Placard

Dossiê Pedagógico da UC

Dossiê Pedagógico da UC
Tudo o que quer saber sobre AC1 e tem medo de perguntar

Toda a informação sobre o funcionamento da UC (leitura obrigatória)

Apresentação da disciplina

Powerpoint
Resumo do dossiê pedagógico

Resumo da informação sobre o funcionamento da UC e dos seus processos de avaliação.

Avaliação

Regras de avaliação e documentos complementares

- **Componente teórica: avaliação periódica [peso de 60%]**
  - TT1 (45%): 28/10/2023 às 10h00 (a confirmar)
  - TT2 (55%): 10/01/2024 às 14h00, na época de exames
  - Exame em Recurso (26/02/2024 às 14h00)
- **Componente prática: avaliação periódica [peso de 40%]**
  - TP1 (40%): 28/10/2023 às 14h00 (a confirmar)
  - TP2 (60%): 16/12/2023 às 10h00 (a confirmar)
  - Nota prática = (0,40 * TP1 + 0,60 * TP2)
  - Exame em Recurso (26/02/2024 às 18h00)
- **Nota mínima em qualquer das componentes para efeitos de aprovação à disciplina: 7,5 valores (nota obtida por arredondamento com 1 casa decimal)**

Organização e regras para realizar um teste de escolha múltipla

Aula 1

- Conceitos fundamentais em Arquitetura de Computadores
  - Arquitetura básica de um sistema computacional
  - Arquitetura básica do CPU
  - O ciclo de execução de uma instrução
  - Níveis de representação
  - Codificação de instruções
  - *Instruction Set Architecture* (ISA)
  - Classes de instruções

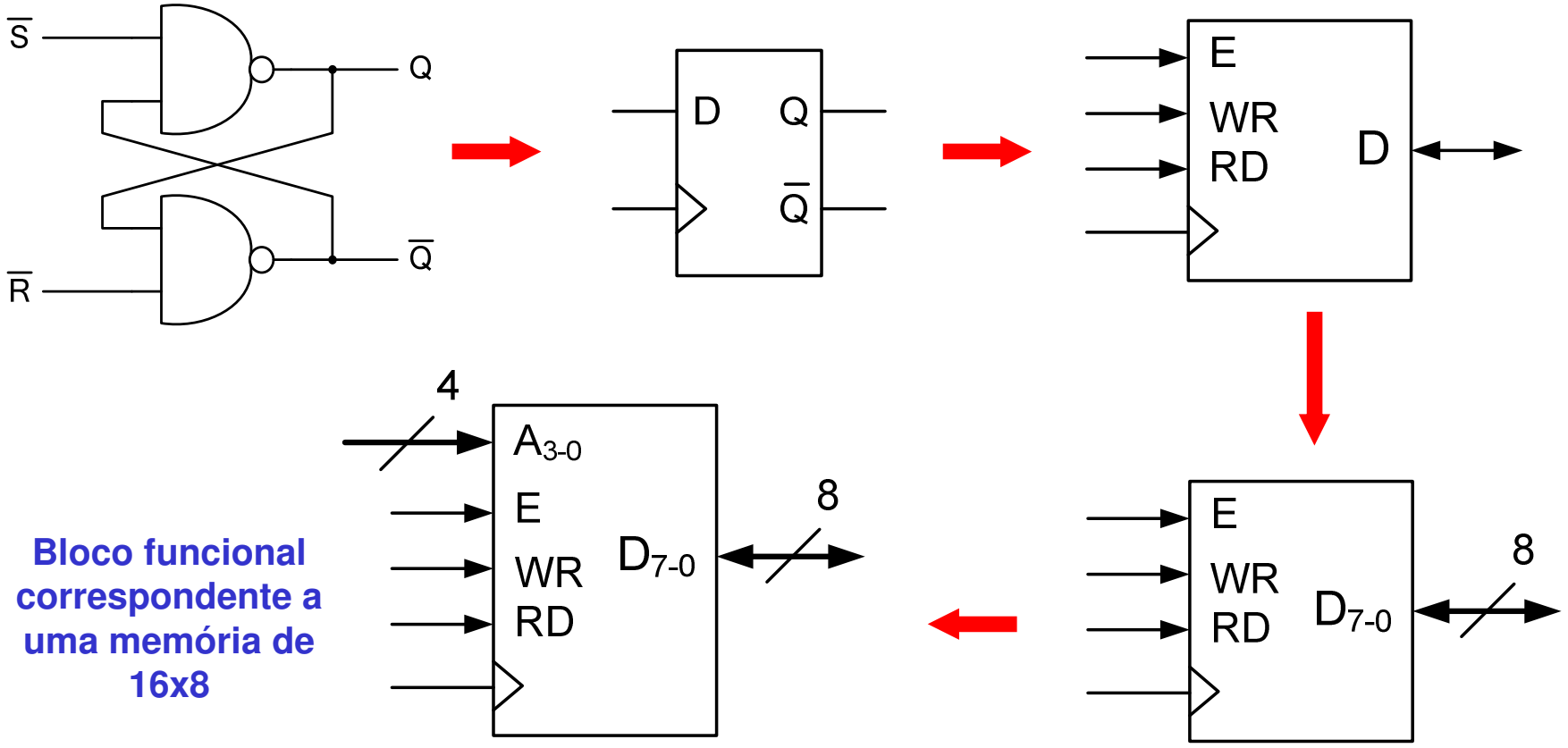
Bernardo Cunha, José Luís Azevedo, Arnaldo Oliveira

