ALU e Memórias

Prof. Denis Fantinato
Profa. Letícia Rittner

O que será visto nesta aula

- ALU

- Memórias

Projeto de Contadores Síncronos

Uma **Unidade Lógica e Aritmética**, ou **ALU**, ou **ULA**, é uma rede combinacional, que implementa uma função de suas entradas com base em **operações lógicas ou aritméticas**

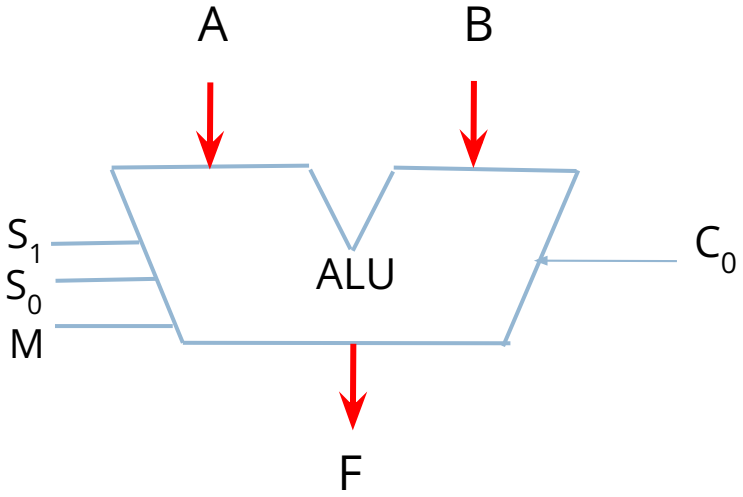
ALUs constituem o elemento central dos computadores e da maioria dos sistemas digitais

Nesta aula, vamos aprender sobre algumas partes desse sistema

ALU Genérica

Operações lógicas e Aritméticas

M = 0 – Modo Lógico
M = 1 – Modo Aritmético



Prof. Denis Fantinato
Profa. Letícia Rittner

ALU Genérica

Operações lógicas e Aritméticas

M = 0, Logical Bitwise Operations

S1	S0	Function	Comment
0	0	Fi = Ai	Input Ai transferred to output
0	1	Fi = not Ai	Complement of Ai transferred to output
1	0	Fi = Ai xor Bi	Compute XOR of Ai, Bi
1	1	Fi = Ai xnor Bi	Compute XNOR of Ai, Bi

M = 1, C0 = 0, Arithmetic Operations

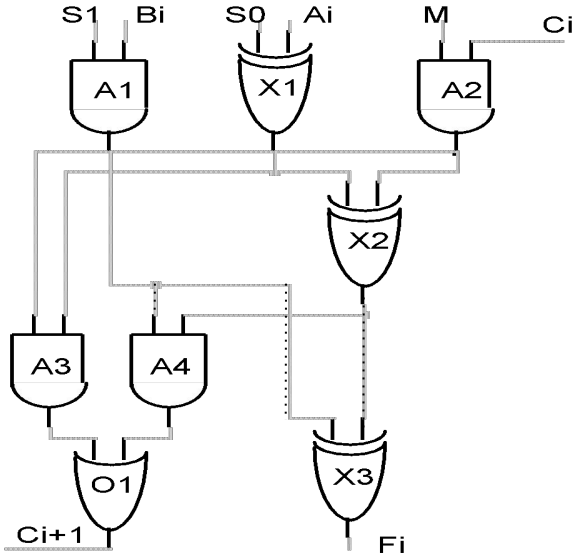
0	0	F = A	Input A passed to output
0	1	F = not A	Complement of A passed to output
1	0	F = A plus B	Sum of A and B
1	1	F = (not A) plus B	Sum of B and complement of A

M = 1, C0 = 1, Arithmetic Operations

0	0	F = A plus 1	Increment A
0	1	F = (not A) plus 1	Twos complement of A
1	0	F = A plus B plus 1	Increment sum of A and B
1	1	F = (not A) plus B plus 1	B minus A

ALU Genérica

Operações lógicas e Aritméticas



8 Portas Lógicas (porém 3 XOR)

S1 = 0 bloqueia Bi
Isto ocorre para operações que envolvem apenas Ai

O mesmo se aplica para Ci quando M = 0

A Adição (Ai xor Bi) ocorre quando M = 1

Ci, Bi passam para as XORs X2, X3

S0 = 0, X1 passa A

S0 = 1, X1 passa A´

Modo Aritmético (M=1):

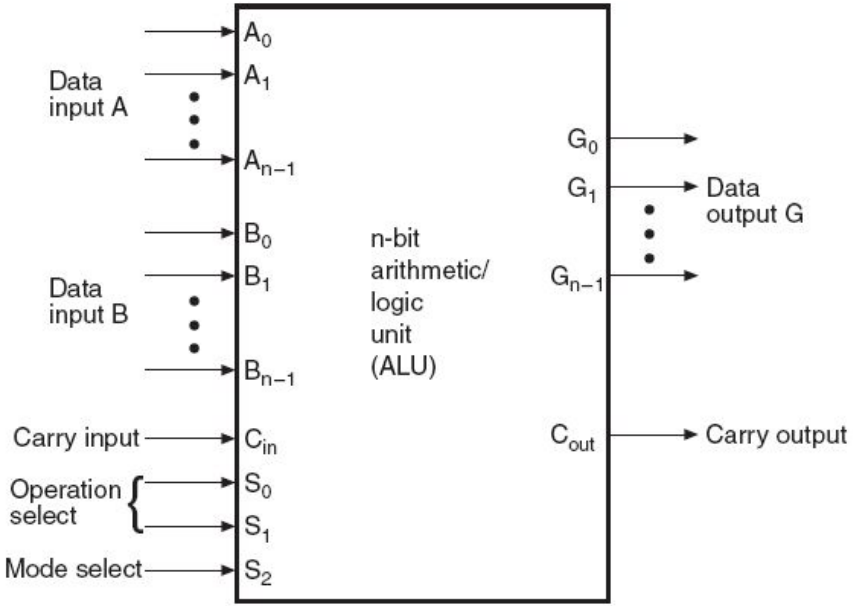
O Vai-Um na porta OR (O1) é Ai Ci + Bi (Ai xor Ci)

Modo Lógico (M=0):

