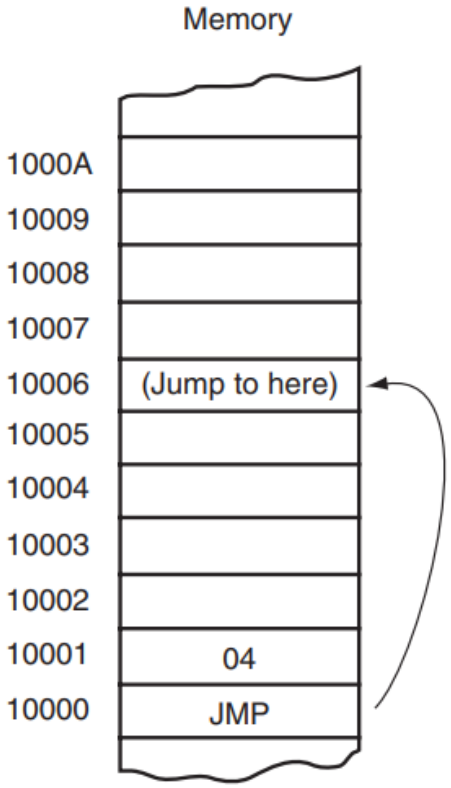
Unconditional Jump (JMP)

- Short jumps are called relative jumps because they can be moved, along with their related software, to any location in the current code segment without a change.

Displacement between +127 and -128

IP= IP+ Sign Extend(Displacement)





A short jump to four memory locations beyond the address of the next instruction

CS = 1000H
IP = 0002H
New IP = IP + 4
New IP = 0006H

Conditional Jump

- Conditional jump instructions are always short jumps in the 8086
- This limits the range of the jump to within +127 bytes and -128 bytes from the location following the conditional jump
- The conditional jump instructions test the following flag bits: sign (S), zero (Z), carry (C), parity (P), and overflow (0).
- If the condition under test is true, a branch to the label associated with the jump instruction occurs.
- If the condition is false, the next sequential step in the program executes.

E.g. JC Jump if carry is set

Assembly Language	*Tested Condition*	*Operation*
JA	Z = 0 and C = 0	Jump if above
JAE	C = 0	Jump if above or equal
JB	C = 1	Jump if below
JBE	Z = 1 or C = 1	Jump if below or equal
JC	C = 1	Jump if carry
JE or JZ	Z = 1	Jump if equal or jump if zero
JG	Z = 0 and S = 0	Jump if greater than
JGE	S = 0	Jump if greater than or equal
JL	S != O	Jump if less than
JLE	Z = 1 or S != O	Jump if less than or equal
JNC	C = 0	Jump if no carry
JNE or JNZ	Z = 0	Jump if not equal or jump if not zero
JNO	O = 0	Jump if no overflow
JNS	S = 0	Jump if no sign (positive)
JNP or JPO	P = 0	Jump if no parity or jump if parity odd
JO	O = 1	Jump if overflow
JP or JPE	P = 1	Jump if parity or jump if parity even
JS	S = 1	Jump if sign (negative)
JCXZ	CX = 0	Jump if CX is zero
JECXZ	ECX = 0	Jump if ECX equals zero
JRCXZ	RCX = 0	Jump if RCX equals zero (64-bit mode)

Thank You