
 Universidade do Minho	<p>Módulo 8</p> <p><b>Y86 PIPE: Resolução de anomalias e escalonamento dinâmico</b></p>	
--	---	---

## 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;
- avaliar as potencialidades e limitações inerentes ao processo de escalonamento de instruções em arquiteturas super-escalares

### 1.1. Conteúdos e Resultados de Aprendizagem relacionados

<b>Conteúdos</b>	3.3 – Dependências de Dados e Controlo
	3.4 – Arquiteturas super-escalares
<b>Resultados de Aprendizagem</b>	R3.3 – Caracterizar limitações inerentes a organizações encadeadas (dependências) e conceber potenciais soluções
	R3.4 – Analisar o impacto de múltiplas unidades funcionais no desempenho

## 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** - Identifique para cada ciclo do relógio a ocupação de cada estágio do processador, para a versão PIPE- do Y86. 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						
B1						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

Usando o mesmo código, 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 valores encaminhados.

	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. Este valor é realimentado para o estágio D, podendo ser usado durante o ciclo 5.

Ciclo 5:

D : encaminhar o valor no registo W para I4

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  $0x0100 + 3 \times 6 = 0x0112$ . A instrução I4 tem 2 bytes de comprimento, logo PC = 0x0114

**EXEMPLO 2**

Identifique para cada ciclo do relógio a ocupação de cada estágio do processador, para a versão PIPE- do Y86 – injeção 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.

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 valores encaminhados.

	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. Este sinal é realimentado para o estágio D, podendo ser usado no final do ciclo 3.

Ciclo 3 - D : Encaminhar para I2 o valor produzido no estágio E

A instrução I3 depende de I1 e I2. Embora o valor de %ebx pudesse ser lido do registo M 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 injeção de uma bolha. No final do ciclo 5 o valor de %ebx pode ser lido do registo W e o valor de %eax pode ser encaminhando do estágio M.

Ciclo 5 – D: Encaminhar para I3 os valores do registo W e do estágio M



**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

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 valores encaminhados.

```

I1:    pushl %eax
I2:    popl %ebx
I3:    irmovl %20, %eax
I4:    addl %ebx, %eax
I5:    call I7
I6:    halt
I7:    subl %esi, %eax
I8:    ret
I9:    ...

```

[illegible]

Justificação	

**Exercício 4**

Considere o seguinte programa escrito em Y86:

```

    irmovl 4, %esi
    irmovl $0, %eax
    irmovl -1024, %ecx
soma: mrmovl 0(%ecx), %edx
    addl %edx, %eax
    rmmovl %eax, $1000(%ecx)
    addl %esi, %ecx
    bnz soma

```

**Exercício 4.1**

Identifique as dependências de dados existentes neste programa. Será possível reordenar as operações por forma a diminuir o número de bolhas injetadas pelo processador?

**Exercício 4.2**

Apresente o escalonamento deste programa numa arquitetura superescalar de duas vias, com a capacidade de executar uma operação aritmética/salto simultaneamente com um acesso à memória. Qual o CPI obtido na execução do programa. (Nota: faça o escalonamento apenas para as instruções executadas no ciclo)

Ciclo	Aritmética/salto	Acesso à memória
1		
2		
3		
4		
5		

**Exercício 4.3**

Desdobre o corpo do ciclo 2 vezes e efetue novamente esse escalonamento, reordenando as instruções. Considere que pode utilizar um número adicional de registos designados por %r8 ... %r15. Qual o CPI obtido agora na execução do programa.

Ciclo	Aritmética/salto	Acesso à memória
1		
2		
3		
4		
5		
6		
7		