MO401

Arquitetura de Computadores I

2006

Prof. Paulo Cesar Centoducatte

ducatte@ic.unicamp.br

www.ic.unicamp.br/~ducatte

MO401

Arquitetura de Computadores I

Projeto de Hierarquia de Memória

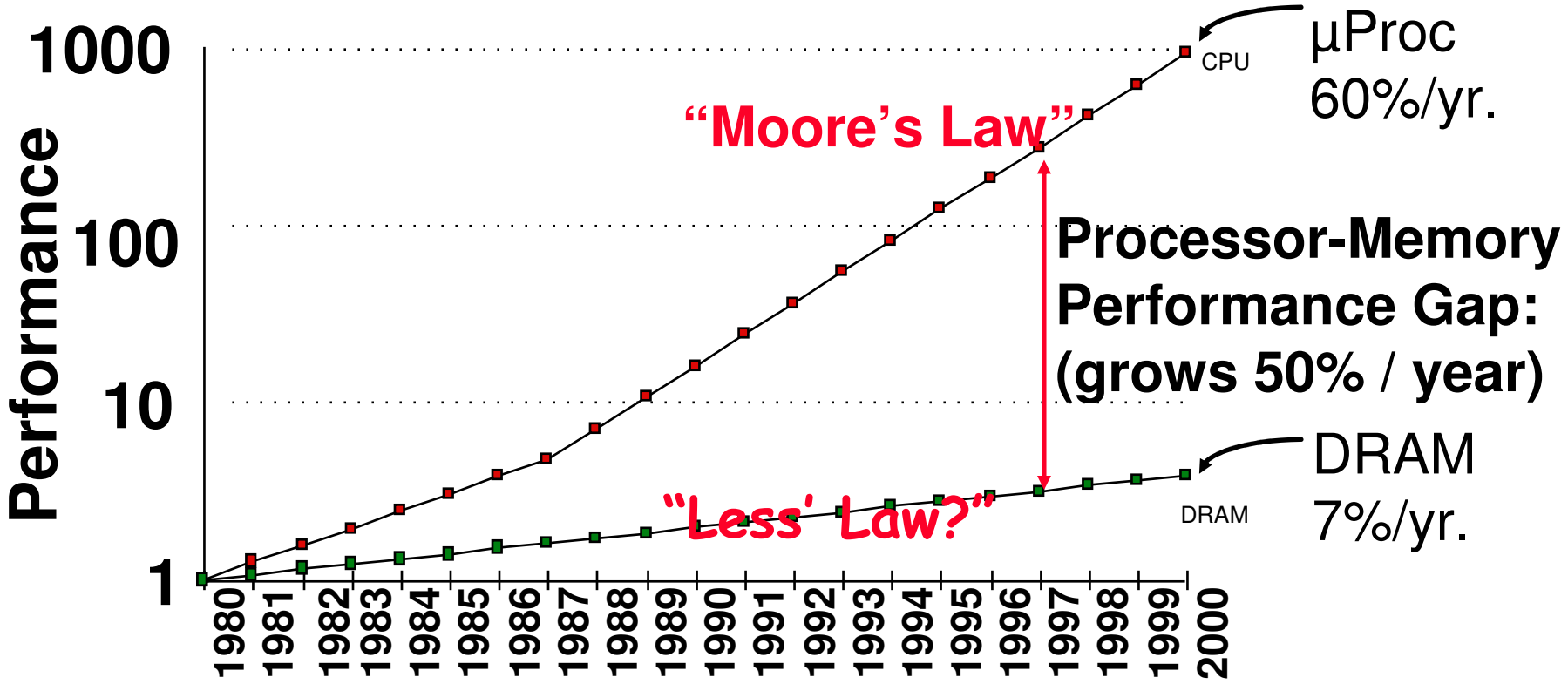
"Computer Architecture: A Quantitative Approach" - (Capítulo 5)

Projeto de Hierarquia de Memória

- Introdução
- Desempenho de Cache
- Reduzindo Misses
  - Classificação dos Misses
- Redução do Cache Miss Penalty
- Redução do Miss Rate
- Redução de Cache Miss Penalty e Miss Rate usando Paralelismo
- Redução do Hit Time

Introdução

CPU-DRAM Gap



- 1980: não existia cache nos µproc; 1995 2-level cache no chip
- 1989 primeiro µproc Intel com cache no chip

Introdução: Desempenho da Cache

- Orientado ao **Miss** no acesso à Memória:

$$CPUtime = IC \times \left(CPI_{Execution} + \frac{MemAccess}{Inst} \times MissRate \times MissPenalty \right) \times CycleTime$$

$$CPUtime = IC \times \left(CPI_{Execution} + \frac{MemMisses}{Inst} \times MissPenalty \right) \times CycleTime$$

  - $CPI_{Execution}$ inclui instruções da ALU e de acesso à Memória

- Isolando o acesso à Memória
  - AMAT = Average Memory Access Time
  - CPI_{ALUOps} não inclui as instruções de memória

$$CPUtime = IC \times \left(\frac{AluOps}{Inst} \times CPI_{AluOps} + \frac{MemAccess}{Inst} \times AMAT \right) \times CycleTime$$

$$AMAT = HitTime + MissRate \times MissPenalty$$
$$= (HitTime_{Inst} + MissRate_{Inst} \times MissPenalty_{Inst}) +$$
$$(HitTime_{Data} + MissRate_{Data} \times MissPenalty_{Data})$$

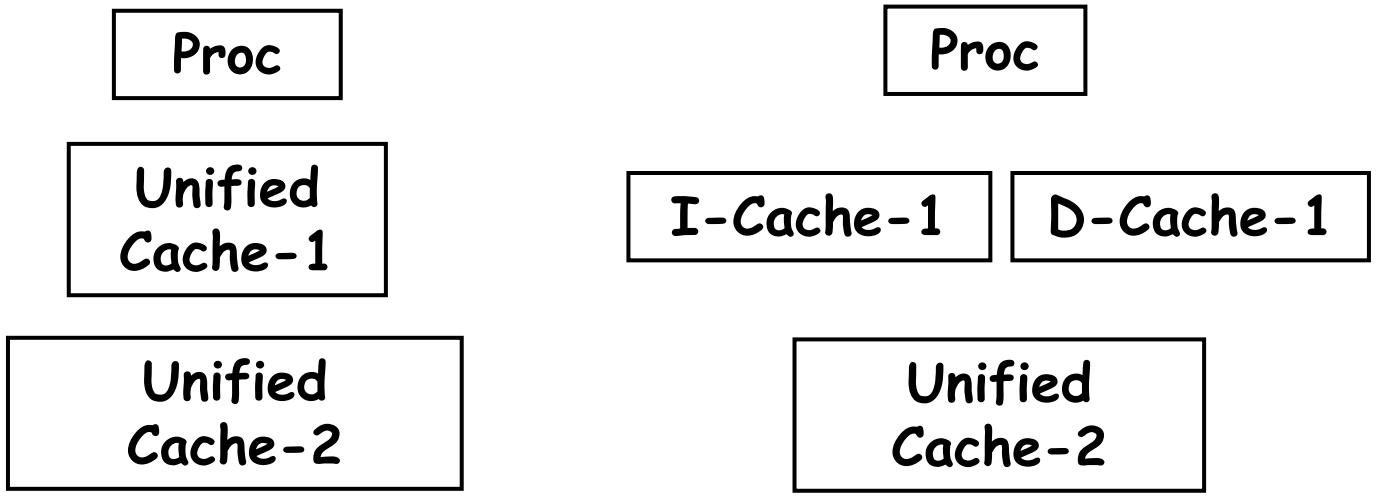
Impacto no Desempenho

- Suponha um processador executando:
  - Clock Rate = 200 MHz (5 ns por ciclo), CPI Ideal (sem misses) = 1.1
  - 50% aritmética/lógica, 30% ld/st, 20% controle
- Suponha que 10% das operações de memória gastam 50 ciclos de miss penalty
- Suponha que 1% das instruções tenham o mesmo miss penalty
- CPI = ideal CPI + average stalls per instruction

  1.1(cycles/ins) +
  [0.30 (DataMops/ins)
  x 0.10 (miss/DataMop) x 50 (cycle/miss)] +
  [1 (InstMop/ins)
  x 0.01 (miss/InstMop) x 50 (cycle/miss)]
  = (1.1 + 1.5 + 0.5) cycle/ins = 3.1
- 64% do tempo do processador é devido a stall esperando pela memória!, 48% devido a espera por dados!
- AMAT=(1/1.3)x[1+0.01x50]+(0.3/1.3)x[1+0.1x50]=2.54

Exemplo: Arquitetura Harvard

- Cache Unificada vs Separada I&D (Harvard)



Exemplo: Arquitetura Harvard

- Suponha:
  - 16KB I&D: Inst miss rate = 0.64%, Data miss rate = 6.47%
  - 32KB unificada: miss rate = 1.99% (agregado)
  - 33% das instruções são ops de dados (load ou store)

- Qual é melhor (ignore cache L2)?
  - 33% ops de dados ⇒ 75% dos acessos são devidos a fetch das instruções (1.0/1.33)
  - hit time = 1, miss time = 50
  - Note que *data* hit tem 1 stall para a cache unificada (somente uma porta)

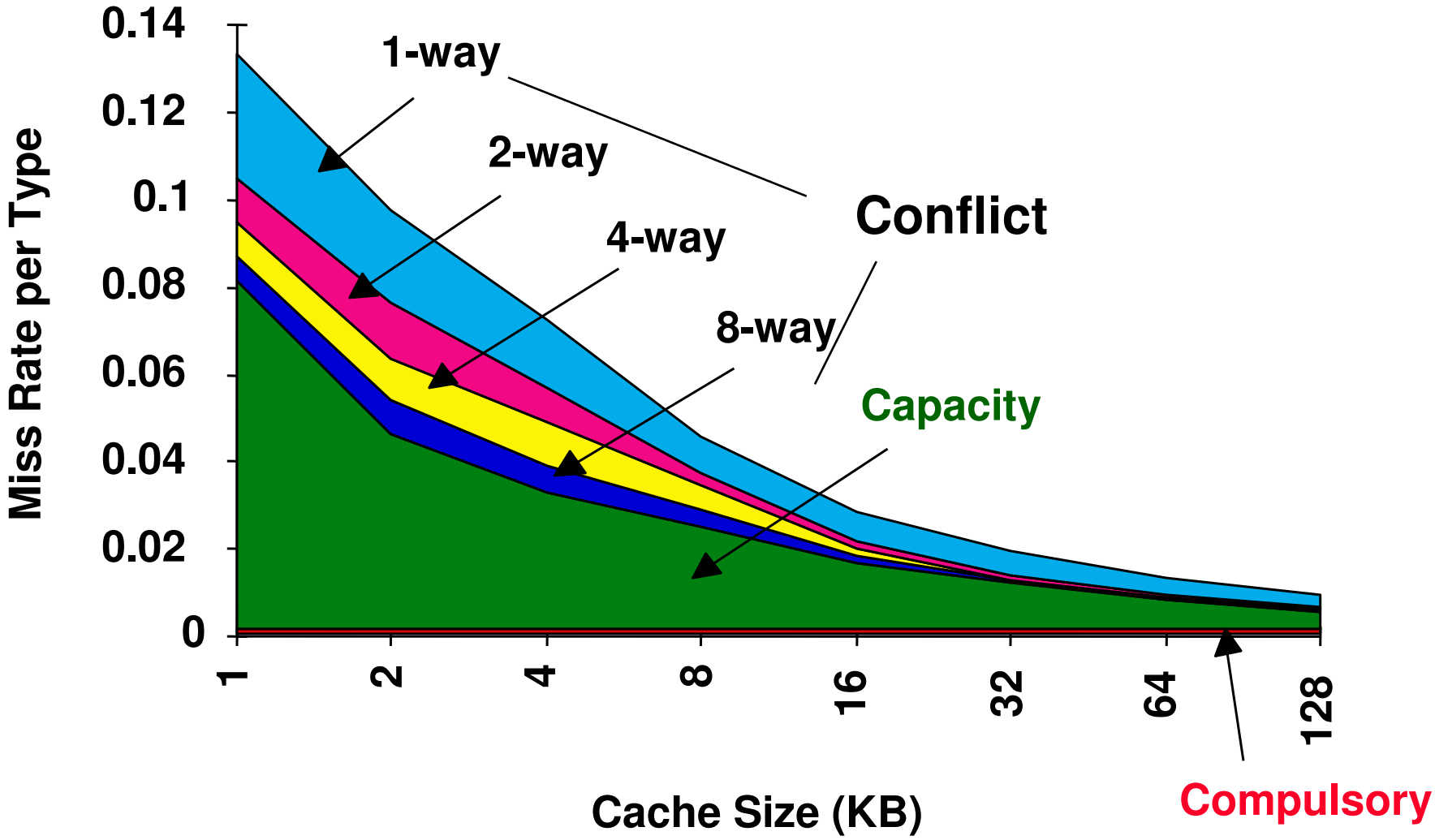
$AMAT_{Harvard}$ = 75%x(1+0.64%x50) + 25%x(1+6.47%x50) = 2.05

$AMAT_{Unified}$ = 75%x(1+1.99%x50) + 25%x(1+1+1.99%x50)= 2.24

Introdução: Reduzindo Misses

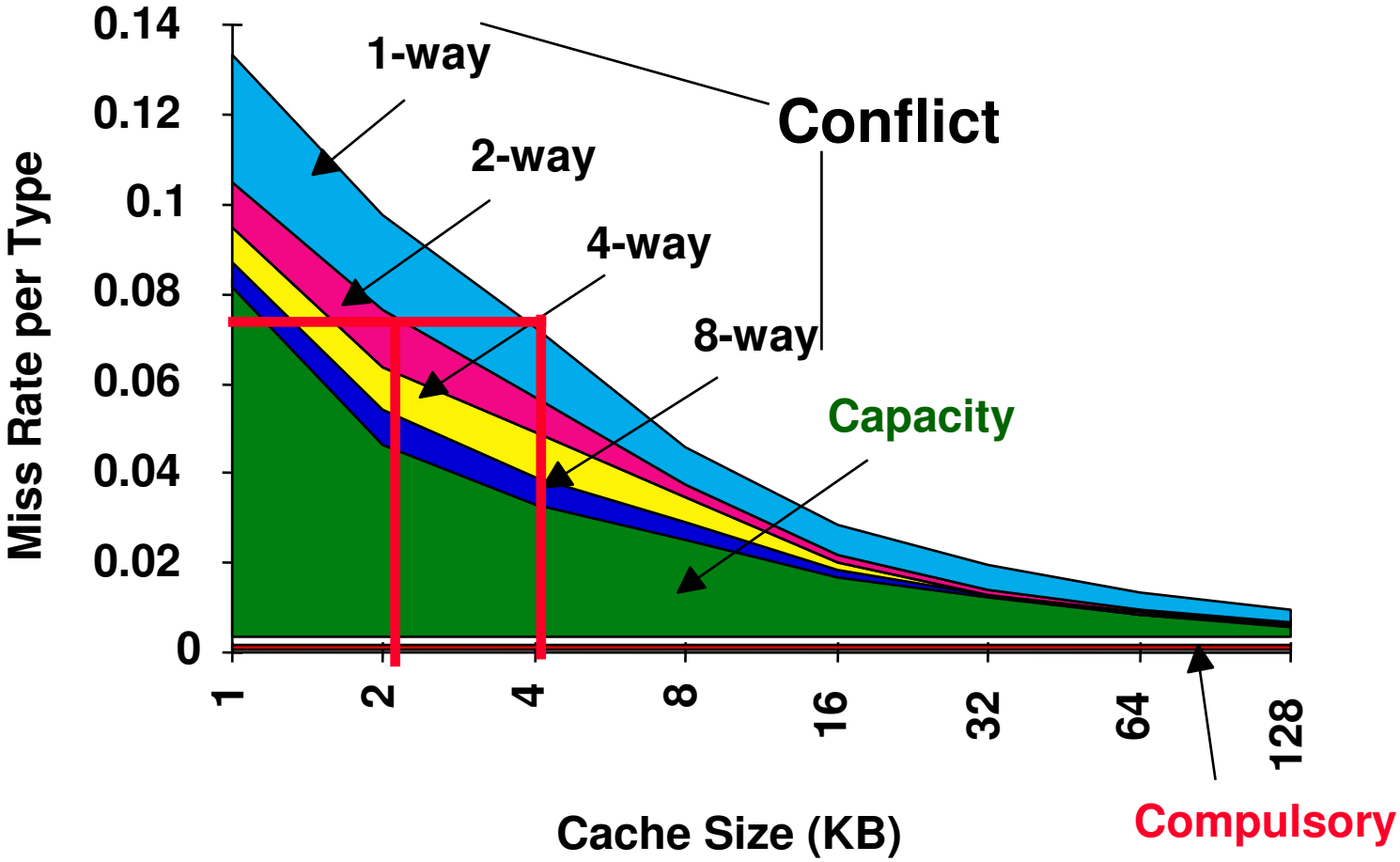
- Classificação dos Misses: 3 Cs
  - Compulsory — Misses em uma Cache "Infinita"
  - Capacity — Misses em Fully Associative
  - Conflict — Misses em N-way Associative
- Mais recentemente, 4º "C":
  - *Coherence* — Misses causados pela coerência de Cache.

