Arquitetura de Computadores

PROF. ISAAC

Memória

RAMs

- **RAMs** podem ser de duas variedades, estáticas e dinâmicas.
- Nas estáticas (**Static RAMs** – **SRAMs**), a construção interna usa circuitos similares ao nosso *flip-flop* D básico.
- RAMS dinâmicas (**Dynamic RAMs** – **DRAMs**), ao contrário, não usam flip-flops.
- Em vez disso, uma RAM dinâmica é um arranjo de células, cada uma contendo um transistor e um pequenino capacitor.

RAMs

- RAM Dinâmica (**DRAM**)
  - Células armazenam dados com a carga de capacitores;
  - É necessário um circuito de regeneração (refresh);
  - Usada na Memória Principal.

- RAM Estática (**SRAM**)
  - Valores são armazenados usando configurações de flip-flops com portas lógicas
  - Não é necessário o circuito de regeneração;
  - Usada na Memória Cache.

ROMs

- Em muitas aplicações, como brinquedos, eletrodomésticos e carros, o programa e alguns dos dados devem permanecer armazenados mesmo quando o fornecimento de energia for interrompido.
- Uma vez instalados, nem o programa nem os dados são alterados.
- Esses requisitos levaram ao desenvolvimento de **ROMs** (**Read-Only Memories** – **memórias somente de leitura**), que não podem ser alteradas nem apagadas, seja intencionalmente ou não.

ROMs

- ROM programável (**PROM**)
  - Mais barata que a ROM;
  - Pode ser escrita (eletricamente) apenas uma vez;
  - Necessário um equipamento especial para o processo de escrita ou "programação".

- ROM programável e apagável (**EPROM**)
  - Lida e escrita eletricamente;
  - Antes da escrita todas as células de armazenamento são apagados através da exposição à luz ultravioleta intensa;
  - Mais cara que a PROM.

ROMs

- ROM programável e apagável eletronicamente (**EEPROM**)
  - Escrita pode ser feita somente nos bytes endereçados, sem modificar os demais;
  - Mais cara que a EPROM e menos densa.

- **Flash**
  - Intermediária entre a EPROM e EEPROM tanto no custo quanto na funcionalidade;
  - Usa tecnologia elétrica de apagamento;
  - Alta densidade.

Arquitetura de Von Neumann

- Essa arquitetura foi proposta em junho de 1945 por von Neumann que, em seu trabalho “First Draft of a Report on the EDVAC”, apresentou três postulados para a construção de computadores:

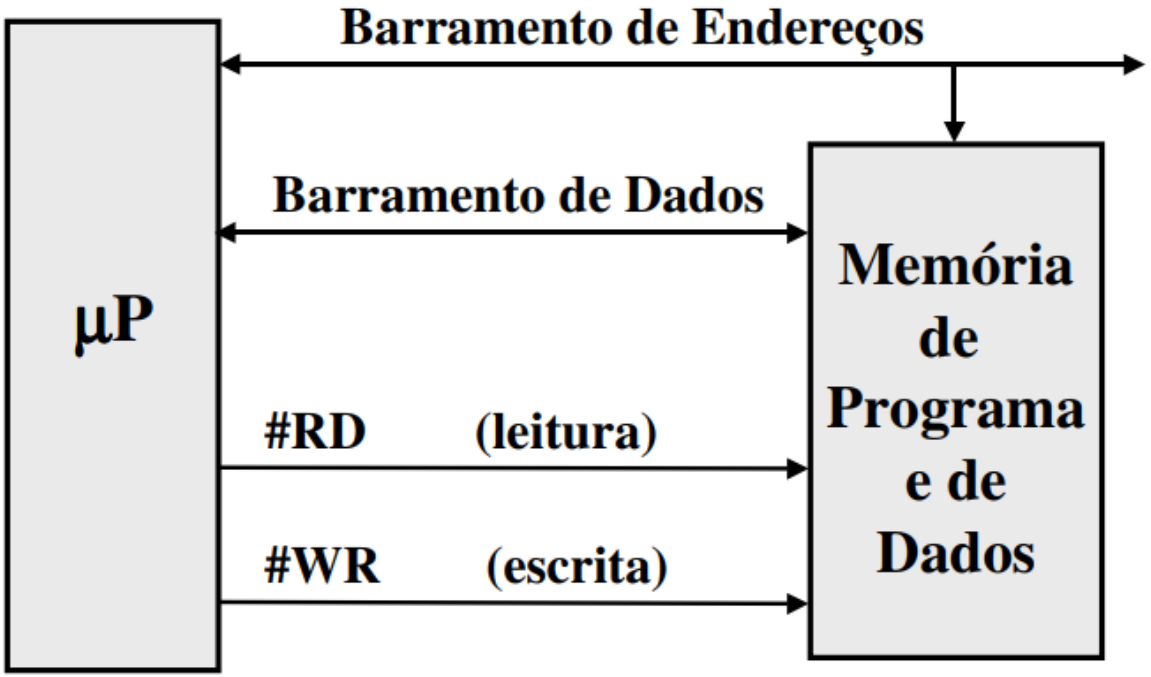
- **Um único controle central** – de maneira bem simples, é o fato de ter-se nos processadores um único contador de programa, ou PC (Program Counter);

