IC-UNICAMP

MC 613

IC/Unicamp

Prof Guido Araújo

Prof Mario Côrtes

Memória

IC-UNICAMP

Tópicos

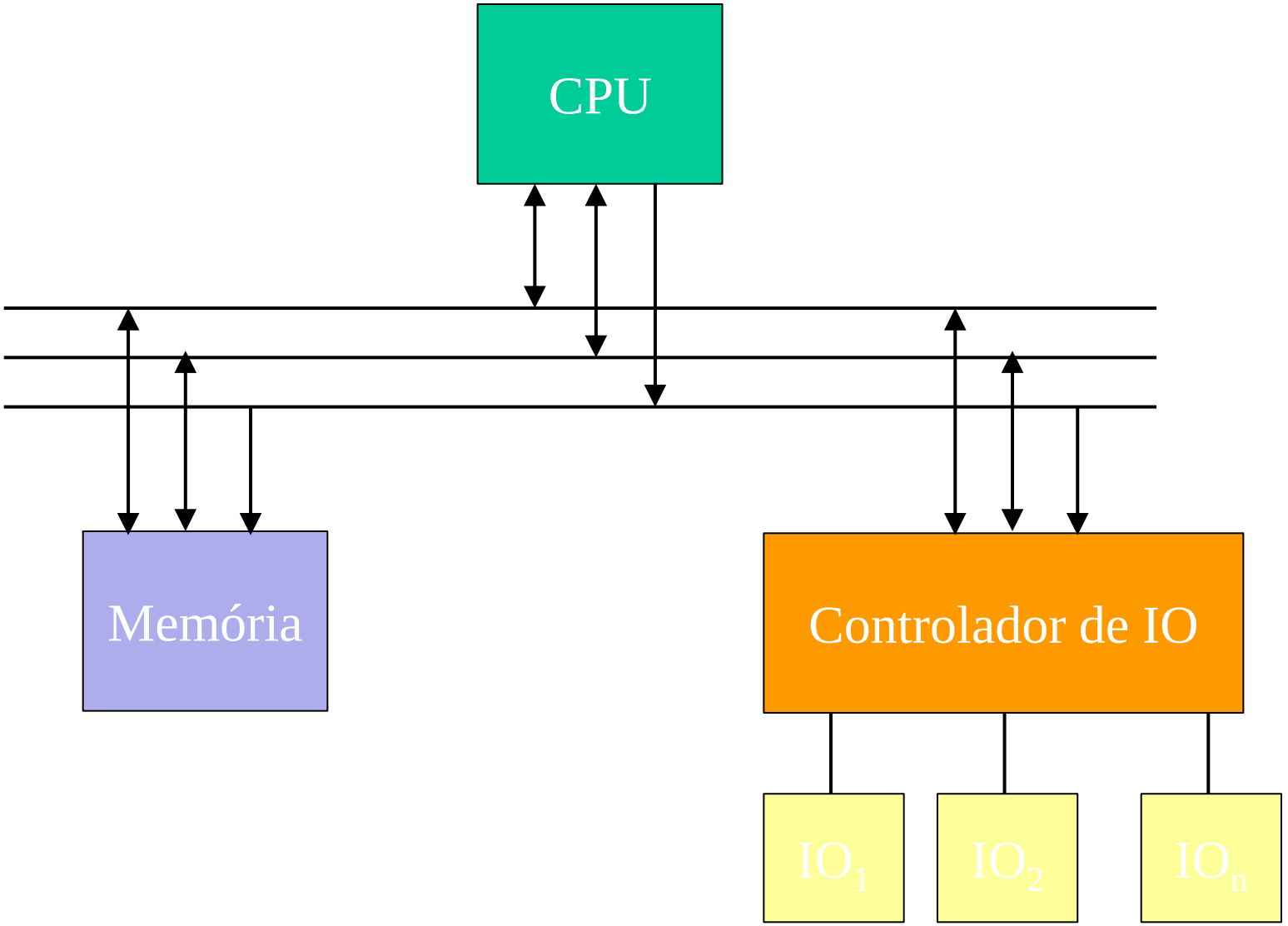
- Tipos de memórias
- Organização
- Decodificação de endereço
- Memórias em VHDL
- Usando memórias na DE1

IC-UNICAMP

Introdução

- 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



Organização e dimensões

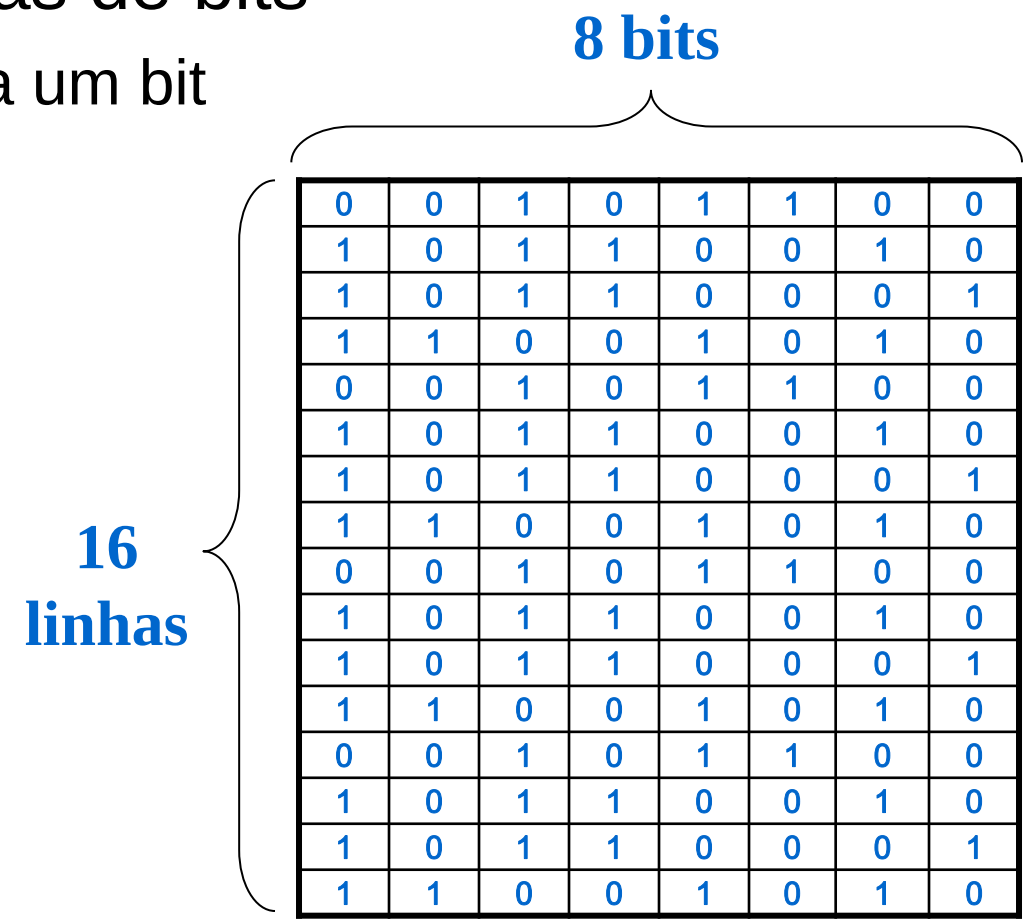
Conceitualmente: uma tabela com linhas de dados

Organizadas como uma matriz (array) de duas dimensões de células de bits

- Cada célula armazena um bit

No exemplo

- 16 linhas de dados
- palavras de 8 bits



Organização e dimensões

Largura (width):

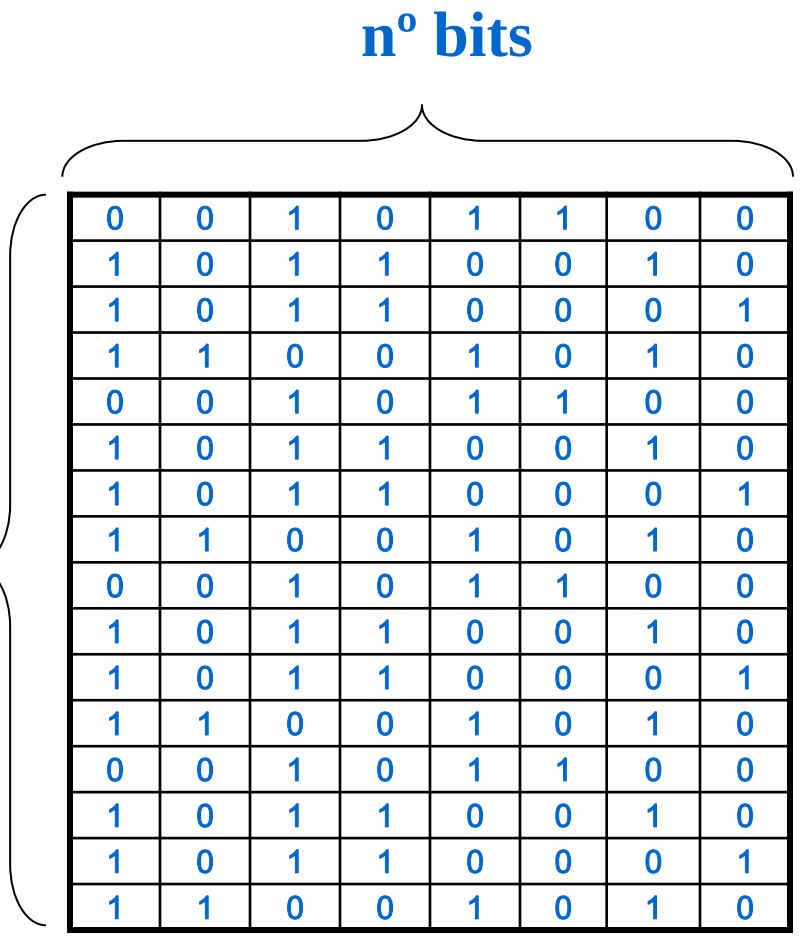
nº de colunas no array
= nº de bits na linha de dados
= word size

Profundidade (Depth):

número de linhas do array

Tamanho do array

largura x profundidade
= (nº de linhas) * (bits/linha)
= (nº de linhas) * (word size)



