CAPíTULO VI

PORTAS PARALELAS

A família MCS-51 oferece 4 portas paralelas, denominadas P0, P1, P2 e P3 e para cada porta existe um SFR. De acordo com a configuração do hardware, uma ou mais portas estarão totalmente ou parcialmente disponíveis.

6.1. REGISTROS ENVOLVIDOS

Os 4 registros, P0, P1, P2 e P3 são na realidade os latches das portas e não os pinos da CPU. Algumas instruções operam com o conteúdo destes latches e outras com os valores dos pinos. As portas paralelas são utilizadas pela CPU para efetuar várias tarefas:

P0 → byte inferior de endereços e dados ==> BUS

P2 → byte superior de endereços

P3 → 0 → RXD - entrada serial
1 → TXD - saída serial
2 → *INT0 - interrupção externa 0
3 → *INT1 - interrupção externa 1
4 → T0 - entrada externa para o contador 0
5 → T1 - entrada externa para o contador 1
6 → *WR - strobe para escrita na memória de dados externa
7 → *RD - strobe para leitura da memória de dados externa

Durante uma operação de escrita ou leitura de memória de dados, por exemplo, os dados das portas P0 e P2 são removidos e por eles se emitem os endereços e dados. Terminadas as operações, o conteúdo do latch reaparece nos pinos da CPU.

Existem detalhes sobre a utilização de cada porta como entrada ou saída mas, de uma forma geral, pode-se dizer que:

Saída: Basta escrever 0 ou 1 na porta que o nível lógico aparece nos pinos corretos.

Entrada: todos as portas (exceto P0) possuem um pull-up interno; quando se escreve 1 em um bit da porta, diz-se que este bit está programado como entrada pois dispositivos externos podem colocar (forçar) este 1 em 0. Assim, para ter uma porta como entrada, escreve-se 1 e depois com a leitura será lido 1 ou 0 de acordo com o nível que externamente está aplicado no pino. Devido a isto, as portas são chamadas **quasi-bidirecionais**.

6.2. DESCRIÇÃO DO FUNCIONAMENTO

Cada porta paralela é constituída por três partes:

- um registro latch (SFR ==> P0,P1,P2,P3)
- um driver de saída
- Um buffer de entrada

Os drivers de saída de P0 e P2 e o buffer de entrada de P0 são usados para acessar a memória externa de programa ou de dados.

6.2.1. Porta P1

A figura 6.1 ilustra o esquema elétrico da porta 1.

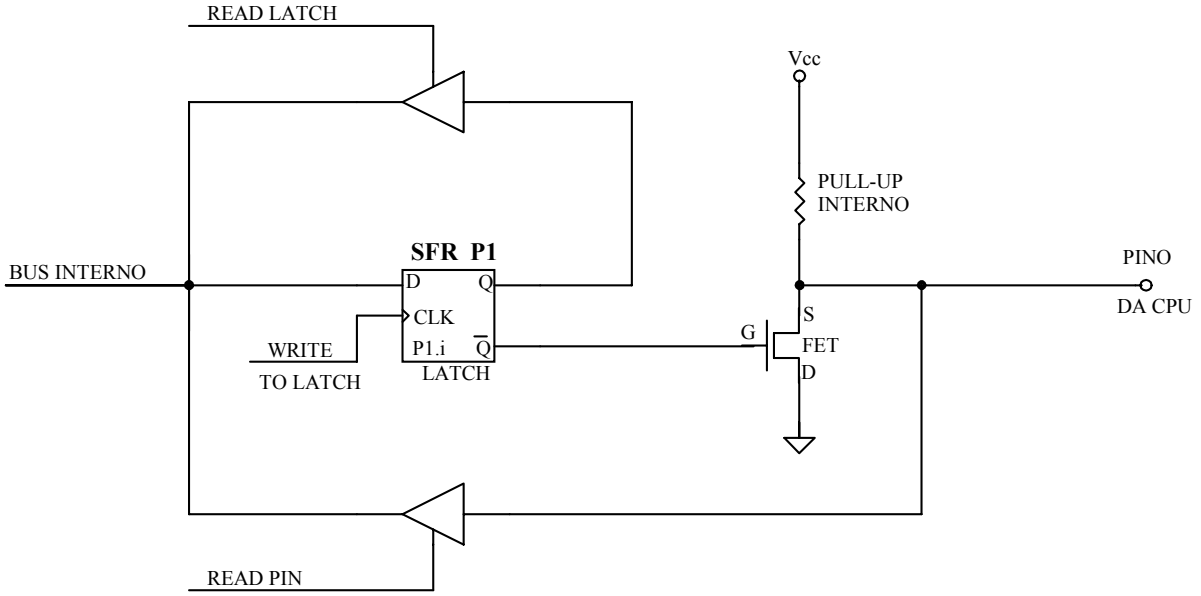

Figura 6.1. Esquema para um bit da porta P1.

