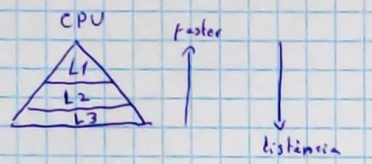
(3)

Pedro Almeida - 87205

5- A cache pode estar organizada ~~[illegible]~~ / em três maneiras: direct mapped (quando só há um sítio onde o bloco pode estar), fully associative (quando o bloco pode estar em qualquer lado) e set-associative (quando há um conjunto agregado de sítios (set) onde o bloco pode estar).
As três organizações têm o mesmo objetivo que é manter o hardware simples e os acessos à cache eficientes.

6- Normalmente são usados 3 níveis de cache para tirar partido do princípio da localidade temporal ao manter as instruções e dados recentemente usados mais perto do processador e do princípio da localidade espacial ao mover blocos para um nível mais alto em vez de só o dado pedido pelo processador.

Há assim uma hierarquia de memória em que quanto mais próximo do processador mais rápido será o acesso.



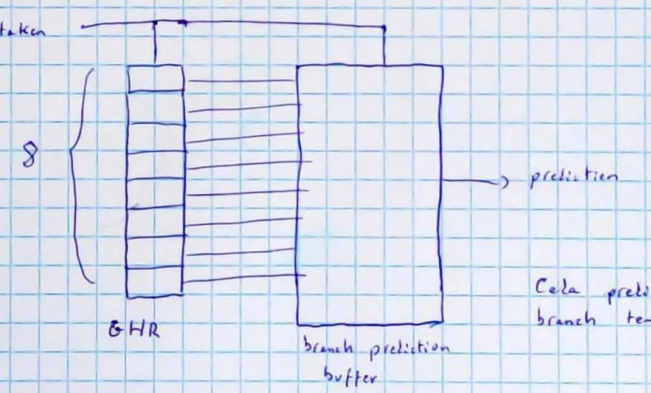
Ir à main memory é muito custoso.
Compromisso entre custo e acesso.

ii) O nível 1 está dividido em instruções e dados porque durante a execução de um programa o processador vê instruções e dados ~~[illegible]~~ e essa leitura pode acontecer ao mesmo tempo, o que num sistema unificado iria originar um hazard estrutural.

iii) Na cache L1 e L2 normalmente é aplicada a write policy write-through (os dados são escritos tanto na cache como no nível mais baixo).
Na cache L3 normalmente é aplicada a write policy write-back (os dados são escritos na cache e só mais tarde no nível mais baixo, quando o bloco é trocado).

7- branch taken/untaken

(8,2)



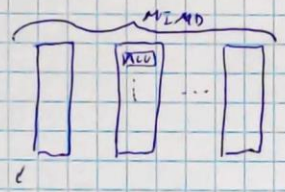
Cada preditor de branch tem 2 bits.

$2^m \cdot n \cdot 2^k \iff 2^8 \times 2 \times 2^4 = 2^{13} = 8$ Kb

(4) Pedro Almeida - 89205

8- Os hazards do tipo RAW são evitados pela execução de uma instrução apenas quando os seus operandos estão disponíveis.
Os hazards do tipo WAR e WAW são resolvidos através da renomeação de registos. A renomeação de registos elimina os hazards alterando simplesmente o nome de todos os registos de destino. Esta renomeação de registos é fornecida por reservations stations que preservam os operandos das instruções à espera de serem lançadas.

9- MIMD (multiple instructions multiple data)
SIMD (single instructions multiple data)



Uma GPU ao processar uma SIMD e cada warp pode ser visto como um MIMD

1 processador SIMD

10-

Pedro Almeida - 89205

Parte 1

1-

$$\begin{array}{l|l} 3674 & 16 \\ \hline 32 & 229 \;\; \underline{|16} \\ 47 & 16 \;\; (14) \\ 32 & 69 \\ 154 & 64 \\ 144 & (5) \\ (10) & \end{array}$$

$E5A_{16} = 0000E5A_{16}$

$$\begin{array}{l|l|l|l} 5932 & 16 & & \\ \hline 48 & 370 & 16 & \\ 113 & 32 & 23 & 16 \\ 112 & 50 & 16 & (7) \\ (12) & 48 & (7) & \\ & (2) & & \end{array}$$

$172C_{16} = 0000172C_{16}$

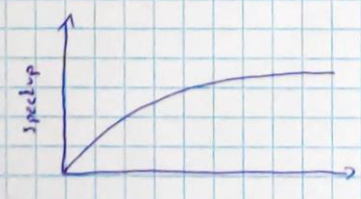
32 bits = 4 bytes, ~~(os endereços)~~ cada endereço tem 4 bytes, ou seja, para ser um endereço estar alinhado tem de ser múltiplo de 4.
Um hexadecimal pode terminar em 0, 4, 8, C.
Assim, o endereço de memória que não esta alinhado é 3674_{16}.

2-

A lei que representa a frase é a Lei de Amdahl.

$$speedup = \frac{1}{(1-P) + \frac{P}{N}}$$

Como o aumento do speedup está limitado pela fração que se pode melhorar ~~(melhorias drásticas vão ser)~~ mesmo que se consiga melhorar bastante o processador os ganhos são poucos.



Cada vez que se dobrar a capacidade de processamento, a relação do speedup vai diminuindo.

Pedro Almeida - 89205

Parte 2

3 - O throughput de execução de instruções numa versão não pipeline é: Mt
(M é o número de instruções)
Como o pipeline tira partido do paralelismo de instruções, e apesar de aumentar o tempo da execução de uma instrução isolada, o throughput é:

$(M + K - 1) \cdot t$.

Os tipos de hazards que podem acontecer são: estruturais, dados e controlo.

Estruturais - provêm de conflitos de partilha de recursos.

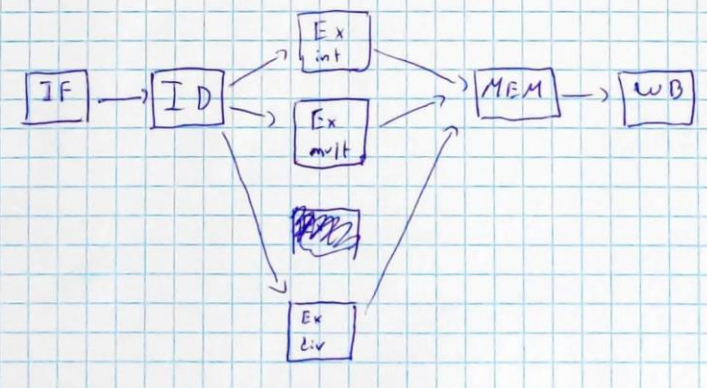
Dados - quando uma instrução depende de um resultado de uma instrução anterior que ainda não foi calculado

Controlo - numa instrução de branch, na fase ID, pode dar-se o caso dos registos para a condição de salto ainda não terem o valor certo na sequência da instrução em causa.

Para resolver os hazards estruturais são resolvidos com stalls (Unidade de interlocking gere o número de stalls necessários).
Os hazards de dados podem ser resolvidos usando forwarding, contudo isto pode não ser suficiente e ter de se efetuar stalls também.

Para resolver hazards de controlo pode-se usar predição de branches estáticos (Predicted-untaken/taken, Delayed branch slot) ou dinâmicos (Branch-Prediction Buffer).

4 -



O agendamento de instruções é estático logo não há out-of-order execution, contudo pode haver out-of-order completion, isto é, uma instrução lançada depois de outra terminar primeiro. Isto acontece porque a latência para cada operação é diferente. Por exemplo, a divisão tem uma latência muito maior que uma soma.