XORs em cascata formam a saída a partir de Ai e Bi

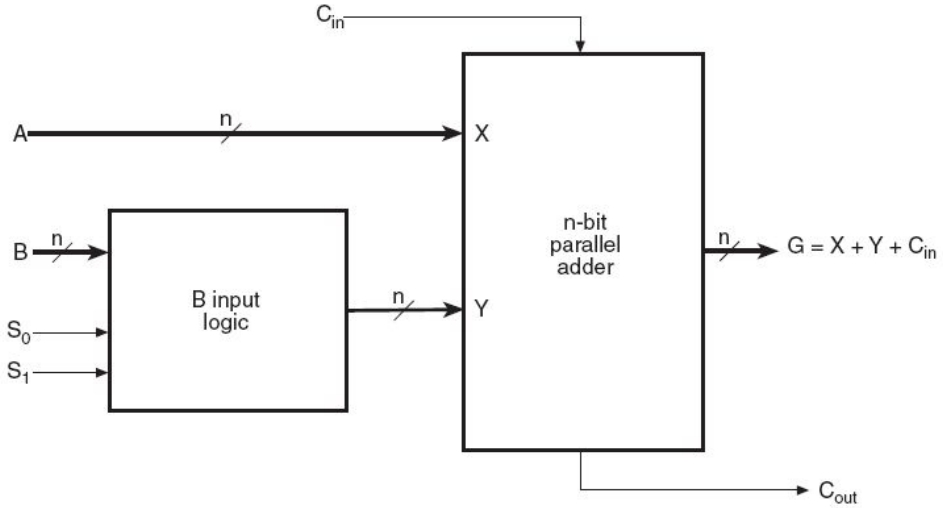
ALU Genérica de n bits

Componente Básico



O componente básico de um circuito aritmético é o somador paralelo, que é construído por vários somadores completos de 1 *bit* ligados em cascata

Diagrama de Blocos do Circuito Aritmético



Nesta configuração, as *n* entradas *B* do somador paralelo são controladas pelas linhas de seleção S_1 e S_0

Desta forma, a entrada *Y* do somador pode receber além das entradas *B*, o complemento destas entradas, um conjunto de *0s* ou *1s etc.*

Tabela de Funções do Circuito Aritmético

Select		Input	$G = A + Y + C_{in}$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0's	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1's	$G = A - 1$ (decrement)	$G = A$ (transfer)

A tabela acima mostra as possíveis operações aritméticas obtidas com a ajuda das linhas de seleção S_1 e S_0

Se as entradas provenientes de *B* forem ignoradas, a entrada *Y* recebe *0s* e a saída *G* simplesmente reproduz o valor das entradas *A*, desde que $C_{in} = 0$ ($G = A + 0 + C_{in}$)

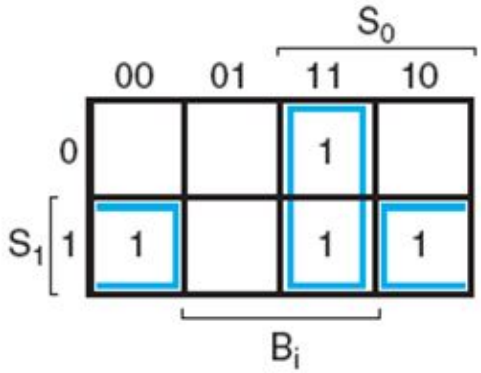
Se o complemento de *B* for aplicado em *Y*, e $C_{in} = 1$, obtém-se a subtração aritmética $G = A - B$, ou seja, *A* é somado com o complemento de *2* de *B* ($G = A + B' + 1$)

Tabela Verdade do Circuito Aritmético

Inputs			Output	
S_1	S_0	B_i	Y_i	
0	0	0	0	$Y_i = 0$
0	0	1	0	
0	1	0	0	$Y_i = B_i$
0	1	1	1	
1	0	0	1	$Y_i = \overline{B}_i$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

Colocando-se numa Tabela da Verdade as entradas S_1, S_0 e B_i (um *bit* de *B* apenas), tendo como saída Y_i, a equação booleana simplificada de Y_i pode ser facilmente obtida como uma *SOP* (Soma de Produtos) de suas entradas

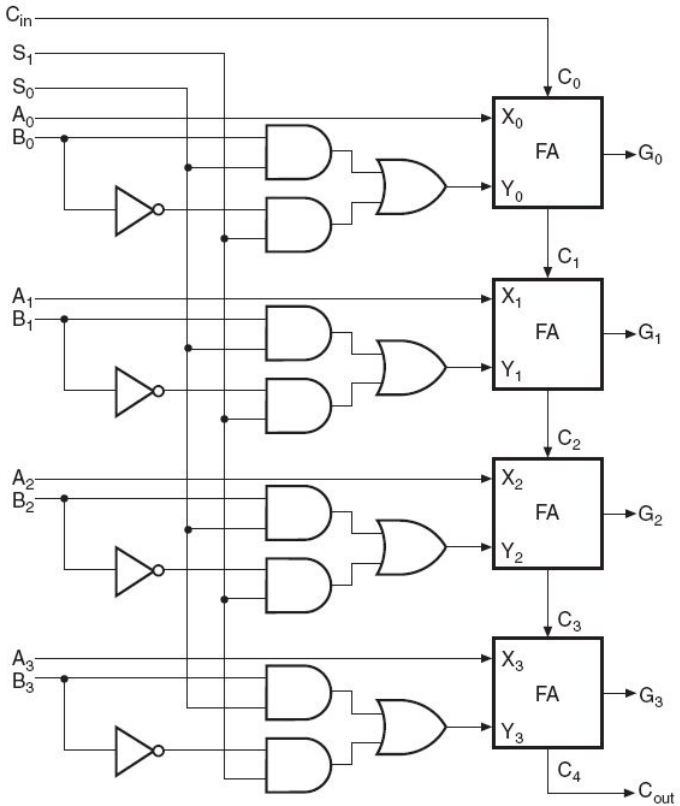
Tabela Verdade do Circuito Aritmético



(b) Map Simplification:
$Y_i = B_i S_0 + \overline{B}_i S_1$

A lógica de entrada de *B* (*B input logic*) pode ser obtida com *n MUXes 4x1* (*00, 01, 10, 11*) ou, com menos portas, através de uma simplificação usando Mapas de Karnaugh

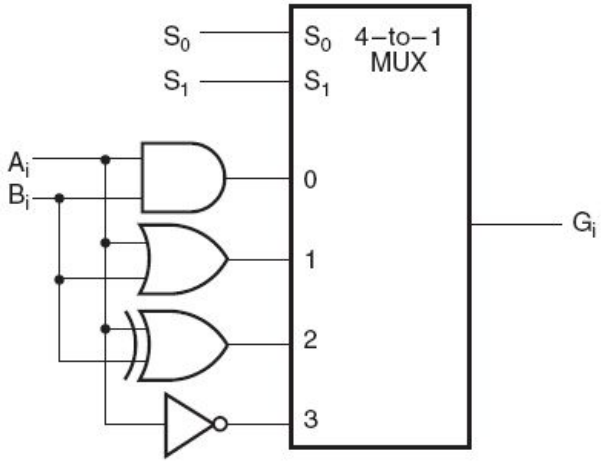
Diagrama Lógico do Circuito Aritmético



A figura mostra o diagrama lógico de um circuito aritmético para n = 4

Os quatro somadores completos (*FA – Full-Adder*) constituem o somador paralelo

Circuito Lógico



S_1	S_0	Output	Operation
0	0	$G = A \wedge B$	AND
0	1	$G = A \vee B$	OR
1	0	$G = A \oplus B$	XOR
1	1	$G = \overline{A}$	NOT

