Hierarquia da Memória:
Conceitos Fundamentais e Desempenho

Arquitectura de Computadores
Lic. em Engenharia Informática

Luís Paulo Santos

Hierarquia da Memória:
Conceitos Fundamentais e Desempenho

Conteúdos	8.1 – Conceitos Fundamentais
	8.2 – Hiato Processador-Memória
	8.3 – Localidade
	8.4 - Desempenho
Resultados de Aprendizagem	R8.1 – Descrever e justificar a necessidade e oportunidade da hierarquia de memória
	R8.2 – Quantificar o impacto da hierarquia da memória no desempenho da máquina

Hiato Processador-Memória

Para cada instrução:

1. Ler instrução
2. Ler operando
3. Escrever Resultado

Hiato Processador-Memória

Suponhamos um processador a executar um programa que consiste numa longa sequência de instruções inteiras:

```
addl reg, [Mem]
```

Se a instrução tiver 6 bytes de tamanho e cada inteiro 4 bytes a execução destas instruções implica um movimento de 6+2*4 = 16 bytes.

Se frequência = 2.5 GHz e o CPI=1 então são executadas $2.5*10^9$ inst/seg

A largura de banda necessária para manter o processador alimentado é de:

$2.5*10^9 * 16 =$ **40 GB/s**

Note: DDR2-xxx d

Hiato Processador-Memória

Standard name (single channel)	Peak transfer rate
DDR2-400	3200 MB/s
DDR2-800	6400 MB/s
DDR2-1066	8533 MB/s
DDR3-1066	8533 MB/s
DDR3-1600	12800 MB/s

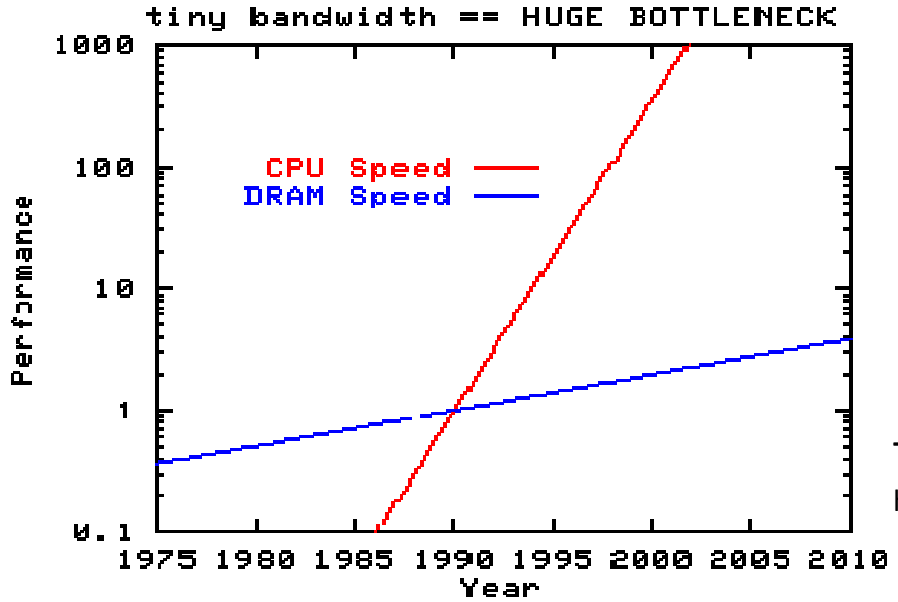
Largura de banda exigida neste exemplo: $2.5*10^9 * 16 =$ **40 GB/s**

Hiato processador-memória:

"A memória é incapaz de alimentar o processador com instruções e dados a uma taxa suficiente para o manter constantemente ocupado"

Hiato Processador-Memória

- O desempenho dos micro-processadores tem vindo a aumentar a uma taxa de cerca de 60% ao ano.
- O desempenho das memórias tem vindo a aumentar a uma taxa de perto de 10% ao ano [1,2]



The STREAM benchmark
http://www.cs.virginia.edu/stream/ref.html

[1] "The Processor-Memory bottleneck: Problems and Solutions."; *Nihar R. Mahapatra and Balakrishna Venkatrao, ACM (http://www.acm.org/crossroads/xrds5-3/pmgap.html)*

