Aluno: Joshua Kook Ho Pereira

1. A quantidade de clocks podem variar dependendo da instrução, pois no multiciclo as instruções são executadas por fases, e algumas instruções executam mais rápido por não percorrerem todas as fases. Os componentes podem ser reutilizados em uma instrução em ciclos de clock diferentes, reduzindo a repetição de componentes.
2. É preciso introduzir registradores intermediários que ficam localizados entre os componentes de cada estágio da execução de uma instrução, de modo a salvar os dados entre cada estágio.
3. Sem pipeline

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ciclo | 1 | 2 | 3 | 4 | 5 | 6 |  |
| subi $t2, $t2, 4 | IF | ID | EX | WB |  |  |  |
| lw $t1, 0($t2) |  |  |  |  | IF | ID |  |
|  | | | | | | | |
| Ciclo | 7 | 8 | 9 | 10 | 11 | 12 |  |
| lw $t1, 0($t2) | EX | MEM | WB |  |  |  |  |
| add $t3, $t1, $t4 |  |  |  | IF | ID | EX |  |
|  | | | | | | | |
| Ciclo | 13 | 14 | 15 | 16 | 17 | 18 |  |
| add $t3, $t1, $t4 | WB |  |  |  |  |  |  |
| add $t4, $t3, $t3 |  | IF | ID | EX | WB |  |  |
| sw $t4, 0($t2) |  |  |  |  |  | IF |  |
|  | | | | | | |  |
| Ciclo | 19 | 20 | 21 | 22 | 23 | 24 |  |
| sw $t4, 0($t2) | ID | EX | MEM |  |  |  |  |
| beq $t2, $0, loop |  |  |  | IF | ID | EX |  |

Com pipeline

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ciclo | 1 | 2 | 3 | 4 | 5 | 6 |  |
| subi $t2, $t2, 4 | IF | ID | EX | MEM | WB |  |  |
| lw $t1, 0($t2) |  | IF | ID | EX | MEM | WB |  |
| Conflito de dados |  |  | stall |  |  |  |  |
| add $t3, $t1, $t4 |  |  |  | IF | ID | EX |  |
| add $t4, $t3, $t3 |  |  |  |  | IF | ID |  |
| sw $t4, 0($t2) |  |  |  |  |  | IF |  |
|  | | | | | | | |
| Ciclo | 7 | 8 | 9 | 10 | 11 | 12 |  |
| add $t3, $t1, $t4 | MEM | WB |  |  |  |  |  |
| add $t4, $t3, $t3 | EX | MEM | WB |  |  |  |  |
| sw $t4, 0($t2) | ID | EX | MEM | WB |  |  |  |
| beq $t2, $0, loop | IF | ID | EX | MEM | WB |  |  |

=

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ciclo | 1 | 2 | 3 | 4 | 5 | 6 |
| div.d F1, F2, F3 | IF | ID | EX | WB |  |  |
|  | bolha | bolha | bolha | bolha | bolha | bolha |
| sub.d F4, F5, F1 |  |  |  | IF | ID | EX |
|  | | | | | | |
| Ciclo | 7 | 8 | 9 | 10 | 11 | 12 |
| sub.d F4, F5, F1 | WB |  |  |  |  |  |
|  | bolha | bolha | bolha | bolha | bolha | bolha |
| s.d F4, 4(F10) | IF | ID | EX | MEM |  |  |
| add.d F5, F6, F7 |  | IF | ID | EX | WB |  |
|  | bolha | bolha | bolha | bolha | bolha | bolha |
| div.d F4, F5, F6 |  |  |  |  | IF | ID |
|  | | | | | | |
| Ciclo | 13 | 14 | 15 | 16 | 17 | 18 |
| div.d F4, F5, F6 | EX | WB |  |  |  |  |
|  |  |  |  |  |  |  |
| div.d F1, F2, F3 | IF | ID | EX | WB |  |  |
| sub.d F4, F5, F1 |  | IF | ID | EX | WB |  |
| s.d F4, 4(F10) |  |  | IF | ID | EX | MEM |
| add.d F5, F6, F7 |  |  |  | IF | ID | EX |
| div.d F4, F5, F6 |  |  |  |  | IF | ID |
| add.d F5, F6, F7 | WB |  |  |  |  |  |
| div.d F4, F5, F6 | EX | WB |  |  |  |  |
|  |  |  |  |  |  |  |

Conflito RAW: primeiro div.d e sub.d com o registrador F1

Conflito RAW: sub.d e s.d com o registrador F4

Conflito RAW: add.d e segundo div.d com o registrador F5

Conflito WAW: sub.d e segundo div.d com o registrador F4

Conflito WAR: sub.d e add.d com o registrador F5

1. CPU tempo ideal = (Ciclos de clock da CPU + Ciclos de Stall da Memória) x Período de ciclo de Clock

- Sempre acertando

CPU tempo ideal = (IC x CPI + 0) x Período de ciclo de Clock

CPU tempo ideal = IC x 1 x Ciclo de Clock

- Situação real:

Ciclos de clock de stall de memória = IC x (1 + 0,5) x 0.02 x 25 = IC x 0,75

CPU tempo real = (IC x 1 + IC x 0,75) x Período de ciclo de Clock =>

CPU tempo real = IC x 1,75 x Período de ciclo de Clock

Comparando os desempenhos com stalls pelo ideal:

a) É quando a escrita ocorre no cache e na memória simultaneamente, assim não se trabalha com dados desatualizados, pois sempre que houver uma alteração do valor a memória principal será atualizada. Devido à essa atualização toda vez q houver alteração, existe um custo maior de tempo.

b) É quando ocorre a escrita somente no Cache, e a memória só é atualizada quando o bloco é substituído. Esse método gasta menos tempo, por não haver uma atualização contínua da memória. Entretanto, existe o risco de se utilizar dados desatualizados.

c) É quando ocorre o reconhecimento de que um dado que foi acessado recentemente tem mais chance de ser usado novamente, do que um dado que já foi usado há muito tempo. Assim os dados mais recentes são copiados na memória RAM.

d) É quando se reconhece que existe uma maior chance de se acessar um dado de posições próximas a um dado já acessado. Assim as instruções próximas às que estão sendo usadas ficam armazenadas na memória cache.