NOTAS DE AULA 06

MICROCONTROLADOR 8051

SUMÁRIO DO VOLUME NOTAS DE AULA 06

NOTAS DE AULA

06

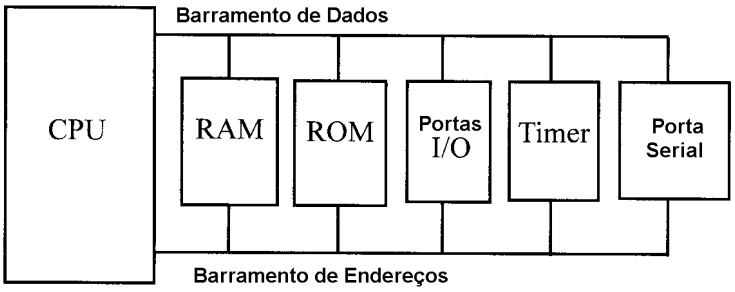
MICROCONTROLADOR 8051

VOL. 01

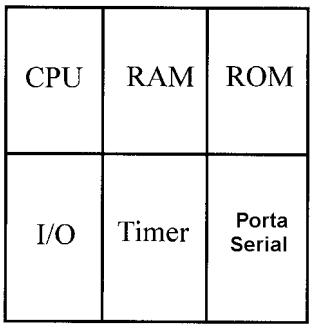
SUMÁRIO

1 INTRODUÇÃO AOS MICROCONTROLADORES

Ao contrário dos microprocessadores, como por exemplo, o 8085 e o 80x86 (apenas citando a família Intel), os microcontroladores tipicamente integram RAM, ROM e I/O, assim como, a CPU no mesmo circuito integrado. Por outro lado, o espaço para armazenamento de programas é limitado (comparando com os microprocessadores) e normalmente o conjunto de instruções é desenvolvido em número inferior aos microprocessadores.



(a) Microprocessador: sistema de proposta geral



(b) Microcontrolador

Muitas das aplicações dos microcontroladores podem ser divididas em três categorias principais:

1. sistemas de controle *open-loop* (controle seqüencial): é usado em aplicações onde o processo ou dispositivo necessita ser controlado por uma seqüência de estados. Ex.: uma máquina de vender refrigerantes: aceita vários valores (diferentes tipos de moedas), reconhece a seleção do produto, vende o produto, encontra o preço e retorna a mudança corrente: se a moeda foi insuficiente ou o produto está fora de estoque, então uma mensagem apropriada deve ser exibida;
2. sistemas de controle *closed-loop*: são caracterizados pelo uso do monitoramento em tempo real de um processo que necessita de um controle contínuo. A saída deste processo é monitorada usando vários transdutores e conversores A/D's e o processo é modificado continuamente para obter o resultado desejado. Exemplos: máquinas automáticas e no campo da robótica;
3. Outras aplicações: manipulação de estruturas de dados, como por exemplo, as utilizadas no campos da visão robótica e sistemas de comunicação de dados.

Típicas aplicações dos microcontroladores:

1. Em casa: telefones, sistemas de segurança, sistemas de abertura de portas de garagem, máquinas de fax, televisões, câmeras de vídeo, controles remotos, vídeos games, telefones celulares, instrumentos musicais, brinquedos, etc;
2. No escritório: telefones, computadores, sistemas de segurança, máquinas de fax, microondas, máquinas xerox, impressoras, etc;
3. No automóvel: monitoramento do motor, air bag, freios ABS, instumentação, sistemas de segurança (alarmes), controle da transmissão, ar-condicionado (controle da climatização), etc.

Critérios básicos a serem analisados na escolha de um microcontrolador:

1. quantidade de bits, pois existem microcontroladores de 8 bits (8051), 16 bits e 32 bits;
2. velocidade;
3. encapsulamento. Extremamente importante, pois isto definirá principalmente o espaço que será necessário no produto final;

4. consumo de potência. Deve ser levado em conta, principalmente em sistemas baseados em bateria;
5. capacidade de memória interna: RAM e ROM;
6. o número de pinos de I/O e timer's internos;
7. facilidade em realizar upgrade para outras versões;
8. custo por unidade;
9. facilidade na sua utilização (programação, montadores existentes, compiladores C, etc...).

1.1 Introdução ao µC 8051

A família µC-51 (a Intel denomina de MCS-51) pode ser dividida em três grupos principais:

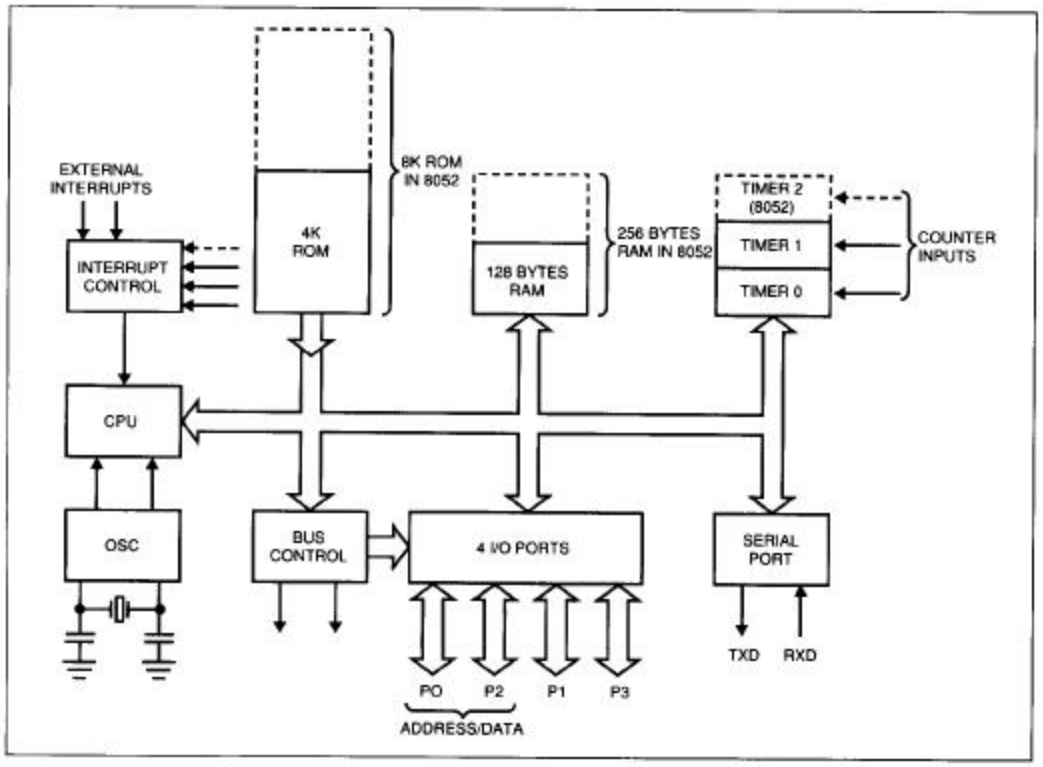
Características internas (on-board)		
8051	**8031**	**8751**
4KB de ROM 128B de RAM 32 pinos de I/O 2 counter/timers de 16 bits 6 fontes de interrupção 1 porta serial duplex 1 processador booleano de bits	Não tem ROM interna e usa memória externa para armazenagem de programa.	O mesmo que o 8051, exceto que a ROM é uma UV-EPROM.

Obs.: 8052, 8032, 8752 → são versões expandidas (on-board): 8KB de ROM, 256B de RAM e 3 timers. Versões CMOS (baixa potência): 80C51, 80C31 e 87C51.

Características	***8051***	***8052***	***8031***
ROM (interna)	4KB	8KB	0KB
RAM	128	256	128
Timers	2	3	2
Pinos de I/O	32	32	32
Porta Serial	1	1	1
Fontes de Interrupção	6	8	6

Obs.: Programas desenvolvidos no 8051 rodam no 8052, mas o inverso não.

Diagrama de blocos do mC 8051.



Outras famílias do µC8051:

1. µC8751: este µC possui somente uma memória interna UV-EPROM de 4KB;
2. Família Atmel (da *Atmel Corporation*): popular versão do µC8051 que possui internamente uma memória ROM no formato de flash memory. A sua vantagem é que esta memória pode ser apagada rapidamente (na ordem de segundos e não é necessário usar um apagador de ROM) quando comparado a 20 minutos ou mais na versão µC8751. Para eliminar a necessidade de um programador de PROM a Atmel apresentou uma versão AT89C51 cuja memória interna pode ser programa através da porta serial (RS232C) do IBM PC;
3. Família *Dallas Semiconductor*: na versão ROM interna é no formato de uma NV-RAM (a programação desta memória pode ser realizada através da porta serial).

Algumas versões da família Atmel 8051.

Versão	*ROM*	*RAM*	*Pinos I/O*	*Timer*	*Interrupção*	*Vcc*
AT89C51	4KB	128B	32	2	6	5V
AT89LV51	4KB	128B	32	2	6	3V
AT89C1051	1KB	64B	15	1	3	3V
AT89C2051	2KB	128B	15	2	6	3V
AT89C52	8KB	128B	32	3	8	5V
AT89LV52	8KB	128B	32	3	8	3V

Obs. A letra C indica CMOS.

Algumas versões da família Dallas Semiconductor.

Versão	*ROM*	*RAM*	*Pinos I/O*	*Timer*	*Interrupção*	*Vcc*
DS5000-8	8KB	128B	32	2	6	5V
DS5000-32	32KB	128B	32	2	6	5V
DS5000T-8	8KB	128B	32	2	6	5V
DS5000T-8	32KB	128B	32	2	6	5V

Obs.: A letra T indica real-time clock (RTC). Esta característica é diferente do timer normal do μC8051. O RTC gera e mantém o tempo do dia (hora-minuto-segundo) e datas (ano-mês-dia) mesmo quando não está sendo alimentado.

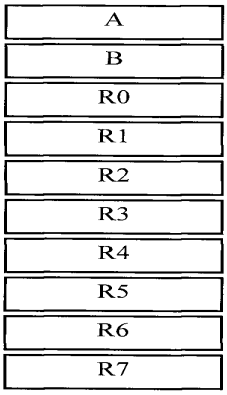
Cabe ressaltar, que existem versões da família 8051, como por exemplo, a família da Philips que incluem dispositivos internos, como por exemplo, conversores A/D's, D/A's, maior quantidade de portas de I/O, etc.

2 PROGRAMANDO EM LINGUAGEM ASSEMBLY: uma introdução ao 8051

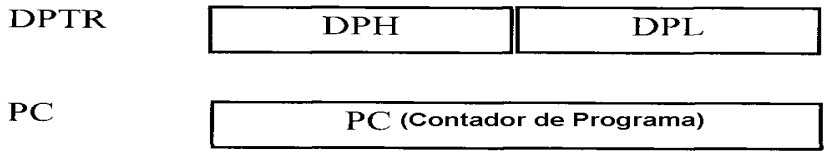
2.1 Registros básicos

Na CPU, os registros (normalmente chamados de registradores no 80x86) são usados para armazenar informações temporariamente (dados e endereços). A maioria dos registros do 8051 são de 8 bits (no 8051 só existe um tipo de dado: de 8 bits). Os registros mais comumente utilizados no 8051 são listados abaixo (**posteriormente estudaremos os outros registros e os endereços de cada um dos registros**).

