

Escola de Engenharia Elétrica, Mecânica e de Computação
Universidade Federal de Goiás



NOTAS DE AULA
MICROPROCESSADORES E
MICROCONTROLADORES

Prof. Dr. José Wilson Lima Nerys

“A vida é a soma de todas as suas escolhas” – Albert Camus

SUMÁRIO

1	<i>Introdução a Microprocessadores</i>	5
1.1	Conceitos Iniciais.....	5
1.2	Um pouco de História	8
1.3	Sistema de Numeração	14
2	<i>Arquitetura e Princípio de Funcionamento de Microprocessadores</i>	19
2.1	Arquiteturas RISC, CISC e Híbrida	19
2.2	Arquitetura 8085, 8088/86, 8051 e PIC.....	21
2.3	Registradores Principais 8085, 8088/8086, 8051	24
2.4	Princípio de Funcionamento 8085, 8088/86, 8051	27
2.5	Formato das Instruções.....	32
2.6	Modos de Endereçamento e Grupos de Instruções.....	35
2.7	Registradores de Flags do 8085, 8088/86 e 8051.....	36
2.8	Funcionamento da Pilha no 8085, 8051 e 8088/86.....	40
3	<i>Microcontrolador 8051</i>	43
3.1	Memórias ROM e RAM.....	45
3.2	Os Registradores de Funções Especiais.....	47
3.3	Instruções Gerais do Microcontrolador 8051	48
3.4	Instruções de Comparação, Decisão e de Desvio.....	50
3.5	Operações com bit	51
3.6	Diretivas de programação.....	51
3.7	Programas Exemplos	52
4	<i>Interrupções</i>	56
4.1	Princípio de Funcionamento e Habilitação	56
4.2	Endereços Desvio das Interrupções	57
4.3	Programas Exemplos com Interrupção.....	58
5	<i>Temporizadores e Contadores do 8051</i>	61
5.1	Princípio de Funcionamento e Modos de Operação	61
5.2	Programas Exemplos usando Temporizadores.....	64
6	<i>Comunicação Serial</i>	72
6.1	Noções Básicas de Comunicação Serial	72
6.2	Roteiros de Programas usando Comunicação Serial	76
7	<i>Expansão e Mapeamento de Memória</i>	81
7.1	Expansão de Memória.....	82
7.2	Mapeamento de Memória	83

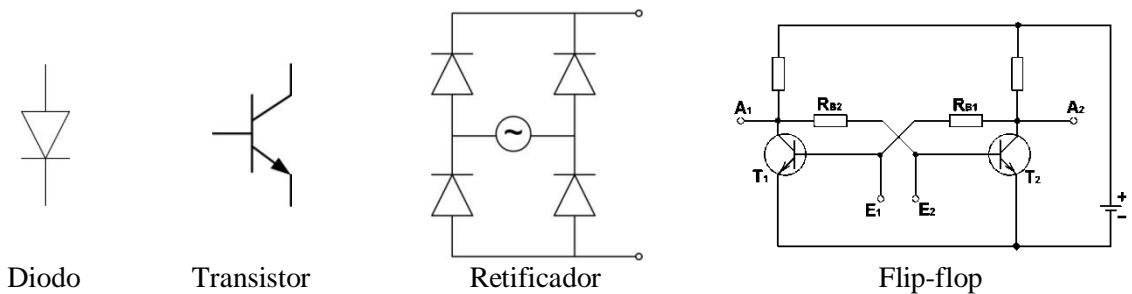
8	<i>Dispositivos para Entrada, Saída e Acionamentos Elétricos</i>	87
8.1	Teclado	87
8.2	Display de 7-Segmentos	91
8.3	Display LCD	94
8.4	Sensores de Presença	100
8.5	Medição de Velocidade	101
8.6	Motor de Corrente Contínua	102
8.7	Motor de Passo	106
8.8	Lâmpada Incandescente	109
9	<i>Bibliografia</i>	110

1 Introdução a Microprocessadores

1.1 Conceitos Iniciais

O presente material tem o intuito de apresentar uma introdução ao estudo de microprocessadores e ao desenvolvimento de projetos com microcontroladores da família 8051. Para iniciar, portanto, é fundamental uma introdução aos conceitos básicos associados a microprocessadores e um pouco de história desse componente, que está cada vez mais presente nas nossas vidas. Está presente, por exemplo, no computador, no aparelho de DVD, no forno micro-ondas, no carro, no telefone celular, em sistemas de alarme, em sistemas de controle de acesso, dentre outros.

O estudo de microprocessadores tem início com o estudo do princípio de funcionamento de suas unidades internas, vistas isoladamente em disciplinas como Materiais Elétricos, Sistemas Digitais e Eletrônica. Nestas disciplinas estuda-se, por exemplo, as junções PN, NPN e PNP e os componentes básicos a partir dessas junções, tais como diodos e transistores. Estudam-se ainda configurações diversas a partir desses componentes, tais como retificadores, amplificadores e flip-flops, Fig. 1.1.



O flip-flop, em especial, pelo princípio de funcionamento e os dois estados na saída (baixo/alto) é usado para compor 1 bit, que é a unidade básica de todos os componentes de um microprocessador e na construção de um tipo de memória chamada de memória estática. Dentre os componentes internos de um microprocessador destaca-se: registradores, contadores, somadores, codificadores e decodificadores. Todos eles encapsulados em uma única pastilha, e adequadamente conectados e sincronizados no funcionamento, integram um microprocessador, Fig. 1.2.

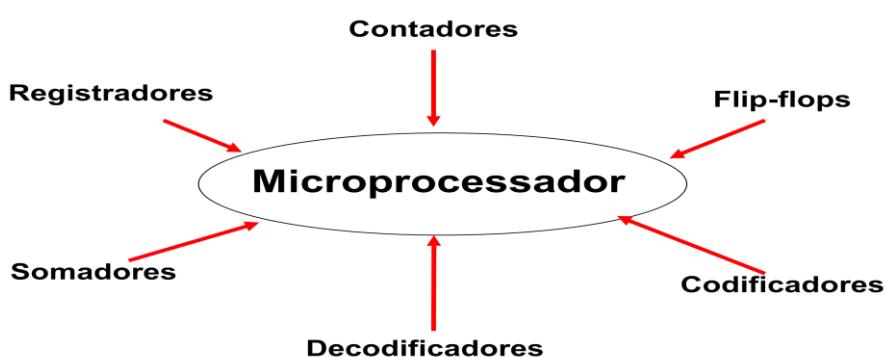


Fig. 1.2: Componentes de um microprocessador

A Fig. 1.3 mostra uma forma mais compacta de um microprocessador, com suas unidades básicas: registradores, unidade lógica e aritmética e unidade de controle. Estas três unidades, em uma única pastilha, compõem o que se denomina de unidade central de processamento (CPU) de um computador. Em resumo, pode-se dizer que é o coração de um computador. A CPU é responsável pela busca e execução de programas na memória e pelo controle de todas as unidades de um computador.

Os registradores são usados na movimentação interna de dados de um microprocessador. A quantidade de registradores disponíveis é fundamental para o desempenho de um microprocessador, uma vez que o acesso à memória, que é externa à pastilha, é mais lento. O principal registrador do microprocessador é denominado de acumulador e é utilizado na maioria das instruções.

A unidade lógica e aritmética (ALU ou ULA) realiza funções básicas de processamento de dados, tais como adição, subtração e operações lógicas. A unidade de controle tem como função o controle do funcionamento de todas as unidades.

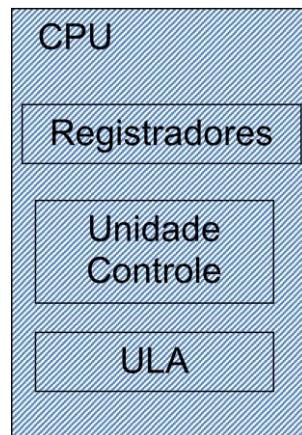


Fig. 1.3 – Unidade Central de Processamento ou Microprocessador

Além da CPU um computador é composto de memória e unidades de entrada/saída. As unidades de entrada/saída permitem a comunicação do microprocessador com o meio externo, através de periféricos tais como mouse, impressora, monitor, scanner e outros.

De um modo geral há dois tipos de memória, a memória de programa (ROM – *Read Only Memory*, ou Memória somente de leitura) e a memória de dados (RAM – *Random Access Memory*, ou Memória de acesso aleatório). A memória ROM, como o próprio nome diz, é uma memória somente de leitura. As informações são previamente gravadas pelo fabricante e não pode ser alteradas. A memória RAM permite a gravação e a leitura de dados durante o funcionamento do sistema. Os dados são perdidos quando há falta de energia.

Foi dito que a memória de programa ROM vem gravada de fábrica com as informações necessárias para o funcionamento do computador e que não podem apagadas. Essa é a ROM básica. No entanto, há outros tipos de ROM. A PROM (*Programmable Read-Only Memory*) é equivalente à ROM básica; ela pode ser gravada pelo usuário, mas não pode ser apagada. A EPROM (*Erasable Programmable Read-Only Memory*) pode ser apagada com luz ultravioleta, e posteriormente reutilizada. A EEPROM (*Electrically Erasable Programmable Read-Only Memory*) pode ser apaga eletricamente e programada novamente.

Quanto à memória RAM, há dois tipos básicos: RAM estática (SRAM) e RAM dinâmica (DRAM). A primeira é construída com flip-flops e, por isso, é de alta densidade. Comparada com a RAM dinâmica, ela ocupa muito mais espaço na pastilha. No entanto, ela tem a vantagem de ser bem mais rápida que a memória dinâmica. A memória dinâmica é construída com capacitores e necessita de um circuito de atualização periódica dos dados (*refresh*). Assim, a memória principal de um computador é construída com RAM dinâmica, pelo pouco espaço necessário.

Com o passar dos anos os processadores tornaram-se cada vez mais rápidos, o mesmo não acontecendo com as pastilhas de memória, que evoluíram de forma bem menos acentuada (em particular, a memória dinâmica, que possui velocidade de acesso bem menor que a estática, mas é bem mais barata). Para evitar com que a baixa velocidade de acesso da memória compromettesse o desempenho dos processadores mais modernos, um tipo especial de memória RAM foi criado: a memória CACHE.

A memória cache consiste numa pequena quantidade de memória RAM estática (SRAM) usada para acelerar o acesso à RAM dinâmica. Quando há necessidade de ler dados da memória dinâmica, estes são antes transferidos para a memória cache. Enquanto o processador lê dados da memória cache, mais dados são antecipadamente transferidos da memória dinâmica para a memória cache, de forma que o processamento torna-se mais rápido.

Cada endereço da memória permite o acesso a um conjunto de dados de 8 bits, ou 1 byte. O bit (binary digit) pode assumir valor 0 (nível lógico baixo) ou 1 (nível lógico alto). A capacidade da memória, ou quantidade de bytes que ela pode armazenar, é normalmente dada em kbytes, Mbytes e Gbytes. Ao contrário de outras grandezas o fator de multiplicação é 1024 (2^{10}), como mostrado a seguir:

$$1 \text{ kbyte} = 2^{10} = 1.024 \text{ bytes};$$

$$1 \text{ Mbyte} = 2^{10} \times 2^{10} = 2^{20} = 1.048.576 \text{ bytes} = 1.024 \text{ kbytes};$$

$$1 \text{ Gbyte} = 2^{10} \times 2^{10} \times 2^{10} = 2^{30} = 1.073.741.824 \text{ bytes} = 1.024 \text{ Mbytes}$$

1 byte							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Nibble Superior				Nibble Inferior			

16 bits (2 bytes) formam 1 word e 2 words formam 1 word dupla (Double Word) ou Dword.

1 word															
Bit15	Bit14	Bit13	Bit12	B11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte superior (High Byte)								Byte Inferior (Low Byte)							

Outros conceitos utilizados no estudo de microprocessadores:

MIPS – *Millions of Instructions Per Seconds* (Milhões de Instruções Por Segundo): É uma unidade de desempenho do microprocessador.

FLOPS – *FLOATing point instructions Per Seconds* (Instruções com Ponto Flutuante Por Segundo). É também uma unidade de desempenho do microprocessador. Indica a capacidade de trabalhar com números decimais.

Representação em Ponto Fixo – Sistema numérico no qual o ponto está implicitamente fixo (à direita do dígito mais à direita);

Representação em Ponto Flutuante – Sistema numérico no qual um número real é representado por um par distinto de numerais: uma mantissa (ou significante) e um expoente. Possibilita representação de números fracionários.

Set de instruções – Conjunto de Instruções. Conjunto de Mnemônicos (siglas que fazem lembrar uma ação) que representam todas as instruções do processador. Cada processador possui o seu set de instruções particular.

CISC – *Complex Instruction Set Computer* (Computador com Conjunto Complexo de Instruções): Tipo de arquitetura de microprocessadores onde o barramento de comunicação entre as unidades que compõem a CPU é comum a todas as unidades, ou seja, não há comunicação direta entre unidades, através de um barramento exclusivo.

RISC – *Reduced Instruction Set Computer* (Computador com Conjunto Reduzido de Instruções): Tipo de arquitetura de microprocessadores cujas principais características são:

- Conjunto de instruções limitado e simples;
- Grande número de registradores de propósito geral;
- Pipeline otimizado. Em outras palavras, há comunicação direta entre algumas unidades, através de barramento exclusivo, possibilitando, assim, o processamento paralelo de instruções.

BIOS – Basic Input/Output System – É o conjunto mínimo de instruções necessárias para a inicialização do computador. Também gerencia o fluxo de dados entre o sistema operacional do computador e os dispositivos periféricos conectados.

Desempenho de microprocessadores – O desempenho de processadores, ou velocidade de processamento, depende de alguns pontos chaves. O aumento de desempenho pode ser obtido através de:

- Aumento de clock

O sinal de clock é responsável pelo sincronismo entre as unidades de processamento internas ao microprocessador e pelas unidades externas. Quanto maior a frequência de clock mais rápido o processamento. No entanto, não se pode aumentar de forma indefinida essa frequência. Isso pode causar falhas de processamento e sobreaquecimento. O aumento depende de pesquisas com o objetivo de reduzir o tamanho dos componentes básicos do microprocessador e aumento da quantidade de componentes, sem perda de estabilidade no funcionamento.

- Aumento do número interno de bits

Uma maior quantidade de bits dos registradores e dos barramentos internos permite a movimentação de uma maior quantidade de dados por unidade de tempo, aumentando o desempenho do microprocessador.

- Aumento do número externo de bits

Um número maior de bits externos permite a movimentação de uma maior quantidade de dados por unidade de tempo com os periféricos, tais como memória, unidade de entrada e saída, controlador de acesso direto à memória (DMA).

- Redução do número de ciclos para executar cada instrução

A execução de uma instrução normalmente é feita em duas etapas: busca (onde a instrução é transferida da memória para a unidade de decodificação) e execução (onde os sinais de controle ativam, em uma sequência lógica, todas as unidades envolvidas na execução). No microprocessador 8085 as instruções mais rápidas são executadas em quatro ciclos de clock; as mais lentas, em até 16 ciclos de clock. A redução do número de ciclos de clock na execução de uma instrução torna o processamento mais rápido.

- Aumento da capacidade e velocidade da memória cache

Como já foi dito anteriormente, ao longo dos anos, o aumento de velocidade de processamento dos microprocessadores tem sido muito maior do que o aumento da velocidade de acesso à memória principal. Assim, a velocidade de acesso à memória principal torna-se um limitador de desempenho dos processadores. Em razão desse problema foi criada a memória cache. A memória cache (constituída de memória RAM estática) é usada para acelerar a transferência de dados entre a CPU e a memória principal (constituída de RAM dinâmica, de menor volume, porém mais lenta). O aumento da capacidade e da velocidade da memória cache resulta no aumento da velocidade de transferência de dados entre a CPU e a memória principal e, consequentemente, resulta no aumento do desempenho global do sistema.

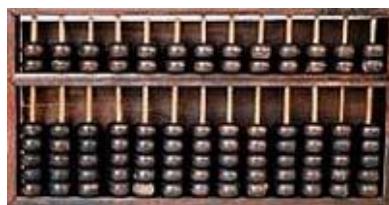
- Execução de instruções em paralelo

O microprocessador 8085 compartilha um barramento comum entre suas unidades internas e seus periféricos, o que significa dizer que não permite a execução simultânea de duas operações que utilizem o barramento. Assim, apenas uma instrução é executada por vez. Uma arquitetura que permita que duas ou mais operações sejam executadas simultaneamente torna o processamento mais rápido.

1.2 Um pouco de História

4000 A.C – ÁBACO – Invenção do *ábaco* pelos babilônios. Instrumento usado para realizar operações aritméticas, onde cada coluna representa uma casa decimal. Era o principal instrumento de cálculo do século XVII e é usado até hoje. Data da mesma época do ábaco o octograma chinês *Yin Yang*, o qual é tido como a primeira representação binária dos números de 0 a 7. Foi criado pelo imperador

chinês *Fou-Hi* para representar a interação entre as duas energias que juntas são o fundamento da “totalidade”.



Ábaco



Octograma chinês Yin Yang

Fig. 1.4 – Ábaco e Octograma

1614 – LOGARITMO – O cientista escocês **JOHN NAPIER** criou os *logaritmos*. Através das tabelas criadas, as operações de multiplicação e divisão tornaram-se mais simples, pois eram substituídas por operações de adição e subtração, reduzindo o tempo de processamento.

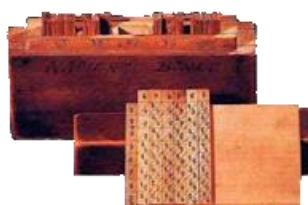
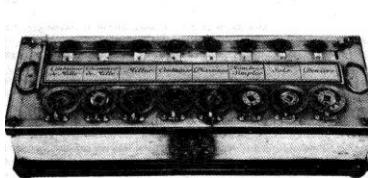
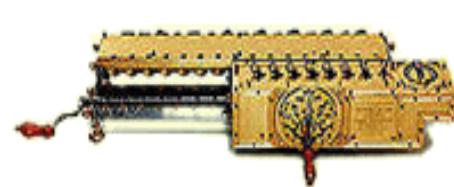


Tabela de logaritmos



Pascaline



Calculadora de 4 funções de Leibniz

Fig. 1.5 – Tabela de logaritmos, Pascaline e Calculadora de Leibniz

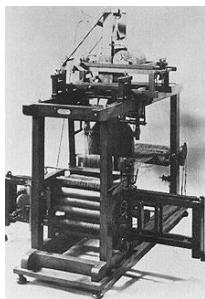
1623 – RELÓGIO DE CALCULAR – **WILHELM SHICKARD**, professor de matemática da Universidade de Tübingen, Alemanha, inventou um relógio de calcular que é considerado a primeira máquina mecânica de calcular da história. Fazia multiplicação e divisão, mas requeria várias intervenções do operador. Usava o princípio desenvolvido por Napier (“Napier’s bones”). Essa calculadora foi desenvolvida para auxiliar o matemático e astrônomo Johannes Kepler.

1642 – PASCALINE – O cientista francês **BLAISE PASCAL** criou uma calculadora capaz de realizar operações de adição e subtração. A máquina implementada utilizava rodas e engrenagens, com as quais era possível representar números decimais de 0 a 9. Pascal desenvolveu essa máquina para ajudar seu pai na coleta de impostos. A máquina teve mais de 50 versões diferentes em uma década.

1671 – O matemático alemão **GOTTFRIED LEIBNIZ** criou uma calculadora de 4 funções, capaz de realizar operações de adição, subtração, multiplicação e divisão. É a antecessora das calculadoras atuais. O problema comum às calculadoras até esta época era a necessidade de entrar com todos os resultados intermediários.

1738 – ANDROIDES PROGRAMÁVEIS – O cientista francês **JACQUES VAUCANSON** criou robôs (imitando a aparência humana). Eram capazes de tocar flautas. Sua criação mais famosa foi “O Pato”. Esse pato mecânico era capaz de imitar todos os movimentos de um pato real (bater asas, movimentar a cabeça, fazer barulho equivalente, comer e evacuar. Em **1749** ele construiu o primeiro **TEAR AUTOMÁTICO**, que aceitava comandos através de um cilindro de metal perfurado.

1801 – CARTÃO PERFORADO – O Tecelão francês **JOSEPH MARIE JACQUARD** aperfeiçoou o tear construído por **Vaucanson**. Ele construiu uma máquina de tear que memorizava em cartões perfurados os padrões de desenho dos tecidos e depois os reproduzia com fidelidade, lendo comandos na presença ou ausência de orifícios. A versão seguinte do Tear, em **1804**, era totalmente automatizada e podia fazer desenhos muito complicados. Esse é considerado o primeiro registro de programação semelhante à de computadores modernos.



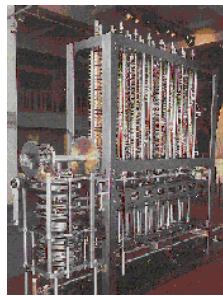
Tear de Vaucanson



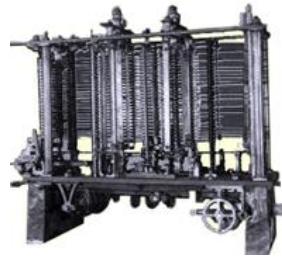
Tear de Jacquard

Fig. 1.6 – Tear de Vaucanson e Tear de Jacquard

1822 – MÁQUINA DE DIFERENÇAS e MÁQUINA ANALÍTICA – Aborrecido pelos inúmeros e frequentes erros que encontrava nas tabelas de *logaritmos*, o professor de matemática **CHARLES BABBAGE** (inglês) decidiu construir uma máquina que eliminasse o trabalho repetitivo de fazer esses cálculos, a "*Máquina de Diferenças*". O modelo apresentado em **1822** encantou o Governo Britânico que decidiu financiá-lo na construção de uma máquina de diferenças completa, movida a vapor e completamente automática, comandada por um programa de instrução fixo capaz de imprimir as tabelas. Baseada em operações de adição e subtração e na técnica de diferenças finitas, era capaz de resolver funções polinomiais e trigonométricas (cálculo de tabelas de navegação).



Máquina de diferenças



Máquina analítica

Fig. 1.7 – Máquina de diferenças e analítica

O projeto da sua nova máquina levou 10 anos e foi abandonada em **1833**, quando decidiu criar a "*Máquina Analítica*", um computador mecânico-automático totalmente programável, função que designou para a condessa *Ada Lovelace* (filha de **Lord Byron**). O novo computador decimal paralelo a vapor operaria números de 50 dígitos e faria uso de uma memória de 1000 números, usando cartões perfurados e condicionais (IF), além de instruções de desvio. Apesar de ter uma estrutura correta, a metalurgia da época não permitia a simetria e resistência das peças, razão ao qual a máquina nunca funcionou. Seria capaz de fazer uma adição em 1 segundo e uma multiplicação em 1 minuto.

1885 - O CARTÃO DE HOLLERITH – **HERMAN HOLLERITH**, funcionário do Departamento de Estatística dos Estados Unidos, construiu uma máquina de cartão perfurado para fazer o recenseamento da população americana. Antes da máquina o recenseamento durava 7 anos e ocupava 500 empregados. Com a máquina o recenseamento de 1890 durou 1 ano e ocupou 43 empregados. A máquina foi aproveitada nas mais diversas aplicações em repartições públicas, comércio e indústria, e aperfeiçoada para realizar operações aritméticas elementares. Em 1896 Hollerith fundou a TMC (*Tabulation Machine Company*). Para ampliar seus negócios, a TMC se uniu a duas pequenas empresas para formar a **CTRC** (*Computing Tabulation Recording Company*), em 1914. Em 1924, a CTRC se tornou uma empresa internacional e mudou seu nome para **IBM** (*International Business Machine*).

1936 – COMPUTADORES Z1, Z3 e Z4 – O cientista alemão **KONRAD ZUSE** criou o computador - Z1, baseado em relé eletro-mecânico. Criou também o computador Z3, que foi o primeiro computador de propósito geral controlado por programa. Criou ainda o Z4, computador projetado para o desenvolvimento de mísseis. Ele foi destruído por bomba na 2^a guerra mundial.



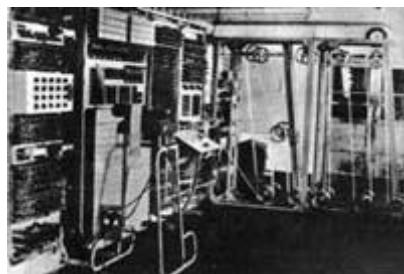
Máquina de Herman Hollerith



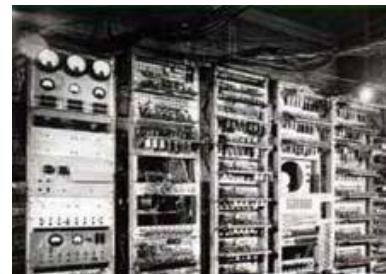
Computador Z4

Fig. 1.8 – Máquina de Herman Hollerith e Computador Z4

1943 – COLOSSO – Na Inglaterra, em 1943, **ALAN TURING**, do Serviço de Inteligência Britânico, construiu o **Colosso**, de dimensões gigantescas. A máquina, abrigada em **Bletchley Park**, tinha 2000 válvulas e lia símbolos perfurados numa argola de fita de papel, inserida na máquina de leitura fotoelétrica, comparando a mensagem codificada com sequências conhecidas até encontrar uma coincidência. Processava cerca de 5 mil caracteres por segundo e foi usada para descodificar as mensagens dos alemães, tendo sido decisiva no resultado final da guerra.



Colosso



Mark I

Fig. 1.9 – Computadores Colosso e Mark I

1944 – MARK I – Na Universidade de Harvard em 1937, o professor **Howard Aiken**, financiado pela IBM, começou a construir o **Mark I**, concluído em 1944. Baseado em um sistema decimal, manipulava números de até 23 dígitos e tinha medidas grotescas: 15 m de comprimento e 2,5 m de altura; 760.000 peças envoltas em vidro e aço inoxidável brilhante; 800 km de fios e 420 interruptores para controle. Trabalhava sob o controle de um programa perfurado em uma fita de papel. Adição e subtração em 0,3 s, multiplicação em 3 s e divisão em 12 s,

1946 – ENIAC – (*Electronic Numerical Integrator and Computer*) - 1º Computador de propósito geral a válvula: 18.000 válvulas, 30 toneladas, 15.000 pés quadrados, 140 kW, representação e aritmética com números decimais, 5.000 adições/seg. Projetado pela *Ballistics Research Labs*. Foi aproveitado no desenvolvimento da Bomba “H”.

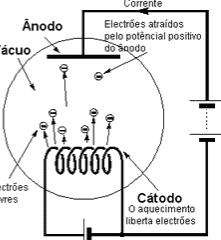
1946 – VON NEWMANN MACHINE – A Máquina de Von Newman, ou “Máquina de Touring” introduziu o conceito de programa armazenado (Stored Program Concept) no qual a memória conteria, além de dados, programas. Os computadores modernos são baseados na máquina de Von Newman.

1947 – TRANSISTOR – Invenção do transistor pelos cientistas John Bardeen, William Shockley e Walter Brattain. Passou a ser usado em escala comercial somente em 1952 pela *Bell Laboratories*.

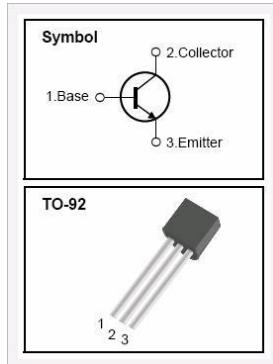
1950 – UNIVAC – (Universal Automatic Computer) – Lançado pela SPERRY, foi o 1º Computador de aplicação científica e comercial. Seguiram **UNIVAC II** e **UNIVAC 1100 series**.

1953 – IBM 701 – Computador desenvolvido para aplicações científicas.

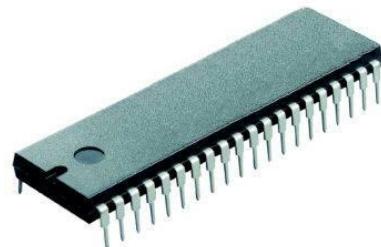
1958 – CIRCUITO INTEGRADO – O engenheiro Jack Kilby, da Texas Instruments, criou o Circuito Integrado.



Válvula



Transistor



Círculo Integrado

Fig. 1.10 – Válvula, transistor e círculo integrado

1960 – IBM 7090, 7094 – Computador transistorizado. Utilização de linguagens de programação de alto nível, tais como FORTRAN, COBOL e PASCAL.

1964 – IBM 360 – Primeira família planejada de computadores.

DEC PDP 8 – Introduziu o conceito de Minicomputador. Criou a estrutura de barramento, ou seja, unidade de Entrada e Saída, Memória e CPU interligados por um conjunto de condutores.

1970 (início da década) – **CP 1600 (General Instruments)** – Microprocessador de 16 bits criado pela General Instruments.

1971 – 4004 (INTEL) - 1º microprocessador a ser lançado, de 4 bits, com aplicação voltada para calculadoras (manipulação de números em BCD) - 45 instruções - 640 Bytes de memória - clock de 108 kHz - 60.000 instruções/seg. (OBS: desempenho superior ao ENIAC) - 2.300 transistores (tamanho do transistor: 10 µm).

1972 – 8008 (INTEL) - 1º microprocessador de 8 bits, com aplicação voltada para terminais (que trabalham com caracteres - codificação ASCII) - 48 instruções - 16KB de memória - clock de 200 KHz - 300.000 instruções/seg. 3500 transistores.

1974 – 8080 (INTEL) - Processador de 8 bits, de propósito geral - 72 instruções - opera com 12V - clock de 2 MHz - 640.000 instruções/s. 64KB de memória. 6.000 transistores.

1975 – Z80 (ZILOG), 6502 (MOS) – Utilizado pelo 1º APPLE (APPLE 1) em 1976 por Steve Wozniak e Steve Jobs (data da fundação da APPLE).

1975 – PIC (Peripheral Interface Controller) – Interface controladora de periféricos, projetada pela General Instruments para servir como porta de entrada e saída para o microprocessador CP 1600. A arquitetura projetada em 1975 é substancialmente a arquitetura de hoje do PIC16C5x

1976 – 8085 (INTEL) – “8080” operando com 5V - 2 instruções a + que o 8080 - melhor performance. 5 MHz - 370.000 instruções/s. 6500 transistores.

1978 – 8086 (INTEL) - Processador 16 bits (barramento externo de 16 bits e registradores de 16 bits). 5 MHz - 0.33 MIPS, 8 MHz - 0.66 MIPS e 10 MHz - 0.75 MIPS. 29.000 transistores.

1979 – 8088 (INTEL) - Processador 16 bits (barramento externo de 8 bits e registradores de 16 bits) - 133 instruções - chip utilizado no primeiro PC em 1981. O PC/XT seria lançado em 1983 com HD de 10 MB e 128 Kbyte RAM. 29.000 transistores. Lançado o 68.000 (MOTOROLA) que foi utilizado no Machintosh em 1984

1980 – Co-processador 8087 (processador matemático).

8051 (INTEL) – Lançado o microcontrolador 8051: microprocessador + periféricos (RAM, ROM, Serial, Timer, Controlador de Interrupção, etc.) num único chip, voltado para aplicações de controle

1982 – 80186/188 - 80286 - 80287 (INTEL) – PC/AT – 16 bits, modo protegido, 24 linhas endereços.

1985 – 80386 (INTEL) – Processador de 32 bits - bus externo de dados de 32bits - 275.000 transistores. 16MHz - 2.5 MIPS, 20 MHz - 2.5 MIPS, 25 MHz - 2.7 MIPS, 33 MHz - 2.9 MIPS.

1985 – Microcontrolador PIC (Microchip) – Microcontrolador criado pela recém fundada Microchip. Arquitetura RISC.

1989 – 80486 (INTEL) - Processador de 32 bits: “386” que incorpora o 387 (coprocessador), cache interna (L1) de 8KB e maior performance - 235 instruções - 1,2 milhões de transistores. 25 MHz - 20 MIPS, 33 MHz - 27 MIPS, 50 MHz - 41 MIPS.

1991 – WEB – Tim Berners-Lee desenvolve a Rede Mundial de Computadores (World Wide Web). O primeiro servidor Web é lançado. O conceito de conexão de vários usuários a um único computador por via remota nasceu no MIT no final da década de 50 e início da década de 60. As idéias básicas da Internet foram desenvolvidas em 1973 por **Bob Kahn e Vint Cerf**.

1993 – Pentium 60 MHz e 66 MHz - Processador de 32 bits – bus ext. de 64 bits - 5V - 3 milhões de transistores. Primeiro processador de 5^a geração.

1994 – Pentium 90 MHz e 100 MHz - Alimentação de 3,3V (maior confiabilidade). 3.2 milhões de transistores.

1996 – Pentium Pro 200 - Incorpora cache L2 de 256kB, utilizando tecnologia MCM (*Multi-Chip Module*) - 5 milhões de transistores - idealizado para programas de 32 bits. Usa memória de 64 bits.

1997 – Pentium 200MMX (Pentium MultiMidia eXtensions): contém 57 novas instruções dedicadas para programas de Multimídia. 4.5 milhões de transistores. 200 MHz e 166 MHz. Barramento de 64 bits. Cada instrução MMX equivale a várias instruções comuns.

1997 – Pentium II 233, 266, 300MHz – utiliza o slot I. 7,5 milhões de transistores (tecnologia 0.35 micron), cache L2 com 512kB - 242 pinos - 64GB de memória endereçável. Poder de processamento de 32 bits do Pentium Pro e maior eficiência no processamento de 16 bits. Instruções MMX.

1998 – Pentium II 450 MHz - Cache L2 de 512 kB, 7.5 milhões de transistores, tecnologia 0.25 micron, barramento de 64 bits. 64 GB de memória endereçável.

1999 – Pentium III 450 e 500 MHz (até 1,2 GHz) – Barramento de sistema de 100 MHz ou 133 MHz, cache L2 de 512 kB, processador de 32 bits, 9,5 milhões de transistores, tecnologia 0.25 micron, 64 GB de memória endereçável. 70 novas instruções voltadas para multimídia e processamento 3D.

2000 – Pentium IV – até 2 GHz, barramento de sistema de 400 MHz, Cachê L1 de 32 kB e L2 de 256 kB, 42 milhões de transistores.

2004 – Pentium 4 (Prescott) - processador de 32/64 bits, 125 milhões de transistores, 7.000 MIPS, tecnologia de 0,09 µm.

2005 – Pentium D – processador de 32 bits, 230 milhões de transistores, 26.000 MIPS, tecnologia de 90nm.

2006 – Core2 – processador de 32 bits, 152 milhões de transistores, 26.000 MIPS, tecnologia de 65 nm.

2007 – Core2 Duo – processador de 64 bits, 820 milhões de transistores, 53.000 MIPS, tecnologia de 45nm.

2008 – Core i7 – processador de 64 bits, 731 milhões de transistores, 76.000 MIPS, tecnologia de 45 nm

2008 – N270 – Família Átomo – Intel lança uma nova família de microprocessadores visando o mercado de dispositivos móveis, tais como notebooks, palmtops e iphones. São menores e mais eficientes. Tecnologia de 45 nm, cache de 512 kB L2, clock de até 2.13 GHz.

2009 – N450 – Intel lança uma nova geração de processadores da família átomo visando melhorar o desempenho de laptops e aumentar a vida útil das baterias desses equipamentos. É 60% menor que seus antecessores e consome 20% menos energia. A potência dissipada é de 5,5 W. Frequência de clock de 1,66 GHz.

2010 - CoreTM i5-661 – processador de 64 bits, 559 milhões de transistores de 32 nm e 3,33 GHz de frequência. Memória cache de 4 MB.

2012 - CoreTM i7-3930 – processador de 64 bits (6 núcleos), 3,2 GHz de frequência, memória cache de 12 MB.

Resumindo, o primeiro processador que a Intel lançou, em 1971 (Intel 4004) funcionava com uma frequência de 108 kHz e possuía 2.300 transistores, sendo que cada transistor tinha 10 µm de tamanho. O novo processador Intel® CoreTM i5-661, lançado em 2010, funciona com frequência de 3,33 GHz e possui 559 milhões de transistores de 32 nm.

Observa-se, ao longo da linha de tempo, um aumento no número de transistores nos microprocessadores e uma redução no tamanho desses componentes (tecnologia empregada). A redução no tamanho dos transistores resulta no aumento da velocidade de operação e também na redução das conexões internas, além de permitir a inserção de um número cada vez maior de transistores numa única pastilha. O aumento da capacidade de integração de transistores resulta ainda na redução do consumo de energia elétrica e do custo dos microprocessadores. Há um postulado que diz que o gate de um transistor não pode ser menor do que a largura correspondente a 10 átomos. A previsão de pesquisadores da Intel é que a dimensão do gate dos transistores alcançará esse valor por volta do ano 2017.

1.3 Sistema de Numeração

No estudo de microprocessadores o sistema de numeração mais utilizado é o hexadecimal, uma vez que os dados e os endereços são manipulados em hexadecimal. Algumas análises, no entanto, são feitas utilizando-se o sistema binário. Por exemplo, as operações lógicas e a rotação de dados do acumulador. Também utiliza-se o sistema decimal, principalmente na entrada e saída de dados, de modo a facilitar a interação com o usuário. Assim, são apresentados a seguir os principais sistemas de numeração e a conversão de valores entre eles.

1.3.1 Sistema Decimal

O sistema decimal utiliza 10 dígitos, que vão de 0 a 9. **Exemplo: 346₁₀**

1º dígito: Armazena as unidades (ou $10^0 = 1$). No ex.: seis unidades (ou 6×10^0);

2º dígito: Armazena as dezenas (ou $10^1 = 10$). No ex.: quatro dezenas (ou 4×10^1);

3º dígito: Armazena as centenas (ou $10^2 = 100$). No ex.: três centenas (ou 3×10^2);

A soma destas parcelas equivale a: $300 + 40 + 6 = 346_{10}$

A ponderação é dada pelo número 10 elevado à potência representada pela coluna, sendo que a 1^a coluna da direita é 0.

1.3.2 Sistema Binário

O sistema binário é o sistema de numeração que o computador entende. Utiliza 2 dígitos, 0 e 1 ou (OFF e ON) ou (0V e 5V), ou (0V e 3,3V). **Exemplo:** **11001011₂**

1º dígito: Armazena o equivalente a 2^0 (1). No ex.: 1×2^0

2º dígito: Armazena o equivalente a 2^1 (2). No ex.: 1×2^1

3º dígito: Armazena o equivalente a 2^2 (4). No ex.: 0×2^2

...

8º dígito: Armazena o equivalente a 2^7 : No ex.: 1×2^7

A soma destas parcelas resulta no seguinte equivalente decimal:

$$128 + 64 + 0 + 0 + 8 + 0 + 2 + 1 = 203_{10}$$

A ponderação é dada pelo número 2 elevado à potência representada pela coluna, sendo que a 1ª coluna é 0, a segunda coluna é 1 e assim sucessivamente.

1.3.3 Sistema BCD (Binary-Coded Decimal)

O Sistema BCD é o sistema em que se combina o sistema binário e o sistema decimal. É utilizado como formato de saída de instrumentos. Utiliza 2 dígitos: 0 e 1 que são dispostos em grupos de 4 dígitos, utilizados para representar um dígito decimal (número 0 até 9). A representação de um número maior que 9 deve ser feita por outro grupo de 4 bits, com a ponderação dada pelo sistema decimal.

Exemplo: **973₁₀ = 1001 0111 0011₂**.

Note a diferença entre este valor e o valor do número binário **1001 0111 0011₂ = 2419₁₀**

1.3.4 Sistema Octal

O Sistema Octal é baseado nos mesmos princípios do decimal e do binário, apenas utilizando base 8. Utiliza 8 dígitos: 0 a 7. **Exemplo:** **3207₈**

1º dígito: Armazena o equivalente a 8^0 (1). No ex.: 7×8^0

2º dígito: Armazena o equivalente a 8^1 (8). No ex.: 0×8^1

3º dígito: Armazena o equivalente a 8^2 (64). No ex.: 2×8^2

4º dígito: Armazena o equivalente a 8^3 (512). No ex.: 3×8^3

O equivalente decimal é: $1536 + 128 + 0 + 7 = 1671_{10}$

1.3.5 Sistema Hexadecimal

O Sistema Hexadecimal é baseado nos mesmos princípios do decimal, apenas utiliza base 16. Utiliza 16 dígitos: 0 a 9, A, B, C, D, E e F. Exemplo: **Ex.: 20DH ou 20Dh ou 20D₁₆**

1º dígito: Armazena o equivalente a 16^0 (1). No ex.: 13×16^0

2º dígito: Armazena o equivalente a 16^1 (16). No ex.: 0×16^1

3º dígito: Armazena o equivalente a 16^2 (256). No ex.: 2×16^2

O equivalente decimal é: $512 + 0 + 13 = 525_{10}$

1.3.6 Conversão de Base

O sistema hexadecimal é mais fácil de trabalhar que o sistema binário e é utilizado na codificação de programas no microprocessador. Na conversão de hexadecimal para binário, cada dígito hexadecimal é convertido em 4 dígitos binários equivalentes. Na conversão de binário para hexadecimal, cada grupo de 4 dígitos binários é convertido em 1 dígito hexadecimal equivalente.

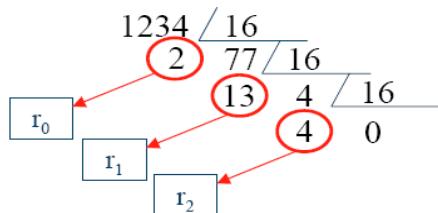
Exemplos:

$$7 \text{ D } 3 \text{ F}_{16} \quad (7D3FH) = 0111\ 1101\ 0011\ 1111_2$$

$$1010000110111000_2 = 1010\ 0001\ 1011\ 1000_2 = A\ 1\ B\ 8_{16} \text{ ou A1B8H}$$

Um algoritmo para a conversão de base pode ser obtido. A divisão termina quando o quociente é zero. Um exemplo de decimal para hexadecimal é dado.

$$\begin{array}{r} \text{Valor} \diagup \text{Base} \\ r_0 \quad q_0 \quad \diagup \text{Base} \\ \quad \quad \quad r_1 \quad q_1 \quad \diagup \text{Base} \\ \quad \quad \quad \quad \quad r_2 \quad 0 \end{array}$$



Representação: $r_2\ r_1\ r_0$

Valor hexadecimal correspondente a 1234: 4D2 H

Algoritmo genérico:

$$\text{Se } q_0 = 0 \rightarrow \text{Valor} = q_0 B + r_0 = 0.B + r_0 = r_0$$

$$\text{Valor} = r_0 \rightarrow \text{Representação: } r_0$$

$$\text{Se } q_1 = 0 \rightarrow \text{Valor} = q_0 B + r_0; \quad q_0 = q_1 \cdot B + r_1 = 0.B + r_1 = r_1 \quad \text{ou,}$$

$$\text{Valor} = r_1 \cdot B + r_0 \rightarrow \text{Representação: } r_1\ r_0$$

$$\text{Se } q_2 = 0 \rightarrow \text{Valor} = q_0 B + r_0; \quad q_0 = q_1 \cdot B + r_1; \quad q_1 = q_2 \cdot B + r_2 = 0.B + r_2 = r_2$$

Daí, $q_0 = r_2 \cdot B + r_1$ e $\text{Valor} = (r_2 \cdot B + r_1) B + r_0$. Logo,

$$\text{Valor} = r_2 B^2 + r_1 B + r_0 \rightarrow \text{Representação: } r_2\ r_1\ r_0$$

1.3.7 Representação de Números Positivos e Negativos

Na representação de números no microprocessador pode-se utilizar tanto a representação de números com sinal, quanto a de números sem sinal. O círculo da figura a seguir mostra as possibilidades de um número de 4 bits. Na representação de número com sinal, o bit mais significativo representa o sinal do número. No caso do número de 4 bits, o bit 3 indica se o número é positivo ou negativo. Se ele é zero o número é positivo, se for 1, é negativo.

