

This repository

Search

Pull requests

Issues

Gist

AmauriCJr / Micro

Watch

0

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Pulse

Graphs

Branch: master

Micro / Aula 6-7 Respostas

Find file

Copy path

AmauriCJr Update Aula 6-7 Respostas

1e3ebb4 2 days ago

1 contributor

481 lines (418 sloc) 10.2 KB

Raw

Blame

History

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

2

3 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'

4

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

6

7

8 O protótipo da função é:

9

10 unsigned int Raiz_Quadrada(unsigned int S);

11

12 unsigned int Raiz_Quadrada(unsigned int S)

13 {

14 float x;

15 int n;

16 x = S/2;

17 for(n=0;n<=100;n++)

18 {

19 x = (x + (S/x))/2;

20 }

21 return x;

22 }

23

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

25

26 Raiz:

27 clr.w R9

28 Inicio:

29 clr.w R12

30 mov.w R15, R8

31 mov.w #2, R14 ;R14 é o divisor

32 call #Divisao

33 Continua:

34 mov.w R15, R8

35 mov.w R10, R13 ;x = S/2

36 mov.w R13, R14 ;Divisor = x

37 call #Divisao

38 add.w R13, R10 ;x + S/x

39 mov.w #2, R14 ;Divisor = 2

40 mov.w R10, R8

41 call #Divisao

42 jmp Loop

43

44

45 Divisao:

46 cmp #1, R14

47 jeq Condicao1

48 mov.w R14, R11

49 sub.w R14, R8

50 add.w #1, R12

51 Comparar:

52 dec.w R11

53 cmp R11, R8

54 jeq Resultado

55 cmp #0, R11

56 jeq Divisao

```

57 jmp Comparar
58 Condição1
59 mov.w R8, R12
60 Resultado:
61 mov.w R12, R10
62 clr.w R14
63 clr.w R12
64 clr.w R11
65 clr.w R8
66 ret
67 Loop:
68 inc.w R9
69 cmp #64, R9
70 jne Continua
71 mov.w R10, R15
72
73
74 2. (a) Escreva uma função em C que calcule 'x' elevado à 'N'-ésima potência, seguindo o seguinte protótipo:
75
76     int Potencia(int x, int N)
77     {
78         int y;
79         if(N==1) return x;
80         if(N==0) return 1;
81         y = x*Potencia(x, --N);
82         return y;
83     }
84
85 (b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. 'x' e 'n' são fornecidos através dos registradores R15 e R14, respectivamente.
86
87 Potencia:
88 cmp #1, R14
89 jeq FIM
90 cmp #0, R14
91 jeq COND
92 mov.w R15, R12
93 mov.w R15, R11
94 clr.w R15
95 mov.w R12, R13
96 Multiplicacao:
97 add.w R12, R15
98 cmp.w #1, R13
99 dec.w R13
100 jne Multiplicacao
101 mov.w R11, R13
102 dec.w R14
103 cmp #1, R14
104 mov.w R15, R12
105 jeq FIM
106 clr.w R15
107 jmp Multiplicacao
108 COND:
109 mov.w #1, R15
110 FIM:
111
112
113 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.
114
115 clr.w R12
116 Divisao:
117 mov.w R14, R11
118 sub.w R14, R15
119 add.w #1, R12
120 Comparar:
121 dec.w R11
122 cmp R11, R15
123 jeq Resultado
124 cmp #0, R11
125 jeq Divisao
126 jmp Comparar
127 Resultado:
128 mov.w R12, R15
129
130