Arquitetura de Computadores e Sistemas Digitais

- Arquitetura de Computadores é uma das áreas de aplicação direta dos conceitos, técnicas e metodologias aprendidas nas duas UCs de Sistemas Digitais
- Em Arquitetura de Computadores, contudo, trabalha-se num nível de abstração diferente
- Recorre-se, na maior parte das vezes, a **blocos funcionais complexos** com cuja síntese, normalmente, não temos que nos preocupar (isso não significa que a sua funcionalidade não tenha que ser totalmente compreendida)

Exemplo: memória RAM 16x8

- Por exemplo, uma "Memória“ (um dispositivo com capacidade para armazenar informação digital binária) pode ser construída à custa de blocos básicos bem conhecidos dos sistemas digitais: **flip-flops**

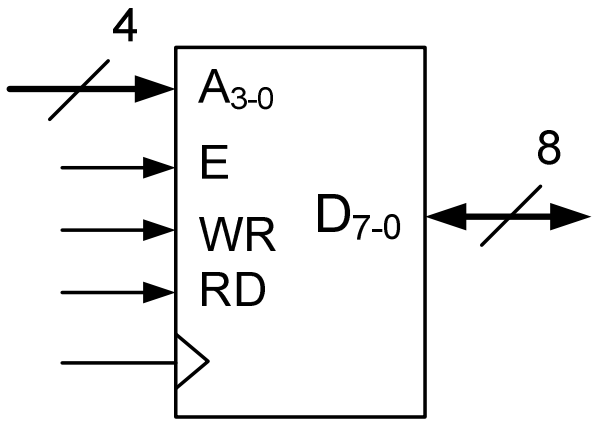


Bloco funcional correspondente a uma memória de 16x8

Exemplo: memória RAM 16x8 – VHDL

- O mesmo bloco pode, contudo, ser modelado numa linguagem de descrição de hardware, por exemplo VHDL, usando para isso uma mera descrição comportamental:

```
entity RAM_16_8 is
   port(clk      : in std_logic;
        addr     : in std_logic_vector(3 downto 0);
        enable   : in std_logic;
        wr       : in std_logic;
        rd       : in std_logic;
        data_io  : inout std_logic_vector(7 downto 0));
end RAM_16_8;
```

