Programação de Microcontroladores I

M.M. / 2001 (revisão)
L.M.S. e P.S.C. / 2001 (revisão)
E.T.M./2004 (revisão)
E.T.M./2005 (revisão)
E.T.M./2010 (revisão)
E.T.M./2011 (revisão da parte experimental)

RESUMO

Nesta experiência serão estudados os conceitos básicos de microprocessadores e microcontroladores e de sua programação. A parte experimental consiste no desenvolvimento de atividades para a compreensão do funcionamento de programas na linguagem de montagem ou *assembly* e testes dos mesmos numa placa experimental do microcontrolador Intel 8051.

1. Parte Teórica

1.1. Microcomputadores

Os microcomputadores são muito populares hoje em dia, pois servem para várias aplicações, desde balanças eletrônicas, piloto automático de carros, reserva de passagens e controle de orçamento doméstico, até controle de reatores nucleares.

Esta popularização deve-se em grande parte ao avanço da Microeletrônica que disponibiliza rapidamente componentes cada vez mais poderosos. Além disto, os programas que implementam estas aplicações estão se tornando cada vez mais complexos e necessários nos dias atuais.

O elemento central dos microcomputadores é o microprocessador. Adicionando-se memórias e interfaces de entrada/saída (E/S), teremos a arquitetura básica de todos os microcomputadores comerciais, conforme a Figura 1.1.

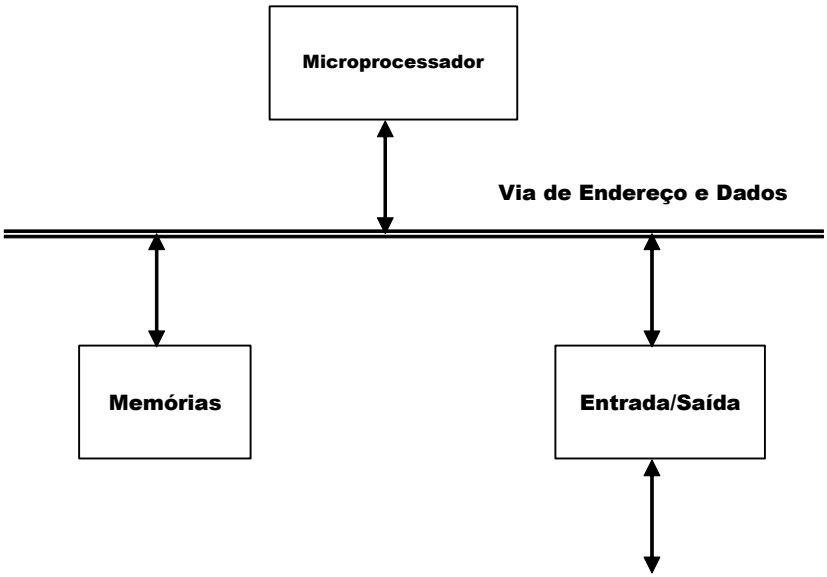


Figura 1.1 – Arquitetura Básica de um Microcomputador.

1.2. O Microprocessador 8080

O primeiro microprocessador a ser mais conhecido no mundo inteiro e que impulsionou o desenvolvimento de outros microprocessadores até os dias de hoje é o microprocessador 8080 da Intel, que foi lançado em dezembro de 1973. Apesar de antigo é importante conhecer a sua arquitetura e verificar que os seus conceitos básicos estão presentes nos microprocessadores mais recentes. O diagrama funcional do microprocessador 8080 pode ser visto na Figura 1.2.

A memória local do microprocessador 8080 é constituída de um conjunto de registradores temporários, de uso geral, ponteiro de pilha (SP) e contador de instrução (PC). Os registradores de uso geral (B, C, D, E, H e L) são endereçáveis diretamente por instruções de programa e podem ser utilizados aos pares (registradores de 16 bits) ou isoladamente (registradores de 8 bits). Os registradores temporários são utilizados pela unidade de controle e não são acessíveis pelo programador. O ponteiro de pilha define, no começo do programa, o início da pilha. O contador de instrução é automaticamente incrementado, através do acionamento do circuito de incremento/decremento, durante cada fase de busca de instrução (*fetch*), e indica o endereço da próxima instrução a ser executada.

A comunicação entre a memória local e a via interna de dados é feita através do multiplexador, em blocos de 8 bits. A via externa de dados é acessada através do registrador de dados de saída. O registrador possui excitadores que operam em três modos: desligado (a via externa fica isolada da interna); saída (os dados existentes na via interna são colocados na externa); e entrada (os dados da via externa são colocados na via interna, de acordo com a operação que está sendo executada).

As operações lógicas e aritméticas são realizadas pela ULA, que tem como entradas os registradores *Reg Temp*, *Reg Acum* e o *Flag* (vai-um). As instruções aritméticas são realizadas, em geral, com o Acumulador. O registrador *Flags*, constituído dos bits "zero", "vai um", "sinal", "paridade" e "vai um auxiliar", é atualizado por instruções aritméticas e lógicas, conforme o resultado da operação.

A Unidade de Controle, em função do código da operação e tendo como referência sinais de relógio, fornece os sinais de controle do fluxo de dados (interno), além de gerar os sinais para controle externo.

