Insper

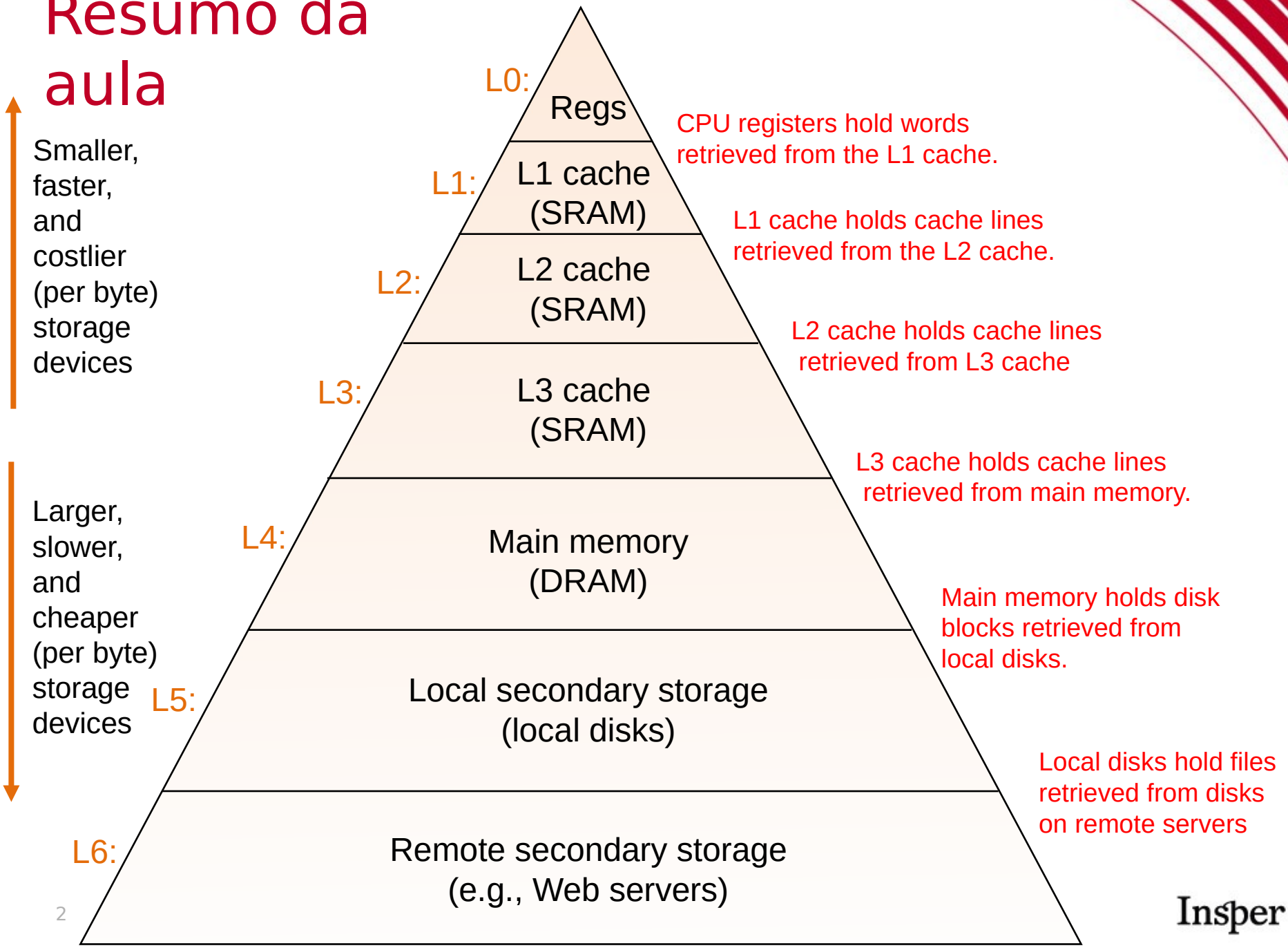
Sistemas Hardware-Software

Aula 13 – Hierarquia de memória

2019 – Engenharia

Igor Montagner, Fábio Ayres <igorsm1@insper.edu.br>

Resumo da aula



Insper

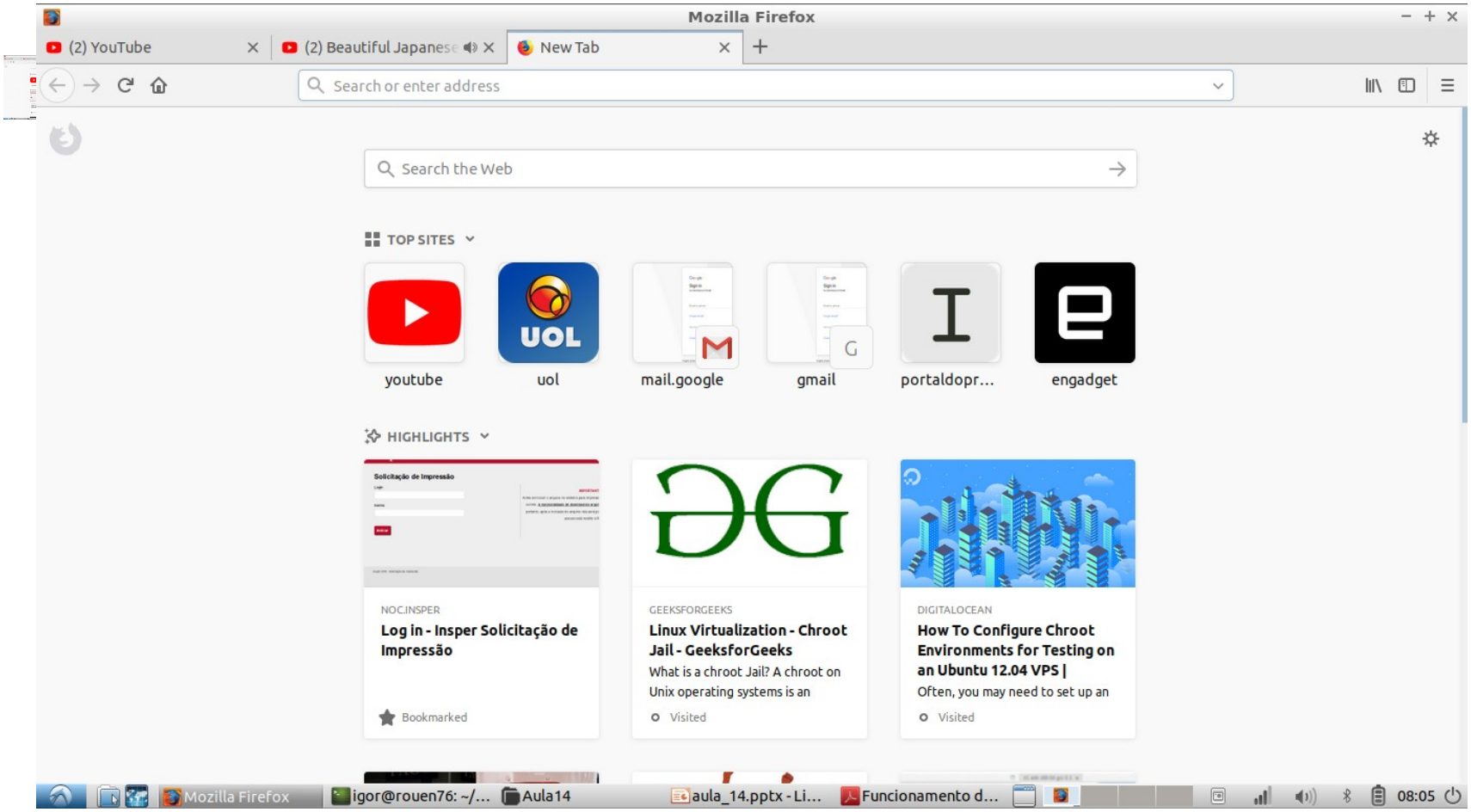
Agenda

Hierarquia de memória

Tecnologias de memória e armazenamento

Memória Cache

Estudo de caso



Navegador de Internet

Insper

Estudo de caso

Quando acessamos um site,

1) O navegador envia uma requisição para um endereço
2) e recebe de volta uma resposta em HTML
3) Ele então interpreta o HMTL e começa a baixar
    - Imagens
    - Scripts
    - Arquivos de estilo CSS
4) Quando tudo acaba a página está pronta e pode ser usada

Estudo de caso

Quando acessamos um site novamente,

1) É preciso baixar tudo de novo, do zero?

Quando acessamos um site que carrega um script de uma CDN que já foi usado em outro site,

2) Precisamos baixá-lo de novo?

Grande ideia da aula de hoje

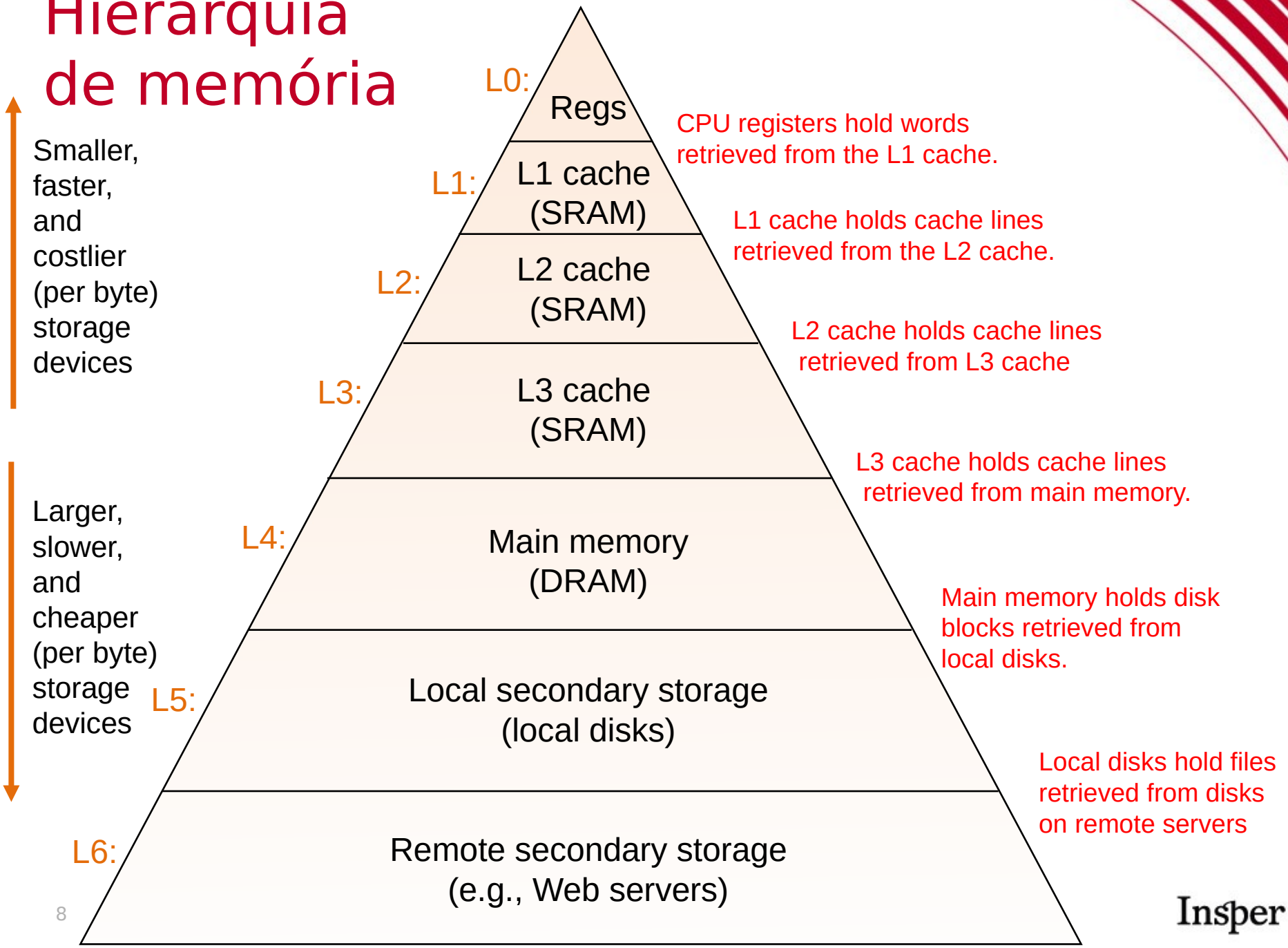
Um sistema possui vários tipos de memórias com diferentes velocidades. Armazenamos

- Em memórias mais rápidas os dados mais frequentemente usados
- Em memórias mais lentas os dados menos frequentemente usados

Levando em conta

- Temporalidade: um dado é acessado repetidamente
- Localidade: dados são acessados de maneira conjunta

