



IC-UNICAMP

# MC 602

## Circuitos Lógicos e Organização de Computadores

IC/Unicamp

Prof Mario Côrtes

### Capítulo MC9

## Memórias – Implementação e Organização



# Tópicos

- Tipos de memórias
- Organização
- Decodificação de endereço

# Introdução



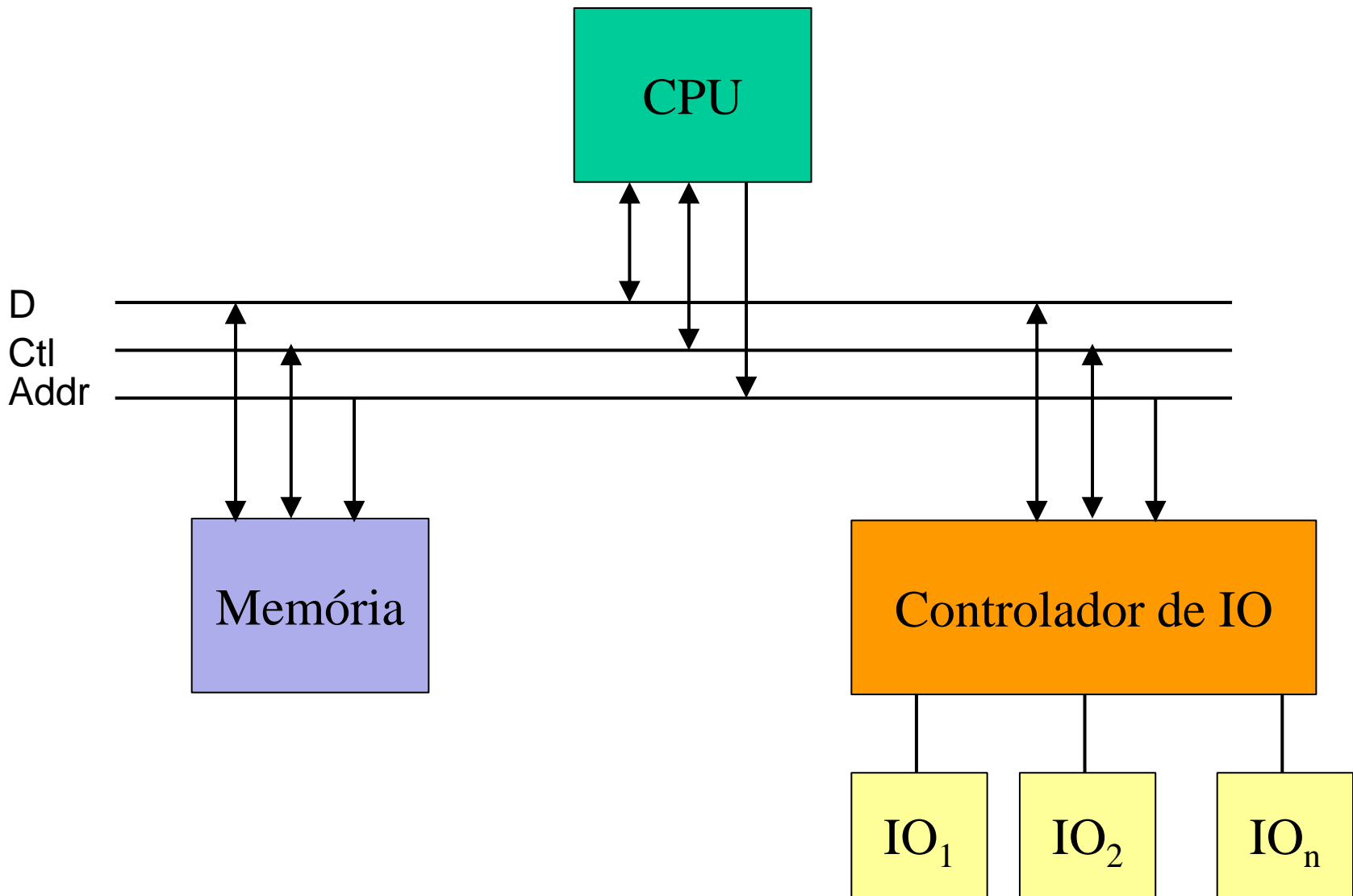
IC-UNICAMP

- Memória: dispositivos capazes de armazenar eficientemente grande quantidade de dados
- Organização: semelhante a uma tabela de dados
  - $n$  linhas, com  $m$  bits cada
- Operações: leitura e escrita

# Sistema de memória: uso típico



IC-UNICAMP



# Organização e dimensões

- Conceitualmente: uma tabela com linhas de dados
- Organizadas como uma matriz (array) de duas dimensões de células de bits
  - Cada célula armazena um bit
- No exemplo
  - 18 linhas de dados
  - palavras de 8 bits

**8 bits**

0	0	1	0	1	1	0	0
1	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1
1	1	0	0	1	0	1	0
0	0	1	0	1	1	0	0
1	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1
1	1	0	0	1	0	1	0
0	0	1	0	1	1	0	0
1	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1
1	1	0	0	1	0	1	0
0	0	1	0	1	1	0	0
1	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1
1	1	0	0	1	0	1	0

**16 linhas**

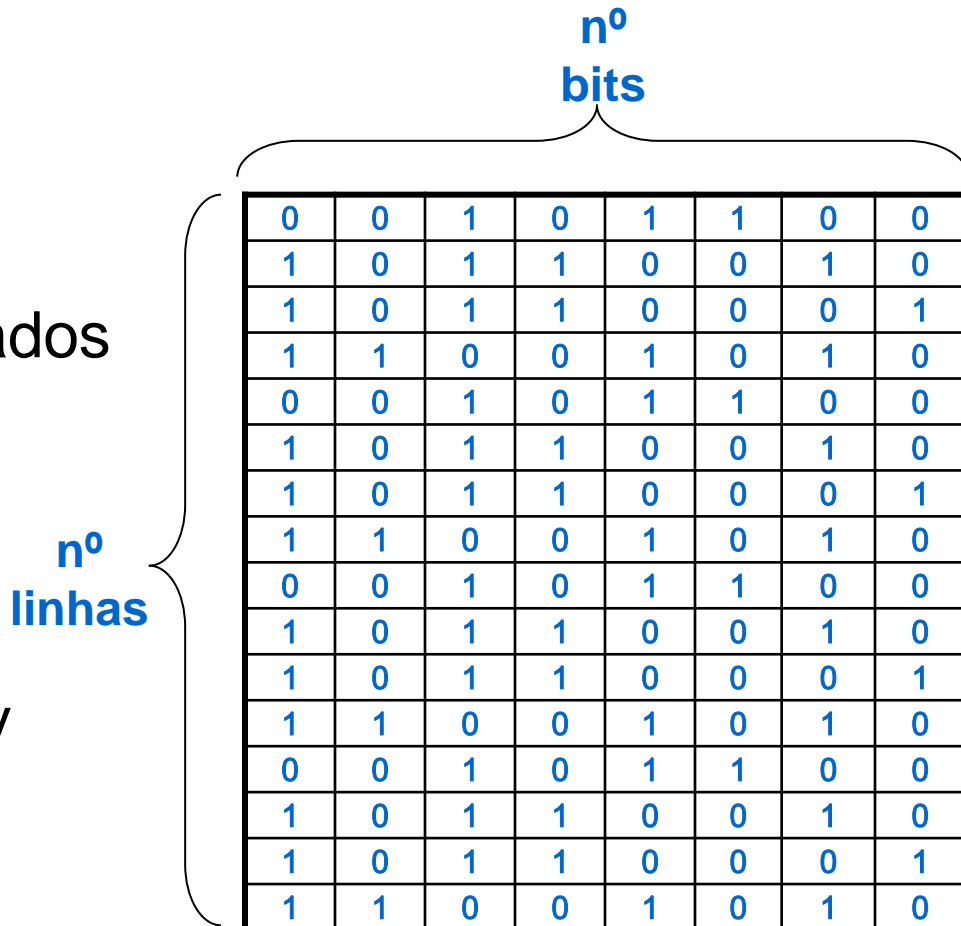


# Organização e dimensões

- Largura (width):
  - n° de colunas no array
  - = n° de bits na linha de dados
  - = word size

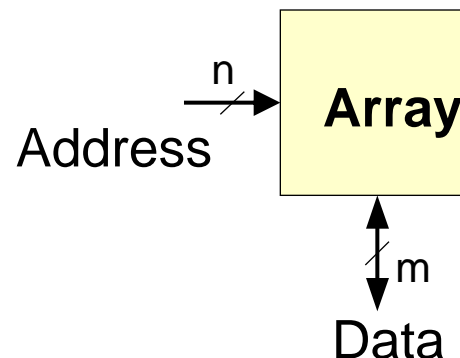
- Profundidade (Depth):
  - número de linhas do array

