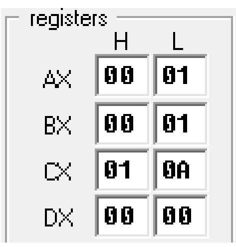
CSE341: Microprocessor LAB
Assembly Language
emu8086 Emulator Problem-Solution
#Use of Data Registers#

Task 01
Take input in the register AX, and then move it to BX using the MOV instruction.

Solution
```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 1
    mov bx, ax
```

```
.model small                    ; Defines the memory model as 'small', meaning both code and data fit within 64KB each.
.stack 100h                     ; Reserves 256 bytes (100h) for the stack.
.data                           ; Start of the data segment.

.code                           ; Start of the code segment.
main proc                       ; Beginning of the main procedure (entry point of the program).
    mov ax, @data               ; Load the address of the data segment into AX.
    mov ds, ax                  ; Move the value in AX into DS to initialize the data segment register.

    mov ax, 1                   ; Load the value 1 into the AX register.
    mov bx, ax                  ; Copy the value of AX into BX (now BX = 1).
```

Result:



Task 02
Swap two numbers, using a maximum of 3 registers.
Hint: Use the MOV instruction.

Solution
```
.model small
```

```
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 1
   mov bx, 2
   mov cx, ax
   mov ax, bx
   mov bx, cx
```

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 1        ; Loads the value 1 into register AX.
    mov bx, 2        ; Loads the value 2 into register BX.
    mov cx, ax       ; Copies the value in AX (which is 1) into CX.
    mov ax, bx       ; Copies the value in BX (which is 2) into AX.
    mov bx, cx       ; Copies the value in CX (which is 1) into BX.
```

Result

registers

	H	L
AX	00	02
BX	00	01
CX	00	01
DX	00	00

Task 03

Add two numbers using two registers.

Solution

```
.model small
.stack 100h
.data

.code
main proc
   mov ax,@data
```

```
   mov ds,ax

   mov ax,1
   mov bx,2

   ;adding ax with bx
   add ax,bx  ;ax = ax + bx
```

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 1         ; Loads the value 1 into the AX register.
    mov bx, 2         ; Loads the value 2 into the BX register.

    ; Adding AX with BX
    add ax, bx        ; Adds the value in BX (2) to the value in AX (1).
                      ; Now AX = AX + BX = 1 + 2 = 3
```

Result

registers

	H	L
AX	00	03
BX	00	02
CX	01	0D
DX	00	00

Task 04

Subtract two numbers using two registers. Do you always get the correct answer? What happens when you subtract larger number from the smaller one?

Solution

#Subtracting small number from a larger number:

```
.model small
.stack 100h
.data
.code
main proc
   mov ax ,@data
   mov ds, ax

   mov ax, 1
   mov bx, 2
```

```
;subtracting ax (small) from bx (large)
sub bx, ax  ;bx = bx - ax = 2 - 1 = 1
```

```
.model small
.stack 100h
.data
.code
main proc
     mov ax ,@data
     mov ds, ax

     mov ax, 1
     mov bx, 2

     ;subtracting ax (small) from bx (large)
     sub bx, ax   ;bx = bx - ax = 2 - 1 = 1
```

Result

registers

	H	L
AX	00	01
BX	00	01
CX	01	0D
DX	00	00

#Subtracting a lager number from a small number:

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 1
   mov bx, 2

   ;subtracting bx (large) from ax (small)
   sub ax, bx  ;ax = ax - bx = 1 - 2 = -1
```

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 1
    mov bx, 2

    ;subtracting bx (large) from ax (small)
    sub ax, bx  ;ax = ax - bx = 1 - 2 = -1

```

Result

registers

	H	L
AX	FF	FF
BX	00	02
CX	01	0D
DX	00	00

Task 05

Swap two numbers using ADD/SUB instructions only.

