MINI-CURSO DE MICROCONTROLADOR

José Edson dos Santos Marinho
Ednaldo dos Santos Marinho

INTRODUÇÃO

Com o avanço da tecnologia e a utilização da eletrônica digital por grande parte das empresas, o emprego de microcontroladores vêm sendo muito requisitado para um melhor desenvolvimento da produção, diminuindo os custos e trazendo benefícios para as empresas que utilizam esse sistema. É importante salientar que, considerando a relação custo/benefício, os microcontroladores podem não só ser usados em empresas de médio/grande porte, como podem também ser utilizados em vários projetos de eletrônica, na substituição de vários componentes digitais, obtendo-se assim no final do projeto um melhor acabamento – pois um microcontrolador ocuparia um menor espaço físico - e uma maior eficiência e praticidade, uma vez que todos os comandos seriam executados via software.

Antes de um aprofundamento no assunto microcontroladores, é importante conhecermos um pouco da história desses componentes desde as suas origens.

Na década de 70 começaram a ser utilizados microprocessadores em computadores para uma maior eficiência no processamento de dados. O microprocessador Intel foi um dos precursores e, a partir daí, houve uma preocupação em melhorar cada vez mais o sistema de processamento de dados através desses componentes. Baseado na arquitetura de um microprocessador e seus periféricos, foi criado um componente que (fisicamente em uma unidade) comportasse todo um sistema que equivalesse a um microprocessador e seus periféricos; assim surgiu o microcontrolador. Todas as informações e explicações citadas neste trabalho baseiam-se nos microcontroladores 8051 e 8031 da Intel.

UNIDADE CENTRAL DE PROCESSAMENTO (CPU)

A unidade central de processamento controla todas as funções realizadas pelo sistema.

A CPU de qualquer sistema de computador contém os seguintes grupos de unidades funcionais:

- Registradores e contadores

Os registradores e contadores são unidades funcionais usadas para o armazenamento temporário de bits dentro da CPU.

- Unidade Lógica e Aritmética (ULA)

A unidade lógica e aritmética é a unidade funcional da CPU que executa operações lógicas e aritméticas entre palavras binárias, gerando uma outra palavra na saída .

- Unidade de controle e sincronização

A unidade de controle e sincronização coordena e controla todas as unidades funcionais em uma seqüência lógica e sincronizada.

PROCESSAMENTO

O processador ou unidade central de processamento (CPU) é a parte do sistema que faz o processamento das informações para que as instruções sejam executadas; as instruções devem estar armazenadas na memória de programa em seqüência, formando assim o programa .

A CPU possui um registrador chamado de contador de programa (PC) que contém o endereço da próxima instrução que deve ser executada.

Toda vez que uma instrução é retirada da memória pela unidade central de processamento, automaticamente o contador de programa é incrementado para que, após o processamento desta instrução, quando a CPU for buscar a próxima instrução, baste usar o endereço contido no contador de programa.

Toda vez que a CPU é ligada ou *resetada*, automaticamente o seu contador de programa é zerado, desta forma, a primeira tarefa que a CPU irá realizar é a execução da instrução contida na posição de memória de endereço "0000". Cada instrução possui duas fases distintas: o ciclo de busca e o ciclo de execução. Durante o ciclo de uma instrução a CPU faz com que o conteúdo do contador de programa seja colocado no barramento de endereços, endereçando, desta maneira, a posição de memória que contém a instrução que deve ser executada.

UNIDADES DE ENTRADA/SAÍDA (I/O)

As unidades de entrada/saída são os meios pelos quais o usuário se comunica com o sistema. Essas unidades possuem interfaces que permitem a conexão com dispositivos chamados de periféricos, tais como teclado, monitores, LCD´s, etc.

ARQUITETURA

A performance do microcontrolador depende da sua arquitetura interna, ou seja, do modo em que o microcontrolador foi projetado tanto para o hardware como para software. No hardware apresentaremos a arquitetura Von-Neumann, na qual se refere o software CISC.

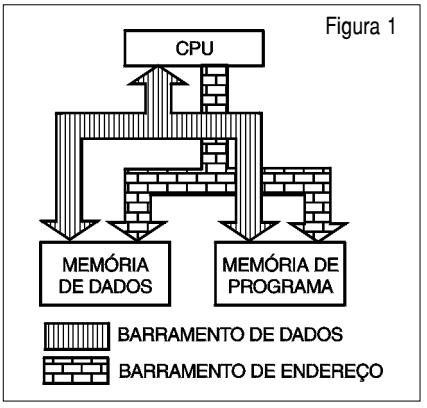
- Arquitetura Von-Neumann

Na arquitetura Von-Neumann, os barramentos de dados e endereços são compartilhados entre memórias de programas e memórias de dados na comunicação com a CPU (figura1). Nesse tipo de arquitetura, quando a CPU está acessando a memória de programa não pode acessar a memória de dados, porque usa os mesmos barramentos para as duas memórias.

CISC (Complex Instruction Set Computer)

CISC: **C**omputador com **S**et de **I**nstrução mais **C**omplexo, quanto maior a complexidade da instrução que deve ser executada, mais espaço ela ocupa no *chip*. Desse modo, chegará um momento que passaremos a ter um *set* de instruções tão grande que começará a afetar o desempenho, dificultando a possibilidade de implementar outras funções

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001



importantes. Ter um complexo (grande) *set* de intruções "CISC" nem sempre é interessante para um bom desempenho do processador.

Numa análise feita pelo laboratório da IBM sobre como estavam sendo usado os diversos tipos de instruções, concluiram que num microprocessador que usava um *set* de instruções de, por exemplo, 200 instruções, a maior parte do processamento era feita apenas com umas 10 instruções.

Uma grande parte das instruções era pouco usada, às vezes até uma única vez em um longo programa, de modo que elas poderiam ser implementadas pelas instruções básicas mais usadas.

Daí o aparecimento da nova arquitetura com o *set* de instruções reduzido "RISC".

MEMÓRIAS

Memórias são os dispositivos que armazenam informações e são usadas em todos os sistemas microcontrolados. Existem vários tipos de memórias que podem ser classificadas por vários ítens diferentes. Vamos analisar os principais:

- Acesso

As memórias armazenam informações em lugares que se denominam "localidades de memória". Cada localidade de memória guarda um conjunto de *bits* e tem um endereço. No acesso desses endereços podemos analisar :

O tempo de acesso: é o tempo que a memória necessita para que sejam escritos ou lidos os dados em suas localidades;

Acesso seqüencial: nas memórias que têm acesso seqüencial, para acessar um endereço de uma certa localidade, precisa-se passar por endereços intermediários (as memórias mais comuns desse tipo são as que utilizam fita magnética);

Acesso aleatório: as memórias que utilizam esse tipo de acesso são as que permitem que seja acessado qualquer dado em qualquer endereço sem a necessidade de ter que passar por outros endereços intermediários.

- Volatilidade

Memórias voláteis: são aquelas que perdem as informações quando é cortada sua alimentação. São memórias que geralmente usam como elemento de memória o *flip-flop*.

Memórias não voláteis: são memórias que mesmo desligando-se sua alimentação, não perdem as informações armazenadas. Dentre essas se destacam as magnéticas e as eletrônicas ROM, PROM, EPROM, EEPROM , e outras.

- Memórias de escrita/leitura ou somente leitura

Escrita/leitura: são memórias que podem ser acessadas pela CPU tanto para leitura quanto para escrita; elas são usadas para armazenar dados que serão utilizados durante a execução do programa (memórias RAM´s, EEPROM´s).

Somente leitura: são as memórias que armazenam o programa, ou seja são as memórias que só serão lidas pela CPU e que já vêm gravadas para o sistema (memórias ROM´s ,PROM´s , etc).

- Tipo de armazenamento

Estáticas: memórias estáticas são aquelas nas quais as informações permanecem armazenadas enquanto não houver escrita ou não faltar energia.

Dinâmicas: memórias dinâmicas são memórias que perdem informações armazenadas mesmo com alimentação. Na RAM dinâmica (ou DRAM) isso acontece porque cada célula tem um transistor MOSFET e um capacitor que armazena um dado (1bit).

- Tipos de memórias

Veremos a seguir alguns tipos de memórias existentes no mercado e que são muito utilizadas:

Memórias RAM *(Random Acces Memory)*

Essas memórias são de acesso aleatório, que podem ser acessadas a qualquer momento e em qualquer endereço. Elas podem ser estáticas ou dinâmicas e também podem ser gravadas pelo sistema com a tensão de 5V. São memórias consideradas voláteis.

Memórias ROM *(Ready Only Memory)*

Essas memórias são utilizadas no sistema somente para a leitura.

Memórias PROM *(Programmable Ready Only Memory)*

Essas memórias são utilizadas no sistema somente para a leitura; geralmente usadas como memórias de programa, só podem ser gravadas com gravadores específicos e só uma vez. São as memórias não voláteis;

Memórias EPROM *(Erasable Programmable Ready Only Memory)*

Essas memórias são utilizadas no sistema somente para a leitura , também empregadas como memórias de programa e só podem ser gravadas com gravadores específicos. Podem ser apagadas por raios ultravioleta e regravadas por muitas vezes. São chamadas memórias não voláteis.

Memórias EEPROM ou E²PROM *(Electrically Erasable Programmable Ready Only Memory)*

Essas memórias podem ser usadas no sistema tanto para leitura como para escrita, podem ser gravadas com gravadores específicos ou pelo sistema; são apagadas eletricamente e regravadas por muitas vezes; são consideradas memórias não voláteis.

Memórias PEROM *(Programmable Erasable Ready Only Memory)*

Uma linha de memórias programáveis e apagáveis apenas para leitura, de 3V e 5V, apenas dentro do sistema. Fabricadas com a avançada tecnologia CMOS, não voláteis, suas características incluem:

· Operação de leitura e programação em apenas 3V e 5V
· Proteção de dados de software e hardware
· Operação de programação por setor
· 1000 ciclos de programa
· Retenção de dados de 10 anos
· Baixa dissipação de potência
· Tempo de ciclo de programa rápido
· Detecção de fim de programa

Memórias FLASH

A memória FLASH é um dispositivo de armazenamento confiável, não volátil, de boa relação custo/ benefício e que possui características de leitura da EPROM, EEPROM e SRAM , porém quando aplica-se 12V sobre o dispositivo, este pode ser gravado com base em *bytes*. No caso da memória FLASH - 5V estes dispositivos foram projetados para serem programados dentro do sistema com o fornecimento padrão de 5V. Em programadores de EPROM convencionais não há necessidade de 12Vpp, nem para programação, nem para apagamento. É composta de uma arquitetura de apagamento de setor (qualquer combinação pode ser apagada simultaneamente) e 100.000 ciclos de apagamento/programação.

Memória FLASHFILE

A memória FLASHFILE, simetricamente bloqueada, da Intel, oferece uma solução não volátil com leitura e programação de mais alta densidade para

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

armazenamento em massa. O armazenamento de aplicações de software e a operação com código de sistema em RFAs *(Residential Flash Arrays)* proporcionam execução instantânea, rápida e no local *(in place).* RFAs são protegidos também contra o envelhecimento do software, já que este pode ser atualizado no sistema. O software RFA prolonga a vida da bateria e aumenta a confiabilidade do sistema através da redução do número de acessos ao *disk-drive.* 100.000 ciclos de apagamento/ programação.

Memórias FIELD

Estes dispositivos são para utilização em filmes digitais e sistemas multimídia. Eles fornecem dados através de acesso serial de alta velocidade. Sua capacidade de memória preenche um arquivo de uma tela de TV NTSC. Cada um dos *bits* possui porta de leitura e gravação assíncronas, de controle independente a diferentes velocidades de clock, proporcionando uma operação FIFO , renovando a células de armazenamento RAM automaticamente.

Memórias FIFO

Os dispositivos FIFO proporcionam armazenamento temporário de dados em sequência de tal forma que a primeira palavra na porta de entrada será a primeira na porta de saída. As portas operam de forma independente e os dados podem ser lidos e gravados em velocidades diferentes.

Os dispositivos FIFO possuem posições de memória que inibem a entrada de dados adicionais caso estejam ocupadas, podendo apenas enviar dados armazenados para fora.

O tempo utilizado para completar uma operação chama-se tempo de acesso, esse valor pode determinar a velocidade do sistema no qual o dispositivo está operando.

Memórias Seriais

Estes dispositivos são de tamanho reduzido podendo ser ligados a um barramento serial I²C *(Inter-Integrated Circuit Bus)* ou SPI *(Serial Peripheral Interface)* junto com outros dispositivos seriais, com muitas vantagens em relação às memórias paralelas.

MICROCONTROLADORES

Existem no mercado muitos tipos de microcontroladores, sendo o 8051 o mais popular. O microcontrolador também é conhecido com microcomputador de um só chip reunindo 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. A memória de programa pode ser ROM, FLASH ou outro tipo. No caso do microcontrolador 8051 ele pode funcionar como um microcontrolador ou como um microprocessador. Na figura 2 mostrado o diagrama de blocos do 8051. A ATMEL possui uma enorme família de componentes com as mesmas características do 805, alguns até com as mesmas pinagens dos registradores; outros com pinagens diferentes, mas com o mesmo *set* de instruções, com *clock* de 12 MHz até aproximadamente 35 MHz.

Falaremos um pouco mais dessa família mais adiante. A DALLAS Semiconductor tem um microcontrolador de alta performance, de até 90MHz, compatível com 8051.

Descrição da pinagem do 8051

Do pino 1ao pino 8 temos:

O port P1, que vai de P1.0 a P1.7. Estes pinos são bidirecionais, podendo ser endereçáveis individualmente ou como porta de 8 bits; possuem resistores de pull-up internos, forçando assim nível lógico alto. Cada pino pode acionar até 4 portas TTL-LS.

Pino 9

RST/VPD. Estes pinos *resetam* o sistema com a aplicação de um nível lógico alto por, pelo menos, dois ciclos de máquina, tendo um resistor de *pull-down* interno permitindo que se use apenas um capacitor externo para obter o *reset* por *power-on*.

Do pino 10 ao pino 17 temos:

O port P3, que vai de P3.0 a P3.7 tem as mesmas características de funcionamento do port P1, tendo também outras funções especiais que estão descritas abaixo.

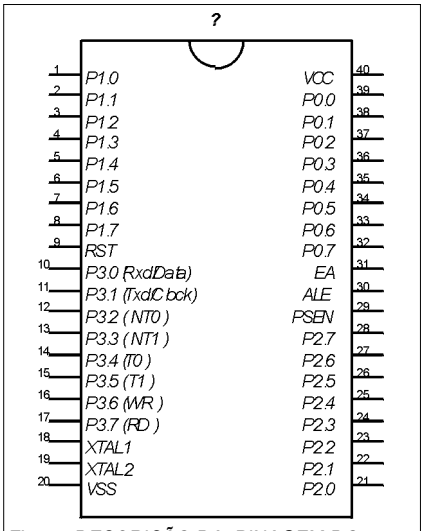


Fig. 3 - DESCRIÇÃO DA PINAGEM DO 8051.