Hierarquia de memória



Insper

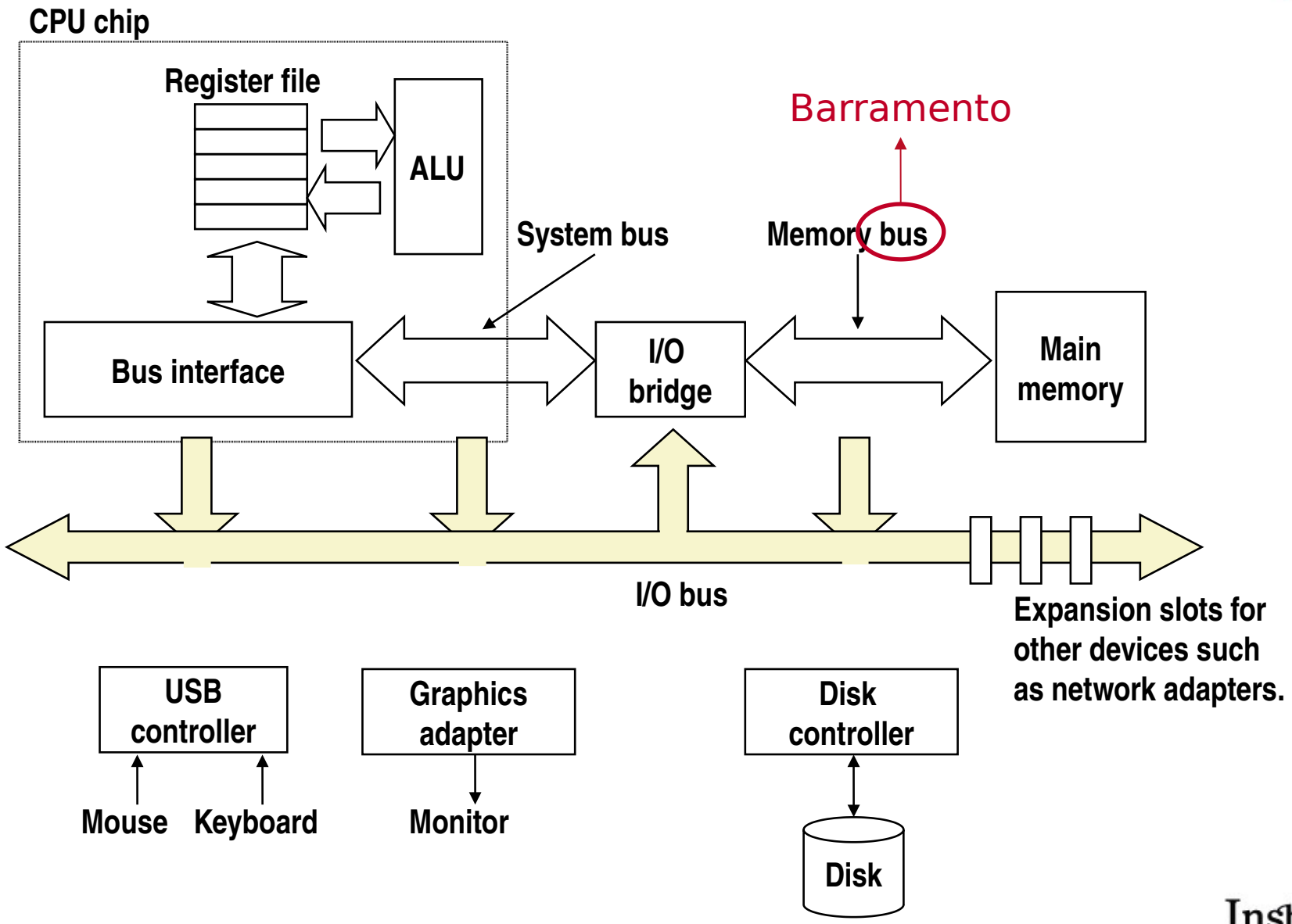
Agenda

Hierarquia de memória

Tecnologias de memória e armazenamento

Memória Cache

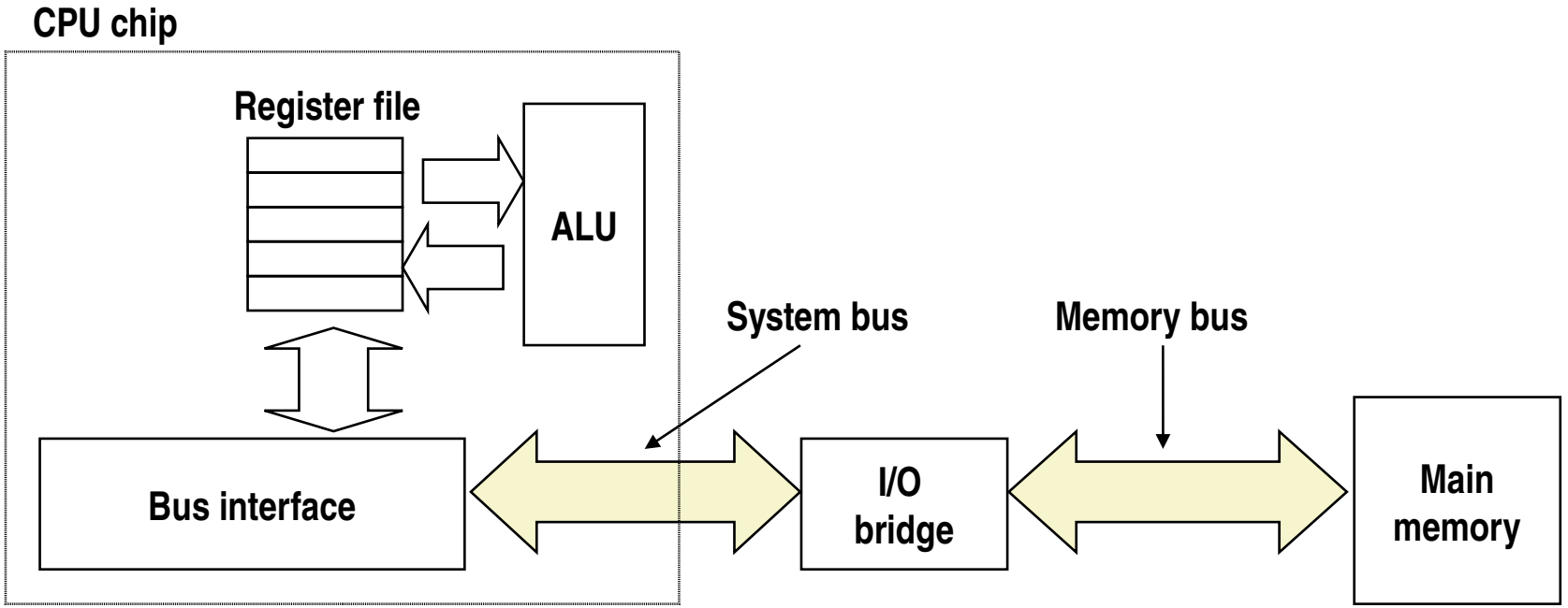
Interface entre CPU, memória e disco



Insper

Conectando a CPU e a RAM

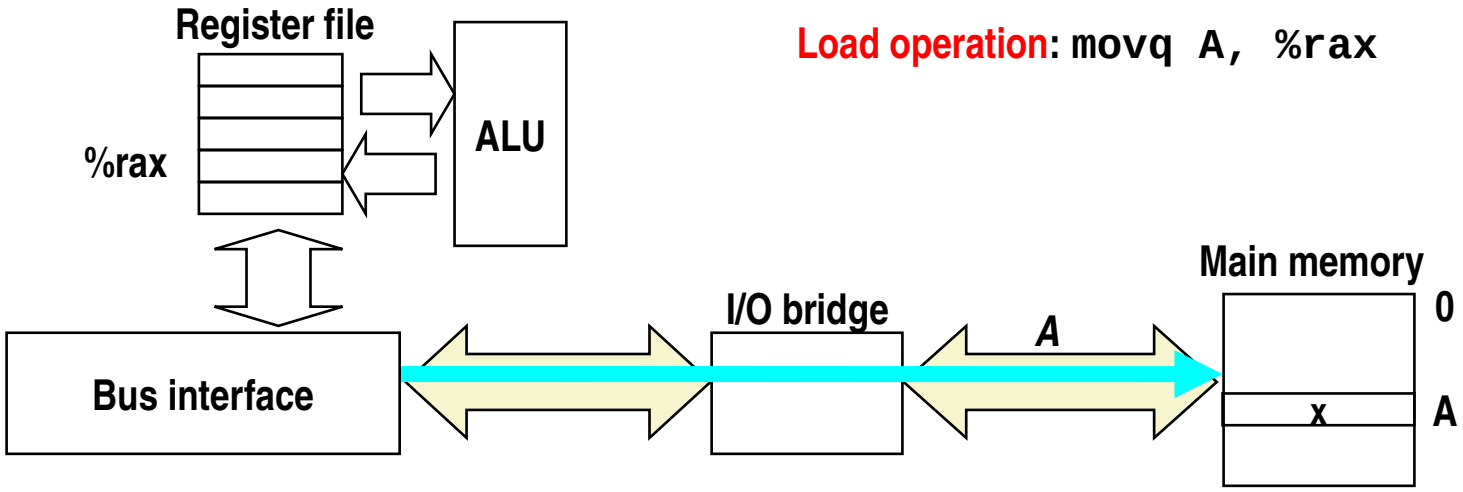
Uma *bus* é uma coleção de fios paralelos que carregam endereços, dados, e sinais de controle. *Buses* são normalmente compartilhadas entre vários dispositivos.



Insper

Leitura de dados

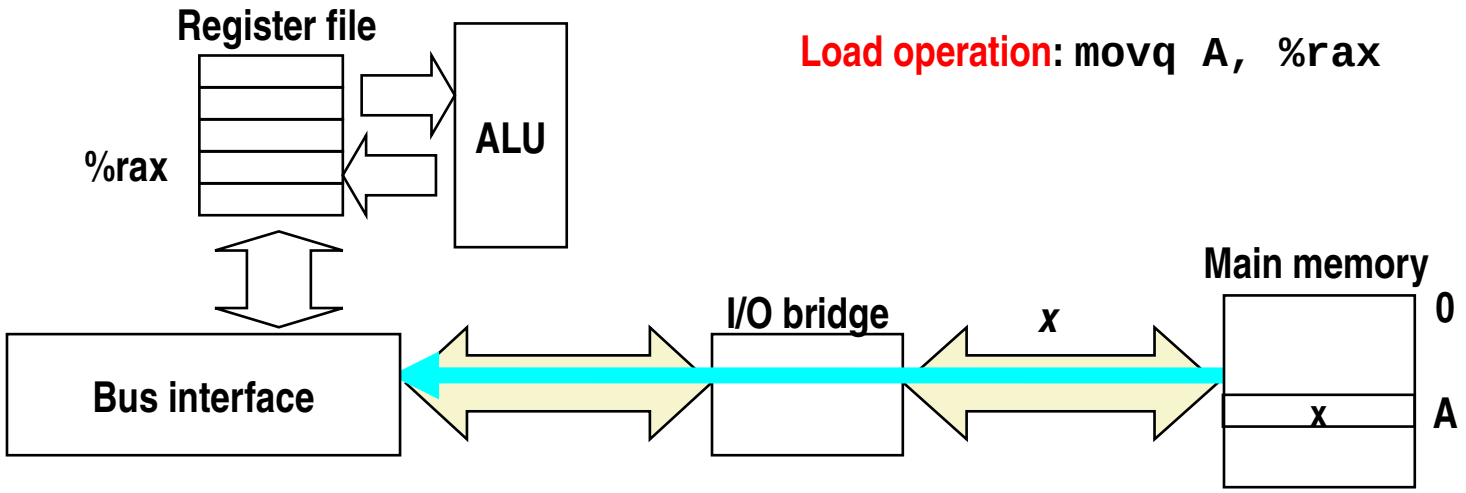
CPU coloca endereço **A** na bus de memória



Insper

Leitura de dados (II)

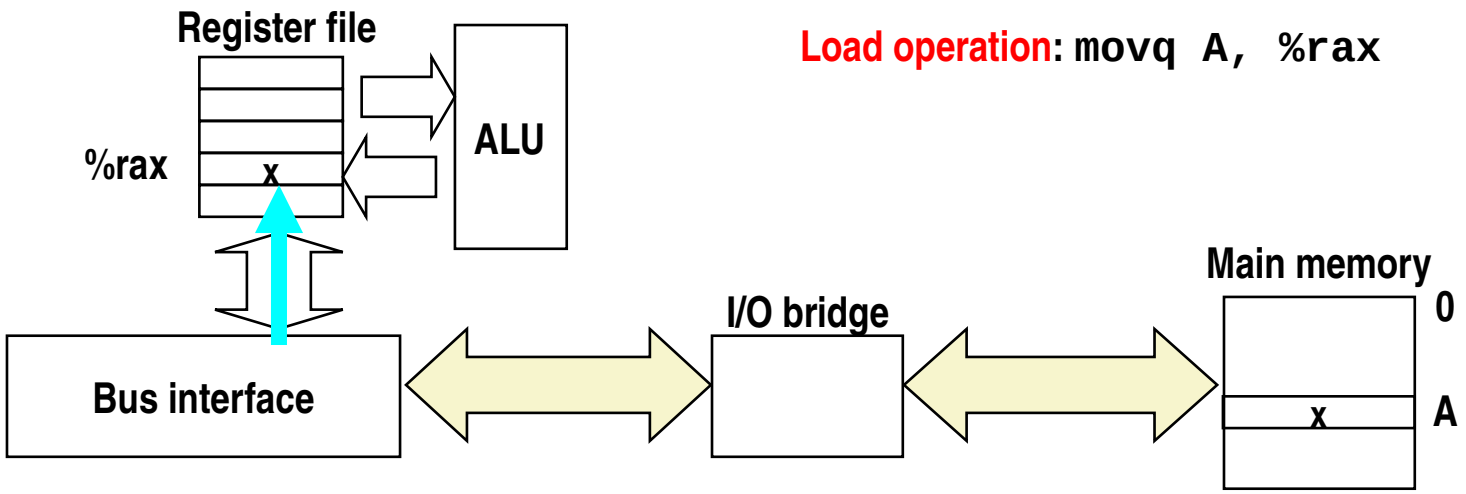
Memória principal acessa o endereço **A**, recuperando o valor **x**, e coloca este valor na bus



Insper

Leitura de dados (III)

