Noções básicas de programação para sistemas microprocessados.

Os sistemas microprocessados tem 2 partes básicas: o software e o hardware.

O hardware é a parte física do sistema, envolvendo os circuitos integrados, fios, chaves, displays e demais componentes. Dentre estes componentes os que estão sempre presentes são o microprocessador e a memória ou a junção destes num único circuito integrado, chamado de microcontrolador. Normalmente o microcontrolador também tem outros dispositivos internos como temporizadores e sistemas de entrada e saída, com a finalidade de gerar sistemas mais compactos e baratos.

O software é um conjunto de instruções colocadas de forma ordenada com a finalidade de realizar uma tarefa, esta seqüência é conhecida como programa. Este programa indica ao processador qual ação ele deve realizar. O programa está vinculado ao conjunto de instruções que o microcontrolador pode realizar, assim, cada microcontrolador tem um conjunto de instruções específico e próprio, desta forma um programa para o microcontrolador 8051 não funcionará adequadamente no microprocessador PIC.

O microcontrolador para poder executar o programa necessita que o mesmo esteja armazenado na memória, e como os dados na memória são informações binárias, esta seqüência é chamada de programa em linguagem de máquina. Abaixo é mostrado um programa em linguagem de máquina para que o 8051 ligue um led na sua porta de saída 1, mostrado em hexadecimal:

```
02 00 03 C2 AF 75 81 2F 75 D0 00 74 19 79 B3 75 E2 90 F0 D2 90
```

Notamos que para nós este programa em linguagem de máquina é incompreensível, sendo pouco útil desta forma quando visamos o nosso entendimento. Para facilitar o uso dos processadores o programa é escrito em linguagens próximas a linguagem humana e depois é traduzido para linguagem de máquina. A linguagem quanto mais próxima da linguagem humana maior o seu nível de compreensão e mais difícil é o papel do programa de tradução, por este motivo, normalmente os programas traduzidos de linguagens de alto nível geram grandes programas em linguagem de máquina. Por outro lado, linguagens de baixo nível têm fácil tradução para linguagem de máquina gerando programas executáveis em linguagem de máquina de pequeno tamanho. A programação neste tipo de linguagem é considerado difícil pois cada ação da máquina é muito simples, o controle do processador no entanto é total.

Vamos trabalhar em um nível acima da linguagem de máquina, o chamado ***assembler****. O assembler é a tradução direta das instruções que podem ser executadas pelo processador com uma nomenclatura um pouco mais compreensível e mais algumas pseudo instruções para orientar ao programa tradutor ou facilitar a compreensão do programa. Cada processador então tem o seu assembler próprio, uma vez que cada um deles tem características diferentes, sendo que eles podem agrupados em famílias de processadores semelhantes e que utilizam o mesmo conjunto básico de instruções.*

Pelo exposto para que um programa seja executado num sistema microprocessado nós devemos: escrever o programa em assembler ➔ traduzir o programa para linguagem de máquina ➔ transferir o programa para a memória e executar o programa.

A escrita do programa em assembler pode ser feita em qualquer editor de textos, tipo bloco de notas ou wordpad do Windows. O arquivo gerado normalmente deve ser no padrão

*texto (ASCII) e deve ser armazenado ter um nome com uma extensão específica (**.ASM** no nosso caso).*

A tradução do programa pode ser de dois tipos: interpretação ou compilação.

O interpretador gera o código em linguagem de máquina e já envia para que o processador possa executá-lo em seguida, cada linha do programa é então traduzido e executado.

O compilador gera um arquivo com os códigos em linguagem de máquina que pode ser transferido para a memória e executado no momento que for mais adequado. Este será o método por nós empregado durante nossas aulas.

