Aula 06 - Memórias e Endereçamento 2/2

Prof. Sergio R. M. Canovas

PCS - Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Setembro, 2020

Agenda

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

1 Associação de Memórias e I/O

2 Memórias em VHDL

3 Hierarquia de Memória

4 Cache

Ao final da aula você saberá:

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- Como associar diversos *chips* de memória, além de dispositivos de entrada e saída, em um mesmo barramento de um sistema, formando um espaço de endereçamento;
- Como operações de entrada e saída podem ser realizadas por operações de leitura e escrita em memória;
- Como descrever memórias em VHDL no estilo comportamental;
- O conceito de hierarquia de memórias, por que ele é necessário e por que ele funciona;
- Memória cache, uma aplicação de hierarquia de memórias no nível mais próximo do processador.

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

Associação de Memórias e I/O

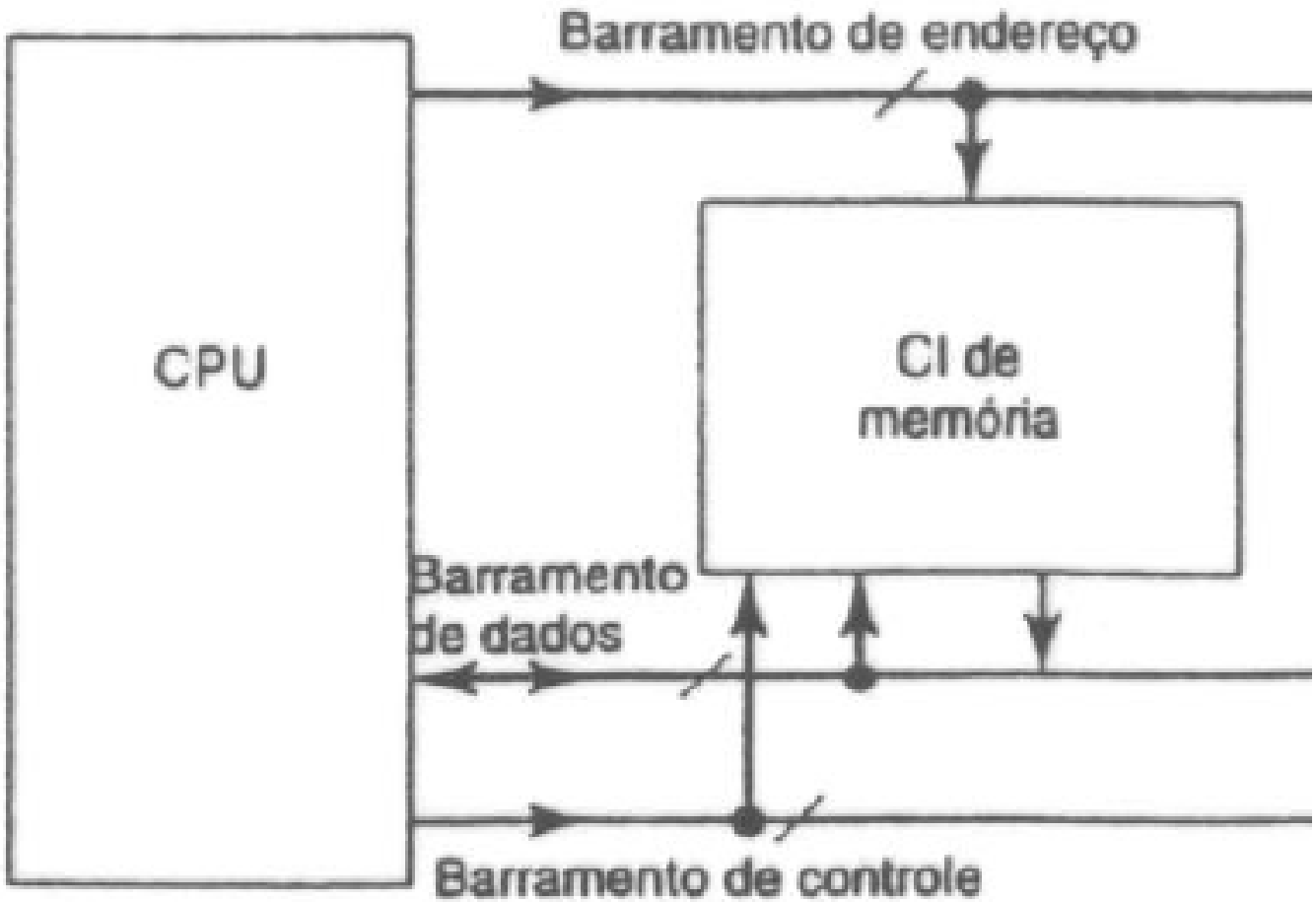
Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- No contexto do computador, a interface entre CPU e memória ocorre por meio do seguintes barramentos:
  - Barramento de endereços (*address bus*);
  - Barramento de dados (*data bus*);
  - Barramento de controle (*control bus*);

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache



Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Vamos considerar que o CI de memória apresentado tenha a capacidade de 256 x 8;
- O barramento de endereços possui 8 bits ($2^8 = 256$) e o barramento de dados também possui 8 bits;
- Nesse caso, todo o **espaço de endereçamento** está ocupado com o mesmo CI de memória, isto é, os endereços 0 a 255 (00h a FFh) acessam posições do único CI de memória da figura;

