

Sistemas Microcontrolados

Linguagem C

Prof. Guilherme Peron

Introdução

- Qual o objetivo de fazer um programa utilizando a linguagem C ao invés de *assembly*?

Introdução

- Resposta: Facilitar a vida do programador.

Estrutura básica

- Um arquivo fonte escrito em linguagem C tem a seguinte estrutura básica:
 1. Documentação;
 2. Diretivas do pré-processador;
 3. Declarações globais;
 4. Subrotinas (incluindo o **main**).

Estrutura básica

- Exemplo:

```
/**1. Documentação
//Este programa mostra as seções de um programa em C
//Autor: Fulano de Tal e Cicrano Alguém
//Data: xx/xx/xx
// 2. Diretivas do Pré-processador
#include <stdint.h> //Definições do C99 → < > para lib do sistema
#include "teste.h" // " " para lib de usuário
// 3. Declarações globais
uint32_t varGlobal;
// 4. Subrotinas
// MAIN: Obrigatória
int main(void)
{
    while (1) //laço principal
    {

    }
}
```

Tipos de Dados

- Tipos de dados **inteiros**

C type	stdint.h type	Bits	Sign	Range
char	uint8_t	8	Unsigned	0 .. 255
signed char	int8_t	8	Signed	-128 .. 127
unsigned short	uint16_t	16	Unsigned	0 .. 65,535
short	int16_t	16	Signed	-32,768 .. 32,767
unsigned int	uint32_t	32	Unsigned	0 .. 4,294,967,295
int	int32_t	32	Signed	-2,147,483,648 .. 2,147,483,647
unsigned long long	uint64_t	64	Unsigned	0 .. 18,446,744,073,709,551,615
long long	int64_t	64	Signed	-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807

- Tipos de dados de **ponto flutuante**

C type	IEEE754 Name	Bits	Range
float	Single Precision	32	-3.4E38 .. 3.4E38
double	Double Precision	64	-1.7E308 .. 1.7E308

Representações Numéricas

```
uint32_t contador;  
contador = 123; //decimal  
contador = 0x7F //hexa
```

```
uint8_t texto[6] = "UTFPR"; //string  
uint8_t caractere = 'U'; //modo caractere
```

Variáveis

```
int32_t x1 = 4000;  
int32_t x2 = 5123;  
uint8_t y = 12;  
int64_t z;  
uint8_t texto = 'a';  
float pi = 3.14, resul;  
double xis;
```


Variáveis

- Ao adicionar a palavra **const** da declaração de uma variável, ela será definida na memória de código

```
const int32_t seculo = 21;
```

```
const char curso[] = "Microcontroladores";
```

Variáveis

- Variáveis **globais** são declaradas diretamente na memória RAM
- Variáveis **locais** são armazenadas em registradores e na pilha.

Operadores

Aritméticos:

Operador	Descrição	Exemplo
+	Soma dos argumentos	$a + b$
-	Subtração dos argumentos	$a - b$
*	Multiplicação dos argumentos	$a * b$
/	Divisão dos argumentos	a / b
%	Resto da divisão	$a \% b$
++	Soma 1 ao argumento ($a=a+1$)	$a++$
--	Subtrai 1 ao argumento ($a=a-1$)	$a--$

Relacionais:

Operador	Descrição
>	Maior que
<	Menor que
>=	Maior ou igual que
<=	Menor ou igual que
==	Igual
!=	Diferente

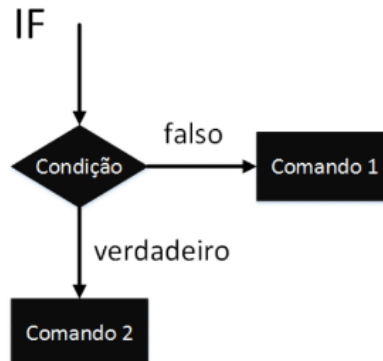
Bit a bit:

Operador	Descrição
&	Lógica E (AND)
	Lógica ou (OR)
^	Lógica OU-Exclusivo
~	Complemento (NOT)
>>	Deslocamento à direita
<<	Deslocamento à esquerda

Lógicos:

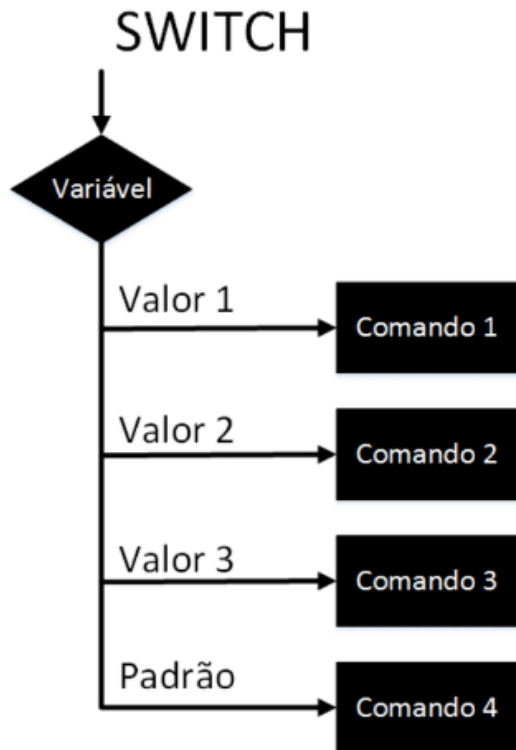
Operador	Descrição
&&	Lógica E (AND)
	Lógica OU (OR)
!	Complemento (NOT)

Controle de fluxo



```
char valor = GPIO_PORTB_AHB_DATA_R; // a variável valor recebe o valor da porta
if (valor == 0x0A) // se o valor lido for 0x0A
{
    GPIO_PORTJ_AHB_DATA_R = 0x1; // Executa se for verdadeiro
}
else
{
    GPIO_PORTJ_AHB_DATA_R = 0x2; // Executa se for falso
}
```

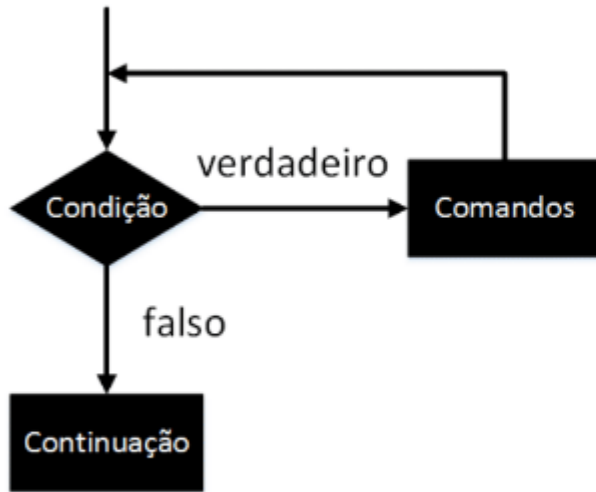
Controle de fluxo



```
char valor = GPIO_PORTB_AHB_DATA_R;
switch(valor)
{
    case 'A':
        GPIO_PORTJ_AHB_DATA_R = 0x1;
        break;
    case 'B':
        GPIO_PORTJ_AHB_DATA_R = 0x2;
        break;
    default:
        GPIO_PORTJ_AHB_DATA_R = 0;
}
```

