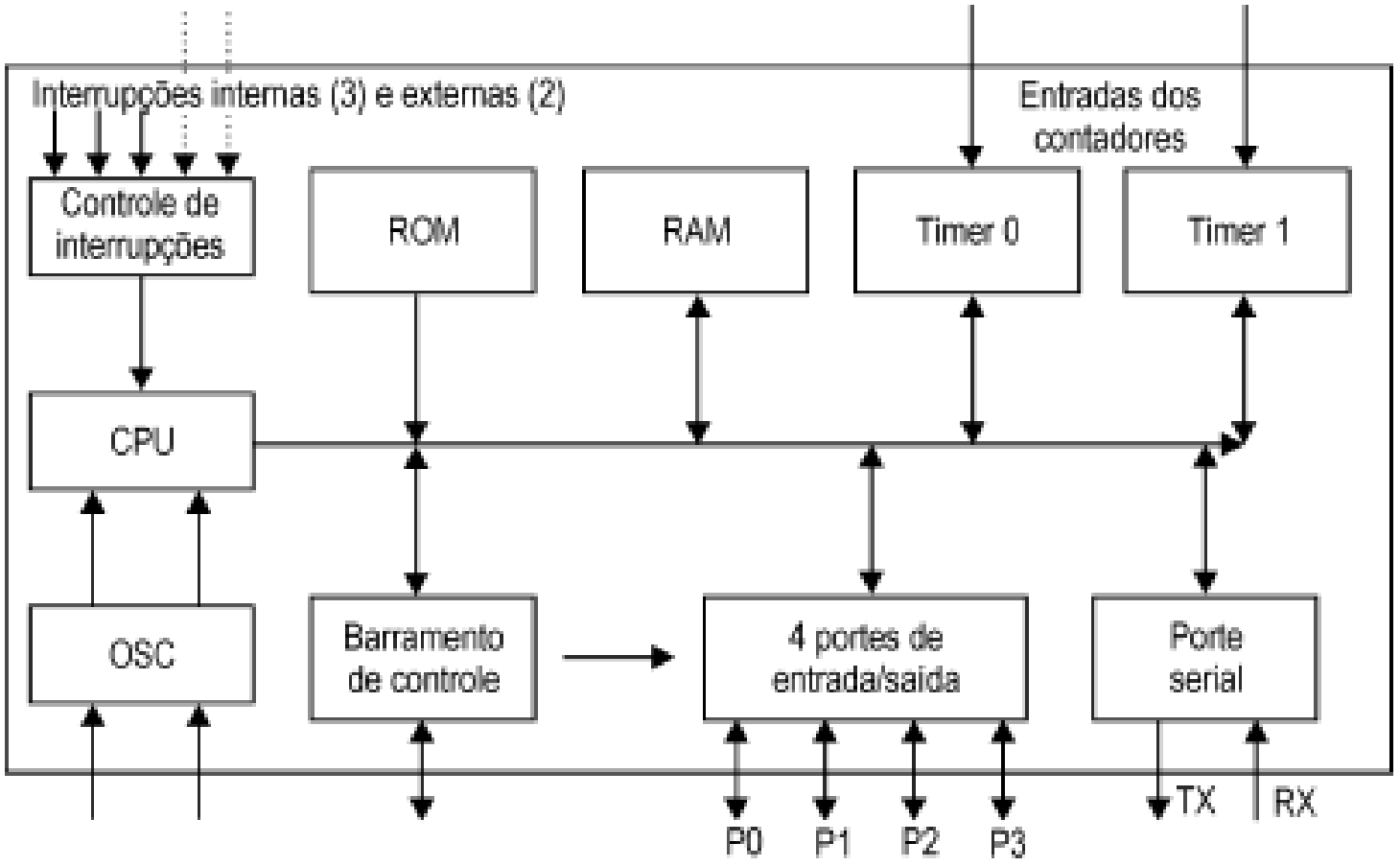
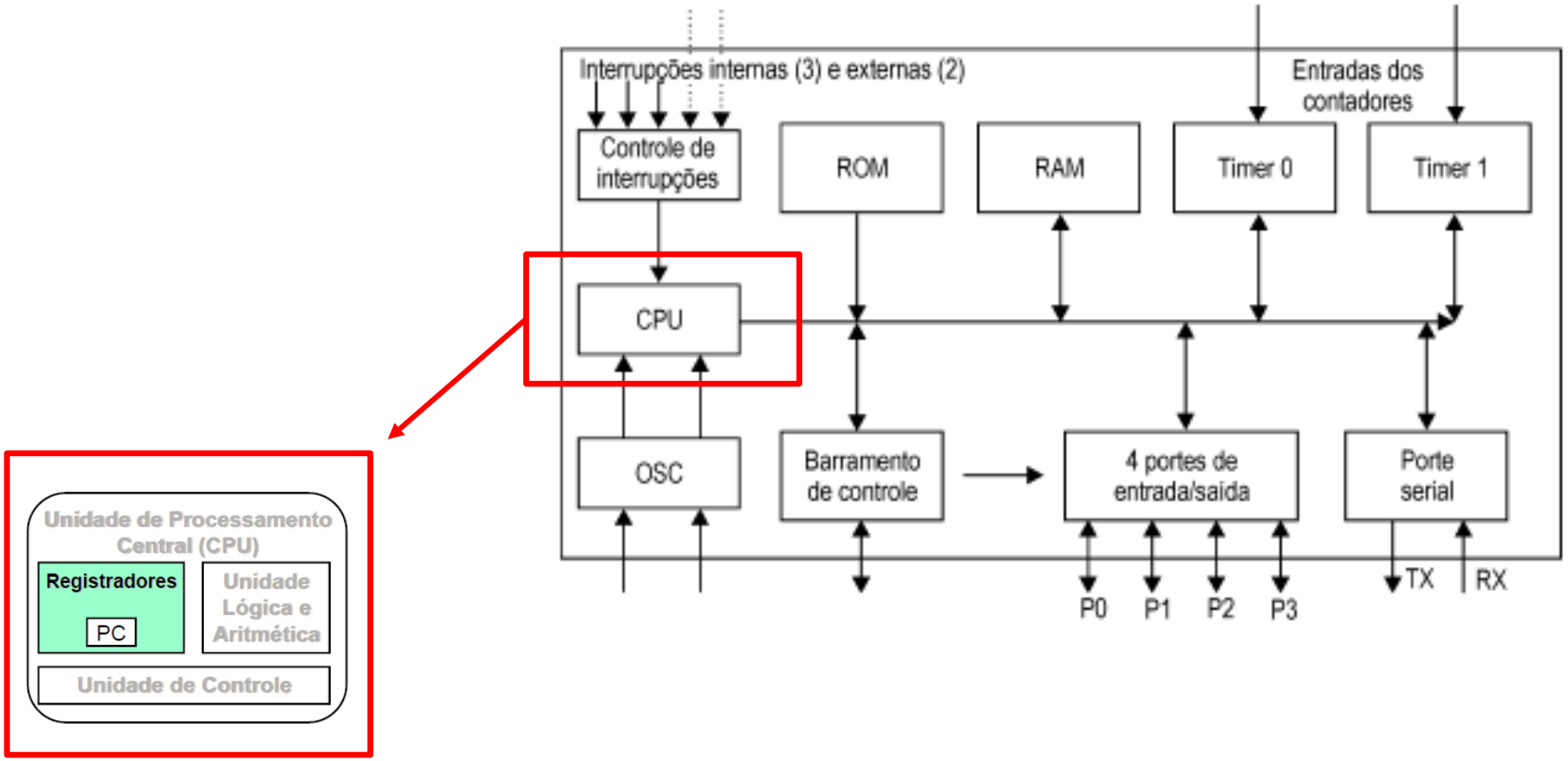
Arquitetura de Computadores

