Arquitetura de Computadores I Ano Letivo 2016/17 - 1º Semestre

Teste	Prático	1_	18/	11	/2.01	6
1 6516	I I auco	1 —	10/		401	u

Nº Mec.:	Nome	:

NOTE BEM: Leia atentamente todas as questões, comente o código usando a linguagem C e respeite a convenção de passagem de parâmetros e salvaguarda de registos que estudou. Na tradução para o Assembly do MIPS respeite rigorosamente os aspetos estruturais e a sequência de instruções indicadas no código original fornecido.

O código em C apresentado pode não estar funcionalmente correto, pelo que **não deve ser interpretado**.

Este teste é constituído por 3 folhas.

.text

1) Codifique em assembly do MIPS a seguinte função is_paldr e preencha a tabela com os registos que usou para cada variável:

```
int is_paldr(char *str) {
                                                             Variável
                                                                             Registo(s)
   char *p_st;
   char *p_end;
                                                             str
                                                                                $a0
   int is_palin;
                                                             p_st
                                                                                $t0
                                                             p end
                                                                                $t1
   p_st = str;
                                                            is_palin
                                                                                $v0
   p_end = str;
                                                             *p_st
                                                                                $t2
   is palin = 1;
   while (*p_end != '\0')
                                                             *p_end
                                                                                $t3
      p_end++;
   p_end--;
   while( (p_st < p_end) && (is_palin == 1)) {</pre>
      if(*p_st != *p_end) {
         is_palin = 0;
      p_st++;
      p_end--;
   return is_palin;
```

```
is_paldr:
                  $t0, $a0
           move
                                        # p_st = str
           move
                  $t1, $a0,
                                        # p_end = str
           li
                  $v0,
                       1
                                        # is_palin = 1
                  $t3, 0($t1)
                                        # while (*p_end != '\0')
wh1:
           lb
           beq
                  t3, '\0', whl_e
           addiu $t1, $t1, 1
                                         # p_end++
                  wh1
           j
           addiu $t1, $t1, -1
                                        # p_end--
wh1 e:
#
                  $t0, $t1, wh2_e
wh2:
                                        # while( p_st < p_end</pre>
           bge
           bne
                  $v0, 1, wh2_e
                                         #
                                                  && is_palin == 1) {
                  $t2, ($t0)
                                         #
                                              if(*p_st != *p_end) {
if:
           lb
           lb
                  $t3, ($t1)
                                         #
           beq
                  $t2, $t3, endif
                                         #
           li
                  $v0, 0
                                         #
                                                is_palin = 0;
                                         #
endif:
           addiu $t0, $t0, 1
                                         #
                                               p_st++
           addiu $t1, $t1, -1
                                        #
                                               p_end--;
                                         #
                  wh2
           j
                                         # return is_palin
                  $ra
wh2_e:
           jr
```

Nº Mec.:Nome:	
---------------	--

2) Codifique em *assembly* do MIPS a seguinte função soma e preencha a tabela com os registos que usou para cada variável:

```
int square(int );
                                                              Variável
                                                                              Registo(s)
int soma(int *lista, int min, int max, int size)
                                                              lista
                                                                              $a0 $s0
     int n,soma;
                                                              &lista[n]
                                                                              $t0
     for(n=0, soma=0; n < size; n++)</pre>
                                                              lista[n]
                                                                              $a0
                                                                              $a1
                                                              min
                                                                                    $s1
         if ( (lista[n] > min) && (lista[n] < max) )</pre>
                                                                              $a2
                                                                                    $s2
                                                              max
                 soma += square( lista[n] );
                                                              size
                                                                              $a3
                                                                                    $s3
      }
                                                                              $s4
                                                              soma
                                                                                    $v0
     return soma;
                                                                              $s5
                                                              n
```

Label	Instrução em assembly	Comentário em C
soma:	subu \$sp,\$sp,28	
	sw \$ra,0(\$sp)	
	sw \$s0, 4(\$sp)	
	sw \$s1, 8(\$sp)	
	sw \$s2, 12(\$sp)	
	sw \$s3, 16(\$sp)	
	sw \$s4, 20(\$sp)	
	sw \$s5, 24(\$sp)	
	move \$s0,\$a0	
	move \$s1,\$a1	
	move \$s2,\$a2	
	move \$s3,\$a3	
	li \$s5, 0	#n = 0;
	li \$s4,0	#soma = 0;
for:	bge \$s5,\$s3,efor	for(n=0, soma=0;)
	sll \$t0,\$s5,2	#t0 = 4*n
	addu \$t0,\$s0,\$t0	#t0 = &lista[n]
	lw \$a0,0(\$t0)	#a0 = lista[n]
if:	ble \$a0,\$s1, endif	#if ()
	bge \$a0,\$s2, endif	
	jal square	
	add \$s4,\$s4,\$v0	#soma += square(lista[n]);
endif:	addi \$s5,\$s5,1	
	j for	
efor:	move \$v0,\$s4	#return soma
		······································

Label	Instrução em assembly	Comentário em C
	lw \$ra,0(\$sp)	
	lw \$s0, 4(\$sp)	
•	lw \$s1, 8(\$sp)	
	lw \$s2, 12(\$sp)	
	lw \$s3, 16(\$sp)	
	lw \$s4, 20(\$sp)	
	lw \$s5, 24(\$sp)	
	lw \$ra,0(\$sp)	
	addu \$sp,\$sp,28	
	jr \$ra	

Nº Mec.:	ome:
----------	------

3) Codifique em *assembly* do MIPS a seguinte função pluv e preencha a tabela com os registos que usou para cada variável:

```
int pluv(int *plu_m, int *plu_h)
                                                             Variável
                                                                             Registo(s)
   int *ultimo;
                                                             plu_m
                                                                                 $a0
   int dd, tot_m, tot_a;
                                                             plu_h
                                                                                $a1
                                                             ultimo
                                                                                $t0
   tot_a = 0;
                                                             dd
                                                                                $t1
   ultimo = plu_m + 12;
                                                                                $t2
                                                             tot_m
   {
                                                                                $v0
                                                             tot_a
      tot_m = 0;
      for (dd = 1; dd <= 30; dd++, plu_h++ )
         tot_m += *plu_h;
      *plu_m = tot_m;
      tot_a += tot_m;
      plu_m++;
   } while( plu_m < ultimo );</pre>
   return tot_a;
```

Label	Instrução em	Comentário em C
	assembly	
	.text	
pluv:	li \$v0, 0	tot_a = 0;
	addiu \$t0,\$a0, 48	ultimo=plu_m+12;
do:	li \$t2, 0	tot_m = 0;
	li \$t1, 1	dd = 1;
for:	bgt \$t1, 30, efor	Salta se dd>30
	lw \$t4, 0(\$a1)	Get *plu_h
	add \$t2, \$t2, \$t4	tot_m += *plu_h;
	addi \$t1, \$t1, 1	dd++
	addi \$a1, \$a1, 4	plu_h++
	j for	
efor:	sw \$t2, 0(\$a0)	*plu_m = tot_m;
	add \$v0, \$v0, \$t2	tot_a += tot_m;
	addiu \$a0, \$a0, 4	plu_m++;
	bltu \$a0, \$t0, do	<pre>while(plu_m<ulti mo);<="" pre=""></ulti></pre>
	jr \$ra	return tot_a;
		t

Label	Instrução em	Comentário em C
Label	nistrução en	Comentario em C
	assembly	
<u> </u>		