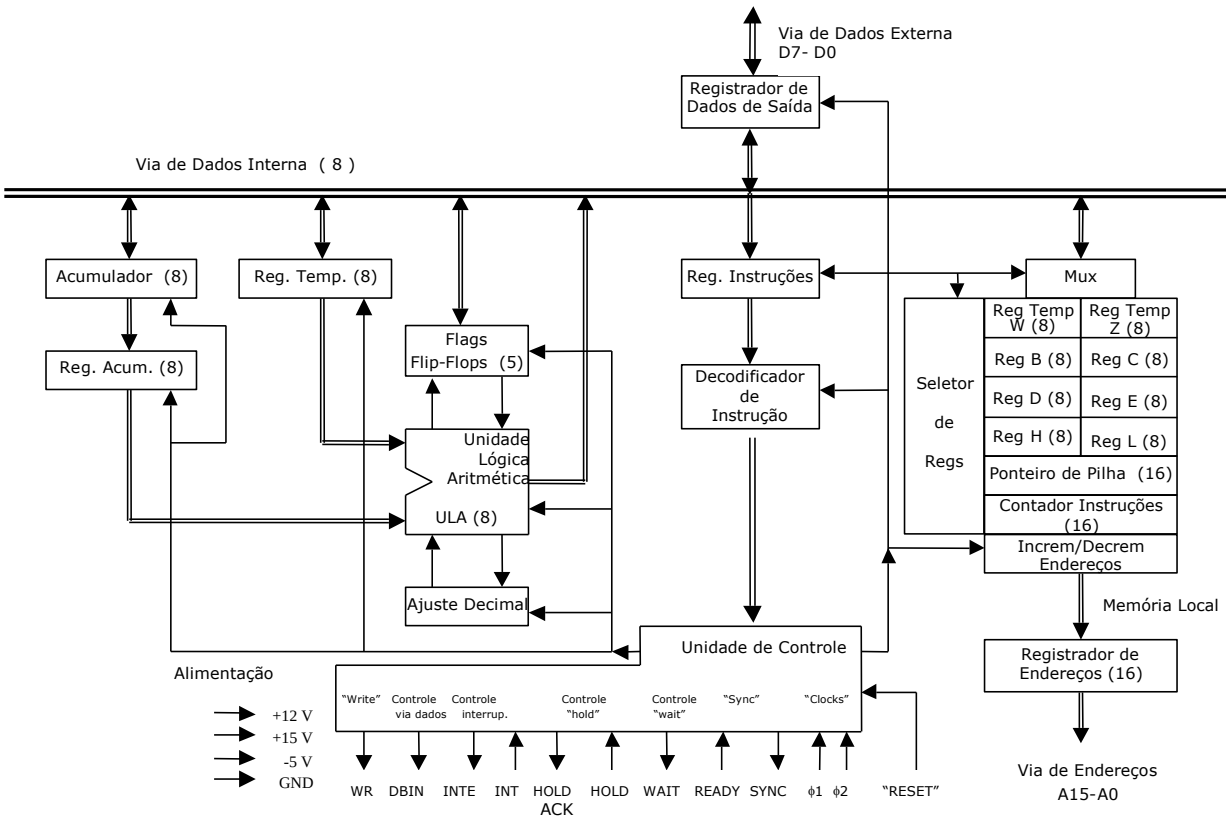


Figura 1.2 – Diagrama Funcional do Microprocessador 8080.

1.3. Microcontroladores

Um microcontrolador é um componente que apresenta, além da CPU, as memórias RAM e ROM, interfaces para entrada e saída de periféricos paralelos ou seriais e, opcionalmente, temporizadores e contadores, todos integrados em uma única pastilha. Um microcontrolador típico apresenta instruções de manipulação de bits, dispõe de acesso fácil e direto às interfaces de entrada e saída, bem como processa interrupções rápida e eficientemente.

O microcontrolador é amplamente utilizado em aplicações voltadas para robótica, controle e automação industrial, indústria automotiva, controle de periféricos e equipamentos de comunicação de dados.

Muitos microcontroladores são baseados no conceito *CISC (Complex Instruction Set Computer*) dispondo, geralmente, de mais de 80 instruções, muitas das quais poderosas e destinadas especificamente a tarefas de controle. É muito comum que a execução das instruções seja realizada de forma diferenciada, sendo que muitas delas operem somente sobre certos intervalos de endereços ou registradores, e outras reconheçam somente determinados modos de endereçamento.

A arquitetura RISC (*Reduced Instructions Set Computers*) compõe outra parte do mercado dos microcontroladores, incorporando-lhes suas vantagens como: área menor de silício, encapsulamento com menor número de pinos e menor consumo de potência.

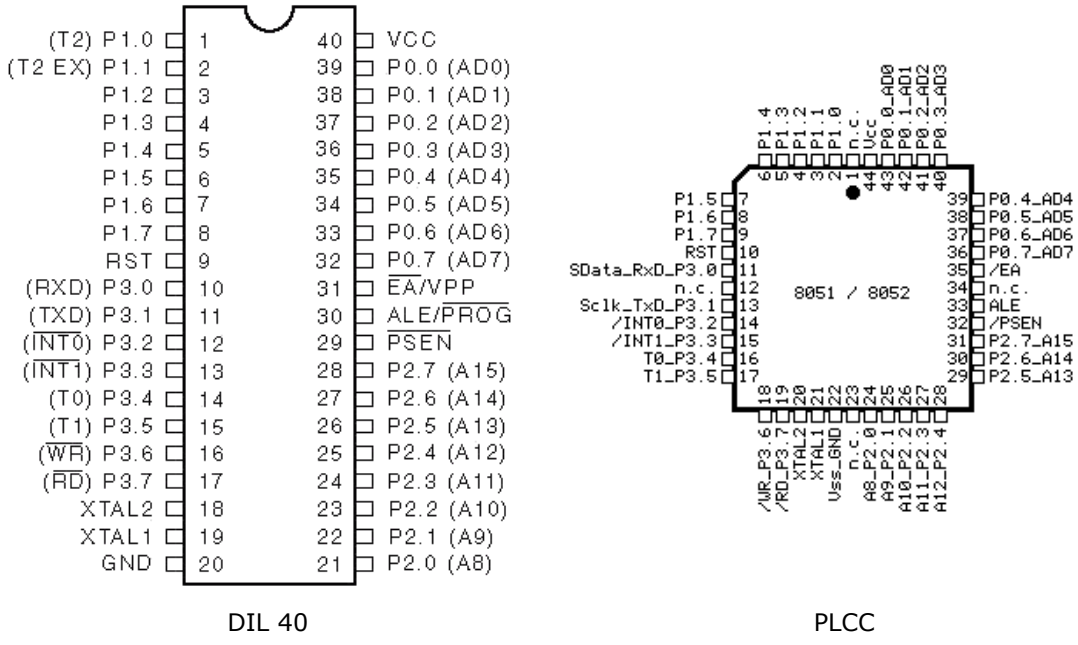
1.4. O Microcontrolador 8051

O 8051 é um microcontrolador de 8 bits desenvolvido em 1980 pela empresa americana Intel. Este microcontrolador faz parte de uma família de componentes desenvolvidos originalmente para funções especiais como controle de teclado, mouse e impressora. Posteriormente, a utilização do 8051 estendeu-se para aplicações mais genéricas. Atualmente o mercado contempla um grande número de versões dos mais diferentes fabricantes.

