

# EEL7030 - Microprocessadores

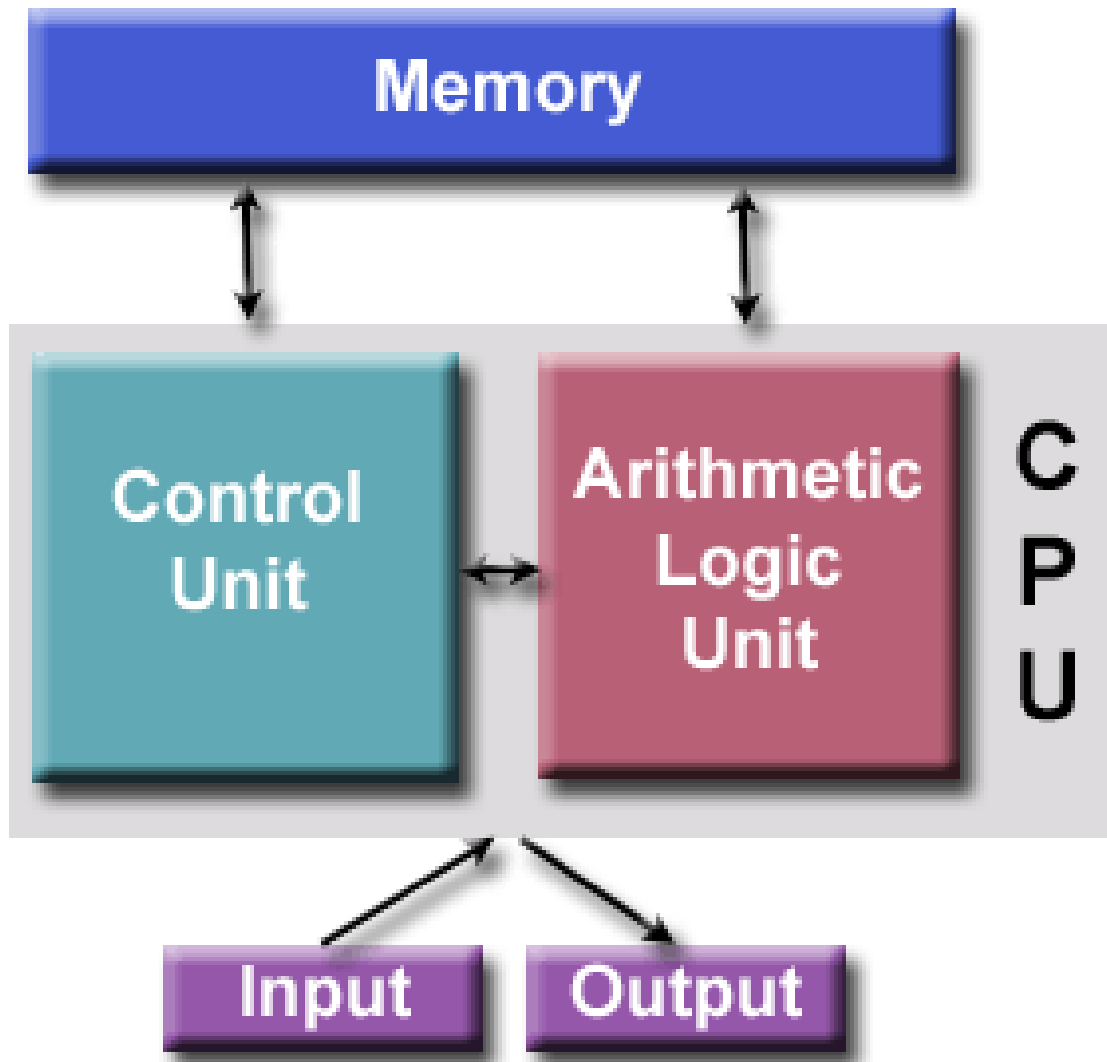


**LCS**

Laboratório de  
Comunicações  
e Sistemas  
Embarcados

**Prof. Raimes Moraes**  
**EEL - UFSC**

# Arquitetura Von Neuman de Computadores



- ★ **UCP** - Unidade central de processamento
- ★ **Memória:** Armazenamento de programas e dados
- ★ **I/O:** Transferência de dados entre computador e ambiente externo
- ★ **Barramentos:** estabelece comunicação entre UCP, memória e I/O

# Tipos de Memória

## Não Voláteis (dados permanentes)

❑ **MROM ou ROM:** *Mask Read Only Memory*: Memória com conteúdo definido em fábrica

❑ **PROM:** *Programmable Read Only Memory*: Gravada em campo uma única vez

❑ **EPROM:** *Erasable Programmable Read Only Memory*: regravável.

❑ **Flash:** Apaga-se e grava-se setores da memória

❑ **EEPROM:** *Electrically Erasable Programmable Read Only Memory*

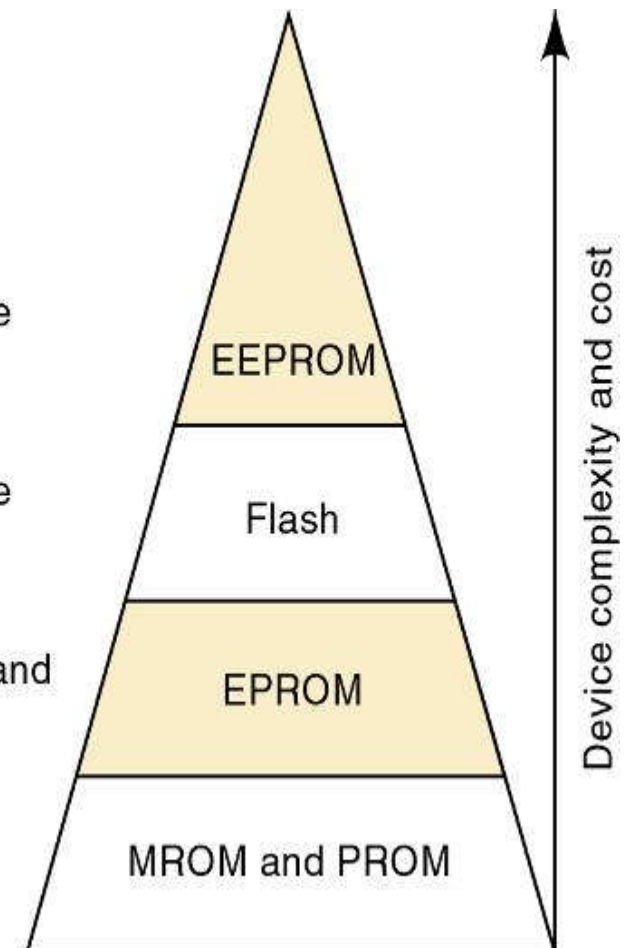


In-circuit, electrically erasable byte-by-byte

In-circuit, electrically erasable by sector or in bulk (all cells)

UV erasable in bulk; erased and reprogrammed out of circuit

Cannot be erased and reprogrammed



# Tipos de Memória

**Voláteis** (dados perdidos quando desenergizadas)

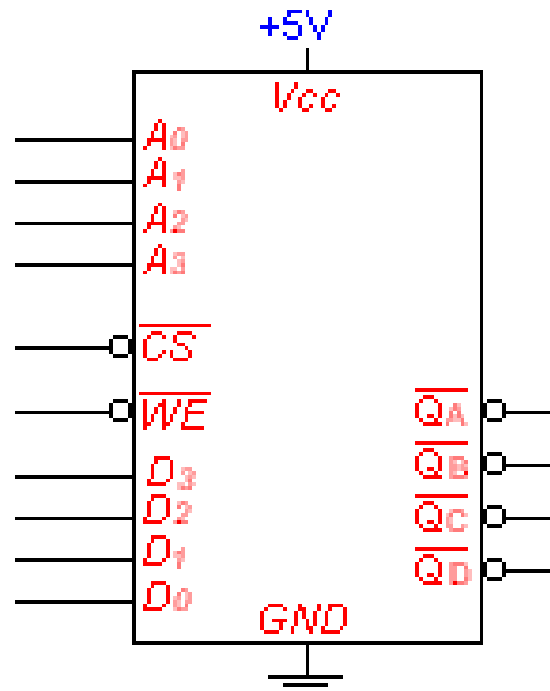
- ❑ **RAM: *Random-Access Memory*** – pode-se escrever e ler nas mesmas
- ❑ **SRAM: RAM estática** – utiliza flip-flops
- ❑ **DRAM: RAM dinâmica** – utiliza capacitores MOS – requer recarga periódica (*refresh*)

## Exemplo de Memória SRAM (Static Random Access Memory) 16x4: 74189

Barramento de  
endereços

Chip Select  
Write Enable

Barramento de  
dados de entrada



Barramento de  
dados de saída (invertidas)

$2^4 = 16$  palavras de 4 bit

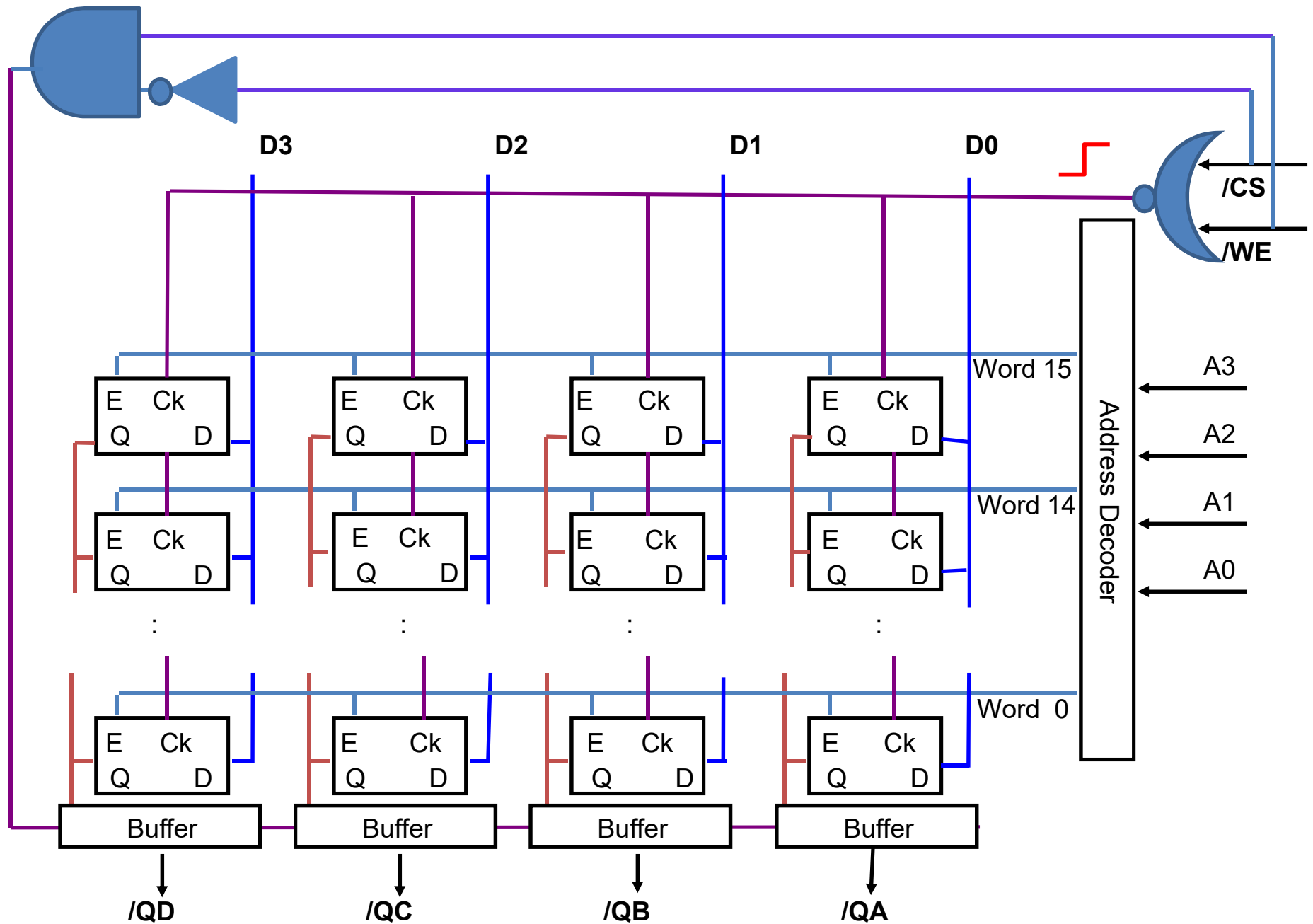
Endereços de 0000B a 1111B

Modo de operação	Entradas de controle		Saídas
	$\overline{CS}$	$\overline{WE}$	$\overline{O}$
Escrever	0	0	Estado lógico 1
Ler	0	1	Complemento dos dados armazenados na memória
Manter	1	X	Estado lógico 1

