

Microcontroladores

Arquitetura - Parte 2

Prof. Guilherme Peron

Prof. Ronnier Rohrich

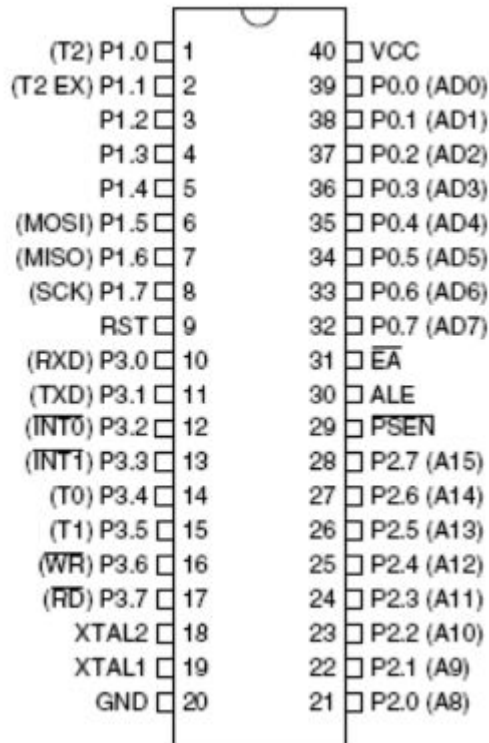
Prof. Rubão

Encapsulamentos/Pinagem

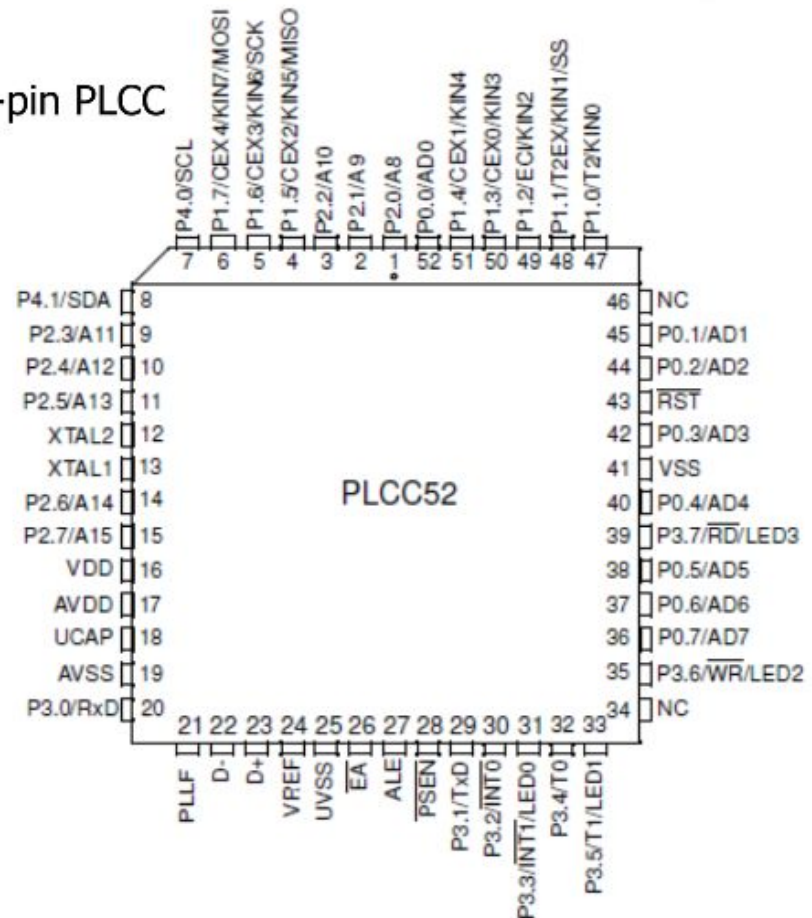
AT89S52

AT89C5131A

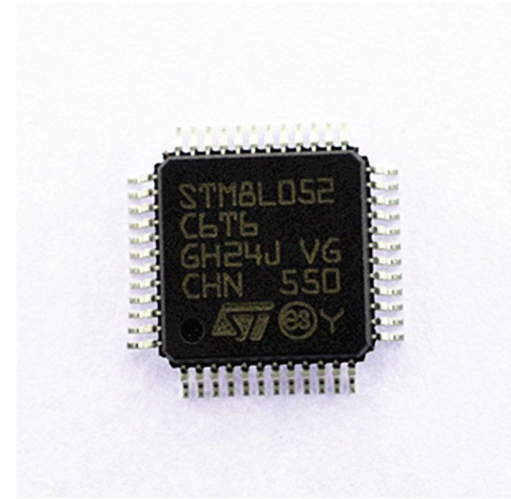
40-lead PDIP



52-pin PLCC



Outros Fabricantes



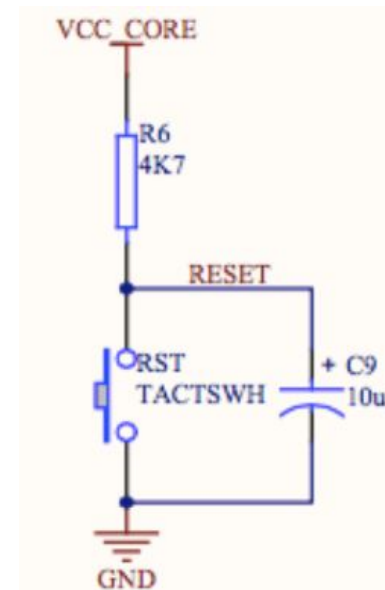
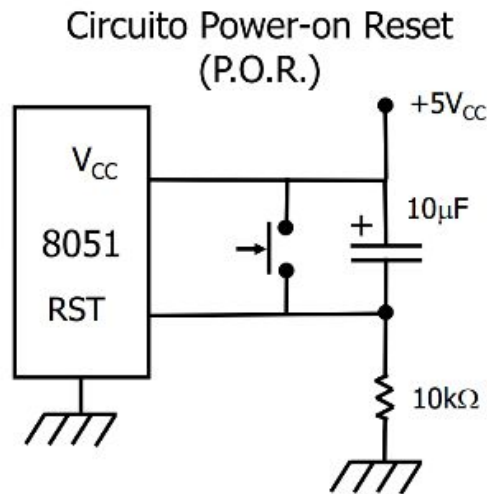
Reset



???

Reset

- No **8051** genérico o pino (FÍSICO) **RST** é um pino ativo em **ALTO**, devendo ser levado a nível lógico 1 por dois ou mais ciclos de máquina durante a energização do chip.
- No **AT89C5131A**, o pino **\RST** é um pino ativo em **BAIXO**, devendo ser levado a nível lógico 0.