Faixa de Endereço (hex)	CI mapeado
00 a FF	CI 1

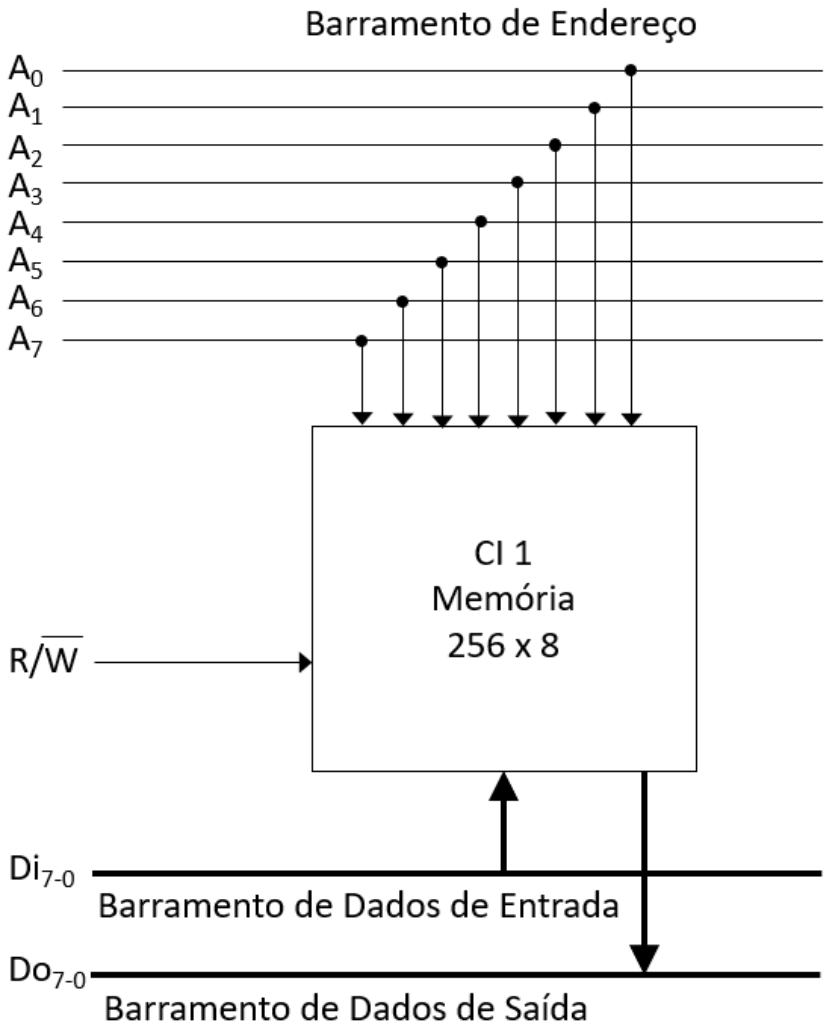
Associação de Memórias e I/O

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache



Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Podemos expandir a capacidade de memória do sistema adicionando novos CIs de memória aos barramentos;
- Mas vimos que neste exemplo todo o espaço de endereçamento está mapeado no CI 1;
- Como fazer?

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Adicionemos um 9º bit de endereço A_8 ao barramento de endereços, que passa a suportar $2^9 = 512$ posições;
- Adicionemos também um novo CI de memória, o CI 2 (igual ao CI 1), e vamos ligar o novo bit de endereço adequadamente ao *Chip Select* dos CIs;

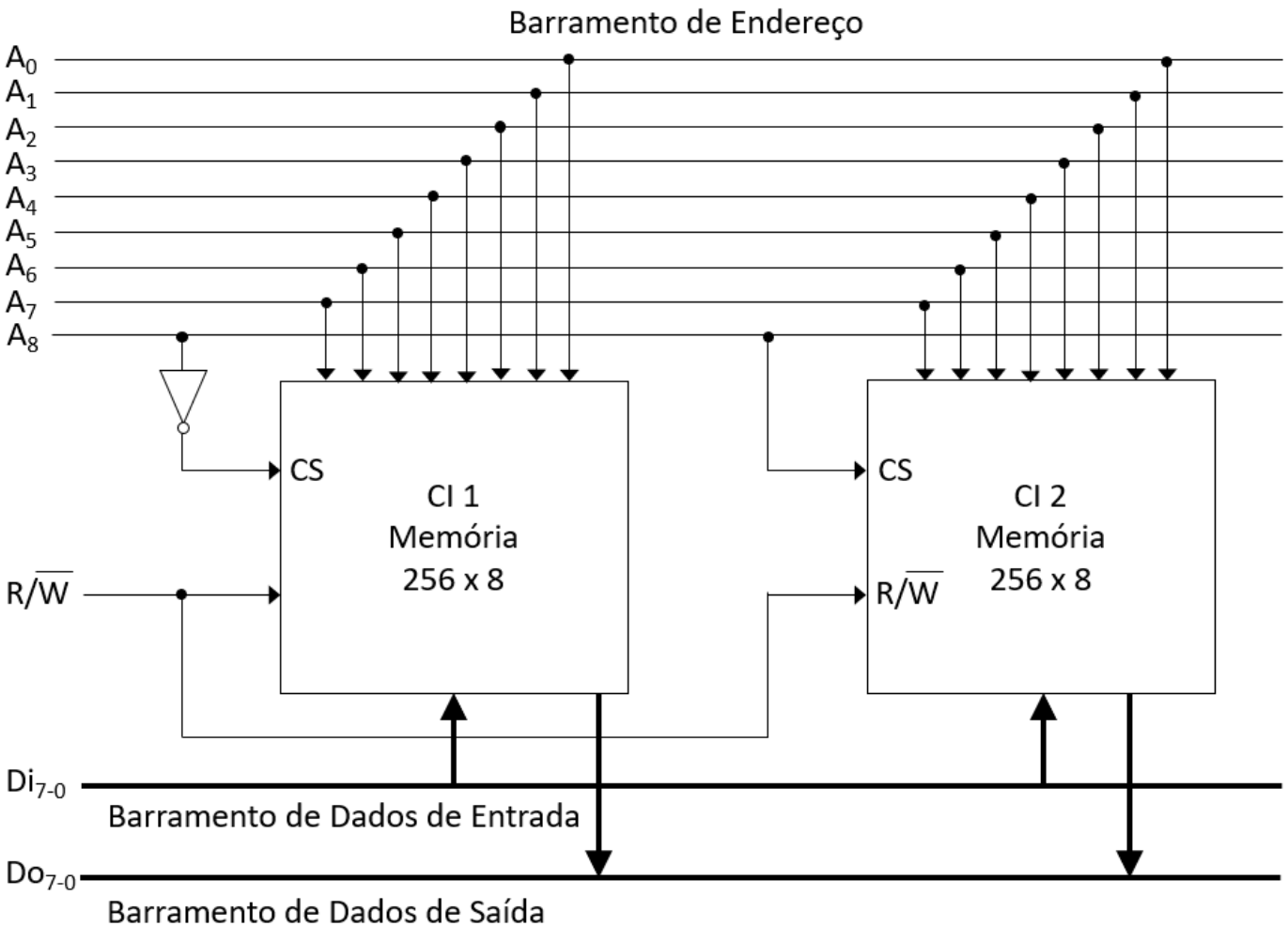
Associação de Memórias e I/O

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache



Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Relembrando o *Chip Select*:

TABLE 7-1
Control Inputs to a Memory Chip

Chip Select CS	Read/$\overline{\text{Write}}$ R/$\overline{\text{W}}$	Memory Operation
0	×	None
1	0	Write to selected word
1	1	Read from selected word

Figura: Sinais de controle de uma memória RAM - Fonte: [1].7 - Tabela 7.1

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Quando $A_8 = 0$, o *Chip Select* do CI 1 é ativado e o do CI 2 é desativado. O inverso ocorre quando $A_8 = 1$;
- Quando o *Chip Select* está ativo, é realizada a operação de leitura ou escrita conforme $R/\overline{W}$;
- **Quando o *Chip Select* está inativo, nenhuma operação é realizada e as saídas de dados do *chip* ficam em alta impedância**;

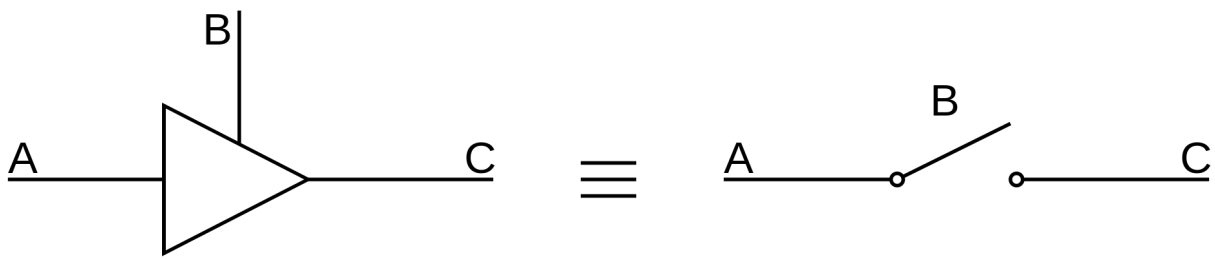


Figura: Buffer tri-state - Fonte: pt.wikipedia.org/wiki/Tristate

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- **Nota:** Em VHDL, os tipos *bit* e *bit_vector* só aceitam os valores lógicos 0 e 1. O tipo ***std_logic*** é mais "flexível", aceitando o valor Z, que representa alta impedância;

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- É utilizada a mesma via de dados para os dois CIs de memória. Apenas um dos *Chip Select* estarão ativos em um certo instante;
- A_8 é enxergado pelo sistema como mais um bit normal de endereço, e permitiu nesse exemplo associarmos diferentes pastilhas para compor um espaço de endereçamento maior;

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Passamos a ter o seguinte espaço de endereçamento:

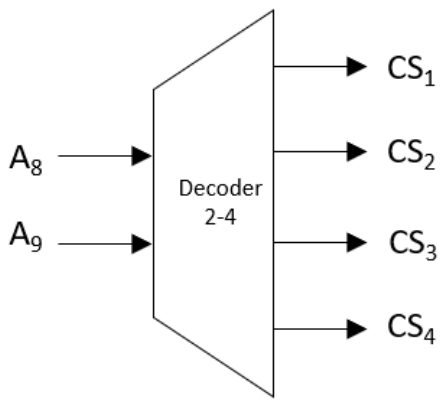
Faixa de Endereço		CI mapeado
hex	dec	
000 a 0FF	0 a 255	CI 1
100 a 1FF	256 a 511	CI 2

- Após a modificação, como o barramento de endereço ficou com 9 bits, não existem endereços acima de 1FFh = 511_{10};

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- De modo geral, podemos adicionar mais bits ao barramento de endereço e usar decodificadores nesses bits para gerar os sinais de *Chip Select* adequados;
- Por exemplo, se em vez de adicionar apenas o bit A_8 tivéssemos adicionado também um 10º bit A_9, poderíamos associar 4 CIs no mesmo barramento;
- Para isso precisaríamos gerar 4 sinais *Chip Select*:



Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- A conclusão é que um espaço de endereçamento pode ser construído via associação de memórias para endereçar diferentes pastilhas físicas;
- Um processador com barramento de endereço de 20 bits e barramento de dados de 8 bits, como o Intel 8088, pode endereçar até $2^{20} = 1.048.576 = 1\text{Mi}$ bytes;

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- É perfeitamente normal que existam faixas de endereços vazias em um espaço de endereçamento. Imagine que removêssemos o CI 1 do exemplo do *slide* 11. A faixa 000h a 0FFh passaria a não endereçar nenhum dispositivo físico;
- Se realizadas, operações de escrita não teriam efeito e operações de leitura retornariam algum valor *default* conforme a construção do barramento;

