PAMANTASAN NG LUNGSOD NG MAYNILA
(University of the City of Manila)
Intramuros, Manila

MICROPROCESSOR (LECTURE)

Assembly Language Exercise

Score

Submitted by:
Santiago, Fernand D.
Sat 1:00-4:00PM / CpE-412-2

Date Submitted
21/10/2023

Submitted to:

Engr. Maria Rizette H. Sayo

1. **Explain why each of the following MOV statements are invalid:**

```
.data

bVal BYTE 100

bVal2 BYTE ?

wVal WORD 2

dVal DWORD 5

.code

        mov ds, 45
```

; This is invalid because mov instruction cannot be used to move an immediate value into a segment register.

```
        mov esi, wVal
```

; This is invalid because of size mismatch as the source operand is 16bits, while the destination operand is a register 32 bits.

```
        mov eip, dVal
```

; This is invalid because the instruction cannot be used to directly move a value into the instruction pointer (EIP).

```
        mov 25, bVal
```

; This is invalid because the destination operand of the instruction cannot be an immediate value; should be a valid memory.

```
        mov bVal2, bVal
```

; This is invalid because the instruction is used to move the data from one byte register to another byte register.

2. **Show the value of the destination after each of the following instructions executes.**

```
.data

myByte BYTE 0FFh, 0

.code

        mov al, myByte              ; Al = 0FFh

        mov ah, [myByte + 1]        ; AH = 0

        dec ah                      ; AH = FFh

        inc al                      ; AL = 0

        dec ax                      ; AX = FFFFh
```

Initially, it loads AL with the value 0FFh from myByte, making AL equal to 0FFh. It then loads AH with the second byte of myByte, which is 0, setting AH to 0. Afterward, the code decrements AH, and it wraps around to 0FFh due to underflow when decreasing from 0. Next, AL is incremented, but because it overflows, AL wraps around from 0FFh back to 0. Finally, it decrements AX, making AX equal to FFFFh, where AH is 0xFF, and AL is 0.

3. **For each of the following marked entries, show the values of the destination operand and the Sign, Zero, and Carry flags:**

```
mov ax, 00FFh

add ax, 1         ; AX = 011h       SF= 0      ZF= 0      CF= 0

sub ax, 1         ; AX = 00FFh      SF= 0      ZF= 0      CF= 0

add al, 1         ; AL = 00h        SF= 0      ZF= 1      CF= 1

mov bh, 6Ch

add bh, 95h       ; BH = 01h        SF= 0      ZF= 0      CF= 1

mov al, 2

sub al, 3         ; AL = FFh        SF= 1      ZF= 0      CF= 1
```

The mov ax, 00FFh instruction loads the AX register with the value 00FFh. The add ax, 1 instruction adds the value 1 to the AX register, resulting in a value of 0100h. The sub ax, 1 instruction subtracts the value 1 from the AX register, resulting in a value of 00FFh. The add al, 1 instruction adds the value 1 to the AL register (the lower 8 bits of the AX register), resulting in a value of 00h. The mov bh, 6Ch instruction loads the BH register with the value 6Ch. The add bh, 95h instruction adds the value 95h to the BH register, resulting in a value of 01h. The mov al, 2 instructions loads the AL register with the value 2. The sub al, 3 instruction subtracts the value 3 from the AL register, resulting in a value of FFh.

4. **What will be the value of the Overflow flag?**

```
mov al, 80h

add al, 92h          ; OF = 1

mov al, -2

add al, +127         ; OF = 0
```

In the second instruction, add al, 92h, the value 92h is added to the AL register. The result of this operation was 112h, which is greater than 255. The OF flag is set to indicate that the result of the operation is too large to be stored in the AL register. In the fourth instruction, add al, +127, the value 127 is added to the AL register. The result of this operation is 125h, which is within the range of the AL register. The OF flag is cleared.

5. **What will be the value of the Carry and Overflow flags after each operation?**

```
mov al, -128

neg al                     ; CF = 1          OF = 1


mov ax, 8000h

add ax, 2                  ; CF = 0          OF = 0


mov ax, 0

sub ax, 2                  ; CF = 1          OF = 1


mov al, -5

sub al, +125               ; CF = 1          OF =  1
```

The second instruction negates the value of the AL register. The result of this operation is too large to be stored in the register, so the Carry flag (CF) and Overflow flag (OF) are set. The fourth instruction adds the value 2 to the AX register. The result of this operation is within the range of a word-sized register, so the CF and OF flags are cleared. The sixth instruction subtracts the value 2 from the AX register. The result of this operation is too small to be stored in a word-sized register, so the CF and OF flags are set. Finally, the eighth instruction subtracts the value 125 from the AL register. The result of this operation is too small to be stored in the register, so the CF and OF flags are set.

6. **What will be the final value of AX? How many times will the loop execute?**

```
 mov ax, 6
  mov ecx, 4
L1:
 inc ax
  loop L1
```

Final value: **10** ; the value of ECX is initially 4. The inc ax instruction increments the value of AX to 7. The loop L1 instruction decrements the value of ECX to 3 and jumps to the instruction labeled L1. This process repeats until the value of ECX is 0. At the end of the loop, the value of AX will be 10.

```
 mov ecx, 0
X2:
 inc ax
  loop X2
```

Loop execution: **Will not execute** ; The loop will not execute any times because the value of ECX is 0. When the loop instruction is executed, it will decrement the value of ECX to -1. However, the loop instruction will not jump to the instruction labeled X2 because the value of ECX is now less than 0. The execution of the program will terminate.