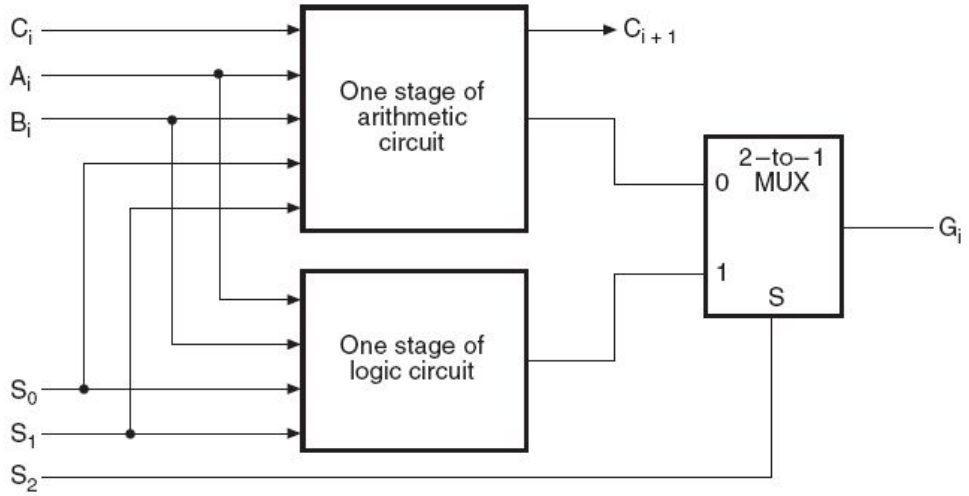
(b) Function Table

Microoperações lógicas manipulam os *bits* dos operandos considerando cada *bit* em um registrador como uma variável binária, desta forma realizando operações *bit* a *bit*

A partir das quatro operações lógicas disponíveis nas entradas do *MUX 4x1*, é possível obter outras operações lógicas

Para um circuito lógico de *n bits*, este diagrama precisa ser repetido *n* vezes

ALU



A *ALU* é construída combinando-se o circuito aritmético com o lógico, sendo as entradas de seleção S_1 e S_0 comum aos dois circuitos, com S_2 decidindo sobre o modo de operação, aritmético ($S_2 = 0$) ou lógico ($S_2 = 1$)

Para uma *ALU* de *n bits*, este diagrama precisa ser repetido *n* vezes

Tabela de Funções da ALU

Operation Select					
S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	$G = A$	Transfer A
0	0	0	1	$G = A + 1$	Increment A
0	0	1	0	$G = A + B$	Addition
0	0	1	1	$G = A + \underline{B} + 1$	Add with carry input of 1
0	1	0	0	$G = A + \underline{B}$	A plus 1's complement of B
0	1	0	1	$G = A + \underline{B} + 1$	Subtraction
0	1	1	0	$G = A - 1$	Decrement A
0	1	1	1	$G = A$	Transfer A
1	0	0	X	$G = A \wedge B$	AND
1	0	1	X	$G = A \vee B$	OR
1	1	0	X	$G = \underline{A} \oplus B$	XOR
1	1	1	X	$G = \overline{A}$	NOT (1's complement)

A ALU do exemplo oferece oito operações aritméticas e quatro lógicas, selecionadas através de S_2, S_1, S_0 e C_{in} (durante as operações lógicas, C_{in} não tem nenhum efeito sobre o resultado, podendo seu valor ser 0 ou 1)

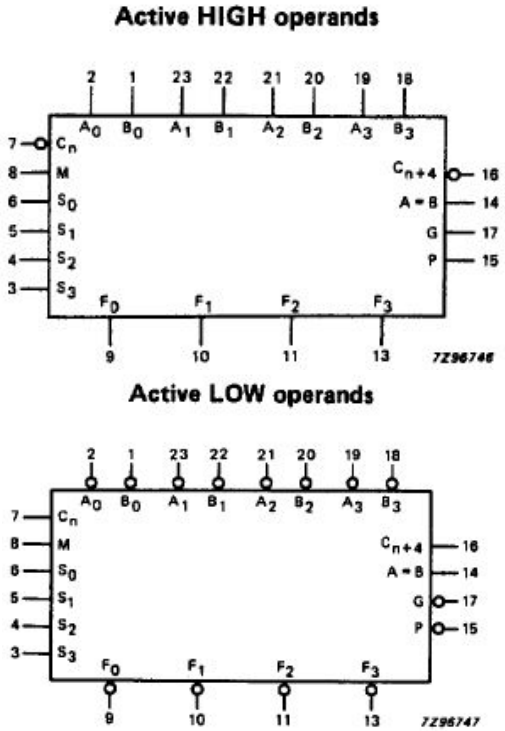
Exemplo: ALU 74HC181

As entradas e saídas da 74x181 são normalmente ativas em *LOW*, mas esta ALU pode ser utilizada com suas entradas e saídas ativas em *HIGH* (fazendo operações complementares)

MODE SELECT INPUTS				ACTIVE HIGH INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC[2] (M=L; C_n=H)
L	L	L	L	$\overline{A}$	A
L	L	L	H	$\overline{A+B}$	A + B
L	L	H	L	$\overline{A}B$	A + $\overline{B}$
L	L	H	H	logical 0	minus 1
L	H	L	L	$\overline{AB}$	A plus A$\overline{B}$
L	H	L	H	$\overline{B}$	(A + B) plus A$\overline{B}$
L	H	H	L	$A \oplus B$	A minus B minus 1
L	H	H	H	A$\overline{B}$	A$\overline{B}$ minus 1
H	L	L	L	$\overline{A}+B$	A plus AB
H	L	L	H	$\overline{A \oplus B}$	A plus B
H	L	H	L	B	(A + $\overline{B}$) plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	logical 1	A plus A[1]
H	H	L	H	A + $\overline{B}$	(A + $\overline{B}$) plus A
H	H	H	L	A + B	(A + $\overline{B}$) plus A
H	H	H	H	A	A minus 1