PROF. DR. ISAAC

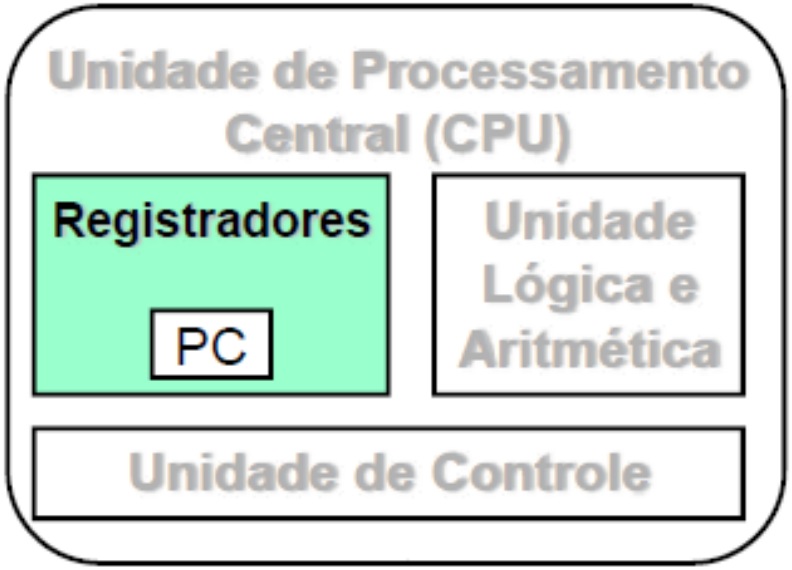
Microcontrolador 8051



Microcontrolador 8051



Microcontrolador 8051



CPU do microcontrolador

8051 -Registradores

- Usados para armazenar temporariamente informações enquanto os dados estão sendo processados.
- São as estruturas de memória mais rápidas e caras.
- Registradores mais comuns:
  - A, B, R0 - R7: registradores de 8 bits.
  - DPTR : [DPH:DPL] Registrador de 16 bits.
  - PC : Contador do Programa–16 bits.
  - 4 conjuntos de bancos de registradores R0-R7.
  - Ponteiro da pilha – SP.
  - PSW: Program Status Word (flags).
  - SFR : Special Function Registers. Controla os periféricos onboard.