- Tamanho do array
  - largura x profundidade
  - = (n° de linhas) \* (bits/linha)
  - = (n° de linhas) \* (word size)



# Organização, entradas, saídas

- Entradas
  - Endereço:  $n$  bits selecionam  $2^n$  linhas
  - Dados (bidirecional):  $m$  bits de dados de escrita ou leitura
  - Controle: WR, RD, OutputEnable
- Tamanho da memória
  - $2^n * m$  bits
  - Exemplo: se  $m = 8$  (1 Byte) e  $n = 10$ 
    - 1024 linhas (1K) e 8 colunas
    - tamanho da memória = 1 KB
    - ou 1K x 1B
    - ou 8 Kb

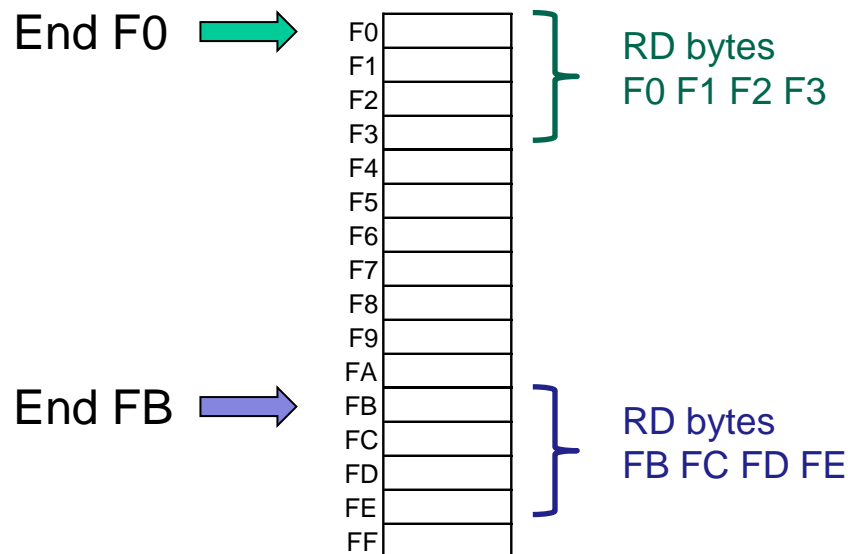




# Endereçamento

- Primeiro caso: endereçamento acessa uma linha somente
  - Exmpl1: endereçamento → byte. Endereço aponta para byte. Largura da memória = 1 Byte
  - Exmpl2: endereçamento → palavra de 32 bits. Largura da memória = 4 Bytes

- Segundo caso: endereçamento a byte mas Read tem output de 4 bytes
  - Aplicação: dados e instruções





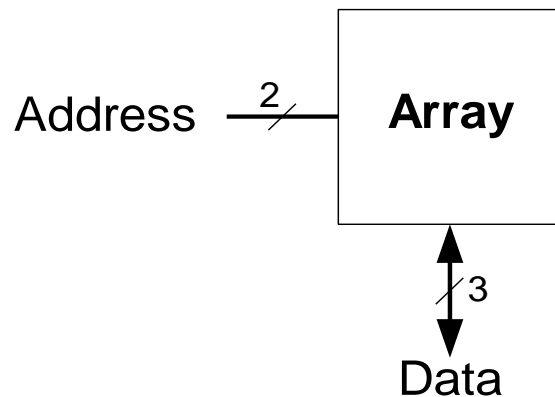


# Memórias

- Principais tipos memórias:
  - Memória somente de leitura – Read only memory (ROM)
  - Memórias de leitura e escrita – Random Access Memory (RAM)
    - Memórias dinâmicas – Dynamic random access memory (DRAM)
    - Memórias estáticas – Static random access memory (SRAM)
- Um dado de valor de  $M$ -bit pode ser lido ou escrito por vez em um endereço de  $N$ -bit.

# Memória : Exemplo

- Array de  $2^2 \times 3$ -bit
- Word size de 3-bits

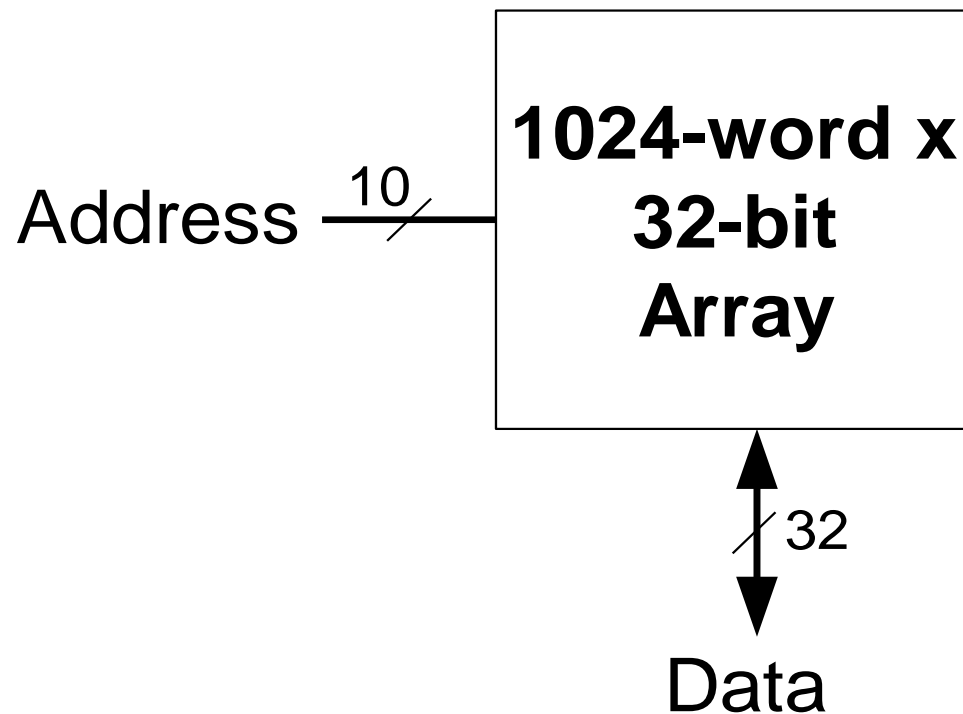


Address	Data		
11	0	1	0
10	1	0	0
01	1	1	0
00	0	1	1

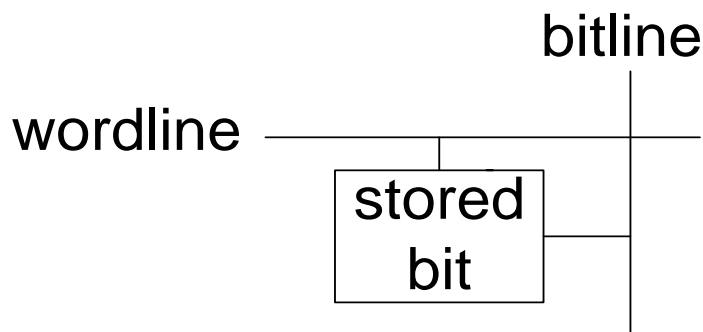
Diagram illustrating the memory array structure. The address is shown on the left, and the data is shown in a 3-bit word. The depth of the array is indicated by a vertical double-headed arrow, and the width is indicated by a horizontal double-headed arrow.

# Memória : Exemplo

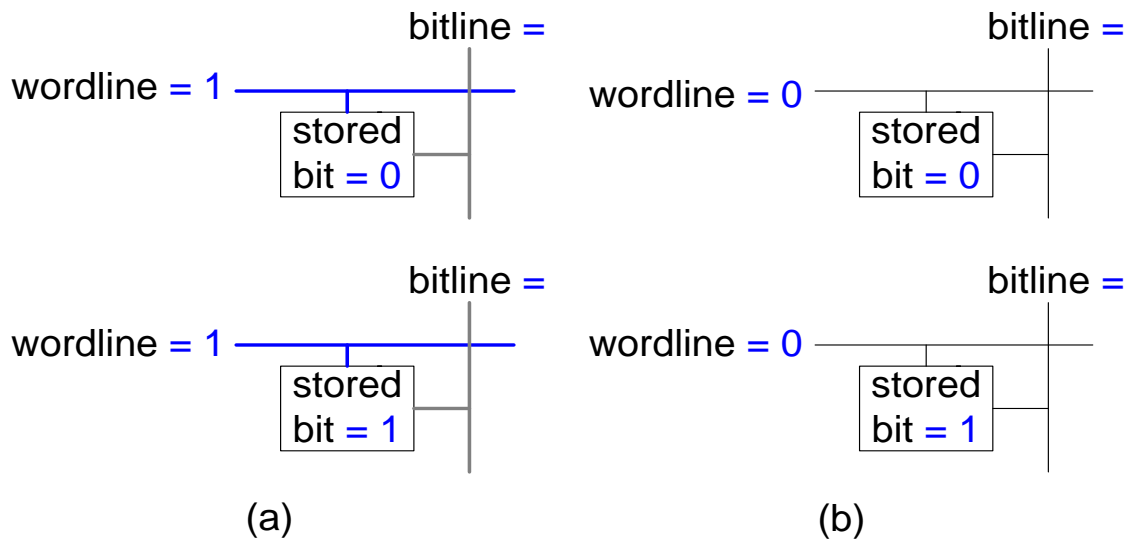
- N<sup>o</sup> de linhas =  $2^{10} = 1024 = 1K$
- N<sup>o</sup> de colunas = word size = 32 bits = 4B
- Tamanho
  - 1K x 4B
  - ou 4KB
  - ou 32 Kb



