CEFET-PR / Pato Branco

MICROCONTROLADORES

Pós-graduação

em

Automação Industrial

Prof. Msc. Rogério Malta Branco

Microcontroladores

Parte I – Definições gerais básicas

Parte II – Microcontrolador 8051

Parte III – Aplicações práticas com 8051

Parte IV – Comparativos entre famílias

PARTE I

Definições gerais básicas

Parte I

1. Introdução
2. Revisão de conceitos básicos
3. O sistema microprocessado
4. O microcontrolador
5. Arquiteturas básicas
6. Filosofias R.I.S.C e C.I.S.C.

Referências bibliográficas:
- Mini-curso Saber jan/2001
- Nicolosi
- Vidal
- Paixão

PARTE I – Revisão de conceitos

1. Representação da informação
   (sistemas de numeração)
2. Aritmética binária
3. Lógica combinacional
   (somadores, subtratores, codificadores, decodificadores)
4. Lógica seqüencial
   (flip-flops, registradores)
5. Memórias
   (RAMs, ROMs)

PARTE I – Sist. Microprocessado

1. Definição
2. Partes fundamentais

Leitura sugerida: Mini-curso Saber
Microcontrolador 8051 detalhado
Aplicaç. Práticas 8051 (Vidal)

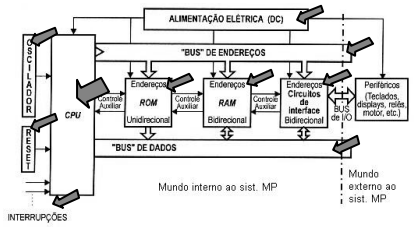
Sist. µP - Definição

Tem como propósito executar uma tarefa específica gravada em sua memória de código ROM, e em geral comunica-se com o mundo real para obter informações do sistema que está inserido e também poder atuar sobre ele.

É uma máquina sequêncial e síncrona, necessitando de sinais de ***clock*** para funcionar.

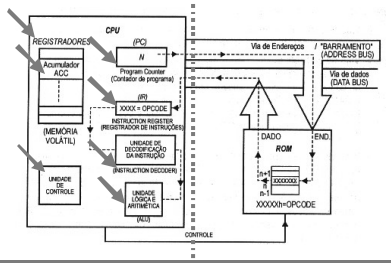
Sist. µP – Partes Fundamentais

É constituído basicamente de: ***cpu, memórias, programa, disp. I/O, clock.***



Sist. µP – Partes Fundamentais

CPU – Controla todas as funções realizadas pelo sistema. Composta de registradores, contadores, ULA ...



Sist. µP – Partes Fundamentais

Registradores: Equivalentes a uma RAM, interna a CPU. Armazenam temporariamente uma informação de utilidade interna ou externa à CPU. Podem ter ou não nomes e serem de função geral ou específica.

Program Counter: Aponta o próximo endereço da próxima instrução a ser executada pela CPU.

Instruction Register:É o local onde fica registrada a instrução trazida da ROM para ser executada.

Decode Unit:É o local onde a instrução locada no IR é traduzida. Atua junto com a Unidade de Controle

Ula:Local de operação de lógica e aritmética.

Accumulator:Registrador principal. Várias operações o envolvem.

Ontrol Unit:.è onde são processados os controles de fluxo das informaçõesa fim de realizar a instrução recebida.

PARTE I – O µC

1. Definição
2. Diferenças entre µP e µC
3. Propósito de aplicação
4. Comparativos com a lógica fixa

O µC - Definição

O microcontrolador é conhecido como ***microcomputador de um só chip***, pois reúne num único componente vários elementos de um sistema, antes baseado em microprocessador e que eram desempenhados por vários componentes independentes tais como RAM, ROM, comunicação serial, etc.

O µC - Diferenças entre µP e µC

O ***hardware*** interno do µC é diferente, ou seja, tem mais funções que o do µP.

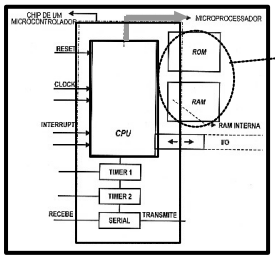
É importante enfatizar que, para a maioria das aplicações, necessita-se, além do ***chip*** do µP, da ROM, do ***latch*** (bus demux), da adaptação serial, etc...

A figura a seguir ressalta algumas diferenças: mostra um µC e indica neste as partes correspondentes a um µP.

O µC - Diferenças entre µP e µC

O µP é o ***chip*** que contém: IR, PC, ALU, INT, etc.

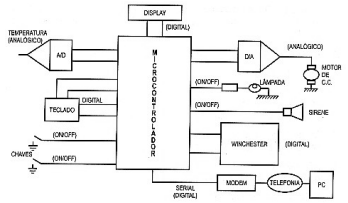
O µC inclui num só chip, o µP, TIMER, SERIAL, e um pedaço da RAM e/ou ROM.



Existem µCs que englobam ainda uma ROM ou EPROM dentro do ***chip***, então não tem-se nada fora. Esta é uma real utilização do µC, ou seja, um chip só com tudo integrado.

O µC – Propósito de aplicação

Controlar um processo industrial, servir de interface homem-máquina, controlar um terminal bancário, controlar uma impressora, um brinquedo, atuar junto a sensores no sistema de injeção de combustível de um motor, etc.



O µC – Comparat. com lógica fixa

Os µP/µC têm preço baixíssimo e está difícil comparar com os custos de circuitos montados com CI´s digitais com lógica fixa, pois estes gastam mais área de placa de PCI, diminuem a confiabilidade (quanto maior, maior a possibilidade de falhas em soldas, por ex.) e não são flexíveis como o é uma máquina programável.

Entretanto, quando o assunto é nanosegundos, os µP/µC esbarram em suas limitações de clock.

Na ordem de microsegundos, os µP/µC são imbatíveis (existem µC com ROM interna por U$0,70).

