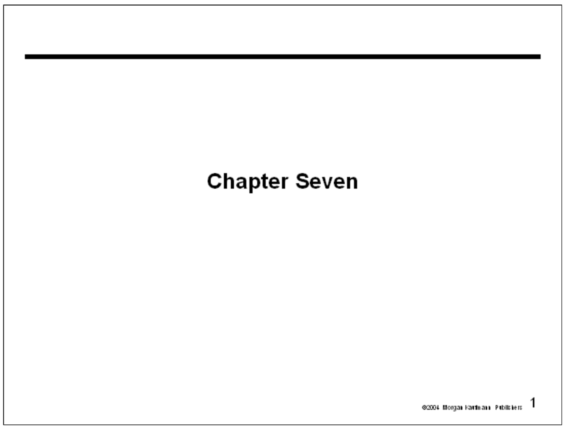
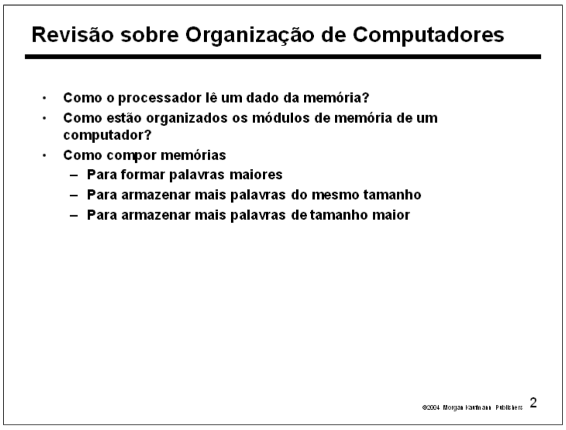
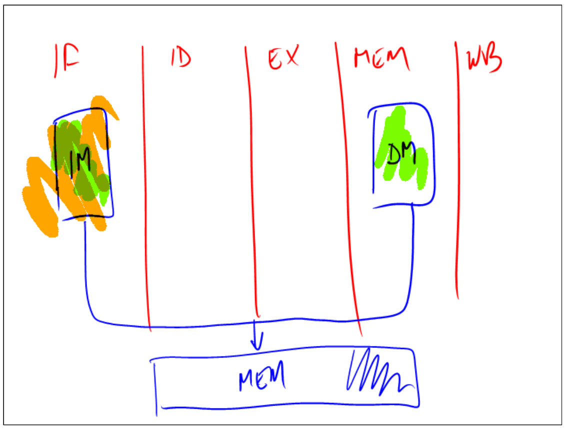


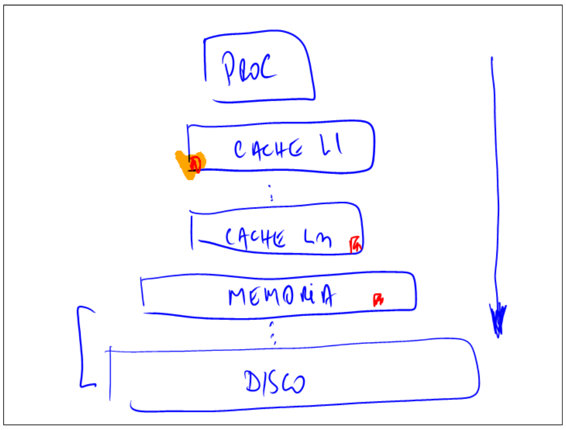
Chapter Seven

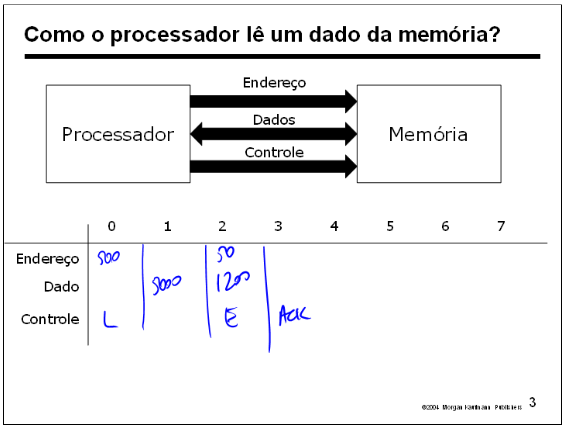


Revisão sobre Organização de Computadores

- Como o processador lê um dado da memória?
- Como estão organizados os módulos de memória de um computador?
- Como compor memórias
  - Para formar palavras maiores
  - Para armazenar mais palavras do mesmo tamanho
  - Para armazenar mais palavras de tamanho maior

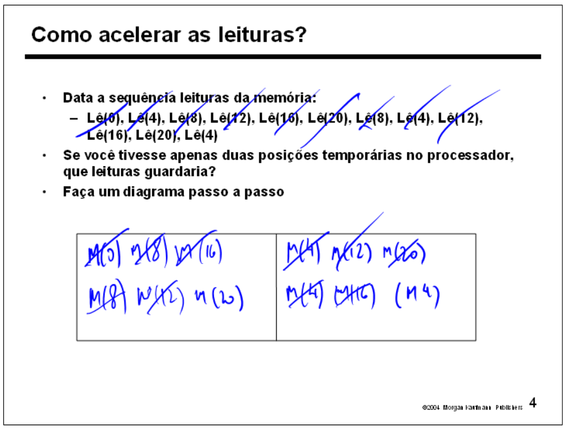






Como o processador lê um dado da memória?