# Memória: Célula de bit



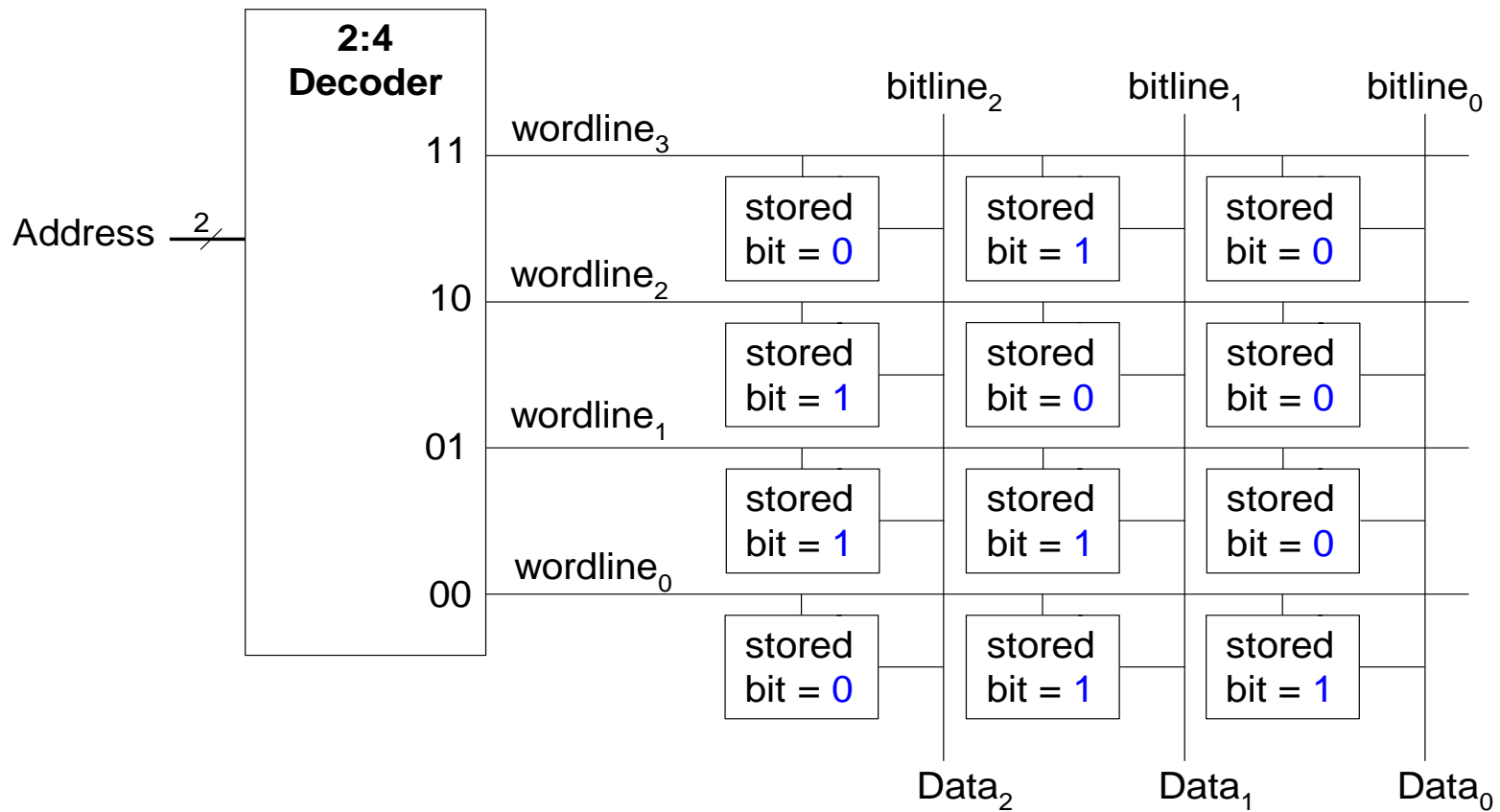
## Exemplo



- Procedimento para leitura
  - Endereço seleciona (decodificador) 1 linha (1 wordline)
  - Cada célula selecionada na wordline aciona o bitline, levando o valor para a saída
- Procedimento para escrita
  - Endereço seleciona (decodificador) 1 linha (1 wordline)
  - Valor a ser escrito colocado na bitline (bidirecional)
  - Sinal de controle WR ativa a escrita do valor do bitline na célula



# Memória: 4x3





# Tipos de Memórias

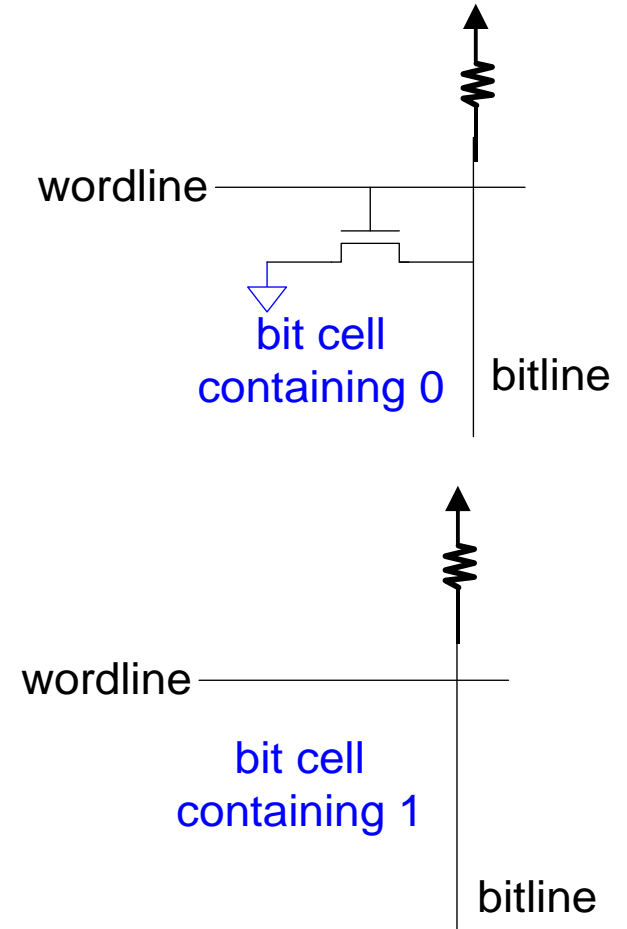
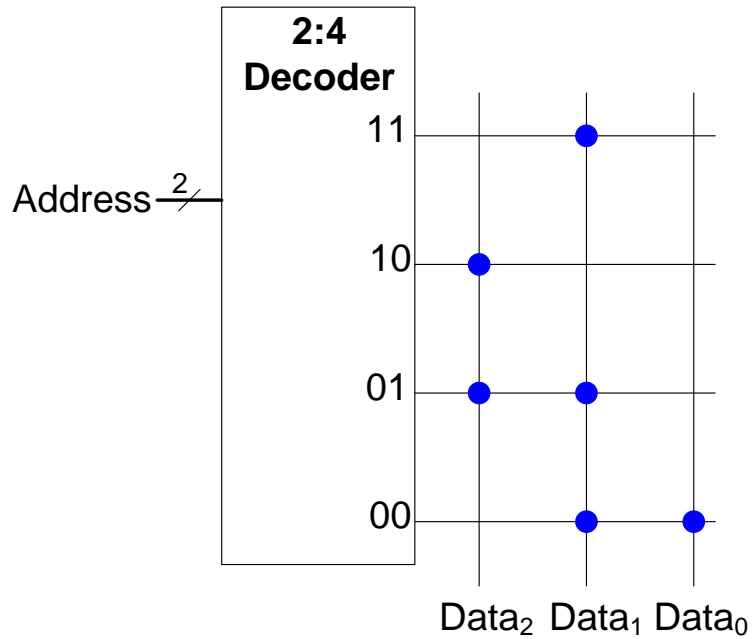
- Read only memory (ROM): não volátil
- Random access memory (RAM): volátil

