

EEL7030 - Microprocessadores

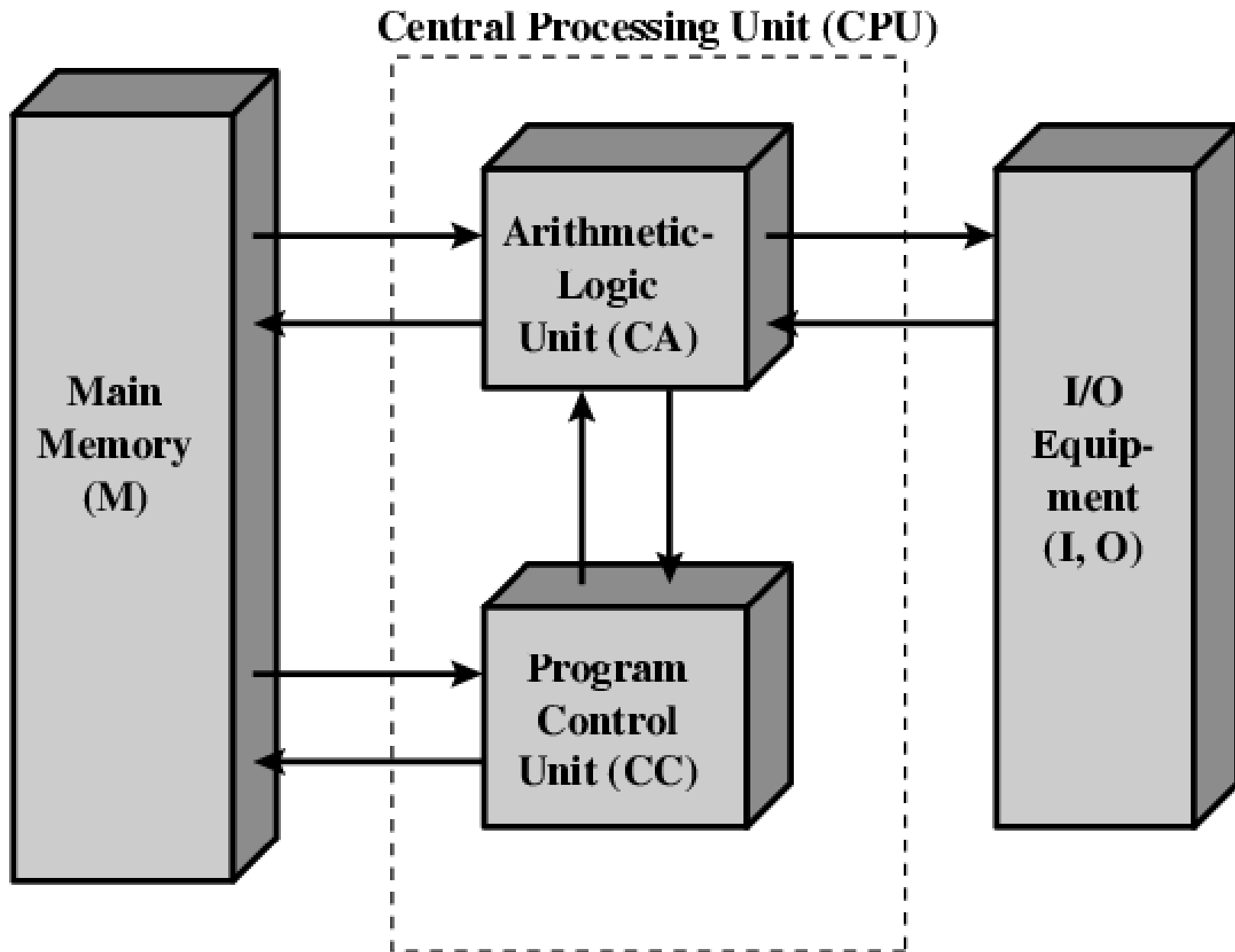


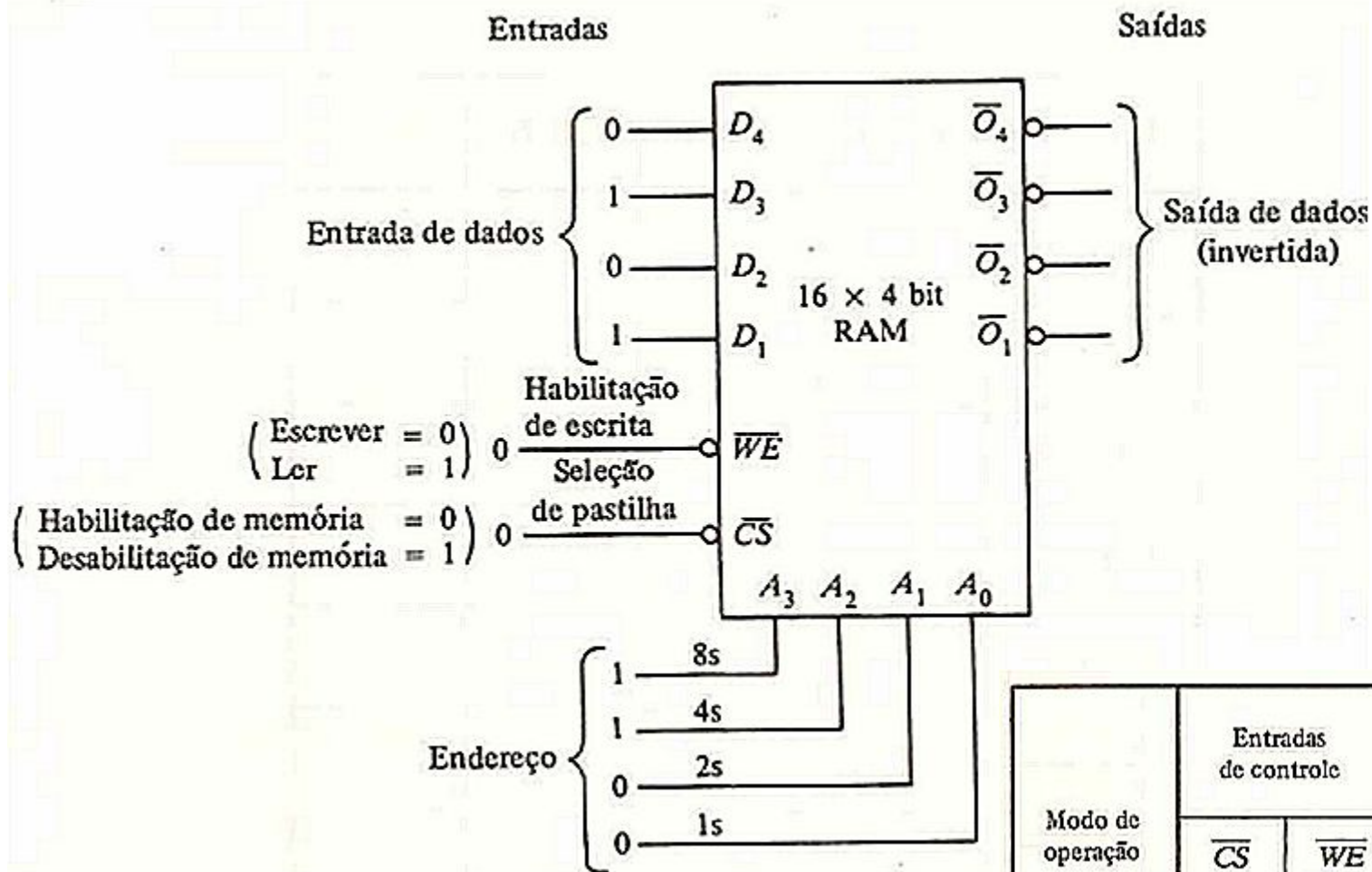
Prof. Raimes Moraes

GpqCom – EEL

UFSC

Arquitetura Von Neuman de Computadores



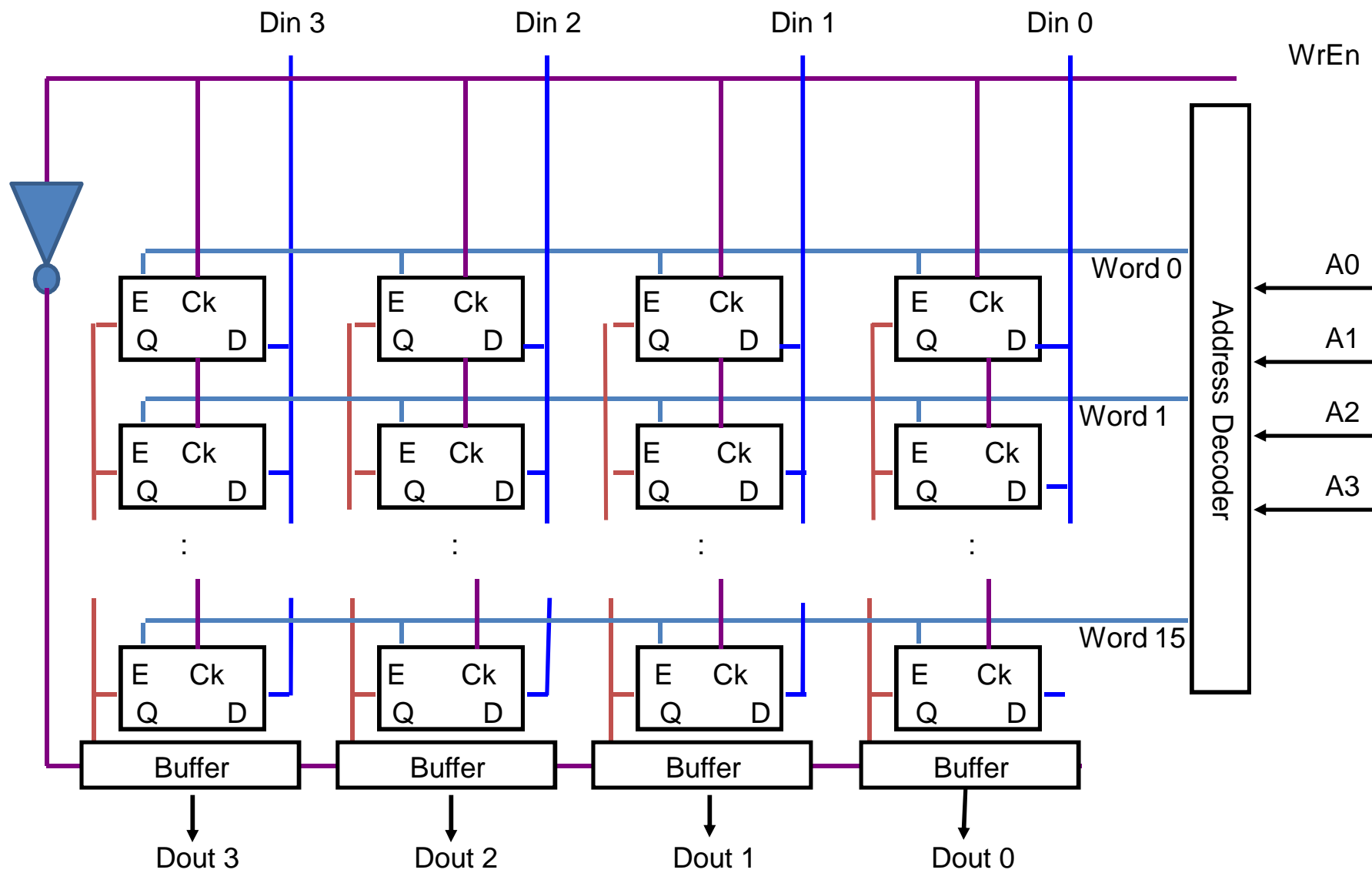


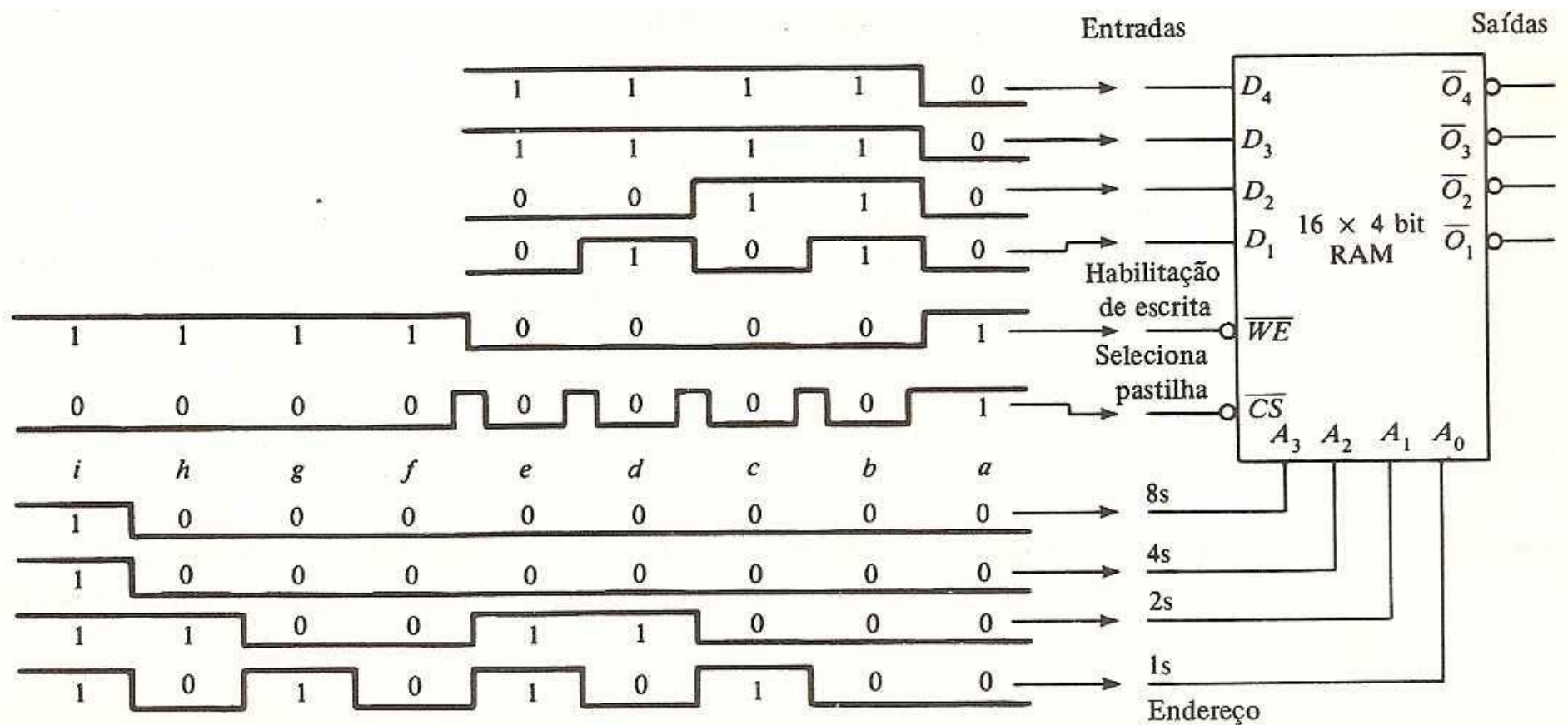
Exemplo de RAM 16x4 74189

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

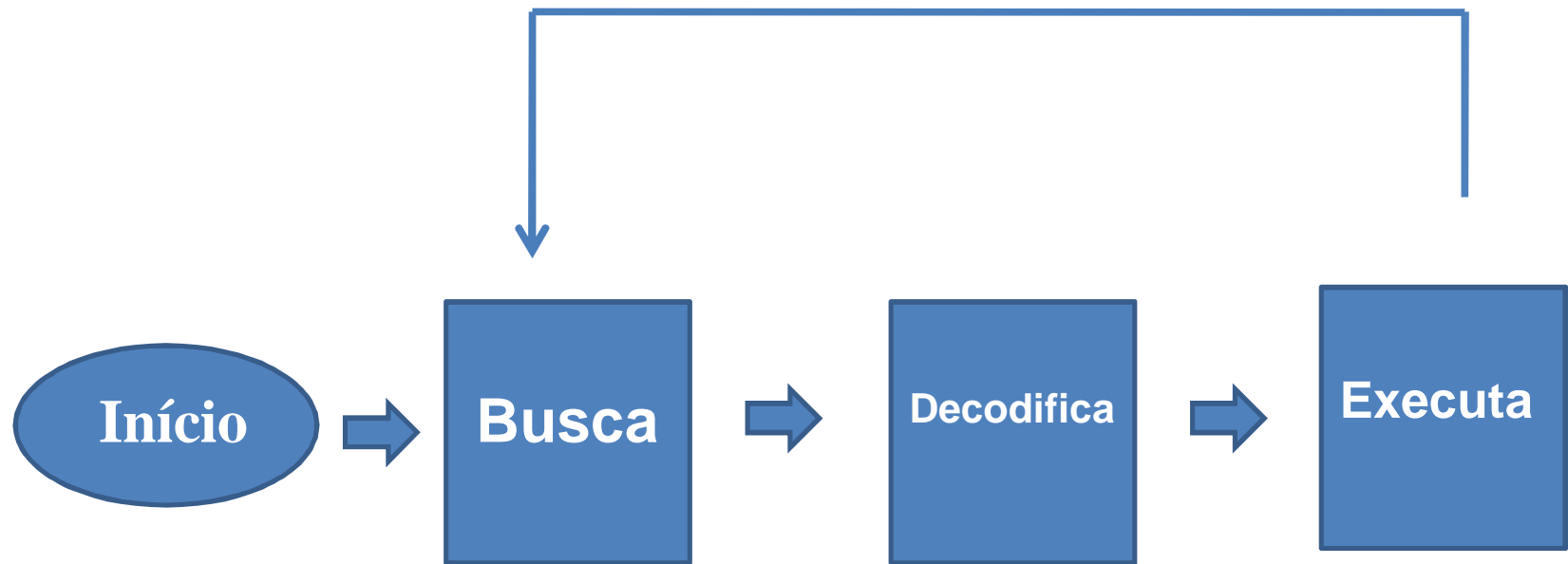
(c) Tabela-verdade de modos para a RAM de 64 bits





