

Arquitetura de Computadores

PROF. DR. ISAAC

8051 -Registadores

- 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.

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 MSC-51

Tipo	Quantidade
Aritméticas	24
Lógicas	25
Transferência (cópia) de Dados	28
Booleanas	17
Saltos	17

Instruções aritméticas: envolvem operações do tipo soma, subtração, multiplicação, divisão, incremento e decremento.

Instruções lógicas: fazem operações bit a bit com registradores e também rotações.

Instruções do MSC-51

Instruções de transferência (cópia) de dados: copiam bytes entre os diversos registradores e a RAM interna.

Instruções booleanas: essas instruções são denominadas booleanas porque trabalham com variáveis lógicas (variável de 1 bit). Como o próprio nome sugere, elas são talhadas para resolver expressões booleanas.

Instruções de desvio: desviam o fluxo de execução do programa, chamam subrotinas, fazem desvios condicionais e executam laços de repetição.

Instruções do 8051

Lógica	Aritmética	Memória	Outros
ANL ✓	ADD ✓	MOV ✓	NOP
ORL ✓	ADDC ✓	MOVC ✓	RET e RETI
XRL ✓	SUBB ✓	MOVB ✓	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 de Desvio

Instruções de Desvio:

Salto Condicionais

Instrução - JZ

Operação: JZ

Função: Desvia se o acumulador for zero

Sintaxe: JZ *endereço*

Descrição : JZ desvia para o endereço especificado em *endereço* se o valor armazenado no acumulador for zero.

Exemplo:

- JZ LABEL

Instrução - JNZ

Operação: JNZ

Função: Desvia se o acumulador não for zero

Sintaxe: JNZ *endereço*

Descrição : JNZ desvia para o endereço especificado em *endereço* se o valor armazenado no acumulador não for zero.

Exemplo:

- JNZ LABEL

Instrução - CJNE

Operação: CJNE

Função: Compara e desvia se não igual

Sintaxe: CJNE *operando1*, *operando2*, *endereço*

Descrição : CJNE compara o valor dos operando e salta para o endereço se os dois não forem iguais. O bite de Carry (**C**) é setado se o *operando1* for menor que o *operando2*, caso contrário, C=0.

Exemplo:

- CJNE @R1, #24H, LABEL
- CJNE A, #01H, LABEL
- CJNE A, 60h, LABEL
- CJNE R6, #12H, LABEL

Instruções JZ, JNZ e CJNE

Condição			Bytes	MC	Op1	Op2	Op3
A = 0	JZ	rel	2	2	60	rel	-
A ≠ 0	JNZ				70	rel	-
A ≠ (end8)	CJNE	A ,end8 ,rel	3	2	B5	end8	rel
A ≠ #dt8	CJNE	A	3	2	B4	dt	rel
Rn ≠ #dt8		Rn ,#dt ,rel			B8+n	dt	rel
@Ri ≠ #dt8		@Ri			B6+i	dt	rel

Instruções de saltos condicionais.

Instruções de Desvio:

Laços de Repetição

Instrução - DJNZ

Operação: DJNZ (decrement and jump if not zero).

Função: Decrementa e desvia se não for zero

Sintaxe: DJNZ *registrador*, *endereço*

Descrição : DJNZ decrementa o valor do *registrador*. Se o novo valor do registrador não for zero, o programa desviará para o endereço especificado em *endereço*. Essa instrução é análoga ao comando for que existe em C.

Exemplo:

- DJNZ 40h, LABEL
- DJNZ R6, LABEL

Instrução - DJNZ

			Bytes	MC	Op1	Op2	Op3
DJNZ	Rn	,rel	2	2	D8+n	rel	-
	end8		3	2	D5	end8	rel

Instruções para construir laços de programa.

Instruções de Desvio:

Desvios Baseados em Bits

Instrução - JC

Operação: JC

Função: Desvia se $\text{Carry} == 1$

Sintaxe: JC *endereço*

Descrição : JC desvia para o endereço especificado em *endereço* se o bit de $\text{Carryr}(C)$ estiver setado.