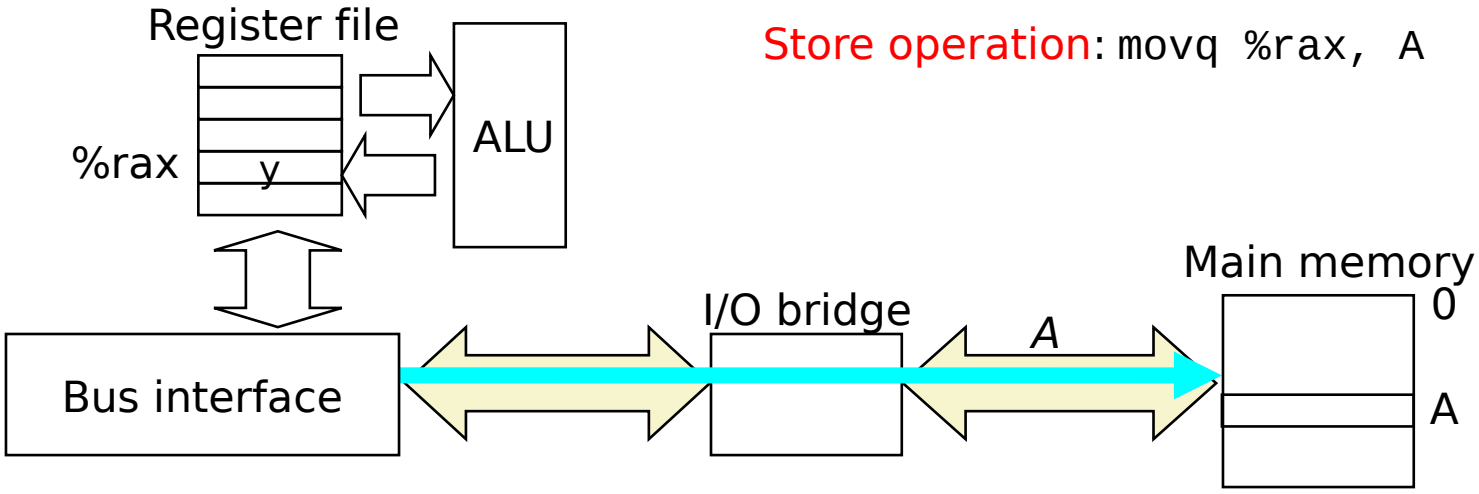
CPU lê o valor **x** da bus e coloca no registrador **%rax**



Insper

Escrita de dados

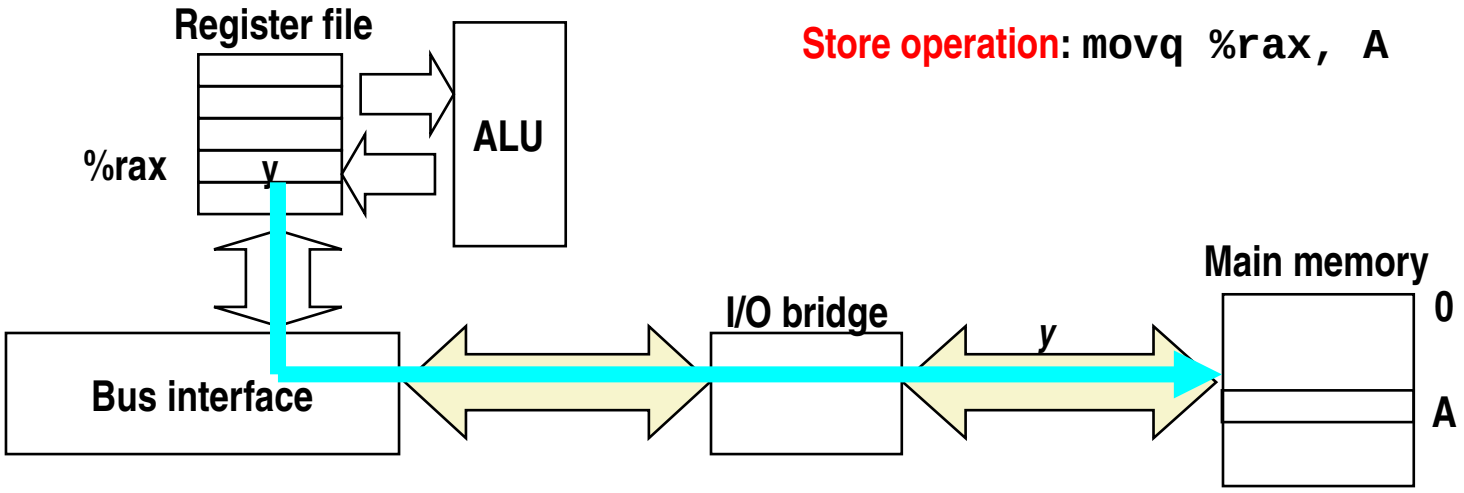
CPU escreve endereço **A** na bus. Memória principal guarda **A** em um buffer local e aguarda a palavra de dados correspondente.



Insper

Escrita de dados (II)

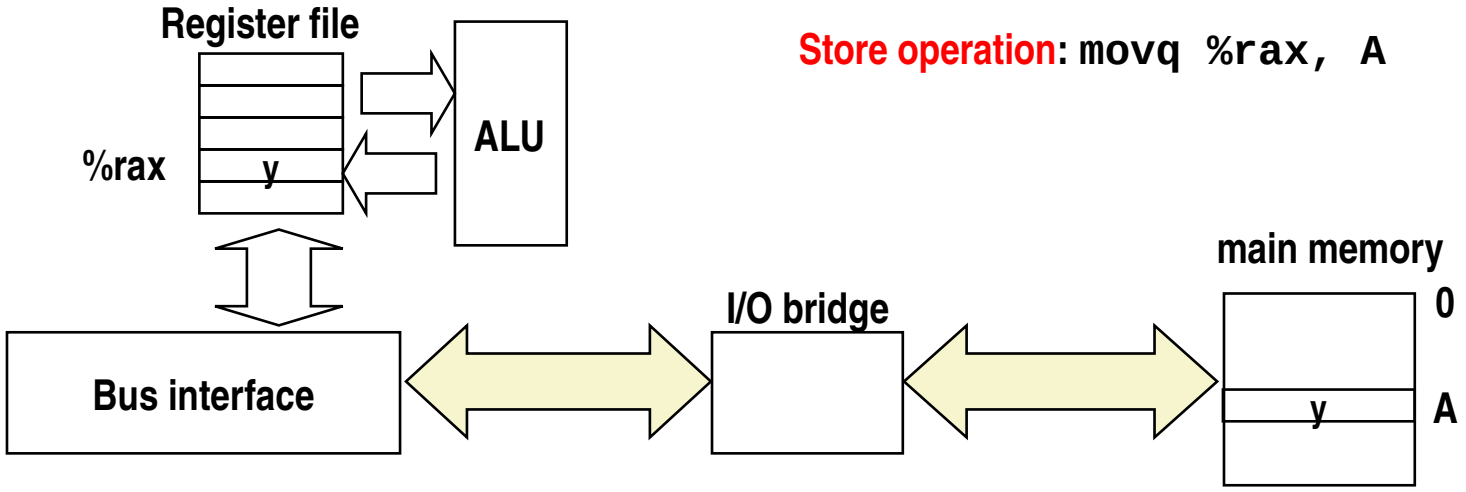
CPU coloca dado **y** na bus



Insper

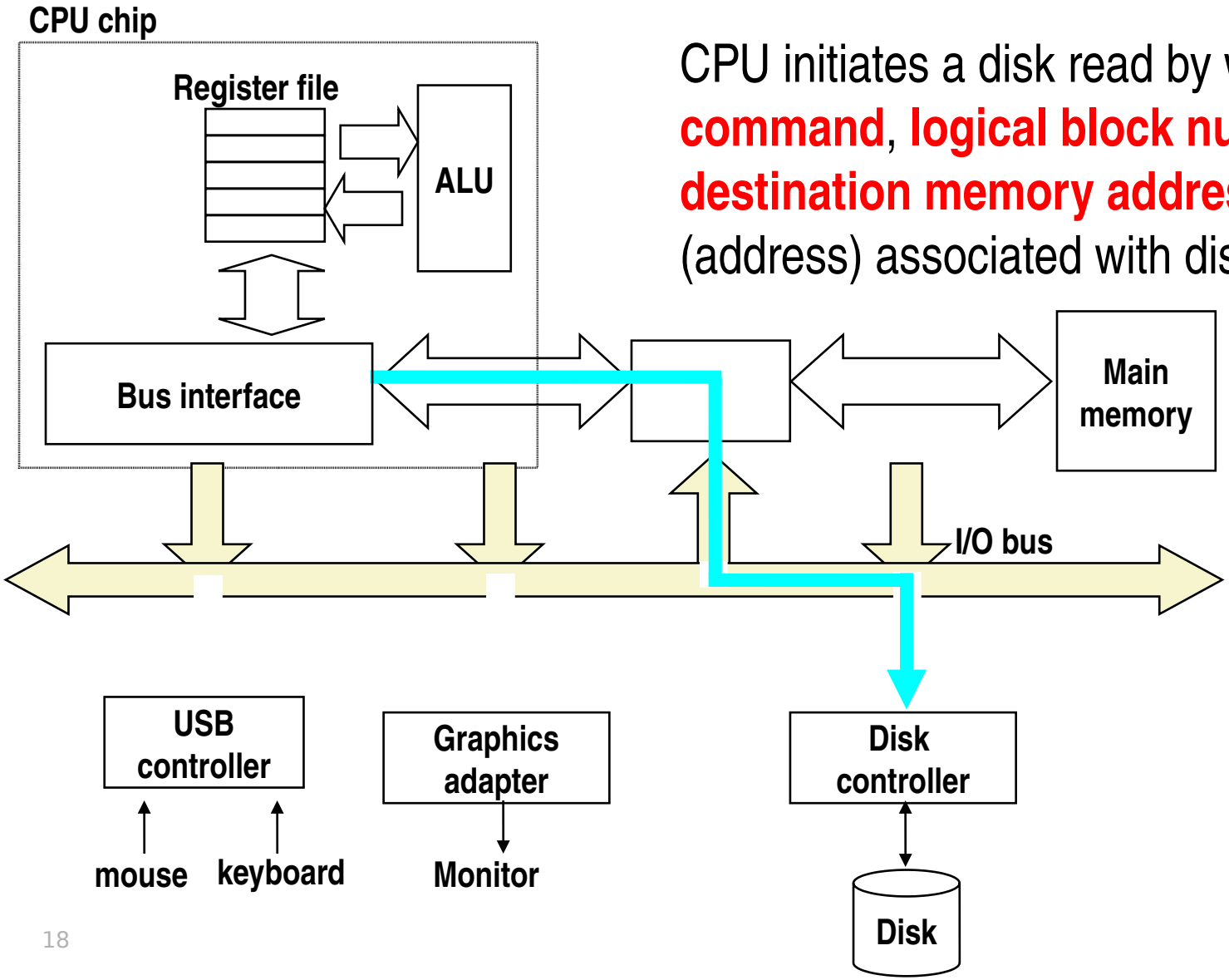
Escrita de dados (III)

Memória principal lê **y** da bus e guarda no endereço **A**



Insper

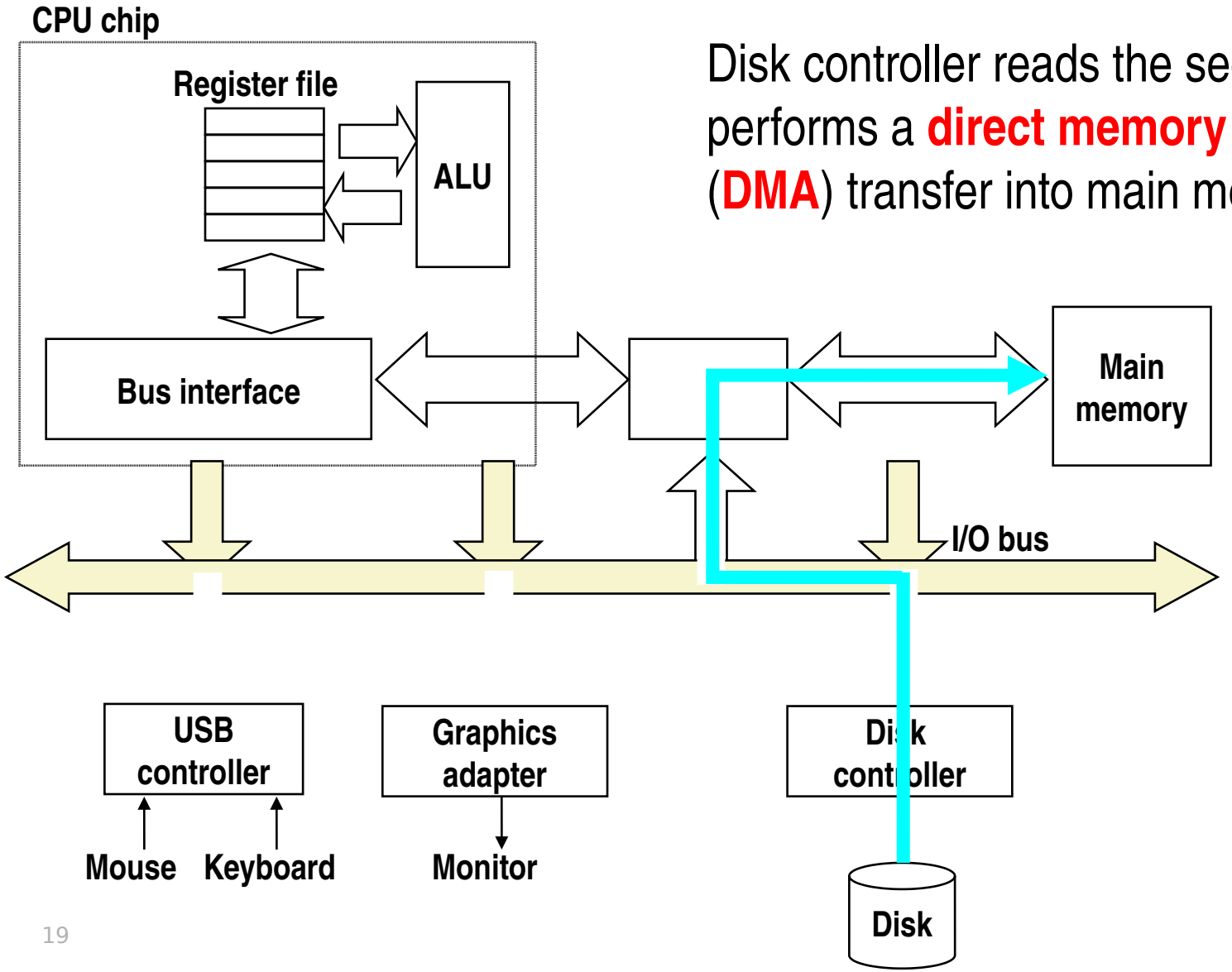
Lendo dados do disco



CPU initiates a disk read by writing a **command**, **logical block number**, and **destination memory address** to a port (address) associated with disk controller.

