

## Lista 2 - AOC

Débora Bianca

Ewelly Fabiane

### 1) Quais as vantagens de um processador multiciclo em relação a um uniciclo?

No **multiciclo** a execução é dividida em etapas, diferente da **uniciclo**, onde é executada como um ciclo de relógio e seu ciclo é determinado pela instrução mais lenta. Em um ciclo de relógio, apenas uma das etapas é executada para uma dada instrução, com ela dividida em etapas, as instruções diferentes poderão usar quantidades diferentes de ciclos de clock.

Além de que em um processador **uniciclo** existe a duplicação de componentes (somadores, deslocadores, multiplexadores, etc), já o multiciclo favorece o reuso de hardware e

### 2) Quais as modificações necessárias em um processador multiciclo simples para que se introduza a função de pipeline?

A utilização de registradores para armazenar os valores em cada parte dedicada ao pipeline que poderão ser utilizados durante a execução do próximo ciclo.

### 3) Considerando o pipeline do MIPS (simples com MEM compartilhada para instrução e dados) e uma iteração de loop conforme o trecho de programa abaixo, relacione os conflitos que podem ocorrer e seus consequentes stalls. Qual o speedup (por iteração) para o programa e relação à versão sem pipeline?

#### Loop:

*subi \$t2, \$t2, 4*

*lw \$t1, 0(\$t2)*      Stall, t2 necessita que o valor seja calculado no subi da linha anterior.

*add \$t3, \$t1, \$t4*      Stall, aguardando o valor ser carregado em t1 pelo lw na linha 2.

*add \$t4, \$t3, \$t3*      Stall, valor de t3 ainda esta sendo calculado no add na linha 3.

*sw \$t4, 0(\$t2)*      Stall, aguardando o valor de t4 ser calculado no add na linha 4.

*beq \$t2, \$0, loop*      Stall, necessário que o valor seja carregado para t2 no sw na linha 5.

*Tempo sem pipeline é de 38ns*

*Tempo com pipeline é de 33ns*

*Logo o **speedup** é  $38/33 = 1,15$  vezes mais rápido*

**4) Na questão anterior, assuma que a memória de instruções e dados podem ser segmentadas e que o processador aplica a técnica de bypassing. Como ficarão os conflitos e seus stalls, e o speedup?**

Com as memórias segmentadas e a utilização do bypassing, não irá ocorrer mais conflitos, portanto não haverá stalls. O tempo para execução de uma iteração do loop cai para 18ns Logo o speedup é  $38/18 = 2,1$  vezes mais rápido

**5) No programa abaixo, relacione as dependências (dados, WAR, WAW e outros) existentes.**

```
1    div.d F1, F2, F3
2    sub.d F4, F5, F1
3    s.d F4, 4(F10)
4    add.d F5, F6, F7
5    div.d F4, F5, F6
```

*Dependências reais:*

DIV.D(primeiro DIV.D) e SUB.D

SUB.D e S.D

ADD.D e DIV.D (segundo DIV.D)

*Dependência de saída:*

Entre o SUB.D e DIV.D, onde SUB.D pode terminar depois de DIV.D.

WAW: Uso do F4.

*Antidependência*

Entre SUB.D e ADD.D.

ADD.D pode escrever em reg. que SUB.D lê. WAR: Uso de F5 por SUB.D.

**6) Apresente 5 processadores diferentes que trabalham paralelismo. Adicionalmente, descreva o funcionamento e o tipo de paralelismo de cada um dos processadores, bem como suas vantagens e desvantagens.**

**Whirlwind I** foi um computador desenvolvido na guerra fria e trabalhava com Bit-level paralelismo, esse tipo de paralelismo se baseia em aumentar o tamanho da palavra, porque ao fazer isso, reduz-se o número de instruções que o processador terá que executar em operações cujas variáveis têm um tamanho maior do que o comprimento da palavra.

O **Pentium** trabalha com paralelismo em nível de instrução dinâmico, em que o processador decide quais instruções desempenhar em paralelo. Por exemplo, se temos 3 instruções em que só uma depende das anteriores, as que são independentes são executadas simultaneamente e depois a terceira. Reduzindo de

3 ciclos para 2. O problema dessa técnica se dá pela crescente disparidade entre a frequência de operação do processador e o tempo de acesso à memória. Embora seja possível tolerar essa latência na memória, o preço por dissipação de energia seria desproporcional.

O **Itanium** também apresenta paralelismo em nível de instrução, só que estático (que conta como outro tipo). Nesse o compilador é quem decide quais instruções desempenhar em paralelo.

**GPUs**, tipo Nvidia, apresentam paralelismo de dados. É a execução simultânea da mesma função em múltiplos núcleos nos elementos de um conjunto de dados.

**Intel core's**, família i. paralelismo de tarefas. Distribui diversas tarefas para serem desempenhadas simultaneamente e sobre os mesmos dados. Uma vantagem seria o forward scaling que adapta o número de tasks com o tamanho do problema a ser resolvido.

**7) Em relação a memória cache. Um computador tem CPI 1 quando todos os acessos à memória acertam no cache. Loads e Stores totalizam 50% das instruções. Se a penalidade por miss é de 25 ciclos e o miss rate é 2%, qual o desempenho relativo se o computador acertar todos os acessos?**

$$\text{CPU Execution Time} = \text{IC} \times 1.0 \times \text{ClockCycle}$$

$$\text{Memory Stall Cycles} = \text{IC} \times 0.75$$

$$\text{CPU Execution Time}_{\text{cache}} = 1.75 \times \text{IC} \times \text{ClockCycle}$$

$$\frac{\text{CPU Execution Time}_{\text{cache}}}{\text{CPU Execution Time}} = \frac{1.75 \times \text{IC} \times \text{ClockCycle}}{1.0 \times \text{IC} \times \text{ClockCycle}}$$

Resultado final: 1,75

**8) Descreva os seguintes conceitos:**

**a) Write through:** escrita ao mesmo tempo no cache e na memória.

**b) Write back:** escrita somente no cache e atualização na memória somente na substituição do bloco.

**c) Localidade Temporal:** Se um item é referenciado, ele tende a ser referenciado novamente dentro de um espaço de tempo curto. Se o estudante tiver trazido o livro recentemente para sua mesa, é provável que o faça em breve novamente.

**d) Localidade Espacial:** Se um item é referenciado, itens cujos endereços sejam próximos dele tendem a ser logo referenciados.