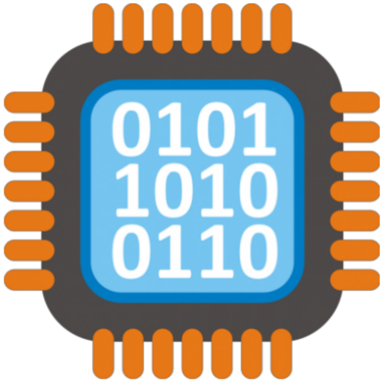
بسم الله الرحمن الرحيم

Secure Assembly Coding

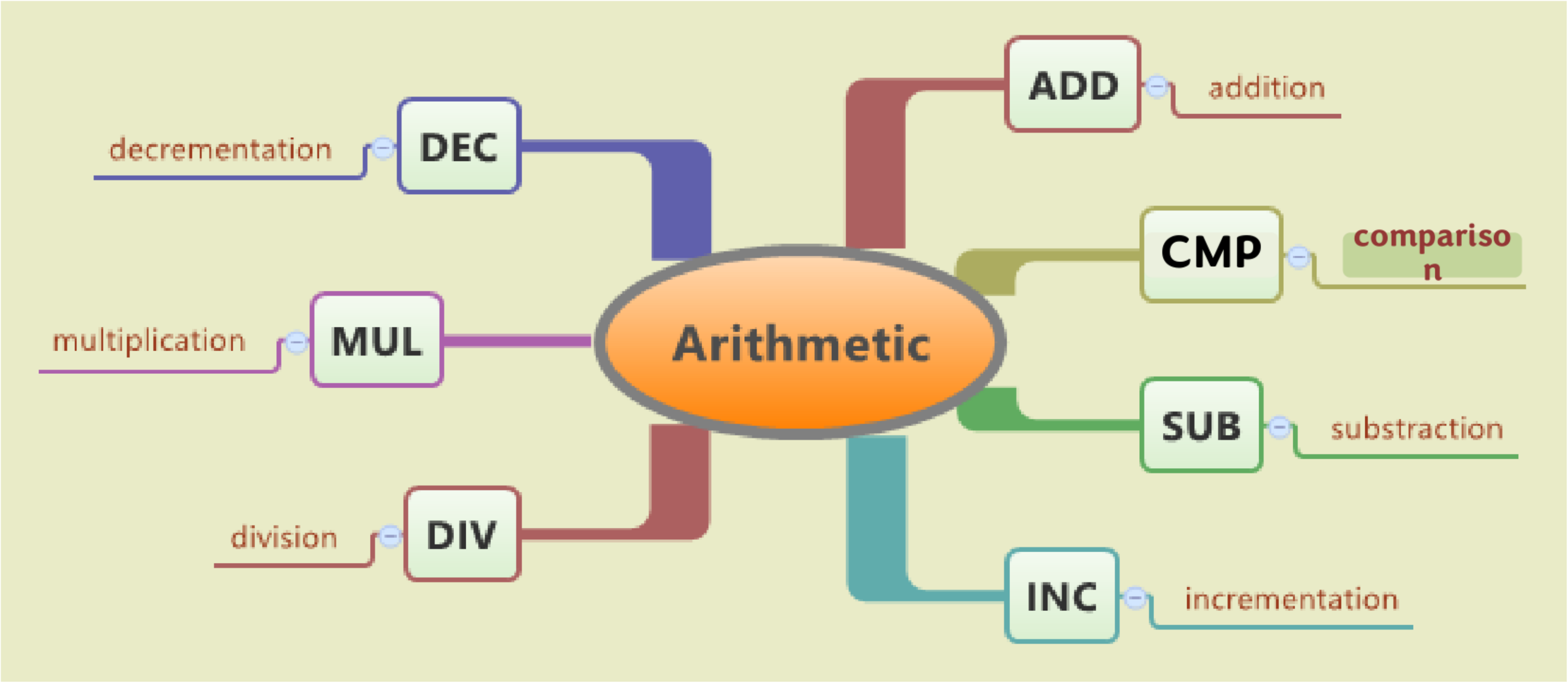
Week # 7 Lectures

Dr. Qasem Abu Al-Haija,
Department of Cybersecurity,



8086 Instruction Set

Group 2 : Arithmetic Instructions



Arithmetic Instructions:

ADD, ADC, SUB, SBB, INC, DEC, NEG, CMP, MUL, IMUL, DIV, IDIV, DAA, DAS, AAA, AAS, AAD, AAM, CBW, CWD.

