```
Para cada questão, escreva funções em C e/ou sub-rotinas na linguagem Assembly do MSP430. Reaproveite funções e sub-rotinas de uma questão

1. (a) Escreva uma função em C que calcule a raiz quadrada 'x' de uma variável 'S' do tipo float, utilizando o seguinte algoritmo: após 'n+

        x(n+1) = (x(n) + S/x(n))/2


O protótipo da função é:

        unsigned int Raiz_Quadrada(unsigned int S);

unsigned int Raiz_Quadrada(unsigned int S)
    {
        float x;
        int n;
        x = S/2;
        for(n=0;n<=100;n++)
        {
            x = (x + (S/x))/2;
        }
        return x;
        }

(b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. A variável 'S' é fornecida pelo registrador R15, e a raiz quadrada de

Raiz:
clr.w R9
Inicio:
clr.w R12
mov.w R15, R8
mov.w #2, R14   ;R14 é o divisor
call #Divisao
Continua:
mov.w R15, R8
mov.w R10, R13  ;x = S/2
mov.w R13, R14  ;Divisor = x
call #Divisao
add.w R13, R10  ;x + S/x
mov.w #2, R14   ;Divisor = 2
mov.w R10, R8
call #Divisao
jmp Loop


Divisao:
cmp #1, R14
jeq Condicao1
mov.w R14, R11
sub.w R14, R8
add.w #1, R12
Comparar:
dec.w R11
cmp R11, R8
jeq Resultado
cmp #0, R11
jeq Divisao
```

```
jmp Comparar
Condicao1
mov.w R8, R12
Resultado:
mov.w R12, R10
clr.w R14
clr.w R12
clr.w R11
clr.w R8
ret
Loop:
inc.w R9
cmp #64, R9
jne Continua
mov.w R10, R15


2. (a) Escreva uma função em C que calcule 'x' elevado à 'N'-ésima potência, seguindo o seguinte protótipo:

        int Potencia(int x, int N)
        {
            int y;
                if(N==1) return x;
                if(N==0) return 1;
                y = x*Potencia(x, --N);
                return y;
        }

(b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. 'x' e 'n' são fornecidos através dos registradores R15 e R14, respect

Potencia:
cmp #1, R14
jeq FIM
cmp #0, R14
jeq COND
mov.w R15, R12
mov.w R15, R11
clr.w R15
mov.w R12, R13
Multiplicacao:
add.w R12, R15
cmp.w #1, R13
dec.w R13
jne Multiplicacao
mov.w R11, R13
dec.w R14
cmp #1, R14
mov.w R15, R12
jeq FIM
clr.w R15
jmp Multiplicacao
COND:
mov.w #1, R15
FIM:


3. Escreva uma sub-rotina na linguagem Assembly do MSP430 que calcula a divisão de 'a' por 'b', onde 'a', 'b' e o valor de saída são inteir

clr.w R12
Divisao:
mov.w R14, R11
sub.w R14, R15
add.w #1, R12
Comparar:
dec.w R11
cmp R11, R15
jeq Resultado
cmp #0, R11
jeq Divisao
jmp Comparar
Resultado:
mov.w R12, R15