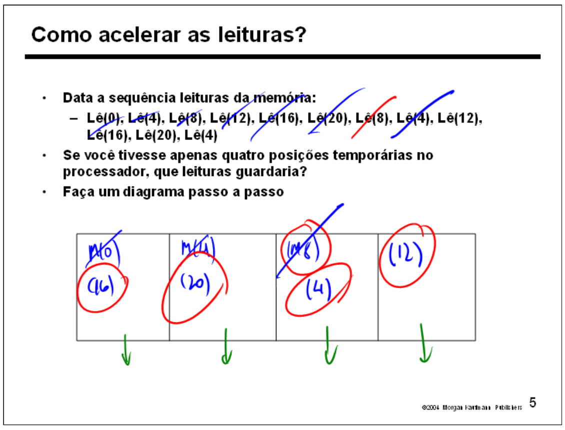
	0	1	2	3	4	5	6	7
Endereço	500		50					
Dado		5000	1200					
Controle	L		E	Ack				



Como acelerar as leituras?

- Data a sequência leituras da memória:
  - Lê(0), Lê(4), Lê(8), Lê(12), Lê(16), Lê(20), Lê(8), Lê(4), Lê(12), Lê(16), Lê(20), Lê(4)
- Se você tivesse apenas duas posições temporárias no processador, que leituras guardaria?
- Faça um diagrama passo a passo

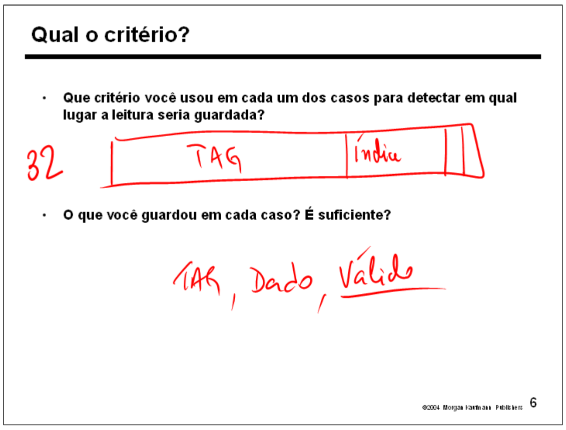
~~M(0)~~ ~~M(8)~~ ~~M(16)~~ ~~M(8)~~ ~~M(12)~~ M(20)	~~M(4)~~ ~~M(12)~~ ~~M(20)~~ ~~M(4)~~ ~~M(16)~~ (M 4)



Como acelerar as leituras?

- Data a sequência leituras da memória:
  - Lê(0), Lê(4), Lê(8), Lê(12), Lê(16), Lê(20), Lê(8), Lê(4), Lê(12), Lê(16), Lê(20), Lê(4)
- Se você tivesse apenas quatro posições temporárias no processador, que leituras guardaria?
- Faça um diagrama passo a passo

©2004 Morgan Kaufmann Publishers 5



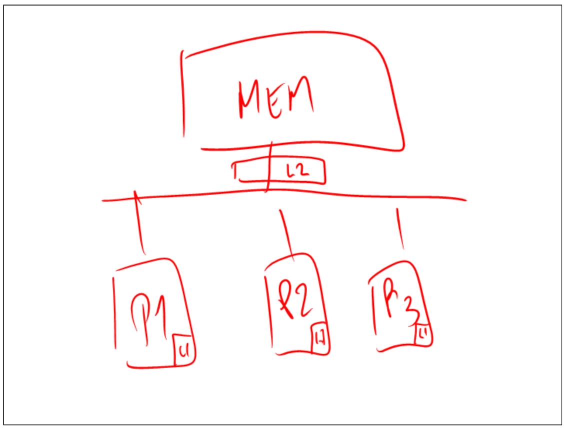
Qual o critério?

