Universidade de Aveiro, DETI

Arquitetura de Computadores I, Teste Prático 1 – 18/11/2017

Ano Letivo 2017/18 - 1º Semestre

Nº Mec.:	Nome:

<u>NOTE BEM</u>: 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 correcto, pelo que **não deve ser interpretado**.

Este teste é constituído por 4 folhas.

1) Analise o programa Assembly seguinte e responda às questões que se seguem:

```
0x10010000
        .data
                              #
        .asciiz "TEST1"
X1:
                              #
                              #
        .align 2
                              #
x2:
        .space 20
x3:
        .text
                              # 0x00400000
        .globl main
main:
                $t4,X2
        la
        ori
                $t5,$0,4
                              #
                $t0,$t0,$t0
                              #
        xor
                $t1,$t1,$t1
        xor
L1:
        beq
                $t0,$t5,L2
                              #
                $t2,$t0,$t0
        add
        add
                $t3,$t2,$t2
        addu
                $t3,$t3,$t4
        sw
                $t2,0($t3)
        add
                $t1,$t1,$t2
                              #
                              #
                $t0,$t0,1
        addi
                              #
        Ė
                L1
                              #
L2:
                $t1,4($t3)
        SW
        jr
                $ra
```

- a) Qual o espaço total de memória ocupado pela string "x1"?
 - 6 bytes, 5 + terminador (de 0×10010000 a 0×10010005)
- b) Qual o endereço de memória a que corresponde o *label* "x2"?

```
0x10010008 (1° endereço múltiplo de 4 a seguir a 0x10010005)
```

- c) Se "x2" for o endereço inicial de um array de inteiros, qual a dimensão máxima desse array? 20 / 4 = 5 inteiros
- d) Se "x2" for o endereço inicial de um array de inteiros, qual o endereço de memória da posição x2[3] desse array? &x2[3] = 0x10010008 + 4 * 3 = 0x10010014
- e) Qual o número total de bytes de memória usado pelo segmento de dados (x3-x1)? 28 bytes
- f) Considerando que a primeira instrução do trecho de código fornecido está armazenada a partir do endereço 0x0040000, quais os endereços a que correspondem os labels "L1" e "L2"? (note que a instrução "1a" é decomposta em duas instruções nativas).

L2: 0x00400034

```
L1: 0x00400014
```

g) Quantas vezes é realizado o ciclo de programa?

```
4 vezes ($t0=0, $t0=1, $t0=2, $t0=3)
```

- h) Qual o valor da word de 32 bits armazenada pelo programa na posição x2[3] do array? x2[3] = 2*3 = 6
- i) Qual o valor do registo \$t1 no fim do programa?

```
st1: 2*0 + 2*1 + 2*2 + 2*3 = 12 = 0x0000000C
```

j) Qual o endereço de memória acedido pela instrução "sw \$t1,4(\$t3)"? &X2[3+1]

```
= 0 \times 10010008 + 4 * 4= 0 \times 10010018
```

Cotações: 1-25%, 2-25%, 3-25%, 4-25%

N.º Mec.: _____ Nome: _____

2) Codifique em Assembly do MIPS a seguinte função split odd():

```
int split odd(int *a, int N, int *p odd )
                                                              Variável
                                                                        Registo(s)
                                                                       $a0
  int n even = 0;
                                                                 а
                                                                       $a1
  int *p;
                                                                 Ν
                                                                       $a2
                                                               p_odd
  for (p = a; p < (a + N); p++)
                                                                       $t0
                                                               n_even
                                                                       $t1
    if( (*p % 2) != 0 )
                                                                 *р
                                                                       $t2
      *p odd = *p;
      p_odd++;
    else
      n even++;
  return (N - n_even); Função terminal
```

Label	Instrução em assembly		Comentário em C		Label	Instrução em assembly	Comentário em C
split	odd:						
-	li	\$t0,0	#	n_even=0			
		\$t3,\$a1,2	#				
	addu	\$t3,\$t3,\$a0	#	\$t3=a+N			
		\$t1,\$a0					
for:	bgeu	\$t1,\$t3,end	for #	<pre>while(p<</pre>	(a+N)		
if:		\$t2,0(\$t1)		: {			
		\$t4,\$t2,2	#				
	_	\$t4,\$0,else) {	
		\$t2,0(\$a2)	•		_	•	
		\$a2,\$a2,4	:	p_ode	1+t		
	j	endif	#				
else:			#	else			
		\$t0,\$t0,1			en++		
	!	\$t1,\$t1,4		p++			
	_	for		-		i	
endfor			:		N-n_ever	1)	
	jr	\$ra	#	:			
			ļ				