Na configuração básica, os componentes da família 8051 possuem as seguintes características:

- 4 Kbytes de memória ROM
- 128 bytes de memória RAM
- 4 portas de I/O de 8 bits cada uma, bits estes individualmente endereçáveis
- 2 temporizadores / contadores de 16 bits
- 1 porta serial UART *full duplex*, que permite a expansão de I/O.
- interrupções com estrutura "*nesting*" com cinco fontes selecionáveis e níveis de prioridade a escolher
- oscilador interno

A Figura 1.3 ilustra a pinagem do 8051 em encapsulamento DIL e em encapsulamento PLCC e a Figura 1.4 mostra um diagrama de blocos simplificado da arquitetura interna dos componentes da família básica Intel MCS 51.



DIL 40 PLCC

Figura 1.3 – A pinagem do 8051.

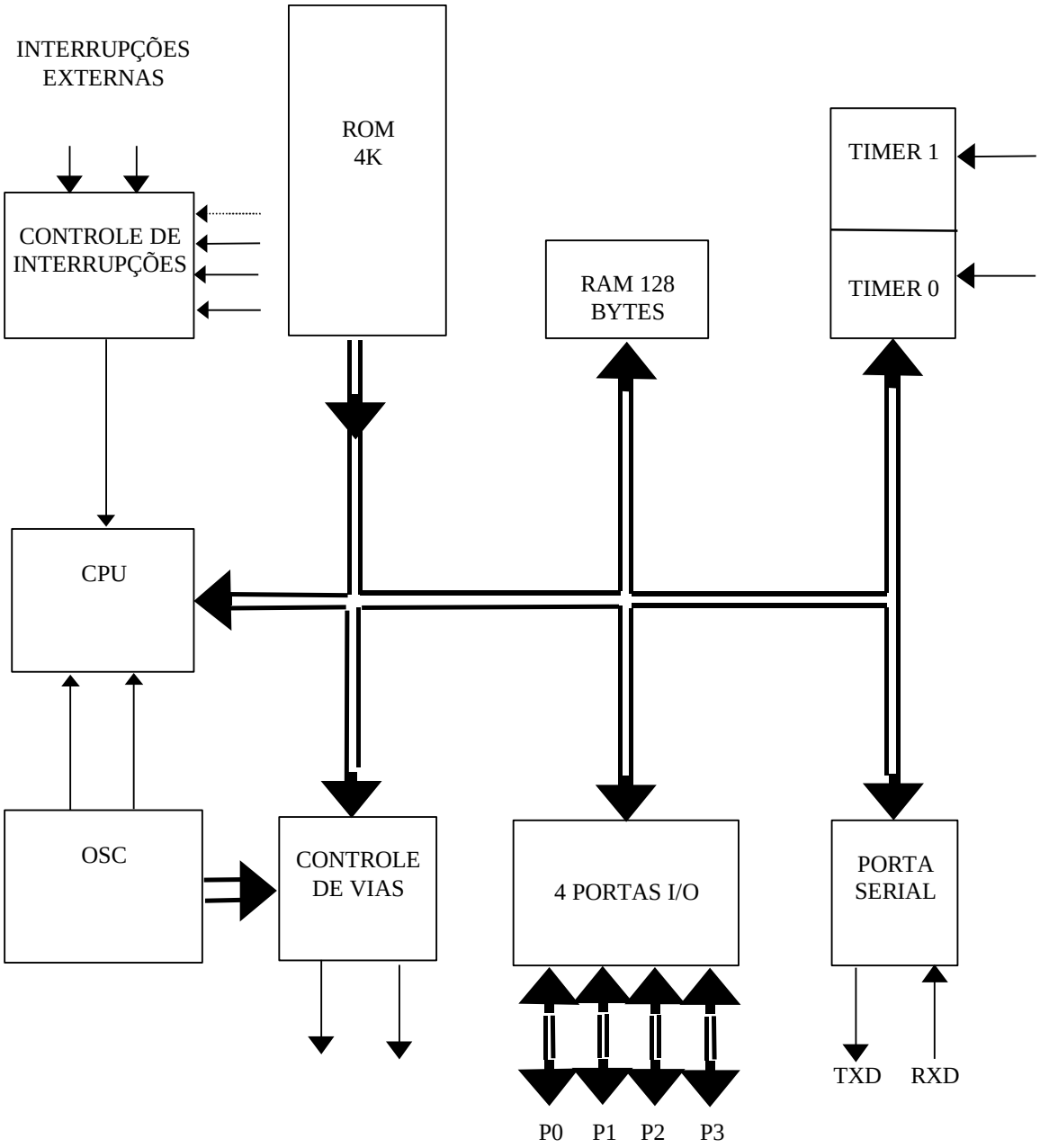


Figura 1.4 – Diagrama Simplificado do Microcontrolador 8051.

1.4.1. Organização da Memória

O 8051 apresenta áreas de armazenamento distintas para programas e dados. Inicialmente endereça 4Kbytes de memória de programa e 128 bytes de RAM, além dos registradores de funções especiais, cujos endereços variam de 80h a FFh.

Para a expansão externa da memória de programa, há duas possibilidades: são empregados 64Kbytes de memória externa ou 4Kb de memória interna e 60 Kbytes de memória externa.

No reset, a CPU inicia a busca de instruções no endereço 0000h. A localização física deste endereço depende do sinal EAB. O acesso aos endereços 0000h a 0FFFh será na memória interna, se EAB for igual a 1, e na memória externa, se EAB for igual a 0. Os primeiros endereços da memória de programa devem ser reservados para alocar os vetores das interrupções e reset.

Para a expansão da memória externa de dados podem ser utilizados 64 Kbytes de memória externa, independentemente da utilização da memória RAM interna. O acesso à localização física da memória de dados se diferencia pelos tipos de instruções e modos de endereçamentos.