# ROM

- Read only memory (ROM)
  - Não volátil: não perdem seus dados quando a alimentação é desligada
  - Pode ser lida rapidamente, porém a escrita é lenta (no caso das ROMs reprogramáveis)
  - Memórias em câmeras digitais, pen drives são ROMs
  - Historicamente denominadas de *read only memory* porque as primeiras ROMs eram fabricadas já com os dados ou escritas posteriormente queimando-se fusíveis → somente leitura

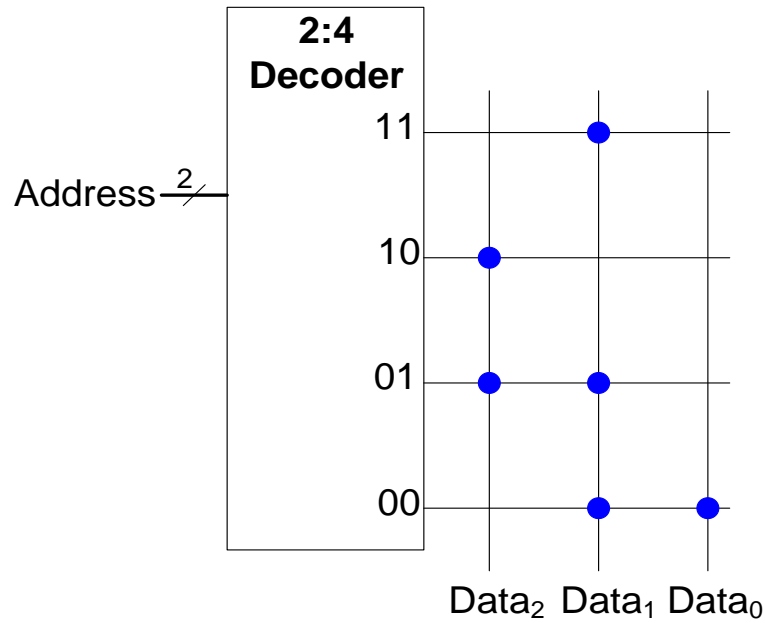


# ROM





# ROM



Address	Data		
11	0	1	0
10	1	0	0
01	1	1	0
00	0	1	1

depth

width

# Detalhes da ROM



IC-UNICAMP

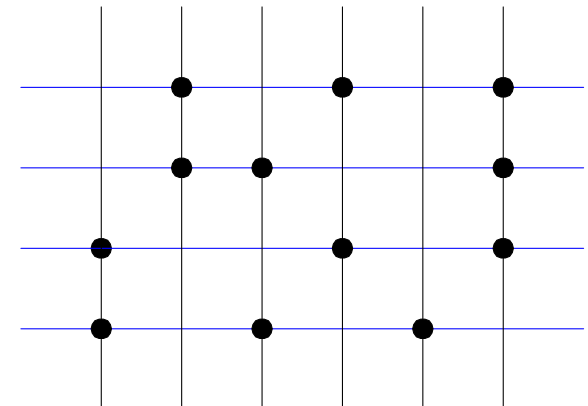
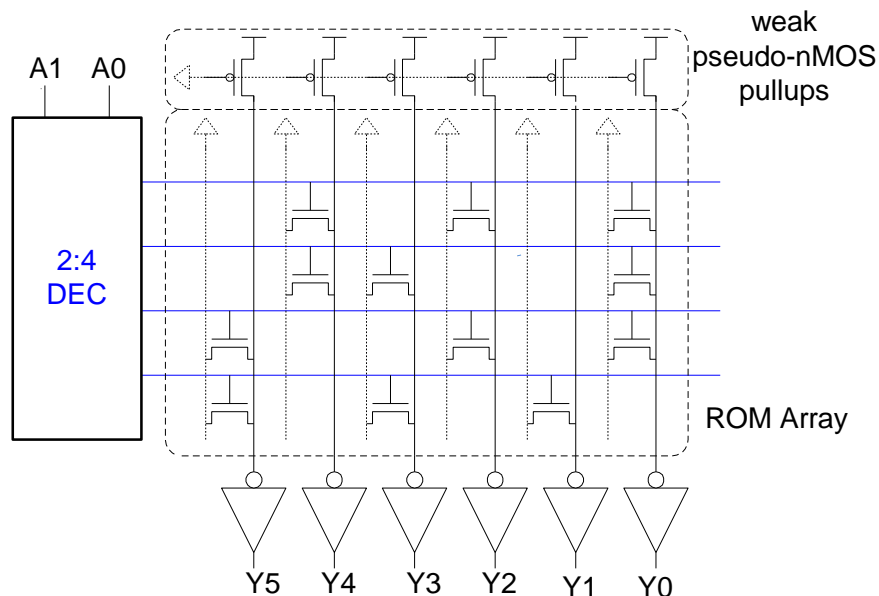
- 4-word x 6-bit ROM
  - Representada por diagrama de pontos
  - Pontos indicam 1's na ROM

Word 0: **010101**

Word 1: **011001**

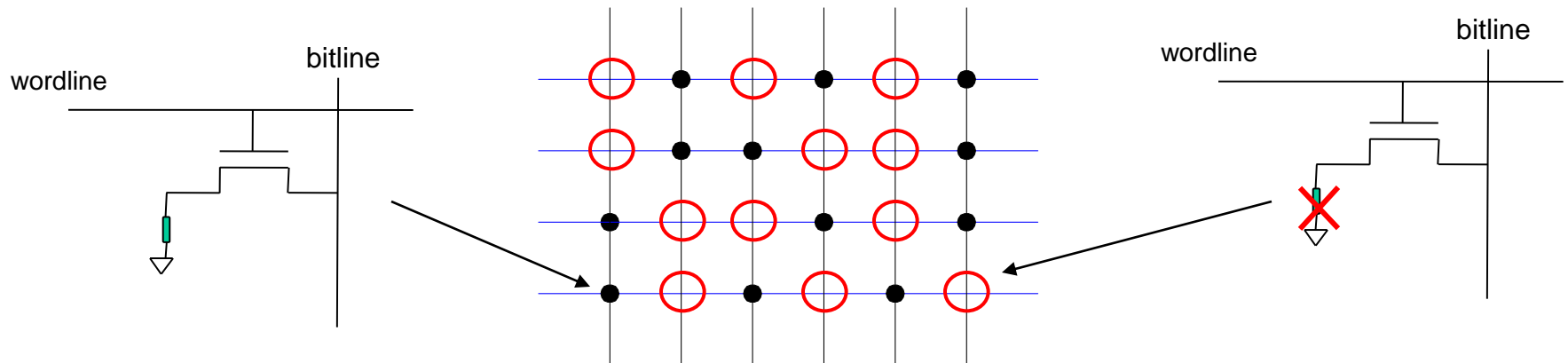
Word 2: **100101**

Word 3: **101010**



# ROM Programável (PROM)

- Arquitetura semelhante à ROM
- Chip é uma matriz de transistores completa
- Fusíveis selecionados são queimados após fabricação para desconectar transistores (resulta no bit zero)

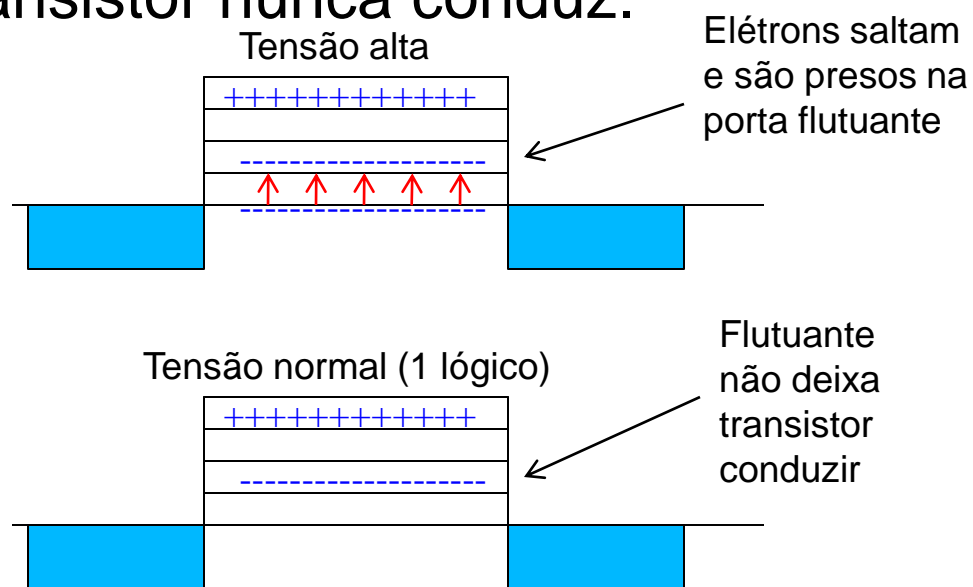


# ROM Programável Apagável

- EPROM(Erasable), EEPROM (Electrically Erasable) e Flash
  - Usam um transistor com mais uma porta (“flutuante”)
  - Uma tensão elevada na porta normal injeta elétrons na porta “porta flutuante”
  - Elétrons na “porta flutuante” bloqueiam tensão da porta normal, e o transistor nunca conduz.

