Operações aritméticas e lógicas

1. Indicar o conteúdo dos registos usados e da *flag* de *carry* (CF) após a execução de cada fragmento de código.

a)
```
mov   bl, 1
xor   ax, ax
add   ax, -1
mov   al, bl
add   ah, al
```

b)
```
mov   eax, 66666666H
mov   ebx, 0F000000FH
and   eax, ebx
xor   eax, ebx
or    eax, 66666666H
```

c)
```
mov   al, 25
sar   al, 1
shr   al, 1
neg   al
ror   al, 2
rol   al, 2
rcr   al, 2
```

d)
```
mov   al, 5
xor   bl, bl
shr   al, 1
rcr   bl, 1
ror   bl, 8
rol   al, 2
```

e)
```
mov   bx, 0BEEFH
add   bx, 8000H
sbb   bx, 3EEEH
adc   bx, bx
```

f)
```
mov   ebx, 7FFFFFFFH
mov   ecx, 7FFFFFF5H
add   ecx, 0FH
cmovo ecx, ebx
```

2. Implementar um programa que determine a posição (peso) do bit 1 mais significativo de um valor não nulo do tipo `dword`. Por exemplo, se o valor for `00000009H` o resultado é 3.

3. Considere o seguinte fragmento de um programa:

```
mov       ebx, eax
sub       ebx, 1
and       ebx, eax
```

a) Determine o valor de EBX após a execução do fragmento de código considerando que antes da execução o valor de EAX é:
i. 16; **ii.** 18.

b) Identifique o que faz o código relativamente ao valor de EAX.

4. Implementar um programa em *assembly* IA-32 para calcular o valor das expressões seguintes, assumindo que os operandos e os resultados intermédios são inteiros de 32 bits com sinal.

a) $(a + b) - 123$ **b)** $4 \times (a - b) - c$ **c)** $(a + b)/5$

5. Escrever um fragmento de código para multiplicar dois valores `val1` e `val2`, com 16 bits, devolvendo o resultado em EAX.

6. Escrever um fragmento de código para calcular o produto de EAX por 18:

a) Usando instruções de multiplicação;

b) Sem usar instruções de multiplicação nem ciclos.

7. Escrever um programa para calcular o produto interno de dois vetores de números inteiros de 32 bits (do tipo `SDWORD`). Caso ocorra *overflow*, o programa deve assinalar essa situação.

8. Escrever um programa para determinar os valores máximo e mínimo de:

a) Uma sequência de elementos do tipo `word`;

b) Uma sequência de elementos do tipo `sword`.

9. Apresentar o código *assembly* que realiza os testes indicados abaixo. Considerar apenas números sem sinal.

a)
```
if ((AL>AH) and (BL>BH)) or  (AH<CL)
   ECX = ECX + 1
```

b)
```
if ((AL>AH) or (BL>BH)) and (AH<CL)
   ECX = ECX - 1
```

10. Considere uma sequência `vec` de números inteiros de 32 bits (com sinal). O número de elementos da sequência é dado por `vecSize`, uma variável global do tipo WORD. Implemente um fragmento de código *assembly* IA-32 que:

a) Determina quantos elementos da sequência são iguais, em valor absoluto, ao conteúdo de EAX (um número positivo). O resultado deve ficar guardado no registo ECX;

b) Substitua por zero os elementos da sequência com valor absoluto inferior a `0FFH`;

c) Conte quantos elementos da sequência pertencem ao intervalo $[a; b]$ ($a \leq x \leq b$). Assuma que os números a e b estão contidos nos registos EAX e EBX, respetivamente, e que o resultado fica no registo ECX.

11. Escrever um programa que calcula o valor médio (arredondado às unidades) de uma sequência de valores do tipo `DWORD`. [Não usar instruções de vírgula flutuante.]

a) A primeira versão assume que a soma dos valores da sequência não produz *overflow*.

b) A segunda versão deve funcionar corretamente para qualquer sequência, indicando *overflow* se existir (e interrompendo os cálculos nesse caso).

12. A representação BCD (Binary-Coded Decimal) representa cada dígito decimal por um grupo de 4 bits. Escrever e testar um programa que converte entre a representação em cadeia de carateres com 8 dígitos e a representação BCD compactada em 32 bits (`DWORD`).

Exemplo: A cadeia de carateres `'45187023'` corresponde em BCD ao valor (representado em binário) `0100 0101 0001 1000 0111 0000 0010 0011`.
(Nota: interpretado como número binário puro, este valor seria 1159229475_{10}.)