

# Curso: Engenharia de Computação

**Arquitetura de Computadores**

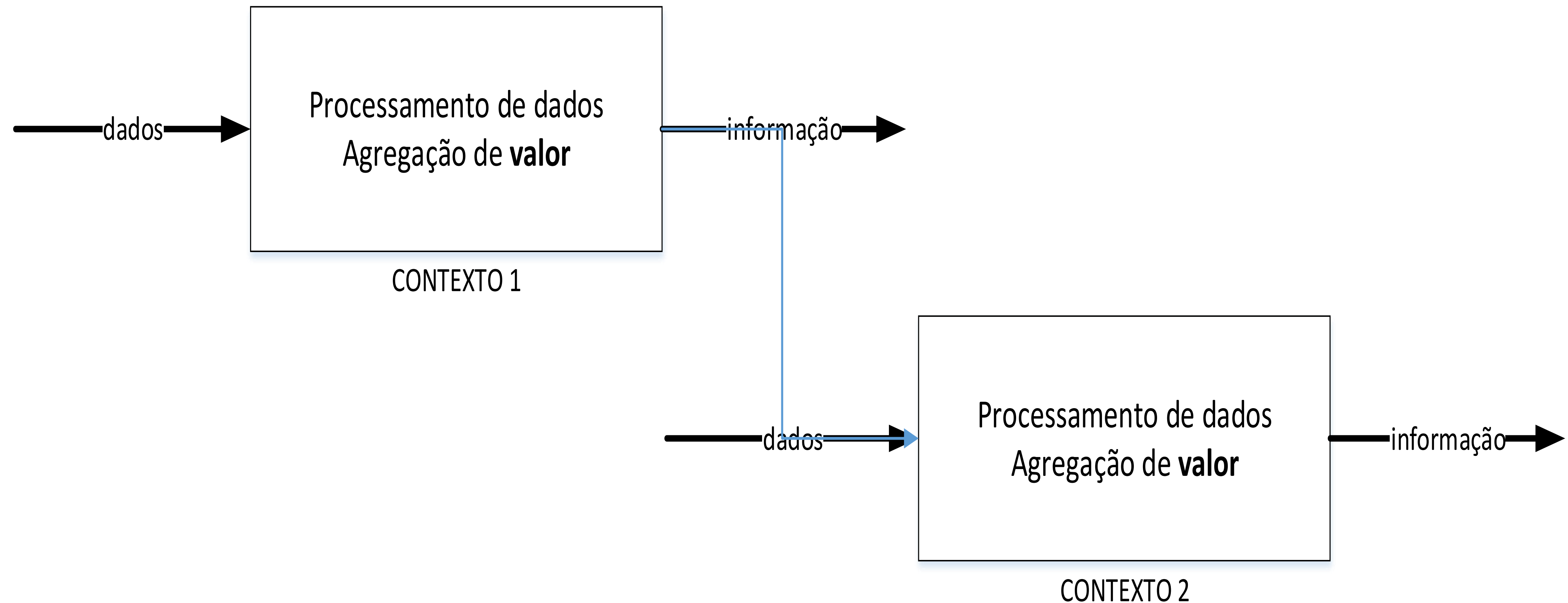
Prof. Clayton J A Silva, MSc  
clayton.silva@professores.ibmec.edu.br



# Ferramentas

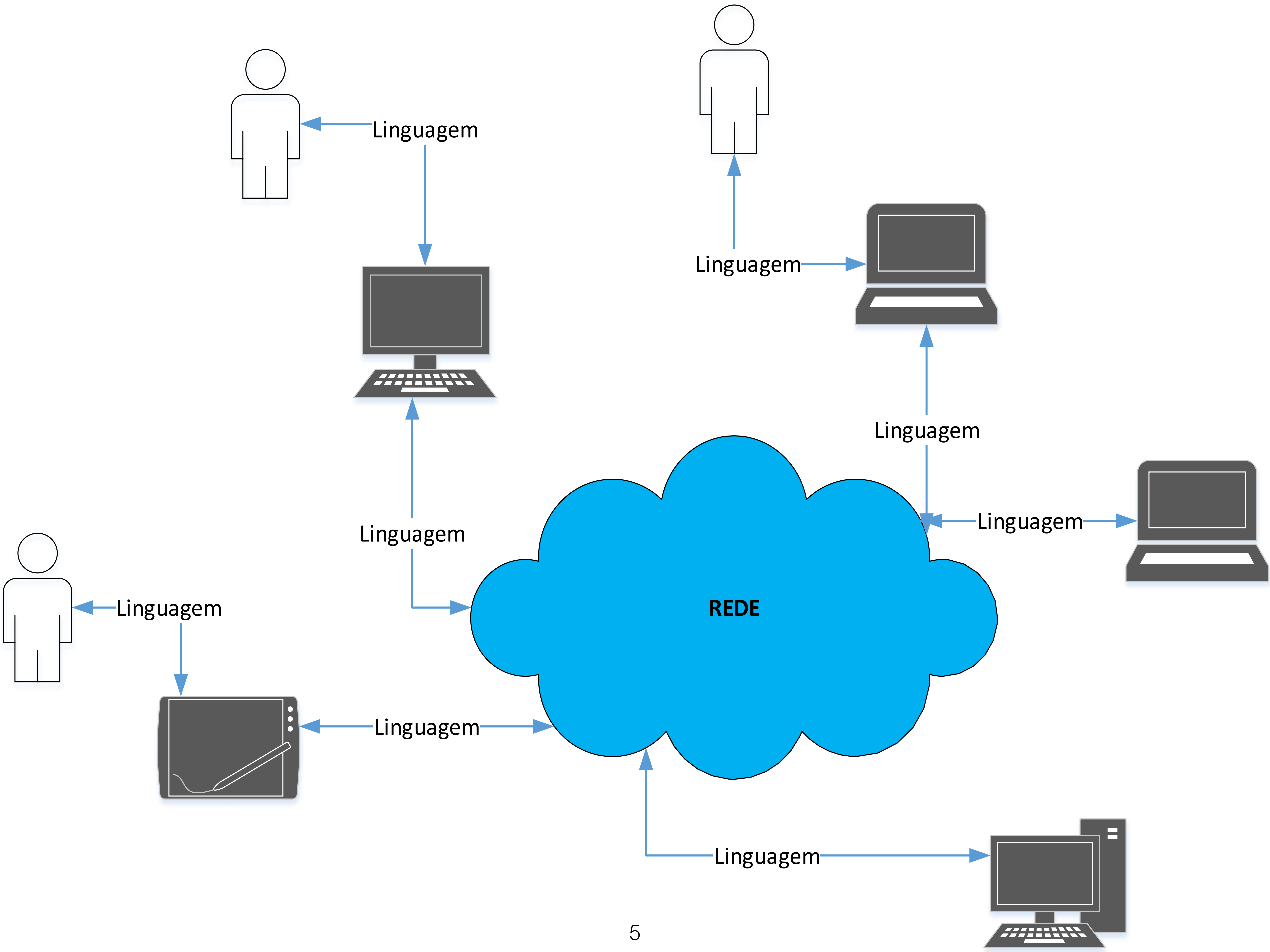
- [Logisim](#)
- [EasyEDA](#)
- [Microchip Studio](#)
- [Arduíno IDE](#)

# Dados, linguagens e programas



# Dados, linguagens e programas

- dados x informação: contexto
- **Programa:** conjunto finito de instruções escritas em uma das linguagens entendidas pelas máquinas (linguagens de programação) para manipulação de dados.
- **Linguagem de Programação:** sistema de símbolos, sinais ou objetos instituídos como signos – código – para comunicação, que obedece a regras de sintaxe e semântica.



# Organização estruturada de computadores

- **Lacuna:** As instruções primitivas de um sistema computacional, escritas em **linguagem de máquina**, precisam ser **simples**, portanto diferentes das instruções compreendidas pelo homem, nas linguagens naturais
- **Solução técnica:** Escrever as instruções em linguagem próxima da linguagem natural e **convertê-las** para as instruções na linguagem de máquina.
- **Abordagens:** **tradução ou interpretação.**

# tradutores e interpretadores

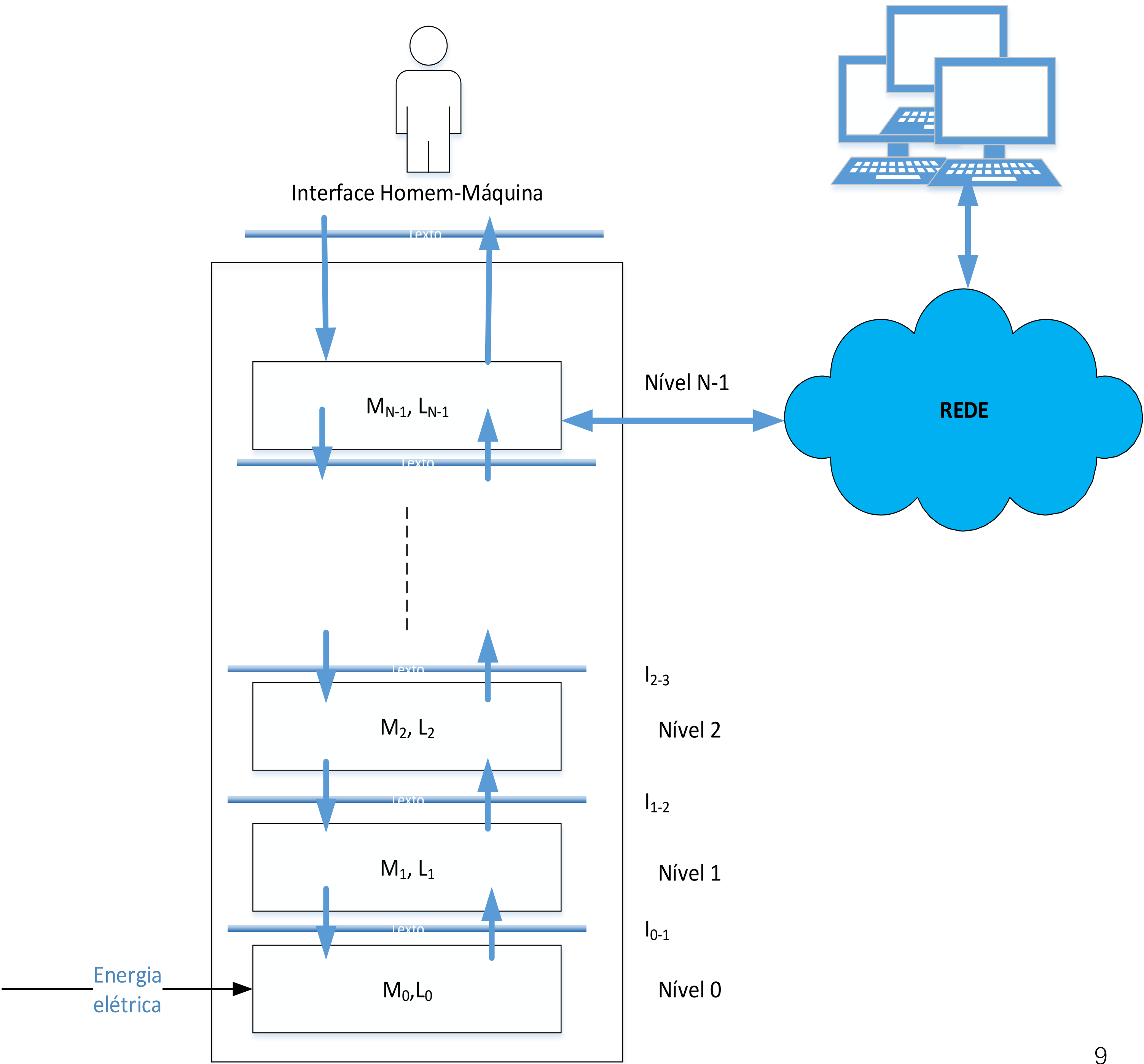
- Na **tradução**: instruções e dados do código da máquina de origem são substituídos pelos equivalentes no código da máquina de destino, em que o programa será executado.
- Na **interpretação**: instruções e dados da linguagem da máquina de origem são mantidos conforme o código original, convertendo-se para a linguagem da máquina virtual de destino conforme a necessidade da execução do código de origem.

# Organização estruturada de computadores

- A organização de um computador pode ser entendida como um conjunto de máquinas  $M_i$ , cada uma com uma Linguagem de Programação  $L_i$ , para as quais são escritos os programas
- Cada máquina ocupa uma camada ou **nível  $N$**
- A distância entre as linguagens das máquinas não pode ser muito grande, sob pena do processo de tradução ou interpretação ser muito complexo.



# Modelo de máquina de níveis



## Modelo de máquina de níveis

- Sistema de camadas ou **níveis**
- $N$  níveis,  $i=0$  a  $N-1$
- Cada nível  $i$  executa **programas** que contêm **instruções** de uma linguagem ( $L_i$ )
- Cada instrução de um nível  $i$  pode ser **convertida** em instruções de um nível  $i-1$  (inferior)
- São necessárias interfaces de comunicação: **tradutores** ou **interpretadores**

# Modelo de máquina de seis níveis

**Nível 0 – nível de lógica digital.** Mais elementar. Utiliza portas lógicas (*gates*), que podem ser combinadas em série e em paralelo. Os dados são sinais elétricos, abstrações dos **bits** (*binary digits*).

**Nível 1 – nível de microarquitetura.** Circuitos especializados: registradores; Unidade Lógica e Aritmética; barramentos internos – caminho de dados; Unidade de Controle, constituída de microprogramas ou de circuitos eletrônicos.

**Nível 2 – nível ISA (*Instructions Set Architecture*).** Nível do processador. As instruções são definidas pelo fabricante.

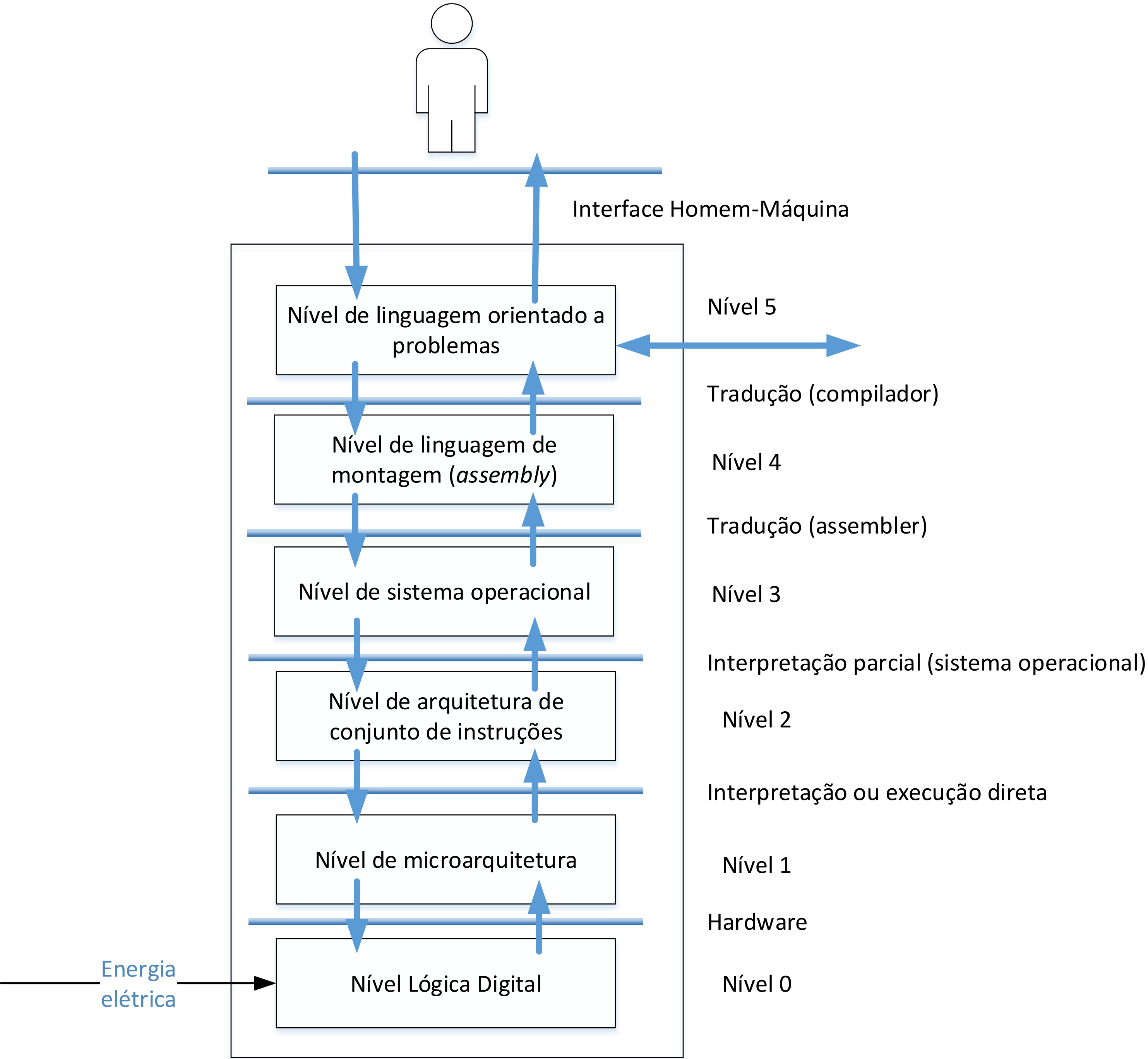
# Modelo de máquina de seis níveis

**Nível 3 – nível de sistema operacional.** Possui instruções próprias, assim como usa instruções do próprio nível ISA. Considera-se como um nível híbrido.

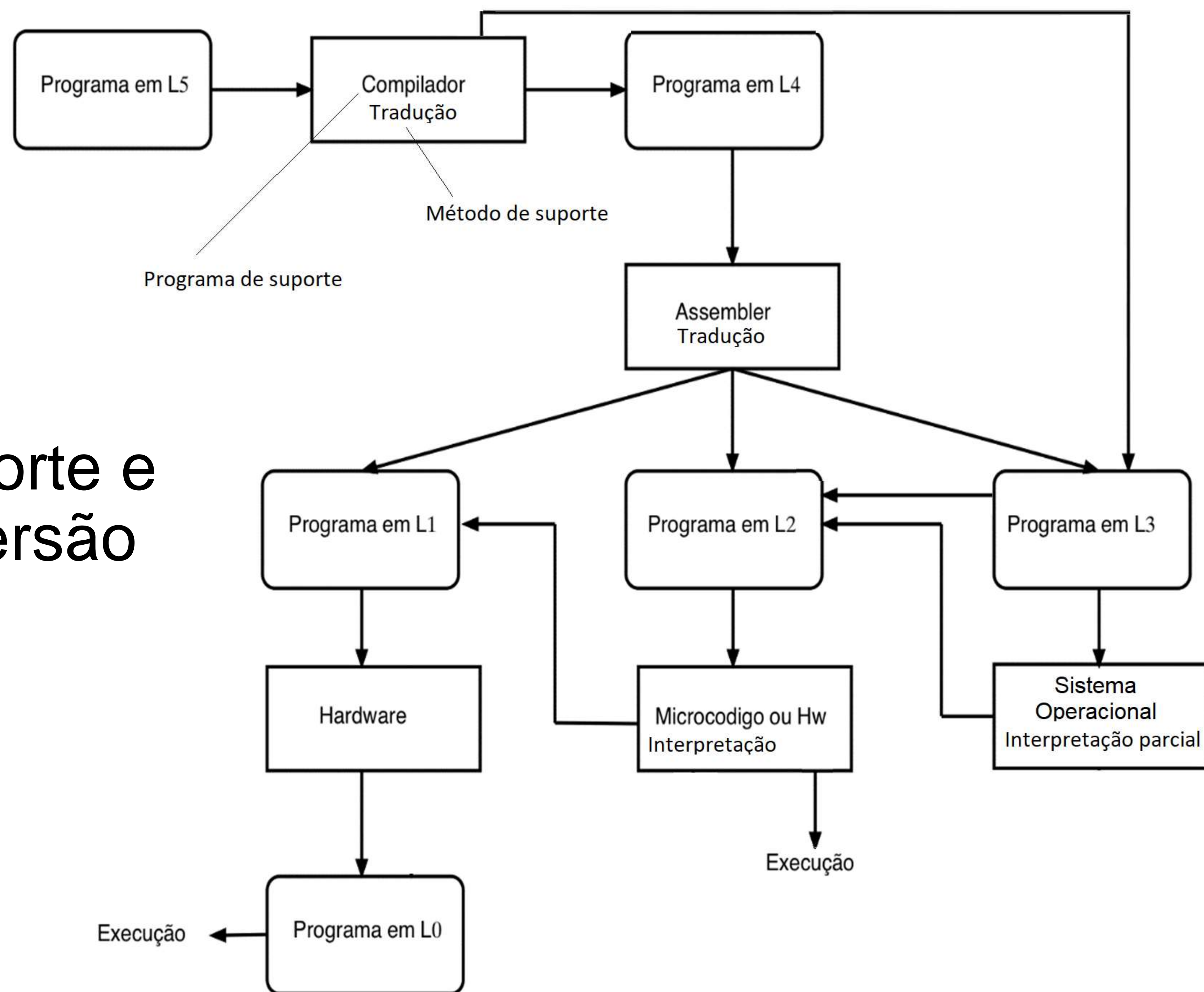
**Nível 4 – nível de linguagem de montagem (*assembly*).** Contempla instruções simbólicas que podem ser executadas pelos níveis inferiores (1, 2 e 3), dispensando o conhecimento detalhado dos elementos físicos, usando o programa *assembler* (montador).

**Nível 5 – nível de aplicações.** Usam as chamadas linguagens de alto nível, traduzidas para a linguagem de montagem (*assembly*) por programas chamados compiladores.

