UFSM Universidade Federal de Santa Maria
Centro de Tecnologia – DELC
ELC1011 – Organização de Computadores

Nome: GABARITO _____

Prova 1 (07/12/2022)

Responda a cada uma das questões de forma clara e organizada. Apresente todos os passos da solução.

Questão 01 (1,5 pontos, 15 min.) – Traduza o seguinte trecho de código da linguagem C para assembly do MIPS.

```
/* variáveis globais -- traduzir */
int b[5];
int c[4];
int i, j, k;

/*...trecho de código -- traduzir */
        if(c[i] > a){
                if(b[j] == c[i]){
                        b[j] = b[2] + 34;
                } else {
                k++;
                        goto fim;
                }
        }
fim:
```

Listagem 1: Trecho de código em C do problema 1.

Solução:

A solução deste problema está no diretório `q1`. Neste diretório, `q1.c` é o trecho do programa que será traduzido e `q1.asm` a solução em *assembly*.

Questão 02 (1,5 pontos, 15 min.) – Traduza o seguinte trecho de código da linguagem C para assembly do MIPS.

```
/* variáveis globais -- traduzir */
int b[5];
int c[4];
int i, j, k;

/*...trecho de código -- traduzir */
        while((b[i] < c[j]) && (b[j] > a)){
                b[i]++;
        }
```

Listagem 2: Trecho de código em C do problema 2.

Solução:

A solução deste problema está no diretório `q2`. Neste diretório, `q2.c` é o trecho do programa que será traduzido e `q2.asm` a solução em *assembly*.

Questão 03 (1,5 pontos, 15 min.) – Traduza o seguinte trecho de código da linguagem C para assembly do MIPS.

```
/* variáveis globais -- traduzir */
int b[5];
int c[4];
int i, j, k;

/*...trecho de código -- traduzir */
        for(i=0; i<4; i++){
                c[i] = 0;
        }
```

Listagem 3: Trecho de código em C do problema 3.

Solução:

A solução deste problema está no diretório `q3`. Neste diretório, `q3.c` é o trecho do programa que será traduzido e `q3.asm` a solução em *assembly*.

Questão 04 (3,0 pontos, 20 min.) – Traduza o seguinte trecho de código da linguagem C para assembly do MIPS.

```
/* variáveis globais -- traduzir */
int x;
int y;

/*...trecho de código -- traduzir */
int P1(int x, int y)
{
        int a[3]; /* variável local */
        int i;    /* variável local */

        for(i=0; i<3; i++){
                a[i] = x + y + i;
        }
        a[2] = a[1] - a[0];
        return a[2];
}

int P2(int x, int y)
{
        x = P3(x, y);
        y++;
        return x + y;
}

int P1(int x, int y)
{
        x = P2(x, y);
        y++;
        return x + y;
}

int main(void)
{
        x = 7;
        y = 5;

        x = P1(x, y);
        return 0;
}
```

Listagem 4: Trecho de código em C do problema 4.

Solução:

A solução deste problema está no diretório `q4`. Neste diretório, `q4.c` é o trecho do programa que será traduzido e `q4.asm` a solução em *assembly*.

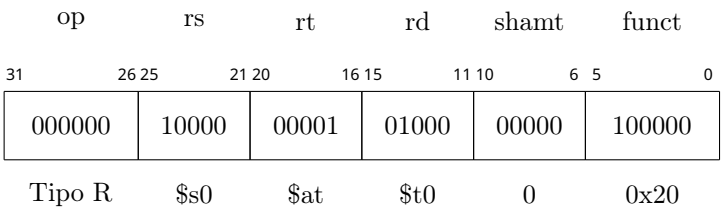
Questão 05 (1,0 ponto, 10 min.) – Traduza as seguintes instruções da linguagem assembly para instruções em linguagem de máquina do processador MIPS.

(A) add $t0, $s0, $at

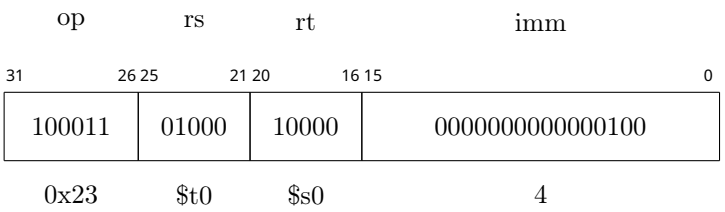
(B) lw $s0, 4($t0)

Solução:

(A) add $t0, $s0, $at = 0x02014020



(B) lw $s0, 4($t0) = 0x8D100004



Questão 06 (1,0 ponto, 15 min.) – Converta os números A=3,1 e B=-7,2 para números em ponto fixo *sem sinal* com sinal (complemento de 2). Use 5 bits para a parte inteira e 3 bits para a parte fracionária. Faça a soma A+B. Faça a subtração A-B. Verifique se houve overflow nas operações.

Solução:

Usamos o método da divisão longa e multiplicação longa para converter os números $|A|$ e $|B|$ (números sem sinal) de decimal para ponto fixo.

$$|A| = 00011,000_2$$
$$|B| = 00111,001_2$$

O número A é positivo. A representação de A em complemento de 2 é igual a $|A|$. O número B é negativo. Tomamos $|B|$e complementamos.

$$A = 00011,000_2$$
$$B = 11000,111_2$$

Realizamos a soma $A + B$:

$$\begin{array}{lll} \text{Vai-um} & & 000000000 \\ A & & 00011000 \\ B & + & 11000111 \\ \hline & & 11011111 \end{array}$$

O vem-um e o vai-um da coluna com os bits mais significativos são iguais. Não há estouro (*overflow*) nesta operação de soma.