Solution

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 3
    mov bx, 4

    add ax, bx   ;ax = 3 + 4 = 7
    sub bx, ax   ;bx = 4 - 7 = -3
    neg bx       ;bx = 3
    sub ax, bx   ;ax = 7 - 3 = 4
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 3
     mov bx, 4

     add ax, bx    ;ax = 3 + 4 = 7
     sub bx, ax    ;bx = 4 - 7 = -3
     neg bx        ;bx = 3
     sub ax, bx    ;ax = 7 - 3 = 4
```

Result

registers

	H	L
AX	00	04
BX	00	03
CX	01	13
DX	00	00

Task 06

Perform the following arithmetic instructions. A, B, C are three variables to be declared beforehand

1. A = B - A
2. A = -(A + 1)
3. C = A + (B + 1); use inc
4. A = B – (A – 1); use dec

Solutions

1. **A = B - A**

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 3  ;A = 3
```

```
    mov bx, 4  ;B = 4

    sub bx, ax ;B = B - A
    mov ax, bx ;A = B - A
```

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 3   ;A = 3
    mov bx, 4   ;B = 4

    sub bx, ax ;B = B - A
    mov ax, bx ;A = B - A
```

Result

registers

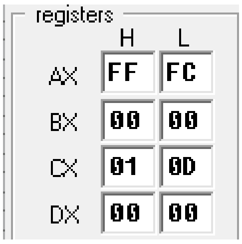
	H	L
AX	00	01
BX	00	01
CX	01	0F
DX	00	00

2. A = -(A + 1)

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 3  ;A = 3

    add ax, 1  ;A = A + 1 = 3 + 1 = 4
    neg ax     ;A = -(A + 1) = -4
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 3   ;A = 3

     add ax, 1   ;A = A + 1 = 3 + 1 = 4
     neg ax      ;A = -(A + 1) = -4
```

Result



3. C = A + (B + 1); use inc

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 3  ;A = 3
   mov bx, 4  ;B = 4

   inc bx     ;B = B + 1 = 5
   add ax, bx ;A = A + (B + 1) = 3 + 5 = 8
   mov cx, ax ;C = A + (B + 1) = 8
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 3   ;A = 3
     mov bx, 4   ;B = 4

     inc bx      ;B = B + 1 = 5
     add ax, bx  ;A = A + (B + 1) = 3 + 5 = 8
     mov cx, ax  ;C = A + (B + 1) = 8
```

Result

registers

	H	L
AX	00	08
BX	00	05
CX	00	08
DX	00	00

4. A = B – (A – 1); use dec

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 3  ;A = 3
   mov bx, 4  ;B = 4

   dec ax     ;A = A - 1 = 2
   sub bx, ax ;B = B - (A - 1) = 4 - 2 = 2
   mov ax, bx ;A = B - (A - 1) = 2
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 3   ;A = 3
     mov bx, 4   ;B = 4

     dec ax      ;A = A - 1 = 2
     sub bx, ax  ;B = B - (A - 1) = 4 - 2 = 2
     mov ax, bx  ;A = B - (A - 1) = 2
```

Result

registers

	H	L
AX	00	02
BX	00	02
CX	01	10
DX	00	00

Task 07

Perform the following arithmetic operations

1. X * Y
2. X / Y
3. X * Y / Z

Soultions

1. **X * Y**

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 3  ;X = 3
   mov bx, 4  ;Y = 4

   mul bx     ;X = X * Y
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 3   ;X = 3
     mov bx, 4   ;Y = 4

     mul bx      ;X = X * Y
```

Result

registers

	H	L
AX	00	0C
BX	00	04
CX	01	0D
DX	00	00

2. X / Y

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 4  ;X = 4
   mov bx, 3  ;Y = 3

   div bx
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 4   ;X = 4
     mov bx, 3   ;Y = 3

     div bx      ;ax = ax / bx = 4 / 3 = 1(quotient)
                 ;dx = 1(remainder)

```

Result

registers

	H	L
AX	00	01
BX	00	03
CX	01	0D
DX	00	01

3. X * Y / Z

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 4  ;X = 4
   mov bx, 3  ;Y = 3
   mov cx, 2  ;Z = 2

   mul bx     ;ax = ax * bx = 4 * 3 = 12
   div cx     ;ax = ax / cx = 12 / 2 = 6(quotient)
              ;dx = 0(remainder)
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 4   ;X = 4
     mov bx, 3   ;Y = 3
     mov cx, 2   ;Z = 2

