Arquitetura e Organização de Computadores II
Lista 2

1)Mostrar o valor do operando destino após a execução da instrução.

```
.data
myByte BYTE 0, 0FFh
.code
mov al,myByte    ; AL =
mov ah,[myByte+1]      ; AH =
dec ah                 ; AH =
inc al                 ; AL =
dec ax                 ; AX =
```

2)Escrever o código que traduz a expressão Rval = Xval - (-Yval + Zval) em linguagem Assembly, usando instruções add, sub, neg e mov. As variáveis são definidas pelas sentenças:

```
Xval SDWORD ?
Yval SDWORD ?
Zval  SDWORD ?
```

3)Para cada uma das instruções aritméticas do programa seguinte, mostrar os valores do operando destino e os flags de Sign, Zero e Carry:

```
mov ax,0FF00h
add ax,100h          ; AX=          SF=    ZF=    CF=
sub ax,1             ; AX=          SF=    ZF=    CF=
add al,1             ; AL=          SF=    ZF=    CF=
mov bh,6Ch
add bh,95h           ; BH=          SF=    ZF=    CF=
mov al,2
sub al,3             ; AL=          SF=    ZF=    CF=
```

4) Quais são os flags após as operações

```
mov bl,-128
neg  bl              ; CF =    OF =
mov ax,8000h
add  ax,9000h        ; CF =     OF =
mov al,0
sub   al,1           ; CF =     OF =
mov bl,-4
sub  bl,+125         ; CF =    OF =
```

5) Completar o preenchimento dos valores de **esi**, nos comentários abaixo:

```
data
bVal BYTE 5 DUP (?)
wVal WORD  ?
dVal DWORD ?
.code
mov esi,OFFSET bVal    ; esi = 00404000
mov esi,OFFSET wVal    ; esi =
mov esi,OFFSET dVal    ; esi =
```

6) Escrever os valores dos operandos destino:

```
.data
varB    BYTE   44h,32h,03h,0Ah,
                  0Dh
varW    WORD  0FF43h,3302h
        BYTE          ?
varD    DWORD          11223344h
.code
mov     ax, WORD PTR [varB+3]          ; ax   =
mov     bl, BYTE PTR varD              ; bl   =
mov     bl, BYTE PTR [varW+3]          ; bl   =
mov     ax, WORD PTR [varD+2]          ; ax   =
mov     ax, DWORD PTR varW             ; eax  =.
mov     al,TYPE varB                   ; al   =
mov     al,TYPE varD                   ; al   =
mov     eax, LENGTHOF varB             ; eax  =
mov     ebx, LENGTHOF varW             ; ebx  =
mov     eax, SIZEOF varB               ; eax  =
mov     ebx, SIZEOF varW               ; ebx  =
mov     ecx, SIZEOF varD               ; ecx  =
```

7) Escrever os valores do operando destino:

```
.data
valorD   LABEL DWORD
valorW   LABEL WORD
valorB   BYTE 10h,20h,30h,40h
.code
mov     eax, valorD       ;  eax =
mov     ax, valorW        ;   ax  =
mov     al, valorB        ;   al  =
```

8) Escrever um código que carrega em **esi** o endereço de **vetor1** e faz a soma dos 3 DWORD's inicializados pela sentença abaixo, com o resultado em **eax**:

```
vetor1 DWORD 10, 20, 30
```

9) Escrever um código que faz a soma dos 3 DWORD's do exercício anterior, usando endereçamento indexado, com o resultado em **eax**.

Nota: endereçamento indexado – **esi** contem o índice, ou seja endereço = **vetor1 + esi** ou endereço= **vetor1[esi]**.

10) Escrever um código que use o endereçamento indexado com escala (**type vetor1**) para a mesma soma dos exercícios anteriores.

11) Alterar o exercício 5, usando a declaração da variável **pvetor1** inicializado como ponteiro de **vetor1.**

12) Reescrever o programa abaixo usando endereçamento indireto

```
        .data
        source  BYTE  "This is the source string",0
        target  BYTE  SIZEOF source DUP(0)
        .code
        mov  esi,0                      ; index register
        mov  ecx,SIZEOF source          ; loop counter
L1:     mov  al,source[esi]             ; get char from source
        mov  target[esi],al             ; store it in the target
        inc  esi                        ; move to next character
        loop L1                         ; repeat for entire string
```