Reset

- A operação do **reset** consiste em forçar alguns registradores a estados definidos:

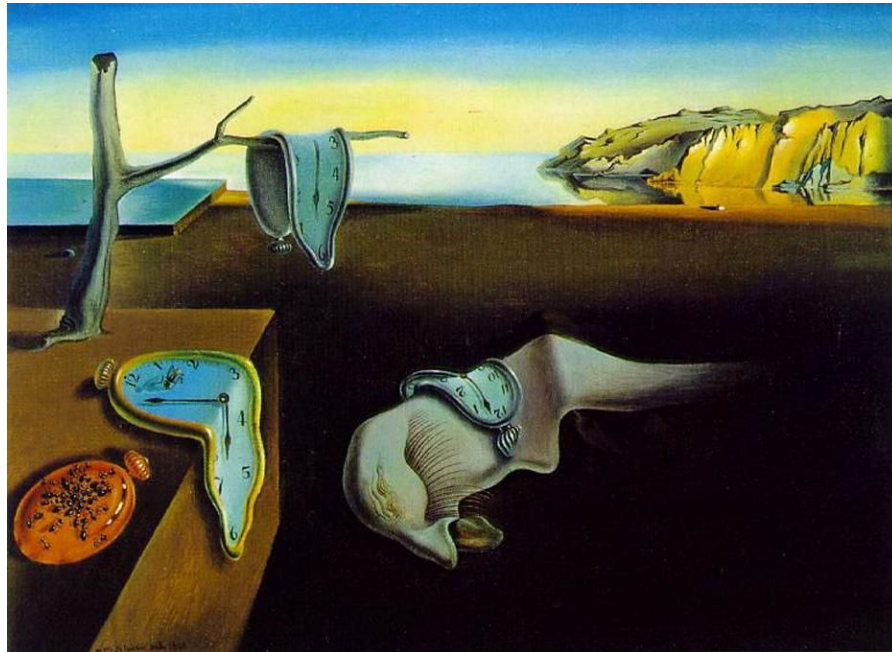
Registrador	Valor Após o <i>Reset</i>
A, B, PSW, DPTR, PC Registros de Temporizadores/Contadores	Zerados
Ports (P0, P1, P2, P3)	FFh
Pilha (SP)	07h
RAM Interna	RESET Forçado - Não altera RESET Alimentação - Aleatório
SCON	00h
SBUF	XXXXXXXXb
PCON	0XXXXXXXXb HMOS 0XXX0000b CMOS
IE e IP	0XX00000b e XXX0000b

Reset

Resumindo...

- 1) Existe um circuito ligado ao pino de RESET (verificar na placa de vocês qual circuito é esse)!
- 2) Quando “o chip é ligado” este circuito é acionado ou por vontade do dono da placa (assim como **nóiz faiz** quando trava nosso PC)!

Clock

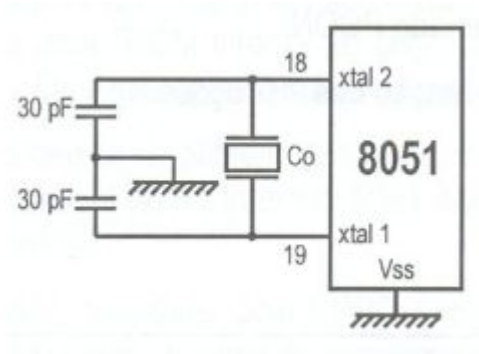


Clock

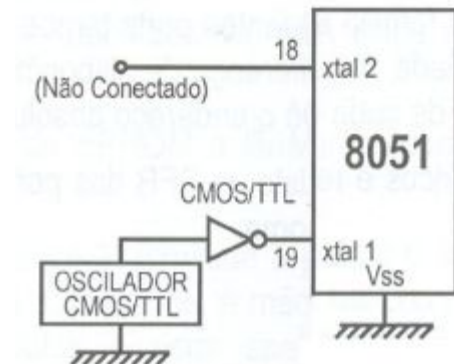
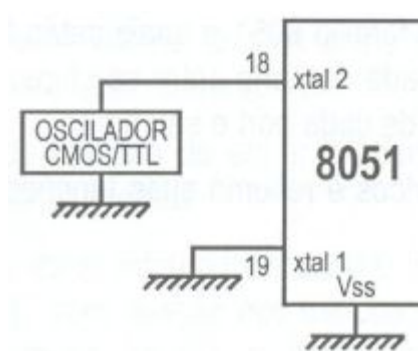
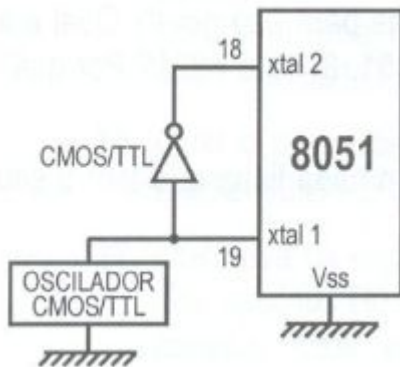
- É o relógio interno do microprocessador para execução sequencial de qualquer atividade interna ou externa à máquina.
- A ligação externa pode ser com um cristal ou oscilador.
- Alguns microcontroladores têm frequência mínima de clock para funcionar (por exemplo 3,5MHz)
- **Isso é rápido ou devagar??? Pense no seu Computador....**