- **Uma única memória para programas e dados** – é a memória RAM dos computadores compartilhada para código e dados;

- **As instruções devem fazer operações elementares sobre os dados** – sugere que as operações efetuadas pelas instruções sobre os dados deveriam ser operações simples.

Não seguir um desses três postulados é dito ter uma arquitetura “não von Neumann”.

Arquitetura de Von Neumann

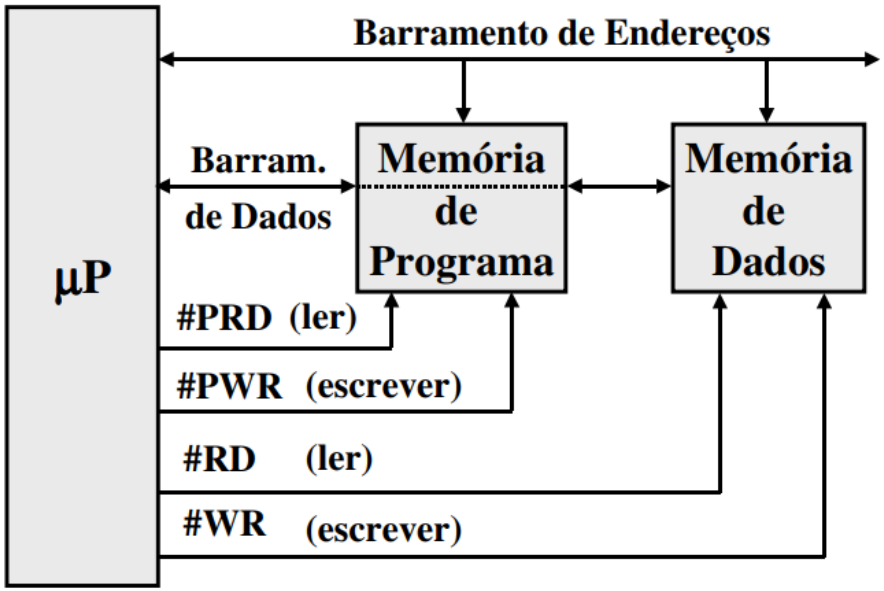


Arquitetura de Harvard

- Na arquitetura de Harvard o processador trabalha com a memória de programa separada da memória de dados.

- Uma das vantagens dessa arquitetura é evitar que a memória se torne um gargalo, pois é possível fazer acesso simultâneo às duas memórias, desde que se disponibilize barramentos separados.

- Nos processadores modernos, as memórias cache L1 estão separadas em cache de programa e cache de dados.



Memória do 8051

- **ROM** de até 64KB de endereçamento;
- **RAM**:
  - 256B de dados interna;
  - Endereçamento externo até 64KB;
  - Pode ser utilizada RAM externa (perdendo alguns pinos de I/O);