3Cs - Miss Rate Absoluto (SPEC92)



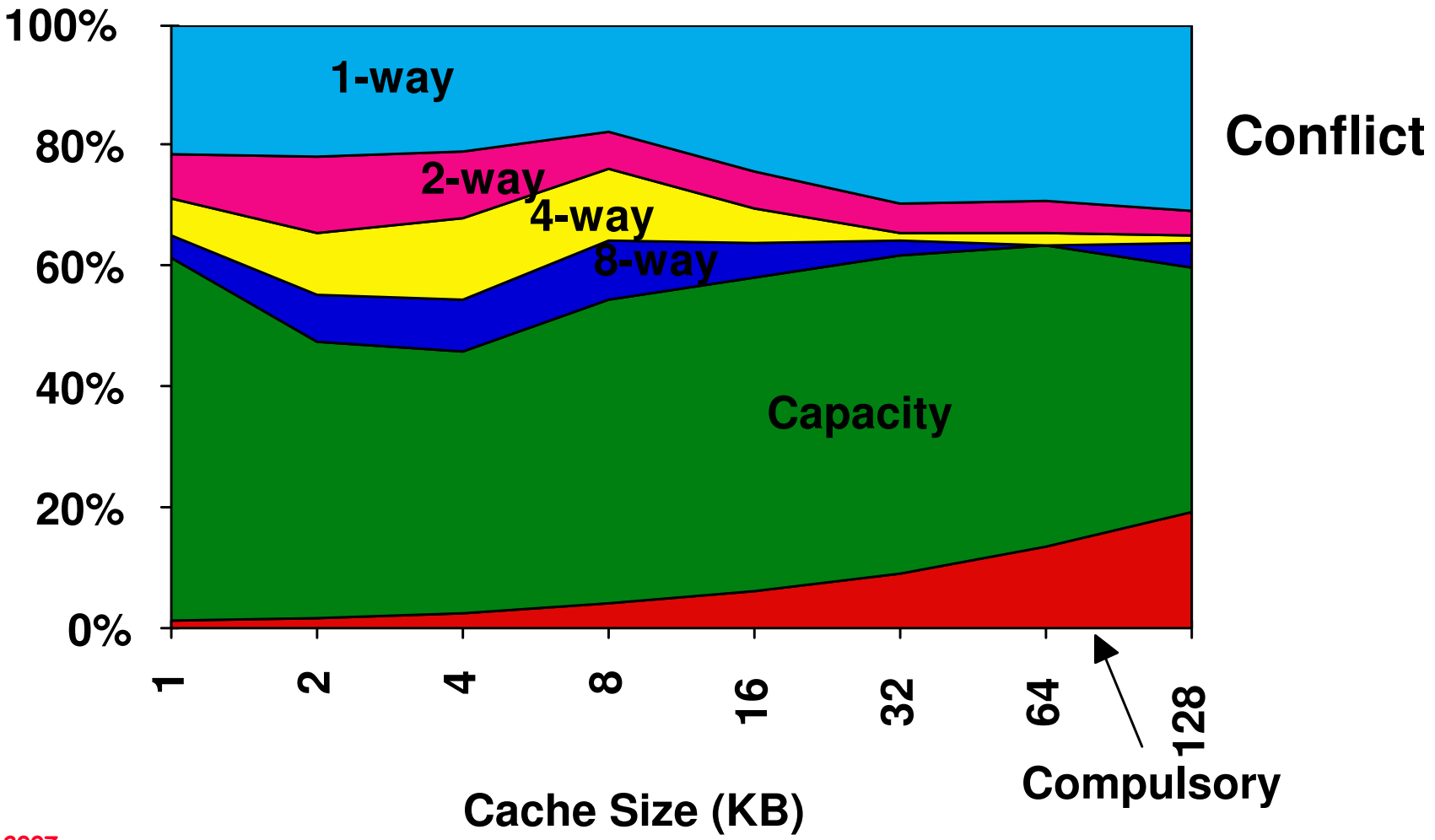
Cache Misses

miss rate 1-way associative cache size X
= miss rate 2-way associative cache size X/2



3Cs Miss Rate Relativo

Flaws: for fixed block size
Good: insight => invention



Como Reduzir os Misses?

- 3 Cs: Compulsório, Capacidade e Conflito
- Assuma para todos os casos que o tamanho (em bytes) da cache não é alterado:
- O que ocorre se:

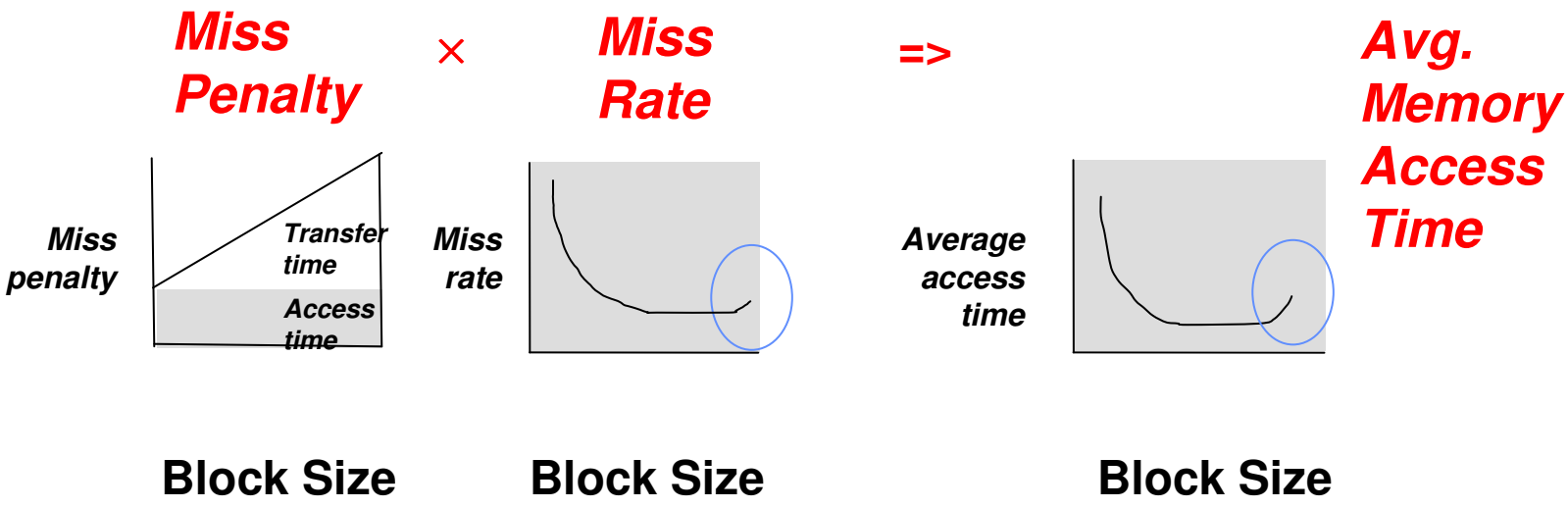
1) O Block Size for alterado:
   Quais dos 3Cs são afetados?

2) A Associatividade for alterada:
   Quais dos 3Cs são afetados?

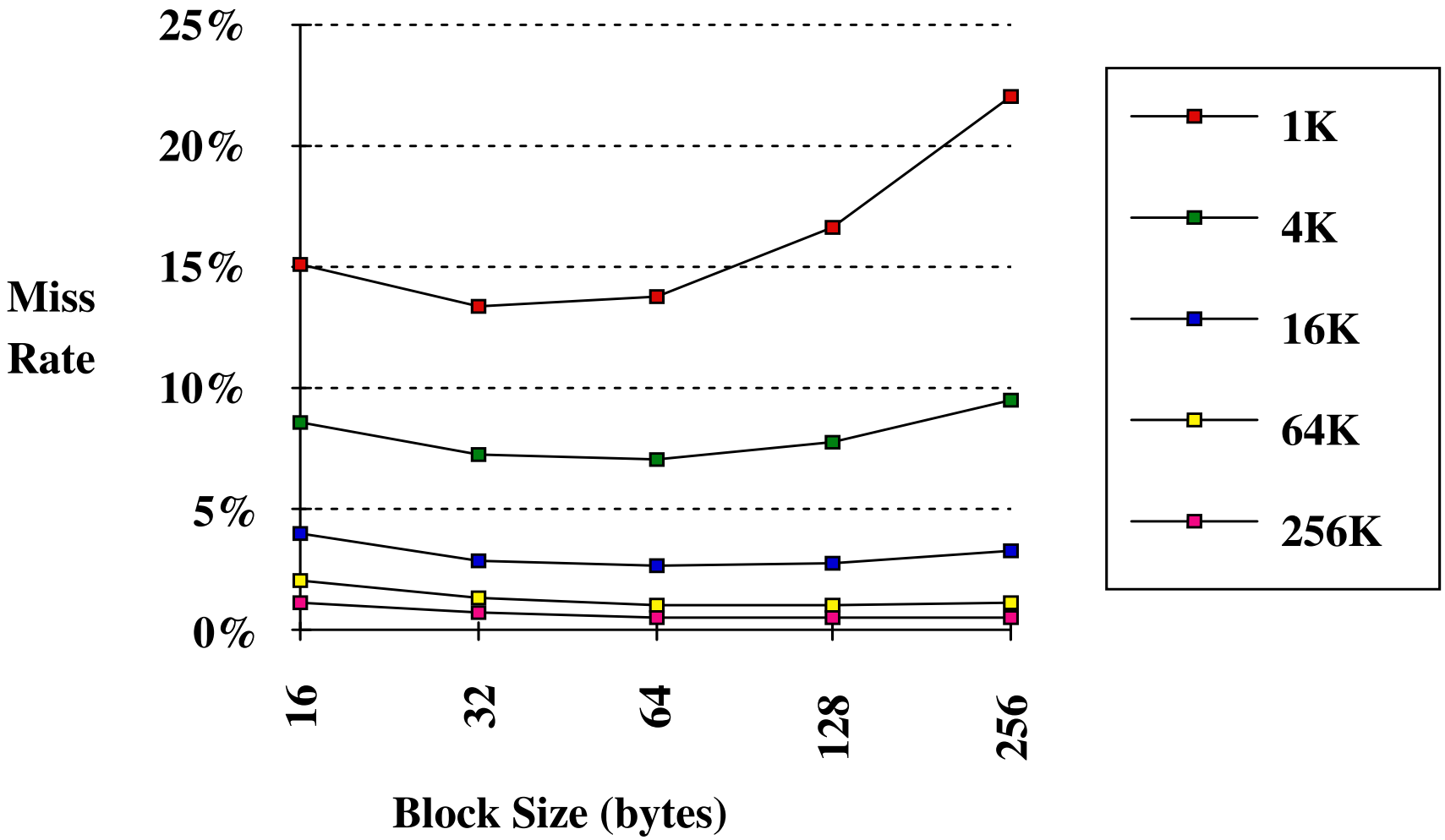
3) O Compilador for alterado:
   Quais dos 3Cs são afetados?

Miss Penalty e Miss Rate vs Tamanho do Bloco

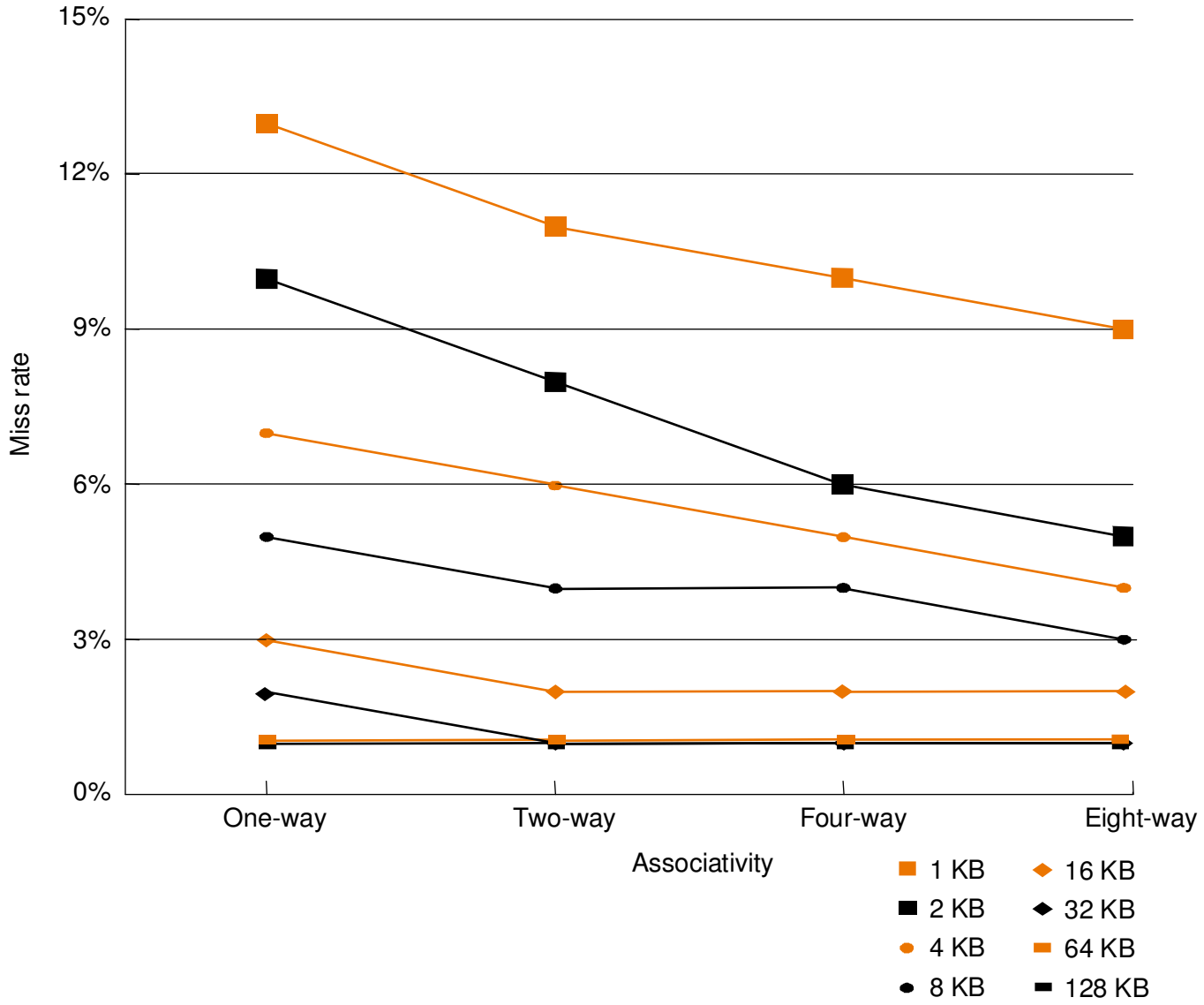
- Aumentando o tamanho do bloco aumenta-se o miss penalty



Miss Rate vs Tamanho do Bloco



Miss Rate vs Associatividade



Melhorando o Desempenho da Cache

1. Reduzindo o miss rate,
2. Reduzindo o miss penalty, ou
3. *Reduzindo o tempo de hit na cache.*

$$AMAT = HitTime + MissRate \times MissPenalty$$

Redução do Miss Rate

$$AMAT = HitTime + MissRate \times MissPenalty$$

Average Memory Access Time

- 3 Cs: Compulsório, Capacidade, Conflito

1. Redução dos Misses Aumentando o Block Size v
2. Redução dos Misses Aumentando a Associatividade v
3. Redução dos Misses via Victim Cache
4. Redução dos Misses via Pseudo-Associatividade
5. Redução dos Misses por HW Prefetching Instr e/ou Dado
6. Redução dos Misses por SW Prefetching Dado
7. Redução dos Misses por Otimizações do Compilador

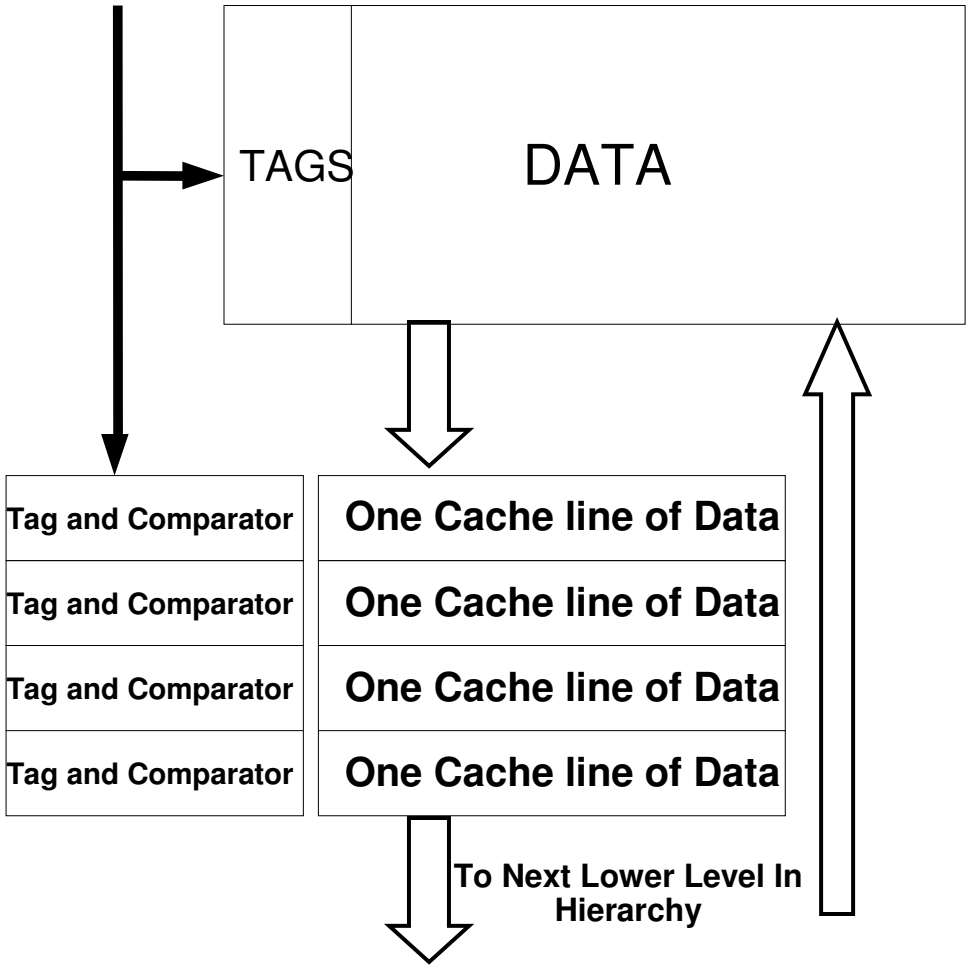
Victim Caches

- Como combinar o melhor hit time da direct mapped evitando conflict misses?

- Adicionar um buffer para guardar o dado descartado da cache
- Uma victim cache de 4-entradas remove 20% a 95% dos conflitos para uma direct mapped data cache de 4 KB
- Usado nas máquinas Alpha e HP.