O latch de um bit da porta P1 (um bit do SFR P1) é representado por um flip-flop D, no qual se escreve um valor do bus interno através de um pulso Write to Latch gerado pela CPU. A saída do latch é colocada no bus interno através de um sinal Read Latch gerado pela CPU. O nível do pino da porta é colocado no bus interno através do sinal Read Pin, também gerado pela CPU. Algumas instruções que lêem o estado da porta operam com Read Latch enquanto outras operam com Read Pin. Isto será visto com mais detalhes.

Quando escreve na porta:

$0 \rightarrow$ *Q=1 $\rightarrow$ FET ON $\rightarrow$ saída=0

$1 \rightarrow$ *Q=0 $\rightarrow$ FET OFF $\rightarrow$ saída=1 (pull-up)

Para ser usado como entrada, o latch da porta deve estar em 1; isso desconecta o driver FET da saída. Assim, o pino da porta vai para um nível alto, levado pelo pull-up. Esse pino poderá ser

levado para o nível baixo por qualquer elemento externo. Devido ao pull-up interno, essa porta é denominada "quasi-bidirecional". Quando está configurado para entrada, este vai para 1 e, se externamente é levado para baixo, então fornece corrente. Se a porta estivesse em alta impedância quando configurado como entrada, então seria classificado como "bidirecional verdadeiro".

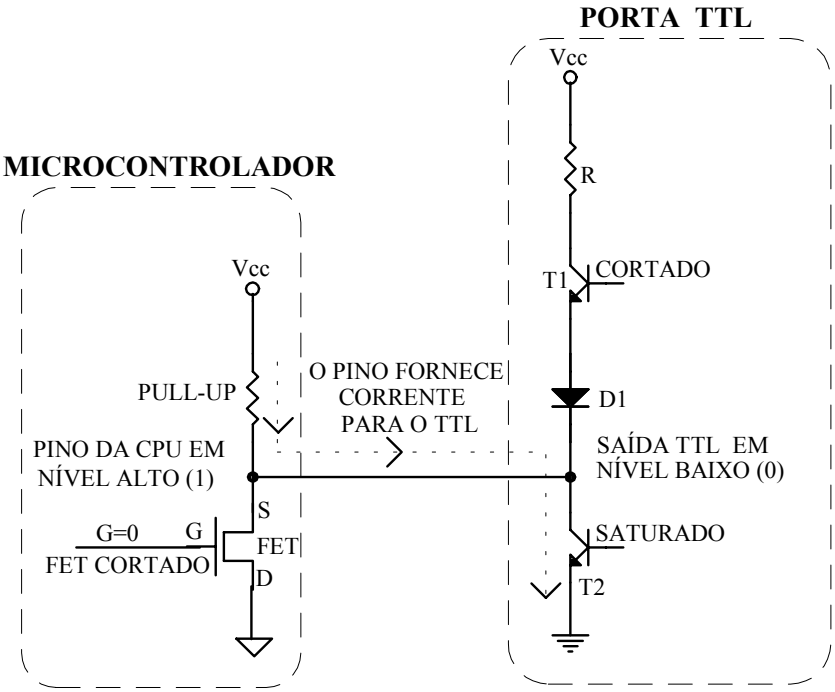

Figura 6.2. Pino do microcontrolador sendo levado a 0 por uma saída TTL.

6.2.2. Porta P3

Na figura 6.3 está o esquema elétrico da porta P3. Pela porta P3 têm-se diversas funções alternativas: *RD, *WR, T0, T1, TXD, RXD, *INT0, *INT1.