Clock

Com cristal oscilador

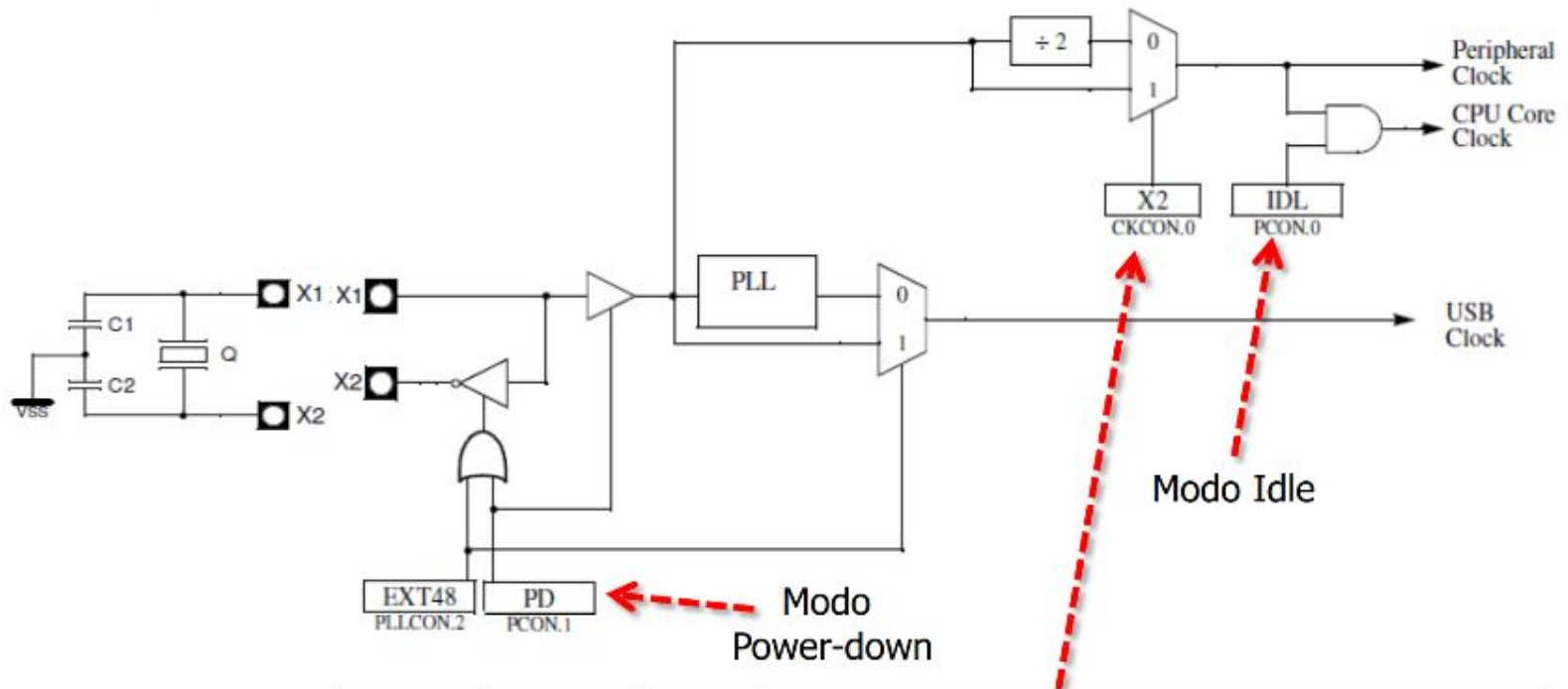


Com oscilador externo



Clock

- Enquanto isso no AT89C5131A...



0	X2	System Clock Control bit Clear to select 12 clock periods per machine cycle (STD mode, $F_{CPU} = F_{PER} = F_{OSC}/2$). Set to select 6 clock periods per machine cycle (X2 mode, $F_{CPU} = F_{PER} = F_{OSC}$).
---	----	---

Modos *Idle* e *Power-down*

- *Idle*:
 - CPU inativa, periféricos funcionando. **MDI** e **SFR** são preservados;
 - Formas de sair deste modo:
 - **Reset**;
 - Através de alguma **interrupção** que esteja habilitada;
 - Ativado pelo bit **IDL** (bit 0) do registrador **PCON** (87h);
 - **lcc=6,5mA** (AT89S52), **lcc=0,3 x F(MHz)+5mA** (AT89C5131A).
- *Power-down*
 - O gerador de *clock* é desligado e tudo para;
 - MDI e SFR são preservados.
 - Forma de sair deste modo:
 - **Reset**.
 - Ativado pelo bit **PD** (bit 1) do registrador **PCON** (87h)
 - **lcc=50 μA** (AT89S52), **lcc= 100 μA** (AT89C5131A).

PCON	SMOD	-	-	-	GF1	GF0	PD	IDL
87h								

Curiosidade

“Durante o projeto e execução de Hardware sempre procure posicionar o circuito do cristal e dos capacitores o mais próximo do microprocessador, assim evita-se problemas de oscilação instável”

Ports



Ports

- Capacidade de corrente:
 - P0: (equivalente a 8 x LS TTL), max. 26 mA
 - P1, P2, P3: (equivalente a 4 x LS TTL), max 15 mA
 - Máxima corrente de saída por pino: 10 mA
- Configuração:
 - P0: quase-bidirecional, isto é, precisa de resistores de *pull-up*
 - P1, P2, P3: bidirecional

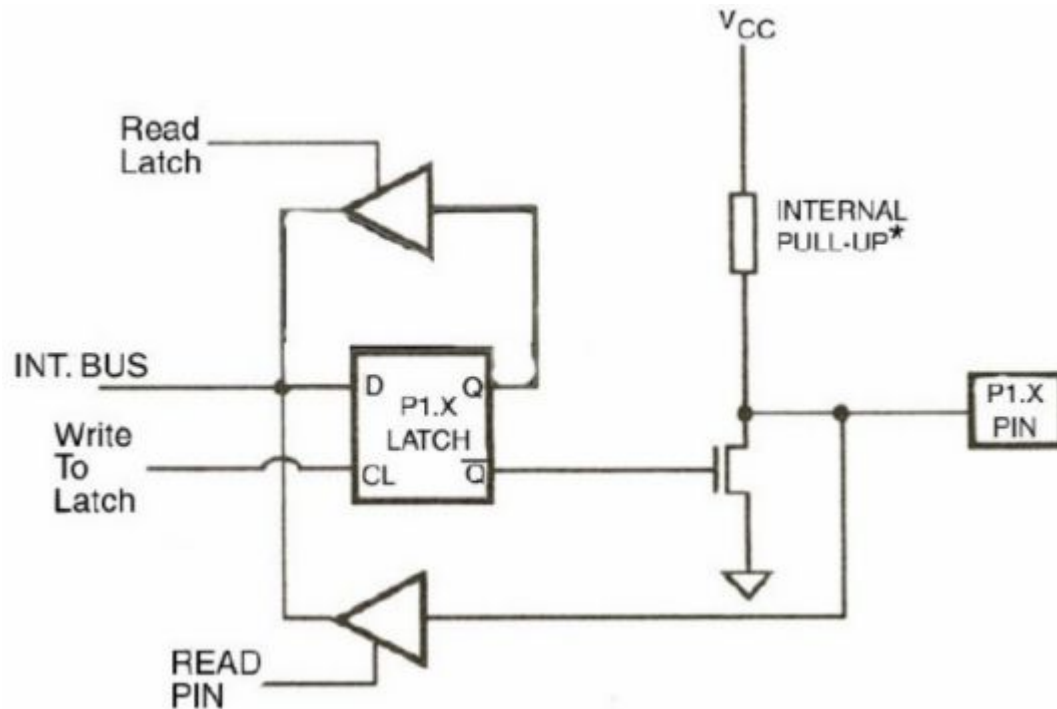
Resistor de Pull-up

O que é isso?