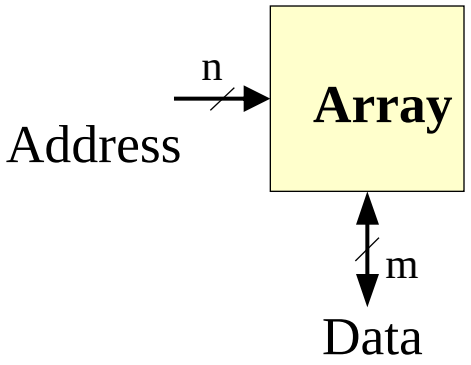
Organização, entradas, saídas

Entradas

- Endereço: n bits selecionam 2^n linhas
- Dados (bidirecional): m bits de dados de escrita ou leitura
- Controle: WR, RD, OutputEnable

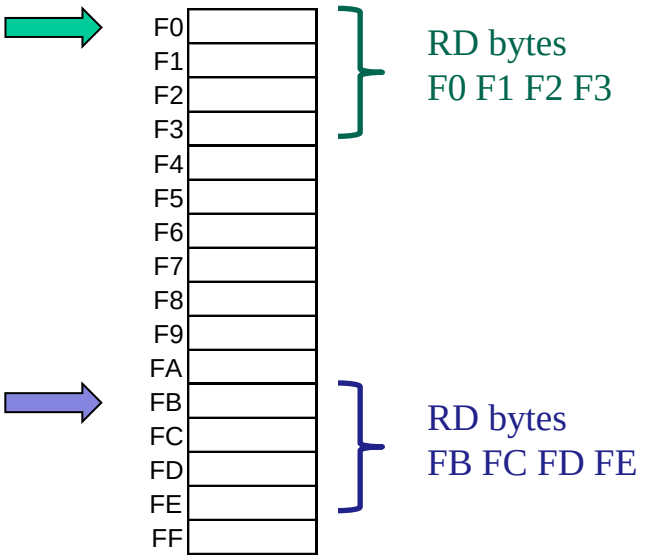
Tamanho da memória

- $2^n * m$ 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



IC-UNICAMP

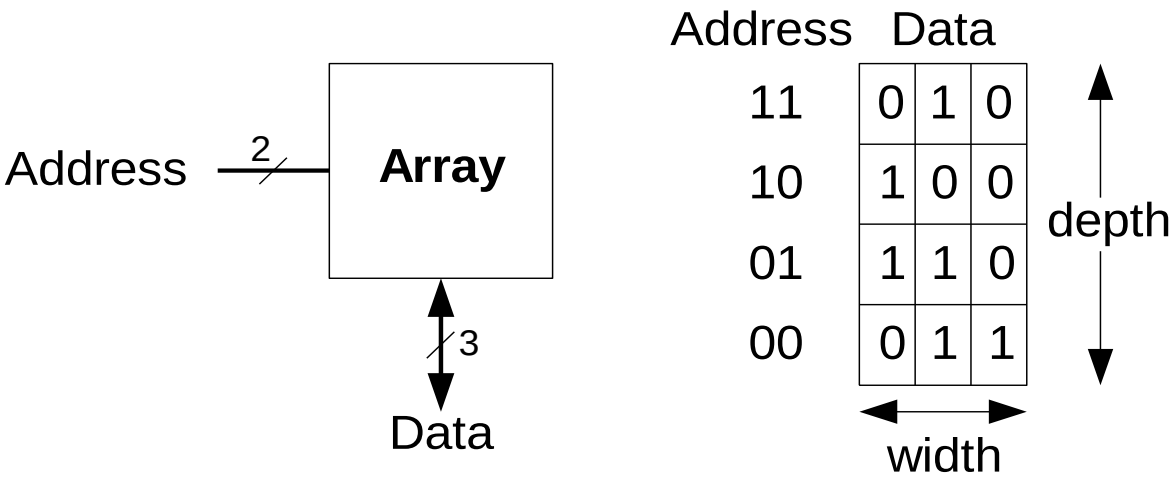
Memórias

- 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 × 3-bit

Word size de 3-bits



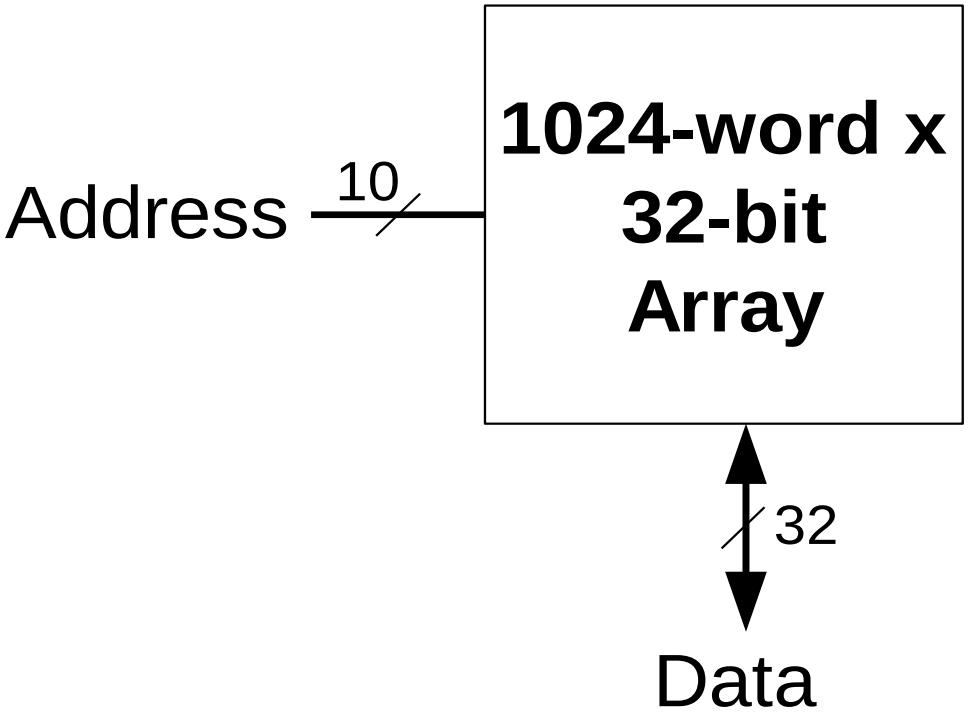
Memória : Exemplo

Nº de linhas = 2^{10} = 1024 = 1K

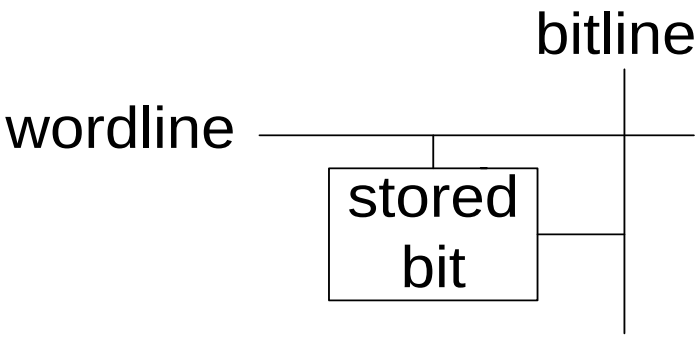
Nº de colunas = word size = 32 bits = 4B