[2] " The Memory Gap and the Future of High Performance Memories"; Maurice V.Wilkes, ACM (http://www.cl.cam.ac.uk/research/dtg/attarchive/pub/docs/ORL/tr.2001.4.pdf)

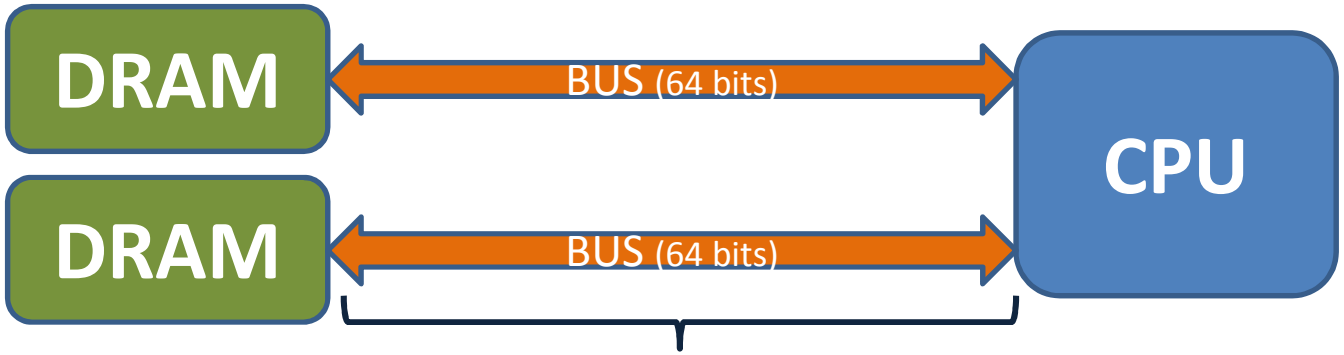
Hiato Processador-Memória

- As diferentes taxas de aumento do desempenho destes dois componentes essenciais levam a um aumento do hiato Processador-Memória (*"the memory gap"*) com o tempo
  - Em 1990 um acesso à memória central custava entre 8 a 32 ciclos do relógio
  - Em 2000 custava, numa estação Alpha 21264 667 MHz, cerca de 128 ciclos
  - O custo de cada acesso (medido em ciclos) tende a duplicar cada vez que o desempenho dos processadores duplica [2], isto é, cada período de [1,5 .. 2] anos

- O hiato processador-memória é o principal obstáculo à melhoria do desempenho dos sistemas de computação

Soluções: Largura do Barramento

- Utilizando mais do que 1 banco de memória e mais do que 1 barramento processador-memória, é possível aumentar a largura de banda teórica sem diminuir o tempo de acesso a cada módulo



Name	Peak rate (single channel)	Peak rate (dual channel)
DDR2-400	3200 MB/s	6400 MB/s
DDR2-800	6400 MB/s	12800 MB/s
DDR2-1066	8533 MB/s	17066 MB/s

Soluções: Largura do Barramento

- Actualmente, sistemas baseados em *"dual channel"* raramente atingem o dobro da largura de banda, sendo que o desempenho é frequentemente apenas 1.05 vezes superior ao de sistemas "*single channel*" [3]

- Adicionalmente, existem uma quantidade de problemas em aumentar a largura do barramento, por exemplo, aumento do número de pinos, aumento da potência de I/O, etc.

- No entanto, alguns sistemas actuais combinam mais do que 2 barramentos: 4 no caso do Mac Pro, 5 para o Nvidia 8800GTS, 6 para o Nvidia 8800GTX

[3] "Parallel Processing. Part 2: RAM and HDD."; Patrick Schmid *(http://www.tomshardware.com/reviews/PARALLEL-PROCESSING,1705.html)*

