8051
Portas de Entrada e Saída

Microcontrolador 8051

- Possui 32 bits de entrada e saída;
- Podem ser acessados individualmente ou como portas de 8 bits;
- O acesso é bastante simples, pois cada porta possui apenas um registrador associado;
- Para se acionar um pino, basta escrever no bit correspondente neste registrador;
- Para se ler o estado de um pino basta ler o bit correspondente deste registrador;

Microcontrolador 8051

- Os registradores que controlam estes bits são:
  - P0 (endereço 0x80);
  - P1 (endereço 0x90);
  - P2 (endereço 0xA0);
  - P3 (endereço 0xB0);
- Estes registradores são bit endereçáveis;
- Também é possível acessar estes bits com instruções lógicas, aritméticas, lógicas, booleanas e de controle;

Microcontrolador 8051

- O *port* 0 possui função multiplexada, podendo atuar como barramento de dados e a parte baixa dos endereços;
- O *port* 2 também possui função indexada, podendo atuar como a parte alta do barramento de endereços;
- O *port* 1, originalmente,não possuía função alternativa, porém alguns modelos apresentam algumas funções multiplexadas (como é o caso do 89S52);
- O *port* 3 possui funções alternativas individuais para cada bit;

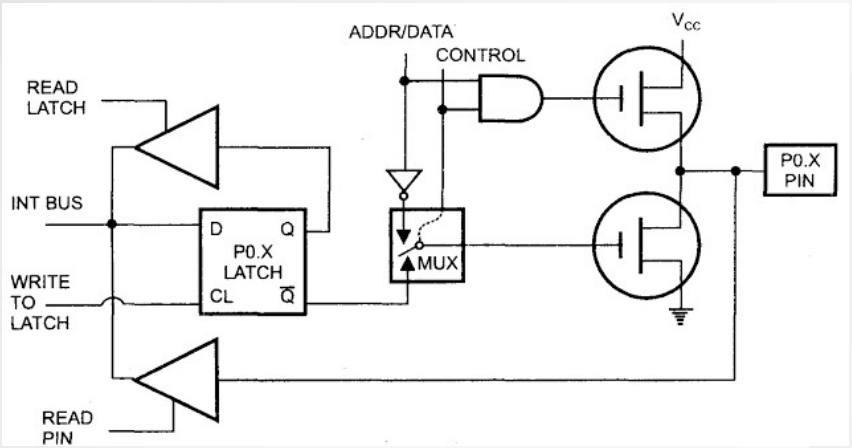
Microcontrolador 8051

- As funções alternativas do *port* 3 são:
  - P3.0: RXD/Data;
  - P3.1: TXD/Clock;
  - P3.2: INT0;
  - P3.3: INT1;
  - P3.4: T0;
  - P3.5: T1;
  - P3.6: $\overline{WR}$;
  - P3.7: $\overline{RD}$;

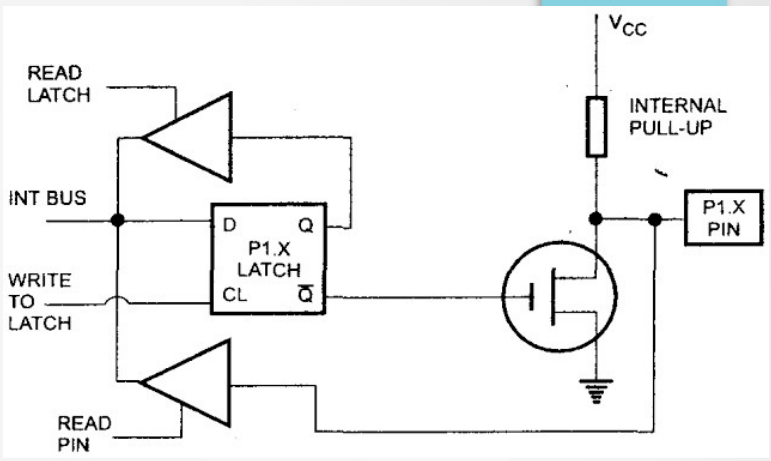
Microcontrolador 8051

- As funções alternativas do *port* 1, para o 89S52 são:
  - P1.0: T2;
  - P1.1: T2 EX;
  - P1.5: MOSI;
  - P1.6: MISO;
  - P1.7: SCK;