# Modelo de máquina de seis níveis

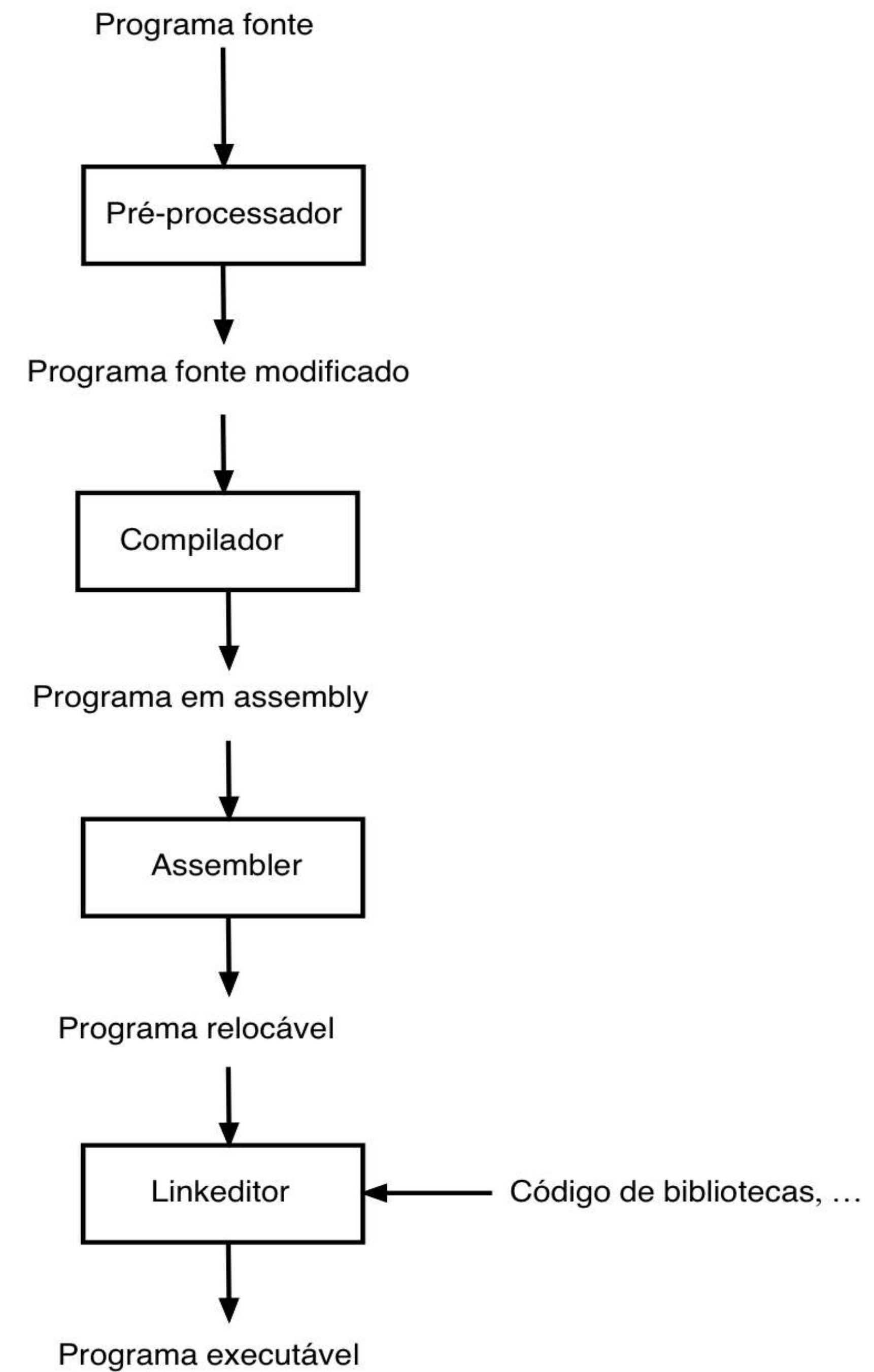


# Programa de suporte e método de conversão





# O compilador





Origem: Wikipédia, a enciclopédia livre.

O **computador IAS** foi o primeiro **computador** eletrônico construído pelo **Instituto de Estudos Avançados de Princeton** (IAS). O artigo descrevendo o projeto do computador IAS foi editado por **John von Neumann**, um professor de matemática da **Universidade de Princeton** e do Instituto de Estudos Avançados. O computador foi construído de 1942 até 1 de junho de 1952 quando se tornou operacional.<sup>[1]</sup>

O computador era uma máquina **binária** com uma **palavra** para 1024 palavras (5,1 kilobytes). Números negativos (AC) e Multiplicador/Quociente (MQ). Embora alguns computadores de memória, isto já havia sido implementado quatro anos antes.

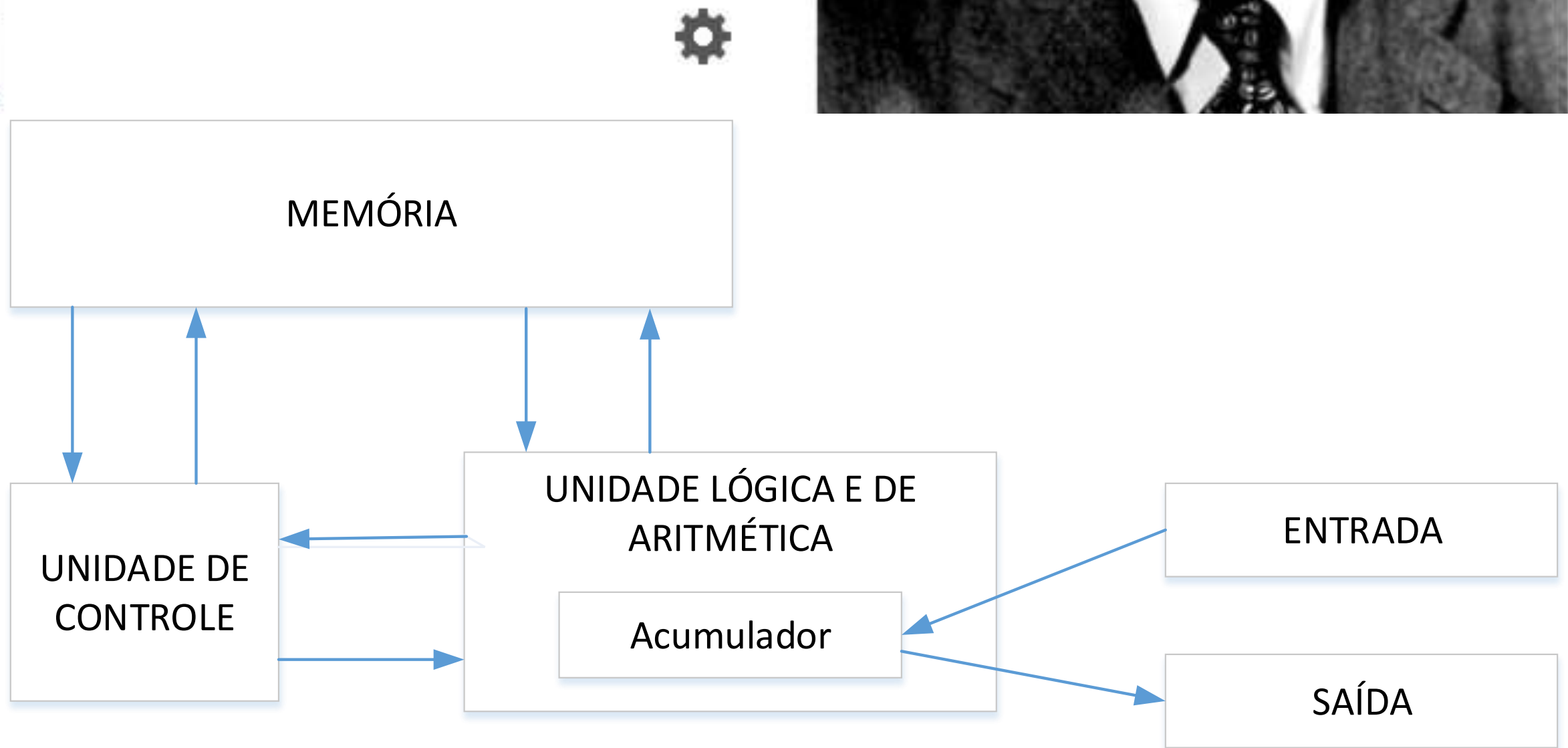
**John von Neumann**, nascido **Margittai Neumann János Lajos** foi um matemático húngaro de origem judaica, naturalizado estadunidense.



verão de 1951 até 10 de junho de 1952. A memória tinha capacidade de 100 palavras. Tradutores: Acumulador e dados numa única palavra.

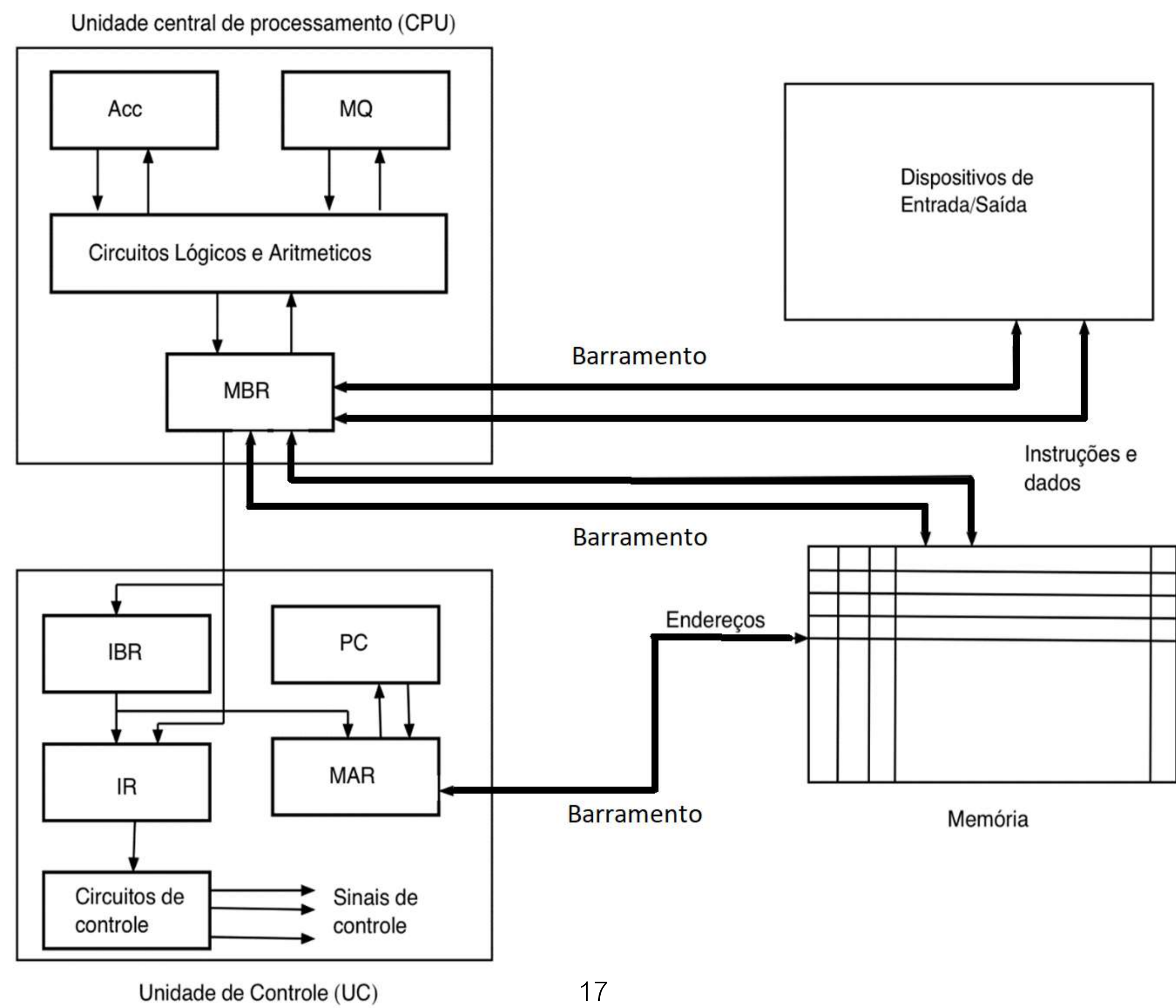
## Referências

- ↑ «IAS Computer»  (em inglês). National Museum of American History. Consultado em 10 de junho de 2012
- ↑ «The Manchester Small Scale Experimental Machine»  em 4 de junho de 2012

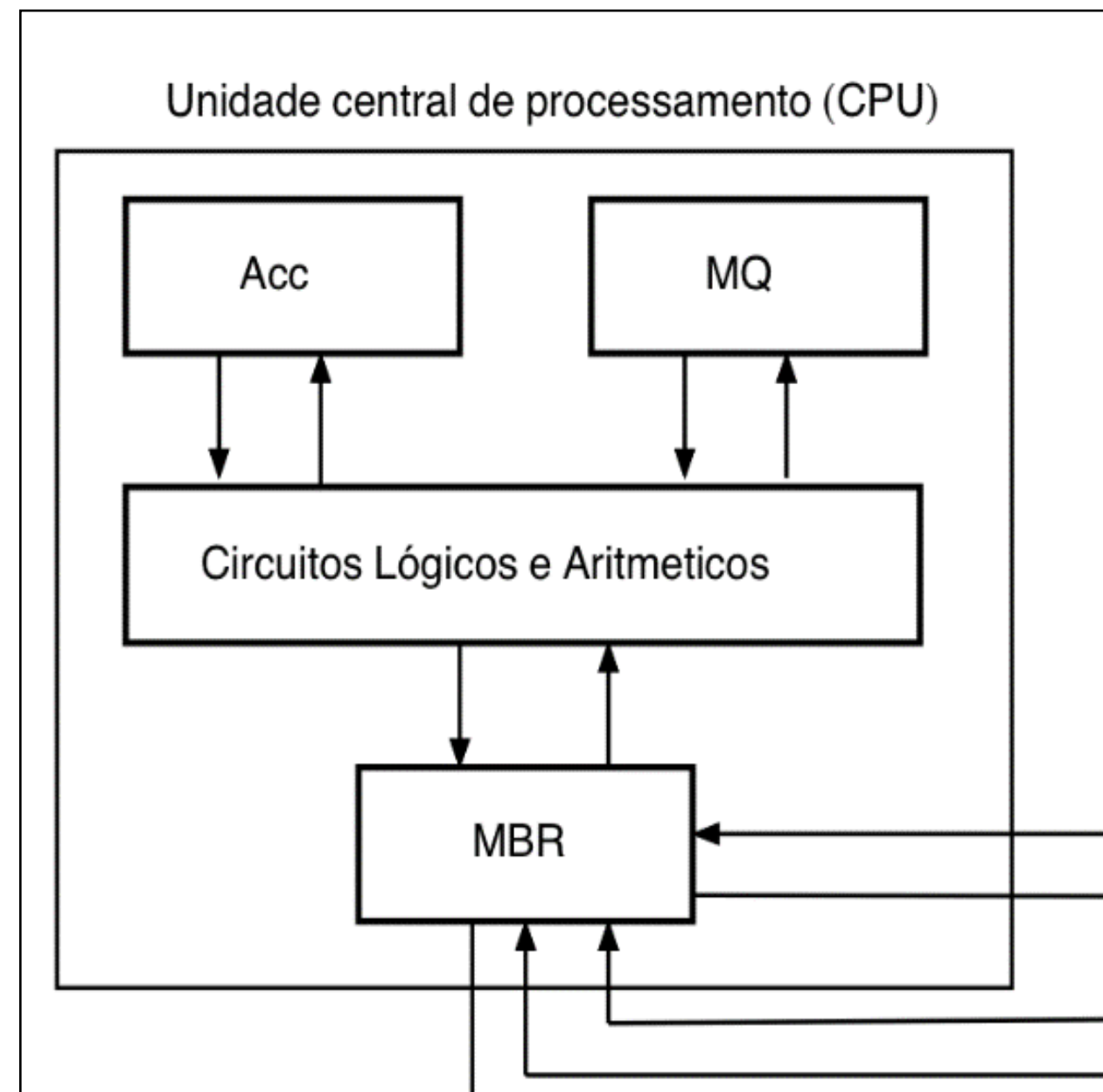




# Estrutura detalhada da máquina de von Neumman



# Elementos da arquitetura de von Neumman

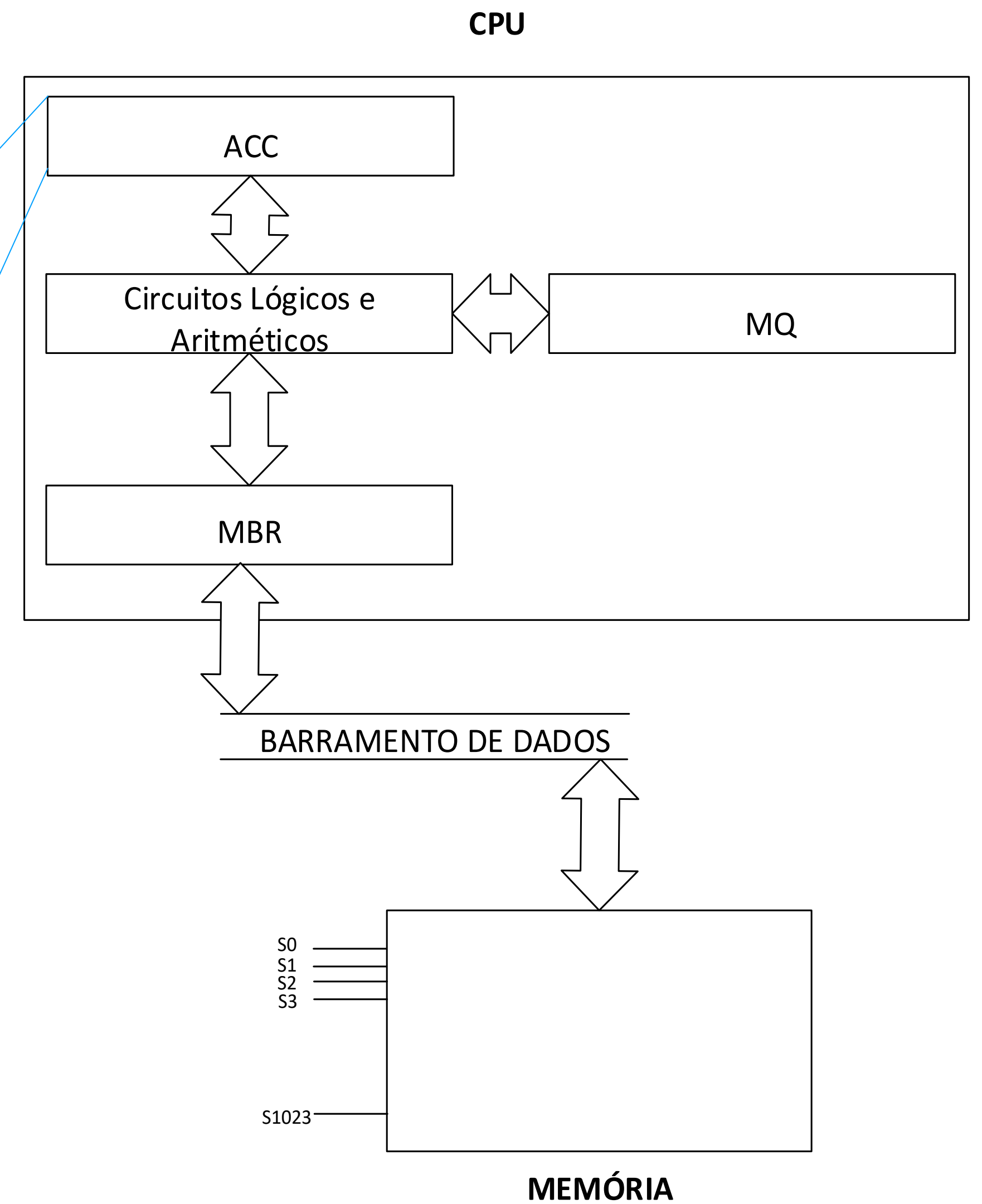
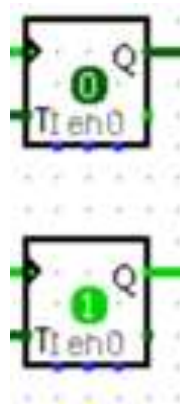


- **Registrador Temporal da Memória (MBR, *Memory Buffer Register*)**: capacidade de armazenar 40 bits e contém uma **palavra** (**número** ou **par de instruções**) a ser **lida** ou **escrita** na memória.
- **Acumulador (Acc) e Quociente de Multiplicação (MQ, *Multiplier Quotient*)**: capacidade de armazenar 40 bits e armazenam temporariamente os **operandos** e o resultado das operações realizadas pelos circuitos lógicos e aritméticos da ULA. Em operações com mais de 40 bits, o Acc armazena os 40 bits mais significativos e o MQ armazena os 40 bits menos significativos.

# Registradores

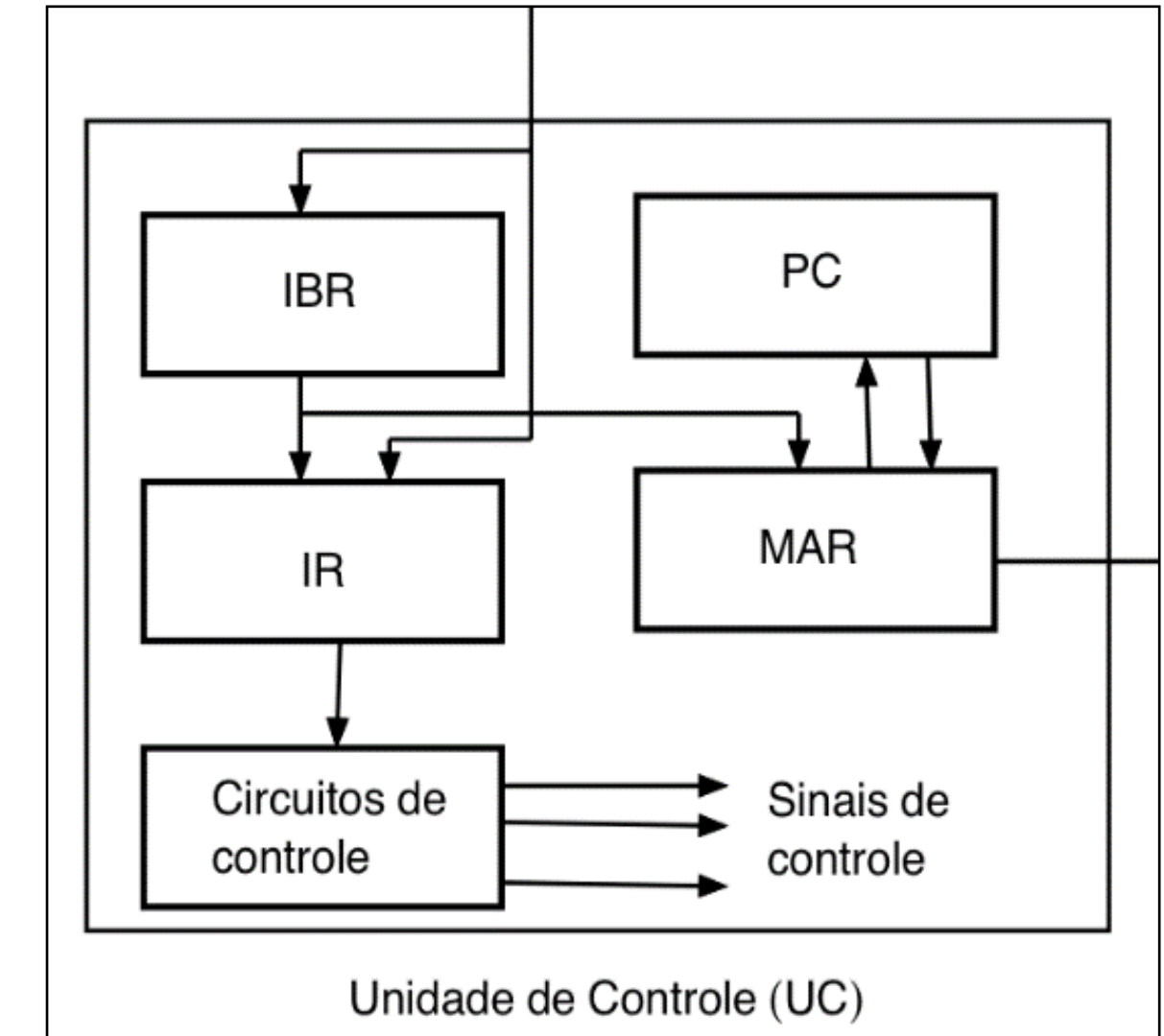
- Dispositivos que armazenam algumas unidades de bits.
- Possuem tempo de resposta muito baixa.

