EEL7030 Microprocessadores – Roteiro 2
Prof. Raimes Moraes

Compilação de Código Assembly e C

Como pôde ser visto na aula anterior, a inserção direta de código na memória do simulador do 8051 é trabalhosa, principalmente devido à necessidade de se consultar o manual do 8051 para obter o código binário que o processador decodifica. Existem ferramentas de software para simplificar tal tarefa.

É possível criar arquivo texto (arquivo fonte) utilizando os mnemônicos do 8051 e diretivas para orientar o compilador (Apêndices A e B da apostila). Compiladores têm a tarefa de proceder a interpretação do arquivo fonte e gerar arquivo em outro formato que contém informações e o código a ser gravado na memória de programa. Um dos formatos possíveis para tal arquivo tem extensão .hex. Há códigos (*bootloaders*) que realizam a transferência do arquivo .hex para a memória de programa dos microcontroladores. Estes programas atuam, geralmente, em conjunto com circuitos eletrônicos que permitem a gravação do código na memória de programa (OTPROM, EPROM, EEPROM).

O arquivo .hex pode ser aberto no *notepad* possuindo formato similar ao apresentado na Tabela 1.

Tabela 1 – Exemplo de conteúdo de arquivo hex resultante da compilação de arquivo fonte em Assembly.

```
:10200000 3180203E1B30211020FB3A1020C30A20D3
:01201000 00CF
:03 20CE 00 34FBC917
:00000001FF
```

Cada campo da primeira linha do arquivo .hex foi destacado com diferentes cores para permitir que se faça a sua correspondência com a Tabela 2.

Tabela 2 – Função de cada campo de linha do arquivo hex resultante da compilação de arquivo fonte em Assembly.

Campo		Nro. De Nibbles	Descrição (Valores hexadecimais)
1	**Caractere inicial**	**2**	**":"** (ASCII = 0x3A)
2	**Nro de Bytes**	**2**	**Nro de bytes no campo de dados**
3	**Endereço**	**4**	**Endereço para carregamento dos dados na memória**
4	**Tipo**	**2**	**00 (**Linha de dados**) ou 01 (**Linha de encerramento)
5	**Dados**	**2**	**0 a n bytes de código ou de dados. n é geralmente menor ou igual a 32 (20H).**
6	**Checksum**	**2**	**Byte menos significativo (LSB) do complemento de dois da soma dos valores representados pelos bytes da linha, excetuando o caractere inicial e o próprio checksum.**

A partir da Tabela 3, tem-se que a linha de encerramento do arquivo .hex é composta pelos seguintes caracteres:

:	Caractere inicial
00	Número de dados.
00	Campo de endereços: MSB
00	Campo de endereços: LSB
01	Linha de encerramento.
FF	Checksum da linha de encerramento.

Crie projeto no compilador Keil que contenha arquivo texto (extensão .a51 com o conteúdo abaixo.

```
; Programa      FIRST.a51
                MOV   02H,#3
VOLTA:          MOV   A,R2
                ADD   A,32H
                MOV   32H,A
                JMP   VOLTA
                END
```

Para gerar arquivo hex, pressione Alt+F7; na janela Output, selecione ‘Create HEX File’ (Figura 1). Compile o código e abra o arquivo .hex gerado para observar o formato. Simule o programa no ambiente Keil.

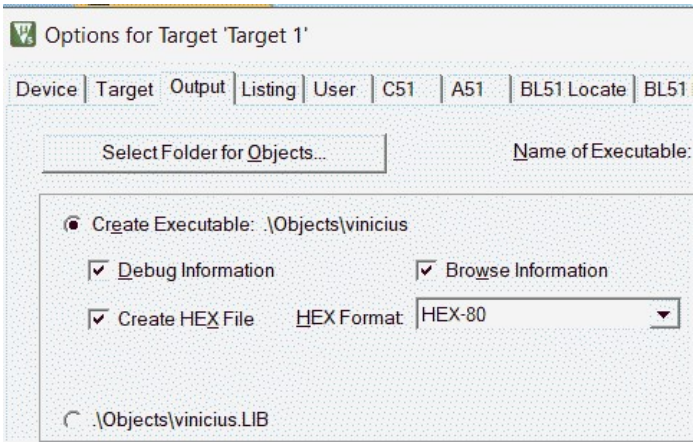

Figura 1 - Janela do compilador Keil na qual é solicitada geração de arquivo em formato hex.