Tamanho

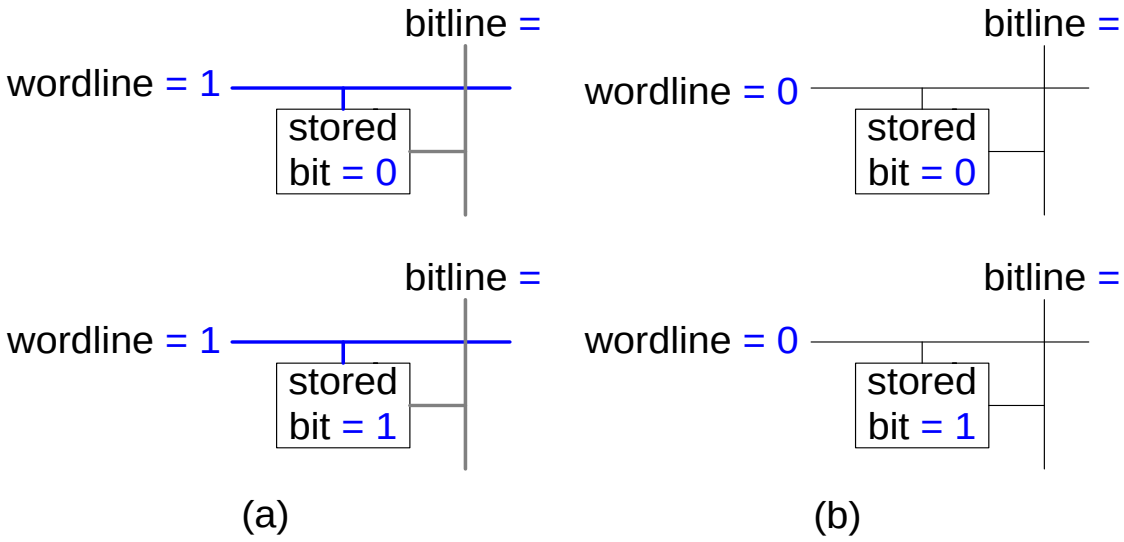
- 1K x 4B
- ou 4KB
- ou 32 Kb



Memória: Célula de bit



Exemplo



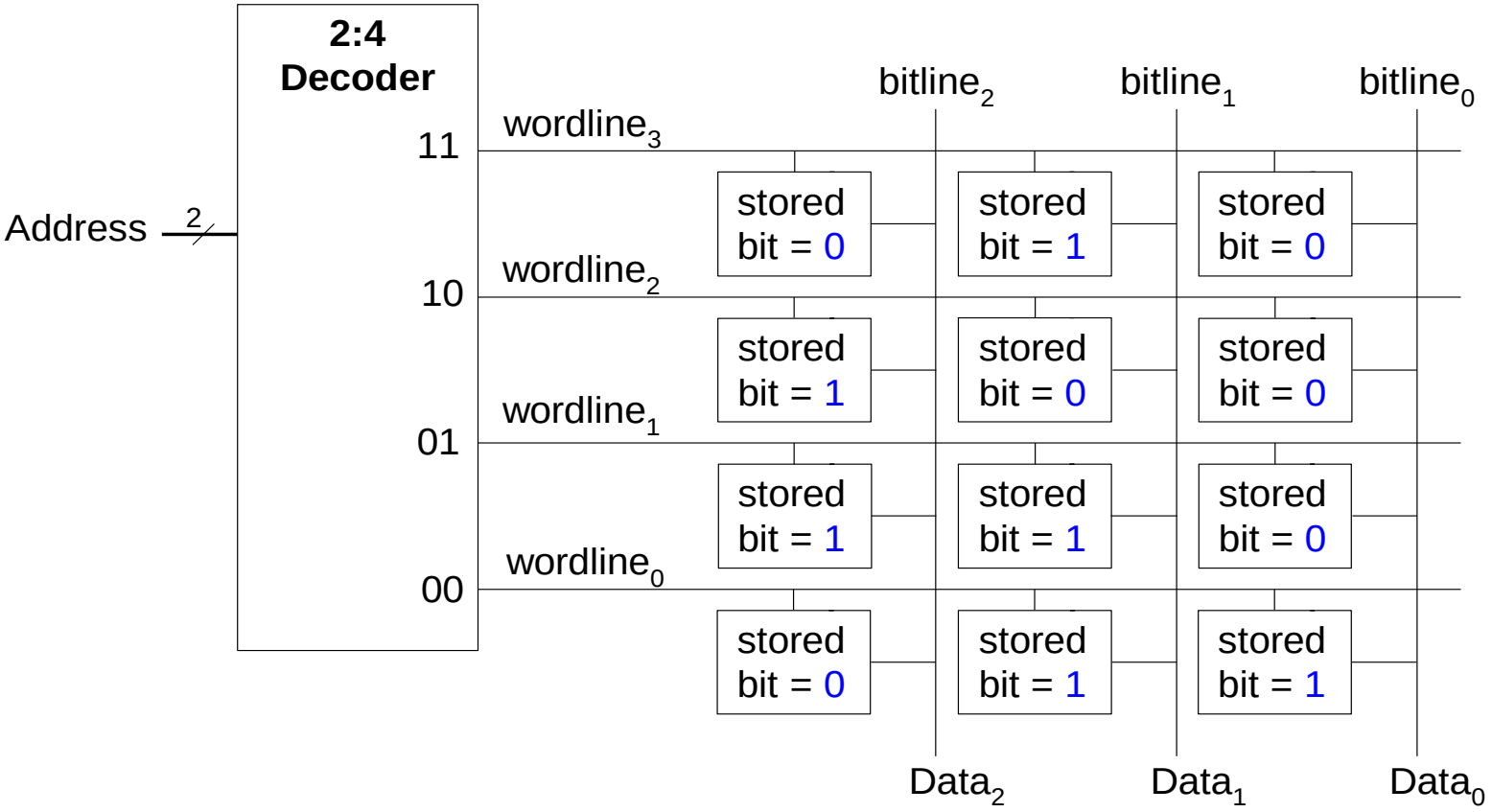
Procedimento para leitura

- 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


2:4
Decoder
Address
2
11
10
01
00
wordline3
wordline2
wordline1
wordline0
bitline2
bitline1
bitline0
stored bit = 0
stored bit = 1
stored bit = 0
stored bit = 1
stored bit = 0
stored bit = 0
stored bit = 1
stored bit = 1
stored bit = 0
stored bit = 0
stored bit = 1
stored bit = 1
Data2
Data1
Data0

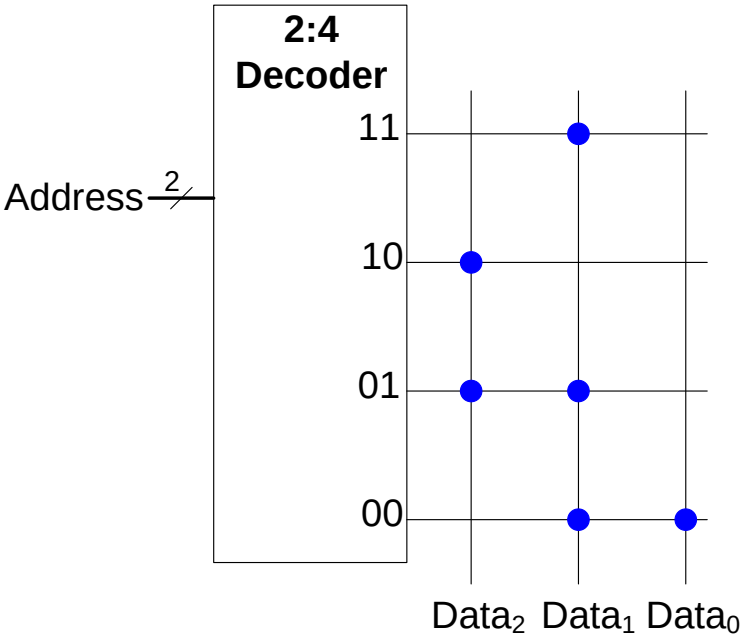
Tipos de Memórias

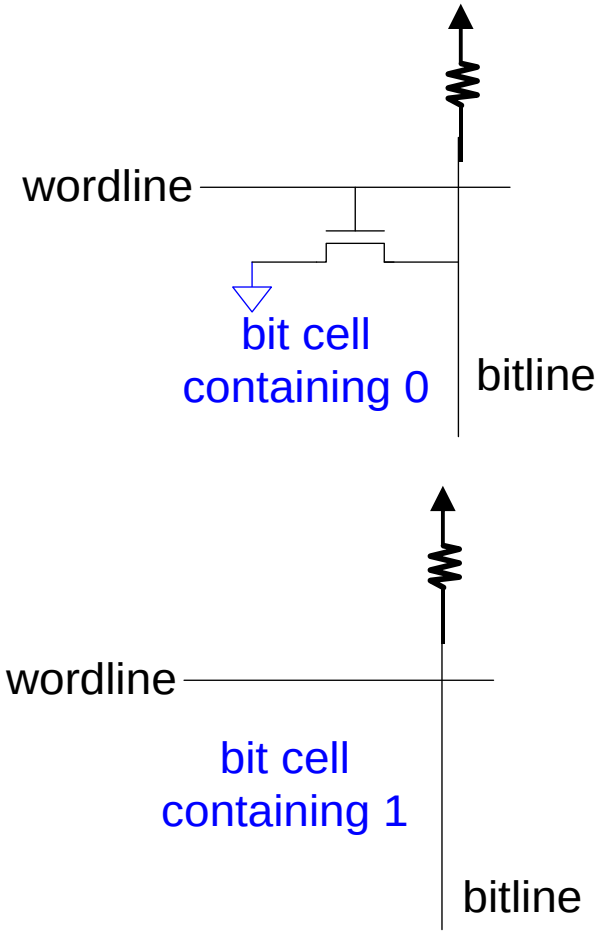
- Read only memory (ROM): não volátil
- Random access memory (RAM): volátil

ROM

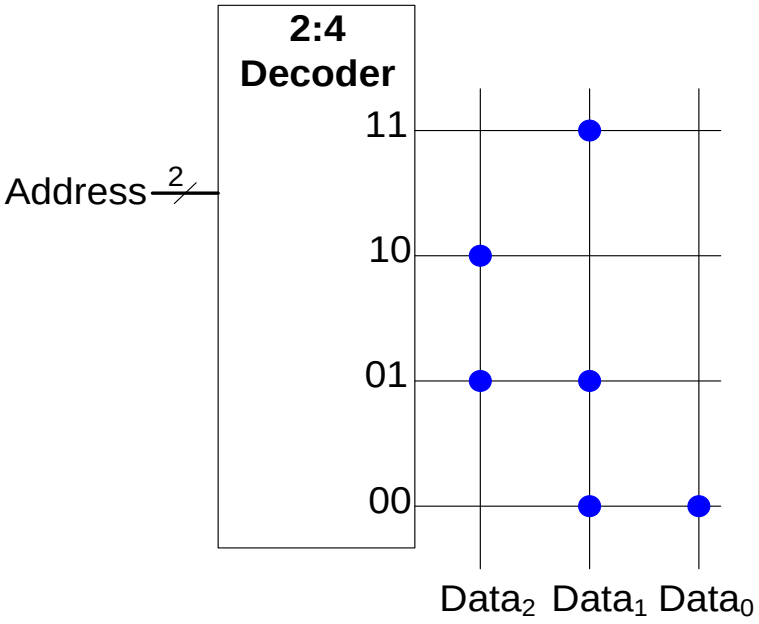
- Read only memory (ROM)
  - Não volátil: não perdem seus dados quando a alimentação é desligada
  - Pode ser lida rapidamente, porém a escrita é lenta (no caso das ROMs reprogramáveis)
  - Memórias em câmeras digitais, pen drives são ROMs
  - Historicamente denominadas de *read only memory* porque as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