*Flip-flops*



# Elementos da arquitetura de von Neumman

- **Contador do Programa (PC, *Program Counter*):** **contador** de 10 bits e contém o endereço do próximo par de instruções. **Incrementa automaticamente 1 bit** em cada instrução executada.
- **Registrador de Endereçamento à Memória (MAR, *Memory Address Register*):** capacidade de armazenar 12 bits e contém o endereço da palavra.
- **Registrador Temporário de Instruções (IBR, *Instruction Buffer Register*):** capacidade de armazenar 20 bits e contém o **código da instrução (opcode)** da instrução à direita da palavra.
- **Registrador de Instruções (IR, *Instruction Register*):** capacidade de armazenar 8 bits e contém o código da instrução (opcode) que está sendo executada.

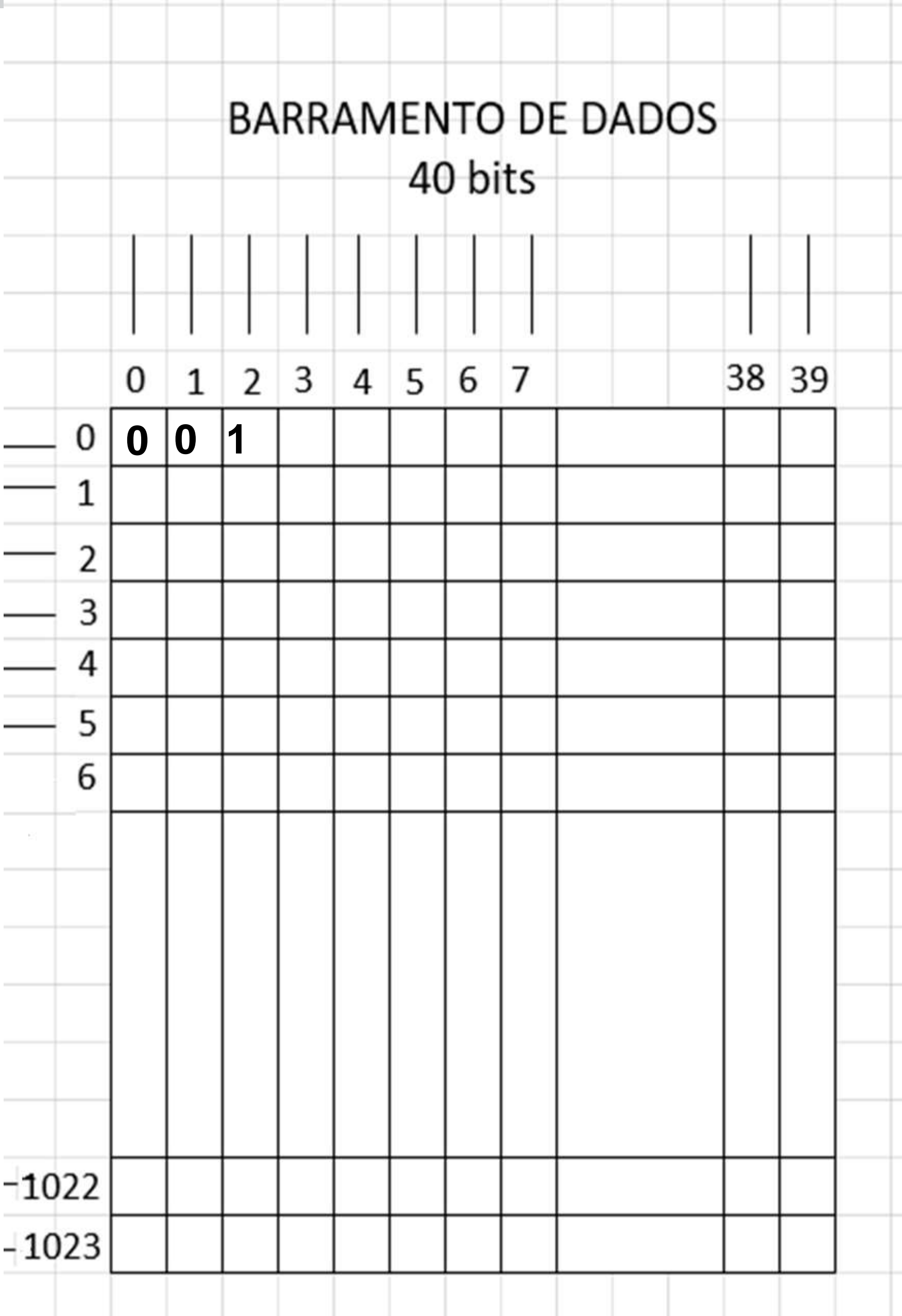




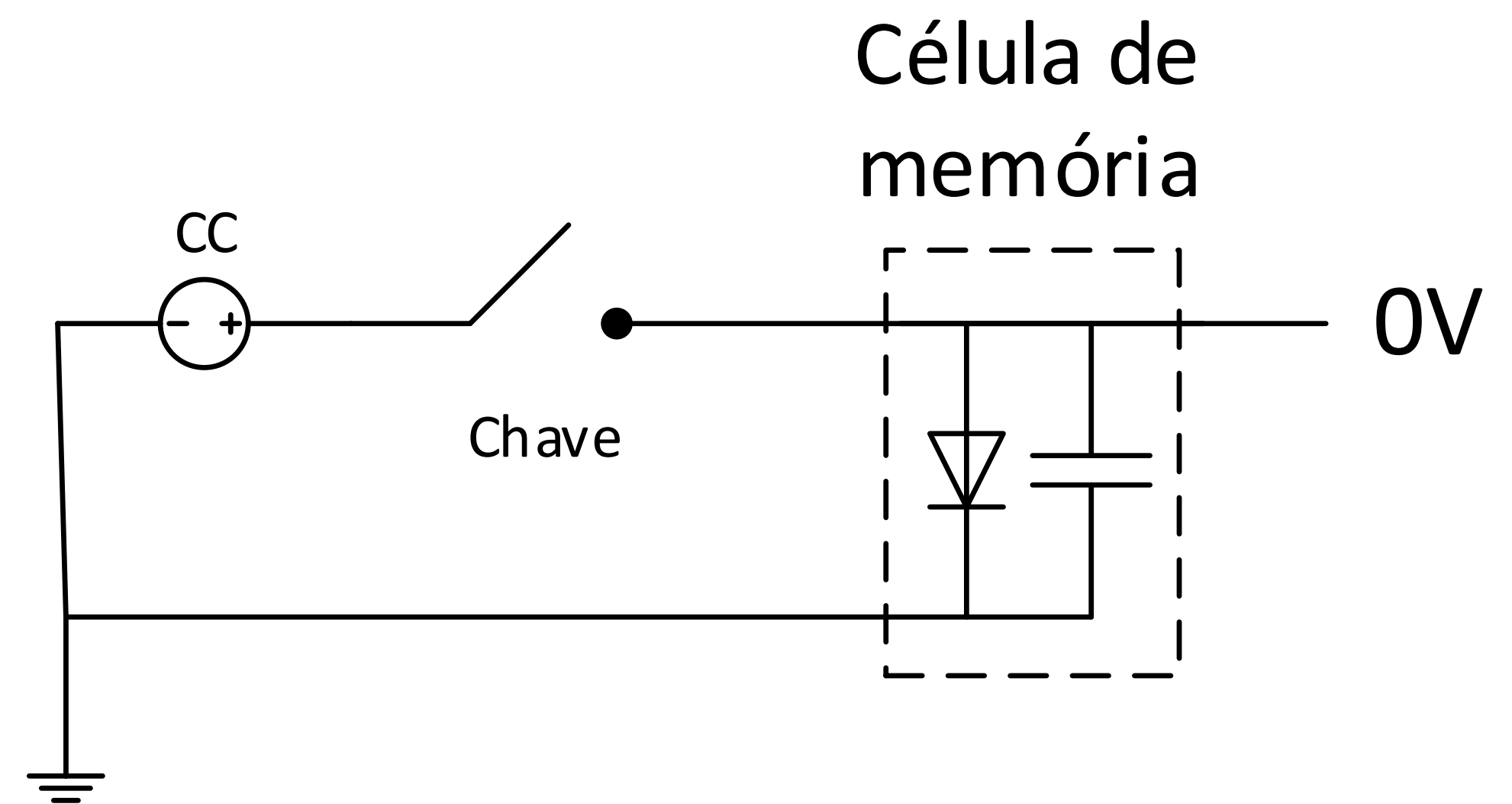
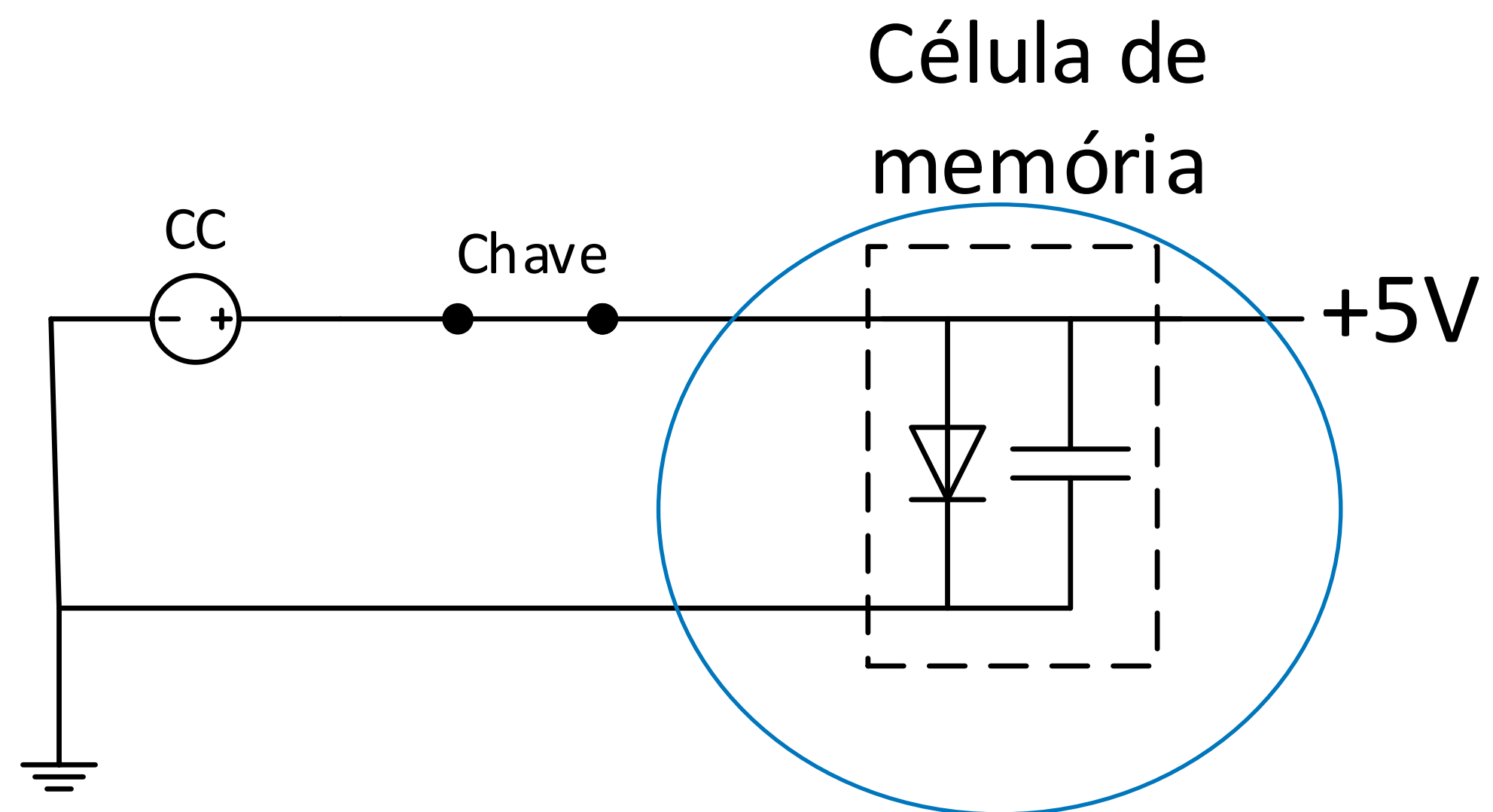
# Elementos da máquina de von Neumman

- **Memória (*Memory*):** capacidade de armazenar, em cada linha, 1024 palavras (unidade de informação) de 40 bits cada uma.
- **Dispositivos de Entrada/Saída (*Input/Output – I/O*):** leitura e escrita de dados dos dispositivos externos da/para memória.
- **Barramento (*Bus*):** meio de comunicação para tráfego dos bits entre os elementos do sistema, para trafegar **dados, endereços** ou sinais de **controle**.