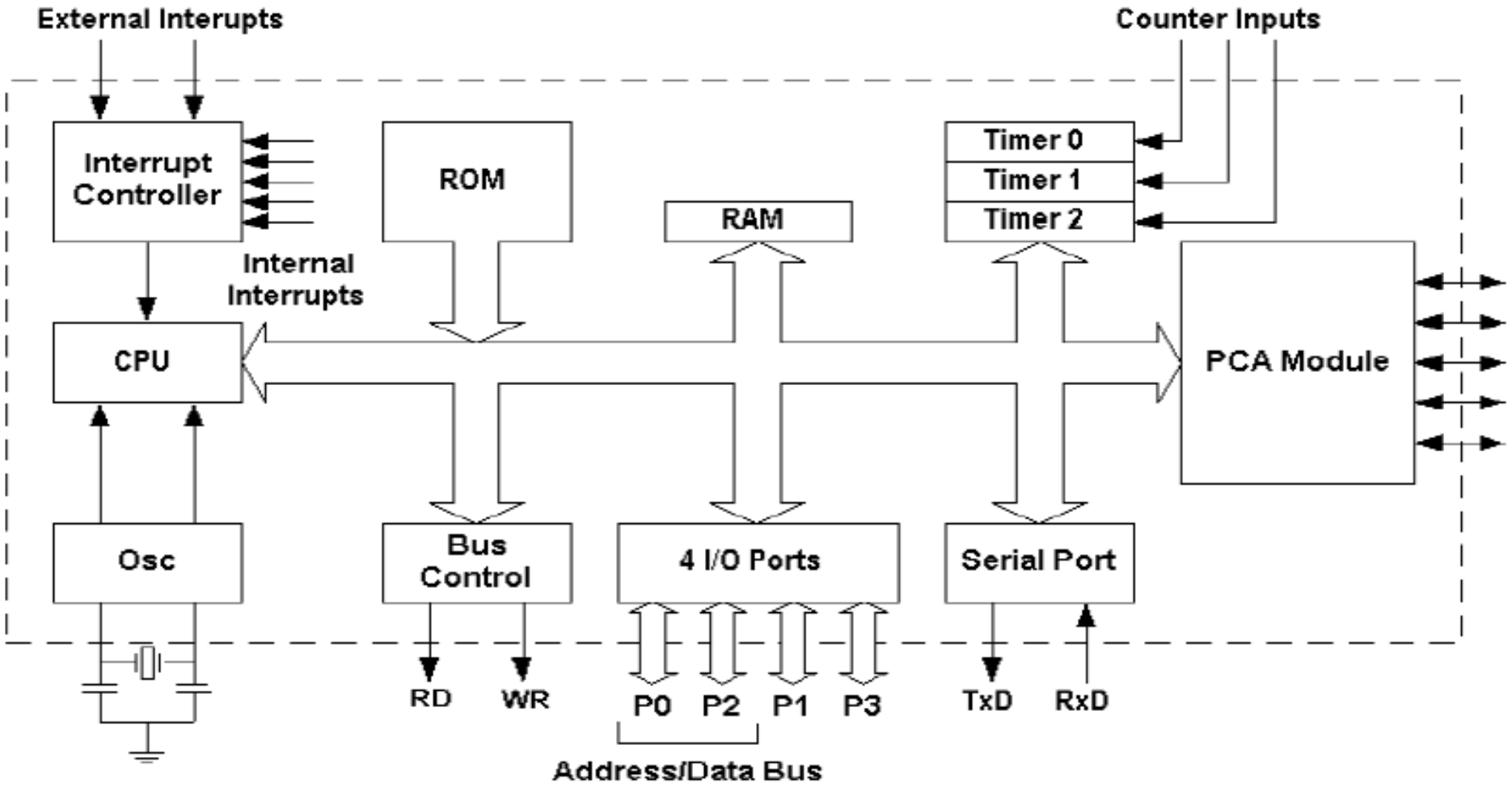
ROM x RAM

- ROM armazena o programa (firmware);
- RAM armazena os dados do programa durante a execução;
- No 8051 as memórias são separadas logicamente.