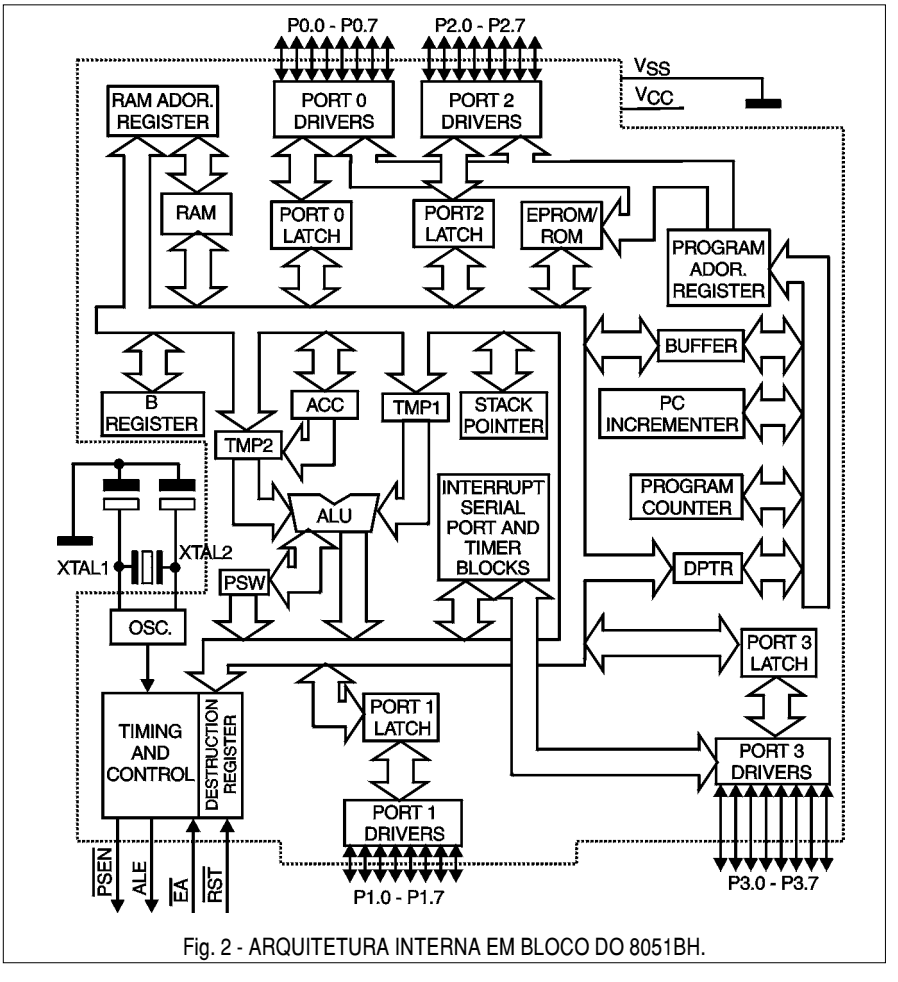


Fig. 2 - ARQUITETURA INTERNA EM BLOCO DO 8051BH.

pino 10 = P3.0 RXD/ Data : entrada de dados serial.

pino 11 = P3.1 TXD/Clock: saída de dados serial.

pino 12 = P3.2 INT0: interrupção externa de número 0, ativo em nível lógico baixo.

pino 13 = P3.3 INT1: interrupção externa de número 1, ativo em nível lógico baixo.

pino 14 = P3.4 T/C0 : entrada externa para o temporizador/contador de eventos.

pino 15 = P3.5 T/C1: entrada externa para o temporizador/ contador de eventos.

pino 16 = P3.6 WR: *strobe* (sinalizador) de escrita de dados externo

pino 17 = P3.7 RD : *strobe* (sinalizador) de leitura de dados externo

Obs: estas funções serão mais detalhadas na continuação da apostila

Os pinos 18 e 19 são:
XTAL 1 (18)
XTAL 2 (19)

Pino 20:
VSS conectado ao terra do circuito

Do pino 21 ao pino 28 temos:
O port P2, que vai de P2.0 a P2.7. Tem as mesmas características de funcionamento do port P1, sendo estes pinos usados como pinos de endereçamento externo endereçando a parte mais significativa dos 16 bits (A8 a A15).

Pino 29:
PSEN *(Program Store Enable)*, saída para habilitação do programa externo, sinalizador de leitura da memória de programa externa; quando o microcontrolador busca instruções este pino vai ao nível zero.

Pino 30:
ALE *(Address Latch Enable)*, saída habilitadora do *latch* de endereços, separando o barramento de dados da parte menos significativa do barramento de endereços que são multiplexados pelo port P0.

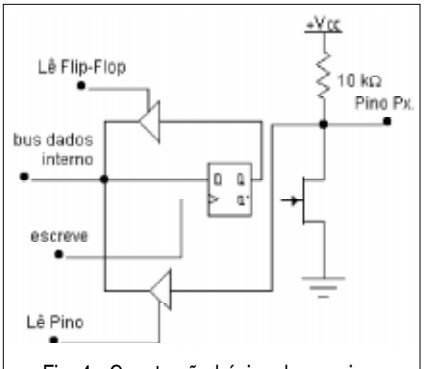


Fig. 4 - Construção básica de um pino.

Pino 31:
EA *(External Enable)* , entrada de seleção de memórias. Quando colocado em nível lógico baixo, a CPU executa somente as instruções da memória de programa externa; quando em nível lógico alto, a CPU executa as instruções da memória de programa interna (se existir).

Pino 32 ao pino 39 temos:
Port P0: que vai de P0.0 a P0.7. Este é um *port* de 8 bits bidirecional com dreno aberto.

Sem resistores de *pull-up* internos, funciona como um barramento de dados, e a parte menos significativa do barramento de endereços.

Pino 40, temos
VCC *(Alimentação positiva)*

Basicamente, os pinos do microcontrolador são construídos como mostra a figura 4 abaixo com algumas diferenças uns dos outros, mas para explicação de um modo geral pode ser considerado um *flip-flop tipo D* com dois *buffers tri-state*, um transistor e um resistor.

Quando o microcontrolador vai escrever no pino, o dado é enviado para a via de dados interna e também é dado um *clock* no *flip-flop*, que terá o mesmo sinal na saída "Q" (e o complemento deste sinal na saída Q/), que por sua vez está ligada ao *gate* de um transistor, o qual entra em saturação quando for aplicado nível "1".

Se o sinal que estiver na via de dados for "1" e o *flip-flop* receber um *clock*, a saída Q terá um sinal de nível alto, e Q/ terá um sinal de nível baixo que entrará no *gate* do transistor deixando-o em corte e, então, o sinal no pino será o do resistor, ou seja , nível "1". Agora, se o sinal que estiver na via de dados interna for nível baixo e novamente o *flip-flop* receber um *clock*, na saída Q teremos um sinal de nível baixo, e na saída Q/ um sinal de nível alto que levará o transistor à saturação e com isso o pino terá um nível baixo. Quando se quer ler um pino, o microcontrolador deve enviar um sinal para o *buffer* "lê pino", que pega o sinal que estiver nele e envia para via de dados interna; quando se quer ler o *flip-flop*, o sinal é enviado para o buffer "lê latch" que libera o sinal da saída Q para a via de dados interna. No microcontrolador existem algumas instruções responsáveis pela leitura do *flip-flop*, e outras do pino. Para que se possa ler um sinal externo no pino, deve-se primeiro escrever nele um nível alto evitando assim que o transistor conduza e mande um sinal baixo no pino, pois se for entrar um sinal de nível alto no pino, o transistor não poderá estar saturado, senão entrará em conflito prejudicando a leitura. Isso acontece em todos os PORT's do microcontrolador. Quando estão sendo usado os pinos de I/O como PORT's (empregando somente as memórias internas, é necessário o uso de resistores de *pull-up* no port P0 .

CLOCK E CICLO DE MÁQUINA

O microcontrolador é um circuito dinâmico, e para o seu funcionamento necessita de um sinal de *clock* (relógio) para sincronização de suas operações. Este sinal de clock é gerado por um oscilador que fornece um seqüência ininterrupta de pulsos com períodos constantes. O chamado ciclo de máquina é uma quantidade de pulsos de clock que o processador requer para fazer suas funções; e o ciclo de instrução é a quantidade de ciclos de máquina que são necessários para a execução de uma instrução.

Um exemplo genérico de uma instrução: suponhamos que um processador qualquer leve para a execução de uma instrução 5 (cinco) ciclos de máquina, e que cada ciclo de máquina desse processador seja igual a 2 (dois) clocks. Para executar essa instrução o processador levará 10 (dez) clocks; supondo também que desses 5 (cinco) ciclos de máquina, 2(dois) seriam para a busca da instrução na memória de programa e os outros 3 (três) para a execu-

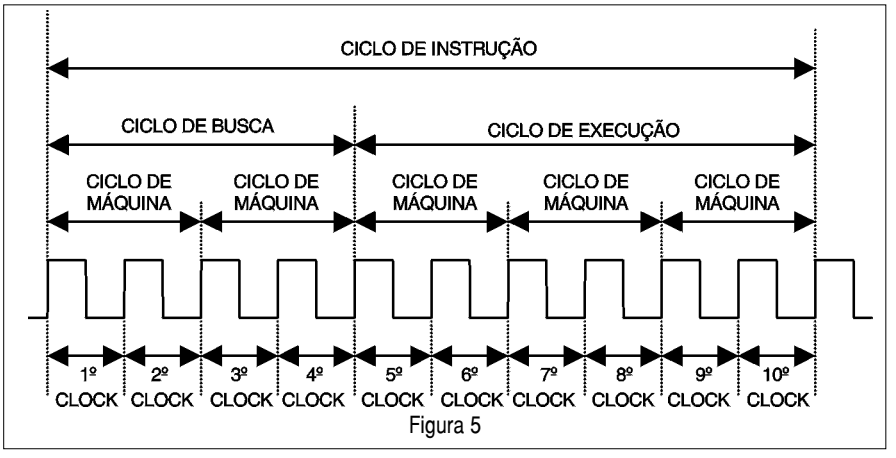


Figura 5

ção dessa instrução, como mostra o gráfico abaixo (figura 5).

No microcontrolador 8051 um ciclo de máquina corresponde a 12 períodos, e as instruções levam de 1 (um) a 3 (três) ciclos de máquina para ser realizadas dependendo da instrução. Conforme o componente, o ciclo de máquina pode ser em maior ou menor número de períodos, outros microcontroladores têm os ciclos de máquina diferentes.

O 8051 possui internamente um circuito oscilador com um inversor linear de estágio simples, oferecendo duas possibilidades de clock distintas, sendo um externa e outra interna.

Para o uso de oscilador externo deve-se aterrar o pino 19 e injetar o sinal externo no pino 18 que, desta forma, não atuará no gerador interno.

Para usarmos o clock gerado internamente devemos intercalar entre os pinos 18 e 19 um cristal com filtro cerâmico na freqüência desejada, como mostra na figura 6.

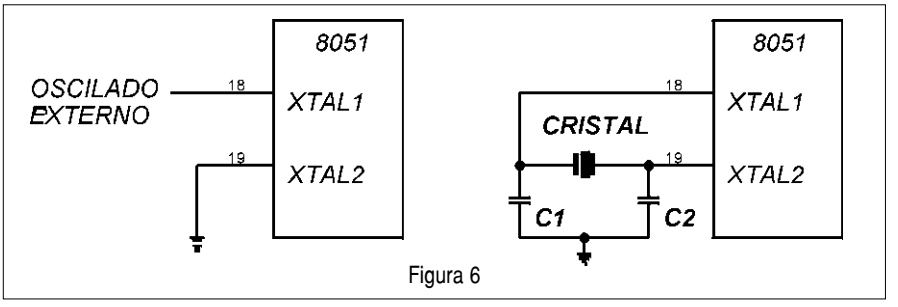


Figura 6

RESET

O reset não é bem uma interrupção, mas às vezes o chamamos assim por sua ação semelhante a uma interrupção, já que ele interrompe o processo e reinicia o sistema. No 8051 o reset é ativo quando o pino 9 permanecer em nível alto por pelo menos 2 ciclos de máquina. Quando os pinos do 8051 estiverem todos em tri-state, e enquanto estiver no processo de reset, os pinos permanecerão em alta impedância.

Quando acontece uma interrupção por reset, a CPU se reorganiza e inicializa com os seguintes valores nos registradores.

Valores dos Registradores após o RESET

Registrador	Valor
PC	0000
ACC	00
PSW	00
SP	07
DPTR	0000
P0-P1-P2-P3	FF
IP	Xxx00000
IE	0xx00000
TMDO	00
TCOM	00
TH0	00
TL0	00
TH1	00
TL1	00
SCOM	00
SBUF	Indeterminado
PCOM	0xxxxxxx*
B	00

TABELA 1

O 8051 pode ser automaticamente resetado toda vez que for ligado (POWER-ON), colocando um resistor e um capacitor no pino 9 (RESET) como mostra a figura 7 abaixo. O auto-reset é usado para que se tenha um tempo para a fonte se estabilizar e, assim, todos os componentes do sistema serão alimentados corretamente evitando níveis de tensões errados.

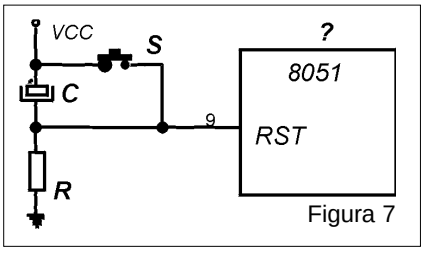


Figura 7

Outro modo de se obter o reset é através de um “push-button” ligado ao pino 9, em paralelo com o capacitor (‘reset forçado’).

Desse modo o reset pode ser dado a qualquer instante sem a necessidade de desligar o sistema.

ORGANIZAÇÃO DA MEMÓRIA

No microcontrolador 8051 a memória está organizada do seguinte modo: memórias internas de 256 bytes de memória RAM, sendo 128 bytes de registradores de função especial e 128 de registradores comuns; e até 4 kbytes de memória de programa interna.

Em relação à memória externa, o microcontrolador pode endereçar até 64 kbytes de memória de programa externa, ou 4kbytes internos e 60 kbytes externos

RESISTRADORES DE FUNÇÕES ESPECIAIS (SFR)

Na área de memória SFR (Registradores de Funções Especiais) existem alguns registradores que são bytes e bits endereçáveis, e outros que só são bytes endereçáveis. Veja figura 8.

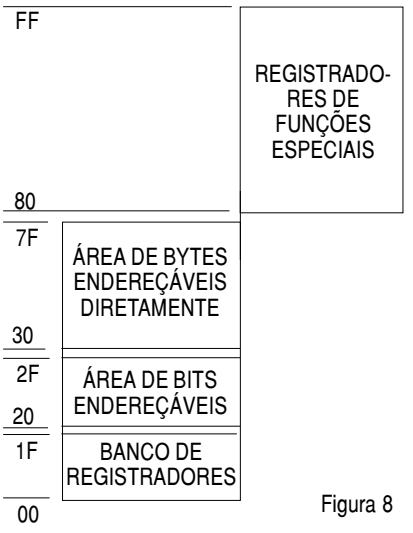


Figura 8

Os registradores que são bytes e bits endereçáveis podem ser acessados bit a bit individualmente, ou pode-se acessar os 8 bits (byte) de uma vez .