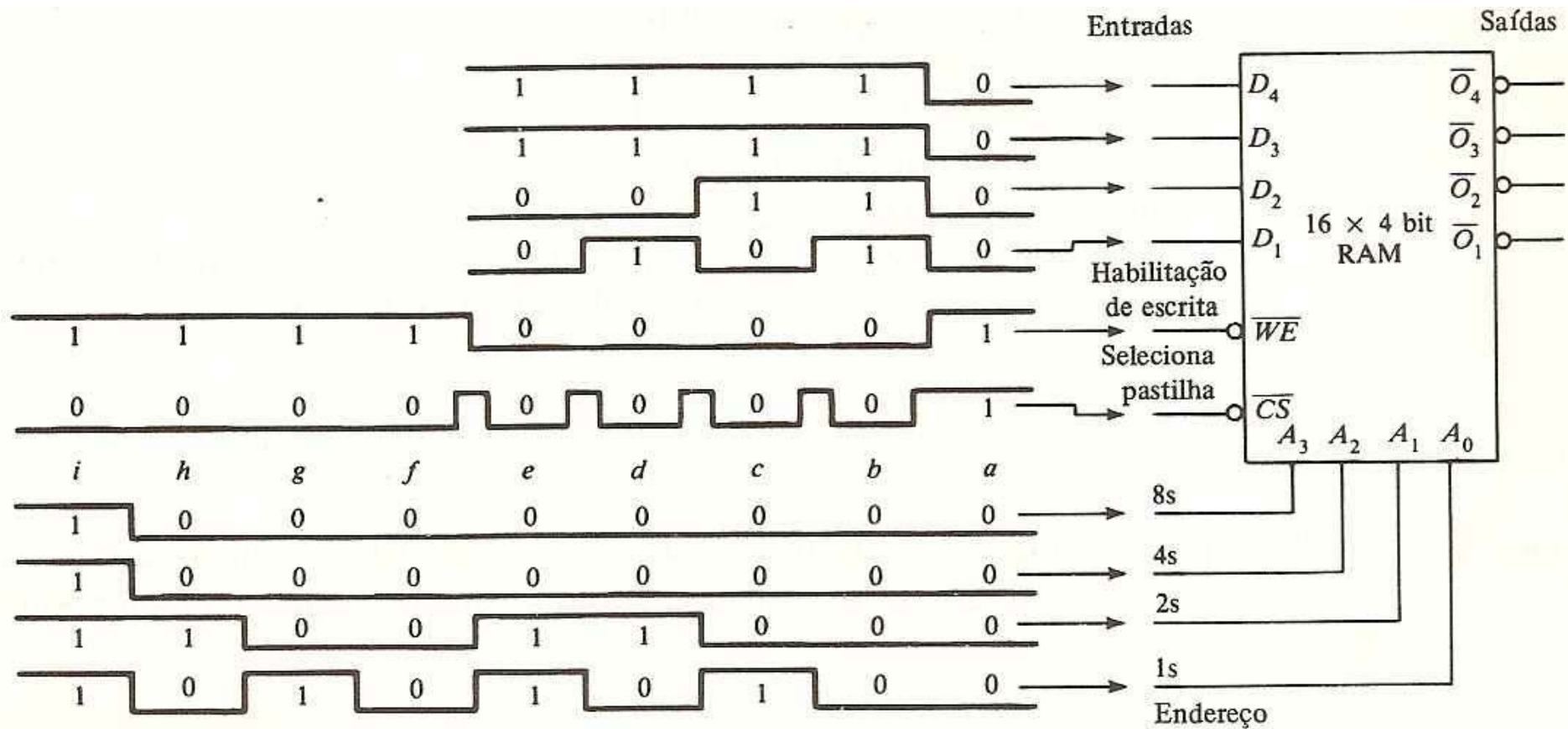
X = irrelevante

(c) Tabela-verdade de modos para a RAM de 64 bits

# Representação da 74189

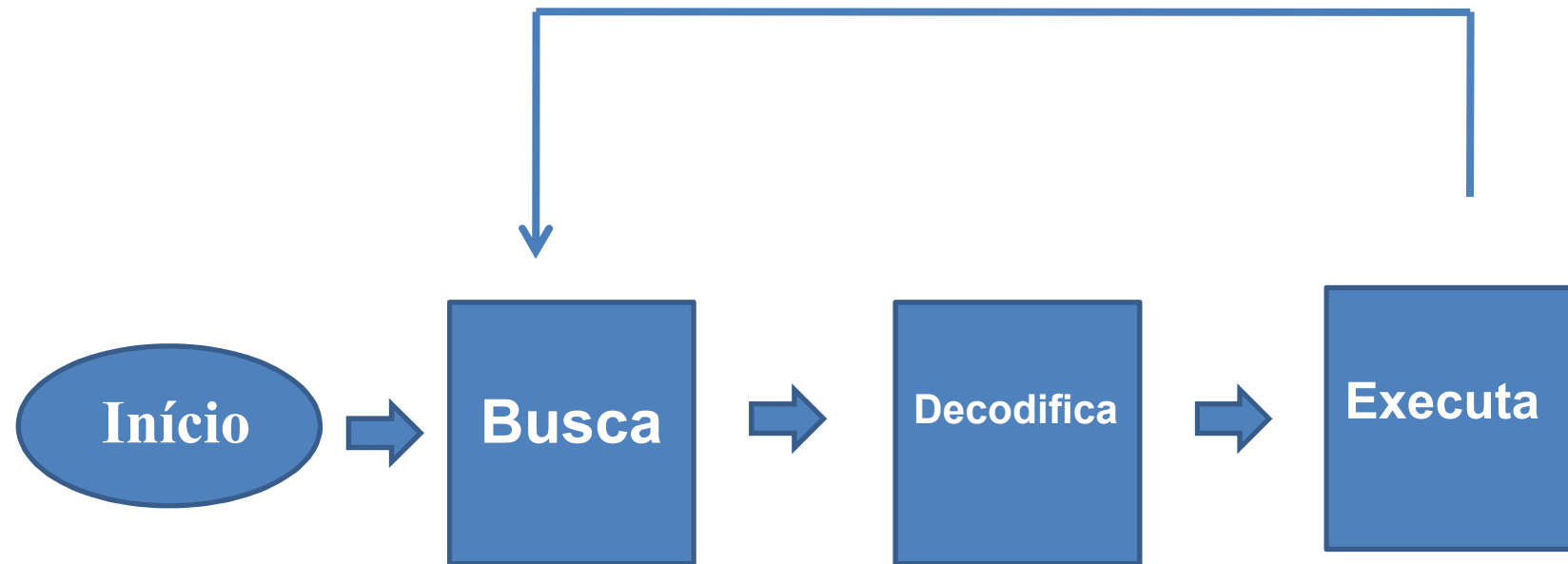


## Exemplo de Acesso a RAM 16x4



- Memórias mais atuais armazenam palavras de oito bits; barramento de dados único e bidirecional; quando  $\overline{CS}=1$ , barramento de saída em estado de alta impedância

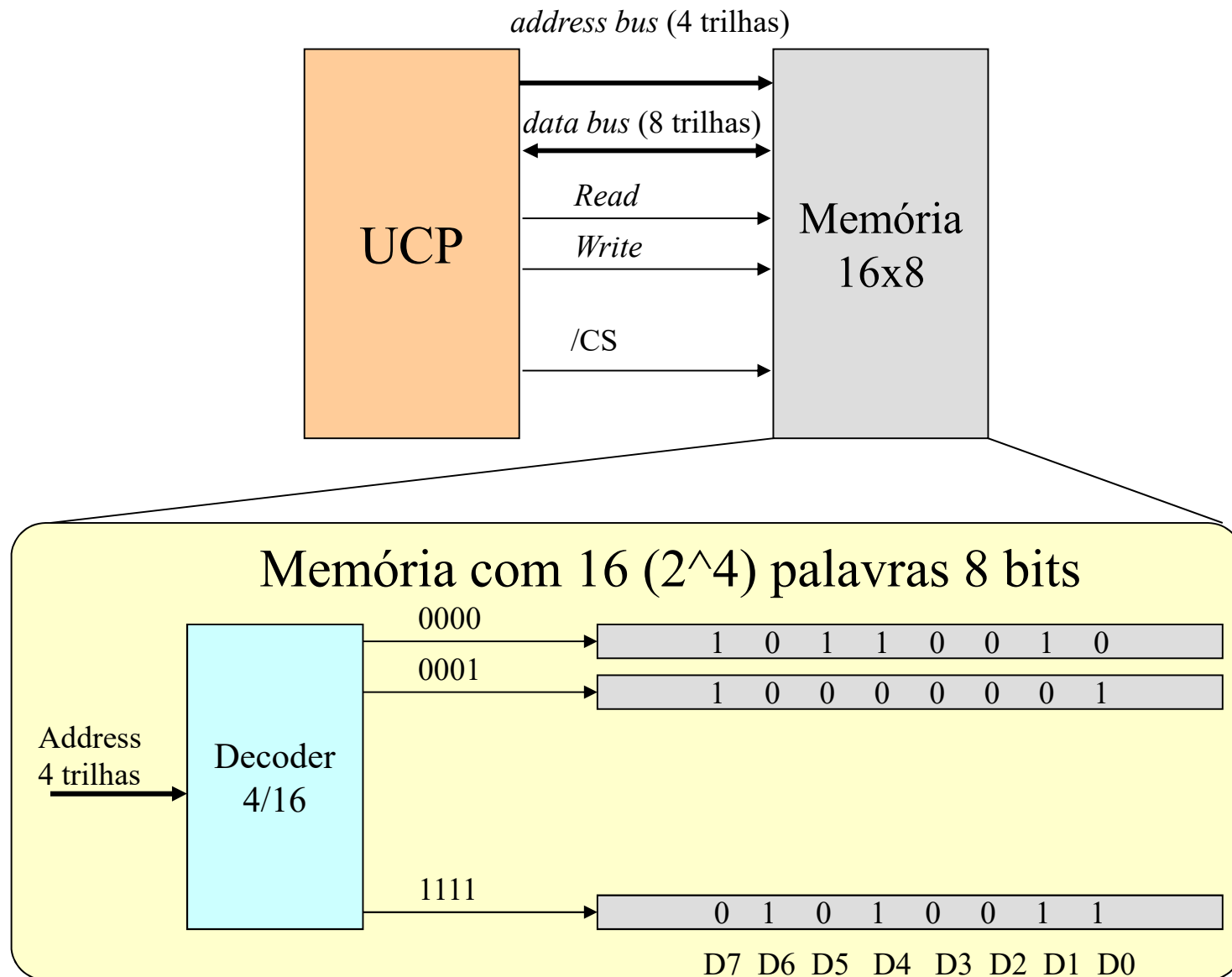
## Funções Básicas de Unidade Central de Processamento (UCP)



- **Busca instruções armazenadas em memória de programa, decodifica-as e executa-as;**
- **Supre sinais de controle para acesso à memória e outros periféricos;**
- **Transfere dados da ou para a memória e periféricos;**
- **Atende demanda dos periféricos (interrupções).**