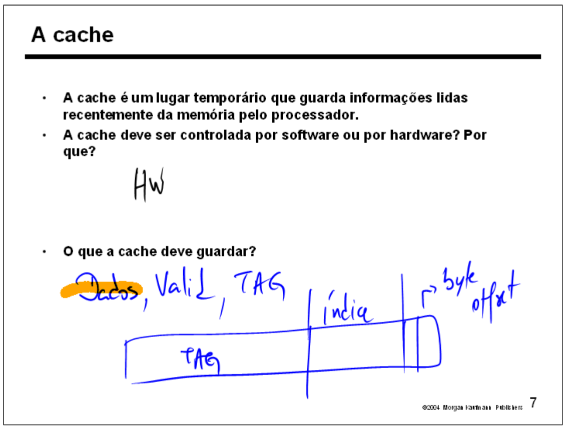
- Que critério você usou em cada um dos casos para detectar em qual lugar a leitura seria guardada?

- O que você guardou em cada caso? É suficiente?

TAG, Dado, Válido

©2004 Morgan Kaufmann Publishers 6





A cache

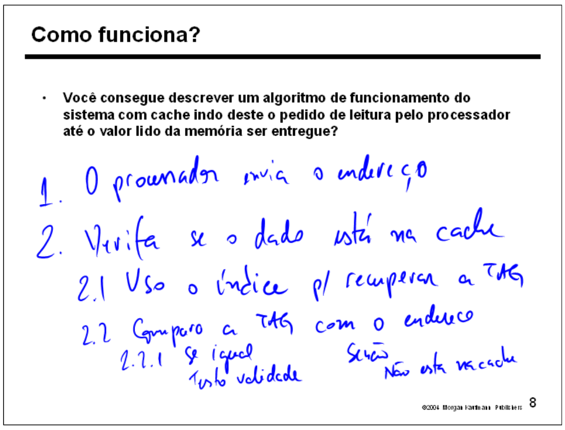
- A cache é um lugar temporário que guarda informações lidas recentemente da memória pelo processador.
- A cache deve ser controlada por software ou por hardware? Por que?

HW

- O que a cache deve guardar?

Dados, Valid, TAG

©2004 Morgan Kaufmann Publishers 7

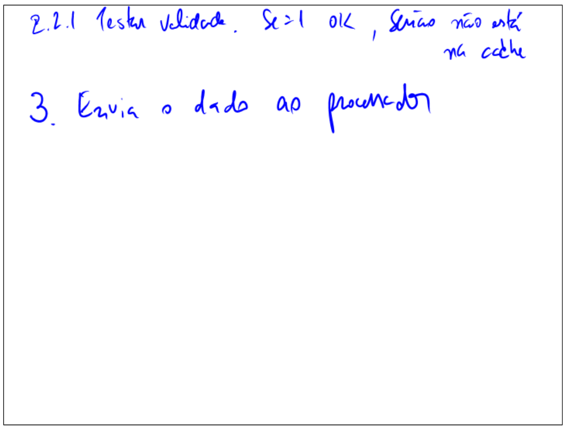


Como funciona?

- Você consegue descrever um algoritmo de funcionamento do sistema com cache indo deste o pedido de leitura pelo processador até o valor lido da memória ser entregue?

1. O processador envia o endereço
2. Verifica se o dado está na cache
   2.1 Uso o índice p/ recuperar a TAG
   2.2 Comparo a TAG com o endereço
      2.2.1 Se igual Testo validade Senão Não está na cache

©2004 Morgan Kaufmann Publishers 8



2.2.1 Testar validade. Se =1 OK, Senão não está na cache

3. Envia o dado ao processador

Caches de Instruções e de Dados

- As duas memórias que vimos no *datapath* monociclo e no *pipeline* são, na realidade, caches. Os processadores têm uma cache de instruções e uma de dados. A cache de dados difere da cache de instruções pela capacidade de suportar escritas. O que você modificaria no seu modelo anterior para suportar escritas?

0. Verifica se o dado está na cache

1. Acessa a cache usando o índice p/ escrever TAG, Dado, Valid = 1

2. Write-through ou Write-back

Unidade de armazenamento (bloco)

- A unidade de armazenamento da cache é chamada de bloco. Toda transferência entre a cache e a memória é feita por blocos. Qual o tamanho do bloco de sua cache? Qual o tamanho dos dados transferidos entre a cache e o processador? Justifique as duas respostas.

- Faz sentido ter um bloco menor que uma palavra? E maior?

Hit e Miss

- Quando uma cache contém o dado que o processador solicitou, dizemos que acontece um HIT na cache. Quando ela não contém o dado, dizemos que acontece um MISS.
- Se uma cache tem uma taxa de MISS de 5%, qual será a taxa de HIT?

95%

- Se o tempo de acesso à cache é de 1 ciclo e o de acesso à memória é 100 ciclos, quanto tempo será gasto na leitura de 1000 instruções com a taxa de HIT acima?

HIT: 950 ciclos 5950 ciclos

MISS: 50 × 100 = 5000 ciclos

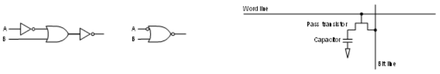
- Se um processador tem o CPI = 1 sem considerar a cache, qual será o novo CPI considerando os dados acima?

