UFPEL

Microcontroladores

Aula 11 – Programação em C para o 8051: introdução

Prof. Dr. Alan Carlos Junior Rossetto

Introdução

- A utilização de uma linguagem de médio ou alto nível para a programação de um µC propicia versatilidade ao programador, i.e.:
  - Permite **abstrair** alguns aspectos do programa;
  - Evita lidar **diretamente** com um conjunto reduzido de instruções;
  - Etc.
- Uma linguagem de programação com esta característica e largamente utilizada para a programação de µCs é a **linguagem C**;
- Um código gerado em linguagem C é tipicamente **maior** e **menos otimizado** que um código em assembly nativo. Por outro lado, há um ganho significativo na **facilidade** de programação;
- A "tradução" dos comandos da linguagem C em instruções assembly é feito pelo **compilador**;
- A compilação de um **sistema embarcado** traz consigo algumas peculiaridades e é ligeiramente distinta da programação para *desktop*, por exemplo.

Introdução

- Além de lidar com código geral, o **compilador** cujo programa é voltado a um uC precisa:
  - Gerenciar o espaço e a alocação dos dados;
  - Alocar funções para interrupções;
  - Definir variáveis com endereços específicos, permitindo o acesso a periféricos e / ou SFRs;
  - Etc.
- Eventualmente o compilador precisa oferecer também uma extensão, i.e., comandos/diretivas fora do padrão ANSI-C, por exemplo;
- Um compilador largamente utilizado para este propósito é o *Small Device C Compiler*, ou SDCC;

Compilador SDCC

- O SDCC é um compilador de **domínio público** voltado à programação de **microcontroladores** de 8 *bits*;
- Como de todo compilador, sua função é **traduzir** o código criado em C para uma linguagem de montagem suportada pelo processador e fornecida pelo fabricante, observando as **características** particulares dos dispositivos;
- Dentre as famílias suportadas por ele, temos a família MCS-51, Zilog Z80 e PIC, por exemplo;
- Existem ainda compiladores proprietários, como o mikroC e o C51 (Keil), por exemplo. Estes ainda podem se apresentar **incorporados** à uma IDE.

Linguagem C

- A linguagem C é uma linguagem de programação de **finalidade geral** criada orginalmente nos anos 70 para desenvolver o sistema operacional Unix;
- Faz parte das linguagens denominadas **imperativas**, nas quais a computação pode ser vista como uma sequência de **instruções** que manipulam valores de **variáveis**;
- É uma linguagem **estruturada**, i.e., todos os possíveis programas podem ser reduzidos a apenas três estruturas: sequência, decisão e iteração;
- Combina diferentes **níveis** de abstração, permitindo a manipulação direta de *bits*, *bytes* e endereços de memória;
- Possui fluxos de controle e estruturas de dados típicos da maioria das linguagens imperativas:
  - Agrupamento de comandos;
  - Tomadas de decisão;
  - Laços de repetição.

Tipos de dados

- Toda a variável usada no programa precisa ser **declarada**, e por ocasião da declaração, ela deve assumir um determinado **tipo**;
- Para µCs de 8 *bits*, os **tipos de dados** tipicamente disponíveis são:

Tipo de dado	*Bits*	*Bytes*	Alcance
`bit`	1	-	0 a 1
`signed char`	8	1	-128 a +127
`unsigned char`	8	1	0 a 255
`signed int / short`	16	2	-32768 a +32767
`unsigned int / short`	16	2	0 a 65535
`signed long`	32	4	-2147483648 a +2147483648
`unsigned long`	32	4	0 a 4294967295
`float`	32	4	±1,175494E-38 a ±3,402823E+38

- Note que para µCs com mais *bits* (16, 32, ...), esta tabela pode mudar, além de existirem tipos de dados adicionais.