PARTE I – Arquiteturas Básicas

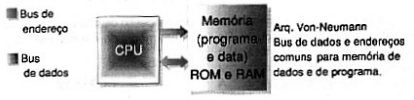
1. Von-Newmann
2. Harvard

Leitura sugerida Saber 360 – Comparativo Técnico Parte I

Von-Newman

Nesta arquitetura, os barramentos de dados e endereços são **compartilhados** entre memórias de programas e dados**.**

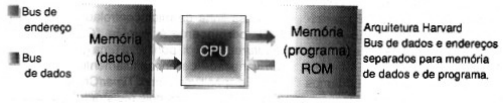
Internamente, só existe um barramento por onde trafegam dados e instruções.



Harvard

Apresenta barramentos de dados e endereços **distintos** , assim enquanto uma instrução é executada, outra é buscada da memória de programa. Este sistema de busca/execução é também conhecido como **PIPELINE.**

Internamente tem-se barramentos distintos para dados e instruções.



PARTE I – C.I.S.C e R.I.S.C

1. CISC
2. RISC

Leitura sugerida: Saber 360 – Comparativo Técnico Parte I;
Desbravando o PIC;
Mini-curso Saber

C.I.S.C.

A filosofia **COMPLEX INSTRUTION SET COMPUTER** baseia-se em quanto maior a complexidade da instrução, mais espaço ela ocupará no ***chip***.

Ter um ***complexo set*** de instruções **C.I.S.C.** nem sempre é interessante, pois pode comprometer o desempenho do processador.

A IBM avaliou que de 200 instruções disponíveis em um dado ***set***, apenas 35 eram comumente usadas. Muitas podiam ser implementadas por meio de outras, surgindo assim a filosofia R.I.S.C.

R.I.S.C.

A **REDUCED INSTRUCTION SET COMPUTER** baseia-se em um ***SET*** reduzido de instruções, muito menos que os microcontroladores da filosofia ***CISC***.

Apresenta a vantagem de ter um aprendizado muito mais din^mico das instruç~es e a desvantagem de ter de construir muitas funções por não existir uma instrução direta, o que implica em maior habilidade do programador.

PARTE II

Definições gerais básicas

Parte II

1. Introdução
2. Arquitetura interna
3. Descrição externa
4. Organização das memórias
5. O Reset
6. Os Ports de I/O
7. Instruções Assembly
8. Interrupção
9. Timers/counters
10. Canal serial

Referências bibliográficas:

- Mini-curso Saber jan/2001
- Nicolosi
- Vidal
- Aplication notes Atmel

Parte II - Introdução

No início da década de 80, a Intel lançou o 8051, fruto de uma evolução da já existente família MCs 48.

Inicialmente existiam: ***8051***, com ROM interna programável de fábrica; ***8751,*** com EPROM interna programável pelo usuário; ***8031,*** com a necessidade de ter chips de EPROM externamente.

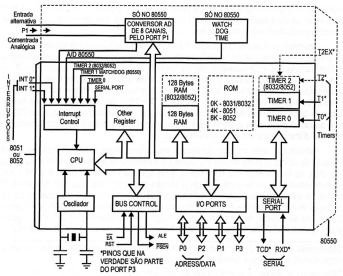
Após, a linha expandiu-se para os modelos: ***8052***, com um Timer a mais que o ***8051***; ***8752***, com EPROM interna; ***8032***, sem ROM/EPROM interna. Surgiu ainda o ***8052 – basic***, com um interpretador basic interno, que permite a programação em basic.

Parte II – Arquitetura Interna

De maneira geral, o 8051 contém internamente:

- RAM interna de 128 bytes Uso Geral + 128 bytes SFRs;
- ROM interna de 4Kbytes;
- 4 portas de I/O;
- 2 Timers/Counters de 16 bits;
- 1 Interface serial;
- Capacidade máx. 64Kbytes de RAM;
- Capacidade máx. 64Kbytes de ROM;
- Ciclos típicos de 1 e 2μs a 12MHz;
- Instrução direta de divisão e multiplicação;
- Entradas de interrupção externa;

Parte II – Arquitetura Interna



Parte II – Arquitetura Interna

Código	ROM (Bytes) dentro do	RAM de uso geral (Bytes) dentro do chip	A/D	Timers	Serial	Utilização dos Ports P3 e P1							
						P1							
						0	1	2	3	4	5	6	7
8051	4K	128	Não tem	2	1	P10	P11	P12	P13	P14	P15	P16	P17
8031	Não tem	128	Não tem	2	1	*	*	*	*	*	*	*	*
8751	4 K - Eprom	128	Não tem	2	1	*	*	*	*	*	*	*	*
8052	8 K	256	Não tem	3	1	T2	T2EX	*	*	*	*	*	*
8032	Não tem	256	Não tem	3	1	T2	T2EX	*	*	*	*	*	*
8752	8 k Eprom	256	Não tem	3	1	T2	T2EX	*	*	*	*	*	*
80550	Não tem	128	8 Canais	2	1	A/D	A/D	A/D	A/D	A/D	A/D	A/D	A/D
83550	4 K	128	8 Canais	2	1	A/D	A/D	A/D	A/D	A/D	A/D	A/D	A/D
85550	4K - Eprom	128	8 Canais	2	1	A/D	A/D	A/D	A/D	A/D	A/D	A/D	A/D

P3							
0	1	2	3	4	5	6	7
RX	TX	INT0	INT1	T0	T1	WR	RD

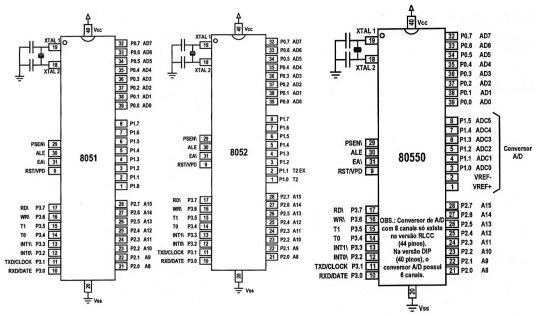
Parte II – Descrição externa

As portas de I/O P0, P1, P2, P3, cada uma com 8 linhas, são destinadas à comunicação externa.

