Universidade Estadual do Rio de Janeiro
Instituto de Matemática e Estatística
Departamento de Informática e Ciência da Computação

Arquitetura de Computadores I
Prof.: Jacques Alves da Silva

Lista de Exercícios I

1) Quais são os cinco componentes básicos de um computador? Explique a função de cada um deles.
2) Quais são as funções do compilador e do montador (*assembler*)?
3) Quais são os tamanhos e os campos das instruções MIPS nos formatos R, I e J?
4) Qual é a função do registrador $ra no processador MIPS?
5) Quais são os modos de endereçamento do MIPS? Explique cada um deles.
6) Converta os números a seguir para as bases indicadas em cada letra:
   a) $(200{,}828125)_{10} = (\ ?\ )_2$
   b) $(110100{,}1001)_2 = (\ ?\ )_{10}$
   c) $(1000)_{10} = (\ ?\ )_{16}$
   d) $(12C)_{16} = (\ ?\ )_{10}$
   e) $(F63DA)_{16} = (\ ?\ )_2 = (\ ?\ )_8$
   f) $(572346)_8 = (\ ?\ )_2 = (\ ?\ )_{16}$
7) Suponha que se queira representar a fração decimal 0,6 em binário, usando uma palavra de quatro bits:
   a) Encontre a fração binária que fornece o valor aproximado.
   b) Qual o erro percentual?
   c) Repita com uma palavra de 8 bits.
8) Considere o número decimal x=0,3141. Qual é a menor palavra binária que pode ser utilizada para representar este valor com arredondamento menor que 0,3?
9) Adicione comentários ao código do processador MIPS, a seguir, e expresse em uma frase o que este trecho de código faz. Suponha que os registradores *$a0* e *$v0* são usados para a entrada e a saída, respectivamente. Além disso, suponha que inicialmente o registrador de entrada contém um valor *n* que é um inteiro positivo.

```
        addi $t0, $zero, 0
        addi $t1, $zero, 1
loop:   slt $t2, $a0, $t1
        bne $t2, $zero, finish
        add $t0, $t0, $t1
        addi $t1, $t1, 2
        j loop
finish: add $v0, $t0, $zero
```

10) Qual é a instrução em *assembly* correspondente a esta instrução de máquina?

    $000880C0_{16}$

    Observe que a instrução está em hexadecimal.

11) O fragmento de código mostrado, a seguir, processa um *array* e calcula dois valores armazenados nos registradores *$v0* e *$v1*. Suponha que o *array* possua 500 palavras, indexadas de 0 a 499, e que seu endereço base e tamanho estejam armazenados nos registradores *$a0* e *$a1*. Descreva o que faz este trecho de código. Mais especificamente o que é retornado nos registradores *$v0* e *$v1*?

```
        sll $a1, $a1, 2
        add $v0, $zero, $zero
        add $t0, $zero, $zero
outer:  add $t4, $a0, $t0
        lw $t4, 0($t4)
        add $t5, $zero, $zero
        add $t1, $zero, $zero
inner:  add $t3, $a0, $t1
        lw $t3, 0($t3)
        bne $t3, $t4, skip
        addi $t5, $t5, 1
skip:   addi $t1, $t1, 4
        bne $t1, $a1, inner
        slt $t2, $t5, $v0
        bne $t2, $zero, next
        add $v0, $t5, $zero
        add $v1, $t4, $zero
next:   addi $t0, $t0, 4
        bne $t0, $a1, outer
```

12) O programa a seguir foi escrito para copiar palavras que estão armazenadas na memória a partir de um determinado endereço que se encontra em *$a0* para outra área de memória cujo endereço inicial se encontra em *$a1*. O número de palavras copiadas deve ser armazenado no registrador *$v0*. O programa termina de copiar quando o conteúdo de uma palavra for 0. O conteúdo dos registradores *$v1*, *$a0* e *$a1* não precisa ser preservado e a palavra 0 que indica fim de cópia deve ser copiada mas não deve ser contada como uma palavra copiada.

```
loop:   lw $v1, 0($a0)
        addi $v0, $v0, 1
        sw $v1, 0($a1)
        addi $a0, $a0, 1
        addi $a1, $a1, 1
        bne $v1, $zero, loop
```

Existem 4 erros no programa acima. Indique os erros e escreva um programa que seja livre de erros e execute a cópia desejada.

13) Considere o seguinte trecho de código escrito em linguagem de alto nível:

```
while (save[i] == k)
        i += 1;
```

O código a seguir foi gerado por um compilador para a linguagem *assembly* do MIPS, onde *i, j,* e *k* estão armazenadas nos registradores *$s3*, *$s4*, e *$s5*, respectivamente, e o endereço inicial do vetor *save* está no registrador *$s6*.

```
Loop:  sll $t1, $s3, 2
       add $t1, $t1, $s6
       lw $t0, 0($t1)
       bne $t0, $s5, Exit
       addi $s3, $s3, 1
       j Loop
Exit:
```

Reescreva o código acima de modo que seja utilizada no máximo uma instrução de desvio condicional ou de "*jump*" dentro do *loop*. Quantas instruções são executadas antes e depois da otimização se a quantidade de iterações do loop for 10 e na décima primeira iteração *save[i]* seja diferente de *k*?

14) Considere o seguinte trecho de código escrito em linguagem de alto nível:

```
for (i = 0; i <= 100; i ++) {
       a[i] = b[i] + c;
}
```

Suponha que *a* e *b* são *arrays* de palavras e que o endereço base de *a* está em *$a0* e o endereço de *b* está em *$a1*. O registrador *$t0* está associado à variável *i* e *$s0* a variável *c*. Escreva o código MIPS correspondente a este fragmento de código. Quantas instruções são executadas durante o processamento deste trecho de código? Quantas referências a palavras de dados na memória serão feitas durante a execução?

15) Considere as seguintes instruções em C, onde as variáveis *a*, *b*, *c*, *d* e *e* correspondem aos cinco registradores de $s1 a $s5. Mostre os códigos em linguagem *assembly* e em linguagem de máquina no MIPS.
   a) a = b + c;
   b) a = b + 20;
   c) c = d << 10;
   d) b = c & d;
   e) d = e | 100;

Instrução	op	funct	Instrução	op	funct
add	0	32	sll	0	0
sub	0	34	srl	0	2
addi	8	-	beq	4	-
lw	35	-	bne	5	-
sw	43	-	slt	0	42
and	0	36	slti	10	-
or	0	37	j	2	-
nor	0	39	jr	0	8
andi	12	-	jal	3	-
ori	13	-	lui	15	-

Nome	Registrador
$zero	0
$v0-$v1	2-3
$a0-$a3	4-7
$t0-$t7	8-15
$s0-$s7	16-23
$t8-$t9	24-25
$gp	28
$sp	29
$fp	30
$ra	31