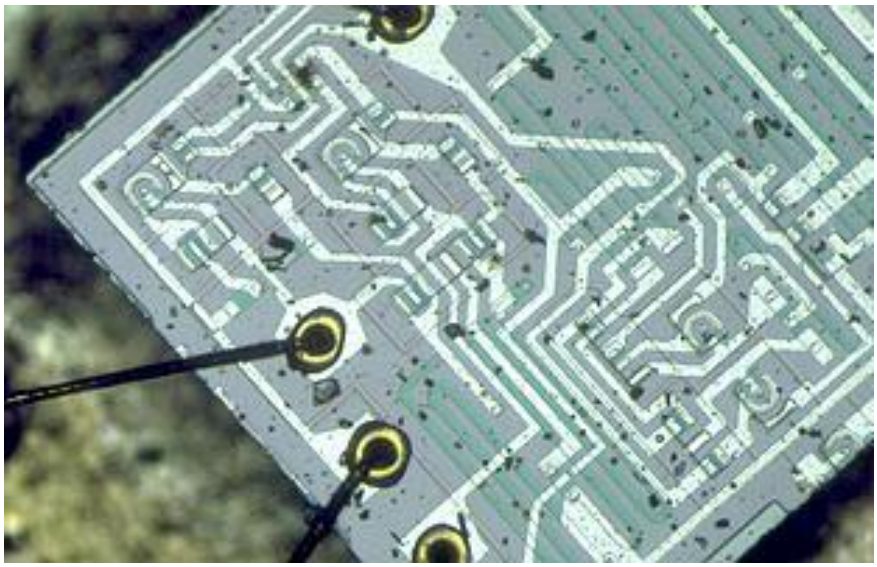
APONTAMENTOS SOBRE O 8051

© Prof. Eng° Luiz Antonio Vargas Pinto

Versão revista e ampliada em 01/07/2009
Revisada em 28/02/2006
Revisada em 18/11/2007
Revisada em 17/03/2011
Revisada em 13/01/2013



1. Introdução:

De forma geral, minhas apostilas iniciam com aulas ministradas, apontamentos e finalmente na forma digital como é o caso desta apostila.

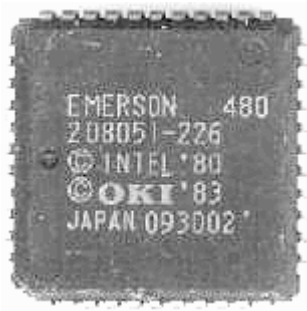
Longe de ser dono da verdade, erros podem ser constantemente cometidos. Assim, peço a sua colaboração, caro leitor, se houver alguma dúvida ou discrepância.

Há mais material sobre microcontrolador e eletrônica em geral em meu site

www.vargasp.com. Lá você também poderá me conhecer melhor e me contatar por email.

Desejo sucesso em seus estudos.