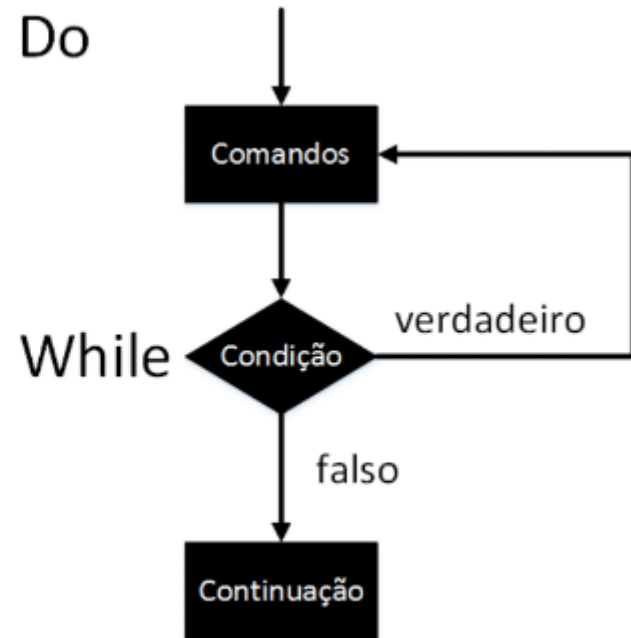
Controle de fluxo

While



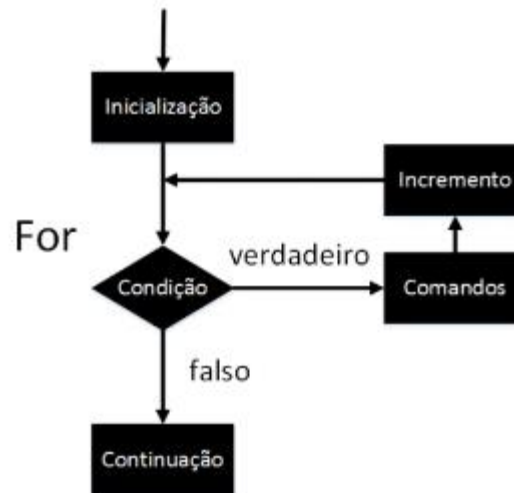
```
while (1) //loop infinito
{
    GPIO_PORTN_DATA_R = 0x1; //LED aceso
    SysTick_Wait1ms(500);    //delay 500ms
    GPIO_PORTN_DATA_R = 0x0; //LED apagado
    SysTick_Wait1ms(500);    //delay 500ms
}
```

Do



```
do (1)
{
    GPIO_PORTN_DATA_R = 0x1; //LED aceso
    SysTick_Wait1ms(500);    //delay 500ms
    GPIO_PORTN_DATA_R = 0x0; //LED apagado
    SysTick_Wait1ms(500);    //delay 500ms
    //loop condicional
} while (GPIO_PORTJ_AHB_DATA_R == 0x01)
```

Controle de fluxo



```
for (int i=0; i < 8; i++) //loop com for
{
    GPIO_PORTK_DATA_R = i;
    while (GPIO_PORTJ_AHB_DATA_R == 0x03); //fica esperando até apertar um botão
}
```

Matrizes e Vetores

```
uint8_t calendar[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30,  
31, 30, 31};
```

```
int32_t valores[12];
```

```
int32_t matriz[3][2];
```

```
char msg[]="Microcontroladores";
```

```
const char msg2[]="Microcontroladores";
```


Funções

```
tipo_de_retorno nome_da_funcao (tipo argumento1, tipo  
argumento2, ...)  
{  
    (código)  
    ...  
    return <var>;  
}
```

Funções

- Exemplo:

```
int32_t soma(int32_t a, int32_t b); //protótipo da função

int main(void)
{
    int32_t var1, var2, resultado;
    var1 = 6;
    var2 = 10;
    while (1)
    {
        resultado = soma(var1, var2);
        var1++;
    }
}

int32_t soma(int32_t a, int32_t b) //declaração da função
{
    return (a + b)
}
```

Macros

- Para definir constantes ou expressões

```
#define TAMANHO 10
#define ESTADO_INIT 0
#define ESTADO_ESPERANDO 1
#define ESTADO_TECLA_PRESSIONADA 2

#define SOMA(a, b) (a+b)
```

- Para definir posições da memória

```
#define SENSOR (*(volatile uint32_t *)0x4002400C)
#define PJ0 (*(volatile uint32_t *)0x40060004)
```

* **volatile** significa que a variável pode ser modificada sem o conhecimento do compilador, não devendo otimizar.