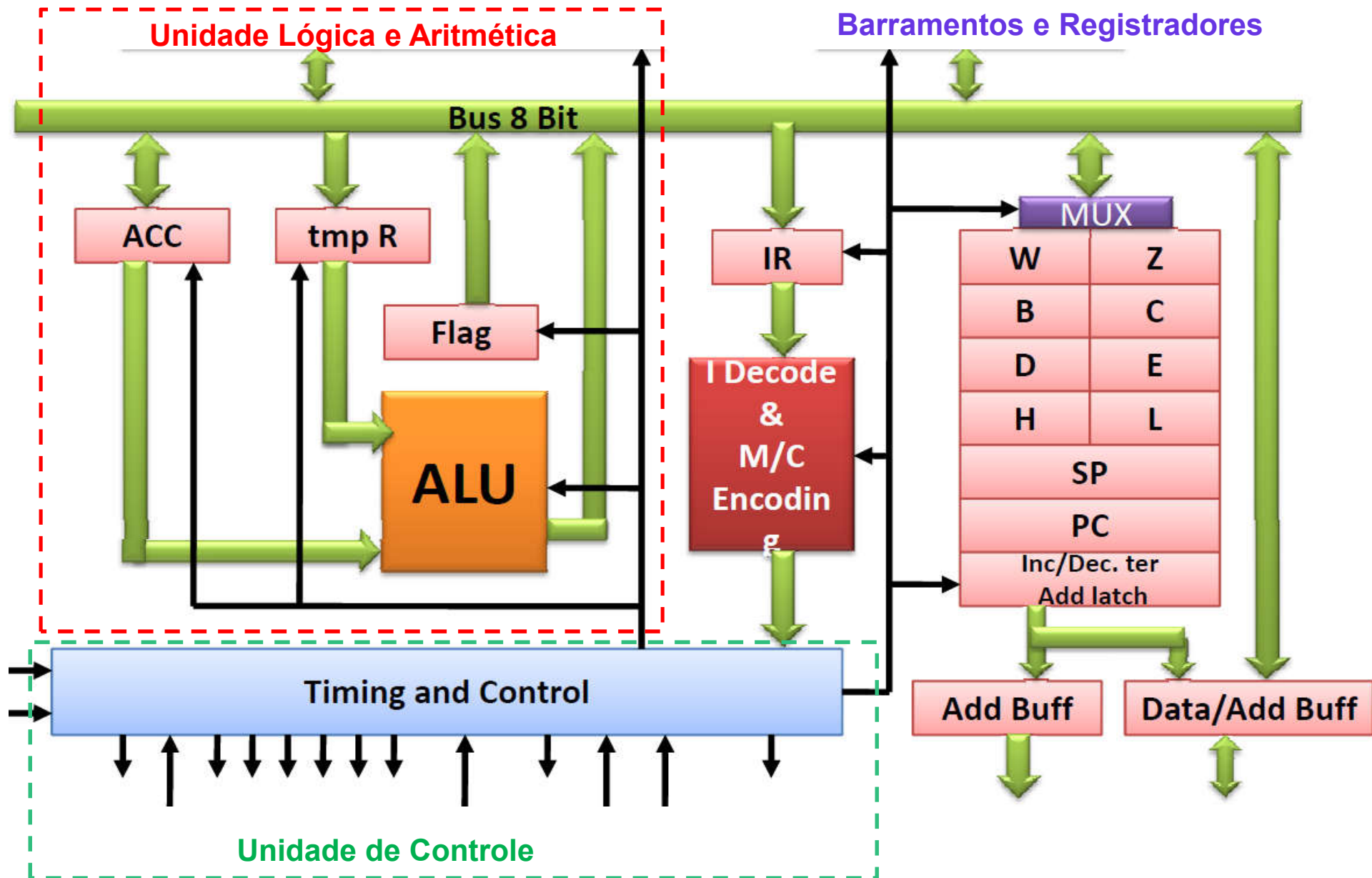


# Interface entre processador e memória RAM



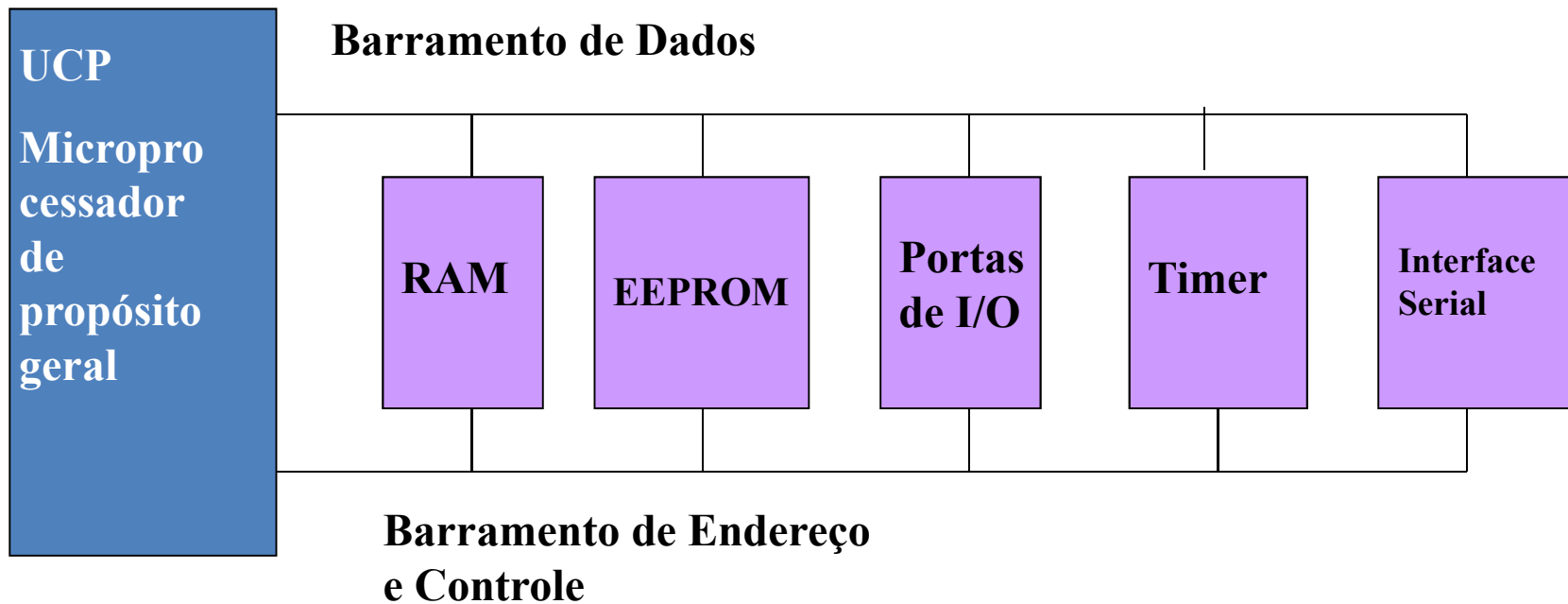
- Barramentos de endereço, dados e controle (*Read*, *Write*, */CS*)

# Esquema simplificado de uma UCP



# Microprocessador de Propósito Geral

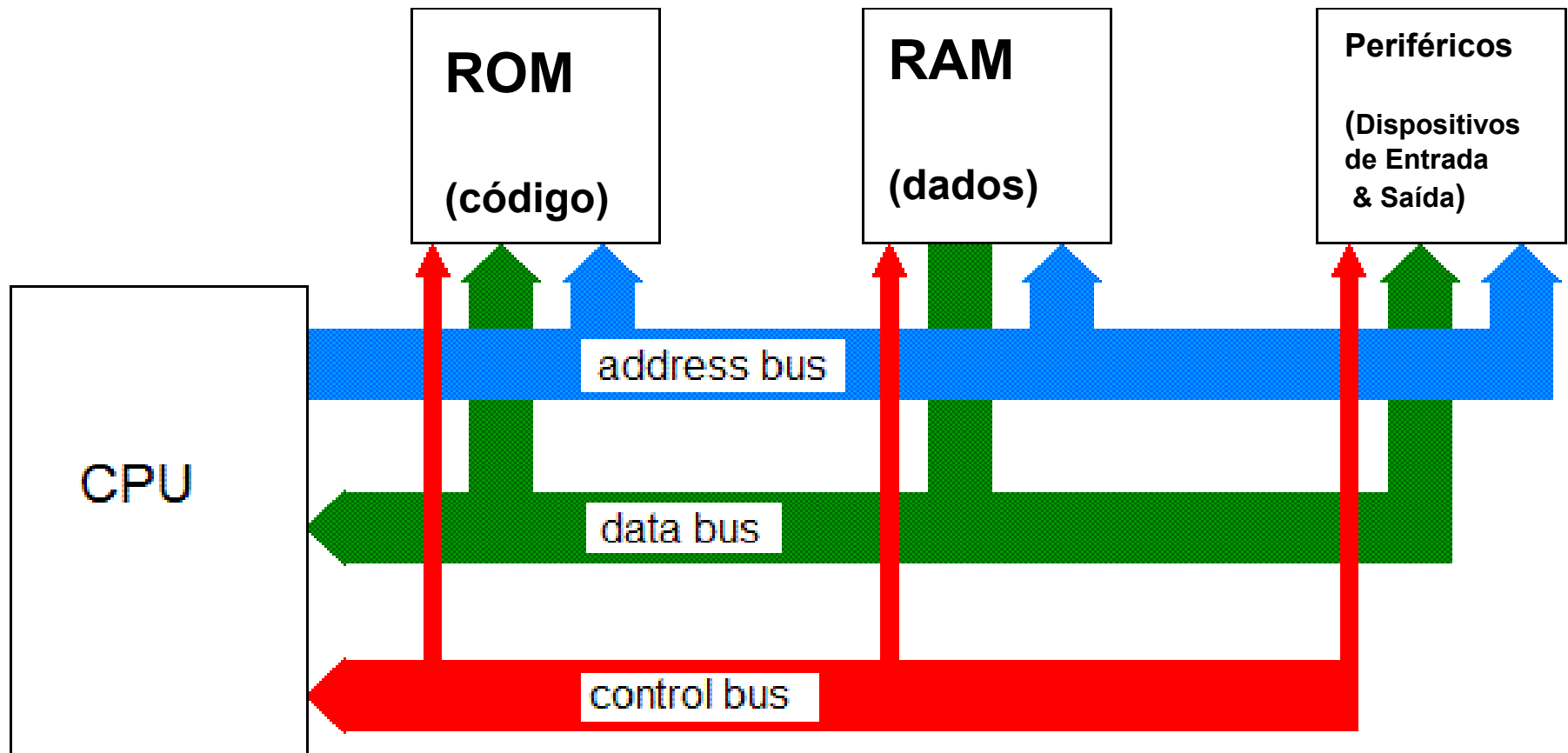
**Sem RAM, ROM ou dispositivos de entrada e saída (I/O)**



**Vantagem:** flexibilidade, sistema expansível ;

**Desvantagem:** custo, roteamento de placa e dimensões do circuito.

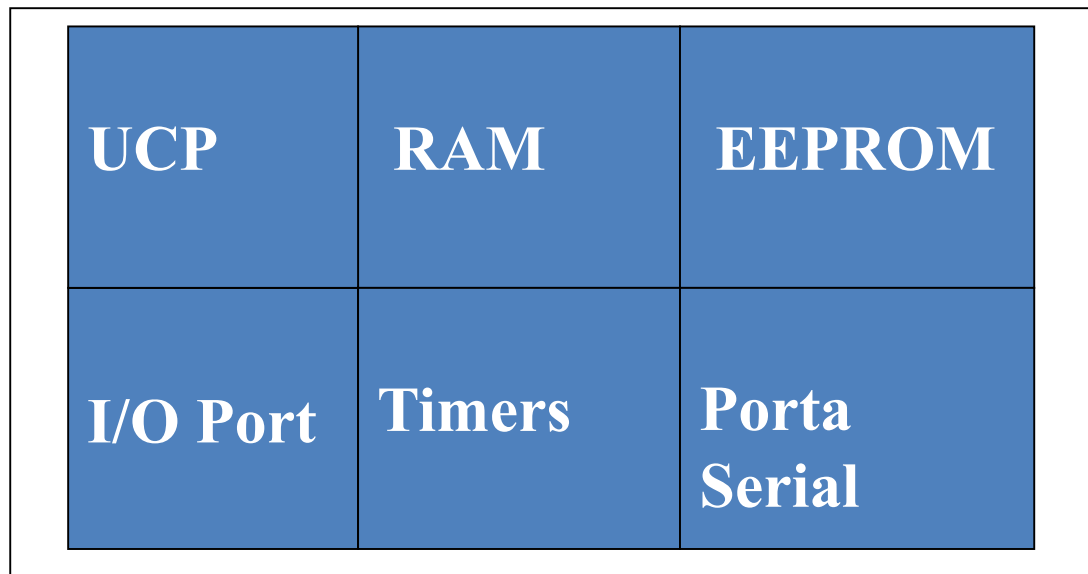
## Conexão de Microprocessador de Propósito Geral a Memórias e Periféricos



**O número de trilhas em cada barramento varia em função da capacidade de armazenamento da memória; tamanho da palavra armazenada nas memórias e características do processador.**

# Microcontrolador

## ▪UCP + “Periféricos”



**Vantagem: menor custo, menor dimensão, rápido desenvolvimento;**

**Desvantagem: baixa flexibilidade, não expansível;**

# Microcontrolador

- **Módulos geralmente encontrados:**
  - **Portas de entrada e saída**
  - **Interface Serial (CAN<sup>1</sup>, SPI<sup>2</sup>, USB, RF e etc)**
  - **Memórias (ROM, RAM, EEPROM, XRAM, etc)**
  - **Conversores Analógico/Digital e Digital/Analógico**
  - **Timers**
  - **...**
  
- **Família de Microcontrolador: UCP + diferentes módulos**

<sup>1</sup> **CAN:** *Controller Area Network*

<sup>2</sup> **SPI:** *Serial Peripheral Interface*

# Upgrades da família 8051 da Atmel

Device	Flash (KiB)	USB	EEPROM (kiB)	RAM (Bytes)	F.max (MHz)	I/O Pins	UART	Timers	ADC
AT89C5115	16	--	2	512	40	20	1	2	Yes
AT89C5130A-M	16	Yes	1	1280	48	18/34	Yes	Yes	--
AT89C5131A-L	32	Yes	1	1280	48	18/34	Yes	Yes	--
AT89C5131A-M	32	Yes	1	1280	48	18/34	Yes	Yes	--
AT89C5132	64	Yes	--	2304	20	44	Yes	Yes	Yes
AT89C51AC2	32	--	2	1280	40	34	1	3	Yes
AT89C51AC3	64	--	2	2304	60	32	1	3	Yes
AT89C51CC01	32	--	2	1280	40	34	1	2	Yes
AT89C51CC02	16	--	2	512	40	20	1	1	Yes
AT89C51CC03	64	--	2	2304	40	34/37	1	2	Yes
AT89C51ED2	64	--	2	2048	60	32	1	3	--
AT89C51IC2	32	--	--	1280	60	34	1	3	--
AT89C51ID2	64	--	2	2048	60	32	1	3	--
AT89C51RB2	16	--	--	1280	60	32	1	3	--
AT89C51RC	32	--	--	512	33	32	1	3	--
AT89C51RC2	32	--	--	1280	60	32	1	3	--

# Escolhendo um Microcontrolador

1. **Considerar: consumo, desempenho, capacidade de memória, número de portas de entrada e saída, encapsulamento, tamanho, interfaces de comunicação, faixa de temperatura de operação, e demais módulos necessários ao projeto;**
2. **Custo e disponibilidade no mercado. Facilidade para *upgrade* (grande família)**
3. **Disponibilidade de ferramentas de desenvolvimento:**
  - **Compiladores, emuladores, simuladores, suporte técnico**



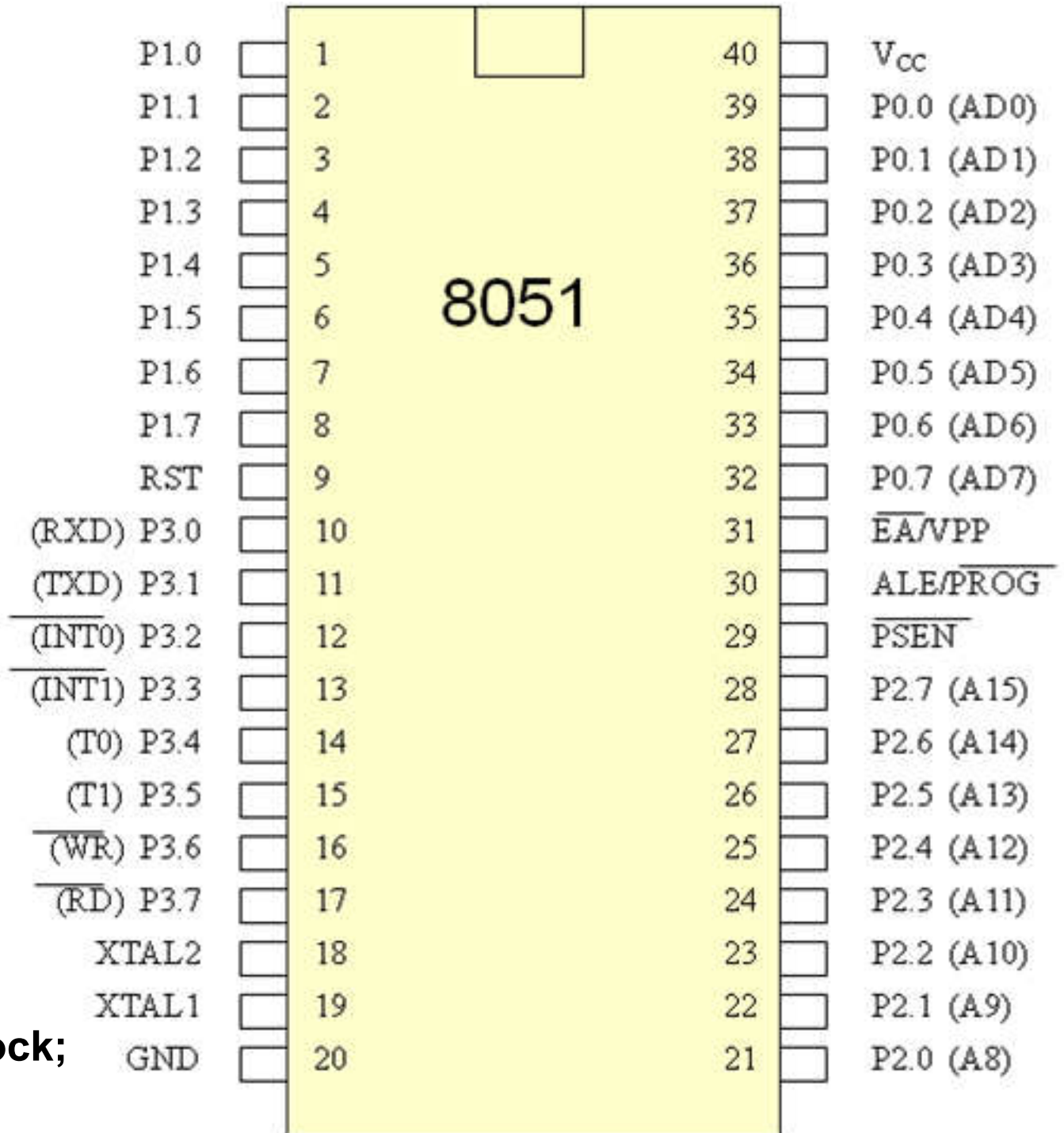
# 8051

- 1981
- UCP de 8 bits
- Clock até 12 MHz
- Vcc=+5V; 40 pinos;
- Endereça até 64 KiB  
(kibibyte = 1024 bytes  
IEEE 1541-2002):