2. Foto

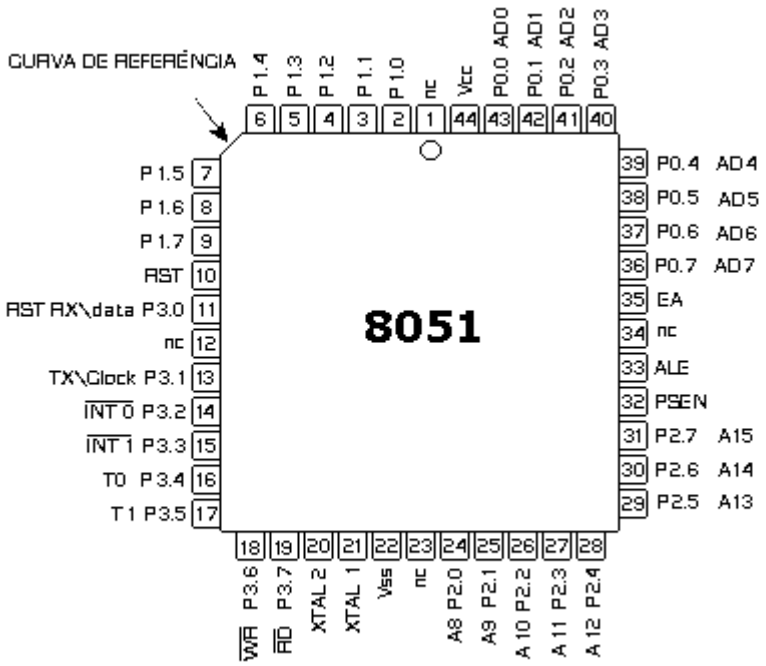


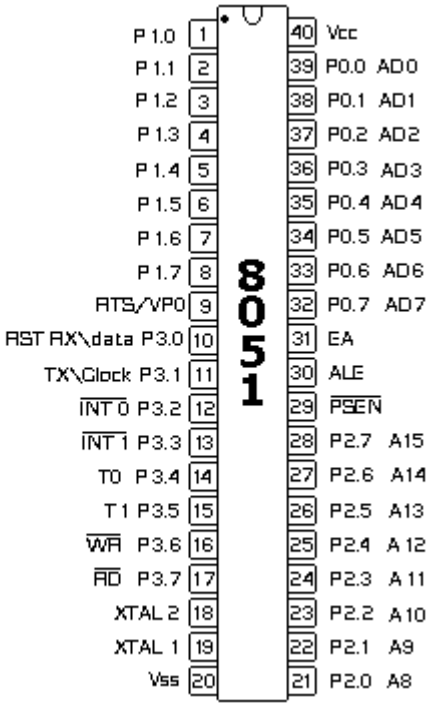
DIP 44

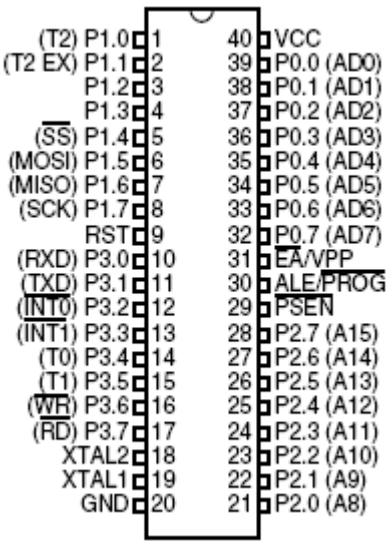
DIP 40 (foto do site www.cpu-world.com)