5,95

Memories: Review

- SRAM:
  - value is stored on a pair of inverting gates
  - very fast but takes up more space than DRAM (4 to 6 transistors)

- DRAM:
  - value is stored as a charge on capacitor (must be refreshed)
  - very small but slower than SRAM (factor of 5 to 10)



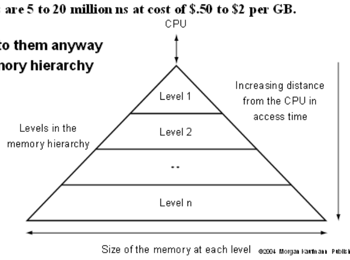
Exploiting Memory Hierarchy

- Users want large and fast memories!

SRAM access times are .5 – 5ns at cost of $4000 to $10,000 per GB.
DRAM access times are 50-70ns at cost of $100 to $200 per GB.
Disk access times are 5 to 20 million ns at cost of $.50 to $2 per GB.

2004

- Try and give it to them anyway
  - build a memory hierarchy


CPU
Level 1
Level 2
..
Level n
Levels in the memory hierarchy
Increasing distance from the CPU in access time
Size of the memory at each level

Locality

- A principle that makes having a memory hierarchy a good idea

- If an item is referenced,

temporal locality: it will tend to be referenced again soon
spatial locality: nearby items will tend to be referenced soon.

Why does code have locality?

- Our initial focus: two levels (upper, lower)
  - block: minimum unit of data
  - hit: data requested is in the upper level
  - miss: data requested is not in the upper level

Cache

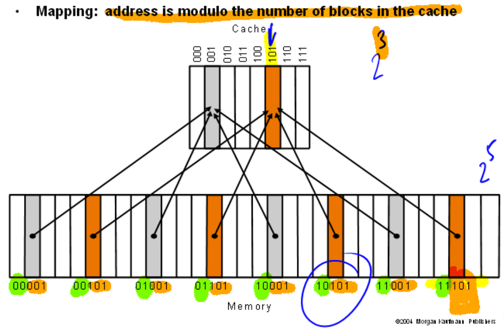
- Two issues:
  - How do we know if a data item is in the cache?
  - If it is, how do we find it?
- Our first example:
  - block size is one word of data
  - "direct mapped"

For each item of data at the lower level, there is exactly one location in the cache where it might be.

e.g., lots of items at the lower level share locations in the upper level

Direct Mapped Cache

- Mapping: address is modulo the number of blocks in the cache

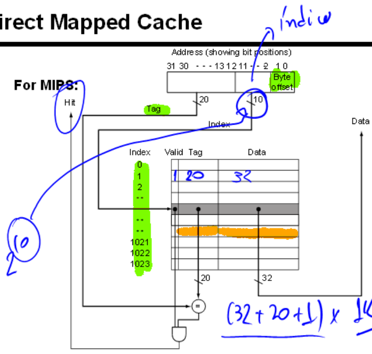


Direct Mapped Cache

índice

- For MIPS:

4KB → TAMANHO



$(32+20+1) \times 1K$

53Kbits

What kind of locality are we taking advantage of?

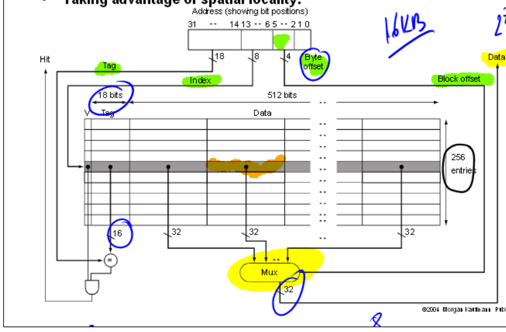
Direct Mapped Cache

TAMANHO: $2^8 \times 2^9 = 2^{17}$ bits

- Taking advantage of spatial locality:

16KB

2^7 Kbits



8

Exemplo

Index	V	Tag	Valor
000			
001			
010			
011			
100			
101			
110			
111			

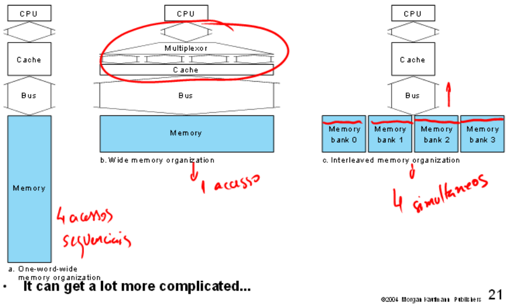
Hits vs. Misses

- Read hits
  - this is what we want!
- Read misses
  - stall the CPU, fetch block from memory, deliver to cache, restart
- Write hits:
  - can replace data in cache and memory (write-through)
  - write the data only into the cache (write-back the cache later)
- Write misses:
  - read the entire block into the cache, then write the word

Hardware Issues

bloco = 4 palavras

- Make reading multiple words easier by using banks of memory



4 acessos sequenciais

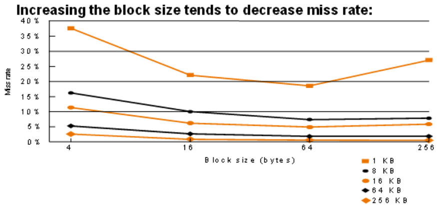
1 acesso

4 simultaneos

- It can get a lot more complicated...