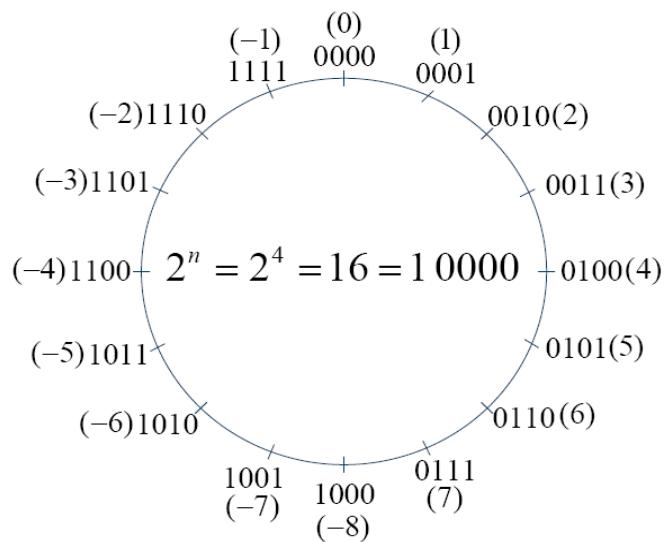


Fig. 1.11 – Círculo para um número de 4 bits

Valor Simétrico de um Número

Número binário: $-a = (\text{complemento de } 1 \text{ de } a) + 1 = \text{complemento de } 2 \text{ de } a = 2^n - a$

Número decimal: $-a = (\text{complemento de } 9 \text{ de } a) + 1 = \text{complemento de } 10 \text{ de } a = 10^n - a$

Subtração usando adição de um número binário:

$$a - b = a + (\text{complemento de } 2 \text{ de } b) \rightarrow a - b = a + (2^n - b)$$

Exemplo para um número binário de 4 dígitos:

$$a - 1 = a + (2^4 - 1) = a + (10000 - 0001) = a + 1111$$

$$a - 3 = a + (2^4 - 3) = a + (10000 - 0011) = a + 1101$$

Se $a = 1001$ (9_{10}), então a subtração no modo direto fica:

$$a - 1 = 1001 - 0001 = 1000$$

$$a - 3 = 1001 - 0011 = 0110$$

E a subtração usando a adição com o complemento de 2 do número fica:

$$a - 1 = 1001 + 1111 = 1 \underline{1000} \text{ (despreza-se o quinto dígito)}$$

$$a - 3 = 1001 + 1010 = 1 \underline{0110} \text{ (o número é de 4 dígitos)}$$

Subtração usando adição de um número decimal:

$$a - b = a + (\text{complemento de } 10 \text{ de } a) \rightarrow a - b = a + (10^n - b)$$

Exemplo para um número decimal de 2 dígitos:

$$a - 1 = a + (10^2 - 1) = a + (100 - 1) = a + 99$$

$$a - 3 = a + (10^2 - 3) = a + (100 - 3) = a + 97$$

Se $a = 94$, então a subtração de modo direto fica:

$$a - 1 = 94 - 1 = 93$$

$$a - 3 = 94 - 3 = 91$$

E a subtração usando a adição com o complemento de 10 do número fica:

$$a - 1 = 94 + 99 = 1 \underline{93} \text{ (despreza-se o terceiro dígito)}$$

$$a - 3 = 94 + 97 = 1 \underline{91} \text{ (o número é de 2 dígitos)}$$

2 Arquitetura e Princípio de Funcionamento de Microprocessadores

2.1 Arquiteturas RISC, CISC e Híbrida

Há dois tipos básicos de arquitetura de microprocessadores: CISC (Complex Instruction Set Computer – Computador com Conjunto Complexo de Instruções) e RISC (Reduced Instruction Set Computer – Computador com Conjunto Reduzido de Instruções), e uma terceira, que utiliza características dos dois tipos básicos. Uma das características que diferenciam as duas arquiteturas é que na arquitetura CISC o conjunto de instruções contém muito mais instruções do que na arquitetura RISC. A Tabela 2.1 lista essa e outras características das duas arquiteturas.

Tabela 2.1: Comparação entre arquiteturas CISC e RISC

CISC	RISC
Conjunto com dezenas ou centenas de instruções	Conjunto com poucas dezenas de instruções
Instruções complexas, com vários ciclos de execução	Instruções simples, normalmente executadas em 1 ciclo (1 ciclo de busca e 1 de execução)
Muitas instruções com acesso à memória	Uso reduzido da memória
Número reduzido de registradores	Quantidade grande de registradores. Uso preferencial de registradores, ao invés de memória
Menor número de instruções em assembly por programa, porém mais lento na execução	Programa compilado tem maior número de instruções em assembly, mas a execução é mais rápida
Uso de micro-códigos gravados no processador. Necessidade de interpretação das instruções	Não usa micro-códigos gravados no processador. As instruções RISC já são semelhantes aos micro-códigos e são executadas diretamente no hardware
Instruções com diferentes tamanhos	Codificação das instruções em uma palavra de tamanho fixo
Arquitetura mais complexa	Arquitetura mais simples
Processo de fabricação mais caro que a dos processadores RISC	Processo de fabricação mais barato
Exemplos: 386 e 486	Exemplos: MIPS R10000 e HP PA-8000
Vários modos de endereçamento	Poucos modos de endereçamento
Nem todos os processadores usam pipeline. Por exemplo, o 8085.	Uso intenso de pipelines
Pelo menos um dos argumentos de cada instrução é buscado na memória.	As operações com a memória principal estão restritas a transferência através das instruções load e store.
Endereços de retorno de subrotinas e interrupções são guardados na pilha, uma região previamente definida na memória	Endereços de retorno de subrotinas são normalmente guardados em registradores
Devido ao número reduzido de registradores, as variáveis de um programa muitas vezes são associadas a posições de memória	Uso de registradores para a alocação das variáveis de um programa

Um processador híbrido usa as características de operação da arquitetura CISC, mas possui um núcleo RISC. Uma das vantagens da arquitetura CISC é o fato de já possuir as micro-instruções previamente gravadas no processador, facilitando o trabalho dos programadores. Por outro lado, os micro-códigos são usados após a decodificação da instrução, ou seja, após o ciclo de busca. Na arquitetura CISC, antes de executar uma instrução, há necessidade de busca da instrução na memória e de decodificação. Utiliza-se micro-códigos gravados no processador, para a execução das instruções.

Na arquitetura RISC as instruções são equivalentes aos micro-códigos da arquitetura CISC, não precisando de decodificação. Assim, na arquitetura RISC as instruções são executadas em um único ciclo de via de dados. De fato, as instruções são buscadas em um ciclo de clock e executadas, normalmente, no próximo ciclo. Efetivamente, devido ao uso de pipeline, explicado mais adiante, a cada

ciclo uma nova instrução é buscada. Não é possível ter instruções de multiplicação e divisão, por exemplo, por exigir muitos ciclos para execução. Multiplicações são resolvidas com adições e deslocamentos. A Fig. 2.1 compara as etapas de execução na arquitetura RISC e na CISC.

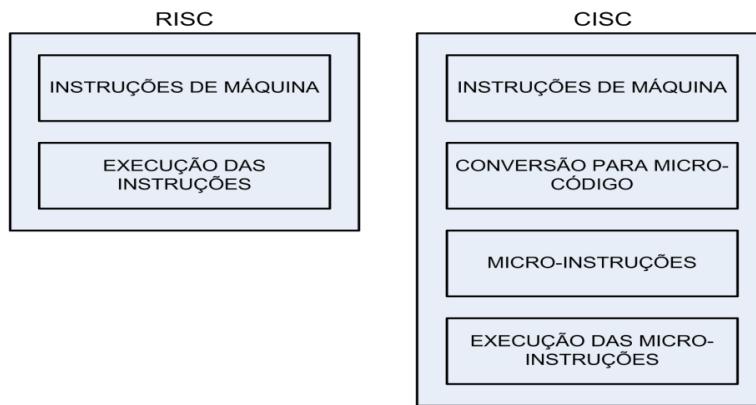


Fig. 2.1 – Etapas de execução nas arquiteturas RISC e CISC

O pipeline é uma técnica usada para acelerar a execução de instruções. A cada ciclo de clock, enquanto uma instrução está na etapa de execução, a instrução seguinte está sendo buscada. O resultado global é que, a cada ciclo, uma nova instrução é iniciada e uma instrução é encerrada. No caso mostrado a seguir a instrução B faz referência à memória e, assim, enquanto a instrução A necessita de apenas um ciclo para busca e um para execução, a instrução B precisa de um ciclo para busca e dois para execução.

Ciclos	1	2	3	4	5
Busca da Instrução	A	B	C	D	E
Execução da Instrução		A	B	C	D
Referência à Memória				B	

Caso a instrução B interfira na etapa de execução da instrução C (por exemplo, usando o mesmo registrador ou quando a instrução C precisa do resultado da instrução B) é necessário aguardar o término da instrução B antes de executar a instrução C.

Ciclos	1	2	3	4	5	6
Busca da Instrução	A	B	C	NOP	D	E
Execução da Instrução		A	B	NOP	C	D
Referência à Memória				B		

Assim, a arquitetura RISC apresenta algumas vantagens quando comparada com a arquitetura CISC:

- Velocidade de execução;
- O uso de pipeline torna os processadores RISC duas a quatro vezes mais rápidos que um CISC de mesmo clock;
- Simplicidade de Hardware;
- Ocupa menos espaço no chip, devido ao fato de trabalhar com instruções simples;
- Instruções de máquina simples e pequenas, o que aumenta seu desempenho.

No entanto, a arquitetura RISC também apresenta desvantagens com relação à arquitetura CISC:

- O desempenho de um processador RISC depende diretamente do código gerado pelo programador. Um código mal desenvolvido pode resultar em tempo de execução muito grande;
- Um programa originalmente compilado para uma máquina CISC tem um equivalente compilado para máquina RISC com uma quantidade muito maior de códigos assembly, ocupando um espaço maior na memória;

- A arquitetura RISC requer sistema de memória rápida para alimentar suas instruções. Normalmente possuem grande quantidade de memória cachê interna, o que encarece o projeto.

A equação a seguir pode ser usada para avaliar o desempenho de um processador:

$$\text{Tempo de processamento} = \frac{\text{tempo}}{\text{ciclo}} \times \frac{\text{número de ciclos}}{\text{instrução}} \times \frac{\text{instruções}}{\text{programa}}$$

A arquitetura CISC prioriza a redução do número de instruções por programa, para obter um menor tempo de processamento; a arquitetura RISC prioriza a redução do número de ciclos por instrução, em detrimento do número de instruções por programa. A multiplicação entre dois números, armazenados em memória é um bom exemplo para ilustrar o procedimento usado nas duas arquiteturas:

Suponha dois valores armazenados nas posições M1 e M2 da memória principal. Deseja-se multiplicar esses valores e armazenar o resultado na posição de memória M1.

Procedimento CISC:

Normalmente uma única instrução é necessária. Por exemplo:

MUL M1,M2.

Essa instrução busca o código da instrução na memória, identifica a instrução, busca o conteúdo das posições indicadas da memória, faz a multiplicação e armazena o resultado em M1. A instrução é única, mas são necessários vários ciclos para executá-la.

Procedimento RISC:

Uma instrução é usada para transferir o conteúdo da memória para um registrador:

LOAD A,M1
LOAD B,M2

Uma instrução é usada para fazer o produto desses valores:

PROD A,B

Uma instrução é usada para armazenar o resultado na memória:

STORE M1,A

2.2 Arquitetura 8085, 8088/86, 8051 e PIC

O microprocessador 8085 e o microcontrolador 8051 são equivalentes em arquitetura. A diferença está em que, sendo um microcontrolador, o 8051 possui várias unidades adicionais (portas de entrada e saída paralela e serial, memória ROM, memória RAM, temporizadores etc) além da CPU. O microprocessador 8085 equivale à CPU do 8051. Os registradores de ambos são de 8 bits, exceto o registrador PC (*Program Counter* – Contador de Programa), que é de 16 bits. O Apontador de Pilha, SP (*Stack Pointer*) é um registrador de 16 bits no 8085 e de 8 bits no 8051. O Apontador de Dados, D PTR, do 8051, é um registrador de 16 bits, formado por dois registradores de 8 bits (DPH e DPL), e equivale ao par HL (16 bits) do 8085, formado pelos registradores H e L, de 8 bits). As Fig. 2.2 e 2.3 mostram, respectivamente, a arquitetura do 8085 e do 8051.

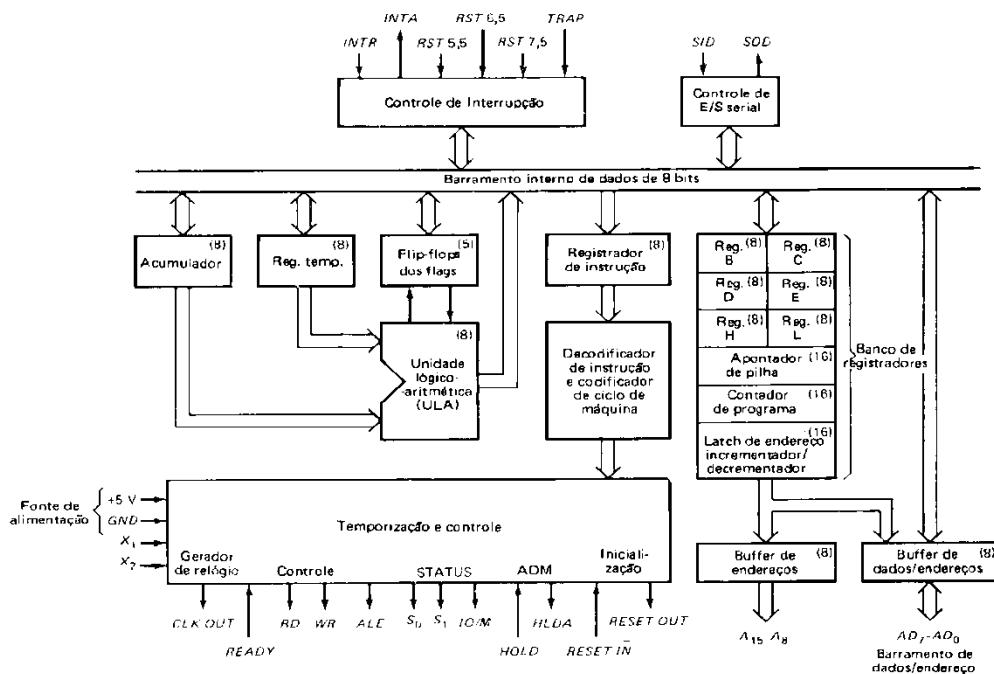


Fig. 2.2 – Estrutura interna do microprocessador 8085 (CISC)

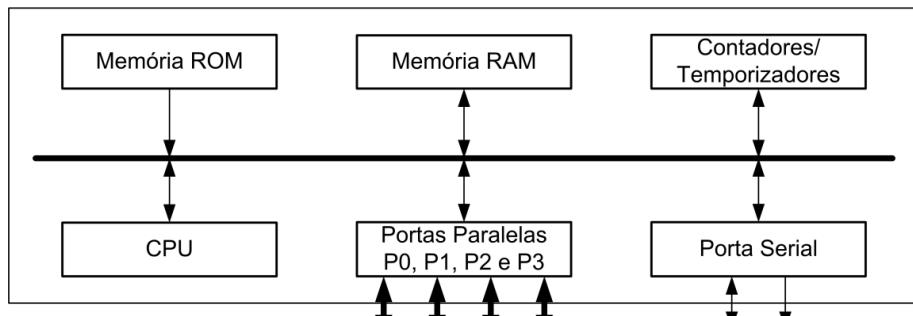


Fig. 2.3 – Estrutura básica do microcontrolador 8051 (CISC).

Na comparação entre o microprocessador 8086/88 e o 8085 a principal diferença está nos registradores de 16 bits do 8086/88, em contraste com os registradores de 8 bits do 8085. Outra diferença fundamental é que o 8086 possui unidade de busca independente da unidade de execução, o que permite buscar uma nova instrução (e colocar numa fila) antes de terminar de executar a instrução anterior. No 8085 a busca da próxima instrução é feita após a execução da instrução anterior. Busca-se a primeira instrução, executa-se essa instrução e então busca-se a próxima instrução.

A Tabela 2.1 mostra algumas características desses microprocessadores. Dentre elas destaca-se a diferença básica entre o 8086 e o 8088, que é o barramento externo de 8 bits do 8088, o que permite o interfaceamento com periféricos de 8 bits e com estruturas previamente construídas para o 8085.

Tabela 2.1 – Comparação entre os microprocessadores 8085 e 8086/88

Característica	Microprocessador 8085	Microprocessador 8088	Microprocessador 8086
Barramento de endereço	16 bits	20 bits	20 bits
Capacidade de endereçamento de memória	65.536 (64 kB)	1.048.576 (1 MB)	1.048.576 (1 MB)
Barramento de dados	8 bits	Interno: 16 bits Externo: 8 bits	Interno: 16 bits Externo: 16 bits
Manipulação de STRINGS	Não	Sim	Sim

Registradores Internos	8 bits e 16 bits	16 bits	16 bits
Uso de segmentação para endereçamento	Não	Sim	Sim
Aritmética Decimal completa	Não	Sim	Sim
Etapas de Busca e Execução	Em sequência: Busca → Executa	Unidades Independentes: Unidade de Interfaceamento com Barramento (BIU) – responsável pela Busca e Unidade de Execução (EU)	Unidades Independentes: Unidade de Interfaceamento com Barramento (BIU) – responsável pela Busca e Unidade de Execução (EU)

A Fig. 2.4 mostra a estrutura básica do microprocessador 8086/88, destacando a Unidade de Interfaceamento com o Barramento (BIU), responsável pela busca de instruções e agrupamento em uma fila (queue) e a Unidade de Execução, responsável pela execução das instruções previamente guardadas na fila.

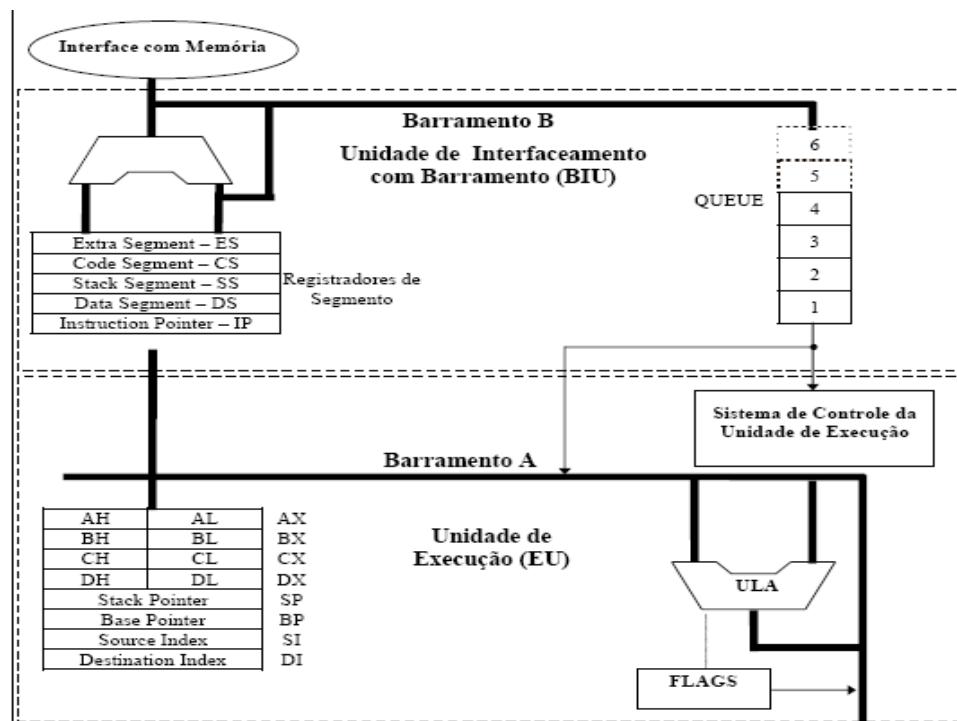


Fig. 2.4 – Estrutura básica do microprocessador 8086/88 (CISC).

Observando as arquiteturas das Fig. 2.2, 2.3 e 2.4, verifica-se uma característica chave da arquitetura CISC: nessa arquitetura os barramentos de dados e endereços são comuns a todas as unidades internas. Assim, a comunicação entre unidades é sempre através desse barramento comum, não permitindo operações em paralelo. Na arquitetura RISC, mostrada nas Fig. 2.5 e 2.6, há mais de uma via de comunicação entre unidades. Destaca-se aqui o fato da unidade lógica e aritmética está conectada à memória de programa (instruções) e à memória de dados através de canais diferentes, o que permite operações simultâneas de busca de instrução e de dados. A Fig. 2.6 mostra a estrutura de um microcontrolador da família PIC (arquitetura RISC).

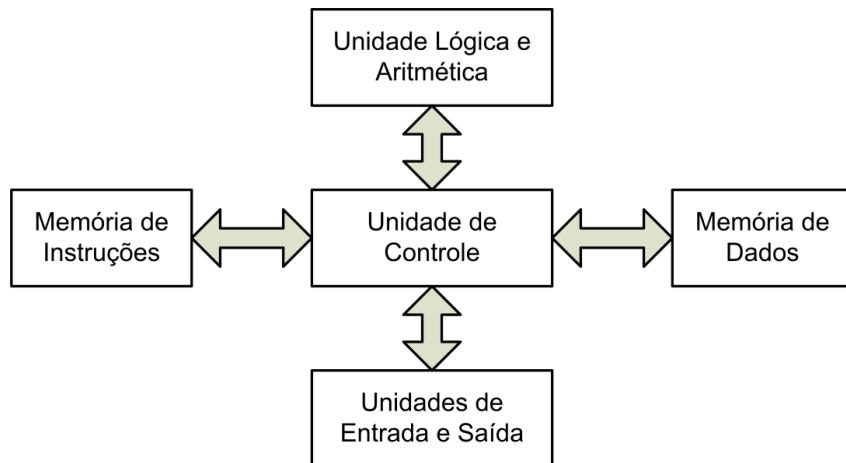


Fig. 2.5 – Estrutura Básica de microcontroladores com arquitetura RISC

FIGURE 3-1: PIC16F8X BLOCK DIAGRAM

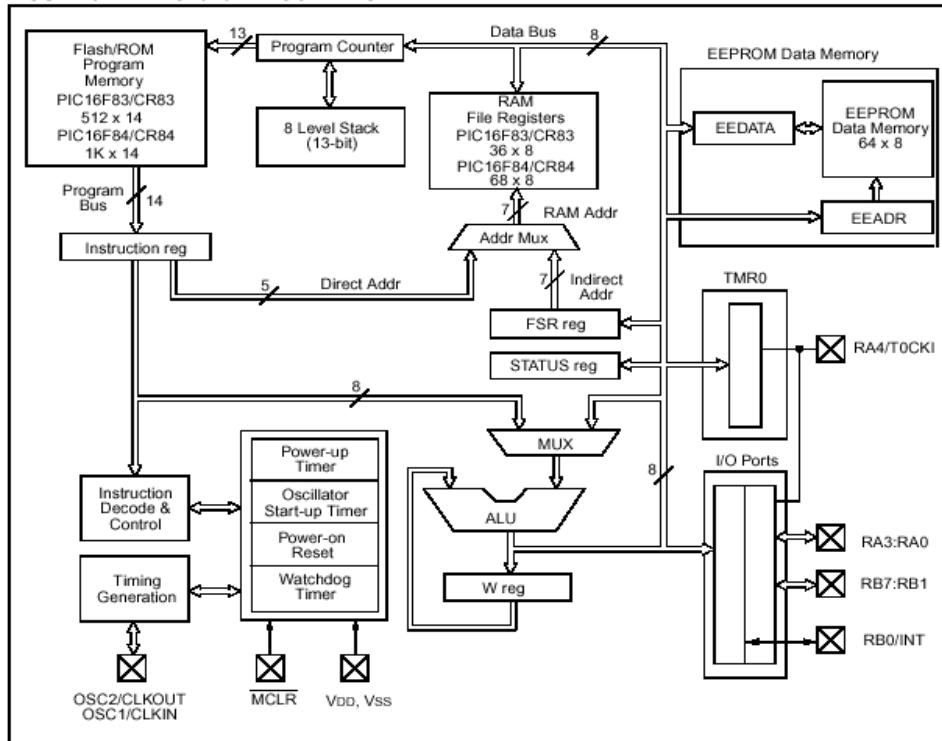


Fig. 2.6 – Estrutura do microcontrolador da família PIC 16F8X (RISC)

2.3 Registradores Principais 8085, 8088/8086, 8051

A quantidade de registradores em um microprocessador/microcontrolador é fundamental para o desempenho na execução de tarefas porque a movimentação de dados entre registradores é mais rápida do que as operações que utilizam a memória. Os microprocessadores 8085 e 8086/8088 possuem poucos registradores quando comparados com os microcontroladores da família 8051 e da família PIC. Assim, as variáveis de um programa são alocadas em posições de memória, ao invés de registradores. E nesse processo são necessários vários modos de endereçamento. Na arquitetura RISC há uma quantidade grande de registradores (em média 512 – com 32 visíveis por vez: 8 para variáveis globais e ponteiros, 8 para parâmetros de entrada, 8 para variáveis locais e 8 para parâmetros de saída). Dessa forma, há um número reduzido de acesso à memória (o acesso à memória torna o processamento mais lento). A alocação de variáveis é feita em registradores e há apenas um ou dois modos de endereçamento para acesso à memória.

A Tabela 2.2 mostra os registradores do 8085 e do 8086/8088. Na comparação entre eles destaca-se: o acumulador do 8085 (principal registrador) é equivalente ao byte inferior do acumulador primário do 8086/8088; o par HL equivale ao registrador BX, que é composto pelos registradores BH e BL; os registradores B e C, equivalentes aos registradores CH e CL, que formam o registrador CX e os registradores D e E equivalentes aos registradores DH e DL, que formam o registrador DX. Os apontadores de pilha, SP, são equivalentes. Além do apontador de pilha, o 8086/8088 possui outros três registradores apontadores (BP, SI, DI), que são usados como *offset* na formação do endereço absoluto. O registrador PC do 8085 é equivalente ao ponteiro de instrução IP, 8086. Enquanto o registrador PC já fornece diretamente o endereço da próxima instrução, o ponteiro IP precisa do registrador de segmento CS para a formação do endereço absoluto que indica a próxima instrução. Os outros registradores de segmento (DS, SS, ES) permitem a seleção de outras áreas de memória de 64 kB, dentro da memória principal de 1 MB. Os registradores de flags diferem no número de bits e no número de flags. No 8085 são 5 flags e no 8086/8088 são 9 flags (6 refletem o resultado das operações lógicas e aritméticas e 3 são bits de controle).

Tabela 2.2 – Registradores dos microprocessadores 8085 e 8086/8088

Registradores do 8085			Registradores do 8086/8088							
	A	Acumulador	AH	AL(A)	AX – Acumulador Primário					
H	L	Apontador de dados	BH	BL	BX – Acumulador e Registrador Base					
B	C		CH	CL	CX – Acumulador e Contador					
D	E		DH	DL	DX – Acumulador e Endereçador de I/O					
SP		Apontador de pilha	SP	Apontador de pilha						
			BP	Apontador Base – usado na pilha						
			SI	Índice da Fonte – usado para indexação						
			DI	Índice de Destino – usado para indexação						
PC		Contador de Programa	IP	Ponteiro de Instrução						
			CS	Segmento de Código		Registradores de Segmento. Usados para a formação do endereço absoluto				
			DS	Segmento de Dados						
			SS	Segmento de Pilha						
			ES	Segmento Extra						
	F	Registrador de Flags	FLAGS	Registrador de Flags						

Há dois grupos de registradores no microcontrolador 8051: os registradores de propósito geral, num total de 32, que ocupam a parte baixa da memória RAM e os registradores especiais, cujos endereços são equivalentes aos endereços da parte alta da memória RAM. A Tabela 2.3 mostra os registradores especiais e a Fig. 2.6 mostra os registradores de propósito geral.

Os registradores podem ser referidos pelo mnemônico ou pelo endereço. Na região dos registradores especiais, por exemplo, o acumulador pode ser referido como **A**, **ACC** ou pelo endereço **E0 H**. Esse registrador especial e outros da lista podem ainda ser acessados por bit. Os latches das portas de entrada/saída (P0, P1, P2 e P3) também são registradores especiais cujos endereços são, respectivamente **80 H**, **90 H**, **A0 H** e **B0 H**.

Tabela 2.2: Principais Registradores Especiais

Registrador	Mnemônico	Endereço	Endereços individuais dos Bits e denominações de alguns bits							
			87	86	85	84	83	82	81	80
Latch da Porta 0	P0	80 H								
Apontador de Pilha	SP	81 H								
Apontador de Dados	DPTR	82H – 83H								
LSB do Apontador de Dados	DPL	82 H								
MSB do Apontador de Dados	DPH	83 H								
Controle de Energia	PCON	87 H	SMOD							
Controle do Contador/Temporizador	TCON	88 H	8F	8E	8D	8C	8B	8A	89	88
			TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Controle do Modo do Temporizador/	TMOD	89 H	G1	C/T1	M11	M01	G0	C/T0	M10	M00

Contador									
LSB do Temporizador/Contador 0	TL0	8A H							
LSB do Temporizador/Contador 1	TL1	8B H							
MSB do Temporizador/Contador 0	TH0	8C H							
MSB do Temporizador/Contador 1	TH1	8D H							
Latch da Porta 1	P1	90 H	97	96	95	94	93	92	91
Controle da Porta Serial	SCON	98 H	9F	9E	9D	9C	9B	9A	99
			SM1	SM2	SM3	REN	TB8	RB8	TI
Porta de Dados Seriais	SBUF	99 H							
Latch da Porta 2	P2	A0 H	A7	A6	A5	A4	A3	A2	A1
Habilitador de Interrupção	IE	A8 H	AF	AE	AD	AC	AB	AA	A9
			EA			ES	ET1	EX1	ET0
Latch da Porta 3	P3	B0 H	B7	B6	B5	B4	B3	B2	B1
Controle de Prioridade da Interrup.	IP	B8 H	BF	BE	BD	BC	BB	BA	B9
						PS	PT1	PX1	PT0
Registrador de Estado do Programa	PSW	D0 H	D7	D6	D5	D4	D3	D2	D1
			CY	AC	F0	RS1	RS0	OV	P
Acumulador	ACC ou A	E0 H	E7	E6	E5	E4	E3	E2	E1
Registrador B	B	F0 H	F7	F6	F5	F4	F3	F2	F1
									F0

Na região de registradores de propósito geral verifica-se que há quatro bancos de registradores, cada um com 8 registradores. Os registradores de cada banco são denominados de **R0** a **R7**. Os bits responsáveis pela seleção do **R0** do banco 0 ou **R0** do banco 3, por exemplo, são os bits **RS1** e **RS0** do registrador especial **PSW** (Program Status Word), mostrado a seguir, e que também contém as flags do 8051. As flags serão comentadas em uma seção posterior.

PSW (Program Status Word) - D8H

D7H	D7H	D5H	D4H	D3H	D2H	D1H	D0H
CY	AC	F0	RS1	RS0	OV	x	P

RS1	RS0	Banco Selecionado
0	0	0
0	1	1
1	0	2
1	1	3

Ao invés de fazer referência aos registradores R0 a R7 pelo nome, pode-se usar os endereços desses registradores, que variam de **00 H** (R0 do banco 0) até **1F H** (R7 do banco 3).

	R7	1F h	
	R6	1E h	
	R5	1D h	
	R4	1C h	
	R3	1B h	
	R2	1A h	
	R1	19 h	
	R0	18 h	
	R7	17 h	
	R6	16 h	
	R5	15 h	
	R4	14 h	
	R3	13 h	
	R2	12 h	
	R1	11 h	
	R0	10 h	
	R7	0F h	
	R6	0E h	
	R5	0D h	
	R4	0C h	
	R3	0B h	
	R2	0A h	
	R1	08 h	
	R0	08 h	
	R7	07 h	
	R6	06 h	
	R5	05 h	
	R4	04 h	
	R3	03 h	
	R2	02 h	
	R1	01 h	
	R0	00 h	

Fig. 2.6 – Registradores de propósito geral do 8051 – parte baixa da memória RAM

2.4 Princípio de Funcionamento 8085, 8088/86, 8051

2.4.1 Microprocessador 8085

A execução de uma instrução no microprocessador 8085 começa com a busca da instrução na memória e na decodificação dessa instrução. Isso ocorre em três ciclos de clock.

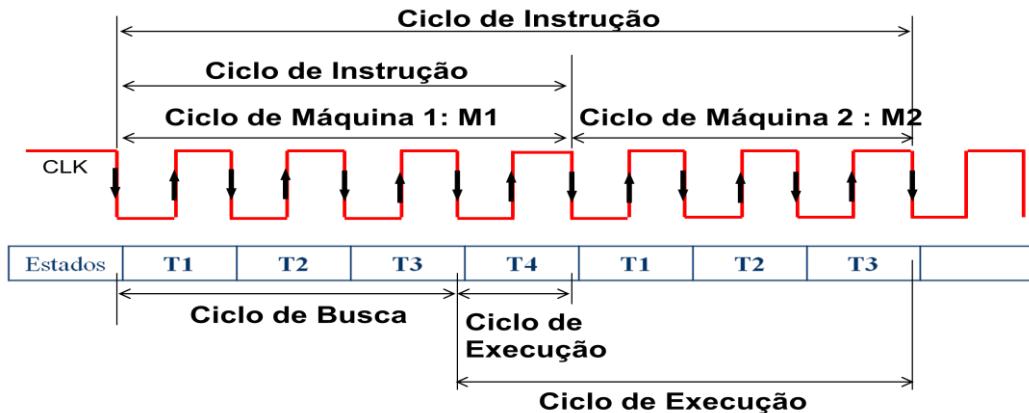


Fig. 2.7 – Ciclos de Máquina e de Instrução do 8085

O três primeiros estados de uma instrução (T1, T2 e T3) correspondem ao ciclo de busca. Cada estado corresponde a um ciclo de clock. A frequência que corresponde a um ciclo de clock é a frequência do cristal oscilador dividida por 2. Assim, se o cristal oscilador for de 2 MHz, a frequência de clock é 1 MHz e o período de clock é 1 μ s. O microprocessador 8085A-2 pode ser operado com frequência do cristal de 500 kHz a 5 MHz.

Após a busca e decodificação da instrução o microprocessador passa para a etapa de execução, que pode ocorrer em 1 ou mais ciclos de clock. Uma vez completada a execução da instrução, está completo o ciclo de instrução, que é composto pelo ciclo de busca mais o ciclo de execução.

Se o ciclo de execução ocorrer em mais de um ciclo de clock inicia-se um novo ciclo denominado de ciclo de máquina. A cada início de um novo ciclo de máquina tem um novo valor do contador de programa PC. O primeiro ciclo de máquina tem três ciclos de clock (T1 a T4), os demais tem três ciclos de clock (T1, T2 e T3). O diagrama de temporização mostrado na Fig. 2.8 ilustra os sinais de controle presentes durante a execução de uma instrução. São mostrados apenas os dois primeiros ciclos de máquina.

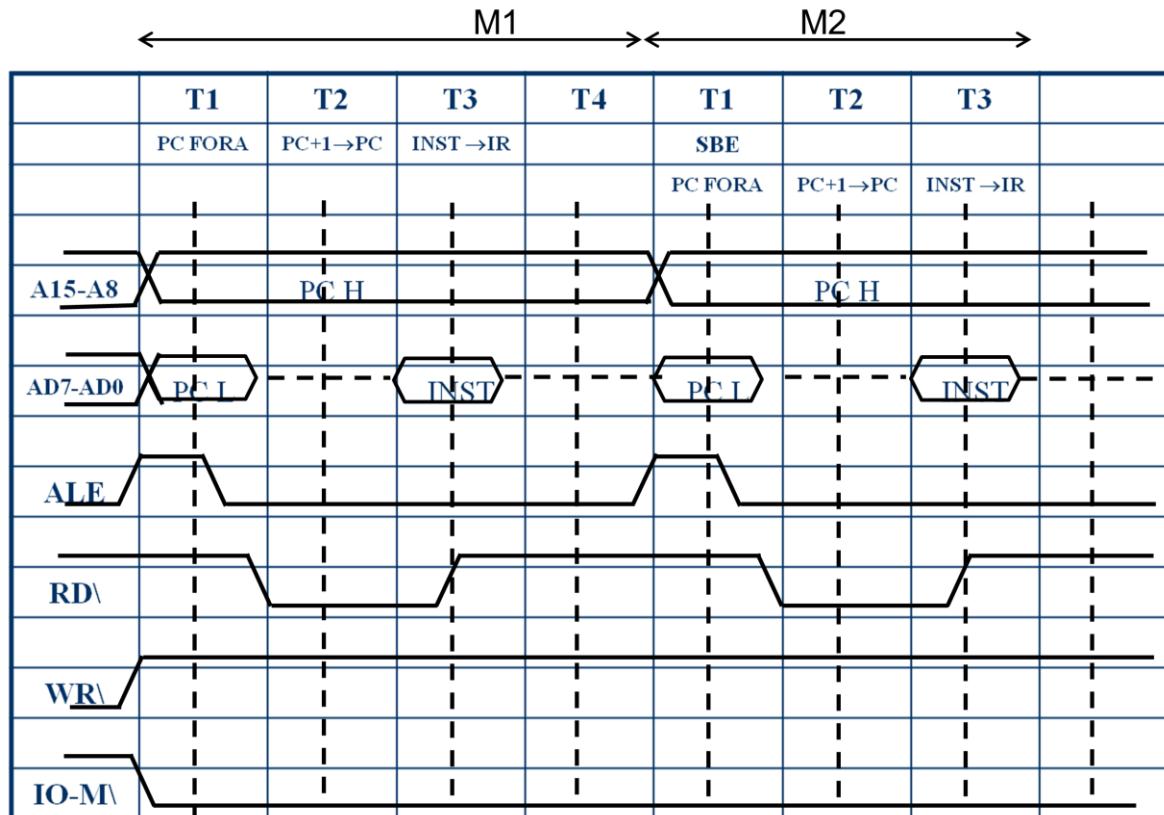


Fig. 2.8 – Ciclos de Máquina e de Instrução do 8085

Observa-se no diagrama de temporização que o sinal ALE (Adress Latch Enable) está presente no início de cada ciclo de máquina. Ele habilita o registrador PC a colocar no barramento o endereço da próxima instrução (ou endereço do dado a ser buscado na memória). Esse sinal é também usado no controle de periféricos.

O barramento denominado A15-A8 contém a parte alta do endereço de 16 bits (PC H). O barramento AD7-AD0 contém a parte baixa do endereço a ser acessado na memória, ou a instrução que foi buscada na memória ou ainda dados originados da memória ou de registradores.

Os sinais RD\, WR\ e IO-M\ são sinais de controle de periféricos. Eles indicam o estado das operações em andamento. RD\ baixo ($RD\ = 0$) indica que o microprocessador está em período de leitura, que pode ser de memória ($IO-M\ = 0$) ou de uma porta de entrada ($IO-M\ = 1$). WR\ baixo ($WR\ = 0$) indica uma operação de escrita em memória ($IO-M\ = 0$) ou em uma porta de saída ($IO-M\ = 1$).

O que ocorre em cada estado do diagrama de temporização é mostrado na parte superior da Fig. 2.8:

Estado T1 (ciclo de máquina M1): o conteúdo do registrador PC (contador de programa) é colocado no barramento e indica o endereço de leitura da memória para obtenção do código de operação (opcode) da instrução.

Estado T2 (ciclo de máquina M1): o barramento está liberado. O conteúdo do registrador PC é incrementado em uma unidade, preparando-se para a busca da próxima instrução ou do segundo byte da instrução em andamento.

Estado T3 (ciclo de máquina M1): a instrução é lida da memória e enviada para o registrador de instrução para ser decodificada. No final deste estado a instrução já está decodificada e a unidade de controle inicia o comando das unidades envolvidas na execução da instrução.

Estado T4 (ciclo de máquina M1): nesse período normalmente começa a execução da instrução. Na instrução **MOV A,B**, por exemplo, nesse período ocorre a transferência do conteúdo de B em um registrador temporário, **TMP**, para depois ser transferido para o registrador A, durante o estado T2 da próxima instrução.

Estado T1 (ciclo de máquina M2): o conteúdo do registrador PC (contador de programa) é colocado no barramento e indica o endereço de leitura da memória para obtenção do código de operação (opcode) da instrução. Nesse ciclo de máquina M2 o conteúdo buscado pode ser o opcode da próxima instrução ou o segundo byte da instrução em andamento. Os estados T1 e T2 são os únicos estados onde pode ocorrer sobreposição entre duas instruções. Durante o período de busca da nova instrução a instrução anterior é encerrada. É a sobreposição entre a etapa de busca de uma instrução e a etapa de execução da próxima instrução (SBE – Sobreposição Busca Execução).

Estado T2 (ciclo de máquina M2): o barramento está liberado. O conteúdo do registrador PC é incrementado em uma unidade, preparando-se para a busca da próxima instrução ou do terceiro byte da instrução em andamento. Nesse período, o barramento é usado para encerramento da instrução anterior.

Estado T3 (ciclo de máquina M2): a instrução é lida da memória e enviada para o registrador de instrução para ser decodificada. No final deste estado a instrução já está decodificada e a unidade de controle inicia o comando das unidades envolvidas na execução da instrução.

2.4.2 Microcontrolador 8051

A busca e execução da grande maioria das instruções do 8051 ocorre em um ou dois ciclos de máquina. As exceções são as instruções de divisão e multiplicação que são executadas em 4 ciclos de máquina. Cada ciclo de máquina é composto por 6 estados (S1 a S6), cada um equivalente a 2 períodos do oscilador. Assim, cada ciclo de máquina corresponde a 12 períodos do oscilador, ou período de clock. Se o cristal oscilador for de 12 MHz, 1 ciclo de clock corresponde a

$$T_{clock} = \frac{1}{f_{clock}} = \frac{1}{12MHz}$$

E o ciclo de máquina é:

$$T_{(ciclo\ de\ máquina)} = 12 * T_{(clock)} = 1 \mu s$$

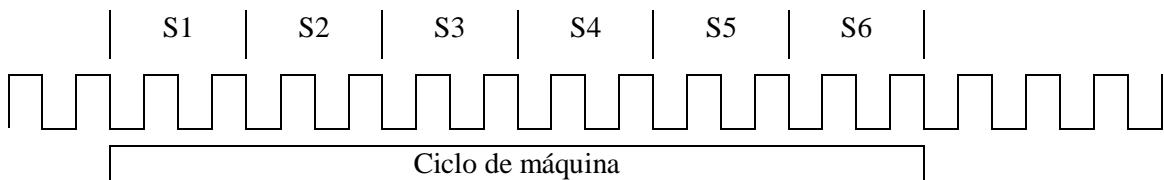


Fig. 2.9 – Ciclo de máquina do 8051

Operações em cada estado do ciclo de máquina:

Estado S1: a próxima instrução é buscada na ROM, colocada no barramento principal e encaminhada para o registrador IR.

Estado S2: a instrução é decodificada e o PC é incrementado.

Estado S3: os operandos da instrução são preparados

Estado S4: os operandos são enviados para os registradores temporários TMP1 e TMP2, na entrada da ULA

Estado S5: a ULA executa a instrução

Estado S6: o resultado da ULA é colocado no barramento principal e encaminhado para o registrador final.

A Tabela 2.3 mostra algumas instruções e o número de ciclos de máquina necessários para sua busca e execução. V1 é um endereço de desvio dentro da faixa de endereços permitidos. A seção de ajuda do simulador PEQui apresenta a codificação de cada instrução e o número de ciclos de clock para sua execução.

Tabela 2.3 – Algumas instruções e o número de ciclos de máquina

Instrução	Número de ciclos de máquina	Instrução	Número de ciclos de máquina
MOV A,#50H	1	MOV A,R1	1
MOV 10H,#30H	2	MOV DPTR,#1000H	2
DJNZ R1,V1	2	CJNE A,#40H,V1	2
JMP V1	2	RLC A	1
MUL AB	4	DIV AB	4

Uma frequência muito comum para o cristal oscilador no uso do 8051 é 11,0592 MHz. Nesse caso, o ciclo de máquina é de 1,085 µs.

2.4.3 Microprocessador 8086/8088

A execução de uma instrução no microprocessador 8086/8088 é feita em duas etapas: a etapa de busca e a etapa de execução propriamente dita. A diferença com relação ao 8085 é que no 8086 as etapas de busca e execução são independentes. A Unidade de Interfaceamento com o Barramento (**BIU**) é responsável pela busca das instruções e empilhamento das mesmas em uma fila (queue). A Unidade de Execução (**EU**) executa as instruções da fila. A sequência é:

- A **BIU** coloca o conteúdo do **IP** (que é somado ao registrador **CS**) no barramento para efetuar a busca de instrução;
- O registrador **IP** é incrementado (aponta para a próxima instrução);
- A instrução lida é passada para a fila;
- A **EU** pega a primeira instrução da fila;
- Enquanto a **EU** executa esta instrução a **BIU** faz uma nova busca de instrução para preencher a fila.
- Se a instrução a ser executada pela **EU** for muito demorada a **BIU** preenche toda a fila.