SMD

3. Pinagem

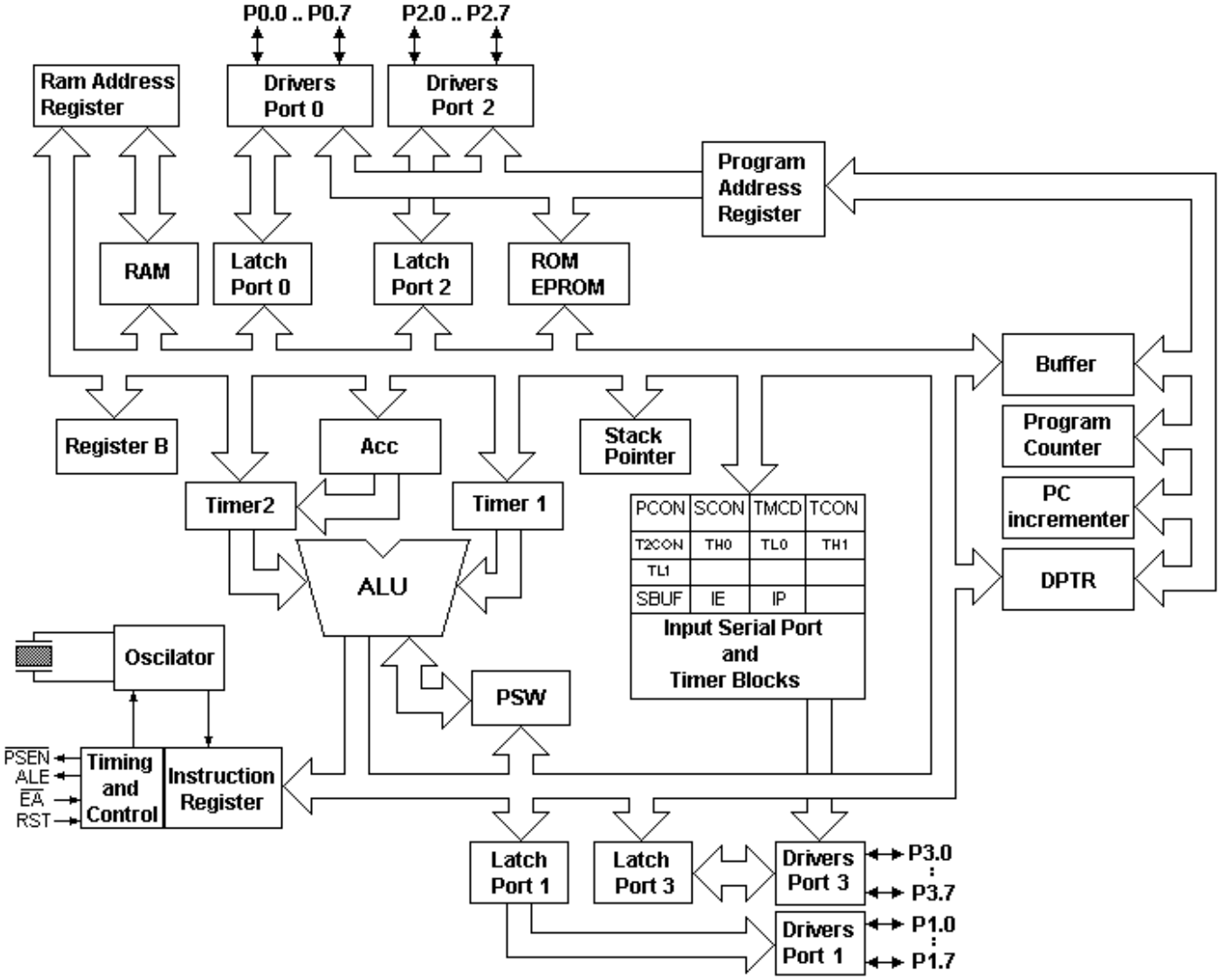




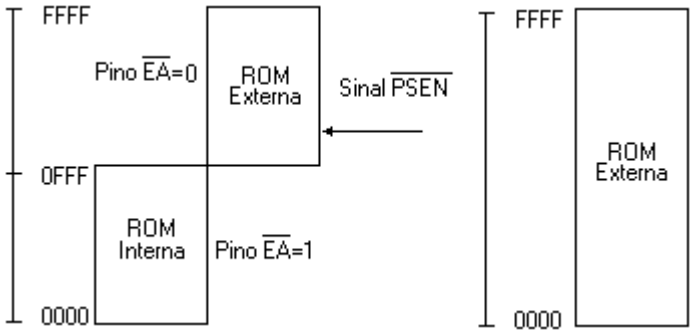


ATMEL AT89S8253

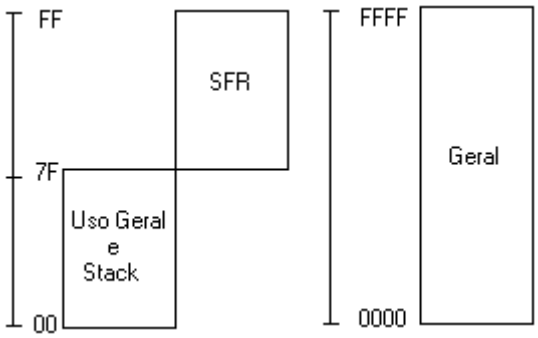
4. Estrutura Interna



5. Organização de memória ROM:



6. Organização de memória RAM:



7. Porque usar a memória ROM ?

Quando criamos um programa e, após compilá-lo, gerando o arquivo **HEX**, este deve ser gravado na CPU para que possa funcionar independentemente da fonte geradora do programa. Logo, este deve, em primeira instância ser gravado em um dispositivo que, mesmo desligado mantenha o programa memorizado, por isso ROM.

