UFPR – Bacharelado em Ciência da Computação
CI064 – Software Básico – 2017-1

Primeira Prova

1. Re-escreva o código fonte do laço **for**, para obter um tempo de execução menor. Você deve aplicar, ao menos, duas técnicas de otimização de código fonte. [10 pontos]

```
#define N 1024
int A[N], B[N], C[N];
...
k = 1;
for (i=0; i < N; i+=1,k+=2) {
   C[i] = fun(i,k)*A[i] + B[i];
}
...
```

```
int fun(int m, int n) {
   int shf, mul;
   min = n << 2;
   shf = m << n;
   return(min<shf ? min : shf);
}
```

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

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

(b) desenhe um diagrama simplificado do processador e indique quais os circuitos de adiantamento necessários;

(c) desenhe um diagrama de tempo (diagonal do tempo) indicando a evolução das instruções ao longo dos estágios a cada ciclo.

```
add r2, r3, r4
lw  r8, 0(r2)
sub r6, r7, r8
beq r6, r2, dest
nop
```

3. Esta questão tem cinco itens. Suas respostas devem ser concisas e precisas. Enrolação incorre em desconto(s).

(a) Para que servem, na programação em *assembly*, as *pseudoinstruções* e as *diretivas*? [2 pts]

(b) enuncie o conceito de *abstração* e dê três exemplos de abstrações providas pelo sistema operacional Unix e seus derivados; [2 pts]

(c) descreva o funcionamento de um *pipeline* no Unix e justifique sua utilidade; [2 pts]

(d) enuncie a diferença entre listas de comandos Bash separados por '**&&**', por '**||**', e por '**;**'. Cada lista usa somente um dos tipos de operadores; [2 pts]

(e) descreva o comportamento do comando abaixo. [2 pts]

```
for F in *.c ; do
   if [ ${F} -nt x.h ] ; then
       mv ${F} ${F%.c}.cpp
   fi
done
```

Segunda Prova Semestral

4. Considere um sistema em que três processo ficam prontos para executar no mesmo instante, que P1 necessita de 1800μs para completar, que P2 necessita de 1600μs, e que P3 necessita de 2000μs. Um *quantum* dura 500μs, e o tempo de uma troca de contexto pode ser ignorado. Quais os tempos totais de execução em cada uma das duas formas de escalonamento estudadas? [5 pontos]

5. Existem sistemas em que o escalonamento dos processos é baseado em prioridade. Em todas as trocas de contexto, o processo de maior prioridade é escolhido para executar. Fazendo uso dos dados da questão anterior, e considerando que as prioridades são os números dos processos ($P3 > P2 > P1$), compute o tempo total de execução dos três processos. [2 pontos]

6. Você foi encarregado de escrever o *software* para um sistema embarcado que contém um periférico que transmite um octeto sempre que ocorrer uma escrita em seu registrador de dados. Sua tarefa é escrever a rotina que transmite uma *string*. Se programado para tal, o periférico gera uma interrupção sempre que um octeto for escrito no registrador de dados. Assim que enviar o octeto, o bit `status.sent`=1. O registrador de controle é alocado no endereço `0xc000.0000` e `contrl.interr`=1 programa a geração da interrupção quando o octeto é transmitido, e `contrl.clrInt`=1 remove o pedido de interrupção. O registrador de *status* é alocado ao endereço `0xc000.0004` e o registrador de dados ao endereço `0xc000.0008`.

Escreva em pseudo-C – tão completo quanto possível – a rotina de envio de uma *string*, que deve ser lida do endereço `0x0040.0000`, e também em C, o pseudocódigo do tratador da interrupção de transmissão. [10 pontos]

7. Descreva a estrutura de dados, e os algoritmos de inserção e busca em uma tabela de *string*s. [3 pontos]

8. Mostre como uma tabela de *string*s pode ser usada numa tabela de espalhamento com colisões. [2 pontos]

9. Por que a ligação com uma biblioteca compartilhada estática é mais eficiente, em termos de tempo de execução, do que a ligação com uma biblioteca compartilhada dinâmica?[3 pontos]

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.

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.

10. 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. [30 pontos]

11. Sua tarefa é escrever a rotina 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. [10 pontos]

12. Traduza para *assembly* do MIPS o programa abaixo. 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, *branch* e *load delay slots*. Otimize seu programa *assembly*. [20 pontos]
Para facilitar a correção indique os registradores como ra, rb, *etc.*

```
#define N 1024
int a, X[N], A[N],B[N];
...
a = 32;
for(i=0; i < N; i+=2) {
   X[i] = (int)(A[i] * B[i] + 16*a);
   X[i+1] = (int)(A[i+1] * B[i+1] + 16*a);
}
```

O diagrama abaixo é base para a resposta às questões que seguem. Os endereços são todos representados em hexadecimal.

```
# módulo 1
     .text
     .global main,k
     .extern r,f,z
main:
000: addi sp,sp,-64
     ...
010: la s2, z
014: lw a0, 0(s2)
018: la t1, r
01c: lw a1, 0(t1)
020: jal f
024: nop
028: sw v0,16(s2)
     ...
1f8: jr, ra
1fc: addi sp,sp,64

     .data
     .global k
     .org 0x1000
k:   .space 0x10
l:   .space 4
m:   .space 4
     ...
```

```
# módulo 2
     .text
     .global f,p,r
     .extern y,k,g
f:
000: addi sp,sp,-16
     ...
020: la s3, k
024: lw a0, 0(s3)
028: la t2, y
02c: lw a1, 0(t2)
030: jal g
034: nop
038: sw v0,8(s3)
     ...
0f8: jr, ra
0fc: addi sp,sp,16

     .data
     .global p,r
     .org 0x1000
p:   .space 4
q:   .space 4
r:   .space 4
     ...
```

```
# módulo 3
     .text
     .global g,y,z
     .extern f,k,p
g:
000: addi sp,sp,-32
     ...
040: la s4, p
044: lw a0, 512(s4)
048: la s5, k
04c: lw a1, 12(s5)
050: nop
054: jal f
058: nop
05c: sw v0,4(s4)
     ...
2f8: jr, ra
2fc: addi sp,sp,32

     .data
     .global x,y,z
     .org 0x1000
x:   .space 0x20
y:   .space 4
z:   .space 4
```

13. Construa a tabela de símbolos local para os três módulos. As tabelas devem conter informação de relocação. [15 pontos]

14. Mostre a alocação dos três módulos em memória. O endereço inicial do segmento TEXTO é 0x2000 e o endereço inicial do segmento DADOS é 0x8000. [15 pontos]

15. Construa uma tabela com os endereços alocados e relocados. [10 pontos]