Aluno: Mateus Alves da Rocha **Matrícula:** 11/0132661

Data: 31/03/2017

Para cada questão, escreva funções em C e/ou sub-rotinas na linguagem Assembly do MSP430. Reaproveite funções e sub-rotinas de uma questão em outra, se assim desejar. Leve em consideração que as sub-rotinas são utilizadas em um código maior, portanto utilize adequadamente os registradores R4 a R11. As instruções da linguagem Assembly do MSP430 se encontram ao final deste texto.

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+1' iterações, a raiz quadrada de 'S' é dada por

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

O protótipo da função é:

```
unsigned int Raiz_Quadrada(unsigned int S);
```

```
unsigned int Raiz_Quadrada(unsigned int S){

    int t=1;
    float x = S/2;

    while (t!=0)
    {
```

```
            float x_anterior = x;
        x= (x_anterior + S/x_anterior)/2;

            float aux = x-x_anterior;

            if (aux >= 0.0)
                    if (aux < 0.001)
                    t=0;
                else
                    if (-aux < 0.001)
                    t=0;
        }
        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 'S' (ou seja, a variável 'x') é fornecida pelo registrador R15 também.

```
Raiz_Quadrada:
    Mov.w R15, R13          ; R13 = R15 = S
    RRA.w R13               ; R13 = S/2;
While_loop:
    Mov.w R13, R12          ; R12= x_anterior = S/2;
    PUSH R12                ; Guarda R12 na pilha
    Call #Div_unsigned      ; chama a função para calcular S/x_anterior
```

```
        Add.w R12, R15;         ; soma o resultado da divisão com R12
        RRA.W R15;              ; resultado da soma dividido por 2
        POP R12                 ; recupera R12 da pilha
        Mov.w R15, R14;         ; R14 recebe o resultado
        Sub.w R12, R14          ; R14 = X – X_anterior
        TST R14                 ; R14 = 0?
        Jeq Exit_loop
        Jmp while_loop
Exit_loop:


Div_unsigned:
        Clr.w R11           ; i=0
For_div:
        cmp R15, R12        ; compara R15 = S com R12 = x_anterior
        jl End_for_div      ; sai do for
        sub R12, R15        ; R15 =  R15 – R12
        inc R13             ; Incrementa R13
        Jmp    For_div      ; repete o loop
End_for_div
        Mov R13, R15        ; retorna o valor final da divisao para R15
        ret
```

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

```
        #include <stdio.h>
```

```
#include <stdlib.h>



unsigned int mult_unsigned (unsigned int a, unsigned int b){

   unsigned int c = 0;
   int i = 0;

   for (i=1; i<=b; i++)
   {
      c += a;
   }
   return c;
}

unsigned int div_signed (unsigned int a, unsigned int b){

   unsigned int i;
   for (i =0; a>=b; i++)
   {
      a = a-b;
   }
   return i;
}
```

```
int potencia(int x, int N){

   int sign=0;
   int p=1;

   if(N < 0)
   {
      N = -N;
      sign = 1;

   }
   while ( N>0)
   {
      p = mult_unsigned (x,p);
      N--;
   }
   if(sign ==1)
      return div_signed(1, p);
      else
         return p;

}
```

```
int main ()
{
   int S = 2;
   int a = 3;
        float x ;

        x = potencia (2,3);
        printf("%.2f\n", x);

        return(0);

}
```

(b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. 'x' e 'n' são fornecidos através dos registradores R15 e R14, respectivamente, e a saída deverá ser fornecida no registrador R15.

```
mult_unsigned:    clr R13              ; c=0
                  mov #1, R12          ; i=1
for_mult:         cmp R12, R14
                  jl end_for_mult
                  add R15, R13
                  inc R12
                  jmp for_mult
end_for_mult:     mov R13, R15
                  ret
```

```
div_unsigned:   clr R13                 ; i=0
for_div:        cmp R14, R15
                jl end_for_div
                sub  R14, R15
                inc R13
                jmp for_div
end_for_div:    mov R13, R15
                ret
potencia:       push R4
                push R5
                clr R4                  ; sign=0
                mov #1, R5              ; p=1
                cmp #1, R14
                jl while_pot
                inv R14
                inc R14
                mov #1, R4
while_pot:      tst R14
                jz while_pot_end
                push R15
                push R14
                mov R5, R14
                call mult_unsigned
```

```
                    mov R15, R5
                    pop R14
                    pop R15
                    dec R14
                    jmp while_pot
while_pot_end:  cmp #1, R4
                    jne pot_end
                    mov #1, R15
                    mov R5, R14
                    call div_unsigned
                    pop R5
                    pop R4
                    ret
pot_end:        mov R5, R15
                    pop R5
                    pop R4
                    ret
```

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 inteiros de 16 bits. 'a' e 'b' são fornecidos através dos registradores R15 e R14, respectivamente, e a saída deverá ser fornecida através do registrador R15.