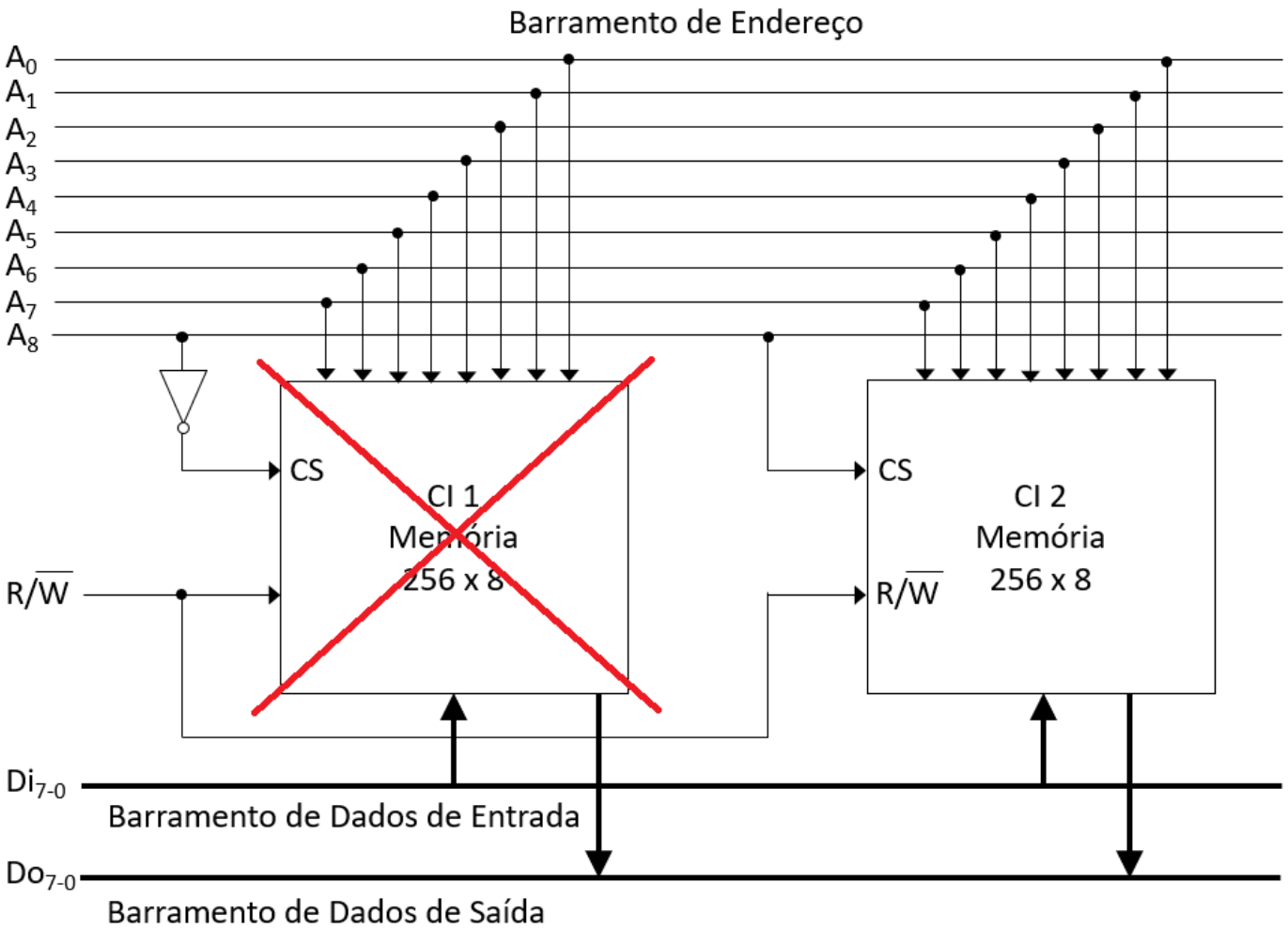
Associação de Memórias e I/O

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache



Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Este esquema pode ser utilizado para endereçar e acessar dispositivos de entrada/saída (I/O);
- Um **dispositivo de I/O mapeado em memória** é visto pelo sistema como se fosse uma memória normal, mas as operações de leitura e escrita têm alguma ação de entrada/saída;
- A memória e/ou registradores do dispositivo de I/O são associados a faixas de endereços;
- Assim, quando a CPU acessa certo endereço, ele pode estar associado não necessariamente a uma RAM física, mas também a um dispositivo de I/O;

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

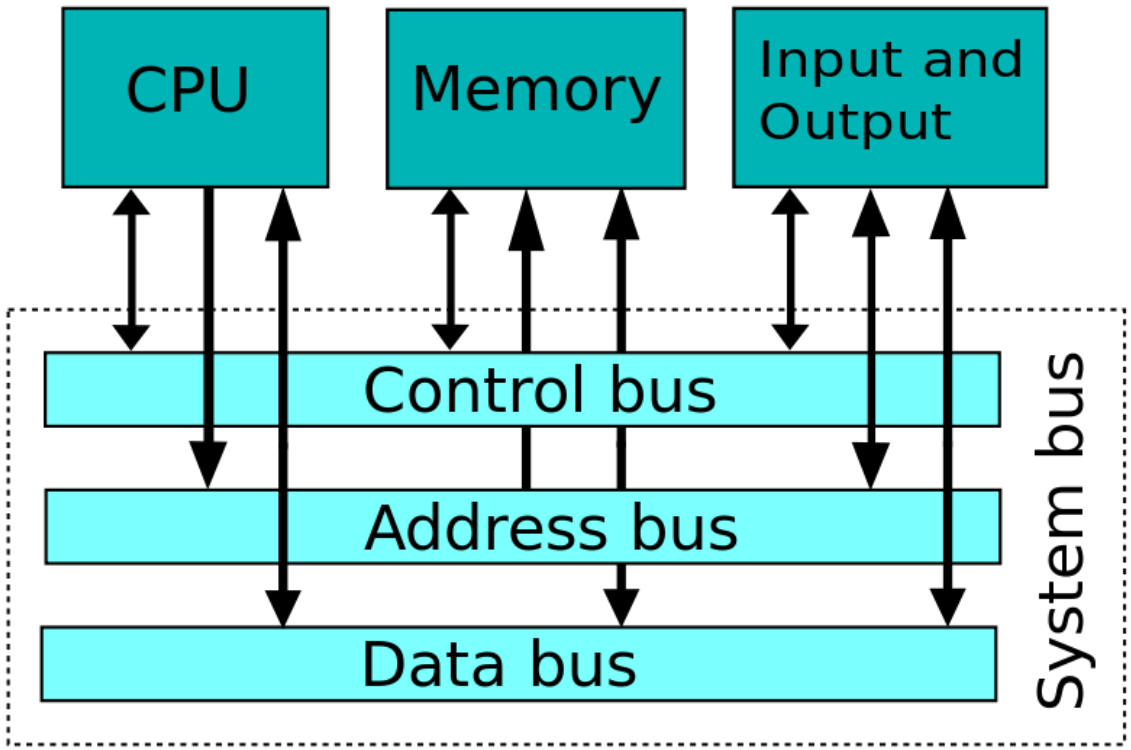


Figura: Dispositivo de I/O mapeado em memória - Fonte: https://en.wikipedia.org/wiki/System_bus

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Um exemplo é a **memória de vídeo**;
- A memória de vídeo é uma faixa de endereços de memória do computador associada ao dispositivo de saída de vídeo. Escrever em determinadas posições nesta faixa corresponde a alterar a saída gráfica que a placa de vídeo envia para o monitor;
- Tomemos como exemplo um sistema com espaço de endereçamento de 16 bits, que endereça um total de $2^{16} = 65.536$ bytes (fonte: https://en.wikipedia.org/wiki/Memory-mapped_I/O);

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

Faixa de Endereço (hex)	Tamanho	Dispositivo
0000 a 7FFF	32 KiB	RAM
8000 a 80FF	256 bytes	I/O geral
8100 a 8FFF	3 KiB	(vazio)
9000 a 90FF	256 bytes	Controlador de som
9100 a 9FFF	3 KiB	(vazio)
A000 a A7FF	2 KiB	Controlador de vídeo
A800 a BFFF	6 KiB	(vazio)
C000 a FFFF	16 KiB	ROM

- Suponha que o 4º byte da memória de vídeo esteja associado a um registrador que determina a cor de fundo da tela;

Associação de Memórias e I/O

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Escrever na posição A003h, como se fosse uma posição de memória RAM normal, corresponde, portanto, a alterar a cor de fundo da tela. Assim, realiza-se uma operação de saída com uma operação de acesso a memória;
- Observe que o espaço de endereços deste exemplo contém faixas vazias, o que já vimos ser normal.