# As células de memória

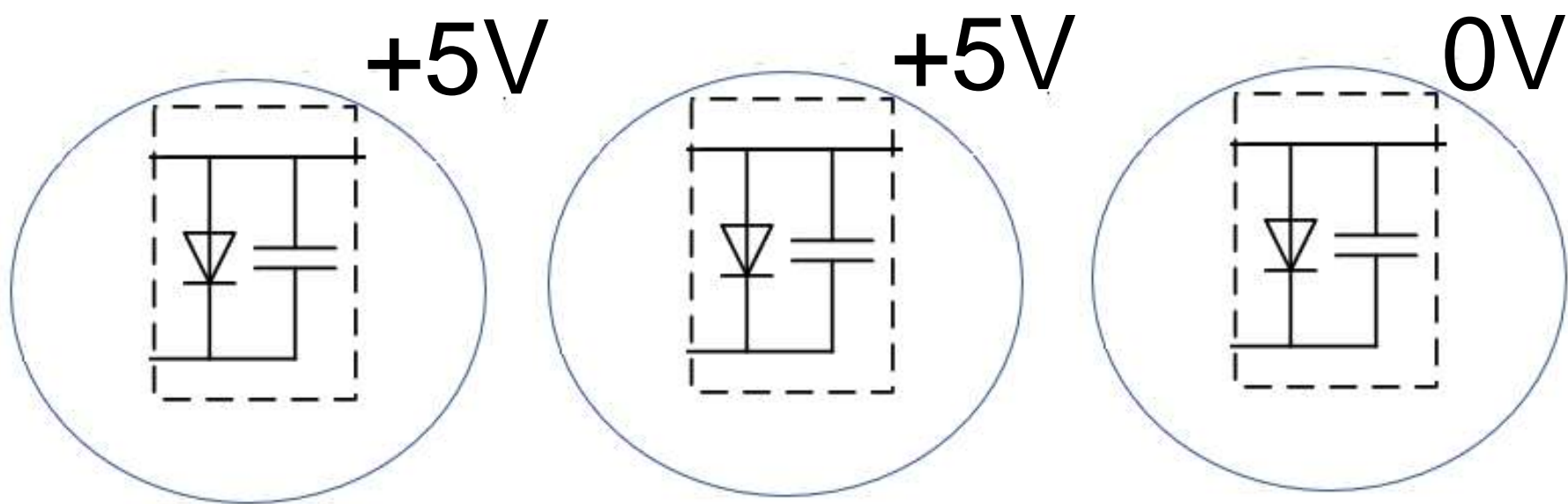


# célula de memória



# Uma linha de memória

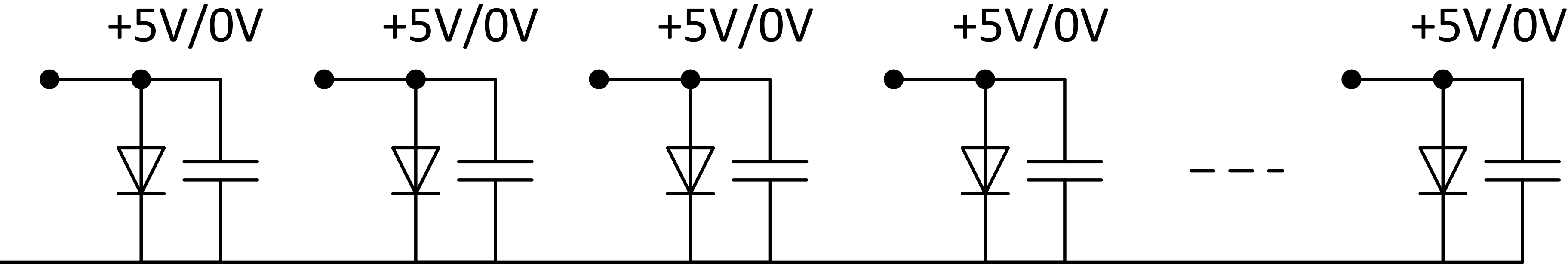
com **três células**



com **3 bits**

b0	b1	b2
1	1	0

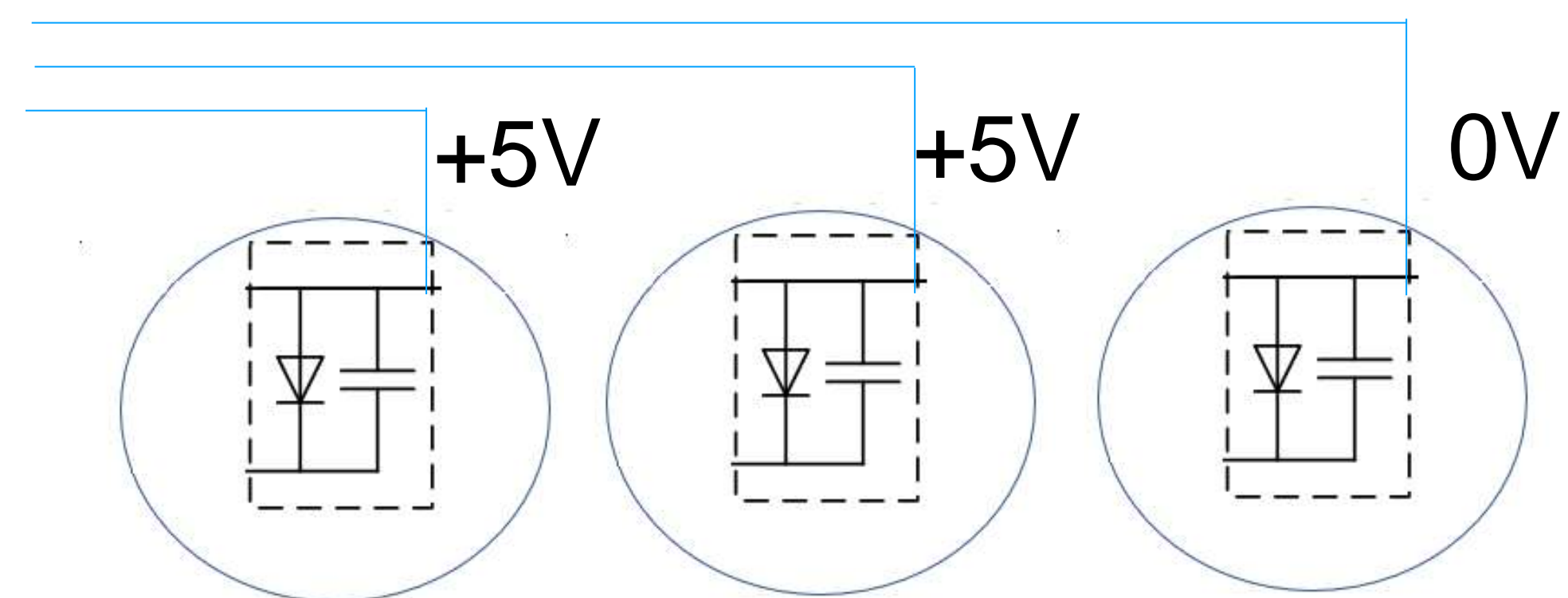
com **palavra** de várias células



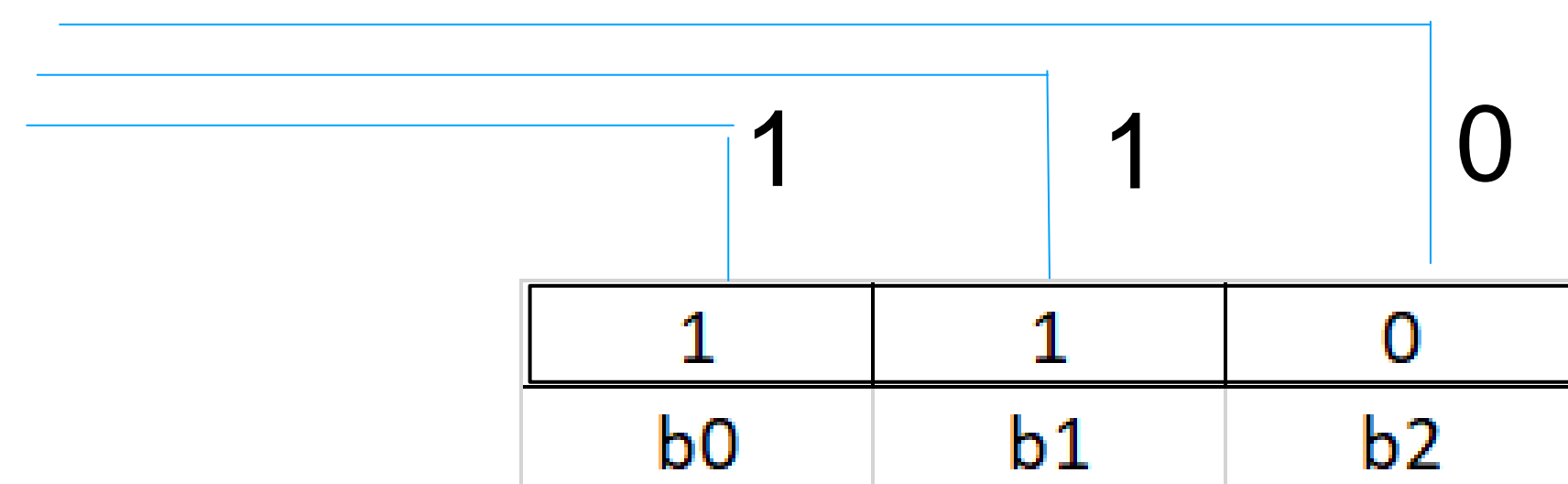


# Conexão da linha ao barramento de dados

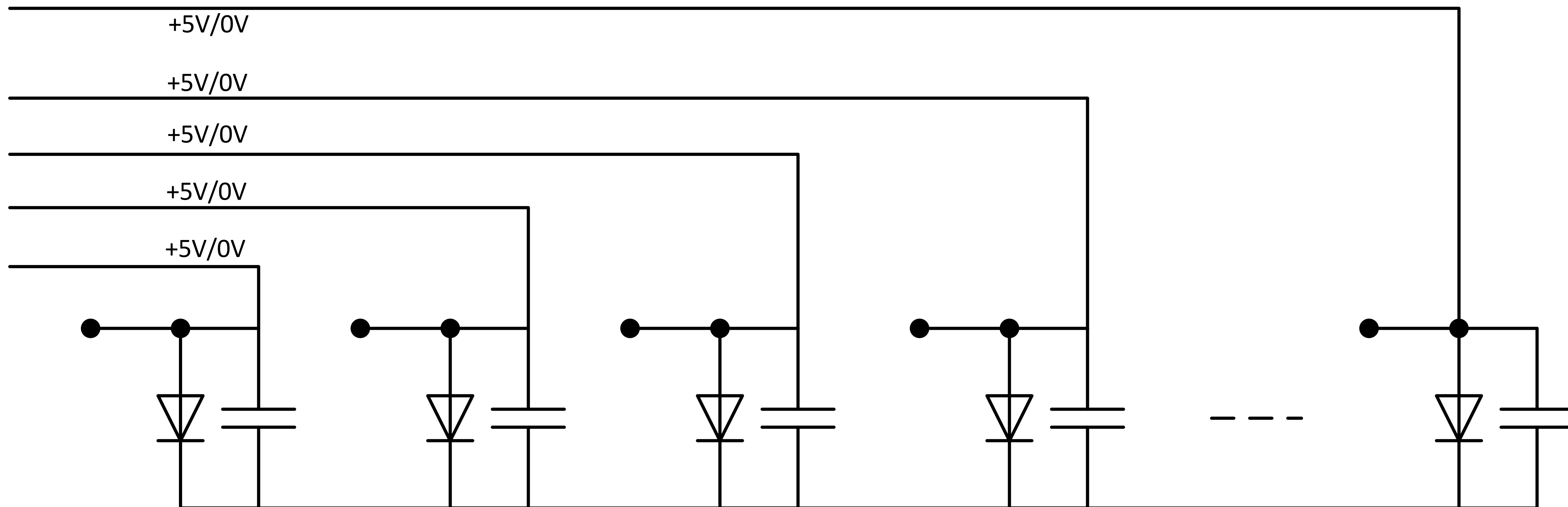
três células de memória



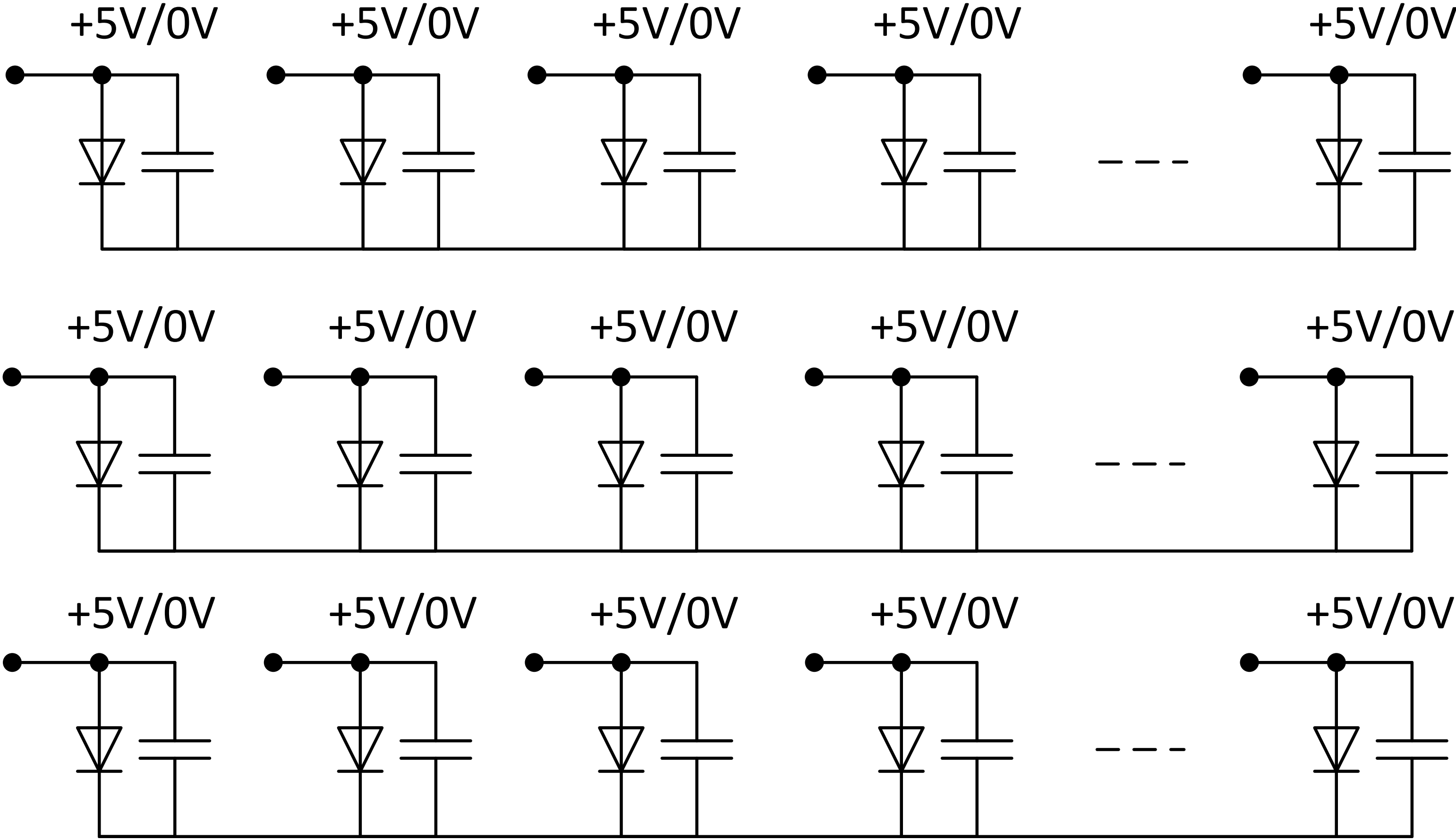
palavra de 3 bits



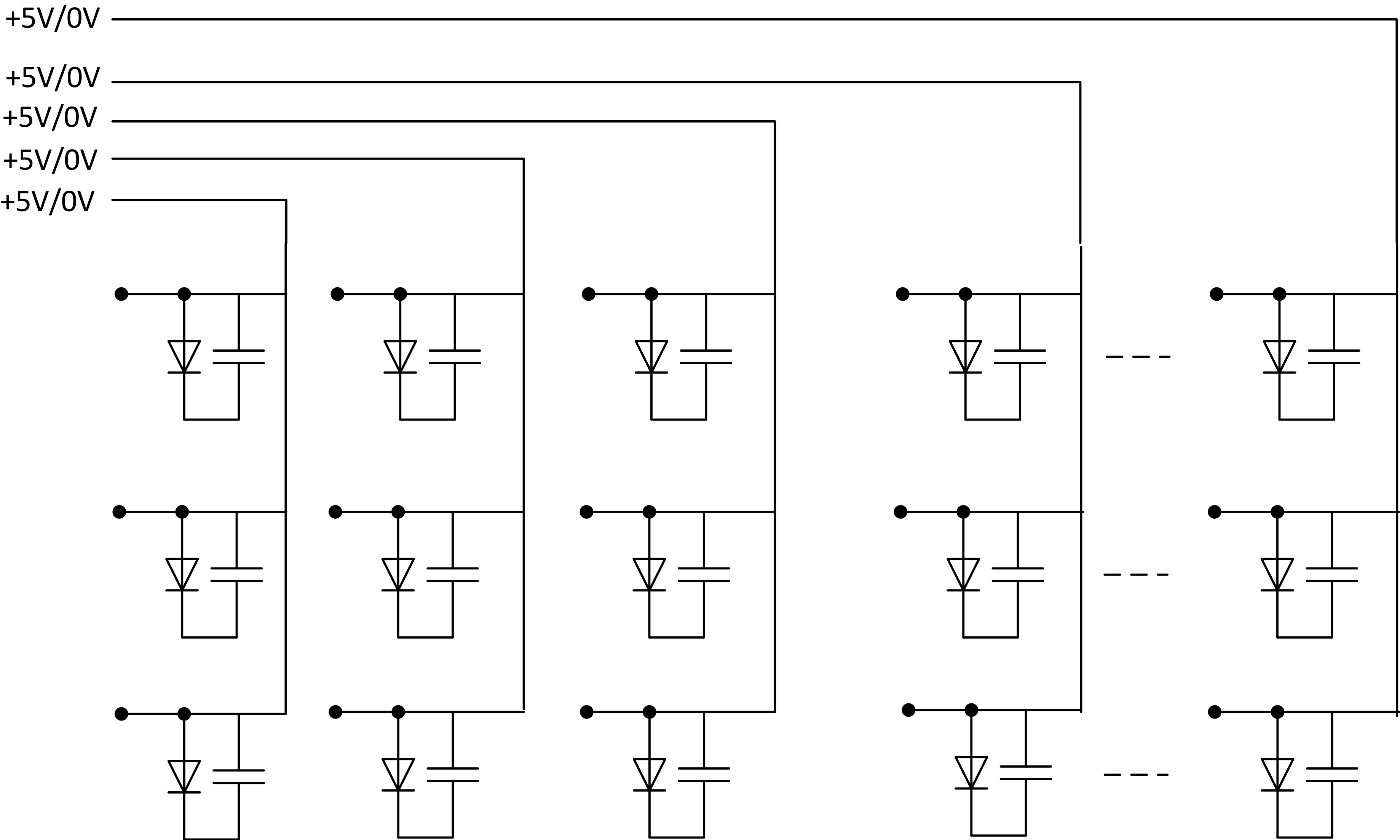
# 1 linha com células de memória de uma **palavra** conectadas ao barramento de dados



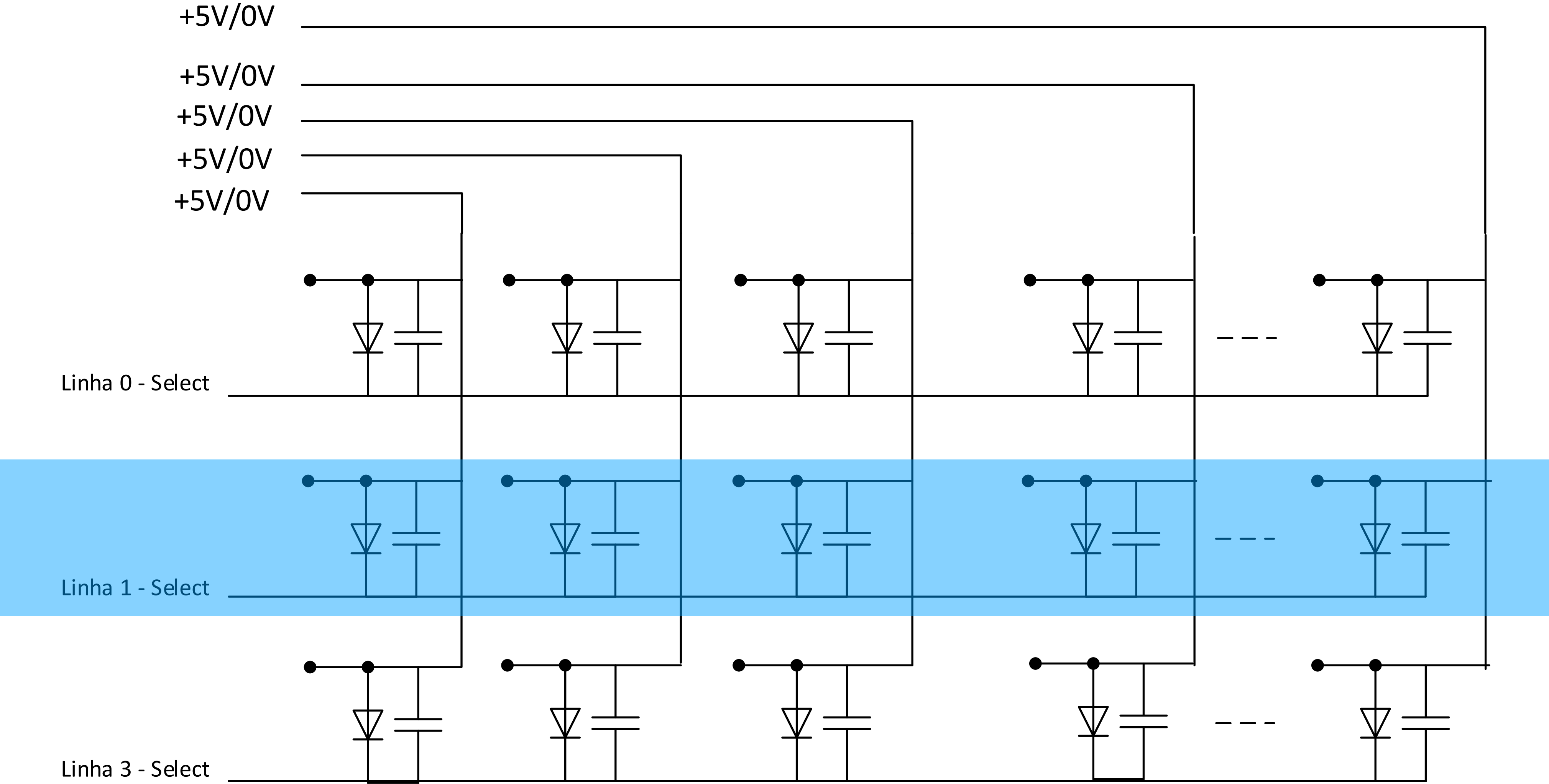
# 3 linhas com células de memória de uma **palavra** conectadas ao barramento de dados



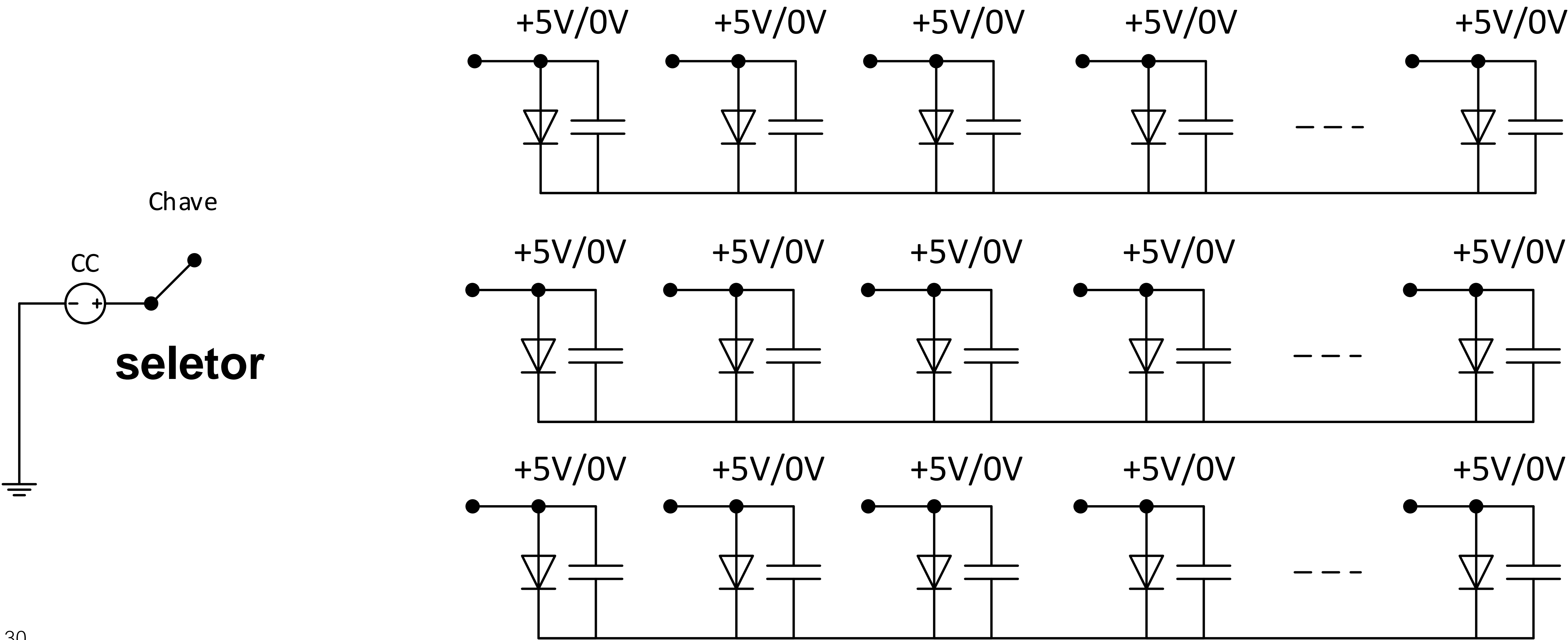
# Conexão das linhas ao barramento de dados



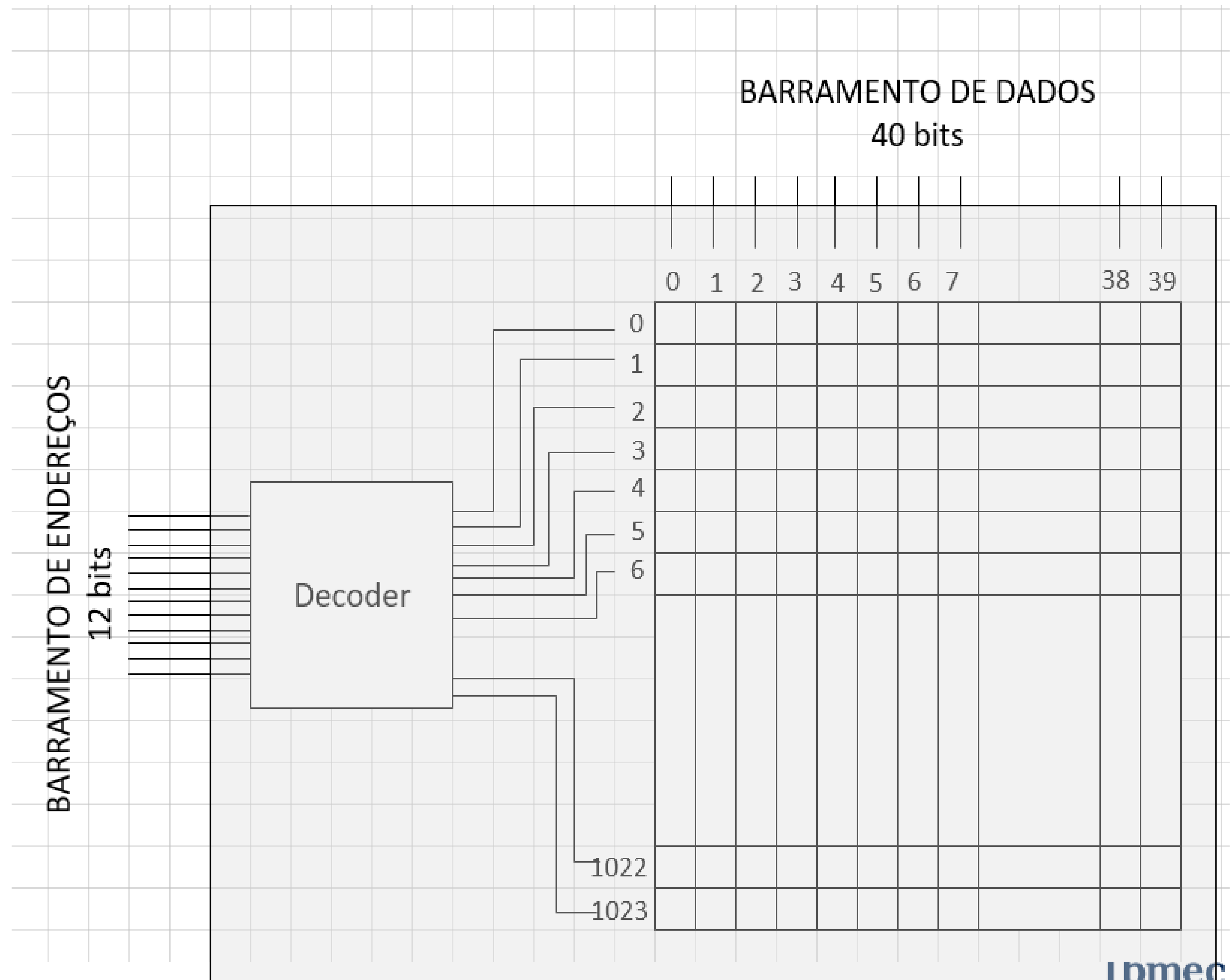
# Seleção de uma linha para o barramento de dados



# Seleção de uma linha para o barramento de dados



# Memória com decoder



# Palavra de Memória

## Linha de dados

<i>b0</i>	<i>b1</i>	<i>b2</i>	<i>b3</i>	<i>b4</i>	<i>b5</i>	<i>b6</i>	<i>b7</i>	<i>b8</i>	<i>b9</i>	<i>b10</i>	<i>b11</i>	<i>b12</i>	<i>b13</i>	<i>b14</i>	<i>b15</i>	<i>b16</i>	<i>b17</i>	<i>b18</i>	<i>b19</i>	<i>b20</i>	<i>b21</i>	<i>b22</i>	<i>b23</i>	<i>b24</i>	<i>b25</i>	<i>b26</i>	<i>b27</i>	<i>b28</i>	<i>b29</i>	<i>b30</i>	<i>b31</i>	<i>b32</i>	<i>b33</i>	<i>b34</i>	<i>b35</i>	<i>b36</i>	<i>b37</i>	<i>b38</i>	<i>b38</i>	<i>b39</i>	
1	1	1	0	0	1	1	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	

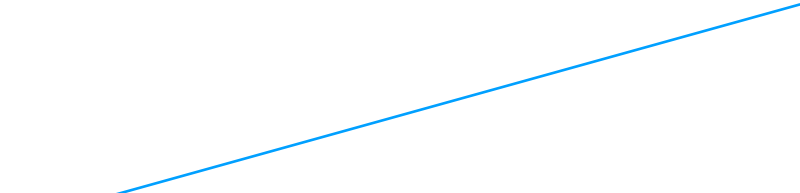
- Cada **linha** de dados aponta para uma **palavra**, que ocupa um endereço de memória
- Os endereços são **inequívocos**, ou seja, não existem dois endereços iguais



# Memória

Número em representação binária

Bit de sinal



<i>b0</i>	<i>b1</i>	<i>b2</i>	<i>b3</i>	<i>b4</i>	<i>b5</i>	<i>b6</i>	<i>b7</i>	<i>b8</i>	<i>b9</i>	<i>b10</i>	<i>b11</i>	<i>b12</i>	<i>b13</i>	<i>b14</i>	<i>b15</i>	<i>b16</i>	<i>b17</i>	<i>b18</i>	<i>b19</i>	<i>b20</i>	<i>b21</i>	<i>b22</i>	<i>b23</i>	<i>b24</i>	<i>b25</i>	<i>b26</i>	<i>b27</i>	<i>b28</i>	<i>b29</i>	<i>b30</i>	<i>b31</i>	<i>b32</i>	<i>b33</i>	<i>b34</i>	<i>b35</i>	<i>b36</i>	<i>b37</i>	<i>b38</i>	<i>b38</i>	<i>b39</i>
1	1	1	0	0	1	1	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0

# Memória

Instruções em representação binária

Instrução 1

Instrução 2

b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19	b20	b21	b22	b23	b24	b25	b26	b27	b28	b29	b30	b31	b32	b33	b34	b35	b36	b37	b38	b38	b39
1	1	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0

20 bits

20 bits

# Memória

## Instruções em representação binária

Instrução 1

Instrução 2

*Opcode*

*Operando*

*Opcode*

*Operando*

b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19	b20	b21	b22	b23	b24	b25	b26	b27	b28	b29	b30	b31	b32	b33	b34	b35	b36	b37	b38	b38	b39
1	1	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0