8051 -Registradores

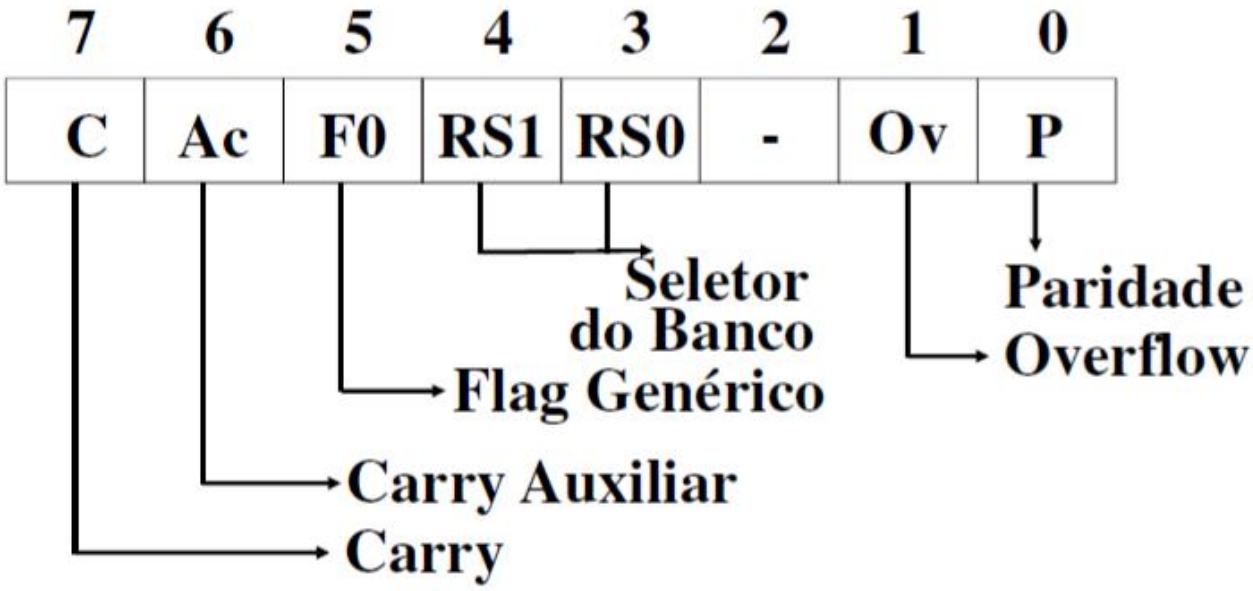
- Usados para armazenar temporariamente informações enquanto os dados estão sendo processados.
- São as estruturas de memória mais rápidas e caras.
- Registradores mais comuns:
  - A, B, R0 - R7: registradores de 8 bits.
  - DPTR : [DPH:DPL] Registrador de 16 bits.
  - PC : Contador do Programa–16 bits.
  - 4 conjuntos de bancos de registradores R0-R7.
  - Ponteiro da pilha – SP.
  - PSW: Program Status Word (flags).
  - SFR : Special Function Registers. Controla os periféricos onboard.

8051 –Registrador PSW

Todo processador possui um registrador especial onde ficam armazenadas informações sobre o estado do processamento e também sobre a última operação realizada pela unidade de lógica e aritmética. Usam-se vários nomes para designá-lo, sendo o mais comum Registrador de Flags.

Na arquitetura 8051, ele recebe o nome **Palavra de Estado do Programa,** representado pela sigla **PSW (Program Status Word).**

8051 –Registrador PSW

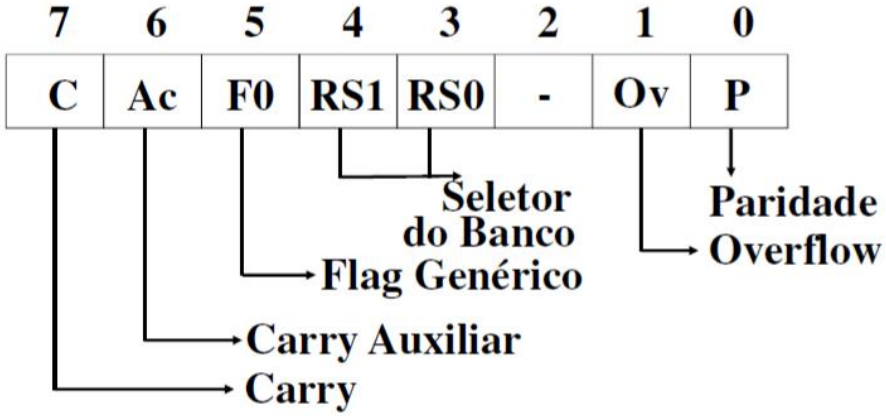


8051 –Registrador PSW

P é o bit de paridade gerado a partir do conteúdo do acumulador.