MODE SELECT INPUTS				ACTIVE LOW INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC[2] (M=L; C_n=L)
L	L	L	L	$\overline{A}$	A minus 1
L	L	L	H	$\overline{AB}$	AB minus 1
L	L	H	L	$\overline{A}+B$	A$\overline{B}$ minus 1
L	L	H	H	logical 1	minus 1
L	H	L	L	$\overline{A+B}$	A plus (A + $\overline{B}$)
L	H	L	H	$\overline{B}$	AB plus (A + $\overline{B}$)
L	H	H	L	$\overline{A \oplus B}$	A minus B minus 1
L	H	H	H	A + $\overline{B}$	A + $\overline{B}$
H	L	L	L	$\overline{A}B$	A plus (A + B)
H	L	L	H	$A \oplus B$	A plus B
H	L	H	L	B	A$\overline{B}$ plus (A + B)
H	L	H	H	A + B	A + B
H	H	L	L	logical 0	A plus A[1]
H	H	L	H	A$\overline{B}$	AB plus A
H	H	H	L	AB	A$\overline{B}$ plus A
H	H	H	H	A	A

Exemplo: ALU 74HC181



A pinagem da 74x181 tanto para o operação com suas entradas e saídas ativas em *HIGH* ou *LOW* é a mesma, mas os conjuntos de operações lógicas e aritméticas são distintos

As entradas (A_3-A_0 e B_3-B_0) e as saídas (F_3-F_0) são para quatro *bits*.

Quando *M=1*, são selecionadas através de S_3-S_0 as operações lógicas, cujos resultados dependem apenas das entradas A_i e B_i, sendo C_n (*Vem_Um*) e C_{n+4} (*Vai_Um*) ignorados

Quando *M=0*, são selecionadas as operações aritméticas e o *Vem_Um* se propaga entre os estágios

Memórias

- Uma das principais vantagens dos sistemas digitais com relação aos analógicos é a facilidade de armazenamento de dados.

- As memórias são partes fundamentais dos computadores.

- Vimos que os registradores construídos com flip-flops são um tipo de memória.

- Veremos agora outros tipos de memórias, seus princípios de funcionamento, classificações e aplicações.

Memórias

- Termos importantes relacionados à memórias:
  - Célula.
  - Palavra/Largura.
  - Capacidade.
  - Densidade.

- Classes de Memórias:
  - Memórias voláteis – maioria das RAM (Random Access Memory).
  - Memórias não-voláteis – todas as ROM (Read Only Memory).

Memórias

- A escolha da memória mais adequada para cada situação depende de um compromisso entre: **custo, velocidade, densidade**, consumo de energia, durabilidade e persistência dos dados.

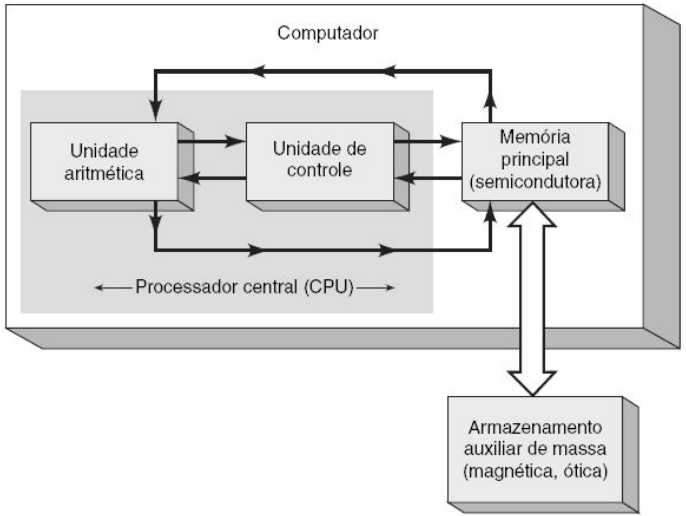
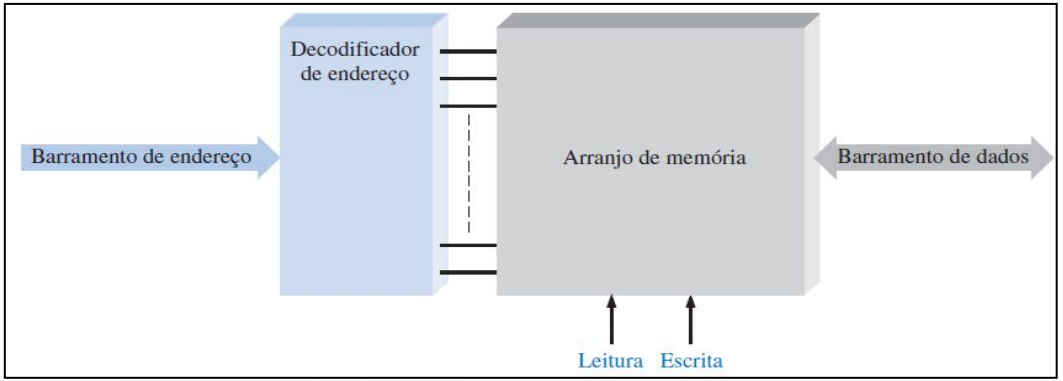
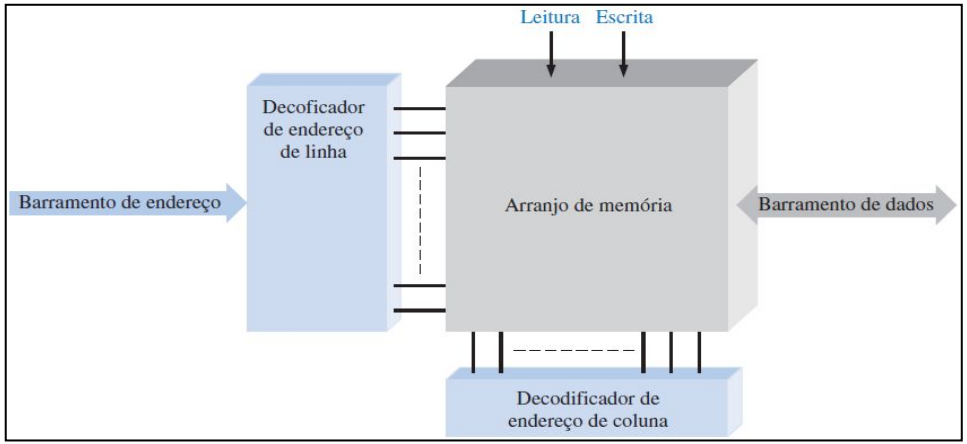