Performance

- Increasing the block size tends to decrease miss rate:



- Use split caches because there is more spatial locality in code:

Program	Block size in words	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

Performance

- Simplified model:

execution time = (execution cycles + stall cycles) × cycle time

stall cycles = # of instructions × miss ratio × miss penalty

- Two ways of improving performance:
  - decreasing the miss ratio ✓
  - decreasing the miss penalty ↓

What happens if we increase block size?

3 C's

Miss Compulsório → aumentando tam bloco

+

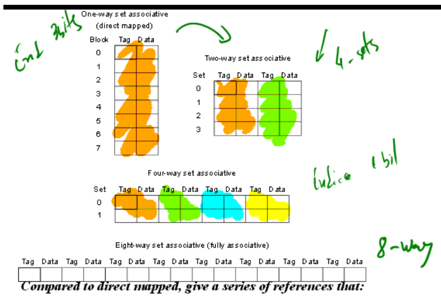
Miss Capacidade

+

Miss Conflito → associatividade

100%

Decreasing miss ratio with associativity



End 3bits

4 sets

índice 1 bit

8-way

Compared to direct mapped, give a series of references that:

- *results in a lower miss ratio using a 2-way set associative cache*
- *results in a higher miss ratio using a 2-way set associative cache*

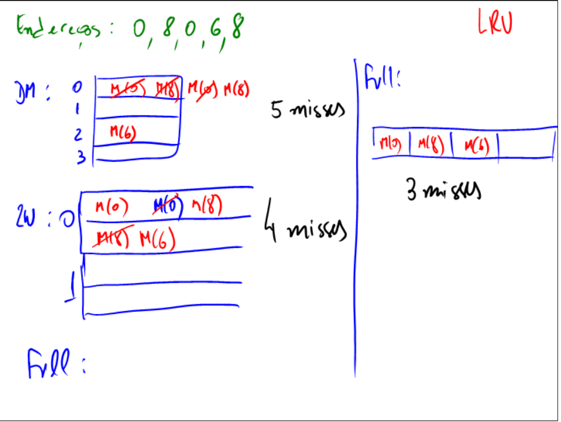
assuming we use the "least recently used" replacement strategy

DM (8 linhas): End (12) → 4 (12 mod 8) 1 busca

2way (4 linhas): End (12) → (12 mod 4) = 0

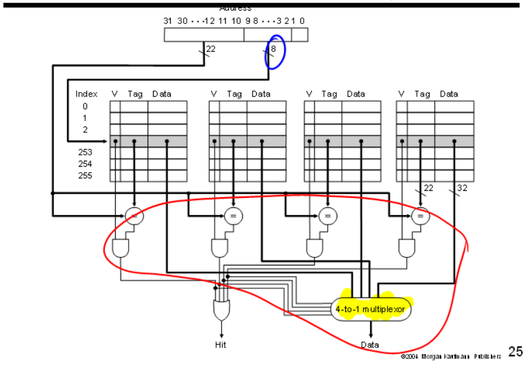
2 buscas

Fully : qualquer

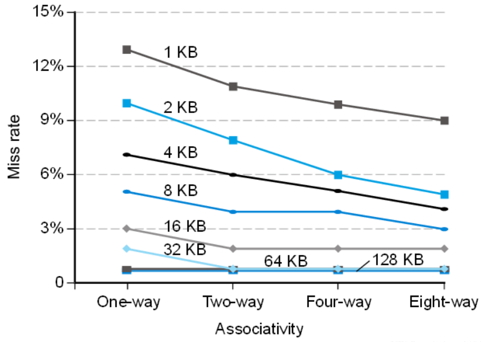


An implementation

TAMANHO: $2^{8} \times 4 \times 4 = 2^{12}$



Performance



Decreasing miss penalty with multilevel caches

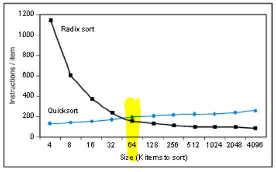
- **Add a second level cache:**
  - **often primary cache is on the same chip as the processor**
  - **use SRAMs to add another cache above primary memory (DRAM)**
  - **miss penalty goes down if data is in 2nd level cache**

- **Example:**
  - CPI of 1.0 on a 5 Ghz machine with a 5% miss rate, 100ns DRAM access
  - Adding 2nd level cache with 5ns access time decreases miss rate to .5%