- Como efeito, obtém-se o mesmo comportamento da associatividade, porém somente para as cache lines que realmente precisam disso.

Victim Caches



Pseudo-Associatividade

- Como combinar o melhor hit time da direct mapped evitando conflict misses como na 2-way SA cache?
- Dividir a cache: em um miss, checar a outra metade da cache, se encontrou então tem-se um pseudo-hit (hit mais lento)

- Desvantagem: pepilene da CPU mais complicado se o hit pode levar 1 ou 2 ciclos
  - Melhor para caches que não estão diretamente ligadas ao processador (L2)
  - Usada na cache L2 do MIPS R1000, similar no UltraSPARC

Hardware Prefetching de Instruções e Dados

- Prefetching de Instruções
  - Alpha 21064 fetches 2 blocos nos miss
  - Bloco Extra colocado no "stream buffer"
  - No miss, verifica o stream buffer

- Também funciona com blocos de dados:
  - Jouppi [1990]: 1 data stream buffer captura até 25% dos misses para uma cache de 4KB; e 4 streams captura 43%
  - Palacharla & Kessler [1994]: para programas científicos uma 8 streams captura de 50% a 70% dos misses para duas (I&D) 64KB, 4-way set associative caches

Software Prefetching de Dados

- **Data Prefetch**
  - Carregar o dado no registrador (HP PA-RISC loads)
  - Cache Prefetch: Carregar na cache (MIPS IV, PowerPC, SPARC v. 9)
  - Instruções especiais para prefetching, não podem causar exceções; uma forma de execução especulativa

- **Duas abordagens para o Prefetching:**
  - Binding prefetch: O load é realizado diretamente no registrador.
    - » O endereço e o registrador devem ser os corretos
  - Non-Binding prefetch: O Load é realizado na cache.
    - » Pode ser incorreto. O HW/SW podem adivinhar!

Reduzindo Misses com o Compilador

- McFarling [1989] – redução de 75% dos caches misses em uma 8KB direct mapped cache, 4 byte blocks em software
- Instruções
  - Reorganização dos procedimentos (funções) na memória para reduzir os conflict misses
  - Profiling verificando os conflitos
- Dados
  - *Merging Arrays*: melhor localidade espacial apresentada pelos arrays simples em relação aos arrays de 2 dimensões
  - *Loop Interchange*: alteração dos nesting loops para realizar os acessos aos dados na ordem em que estão armazenados na memória
  - *Loop Fusion*: Combinação de 2 loops independentes que possuem compartilhamento de variáveis
  - *Blocking*: Melhoria da localidade temporal acessando "blocos" de dados vs. acesso a toda uma coluna ou linha

Exemplo de Merging Arrays

```
/* Before: 2 sequential arrays */
int val[SIZE];
int key[SIZE];
```

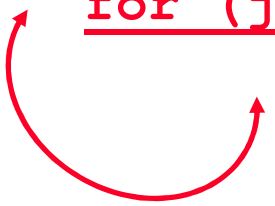
```
/* After: 1 array of stuctures */
struct merge {
  int val;
  int key;
};
struct merge merged_array[SIZE];
```

Redução de conflitos entre val & key; melhora a localidade espacial

Exemplo Loop Interchange

```
/* Before */
for (k = 0; k < 100; k = k+1)
  for (j = 0; j < 100; j = j+1)
      for (i = 0; i < 5000; i = i+1)
            x[i][j] = 2 * x[i][j];
```

```
/* After */
for (k = 0; k < 100; k = k+1)
  for (i = 0; i < 5000; i = i+1)
      for (j = 0; j < 100; j = j+1)
            x[i][j] = 2 * x[i][j];
```



Acesso seqüencial, por linha; melhora a localidade espacial (em geral arrays são organizados por linha pelos compiladores

Exemplo Loop Fusion

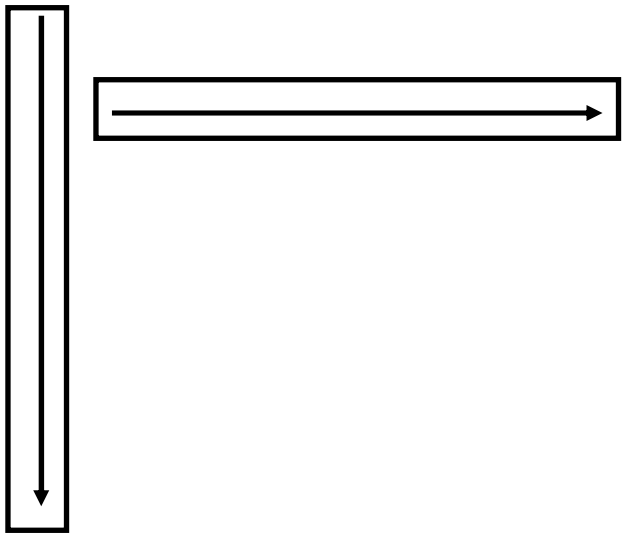
```
/* Before */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
      a[i][j] = 1/b[i][j] * c[i][j];
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
      d[i][j] = a[i][j] + c[i][j];

/* After */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
  {   a[i][j] = 1/b[i][j] * c[i][j];
      d[i][j] = a[i][j] + c[i][j];}
```

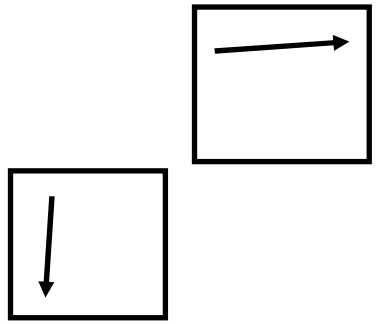
2 misses por acessos a a & c vs. 1 miss por acesso; melhora a localidade espacial

Exemplo de Blocking

```
/* Before */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
     {r = 0;
      for (k = 0; k < N; k = k+1){
        r = r + y[i][k]*z[k][j];};
      x[i][j] = r;};
```



- Dois Inner Loops:
  - Lê todos os NxN elementos de z[]
  - Lê N elementos de 1 linha de y[] repetidas vezes
  - Escreve N elementos de 1 linha de x[]
- Capacity Misses como função de N & Cache Size:
  - $2N^3 + N^2$ => (assumindo sem conflito; ...)
- Idéia: computar a sub-matriz BxB

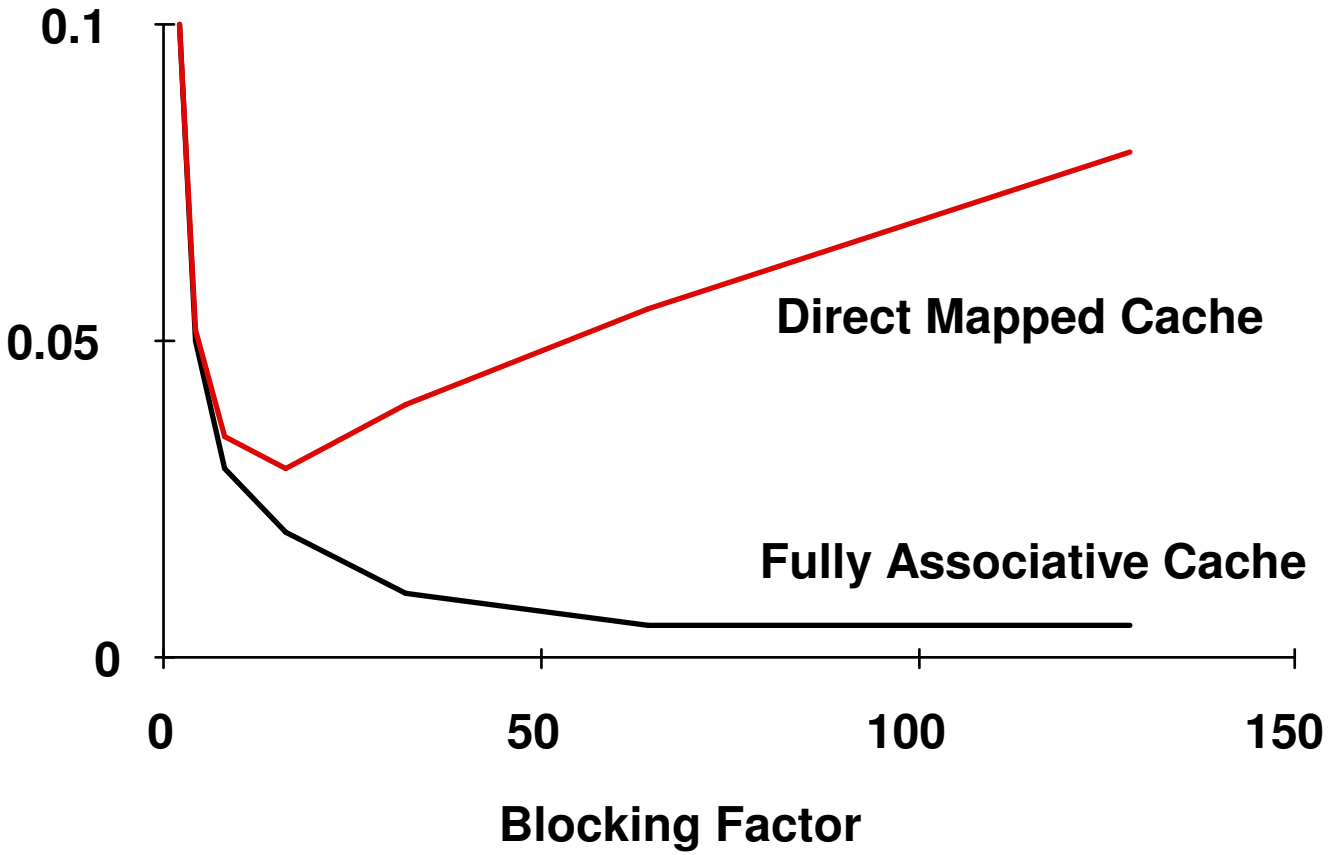


Exemplo de Blocking

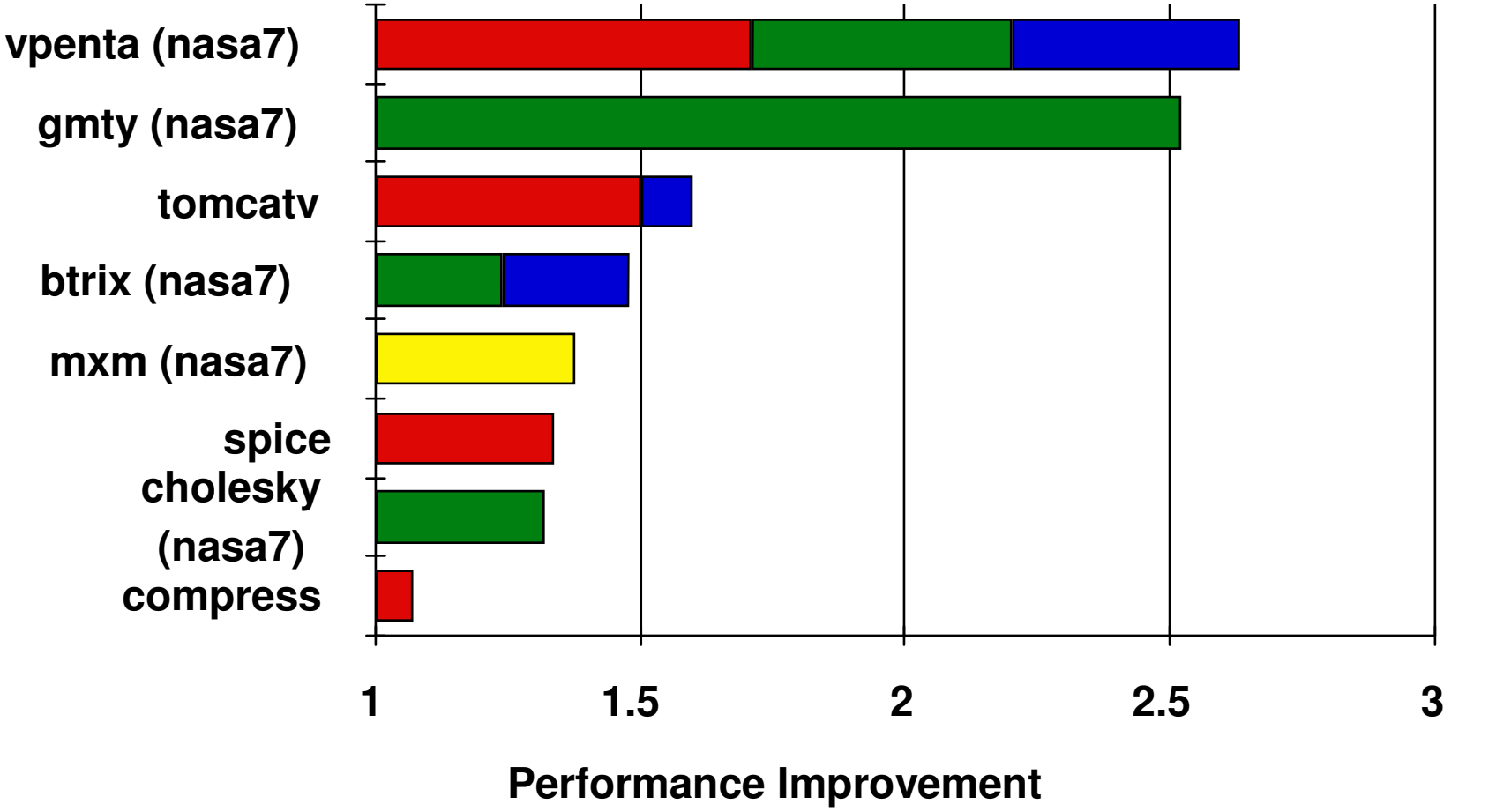
```
/* After */
for (jj = 0; jj < N; jj = jj+B)
for (kk = 0; kk < N; kk = kk+B)
for (i = 0; i < N; i = i+1)
   for (j = jj; j < min(jj+B-1,N); j = j+1)
     {r = 0;
      for (k = kk; k < min(kk+B-1,N); k = k+1) {
        r = r + y[i][k]*z[k][j];};
      x[i][j] = x[i][j] + r;
     };
```

- B é chamado de *Blocking Factor*
- Capacity Misses de $2N^3 + N^2$ para $N^3/B+2N^2$
- Conflict Misses?

Reduzindo Conflict Misses por Blocking



Sumário Sobre Redução de Misses com Uso de Compiladores



Melhorando o Desempenho da Cache

1. Reduzindo o miss rate,
2. *Reduzindo o miss penalty,* ou
3. Reduzindo o tempo de hit da cache.

$$AMAT = HitTime + MissRate \times MissPenalty$$

Average Memory Access Time

O Que Ocorre em um Cache miss?

- Para um pipeline in-order, 2 opções:
  - Congelar o pipeline no estágio Mem (Sparc, R4000)

```
IF   ID   EX   Mem stall stall … stall Mem   Wr
     IF   ID   EX  stall stall stall … stall Ex  Mem Wr
```

O Que Ocorre em um Cache miss?

- Usar Full/Empty bits nos registradores + MSHR queue
  - » MSHR = "Miss Status/Handler Registers" (Kroft)
    Cada entrada na fila mantém informações sobre as requisições de uma linha completa de memória.
    - Para cada cache-line: mantém informações dos endereços de memória.
    - Para cada word: o registrador (se existir) que está esperando o resultado.
    - Usado para fazer o "merge" de múltiplas requisições a uma linha de memória
  - » Um novo load cria uma entrada MSHR e seta o registrador destino para "Empty". O Load é liberado do pipeline.
  - » O uso do registrador antes do retorno do resultado causa um bloqueio da instrução no estágio decode.
  - » Execução "out-of-order" limitada com respeito a loads.
    Popular com arquiteturas superscalar in-order.
- Pipelines out-of-order já possuem essa funcionalidade built in... (load queues etc).

