

EEL7030 - Microprocessadores



Prof. Raimes Moraes

GpqCom – EEL

UFSC

Manipulação de Bits

1) Setar bits

(Exemplo – bit 6 e bit 3):

```
MOV  A,00h
ORL  A,#01001000b
MOV  00h,A
```

3) Complementar bits

(Exemplo – bit 7 e bit 3):

```
MOV  A,07h
XRL  A,#10001000b
MOV  07h,A
```

2) Limpar bits

(Exemplo – bit 5 e bit 4):

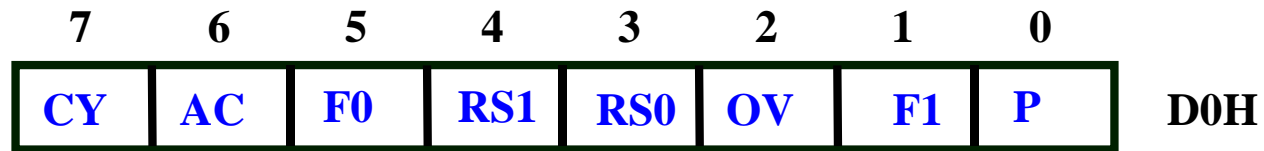
```
MOV  A,1Fh
ANL  A,#11001111b
MOV  1Fh,A
```

4) Testar bits

(Exemplo – bit 2):

```
MOV  A,03
ANL  A,#00000100b
JZ   LABEL
```

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.

Tomando Decisões – 1

Com dados previamente carregados nos registradores A e B:

- somar o conteúdo de ambos se $[A]=[B]$;
- subtrair se $[A]\neq[B]$;

	CJNE	A,B,SUBTRAI
	ADD	A,B
	JMP	\$
SUBTRAI:	CLR	C
	SUBB	A,B; $[A] \leftarrow [A] - [Cy] - [B]$
	JMP	\$

Tomando Decisões – 2

Com dados previamente carregados nos registradores A e B:

- somar o conteúdo de ambos se $[A] < [B]$;
- subtrair se $[A] \geq [B]$;

	CJNE	A,B,SOMA
SUBTRAI :	SUBB	A,B
	JMP	\$
SOMA:	JNC	SUBTRAI
	ADD	A,B
	JMP	\$

Tomando Decisões – 3

Com dados previamente carregados nos registradores A e B:

- somar o conteúdo de ambos se $[A] \geq [B]$;
- subtrair se $[A] < [B]$;

	CJNE	A,B, SUBTRAI
SOMA:	ADD	A,B
	JMP	\$
SUBTRAI:	JNC	SOMA
	CLR	C
	SUBB	A,B
	JMP	\$

Tomando Decisões – 4

Com dados previamente carregados nos registradores
A e B:

- somar o conteúdo de ambos se $[A] \leq [B]$;
- subtrair se $[A] > [B]$;

SOMA:

```
CJNE    A,B, SUBTRAI
ADD     A,B
JMP     $
```

SUBTRAI:

```
JC      SOMA
SUBB    A,B
JMP     $
```

Tomando Decisões – 5

Com dados previamente carregados nos registradores A e B, sendo que
[A]≠[B]=>

- somar o conteúdo de ambos se [A]>[B];
- caso [A]<[B], somar 6 a [A]

CLR	C
PUSH	ACC
SUBB	A,B
POP	ACC
JC	SOMA6
ADD	A,B
JMP	\$
ADD	A,#6
JMP	\$

SOMA6:

Multiplicação por 10 no 8051

MOV A,#dado

MOV B,#10

MUL AB

OBS:

- **Multiplica valores inteiros de 8 bits sem sinal**
- **LSB do resultado de 16-bit vai para o acumulador**
- **MSB do resultado de 16-bit vai para B**
- **Se o produto for maior que 255 (0FFH), o flag de *overflow* é setado.**

Representação de nros negativos em formato binário

- n bits $\Rightarrow 2^n$ valores sem sinal: 0 a (2^n-1) ;
- **Ex: 8 bits $\Rightarrow 2^8$ valores sem sinal: 0 a 255;**

Complemento de 1: nro. negativo é o complemento da representação binária do nro. positivo: $-((2^{n-1})-1)$ a $((2^{n-1})-1)$

- **Ex: 8 bits $\Rightarrow -((2^7)-1)$ a $((2^7)-1) \Rightarrow -127$ (80H) a 127 (7FH)**
- Duas representações para o valor 0 (00H e FFh).

- **Complemento de 2:** nro. negativo é o complemento da representação binária do nro. positivo somado a 1 : $-(2^{n-1})$ a $((2^{n-1})-1)$.