Há duas situações em que as instruções contidas na fila não são aproveitadas:

- Na execução de instruções de desvio. Neste caso a fila é descartada (ou seja, é sobreescrita);
- Quando a instrução faz referência à memória.

Foi dito que no início do ciclo de busca o conteúdo do **IP** (Ponteiro de Instrução) é somado ao conteúdo do registrador **CS** para a formação do endereço absoluto necessário para a busca da instrução.

Não é uma soma direta, mas a combinação de dois registradores de 16 bits para formar um endereço de 20 bits. Esse processo é denominado de segmentação. O registrador **CS** aponta para o início de uma área de memória dentro da região de 1 MB (20 bits $\rightarrow 2^{20} = 1.048.596$ posições diferentes de memória) e o registrador **IP** é usado como offset dentro da região com início definido por **CS**. Assim, como o registrador **IP** é de 16 bits, tem-se uma região de 64 kB de memória que pode ser acessada por esse conjunto de registradores **CS:IP**. A Fig. 2.10 ilustra a segmentação da memória em regiões e a Fig. 2.11 ilustra o processo de formação do endereço absoluto através da segmentação. A formulação usada na determinação do endereço absoluto é:

$$\text{Endereço absoluto} = \text{Conteúdo do Registrador de Segmento} \times 16 + \text{Conteúdo do Registrador de Offset}$$

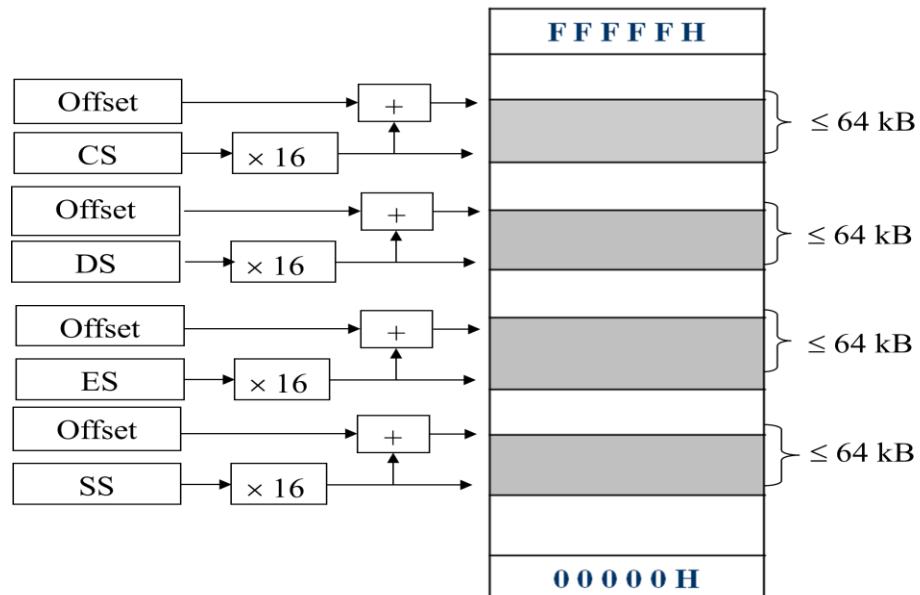


Fig. 2.10 – Segmentação da memória em regiões

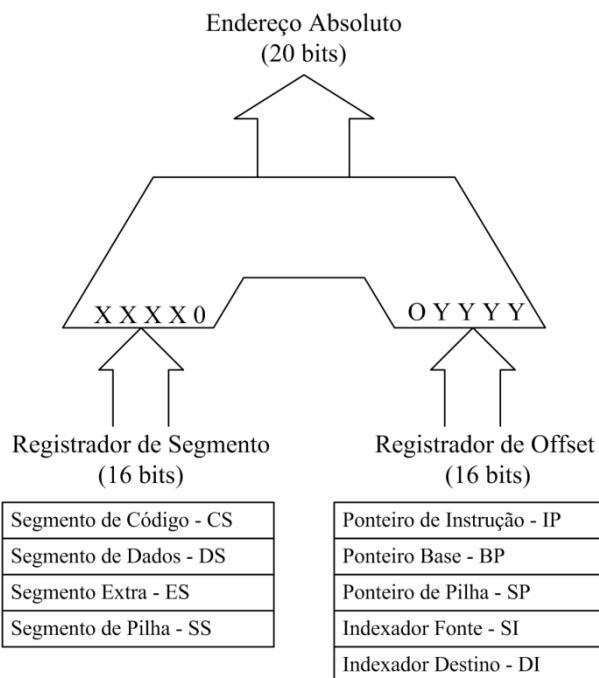


Fig. 2.11 – Formação do endereço absoluto para o 8086/8088

Exemplos de formação de endereço absoluto a partir dos registradores de segmento e offset:

- Exemplo 1: Segmento CS = 2000H; Offset IP = 2000H
 Representação: CS:IP = 2000H:2000H
 Endereço Absoluto (ou Físico) = 20000H + 02000H = 22000H
- Exemplo 2: Segmento SS = 4000H; Offset SP = 2000H
 Representação: SS:SP = 4000H:2000H
 Endereço Absoluto (Físico) = 40000h + 02000h = 42000h

A segmentação da memória tem a vantagem de permitir o armazenamento de diferentes tipos de informação (códigos, dados, pilha...) em diferentes áreas da memória, possibilitando a manipulação de diferentes conjuntos de dados, por exemplo, em um ambiente multitarefa, onde um programa atende a várias entradas de dados. Outra vantagem é a possibilidade de armazenamento e realocação de programas dentro da memória que vai de (00000 H a FFFFF H).

2.5 Formato das Instruções

Cada microprocessador apresenta um conjunto de instruções com formatos pré-definidos. A seguir é mostrada uma forma geral das instruções do 8085, 8051, 8086/8088 e PIC.

2.5.1 Formato das Instruções do 8085

O microprocessador 8085 apresenta instruções com três formatos diferentes: instruções de 1, 2 ou 3 bytes, como ilustrado na Tabela 2.4.

Tabela 2.4 – Formato geral das instruções do 8085

Tipo de instrução	Características	Exemplos
1 byte	O byte da instrução é o próprio código de operação (opcode)	MOV A,C ADD B RLC DCR C
2 bytes	O primeiro byte é o opcode e o segundo byte é o dado de 8 bits necessário para a instrução	MVI A,35H ADI 05H ORI 01H
3 bytes	O primeiro byte é o opcode; o segundo e o terceiro bytes correspondem a um dado de 16 bits	LDA 2030H STA 2040H LXI H, 2080H

A Tabela 2.5 apresenta alguns exemplos de instrução com suas respectivas codificações. Observar que nas instruções de 3 bytes o byte menos significativo da instrução precede o byte mais significativo na memória.

Tabela 2.5 – Exemplos de instruções do 8085 e códigos assembly

Instrução	1º byte (opcode)	2º byte (menos significativo)	3º byte (mais significativo)	Código assembly (HEX)
MOV A,C	79 H			79
ADD B	80 H			80
RLC	07 H			07
MVI A,35H	3E H	35 H		3E 35
ADI 05H	C6 H	05 H		C6 05
ORI 01H	F6 H	01 H		F6 01

LDA 2030H	3A H	30 H	20 H	3A 30 20
STA 2050H	32 H	50 H	20 H	32 50 20
LXI H,2080H	21 H	80 H	20 H	21 80 20

2.5.2 Formato das Instruções do 8051

O microcontrolador 8051 também apresenta instruções com três formatos de 1, 2 e 3 bytes, mas há diferenças de codificação para cada tipo, como mostrado na Fig. 2.12.

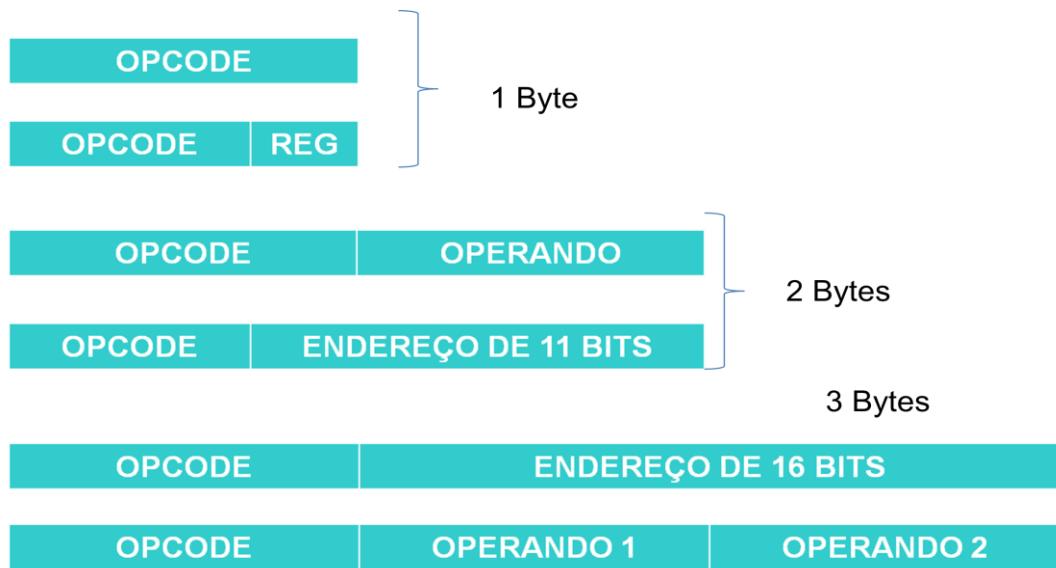


Fig. 2.12 – Formato das instruções do 8051

Nas Tabelas 2.6, 2.7 e 2.8 são mostrados alguns exemplos de instruções respectivamente de 1, 2 e 3 bytes.

Tabela 2.6 – Codificação de instruções de 1 byte do 8051

Instrução	Codificação	Código Assembly (H)
RLC A	0 0 1 1 : 0 0 1 1	33
CLR A	1 1 1 0 : 0 1 0 0	E4
CLR C	1 1 0 0 : 0 0 1 1	C3
ADD A,Rn	0 0 1 0 : 1 r r r (rrr = 000 a 111)	2X (X = 8 a F)
ADD A,R0	0 0 1 0 : 1 0 0 0	28
ADD A,R1	0 0 1 0 : 1 0 0 1	29
MOV A,Rn	1 1 1 0 : 1 r r r (rrr = 000 a 111)	EX (X = 8 a F)
MOV A,R0	1 1 1 0 : 1 0 0 0	E8

Tabela 2.7 – Codificação de instruções de 2 bytes do 8051

Instrução	Codificação	Código Assembly (H)
ADD A,#dado	0 0 1 0 : 0 1 0 0 d7 d6 d5 d4 : d3 d2 d1 d0	24 XX
ADD A,#35H	0 0 1 0 : 0 1 0 0 0 0 1 1 : 0 1 0 1	24 35
MOV A,#dado	0 1 1 1 : 0 1 0 0 d7 d6 d5 d4 : d3 d2 d1 d0	74 XX
MOV A,#35H	1 1 1 0 : 0 1 0 0 0 0 1 1 : 0 1 0 1	74 35
SJMP Relativo	1 0 0 0 : 0 0 0 0 r7 r6 r5 r4 : r3 r2 r1 r0	80 XX

SJMP 80H (*)	1 0 0 0 : 0 0 0 0 0 0 1 0 : 1 1 1 0	80 2E
AJMP end 11 bits	a10 a9 a8 0 : 0 0 0 1 d7 d6 d5 d4 : d3 d2 d1 d0	XX XX
AJMP 620 H	1 1 0 0 : 0 0 0 1 0 0 1 0 : 0 0 0 0	C1 20

* Supondo que a instrução **SJMP 80H** esteja na posição 50 H, após a execução da mesma o contador de programa **PC** estará apontando para a posição 52 H, uma vez que esta instrução tem 2 bytes. Assim, o endereço relativo na codificação da instrução é 80 H – 52 H = 2E H.

Um endereço de 11 bits pode acessar 2048 posições diferentes de memória ($2^{11} = 2048$). Assim, o endereço de 11 bits varia de 0 a 2047 ou de 000 H a 7FF H. Para o endereço 620 H tem-se:

$$620 \text{ H} = 0 1 1 0 | 0 0 1 0 | 0 0 0 0 \text{ b} \rightarrow a10 = 1, a9 = 1 \text{ e } a8 = 0 \rightarrow$$

Daí: a10 a9 a8 0: 0 0 0 1 → 1 1 0 0 : 0 0 0 1 = C1 H (primeiro byte da instrução)

E o segundo byte da instrução é 20 H.

Tabela 2.8 – Codificação de instruções de 3 bytes do 8051

Instrução	Codificação	Código Assembly (H)
LCALL End.16 bits	0 0 0 1:0 0 1 0 a15 a14 a13 a12:a11 a10 a9 a8 a7 a6 a5 a4:a3 a2 a1 a0	12 XX XX
LCALL 0100 H	0 0 0 1:0 0 1 0 0 0 0 0:0 0 0 1 0 0 0 0:0 0 0 0	12 01 00
LJMP End.16 bits	0 0 0 0:0 0 1 0 a15 a14 a13 a12:a11 a10 a9 a8 a7 a6 a5 a4:a3 a2 a1 a0	02 XX XX
LCALL 0125 H	0 0 0 0:0 0 1 0 0 0 0 0:0 0 0 1 0 0 1 0:0 1 0 1	02 01 25
DJNZ direto,relativo	1 1 0 1:0 1 0 1 a7 a6 a5 a4 : a3 a2 a1 a0 r7 r6 r5 r4 : r3 r2 r1 r0	D5 XX XX
DJNZ 01H,\$	1 1 0 1:0 1 0 1 0 0 0 0:0 0 0 1 1 1 1 1:1 1 0 1	D5 01 FD

Na instrução **DJNZ 01H,\$**, na Tabela 2.8, o registrador 01 H é decrementado em 1 e, caso o resultado não seja zero, o programa volta para o início da instrução. O valor FD H da codificação (-3 na notação de complemento de 2) é o valor que é adicionado ao endereço atual (PC após a execução desta instrução), para obter o endereço de desvio. Como a instrução é de 3 bytes, adicionando-se -3 ao endereço no final da instrução, volta-se para o início.

2.5.3 Formato das Instruções do 8086/8088

As instruções de transferência de dados do microprocessador 8086/8088 possuem o formato mostrado na Fig. 2.13. Ela é composta de 4 bytes, onde o primeiro byte traz o opcode e a definição com relação a manipulação de byte ou Word.

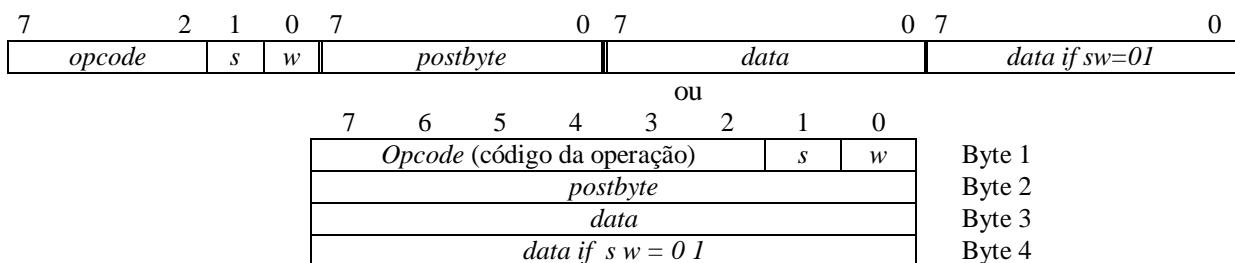


Fig. 2.13 – Formato das instruções de transferência de dados do 8086/8088

As Tabelas 2.9 e 2.10 mostram, respectivamente, o efeito dos bits *s* e *w* na operação e a subdivisão do segundo byte da instrução (*postbyte*). Se a instrução manipular byte, o último da instrução não é utilizado. Só haverá dados nesse byte se a instrução manipular word, o que é definido através de *s* = 0 e *w* = 1.

Tabela 2.9 – Bits de definição do tipo de dado

<i>s</i>	<i>w</i>	Efeito
----------	----------	--------

0	0	A instrução manipula byte
0	1	A instrução manipula word

Tabela 2.10 – Postbyte

7	6	5	4	3	2	1	0
mod	reg			r/m			

Se mod = 1 1, então o operando da instrução é um registrador, que é identificado em r/m. Se a instrução envolver dois registradores, então o campo “reg” identifica o segundo registrador.

2.5.4 Formato das Instruções do PIC 16F628

As instruções do microcontrolador PIC 16F628 são compostas de 14 bits e divididas em quatro tipos: operações com registradores orientadas por byte, operações com registradores orientadas por bit, operações de controle e literal e operações de desvio.

Tipo 1: Operações com registradores orientadas por byte

13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE					d	f (endereço do registrador)							

OPCODE → Código da operação

d = 0 → o destino do resultado é o registrador W, na saída da unidade lógica e aritmética.

d = 1 → o destino do resultado é o registrador f, cujo endereço é fornecido nos 7 primeiros bits.

f → Registrador de endereços de 7 bits (operando)

Tipo 2: Operações com registradores orientadas por bit

13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE					b	f (endereço do registrador)							

OPCODE → Código da operação

b = 0 → endereço de 3 bits

f → Registrador de endereços de 7 bits (operando)

Tipo 3: Operações de controle e literal

13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE					k (endereço do registrador)								

OPCODE → Código da operação

k → Indica um valor constante ou literal

Tipo 4: Operações de desvio

13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE					k (literal)								

OPCODE → Código da operação

k → Indica um valor constante ou literal de 11 bits.

2.6 Modos de Endereçamento e Grupos de Instruções

As Tabelas 2.11 e 2.12 mostram, respectivamente, os modos de endereçamento e os grupos de instruções dos microprocessadores 8085 e 8086/8088.

Tabela 2.11 – Modos de endereçamento

8085		8086/8088	
Endereçamento	Exemplo	Endereçamento	Exemplo

Imediato	MVI A,15H	Imediato	MOV AX,1000H
Por registrador	MOV A,B	Por registrador	MOV AX,BX
Direto	JMP 2005H	Absoluto ou Direto	MOV AX,[1000H]
Indireto por registrador	MOV M,A	Indireto por registrador	MOV AX,[BX]
		Indexado	MOV AX,0100H[BX]
		Baseado	MOV [BX+0100H],AX
		Baseado e indexado	MOV AX,[BX+SI]
		Baseado e indexado com deslocamento	MOV AX,[BX+SI+5]
		Strings	MOVSB

Tabela 2.11 – Grupos de Instruções

8085		8086/8088	
Tipo de Instrução	Exemplo	Tipo de Instrução	Exemplo
Transferência de dados	MVI A,15H MOV A,B	Transferência de dados	MOV AX,1000H MOV AX,BX MOV DL,23H
Aritmético	ADD B SUB C	Aritmético	ADD SI,DX SUB AX,DX
Lógico	ANA B ORI 0FH	Lógico	NOT BX AND CX,DX
Desvio	JMP 2005 H JNZ 2010 H	Desvio	JMP BX
Controle, Pilha e E/S	PUSH PSW IN Porta EI	Controle	CLC STC
		Strings	MOVSB STOSW

2.7 Registradores de Flags do 8085, 8088/86 e 8051

2.7.1 Flags do microprocessador 8085

As flags são bits de um registrador especial que indicam o estado da última operação realizada na Unidade Lógica e Aritmética. A Fig. 2.14 ilustra a presença desse registrador para o caso do microprocessador 8085. São cinco as flags desse microprocessador: flag de carry (CY), flag de paridade (P), flag auxiliar de flag (AC), flag de zero (Z) e flag de sinal (S).

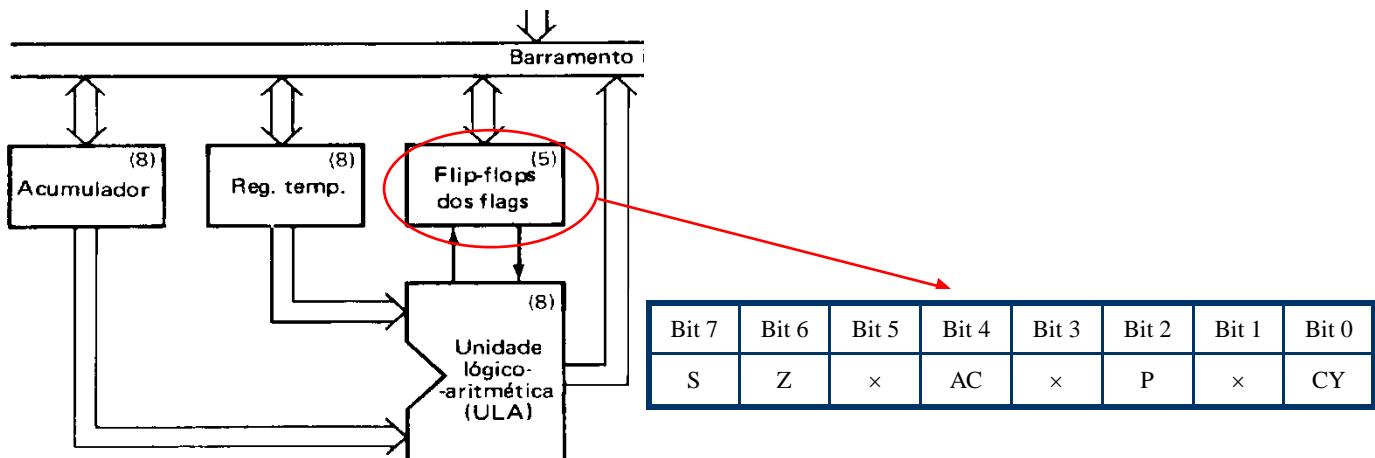


Fig. 2.14 – Registrador de flag do 8085

Flag de Carry: Assume valor 1 quando há transporte do bit 7 para o bit 8 (O Bit 8 é fora do acumulador)

Flag de Paridade: Assume valor 1 quando há uma quantidade par de dígitos 1 no acumulador. Assume valor 0 quando há uma quantidade ímpar.

Flag Auxiliar de Carry: Assume valor 1 quando há transporte do bit 3 para o bit 4.

Flag de Zero: Assume valor 0 para número diferente de zero e 1 para número igual a zero.

Flag de Sinal: Assume valor 0 para número positivo (bit 7 = 0) e 1 para negativo (bit 7 = 1)

Exemplo: A adição dos números decimais 150 e 120 resulta em 270. O maior valor possível de se armazenar em um registrador do 8085 é 255. A adição citada é mostrada a seguir em hexadecimal e binário.

Decimal	Hexa		CY	Acumulador							
			bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
150	96			1	0	0	1	0	1	1	0
120	78			0	1	1	1	1	0	0	0
270	10E		1	0	0	0	0	1	1	1	0

Observando o resultado da operação efetuada, conclui-se:

$CY = 1 \rightarrow$ Houve transporte do bit 7 para o bit 8 (fora do acumulador).

$P = 0 \rightarrow$ Há uma quantidade ímpar de dígitos “1” no resultado da operação e, portanto, a flag de paridade é 0.

$AC = 0 \rightarrow$ Não houve transporte do bit 3 para o bit 4.

$Z = 0 \rightarrow$ O resultado da operação é diferente de zero e, portanto, $Z = 0$.

$S = 0 \rightarrow$ O bit 7 do resultado é zero e, portanto, $S = 0$, indicando um número positivo para operações com sinal.

2.7.2 Flags do microcontrolador 8051

O registrador de flags do microcontrolador 8051 faz parte dos registradores especiais. É o registrador PSW (Program Status Word), mostrado a seguir.

Registrador PSW							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	AC	F0	RS1	RS0	OV	×	P

As flags de carry (CY), auxiliar de carry (AC) e de paridade (P) são idênticas àquelas do microprocessador 8085. A flag F0 é uma flag de uso geral que pode ser usada pelo programador.

A flag de overflow (OV) é uma flag útil em operações com número sinalizado representados na forma de complemento de 2. Ela é setada quando há um carry do bit 7, mas não do bit 6 ou um carry do bit 6, mas não do bit 7. Há duas situações que resultam em OV setado:

- Se a soma de dois números positivos for maior que 7F H e menor que FFH a flag de overflow indica que o número não deve ser interpretado como número negativo.

- Se a soma de dois números negativos (bit 7 = 1) resultar em um número no intervalo de 00 H a 7F H (ou 100 H a 17F H, considerando a flag de carry, que sempre estará presente nessa situação), a flag de overflow indicará que o número não é pra ser interpretado como número positivo.

Em outras palavras, a flag de overflow (**OV**) é o resultado de uma operação **XOR** do carry-in (transporte do bit 6 para o bit 7) com o carry-out (transporte do bit 7 para o bit 8). A Fig. 2. 15 ilustra o carry-in e o carry-out e a Tabela 2.12 mostra as situações possíveis na determinação da flag de overflow.

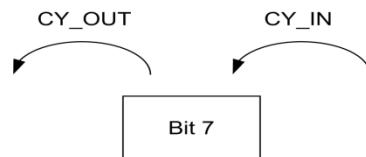


Fig. 2.15 – Carry-in e carry-out

Tabela 2.12 – Determinação da flag OV

CY_IN	CY_OUT	OV = (XOR)
0	0	0
0	1	1
1	0	1
1	1	0

Exemplo 1: Os número 64 H e 2C H estão no intervalo 00 H a 7F H e, portanto, são números positivos nas operações com número sinalizado. No entanto, a soma desses número resulta em 90 H, que está na faixa de 80 H a FF H, ou seja, número negativo nas operações com número sinalizado. Nesse caso, a flag de overflow (OV) é setada, para indicar que o resultado é um número positivo fora da faixa de magnitude dos números positivos. Verifica-se na ilustração a seguir que há transporte do bit 6 para o bit 7, mas não há do bit 7 para bit 8.

Decimal	Hexa		CY	Acumulador							
				bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
100	64			0	1	1	0	0	1	0	0
44	2C			0	0	1	0	1	1	0	0
144	90		0	1	0	0	1	0	0	0	0

Exemplo 2: Se o número 01 H for adicionado ao resultado anterior (90 H) o resultado é 91 H, dentro do intervalo de 80 H a FF H, no entanto, não há flag de overflow (OV = 0) porque, na operação com números sinalizados, foram adicionados um número positivo (01 H) e um número negativo (90 H). Verifica-se na ilustração a seguir que não há transporte do bit 6 para o bit 7, nem do bit 7 para i bit 8, ou seja, ambos, carry-in e carry-out são zero, resultando em OV = 0.

Decimal	Hexa		CY	Acumulador								
				bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
144	90			1	0	0	1	0	0	0	0	0
01	01			0	0	0	0	0	0	0	0	1
145	91		0	1	0	0	1	0	0	0	0	1

Exemplo 3: Os número 80 H e 85 H estão no intervalo 80 H a FF H e, portanto, são números negativos nas operações com número sinalizado. No entanto, a soma desses números resulta em 05 H (ou 105 H, levando-se em conta a flag de carry CY, fora do acumulador). O resultado dentro do acumulador está na faixa de 00 H a 7F H, ou seja, número positivo nas operações com número sinalizado. Nesse caso,

a flag de overflow (**OV**) é setada para indicar que o resultado não é um número positivo. Verifica-se na ilustração a seguir que há transporte do bit 7 para o bit 8, mas não há do bit 6 para bit 7.

Decimal	Hexa		CY	Acumulador							
			bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
128	80			1	0	0	0	0	0	0	0
133	85			1	0	0	0	0	1	0	1
261	1 05		1	0	0	0	0	0	1	0	1

2.7.3 Flags do microprocessador 8086/8088

O registrador de flags do microprocessador 8086/8088 também é denominado de registrador PSW (Program Status Word), como no caso do 8051, mas tem 16 bits, dos quais 9 são usados como flags, sendo 3 bits de controle. Esse registrador é mostrado na Fig. 2.16 seguir.

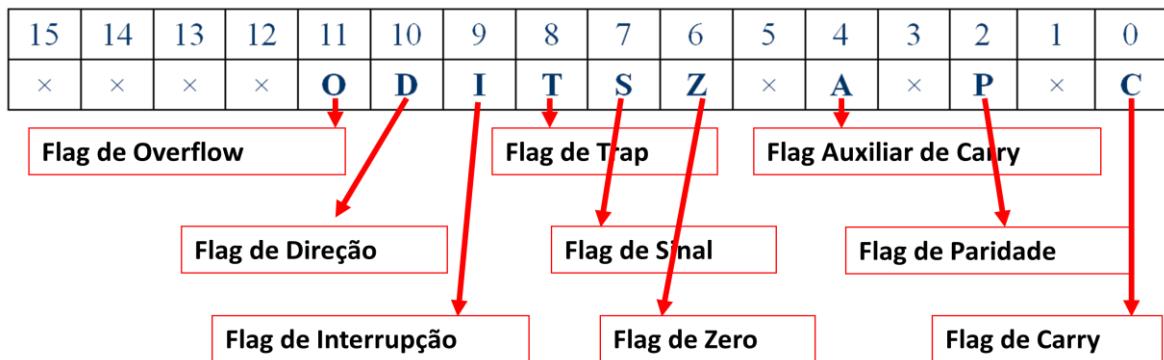


Fig. 2.16 – Flags do microprocessador 8086/8088

C – Flag de carry – reflete o ‘vai um’ do bit mais significativo, nas operações aritméticas (de 8 ou 16 bits). Ele também é modificado por algumas instruções de rotação e deslocamento. Nas operações de subtração (aritmética em complemento dois) o ‘carry’ é invertido e passa a funcionar como ‘borrow’ (emprestímo). Se, após uma operação de subtração, obtém-se $C = 1$, isso indica que não houve ‘borrow’, mas $C=0$, indica que houve ‘borrow’.

P – Flag de Paridade – indica a paridade (par), dos 8 bits menos significativos, do resultado da operação realizada.

$P = 1 \rightarrow$ número par de ‘1’ nos 8 bits menos significativos

$P = 0 \rightarrow$ número ímpar de ‘1’ nos 8 bits menos significativos

A – Flag Auxiliar de Carry – reflete o ‘vai um’ do bit 3, em uma operação de 8 bits.

Z – Flag de Zero – indica se uma operação teve zero como resultado.

$Z = 1 \rightarrow$ se o resultado da operação for igual a zero

$Z = 0 \rightarrow$ se o resultado da operação for diferente de zero

S – Flag de Sinal – é igual ao bit de mais alta ordem do resultado de uma operação aritmética.

$S = 0 \rightarrow$ resultado positivo

$S = 1 \rightarrow$ resultado negativo

O – Flag de Overflow – seu conteúdo é obtido através de uma operação XOR do ‘carry in’ com o ‘carry out’ do bit de mais alta ordem do resultado de uma operação aritmética (Fig. 2.17). Ele indica um ‘overflow’ de magnitude, em aritmética binária com sinal. Indica que o resultado é muito grande para o campo destino.

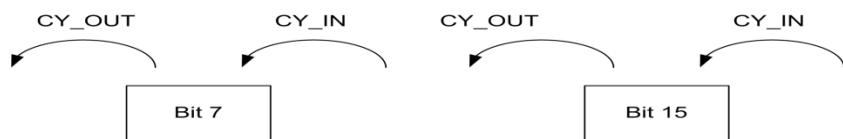


Fig. 2.17 – Transporte do bit 7 e do bit 15

Flags de Controle

T – Flag de Trap (armadilha) – usada para a depuração de programas. Coloca o 8086 no modo passo a passo. Após cada instrução uma interrupção é gerada automaticamente.

I – Flag de Interrupção – habilita ou desabilita a interrupção externa (pedida pelo pino INTR). Ao contrário do 8085, onde as interrupções RST 7.5, RST 6.5 e RST 5.5 podem ser habilitadas/desabilitadas individualmente, no 8086 todas são habilitadas ou desabilitadas ao mesmo tempo. A habilitação/desabilitação individual pode ser feita através do controlador de interrupção 8259.

$I = 1 \rightarrow$ interrupção habilitada $I = 0 \rightarrow$ interrupção desabilitada

D – Flag de Direção – determina se as operações com ‘strings’ vão incrementar ou decrementar os registradores de indexação (SI e DI).

$D = 1 \rightarrow$ SI e DI serão decrementados, ou seja, a ‘string’ será acessada a partir do endereço mais alto em direção ao mais baixo.

$D = 0 \rightarrow$ SI e DI serão incrementados, ou seja, a ‘string’ será acessada a partir do endereço mais baixo em direção ao mais alto.

2.8 Funcionamento da Pilha no 8085, 8051 e 8088/86

A pilha, nos microprocessadores 8085 e 8086/8088 e no microcontrolador 8051, é uma região da memória reservada pelo usuário, para armazenar temporariamente valores que se deseja guardar enquanto o registrador associado é utilizado em outras operações e para guardar endereços de retorno em chamadas de subrotinas e em interrupções.

A pilha é uma estrutura LIFO (Last In First Out), ou seja, o último valor colocado na pilha é o primeiro a ser retirado.

Nos três componentes citados: 8085, 8086/8088 e 8051 a instrução PUSH é usada para guardar um valor na pilha e a instrução POP é usada para retirar um valor da pilha e o registrador SP é usado para apontar o endereço do último valor colocado na pilha.

No 8085 e no 8086/8088 a pilha cresce no sentido decrescente do endereço de memória (Fig. 2.18) e no 8051 (Fig. 2.19) ela cresce no sentido crescente do endereço de memória.

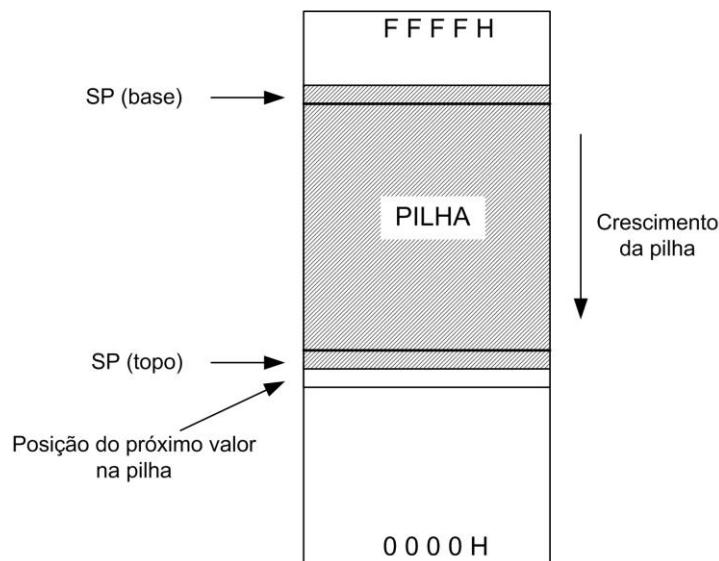


Fig. 2.18 – Pilha no 8085 e no 8086/8088

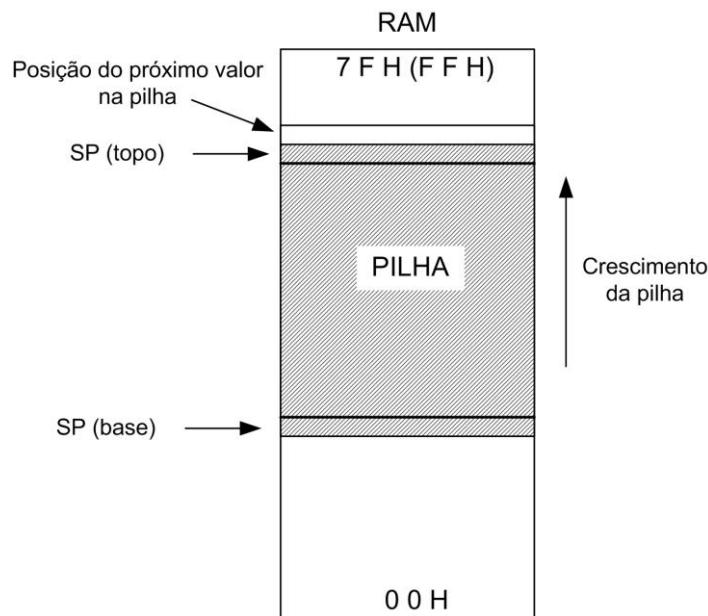


Fig. 2.19 – Pilha no 8051

Embora cada posição de memória tenha apenas 8 bits, as operações com a pilha envolvem a transferência de 16 bits. Quando se trata de endereço, automaticamente o microprocessador armazena os dois bytes do endereço de retorno em duas posições subsequentes da pilha. O byte mais significativo é guardado primeiro. E o byte menos significativo é retirado primeiro da pilha (LIFO). O apontador de pilha SP sempre aponta para o topo da pilha (último valor armazenado).

A sequência de operação para as instruções PUSH e POP é:

PUSH reg16 → guarda conteúdo do registrador de 16 bits na pilha

1. O valor de SP é decrementado em 1
2. O byte mais significativo é armazenado na posição SP – 1
3. O valor de SP é decrementado em 1
4. O byte menos significativo é armazenado na posição SP – 2

POP reg16 → carrega registrador de 16 bits com conteúdo da pilha

1. O conteúdo apontado por SP é copiado para o byte menos significativo
2. O valor de SP é incrementado em 1
3. O conteúdo apontado por SP + 1 é copiado para o byte mais significativo
4. O valor de SP é incrementado em 1

No microprocessador 8085 os registradores de 16 bits (reg16) que podem ser armazenados na pilha são formados por dois registradores de 8 bits, como mostrado na Tabela 2.13, onde são indicados os registradores envolvidos para a instrução PUSH.

Tabela 2.13 – Operações PUSH no 8085

Instrução PUSH	Registradores
PUSH PSW	A e Flags
PUSH B	B e C
PUSH D	D e E
PUSH H	H e L

No microprocessador 8086/8088 os registradores já são de 16 bits e, assim, as instruções PUSH armazena um registrador por vez, como por exemplo, nas instruções PUSH AX e PUSH CS.

Além das instruções PUSH e POP e das interrupções, as instruções CALL e RET manipulam a pilha. Ao ser executada uma instrução CALL o endereço da instrução seguinte é guardado na pilha. A execução da instrução RET faz o processamento retornar para a instrução seguinte à instrução CALL.

No microcontrolador 8051 as instruções PUSH e POP manipulam apenas dados de 8 bits. Por exemplo: PUSH ACC guarda na pilha o conteúdo do acumulador. PUSH DPH guarda na pilha o byte mais significativo do registrador de 16 bits DPTR.

A sequência para as instruções PUSH e POP no 8051 é:

PUSH reg8 → guarda conteúdo do registrador de 8 bits na pilha

1. O valor de SP é incrementado em 1
2. O conteúdo do registrador é armazenado na posição SP + 1

POP reg8 → carrega registrador de 8 bits com conteúdo da pilha

1. O conteúdo apontado por SP é copiado para o registrador
2. O valor de SP é decrementado em 1

3 Microcontrolador 8051

As seções anteriores já apresentaram uma breve introdução ao microcontrolador da família 8051. A partir desta seção é feito o estudo desse microcontrolador visando aplicações diversas, tais como acionamento de motor de passo, motor de corrente contínua e lâmpadas e uso de teclado, display de 7-segmentos, display LCD e sensor de presença.

Ao longo dos últimos anos, na disciplina de microprocessadores, já foram utilizadas várias versões da família 8051. A Tabela 3.1 mostra as versões utilizadas e algumas características das mesmas.

Tabela 3.1: Algumas versões de microcontroladores da família 8051

Versão do Microcontrolador	Memória interna			Frequência de Clock	Número de Temporizadores
	Memória de Programa	Memória de Dados (RAM)	Memória de Dados (EEPROM)		
8031/8032	Não tem	128kB/256kB	Não tem	12 MHz	2 / 3
8051/8052	ROM - 4kB/8kB	128kB/256kB	Não tem	12 MHz	2 / 3
8751/8752	EPROM- 4kB/8kB	128kB/256kB	Não tem	12 MHz	2 / 3
8951/8952	Flash - 4kB/8kB	128kB/256kB	Não tem	24 MHz	2 / 3
AT89S8252	Flash - 4kB/8kB	256kB	2 kB	24 MHz	3
AT89S52	Flash - 4kB/8kB	256kB	Não tem	33 MHz	3

Verifica-se na Tabela 3.1 que as versões com terminação “1” possuem memória de dados (RAM) de 128 bytes, enquanto a capacidade da memória RAM das versões com terminação “2” é de 256 bytes. Todas as versões “2” possuem 8 kB de memória de programa, o dobro das versões “1”.

Quanto aos contadores/temporizadores, as versões “1” possuem 2, enquanto que as versões “2” possuem 3. Todos eles são de 16 bits, podendo trabalhar em quatro modos diferentes: modo de 13 bits, modo de 16 bits, modo único de 8 bits com recarga automática e modo duplo independente de 8 bits. O contador é caracterizado por um clock externo, enquanto o temporizador usa o clock interno do microcontrolador. Essa unidade trabalha de forma independente da CPU e pode ser ativada tanto por software quanto por hardware.

Das versões mostradas na Tabela 3.1, somente uma, AT89S8252, possui memória EEPROM interna, permitindo o armazenamento de até 2 kB de dados, o que dispensaria uma memória EEPROM externa quando o volume de dados a serem guardados é pequeno. A Fig. 3.1 mostra a pinagem de uma versão básica de 40 pinos.

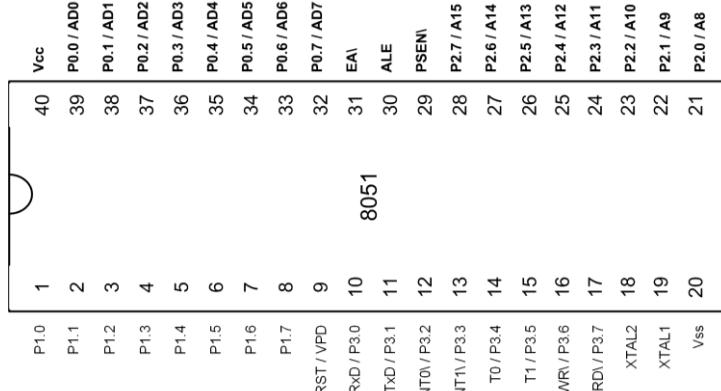


Fig. 3.1: Pinagem do microcontrolador de 40 pinos da família 8051.

As portas paralelas, num total de 4 (quatro), são numeradas de P0 a P3 e são todas de 8 bits. A porta P3 tem dupla função: além de servir como canal de entrada e saída de dados também pode ser usada para a comunicação serial (P3.0 e P3.1 = RxD e TxD), interrupções externas (P3.2 e P3.3 = INT0\ e INT1\), canal de clock para o contador (P3.4 e P3.5 = T0 e T1) e canal de controle de gravação e

leitura de periféricos ($P3.6 = WR\backslash$ e $P3.7 = RD\backslash$). A porta P0 tem uma característica diferente das outras portas: ela é de coletor aberto, o que significa que necessita de um resistor de *pull-up* e pode drenar uma corrente maior que as demais. Outra característica da porta P0 é que ela também é o canal de dados e parte baixa do endereço para operações com periféricos ($AD0 \dots AD7$). A porta P2, além da função de entrada e saída de dados, é o canal da parte alta do endereço nas operações com periféricos ($A0 \dots A7$).

Os pinos 18 (XTAL2) e 19 (XTAL1) são usados para a conexão do cristal oscilador para a geração da frequência de clock (Fig. 3.2), que determina o ciclo de máquina. A frequência de oscilação do cristal deve estar na faixa de 3,5 MHz a 12 MHz, para as versões básicas. Os valores recomendados para os capacitores no caso de oscilador a cristal são entre 20 pF e 40 pF.