Políticas de Write:
Write-Through vs Write-Back

- **Write-Through: os writes atualizam a cache e as memórias/caches subordinadas**
  - Sempre pode-se descartar dados da cache – a maioria dos dados atualizados estão na memória
  - Cache control bit: somente *valid* bit

- **Write-Back: os writes só atualizam a cache**
  - Não se pode descartar dados da cache – ele deve ser escrito de volta na memória
  - Cache control bits: *valid* e *dirty* bits

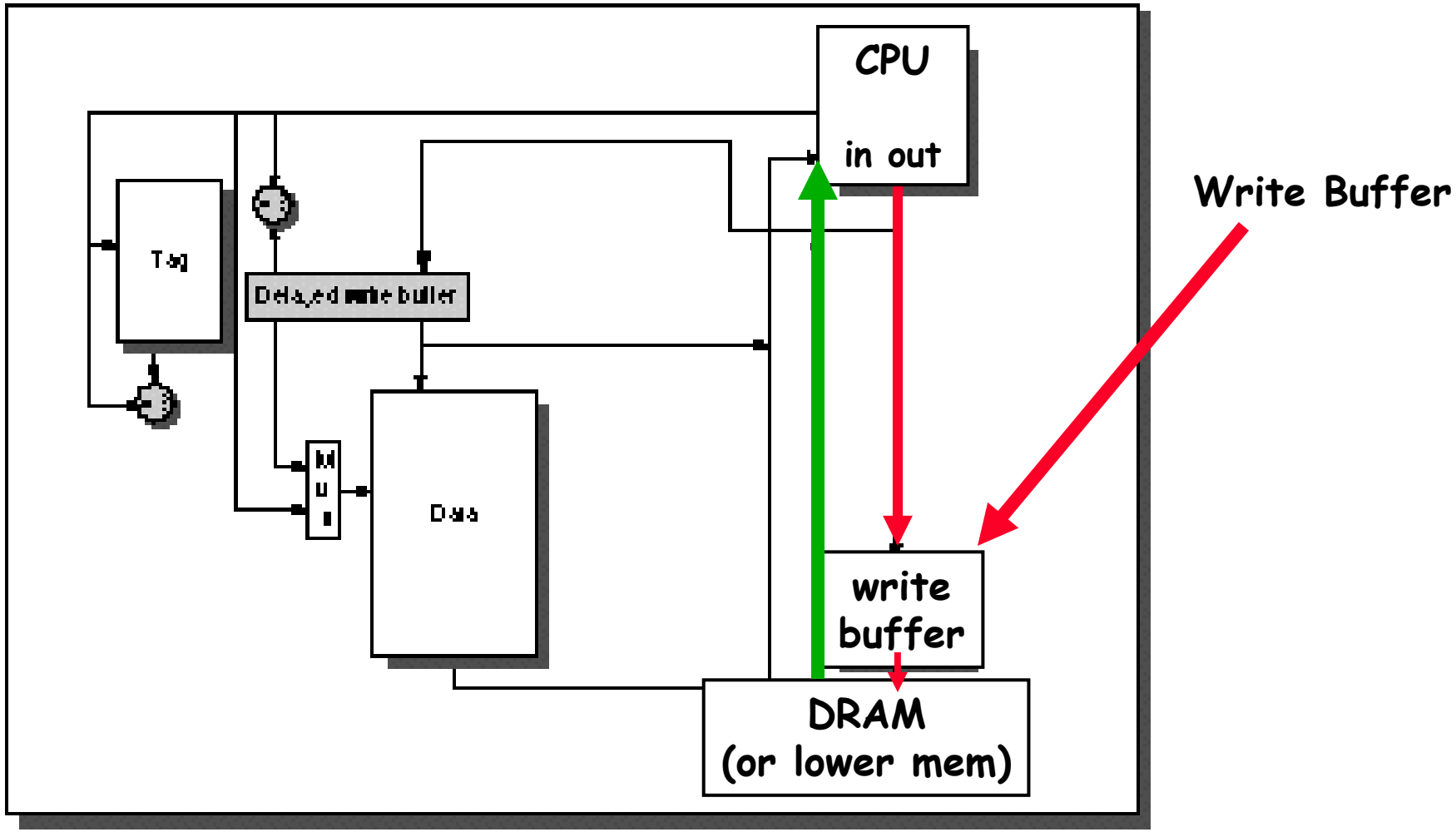
Políticas de Write:
Write-Through vs Write-Back

- Vantagens:
  - Write-Through:
    - » A memória (ou outros processadores) sempre tem o dado atualizado
    - » Gerenciamento da cache mais simples
  - Write-Back:
    - » menor bandwidth (os dados não são reescritos várias vezes)
    - » Melhor tolerância à memórias com grandes latências

Políticas de Write:
Write Allocate vs Non-Allocate
(O que ocorre em um write-miss)

- Write Allocate: aloca uma nova cache line na cache
  - Usualmente significa que deve haver um "read miss" para preencher o resto da cache-line
  - Alternativa: valid bits por word
- Write Non-Allocate (ou "write-around"):
  - Simplesmente envia o dado para a memória/cache subordinada – não aloca uma nova cache line.

Reduzindo Miss Penalty:
Prioridade do Read Sobre o Write em Miss



Reduzindo Miss Penalty:
Prioridade do Read sobre o Write em Miss

- Write-Through com write buffers possui conflito do tipo RAW em leituras da memória em cache misses
  - Se se espera até o write buffer estar vazio, pode-se aumentar o read miss penalty
  - Checar o conteúdo do write buffer antes da leitura; se não há conflito deixa-se que o acesso à memória continue

- Write-Back também necessita de buffer para manter misplaced blocks
  - Read miss → recolocação em um dirty block
  - Normal: Escreve o dirty block na memória, e então faz o Read
  - Solução: copia o dirty block para o write buffer, então executa o Read e depois o Write
  - Menos CPU stall já que ela é reiniciada assim que o Read é executado

Reduzindo Miss Penalty:
Reinicio Antecipado e Word Crítica Primeiro

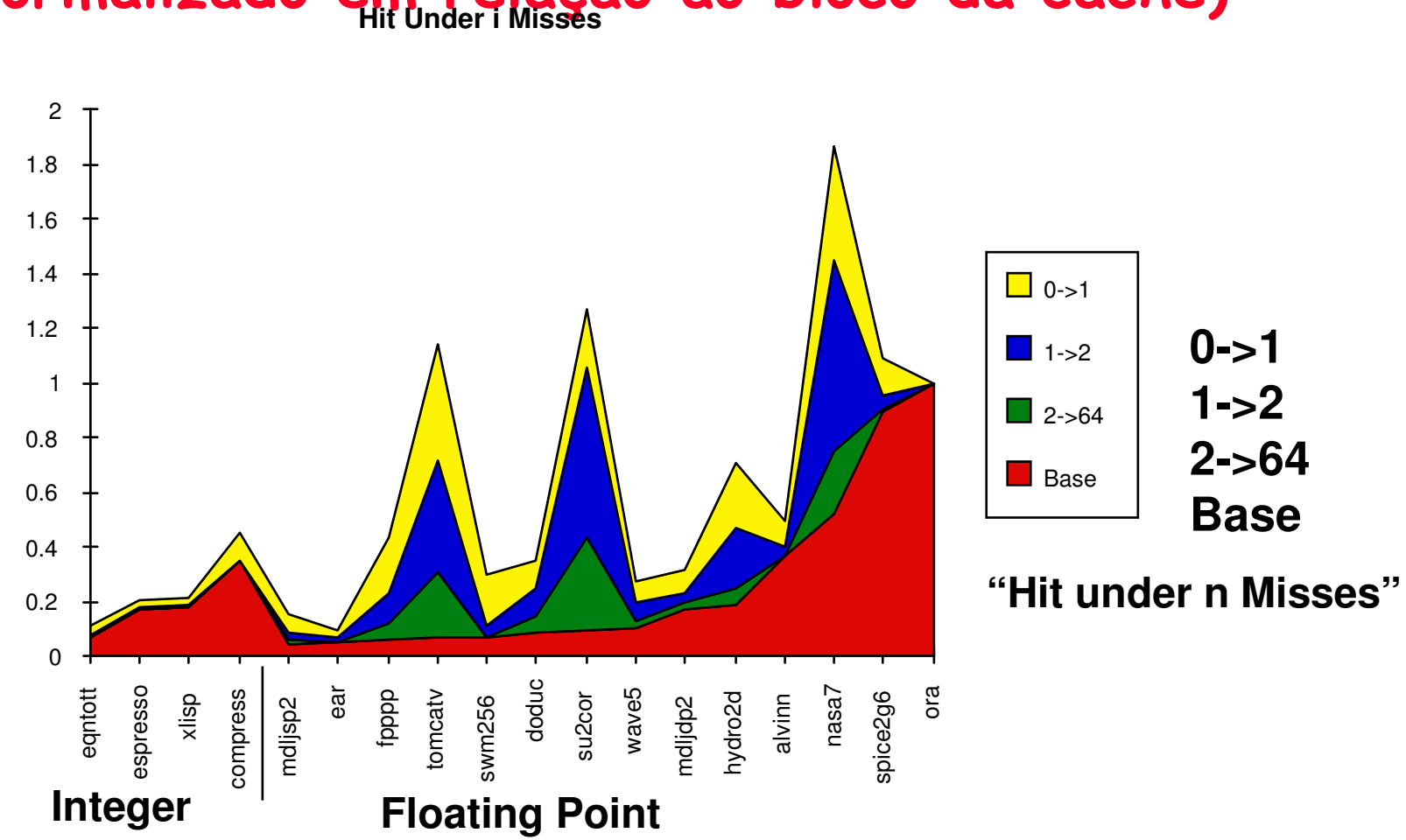
- Não esperar até todo o bloco estar carregado para "reiniciar" a CPU
  - *Reinicio Antecipado* — Assim que a palavra requisitada do bloco chegar da memória entrega-la à CPU e deixar a CPU continuar a execução
  - *Word Crítica Primeiro* — Requisitar a missed word primeiro da memória e envia-la à CPU assim que chegar; deixar a CPU continuar a execução enquanto preenche o resto das palavras no bloco. Também chamado de *wrapped fetch* e *requested word first*
- Geralmente útil somente com blocos grandes,
- Localidade espacial representa um problema; tende querer a próxima palavra seqüencialmente não ficando claro se há benefícios com o reinicio antecipado

block

Reduzindo Miss Penalty:
Non-Blocking Caches para Reduzir Stalls em Misses

- *Non-blocking cache* ou *lockup-free cache* permite a cache de dados continuar fornecendo cache hits durante um miss
  - requer F/E bits nos registradores ou execução out-of-order
  - Requer memórias multi-bank
- "*hit under miss*" reduz o miss penalty efetivo trabalhando durante um miss vs. ignorando requisições da CPU
- "*hit under multiple miss*" ou "*miss under miss*" pode reduzir o miss penalty efetivo sobrepondo múltiplos misses
  - Aumenta significativamente a complexidade do controlador da cache (múltiplos acessos à memória)
  - Requer *multiple memory banks*
  - Pentium Pro permite 4 memory misses

Valores de Hit Sob Miss para o SPEC (Normalizado em relação ao bloco da cache)



- FP programs on average: AMAT= 0.68 -> 0.52 -> 0.34 -> 0.26
- Int programs on average: AMAT= 0.24 -> 0.20 -> 0.19 -> 0.19
- 8 KB Data Cache, Direct Mapped, 32B block, 16 cycle miss

Segundo Nível de Cache

- Equações para L2

$$AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times Miss\ Penalty_{L1}$$

$$Miss\ Penalty_{L1} = Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L2}$$

$$AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times (Hit\ Time_{L2} + Miss\ Rate_{L2} + Miss\ Penalty_{L2})$$

- Definições:
  - *Local miss rate* — misses na cache dividido pelo número total de acessos à memória *para a cache* (Miss $rate_{L1}$; Miss $rate_{L2}$)
  - *Global miss rate* — misses na cache dividido pelo número total de acessos à memória *gerados pela CPU* (Miss $Rate_{L1}$; Miss $Rate_{L1}$ x Miss $Rate_{L2}$)
  - Global Miss Rate é o que importa

Segundo Nível de Cache

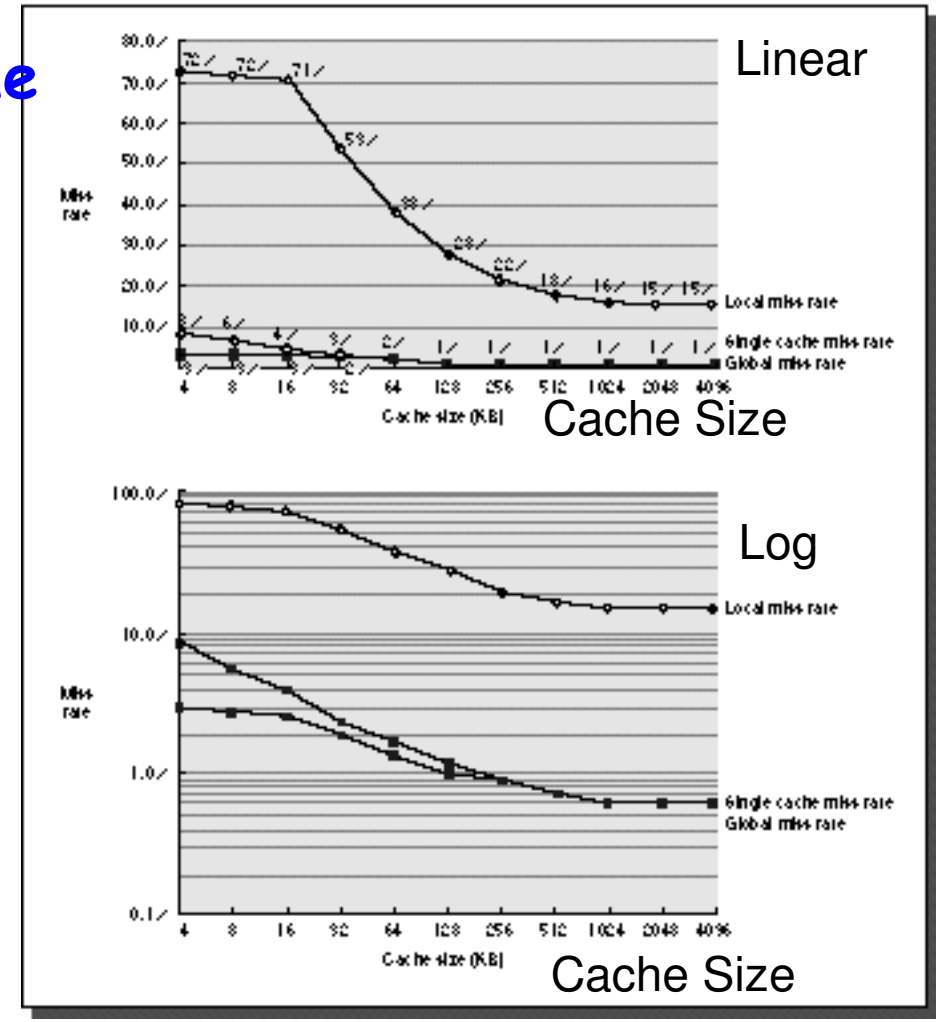
- Misses por Instrução - Para evitar confusão com miss rate global ou local usa-se o conceito de memory stalls por instrução, incluindo o impacto do segundo nível de cache:

$$AMSI = \text{Misses per instruction}_{L1} \times \text{Hit Time}_{L2} + \text{Misses per instruction}_{L2} \times \text{Miss Penalty}_{L2}$$

Average Memory Stalls per Instruction

Comparando Local e Global Miss Rates

- 32 KByte 1º level cache; Aumentando o 2º level cache
- Global miss rate próximo a single level cache rate
- Não use local miss rate

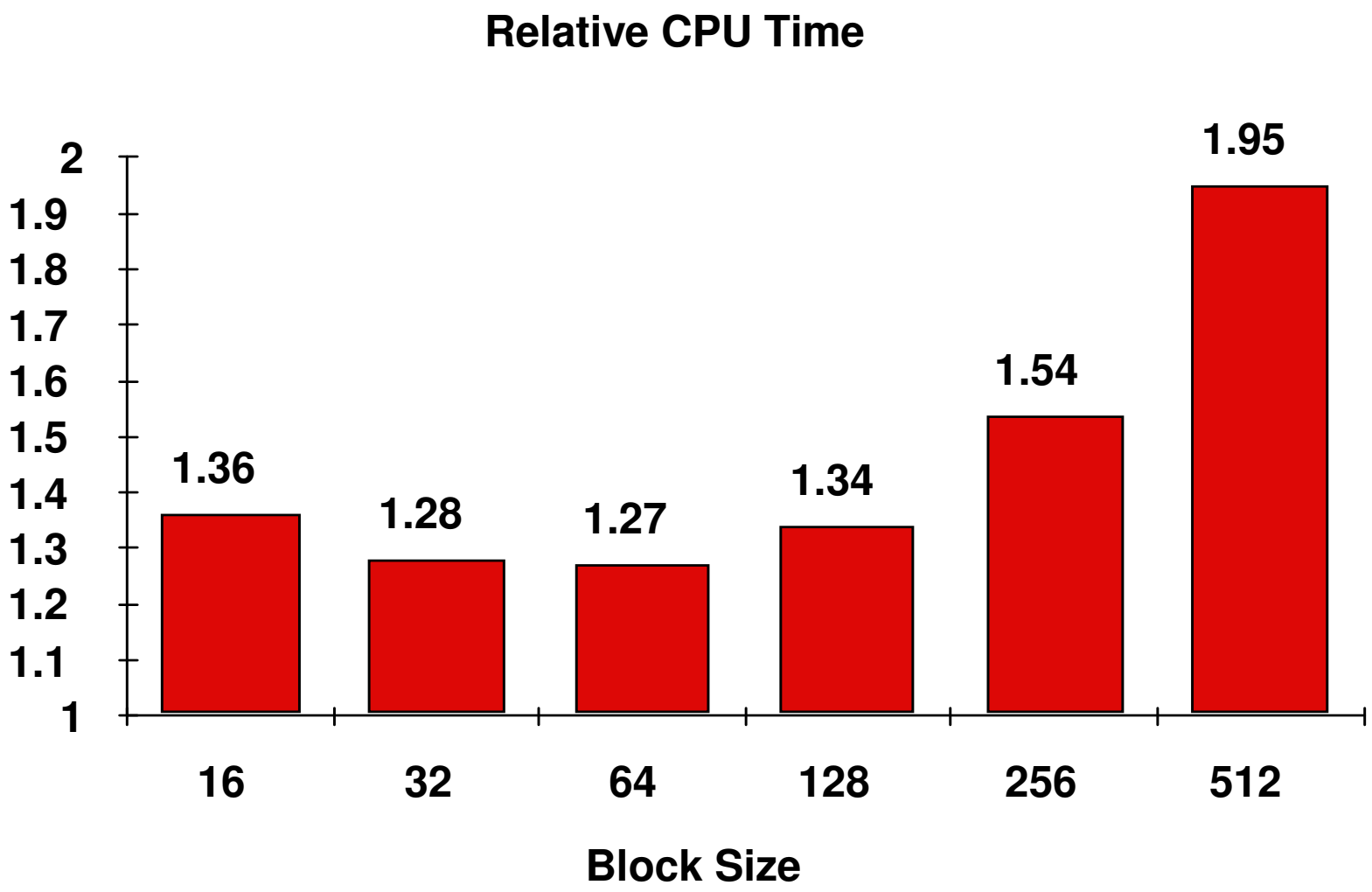


Reduzindo Misses:
O que pode ser aplicado à L2 Cache?

- Reduzindo Miss Rate

1. Redução dos Misses Aumentando o Block Size
2. Redução dos Conflict Misses Aumentando a Associatividade
3. Redução dos Conflict Misses via Victim Cache
4. Redução dos Conflict Misses via Pseudo-Associatividade
5. Redução dos Misses por HW Prefetching Instr e/ou Dado
6. Redução dos Misses por SW Prefetching Dado
7. Redução de Capacity/Confl. Misses por Otimizações do Compilador

L2 Cache Block Size & A.M.A.T.