- **Ex: 8 bits $\Rightarrow -(2^7)$ a $((2^7)-1) \Rightarrow -128$ (80H) a 127 (7FH)**

Complemento de 1 para n=3

Nros. Positivos		Nros. Negativos	
0	000	0	111
1	001	-1	110
2	010	-2	101
3	011	-3	100

Ex: 3 bits => -((2²)-1) a ((2²)-1)

Aritmética em Complemento de 1

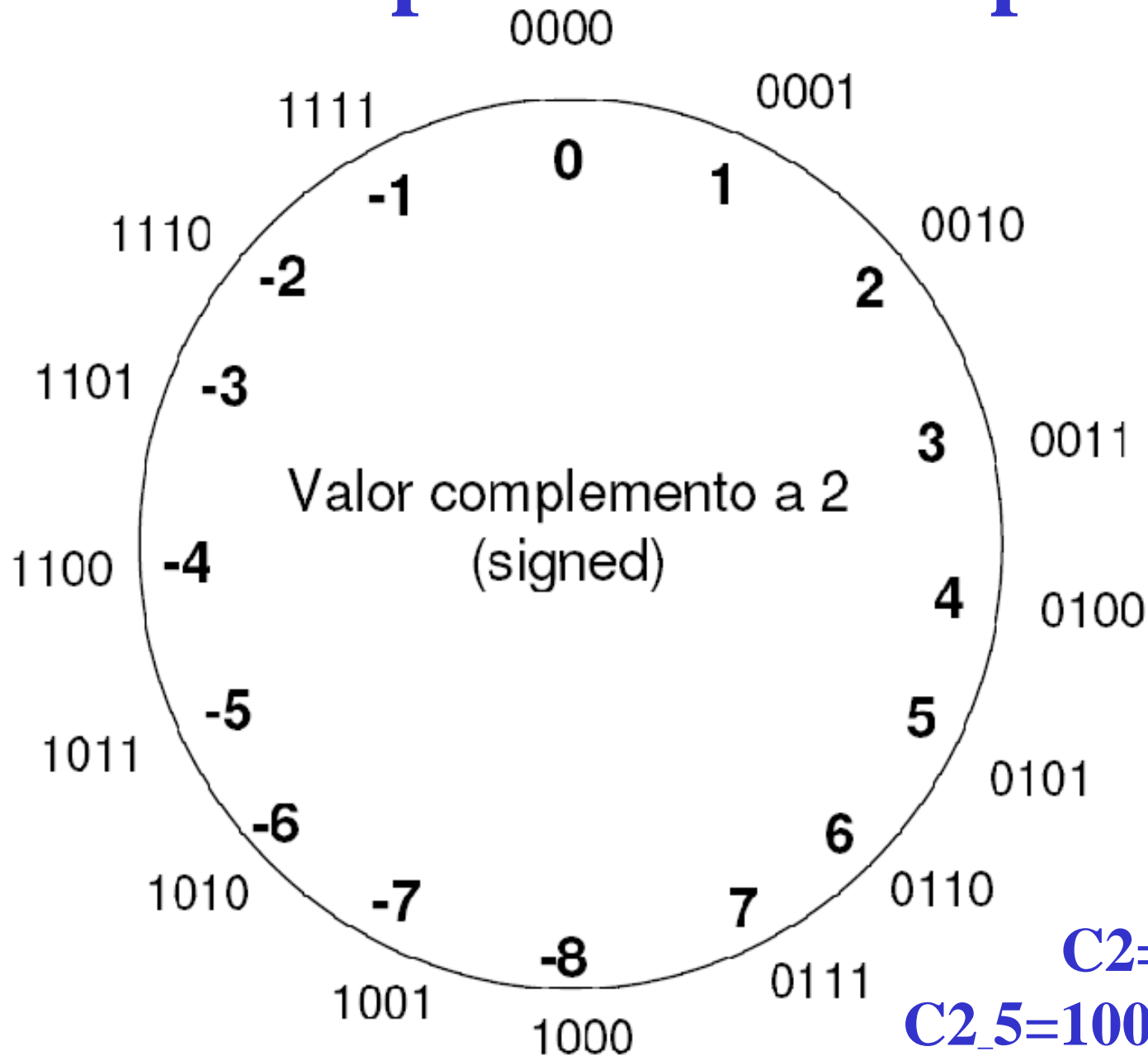
- Possui duas representações para 0.
 - 1) Somar bit a bit, inclusive o bit de sinal.
 - 2) Avaliar "vai-um":
 - a) Ausente para o bit de sinal ou *carry*: fim;
 - b) "vai-um" apenas para o bit de sinal: resultado excede capacidade.
 - c) "vai-um" para *carry*: soma-se o "vai-um" ao resultado da soma;
 - c.1) se ocorrer "vai-um" após c: nro de “vai-uns” total for par (para bit de sinal e carry), correto, de outra forma, *overflow*;

Complemento de 1

D	H
-3	100
+2	010
-1	110

D	H
-2	101
-1	110
	1011
	1
-3	100

Complemento de 2 para n=4



$$C2 = 2^n - \text{nro}$$

$$C2_5 = 10000 - 0101 = 1011$$

Complemento de 2 para n=3

Nros. Positivos		Nros. Negativos	
0	000	-1	111
1	001	-2	110
2	010	-3	101
3	011	-4	100