Os registradores de funções especiais de bytes e bits endereçáveis e seus endereços, assim como os endereços dos bytes, são mostrados na tabela 2.

OBS.: estes registradores fazem parte da RAM interna do 8051.

Na tabela 2 temos a parte da RAM dos registradores de funções especiais com nome e endereço individual dos bits.

Alguns bits têm nomes e endereços, outros só nomes e outros só endereços; dependendo do microcontrolador usado pode-se ter mais registradores com outras funções .

Descrevemos agora as funções de cada um dos registradores :

O acumulador (E0)

BIT7	**ACC7**	**E7**
BIT6	**ACC6**	**E6**
BIT5	**ACC5**	**E5**
BIT4	**ACC4**	**E4**
BIT3	**ACC3**	**E3**
BIT2	**ACC2**	**E2**
BIT1	**ACC1**	**E1**
BIT0	**ACC0**	**E0**

O acumulador é um registrador de uso geral e é um operador em várias instruções, sendo também o lugar onde fica o resultado de varias operações. Ele é muitas vezes parte do operando da instrução.

Obs.: notem que o endereço do byte do acumulador tem o mesmo no nome do endereço do bit zero, então, como acessar o endereço do bit sem alterar o byte? É simples: no momento de acessar qualquer endereço, a instrução de bit é diferente da instrução de byte, logo o código em hexadecimal também

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

TABELA 2

Endereço Bytes	ENDEREÇOS INDIVIDUAIS DOS BITS								Registrador
E0	**E7**	**E6**	**E5**	**E4**	**E3**	**E2**	**E1**	**E0**	ACC
F0	F7	F6	F5	F4	F3	F2	F1	F0	**B**
80	P0.7 87	**P0.6** 86	**P0.5** 85	**P0.4** 84	**P0.3** 83	**P0.2** 82	**P0.1** 81	**P0.0** 80	**P0**
90	P1.7 **97**	**P1.6** **96**	**P1.5** **95**	**P1.4** **94**	**P1.3** **93**	**P1.2** **92**	**P1.1** **91**	**P1.0** **90**	**P1**
A0	P2.7 A7	**P2.6** A6	**P2.5** A5	**P2.4** A4	**P2.3** A3	**P2.2** A2	**P2.1** A1	**P2.0** A0	**P2**
B0	P3.7 B7	**P3.6** B6	**P3.5** B5	**P3.4** B4	**P3.3** B3	**P3.2** B2	**P3.1** B1	**P3.0** B0	**P3**
A8	EA AF			**ES** AC	**ET1** AB	**EX1** AA	**ET0** A9	**EX0** A8	**IE**
B8				PS BC	**PT1** BB	**PX1** BA	**PT0** B9	**PX0** B8	**IP**
D0	CY D7	**AC** D6	**F0** D5	**RS1** D4	**RS0** D3	**OV** D2	 D1	**P** D0	**PSW**
98	SM1 **9F**	**SM2** **9E**	**SM3** **9D**	**REN** **9C**	**TB8** **9B**	**RB8** **9A**	**TI** **99**	**RI** **98**	**SCN**
88	TF1 8F	**TR1** 8E	**TF0** 8D	**TR0** 8C	**IE1** 8B	**IT1** 8A	**IE0** 89	**IT0** 88	**TCOM**
89	Gate *	**C/T** *	**M1.1** *	M0.1 *	Gate *	C/T *	M1.0 *	M0.0 *	TMOD *

Obs: ' * ' = endereço não disponível em bits, só em bytes;

é diferente e isso serve para todos registradores.

O registrador B (F0)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
F7	F6	F5	F4	F3	F2	F1	F0

Esse registrador em algumas instruções tem seu nome referenciado: MUL AB , DIV AB , que são operações entre os registradores A e B, fora isso ele é um registrador como outro qualquer.

P0 , P1 , P2 e P3

São os registradores que espelham a situação atual dos pinos físicos dos ports.

Registrador P0 (80)

BIT7	87	P0.7
BIT6	86	P0.6
BIT5	85	P0.5
BIT4	84	P0.4
BIT3	83	P0.3
BIT2	82	P0.2
BIT1	81	P0.1
BIT0	80	P0.0

Registrador P1 (90)

BIT7	97	P1.7
BIT6	96	P1.6
BIT5	95	P1.5
BIT4	94	P1.4
BIT3	93	P1.3
BIT2	92	P1.2
BIT1	91	P1.1
BIT0	90	P1.0

Registrador P2 (A0)

BIT7	A7	P2.7
BIT6	A6	P2.6
BIT5	A5	P2.5
BIT4	A4	P2.4
BIT3	A3	P2.3
BIT2	A2	P2.2
BIT1	A1	P2.1
BIT0	A0	P2.0

Registrador P3 (B0)

BIT7	B7	P3.7
BIT6	B6	P3.6
BIT5	B5	P3.5
BIT4	B4	P3.4
BIT3	B3	P3.3
BIT2	B2	P3.2
BIT1	B1	P3.1
BIT0	B0	P3.0

Registrador IE Interrupt Enable(A8)

Este registrador é responsável pela a habilitação das interrupções

BIT7	AF	EA
BIT6	AE	X
BIT5	AD	X
BIT4	AC	ES
BIT3	AB	ET1
BIT2	AA	EX1
BIT1	A9	ET0
BIT0	A8	EX0

EA Enable All: (habilita todas as interrupções). Quando em "0", desabilita todas as interrupções ; quando em "1", permite a escolha de habilitar qualquer interrupção através dos bits de controle individual a seguir

ES Enable Serial (habilita serial). Quando em nível lógico 1 habilita a interrupção pedida pelo canal serial , quando em nível lógico 0 inibe a interrupção pedida pelo canal serial.

ET1 Enable Timer 1 (habilita timer 1). Quando em nível lógico 1 habilita a interrupção pelo timer/count 1. Se EA estiver habilitado, quando em nível lógico 0 inibe a interrupção pedida pelo timer/ count 1 .

EX1 Enable external 1 (habilita INT1). Quando em nível lógico 1, habilita a interrupção pedida pelo dispositivo externo ligado no pino INT1 se AE estiver=1; quando em nível lógico 0 inibe a interrupção pedida pelo dispositivo externo ligado ao pino INT1.

ET0 Enable Timer 0 (habilita timer 0). Quando em nível lógico 1 habilita a interrupção pelo timer/count 0. Se EA estiver habilitado, quando em nível lógico 0 inibe a interrupção pedida pelo timer/count 0.

EX0 Enable external 0 (habilita INT0). Quando em nível lógico 1, habilita a interrupção pedida pelo dispositivo externo ligado no pino INT0 se AE estiver = 1; quando em nível lógico 0 inibe a interrupção pedida pelo dispositivo externo ligado ao pino INT0.

Registrador IP Interrupt Priority (B8)

Através dos bits desse registrador pode-se alterar a prioridade de interrupção.

BIT7	BF	X
BIT6	BE	X
BIT5	BD	X
BIT4	BC	PS
BIT3	BB	PT1
BIT2	BA	PX1
BIT1	B9	PT0
BIT0	B8	PX0

PS Priority Serial (Prioridade Serial). Quando em nível 1 indica prioridade para interrupção gerada pela serial, se a mesma estiver habilitada . Quando em nível 0 indica baixa prioridade.

PT1 Priority Timer 1 (Prioridade do temporizador/contador 1). Se em nível 1, indica prioridade para interrupção pedida pelo temporizador / contador,

PX1 Priority External 1(Prioridade Interrupção Externa). Quando em nível 1, indica que a prioridade é para o dispositivo que esteja ligado no pino 13 (INT1/P3.3) do 8051; no caso de nível 0 indica baixa prioridade.

PT0 Priority Timer 0 (Prioridade do temporizador/contador 0). Se em nível 1, indica prioridade para interrupção pedida pelo temporizador/contador.

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

PX0 Priority External 0 (Prioridade de Interrupção Externa 0) quando em nível 1, indica que a prioridade é para o dispositivo que esteja ligado no pino 12 (INT1/P3.2) do 8051; no caso de nível 0 indica baixa prioridade.

Registrador PSW Program Status Word (D0)

Este registrador contém *flags* (bits) que indicam as ocorrências da ULA a cada operação lógica e aritmética que houver, e também avisa em que banco de registradores está ativo.

BIT7	D7	CY
BIT6	D6	AC
BIT5	D5	F0
BIT4	D4	RS1
BIT3	D3	RS0
BIT2	D2	0V
BIT1	D1	X
BIT0	D0	P

CY *Carry Flag*
AC *Carry Auxiliar*
OV *Overflow Flag*

Estas flags (CY, AC, OV) são modificadas por algumas instruções, como mostra a tabela abaixo

Instruções	**CY**	**AC**	**VO**
ADD	x	x	x
ADDC	x	x	x
ANL C,bit	x	-	-
ANL C,\bit	x	-	-
CLR C	0	-	-
CPL C	x	-	-
CJNE	x	-	-
DA	x	-	-
DIV	0	-	x
MUL	0	-	x
MOV C, bit	x	-	-
ORL C, bit	x	-	-
ORL C, \bit	x	-	-
RLC	x	-	-
RRC	x	-	-
SETB C	1	-	-
SUBB	x	x	x

RS1 Register bank Select 1
RS0 Register bank Select 0

Estas duas flags selecionam o banco de registradores do começo da RAM interna

RS1	**RS0**	**Banco**
0	0	0
0	1	1
1	0	2
1	1	3

F0 Flag 0

É um bit de uso geral. Quando sobra no registrador não tem função

P Parity Flag

Bit que vai para o nível lógico 1 quando o acumulador tiver um número par de "1".

Registrador SCON Serial Control (98)

Este registrador serve para o controle do canal serial.

BIT7	9F	SM0
BIT6	9E	SM1
BIT5	9D	SM2
BIT4	9C	REN
BIT3	9B	TB8
BIT2	9A	BB8
BIT1	99	TI
BIT0	98	RI

SM0 e SM1: Estes selecionam o modo em que o 8051 irá operar com o canal serial, como segue abaixo

SM0	SM1	Modo	Taxa de transmissão
0	0	0	Freq. Clock /12
0	1	1	Variável (Timer 1)
1	0	2	Freq. Clock /32 ou 64
1	1	3	Variável (Timer 1)

SM2

Possui várias finalidades, dependendo de que modo foi selecionado:

Modo 0: não tem qualquer efeito no funcionamento do canal serial, devendo ficar em zero.

Modo 1: não gera interrupção se estiver setado e o stop bit recebido for ilegal.

Modo 2 e 3: habilita a comunicação entre vários microcontroladores, nestes modos não é gerada interrupção se estiver setado, e se o nono bit de dados enviado for zero.

REN Reception Enable (habilita recepção)

Se estiver setado habilita a recepção tão logo um start bit seja detectado, se estiver resetado desabilita a recepção.

TB8

Nos modos 2 e 3 indica o estado do nono bit a ser transmitido, pode ser setado ou resetado por software.

RB8

Não é usado no modo 0 , e no modo 1 indica o estado do stop bit recebido, desde que SM2 esteja zerado. Nos modos 2 e 3 indica o estado do nono bit que foi recebido.

TI

É uma flag de requerimento de interrupção de transmissão, é setada pelo hardware após a transmissão do oitavo bit de dados quando no modo 0, e nos outros modos ao início do stop bit e deve ser zerado por software da rotina de atendimento para permitir novas interrupções.

RI

É uma flag de requerimento de interrupção na recepção, é setada pelo hardware após a recepção do oitavo bit de dados quando no modo 0, e nos outros modos a meio tempo de recepção do stop bit e deve ser zerado por software da rotina de atendimento para permitir novas interrupções.

Registrador TCON time control (88)

BIT7	8F	TF1
BIT6	8E	TR1
BIT5	8D	TF0
BIT4	8C	TR0
BIT3	8B	IE1
BIT2	8A	IT1
BIT1	89	IE0
BIT0	88	IT0

TF1

Sempre que ocorrer um estouro de contagem no contador 1, este bit será levado a nível alto gerando o pedido de interrupção do T/C 1 e será levado ao nível baixo automaticamente depois de ser atendida a interrupção.

TR1

Será levado ao nível alto para ligar o contador 1(iniciando a contagem) e levado ao nível baixo quando quiser desligar o contador 1(parar a contagem) por software.

TF0

Sempre que ocorrer um estouro de contagem no contador 0, este bit será levado ao nível alto gerando o pedido de interrupção do T/C 0 e será levado ao nível baixo automaticamente depois de ser atendida a interrupção

TR0

Será levado ao nível alto para ligar o contador 0(iniciando a contagem) e levado ao nível baixo quando quiser desligar o contador 0 (parar a contagem) por software.

IE1

Esse bit será levado ao nível alto quando for detectado um sinal de nível baixo no pino INT1, sinalizando o pedido de interrupção, e será levado ao nível baixo logo depois que a interrupção for atendida via hardware.

IT1

Indica qual o processo para que haja a chamada de interrupção INT1. Se em nível alto a interrupção será aceita na transição negativa neste pino permanecendo pelo menos 12 períodos de clock, se em nível baixo a interrupção será aceita apenas por nível baixo no pino.

IE0

Esse bit será levado ao nível alto quando for detectado um sinal de nível baixo no pino INT0 sinalizando o pedido de interrupção, e será levado ao nível baixo logo depois que a interrupção for atendida via hardware.

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

IT0

Indica qual o processo para que haja a chamada de interrupção INT0. Se em nível alto a interrupção será aceita na transição negativa neste pino permanecendo pelo menos 12 períodos de *clock*, se em nível baixo a interrupção será aceita apenas por nível baixo no pino.

Registrador TMOD time mode (89)

BIT7	Gate
BIT6	C/T1
BIT5	M1.1
BIT4	M0.1
BIT3	Gate
BIT2	C/T0
BIT1	M1.0
BIT0	M0.0

Gate

Este bit tem por função escolher como o T/C1 será habilitado. Se for 0 (função temporizador), o T/C1 estará habilitado, iniciando tal contagem quando o bit TR1(no registrador TCON) for 1. Se o GATE for 1 (função contador), o T/C1 estará habilitado quando TR1 e INT1 estiverem em 1 simultaneamente.

C/T

Este bit seleciona a função de contador ou temporizador. Se o bit T/C1 for 0, a função é de temporizador e o sinal será interno. Se o bit T/C1 for 1, a função será de contador e o sinal será externo.

M1.1 e M0.1

Estes dois bits servem para determinar em que modo o C/T1 irá trabalhar, como segue a tabela a baixo.

M1.1	M0.1	Modo	Operação
0	0	0	C/T 1 de 8 bits com clock /32
0	1	1	C/T1 de 16bits
1	0	2	C/T1 de 8 bits com recarga automática
1	1	3	C/T1 de 8 bits. Dois contadores de 8 bits

Gate

Este bit tem por função escolher como o T/C0 será habilitado. Se for 0 (função temporizador), o T/C0 estará habilitado iniciando tal contagem quando o bit TR0(no registrador TCON) for 1. Se o GATE for 1 (função contador) o T/C0 estará habilitado quando TR0 e INT0 estiverem em 1 simultaneamente .