Alguns registros de 8 bits do 8051:



Alguns registros de 16 bits do 8051:



Nomes:

A → Acumulador → Usado em todas as instruções aritméticas e lógicas.

DPTR → Apontador de Dados

PC → Contador de Programas

Comparação básica entre o µC8051 e o µP80x86.

	µP 80x86	µC 8051
Registradores de 8 bits	AL, AH, BL, BH, CL, CH, DL, DH	A, B, R0, R1, R2, R3, R4, R5, R6, R7 (R0-R7 → 4 banc os de registros)
Registradores de 16 bits (apontador de dados)	BX, SI, DI	DPTR
Contador de Programas (PC)	IP (16 bits)	PC (16 bits)
Entrada	MOV DX, endereço da porta IN AL, DX	MOV A, Pn ; (n = 0 a 3)
Saída	MOV DX, endereço da porta OUT DX, AL	MOV Pn, A ; (n= 0 a 3)
Loop	DEC CL JNZ Rótulo	DJNZ R3, Rótulo (usando R0 a R7)
Pilha (Stack Pointer)	SP (16 bits) PUSH → empilha e decrementa SP POP → desempilha e incrementa SP	SP (8 bits) PUSH → empilha e incrementa o SP POP → desempilha e decrementa o SP
Movimento de dados		
Do segmento de código	MOV AL, CS:[SI]	MOVC A, @A+PC
Do segmento de dados Da RAM	MOV AL, [SI] MOV AL, [SI] ; usar somente SI, DI ou BX	MOVX A, @DPTR MOV A, @R0 ; usar somente R0 ou R1
Para a RAM	MOV [SI], AL	MOV @R0, A

a) Instrução MOV

MOV destino, fonte ; copia o operando fonte para o operando destino

Exemplos:

```
MOV   A, #55H      ; A ← 55H, ou seja, carrega o valor 55H no registro A
MOV   R0, A        ; copia o  conteúdo de A para R0 (ao ser processado: A=R0=55H)
MOV   R1, A        ; copia o conteúdo de A para R1 (ao ser processado: A=R0=R1=55H)
MOV   R2, A        ; copia o conteúdo de A para R2
                   ; (ao ser processador: A=R0=R1=R2=55H)
MOV   R3, #95H     ; carrega o valor 95H em R3 (ao ser processado: R3 = 95H)
MOV   A, R3        ; copia o conteúdo de R3 para A (ao ser processador: A = R3 = 93H)
```

Pode-se observar que:

1. valores (constantes) podem ser carregadas diretamente nos registros A, B ou R0 a R7. Porém, é necessário indicar que é um valor imediato utilizando-se o símbolo **#**

Exemplos:

```
MOV A, #23H     ; A = 23H
MOV R6, #12     ; R6 = 0CH, pois 12 é decimal (base default)
MOV B, #3CH     ; B = 3CH
; observe que existe a necessidade de utilizar o caracter 0 (zero) quando o número mais
; significativo é uma letra, como por exemplo:
MOV R5, #0F9H  ; R5 = F9H
MOV R5, #F9H    ; erro
```

2. se o valor de 0 a F são movidos para um registro de 8 bits, o resto dos bits são assumidos como zeros. Por exemplo:

```
MOV A, #5       ; A = 0000 0101B
```

3. Erros freqüentes:

```
MOV A, #7F2H    ; erro pois 7F2H > 8 bits (FFH)
MOV R2, 456     ; erro pois 456 > 255 decimal (FFH)
```

4. Observe este erro freqüente (as duas instruções são válidas):

```
MOV A, #17H     ; A = 17H = 0001 0111B, # indica que precede um valor
MOV A, 17H      ; move para o registro A o valor armazenado na localidade de
                ; memória 17H (para o endereço do registro A o conteúdo da
                ; posição do endereço 17H). Não esqueça que o µC8051 só possui
                ; memórias, ou seja, estes registros (A, B, R0, etc) nada mais são do que
                ; localidades de memória com endereços fixos
```

b) Instrução ADD

```
ADD A, fonte    ; soma o operando fonte ao acumulador
```

Exemplos:

(a)

```
MOV   A, #25H       ; carrega 25H no acumulador
MOV   R2, #34H      ; carrega 34H no registro R2
ADD   A, R2         ; soma o conteúdo do R2 com o conteúdo de A e armazena o resultado
                    ; em A ( A = A + R2)
```

(b) o que ocorre?

```
MOV   R5, #25H
MOV   R7, #34H
MOV   A, #0H
ADD   A, R5
ADD   A, R7
```

(c) o que ocorre?

```
MOV A, #25H
ADD A, #34H
```

2.2 Introdução a Linguagem Assembly do 8051

Cabe ressaltar, que a montagem de um programa assembly para o 8051, é similar a montagem de um programa em assembly para o 80x86, ou seja, desenvolve-se um arquivo fonte (programa fonte) → realiza-se a montagem → realiza-se a ligação. As diferenças que ocorrem é no uso do montador assembler e as instruções utilizadas para se chegar ao programa executável (basta verificar a documentação do montador assembler utilizado). A estrutura básica de um programa assembly segue no exemplo abaixo:

```
                                ; início do programa
        ORG   0H        ; Início na localização 0 (origem) – ORG → diretiva
        MOV   R5, #25H
        MOV   R7, #34H
        MOV   A, #0
        ADD   A, R5
        ADD   A, R7
        ADD   A, #12H
AQUI:   SJMP  AQUI      ; uso de um salto e seu correspondente rótulo (fica neste
                        ; rótulo). SJMP = Short Jump
        END             ; fim do programa
```

Percebe-se que a estrutura do programa para o 8051 é extremamente similar a estrutura do programa para o processador 80x86. O programa é constituído de Pseudo-instruções (DIRETIVAS: ORG, END, etc), instruções (os mnemônicos: ADD, MOV, SJMP, etc) e endereços (rótulos). Da mesma forma que no microprocessador 80x86, as instruções para o montador assembler do microcontrolador 8051 consiste de 4 campos:

[Rótulo:] mnemônico [operandos] ;[comentários]

; [ ] = é opcional

2.3 Montagem e execução do 8051

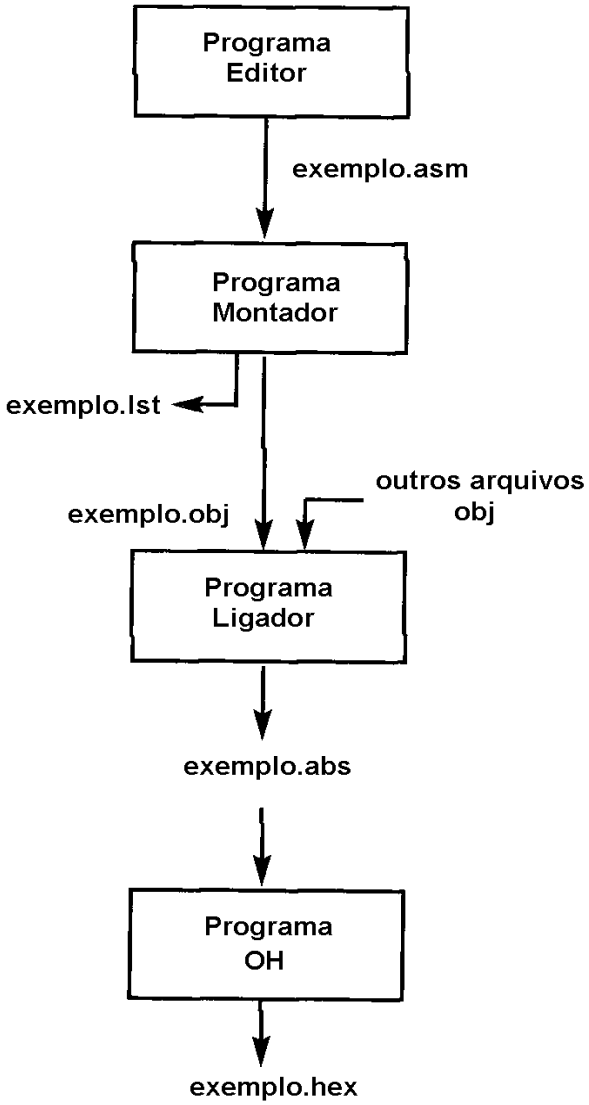
Os três passos básicos de um programa são: criar (arquivo fonte) → montar → executa-lo. Os passos básicos são:

1°) usar um editor de texto (este editor precisa estar apto a criar arquivos ASCII, como por exemplo, o EDIT do DOS). O arquivo fonte precisa ter a extensão .ASM ou .SRC (fonte) dependendo do montador usado;

2°) o arquivo Exemplo.ASM contém o programa fonte. O montador assembler do 8051 monta o programa fonte, ou seja, converte o arquivo fonte no correspondente código de máquina. O montador gera dois arquivos básicos: .obj e .lst;

3°) o terceiro passo é a ligação (linkagem). O programa ligador gera um arquivo .abs (arquivo objeto absoluto). Este arquivo é usado pelos treinadores do 8051 que monitoram o programa;

4°) O último passo é transformar o arquivo .abs usando um programa chamado, em geral, de Programa OH que converte o objeto para hexadecimal. Este programa segue com todos os programas montadores do 8051. Recentemente alguns montadores baseados no Windows realizam os passos 2 a 4 em um só passo.



Observação: o arquivo .lst (que é opcional sua criação) é extremamente útil para verificar os opcodes, endereços e possíveis erros que o montador detectou. Este arquivo pode ser aberto no editor EDIT do DOS. Abaixo segue um exemplo de um arquivo .lst:

```
1 0000                  ORG 0H
2 0000 7D25             MOV R5,#25H
3 0002 7F34             MOV R7,#34H
4 0004 7400             MOV A,#0
5 0006 2D               ADD A,R5

6 0007 2F               ADD A,R7

7 0008 2412             ADD A,#12H

8 000A 80FE       AQUI: SJMP AQUI
9 000C                  END
```

2.4 O Contador de programa (PC) e a ROM no 8051

Contador de Programa (PC): o PC aponta para o endereço da próxima instrução a ser executada. Como a CPU realiza a busca (ciclo de fetch) do opcode na ROM, o PC é incrementado para apontar para a próxima instrução. Este registro no 8051 é de 16 bits, logo, indica que o 8051 pode acessar a memória de programa (ROM) de faixa de endereço de 0000H a FFFFH, ou seja, um total de 64KB.

Observe que quando o microprocessador é ligado ou após um RESET (Vcc no pino de RESET), o PC é inicializado para 0000H (PC ← 0000H), ou seja, irá realizar o primeiro ciclo de fetch na posição de memória 0000H o que significa que o programa desenvolvido deve estar armazenado na primeira posição de memória da ROM (utiliza-se para isto ORG 0H).