     mul bx      ;ax = ax * bx = 4 * 3 = 12
     div cx      ;ax = ax / cx = 12 / 2 = 6(quotient)
                 ;dx = 0(remainder)


```

Result

registers

	H	L
AX	00	06
BX	00	03
CX	00	02
DX	00	00

Task 08

Perform the following arithmetic operations

1. 236DF * AF
2. 8A32F4D5 / C9A5
3. CA92 * BAF9
4. C2A2 * ABCD / BED

Solutions

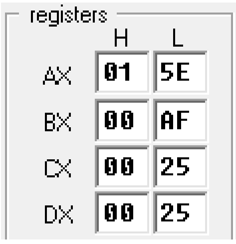
1. 236DF * AF

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data    ; Initialize data segment
   mov ds, ax
```

```
; Load the lower and upper parts of 236DFh
mov ax, 36DFh    ; Load lower 16 bits (36DF) into AX
mov dx, 2        ; Load upper part (2) into DX

mov bx, 0AFh     ; Load AF into BX

; Perform the multiplication of lower part (36DF) with AF
mul bx           ; Multiply AX by BX, result is in DX:AX (for lower part only)

; Save result of lower multiplication
mov cx, dx       ; Store DX result of 36DF * AF in CX
mov dx, 0        ; Clear DX for the next multiplication

; Multiply the upper part (2) with AF
mov ax, 2        ; Load upper part again (2) into AX
mul bx           ; Multiply AX (2) by BX (AF)

; Add the results to form the full 32-bit result
add dx, cx       ; Add saved higher part of 36DF * AF to DX
; DX:AX now contains the final 32-bit result of 236DF * AF

; Exit
```

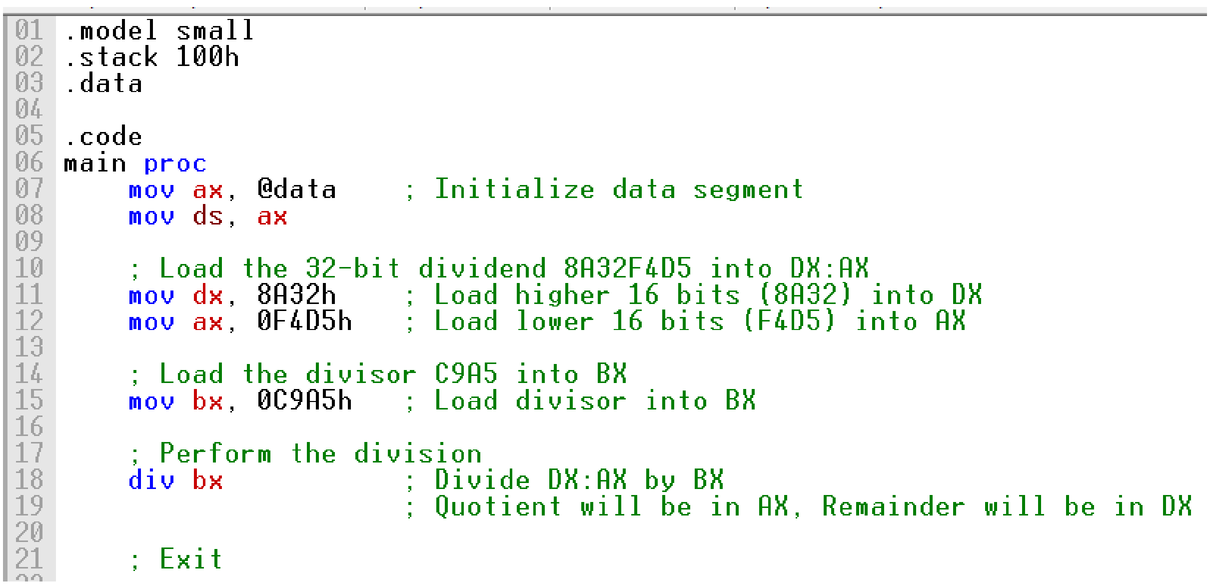
```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data    ; Initialize data segment
    mov ds, ax

    ; Load the lower and upper parts of 236DFh
    mov ax, 36DFh    ; Load lower 16 bits (36DF) into AX
    mov dx, 2        ; Load upper part (2) into DX

    mov bx, 0AFh     ; Load AF into BX

    ; Perform the multiplication of lower part (36DF) with AF
    mul bx           ; Multiply AX by BX, result is in DX:AX (for lower part only)

    ; Save result of lower multiplication
    mov cx, dx       ; Store DX result of 36DF * AF in CX
    mov dx, 0        ; Clear DX for the next multiplication

    ; Multiply the upper part (2) with AF
    mov ax, 2        ; Load upper part again (2) into AX
    mul bx           ; Multiply AX (2) by BX (AF)

    ; Add the results to form the full 32-bit result
    add dx, cx       ; Add saved higher part of 36DF * AF to DX
    ; DX:AX now contains the final 32-bit result of 236DF * AF

    ; Exit
```

