**Pontifícia Universidade Católica do Paraná**

**MICROPROCESSADORES I**

Prof.: Vilson Rodrigo Mognon/Afonso Ferreira Miguel - 2º Semestre/2020

ALUNO:Gabriel Vitor Cezário Grupo____Data: 30/10/2021

# LABORATÓRIO – CRONOMETRO COM DISPLAY DE 7 SEGMENTOS

**Objetivo:** Desenvolver um programa em Assembly para implementar um cronômetro digital utilizando o conceito de interrupções

1) Monte o circuito com 4 displays de sete segmentos do tipo catodo comum conforme esquemático a seguir.

| Segmento | Bit em PORTD |
|---|---|
| A | PD0 |
| B | PD1 |
| C | PD2 |
| D | PD3 |
| E | PD4 |
| F | PD5 |
| G | PD6 |
| DP | PD7 |

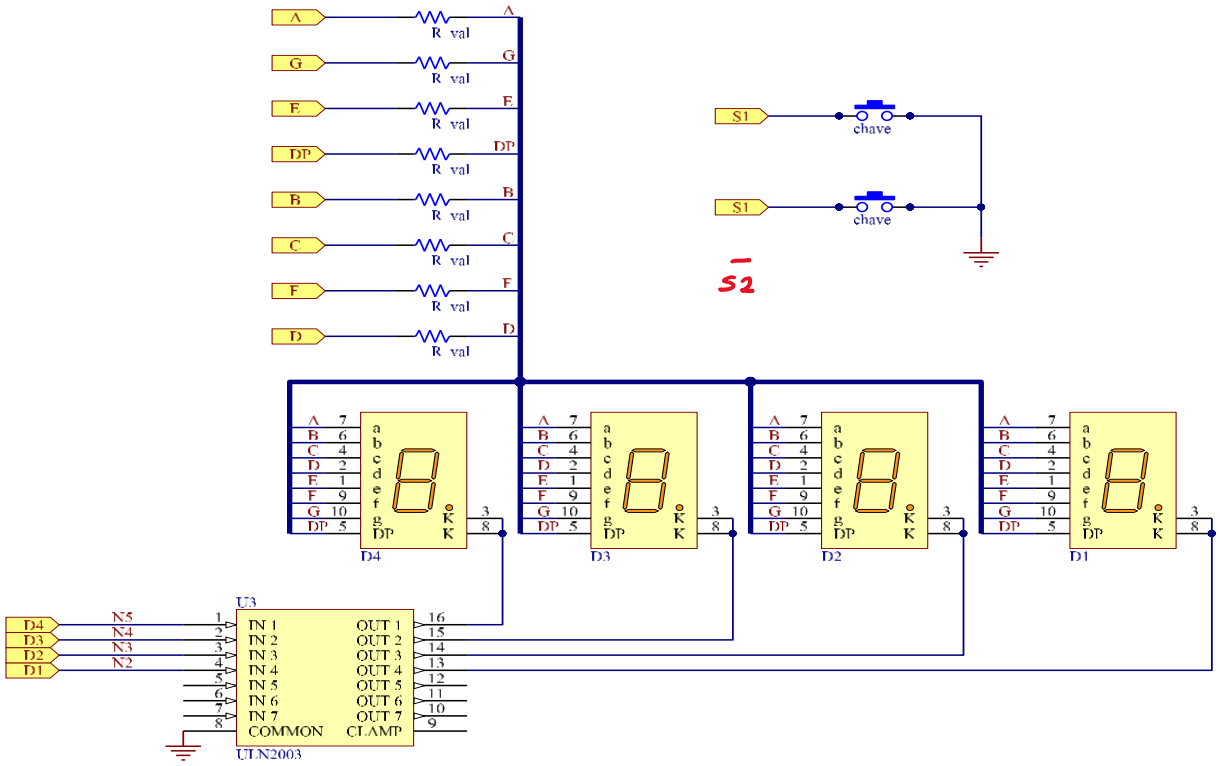
| Seletor de Display | Bit em PORTB |
|---|---|
| D1 | PB0 |
| D2 | PB1 |
| D3 | PB2 |
| D4 | PB3 |

| Push-buttom | Bit em PORTB |
|---|---|
| S1 | PB4 |
| S2 | PB5 |

2) Qual o valor dos registradores abaixo para configurar corretamente as portas de IO da placa do display de 7 segmentos e as teclas, considerando o estado inicial do display desligado.

| **BIT** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **DDRD** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **DDRB** | **1** | **1** | **0** | **0** | **1** | **1** | **1** | **1** |
| **PORTB** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

| **HEXA** |
|---|
| **0xFF** |
| **0xCF** |
| **0xFF** |




Para testar esta montagem, foi feito um programa padrão de teste disponível no comentário do vídeo de demonstração: https://youtu.be/eXzCsbfUh8U
O programa acende sequencialmente cada segmento dos displays. Ao pressionar S1 ou S2, a velocidade do sequencial muda. Este programa deve ser aberto e programado pelo Arduino IDE

3) Utilizando a planilha de mapeamento do display de sete segmentos, complete a tabela abaixo de maneira a formar os caracteres no display de 7 segmentos.

| **Caractere** | **BIT / SEGMENTO** | | | | | | | | **HEXA** |
|---|---|---|---|---|---|---|---|---|---|
| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | |
| **SEGMENTO →** | **dp** | **g** | **f** | **e** | **d** | **c** | **b** | **a** | |
| **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0x3F** |
| **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0x06** |
| **2** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0x5B** |
| **3** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0x47** |
| **4** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0x66** |
| **5** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **0x6E** |
| **6** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0x7E** |
| **7** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **0x07** |
| **8** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0x7F** |
| **9** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **1** | **0x67** |

