UFPR – Bacharelado em Ciência da Computação
CI064 – Software Básico — 2018-1

**Primeira Prova**

**1.** Re-escreva o código fonte do laço `for`, para obter um tempo de execução menor. Traduza seu código otimizado para *assembly* e o otimize para execução no processador segmentado de 5 estágios. Seu código *assembly* deve empregar as convenções de programação do MIPS.

O resultado da multiplicação só pode ser usado por **`mfhi,mflo`** após um ciclo de espera. O processador possui adiantamento, e necessita de *branch* e *load delay slots*. [10 pontos]

Para facilitar a correção indique os registradores como `ra`, `rb`, etc.
Escreva seu programa nas páginas do meio da folha.

```
#define N 1024
int A[N], B[N], C[N];
...
s = 0;
for (i=0; i < N; i = i+1) {
   s = s + (A[i] * B[i]);
}
...
```

**2.** Esta questão tem três itens. [10 pontos]

(a) Desemhe um diagrama que indique claramente as dependências de dados no código;

(b) indique claramente quaisquer modificações no código para garantir que ele seja executado corretamente num processador que não possua circuitos de adiantamento (*forwarding*);

(c) desenhe um diagrama simplificado do processador e indique quais os circuitos de adiantamento são necessários especificamente para este trecho de código.

```
add r2, r3, r4
sw  r2, 0(r2)
lw  r6, 0(r2)
jr  r6
sw  r6, 0(r2)
```

**3.** Escreva um *shell script* que se comporta como `make`: no diretório corrente, se qualquer dos arquivos de cabeçalho (`.h`) for modificado, todos, e somente, os arquivos com código C (`.c`) que dependam do(s) arquivos modificados (`.h`) devem ser re-compilados. [5 pontos]

Espaço em branco proposital.

**Segunda Prova**

Você foi encarregado de escrever o *software* para um sistema embarcado que contém um periférico que transfere um octeto para um dispositivo remoto sempre que ocorrer uma escrita no registrador de dados deste periférico.

O registrador de controle é alocado no endereço `0xc000.0000` e `contrl.setInt=1` provoca uma interrupção depois que o octeto é transmitido, e `contrl.clrInt=1` remove o pedido de interrupção. O bit `contrl.interrOut=1` programa uma interrupção após o envio de um octeto; e o bit `contrl.interrInp=1` programa uma interrupção após o recebimento de um octeto.

O registrador de *status* é alocado ao endereço `0xc000.0004` e o registrador de dados ao endereço `0xc000.0008`. Assim que um octeto é transmitido, o bit `status.sent=1`. Assim que um octeto é recebido, o bit `status.recv=1`.

**4.** Sua tarefa é escrever a rotina que transmite uma *string*, que deve ser lida do endereço `0x0040.0000`. O periférico deve gerar uma interrupção sempre que um octeto for escrito no registrador de dados.

Escreva em pseudo-C – tão completo quanto possível – a rotina de envio de uma *string*, e também em C, o pseudocódigo do tratador da interrupção de transmissão. [10 pontos]

**5.** Sua tarefa é escrever em pseudo-C – tão completo quanto possível – a rotina de espera ocupada que recebe uma *string* do periférico e a deposita a partir do endereço `0x0044.0000`. O periférico entrega um octeto sempre que o bit `status.new=1`. [5 pontos]

**6.** Esta questão tem três itens. [10 pontos]

(a) O que é um 'processo' e como este é representado/mantido pelo Sistema Operacional?

(b) Quais são os possíveis estados de um processo? Para todos os estados, dê um exemplo de ação do programador/usuário que causa a mudança de um estado para outro.

(c) Liste os componentes do registro de ativação de `main()`.

**Exame Final**

Você foi encarregado de escrever o *software* para um sistema embarcado que contém um periférico que transfere um octeto para um dispositivo remoto sempre que ocorrer uma escrita no registrador de dados deste periférico, que não se comporta como uma interface serial.

O registrador de controle é alocado no endereço `0xc000.0000` e `contrl.setInt=1` provoca uma interrupção depois que o octeto é transmitido, e `contrl.clrInt=1` remove o pedido de interrupção. O bit `contrl.interrOut=1` habilita a interrupção após o envio de um octeto.

O registrador de *status* é alocado ao endereço `0xc000.0004` e o registrador de dados ao endereço `0xc000.0008`. Assim que um octeto é transmitido, o bit `status.sent=1`. Assim que um octeto é recebido, o bit `status.recv=1`. A leitura do registrador de *status* faz com que todos os seus bits mudem para zero.

**7.** Sua tarefa é escrever a rotina que transmite uma *string*, que deve ser lida do endereço `0x0040.0000`. O periférico deve gerar uma interrupção sempre que um octeto for escrito no registrador de dados. Escreva em pseudo-C – tão completo quanto possível – a rotina de envio de uma *string*. [20 pontos]

**8.** Escreva em pseudo-C, o pseudocódigo do tratador da interrupção de transmissão.[20 pontos]

**9.** Descreva o processo de ligação estática de um arquivo `a.out` com a biblioteca `libZumbi.a`. Esta biblioteca provê as funções `morde()`, `contamina()` e `degola()`. [30 pontos]

**10.** Esta questão tem quatro itens. Considere, em todos os itens, que ocorre adiantamento através do banco de registradores. [30 pontos]

**(a)** Indique as dependências entre as instruções.

```
sw  r1, 64(r2)

lw  r1, 0(r2)

lw  r3, 0(r1)

jr  r3

add r4, r1, r2
```

**(b)** Acrescente o que for necessário para garantir a execução correta num processador sem circuitos de adiantamento.

```
sw  r1, 64(r2)

lw  r1, 0(r2)

lw  r3, 0(r1)

jr  r3

add r4, r1, r2
```

**(c)** Acrescente o que for necessário para garantir a execução correta num processador com circuitos de adiantamento.

```
sw  r1, 64(r2)

lw  r1, 0(r2)

lw  r3, 0(r1)

jr  r3

add r4, r1, r2
```

**(d)** Justifique a sua resposta Para o item (c): quais as razões para as inclusões, quais as razões para não incluir.