Exemplo:

- JC LABEL

Instrução - JNC

Operação: JNC

Função: Desvia se $\text{Carry} == 0$

Sintaxe: JNC *endereço*

Descrição : JNC desvia para o endereço especificado em *endereço* se o bit de carry for 0.

Exemplo:

- JNC LABEL

Instrução - JB

Operação: JB

Função: Salta se o bit estiver setado

Sintaxe: JB *bit*, *endereço*

Descrição : JB desvia para o endereço especificado em *endereço* se o bit indicado por *bit* estiver setado.

Exemplo:

- JB P1.2 LABEL

Instrução - JNB

Operação: JNB

Função: Salta se bit não setado

Sintaxe: JNB *bit*, *endereço*

Descrição : JNB desvia para o endereço indicado por *endereço* se o bit indicado em *bit não estiver setado*.

Exemplo:

- JNB P1.3, LABEL

Instrução - JBC

Operação: JBC

Função: Salta se bit estiver setado

Sintaxe: JBC *bit, endereço*

Descrição : JBC que faz um desvio se o bit especificado estiver em 1 e logo em seguida complementa o bit.

Exemplo:

- JBC P1.3, LABEL

Instruções JC, JNC, JB, JNB e JBC

		Bytes	MC	Op1	Op2	Op3
JC	rel	2	2	40	rel	-
JNC	rel	2	2	50	rel	-
JB	bit,rel	3	2	20	bit	rel
JNB				30	bit	rel
JBC				10	bit	rel

Instruções de desvios baseados em bits.

Instruções de Desvio:

Chamadas de Subrotinas

Chamadas de Subrotinas

As **subrotinas** são úteis para evitar a repetição de trechos de programas. Antes de efetivar o desvio para a subrotina, o processador armazena na pilha o endereço de retorno, para que possa recuperá-lo por ocasião do regresso.

Esse empilhamento de endereços permite que, de dentro de uma subrotina, se faça chamada para uma outra subrotina.

Instrução - ACALL

Operação: ACALL

Função: Chamada absoluta dentro do bloco de 2K

Sintaxe: ACALL *endereço*

Descrição : ACALL chama uma subrotina localizada no endereço indicado. Nenhuma flag é afetada.

Exemplo:

- ACALL LABEL

Instrução - LCALL

Operação: LCALL

Função: Long Call

Sintaxe: LCALL *endereço_codigo*

Descrição : LCALL chama uma subrotina do programa. O endereço da próxima instrução a ser executada é inserido na pilha antes que o PC desvie para a subrotina.

Exemplo:

- LCALL SUB1

Instrução - LCALL

		Bytes	MC	Op1	Op2	Op3
LCALL	end16	3	2	12	MSB(end16)	LSB(end16)
ACALL	end11	2	2	$[(\text{MSB}(\text{end11})) \ll 5] \text{OU} 11\text{H}$	LSB(end11)	-

Instruções de chamada de subrotinas.

Instruções de Desvio:

Retorno das Subrotinas

Instrução - RET

Operação: RET

Função: Retorna de uma subrotina

Sintaxe: RET

Descrição : RET é usado para retornar de uma subrotina chamada por LCALL ou ACALL. A execução do programa continua do endereço (2 bytes) restaurados da pilha. Primeiro o byte mais significativo é retirado, seguido pelo menos significativo.

Exemplo:

- RET

Instrução - RETI

Operação: RETI

Função: Retorna da Interrupção

Sintaxe: RETI

Descrição : RETI é usado para retornar de um serviço de interrupção.

Exemplo:

Instruções – RET e RETI

	Bytes	MC	Op
RET	1	2	22
RETI	1	2	32

Instruções de retorno de sub-rotinas.

Instruções de Desvio:

Nenhuma Operação

Instrução - NOP

Operação: NOP

Função: Nenhuma operação é executada

Sintaxe: NOP

Descrição: NOP usada para realizar delays. Nenhuma flag é afetada.