8086 Instruction Set

Group 2 : Arithmetic Instructions

	Addition
ADD a, b	Add byte or word
ADC a, b	Add byte or word with carry
INC reg/mem	Increment byte or word by one

Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

(a)

Destination
Reg 16
Reg 8
Memory

(b)

(a) Allowed operands for ADD and ADC
(b) Allowed operands for INC

The AF, CF, OF, PF, SF and ZF flags are affected by the execution of ADD/SUB instruction

	Subtraction
SUB a, b	Subtract byte or word
SBB a, b	Subtract byte or word with borrow
DEC reg/mem	Decrement byte or word by one
NEG reg/mem	Negate byte or word
CMP a, b	Compare byte or word

Destination	Source
Register	Register
Register	Memory
Memory	Register
Accumulator	Immediate
Register	Immediate
Memory	Immediate

(b)

Destination
Reg16
Reg 8
Memory

(c)

Destination
Register
Memory

(d)

(b) Allowed operands for SUB and SBB instructions
(c) Allowed operands for DEC instruction
(d) Allowed operands for NEG instruction

a = “reg” or “mem,” b = “reg” or “mem” or “data.”

8086 Instruction Set
Group 2 : Arithmetic Instructions

Multiplication

MUL reg/mem	Multiply byte or word unsigned	for byte
IMUL reg/mem	Integer multiply byte or word (signed)	[AX] ← [AL] · [mem/reg]
		for word
		[DX][AX] ← [AX] · [mem/reg]

Source =
Mem8/Mem16/Reg8/Reg16

Division

DIV reg/mem	Divide byte or word unsigned	16 ÷ 8 bit; $[AX] \leftarrow \frac{[AX]}{[mem/reg]}$
IDIV reg/mem	Integer divide byte or word (signed)	[AH] ← remainder
		[AL] ← quotient
		32÷16 bit; $[DX:AX] \leftarrow \frac{[DX:AX]}{[mem/reg]}$
		[DX] ← remainder
		[AX] ← quotient

Source =
Mem8/Mem16/Reg8/Reg16

—*NOTE: if you are accessing memory with a single operand operation such as MUL, DIV, INC…, then you will have to specify the type of data (byte or word) ==>Two assembler directives are used for this purpose:*

BYTE PTR & WORD PTR

8086 Instruction Set
Group 2 : Arithmetic Instructions

- **Examples :**
  - ADD BL, 80H ; Add immediate data 80H to BL
  - ADD CX, 12B0H ; Add immediate data 12B0H to CX
  - ADD AX, CX ; Add content of AX and CX and store result in AX
  - ADD AL, [BX] ; Add AL to the byte from memory at [BX] and store result in AL.
  - ADD CX, [SI] ; Add CX and the word from memory at [SI] and store result in CX.
  - ADD [BX], DL ; Add DL with the byte from Mem at [BX] & store result in Mem at [BX].
  - SUB AL, BL ; Subtract BL from AL and store result in AL
  - SUB CX, BX ; Subtract BX from CX and store result in CX
  - SUB BX, [DI] ; Subtract the word in memory at [DI] from BX and store result in BX
  - SUB [BP], DL ; Subtract DL from the byte in Mem at [BP] & store result in Mem at [BP].
  - INC CL ; Increment content of CL by 1
  - INC AX ; Increment content of AX by 1
  - INC BYTE PTR [BX] ; Increment byte in memory at [BX] by 1
  - INC WORD PTR [SI] ; Increment word in memory at [SI] by 1

8086 Instruction Set

Group 2 : Arithmetic Instructions

- **Examples :**

– MUL CH	; Multiply AL and CH and store result in AX
– MUL BX	; Multiply AX and BX and store result in DX-AX
– MUL BYTE PTR [BX]	; Multiply AL with byte in memory at [BX] & store result in AX
– IMUL BL	; Multiply AL with BL and store result in AX
– IMUL AX	; Multiply AX and AX and store result in DX-AX
– IMUL BYTE PTR [BX]	; Multiply AL with byte from memory at [BX] & store result in AX
– IMUL WORD PTR [SI]	; Multiply AX with word from memory at [SI] & store result in DX-AX
– DIV DL	; Divide word in AX by byte in DL.
	; Quotient is stored in AL and remainder is stored in AH
– DIV CX	; Divide double word (32 bits) in DX-AX by word in CX.
	; Quotient is stored in AX and remainder is stored in DX
– DIV BYTE PTR [BX]	; Divide word in AX by byte from memory at [BX].
	; Quotient is stored in AL and remainder is stored in AH.