C/T

Este bit seleciona a função de contador ou temporizador. Se o bit T/C0 for 0, a função é de temporizador e o sinal será interno. Se o bit T/C0 for 1, a função será de contador e o sinal será externo

M1.0 e M0.0

M1.0	M0.0	Modo	Operação
0	0	0	C/T 0 de 8 bits com clock /32
0	1	1	C/T0 de 16bits
1	0	2	C/T0 de 8 bits com recarga automática
1	1	3	C/T0 de 8 bits.Dois contadores de 8 bits

Essa tabela também faz parte dos registradores de funções especiais, porém não são bits endereçáveis , portanto seus bits não tem nomes, e os registradores só podem ser acessados pelo endereço dos bytes.

8D	**Time High 1**	**TH1**
8C	**Time High 0**	**TH0**
8B	**Time Low 1**	**TL1**
8A	**Time Low 0**	**TL0**
87	**Power Control Register**	**PCON**
83	Data Pointer High	DPH
82	Data Pointer Low	DPL
81	Stack Pointer	SP

TH1 Time High 1 (8D) e TL1 Time Low 1 (8B)

São registradores de 8 bits que em conjunto com registradores de 16 bits são chamados de Time 1, Temporizador e contador.

TH0 Time High 0 (8C) e TL0 Time Low 0 (8A)

São registradores de 8 bits que em conjunto com registradores de 16 bits são chamados de Time 0: Temporizador e contador.

PCON Power Control (87)

Esse registrador é usado para alterar modos de funcionamento do microcontrolador com relação ao canal serial e consumo de potência.

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
SMOD	*	*	*	GF1	GF2	PD	IDL

* = bit sem função

SMOD

Dobra a relação de divisão de frequência na serial.

GF1

Bit de uso geral.

FG0

Bit de uso geral.

PD

Bit de Power-Down: modo especial de trabalho do microcontrolador da série CMOS, em que o microcontrolador "congela" suas atividades.

IDL

Bit que ativa o modo "idle", modo especial de trabalho do microcontrolador da série CMOS em que o microcontrolador "congela" suas atividades.

DPH Data Pointer High (83) e DPL Data Pointer Low (82)

Esses registradores juntos formam um registrador de 16 bits chamado de DPTR (Data Pointer) e permite criar um endereço de acesso externo.

SP Stack Pointer (81)

É um registrador utilizado com pilha de endereços de retorno de sub-rotina, ele é indiretamente manipulado pelas instruções CALL , RET e RETI , e acessado diretamente pelas instruções PUSH e POP.

Do endereço 30H ao endereço 7F H da RAM temos registradores comuns com acesso por byte; já do endereço 20H até o endereço 2F H, os registradores podem ser acessados tanto por byte como por bits. Do endereço 00 H ao endereço 1F H existem quatro bancos de registradores que só são bytes endereçáveis, como mostra a tabela abaixo.

7F 30	REGISTRADORES DE USO GERAL ENDEREÇÁVEIS POR BYTES							
2F	**7F**	**7E**	**7D**	**7C**	**7B**	**7A**	**79**	**78**
2E	77	76	75	74	73	72	71	70
2D	**6F**	**6E**	**6D**	**6C**	**6B**	**6A**	**69**	**68**
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	21	20
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**

Do endereço 00H ao endereço 1FH temos conjuntos de registradores divididos em 4 bancos, sendo que cada banco com 8 registradores. Esses registradores têm nomes que vão de R0 a R7 ; os registradores de cada banco têm os mesmos nomes, só mudando o endereço. Os bancos são acessados por duas flags no registrador PSW, como é mostrado na tabela da próxima página.

Qualquer registrador de qualquer banco pode ser acessado pelo endereço imediato ou através do nome (R0 , R1 , R2 , R3 , R4 , R5 , R6 e R7), basta para isso ter selecionado o banco correto. Obs.: o ponteiro de pilha está apontado para o registrador do endereço 07H (R7 do banco 0) sempre que se liga o microcontrolador, mas poderá ser mudado para outro endereço da RAM pelo programador, se for necessário.

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

Endereço absoluto	Nome dos Registradores	Nome dos Conjuntos	Bits no PSW
1F	**R7**		
1E	**R6**		
1D	R5		
1C	R4	BANCO	RS1 RS0
1B	R3	3	1 1
1A	R2		
19	R1		
18	R0		
17	R7		
16	R6		
15	R5		
14	R4	BANCO	RS1 RS0
13	R3	2	1 0
12	R2		
11	R1		
10	R0		
0F	R7		
0E	R6		
0D	R5		
0C	R4	BANCO	RS1 RS0
0B	R3	1	0 1
0A	R2		
09	R1		
08	R0		
07	R7		
06	R6		
05	R5		
04	R4	BANCO	RS1 RS0
03	R3	0	0 0
02	R2		
01	R1		
00	**R0**		

INTERRUPÇÃO

A interrupção é uma das mais importantes ferramentas nos sistemas de controle de um microcontrolador , pois é o processo pelo qual se interrompe a execução de um programa que está em andamento para uma rotina que trata o que o programador faz de acordo com a necessidade de um evento externo ou interno. A vantagem da interrupção está na simplicidade do hardware e do software, pois o sistema não precisa ficar monitorando o funcionamento de certos periféricos. Deve-se, antes de mais nada, conhecer alguns conceitos sobre propriedades das interrupções.

Prioridade

Quando as interrupções estão habilitadas (mais de uma interrupção), para atendê-las deve ter uma ordem de atendimento, impondo qual delas tem que ser atendida primeiro, no caso de duas interrupções chegarem simultaneamente.

Mascaramento

É a forma de se evitar que aconteçam certas interrupções, esse processoé feito geralmente por software. Existem sistemas que não possuem o mascaramento , o que não possibilita que se desabilite as interrupções via software .

Vetorada e não Vetorada

As interrupções vetoradas são aquelas que possuem um vetor de interrupção (endereço de início de interrupção) fixo e que não pode ser mudado pelo programador. As não vetoradas são aquelas em que o programador pode definir o endereço inicial de interrupção.

Tipos de sinais

Existem três tipos de sinais lógicos, que um sistema pode reconhecer:

· Por nível , alto ou baixo
· Por borda , de subida ou de descida
· Por borda , de subida ou de descida e um nível correspondente.

Tempo para atendimento

O reconhecimento da interrupção demora um certo tempo para que sejam feitas as alterações nos registradores internos e efetuado o desvio de endereço, dependendo do tipo de instrução que esteja sendo executada e em que ponto do clock esta instrução está.

INTERRUPÇÕES NA FAMÍLIA 8051

No 8051 temos três maneiras de solicitar as interrupções: por software (instrução), por um periférico externo ou pela solicitação de uma interrupção por periférico interno(timer / counter , porta serial etc.). Existem 5 fontes de interrupções com seus endereços definidos e elas são as mascaráveis INT0 , INT1, TIMER 0, TIME 1 e SERIAL. Cada interrupção pode ser individualmente habilitada ou desabilitada, e também podem ser habilitadas ou desabilitadas todas de uma só vez, como mostra a tabela abaixo (obs.: em outros chips da família 8051 pode-se ter diferença na quantidade de interrupções. O básico é dado pelo 8051).

ENDEREÇO DE DESVIO DE INTERRUPÇÃO

As interrupções no 8051 são "nesting", ou seja, uma interrupção interrompe a outra que já esteja sendo executada, dependendo da prioridade de cada uma . Vamos dar principal atenção ao chip 8051. Os outros da família podem ter detalhes de diferença na quantidade de interrupções, mas o básico é dado pelo 8051. As interrupções do microcontrolador 8051 são as seguintes:

Interrupção externa INT0 (endereço B2h): É um pino físico de interrupção que tem que ser habilitado (e também determinada sua prioridade de atuação) via software, conforme veremos futuramente.

Interrupção externa INT1 (endereço B3h): É outro pino físico de interrupção semelhante ao INT0.

OBS.: As interrupções INT0 e INT1 são pinos do Port P3, exatamente os pinos P3.2 e P3.3, endereços B2h e B3h, respectivamente, que, se utilizados para tal, diminuem o tamanho do Port P3 para outras aplicações.

Interrupção interna gerada pelo TIMER/COUNTER 0: É uma ação de interrupção interna gerada pelo TIMER _0, que é um periférico interno ao microprocessador.

Interrupção interna gerada pelo TIMER/COUNTER 1: É uma ação de interrupção interna gerada pelo TIMER_1, que é o segundo periférico interno ao microprocessador.

Interrupção pela Serial: É uma ação de interrupção interna gerada pelo periférico SERIAL.

Observação: Estes endereços são relativos à ROM/EPROM do microcontrolador. Logo, se utilizarmos estas interrupções, não se poderá gravar um software em cima destes endereços das referidas interrupções, senão o sistema se perde! Observe que listamos o pino de "Reset – RST " como sendo de interrupção, pois ele, na verdade, é uma interrupção não vetorada com o único propósito de retornar o software ao seu ponto inicial.

TEMPORIZADORES / CONTADORES (TIMER 0 e TIMER 1)

O 8051 tem dois registradores de 16 bits para timer / counter, timer 0 e timer

Nível de prioridade	Fonte	Tipo	Pino ou Bit	Endereço para atendimento
	RESET	NÍVEL	9	0000
1 º	INT0	NÍVEL ou BORDA	P3.2	0003
2 º	TIMER0	INTERNO/EXTERNO	SOFT / P3.4	000B
3 º	INT1	NÍVEL ou BORDA	P3.3	0013
4º	TIMER1	INTERNO/EXTERNO	SOFT / P3.5	001B
5 º	SERIAL RXD	DADOS	P3.0	0023

1. Todos podem ser configurados para operar como timer ou counter (temporizador ou contador)

Na configuração timer, o registrador é incrementado todo ciclo de máquina, desde que o ciclo de máquina consista em 12 períodos do oscilador, proporção de contagem de 1/12 da freqüência do oscilador.

Na função counter (contador), o registrador é incrementado em resposta à transição de 1 para 0 na entrada externa correspondente T0 ou T1. Nesta função a entrada é amostrada em todo ciclo de máquina, quando uma amostragem tem um nível 1 em um ciclo e um nível 0 no próximo ciclo , o contador é incrementado.

Um novo valor aparecerá no registrador no ciclo seguinte , aquele em que a transição foi detectada . Se for necessário, por exemplo, dois ciclos para detectar uma transição, a máxima razão de contagem é de 1 / 24 da freqüência do oscilador , não existindo restrições sobre o duty cycle do sinal de entrada. Através do registrador TMOD podemos configurar os T/C's em 4 modos que são:

MODO 0

Contador ou temporizador de 8 bits com divisão de até 32 vezes. Os registradores TH0 ou TH1 recebem o valor de contagem e pode ser até FFh com o valor escrito pelo software, que também pode ser lido a qualquer momento; se ocorrer um estouro *(overflow)* o T/C em questão gera um pedido de interrupção que pode ou não ser aceito pela CPU interna do microcontrolador. Ainda nesse modo os registradores TL0 ou TL1 com os bits 0, 1, 2, 3 e 4 servem para determinar em quanto será dividido o sinal de contagem (interno ou externo) podendo ir até 32 pela combinação binária desses bits; os bits 5 a 7 destes registradores deverão ser ignorados. A figura 8 mostra o funcionamento.

MODO 1

Neste modo temos um T/C de 16 bits e, desta forma, usamos os pares de registradores TH0/TL0 ou TH1/TL0 para efetuar a contagem. Quando um estouro acontecer ou seja quando o par de registradores passa de FFFFh para 0000h, é setado o bit de overflow (TF0 ou Tf1) forçando uma interrupção, se esta tiver sido previamente programada via software. Ver figura 9.

MODO 2

Nos registradores TL0 ou TL1 teremos o valor de onde começará a contagem, e nos registradores TH0 ou TH1 teremos o valor de recarga, os quais serão recarregados em TL0 ou TL1 sempre que um estouro acontecer nos mesmos. Como cada registrador pode ser alterado a qualquer momento pelo

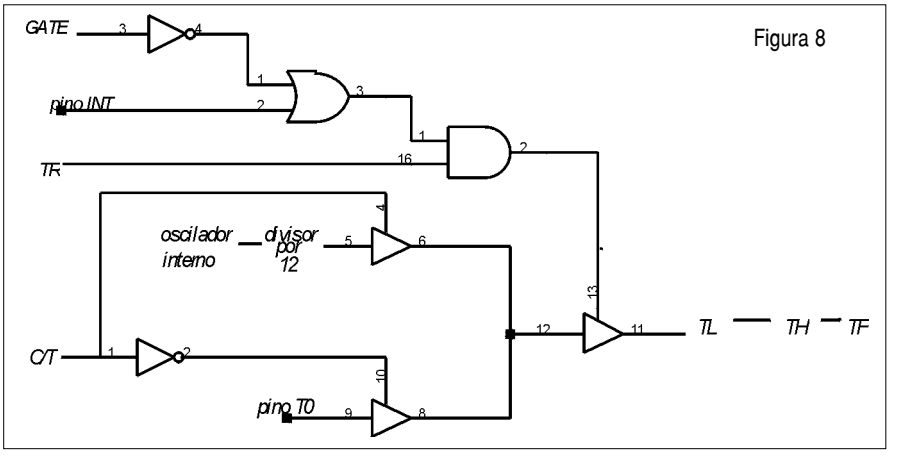


Figura 8

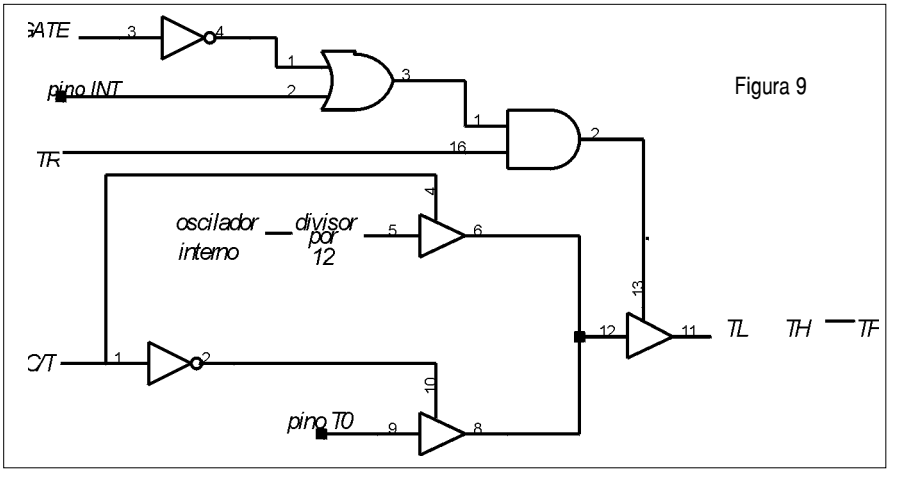


Figura 9

software , temos uma grande flexibilidade para trabalhar com este modo. Observe a figura 10.

MODO 3

Neste modo o T/C1 pára sua operação sem receber pulsos de contagem, e o T/C0 fica com dois sistemas de 8 bits, um em TH0 e outro em TL0.

O T/C de 8 bits TL0 será controlado pelos os bits TR0 e TF0, enquanto que o outro T/C de 8 bits será controlado pelos bits TR_1 e TF_1.

TR1 e TF1

Neste modo. T/C 1 pára sua operação e fica inerte, sem receber pulsos de contagem. Para o T/C 0 temos dois sistemas de 8 bits, um em THO e outro em TL0. O T/C de 8 bits TL0 será controlado pelos bits TR0 e TF0, e o T/C de 8 bits THO será controlado pelos bits TR1 e TF1.