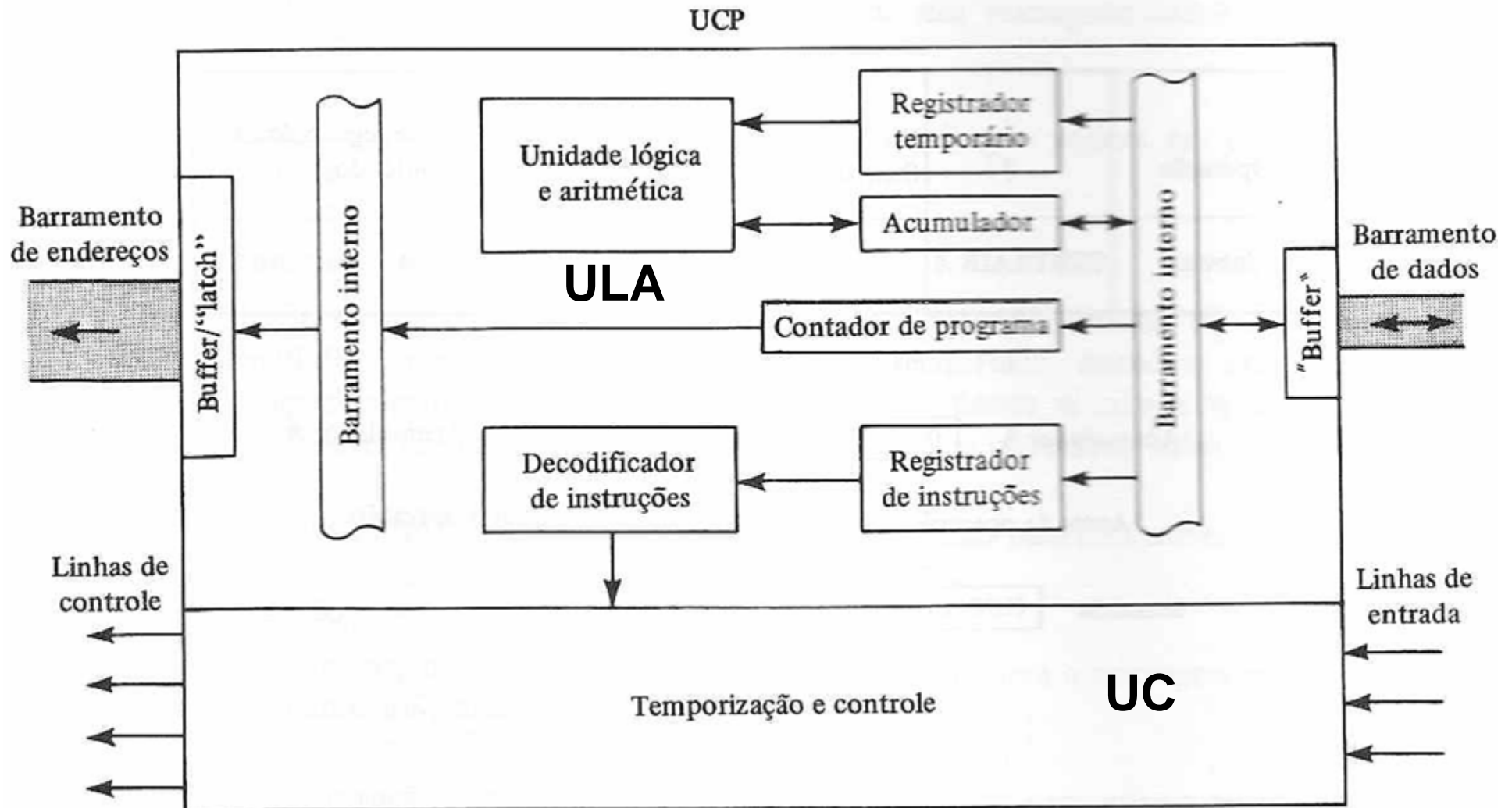
Exemplo de Acesso a RAM 16x4
 (Memórias geralmente armazenam palavras de oito bits)

Funções Básicas da UCP (Unidade Central de Processamento)



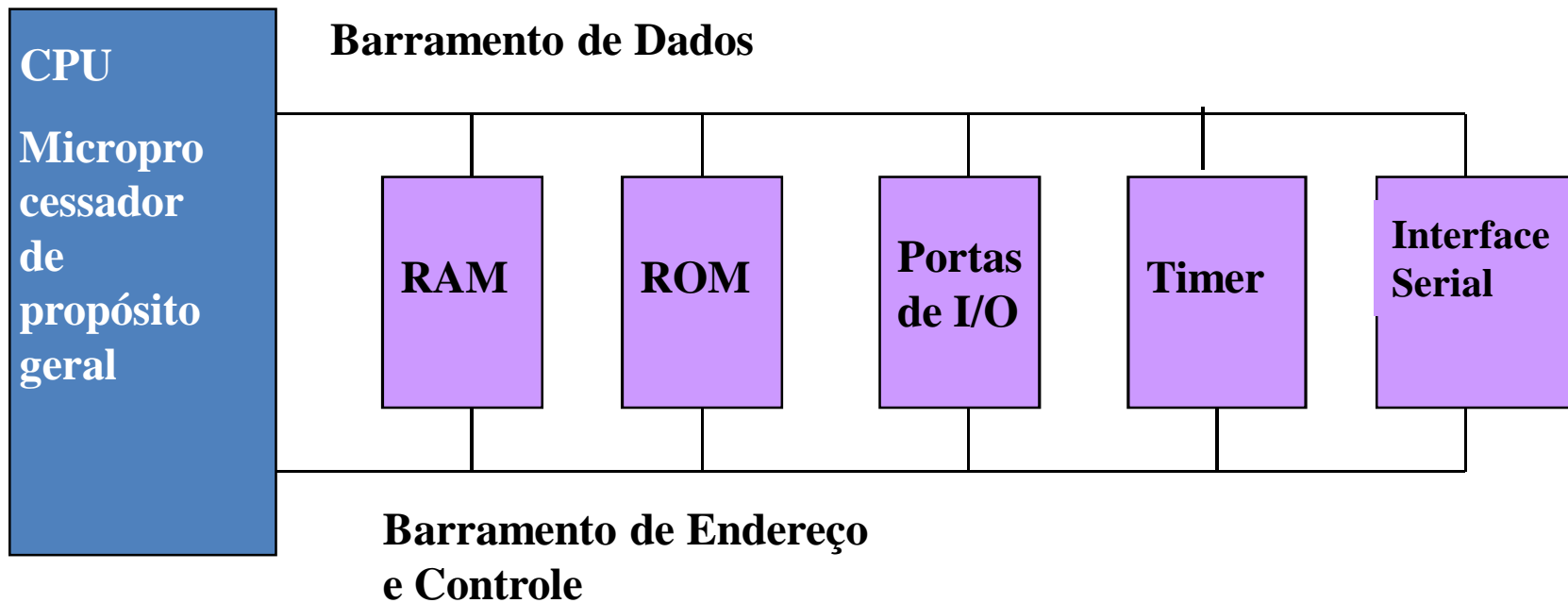
- **Busca, decodifica e executa instruções;**
- **Supre sinais de controle para memória e outros periféricos;**
- **Transfere dados da e para memória e periféricos;**
- **Atende demanda dos periféricos (interrupções).**