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

Memórias em VHDL

Memórias em VHDL

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Lembrando que uma memória é, conceitualmente, um vetor unidimensional de palavras, podemos definir o seguinte tipo em VHDL:

```
type mem_tipo is array (0 to 255) of bit_vector (7 downto 0);
```

- Com o tipo definido, no lugar apropriado da *architecture*, podemos declarar um *signal* desse tipo:

```
signal mem: mem_tipo;
```

Memórias em VHDL

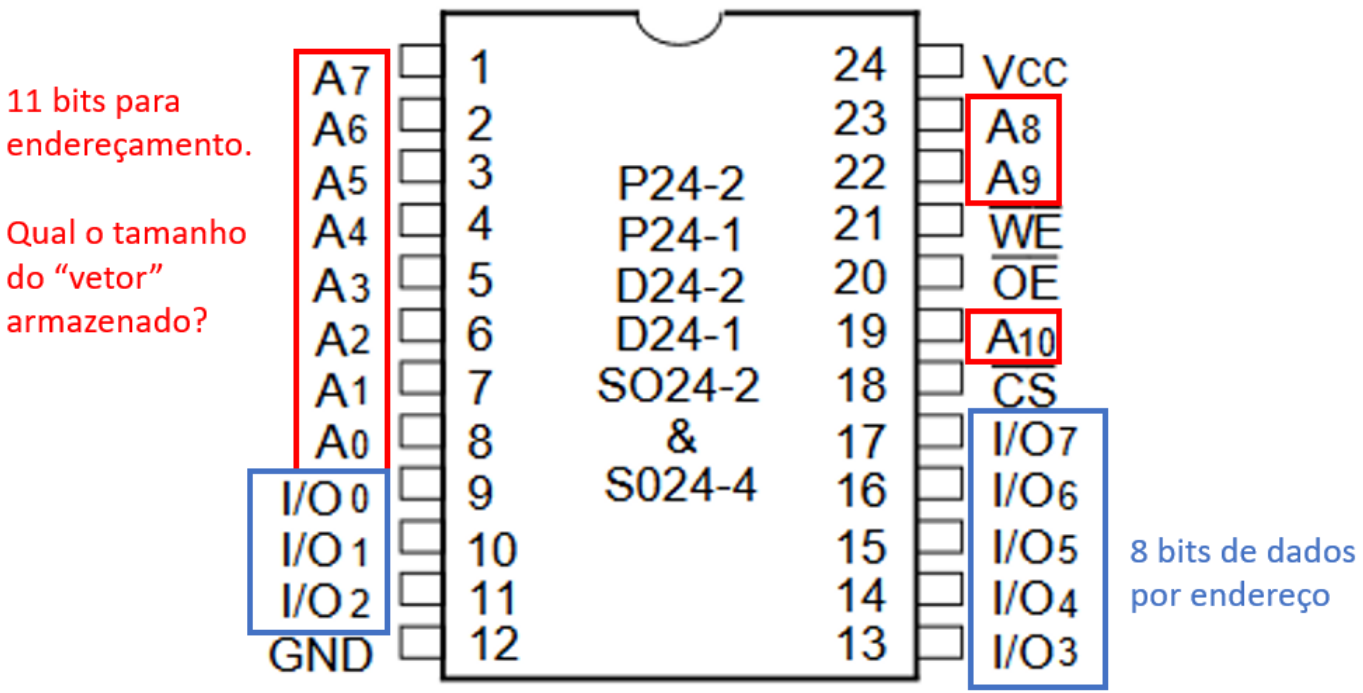
Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- Relembrando o CI 6116 (RAM):



Memórias em VHDL

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity ram6116 is
6      port (
7         -- 11 bits de endereço:
8         A: in std_logic_vector(10 downto 0);
9         -- 8 bits de tamanho da palavra:
10        IO: inout std_logic_vector(7 downto 0);
11        -- Entradas de controle:
12        WE_b: in std_logic; -- Write Enable  (ativo baixo)
13        CS_b: in std_logic; -- Chip Select   (ativo baixo)
14        OE_b: in std_logic  -- Output Enable (ativo baixo)
15     );
16  end ram6116;
17
```

Memórias em VHDL

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

```
18  architecture ram6116_arch of ram6116 is
19     -- Tipo do vetor com tamanho 2Ki (2^11 - 1 = 2047):
20     type mem_tipo is array (0 to 2047) of std_logic_vector(7 downto 0);
21     -- Vetor de palavras:
22     signal mem: mem_tipo;
23  begin
24     process (A,WE_b,CS_b,OE_b) is
25     begin
26        IO <= "ZZZZZZZZ";
27        if CS_b = '0' then  -- Chip Select ativo?
28           if WE_b = '0'then
29              -- Write Enable ativo: escrita.
30              mem(to_integer(unsigned(A))) <= IO;
31           end if;
32           if (WE_b = '1') and (OE_b = '0') then
33              -- Write Enable inativo e Output Enable ativo: leitura
34              IO <= mem(to_integer(unsigned(A)));
35           else
36              IO <= "ZZZZZZZZ";
37           end if;
38        end if;
39     end process;
40  end ram6116_arch;
```

Memórias em VHDL

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- E como seria a descrição uma ROM?

Memórias em VHDL

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity rom is
6      port (
7         -- 4 bits de endereço:
8         address: in std_logic_vector(3 downto 0);
9         -- 8 bits de tamanho de palavra de dados:
10        data: out std_logic_vector(7 downto 0)
11     );
12  end rom;
13
```

Memórias em VHDL

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

```
14  architecture rom_arch of rom is
15    type mem_tipo is array (0 to 15) of std_logic_vector(7 downto 0);
16    constant mem: mem_tipo :=
17     (0  => "00000000",
18      1  => "00000001",
19      2  => "00000010",
20      3  => "00000011",
21      4  => "00000100",
22      5  => "11110000",
23      6  => "00001111",
24      7  => "11110000",
25      8  => "11110000",
26      9  => "11110000",
27      10 => "11111111",
28      11 => "11111111",
29      12 => "11110000",
30      13 => "01010101",
31      14 => "10101010",
32      15 => "11111111");
33  begin
34     data <= mem(to_integer(unsigned(address)));
35  end rom_arch;
```