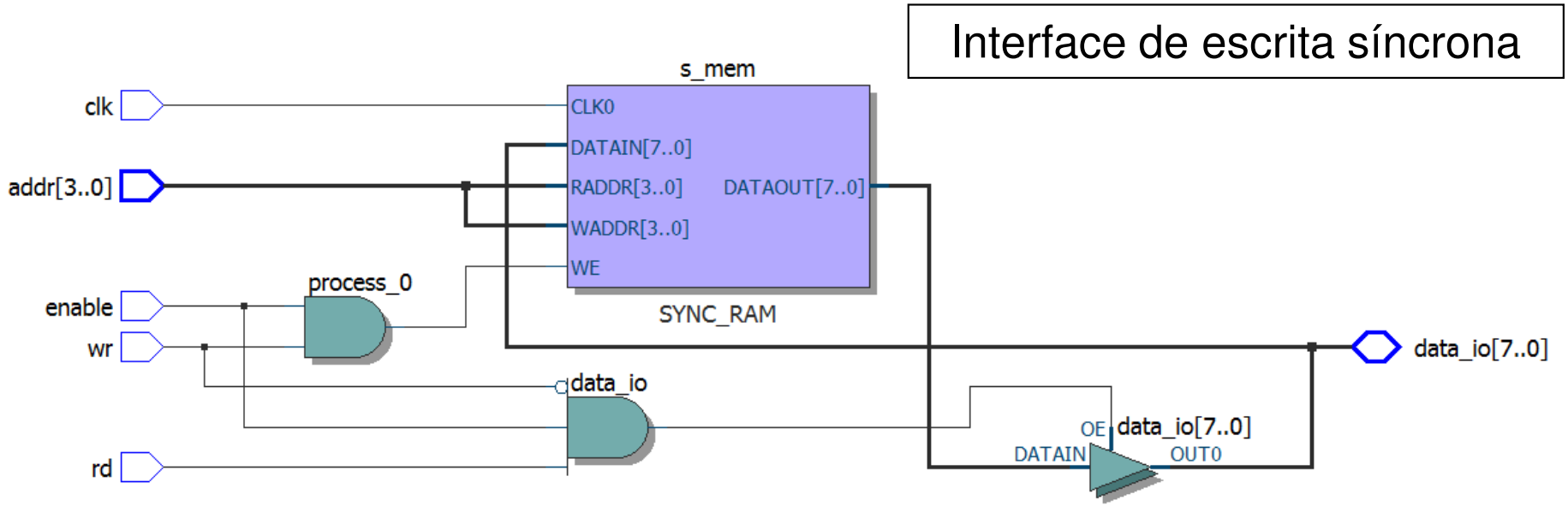


Escrita síncrona e leitura assíncrona.
O barramento de dados é bidirecional.