N.º Mec.: _____ Nome: _____

3) Codifique em Assembly do MIPS a seguinte função main ():

```
#define SIZE 7
                                                                  Variável
                                                                            Registo(s)
int splito(int *, int, int *);
                                                                   nodd
                                                                            $t0
int main(void)
                                                                            $t1
                                                                     i
  static int val[SIZE] = {8, 4, 15, -1987, 9, 27, 16};
  static int odd[SIZE];
  int nodd, i;
  nodd = splito( val, SIZE, odd );
  print string("Result is:");
  for (\overline{i}=0; i < nodd; i++) {
    print int10( odd[i] );
  return 1;
                                Função intermédia
```

Label	Instrução em assembly	Comentário em C	Label	Instrução em assembly	Comentário em C
	.data				
	.eqv SIZE,7				
	.eqv PRINT_STE	₹,4			
	.eqv PRINT_INT	[10,1			
str:	.asciiz "Result	is:"			
val:					
odd:	.space 28	# SIZE * 4 =	28		
	.text				
	.globl main				
main:	subu \$sp,\$sp,4	#		:	
	sw \$ra,0(\$sp)	#			
	la \$a0,val	#			
	li \$a1,SIZE	#			
	la \$a2,odd	#			
	jal splito	#			
	move \$t0,\$v0	# nodd = spl	ito(val	,SIZE,odd)	
	li \$v0,PRINT_	_STR#			
	la \$a0,str	#			
	syscall	<pre># print_str(</pre>	"Result	is:")	
	li \$t1,0	# i=0			
for:	bge \$t1,\$t0,er	ndf # while(i <	nodd) {		
	sll \$t2,\$t1,2	#			
	la \$t3,odd	#			
	addu \$t3,\$t3,\$t	c2 # \$t3 = &c	odd[i]		
	lw \$a0,0(\$t3)				
	li \$v0,PRINT_	_INT10			
	syscall	# print in	t10 (odd	[i])	
	addi \$t1,\$t1,1	# <u>i++</u>			
	j for	# }			
endf:	li \$v0,1	# return 1			
	lw \$ra,0(\$sp)	#			
	addu \$sp,\$sp,4	;			
	jr \$ra	#		:	

N.º Mec.: _____ Nome: _____

4) Codifique em Assembly do MIPS a seguinte função count ():

```
int isn( char, char );
                                                              Variável
                                                                      Registo(s)
                                                                     $a0 -> $s0
int count(char *p, char c)
                                                                 р
                                                                      $a1 -> $s1
                                                                 С
 int n=0;
                                                                      $s2
                                                                 n
 while ( *p != '\0' )
                                                                      $a0
                                                                *P
    n = n + isn(*p, c);
    p++;
  return n;
                              Função intermédia
```

Label	Instrução	em assembly	Come	ntário em C	Label	Instrução	em assembly	Comentário em C
count:	subu	\$sp,\$sp,16	#	Reserva e	spaço na	stack	para 4 reg	istos
	sw	\$ra,0(\$sp)	#	Guarda \$ra	a			
	SW	\$s0,4(\$sp)	#	Guarda \$s	0			
	SW	\$s1,8(\$sp)	#	Guarda \$s	1			
	SW	\$s2,12(\$sp)	#	Guarda \$s	2			
	move	\$s0,\$a0	#	p tem que	ser gua	rdado	num registo	\$ s
	move	\$s1,\$a1	#	c tem que	ser gua	rdado 1	num registo	\$s
	li	\$s2,0	#	n = 0 (to	em que r	esidir	num regist	o \$s)
while:	lb	\$a0,0(\$s0)	#	while(*p	!= '\0')			
	beq	\$a0,'\0',en	dw#	{				
	move	\$a1,\$s1	#					
	jal	isn	#	isn(*p,	c)			
	add	\$s2,\$s2,\$v0	#	n += is:	n(*p,c)			
	addiu	\$s0,\$s0,1	#	p++				
	j	while	#	}				
endw:	move	\$ v 0,\$s2	#	return n		ļ		
	-lw	\$ra,0(\$sp)	#	Repõe \$ra				
	lw	\$s0,4(\$sp)	:	_				
	lw	\$s1,8(\$sp)	#	Repõe \$s1				
	lw	\$s2,12(\$sp)	. i	Repõe \$s2				
	addu	\$sp,\$sp,16	#	Liberta e	spaço oc	upado 1	na stack	
	jr	\$ra	#					
			ļ			ļ		
						1		
						 		
			ļ			ļ		
			<u>.</u>					
						†		
			ļ			ļ		