## Remoção dos elétrons

- EPROM: por ultravioleta
- EEPROM: por tensão reversa
- Flash: por tensão reversa



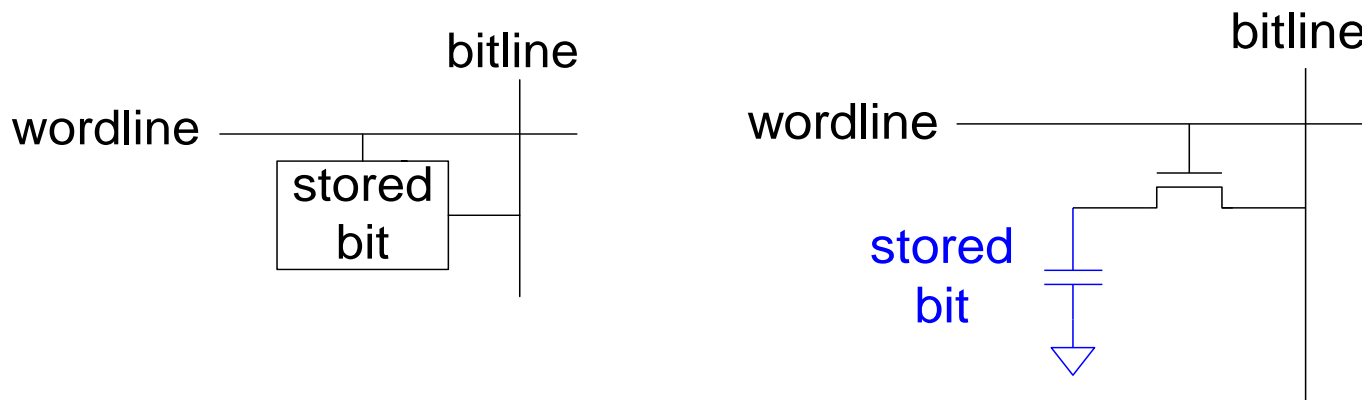
- Random access memory
  - Volátil: perde o dado quando a alimentação é desligada
  - Pode ser lida ou escrita rapidamente
  - A memória principal do seu computador é RAM
    - Memória principal DRAM
    - Memória cache SRAM
  - Historicamente denominada de *random access memory* porque qualquer palavra de dado pode ser acessada como qualque outra (em contraste com sequential access memories como fita magnética).

# Tipos de RAM

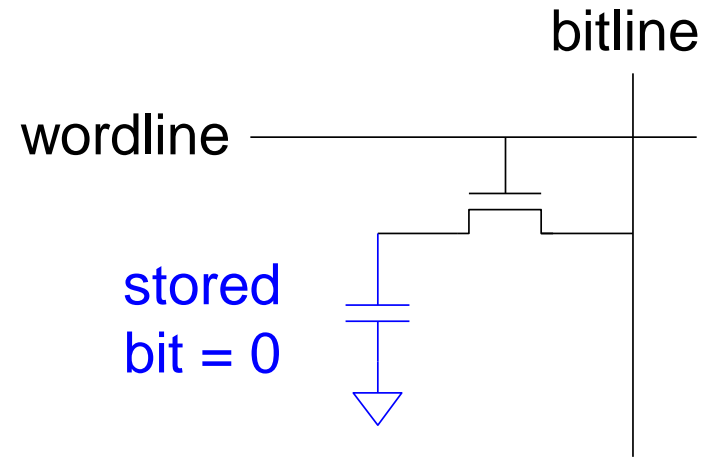
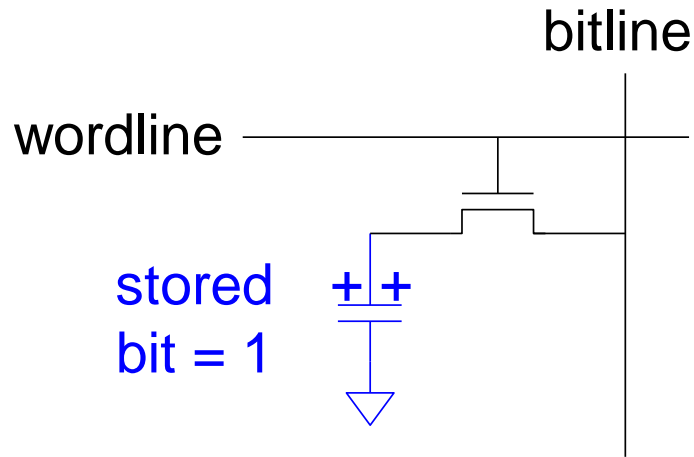
- Tipos de RAM
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)
- Diferença: modo de armazenar dados
  - DRAM usa um capacitor
    - devido às correntes de fuga, carga armazenada vai se perdendo e precisa ser refrescada (refresh) periodicamente
  - SRAM usa cross-coupled inverters (“latch”)
    - estado é mantido estável (sem degradação) desde que a alimentação esteja ligada → não precisa de refresh

# DRAM

- Data bits são armazenados em um capacitor
- DRAM denominado de *dynamic* porque os valores necessitam ser reescritos (refreshed) periodicamente e após serem lidos por que:
  - A corrente de fuga do capacitor degrada o valor
  - A leitura destrói o valor armazenado



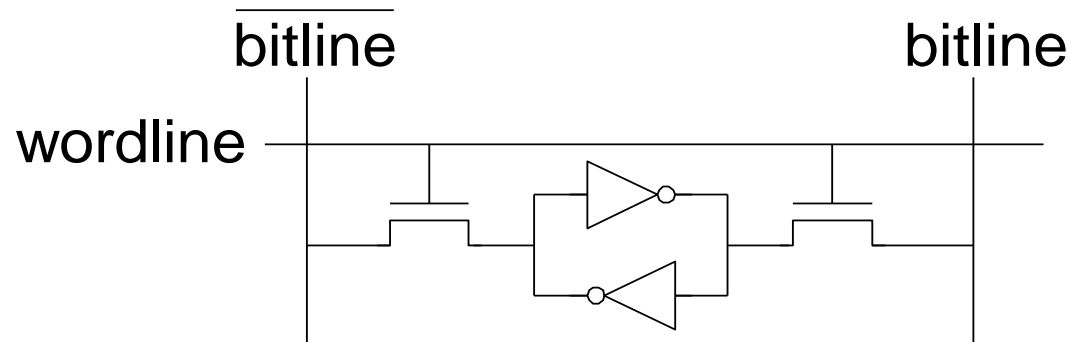
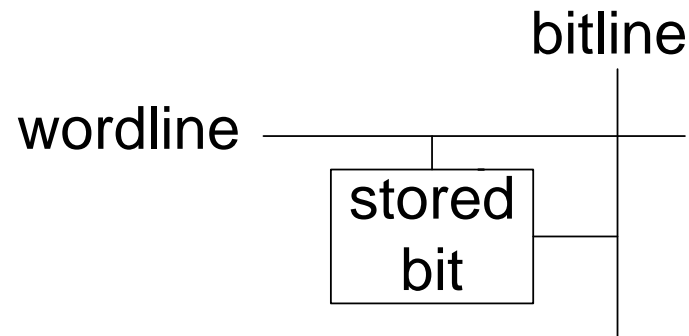
# DRAM