```

```

131
132 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
133
134 clr.w R12
135 Divisao:
136 mov.w R14, R11
137 sub.w R14, R15
138 add.w #1, R12
139 Comparar:
140 dec.w R11
141 cmp R11, R15
142 jeq Resultado
143 cmp #0, R11
144 jeq Divisao
145 jmp Comparar
146 Resultado:
147 mov.w R12, R15
148 Resto:
149 mov.w R11, R15
150
151
152 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,
153
154     int Primalidade(unsigned int x);
155
156
157 int Primalidade(unsigned int x)
158 {
159     int i, v;
160     for(i=2;i<x;i++)
161     {
162         v = x%i;
163         if(v == 0)
164         {
165             return 0;
166         }
167     }
168     return 1;
169 }
170 }
171
172 (b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. A variável de entrada é fornecida pelo registrador R15, e o valor de
173
174 cmp #2, R15
175 jeq PRIMO
176 cmp #1, R15
177 jeq PRIMO
178 mov.w #2, R14
179 mov.w R15, R13 ;Move o valor do numero a ser testado para um registrador que será usado na verificação
180 clr.w R12
181 Divisao:
182 mov.w R14, R11
183 sub.w R14, R15
184 add.w #1, R12
185 Comparar:
186 dec.w R11
187 cmp R11, R15
188 jeq Verificador
189 cmp #0, R11
190 jeq Divisao
191 jmp Comparar
192 Verificador:
193 cmp #0, R11 ;Compara o resto com 0, se for igual o numero possui um divisor portanto n é primo
194 jeq NPRIMO
195 dec.w R13 ;Subtrai um do reg de verificação pq o divisor n pode ser igual ao dividendo
196 cmp R14, R13 ;Compara o divisor com o R13 e se não for igual reinicia o processo aumentando o divisor
197 jeq PRIMO
198 inc.w R13
199 mov.w R13, R15
200 clr.w R12
201 inc.w R14
202 jmp Divisao
203 NPRIMO:
204 mov.w #0, R15

```

```

205 jmp FIM
206 PRIMO:
207 mov.w #1, R15
208 FIM:
209
210
211 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.
212 O protótipo da função é:
213
214 unsigned long long DuploFatorial(unsigned long long n);
215
216 unsigned long long DuploFatorial(unsigned long long n)
217 {
218
219     long long int s, i;
220     s = 1;
221     for(i=1;i<=n;i++)
222     {
223         if(n%2 == 0)
224         {
225             if(i%2 == 0)
226             {
227                 s = i*s;
228             }
229         }
230         if(n%2 == 1)
231         {
232             if(i%2 == 1)
233             {
234                 s = i*s;
235             }
236         }
237     }
238     }
239     n = s;
240     return n;
241 }
242
243
244 7. (a) Escreva uma função em C que calcula a função exponencial da seguinte forma:
245
246 Considere o cálculo até o termo n = 20. O protótipo da função é double ExpTaylor(double x);
247
248 double ExpTaylor (double x)
249 {
250     // e^x = (x^i)/i!
251     double e, t, s;
252     double n, i, f;
253     f = 1;
254     s = 0;
255     for(i=0;i<=20;i++)
256     {
257         if(i == 0)
258         {
259             f = 1;
260             e = pow(x, i);
261             t = e/f;
262             s = t + s;
263         }else
264         {
265             f = i*f;
266             e = pow(x, i);
267             t = e/f;
268             s = t + s;
269         }
270     }
271     x = s;
272     return x;
273 }
274
275 (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.
276
277 ;O programa não dá conta de fazer mais do que 7 iterações, impossibilitando um resultado preciso.
278