Exemplo: memória RAM 16x8 – VHDL

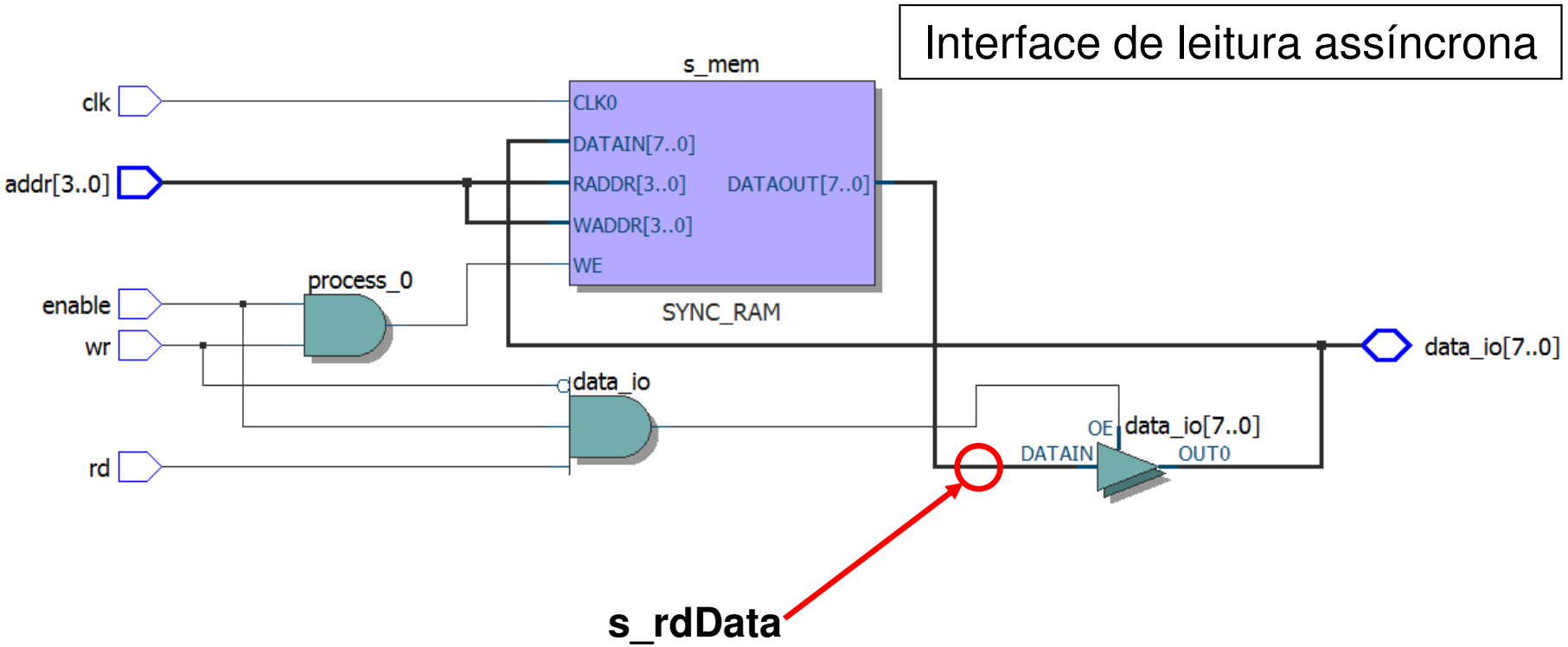
```
architecture Behavioral of RAM_16_8 is
   subtype TData is std_logic_vector(7 downto 0);
   type TMemory is array(0 to 15) of TData;
   signal s_mem : TMemory;
   signal s_rdData : std_logic_vector(7 downto 0);
begin
   process(clk)
   begin
     if(rising_edge(clk)) then
       if(enable = '1' and wr = '1') then
         s_mem(to_integer(unsigned(addr))) <= data_io;
       end if;
     end if;
   end process;
   s_rdData <= s_mem(to_integer(unsigned(addr)));
   data_io <= s_rdData when enable = '1' and rd = '1' and
              wr = '0' else (others => 'Z');
end Behavioral;
```

Exemplo: memória RAM 16x8 - síntese




```
process(clk)
begin
   if(rising_edge(clk)) then
     if(enable = '1' and wr = '1') then
       s_mem(to_integer(unsigned(addr))) <= data_io;
     end if;
   end if;
end process;
```

Exemplo: memória RAM 16x8 - síntese




```
s_rdData <= s_mem(to_integer(unsigned(addr)));
data_io  <= s_rdData when enable = '1' and
                          rd = '1'     and
                          wr = '0'     else
            (others => 'Z');
```

A máquina e a sua linguagem

- Princípios básicos dos computadores atuais:
  - As instruções são representadas da mesma forma que os números
  - Os programas são armazenados em memória, para serem lidos e escritos, tal como os números

- Estes princípios formam os fundamentos do conceito da arquitetura *"**stored-program**"*
  - O conceito "stored-program" implica que na memória possa residir, ao mesmo tempo, informação de natureza tão variada como: o código fonte de um programa em C, um editor de texto, um compilador, e o próprio programa resultante da compilação