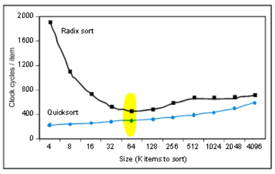
- **Using multilevel caches:**
  - **try and optimize the hit time on the 1st level cache**
  - **try and optimize the miss rate on the 2nd level cache**

Cache Complexities

- **Not always easy to understand implications of caches:**



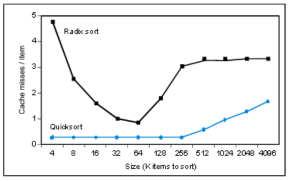
Theoretical behavior of Radix sort vs. Quicksort



Observed behavior of Radix sort vs. Quicksort

Cache Complexities

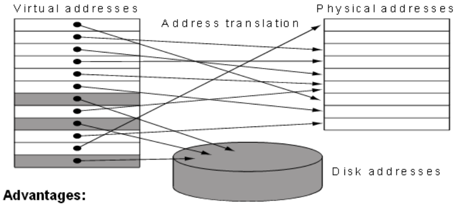
- Here is why:



- Memory system performance is often critical factor
  - multilevel caches, pipelined processors, make it harder to predict outcomes
  - Compiler optimizations to increase locality sometimes hurt ILP
- Difficult to predict best algorithm: need experimental data

Virtual Memory

- Main memory can act as a cache for the secondary storage (disk)



- Advantages:
  - illusion of having more physical memory
  - program relocation
  - protection

1) Qtas conversões de endereço um lw precisa?

2) E um add?

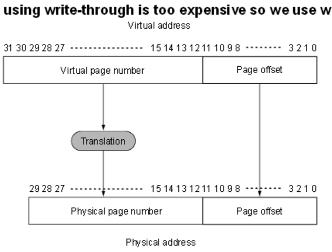
3) Se a tabela fica na memória, qtos acessos são necessários nos 2 casos?

4) Quem gerencia cache? SW/HW?

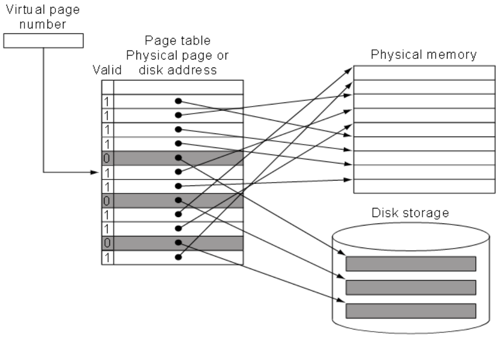
5) Quem gerencia memória virtual? SW/HW?

Pages: virtual memory blocks

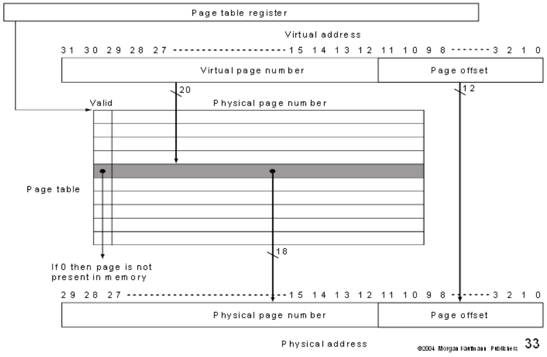
- Page faults: the data is not in memory, retrieve it from disk
  - huge miss penalty, thus pages should be fairly large (e.g., 4KB)
  - reducing page faults is important (LRU is worth the price)
  - can handle the faults in software instead of hardware
  - using write-through is too expensive so we use writeback

