Arquitetura de Computadores I

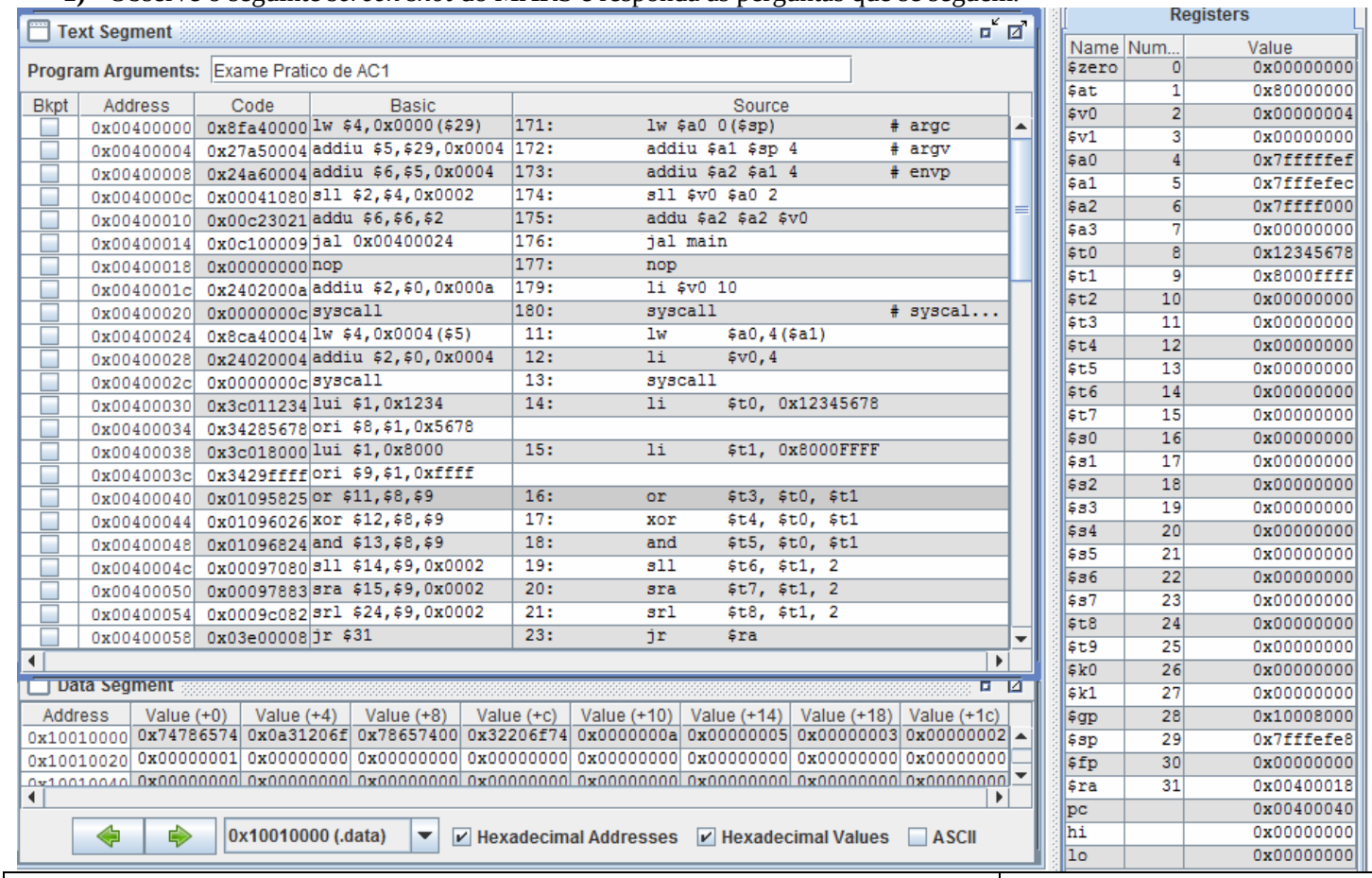
Ano Letivo 2011/12 - 1º Semestre

Teste Prático I

Nº Mec.: _____ Nome: _____ Data: 19-Nov-2011

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**.

1) Observe o seguinte *screen shot* do MARS e responda às perguntas que se seguem.



