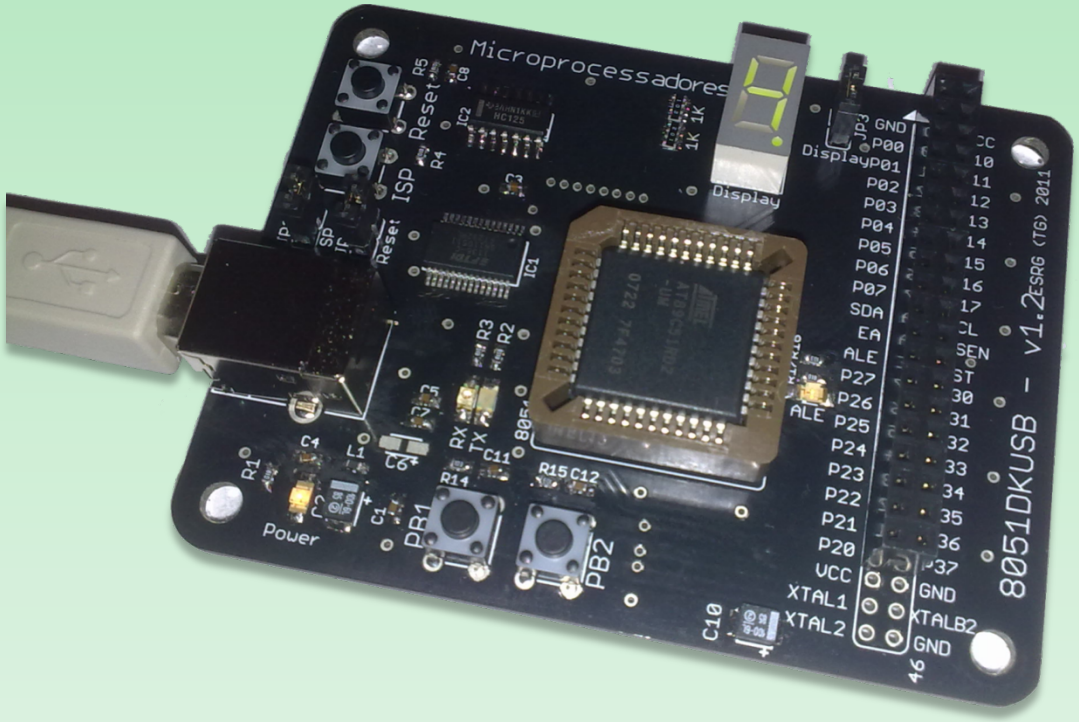
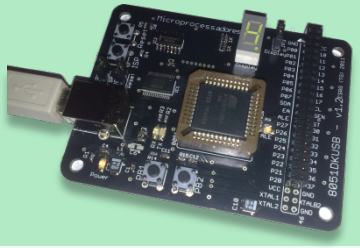
Mestrado Integrado em Eng. Electrónica Industrial e Computadores