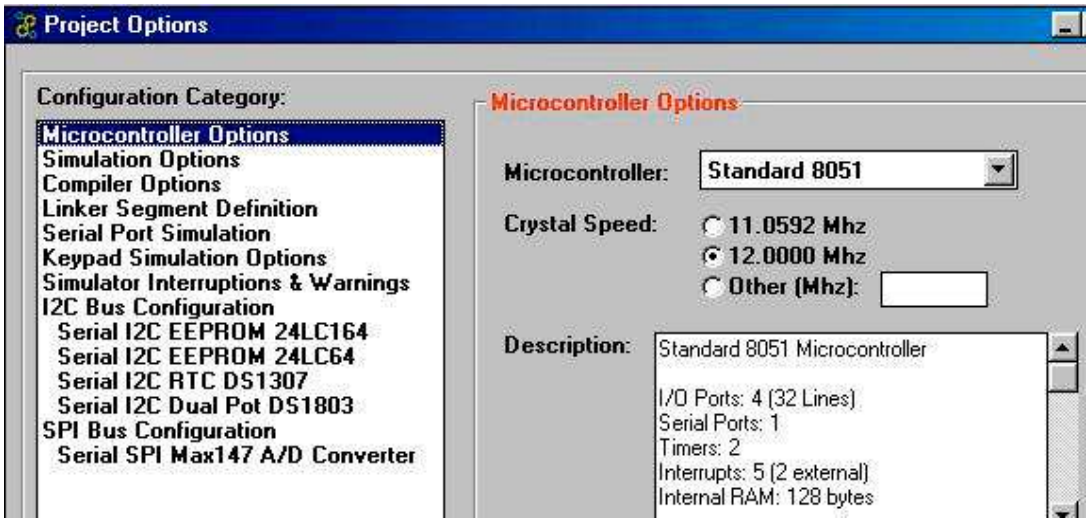
A transferência e execução são os últimos passos do processo, normalmente cada sistema / kit de processador tem um método específico, normalmente baseado no padrão original adotado pelo fabricante do processador.

Para fins didáticos e para facilidade em corrigir erros (= bug) de programas (retirar os bug's = debugar), normalmente utiliza-se um simulador do processador, que muitas vezes tem alguns destes processos já embutidos, facilitando o trabalho.

Vamos utilizar o Pinnacle 52 da Vault Information Services (VIS) que é um programa para o Windows e que pode ser utilizado livremente. Ele pode ser baixado no endereço: http://www.vaultbbs.com/pinnacle

Escrevendo e compilando um programa.

Uma vez instalado e inicializado o Pinnacle (clique Iniciar ➔ Programas ➔ Pinnacle ➔ Pinnacle 52), vamos começar um novo projeto clicando em Project ➔ New uma janela para dar nome e características do projeto é aberta e pode ser completada com por exemplo: nome = aula1. O passo seguinte é configurar o projeto, isto pode ser feito clicando em Project ➔ Project Options. A janela abaixo é aberta e pode ser configurada conforme abaixo.



Atente principalmente neste momento para os itens microcontroller em standart 8051 e crystal speed em 12.0000 MHz. Finalize esta configuração clicando em OK.

Para digitar o programa em assembler deve-se clicar em File ➔ New, e um editor de textos apropriado para digitação de programas em assembler para o microcontrolador 8051 é aberto. Digite o programa abaixo:

```
$PAGINATE TITLE (EXERCICIO PROGRAMA 1.1)
$SUBTITLE      (BLOCO UNICO 8031 AVMAC)
;
;EXERCICIO PROGRAMA 1.1
;
;PROGRAMA QUE ACENDE APENAS O LED1 LIGADO A PORTA P1 DO KIT
;DIDATICO.
;
          LED1          EQU    P1.0
          ORG           0000
          LJMP          MAIN
;INICIO DO PROGRAMA PRINCIPAL
MAIN:     CLR   EA              ;DESABILITA TODAS AS INTERRUPCOES
          MOV   SP,#2FH         ;DEFINE ENDERECO INICIAL DA PILHA
          MOV   PSW,#00H        ;HABILITA BANCO 0 DE REGISTRADORES
          MOV   A,#25           ;COLOCA O VALOR 25 DECIMAL NO
                                ;ACUMULADOR
          MOV   R1,#10110011B
                                ;COLOCA UM VALOR EM BINÁRIO NO REG. R1
          MOV   P1,#0F0H        ;DESLIGA LED1,LED2,LED3 E LED4
          SET   LED1            ;LIGA LED1

          END                   ;FIM DO PROGRAMA PRINCIPAL
```

Note que:

- *As instruções vão mudando de cor automaticamente conforme o texto vai sendo digitado.*
- *O sinal de ponto e vírgula indica um comentário que não será traduzido pelo compilador.*

Salve o arquivo digitado em File ➔ Save As, sendo que a extensão deve ser .ASM e o local do arquivo deve ser preferencialmente o diretório C:\C51. Este arquivo deve ser incluído no projeto clicando-se Project ➔ Edit Project. Uma janela será aberta e clicando-se em Include indica-se o arquivo recém digitado. Clicando-se Project ➔ Compile & Link File (ou somente Ctrl + F2) o programa gera outros arquivos com as extensões .OBJ, .MAP, .HEX e .PRN. Os mais importantes são os .HEX que é o arquivo em hexadecimal padrão INTEL que será utilizado para transferir o programa para o KIT didático e o .PRN que é um arquivo típico para documentação e apontar eventuais erros de compilação.

O Pinnacle deve ter mostrado como resposta a compilação uma tela com o seguinte texto:

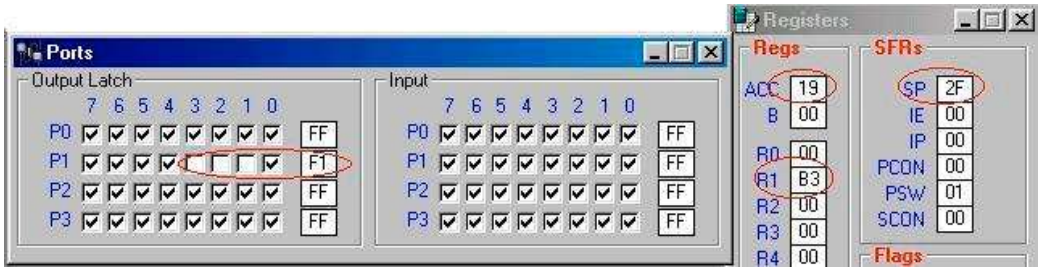
```
Initializing Compiler...
** Not registered -- assembled and linked programs limited to 2k.
Compiling C:\C51\PROG1_1.ASM
C:\C51\PROG1_1.ASM(26): Error [E2012]: Syntax error: SET
Build terminated. 1 error(s), 0 warning(s)
```

A linha com a inscrição: C:\C51\PROG1_1.ASM(26): Error [E2012]: Syntax error: SET mostra que na linha 26 (no canto inferior direito da janela do editor há a indicação de linha) ocorreu um erro de sintaxe, ou seja, a instrução não foi corretamente digitada. Feche esta janela e corrija a linha 26 a instrução para SETB*. Realize novamente a compilação do programa*

*e note que a última linha do aviso gerado pelo Pinnacle aponta que o programa não tem erros e que o arquivo tipo .**HEX** foi criado com exito.*

```
Initializing Compiler...
** Not registered -- assembled and linked programs limited to 2k.
Compiling C:\C51\PROG1_1.ASM
Linking...
Outfile created: C:\C51\PROG1_1.HEX (20 bytes)
Build complete. 0 error(s), 0 warning(s)
```

Para visualisar o programa sendo executado de forma simulada, é preciso deixar algumas janelas a mostra: a de registradores e a de portas. Selecione as opções Registers e Ports clicando em View. Inicialmente os registradores estão zerados e os bits das portas estão selecionados, indicando o nível lógico 1. O programa pode ser executado de uma única vez ou passo a passo. Utilize a tecla de função F8 para que o programa seja executado passo a passo. Verifique que aos poucos os valores dos registradores vão mudando, até chegar no mostrado abaixo (as mudanças são mostradas em vermelho):



Transferindo o programa.

Com o kit ligado ao computador pela porta de comunicação serial COM1, o programa pode ser transferido utilizando-se o programa RUN31, que é executado em ambiente DOS. Abra o prompt do DOS em Iniciar ➔ Programas ➔ Prompt MS-DOS e mude o diretório com o comando: CD C:\C51. Faça a transferência digitando RUN31 NOME_DO_SEU_PROGRAMA.