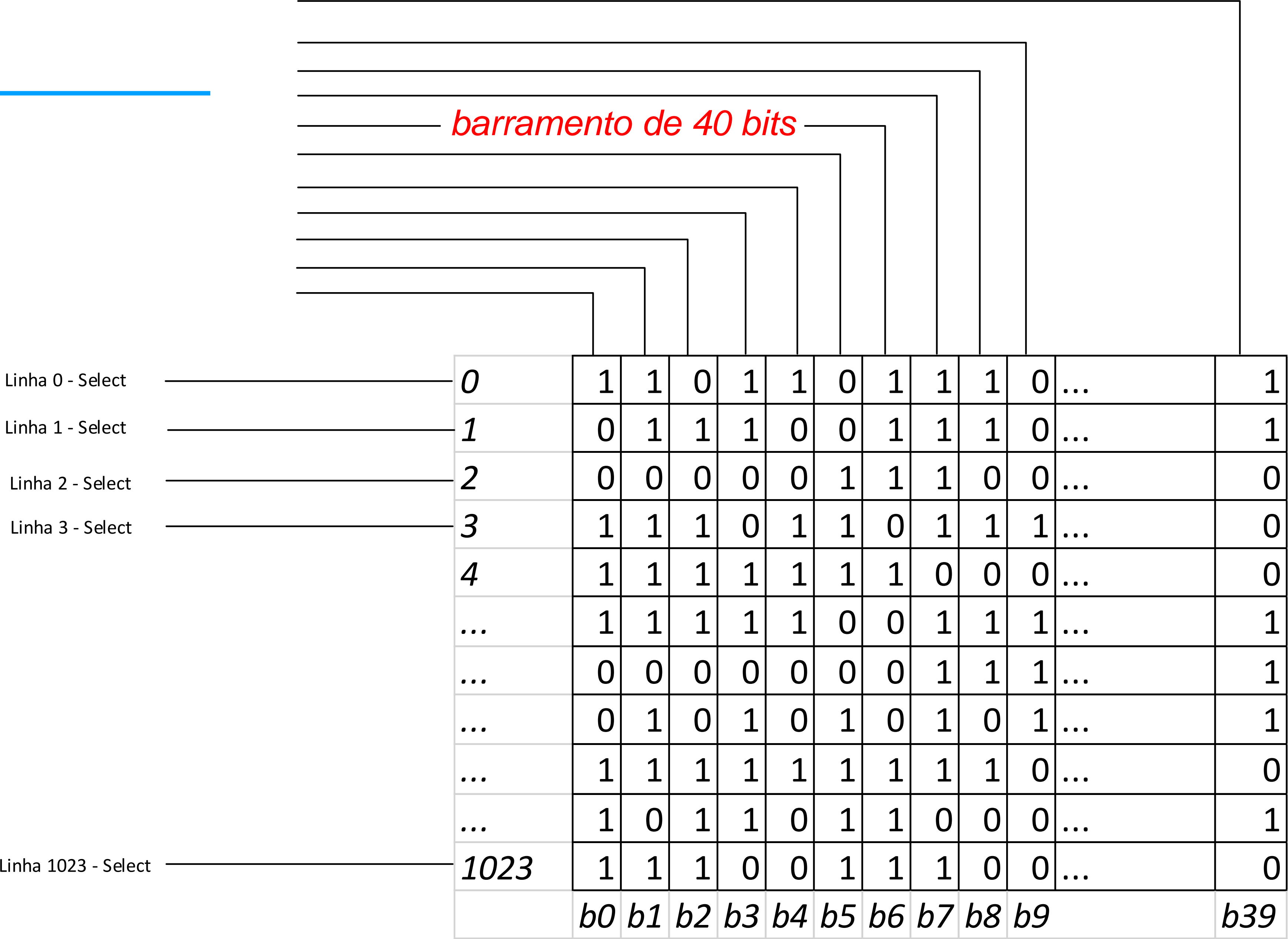
8 bits

12 bits

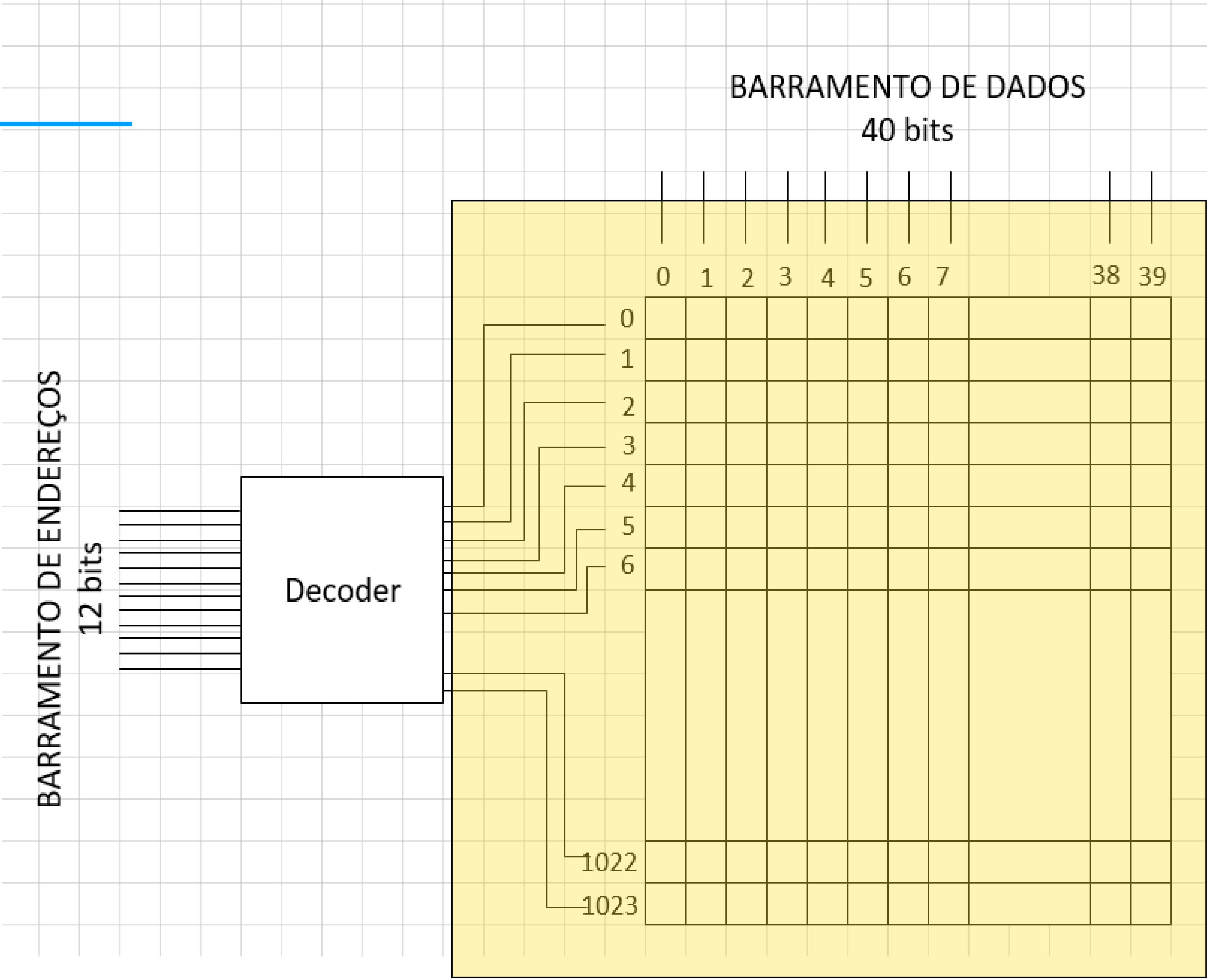
8 bits

12 bits

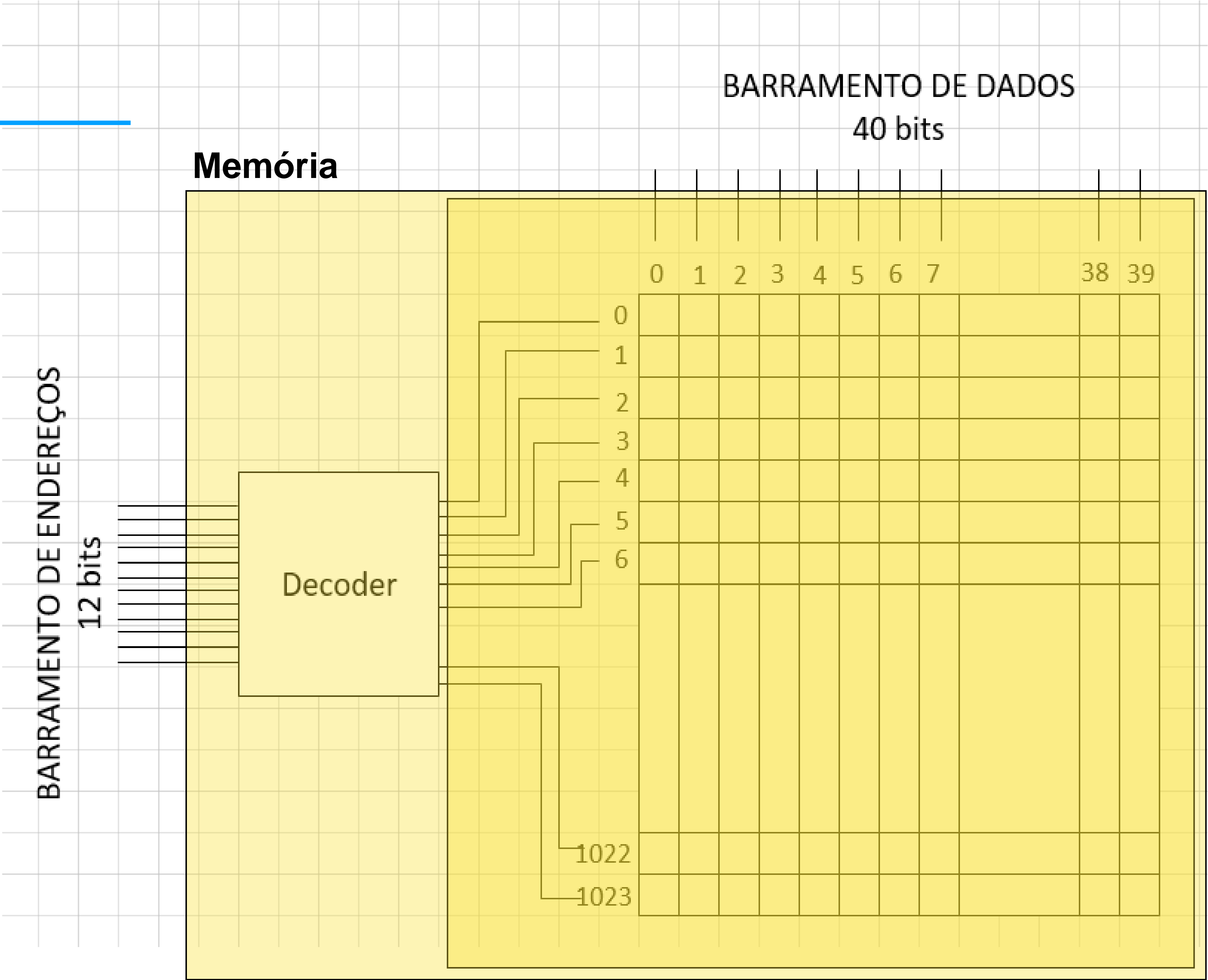
# IAS (von Neuman)



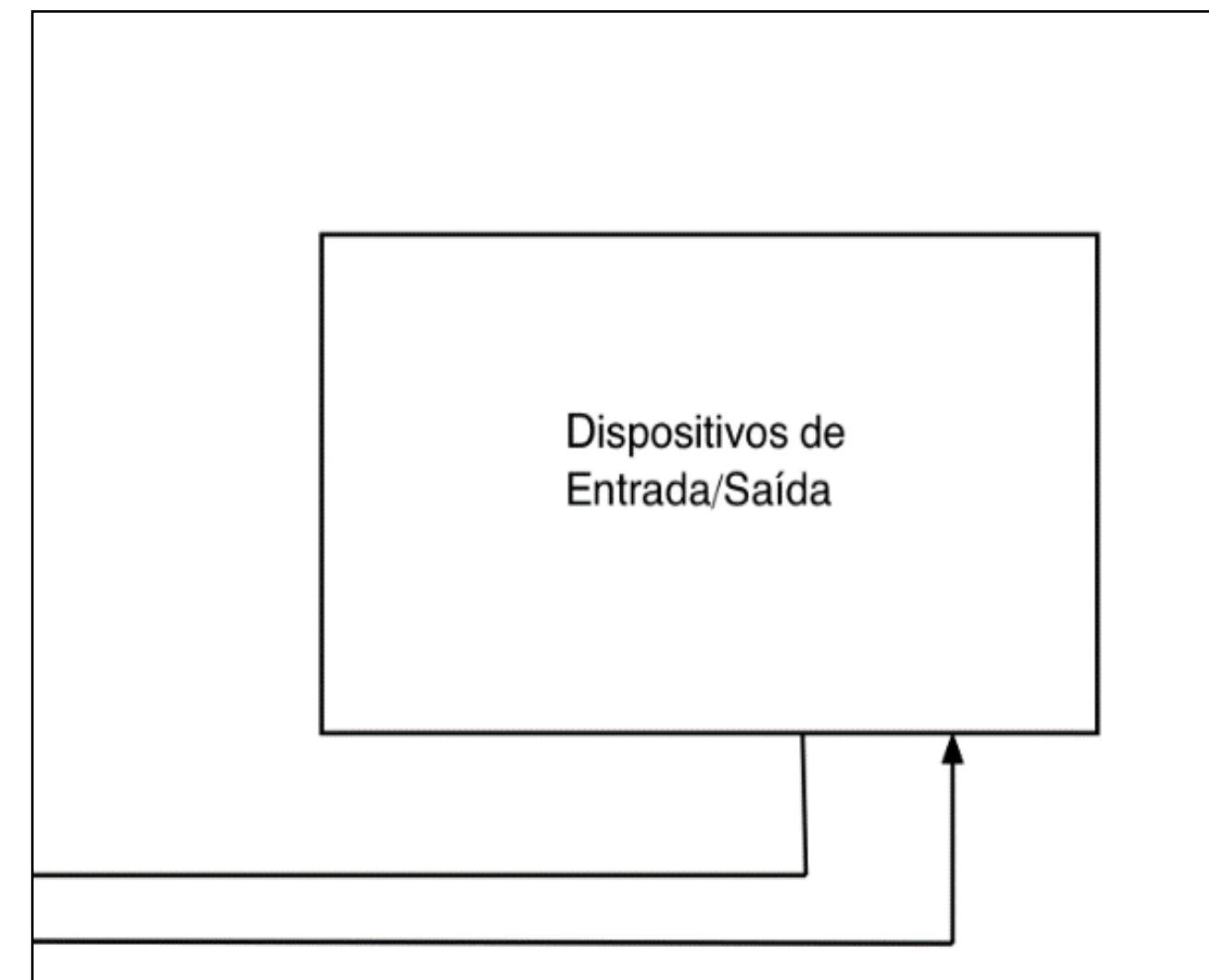
# IAS (von Neuman)



# IAS (von Neumman)



# Dispositivos de E/S

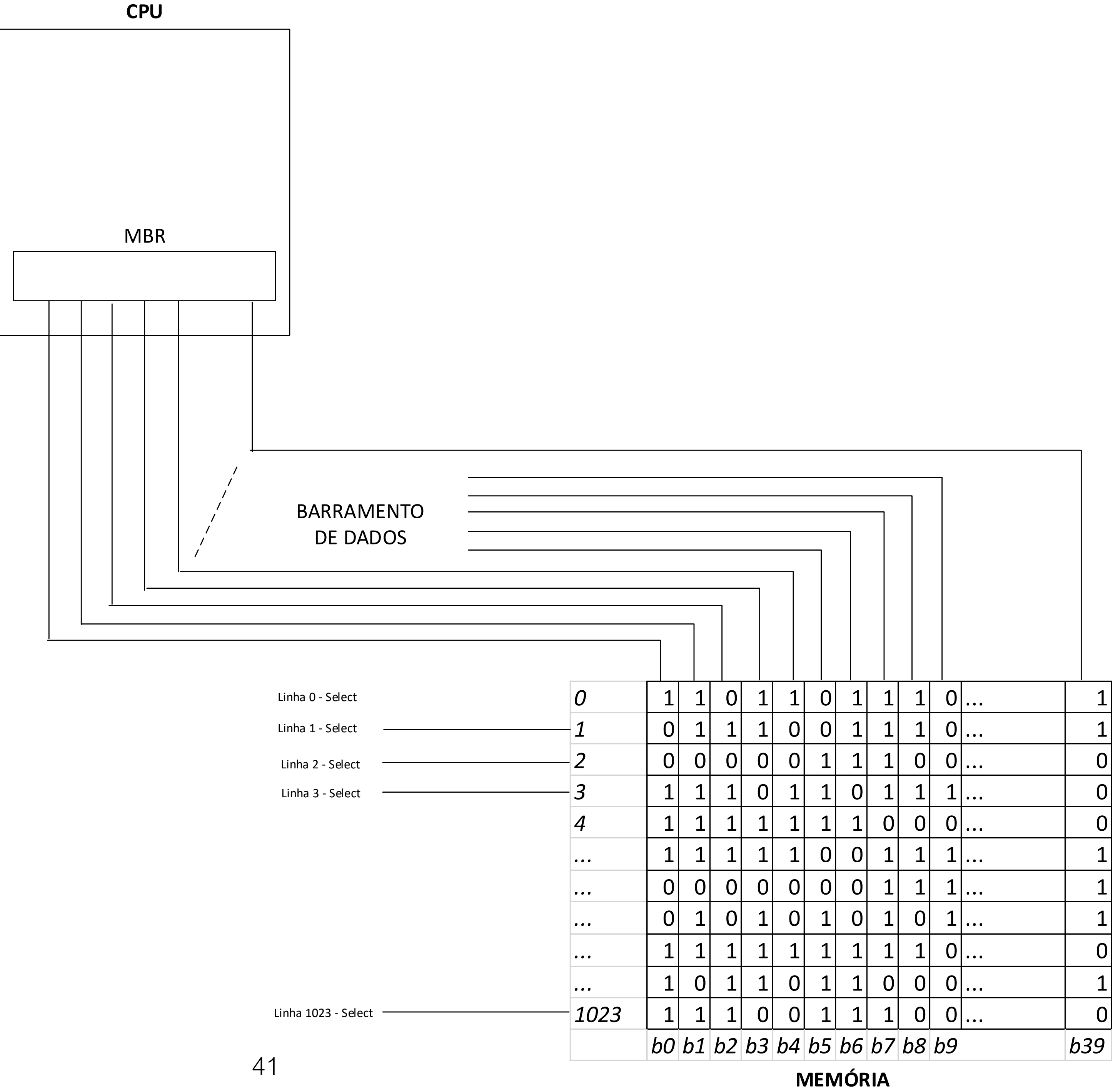


Como são usados 12 bits para identificar os endereços das palavras armazenadas na memória, considerando que  $2^{12}=4096$  combinações binárias, ainda permanecem combinações de endereço ( $4096 - 1024$ ) que podem ser usadas para escrever/ler dados de dispositivos de entrada e saída.