Vamos novamente examinar o arquivo .lst fornecido anteriormente:

```
1  0000                    ORG  0H
2  0000  7D25              MOV  R5,#25H
3  0002  7F34              MOV  R7,#34H
4  0004  7400              MOV  A,#0
5  0006  2D                ADD  A,R5

6  0007  2F                ADD  A,R7

7  0008  2412              ADD  A,#12H

8  000A  80FE       AQUI:  SJMP AQUI
9  000C                    END
```

Endereço da ROM	Linguagem de Máquina	Linguagem Assembly
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80FE	AQUI: SJMP AQUI

Após o programa ser armazenado na ROM de um membro da família 8051, o opcode e operando são colocados na localidade 0000H da ROM como mostrado na tabela abaixo.

Endereço	Código
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

Por exemplo, o endereço 0000H contém 7D que é o opcode para mover um valor para o registro R5, e o endereço 0001H contém o operando (neste caso é o 25H) para ser movido para o registro R5. Portanto, a instrução MOV R5, #25H tem um código de máquina 7D25, onde o 7D é o opcode e 25 o operando.

2.5 Tipos de dados e diretivas no 8051

Tipos de dados: o 8051 tem somente um tipo de dado: de 8 bits. Este dado pode ser positivo ou negativo. Caso seja necessário trabalhar com dados maiores do que 8 bits o programador deverá quebrar este dados em partes.

Diretivas (Pseudo-instruções):

A diretiva **DB** (Define Byte) é a mais utilizada no 8051, pois é um microcontrolador de 8 bits. Seu uso é similar ao do 80x86, como por exemplo:

```
        ORG   500H
DADO1:  DB    28            ; decimal (1CH)
DADO2:  DB    00110101B     ; binário (35H)
DADO3:  DB    39H           ; hexadecimal
        ORG   510H
DADO4:  DB    "2591"        ; números ASCII – pode também ser utilizado ‘  ’
        ORG   518H
DADO6:  DB    "Meu nome e´ Alexandre"     ; caracteres ASCII
```

ORG (Origem): é usada para indicar o início do endereço (similar ao uso no 80x86). O número que segue após a diretiva pode estar na base Hexadecimal ou em Decimal (o montador converte para hexadecimal). Cuidado, alguns montadores utilizam a seguinte sintaxe: .ORG

EQU (equate): usado para definir uma constante sem ocupar espaço de memória (similar ao uso no 80x86). Exemplo:

```
CONTADOR        EQU   25
...
MOV   R3, #CONTADOR
```

END (fim): indica ao montador o fim do arquivo fonte (.asm). Cuidado, alguns montadores utilizam a seguinte sintaxe: .END

Regras para rótulos:

1. o primeiro caracter do rótulo precisa ter uma letras (não pode ser um número);
2. não podem ser utilizadas palavras reservadas (instruções e diretivas).

2.6 Flags e o registro PSW (Program Status Word) no 8051

Assim como toda unidade microprocessada, o 8051 necessita de um registro de flagas (PSW) para indicar o comportamento (estado) de determinadas instruções, como por exemplo, a existência ou não de Carry.

O registro PSW é de 8 bits (somente 6 são utilizados pelo 8051 e os 2 restantes podem ser utilizados pelo usuário, ou seja, são flags definidos pelo usuário). Quatro flags são chamados de flags condicionais, pois indicam alguma condição resultante após a execução de uma determinada instrução: CY (carrr), AC (auxiliary carry), P (parity) e OV (overflow).

CY	AC	F0	RS1	RS0	OV	--	P

CY	PSW.7	Flag Carry
AC	PSW.6	Flag Auxiliar Carry
--	PSW.5	Disponivel para o usuario
RS1	PSW.4	Seleçao do Banco de Registro (bit 1)
RS0	PSW.3	Seleçao do Banco de Registro (bit 0)
OV	PSW.2	Flag Overflow
--	PSW.1	Disponivel para o usuario
P	PSW.0	Flag Paridade

RS1	RS2	Bando de Registros	Endereços
0	0	0	00H - 07H
0	1	1	08H - 0FH
1	0	2	10H - 17H
1	1	3	18H - 1FH

Flag Carry (CY): é setado quando ocorre um flag do D7 bit. Pode também ser modificado, ou seja setado ou ressetado através das instruções SETB C (seta o CY) e CLR C (limpa o CY).

Flag Auxiliar Carry (AC): é setado quando ocorre um carry do D3 ao D4 durante as operações das instruções SUB ou ADD. Este flag é normalmente usado nas instruções que realizam operações BCD.

Flag Paridade (P): indica o número de 1's no A (acumulador) somente. Se A contém um número ímpar de 1's, então P = 1; caso contrário, P = 0 (número par de 1's).

Flag Overflow (OV): este flag é setado quando o resultado de uma operação com sinal é muito grande. Em geral, é usado para detectar erros.

Instruções que afetam o PSW.

Instrução	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
SETB C	1		
CLR C CPL C	0 X		
ANL C, bit	X		
ANL C, /bit	X		
ORL C, bit	X		
ORL C, /bit	X		
MOV C, bit	X		
CJNE	X		

X → pode ser 0 ou 1 dependendo do resultado obtido.

Exemplos: determine o comportamento dos flags CY, AC e P após a execução das rotinas abaixo:

(a)

MOV A, #38H

ADD A, #2FH

Solução:

38 + 2F = 67H

0011 1000 + 0010 1111 = 0110 0111

CY = 0, pois não ocorreu carry do D7

AC = 1, pois ocorreu carry do D3 ao D4

P = 1, pois A possui um número ímpar de 1's.

(b)

MOV A, #9CH

ADD A, #64H

(c)

MOV A, #88H

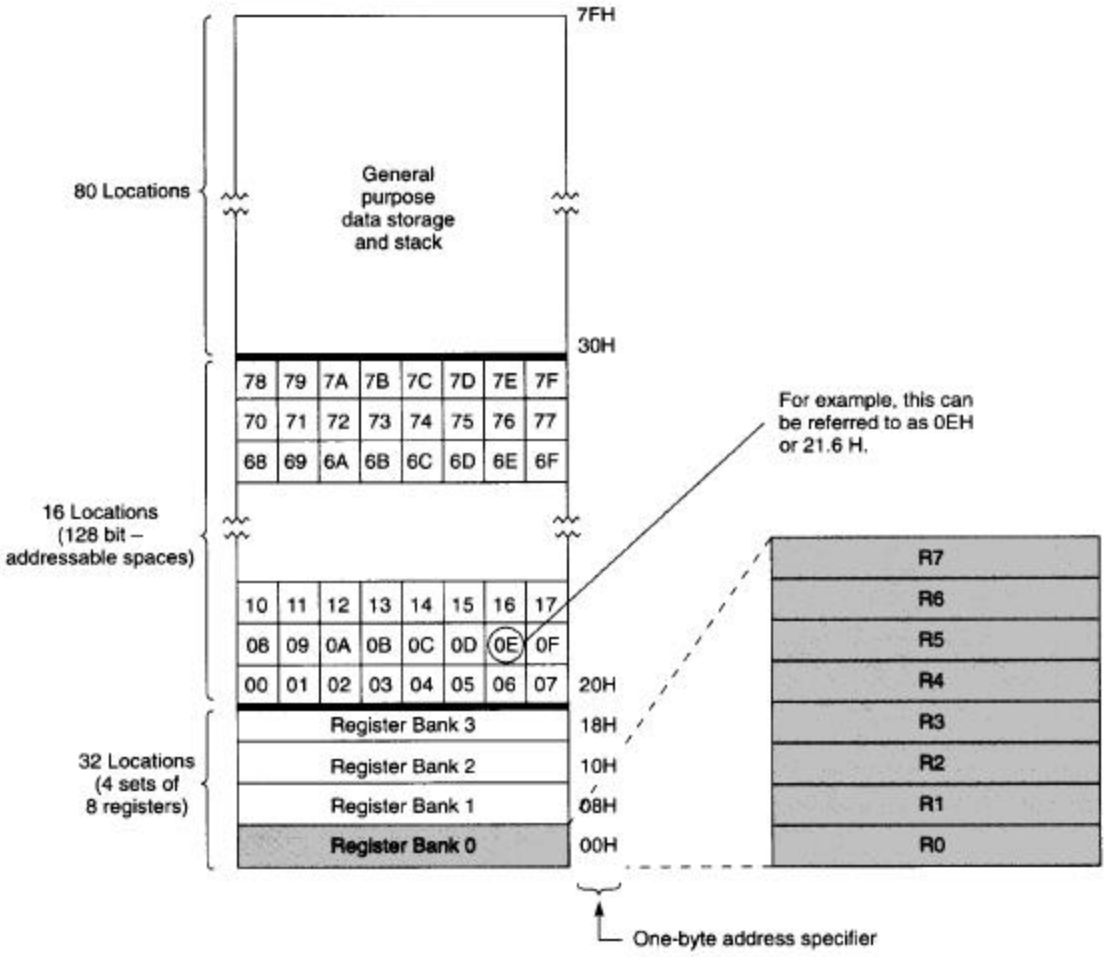
ADD A, #93H

2.7 Banco de registros (R0 a R7) e a pilha

Não esqueça o 8051 possui 128B de RAM. Esse espaço é apontado pela faixa de endereço de 00H a 7FH (será visto posteriormente que este espaço pode ser acessado diretamente como localidade de memória). Basicamente é dividido em três grupos:

1. um total de 32B (das localidades 00H a 1FH) são o conjunto dos Bancos de Registros (4 bancos ao todo) e a pilha;
2. um total de 16B (das localidades 20H a 2FH) são os espaços da memória de leitura/escrita bit endereçável;

3. um total de 80B (das localidades 30H a 7FH) são usados para leitura e escrita. São utilizados para armazenar os dados e parâmetros pelos programadores (conhecida como área *scratch pad*).



Banco de Registros (o Banco 0 é o default):

BANCO 0		BANCO 1		BANCO 2		BANCO 3	
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0
Default							

OBSERVAÇÃO: O BANCO 1 USA O MESMO ESPAÇO DA PILHA. ISTO É UM PROBLEMA PARA O PROGRAMA. PODEMOS ALOCAR OUTRA ÁREA COMO PILHA OU NÃO USAR O BANCO 1.

Exemplo_1: O que ocorre com as localidades da RAM após a execução das instruções abaixo:

```
MOV  R0, #99H      ; carrega R0 com o valor 99H
MOV  R1, #85H      ; carrega R1 com o valor 85H
MOV  R2, #3FH      ; carrega R2 com o valor 3FH
MOV  R7, #63H      ; carrega R7 com o valor 63H
MOV  R5, #12H      ; carrega R5 com o valor 12H
```

Solução:

Após a execução das instruções fornecidas anteriormente temos (o banco utilizado é o default Banco 0):

- a localidade 0 da RAM tem o valor 99H
- a localidade 1 da RAM tem o valor 85H
- a localidade 2 da RAM tem o valor 3FH
- a localidade 5 da RAM tem o valor 12H
- a localidade 7 da RAM tem o valor 63H.