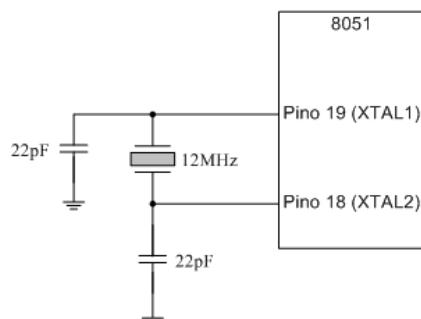


Fig. 3.2: Conexão do cristal oscilador

O pino 9 é usado para reinicialização (reset) do microcontrolador. A Fig. 3.3 mostra o circuito típico para o reset. A colocação de um capacitor de $10\mu F$ em série com um resistor de $8,2\text{ k}\Omega$ conectados ao pino 9 do 8051 garante que o pino 9 fique em nível lógico alto por pelo menos 2 ciclos de máquina (24 períodos do oscilador) na energização do sistema. Com isso, os registradores assumem os valores mostrados na Tabela 3.2, e o 8051 está pronto para iniciar o processamento. O botão em paralelo com o capacitor é usado para o Reset forçado. Para algumas versões da família 8051 não é necessário o resistor de $8,2\text{ k}\Omega$, por já existir um resistor interno.

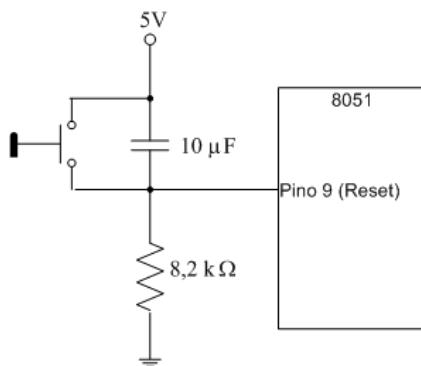


Fig. 3.3: Circuito de Reset do 8051.

Tabela 3.2: Valores dos registradores especiais após o Reset

Registro	Valor	Registro	Valor
PC	0000h	TCON	00h
A	00h	TH0	00h
B	00h	TL0	00h
PSW	00h	TH1	00h
SP	07h	TL1	00h
DPTR	0000h	SCON	00h
P0 - P3	FFh	SBUF	Indeterminado
IP	xxxx0000b	PCON(NMOS)	0xxxxxxxxb
IE	0xx00000b	PCON(CMOS)	0xxx0000b
TMOD	00h		

3.1 Memórias ROM e RAM

Nas versões que possuem memória **ROM**, ela pode ser expandida até o limite de 65.536 bytes (64 kBytes) com uma pastilha externa, ou ainda usar somente uma pastilha externa de 64 kbytes (0000 a FFFF h). O pino EA\ (pino 31) é usado para selecionar a memória externa ou interna. Caso EA\ = 0, o microcontrolador busca o programa na memória ROM externa; caso EA\ = 1, o programa é buscado na memória ROM interna.

O pino PSEN\ é usado para habilitação da leitura da memória externa. Caso PSEN\ = 1 a leitura da memória ROM interna está habilitada; caso PSEN\ = 0, a leitura da memória ROM externa está habilitada. Enquanto é usada a instrução **MOV** para manipulação de códigos na ROM interna, usa-se **MOVC** (MOV Code) para as mesmas operações com a **ROM externa**. A Fig. 3.4 ilustra a composição da memória ROM. São duas opções excludentes: usa-se memória externa para expandir a memória interna ou somente memória externa.

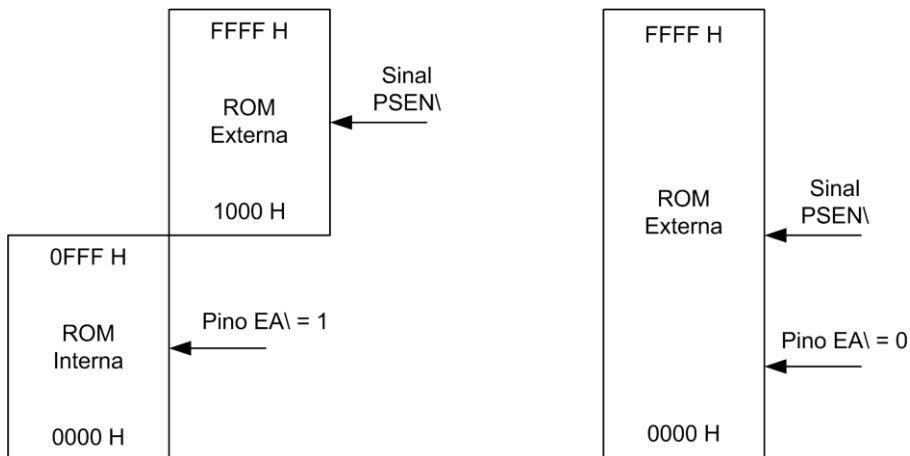


Fig. 3.4: Memórias ROM interna e externa

A memória **RAM**, em qualquer versão, pode ser adicionada em 64 kbytes (0000 a FFFFh), além dos 128 ou 256 bytes de memória interna (Fig. 3.5).

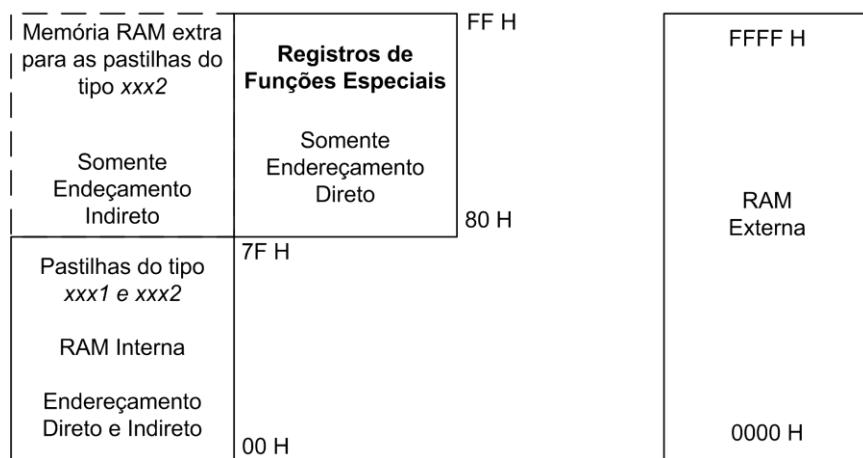


Fig. 3.5: Memórias RAM interna e externa

A memória RAM interna é subdividida em duas regiões básicas:

Endereço de 00 a 7F h (Parte baixa da RAM) – Contém 128 bytes. Todas as versões do 8051 possuem essa região. O acesso pode ser direto ou indireto.

Endereço de 7F a FFh (Parte alta da RAM) – Essa faixa de endereço só está presente nas versões xxx2. Ela contém 128 bytes, cujo acesso é sempre através de **endereçamento indireto**. Essa

faixa de endereços coincide com a faixa de endereços dos registradores especiais. A diferença está no tipo endereçamento para acesso. Os registradores especiais são acessados sempre através de endereçamento direto. A manipulação de dados na memória RAM interna é através da instrução MOV, enquanto que na **RAM externa** é com uso da instrução **MOVX** (MOV eXtern). Os sinais RD\ e WR\ são usados na operação com a memória RAM externa. A Fig. 3.6 mostra a região baixa em detalhes.

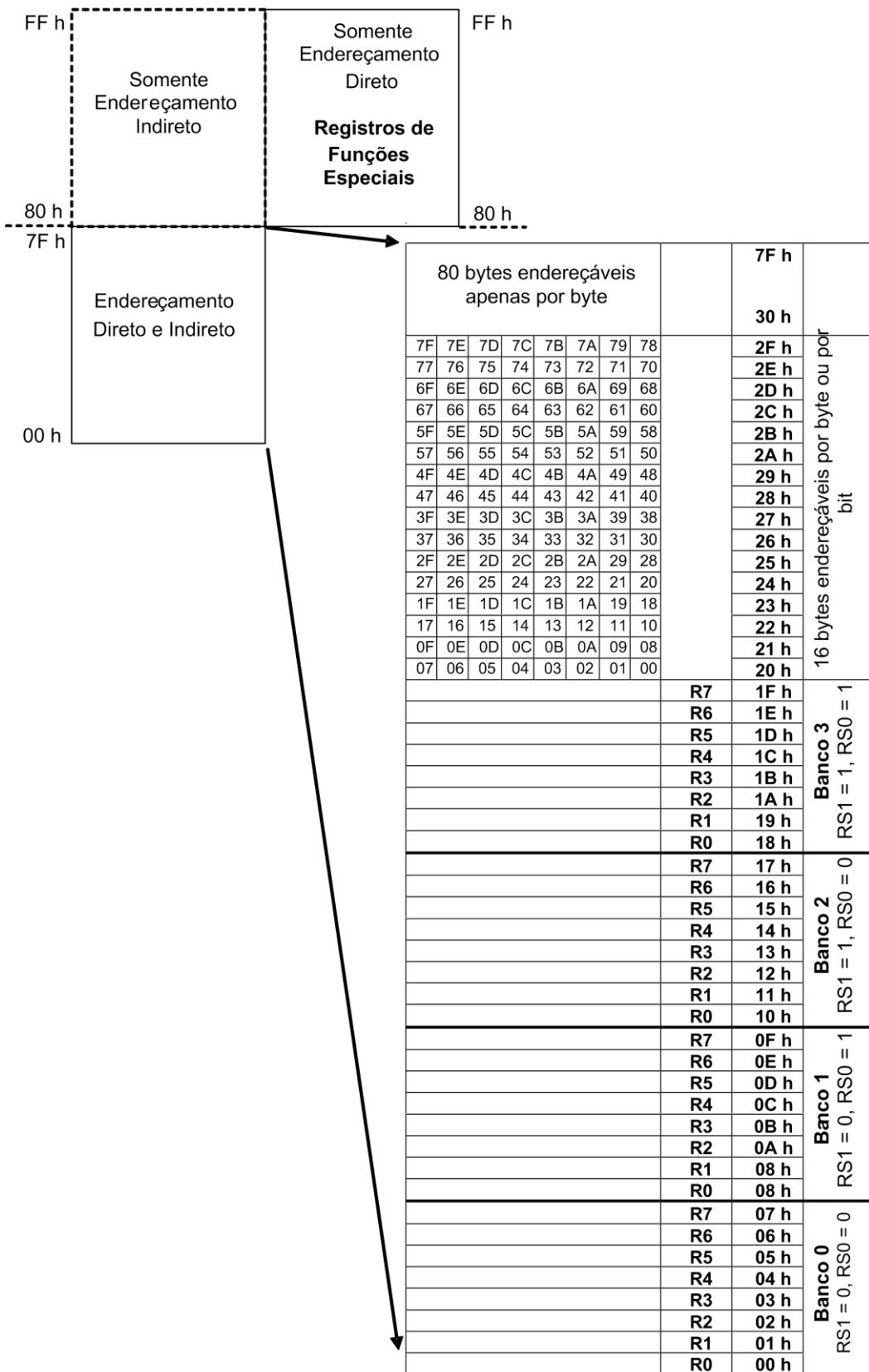


Fig. 3.6: Detalhes da Parte baixa da memória RAM interna

Como já foi mostrado em uma seção anterior, a seleção dos bancos de registradores (Banco 0 a Banco 3) é feita através dos bits RS1 e RS0 do registrador especial **PSW** (Program Status Word). Acima dos registradores de propósito geral tem-se duas regiões distintas: a região mais alta, do endereço 30 h até 7F h, é endereçável apenas por byte; no entanto, a região do endereço 20 h até 2F h é endereçável por byte e por bit, ou seja, os bits dessa região podem ser referenciados pelos seus endereços individuais ou através dos bytes dos quais eles fazem parte, como exemplificado a seguir:

Bit 00h ≡ 20h.0;	bit 01h ≡ 20h.1	...	bit 07h ≡ 20h.7
Bit 08h ≡ 21h.0;	bit 09h ≡ 21h.1	...	bit 0Fh ≡ 21h.7
Bit 10h ≡ 22h.0;	bit 11h ≡ 22h.1	...	bit 17h ≡ 22h.7
		...	bit 7Fh ≡ 2Fh.7

3.2 Os Registradores de Funções Especiais

A Tabela 3.3 é uma repetição da Tabela 2.2. Ela mostra os principais **Registradores Especiais**, que ficam localizados na região de **80h a FFh**. Os registradores dessa região, com endereços de final **0** ou **8**, são endereçáveis por byte ou por bit. Os demais, apenas por byte. A tabela mostra também os bits individuais desses registradores de final 0 e 8.

Tabela 3.3: Principais Registradores Especiais

Registrador	Mnemônico	Endereço	Endereços individuais dos Bits e denominações de alguns bits								
Latch da Porta 0	P0	80 H	87	86	85	84	83	82	81	80	
Apontador de Pilha	SP	81 H									
Apontador de Dados	DPTR	83H – 82H									
LSB do Apontador de Dados	DPL	82 H									
MSB do Apontador de Dados	DPH	83 H									
Controle de Energia	PCON	87 H	SMOD								
Controle do Contador/Temporizador	TC0	88 H	8F	8E	8D	8C	8B	8A	89	88	
			TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
Controle do Modo do Temporizador/Contador	TMOD	89 H	G1	C/T1	M11	M01	G0	C/T0	M10	M00	
LSB do Temporizador/Contador 0	TL0	8A H									
LSB do Temporizador/Contador 1	TL1	8B H									
MSB do Temporizador/Contador 0	TH0	8C H									
MSB do Temporizador/Contador 1	TH1	8D H									
Latch da Porta 1	P1	90 H	97	96	95	94	93	92	91	90	
Controle da Porta Serial	SCON	98 H	9F	9E	9D	9C	9B	9A	99	98	
			SM1	SM2	SM3	REN	TB8	RB8	TI	RI	
Porta de Dados Seriais	SBUF	99 H									
Latch da Porta 2	P2	A0 H	A7	A6	A5	A4	A3	A2	A1	A0	
Habilitador de Interrupção	IE	A8 H	AF	AE	AD	AC	AB	AA	A9	A8	
			EA			ES	ET1	EX1	ET0	EX0	
Latch da Porta 3	P3	B0 H	B7	B6	B5	B4	B3	B2	B1	B0	
Controle de Prioridade da Interrup.	IP	B8 H	BF	BE	BD	BC	BB	BA	B9	B8	
						PS	PT1	PX1	PT0	PX0	
Registrador de Estado do Programa	PSW	D0 H	D7	D6	D5	D4	D3	D2	D1	D0	
			CY	AC	F0	RS1	RS0	OV		P	
Acumulador	ACC ou A	E0 H	E7	E6	E5	E4	E3	E2	E1	E0	
Registrador B	B	F0 H	F7	F6	F5	F4	F3	F2	F1	F0	

Deve ser enfatizado que os registradores especiais ocupam os endereços de **80h** a **FFh**, que coincide com os 128 bytes superiores da RAM interna dos microcontroladores xxx2. A diferença entre o acesso aos Registradores especiais e a parte superior da RAM interna é o tipo de endereçamento. Os **registradores especiais** são acessados sempre por **endereçamento direto**, enquanto a parte superior da **RAM interna** é acessada somente por **endereçamento indireto**.

Exemplos:

MOV A, 80h	Carrega o acumulador com o conteúdo do registrador especial 80 h (Porta P0)
MOV RO, #80h MOV A,@RO	Carrega o acumulador com o conteúdo da posição de memória RAM apontada por R0, ou seja, endereço 80 h da memória superior

Algumas observações:

Apontador de Pilha (Stack Pointer): SP: 81h - Como valor “*default*”, SP aponta para o endereço hexadecimal **07h** da memória RAM interna. Ao contrário do Microprocessador 8085, o endereço do apontador de pilha é incrementado a cada endereço guardado na pilha e o byte menos significativo é guardado primeiro.

Registradores DPH (83h) e DPL (82h) - Equivalem aos registradores H e L do 8085, que juntos formam o par HL. No 8051 DPH e DPL formam o registrador de 16 bits **DPTR**, usado principalmente no manuseio de tabelas.

Registradores dos Temporizadores/Contadores – TH e TL armazenam as partes alta e baixa, respectivamente, da contagem dos temporizadores/contadores; **TMOD (89 H)** define o modo de operação dos temporizadores/contadores e **TCON (88 H)** controla o início e o fim de uma contagem.

Controle da Porta Serial: SCON (98 H) – Registrador que contém todos os bits que definem o modo de operação e o controle da porta serial. **SBUF (99 H)** é o registrador que armazena tanto os dados a serem transmitidos quanto os dados recebidos via serial. Efetivamente há dois registradores de mesmo nome **SBUF** e mesmo endereço (**99 H**), um responsável pela transmissão e outro pela recepção de dados.

3.3 Instruções Gerais do Microcontrolador 8051

As instruções do 8051 podem ser digitadas em maiúsculas ou minúsculas. A seguir são mostradas algumas dessas instruções, com exemplos. O símbolo “#” é necessário para diferenciar dado de registrador. Os dados seguidos de “H” ou “h” estão no sistema hexadecimal; os dados seguidos de “B” ou “b” estão em binário e os dados sem nenhuma indicação estão no sistema decimal.

Instrução	Descrição e exemplos
MOV A,#DADO	Carrega o acumulador com o valor de “dado”. MOV A,#25 → Carrega acumulador com valor decimal 25 (19 hexadecimal) MOV A,#15H → Carrega acumulador com valor hexadecimal 15H MOV A,#01011001b → Carrega acumulador com o binário equivalente a 59H
MOV A,DIRETO	Copia no acumulador o conteúdo do registrador cujo endereço é “direto”. MOV A,15H → Copia no acumulador o conteúdo do registrador R5 (15H), do banco 2.
MOV A,REG	Copia no acumulador o conteúdo do registrador “reg”, sendo reg = R0, R1, ..., R7, do banco de registradores que estiver ativo. MOV A,R6 → Copia no acumulador o conteúdo do registrador R6.

Instrução	Descrição e exemplos
MOV dir2,dir1	Copia no registrador cujo endereço é “dir2” o conteúdo do registrador cujo endereço é “dir1”. MOV 02H,05H → Copia em R2 (02H) o conteúdo do registrador R5 (05).
MOV R0,#20H MOV @R0,#55H	Carrega registrador R0 com valor 20h Copia o valor 55h na posição apontada pelo registrador R0, ou seja, endereço 20H, que é a primeira posição acima do banco de registradores.
MOV DPTR,#200H MOVC A,@A+DPTR	Carrega registrador de 16 bits “dptr” com valor 200H Carrega acumulador com o conteúdo da posição apontada por “a + dptr”. Se, por exemplo, A = 04H, então carrega acumulador com o conteúdo da posição 204H.
ADD A,REG	Adiciona o conteúdo do registrador “reg” ao conteúdo do acumulador. ADD A,R1 → Se A = 07 H e R1 = 03 H, então, após a instrução, a = 0AH.
ADD A,DIRETO	Adiciona o conteúdo do registrador de endereço “direto” ao conteúdo do acumulador: A = A + (direto) ADD A,10H → Se A = 07 H e 10H = 03 H, então, após a instrução, A = 0AH.
ADD A,#DADO	Adiciona ao conteúdo do acumulador o valor “dado”: A = A + dado ADD A,#04h → Se a = 07 H, então, após a instrução, A = 0BH.
ADD A,@Rn	Adiciona ao conteúdo do acumulador o conteúdo da posição apontada por Rn. A = A + ((Rn)). MOV R0,#20h ADD A,@R0 → Se A = 07 H e registrador 20H = #03H, então, após a instrução, A = 0AH.
SUBB A,#DADO	Subtrai o conteúdo do acumulador do “DADO”. A = A – DADO. SUBB A,#05H → Se A = 07 H, então, após a instrução, A = 02 H.
RL A	Rotaciona o conteúdo do acumulador para a esquerda (rotate left). Por exemplo, se originalmente A= 21 H (0010 0001b), após a instrução, tem-se: A = 42 H (0100 0010b).
RR A	Rotaciona o conteúdo do acumulador para a direita (rotate right). Por exemplo, se originalmente A= 8C H (1000 1100b), após a instrução, tem: A = 46 H (0100 0110b).
INC REG	Incrementa conteúdo do registrador “reg”. Por exemplo, se R1 = 05H, então INC R1 resulta em R1 = 06 H.
DEC REG	Decrementa conteúdo do registrador “reg”. Por exemplo, se R2 = 0B H, então DEC R2 resulta em R2 = 0A H.
CPL A	Complementa o conteúdo do acumulador. Por exemplo, se originalmente, A = 55 H, então, após a instrução, A = AA H.
SWAP A	Faz a troca dos nibbles do acumulador, ou seja, o nibble mais significativo passa a ocupar os quatro primeiros bits do acumulador e o nibble menos significativo passa a ocupar os quatro últimos bits. Por exemplo, se originalmente, A = 35 H, após a instrução, A = 53 H.
DA A	Faz o ajuste decimal do acumulador. Adiciona “6” ao dígito que esteja no intervalo de A a F. Por exemplo, se originalmente A = 7A H, após a instrução torna-se A = 80 H.
MUL AB	Multiplica o conteúdo de A pelo conteúdo de B. O resultado está em B A. O resultado da multiplicação é um número de 16 bits, por isso precisa de dois registradores para o resultado. MUL AB → se A = 25 H e B = 30 H, após a instrução, tem-se: B = 06 H e A = F0 H, pois o resultado da multiplicação é: 6F0 H

Instrução	Descrição e exemplos
DIV AB	Divide o conteúdo de A pelo conteúdo de B. A recebe o quociente e B o resto. DIV AB → se A = CAH (202) e B = 19H (25), após a instrução, tem-se: A = 08H e B = 02, pois a divisão em decimal (202/25) resulta em quociente 8 e resto 2.
ANL A,#DADO	Faz uma operação AND entre acumulador e DADO. A = A (AND) DADO. ANL A,#0FH → se originalmente A = 35H, após a instrução torna-se: A = 05H.
ORL A,#DADO	Faz uma operação OR entre o acumulador e DADO. A = A (OR) DADO. ORL A,#20H → se originalmente A = 07H, após a instrução torna-se: A = 27H.

3.4 Instruções de Comparação, Decisão e de Desvio

As instruções desta seção são de desvio incondicional e desvio que depende do estado de flags.

Instrução	Descrição e exemplos
SJMP DESVIO	Desvio incondicional curto (Short Jump) relativo. Pula até 127 bytes para a frente e até 128 bytes para trás.
AJMP DESVIO	Instrução de desvio para distâncias correspondentes a até 2048 bytes. Endereço de 11 bits.
LJMP DESVIO	Desvio incondicional longo, para qualquer posição da memória de programa. Endereço de 16 bits.
JNZ DESVIO	Instrução de desvio condicional: Jump IF Not Zero. Pula para “desvio” se a operação anterior não resultar em zero. Verifica automaticamente a flag de zero.
LCALL SUBROT	Chamada de subrotina. Desvia para o endereço onde a subrotina está localizada. Ao encontrar a instrução RET, retorna para a instrução que vem logo após a chamada de subrotina.
JC DESVIO	Desvio condicional para a posição indicada por “desvio”. Desvia se a flag de CARRY estiver setada.
JNC DESVIO	Desvio condicional para a posição indicada por “desvio”. Desvia se a flag de CARRY não estiver setada.
DJNZ REG,DESVIO	Decrementa registrador “reg” e pula para a posição “desvio” se o resultado não for zero. É uma combinação das instruções “DEC” e “JNZ” do microprocessador 8085. MOV R5,#10 V1: DJNZ R5,V1 → O registrador R5 é decrementado até tornar-se zero
CJNE A,#DADO,V1	Compara conteúdo do acumulador com “dado” e pula para a posição “V1” se não forem iguais. MOV A,#00H V1: INC A CJNE A,#20H,V1 → Compara o conteúdo de A com 20 hexadecimal e, caso não seja igual pula para V1 para incrementar A. Quando for igual, pula para a próxima linha.

A diferença entre LJMP e SJMP é que a primeira instrução refere-se a um endereço de 16 bits e é codificada em 3 bytes: o opcode e os dois bytes de endereço. A instrução SJMP é codificada em 2 bytes sendo o segundo byte o valor que deve ser adicionado à posição atual do apontador de programa PC, para determinar o endereço de desvio. O exemplo a seguir mostra um programa e sua codificação, com as instruções SJMP e LJMP. No programa mostrado o código de **SJMP V1** é **8009**, onde **80H** é o opcode da instrução e **09H** é o valor que deve ser adicionado ao contador de programa PC para indicar a próxima instrução a ser executada. Após a execução de **SJMP V1** o valor de PC é **0037H**. Adicionando 09H chega-se a **0040H**, endereço da instrução **ADD A,#53H**.

O código da instrução **LJMP INICIO** é **020030**, onde **02H** é o opcode da instrução e **0030H** é o endereço de desvio, ou seja, a posição de início do programa, para execução da instrução **MOV A,#35H**.

Endereço	Codificação	Rótulo	Mnemônico
			\$MOD51
0000			ORG 00H
0000	020030		LJMP INICIO
0030			ORG 30H
0030	7435	INICIO:	MOV A,#35H
0032	75F045		MOV B,#45H
0035	8009		SJMP V1
0037			
0040			ORG 40H
0040	2453	V1:	ADD A,#53H
0042	020030		LJMP INICIO
			END

3.5 Operações com bit

As instruções mostradas a seguir são algumas das instruções usadas nas operações com bit, ao invés de byte. O bit pode ser de um registrador especial (daqueles que permitem controle individual por bit) ou da região da memória RAM que vai do endereço 20H até 2FH.

Instrução	Descrição e exemplos
JB BIT,DESVIO	Desvia para a posição “desvio”, caso o “bit” esteja setado. JB LIGADO,DESLIGA → Se o bit <i>ligado</i> = 1, então o programa desvia para a posição “desliga”.
JNB BIT,DESVIO	Desvia para “desvio”, caso o “bit” NÃO esteja setado. JNB LIGADO,LIGA → Se o bit <i>ligado</i> = 0, então o programa desvia para a posição “liga”.
SETB BIT	Seta o “bit”. SETB LIGADO → Torna o bit “ligado” igual a 1.
CLR BIT	Limpa o “bit” CLR LIGADO → Torna o bit “ligado” igual a zero

3.6 Diretivas de programação

Durante a programação em assembly, são necessárias algumas informações para o compilador. Essas informações não são compiladas, mas apenas informam sobre variáveis, sobre posicionamento na memória e sobre dados. As principais diretivas são dadas a seguir:

org endereço → Informa ao compilador o endereço onde deve ser armazenada a próxima instrução.
 Exemplo:
 org 30 H
 mov sp,#2Fh → Esta instrução será armazenada na posição 30 H da memória ROM.

variável equ ender. reg. → informa ao compilador que a “variável” *equivale* ao registrador cujo endereço é “ender. reg”.

Exemplo:

velocidade equ 05H → Esta diretiva diz ao compilador que as operações com a variável “velocidade” equivalem às operações com o registrador R5 do banco 0 (endereço do registrador: 05 H). Por exemplo: mov velocidade,#52H equivale à instrução mov R5,#52H.

variável bit ender. bit → informa ao compilador que a “variável” é do tipo bit e será armazenada no endereço dado por “ender.bit”.

Exemplo:

sentido bit 00H → Esta diretiva diz ao compilador que a variável “sentido” é do tipo bit e será armazenada no endereço 00H da região acima dos bancos de registradores. O endereço do **bit 00H** corresponde ao primeiro endereço dessa região, ou seja, posição **20.0H**.

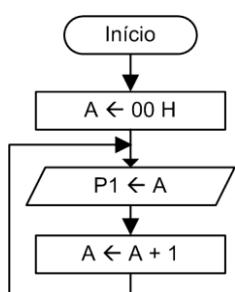
db byte → Esta diretiva diz ao compilador que o byte a seguir é um dado e não uma instrução.

Exemplo:

db 45H → O valor 45H é tratado como um dado, não como uma instrução.

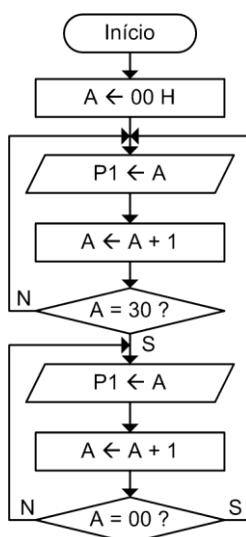
3.7 Programas Exemplos

(a) Programa que faz uma contagem hexadecimal crescente ininterrupta de 00 H a FF H.



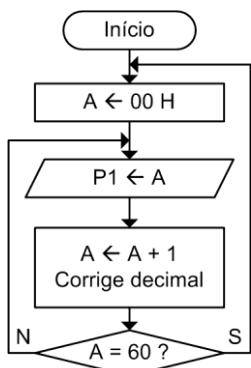
Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	A próxima instrução estará no endereço 00h
	LJMP INICIO	Pula para o endereço indicado com o rótulo ‘inicio’
	ORG 30H	A próxima instrução estará no endereço 30h
INICIO:	MOV A,#00H	Carrega acumulador com valor 00h
VOLTA:	MOV P1,A	Transfere para a porta P1 o conteúdo do acumulador
	INC A	Incrementa o conteúdo do acumulador
	SJMP VOLTA	Pula para o endereço indicado pelo rótulo ‘volta’
	END	Instrução obrigatória no fim de todo programa

(b) Programa que faz uma contagem hexadecimal ininterrupta na sequência 00 → 30 h → 00 h.



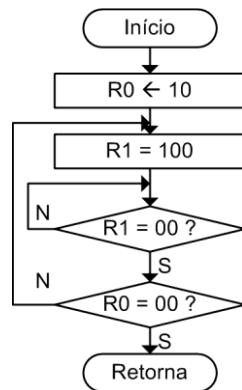
Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	A próxima instrução estará no endereço 00h
	LJMP INICIO	Pula para o endereço indicado com o rótulo ‘inicio’
	ORG 30H	A próxima instrução estará no endereço 30h
INICIO:	MOV A,#00H	Carrega acumulador com valor 00h
VOLTA:	MOV P1,A	Transfere para a porta P1 o conteúdo do acumulador
	INC A	Incrementa o conteúdo do acumulador
	CJNE A,#30H,VOLTA	Compara conteúdo do acumulador com “30h”. Caso não seja igual, “volta”
VOLTA2:	MOV P1,A	Transfere para a porta P1 o conteúdo do acumulador
	DJNZ ACC,VOLTA2	Decrementa conteúdo do acumulador e vai para “volta2” se não for “zero”
	SJMP VOLTA	Pula para “volta”. Não precisa usar “LJMP” porque a distância é curta.
	END	Instrução obrigatória no fim de todo programa

(c) Programa que faz uma contagem decimal ininterrupta de 0 a 59 h.



Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	A próxima instrução estará no endereço 00h
	LJMP INICIO	Pula para ‘início’
	ORG 30H	A próxima instrução estará no endereço 30h
INICIO:	MOV SP,#2FH	Carrega apontador de pilha “SP” com valor 2Fh
	MOV A,#00H	Carrega acumulador com valor 00h
VOLTA:	MOV P1,A	Transfere conteúdo do acumulador para porta P1
	ADD A,#01H	Adiciona 01 ao conteúdo do acumulador
	DA A	Faz o ajuste decimal do conteúdo do acumulador
	CJNE A,#60H,VOLTA	Compara conteúdo do acumulador com 60h. Caso seja diferente, “volta”
	SJMP INICIO	Pula para “início”
	END	Instrução obrigatória no fim de todo programa

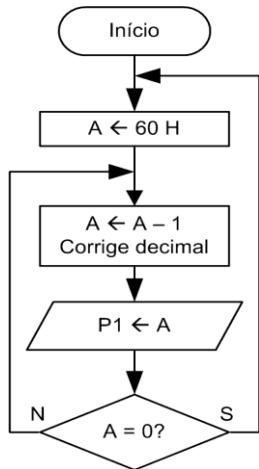
A contagem do programa acima não apresenta qualquer atraso de tempo, o que pode dificultar a visualização da contagem durante a simulação e durante a implementação em laboratório. Dessa forma, é apresentada abaixo uma subrotina de atraso de tempo implementada com dois registradores. É suposto que a frequência do cristal oscilador é de 12 MHz, o que significa um período de clock de 1/12 μ s e um ciclo de máquina de 1 μ s. Observar que as instruções “djnz” e “ret” são executadas em dois ciclos de máquina, enquanto a instrução “mov” é executada em apenas 1 ciclo.



Rótulo	Mnemônico	No. ciclos	No. vezes	Tempo de atraso	Comentário sobre o Efeito da Operação
ATRASO:	MOV R0,#10	1	1	1 μ s	Carrega R0 com 10 decimal
REPETE:	MOV R1,#100	1	10	10 μ s	Faz R1=100 decimal. Repete instrução 10 vezes.
	DJNZ R1,\$	2	1000	2000 μ s	Decrementa R1 1000 vezes: 10 passagens com 100 decrementos
	DJNZ R0,REPETE	2	10	20 μ s	Decrementa R0 10 vezes
	RET	2	1	2 μ s	
				2033 μ s	Atraso de tempo de 2 ms

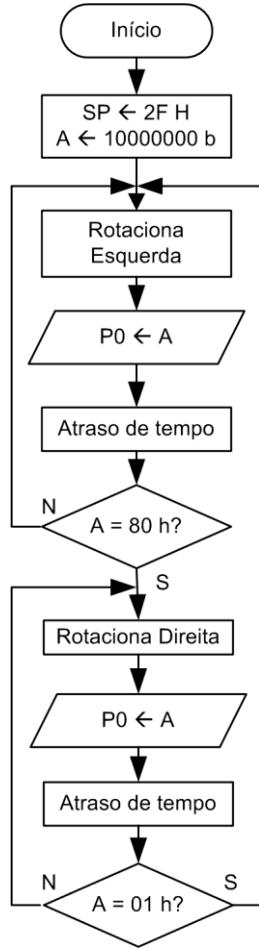
A subrotina de atraso deve ser incluída após a instrução “sjmp inicio”. Deve-se incluir também a chamada da subrotina “lcall atraso”, logo após a instrução “MOV P1,A”.

- (d) Programa que faz uma contagem decimal decrescente interrupta de 59 a 0. Acrescente a subrotina de atraso dada anteriormente.



Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	A próxima instrução será alocada no endereço 00h
	LJMP INICIO	Pula para o “início do programa”
	ORG 30H	A próxima instrução será alocada no endereço 30h
INICIO:	MOV SP,#2FH	Define a pilha na posição 2Fh da memória RAM
	MOV A,#60H	Carrega acumulador com valor 60h
VOLTA:	ADD A,#99	Aciona 99 ao acumulador
	DA A	Corrigir para decimal
	MOV P1,A	Transfere conteúdo do acumulador para a Porta P1
	CJNE A,#00H,VOLTA	Se conteúdo de A for diferente de 0 pula para “volta”
	SJMP INICIO	Pula (Short Jump) para “inicio”
	END	

- (e) Programa que rotaciona um bit alto na porta P0 para a esquerda e para a direita de forma ininterrupta.

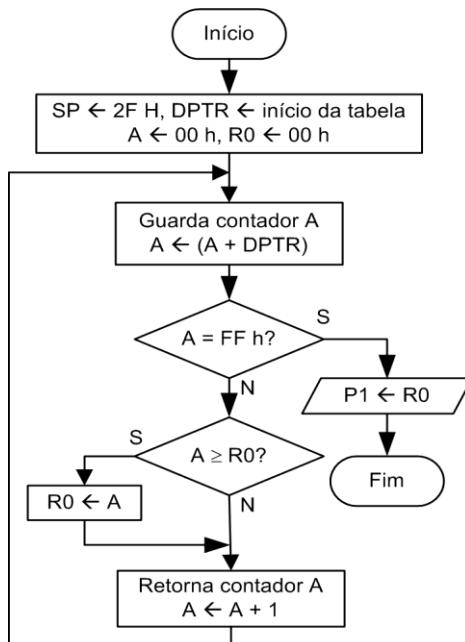


Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	A próxima instrução será alocada no endereço 00h
	LJMP INICIO	Pula para o “início do programa”
	ORG 30H	A próxima instrução será alocada no endereço 30h
INICIO:	MOV SP,#2FH	Define a pilha na posição 2Fh da memória RAM
	MOV A,#10000000B	Carrega acumulador com valor binário “10000000b”
V1:	RL A	Rotaciona para a esquerda o conteúdo do acumulador
	MOV P0,A	Transfere para a porta P0 o conteúdo do acumulador
	LCALL ATRASO	Chama subrotina de atraso de tempo
	CJNE A,#80H,V1	Se A for diferente de “80h” desvia para “VOLTA”
V2:	RR A	Rotaciona para a direita o conteúdo do acumulador
	MOV P0,A	Transfere para a porta P0 o conteúdo do acumulador
	LCALL ATRASO	Chama subrotina de atraso
	CJNE A,#01H,V2	Se A for diferente de “01h” desvia para “VOLTA2”
	SJMP VOLTA	Pula para “VOLTA”
ATRASO:	MOV R0,#10	
V3:	MOV R1,#100	
	DJNZ R1,\$	
	DJNZ R0,V3	
	RET	
	END	

- (f) Programa que lê os valores de uma tabela e mostra na porta P1 o maior desses valores. O maior valor será guardado em R0. O último elemento da tabela é FFH. DPTR aponta o início da tabela.
Observações:

- A instrução **MOV DPTR,#TABELA** faz com que o registrador de 16 bits DPTR assuma o valor correspondente ao endereço inicial da tabela. Nesse programa DPTR = 0100 H.

- Antes de fazer “**SUBB A,R0**” fez-se necessário guardar em B o valor de A, para o caso de A ser menor que R0. Porque, nesse caso, é preciso trocar o valor de R0 pelo valor de A (que agora está em B).



Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	Diretiva que indica que a próxima instrução será alocada no endereço 00h
	LJMP INICIO	Pula para o “início do programa”
	ORG 30H	Diretiva que indica que a próxima instrução será alocada no endereço 30h
INICIO:	MOV SP,#2FH	Pilha na posição 2Fh da memória RAM
	MOV A,#00H	Carrega acumulador com valor 00h
	MOV R0,#00H	Carrega registrador R0 com valor 00h
	MOV DPTR,#TABELA	DPTR recebe endereço inicial da tabela, que nesse programa é 100h
VOLTA:	PUSH ACC	Guarda na pilha o conteúdo do acumulador
	MOVC A,@A+DPTR	acumulador recebe o conteúdo da posição (a + dptr)
	CJNE A,#0FFH,PULA	Compara o conteúdo de “a” com FFh. Se for diferente “pula”
	MOV P1,R0	Transfere para P1 o conteúdo de R0
	LJMP FIM	sjmp não funcionaria aqui porque o “pulo” seria maior que 127 bytes.
PULA:	MOV B,A	guarda em B o conteúdo do acumulador, para recuperá-lo após SUBB A,R0
	SUBB A,R0	Faz a subtração: A = A – R0
	JC PULA2	Vai para “pula2” caso a flag de carry esteja setada
	MOV R0,B	Carrega R0 com o conteúdo de b
PULA2:	POP ACC	Recupera conteúdo do acumulador
	INC A	Incrementa o conteúdo do acumulador
	SJMP VOLTA	Pula para “volta”
	ORG 100H	Diretiva com endereço do início da tabela
TABELA:	DB 05H	Primeiro valor da tabela de dados
	DB 35H	
	DB 12H	
	DB 98H	
	DB 0A1H	Dados que começam com “letra” devem ser precedidos de “0”
	DB 0B5H	
	DB 5AH	
	DB 09H	
	DB 72H	
	DB 40H	Último dado de interesse da tabela
	DB OFFH	Valor usado para indicar fim da tabela
FIM:	NOP	
	END	

4 Interrupções

4.1 Princípio de Funcionamento e Habilitação

Interrupção é o processo pelo qual a execução de um programa é interrompida para a execução de um outro processamento que pode ser solicitado por uma das três fontes abaixo:

- Interrupção por software (instrução)
- Interrupção pedida por periférico externo
- Interrupção pedida por periférico interno (temporizador/contador, porta serial...)

O microcontrolador 8051 pode ser interrompido de **cinco** maneiras diferentes:

- Pela interrupção externa INT0\ - pino 12 (P3.2)
- Pelo timer/counter (temporizador/contador) interno TIMER0
- Pela interrupção externa INT1\ - pino 13 (P3.3)
- Pelo timer/counter (temporizador/contador) interno TIMER1
- Pelo canal de comunicação serial (Pinos 10 e 11 = P3.0 e P3.1)

O pedido de interrupção pode ou não ser atendido, de acordo com a condição de certos registradores. No 8051 os registradores que comandam a interrupção são

- Registrador de Habilitação: **IE** (*Interrupt Enable*) = Reg. A8h
- Registrador de Prioridades: **IP** (*Interrupt Priority*) = Reg. B8h
- Registrador de Controlador: **TCON** (*Timer Control*) = Reg. 88h

Registrador IE : (Reg. A8h)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	EA	x	x	ES	ET1	EX1	ET0	EX0

EA (Enable All) - Quando está zerado (EA = 0), todos as interrupções estão desabilitadas (mascaradas), independentemente de seus bits individuais de controle. Quando está setada (EA = 1), cada uma das interrupções pode ser habilitada ou desabilitada fazendo seus bits de controle 1 ou 0.

EX0 (Enable External Interrupt 0) - Quando está zerado (EX0 = 0) a interrupção externa, cujo pedido vem através do pino INT0\ está desabilitada. Quando está setado (EX0 = 1), a interrupção INT0\ fica habilitada.

ET0 (Enable Timer 0) - Quando ET0 = 0, a interrupção pedida pelo temporizador/contador 0 fica desabilitada. Quando ET0 = 1, a interrupção vinda do temporizador/contador 0 fica habilitada.

EX1 (Enable External Interrupt 1) - Quando está zerado (EX1 = 0) a interrupção externa, cujo pedido vem através do pino INT1\ está desabilitada. Quando está setado (EX1 = 1), a interrupção INT1\ fica habilitada.

ET1 (Enable Timer 1) - Quando ET1 = 0, a interrupção pedida pelo temporizador/contador 1 fica desabilitada. Quando ET1 = 1, a interrupção vinda do temporizador/contador 1 fica habilitada.

ES (Enable Serial) - Quando ES = 0, a interrupção pedida pela porta serial fica desabilitada. Quando ES = 1 essa interrupção fica habilitada.

Registrador IP : (Reg. B8h)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	x	x	x	PS	PT1	PX1	PT0	PX0

- PX0** (*Priority of External Interrupt 0*) - Quando PX0 = 1 a interrupção externa INT0\ recebe prioridade alta.
- PT0** (*Priority of Timer/Counter Interrupt 0*) - Quando PT0 = 1 a interrupção pedida pelo temporizador/contador 0 recebe prioridade alta.
- PX1** (*Priority of External Interrupt 1*) - Quando PX1 = 1 a interrupção externa INT1\ recebe prioridade alta.
- PT1** (*Priority of Timer/Counter Interrupt 1*) - Quando PT1 = 1 a interrupção pedida pelo temporizador/contador 1 recebe prioridade alta.
- PS** (*Priority of Serial Port Interrupt*) - Quando PS = 1 a interrupção pedida através da porta serial recebe prioridade alta.

Quando PX0, PT0, PX1, PT1 e PS são zero, elas são de prioridade baixa. Caso uma interrupção de prioridade 1 seja solicitada durante a execução de uma de prioridade 0, o processamento é interrompido para o atendimento da interrupção de prioridade maior.

No caso de todas as interrupções terem a mesma prioridade (0 ou 1), a ordem de atendimento das interrupções é:

- Interrupção externa 0 → Maior prioridade
- Temporizador/contador 0
- Interrupção externa 1
- Temporizador/contador 1
- Canal serial → Menor prioridade

As interrupções externas podem ser ajustadas para serem detectadas por nível 0 ou pela transição do nível 1 para o nível 0. O ajuste é feito através o registrador TCON, dado abaixo.

Registrador TCON:
(Reg. 88h)

Controle do Temporizador				Bit 3	Bit 2	Bit 1	Bit 0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT0 (*Interrupt 0 Type*) - Quando IT0 = 1 a interrupção externa 0 será reconhecida pela transição de 1 para 0 no pino INT0\. Quando IT0 = 0, a interrupção é reconhecida quando o sinal no pino INT0\ está em nível baixo (0).

IE0 (*Interrupt 0 Edge Flag*) - É setado pelo hardware quando uma interrupção externa através de INT0\ é detectada. É zerada quando da execução da instrução RETI (retorno da subrotina de atendimento).

IT1 (*Interrupt 1 Type*) - Quando IT1 = 1 a interrupção externa 1 será reconhecida pela transição de 1 para 0 no pino INT1\. Quando IT1 = 0, a interrupção é reconhecida quando o sinal no pino INT1\ está em nível baixo (0).

IE1 (*Interrupt 1 Edge Flag*) - É setado pelo hardware quando uma interrupção externa através de INT1\ é detectada. É zerada quando da execução da instrução RETI (retorno da subrotina de atendimento).