ROM


2:4
Decoder
Address
2
11
10
01
00
Data2 Data1 Data0


wordline
bit cell
containing 0
bitline
wordline
bit cell
containing 1
bitline

ROM



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

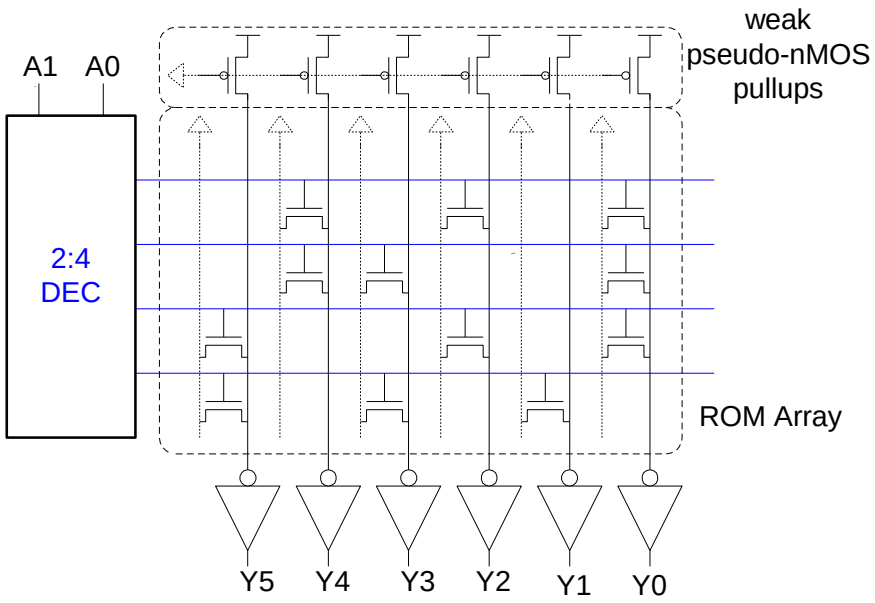
depth

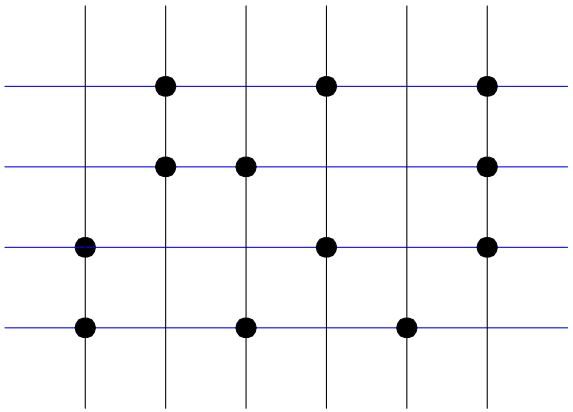
width

Detalhes da ROM

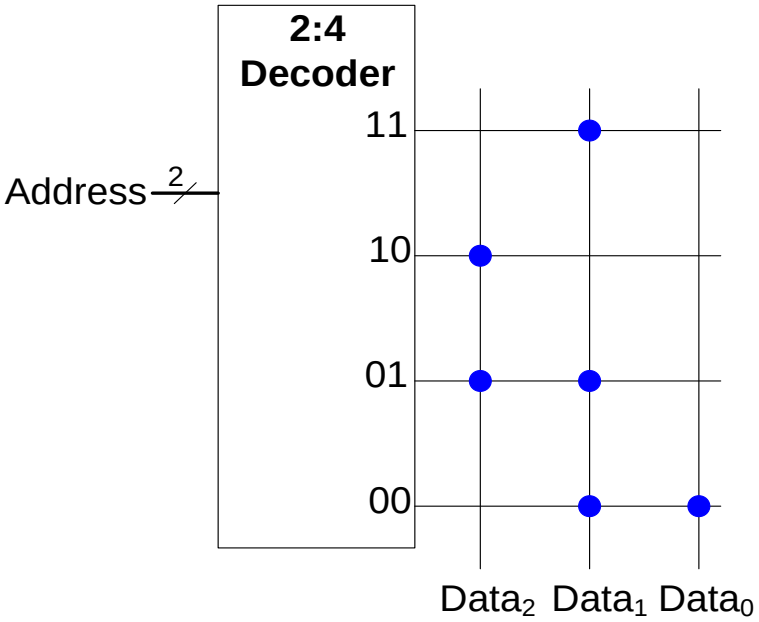
- 4-word x 6-bit ROM
  - Representada por diagrama de pontos
  - Pontos indicam 1's na ROM

```
Word 0: 010101
Word 1: 011001
Word 2: 100101
Word 3: 101010
```





Lógica com ROM



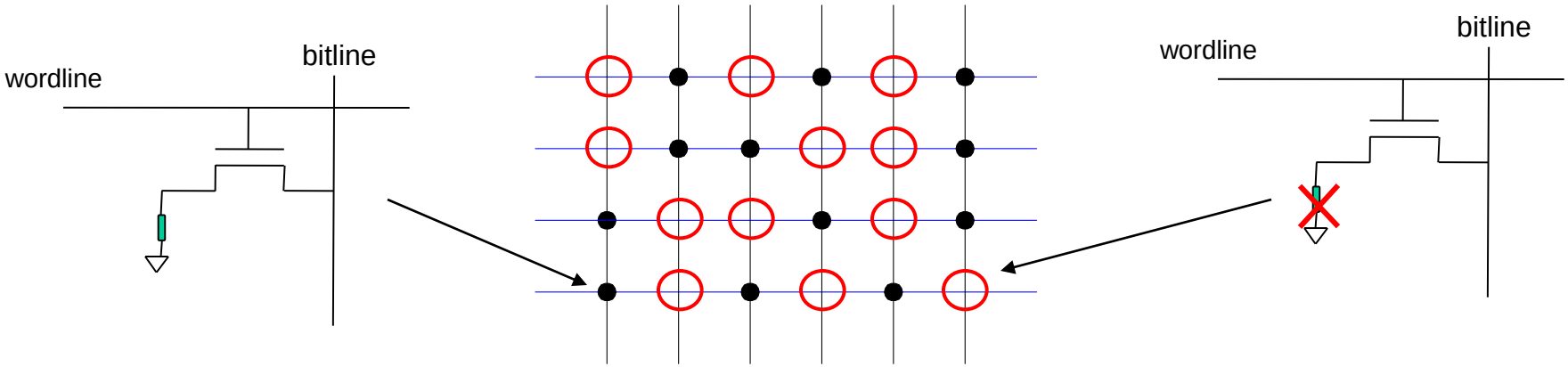
$$Data_2 = A_1 \oplus A_0$$

$$Data_1 = \sim(A_1 \,.\, \sim A_0)$$

$$Data_0 = \sim A_1 . \sim A_0$$

ROM Programável (PROM)

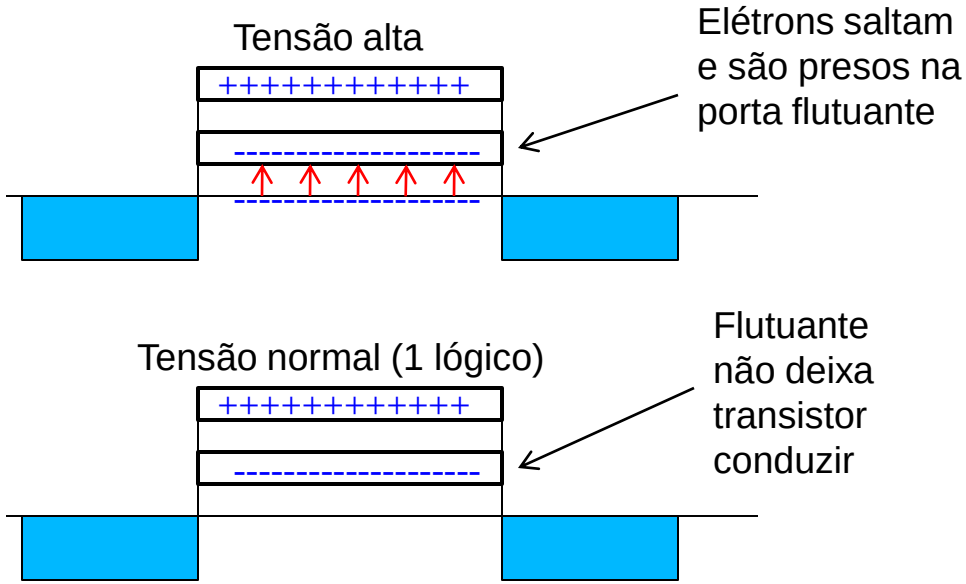
- Arquitetura semelhante à ROM
- Chip é uma matriz de transistores completa
- Queima fusíveis após fabricação para desconectar transistores (resulta no bit zero)



ROM Programável

- EPROM, EEPROM 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



RAM

- 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 (specificamente, DRAM)
  - Historicamente denominada de *random* access memory porque qualquer palavra de dado pode ser acessada como qualquer outra (em contraste com sequential access memories como fita magnética).

IC-UNICAMP

Tipos de RAM

- Os dois tipos de RAM são:
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)

- A diferença é como armazenam os dados:
  - DRAM usa um capacitor
  - SRAM usa cross-coupled inverters (“latch”)

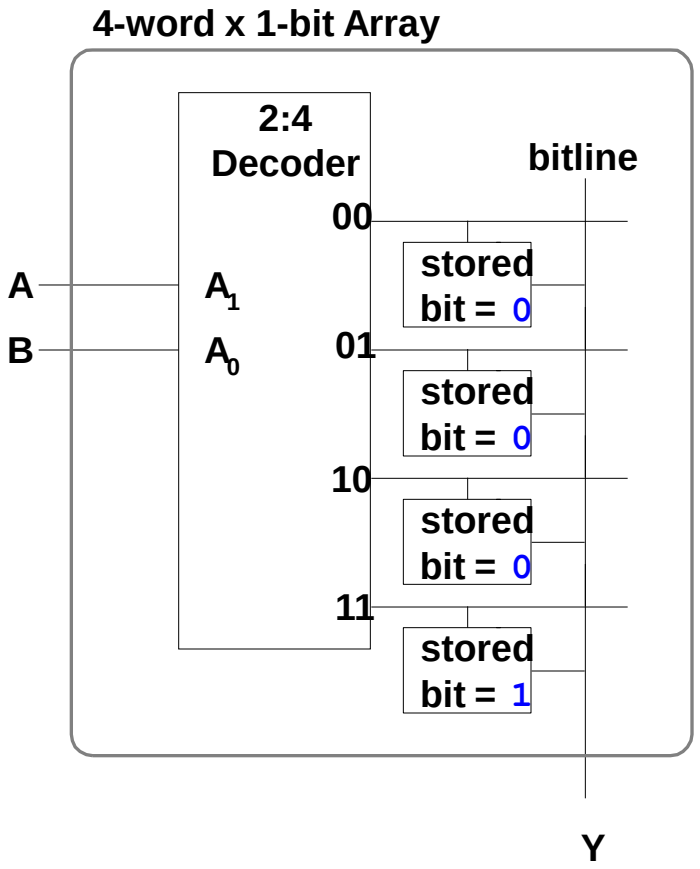
Lógica com Memória

- A memória usada para executar funções lógicas é denominada *lookup tables* (LUT).

- O usuário tem o valor de saída para cada combinação das entradas (address).