Arquitetura simplificada de uma UCP



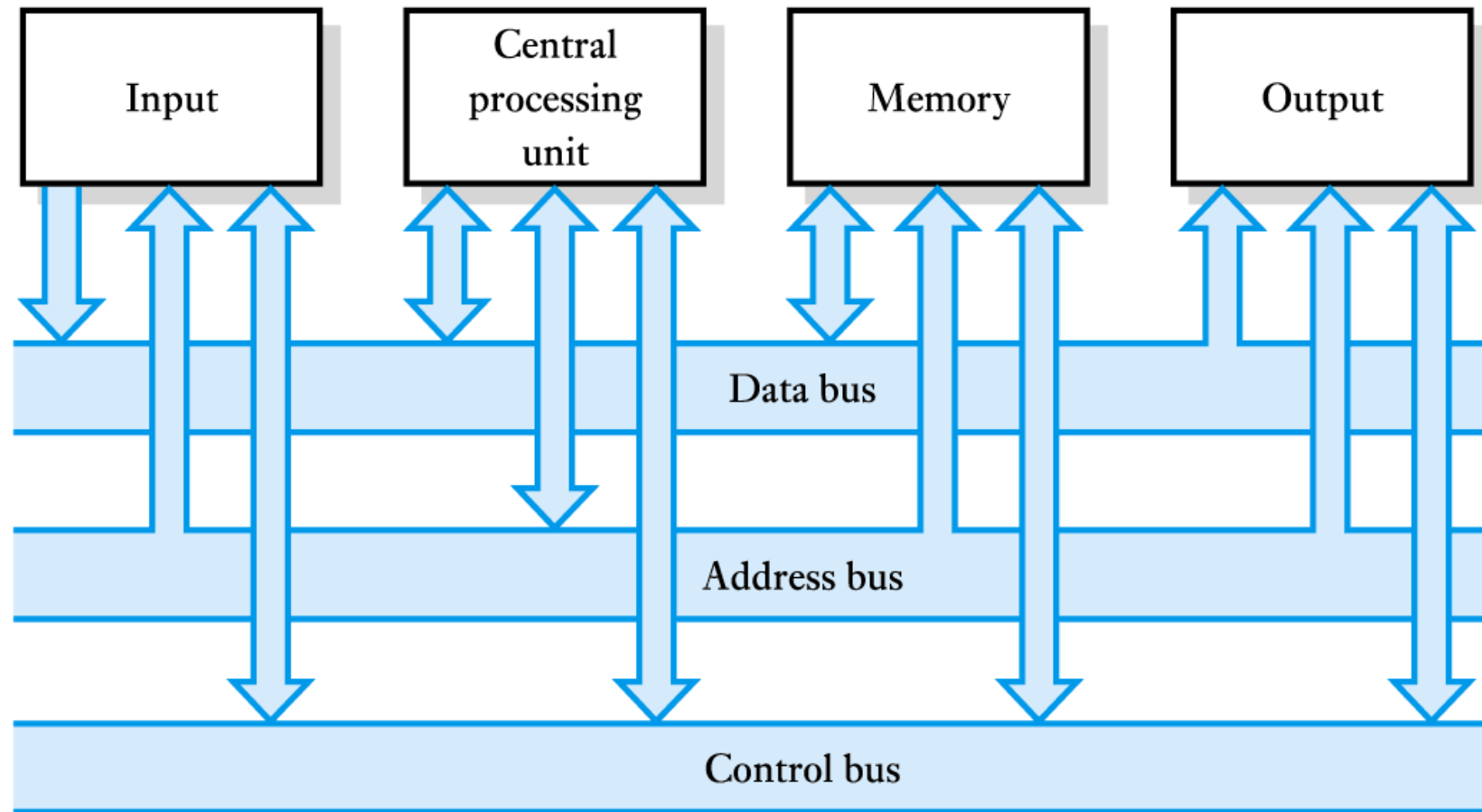
Microprocessador de Propósito Geral

Sem RAM, ROM ou dispositivos de I/O



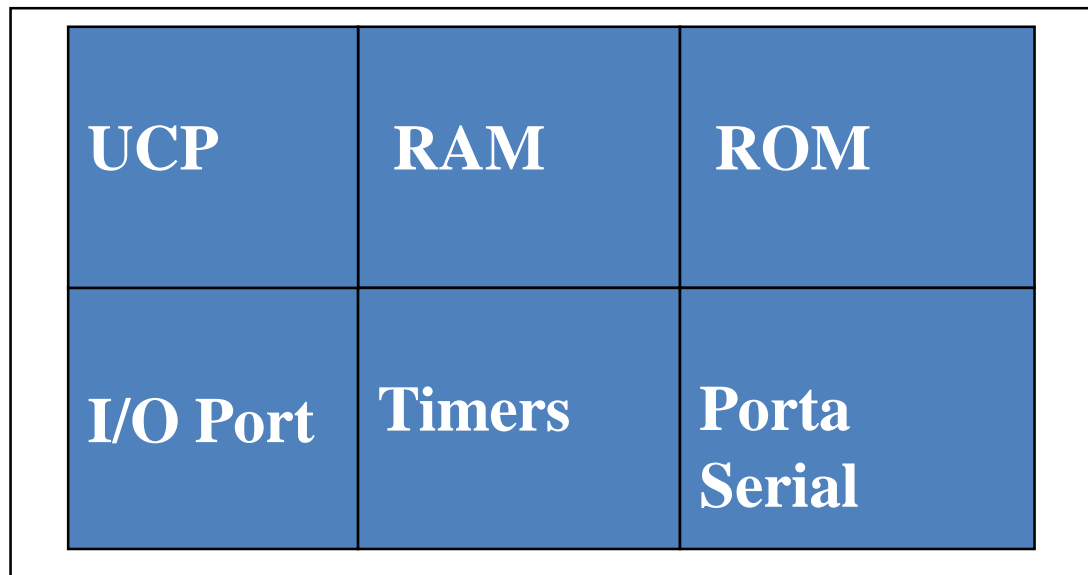
Vantagem: flexibilidade, sistema expansível ;

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



Microcontrolador

▪UCP + “Periféricos”



Vantagem: menor custo, menor dimensão, rápido desenvolvimento;

Desvantagem: baixa flexibilidade, não expansível;

Microcontrolador

- **Módulos:**
 - **Portas de entrada e saída**
 - **Interface Serial (CAN, SPI, USB, RF e etc)**
 - **Memórias (ROM, RAM, Flash, XRAM, etc)**
 - **Conversores Analógico/Digital e Digital/Analógico**
 - **Timers**
 - **...**

- **Família de Microcontrolador: CPU + diferentes módulos**

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:**
 - **assemblers, debuggers, compiladores, emuladores, simuladores, suporte técnico**

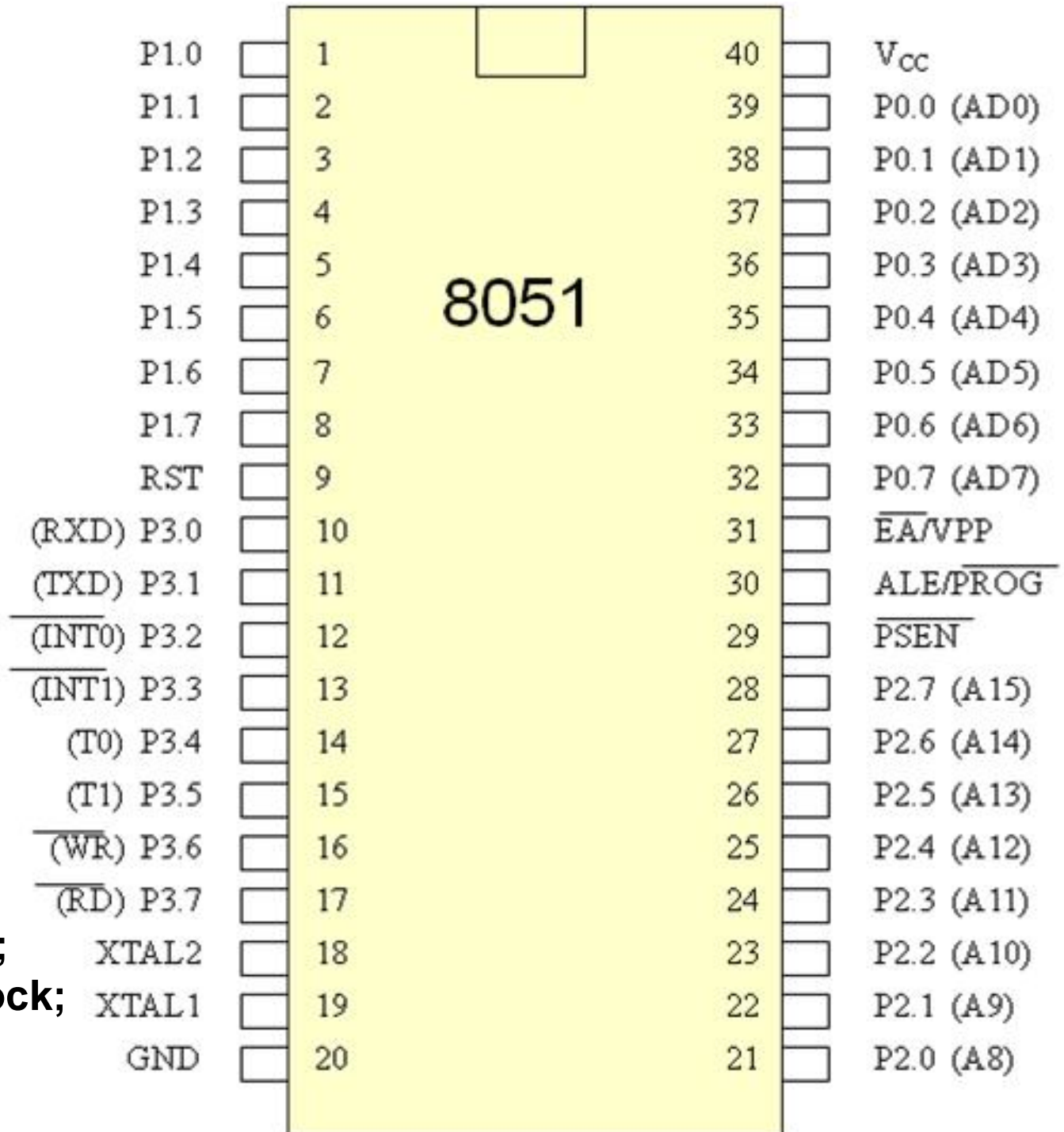
8051

- 1980
- CPU de 8 bits
- Clock até 12 MHz
- Vcc=+5V; 40 pinos;
- Endereça até 64 KiB
(kibibyte = 1024 bytes):

1. de dados;
2. de programa;