Ex: 3 bits => $-(2^2)$ a $((2^2)-1)$

Complemento de 2

D	H
-3	101
+2	010
-1	111

D	H
-2	110
-1	111
	1101
-3	101

OBS: Se houver carry sem "vai-um" para o bit de sinal:
overflow

Exemplo de Adição BCD

Somar os operandos 1897 a 2905 em BCD

Exemplo de Adição BCD

$$\begin{array}{r} 0001 \quad 1000 \\ + 0010 \quad \underline{1001} \end{array}$$

$$\begin{array}{r} 1001 \\ \underline{0000} \end{array}$$

$$\begin{array}{r} 0111 \\ \underline{0101} \\ 1100 \text{ (12 > 9)} \\ + 0110 \text{ (6)} \\ \hline \end{array}$$

Exemplo de Adição BCD

1

$$\begin{array}{r} 0001 \quad 1000 \\ + \underline{0010} \quad \underline{1001} \end{array}$$

$$\begin{array}{r} 1001 \\ \underline{0000} \end{array}$$

$$\begin{array}{r} 0111 \\ \underline{0101} \\ 1100 \text{ (12 > 9)} \\ + \underline{0110 \text{ (6)}} \\ 0010 \end{array}$$

Exemplo de Adição BCD

		1	1
	0001	1000	
+	<u>0010</u>	<u>1001</u>	
		1001	0111
		<u>0000</u>	<u>0101</u>
		1010	1100
		(10 > 9)	(12 > 9)
		+0110(6)	+0110 (6)
		<u>0000</u>	<u>0010</u>

Exemplo de Adição BCD

	1	1		1		
	0001	1000		1001		0111
+	<u>0010</u>	<u>1001</u>		<u>0000</u>		<u>0101</u>
		10010(cy)		1010(10>9)		1100 (12>9)
	+ 0110(6)			+0110(6)		+0110 (6)
	<u> </u>			<u> </u>		<u> </u>
	1000			0000		0010

Exemplo de Adição BCD

	1	1		1		
	0001	1000		1001		0111
+	<u>0010</u>	<u>1001</u>		<u>0000</u>		<u>0101</u>
	0100	10010	(18>9)	1010	(10>9)	1100 (12>9)
		+ 0110(6)		+0110(6)		+0110 (6)
		<u>1000</u>		<u>0000</u>		<u>0010</u>
	0100	1000		0000		0010
	4	8		0		2

$$1897_{\text{BCD}} + 2905_{\text{BCD}} = 4802_{\text{BCD}}$$

ASCII American Standard Code for Information Interchange (ASCII)

**ASCII contém 128 caracteres; 33 são caracteres
de controle; 94 caracteres e espaço.**

ASCII

HEX	DEC	CHR		HEX	DEC	CHR		HEX	DEC	CHR		HEX	DEC	CHR
00	0	NUL		20	32	SPC		40	64	@		60	96	`
01	1	SOH		21	33	!		41	65	A		61	97	a
02	2	STX		22	34	"		42	66	B		62	98	b
03	3	ETX		23	35	#		43	67	C		63	99	c
04	4	EOT		24	36	\$		44	68	D		64	100	d
05	5	ENQ		25	37	%		45	69	E		65	101	e
06	6	ACK		26	38	&		46	70	F		66	102	f
07	7	BEL		27	39	'		47	71	G		67	103	g
08	8	BS		28	40	(48	72	H		68	104	h
09	9	HT		29	41)		49	73	I		69	105	i
0A	10	LF		2A	42	*		4A	74	J		6A	106	j
0B	11	VT		2B	43	+		4B	75	K		6B	107	k
0C	12	FF		2C	44	,		4C	76	L		6C	108	l
0D	13	CR		2D	45	-		4D	77	M		6D	109	m
0E	14	SO		2E	46	.		4E	78	N		6E	110	n
0F	15	SI		2F	47	/		4F	79	O		6F	111	o
10	16	DLE		30	48	0		50	80	P		70	112	p
11	17	DC1		31	49	1		51	81	Q		71	113	q
12	18	DC2		32	50	2		52	82	R		72	114	r
13	19	DC3		33	51	3		53	83	S		73	115	s
14	20	DC4		34	52	4		54	84	T		74	116	t
15	21	NAK		35	53	5		55	85	U		75	117	u
16	22	SYN		36	54	6		56	86	V		76	118	v
17	23	ETB		37	55	7		57	87	W		77	119	w
18	24	CAN		38	56	8		58	88	X		78	120	x
19	25	EM		39	57	9		59	89	Y		79	121	y
1A	26	SUB		3A	58	:		5A	90	Z		7A	122	z
1B	27	ESC		3B	59	;		5B	91	[7B	123	{
1C	28	FS		3C	60	<		5C	92	\		7C	124	
1D	29	GS		3D	61	=		5D	93]		7D	125	}
1E	30	RS		3E	62	>		5E	94	^		7E	126	~
1F	31	US		3F	63	?		5F	95	_		7F	127	DEL