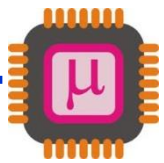


Fall 2018/19 – Lecture Notes # 9

- **Unsigned Subtraction**
- **Unsigned Multiplication and Division**
- **Logic Instructions**



Unsigned Subtraction

• Subtraction of Unsigned Numbers

▪ SUB (Subtract) Instruction

Format: **SUB dest, source** ; dest = dest - source

❖ In subtraction 2's complement method is used.

❖ Execution of SUB instruction

1. Take the 2's complement of the subtrahend (source operand)
2. Add it to the minuend (destination operand)
3. Invert the carry

Ex:

```

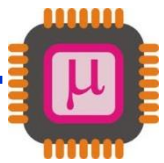
MOV    AL,3FH           ; load AL=3FH
MOV    BH,23H          ; load BH=23H
SUB    AL,BH           ; subtract BH from AL. Place result in AL

```

Execution steps:

AL	3F	0011 1111	0011 1111	
-BH	-23	-0010 0011	+ 1101 1101	(2's complement)
AL	1C	0001 1100	1 0001 1100	(CF=0) Step 3

CF=0, ZF=0, AF=1, PF=0, SF=0.



Unsigned Subtraction

• Subtraction of Unsigned Numbers

▪ SUB (Subtract) Instruction

- If the **CF=0**, the result is **positive** and the destination has the result.
- If the **CF=1**, the result is **negative** and the destination has the **2's complement** of the result. NOT and INC increment instructions can be used to change it.

Ex: ;from the data segment:

```
DATA1      DB      4CH
DATA2      DB      6EH
RESULT     DB      ?
```

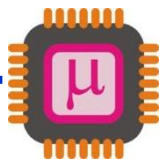
;from the code segment:

```
      MOV  DH, DATA1      ;load DH with DATA1 value (4CH)
      SUB  DH,DATA2        ;subtract DATA2 (6E) from DH (4C)
      JNC  NEXT           ;if CF=0 jump to NEXT target
      NOT  DH              ;if CF=1 take the 1's complement
      INC  DH              ;and increment to get 2's complement
NEXT:  MOV  RESULT,DH      ;save DH in RESULT
```

Analysis: Following the 3 steps for "SUB DH,DATA2"

4C	0100 1100		0100 1100
- 6E	0110 1110	2's comp	+ 1001 0010
- 22			0 1101 1110

CF=1(Step 3) the result is negative



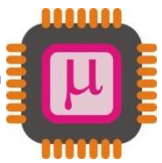
Unsigned Subtraction

- **Subtraction of Unsigned Numbers**

- **SBB (Subtract with borrow) Instruction**

Format: **SBB dest, source** ; dest = dest - source - CF

- Used in multibyte (multiword) numbers.
- If CF=0, SBB works exactly like SUB
- If CF=1, SBB subtracts 1 from the result



Unsigned Subtraction

- **Subtraction of Unsigned Numbers**

- **SBB (Subtract with borrow) Instruction**

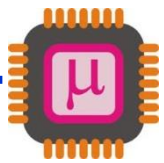
Ex: Analyze the following program:

```
.DATA
DATA_A      DD      62562FAH
DATA_B      DD      412963BH
RESULT      DD      ?
.....

      MOV     AX,WORD PTR DATA_A      ;AX=62FA
      SUB     AX,WORD PTR DATA_B      ;AX=AX – 963B
      MOV     WORD PTR RESULT,AX      ;save the result
      MOV     AX,WORD PTR DATA_A +2   ;AX=0625
      SBB     AX,WORD PTR DATA_B +2   ;SUB 0412 with borrow
      MOV     WORD PTR RESULT +2,AX    ;save the result
```

Note: PTR (Pointer) Directive is used to specify the size of the operand. Among the options for size are BYTE, WORD, DWORD and QWORD.

Solution: After the SUB, $AX = 62FA - 963B = CCBF$ and the carry flag is set. Since $CF=1$, when SBB is executed, $AX = 625 - 412 - 1 = 212$. Therefore, the value stored in RESULT is 0212CCBF.