P0 e P2 destinam-se a gerenciar as vias de dados e endereços da comunicação do microcontrolador com a ROM, RAM ou periféricos tipo I/O Mapeado.

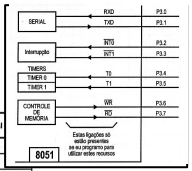
P1 e P3 destinam-se às vias de comunicação externa, sendo tipicamente usadas para interfaces com o mundo externo.

Parte II – Descrição externa



Parte II – Descrição externa

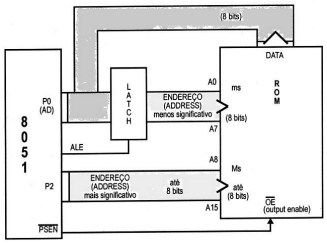
Dentre as Portas de I/O, P3 tipicamente é a mais compartilhada ...



Nome	Número do Pino*	Função Especial**	Função Normal	Função Especial	Comentários da Função Especial
P3.0	10	RXD	I / O	RDX, Receive Data	Usado como receptor de dados serial
P3.1	11	TXD	I / O	TXD, Transmit Data	Usado como transmissor de dados serial
P3.2	12	$\overline{INT0}$	I / O	External interrupt 0	Usado para algum evento externo interromper o Mc
P3.3	13	$\overline{INT1}$	I / O	External interrupt 1	Usado para outro evento externo interromper o Mc
P3.4	14	T0	I / O	Timer/counter 0: External input	Usado quando se quer que o timer zero se torne um contador de eventos externos
P3.5	15	T1	I / O	Timer/counter 1: External input	Usado quando se quer que o timer1 se torne um contador de eventos externos
P3.6	16	$\overline{WR}$	I / O	External data: Memory write strobe	Usado quando se conecta RAM externa no chip. Sinaliza que o Mp vai "escrever" na RAM
P3.7	17	$\overline{RD}$	I / O	External Data: Memory read strobe	Usado quando se conecta a RAM externa no chip. Sinaliza que o Mp vai "ler" da RAM

Parte II – Descrição externa

E P0 e P2 atuam no endereçamento externo do 8051 ...



Parte II – Descrição externa

Dentre os pinos aplicados ao controle, tem-se:

PSEN: Program Store Enable, é um dos quatro pinos de controle do ***chip***. Aciona automaticamente a memória de programa externa nas operações de busca de instrução.

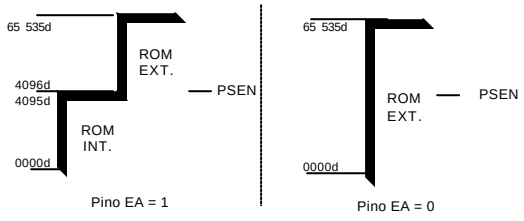
ALE: Address Latch Enable, é o pino que comanda a demultiplexação das informações de dados e endereços menos significativos do Port 0. Acionado automaticamente.

EA: External Access, determin qual memória de programa será acionada: a interna, quando em 1 lógico e a externa, quando em 0 lógico. Quando em 1, o chip irá ler internamente e, após acabar todo o espaço interno, irá automaticamente acessar a mem. Externa, caso ela exista.

Reset: Quando em nível lógico 1 por mais de 2 ciclos de máquina, organiza os valores ***default*** dos registradores, a fim de começas o serviço adequadamente.

Parte II – Organização da memória

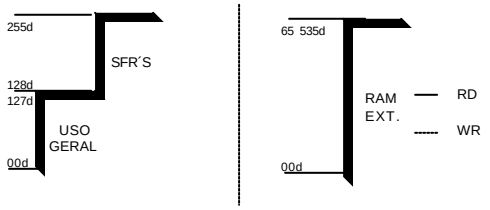
Tipicamente, o ***8051*** separa dados de instruções, e pode gerenciar ROM/EPROM interna ou externa, e ainda ....



ou

Parte II – Organização da memória

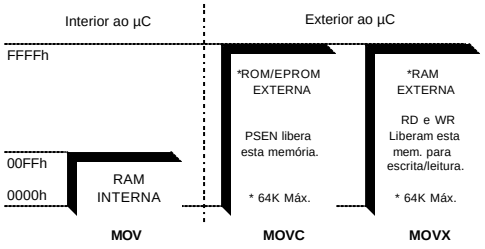
... pode fazer um controle semelhante na RAM interna ou externa.



e

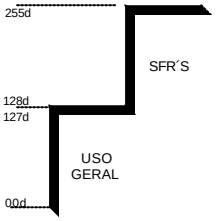
Parte II – Organização da memória

Quanto ao acesso, pode-se utilizar instruções específicas para acessar cada tipo de memória já mencionada.

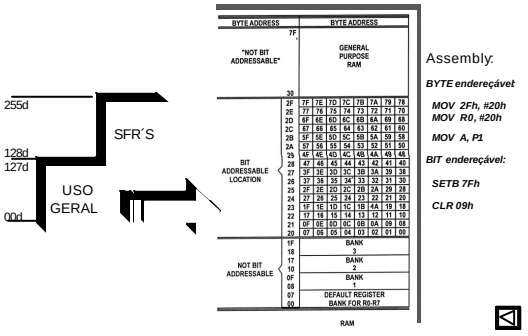


Parte II – Organização da memória

Observando mais detalhadamente a RAM interna, vê-se os registradores que a compõe, sejam de uso geral, sejam de funções especiais.