Cada família de microprocessadores tem seu conjunto próprio de mnemônicos que correspondem ao conjunto de códigos binários que são decodificados e executados por estas famílias. Para permitir que o programador possa escrever código para diferentes famílias sem a necessidade de aprender os vários conjuntos de mnemônicos (aumentando assim, sua produtividade), existem as denominadas linguagens de mais alto nível (tal como a linguagem C). Estas linguagens podem ser usadas para programar diferentes famílias de microcontroladores/microprocessadores. O compilador tem a função de gerar o código binário para cada família a partir do código escrito em linguagem C, por exemplo.

Tendo compreendido o procedimento de criação de projetos no sistema de desenvolvimento da Keil, crie um novo projeto com o programa abaixo, salve-o com extensão .c, compile-o e simule-o no Keil. Para programa escrito em liguagem C, adicione o arquivo startup.a51. O propósito do código é mostrar o dígito 1 no display 0 no circuito simulado pelo software EDSIM51 (Figura 2 - http://www.edsim51.com). Para permitir a escrita no *display*0 do EDSIM51, deve-se fazer para P0.7=1, P3.4=P3.2=0. A função **converte_7seg** recebe como parâmetro, o valor **1** e retorna o valor a ser escrito no *display* de 7 segmentos (através da porta P1) para mostrar o dígito. Compile o programa no

Keil, gere .HEX (pressione Alt+F7; selecione o tab ***Output*** e selecione Create HEX file) e o carregue no EDSIM51 para observar a sua execução.

```
// diretiva de compilação
#include <reg51.h> // inclusão de arquivo com endereços de registradores do 8051

// declaração dos protótipos de funções
unsigned char converte_7seg (unsigned char);

// declaração de constantes
#define CS  0x80;

// declaração de variáveis globais
unsigned char dsp_0 = 0xE7;                    // 11100111 B

// função main
void main (void)
{
        unsigned char digito = 1 ;

        P0 = CS;
        P3 = dsp_0;
        P1 = converte_7seg(digito);
        while(1);
 }

// função converte_7seg
// retorna valor a ser enviado aos displays para formar o caractere

unsigned char converte_7seg (unsigned char dado)
{

switch (dado)                                          // GFEDCBA
        {       case 0:   dado  = 0x40; break;   // "1000000";
                case 1:   dado  = 0x79; break;   // "1111001";
                case 2:   dado  = 0x24; break;   // "0100100";
                case 3:   dado  = 0x30; break;   // "0110000";
                case 4:   dado  = 0x19; break;   // "0011001";
                case 5:   dado  = 0x12; break;   // "0010010";
                case 6:   dado  = 0x02; break;   // "0000010";
                case 7:   dado  = 0x78; break;   // "1111000";
                case 8:   dado  = 0x00; break;   // "0000000";
                case 9:   dado  = 0x10; break;   // "0010000";
                case 10:  dado  = 0x08; break;   // "0001000";
                case 11:  dado  = 0x03; break;   // "0000011";
                case 12:  dado  = 0x46; break;   // "1000110";
                case 13:  dado  = 0x21; break;   // "0100001";
                case 14:  dado  = 0x06; break;   // "0000110";
                case 15:  dado  = 0x0E; break;   // "0001110";
                default:    dado  = 0x80;              // "0000000";
        }
  return dado;

    } // end converte_7seg
```