A resposta em sua tela deve ser a mostrada abaixo:

```
C:\c51>run31 aula1
_______  Transferindo arquivo para o Kit31  QualiTech v1.0  _______
 Nome:AULA1.HEX
Tamanho:58
 I/O:3F8
Escrevendo...
C:\c51>
```

Se tudo ocorrer corretamente, o display de cristal líquido do kit deve indicar: Transferindo: NOME_DO_SEU_PROGRAMA e somente um dos 4 led's devem estar acesos.

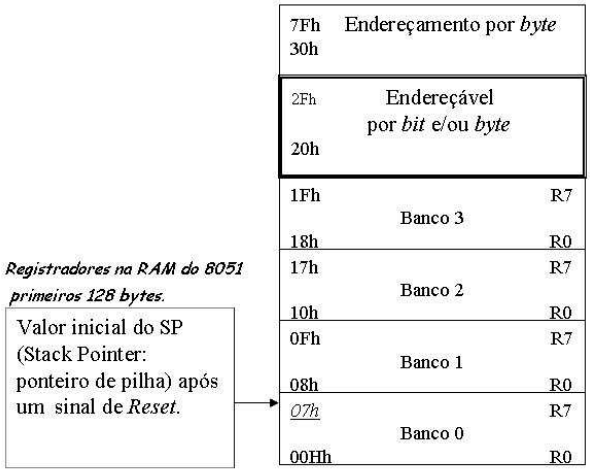
Caso não tenha notado o ocorrido, pressione a tecla reset do kit e realize novamente a transferência.

Registradores e instruções de movimentação.

Instruções de Movimentação

Nesta aula veremos também as instruções capazes de colocar e transferir valores nos registradores do 8051 e na sua memória associada.

O 8051 é um microcontrolador da Intel de 8 bits, derivado do microprocessador 8085 (de 8 bits). A maioria dos registradores do 8051 são de 8 bits com algumas exceções que podem armazenar até 16 bits. No 8051 existem 32 registradores de uso geral que ocupam os endereços de memória de 00 até 1Fh eles estão agrupados em 4 bancos com 8 registradores e os registradores são chamados de R0, R1, R2 ... R7.



*Ainda trabalhamos com outros dois registradores: o acumulador conhecido simplesmente como **A**, muito utilizado para as operações envolvendo a unidade lógico e aritmética e o registrador auxiliar **B** utilizado essencialmente em operações de multiplicação e divisão.*

Todos estes registradores podem ser trabalhados de modo geral, porém alguns têm características de programação específicas que serão vistas no decorrer do curso.

Instruções de movimentação

São usadas para atribuir conteúdo aos operadores. Cada instrução pode ser usada com diferentes modos de endereçamento. A próxima tabela tem as principais instruções de movimentação. Uma das principais instruções de movimentação é a instrução MOV:

INSTRUÇÃO MOV

Propósito: Transferência de dados entre células de memória, registradores e o acumulador.

Sintaxe:

*MOV **Destino**, **Fonte***

***Destino** é o lugar para onde o dado será movido e **Fonte** é o lugar onde o dado está.*

Os diferentes movimentos de dados permitidos para esta instrução são, por exemplo:

***Destino**: memória*	***Fonte**: acumulador*
***Destino**: acumulador*	***Fonte**: memória*

***Destino**: registrador*	***Fonte**: registrador*
***Destino**: registrador*	***Fonte**: memória*
***Destino**: memória*	***Fonte**: registrador*
***Destino**: registrador*	***Fonte**: dado imediato*
***Destino**: memória*	***Fonte**: dado imediato*

Exemplo:

```
MOV A, #06h     ;➔ coloca o valor 6 no registrador A ( acumulador )
MOV R0, A       ;➔ coloca o valor que está em A no registrador R0, o valor em
                ;➔ A não é alterado
MOV R1, 4Ch     ;➔ coloca o valor que está na memória de endereço 4Ch
                ;➔no  registrador R1
```

Este pequeno programa coloca o valor 06h para o acumulador A, então ele move o conteúdo de A (A = 06h) para o registrador R0, e finalmente move o valor que está armazenado na memória 4Ch para o registrador R1.