Resumo: Reduzindo Miss Penalty

$$AMAT = HitTime + MissRate \times MissPenalty$$

- Quatro técnicas
  - Prioridade para Read sobre Write durante um miss
  - Reinicio Antecipado e Word Crítica Primeiro em miss
  - Non-blocking Caches (Hit sob Miss, Miss sob Miss)
  - Segundo Nível de Cache
- Pode ser aplicado recursivamente para Multilevel Caches
  - O perigo é que o tempo para DRAM irá crescer para múltiplos níveis
  - L2 caches pode tornar as coisas piores já que piora o pior caso

Memória Principal

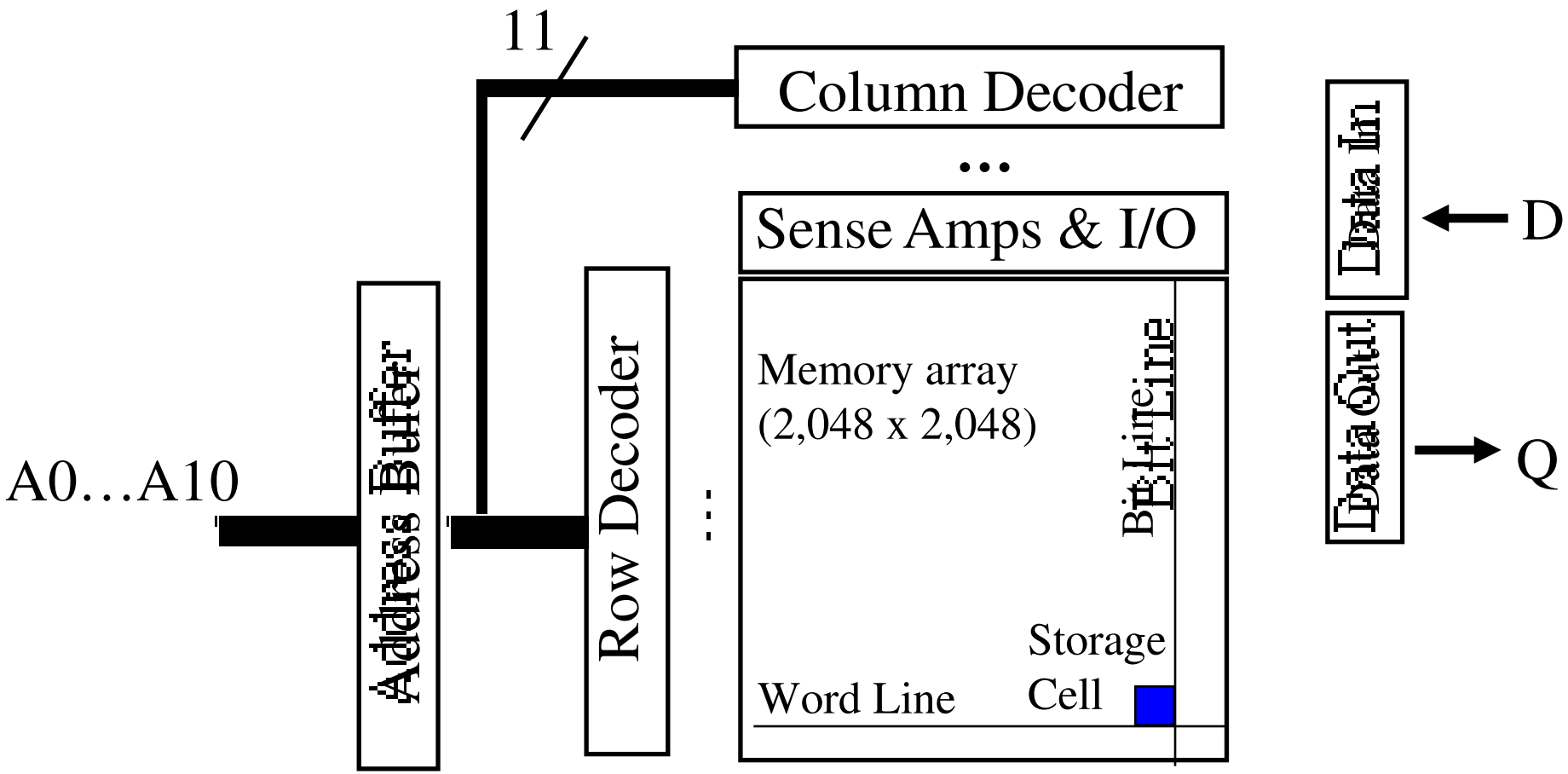
- Desempenho da memória Principal:
  - Latência: Cache Miss Penalty
    - » *Access Time*: tempo entre a requisição e a chegada da palavra
    - » *Cycle Time*: tempo entre requisições
  - Bandwidth: I/O & Large Block Miss Penalty (L2)

- Memória Principal: *DRAM* - Dynamic Random Access Memory
  - Dinâmica já que tem que ser atualizada (refreshed) periodicamente (8 ms)
  - Endereços divididos em duas metades (Memória como uma matriz 2D):
    - » *RAS ou Row Access Strobe*
    - » *CAS* ou *Column Access Strobe*

- Cache usa *SRAM* - Static Random Access Memory
  - Sem refresh (6 transistores/bit vs. 1 transistor)

  Size: DRAM/SRAM - *4-8*,
  Cost/Cycle time: SRAM/DRAM - *8-16*

DRAM: Organização lógica (4 Mbit)



Timing em DRAM

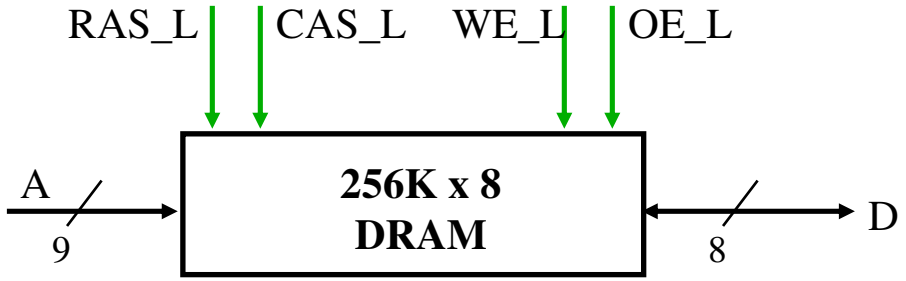
- t_{RAC}: Tempo mínimo entre a decida do sinal RAS e o dado válido na saída.
  - Considerado como a velocidade da DRAM
  - Valor típico para uma 4Mb DRAM t_{RAC} = 60 ns

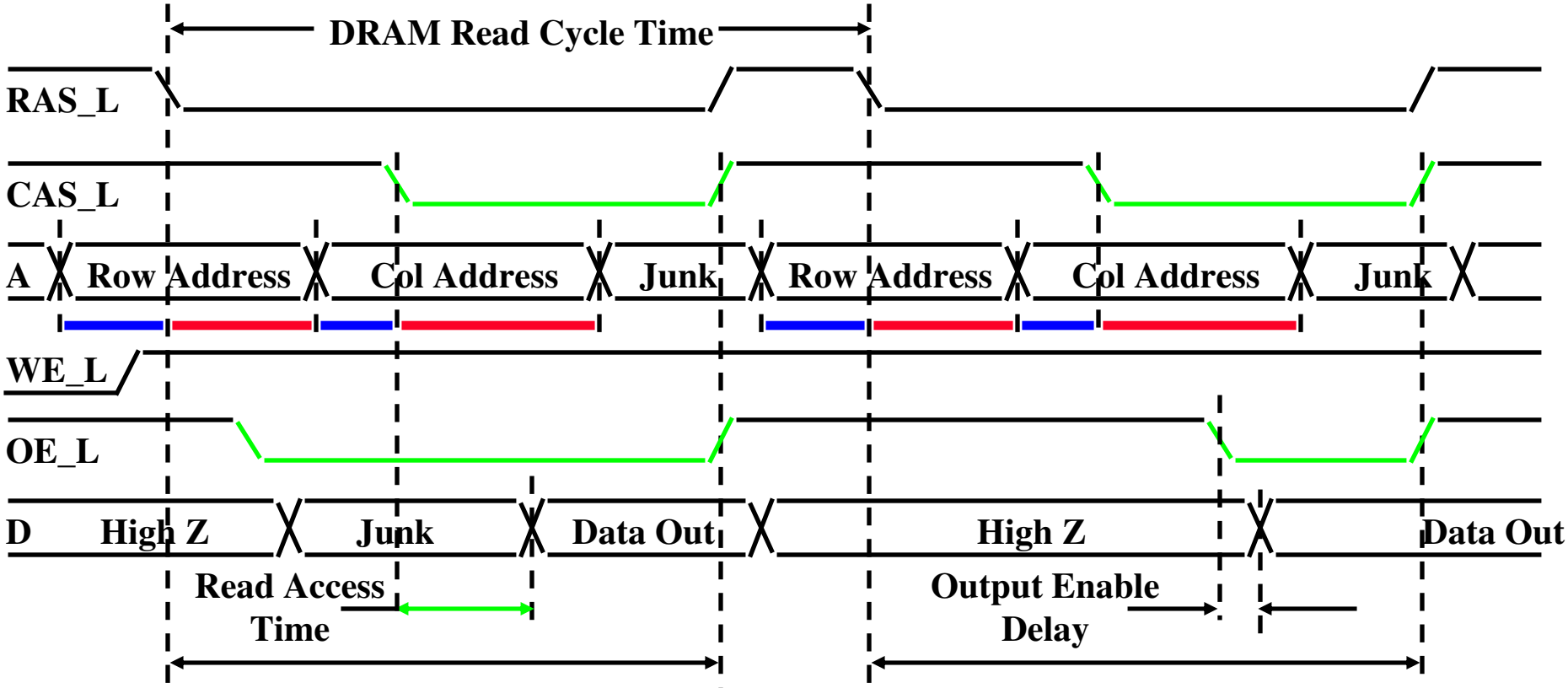
- t_{RC}: Tempo mínimo entre o início de um acesso (row) ao início do próximo acesso.
  - t_{RC} = 110 ns para uma 4Mbit DRAM com t_{RAC} de 60 ns

- t_{CAC}: Tempo mínimo entre a decida do sinal CAS e o dado válido na saída.
  - 15 ns para uma 4Mbit DRAM com t_{RAC} de 60 ns
- t_{PC}: Tempo mínimo entre o início de um acesso (column) ao início do próximo acesso.
  - 35 ns para uma 4Mbit DRAM com t_{RAC} de 60 ns

DRAM Read Timing

- Todo acesso a DRAM começa com:
  - Ativação de RAS_L
  - 2 formas de leitura: OE_L antes ou depois do CAS





Early Read Cycle: OE_L asserted before CAS_L

Late Read Cycle: OE_L asserted after CAS_L

DRAM - Desempenho

- Uma DRAM de 60 ns (t_{RAC}) pode:
  - Executar um acesso à uma nova linha somente a cada 110 ns (t_{RC})
  - Executar acesso à uma nova coluna (t_{CAC}) em 15 ns, porém o tempo entre dois acessos à colunas é no mínimo 35 ns (t_{PC}).
    - » Na prática o delay externo para o endereço e ajustes no barramento o torna de 40 a 50 ns
- OBS.: Estes tempos não inclui o tempo de endereçamento da cpu nem o overhead do controlador de memória.

DRAM - Histórico

- DRAMs: capacidade +60%/ano, custo –30%/ano
  - 2.5X cells/area, 1.5X die size em 3 anos
- '98 DRAM custo de fabricação $2B
  - DRAM: densidade, leakage v. velocidade
- Conta com o crescimento no nº computadores e de memória por computador (60%)
  - SIMM ou DIMM (dual inline memory modules)
    => computadores usam qualquer geração de DRAM
- Commodity
  => alto volume, conservador
  - Pequenas inovações na organização em 20 anos
- Ordem de importância: 1) Custo/bit; 2) Capacidade
  - Primeira RAMBUS: 10X BW, +30% custo => pouco impacto

DRAM - Histórico

- DRAM: RAS/CAS; RAS/CAS ....
- FPM (Fast Page Mode): RAS/CAS; CAS; CAS ... RAS/CAS (+30%)
- EDO (Extended Data Output): FPM + truques que permitem um acesso iniciar antes que o anterior termine (+25%)
- BEDO (Burst Extended Data Output RAM): pipeline no acesso (+30%)
- SDRAM (Synchronous Dynamic RAM): Acesso sincronizado com o clock da placa (PC-100, PC-133)
- DDR-SRAM (Double Data Rate SRAM): duas transferências de dado por ciclo de clock
- RDRAM (RAMBUS DRAM)

DRAM: 1 Gbit DRAM (ISSCC '96; produção prevista em '02?)

	Mitsubishi	Samsung
• Blocks	512 x 2 Mbit	1024 x 1 Mbit
• Clock	200 MHz	250 MHz
• Data Pins	64	16
• Die Size	24 x 24 mm	31 x 21 mm
– O tamanho pode ser menor quando produzido		
• Metal Layers	3	4
• Technology	0.15 micron	0.16 micron

Sistema de Memória Mais Rápida: DRAM

- Múltiplos acessos CAS: vários nomes (page mode)
  - *Extended Data Out (EDO)*: 30% mais rápida que page mode
- Novas DRAMs (tentando resolver o gap de endereçamento) Custo?, sobreviverão?

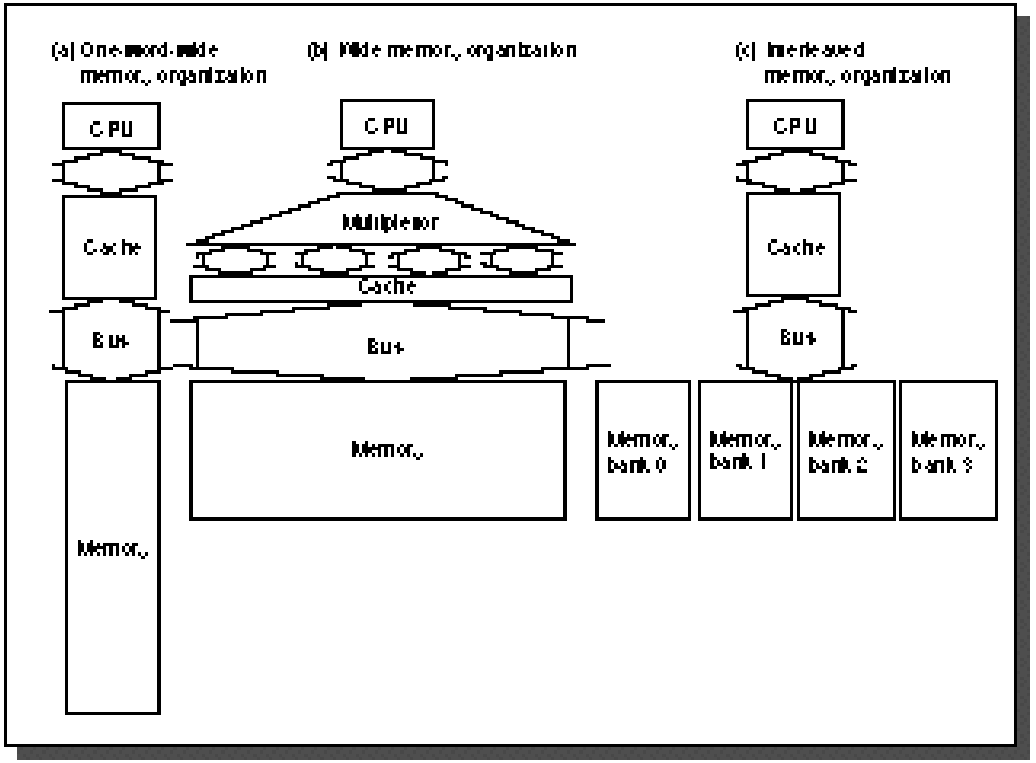
  - *RAMBUS* (startup company): interface DRAM reinventada
    - » Cada Chip é um módulo (mais um sistema do que um componente)
    - » Cada chip possui interleaved memory
    - » Define uma interface entre a CPU e os chips de memória
    - » Faz o próprio refresh; RAS/CAS substituídos por um barramento
    - » Retorna um número variável de dados
    - » 1 byte / 2 ns (500 MB/s por chip)
    - » Aumento de 20% na área da DRAM
    - » 1ª geração RDRAM; 2ª geração DRDRAM (direct RDRAM)

Sistema de Memória Mais Rápida: DRAM

- Novas DRAMs (tentando resolver o gap de endereçamento) Custo?, sobreviverão? – cont.
  - *Synchronous DRAM*:
    - » 2 bancos em um chip,
    - » um sinal de clock para a DRAM,
    - » transferência síncrona (com o clock do sistema 66-150 MHz)
- Nichos para memórias ou memória principal?
  - Exp:
    - » Video RAM para frame buffers, DRAM + saída serial mais rápida, …
    - » Memórias para sistemas embarcados (ROM, Flash)
    - » Sistema de memória para SoC (interface com NoC)

Memória Principal: Organização

- ***Simple***:
  - CPU, Cache, Bus, Memória: mesmo tamanho de barramento (32 or 64 bits)
- ***Wide***:
  - CPU/Mux: 1 word; Mux/Cache, Bus, Memória: N words (Alpha: 64 bits & 256 bits; UtraSPARC 512)
- ***Interleaved***:
  - CPU, Cache, Bus: 1 word: N Módulos de Memória (exp. 4 Módulos; *word interleaved)*