Ex: Você tem um pino configurado como entrada. Porém, não tem nada conectado nele, assim a “leitura” dele pode ser incerta (ficará flutuando). Este resistor garante que o pino está em baixo ou alto.

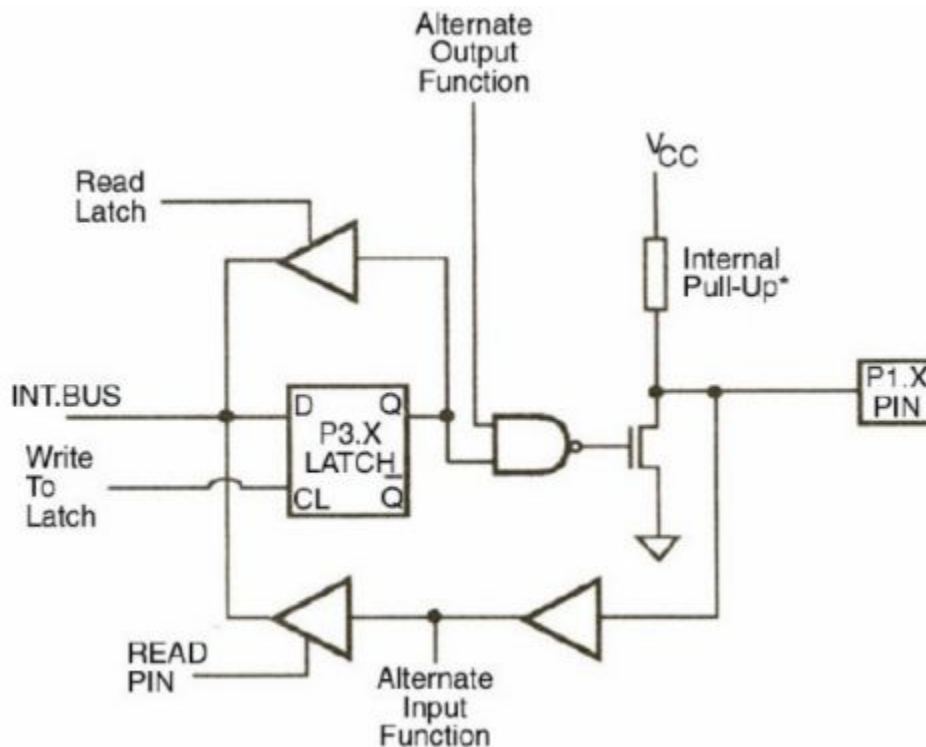
Port P1

- 8 bits com resistor de “*pull-up*” interno para I/O
- Fornece/Drena 1 carga TTL (4 LS TTL)



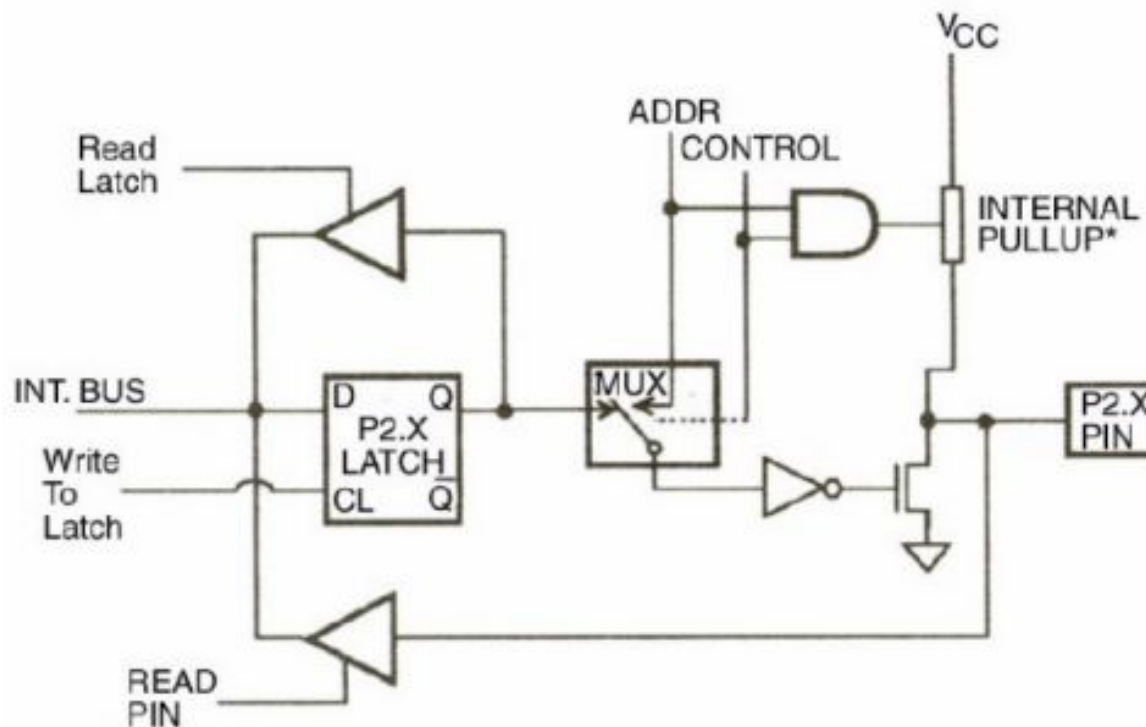
Port P3

- 8 bits com resistor de “*pull-up*” interno
- Para I/O ou funções especiais
- Fornece/Drena 1 carga TTL (4 LS TTL)