1. de dados;
2. de programa;

- Possui pinos:

- ❖ Endereçamento;
- ❖ Dados;
- ❖ Controle;
- ❖ Energização e clock;
- ❖ e outros;



# 8051

Tipo de encapsulamento do 8051

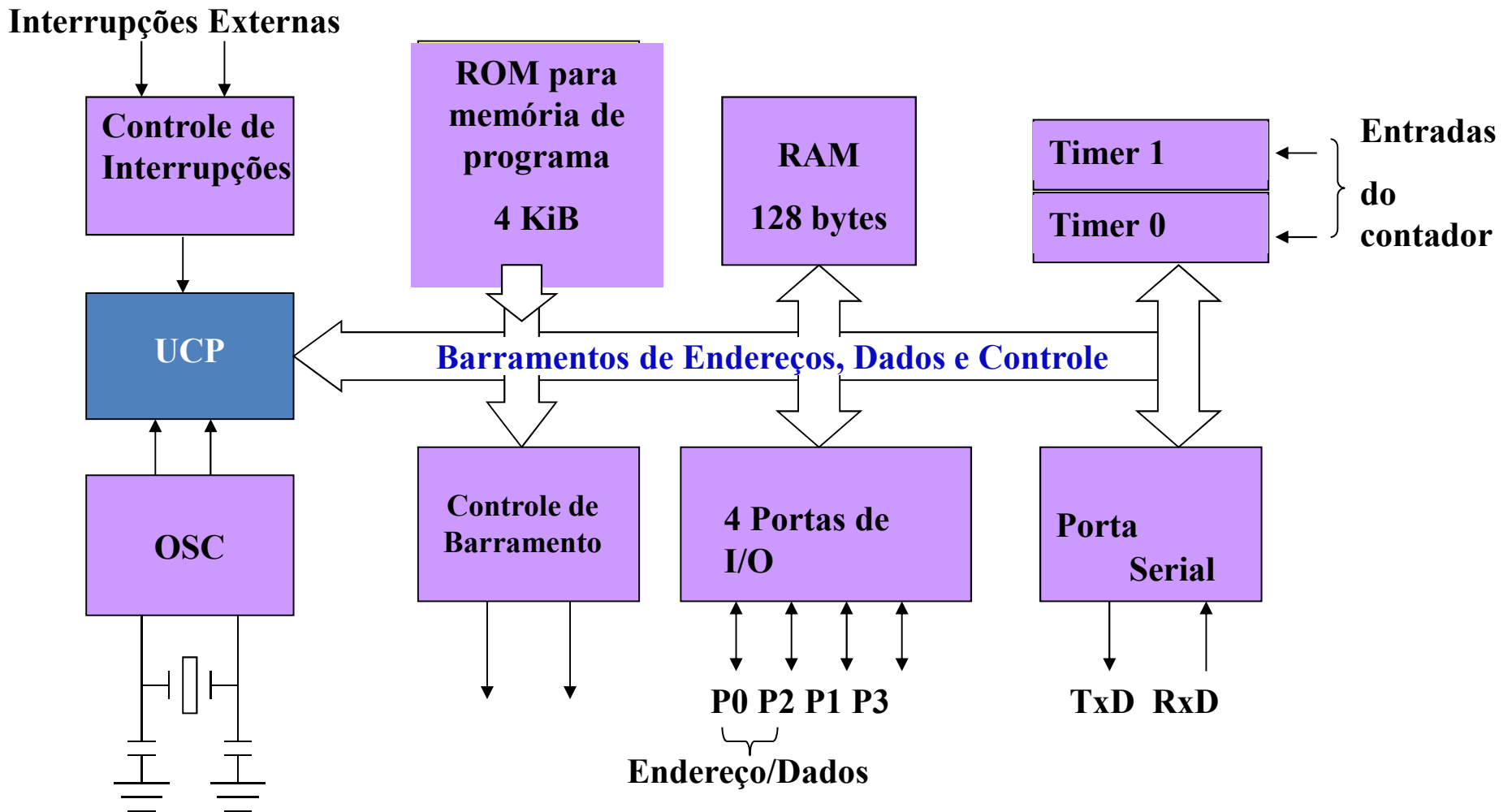


# 8051

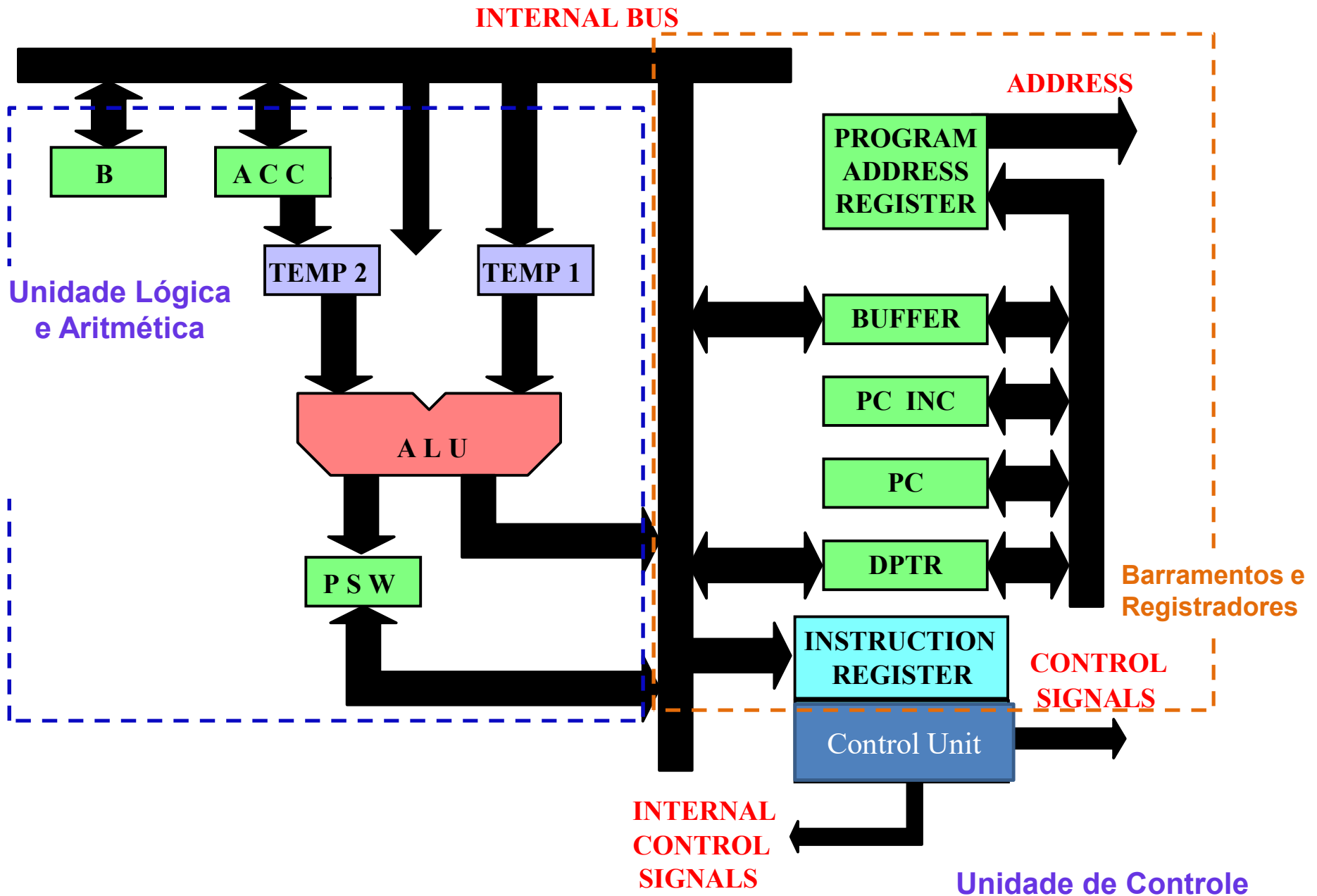
## ■ Características Básicas:

- ☐ UCP de 8 bits;
- ☐ endereça 64 KiB (kibibyte) de memória de programa externa;
- ☐ endereça 64 KiB de memória de dados externa;
  
- ☐ 4 KiB de memória ROM interna para programas;
- ☐ 128 bytes de memória RAM interna para dados;
- ☐ 4 portas de entrada e saída (8 pinos cada);
- ☐ 5 vetores de interrupção com 2 níveis de prioridade:
  - 2 interrupções externas
  - 2 temporizadores / contadores
  - 1 interface serial

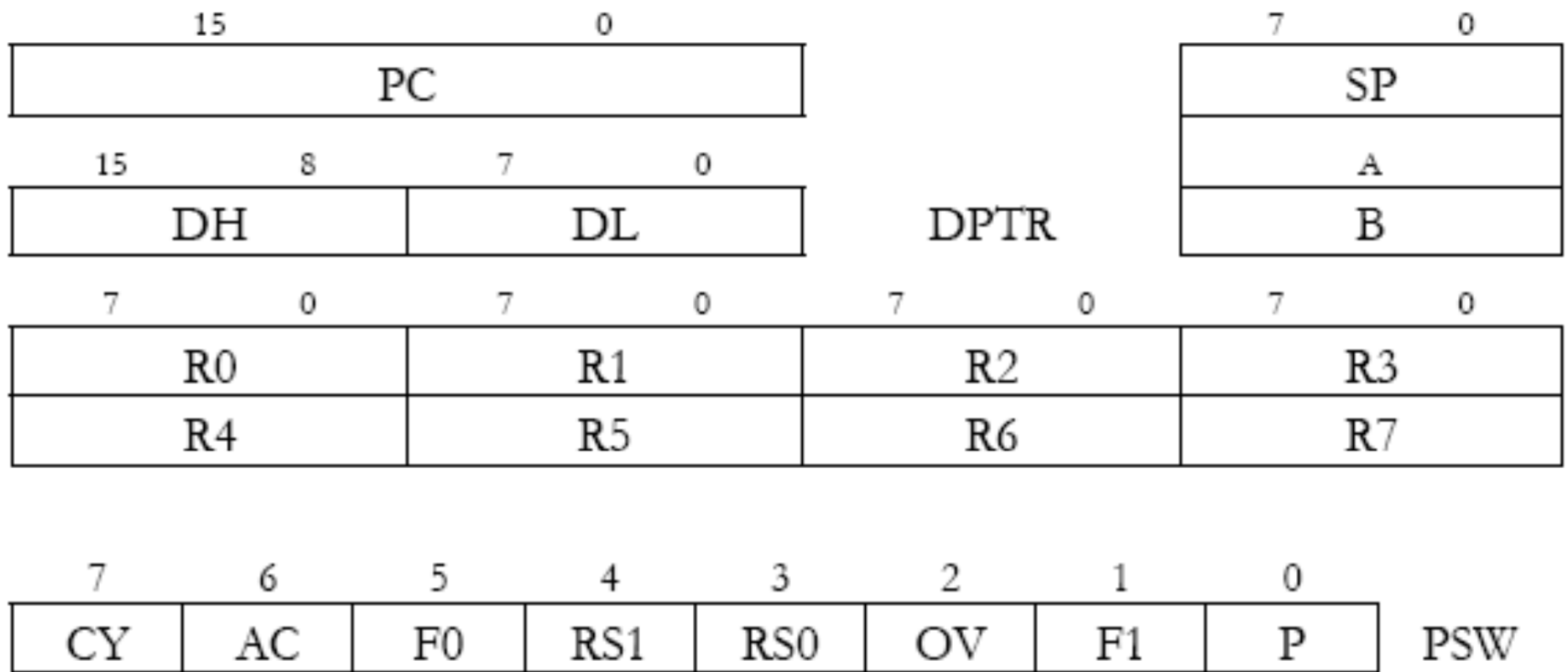
# 8051



## Arquitetura simplificada da UCP do 8051 + registradores utilizados pela mesma

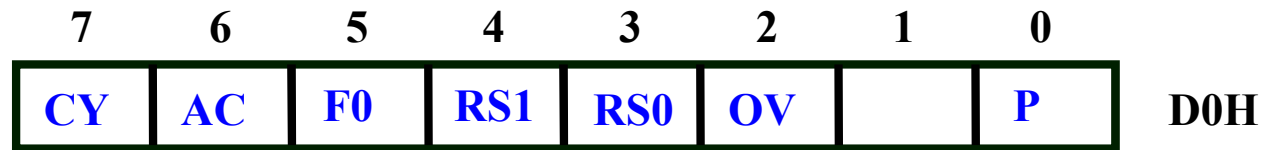


# Registradores do 8051



# Registrador de Flags

**PSW - *Program Status Word - Bit Addressable***



Nome	Localização	Descrição
CY	PSW.7	Carry flag
AC	PSW.6	Auxiliary carry flag
F0	PSW.5	Definido pelo usuário
RS1	PSW.4	Bit 1 do seletor de Register Bank
RS0	PSW.3	Bit 0 do seletor de Register Bank
OV	PSW.2	Overflow flag
	PSW.1	Definido pelo usuário
P	PSW.0	Flag de paridade. 1 = ímpar.