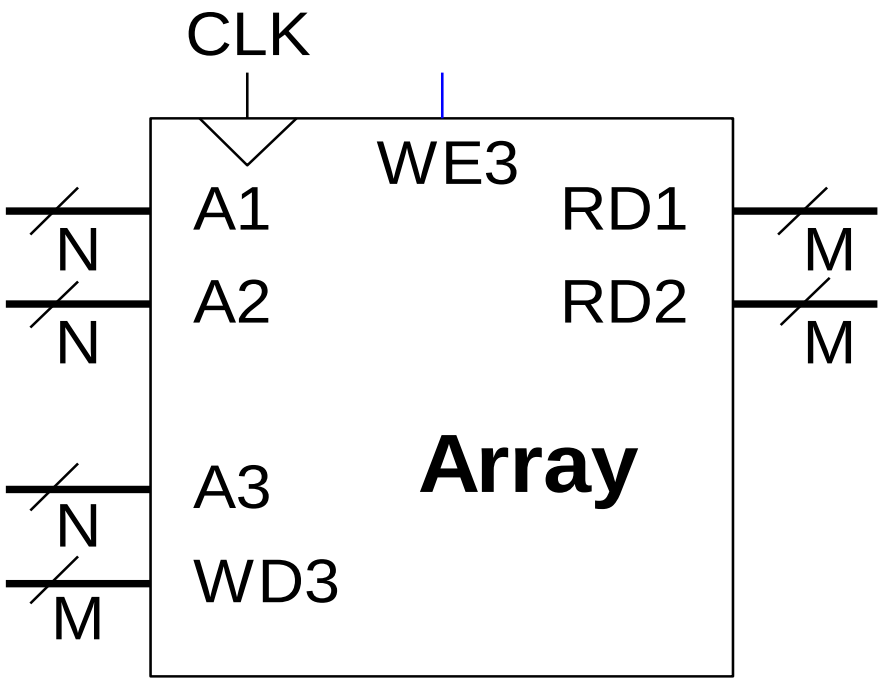
Truth Table

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

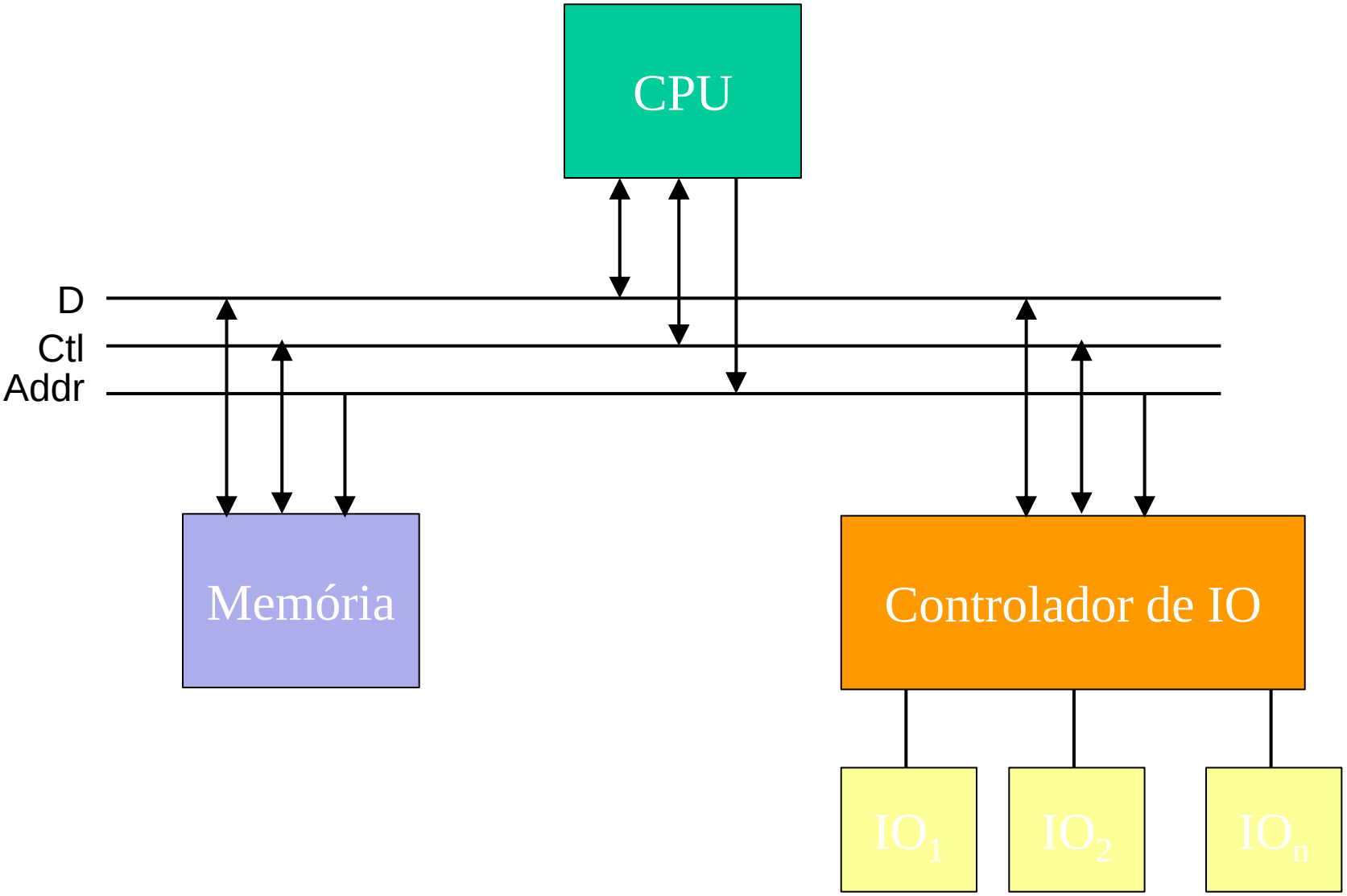


Memórias Multi-Portas

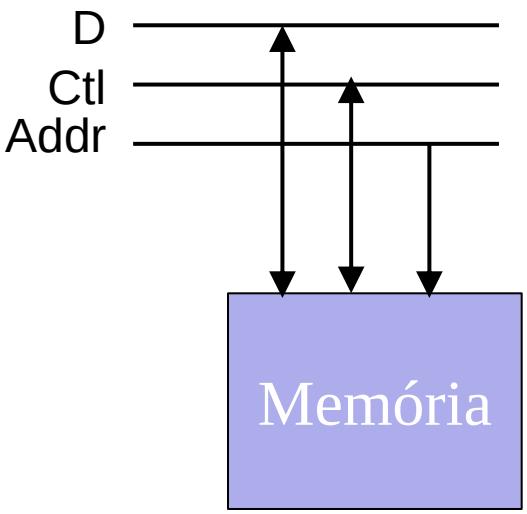
- Porta: par endereço/dado (address/data)
- Memória 3-portas
  - 2 portas de leitura (A1/RD1, A2/RD2)
  - 1 porta de escrita (A3/WD3, WE3 enables writing)



Sistema de memória: uso típico



Dispositivo de memória: interfaces



- Dados:
  - bidirecional: dados a serem escritos ou lidos
- Address:
  - input apenas: endereço da posição de memória onde os dados serão escritos ou de onde serão lidos
- Control
  - Inputs:
    - RD, WR → indicam a operação a ser executada
    - OE: output enable (saída Z)
  - Outputs: não é usual
    - poderia ser status = ready, por exemplo

IC-UNICAMP

Constituição de um sistema de memória com componentes

- Um sistema de memória: ex. 8GB
  - endereçamento a bytes: necessários 33 bits
  - controle: RD e WR
  - dados: 8 bits
- Pode ser constituído por 8 chips de memória de 1 GB cada. Cada chip:
  - endereçamento: necessários 30 bits → usar os 30 LSB dos 33 bits do sistema
  - controle: RD e WR comuns a todos os chips
  - dados: 8 bits comuns a todos os chips

Decodificação de endereço

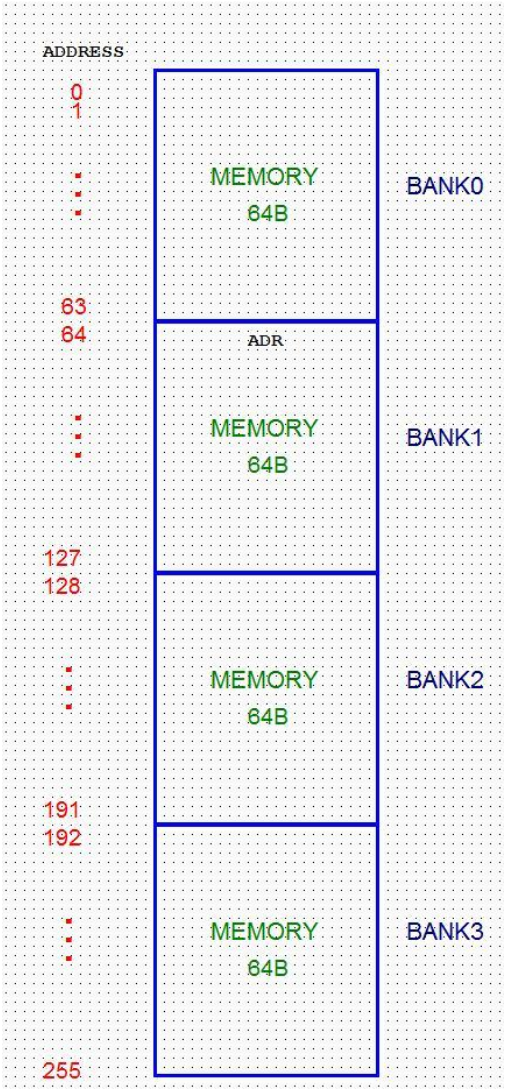
- Em geral, sistema de memória
  - n módulos (chips) de tamanho fixo
  - barramentos grandes, para permitir expansão
  - decodificação de endereço
- Exemplo em seguida

Mapa da Memória

- Descreve como bancos de memória (chips) podem ser ligados para formar a memória global

- Supor sistema de memória com os seguintes sinais:
  - adr: endereços (8-bits)
  - memdata: leitura de dados (8-bits)
  - writedata: escrita de dados (8-bits)
  - clk: clock
  - memwrite: habilita escrita

