# Exercícios Práticos de Arquitectura de Computadores
## 1ª Série

1) Faça uma subrotina que some três números de 32 bits, e um programa que a utilize para adicionar três números.

```
SOMA3 ( n1 n2 n3 -- soma )
```

2) Usando a subrotina anterior faça outra que some seis números de 32 bits, e um programa que a utilize.

```
SOMA6 ( n1 n2 n3 n4 n5 n6 -- soma )
```

3) Faça uma subrotina que a partir de dois números, determine o maior deles, deixando apenas na stack o maior. Suponha que os dois números possuem sinal.

```
MAIOR2 ( n1 n2 -- maior )
```

4) Usando a rotina anterior faça uma rotina que determine o maior de quatro números.

```
MAIOR4 ( n1 n2 n3 n4 -- maior )
```

5) Faça um programa que a partir das variáveis de 32 bits VAR1 e VAR2 calcule a sua soma e coloque o resultado em VAR3 também de 32 bits.

6) Faça uma subrotina que leia um byte da memória.

```
LERB ( ender -- byte )
```

7) Faça uma subrotina que leia uma word da memória.

```
LERW ( ender -- word )
```

8) Faça uma subrotina para escrever um byte na memória.

```
ESCRB ( byte ender -- )
```

9) Faça uma subrotina para escrever uma word na memória.

```
ESCRW ( word ender -- )
```

10) Faça um programa que a partir das variáveis de 8 bits VAR1 e VAR2 calcule a sua soma e coloque o resultado em VAR3 também de 8 bits.

11) Faça um programa que a partir das variáveis de 16 bits VAR1 e VAR2 calcule a sua soma e coloque o resultado em VAR3 também de 16 bits.

12) Faça uma rotina que a partir de quatro números de 32 bits colocados na stack os some como se fossem dois números de 64 bits.

```
SOMA64 ( n1_low n1_high n2_low n2_high -- n3_low n3_high)
```

13) Faça uma rotina que dados 4 bytes junte esses 4 bytes num número de 32 bits pela ordem correcta.

```
JUNTA4B  ( b0 b1 b2 b3 -- n )
```

14) Faça uma rotina que efectue a divisão sem sinal de um número de 64 bits por um número de 32 bits. Nota: esta rotina como as outras todas, deve na medida do possível, deixar intactos os conteúdos dos registos do processador.

```
MYDIVU ( n1_low n1_high n2 -- quociente resto )
```

15) Faça uma rotina que efectue a multiplicação sem sinal de dois números de 32 bits dando como resultado um número de 64 bits.

```
MYMULU ( n1 n2 -- n3_low n3_high )
```

16) Faça uma rotina que efectue a multiplicação com sinal de dois números de 32 bits dando como resultado um número de 64 bits.

```
MYMULS ( n1 n2 -- n3_low n3_high )
```

17) Faça uma rotina para o cálculo de uma promoção do tipo "Leve 3 pague 2". A partir do número de unidades compradas, e do preço unitário calcule a quantia total a pagar.

```
L3P2 ( num pr_un -- quantia )
```

18) Faça uma rotina que a partir do preço de um produto e do escalão de IVA a pagar calcule o preço de venda ao público do produto. Considere os seguintes escalões de IVA: 0=0%, 1=7%, 2=13% e 3=20%. Para efectuar os cálculos pode usar aproximações do estilo 1,17=1170/1000, o que lhe permitirá efectuar os cálculos sem operações de vírgula flutuante.

```
CALCPVP ( preco escalao -- pvp )
```

19) Faça uma subrotina que transforme o caracter no topo da stack numa maiúscula se este fôr uma minúscula.

```
UPPER ( b1 -- b2 )
```

20) Faça uma subrotina que transforme o caracter no topo da stack numa minúscula se este fôr uma maiúscula.

```
LOWER ( b1 -- b2 )
```

21) Faça uma subrotina que verifique se o número de 32 bits sem sinal no topo da stack é ou não uma capicua escrito em hexadecimal. Se sim a rotina deixa -1 na stack, senão deixa 0.

```
CAPIH ( n1 -- n2 )
```

22) Faça uma subrotina que verifique se o número de 32 bits sem sinal no topo da stack é ou não uma capicua escrito em binário. Se sim a rotina deixa -1 na stack, senão deixa 0.

```
CAPIB ( n1 -- n2 )
```

23) Dado um vector de números de 32 bits faça uma rotina que determine o maior dos seus elementos.

```
MAIOR_V ( ender num_el -- maior )
```

24) Dado um vector de bytes faça uma rotina que determine a média dos seus elementos, supondo que estes são números sem sinal.

```
MEDIA_B ( ender num_el -- media )
```

25) Dado um vector de bytes faça uma rotina que determine a média dos seus elementos, spondo que estes são números com sinal.

```
MEDIA_BS ( ender num_el -- media )
```

26) Faça uma rotina que imprime no terminal uma string colocada na memória. A rotina recebe como parâmetros o endereço do primeiro caracter, e o número de caracteres a imprimir.

```
WRITESTRING ( ender num -- )
```

27) Faça uma rotina que imprime no terminal uma string colocada na memória. A rotina recebe como parâmetros o endereço da string. A string possui no seu início um número de 32 bits que contém o número de caracteres da string.

```
WRITESTRINGN ( ender -- )
```

28) Faça uma rotina que imprime no terminal uma string colocada na memória. A rotina recebe como parâmetros o endereço da string. A string possui no seu fim um byte com o valor 0 (zero).

```
WRITESTRING0 ( ender -- )
```

29) Faça uma rotina que lê do "terminal" uma string terminada por CR. A rotina deve escrever na memória os caracteres lidos a partir de um determinado endereço, e guardar no fim da string o byte 0 (zero) deixando na stack o número de caracteres lidos. O "Carriage Return" não deve ser guardado na memória nem deve ser contado.

```
BAD_READSTRING ( ender -- num )
```

30) A rotina anterior está errada porque não possui um limite para o número de caracteres lidos. Assim o utilizador pode escrever caracteres "a mais", ultrapassando os limites para a string. Faça uma nova versão da rotina anterior na qual existe como parâmetro um número máximo de caracteres a ler.

```
READSTRING0 ( ender num_max -- num )
```