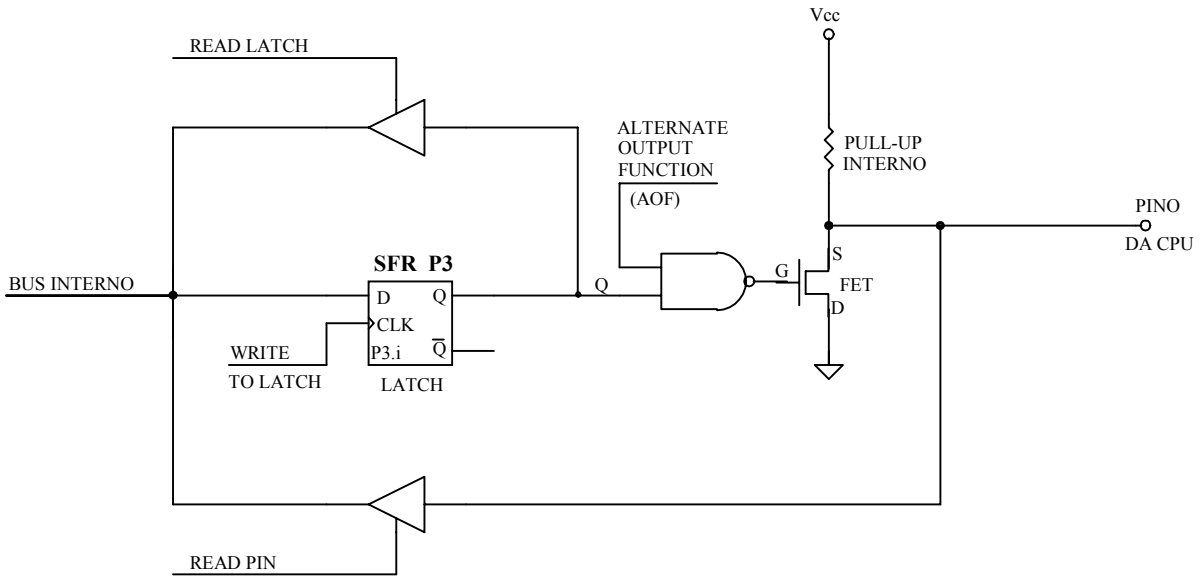

Figura 6.3. Esquema para um bit da porta P3.

Quando AOF=1, tem-se na saída a porta P3. Assim, se AOF=1 e for escrito:

$0 \rightarrow Q=0 \rightarrow G=1 \rightarrow$ FET ON $\rightarrow$ saída=0

$1 \rightarrow Q=1 \rightarrow G=0 \rightarrow$ FET OFF $\rightarrow$ saída=1 (pull-up)

Quando o latch de P3 contém 1, a saída pode ser controlada pelo sinal "Alternate Output Function - AOF". Já foram estudadas as funções alternativas geradas através de P3. Se o latch de P3 está em 0, o pino da porta estará em zero e as diversas funções alternativas não estarão disponíveis. P3 também é uma porta quasi-bidirecional.

6.2.3. Porta P2

O esquema da figura 6.4 ilustra a porta P2. Por esta porta também sai o byte mais significativo dos endereços.

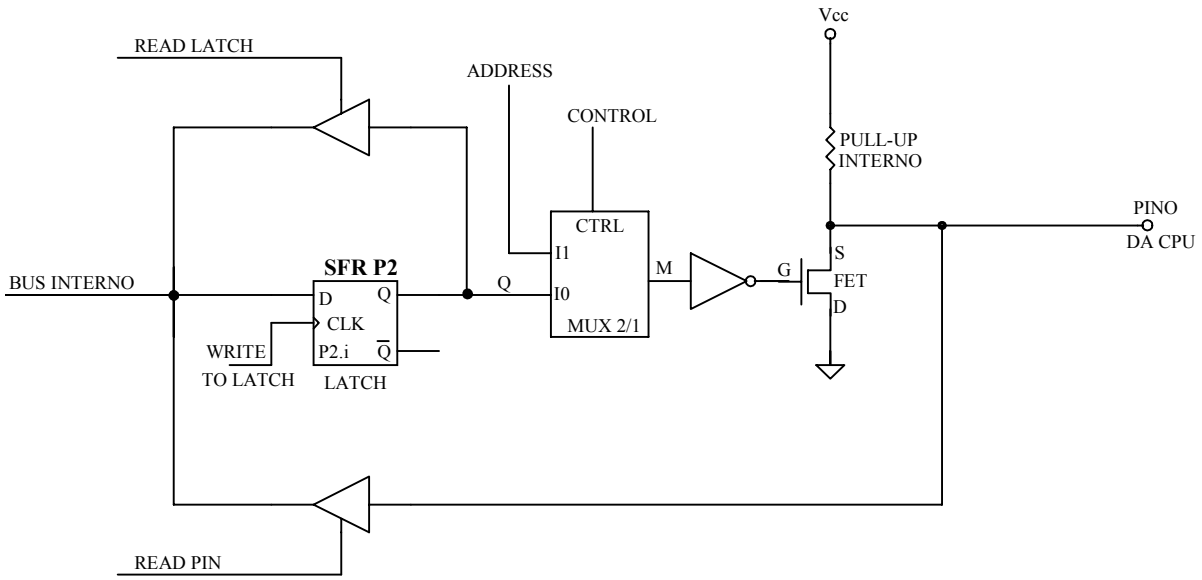

Figura 6.4. Esquema para um bit da porta P2.

O funcionamento é muito semelhante aos casos anteriores.