Para o caso de programarmos o T/C 0 para o modo 3, podemos programar T/C 1 para qualquer um dos outros mo-

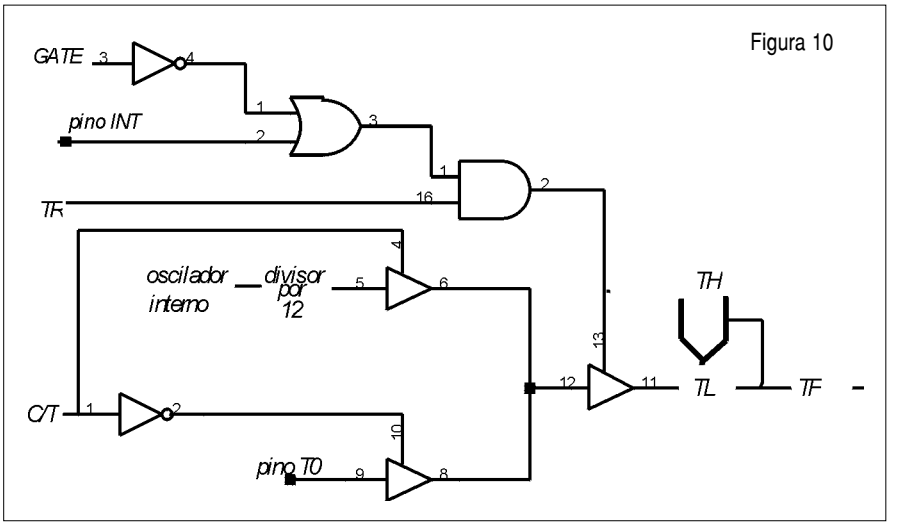


Figura 10

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

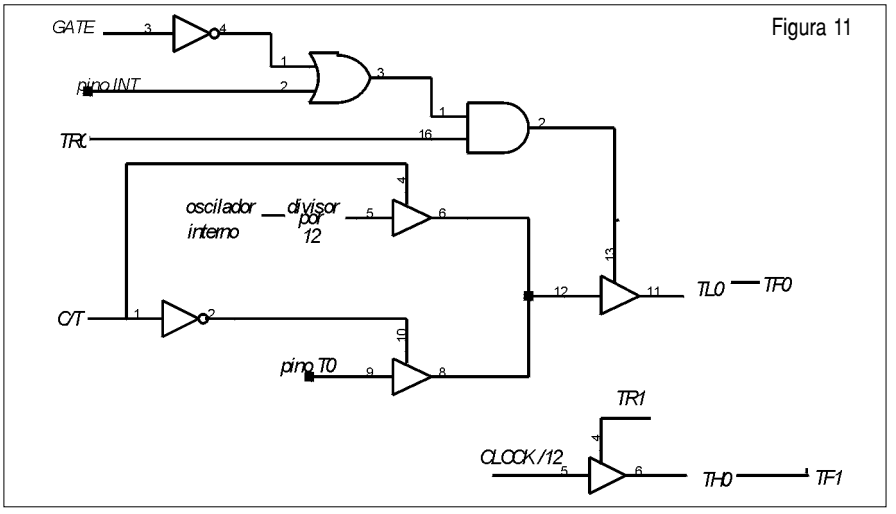


Figura 11

dos restantes, mas este não poderá gerar pedido de interrupção, pois o bit de requisição de interrupção do T/C 1 estará sendo usado por T/C 0, mas mesmo assim o overflow neste contador poderá ser utilizado para chavear o gerador de taxa de transmissão do canal serial. Neste modo, o T/C 0 será habilitado pelo bit de controle do T/C 1, mais especificamente TR1, no registro TCON e, ao ocorrer o overflow de TH0, quem será setado é o bit TF1 e não o TF0. Veja a figura 11.

Se configurarmos o T/C 1 para o modo 3, o registro TL0 poderá ser usado como T/C de 8 bits, mas será controlado pelos bits GATE, C/T, TR0 e TF0. Neste modo o overflow em TH0 acionará o flag de requisição de interrupção referente ao T/C 1, e o overflow de TL0 acionará o flag de requisição de interrupção referente ao TC0.

COMUNICAÇÃO DE DADOS

Para se transmitir dados entre equipamentos são necessários dispositivos de comunicação de dados.

A transmissão de dados pode ser classificada pela sua forma (paralela ou serial), pela forma do sinal transmitido (analógico ou digital) e pelo sentido do sinal na transmissão *(simplex, half-duplex e duplex)*. Para uma transmissão de dados a longa distância, para uma maior fidelidade é necessário que se convertam os sinais digitais em analógicos ao transmitir, e ao receber converter os sinais analógicos em digitais através de aparelhos chamados modems. A denominada comunicação de dados pode ser feita de duas formas: serial e paralela. Neste trabalho será dada ênfase à comunicação serial, que é mais utilizada.

COMUNICAÇÃO SERIAL

A comunicação serial é feita com a transmissão de bytes ou caracteres de bit em bit, um por vez na sequência. Este é um modo de comunicação muito recomendado para transmissões de dados a longa distância. A comunicação serial usa níveis de tensão de 0 e a 5 V TTL e, para facilitar a transmissão de dados entre equipamentos diferentes, utiliza-se do código ASCII (American Standard Code for Interchange of Information), que representa cada caracter como uma palavra binária de 8 bits. Na comunicação serial em níveis de tensão pode-se ter problemas relacionados à distância (como ruídos, por exemplo). Daí o uso de um outro padrão de tensões (no caso do sistema RS-232, utiliza-se de tensões de 30 V), facilitando a transmissão de dados por uma grande distância com poucas interferências).

A comunicação serial pode ser síncrona ou assíncrona: na primeira, além dos bits de dados tem-se também os denominados *bits de sincronismo* (bits que ajustam o clock interno do receptor para que este tenha compatibilidade suficiente para receber os dados do transmissor). Além dos bits de sincronismo, existem os chamados *bits de parada* (bits usados para informar ao receptor o fim dos bits de dados). Esses bits de parada permitem que o receptor confirme se recebeu ou não uma informação corretamente.

No caso da comunicação assíncrona, é transmitido um caracter por vez, bit a bit; onde existe um bit para indicar o início da transmissão, chamado de *start bit*, e outro para indicar o fim da transmissão, o chamado *stop bit.* O start bit é reconhecido como uma transição do nível alto presente na linha para nível baixo, enquanto que o stop bit é reconhecido pela transição do nível baixo para o nível alto. Se o último bit referente ao dado for em nível alto, o sistema aguarda um novo start bit, para iniciar a recepção de um novo caracter.

Para a comunicação serial existem vários tipos de protocolos de comunicação (normas padronizadas para transmissão e recepção de dados), tais como o RS-485 e o RS-232, sendo este último o mais utilizado no mundo atualmente, pois define todos os padrões (elétricos, mecânicos e de transmissão) a serem empregados numa comunicação de dados. Na comunicação serial opera-se com três sistemas de interligação digitais: *simplex, half–duplex e full-duplex*. Figura 12.

No modo simplex, temos apenas um elemento que transmite e um outro que recebe os dados, este modo é muito utilizado em impressoras e terminais de dados de bancos.

No modo half-duplex, também temos um elemento que transmite e um outro que recebe, só que não é possível transmitir e receber as informações ao mesmo tempo, de modo simultâneo, como nos *walkie-talkies*, por exemplo.

No modo full-duplex ou apenas duplex, é possível transmitir e receber dados simultaneamente. O microcontrolador 8051 tem uma interface serial interna que utiliza exatamente deste modo de interligação de sistema digital.

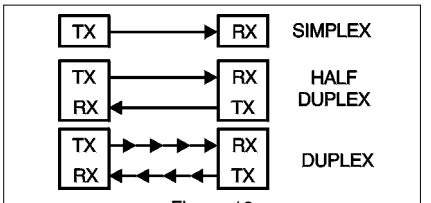


Figura 12

INTERFACE SERIAL NO 8051

No 8051 a interface serial já é implementada no próprio chip, e ela é do tipo Full-Duplex ou, conforme usaremos no decorrer do texto duplex onde, neste caso, dados podem estar sendo recebidos e enviados simultaneamente no hardware sem necessitar a intervenção direta do programa (a CPU é sinalizada via interrupção, quando alguma atitude do programa for precisa: recepção ou envio de um byte).

Isto significa que o sistema pode receber e transmitir informações simultaneamente, sendo que para tal temos registros especialmente destinados para este fim. Este registro chama-se **SBUF** *(Serial buffer)* e uma escrita no mesmo implica em transmissão automática do dado escrito: assim como um certo dado que chegue no pino de recepção implicará na automática operação de recepção por parte do sistema, independentemente do controle do usuário (desde que o canal serial esteja habilitado).

Pode parecer, à primeira vista, que temos um único registro físico para efetuar este trabalho. Isto não é verdade. Temos, na realidade, dois registros com o mesmo nome SBUF, sendo que um é para recepção e outro para transmissão.

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

O reconhecimento é feito pelo sistema através das instruções que o utilizam. Se for instrução de escrita, o registro de transmissão será alterado, e se for instrução de leitura, o dado será captado no registro de recepção.

A transmissão é automática ao escrevermos um dado no registro SBUF, logo, se não desejarmos usar transmissão serial, basta não endereçarmos este registro como destino. É claro que para fazermos uma transmissão é necessário antes configurar parâmetros caracterizadores da comunicação serial: modo de transmissão, velocidade de transmissão, quantidade de bits de dados, quantidade de *stop bits* e tipo de paridade adotada.

MODO 1 DE OPERAÇÃO DO CANAL SERIAL

Neste modo de operação e nos próximos (assíncronos), o pino de recepção é o RXD (pino 10, porta P3.0) e o de transmissão é o TXD (pino 11, porta P3.1).

Neste modo são transmitidos/recebidos 10 bits em cada pacote, sendo um *start bit* (nível 0) seguido de 8 bits de dados e o stop bit *(nível 1)* (figura 13). A taxa de transmissão é variável e sua geração será vista em um item a seguir.

Neste modo, na recepção o stop bit vai para o bit RB8 no SCON.

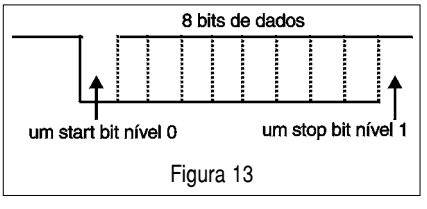


Figura 13

MODO 2 DE OPERAÇÃO DO CANAL SERIAL

Neste modo, em cada pacote são transmitidos/recebidos 11 bits, sendo um start bit (nível 0) seguido de 8 bits de dados, um nono bit (de livre escolha do usuário, 0 ou 1) e um stop bit (nível 1)(figura 14). A taxa de transmissão pode ser escolhida para 1/32 ou 1/64 da freqüência de clock do sistema.

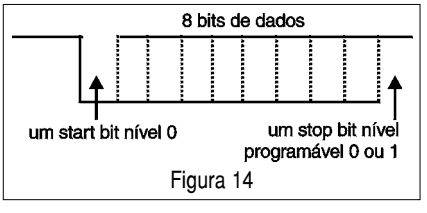


Figura 14

No caso da transmissão, o nono bit pode ser escolhido entre 0 ou 1, apenas escrevendo-se o valor desejado no bit TB8 do registro SCON. Assim como no modo 1, o nono bit recebido vai para o bit RB8 no SCON e o stop bit será ignorado.

MODO 3 DE OPERAÇÃO DO CANAL SERIAL

Este modo é idêntico ao modo 2, exceto pela taxa de transmissão que é variável.

GERAÇÃO DAS TAXAS DE SINALIZAÇÃO

Neste item veremos como gerar taxas de sinalização variáveis, nos modos que as permitam. Para isto será necessário estudarmos o registro de função especial PCON.

O registrador PCON contém bits para controle dos modos de potência no chip (para que em caso de falta de energia elétrica, não percamos memória), flags de uso geral é um bit em particular, chamado SMOD, que controla a divisão da taxa de freqüência do canal serial. Como este bit não é endereçável devemos carregar o bit SMOD indiretamente e se não utilizarmos as funções dos outros bits, poderemos deixá-lo em 0. Nos modos 1e 3 a taxa de transmissão é fornecida pelo Timer / Counter, onde cada ordem de transmissão é gerada ao ocorrer o *overflow* deste contador. Para isso, devemos desabilitar a interrupção deste **Timer/Counter**.

O Timer/counter 1 pode ser configurado para temporizador ou contador em qualquer um dos três modos de operação. O mais comum é usá-lo no modo de recarga automática (timer de 8 bits) e neste caso temos a seguinte fórmula simplificada.

A formula é: (todos os valores em decimal depois transformar o resultado em Hex)

$$\text{Baud-Rate} = \frac{\text{Clock}}{N\,(256 - TH1)} \qquad \text{ou} \qquad TH1 = 256 - \left(\frac{\text{Clock}}{N \,.\, \text{Baud-Rate}}\right)$$

N = 384, se SMOD for = 1.
N = 192, se SMOD for = 0.

COMUNICAÇÃO ENTRE VÁRIOS 8051

Os modos 2 e 3 nos permite interligar vários 8051, sendo um deles o mestre e os demais escravos. Esta interligação pode ser vista na Figura abaixo 15.

Nestes modos, os bytes de dados são compostos de:
- um start bit.
- 8 bits de dados.
- Um nono bit que pode ser 0 ou 1 e que vai para o bit RB8.
- Um stop bit.

Na recepção, se SM2 = 1 e RB8 = 1, a interrupção do canal serial será atendida. Assim, podemos criar condições para que um 8051 mestre possa enviar dados serialmente para vários 8051 'escravos', da seguinte forma:

1 – No início do trabalho, todos os 'escravos' estarão com SM2 = 1 (prontos para receber dados, cujo nono bit seja 1).

2 – Quando o 8051 mestre desejar enviar dados para alguns dos 'escravos'.

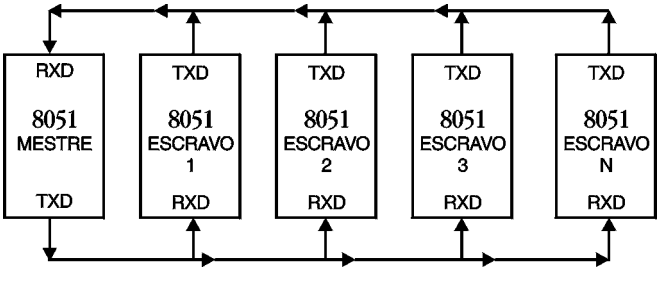
3 – Escreverá um em seu bit TB8.

4 – Enviará serialmente o endereço do escravo desejado (8 bits).

5 – Como teremos todos os bits dos 'escravos' com RB8 em 1, então, todos serão interrompidos para poder verificar se é seu endereço o enviado.