# MEMÓRIA DE PROGRAMA

## Contendo códigos de operação (Opcodes)

### Representação no Keil

ENDEREÇOS (HEXADECIMAL)	Instruções (BINÁRIO)
0000	1110 1010
0001	0010 0101
0002	0011 0010
0003	1000 0101
0004	1110 0000
0005	0011 0010
0006	1000 0000
....	

ENDEREÇOS (HEXADECIMAL)	Instruções (HEXADECIMAL)
0000	EA
0001	25
0002	32
0003	85
0004	E0
0005	32
0006	80
....	....

### Exemplo de Alocação de Código na Memória de Programa



<b>Mnemônicos</b>	<b>Descrição</b>	<b>Operação / Exemplo</b>	<b>Flags</b>
<b>MOV A,Rn</b>	<b>A = Rn</b>	<b>MOV A,R3</b>	<b>P</b>
<b>ADD A,Rn</b>	<b>A = A+Rn</b>	<b>ADD A, R7</b>	<b>C, OV, AC, P.</b>
<b>DA A</b>	<b>O conteúdo de A é convertido para nro decimal de 2 dígitos</b>	<b>Se <math>[A_{3-0} &gt; 9</math> ou <math>AC = 1</math>]</b> <b>então <math>(A_{3-0}+6)</math></b>  <b>Se <math>[A_{7-4} &gt; 9</math> ou <math>C = 1</math>]</b> <b>então <math>(A_{7-4}+6)</math></b>	<b>C, P</b>
<b>RL A</b>	<b>rotaciona o conteúdo do acumulador para a esquerda. O bit 7 é carregado com bit 0.</b>	<b><math>(A_{n+1}) \leq (A_n)</math></b>  <b><math>(A_0) \leq (A_7)</math></b>	

**Rn:** pode ser qualquer um dos registradores de R0 a R7

## Instrução **MOV A,Rn**

5 bits Código da instrução ( <i>opcode</i> )	3 bits Operando
1    1    1    0	1    R    R    R

## Instrução **MOV A,direct (endereço)**

8 bits Código da instrução ( <i>opcode</i> )	8 bits
1    1    1    0    0    1    0    1	endereço

## Valores possíveis para o campo RRR do slide anterior e seguinte

REGISTRADOR	RRR
<b>R0</b>	<b>000</b>
<b>R1</b>	<b>001</b>
<b>R2</b>	<b>010</b>
<b>R3</b>	<b>011</b>
<b>R4</b>	<b>100</b>
<b>R5</b>	<b>101</b>
<b>R6</b>	<b>110</b>
<b>R7</b>	<b>111</b>

## Instrução **ADD A,Rn**

5 bits Código da instrução ( <i>opcode</i> )	3 bits Operando
0   0   1   0	1   R   R   R

## Instrução **ADD A,direct**

8 bits Código da instrução ( <i>opcode</i> )	8 bits
0   0   1   0   0   1   0   1	endereço

## Exercício: Obter o código das instruções abaixo

MOV A,R2  
ADD A,32H

Mnemônico	Código Binário	Código Hexadecimal
MOV A,R2	1110 1010	EA
ADD A,32H	0010 0101 0011 0010	25 , 32

# MEMÓRIA DE PROGRAMA

## Contendo códigos das instruções (*opcodes*)

### Representação no Keil

ENDEREÇOS (HEXADECIMAL)	Instruções (BINÁRIO)
0000	1110 1010
0001	0010 0101
0002	0011 0010
0003	....
0004	....
0005	....
0006	....
....	....

ENDEREÇOS (HEXADECIMAL)	Instruções (HEXADECIMAL)
0000	EA
0001	25
0002	32
0003	....
0004	....
0005	....
0006	....
....	....

### Exemplo de Alocação de Código na Memória de Programa

## **Flags do registrador PSW alterados por instruções** (diferentes instruções podem afetar os flags de diferentes maneiras)

**P** (Parity) = 1 => Se o acumulador contém nro ímpar de 1's.

**AC** (Auxiliary Carry) = 1 => ocorreu vai um do flip-flop (FF) 3 para o 4;

**CY** (Carry) = 1 => ocorreu vai um do FF 7;  
Indica que operação extrapola capacidade do registrador;

**OV** (Overflow) = 1 => ocorreu vai um do FF 6 para 7, mas não do FF 7;  
ou ocorreu vai um do FF 7, mas não do FF 6 para o 7;

Ao somar inteiros com sinal, OV indica nro negativo resultante da soma de nros positivos ou nro positivo resultante da soma de nros negativos.

## *Auxiliary Flag & BCD (Binary Coded Decimal)*



### **Soma BCD**

**Supondo flags atuais como**



CY	AC	OV	P
0	0	0	1

**[A]=98H; [B]=08H    ADD A,B**



CY	AC	OV	P
0	1	0	0

**[A]=A0H; [B]=08H    DA A**



CY	AC	OV	P
1	1	0	0

**[A]=06H; [B]=08H**

*Decimal-adjust Accumulator*



## Flag de Overflow

**OV** (*Overflow*) = 1 => ocorreu vai um do FF 6 para FF 7, mas não do FF 7 para *carry*; ou ocorreu vai um do FF 7, mas não do FF 6 para o FF 7;

CY	AC	OV	P
----	----	----	---

1) Supondo *flags* atuais como



0	0	0	1
---	---	---	---

[A]=7FH; [B]=01H

ADD A,B



0	1	1	1
---	---	---	---

[A]=80H; [B]=01H

2) Supondo *flags* atuais como



0	0	0	1
---	---	---	---

[A]=80H; [B]=80H (-128D)

ADD A,B



1	0	1	0
---	---	---	---

[A]=00H; [B]=80H

3) Supondo *flags* atuais como



0	0	0	0
---	---	---	---

[A]=C0H; [B]=C0H (-64D)

ADD A,B



1	0	0	1
---	---	---	---

[A]=80H; [B]=C0H

## Instruções que afetam os *flags* C, OV e AC

	C	OV	AC
<b>ADD</b>	<b>X</b>	<b>X</b>	<b>X</b>
<b>ADDC</b>	<b>X</b>	<b>X</b>	<b>X</b>
<b>SUBB</b>	<b>X</b>	<b>X</b>	<b>X</b>
<b>MUL</b>	<b>0</b>	<b>X</b>	
<b>DIV</b>	<b>0</b>	<b>X</b>	
<b>DA</b>	<b>X</b>		
<b>RRC</b>	<b>X</b>		
<b>RLC</b>	<b>X</b>		
<b>CJNE</b>	<b>X</b>		

**OBS:** *Flag* de paridade reflete qualquer alteração de conteúdo do acumulador; se o acumulador contém nro ímpar de 1's, P='1'.

# Conversão de linguagem de alto nível para código de máquina

High-level  
language  
program  
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Compiler

Assembly  
language  
program  
(for MIPS)

```
swap:
    muli $2, $5, 4
    add  $2, $4, $2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```

Assembler

Binary machine  
language  
program  
(for MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
000000111110000000000000000001000
```

## Modos de Endereçamento

**MOV A,Rn** ; via registrador -- ADD A,R7

**MOV A,direct** ; direto -- ADD A,7FH

**MOV A,@Ri** ; indireto – MOVX A,@DPTR

**DA A** ; registrador específico

**MOV A,#data** ; imediato -- ADD A,#127

**MOVC A,@A+DPTR** ; indexado -- MOVC A,@A+PC

**Rn** - registrador R0 a R7 do banco selecionado (PSW).

**direct** - endereçamento direto. *Direct* é o endereço de uma posição da memória RAM interna ou SFR (*Special Function Registers*)

**@Ri** - endereçamento indireto a uma posição de memória RAM interna (**Ri=R0 ou R1**) ou externa (MOVX; P2 deve conter byte mais significativo do endereço da posição de memória; P0=Ri)

**#data** - **data** é um valor de 8 bits incluído no corpo da instrução.

**#data16** - **data16** é um valor de 16 bits incluído no corpo da instrução.

## Exercício 1

1:  $B = N, A = 0$

2:  $A = A + B; B = B - 1$

3: Se  $B \neq 0$  então VÁ\_PARA 2 (?)

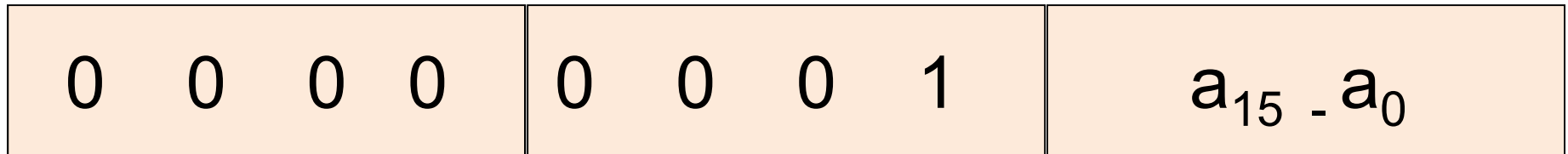
4: total = valor final de A

OBS: Fazer total corresponder à posição de memória  
RAM interna 20h

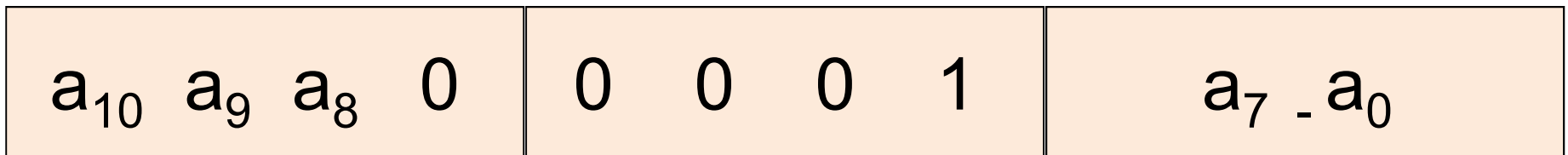
# Mnemônicos para Desvio Incondicional

## (2 cycles)

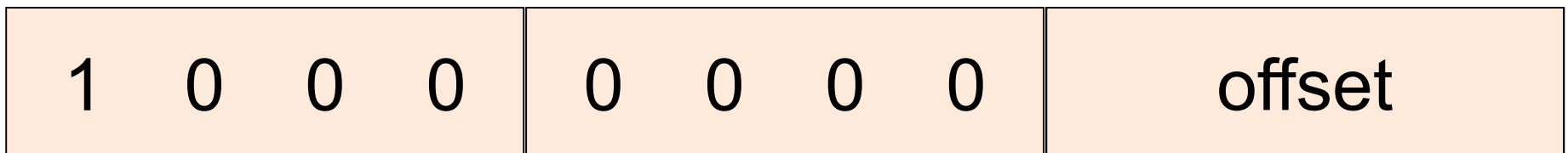
**LJMP:** Especifica endereço de 16 bits. A instrução possui 3 bytes (opcode + 16 bits de endereço).



**AJMP:** Especifica endereço de 11 bits. A instrução possui 2 bytes (opcode + 11 bits de endereço).



**SJMP:** Especifica *offset* (-128 to +127) a ser somado ao PC para acessar a próxima instrução. A instrução possui 2 bytes (opcode + *offset*).



# Mnemônicos para Desvio Condicional

Mnemônico	Descrição
<b>JZ</b> <rel addr>	Salta se $A = 0$
<b>JNZ</b> <rel addr>	Salta se $A \neq 0$
<b>JC</b> <rel addr>	Salta se $C = 1$
<b>JNC</b> <rel addr>	Salta se $C \neq 1$
<b>JB</b> <bit>, <rel addr>	Salta se bit = 1
<b>JNB</b> <bit>, <rel addr>	Salta se bit $\neq 1$
<b>JBC</b> <bit>, <rel addr>	Salta se bit = 1, limpa bit

**<rel addr>** byte (em complemento de dois) que é somado ao  $PC_{LSB}$ , sendo este substituído pelo resultado da adição, especificando assim, o endereço de desvio do fluxo de execução do programa

# Mnemônicos para Desvio Condicional

Mnemônico	Descrição
<b>CJNE A, direct, &lt;rel addr&gt;</b>	Compara conteúdo de A a conteúdo de memória. Salta se não igual
<b>CJNE A, #data, &lt;rel addr&gt;</b>	Compara conteúdo de A a dado. Salta se não igual
<b>CJNE Rn, #data, &lt;rel addr&gt;</b>	Compara conteúdo de Rn a dado. Salta se não igual
<b>CJNE @Ri, #data, &lt;rel addr&gt;</b>	Compara conteúdo de memória apontada por Ri a dado. Salta se não igual
<b>DJNZ Rn, &lt;rel addr&gt;</b>	Decrementa conteúdo de Rn e salta se não zero
<b>DJNZ direct, &lt;rel addr&gt;</b>	Decrementa conteúdo de memória e salta se não zero