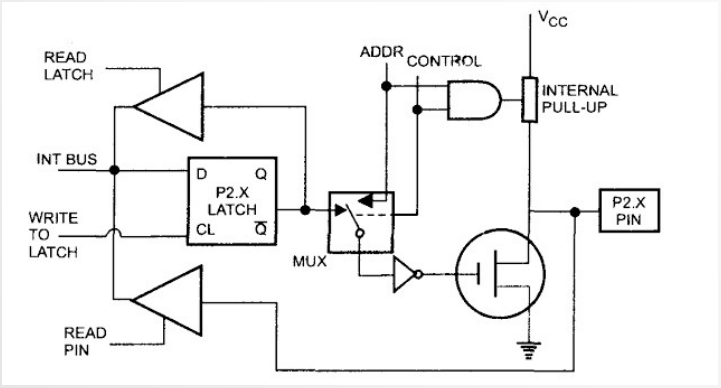
Microcontrolador 8051



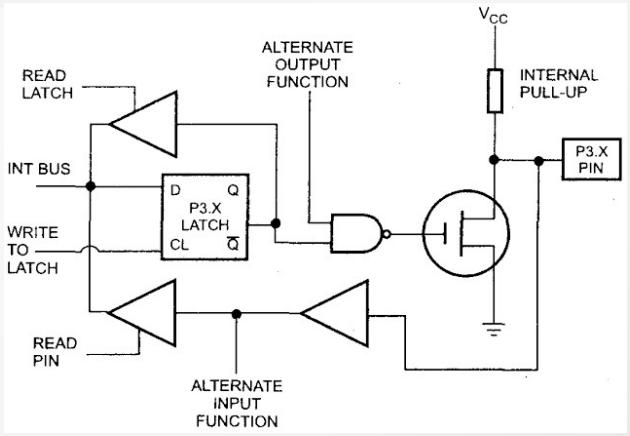
Port 0



Port 1



Port 2

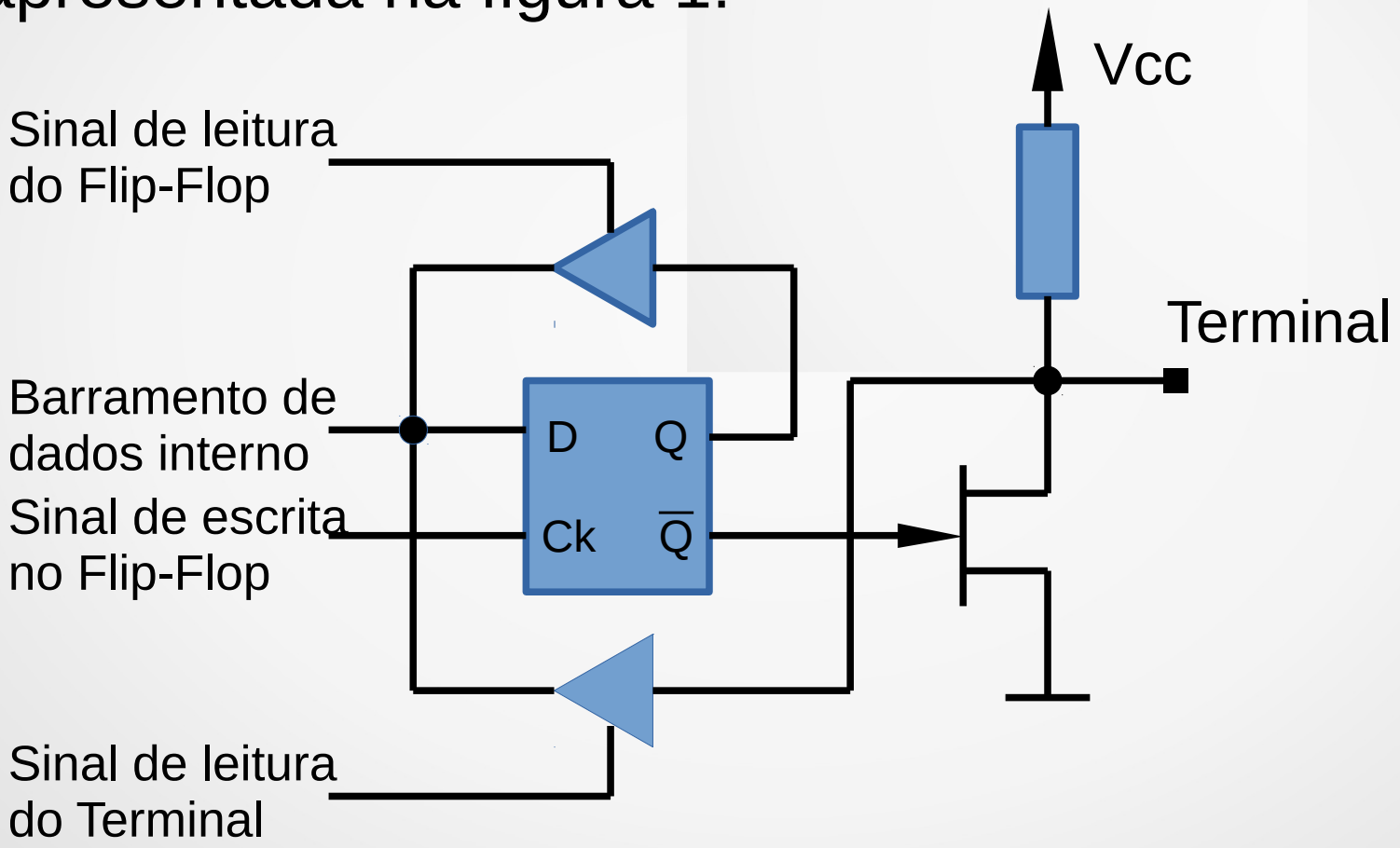


Port 3

Fonte: http://mediatoget.blogspot.com.br/2013/04/inputoutput-pins-ports-circuits-of.html

Microcontrolador 8051

- A estrutura simplificada de um terminal do *port* 1 é apresentada na figura 1:



Microcontrolador 8051

- Quando ocorre um escrita no pino, o bit chega à entrada D do flip-flop através do barramento de dados interno e, depois de um pulso de clock, chega a saída que controla o FET de saída;
- A leitura pode ocorrer de dois pontos diferentes:
  - As instruções que leem e alteram o conteúdo do registrador leem o valor do flip-flop (INC, DEC, CPL, JBC, DJNZ, ANL, ORL, XRL);
  - As demais instruções leem o valor do pino;

Microcontrolador 8051

- Para atuar como entrada, um bit de nível alto deve ser escrito no flip-flop. Nesta situação, o FET permanece cortado e o valor presente no pino pode ser lido;
- Os *ports* 1, 2 e 3 possuem resistores de pull-up internos;
- O *port* 0 não possui resistor de pull-up, ou seja o dreno está “flutuando”;
- Os *ports* 1, 2 e 3 podem acionar apenas uma carga TTL e o *port* 0 pode acionar até 2 cargas TTL;

Microcontrolador 8051

- Exemplos de acesso aos *ports*:
  - Escrita de um byte no *port* 1:

```
mov P1, #33
```

  - Leitura de um byte do *port* 2:

```
mov A, P2
```

  - Escrita de um bit no pino P2.0:

```
mov P2.0, C
```

  - Verificando o estado do pino P1.3:

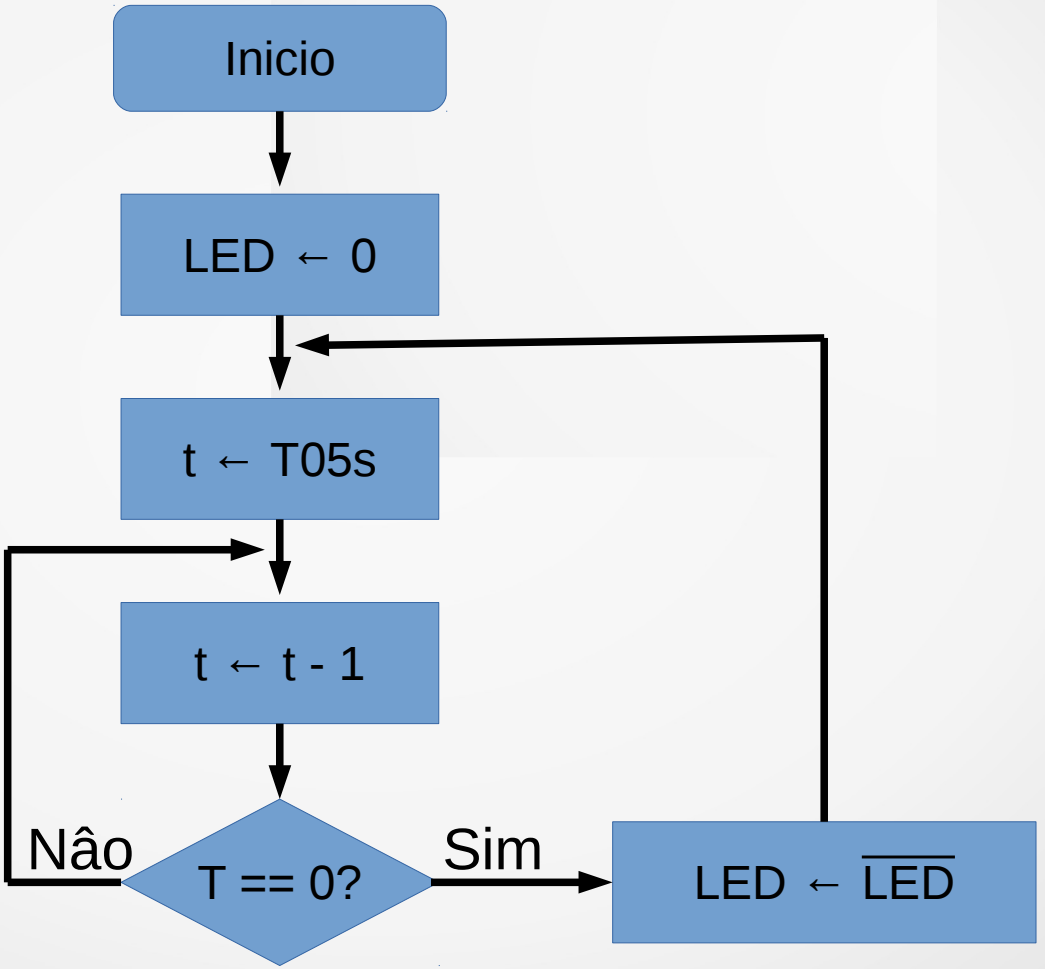
```
jb P1.3, SETADO
```

8051
Exercícios

Microcontrolador 8051

- Faça um programa em assembly para o 89S52 onde um led ligado entre o pino P0.0 e o Vcc (através de um resistor de 330ohms ou 470ohms) mude de estado (aceso e apagado) a cada 500us.

Fluxograma



Criando um cabeçalho:

```
;*************************************************;
; Exemplo simples de programacao em assembly para ;
; o microcontrolador 89S52.                       ;
; Fazer um led piscar pelo pino P0.1 a uma freq.  ;
; de 0,5Hz                                        ;
; Gabriel Kovalhuk                    15/10/2014 ;
;*************************************************;
```

Definindo algumas constantes:

```
LED   equ   P0.1      ; define um nome para P0.1
T     equ   R0        ; define a variavel T em R0
T05   equ   0F9h        ; define o valor de T05 para
                      ; gerar um tempo de 0,5s
```

Definindo o endereço de inicio do programa:

```
Org 0000        ; define o endereco do inicio
                ; do programa
```

O programa

```
        org 0000        ; define o endereco do
                        ; inicio do programa
RESET:  clr   EA        ; desabilita todas as
                        ; interrupcoes
        clr   LED   ; LED ← 0
LOOP:   mov   T, #T05s; inicializa contador de tempo
TEMPO:  djnz  T, TEMPO; enquanto T>0 decrementa T
        cpl     LED   ; complementa o estado de LED
        jmp     LOOP  ; retorna para mais um ciclo
        end
```

Descrição de cada instrução

Instrução CLR (clear)

(1)Instrução booleana:

CLR bit: **zera o bit especificado**

CLR C: **zera o flag C**

(2)Instrução lógica:

CLR A: **zera o conteúdo do acumulador**

Instrução MOV (move)

MOV <destino>,<origem>: copia o conteúdo da <origem> para o <destino>

origem e destino pode ser qualquer endereço de memória interna, tanto da área do usuário quanto da área dos registradores de função especial.

Exemplos:

```
mov A, R0  ;copia o conteúdo de R0 para o acumulador

mov A, R0  ;copia o conteúdo de R0 para o acumulador
```

Instrução DJNZ (decrement and jump if not zero)

DJNZ Rn, endereco: decrementa o valor do registrador Rn e salta para o **endereco** se Rn == 0;

DJNZ direto, endereco: decrementa o valor da posição de memória em **direto** e salta para o **endereco** se **direto** == 0;

(1) endereco é um endereço da **memória de programa**, relativo ao endereço atual. Esta instrução pode saltar somente 127 bytes acima e 128 bytes abaixo do endereço atual;

(2) **Rn**, onde é um valor entre 0 e 7, é um dois 8 registradores do banco de registradores em uso;

(3) direto é um endereço da **memória de dados interna** do 8051;

Instrução CPL (complement)

(1) Instrução lógica:

CPL A: complementa (inverte) todos os bits do registrador A;

(2) Instrução booleana:

CPL C: complementa (inverte) o bit *carry* do registrador PSW;

CPL bit: complementa (inverte) o bit endereçável;

Instrução JMP (jump)

JMP endereço: salta para o endereço especificado. Esta é a forma genérica que os compiladores (montadores) aceitam. Na prática existem 3 tipos diferentes de instrução jump:

(1)AJUMP (Absolut JUMP): salta para o endereço absoluto indicado. O endereço desta instrução possui 11 bits;

(2)LJUMP (Long JUMP): salta para o endereço longo especificado. O endereço desta instrução possui 16 bits;

(3)SJUMP (Short JUMP): salta para o endereço relativo especificado. Note que o endereço relativo vai de -128 a +127 bytes a partir do endereço atual.