Alocação de variáveis

- Além do tipo de dado armazenado por uma variável, esta também pode ser definida em diferentes tipos no que tange o seu **comportamento** dentro do programa;
- São elas:
  - Variáveis **automáticas** (`auto`): são declaradas dentro de uma função e existem **somente** durante a execução desta, permitindo o reuso da posição de memória antes ocupada;
  - Variáveis **estáticas** (`static`): podem ser declaradas dentro ou fora de funções e existem durante **todo o ciclo** de execução do programa, sendo inicializadas com zero ou com o valor definido no programa;
  - Variáveis **constantes** (`const`): é tipicamente declarada como global e o valor é uma constante, devendo ser alocada em memória de programa, poupando memória de dados;
  - Variáveis **voláteis** (`volatile`): o valor deste tipo de variável pode mudar por **ações externas** ao programa (PSW, periférico, etc).

Alocação de variáveis

- Além do tipo de dado armazenado por uma variável, esta também pode ser definida em diferentes tipos no que tange o seu **comportamento** dentro do programa;
- Exemplos:

```
const float pi = 3.141592653;
...
static unsigned int COUNTS;
...
volatile unsigned char VAR;
```

Alocação de variáveis

- Por ocasião da declaração, é possível também escolher a **porção de memória** na qual uma determinada variável vai ser armazenada, a saber:
  - **`data`**: região de memória RAM interna dos SFRs;
  - **`idata`**: região de memória RAM interna acessada de forma indireta;
  - **`xdata`**: região de memória RAM externa;
  - **`code`**: região de memória de programa (ROM);
  - **`bit`**: um único bit na região de memória RAM interna desde **`20h`** a **`2Fh`**.

Alocação de variáveis

- Exemplos:

```
data unsigned char var;
...
code const float pi;
...
xdata at 0x8000 int var; // É possível definir também a
                         // posição de memória a ser usada.
```

- **Atenção:** ao utilizar o compilador C51 do Keil, a porção de memória a ser utilizada deve ser inserida **depois** do tipo de variável, e.g.:

```
unsigned long xdata array[100];
...
float idata x,y,z;
...
unsigned int data dimension;
...
unsigned char xdata vector[10][4][4];
```

- Mais informações em: https://www.keil.com/support/man/docs/c51/c51_le_memtypes.htm

Palavras reservadas

- Algumas palavras **não podem** ser usadas para nomear variáveis ou declarar constantes. Tais restrições advém do **compilador**, pois para essas expressões são “interpretadas” no momento da compilação do programa. São elas:

Palavras reservadas da linguagem C – ANSI-C				
auto	do	goto	short	union
brake	double	if	signed	unsigned
case	else	int	sizeof	void
char	enum	long	static	volatile
const	extern	main	struct	while
continue	float	register	switch	
default	for	return	typedef	

Palavras reservadas adicionadas pelo SDCC – ANSI-C			
bit	interrupt	sfr	using

Registradores de função especial

- Além das palavras reservadas, os SFRs e seus *bits* (somente os acessíveis) também representam variáveis que não podem ser usadas para outros fins que não a sua funcionalidade particular;
- Os SFRs não precisam ser declarados no arquivo principal do programa, uma vez que são incluídos pela biblioteca / *header* (**`<reg52.h>`** ou **`<at89x52.h>`**, por exemplo).

Registradores de função especial (SFRs)	
`PCON`	*Power control*
`SCON`	*Serial control*
`TCON`	*Timer control*
`TMOD`	*Timer mode*
`SBUF`	*Serial buffer*
`IE`	*Interrupt enable*
`IP`	*Interrupt priority control*
`PSW`	*Program status word*
`ACC`	*Acumulator*

Registradores de função especial

- Além das palavras reservadas, os SFRs e seus *bits* (somente os acessíveis) também representam variáveis que não podem ser usadas para outros fins que não a sua funcionalidade particular;
- Os SFRs não precisam ser declarados no arquivo principal do programa, uma vez que são incluídos pela biblioteca / *header* (**`<reg52.h>`** ou **`<at89x52.h>`**, por exemplo).