Soluções: Hierarquia de Memória

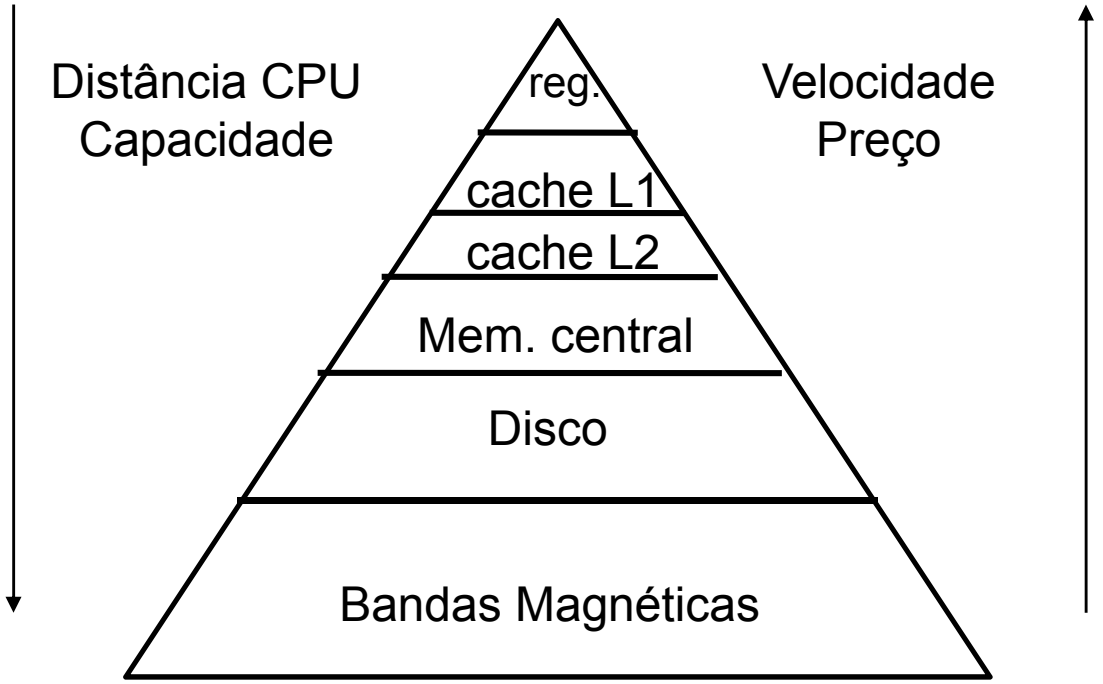
- *“smaller is faster”*
  No caso das memórias, componentes com menor capacidade (e logo menor tamanho) sofrem menos atrasos de propagação de sinal e descodificação de endereços do que componentes de grande capacidade.

  É possível também aplicar mais potência e dispender mais dinheiro por célula de memória (e.g.,*byte),* porque a capacidade é menor.

- Consequentemente, as memórias mais rápidas, exibem menor capacidade e são mais caras por *byte*.

Soluções: Hierarquia de Memória

- Dotar a máquina de vários níveis de memória, tão mais rápidos (e mais caros por *byte* e com menor capacidade) quanto mais perto se encontram do processador.
- Cada nível contêm uma cópia do código e dados mais usados em cada instante.



Hierarquia da Memória: Localidade

É o **princípio da localidade**, exibido pela maior parte dos programas no acesso à memória, que permite acelerar os acessos à mesma com a hierarquia

O **princípio da localidade** divide-se em 2 componentes:

- **Localidade temporal**
- **Localidade espacial**

Hierarquia da Memória: Localidade Temporal

Localidade Temporal – um elemento de memória acedido pelo CPU será, com grande probabilidade, acedido de novo num futuro próximo.

Exemplos: tanto as instruções dentro dos ciclos, como as variáveis usadas como contadores de ciclos, são acedidas repetidamente em curtos intervalos de tempo.

Consequência – a 1ª vez que um elemento de memória é acedido deve ser lido do nível mais baixo (por exemplo, da memória central).

Mas da 2ª vez que é acedido existem grandes hipóteses que se encontre na *cache*, evitando-se o tempo de leitura da memória central.

Hierarquia da Memória: Localidade Espacial

Localidade Espacial – se um elemento de memória é acedido pelo CPU, então elementos com endereços na proximidade serão, com grande probabilidade, acedidos num futuro próximo.

Exemplos: as instruções são acedidas em sequência, assim como, na maior parte dos programas os elementos dos *arrays*.