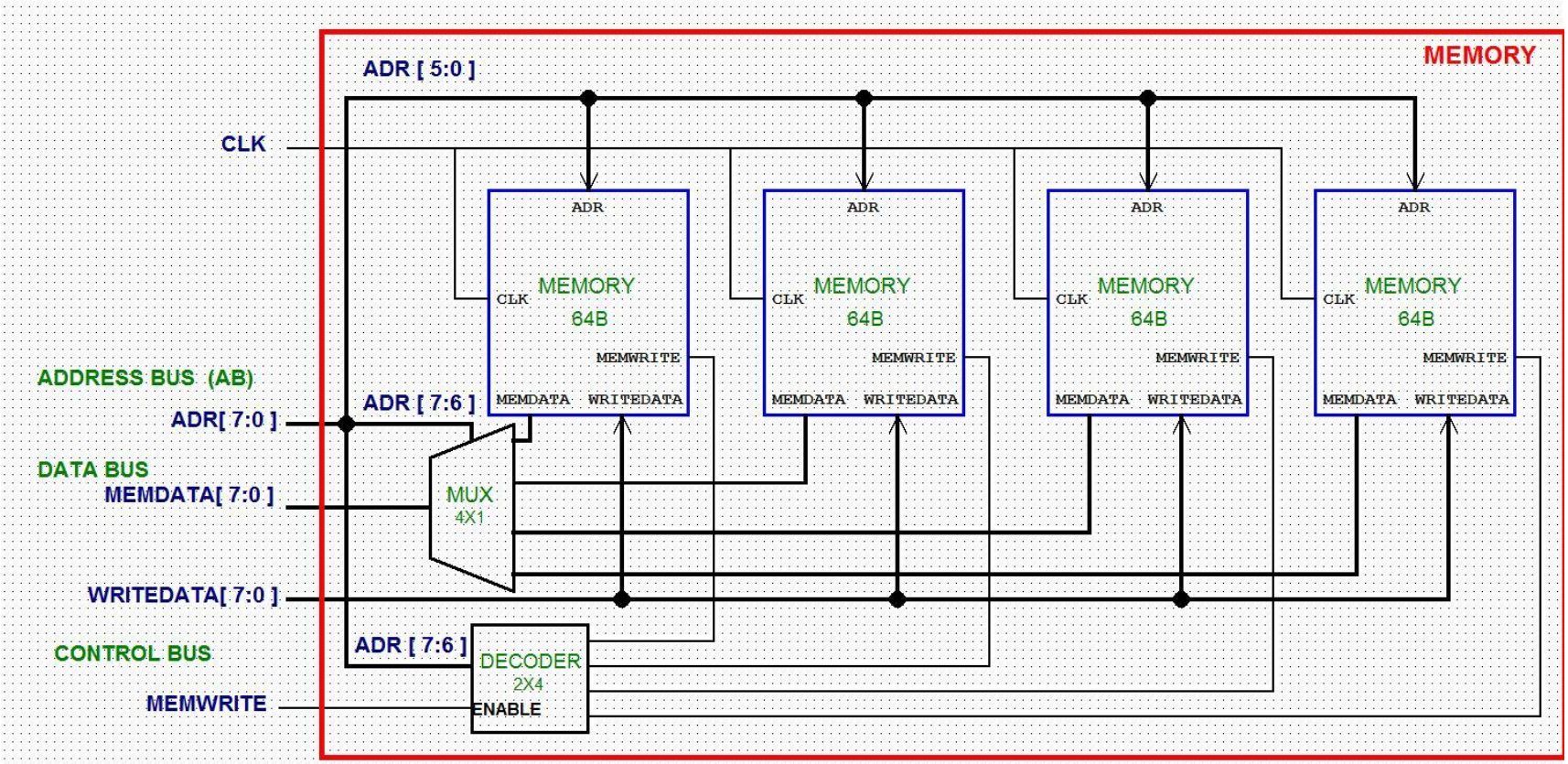
- Como conectar os pinos dos vários chips de memória com os sinais do processador?



IC-UNICAMP

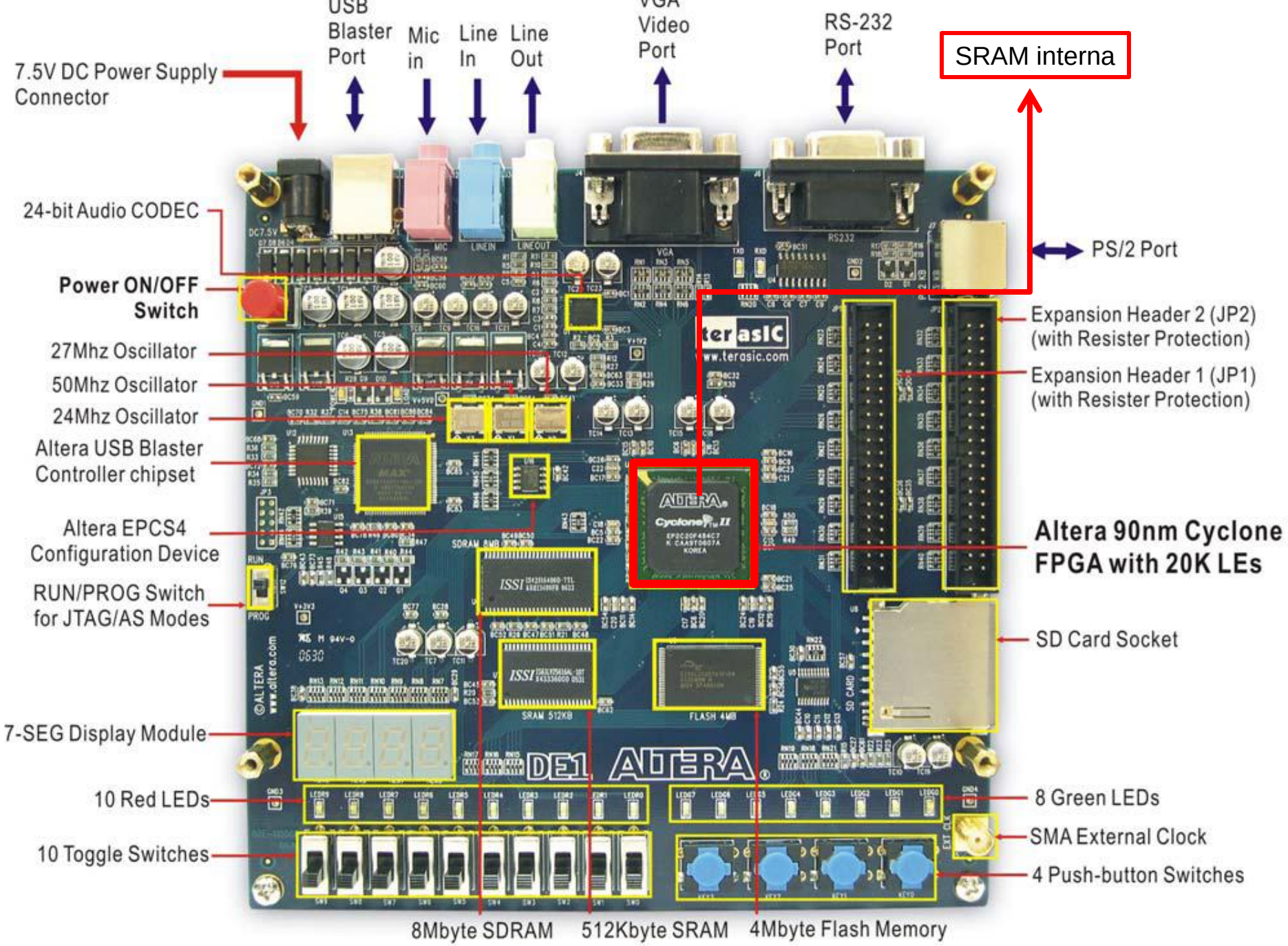
Decodificação de Endereços

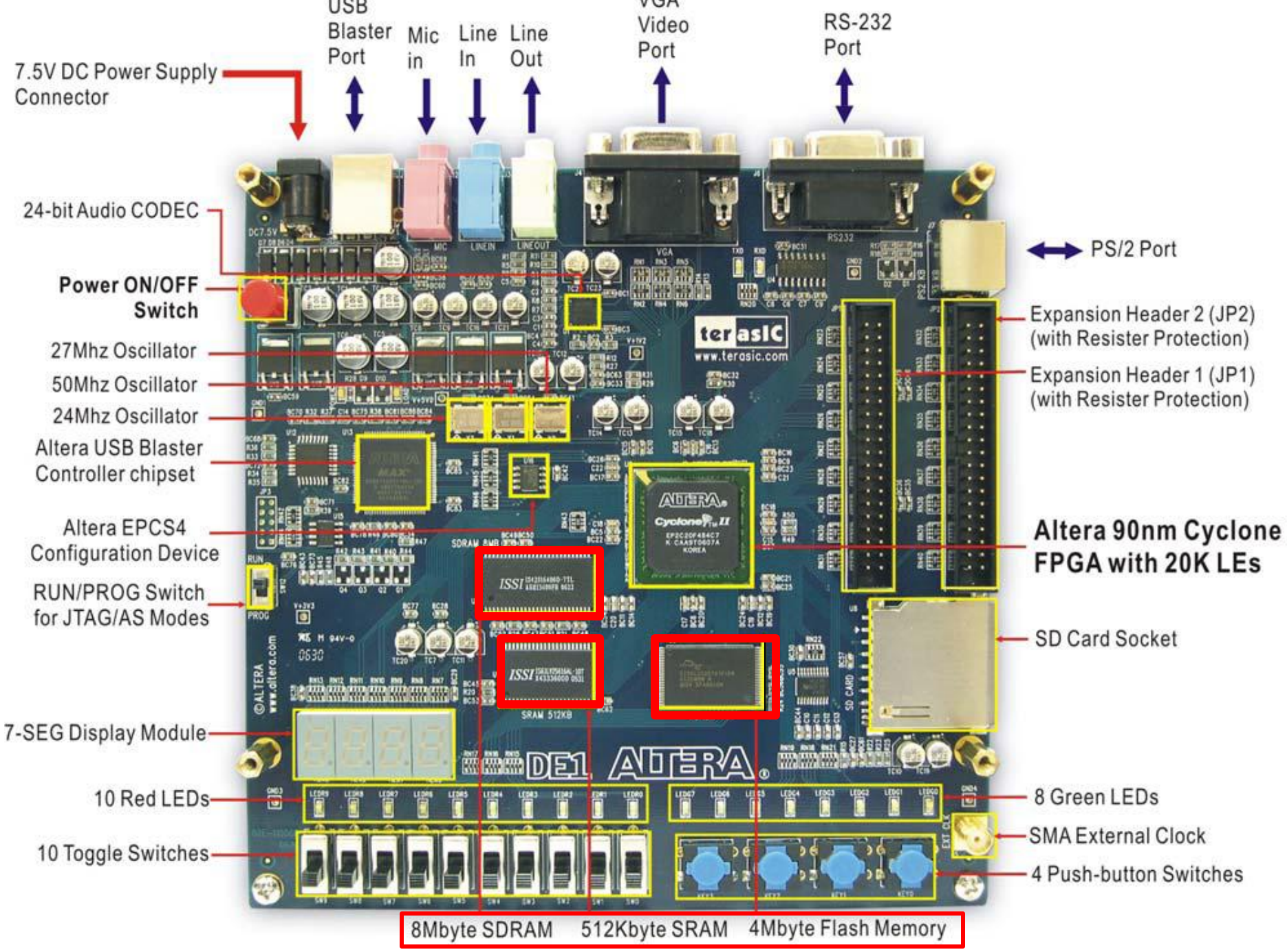
- Exemplo
  - Chips (bancos) de memória são de 64B
  - Memória total desejável é de 256B
  - Barramento de endereços e dados com 8 bits



