Universidade Federal de Ouro Preto

Instituto de Ciência Exatas e Aplicadas

Departamento de Engenharia Elétrica

Laboratório de Microprocessadores e Microcontroladores – CEA580

Prática 4

Alunos: Matheus Fernandes Gomes e Graziele de Cassia Rodrigues

Matrículas: 17.2.5941 e 21.1.8120

Objetivos

Compreender o princípio de funcionamento das instruções de salto e chamadas de sub-rotinas.

Referências

Mazidi and Naimi “The STM32F103 Arm Microcontroller and Embedded Systems”, Cap. 3 e 4.

Installing the Keil for STM32F10x step by step tutorial

STM32 Assembly Programming in Keil step by step tutorial

Lista de Materiais

Hardware

Item	Descrição	Quantidade
1	STM32F103C8 Blue Pill	**1**
2	ST-Link V2	**1**
3	*Protoboard*	**1**
4	Osciloscópio Digital	**1**
5	Resistor 1k	**3**
6	Led RGB	**1**
7	Cabos para conexão	

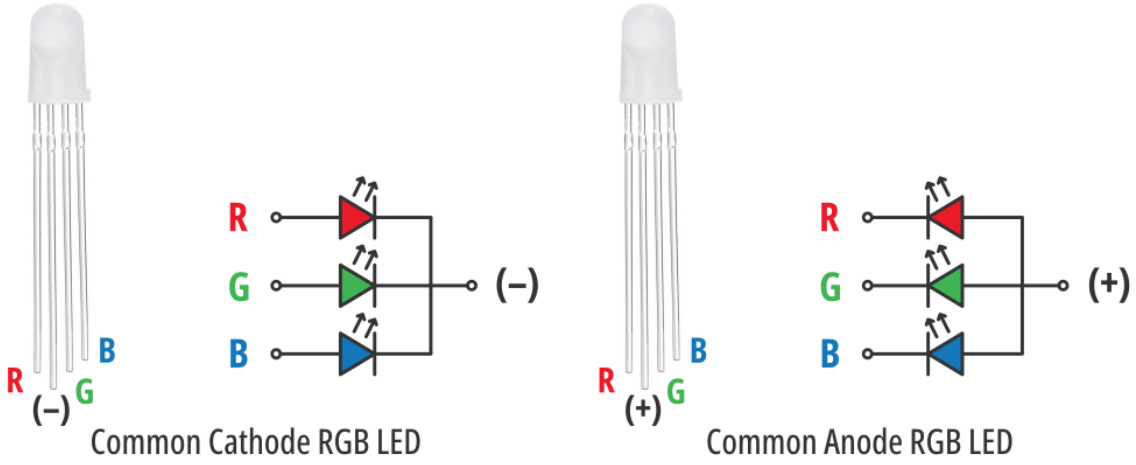
Software

Descrição	Link para download
Keil IDE - MDK-ARM	http://www.keil.com/
Keil.STM32F1xx_DFP.2.4.0.pack	https://www.keil.com/dd2/pack/

Conexões elétricas

A Figura 1 apresenta o diagrama de pinos de LEDs RGB com ânodo comum e cátodo comum. Verifique o tipo de LED utilizado na aula antes de iniciar as atividades.

Figura 1 – Pinagem de LEDs RGB com cátodo comum e ânodo comum.



Conecte as entradas R, G e B nas saídas A0, A1 e A2 do microcontrolador. Para limitar a corrente das saídas digitais, utilize resistores de 1k.

Atividades

1 – Escrever uma sub-rotina para gerar um delay de 100ms.

Para verificar se o delay ajustado está correto, escrever um algoritmo para:

A – Definir o nível lógico do pino A0 como alto;

B – Aplicar o delay de 100ms;

C – Definir o nível lógico do pino A0 como baixo;

D – Aplicar o delay de 100ms;

E – Voltar a etapa A.

Para medir os tempos ajustados em B e D, utilize um osciloscópio digital. Para isso, conecte o terminal positivo da ponta de prova ao pino A0, e o terminal negativo ao GND.

```
RCC_APB2ENR EQU 0x40021018

GPIOA_CRL       EQU 0x40010800
GPIOA_CRH       EQU 0x40010804
GPIOA_IDR       EQU 0x40010808
GPIOA_ODR       EQU 0x4001080C

GPIOB_CRL       EQU 0x40010C00
GPIOB_CRH       EQU 0x40010C04
GPIOB_IDR       EQU 0x40010C08
GPIOB_ODR       EQU 0x40010C0C

        EXPORT __main
        AREA MAIN, CODE, READONLY
__main
        LDR     R1,=RCC_APB2ENR
        LDR R0,[R1]
        ORR R0,R0,#0xFC                 ; enable the clocks for GPIOs
        STR R0,[R1]

        LDR R1,=GPIOB_CRL
        LDR R0,=0x33333333
        STR R0,[R1]                     ; PB0 to PB7 as outputs

        LDR R1,=GPIOB_CRH
        LDR R0,=0x33333333
        STR R0,[R1]                     ; PB8 to PB15 as outputs

        LDR R1,=GPIOA_CRL
        LDR R0,=0x88888888
        STR R0,[R1]                     ; PA0 to PA7 as inputs

        LDR R1,=GPIOA_CRH
        LDR R0,=0x88888888
        STR R0,[R1]                     ; PA8 to PA15 as inputs

        LDR R1,=GPIOA_ODR
        LDR R0,=0x0000
        STR R0,[R1]                     ; PA0 to PA15 pull-down

        MOV R5, #0      ; R5 = contador para controle de delay
        MOV R6, #0      ; R6 = valor do LED (inicialmente desligado)

LOOP
        ; LEITURA DAS CHAVES
        LDR     R10,=GPIOA_IDR
        LDR     R0,[R10]                ; R0 = value of GPIOA_IDR

        ; CHAVES
```

```
        ; R1 = SW1
        AND R1,R0,#0x01
        ; R2 = SW2
        LSR R0,R0,#0x01                 ; Logical Shift Right
        AND R2,R0,#0x01

        ; Inicio do algoritmo de controle

        ; LED RGB
        MOV R12, #2_111 ; R12 = 2_111 - cor branca inicial
        ORR R12, R12, R6, LSL #5        ; Adiciona valor do LED ao R12

        ; ATUALIZA LED RGB
        LDR R10,=GPIOB_ODR
        STR R12,[R10]     ; GPIOB_ODR = R12

        ; Delay
        MOV R5, #0x1000000
         B DELAY_LOOP

DELAY_LOOP
   SUBS R7, R5, #1 ; Subtrai R5 - 0xFFFFFF e atualiza as flags
   BNE DELAY_LOOP       ; Pula se o resultado não for zero
   B LOOP

        END



        END
```