- P = 1 indica uma quantidade ímpar de bits “1”.
- P = 0 indica uma quantidade par de bits “1”.

Ov é o bit de overflow, ou estouro.

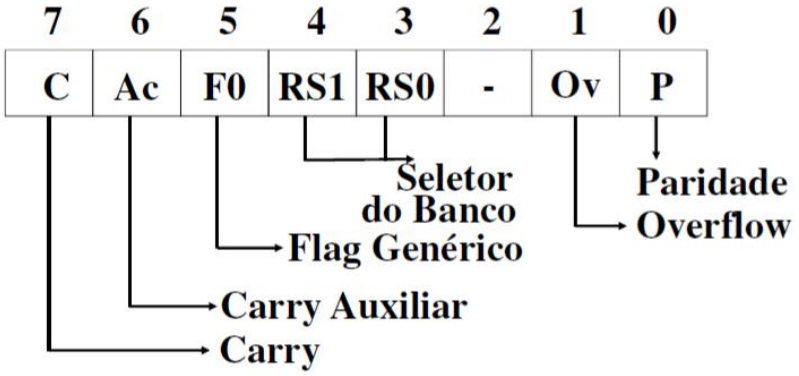


8051 –Registrador PSW

F0 é um bit de uso genérico.

Ac é o carry auxiliar, correspondendo ao “vai-um” do bit 3 para o bit 4. Ele é empregado pela instrução de ajuste decimal (DA A), que deve ser executada logo após uma soma de números representados em BCD.

C é o carry das operações lógicas e aritméticas. Tem uso nas mais diversas operações e corresponde também ao “vai-um” do bit 7 para o bit 8 nas operações de soma.



Instrução - ADD

Operação: ADD

Função: Adiciona o valor do operando ao valor do acumulador.

Sintaxe: ADD A, *operando*

Descrição : ADD adiciona o valor do operando ao valor do acumulador.

- **Carry bit (C)** – setado se existe um carry saindo do bit 7. Em outras palavras, se a soma do valor no acumulador e o operando excede 255.
- **Overflow (OV)** – setado se existir um carry saindo do bit 6 ou do bit 7, mas não dos dois. Em outras palavras, se a soma do acumulador e operando excede a faixa do valor armazenado em um byte com sinal (-128 até +127) o OV é setado, caso contrário, seu valor estará em 0.

Exemplo:

- ADD A, #03h
- ADD A, R0
- ADD A, @R1

Instrução - ADD

Operação: ADD

Função: Adiciona o valor do operando ao valor do acumulador.

Sintaxe: ADD A, *operando*

Descrição : ADD adiciona o valor do operando ao valor do acumulador.

- **Carry bit (C)** – setado se existe um carry saindo do bit 7. Em outras palavras, se a soma do valor no acumulador e o operando excede 255.
- **Overflow (OV)** – setado se existir um carry saindo do bit 6 ou do bit 7, mas não dos dois. Em outras palavras, se a soma do acumulador e operando excede a faixa do valor armazenado em um byte com sinal (-128 até +127) o OV é setado, caso contrário, seu valor estará em 0.

Exemplo:

- ADD A, #03h
- ADD A, R0
- ADD A, @R1

Instruções do MSC-51

Convenções empregadas no estudo do conjunto de instruções.

Símbolo	Significado
Rn	Qualquer um dos registradores: R0, R1, R2, R3, R4, R5, R6, R7.
@Ri	Qualquer um dos registradores: R0, R1.
#dt8	Um número de 8 bits.
#dt16	Um número de 16 bits.
end8	Um endereço de 8 bits, faz referência à RAM interna.
end11	Um endereço de 11 bits, faz referência à memória de dados externa.
end16	Um endereço de 16 bits, faz referência à memória de dados externa.
rel	Um deslocamento relativo de 8 bits, em complemento 2: de –128 a +127.
bit	Endereço de um bit da RAM interna (da área acessível bit a bit).
A	Acumulador.
Acc	Endereço do acumulador (E0H).