```

```
4. Escreva uma sub-rotina na linguagem Assembly do MSP430 que calcula o resto da divisão de 'a' por 'b', onde 'a', 'b' e o valor de saída s

clr.w R12
Divisao:
mov.w R14, R11
sub.w R14, R15
add.w #1, R12
Comparar:
dec.w R11
cmp R11, R15
jeq Resultado
cmp #0, R11
jeq Divisao
jmp Comparar
Resultado:
mov.w R12, R15
Resto:
mov.w R11, R15


5. (a) Escreva uma função em C que indica a primalidade de uma variável inteira sem sinal, retornando o valor 1 se o número for primo, e 0,

        int Primalidade(unsigned int x);


int Primalidade(unsigned int x)
{
    int i, v;
    for(i=2;i<x;i++)
    {
        v = x%i;
        if(v == 0)
        {
            return 0;
        }
    }
    return 1;

}

(b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. A variável de entrada é fornecida pelo registrador R15, e o valor de

cmp #2, R15
jeq PRIMO
cmp #1, R15
jeq PRIMO
mov.w #2, R14
mov.w R15, R13  ;Move o valor do numero a ser testado para um registrador que será usado na verificação
clr.w R12
Divisao:
mov.w R14, R11
sub.w R14, R15
add.w #1, R12
Comparar:
dec.w R11
cmp R11, R15
jeq Verificador
cmp #0, R11
jeq Divisao
jmp Comparar
Verificador:
cmp #0, R11     ;Compara o resto com 0, se for igual o numero possui um divisor portanto n é primo
jeq NPRIMO
dec.w R13       ;Subtrai um do reg de verificação pq o divisor n pode ser igual ao dividendo
cmp R14, R13    ;Compara o divisor com o R13 e se não for igual reinica o processo aumentando o divisor
jeq PRIMO
inc.w R13
mov.w R13, R15
clr.w R12
inc.w R14
jmp Divisao
NPRIMO:
mov.w #0, R15
```

```
jmp FIM
PRIMO:
mov.w #1, R15
FIM:
```

6. Escreva uma função em C que calcula o duplo fatorial de n, representado por n!!. Se n for ímpar, n!! = 1*3*5*...*n, e se n for par, n!!
O protótipo da função é:

```
        unsigned long long DuploFatorial(unsigned long long n);

        unsigned long long DuploFatorial(unsigned long long n)
{

    long long int s, i;
    s = 1;
    for(i=1;i<=n;i++)
    {
        if(n%2 == 0)
        {
            if(i%2 == 0)
            {
                s = i*s;
            }
        }
        if(n%2 == 1)
        {
            if(i%2 == 1)
            {
                s = i*s;
            }
        }

    }
    n = s;
    return n;

}
```

7. (a) Escreva uma função em C que calcula a função exponencial da seguinte forma:

Considere o cálculo até o termo n = 20. O protótipo da função é double ExpTaylor(double x);

```
double ExpTaylor (double x)
{
    // e^x = (x^i)/f
    double e, t, s;
    double n, i, f;
    f = 1;
    s = 0;
    for(i=0;i<=20;i++)
    {
        if(i == 0)
        {
            f = 1;
            e = pow(x, i);
            t = e/f;
            s = t + s;
        }else
        {
            f = i*f;
            e = pow(x, i);
            t = e/f;
            s = t + s;
        }
    }
    x = s;
    return x;
}
```