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

Hierarquia de Memória

Hierarquia de Memória

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- O desejável seria que os computadores dispusessem de memórias com muita capacidade de armazenamento, baixo tempo de acesso (rápidas) e com baixo custo por bit (baratas);
- Como não é possível conseguir tudo em uma só tecnologia, utiliza-se a técnica de **hierarquia de memória** para criar uma "ilusão" que se aproxima da situação desejada;

Hierarquia de Memória

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- Uma **hierarquia de memória** é uma estrutura que usa múltiplos níveis de memória baseadas em diferentes tecnologias: à medida que a distância entre o nível e o processador aumenta, o tamanho (capacidade de armazenamento) e o tempo de acesso da memória no nível em questão também aumentam [2].5;

Hierarquia de Memória

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

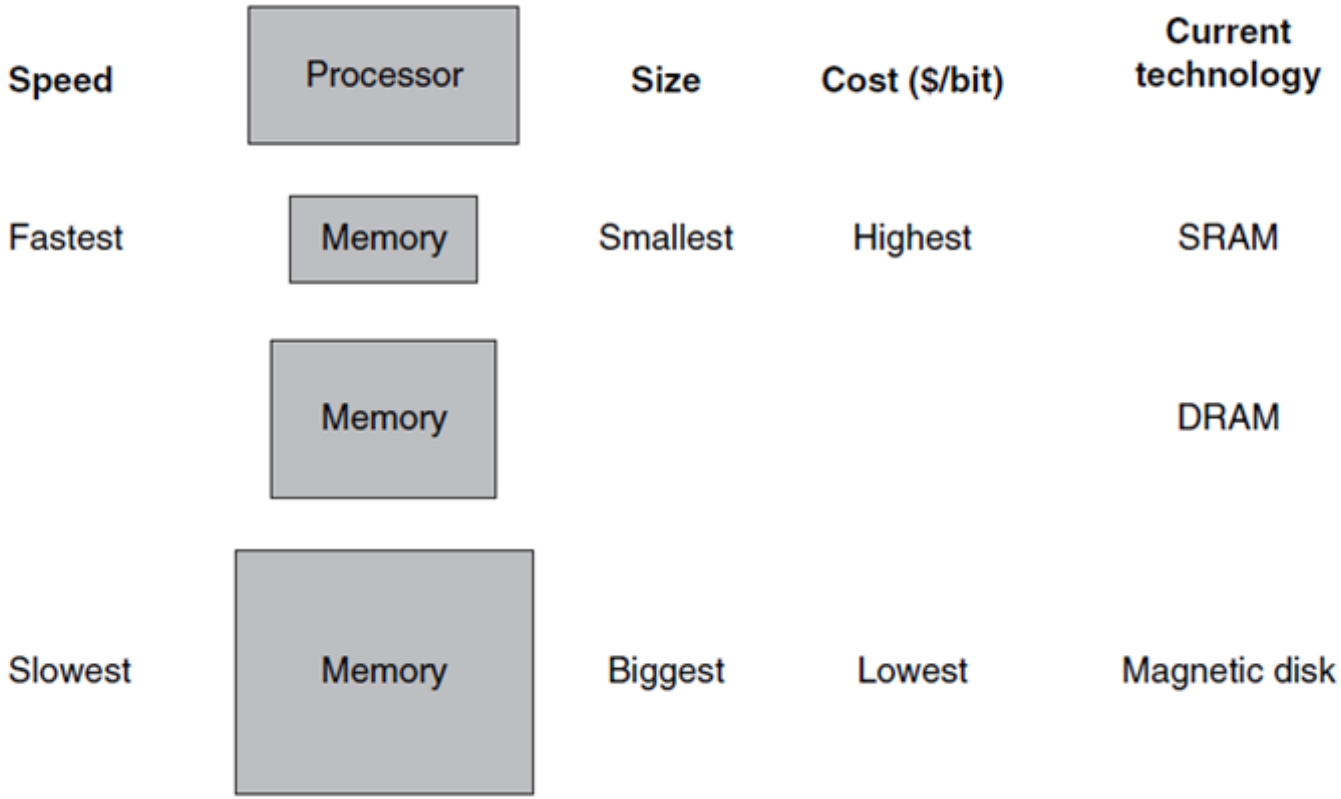


FIGURE 5.1 The basic structure of a memory hierarchy. By implementing the memory system as a hierarchy, the user has the illusion of a memory that is as large as the largest level of the hierarchy, but can be accessed as if it were all built from the fastest memory. Flash memory has replaced disks in many personal mobile devices, and may lead to a new level in the storage hierarchy for desktop and server computers; see Section 5.2.

Hierarquia de Memória

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- **Analogia:** Considere a analogia de um estudante em uma biblioteca fazendo uma pesquisa em livros [2].5;
- O estudante trabalha sentado em uma mesa, mas os livros estão nas prateleiras da biblioteca;
- Cada vez que precisa de um livro, o estudante precisa se levantar, caminhar até a prateleira, pegar o livro e trazer à mesa;

Hierarquia de Memória

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Quando livro está na prateleira, o "tempo de acesso" ao livro é lento, mas quando já está na mesa, é rápido;
- Porém, não é possível trazer todos os livros da biblioteca à mesa, pois a "capacidade" da mesa é bem menor que a de toda a biblioteca;
- O estudante precisa ter uma estratégia para manter o conjunto de livros que mais vai precisar por mais tempo na mesa, minimizando o número de vezes que precisa se deslocar à prateleira para substituir livros;

Hierarquia de Memória

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- Em uma pesquisa, é provável que o estudante precise consultar vários livros sobre o mesmo o assunto, e não apenas um;
- A biblioteca já é organizada para que livros sobre o mesmo assunto fiquem próximos, então o estudante pode trazer de uma só vez vários livros que estavam próximos (princípio da **localidade espacial**);

Hierarquia de Memória

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- Também é provável que o estudante fique por algum tempo consultando os mesmos livros, isto é, ele não vai interromper a leitura de um conjunto de livros com frequência para depois retomar aos mesmos;
- Ou seja, se um conjunto de livros está sendo consultado agora, ele tende a ser consultado nos próximos instantes (princípio da **localidade temporal**);

