UFPR – BCC

CI312 – Arquiteturas Avançadas de Computadores – 2015-1

Primeira Prova

Responda as questões abaixo supondo que o programa é executado num processador MIPS R10000, que emprega o conjunto de instruções MIPS-64, e é descrito em `http://www.inf.ufpr.br/roberto/ci312/mips10k.pdf`. Suas respostas devem explicitar quaisquer suposições a respeito do comportamento do processador MIPS R10000, e se alguma outra fonte de informações for consultada, ela deve ser citada. Aspectos relacionados ao processamento paralelo (*thread-level*) devem ser ignorados. Se você usar uma planilha para resolver as questões, entregue-a (por e-mail) como parte de sua resposta (prefiro gnumeric).

1 Supondo que não ocorram faltas nas caches nem nas TLBs, calcule o CPI ao executar o trecho de código abaixo. Indique claramente o custo (em ciclos) de cada instrução, os riscos entre as instruções e os atrasos decorrentes de cada risco. Ignore faltas causadas por instruções. [10 pontos]

```
0:      la    r1, 0x04000000  # r1 <- X
1:      la    r2, 0x04000000 + 3*8*8192
2:      move  r3, r2          # r2,r3 <- Y
3: L:   ld.d  f2, 0(r1)       # f2 = X[i]
4:      ld.d  f4, 8(r1)       # f4 = X[i+1]
5:      mul.d f6, f2, f4      # X[i]*X[i+1]
6:      ld.d  f8, 0(r2)       # f8 = Y[i]
7:      mul.d f8, f2, f8      # X[i]*Y[i]
8:      st.d  f8, 8(r2)       # Y[i+1] = f8
9:      st.d  f6, 16(r1)      # X[i+2] = f6
A:      daddi r2, r2, 16      # Y += 2
B:      daddi r1, r1, 24      # X += 3
C:      bne   r1, r3, L       # X[8192]?
```

2 Repita a questão anterior, agora considerando faltas nas caches L1 e L2, mas acertos nas TLBs. Suponha que os vetores `X[]` e `Y[]` estão carregados em RAM. Compute a taxa de acertos nas duas caches. Ignore faltas causadas por instruções. [5 pontos]

3 Repita a questão anterior, agora considerando faltas nas caches L1 e L2 e nas TLBs. Suponha que os vetores `X[]` e `Y[]` estão carregados em RAM. Compute a taxa de acertos nas duas TLBs. Ignore faltas causadas por instruções. [5 pontos]

4 Reescreva o código para minimizar as bolhas e recompute seus resultados para a questão (1). O ganho de desempenho obtido afeta sua resposta à questão (2)? Se sim, como afeta? [10 pontos]

Segunda Prova

Escreva um programa com um par produtor-consumidor para ser executado no cMIPS. O programa pode ser escrito por duplas de alunos. As respostas deverão ser enviadas por e-mail para `roberto@inf.ufpr.br` até as 12:00 do dia 27 de junho, sábado. Se você não receber um 'recebido' para sua mensagem, dentro de uma hora do envio, mude o sufixo do arquivo para `.ttt` e envie novamente. Esta prova vale 30 pontos.

O produtor é um programa em C que computa N primos com o crivo de Eratóstenes. Veja, como modelo, o laboratório sobre o relógio de tempo-real em
`http://www.inf.ufpr.br/roberto/ci064/labContador.html`

Cada número primo computado deve ser depositado numa fila circular com 16 posições, que é compartilhada com o consumidor.

O consumidor é o tratador da interrupção do contador externo. Um modelo para o tratador é discutido no laboratório indicado acima. O consumidor, no tratador de interrupções, deve remover os elementos da fila circular e copiá-los para o armazenador `buffer`, que pode ser alocado no *handler* e declarado como `extern` no produtor.

A variável `number`, que deve ser declarada em `include/handlers.s`, deve ser usada para indicar o número de elementos na fila. Esta variável deve ser declarada como `extern` no produtor.

Decorridas ≥ 20 interrupções e computados ≥ 200 primos – no mínimo 20 interrupções e no mínimo 200 primos – as interrupções devem ser paralizadas e então o produtor imprime todo o conteúdo do armazenador. Ajuste o intervalo entre as interrupções para atender aos dois limites.

Seu programa deve usar as instruções **`ll`** e **`sc`** (*load-linked* e *store-conditional*) para sincronizar produtor e consumidor.

O início do tratador da interrupção do relógio deverá ficar parecido com o código abaixo. Se necessário, aumente o tamanho de `_counter_saves`.

```
#-------------------------------------------------------------
# interrupt handler for external counter attached to IP5=HW3
# for extCounter address see vhdl/packageMemory.vhd
.bss
.align  2
.set noreorder
.global _counter_val, number, queue, q_hd, q_tl, buffer
.comm   _counter_val 1*4  # accumulate number of interrupts
.comm   number       1*4  # number of elemente enqueued
.comm   queue       16*4  # circular queue
.comm   q_hd         1*4
.comm   q_tl         1*4
.comm   buffer     256*4  # buffer for printing
# _counter_saves[0]=a0, [1]=a1, [2]=a2, ...
.comm   _counter_saves 8*4 # area to save up to 8 registers
```

Seu código será simulado para a atribuição da nota. Serão avaliadas a corretude da solução (60%) e a clareza e organização do código (40%).

Os dois programas, um em C e a nova versão de `handlers.s`, **devem** ser entregues de acordo com a página 3 da especificação do primeiro trabalho. Esta parte da resposta não acrescenta pontuação mas implica em desconto de 30% da nota caso ignorada.

Podem ser úteis os capítulos 8 e 9 de `http://www.inf.ufpr.br/roberto/ci064/swbas.pdf`.