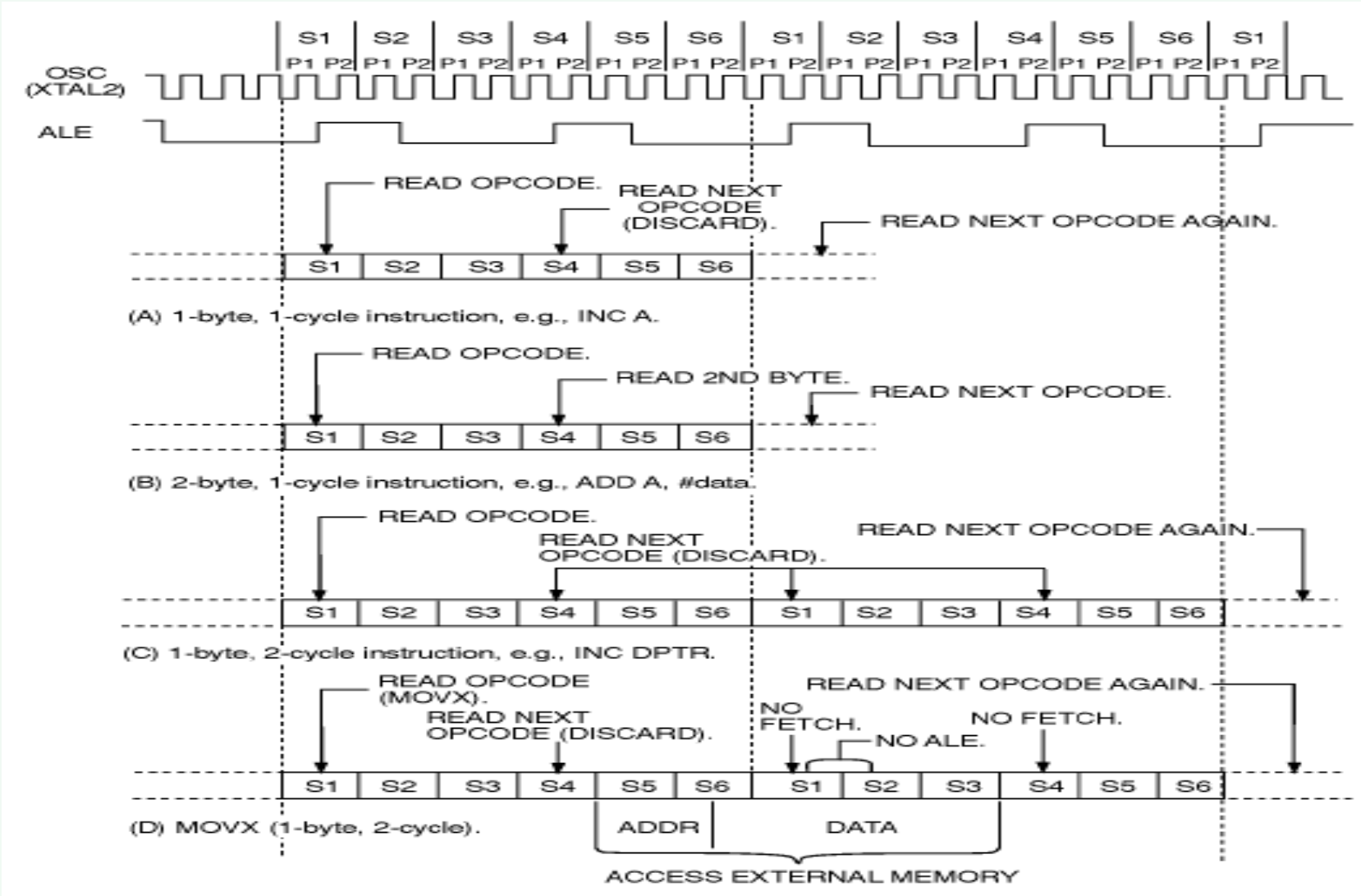
Instruction Set

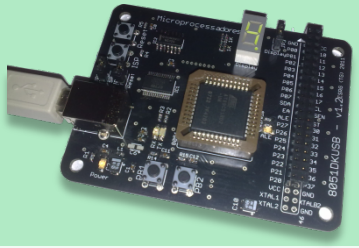
Microcontroladores

2º Ano – A06



Fetch do opcode

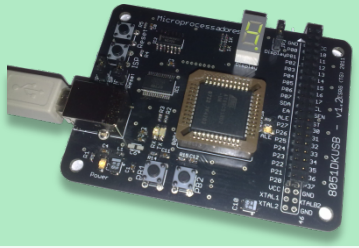




Directivas assembly básicas

- CSEG AT X
  - coloca a próxima instrução no endereço X da memória de código/programa (ROM);
  - ex: CSEG AT 0H
- END
  - Indica ao assembler que o ficheiro fonte terminou.
- “Etiquetas”
  - Em vez de calcularmos o endereço de cada salto, podemos utilizar etiquetas ou *labels* para marcar esses endereços:
  - ex:

```
        CSEG AT 0H
        JMP MAIN ;dependendo da distância do salto, o assembler escolhe a instrução de salto ideal
        ....
MAIN:
        MOV R0,#25
```



Características eléctricas

- Portos de Entrada/Saída

  V_{OL} – tensão de saída nível lógico baixo

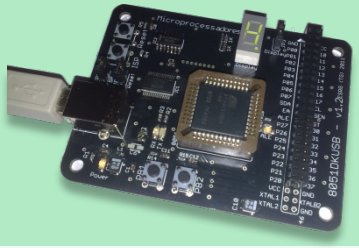
		V_{CC} = 4.5V to 5.5V	V_{CC} = 4.5V to 5.5V
0.3	V	I_{OL} = 100 µA[4]	I_{OL} = 200 µA[4]
0.45	V	I_{OL} = 1.6 mA[4]	I_{OL} = 3.2 mA[4]
1.0	V	I_{OL} = 3.5 mA[4]	I_{OL} = 7.0 mA[4]

  V_{OH} – tensão de saída nível lógico alto

		V_{CC} = 5V ± 10%	V_{CC} = 5V ± 10%
V_{CC} - 0.3	V	I_{OH} = -10 µA	I_{OH} = -200 µA
V_{CC} - 0.7	V	I_{OH} = -30 µA	I_{OH} = -3.2 mA
V_{CC} - 1.5	V	I_{OH} = -60 µA	I_{OH} = -7.0 mA

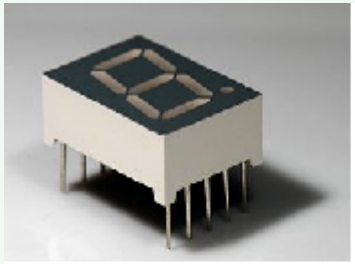
  Em condições de regime permanente (não transitórias), I_{OL} deve ser externamente limitada de modo a garantir:

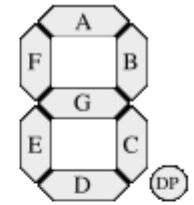
  - I_{OL} máxima por pino de porto: 10mA
  - I_{OL} máxima por porto (8-bit): 15mA (P1, P2 e P3) e 26mA (P0)
  - I_{OL} total máxima para todos os pinos de saída: 71mA