Parte II – Organização da memória



Assembly:

BYTE endereçável

MOV 2Fh, #20h
MOV R0, #20h

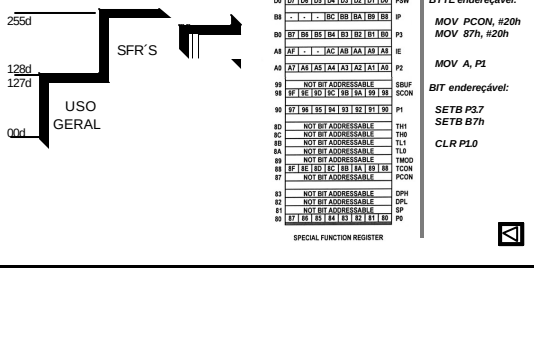
MOV A, P1

BIT endereçável:

SETB 7Fh

CLR 09h

Parte II – Organização da memória



Assembly:

BYTE endereçável:

MOV PCON, #20h
MOV 87h, #20h

MOV A, P1

BIT endereçável:

SETB P3.7
SETB B7h

CLR P1.0

Parte II – Organização da memória

Os Registradores podem ser melhor apresentados:

ACC: Acumulador – Utiliza-se como operando na maioria das instruções do µC.

Ex. de instruções:

add A, Rn: conteúdo de Rn + conteúdo de A e guarda em A;
anl A,#dado: faz AND lógico do ***dado*** com o conteúdo do A;
mov A, Rn: copia o conteúdo de Rn para A

Nome: ACC

End. Bit:

E7	E6	E5	E4	E3	E2	E1	E0
ACC7	ACC6	ACC5	ACC4	ACC3	ACC2	ACC1	ACC0

End. Byte: E0h

Parte II – Organização da memória

B: Registrador Auxiliar B – Em poucas funções existe seu nome explicitamente citado, como abaixo.

Ex. de instruções:

amul AB: multiplica os conteúdos de A e B e guarda em A
div AB: divide o conteúdos de A por B e guarda em A

Nome: B

End. Bit:

F7	F6	F5	F4	F3	F2	F1	F0
-	-	-	-	-	-	-	-

End. Byte: F0h

Parte II – Organização da memória

P0, P1, P2 e P3: Regitradores que espelham a situação atual dos pinos físicos dos ports do microcontrolador.

Ex. de instruções:

MOV P1, #00h
MOV A, P1
SETB P3.7
CLR P1.0

Nome: P0

End. Bit:

87	86	85	84	83	82	81	80
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

End. Byte: 80h (P1=90H, P2=A0h, P3=B0h)

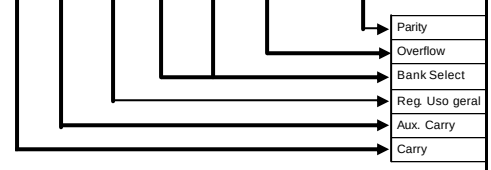
Parte II – Organização da memória

PSW: Program Status Word – Registrador de ***status*** da última operação realizada no Acc.

End. Bit:

D7	D6	D5	D4	D3	D2	D1	D0
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	AC	FO	RS1	RS0	OV	-	P

End. Byte: D0h



Parte II – Organização da memória

Carry: Bit indicador de "vai um" em operações aritméticas – vai para "1" quando ocorre "estouro" da capacidade de A.

Ex: MOV A, #0FFh; ADD A,#01h; ***/* FFh + 01h => A=00h e CY=1 */***

Auxiliary Carry: Bit indicador de "vai um" do nible inferior para o superior, também conhecido como ***HALF CARRY***. Muito usado em aritmética com ***BCD***, onde núm. binários repres. decimais de 0 a 9.

Flag 0: Bit sem função especificada.

RS1 e RS0: Bits de habilitação de banco de registradores ativos. Banco 0 é ***default***.

RS1	RS0	BANCO
0	0	0
0	1	1
1	0	2
1	1	3

Overflow: Bit indicador de resultao > +127 ou < -128 (signed)

Parity: Quando a paridade do Acc for 1, o BIT é setado..

Parte II – Organização da memória

IE: Interrupt Enable – Permite habilitar as interrupções;

IP: Interrupt Priority – Permite controlar a prioridade das interrupções. Serão abordados à posteriori.

DPTR, DPH e DPL: Par de registradores compostos como se fossem uma só palavra de 16 bits. Com 16 bits pode-se acessar dados externos até 64Kbytes. Existem instruções que usam DPTR como ponteiro de memória.

Ex. ***MOV DPTR,#2000h***
MOVX A, @DPTR

Parte II – Organização da memória

PCON: Power Control Register – É um byte miscelânea.

SMOD: Bit usado no cálculo de tempos do Timer, quando envolver canal serial.

GF1 e GF0: Sem função específica. Bits de uso geral.

PD: Power down – Forçando 1, será a última instrução a ser executada pelo mc antes de entrar em "Power down"(Oscilador para, todas as funções param, valores da RAM int. ficam mantidos, Níveis dos ports são matidos, ALE e \PSEN ficam em 0, só com RESET para sair deste estado).

IDL: Idle mode – Forçando 1, será a última instrução a ser executada antes de entrar neste modo (o clock é liberado para a CPU, mas não para as interrupções, timers e serial; estados de CPU e RAM interna são mantidos; níveis dos timers são mantidos; ALE e PSEN ficam em nível 1; finalizado por RESET ou por alguma interrupção já liberada, que zeram autom. O bit IDL)

-	-	-	-	-	-	-	-
SMOD	X	X	X	GF1	GF0	PD	IDL

End. Byte: 87h

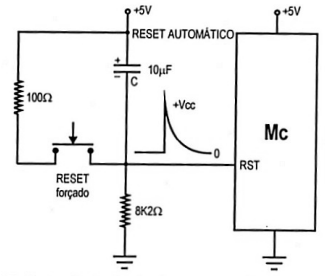
Parte II – O Reset

O ***RESET*** do microcontrolador é um ***pino*** físico do chip, chamado ***RST***.