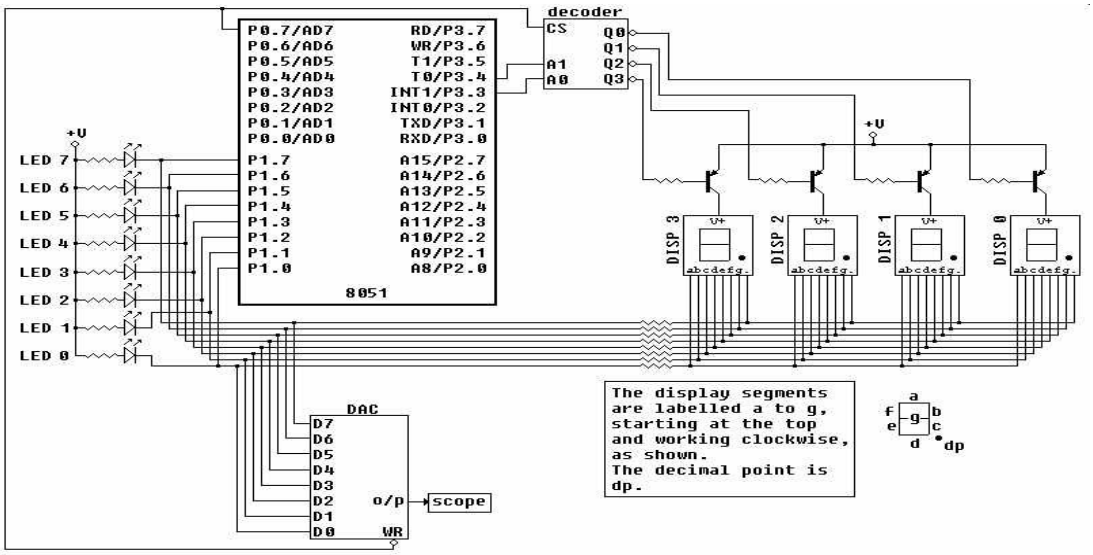

Figura 2 - Esquemático de circuito simulado pelo EDSIM51 que contém 4 displays, decodificador e 8051.

Exercícios:

1) Modifique o programa anterior para mostrar uma contagem decrescente (9 a 0) no *display* 3. OBS: Insira atraso entre rotações para visualizar cada digito (contagem de 15000 em *loop* de *for*).

2) Modifique o programa anterior para mostrar valor lido das chaves (0 a FH) conectadas à porta P2 (*Switch Bank* – http://www.edsim51.com/simInstructions.html#switches) no display 0. Valor lido de chave aberta equivale a '1'; de chave fechada, '0'. Portanto, complemente o valor lido de P2 antes de apresentá-lo à função de conversão. Mascarar valor lido referente às 4 chaves mais significativas para evitar influência das mesmas.

3) Modifique o programa anterior para mostrar no *display* 1, o nro de chaves fechadas.

4) No EDSIM51, há também leds conectados à Porta P1 (LED *Bank* - http://www.edsim51.com/simInstructions.html#leds). Faça um programa que rotacione um led aceso da esquerda para a direita

inserindo atraso entre rotações (contagem em *loop* de *for*). Fazer que o procedimento seja cíclico (repita-se). . OBS: Inibir CS do *display*.

5) Faça um programa que acenda todos os 8 leds, um a um (da esquerda para a direita) com inserção de atraso; quando todos os leds estiverem acesos, apague-os um a um (da direita para a esquerda). Fazer que o procedimento seja cíclico.

6) Faça um programa que teste a chave conectada a P2.7. Se a mesma estiver fechada (P2.7=’0’), rotacione um led aceso para a esquerda (se P2.6=’1’) ou para a direita (se P2.6=’0’) pelo número de vezes especificado pelo complemento do valor das 4 chaves menos significativas (P2.3 a P2.0). Se a chave P2.7 estiver aberta, fique aguardando ser alterada. Depois que a chave conectada a P2.7 for fechada, faça o solicitado e aguarde a mesma retornar para nível lógico baixo.