```

```
279 Exponencial:
280 mov.w R15, R5 ;Entrada que n sera mudada
281 clr.w R8 ;R8 = i
282 mov.w R8, R6 ;R6 = i incrementador
283 clr.w R12
284 clr.w R4
285 Inicio:
286 mov.w R6, R8
287 mov.w R8, R14
288 call #Potencia
289 clr.w R14
290 mov.w R8, R13 ;R13 = Decrementador do fatorial
291 call #Fatorial
292 mov.w R8, R14 ;R14 = Divisor
293 mov.w R11, R8 ;R8 = Dividendo
294 call #Divisao
295 inc.w R6 ;incrementa i
296 mov.w R10, R9 ;Variavel para realizar a soma dos termos
297 add.w R9, R4 ;Resultado final
298 cmp #7, R6
299 jeq ACABOU
300 jmp Inicio
301
302
303 Divisao:
304 cmp R14, R8
305 jl Condi1
306 cmp #1, R14
307 jeq Condiacao1
308 mov.w R14, R11
309 sub.w R14, R8
310 add.w #1, R12
311 Comparar:
312 dec.w R11
313 cmp R11, R8
314 jeq Resultado
315 cmp #0, R11
316 jeq Divisao
317 jmp Comparar
318 Condiacao1:
319 mov.w R8, R12
320 jmp Resultado
321 Condi1:
322 mov.w #0, R12
323 jmp Resultado
324 Resultado:
325 mov.w R12, R10 ;Resultado dado em R10
326 clr.w R14
327 clr.w R12
328 clr.w R11
329 clr.w R8
330 ret
331
332
333
334 Potencia:
335 mov.w R5, R15
336 cmp #1, R14
337 jeq FIM
338 cmp #0, R14
339 jeq COND
340 mov.w R15, R12
341 mov.w R15, R11
342 clr.w R15
343 mov.w R12, R13
344 Multiplicacao:
345 add.w R12, R15
346 cmp.w #1, R13
347 dec.w R13
348 jne Multiplicacao
349 mov.w R11, R13
350 dec.w R14
351 cmp #1, R14
352 mov.w R15, R12
```

```
353 jeq FIM
354 clr.w R15
355 jmp Multiplicacao
356 COND:
357 mov.w #1, R15
358 FIM:
359 mov.w R15, R11 ;R10 = Resposta
360 clr.w R12
361 clr.w R13
362 clr.w R14
363 ret
364
365 Fatorial:
366 mov.w R6, R7
367 clr.w R10 ;soma
368 clr.w R12 ;controlador do loop de somas
369 clr.w R14 ;serve pra atribuir o valor da soma passada ao novo somador
370 cmp #0, R7
371 jeq Cond1
372 cmp #1, R7
373 jeq Cond1
374 cmp #2, R7
375 jeq Cond2
376 mov.w R7, R13 ; controlador do numero de somas
377 Loop1:
378 dec.w R13
379 cmp #1, R13
380 jeq FM
381 cmp #0, R14
382 jne Igualador
383 Soma1:
384 add.w R7, R10
385 Loop2:
386 inc.w R12
387 cmp.w R12, R13
388 jeq Loop1
389 inc.w R14
390 jne Soma1
391 Cond1:
392 mov.w #1, R10
393 jmp FM
394 Cond2:
395 mov.w #2, R10
396 jmp FM
397 Igualador:
398 mov.w R10, R7
399 mov.w #1, R12
400 jmp Soma1
401 FM:
402 mov.w R10, R8
403 clr.w R13
404 clr.w R12
405 clr.w R14
406 clr.w R10
407 ret
408
409 ACABOU:
410 mov.w R4, R15
411
412
413 8. Escreva uma sub-rotina na linguagem Assembly do MSP430 que indica se um vetor esta ordenado de forma decrescente. Por exemplo:
414 [5 4 3 2 1] e [90 23 20 10] estão ordenados de forma decrescente.
415 [1 2 3 4 5] e [1 2 3 2] não estão.
416 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
417
418 Decrescente:
419 dec.w R14
420 rla R15
421 cmp 1(R15), 0(R15)
422 jl Crescente
423 cmp #1, R14
424 jeq Decre
425 jne Decrescente
426 Crescente:
```

```
427 mov.w #0, R15
428 jmp FIM
429 Decre:
430 mov.w #1, R15
431 jmp FIM
432 FIM:
433
434
435 9. Escreva uma sub-rotina na linguagem Assembly do MSP430 que calcula o produto escalar de dois vetores, 'a' e 'b':
436
437 O primeiro endereço do vetor 'a' deverá ser passado através do registrador R15, o primeiro endereço do vetor 'b' deverá ser passado através
438
439 10. (a) Escreva uma função em C que indica se um vetor é palíndromo. Por exemplo:
440     [1 2 3 2 1] e [0 10 20 20 10 0] são palíndromos.
441     [5 4 3 2 1] e [1 2 3 2] não são.
442 Se o vetor for palíndromo, retorne o valor 1. Caso contrário, retorne o valor 0. O protótipo da função é:
443
444     int Palindromo(int vetor[ ], int tamanho);
445
446 int Palindromo(int vetor[], int tamanho)
447 {
448     int i, t, s, v;
449     s = 0;
450     t = 0;
451     if (tamanho%2 == 0)
452     {
453         for(i=0;i<=tamanho-1;i++)
454         {
455             t = vetor[i] - vetor[tamanho-i-1];
456             s = t + s;
457             t = 0;
458         }
459     }else if(tamanho%2 == 1)
460     {
461         v =(tamanho/2);
462         for(i=0;i<=v;i++)
463         {
464             t = vetor[i] - vetor[tamanho-i-1];
465             s = t + s;
466             t = 0;
467         }
468     }
469     if(s == 0)
470     {
471         return 1;
472     }else
473     {
474         return 0;
475     }
476 }
477
478
479
480 (b) Escreva a sub-rotina equivalente na linguagem Assembly do MSP430. O endereço do vetor de entrada é dado pelo registrador R15, o tamanho
```