Instruções do 8051

Lógica	Aritmética	Memória	Outros
ANL	ADD ✓	MOV ✓	NOP
ORL	ADDC ✓	MOVC	RET e RETI
XRL	SUBB ✓	MOVX	ACALL e LCALL
CLR	MUL ✓	PUSH	JMP
CPL	DIV ✓	POP	AJMP
RL	INC ✓	XCH	LJMP
RLC	DEC ✓	XCHD	SJMP
RR	DA		JB e JNB
RRC			JZ e JNZ
SWAP			JC e JNC
SETB			JBC
			DJNZ
			CJNE

Instruções Lógicas

Instrução - ANL

Operação: ANL

Função: AND bit a bit

Sintaxe: ANL *operando1*, *operando2*

Descrição : ANL realiza um AND entre o *operando1* e *operando2*, deixando o resultado no *operando2*.

Exemplo:

- ANL A, #10H
- ANL 20h, #00H

Instrução - ANL

		Bytes	MC	Op1	Op2	Op3
ANL A,	Rn	1	1	58+n	-	-
	end8	2	1	55	end8	-
	@Ri	1	1	56+i	-	-
	#dt8	2	1	54	dt8	-
ANL end8,	A	2	1	52	end8	-
	#dt8	3	2	53	end8	dt8

Instruções para realizar o AND Lógico.

***MC (Machine Cycle) ciclos de máquina.**

Instrução - ORL

Operação: ORL

Função: OR bit a bit

Sintaxe: ORL *operand1*, *operand2*

Descrição : ORL realiza um OU lógico bit a bit entre o *operando1* e *operando2*, deixando o resultado no *operando2*.

Exemplo:

- ORL A, #10H
- ORL 20h, #00H

Instrução - ORL

		Bytes	MC	Op1	Op2	Op3
ORL A,	Rn	1	1	48+n	-	-
	end8	2	1	45	end8	-
	@Ri	1	1	46+i	-	-
	#dt8	2	1	44	dt8	-
ORL end8	A	2	1	42	end8	-
	#dt8	3	2	43	end8	dt8

Instruções para realizar o OR Lógico.

***MC (Machine Cycle) ciclos de máquina.**

Instrução - XRL

Operação: XRL

Função: OU exclusivo bit a bit

Sintaxe: XRL *operando1*,*operando2*

Descrição : XRL realiza um ou exclusivo entre o *operando1* e o *operando2*, deixando o resultado no *operando1*. Um ou exclusivo compara os valor dos bits de cada operando e seta o bit correspondente se um dos bits (mas não os dois) dos operandos estiver setado.

Exemplo:

- **XRL A, #10H**

Instrução - XRL

		Bytes	MC	Op1	Op2	Op3
XRL A,	Rn	1	1	68+n	-	-
	end8	2	1	65	end8	-
	@Ri	1	1	66+i	-	-
	#dt8	2	1	64	dt8	-
XRL end8	A	2	1	62	end8	-
	#dt8	3	2	63	end8	dt8

Instruções para realizar o XOR Lógico.

***MC (Machine Cycle) ciclos de máquina.**

Exemplos com as instruções ANL, ORL e XRL

Zerar os bits 3, 5 e 6 do acumulador	Ativar os bits 3, 5 e 6 do acumulador	Inverter os bits 3, 5 e 6 do acumulador
`ANL A,#1001 0111B`	`ORL A,#0110 1000B`	`XRL A,# 0110 1000B`
X X X X X X X X **1 0 0 1 0 1 1 1** **X 0 0 X 0 X X X**	**X X X X X X X X** **0 1 1 0 1 0 0 0** **X 1 1 X 1 X X X**	**X X X X X X X X** **0 1 1 0 1 0 0 0** **X $\overline{X}$ $\overline{X}$ X $\overline{X}$ X X X**

Neste exemplo a operação AND serve para zerar bits, a operação OR serve para ativar (colocar em 1) bits e a operação XOR serve para inverter bits.

Instruções Lógicas:
Operações Lógicas com o Acumulador

Instruções de operações lógicas com o acumulador

CLR A: inicializa o acumulador com zeros;

CPL A: calcula o complemento 1 do acumulador (inverter todos os bits);

RL A: roda o acumulador à esquerda;