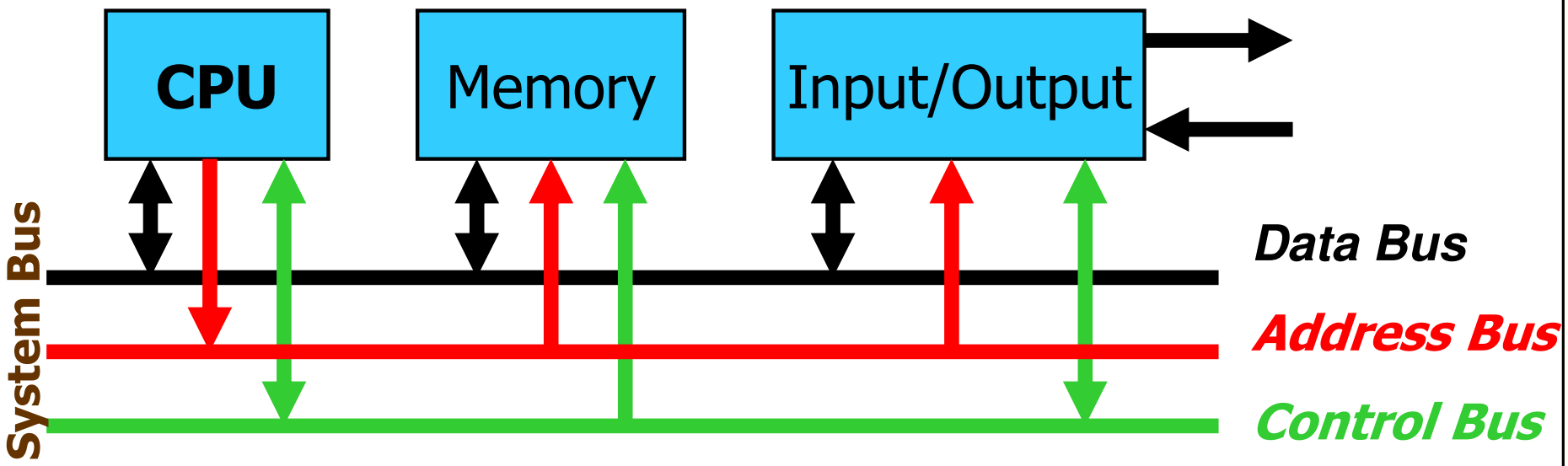
Arquitetura básica de um sistema computacional

- Unidades fundamentais que constituem um computador
  - **CPU** – responsável pelo processamento da informação através da execução de uma sequência de instruções (programa) armazenadas em memória
  - **Memória** – responsável pelo armazenamento de:
    - Programas
    - Dados para processamento
    - Resultados
  - **Unidades de I/O** – responsáveis pela comunicação com o exterior
    - **Unidades de entrada** – permitem a receção de informação vinda do exterior (dados, programas) e que é armazenada em memória
    - **Unidades de saída** – permitem o envio de resultados para o exterior
- Um computador é um sistema digital complexo

Cada um destes blocos é um sistema digital!

Arquitetura básica de um sistema computacional

- Modelo de von Neumann



- **Data Bus:** barramento de transferência de informação (CPU↔memória, CPU↔Input/Output)
- **Address Bus:** identifica a origem/destino da informação (na memória ou nas unidades de input/output)
- **Control Bus:** sinais de protocolo que especificam o modo como a transferência de informação deve ser feita

