UFPR – PPGInf – BCC

CI702 – Arquitetura de Computadores
CI312 – Arquiteturas Avançadas de Computadores
2013-1

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 outra fonte de informações for consultada, ela deve ser citada. Aspectos relacionados a 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, qual o CPI ao executar o trecho de código abaixo? Indique claramente o custo (em ciclos) de cada instrução, os riscos entre elas e os atrasos decorrentes de cada risco. Ignore faltas nas caches de 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 nas caches de 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 nas caches e TLBs de 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)? Como afeta? [5 pontos]

Segunda Prova

1) Traduza o trecho de código abaixo, escrito para um processador escalar, para a versão vetorial do MIPS. O conjunto das instruções vetoriais é listado abaixo. [10 pontos]

Programa versão escalar

```
0:      la    r1, 0x04000000  # r1 <- X
1:      addiu r9, r1, 8*256   # r2 <- Y
    L:  ld.d  f2, 0(r1)       # f2 <- X[i]
3:      ld.d  f4, 2048(r1)    # f4 <- Y[i]
4:      mul.d f6, f2, f4      # X[i]*Y[i]
5:      ld.d  f8, 4096(r1)    # f8 <- Z[i]
6:      mul.d f8, f2, f8      # X[i]*Y[i]
7:      add.d f0, f6, f8      # f6+f8
8:      st.d  f0, 6144(r1)    # W[i] <- f0
9:      daddi r1, r1, 8       # X += 1
A:      bne   r1, r9, L       # X[256]?
```

Instruções vetoriais

```
                  # XXX one of add,sub,mul,div; YYY one of sub,div
XXXvv.d v1,v2,v3  # double vetor-vetor
XXXvs.d v1,v2,f0  # double vetor-escalar
YYYsv.d v1,f0,v2  # double escalar-vetor
                  # CC in {eq,ne,gt,lt,ge,le}
sCCvs             # set vectorMask if (v[i] CC scalar)
sCCvv             # set vectorMask if (v1[i] CC v2[i])
lv      v1,r1     # load vector v1,  ender fonte em r1
sv      v1,r1     # store vector v1, ender destino em r1
lvws v1,(r1,r2)   # load vec with stride,  v[i] <- M[r1 + i*r2]
svws (r1,r2),v1   # store vec with stride, M[r1 + i*r2] <- v[i]
lvi v1,(r1,v2)    # load vec indexed,  v1[i] <- M[r1 + v2[i]]
svi (r1,v2),v1    # store vec indexed, M[r1 + v2[i]] <- v1[i]
```

2) Quais as diferenças, em termos de desempenho global, dos protocolos de coerência de caches por invalidação e por atualização? Justifique sua resposta. [5 pontos]

3) Mostre como se pode implementar um semáforo binário com as instruções `ll` (*load-linked*: `ll r1,desl(r2)`) e `sc` (*store-conditional*: `sc r1,desl(r2)`). [5 pontos]

4) Considere um multiprocessador simétrico com 3 processadores interligados por um barramento. As caches são mantidas coerentes por um protocolo de invalidação, cuja máquina de estados está no diagrama abaixo. Os blocos das caches são de 8 palavras de 32 bits. Qual o estado final das caches, supondo o estado inicial mostrado abaixo, e a execução dos seguintes comandos pelos três processadores. Os comandos estão mostrados na ordem absoluta de tempo; os índices nos blocos das caches são irrelevantes, e as caches são infinitas. [10 pontos]

P0: | EXCL | `t[0]..t[7]` | P1: | SHAR | `u[0]..u[7]` | P2: | SHAR | `u[0]..u[7]` |

```
P0                              P1                              P2
...                             ...                             for(i=0; i<7; i++)
...                             ...                               t[i] = u[i] * v[i];
...                             ...                             ...
...                             v[6] = 0;                       ...
...                             ...                             ...
a=0;                            ...                             ...
for(j=0; j<7; j++)              ...                             ...
  a = a + t[j] + v[j+4];        ...                             ...
...                             ...                             ...
...                             v[7] = 0;                       ...
...                             ...                             ...
...                             ...                             for(i=8; i<16; i++)
...                             ...                               t[i] = u[i] * v[i];
...                             ...                             ...
for(j=0; j<15; j++)             ...                             ...
  u[j+8] = t[j+4] * v[j];       ...                             ...
...                             v[8] = 0;                       ...
```

O protocolo usa escrita forçada na primeira escrita em um bloco; após a primeira escrita, o protocolo usa escrita preguiçosa. Quando um bloco `sujo` é substituído na cache, todo o bloco deve ser copiado para a linha em memória. As linhas contínuas denotam operações iniciadas pelo processador; linhas tracejadas denotam ações remotas iniciadas pelos controladores das outras caches e difundidas no barramento. O diagrama de estados mostra as transições que ocorrem em função dos comandos de coerência gerados pelo processador local (*P-∗*) ou pelos controladores de cache remotos (*Le-∗* ou *Esc-∗*). Este protocolo usa os estados *exclusive* e *modified* para reduzir o tráfego causado por escritas.