

Part 1

$$\begin{array}{r} 3674 \text{ } \underline{16} \\ 10 \text{ } \underline{229} \text{ } \underline{16} \\ 5 \text{ } \underline{14} \text{ } \underline{16} \\ 14 \text{ } 0 \end{array}$$

$$\begin{aligned} 3674_{10} &= \cancel{\dots} \\ &= 00000ESA_{16} \end{aligned}$$

$$\begin{array}{r} 5932 \text{ } \underline{16} \\ 12 \text{ } \underline{370} \text{ } \underline{16} \\ 2 \text{ } \underline{23} \text{ } \underline{16} \\ 7 \text{ } \underline{1} \text{ } \underline{16} \\ 1 \text{ } 0 \end{array}$$

$$5932_{10} = \cancel{\dots}0000172C_{16}$$

~~Endereço 3674₁₀ é um endereço operando?~~

~~O endereço 3674₁₀ é um endereço operando?~~

O endereço 3674₁₀ não pode representar a localização de um operando de 32 bits porque não é múltiplo de 4. 32 bits são 4 bytes, logo operandos de 32 bits encontram-se ~~na~~ na memória de 4 em 4 ~~bytes~~ endereços.

$$2. \text{ Speed up} = \frac{1}{(1-p) + \frac{p^2}{N}}$$

A lei ~~de Amdahl~~ que a equação representa é a lei de Amdahl. Diz-se que a lei de Amdahl é de "diminished returns" ~~de~~. Porque a velocidade de um sistema computacional está limitada pelos seus componentes mais lentos. Sendo assim, o Speed up, à medida que se aumenta a velocidade de uma parte das operações, aumenta cada vez menos devido às ~~demais~~ operações ~~que~~ não são aceleradas.

Part 2

3. Num processador pipelined ideal o número de CPI é 1, ou seja, em cada ciclo de relógio entra 1 e sai 1 instrução da pipeline. Isto é muito vantajoso pois, apesar de cada instrução demorar o mesmo número de ciclos de relógio, devido à paralelização de instruções em diferentes fases de execução, não haverá intervalo de tempo e possível executar mais instruções ~~do~~ na versão pipelined do que na versão não pipelined.

Numa implementação pipelined podem surgir 3 tipos de hazards:

- structural hazards
- data hazards
- control hazards

S. A cache pode estar organizada em 3 formas diferentes:

- direct mapped
- Fully associative
- Set associative

~~Organizações de cache~~

Numa organização "direct mapped" cada bloco ocupa sempre a mesma posição na cache. Esta organização é simples de implementar mas pode levar a fenômenos de thrashing.

Na organização Fully associative qualquer bloco pode ocupar qualquer linha da ~~cache~~ cache. Isto minimiza o miss rate mas ~~requer~~ requer uma maior complexidade de implementação.

A solução "set associative" é entre as duas anteriores, divide a cache em sets e cada bloco pode estar em qualquer um dos sets sempre na ~~mesma~~ mesma posição relativa. Isto diminui o miss rate em relação a uma cache "direct mapped" sem aumentar demasiado a complexidade da implementação.

b. i) ~~Cache~~ Utiliza-se tipicamente 3 níveis de cache em processadores multicore pois permite ter uma cache partilhada de dimensão maior (L_3), ~~que~~ e para cada core temos uma ~~cache~~ cache super rápida e uma intermediária.

ii) Durante a ~~execução~~ execução de um programa, o processador tem que aceder à ~~memória~~ memória para obter instruções e dados. Se isto for feito através

3. (cont) Existem várias estratégias para lidar com os hazards. Por exemplo:

- Structural hazards: Surgem quando diferentes partes do pipeline precisam do mesmo recurso ao mesmo tempo. Pode ser resolvidos separando a memória de dados da memória de instruções e lendo e escrevendo nos registos em ~~transições~~ transições diferentes do sinal de relógio, por exemplo.