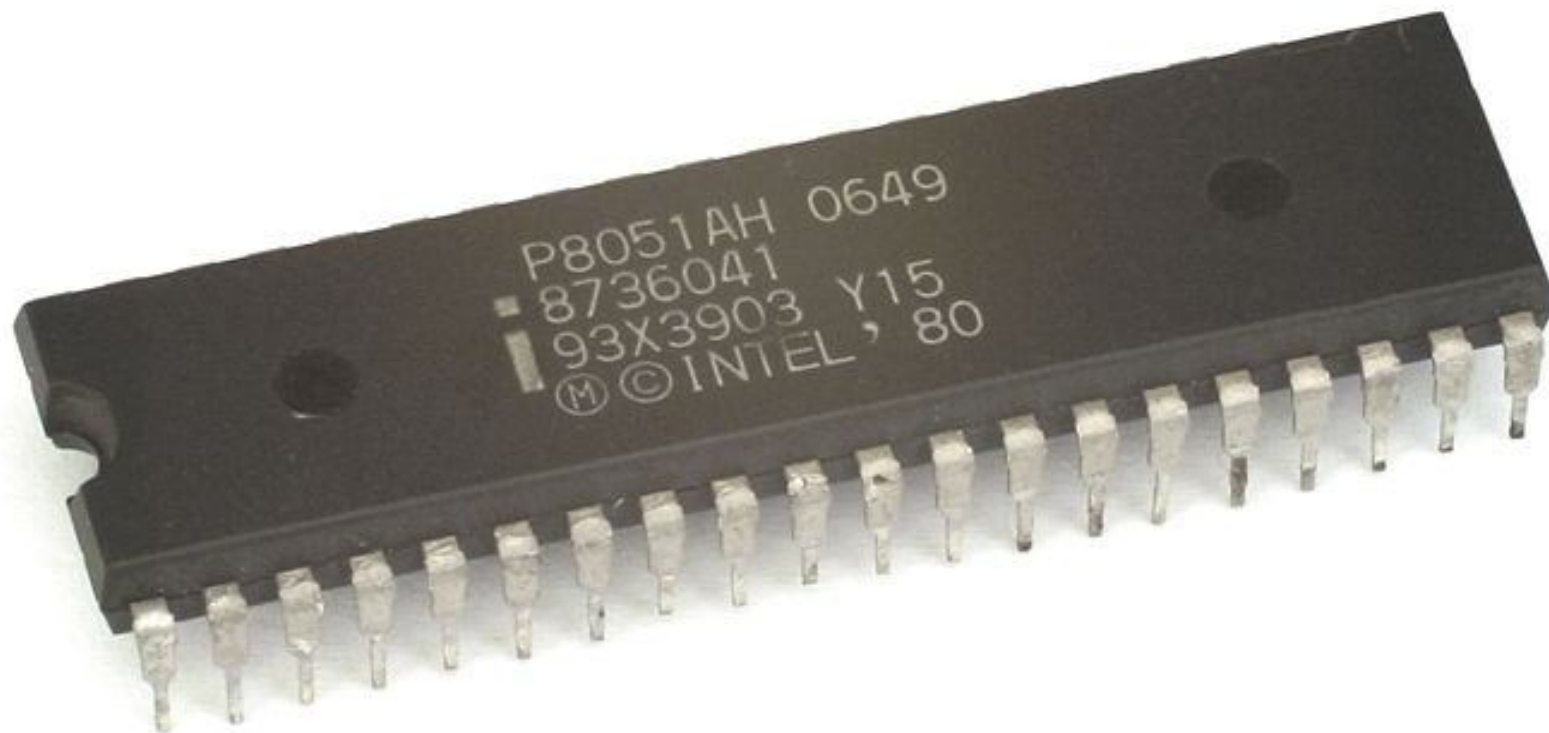
- Possui pinos:

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



8051

Tipo de encapsulamento do 8051



8051

▪ Características Básicas:

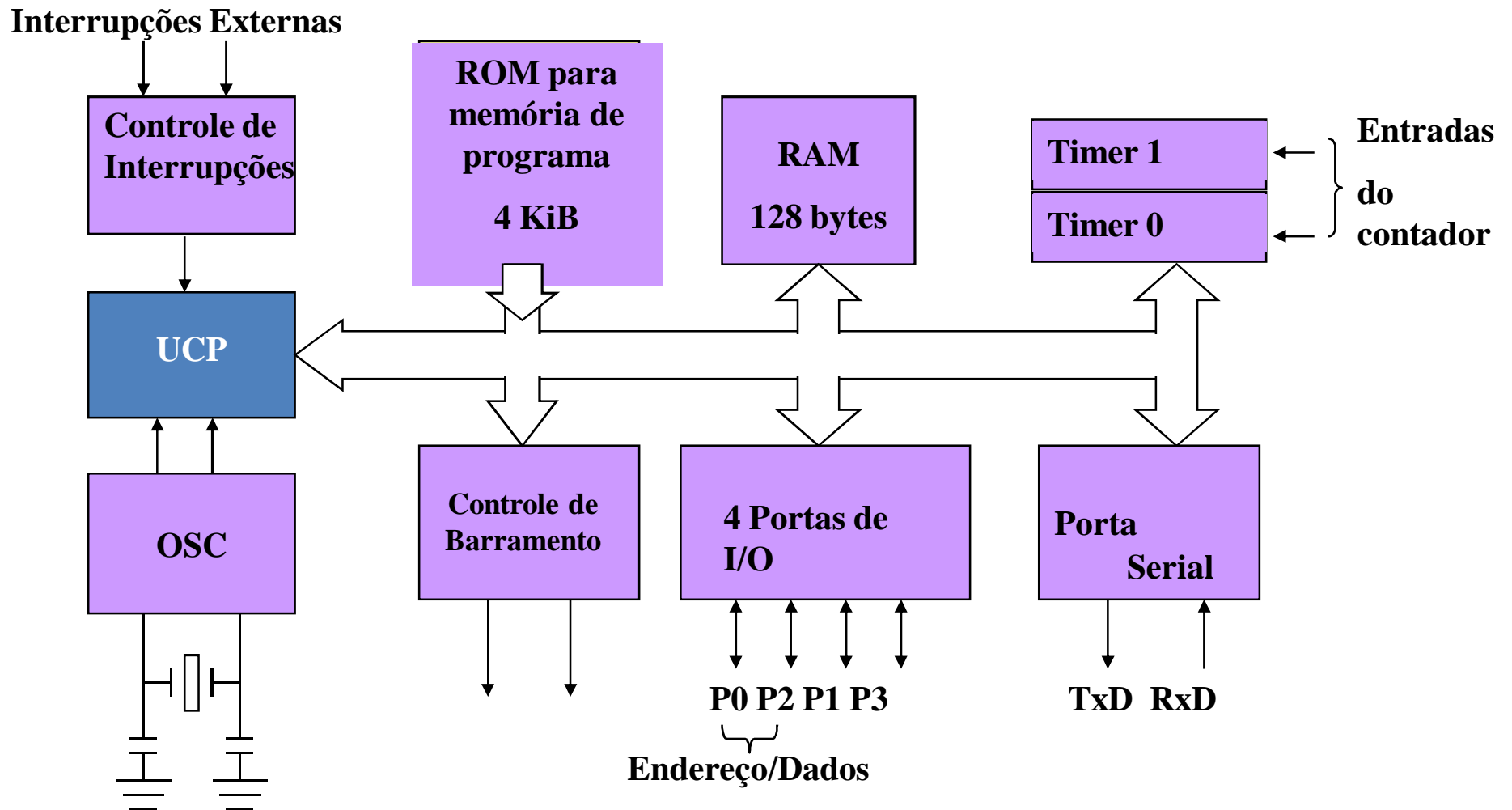
- ❑ CPU de 8 bits;
- ❑ endereça 64 KiB de memória de programa externa;
- ❑ endereça 64 KiB de memória de dados externa;