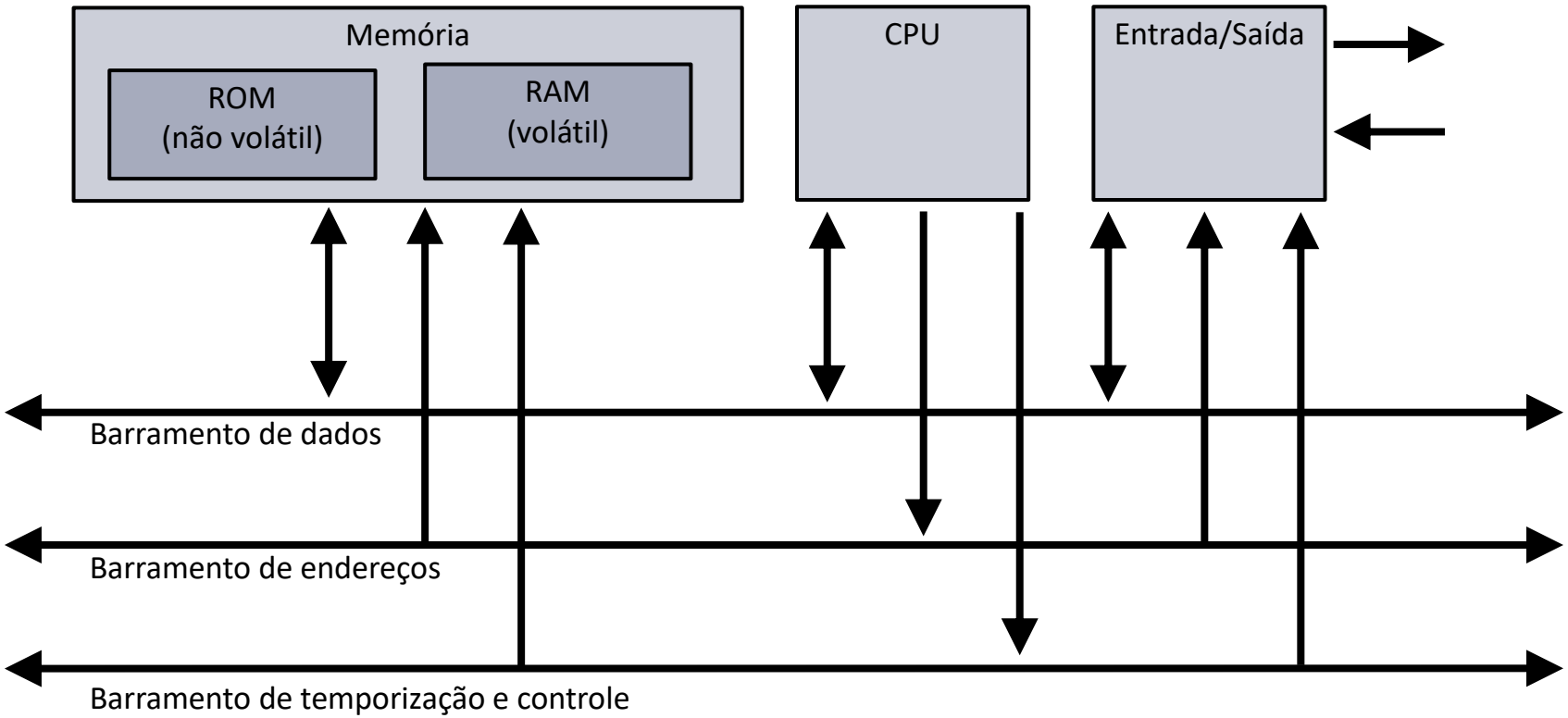
Separação Lógica

- A separação é feita pela forma como os endereços são utilizados;
- Memória de programa usa endereçamento de:
  - 16 Bits (0000H até FFFFH);
- Memória de dados (RAM) usa endereçamento:
  - 8 Bits para interno (00H até FFH);
  - 16 Bits para externo;

Visão geral em Diagrama de Blocos



Microcontrolador



Barramentos

- **Barramento de endereço**: Barramento unidirecional, por onde trafega os sinais que endereçam os dispositivos de memória e I/O.