Text Segment

Program Arguments: Exame Pratico de AC1

Bkpt	Address	Code	Basic	Source
	0x00400000	0x8fa40000	lw $4,0x0000($29)	171: lw $a0 0($sp) # argc
	0x00400004	0x27a50004	addiu $5,$29,0x0004	172: addiu $a1 $sp 4 # argv
	0x00400008	0x24a60004	addiu $6,$5,0x0004	173: addiu $a2 $a1 4 # envp
	0x0040000c	0x00041080	sll $2,$4,0x0002	174: sll $v0 $a0 2
	0x00400010	0x00c23021	addu $6,$6,$2	175: addu $a2 $a2 $v0
	0x00400014	0x0c100009	jal 0x00400024	176: jal main
	0x00400018	0x00000000	nop	177: nop
	0x0040001c	0x2402000a	addiu $2,$0,0x000a	179: li $v0 10
	0x00400020	0x0000000c	syscall	180: syscall # syscal...
	0x00400024	0x8ca40004	lw $4,0x0004($5)	11: lw $a0,4($a1)
	0x00400028	0x24020004	addiu $2,$0,0x0004	12: li $v0,4
	0x0040002c	0x0000000c	syscall	13: syscall
	0x00400030	0x3c011234	lui $1,0x1234	14: li $t0, 0x12345678
	0x00400034	0x34285678	ori $8,$1,0x5678	
	0x00400038	0x3c018000	lui $1,0x8000	15: li $t1, 0x8000FFFF
	0x0040003c	0x3429ffff	ori $9,$1,0xffff	
	0x00400040	0x01095825	or $11,$8,$9	16: or $t3, $t0, $t1
	0x00400044	0x01096026	xor $12,$8,$9	17: xor $t4, $t0, $t1
	0x00400048	0x01096824	and $13,$8,$9	18: and $t5, $t0, $t1
	0x0040004c	0x00097080	sll $14,$9,0x0002	19: sll $t6, $t1, 2
	0x00400050	0x00097883	sra $15,$9,0x0002	20: sra $t7, $t1, 2
	0x00400054	0x0009c082	srl $24,$9,0x0002	21: srl $t8, $t1, 2
	0x00400058	0x03e00008	jr $31	23: jr $ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x74786574	0x0a31206f	0x78657400	0x32206f74	0x0000000a	0x00000005	0x00000003	0x00000002
0x10010020	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Registers

Name	Num...	Value
$zero	0	0x00000000
$at	1	0x80000000
$v0	2	0x00000004
$v1	3	0x00000000
$a0	4	0x7fffffef
$a1	5	0x7fffefec
$a2	6	0x7ffff000
$a3	7	0x00000000
$t0	8	0x12345678
$t1	9	0x8000ffff
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffefe8
$fp	30	0x00000000
$ra	31	0x00400018
pc		0x00400040
hi		0x00000000
lo		0x00000000

Pergunta:	**Resposta:**
a) Admitindo que o programa começou a ser executado no endereço 0x00400000 qual o resultado do ***syscall*** na instrução presente no endereço 0x0040002c?	**Pratico**
b) Qual o valor actual do *Stack Pointer* ?	**0x7FFFEFE8**
c) Qual a próxima instrução a ser executada?	**or $t3, $t0, $t1 (or $11, $8, $9)**
d) Indique qual o código *ascii* do caracter armazenado no endereço 0x1001000e	**0x20**
e) Indique o código máquina da instrução **`and $t5, $t0, $t1`**	**0x01096824**

f) Determine e apresente na tabela, **em hexadecimal**, os valores armazenados nos registos **`$t3-$t8`**, após a execução de todas as instruções a partir do endereço atual do PC até ao endereço 0x00400054.

`$t3`	**0x9234FFFF**	**`$t5`**	**0x00005678**	**`$t7`**	**0xE0003FFF**
`$t4`	**0x9234A987**	**`$t6`**	**0x0003FFFC**	**`$t8`**	**0x20003FFF**

Zona de rascunho

2) Codifique em *assembly* do MIPS a seguinte função **`main`**:

```
int proc_string(char *);

int main(int argc, char *argv[])
{
   static int array[10];
   int *p;
   int res = 0;

   if( (argc >= 10) || (argc < 2) )
      return -1;
   for(p = array; p < (array + argc); p++)
   {
      *p = proc_string( *argv );
      res += *p;
      argv++;
   }
   print_int( res );
   return 0;
}
```

3) Codifique em *assembly* do MIPS a seguinte função **`proc1`**:

```
int proc2(char *, char);

int proc1(char *p, char c, int n)
{
   int res=0;
   do
   {
      if(*p == c)
         res += proc2(p, c + *p);
      else
         *p = *p + 'A' - 'a';
      n--;
   } while (n > 0);

   return res;
}
```

4) Codifique em *assembly* do MIPS a seguinte função **`array_proc`**:

```
int *array_proc(int *list, int num, int *count)
{
   int i;
   int j = num-1, k = 0;

   for(i = 0; i < num; i++)
   {
      if( (list[i] < 200) || (list[i] >= list[j]) )
      {
         list[j] = list[i] ^ list[j];        // exclusive or
         k++;
      }
      j--;
   }
   *count = k;
   return list;
}
```