Hierarquia de Memória

Associação de Memórias e I/O
Memórias em VHDL
Hierarquia de Memória
Cache

- Observa-se que o funcionamento de programas de computador, de modo geral, apresentam esses dois princípios:
- **Localidade espacial**: Instruções normalmente são acessadas sequencialmente pelo processador, assim como dados por um programa (ex. acesso sequencial a elementos de um vetor);
- **Localidade temporal**: *Loops* são estruturas muito comuns em vários programas. Sendo assim, instruções e dados provavelmente serão acessados repetidamente;

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

Cache

Cache

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- ***Cache*** é o nome do nível de memória que fica entre o processador e a memória principal;
- Corresponde à aplicação do conceito de hierarquia de memórias no nível mais próximo do processador, sendo baseadas em SRAM;
- Atualmente, memórias *cache* são integradas no mesmo *chip* que o processador, não sendo constituídas por chips SRAM independentes;

Cache

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- A ideia é que o processador tenha à disposição de forma bem rápida os dados que está manipulando em um certo momento, usando a localidade temporal e espacial como ideias chave, como no exemplo da biblioteca;
- Isso é conseguido trazendo-se "para perto" (na SRAM do próprio processador) os blocos de dados da memória principal (DRAM) que precisam ser acessados;

Cache

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

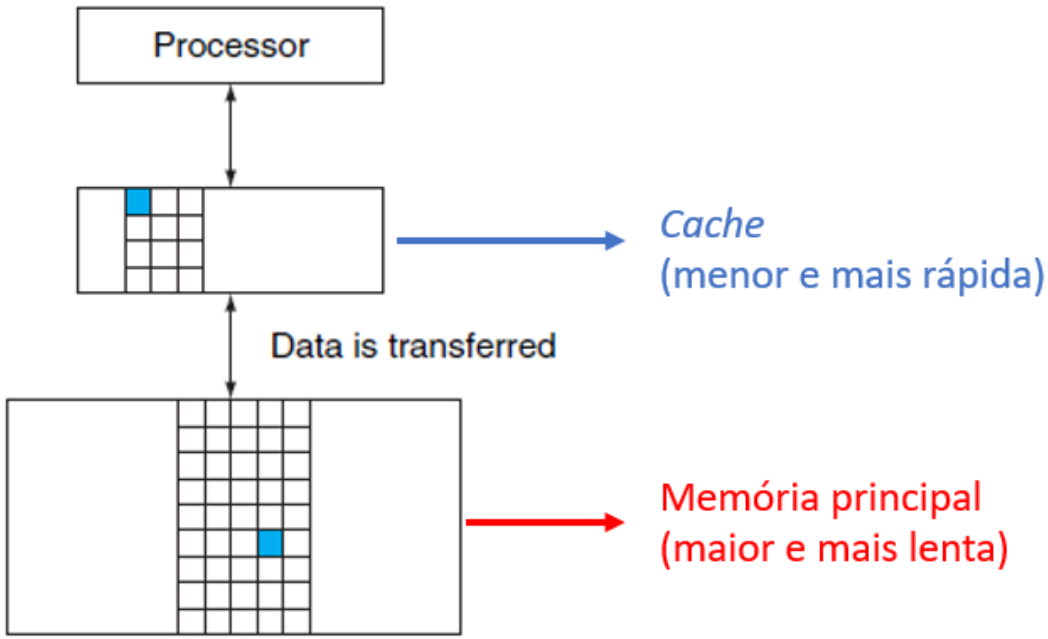


FIGURE 5.2 Every pair of levels in the memory hierarchy can be thought of as having an upper and lower level. Within each level, the unit of information that is present or not is called a *block* or a *line*. Usually we transfer an entire block when we copy something between levels.

Cache

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- Ao precisar de acesso a um bloco que já está no *cache*, o processador o obtém de forma muito rápida, correspondendo a um **acerto** (***hit***);
- Ao precisar de acesso a um bloco que ainda não está no *cache*, tal bloco precisa ser trazido da memória principal ao *cache*, correspondendo a um **engano** (***miss***);
- Esse tempo (***miss penalty***) é compensado pelos próximos acessos que provavelmente serão feitos e já estarão no *cache*;

Cache

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

- A "ilusão" de que o processador tem uma memória grande e rápida à sua disposição ocorre se a taxa de acerto (*hit rate*) for alta, ou seja, se na maioria das vezes em que precisar de um dado ele estará pronto à disposição no *cache*;
- A alta taxa de acerto é uma realidade graças aos conceitos de localidade espacial e temporal;
- Observa-se na prática taxas de acerto superiores a 95% [2].5 (Fig. 5.11);