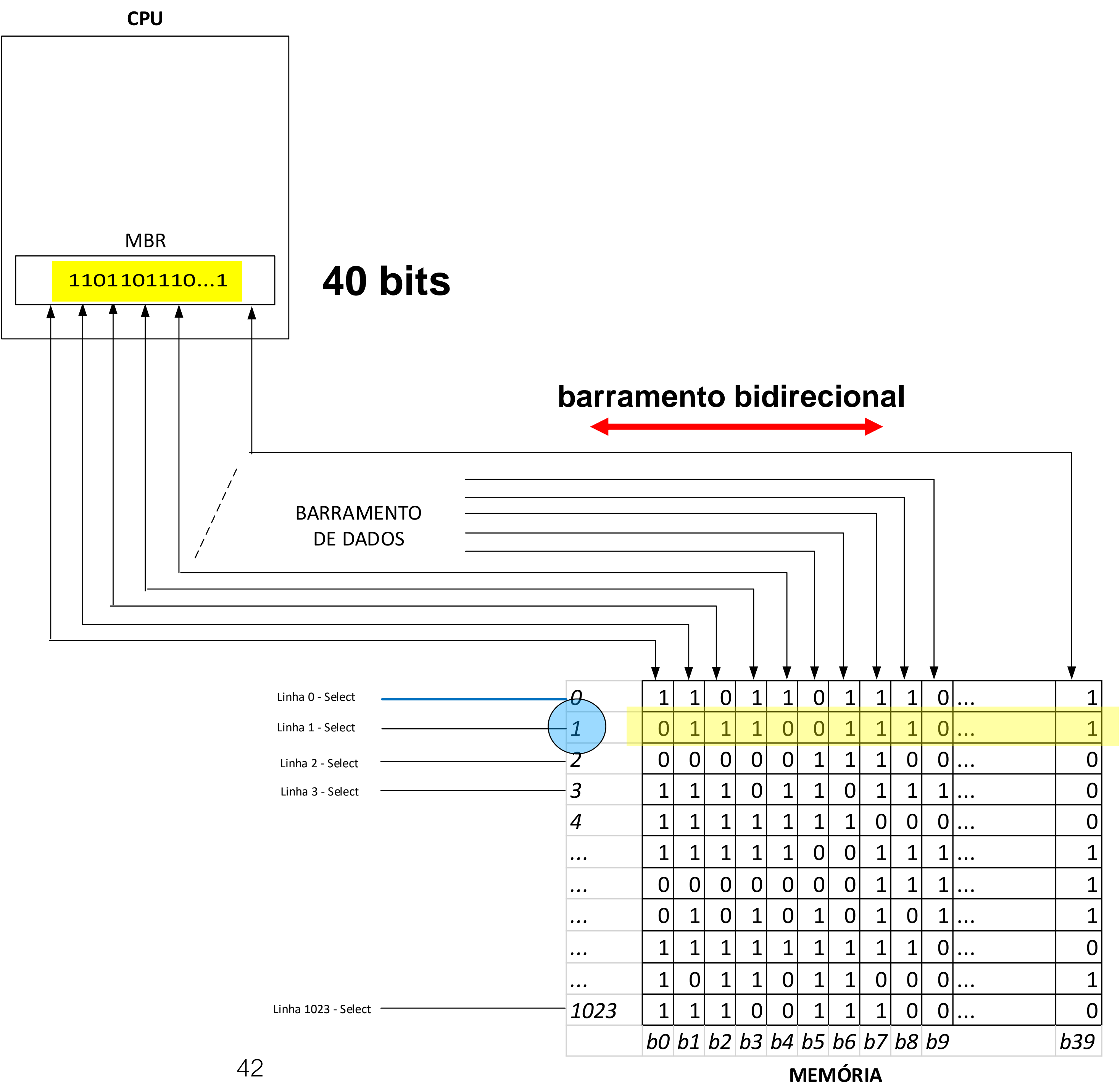
# Ligações



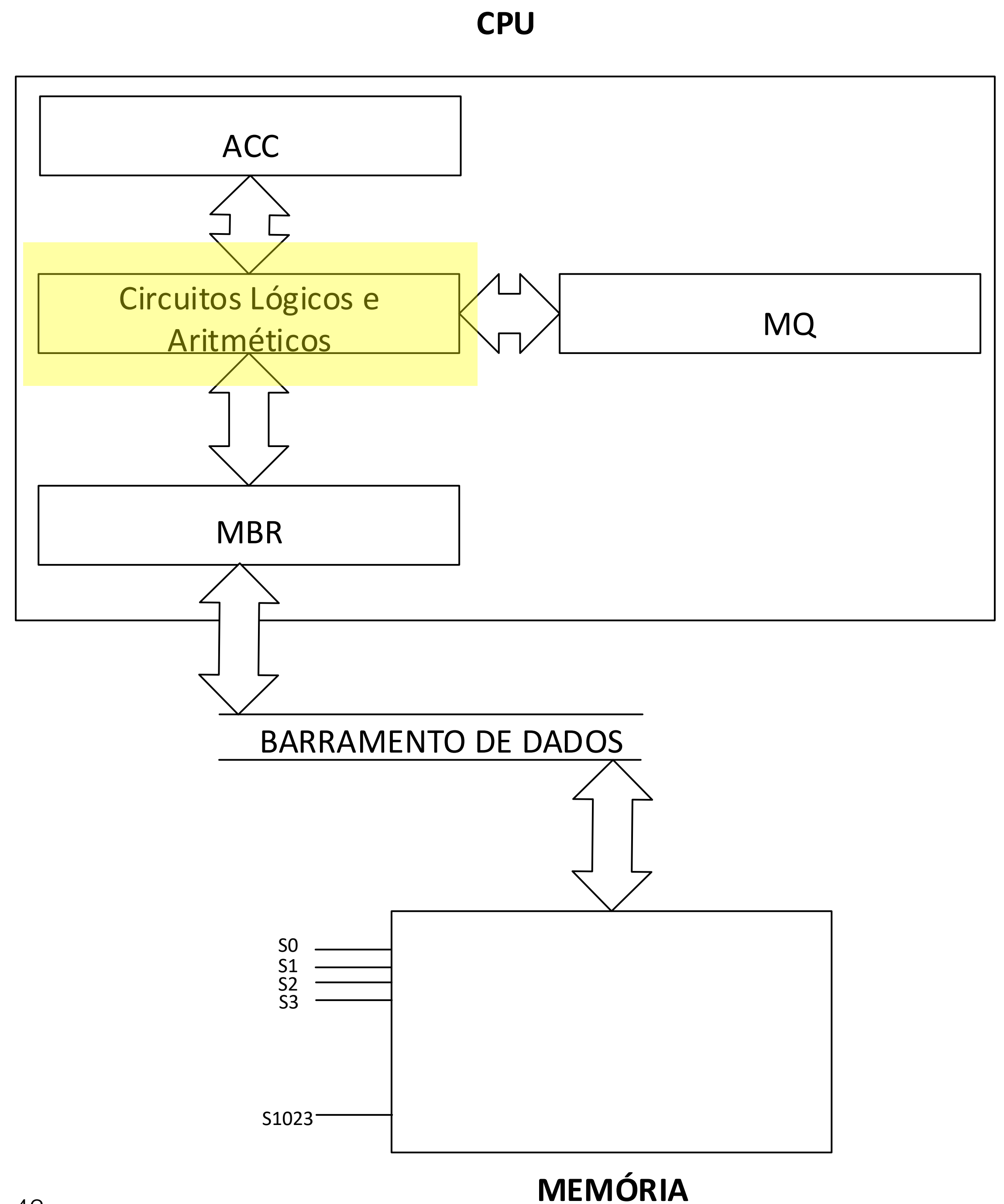
# MBR



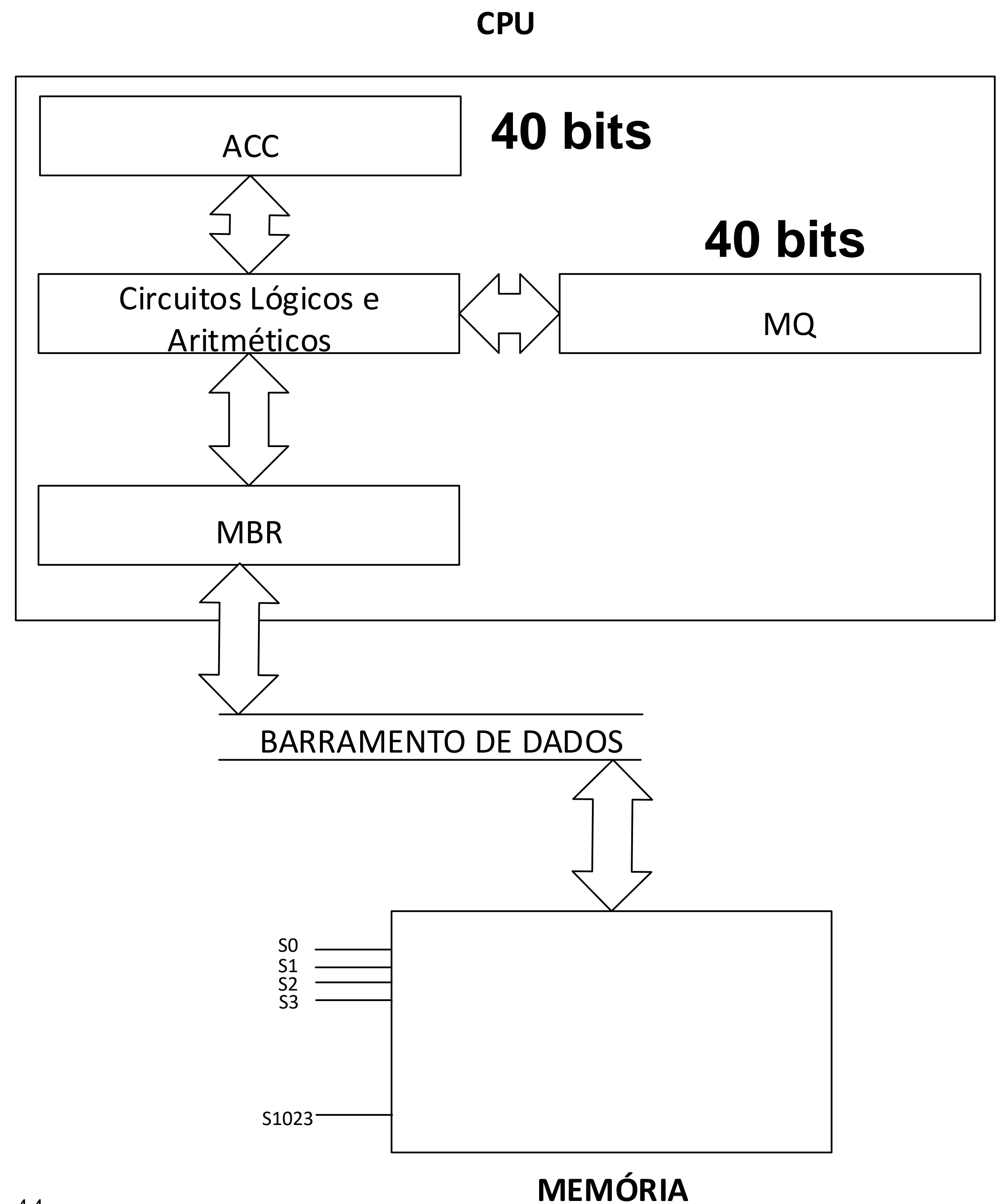
# MBR



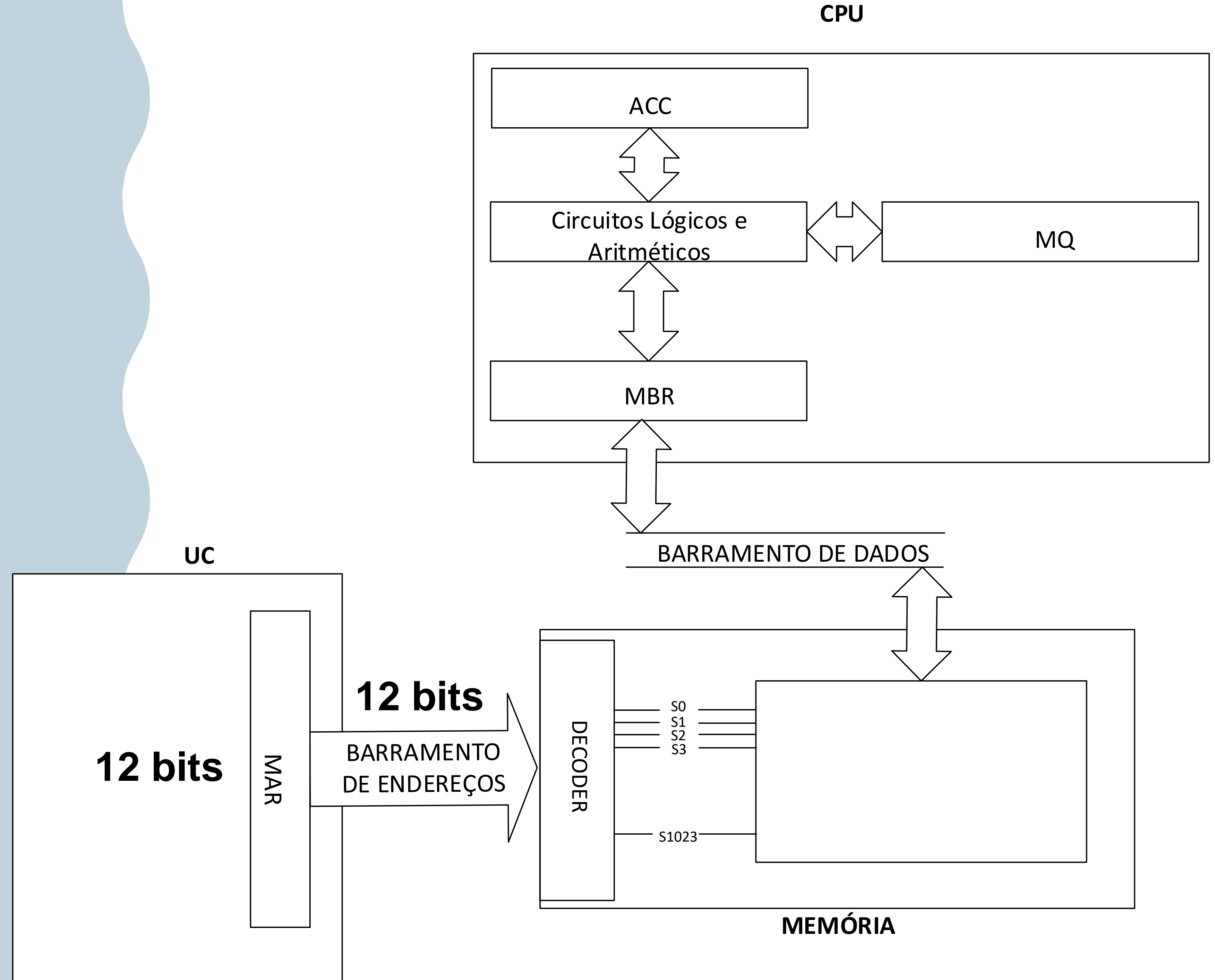
# Unidade Lógica e Aritmética



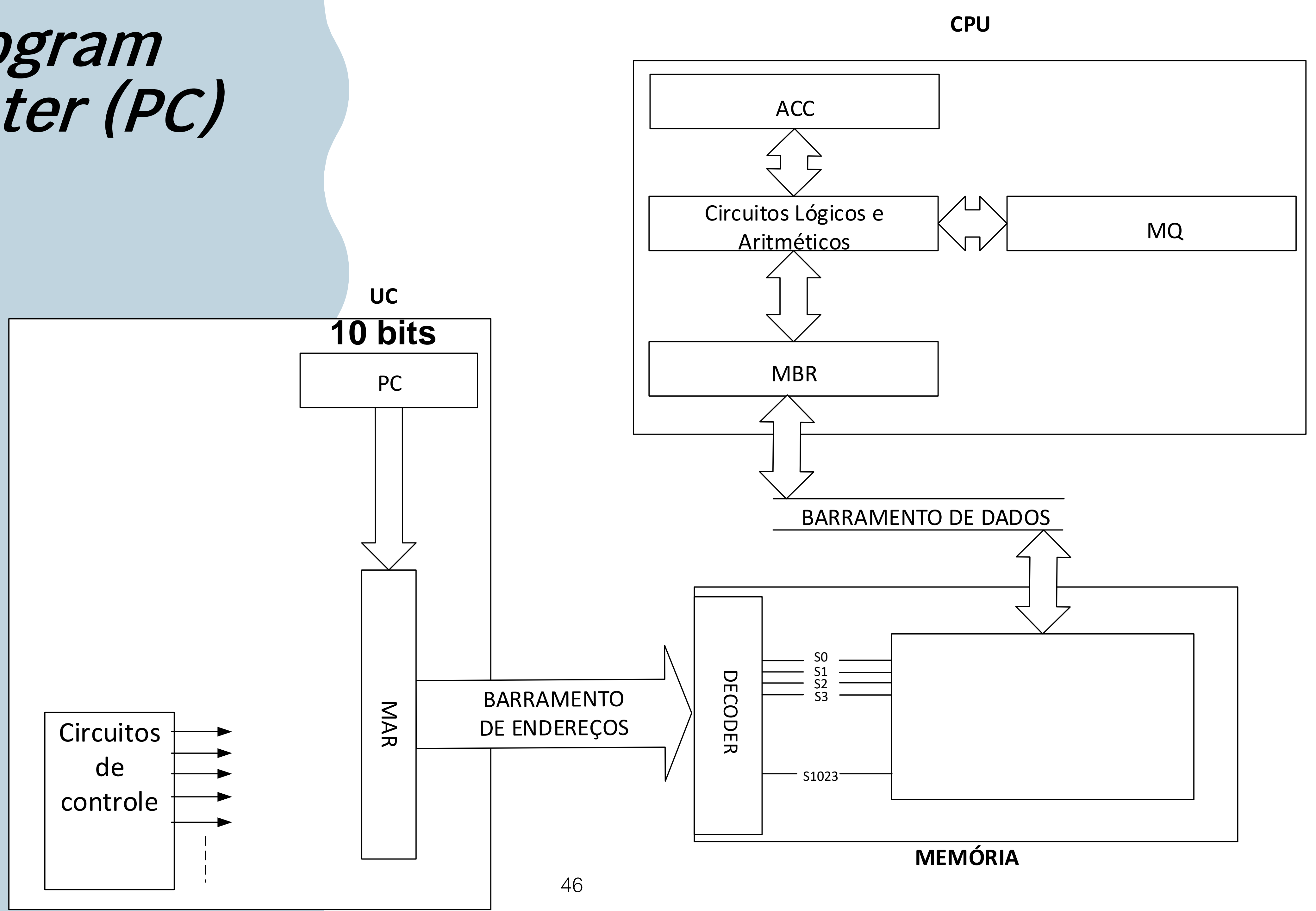
# Acc e MQ



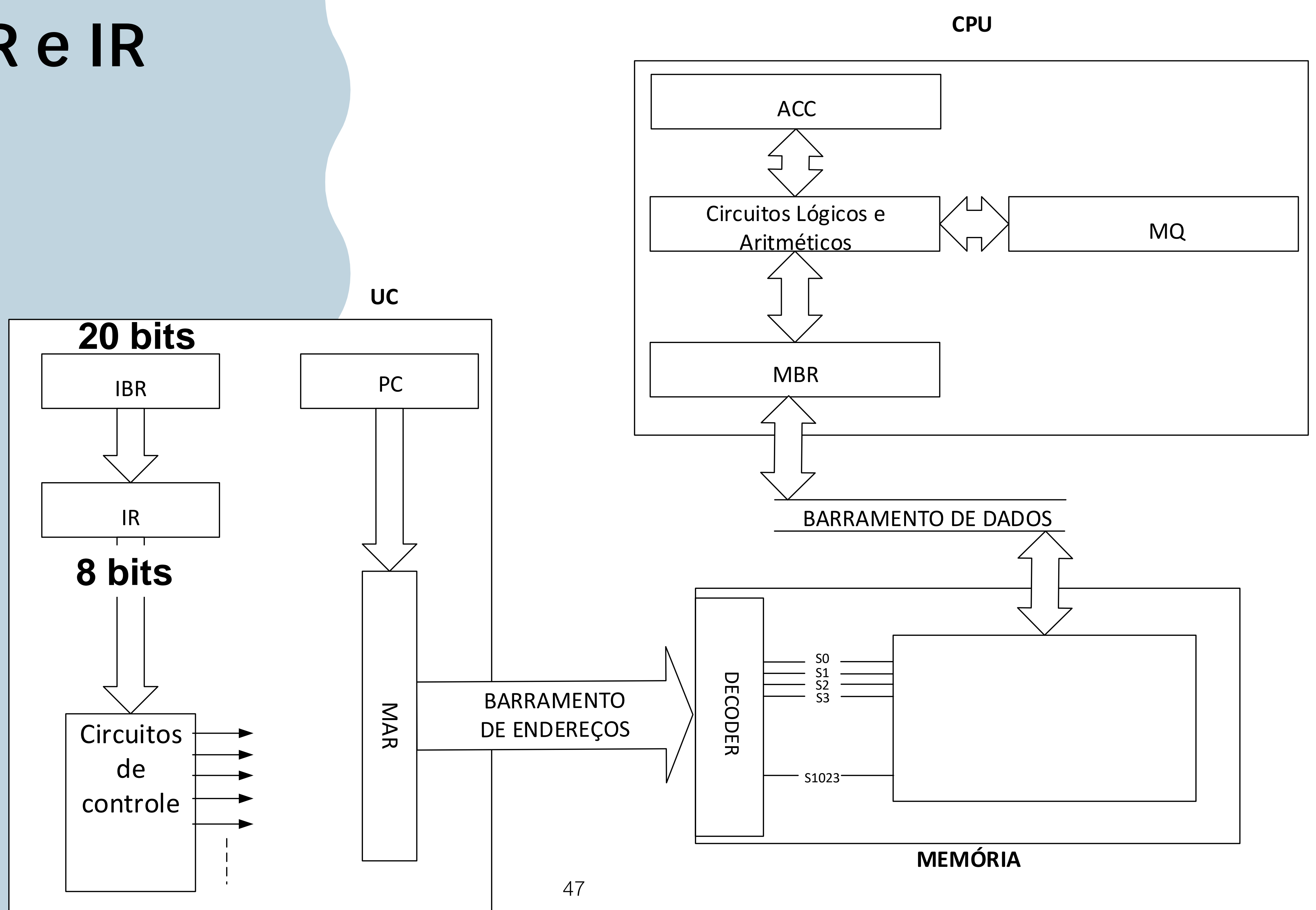
# Acc e MQ



# *Program Counter (PC)*



# IBR e IR

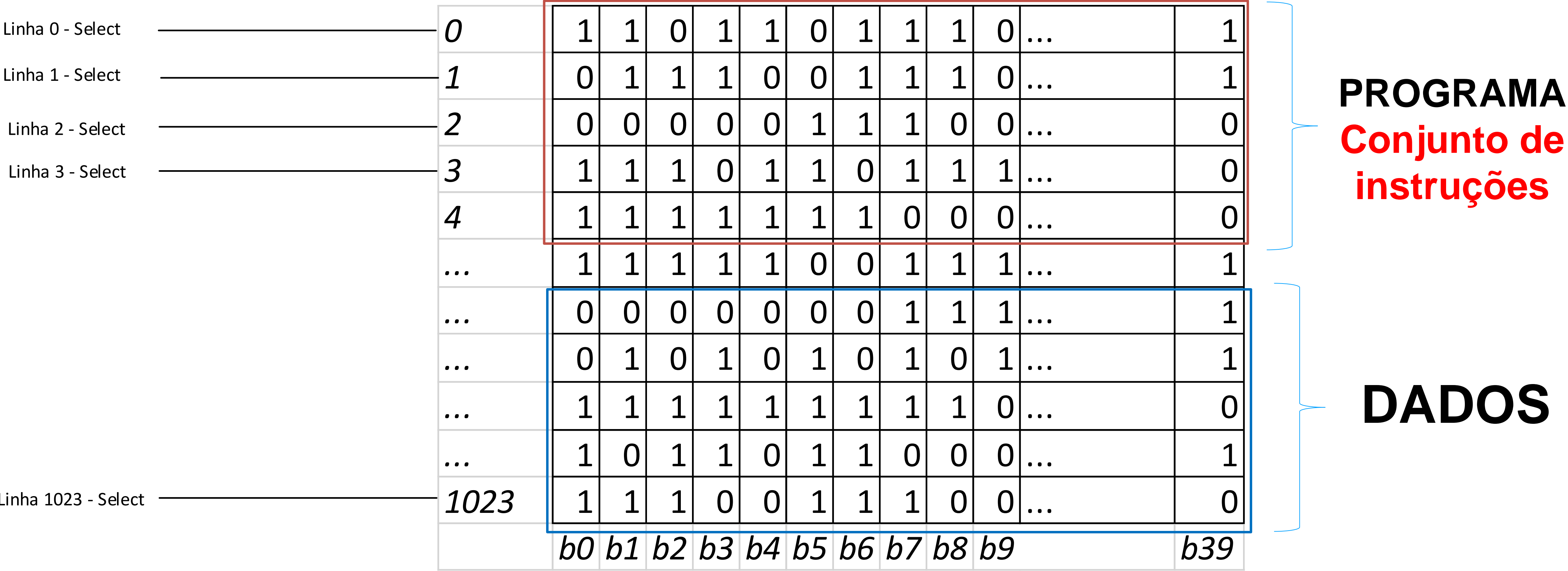


Execução do programa  
máquina de von Neumann

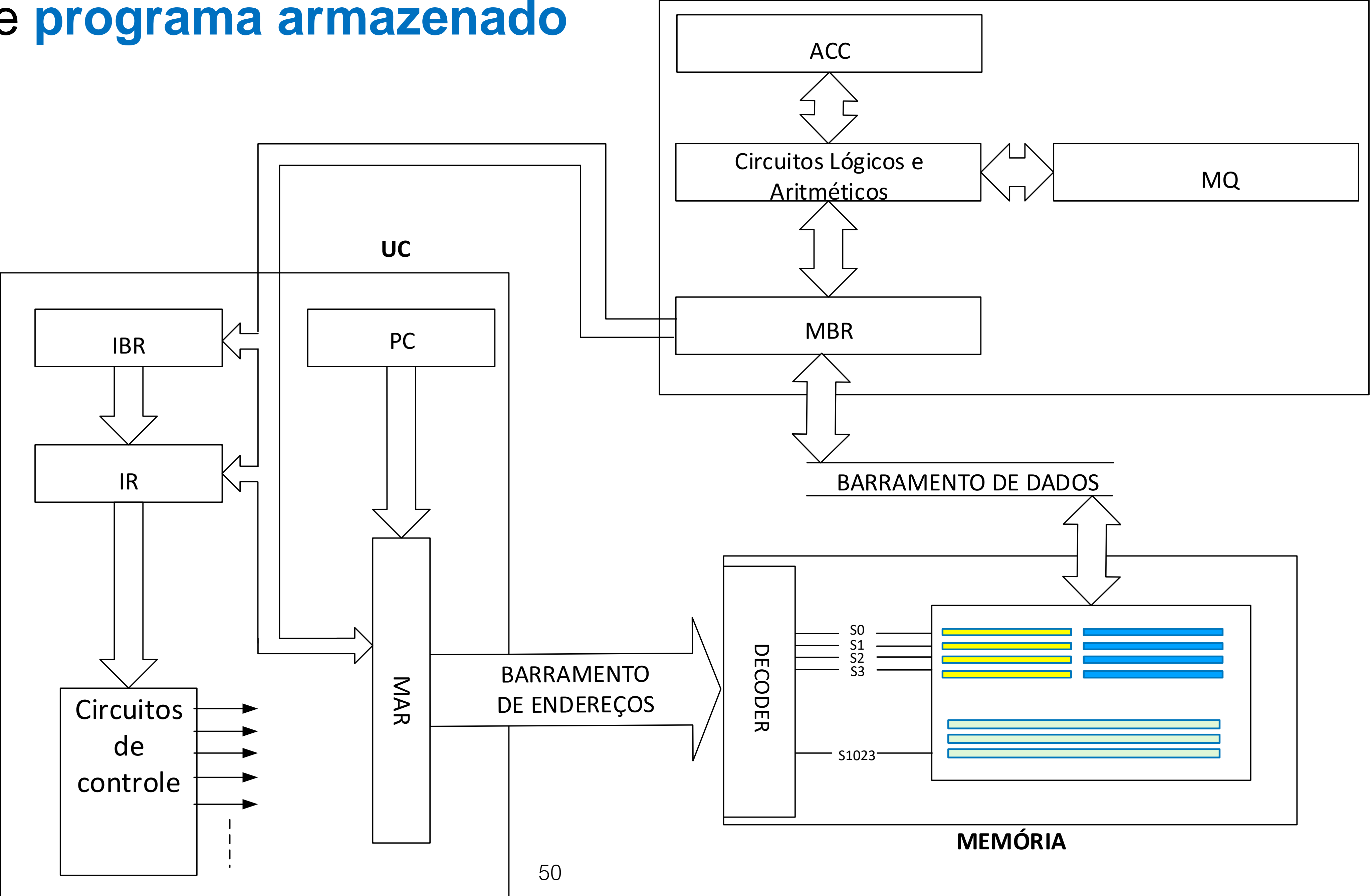
Conceito de programa  
armazenado



# Conceito de **programa armazenado**



# Conceito de **programa armazenado**



# Execução do programa

- O programa consiste na execução das instruções armazenadas em memória
- As instruções são normalmente armazenadas em posições de memória adjacentes e executadas sequencialmente
- As etapas (suboperações) de execução de cada operação variam de acordo com cada instrução do programa
- As instruções são **executadas sincronizadamente**. O sincronismo é dado pelos circuitos de controle.
- Após a execução de cada instrução os elementos apresentam um valor. Os valores de cada elemento definem o **estado** da máquina.