Exemplos:

```
; endereçamento direto e instrução MOV
MOV 30H, A      ; move o conteúdo do Acumulador para o endereço 30h da RAM interna

; endereçamento indireto e instrução MOVX
MOV R0, #30H    ; move para o registrador R0 o número 30h
MOVX @R0, A     ; move o conteúdo do acumulador para a posição 30h da RAM externa
                ; indiretamente endereçada por R0.
```

A Figura 1.5 ilustra a organização da memória para o 8051.

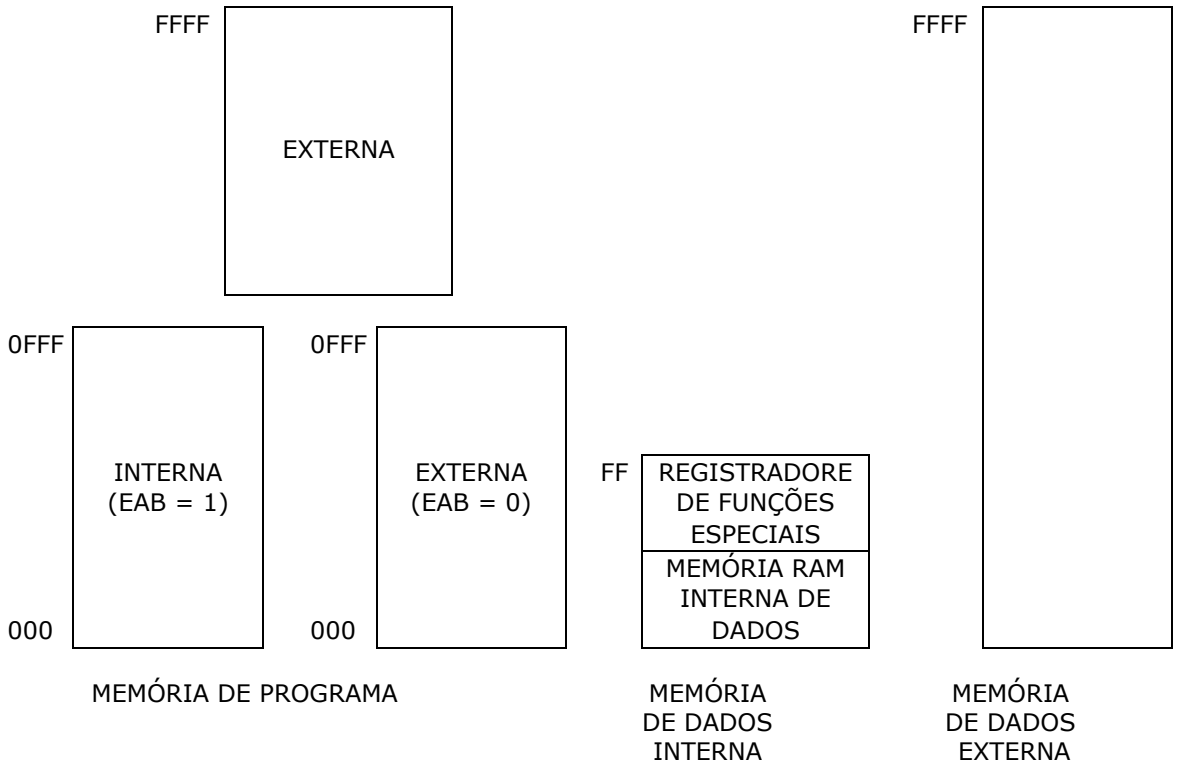


Figura 5 – Organização da memória para o 8051.

A área da RAM interna de 128 bytes está dividida em três partes:

- banco de registradores (endereços 00 a 1Fh).
- área de bits e bytes endereçáveis (endereços 20h a 2Fh).
- memória com bytes endereçáveis (30h a 7Fh).

A Figura 1. 6 apresenta o diagrama dos 128 bytes da memória RAM.

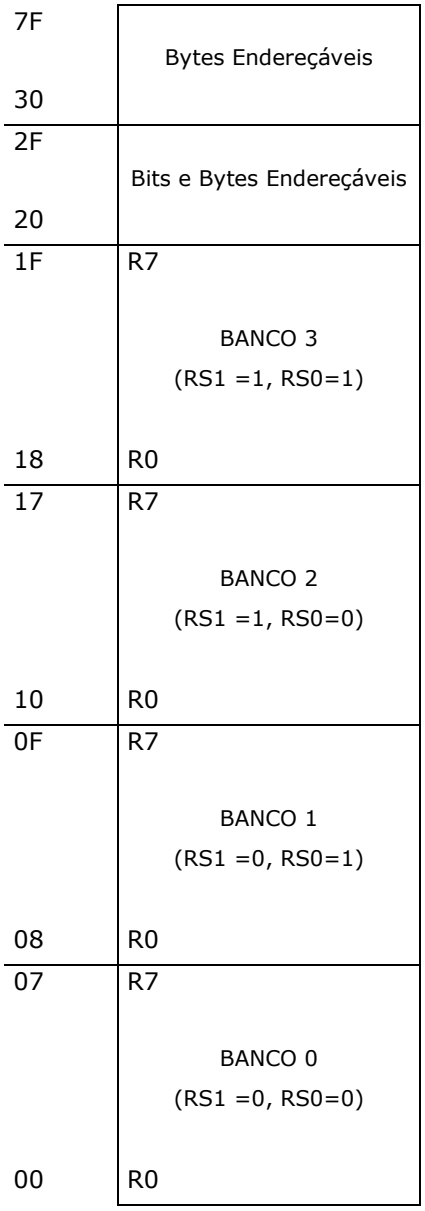


Figura 1.6 – Diagrama da parte baixa da RAM.

1.4.2. Banco de Registradores

Os bancos de registradores são posições da memória RAM que permitem seu endereçamento pelo nome dado a cada registrador, além de seu endereçamento pela posição da memória. É composto por quatro conjunto de oito registradores cada (R0 a R7). Somente um banco é disponível por vez e sua seleção se faz pelos bits RS1 e RS0 do registrador de função especial PSW. O banco 0 é o *default* no *reset*.