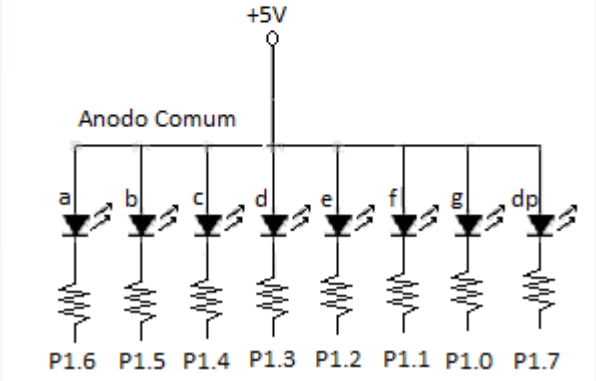


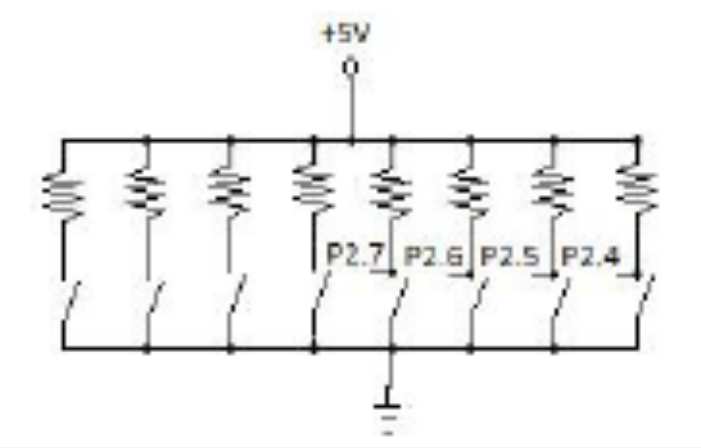
Laboratórios - Interface

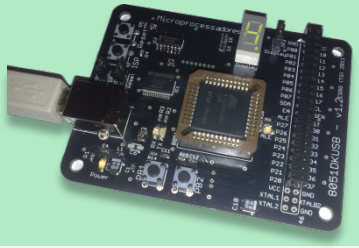
- Problema:
  - Com base no valor de 4 pinos de entrada do porto 2 (P2.4 a P2.7), ou seja, do *nibble* (4-bit) mais significativo de P2, escrever no *display* de 7-segmentos o caracter hexadecimal correspondente ao valor do *nibble*.
- Hardware:



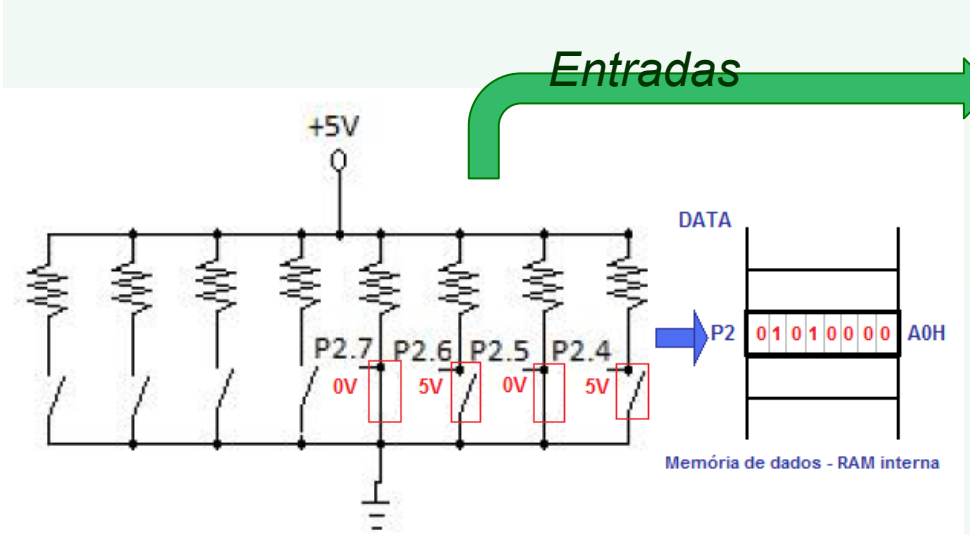


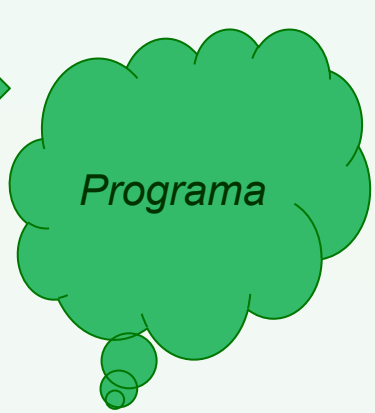


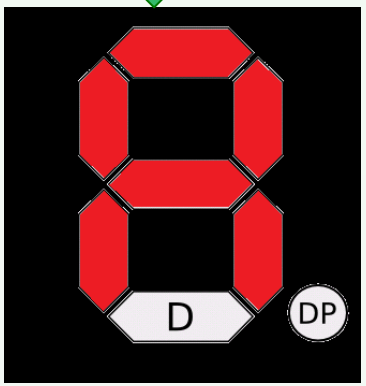




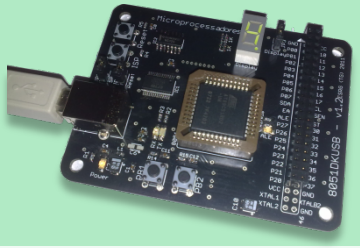
Exemplo



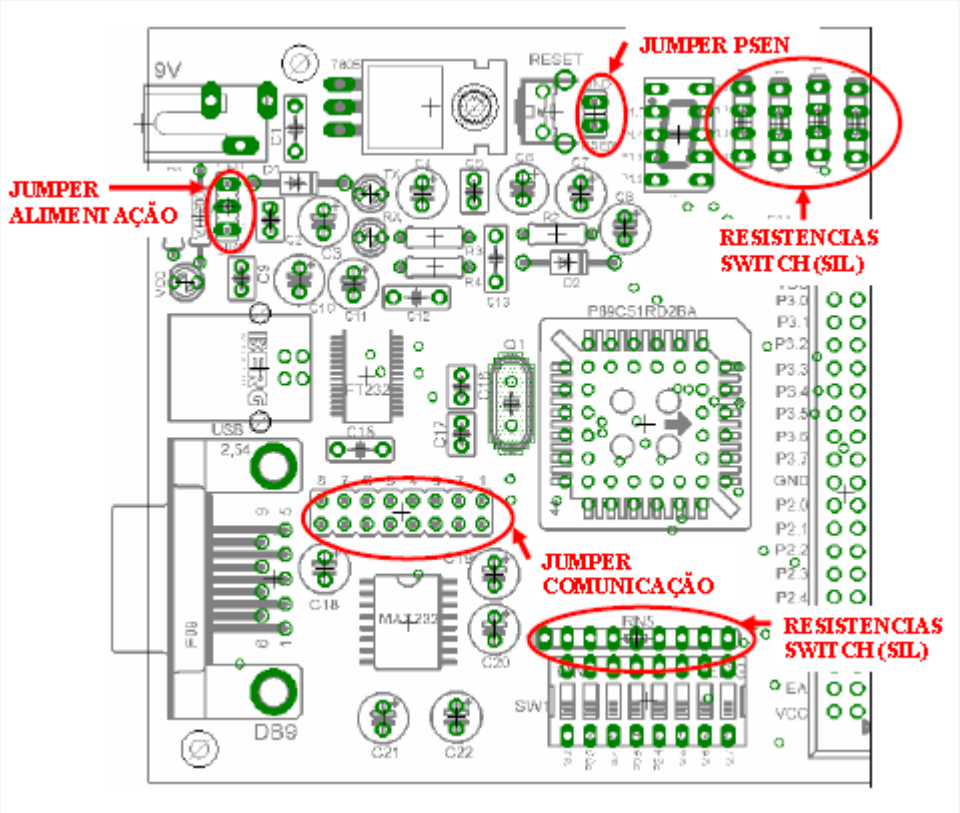


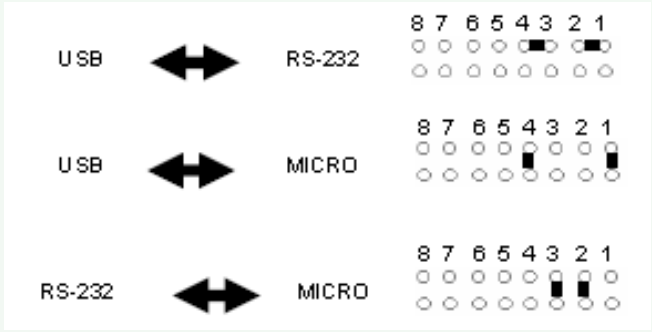


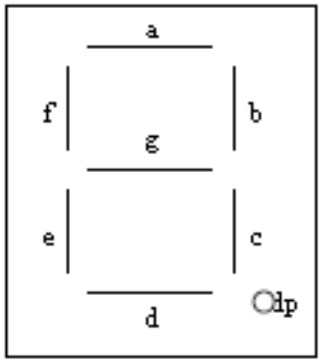
- O microcontrolador automaticamente coloca no endereço A0H da memória de dados interna, a representação binária das tensões lidas nos pinos de entrada P2.7 a P2.4;
- No exemplo os bits P2.7 a P2.4 foram definidos como entradas digitais e os bits P2.3 a P2.0 como saídas digitais, usando a instrução: MOV P2,#0F0H;
- Reparar que os bits foram lidos em lógica negativa devido à configuração do hardware;
- O programa lê os bits do porto E/S para o acumulador: MOV A,P2
- O programa realiza operações sobre o acumulador;
- Após as verificações e conversões o programa tem no acumulador os bits (bit a 0 LED liga, bit a 1 LED desliga) a colocar nos pinos do porto de E/S (P1);
- MOV P1,A ;acendem-se os segmentos desejados.



KIT8051- Esquema de Ligações

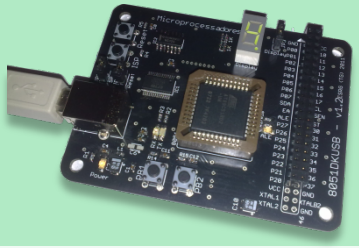






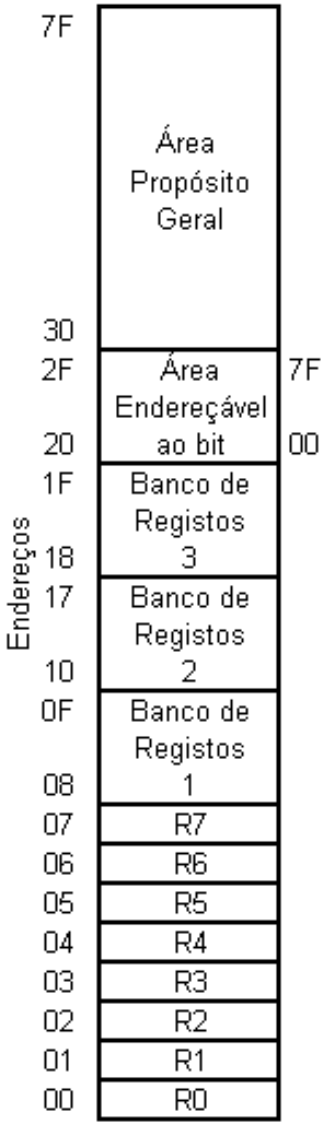
a	**P1.3**
b	**P1.2**
c	**P1.0**
d	**P1.6**
e	**P1.5**
f	**P1.7**
g	**P1.4**
dp	**P1.1**

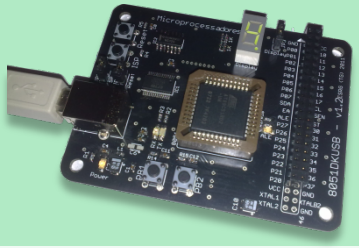
*Escrever **0** em P1.3 liga o segmento **a**.*
Escrever 1 desliga



DATA (00h:7Fh)