Se o CONTROL = 0 e for escrito:

$0 \rightarrow Q=0 \rightarrow M=0 \rightarrow G=1 \rightarrow$ FET ON $\rightarrow$ saída=0

$1 \rightarrow Q=1 \rightarrow M=1 \rightarrow G=0 \rightarrow$ FET OFF $\rightarrow$ saída=1 (pull-up)

Se o CONTROL = 1, ADDRESS controla o nível no pino. Assim, com CONTROL = 1:

ADDR=0 $\rightarrow M=0 \rightarrow G=1 \rightarrow$ FET ON $\rightarrow$ saída=0

ADDR=1 $\rightarrow M=1 \rightarrow G=0 \rightarrow$ FET OFF $\rightarrow$ saída=1 (pull-up)

De acordo com o nível lógico do sinal de CONTROL, permite-se a saída ao latch de P2 ou ao byte alto de endereço.

6.2.4. Porta P0

A figura 6.5 ilustra o esquema elétrico da porta P0. Esta porta é a mais complexa pois atende a três funções:

- bus de dados da CPU
- byte menos significativo dos endereços
- porta paralela (bidirecional verdadeira)

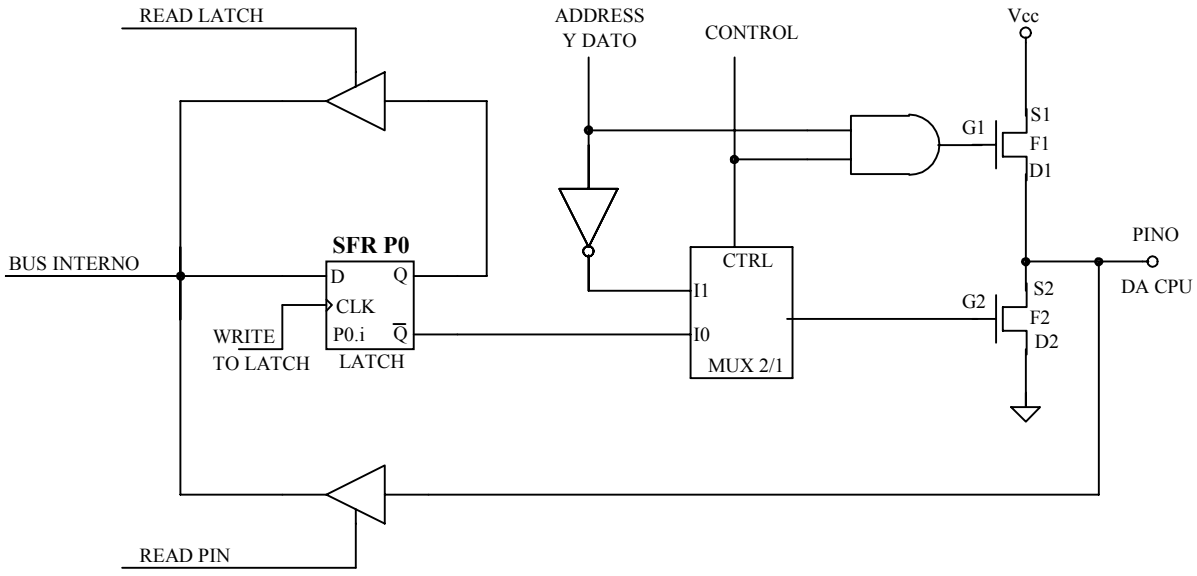

Figura 6.5. Esquema para um bit da porta P0.

A porta P0 difere das demais por ser utilizada para transportar dados durante as operações com a memória e portanto necessita ser bidirecional verdadeira. Se ela está sendo utilizada como porta paralela, então CONTROL = 0. Com isso G1=0 e F1 está sempre cortado (não existe pull-up).

Se CONTROL=0 e for escrito: :

$0 \rightarrow$ *Q=1 $\rightarrow$ G2=1 $\rightarrow$ F2 ON $\rightarrow$ saída=0

$1 \rightarrow$ *Q=0 $\rightarrow$ G2=0 $\rightarrow$ F2 OFF $\rightarrow$ a saída flutua (não há pull-up).

Note que, quando se escreve 1, a porta pode ser utilizada como uma entrada de alta impedância. Para transformá-la em uma porta quase-bidirecional basta colocar externamente resistores de pull-up.

Quando a porta é utilizada para enviar endereços ou dados, CONTROL =1 e a saída depende de ADDRESS/DATA (ADR/DT):

ADR/DT=0 $\rightarrow$ G1=0 $\rightarrow$ F1 OFF e G2=1 $\rightarrow$ F2 ON $\rightarrow$ saída=0

ADR/DT=1 $\rightarrow$ G1=1 $\rightarrow$F1 ON e G2=0 $\rightarrow$ F2 OFF $\rightarrow$ saída=1

(nunca se terá F1 ON e F2 ON ==> curto-circuito).