Arquitetura básica de um sistema computacional

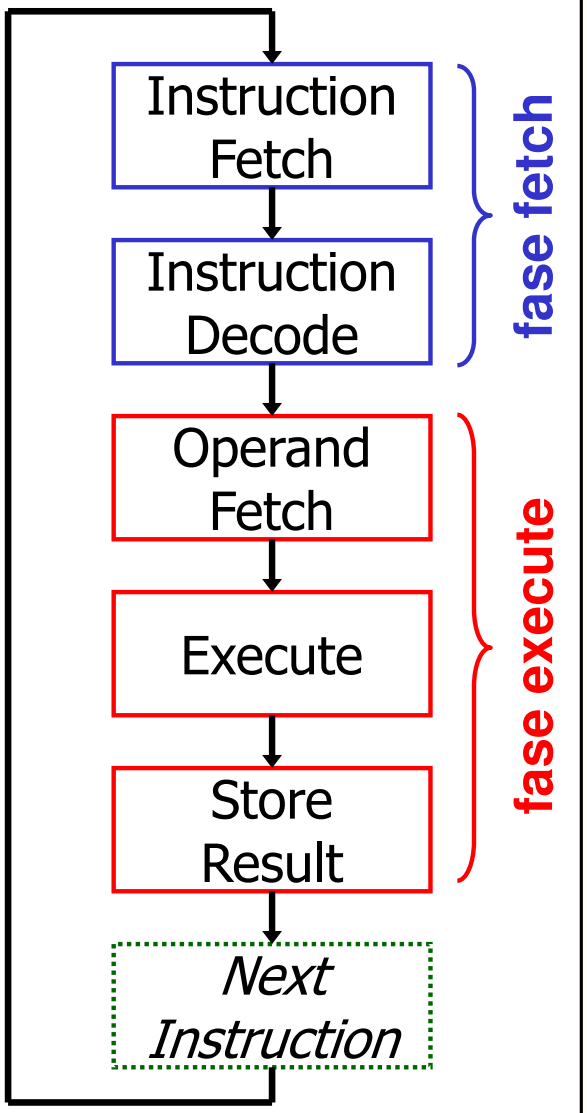
- **Endereço** (*address*) – um número (único) que identifica cada registo de memória. Os endereços são contados sequencialmente, começando em 0
  - Exemplo: o conteúdo da posição de memória 0x2000 é 0x32 – (0x2000 é o endereço, 0x32 o valor armazenado)
- **Espaço de endereçamento** (*address space*) – a gama total de endereços que o CPU consegue referenciar (depende da dimensão do barramento de endereços).
  - Exemplo: um CPU com um barramento de endereços de 16 bits pode gerar endereços na gama: 0x0000 a 0xFFFF (i.e., 0 a 2^{16}-1)
  - Qual o espaço de endereçamento de um processador com um barramento de endereços de 32 bits?

Arquitetura básica do CPU

- **Secção de dados** (*datapath*) – elementos operativos/funcionais para encaminhamento, processamento e armazenamento de informação
  - Multiplexers
  - Unidade Aritmética e Lógica (ALU) – Add, Sub, And, Or...
  - Registos internos
- **Unidade de controlo** – responsável pela coordenação dos elementos do *datapath*, durante a execução de um programa
  - Gera os sinais de controlo que adequam a operação de cada um dos recursos da secção de dados às necessidades da instrução que estiver a ser executada
  - Dependendo da arquitetura, pode ser uma máquina de estados ou um elemento meramente combinatório
- Independentemente da Unidade de Controlo ser combinatória ou sequencial, **o CPU é sempre uma máquina de estados síncrona**

Ciclo-base de execução de uma instrução

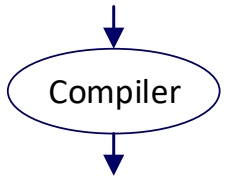
- **Instruction fetch:** leitura do código máquina da instrução (instrução reside em memória)
- **Instruction decode:** descodificação da instrução pela unidade de controlo
- **Operand fetch:** leitura do(s) operando(s)
- **Execute:** execução da operação especificada pela instrução
- **Store result:** armazenamento do resultado da operação no destino especificado na instrução



Níveis de Representação

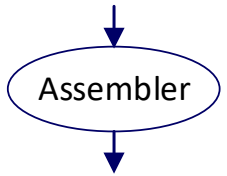
High-level language
program (in C)

```
unsigned char toUpper(unsigned char c)
{
    if(c >= 'a' && c <= 'z')
        return (c - 0x20);
    else
        return c;
}
```



Assembly language
program (for MIPS)

```
toUpper:    addiu $5,$4,-97
            sltiu $1,$5,26
            or    $2,$4,$0
            beq   $1,$0,else
            addiu $2,$4,-32
else:       jr    $31
```