Uso de Memória na DE1

- Acesso à memória interna à FPGA
  - Sintetizado a partir da descrição VHDL adequada (ver manual do Quartus II)
  - Ou via instanciação de megafunções específicas da Altera (LPM).
- Acesso às unidades externas (SRAM, SDRAM, Flash)
  - Via programação dos pinos específicos interligados à FPGA, de acordo com as especificações da unidade de memória
  - Ver manual DE1_UserManual_v1018.pdf


USB Blaster Port
Mic in
Line In
Line Out
VGA Video Port
RS-232 Port
SRAM interna
7.5V DC Power Supply Connector
24-bit Audio CODEC
Power ON/OFF Switch
27Mhz Oscillator
50Mhz Oscillator
24Mhz Oscillator
Altera USB Blaster Controller chipset
Altera EPCS4 Configuration Device
RUN/PROG Switch for JTAG/AS Modes
7-SEG Display Module
10 Red LEDs
10 Toggle Switches
PS/2 Port
Expansion Header 2 (JP2) (with Resister Protection)
Expansion Header 1 (JP1) (with Resister Protection)
Altera 90nm Cyclone II FPGA with 20K LEs
SD Card Socket
8 Green LEDs
SMA External Clock
4 Push-button Switches
8Mbyte SDRAM
512Kbyte SRAM
4Mbyte Flash Memory


USB Blaster Port
Mic in
Line In
Line Out
VGA Video Port
RS-232 Port
7.5V DC Power Supply Connector
24-bit Audio CODEC
Power ON/OFF Switch
27Mhz Oscillator
50Mhz Oscillator
24Mhz Oscillator
Altera USB Blaster Controller chipset
Altera EPCS4 Configuration Device
RUN/PROG Switch for JTAG/AS Modes
7-SEG Display Module
10 Red LEDs
10 Toggle Switches
PS/2 Port
Expansion Header 2 (JP2) (with Resister Protection)
Expansion Header 1 (JP1) (with Resister Protection)
Altera 90nm Cyclone II FPGA with 20K LEs
SD Card Socket
8 Green LEDs
SMA External Clock
4 Push-button Switches
8Mbyte SDRAM
512Kbyte SRAM
4Mbyte Flash Memory

IC-UNICAMP

Instanciação de memórias via MegaWizard Plug-in Manager

- Wizard auxilia a instanciação de megafunctions
- Tools > MegaWizard Plug-in Manager > Create New Custom Megafunction Variation > Memory Compiler
  - vamos trabalhar com RAM 1 Port; selecionar VHDL e indicar nome do arquivo
  - definir largura, profundidade e single clock
  - Next: escolher ClkEnable (não), FF em q output port (não), assynchronous clear (não)
  - Next: conteúdo inicial. Caso necessário definir arquivo .mif (por exemplo, código binário na memória de instruções)
  - Finish

IC-UNICAMP

Código gerado para altsyncram

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY altera_mf;
USE altera_mf.all;

ENTITY my_ram IS
PORT
(
    address    : IN STD_LOGIC_VECTOR (6 DOWNTO 0);
    clock      : IN STD_LOGIC  := '1';
    data       : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    wren       : IN STD_LOGIC ;
    q          : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
);
END my_ram;
```

Observar entradas e saída

IC-UNICAMP

Declaração de component (pode ser colocada em package)

```
ARCHITECTURE SYN OF my_ram IS
SIGNAL sub_wire0 : STD_LOGIC_VECTOR (7 DOWNTO 0);
COMPONENT altsyncram
GENERIC (
    clock_enable_input_a, clock_enable_output_a,
    init file, intended_device_family       : STRING;
    lpm_hint, lpm_type                      : STRING;
    numwords_a                          : NATURAL;
    operation_mode, outdata_aclr_a      : STRING;
    outdata_reg_a, power_up_uninitialized     : STRING;
    widthad_a, width_a, width_byteena_a     : NATURAL
);
PORT (
       wren_a, clock0   : IN STD_LOGIC ;
       address_a : IN STD_LOGIC_VECTOR (6 DOWNTO 0);
       q_a: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
       data_a : IN STD_LOGIC_VECTOR (7 DOWNTO 0)
);
END COMPONENT;
```

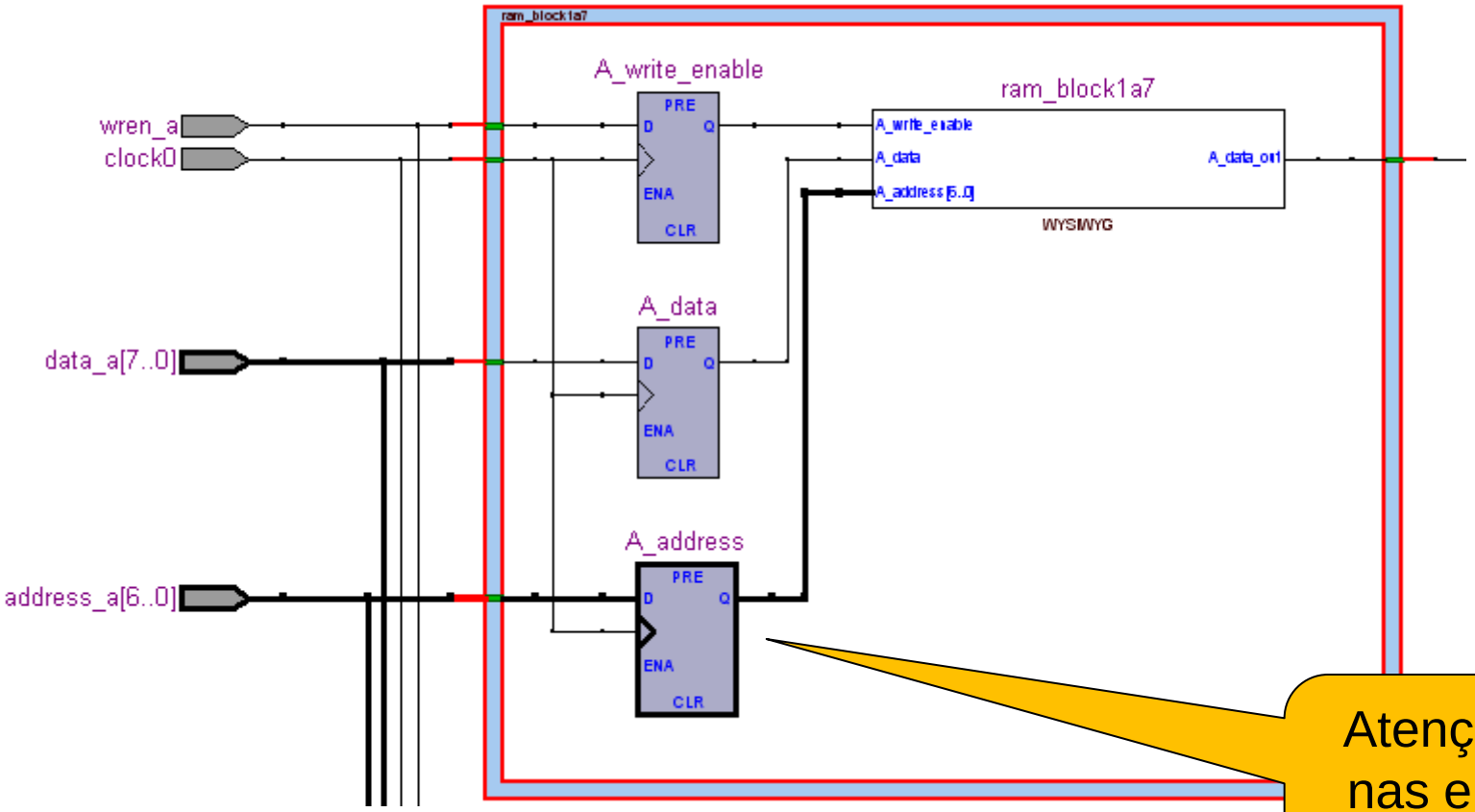
IC-UNICAMP

Architecture

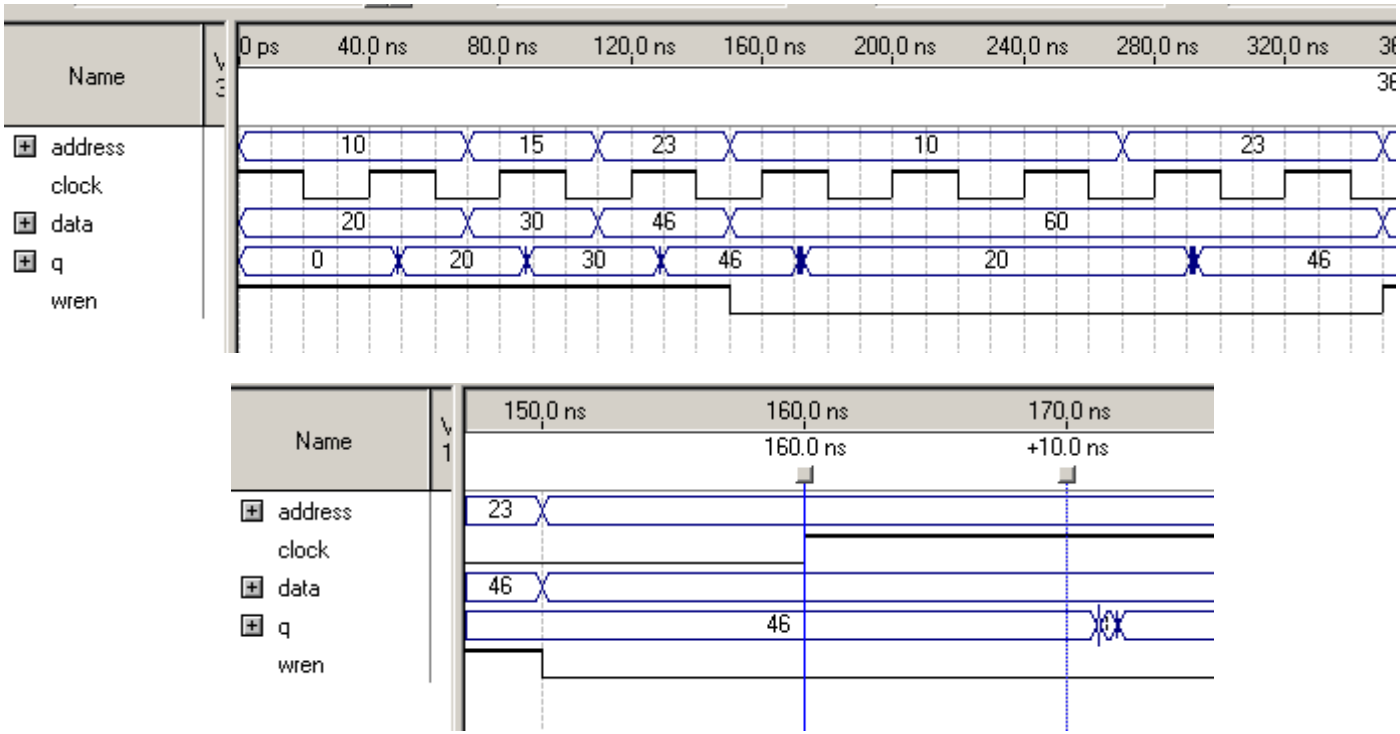
```
BEGIN
q    <= sub_wire0(7 DOWNTO 0);
altsyncram_component : altsyncram
GENERIC MAP (
    clock_enable_input_a => "BYPASS",
    clock_enable_output_a => "BYPASS",
    init_file => "test.mif",
    intended_device_family => "Cyclone II",
    lpm_hint => "ENABLE_RUNTIME_MOD=NO",
    lpm_type => "altsyncram",     numwords_a => 128,
    operation_mode => "SINGLE_PORT",
    outdata_aclr_a => "NONE",
    outdata_reg_a => "UNREGISTERED",
    power_up_uninitialized => "FALSE",
    widthad_a => 7,  width_a => 8, width_byteena_a => 1
)
PORT MAP (
    wren_a => wren,
    clock0 => clock,
    address_a => address,
    data_a => data,
    q_a => sub_wire0
);
END SYN;
```

IC-UNICAMP

Netlist gerado (detalhe)



Simulação



- Escrita: wren =1 e uma borda do clock
- Leitura: wren = 0 e uma borda do clock
  - t_p entre o clock e q ~ 10 ns

IC-UNICAMP

Exemplo de arquivo .mif