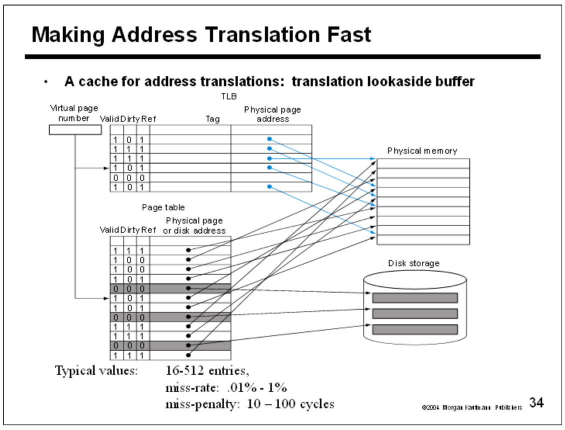


Page Tables



Page Tables

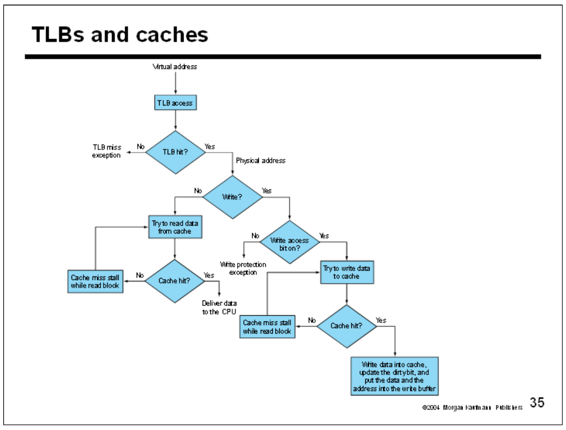




Making Address Translation Fast

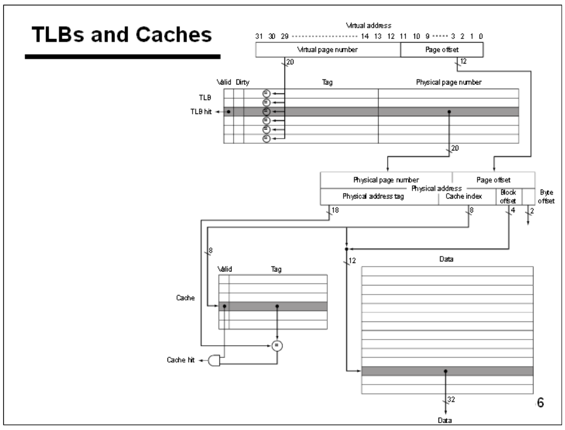
- A cache for address translations: translation lookaside buffer

Typical values: 16-512 entries,
miss-rate: .01% - 1%
miss-penalty: 10 – 100 cycles

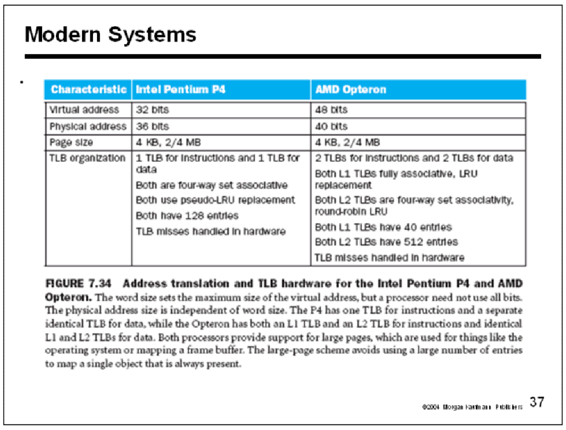


TLBs and caches

Virtual address
TLB access
TLB hit?
No
TLB miss exception
Yes
Physical address
Write?
No
Try to read data from cache
Yes
Write access bit on?
No
Write protection exception
Yes
Try to write data to cache
Cache hit?
No
Cache miss stall while read block
Yes
Deliver data to the CPU
Cache hit?
No
Cache miss stall while read block
Yes
Write data into cache, update the dirtybit, and put the data and the address into the write buffer



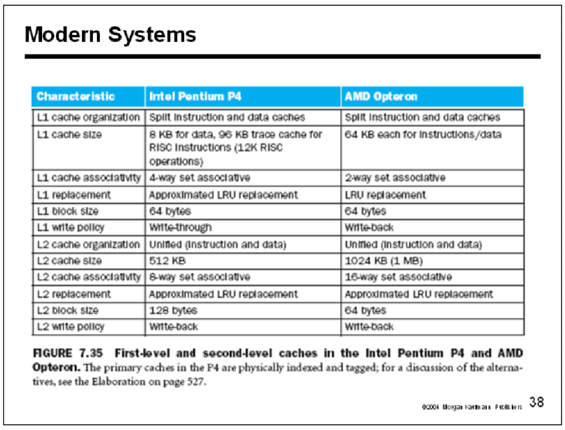
TLBs and Caches



Modern Systems

Characteristic	Intel Pentium P4	AMD Opteron
Virtual address	32 bits	48 bits
Physical address	36 bits	40 bits
Page size	4 KB, 2/4 MB	4 KB, 2/4 MB
TLB organization	1 TLB for instructions and 1 TLB for data Both are four-way set associative Both use pseudo-LRU replacement Both have 128 entries TLB misses handled in hardware	2 TLBs for instructions and 2 TLBs for data Both L1 TLBs fully associative, LRU replacement Both L2 TLBs are four-way set associativity, round-robin LRU Both L1 TLBs have 40 entries Both L2 TLBs have 512 entries TLB misses handled in hardware