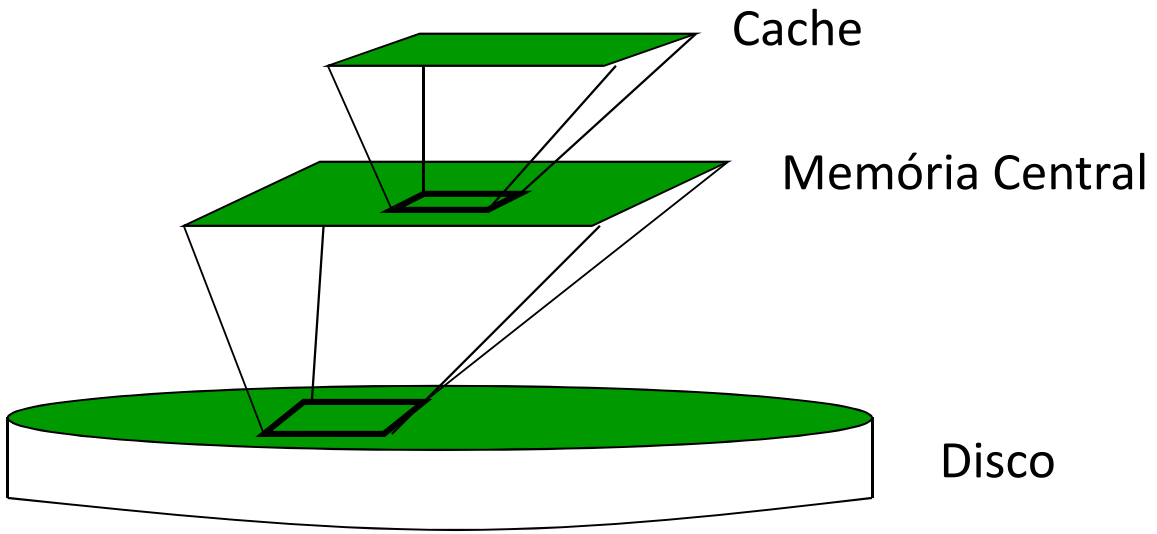
Consequência – a 1ª vez que um elemento de memória é acedido, deve ser lido do nível mais baixo (por exemplo, memória central) não apenas esse elemento, mas sim um **bloco** de elementos com endereços na sua vizinhança.

Se o processador, nos próximos ciclos, acede a um endereço vizinho do anterior (ex.: próxima instrução ou próximo elemento de um *array*) aumenta a probabilidade de esta estar na *cache*.

Hierarquia de Memória: Inclusão

Os dados contidos num nível mais próximo do processador são um sub-conjunto dos dados contidos no nível anterior.

O nível mais baixo contem a totalidade dos dados.



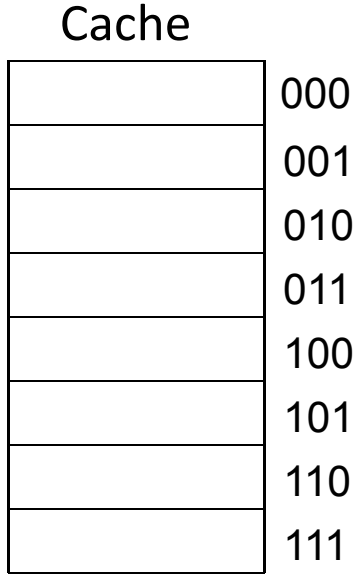
Hierarquia de Memória: Terminologia

Linha – a cache está dividida em linhas. Cada linha tem o seu endereço (índice) e tem a capacidade de um bloco

Bloco – Quantidade de informação que é transferida de cada vez da memória central para a cache. É igual à capacidade da linha.

***Hit* –** Diz-se que ocorreu um *hit* quando o elemento de memória acedido pelo CPU se encontra na cache.

***Miss* –** Diz-se que ocorreu um *miss* quando o elemento de memória acedido pelo CPU não se encontra na cache, sendo necessário lê-lo do nível inferior da hierarquia.



Hierarquia de Memória: Terminologia

Hit rate – Percentagem de *hits* ocorridos relativamente ao total de acessos à memória.