```
%  multiple-line comment   between %%  %
-- single-line comment
DEPTH = 32;                         -- The size of data in bits
WIDTH = 8;                          -- The size of memory in words
ADDRESS_RADIX = HEX;        -- The radix for address values
DATA_RADIX = BIN;                   -- The radix for data values
CONTENT                     -- start of (address : data pairs)
BEGIN
00 : 00000000;                  -- memory address : data
01 : 00000001;
02 : 00000010;
03 : 00000011;
04 : 00000100;
05 : 00000101;
06 : 00000110;
07 : 00000111;
08 : 00001000;
09 : 00001001;
0A : 00001010;
0B : 00001011;
0C : 00001100;

END;
```

IC-UNICAMP

Instanciação direta de memória

- Alternativa: instanciação direta da memória
- Vantagens
  - mais simples
  - não depende da ferramenta

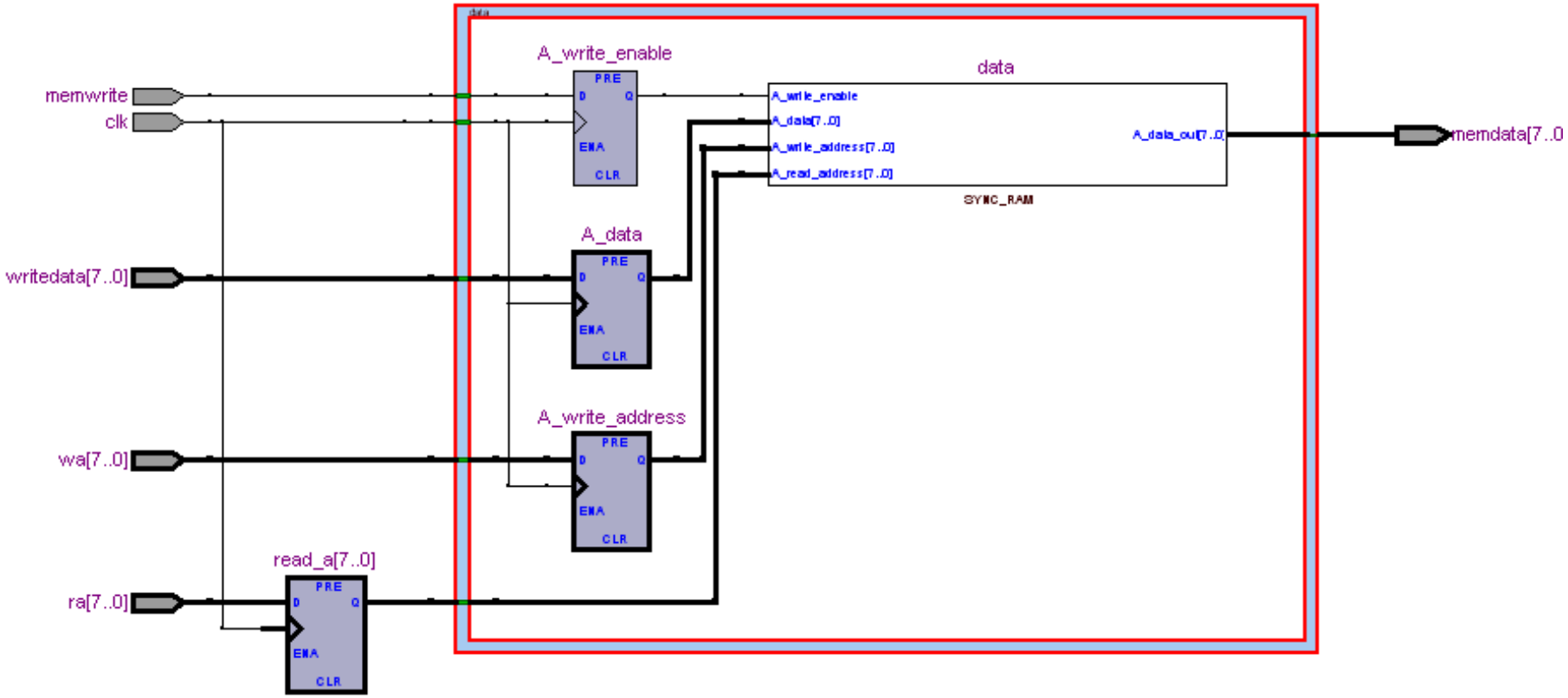
Instanciação direta de memória: Entity

```
library ieee;use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ram_s_wizard is
   generic(
      depth : integer range 1 to 8 := 8;
      width: integer range 1 to 8 := 8;
      init_file : string := "init_file.mif");
         -- .mif filename

   port(clk : in std_logic;
      memwrite:  in std_logic
         -- write control signal
      ra, wa : in std_logic_vector(depth-1 downto 0);
         -- read and write addresses
      writedata : in std_logic_vector(width-1 downto 0);
         -- data to be written
      memdata : out std_logic_vector(width-1 downto 0));
         --    memory read output
end ram_s_wizard;
```

Instanciação direta de memória: Arch

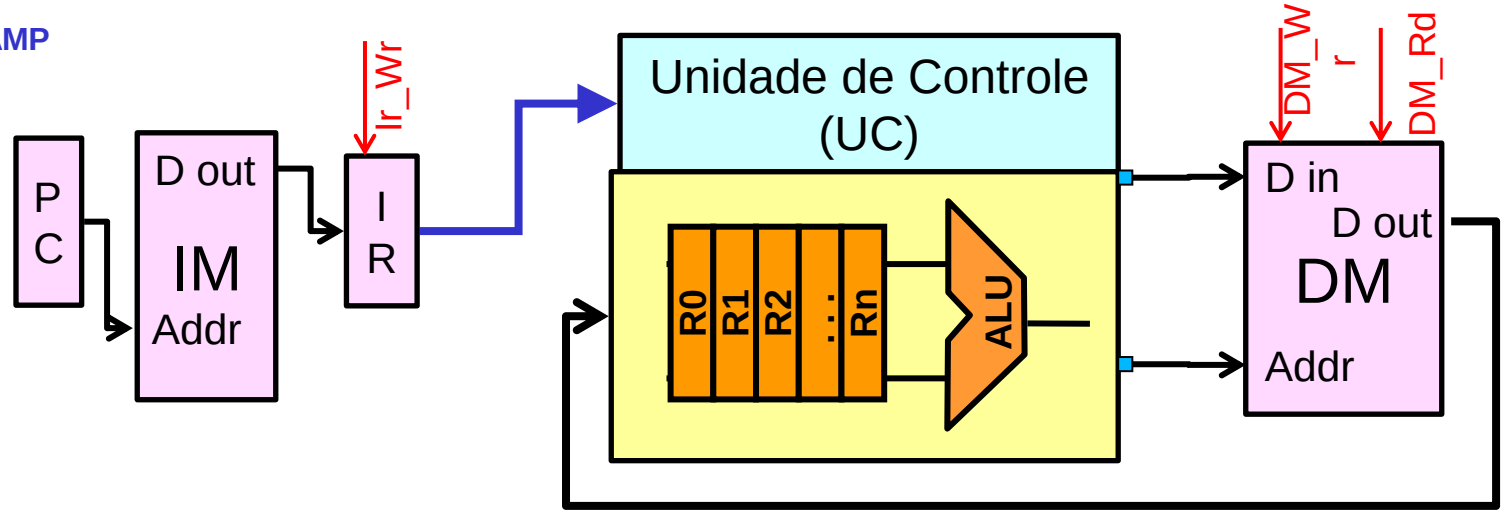
```
architecture a_ram of ram_s_wizard is
   type mem_type is array (0 to 2**depth-1) of
           std_logic_vector(width-1 downto 0);
   attribute ram_init_file : string;
   signal data : mem_type;
   signal read_a : std_logic_vector(depth-1 downto 0);
   attribute ram_init_file of data : signal is init_file;
begin
   process(clk)
      begin
         if clk'event and clk = '1' then
            if memwrite = '1' then
               data(to_integer(unsigned(wa))) <= writedata;
            end if;
            read_a <= ra;
         end if;
      end process;
   memdata <= data(to_integer(unsigned(read_a)));
end a_ram;
```

Netlist gerado (detalhe)



Obs: antes de utilizar (instanciar), simular para ver temporização

Memórias no m1ps



- IM: Instruction Memory
  - Só leitura: wren = 0
  - Address = 26 bits
- DM: Data Memory
  - Leitura e escrita
  - Address = barramento de 32 bits
- Larguras de ambas memórias: 32 bits

Memórias no m1ps

- Mas ao projetar o m1ps, limitação de tamanho na DE1
- Memórias implementadas serão menores, com menos linhas, suficientes para os experimentos
- IM e DM
  - 256 linhas
  - Bastam 8 bits de endereço
- Necessidade de decodificação de endereço