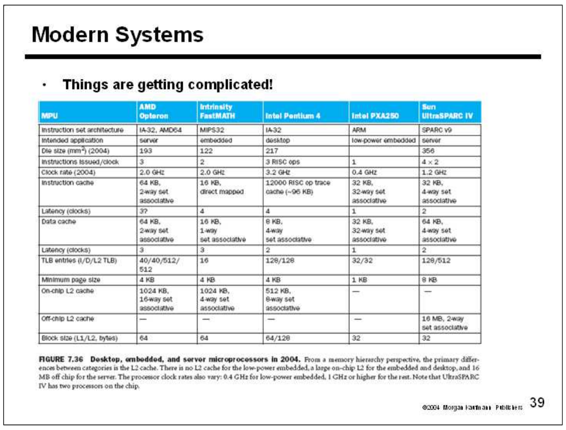
FIGURE 7.34 Address translation and TLB hardware for the Intel Pentium P4 and AMD Opteron. The word size sets the maximum size of the virtual address, but a processor need not use all bits. The physical address size is independent of word size. The P4 has one TLB for instructions and a separate identical TLB for data, while the Opteron has both an L1 TLB and an L2 TLB for instructions and identical L1 and L2 TLBs for data. Both processors provide support for large pages, which are used for things like the operating system or mapping a frame buffer. The large-page scheme avoids using a large number of entries to map a single object that is always present.



Modern Systems

Characteristic	Intel Pentium P4	AMD Opteron
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	8 KB for data, 96 KB trace cache for RISC instructions (12K RISC operations)	64 KB each for instructions/data
L1 cache associativity	4-way set associative	2-way set associative
L1 replacement	Approximated LRU replacement	LRU replacement
L1 block size	64 bytes	64 bytes
L1 write policy	Write-through	Write-back
L2 cache organization	Unified (instruction and data)	Unified (instruction and data)
L2 cache size	512 KB	1024 KB (1 MB)
L2 cache associativity	8-way set associative	16-way set associative
L2 replacement	Approximated LRU replacement	Approximated LRU replacement
L2 block size	128 bytes	64 bytes
L2 write policy	Write-back	Write-back

FIGURE 7.35 First-level and second-level caches in the Intel Pentium P4 and AMD Opteron. The primary caches in the P4 are physically indexed and tagged; for a discussion of the alternatives, see the Elaboration on page 527.

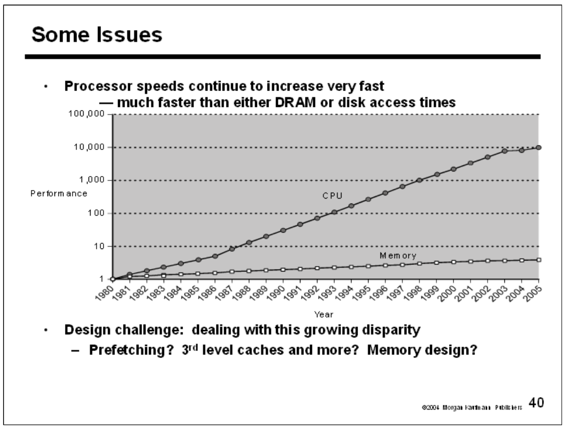


Modern Systems

- Things are getting complicated!

MPU	AMD Opteron	Intrinsity FastMATH	Intel Pentium 4	Intel PXA250	Sun UltraSPARC IV
Instruction set architecture	IA-32, AMD64	MIPS32	IA-32	ARM	SPARC v9
Intended application	server	embedded	desktop	low-power embedded	server
Die size (mm^2) (2004)	193	122	217		356
Instructions issued/clock	3	2	3 RISC ops	1	4 × 2
Clock rate (2004)	2.0 GHz	2.0 GHz	3.2 GHz	0.4 GHz	1.2 GHz
Instruction cache	64 KB, 2-way set associative	16 KB, direct mapped	12000 RISC op trace cache (~96 KB)	32 KB, 32-way set associative	32 KB, 4-way set associative
Latency (clocks)	3?	4	4	1	2
Data cache	64 KB, 2-way set associative	16 KB, 1-way set associative	8 KB, 4-way set associative	32 KB, 32-way set associative	64 KB, 4-way set associative
Latency (clocks)	3	3	2	1	2
TLB entries (I/D/L2 TLB)	40/40/512/512	16	128/128	32/32	128/512
Minimum page size	4 KB	4 KB	4 KB	1 KB	8 KB
On-chip L2 cache	1024 KB, 16-way set associative	1024 KB, 4-way set associative	512 KB, 8-way set associative	—	—
Off-chip L2 cache	—	—	—	—	16 MB, 2-way set associative
Block size (L1/L2, bytes)	64	64	64/128	32	32

FIGURE 7.36 Desktop, embedded, and server microprocessors in 2004. From a memory hierarchy perspective, the primary differences between categories is the L2 cache. There is no L2 cache for the low-power embedded, a large on-chip L2 for the embedded and desktop, and 16 MB off chip for the server. The processor clock rates also vary: 0.4 GHz for low-power embedded, 1 GHz or higher for the rest. Note that UltraSPARC IV has two processors on the chip.



Some Issues

- Processor speeds continue to increase very fast
  — much faster than either DRAM or disk access times

- Design challenge: dealing with this growing disparity
  - Prefetching? 3rd level caches and more? Memory design?

40