Universidade Federal de Santa Catarina, INE/CTC
INE 5411 – Organização de Computadores
Avaliação P1 - 2008-2

Nome:_____**Matrícula**:_____

Instruções e definição de critério de avaliação:

- A interpretação das questões é parte integrante desta avaliação. As respostas devem ser fornecidas no espaço para elas reservado abaixo de cada questão. O verso da folha de prova não será avaliado.
- Nas questões que solicitam resposta e justificativa, a resposta não será pontuada sem a devida justificativa, nem se esta última estiver incorreta.
- A pontuação de cada item ("a", "b", etc.) de uma questão é indivisível. Para um item ser considerado correto, toda sua resposta deve estar correta. Um erro de sintaxe em linguagem de montagem anula o item (por isso os códigos solicitados são propositadamente curtos): na dúvida, consulte o manual. Erros de cálculo também anulam um item: confira cuidadosamente seus cálculos.

Parte I - Compreensão de conceitos básicos [3,0 pontos]

1. [valor: 6 × 0,5 = 3,0] Responda **sucintamente** as perguntas abaixo:

a) O processador 80286 (precursor da arquitetura x86-32) usava endereços de memória codificados em 24 bits e 16 bits para representação de dados. Qual o número máximo de bytes endereçáveis (NB) e qual o máximo inteiro sinalizado representável (MAXINTEGER)? As respostas devem ser expressas em potências de dois.
NB = **MAXINTEGER =**

b) Seja a estrutura de dados abaixo, descrita na linguagem C, a ser compilada para um processador de 32 bits. Assuma que os elementos da estrutura `record` são alocados dentro de um segmento de 8 bytes contígüos de memória, conservando-se a ordem em que são declarados. Supondo que o endereço-base da estrutura é 0x10008000 e que a alocação foi efetuada de forma que todos os dados estejam alinhados, quais os endereços dos elementos `b` e `c`?

```
struct record{
char a;
short b;                  // endereço de b = 0x
int c; }                  // endereço de c = 0x
```

c) Suponha que os registradores tenham sido assim inicializados: $gp = 0x10008000 e $s1=0x12345678. Preencha o conteúdo de cada um dos bytes da palavra de memória cujo endereço é 0x10008000, imediatamente após a execução do código abaixo. Afirmação: "O processador que exibe o comportamento mostrado na tabela abaixo é do tipo *big endian*." A afirmação é verdadeira ou falsa?
Resposta: . (O preenchimento da tabela é a sua justificativa)

```
sb $s1, 0($gp)
srl $s1, 8
sb $s1, 1($gp)
srl $s1, 8
sb $s1, 2($gp)
srl $s1, 8
sb $s1, 3($gp)
```

	MSB			LSB
conteúdo	0x	0x	0x	0x
endereço	0x10008000	0x10008001	0x10008002	0x10008003

d) Como resultado da execução da instrução `mult $s1, $s2`, obteve-se `HiLo = 0x FFFFFFFF80000000`. Afirmação: "O produto obtido poderia ser representado com apenas 32 bits". A afirmação é verdadeira ou falsa? **Justifique**
Resposta: Justificativa:

e) Seja `Ri` um registrador de uso geral e seja `(Ri)` a notação para o modo de endereçamento indireto via `Ri`. Sabe-se que a arquitetura IA-32 admite instruções `add (R1), R2` e `add R1, (R2).`
Afirmação: "As instruções citadas fazem ambas um único acesso à memória". A afirmação é verdadeira ou falsa? **Justifique.**
Resposta: . Justificativa:

f) Escreva um código em linguagem de montagem, com apenas duas instruções nativas do MIPS, que desvie para o endereço simbólico L quando os 4 bits menos significativos do registrador $s0 são nulos, sem destruir o conteúdo de $s0. **Resposta**?

Parte II – Aplicação de conceitos básicos [4,0 pontos]

2. **[valor: 2 x 0,5 = 1,0]** O função `int sum-of-squares (int i)` retorna a soma dos quadrados dos números inteiros no intervalo [1, i]. Suponha que a função execute em um processador operando a 223,89 MHz e que as instruções de soma, multiplicação e de desvio requeiram, respectivamente, 4, 8 e 3 ciclos. Responda às questões abaixo e **justifique suas respostas mostrando seus cálculos intermediários.**

```
sum-of-squares:     add $v0, $zero, $zero
loop:               mul $t1, $a0, $a0
                    add $v0, $v0, $t1
                    addi $a0, $a0, -1
                    beq $a0, $zero, exit
                    j loop
   exit:            jr ra
```

a) Qual o número médio de ciclos por instrução para essa função quando i = 10 (deve ser expresso em decimal com precisão de 2 casas após a vírgula)?
$CPI_{médio}$ = . Cálculos: $CPI_{médio}$ =

b) Qual o tempo de execução do Programa 1 quando i = 10 (deve ser expresso em potência de 10)?
$t_{execução}$ = . Cálculos: $t_{execução}$ =