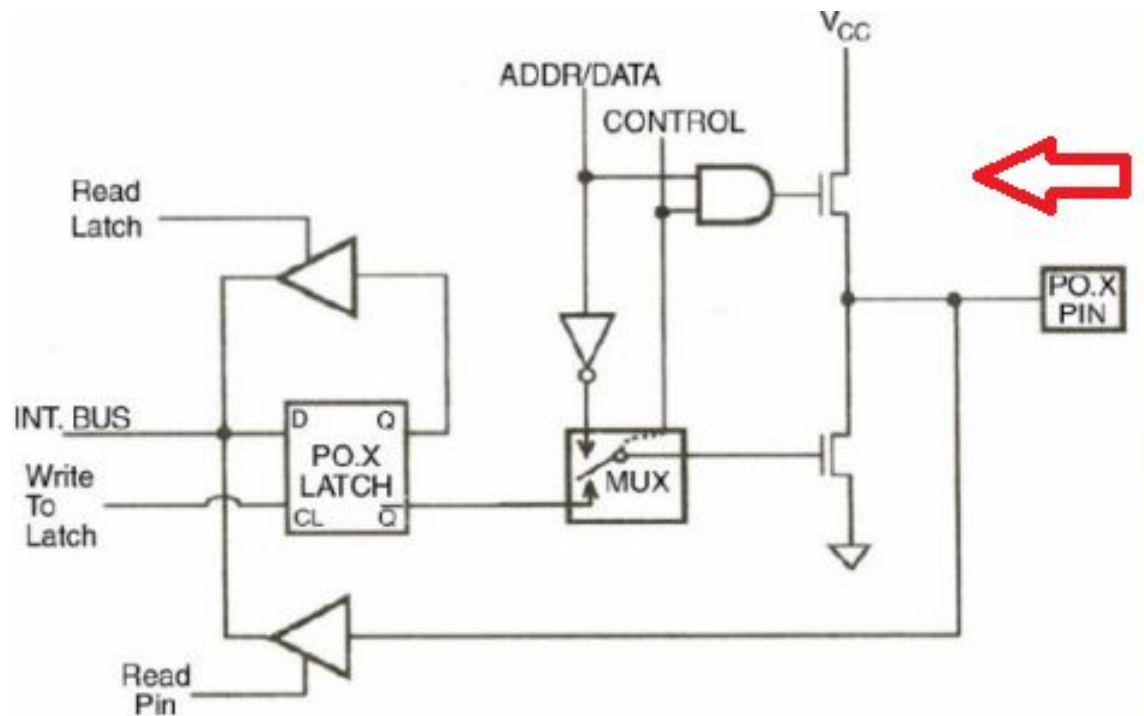
Port P2

- 8 bits com resistor de “*pull-up*” interno
- Para I/O ou MSB do end. mem. ext (A8 a A15)
- Fornece/Drena 1 carga TTL (4 LS TTL)



Port P0

- 8 bits com dreno/coletor aberto
- Para I/O ou LSB do end. mem. ext (A0 a A7) e os dados (D0 a D7)
- Fornece/Drena 2 cargas TTL (8 LS TTL)



Ports

- Considerações
 - Para programar as portas como entrada, basta escrever “1” no bit correspondente.
 - P0 não tem *pull-up*, flutua quando programado como entrada

Circuitos *drivers*

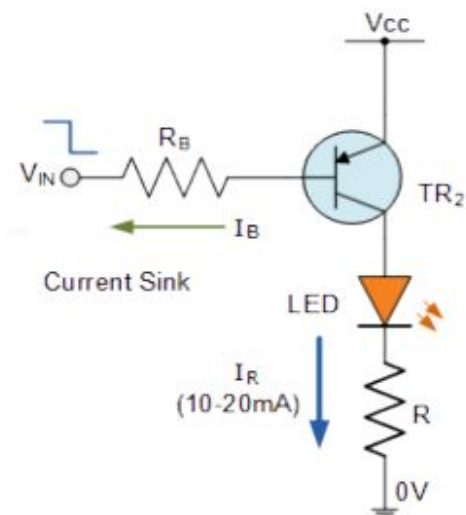
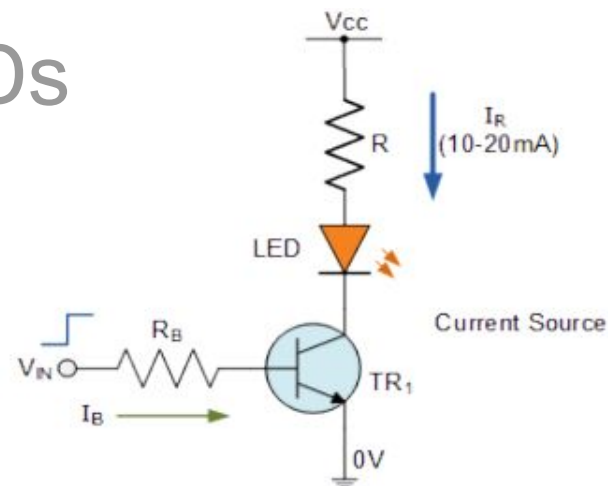
- Por exemplo *drivers* de LEDs
 - Dada a baixa corrente fornecida pelas portas, é recomendável utilizar circuitos que atuem como *drivers* para estes dispositivos.

Circuitos *drivers*



Circuitos *drivers*

- Por exemplo *drivers* de LEDs
 - Circuitos com transistores
 - Tarefa para casa!!!!
 - Calcular: R_C e R_B
 - Região corte-saturado



LEDs		
Cor do LED	Tensão em Volts (V)	Corrente em Miliamperes (mA)
Vermelho	1,8V – 2,0V	20 mA
Amarelo	1,8V – 2,0V	20 mA
Laranja	1,8V – 2,0V	20 mA
Verde	2,0V – 2,5V	20 mA
Azul	2,5V – 3,0V	20 mA
Branco	2,5V – 3,0V	20 mA

Circuitos *drivers*



Circuitos *drivers*

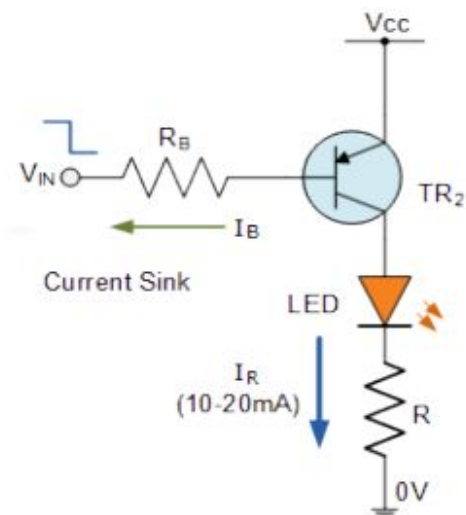
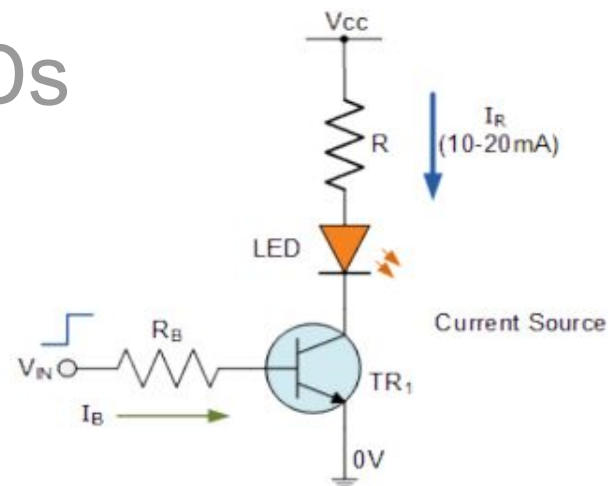
- Por exemplo *drivers* de LEDs
 - Circuitos com transistores
 - Calcular: R_C e R_B
 - Região corte-saturado