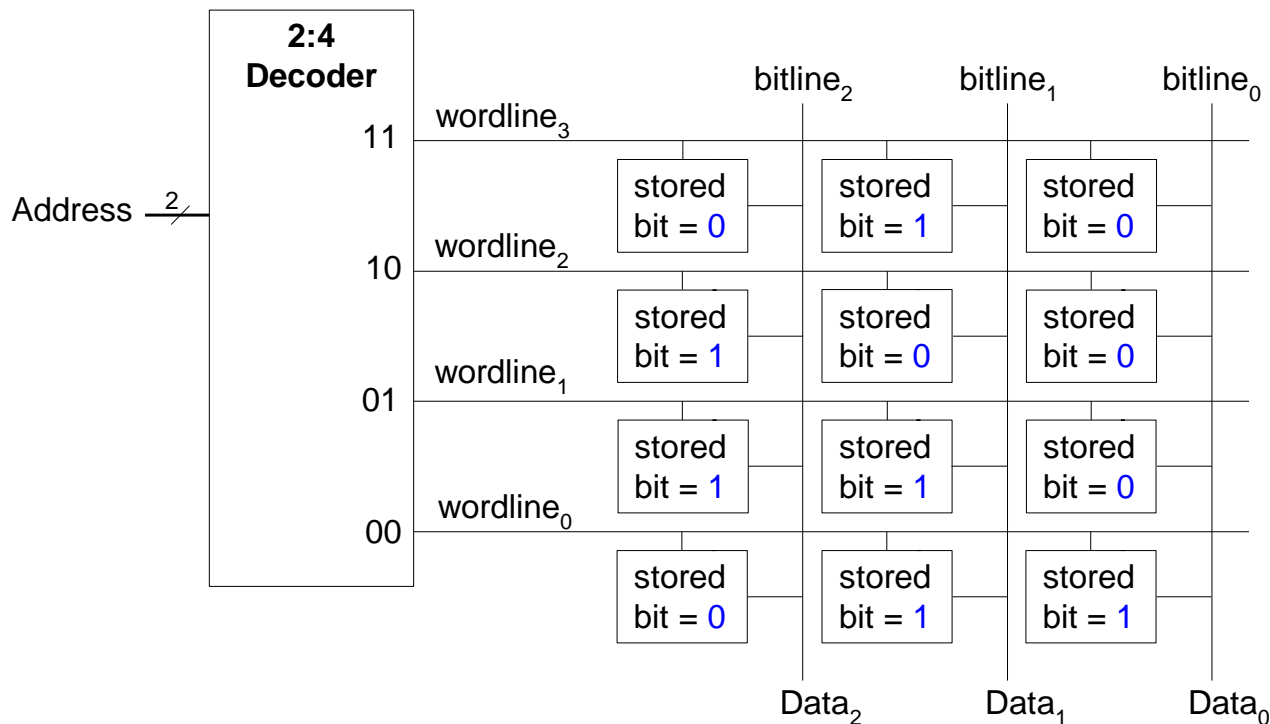


# SRAM

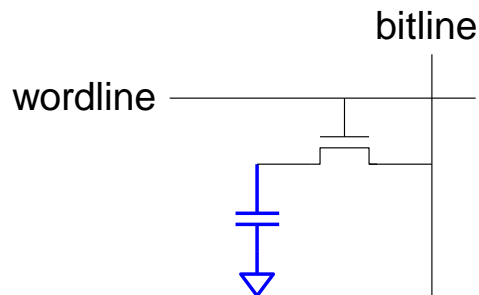
- Estática: o sinal armazenado não se degrada



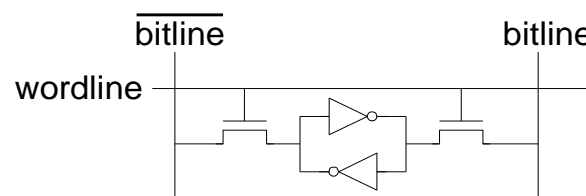
# Dados armazenados



## célula dinâmica

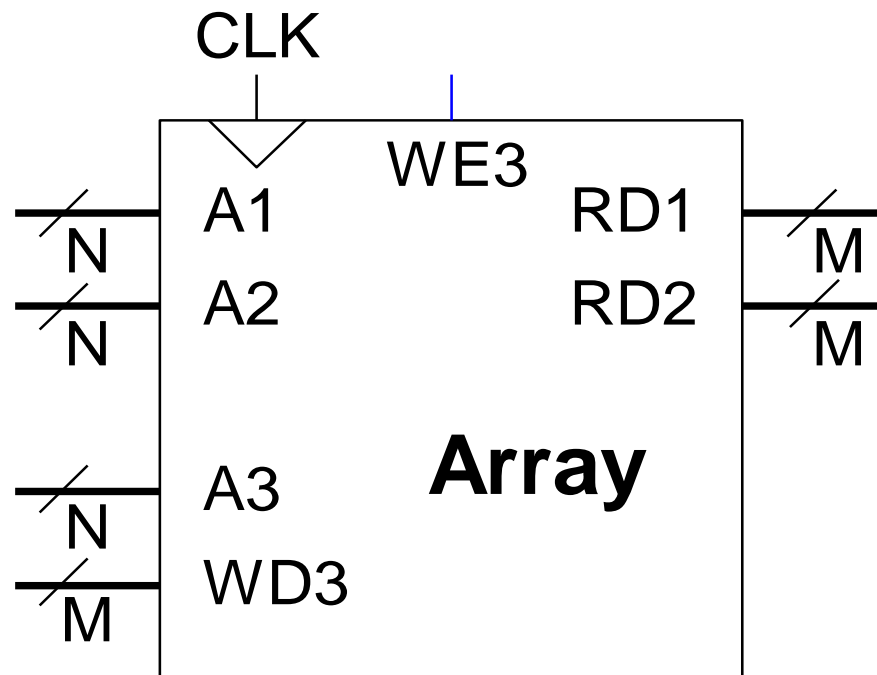


## célula estática



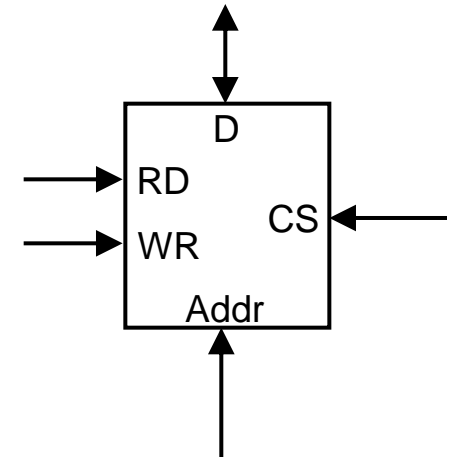
# Memórias Multi-Portas

- Porta: par endereço/dado (address/data)
- Memória 3-portas
  - 2 portas de leitura (A1/RD1, A2/RD2)
  - 1 porta de escrita (A3/WD3, WE3 enables writing)