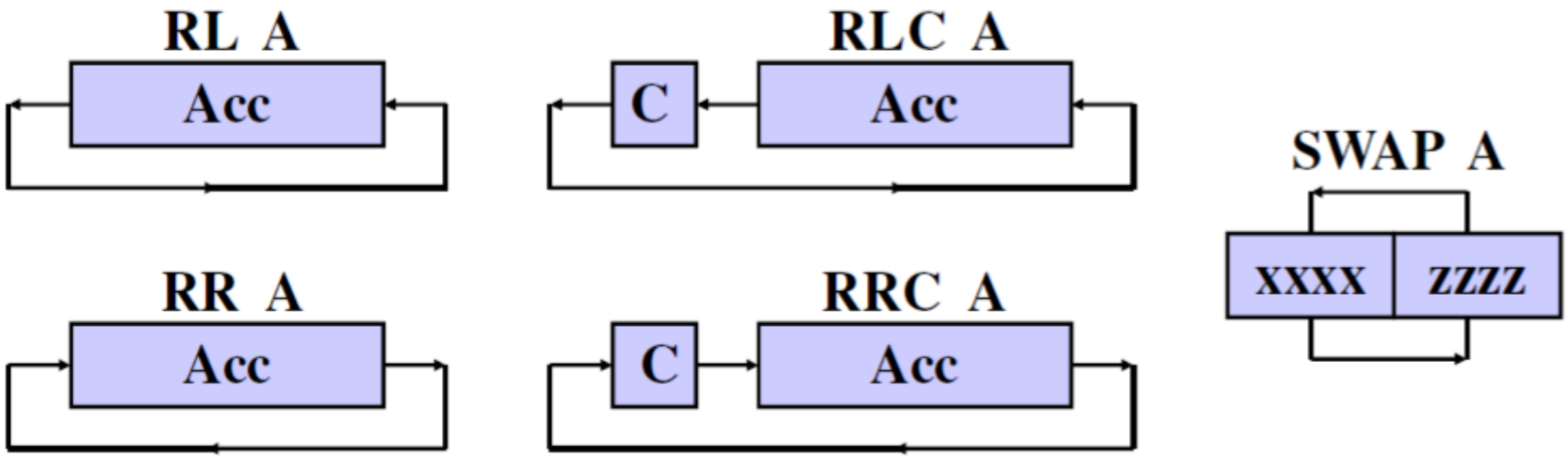
RLC A: roda o acumulador à esquerda, usando o carry;

RR A: roda o acumulador à direita;

RRC A: roda o acumulador à direita, usando o carry;

SWAP A: troca de posição as nibbles do acumulador.

Instruções de operações lógicas com o acumulador



Operações Lógicas com o acumulador.

Instruções de operações lógicas com o acumulador

		Bytes	MC	Op
CLR	A	1	1	E4
CPL				F4
RL				23
RLC				33
RR				03
RRC				13
SWAP				C4

Instruções para operações lógicas com o acumulador.

***MC (Machine Cycle) ciclos de máquina.**

Instruções Booleanas

Instrução - CLR

Operação: CLR

Função: zera (colocar em 0) o valor do bit.

Sintaxe: CLR bit

Descrição : Zera (coloca 1) o valor do bit especificado.

Exemplo:

- CLR RS0
- CLR C
- CLR 20h.1

Instrução - CPL

Operação: CPL

Função: Realiza o complemento (inverter o nível lógico) do bit.

Sintaxe: CPL bit

Descrição : CPL realiza o complemento do bit, que pode ser uma flag, bit de memória, porta ou Carry (C).

Exemplo:

- CPL P1.3
- CPL RS1
- CPL 20h.7

Instrução - SETB

Operação: SETB

Função: ativa (colocar em 1) o valor do bit.

Sintaxe: SETB *bit*

Descrição : Seta (coloca 1) no bit especificado, ou no Carry (C).

Exemplo:

- SETB P1.3
- SETB C
- SETB RS0
- SETB 20h.0

Instruções de operações lógicas com o acumulador

		Bytes	MC	Op1	Op2
CLR	C	1	1	C3	-
	bit	2	1	C2	bit
SETB	C	1	1	D3	-
	bit	2	1	D2	bit
CPL	C	1	1	B3	-
	bit	2	1	B2	bit

Instruções zerar/ativar/complementar um bit.

***MC (Machine Cycle) ciclos de máquina.**

Instrução - AND / OR Booleano

A instrução **ANL C, bit** executa a operação “C ⇐ C AND bit”.

A instrução **ORL C, bit** executa a operação “C ⇐ C ORL bit”.

Exemplo:

- ANL C, P0.1
- ANL C, 25h.1
- ORL C, /P0.1
- ORL C, B.2

Instrução - AND / OR Booleano

		Bytes	MC	Op1	Op2
ANL C,	bit	2	2	82	bit
	/bit	2	2	B0	bit
ORL C,	bit	2	2	72	bit
	/bit	2	2	A0	bit

Instruções de AND e OR com um bit.

Instrução - Transferência (Cópia) de Bits

Nas instruções de transferência (cópia) de bits, a flag carry (C) sempre está envolvida. Não é permitida a transferência (cópia) direta entre bits, mas sim apenas por intermédio do carry.