Exemplo_2: Repetir o exemplo anterior utilizando os endereços da RAM no lugar do nome dos registros:

Solução:

```
MOV  00, #99H      ; carrega R0 com o valor 99H
MOV  01, #85H      ; carrega R1 com o valor 85H
MOV  02, #3FH      ; carrega R2 com o valor 3FH
MOV  07, #63H      ; carrega R7 com o valor 63H
MOV  05, #12H      ; carrega R5 com o valor 12H
```

Como alterar o banco a ser utilizado: como definido anteriormente o Banco 0 é o banco default quando o 8051 é ligado. Porém os outros três bancos podem ser utilizados, bastando programar os bits RS1 e RS2 do PSW.

Bits do PSW para seleção dos bancos de registros.

Banco	RS1 (PSW.4)	RS0 (PSW.3)
0	0	0
1	0	1
2	1	0
3	1	1

Exemplo:

```
SETB   PSW.4        ; seleciona o Banco 2 (RS1 = 1)
MOV   R0, #99H      ; carrega R0 com o valor 99H
MOV   R1, #85H      ; carrega R1 com o valor 85H
MOV   R2, #3FH      ; carrega R2 com o valor 3FH
MOV   R7, #63H      ; carrega R7 com o valor 63H
MOV   R5, #12H      ; carrega R5 com o valor 12H
```

Após a execução das instruções fornecidas anteriormente temos (o banco utilizado é 2):

- a localidade 10 da RAM tem o valor 99H
- a localidade 11 da RAM tem o valor 85H
- a localidade 12 da RAM tem o valor 3FH
- a localidade 15 da RAM tem o valor 12H
- a localidade 17 da RAM tem o valor 63H.

A Pilha: como no 80x86 a pilha no 8051 é utilizada para armazenamento temporário de dados ou endereço. Como a pilha é um pedaço da RAM é necessário a existência de um Registro para acessar a pilha SP (Stack Pointer). É um registro de 8 bits. Quando o 8051 é ligado, o SP é inicializado com 07H o que significa que a localidade 08 (também do Banco 1) é a primeira localidade a ser usada pela pilha do 8051.

Exemplo_1: Mostrar o que ocorre com a pilha e o SP para as instruções que seguem (assumir a área da pilha default = Banco 1):

```
MOV   R6, #25H
MOV   R1, #12H
MOV   R4, #0F3H
PUSH  6
PUSH  1
PUSH  4
```

Solução:

			Após PUSH 6			Após PUSH 1			Após PUSH 4	
0B			0B			0B			0B	
0A			0A			0A			0A	F3
09			09			09	12		09	12
08			08	25		08	25		08	25
SP = 07			SP = 08			SP = 09			SP = 0A	

Exemplo_2: Mostrar o que ocorre na pilha abaixo e com o SP para as instruções que seguem:

```
POP   3     ; POP em R3
POP   5     ; POP em R5
POP   2     ; POP em R2
```

Solução:

			Após POP 3			Após POP 5			Após POP 2	
0B	54		0B			0B			0B	
0A	F9		0A	F9		0A			0A	
09	76		09	76		09	76		09	
08	6C		08	6C		08	6C		08	6C
SP = 0B			SP = 0A			SP = 09			SP = 08	

3 INSTRUÇÕES: JUMPS, LOOPS E CALL

3.1 Loop e instruções para salto

Uma seqüência repetida de instruções um certo número de vezes é chamado de LOOP`. No 8051 o loop é realizado pela instrução **DJNZ reg, rótulo**. Nesta instrução, ***o registro é decrementado; se não é zero salta para o rótulo***, ou seja, para o endereço correspondente ao rótulo.

Exemplo_1: Escrever um programa para (a) limpar o acumulador e (b) adicionar 3 ao acumulador 10 vezes.

Solução:

```
                MOV   A, #0              ; A = 0, limpa o acumulador
                MOV   R2, #10            ; uso do R2 como contador
NOVAMENTE:      ADD   A, #03             ; adiciona 03 ao acumulador
                DJNZ  R2, NOVAMENTE      ; decrementa R2 (R2← R2 –1)
                                         ; e salta se R2 ≠ 0
                                         ; repete até R2 = 0 (10 vezes)
                MOV   R5, A              ; salva A no R5
```

Exemplo_2: Qual é o máximo número de vezes que o loop pode ser repetido?

Solução: Como R2 é um registro de 8 bits, pode armazenar no máximo FFH (255D); portanto, o loop pode ser repetido no máximo 256 vezes.

Exemplo_3: Escrever uma rotina para (a) carregar o acumulador com o valor 55H e (b) complementar o acumulador 700 vezes.

Solução:

perceba que 700 vezes é maior do que 256 (a máxima capacidade de qualquer registro), portanto é necessário utilizar mais do que um registro.

```
            ; ROTINA COM LOOP INTERNO
            MOV   A, #55H
            MOV   R3, #10
PRÓXIMO:    MOV   R2, #70
NOVO:       CPL   A
            DJNZ  R2, NOVO
            DJNZ  R3, PRÓXIMO
```

Saltos condicionais no 8051.

Instruções	***Ação***
JZ	Salta se A = 0
JNZ	Salta se A ≠ 0
DJNZ	Decrementa e salta se A ≠ 0
CJNE A, byte	Salta se A ≠ byte
CJNE reg, #dado	Salta se byte ≠ #dado
JC	Salta se CY = 1
JNC	Salta se CY = 0
JB	Salta se bit = 1
JNB	Salta se bit = 0
JBC	Salta se bit = 1 e reseta o bit (clear bit)

Exemplo_1: Escrever uma rotina para determinar se R5 contém o valor 0. Se for verdadeiro, armazene 55H.

Solução:

```
        MOV   A, R5        ; copia R5 para A
        JNZ   SALTO        ; salta se A não é zero (A ≠ 0)
        MOV   R5, #55H
SALTO:  ...
```

Exempplo_2: Encontrar a soma dos valores 79H, F5H e E2H. Armazenar a soma nos registros R0 (byte baixo) e R5 (byte alto).

Solução:

```
        MOV   A, #0
        MOV   R5, A
        ADD   A, #79H
        JNC   SALTO_1
        INC   R5
SALTO_1: ADD  A, #0F5H
        JNC   SALTO_2
        INC   R5
SALTO_2: ADD  A, #0E2H
        JNC   SALTO_3
        INC   R5
SALTO_3: MOV  R0, A
; observe que todos os saltos condicionais  são saltos curtos, ou seja, que o endereço do salto
; só pode estar no intervalo –128 a +127 bytes.
```

Saltos incondicionais: da mesma forma que no 80x86 existe a possibilidade de realizar um salto incondicional, ou seja, independente da condição do PSW. No 8051existem 2 saltos incondicionais no 8051: LJMP (long jump) e SJMP (short jump).

LJMP (long jump): é um salto incondicional. É uma instrução de 3 bytes em que o primeiro byte é o opcode e o segundo e terceiro bytes representam os 16 bits de endereço representados pelo rótulo: 0000H a FFFFH.

Lembre-se que o PC, no 8051, é um registro de 16 bits, possibilitando fornecer um endereço de 16 bits para endereçamento da ROM (até 64KB). Porém, o 8051 original possui somente 4KB de ROM interna, conseqüentemente, são necessários apenas 8 bits. Por esta razão, existe o salto incondicional SJMP (short jump).

SJMP (short jump): é uma instrução de 2 bytes, o primeiro byte é o opcode e o segundo byte é o endereço da localidade de memória (00H a FFH).

3.2 Instrução CALL

Utilizada para chamada de sub-rotina (uso similar ao do 80x86). No 8051, existem duas instruções para chamada de sub-rotina: LCALL (long call) e ACALL (absolute call) e a decisão do uso depende do endereço.

LCALL (long call): instrução de 3 bytes, o primeiro byte é o opcode e o segundo e terceiro bytes são usados para os endereços. Portanto, LCALL pode ser usado para chamada de sub-rotinas localizadas no espaço de endereço de 64KB.

Exemplo_1: examinar o que ocorre na rotina abaixo.

```
            ORG   0
SALTO_1:    MOV   A, #55H
            MOV P1, A                   ; envia 55H para a Porta 1
            LCALL       ATRASO
            MOV   A, #0AAH
            MOV   P1, A
            LCALL       ATRASO
            SJMP  SALTO_1
; rotina de atraso
            ORG   300H
ATRASO:     MOV   R5, #0FFH
SALTO_2:    DJNZ  R5, SALTO_2
            RET
            END
```

Exemplo_2: analisar a rotina abaixo e verificar a utilidade do uso da pilha após a execução do primeiro LCALL:

```
            ORG   0
SALTO_1:    MOV   A, #55H
            MOV   P1, A
            LCALL       ATRASO
            MOV   A, #0AAH
            MOV   P1, A
            LCALL       ATRASO
            SJMP  SALTO_1
            ORG   300H
ATRASO:
            MOV   R5, #0FFH
SALTO_2:    DJNZ  SALTO_2
            RET
            END
```

Exemplo_3: analisar a rotina abaixo e verificar a utilidade do uso da pilha após a execução do primeiro LCALL:

```
            ORG     0
SALTO_1:    MOV     A, #55H
            MOV     P1, A
            MOV     R4, #99H
            MOV     R5, #67H
            LCALL   ATRASO
            MOV     A, #0AAH
            MOV     P1, A
            LCALL   ATRASO
            SJMP    SALTO_1
            ORG     300H
ATRASO:     PUSH    4
            PUSH    5
            MOV     R4, #0FFH
SALTO_2:    MOV     R5, #0FFH
SALTO_3:    DJNZ    R5, SALTO_3
            DJNZ    R4, SALTO_2
            POP     5               ; POP em R5
            POP     4               ; POP em R4
            RET
            END
```

Em geral, a estrutura de um programa no 8051 segue:

```
; Programa Principal
                ORG   0
INÍCIO:         LCALL       SUBR_1
                LCALL       SUBR_2
                LCALL       SUBR_3
AQUI:           SJMP        AQUI
; fim do programa principal
;
SUBR_1:         ...
                ...
                RET
SUBR_2:         ...
                ...
                RET
SUBR_3:         ...
                ...
                RET         ; fim da sub-rotina 3
;
                END         ;  fim do programa
```

ACALL (absolute call): instrução de 2 bytes, porém somente 11 bits do 2bytes podem ser utilizados para endereço. Por outro lado, não existe diferença entre ACALL e LCALL em termos de salvamento do PC na pilha ou o uso da instrução RET.

3.3 Geração de atraso (DELAY) e seu cálculo

Ciclo de máquina: para a CPU executar uma instrução precisa de um certo número de ciclos de clock. Na família 8051, esses ciclos de clock são denominados de ciclos de máquina. Para isto, deve-se buscar a documentação das instruções para verificar a quantidade de ciclos de máquina para cada instrução. Na família 8051, o comprimento do ciclo de máquina depende da freqüência do oscilador (cristal) conectado ao sistema do 8051. O cristal, na família 8051, pode variar de 4MHz a 30MHz dependendo do modelo e do fabricante. Muitas vezes, utiliza-se um cristal de 11,0592 MHz para tornar o sistema do 8051 compatível com a porta serial do IBM PC. ***No 8051, um ciclo de máquina dura 12 períodos do oscilador***.