- Data hazards: Surgem quando ~~a sequência de operações~~ ~~de resultados~~ pipeline força a que as operações de ~~leitura~~ leitura e escrita aconteçam fora da ordem definida no programa. Pode ser resolvidos com forwarding ou pela introdução de stalls quando não é possível de outra forma.

- Control hazards: Surgem quando existe um branch e o resultado desse branch só é conhecido ao fim de alguns ciclos de relógio. Pode ser resolvido com a introdução de stalls.

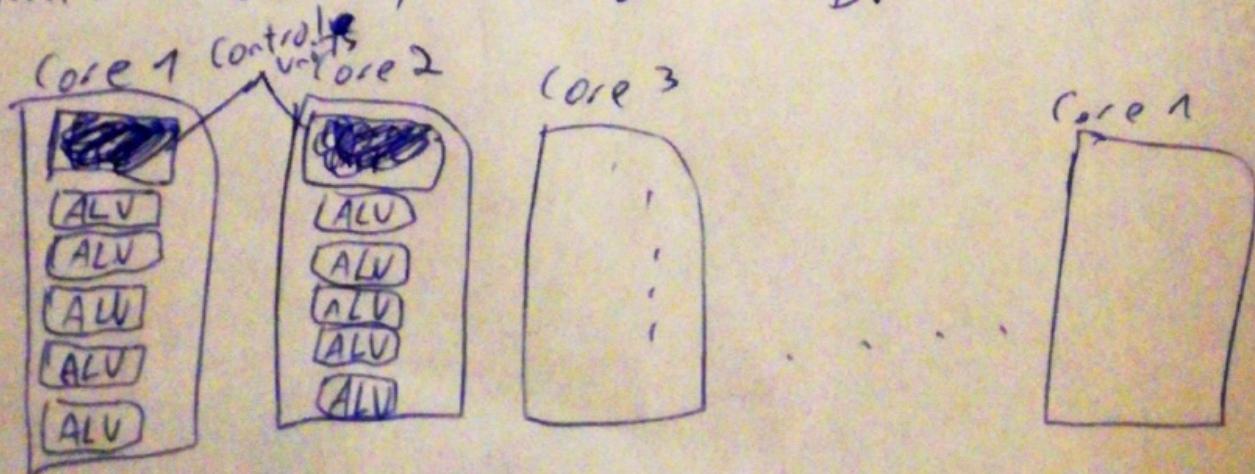
$$2 \text{ (cont.) } 2^8 \cdot 2 \cdot 2^4 = 2^{13} = 8192$$

O branch prediction buffer tem um tamanho de 8K.

8. Nesta organização, os hazards do tipo RAW são resolvidos pois uma instrução apenas é executada quando os seus operandos estão disponíveis. Os hazards de WAR and WAW são resolvidos por register renaming que altera o nome dos registos de destino de forma a que a escrita fora de ordem não afete as instruções que dependem do valor antigo.

Part 5.