1.4.3. Registradores de Funções Especiais

Na parte superior da memória de dados interna estão localizados os registradores de funções especiais. Nem todas as 128 posições de memória estão ocupadas por estes registradores e alguns deles apresentam bits endereçáveis (ver Tabela I).

Tabela I - Registradores de Funções Especiais.

Registrador	Endereço	Função
P0	80H*	**Porta 0**: contém os dados da porta 0. Uma operação de escrita nesse registrador altera, automaticamente, o conteúdo presente na saída da porta 0 do 8051. Uma operação de leitura armazena no registrador os valores presentes na porta 0.
SP	81H	**Stack Pointer**: indica o último endereço de armazenamento na pilha. Permite uma pilha com no máximo 256 bytes. É incrementado antes do dado ser armazenado, ao serem utilizadas as instruções PUSH e CALL. A pilha pode estar localizada em qualquer local da RAM interna. No reset, o SP é inicializado com o valor 07h e, portanto, a pilha inicia no endereço 08h (banco 1 de registradores). (Obs. Se o usuário utilizar o banco 1 de registradores, deverá mover a pilha para endereços mais altos da RAM).
DPL	82H	**Data Pointer** (**low**). Este registrador, juntamente com o DPH, pode ser referenciado como um registrador de 16 bits, denominado DPTR. É empregado como ponteiro nas instruções que utilizam endereçamento indireto na leitura de constantes armazenadas na memória de programa. Também é empregado para operações de leitura e escrita de variáveis na memória externa de dados e para operação de desvio para a memória de programa.
DPH	83H	**Data Pointer** (**high**) (Ver DPL)
TCON	88H*	**Timer Register**: registrador de controle para a programação dos temporizadores/contadores.
TMOD	89H	**Timer Mode Register**: registrador de modo de operação para a programação dos temporizadores/contadores.
TL0	8AH	**Timer 0 Low byte**: registrador de dados (byte menos significativo) do temporizador/contador 0.
TL1	8BH	**Timer 1 Low byte**: registrador de dados (byte menos significativo) do temporizador/contador 1.
TH0	8CH	**Timer 0 High byte**: registrador de dados (byte mais significativo) do temporizador/contador 0.
TH1	8DH	**Timer 1 High byte**: registrador de dados (byte mais significativo) do temporizador/contador 1.
P1	90H*	**Porta 1**: idem à P0, para a porta 1.
SCON	98H*	**Serial Port Control Register**: registrador de controle para a utilização da porta serial.
SBUF	99H	**Serial Port Data Buffer**: registrador que armazena os dados recebidos ou enviados pela porta serial.
P2	0A0H*	**Porta 2**: idem à P0, para a porta 2.
IE	0A8H*	**Interrupt Enable Register**: registrador que seleciona as interrupções a serem habilitadas ou desabilitadas.
P3	0B0H*	**Porta 3**: idem à P0, para a porta 3. Para funções especiais, os bits desse registrador são utilizados independentemente.
IP	0B8H*	**Interrupt Priority Register**: indica prioridade de cada uma das interrupções.
PSW	0D0H*	**Program Status Word**: os bits desse registrador indicam quais as ocorrências da Unidade Lógica e Aritmética na última operação. Também indica qual dos bancos de registradores foi acessado por último.
ACC	0E0H*	**Acumulador**: armazena um dos operandos, bem como o resultado das instruções aritméticas e lógicas, e também pode ser utilizado como registrador de índice.
B	0F0H*	**Registrador B**: tem função específica nas operações de multiplicação e divisão, podendo também ser utilizado como um registrador de propósitos gerais.

* *denota registrador com bits endereçáveis*

A Figura 8 apresenta os registradores de funções especiais que apresentam bits endereçáveis. Observe-se que o endereço do bit menos significativo coincide com o endereço de byte do registrador.

Endereço dos bytes	**Endereço dos bits**								**Registrador**
	MSB							LSB	
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
B8H	-	-	-	BC	BB	BA	B9	B8	IP
B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8H	AF	-	-	AC	AB	AA	A9	A8	IE
A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Figura 1.7 – Registradores de funções especiais com bits endereçáveis.

Será deixado para cursos posteriores o detalhamento dos seguintes registradores de funções especiais:

- TH0, TL0, TH1, TL1, TMOD e TCON: referentes aos temporizadores/contadores 1 e 0.
- SBUF, SCON: referentes à porta serial.
- IP, IE: referentes ao tratamento de interrupções.

1.4.4. O Registrador PSW

O registrador *Program Status Word* contém os bits de *flags* como representado abaixo. Cada bit é designado pelo seu nome ou pelo nome do registrador, seguido pelo ponto decimal e o número da posição do bit dentro do registrador.

CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**-**	**P**
D7	D6	D5	D4	D3	D2	D1	D0

onde:

PSW.7 ou CY = *flag* de *carry*.

PSW.6 ou AC = *flag* auxiliar de *carry*.

PSW.5 ou F0 = *flag* geral, tem seu estado definido pelo *software*.

PSW.4, PSW.3 ou RS1, RS0, respectivamente: selecionam os bancos de registradores.

00 = banco 0 (00 - 07H)

01 = banco 1 (08 - 0FH)

10 = banco 2 (10 - 17H)

11 = banco 3 (18 - 1FH)

PSW.2 ou OV = *flag* de *overflow*.

PSW.0 ou P = *flag* de paridade.

1.4.5. Modos de Endereçamento

As instruções presentes no conjunto de instruções do microcontrolador 8051 apresentam cinco modos de endereçamento, a saber:

- Registrador
- Direto
- Indireto por Registrador
- Imediato
- Indexado

Endereçamento por Registrador

Neste modo de endereçamento, o nome do registrador a ser acessado está incluído no código de operação. O registrador a ser usado pertence ao banco de registradores selecionado pelos bits RS1 e RS0 do registrador PSW.