Da Intel temos:

Mnemonic	Byte	Machine Cycle
Program Branching		
ACALL addr11	2	2
LCALL addr16	3	2
RET	1	2
RETI	1	2
AJMP addr11	2	2
LJMP addr16	3	2
SJMP rel	2	2
JMP @A+DPTR	1	2
JZ rel	2	2
(continued)		

Mnemonic	Byte	Machine Cycle
Program Branching *(continued)*		
JNZ rel	2	2
CJNE A,direct,rel	3	2
CJNE A,#data,rel	3	2
CJNE Rn,#data,rel	3	2
CJNE @Ri,#data,rel	3	2
DJNZ Rn,rel	2	2
DJNZ direct,rel	3	2
NOP	1	2

NOP → 1 CICLO DE MÁQUINA.

Mnemonic		Byte	Machine Cycle
Arithmetic Operations			
ADD	A,Rn	1	1
ADD	A,direct	2	1
ADD	A,@Ri	1	1
ADD	A,#data	2	1
ADDC	A,Rn	1	1
ADDC	A,direct	2	1
ADDC	A,@Ri	1	1
ADDC	A,#data	2	1
SUBB	A,Rn	1	1
SUBB	A,direct	2	1
SUBB	A,@Ri	1	1
SUBB	A,#data	2	1
INC	A	1	1
INC	Rn	1	1
INC	direct	2	1
INC	@Ri	1	1
DEC	A	1	1
DEC	Rn	1	1
DEC	direct	2	1
DEC	@Ri	1	1
INC	DPTR	1	2
MUL	AB	1	4
DIV	AB	1	4
DA	A	1	1
Logical Operations			
ANL	A,Rn	1	1
ANL	A,direct	2	1
ANL	A,@Ri	1	1
ANL	A,#data	2	1
ANL	direct,A	2	1
ANL	direct,#data	3	2
ORL	A,Rn	1	1
ORL	A,direct	2	1
ORL	A,@Ri	1	1
ORL	A,#data	2	1
ORL	direct,A	2	1
ORL	direct,#data	3	2
XRL	A,Rn	1	1
XRL	A,direct	2	1
XRL	A,@Ri	1	1
XRL	A,#data	2	1
XRL	direct,A	2	1
XRL	direct,#data	3	2
CLR	A	1	1
CPL	A	1	1
RL	A	1	1
RLC	A	1	1
RR	A	1	1
RRC	A	1	1
SWAP	A	1	1

Mnemonic		Byte	Machine Cycle
Data Transfer			
MOV	A,Rn	1	1
MOV	A,direct	2	1
MOV	A,@Ri	1	1
MOV	A,#data	2	1
MOV	Rn,A	1	1
MOV	Rn,direct	2	2
MOV	Rn,#data	2	1
MOV	direct,A	2	1
MOV	direct,Rn	2	2
MOV	direct,direct	3	2
MOV	direct,@Ri	2	2
MOV	direct,#data	3	2
MOV	@Ri,A	1	1
MOV	@Ri,direct	2	2
MOV	@Ri,#data	2	1
MOV	DPTR,#data16	3	2
MOVX	A,@Ri	1	2
MOVX	A,@DPTR	1	2
MOVX	@Ri,A	1	2
MOV	@DPTR,A	1	2
PUSH	direct	2	2
POP	direct	2	2
XCH	A,Rn	1	1
XCH	A,direct	2	1
XCH	A,@Ri	1	1
XCHD	A,@Ri	1	1
Boolean Variable Manipulation			
CLR	C	1	1
CLR	bit	2	1
SETB	C	1	1
SETB	bit	2	1
CPL	C	1	1
CPL	bit	2	1
ANL	C,bit	2	2
ANL	C,/bit	2	2
ORL	C,bit	2	2
ORL	C,/bit	2	2
MOV	C,bit	2	1
MOV	bit,C	2	2
JC	rel	2	2
JNC	rel	2	2
JB	bit,rel	3	2
JNB	bit,rel	3	2
JBC	bit,rel	3	2

Exemplo_1: Considerando os seguintes cristais: (a) 11,0592MHz e (b) 16MHz e (c) 20MHz, encontrar o período do ciclo de máquina:

Solução:

(a) 11,0592MHz / 12 = 921,6KHz; ciclo de máquina é 1 / 921,6KHz = 1,085μs.

(b) 16MHz / 12 = 1,333MHz; ciclo de máquina é 1 / 1,133MHz = 0,75μs.

(c) 20MHz / 12 = 1,66 MHz; ciclo de máquina é 1 / 1,66MHz = 0,60μs.

Exemplo_2: para um sistema 8051 com oscilador de 11,0592MHz, encontrar o tempo de execução de cada uma das instruções abaixo:

(a) MOV R3, #55

(b) DEC R3

(c) LJMP

Solução:

; da documentação temos:

MOV R3, #55 → ciclo de máquina = 1

DEC R3 → ciclo de máquina = 1

LJMP → ciclo de máquina = 2

O ciclo de máquina para este sistema é de 11,0592MHz é 1,085μs. Logo:

MOV R3, #55 → ciclo de máquina = 1 → tempo de execução = 1 X 1,085μs = 1,085μs

DEC R3 → ciclo de máquina = 1 → tempo de execução = 1 X 1,085μs = 1,085μs

LJMP → ciclo de máquina = 2 → tempo de execução = 2 X 1,085μs = 2,17μs

Exemplo_3: encontrar o tamanho do atraso (delay) na rotina abaixo, se a freqüência do cristal é de 11,0592MHz.

```
            MOV   A, #55H
SALTO_1:    MOV   P1, A
            ACALL       ATRASO
            CPL   A
            SJMP  SALTO_1
ATRASO:     MOV   R3, #200
AQUI:       DJNZ  R3, AQUI
            RET
```

Solução:

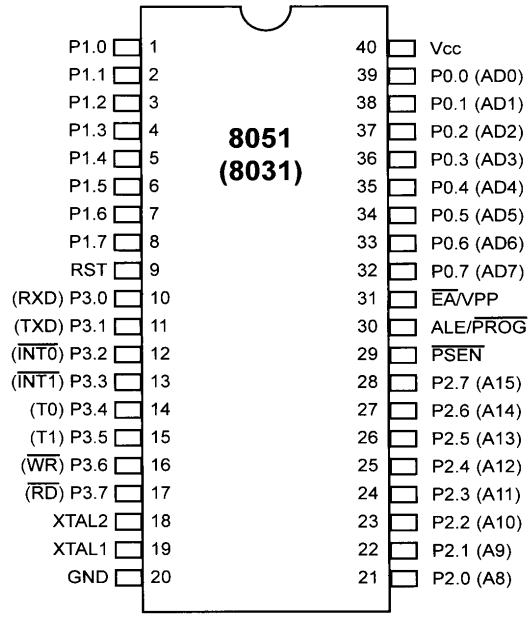
Verificar o ciclo de máquina de cada instrução (tabelas anteriores).

```
ATRASO:     MOV   R3, #200      → ciclo de máquina = 1
AQUI:       DJNZ  R3, AQUI      → ciclo de máquina = 2
            RET                 → ciclo de máquina = 1
```

Portanto, o atraso é de = [(200 x 2) + 1 + 1] x 1,085μs = 436,17μs.

4 DESCRIÇÃO DOS PINOS DO 8051 E UTILIZAÇÃO DAS PORTAS DE I/O

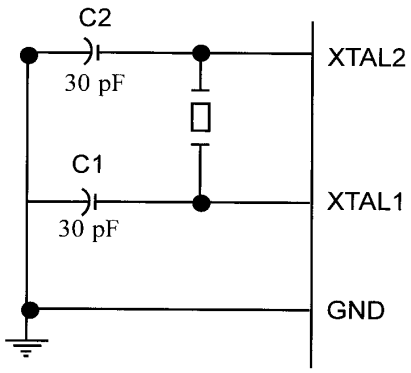
4.1 Pinagem e descrição dos pinos do 8051



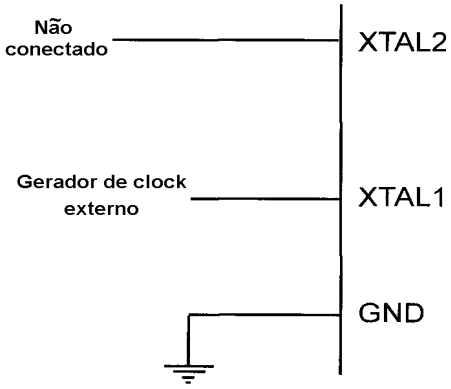
Dos 40 pinos temos:

- 32 pinos para portas de I/O de 8 bits (P0, P1, P2 e P3);
- Vcc: pino para fornecimento de alimentação ao μC (+ 5V);
- GND: pino de terra;
- XTAL1 e XTAL2: o μC 0851 possui um gerador de clock interno que requer a conexão de um cristal de quartzo nas entradas XTAL1 e XTAL2, como mostrado no esboço abaixo. Correntemente, a máxima freqüência é de 12MHz dividido internamente por 12, gerando uma taxa de 1μs para cada instrução básica (existem outras instruções com taxa de 2 e 4 μs). É importante observar, que dependendo do modelo e do fabricante da família 8051, pode-se ter dispositivos com freqüências diferentes. Por exemplo, para um μC 0851 de 12MHz, é necessário conectar um cristal de 12MHz no máximo (quando conectado ao cristal e ligado, pode-se observar a freqüência no pino XTAL2 com o uso de um osciloscópio). Pode-se usar um gerador de clock externo como mostrado na figura a seguir;

Uso do gerador de clock interno:



Uso do gerador de clock externo:

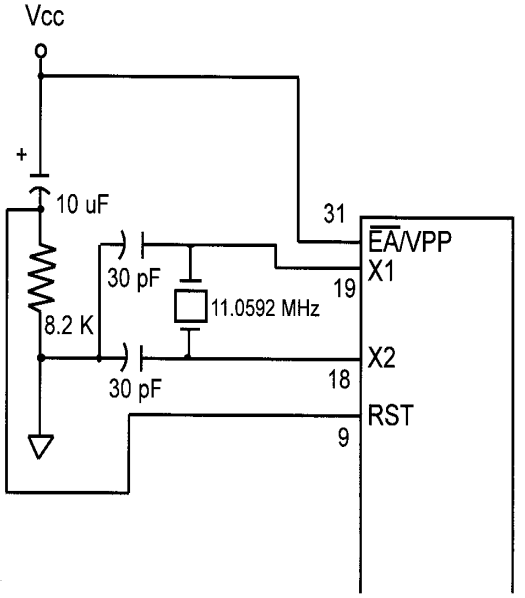


- RST: pino de RESET. É uma entrada ativa alta (normalmente baixa). Aplicando um pulso alto neste pino, o μC 0851 reset e termina todas as atividades. Os esboços a seguir apresentam duas formar de realizar o RESET no μC 0851. Para o RESET ser efetivado é necessário ter no mínimo uma duração de 2 ciclos de máquina;

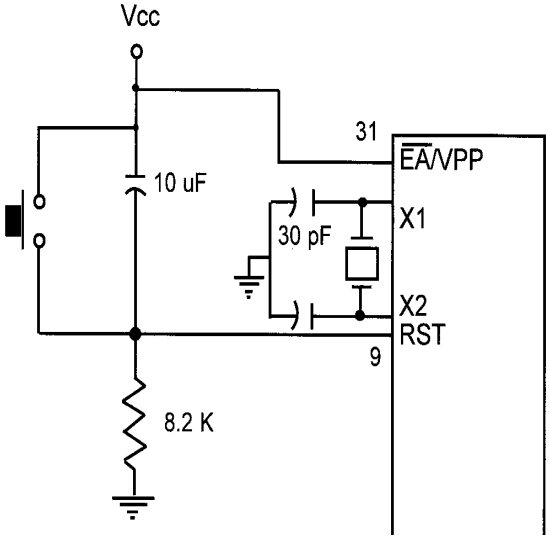
Valor de alguns registros após o RESET.