Para que a porta possa ser usada na leitura de memória é necessária alta impedância; nesse caso CONTROL=0 e se força uma escrita (com 1) no latch de P0.

Se CONTROL=0 → G1=0 → F1 OFF

Se Q=1 → *Q=0 → G2=0 → F2 OFF

Se F1 e F2 estão em OFF → alta impedância.

A leitura da memória é feita com o sinal READ PIN.

6.3. ESCRITA NAS PORTAS

Na execução de qualquer instrução que altere o latch de uma porta, o novo valor chega ao latch durante S6P2, que é o final do ciclo de instrução. Entretanto, os latches são na realidade enviados até os buffers de saída durante a fase P2 de qualquer período de clock (durante P1 se mantém o mesmo valor). Assim, o novo valor aparecerá no pino durante P1 do próximo ciclo (que é S1P1).

Se for necessária uma mudança de 0 para 1 nas portas P1, P2 e P3, o PULL-UP deverá entregar bastante corrente para que essa transição seja rápida. Nas transições de 0 para 1, durante S1P1 e S1P2, entra em atividade um PULL-UP adicional que pode fornecer 100 vezes mais corrente que o PULL-UP normal. Note que os PULL-UP são FETs e não transistores bipolares.

O PULL-UP normal é um FET (depletion-mode) com a porta (G) conectada à fonte (S). Quando está conectado à terra, circula por ele uma corrente de 0,25 mA.

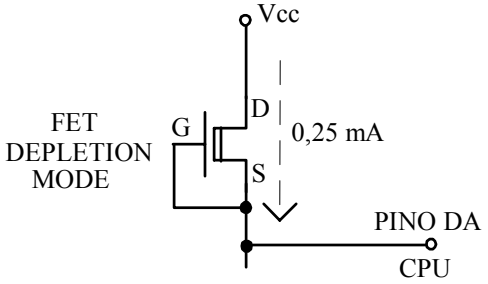

Figura 6.6. O FET (depletion mode) usado como PULL-UP para as portas paralelas. (HMOS)

Na figura 6.7. está o PULL-UP adicional que acelera as transições de 0 para 1. Note que quando A=B=0, a saída S vai para 1 e com isto o transistor entra no circuito fornecendo até 30 mA.

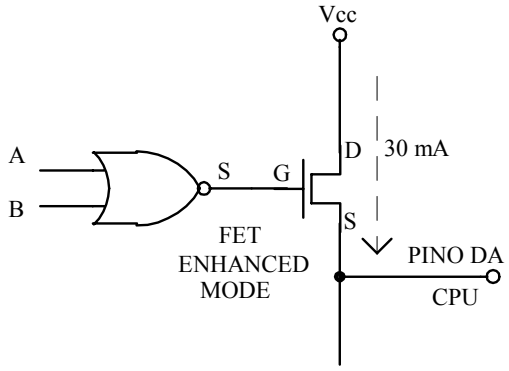

Figura 6.7. Controle para o FET (enhanced mode). (HMOS)

Na figura 6.8 está o esquema completo da saída da porta paralela.

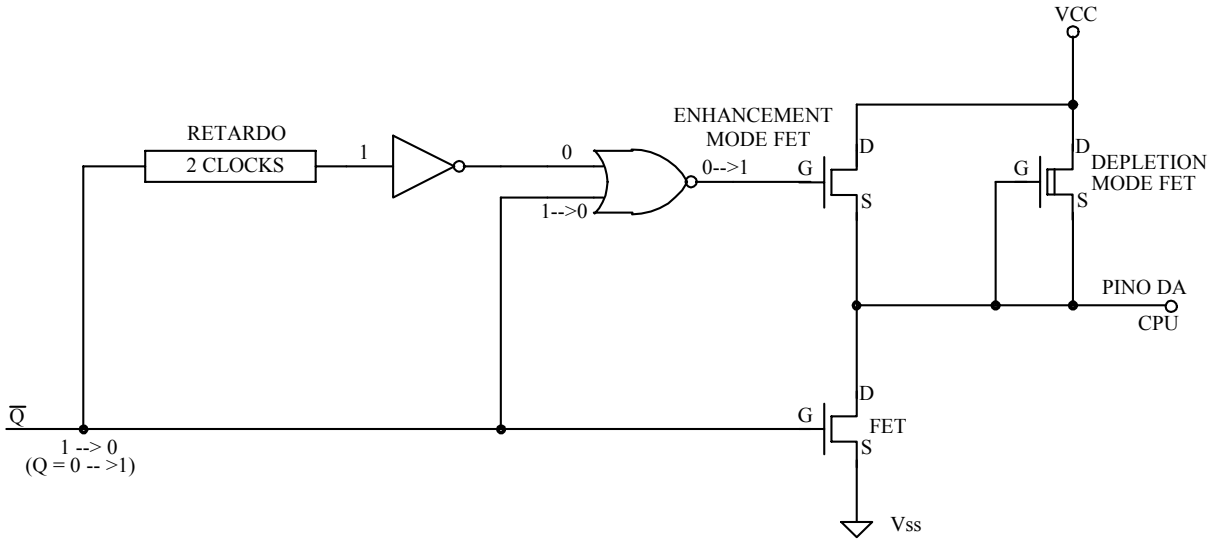

Figura 6.8. Acionamento do FET (enhanced mode) nas transições de 0→1. (HMOS)