Unsigned multiplication and Division

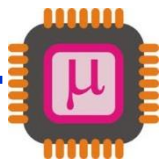
- **Multiplication of unsigned numbers**

- byte x byte :

- One of the operands must be in AL . The other operand can be either in a register or in memory.
- After the multiplication the result is in AX.

Ex:

```
.DATA
RESULT  DW      ?           ; result is defined in data segment
.CODE
.....
MOV     AL,25H             ; a byte is moved to AL
MOV     BL,65H            ; immediate data must be in a register
MUL     BL                ; AL= AL x BL = 25 x 65
MOV     RESULT,AX        ; result is saved
```



Unsigned multiplication and Division

- **Multiplication of unsigned numbers**

- byte x byte :

Ex:

```
.DATA  
DATA1 DB 25H  
DATA2 DB 65H  
RESULT DW ?  
;from the code segment:
```

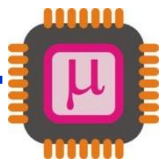
```
MOV AL,DATA1  
MOV BL,DATA2  
MUL BL  
MOV RESULT,AX
```

;register addressing mode

or

```
MOV AL,DATA1  
MUL DATA2  
MOV RESULT,AX
```

;direct addressing mode



Unsigned multiplication and Division

- **Multiplication of unsigned numbers**

- word x word :

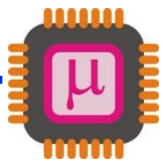
- One of the operands must be in AX . The other operand can be either in a register or in memory.
- After the multiplication the lower word is in AX and the higher word is in DX.

Ex:

```
.DATA  
DATA3    DW    2278H  
DATA4    DW    2F79H  
RESULT1  DW    2 DUP?
```

...

```
MOV      AX,DATA3      ; load first operand into AX  
MUL      DATA4        ; multiply it by the second operand  
MOV      RESULT1,AX    ; store the lower word of the result  
MOV      RESULT1+2,DX  ; store the higher word of the result
```

Unsigned multiplication and Division

- **Multiplication of unsigned numbers**

- word x byte :

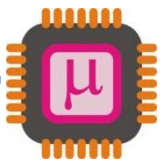
- Similar to word x word, but AL contains byte operand and AH must be zero.

Ex:

```
.DATA
DATA5    DB        6BH
DATA6    DW        12C3H
RESULT3  DW        2 DUP?
...
MOV      AL,DATA5           ; AL holds byte operand
SUB      AH,AH              ; AH must be cleared
MUL      DATA6             ; byte in AL multiplied by word operand
MOV      BX,OFFSET RESULT3 ; BX points the storage for product
MOV      [BX],AX            ; AX holds lower word
MOV      [BX]+2,DX          ; DX holds higher word
```

Unsigned Multiplication summary

Multiplication	Operand 1	Operand 2	Result
byte x byte	AL	register or memory	AX
word x word	AX	register or memory	DX AX
word x byte	AL=byte, AH=0	register or memory	DX AX



Unsigned multiplication and Division

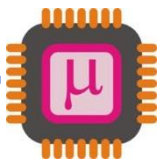
- **Division of unsigned numbers**

- byte / byte

- Numerator must be in AL and AH must be set to zero
- Denominator **cannot be immediate** but can be in memory or in a register.
- After the division AL will have the quotient and AH will have the remainder.