## Exercício 1 - Solução

Total EQU 20h

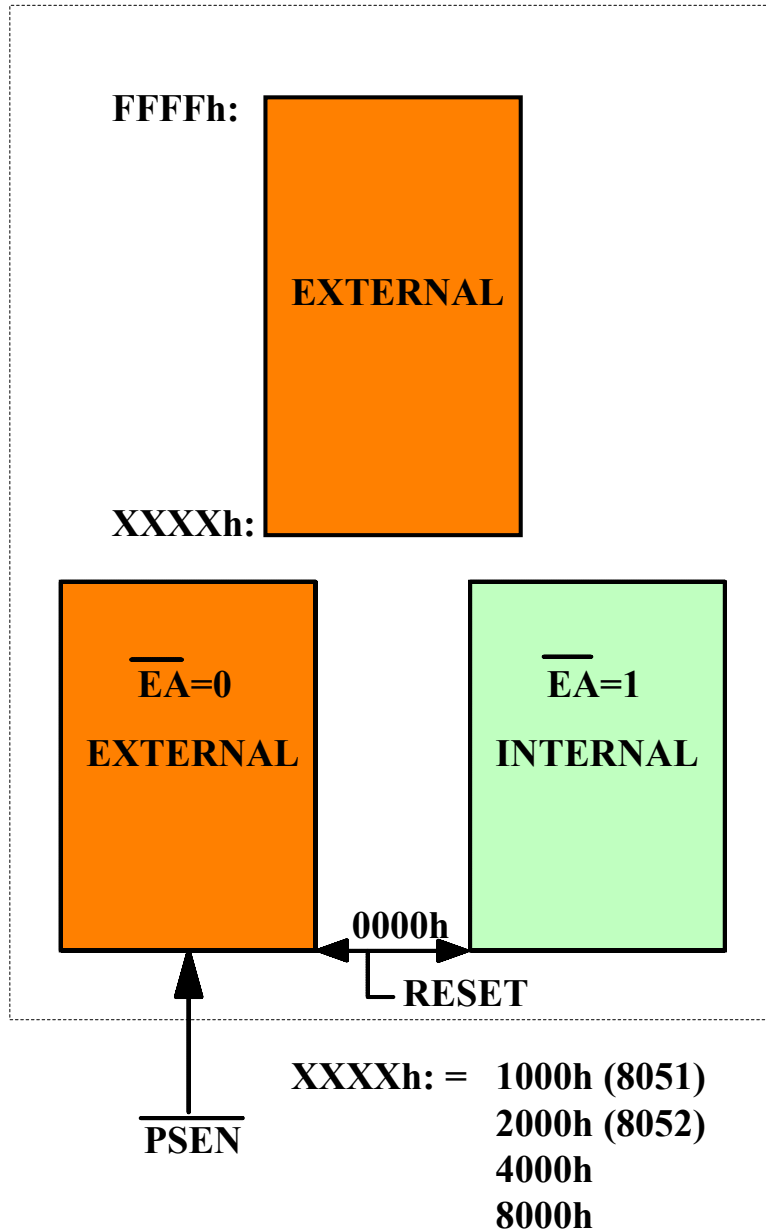
; Endereço de memória RAM

```

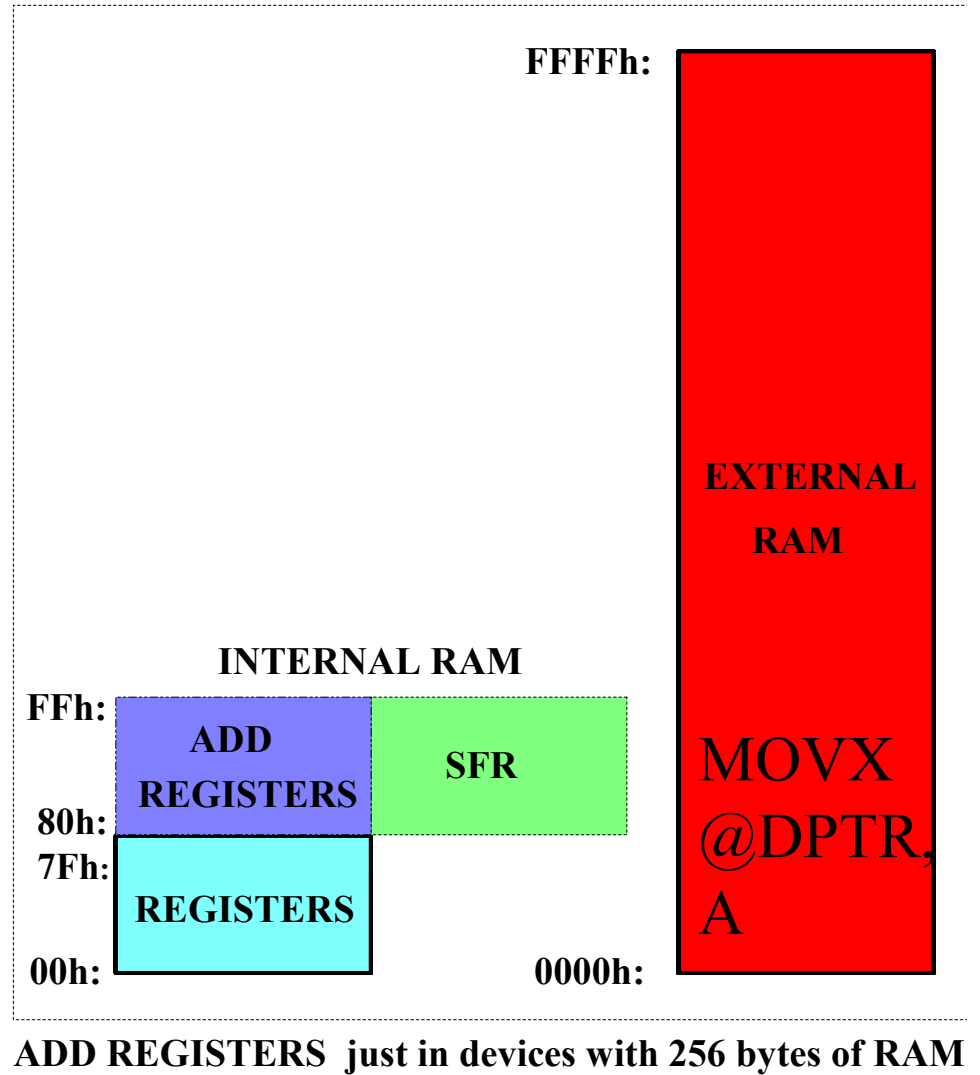
    MOV    A,#0
    MOV    DPTR,#N
    MOVC   A,@A+DPTR
    MOV    B, A                ; B = N
    XRL    A,Acc                ; soma = A  $\oplus$  A = 0
Loop:  ADD  A,B                  ; soma = soma + B
    DJNZ   B,Loop              ; B = B - 1; Se B  $\neq$  0 então VÁ_PARA Loop
    MOV    Total,A             ; Total = soma
FIM:   JMP  FIM                 ; Equivalente a jmp $
N:     DB   5
    END
```

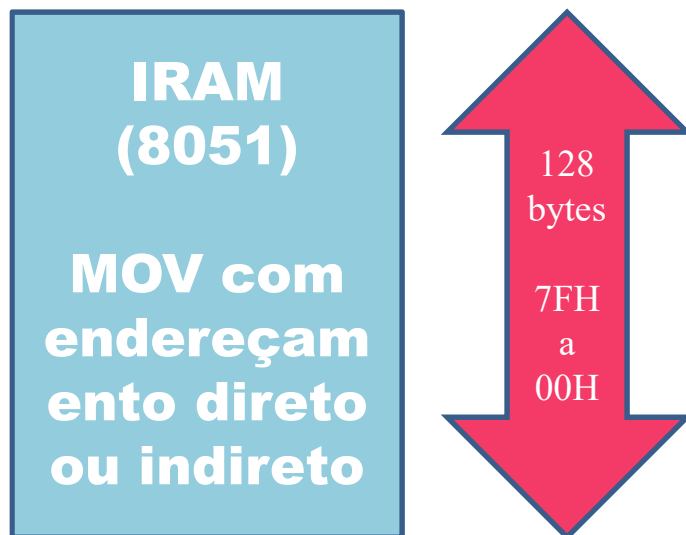
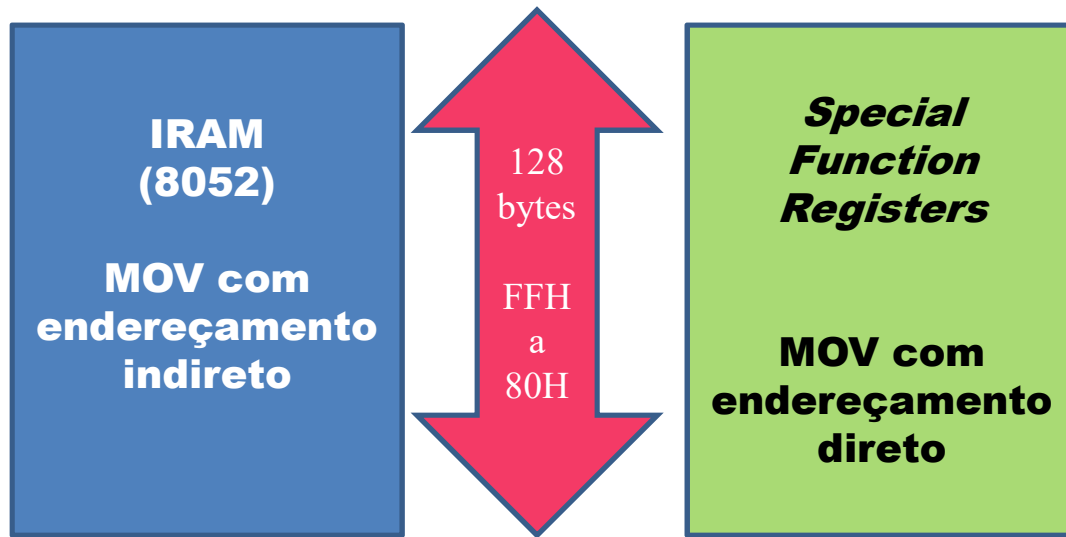
# Organização da Memória

## MEMÓRIA DE PROGRAMA

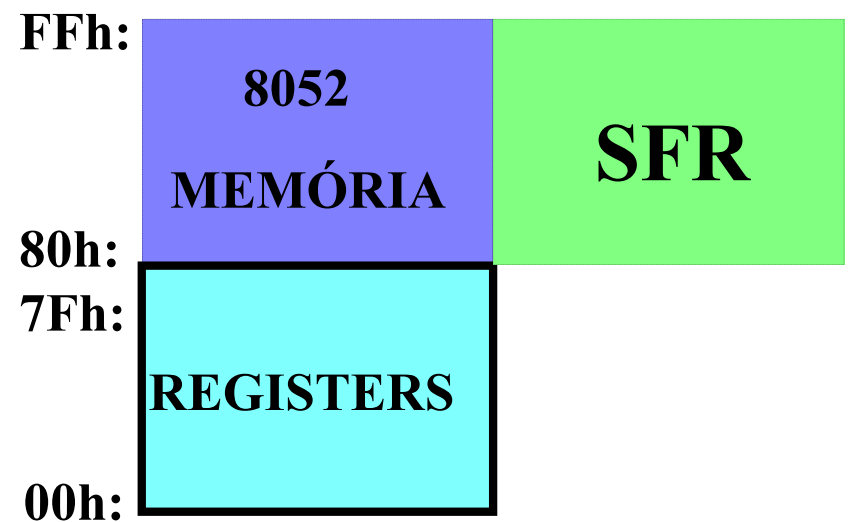


## MEMÓRIA DE DADOS






## Organização da RAM Interna (IRAM)

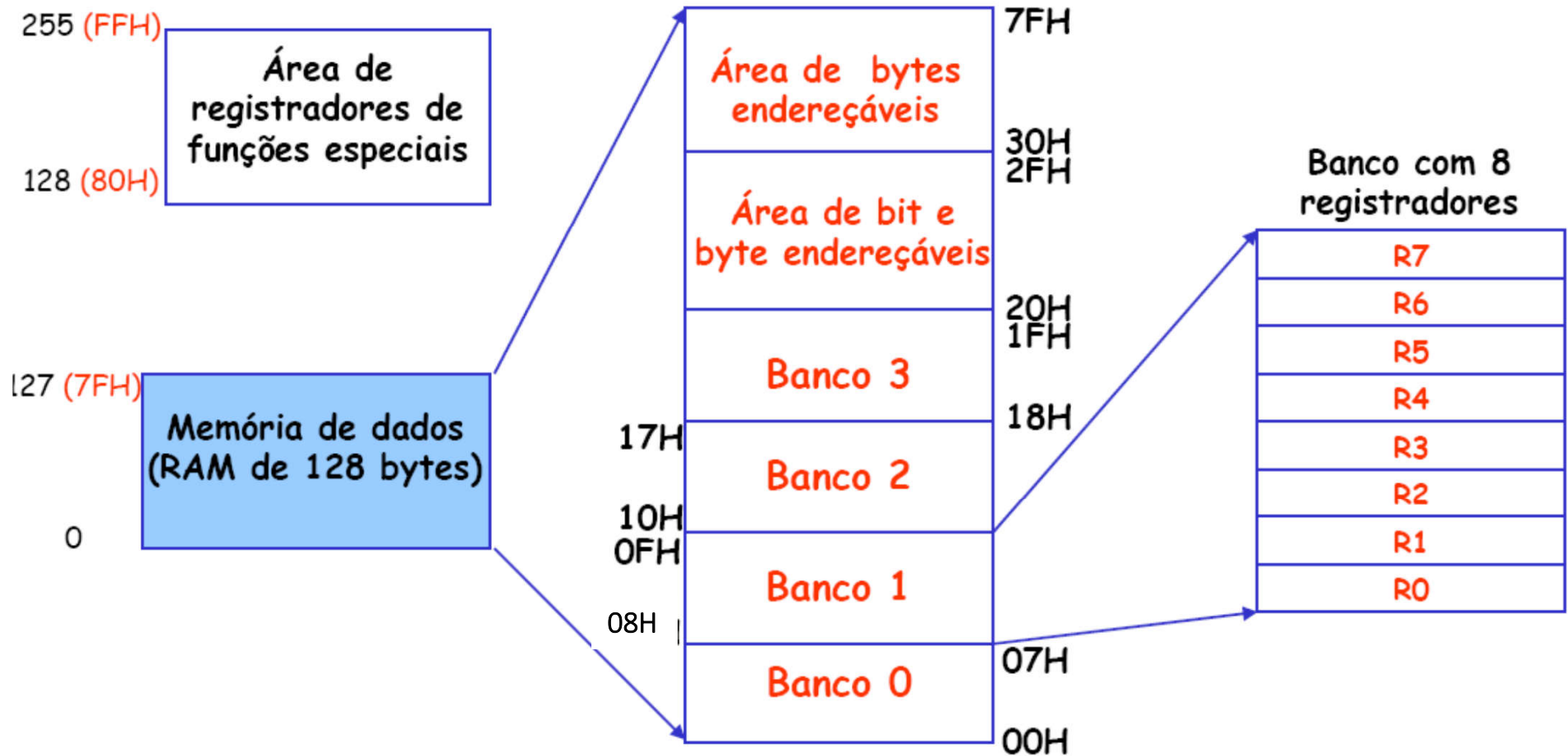


# *Special Function Registers (SFR)*

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87


 Bit-addressable Registers

# Organização da RAM Interna



**RS0 e RS1 do  
registrador PSW**

	RS0=0	RS0=1
RS1=0	<b>Banco 0</b>	<b>Banco 1</b>
RS1=1	<b>Banco 2</b>	<b>Banco 3</b>

## Exemplo de uso de bancos de registradores

(OBS: Ver pilha e subrotina para compreender modesta redução do código da subrotina com uso de banco)

### ; Programa Principal

```
CLR    RS0
CLR    RS1
MOV     SP,#10H
```

.....

```
MOV     R0,#5
MOV     R1,#0
```

```
RPSUB: MOV     A,P2
        PUSH    ACC
        CALL    SUBRT
        POP     ACC
        ADD     A,R1
        MOV     R1,A
        DJNZ    R0, RPSUB
```

.....