Registradores de função especial (SFRs)	
`B`	*Register B*
`SP`	*Stack pointer*
`DPL`	*Data pointer low byte*
`DPH`	*Data pointer high byte*
`TL0`	*Timer / counter 0 low byte*
`TL1`	*Timer / counter 1 low byte*
`TH0`	*Timer / counter 0 high byte*
`TH1`	*Timer / counter 1 high byte*
`P0,...,P3`	*Ports P0 to P3*

Header `<at89x52.h>`

- O *header* é o arquivo que contém as **definições** dos registradores e outras **características** do µC que são imprescindíveis para o compilador, sendo adicionado pela diretiva **`#include`**;
- Fragmento do conteúdo do *header* **`<at89x52.h>`**:

```
sfr at 0x80 P0      sfr at 0x99 SBUF    /* P0 */
sfr at 0x81 SP      sfr at 0xA0 P2      sbit at 0x80 P0_0
sfr at 0x82 DPL     sfr at 0xA8 IE      sbit at 0x81 P0_1
sfr at 0x83 DPH     sfr at 0xB0 P3      sbit at 0x82 P0_2
sfr at 0x87 PCON    sfr at 0xB8 IP      sbit at 0x83 P0_3
sfr at 0x88 TCON    sfr at 0xC8 T2CON   sbit at 0x84 P0_4
sfr at 0x89 TMOD    sfr at 0xC9 T2MOD   sbit at 0x85 P0_5
sfr at 0x8A TL0     sfr at 0xCC TL2     sbit at 0x86 P0_6
sfr at 0x8B TL1     sfr at 0xCD TH2     sbit at 0x87 P0_7
sfr at 0x8C TH0     sfr at 0xD0 PSW     ...
sfr at 0x8D TH1     sfr at 0xE0 ACC
sfr at 0x90 P1      sfr at 0xE8 A
sfr at 0x98 SCON    sfr at 0xF0 B
...
```

Inclusão de *headers*

- Outras definições ou funcionalidades não disponíveis no *header* principal também podem ser incluídas pela diretiva **`#include`**;
- Estes podem ser arquivos padrões já existentes ou criados pelo usuário;
- Exemplo de *headers* disponíveis na biblioteca padrão do C51 (Keil):

```
ctype.h      // Funções para manipulação de caracteres.
float.h      // Limites para números em ponto flutuante.
math.h       // Funções matemáticas.
stdio.h      // Funções padrão de I/O.
string.h     // Funções para manipulação de strings.
...          // Lista completa em: ...\Keil\C51\INC
             // ou https://www.keil.com/support/man/docs/
             // c51/c51_lib_includes.htm
```

Estrutura de programa em C

- A estrutura típica de um programa em C para microcontrolador não difere da estrutura de típica de um programa para *desktop*, por exemplo;
- O programa conta tipicamente com:
  - Diretivas pré-processamento;
  - Declaração de variáveis globais;
  - Declaração de protótipos de funções;
  - Definições de funções;
  - Programa principal;
  - Declaração de variáveis locais.

Exemplo de programa em C

- A estrutura típica de um programa em C para o µC 8051 (versão AT89S51) é mostrada a seguir:

```
#include<at89x52.h> // Inclusão da biblioteca com as
                    // definições para o uC em questão.
#define LED P0_0    // Define uma variável como sendo um
                    // pino físico do uC.

int COUNTS;         // Variável global.

void delay(void);   // Protótipo de função.
...

void delay(void){   // Definição de função.
       ...
}

void main (void){   // Programa principal.
       int VAR;     // Variável local.
       ...
}
```

Exemplo prático

- Implemente a especificação da Tarefa 1 – Parte 1 em linguagem C.
  - Contador de 0 a 9 cíclico mostrado em display de 7 segmentos porém com codificação BCD e ligado à Porta P0.
  - Utilize a placa **V0.8**.

- Implemente a especificação da Tarefa 1 – Parte 2 em linguagem C
  - Utilize a placa **V0.8**.

Bibliografia e sugestão de leitura

- NICOLOSI, D. E. C.; BRONZERI, R. B. Microcontrolador 8051 com linguagem C prático e didático: família AT89S8252 Atmel. 2ª ed., São Paulo: Érica, 2009.
  - Livro [2] da bibliografia;
  - Capítulo 2: o compilador e a linguagem C para o µC 8051;
  - Capítulo 3: exemplos de programas para o µC 8051.