Note que se *Q vai de 1→0, durante dois períodos de clock a saída (S) da porta NOR será 1, permitindo que o enhancement mode FET entre em paralelo com o FET (pull-up) normal, dando maior capacidade de corrente. Este esquema é utilizado em HMOS. Nas famílias CHMOS o esquema varia um pouco mas a idéia é a mesma.

Os buffers de saída das portas P1, P2 e P3 podem acionar até 4 cargas LS TTL. Eles podem ser acionados (trabalhando como entrada) por circuitos TTL e NMOS. Como possuem pull-up interno, essas portas também podem ser acionadas por TTL de coletor aberto mas as transições de 0 →1 não serão rápidas porque o pull-up tem baixa capacidade de corrente.

A porta P0 pode acionar até 8 LS TTL (modo BUS). Quando opera como porta paralela, é necessário pull-up externo para acionar outras entradas.

Algumas instruções de leitura utilizam o dado armazenado no latch, enquanto outras usam o estado do pino. As instruções que usam o dado do latch são aquelas que lêem o valor, (possivelmente) o alteram e o escrevem de novo (read-modify-write). Quando o destino do operando é uma porta ou um bit da porta, é utilizado o dado do latch e não o valor do pino. A seguir está uma lista destas instruções que operam com o dado do latch:

Instrução	Exemplo	
ANL	ANL	P1,A
ORL	ORL	P2,A
XRL	XRL	P3,A
JBC	JBC	P1.1,LB (pula se bit=1 e zera o bit)
CPL	CPL	P3.0
INC	INC	P2
DEC	DEC	P2

DJNZ		DJNZ	P3,label
MOV	PX.Y,C	MOV	P1.0,C
CLR	PX.Y	CLR	P1.2
SETB	PX.Y	SETB	P1.3

Pode parecer que as 3 últimas instruções não são do tipo "read-modify-write". Na realidade, é lido todo o byte da porta, o bit considerado é alterado e o novo byte é devolvido para a porta.

Uma razão para usar o dado do latch e não o valor do pino é evitar um equívoco na interpretação do nível de tensão do pino, como por exemplo, quando um bit de uma porta está sendo usado para acionar a base de um transistor.

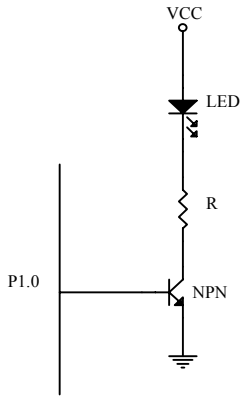

Figura 6.9. Bit 0 da porta 1 sendo usado para acender um led através de um transistor.

Quando P1.0=1 $\rightarrow$ LED aceso
Quando P1.0=0 $\rightarrow$ LED apagado

Sem dúvida quando P1.0=1 a tensão no pino vai estar em 0,7 V, o que é um 0 lógico em nível TTL. Espera-se que a instrução CPL P1.0 inverta o estado do led. Supondo que o led esteja aceso (P1.0=1), se a CPU usa o valor do pino, ela vai obter 0 e isto fará P1.0=1, quer dizer, não vai mudar o estado do led. O melhor é usar o dado do latch, que está em 1.

6.4. EXERCÍCIOS

Todos as exercícios e exemplos estão baseados no circuito de práticas.

EXERCÍCIO 6.1. PISCA1 E PISCA2

Acender os 3 leds em seqüência. Não esquecer de colocar um retardo. O led se acende quando se põe nível alto em um bit da porta.

```
;PISCA1.ASM
;
RTD             EQU             60000           ;65536>RTD>256
VERMELHO        EQU             P1.0
AMARELO         EQU             P1.1
VERDE           EQU             P1.2
;
                DEFSEG PROG, CLASS=CODE, START=0
                SEG             PROG
                ORG             RESET
INIC            MOV             P1,#0           ;APAGA OS LEDS
AQUI            SETB            VERMELHO
                LCALL           RETARD
                CLR             VERMELHO
                SETB            AMARELO
                LCALL           RETARD
                CLR             AMARELO
                SETB            VERDE
                LCALL           RETARD
                CLR             VERDE
                SJMP            AQUI
;
;ROTINA PARA GERAR UM RETARDO
RETARD          MOV             R7,# HIGH RTD
L1              MOV             R6,# LOW  RTD
L2              DJNZ            R6,L2
                DJNZ            R7,L1
                RET
                END
```

Uma outra solução, muito mais simples, pode ser utilizada. Como não são usados os demais pinos da porta P1, pode-se escrever qualquer coisa neles. Usa-se então o Acc e o CY para obter um total de 9 bits e, através de rotações, consegue-se gerar os códigos para acender os leds em seqüência. A figura 6.10 ilustra a idéia.