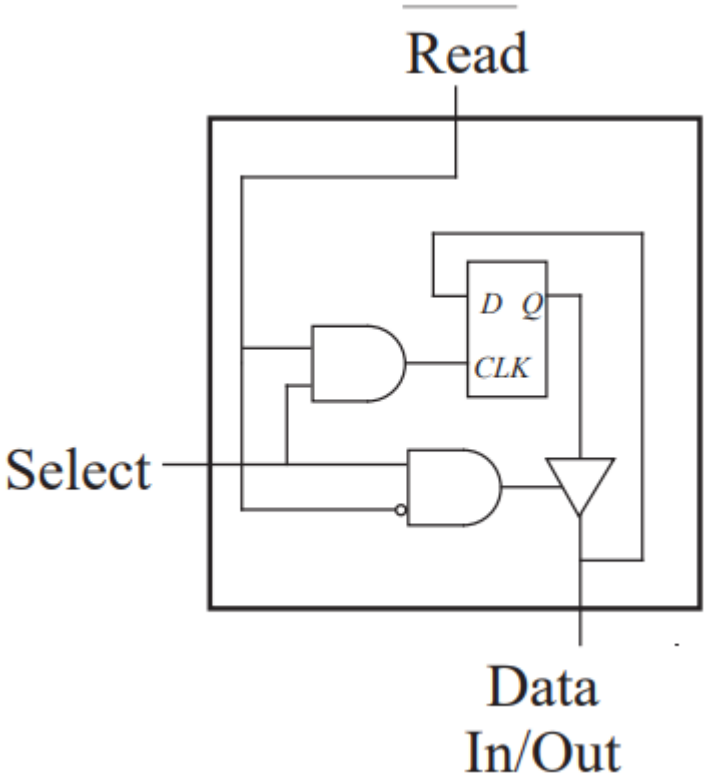
- **Barramento de dados**: Barramento bidirecional por onde trafegam dados entre os dispositivos externos e o microcontrolador. O tamanho deste barramento define o tamanho do microcontrolador. O 8051 é de 8 bits.

- **Barramento de controle**: Barramento por onde o microcontrolador controla os dispositivos.

Conceitos Sobre Memória

A figura mostra o elemento de memória como um flip-flop D, com controles adicionais para permitir que a célula seja selecionada para leitura ou escrita.

Existe uma linha de dados (bidirecional) para entrada e saída de dados de dados.



Conceitos Sobre Memória

- **Barramento de endereço**:

  $A_0 - A_{n-1}$

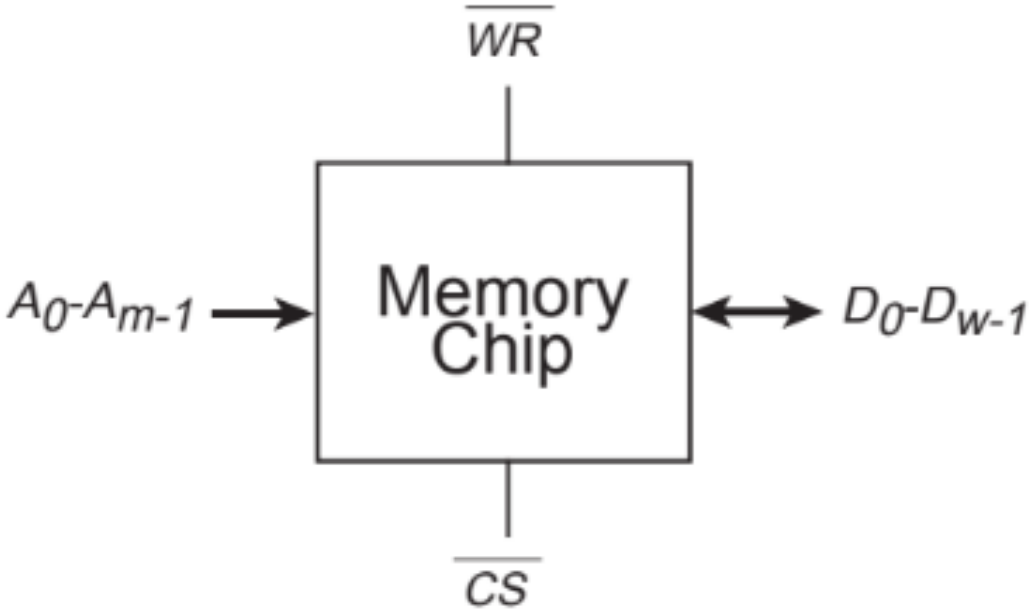
- **Barramento de dados**:

  $D_0 - D_{w-1}$

- **Barramento de controle**:

  CS (Chip Select)

WR - Para escrita ou leitura de dados.



Conceitos sobre Memória

- Para que serve o CS (Chip Select) ?

  - Porque geralmente temos vários chips de Memória.