2 – Escrever um algoritmo para alterar as cores do LED RGB. Para isso, considere as opções de cores definidas na Tabela 1.

Tabela 1 – Tabela de cores para LED RGB

	R	G	B
Vermelho	255	0	0
Verde	0	255	0
Azul	0	0	255
Branco	255	255	255
Amarelo	255	255	0
Magenta	255	0	255
Ciano	0	255	255

Procedimento:

A – Ligar led RGB com a primeira cor (vermelho) por 100ms;

B – Desligar led RGB por 100ms;

C – Repetir etapas A e B mais 4 vezes (100ms on, 100ms off, 100ms on, ... 100ms off)

D – Trocar a cor do led RGB e voltar para a etapa A.

Obs.: Quando o algoritmo atingir a última cor (ciano), reiniciar o processo com a cor inicial (vermelho).

```
RCC_APB2ENR EQU 0x40021018

GPIOA_CRL       EQU 0x40010800
GPIOA_CRH       EQU 0x40010804
GPIOA_IDR       EQU 0x40010808
GPIOA_ODR       EQU 0x4001080C

GPIOB_CRL       EQU 0x40010C00
GPIOB_CRH       EQU 0x40010C04
GPIOB_IDR       EQU 0x40010C08
GPIOB_ODR       EQU 0x40010C0C

        EXPORT __main
        AREA MAIN, CODE, READONLY
__main
        LDR     R1,=RCC_APB2ENR
        LDR R0,[R1]
        ORR R0,R0,#0xFC                 ; enable the clocks for GPIOs
        STR R0,[R1]
```

```
        LDR R1,=GPIOB_CRL
        LDR R0,=0x33333333
        STR R0,[R1]                     ; PB0 to PB7 as outputs

        LDR R1,=GPIOB_CRH
        LDR R0,=0x33333333
        STR R0,[R1]                     ; PB8 to PB15 as outputs

        LDR R1,=GPIOA_CRL
        LDR R0,=0x88888888
        STR R0,[R1]                     ; PA0 to PA7 as inputs

        LDR R1,=GPIOA_CRH
        LDR R0,=0x88888888
        STR R0,[R1]                     ; PA8 to PA15 as inputs

        LDR R1,=GPIOA_ODR
        LDR R0,=0x0000
        STR R0,[R1]                     ; PA0 to PA15 pull-down

        MOV R6, #0          ; R6 = valor do LED (inicialmente desligado)

LOOP
        ; LEITURA DAS CHAVES
        LDR     R10,=GPIOA_IDR
        LDR     R0,[R10]                ; R0 = value of GPIOA_IDR

        ; CHAVES
        ; R1 = SW1
        AND R1,R0,#0x01
        ; R2 = SW2
        LSR R0,R0,#0x01                 ; Logical Shift Right
        AND R2,R0,#0x01

        ; Inicio do algoritmo de controle
        CMP R1, #1          ; Verifica se SW1 está pressionado
        BEQ INCREMENTA

        CMP R2, #1          ; Verifica se SW2 está pressionado
        BEQ DECREMENTA

        B NAO_PRESSIONADO

INCREMENTA
        ADD R6, R6, #1      ; Incrementa valor do LED
        CMP R6, #8          ; Limite superior para o LED
        BGT LIMITE_SUPERIOR

        B NAO_PRESSIONADO
```

```
DECREMENTA
        SUB R6, R6, #1      ; Decrementa valor do LED
        CMP R6, #0          ; Limite inferior para o LED (pode ser ajustado)
        BLT LIMITE_INFERIOR

        B NAO_PRESSIONADO

LIMITE_SUPERIOR
        MOV R6, #7          ; Limite superior para o LED

        B NAO_PRESSIONADO

LIMITE_INFERIOR
        MOV R6, #0          ; Limite inferior para o LED

       B NAO_PRESSIONADO

NAO_PRESSIONADO
        ; Fim do algoritmo de controle

        ; LED RGB
        MOV R12, R6;
        LSL R12,R12,#5


        ; ATUALIZA LED RGB
        LDR R10,=GPIOB_ODR
        STR R12,[R10]       ; GPIOB_ODR = R12

        ; Delay
        MOV R5, #0x1000000
        B DELAY_LOOP

DELAY_LOOP
  SUBS R5, R5, #1 ; Subtrai R5 - 1 e atualiza as flags
  BNE DELAY_LOOP        ; Pula se o resultado não for zero
  B LOOP

        END
```