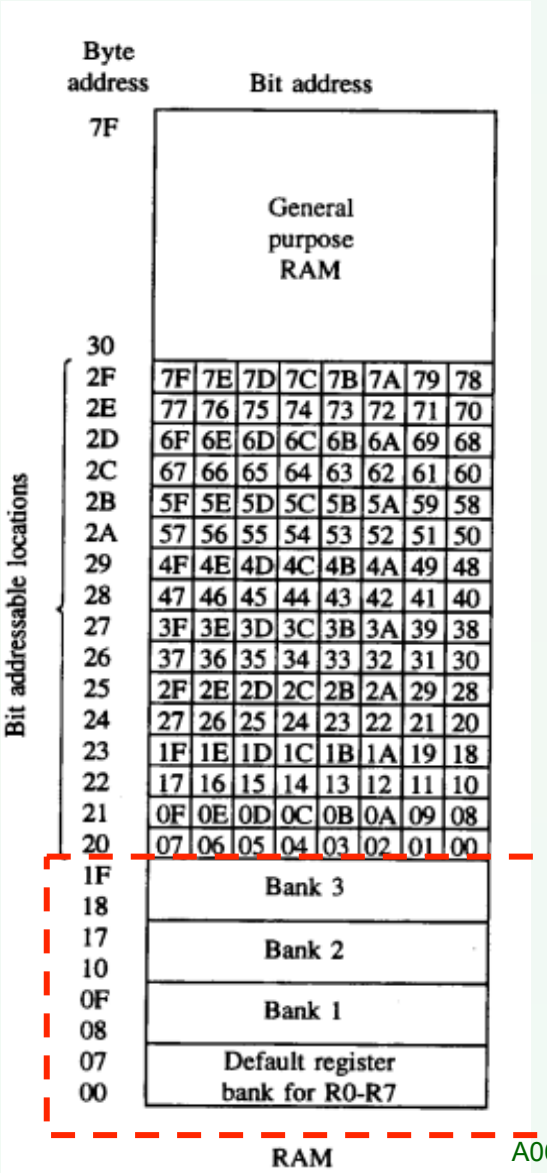
- Registos de trabalho (R0..R7)
  - Banco 0: 00h:07h
  - Banco 1: 08h:0Fh
  - Banco 2: 10h:17h
  - Banco 3: 18h:1Fh
- Selecção de banco
  - Bits RS0 (bit 3) e RS1 (bit 4) do registo PSW
- Zona endereçável ao bit
  - DATA - 20h:2Fh
  - Processamento booleano
- Uso Geral
  - DATA – 30h:7Fh
- Endereçamento directo e indirecto

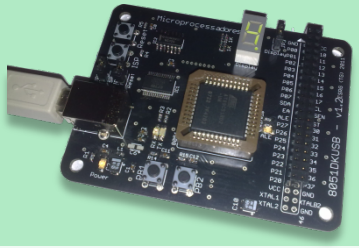




Modos de endereçamento

- Existem 4 bancos de registos, estando apenas um activo num dado instante
  - Os bits 3 e 4 (RS0 e RS1) do registo PSW (endereço 0D0h do SFR) especificam qual o banco que está activo;
  - *MOV PSW, #000 11 000B* activa o banco 3
- Os bancos de registo ocupam os primeiros 32 *bytes* da RAM interna;





DATA (20h:2Fh) - BIT

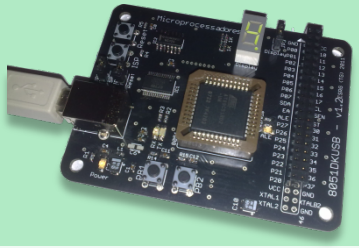
- Zona endereçável ao bit
  - Operações booleanas;
  - Nota:

    esta zona da DATA é endereçável ao byte (16 bytes)
    e ao bit (128 bits)

  - MOV A,2AH ;Acumulador armazena bits 50h:57h
  - SETB 2AH ;Bit 2Ah é colocado a um

  - Instruções:
    - MOV C,bit# MOV bit#,C
    - CLR bit# SETB bit#
    - CPL bit#
    - JB bit#,addr JNB bit#,addr

End.	nº do bit							
	7	6	5	4	3	2	1	0
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00

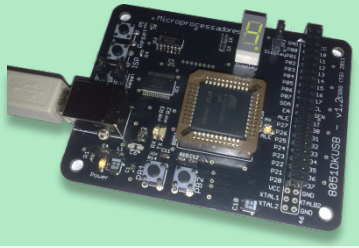


DATA (80h:FFh)

- Duas zonas distintas nos mesmos endereços

Selecção da área de memória SFR ou IDATA feita pelo modo de endereçamento:

- Endereçamento Directo - SFR
  - Acesso aos periféricos internos através dos SFRs
  - Alguns dos SFRs são endereçáveis ao bit
- Endereçamento Indirecto - IDATA
  - Área de propósito geral
  - Nem todas as versões têm esta memória implementada
  - Útil para colocar a stack

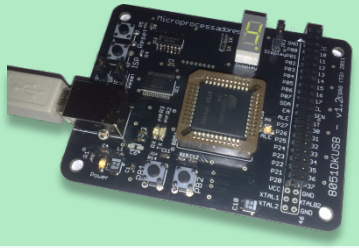


SFR – DATA – 80h:FFh

- Special Function Registers – SFRs
- Registos Aritméticos:
  - Acumulador: A ou Acc
  - Registo B
  - Registo de Estados: PSW
- Apontadores:
  - Apontador da Stack
  - Apontador de Dados: DPTR
    - DPH – MSB de DPTR
    - DPL – LSB de DPTR
- Portos de E/S
  - P0, P1, P2 e P3
- Sistema de Interrupções
  - IE e IP
- Comunicações série RS-232
  - SCON e SBUF
- Power Managment:
  - PCON

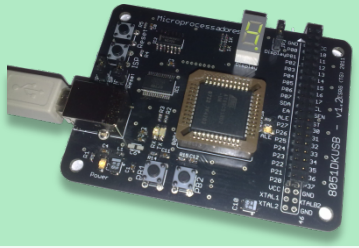
F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								DF
D0	PSW							D7
C8								CF
C0								C7
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TL0	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPH				PCON 87

Temporizadores/Contadores
Portos Entrada/Saída Digitais
Controlo de Interrupções
Registos de Trabalho (A e B) Flags de Estado (PSW)
Modos de energia
Apontador para a Stack



