



# 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 diretamente no bit de um registrador, deve-se colocar o nome do registrador seguido do nome ou o endereço do bit (f0, f1, f2...):

```
trisb.trisb2 = 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	$\pm 1.18E-38$ a $\pm 6.8E+38$
double	4	$\pm 1.18E-38$ a $\pm 6.8E+38$
void	0	nenhum

# 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 um certo tempo na porta RB4 quando
um botão colocado na porta RA2 é pressionado.

Para o PIC18F45k22

*/

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

void main()
{
ansela = 0;           // Configurando a Porta A como digital
anselb = 0;           // Configurando a Porta A como digital
trisa.f2 = 1;         // Configurando a Porta RA2 como entrada
trisa = 0b00000100; // mesmo comando da instrução anterior

trisb = 0;           // configurando todas as portas b como saída

latb.f4 = 0;          // inicialmente, apaga o led colocado na porta RB4
latb = 0b00000000;   // mesmo comando da instrução anterior

while(1)
{
if (porta.ra2 == 0) // testa se o botão está pressionado (=0)
{
// poderia ter usado a instrução: if (!porta.ra2)
latb.f4 = 1;        // se pressionado, acende LED
tempo(30000);       // espera um tempo
latb.f4 = 0;        // apaga o led
}
else
latb.f4 = 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)

**FIM**