Miss rate – Percentagem de *misses* ocorridos relativamente ao total de acessos à memória. *Miss rate = (1 – hit rate)*

Hit time – Tempo necessário para aceder à cache, incluindo o tempo necessário para determinar se o elemento a que o CPU está a aceder se encontra ou não na cache.

Miss penalty – Tempo necessário para carregar um bloco da memória central para a cache quando ocorre um *miss.*

Hierarquia da memória - Desempenho

$$T_{exec} = \#I * CPI * T_{cc}$$

Como é que a hierarquia de memória influencia Texec?

#I – O número de instruções a executar depende do algoritmo, do conjunto de instruções e do compilador.

Tcc – é fixo para cada máquina. Não é alterado modificando a organização da memória.

Hierarquia da memória - Desempenho

$$T_{exec} = \#I * CPI * T_{cc}$$

$$CPI = CPI_{CPU} + CPI_{MEM}$$

CPI_{CPU} – nº de ciclos que o processador necessita, em média, para executar cada instrução;

O *hit time* considera-se incluído no CPI_{CPU}

CPI_{MEM} – nº de ciclos que o processador pára, em média, à espera de dados da memória central, por que não encontrou estes dados na cache. Estes são vulgarmente designados por ***memory stall cycles*** ou ***wait states***.

$$T_{exec} = \#I * (CPI_{CPU} + CPI_{MEM}) * T_{cc}$$

Hierarquia da memória - Desempenho

$$CPI_{MEM} = \%acessosMem * missrate * misspenalty$$

Os acessos à memória devem-se ao *fetch* de instruções e ao acesso a dados. Como estes têm comportamentos diferentes usam-se diferentes percentagens de acesso à memória e miss rate para os dois casos.

Instruções – Todas as instruções são lidas da memória, logo a % de acesso à memória é de 100%. ***missrate***$_I$ refere-se ao acesso às instruções. Esta é geralmente menor que a dos dados devido à localidade espacial.

Dados – Apenas uma determinada percentagem de instruções acede à memória (%Mem). **missrate**$_D$ refere-se ao acesso a dados.

$$CPI_{MEM} = (missrate_I + \%Mem * missrate_D) * misspenalty$$

Hierarquia da memória - Desempenho

Abreviando *missrate* por *mr* e *misspenalty* por *mp* temos

$$T_{exec} = \#I * (CPI_{CPU} + CPI_{MEM}) * T_{cc}$$

$$CPI_{MEM} = (mr_I + \%Mem * mr_D) * mp$$

substituindo

$$T_{exec} = \#I * [CPI_{CPU} + (mr_I + \%Mem * mr_D) * mp] * T_{cc}$$

NOTA: A *miss penalty (mp)* tem que ser expressa em ciclos do *clock*.

Hierarquia da memória - Desempenho

Considere uma máquina com uma *miss rate* de 4% para instruções, 5% para dados e uma *miss penalty* de 50 ciclos. Assuma ainda que 40% das instruções são *loads* ou *stores*, e que o CPI_{CPU} é 1. Qual o CPI total?

$$CPI = CPI_{CPU} + CPI_{MEM} = CPI_{CPU} + (mr_I + \%Mem * mr_D) * mp$$

$$CPI = 1 + (0.04 + 0.4 * 0.05) * 50 = 1 + 3 = 4$$

Se a frequência do relógio for de 1 GHz e o programa executar 10^9 instruções qual o tempo de execução?

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 4 * \frac{1}{10^9} = 4s$$

Hierarquia da memória - Desempenho

Considere um programa com as características apresentadas na tabela, a executar numa máquina com memória de tempo de acesso 0. Se a frequência do processador for 1 GHz, qual o CPI médio e o tempo de execução?