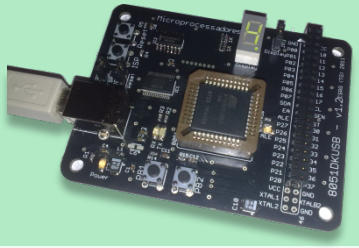
Instruções e operandos

- As instruções do processador são armazenadas na forma de "*opcodes*" na memória de programa (CODE ou XCODE) do processador;
- A cada instrução está associado um "*opcode*" único;
- O PC tem o endereço da memória de programa onde está armazenado o próximo "*opcode*" a ser executado;
- Esse "*opcode*" é lido para o IR e processado pelo CPU;
- O 8051 lê sempre pelo menos um byte da memória de programa interna. No caso da memória externa lê sempre 2 byte.
  - O tamanho das instruções pode ser de 1-, 2- ou 3-byte;



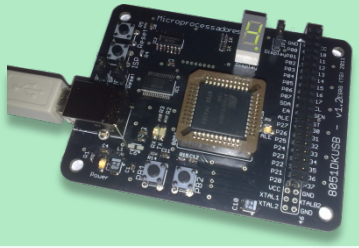
Instruções e operandos

- As instruções do processador operam sobre dados (operandos);
- Os operandos podem ser de entrada (dados de entrada: *input*) ou de saída (resultado: *output*);
- O segundo (e em alguns casos o 3º) byte lido da memória de programa corresponde ao operando de entrada da instrução;
- A versatilidade na programação de um processador encontra-se na diversidade de instruções (operações) e na forma como os operandos de entrada e de saída são obtidos (modos de endereçamento).



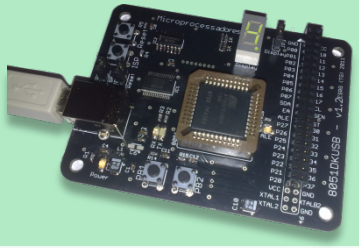
Conjunto de Instruções

- **Tipos de instrução**
  - O 8051 possui 5 grupos funcionais de instruções:
    - Instruções aritméticas
    - Instruções lógicas
    - Instruções de transferência de dados
    - Instruções para a manipulação de variáveis booleanas
    - Instruções de controlo de fluxo de execução



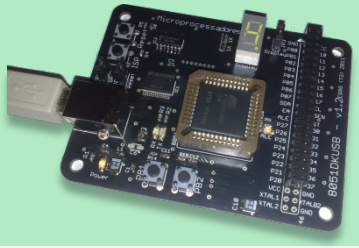
Instruções Aritméticas

Mnemonic	Operation	Addressing Modes				Execution Time in X1 Mode @12 MHz (µs)
		Dir	Ind	Reg	Imm	
ADD A, <byte>	A = A + <byte>	X	X	X	X	
ADDC A, <byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A, <byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A – 1	Accumulator only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B × A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1



Conjunto de Instruções

- **Instruções aritméticas**
  - A maioria das instruções são executadas num único ciclo máquina, exceptuando:
    - INC DPTR (2 ciclos)
    - MUL AB (4 ciclos)
    - DIV AB (4 ciclos)
  - Ao usá-las deve-se prestar atenção à forma como:
    - A instrução afecta o registo de estado (PSW);
    - O comportamento da instrução é afectado pelo estado do registo PSW.



Conjunto de Instruções

- **Instruções aritméticas**

Exemplos:

R7 = R7 – R6

```
MOV     A, R7
CLR     C
SUBB    A, R6
MOV     R7, A
```

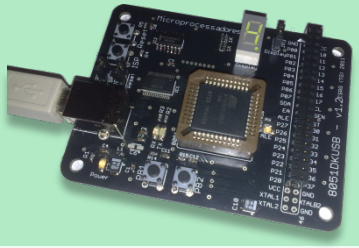
SUBB A, Rn

Bytes:	**1**
Cycles:	**1**
Encoding:	**10011rrr**
Operation	**(A) ←(A) – (C) – (Rn)**

Deve ser explicitamente inicializado a zero caso o valor seja desconhecido

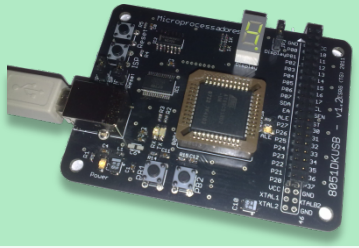
Decrementar DPTR

```
DEC     DPL             ; decrementa o LSB do DPTR
MOV     R7, DPL         ; guarda o resultado em R7
CJNE    R7, #0FFH, SKIP ; verifica se houve underflow para 0FFH
DEC     DPH             ; caso contrário decrementa o MSB do DPTR
SKIP: ...
```



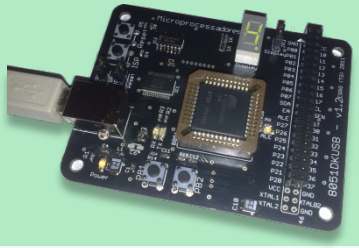
Instruções Lógicas

Mnemonic	Operation	Addressing Modes				Execution Time @ 12MHz (µs)
		Dir	Ind	Reg	Imm	
ANL A, <byte>	A = A AND <byte>	X	X	X	X	1
ANL <byte>, A	<byte> = <byte> AND A	X				1
ANL <byte>, # data	<byte> = <byte> AND # data	X				2
ORL A, <byte>	A = A OR <byte>	X	X	X	X	1
ORL <byte>, A	<byte> = <byte> OR A	X				1
ORL <byte>, # data	<byte> = <byte> OR # data	X				2
XRL A, <byte>	A = A XOR <byte>	X	X	X	X	1
XRL <byte>, A	<byte> = <byte> XOR A	X				1
XRL <byte>, # data	<byte> = <byte> XOR # data	X				2
CLR A	A = 00H	Accumulator only				1
CLP A	A = NOT A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1