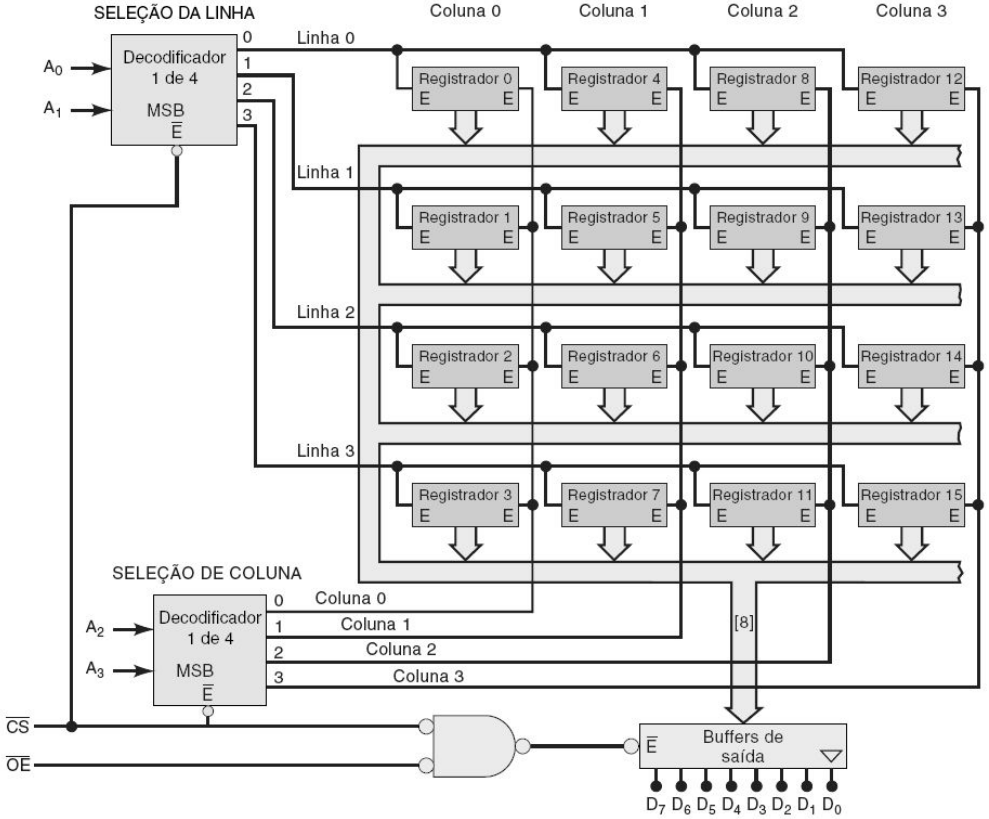
FIGURA 12.1 Um sistema de computador geralmente usa uma memória principal de alta velocidade e uma memória auxiliar externa mais lenta.

Arranjos Básicos de Memória

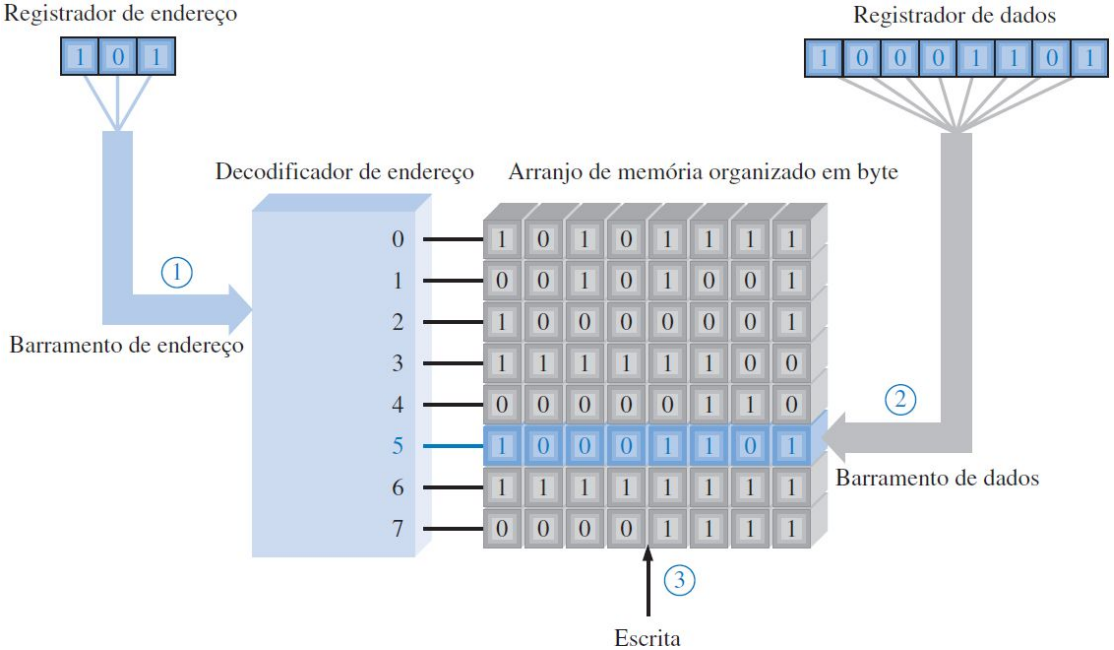




Arranjos Básicos de Memória



Operação de Escrita em Memórias

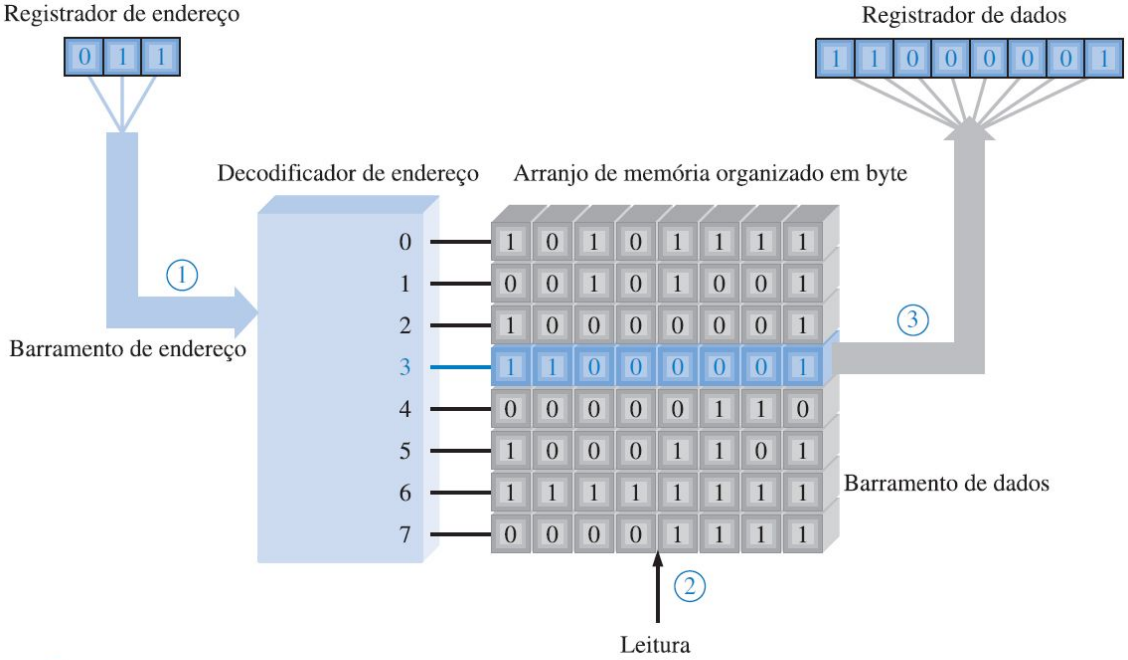


① O código de endereço 101 é colocado no barramento de endereço e o endereço 5 é selecionado.