$$I_C = (V_{CC} - V_{LED} - V_{CE}) / R_C$$

$$I_B = I_C / \beta$$

$$I_B' = 5I_B$$

$$R_B = (V_{IN} - V_{BE}) / I_B'$$




LEDs		
Cor do LED	Tensão em Volts (V)	Corrente em Miliamperes (mA)
Vermelho	1,8V – 2,0V	20 mA
Amarelo	1,8V – 2,0V	20 mA
Laranja	1,8V – 2,0V	20 mA
Verde	2,0V – 2,5V	20 mA
Azul	2,5V – 3,0V	20 mA
Branco	2,5V – 3,0V	20 mA

Instruções com *Ports*

- Instruções que apenas lêem pinos

Instrução	Exemplo
JB	P1.0, LABEL1
JNB	P2.3, LABEL2
MOV	MOV A, P1

- Instruções do tipo *Read-Modify-Write*: instruções cujo resultado é colocado no *latch* do *port* (leem o *Latch*)
- Cuidado ao utilizar 

Instrução	Exemplo
INC	INC P1
DEC	DEC P3
CPL	CPL P1
JBC	JBC P1.0, #0ABh
DJNZ	DJNZ P1, #0ABh
ANL	ANL P0, A
ORL	ORL P0, A
XRL	XRL P1, #10h

Ciclo de Máquina

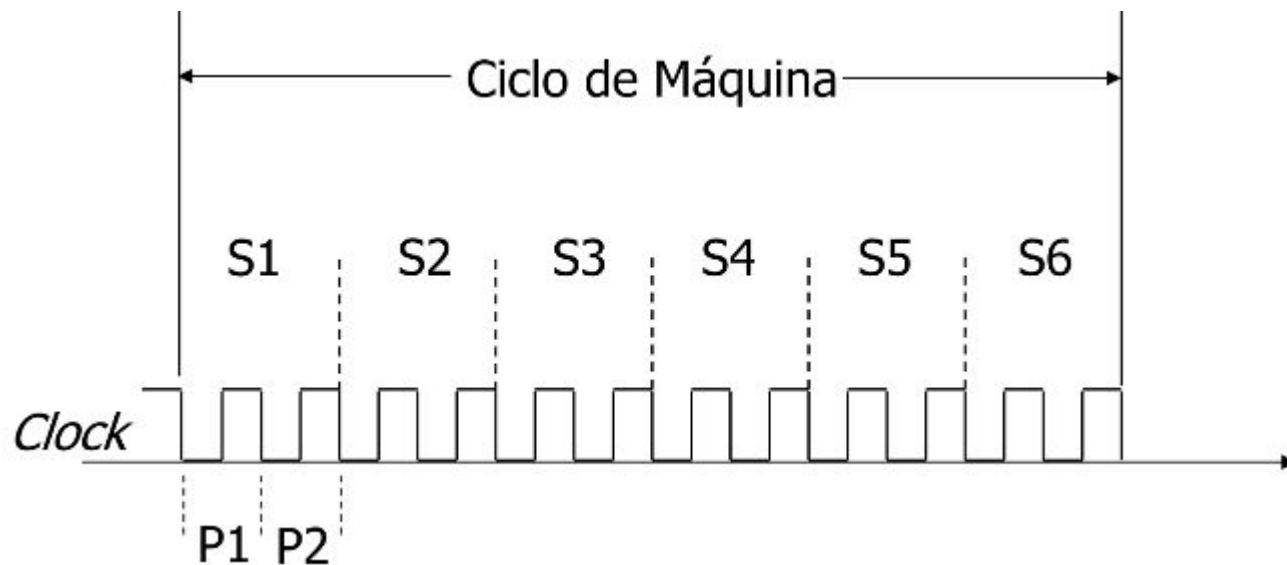
Ciclos de máquina

- O clock é o elemento que gera e controla os ciclos de trabalho da máquina. Cada ciclo de oscilação pode ser chamado de pulso (**P**).
- A cada dois pulsos (**P**) se caracteriza um estado (**S**)
- Uma sequência de seis estados, **S1** a **S6** corresponde a um ciclo de máquina.
- Todas as atividades do μ C são comandadas por esses pulsos e seus seis estados.

1 Ciclo de Máquina => 6 Estados => 12 Pulsos de Clock

Ciclos de máquina

- Todas as instruções do 8051 (padrão) são executadas em 1 ou 2 ciclos de máquina, exceto MUL e DIV que são 4 ciclos



Ciclos de máquina

- Eventos que ocorrem em um ciclo de máquina:
 - Obtenção do endereço de memória que contém a instrução;
 - Busca da instrução na MP (*Instruction Fetch*);
 - Decodificação da instrução;
 - Obtenção do endereço dos operandos;
 - Busca do operando na MP/MD (*Operand Fetch*);
 - Execução da operação;
 - Armazenamento do resultado na MD/registrador.

Ciclos de máquina

- Escrita e leitura nos *ports*:
 - Na escrita, o dado só é escrito no *latch* do *port* em **P2S6**, logo, só está disponível no pino em **P1S1** do próximo ciclo.
 - Na leitura, o pino é amostrado em **S5** e deve permanecer estável por um período maior do que um ciclo de máquina.

Ciclos de máquina

- Velocidade de processamento:
 - Considerando um cristal oscilador de 24 MHz (que é o caso do kit P51USB), o número de instruções (de 1 ciclo de máquina) realizadas por um AT89C5131A é:
 - $24.000.000 \text{ Hz} / 12 = \mathbf{2.000.000 \text{ instruções de 1}}$ ciclo de máquina cada
 - Estimativa mais realista para programas com instruções de 1 e 2 ciclos de máquina: média de **$\sim 1.334.000 \text{ instruções/segundo}$** .