- ❑ 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

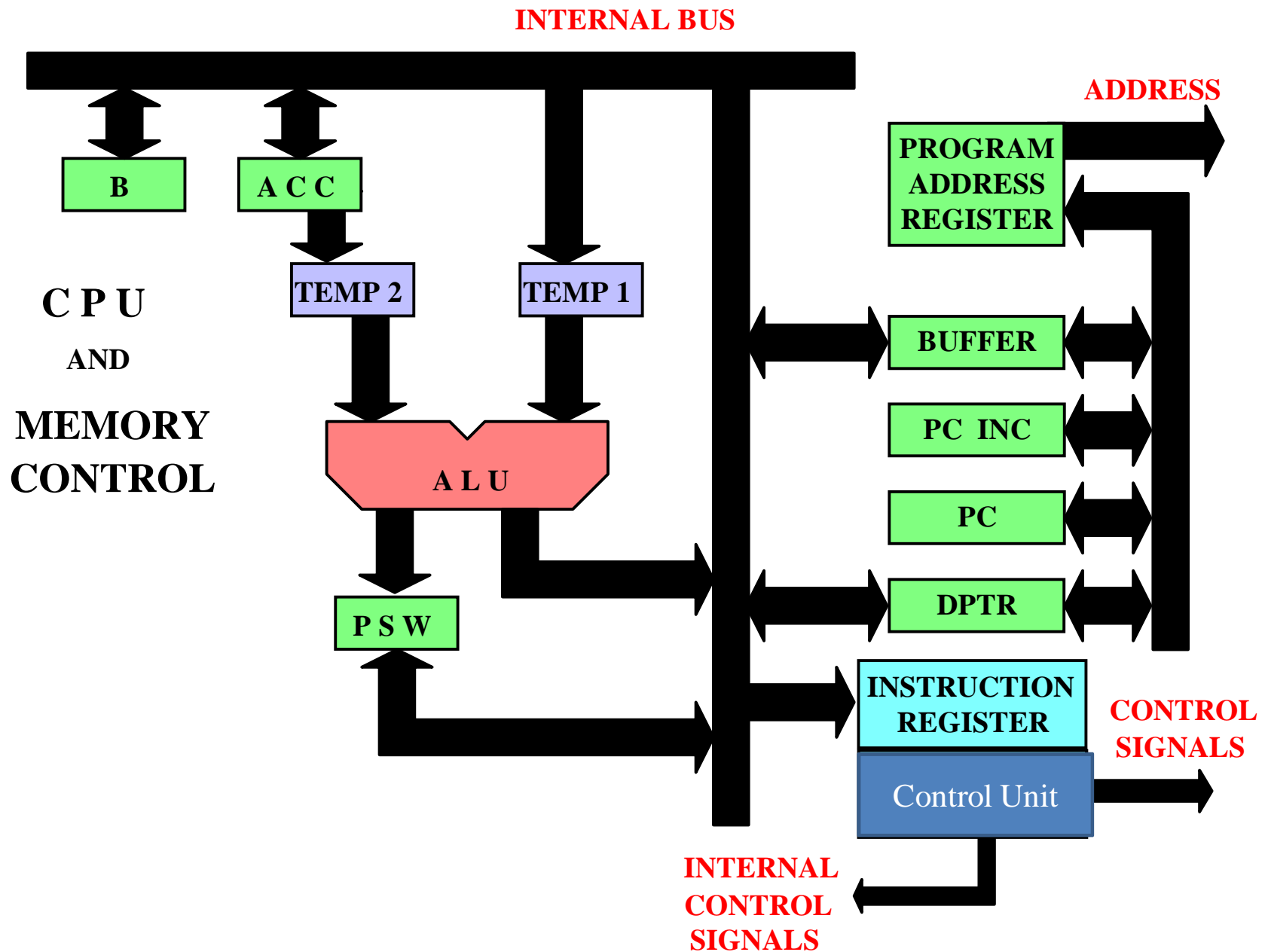
Alguns uC da família 8051 da Atmel

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	--

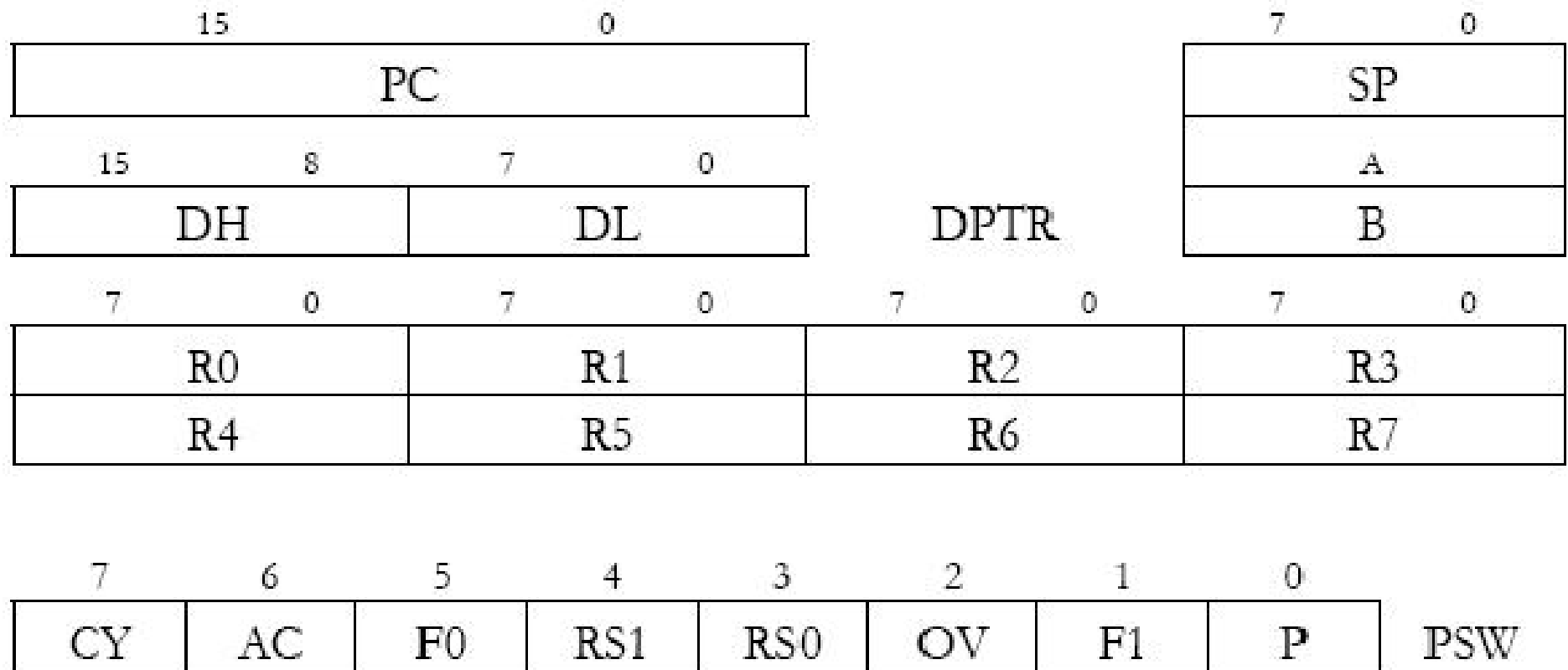
8051



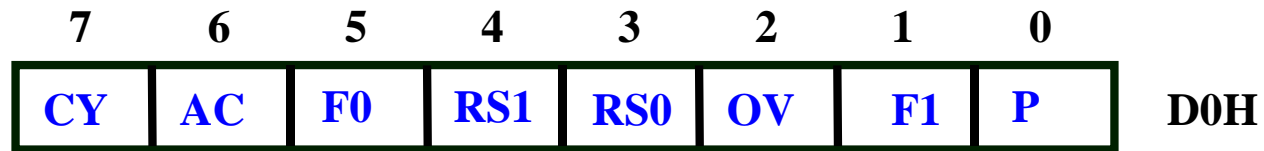
Arquitetura simplificada da UCP do 8051



Registradores do 8051

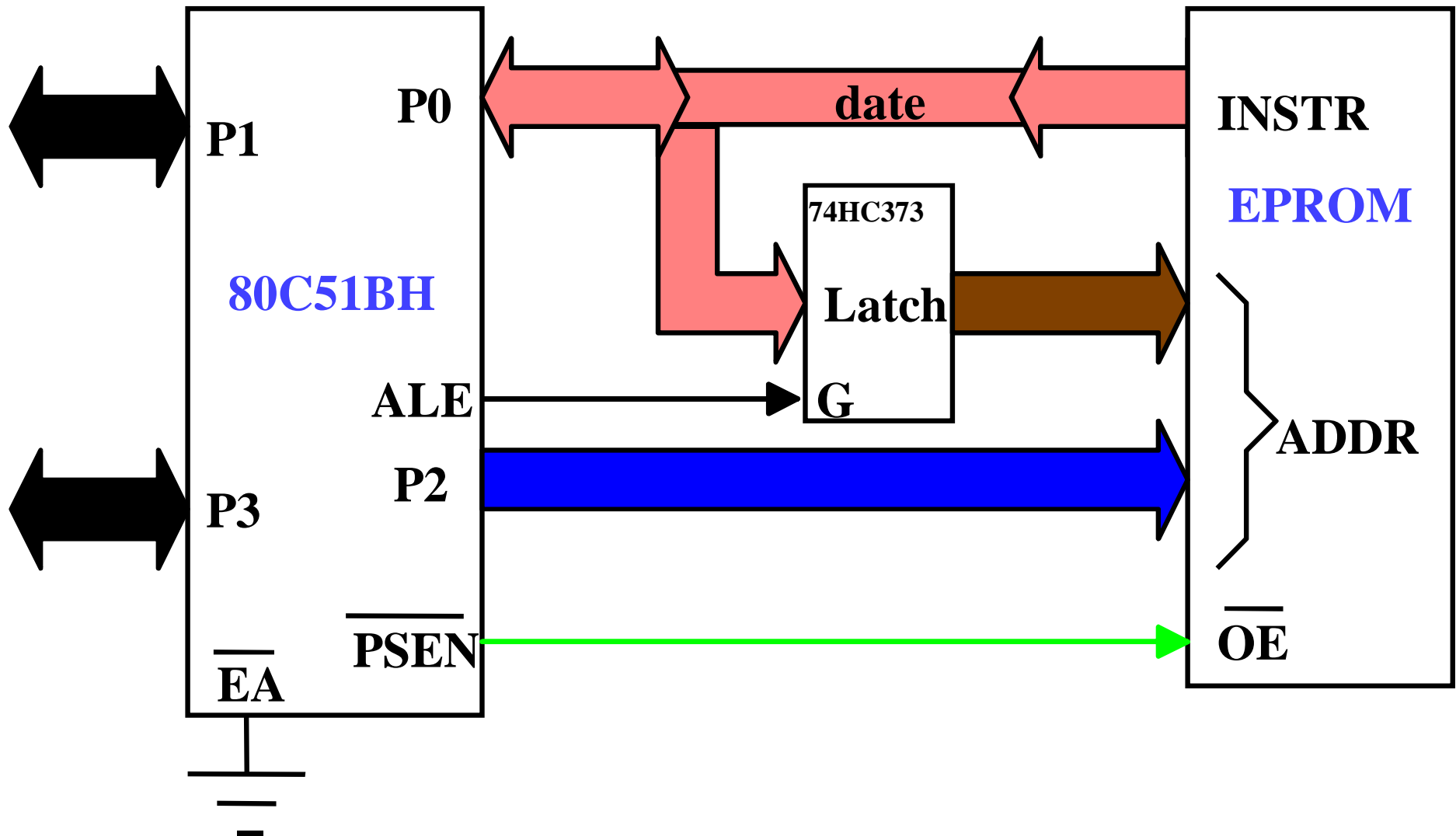


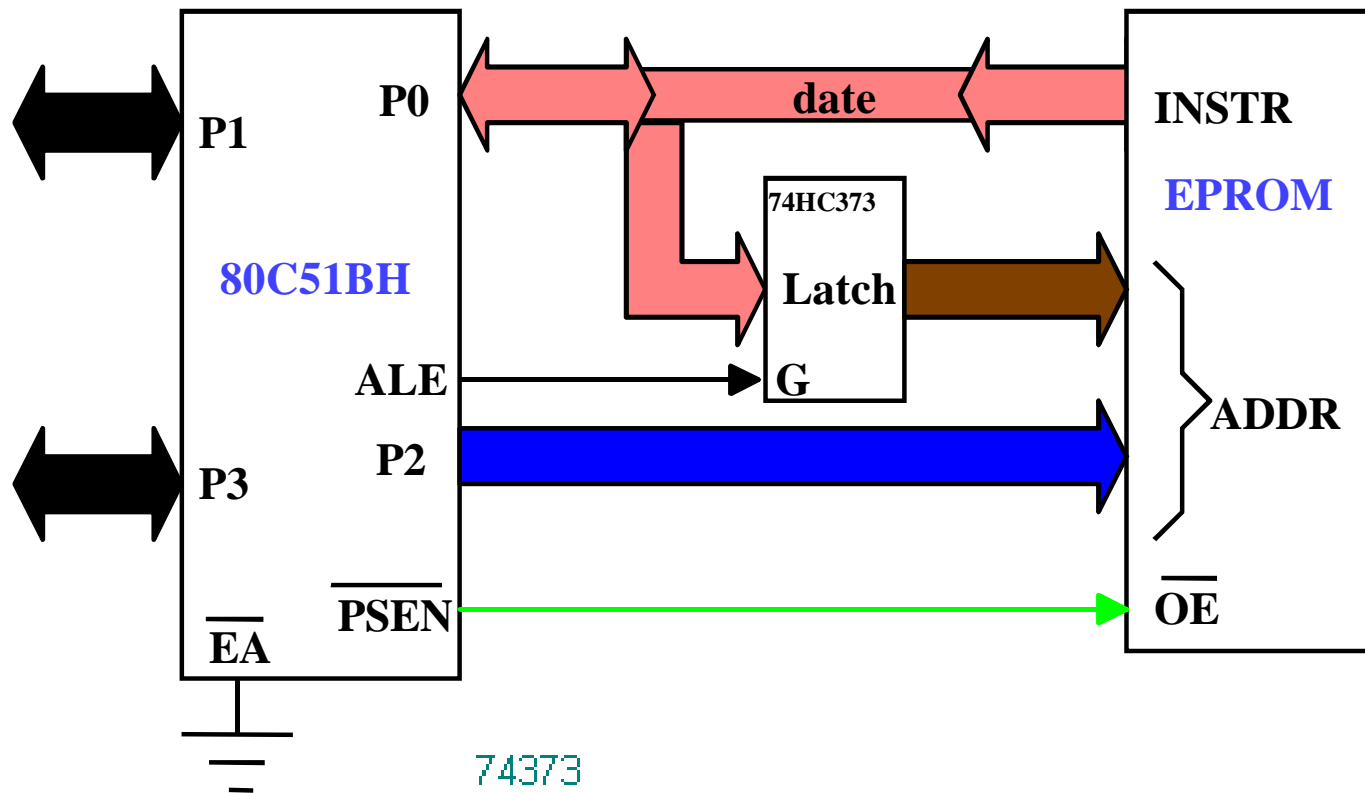
PSW - Program Status Word - Bit Addressable



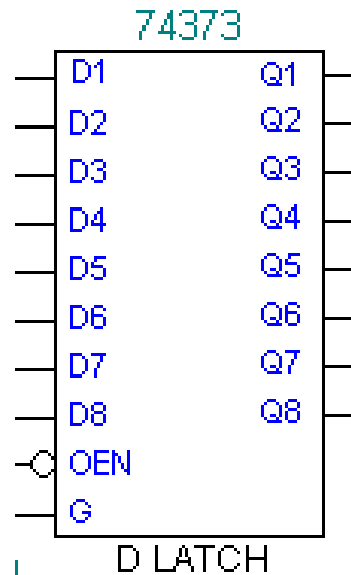
Nome	Localização	Descrição
CY	PSW.7	Carry flag
AC	PSW.6	Auxiliary carry flag
F0	PSW.5	Definido pelo usuário
RS1	PSW.4	Bit 1 do seletor de Register Bank
RS0	PSW.3	Bit 0 do seletor de Register Bank
OV	PSW.2	Overflow flag
F1	PSW.1	Definido pelo usuário
P	PSW.0	Flag de paridade. 1 = ímpar.