Registro	*Valor após o RESET*
PC	0000
ACC	0000
B	0000
PSW	0000
SP	0007
DPTR	0000

Circuito denominado de Power-On Reset.



Circuito Power-On Reset com push-botton.



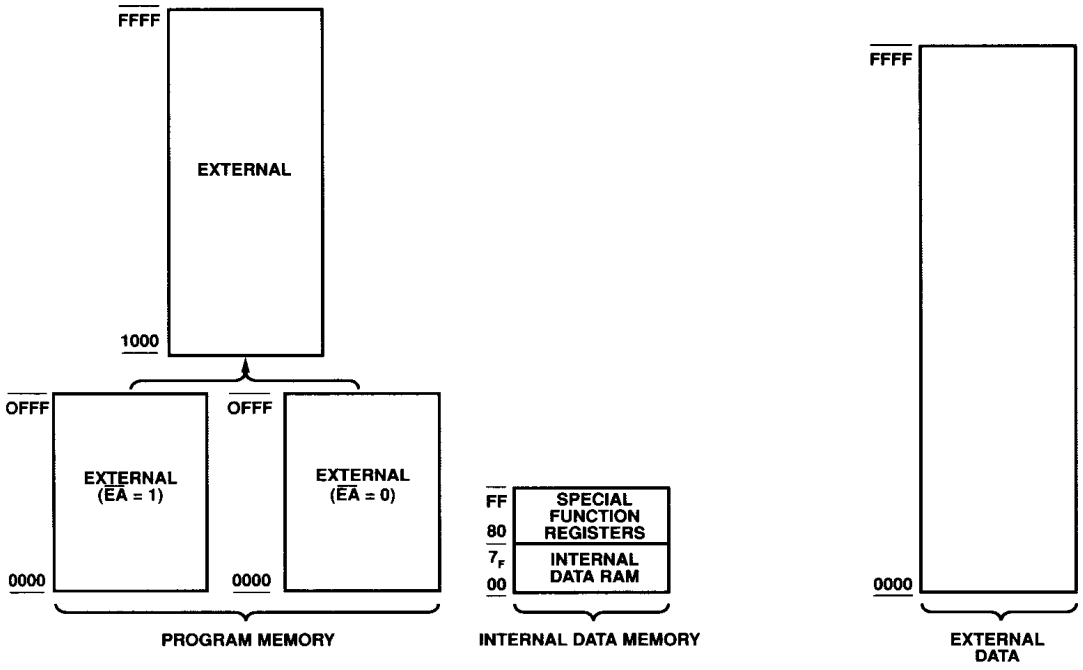
- pino EA' (external access): utilizado para determinar quando a memória ROM é interna ou não. Nos modelos da família µC 0851 que possuem ROM interna (como por exemplo, o 8751, 89C51, etc), o pino EA' deve estar ao Vcc. Para os membros da família µC 0851 que não possuem ROM interna (como por exemplo, 8031, 8032, etc) e, portanto, o programa é armazenado em uma ROM externa e o ciclo de busca deve ocorrer na memória externa, o pino EA' deve estar conectado ao GND para indicar que o programa está armazenado externamente. Observer que este pino não pode estar nunca desconectado;
- pino PSEN' (Program Store Enable - utilizado principalmente nos sistemas baseados no 8031): em sistemas com memória ROM externa este pino é conectado ao pino OE da ROM;
- pino ALE (Address Latch Enable - utilizado principalmente nos sistemas baseados no 8031): é um pino de saída ativo alto. Quando o 8031 está conectado a uma memória externa, a Porta 0 serve tanto como barramento de endereços e dados (barramento multiplexado, como no 8086 e 8085). Este pino é utilizado para demultiplexar o endereço e dado pela conexão deste pino ao pino G do buffer 74LS373 (conforme visto no 8086).

4.2 Organização da memória no µC8051

Separação do espaço de memória e tamanho

O 8051 (ao contrário do 80x86) separa os espaços de endereço: um para armazenamento do programa (usualmente ROM) e outro para armazenamento de dados (RAM). Um dado endereço numérico pode estar relacionado a 2 locações de memória diferentes logicamente e fisicamente, dependendo do tipo de instrução que usa o referido endereço.

Expansão de memória (on-chip + externa)	
Memória de programa Internamente 4KB	**Memória de dados internamente 128B**
Expansão até 64KB.	Expansão até 64KB (externo).
Configurações possíveis: 60KB externos + 4K internos OU 64 KB externos.	Configurações possíveis: 128B internos E 64KB externos.

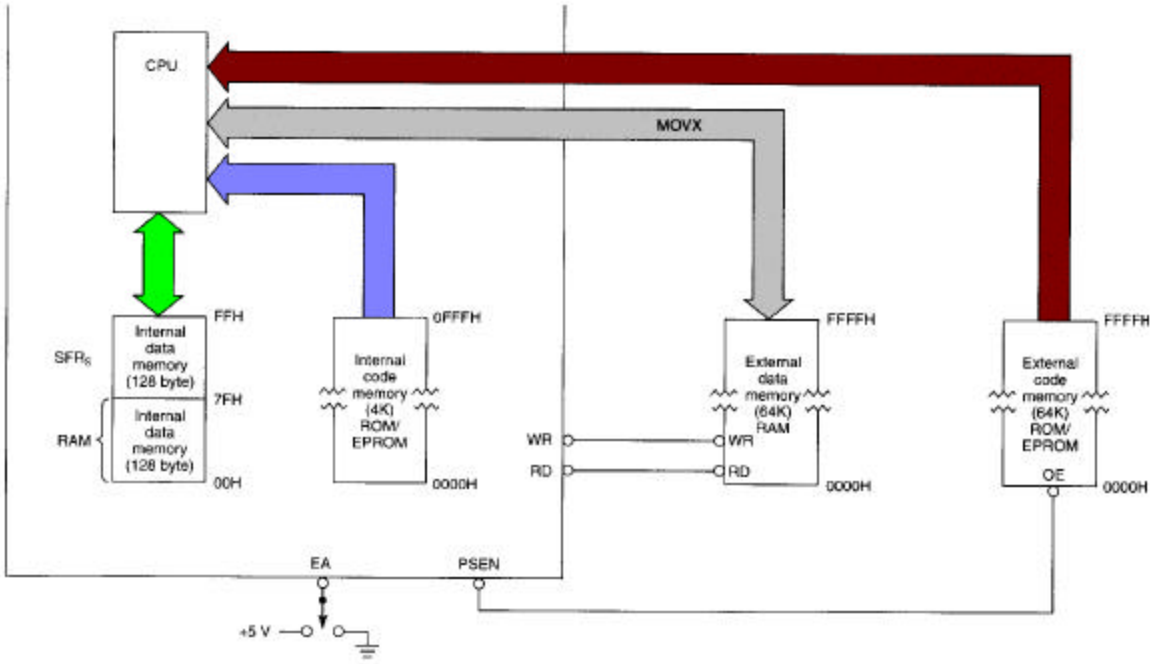


Após o RESET, a CPU inicia a busca de instruções (ciclo de fetch) iniciando no endereço 0000H. A localização física do endereço 0000H é igual no dispositivo on-board ou externo, dependendo do pino do 8051 $\overline{EA}$ (external address).

Se $\overline{EA} = 0$***, o endereço 0000H e todos os outros endereços de programas armazenados está relacionado à memória externa.***

Se $\overline{EA} = 1$***, os endereços 0000H a 0FFFH (até 1FFFFH para o 8052) está referenciado a ROM interna; endereços mais altos automaticamente estão relacionados à memória externa*** *(no 8031 →* $\overline{EA} = 0$*, pois não contém ROM interna (on-chip)). Alguns pinos de I/O são usados para endereços e dados quando usamos a memória externa.*

Espaço de endereçamento do 8051.