Deve haver um circuito no referido pino para que, no ligamento do chip, tenha-se estado lógico 1, ao menos por dois ciclos de máquina. A este circuito dá-se o nome de ***Reset automático na energização***. Assim, pode-se garantir que o chip será iniciado sempre da mesma forma.

É importante que exista, um botão para se forçar o ***Reset*** quando necesário.

Parte II – O Reset



Parte II – O Reset

Program counter	PC	0000H
Acumulador	ACC	00H
Registrador B	B	00H
Stack Pointer	SP	07H
Program Status Word	PSW	00H
Data Pointer	DPTR	0000H
P0, P1, P2, P3	P0..P3	FFH
Interrupt Enable	IE	0XX00000b 8051 / 0X000000b 8052
Interupt Priority	IP	XXX00000b 8051 / XX000000b 8052
TH e TL	TH e TL	00H
Timer Control	TCON	00H
Timer Mode	TMOD	00H
Serial Comunication	SCON	00H
Serial Buffer	SBUF	XXH
Power control	PCON	0XXXXXXXb HMOS / 0XXX0000b CMOS
Ram ao ligar	-	INDEFINIDO
Ram após reset forçado	-	NÃO ALTERA

Parte II – Os Ports de I/O

São eles: ***P0, P1, P2, P3.***

Têm capacidade para 8 cargas digitais tipo "LS" para P0 e 4 para os demais ports. Equivalente a duas cargas TTL no P0 e uma nos demais ports.

P0: Bidirecional – quando usado como port é como se fosse ***open drain***, isto é, deve-se utilizar resistores de ***pull-up***, afim de excitarmos devidamente o ambiente externo ao µC. Quando usado no controle da memória externa, estes resistores são desnecessários. Neste caso, o port P0 é visto como um port "***tri-state***".

Parte II – Os Ports de I/O

P1, P2 e P3: Quase-bidirecionais – Estes ports diferem de P0 pois já apresentam resistores de ***pull-up*** internos, logo nunca ficarão realmente em ***Tri-state***.

CUIDADO com as instruções que alteram diretamente o conteúdo dos ports – *Read-Modify-Write*.

Ex: orl P1,#11111110b (será lido P1 pelo latch, operar a instrução OR com o dado e devolver o resultado em P1. Se a intenção era ler o bit msb, empregou-se erroneamente a instrução, pois serão ligados todos os demais bits)

```
Ex: mov A, P1
    orl A,#11111110b
```

Parte II – Os Ports de I/O

Instrução e forma básica	Função exemplo
INC	INC P1
DEC	DEC P3
CPL	CPL P1
JBC	JBC P1.0,#XX
DJNZ	DJNZ P1,#XX
ANL	ANL P0,A
ORL	ORL P0,A
XRL	XRL P1,#XX

Parte II – Instruções Assembly

O µC entende apenas algumas instruções, em contraste ao nosso vocabulário, assim, precisa-se conhecer as suas instruções para que se possa desenvolver tarefas para ele.

Algumas características básicas de instruções podem ser desenvolvidas:

a) Não há instrução para somar, diretamente, dado de uma posição de memória com outra;

b) Não há instrução que some uma posição de mem. externa diretamente com o Acc;

c) Posições externas de mem. são de 16 bits e não cabem direto dentro da instrução; só indiretamente. Refere-se a posição apontada por DPTR. Ex: movx A , @DPTR

Parte II – Instruções Assembly

Modos de endereçamento:

A fim de que qualquer ação possa existir, é importante que o µC tenha acesso aos dados, e para tanto, exitem métodos típicos através dos quais o processador pode fazê-lo.

Modo registrador: Conhecidos genericamente por Rn, os registradores R0, R1, ..., R7 podem ser acessados com um único byte, pois o ***opcode*** contém o indicador do registrador.

```
Ex:   MOV R4,A;              (R4 <= A)
      MOV A, R2;             (A <= R2)
      DEC R2;                (R2 <= R2 - 1)
      XRL A,R1;              (A <=A XOR R1)
      XCH A,R6;              (A<=R6; R6<=A)
      CJNE R5, #33, VOLTA;   (Compara R5 c\ 33d e salta
                              para VOLTA se não igual.
                               Se igual, segue)
```

Parte II – Instruções Assembly

Modo direto: Empregado para acessar os 128 bytes da RAM interna do 8051 ou os endereços de I/O ou ainda, os registradores de status e controle.

```
Ex:   MOV A, 77H;     (A <= cont. end 77H)
      MOV 40H,A;      (end. 40h <= A)
      MOV 40H, 45H;   (end. 40h <= cont. end. 45h)
      ADD A,55H;      (A <= A + cont. end. 45h)
      ANL A,55;       (A <= A AND cont. end. 55d)
      DEC P1; (Decrementa o registrador P1)
```

Modo indireto: O endereço alvo é obtido através dos registradores R0 ou R1, designados nesta condição como Ri.

```
Ex:   Supondo R1 <= 44H
      MOV @R1, #77H;(end. @pontado por R1 (44h) <= 77H)
      ANL A,@R1;    (A <= A AND cont. end. @pontado por R1 )
```

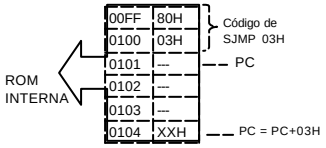
Parte II – Instruções Assembly

Modo Imediato: O código da instrução já é codificado junto com uma constante, ou "dado imediato".

```
Ex:   MOV A,#25H;            (A <= 25H)
      MOV A, #00100101B;     (A <= 00100101B)
      MOV DPTR,#1FFFH ;      (DPTR <= 1FFFH)
      ORL A,#67H;            (A <= A OR 67H)
```

Modo Relativo: Utilizado em instruções de salto, empregando um byte sinalizado para tanto. Pode deslocar-se da posição do PC em -128 a +127 posições.