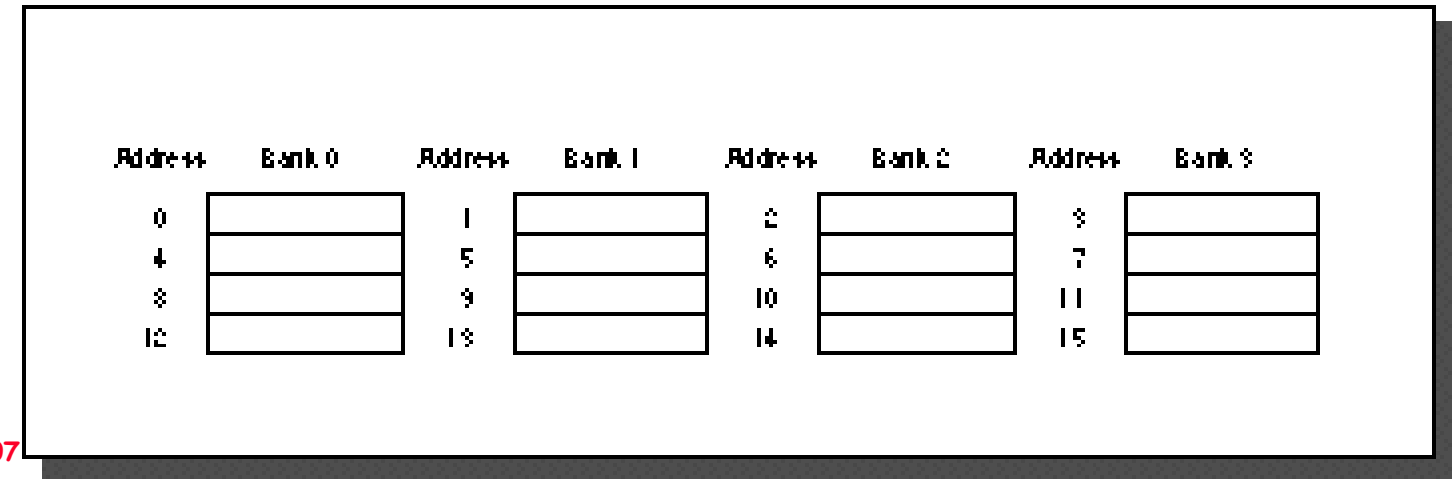


Memória Principal: Desempenho

Exemplo:

- **Modelo de Timing (word size = 32 bits)**
  - 1 para envio do endereço,
  - 6 de access time, 1 para envio do dado
  - Cache Block = 4 words

- ***Simple M.P.*** = 4 x (1+6+1) = 32
- ***Wide M.P.*** = 1 + 6 + 1 = 8
- ***Interleaved M.P.*** = 1 + 6 + 4x1 = 11

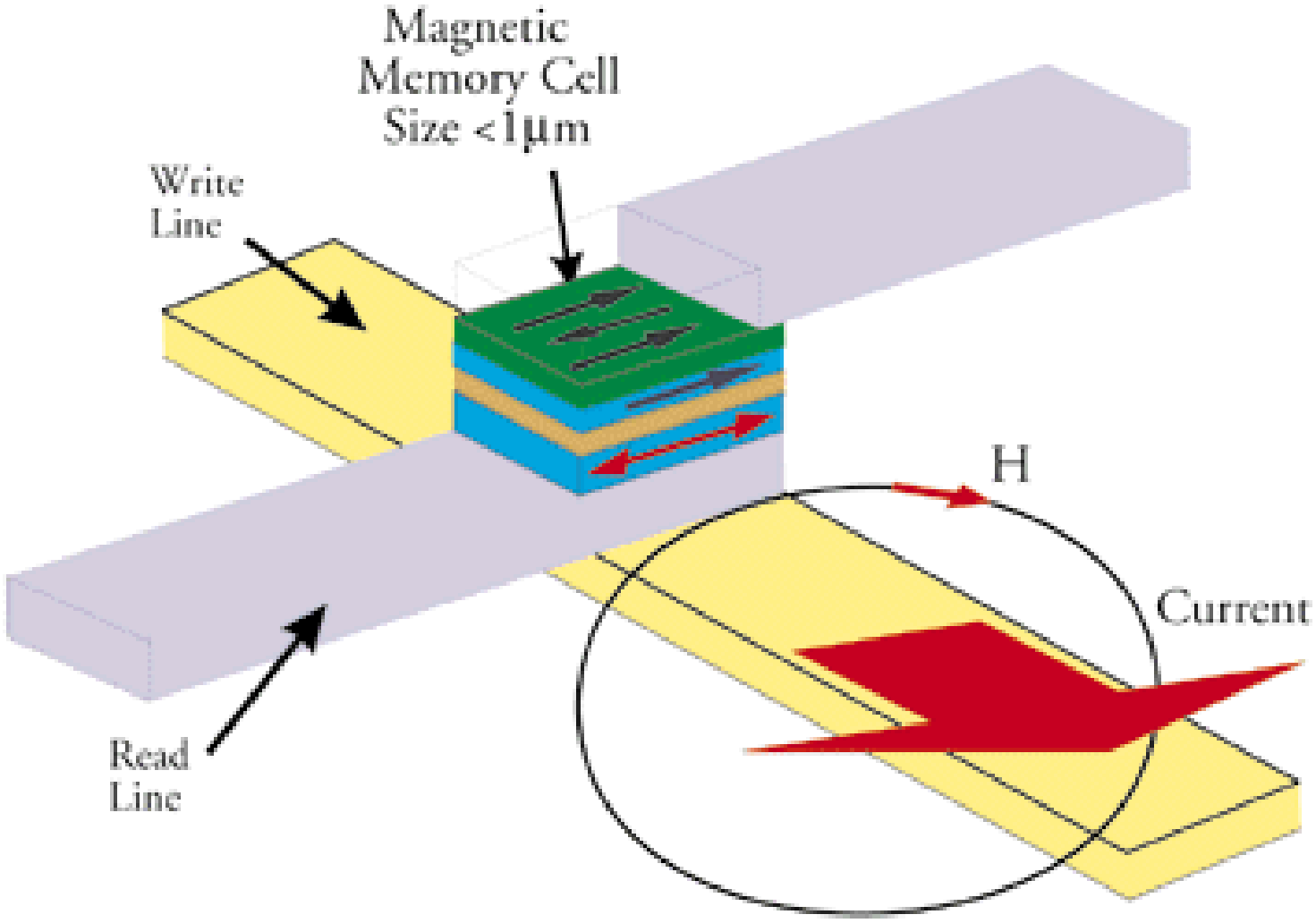


Técnica	Miss Penalty	Miss Rate	Hit Time	Complex HW	OBS.
Multlevel Cache	+			2	
Word Crítica primeira e reinicio antecipado	+			2	Bastante usado
Prioridade para read miss sobre write miss	+			1	Fácil para 1 proc., usado
Merge write buffer	+			1	Usado com write through
Victim cache	+	+		2	AMD Athlon (8 entradas)
Bloco maior	-	+		0	Fácil, Pentium 4 L2 usa 128 bytes
Maior cache		+	-	1	Muito usado, L2
Maior associatividade		+	-	1	Muito usado
Way-prediction cache		+		2	Usado em I$ (sparc) e D$ (R4300)
Pseudo-associatividade		+		2	Usaddo em L2 do MIPS R10000
Campilador – cache miss		+		0	
Nonblocking cache	+			3	Usado em cpu out-of-order
Hw prefetching (I/D)	+	+		2/3	Usado
SW prefetching	+	+		3	Necessita de Nonblocking cache
Caches simples e pequena	-	-	+	0	Fácil, muito usada
Evitar transf. de enderreços			+	2	Fácil se cache pequena
Pipeline cache access			+	1	Bastante usado
Trace cache			+	3	Usado no Pentium 4

Outras tecnologias de Armazenamento

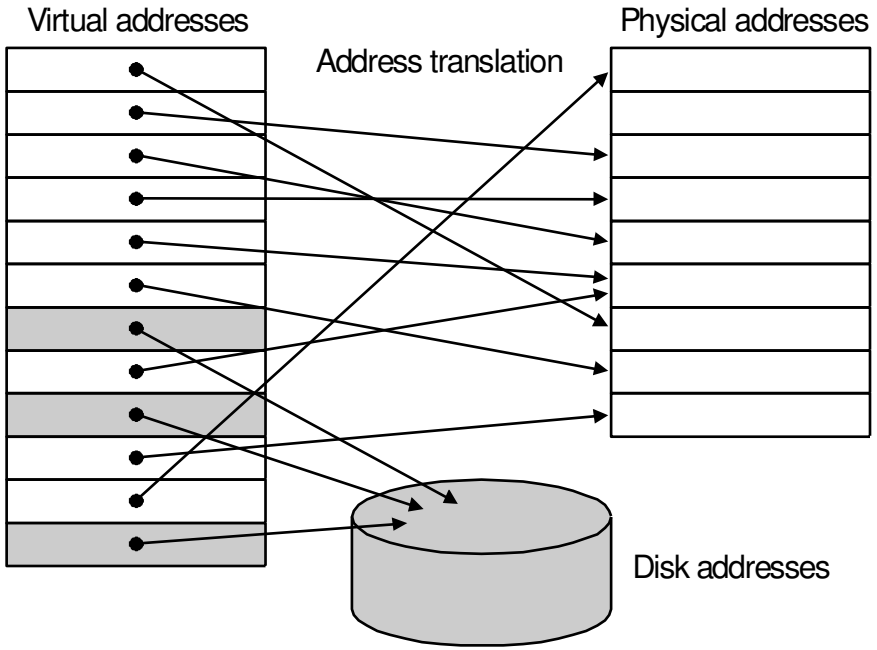
- Tunneling Magnetic Junction RAM (TMJ-RAM):
  - Velocidade de SRAM, densidade de DRAM, não volátil (sem refresh)
  - Novo campo denominado "Spintronics": combinação de "quantum spin" e eletrônica
  - Tecnologia usada em em disk-drives de alta densidade

Tunneling Magnetic Junction



Memória Virtual

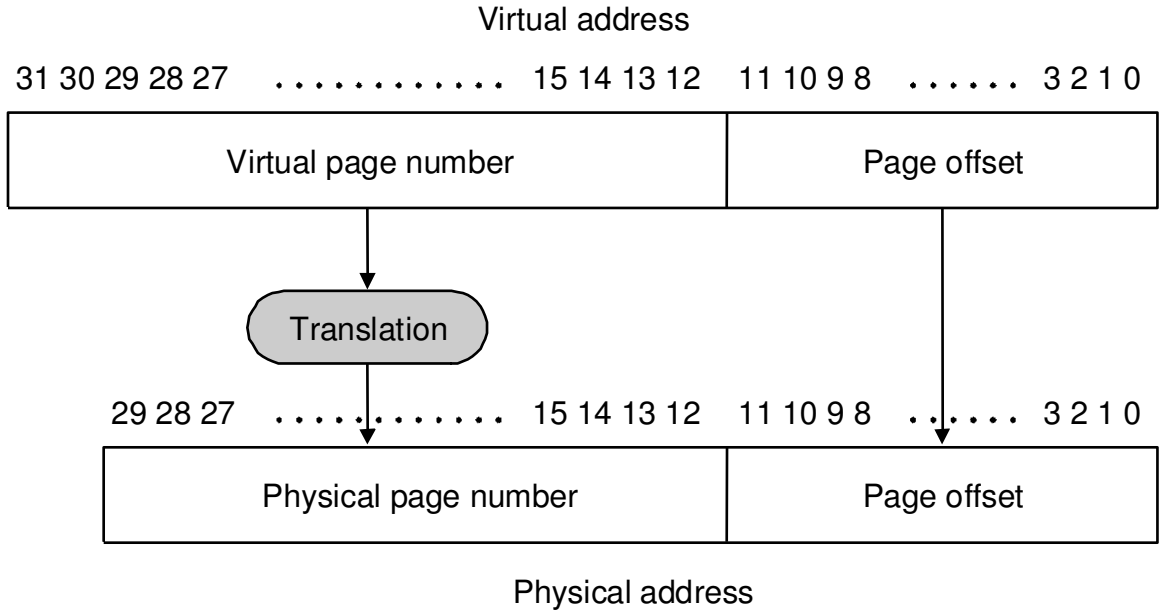
- A memória principal pode agir como uma cache para o armazenamento secundário (disco)



- Vantagens:
  - Ilusão de se ter mais memória física (o programa independe da configuração do hardware)
  - Realocação de programas
  - proteção (address space)

Páginas: virtual memory blocks

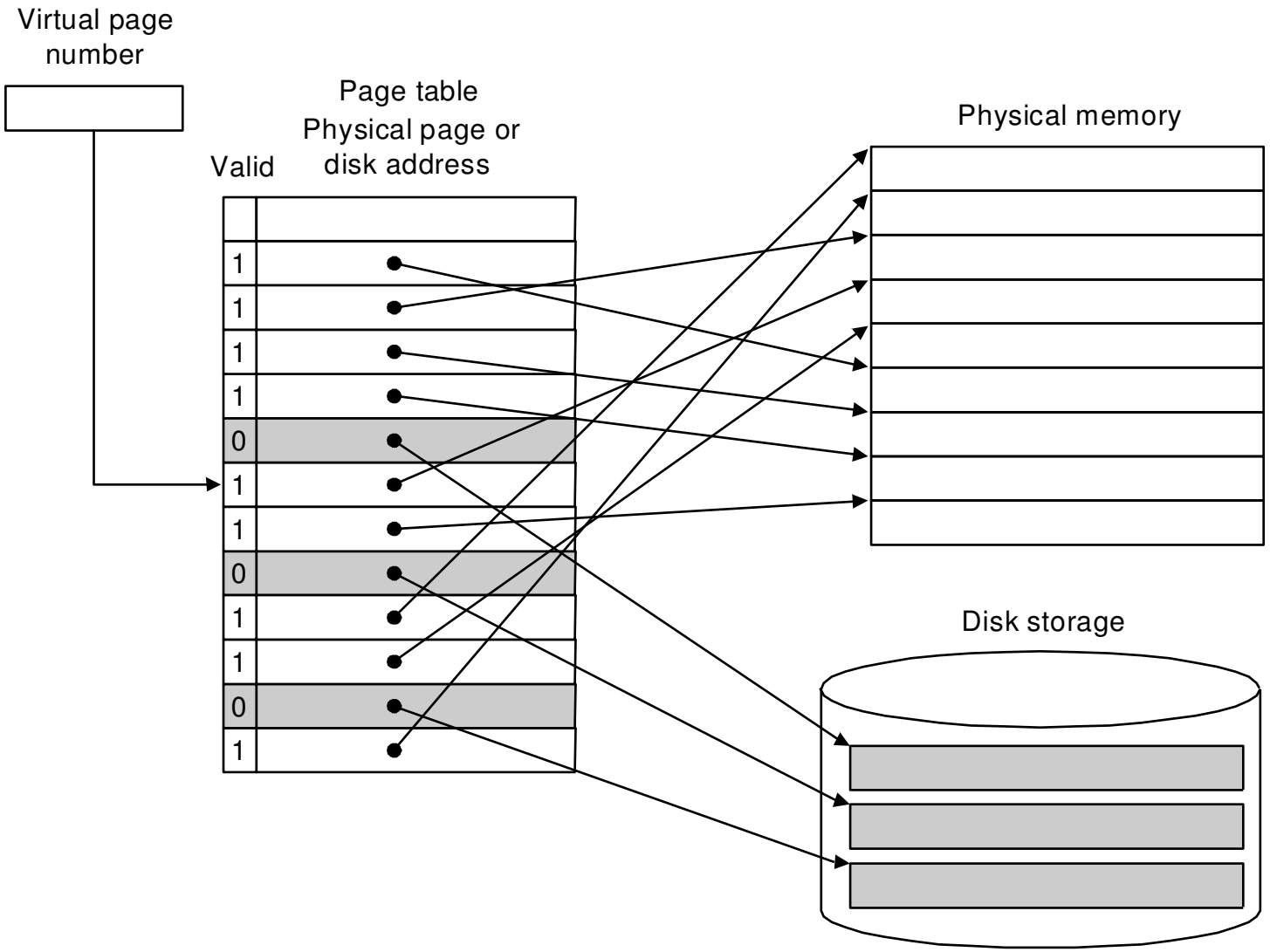
- **Page faults**: o dado não está na memória, deve ser feito um acesso ao disco
  - miss penalty elevado, assim as páginas devem ser "grandes" (e.g., 4KB)
  - reduzir page faults é **muito** importante (LRU)
  - faults são tratadas por software e não por hardware
  - Uso de write-through é muito dispendioso -> usa-se write-back



12 bits: 4 KB
VPN: 20 bits
1M page
4 GB

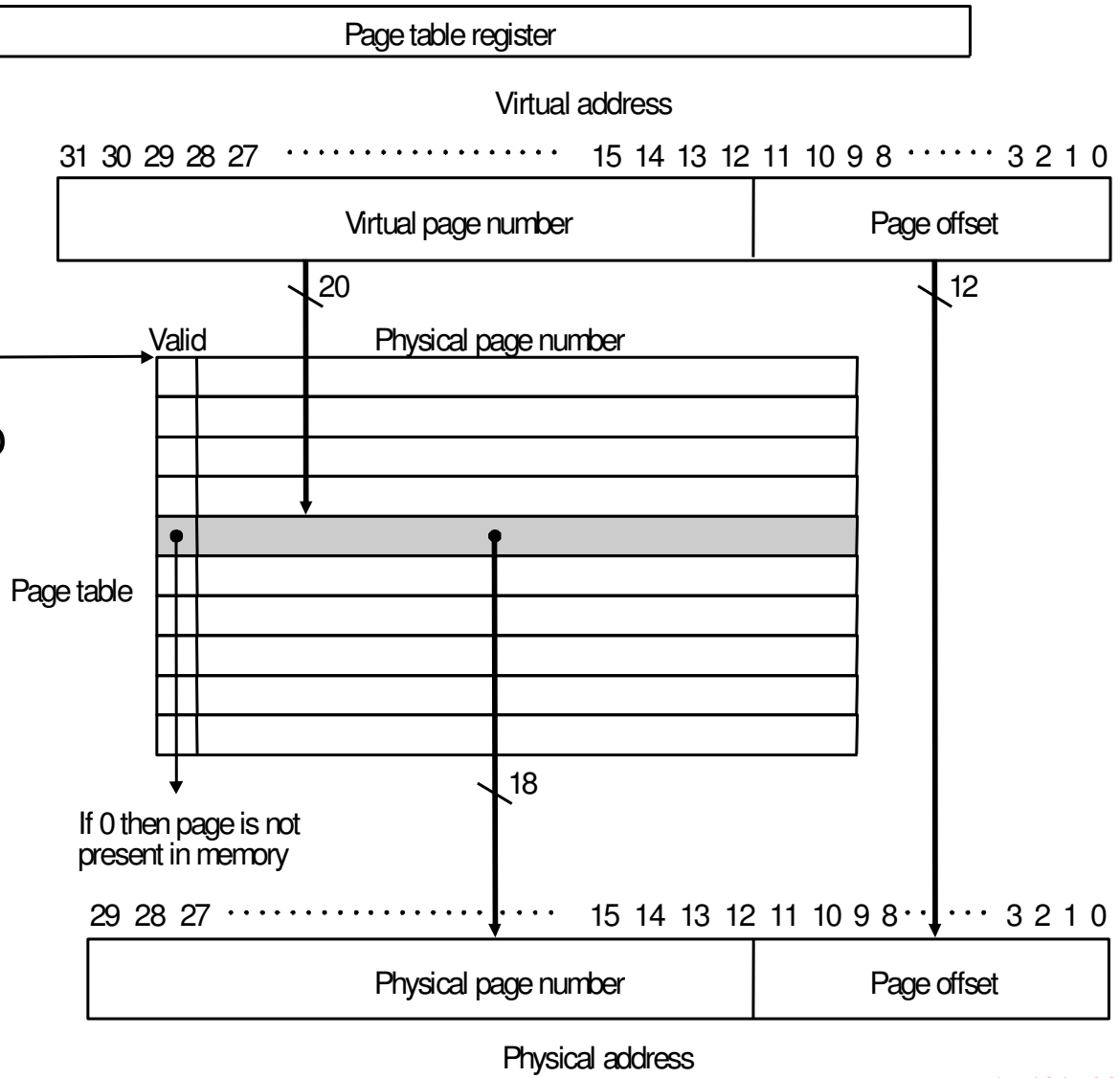
PPN: 18 bits
256 K page
1 GB

Tabelas de Páginas



Tabelas de Páginas

- uma Tabela por processo
- estado (do processo):
  - PT – page table
  - PC
  - registradores



