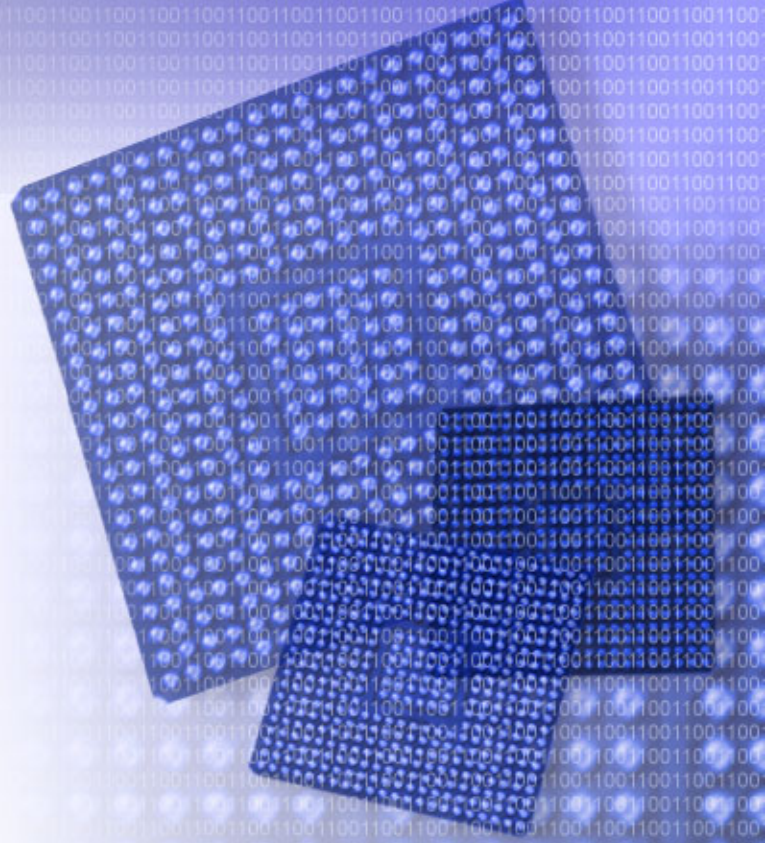
SEL–0629

Aplicação de Microprocessadores I

Aula 4
Programação em Linguagem C para PIC

Marcelo Andrade da Costa Vieira



Linguagem C

- Linguagem que pode ser utilizada atualmente na programação de quase todos os microcontroladores;
- Há microcontroladores com instruções otimizadas para programação em C;
- O compilador transforma as instruções em C no código em assembly;
- Há a possibilidade de colocar instruções em assembly no código em C;
- Geralmente o programa fica maior e menos eficiente em linguagem C, mas a facilidade na programação é muito maior;

Programação em C

Mikro C for PIC

Linguagem C

- A primeira função a ser executada é:

```
void main()
{
}
```

ela não recebe e não retorna nenhum parâmetro

- Comentários:

```
// comentário em uma linha
/* comentário em mais de
uma linha */
```

Linguagem C

- Há uma biblioteca para cada microcontrolador que especifica os registradores especiais (SFR)
- Para carregar um valor em um registrador, basta colocar o nome dele:

```
trisb = 0;
status = 0x0f;
latc = 255;
```

- Não é necessário especificar o banco do registrador!

Linguagem C

- Para carregar um valor em um bit, deve-se colocar o nome do registrador e o endereço do bit (f0, f1, f2...) ou o nome do bit, se for conhecido:

```
trisb.rb0 = 1;
status.c = 0;
status.f0 = 0;
```

- Ou, pode apenas colocar o nome do bit, seguido do termo <_*bit*>:

```
latb3_bit = 1;
c_bit = 0;
trisb2_bit = 0;
t0ie_bit = 1;
```

Linguagem C

- Representação numérica:

```
latb = 255; // decimal
latb = 0xff; // hexadecimal
latb = 0377; // octal
latb = 0b11111111; // binário
```

Linguagem C

- Variáveis:

Tipo	Tamanho em bytes	Intervalo
char	1	0 a 255
int	2	0 a 65535
float	4	-1.5E-45 a 3.4E+38
void	0	Nenhum valor

- Não há distinção entre variáveis com letra maiúscula ou minúscula, mas as funções devem ser em letra minúscula (void, if, while, for).

Linguagem C

- Modificadores de tipo:

Tipo	Tamanho em bytes	Intervalo
(unsigned) char	1	0 a 255
signed char	1	-128 a +127
(signed) short (int)	1	-128 a +127
unsigned short (int)	1	0 a 255
(signed) int	2	-32768 a +32767
unsigned int	2	0 a 65535
(signed) long int	4	-2.15E+9 a +2.15E+9
unsigned long int	4	0 a 4.3E+9
float	4	±1.18E-38 a ±6.8E+38
double	4	±1.18E-38 a ±6.8E+38
long double	4	±1.18E-38 a

Linguagem C

- Declaração de variáveis em endereço específico:

```
char i absolute 0x20
```

- Declaração de flags:

```
bit flagteste
```

Linguagem C

- Operadores aritméticos:

Operador	Ação
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão inteira
++	Incremento
- -	Decremento

Linguagem C

- Operadores bit a bit:

Operador	Ação
&	AND
|	OR
^	XOR
~	Complemento
>>	Deslocamento à direita
<<	Deslocamento à esquerda

Linguagem C

- Operadores relacionais:

Operador	Ação
==	Igual a
>	Maior que
<	Menor que
!=	Diferente de
>=	Maior ou igual a
<=	Menor ou igual a

Linguagem C

- Operadores relacionais booleanos:

Operador	Ação
&&	E
||	OU
!	NOT

Linguagem C

- Interrupção:

```
Void Interrupt_High() iv 0x0008 ics ICS_AUTO {

}
```

```
Void Interrupt_Low() iv 0x0018 ics ICS_AUTO {

}
```

Linguagem C

- Programa em Assembly:

```
asm{

}
```

Exemplo de programa em C no MikroC for PIC

```
/* Programa para acender um LED durante 0,5s (conectado na porta RB4) quando
 um botão (colocado na porta RA2) é pressionado. Para o PIC18F4550.*/

void tempo (long int a) // função para gastar tempo
{
long int i;
for (i = 0; i < a; i++);
}

void main()
{
 ADCON1 = 0b00001111;       // Configurando todas as portas do PIC como digitais
 ADCON1 = 0x0F;             // mesmo comando da instrução anterior

 trisa.trisa2 = 1;          //  Configurando a Porta RA2 como entrada
 trisa.f2 = 1;              //  mesmo comando da instrução anterior
 trisa2_bit = 1;            //  mesmo comando da instrução anterior
 trisa = 255;               // configurando todas as portas A como entrada
 trisa = 0b11111111;        //  mesmo comando da instrução anterior
 trisa = 0xFF;              //  mesmo comando da instrução anterior

 trisb.trisb4 = 0;          //  Configurando a Porta RB4 como saída
 trisb.f4 = 0;              //  mesmo comando da instrução anterior
 trisb4_bit = 0;            //  mesmo comando da instrução anterior
 trisb = 0;                 // configurando todas as portas B como saída
 trisb = 0b00000000;        //  mesmo comando da instrução anterior
 trisb = 0x00;              //  mesmo comando da instrução anterior

 latb4_bit = 0;             // inicialmente, apaga o led conectado na Porta RB4
 latb.latb4 = 0;            //  mesmo comando da instrução anterior
 latb.f4 = 0;               //  mesmo comando da instrução anterior
 latb = 0b00000000;        //  apaga o led da porta RB4 e coloca zero nas outras portas B

 while(1)
 {
  if (porta.ra2 == 0)       // testa se o botão está pressionado (=0)
  {                         // poderia ter usado a instrução: if (!porta.ra2)
   latb4_bit = 1;           // se pressionado, acende LED
   tempo(45000);            // espera um tempo (Aprox. 0,5s)
   latb4_bit = 0;           // apaga o led
  }
  else
  latb4_bit = 0;            // se botão não pressionado, mantém led apagado
 }
}
```

Em Assembly...

```
_tempo:

;Exemplo_Aula.c,5 ::            void tempo (long int a) // função para gastar tempo
;Exemplo_Aula.c,8 ::            for (i = 0; i < a; i++);
        CLRF        R1
        CLRF        R2
        CLRF        R3
        CLRF        R4
L_tempo0:
        MOVLW       128
        XORWF       R4, 0
        MOVWF       R0
        MOVLW       128
        XORWF       FARG_tempo_a+3, 0
        SUBWF       R0, 0
        BTFSS       STATUS+0, 2
        GOTO        L__tempo8
        MOVF        FARG_tempo_a+2, 0
        SUBWF       R3, 0
        BTFSS       STATUS+0, 2
        GOTO        L__tempo8
        MOVF        FARG_tempo_a+1, 0
        SUBWF       R2, 0
        BTFSS       STATUS+0, 2
        GOTO        L__tempo8
        MOVF        FARG_tempo_a+0, 0
        SUBWF       R1, 0
L__tempo8:
        BTFSC       STATUS+0, 0
        GOTO        L_tempo1
        MOVLW       1
        ADDWF       R1, 1
        MOVLW       0
        ADDWFC      R2, 1
        ADDWFC      R3, 1
        ADDWFC      R4, 1
        GOTO        L_tempo0
L_tempo1:
;Exemplo_Aula.c,9 ::            }
L_end_tempo:
        RETURN      0
; end of _tempo

_main:

;Exemplo_Aula.c,11 ::           void main()
;Exemplo_Aula.c,13 ::           ADCON1 = 0b00001111;      // Configurando todas as portas do PIC como digitais
        MOVLW       15
        MOVWF       ADCON1+0
;Exemplo_Aula.c,14 ::           ADCON1 = 0x0F;            // mesmo comando da instrução anterior
        MOVLW       15
        MOVWF       ADCON1+0
```

Configuration Bits

- São propriedades do PIC que são configuradas na gravação (hardware) e não no programa:
  - Alguns exemplos:
    - Watchdog Timer (On/Off)
    - Code Protect
    - Power On Reset
    - Brown Out Detect
    - Low Voltage Power
    - Data EEPROM Protect
    - Tipo de Oscilador (RC, LP, XT e HS)

Mikro C Pro for PIC

SimulIDE

FIM