# Arquitetura e Organização de Computadores II
## Lista 3

Os exercícios abaixo são relativos à linguagem Assembly da arquitetura IA-32

1) 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

2) 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]**.*

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

4) Alterar o exercício 1, usando a declaração da variável **pvetor1** inicializado como ponteiro de **vetor1.**

5) 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
```

6) Escrever um programa que faz a leitura de quatro dwords (vetord) da memória e os escreve na tela em hexadecimal, separados por um espaço. Usar writehex, com argumento em eax, e writechar, com argumento em al.
   .data
   vetord  DWORD  1, 2, 3, 4
   code

7) Escrever um programa que limpa a tela, escreve os quatro dwords do exercício 1, porém introduzindo um atraso de 5 segundos entre uma escrita de dword.

8) Dada  uma mensagem na área de dados, escrever o código que escreve essa mensagem na tela.
   .data
   Mensagem byte ‘Exemplo de mensagem numero 1!’,0dh, 0ah,0
   .code

9) Escrever um programa que escreve um vetor de 4  inteiros definidos  na memória em  decimal e hexadecimal, separados por vírgulas, conforme o formato abaixo:
   Vetor em decimal:  1,  2,  3,  4
   Vetor em hexadecimal: 00000001h, 00000002h, 00000003h, 00000004h

10) Mostrar o maior número inteiro sem sinal de 32 bits na tela,  em decimal, hexadecimal e binário.

11) Escrever um programa para escrever na memória uma seqüência de até 20 caracteres digitados no teclado, seguidos de enter.

12) Acrescentar no programa do exercício anterior, o código para mostrar o conteúdo de memória relativo à sequência de caracteres digitados.

13) Escrever um programa para gerar e mostrar 10 números pseudoaleatorios em inteiro com sinal na tela, separados por espaço. Usar instrução loop.

14) Escrever um programa que atribui valores inteiros a EAX, EBX, ECX, EDX, ESI e EDI e em seguida, usa PUSHAD para carregar os registradores de uso geral na pilha. Usando um loop, o programa deve fazer o pop de cada inteiro da pilha e mEscrever 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

15) 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]**.*

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

17) Alterar o exercício 1 usando a declaração da variável **pvetor1** inicializado como ponteiro de **vetor1.**

18) 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
```

19) Escrever um programa que faz a leitura de quatro dwords (vetord) da memória e os escreve na tela em hexadecimal, separados por um espaço. Usar writehex, com argumento em eax, e writechar, com argumento em al.

```
.data
vetord  DWORD  1, 2, 3, 4
.code
```

20) Escrever um programa que limpa a tela, escreve os quatro dwords do exercício 20, porém introduzindo um atraso de 5 segundos entre uma escrita de dword.

21) Dada uma mensagem na área de dados, escrever o código que escreve essa mensagem na tela.

```
.data
Mensagem byte ‘Exemplo de mensagem numero 1!’,0dh, 0ah,0
.code
```

22) Escrever um programa que dada uma cadeia terminada por um valor zero, na memória, seja invertida, usando instruções de pilha.

23) Modificar o programa anterior tal que seja introduzida uma lista de inteiros de 32 bits pelo usuário (ReadInt), faça a inversão desses inteiros, mostre os inteiros em ordem inversa e o seu somatório na tela.

24) Escrever o programa anterior usando o procedimento ArraySum (abaixo) de soma de inteiros:

```
ArraySum PROC
; Recebe: ESI aponta a um vetor de doublewords,
;   ECX = número de elementos do vetor.
; Retorna: EAX = sum
;-----------------------------------------------------
mov eax,0       ; set the sum to zero
L1:     add eax,[esi]   ; add each integer to sum
add esi,4       ; point to next integer
loop L1         ; repeat for array size
ret
ArraySum END
```