Listas de exercícios para a P2 e Psub de OAC

Thiago Tomás de Paula

Universidade de Brasília, janeiro de 2023

Sumário

1 Processadores

1.1 Datapath e CPI

1. Para os caminhos de dados dos processadores vistos em aula com ISA de 9 instruções,

   a) cite os componentes (blocos) que compõem cada processador – Uniciclo, Multiciclo e Pipeline –, resumindo suas funções;

   b) para cada processador citado acima, diga quais blocos são necessários para executar

      – uma instrução do tipo R;
      – uma instrução de load;
      – uma instrução de store;
      – uma instrução de branch.

2. (Exemplo no slide 15, página 24)
   Considerando o workload do compilador gcc, qual a CPI média do RISC-V multiciclo implementado?

   - load: 22% (5 ciclos)
   - store: 11% (4 ciclos)
   - operações lógico-aritméticas: 49% (4 ciclos)
   - desvios condicionais: 16% (3 ciclos)
   - desvios incondicionais: 2% (3 ciclos)

3. (Exemplo no slide 14, página 16)
   Considerando um workload de 25% loads, 10% stores, 11% branches, 2% jals e 52% operações Tipo-R, qual a CPI média para o processador Multiciclo implementado em sala?

4. (Exemplo no slide 17, página 25)
   Suponha os seguintes tempos das unidades operativas:

   - acesso à memória: 200 ps;
   - operação da ULA: 100 ps;
   - acesso ao banco de registradores: 50 ps.

   Dado o workload composto de:

   - 25% loads;
   - 10% stores;
   - 11% branches;
   - 2% jumps;
   - 52% operações com a ULA,

Compare o desempenho entre os três tipos de processador implementados, considerando para o pipeline que 50% dos loads é seguido de uma operação que requer o argumento; 25% dos desvios são previstos erradamente e que o atraso da previsão errada é 1 ciclo de clock; jumps usam 2 ciclos de clock; ignore outros hazards.

5. Suponha que uma instrução num processador Uniciclo tenha período T. No Pipeline equivalente, essa instrução é a junção de N etapas balanceadas. Compare a forma de onda do clock para executar a instrução em cada processador, justificando as diferenças nos desenhos.

   Feito isso, pondere: se a instrução agora fosse executada, ainda em tempo T, por um processador Multiciclo, seria possível saber que o processador é Multiciclo *apenas* pela forma de onda do clock? E se a forma de onda mudasse, sendo agora o resultado de várias instruções distintas, de durações conhecidas?

6. Suponha que a implementação de um processador Uniciclo demande tempo T para a execução da instrução mais longa. A criação de um processador em Pipeline pode ser feita a partir da divisão deste tempo em N etapas perfeitamente balanceadas. Sabendo que

   i. a inclusão de um registrador de pipeline em um estágio incrementa o tempo do estágio em t devido ao seu setup time;

   ii. para um dado workload, o processador pipeline de m estágios possui uma CPI média de C_m; e

   iii. a adição de um estágio a mais é penalizado pelo incremento de CPI_{inc} na CPI média, devido ao aumento da probabilidade de ocorrência de hazards,

   encontre o número de etapas (N) ótimo que minimiza o tempo médio de execução de uma instrução no processador em pipeline.

   Dica: primeiro deduza quais grandezas, CPI ou período de clock, são afetadas pelas informações i, ii e iii.

1.2 Hazards

1. Explique e exemplifique cada tipo de hazard abaixo:

   a) de controle;

   b) de dados;

   c) estrutural.

   Feito isso, responda: como seria possível causar hazard estrutural nos processadores de 9 instruções feitos em sala?

2. As instruções de branch a princípio geram hazards de controle. No contexto da programação em Assembly RISC-V, qual é uma solução via software para o hazard? Explique o por quê.

2 Memória cache

1. Explique detalhadamente por que a introdução de memórias cache melhoram o desempenho de um processador. Na sua explicação, também deixe evidente por que é vantajoso o uso de caches secundárias.

2. (Exemplo no slide 19, página 26)
   Suponha que uma taxa de falhas de cache de instruções para um programa seja de 2% e que uma taxa de falhas de cache de dados seja de 4%. Se um processador possui uma CPI de 2 sem qualquer stall de memória e a penalidade de falha é de 100 ciclos para todas as falhas, determine o quanto mais rápido um processador executaria com uma cache perfeita que nunca falhasse. Considere o programa com 36% acesso à memória de dados.

3. (Exemplo no slide 19, página 28)
   Suponha que aumentemos o desempenho do processador do exemplo anterior dobrando a frequência do clock. Como a frequência da memória principal é improvável de ser alterada, considere que o tempo absoluto para manipular uma falha de cache não mude. O quanto mais rápido será o computador com maior frequência de clock, considerando a mesma taxa de falhas do exemplo anterior?

4. (Exemplo no slide 19, página 30)
   Suponha que tenhamos um processador com CPI básico de 1.0, considerando que todas as referências a dados acertem na cache primária e uma frequência de clock de 5GHz. Considere um tempo de acesso à memória principal de 100ns, incluindo todo tratamento de falhas. Suponha que a taxa de falhas por instrução da cache primária seja de 2%. O quanto mais rápido será o processador se acrescentarmos uma cache secundária que tenha um tempo de acesso de 5ns para um acerto ou uma falha e que seja grande o suficiente para reduzir a taxa de falhas para a memória principal para 0.5%?

5. Suponha que tenhamos um processador com CPI básico $\text{CPI}_0 > 1$, considerando que todas as referências a dados acertem na cache primária e uma frequência de clock de f_0 GHz. Considere ainda um tempo de acesso à memória principal de T ns, incluindo todo tratamento de falhas. Suponha que a taxa de falhas por instrução da cache primária seja de P%. Qual será o fator de desempenho de um novo sistema computacional frente ao anterior se aumentarmos a frequência de clock do processador para $f > f_0$ GHz, reduzirmos a CPI para 1 e acrescentarmos uma cache secundária que tenha um tempo de acesso de $t \ll T$ ns para um acerto ou uma falha, e que seja grande o suficiente para reduzir a taxa de falhas para a memória principal para p%?

3 Assembly RISC-V

1. Respeitando a convenção do uso dos registradores do RISC-V, escreva em Assembly RV32IMF função que percorra os valores num array de inteiros (words) e os imprima no terminal caso sejam divisíveis por `a1`> 0. Suponha que o endereço em memória do array foi fornecido em `a0`.

2. Se no código acima `a0` se tornasse agora um array de bytes, quais mudanças deveriam ser feitas para manter a funcionalidade do programa?

3. Se no código do item 1 `a0` se tornasse agora um array de floats de precisão simples, quais mudanças deveriam ser feitas para manter a funcionalidade do programa?

4. Respeitando a convenção do uso dos registradores do RISC-V, compile para Assembly RV32IMF a função F1 ao abaixo.

```
float F1 ( float x[], int pos, float h[], int N )
{ int k;
  float y;
  y = -5.0;
  for (k = 0; k < N; k++)
    y += 3.5 * h[k] * x[(pos + k) % N];
  printf("y[%d]=%f\n",pos,y);
  return y;
}
```

   Na sua resposta, faça as suposições que julgar necessárias, e. g., `float x[]` em `a0`.