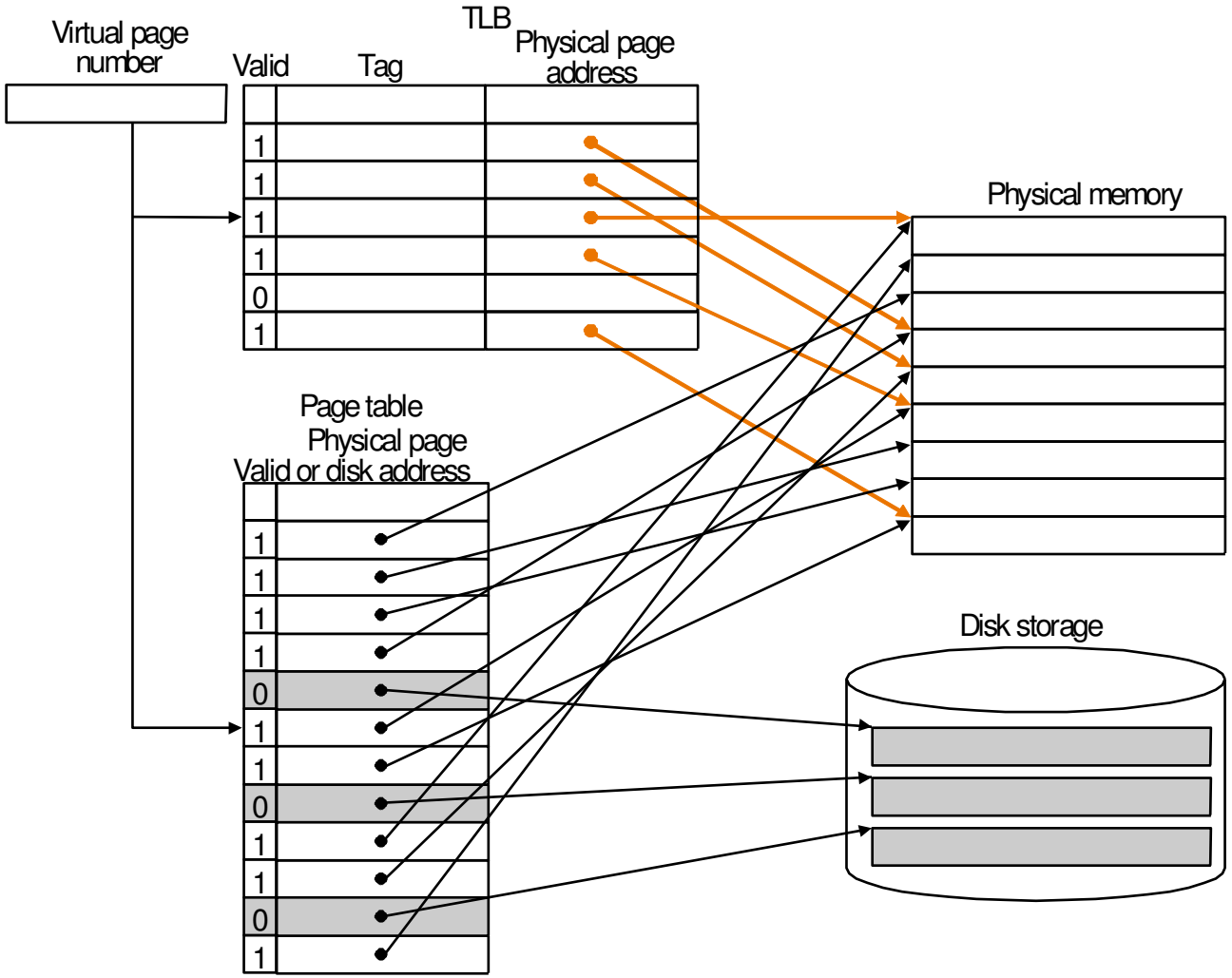
Política de Substituição e Tamanho da PT

- Se page fault (bit válido = 0)
  - sistema operacional executa a carga da página
- Para minimizar page faults, política de substituição mais usada: LRU

- Tamanho da PT (exemplo: endereço de 32 bits, páginas de 4KB e 4B por linha da PT)
  - número de linhas: $2^{32} / 2^{12} = 2^{20}$
  - tamanho da PT = 4 MB ($4B \times 2^{20}$)
  - 1 PT por programa ativo !!
  - para reduzir área dedicada para PT: registradores de limite superior e inferior

- PT também são paginados

TLB: Translation Lookaside Buffer

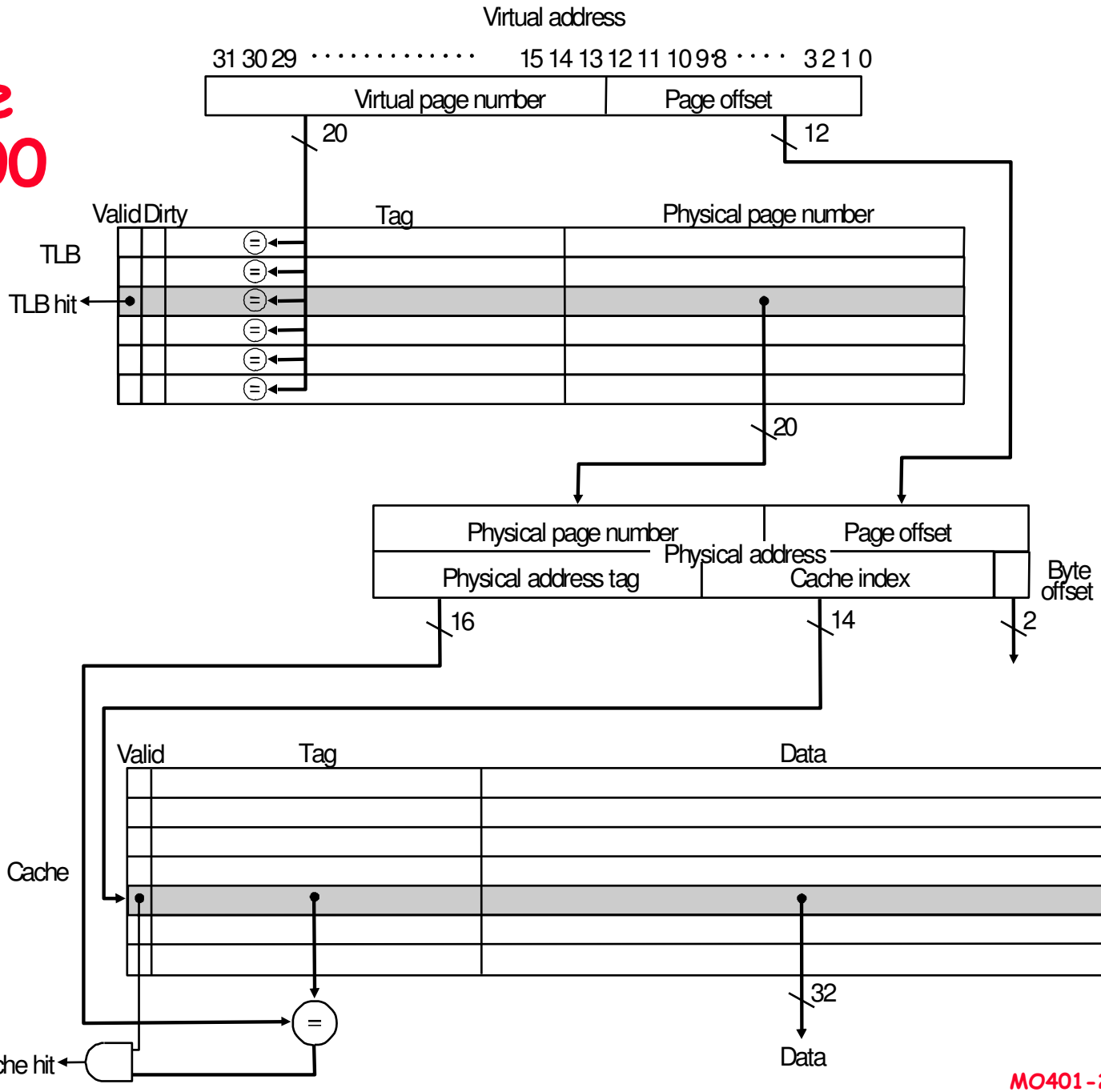


Valores típicos

- TLB size: 32 - 4,096 entries
- Block size: 1 - 2 page table entries
- Hit time: 0.5 - 1 clock cycle
- Miss penalty: 10 - 30 clock cycle
- Miss rate: 0.01% - 1%
- map direto ou fully associativo

TLBs e Cache DEC 3100

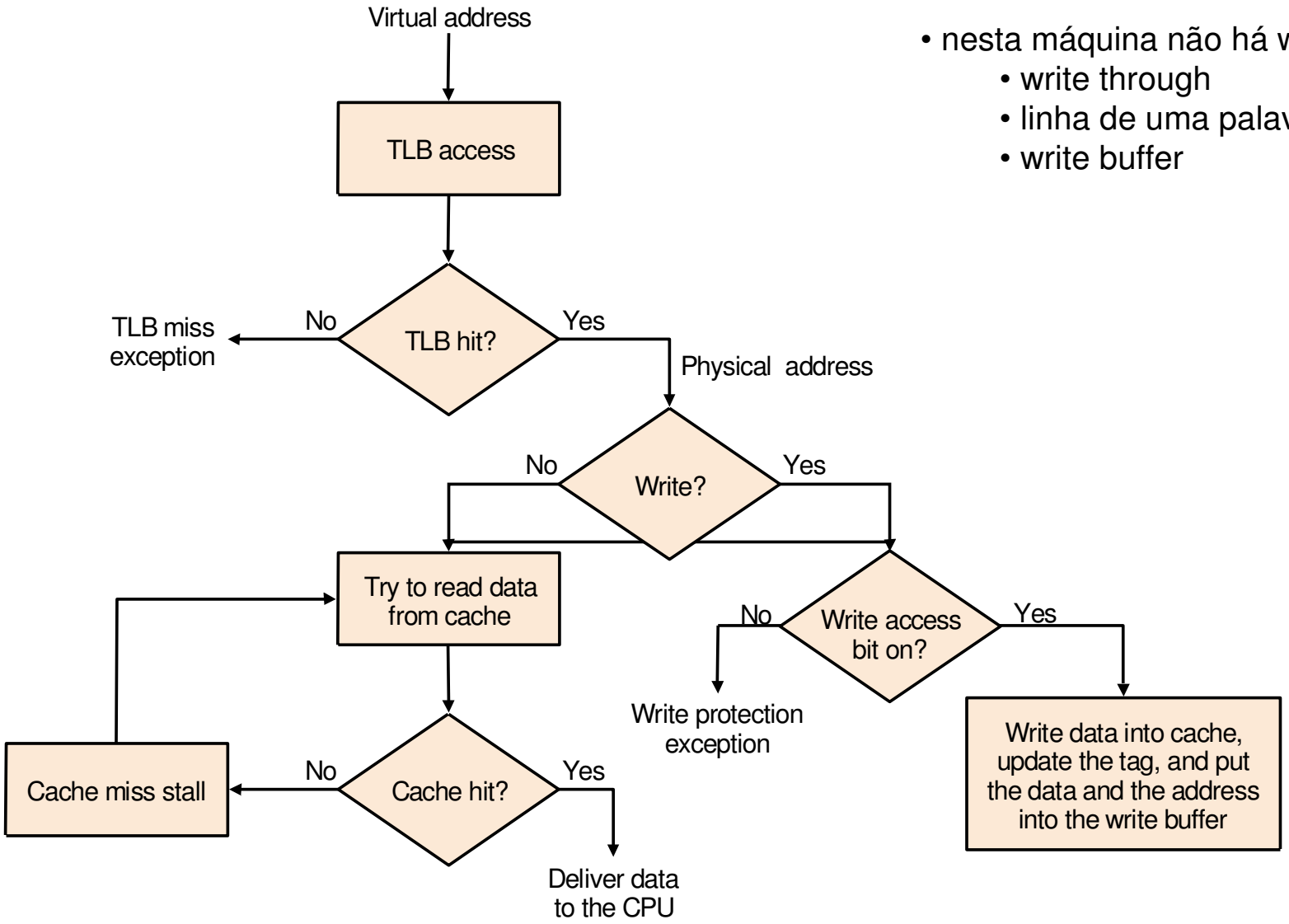
- mapeamento fully associative



- mapeamento direto

- pior caso: 3 misses TLB, PT, cache

TLBs e Caches (DEC 3100)



- nesta máquina não há write hit
  - write through
  - linha de uma palavra
  - write buffer

TLB, Memória Virtual e Cache

Cache	TLB	Virtual memory	Possible? If so, under what circumstance?
Miss	Hit	Hit	Possible, although the page table is never really checked if TLB hits.
Hit	Miss	Hit	TLB misses, but entry found in page table; after retry data is found in cache.
Miss	Miss	Hit	TLB misses, but entry found in page table; after retry data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Miss	Hit	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Hit	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Miss	Impossible: data cannot be allowed in cache if the page is not in memory.

Proteção com Memória Virtual

- Suporte a no mínimo dois modos de execução
  - Processo de usuário
  - Processo do Sistema Operacional (kernel, supervisor, executive)

- A CPU no estado processo de usuário pode ler, porém não pode escrever a *page table* e *TLB*
  - Instruções especiais disponíveis somente no modo supervisor
- Mecanismo para a CPU tocar o modo de execução de usuário para supervisor e vice e versa
  - Usuário para supervisor : *system call exception*
  - supervisor para usuário : *return from exception (RFE)*

- OBS: as page tables ficam no espaço de endereçamento do sistema operacional

Page Faults e TLB misses

- TLB miss (*software ou hardware*).
  - Se a página está na memória, é necessário somente criar uma nova entrada na TLB.
  - Se a página não está na memória, é necessário transferir o controle ao sistema operacional para tratar a page fault.
- Page fault (mecanismo de *exceção*).
  - OS salva o estado do processo ativo.
  - EPC = virtual address da faulting page.
  - OS deve executar três passos:
    - » Usando o endereço virtual, procurar a entrada na page table e encontrar a localização da página no disco.
    - » Escolher uma página física para ser trocada; se a página escolhida está dirty, ela deve ser escrita no disco.
    - » Iniciar a leitura da página no disco e coloca-la na memória.