### ; V1 - SEM MUDAR O BANCO DE REGISTRADORES SUBRT:

```
        PUSH    00H
        PUSH    01H
        MOV     R0,#8
        MOV     R1,#0
Loop:   RL      A
        JNC     Salta
        INC     R1
Salta :  DJNZ    R0,Loop
        MOV     P1,R1
        POP     01H
        POP     00H
        RET
```

### ; V2 - MUDANDO O BANCO DE REGISTRADORES SUBRT:

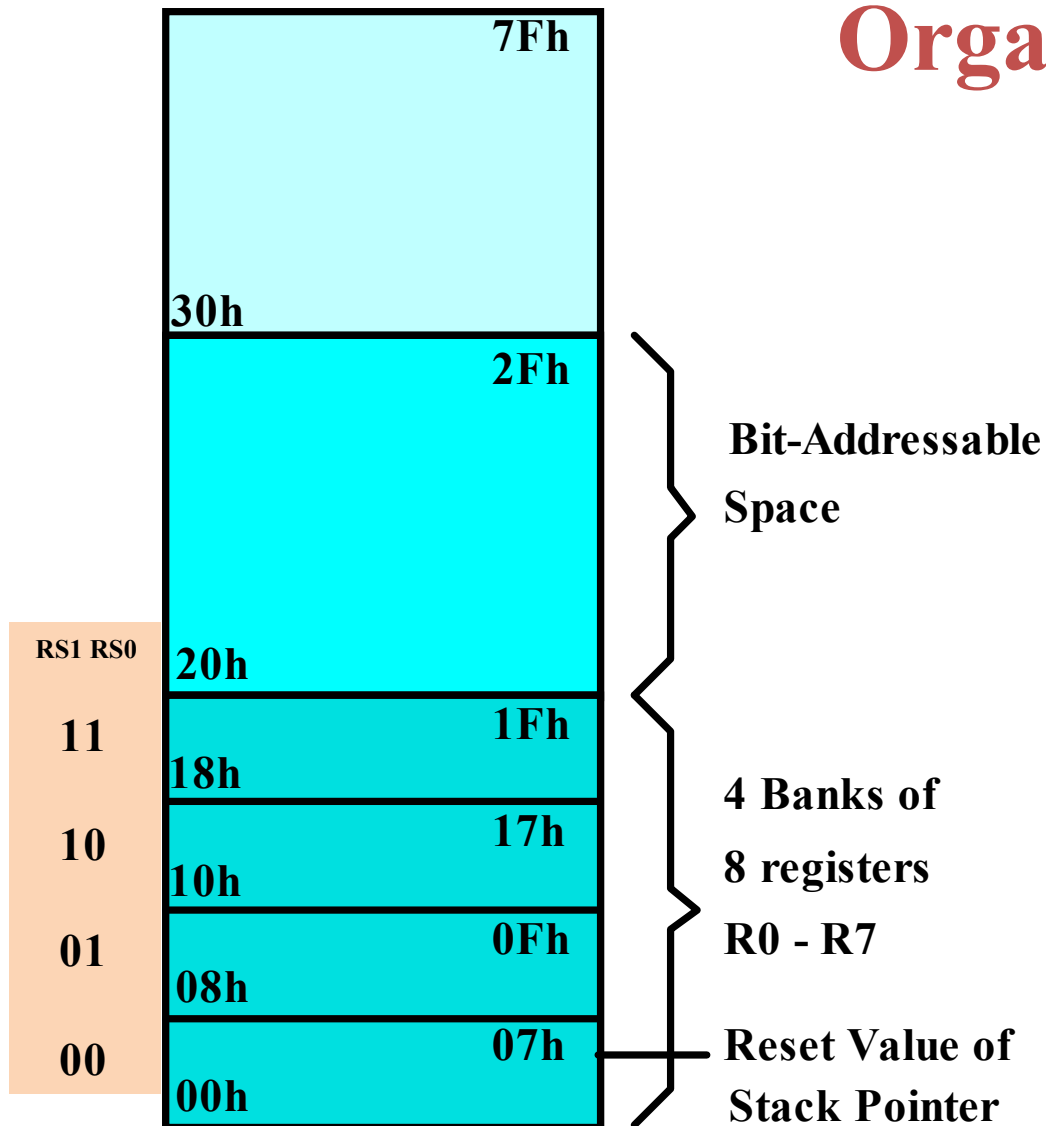
```
        SETB    RS0
        MOV     R0,#8
        MOV     R1,#0
Loop:   RL      A
        JNC     Salta
        INC     R1
Salta :  DJNZ    R0,Loop
        MOV     P1,R1
        CLR     RS0
        RET
```

# Organização da RAM Interna

7F	7E	7D	7C	7B	7A	79	78	2FH
...								...
0F	0E	0D	0C	0B	0A	09	08	21H
07	06	05	04	03	02	01	00	20H
R0 - R7 Banco 3								1FH 18H
R0 - R7 Banco 2								17H 10H
R0 - R7 Banco 1								0FH 08H
R0 - R7 Banco 0								07H 00H

Low 128 bytes of Internal RAM

# Organização da RAM Interna



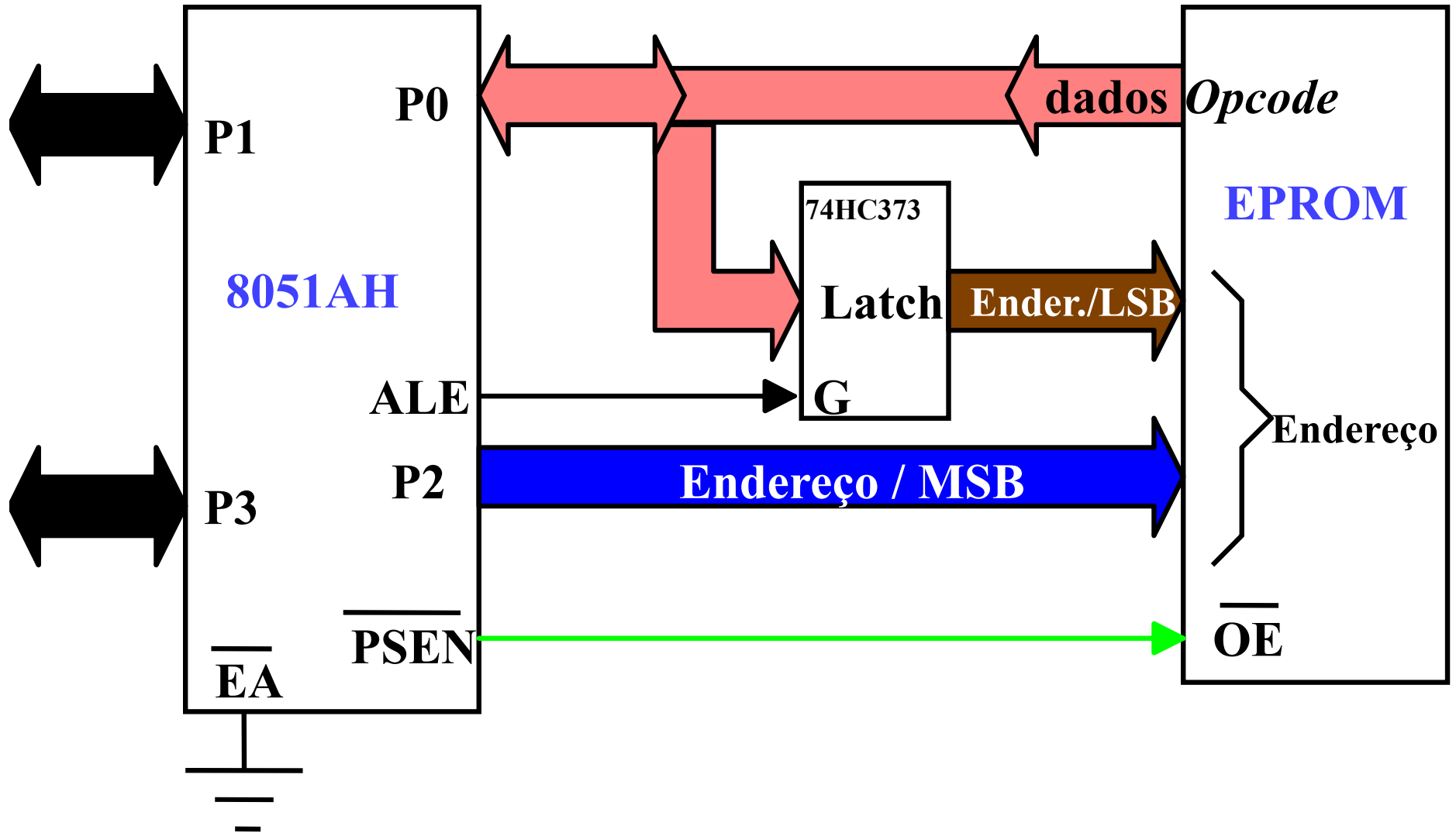
Accessible by Direct and Indirect Addressing



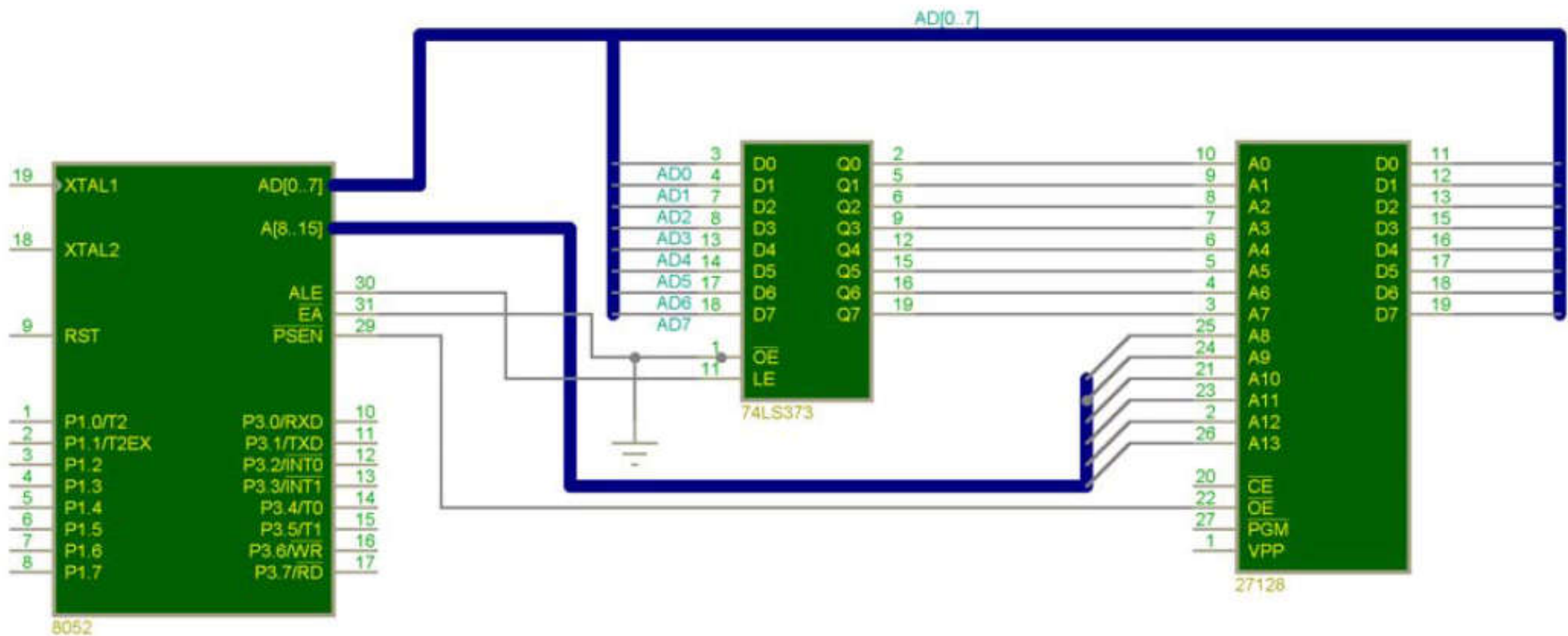
## Instruções para Manipulação de Bits

Mnemônico	Descrição
CLR C	Reseta flag CY
CLR bit	Reseta o bit endereçado
SETB C	Seta o flag CY
SETB bit	Seta o bit endereçado
CPL C	Complementa o flag CY
CPL bit	Complementa o bit endereçado
ANL C,bit	AND entre o bit endereçado e o flag CY
ANL C,/bit	AND entre o complemento do bit endereçado e o flag CY
ORL C,bit	OR entre o bit endereçado e o flag CY
ORL C,/bit	OR entre o complemento do bit endereçado e o flag CY
MOV C,bit	Copia bit endereçado para CY
MOV bit,C	Copia CY para bit endereçado