8086 Instruction Set

Group 2 : Arithmetic Instructions

EX: if (AX) = 0005_{16} & (CL) = 02_{16} → DIV CL → (AH) = 01_{16} (Rem) & (AL) = 02_{16} (Quot).

EX: If (CX) = 2 and (DX AX) = -5_{10} = $FFFFFFFB_{16}$ → IDIV, after this IDIV, DX and AX will contain:

DX	AX
FFFF	FFFE
16-bit remainder = -1_{10}	16-bit quotient = -2_{10}

EX: If (AL)=20_{16} & (BL) = 02_{16} → MUL BL. → AX will contain 0040_{16}

EX: If (CL) = FDH = -3_{10} & (AL) = FEH = -2_{10} → IMUL CL → AX contains 0006H.

EX : If (AL) = FF_{16}= -1_{10} and (DH) = 02_{10} → IMUL DH → AX = $FFFE_{16}$ (-2_{10}).

Example:

```
MOV    BX, 0050H
MOV     CX, 3000H
MOV    DS, CX
MOV    [BX],0006H
MOV    AX, 0002H
MUL    WORD PTR [BX]
```

registers

	H	L
AX	00	0C
BX	00	50
CX	30	00
DX	00	00
CS	F400	
IP	0154	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	3000	
ES	0700	

Remember, signed numbers:

if 8 bit (−128 to 127)

if 16 bit (−32768 to 32767)

8086 Instruction Set

Group 2 : Arithmetic Instructions

EX: ADC AX, [BX] ; 0020+0100+1 = 0121

Before				After			
AX	0020	Memory locations		AX	0121	Memory locations	
DS	2020	20500	00	DS	2020	20500	00
BX	0300	20501	01	BX	0200	20501	01
CF	1			CF	0	PF = 1, AF = 0, ZF = 0, SF = 0, OF = 0	

EX: SBBCH,DL ; 03 – 02 – 1 = 0

Before		After		
CH	03	CH	0	
DL	02	DL	02	
CF	1	CF	0	PF =0, AF = 1, ZF = 1, SF = 0, OF = 0

EX: CMP DH, BL.

Before Execution:

Assume:

(DH) = 40H

(BL) = 30H

After Execution:

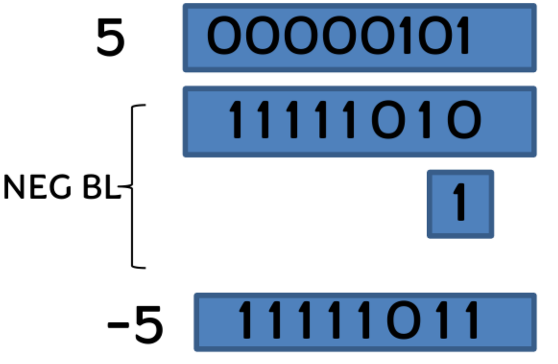
Result 10H is not provided

Flags are: CF= 0, PF=0,

AF=0, ZF=0, SF=0, & OF= 0

NEG (2'S COMPLEMENT)

- NEG DESTINATION
- DESTINATION REG., MEMORY (8-BIT OR 16-BIT)
- EXAMPLE:
- MOV BL, 5
- NEG BL



Flags affected: ZF, OF, SF, CF

8086 Instruction Set

Group 2 : Arithmetic Instructions

- **CBW: Convert byte to word (No Operand)**

if high bit of AL = 1 then: AH = 255 (0FFh)
Else, AH = 0

Example:

```
MOV AX, 0   ; AH = 0, AL = 0
MOV AL, -5  ; AX = 000FBh (251)
CBW         ; AX = 0FFFBh (-5)
RET
```