```
Divisao_a_b:
                    Clr R13         ; limpa R13
                    Rla R15    ; se R15(a) for negativo, o carry sera 1
                    JC Comp2_de_A
```

```
                    Rla R14  ; se R14(b) for negativo, o carry sera 1
                    JC Comp2_de_B
                    Call divisão_unsigned
                    Ret

Comp2_de_A:         ; a<0
                    INV     R15
                    INC     R15
                    Mov #1, R13  ; Variavel para sinalizar que houve o
                    complemento de 2
                    Ret

Comp2_de_B:         ; b<0
                    INV     R14
                    INC     R14
                    Mov #1, R13  ; Variavel para sinalizar que houve o
                    complemento de 2
                    Ret

divisao_unsigned:
                    Clr R12         ;  limpa R12
                    CMP R14,R15           ; R15 = a, R14 = b
                    JGE divisao_sub    ; if (a >= b)
                     Mov #0, R15            ;retorna zero caso A seja
                     menor que b
                    RET
```

```
divisao_sub:
                    Sub  R14,R15          ; R15 = R15 - R14
                    INC R12 ;
                    Call  divisao_unsigned
                    ADD.W R12,R15
                    RET
```

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 são inteiros de 16 bits. 'a' e 'b' são fornecidos através dos registradores R15 e R14, respectivamente, e a saída deverá ser fornecida através do registrador R15.

```
rest_of_division:    cmp R14, R15
                    jl rest_end                    ;    R15    <    R14
termina o laço
                    sub R14, R15                   ; R15 = R15 - R14
                    jmp rest_of_division       ; repete o laço
rest_end:                ret
```

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, caso contrário. Siga o seguinte protótipo:

```
int Primalidade(unsigned int x);'
```

```
int rest_of_division (int a, int b){
	//a/b

	while (a >= b)
	{
		a -= b;
	}
	return a;
}
int Primalidade(unsigned int x) {

			int i = 3;

			if (a == 0 || a== 1)		; Zero e 1 não são números primos
			{
				return 0;
			}
			else if (a == 2)
				{
					return 1;
				}
				else if ( rest_of_division(a,2)==0)
				{
					return 0;
```

```
                }
                else
                {
                        while ( rest_of_division(a,i) != 0 && i<a)
                        {
                                i += 2;
                        }
                        if (i==a)
                        {
                                return 1;
                        }
                        else
                        {
                                return 0;
                        }
                }
}
```

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

```
rest_of_division:    cmp R14, R15
                     jl rest_end                    ; R15 < R14
termina o laço
                     sub R14, R15                   ; R15 = R15 - R14
                     jmp rest_of_division       ; repete o laço
```

```
rest_end:                   ret

Primalidade:
    TST    R15              ; R15 == 0?
    JEQ    Nao_primo
    CMP    #1,R15           ; R15 == 1?
    JEQ    Nao_primo
    CMP    #2,R15           ; R15 == 2?
    JEQ    E_primo
    MOV.W  #3,R14           ; R14 = i = 3

While_verificar:
    CMP    R14,R15
    JEQ    E_primo
    PUSH   R14              ; Guarde R14 na pilha
    PUSH   R15              ; Guarde R15 na pilha
    CALL   rest_of_division ; Faça a%i
    TST    R15
    JZ     Nao_primo
    POP    R15              ; Recupere R15
    POP    R14              ; Recupere R14
    ADD.W  #2,R14           ; i += 2
    JMP    While_verificar
```

```
Nao_primo:
    MOV.W   #0,R15          ; return 0
    RET

Nao_primo_2:

    MOV.W   #0,R15          ; return 0
    RET
E_primo
    MOV.W   #1,R15          ; return 1
    RET
```

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!! = 2*4*6*...*n. Por exemplo, 9!! = 1*3*5*7*9 = 945 e 10!! = 2*4*6*8*10 = 3840. Além disso, 0!! = 1!! = 1.

O protótipo da função é:

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

```
#include <stdio.h>
#include <stdlib.h>

unsigned long long rest_of_division (unsigned long long a, unsigned long long b){
    //a/b

    while (a >= b)
```

```
        {
                a -= b;
        }
        return a;
}



unsigned long long Duplo_Fatorial ( unsigned long long n)
{
        if(n==0 || n==1)
        {
                return 1;
        }

        else if(rest_of_division(n,2)==0)
        {
                if(n>0)
                {

                        n = n*Duplo_Fatorial(n-2);
                        return n;
                }
                else
```

```
        {
            return 1;
        }

    }
    else
    {
        if(n>1)
        {
            n = n*Duplo_Fatorial(n-2);
        }
        else
        {
            return 1;
        }
    }
}
```

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);

(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 bits. A variável de entrada é fornecida pelo registrador R15, e o valor de saída também.

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 fornecida no registrador R15, valendo 1 quando o vetor estiver ordenado de forma decrescente, e valendo 0 em caso contrário.

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 do registrador R14, e o tamanho do vetor deverá ser passado pelo registrador R13. A saída deverá ser fornecida no registrador R15.

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);
```

(b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. O endereço do vetor de entrada é dado pelo registrador R15, o tamanho do vetor é dado pelo registrador R14, e o resultado é dado pelo registrador R15.