4.2 Endereços Desvio das Interrupções

Quando ocorre uma das cinco interrupções do 8051 o processamento é desviado para os endereços abaixo. Como há pouco espaço em bytes nesses endereços, deve-se usar uma instrução de desvio para um outro endereço onde seja possível escrever toda a rotina de atendimento da interrupção.

Interrupção Solicitada	Endereço de desvio
Reset	0000h
INT0\	0003h
Timer/counter 0	000Bh
INT1\	0013h
Timer/counter 1	001Bh
Canal Serial	0023h

Para o caso dos microcontroladores com 3 temporizadores, como é o caso do AT89S52, o endereço de desvio para o caso de interrupção do temporizador 2 é o **002BH**. O bit de habilitação é o bit 5 (**ET2**), do registrador **IE**. A prioridade 1 para o temporizador 2 é dada através do bit 5 (**PT2**) do registrador **IP**. Os registradores associados estão no próximo capítulo.

4.3 Programas Exemplos com Interrupção

1. Programa com interrupção externa 0. Quando a interrupção INT0 é solicitada através do pino P3.2 o processamento é desviado para a posição 03 h da memória ROM e, em seguida, é desviado para uma subrotina que manda uma contagem crescente para a porta P1.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 03H	Endereço de desvio quando há um pedido da interrupção INT0 (pino P3.2).
	LJMP ATENDE	Quando há um pedido de interrupção através do pino P3.2, desvia para <i>atende</i>
	ORG 30H	
INICIO:	MOV SP, #2FH	
	MOV IE, #81H	Habilita a interrupção externa 0 (EA = 1 e EX0 = 1)
	MOV TCON, #01H	INT0\ reconhecida pela transição de 1 para 0 (IT0 = 1). Poderia ser SETB IT0
	MOV A, #00H	
	CJNE A,#01H, \$	Aguarda interrupção. Enquanto a ≠ 01h, fica aguardando nessa linha.
	SJMP FIM	O caractere “\$” indica desvio para a mesma linha.
ATENDE:	MOV P1,A	Subrotina de atendimento da interrupção INT0. Faz P1 = a
	INC A	Incrementa valor de “A”
	LCALL ATRASO	Chama subrotina de atraso
	CJNE A,#00H,ATENDE	Compara a com 00h. Se for diferente, vai para “atende”
	MOV A,#01H	Quando a = 00h, na linha anterior, então faz A = 01h, nesta linha.
	RETI	Retorna da subrotina de interrupção
ATRASO:	MOV R0,#10	Subrotina de atraso de tempo
REPETE:	MOV R1,#100	
	DJNZ R1, \$	Repete a instrução que decrementa r1 até zerar r1
	DJNZ R0, REPETE	
	RET	
FIM:	NOP	
	END	

Obs.: O pedido de interrupção é atendido quando pino P3.2 do microcontrolador passa de 1 para 0.

2. Programa com as interrupções externas 0 e 1. Neste programa, quando a interrupção INT0 é acionada através do pino P3.2, o processamento é desviado para uma subrotina que faz uma contagem crescente na porta P1. Quando a interrupção INT1 é solicitada através do pino P3.3, uma contagem decrescente é enviada também para a porta P1. Lembrar de guardar o acumulador na pilha em cada atendimento de subrotina, uma vez que ele é utilizado nas duas subrotinas.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 03H	
	LJMP ATENDE0	Desvia para a subrotina de atendimento da interrupção INT0
	ORG 13H	
	LJMP ATENDE1	Desvia para a subrotina de atendimento da interrupção INT1
	ORG 30H	
INICIO:	MOV SP, #2fH	
	MOV IE, #85H	Habilita interrupções externas 0 e 1 (EA = 1, EX0 = 1 e EX1 = 1)
	MOV TCON, #05H	INT0 e INT1 por transição de 1 para 0 (IT0 = 1 e IT1 = 1).
	MOV A, #00H	
	SJMP \$	Aguarda interrupção em um laço infinito
ATENDE0:	PUSH ACC	Início da subrotina de atendimento da interrupção 0.
	MOV A,#00H	Faz acumulador igual a zero
V1:	MOV P1,A	Conteúdo de A é transferido para a porta P1
	INC A	Incrementa acumulador
	LCALL ATRASO	Chama subrotina de atraso de tempo
	CJNE A,#00H,V1	Compara acumulador com “00h”. Se for diferente desvia para “V1”
	POP ACC	Recupera da pilha o conteúdo do acumulador
	RETI	retorna de subrotina de interrupção
ATENDE1:	PUSH ACC	Início da subrotina de atendimento da interrupção 1.
	MOV A,#0ffH	
V2:	MOV p1,A	
	LCALL ATRASO	
	DEC A	Decrementa conteúdo do acumulador
	CJNE A,#0FFH,V2	Compara A com “FFh”. Se for diferente desvia para “V2”
	POP ACC	Recupera conteúdo do acumulador que foi guardado na pilha
	RETI	
ATRASO:	MOV R0,#10	
REPETE:	MOV R1,#100	
	DJNZ R1, \$	
	DJNZ R0, REPETE	
	RET	
	END	

3. Este programa funciona como o anterior, mas ao invés de uma contagem crescente e uma decrescente, ele faz um deslocamento à esquerda dos bits da porta P1 para um pedido de INT0 e um deslocamento à direita dos mesmos bits para um pedido de INT1.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 03H	
	LJMP ATENDE0	Desvia para a subrotina de atendimento da interrupção INT0
	ORG 13H	
	LJMP ATENDE1	Desvia para a subrotina de atendimento da interrupção INT1
	ORG 30H	
INICIO:	MOV SP, #2FH	
	MOV IE, #85H	Habilita interrupções externas 0 e 1 (EA = 1, EX0 = 1 e EX1 = 1)
	MOV TCON, #05H	INT0 e INT1 por transição de 1 para 0 (IT0 = 1 e IT1 = 1).
	MOV A, #00H	Acumulador assume o valor inicial 00.
	SJMP \$	aguarda interrupção num loop infinito nessa linha
ATENDE0:	PUSH ACC	Subrotina de atendimento da interrupção 0. Guarda acumulador na pilha
	MOV A,#01H	O bit 0 do acumulador assume o valor 1. Os demais são zero.
V1:	MOV P1,A	Transfere para a porta P1 o conteúdo do acumulador
	LCALL ATRASO	Chama subrotina de atraso de tempo
	RL A	Rotaciona para a esquerda o conteúdo do acumulador
	CJNE A,#01H,V1	Compara acumulador com “01h”. Se for diferente desvia para “V1”
	POP ACC	Recupera da pilha o conteúdo do acumulador
	RETI	retorna de subrotina de interrupção
ATENDE1:	PUSH ACC	Subrotina de atendimento da interrupção 1. Guarda acumulador na pilha
	MOV A,#80H	O bit 7 do acumulador assume o valor 1. Os demais são zero.
VOLTA2:	MOV P1,A	Transfere para a porta P1 o conteúdo do acumulador
	LCALL ATRASO	Chama subrotina de atraso de tempo
	RR A	Rotaciona para a direita o conteúdo do acumulador
	CJNE A,#80H,VOLTA2	Compara A com “80h”. Se for diferente desvia para “V2”
	POP ACC	Recupera da pilha o conteúdo do acumulador
	RETI	Retorna de subrotina de interrupção
ATRASO:	MOV R0,#10	
REPETE:	MOV R1,#100	
	DJNZ R1, \$	
	DJNZ R0, REPETE	
	RET	
	END	

Obs.:

1. Lembrar que, na simulação, o pedido de interrupção é feito posicionando o cursor nos *bits 2 e 3 da porta P3* e digitando 0 no lugar do 1 presente. As interrupções serão atendidas sempre na transição de 1 para zero, uma vez que TCON = 05h (0000 0101b).
2. Durante uma das contagens, simule dois pedidos de interrupção simultâneos fazendo a transição de 1 para 0 nos pinos P3.2 e P3.3. Verifique então que a interrupção INT0 será atendida primeiro, porque ela tem prioridade sobre a INT1.

5 Temporizadores e Contadores do 8051

5.1 Princípio de Funcionamento e Modos de Operação

O 8051 tem dois temporizadores/contadores de 16 bits que podem trabalhar em 4 modos distintos, mostrados na Tabela 5.1.

Tabela 5.1: Modos de operação dos temporizadores 0 e 1

Modo	Descrição
0	Contador/Temporizador de 13 bits (8 bits com divisor de frequência de até 32 vezes). Pode contar até $8192 = 32 \times 255$.
1	Contador/Temporizador de 16 bits (pode contar até 65535)
2	Contador/Temporizador de 8 bits com recarga automática
3	2 contadores/temporizadores independentes de 8 bits. Esse modo não vale para o Contador/Temporizador 1, que fica inativo nesse modo.

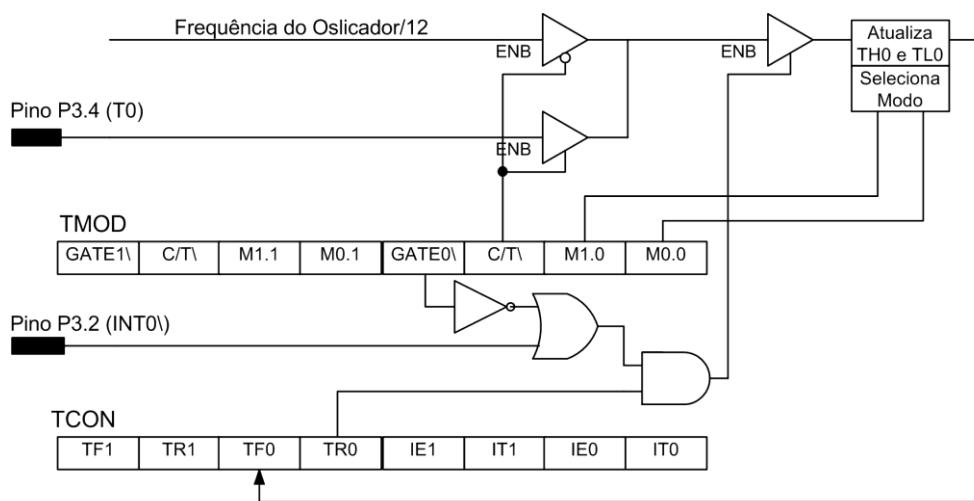


Fig. 5.1: Diagrama de blocos simplificado do temporizador/contador 0

O diagrama da Fig. 5.1 auxilia na análise do funcionamento do temporizador/contador. São mostrados os registradores **TMOD** e **TCON**, através dos quais o temporizador é configurado e controlado. No registrador **TMOD** o nibble inferior refere-se ao temporizador/contador zero e o nibble superior refere-se ao temporizador/contador 1. Os bits **M0** e **M1** configuram o modo de operação (0, 1 2 ou 3). O bit **C/T** define se o funcionamento é como contador (**C/T\ = 1**), ou como temporizador (**C/T\ = 0**). Na operação como temporizador, o clock é interno, vindo do oscilador. A frequência é 1/12 da frequência do cristal oscilador. Na operação como contador de eventos, o clock é externo, vindo através de **T0** (**P3.4**), para o contador zero e **T1** (**P3.5**), para o contador 1. O pino **GATE** define se o sinal de disparo do contador/temporizador vem através de software (bit **TR**) do registrador **TCON**, ou de um sinal externo, através do pino **INT0** (**P3.2**), para o temporizador/contador zero e **INT1** (**P3.3**), para o temporizador/contador 1. Se **GATE\ = 0**, o comando **SETB TR0** dispara o contador/temporizador zero e **CLR TR0** interrompe. Para disparar e parar o temporizador/contador 1 utiliza-se, respectivamente, as instruções **SETB TR1** e **CLR TR1**.

Além dos bits **TR0** e **TR1**, o registrador **TCON** possui os bits **TF0** e **TF1** que se referem aos temporizadores/contadores zero e 1. O bit **TF** é setado pelo temporizador/contador ao final de cada contagem, o que gera um pedido de interrupção. Se as interrupções desses temporizadores estiverem setadas há um desvio para os endereços **0BH** ou **1BH**, respectivamente, para os temporizadores zero e 1. O nibble inferior do registrador **TCON** refere-se às interrupções e já foi estudado.

A contagem do contador/temporizador começa sempre do valor definido pelo programador, através dos registradores **TH** e **TL** e vai até o valor máximo de contagem, por exemplo, **FFFFH**, para o

caso do modo 1 (temporizador de 16 bits). A partir da segunda contagem o temporizador/contador começa do valor 0000H, a menos que o programador recarregue os registradores **TH** e **TL** com o início desejado. No modo 2 (recarga automática) não há necessidade dessa recarga pelo programador; ela é feita automaticamente pelo microcontrolador. Nesse caso, modo 2, o contador é o registrador **TL** (8 bits) e o valor da recarga é dado através do registrador **TH**. Ou seja, a cada nova contagem, o microcontrolador transfere o conteúdo de **TH** para **TL** e segue a contagem até FFH.

No caso de existência de 3 temporizadores, os registradores especiais associados ao temporizador 2 são: **T2CON**, **T2MOD**, **RCAP2L**, **RCAP2H**, **TL2** e **TH2**. Os pinos externos associados a esse temporizador são: P1.0 (**T2**) e P1.1 (**T2EX**).

O Temporizador 2 pode funcionar como temporizador e como contador de eventos, assim como os temporizadores zero e 1. O tipo de operação é selecionado através do bit **C/T2** do registrador especial **T2CON** (Registrador Especial **C8H**), mostrado na Tabela 5.2.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2CON	T2F	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2\	CP/RL2\

Tabela 5.2: Descrição dos bits do registrador T2CON.

Bit	Descrição
T2F	Flag de overflow do temporizador 2. É setada por hardware e deve ser zerada por software. Essa flag não será setada nos casos de TCLK = 1 ou RCLK = 1.
EXF2	Flag externa do Temporizador 2. É setada quando uma captura ou recarga automática é causada por uma transição de 1 para 0 no pino T2EX (P1.1) e EXEN2 = 1. Quando a interrupção do temporizador 2 está habilitada, EXF2 = 1 fará com que o processamento seja desviado para a subrotina de atendimento da interrupção (endereço 2BH). A flag EXF2 deve ser zerada por software. A flag EXF2 não provoca interrupção no modo de operação como contador crescente/decrescente (DCEN = 1).
RCLK	Habilita clock de recepção. Quando setado faz com que os pulsos de overflow do temporizador 2 seja usado como clock de recepção da porta serial nos modos 1 e 3. RCLK = 0 faz com que o temporizador 1 seja usado para gerar o clock de recepção.
TCLK	Habilita clock de transmissão. Quando setado faz com que os pulsos de overflow do temporizador 2 seja usado como clock de transmissão da porta serial nos modos 1 e 3. TCLK = 0 faz com que o temporizador 1 seja usado para gerar o clock de transmissão.
EXEN2	Habilitação externa do temporizador 2. Quando setado permite a captura ou recarga como o resultado de uma transição negativa no pino T2EX (P1.1) se o temporizador 2 não estiver sendo usado para gerar o clock da porta serial. EXEN2 = 0 faz com que o temporizador 2 ignore eventos no pino T2EX.
TR2	Controle de partida (TR2 = 1) /parada (TR2 = 0) do temporizador 2.
C/T2\	Seleciona o modo como temporizador (C/T2\ = 0) ou como contador (C/T2\ = 1).
CP/RL2\	Bit de seleção dos modos de recarga/captura. CP/RL2\ = 1 ativa a captura na transição 1 para 0 no pino T2EX (P1.1), se EXEN2 = 1. CP/RL2\ = 0 ativa recarga automática quando há overflow do temporizador 2 ou na transição de 1 para 0 no pino T2EX, quando EXEN2 = 1. Quando RCLK ou TCLK = 1, este bit é ignorado e o temporizador é forçado a operar no modo de recarga no overflow do temporizador 2.

O temporizador 2 possui 3 modos de operação: modo de captura, modo de recarga automática (como contador crescente ou decrescente) e gerador de clock para a comunicação serial. Os modos de operação são selecionados através de bits do registrador **T2CON**, como mostrado na Tabela 5.3. O temporizador 2 consiste de dois registradores de 8 bits, **TH2** e **TL2**. No modo Temporizador o registrador **TL2** é incrementado a cada ciclo de máquina. Uma vez que um ciclo de máquina consiste de 12 ciclos de clock, a frequência do temporizador corresponde a 1/12 da frequência do cristal oscilador.

Tabela 5.3: Modos de operação do temporizador 2

RCLK + TCLK	CP/RL2\	TR2	Modo
0	0	1	Modo de recarga automática de 16 bits
0	1	1	Modo de captura de 16 bits
1	X	1	Gerador de clock para a porta serial
X	X	0	Desligado

Na função de contador, o registrador é incrementado na transição de 1 para 0 no pino externo T2 (P1.0). Nesta função, a entrada externa é amostrada durante o estado S5P2 de cada ciclo de máquina. Quando as amostras mostram um nível alto em um ciclo e baixo no próximo ciclo, o contador é incrementado. O novo valor do contador aparece no registrador durante o estado S3P1 do ciclo seguinte ao ciclo de detecção da transição. Desde que são necessários dois ciclos de máquina para o reconhecimento de uma transição de 1 para 0, a taxa máxima de contagem é 1/24 da frequência do cristal oscilador. Para assegurar que um determinado nível seja detectado pelo menos uma vez antes que ele mude de nível, o nível deveria ser mantido por pelo menos um ciclo de máquina completo.

No modo de captura, duas opções são selecionadas pelo bit EXEN2 do registrador **T2CON**. Se EXEN2 = 0, o temporizador 2 é um temporizador ou contador de 16 bits que, quando ocorre overflow, seta o bit TF2 do registrador **T2CON**. Esse bit pode ser usado para gerar uma interrupção. Se EXEN2 = 1, o temporizador 2 executa a mesma operação, mas uma transição de 1 para 0 no bit externo T2EX (pino P1.1) também faz com que o conteúdo atual dos registradores TH2 e TL2 seja transferido (capturado) para os registradores RCAP2H e RCAP2L, respectivamente. A transição de 1 para 0 em T2EX seta ainda o bit EXF2 do registrador T2CON. A flag EXF2, assim como a flag TF2, pode gerar uma interrupção. A Fig. 5.2 ilustra a operação no modo de captura.

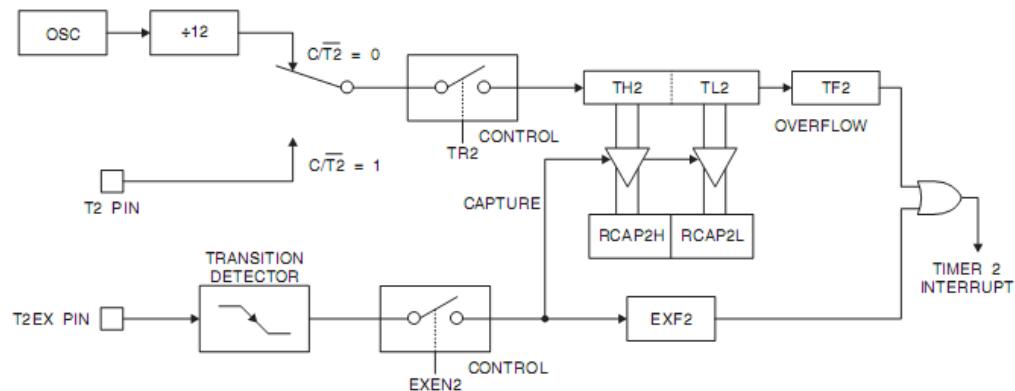


Fig. 5.2: Diagrama de blocos mostrando o modo de captura

O temporizador 2 pode ser programado para uma contagem crescente ou decrescente quando configurador no modo de 16 bits com recarga automática. Esta seleção é feita através do bit DCEN (habilitação de contagem decrescente) do registrador especial **T2MOD** (registrador C9H).

T2MOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	X	X	X	X	X	X	T2OE	DCEN

Tabela 5.3: Descrição dos bits do registrador **T2MOD**.

Bit	Descrição
T2OE	Bit que habilita a saída do temporizador 2
DCEN	Quando setado, este bit permite que o temporizador 2 seja configurado como contador crescente ou decrescente, dependendo do valor do pino externo T2EX (pino P1.1). Se DCEN = 0 o temporizador 2 faz uma contagem crescente.

A Fig. 5.3 ilustra a operação no modo recarga automática crescente (DCEN = 0). Quando DCEN = 0 o temporizador 2 automaticamente faz uma contagem crescente com recarga automática. O valor da recarga é armazenado nos registradores RCAP2H e RCAP2L. O bit EXEN2 do registrador **T2CON** permite duas opções de operação: com EXEN2 = 0, a cada fim de contagem, quando o valor dos registradores TH2 e TL2 chega a FFFFH, a flag TF2 é setada e o conteúdo de RCAP2H e RCAP2L é recarregado em TH2 e TL2. Se EXEN2 = 1 a recarga automática pode ocorrer a cada final de contagem (overflow do temporizador) ou pela transição de 1 para 0 do sinal no pino T2EX (pino P1.1). Essa transição também seta a flag EXF2 que, do mesmo modo que a flag TF2, pode gerar uma interrupção, se a interrupção do temporizador 2 estiver habilitada.

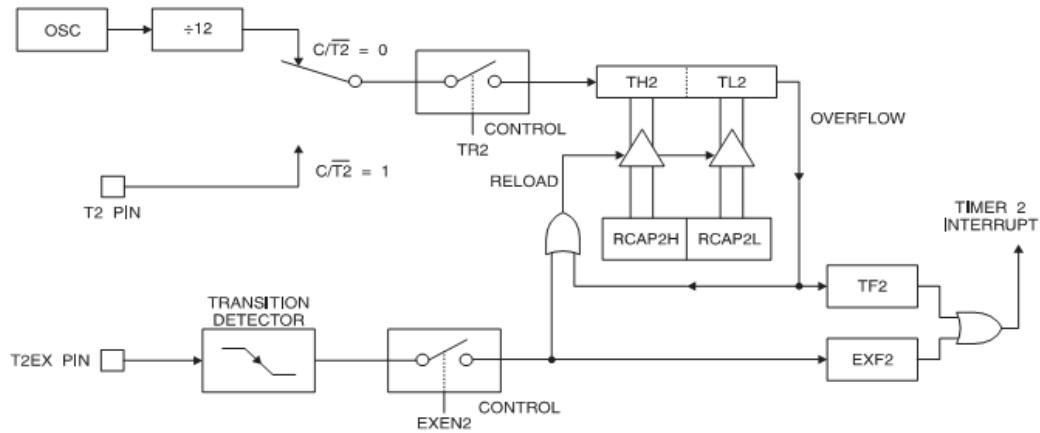


Fig. 5.3: Diagrama de blocos mostrando o modo de contagem crescente com recarga automática

A Fig. 5.4 ilustra a operação no modo recarga automática crescente ou decrescente ($DCEN = 1$). Quando $DCEN = 1$ o temporizador 2 pode fazer uma contagem crescente ou decrescente, dependendo do estado do pino de controle $T2EX$ (pino P1.1). $T2EX = 1$ habilita uma contagem crescente do temporizador 2. Haverá overflow quando a contagem chegar a FFFFH e o bit $TF2$ será setado (pedido de interrupção pendente). Haverá ainda a recarga dos valores armazenados em $RCAP2H$ e $RCAP2L$ nos registradores $TH2$ e $TL2$, respectivamente.

$T2EX = 0$ habilita uma contagem decrescente do temporizador 2. Haverá um underflow (ultrapassagem para baixo) quando a contagem em $TH2$ e $TL2$ for igual ao valor armazenado em $RCAP2H$ e $RCAP2L$. O underflow seta o bit $TF2$ e provoca a recarga do valor FFFFH nos registradores $TH2$ e $TL2$. O bit $EXF2$ alterna seu valor sempre que ocorre um overflow ou um underflow do temporizador 2. Esse bit pode ser usado como o 17º bit de resolução do contador. Neste modo de operação esse bit não gera interrupção.

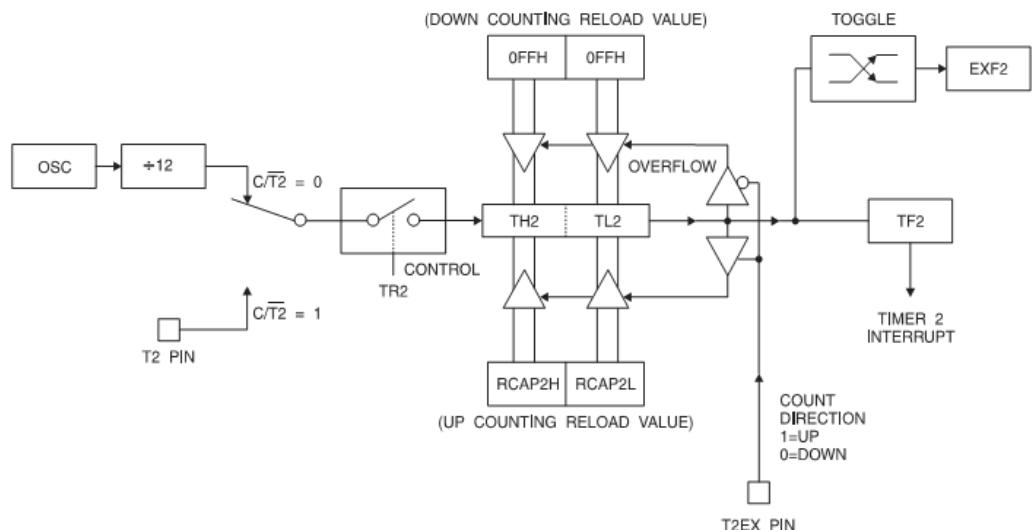


Fig. 5.4: Diagrama de blocos mostrando o modo de contagem crescente/decrescente com recarga automática

5.2 Programas Exemplos usando Temporizadores

Exemplo 1: Programa em que o temporizador 0 faz uma contagem ininterrupta de 0 a 65535 (no modo 1) e o temporizador 1 faz uma contagem de 0 a 8192 (no modo 0). Observar no simulador a evolução dos registradores $TH0$ e $TL0$, para o temporizador 0 e $TH1$ e $TL1$ para o temporizador 1.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	Pula para o início do programa
	ORG 30H	
INICIO:	MOV SP, #2FH	Faz apontador de pilha SP = 2FH
	MOV TMOD,#01H	Temporizador 0 no modo de 16 bits (modo 1) e temporizador 1 no modo 0
	SETB TR0	Dispara temporizador 0
	SETB TR1	Dispara temporizador 1
	SJMP \$	Laço infinito
	END	

Exemplo 2: Programa em que o temporizador 0, no modo 1, faz uma contagem ininterrupta de 10.000 pulsos (de 55535 a 65535) e gera uma onda quadrada no pino P1.0. Observar no simulador a evolução dos registradores TH0 e TL0 e a mudança de estado do pino P1.0.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	Pula para o início do programa
	ORG 30H	
INICIO:	MOV SP, #2FH	Faz apontador de pilha SP = 2FH
	MOV TMOD,#01H	Temporizador 0 no modo de 16 bits (modo 1) e temporizador 1 no modo 0
V1:	MOV TH0,#0D8H	Carrega TH0/TL0 com valor hexadecimal D8EFH = 55.535 decimal
	MOV TL0,#0EFH	
	SETB TR0	Dispara temporizador 0
	JNB TF0,\$	Aguarda final de contagem do temporizador 0
	CLR TF0	Limpa flag de final de contagem
	CPL P1.0	Complementa o pino 0 da porta P1 → Gera onda quadrada no pino P1.0
	SJMP V1	Desvia para recarregar valor inicial da contagem
	END	

A Fig. 5.5 mostra a onda gerada no pino P1.0 pelo programa do Exemplo 2. O cristal oscilador utilizado é de 12 MHz, o que significa um clock de 1 MHz do temporizador. Assim, um período de clock do temporizador é 1 μ s, e uma contagem de 10.000 pulsos resulta em um período de tempo de 10 ms.

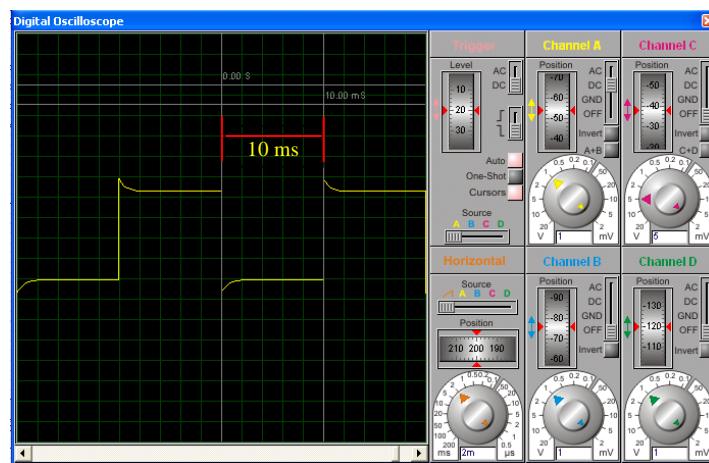


Fig. 5.5: Onda quadrada gerada com o temporizador 0 modo 1

Exemplo 3: Programa em que o temporizador 0, no modo 1, faz uma contagem ininterrupta de 10.000 pulsos (de 55535 a 65535) e gera uma onda quadrada no pino P1.0 através de interrupção do temporizador. Observar no simulador o desvio para o endereço 0BH para atendimento da interrupção.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	Pula para o início do programa
	ORG 0BH	Endereço da interrupção do temporizador 0
	MOV TH0,#0D8H	Carrega TH0/TL0 com valor hexadecimal D8EFH = 55.535 decimal
	MOV TL0,#0EFH	
	CLR TF0	Limpa flag de final de contagem
	CPL P1.0	Complementa o pino 0 da porta P1 → Gera onda quadrada no pino P1.0
	RETI	
	ORG 30H	
INICIO:	MOV SP, #2FH	Faz apontador de pilha SP = 2FH
	MOV IE,#82H	Habilita a interrupção do temporizador 0
	MOV TMOD,#01H	Temporizador 0 no modo de 16 bits (modo 1) e temporizador 1 no modo 0
	MOV TH0,#0D8H	Carrega TH0/TL0 com valor hexadecimal D8EFH = 55.535 decimal
	MOV TL0,#0EFH	
	SETB TR0	Dispara temporizador 0
	SJMP \$	Aguarda em um laço infinito. Sai do laço a cada final de contagem
	END	

Exemplo 4: Programa que gera onda quadrada no pino P1.0 e cuja subrotina de atraso de tempo utiliza o temporizador 1 no modo 1. Se o cristal oscilador for de 12 MHz o tempo de atraso corresponderá a aproximadamente 1 s.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP, #2FH	
	MOV TMOD,#10H	Temporizador 1 no modo de 16 bits (modo 1)
V1:	CPL P1.0	Complementa o bit 0 da porta P1
	LCALL ATRASO	Chama subrotina de atraso de tempo
	SJMP V1	
ATRASO:	MOV R0,#20	Registrador R0 recebe valor 20 decimal
V2:	MOV TH1,#3CH	TH1 recebe valor 3CH
	MOV TL1,#0AFH	TL1 recebe valor AFH. Contador conta de 15535 até 65535 = 50000 pulsos
	SETB TR1	TR1 = 1 → dispara o temporizador 1
ESPERA:	JNB TF1, ESPERA	Espera a flag de fim de contagem ser setada → Espera TF0 = 1.
	CLR TF1	Limpa flag que indica fim da contagem.
	DJNZ R0, V2	Decrementa R0 e desvia para V2 se não for zero
	CLR TR1	Pára contador após o fim de 20×50000 pulsos = 1.000.000 pulsos.
	RET	Retorna da subrotina. Se cada pulso corresponder a $1 \mu\text{s}$ → atraso ≥ 1 s
	END	

A Fig. 5.6 mostra uma onda quadrada com meio período de 1 s, gerada no pino P1.0 pelo programa do Exemplo 4. O cristal oscilador utilizado é de 12 MHz, o que significa um clock de 1 MHz do temporizador. Assim, um período de clock do temporizador é $1 \mu\text{s}$ e são necessários 1.000.000 de pulsos para se obter o tempo de 1 s. Uma vez que o temporizador no modo 1 conta somente até 65.535, o que resultaria em um tempo de 65,535 ms, optou-se por gerar 50.000 pulsos (50 ms), repetida por 20 vezes. Os 50.000 pulsos são gerados fazendo uma contagem de 15.535 (3CAFH) até 65.535 (FFFFH).

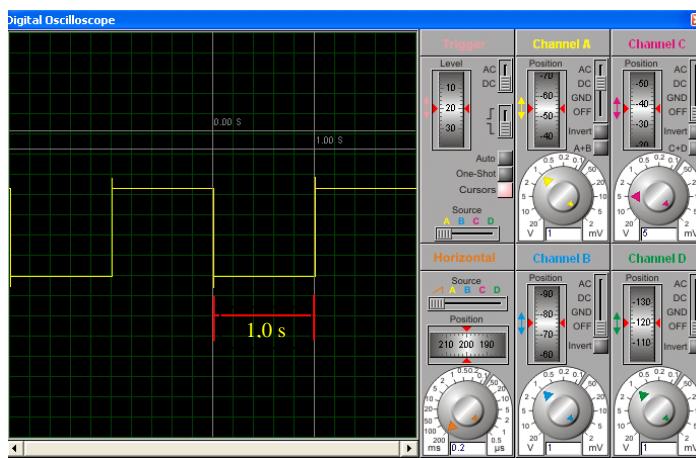


Fig. 5.6: Onda quadrada gerada com o temporizador 1 modo 1 – atraso de 1 s

Exemplo 5: Uso do temporizador 1 no modo 2, com interrupção. Nesse exemplo, a cada vez que o temporizador 1 chega ao final da contagem (FFH), há um desvio para a subrotina “atende1”, onde o pino 0 da porta P1 é complementado e o valor de recarga TH1 é também complementado. Observar que a onda gerada no pino P1.0 possui tempo ligado diferente do tempo desligado, devido à complementação de TH1 a cada interrupção (Geração de sinal PWM: Modulação de Largura de Pulso).

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 1BH	Endereço de desvio para a interrupção do temporizador 1.
	LJMP ATENDE1	Desvia para subrotina de atendimento de interrupção do temporizador 1
	ORG 30H	
INICIO:	MOV SP, #2FH	
	MOV IE,#88H	Habilita interrupção do temporizador 2
	MOV TMOD,#20H	Temporizador 1 no modo de recarga automática (modo 2)
	MOV TH1,#3FH	Carrega registrador de recarga TH1 com valor 3FH
	MOV TL1,TH1	Transfere para TL1 o conteúdo de TH1. TL1 é o contador no modo 2.
	SETB TRI	Dá início à contagem do temporizador 1
	SJMP \$	Fica num <i>loop</i> infinito, saindo a cada fim de contagem do temporizador 1
ATENDE1:	CPL P1.0	Complementa o pino P1.0
	CLR TF1	Limpa a flag de final de contagem
	MOV A,TH1	Transfere conteúdo de TH1 para acumulador
	CPL A	Complementa acumulador
	MOV TH1,A	Devolve para TH1 o conteúdo complementado
	RETI	Retorna da subrotina de interrupção
	END	

A Fig. 5.7 mostra uma onda modulada gerada pelo temporizador 1 no modo 2. O período da onda corresponde ao período completo do temporizador nesse modo (255 µs), uma vez que o contador é o registrador TL1 (8 bits) e o cristal oscilador utilizado é de 12 MHz, ou seja, um período de clock do temporizador é 1 µs. O tempo ligado é de 63 µs, uma vez que nesse intervalo o contador conta de C0H até FFH, que corresponde a 63 pulsos. O tempo desligado é de 192 us, que corresponde a uma contagem de 3FH até FFH do contador no modo 2.

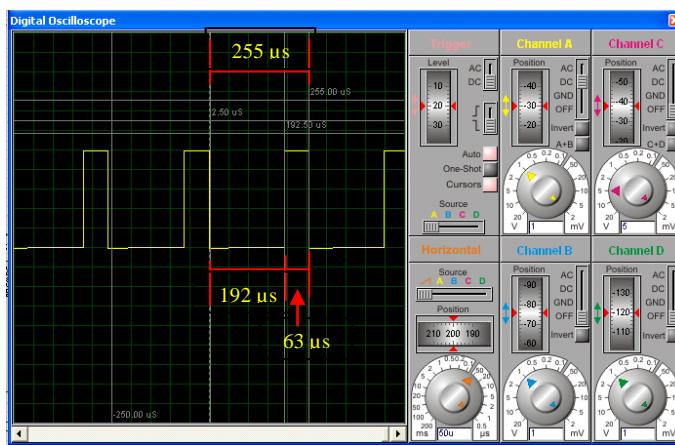


Fig. 5.7: Onda modulada gerada pelo temporizador 1 no modo 2

Exemplo 6: O programa gera uma onda modulada no pino P1.0 (sinal PWM) usando o temporizador 1 no modo 2, com interrupção externa. A interrupção externa 0 é usada para aumentar o tempo ligado de P1.0 e a externa 1 é usada para diminuir o tempo ligado. Ambas são por transição. Na subrotina de interrupção externa 0 o período ligado TON é incrementado e seu complemento é transferido para TOFF. Na subrotina da interrupção externa 1 o período ligado TON é decrementado e seu complemento é transferido para TOFF.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	TON EQU 0AH	Registrador 0AH recebe o nome de TON (Tempo ligado de P1.0)
	TOFF EQU 0BH	Registrador 0BH recebe o nome de TOFF (Tempo desligado de P1.0)
	ORG 00H	
	LJMP INICIO	
	ORG 03H	Endereço de desvio para a interrupção externa 0
	LJMP EXTERNA0	Desvia para a subrotina de atendimento da interrupção externa 0
	ORG 13H	Endereço de desvio para a interrupção externa 1
	LJMP EXTERNA1	Desvia para a subrotina de atendimento da interrupção externa 1
	ORG 1BH	Endereço de desvio para a interrupção do temporizador 1.
	LJMP ATENDE1	Desvia para subrotina de atendimento de interrupção do temporizador 1
	ORG 30H	
INICIO:	MOV SP, #2FH	Apontador de pilha SP = 2FH
	MOV IE,#8DH	Habilita interrupção do temporizador 2 e as externas 0 e 1. IE = 1000 1101
	MOV TCON,#05H	As interrupções externas 0 e 1 são por transição
	MOV TMOD,#20H	Temporizador 1 no modo de recarga automática (modo 2)
	MOV TON,#3FH	Carrega registrador TON com valor 3FH
	MOV A,TON	Carrega acumulador com valor do tempo ligado
	CPL A	Complementa o tempo ligado
	MOV TOFF,A	Carrega registrador de tempo desligado TOFF com complemento de TON
	CLR P1.0	Limpa o pino P1.0
	MOV TL1,TON	Carrega TL1 com tempo ligado. TL1 é o contador no modo 2
	MOV TH1,TOFF	Carrega valor da próxima recarga TH1 com tempo desligado TOFF
	SETB TR1	Dá início à contagem do temporizador 1
	SJMP \$	Fica num loop infinito, saindo a cada fim de contagem do temporizador 1
ATENDE1:	CLR TF1	Limpa a flag de final de contagem
	JB P1.0,DESLIGA	Se P1.0 = 1 desvia para “DESLIGA”. Se P1.0 = 0 vai para próxima linha
LIGA:	SETB P1.0	Liga o pino P1.0
	MOV TH1,TON	Carrega TH1 com o próximo valor de recarga
	RETI	

DESLIGA:	CLR P1.0	Desliga pino P1.0
	MOV TH1,TOFF	Carrega TH1 com o próximo valor de recarga
	RETI	
EXTERNAO:	INC TON	Aumenta o tempo ligado
	MOV A,TON	Carrega acumulador com o valor do tempo ligado
	CPL A	Complementa o tempo ligado
	MOV TOFF,A	Carrega o tempo desligado com o complemento do tempo ligado
	RETI	
EXTERNAO:	DEC TON	Diminui o tempo ligado do pino P1.0
	MOV A,TON	Carrega acumulador com valor do tempo ligado
	CPL A	Complementa o tempo ligado
	MOV TOFF,A	Carrega o tempo desligado com o complemento do tempo ligado
	RETI	
	END	

Exemplo 7: Uso do temporizador 1 no modo 2, com interrupção. O programa gera no pino P1.0 uma onda cujos tempos ligado e desligado variam devido ao incremento do valor de recarga TH1. Na subrotina de atendimento da interrupção do temporizador 1 o valor de recarga TH1 é incrementado a cada final de contagem. Assim, a recarga seguinte está incrementada em uma unidade. Assim, a contagem começa sempre uma unidade a mais que a anterior e o período da onda quadrada diminui a cada nova contagem.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 1BH	Endereço de desvio para a interrupção do temporizador 1.
	LJMP ATENDE1	Desvia para a subrotina que atende o temporizador 1.
	ORG 30H	
INICIO:	MOV SP, #2FH	
	MOV IE,#88H	Habilita a interrupção do temporizador 2
	MOV TMOD,#20H	Temporizador 1 no modo de recarga automática (modo 2)
	MOV TH1,#00H	Carrega registrador de recarga TH1 com valor 00h
	MOV TL1,TH1	Transfere para TL1 o conteúdo de TH1. TL1 é o contador no modo 2.
	SETB TR1	Dá início à contagem do temporizador 1
	SJMP \$	Fica num <i>loop</i> infinito, saindo a cada fim de contagem do temporizador 1
ATENDE1:	CLR TF1	Limpa a flag de final de contagem
	CPL P1.0	Complementa o bit zero da porta P1 a cada fim de contagem do TEMP1
	INC TH1	Incrementa TH1. Cada nova contagem começa 1 unidade acima da anterior
	RETI	Retorna da subrotina que atende a interrupção do temporizador 1.
	END	

Exemplo 8: Uso do temporizador 1 no modo 2, com interrupção. O programa gera no pino P1.0 uma onda cujos tempos ligado e desligado variam devido à variação do valor de recarga TH1.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 1BH	Endereço de desvio para a interrupção do temporizador 1.
	LJMP ATENDE1	Desvia para a subrotina que atende o temporizador 1.
	ORG 30H	
INICIO:	MOV SP, #2FH	
	MOV IE,#88H	Habilita a interrupção do temporizador 2

	MOV TMOD,#20H	Temporizador 1 no modo de recarga automática (modo 2)
	MOV TH1,#00H	Carrega registrador de recarga TH1 com valor 00h
	MOV TL1,TH1	Transfere para TL1 o conteúdo de TH1. TL1 é o contador no modo 2.
	CLR F0	Flag que indica se TH1 está sendo incrementado (F0 = 0) ou decrementado
	SETB TR1	Dá início à contagem do temporizador 1
	SJMP \$	Fica num <i>loop</i> infinito, saindo a cada fim de contagem do temporizador 1
ATENDE1:	CPL P1.0	Completa o bit zero da porta P1 a cada fim de contagem do TEMP1
	CLR TF1	Limpa a flag de final de contagem
	JB F0,DECREM	Se F0 = 1, decrementa TH1
	INC TH1	Incrementa valor de recarga TH1
	MOV A,TH1	Transfere TH1 para acumulador para verificar se já alcançou FAH
	CJNE A,#0FAH,SAI	Enquanto TH1 menor que FAH continua aumentando TH1
	SETB F0	Seta flag que indica se TH1 deve ser incrementado ou decrementado
	RETI	
DECREM:	DEC TH1	Decrementa valor de recarga TH1
	MOV A,TH1	Transfere TH1 para acumulador para verifica se já alcançou 05H
	CJNE A,#05H,SAI	Enquanto TH1 maior que 05H, continua diminuindo TH1
	CLR F0	Limpa flag que indica se TH1 deve ser incrementado ou decrementado
SAI:	NOP	
	RETI	
	END	

Exemplo 9: O programa usa a interrupção externa 0, por transição, para alternar entre os modos 0 e 1 a operação do temporizador 0. A flag “**F0**”, do registrador especial **PSW** (F0 = PSW.5), é usada para a alternância entre os dois modos de operação.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 03H	Endereço de desvio para a interrupção externa 0
	LJMP MUDA	Desvia para a subrotina que define o modo de operação do temporizador
	ORG 0BH	Endereço de desvio para a interrupção do temporizador 0
	CPL P1.0	Gera uma onda quadrada no pino P1.0. Período depende do modo 0 ou 1
	CLR TF0	Limpa a flag de final de contagem
	RETI	
	ORG 30H	
INICIO:	MOV SP,#2FH	Apontador de pilha SP = 2FH
	MOV TMOD,#00H	Temporizador 0 no modo 0
	MOV IE,#83H	Habilita interrupção do temporizador 0 e da interrupção externa 0
	MOV TCON,#01H	Interrupção externa 0 é por transição
	SETB TR0	Dispara temporizador 0
	SJMP \$	Fica em um <i>loop</i> infinito, saindo apenas quando há interrupção
MUDA:	CPL F0	Limpa a flag que indica modo 0 (F0=0) ou modo 1 (F0=1)
	JB F0, MODO_UM	Se F0=1, desvia para executar modo 1. Se F0=0, executa modo 0
MODO_ZERO:	MOV TMOD,#00H	Configura temporizador 0 para operar no modo 0 (13 bits)
	RETI	Retorna da interrupção
MODO_UM:	MOV TMOD,#01H	Configura temporizador 0 para operar no modo 1 (16 bits)
	RETI	Seta flag que indica se TH1 deve ser incrementado ou decrementado
	END	

Exemplo 10: O programa usa o temporizador 2 para gerar uma onda quadrada no pino P1.0 a cada 50.000 pulsos. Os registradores relacionados ao temporizador 2 são definidos no início do programa porque não estão presentes na biblioteca “MOD51”.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	T2MOD EQU 0C9H ; Registrador de Configuração do Temporizador 2	
	RCAP2L EQU 0CAH ; Byte Inferior do registrador de recarga do Temporizador 2	
	RCAP2H EQU 0CBH ; Byte Superior do registrador de recarga do Temporizador 2	
	TR2 BIT 0CAH ; Bit de disparo do Temporizador 2 (=T2CON.2)	
	TF2 BIT 0CFH ; Bit que indica fim de contagem do Temporizador 2 (=T2CON.7)	
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	Apontador de pilha SP = 2FH
	MOV T2MOD,#01H	Configura Temporizador 2 para operar crescente ou decrescente
	MOV RCAP2H,#3CH	Byte superior de recarga do temporizador 2
	MOV RCAP2L,#0AFH	Byte inferior de recarga do temporizador 2
	SETB TR2	Dispara do temporizador 2
V1:	JNB TF2,V1	Aguarda fim da contagem do temporizador 2
	CPL P1.0	Complementa pino P1.0. Gera onda quadrada
	CLR TF2	
	SJMP V1	
	END	

A Fig. 5.8 mostra a onda quadrada usada com o temporizador 2. Esse temporizador funciona com recarga automática. Foi definido o valor 3CAFH (15.535) como recarga para gerar 50.000 pulsos a cada contagem. Sendo o cristal oscilador de 12 MHz, o ciclo de clock do temporizador é 1 μ s e cada contagem corresponde a 50 ms.