O modo como os dados são encontrados é chamado de modo de endereço. Como o conjunto de instruções da família 8051 foi otimizado para aplicação de controle em 8 bits ele tem uma variedade de rápidos modos de endereçamento para acessar a RAM interna, facilitando operações de byte em pequenas estruturas de dados para o 8051 temos 4 modos, mostrados na figura abaixo:

Modo de endereçamento	*Operandos (Registradores e Memória)*
Endereçamento Imediato	Memória de Programa
Por Registrador	R0-R7 e ACC (A), B, C (*carry-bit*) e DPTR
Direto	Os 128 *bytes* menos significativos da RAM Interna e Registradores de Funções Especiais
Indireto por Registrador	RAM interna (@R0, @R1 e SP) e Memória de Dados Externa (@R0, @R1 e @DPTR)
Registrador Base mais indireto indexado por registrador	Memória de Programa (@DPTR+A e @PC+A)

Modo de endereçamento DIRETO

Neste modo o operando é especificado na instrução por um campo de endereço de 8 bits. Somente a RAM de dados interna e os registros de função especial (primeiras 256 posições de memória) é que poderão ser endereçados diretamente.

```
Exemplo a : MOV A, 25h          ;A ← ( 25h )
Exemplo b : MOV 90h, A          ;Obs.: 90h = porta 1... ( 90h ) ← A
```

Modo de endereçamento de registro ou modo REGISTRADOR

Os bancos de registros, contendo de R0 a R7, serão acessados por certas instruções onde a especificação do registro será feita por três bits do próprio opcode (código de

operação). As instruções de acesso aos registros são eficientes, visto que nenhum endereço é necessário. Quando o instrução for executada, um dos oito registros do banco selecionado será acessado. Um dos quatro bancos de registro é selecionado pelos bits de seleção de bancos do registro PSW (bits RS1 e RS0).

Exemplo a : MOV R5, A ;R5← A

Exemplo b : MOV A, R0 ;A← R0

Modo de endereçamento INDIRETO

Neste modo a instrução especifica que o registro contém o endereço do operando. Tanto a memória interna quanto a externa poderão ser endereçadas indiretamente.

O registro de endereçamento usado para endereços de 8 bits deverá ser o registro R0 ou R1 do banco selecionado e para endereços de 16 bits será somente o registro apontador de dados (DPTR).

Exemplo a : MOV @ R1, 15h ;(R1) ← (15h)

Exemplo b : MOVX @ DPTR, A ;(DPTR) ← A

Obs.: @ é utilizado para indicar endereçamento indireto. Ou seja: @ = endereçado pelo conteúdo de ...

Modo de endereçamento de registros específicos ou ESPECÍFICO A REGISTRO

Algumas instruções referem-se a certos registros. Por exemplo algumas instruções operam o acumulador, o registro DPTR, etc., assim nenhum byte de endereço será necessário, o opcode já define qual o registro que será afetado.

Exemplo a : DA A ;Faz o ajuste decimal do acumulador

Exemplo b : CLR A ; Zera o acumulador: A ← 00h

Exemplo c : INC DPTR ; DPTR + 1← DPTR

Modo de endereçamento IMEDIATO ou CONSTANTE IMEDIATA

Neste modo de endereçamento o opcode é seguido de um valor de uma constante que será operada. Na linguagem assembly, este modo é indicado através do simbolo #.

Exemplo a : MOV B, #252 ;B ← FCh

Exemplo b : MOV A, #100 ;A ← 64h

Exemplo c : MOV DPTR, #05FEh ;DPTR ← 05FEh

Obs.: # indica valor constante. Quando após a constante aparecer um "H", o valor da constante é hexadecimal, quando tiver um "B" é binário, e quando a letra for omitida ou aparecer um "D" o valor será decimal.

Modo de endereçamento INDEXADO

Somente a memória de programa (ROM) poderá ser acessada com endereçamento indexado e somente poderá ser lida.

O endereço efetivo é a soma do acumulador e um registro de 16 bits (DPTR ou PC).

Este modo é usado para leituras de tabelas colocadas na memória de programa (ROM). Por exemplo, tabelas de conversão, ou de mensagens.

Um registro base de 16 bits, tal como o registro DPTR, ou contador de programa (PC), aponta para a base da tabela e o acumulador recebe o deslocamento dentro da tabela. Assim o endereço de entrada da tabela será formado com a soma do conteúdo do acumulador e o registro base.