(b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430, mas considere que os valores de entrada e de saída são inteiros de 16

```
;O programa não da conta de fazer mais do que 7 iterações, impossibilitando um resultado preciso.
```

```
Exponencial:
mov.w R15, R5   ;Entrada que n sera mudada
clr.w R8        ;R8 = i
mov.w R8, R6    ;R6 = i incrementador
clr.w R12
clr.w R4
Inicio:
mov.w R6, R8
mov.w R8, R14
call #Potencia
clr.w R14
mov.w R8, R13   ;R13 = Decrementador do fatorial
call #Fatorial
mov.w R8 , R14  ;R14 = Divisor
mov.w R11, R8   ;R8 = Dividendo
call #Divisao
inc.w R6        ;incrementa i
mov.w R10, R9   ;Variavel para realizar a soma dos termos
add.w R9, R4    ;Resultado final
cmp #7, R6
jeq ACABOU
jmp Inicio


Divisao:
cmp R14, R8
jl Condi1
cmp #1, R14
jeq Condicao1
mov.w R14, R11
sub.w R14, R8
add.w #1, R12
Comparar:
dec.w R11
cmp R11, R8
jeq Resultado
cmp #0, R11
jeq Divisao
jmp Comparar
Condicao1:
mov.w R8, R12
jmp Resultado
Condi1:
mov.w #0, R12
jmp Resultado
Resultado:
mov.w R12, R10  ;Resultado dado em R10
clr.w R14
clr.w R12
clr.w R11
clr.w R8
ret



Potencia:
mov.w R5, R15
cmp #1, R14
jeq FIM
cmp #0, R14
jeq COND
mov.w R15, R12
mov.w R15, R11
clr.w R15
mov.w R12, R13
Multiplicacao:
add.w R12, R15
cmp.w #1, R13
dec.w R13
jne Multiplicacao
mov.w R11, R13
dec.w R14
cmp #1, R14
mov.w R15, R12
```

```
jeq FIM
clr.w R15
jmp Multiplicacao
COND:
mov.w #1, R15
FIM:
mov.w  R15, R11 ;R10 = Resposta
clr.w R12
clr.w R13
clr.w R14
ret

Fatorial:
mov.w R6, R7
clr.w R10       ;soma
clr.w R12       ;controlador do loop de somas
clr.w R14       ;serve pra atribuir o valor da soma passada ao novo somador
cmp #0, R7
jeq Cond1
cmp #1, R7
jeq Cond1
cmp #2, R7
jeq Cond2
mov.w R7, R13   ; controlador do numero de somas
Loop1:
dec.w R13
cmp #1, R13
jeq FM
cmp #0, R14
jne Igualador
Soma1:
add.w R7, R10
Loop2:
inc.w R12
cmp.w R12, R13
jeq Loop1
inc.w R14
jne Soma1
Cond1:
mov.w #1, R10
jmp FM
Cond2:
mov.w #2, R10
jmp FM
Igualador:
mov.w R10, R7
mov.w #1, R12
jmp Soma1
FM:
mov.w R10, R8
clr.w R13
clr.w R12
clr.w R14
clr.w R10
ret

ACABOU:
mov.w R4, R15


8. Escreva uma sub-rotina na linguagem Assembly do MSP430 que indica se um vetor esta ordenado de forma decrescente. Por exemplo:
[5 4 3 2 1] e [90 23 20 10] estão ordenados de forma decrescente.
[1 2 3 4 5] e [1 2 3 2] não estão.
O primeiro endereço do vetor é fornecido pelo registrador R15, e o tamanho do vetor é fornecido pelo registrador R14. A saída deverá ser fo

Decrescente:
dec.w R14
rla R15
cmp 1(R15), 0(R15)
jl Crescente
cmp #1, R14
jeq Decre
jne Decrescente
Crescente:
```

```
mov.w #0, R15
jmp FIM
Decre:
mov.w #1, R15
jmp FIM
FIM:


9. Escreva uma sub-rotina na linguagem Assembly do MSP430 que calcula o produto escalar de dois vetores, 'a' e 'b':

O primeiro endereço do vetor 'a' deverá ser passado através do registrador R15, o primeiro endereço do vetor 'b' deverá ser passado através

10. (a) Escreva uma função em C que indica se um vetor é palíndromo. Por exemplo:
        [1 2 3 2 1] e [0 10 20 20 10 0] são palíndromos.
        [5 4 3 2 1] e [1 2 3 2] não são.
Se o vetor for palíndromo, retorne o valor 1. Caso contrário, retorne o valor 0. O protótipo da função é:

        int Palindromo(int vetor[ ], int tamanho);

int Palindromo(int vetor[], int tamanho)
{
    int i, t, s, v;
    s = 0;
    t = 0;
    if (tamanho%2 == 0)
    {
        for(i=0;i<=tamanho-1;i++)
        {
            t = vetor[i] - vetor[tamanho-i-1];
            s = t + s;
            t = 0;
        }
    }else if(tamanho%2 == 1)
    {
        v =(tamanho/2);
        for(i=0;i<=v;i++)
        {
            t = vetor[i] - vetor[tamanho-i-1];
            s = t + s;
            t = 0;
        }
    }
    if(s == 0)
    {
        return 1;
    }else
    {
        return 0;
    }
}



(b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. O endereço do vetor de entrada é dado pelo registrador R15, o tamanho
```