Memória do 8051

- **ROM** de até 64KB de endereçamento;
- **RAM**:
  - 256B de dados interna;
  - Endereçamento externo até 64KB;
  - Pode ser utilizada RAM externa (perdendo alguns pinos de I/O);

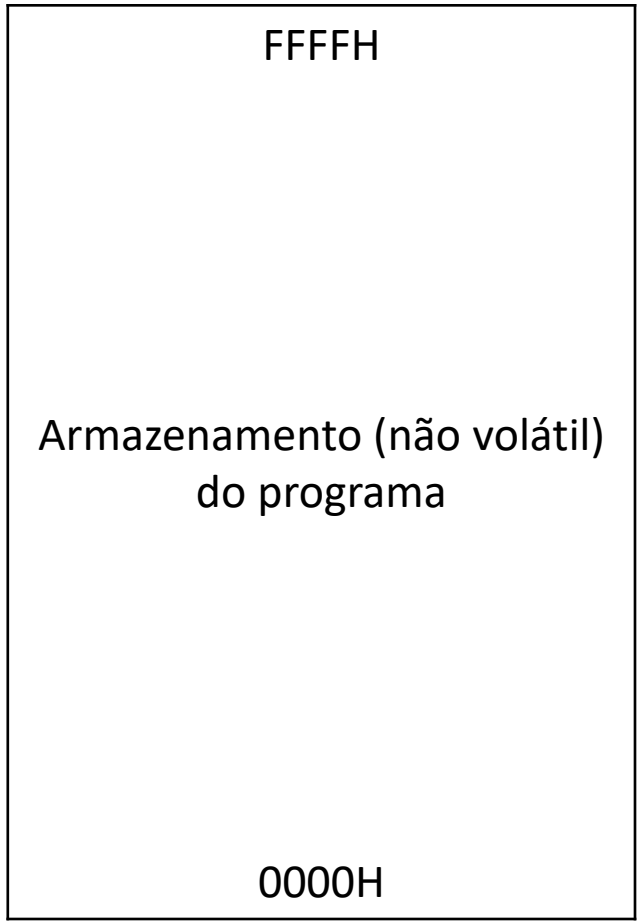
Memória do 8051 – ROM

Endereçamento de 16-bits;

- Início em 0000H (0);
- Final em FFFFH (65.535);

Cada posição aponta para 1 Byte;

No máximo, 64KB para armazenar o programa (podendo ser menor conforme o fabricante);



Memória do 8051 – RAM

Endereçamento de 16-bits;

Para o programa:

- Início em 00H
- Final em 7FH
- Área que o programa pode usar para armazenar os dados

Uso interno:

- Início em 80H
- Final em FFH

Registradores

Endereços de memória que possuem funções especiais;

No 8051:

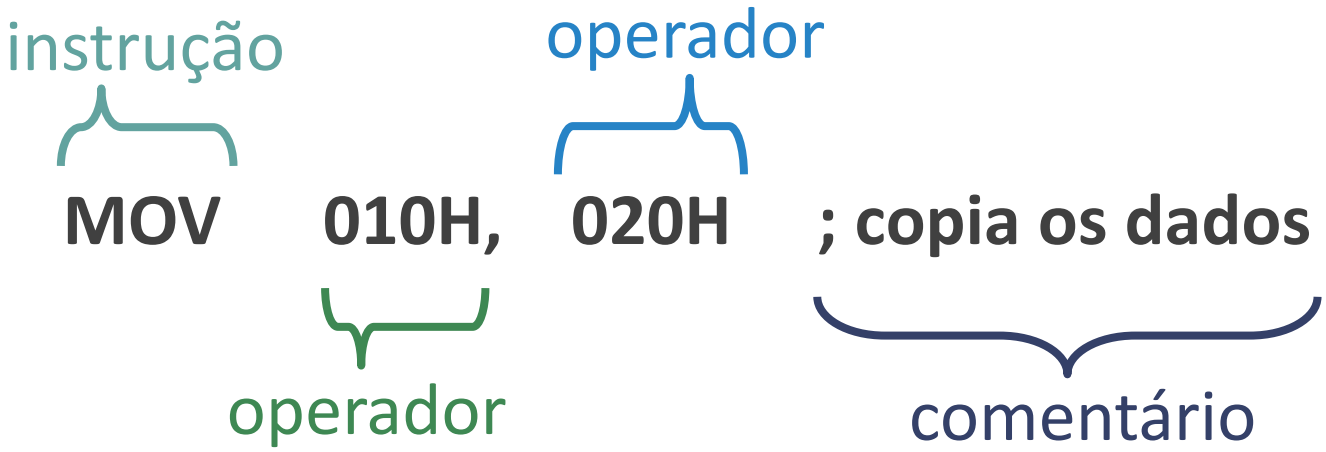
- A (acumulador): utilizado em (quase) todas as operações aritméticas;
- R0 até R7: são 8 registradores que podem ser acessados diretamente, além dos 127B de RAM;
- R0 e R1: também podem ser acessados indiretamente;
- DPTR: ponteiro na RAM externa;
- PSW: Program Status Word.