Insper

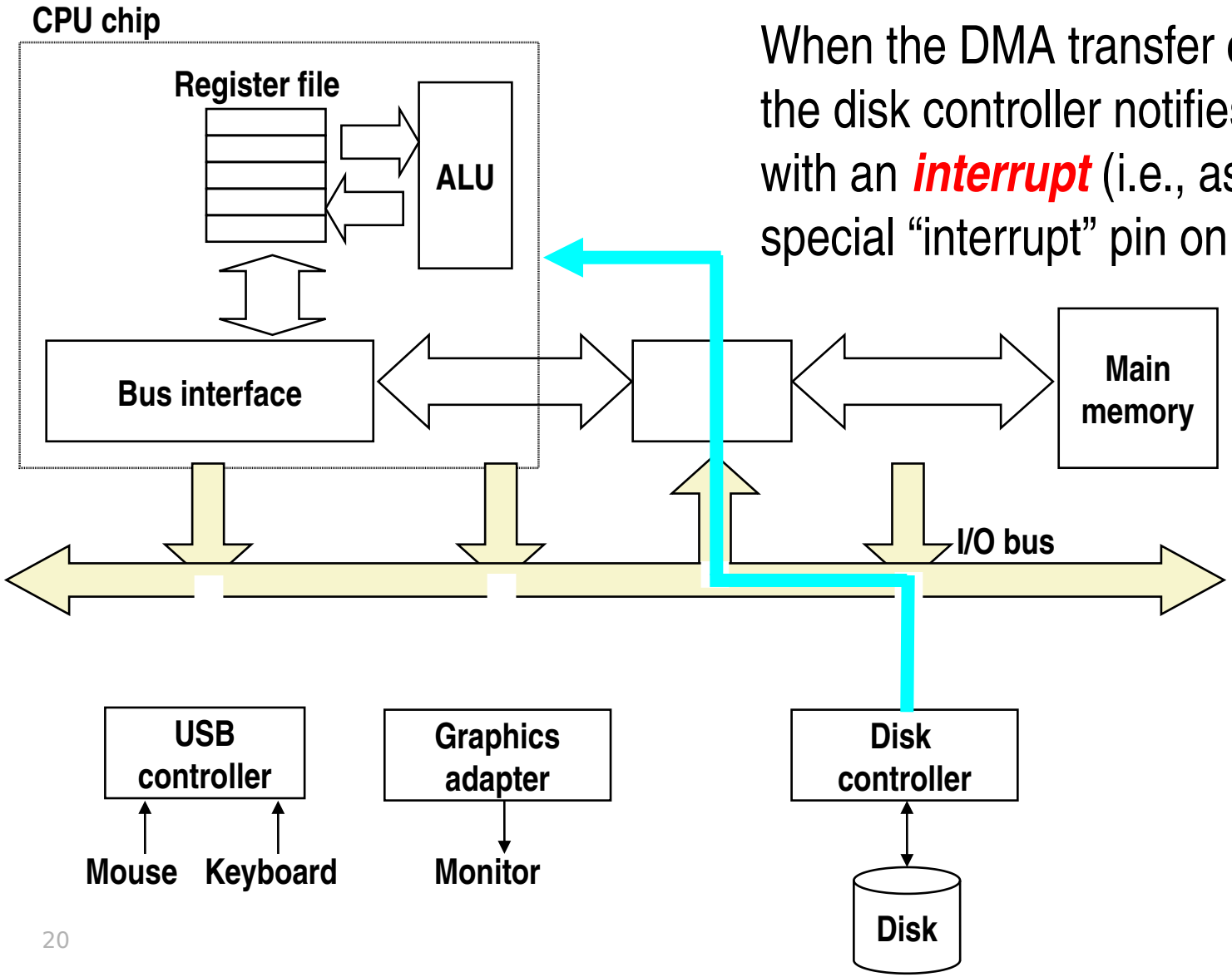
Lendo dados do disco (II)



Disk controller reads the sector and performs a **direct memory access** (**DMA**) transfer into main memory.

Insper

Lendo dados do disco (III)



When the DMA transfer completes, the disk controller notifies the CPU with an ***interrupt*** (i.e., asserts a special “interrupt” pin on the CPU)

Insper

Agenda

Tecnologia de memória e armazenamento

Hierarquia de memória

Memória Cache

Armazenamento de dados

1. Static RAM
2. Dynamic RAM
3. Read Only Memory – (SSDs)
4. Disco rígido mecânico

Random Access Memory (RAM)

- Static RAM (SRAM)
  - Feita de células bi-estáveis (e.g. flip-flops)
  - Rápida
  - Cara

  - Usos: memória cache

- Dynamic RAM (DRAM)
  - Pequenos capacitores
  - Barata
  - Mais lenta

  - Usos: memória principal, memória de vídeo

SRAM versus DRAM

	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1X	No	Maybe	100x	Cache memories
DRAM	1	10X	Yes	Yes	1X	Main memories, frame buffers

Read-Only Memory (ROM)

- SRAM e DRAM são voláteis: os dados são perdidos com o desligamento da máquina.

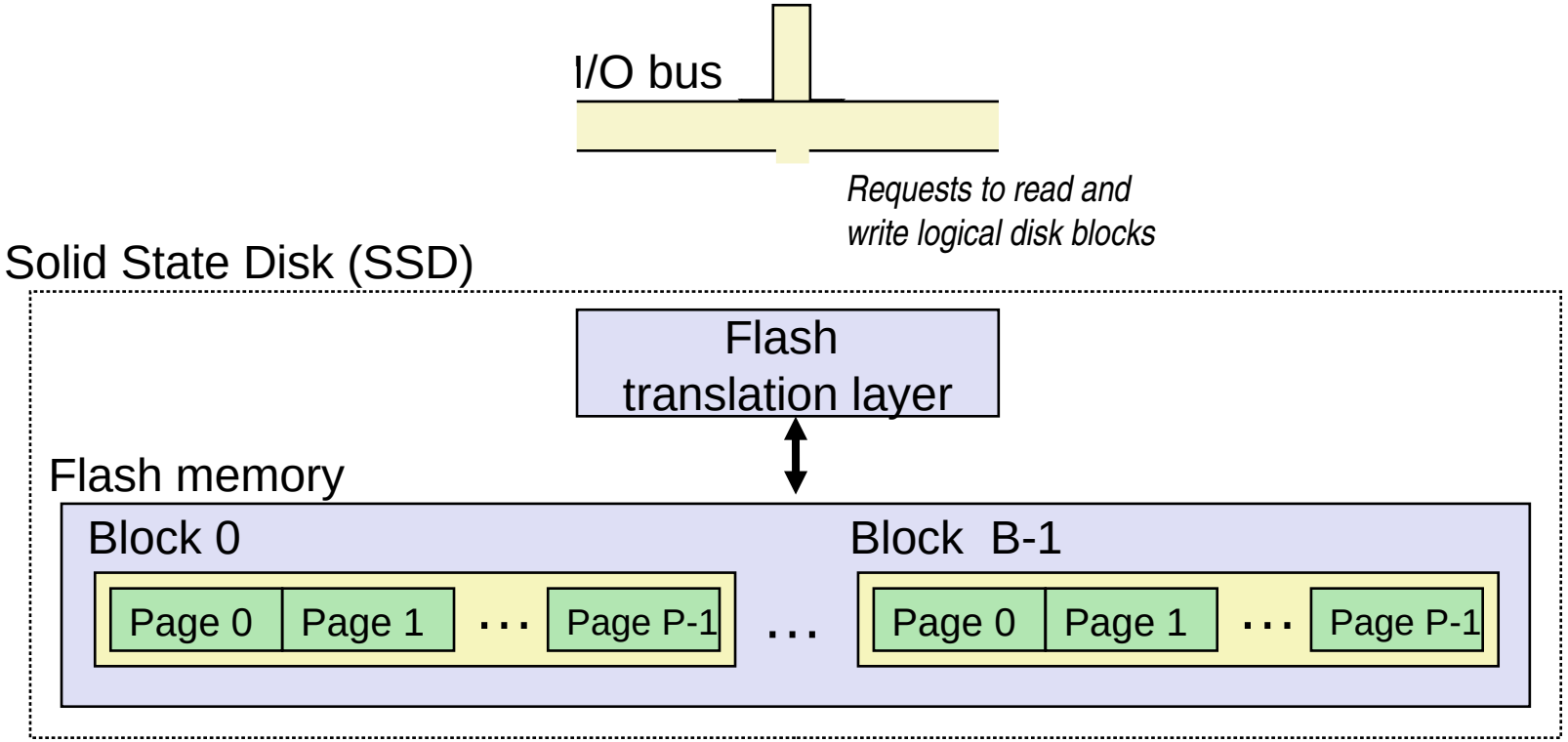
- Memórias não voláteis mantém o armazenamento mesmo com a máquina desligada
    - Read-only memory (ROM): programada em produção
    - Programmable ROM (PROM): Pode ser programada apenas uma vez
    - Eraseable PROM (EPROM): Pode ser apagada com equipamento especial (UV, X-Ray)
    - Electrically eraseable PROM (EEPROM): pode ser apagada eletronicamente
    - Memória Flash: EEPROMs com capacidade de apagamento parcial (por blocos)
        - “Gasta” depois de cerca de 100 000 apagamentos

Usos da ROM

- Firmware
  - BIOS
  - Controladoras de disco, placas de rede, GPUs, etc
- Solid-state disks (SSDs)
- Flash memory

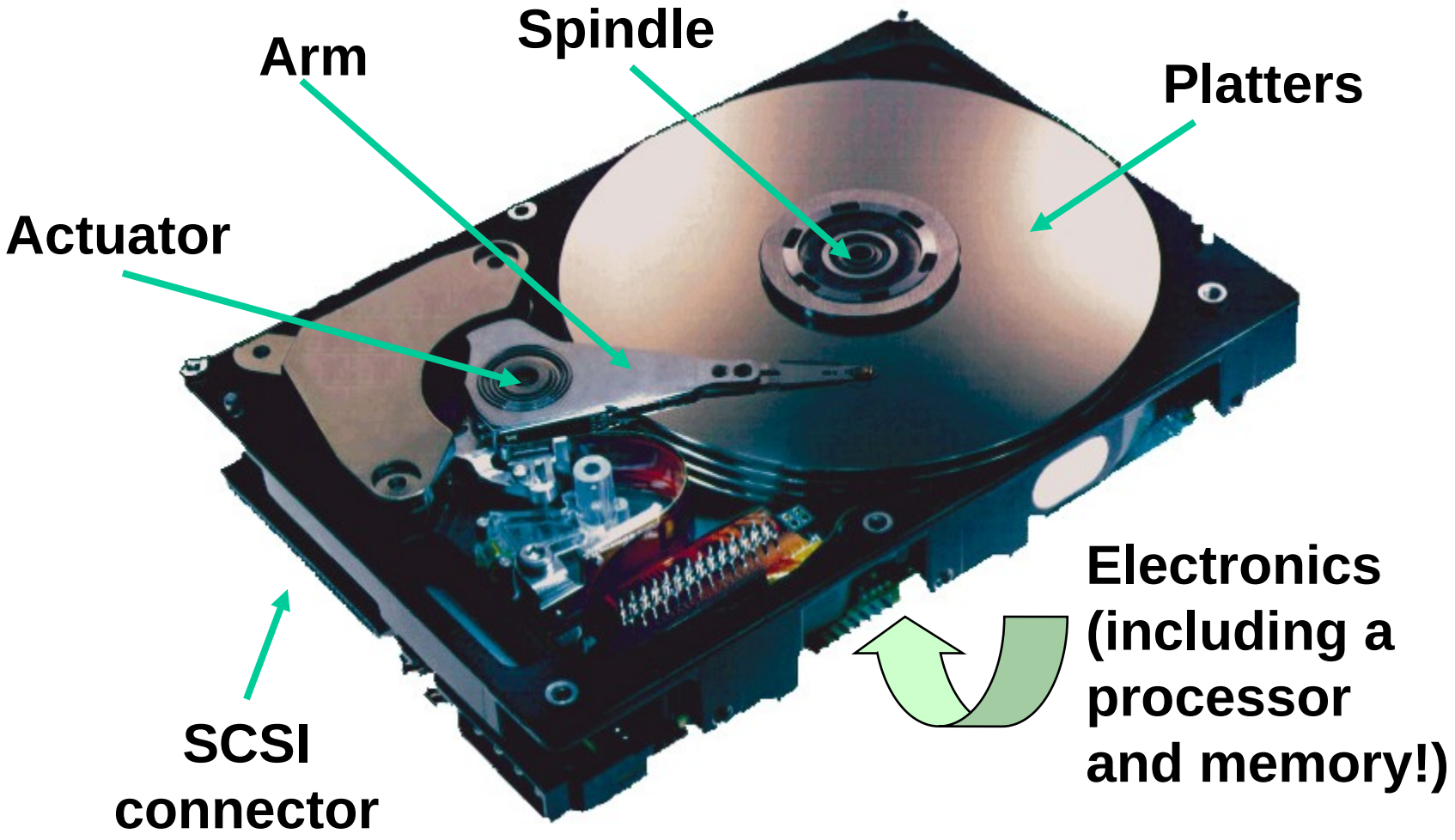
Insper

Solid State Disks (SSD)



- Páginas (*pages*): 4KB a 512KB. Blocos (*blocks*): 32 a 128 páginas
- Dados são lidos e escritos em unidades de páginas
- Página só pode ser escrita quando o bloco é apagado
- Um bloco desgasta depois de cerca de 100 000 escritas

Discos


Arm
Spindle
Platters
Actuator
Electronics (including a processor and memory!)
SCSI connector

Performance SSD vs HDD

Intel SSD 730 (480Gb)

Sequential read tput	**517 MB/s**	**Sequential write tput**	**476 MB/s**
Random (4k) read tput	**32 MB/s**	**Random (4k) write tput**	**105 MB/s**

Source: http://ssd.userbenchmark.com/Intel-730-Series-480GB/Rating/3479

Barracuda 1Tb, 7200rpm

Sequential read tput	**160 MB/s**	**Sequential write tput**	**150 MB/s**
Random (4k) read tput	**0.66 MB/s**	**Random (4k) write tput**	**1.56 MB/s**

Source: http://hdd.userbenchmark.com/SpeedTest/2870/HGST-HTS545050A7E680

Performance SSD vs HDD

Intel SSD 730 (480Gb)

