# Módulo 8

## Super Escalaridade II

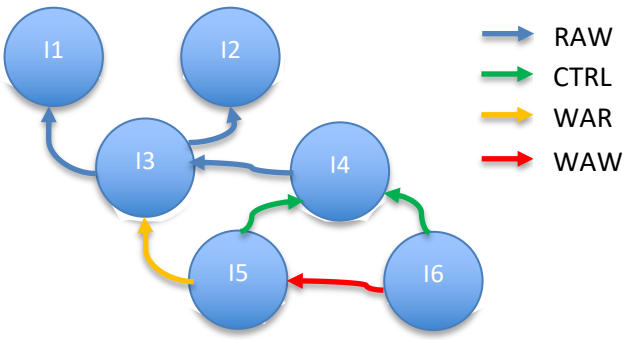
As máquinas superescalares dispõem de várias unidades funcionais, podendo seleccionar para execução várias instruções no mesmo ciclo.

As limitações à possibilidade de executar uma instrução num determinado ciclo são essencialmente:

- Disponibilidade de uma unidade funcional adequadas à instrução em causa (pode estar ocupada com a execução de outra instrução)
- Dependência de controlo: qual a instrução a executar após um salto condicional? Tipicamente resolvida com previsão de saltos
- Dependências de dados. Importa aqui considerar 3 dependências:
    - Read After Write (RAW): a instrução $I_j$ que depende de $I_i$ tem que ser executada depois desta última;
    - Write After Read (WAR) : a instrução $I_j$ que depende de $I_i$ pode ser executada em simultâneo com esta última, mas nunca antes;
    - Write After Write (WAW) : a instrução $I_j$ que depende de $I_i$ tem que ser executada depois desta última;
- O escalonamento de instruções feito pelo processador pode ser *in-order* ou *out-of-order*. No primeiro caso uma instrução $I_i$ não pode ser seleccionada para execução num ciclo anterior à execução de $I_{i+1}$.

Para criar um escalonamento identificam-se primeiro as dependências, através do respectivo grafo.

```
I1:   mov $10, %eax
I2:   mov $20, %ebx
I3:   cmp %eax, %ebx
I4:   je I6
I5:   mov $10, %ebx
I6:   mov %eax, %ebx
```




Conhecendo a micro-arquitectura o grafo de dependências é suficiente para criar o escalonamento. Suponha um processador com 3 unidades funcionais:

- UF0 e UF1 – operações sobre inteiros e saltos
- UF2 – acessos à memória

Este processador faz escalonamento *in-order* sem previsão de saltos.

| | UF0 (int+B) | UF1 (int+B) | UF2 (Mem) |
|---|---|---|---|
| 1 | I1 | I2 | |
| 2 | I3 | | |
| 3 | | I4 | |
| 4 | I5 | | |
| 5 | | I6 | |
| CPI | CPI = 5 / 6 = 0.83 | | |

**Exercício 1 –**

a) Desenhe o grafo de dependências para o código abaixo. Faça-o para duas iterações do ciclo.

```
I1:   mov $10, %ecx
I2:   mov $20, %ebx
I3:   mov 0(%ebx), %eax
I4:   add $100, %eax
I5:   mov %eax, 0(%ebx)
I6:   add $4, %ebx
I7:   dec %ecx
I8:   jnz I3
```

b) Faça o unrolling do código, processado dois elementos do vector por iteração. Desenhe o respectivo grafo de dependências apenas para uma iteração do ciclo.

c) Para as duas versões do código apresente o escalonamento, e calcule os respectivos CPIs, para o processador descrito anteriormente.

| | Sem *unrolling* | | | | *Unrolling* =2 | | |
|---|---|---|---|---|---|---|---|
| | UF0 (int+B) | UF1 (int+B) | UF2 (Mem) | | UF0 (int+B) | UF1 (int+B) | UF2 (Mem) |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | | |
| 10 | | | | | | | |
| 11 | | | | | | | |
| 12 | | | | | | | |
| 13 | | | | | | | |
| 14 | | | | | | | |
| CPI | | | | | | | |

d) Repita a alínea c) considerando agora que o processador prevê os saltos condicionais como tomados.

| | Sem *unrolling* | | | | *Unrolling* =2 | | |
|---|---|---|---|---|---|---|---|
| | UF0 (int+B) | UF1 (int+B) | UF2 (Mem) | | UF0 (int+B) | UF1 (int+B) | UF2 (Mem) |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | | |
| 10 | | | | | | | |
| 11 | | | | | | | |
| 12 | | | | | | | |
| 13 | | | | | | | |
| 14 | | | | | | | |
| CPI | | | | | | | |

e) Repita a alínea d) considerando agora que o processador faz escalonamento *out-or-order*.

| | Sem *unrolling* | | | | *Unrolling* =2 | | |
|---|---|---|---|---|---|---|---|
| | UF0 (int+B) | UF1 (int+B) | UF2 (Mem) | | UF0 (int+B) | UF1 (int+B) | UF2 (Mem) |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | | |
| 10 | | | | | | | |
| 11 | | | | | | | |
| 12 | | | | | | | |
| 13 | | | | | | | |
| 14 | | | | | | | |
| CPI | | | | | | | |

**Exercício 2 –**

a) Desenhe o grafo de dependências para o código abaixo. Faça-o para duas iterações do ciclo.

```
I1:   pop %eax
I2:   cmp $0, %eax
I3:   je I9
I4:   mov 0(%ebx), %ecx
I5:   add %eax, %ecx
I6:   mov %ecx, 0(%ebx)
I7:   add $4, %ebx
I8:   jmp I1
```

Considere um processador com 3 unidades funcionais:

UF0, UF1 – acesso à memória. Considere que operações de *load* consomem dois ciclos; operações de *store* consomem apenas 1 ciclo;

UF2 – operações sobre inteiros e saltos

Este processador faz escalonamento *in-order*, com previsão de saltos não tomados.

b) Apresente o escalonamento na tabela abaixo.

| | UF0 (Mem) | UF1 (Mem) | UF2 (int+B) |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |
| 15 | | | |
| 16 | | | |
| CPI | | | |

c) Considere agora escalonamento *out-of-order*. Apresente o escalonamento na tabela abaixo.

| | UF0 (Mem) | UF1 (Mem) | UF2 (int+B) |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |
| 15 | | | |
| 16 | | | |
| CPI | | | |