Exemplos:

```
MOV R1,A      ;move o conteúdo do acumulador para o registrador R1.
MOV 01,A      ;sendo 01 o endereço de R1, move o conteúdo do acumulador para o
              ;registrador R1.
MUL AB        ;multiplica o conteúdo do acumulador pelo conteúdo do registrador B e
              ;armazena o resultado no par constituído pelos registradores B e A.
```

Endereçamento Direto

Estas instruções possuem dois bytes: um código de operação seguido de um endereço de 8 bits. O endereço refere-se à memória RAM interna ou a algum dos registradores de funções especiais. O endereço pode se referir a um byte ou a um bit, conforme a instrução.

Exemplos:

```
MOV A,80H     ;move o conteúdo da porta 0 para o acumulador.
MOV A,07H     ;move o conteúdo do endereço 07h da RAM interna para o acumulador.
CLR 07H       ;trata-se de uma instrução de referência a bit. Portanto, torna zero o bit de
              ;endereço 07 que é o bit mais significativo do endereço 20h.
```

Endereçamento Indireto por Registrador

Neste tipo de endereçamento é especificado o registrador que contém o endereço do operando. Usando este modo pode-se acessar tanto a memória RAM externa como a interna e os endereços podem ser de 8 bits (apenas para os registradores R0 e R1) ou de 16 bits. Neste último caso, utiliza-se o registrador DPTR (ver Tabela I, registradores DPL e DPH).

Exemplos:

```
MOV A,@R0     ;copia o conteúdo do endereço da RAM interna apontada por R0 para o acumulador.
MOVX A,@DPTR  ;copia o conteúdo do endereço da RAM externa apontada por DPTR para o acumulador.
```

Endereçamento Imediato

Essas instruções usam como operando imediato um valor de constante numérica que se segue ao *opcode*. A constante pode ser de 8 ou 16 bits, dependendo da instrução.

Exemplos:

```
MOV A,#25H        ;armazena no acumulador a constante 25h.
MOV DPTR,#2515H   ;armazena a constante 2515h no registrador DPTR.
```

Endereçamento Indexado

Neste modo de endereçamento, o endereço efetivo é a soma do conteúdo do acumulador e de um registrador de 16 bits armazenado no registrador PC ou no registrador DPTR.

Exemplos:

```
MOV A,@A+PC   ;copia o byte presente no endereço obtido pela soma dos registradores A e
              ;PC, no acumulador.
JMP @A+DPTR   ;desvia para o endereço obtido pela soma dos conteúdos dos registradores
              ;A com DPTR
```

1.4.6. Instruções Referentes à Pilha

Pilhas são estruturas de dados gerenciados automaticamente pelo microprocessador, cujo armazenamento de dados ou endereços se procede de tal forma que o elemento inserido por último é o primeiro a ser removido (LIFO). A pilha é endereçada pelo registrador denominado *Stack Pointer* (SP).

No microcontrolador 8051 são possíveis duas operações de pilha: PUSH e POP. O modo de endereçamento dessas instruções é direto (Observe-se que as instruções de endereçamento direto referenciam a área de memória RAM interna).

A instrução **PUSH** insere um elemento no topo da pilha. Em sua execução, o *stack pointer* é incrementado de 1 e o valor do operando endereçado é copiado na memória cuja posição é endereçada pelo *stack pointer*.

Exemplos:

```
PUSH DPL      ;move o conteúdo do registrador DPL para a posição de memória endereçada pelo
              ;registrador SP.
PUSH A        ;move o conteúdo do registrador A para a posição de memória endereçada por SP
```

A instrução **POP** remove o elemento armazenado no topo da pilha. Na execução dessa instrução, o conteúdo da memória cuja posição é endereçada pelo SP é lido e transferido para o operando presente na instrução. O valor do registrador SP é decrementado de 1.

Exemplos:

```
POP DPL       ; move o conteúdo da memória endereçada pelo registrador SP para o registrador DPL.
POP R1        ; move o conteúdo da memória endereçada por SP para R1.
```

1.4.7. Utilização da Pilha

Chamada e Retorno de Sub-rotina

Há três operações que devem ser realizadas pelo microprocessador, quando uma instrução de chamada de sub-rotina é executada, a saber:

- Conteúdo do registrador PC, que coincide com o endereço da instrução a ser executada imediatamente após a chamada de sub-rotina, deve ser salvo.
- Execução da sub-rotina chamada.
- Uma vez concluída a execução da sub-rotina, o valor salvo no primeiro passo é copiado novamente para o registrador PC.

A pilha é utilizada nas instruções de chamada e retorno de sub-rotina para salvar o endereço da instrução a ser executada imediatamente após a chamada de sub-rotina, bem como para recuperá-lo após a execução da mesma. Se este procedimento não fosse disponível, o programador deveria ser responsável por tratar o endereço de retorno da sub-rotina.

Passagem de Parâmetros

Há três métodos de passagem de parâmetros de sub-rotinas.

a) A forma mais simples de passagem de parâmetros é aquela em que o programador seleciona um determinado espaço da **memória** para os dados ou para os endereços dos dados da sub-rotina. A desvantagem desse método é que se deve assegurar que na execução do programa instruções de desvio não referenciem essa área de dados.

b) Outro método é empregar **registradores** de uso geral. Os parâmetros são armazenados nos registradores antes da chamada de sub-rotina que os referenciará durante a execução. Uma das desvantagens deste procedimento se encontra na necessidade de salvar o conteúdo dos registradores antes do armazenamento dos parâmetros, caso sejam empregados posteriormente. Outra desvantagem é que dado o número limitado de registradores, o número de parâmetros também é limitado.

c) Num terceiro método as sub-rotinas acessam indiretamente os parâmetros a partir do ponteiro da **pilha**. Tal procedimento evita que as instruções de desvio referenciem a área onde os valores dos parâmetros estão armazenados, dado que a área de dados e de programa são distintas daquela da pilha.

Exemplo:

```
; exemplo de utilização da pilha para passagem de parâmetros
      ORG   8000H
;
; PROGRAMA PRINCIPAL
; envia valor 15H para subrotina SROT e armazena valor de retorno
; para a posição de memória interna de endereço 33H
;
      MOV   B,#15H
      PUSH  B
      CALL  SROT
      POP   33H
LOOP: NOP
      JMP LOOP
;
; SUBROTINA SROT
; incrementa valor recebido pela pilha
;
SROT: MOV   R0,SP
      DEC   R0
      DEC   R0
      MOV   A,@R0
      INC   A
      MOV   @R0,A
      RET
;
      END
```

1.5. Os Programas X8051, Link2, XTALK e Monitor

Nesta experiência, são usados os seguintes programas: no PC, o *cross-assembler* **X8051**, o *link-editor* **LINK2** e o programa de comunicação serial **XTALK**, e o programa **Monitor** residente da placa experimental.