Conexão do 8051 com Memória de Programa Externa

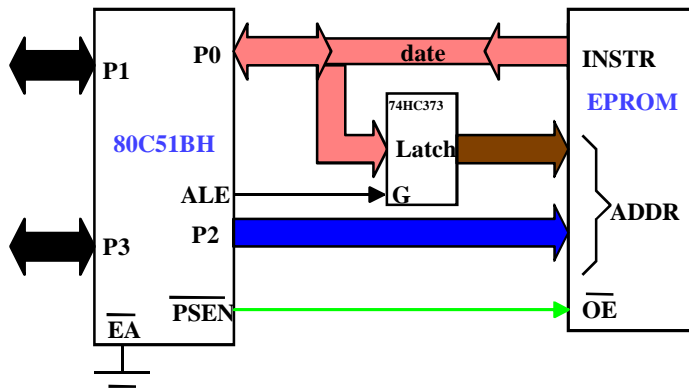




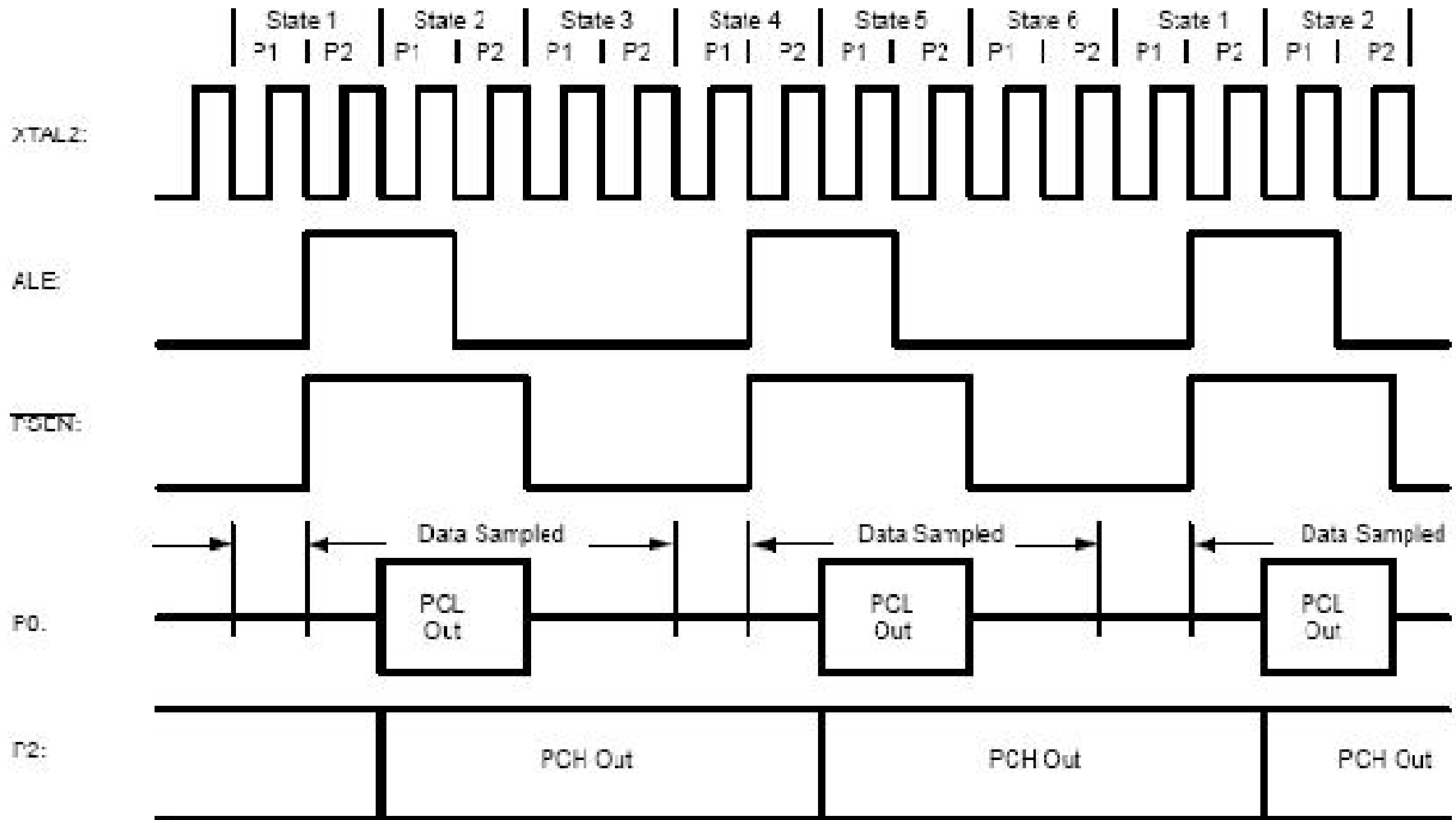
**Interface do
8051
Com
Memória de Programa
Externa**



Inputs			Outputs	
OEN	G	D	Q	
H	X	X	Z	
L	X	X	X	
L	H	L	L	
L	H	H	H	
L	L	X	Q ₀	

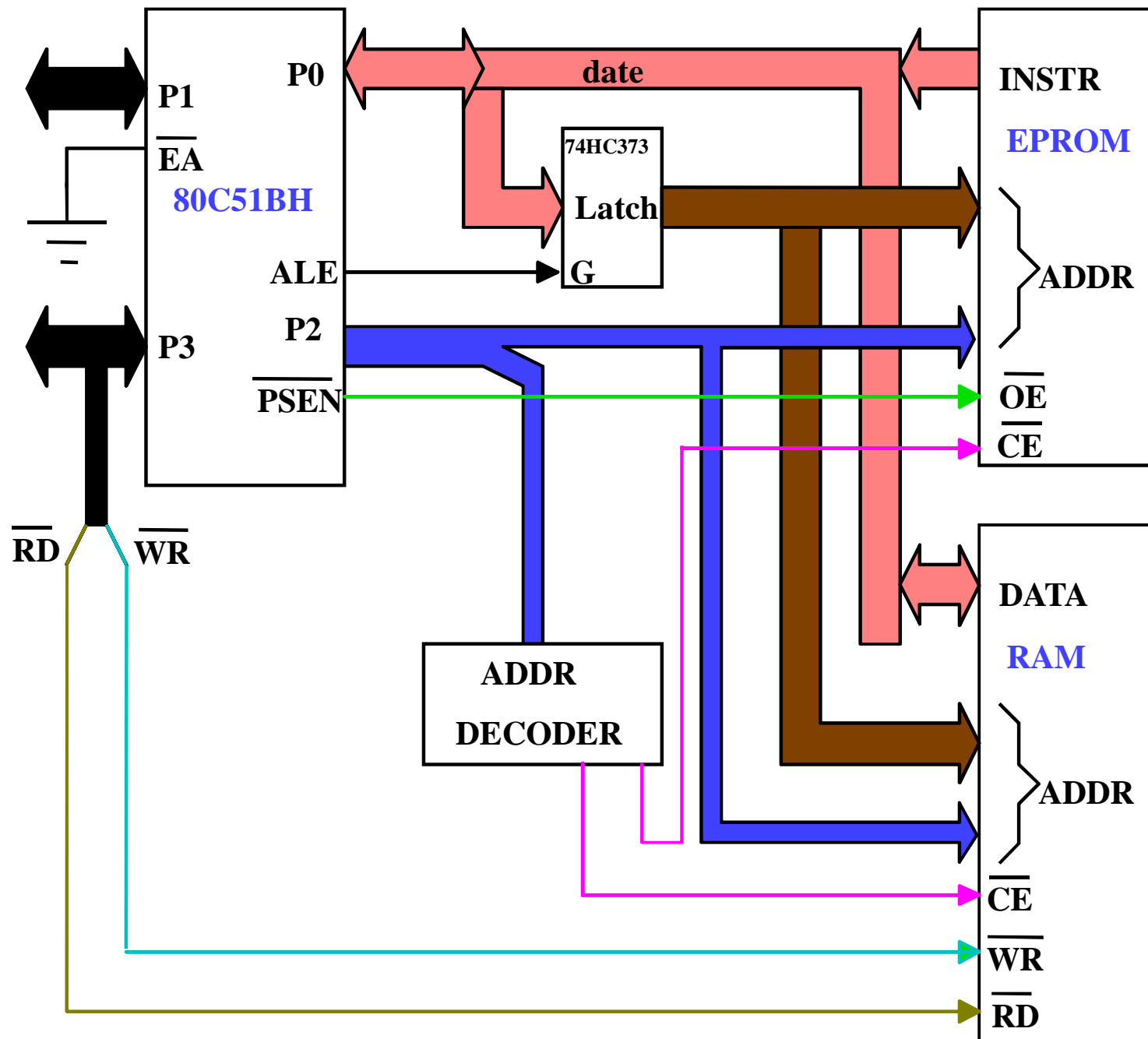


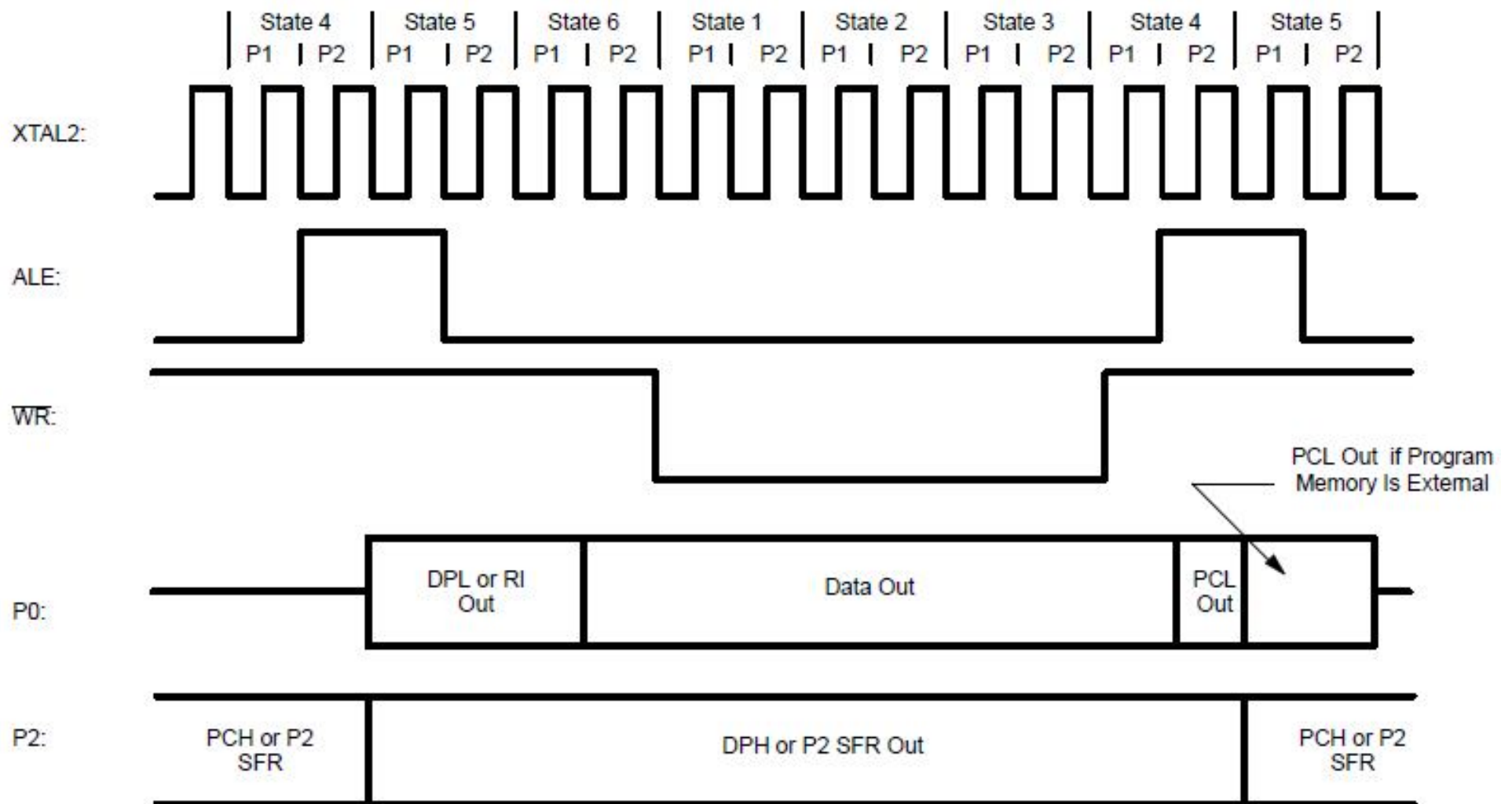
Exemplo de Leitura da Memória de Programa Externa



Passos para execução de instrução

- 1) No reset, endereço inicial (0000H) contido no registrador PC é colocado no barramento de endereços (AD0-AD7/P0) & (A8_A15/P2). PC é incrementado.**
- 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 salvar 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.**
- 6) Após decodificar a instrução, UC emite sinais de controle para executar tarefa demandada.**





MEMÓRIA DE PROGRAMA

Contém códigos de operação (Opcodes)

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

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

Exemplo de Alocação de MEM

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

Rn - registrador de R0 a R7.