Binary machine language
program (for MIPS)

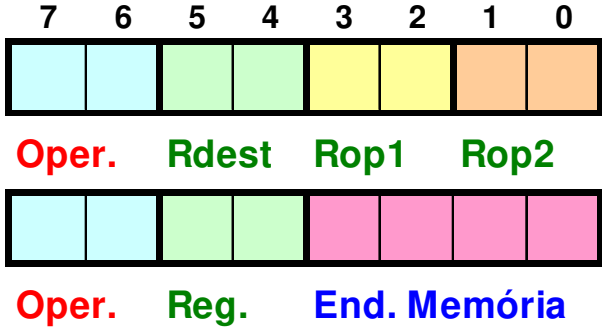
```
00100100100001001111111110011111   (0x2485ff9f)
00101100101000010000000000011010   (0x2ca1001a)
00000000100000000001000000100101   (0x00801025)
00010000001000000000000000000001   (0x10200001)
00100100100001011111111111100000   (0x2482ffe0)
00000011111100000000000000001000   (0x03e00008)
```

Codificação das instruções

- A **codificação de uma instrução**, sob a forma de um número expresso em binário, terá que ter toda a informação de que o CPU necessita para a sua execução
- Qual a operação a realizar?
- Qual a localização dos operandos (se existirem)?
  - podem estar em **registos internos do CPU** ou na **memória externa**. No 1º caso deverá ser especificado o número de um registo; no 2º um endereço de memória
- Onde colocar o resultado?
  - **Registos internos** / **memória**
- Qual a próxima instrução a executar?
  - em condições normais é a instrução seguinte na sequência e, portanto, não é, normalmente, explicitamente mencionada
  - em instruções que **alteram a sequência de execução** a instrução deverá fornecer o endereço da próxima instrução a ser executada

Exemplo - CPU hipotético

Formato de codificação das instruções (8 bits)



Formato 1

Formato 2

Registos Internos do CPU:

- 00 Reg. 0
- 01 Reg. 1
- 10 Reg. 2
- 11 Reg. 3

Operações possíveis:

- 00 Somar o conteúdo de dois registos
- 01 Ler da memória para um registo interno do CPU (LOAD)
- 10 Escrever o conteúdo de um registo interno na memória (STORE)
- 11 Não definida (N.D.)

Exemplo de programa em código máquina para este processador

Qual é a expressão aritmética implementada neste programa?

Hex	Binary	
0x58	01011000	Ler o conteúdo da posição de memória 8 para o registo interno 1
0x79	01111001	Ler o conteúdo da posição de memória 9 para o registo interno 3
0x15	00010101	Somar o conteúdo do reg. 1 c/ o reg. 1 e depositar o result. no reg. 1
0x07	00000111	Somar o conteúdo do reg. 1 c/ o reg. 3 e depositar o result. no reg. 0
0x8A	10001010	Escrever o conteúdo do reg. 0 na posição de memória 10

Arquitetura do Conjunto de Instruções (ISA)

- **Instruction Set**: coleção de todas as operações/instruções que o processador pode executar

- **Instruction Set Architecture** (ISA)
  - *... **os atributos de um sistema computacional tal como são vistos pelo programador**, i.e. a estrutura concetual e o comportamento funcional, de forma distinta e independente da organização do fluxo de informação e dos respetivos elementos de controlo, do desenho lógico e da implementação física.* — Amdahl, Blaaw, and Brooks, 1964

- Arquitetura de Computadores =
  Arquitetura do Conjunto de Instruções (ISA) +
  Organização da Máquina

Arquitetura do Conjunto de Instruções (ISA)