6 – O escravo que for selecionado zerará seu bit SM2 e estará preparado para receber os dados, os quais deverão ter agora o nono bit (RB8) em 0. Os demais 'escravos' permanecerão com SM2 em 1 e, desta forma, não serão

Figura 15



Opções de Baud-Rate no modo 2

Clock utilizado	Baud-Rate desejado	Baud-Rate obtido	Erro %	Bit SMOD	Carga do Time 1
11,059 MHz	1.200	1.200	0	0	E8h
11,059 MHz	2.400	2.400	0	0	F4h
11,059 MHz	9.600	9.600	0	0	FDh
11,059 MHz	19.200	19.200	0	1	FDh
12,000 MHz	1.200	1.202	0.16	0	E6h
12,000 MHz	2.400	2.404	0.16	1	E6h
12,000 MHz	9.600	8.923	7	1	F9h
12,000 MHz	19.200	20.833	8.5	1	FDh

mais interrompidos, pois os dados têm RB8 = 0 e poderão continuar suas atividades normalmente.

7 – Quando o mestre desejar novamente enviar um endereço, bastará enviar novamente o nono bit em 1, para habilitara interrupção em todos os 'escravos'.

EXPANSÃO DE I / O

8255

A PPI 8255 é uma interface de finalidade geral programável, compatível com o 8051, que apresenta as seguintes características :

* Alimentação = 5V
* três portas de oito bits cada , programáveis por software
* programação feita utilizando-se instruções de entrada e saída (IN e OUT) e uma palavra de controle
*operação realizada em três modos distintos: modo 0, modo 1, modo 2.

Pinagem

Os pinos da 8255 têm a seguinte configuração mostrados na figura 16, enquanto a figura 17 mostra o endereçamento.

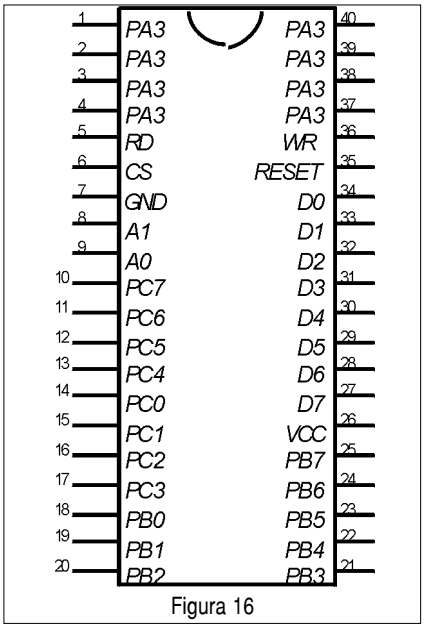


Figura 16

Correspondência entre pino e sinais

PINO	SINAL	FUNÇÃO
1	PA3	Bit 3 da porta A
2	PA2	Bit 2 da porta A
3	PA1	Bit 1 da porta A
4	PA0	Bit 0 da porta A
5	RD	Sinal ativo em zero que habilita o envio de dados através da interface para o microcontrolador através do barramento de dados
6	CS	Sinal ativo em zero que habilita o funcionamento da interface
7	GND	Referência 0V
8	A1	A1 e A0 são conectados no barramento de endereço do sistema , usado para selecionar uma das três portas ou registrador da palavra de controle do 8255
9	A0	
10	PC7	Bit 7 da porta C
11	PC6	Bit 6 da porta C
12	PC5	Bit 5 da porta C
13	PC4	Bit 4 da porta C
14	PC0	Bit 0 da porta C
15	PC1	Bit 1 da porta C
16	PC2	Bit 2 da porta C
17	PC3	Bit 3 da porta C
18	PB0	Bit 0 da porta B
19	PB1	Bit 1 da porta B
20	PB2	Bit 2 da porta B
21	PB3	Bit 3 da porta B
22	PB4	Bit 4 da porta B
23	PB5	Bit 5 da porta B
24	PB6	Bit 6 da porta B
25	PB7	Bit 7 da porta B
26	VCC	Tensão de alimentação
27	D7	Bit 7 de dados
28	D6	Bit 6 de dados
29	D5	Bit 5 de dados
30	D4	Bit 4 de dados
31	D3	Bit 3 de dados
32	D2	Bit 2 de dados
33	D1	Bit 1 de dados
34	D0	Bit 0 de dados
35	RESET	Sinal vindo do sistema para limpar os registradores ativos em nível 1
36	WR	Sinal de controle, ativo em nível zero que habilita a operação de escrita
37	PA7	Bit 7 da porta A
38	PA6	Bit 6 da porta A
39	PA5	Bit 5 da porta A
40	PA4	Bit 4 da porta A

SOFTWARE

O software, também conhecido como programa, é um conjunto de instruções colocadas em um seqüência lógica, pela qual o processador poderá tomar algumas "decisões" já definidas pelo programador.

A programação de um sistema é a construção correta dessa seqüência de instruções que ele deve realizar para conduzir à solução de um problema. Essa seqüência é variável, pois o programador possui uma ampla variedade de alternativas e opções para atingir os mesmos resultados.

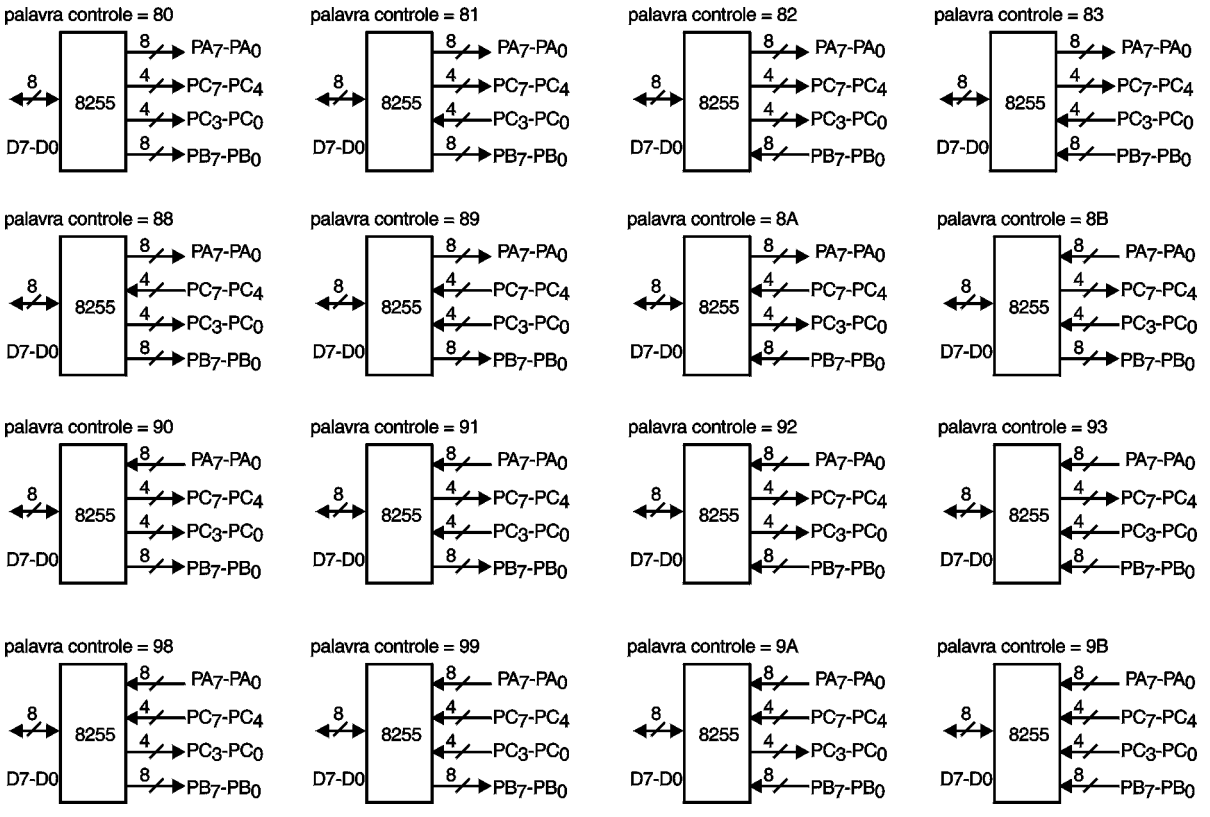
Para se programar um sistema é necessário que se conheça a parte do hardware, ou seja , a arquitetura interna do microcontrolador a ser programado, seus periféricos, o set de instrução, etc.

INSTRUÇÃO

A instrução pode ser definida como sendo uma "ordem dada para executar uma certa tarefa". No microprocessador essas tarefas podem ser :

- ler ou escrever numa posição de memória;
- fazer uma operação aritmética e lógica;
- fazer qualquer outra manipulação de dados, etc.

A instrução para um microprocessador pode ser representada por linguagem de máquina, códigos em hexadecimal ou binário.



Endereçamento

O endereçamento é feito por meio da combinação dos bits de endereço A0 e A1.

A1	A0	Porta selecionada
0	0	Porta A
0	1	Porta B
1	0	Porta C
1	1	Unidade de controle

Fig. 17 - Endereçamento.

Outra maneira de representação pode ser feita através dos mnemônicos (códigos que representam uma sequência de palavras) de um programa fonte, que serão transformados em linguagem de máquina (ou linguagem objeto) por um programa compilador (tradutor).

LINGUAGEM ASSEMBLY

A linguagem ASSEMBLY foi a primeira linguagem que surgiu nesta área de computação, sendo formada por mnemônicos.

Cada mnemônico possui o um código correspondente em hexadecimal, e cada 'família' de microcontrolador possui um conjunto de mnemônicos diferente. Esta linguagem tem como finalidade facilitar o manuseio das instruções para desenvolver programas, desde uma simples rotina até um sofisticado programa para controle industrial.

Todo microcontrolador precisa de um lista de instruções para ser seguida passo-a-passo, lista essa chamada de programa, que diz exatamente o que o microcontrolador deve fazer.

O microcontrolador só faz duas coisas: lê e processa números binários.

Cada fabricante de microcontrolador define os mnemônicos para os seus produtos .

Algumas linguagens assembly utilizam o mesmo código (instrução de máquina) para diversas CPU´s (Unidade Central de Processamento) porém antes de iniciar a tradução deve-se informar ao tradutor para qual CPU deve ser feita a tradução.

A linguagem assembly é classificada de baixo nível , porque na elaboração do programa há a necessidade de elaborar rotinas para o completo controle da CPU.

ASSEMBLER:

O tradutor assembler é um programa de computador que produz um código binário correspondente a cada mnemônico. Os programas de tradução avisam quando encontram algum erro no programa fonte (programa elaborado para a CPU).

ESTRUTURA DA LINGUAGEM ASSEMBLY

A linguagem ASSEMBLY é dividida em grupos , e nessa ordem:

Label	Código de Operação	Operando ou Endereço	Comentário

LABEL

A principal característica da linguagem ASSEMBLY é o uso de label (lê-se 'lêibol).

O label torna a programação em ASSEMBLY mais rápida e segura, porque ao invés de lidar com um endereço absoluto, lida-se com um conjunto de caracteres (label). Este artifício é útil para se seguir o fluxo de dados, e também quando se acrescenta ou se retira alguma instrução, pois o novo endereço do label é automaticamente corrigido pelo programa compilador Assembler.

A única restrição é que o primeiro caracter do label deve ser sempre uma letra, isto é, ele não pode começar com um número.

CÓDIGO DE OPERAÇÃO

O código de operação é simplesmente a instrução em si. No código de operação aparecerá apenas o conjunto de caracteres, que representará o tipo de instrução, tal como soma, carregamento, teste, entre outras.

OPERANDO OU ENDEREÇO

Este grupo é dividido em duas partes. A primeira parte da informação é chamada de 'operando', sendo a designação dada para mencionar o fluxo de dados. A segunda é chamada de 'endereço' e é utilizada pelas instruções de "CALL" e "JMP".

COMENTÁRIO

Este grupo se destina ao comentário. É considerado muito importante num programa.

Os comentários facilitam a visualização do fluxo do programa. O comentário deve, na maioria das vezes, dizer para que se destina a instrução, e não simplesmente descrevê-la.

DELIMITADORES

Os delimitadores são usados para separação dos grupos : label, código de operação, operando ou endereço e comentários.

Um espaço ou um conjunto de espaços em branco serve para separar cada grupo.

Dois pontos delimitam o fim de um label. Em uma 'pseudo-instrução' (instrução que não pertence ao conjunto de instruções da CPU) não há necessidade de dois pontos depois de um label.

A vírgula separa o operando de endereço e o ponto-e-vírgula indica o início de comentário.

PSEUDO-INSTRUÇÃO

Pseudo-Instrução é um termo usado em linguagem ASSEMBLY, também chamado de "Instrução Especial".

Pseudo-Instrução pode ser empregada na definição de label, definição de nome de programa, definição de 'string'(palavra) , definição do endereço inicial do programa, reserva de área de memória e definição de byte.

Sempre que se define um dado hexadecimal em um programa que começa com letra (A, B, C, D,E, ou F) deve-se colocar um zero na frente; os números em hexadecimal deverão ter a letra "H" no final; os números em binário deverão ter a letra "B"; os números em octal deverão ter as letras "O" ou "Q" e os números em decimal deverão ter a letra "D" ou somente o próprio número.

Abaixo estão relacionadas algumas pseudo-instruções importantes:

TITTLE (nome do programa)
Esta pseudo-instrução serve para nomear um programa e, quando este programa for impresso, esta pseudo-instrução aparecerá em todas as folhas.

DB " X"
Esta pseudo-instrução é usada para definir um byte para quando se quer definir um tabela de dados ou para inserir um dado no meio do programa.

Exemplos:
DB "F"
DB "FRASES"

ORG (endereço)
Esta pseudo-instrução é usada para definir uma posição de memória de onde o programa deve começar , ou seja, carrega o PC (Program Counter) com o endereço definido

Exemplo:
ORG 08H

EQU
Esta pseudo-instrução é usada para definir um label , (NOME EQU NN)

Exemplos:
AUXI EQU 82H

END
Esta pseudo-instrução é usada para indicar o fim de um programa.

$
Esta pseudo-instrução é usada para representar o endereço onde se localiza o 'program counter', geralmente usado em JUMP relativos.

Exemplo:
JMP $

CONJUNTO DE INSTRUÇÕES DO 8051

No conjunto de instruções do 8051, elas podem ocupar de um a três bytes na posição de memória de programa e gastar de um a quatro ciclos de máquina para serem executadas; esta variação no tempo de execução existe porque as formas de endereçamento são diferentes e têm instruções que usam mais de um ciclo de busca. Cada ciclo de máquina gasta 12 períodos de clock, com um cristal de 12 MHz e cada ciclo de máquina tem o tempo de 1µs.

ENDEREÇAMENTOS:

ENDEREÇAMENTO MODO DIRETO

Neste modo são as instruções que utilizam diretamente os primeiros 128 endereços das RAM internas , registradores de I/O ,status e controle, logo após o seu 'opcode' (código da instrução).

Exemplos:

MOV A, XXH = Move para o acumulador o conteúdo do endereço XXH

MOV XXH, A = Move para o endereço XXH, o conteúdo do acumulador

MOV XXH,YYH = Move para o endereço XXH, o conteúdo do endereço YYH

ADD A, XXH = Grava no acumulador o resultado da soma deste com o conteúdo de memória do endereço XXH

ENDEREÇAMENTO MODO INDIRETO

Neste modo o endereço alvo é obtido indiretamente através dos registradores R0 e R1.