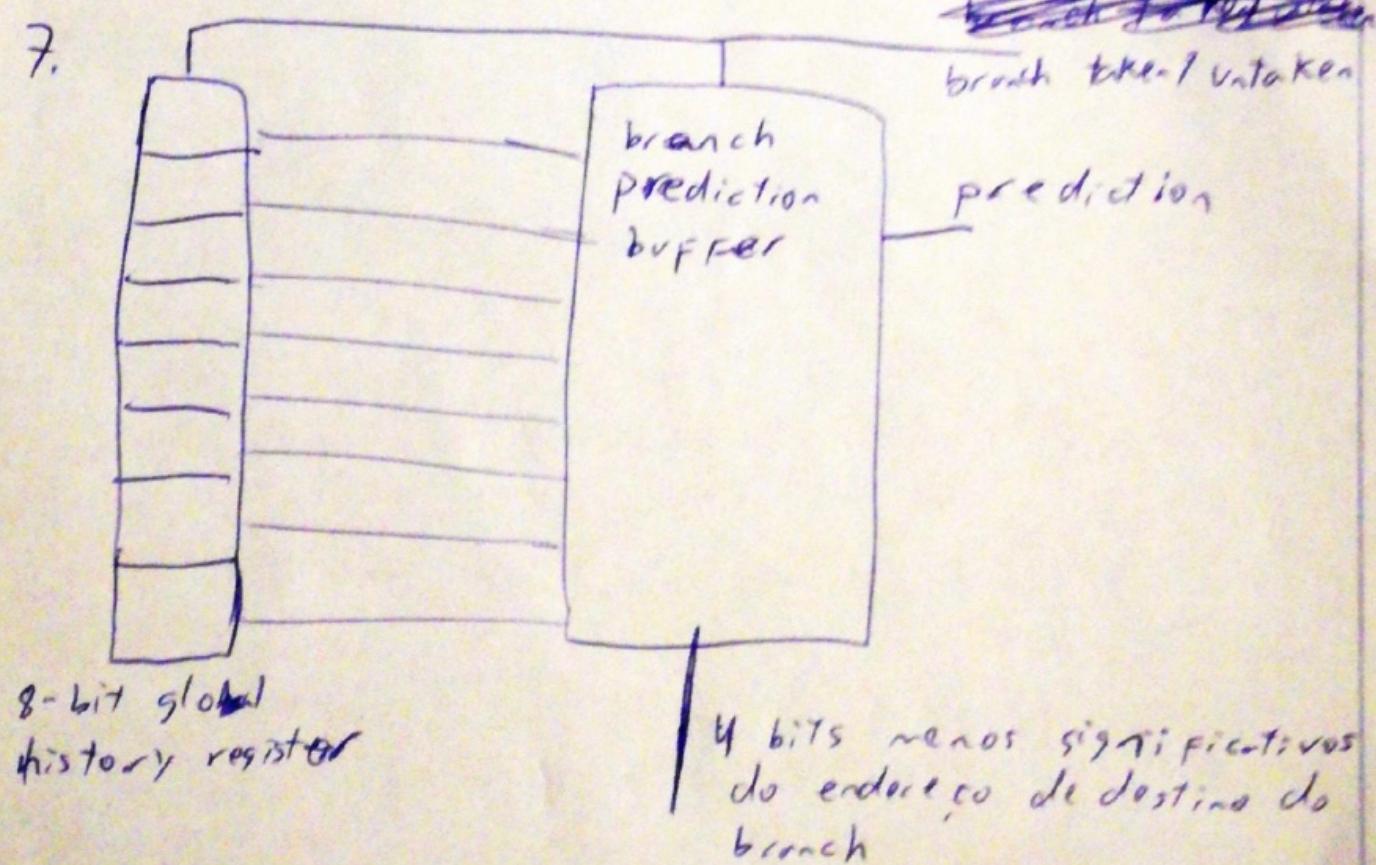
9. Cada core de uma GPU consiste numa unidade de controlo e múltiplos unidades de execução, permitindo assim executar várias streams de dados, isto constitui um SIMD. Considerando todos os cores da GPU, podemos considerar que o conjunto dos vários SIMD processadores SIMD pode ser considerado um processador MIMD.



de uma ~~cache~~ cache unica, isto pode gerar um engarrafamento de comunicação. De forma a evitar este hazard ~~estrutural~~ é comum dividir a memória entre dados e instruções no primeiro nível da cache.

- iii) ~~A~~ Nas caches L1 e L2 é necessário implementar uma política de "write-through" ~~para que~~ e ~~os~~ os blocos estejam sincronizados nos diferentes cores. ~~O~~ A cache L3 pode implementar ~~qualquer~~ qualquer política.

Part 4



~~Branch prediction buffer~~ ~~branch predictor~~ O global history register é usado para selecionar um dos branch predictors em conjunto com os últimos 4 bits do endereço de destino do branch.

4.