Exemplo:

- NOP

	Bytes	MC	Op1
NOP	1	1	00

Exercícios

Exercício 1

Exercício 1:

Qual o valor do bit “00h”, após:

```
setb  C  
jc    DESVIO  
mov   00h, C
```

```
DESVIO: cpl  C  
         mov  00h, C
```

Exercício 2

Exercício 2:

Qual o valor final de R1 após o seguinte programa:

```
mov  R0, #07h
mov  R1, #00h
djnz R0, CONTA
sjmp SAIDA
```

```
CONTA: inc  R1
```

```
SAIDA:  nop
```

Exercício 3

Exercício 3:

Crie uma subrotina que escreva zero em todas as posições da RAM interna, ou seja, do endereço 0 até o endereço 127 da RAM interna.

Resposta do exercício 3.

Exercício 3:

Crie uma subrotina que escreva zero em todas as posições da RAM interna, ou seja, do endereço 0 até o endereço 127 da RAM interna.

```
;Subrotina para zerar a RAM interna
;RETORNA: posições de 0 a 127 da RAM interna zeradas
;USA: A e R0
;
ZERAR: CLR  A           ;A = 0, valor a ser escrito
      MOV  R0,#127      ;R0 = endereço mais alto
ROT:   MOV  @R0,A        ;zera posição apontada por R0
      DJNZ R0,ROT       ;decrementa ponteiro e contador
      RET              ;retorna da subrotina
```

Exercício 4

Exercício 4:

Monte o código de máquina do programa abaixo:

```
COMP:  MOV  A,R7      ;coloca primeiro número em A (A = R7)
        CLR  C        ;zera o carry, pois seu valor é desconhecido
        SUBB A,R6     ;A - R6, se C = 0  $\Rightarrow$  A  $\geq$  R6 (não troca)
                        ;           se C = 1  $\Rightarrow$  A < R6 (troca)
        JNC  ROT1     ;se C = 0, finaliza
        XCH  A,R7     ;
        XCH  A,R6     ;troca conteúdos de R6 e R7
        XCH  A,R7     ;
ROT1:   SJMP  $        ;para em um laço infinito
```

Instrução	opcode
MOV A,R7	
CLR C	
SUBB A,R6	
JNC ROT1	


Instrução	opcode
XCH A,R7	
XCH A,R6	
XCH A,R7	
SJMP \$	

Resposta do exercício 4.

200	EF	COMP :	MOV	A, R7
201	C3		CLR	C
202	9E		SUBB	A, R6
203	50		JNC	ROT1
204	??			
205	CF		XCH	A, R7
206	CE		XCH	A, R6
207	CF		XCH	A, R7
208	80	ROT1 :	SJMP	\$
209	??			

Para calcularmos os valores dos desvio relativo, precisamos contar as posições de memória.

Resposta do exercício 4.