Exemplos:

MOV R1, #44H = Grava no registrador R1 o valor 44 H

MOV @R1, #77 = Grava o valor 77 no endereço 44H, que foi obtido indiretamente

ANL A, @R1 = Grava o resultado da operação AND entre o acumulador e o que estiver no endereço 44H. Esse resultado é gravado no acumulador.

MODO REGISTRADOR

Na utilização deste modo há a inclusão do nome do registrador no opcode, ocasionando a economia de um byte de endereço. Os bits RS1 e RS0 do registrador PSW selecionarão o banco de registradores (0, 1, 2 e 3) que irão afetar o registrador.

Exemplos:

MOV R4, A = Grava o conteúdo do acumulador no registrador R4

MOV A, Rn = Grava o conteúdo do registrador Rn no acumulador

MOV R3, #44H = Grava o valor hexadecimal de 44 no registrador R3

DEC R2 = Decrementa o conteúdo do registrador R2

XRL A, Rn = Grava o resultado da operação XOR entre o acumulador e o registrador Rn. Este resultado é gravado no acumulador

XCH A, R6 = Alterna os dados entre o registrador R6 e o acumulador

CJNE R5, #33.rel = Faz a comparação do conteúdo do registrador R5 com o valor 33. Se estes forem diferentes, daí o fluxo de execução é desviado para o endereço indicado por 'rel'.

MODO ESPECÍFICO A REGISTRADOR

Aqui utiliza-se de instruções de apenas um byte, uma vez que o registrador (no caso o do exemplo) já estará incluído no opcode.

Exemplos:

DA A = Converte o conteúdo do acumulador em código BCD

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

RL A = Rotaciona para a esquerda os bits do acumulador

MUL AB = Multiplica o conteúdo do acumulador pelo conteúdo do registrador B, gravando a parte mais significativa do resultado em B e a menos significativa no acumulador

CLR C = Escreve no flag carry o dígito "0"

INC DPTR = Incrementa o conteúdo do registrador de 16 bits. DPTR.

MODO CONSTANTE IMEDIATO

Este modo permite que possa operar com dados diretamente na instrução.

Exemplos:

MOV B, #54 = Grava o valor 54 no registrador B

MOV R3, #87H = Grava o valor 87 em hexadecimal no registrador R3

ORL A, #67 = Grava no acumulador o resultado de uma operação OR entre o acumulador e o valor 67.

MODO ENDEREÇAMENTO À MEMÓRIA DE PROGRAMA

Este modo permite a utilização de instruções que fazem somente a leitura da memória de programa, sendo muito utilizadas para a leitura de tabelas na EPROM. O endereço denominado 'alvo' é formado pela soma do conteúdo do registrador de 16 bits DPTR (data pointer) ou PC (program counter) com o conteúdo do acumulador.

Exemplos:

MOVC A, @A+DPTR = Grava o byte lido no endereço resultante da soma do acumulador com o registrador de 16 bits DTPR , no acumulador.

MOVC A, @A+PC = Grava o byte lido no endereço resultante da soma do acumulador com o registrador de 16 bits PC , no acumulador.

MODO ENDEREÇAMENTO À MEMÓRIA DE DADOS EXTERNA

Este modo permite o acesso à memória RAM externa, pela instrução MOVX. Acesso este feito através de ponteiros de 16 bits (@ DPTR e @Ri), de forma indireta. Na utilização do ponteiro @Ri, os 8 bits menos significativos estarão no registrador Ri (R0 ou R1) e os 8 bits restantes estarão na porta P2, formando assim o endereço de 16 bits.

Exemplos:

MOVX @DPTR, A = Grava o valor do acumulador no endereço DPTR, na memória RAM externa.

MOVX A, @Ri = Grava o conteúdo do endereço X: (Ri + P2), no acumulador.

TIPOS DE INSTRUÇÕES

Os microcontroladores da família 8051 têm vários tipos diferentes de instruções, sendo cada uma representada por seu mnemônico e pelo código em hexadecimal.

Estas instruções são mostradas a seguir:

As anotações seguintes são usadas nas tabelas a seguir (modo de endereçamento).

Rn	pode ser registradores de R0 a R7
Ri	indica registrador R0 ou R1
@Ri	endereçado pelo valor de R0 ou R1
#Dado	valor constante, numeral de 8 bits: #20H(hex), #30(dec), #01010101B(bin)
#Dado16	valor constante, numeral de 16 bits: #1FF2H(hex)
Direto	um endereço de memória RAM interna (8 bits), registradores de status e controles, e portas
End16	endereço de 16 bits para ROM (usado por LCALL e LJMP)
End11	endereço de 11 bits para ROM (usado por ACALL e AJMP)
rel	endereço relativo ou utilização de label
bit	variável da RAM interna, bits de I/O , bits de status e controle

TABELAS

As tabelas a seguir mostram os mnemônicos e suas descrição.

Tabela de Operações Aritméticas
Mnemônico Descrição
ADD A,Rn Soma o conteúdo de Rn com ACC, o resultado da soma é gravado no ACC.
ADD A, direto Soma o conteúdo da posição de memória com ACC, o resultado é gravado no ACC.
ADD A, @Ri Soma o conteúdo da posição de memória indicado por Ri (R1 ou R0) com ACC, gravando o resultado no ACC.
ADD A, # dado Soma o dado ao ACC. O resultado é gravado no ACC.
ADDC A, Rn Soma o conteúdo de Rn (R0 à R7) ao ACC, e com a flag carry, o resultado é gravado no ACC.
ADDC A, direto Soma o conteúdo da posição de memória com o ACC, e com flag carry, o resultado é gravado no ACC.
ADDC A, @Ri Soma o conteúdo da posição de memória indicado por Ri (R1 ou R0) com o ACC, e com a flag carry, o resultado é gravado no ACC.
ADDC A, # dado Soma o dado ao ACC, e a flag carry, o resultado é gravado no ACC.
SUBB A, Rn Subtrai do ACC, o conteúdo de Rn (R0 à R7) e o 'vem um' (borrow), o resultado é gravado no ACC.
SUBB A, direto Subtrai do ACC. o conteúdo da posição de memória e o 'vem um' (borrow), o resultado é gravado no ACC
SUBB A, @Ri Subtrai do ACC, o conteúdo da posição de memória indicado por Ri (R1 ou R0) e o vem um (se existir) o resultado é gravado no ACC.
SUBB A, # dado Subtrai do ACC, o dado e o 'vem um' (se existir), o resultado é gravado no ACC.
INC A Soma 1 ao ACC.
INC Rn Soma 1 ao conteúdo de Rn (R0 a R7).
INC direto Soma 1 ao conteúdo da posição de memória.
INC @Ri Soma 1 ao conteúdo de memória indicado por Ri (R0 ou R1).
DEC A Subtrai 1 do ACC.
DEC Rn Subtrai 1 do conteúdo de Rn (R0 a R7).
DEC direto Subtrai 1 do conteúdo da posição de memória.
DEC @Ri Subtrai 1 do conteúdo da posição de memória indicado por Ri (R0 ou R1).
INC DPTR Soma 1 ao registrador DPTR.
MUL AB Multiplica o conteúdo do ACC pelo conteúdo do registrador B. O resultado fica em B (MSB) e ACC (LSB)
DIV AB Divide o conteúdo do ACC pelo conteúdo do registrador B. O resultado fica em A e o resto em B.
DA A Converte em BCD, o conteúdo do ACC.

Tabela de Operações Lógicas
Mnemônico Descrição
ANL A, Rn Grava no acumulador o resultado da operação lógica AND entre o acumulador e registrador Rn

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

ANL A, direto
Grava no acumulador o resultado da operação lógica AND entre o acumulador e conteúdo do endereço "direto"

ANL A, @Ri
Grava no acumulador o resultado da operação lógica AND entre o acumulador e o conteúdo endereçado pelo registrador Ri

ANL A, #dado
Grava no acumulador o resultado da operação lógica AND entre o acumulador e o dado

ANL direto, A
Grava no endereço "direto" o resultado da operação lógica AND entre o endereço "direto" e o acumulador

ANL direto,#dado
Grava no endereço "direto" o resultado da operação lógica AND entre o endereço "direto" e o dado

ORL A, Rn
Grava no acumulador o resultado da operação lógica OR entre o acumulador e registrador Rn

ORL A, direto
Grava no acumulador o resultado da operação lógica OR entre o acumulador e conteúdo do endereço "direto"

ORL A, @Ri
Grava no acumulador o resultado da operação lógica OR entre o acumulador e o conteúdo endereçado pelo registrador Ri

ORL A, #dado
Grava no acumulador o resultado da operação lógica OR entre o acumulador e dado

ORL direto, A
Grava no endereço "direto" o resultado da operação lógica OR entre o endereço "direto" e o acumulador

ORL direto,#dado
Grava no endereço "direto" o resultado da operação lógica OR entre o endereço "direto" e o dado

XRL A, Rn
Grava no acumulador o resultado da operação lógica XOR entre o acumulador e registrador Rn

XRL A, direto
Grava no acumulador o resultado da operação lógica XOR entre o acumulador e conteúdo do endereço "direto"

XRL A, @Ri
Grava no acumulador o resultado da operação lógica XOR entre o acumulador e o conteúdo endereçado pelo registrador Ri

XRL A, #dado
Grava no acumulador o resultado da operação lógica XOR entre o acumulador e o dado

XRL direto, A
Grava no endereço "direto" o resultado da operação lógica XOR entre o endereço "direto" e o acumulador

XRL direto,#dado
Grava no endereço "direto" o resultado da operação lógica XOR entre o endereço "direto" e o dado

CLR A
Zera o acumulador

CPL A
Inverte todos os bits do acumulador

RL A
Rotaciona todos os bits do acumulador para esquerda

RLC A
Rotaciona todos os bits do acumulador para esquerda junto com a flag carry

RR A
Rotaciona todos os bits do acumulador para direita

RRC
Rotaciona todos os bits do acumulador para direita junto com a flag carry

SWAP A
Troca os nibbles do acumulador (equivale a 4 instruções RL A)

Transferência de Dados

Mnemônico
Descrição

MOV A, Rn
Carrega o acumulador com o conteúdo do registrador Rn

MOV A, direto
Carrega o acumulador com o conteúdo do endereço "direto" (endereços dos registradores internos)

MOV A, @Ri
Carrega o acumulador com o conteúdo endereçado pelo registrador Ri

MOV A, #dado
Carrega o acumulador com o dado

MOV Rn, A
Carrega o registrador Rn com o conteúdo do acumulador

MOV Rn, direto
Carrega o registrador Rn com o conteúdo do endereço "direto"

MOV Rn, #dado
Carrega o registrador Rn com o dado

MOV direto, A
Carrega o endereço "direto" com o conteúdo do acumulador

MOV direto, Rn
Carrega o endereço "direto" com o conteúdo do registrador Rn

MOV direto1,direto2
Carrega o endereço "direto1" com o conteúdo endereço2 "direto2"

MOV direto, @Ri
Carrega o endereço "direto" com o conteúdo endereçado pelo registrador Ri

MOV direto, #dado
Carrega o endereço "direto" com o dado

MOV @Ri, A
Carrega o registrador endereçado por Ri com o conteúdo do acumulador

MOV @Ri, direto
Carrega o registrador endereçado por Ri com o conteúdo do endereço "direto"

MOV @Ri, dado
Carrega o registrador endereçado por Ri com o dado

MOV DPTR,dado16
Carrega o registrador DPTR com o dado de 16 bits

MOVC A,@A+DPTR
Carrega o acumulador com o conteúdo endereçado pelo acumulador mais o registrador DPTR

MOVC A, @A+PC
Carrega o acumulador com o conteúdo endereçado pelo acumulador mais o registrador PC

MOVX A, @Ri
Carrega o acumulador com o conteúdo da RAM externa endereçado pelo registrador Ri

MOVX A, @DPTR
Carrega o acumulador com o conteúdo da RAM externa endereçado pelo registrador DPTR

MOVX @Ri, A
Carrega o registrador da RAM externa endereçado pelo registrador de 8 bits Ri , com o conteúdo do acumulador

MOVX @DPTR, A
Carrega o registrador da RAM externa endereçado pelo registrador de 16 bits DPTR, com o conteúdo do acumulador

PUSH direto
Incrementa o registrador SP e salva na pilha o conteúdo do endereço "direto"

POP direto
Grava na memória o conteúdo da pilha e decrementa o registrador SP

XCH A, Rn
Troca os dados do acumulador com o registrador Rn

XCH A, direto
Troca os dados do acumulador com o endereço "direto"

XCH A, @Ri
troca os dados do acumulador com o registrador endereçado por Rn

XCHD A, @Ri
Troca os nibbles menos significativos do acumulador com o registrador endereçado por Ri

Variáveis Booleanas Bit

Mnemônico
Descrição

CLR C - grava 0 no flag carry

CLR Bit
Grava 0 no bit endereçável (bit de pino ou bit de registradores)

SETB C - grava 1 no flag carry

SETB Bit
Grava 1 no bit endereçável (bit de pino ou bit de registradores)

CPL C - Complementa o flag carry

CPL Bit
Complementa o bit endereçável (bit de pino ou bit de registradores)

ANL C, Bit
Grava no flag carry o resultado da operação lógica AND entre o flag carry e o bit endereçável (bit de pino ou bit de registradores)

ANL C, \Bit
Grava no flag carry o resultado da operação lógica AND entre o flag carry e o complemento do bit endereçável (bit de pino ou bit de registradores).

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

ORL C, Bit
Grava no flag carry o resultado da operação lógica OR entre o flag carry e o bit endereçável (bit de pino ou bit de registradores)

ORL C, \Bit
Grava no flag carry o resultado da operação lógica OR entre o flag carry e o complemento do bit endereçável (bit de pino ou bit de registradores)

MOV C, Bit
Grava no flag carry o conteúdo do bit endereçável (bit de pino ou bit de registradores)

MOV Bit, C
Grava no bit endereçável (bit de pino ou bit de registradores) o conteúdo do flag carry

Fluxo de Execução

Mnemônico
Descrição

ACALL End 11
Chamada curta de sub-rotina (11 bits, 2 kbytes da posição atual)

LCALL End 16
Chamada longa de sub-rotina (16 bits , qualquer posição da EPROM)

RET
Retorno de sub-rotina

RETI
Retorno de sub-rotina de interrupção

AJMP End 11
Desvio curto para endereço de 11 bits, 2 kbytes da posição atual

LJMP End 16
Desvio longo para endereço de 16 bits, qualquer posição da EPROM

SJMP rel
Desvio relativo curto

JMP @A+DPTR
Desvio indireto para a posição de memória endereçada pelo acumulador mais o registrador DPTR

JZ rel
Desvia para o endereço "rel" se o acumulador for igual a zero

JNZ rel
Desvia para o endereço "rel" se o acumulador for diferente de zero

JC rel
Desvia para o endereço "rel" se o flag carry estiver setado

JNC rel
Desvia para o endereço "rel" se o flag carry estiver zerado

JB bit, rel
Desvia para o endereço "rel" se o bit endereçável estiver setado

JNB bit, rel
Desvia para o endereço "rel" se o bit endereçável estiver zerado

JBC bit, rel
Desvia para o endereço "rel" se o bit endereçável estiver setado e depois zera o bit

CJNE A,direto,rel
Compara o conteúdo do acumulador com o conteúdo do endereço "direto", e desvia para o endereço "rel" se for diferente

CJNE A,#dado,rel
Compara o conteúdo do acumulador com o dado, e desvia para o endereço "rel" se for diferente

CJNE Rn,#dado,rel
Compare o conteúdo do registrador Rn com o dado, e desvie para o endereço "rel" se for diferente

CJNE @Ri,#dado,rel
Compara o conteúdo do endereçado por Ri com o dado, e desvie para o endereço "rel" se for diferente

DJNZ Rn, rel
Decrementa o registrador Rn e desvia para o endereço "rel" se o resultado for diferente de zero

DJNZ direto, rel
Decrementa o endereço "direto" e desvia para o endereço "rel" se o resultado for diferente de zero

NOP
Não faz nada.