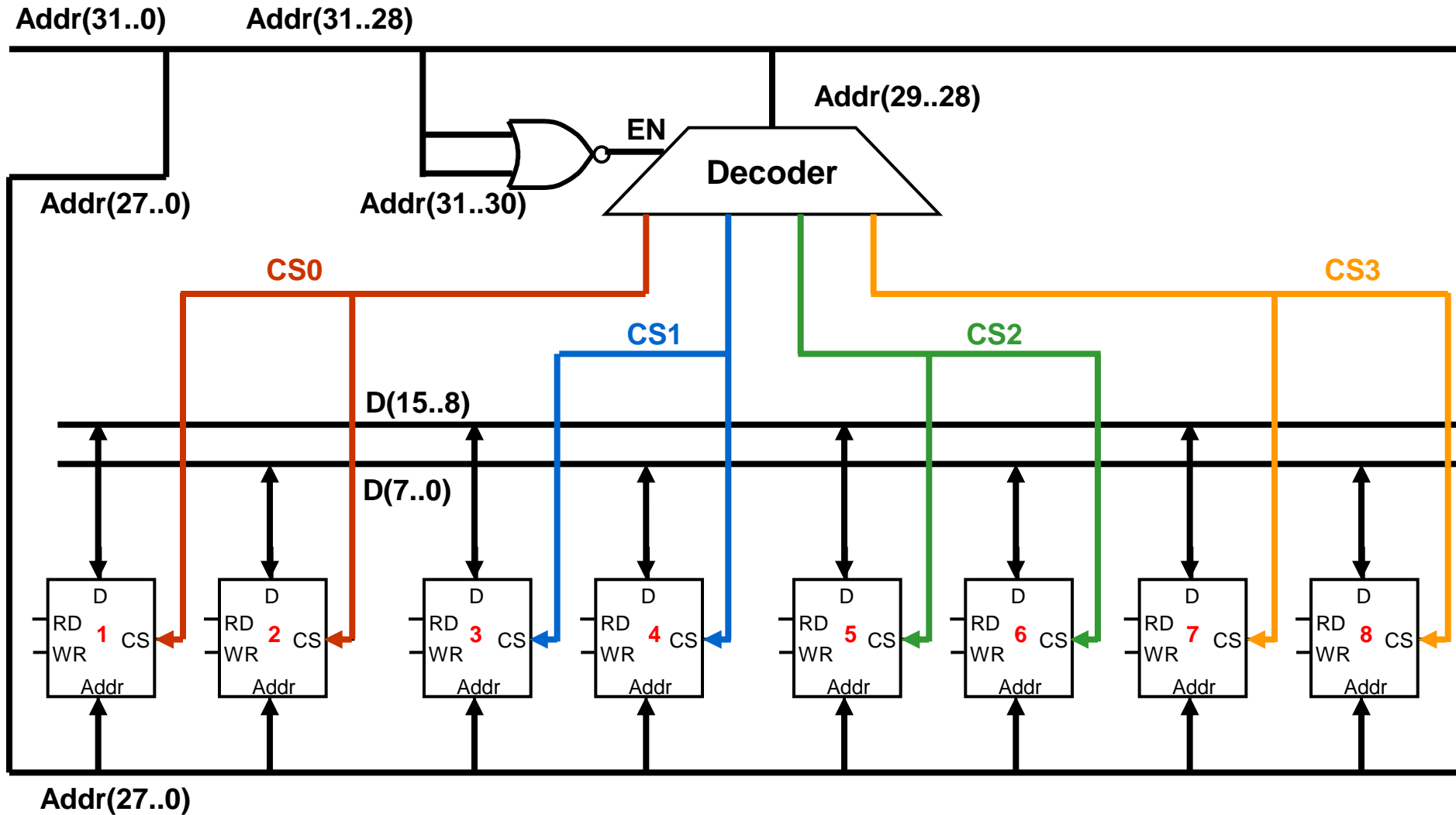


# Organização de um sistema de memória

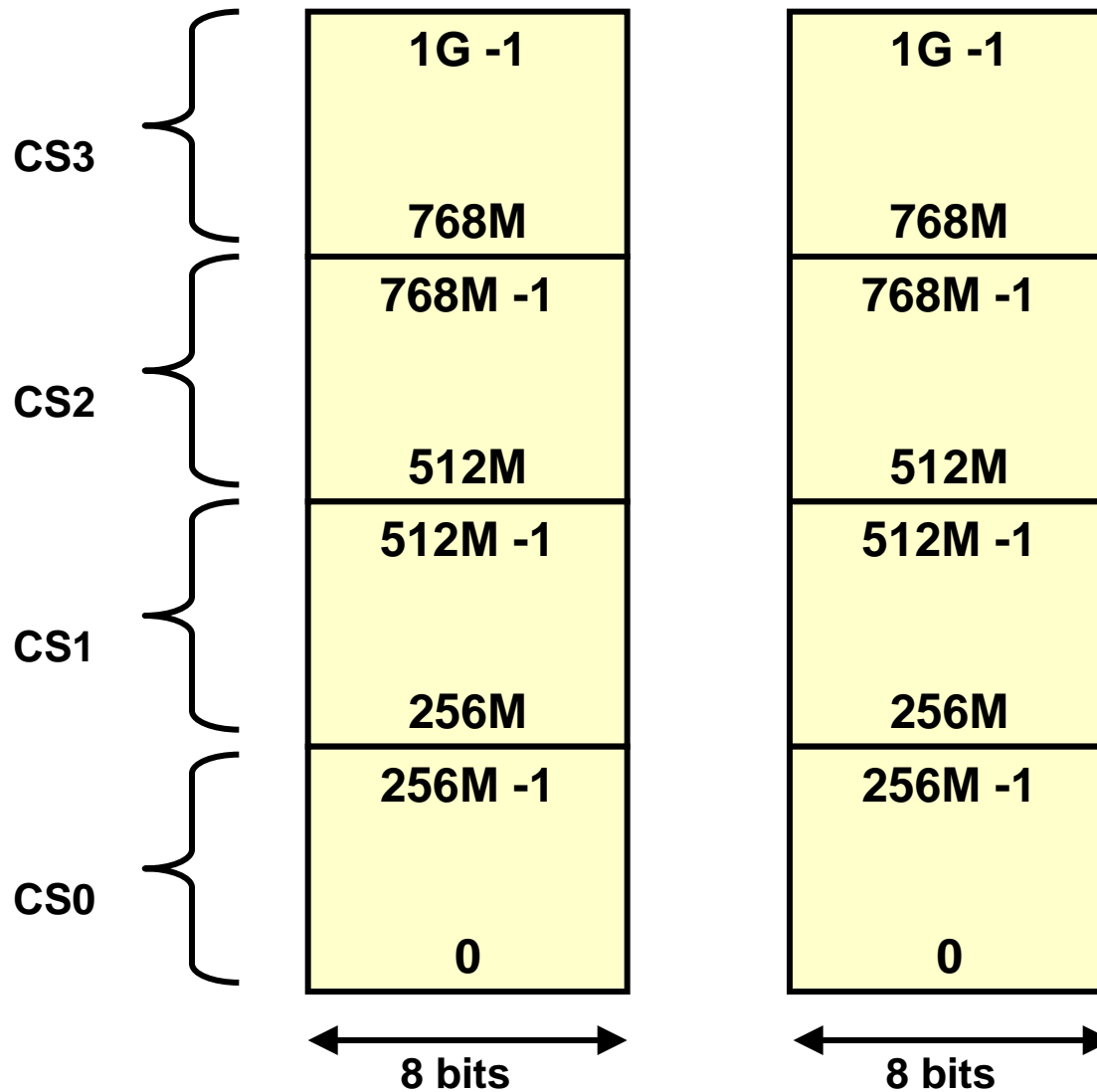
- Sistema de memória: composto por vários chips (ou pentes)
  - normalmente, em número menor do que a capacidade máxima
- Exemplo:
  - chip de memória 256MB (256Mx8b)
    - pinos relevantes
      - Data (InOut): 8b
      - Address (In): 28b
      - Entradas de controle:
        - » WR, RD, ChipSelect: 1b cada
    - Sistema: 1 G Words de 16 bits
      - serão necessários 8 chips
      - mas barramento de endereços tem Address[31..0]



# Decodificador de endereços



# Espaço de endereçamento utilizado



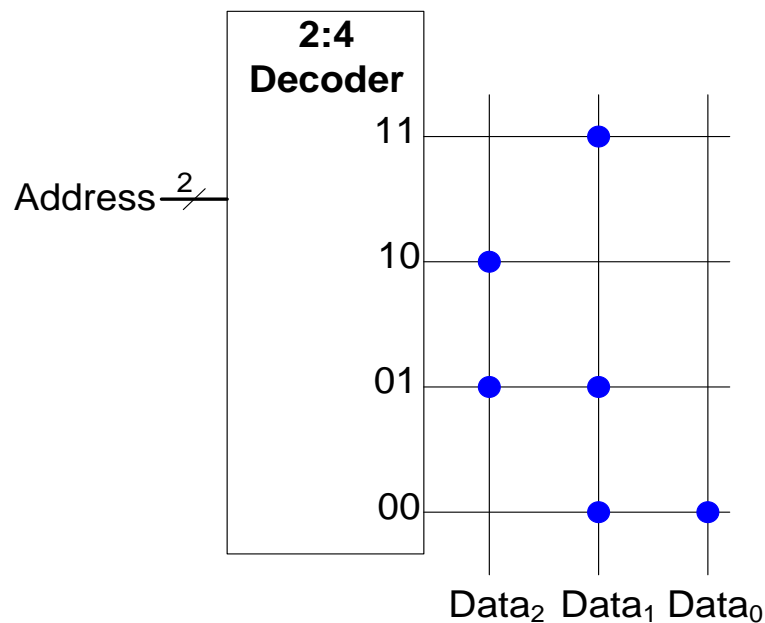
# Espaço de endereçamento completo

Address[31..28]