Realizamos a subtração $A - B$:

$$\begin{array}{lll} \text{Vai-um} & & 001110001 \\ A & & 00011000 \\ \overline{B} & + & 00111000 \\ \hline & & 01010001 \end{array}$$

O vem-um e o vai-um da coluna com os bits mais significativos são iguais. Não há estouro (*overflow*) nesta operação de subtração.

Questão 07 (0,5 ponto, 10 min.) – Um programa utiliza $20\,\%$ do tempo em operações de entrada e de saída (E/S ou I/O). Se as operações de E/S são aceleradas 100 vezes, qual a aceleração do sistema.

Solução:

A aceleração do sistema é dada por:

$$\begin{aligned} a_s &= \frac{1}{(1-p) + \frac{P}{a}} \\ &= \frac{1}{(1-0,2) + \frac{0,2}{100}} \\ a_s &\approx 1,25 \end{aligned}$$

O sistema será acelerado cerca de $25\,\%$.

Fórmulas Desempenho

$$T_{exe} = N_{clk} \cdot T_{clk}$$

$$T_{exe} = N_{inst} \cdot \overline{CPI} \cdot T_{clk}$$

$$T_{exe} = \sum_{i=1}^{N} \left(N_{inst}^{i} \cdot CPI^{i} \right) \cdot T_{clk}$$

$$a_s = \frac{1}{(1-p) + \frac{P}{a}}$$

Registradores

Nome	Número	Nome	Número
$zero	$0	$t8 a $t9	$24 a $25
$at	$1	$k0 a $k1	$26 a $27
$v0 a $v1	$2 a $3	$gp	$28
$a0 a $a3	$4 a $7	$sp	$29
$t0 a $t7	$8 a $15	$fp	$30
$s0 a $s7	$16 a $23	$ra	$31

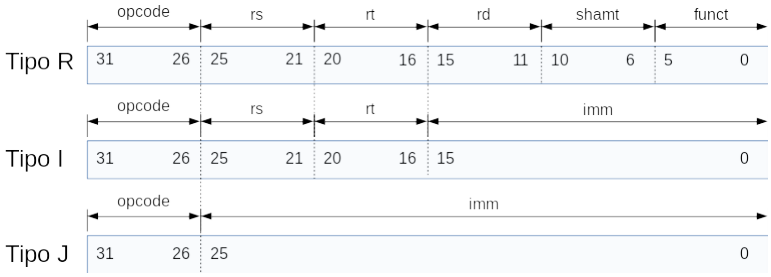


Figura 1: Os campos das instruções R, I e J.

Algumas Instruções

Sintaxe	Significado	Nota
add $1, $2, $3	$1 = $2 + $3	Adiciona os registradores $2 e $3 e armazena o resultado em $1.
sub $1, $2, $3	$1 = $2 - $3	Realiza a subtração $2-$3 e armazena o resultado em $1.
addi $1, $2, imm	$1 = $2 + imm	Adiciona um registrador ($2) com um valor imediato imm de 16 bits (com sinal), após a extensão de sinal.
lw $1, imm($2)	$1 = memória[$2+imm]	Carrega 4 bytes da memória, iniciando no endereço $2+imm, no registrador $1.
lb $1, imm($2)	$1 = memória[$2+imm]	Carrega o byte (com a extensão do sinal para 32 bits) da memória do endereço $2+imm, no registrador $1.
sw $1, imm($2)	memória[$2+imm] = $s1	Armazena os 4 bytes do registrador $1 na memória, iniciando no endereço $2+imm.
sb $1, imm($2)	memória[$2+imm] = $s1	Armazena o byte menos significativo do registrador $1 na memória, no endereço $2+imm.
and $1, $2, $3	$1 = $2 & $3	Operação AND bit a bit entre o registrador $2 e $3. O resultado é armazenado em $1.
addi $1, $2, imm	$1 = $2 & imm	Operação AND bit a bit entre $2 e um valor imediato, após a extensão do sinal (imm sem sinal).
or $1, $2, $3	$1 = $2 | $3	Operação OR bit a bit entre o registrador $2 e $3. O resultado é armazenado em $1.
ori $1, $2, imm	$1 = $2 + imm	Operação OR bit a bit entre $2 e um valor imediato, após a extensão do sinal (imm sem sinal).
beq $1, $2, label	if($1==$2) desvie para label	Desvia para o rótulo label se o valor dos registradores $1 e $2 são iguais.
bne $1, $2, label	if($1!=$2) desvie para label	Desvia para o rótulo label se o valor dos registradores $1 e $2 são diferentes.
j label	Desvie para label	Desvia para o rótulo label incondicionalmente.
jr $1	Desvie o endereço em $1	Desvia para o endereço dado pelo registrador $1.
jal label		Desvia para o rótulo label e guarda o endereço de retorno (PC+4) em $ra.
slt $1, $2, $3	$1 = 1 se ($2<$3)	Faz o registrador $1 igual a 1 se o registrador $2 é menor ou igual a $3, senão, faz o registrador $1 igual a 0.
slti, $1, $2, imm	$1 = 1 se ($2<imm)	Faz o registrador $1 igual a 1 se o registrador $2 é menor ou igual a imm, após a extensão do sinal (complemento de 2), senão 0.
la $1, label	$1 = label	Carrega em $1 o endereço dado por label.
li $1, imm	$1 = imm	Carrega em $1 o valor imediato imm.
lui $1, imm		Carrega o valor imediato imm de 16 bits nos bytes mais significativos de $1. Os bytes menos significativos são feitos iguais a zero.