Em sistemas mais antigos, costumava-se utilizar memória EPROM, e em meios mais modernos, a própria CPU (**80C51**) contém uma Flash ROM onde o programa residirá. Além disso, como o mesmo é sofisticado, da mesma maneira que um microprocessador, ele pode acessar bancos de memória externos em RAM e/ou ROM.

8. Características da parte baixa de RAM (00 á 7F)

Endereço	Registro	Bit RS1	Bit RS0	Banco
00	R0	0	0	0
01	R1	0	0	0
02	R2	0	0	0
03	R3	0	0	0
04	R4	0	0	0
05	R5	0	0	0
06	R6	0	0	0
07	R7	0	0	0
08	R0	0	1	1
0F	R7	0	1	1
10	R0	1	0	2
17	R7	1	0	2
18	R0	1	1	3
1F	R7	1	1	3

9. PSW - Program Status Word

O **PSW** contém todos os bits que refletem o estado corrente da CPU.

O **PSW** reside no espaço **SFR** Special Flags Registers, e contém o carry bit, o auxiliar carry bit (para operações em BCD), os dois bits de seleção dos bancos de registradores (**R01 e R02**), o Overflow bit, o bit de Paridade e dois bits de estado para uso geral (Todo são conhecidos como flag – bandeira). São eles:

PSW 7	CY	Carry, recebe o bit excedente do Bit 7 da ULA
PSW 6	AC	Auxiliar Carry, recebe o "vai um" do bit 3 em operações lógicas complementares.
PSW 5	X	**Não utilizado**
PSW 4	RS1	Bit 1 da seleção do banco de registradores.
PSW 3	RS0	Bit 0 da seleção do banco de registradores.
PSW 2	OV	Overflow, setado por operações aritméticas.
PSW 1	X	**Não utilizado**
PSW 0	P	Bit Paridade do Acumulador, setado ("1") por hardware caso o número de "1s" no Acumulador seja ímpar.

Carry bit : serve como "vai um" para operações aritméticas, servindo também como "Acumulador" de um número em operações booleanas.

RS0 e RS1: Usados para selecionar o banco de registradores a ser usado, sendo definido pelas instruções que se referem a estas locações como **R0 a R7.** (Vide item 6)

Paridade: Reflete o número de "1s" contidos no Acumulador, ou seja, P=1 se Acc contém um número ímpar de "1s" ou, **P=0** se Acc contém um número par de "1s".

Dois bits no PSW são livres e podem ser usados para uso geral de estado do programa.

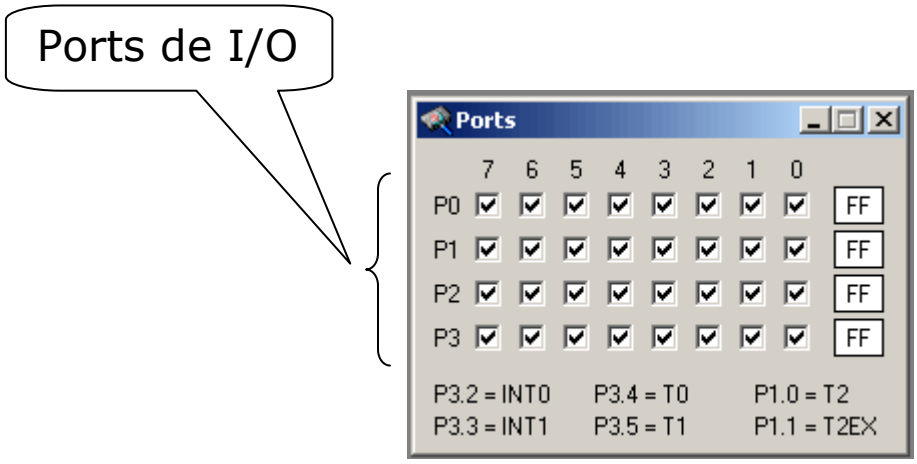
10. Os ports de I/O

Os ports são o meio com o qual o microcontrolador consegue se comunicar com o meio externo a ele.