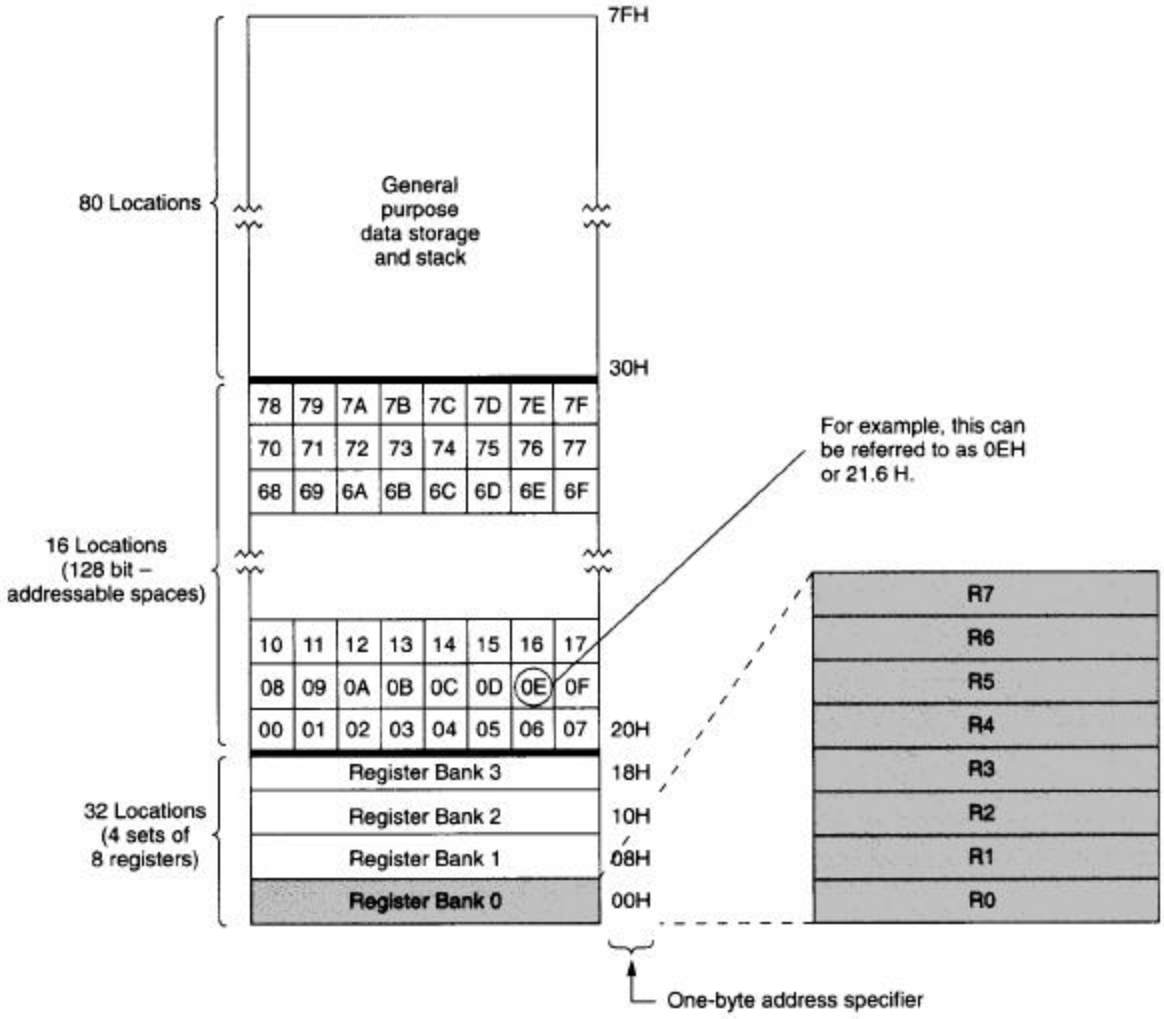


Armazenamento de dados

O armazenamento de dados internos no 8051 consiste de 1 espaço de 128B baixos e 128B altos para os SFR's (Special Function Registers).

Os 32 primeiro bytes do 128B baixos estão agrupados em 4 bancos de 8 registros. Somente um banco por vez pode ser ativado para uso e é selecionado por meio de 2 bits do registro PSW. Os 8 registros do banco ativo são designados por R0 a R7 e podem ser usados por certas instruções.

Os 16 bits acima do banco de registros formam um bloco que pode ser endereçado como bytes ou como 128 bits individuais. Os endereços byte são 20H a 2FH. Os endereços dos bits são de 00H a 7FH. Cuidado, pois o mesmo endereço, como por exemplo 20H, pode referir-se a um byte ou a um bit e cabe a instrução usada determinar quando a referência é um byte ou um bit.



Os 128 bytes mais baixos da RAM interna.

ROM interna (on-chip)

O conteúdo da ROM on-chip é desenvolvido e enviado para o fabricante do circuito intergrado programável. No 8031 não existe ROM interna e usa alguns dos seus pinos para requisitar a memória de programa externa.

Espaço SFR: Special Function Registers (RAM interna – 128 bytes mais altos)

Aos registrados associados com importantes funções do 8051 são fornecidas localizações de memória no espaço on-chip de armazenamento de dados (RAM interna), podendo ser então endereçadas pelas instruções do programa.

Algumas localizações SFR são bit endereçáveis ou byte-endereçáveis (esta característica não é encontrada no 8086).

	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	PI								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Pode-se notar no mapa SFR que nem todos os endereços são usados, em geral, não são utilizados (ou não documentados pelo fabricante) → são reservados para uso em futuras versões do 8051 e não serão usados nos nossos programas (o uso na leitura ou escrita destes espaços é indeterminado).

Portas (P0, P1, P2 e P3)

Os 32 pinos de I/O estão organizados em 4 portas de 8 bits designadas de P0-P3. Cada porta tem um latch de 8 bits associado. Os conteúdos dos latches podem ser lidos ou escritos para um SFR (P0 – 80, P1 – 90, P2 – A0, P3 – B0 – posições da RAM que armazenam os dados das 4 portas de I/O).

Serial Data Buffer (SBUF)

Atualmente é dois registros separados: um somente para leitura e outro somente para escrita. Quando os dados são escritos no SBUF, eles vão para um buffer de transmissão e seguem

para transmissão serial. Quando os dados são lidos do SBUF, eles vão para o buffer de recepção de dados serial.

Timer Registers

Os registros TH0 e TL0 são os bytes altos e baixos, respectivamente, do registro de 16 bits do timer/counter 0. Da mesma forma, TH1 e TL1 são do timer/counter 1.

Control Register

O SFR contém registros usado para o controle e estado do sistema de interrupção, o timer/counters e a porta serial. Eles são o IP (interrupt priority), IE (interrupt enable), TMOD (timer mode), TCON (timer control), SCON (serial port control) e PCON (power control – usado principalmente no 80C52).

Funções alternadas

Todos os pinos da porta 3 tem uma função alternada, como listado na Figura abaixo. Para habilitar uma função alternada, um 1 precisa ser escrito para o correspondente bit na porta latch.

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

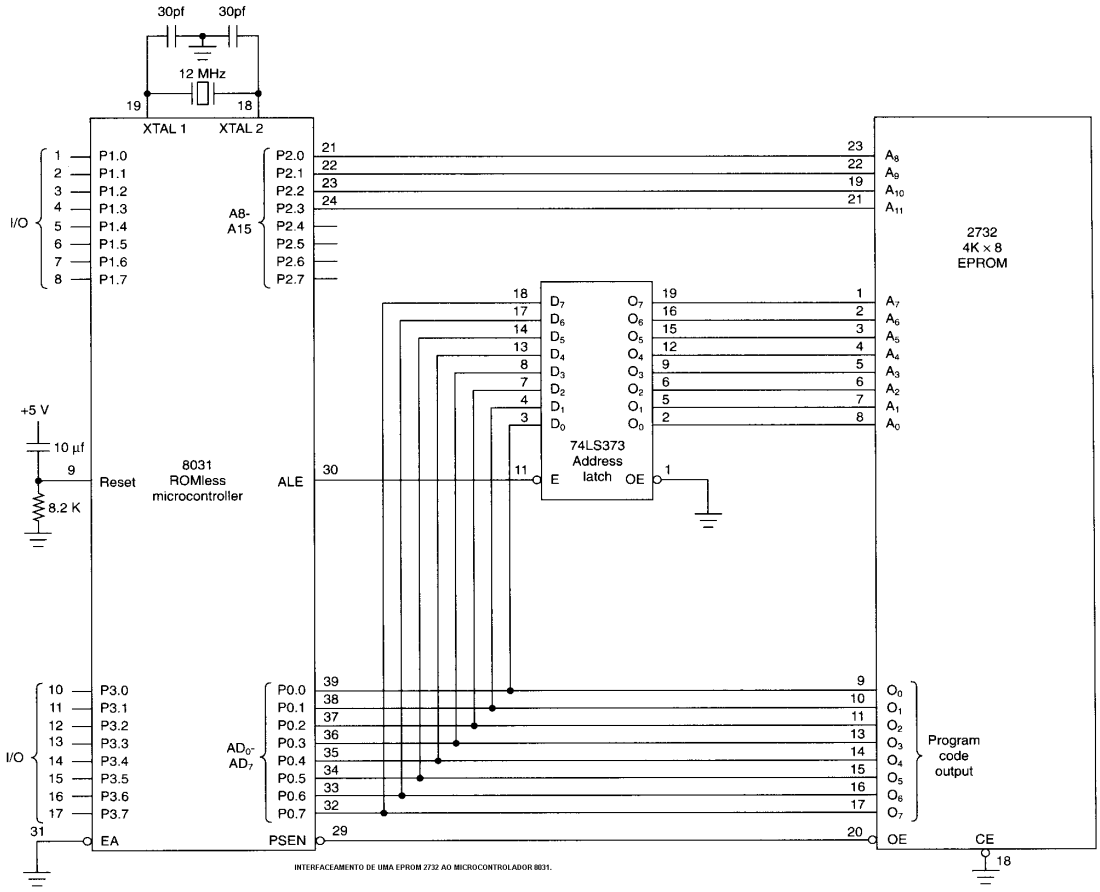
(MSB)							(LSB)
RD	WR	T1	T0	INT1	INT0	TXD	RXD

Symbol	Position	Name and Significance
RD	P3.7	Read data control output. Active low pulse generated by hardware when external data memory is read.
WR	P3.6	Write data control output. Active low pulse generated by hardware when external data memory is written.
T1	P3.5	Timer/counter 1 external input or test pin.
T0	P3.4	Timer/counter 0 external input or test pin.
INT1	P3.3	Interrupt 1 input pin. Low-level or failing-edge triggered.
INT0	P3.2	Interrupt 0 input pin. Low-level or failing-edge triggered.
TXD	P3.1	Transmit Data pin for serial port in UART mode. Clock output in shift register mode.
RXD	P3.0	Receive Data pin for serial port in UART mode. Data I/0 pin in shift register mode.