Ex: SJMP 03H;



Parte II – Instruções Assembly

Modo Absoluto: ACALL e AJMP são instruções que utilizam-se desta modalidade de endereçamento. Permitem um salto de até 2Kbytes de endereços com apenas 2 bytes de instrução.

```
Ex:   0100 H  ACALL 01FFH;   (Faz uma chamada ao end. 01FFh)
      ----    ----           ----
      01FFH   NOP;           (Não faz nada)
      0200H   RET;           (Não faz nada)
```

Modo Longo: LCALL e LJMP são instruções que utilizam-se desta modalidade de endereçamento. Permitem um salto de até 64Kbytes de endereços. Consome 3 bytes de instrução.

```
Ex:   0100 H  LCALL 01FFH;   (Faz uma chamada ao end. 91FFh)
      ----    ----           ----
      9FFFH   NOP;           (Não faz nada)
      9200H   RET;           (Não faz nada)
```

Parte II – Instruções Assembly

Modo Indexado: As instruções que usam este tipo de endereçamento são: JMP e MOVC. Dependem do endereço inserido na instrução aliado ao conteúdo de A.

Ex: JMP @A+DPTR; (Salta p\ end. Apontado por A + DPTR)

DPTR aponta para a "cabeça" da tabela e A permite escursionar por até 256 endereços subsequentes;

```
…
MOV DPTR, #0100H;
MOV A, #01H;
JMP @A+DPTR
…
```

Parte II – Instruções Assembly

Conjunto de Instruções

- Operações Aritméticas
  - ADD, SUBB, INC, DEC, MUL, DIV, DA;
- Operações Lógicas
  - ANL, ORL, XRL, CLR, CPL;
  - RL, RLC, RR, RRC, SWAP;
- Transferências de dados
  - MOV, MOVX, MOVC, PUSH, POP,SCH, XCD;
- Manipulação de variáveis boleanas (bit)
  - CLR, SETB, CPL, ANL, ORL, MOV;
  - JC, JNC, JB, JNB, JBC;
- Controle de Fluxo de execução
  - ACALL, LCALL, RET, RETI, AJMP, SJMP, JMP;
  - JZ, JNZ, CJNE, NOP;

Parte II - Interrupção

A ***interrupção*** é um evento externo ou interno que obriga o µC a suspender temporariamente suas atividades, a fim de atender e este evento que o interrompeu.

Após atendida a interrupção, o µC desvia-se novamente e exatamente para onde estava antes de ter sido interrompido.

Sempre ao término do atendimento de uma interrupção é necessário a presença da instrução ***RETI***.

Parte II - Interrupção

Propriedades da Interrupção.

Vetorada ou Não-vetorada: Quando ocorre a interrupção, o µC deve dirigir-se para o ***endereço da interrupção solicitada.*** Quando este é ***fixo***, diz-se que a ***interupção é vetorada*** – caso do 8051, com endereços pré-definidos para cada interrupção. Caso contrário, é ***não-vetorada,*** e o programador deve definir o endereço de atendimento da interrupção.

Mascaramento: Capacidade de permitir ou não que certo dispositivo o interrompa.

Prioridade: Caso o sistema esteja habiliado a atender mais de uma interrupção, deve-se a sequência de prioridades, onde o µC saberá como agir.

Origem: Pode ser interna ao ***chip*** ou externa.

Tipo de disparo: De propriedade externa ao ***chip***, pode-se programar o ***chip*** para ser interrompido externamente por nível (0 ou 1) ou borda (subida ou descida).

Parte II - Interrupção

Interrupções no 8051:

Não-mascarável: Reset

Mascarável: int0, int1, timer0, timer1, serial

Interrupção	Tipo	End. ROM de desvio
RESET	EXT. – RST	0000h
INT0	EXT. – P3.2	0003h
TIMER/COUNTER 0	Int. – Perif.	000bh
INT1	EXT. – P3.3	0013h
TIMER/COUNTER 1	Int. – Perif.	001bh
SERIAL	Int. – Perif.	0023h
TIMER/COUNTER 2	Int. – Perif.	002bh

Prioridade natural

Parte II - Interrupção

Como programar as interupções:

Interrupt Enable (IE) : Byte de habilitação de interrupções;

EA	-	ET2	ES	ET1	EX1	ET0	EX0

Interrupt Priority (IP): Byte de definição de prioridade de interrupções (2 níveis);

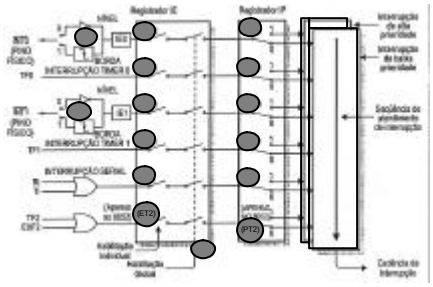
-	-	PT2	PS	PT1	PX1	PT0	PX0

IT0 (TCON.0) e IT1 (TCON.2): Bit de TCON que servem para definir a forma de sinal externo que gerará a interrupção – nível (0) ou borda (1). IE0 está vinculado ao T/C 0 e IE1 está para T/C1.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Parte II - Interrupção

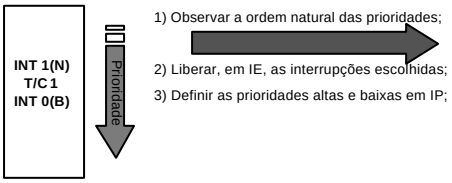
Funcionamento do sistema de interurpção do 8051



Parte II - Interrupção

Exemplo de uso das interrupções:

Programar IE, IP e TCON de modo a:



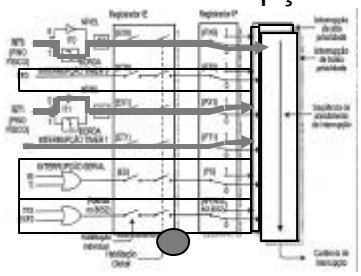
Solução:

1) Observar a ordem natural das prioridades;

2) Liberar, em IE, as interrupções escolhidas;

3) Definir as prioridades altas e baixas em IP;

Parte II - Interrupção



TCON:

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
X	X	X	X	FLAG	0	FLAG	1

IE:

EA	-	ET2	ES	ET1	EX1	ET0	EX0
1	X	0	0	1	1	0	1

IP:

-	-	PT2	PS	PT1	PX1	PT0	PX0
X	X	0	0	1	1	0	0

Parte II – Timers/Counters

Os ***Timers/counters*** são periféricos internos ao microcontrolador. É uma unidade ***autônoma***, que poderia ser um ***chip*** separado. No 8051, tipicamente existe dois destes periféricos.