Exemplo:

- mov C, 20h.0
- mov Acc.0, C
- mov RS0, C

Instrução - Transferência (Cópia) de Bits

				Bytes	MC	Op1	Op2
MOV	C	,	bit	2	2	A2	bit
MOV	bit	,	C	2	2	92	bit

Instruções de transferência de bits.

Instruções de Desvio:

Saltos Incondicionais

Instrução - AJMP

Operação: AJMP

Função: Desvio absoluto dentro do bloco de 2K da memória de programa

Sintaxe: AJMP *endereço*

Descrição : AJMP desvia o programa para o endereço indicado no parâmetro. Apenas código localizado no bloco de 2k do programa pode ser alvo deste desvio (um campo de 11 bits para o endereço de destino).

Exemplo:

- AJMP LABEL
- AJMP 002h

Instrução - LJMP

Operação: LJMP

Função: Salto longo (long jump).

Sintaxe: LJMP *endereço*

Descrição : LJMP é capaz de desviar a execução para qualquer posição da memória de programa, pois oferece um campo de 16 bits para a especificação do endereço de destino.

Exemplo:

- LJMP LABEL
- LJMP 0002h

Instrução - SJMP

Operação: SJMP

Função: Short Jump

Sintaxe: SJMP *valor*

Descrição : O salto curto, SJMP (short jump), toma como base a posição atual da instrução para desviar o fluxo de execução do programa e, por isso, é denominado de salto relativo. Este *valor* precisa estar entre -128 ou +127 bytes de distância da instrução que segue o SJMP.

Exemplo:

- SJMP LABEL
- SJMP -8

Instrução - JMP

Operação: JMP

Função: Desvia o programa para DPTR+A

Sintaxe: JMP @A+DPTR

Descrição : JMP desvia incondicionalmente para o endereço representado pela soma de DPTR e do valor do acumulador.

Exemplo:

- JMP LABEL
- JMP @A+DPTR

Instruções LJMP, AJMP, SJMP e JMP

		Bytes	MC	Op1	Op2	Op3
LJMP	end16	3	2	02	MSB(end16)	LSB(end16)
AJMP	end11	2	2	[(MSB(end11))<<5]OU1	LSB(end11)	-
SJMP	rel	2	2	80	rel	-
JMP	@A+DPTR	1	2	73	-	-

Instruções de saltos incondicionais.

Portas Digitais do 8051:

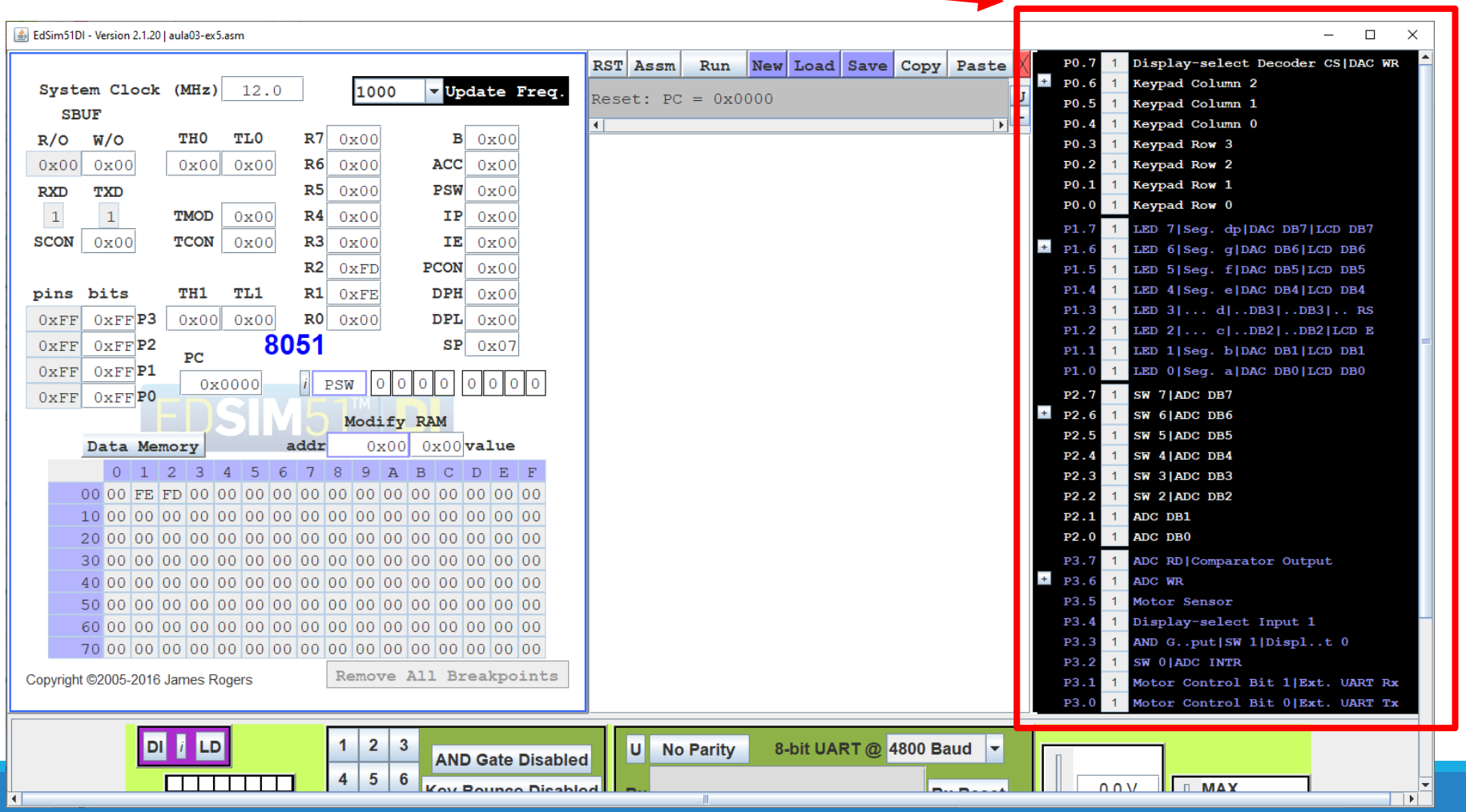
P1, P2, P3 e P4

Portas

- P0 → barramento de dados (D0, ...,D7) e parte baixa do barramento de endereços (A0, ...,A7);
- P1 → porta paralela livre.
- P2 → parte alta do barramento de endereços (A8, ...,A15);
- P3 → Sinais de controle e de comunicação.

EdiSim51 - Portas

A tela preta ao lado apresenta todos os dispositivos ligados aos 32 pinos das 4 portas.



Exercícios

Exercício 1

Teste o programa abaixo e verifique o comportamento na porta P1.

```
        mov A, #0FEh
ROT:
        rr      A
        mov   P1,  A
        ajmp  ROT
```

Exercício 2

Dado o programa abaixo, qual será o valor de R2 após a execução.

```
ØØØØ| MOV R1, #ØFFh
ØØØ2| DEC R1
ØØØ3| SJMP TESTE
      EXØ1:
ØØØ5| DEC R1
ØØØ6| MOV A, R1
ØØØ7| DEC A
ØØØ8| SJMP FIM
      TESTE:
ØØØA| INC R1
ØØØB| SJMP EXØ1
ØØØD| DEC R1
ØØØE| MOV A, R1
      FIM:
ØØØF| MOV R2, A
```

Exercício 3

Teste o programa abaixo e verifique o comportamento na porta P1.

```
        mov A, #0FEh
ROT:
        rl      A
        mov   P1,  A
        ajmp  ROT
```

Exercício 4

Esboce a forma-de-onda que a seguinte rotina gera pelo pino P1.0 e P1.1.

```
CLR P1.0
CLR P1.1
LB:
    INC P1
    SJMP LB
```

		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)

		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
ADDC A,	Rn	1	1	38+n	-
	end8	2	1	35	end8
	@Ri	1	1	36+i	-
	#dt8	2	1	34	dt8

		Bytes	MC	Op1	Op2
SUBB A,	Rn	1	1	98+n	-
	end8	2	1	95	end8
	@Ri	1	1	96+i	-
	#dt8	2	1	94	dt8

		Bytes	MC	Op
MUL	AB	1	4	A4
DIV	AB	1	4	84

		Bytes	MC	Op1	Op2	Op3
ANL A,	Rn	1	1	58+n	-	-
	end8	2	1	55	end8	-
	@Ri	1	1	56+i	-	-
	#dt8	2	1	54	dt8	-
ANL end8,	A	2	1	52	end8	-
	#dt8	3	2	53	end8	dt8

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

	Bytes	MC	Op
INC DPTR	1	2	A3

		Bytes	MC	Op
CLR	A	1	1	E4

		Bytes	MC	Op1	Op2	Op3
ORL A,	Rn	1	1	48+n	-	-
	end8	2	1	45	end8	-
	@Ri	1	1	46+i	-	-
	#dt8	2	1	44	dt8	-
ORL end8	A	2	1	42	end8	-
	#dt8	3	2	43	end8	dt8

		Bytes	MC	Op1	Op2	Op3
XRL A,	Rn	1	1	68+n	-	-
	end8	2	1	65	end8	-
	@Ri	1	1	66+i	-	-
	#dt8	2	1	64	dt8	-
XRL end8	A	2	1	62	end8	-
	#dt8	3	2	63	end8	dt8

Bibliografia

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

Gimenez, Salvador P. Microcontroladores 8051 - Teoria e Prática, Editora Érica, 2010.