Instrução	Nº Instruções	CPI
Cálculo	$3*10^8$	1,1
Acesso à Mem.	$6*10^8$	2,5
Salto	$1*10^8$	1,7
TOTAL:	10^9	

$$CPI = CPI_{CPU} + CPI_{MEM} = (3*1.1 + 6*2.5 + 1*1.7)/10 + 0 = 2$$

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 2 * \frac{1}{10^9} = 2s$$

Hierarquia da memória - Desempenho

Considere o mesmo programa e máquina do acetato anterior, mas agora com um tempo de acesso à memória de 10 ns (por palavra ou instrução). Suponha ainda que esta máquina não tem cache. Qual o CPI efectivo e T_{exec}?

$$CPI = CPI_{CPU} + CPI_{MEM} = CPI_{CPU} + (mr_I + \%Mem * mr_D) * mp$$

Se a máquina não tem cache, então $mr_I = mr_D = 100\%$.

Da tabela tiramos que *%Mem* = 60%.

mp expresso em ciclos do relógio é 10/1 = 10 ciclos (f=1 GHz)

$$CPI = CPI_{CPU} + CPI_{MEM} = 2 + (1 + 0.6 * 1) * 10 = 2 + 16 = 18$$

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 18 * \frac{1}{10^9} = 18s$$

Hierarquia da memória - Desempenho

Considere agora que existe uma *cache* com linhas de 4 palavras; a *miss rate* de acesso às instruções é de 6% e de acesso aos dados é de 10%; o tempo de acesso à memória central é constituído por uma latência de 40 ns mais 10 ns por palavra. Qual o CPI médio e o tempo de execução?

mp = 40 + 10*4 = 80 ns ; em ciclos *mp* = 80/1 = 80 ciclos

$$CPI = CPI_{CPU} + CPI_{MEM} = 2 + (0.06 + 0.6 * 0.1) * 80 = 2 + 9.6 = 11.6$$

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 11.6 * \frac{1}{10^9} = 11.6s$$

Hierarquia da memória - Desempenho

Suponha que a capacidade da *cache* é aumentada para o dobro, resultando numa *miss rate* de acesso às instruções de 3.2% e acesso aos dados de 8%. No entanto, o tempo de acesso à cache (*hit* time) também aumenta, resultando num CPI_{CPU} de 2.5 . Qual o CPI médio e o tempo de execução?

$$CPI = CPI_{CPU} + CPI_{MEM} = 2.5 + (0.032 + 0.6 * 0.08) * 80 = 2.5 + 6.4 = 8.9$$

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 8.9 * \frac{1}{10^9} = 8.9s$$

Hierarquia da memória - Desempenho

Para tirar maior partido da localidade espacial aumentou-se o número de palavras por linha de 4 para 8, reduzindo a *miss rate* de instruções para 1% e de dados para 6%. O tempo de acesso à memória central é composto por uma latência de 40 ns mais 10 ns por palavra. Qual o CPI médio e o tempo de execução?

mp = 40 + 10*8 = 120 ns ; em ciclos *mp* = 120/1 = 120 ciclos

$$CPI = CPI_{CPU} + CPI_{MEM} = 2.5 + (0.01 + 0.6 * 0.06) * 120 = 2.5 + 5.52 = 8.02$$

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 8.02 * \frac{1}{10^9} = 8.02s$$

Hierarquia da memória - Desempenho

Para reduzir a *miss penalty* a memória central foi substituída por outra com uma latência de 40 ns e 5 ns por palavra. Qual o CPI médio e o tempo de execução?

mp = 40 + 5*8 = 80 ns ; em ciclos *mp* = 80/1 = 80 ciclos

$$CPI = CPI_{CPU} + CPI_{MEM} = 2.5 + (0.01 + 0.6 * 0.06) * 80 = 2.5 + 3.68 = 6.18$$

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 6.18 * \frac{1}{10^9} = 6.18s$$

Hierarquia da memória - Desempenho

Finalmente o processador foi substituído por outro com uma frequência de 3 GHz, sem que a memória tenha sofrido qualquer alteração. Qual o CPI médio e o tempo de execução?

O ciclo do relógio é agora de 0.33 ns, logo *mp* = 80/0.33=240 ciclos

$$CPI = CPI_{CPU} + CPI_{MEM} = 2.5 + (0.01 + 0.6 * 0.06) * 240 = 2.5 + 11.04 = 13.54$$

$$T_{exec} = \#I * CPI * T_{cc} = 10^9 * 13.54 * \frac{1}{3 * 10^9} = 4.513s$$