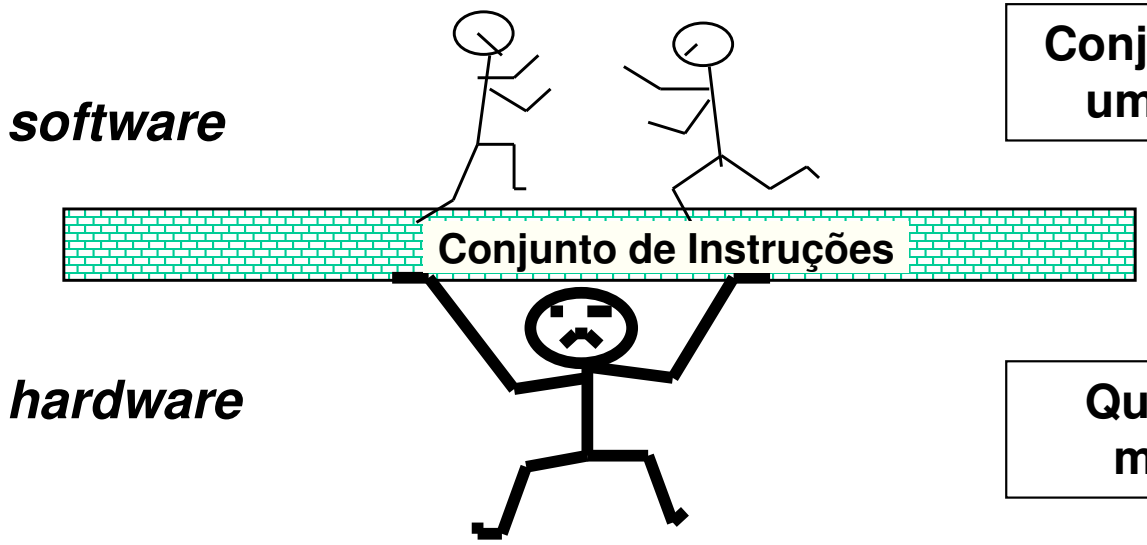
- Também designada por "modelo de programação"
- Descreve tudo o que o programador necessita de saber para programar corretamente, em *assembly*, um determinado processador
- Uma importante abstração que representa a **interface entre o nível mais básico de software e o hardware**
- Descreve a funcionalidade, de forma independente do hardware que a implementa. Pode assim falar-se de "**arquitetura**" e "**implementação de uma arquitetura**"
- Exemplo em que a mesma arquitetura do conjunto de instruções tem 2 implementações físicas distintas:
  - Processadores AMD compatíveis com Intel x86

Arquitetura do Conjunto de Instruções (ISA)

- Alguns exemplos de ISAs:
  - MIPS
  - ARM (Nintendo DS, iPod, Canon PowerShot, smartphones, ...)
  - Intel x86 (PCs, MACs)
  - PowerPC
  - Cell (playstation 3)

Arquitetura do Conjunto de Instruções (ISA)

- Requisitos básicos da Arquitetura do Conjunto de Instruções:
  - Fácil de entender e programar
  - Desenvolvimento de compiladores eficientes
  - Implementação simples e eficiente em hardware
  - Com o melhor desempenho possível
  - Eficiente do ponto de vista energético
  - Com o menor custo possível



Conjunto de Instruções: um Interface Crítico!

Qual dos níveis será mais fácil mudar?

Classes de instruções

- Um dada arquitetura pode ter um ISA com centenas de instruções
- É possível, no entanto, considerar a existência de um grupo limitado de classes de instruções comuns à generalidade das arquiteturas
- Classes de instruções:
  - **Processamento**
    - Aritméticas e lógicas
  - **Transferência de informação**
    - Cópia entre registos internos e entre registos internos e memória
  - **Controlo de fluxo de execução**
    - Alteração da sequência de execução (estruturas condicionais, ciclos, chamadas a funções,…)

Questões

- O que é um endereço?
- O que é o espaço de endereçamento de um processador?
- Como se organiza internamente um processador? Quais são os blocos fundamentais da secção de dados? Para que serve a unidade de controlo?
- O que é o conceito "stored-program"?
- Como se codifica uma instrução? Que informação fundamental deverá ter o código de uma instrução?
- O que é o ISA?
- Quais são as classes de instruções que agrupam as instruções de uma arquitetura?