C	Z	S	O	P	A
unchanged					

- **CWD: Convert word to double word (No Operand)**

if high bit of AX = 1 then: DX = 65535 (0FFFFh)
Else, DX = 0

Example:

```
MOV DX, 0   ; DX = 0
MOV AX, 0   ; AX = 0
MOV AX, -5  ; DX AX = 00000h:0FFFBh
CWD         ; DX AX = 0FFFFh:0FFFBh
RET
```

C	Z	S	O	P	A
unchanged					

8086 Instruction Set

Group 2 : Arithmetic Instructions

- **AAA (ASCII Adjust after Addition) instruction**

— Must always follow the addition of two unpacked BCD operands in AL.

— When AAA is executed, the content of AL is changed to a valid unpacked BCD number and clears the top 4 bits of AL.

— The CF is set and AH is incremented if a decimal carry out from AL is generated.

- **Example:**

Let AL=05 decimal=0000 0101
BH=06 decimal=0000 0110
AH=00H

Consider the execution of the following instructions:

```
ADD AL, BH      ; AL=0BH=11 decimal and CF=0
AAA             ; AL=01 and AH=01 and CF=1
```

ASCII Table

Character	Decimal Number	Binary Number	Character	Decimal Number	Binary Number
blank space	32	0010 0000	^	94	0101 1110
!	33	0010 0001	-	95	0101 1111
"	34	0010 0010	`	96	0110 0000
#	35	0010 0011	a	97	0110 0001
$	36	0010 0100	b	98	0110 0010
A	65	0100 0001	c	99	0110 0011
B	66	0100 0010	d	100	0110 0100
C	67	0100 0011	e	101	0110 0101
D	68	0100 0100	f	102	0110 0110
E	69	0100 0101	g	103	0110 0111
F	70	0100 0110	h	104	0110 1000
G	71	0100 0111	i	105	0110 1001
H	72	0100 1000	j	106	0110 1010
I	73	0100 1001	k	107	0110 1011
J	74	0100 1010	l	108	0110 1100
K	75	0100 1011	m	109	0110 1101
L	76	0100 1100	n	110	0110 1110
M	77	0100 1101	o	111	0110 1111
N	78	0100 1110	p	112	0111 0000
O	79	0100 1111	q	113	0111 0001
P	80	0101 0000	r	114	0111 0010
Q	81	0101 0001	s	115	0111 0011
R	82	0101 0010	t	116	0111 0100
S	83	0101 0011	u	117	0111 0101
T	84	0101 0100	v	118	0111 0110
U	85	0101 0101	w	119	0111 0111
V	86	0101 0110	x	120	0111 1000
W	87	0101 0111	y	121	0111 1001
X	88	0101 1000	z	122	0111 1010

Since 05+06=11(decimal)=0101 H stored in AX in unpacked BCD form. When this result is to be sent to the printer, the ASCII code of each decimal digit is easily formed by adding 30H to each byte.

8086 Instruction Set
Group 2 : Arithmetic Instructions

- **AAS: ASCII Adjust after Subtraction**
  - This instruction always follows the subtraction of one unpacked BCD operand from another unpacked BCD operand in AL.
  - It changes the content of AL to a valid unpacked BCD number and clears the top 4 bits of AL.
  - The CF is set, and AH is decremented if a decimal carry occurred.

- **Example:**

```
Let     AL=09 BCD=0000 1001
        CL=05 BCD =0000 0101
        AH=00H
```

Consider the execution of the following instructions:

```
SUB AL, CL      ; AL=04 BCD
AAS             ; AL=04 BCD and CF=0, AH=00H
```

- AAA and AAS affect AF and CF flags and OF, PF, SF and ZF are left undefined.
- Another salient feature of the above two instructions are that the input data used in the addition or subtraction can be even in ASCII form of the unpacked decimal number and still we get the result in ordinary unpacked decimal number form and by adding 30H to the result, again we get ASCII form of the result.

8086 Instruction Set

Group 2 : Arithmetic Instructions

- **AAD: The ASCII adjust AX before Division instruction**

— Modifies the dividend in AH and AL, to prepare for the division of two valid unpacked BCD operands.