Conjunto de Instruções

- **Instruções lógicas**
  - Todas as instruções lógicas que usam o acumulador são executados num único ciclo máquina;
    - As restantes são executados em 2 ciclos máquina
  - Todas as instruções lógicas podem manipular directamente (endereçamento directo) qualquer *byte* da memória interna;
  - As instruções e-lógico, ou-lógico, ou-exclusivo-lógico e negação-lógica podem manipular tanto *bytes* como *bits;*



Conjunto de Instruções

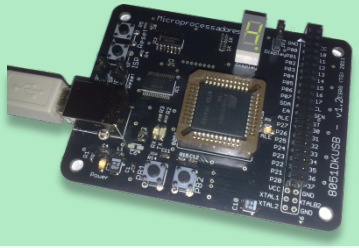
- **Instruções lógicas**

Exemplo:

– Converter binário em A para BCD

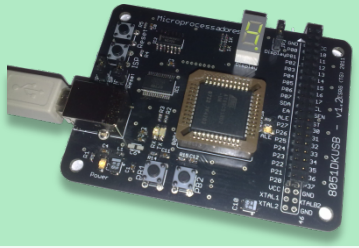
$< 100_{10}$

```
MOV   B, #10   ; carregar B com o divisor  da base decimal
DIV   AB       ; dividir o número no acumulador por 10
               ; deixa em A a parte inteira da divisão e em B o
               resto
SWAP  A        ; mover o dígito das dezenas para o nibble mais
                   ; significativo do acumulador
ANL   A,#0F0H  ; linha seguinte podia ser ORL?
ADD   A, B     ; adiciona ao valor no acumulador (dígito das
                   ; dezenas) o dígito das unidades
```



Instruções Transferência de Dados

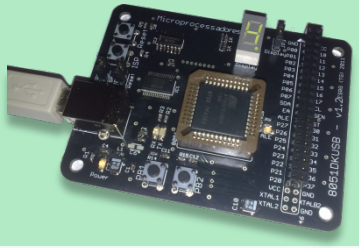
Mnemonic	Operation	Addressing Modes				Execution Time @ 12MHz (µs)
		Dir	Ind	Reg	Imm	
MOV A, <src>	A = <src>	X	X	X	X	1
MOV <dest>, A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR, # data 16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP: MOV "@SP", <scr>	X				2
POP <dest>	MOV <dest>, "@SP": DEC SP	X				2
XCH A, <byte>	ACC and <byte> Exchange Data	X	X	X		1
XCHD A, @Ri	ACC and @ Ri exchange low nibbles		X			1



Transferência de Dados

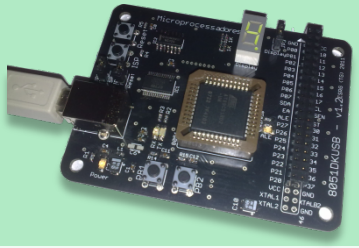
Address Width	Mnemonic	Operation	Execution Time @ 12MHz (µs)
8 bits	MOVX A, @Ri	Read external RAM @ Ri	2
8 bits	MOVX @ Ri, A	Write external RAM @ Ri	2
16 bits	MOVX A, @ DPTR	Read external RAM @ DPTR	2
16 bits	MOVX @ DPTR, A	Write external RAM @ DPTR	2

Mnemonic	Operation	Execution Time @ 12MHz (µs)
MOVC A, @A + DPTR	Read Pgm Memory at (A + DPTR)	2
MOVC A, @A + PC	Read Pgm Memory at (A + PC)	2



Conjunto de Instruções

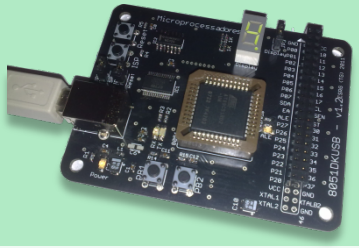
- **Instruções de transferência de dados**
  - Toda a movimentação de dados no interior da memória interna são executados em 1 ou 2 ciclos máquina.
  - A movimentação de dados entre a memória interna e externa realiza-se através do endereçamento indirecto.
  - Todas as movimentações que operam na memória externa são executadas em 2 ciclos máquina.
    - Usam o acumulador como fonte ou destino
  - O *strobe* de leitura/escrita (/RD e /WD) são activados apenas durante a execução da instrução MOVX.



Conjunto de Instruções

- **Instruções de transferência de dados**
  - Exemplo: Movimentação de dados a partir da memória externa
    - Fonte de dados: *endereços 10F4H e 10F5H (memória externa)*.
    - Destino dos dados: *registos R6 e R7, respectivamente*.

```
MOV     DPTR, #10F4H    ; inicializa o apontador de dados de 16 bit com
                        ; o menor endereço fonte.
MOVX    A, @DPTR        ; lê o dado apontado pelo apontador de dados de
                            ; 16 bits e coloca-o no acumulador
MOV     R6, A           ; transfere o dado lido do acumulador para R6
INC     DPTR            ; aponta para o próximo endereço fonte (10F5H)
MOVX    A, @DPTR        ; lê o dado apontado pelo apontador de dados de
                            ; 16 bits e coloca-o no acumulador
MOV     R7, A           ; transfere o 2º dado lido do acumulador para R7
```



Conjunto de Instruções

- **Instruções de transferência de dados**
  - A instrução MOV R1,R2 não existe!
  - Uma maneira de a implementar seria:

```
MOV     A,R2
MOV     R1,A
```

  - No entanto os registos estão implementados em memória. Portanto podemos:

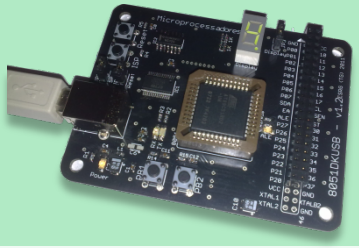
```
MOV     R1,2        ; 2 é o endereço de R2 (banco 0) na memória RAM interna
```