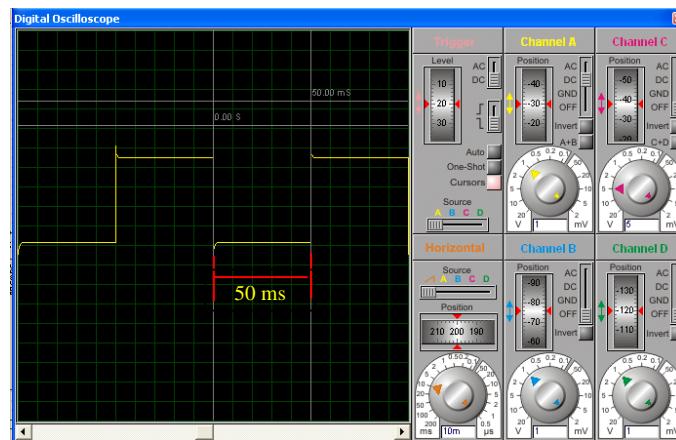


Fig. 5.8: Onda quadrada gerada pelo temporizador 2

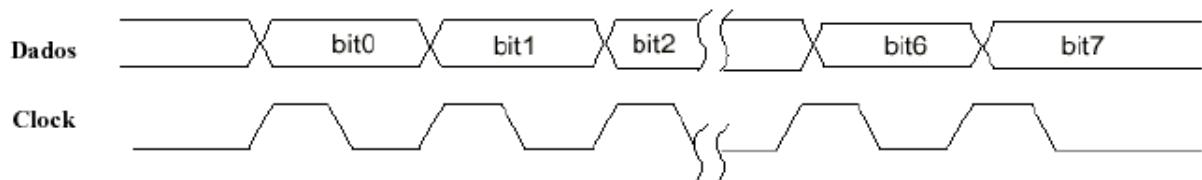
6 Comunicação Serial

6.1 Noções Básicas de Comunicação Serial

A comunicação serial consiste em enviar ou receber pacotes de informação bit a bit. No caso do 8051 o canal de comunicação serial é do tipo “full duplex”, o que significa que ele pode, ao mesmo tempo, receber e transmitir dados.

Uma grande questão da transmissão serial é como informar o receptor do início e do final do pacote de informação, ou seja, qual o primeiro bit da informação e qual o último. Assim, existem dois tipos de comunicação: síncrona e assíncrona.

Na comunicação serial síncrona, são utilizados dois canais: um para transmitir e receber os dados e outro para transmitir um sinal de sincronismo. O transmissor, portanto, é o responsável pela sincronização. A cada sinal de sincronismo recebido o receptor lê o canal de dados.



No caso do 8051 a transmissão e também a recepção síncrona de dados são feitas através do pino RxD (pino P3.0). O pino TxD (pino P3.1) é usado para o sinal de sincronismo. Na transmissão serial síncrona via MODEM, que não é o caso tratado aqui, a informação de sincronismo é enviada junto com os dados. Para isso existem técnicas especiais de codificação, que não são tratadas nesta apostila.

Na comunicação assíncrona não há um sinal de sincronismo e, portanto, alguns cuidados especiais devem ser tomados:

- As taxas de recepção e de transmissão devem ser iguais
- Um bit de início da transmissão deve ser enviado
- Um bit de fim de transmissão deve ser enviado

Assim, um pacote de informações pode ser ilustrado como a seguir:

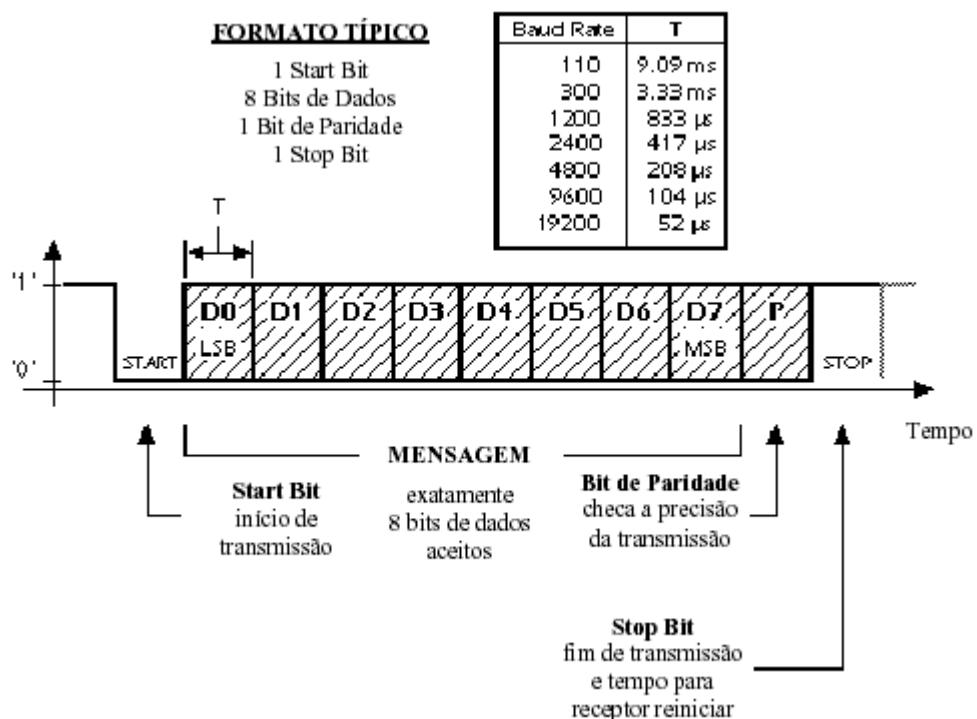


Fig. 6.1: Pacote de informações na comunicação serial assíncrona

Observe que o bit de início de transmissão é zero, isto porque o canal normalmente fica em repouso no nível lógico alto. Assim, a primeira passagem para zero, após a habilitação da transmissão, é

interpretada como o sinal de início. O sinal de parada é de nível lógico alto, após ser recebida a quantidade de bits especificada. Observe ainda que, além do bit de início (Start bit) e do bit de fim (Stop bit), também pode existir um terceiro bit extra, que é o bit de paridade, usado para verificar a consistência dos dados. Assim, se houver erro na transmissão que implique na alteração da paridade, esse bit extra detecta o erro.

A verificação de erros através do bit de paridade não detecta erro quando há inversão, por exemplo, de dois bits de dados. Assim, normalmente são utilizados outros métodos de verificação de erros, principalmente quando a informação é constituída de vários bytes. São utilizados ainda os chamados protocolos de comunicação, que garantirão uma transmissão serial mais segura.

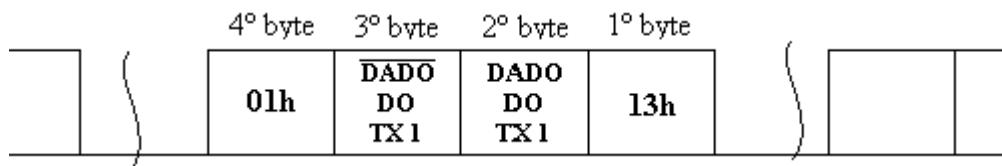
Um método utilizado para detecção de erros na transmissão de vários bytes é o método do “checksum”, onde o complemento de 2 da soma dos dados é acrescentada ao pacote de informações. Assim, na recepção, o processador adiciona todos os bytes e verifica se o resultado é zero. Caso não seja, houve um erro na transmissão. Veja a exemplificação abaixo:

$$\begin{array}{r}
 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1 \\
 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0 \\
 +\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \text{ Dados} \\
 \hline
 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0
 \end{array}$$

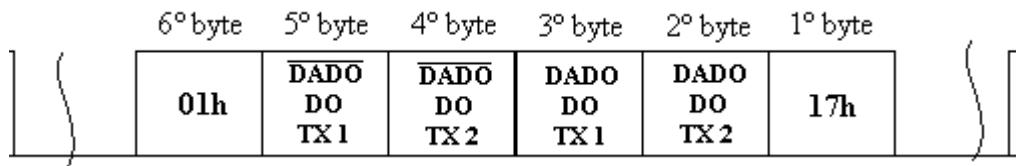
0 0 1 1 0 0 1 0 0 0 1 0 Soma Aritmética

$$\begin{array}{r}
 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0 \text{ Soma truncada - 8 bits} \\
 +\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0 \text{ Checksum (complemento de 2)} \\
 \hline
 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \text{ Soma + Checksum = 0}
 \end{array}$$

No projeto de irrigação desenvolvido na EEEC/UFG dois pacotes de informação são utilizados: no primeiro pacote cada byte de dados (pressão numa tubulação de água) do transmissor 1 é enviado juntamente com seu complemento. O primeiro byte é um byte que indica início dos dados (foi escolhido o byte 13h) e o último byte indica fim do pacote (foi escolhido o byte 01h).



O segundo pacote de informações é composto pelos dados do transmissor 1 e os dados do transmissor 2. Para o byte de início foi escolhido o byte 17h e o byte de fim escolhido foi o byte 01h. Novamente o complemento dos dados é enviado. Na recepção os dados são verificados. Se for detectado erro de transmissão, os dados são enviados novamente.



É necessário destacar outro ponto da comunicação serial: normalmente a comunicação serial entre dois dispositivos se dá usando o padrão ASCII. Um exemplo é o display de cristal líquido (LCD). Para mostrar o número “1” no display LCD deve-se enviar seu código ASCII, ou seja, “31h”. Da mesma forma, na comunicação entre o computador e o 8051, o código ASCII é utilizado. A tabela ASCII é mostrada a seguir.

Tabela 6.1: Códigos ASCII

HEX	DEC	CHR									
00	00	NUL	20	32	SPC	40	64	@	60	96	`
01	01	SOH	21	33	!	41	65	A	61	97	a
02	02	STX	22	34	"	42	66	B	62	98	b
03	03	ETX	23	35	#	43	67	C	63	99	c
04	04	EOT	24	36	\$	44	68	D	64	100	d
05	05	ENQ	25	37	%	45	69	E	65	101	e
06	06	ACK	26	38	&	46	70	F	66	102	f
07	07	BEL	27	39	'	47	71	G	67	103	g
08	08	BS	28	40	(48	72	H	68	104	h
09	09	HT	29	41)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91		7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	
1D	29	GS	3D	61	=	5D	93	_	7D	125	}
1E	20	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	—	7F	127	DEL

Tabela 6.2: Registrador de Controle da Comunicação Serial

$$(SCON) = \boxed{SM0 \ SM1 \ SM2 \ REN \ TB8 \ RB8 \ TI \ RI}$$

<u>SM0</u>	<u>SM1</u>	<u>Modo</u>	<u>Descrição</u>	<u>Baud Rate</u>
0	0	0	Registrador de Deslocamento	$f_{osc}/12$
0	1	1	UART de 8 bits	variável
1	0	2	UART de 9 bits	$f_{osc}/64$ ou $f_{osc}/32$
1	1	3	UART de 9 bits	variável
<u>Símbolo</u>			<u>Nome e Significado</u>	
SM2			Habilita a característica de comunicação de multiprocessadores no modo 2 e 3. Nesses modos, se SM2=1, RI não será ativado se o nono bit de dado recebido for igual a 0. No modo 1, se SM2=1, RI não será ativado se um stop bit válido não for recebido. No modo 0, deverá ser 0.	
REN			Bit habitador da recepção serial. Setado/limpado por <i>software</i> para habilitar ou desabilitar a recepção serial.	
TB8			É o nono bit de dado que será transmitido no modo 2 e 3. Setado ou limpado por <i>software</i> .	
RB8			No modo 2 e 3, é o nono bit de dado que foi recebido. No modo 1, se SM2=0, RB8 é o stop bit que foi recebido. No modo 0, RB8 não é usado.	
TI			É o flag de interrupção de transmissão. Setado por <i>hardware</i> no final do tempo do 8º bit no modo 0 ou no início do stop bit em outros modos, em qualquer transmissão serial. Deverá ser limpado por <i>software</i> .	
RI			É o flag de interrupção de recepção. Setado por <i>hardware</i> no final do tempo do 8º bit no modo 0 ou na metade do tempo do stop bit em outros modos, em qualquer recepção serial. Deverá ser limpado por <i>software</i> .	

O canal serial do 8051 pode operar em 4 modos diferentes, definidos através do registrador especial **SCON** (mostrado na tabela 6.2), cujos bits podem ser manipulados individualmente. O modo 0 é uma comunicação síncrona. Os demais modos são do tipo assíncrono.

Modo 0:

Taxa de transmissão fixa e igual à frequência de clock dividida por 12. A recepção tem início com **REN = 1** e **RI = 0**. Ao final da recepção o bit **RI** é setado por hardware e o conteúdo recebido é transferido para um registrador denominado por **SBUF**. **RI** deve ser ressetado antes da próxima recepção.

A transmissão é iniciada automaticamente quando o conteúdo do acumulador é transferido para o **SBUF**. Quanto ao registrador **SBUF** da transmissão, embora tenha o mesmo nome do registrador da recepção, trata-se de um outro registrador específico para a transmissão. Ao final da transmissão o bit **TI** é setado por hardware. **TI** deve ser ressetado antes da próxima transmissão.

Modo 1:

Comunicação assíncrona com taxa de transmissão a ser definida pelo usuário. São transmitidos 8 bits de dados, além de um bit de início e um bit de fim. A recepção tem início quando há uma transição do nível lógico 1 para 0 no pino **RxD** (pino P3.0) e o bit **RI** está zerado. Ao final da recepção o **RI** é setado por hardware.

A transmissão é iniciada quando há uma transferência de dados para **SBUF**. Ao final da transmissão o bit **TI** é setado.

Modo 2:

Modo assíncrono onde 11 bits são transmitidos (bit de início + 9 bits de dados + bit de fim). A taxa de transmissão/recepção pode ser 1/32 ou 1/64 da frequência de clock. A recepção tem início quando há uma transição do nível lógico 1 para 0 no pino **RxD** (pino P3.0) e o bit **RI** está zerado. Ao final da recepção o bit **RI** é setado por hardware. O nono bit de dados é guardado em **RB8**. Esse bit pode ser o bit de paridade.

A transmissão é iniciada quando há uma transferência de dados para **SBUF**. Ao final da transmissão o bit **TI** é setado. O nono bit a ser transmitido é guardado em **TB8**. Esse bit pode ser o bit de paridade. Expressão que define a taxa de transmissão no modo 2.

$$\text{Baud rate} = 2^{\text{SMOD}} * \frac{f_{\text{osc}}}{64} (\text{bits / s})$$

Se o bit **SMOD** for zero, a taxa é 1/64, caso seja igual a 1, a taxa é 1/32.

Modo 3:

É semelhante ao modo 1; a diferença está no bit a mais de dados no modo 3. Tanto no modo 1 quanto no modo 3 a taxa de transmissão é definida pelo usuário, seguindo a equação a seguir:

$$\text{Baud rate} = \frac{2^{\text{SMOD}}}{32} * \frac{f_{\text{osc}}}{12 * (256 - \text{TH1})} (\text{bits / s})$$

Nesses dois modos (1 e 3) o temporizador 1 deve ser configurado para operar no modo de recarga automática (modo 2). O valor da recarga (**TH1**), juntamente com o bit **SMOD**, é que define a frequência de comunicação (transmissão e recepção). A tabela a seguir fornece alguns dos valores mais comuns de taxa de transmissão.

Tabela 6.3: Taxas de transmissão mais comuns

<u>Baud Rate (bits/seg)</u>	<u>Freq. Osc. (MHz)</u>	<u>SMOD</u>	<u>C/Tbarra</u>	<u>Modo</u>	<u>Valor Recar.</u>	<u>Timer 1</u>
Modo 0 Máx: 1MHz	12	X	X	X	X	
Modo 2 Máx: 375K	12	1	X	X	X	
Modo 1, 3: 62,5K	12	1	0	2	FFh	
19,2K	11,059	1	0	2	FDh	
9,6K	11,059	0	0	2	FDh	
4,8K	11,059	0	0	2	FAh	
2,4K	11,059	0	0	2	F4h	
1,2K	11,059	0	0	2	E8h	
137,5	11,059	0	0	2	1Dh	
110	6	0	0	2	72h	
110	12	0	0	1	FEEBh	

6.2 Roteiros de Programas usando Comunicação Serial

1. No programa a seguir a porta serial é configurada no **modo 0** e usada para transmitir uma contagem crescente decimal, que é também enviada para a porta P1, onde um conjunto de LEDs pode ser usado para mostrar a contagem. O pino **P3.0 (RxD)** é usado para a transmissão dos dados e o pino **P3.1 (TxD)** é usado para a frequência de clock.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV R0,#00H	; Registrador R0 assume o valor 0
V1:	MOV SBUF,R0	; Transfere o conteúdo de R0 para o Registrador SBUF
	JNB TI,\$; Aguarda o final da transmissão do conteúdo de R0
	MOV P1,R0	; Transfere para a porta P1 o conteúdo de R0
	MOV A,R0	; Transfere para o acumulador o conteúdo de R0
	ADD A,#01	; Adiciona 1 ao conteúdo do acumulador
	DA A	; Faz o ajuste decimal do conteúdo do acumulador
	MOV R0,A	; Devolve para R0 o conteúdo atualizado de R0
	LCALL ATRASO	; Chama subrotina de atraso de tempo
	SJMP V1	; Volta para transferir o valor atualizado de R0
ATRASO:	MOV R4,#100	
V2:	MOV R5,#250	
	DJNZ R5,\$	
	DJNZ R4,V2	
	RET	
	END	

A Fig. 6.2 mostra um instante da contagem. No instante mostrado o valor transmitido corresponde a 12 H (0 0 0 1 0 0 1 0). O bit menos significativo da contagem está à esquerda na tela do osciloscópio. O canal TxD transmite a frequência de clock. Como o cristal oscilador, nesse caso, é 12 MHz, a taxa de transmissão no modo síncrono é 1 MHz, o que resulta em um período de 1 μ s.

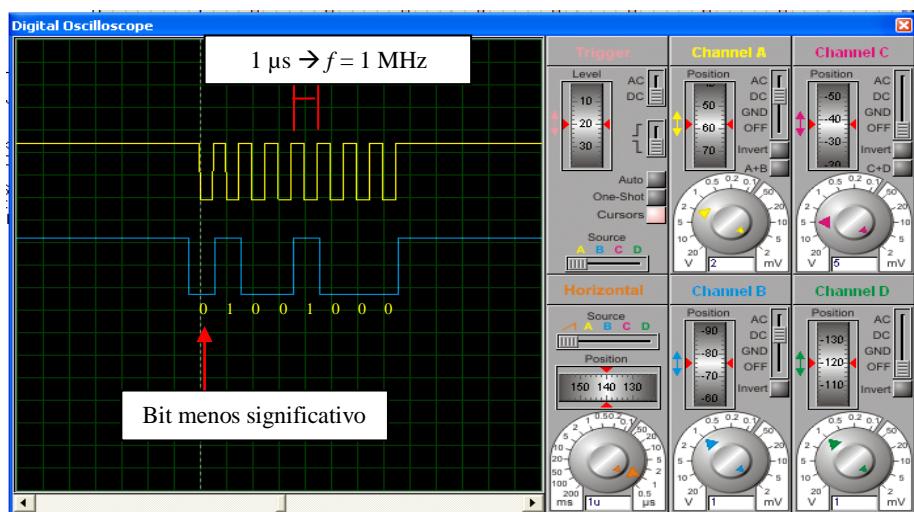


Fig. 6.2: Transmissão síncrona de contagem crescente

2. O programa a seguir é equivalente ao anterior, mas utiliza o modo 1 (assíncrono) de comunicação serial com baud rate de **9600 bps**. Nesse caso, o cristal oscilador é de **11,0592 MHz**, o que resulta (da Tabela 6.3) em um valor de recarga **TH1 = FDH**, para o temporizador 1 no modo 2 (recarga automática).

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV SCON,#40H	; Configura serial para modo 1 assíncrono
	MOV TMOD,#20H	; Configura o temporizador 1 para operar no modo 2
	MOV TL1,#0FDH	; Faz TL1 = FDH → baud rate de 9600 bps para $f = 11,0592 \text{ MHz}$
	MOV TH1,#0FDH	; Carrega TH1 com o valor de recarga automática
	MOV R0,#00H	; Faz R0 = 0 para iniciar contagem
	SETB TR1	; Adiciona 1 ao conteúdo do acumulador
V1:	MOV SBUF,R0	; Transfere conteúdo de R0 para o registrador SBUF da serial
	JNB TI,\$; Aguarda final da transmissão do conteúdo de R0
	MOV P1,R0	; Transfere para a porta P1 o conteúdo de R0
	MOV A,R0	; Transfere para o acumulador o conteúdo de R0
	ADD A,#01	; Adiciona 1 ao conteúdo do acumulador
	DA A	; Faz o ajuste decimal do conteúdo do acumulador
	MOV R0,A	; Devolve para R0 o conteúdo atualizado de R0
	LCALL ATRASO	; Chama subrotina de atraso de tempo
	SJMP V1	; Volta para transferir o valor atualizado de R0
ATRASO:	MOV R4,#100	
V2:	MOV R5,#250	
	DJNZ R5,\$	
	DJNZ R4,V2	
	RET	
	END	

A Fig. 6.3 mostra um instante da contagem. No instante mostrado o valor transmitido corresponde a 37 H (0 0 1 1 0 1 1 1). O bit mais à esquerda no osciloscópio é o bit de “start”, cujo valor é “0”. O bit mais à direita é o bit de “stop”, cujo valor é “1”. O bit menos significativo da contagem é o primeiro bit à direita do bit de “start”. No modo assíncrono, como trata-se de transmissão, o canal TxD é usado. Como a taxa de transmissão escolhida é 9600 bps, um bit corresponde a 104 μs .

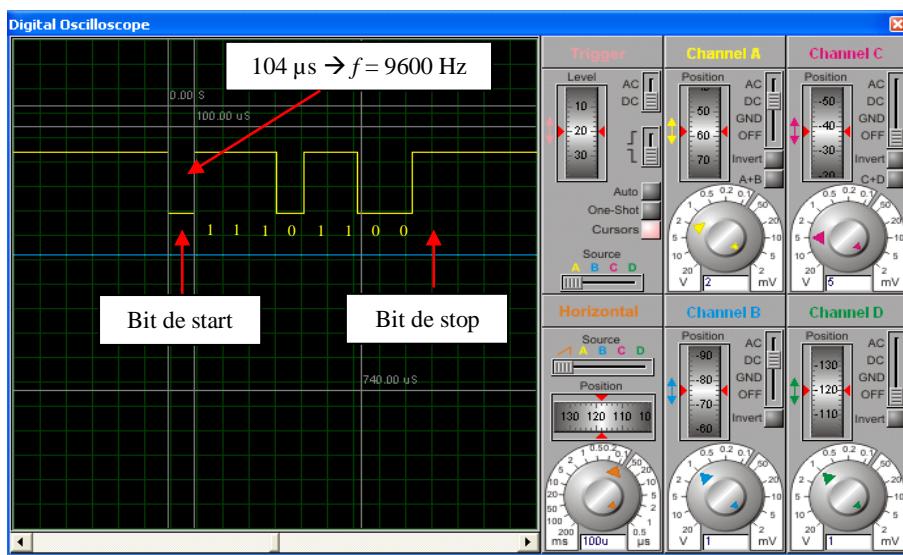


Fig. 6.3: Transmissão assíncrona de contagem crescente

3. No programa a seguir o microcontrolador recebe dados via serial no modo 1, com taxa de recepção de **4800 bps**, usando interrupção. O dado recebido é enviado para a porta P1. A frequência do cristal oscilador é de **11,0592 MHz**. Assim, da Tabela 2, tem-se o valor de recarga **TH1 = FAH**, para o temporizador 1 no modo 2 (recarga automática). Caso o programa seja executado no laboratório, use o **HyperTerminal** do microcomputador, configurado para 4800 bps e 8 bits de dados para transmitir os dados para o microcontrolador.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 23H	
	CLR RI	; Limpa flag de recepção
	MOV A,SBUF	; Transfere para o acumulador conteúdo recebido via serial
	RETI	; Retorna da subrotina de atendimento da serial
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV SCON,#40H	; Configura serial para modo 1 assíncrono
	MOV IE,#90H	; Habilita interrupção da serial. IE = 1 0 0 1 0 0 0 0b
	MOV TMOD,#20H	; Configura o temporizador 1 para operar no modo 2
	MOV TL1,#0FAH	; Faz TL1 = FAH → baud rate de 4800 bps para $f = 11,0592 \text{ MHz}$
	MOV TH1,#0FAH	; Carrega TH1 com o valor de recarga automática
	MOV A,#00H	; Carrega acumulador com 0
	SETB TR1	; Dispara temporizador 1
	SETB REN	; Habilita recepção serial
V1:	MOV P1,A	; Transfere para a porta P1 o conteúdo do acumulador
	SJMP V1	; Loop mostrando o conteúdo de A. Esse valor muda a cada recepção
	END	

A Fig. 6.4 mostra no osciloscópio o dado enviado do computador para o microcontrolador. No caso mostrado o dado enviado é o número “5”. Verificar que o valor mostrado é “35H = 0 0 1 1 0 1 0 1b”, que corresponde ao código ASCII do número “5”. Como a taxa de recepção escolhida é 4800 bps, um bit corresponde a 208 μs.

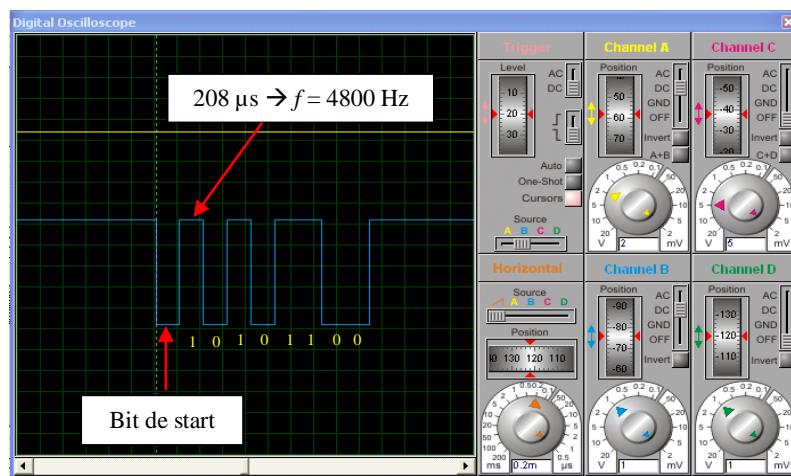


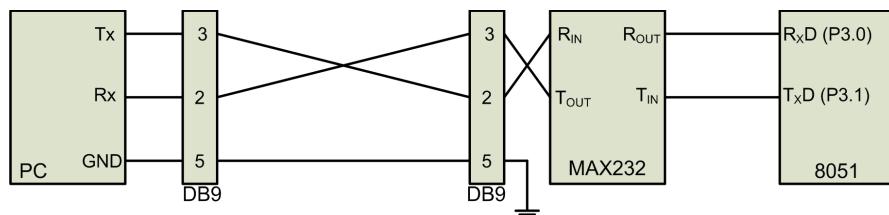
Fig. 6.4: Recepção assíncrona do número “5” (ASCII 35H) enviado pelo computador

4. No programa a seguir o microcontrolador recebe dados via serial no modo 1, com taxa de recepção de **9600 bps**, usando interrupção. O dado recebido é usado para definir a rotação de LEDs na porta P1. Se o dado recebido for a letra “D” os LEDs são rotacionados para a direita. A letra “E” rotaciona os LEDs para a esquerda. Qualquer outra letra ou número faz o programa aguardar. A frequência do cristal oscilador é de **11,0592 MHz**. Assim, da Tabela 2, tem-se o valor de recarga **TH1 = FDH**, para o temporizador 1 no modo 2 (recarga automática).

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 23H	
	CLR RI	; Limpa flag de recepção
	MOV R0,SBUF	; Transfere para o registrador R0 conteúdo recebido via serial
	RETI	; Retorna da subrotina de atendimento da serial
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV SCON,#40H	; Configura serial para modo 1 assíncrono
	MOV IE,#90H	; Habilita interrupção da serial. IE = 1 0 0 1 0 0 0 0b
	MOV TMOD,#20H	; Configura o temporizador 1 para operar no modo 2
	MOV TL1,#0FDH	; Faz TL1 = FAH → baud rate de 4800 bps para $f = 11,0592 \text{ MHz}$
	MOV TH1,#0FDH	; Carrega TH1 com o valor de recarga automática
	MOV R0,#00H	; Carrega registrador R0 com valor 0
	MOV A,#01H	; Carrega acumulador com valor 01H
	SETB TR1	; Dispara temporizador 1
	SETB REN	; Habilita recepção serial
V2:	CJNE R0,#44H,V1	; Se A for diferente de 44H (letra D) verificar se é 45H
	LJMP DIREITA	; Se A = 44H (letra D), desvia para rotina que rotaciona LEDs para a direita
V1:	CJNE R0,#45H,V2	; Se A for diferente de 45H (letra E) voltar para aguardar novo valor
	LJMP ESQUERDA	; Se A = 45H (letra E), desvia para rotina que rotaciona LEDs para a esquerda
DIREITA:	MOV P1,A	; Transfere para a porta P1 o conteúdo do acumulador
	RR A	; Rotaciona para a direita conteúdo do acumulador
	LCALL ATRASO	
	SJMP V2	
ESQUERDA:	MOV P1,A	
	RL A	; Rotaciona para a esquerda o conteúdo do acumulador
	LCALL ATRASO	
	SJMP V2	
ATRASO:	MOV R7,#100	
V3:	MOV R6,#250	
	DJNZ R6,\$	
	DJNZ R7,V3	
	RET	
	END	

Observações:

1. A conexão serial entre o microcontrolador 8051 e um computador exige um componente para adaptação dos níveis de tensão. Enquanto no microcontrolador tensão zero representa nível lógico 0 e tensão de 5 V representa o nível lógico 1, no computador é diferente. No computador o nível lógico 0 é representado por uma tensão de + 12 V e o nível lógico 1 é representado por uma tensão de – 12 V. Essa adaptação entre os níveis de tensão é conseguida com o componente MAX232.
2. O cabo de conexão entre o microcontrolador e o PC pode ser do tipo direto, ou do tipo invertido, conforme mostra o diagrama a seguir.



7 Expansão e Mapeamento de Memória

O microcontrolador básico da família 8051 não possui memória EEPROM interna para o armazenamento de dados. Assim, quando há necessidade de armazenamento de dados, por exemplo, valores de temperatura e umidade ao longo de uma semana, é necessário acrescentar ao projeto uma pastilha de memória EEPROM externa.

Além disso, se houver necessidade de vários periféricos, tais como display de LCD, teclado, display de 7 segmentos e outros, pode acontecer de não ter portas suficientes para todos esses componentes. Nesse caso, há necessidade de um mapeamento de memória, ou seja, acessar esses componentes como se fossem posições de memória. Assim, todos os componentes, inclusive a memória externa são acessados através das portas P2 e P0, com o uso da instrução **MOVX**. A porta P0 transporta os dados e o byte inferior do endereço e a porta P2 transporta o byte superior do endereço.

O chip AT89S8252 possui 2 K de EEPROM interna para o armazenamento de dados. Dessa forma, se a quantidade de dados que se deseja armazenar for pequena, não há necessidade de acrescentar uma memória EEPROM externa. A instrução **MOVX** também é usada para acessar essa memória. O registrador especial WMCON (endereço 96H), mostrado na tabela 7.1, é usado para controle.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WMCON	PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN

Tabela 7.1: Descrição dos bits do registrador WMCON.

Bit	Descrição
PS2 PS1 PS0	Bits que definem o tempo para o modo Watchdog. Quando os três bits são 0, o tempo do watchdog é de 16 ms. Quando os três bits estão em nível alto (valor 1), o tempo é de 2048 ms.
EEMWE	Bit de habilitação da escrita na memória EEPROM interna. Deve ser setado (EEMWE=1) antes de uma instrução de escrita na memória com a instrução MOVX . Depois da escrita na memória deve ser limpo (EEMWE=0).
EEMEN	Bit que habilita o acesso à memória EEPROM interna. Fazendo EEMEN = 1, a instrução MOVX com o DPTR é usado para acessar a EEPROM interna. EEMEN = 0 dá acesso à memória EEPROM externa.
DPS	Bit de seleção do apontador de dados. DPS = 0 habilita o uso do primeiro banco, DP0, de apontador de dados. DPS = 1 seleciona o segundo banco, DP1, de apontador de dados.
WDTRST RDY/BSY\	Bit de reset do Watchdog e flag de Ready/Busy\ da EEPROM interna. Cada vez que esse bit é setado, um pulso de reset do watchdog é gerado. Em seguida o bit WDTRST é automaticamente reset para “0” no ciclo de instrução seguinte. Esse bit é do tipo Write-Only (somente escrita). Este bit também serve como flag de RDY/BSY\ (pronto/ocupado) durante operação de escrita na EEPROM interna. RDY/BSY\ = 1 significa que a EEPROM está pronta para ser programada. Enquanto as operações de programação estão sendo executadas, a flag RDY/BSY\ permanece igual a zero. Ela torna-se automaticamente 1 quando a programação é completada.
WDTEN	Bit de habilitação do watchdog. WDTEN = 1 habilita o temporizador watchdog e WDTEN = 0 desabilita.

O programa-exemplo a seguir usa a memória EEPROM interna para armazenar dados que chegam ao microcontrolador através do nibble superior porta P3 (dados digitados através de um teclado de 16 teclas – códigos de 0 a FH). Os dados são lidos a cada pedido da interrupção externa 0, por transição. Uma vez que os dados chegam através do nibble superior da porta P3, e o nibble inferior não faz parte dos dados, é feita uma operação de troca do nibble superior pelo inferior, através da instrução SWAP e, em seguida, é feita uma operação AND com “0FH”. Assim, o resultado final é o dado digitado através do teclado. A memória EEPROM do chip AT89S8252 é de 2 K e, portanto, seu endereço vai de 0000 H até 07FFH.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	EEMWE BIT 9AH;	Atribui ao bit WMCON.4 o nome de EEMWE
	ORG 00H	
	LJMP INICIO	
	ORG 03H	
	LJMP ATENDE	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV IE,#81H	; Habilita interrupção externa 0. IE = 1 0 0 0 0 0 0 1 b
	MOV TCON,#01H	; Interrupção externa 0 por transição
	MOV DPTR,#0000H	; Carrega DPTR com endereço inicial da memória EEPROM
	SJMP \$	
ATENDE:	MOV A,P3	; Carrega no acumulador o conteúdo da porta P1
	SWAP A	; Troca o nibble superior pelo nibble inferior do acumulador
	ANL A,#0FH	; Elimina o nibble superior usando uma operação AND
	SETB EEMWE	; Habilita a escrita na memória EEPROM interna
	MOVX @DPTR,A	; Escreve na memória EEPROM interna o conteúdo do acumulador
	CLR EEMWE	; Desabilita escrita na memória
	INC DPTR	; Prepara DPTR para armazenar o próximo dado
	RETI	; Retorna da interrupção
	END	

7.1 Expansão de Memória

A Figura 7.1 mostra o esquemático para uma memória externa de 8 K. São necessárias 13 linhas de endereço ($AD_0 \dots AD_{12}$) para acessar todas as 8.192 posições dessa memória. Assim, o endereço interno da memória vai de 0000H até 1FFFH. No entanto, o intervalo de endereço para acesso à memória não precisa, necessariamente, começar em 0000H. Ele pode começar, por exemplo, da posição 2000H, 4000H, E000H, ou outras posições que se desejar. Essa escolha é feita através de linhas de endereços da porta P2. Na Figura 7.1 o endereço escolhido foi 6000H, uma vez que são usadas as linhas de endereço A_{13} e A_{14} para habilitar o latch de endereços 74LS573.

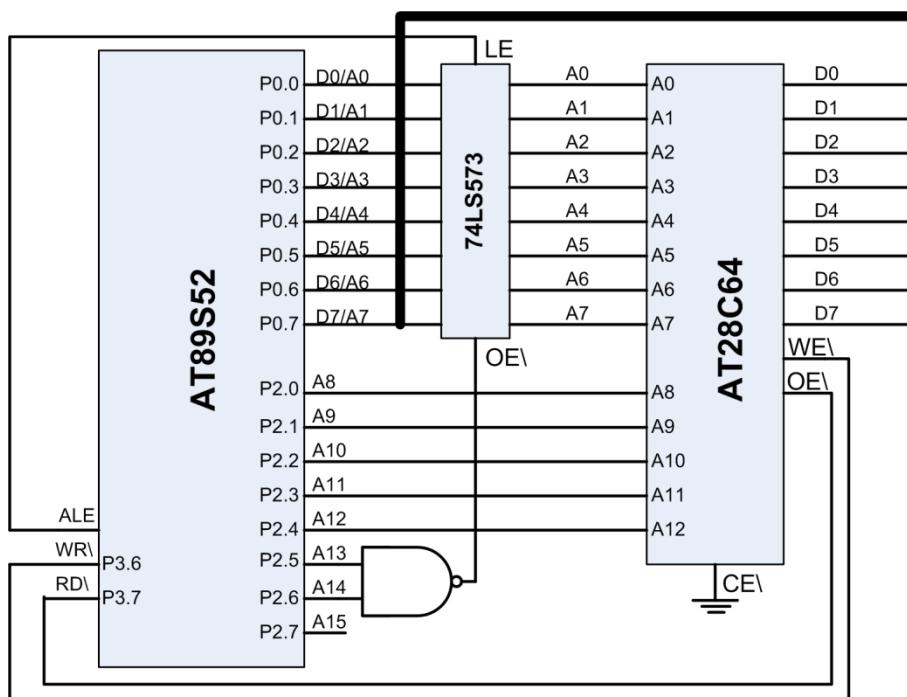


Fig. 7.1: Memória externa com endereço inicial de 6000H

No diagrama da Fig. 7.1 há sombras de memória, uma vez que o pino A_{15} ficou livre, podendo assumir valores 0 ou 1, sem alterar a posição efetiva de memória. Por exemplo, desejando-se acessar a posição 6500H da memória EEPROM mostrada, pode-se usar o endereço 6500H (0110 0101 | 0000 0000) ou o endereço E500H (1110 0101 | 0000 0000), uma vez que o bit A_{15} não afetará o endereçamento. Para eliminar essa sombra de memória pode-se utilizar uma porta NAND de três entradas, como mostrado na Fig. 7.2. Nesse caso, a saída só estará habilitada quando $A_{13} = 1$, $A_{14} = 1$ e $A_{15}\backslash = 0$.

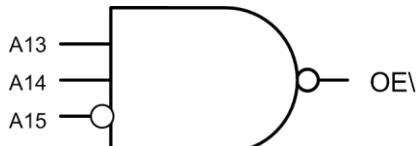


Fig. 7.2: Eliminação de sombras de memória do diagrama da figura 7.1

As linhas de endereço são usadas para habilitar a saída do latch ($OE\backslash$). No entanto, outros sinais de controle estão envolvidos no processo. O sinal de ALE é usado para habilitar o latch (74LS573) e os sinais de READ (RD \backslash) e WRITE (WR \backslash) são usados na habilitação de leitura e escrita da memória EEPROM. A Tabela 7.2 mostra as várias possibilidades de alocação da memória EEPROM da Fig. 7.1 no espaço de 64 K, usando as três linhas de endereço disponíveis A_{13} , A_{14} e A_{15} e a Tabela 7.3 mostra o estado da memória a partir do sinal de controle RD \backslash , conectado ao pino de habilitação de saída da memória ($OE\backslash$) e do sinal WR \backslash , conectado ao pino de habilitação de escrita na memória (WE \backslash).

Tabela 7.2: Regiões de memória

A15	A14	A13	Endereço Inicial	Endereço Final	Região
A15\	A14\	A13\	0000H	1FFFH	0
A15\	A14\	A13	2000H	3FFFH	1
A15\	A14	A13\	4000H	5FFFH	2
A15\	A14	A13	6000H	7FFFH	3
A15	A14\	A13\	8000H	9FFFH	4
A15	A14\	A13	A000H	BFFFH	5
A15	A14	A13\	C000H	DFFFH	6
A15	A14	A13	E000H	FFFFH	7

Tabela 7.3: Estado da memória

RD\	WR\	Estado da memória EEPROM
0	0	Estado inexistente
0	1	Escrita na memória
1	0	Leitura da memória
1	1	Operações internas do microprocessador

7.2 Mapeamento de Memória

A Tabela 7.2 mostra 8 regiões da memória que podem ser selecionadas através das linhas de endereço A_{13} , A_{14} e A_{15} , cada uma com 8 K. Escolhendo-se a região 0 para a memória EEPROM AT28C64, de 8 K, as outras regiões podem ser usadas para outros componentes externos, tais como display LCD, display de 7-segmentos, conjunto de LEDs e conversor analógico/digital.

A Fig. 7.3 mostra um diagrama onde estão presentes a memória EEPROM de 8 K, alocada na região 0 (início em 0000H); um conjunto de 8 LEDs, alocado na região 1 (início em 2000H) e um display LCD, alocado na região 2 (início em 4000H). Vale destacar que o conjunto de LEDs precisa do latch porque não constitui um circuito com pino de habilitação. Assim, sem o latch, todos os dados presentes no barramento seriam mostrados nos LEDs, mesmo aqueles relativos a endereços e enviados para outros componentes.

Quanto ao display LCD, esse componente tem um pino de habilitação (E) e, a princípio, não precisaria de um latch. No entanto, o funcionamento do LCD depende da definição de endereços para leitura e escrita de instruções e leitura e escrita de dados, como mostrado na Tabela 7.4.