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
ch 1 2	ch 3 4	ch 5 6	ch 7 8	inv	inv	inv	inv	inv	inv	inv	inv	inv	inv	inv	inv

ch i j = chips

# Lógica com ROM



$$Data_2 = \overline{A_1 \oplus A_0}$$

$$Data_1 = \overline{A_1 \cdot \overline{A_0}}$$

$$Data_0 = \overline{A_1} \cdot \overline{A_0}$$



# Lógica com ROM

## Exemplo



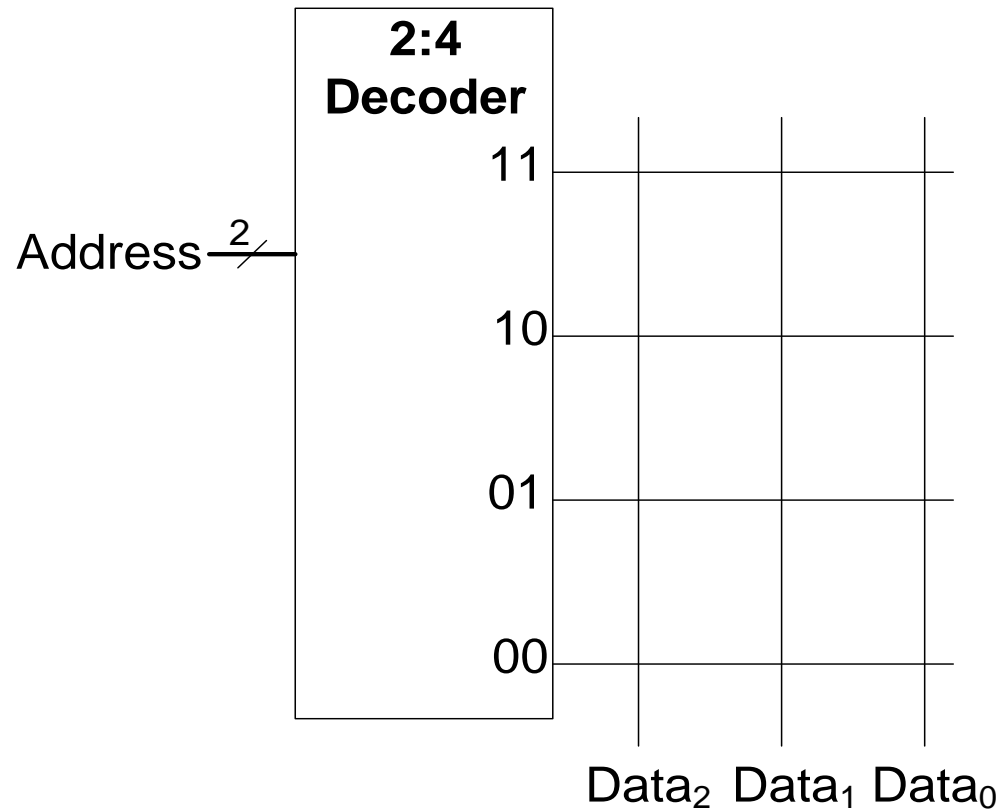
IC-UNICAMP

- Implemente as seguintes funções lógicas usando uma ROM  $2^2 \times 3$ -bit:

$$X = AB$$

$$Y = A + B$$

$$Z = A\bar{B}$$



# Lógica com ROM

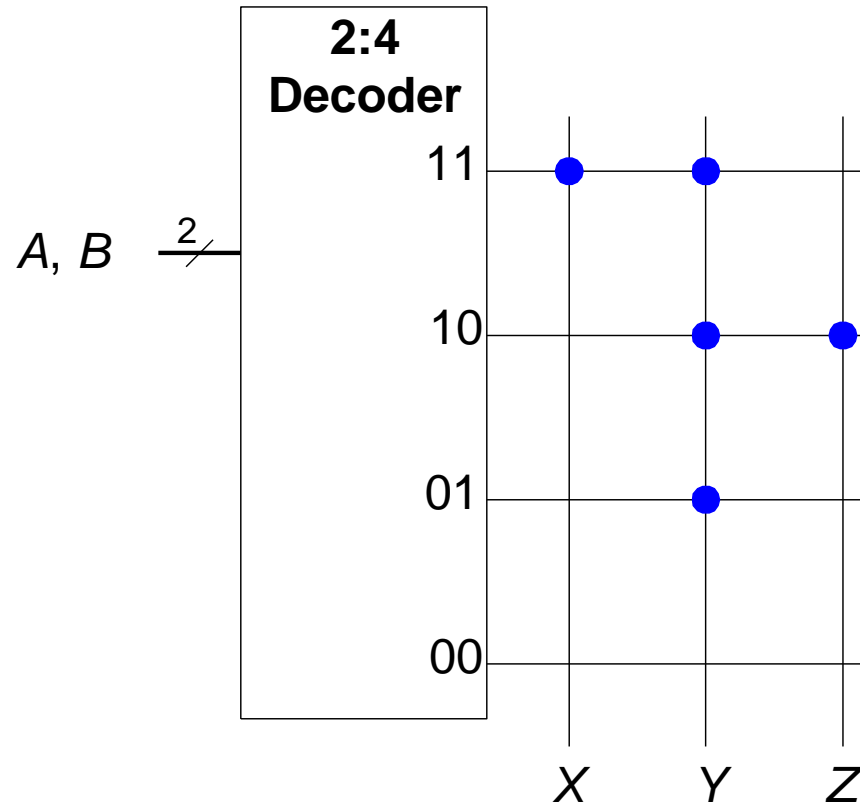
## Exemplo

- Implemente as seguintes funções lógicas usando uma ROM  $2^2 \times 3$ -bit:

$$X = AB$$

$$Y = A + B$$

$$Z = A\overline{B}$$



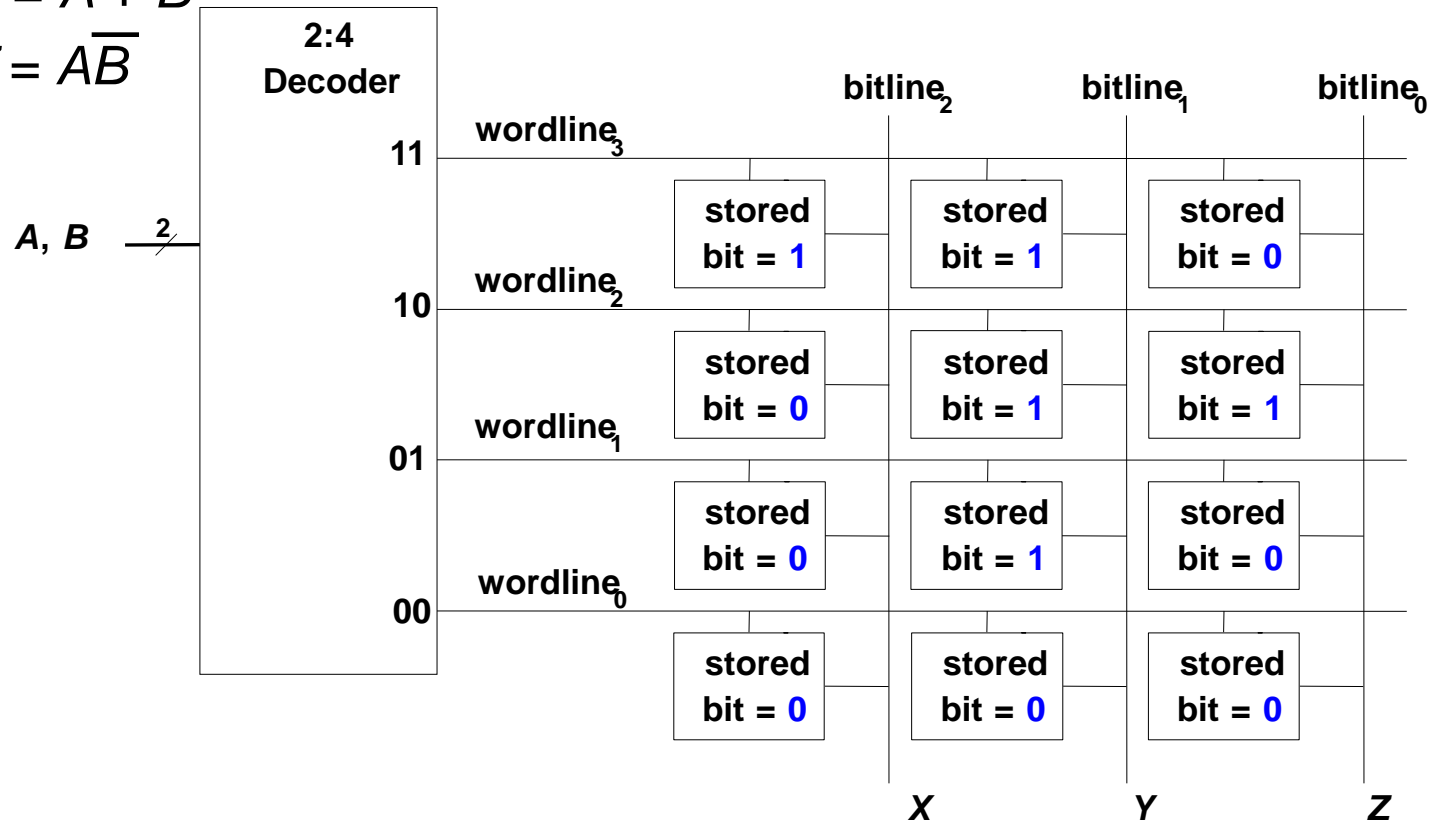
# Lógica com Memória

- Implemente as seguintes funções lógicas com uma memória  $2^2 \times 3$ -bit:

$$X = AB$$

$$Y = A + B$$

$$Z = A\overline{B}$$



# Lógica com LUT (Look Up Table)

Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

- Memórias usadas para executar funções lógicas são denominadas lookup tables (LUT)
- O usuário tem o valor de saída para cada combinação das entradas (address)