1.5.1. O Cross-Assembler X8051

No apêndice B observa-se o conjunto de instruções do microprocessador 8051. Um programa *assembler*, ou montador, tem como função fundamental gerar o código objeto correspondente, executando para isto as seguintes tarefas:

a) conversão dos códigos mnemônicos para o equivalente da linguagem de máquina.
b) conversão dos operandos simbólicos para seus endereços equivalentes na máquina.
c) geração das instruções de máquina no formato adequado.
d) conversão das constantes dos dados especificados no programa fonte em sua correspondente representação de máquina.
e) geração do programa objeto e da listagem assembly.

O X8051 é um Cross-Assembler, ou seja, permite a geração do código executável do 8051 no ambiente PC-MSDOS.

1.5.2. LINK2

Projetos mais complexos costumam ter vários arquivos de programas. Uma vez gerado o código objeto de cada programa, é necessário que os programas sejam conectados e transformados em código executável, fornecendo a informação necessária para que os diversos programas possam se referenciar mutuamente. Essa função é realizada por um programa denominado *linker* (ou ligador).

O *linker* utilizado denomina-se LINK2, que, além da função de ligar módulos, também converte o código objeto para um formato adequado para ser transmitido serialmente (formato "HEX").

1.5.3. XTALK

É o programa que será utilizado para a comunicação do PC com a placa experimental do 8051, permitindo assim que o código executável gerado no ambiente PC-MSDOS seja carregado na placa experimental (através do comando "SE"), bem como o PC possa emular um Terminal de Vídeo a ser utilizado pelo programa monitor do 8051. A comunicação serial entre o Kit e o Microcomputador é feita em 9600 bps. Os comandos fornecidos pelo teclado são repassados, pelo XTALK, para a placa experimental, via comunicação serial. Para que um comando seja fornecido ao próprio XTALK, é preciso, antes, acionar a tecla "ESC".

1.5.4. Programa Monitor

Na placa experimental está disponível um Programa Monitor residente que pode ser comparado, do ponto de vista funcional, ao sistema operacional de um microcomputador. Assim, o monitor opera como uma interface "homem-máquina", permitindo uma série de funções dentre as quais:

- Exibição das opções do programa monitor:
  H - Help
- Armazenamento de instruções de programas em memória:
  L - Lê e carrega HEX
- Exame do conteúdo das posições de memória:
  DM - Display Memory
- Exame do conteúdo dos registradores:
  DR - Display Registers
- Execução de programas:
  G - Executa
- Modificação do conteúdo da memória:
  M *- Modifica memória (sem ler)*
- Substituição do conteúdo da memória após verificação:
  S *- Substitui memória (lê e modifica)*
- Teste da memória RAM:
  T - Testa RAM

O programa monitor faz uso de um terminal de vídeo que, nesta experiência, será emulado pelo microcomputador, através do programa XTALK.

1.6. Procedimento de Geração, Carga e Execução de um Programa para o Kit do Microcontrolador 8051

Siga os procedimentos abaixo para a geração do programa executável no PC e posterior carga e execução no kit do microcontrolador 8051.

a. Edição do programa fonte:

No ambiente do PC, utilizando um editor de texto (p.ex. bloco de notas), digite o programa escrito na linguagem *assembly* do 8051. Salve o arquivo-fonte com a extensão .ASM.

b. Geração do código objeto:

Na janela de Prompt de Comando, digite o comando:

`X8051 NL <nome_do_programa>.ASM <nome_do_programa>.OBJ <enter>`

c. Geração do código executável:

Na janela de Prompt de Comando, digite o comando:

`LINK2 <enter>`

Serão solicitados, o nome do arquivo de entrada (digite o nome do programa com extensão .OBJ), o *offset* (faça *offset* = 0) e o nome do arquivo de saída (digite o nome do arquivo com extensão .HEX).

d. Utilização do programa XTALK para carga do código executável na placa experimental do 8051 e para utilização do programa monitor:

Na linha de comando do MSDOS, digite o seguinte comando:

`XTALK <enter>` [execução do programa XTALK]

`1 <enter>` [configuração da comunicação do 8051]

Pressione o botão *reset* da placa experimental do 8051. Neste momento, aparecerá na tela do computador a mensagem do monitor da placa experimental, indicando a entrada no programa monitor.

e. Forneça, no computador, a seguinte sequencia de comandos, via XTALK:

e.1. ao monitor: `H` [serão mostradas as opções de comando do monitor]

e.2. ao monitor: `L` [seleciona a opção de carga do programa na memória do kit]

e.3. ao XTALK: `<esc> SE <nome_do_programa>.HEX` [transfere o programa para a memória do *kit*]

e.4. ao monitor: `DM <endereço da memória>` [apresenta o conteúdo da memória]

e.5. ao monitor: `G <endereço da memória>` [executa o programa a partir do endereço especificado]

2. PARTE EXPERIMENTAL

2.1 Atividades Pré-Laboratório

Para os dois primeiros itens da parte experimental, considere o programa abaixo (figura 2.1), que subtrai o valor 6 do valor 23 e guarda o resultado na posição 9000H da memória externa. Repare que ao lado de cada instrução em linguagem *assembly* é mostrado o código de máquina correspondente.

a) Consulte o manual da Intel (disponível no *site* da disciplina) e verifique o funcionamento do programa abaixo, examinando o que cada instrução faz.

b) Estude o programa, analisando as instruções. Insira comentários explicando a função de cada instrução no programa fonte.

c) Qual é a função do registrador DPTR no microcontrolador 8051?

```
; Subtração binária (código assembly)        Código de máquina
;
; label    instrução     argumentos
;
            ORG           8000H
;
            MOV           DPTR,#9000H          90 90 00
            MOV           A,#23                74 17
            SUBB          A,#6                 94 06
            MOVX          @DPTR,A              F0
LOOP:       NOP                                00
            LJMP          LOOP                 02 80 08
;
            END
```

Figura 2.1 – Código do programa de subtração de dois números.