Sequential read tput	**517 MB/s**	**Sequential write tput**	**476 MB/s**
Random (4k) read tput	**32 MB/s**	**Random (4k) write tput**	**105 MB/s**

Source: **http://ssd.userbenchmark.com/Intel-730-Series-480GB/Rating/3479**

Meu HDD (Hitachi, 5400rpm, 500Gb)

Sequential read tput	**93.9 MB/s**	**Sequential write tput**	**82.4 MB/s**
Random (4k) read tput	**0.21 MB/s**	**Random (4k) write tput**	**1.15 MB/s**

Source: **http://hdd.userbenchmark.com/SpeedTest/2870/HGST-HTS545050A7E680**

Evolução das memórias

SRAM

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/MB	2,900	320	256	100	75	60	*25*	*116*
access (ns)	150	35	15	3	2	1.5	*1.3*	*115*

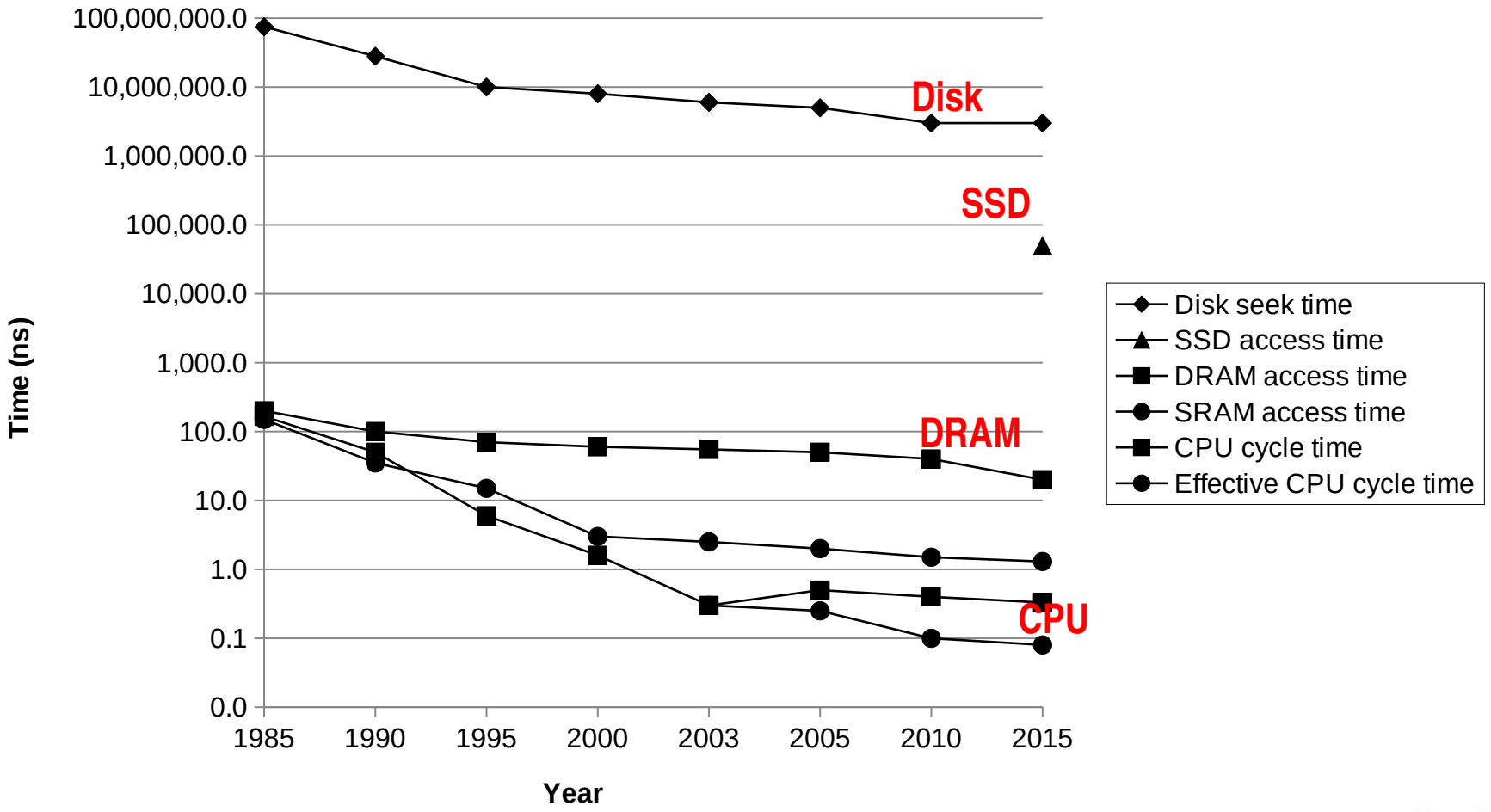
DRAM

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/MB	880	100	30	1	0.1	0.06	0.02	*44,000*
access (ns)	200	100	70	60	50	40	20	*10*
size (MB)	0.256	4	16	64	2,000	8,000	16.000	*62,500*

Disk

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/GB	100,000	8,000	300	10	5	0.3	0.03	*3,333,333*
access (ms)	75	28	10	8	*5*	*3*	*3*	*25*
size (GB)	0.01	0.16	1	20	160	1,500	3,000	*300,000*

Comparação entre a evolução da CPU, memória e discos



Insper

Bom, bonito e barato!

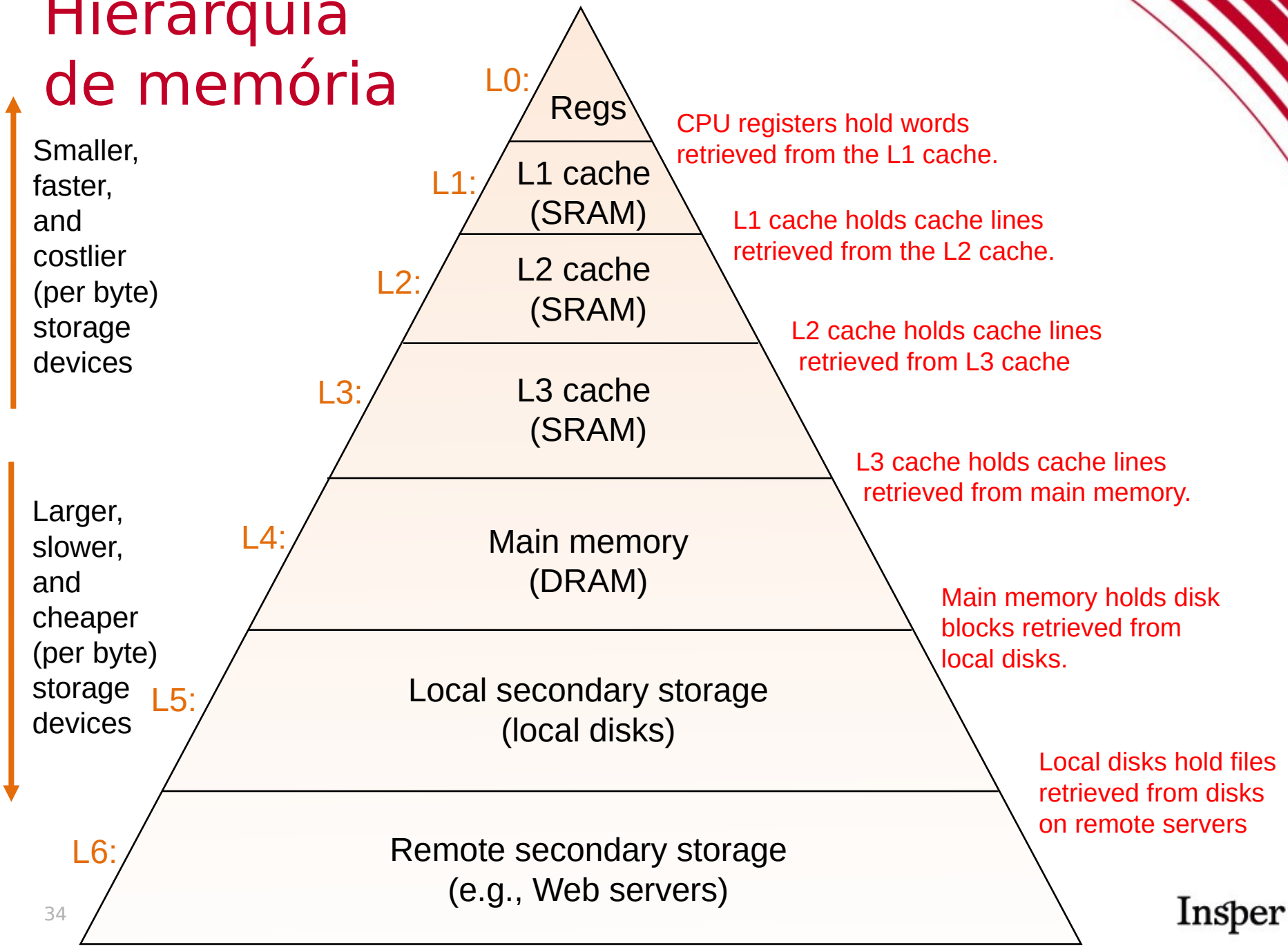
Como ter alto desempenho sem pagar tão caro?

- Um sistema teórico de alto desempenho seria feito inteiramente de memória cache SRAM e armazenamento em SSD!

- Um sistema teórico de baixo custo seria feito de memória DRAM, sem cache, e com discos rígidos mecânicos!

- Nenhum dos dois extremos presta!