Assembly!

Assembly

- Programar em assembly é muito diferente de se programar em C pois, nesse nível, o programador é responsável por tudo que seu programa faz.
- Em especial, quando se trabalha com microcontroladores, já que raramente há a disponibilidade de um sistema operacional. O programador passa a ser responsável por gerenciar a memória, por indicar onde deve carregar o programa, por reservar memória para seus dados, por dar significado às suas variáveis e ainda por trabalhar de maneira correta com a representação numérica adotada.

Sintaxe



- O nome de uma instrução possui entre 2 e 5 letras;
- Instruções podem conter até 3 operadores;
- Um comentário é iniciado por ponto e vírgula depois da instrução;
- Os operadores podem ser valores, portas, endereços de memória, registradores, etc.

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		Bytes	MC	Op1	Op2
INC	A	1	1	04	-
	Rn	1	1	08+n	-
	end8	2	1	05	end8
	@Ri	1	1	06+i	-

Instruções de incremento de 8 bits.

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		Bytes	MC	Op1	Op2
INC	A	1	1	04	-
	Rn	1	1	08+n	-
	end8	2	1	05	end8
	@Ri	1	1	06+i	-

Instruções de incremento de 8 bits.

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Instrução	Cod. HEX	num. bytes
nop	00	1
inc R0	08	1
mov A, #55H	74 55	2
mov R0, #0AAH	78 AA	2
mov R0, 0AAH	A8 AA	2
mov DPTR, #55AAH	90 55 AA	3

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

8051 – Código objeto

Hex	Assembly
0000:	.equ cout, 0x0030
0000:	.equ cin, 0x0032
0000:	.equ esc, 0x004E
8000:	.org 0x8000
8000: 79 61	mov r1, #'a'
8002: 78 1A	mov r0, #26
	next_char:
8004: E9	mov A, r1
8005: 12 00 30	lcall cout
8008: 09	inc r1
8009: D8 F9	djnz r0, next_char
800B: 12 00 32	lcall cin
800E: 02 00 00	ljmp 0x0000

Address	Value
8000	79
8001	61
8002	78
8003	1A
8004	E9
8005	12
8006	00
8007	30
8008	09
8009	D8
800A	F9
800B	12
800C	00
800D	32
800E	02
800F	00
8010	00

Instruções

		Bytes	MC	Op1	Op2
ADD A,	Rn	1	1	28+n	-
	end8	2	1	25	end8
	@Ri	1	1	26+i	-
	#dt8	2	1	24	dt8

		Bytes	MC	Op1	Op2
INC	A	1	1	04	-
	Rn	1	1	08+n	-
	end8	2	1	05	end8
	@Ri	1	1	06+i	-

		Bytes	MC	Op1	Op2
DEC	A	1	1	14	-
	Rn	1	1	18+n	-
	end8	2	1	15	end8
	@Ri	1	1	16+i	-

		bytes	MC	Op1	Op2	Op3
MOV A,	Rn	1	1	E8+n	-	-
	end8	2	1	E5	end8	-
	@Ri	1	1	E6+i	-	-
	#dt8	2	1	74	dt8	-
MOV Rn,	A	1	1	F8+n	-	-
	end8	2	2	A8+n	end8	-
	#dt8	2	1	78+n	dt8	-
MOV end8,	A	2	1	F5	end8	-
	Rn	2	2	88+n	end8	-
	end8	3	2	85	end8 (fonte)	end8 (destino)
	@Ri	2	2	86+i	end8	-
	#dt8	3	2	75	end8	dt8
MOV @Ri	A	1	1	F6+i	-	-
	end8	2	2	A6+i	end8	-
	#dt8	2	1	76+i	dt8	-
MOV DPTR	#dt16	3	2	90	MSB(dt16)	LSB(dt16)