# Conexão do 8051 com Memória de Programa Externa

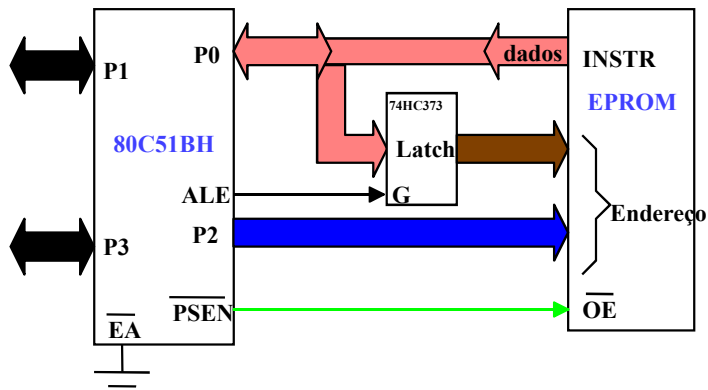


# Conexão do 8051 com Memória de Programa Externa

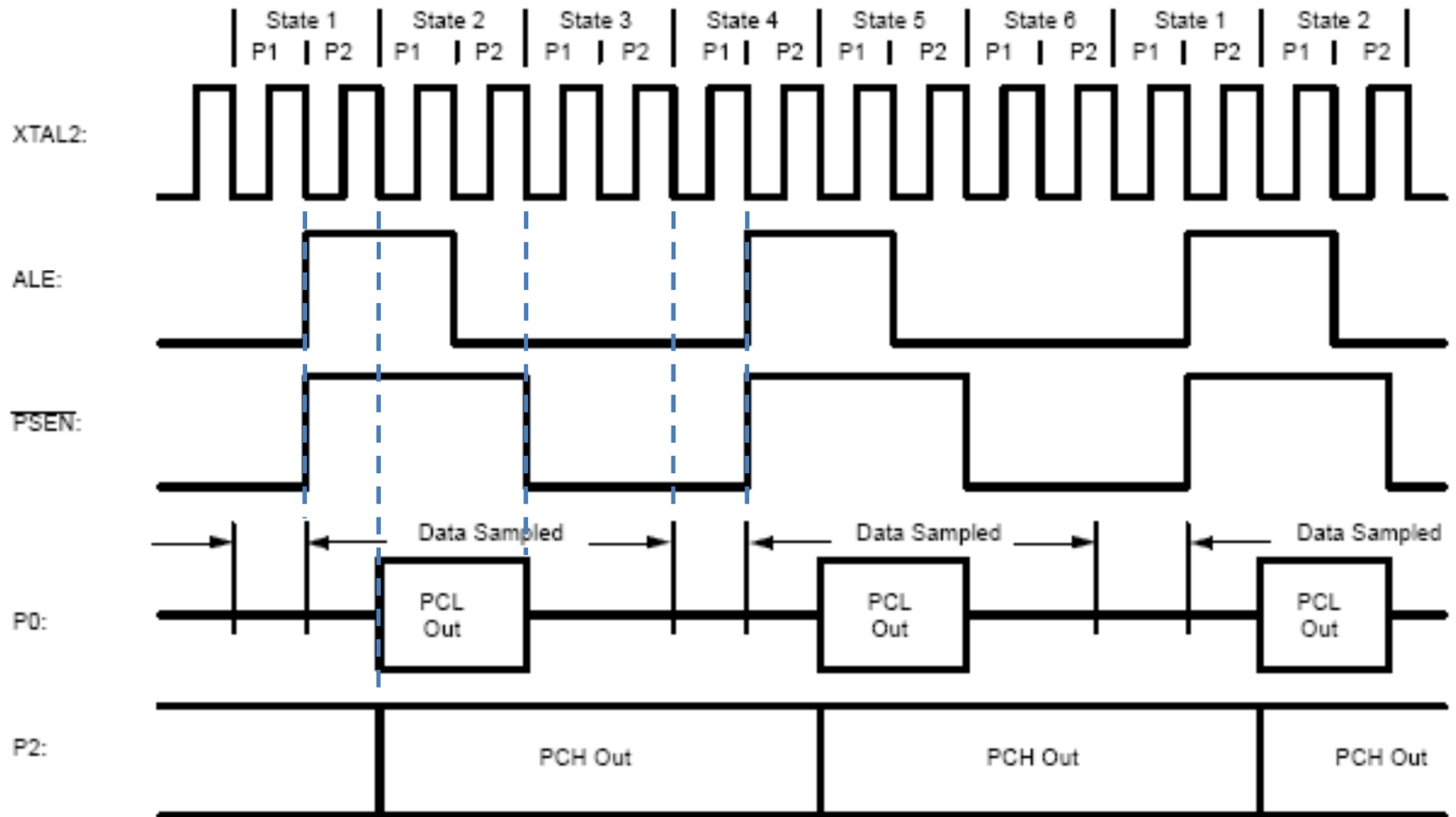


Memória EPROM de  $2^{14} = 16$  KiB





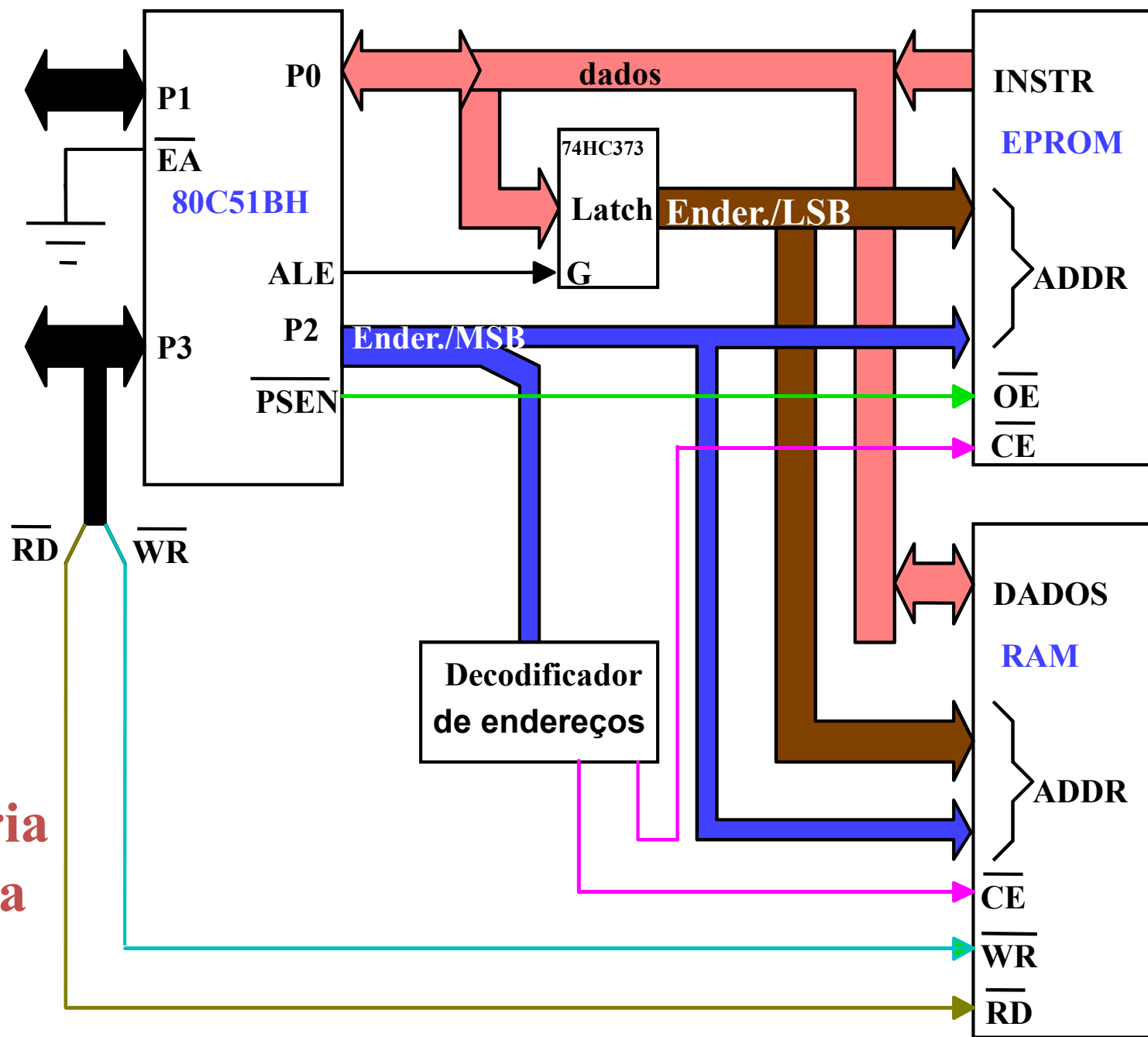
## Exemplo de Leitura da Memória de Programa Externa



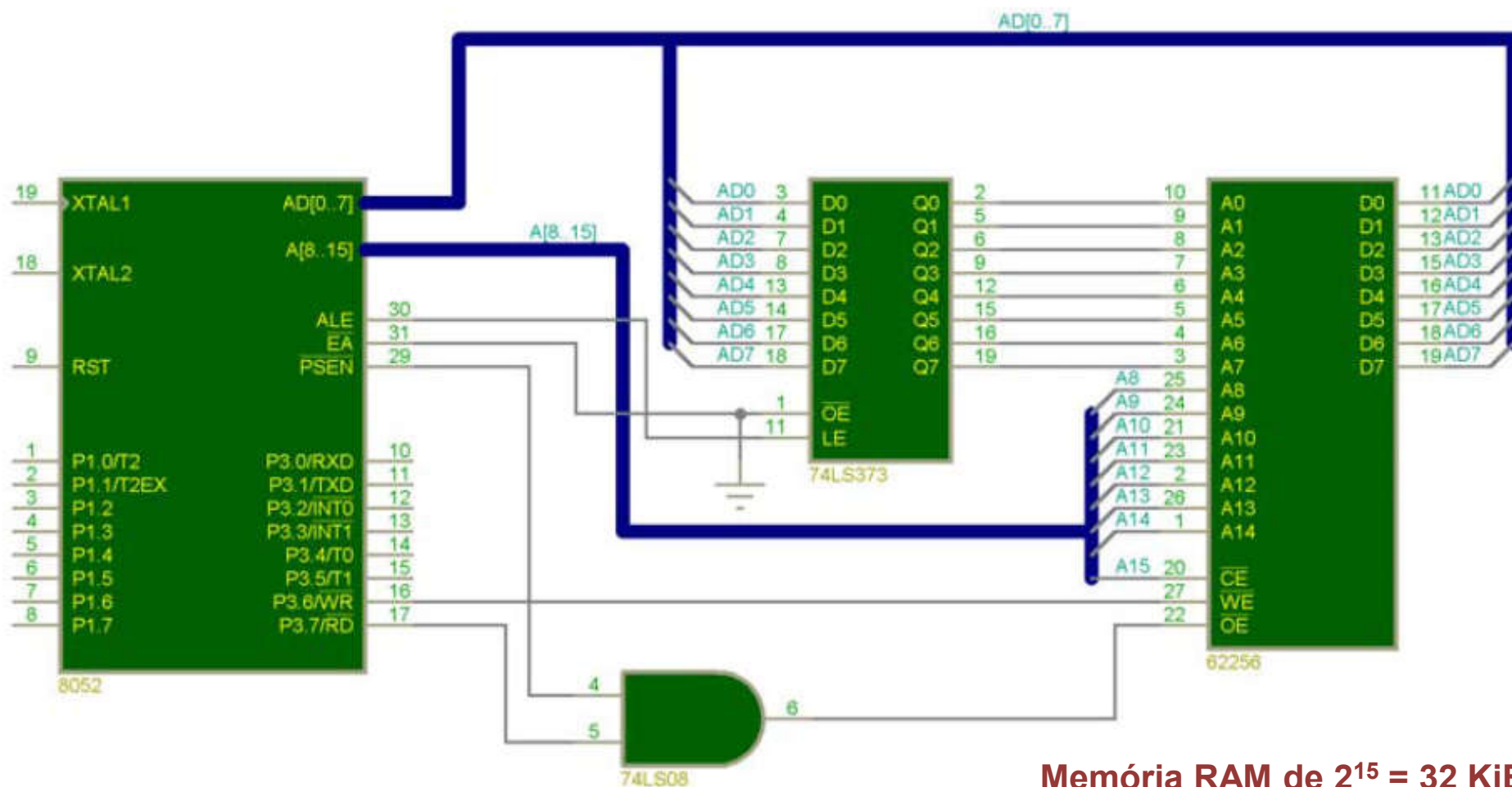
## **Passos para execução de instrução de 1 byte**

- 1) No reset, endereço inicial (0000H) contido no registrador PC é colocado no barramento de endereços (AD0-AD7/P0) & (A8-A15/P2).**
- 2) Unidade de Controle (UC) coloca ALE (*Address Latch Enable*) em '1' durante 2 ciclos de clock. Colocado em '0' no restante do ciclo de leitura. Utilizado para armazenar endereço em AD0-AD7 no latch.**
- 3) Unidade de Controle coloca pino PSEN em '0' .**
- 4) Memória coloca dado no barramento de dados (AD0-AD7)**
- 5) Valor em (AD0-AD7) é transportado para decodificador de instrução (*Instruction Register*). Controlado por PSEN. PC é incrementado.**
- 6) Após decodificar a instrução, UC emite sinais de controle para executar tarefa demandada.**

# Conexão do 8051 com Memória de Programa e de Dados Externas



# Conexão do 8051com Memória de Dados Externa



Memória RAM de  $2^{15} = 32$  KiB



## Escrita na Memória de Dados Externa (MOVX @DPTR,A; MOV P2,#\_\_H, MOVX @Ri,A)