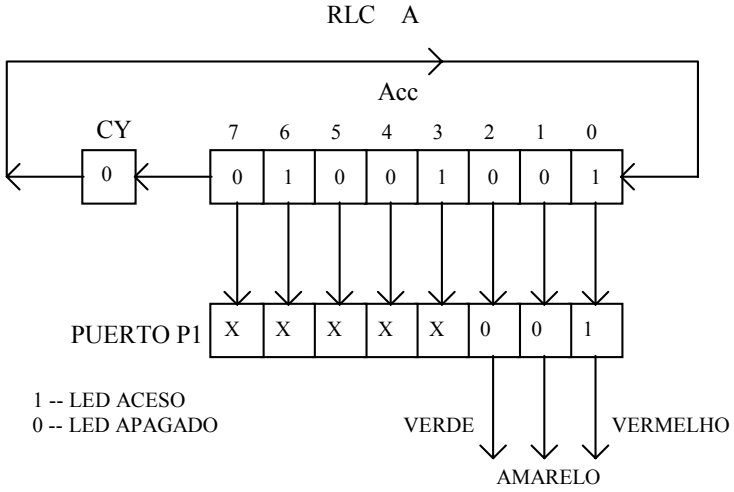

Figura 6.10. Uso do Acc e CY para acender leds em seqüência.

Se Acc e CY são inicializados com estes valores, basta uma rotação com CY (RLC A) para mudar o led que deve ficar aceso. A cada rotação escreve-se o conteúdo de Acc em P1.

```
;PISCA2.ASM
;
RTD             EQU             60000           ;65536>RTD>256
VERMELHO        EQU             P1.0
AMARELO         EQU             P1.1
VERDE           EQU             P1.2
;
                DEFSEG PROG, CLASS=CODE, START=0
                SEG             PROG
                ORG             RESET
INIC            MOV             A,#01001001B    ; 01  001  001
                CLR             C
AQUI            MOV             P1,A
                ACALL           RETARD
                RLC             A
                SJMP            AQUI
;
;ROTINA PARA GERAR UM RETARDO
RETARD          MOV             R7,# HIGH RTD
L1              MOV             R6,# LOW  RTD
L2              DJNZ            R6,L2
                DJNZ            R7,L1
                RET
                END
```

EXERCÍCIO 6.2. CHAVE1, CHAVE2 E CHAVE3

Usando os três leds como um contador de três bits, construir um programa que conte os acionamentos da chave SW2 (a contagem será observada nos três leds). Note que a chave não necessita de pull-up porque usa o pull-up interno da porta. A figura 6.11 ilustra as conexões.

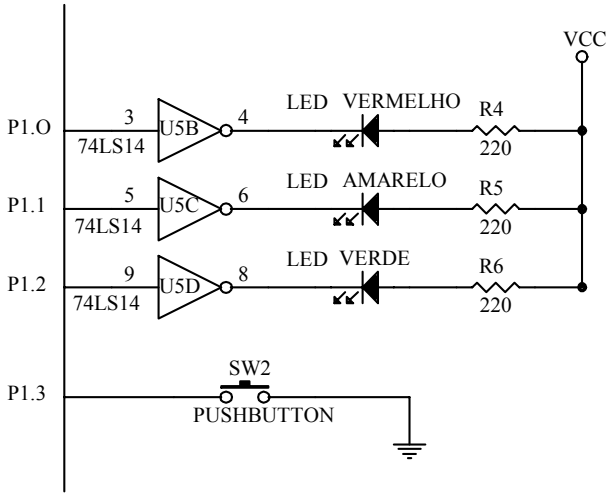

Figura 6.11. Conexões de leds e chave no circuito de práticas.

