UFPR – Bacharelado em Ciência da Computação
CI064 – Software Básico — 2016-1

Primeira Prova

1. Para responder a esta questão, explicite quaisquer suposições necessárias. [10 pontos]
Considere o programa ao lado, para ser executado no processador segmentado mais simples, sem nenhuma forma de adiantamento e nem bloqueios, mas com *branch delay-slot* e *load delay-slot*. (i) Acrescente ao código o que for necessário para garantir a execução correta deste programa. Qual o número total de ciclos necessários para armazenar a redução contida em `r5`? (ii) Otimize o código para reduzir o número de ciclos. Qual o novo número total de ciclos? Qual o ganho com relação ao item anterior?

```
    la r20, 0x40000000
    li r21, 800
    move r5, r0
L:  lw  r10, 0(r20)
    add r5, r5, r10
    addiu r20, r20, 4
    addiu r21, r21, -4
    bne r21, r0, L
    sw  r5, -808(r20)
```

2. Traduza para *assembly* do MIPS o trecho de programa abaixo. Seu código *assembly* deve empregar as convenções de programação do MIPS. [10 pontos]

```
int log2(int n) {
   if (n < 2)
      return 1;
   else
      return ( 1 + log2(n>>1) );
}

   ...
   x = log2(96000);
   ...
```

3. Esta questão tem cinco itens:

(a) enuncie, concisa e precisamente, a diferença entre um `alias` e uma função em Bash;[1 pto]

(b) enuncie, concisa e precisamente, a diferença entre um **`#define`** e uma função em C; [1 pto]

(c) enuncie, concisa e precisamente, a diferença entre um processo Unix executando em *foreground* e um processo executando em *background*; [1 pto]

(d) enuncie, concisa e precisamente, a diferença entre uma lista de comandos Bash separados por '`&&`' e uma lista de comandos Bash separados por '`||`'; [1 pto]

(e) enuncie, concisa e precisamente, a função do programa `cpp`. [1 pto]

Segunda Prova

4. O *Algoritmo da Pilha* é usado para encontrar o elemento LRU de uma sequência de referências. Considere a seguinte sequência de referências a páginas:

1 2 3 4 5 6 7 7 2 1 3 5 1 2 3

Identifique a página LRU a cada momento através do algoritmo da pilha. Mostre o estado da pilha após cada referência. [5 pontos]

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

```
# módulo 1
      .text
      .global main
      .extern r,f
main:
000:  addi sp,sp,-64
      ...
010:  la s2, k
014:  lw a0, 0(s2)
018:  la t1, r
01c:  lw a1, 0(t1)
020:  jal f
024:  nop
028:  sw v0,256(s2)
      ...
1f8:  jr, ra
1fc:  addi sp,sp,64

      .data
      .global k
      .org 800
k:    .space 0x100
m:    .space 4
      ...
```

```
# módulo 2
      .text
      .global f
      .extern y,g
f:
000:  addi sp,sp,-16
      ...
020:  la s3, p
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 800
p:    .space 4
q:    .space 4
r:    .space 4
```

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

      .data
      .global x,y,z
      .org 800
x:    .space 4
y:    .space 4
      .org a00
z:    .space 4
```

5. Construa a tabela de símbolos global para os três módulos. [5 pontos]

6. Mostre a alocação dos três módulos em memória. O endereço inicial do segmento TEXTO é `0x1000` e o endereço inicial do segmento DADOS é `0x8000`. [5 pontos]

7. Construa a tabela de relocação para os três módulos. [5 pontos]

8. Construa a tabela de símbolos já relocada para o executável resultante da ligação dos três módulos. [5 pontos]

Exame Final

9. Escreva uma versão otimizada do trecho de código abaixo e justifique <u>todas</u> as transformações efetuadas no código fonte. [15 pontos]

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

10. Traduza para *assembly* do MIPS o programa otimizado da resposta à Questão 9. 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*. [25 pontos]

Para facilitar a correção indique os registradores como `ra`, `rb`, *etc.*

11. O que ocorre após a detecção de uma interrupção por um processador MIPS32r2? O que é responsabilidade do *hardware* e o que é responsabilidade do *software*? [15 pontos]

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

```
# módulo 1
      .text
      .global main
      .extern r,f
main:
000:  addi sp,sp,-64
      ...
010:  la s2, k
014:  lw a0, 0(s2)
018:  la t1, r
01c:  lw a1, 0(t1)
020:  jal f
024:  nop
028:  sw v0,256(s2)
      ...
1f8:  jr, ra
1fc:  addi sp,sp,64

      .data
      .global k
      .org 0x800
k:    .space 0x100
l:    .space 4
m:    .space 4
      ...
```

```
# módulo 2
      .text
      .global f
      .extern y,g
f:
000:  addi sp,sp,-16
      ...
020:  la s3, p
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 0x800
p:    .space 4
q:    .space 4
r:    .space 4
      ...
```

```
# módulo 3
      .text
      .global g
      .extern f,k
g:
000:  addi sp,sp,-32
      ...
040:  la s4, x
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 0x800
x:    .space 0x200
y:    .space 4
z:    .space 4
```

12. Construa a tabela de símbolos global para os três módulos. Esta tabela deve conter a informação de relocação. [15 pontos]

13. 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]

14. Construa uma tabela com os endereços relocados. [15 pontos]