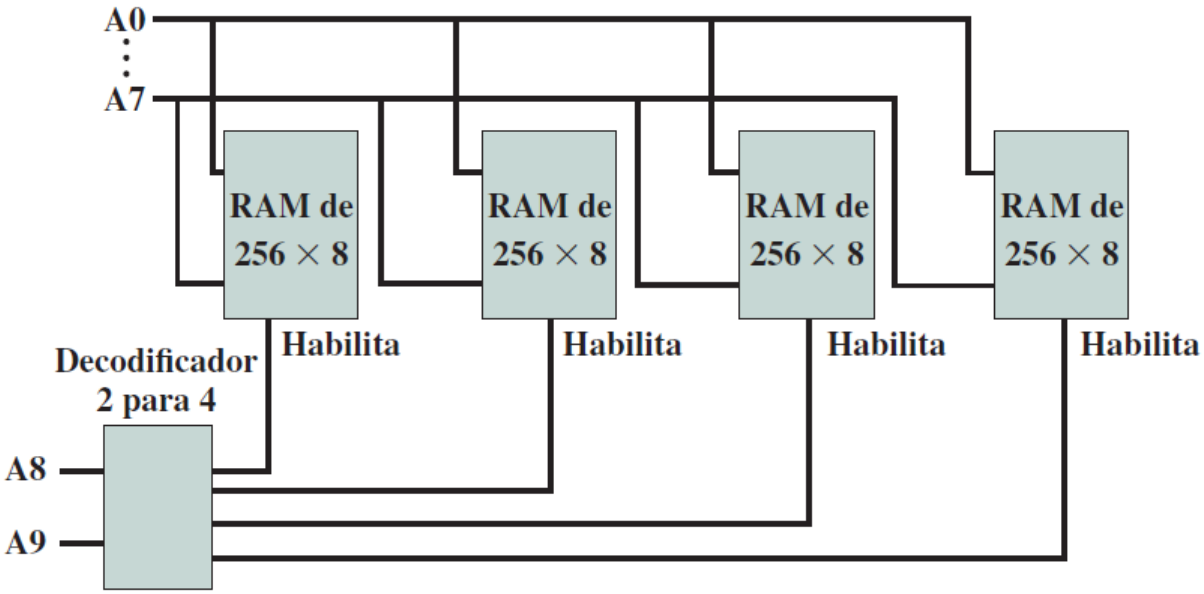
Exercícios

1) O circuito abaixo foi construído com chips de memória de 256 endereços que armazenam 8 bits em cada endereço.

a) Quantos bits são necessários no barramento de endereço para acesso as células dessa memória?

b) Quantos bits são necessários no barramento de dados para acesso ao byte da memória?

c) Qual o tamanho da memória?



Exercícios

2) Um sistema possui memória com 8G endereços e cada célula é composta por 4 bytes.

a) Qual o tamanho da memória?

b) Quantos bits são necessários no barramento de endereço para acesso as células dessa memória?

c) Quantos bits são necessários no barramento de dados para acesso aos 4 bytes dessa memória?

Exercícios

3) Existem programas responsáveis por transformar uma lista de mnemônicos num arquivo executável e, esses programas são denominados montadores, em inglês **assemblers**.

Complete a tabela abaixa com os códigos de operação para as instruções do programa.

Instrução	***opcode***
`MOV A,R1`	
`ADD A,R0`	
`MOV R7,A`	

Exercícios

4) Complete a tabela abaixa com os códigos de operação para as instruções de soma aritmética.

	Opcode	
Instrução	**Byte 1**	**Byte 2**
`ADD A,R0`		
`ADD A,R1`		
`ADD A,R7`		
`ADD A,@R0`		
`ADD A,@R1`		
`ADD A,#43H`		
`ADD A,43H`		

Exercícios

5) Complete a tabela abaixa com os códigos de operação para as instruções do programa.

Instrução	OPCODE
MOV A, R4	
ADD A, R7	
MOV R5, A	
INC A	
ADD A, 10	
MOV A, #0	

Bibliografia

Stallings, Willian. Arquitetura e Organização de Computadores. 10ª Ed, Pearson, 2017.

Murdocca, Miles J., and Vincent P. Heuring. Introdução à arquitetura de computadores. Elsevier, 2001.

David A.Patterson & John Hennessy. Organização e projeto de computadores: A interface de Hardware e Software. 4ª Ed. Elsevier. 2014.

ZELENOVSKY, R.; MENDONÇA, A. Microcontroladores Programação e Projeto com a Família 8051. MZ Editora, RJ, 2005.