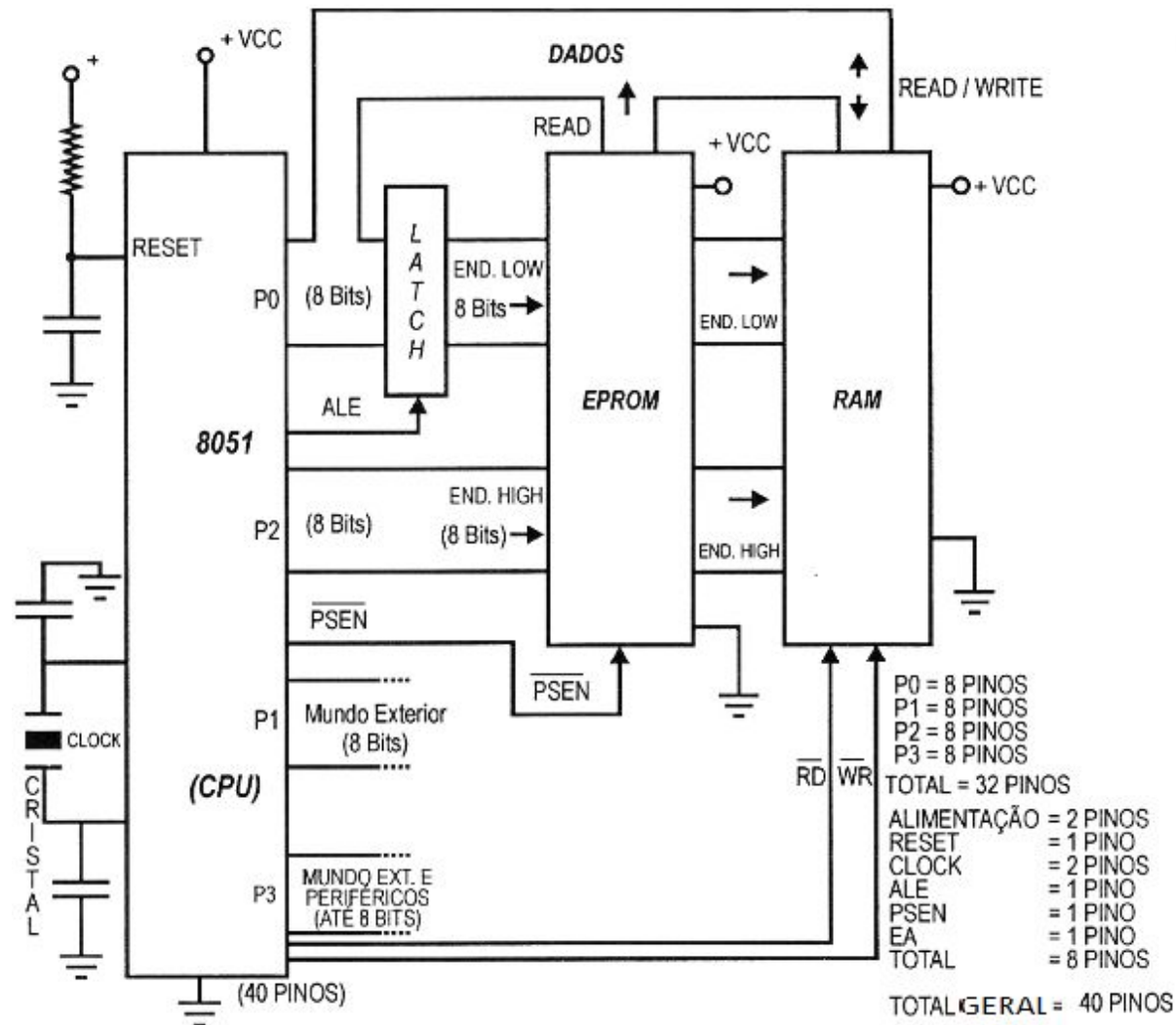
I/O Mapeado em Memória

I/O mapeado em memória

O que é isso?

R: É uma técnica de acessar periféricos de E/S como se estivessem em posições de memória. Para isso, é necessário um decodificador para selecionar o periférico na faixa de endereços adequada.

Interligação básica do 8051

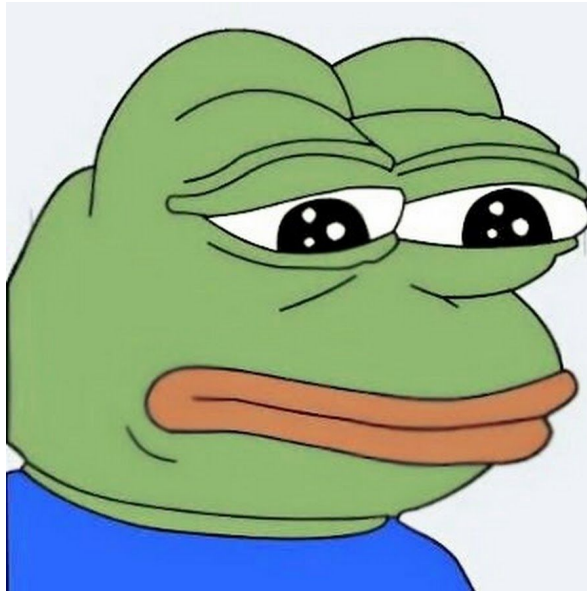


Comprometimento dos *ports*

- Vamos supor que os *ports* estão comprometidos
 - P0 e P2 para ROM e RAM externa;
 - P3 para periféricos internos;
 - P1 para A/D interno.
- Mas eu gostaria de utilizar um port para um teclado e um display de 7 segmentos.