  - Para não estarmos dependentes do banco:

```
USING   0           ; indicar qual o banco que estamos a usar
MOV     R1,AR2      ; AR2 é o endereço de R2 na memória RAM interna no
                    ; banco seleccionado pela directiva USING
```

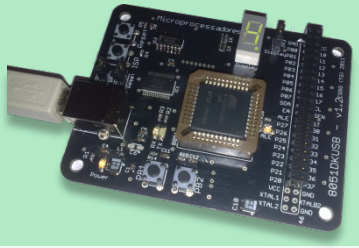
  - Ou seja, podemos sempre utilizar o endereço directo do registo com que pretendemos utilizar.

```
ACC     - Acumulador
```



Instruções Booleanas

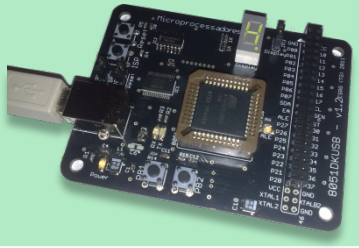
Mnemonic	Operation	Execution Time @ 12MHz (µs)
ANL C,bit	C = C AND bit	2
ANL C,/bit	C = C AND (NOT bit)	2
ORL C,bit	C = C OR bit	2
ORL C,/bit	C = C OR (NOT bit)	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = NOT C	1
CPL bit	bit = NOT bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1 ; CLR bit	2



Conjunto de Instruções

- **Instruções Booleanas**
  - Nas instruções booleanas, o *carry* funciona como o “Acumulador”;
  - Quando é utilizado um bit da memória de dados, esse bit reside na área endereçável ao bit (20h a 2Fh - se tiver um endereço entre 00h e 7Fh) ou pertence a um SFR endereçável ao bit, endereços superiores a 80h (P0.0);
  - Há instruções de salto condicional que se baseiam no estado do bit *carry* (JC e JNC) ou de um bit da memória de dados interna (JB, JNB e JBC):
  - Exemplo:

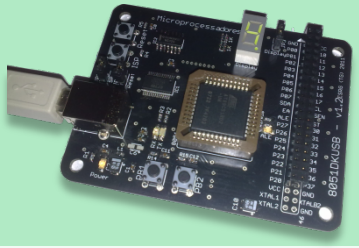
```
        MOV     C,P2.4
        ANL     C,/P2.5
        JNC     CONTINUA
        SETB    P1.7
CONTINUA:
        ...
```



Instruções de Salto

Mnemonic	Operation	Execution Time @ 12MHz (µs)
JMP addr	Jump to addr	2
JMP @A + DPTR	Jump to A + DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

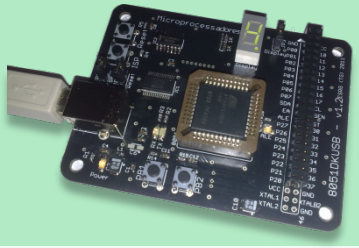
Mnemonic	Operation	Addressing Modes				Execution Time @ 12MHz (µs)
		DIR	IND	REG	IMM	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNZ A,<byte>,rel	Jump if A = <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> = #data		X	X		2



Conjunto de Instruções

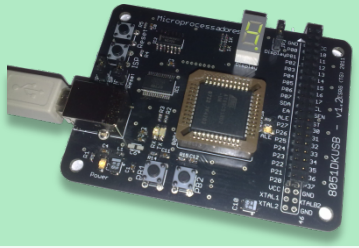
- **Instruções de salto**
  - As instruções de salto permitem alterar a sequência normal de execução das instruções por parte do CPU;
  - A instrução “JMP addr” pode ser utilizada pelo programador, mas na realidade será utilizada uma de três instruções: SJMP, AJMP ou LJMP;
  - Qualquer que seja a instrução seleccionada o programador tem de fornecer ao *assembler* o endereço do destino do salto do mesmo modo: através de um etiqueta ou de um endereço de destino de 16-bit;
  - A instrução JMP @A+DPTR fornece “*case jumps*”. O endereço de destino é calculado em tempo de execução:

```
SWITCH_ACC:
        MOV     DPTR,#CASEJUMP
        RL      A
        JMP     @A+DPTR
CASEJUMP:
        AJMP    CASE0
        AJMP    CASE1
        AJMP    CASE2
```



Conjunto de Instruções

- **Invocação de subrotinas**
  - A instrução “CALL addr” pode ser utilizada pelo programador, mas na realidade será utilizada uma de duas instruções: ACALL ou LCALL;
  - Qualquer que seja a instrução seleccionada o programador tem de fornecer ao *assembler* o endereço da subrotina do mesmo modo: através de um etiqueta ou de um endereço de 16-bit, o assemblador coloca o endereço no formato correcto;
  - Para retornar da subrotina e continuar a execução na instrução seguinte à instrução CALL (invocação da rotina), tem de ser utilizada a instrução RET;
  - O endereço da instrução seguinte, que é o conteúdo do PC, é armazenado na memória de dados (na *Stack*);
  - A instrução RETI é usada para terminar uma rotina de serviço a uma interrupção.



Conjunto de Instruções

- **Saltos condicionais**
  - Alteração do fluxo de execução, em tempo de execução, com base no valor de uma variável (posição de memória), um registo ou de um bit;
  - Reparar que não existe o bit Z (zero) no registo PSW. As instruções JZ e JNZ testam o valor do acumulador directamente;
  - A instrução DJNZ é utilizada para controlo de *loops*:

```
        MOV     R3,#8D
        MOV     A,#1
CTRLOOP:
        MOV     P1,A
        RL      A
        DJNZ    R3,CTRLOOP
        ...
```

  - A instrução CJNE pode também ser utilizada para controlo de *loops*, no entanto a principal utilização é nas comparações “maior que, menor que”. Os dois bytes dos operandos são utilizados, o bit *carry* é colocado a 1 se o primeiro operando for menor que o segundo.