Instrução MOV A,Rn

5 bits Código Instrução	3 bits Operando
1 1 1 0	1 R R R

Instrução MOV A,direct (endereço)

8 bits Código Instrução	8 bits
1 1 1 0 0 1 0 1	endereço

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

Instrução ADD A,Rn

5 bits Código Instrução	3 bits Operando
0 0 1 0	1 R R R

Instrução ADD A,direct

8 bits Código Instrução	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

Flags alterados pela instrução **ADD**

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

AC (Auxiliary Carry) = 1 \Rightarrow ocorreu vai um do bit 3;

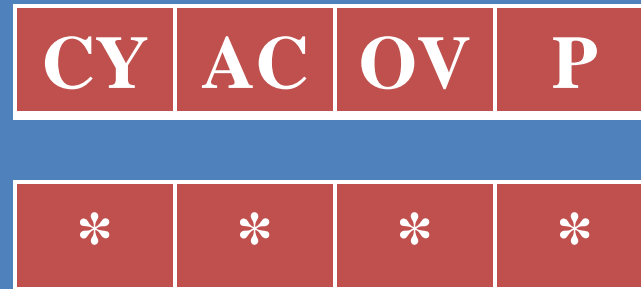
CY (Carry) = 1 \Rightarrow ocorreu vai um do bit 7;
Indica *overflow* na soma de inteiros sem sinal

OV (Overflow) = 1 \Rightarrow ocorreu vai um do bit 6, mas não do bit 7 ;
ou ocorreu vai um do bit 7, mas não do bit 6.

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)

Instrução ADD A,Rn



Soma BCD



Supondo flags atuais como



[A]=98H; [B]=08H ADD A,B



[A]=A0H; [B]=08H DA A



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

Decimal-adjust Accumulator

Overflow Flag

OV (Overflow) = 1 => ocorreu vai um do bit 6, mas não do bit 7 ou ocorreu vai um do bit 7, mas não do bit 6.

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

1) Supondo flags atuais como



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

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

ADD A,B



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

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

2) Supondo flags atuais como



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

[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

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 -- ADD A,@R0

DA A ; registrador específico

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

MOVC A,@A+DPTR ; indexado

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.

@Ri - endereçamento indireto a uma posição de memória RAM interna (Ri=R0 ou R1) ou externa (MOVB; 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: $B = N$, $soma = 0$

2: $soma = soma + B$, $B = B - 1$

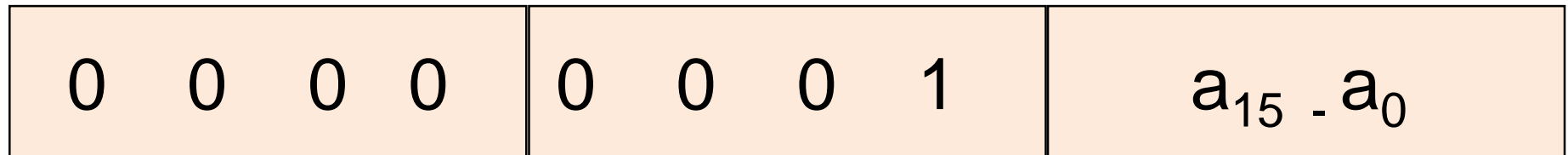
3: IF $B \neq 0$ então VÁ_PARA 2 (?)

4: $total = soma$

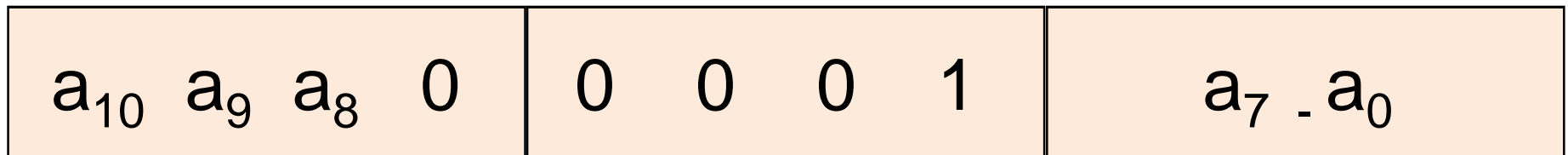
Mnemônicos para Desvio Incondicional

(2 cycles)

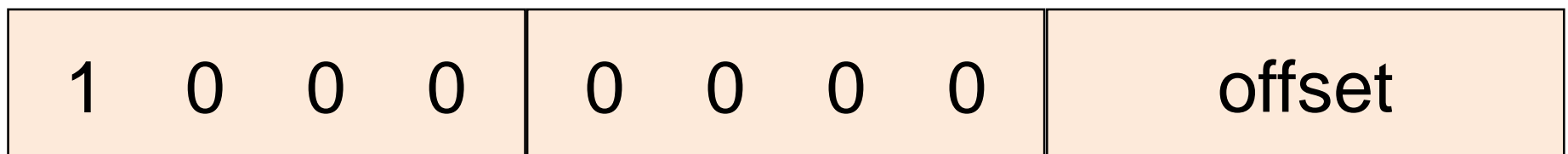
LJMP: Especifica endereço de 16 bits. A instrução possui 3 bytes (opcode + 16 bits de endereço).



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

Mnemônicos para Desvio Condicional

Mnemônico	Descrição
CJNE A, direct, <rel addr>	Compara A e memória. Salta se não igual
CJNE A, #data, <rel addr>	Compara A e dado. Salta se não igual
CJNE Rn, #data, <rel addr>	Compara Rn e dado. Salta se não igual
CJNE @Ri, #data, <rel addr>	Compara Ri e memória. Salta se não igual
DJNZ Rn, <rel addr>	Decrementa Rn e salta se não zero
DJNZ direct, <rel addr>	Decrementa memória e salta se não zero

Exercício

Total EQU 20h

MOV A,#0

MOV DPTR,#N

MOVC A,@A+DPTR

MOV B, A ; B = N

XRL A,Acc ; soma = $A \oplus A = 0$

Loop:

MOV R0,B

ADD A,R0 ; soma = soma + B

DJNZ B,LOOP ; B = B - 1; Se B ≠ 0 então VÁ_PARA Loop

MOV Total,A ; Total = soma

FIM:

JMP FIM ; Equivalente a jmp \$

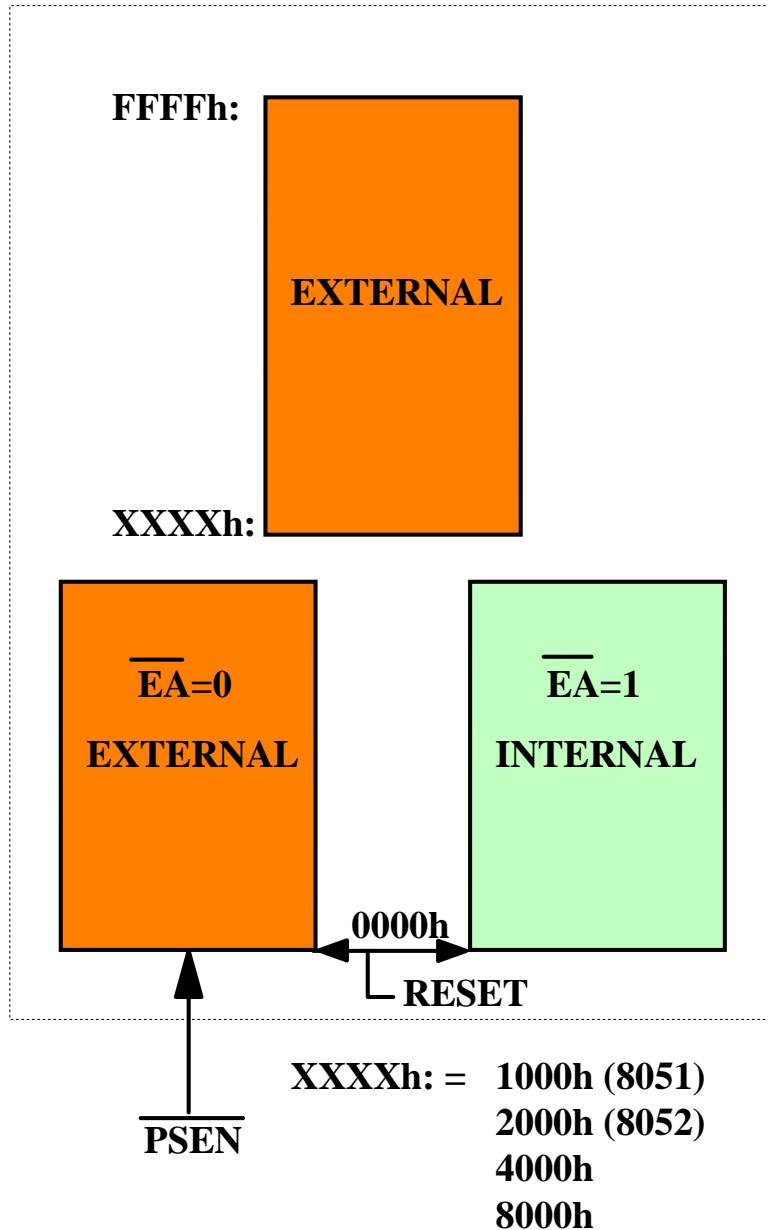
N:

DB 5

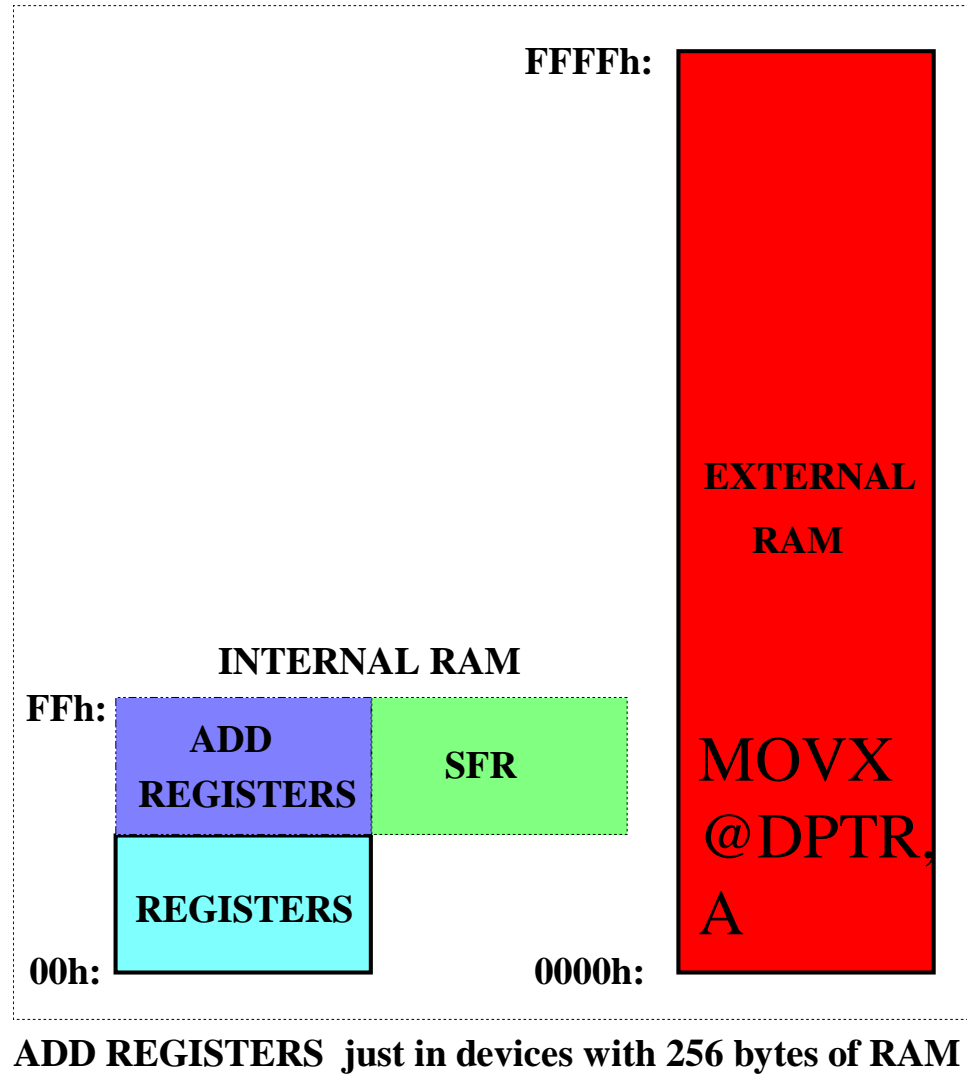
END

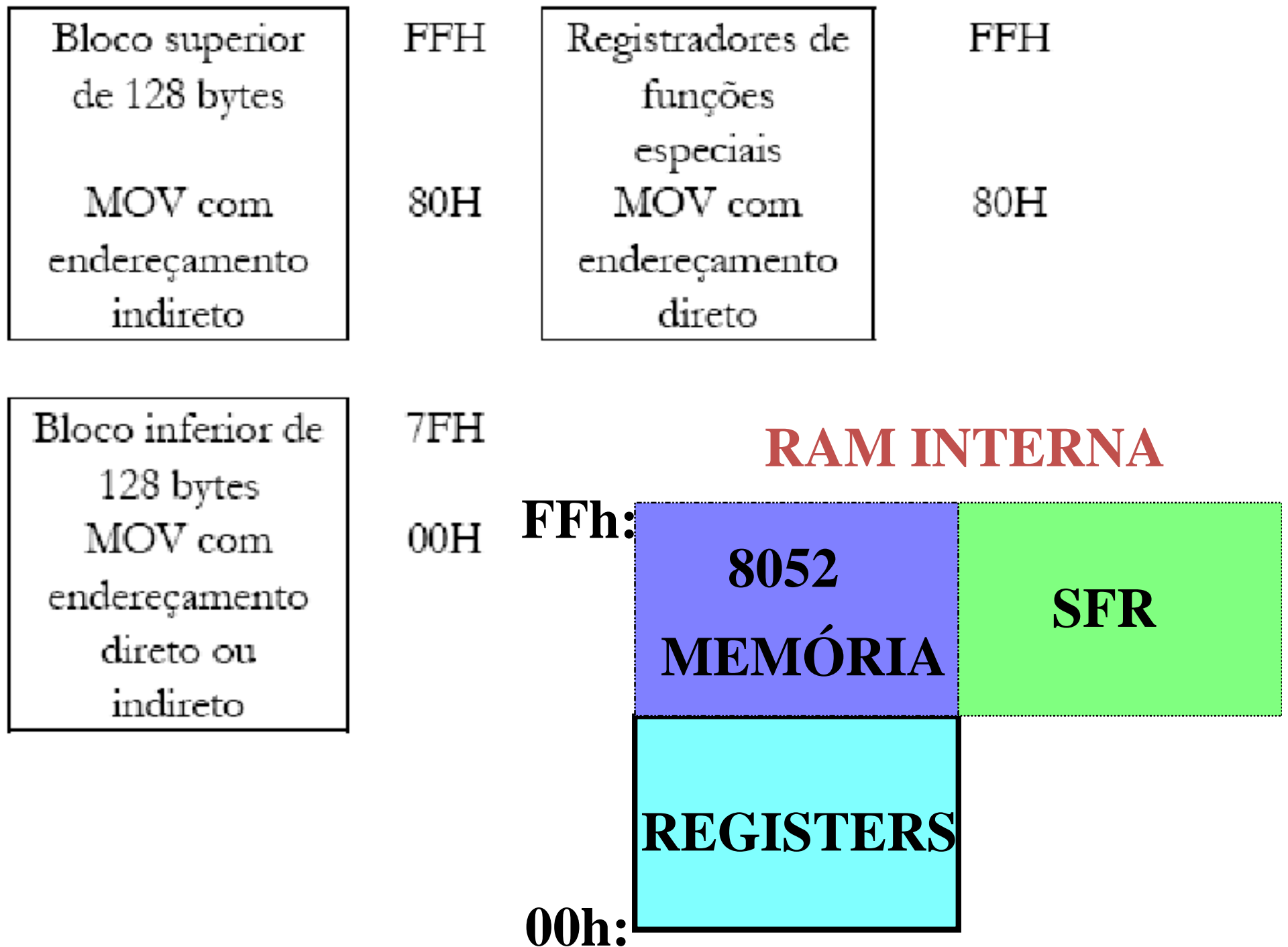
Organização da Memória

MEMÓRIA DE PROGRAMA




MEMÓRIA DE DADOS



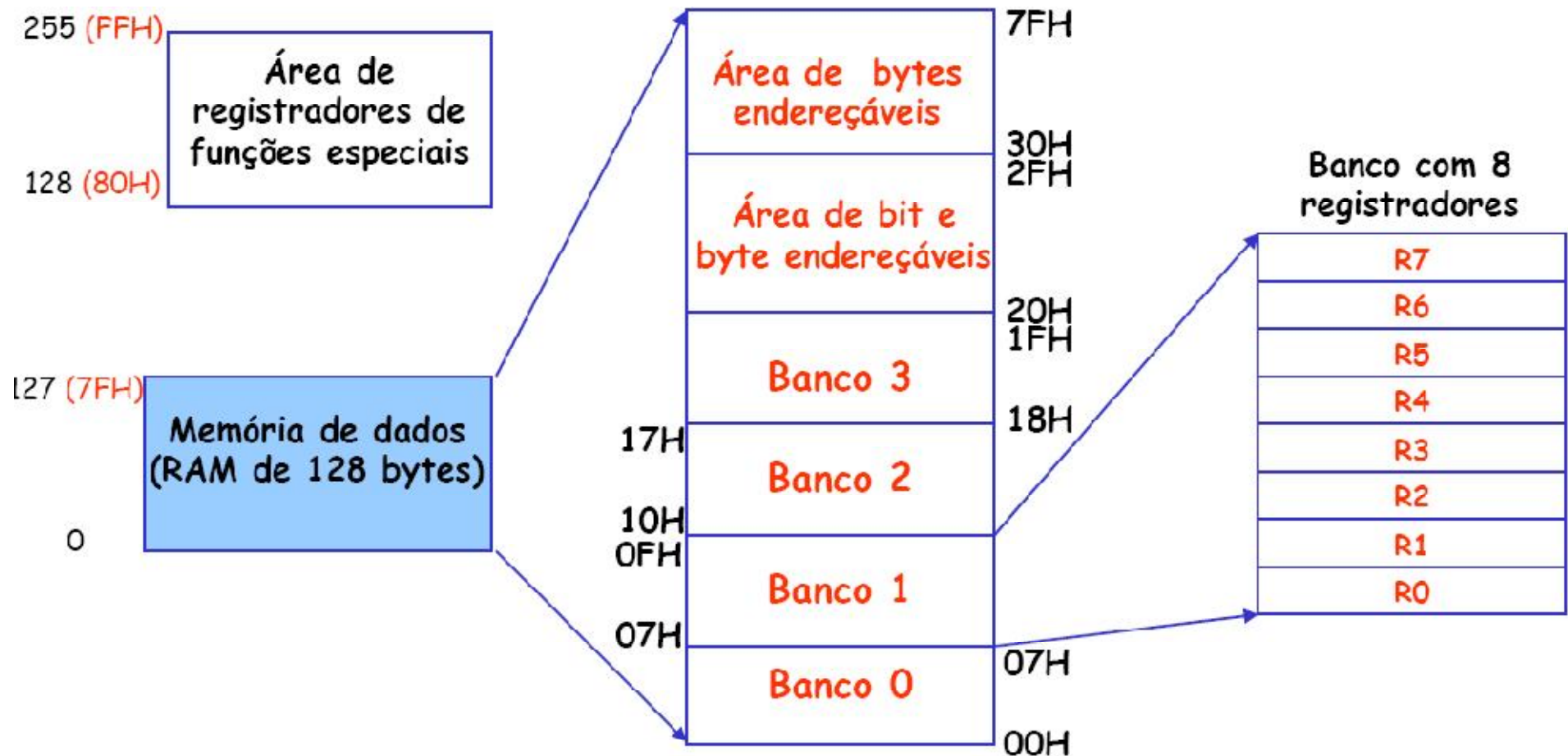


Special Function Registers (SFR)

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

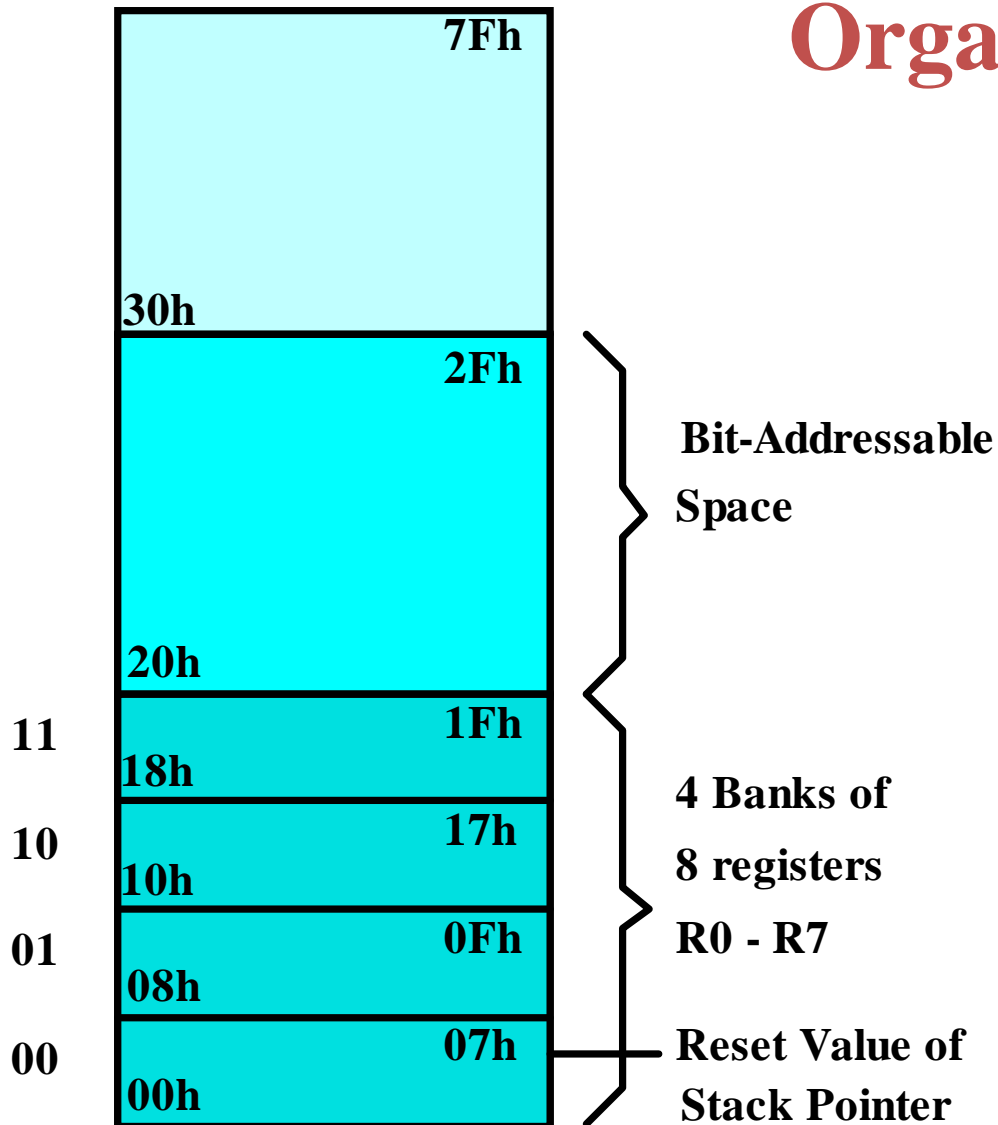


Organização da RAM Interna

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

Organização da RAM Interna

Low 128 bytes of Internal RAM



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

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

	C	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
CJNE	X		

OBS: *Flag* de paridade reflete qualquer alteração de conteúdo do acumulador