② O byte de dados é colocado no barramento de dados.

③ O comando de escrita faz com que o byte de dados seja armazenado no endereço 5, substituindo o dado anterior.

Operação de Leitura em Memórias



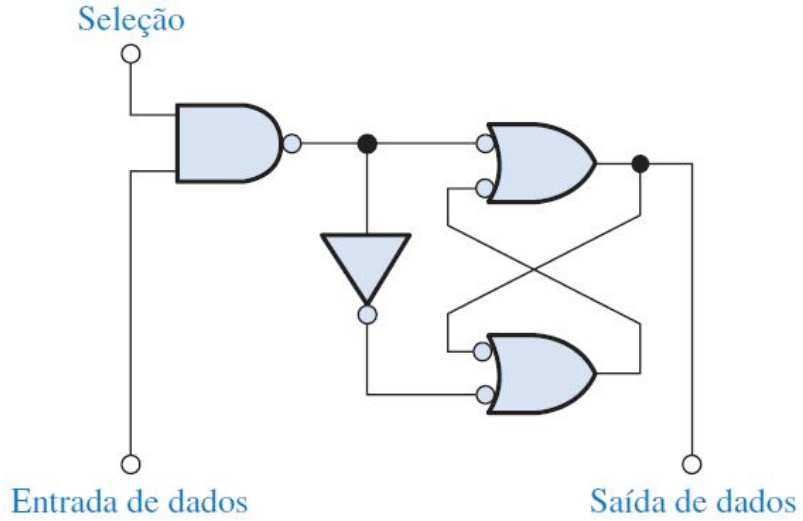
1. O código de endereço 011 é colocado no barramento de endereço e o endereço 3 é selecionado.
2. O comando de leitura é aplicado.
3. O conteúdo do endereço 3 é colocado no barramento de dados e deslocado no registrador de dados. O conteúdo do endereço 3 não é apagado pela operação de leitura.

Memória RAM

- RAM (random access memory): Memória volátil de leitura e escrita. Usada para armazenar dados por um curto intervalo de tempo.
- Os dados podem ser lidos de qualquer endereço em qualquer sequência.
- O tempo de acesso a uma célula de armazenamento independe de sua localização (endereço).
- Dois tipos básicos:
    - SRAM (static RAM)
    - DRAM (dynamic RAM)

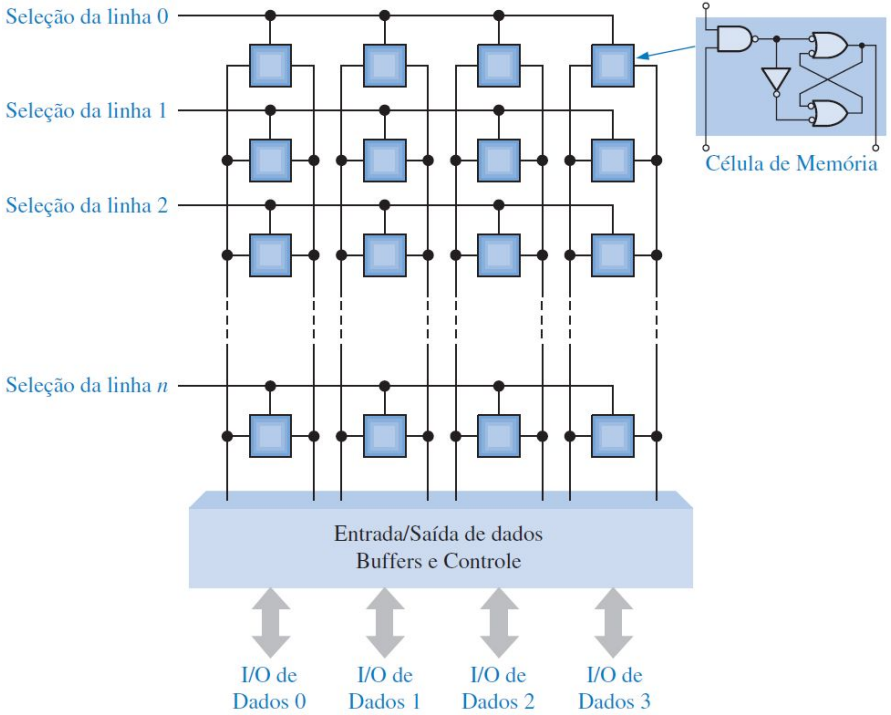
Memória SRAM

- SRAM (static random access memory): as células de memória são essencialmente flip-flops.
  - Armazenam dados indefinidamente (enquanto houver tensão de alimentação).
- Célula de memória:



Memória SRAM

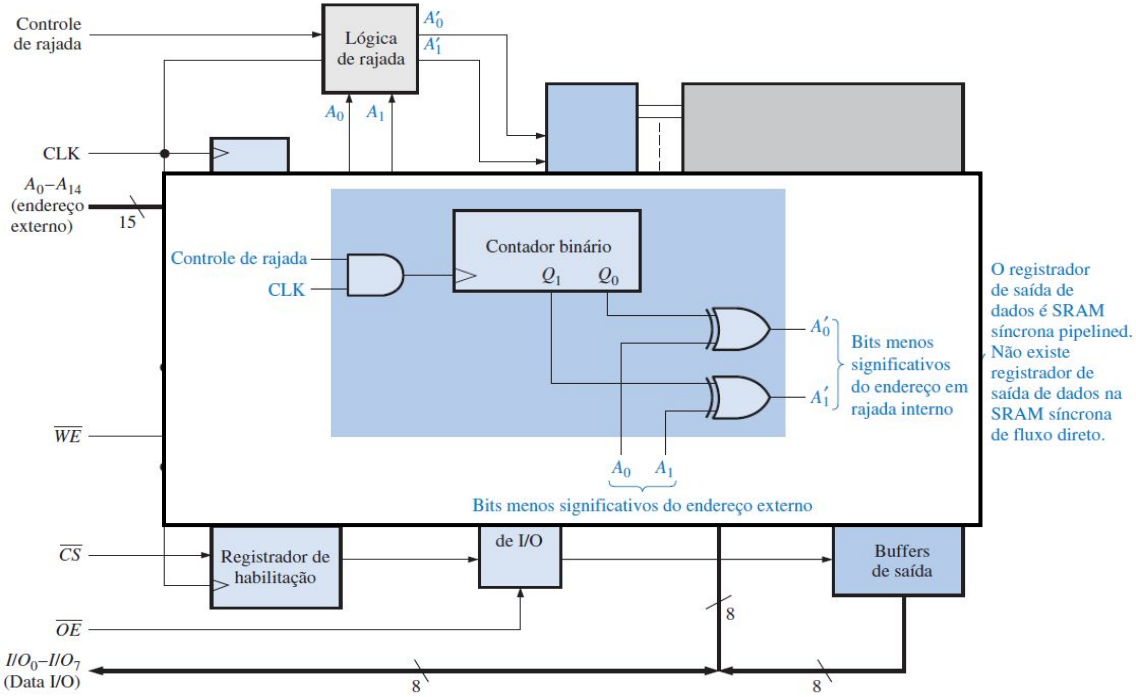
- Arranjo Básico de Células de Memória Estática:



Prof. Denis Fantinato
Profa. Letícia Rittner

Memória SRAM

Organização Básica de uma SRAM Síncrona:



Memória Cache

- Uma das principais aplicações de SRAMs é na memória cache dos computadores.

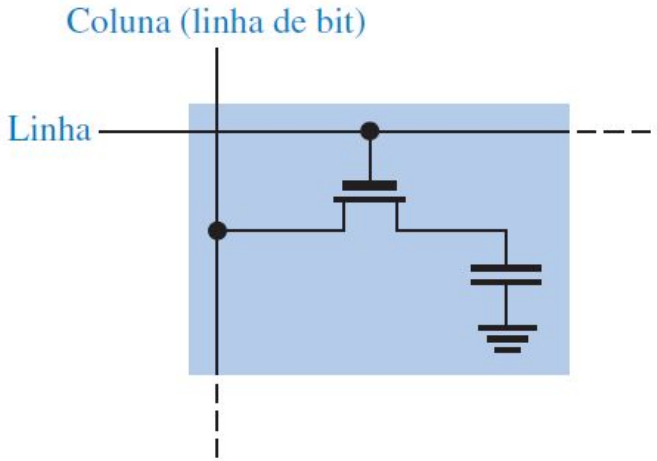
- A memória cache é relativamente pequena, de alta velocidade e que armazena as instruções ou dados mais recentes usados a partir da memória principal que é maior, porém mais lenta.

- A memória cache é uma forma de melhorar o desempenho de um sistema sem recorrer a opção mais custosa de ter toda a memória rápida.

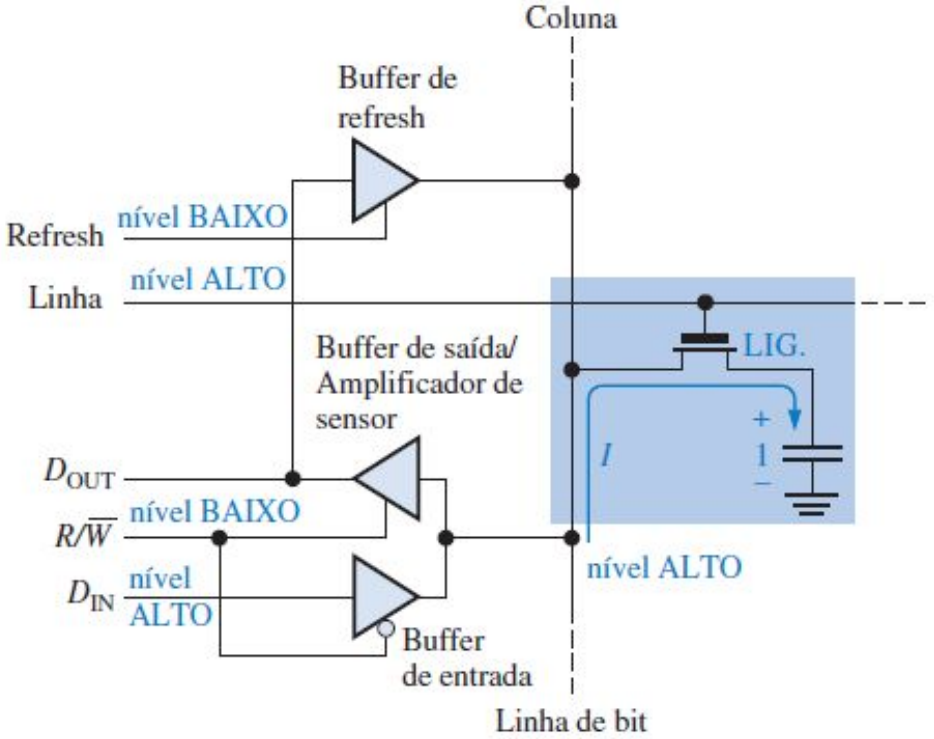
- O conceito de memória cache é baseado na ideia de que os programas de computadores tendem a obter instruções ou dados a partir de uma área da memória principal antes de se mover para uma outra área.

Memória DRAM

- DRAM (*dynamic random access memory*): as células de memória são essencialmente capacitores.
  - Não são capazes de reter dados por longos intervalos de tempo.
  - Os dados precisam ser periodicamente (a cada 2 a 8 ms) regravados (*refresh*).
- Célula DRAM MOS:

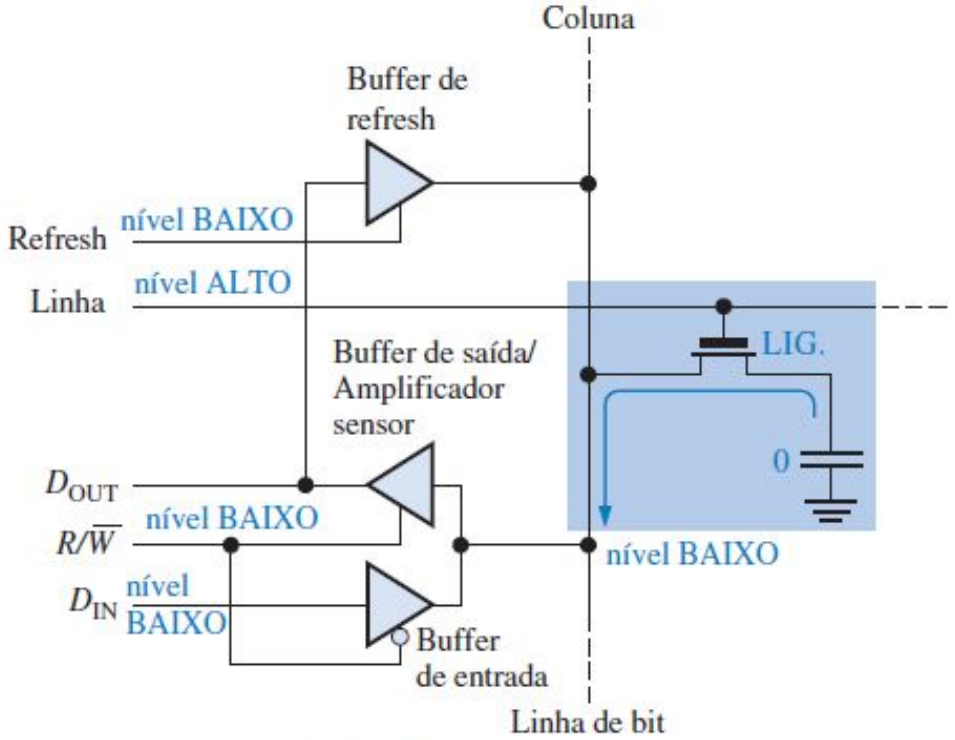


Memória DRAM



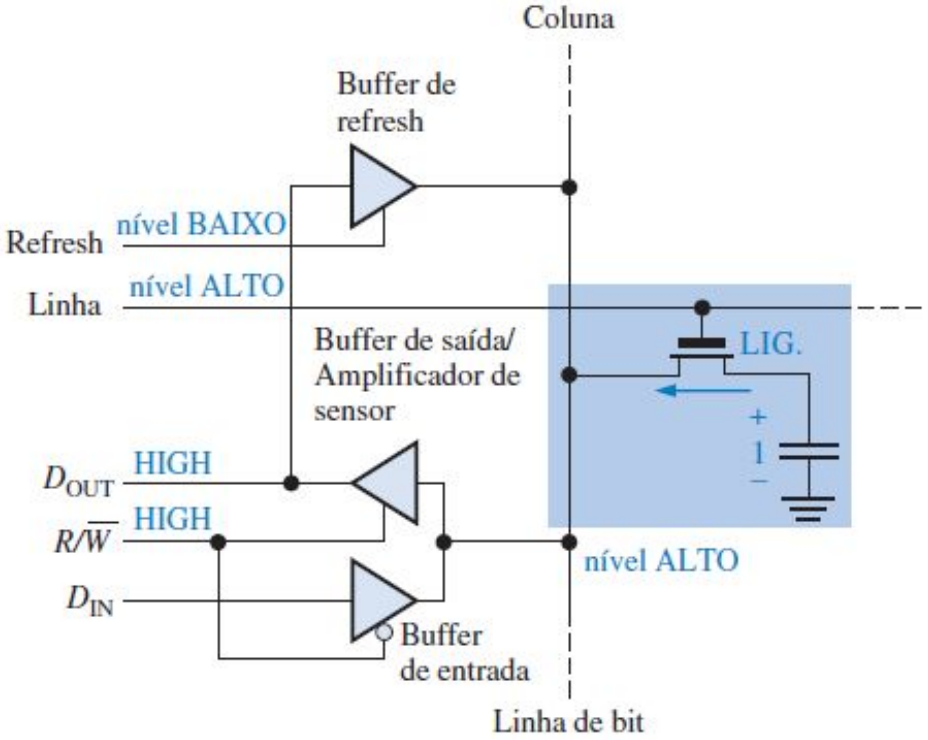
(a) Escrita de um nível 1 na célula de memória

Memória DRAM



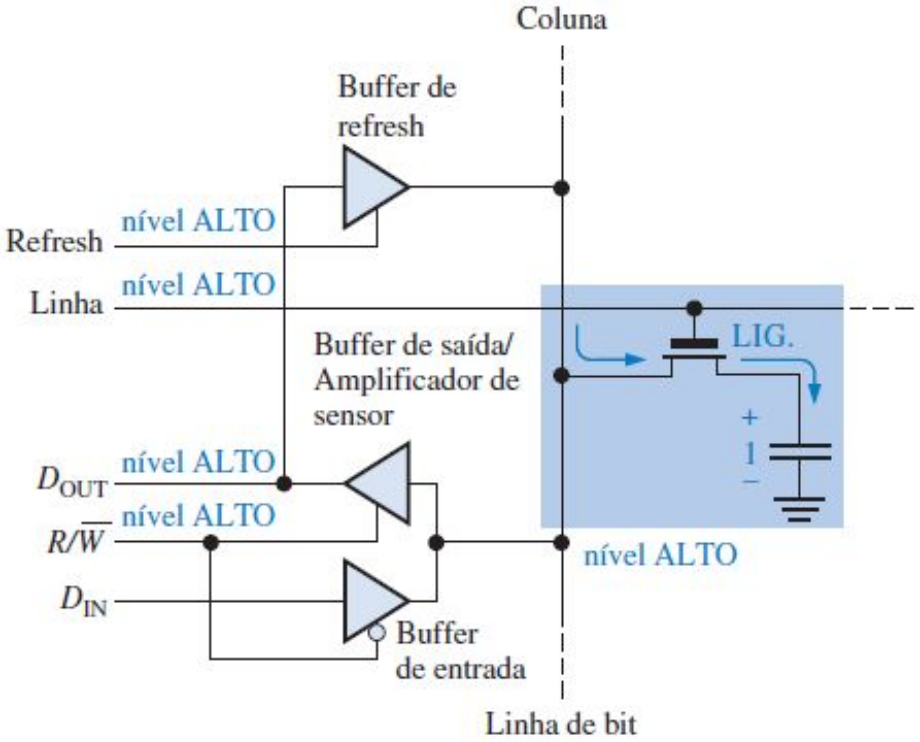
(b) Escrita de um nível 0 na célula de memória

Memória DRAM



(c) Leitura de um nível 1 a partir da célula de memória

Memória DRAM



(d) Refresh de um nível 1 armazenado

Prof. Denis Fantinato
Profa. Letícia Rittner

Memória ROM

- ROM (read only memory): memória não-volátil que permite, durante a operação do sistema, apenas a leitura.

- Tipos de ROM:
  - ROM (read only memory ou ainda mask programmed read only memory)
  - PROM (programmable read only memory)
  - UV-EPROM (ultraviolet erasable programmable read only memory - muitas vezes denominada apenas EPROM erasable programmable read only memory)
  - EEPROM (electrically erasable programmable read only memory)