Ex:

```
.DATA
DATA7      DB      95
DATA8      DB      10
QUOT1      DB      ?
REMAIN1    DB      ?
;using direct mode
MOV        AL,DATA7      ;AL holds numerator
SUB        AH,AH         ;AH must be cleared
DIV        DATA8        ;divide AX by DATA8
MOV        QUOT1,AL      ;quotient = AL = 09
MOV        REMAIN1,AH    ;remainder = AH = 05
;using register addressing mode
MOV        AL,DATA7      ;AL holds numerator
SUB        AH,AH         ;AH must be cleared
MOV        BH,DATA8      ;move denominator to a register
DIV        BH            ;divide AX by BH
MOV        QUOT1,AL      ;quotient = AL = 09
MOV        REMAIN1,AH    ;remainder = AH = 05
;using the immediate addressing mode will give an error
MOV        AL,DATA7
SUB        AH,AH
DIV        10            ; X : immediate mode is not allowed
```



Unsigned multiplication and Division

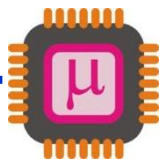
- **Division of unsigned numbers**

- word /word

- Numerator must be in AX and DX must be cleared
- Denominator can be in memory or in a register.
- After the division AX will have the quotient and DX will have the remainder

Ex:

```
.DATA
A          DW          10050
B          DW          100
QUOT2     DW          ?
REMAIN2   DW          ?
.CODE
:
MOV       AX,A          ;AX holds numerator
SUB       DX,DX         ;DX must be cleared
MOV       BX,B          ;BX is used for denominator
DIV       BX            ;divide AX by BX
MOV       QUOT2,AX      ;quotient = AX = 64H (100 )
MOV       REMAIN2,DX    ;remainder = DX = 32H (50)
:
```



Unsigned multiplication and Division

- **Division of unsigned numbers**

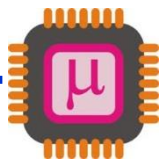
- word / byte

- Numerator must be in AX
- Denominator can be in memory or in a register.
- After the division AL will have the quotient and AH will have the remainder

Ex:

Ex:

```
.DATA
QUO      DB      ?
REM      DB      ?
.CODE
:
MOV      AX,2055      ;AX holds numerator
MOV      CL,100      ;BX is used for denominator
DIV      CL          ;divide AX by CL
MOV      QUO,AL      ;AL holds the quotient = AL = 14H (20)
MOV      REM,AH      ;AH holds the remainder = AH = 37H (55)
:
```



Unsigned multiplication and Division

- **Division of unsigned numbers**

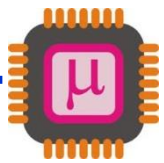
- doubleword/word

- Numerator must be in AX and DX, least significant word in AX and most significant word in DX.
- Denominator can be in memory or in a register.
- After the division AX will have the quotient and DX will have the remainder

Ex:

```
.DATA
DATA1      DD      105432
DATA2      DW      10000
QUOT       DW      ?
REMAIN     DW      ?
.CODE
:
MOV        AX,WORD PTR DATA1      ;AX holds the lower word
MOV        DX,WORD PTR DATA1+2    ;DX holds the higher word of the numerator
DIV        DATA2
MOV        QUOT,AX                  ;AX holds the quotient
MOV        REMAIN,DX                ;DX holds the remainder
:
```

“Divide Error”: If the denominator is zero (dividing any number by 00) and if the quotient is too large for the assigned register, “Divide Error” message will be displayed.



Logic Instructions

- Logic Instructions

- AND Instruction

Format: **AND dest, source** ; dest = dest & source

Ex: MOV BL,35H
 AND BL,0FH ;AND BL with 0FH and place the result in BL (BL=05H)

Solution:

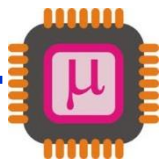
35H	00110101	
<u>0FH</u>	<u>&00001111</u>	
05H	00000101	SF=0, ZF=0, PF=1, CF=OF=0

Ex: AND DH,DH
 JZ NEXT

....

NEXT:

This operation will AND DH with itself and if the result is zero set ZF=1 jumping to NEXT.



Logic Instructions

- Logic Instructions

- OR Instruction

Format: **OR dest, source**; dest = dest ® source

Ex:

```
MOV    AX,0504H
OR     AX,0DA68H
```

Solution:

0504H	0000 0101 0000 0100	
<u>® DA68H</u>	<u>® 1101 1010 0110 1000</u>	
DF6CH	1101 1111 0110 1100	SF=1, ZF=0, PF=1,CF=OF=0