Universidade do Minho

Módulo 3 – Encadeamento

Ficha 2

Encadeamento e resolução de anomalias

1. Introdução

No final deste módulo os alunos deverão ser capazes de:

- avaliar os ganhos/perdas conseguidas com o encadeamento de instruções e com a utilização de atalhos;
- discutir e avaliar as potencialidades e limitações de uma organização encadeada no que respeita à capacidade de realização de características da arquitectura (por exemplo, extensões ao conjunto de instruções).

2. Material de apoio

A bibliografia relevante para este módulo é constituída pelas secções 4.4 e 4.5 do livro "Computer Systems: a Programmer's Perspective", de Randal E. Bryant e David O'Hallaron.

3. Exemplos

EXEMPLO 1.1

Identifique para cada ciclo do relógio a ocupação de cada estágio do processador, para a versão PIPE- do Y86 – injecção de bolhas. O código é apresentado com cada instrução etiquetada. A coluna esquerda da tabela é preenchida com a etiqueta correspondente à instrução apropriada. Justifique a sua solução no espaço abaixo.

```
I1:   irmovl $10, %eax
I2:   irmovl $20, %ebx
I3:   irmovl $30, %ecx
I4:   addl %edx, %eax
```

	1	2	3	4	5	6	7	8	9	10	11	12	13
I1	F	D	E	M	W								
I2		F	D	E	M	W							
I3			F	D	E	M	W						
bolha						E	M	W					
I4				F	D	D	E	M	W				

Justificação
Só existe uma dependência de dados entre I4 e I1, devido à escrita e posterior leitura de %eax. I4 só pode completar o estágio D no ciclo seguinte à escrita (W) de I1, ou seja a leitura de %eax acontece no ciclo 6

EXEMPLO 1.2

Usando o mesmo código do exemplo anterior, identifique para cada ciclo do relógio a ocupação de cada estágio do processador, para a versão PIPE do Y86 – versão com atalhos. Justifique devidamente a sua resposta, indicando quais os sinais de atalho utilizados.

	1	2	3	4	5	6	7	8	9	10	11	12	13
I1	F	D	E	M	W								
I2		F	D	E	M	W							
I3			F	D	E	M	W						
I4				F	D	E	M	W					

Justificação
Só existe uma dependência de dados entre I4 e I1, devido à escrita e posterior leitura de %eax.
No ciclo 5, quando I4 precisa de ler o valor de %eax, a escrita de I1 ainda não foi efectuada. No entanto, o valor a escrever está disponível no registo W, sinal W_valE. Este sinal é realimentado para o estágio D, podendo ser usado durante o ciclo 5.
O sinal a realimentar é o sinal val_B, pois %eax é o registo da direita da instrução addl %edx, %eax
Ciclo 5:
D : val_B = W_valE

EXEMPLO 1.3

Sabendo que o código dos exemplos anteriores está carregado em memória a partir do endereço 0x0100, qual o valor do PC no ciclo 5.

Resposta
As instruções I1, I2 e I3 têm 6 *bytes* de comprimento (icode:ifun + rA:rB + I) logo a instrução I4 está armazenada a partir do endereço 0x0100 + 3*6 = 0x0112
A instrução I4 tem dois *bytes* de comprimento ((icode:ifun + rA:rB) logo a instrução a seguir, que seria lida no ciclo 5, está no endereço 0x0114 e este é o valor do PC neste instante.

EXEMPLO 2.1

Identifique para cada ciclo do relógio a ocupação de cada estágio do processador, para a versão PIPE- do Y86 – injecção de bolhas. O código é apresentado com cada instrução etiquetada. A coluna esquerda da tabela é preenchida com a etiqueta correspondente à instrução apropriada. Justifique a sua solução no espaço abaixo.

```
I1:   irmovl $100, %ebx
I2:   mrmovl $0(%ebx), %eax
I3:   addl %ebx, %eax
```

	1	2	3	4	5	6	7	8	9	10	11	12	13
I1	F	D	E	M	W								
bolha				E	M	W							
bolha					E	M	W						
bolha						E	M	W					
I2		F	D	D	D	D	E	M	W				
bolha								E	M	W			
bolha									E	M	W		
bolha										E	M	W	
I3			F	F	F	F	D	D	D	D	E	M	W

Justificação
Existe uma dependência de dados entre I2 e I1, devido à escrita e posterior leitura de %ebx. I2 só pode completar o estágio D no ciclo seguinte à escrita (W) de I1, ou seja a leitura de %ebx acontece no ciclo 6. Existe outra dependência de dados entre I3 e I2, devido à escrita e posterior leitura de %eax. I3 só pode completar o estágio D no ciclo seguinte à escrita (W) de I2, ou seja a leitura de %eax acontece no ciclo 10.