Hierarquia de Memória: Cache e Memória Virtual

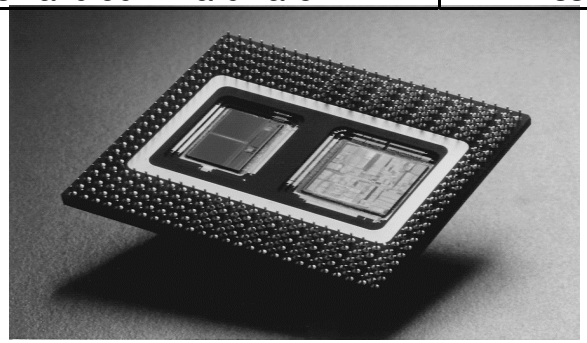
Scheme name	Number of sets	Block per set
Direct mapped	Number of blocks in cache	1
Set associative	$\frac{\text{Number of blocks in cache}}{\text{Associativity}}$	Associativity (typically 2 – 8)
Fully associative	1	Number of block in the cache

Feature	Typical values for cache	Typical values for page memory	Typical values for a TLB
Total size in blocks	1000 –100,000	2000 – 250,000	32 – 4,000
Total size in kilobytes	8 – 8,000	8000 – 8,000,000	0.254 – 32
Block size in bytes	16 – 256	4000 – 64,000	4 – 32
Miss penalty in clocks	10 – 100	1,000,000 – 10,000,000	10 – 100
Miss rate	0.1% -- 10%	0.00001% -- 0.0001%	0.01% -- 2%

Sistemas Modernos

- **Sistemas de Memórias mais Complexos :**

Characteristic	Intel Pentium Pro	PowerPC 604
Virtual address	32 bits	52 bits
Physical address	32 bits	32 bits
Page size	4 KB, 4 MB	4 KB, selectable, and 256 MB
TLB organization	A TLB for instructions and a TLB for data	A TLB for instructions and a TLB for data
	Both four-way set associative	Both two-way set associative
	Pseudo-LRU replacement	LRU replacement
	Instruction TLB: 32 entries	Instruction TLB: 128 entries
	Data TLB: 64 entries	Data TLB: 128 entries
	TLB misses handled in hardware	TLB misses handled in hardware



Characteristic	Intel Pentium Pro	PowerPC 604
Cache organization	Split instruction and data caches	Split intruction and data caches
Cache size	8 KB each for instructions/data	16 KB each for instructions/data
Cache associativity	Four-way set associative	Four-way set associative
Replacement	Approximated LRU replacement	LRU replacement
Block size	32 bytes	32 bytes
Write policy	Write-back	Write-back or write-through

Melhorando O desempenho da Cache

1. Reduzindo o miss rate,
2. Reduzindo o miss penalty, ou
3. *Reduzindo o tempo de hit na cache*.

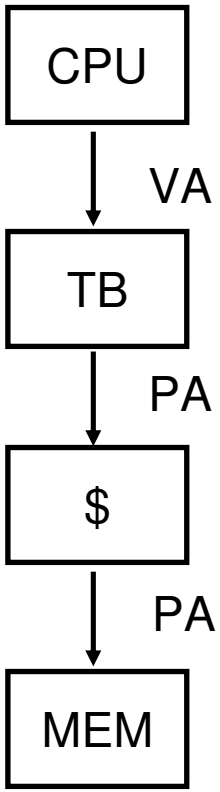
$$AMAT = HitTime + MissRate \times MissPenalty$$

Hit Times Rápido
via Caches Pequenas e Simples

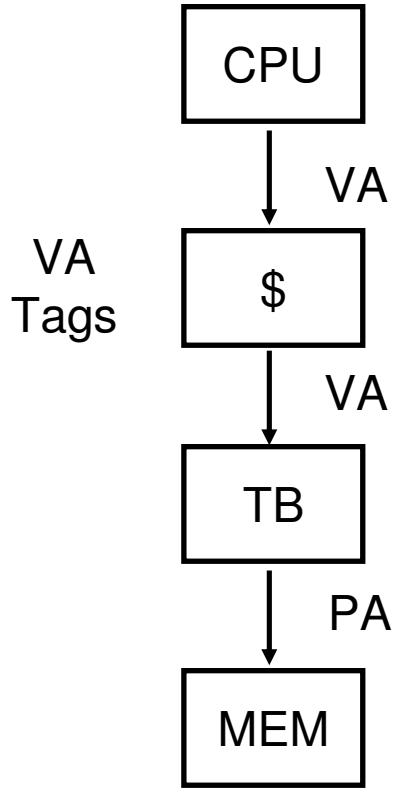
- Por que o Alpha 21164 tem 8KB Instruction e 8KB data cache + 96KB second level cache?
  - clock rate

- Direct Mapped, no chip

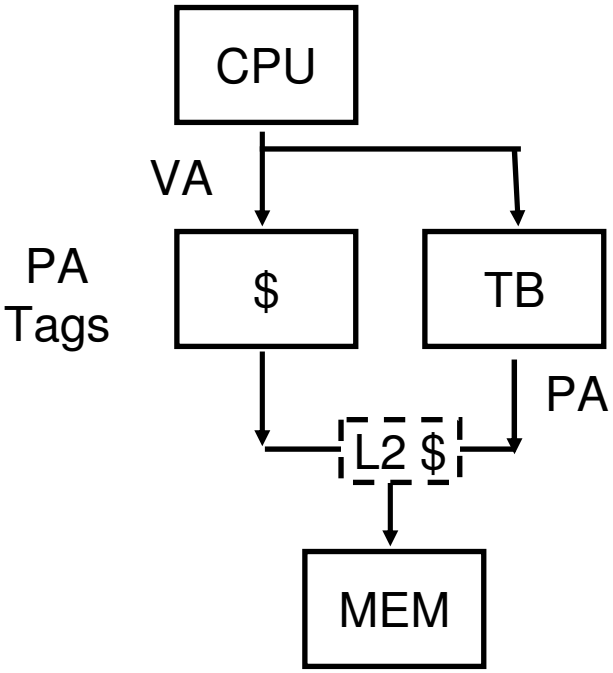
Hits Rápidos Evitando-se Address Translation



Conventional Organization



Virtually Addressed Cache
Translate only on miss
Synonym Problem



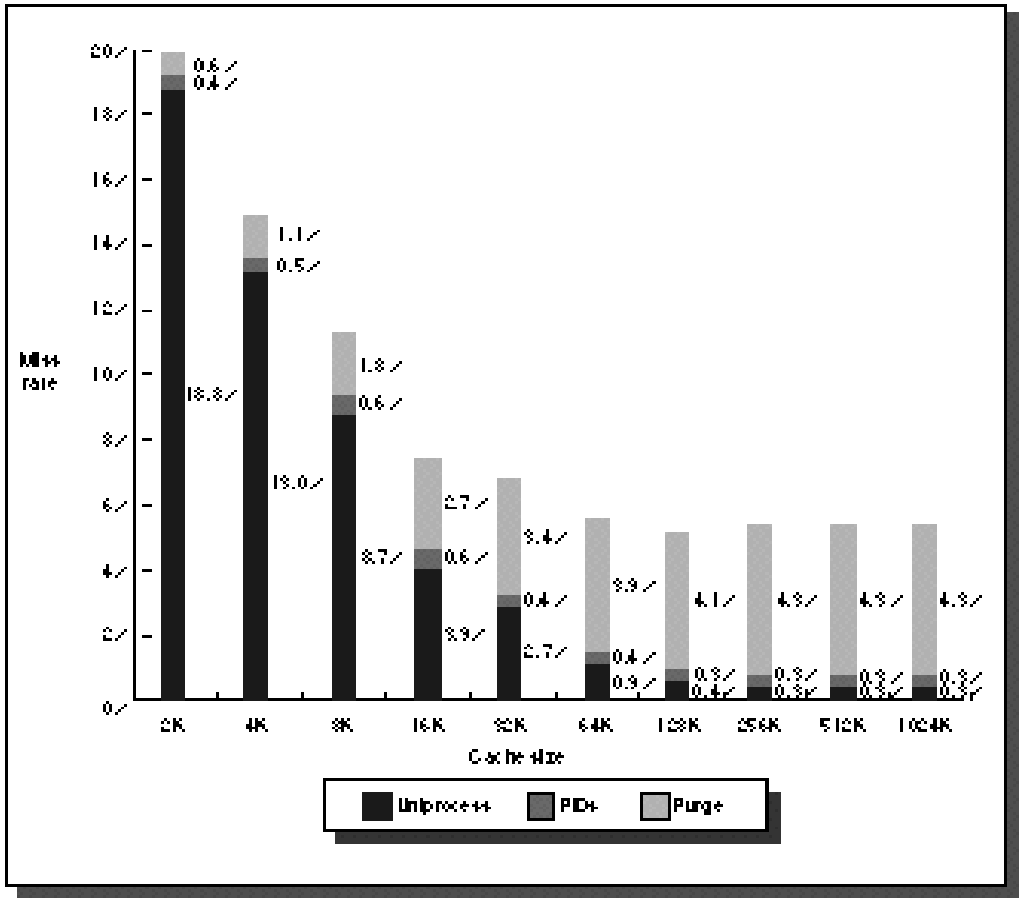
Overlap $ access with VA translation: requires $ index to remain invariant across translation

Hits Rápidos Evitando-se Address Translation

- Enviar o virtual address para a cache? Chamado *Virtually Addressed Cache* ou *Virtual Cache* vs. *Physical Cache*
  - A todo tempo processos são trocados e a cache deve ser esvaziada; se não haverá falsos hits
    - » O Custo é o time to flush + misses compusórios para a cache vazia
  - Tratar *aliases* (também chamado *synonyms*); Dois endereços virtuais diferentes são mapeados em um mesmo endereço físico
  - I/O deve interagir com a cache, então é necessário virtual address
- Solução para aliases
  - O HW garante cobertura para o index field & direct mapped, e eles devem ser únicos; chamado de *page coloring*
- Solução para cache flush
  - Adicionar um *process identifier tag* que identifica o processo bem como os seus endereços: não pode haver um hit se o processo for errado

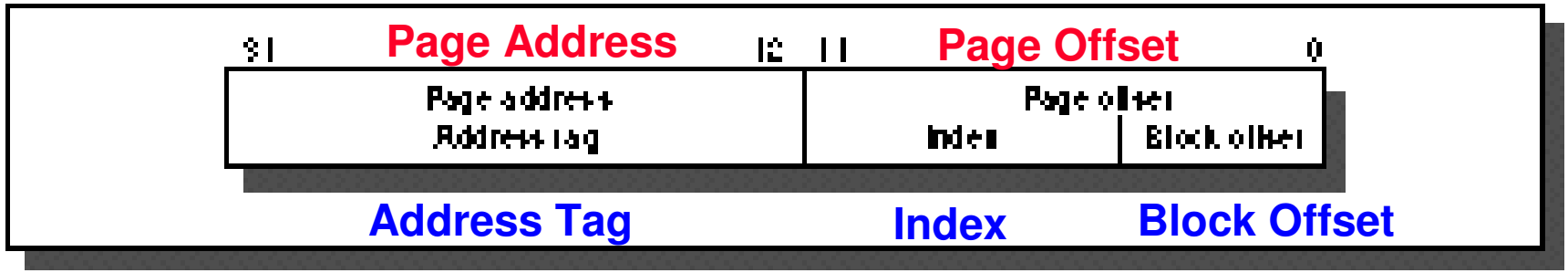
Hits Rápidos Evitando-se Address Translation: Impacto do Process ID

- Preto - uniprocesso
- Cinza claro - multiprocessos com flush cache
- Cinza escuro - multiprocessos com Process ID tag
- Eixo Y: Miss Rates até 20%
- Eixo X: Cache size de 2 KB a 1024 KB



Hits Rápidos Evitando-se Address Translation: Index como Parte do Endereço

- Se o index é parte do Page Offset, pode-se iniciar o acesso à tag em paralelo com o Address Translation

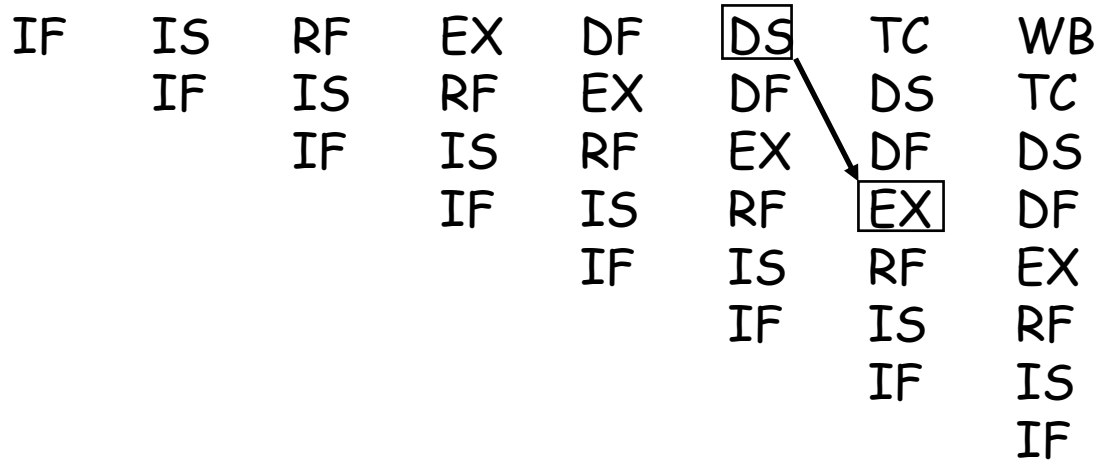


Hits Rápidos por Pipelining Cache
Estudo de Caso: MIPS R4000

- 8 Stage Pipeline:
  - IF – primeira metade do fetching de instrução: seleção do PC e início do acesso à cache de instrução.
  - IS – segunda metade do acesso à cache de instruções.
  - RF – instruction decode e register fetch, hazard checking e detecção de instruction cache hit.
  - EX – execução: inclui cálculo do effective address, operações da ALU e cálculo do branch target e avaliação da condição do branch.
  - DF – data fetch, primeira metade do acesso à cache de dados.
  - DS – segunda metade do acesso à cache de dados.
  - TC – tag check, determina se há hit no acesso ao dado.
  - WB – write back para loads e operações register-register.
- Qual o impacto sobre o Load delay?
  - Necessita 2 instruções entre o load e o uso do dado carregado! (2 delay slots)

Estudo de Caso: MIPS R4000

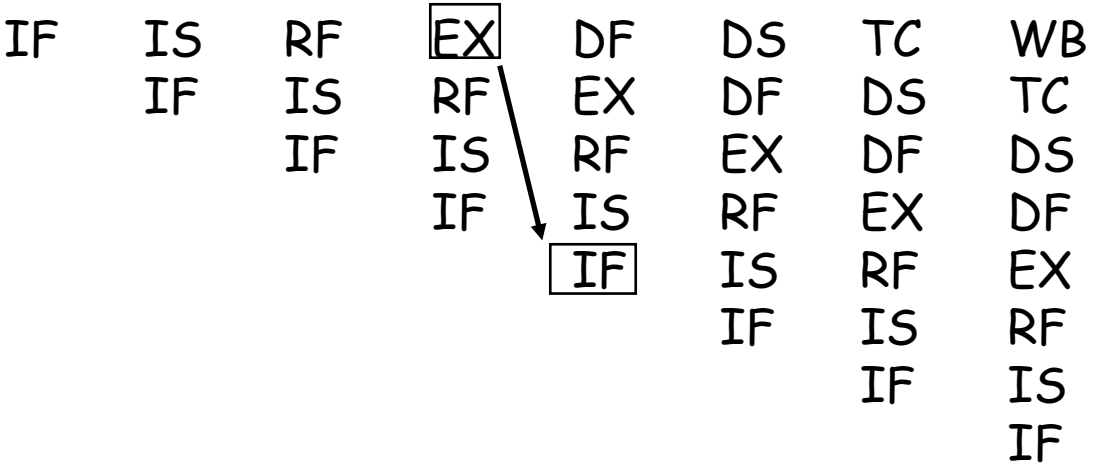
TWO Cycle Load Latency



THREE Cycle Branch Latency

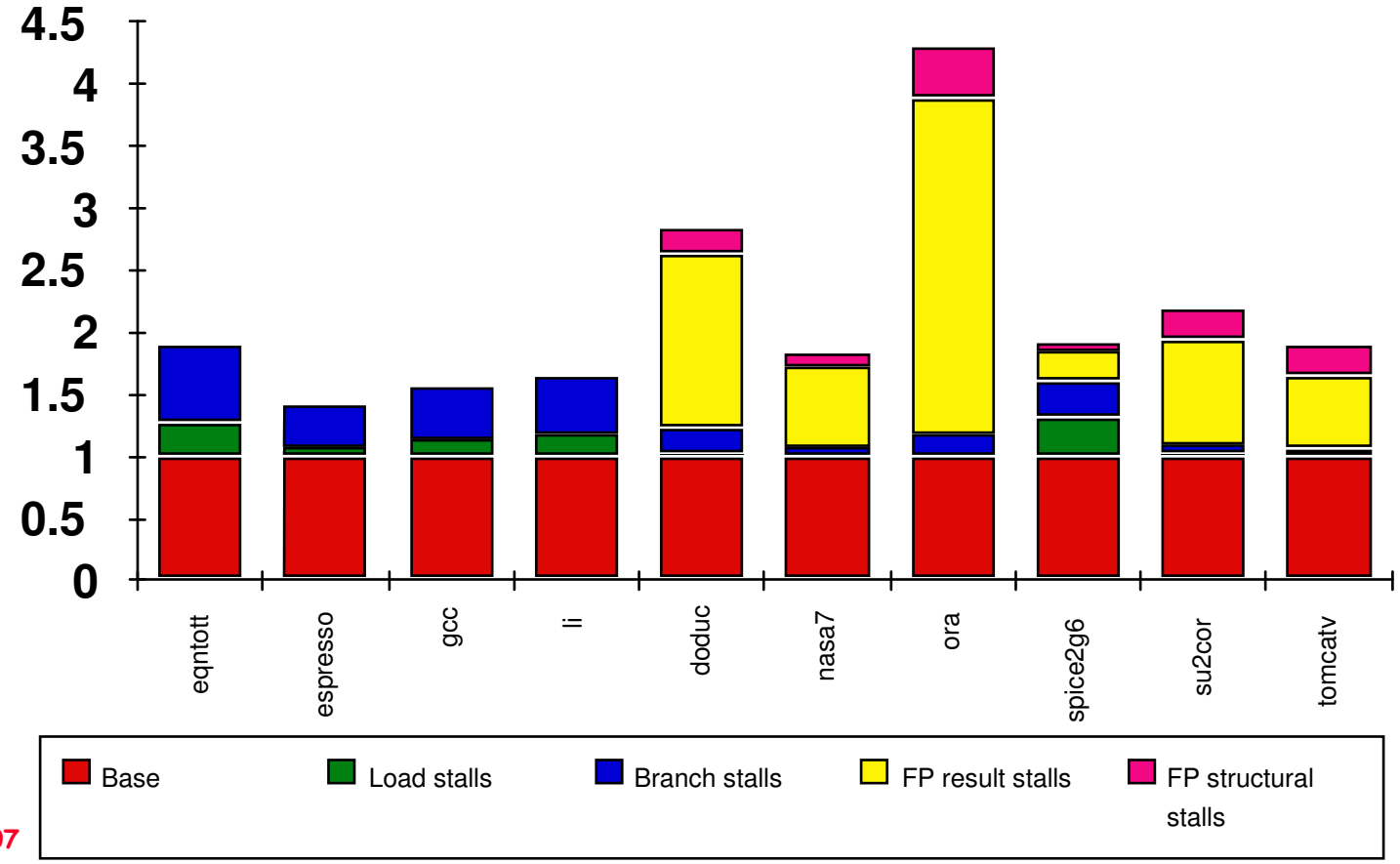
(Condição avaliada durante EX)

2 Delay slot



R4000: Desempenho

- Not ideal CPI of 1:
  - Load stalls (1 ou 2 clock cycles)
  - Branch stalls (2 cycles + unfilled slots)
  - FP result stalls: RAW data hazard (latency)
  - FP structural stalls: Not enough FP hardware (parallelism)



MO401-2007
Revisado

MO401-2006
8.86

ERROR: syntaxerror
OFFENDING COMMAND: --nostringval--

STACK:

331
/pp_by2