Exemplo a : MOVC A, @ A + DPTR ;A ← (A+DPTR) da ROM

Exemplo b : MOVC A, @ A + PC ;A ← (A + PC) da ROM

Tabela resumo de instruções de movimentação:

		Modos de Endereçamento				
Mnemônico	Operação	Dir.	Ind.	Reg.	Imed.	Tempo de Exec. (μs)
MOV A,<*src*>	(A) ← <*src*>	X	X	X	X	1
MOV <*dest*>,A	<*dest*> ← (A)	X	X	X		1
MOV <*dest*>,<*src*>	<*dest*> ← <*src*>	X	X	X	X	2
MOV DPTR,#$data_{16}$	(DPTR) ← #$data_{16}$				X	2
PUSH <*src*>	(SP) ← (SP)+1, ((SP)) ← <*src*>	X				2
POP <*dest*>	<*dest*> ← ((SP)), (SP) ← (SP)-1	X				2
XCH A,<*byte*>	(A) ↔ <*byte*>	X	X	X		1
XCHD A,@Ri	$(A)_{3,0}$ ↔ $((Ri))_{3,0}$		X			1

Operações de transferência de dados da RAM interna.

Mnemônico	Operação	Largura do endereçamento	Tempo de Exec. (μs)
MOVX A,@Ri	(A) ← ((Ri))	8 bits	2
MOVX @Ri,A	((Ri)) ← (A)	8 bits	2
MOVX A,@DPTR	(A) ← ((DPTR))	16 bits	2
MOVX @DPTR,A	((DPTR)) ← (A)	16 bits	2

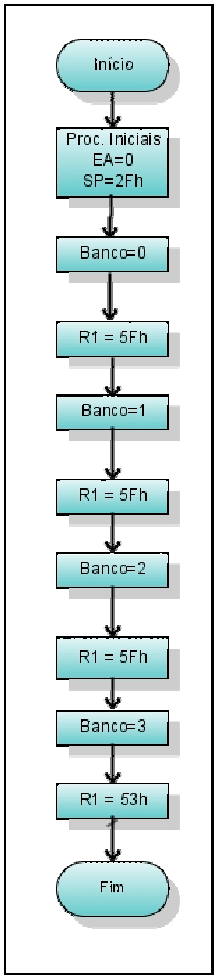
Operações de transferência de dados da RAM externa.

Mnemônico	Operação	Tempo de Exec. (μs)
MOVC A,@A+DPTR	(A) ← ((A) + (DPTR))	2
MOVC A,@A+PC	(A) ← ((A) + (PC))	2

Operações de transferência de dados da ROM.

Programas exercício.

1 – Escreva um programa em assembler para colocar o valor 5Fh nos registradores R1 de todos os bancos.



O Fluxograma para este programa é mostrado na figura ao lado.

A mudança entre os bancos de dados é executada pela alteração dos bits RS1 e RS0 como se segue:

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

Assim, para que os valores sejam colocados no banco de registradores 2, o bit RS1 deve ser setado e o bit RS0 deve ser resetado. A instrução para setar um bit é SETB, para resetar é CLR. A seqüência de instruções para acionar o banco 2 é:

SETB RS1
CLR RS0

Seguindo o fluxograma ao lado, podemos escrever todo o programa:

```
ORG     0000

CLR     EA              ;DESABILITA TODAS AS INTERRUPCOES
MOV     SP,#2FH         ;DEFINE ENDERECO INICIAL DA PILHA
CLR     RS1             ;HABILITA BANCO 0 DE REGISTRADORES
CLR     RS0
MOV     R1,#5Fh

CLR     RS1             ;HABILITA BANCO 1 DE REGISTRADORES
SETB    RS0
MOV     R1,#5Fh

SETB    RS1             ;HABILITA BANCO 2 DE REGISTRADORES
CLR     RS0
MOV     R1,#5Fh

SETB    RS1             ;HABILITA BANCO 3 DE REGISTRADORES
SETB    RS0
MOV     R1,#5Fh
END                     ;FIM DO PROGRAMA PRINCIPAL
```

Digite o programa no pinnacle e simule-o com as janelas Registers e Internal RAM abertas (clique em View e selecione as janelas).