A família **8051**, da qual atualmente usamos o ATMEL AT89S8253 como produto comercial, trabalha com quatro **(04)** Ports de I/O (entrada/saída) numeradas de **P0 a P3** cada uma com 8 bits de **0 a 7**. Por exemplo, de **P0.0 a P0.7;** de **P1.0 a P1.7;** e assim sucessivamente.

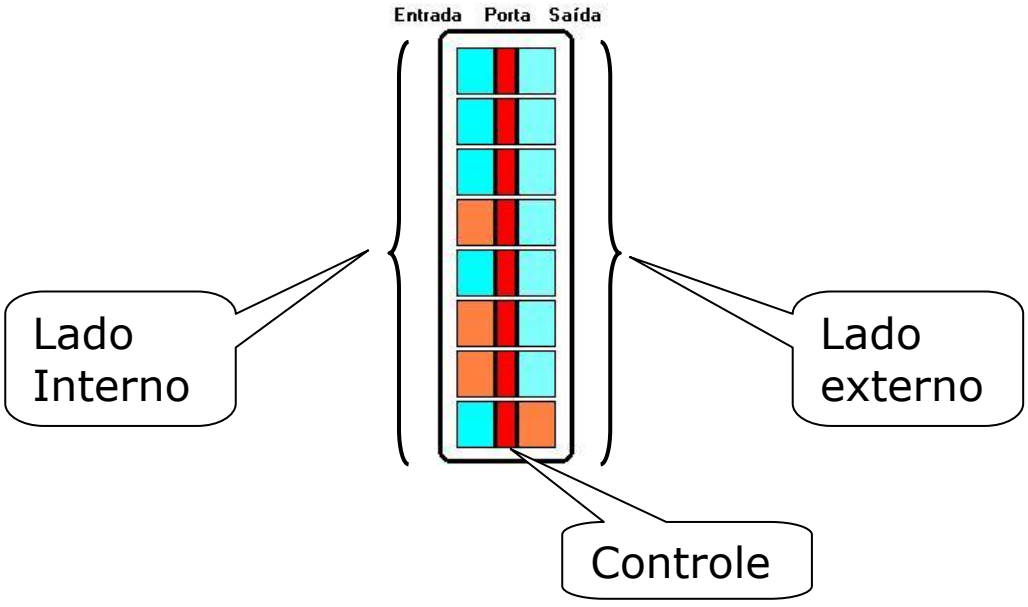
O **8051** pode acessar, individualmente, cada um dos 8 bits de cada um de seus 4 ports de I/O.

Todos os ports são tipo latch.



A palavra **LATCH** indica que existe uma porta separando os dados do exterior dos dados do interior.

Mais ainda, os dados do exterior são "congelados", não sofrendo alterações á menos que seja desejo do programador.



Para ofertar maior flexibilidade de construção de hardware, o **port 0** é mais sofisticado. Ele foi construído com uma tecnologia denominada OD – Open Drain.

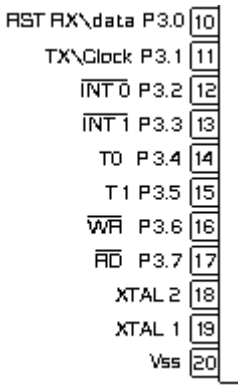
Há uma descrição um pouco mais detalhada de Open-Collector e Tri state em minha apostila de técnicas digitais.

Um resumo de OD seria que isto permite a colocação de níveis DC bem maiores que o usual. Por exemplo, se a alimentação V_{CC} da CPU for de 5 V (padrão TTL) poderíamos conectar (de forma apropriada) um sinal com 15 V_{CC} nos pinos do **port 0**

O **8051** não necessita que se diga que um dado port é entrada ou saída, é necessário sim, o bom senso do projetista de hardware para não usar algum port indiscriminadamente.