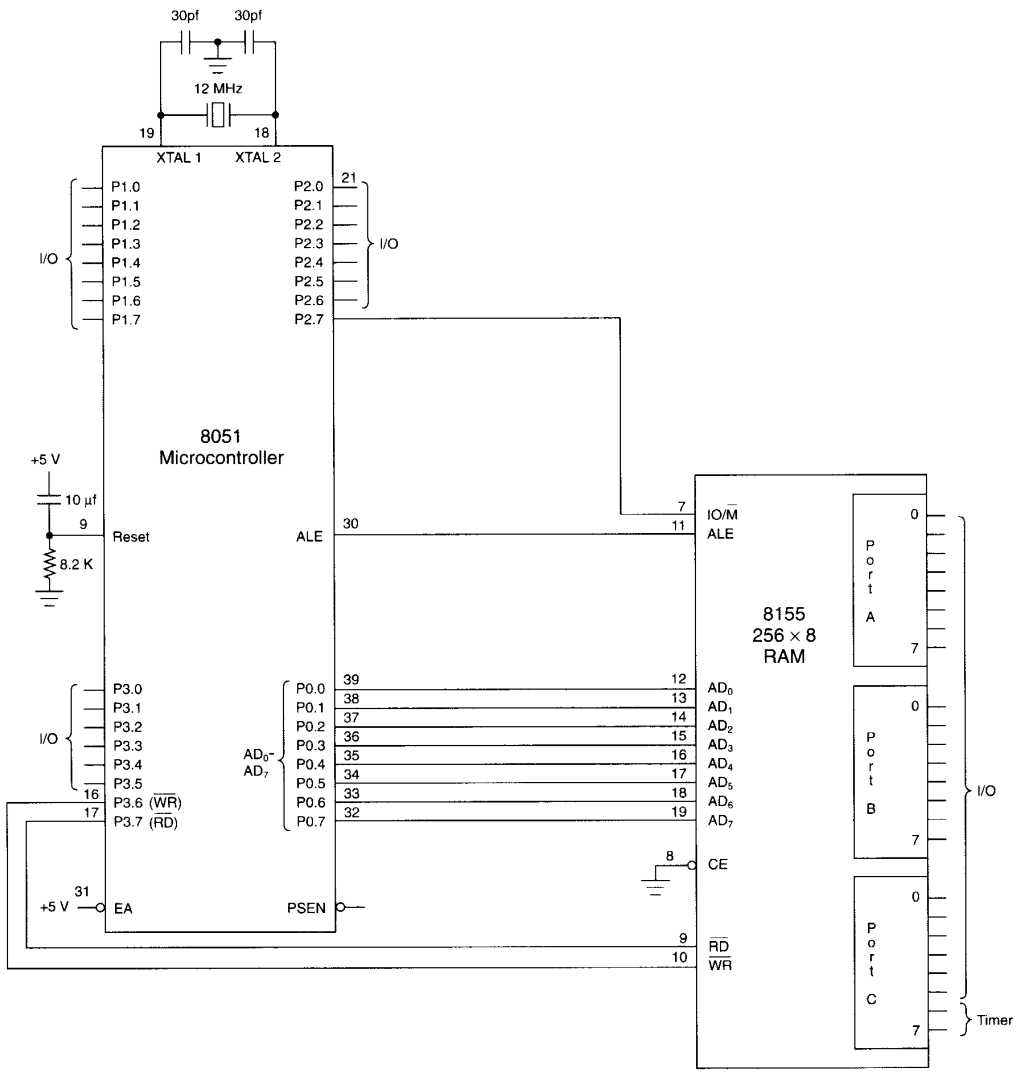
Acessando a memória externa

Como o 8051 separa a memória de programa da memória de dados, utiliza-se diferentes sinais de hardware para acessar os correspondentes dispositivos de armazenagem externos. O sinal PSEN (*program store enable*) é usado como o *read strobe* para a memória de programa, e RD e WR são usados como *read* e *write strobes* para acessar a memória de dados. Note que RD e WR são funções alternadas dos pinos P3.6 e P3.7, como descrito anteriormente. Também as portas 0 e 2 são usadas para acessar a memória externa.

Acessar o programa armazenado externamente sempre utiliza-se 16 bits de endereço. O acesso a memória de dados externa pode-se usar 8 ou 16 bits de endereço, dependendo da instrução executada. No caso de endereço de 16 bits, os 8 bits mais altos são saída na porta 2.



INTERFACEAMENTO DE UMA EPROM 2732 AO MICROCONTROLADOR 8031.



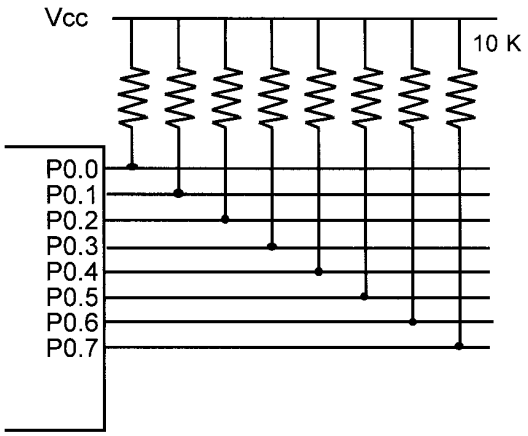
Interfaceamento de uma RAM/IO/Timer 8155 com o microcontrolador 8051.

4.3 Portas de I/O

O 8051 possui 4 portas de I/O (P0, P1, P2 e P3) de 8 bits cada. Todas as portas no RESET são configuradas como saída e podem, portanto serem utilizadas como portas de saída. Para usar estas portas como entradas é preciso programar.

Porta 0 (P0)

Pode ser utilizada como porta de entrada ou saída. Para utilizar esta porta (tanto para entrada como para saída) é necessário que cada pino esteja conectado externamente a um resistor pull-up de 10KΩ (é um porta Open Drain é o termo utilizado para tecnologia MOS da mesma forma que Open Collector para tecnologia TTL).



Porta 0 com resistores pull-up.

A P0 está configurada como saída (para ser entrada é preciso programar). Por exemplo, o que realiza a seguinte rotina:

```
        MOV   A, #55H
VOLTA:  MOV   P0, A
        ACALL  ATRASO
        CPL    A
        SJMP   VOLTA
```

; envia continuamente para a P0 os valores alternados 55H e AAH (0101 0101 = 55H → Complemento de Um = 1010 1010 = AAH).

Porta P0 como entrada

Com os resistores pull-up conectados a P0, se desejar utiliza-la como entrada é necessário programa-la inicialmente escrevendo 1 para todos os bits. Por exemplo:

```
        MOV   A, #0FFH      ; A = FFH
        MOV   P0, A         ; torna P0 entrada
VOLTA:
        MOV   A, P0         ; pega dados da P0
        MOV   P1, A         ; envia-os para a P1
        SJMP  VOLTA         ; retorna
```

Porta 1 (P1)

Pode ser utilizada como porta de entrada ou saída (não necessita de resistores pull-up, pois possui internamente). É configurada inicialmente como saída. Por exemplo:

```
        MOV   A, #55H
VOLTA:  MOV   P1, A
        ACALL ATRASO
        CPL   A
        SJMP  VOLTA
```

; envia continuamente para a P1 os valores alternados 55H e AAH (0101 0101 = 55H → Complemento de Um = 1010 1010 = AAH).

Porta P1 como entrada

Para torna-la uma porta de entrada é necessário programa-la como 1 em todos os bits. Verificar o que ocorre na rotina a seguir:

```
MOV   A, #0FFH          ; A = FFH
MOV   P1, A             ; torna P1 uma porta de entrada
MOV   A, P1             ; pega os dados da P1
MOV   R7, A
ACALL ATRASO
MOV   A, P1
MOV   R6, A
ACALL ATRASO
MOV A, P1
MOV R5, A
```

Porta 2 (P2)

Similar a P1, também não necessita de resistores pull-up e é configurada inicialmente como porta de saída. Por exemplo:

```
          MOV   A, #55H
VOLTA:    MOV   P2, A
          ACALL  ATRASO
          CPL    A
          SJMP   VOLTA
```

Porta P2 como entrada

Similar a P1, como exemplo:

```
          MOV   A, #0FFH     ; A = FFH
          MOV   P2, A        ; torna P2 entrada
VOLTA:    MOV   A, P2        ; pega dados da P2
          MOV   P1, A        ; envia-os para a P1
          SJMP   VOLTA       ; retorna
```

Porta 3 (P3)

Não necessita de resistores pull-up (da mesma forma que nas P1 e P2). Inicialmente é configurada como saída e normalmente é utilizada desta forma, pois possui uma série de pinos com funções especiais (as chamadas Funções Alternadas).

Funções alternadas da P3.

Bits da P3	Função
P3.0	RxD
P3.1	TxD
P3.2	INT0'
P3.3	INT1'
P3.4	T0
P3.5	T1
P3.6	WR'
P3.7	RD'

4.4 Programando I/O e manipulando bit

Diferentes formas de acessar 8 bits:

; verifique o seguinte código para entrada de 8 bits na P1:

```
VOLTA:  MOV   A, #55H
        MOV   P1, A
        ACALL  ATRASO
        MOV   A, #0AAH
        MOV   P1, A
        ACALL ATRASO
        SJMP   VOLTA
```

```
; verifique o código mais eficiente
VOLTA:    MOV   P1, #55H
          ACALL  Atraso
          MOV   P1, #0AAH
          ACALL Atraso
          SJMP   VOLTA

; verifique o uso da técnica abaixo: realiza três ações: (1) leitura da porta, (2) modifica e (3)
; escrever na porta

          MOV   P1, #55H
Novamente: XLR   P1, #0FFH
          ACALL  Atraso
          SJMP   Novamente
```

Portas BIT ENDEREÇÁVEL

Existem diversas aplicações em que não é necessário o trabalho com 8 bits. Uma característica importante do 8051 (não encontrada no 80x86) é a capacidade de trabalhar com as portas de I/O acessando individualmente bits (modo bit endereçável) sem alterar os restantes. Por exemplo:

```
Volta:    CPL    P1.2          ; complementa apenas P1.2 (o 3° bit da P1)
          ACALL Atraso
          SJMP   Volta
; outro exemplo
Novamente:     SETB P1.2        ; P1.2 = nível alto
               ACALL  Atraso
               CLR    P1.2      ; P1.2 = nível baixo
               ACALL  Atraso
               SJMP   Novamente
```

Exercícios:

1. Esboçar a pilha e o comportamento do SP para cada uma das rotinas abaixo:

(a)

```
ORG 0
MOV  R0, #66H
MOV  R3, #7FH
MOV  R7, #5DH
PUSH 0
PUSH 3
PUSH 7
CLR  A
MOV  R3, A
MOV  R7, A
POP  3
POP  7
POP  0
```

(b)

```
ORG  0
MOV  SP, #70H
MOV  R5, #66H
MOV  R2, #7FH
MOV  R7, #5DH
PUSH 5
PUSH 2
PUSH 7
CLR  A
MOV  R2, A
MOV  R7, A
POP  7
POP  2
POP  5
```

2. Encontrar os tempos de atraso (delay) para as sub-rotinas fornecidas abaixo (considere a freqüência do sistema de 11,0592MHz)

(a)

```
ATRASO:   MOV  R3, #150
AQUI:     NOP
          NOP
          NOP
```

```
                DJNAZ       R3, AQUI
                RET
```

(b)

```
ATRASO:         MOV  R3, #200
AQUI:           NOP
                NOP
                NOP
                NOP
                DJNZ  R3, AQUI
                RET
```

(c)

```
ATRASO:         MOV  R5, #100
VOLTA:          MOV  R2, #200
NOVO:           MOV  R3, #250
AQUI:           NOP
                NOP
                DJNZ  R3, AQUI
                DJNZ  R2, NOVO
                DJNZ  R5, VOLTA
                RET
```

3. Escrever um programa para monitorar a P1.2 até que fique alta. Quando P1.2 = nível alto, escrever o valor 45H na P0 e enviar um pulso alto para baixo para P2.3

4.5 Caracterização das portas de I/O

É importante observar a estrutura das portas de I/O (P0-P3) para impedir que ocorram danos ao microcontrolador 8051. Nos casos onde a necessidade de corrente excede a capacidade das portas, pode-se utilizar buffers/drivers tais como o 74LS245 (é usado para barramento de dados bidirecional) e 74LS244 (é usado para barramento de endereços unidirecional).

Quando conectamos uma chave (ou outros periféricos) devemos ter o cuidado para evitar possíveis danos às portas. Se uma chave com Vcc e terra é conectada diretamente ao pino da porta pode ser danificada. Para evitar isto, existem algumas possibilidades:

a) uma maneira é colocar um resistor de 10KΩ no caminho do Vcc da chave para limitar o fluxo através do transistor interno da porta;

b) outro método é usar uma chave com somente o terra (sem Vcc). Neste caso, a leitura de um nível baixo indica que a chave está pressionada;

c) outra maneira é conectar qualquer chave de entrada ao buffer tri-state 74LS244 antes do pino da porta