Result



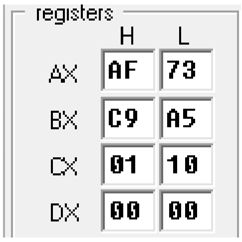
2. 8A32F4D5 / C9A5

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data    ; Initialize data segment
    mov ds, ax

    ; Load the 32-bit dividend 8A32F4D5 into DX:AX
    mov dx, 8A32h    ; Load higher 16 bits (8A32) into DX
    mov ax, 0F4D5h   ; Load lower 16 bits (F4D5) into AX

    ; Load the divisor C9A5 into BX
    mov bx, 0C9A5h   ; Load divisor into BX

    ; Perform the division
    div bx           ; Divide DX:AX by BX
                     ; Quotient will be in AX, Remainder will be in DX

    ; Exit
```



```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data     ; Initialize data segment
    mov ds, ax

    ; Load the 32-bit dividend 8A32F4D5 into DX:AX
    mov dx, 8A32h     ; Load higher 16 bits (8A32) into DX
    mov ax, 0F4D5h    ; Load lower 16 bits (F4D5) into AX

    ; Load the divisor C9A5 into BX
    mov bx, 0C9A5h    ; Load divisor into BX

    ; Perform the division
    div bx            ; Divide DX:AX by BX
                      ; Quotient will be in AX, Remainder will be in DX

    ; Exit
```

Result

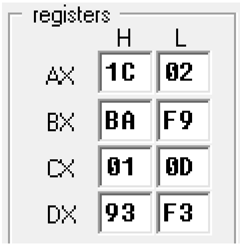


3. CA92 * BAF9

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data    ; Initialize data segment
    mov ds, ax

    ; Load the multiplicands CA92 and BAF9
    mov ax, 0CA92h   ; Load CA92 into AX
    mov bx, 0BAF9h   ; Load BAF9 into BX

    ; Perform the multiplication
    mul bx           ; Multiply AX by BX, result is in DX:AX
                     ; DX:AX now contains the 32-bit result of CA92 * BAF9

    ; Exit
```


```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data     ; Initialize data segment
    mov ds, ax

    ; Load the multiplicands CA92 and BAF9
    mov ax, 0CA92h    ; Load CA92 into AX
    mov bx, 0BAF9h    ; Load BAF9 into BX

    ; Perform the multiplication
    mul bx            ; Multiply AX by BX, result is in DX:AX
                      ; DX:AX now contains the 32-bit result of CA92 * BAF9

    ; Exit

```


Result



4. C2A2 * ABCD / BED

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data       ; Initialize data segment
    mov ds, ax

    ; Step 1: Load the multiplicands C2A2h and ABCDh
    mov ax, 0C2A2h       ; Load C2A2 into AX (lower 16 bits)
    mov bx, 0ABCDh       ; Load ABCD into BX

    ; Perform the multiplication of C2A2h * ABCDh
    mul bx               ; AX = lower 16 bits of C2A2 * ABCD, DX = upper 16 bits of result

    ; Save the result of multiplication (C2A2 * ABCD) in DX:AX
    mov cx, dx           ; Store upper 16 bits of result in CX
    mov dx, 0            ; Clear DX to prepare for division

    ; Step 2: Load the divisor BEDh
    mov bx, 0BEDh        ; Load BED into BX (divisor)

    ; Step 3: Perform the division (DX:AX / BX)
    div bx               ; DX:AX (32-bit number) divided by BX
                         ; AX = quotient, DX = remainder

    ; Exit the program
```

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data          ; Initialize data segment
    mov ds, ax

    ; Step 1: Load the multiplicands C2A2h and ABCDh
    mov ax, 0C2A2h         ; Load C2A2 into AX (lower 16 bits)
    mov bx, 0ABCDh         ; Load ABCD into BX

    ; Perform the multiplication of C2A2h * ABCDh
    mul bx                 ; AX = lower 16 bits of C2A2 * ABCD, DX = upper 16 bits of result

    ; Save the result of multiplication (C2A2 * ABCD) in DX:AX
    mov cx, dx             ; Store upper 16 bits of result in CX
    mov dx, 0              ; Clear DX to prepare for division

    ; Step 2: Load the divisor BEDh
    mov bx, 0BEDh          ; Load BED into BX (divisor)

    ; Step 3: Perform the division (DX:AX / BX)
    div bx                 ; DX:AX (32-bit number) divided by BX
                           ; AX = quotient, DX = remainder

    ; Exit the program
```

Result

registers

	H	L
AX	00	01
BX	0B	ED
CX	82	9E
DX	05	CD

Task 09

Write two examples for each combination of registers possible for the 'mov' instruction.
Hint: See the table above to see all the possible combinations.