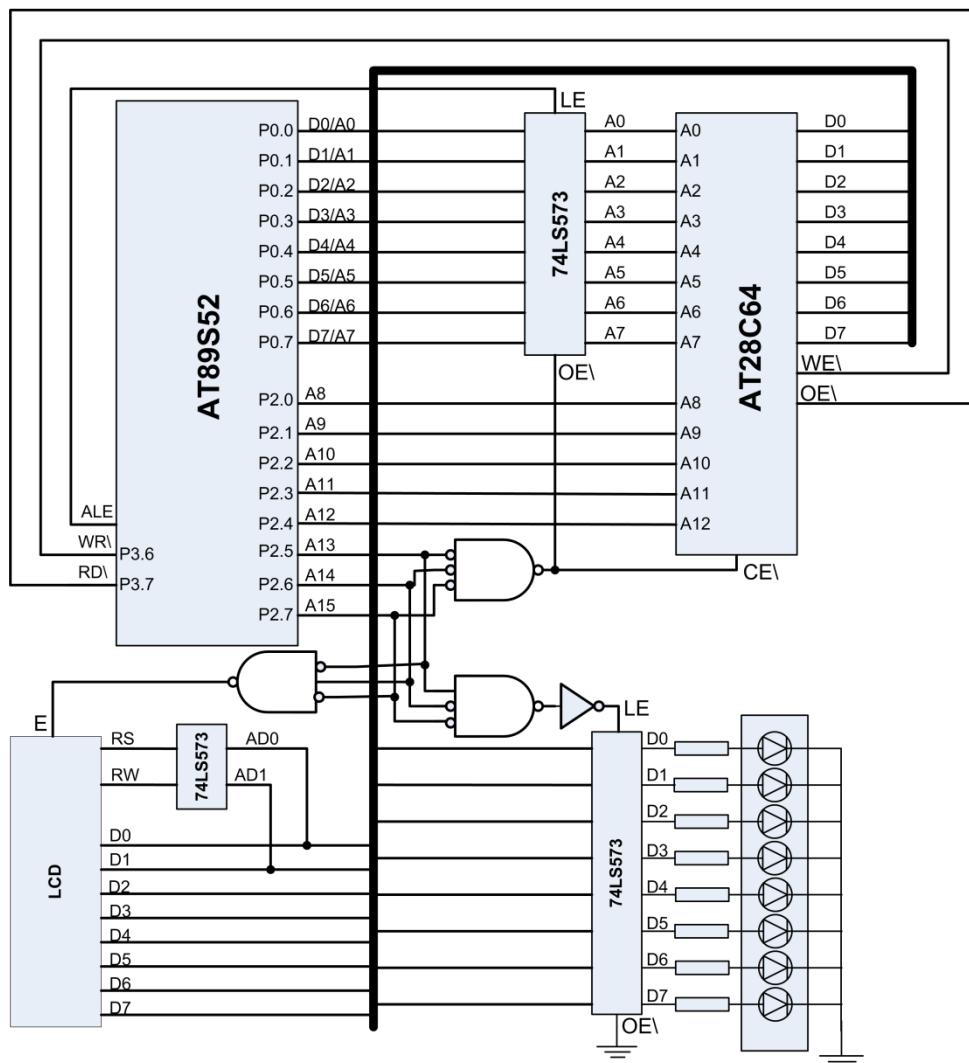


Fig. 7.3: Microcontrolador com memória externa, conjunto de LEDs e display LCD

Tabela 7.4: Endereços do display LCD

RW	RS	Operação	Endereço no Mapeamento da Fig. 7.3
0	0	Escrita de Instrução no LCD	4000H = 0100 0000 0000 0000
0	1	Escrita de Dados no LCD	4001H = 0100 0000 0000 0001
1	0	Leitura de Instrução do LCD	4002H = 0100 0000 0000 0010
1	1	Leitura de Dados do LCD	4003H = 0100 0000 0000 0011

No endereço do mapeamento mostrado na Tabela 7.4 destacam-se os dois primeiros bits, que correspondem aos bits RW e RS, e os três bits mais significativos (A_{15} , A_{14} e A_{13}), usados na habilitação do display LCD.

A opção de usar portas NAND para habilitação dos periféricos da Fig. 7.3 não é a mais adequada quando deseja-se acrescentar mais de um periférico. Uma opção melhor é o uso de um circuito decodificador, por exemplo, o 74LS138, que permite a habilitação de até 8 componentes com três linhas de endereço. O diagrama do 74LS138 é mostrado na Fig. 7.4 e a Tabela-Verdade é mostrada na Tabela 7.5. A saída ativa é baixa.

A Fig. 7.5 mostra as conexões necessárias para o endereçamento dos três componentes mostrados na Fig. 7.3. Os bits de endereço A_{15} , A_{14} e A_{13} são usados para a seleção do componente e os bits RD\ e WR\ são usados para a habilitação do decodificador.

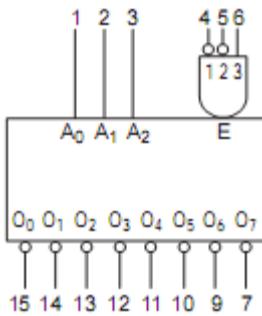


Fig. 7.4: Decodificador 74LS138

Tabela 7.5: Tabela-Verdade do decodificador 74LS138

Entradas						Saídas								
E ₁	E ₂	E ₃	A ₂	A ₁	A ₀	O ₇	O ₆	O ₅	O ₄	O ₃	O ₂	O ₁	O ₀	
1	×	×	×	×	×	1	1	1	1	1	1	1	1	1
×	1	×	×	×	×	1	1	1	1	1	1	1	1	1
×	×	0	×	×	×	1	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1	1

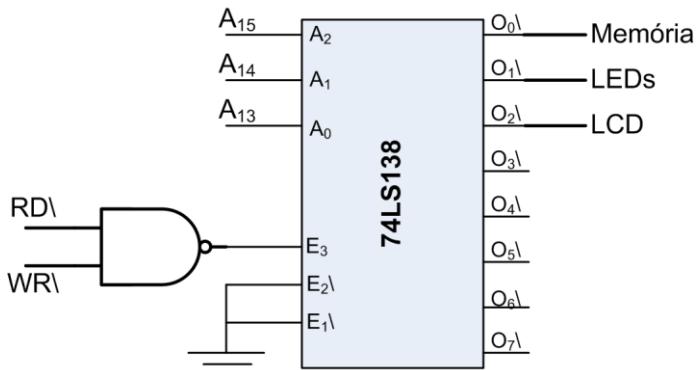


Fig. 7.5: Decodificador para habilitação da memória EEPROM, do conjunto de LEDs e do LCD

No exemplo a seguir um texto é enviado para a memória EEPROM externa. O texto é lido de uma tabela da memória de programa. Cada caractere é lido e, a seguir, enviado para a memória externa. Nesse exemplo o apontador DPTR é usado para duas funções distintas: a primeira é para buscar o caractere do texto na tabela da memória de programa (DPTR fixo e igual ao endereço inicial da tabela) e a segunda é para enviar esse caractere para a memória externa (DPTR com valor inicial 0000H, mas é incrementado a cada novo caractere). São endereços distintos e, por isso, o DPTR relativo à memória, precisa ser guardado na pilha antes da leitura de um novo caractere da tabela.

Esse procedimento de guardar DPTR na pilha, no exemplo mostrado, não é necessário se o microcontrolador utilizado tiver mais de um apontador de dados (DPTR), como é o caso do AT89S8252. Nesse caso pode-se usar apontadores diferentes para cada operação. Observar que o DPTR é guardado na pilha em duas etapas: guarda-se o byte superior (ou inferior) e depois guarda-se o outro byte do apontador. Isso é necessário porque a instrução PUSH guarda apenas um byte por vez na pilha. A recuperação dos valores da pilha é feita na forma inversa ao armazenamento, ou seja, o último valor guardado é recuperado primeiro.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV DPTR,#0000H	; Endereço inicial da memória
	MOV R7,#00	; Valor inicial do contador de leitura do texto
V1:	PUSH DPH	; Guarda na pilha o byte superior do DPTR relativo à memória externa
	PUSH DPL	; Guarda na pilha o byte inferior do DPTR relativo à memória externa
	MOV DPTR,#TEXTO	; Carrega DPTR com endereço inicial da tabela “texto”
	MOV A,R7	; Carrega acumulador com valor atual do contador
	MOVC A,@A+DPTR	; Carrega acumulador com o conteúdo da posição A+DPTR da tabela
	CJNE A,#0FFH,V2	; Verificar se já chegou o final da tabela. Se não, envia caractere para memória
	SJMP FIM	; Desvia para final do programa
V2:	POP DPL	; Recupera da pilha o byte inferior do DPTR relativo à memória externa
	POP DPH	; Recupera da pilha o byte superior do DPTR relativo à memória externa
	MOVX @DPTR,A	; Envia para a memória externa o caractere lido da tabela
	INC R7	; Incrementa contador de leitura da tabela
	INC DPTR	; Incrementa apontador DPTR da memória externa
	SJMP V1	; Volta para ler o caractere seguinte do “texto”
TEXTO:	DB ‘ESTUDAR MICROCONTROLADORES EH GRATIFICANTE’	
	DB 0FFH	
FIM:	NOP	
	END	

No exemplo a seguir faz-se a rotação dos LEDs conectados conforme a Fig. 7.3. Os LEDs estão no endereço 2000H. Na verdade, os LEDs podem ser acessados em toda a faixa de endereços da região 1 (2000H a 3FFFH). seguida, é feita uma operação AND com “0FH”. Assim, o resultado final é o dado digitado através do teclado. A memória EEPROM do chip AT89S8252 é de 2 K e, portanto, seu endereço vai de 0000 H até 07FFH.

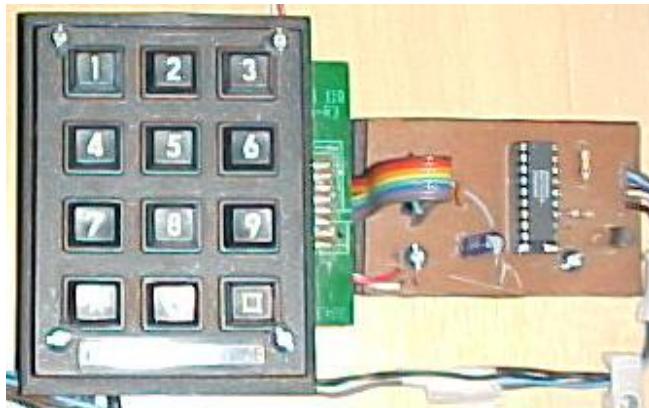
Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV DPTR,#2000H	; Endereço inicial da região 1, que dá acesso aos LEDs
	MOV A,#01H	; Valor inicial do acumulador
V1:	MOVX @DPTR,A	; Envia para os LEDs o conteúdo do acumulador
	LCALL ATRASO	; Chama subrotina de atraso de tempo
	RL A	; Rotaciona o conteúdo do acumulador para a esquerda
	SJMP V1	; Carrega acumulador com valor atual do contador
ATRASO:	MOV R5,#250	; Carrega registrador R5 com valor decimal 250
V2:	MOV R6,#250	; Carrega registrador R6 com valor decimal 250
	DJNZ R6,\$; Aguarda registrador R6 zerar
	DJNZ R5,V2	; Decrementa R5. Enquanto não for zero, volta para recarregar R6
	RET	; Retorna de subrotina de atraso de tempo
	END	

8 Dispositivos para Entrada, Saída e Acionamentos Elétricos

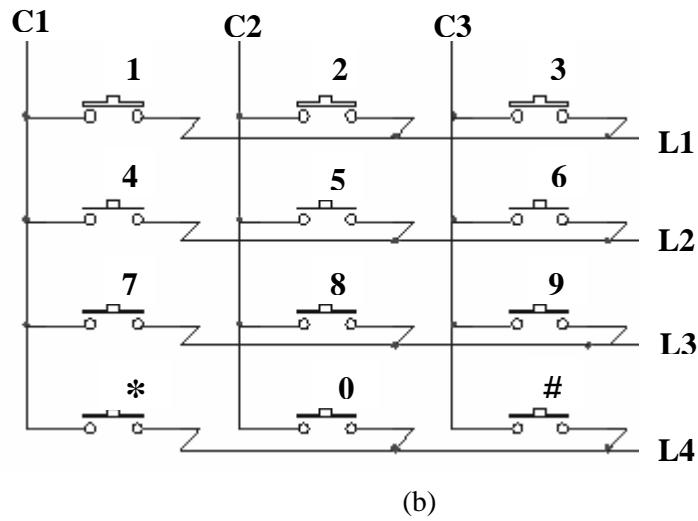
Neste capítulo são apresentados dispositivos de entrada (teclado) e saída de dados (display de 7-segmentos e display LCD) e ainda um sensor de presença e circuitos para acionamento de motores de corrente contínua e motor de passo. Ao mesmo tempo, alguns programas-exemplos são apresentados para ajudar no entendimento do assunto abordado.

8.1 Teclado

A Fig. 8.1(a) mostra um teclado de 4 linhas e 3 colunas e seu circuito decodificador. A Fig. 8.1(b) mostra o esquema básico de conexões para o teclado mostrado.



(a)



(b)

Fig. 8.1: (a) Teclado e (b) esquema matricial de conexões do teclado

A leitura da tecla digitada pode ser feita diretamente, através dos sete canais resultantes da matriz de 4 linhas e 3 colunas ou através de um decodificador de teclado que gera um código de 4 bits a partir do sinal dos sete canais.

No primeiro caso, com leitura direta, pode-se colocar uma das três colunas em nível lógico baixo (**C1**, por exemplo) e monitorar o sinal das quatro linhas. As portas do microcontrolador estão normalmente em nível lógico alto e, assim, se o nível lógico da linha **L1** torna-se baixo, significa que a **tecla 1** foi pressionada. Com uma varredura periódica pode-se detectar cada uma das teclas pressionadas.

Um decodificador que pode ser utilizado para leitura das teclas é o **74LS922**. Nesse caso, o sinal resultante das 4 linhas e 3 colunas originadas do teclado é decodificado em um sinal de 4 bits. Além do código da tecla pressionada o decodificador também gera um bit de “flag” para indicar a disponibilidade

do código na saída. Quando uma tecla é pressionada essa “flag” vai a zero, o que pode ser usado como pedido de interrupção para a leitura do dado disponível.

O decodificador 74LS922 é apropriado para teclado de 16 teclas. No entanto, o teclado da Fig. 8.1 tem apenas 12 teclas, distribuídas em 4 linhas e 3 colunas. Assim, o uso desse decodificador com o teclado mostrado resulta numa saída hexadecimal diferente do número pressionado no teclado, o que exige a utilização de uma tabela-verdade como a mostrada na Tabela 8.1. Verifica-se, por exemplo, que ao pressionar a tecla **1**, o código hexadecimal resultante na saída do decodificador é **0 h** (0000). No entanto, ao se pressionar a tecla **4**, por exemplo, o resultado é **4** (0100). Assim, na elaboração de um programa onde o teclado tem a configuração mostrada, há necessidade de observar os resultados da Tabela 8.1.

Tabela 8.1: tabela-verdade do conjunto teclado + decodificador

Tecla digitada	Saída				Saída Hex	Flag Liberação DA
	D	C	B	A		
1	0	0	0	0	0	0
2	0	0	0	1	1	0
3	0	0	1	0	2	0
4	0	1	0	0	4	0
5	0	1	0	1	5	0
6	0	1	1	0	6	0
7	1	0	0	0	8	0
8	1	0	0	1	9	0
9	1	0	1	0	A	0
0	1	1	0	1	D	0
*	1	1	0	0	C	0
#	1	1	1	0	E	0

A Fig. 8.2 mostra o circuito adotado para o drive do teclado. O decodificador 74LS922 utilizado é de 18 pinos.

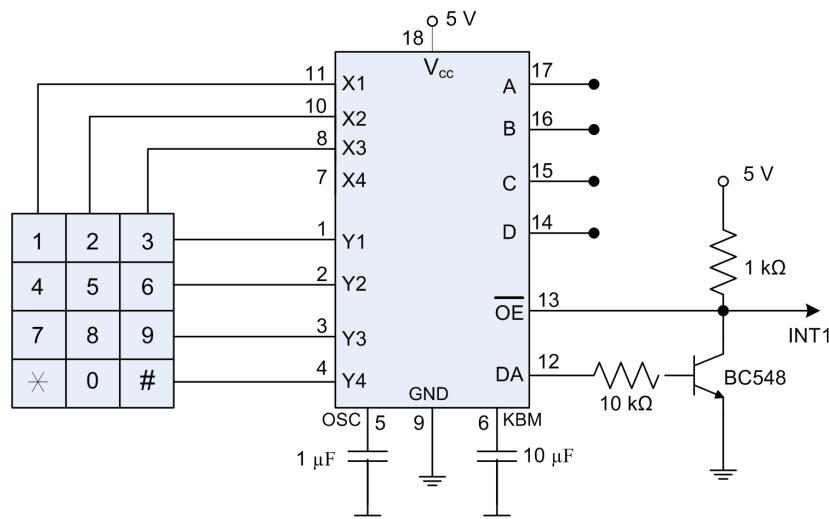


Fig. 8.2: Circuito do decodificador do teclado

São apresentados a seguir três programas-exemplo. No primeiro exemplo (**Teclado 1**) o teclado é lido por varredura. Ele é conectado à porta P2 e os LEDs à porta P1. Assim, cada tecla lida é enviada para os LEDs.

Nos outros dois exemplos o teclado está conectado ao nibble superior da porta P3 e usa a interrupção externa 1 (INT1). No programa **Teclado 2** a tecla digitada é enviada para os LEDs sem qualquer conversão, ou seja, deverá aparecer o número 0 no display ao ser digitado 1; deverá aparecer o número 1 ao ser digitado 2 e o número 4, ao ser digitado 4 (conforme mostrado na Tabela 8.1). No exemplo **Teclado 3** o código da tecla digitada é convertido através da Tabela 8.1, antes de ser enviado para os LEDs.

Deve-se observar que o código do teclado é lido através do nibble superior da porta **P3** (*P3.4, P3.5, P3.6, P3.7*), o que significa que deverá haver uma troca de nibbles (SWAP A) antes de mostrar o dado através do nibble inferior da porta **P1**. Deve-se observar ainda que o dado é mostrado apenas no nibble inferior da porta P1. Assim, antes de enviar o dado para a porta, é feita uma operação AND com “0FH” para eliminar qualquer “lixo” presente no nibble superior.

Teclado 1: O teclado está conectado à porta P2 e é lido por varredura. A tecla digitada é mostrada em P1.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	LINHA_L1 EQU P2.0	
	LINHA_L2 EQU P2.1	
	LINHA_L3 EQU P2.2	
	LINHA_L4 EQU P2.3	
	COLUNA_C1 EQU P2.4	
	COLUNA_C2 EQU P2.5	
	COLUNA_C3 EQU P2.6	
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
VARREDURA:	SETB LINHA_L1	; Preparação para cada varredura
	SETB LINHA_L2	
	SETB LINHA_L3	
	SETB LINHA_L4	
	; VERIFICA AS TECLAS DA LINHA 1	
	CLR LINHA_L1	; Limpa linha L1 do Teclado
	JB COLUNA_C1,TC2	; Verifica se a Tecla 1 foi pressionada.
	MOV P1,#01H	; Tecla 1 foi pressionada
	SJMP VARREDURA	
TC2:	JB COLUNA_C2,TC3	; Verifica se a Tecla 2 foi pressionada
	MOV P1,#02H	; Tecla 2 foi pressionada
	SJMP VARREDURA	
TC3:	JB COLUNA_C3,TC4	; Verifica se Tecla 3 foi pressionada
	MOV P1,#03H	; Tecla 3 foi pressionada
	SJMP VARREDURA	
	; VERIFICA AS TECLAS DA LINHA 2	
TC4:	CLR LINHA_L2	; Limpa linha L2 do Teclado
	JB COLUNA_C1,TC5	; Verifica se a Tecla 4 foi pressionada.
	MOV P1,#04H	; Tecla 4 foi pressionada
	SJMP VARREDURA	
TC5:	JB COLUNA_C2,TC6	; Verifica se a Tecla 5 foi pressionada
	MOV P1,#05H	; Tecla 5 foi pressionada
	SJMP VARREDURA	
TC6:	JB COLUNA_C3,TC7	; Verifica se Tecla 6 foi pressionada
	MOV P1,#06H	; Tecla 6 foi pressionada
	SJMP VARREDURA	
	; VERIFICA AS TECLAS DA LINHA 3	
TC7:	CLR LINHA_L2	; Limpa linha L3 do Teclado
	JB COLUNA_C1,TC8	; Verifica se a Tecla 7 foi pressionada.
	MOV P1,#07H	; Tecla 7 foi pressionada
	SJMP VARREDURA	

TC8:	JB COLUNA_C2,TC9	; Verifica se a Tecla 8 foi pressionada
	MOV P1,#08H	; Tecla 8 foi pressionada
	SJMP VARREDURA	
TC9:	JB COLUNA_C3,TCA	; Verifica se Tecla 9 foi pressionada
	MOV P1,#09H	; Tecla 9 foi pressionada
	SJMP VARREDURA	
	; VERIFICA AS TECLAS DA LINHA 4	
TCA:	CLR LINHA_L2	; Limpa linha L4 do Teclado
	JB COLUNA_C1,TC0	; Verifica se a Tecla * foi pressionada.
	MOV P1,#0AH	; Tecla * foi pressionada
	SJMP VARREDURA	
TC0:	JB COLUNA_C2,TCB	; Verifica se a Tecla 0 foi pressionada
	MOV P1,#00H	; Tecla 0 foi pressionada
	SJMP VARREDURA	
TCB:	JB COLUNA_C3,NADA	; Verifica se Tecla # foi pressionada
	MOV P1,#0BH	; Tecla # foi pressionada
	SJMP VARREDURA	
NADA:	LJMP VARREDURA	
	END	

Teclado 2: A tecla digitada é mostrada em P1 sem conversão, ou seja, o código liberado pelo decodificador para cada tecla digitada é mostrado.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 13H	; Endereço da interrupção externa 1
	LJMP ATENDE_UM	; Desvia para a subrotina de atendimento
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV IE,#84H	; Habilita interrupção externa 1
	MOV TCON,#04H	; Interrupção INT1 por transição
	SJMP \$; Aguarda interrupção em um laço infinito
ATENDE_UM:	MOV A,P3	; Acumulador recebe código da tecla digitada
	SWAP A	; Há inversão entre os nibbles superior e inferior
	ANL A,#0FH	; O “lixo” do nibble superior é removido
	MOV P1,A	; Mostra em P1 código da tecla digitada
	RETI	; Retorna de subrotina de interrupção
	END	

Teclado 3: A tecla digitada é mostrada em P1 após conversão, ou seja, para cada código liberado pelo teclado, é buscado numa tabela de conversão o código a ser mostrado em P1.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 13H	Endereço da interrupção externa 1
	LJMP ATENDE_UM	Desvia para a subrotina de atendimento
	ORG 30H	
INICIO:	MOV SP,#2FH	Apontador de pilha SP = 2FH
	MOV IE,#84H	Habilita interrupção externa 1
	MOV TCON,#04H	Interrupção INT1 por transição

	MOV DPTR,#TABELA	Carrega DPTR com endereço de início da TABELA
	SJMP \$	Aguarda interrupção em um laço infinito
ATENDE_UM:	MOV A,P3	Acumulador recebe código da tecla digitada
	SWAP A	Há inversão entre os nibbles superior e inferior
	ANL A,#0FH	O “lixo” do nibble superior é removido
	MOVC A,@A+DPTR	Carrega A com código convertido com o uso da TABELA
	MOV P1,A	Mostra em P1 código da tecla digitada, convertido pela TABELA
	RETI	Retorna de subrotina de interrupção
TABELA:	DB 01H	; TECLA DIGITADA: 1 → SAÍDA: 0 H (0000)
	DB 02H	; TECLA DIGITADA: 2 → SAÍDA: 1 H (0001)
	DB 03H	; TECLA DIGITADA: 3 → SAÍDA: 2 H (0010)
	DB 0FFH	; NENHUMA TECLA → SAÍDA: 3 H (0111)→ NÃO EXISTENTE
	DB 04H	; TECLA DIGITADA: 4 → SAÍDA: 4 H (1000)
	DB 05H	; TECLA DIGITADA: 5 → SAÍDA: 5 H (0101)
	DB 06H	; TECLA DIGITADA: 6 → SAÍDA: 6 H (0110)
	DB 0FFH	; NENHUMA TECLA: → SAÍDA: 7 H (0111)→ NÃO EXISTENTE
	DB 07H	; TECLA DIGITADA: 7 → SAÍDA: 8 H (1000)
	DB 08H	; TECLA DIGITADA: 8 → SAÍDA: 9 H (1001)
	DB 09H	; TECLA DIGITADA: 9 → SAÍDA: A H (1010)
	DB 0FFH	; NENHUMA TECLA: → SAÍDA: B H (1011)→ NÃO EXISTENTE
	DB 0CH	; TECLA DIGITADA: * → SAÍDA: C H (1100)
	DB 00H	; TECLA DIGITADA: 0 → SAÍDA: D H (1101)
	DB 0EH	; TECLA DIGITADA: # → SAÍDA: E H (1110)
	END	

8.2 Display de 7-Segmentos

O display de 7-segmentos apresentado nesta seção é do tipo catodo comum, cujos terminais são mostrados na Fig. 8.3.

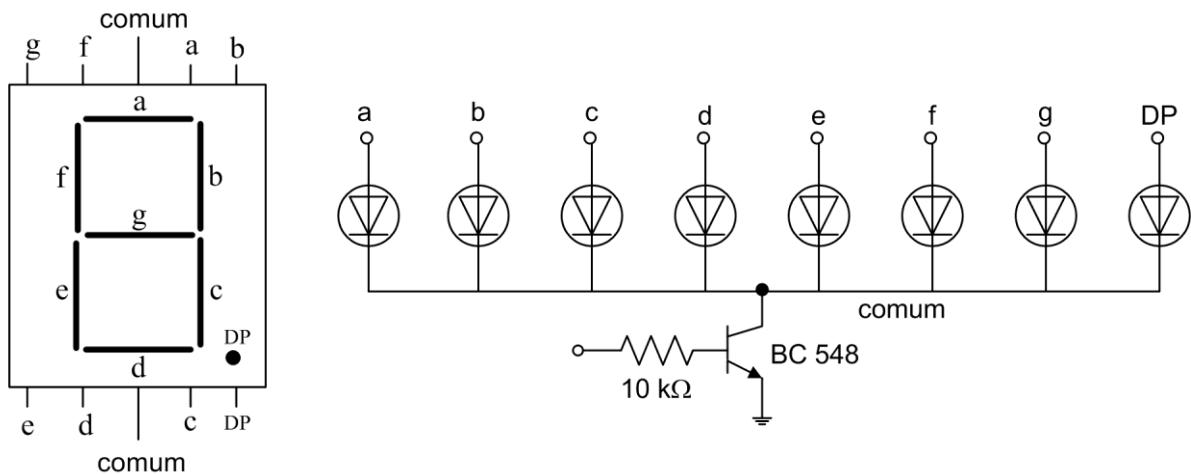


Fig. 8.3: Display de 7-segmentos do tipo catodo comum

O display mostrado pode ser acionado diretamente pelo microcontrolador conectando os terminais *a*, *b*, *c*, *d*, *e*, *f*, *g* e *DP* a uma porta de saída ou através de um driver decodificador.

Para o acionamento direto é necessário criar uma tabela de conversão que associe o número que se deseja com os trechos a serem ativados no display. A Tabela 8.2 mostra um exemplo de tabela que pode ser criada. Assim, desejando-se, por exemplo, mostrar o número 5 no display, o código 6Dh deve ser enviado para a porta escolhida.

Tabela 8.2: código hexadecimal para cada dígito a ser mostrado no display

Número desejado no display	<i>Px.7</i>	<i>Px.6</i>	<i>Px.5</i>	<i>Px.4</i>	<i>Px.3</i>	<i>Px.2</i>	<i>Px.1</i>	<i>Px.0</i>	Hexadecimal resultante
	<i>DP</i>	<i>g</i>	<i>f</i>	<i>e</i>	<i>d</i>	<i>c</i>	<i>b</i>	<i>a</i>	
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	0	1	1	1	67
0	0	0	1	1	1	1	1	1	3F

A outra opção de acionamento utiliza o driver decodificador **CD4511**, cuja entrada é BCD (4 bits) e cuja saída alimenta cada um dos terminais mostrados (*a*, *b*, *c*, *d*, *e*, *f*, *g*). Quando se deseja também mostrar o ponto decimal, um bit extra deve alimentar o terminal *DP* (ponto decimal).

A Fig. 8.4 ilustra o uso de 4 displays, conectados à porta P1 e usando apenas um decodificador. A saída do decodificador alimenta simultaneamente os 4 displays, mas apenas um é selecionado de cada vez, com o uso de transistores conectados como mostrado na Fig. 8.3. Assim, além dos 4 bits de dados enviados para os displays, é necessário enviar mais 4 bits de comando para a seleção de cada display. A Fig. 8.4 mostra os pinos escolhidos para o envio de dados e de comando.

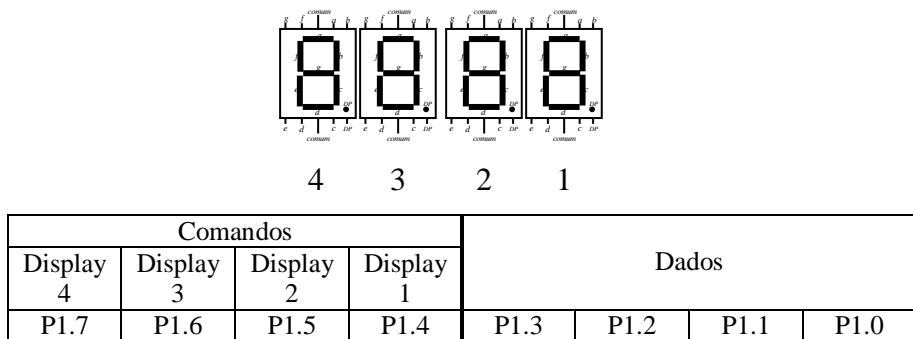


Fig. 8.4: Conjunto de displays de 7-segmentos e suas conexões

São dados a seguir dois programas para ilustrar o uso dos displays de 7-segmentos, conectados à Porta P1. No primeiro exemplo tem-se apenas um display conectado à porta P1, como mostrado através da Tabela 8.2. Uma contagem de 0 a 9 é mostrada. No programa é necessária a decodificação de cada dígito a ser mostrado. No segundo exemplo é utilizado o decodificador CD4511 para o acionamento de 4 displays de 7-segmentos simultaneamente.

Display 1: Programa que mostra uma contagem de 0 a 9 em um único display de 7-segmentos conectado diretamente à porta P1, seguindo as conexões da Tabela 8.2.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV DPTR,#DIGITO	; Carrega DPTR com endereço de início da tabela DIGITO
V2:	MOV R7,#00	; Contador para leitura do DIGITO da tabela
V3:	MOV A,R7	; Acumulador recebe valor atualizado do contador
	CJNE A,#0AH,V1	; Faz a leitura de 10 valores da tabela DIGITO e retorna pro início
	SJMP V2	

V1:	MOVC A,@A+DPTR	; Carrega A com código que equivale ao número desejado
	MOV P1,A	; Transfere para P1 o código de cada número (de 0 a 9)
	INC R7	; Incrementa contador usado na leitura da tabela DIGITO
	LCALL ATRASO	; Chama subrotina de atraso de tempo
	SJMP V3	; Retorna leitura do próximo dígito
ATRASO:	MOV R4,#20	; Carrega registrador R4 com valor decimal 20
V5:	MOV R5,#250	; Carrega registrador R5 com valor decimal 250
V4:	MOV R6,#250	; Carrega registrador R6 com valor decimal 250
	DJNZ R6,\$; Aguarda registrador R6 zerar
	DJNZ R5,V4	; Decrementa R5. Enquanto não for zero, volta para recarregar R6
	DJNZ R4,V5	; Decrementa R4. Enquanto não for zero, volta para recarregar R5
	RET	; Retorna de subrotina de atraso de tempo
DIGITO:	DB 3FH	; Código para mostrar o número 0 no display
	DB 06H	; Código para mostrar o número 1 no display
	DB 5BH	; Código para mostrar o número 2 no display
	DB 4FH	; Código para mostrar o número 3 no display
	DB 66H	; Código para mostrar o número 4 no display
	DB 6DH	; Código para mostrar o número 5 no display
	DB 7DH	; Código para mostrar o número 6 no display
	DB 07H	; Código para mostrar o número 7 no display
	DB 7FH	; Código para mostrar o número 8 no display
	DB 67H	; Código para mostrar o número 9 no display
	END	

Display 2: É mostrada em um conjunto de 4 displays de 7-segmentos uma contagem decimal crescente de 0000 a 9999. O byte mais significativo da contagem é guardado em R1 e o byte menos significativo é guardado em R0. Cada dado é sempre enviado para o display através do nibble inferior da porta P1. O nibble superior da porta P1 é usado para a escolha do display para mostrar “milhar”, “centena”, “dezena” ou “unidade”.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	Apontador de pilha SP = 2FH
	MOV R0,#00	Byte inferior da contagem
	MOV R1,#00	Byte superior da contagem
	MOV R7,#10	Contador
V1:	LCALL CONTAGEM	Chama subrotina que faz a contagem decimal
	LCALL DISPLAY	Chama subrotina que mostra a contagem nos displays
	SJMP V1	
CONTAGEM:	MOV A,R0	Carrega acumulador com valor atual do byte inferior
	ADD A,#01H	Incrementa acumulador em uma unidade
	DA A	Faz o ajuste decimal do byte inferior
	MOV R0,A	Atualiza o valor de R0
	JNC V2	Desvia para V2 se não houver Carry, ou seja, se R0 <= 99
	MOV A,R1	Carrega acumulador com valor atual do byte superior
	ADD A,#01H	Incrementa acumulador em uma unidade
	DA A	Faz o ajuste decimal do byte superior
	MOV R1,A	Atualiza o valor de R1
V2:	NOP	
	RET	
DISPLAY:	PUSH ACC	
	MOV A,#0FOH	Este bloco separa o nibble superior de R1
	ANL A,R1	Instrução que separa o nibble superior de R1
	SWAP A	Inverte nibble superior com inferior
	ORL A,#80H	Junta o quarto dígito com o comando do quarto display
	MOV P1,A	Envia quarto dígito + comando para a porta P1
	LCALL ATRASO	Chama subrotina de atraso de tempo

	MOV A,#0FH	Este bloco separa o nibble inferior de R1
	ANL A,R1	Instrução que separa o nibble inferior de R1
	ORL A,#40H	Junta o terceiro dígito com o comando do terceiro display
	MOV P1,A	Envia terceiro dígito + comando para a porta P1
	LCALL ATRASO	
	MOV A,#0F0H	Este bloco separa o nibble superior de R0
	ANL A,R0	Instrução que separa o nibble superior de R0
	SWAP A	Inverte nibble superior com inferior
	ORL A,#20H	Junta o segundo dígito com o comando do segundo display
	MOV P1,A	Envia segundo dígito + comando para a porta P1
	LCALL ATRASO	Chama subrotina de atraso de tempo
	MOV A,#0FH	Este bloco separa o nibble inferior de R0
	ANL A,R0	Instrução que separa o nibble inferior de R0
	ORL A,#10H	Junta o primeiro dígito com o comando do primeiro display
	MOV P1,A	Envia primeiro dígito + comando para a porta P1
	LCALL ATRASO	
	POP ACC	
	DJNZ R7,DISPLAY	Contador necessário para uma contagem mais lenta
	MOV R7,#10	Recarrega valor do contador
	RET	
ATRASO:	MOV R4,#10	Carrega registrador R4 com valor decimal 20
V3:	MOV R5,#200	Carrega registrador R5 com valor decimal 250
	DJNZ R5,\$	Aguarda registrador R6 zerar
	DJNZ R4,V3	Decrementa R5. Enquanto não for zero, volta para recarregar R6
	RET	Retorna de subrotina de atraso de tempo
	END	

8.3 Display LCD

Esta seção usa um display LCD de 2 linhas x 16 colunas, cujo esquemático é mostrado na Fig. 8.5. Essa informação é enviada na forma de comando para o LCD, numa rotina de inicialização necessária a cada vez que o LCD vai ser usado. Outras informações necessárias são: se o cursor vai ficar piscando, se a mensagem vai rolar para a esquerda ou para a direita, ou não vai rolar, se serão usados 4 ou 8 bits para os dados etc. A Tabela 8.3 mostra os sinais de controle para escrita e leitura do LCD e a Tabela 8.4 mostra as instruções mais comuns utilizadas no uso do LCD.

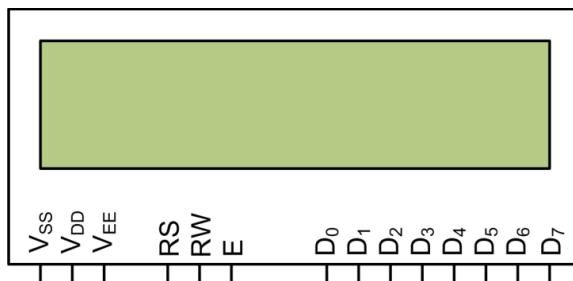


Fig. 8.5: Display LCD de 2 linhas por 16 colunas

Tabela 8.3: Habilitação do display LCD

E	RW	RS	Operação
0	×	×	Display desabilitado
1	0	0	Escrita de Instrução no LCD
1	0	1	Escrita de Dados no LCD
1	1	0	Leitura de Instrução do LCD
1	1	1	Leitura de Dados do LCD

Tabela 8.4: instruções mais comuns utilizadas para o display LCD

DESCRÍÇÃO	MODO	RS	R/W	Código (Hexa)
Display	Liga (sem cursor)	0	0	0C
	Desliga	0	0	0A/08
Limpa Display com Home cursor		0	0	01
Controle do Cursor	Liga	0	0	0E
	Desliga	0	0	0C
	Desloca para Esquerda	0	0	10
	Desloca para Direita	0	0	14
	Cursor Home	0	0	02
	Cursor Piscante	0	0	0D
	Cursor com Alternância	0	0	0F
Sentido de deslocamento do cursor ao entrar com caractere	Para a esquerda	0	0	04
	Para a direita	0	0	06
Deslocamento da mensagem ao entrar com caractere	Para a esquerda	0	0	07
	Para a direita	0	0	05
Deslocamento da mensagem sem entrada de caractere	Para a esquerda	0	0	18
	Para a direita	0	0	1C
Endereço da primeira posição	Primeira linha	0	0	80
	Segunda linha	0	0	C0

A Tabela 8.5 mostra o endereço em decimal de cada posição do LCD de 16 colunas x 2 linhas.

Tabela 8.5: Endereços em decimal do display LCD

Colunas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Linha 1	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
Linha 2	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207

O programa-exemplo a seguir mostra uma mensagem na primeira linha e uma contagem decimal crescente de 0000 a 9999 na segunda linha do LCD. São utilizados 8 bits para os dados e 3 bits de comando. No exemplo a porta **P0** é usada para os dados e os pinos **P2.0**, **P2.1** e **P2.2** são usados para os comandos.