Nada mais são do que flip-flops configurados em ***divisores por 2*** em cascata. No final dos "n" estágios, existe um flag que indica o "estouro", ou ***overflow*** da contagem.

Quando "lê" pulsos externos, funciona como ***contador***. Quando "lê" os pulsos gerados por ***clock*** interno (dividido por 12), está operando como ***timer***.

Parte II – Timers/Counters

Modos de trabalho do Timer/counter

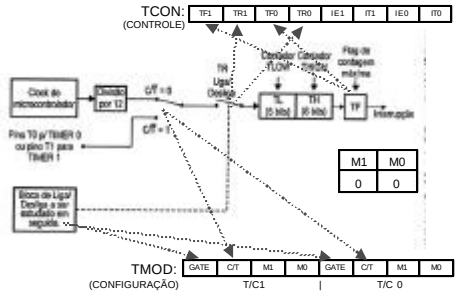
Os dois T/Cs podem ser configurados separadamente em até 4 modos diferentes de funcionamento.

Modo 0: Capacidade máx. 13 bits (8 + 5 pré-scaler)
Modo 1: Capacidade máx. 16 bits (8 + 8)
Modo 2: Capacidade máx. 08 bits c\ autocarga
Modo 3: Misto, a ser definido mais tarde.

São utilizados os SFR´s:
TCON, TMOD, TH0, TL0, TH1 , TF1

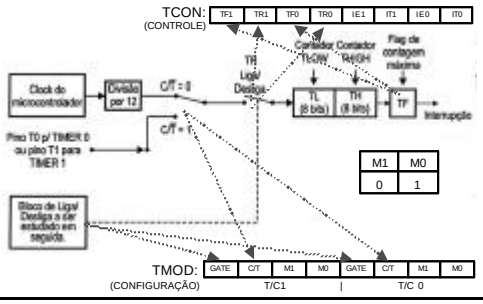
Parte II – Timers/Counters

Modo 0 – 13 bits (8 + 5 – pré-scaler)



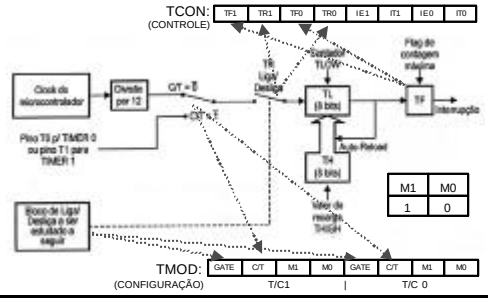
Parte II – Timers/Counters

Modo 1 – 16 bits



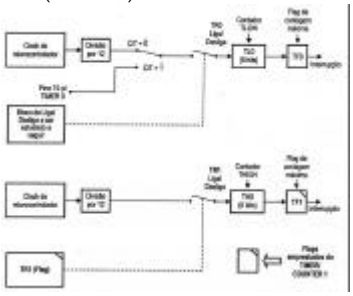
Parte II – Timers/Counters

Modo 2 (8 bits com autocarga)



Parte II – Timers/Counters

Modo 3 (8 bits misto)



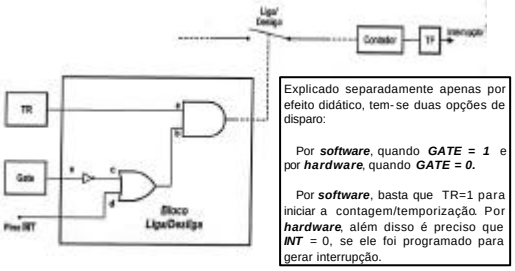
O T/C 0 divide-se em 2 contadores de 8 bits.

T/C 0 em modo 3 => T/C 1 qquer outro modo, mas não gera interrupção.

Os flags TF e TR são emprestados do T/C 1 para este modo.

Parte II – Timers/Counters

O bloco liga-desliga



Explicado separadamente apenas por efeito didático, tem-se duas opções de disparo:

Por ***software***, quando ***GATE = 1*** e por ***hardware***, quando ***GATE = 0.***

Por ***software***, basta que TR=1 para iniciar a contagem/temporização. Por ***hardware***, além disso é preciso que ***INT*** = 0, se ele foi programado para gerar interrupção.

Parte II – Timers/Counters

Observações: tempos curtos / tempos longos

Cada instrução demanda, pelo menos, 1µs de tempo de execução, logo, contar tempos de 2 µs com 12MHz aprox., por exemplo, implicam em não empregar os T/C's.

Se o tempo desejado for superior a 65ms (modo 1 quase ao máx., para 12MHz), já deve-se utilizar também, um contador por ***software***. Ele poderá, por exemplo, decrementar um registrador a cada estouro de 65ms.

Parte II – Canal serial

O canal serial permite que exista comunicação bilateral entre o µC e outro dispositivo.

É denominado ***SERIAL*** por enviar bit a bit, o byte desejado. O receptor irá, também, receber o refrido byte, bit a bit.