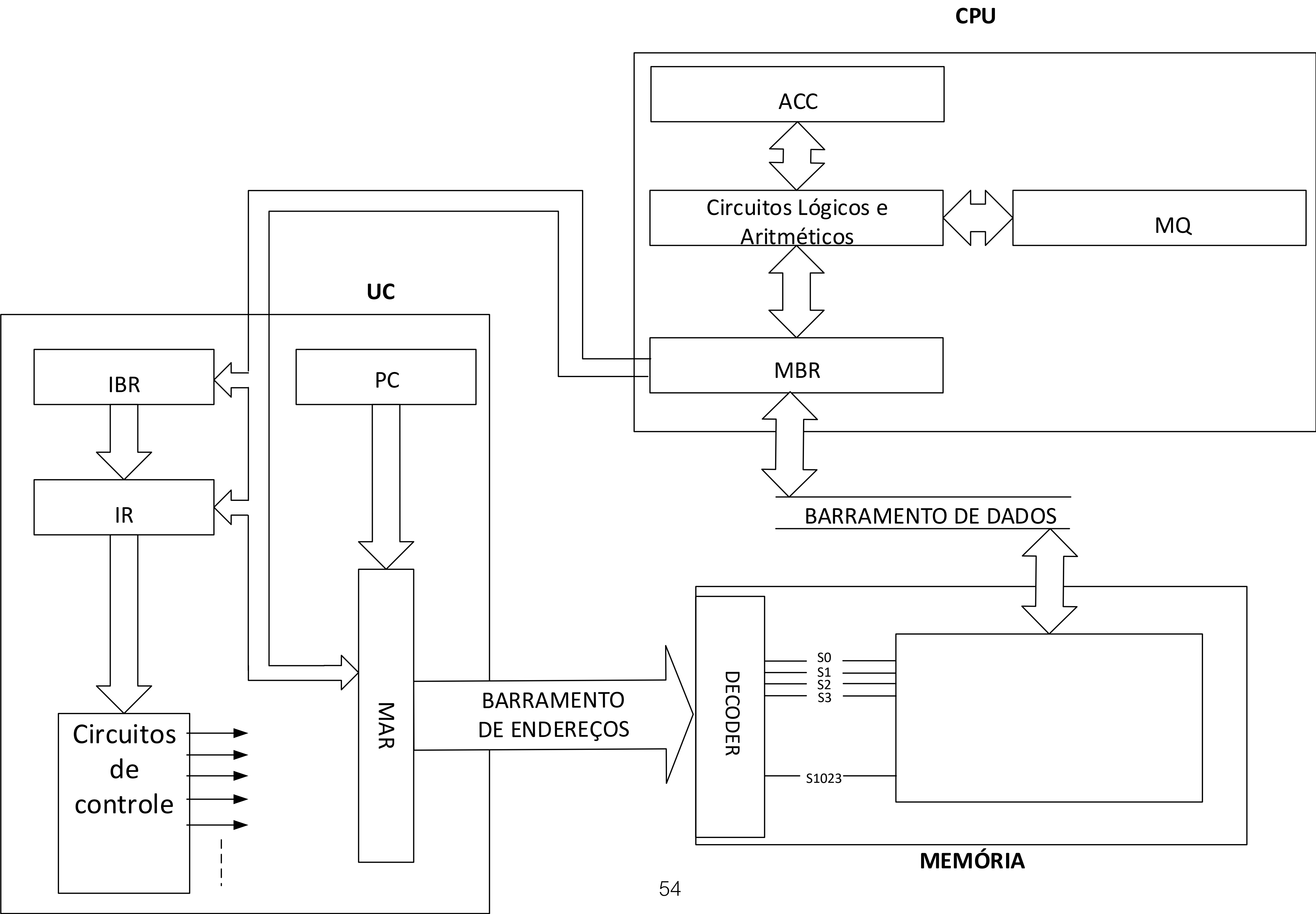
# Ciclo de instrução

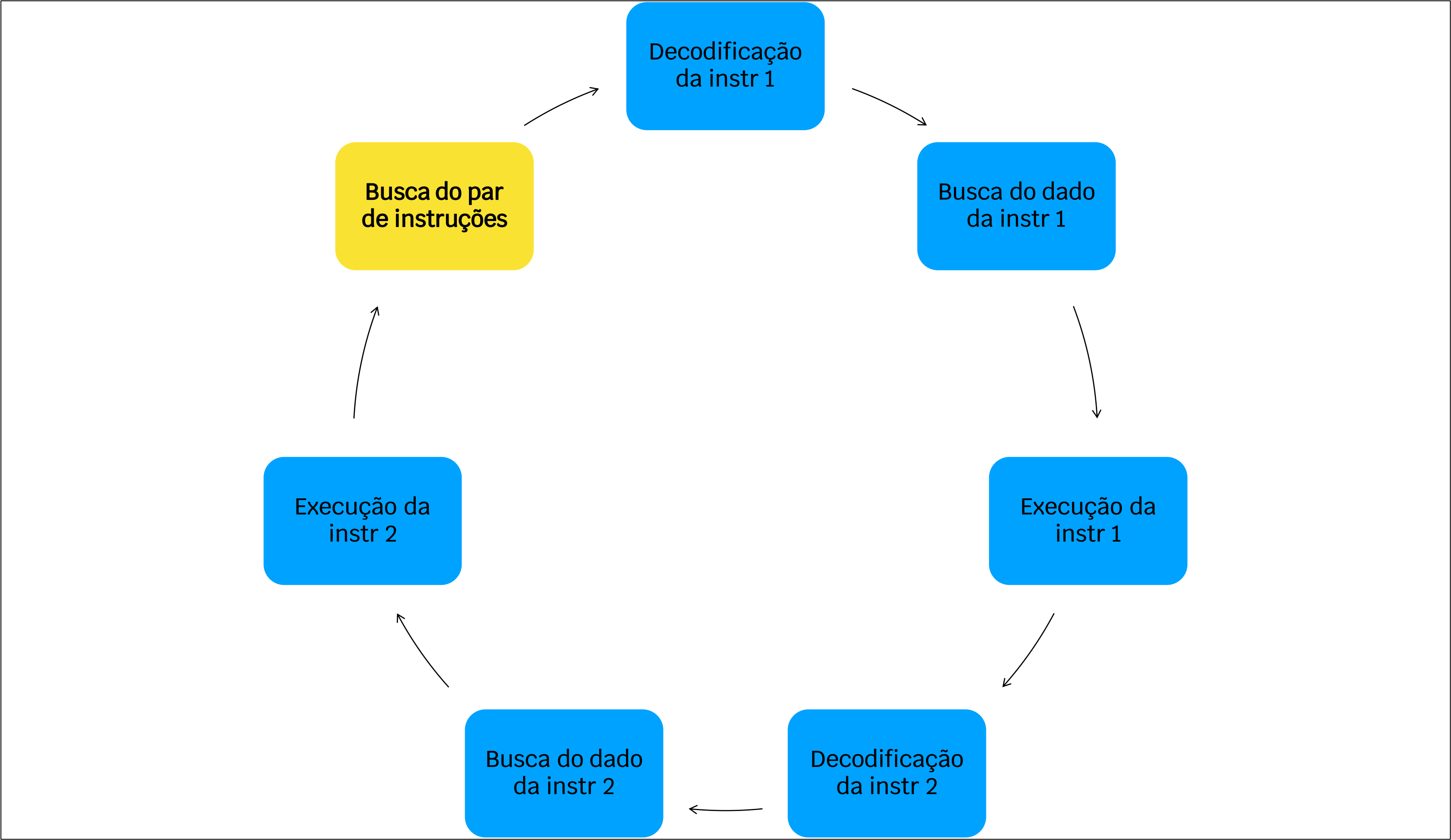


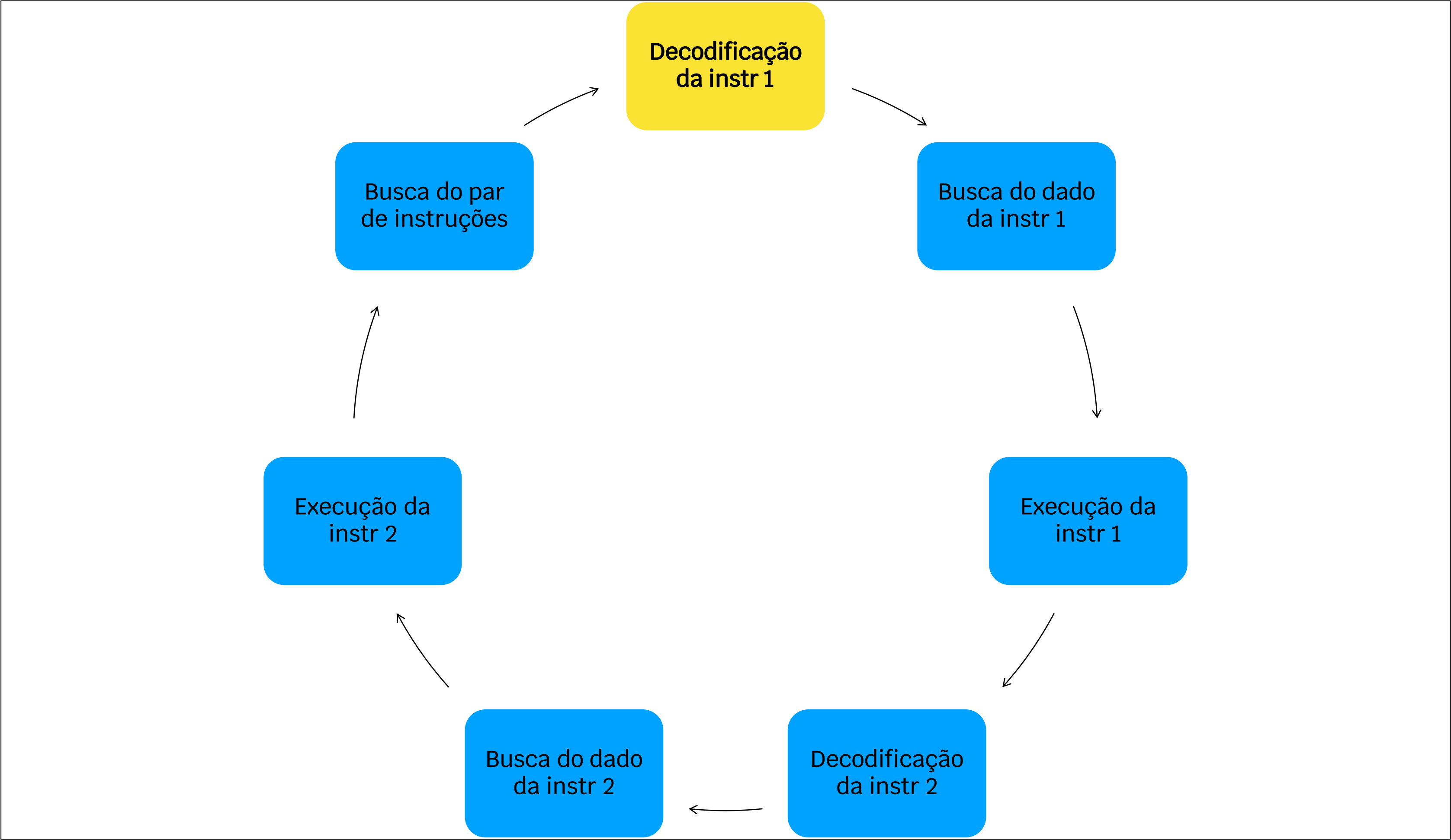
Busca da instrução

Execução da  
instrução

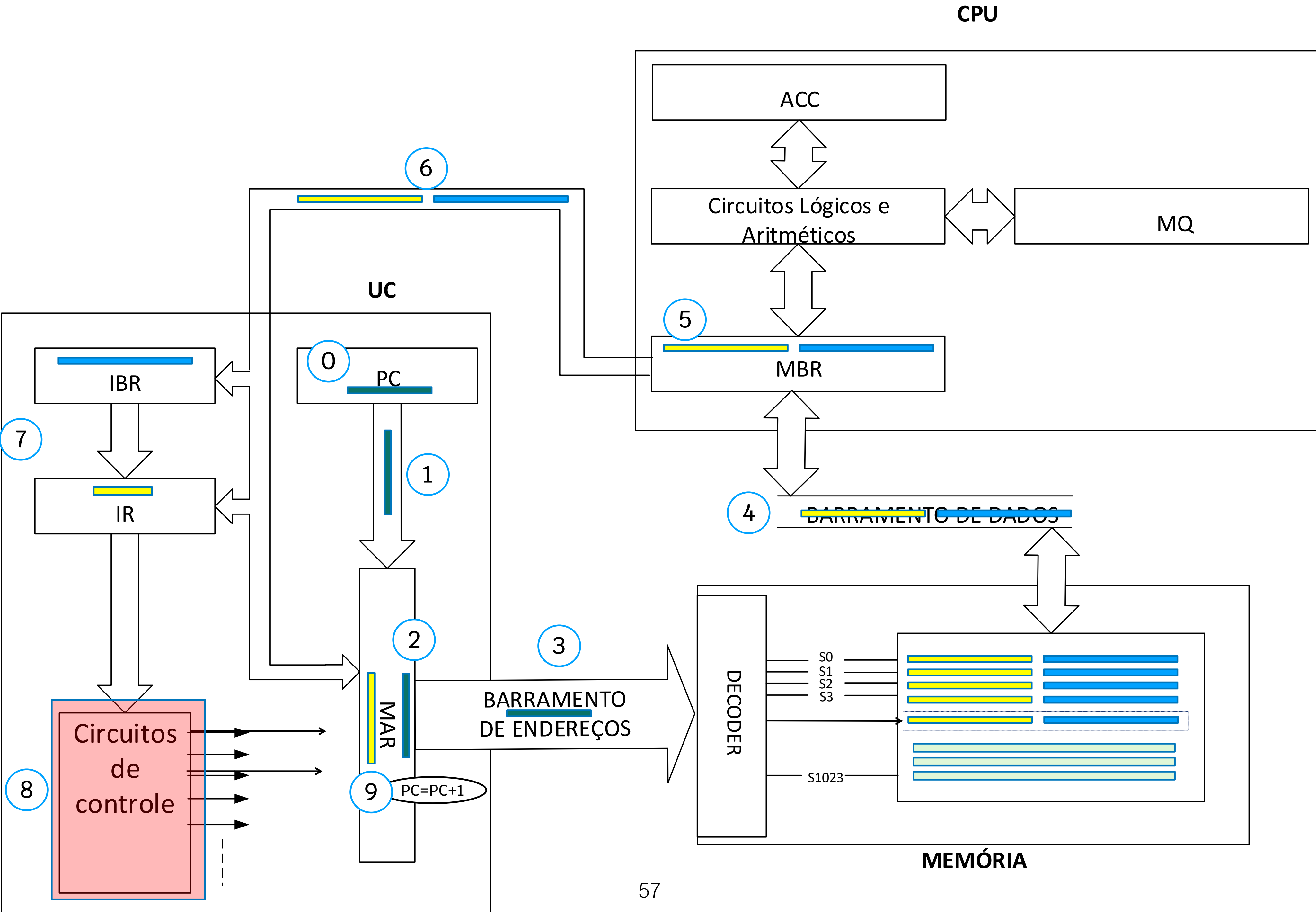


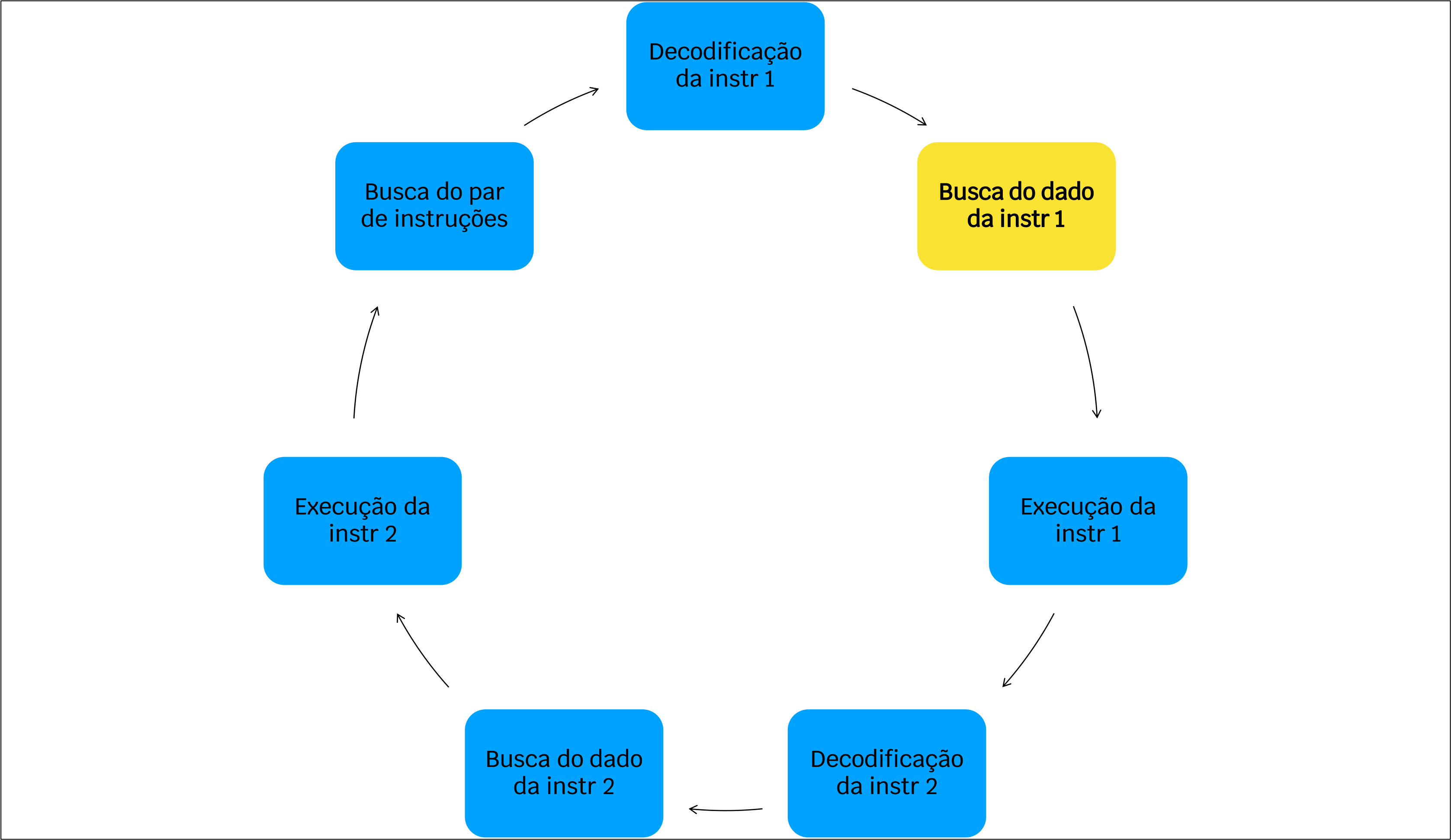


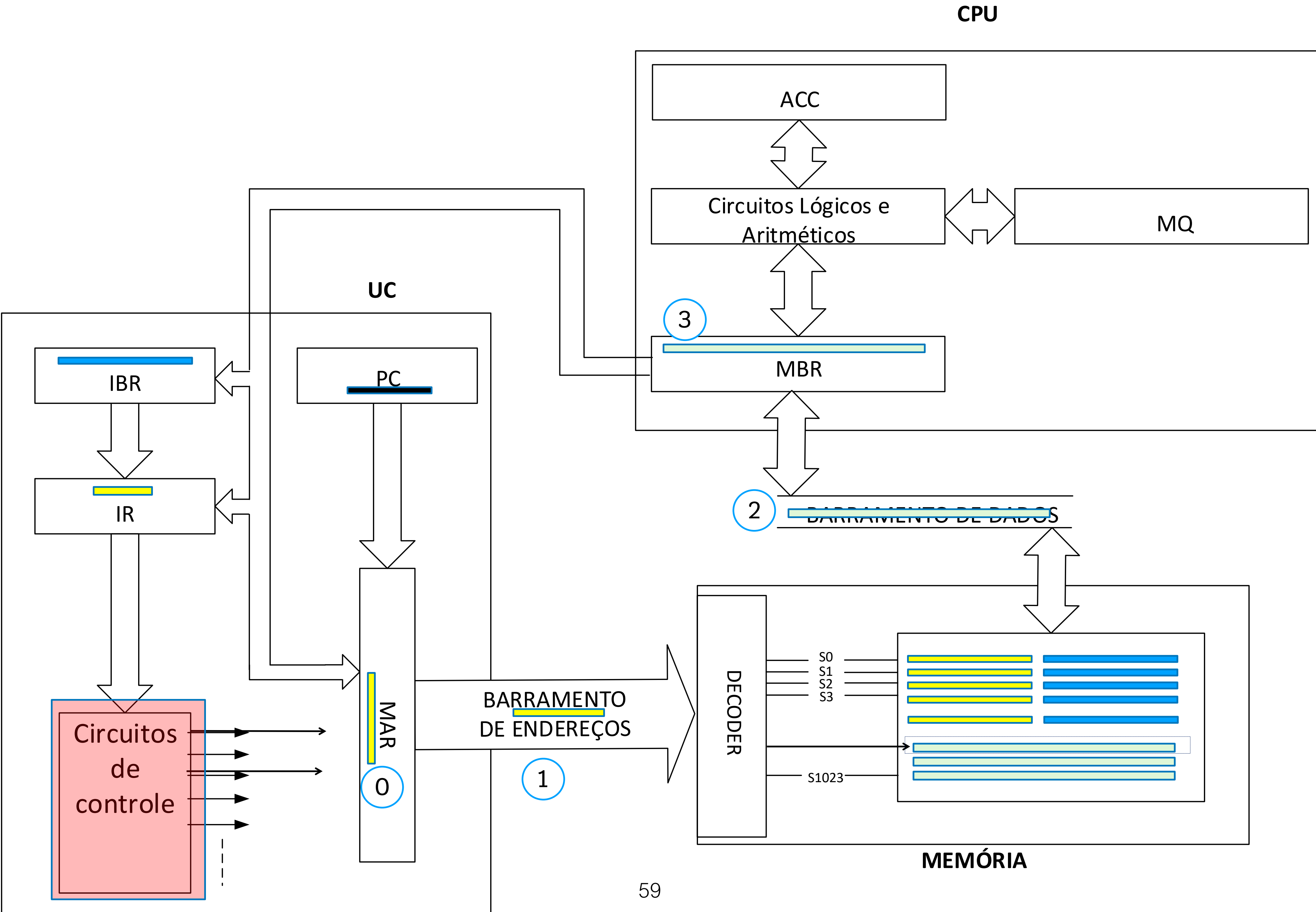


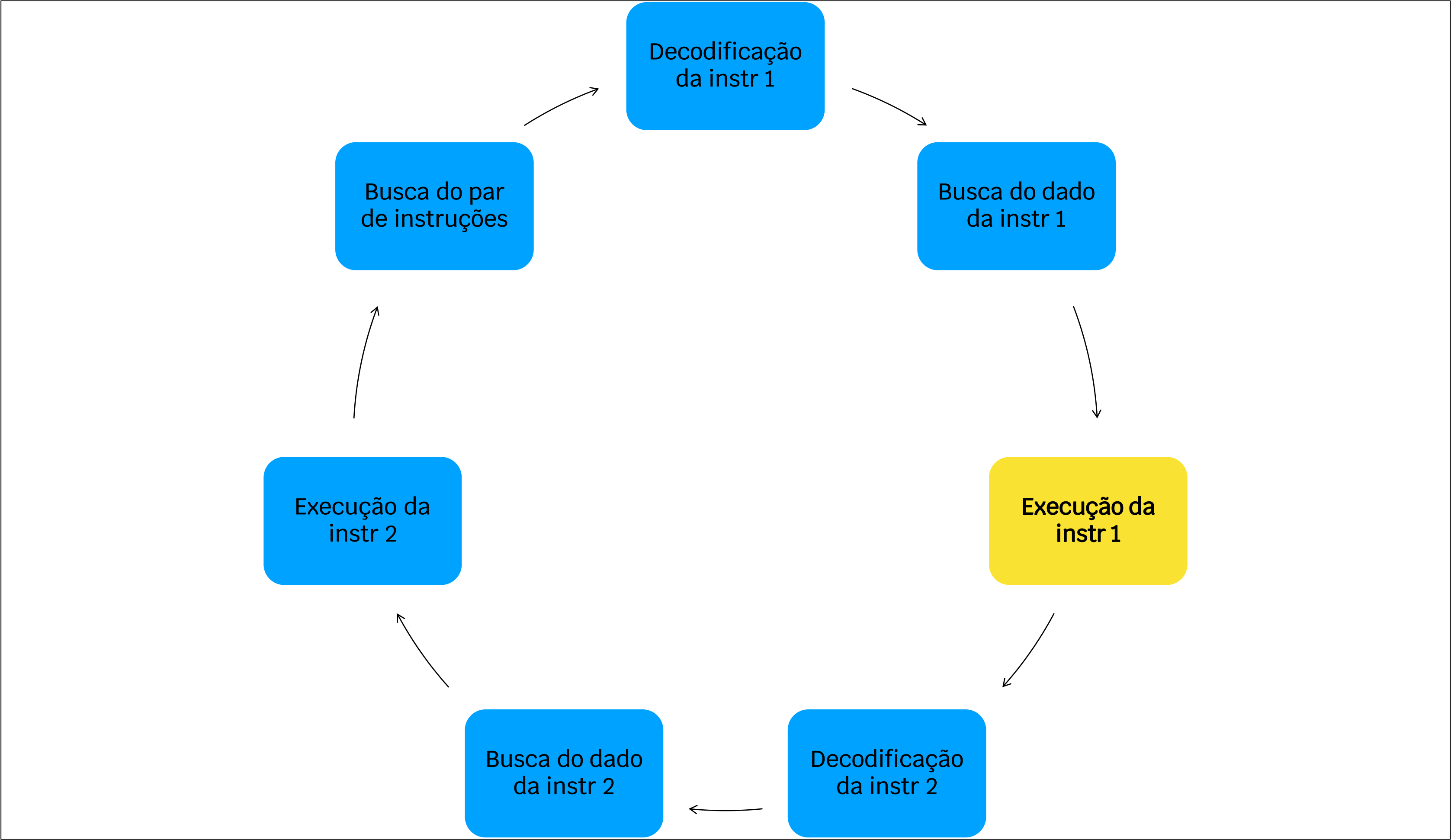


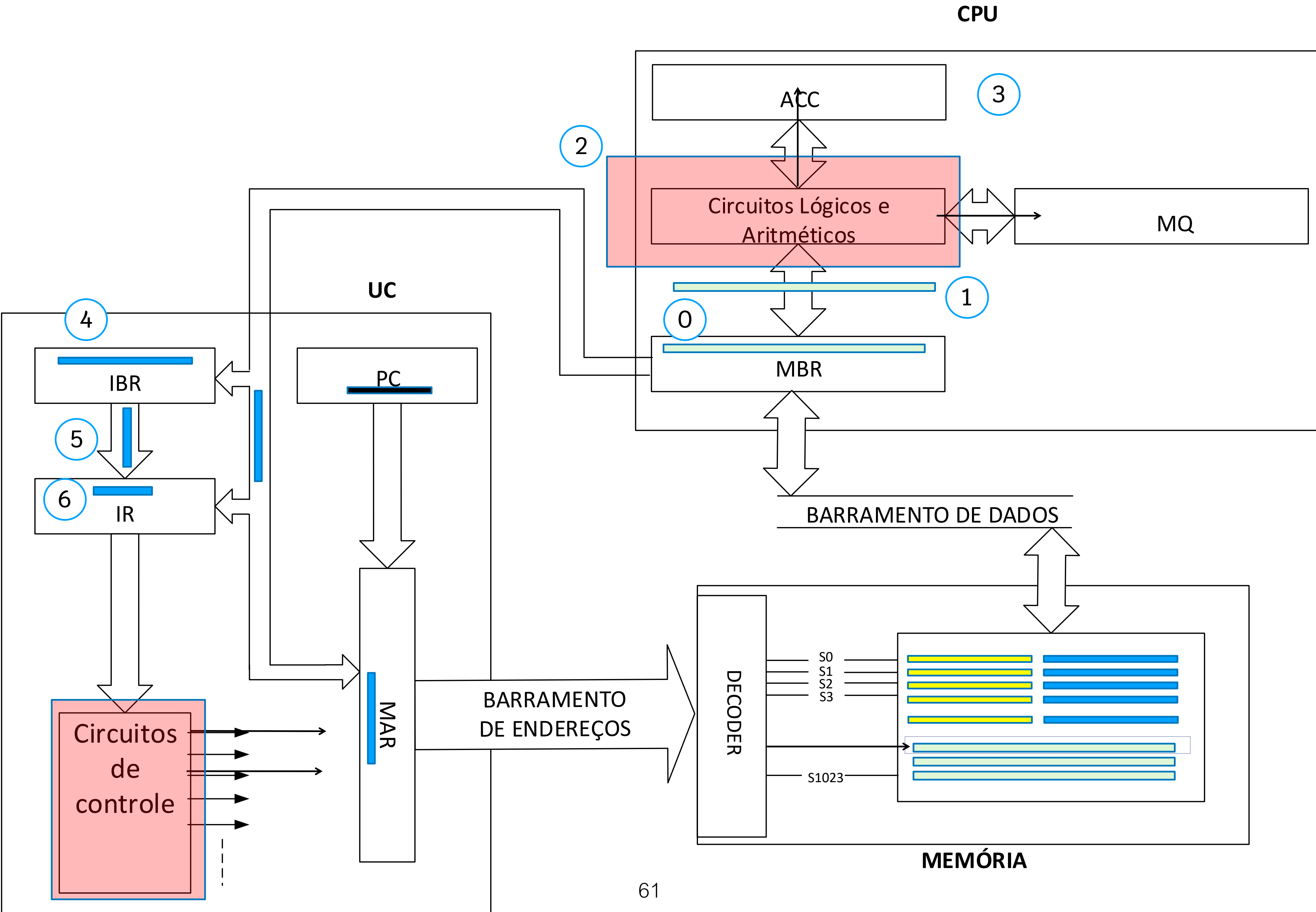


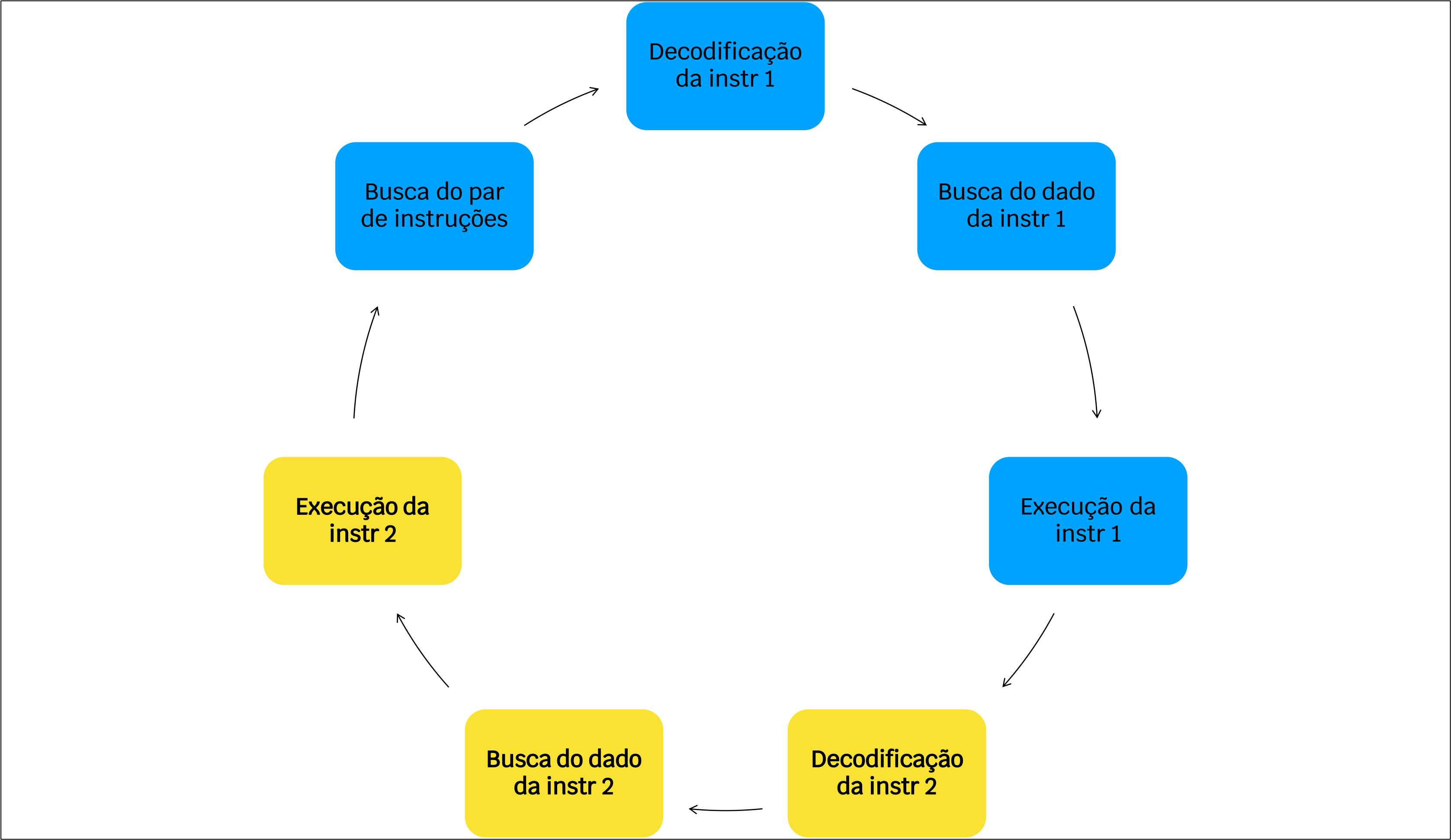




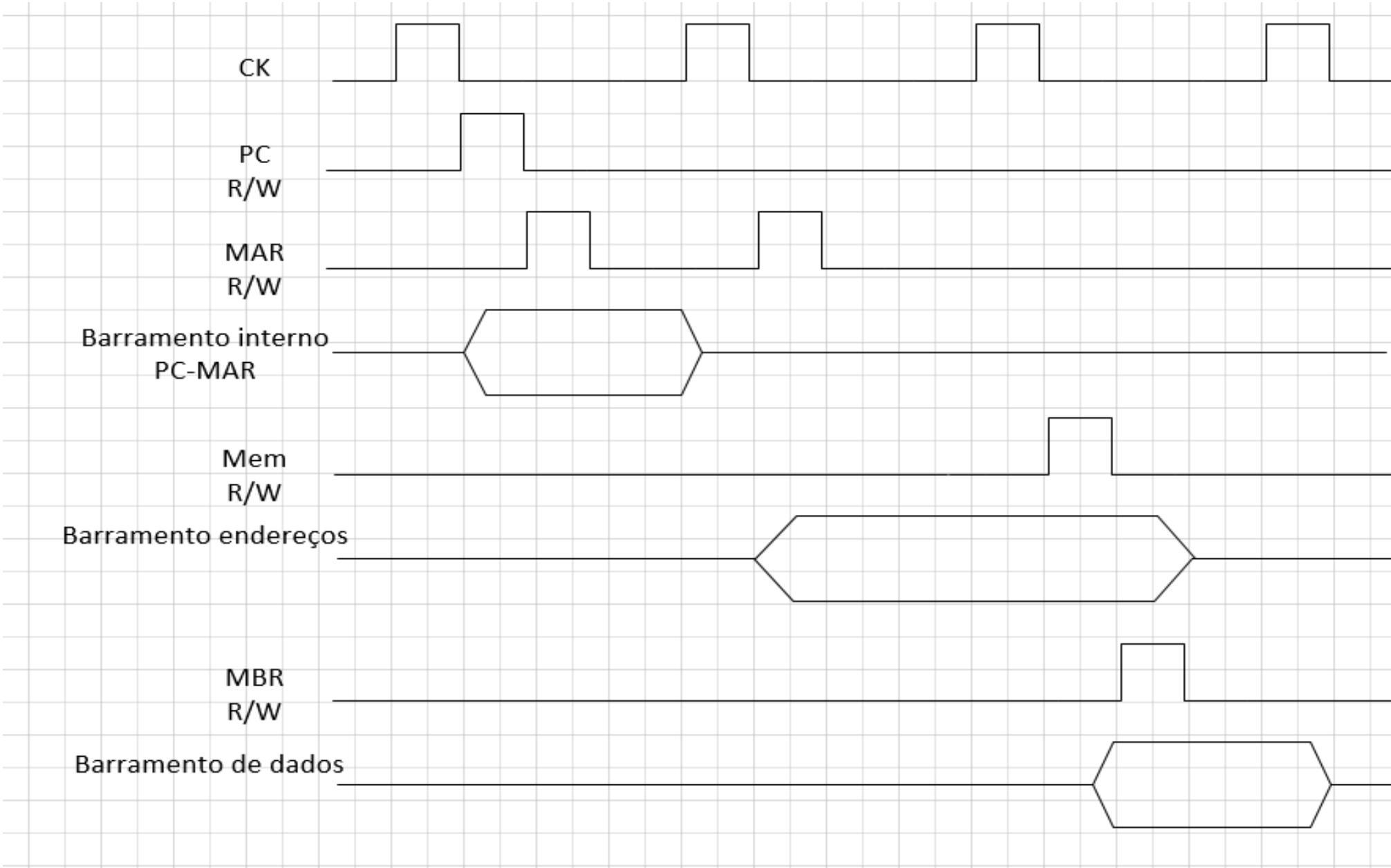








# Tabela de variáveis de estado



MBR	ACC	MQ	IBR	IR	PC	MAR



# Instruções do processador IAS

- **Transferência de dados:** os dados são transferidos entre a memória (M) e os registradores da UCP ou entre registradores.
- **Desvio incondicional:** normalmente a UC executa as instruções em sequência na memória, que pode ser alterada pelo uso desse desvio independentemente de qualquer condição, quando a instrução é executada.
- **Desvio condicional:** o desvio é executado dependendo de um teste de condição estabelecido pela instrução.

# Instruções do processador IAS

- **Lógicas e aritméticas**, como adição, subtração, multiplicação e divisão binárias.
- **Alteração de endereços**: instruções para calcular endereços para inseri-los em instruções armazenadas na memória, propiciando flexibilidade de endereçamento dos programas.

# Representação das instruções

Código de operação	Representação simbólica	Descrição
00001001	LOAD MQ,M(X)	Transfere o conteúdo da posição de memória X para MQ

OPCODE: BINÁRIO, NÍVEL ISA

OPERAÇÃO REALIZADA

OPCODE: SIMBÓLICA, NÍVEL ASSEMBLY



# Representação das instruções

Tipo de Instrução	Código de operação	Representação simbólica	Descrição
Transferência de dados	00001010	LOAD MQ	Transfere o conteúdo do registrador MQ para o acumulador AC
	00001001	LOAD MQ,M(X)	Transfere o conteúdo da posição de memória X para MQ
	00100001	STOR M(X)	Transfere o conteúdo do acumulador para a posição de memória X
	00000001	LOAD M(X)	Transfere M(X) para o acumulador
	00000010	LOAD - M(X)	Transfere - M(X) para o acumulador
	00000011	LOAD  M(X)	Transfere o valor absoluto de M(X) para o acumulador
	00000100	LOAD -  M(X)	Transfere -  M(X)  para o acumulador



# Representação das instruções

Tipo de Instrução	Código de operação	Representação simbólica	Descrição
Desvio incondicional	00001101	JUMP M(X,0:19)	A próxima instrução a ser executada é buscada na metade esquerda de M(X)
	00001110	JUMP M(X,20:39)	A próxima instrução a ser executada é buscada na metade direita de M(X)
Desvio condicional	00001111	JUMP+(X,0:19)	Se o número no acumulador é um valor não-negativo, a próxima instrução a ser executada é buscada na metade esquerda de M(X)