Comprometimento dos *ports*

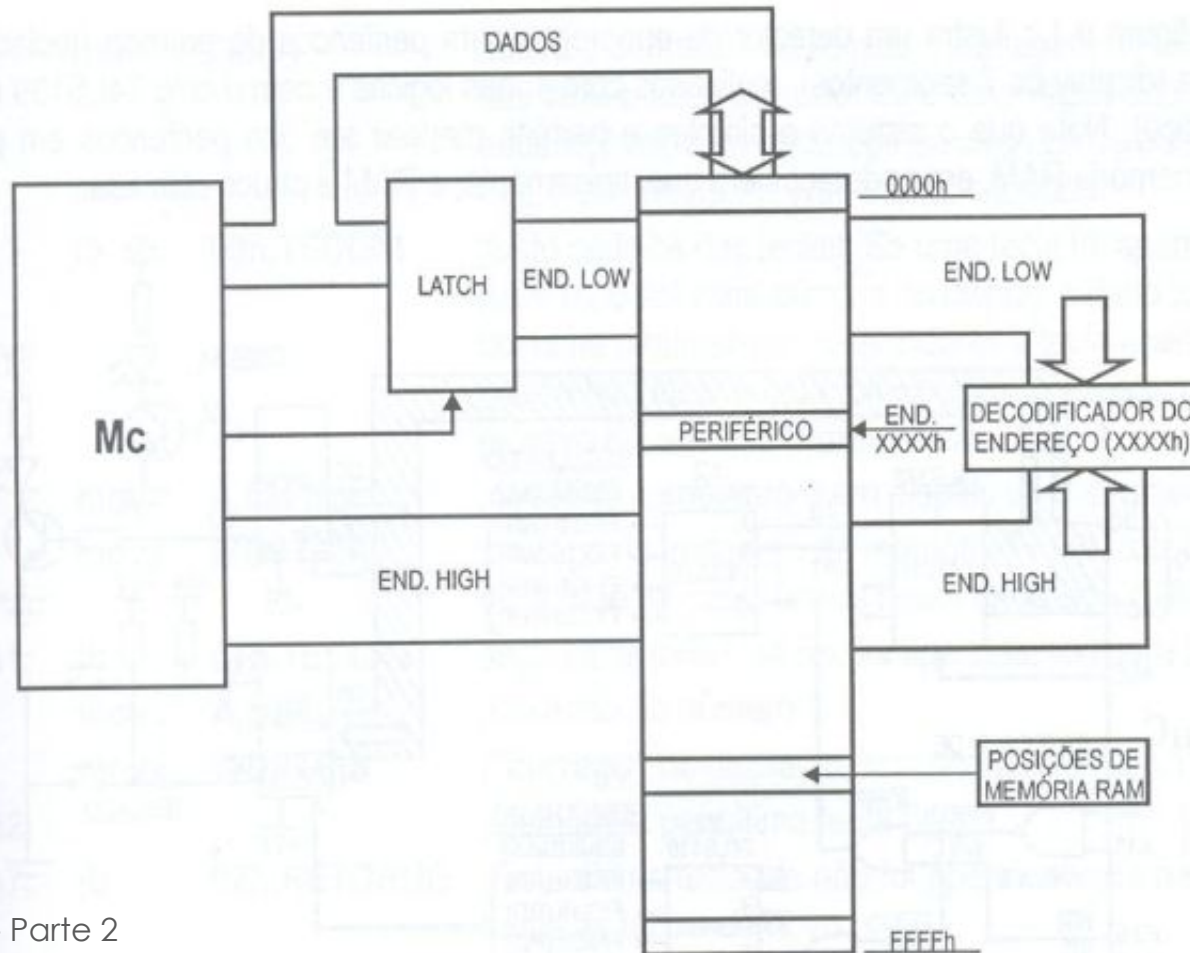
- Vamos supor que os *ports* estão comprometidos
 - P0 e P2 para ROM e RAM externa;
 - P3 para periféricos internos;
 - P1 para A/D interno.



E agora??

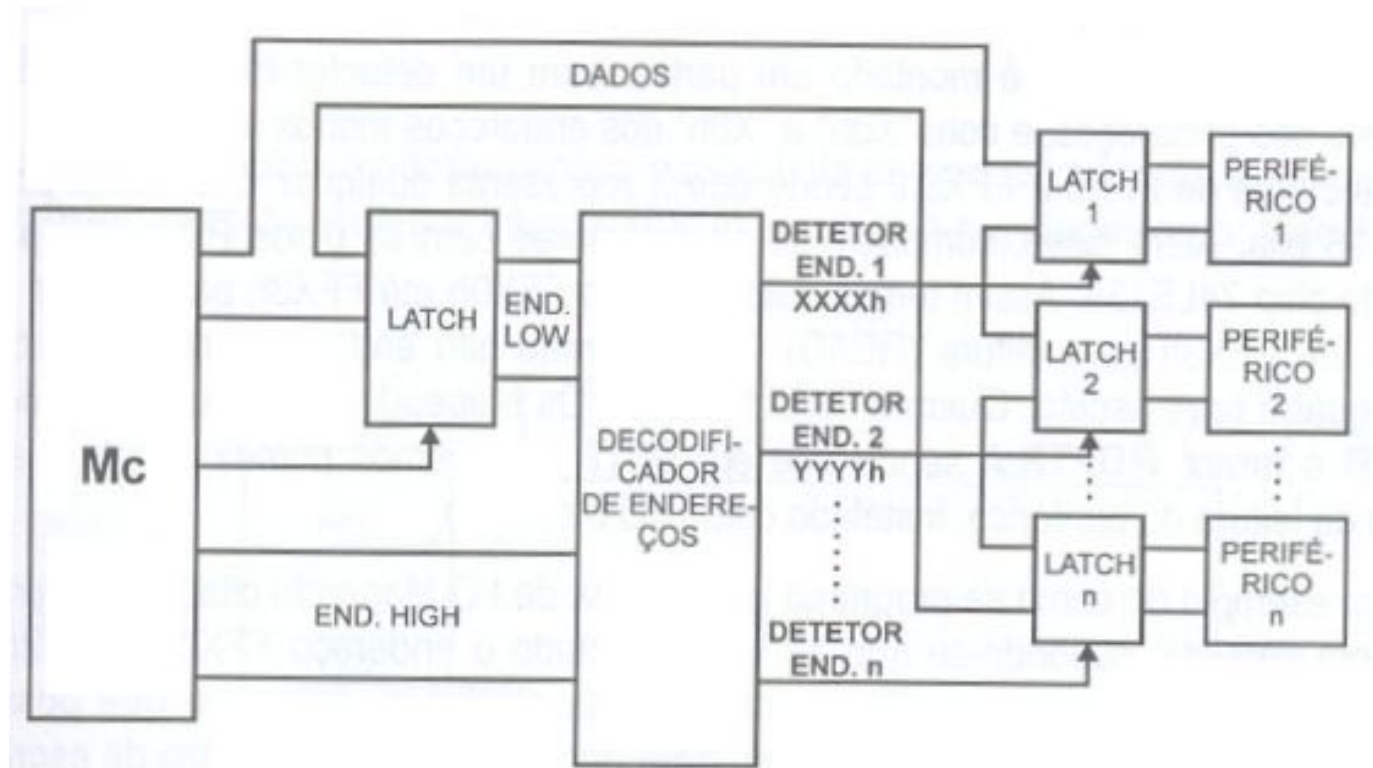
I/O Mapeado em Memória

- Utilizar de uma posição da RAM externa, para mapear um periférico, como sendo uma posição realmente.



I/O Mapeado em Memória

- Em termos mais práticos



- Cada periférico tem um endereço específico e o periférico pode ser do tipo somente entrada, somente saída ou entrada e saída.

I/O Mapeado em Memória

- **Exemplo:** Periféricos de entrada (teclado) e saída (*display* de 7 seg.)