LCD 1: É mostrada no display LCD uma contagem decimal crescente de 0000 a 9999. O byte mais significativo da contagem é guardado em R1 e o byte menos significativo é guardado em R0. Cada um dos quatro dígitos da contagem é convertido para caractere ASCII e depois enviado para o LCD através da porta P0.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	RS EQU P2.0	; RS = 0 → INSTRUÇÃO. RS = 1 → DADO
	RW EQU P2.1	; RW = 0 → ESCRITA. RW = 1 → LEITURA
	EN EQU P2.2	; PINO DE HABILITAÇÃO DO LCD
	CONTADOR EQU 00H	
	DADOS EQU P0	; Porta P0 é o canal de dados
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV R0,#00	; Byte inferior da contagem
	MOV R1,#00	; Byte superior da contagem
	MOV R7,#0FFH	; Contador
	LCALL INICIA	; Chama subrotina de inicialização do LCD
	LCALL LIMPA	; Chama subrotina que limpa LCD
	LCALL LINHA1	; Chama subrotina que mostra mensagem na linha 1
	LCALL LINHA2	; Chama subrotina que mostra mensagem na linha 2

REPETE:	LCALL CONTAGEM	; Chama subrotina de contagem decimal contagem de 0000 a 9999
	LCALL CONVERTE4	; Converte Dígito 4 para ASCII
	LCALL MOSTRA4	; Mostra Dígito 4 no display LCD
	LCALL CONVERTE3	; Converte Dígito 3 para ASCII
	LCALL TEXTO_WR	; Mostra Dígito 3 no display LCD
	LCALL CONVERTE2	; Converte Dígito 2 para ASCII
	LCALL TEXTO_WR	; Mostra Dígito 2 no display LCD
	LCALL CONVERTE1	; Converte Dígito 1 para ASCII
	LCALL TEXTO_WR	; Mostra Dígito 1 no display LCD
	LCALL ATRASO	
	LCALL ATRASO	
	LCALL ATRASO	
	SJMP REPETE	; Volta para o início
	; SUBROTINA DE INICIALIZAÇÃO DO DISPLAY LCD	
INICIA:	MOV A,#38H	; Instrução que indica display de 16 colunas e 2 linhas
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV A,#38H	; Instrução que indica display de 16 colunas e 2 linhas
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV A,#0EH	; Instrução para ligar o cursor
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV A,#06H	; Instrução para deslocar cursor para a direita
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	RET	; Retorna da subrotina de inicialização do LCD
	; SUBROTINA QUE LIMPA O DISPLAY	
LIMPA:	MOV A,#01H	; Instrução para limpar LCD
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	RET	
LINHA1:	INC R7	; Incrementa contador
	MOV A,R7	; Carrega acumulador com o conteúdo do contador
	MOV DPTR,#MSG1	; DPTR recebe o endereço da mensagem "MSG1"
	MOVC A,@A+DPTR	; Acumulador recebe o código do caractere do endereço A+DPTR
	CJNE A,#0FFH,V1	; Se A = OFFH → fim da mensagem. Caso contrário, pula para V1
	RET	
V1:	LCALL TEXTO_WR	
	LCALL ATRASO_LCD	
	SJMP LINHA1	
LINHA2:	MOV A,#192	; Instrução para definir endereço do LCD: 192 = C0H
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV R7,#0FFH	; Contador recebe valor FFH
V3:	INC R7	; Incrementa contador
	MOV A,R7	; Carrega acumulador com o conteúdo do contador
	MOV DPTR,#MSG2	; DPTR recebe o endereço da mensagem "MSG2"
	MOVC A,@A+DPTR	; Acumulador recebe o código do caractere do endereço A+DPTR
	CJNE A,#0FFH,V2	; Se A = OFFH → fim da mensagem. Caso contrário, pula para V2
	RET	
V2:	LCALL TEXTO_WR	; Chama subrotina para escrever dados no LCD
	SJMP V3	
INSTR_WR:	SETB EN	; Habilita LCD
	CLR RW	; Operação de escrita no LCD
	CLR RS	; Operação com instrução
	MOV DADOS,A	; Transfere a instrução para o LCD
	CLR EN	; Desabilita LCD
	LCALL ATRASO_LCD	; Chama subrotina de atraso do LCD

	RET	
TEXTO_WR:	SETB EN	; Habilita LCD
	CLR RW	; Operação de escrita no LCD
	SETB RS	; Operação com dados
	MOV DADOS,A	; Transfere os dados para o LCD
	CLR EN	; Desabilita LCD
	LCALL ATRASO_LCD	; Chama subrotina de atraso do LCD
	RET	
CONTAGEM:	MOV A,R0	Carrega acumulador com valor atual do byte inferior
	ADD A,#01H	Incrementa acumulador em uma unidade
	DA A	Faz o ajuste decimal do byte inferior
	MOV R0,A	Atualiza o valor de R0
	JNC V4	Desvia para V2 se não houver Carry, ou seja, se R0 <= 99
	MOV A,R1	Carrega acumulador com valor atual do byte superior
	ADD A,#01H	Incrementa acumulador em uma unidade
	DA A	Faz o ajuste decimal do byte superior
	MOV R1,A	Atualiza o valor de R1
V4:	NOP	
	RET	
	; SUBROTINAS PARA CONVERTER VALORES EM ASCII	
CONVERTE4:	MOV A,#0FOH	; Prepara a separação do nibble superior de R1
	ANL A,R1	; Separa nibble superior de R1
	SWAP A	; Inverte nibble superior com inferior
	ORL A,#30H	; Converte nibble superior de R1 em ASCII
	RET	
CONVERTE3:	MOV A,#0FH	; Prepara a separação do nibble inferior de R1
	ANL A,R1	; Separa nibble inferior de R1
	ORL A,#30H	; Converte nibble inferior de R1 em ASCII
	RET	
CONVERTE2:	MOV A,#0FOH	; Prepara a separação do nibble superior de R0
	ANL A,R0	; Separa nibble superior de R0
	SWAP A	; Inverte nibble superior com inferior
	ORL A,#30H	; Converte nibble superior de R1 em ASCII
	RET	
CONVERTE1:	MOV A,#0FH	; Prepara a separação do nibble inferior de R0
	ANL A,R0	; Separa nibble inferior de R0
	ORL A,#30H	; Converte nibble inferior de R0 em ASCII
	RET	
	; SUBROTINA PARA MOSTRAR O CONTEÚDO DO ACUMULADOR NO LCD	
MOSTRA4:	PUSH ACC	; Guarda dígito 4 na pilha, antes de definir endereço no LCD
	MOV A,#202	; Endereço do dígito 4
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	POP ACC	; Recupere dígito 4 da pilha
	LCALL TEXTO_WR	; Chama subrotina para escrever dados no LCD – dígito 4
	RET	
ATRASO_LCD:	MOV R4,#10	; Carrega registrador R4 com valor decimal 10
V6:	MOV R5,#80	; Carrega registrador R5 com valor decimal 80
	DJNZ R5,\$; Aguarda registrador R5 zerar
	DJNZ R4,V6	; Decrementa R4. Enquanto não for zero, volta para recarregar R5
	RET	; Retorna de subrotina de atraso de tempo
ATRASO:	MOV R4,#200	; Carrega registrador R4 com valor decimal 200
V5:	MOV R5,#250	; Carrega registrador R5 com valor decimal 250
	DJNZ R5,\$; Aguarda registrador R5 zerar
	DJNZ R4,V5	; Decrementa R4. Enquanto não for zero, volta para recarregar R5

	RET	; Retorna de subrotina de atraso de tempo
MSG1:	DB 'MICROCONTROLADOR', 0FFH	
MSG2:	DB 'CONTAGEM: ',0FFH	
FIM:	NOP	
	END	

LCD 2: A mesma contagem decimal crescente de 0000 a 9999 do exemplo anterior é mostrada. No entanto, utiliza-se o mapeamento de memória do capítulo 7. Os endereços para leitura e escrita de instrução e dados são dados na Tabela 7.4.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ESCREVE_INST EQU 4000H;	Endereço para escrever instrução: RW = 0; RS = 0
	ESCREVE_DADO EQU 4001H;	Endereço para escrever dados: RW = 0; RS = 1
	LE_INSTRUCAO EQU 4002H;	Endereço para leitura de instrução: RW = 1; RS = 0
	LE_DADOS EQU 4003H;	Endereço para leitura de dados: RW = 1; RS = 1
	CONTADOR EQU 00H	
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	MOV R0,#00	; Byte inferior da contagem
	MOV R1,#00	; Byte superior da contagem
	MOV R7,#0FFH	; Contador
	LCALL INICIA	; Chama subrotina de inicialização do LCD
	LCALL LIMPA	; Chama subrotina que limpa LCD
	LCALL LINHA1	; Chama subrotina que mostra mensagem na linha 1
	LCALL LINHA2	; Chama subrotina que mostra mensagem na linha 2
REPETE:	LCALL CONTAGEM	; Chama subrotina de contagem decimal contagem de 0000 a 9999
	LCALL CONVERTE4	; Converte Dígito 4 para ASCII
	LCALL MOSTRA4	; Mostra Dígito 4 no display LCD
	LCALL CONVERTE3	; Converte Dígito 3 para ASCII
	LCALL TEXTO_WR	; Mostra Dígito 3 no display LCD
	LCALL CONVERTE2	; Converte Dígito 2 para ASCII
	LCALL TEXTO_WR	; Mostra Dígito 2 no display LCD
	LCALL CONVERTE1	; Converte Dígito 1 para ASCII
	LCALL TEXTO_WR	; Mostra Dígito 1 no display LCD
	LCALL ATRASO	
	LCALL ATRASO	
	LCALL ATRASO	
	SJMP REPETE	; Volta para o início
		; SUBROTINA DE INICIALIZAÇÃO DO DISPLAY LCD
INICIA:	MOV A,#38H	; Instrução que indica display de 16 colunas e 2 linhas
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV A,#38H	; Instrução que indica display de 16 colunas e 2 linhas
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV A,#0EH	; Instrução para ligar o cursor
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV A,#06H	; Instrução para deslocar cursor para a direita
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	RET	; Retorna da subrotina de inicialização do LCD
		; SUBROTINA QUE LIMPA O DISPLAY
LIMPA:	MOV A,#01H	; Instrução para limpar LCD
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD

	RET	
LINHA1:	INC R7	; Incrementa contador
	MOV A,R7	; Carrega acumulador com o conteúdo do contador
	MOV DPTR,#MSG1	; DPTR recebe o endereço da mensagem "MSG1"
	MOVC A,@A+DPTR	; Acumulador recebe o código do caractere do endereço A+DPTR
	CJNE A,#0FFH,V1	; Se A = 0FFH → fim da mensagem. Caso contrário, pula para V1
	RET	
V1:	LCALL TEXTO_WR	
	LCALL ATRASO_LCD	
	SJMP LINHA1	
LINHA2:	MOV A,#192	; Instrução para definir endereço do LCD: 192 = C0H
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	MOV R7,#0FFH	; Contador recebe valor FFH
V3:	INC R7	; Incrementa contador
	MOV A,R7	; Carrega acumulador com o conteúdo do contador
	MOV DPTR,#MSG2	; DPTR recebe o endereço da mensagem "MSG2"
	MOVC A,@A+DPTR	; Acumulador recebe o código do caractere do endereço A+DPTR
	CJNE A,#0FFH,V2	; Se A = 0FFH → fim da mensagem. Caso contrário, pula para V2
	RET	
V2:	LCALL TEXTO_WR	; Chama subrotina para escrever dados no LCD
	SJMP V3	
INSTR_WR:	MOV DPTR,#ESCREVE_INST	; DPTR assume o endereço de escrita de instrução: 4000H
	MOVX @DPTR,A	; Transfere a instrução para o LCD
	LCALL ATRASO_LCD	; Chama subrotina de atraso do LCD
	RET	
TEXTO_WR:	MOV DPTR,#ESCREVE_DADO	; DPTR assume o endereço de escrita de dados: 4001H
	MOVX @DPTR,A	; Transfere os dados para o LCD
	LCALL ATRASO_LCD	; Chama subrotina de atraso do LCD
	RET	
CONTAGEM:	MOV A,R0	Carrega acumulador com valor atual do byte inferior
	ADD A,#01H	Incrementa acumulador em uma unidade
	DA A	Faz o ajuste decimal do byte inferior
	MOV R0,A	Atualiza o valor de R0
	JNC V4	Desvia para V2 se não houver Carry, ou seja, se R0 <= 99
	MOV A,R1	Carrega acumulador com valor atual do byte superior
	ADD A,#01H	Incrementa acumulador em uma unidade
	DA A	Faz o ajuste decimal do byte superior
	MOV R1,A	Atualiza o valor de R1
V4:	NOP	
	RET	
	; SUBROTINAS PARA CONVERTER VALORES EM ASCII	
CONVERTE4:	MOV A,#0FOH	; Prepara a separação do nibble superior de R1
	ANL A,R1	; Separa nibble superior de R1
	SWAP A	; Inverte nibble superior com inferior
	ORL A,#30H	; Converte nibble superior de R1 em ASCII
	RET	
CONVERTE3:	MOV A,#0FH	; Prepara a separação do nibble inferior de R1
	ANL A,R1	; Separa nibble inferior de R1
	ORL A,#30H	; Converte nibble inferior de R1 em ASCII
	RET	
CONVERTE2:	MOV A,#0FOH	; Prepara a separação do nibble superior de R0
	ANL A,R0	; Separa nibble superior de R0
	SWAP A	; Inverte nibble superior com inferior

	ORL A,#30H	; Converte nibble superior de R1 em ASCII
	RET	
CONVERTE1:	MOV A,#0FH	; Prepara a separação do nibble inferior de R0
	ANL A,R0	; Separa nibble inferior de R0
	ORL A,#30H	; Converte nibble inferior de R0 em ASCII
	RET	
		; SUBROTINA PARA MOSTRAR O CONTEÚDO DO ACUMULADOR NO LCD
MOSTRA4:	PUSH ACC	; Guarda dígito 4 na pilha, antes de definir endereço no LCD
	MOV A,#202	; Endereço do dígito 4
	LCALL INSTR_WR	; Chama subrotina para escrever instrução no LCD
	POP ACC	; Recupere dígito 4 da pilha
	LCALL TEXTO_WR	; Chama subrotina para escrever dados no LCD – dígito 4
	RET	
ATRASO_LCD:	MOV R4,#10	; Carrega registrador R4 com valor decimal 10
V6:	MOV R5,#80	; Carrega registrador R5 com valor decimal 80
	DJNZ R5,\$; Aguarda registrador R5 zerar
	DJNZ R4,V6	; Decrementa R4. Enquanto não for zero, volta para recarregar R5
	RET	; Retorna de subrotina de atraso de tempo
ATRASO:	MOV R4,#200	; Carrega registrador R4 com valor decimal 200
V5:	MOV R5,#250	; Carrega registrador R5 com valor decimal 250
	DJNZ R5,\$; Aguarda registrador R5 zerar
	DJNZ R4,V5	; Decrementa R4. Enquanto não for zero, volta para recarregar R5
	RET	; Retorna de subrotina de atraso de tempo
MSG1:	DB 'MICROCONTROLADOR', 0FFH	
MSG2:	DB 'CONTAGEM: ',0FFH	
FIM:	NOP	
	END	

8.4 Sensores de Presença

São apresentados nesta seção dois tipos de fotosensores: um com nível lógico normalmente alto e outro com nível lógico normalmente baixo. Os dois modelos são mostrados na Fig. 8.6. O primeiro tipo é mostrado com um circuito auxiliar modulador, Fig. 8.7, que diminui a influência da luz ambiente sobre o fotosensor. Esse circuito emite uma luz de cerca de 1 kHz, que ao ser refletida satura o fototransistor. É utilizado o decodificador de frequência NE567. O segundo modelo não usa circuito modulador.

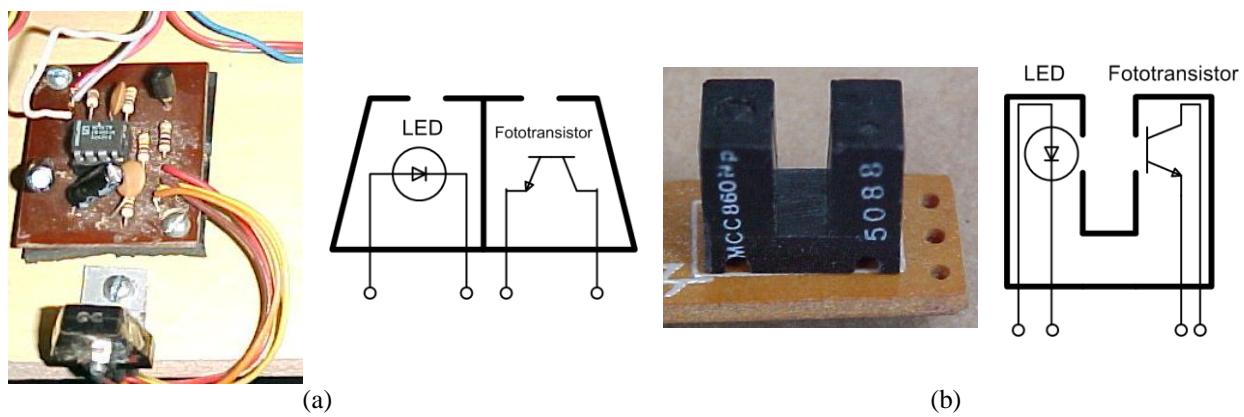


Fig. 8.6: Sensor de presença com circuito auxiliar modulador e sem circuito modulador

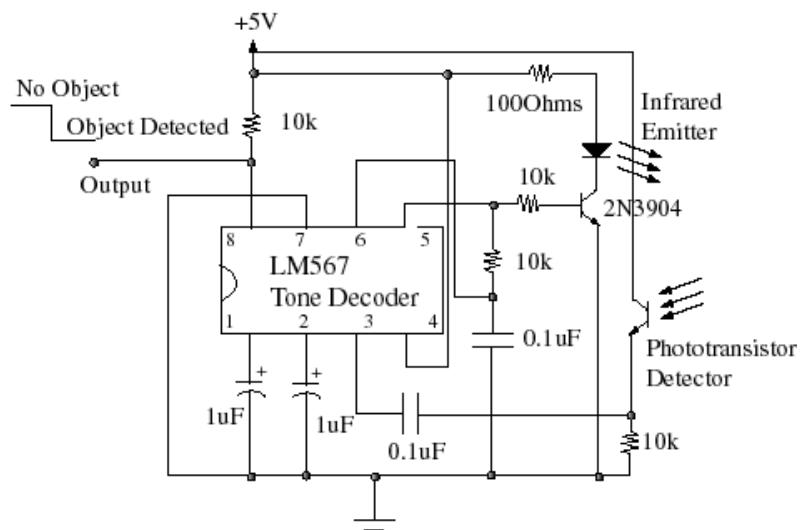


Fig. 8.7: Circuito auxiliar modulador

A diferença básica entre os sensores usados, além do circuito de modulação, está no encapsulamento. Em um deles (Fig. 8.6a) o encapsulamento faz com que o fototransistor fique normalmente cortado (sem presença de luz); ele entra em saturação quando a luz do **LED** é refletida em um obstáculo. Assim, o sinal de saída V_o (Fig. 8.9) passa de nível lógico alto para baixo, na presença de um obstáculo.

No outro tipo de encapsulamento (Fig. 8.6b) a luz do **LED** incide diretamente sobre o fototransistor, fazendo com que ele fique normalmente saturado, ou seja, o sinal de saída V_o (Fig. 8.9) fica inicialmente em nível lógico baixo; na presença de um obstáculo entre os dois componentes o fototransistor é levado ao corte e o sinal de saída vai para o nível lógico alto.

8.5 Medição de Velocidade

O processo de medição de velocidade mostrado nesta seção é digital. Um sensor do tipo apresentado na Fig. 8.6(b) é associado a uma roda com **60 furos** (Fig. 8.8) e usado para medir a velocidade.



Fig. 8.8: conjunto roda de 60 furos e fotosensor

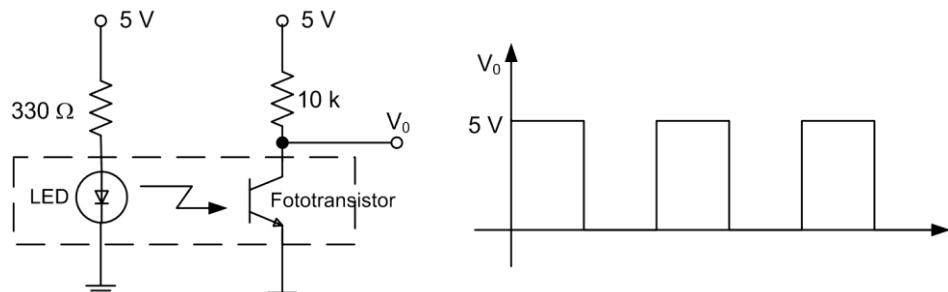


Fig. 8.9: Sinal de saída do sensor de velocidade

A Fig. 8.9 mostra o esquema adotado. O sinal resultante do fotosensor, com a rotação da roda de 60 furos, é uma onda quadrada (Fig. 8.9). Esse sinal pode ser conectado ao pino de uma das interrupções (0 ou 1), definida para ativar por transição (na passagem de nível lógico 1 para 0). Assim, a cada interrupção o registrador com o número atualizado de pulsos é incrementado em 1.

A medição de velocidade é feita estabelecendo-se um tempo de amostragem, ou seja, um tempo fixo em que o registro de pulsos é lido. Mostra-se a seguir que o fato de ter 60 furos na roda faz com que o número de pulsos registrados por segundo (frequência) seja correspondente à velocidade em rotações por minuto (rpm).

$$1 \text{ rotação/segundo} \rightarrow 60 \text{ furos/segundo}$$

$$\rightarrow 60 \text{ rpm} \equiv 60 \text{ furos/s} \rightarrow X \text{ rpm} \equiv X \text{ furos/s}$$

$$1 \text{ rotação/segundo} \rightarrow 60 \text{ rotações/minuto (rpm)}$$

$$\omega \text{ (rpm)} \equiv f \text{ (Hz)}$$

Um tempo de amostragem menor que 1 segundo pode ser adotado, e é aconselhável em muitas aplicações. Sendo assim, deve-se fazer a devida transformação de número de furos lidos no tempo de amostragem para rotações por minuto.

8.6 Motor de Corrente Contínua

O motor de corrente contínua consiste de um enrolamento de campo estacionário e um enrolamento de armadura rotativo. O enrolamento de campo pode ser acionado por corrente contínua, ou ainda consistir de um estator de ímã permanente, não sendo necessária a alimentação. A armadura é acionada com corrente contínua através de escovas e um anel comutador. A Fig. 8.10 mostra o circuito básico de um motor CC, onde os enrolamentos de campo e de armadura são alimentados de forma independente. As expressões básicas também são dadas.

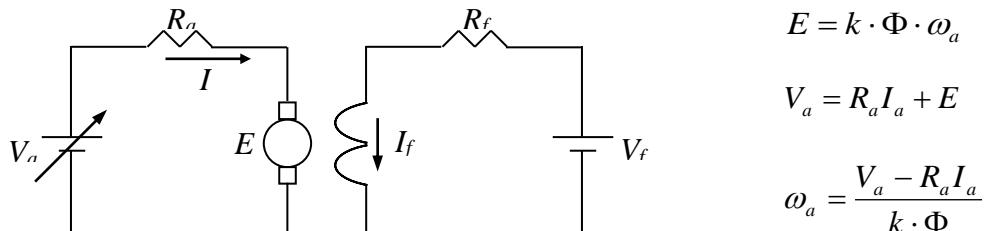


Fig. 8.10: Motor de Corrente Contínua

Uma forma de variar a velocidade do motor CC é variando a tensão de armadura. Uma forma de variar a tensão de armadura é usar modulação PWM, que consiste na definição de um período de acionamento fixo e, dentro desse período, estabelecer um período ligado e outro desligado. A Fig. 8.11 ilustra esse processo. O motor usado é de 12 V e, portanto, a tensão de alimentação deve variar de zero a 12 V para obter-se variação de velocidade de zero até o valor máximo. Isso é feito chaveando-se um transistor a uma frequência alta, por exemplo, 5 kHz, que corresponde a um período de 0,2 ms ou 200 µs.

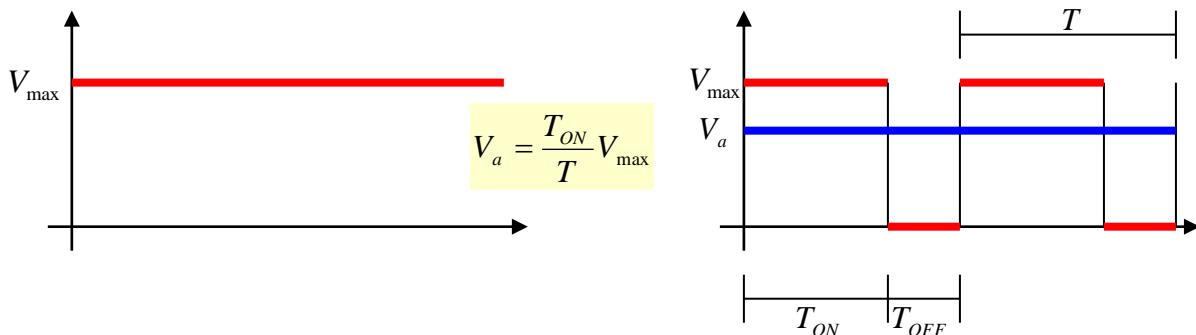


Fig. 8.11: Geração do sinal PWM para controle do motor de contrente contínua

Pode-se preferir definir o período como 255 μ s, por exemplo, o que corresponde a uma frequência de chaveamento de 3,92 kHz. Esse valor, 255, é adequado porque corresponde a uma contagem completa do microcontrolador com o temporizador operando no modo 2 (recarga automática). Se o cristal oscilador for de 12 MHz, tem-se um período de 255 μ s. A variação de velocidade pode então ser obtida variando-se o período ligado (T_{ON}) de 0 a 255 μ s, ao mesmo tempo em que o período desligado (T_{OFF}) deve variar de 255 μ s a 0, para manter constante o período total (T).

O microcontrolador 8051 possui dois temporizadores/contadores, que podem operar em 4 modos diferentes: modo de 13 bits (modo 0), modo de 16 bits (modo 1), modo de 8 bits com recarga automática (modo 2) e 2 modos independentes de 8 bits (modo 3). O modo de recarga automática (modo 2) pode ser usado para gerar o sinal PWM para o controle do motor CC. Nesse modo a contagem é feita através de TL (a primeira contagem começa no valor inicial de TL) e o valor de TH é usado para definir o início da próxima contagem. Se o temporizador começa sempre no valor dado em TH e vai até FFh (255), e sendo $T_{ON} = T - T_{OFF}$, e sendo ainda escolhido $T = 255$, o procedimento usado no programa é:

1. Para o período ligado faz-se $TL = T_{OFF}$; dessa forma, o temporizador conta de T_{OFF} até 255, o que corresponde ao período ligado;
2. Para o período desligado faz-se $TL = T_{ON}$; dessa forma, o temporizador conta de T_{ON} até 255, o que corresponde ao período desligado.
3. No início do programa desliga-se o motor (CLR P2.0 e CLR P2.1) e faz-se $TL = T_{ON} = 09H$, o que faz com que o temporizador, na primeira contagem já conte o período desligado, que começa em T_{ON} e vai até 255.
4. Após fazer $TL = T_{ON}$, encontra-se o complementar de T_{ON} (CPL A), ou seja, T_{OFF} , e carrega-se em TH. Dessa forma, a próxima contagem começará em T_{OFF} , o que significa que o temporizador contará o período ligado.
5. Cada vez que a subrotina de controle é executada define-se o próximo valor de recarga, TH.
6. Para aumentar a velocidade aumenta-se o período ligado T_{ON} . Para diminuir aumenta-se T_{OFF} .

A interface entre o microcontrolador e o motor de corrente contínua (driver) pode permitir o acionamento apenas em um sentido de rotação (Fig. 8.12) ou em ambos os sentidos (Fig. 8.13). Na Fig. 8.12 um pulso alto na base do transistor BC548 leva o transistor BD139 à saturação, o que aciona o motor. Um pulso baixo leva esse transistor ao corte, quando então a corrente do motor decresce circulando pelo diodo 1N4001.

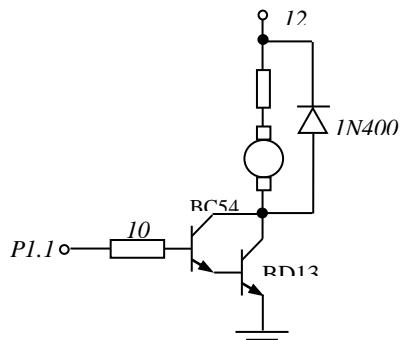


Fig. 8.12: Driver acionamento do motor CC num único sentido

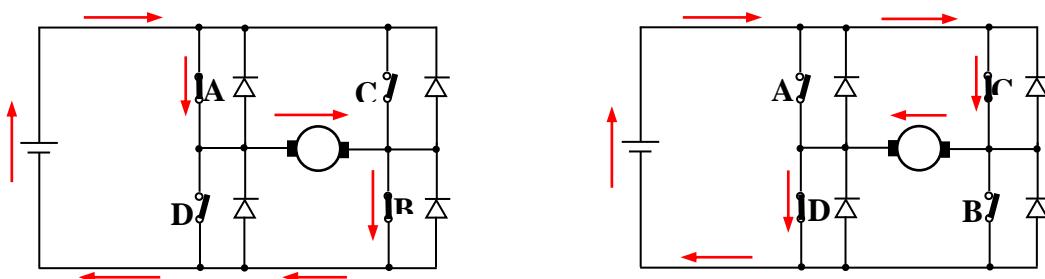


Fig. 8.13: Driver (ponte H) para acionamento do motor CC em ambos os sentidos

A Fig. 8.13 mostra uma configuração denominada de Ponte H, que permite o acionamento em ambos os sentidos. As chaves A, B, C e D são normalmente transistores do tipo MOSFET ou IGBT. Para o acionamento em um dos sentidos as chaves A e B são acionadas; para o acionamento no sentido contrário as chaves C e D são acionadas. A lógica de acionamento dessas chaves não deve permitir o acionamento simultâneo das chaves A e D e das chaves C e B, o que resultaria num curto-circuito da fonte de alimentação. O driver de acionamento em ponte H usado no laboratório (**L298N** – Diagrama na Fig. 8.14) permite o acionamento de um motor com corrente de até 1,5 A através de dois pinos de comando e segue a lógica da Tabela 8.6.

É importante observar que os diodos são fundamentais para o retorno da corrente, quando qualquer uma das chaves é desligada. No momento de desligamento das chaves, há energia armazenada nas indutâncias do motor; sem os diodos as chaves poderiam ser danificadas por sobretensão.

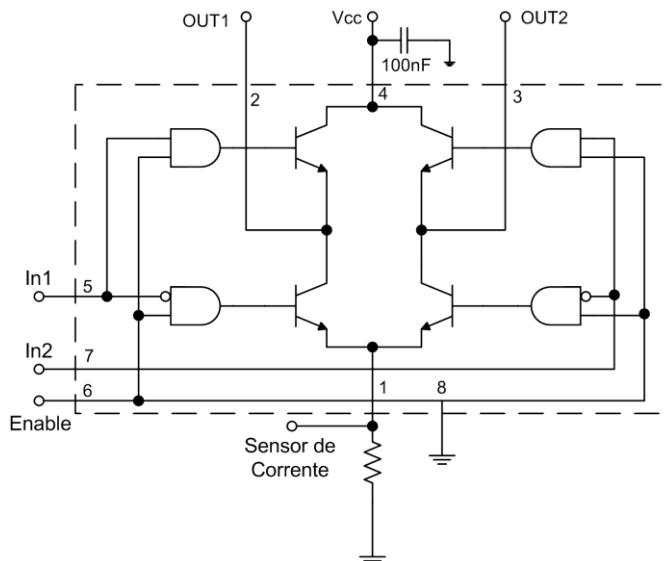


Fig. 8.14: Diagrama de blocos parcial do L298N

Tabela 8.6: Pinos de controle da ponte H

In 1	In 2	Efeito
0	0	Motor parado
0	1	Gira no sentido direto
1	0	Gira no sentido reverso
1	1	Motor parado

O programa-exemplo a seguir é para acionamento de um motor de corrente contínua cujo driver está conectado aos pinos P1.0 e P1.1 do microcontrolador 8051. Ele permite aumentar a velocidade através da interrupção zero e diminuir através da interrupção 1. O temporizador zero é usado no modo 2 (recarga automática) para gerar um sinal de período constante, mas cujo intervalo de tempo em nível alto e baixo é alterado pelas interrupções.

Motor: Acionamento de um motor de corrente contínua com controle de velocidade

Rótulo	Mnemônicos	Comentários
SAIDA0	EQU P1.0	; Bit 0 de P1 – para acionar o motor
SAIDA1	EQU P1.1	; Bit 1 de P1 – para acionar o motor
TON	EQU 03H	; Armazena período ligado
TOFF	EQU 04H	; Armazena período desligado
LIGADO	BIT 00H	; Flag que indica motor ligado/desligado
ORG 00H		
LJMP INICIO		
ORG 03H		
LJMP AUM_VELOC		; Interrupção INT0 – aumenta velocidade
ORG 0BH		
LJMP ATENDE_TEMP		; Interrupção TEMP0 → controla velocidade
ORG 13H		
LJMP DIM_VELOC		; Interrupção INT1 → diminui velocidade

	ORG 30H	
INICIO:	MOV SP,#2FH	
	MOV IE,#87H	; Habilita interrupções INT0, INT1 e TEMP 0
	MOV TCON,#05H	; INT0 e INT1 são por transição
	MOV IP,#02H	; Faz TEMP 0 com prioridade 1
	MOV TMOD,#02H	; TEMP 0 no modo 2 → com recarga
	MOV TON, #01H	; Período ligado TON = 01h
	CLR SAIDA0	; Desliga motor
	CLR SAIDA1	; Desliga motor
	CLR LIGADO	; Limpa flag que indica motor ligado/desligado
	MOV TL0,TON	; TL0 = período ligado
	MOV A,TON	; Faz A = período ligado
	CPL A	; Acha período desligado
	MOV TH0,A	; TH0 = período desligado
	SETB TR0	; Inicia temporizador temp0 a partir de TON
	SJMP \$; Laço de espera infinito.
ATENDE_TEMP:	JB LIGADO, DESLIGA	; Se motor está ligado → desliga
	SETB SAIDA0	; Liga motor, que estava desligado
	SETB LIGADO	; Seta flag que indica motor ligado
	MOV TH0,TON	; TH0 = período ligado. Valor de recarga
	RETI	
DESLIGA:	CLR SAIDA0	; Desliga motor
	CLR LIGADO	; Limpa flag que indica motor ligado
	MOV TH0, TOFF	; Faz A = período desligado
	RETI	
AUM_VELOC:	CLR EX0	; Desabilitar interrupção externa zero
	CLR CY	; Limpa flag de carry
	MOV A, TON	; Faz A = período ligado
	ADD A,#0AH	; Faz A = A + 10
	JNC PULA	; Se CY = 0 desvia para pula
	MOV A,#0FEH	; Faz A = feh, se CY = 1
PULA:	MOV TON,A	; Faz período ligado = A
	CPL A	; Encontra período desligado
	MOV TOFF, A	; Define período desligado
	SETB EX0	; Reabilita interrupção zero
	RETI	
DIM_VELOC:	CLR EX1	; Desabilitar interrupção externa 1
	CLR CY	; Limpa flag de carry
	MOV A, TOFF	; Faz A = período desligado
	ADD A,#0AH	; Faz A = A + 10
	JNC PULA2	; Se CY = 0, desvia para pula2
	MOV A,#0FEH	; Faz A = FEH, se CY = 1
PULA2:	MOV TOFF,A	; Faz período desligado = A
	CPL A	; Encontra período ligado
	MOV TON, A	; Define período ligado
	SETB EX1	; Reabilita interrupção 1
	RETI	
	END	

8.7 Motor de Passo

O motor de passo consiste de um estator contendo bobinas que são acionadas usando corrente contínua e um rotor de ímã permanente, que gira a cada mudança das bobinas acionadas. Embora nas figuras a seguir as quatro bobinas sejam mostradas como únicas (concentradas), na prática elas são distribuídas ao longo do estator. Assim, pode-se obter um passo bem pequeno entre um “pedaço” de bobina e outro. O ângulo de passo típico de um motor de passo é $1,8^\circ$. As figuras a seguir ilustram, de maneira simplificada, o funcionamento de um motor de passo. Observe que, na figura, uma volta completa do motor é alcançada após percorrer todas as bobinas uma vez. Na prática, uma volta completa é conseguida após uma passagem por todas as “partes” de cada bobina. Se cada passo for de $1,8^\circ$, uma volta completa é alcançada com 200 passos.

Na Fig. 8.15 são mostrados os transistores usados no acionamento e as bobinas concentradas. A Tabela 8.7 mostra os comandos que devem ser enviados para a porta de saída de forma que o motor de passo gire de meio em meio passo e com passo completo.

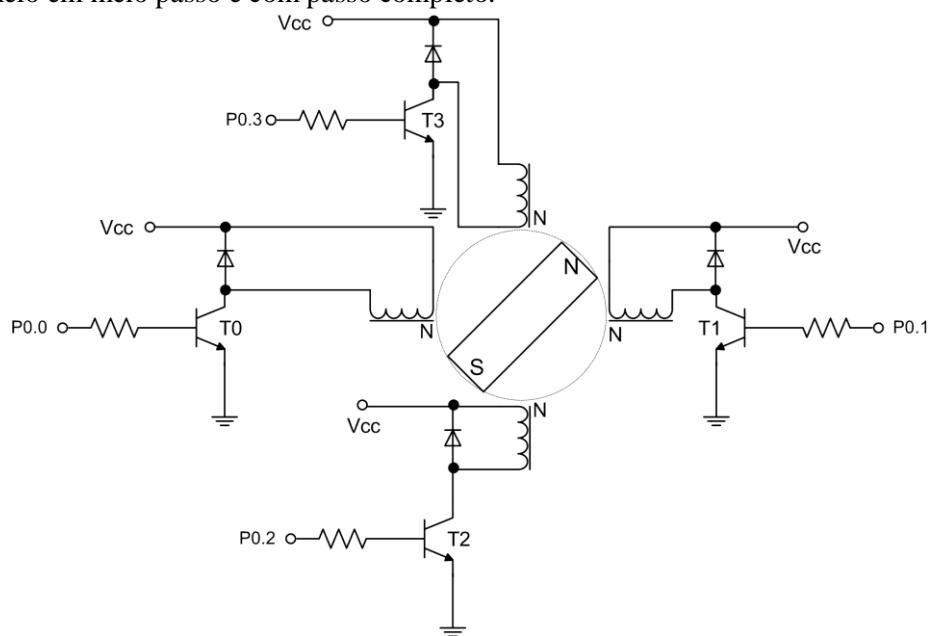


Fig. 8.15: Esquema que ilustra um motor de passo

Tabela 8.7: comandos para meio passo e passo completo

Passo	T0	T1	T2	T3	HEX
1	1	0	1	0	0A
2	0	0	1	0	02
3	0	1	1	0	06
4	0	1	0	0	04
5	0	1	0	1	05
6	0	0	0	1	01
7	1	0	0	1	09
8	1	0	0	0	08
9	1	0	1	0	0A

Passo	T0	T1	T2	T3	HEX
1	1	0	1	0	0A
2	0	1	1	0	06
3	0	1	0	1	05
4	1	0	0	1	09
5	1	0	1	0	0A

O circuito da Fig. 8.15 é o circuito típico utilizado no acionamento de motor de passo, entretanto, há pastilhas integradas que são também utilizadas para essa função. Um desses circuitos é o driver **ULN2004A**. A estrutura interna desse componente é mostrada na Fig. 8.16. Trata-se de um conjunto de transistores do tipo darlington, com capacidade de corrente de 500 mA. Cada uma das 4 bobinas do motor é ligada ao terminal comum (COM), que é conectado ao Vcc, e a uma das saídas (OUT).

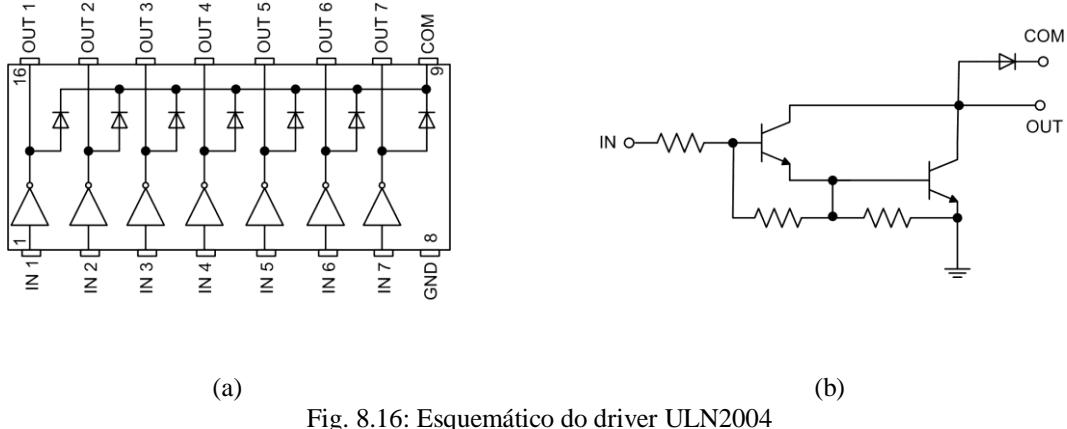


Fig. 8.16: Esquemático do driver ULN2004

Para ilustrar o funcionamento do motor de passo são apresentados a seguir dois exemplos:

Exemplo 1: Neste exemplo um motor de passo, conectado à porta P2, é acionado nos dois sentidos de rotação. Um sensor de presença no pino P3.0 define o sentido direto e um sensor no pino P3.1 define o sentido inverso.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	SENSOR1 EQU P3.0	
	SENSOR2 EQU P3.1	
	ORG 00H	
	LJMP INICIO	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	CLR F0	; Limpa flag que será usada para definir sentido de rotação (PSW.5)
COMEKO:	MOV A,#00H	
DECIDE:	JB SENSOR1,OUTRO	; Se SENSOR1 = 1, verificar o OUTRO sensor
	CLR F0	; Se SENSOR1 = 0, limpa o bit F0, que indica sentido de rotação
OUTRO:	JB SENSOR2,V1	; Se SENSOR2 = 1, pula para acionamento do motor
	SETB F0	; Se SENSOR2 = 0, faz F0 = 1
V1:	JB F0, INVERSO	; Se F0 = 1, desvia para INVERSO
	MOV DPTR,#DIRETO	; Se F0 = 0, faz DPTR igual ao endereço inicial da tabela DIRETO
	SJMP FRENT	; Desvia para acionar o motor no modo direto
INVERSO:	MOV DPTR,#INVERSO	; Se F0 = 1, faz DPTR igual ao endereço inicial da tabela INVERSO
FRENTE:	PUSH ACC	; Guarda contador da tabela
	MOVC A, @A+DPTR	; Carrega A com conteúdo da tabela (endereço A + DPTR)
	CJNE A,#0FFH,V2	; Se A = FFH, chegou ao fim da tabela. Volta pro começo
	POP ACC	; Recupera contador da tabela
	SJMP COMEKO	
V2:	MOV P2,A	; Envia conteúdo da tabela para porta P2
	LCALL ATRASO	
	POP ACC	
	INC A	
	SJMP DECIDE	
ATRASO:	MOV R4,#3FH	; Carrega registrador R4 com valor hexadecimal 3FH
V4:	MOV R5,#3FH	; Carrega registrador R5 com valor hexadecimal 3FH
	DJNZ R5,\$; Aguarda registrador R5 zerar
	DJNZ R4,V4	; Decrementa R4. Enquanto não for zero, volta para recarregar R5
	RET	; Retorna de subrotina de atraso de tempo

DIRETO:	DB 0AH	
	DB 06H	
	DB 05H	
	DB 09H	
	DB OFFH	
INVERSO:	DB 09H	
	DB 05H	
	DB 06H	
	DB 0AH	
	DB OFFH	
	END	

Exemplo 2: Esse exemplo usa a interrupção 1 para inverter o sentido de rotação do motor de passo. Assim, a qualquer pedido de interrupção 1 (por transição) provoca mudança no sentido de rotação.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 13H	; Endereço da interrupção externa 1
	CPL F0	; Complemeta bit usado para mudança de sentido de rotação
	RETI	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	CLR F0	; Limpa flag que será usada para definir sentido de rotação (PSW.5)
	MOV IE,#84H	; Habilita interrupção externa 1
	MOV TCON,#04H	; Interrupção 1 por transição
COMEKO:	MOV A,#00H	
DECIDE:	JB F0, INVERSO	; Se F0 = 1, desvia para INVERSO
	MOV DPTR,#DIRETO	; Se F0 = 0, faz DPTR igual ao endereço inicial da tabela DIRETO
	SJMP FRENT	; Desvia para acionar o motor no modo direto
INVERSO:	MOV DPTR,#REVERSO	; Se F0 = 1, faz DPTR igual ao endereço inicial da tabela REVERSO
FRENTE:	PUSH ACC	; Guarda contador da tabela
	MOVC A, @A+DPTR	; Carrega A com conteúdo da tabela (endereço A + DPTR)
	CJNE A,#0FFH,V2	; Se A = FFH, chegou ao fim da tabela. Volta pro começo
	POP ACC	; Recupera contador da tabela
	SJMP COMEKO	
V2:	MOV P2,A	; Envia conteúdo da tabela para porta P2
	LCALL ATRASO	
	POP ACC	
	INC A	
	SJMP DECIDE	
ATRASO:	MOV R4,#3FH	; Carrega registrador R4 com valor hexadecimal 3FH
V4:	MOV R5,#3FH	; Carrega registrador R5 com valor hexadecimal 3FH
	DJNZ R5,\$; Aguarda registrador R5 zerar
	DJNZ R4,V4	; Decrementa R4. Enquanto não for zero, volta para recarregar R5
	RET	; Retorna de subrotina de atraso de tempo
DIRETO:	DB 0AH	
	DB 06H	
	DB 05H	
	DB 09H	
	DB OFFH	
REVERSO:	DB 09H	
	DB 05H	
	DB 06H	
	DB 0AH	
	DB OFFH	
	END	

8.8 Lâmpada Incandescente

A Fig. 8.17 mostra um circuito para acionamento de uma lâmpada incandescente. É necessário apenas um bit (P1.0) para o controle da lâmpada. No entanto, é fundamental que esse bit seja zerado no início do programa, uma vez que todos os bits das portas ficam em nível alto após o reset e deseja-se que a lâmpada esteja apagada no início do programa. É utilizada uma lâmpada de 20 W / 220V e um relé para o circuito de acionamento.

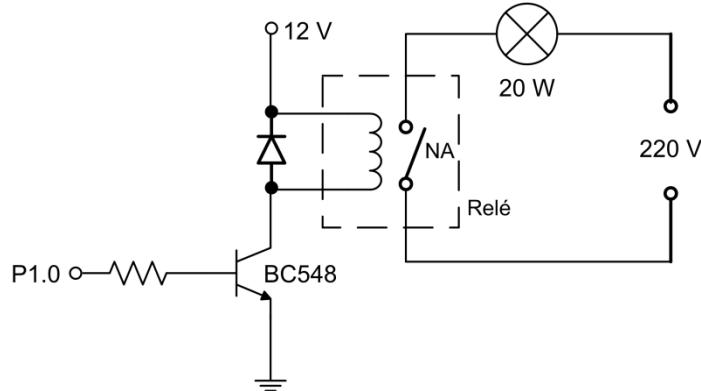


Fig. 8.17 – Acionamento de uma lâmpada incandescente

Lâmpada: Acionamento de lâmpada através de sensor de presença conectado à interrupção zero. O sensor é usado para acender e apagar a lâmpada.

Rótulo	Mnemônico	Comentário sobre o Efeito da Operação
	ORG 00H	
	LJMP INICIO	
	ORG 03H	; Endereço da interrupção externa 1
	CPL P1.0	; Complementa bit usado para acender/apagar a lâmpada
	JNB P3.2,\$; Laço para reduzir interferência da trepidação da chave (debouncing)
	RETI	
	ORG 30H	
INICIO:	MOV SP,#2FH	; Apontador de pilha SP = 2FH
	CLR P1.0	; Limpa bit que aciona a lâmpada (Apaga a lâmpada)
	MOV IE,#81H	; Habilita interrupção externa zero
	MOV TCON,#01H	; Interrupção 0 por transição
	SJMP \$	
	END	

9 Bibliografia

1. PARHAM, Behrooz, "Arquitetura de Computadores: de microprocessadores a supercomputadores," McGraw-Hill, São Paulo, 2007.
2. TANENBAUM, Andrew S., "Organização Estruturada de Computadores," 5^a ed., Pearson Prentice Hall, São Paulo, 2007.
3. SILVA JÚNIOR, Vidal Pereira da, "Aplicações Práticas do Microcontrolador 8051," Érica, São Paulo, 1994.
4. YERALAN, Sencer e AHLUWALIA, Ashutosh, "Programming and Interfacing the 8051 Microcontroller," Addison Wesley, Reading, 1995.
5. MACKENZIE, I. Scott, "The 8051 Microcontroller," Prentice Hall, New Jersey, 1995.
6. GIMENEZ, Salvador P., "Microcontrolador 8051," Prentice Hall, 2002.
7. NICOLOSI, Denys E. C., "Microcontrolador 8051 Detalhado," Érica, São Paulo, 2000.
8. FLEURY, Cláudio A. e BARBACENA, Ilton L., "Microcontrolador 8051," Goiânia, Março 1997.