	00010000	JUMP+M(X,20:39)	Se o número no acumulador é um valor não-negativo, a próxima instrução a ser executada é buscada na metade direita de M(X)



# Representação das instruções

Tipo de Instrução	Código de operação	Representação simbólica	Descrição
Aritmética	00000101	ADD M(X)	Soma M(X) a AC; armazena o resultado em AC
	00000111	ADD  M(X)	Soma  M(X)  a AC; armazena o resultado em AC
	00000110	SUB M(X)	Subtrai M(X) de AC; armazena o resultado em AC
	00001000	SUB  M(X)	Subtrai  M(X)  de AC; armazena o resto em AC
	00001011	MUL M(X)	Multiplica M(X) por MQ; armazena os bits mais significativos do resultado em AC, armazena os bits menos significativos em MQ.
	00001100	DIV M(X)	Divide AC por M(X); armazena o quociente em MQ e o resto em AC.
	00010100	LSH	Multiplica o acumulador por 2 (isto é, desloca os bits uma posição para a esquerda).
	00010101	RSH	Divide o acumulador por 2 (isto é, desloca os bits uma posição para a direita).



# Representação das instruções

Tipo de Instrução	Código de operação	Representação simbólica	Descrição
Alteração de endereço	00010010	STOR M(X,8:19)	Substitui o campo de endereço à esquerda de M(X) pelos 12 bits mais à direita de AC.
	00010011	STOR M(X,28:39)	Substitui o campo de endereço à direita de M(X) pelos 12 bits mais à direita de AC.
	00001000	SUB  M(X)	Subtrai  M(X)  de AC; armazena o resto em AC
	00001011	MUL M(X)	Multiplifica M(X) por MQ; armazena os bits mais significativos do resultado em AC, armazena os bits menos significativos em MQ.
	00001100	DIV M(X)	Divide AC por M(X); armazena o quociente em MQ e o resto em AC.
	00010100	LSH	Multiplifica o acumulador por 2 (isto é, desloca os bits uma posição para a esquerda).
	00010101	RSH	Divide o acumulador por 2 (isto é, desloca os bits uma posição para a direita).



# A palavra de instrução de programa...

b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	...	b39
0	0	0	0	1	0	0	1	0	1	1	...	0

OPCODE: BINÁRIO, NÍVEL ISA

Linha	b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	...	b39
0	0	0	0	0	1	0	0	1	0	1	1	...	0
1													
2													
3													
4													
5													
6													
...													
1023													

# Representações do programa...

Nível assembly

Endereço	OPCODE	OPERANDO
0	LOAD MQ, M(X)	DADO
	...	

ASSEMBLER

Nível ISA

Linha	b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	...	b39
0	0	0	0	0	1	0	0	1	0	1	1	...	0
1													
2													
3													
4													
5													
6													
...													
1023													

# Referências

[1] Tanenbaum, Andrew; Organização Estruturada de Computadores, seções 1.1 e 2.1

[2] Stallings, William; Arquitetura e Organização de Computadores; seções 2.1 e 3.1



IBMEC.BR

 /IBMEC

 IBMEC

 @IBMEC\_OFICIAL

 @IBMEC

 **ibmec**