Cache

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

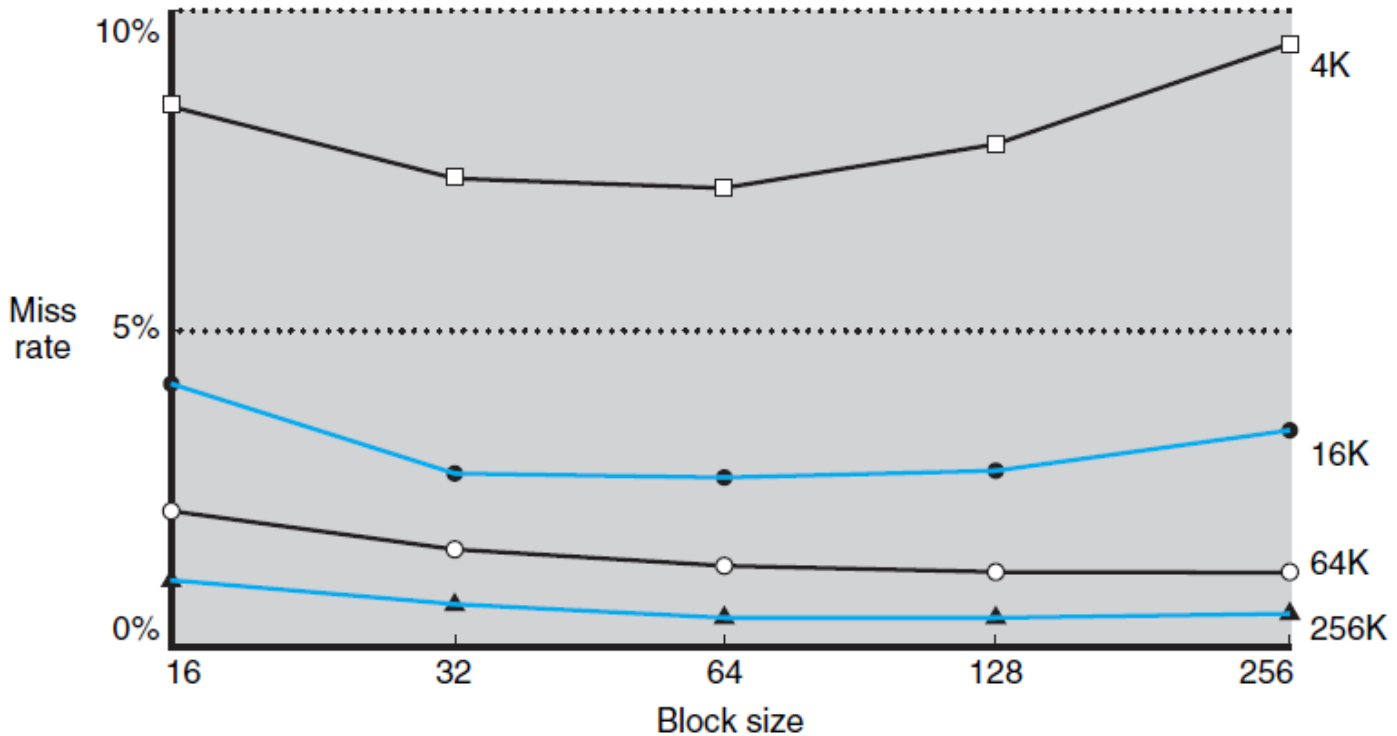


FIGURE 5.11 Miss rate versus block size. Note that the miss rate actually goes up if the block size is too large relative to the cache size. Each line represents a cache of different size. (This figure is independent of associativity, discussed soon.) Unfortunately, SPEC CPU2000 traces would take too long if block size were included, so these data are based on SPEC92.

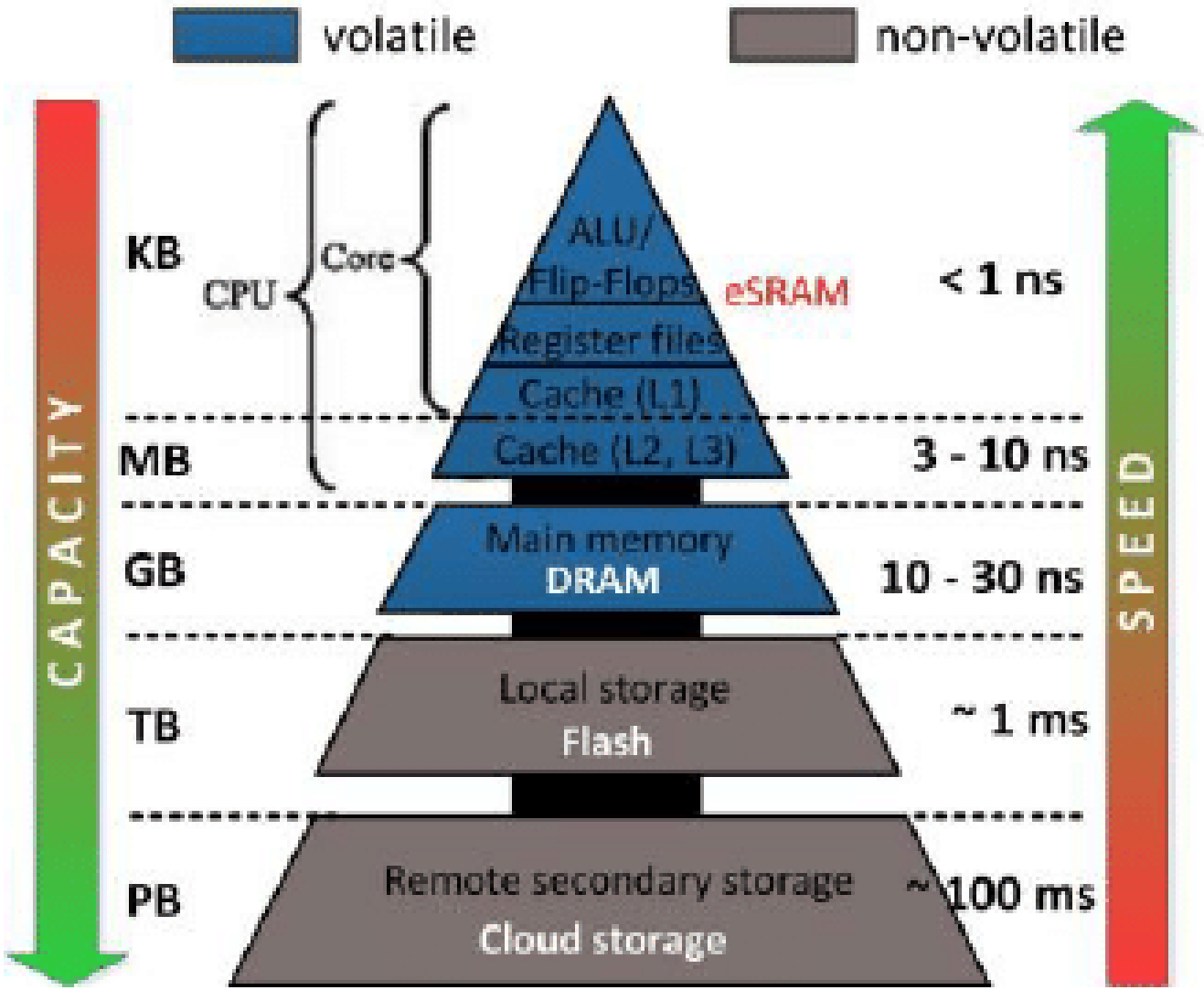
Cache

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache



Obrigado!


Universidade de São Paulo
Departamento de Engenharia de Computação e Sistemas Digitais
PCS

Referências

Associação de Memórias e I/O

Memórias em VHDL

Hierarquia de Memória

Cache

M. Mano, C. Kime, and T. Martin.
Logic and Computer Design Fundamentals.
Pearson Education, Incorporated, 2015.

D. Patterson and J. Hennessy.
Computer Organization and Design ARM Edition: The Hardware Software Interface.
The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science, 2016.