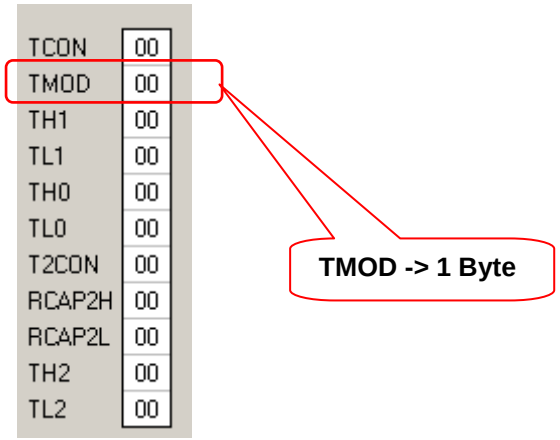
O **port 1** tem pull-up interno (também detalhado em minha apostila de técnicas digitais).

Observe também que o **port P3** tem mais coisas do que as demais: **P3.0 e P3.1** tem também funções utilizadas pela comunicação serial - RX, TX; **P3.2 e P3.3** são utilizadas para entrada de sinais de interrupção externa.



11. Timers no 8051

O **8051** possui 2 Timers independentemente programáveis: **T0** e **T1.** Ambos podem ser programados como Timer/Counter. E essa opção deve ser registrada em **TMOD.**



TMOD	M0	M1		MOD0
	0	0	-------	0
	0	1	-------	1
	1	0	-------	2
	1	1	-------	3

Os dois timers são sensíveis á **borda, nível** ou **ambos.**

Modo 0:

8 Bit Time Counter divide by 32 (13 Bit).

Quando todos os bits são **0** ou todos **1** ⇒ **TF1=1**
São habilitados com **TR=1** e **Gate=0**
TH1 tem **8 bit** e **TL1** tem os **5 bits menos significativos**

Modo 1:

Igual ao modo 0 porém com 16 bit

Modo 2:

Timer/Counter 8 bit TL1
Onde **TL1** não causa **TF=1** mas recarrega **TL1** com o valor de **TH1**

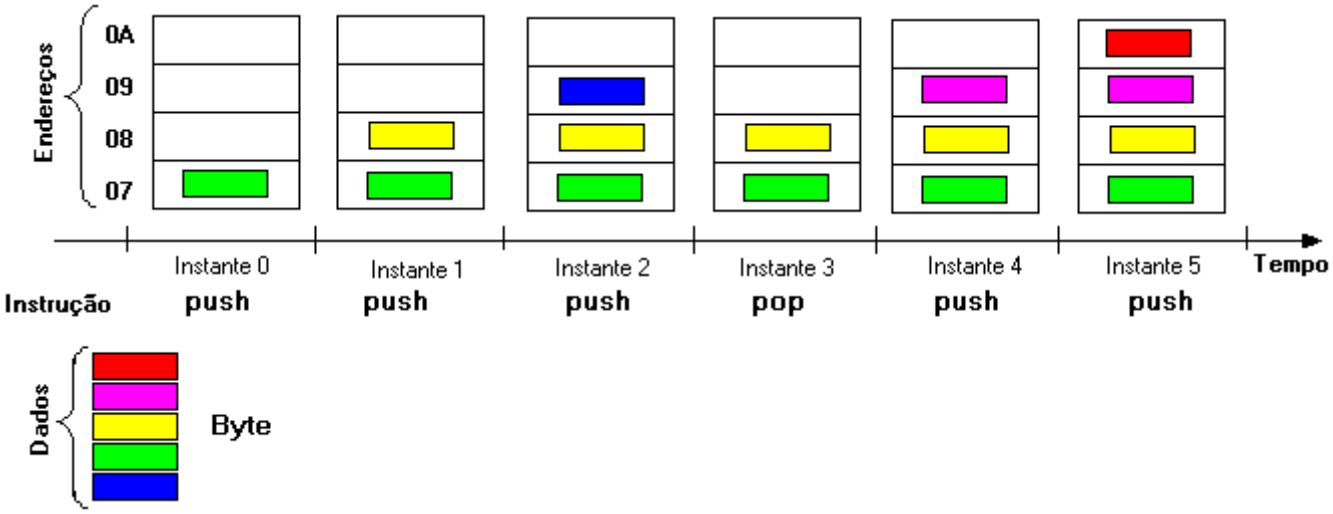
Modo 3:

TL1 e **TH1** são programados individualmente

12. Stack Pointer

Operações com a pilha requerem alguns conhecimentos; primeiro do conceito de pilha **LIFO** – **L**ast **I**n **F**irst **O**ut, depois o uso do registro de localização da pilha, que denominamos de Ponteiro de Pilha ou **Stack Pointer.**

a. Pilha tipo LIFO



b. Uso do Stack Pointer

PUSH – Coloca dado na Pilha endereçada por SP

POP – Extrai dado da Pilha endereçada por SP

c. Outros comandos que usam o Stack Pointer

CALL – Coloca dado na Pilha

RET – Extrai dado da Pilha

Características:

1- Opera com 8 bit em RAM Interna

2- O endereço default (leia **"defô"**) é **07h**

OBS: Quando usar a pilha, comece programando um endereço diferente deste, dê preferencia aos endereços mais altos.

13. Tratamento de Interrupção

Interrupção vetorada

Interrupção vetorada é relativa ao evento, isto é, uma interrupção gera uma instrução CALL á um endereço específico em RAM pré-estabelecido com a seqüência:

Interrupção	Endereço	Prioridade
Int 0	03H	Maior ⇓ Menor
Timer/Counter 0	0BH	
Int 1	13H	
Timer/Counter 1	1BH	
Serial	23H	

- Nesse endereço o programador pode inserir um comando LJMP para o endereço verdadeiro em RAM para o tratamento da interrupção.
- A conclusão da Interrupção sempre deve ser um comando RETI

O Registro IE (Interrupt Enable)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	x	x	ES	ET1	EX1	ET0	EX0

EA : All Interrupt 1 = On 0 = Off
ES : Serial 1 = On 0 = Off
ET1: Timer_1 1 = On 0 = Off
EX1: External Int_1 1 = On 0 = Off
ET0: Timer_0 1 = On 0 = Off
EX0: External Int_0 1 = On 0 = Off

O Registro IP (Interrupt Priority)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
x	x	x	PS	PT1	PX1	PT0	PX0

PS : Serial **0** = Baixa **1** = Alta
PT1: Timer_1 **0** = Baixa **1** = Alta
PX1: External Int_1 **0** = Baixa **1** = Alta
PT0: Timer_0 **0** = Baixa **1** = Alta
PX0: External Int_0 **0** = Baixa **1** = Alta

O Registro TCON (Interrupt por Nível ou Transição)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
x	x	x	x	IE1	IT1	IE0	IT0

IE1: External Int_1 **1** = Borda **0** = Nível
IT1: Timer_1 **1** = Borda **0** = Nível
IE0: External Int_0 **1** = Borda **0** = Nível
IT0: Timer_0 **1** = Borda **0** = Nível

[Pode ser expandida mediante circuito externo]

Em programação é sempre necessário o preparo básico, o qual consiste em preparar os registros de interrupção de acordo com a necessidade do sistema que será projetado.

Quanto ao procedimento dos Timers, existem dois tipos de comportamento:

1- dispara o processo e aguarda o término da temporização antes de retornar – não usa interrupção, apenas acompanha a evolução do timers aguardando sua sinalização de conclusão do processo;

2- dispara o processo e o Timer e retorna ao programa principal. Após isto, continua fazendo as rotinas que estava em execução e aguarda que o Timer sinalize o final com uma solicitação de Interrupção (conforme já vistos, **0Bh** e **1Bh** – item 11 – Interrupção vetorada).

14. Exercício:

Ler o dispositivo anexado ao **Port 2.** Se o **bit 2 = 1** pegue um byte da tabela gravada em **4000h** e que tem 35 elementos e envie-o para o dispositivo do **Port 3** e então passe a apontar para o próximo byte da tabela.

Quando chegar ao último dado, a tabela volta a endereçar o primeiro (Tabela Cíclica)