3. **[valor: 0,25+0,25+0,5+0,5+0,5 = 2,0]** Assuma que, após a montagem do Programa 1 (p. 3), o código-objeto resultante é estaticamente ligado com o da rotina `printf`. Sabe-se que os cabeçalhos dos códigos-objeto são:

	Procedure printf	**Procedure main**
Text size	0x400	0x58
Data size	0x10	0x1C

As Figuras 1a e 1b mostram o layout de memória resultante da montagem de cada um dos procedimentos. A Figura 1c mostra esquematicamente o layout de memória após a ligação. Nessas figuras, `(X)` representa o conteúdo do endereço `X`. Para o código ligado, determine o seguinte:

a) Qual o endereço efetivo de memória associado ao label L3?
Resposta: L3 = 0x

b) Qual o endereço efetivo de memória associado ao label LA?
Resposta: LA = 0x

c) Qual o endereço efetivo de memória associado ao label L1?
Resposta: L1 = 0x

d) Qual a codificação em linguagem de máquina da instrução j L1?
Resposta: 0x

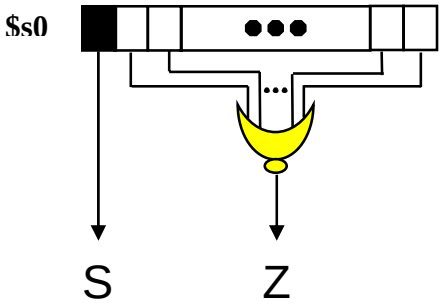
e) Qual a codificação em linguagem de máquina da instrução beq $t2, $zero, L2?
Resposta: 0x

4. **[valor: 2 × 0,5 = 1,0]** O modo de endereçamento indexado com deslocamento, suportado em algumas arquiteturas (como a IA-32)**,** não é suportado pelas instruções nativas do MIPS. Entretanto, ele poderia ser definido através da pseudo-instrução: `lw $rt, K($rb+$ri)# $rt = MEMÓRIA[K+ $rb + $ri]`, onde `rt`, `rb` e `ri` denotam os registradores de destino, de base e de índice e onde K representa uma constante sinalizada representável em 16 bits. Mostre como sintetizar essa pseudo-instrução com apenas duas instruções nativas, sem que o mecanismo de exceção seja acionado em caso de overflow. No código, utilize os símbolos `rt`, `rb` e `ri` para representar registradores arbitrários de uso geral.
Resposta: a)

b)

Parte III – Generalização a partir de conceitos básicos [3,0]

5. **[valor: 4 x 0,25 = 1,0]** A arquitetura IA-32 armazena "flags", que são interpretados por instruções de desvio condicional. Embora o MIPS não possua um registrador específico para armazenar "flags", os desvios condicionais usam sinais do "datapath" para detectar se a condição do desvio é verdadeira ou falsa. Além de `beq` e `bne`, o MIPS32 possui desvios nativos que fazem comparações com zero monitorando o bit de sinal (S) e a saída de um detector de zeros (Z), conforme ilustra a figura abaixo. Assumindo S e Z como variáveis Booleanas, escreva, para cada desvio, a função Booleana que equivalha à condição em que o desvio é tomado. (Use X′ para denotar o complemento da variável Booleana X, "•" para AND e "+" para OR).



	Instrução	Desvia quando:	Função Booleana
a)	`bltz $s0, label`	$s0 < 0	
b)	`blez $s0, label`	$s0 ≤ 0	
c)	`bgez $s0, label`	$s0 ≥ 0	
d)	`bgtz $s0, label`	$s0 > 0	

6. **[valor: 0,25+0,25+0,5+0,5+0,5 = 2,0]** Suponha agora que a rotina `printf`, invocada pelo Programa 1 (abaixo), é carregada usando o mecanismo de DLL. As Figuras 2a e 2b ilustram o "layout" da memória em dois cenários distintos: antes e depois da carga da rotina `printf`, respectivamente. Sabe-se que a ligação dinâmica utiliza a técnica "lazy procedure linkage". Preencha as informações solicitadas abaixo:

a) **LA = 0x** b) **LB = 0x** c)**LC = 0x**

d) **LD = 0x** e) **conteúdo do endereço 0x1000 0100 = 0x**

Programa 1

```
1           .text
2           .align 2
3           .globl main
4     main: subu $sp, $sp, 16
5           sw $ra, 0($sp)
6           sw $a0, 12($sp)
7           sw $0, 4($sp)
8           sw $0, 8($sp)
9     loop: lw $t6, 8($sp)
10          mul $t7, $t6, $t6
11          lw $t8, 4($sp)
12          addu $t9, $t8, $t7
13          sw $t9, 4($sp)
14          addu $t0, $t6, 1
15          sw $t0, 8($sp)
16          ble $t0, 100, loop
17          la $a0, str
18          lw $a1, 8($sp)
19          jal printf
20          move $v0, $0
21          lw $ra, 0($sp)
22          addu $sp, $sp, 16
23          jr $ra
24          .data
25          .align 0
26    str:  .asciiz "The sum from 0 .. 100 is %d\n"
```