As tabelas a seguir mostram os mnemônicos, tamanho em byte, código em hexa e número de ciclo de cada instrução. É importante saber o tempo de cada instrução quando se for trabalhar com rotinas de tempo.

Operações Aritméticas

Mnemônico e operandos	tamanho **em byte**	Código **em hex**	Tempo Ciclo maq.
ADD A,R0	1	28	1
ADD A,R1	1	29	1
ADD A,R2	1	2A	1
ADD A,R3	1	2B	1
ADD A,R4	1	2C	1
ADD A,R5	1	2D	1
ADD A,R6	1	2E	1
ADD A,R7	1	2F	1
ADD A, direto	2	25	1
ADD A, @R0	1	26	1
ADD A, @R1	1	27	1
ADD A, # dado	2	24	1
ADDC A, R0	1	38	1
ADDC A, R1	1	39	1
ADDC A, R2	1	3A	1
ADDC A, R3	1	3B	1
ADDC A, R4	1	3C	1
ADDC A, R5	1	3D	1
ADDC A, R6	1	3E	1
ADDC A, R7	1	3F	1
ADDC A, direto	2	35	1
ADDC A, @R0	1	36	1
ADDC A, @R1	1	37	1
ADDC A, # dado	2	34	1
SUBB A, R0	1	98	1
SUBB A, R1	1	99	1
SUBB A, R2	1	9A	1
SUBB A, R3	1	9B	1
SUBB A, R4	1	9C	1
SUBB A, R5	1	9D	1
SUBB A, R6	1	9E	1
SUBB A, R7	1	9F	1
SUBB A, direto	2	95	1
SUBB A, @R0	1	96	1
SUBB A, @R1	1	97	1
SUBB A, # dado	2	94	1
INC A	1	04	1
INC R0	1	08	1
INC R1	1	09	1
INC R2	1	0A	1
INC R3	1	0B	1
INC R4	1	0C	1
INC R5	1	0D	1
INC R6	1	0E	1
INC R7	1	0F	1
INC direto	2	05	1
INC @R0	1	06	1
INC @R1	1	07	1
DEC A	1	14	1
DEC R0	1	18	1
DEC R1	1	19	1
DEC R2	1	1A	1
DEC R3	1	1B	1
DEC R4	1	1C	1
DEC R5	1	1D	1
DEC R6	1	1E	1
DEC R7	1	1F	1
DEC direto	2	15	1
DEC @R0	1	16	1
DEC @R1	1	17	1
INC DPTR	1	A3	2
MUL AB	1	A4	4
DIV AB	1	84	4
DA A	1	D4	1

Operações Lógicas

Mnemônico e operandos	tamanho **em byte**	Código **em hex**	Tempo Ciclo maq.
ANL A, R0	1	58	1
ANL A, R1	1	59	1
ANL A, R2	1	5A	1
ANL A, R3	1	5B	1
ANL A, R4	1	5C	1
ANL A, R5	1	5D	1
ANL A, R6	1	5E	1
ANL A, R7	1	5F	1
ANL A, direto	2	55	1
ANL A, @R0	1	56	1
ANL A, @R1	1	57	1
ANL A, #dado	2	54	1
ANL direto, A	2	52	1
ANL direto,#dado	3	53	2
ORL A, R0	1	48	1
ORL A, R1	1	49	1
ORL A, R2	1	4A	1
ORL A, R3	1	4B	1
ORL A, R4	1	4C	1
ORL A, R5	1	4D	1
ORL A, R6	1	4E	1
ORL A, R7	1	4F	1
ORL A, direto	2	45	1
ORL A, @R0	1	46	1
ORL A, @R1	1	47	1
ORL A, #dado	2	44	1
ORL direto, A	2	42	1
ORL direto,#dado	3	43	2
XRL A, R0	1	68	1
XRL A, R1	1	69	1
XRL A, R2	1	6A	1
XRL A, R3	1	6B	1
XRL A, R4	1	6C	1
XRL A, R5	1	6D	1
XRL A, R6	1	6E	1

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

XRL A, R7	1	6F	1
XRL A, direto	2	65	1
XRL A, @R0	1	66	1
XRL A, @R1	1	67	1
XRL A, #dado	2	64	1
XRL direto, A	2	62	1
XRL direto,#dado	3	63	2
CLR A	1	E4	1
CPL A	1	F4	1
RL A	1	23	1
RLC A	1	33	1
RR A	1	03	1
RRC A	1	13	1
SWAP A	1	C4	1

Transferência de Dados

Mnemônico e operandos	tamanho **em byte**	Código **em hex**	Tempo Ciclo maq.
MOV A, R0	1	E8	1
MOV A, R1	1	E9	1
MOV A, R2	1	EA	1
MOV A, R3	1	EB	1
MOV A, R4	1	EC	1
MOV A, R5	1	ED	1
MOV A, R6	1	EE	1
MOV A, R7	1	EF	1
MOV A, direto	2	E5	1
MOV A, @R0	1	E6	1
MOV A, @R1	1	E7	1
MOV A, #dado	2	74	1
MOV R0, A	1	F8	1
MOV R1, A	1	F9	1
MOV R2, A	1	FA	1
MOV R3, A	1	FB	1
MOV R4, A	1	FC	1
MOV R5, A	1	FD	1
MOV R6, A	1	FE	1
MOV R7, A	1	FF	1
MOV R0, direto	2	A8	2
MOV R1, direto	2	A9	2
MOV R2, direto	2	AA	2
MOV R3, direto	2	AB	2
MOV R4, direto	2	AC	2
MOV R5, direto	2	AD	2
MOV R6, direto	2	AE	2
MOV R7, direto	2	AF	2
MOV R0, #dado	2	78	1
MOV R1, #dado	2	79	1
MOV R2, #dado	2	7A	1
MOV R3, #dado	2	7B	1
MOV R4, #dado	2	7C	1
MOV R5, #dado	2	7D	1
MOV R6, #dado	2	7E	1
MOV R7, #dado	2	7F	1
MOV direto, A	2	F5	1
MOV direto, R0	2	88	2
MOV direto, R1	2	89	2
MOV direto, R2	2	8A	2
MOV direto, R3	2	8B	2
MOV direto, R4	2	8C	2
MOV direto, R5	2	8D	2
MOV direto, R6	2	8E	2
MOV direto, R7	2	8F	2
MOV direto1,direto2	3	85	2
MOV direto, @R0	2	86	2
MOV direto, @R1	2	87	2
MOV direto, #dado	3	75	2
MOV @R0, A	1	F6	1
MOV @Ri1 A	1	F7	1
MOV @R0, direto	2	A6	2
MOV @R1, direto	2	A7	2
MOV @R0, dado	2	76	1
MOV @R1, dado	2	77	1
MOV DPTR,dado16	3	90	2
MOVC A,@A+DPTR	1	93	2
MOVC A, @A+PC	1	83	2
MOVX A, @R0	1	E2	2
MOVX A, @R1	1	E3	2
MOVX A, @DPTR	1	E0	2
MOVX @R0, A	1	F2	2
MOVX @R1, A	1	F3	2
MOVX @DPTR, A	1	F0	2
PUSH direto	2	C0	2
POP direto	2	D0	2
XCH A, R0	1	C8	1
XCH A, R1	1	C9	1
XCH A, R2	1	CA	1
XCH A, R3	1	CB	1
XCH A, R4	1	CC	1
XCH A, R5	1	CD	1
XCH A, R6	1	CE	1
XCH A, R7	1	CF	1
XCH A, direto	2	C5	1
XCH A, @R0	1	C6	1
XCH A, @R1	1	C7	1
XCHD A, @R0	1	D6	1
XCHD A, @R1	1	D7	1

Variáveis Booleanas Bit

Mnemônico e operandos	tamanho **em byte**	Código **em hex**	Tempo Ciclo maq.
CLR C	1	C3	1
CLR Bit	2	C2	1
SETB C	1	D3	1
SETB Bit	2	D2	1
CPL C	1	B3	1
CPL Bit	2	B2	1
ANL C, Bit	2	82	2
ANL C, \ Bit	2	B0	2
ORL C, Bit	2	72	2
ORL C, \ Bit	2	A0	2
MOV C, Bit	2	A2	1
MOV Bit, C	2	92	2

Fluxo de Execução

Mnemônico e operandos	tamanho **em byte**	Código **em hex**	Tempo Ciclo maq.
ACALL End 11	2	11	2
LCALL End 16	3	12	2
RET	1	22	2
RETI	1	32	2
AJMP End 11	2	01	2
LJMP End 16	3	02	2
SJMP rel	2	80	2
JMP @A+DPTR	1	73	2
JZ rel	2	60	2
JNZ rel	2	70	2
JC rel	2	40	2
JNC rel	2	50	2
JB bit, rel	3	20	2
JNB bit, rel	3	30	2
JBC bit, rel	3	10	2
CJNE A,direto,rel	3	B5	2
CJNE A,#dado,rel	3	B4	2
CJNE R0,dado,rel	3	B8	2
CJNE R1,dado,rel	3	B9	2
CJNE R2,dado,rel	3	BA	2
CJNE R3,dado,rel	3	BB	2
CJNE R4,dado,rel	3	BC	2
CJNE R5,dado,rel	3	BD	2
CJNE R6,dado,rel	3	BE	2
CJNE R7,dado,rel	3	BF	2
CJNE @R0,#dado,rel	3	B6	2
CJNE @R1,#dado,rel	3	B7	2
DJNZ R0, rel	2	D8	2
DJNZ R1, rel	2	D9	2
DJNZ R2, rel	2	DA	2
DJNZ R3, rel	2	DB	2
DJNZ R4, rel	2	DC	2
DJNZ R5, rel	2	DD	2
DJNZ R6, rel	2	DE	2
DJNZ R7, rel	2	DF	2
DJNZ direto, rel	3	D5	2
NOP	1	00	1

ALGORÍTMOS

Algorítmo é a descrição de um conjunto de comandos que resultam numa sucessão de ações. Geralmente, um algoritmo se destina a resolver uma ação, fixar um padrão de comportamento a ser seguido ou uma norma de execução a ser trilhada para se atingir um resultado final ,a solução de um problema.

Exemplo:

Algoritmo do “programa soma” “programa soma” em assembly (8051)

A←10	MOV	A,#0AH
B←05	MOV	B,#05H
R0←A+B	ADD	AB
	MOV	R0,A

FLUXOGRAMA

O fluxograma é uma representação gráfica das tarefas de um programa, por meio de símbolos que fornecem a seqüência de um processo. O fluxograma ajuda a organizar o programa, permitindo o seu delineamento e possibilitando seguir passo- a- passo o que será executado nele.

Existem estruturas, definições rigorosas das regras de um fluxograma e uma variedade de símbolos para a construção do mesmo, o que é interessante seguir, principalmente quando se trata de um projeto complexo onde se encontram várias pessoas no mesmo projeto, programando partes diferentes do fluxograma. Com um fluxograma bem estruturado fica fácil corrigir eventuais erros que possam surgir no decorrer do projeto.

Para o programador de microcontrolador é preferível que ele estruture o seu fluxograma, podendo assim criar um estilo individual, pois não convém determinar princípios fixos ou normas de funcionamento, mesmo porque os pro-

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001

gramas para os microcontroladores são muito pessoais. Programas diferentes podem chegar ao mesmos resultados, e o fluxograma depende do sistema, do tipo de equipamento para qual será feito o programa, do tipo da linguagem, etc.

O fluxograma não é um elemento indispensável ao desenvolvimento de um programa, porém, sem ele se torna mais difícil qualquer alteração ou correção. O fluxograma deve ser feito com a visualização do problema a ser resolvido passo-a-passo, e o programa deve ser feito por partes, e testado a cada parte para que fique mais simples detectar e solucionar qualquer tipo de erro. As alterações devem ser bem analisadas porque poderão influenciar em outras partes do programa.

Essencialmente, o projeto de um programa a ser processado por sistema deve ter os seguintes procedimentos:

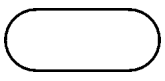
- análise do problema
- elaboração do fluxograma
- escrever o programa em linguagem simbólica
- traduzir o programa para linguagem de máquina
- testar e, se necessário corrigir,

Na análise do problema devem ser determinados de maneira bem clara quais os objetivos a ser alcançados, exatamente que tarefa deve ser realizada e definir todo o hardware do sistema. O fluxograma deve ser dividido em tarefas e a definição da sequência que estas tarefas deve-se executadas incluindo-se decisões.

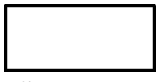
Para escrever o programa deve ser primeiramente determinar que tipo de linguagem será utilizada, se uma linguagem de baixo nível ou alto nível.

A tradução do programa para linguagem de máquina é feita normalmente por um programa copilador já instalado no PC e depende da linguagem usada para escrever o programa. O teste pode ser feito através de um programa simulador ou diretamente no sistema para qual o programa esta sendo escrito.

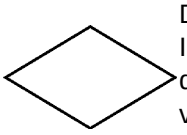
Alguns símbolos utilizado em fluxogramas são apresentados a seguir.



TERMINAL
Ponto de início, término ou interrupção de um programa.

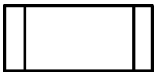


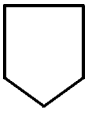
PROCESSAMENTO
Um grupo de instruções que executam uma função de processamento do programa.



DECISÃO
Indica a possibilidade de desvios para diversos pontos do programa

PROCESSO PRÉ-DEFINIDO
Indica uma rotina que será executada fora do programa principal (sub-rotinas)



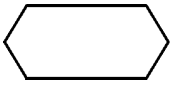


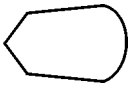
CONECTOR FORA DE PÁGINA
Uma entrada ou saída de uma página para outra página do diagrama

FLUXO
Indica a direção do fluxo de dados ou de um processamento

CONEXÃO
Indica a rota de prosseguimento do fluxograma

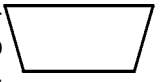
SUB-ROTINA
Um grupo de operações separadas do fluxo do programa





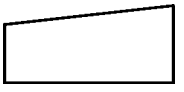
VISOR
Saída de informações através de vídeo ou display

ENTRADA / SAÍDA
Qualquer função relacionada com dispositivo de entrada ou saída genéricos



MODIFICAÇÃO DE PROGRAMA
Qualquer função que altera o próprio programa

TECLADO
Entrada de informações através de teclado.



■

SABER ELETRÔNICA ESPECIAL Nº 2 /JANEIRO/2001