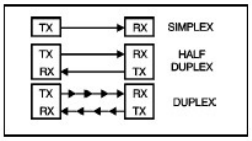
A comunicação serial pode ser ***SÍNCRONA*** ou ***ASSÍNCRONA***. A primeira necessita enviar um sinal de sincronismo para que o receptor pssa saber quando é preciso ler o sinal enviado. É fundamental enviar também, um byte inicial de geração do sincronismo e, no término da transmissão, enviar um outro byte, agora de indicação de fim de transmissão.

A segunda não necessita deste sincronismo, pois cada byte é cercado de um start e um stop bit. O receptor aguarda o start-bit (1 p/ 0), após conta os bits transmitidos e aguarda o stop-bit (1 p/ 0).

Parte II – Canal serial

Na comunicação serial, existem diversos ***protocolos*** – normas padronizadoras de transmissão e recepção de dados, como a RS-232 e RS-485.

São três os sistemas de interligação digital: ***simplex***, ***half duplex*** e ***full duplex.***



Parte II – Canal serial

O 8051 possui um modo de comunicação síncrona (Modo 0) e três modos de comunicação assíncrona (Modos 1, 2, e 3).

O gerador de ***BAUD-RATE*** é vinculado ao ***T/C1***. No 8052, pode-se vincular também ao ***T/C2***.

O periférico serial do 8051 apresenta duas palavras de atuação: ***SCON*** – controle serial; ***SBUF*** – serial buffer.

SCON:

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Parte II – Canal serial

Em termos de ***software***, a transmissão e a recepção serial resumem-se em utilizar o ***SBUF***, como: ***MOV SBUF, A*** ou ainda ***MOV A, SBUF.***

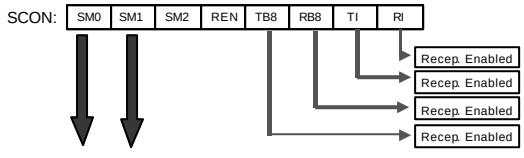
A transmissão inicia-se quando um dado é enviado para ***SBUF***.

A recepção inicia-se, nos ***MODOS 1, 2 e 3***, quando o bit ***REN*** (***SCON.4***) estiver ***SETADO***, e o sistema detecte o ***start-bit.***

No modo 0, além de ***REN=1***, deve-se ter o bit ***RI=0***, pois não há ***START-BIT***.

Parte II – Canal serial

Modos de comunicação serial:



MODO	SM0	SM1	COMUNICAÇÃO	SIZE	TAXA
0	0	0	Síncrona / reg. Desloc.	8 BITS	F OSC/12
1	0	1	Assínc / UART 8BITS	8 BITS	VAR.
2	1	0	Assínc / UART 9BITS	9 BITS	Fosc/32 ou Fosc/64
3	1	1	Assínc / UART 9BITS	9 BITS	VAR.

Parte II – Canal serial

MODO 0 – Síncrono: Basicamente é um ***shift-register*** de 8 bits, em que RXD (P3.0) e TXD (P3.1) são envolvidos.

RXD serve como receptor/transmissor e TXD serve como ***clock*** de referência para RXD.

O ***BAUD-RATE*** é fixo na frequência do clock do 8051 dividida por 12.

Quando escreve-se no ***SBUF***, dá-se início a transmissão.

A recepção é iniciada quando o bit REN=1 e RI=0.

Parte II – Canal serial

MODO 1 – Assíncrono 8 bits: O pino de recepção é o RXD e o de transmissão é o TXD.

Transmite-se/recebe-se 10 bits, sendo: 01 start-bit (nível 0); 08 bits de dados; 01 stop-bit (nível1).

Usa-se o TC1 no modo 2 – autocarga, para gerar a NIbaud-rate ***adequada.***

A ***baud-rate*** é variável: $\frac{k \cdot freq\_clock}{32 \cdot 12 \cdot (256 - TH1)}$ onde: SMOD=0? k=1; SMOD=1? k=2

Na recepção, espera-se o ***start-bit***, que deve ser 0 (1 p/ 0). Aceito, inicia-se a amostragem dos outros 8 bits. Após, espera-se um ***stop-bit*** (0 p/ 1). Não verificadas estas condições, o sistema descarta o suposto bit de start e reinicia a captura de um próximo start. Necessita-se ainda de RI = 0 para receber um dado..

Parte II – Canal serial

MODO 2 – Assíncrono 9 bits: O pino de recepção é o RXD e o de transmissão é o TXD.

Transmite-se/recebe-se 11 bits, sendo: 01 start-bit (nível 0); 08 bits de dados; 01 de paridade; 01 stop-bit (nível1).

A ***baud-rate*** é freq_clock/32 quando SMOD = 0, ou freq_clock/64, quando SMOD = 1. SMOD é bit de PCON.

O nono bit (paridade) é posto em TB8 (SCON.4) quando transmitido e em RB8 (scon.3) quando recebido. O próprio periférico encarrega-se de enviar registrar ou enviar, no MODO 2, tais bits.

Parte II – Canal serial

MODO 3 – Assíncrono 9 bits com Baud-rate variável: O pino de recepção é o RXD e o de transmissão é o TXD.

Transmite-se/recebe-se 11 bits, sendo: 01 start-bit (nível 0); 08 bits de dados; 01 de paridade; 01 stop-bit (nível 1).

A ***baud-rate*** é freq_clock/32 quando SMOD = 0, ou freq_clock/64, quando SMOD = 1. SMOD é bit de PCON.

O nono bit (paridade) é posto em TB8 (SCON.4) quando transmitido e em RB8 (SCON.3) quando recebido. O próprio periférico encarrega-se de enviar registrar ou enviar, no MODO 2, tais bits.

PARTE III

Aplicações práticas com 8051

Parte III

1. Ports de IO
2. Timers/Counters
3. Conversor AD
4. LCD
5. Expansão de IO
6. Serial

Referências bibliográficas:

- Nicolosi
- Vidal
- Paixão