2.2. Programação em Linguagem de Máquina

Nesta seção, vamos trabalhar somente com o código de máquina do programa de subtração. O código de máquina está localizado na coluna da direita da figura 2.1.

ATENÇÃO: EXECUTE A SEQUENCIA DE ITENS ABAIXO NA ORDEM APRESENTADA.

d) Com base no estudo pré-laboratório do programa, qual é o resultado esperado ao final da execução do programa? Em que posição de memória este valor estará armazenado?

e) Usando o comando **M** (ou **S**) do monitor, introduza o código de máquina na memória do *kit*, a partir do endereço 8000H.

f) Antes de executar o programa, observe o conteúdo da memória na posição 9000H, através do comando DM.

g) Modifique o conteúdo da posição 9000H para "00H", através do comando **M** (ou **S**).

h) Execute o programa através do comando "G 8000". O que acontece neste instante? Por que existe o "loop eterno" no final do programa?

i) Localize o botão de *reset* no *kit*. Dê um *reset* na placa. Por que é necessário este *reset*?

j) Verifique o resultado da execução do programa na posição 9000H. Qual é o número hexadecimal obtido? Explique o resultado.

k) Usando o comando **M** (ou **S**) do monitor, apague (preenchendo com zeros) o código de máquina do programa na memória do *kit* (p.ex. apague do endereço 8000H até 800FH).

2.3. Programação em Linguagem Assembly

Nesta seção, vamos trabalhar com o programa em linguagem *assembly* (localizado na coluna da esquerda da figura 2.1). O programa vai ser compilado no PC e, em seguida, carregado na memória do *kit*, para ser executado.

l) A partir do código-fonte em linguagem *assembly* do programa de subtração, edite o programa no PC, gere o programa executável e transfira-o para o *kit*, seguindo os passos descritos na seção 1.6.

m) Repita os passos experimentais dos itens de (f) até (j).

n) Qual foi o resultado obtido? Compare-o com o valor obtido no item (j).

o) Qual a diferença encontrada entre o procedimento em que se insere diretamente o código de máquina e aquele em que se trabalha a partir do código na linguagem *assembly*?

2.4. Divisão de 2 bytes

Nesta seção vamos desenvolver um programa que lê dois números inteiros da memória externa, calcula a divisão inteira entre eles e armazena posteriormente os resultados (quociente e resto) na memória.

p) Procure no manual do microcontrolador 8051 a instrução *assembly* que efetua a divisão de dois bytes. Explique a sintaxe e o funcionamento desta instrução.

q) Como poderia ser codificada uma operação de leitura de um byte de uma posição de memória externa (p.ex. 9900H) para o registrador B do microcontrolador 8051?

r) Como poderia ser codificada uma operação de escrita dos bytes presentes nos registradores A e B para duas posições de memória externa consecutivas (p.ex. 9910H e 9911H).

s) Baseado no programa de substração binária, escreva um programa que divide dois bytes armazenados nas posições de memória externa 9000H e 9001H, armazenando o resultado nas posições 9002H (quociente) e 9003H (resto).

t) Antes de executar o programa, preencha os 2 operandos na memória com valores à sua escolha, e preencha com 0 as posições do resultado (use o comando M ou S do monitor).

u) Execute o programa para vários valores e confira os resultados.

v) Insira comentários junto ao código e anexe o programa fonte no relatório.

2.5. Somador Decimal (OPCIONAL)

w) Escreva um programa que implemente um somador decimal de dois dígitos BCD, utilizando o método da correção posterior de seis. Os dois valores BCD a serem somados estão armazenados nas posições de memória 9000H e 9001H e o resultado deve ser armazenado na posição 9010H.

Recomendações:

- Colocar comentários em todas as instruções do programa.
- Não usar a instrução *assembly* DA.
- Explique como o método da correção posterior de seis é implementado em seu programa.
- Quais casos de teste serão aplicados? Mostre que estes casos cobrem todas as situações possíveis.

x) Agora modifique o programa anterior para transformá-lo em uma subrotina que recebe os valores pela pilha. A subrotina deve retornar o resultado também pela pilha.

Recomendação:

- Acrescente um diagrama indicando o conteúdo da pilha e dos registradores em cada momento da execução do programa.

Perguntas:

1. Qual é a função das macro-instruções ORG e END?
2. Qual a diferença entre os acessos à memória interna e externa do 8051? Qual é a relação com as instruções MOV e MOVX?
3. Como poderia ser codificado um programa que calcula a multiplicação entre dois números?
4. Um programa pode ser compilado para ser armazenado a partir do endereço 9000H no *kit*? Explique.

3. BIBLIOGRAFIA

1. HIRAKAWA, A; CUGNASCA, C. Laboratório de Microprocessadores, Apostila da experiência "**Familiarização com a placa experimental de microcontrolador 8051**", 1998.
2. MATSUNAGA, A., TSGAWA, M. **Sistema de Pesagem Dinâmica**, Projeto de formatura, EPUSP, 1997.
3. SILVA, V. **Microcontrolador 8051**, São Paulo: Érica, 1990.
4. SINHA, P. K. **Microprocessors for Engineers – Interfacing for Real Time Applications**, Ellis Horwood Limitted, 1987.
5. Intel – Embedded Microcontrollers, 1995.
6. Intel – Embedded Applications V.2 1995/1996.
7. 2500AD Software, 8044/51 Cross Assembler for MSDOS.
8. Instruções e comandos do programa XTALK.
9. Intel Home Page ApBUILDER - http://developer.intel.com/design/builder/apbldr/

4. EQUIPAMENTOS NECESSÁRIOS

- 1 placa experimental de microcontrolador 8051, com cabo de conexão serial.
- 1 fonte de alimentação variável de +12Vcc.
- 1 computador PC com interface serial.

5. PROGRAMAS NECESSÁRIOS

- *Cross Assembler* do 8051 (X8051).
- *Linker* do 8051 (LINK2).
- Programa de comunicação com a placa experimental (XTALK).