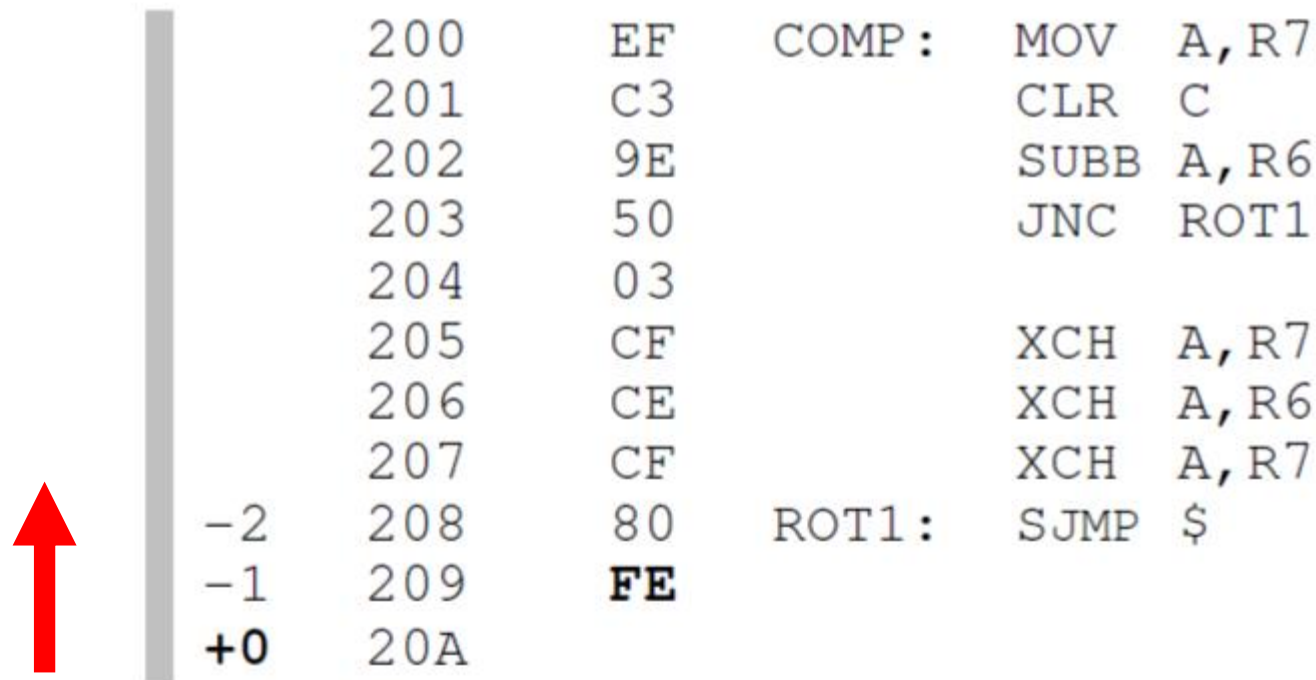


	200	EF	COMP :	MOV	A, R7
	201	C3		CLR	C
	202	9E		SUBB	A, R6
	203	50		JNC	ROT1
	204	03			
+0	205	CF		XCH	A, R7
+1	206	CE		XCH	A, R6
+2	207	CF		XCH	A, R7
+3	208	80	ROT1 :	SJMP	\$
	209	??			

Portanto avançando 3 posições de memória:

$$208H - 205H = 3.$$

Resposta do exercício 4.



	200	EF	COMP:	MOV	A, R7
	201	C3		CLR	C
	202	9E		SUBB	A, R6
	203	50		JNC	ROT1
	204	03			
	205	CF		XCH	A, R7
	206	CE		XCH	A, R6
	207	CF		XCH	A, R7
-2	208	80	ROT1:	SJMP	\$
-1	209	FE			
+0	20A				

Para a instrução **SJMP \$** desviar para si mesma, é necessário retroceder duas posições, ou **-2**, que em **complemento 2** corresponde ao byte **FEh**.

Exercício 5

Exercício 5:

Construir e testar programa-fonte em linguagem assembly que invoca uma sub-rotina que deve carregar (alocar) o valor EEh em 100 bytes consecutivos da RAM interna iniciando no endereço 20h.

Exercício 6

Exercício 6:

Compare dois números inteiros sem sinal que estão localizados em R7 e R6. Armazene o maior em R7 e o menor em R6. Termine o programa com um laço infinito.

Resposta do exercício 6.

A maneira mais fácil de comparar dois números inteiros sem sinal é através da subtração e a posterior interpretação do **borrow (carry)**.

$$X - Y \left| \begin{array}{l} \text{se } C = 0 \Rightarrow X \geq Y \\ \text{se } C = 1 \Rightarrow X < Y \end{array} \right.$$

```
COMP:  MOV  A,R7      ;coloca primeiro número em A (A = R7)
        CLR  C        ;zera o carry, pois seu valor é desconhecido
        SUBB A,R6     ;A - R6, se C = 0  $\Rightarrow$  A  $\geq$  R6 (não troca)
        ;           ;se C = 1  $\Rightarrow$  A < R6 (troca)
        JNC  ROT1     ;se C = 0, finaliza
        XCH  A,R7     ;
        XCH  A,R6     ;troca conteúdos de R6 e R7
        XCH  A,R7 ;
ROT1:   SJMP $        ;para em um laço infinito
```

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.