EXEMPLO 2.2

Usando o mesmo código do exemplo anterior, identifique para cada ciclo do relógio a ocupação de cada estágio do processador, para a versão PIPE do Y86 – versão com atalhos. Justifique devidamente a sua resposta, indicando quais os sinais de atalho utilizados.

	1	2	3	4	5	6	7	8	9	10	11	12	13
I1	F	D	E	M	W								
I2		F	D	E	M	W							
bolha					E	M	W						
I3			F	D	D	E	M	W					

Justificação
Existe uma dependência de dados entre I2 e I1, devido à escrita e posterior leitura de %ebx. No final do ciclo 3, o valor a guardar em %ebx fica disponível à saída da ALU, no sinal e_valE. Este sinal é realimentado para o estágio D, podendo ser usado no final do ciclo 3. O sinal a realimentar é val_B Ciclo 3 - D : val_B = e_valE A instrução I3 depende de I1 e I2. Embora o valor de %ebx pudesse ser lido de M_valE no ciclo 4, o valor de %eax só é lido de memória no ciclo 5, nunca podendo estar disponível no ciclo 4. Esta é uma penalização do tipo load/use e implica a injecção de uma bolha. No final do ciclo 5 o valor de %ebx pode ser lido de W_valE e o valor de %eax pode ser lido de m_valM. Ciclo 5 – D: val_A = W_valE; val_B = m_valM

4. Exercícios

Exercício 1

Usando o código apresentado abaixo, identifique para cada ciclo do relógio a ocupação de cada estágio do processador, para a versão PIPE do Y86 – versão com atalhos. Justifique devidamente a sua resposta, indicando quais os sinais de atalho utilizados.

```
I1:  irmovl %10, %edx
I2:  irmovl %20, %ebx
I3:  irmovl %30, %ecx
I4:  mrmovl $0(%esi), %eax
I5:  xorl %eax, %eax
I6:  jne I9
I7:  rmmovl %eax, $0(%esi)
I8:  halt
I9:  rmmovl %eax, $0(%edi)
I10: halt
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Justificação

Exercício 2.1

Reordene o programa anterior por forma a minimizar o número de bolhas injectadas no processador.

```
I1:
I2:
I3:
I4:
I5:
I6:
I7:
I8:
I9:
I10:
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Exercício 2.2

. Qual o tamanho em *bytes* deste programa?

. Qual o valor do PC no ciclo 7, se o programa estiver armazenado em memória a partir do endereço 0x0050?

Resposta

Exercício 3

A instrução de invocação de funções existente no conjunto de instruções do Y86 é o `call`, correspondente a um salto incondicional. Pretende-se aumentar este conjunto de instruções com invocações condicionais de funções, `callxx`, sendo `xx` correspondente às condições existentes para os saltos.

3.1 - Indique como codificaria estas instruções (icode:ifun, rA:rB, imm).

3.2 – Suponha, para a organização PIPE, que a lógica de previsão de destino dos saltos é idêntica à utilizada para os saltos, isto é, os saltos são previstos como tomados. Quais as dependências de controlo e dados criadas e como podem ser resolvidas? **Nota:** não esquecer que o `call` escreve na pilha se o salto for tomado.

Exercício 4

A instrução de retorno de funções existente no conjunto de instruções do Y86 é o `ret`, correspondente a um salto incondicional. Pretende-se aumentar este conjunto de instruções com o retorno condicional de funções, `retxx`, sendo `xx` correspondente às condições existentes para os saltos.

3.1 - Indique como codificaria estas instruções (icode:ifun, rA:rB, imm).

3.2 – Uma vez que o destino do `ret` só é conhecido após a fase de leitura de memória, a previsão dos saltos condicionais como tomados não faz sentido para esta instrução. Uma alternativa é prever o `retxx` como não tomado, actualizando o `PC` com `valP` e corrigindo posteriormente se a condição for verdadeira. Indique, justificando, se esta instrução, obedecendo a este pressupostos, pode ser implementada na organização PIPE.