Arquitectura de Computadores I

Ano Lectivo 2012/13 - 1º Semestre

Teste Prático de Autoavaliação das Turmas Práticas P1cN, P3a e P5c

Nome: _____ Data: 4-Dez-2012

Número Mecanográfico: _____ Classificação: _____

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 aspectos estruturais e a sequência de instruções indicadas no código original fornecido, bem como as indicações, quando existirem, sobre quais os registos a usar para cada variável.
O código em C apresentado pode não estar funcionalmente correcto, pelo que **não deve ser interpretado**.

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

```
int codlista (int num, char* palavras[], int* array);
double media (int num, int* lista);

int main (int argc, char *argv[])
{
   static int array[10];   /* array de inteiros a criar no segmento de dados */
   int total; double med;

   if (argc < 1 || argc > 10) return 1;

   total = codlista (argc, argv, array);   /* análise da lista de parâmetros */
   print_int (total);   /* SYSCALL */

   med = media (argc, array);   /* cálculo da média */
   print_double (med);  /* SYSCALL */

   return 0;
}
```

Label	Instrução em *Assembly* - Comentário em C	*Label*	Instrução em *Assembly* - Comentário em C
	.data		lw $a0, 4($sp)
array:	.space 40 # static int array[10];		la $a1, array
	.text		jal media
	.globl main		
main:	# void main (int argc, char *argv[])		mov.d $f12, $f0 # med = media (...);
	# {		li $v0, 3
	# int total; /* $v0 */		syscall # print_double (med);
	# double medcod; /* $f0 */		
			li $v0, 0 # return 0;
	subiu $sp, $sp, 8		j endif
	sw $ra, 0($sp) # push $ra;		
	sw $a0, 4($sp) # push argc;	else:	li $v0, 1 # return 1;
	blt $a0, 1, else # if (argc < 1		
	bgt $a0, 10, else # && argc > 10)	endif:	lw $ra, 0($sp) # pop $ra;
	# {		addiu $sp, $sp, 8
	la $a2, array		jr $ra # }
	jal codlista		
	move $a0, $v0		
	li $v0, 1		
	syscall # print_int (total);		

2) Codifique em *Assembly* do MIPS a seguinte função *codlista()*:

```
int codstr (char* str, int* num);
int codlista (int num, char* palavras[], int* array)
{
   int i, total = 0;
   for (i = 0; i < num; i++)
      total += codstr (palavras[i], &array[i]);
   return total;
}
```

Label	Instrução em *Assembly* - Comentário em C
codlista:	# int codlista (int num, char *palavras[], int *array)
	# {
	# i→$s0 / total→$s1 / temp→$t1 = 4xi
	subiu $sp, $sp, 20
	sw $ra, 0($sp) # push $ra;
	sw $a0, 4($sp) # push num;
	sw $a1, 8($sp) # push palavras;
	sw $s0, 12($sp) # push $s0;
	sw $s1, 16($sp) # push $s1;
	li $s0, 0 # i = 0;
	li $s1, 0 # total = 0;
codfor:	lw $a0, 4($sp) # $a0 = num
	bge $s0, $a0, eforc # for (; i < num ;)
	# {
	sll $t1, $s0, 2 # temp = 4*i;
	lw $a1, 8($sp) # $a1=&palavras[0];
	addu $a0, $a1, $t1
	lw $a0, 0($a0) # $a0=palavras[i];
	addu $a1, $a2, $t1 # $a1=&array[i];
	jal codstr
	add $s1, $s1, $v0 # total += codstr (palavras[i], &array[i]);
	addi $s0, $s0, 1 # i++;
	j codfor # }
eforc:	move $v0, $s1 # return total;
	lw $ra, 0($sp) # pop $ra;
	lw $s0, 8($sp) # pop $s0;
	lw $s1, 12($sp) # pop $s1;
	addiu $sp, $sp, 20
	jr $ra # }

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

```
int codstr (char* str, int* num)
{
   int cont = 0, cod = 0;
   while (*str != '\0')
   {
      if ((*str >= '0') && (*str <= '9'))
      {
         *str = (*str - '0' + 2) % 10 + '0';
         cod++;
      }
      cont++;
      str++;
   }
   *num = cod;
   return cont;
}
```

Label	Instrução em *Assembly* - Comentário em C	
codstr:		# int codstr (char *str, int *num)
		# {
		# cont→$v0 / cod→$t1
		# car→$t2 / dif $t3
	li $v0, 0	# cont = 0;
	li $t1, 0	# cod = 0;
while:	lb $t2, 0($a0)	# while ((car = *str) != '\0')
	beq $t2, $0, endwh	# {
	blt $t2, '0', endifc	# if ((car >= '0')
	bgt $t2, '9', endifc	# && (car<='9'))
		# {
	sub $t3, $t2, '0'	
	addi $t3, $t3, 2	# dif=car-'0'+2;
	rem $t3, $t3, 10	# dif=dif%10;
	addi $t2, $t3, '0'	# car=dif+'0';
	sb $t2, 0($a0)	# *str = car;
	addi $t1, $t1, 1	# cod++;
		# }
endifc:	addi $v0, $v0, 1	# cont++;
	addiu $a0, $a0, 1	# str++;
	j while	# }
endwh:	sw $t1, 0($a1)	# *num = cod;
		# return cont;
	jr $ra	# }

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

```
double media (int num, int* array)
{
   int i, soma = 0;
   for (i = 0; i < num; i++)
   {
      soma += *array;
      array++;
   }
   return (double) soma / (double) num;
}
```

Label	Instrução em *Assembly* - Comentário em C	
media:		# double media (int num, int *array)
		# {
		# i→$t0/soma→$t1/*array→$t2
	li $t1, 0	# soma = 0;
	li $t0, 0	# i = 0;
medfor:	bge $t0, $a0, endfm	# for (; i<num ;)
		# {
	lw $t2, 0($a1)	# $t2 = *array
	add $t1, $t1, $t2	# soma += *array;
	addiu $a1, $a1, 4	# array++;
	addi $t0, $t0, 1	# i++;
	j medfor	# }
endfm:	mtc1 $t1, $f6	
	cvt.d.w $f0, $f6	# (double) soma;
	mtc1 $a0, $f6	
	cvt.d.w $f6, $f6	# (double) num;
	div.d $f0, $f0, $f6	# return (double) soma / (double) temp;
	jr $ra	# }