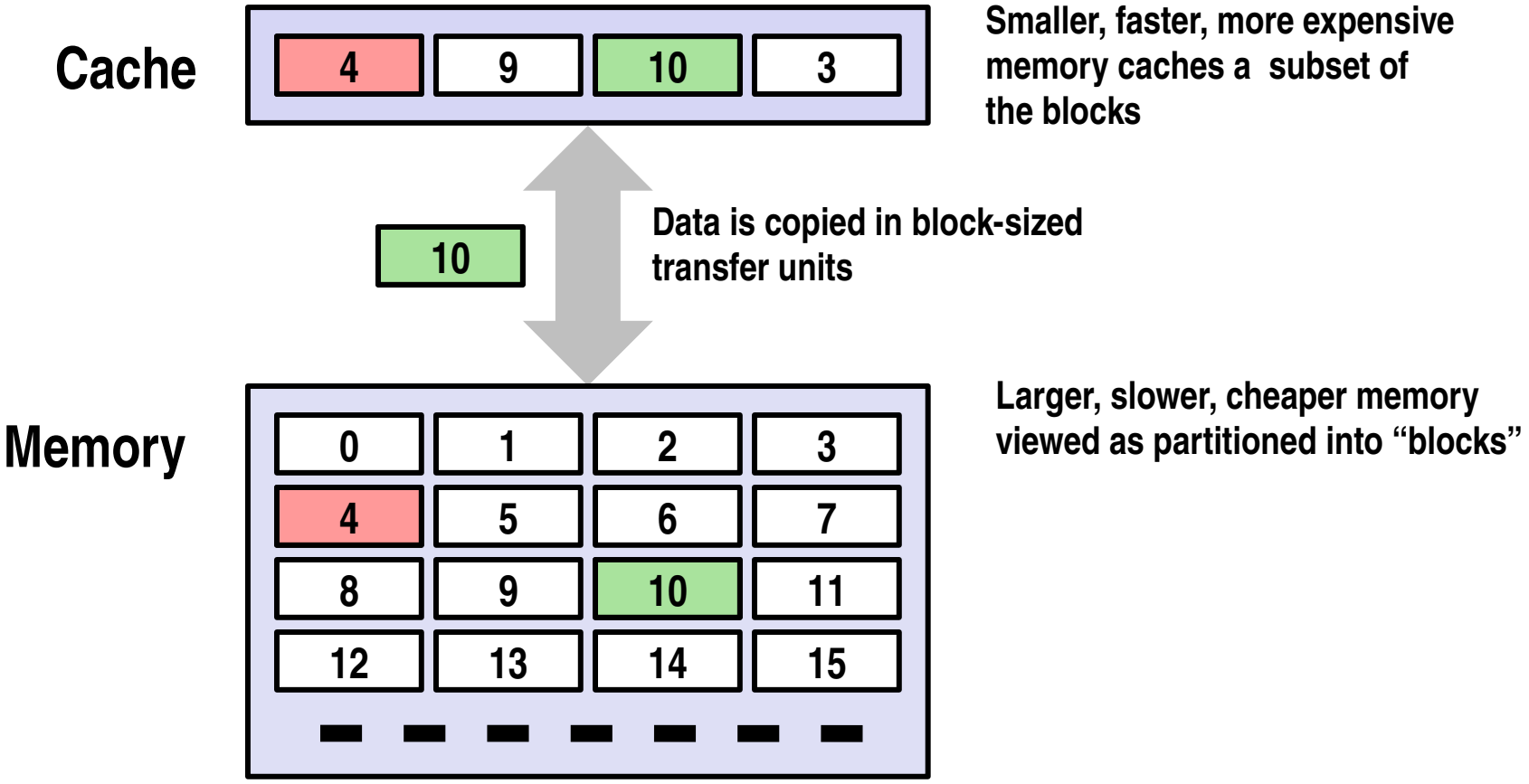
Hierarquia de memória



Insper

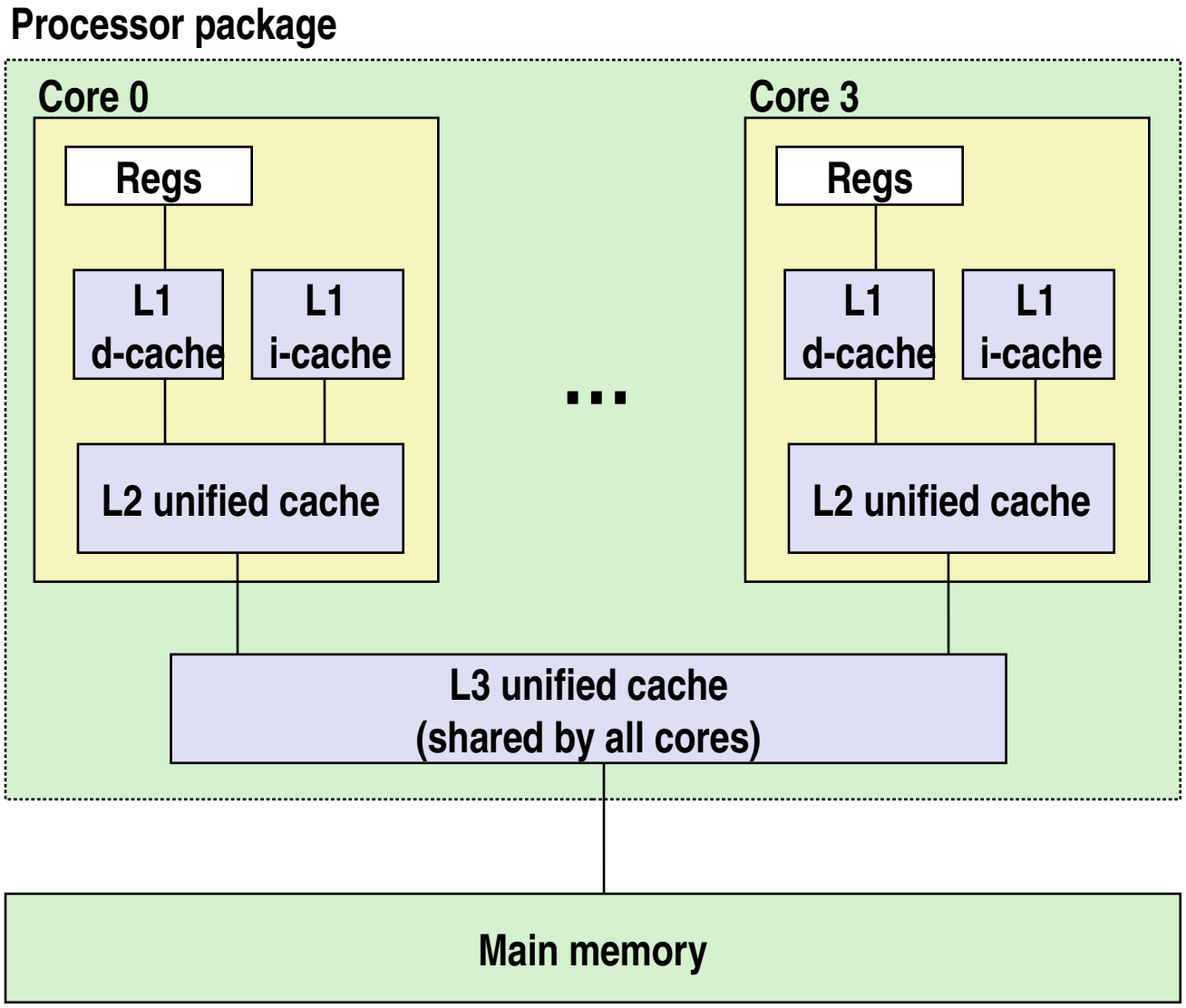
Cache

Guarda os itens mais recentes no cache, para reuso mais rápido.



Insper

Arquitetura Intel i7



L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 10 cycles

L3 unified cache:
8 MB, 16-way,
Access: 40-75 cycles

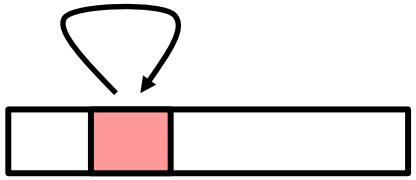
Block size: 64 bytes for all caches

Localidade

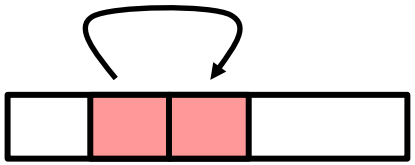
A solução deste dilema é observar que os programas de computador tendem a apresentar **localidade**:

Programas tendem a usar dados e instruções cujos endereços são os mesmos ou próximos àqueles usados recentemente

- Localidade temporal:
  Reutilizar o mesmo endereço



- Localidade espacial:
  Utilizar endereços adjacentes



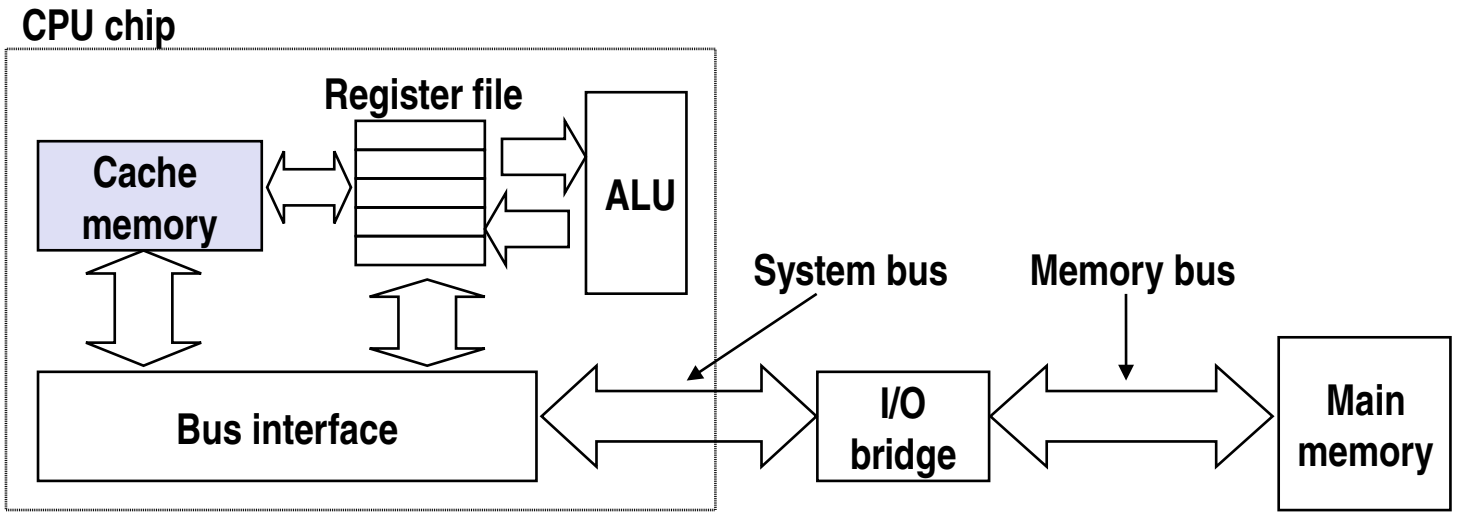
Tipos de cache miss

- ***Cold miss***: O cache estava vazio
- ***Capacity miss***: O *working set* (o conjunto de dados a ser trabalhado) é maior que o cache
- ***Conflict miss***: Apesar de o cache não estar cheio, dois blocos são alocados para exatamente a mesma posição do cache

Arquitetura típica

Cache:

- Mantém blocos de memória acessados frequentemente
- Memória rápida (SRAM), tamanho pequeno
- Gerenciada automaticamente por hardware
- CPU busca dados primeiramente no cache



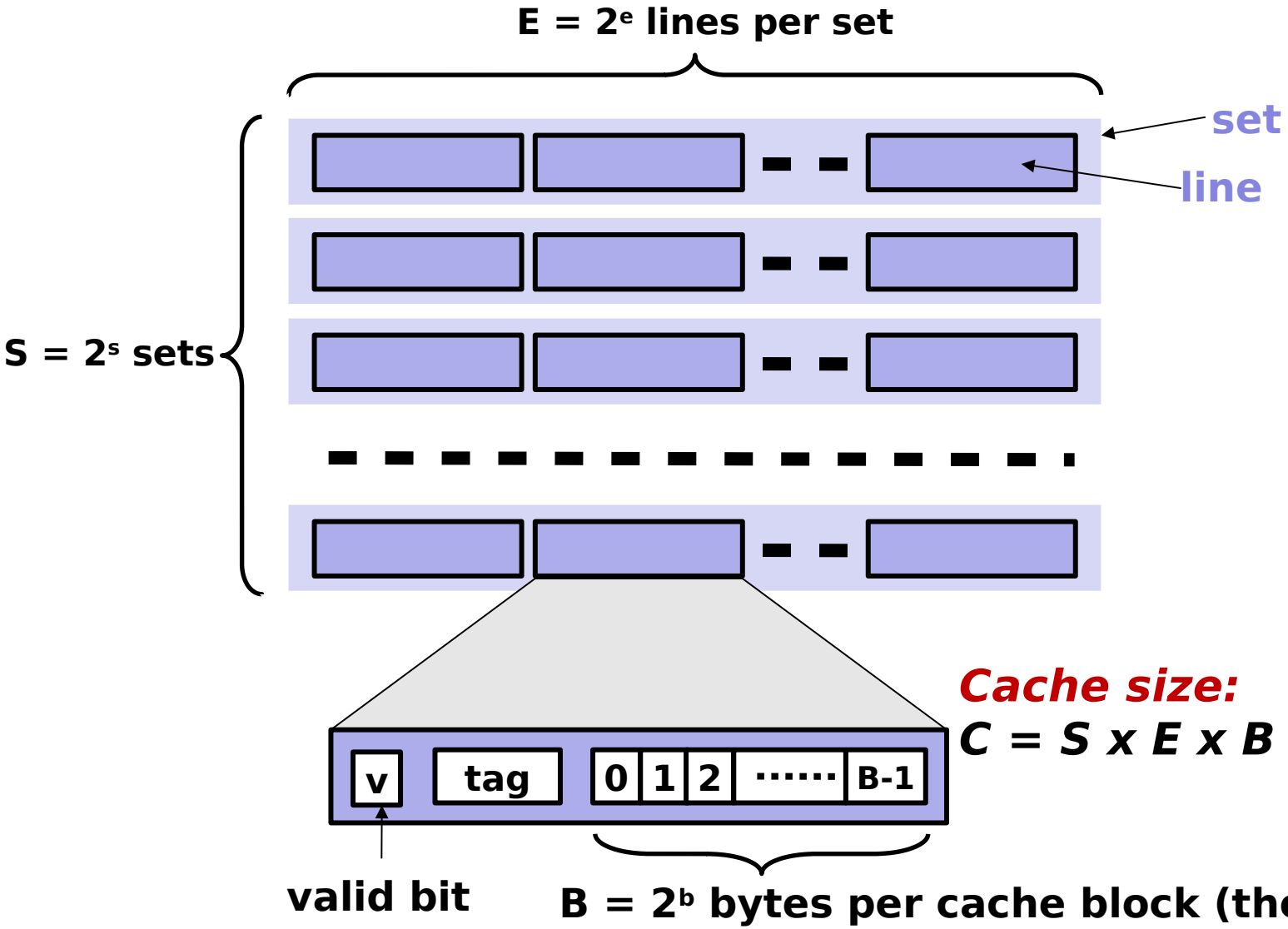
Insper

Organização de um cache

O cache é organizado em conjuntos de linhas, cada linha contendo vários bytes

- Conjuntos: o cache é feito de **S** conjuntos (*sets*)
- Linhas: cada conjunto contém **E** linhas
- Bytes: cada linha contém **B** bytes
- Logo, o tamanho do cache é **C = S x E x B** bytes

Organização do cache (S, E, B)



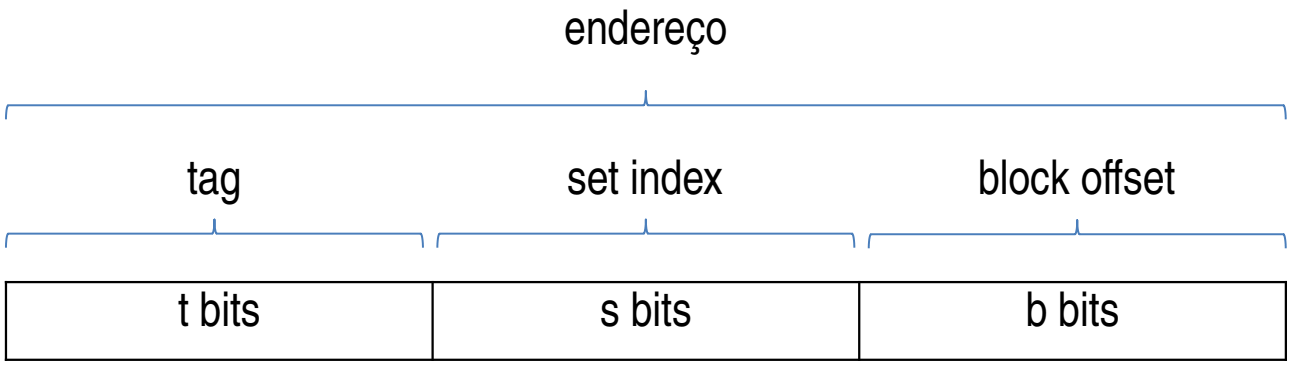
Cache size:

C = S x E x B data bytes

Insper

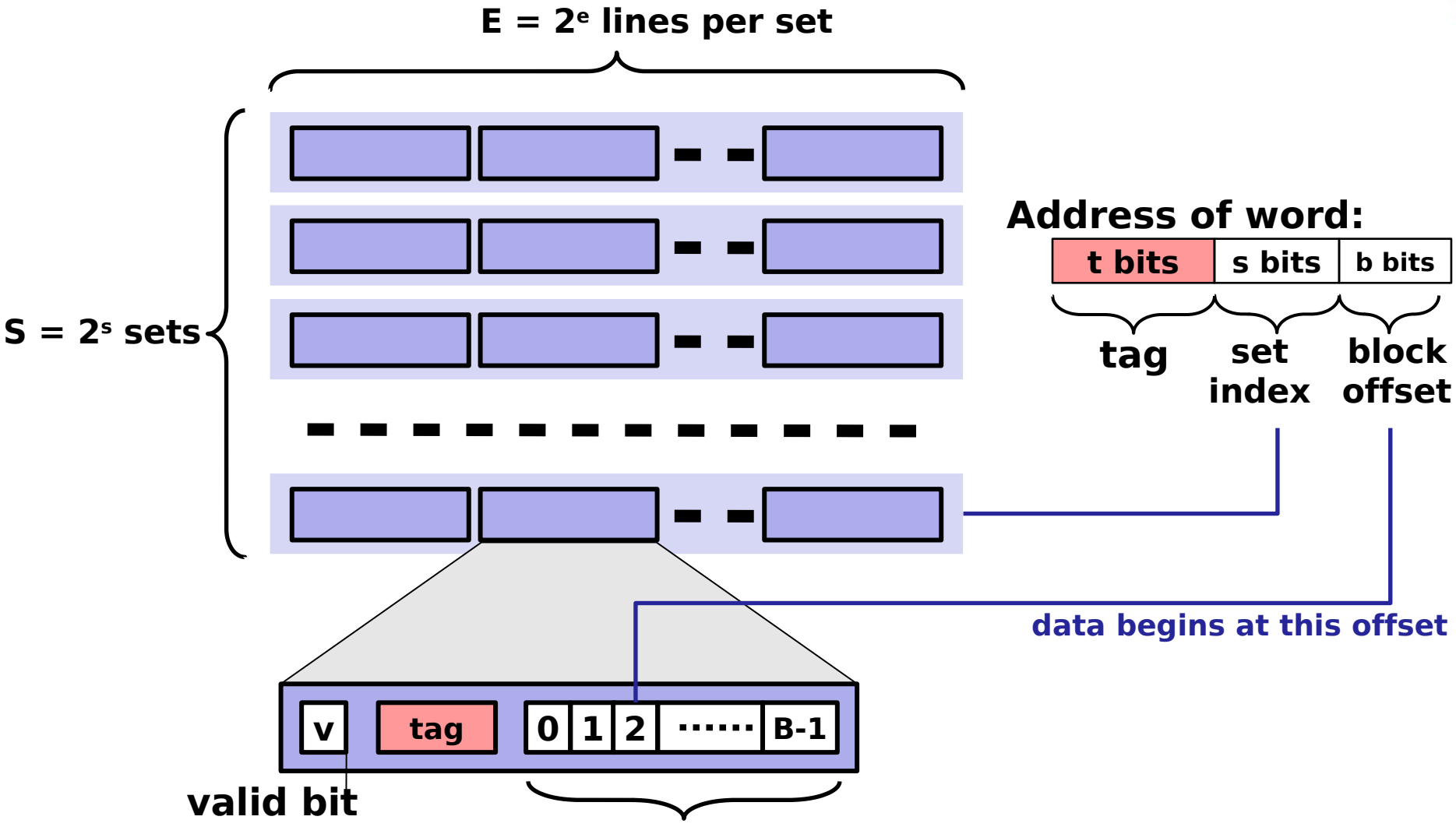
Cache: como usar?

- Tenha em mãos o **endereço de memória** que você quer acessar! Quebre o endereço em três partes:
  - **Tag**: Os primeiros ***t*** bits são o “código de barras” da linha de cache à qual este endereço pertence
  - **Set index**: Os ***s*** bits a seguir indicam o conjunto ao qual a linha de cache pertence
  - **Block offset**: Os ***b*** bits finais do endereço indicam a posição do endereço procurado dentro da linha de cache



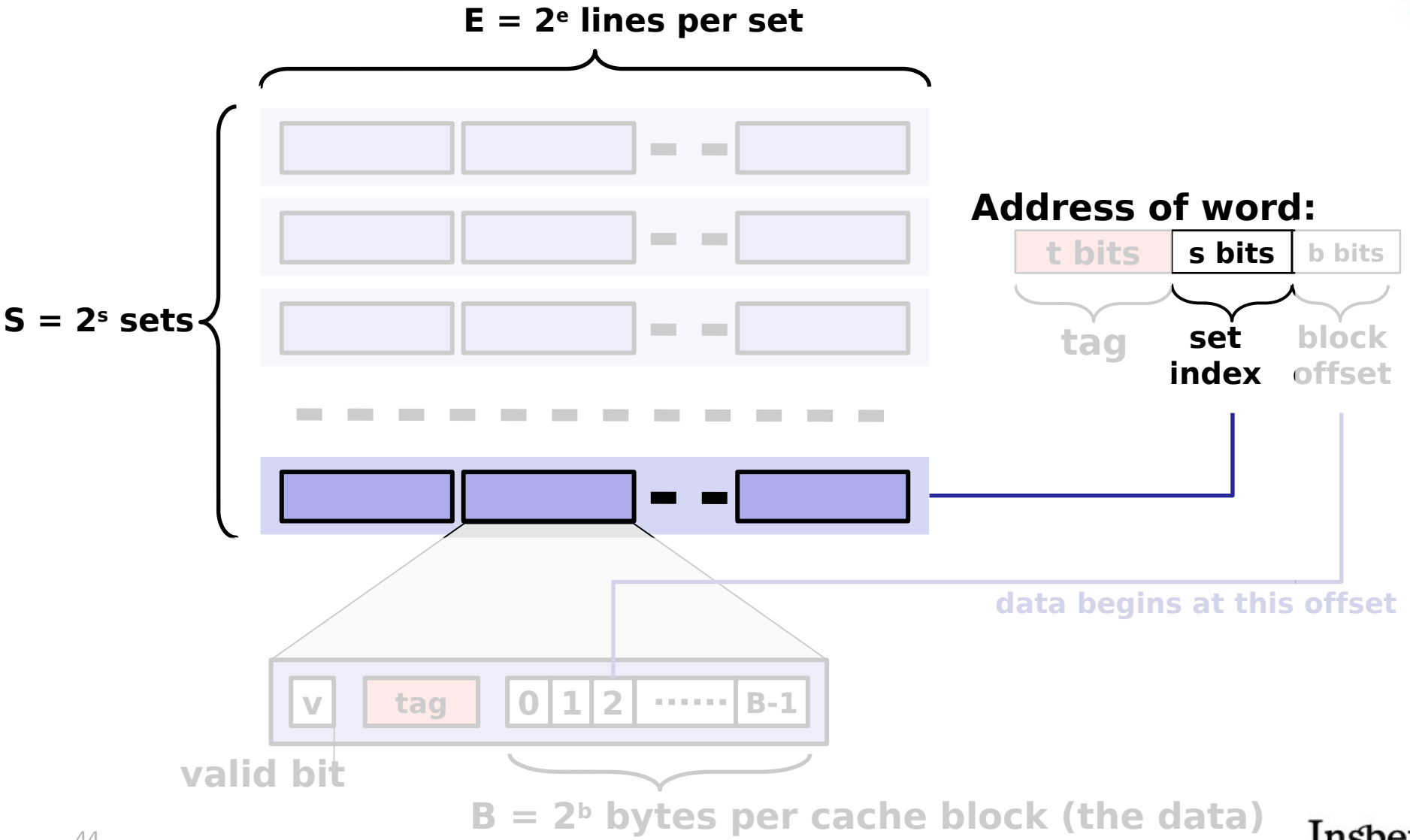
Insper

Cache: como usar?