Structures

- Uma structure tem elementos com diferentes tipos.

```
typedef struct usuario
{
    uint32_t cpf;
    uint8_t idade;
} usuarioTipo;
```

```
usuarioTipo usuarioVars[10];
```

```
usuarioVars[i].cpf = 12345678912;
usuarioVars[i].idade = 35;
```

Enum

- Outro método para definir constantes.

```
typedef enum estAlarme
{
    ESTADO_INIT,
    ESTADO_CONFIG_ALARME,
    ESTADO_ESPERA_ALARME,
    ESTADO_DISPARANDO_ALARME
} estadosAlarme;

int main(void)
{
    estadosAlarme estados = ESTADO_INIT;
    switch (estados)
    {
        case ESTADO_DISPARANDO_ALARME:
            (código)
            break;
    }
}
```

Máquinas de Estado

- Exemplo

```
typedef enum estLampada
{
    ESTADO_ON,
    ESTADO_OFF
} estadosLampada;
estadosLampada estados = ESTADO_OFF;
int main(void)
{
    while(1)
    {
        switch (estados)
        {
            case ESTADO_OFF:
                estado_off_func();
                break;
            case ESTADO_ON:
                estado_on_func();
                break;
        }
    }
}
```

Máquinas de Estado

- Exemplo

```
void estado_off_func(void)
{
    if (on == 1)
        estados = ESTADO_ON;
}
```

```
void estado_on_func(void)
{
    if (on == 0)
        estados = ESTADO_OFF;
}
```

AAPCS

- *ARM Architecture Procedure Call Standard*
- Passagem de parâmetros para uma função:
 - Primeiros parâmetros em R0, R1, R2 e R3;
 - Demais parâmetros pela pilha.
- Retorno da função:
 - R0 (32 bits)
 - R1:R0 (64 bits)
- Alinhamento da pilha deve ser 64 bits:
 - PUSH/POP de número par de registradores antes/depois de chamadas de funções

Chamando uma função em C do ASM

THUMB

IMPORT Conta

IMPORT A

AREA |.text|,CODE,READONLY,ALIGN=2

EXPORT Start

Start

LDR R2, =A ;R2 = &A

MOV R0, #2

STR R0, [R2] ;A=2

Loop

MOV R0, #5 ;Função recebe
;parâmetros em R0 a R3
BL Conta ;Chama a função que irá
;multiplicar A (variável
;global) pelo parâmetro 1
;que está em R0 e coloca o
;resultado em R0

NOP

ALIGN

END

#include <stdint.h>

uint32_t A;

//Multiplica o parâmetro pelo fator A

uint32_t Conta(uint32_t parametro)

{

uint32_t resultado;

resultado = A * parametro;

return resultado;

}

Chamando uma função em ASM do C

```
THUMB
    AREA DATA, ALIGN=2
    EXPORT A [DATA,SIZE=4]
A    SPACE 4
    AREA |.text|, CODE, READONLY, ALIGN=2
        EXPORT Conta
        EXPORT A

;Retorna a multiplicação de A que está na
;memória pelo parâmetro recebido em R0 e
;retorna em R0 para a AAPCS
Conta
    LDR R2, =A      ;R2 = &A
    LDR R1, [R2]    ;R1 = [A]
        MUL R0, R0, R1
    BX LR

ALIGN
END
```

```
#include <stdint.h>

uint32_t Conta(uint32_t parametro);
extern uint32_t A;

int main (void)
{
    uint32_t resultado;
    A = 2;

    while (1)
    {
        resultado = Conta(5);
        A = resultado;
    }
}
```

Exemplo

- Abrir o projeto exemplo em C (anexo)
- *Build* e executar passo-a-passo