— After the execution of AAD, AH will be cleared, and AL will contain the binary equivalent of the original unpacked two-digit numbers.

— Initially AH contains the most significant unpacked digit and AL contains the least significant unpacked digit.

- **Example:** To perform the operation 32 decimal / 08 decimal

```
Let     AH=03H ; upper decimal digit in the dividend
        AL=02H ; lower decimal digit in the dividend
        CL=08H ; divisor
```

Consider the execution of the following instructions:

```
AAD       ; AX=0020H (binary equivalent of 32 decimal in 16-bit form)
DIV  CL   ; Divide AX by CL; AL will contain quotient & AH will contain remainder.
```

— AAD affects PF, SF and ZF flags. AF, CF and OF are undefined after execution of AAD.

8086 Instruction Set

Group 2 : Arithmetic Instructions

- **AAM: The ASCII Adjust AX after Multiplication instruction**

— Corrects the value of a multiplication of two valid unpacked decimal numbers. The higher order digit is placed in AH and the low order digit in AL.

- **Example:**

Let AL=05 decimal
CL=09 decimal

Consider the execution of the following instructions:

```
MUL  CH ; AX=002DH=45 decimal
AAM     ; AH=04 and AL=05 ( unpacked BCD form decimal number of 45)
        ; OR AX, 3030H; To get ASCII code of the result in AH and AL
```

— Note: this instruction is used only when it is needed.
— AAM affects flags the same as that of AAD.

Summary of Arithmetic Instructions

	Flag affected				
Instruction	*Z-flag*	*C-flag*	*S-flag*	*O-flag*	*A-flag*
ADD	Yes	Yes	Yes	Yes	Yes
ADC	Yes	Yes	Yes	Yes	Yes
SUB	Yes	Yes	Yes	Yes	Yes
SBB	Yes	Yes	Yes	Yes	Yes
INC	Yes	No	Yes	Yes	Yes
DEC	Yes	No	Yes	Yes	Yes
NEG	Yes	Yes	Yes	Yes	Yes
CMP	Yes	Yes	Yes	Yes	Yes
MUL	No	Yes	No	Yes	No
IMUL	No	Yes	No	Yes	No
DIV	No	No	No	No	No
IDIV	No	No	No	No	No
CBW	No	No	No	No	No
CWD	No	No	No	No	No

Example: 8086 Assembly Programming Using MASM

Write a program to add two 8-bit data (F0H and 50H) in 8086 and store results in memory.

```
DATA    SEGMENT                        ; Beginning of data segment
OPER1   DB      F0H                    ; First operand
OPER2   DB      50H                    ; Second operand
RESULT  DB      01 DUP (?)             ; A byte of memory is reserved for result
CARRY   DB      01 DUP (?)             ; A byte is reserved for storing carry
DATA    ENDS                           ; End of data segment

CODE    SEGMENT                        ; Beginning of code dement
START:  MOV AX, DATA                   ; Initialize AX with the segment address of DS
        MOV DS, AX                     ; Move AX content to DS
        MOV BX, OFFSET OPER1           ; Move the offset address of OPER1 to BX
        MOV AL, [BX]                   ; Move first operand to AL
        ADD AL, [BX+1]                 ; Add second operand to AL
        MOV SI, OFFSET RESULT          ; Store offset address of RESULT in SI
```

Example: 8086 Assembly Programming Using MASM

```
        MOV [SI], AL          ; Store content of AL in the location RESULT
        INC SI                ; Increment SI to point location of carry
        JC CAR                ; If carry =1, go to the place CAR
        MOV [SI], 00H         ; Store 00H in the location CARRY
        JMP LOC1              ; go to the place LOC1
CAR:    MOV [SI], 01H         ; Store 01H in the location CARRY

LOC1:   MOV AH, 4CH
        INT 21H               ; Return to DOS prompt
CODE    ENDS                  ; End of code segment

END START                     ; Program ends
```

In the above program, the instructions MOV AH, 4CH and INT 21H at the end of the program are used for returning to the DOS prompt after executing the program in the computer. If instead of these two instructions, if one writes HLT instruction, the computer will hang after executing the program as the CPU goes to halt state and the user will not be able to examine the result.

See other examples in the separate ppt file uploaded into Moodle

More Example_1

Thank you