Solution

```
.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 3
    mov bx, 4

    mov cx, ax
    mov dx, bx
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 3
     mov bx, 4

     mov cx, ax
     mov dx, bx
```

Result

registers

	H	L
AX	00	03
BX	00	04
CX	00	03
DX	00	04

Task 10

Write two examples for each combination of registers possible for the 'add' and 'sub' instructions.
Hint: See the table above to see all the possible combinations.

Solution

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 3
   mov bx, 4
   mov cx, 5
   mov dx, 6

   add ax, bx
```

```
add cx, dx
```

```
.model small
.stack 100h
.data

.code
main proc
     mov ax, @data
     mov ds, ax

     mov ax, 3
     mov bx, 4
     mov cx, 5
     mov dx, 6

     add ax, bx
     add cx, dx
```

Result

registers

	H	L
AX	00	07
BX	00	04
CX	00	0B
DX	00	06

Task 11

Perform the following arithmetic operation: (1 + 2) * (3 – 1) / 5 + 3 + 2 – (1 * 2)

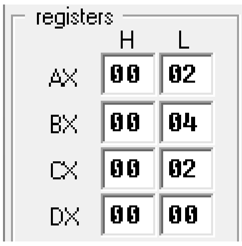
Solution

```
.model small
.stack 100h
.data

.code
main proc
   mov ax, @data
   mov ds, ax

   mov ax, 2
   inc ax        ;ax = ax + 1 = (1+2)=3
   mov bx, 3
   dec bx        ;bx = bx - 1 = (3-1)=2
```

```
mul bx        ;ax = ax * bx = (1+2)*(3-1)=6
mov bx, 5
div bx        ;ax = ax / bx = (1+2)*(3-1)/5=1
              ;mul is done before div, violating BODMAS
              ;can't use dx register to store divisor(5)
              ;since dx is used to store the remainder
mov bx, ax    ;bx = ax = (1+2)*(3-1)/5=1
mov ax, 1
mov cx, 2
mul cx        ;ax = ax * 2 = (1*2)=2
mov cx, 3
add bx, cx    ;bx = bx + cx = (1+2)*(3-1)/5+3=4
mov cx, 2
add bx, cx    ;bx = bx + cx = (1+2)*(3-1)/5+3+2=6
sub bx, ax    ;bx = bx - ax = (1+2)*(3-1)/5+3+2-(1*2)=4
```

```
;(1 + 2) * (3 - 1) / 5 + 3 + 2 - (1 * 2)

.model small
.stack 100h
.data

.code
main proc
    mov ax, @data
    mov ds, ax

    mov ax, 2
    inc ax          ;ax = ax + 1 = (1+2)=3
    mov bx, 3
    dec bx          ;bx = bx - 1 = (3-1)=2
    mul bx          ;ax = ax * bx = (1+2)*(3-1)=6
    mov bx, 5
    div bx          ;ax = ax / bx = (1+2)*(3-1)/5=1
                    ;mul is done before div, violating BODMAS
                    ;can't use dx register to store divisor(5)
                    ;since dx is used to store the remainder
    mov bx, ax      ;bx = ax = (1+2)*(3-1)/5=1
    mov ax, 1
    mov cx, 2
    mul cx          ;ax = ax * 2 = (1*2)=2
    mov cx, 3
    add bx, cx      ;bx = bx + cx = (1+2)*(3-1)/5+3=4
    mov cx, 2
    add bx, cx      ;bx = bx + cx = (1+2)*(3-1)/5+3+2=6
    sub bx, ax      ;bx = bx - ax = (1+2)*(3-1)/5+3+2-(1*2)=4
```

Result


registers
H
L
AX
00
02
BX
00
04
CX
00
02
DX
00
00