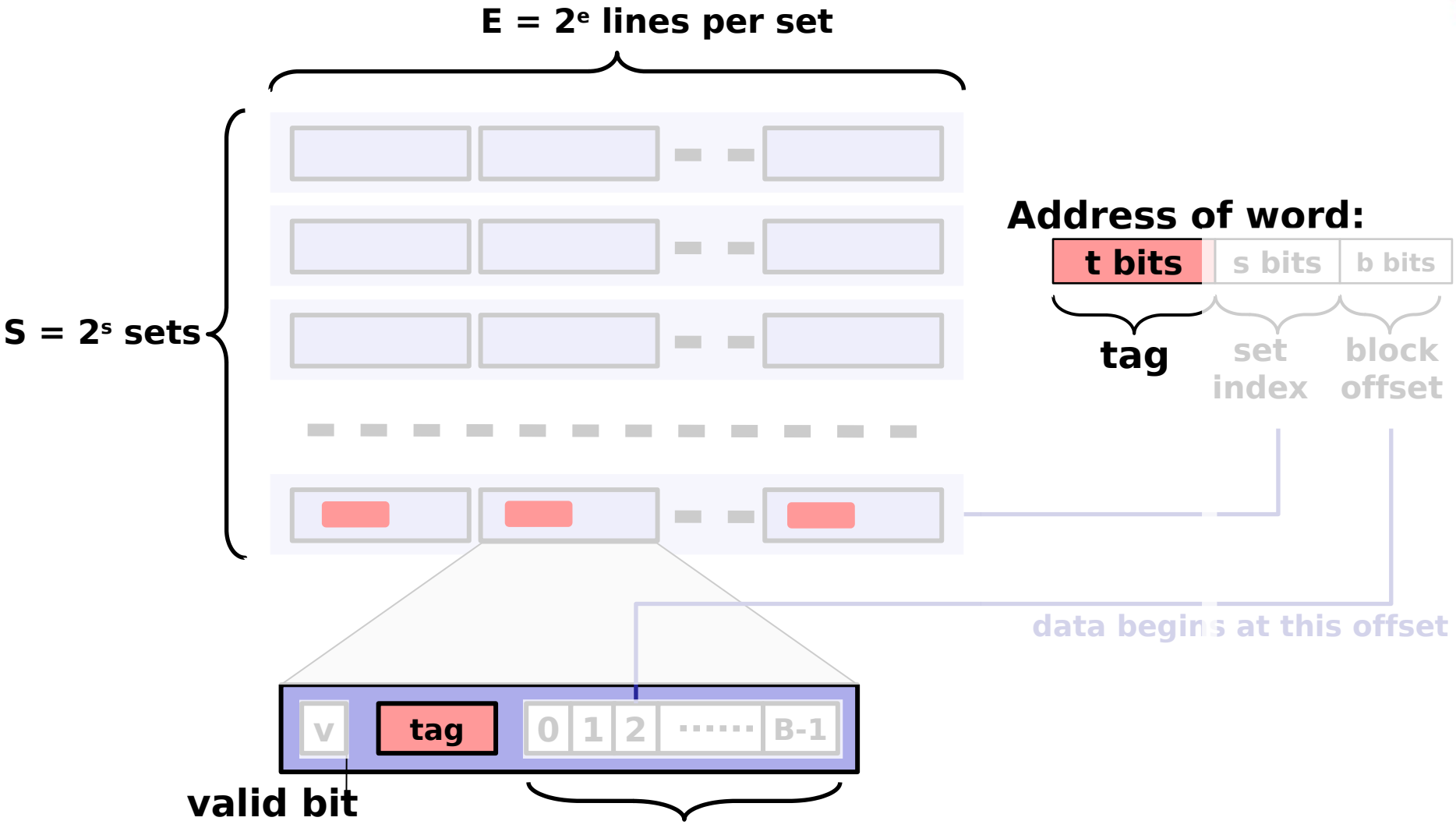
Insper

1. Localize o **set**



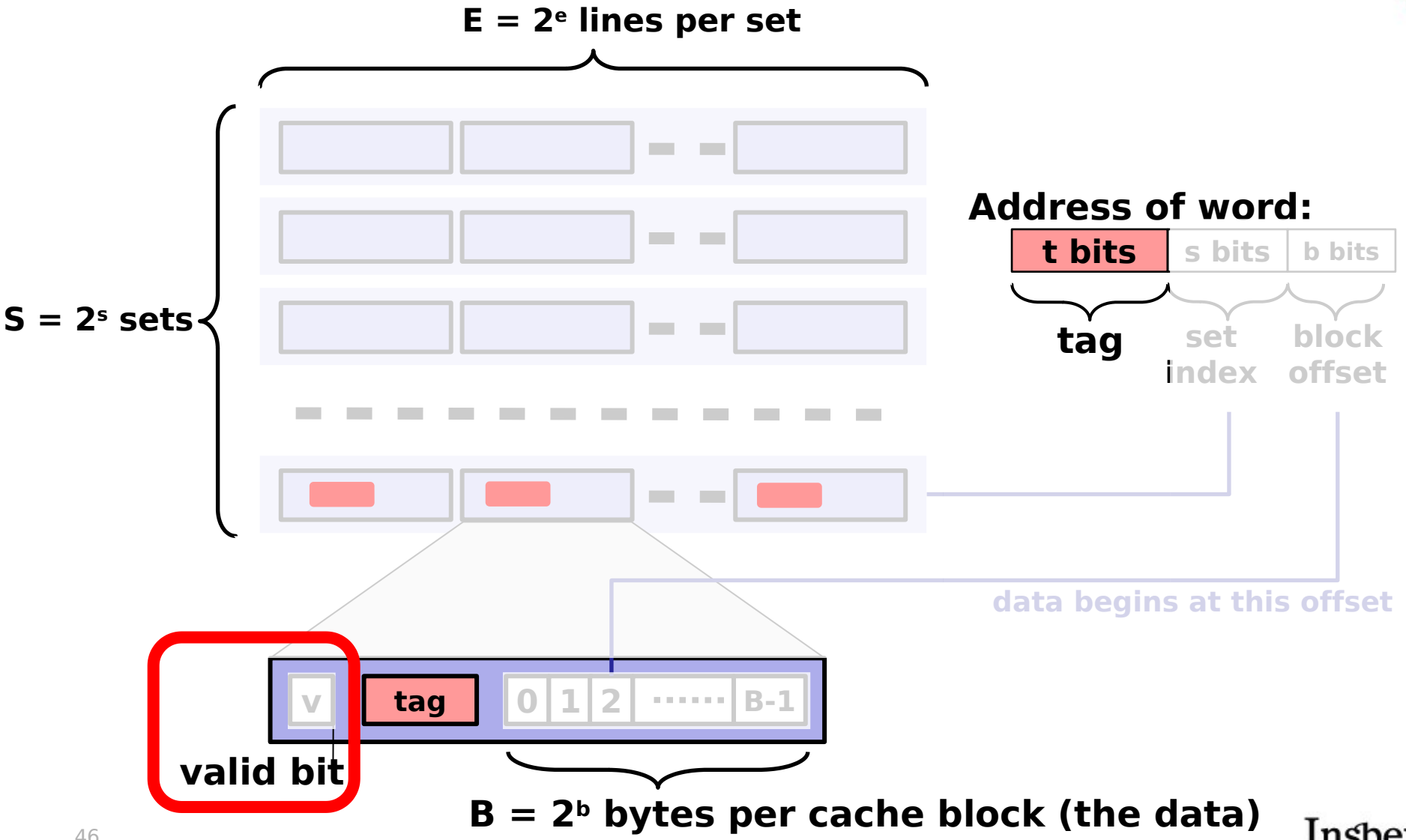
Insper

2. Dentro do set, veja se tem uma linha com o mesmo **tag**



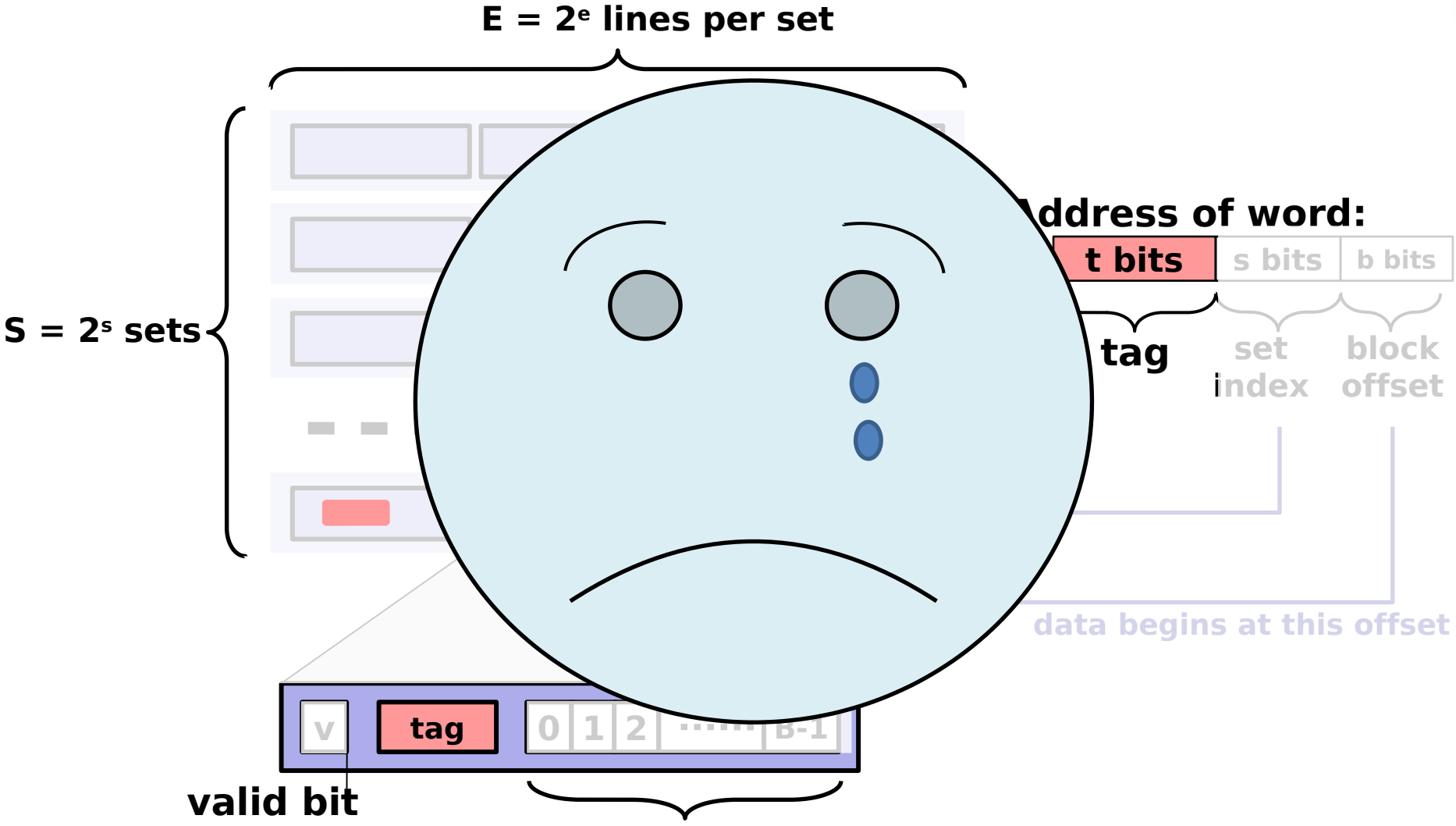
Insper

(Não se esqueça do bit de validade! Pode ser lixo de memória)



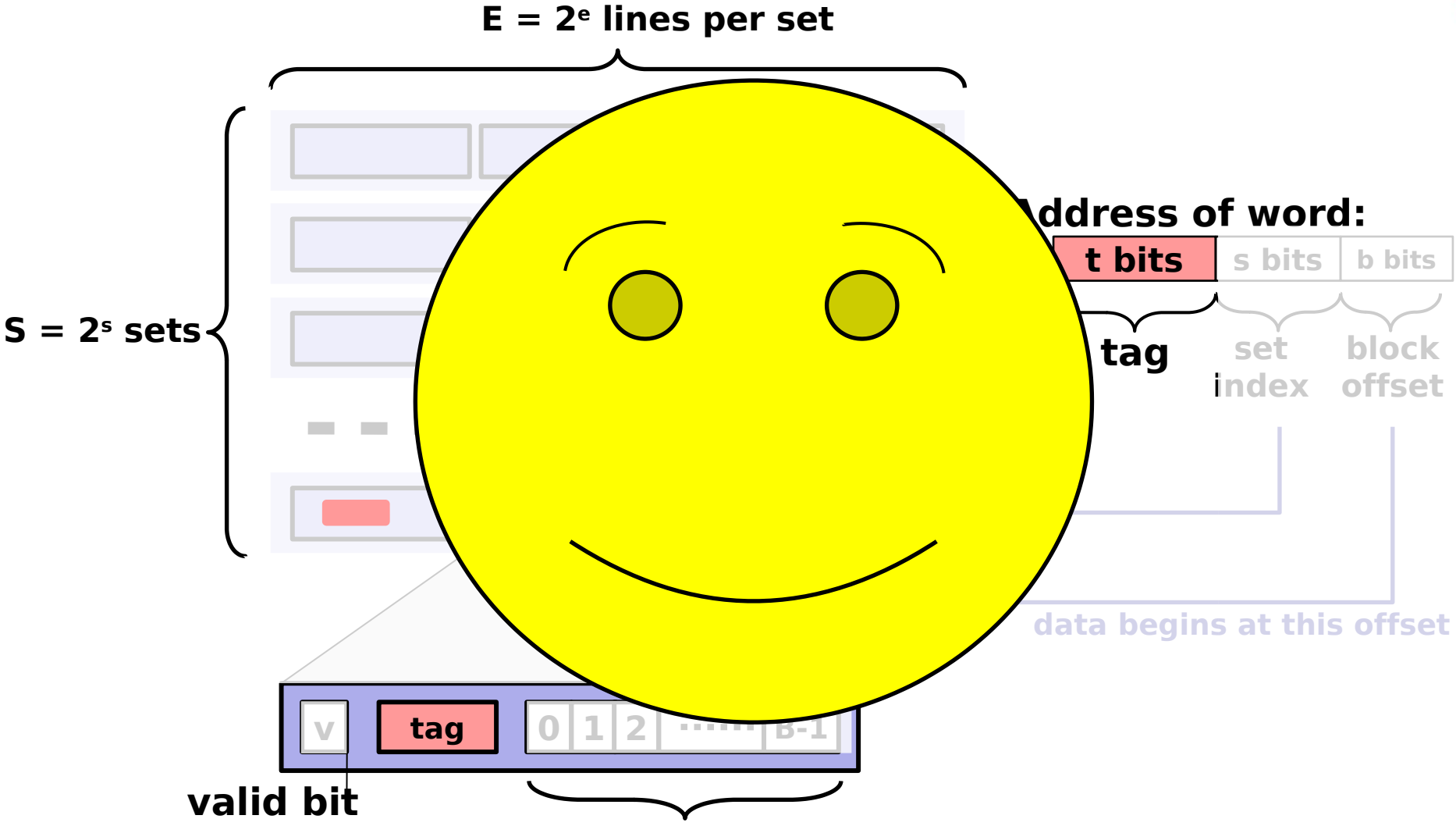
Insper

(Se **não achou** o tag: **cache miss**
Tem que buscar da RAM...)



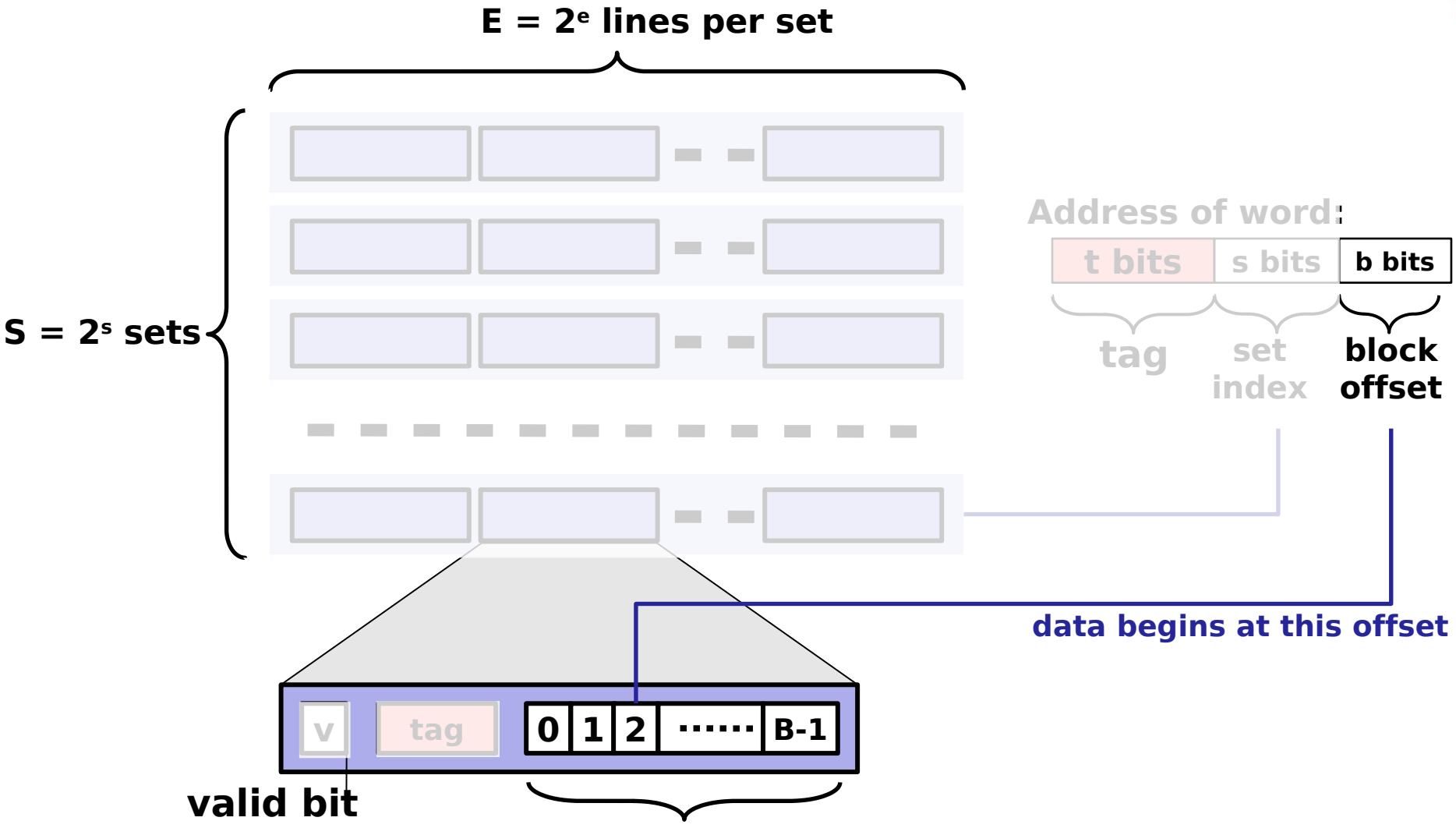
Insper

(Se **achou** o tag: **cache hit**
O dado já está no cache!)



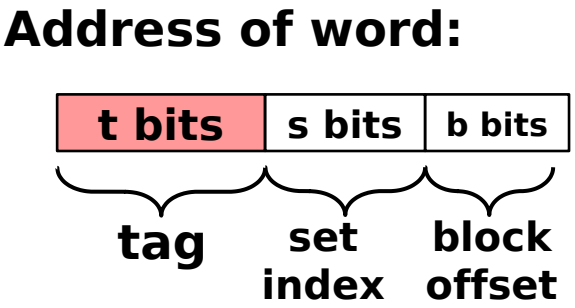
Insper

Se cache hit, use **block offset** para localizar os dados



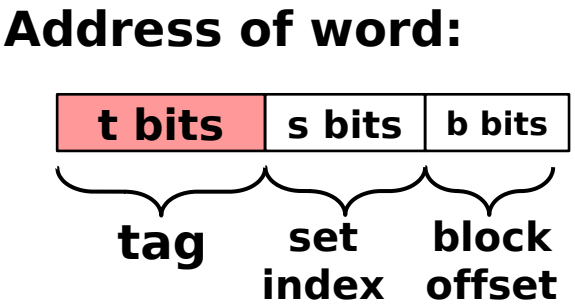
Insper

Se cache hit, use **block offset** para localizar os dados



Por que o tag está no começo?

Se cache hit, use **block offset** para localizar os dados



Por que o tag está no começo?

Evitar colisões entre endereços vizinhos!

Associatividade

Direct mapping: cada set tem apenas uma linha de cache (E = 1, S = C / B)

- Vantagem: procurar pelo tag no set é trivial
- Desvantagem: cache misses mesmo quando o cache está pouco utilizado

Fully associative: o cache inteiro é um único set (S = 1, E = C / B)

- Vantagem: uso máximo do cache
- Desvantagem: procurar pelo tag é muito demorado!

N-way associative: cada set tem E=N > 1 linhas

- Um equilíbrio entre direct mapping e fully associative
- Esta é a opção prática

Código cache-friendly

- Prioriza acessos de dados vizinhos
- Evita buscas desnecessárias a memória
- Reutiliza dados sempre que possível

Como medidos isto?

- Calculamos cache miss rate = cache miss / total de acessos
- Testamos com diferentes tamanhos de entrada
- Isto também pode ser feito usando o valgrind

Insper

www.insper.edu.br