No primeiro programa será usada uma solução simples, que tem o problema de bouncing.

```
;CHAVE1.ASM
;
SW2             EQU             P1.3
;
                DEFSEG PROG, CLASS=CODE, START=0
                SEG             PROG
                ORG             RESET
INIC            CLR             A               ;ZERAR CONTADOR
                MOV             P1,A            ;APAGAR LEDS
                SETB            SW2             ;PROGRAMAR P1.3 COMO ENTRADA
AQUI1           JB              SW2,AQUI1       ;AGUARDAR ACIONAMENTO
                INC             A
                MOV             P1,A            ;DAR SAÍDA AO CONTADOR
                SETB            SW2             ;GARANTIR P1.3 COMO ENTRADA
AQUI2           JNB             SW2,AQUI2       ;AGUARDAR LIBERAR CHAVE
                SJMP            AQUI1
                END
```

No segundo programa é apresentada uma solução para eliminar o bouncing utilizando retardos. Ao detectar uma transição na chave, o programa aguarda um tempo para que se extinga o bouncing. O tempo que se deve aguardar é determinado de forma empírica.

```
;CHAVE2.ASM
;
; NESTE PROGRAMA SE PRETENDE ELIMINAR O BOUNCING COM RETARDOS
; O VALOR DO RETARDO É PARTICULAR PARA CADA CIRCUITO
; (DEPENDE DO TAMANHO DOS CABOS, QUALIDADE DA CHAVE, ETC)
; RECOMENDA-SE: DETERMINAR EMPIRICAMENTE O MELHOR RETARDO PARA CADA CIRCUITO
;
RTD             EQU             500             ;RTD>256
SW2             EQU             P1.3
;
                DEFSEG PROG, CLASS=CODE, START=0
                SEG             PROG
                ORG             RESET
INIC            CLR             A               ;ZERAR CONTADOR
                MOV             P1,A            ;APAGAR LEDS
                SETB            SW2             ;PROGRAMAR P1.3 COMO ENTRADA
AQUI1           JB              SW2,AQUI1       ;AGUARDAR ACIONAMENTO
                INC             A
                MOV             P1,A            ;DAR SAÍDA AO CONTADOR
                SETB            SW2             ;GARANTIR P1.3 COMO ENTRADA
                ACALL           RTD             ;ELIMINAR BOUNCING
AQUI2           JNB             SW2,AQUI2       ;AGUARDAR LIBERAR CHAVE
                ACALL           RTD             ;ELIMINAR BOUNCING
                SJMP            AQUI1
;
;ROTINA PARA GERAR UM RETARDO
RETARD          MOV             R7,# HIGH RTD
L1              MOV             R6,# LOW  RTD
L2              DJNZ            R6,L2
                DJNZ            R7,L1
                RET
                END
```

A solução de aguardar um tempo depois de detetar uma transição na chave funciona bem, mas oferece duas principais desvantagens. A primeira é que se aguarda um intervalo de tempo fixo, ou seja, se a chave de baixa qualidade é trocada por uma de melhor qualidade e que possua pouco bouncing, tem-se que mudar o programa. A segunda desvantagem é sua vulnerabilidade a ruídos pois estes podem provocar acionamentos indevidos. O programa chave 3, apesar de ser simples, soluciona estes problemas. Usam-se duas rotinas que detectam transições de 0→1 e de 1→ 0; o que se especifica é quanto tempo a chave deve estar em 0 ou em 1 para que se considere o acionamento válido (para que se considere o fim do bouncing).

```
;CHAVE3.ASM
;
; SOLUCAO MAIS EFICIENTE PARA ELIMINAR O BOUNCING COM RETARDOS
; USAM-SE ROTINAS QUE AGUARDAM A ESTABILIZACAO DA CHAVE
; HA UMA PARA AS TRANSIÇÕES DE 0 PARA 1 E OUTRA DE 1 PARA 0
; ESTA TECNICA SE ADAPTA MELHOR AS VARIACOES ENTRE OS CIRCUITOS
;
RTD             EQU             100
SW2             EQU             P1.3
;
                DEFSEG PROG, CLASS=CODE, START=0
                SEG             PROG
                ORG             RESET
INIC            CLR             A               ;ZERAR CONTADOR
                MOV             P1,A            ;APAGAR LEDS
                SETB            SW2             ;PROGRAMAR P1.3 COMO ENTRADA
AQUI            ACALL           RBT_1_0         ;TRANSICAO LIMPA DE 1 PARA 0
                INC             A
                MOV             P1,A            ;DAR SAÍDA AO CONTADOR
                ACALL           RBT_0_1         ;TRANSICAO LIMPA DE 0 PARA 1
                SJMP            AQUI
;
;ELIMINAR BOUNCING NAS TRANSIÇÕES DE 0 PARA 1
RBT_0_1         MOV             R7,#RTD
LB1             JNB             SW2,RBT_0_1
                DJNZ            R7,LB1
                RET
;
;ELIMINAR BOUNCING NAS TRANSIÇÕES DE 1 PARA 0
RBT_1_0         MOV             R7,#RTD
LB2             JB              SW2,RBT_1_0
                DJNZ            R7,LB2
                RET
                END
```