4) Qual o valor em hexadecimal de PORTD e PORTB para mostrar no display as seguintes combinações:

| Display | | | | | |
|---|---|---|---|---|---|
| **4** | **3** | **2** | **1** | **PORTD** | **PORTB** |
| **Apagado** | **Apagado** | **Apagado** | **Apagado** | **0x00** | **0x00** |
| **Apagado** | **Apagado** | **Apagado** | **Apagado** | **0x00** | **0x00** |
| **3** | **Apagado** | **Apagado** | **Apagado** | **0x47** | **0x01** |
| **Apagado** | **4** | **Apagado** | **Apagado** | **0x66** | **0x02** |
| **Apagado** | **Apagado** | **A** | **Apagado** | **0x77** | **0x04** |
| **Apagado** | **Apagado** | **Apagado** | **8** | **0x7F** | **0x08** |

5) O que acontece se as combinações anteriores forem repetidas seqüencialmente em uma velocidade maior que a percepção do olho humano?

6) A função a seguir é o tratamento da interrupção de timer, que servirá para a base de tempo do cronometro. Complete as lacunas e faça os comentários da função.

```
;***************************************************************************
;     Interrupção de Timer0 Overflow
;---------------------------------------------------------------------------
TIMER0_OVERFLOW:
      PUSH        R16
      IN          R16, SREG
      PUSH        R16

      LDI         R16, 6
      OUT         TCNT0, R16

      sbr          R16, 0x00
      out          TCCR0B, R16
      rcall  centesimo     // valores
      puros
      sbr          R16, 0x03
      out          TCCR0B, R16

      POP         R16
      OUT         SREG,R16
      POP         R16
      RETI
```

Comentários:
Centesimo é uma subrotina que executa a atualização dos dados, porém necessita de uma aplicação externa para atualizar os bytes de saída da porta D sem que o programa reinicie-se com o WDT

7) Desenvolver um programa em Assembly que simule um cronômetro, com as funções da tabela abaixo e mostrando os displays conforme próxima figura.

| TECLA | FUNÇÃO |
|---|---|
| **S1** | **INICIA/RETOMA A CONTAGEM** |

| S2 | PAUSA CONTAGEM |
|---|---|


Veja a demonstração do funcionamento neste vídeo: https://youtu.be/hEDibCa7JQ8

8) Complete os quadros com as funções em Assembly com os comentários pertinentes.

- Código para criar um vetor de constantes na memória flash e a função para acessar estas informações. Utilizar para fazer a leitura de decodificação da tabela do display de 7 segmentos:

```
centesimo:
       cpi           R20, 9
       breq   unidade
       inc           R20
       rjmp final
unidade:
       clr           R20
       ldi           R24, 0x3F
       cpi           R21, 9
       breq   dezena
       inc           R21
       rjmp final
dezena:
       clr           R21
       ldi           R25, 0x3F
       cpi           R22, 5
       breq   minuto
       inc           R22
       rjmp final
minuto:
       clr           R22
       ldi           R26, 0x3F
       cpi           R23, 9
       breq   reseta
       inc           R23
       rjmp final
reseta:
       clr           R23
       ldi           R27, 0x3F
       final:
       ret
////////////////////////////////
play   R20, R24
       play   R21, R25
       play   R22, R26
       play   R23, R27
////////////////////
.macro play
```

```
        push    R16
        mov             R16, @0
        cpi             R16, 0
        breq    zero
        cpi             R16, 1
        breq    um
        cpi             R16, 2
        breq    dois
        cpi             R16, 3
        breq    tres
        cpi             R16, 4
        breq    quatro
        cpi             R16, 5
        breq    cinco
        cpi             R16, 6
        breq    seis
        cpi             R16, 7
        breq    sete
        cpi             R16, 8
        breq    oito
        cpi             R16, 9
        breq    nove
        rjmp    final_macro
zero:
        ldi             @1, 0x3F
        ret
um:
        ldi             @1, 0x06
        ret
dois:
        ldi             @1, 0x5B
        ret
tres:
        ldi             @1, 0x47
        ret
quatro:
        ldi             @1, 0x66
        ret
cinco:
        ldi             @1, 0x6E
        ret
seis:
        ldi             @1, 0x7E
        ret
sete:
        ldi             @1, 0x07
        ret
oito:
        ldi             @1, 0x7F
        ret
nove:
        ldi             @1, 0x67
        ret
final_macro:
.endm
```

- Código para inicializar o TIMER0 e disparar uma interrupção periodicamente a cada 1kHz

```
.macro insere_IO
        push    R16
        ldi             R16, @1
        out             @0, R16
        pop             R16
.endm
.macro insere_SRAM
        push    R16
```

```
        ldi             R16, @1
        sts             @0, R16
        pop             R16
.endm
insere_IO               TCCR0A, 0x00 ; Clock na forma normal
insere_IO               TCCR0B, 0x03 ; Fator de divisão 64 -> 250 para realizar 1 ms
insere_IO               TCNT0, 0x06  ; Começa em 6 na contagem até 250
insere_SRAM             TIMSK0, 0x01 ; Liga a interrupção por overflow de tempo (Ligada a chave da
parede)
sei                                  ; Liga a chave de interrupção global no SREG
```

- Código completo da interrupção do TIMER0, utilizada para atualizar o relógio

```
.org 0x20
       in            R16, SREG
       push   R16
       ldi           R16, 0x00
       sbr           R16, 0x00
       out           TCCR0B, R16
       rcall  centesimo     // valores puros
       sbr           R16, 0x03
       out           TCCR0B, R16
       pop           R16
       out           SREG, R16
       reti
////////////////////////////////////////////////////////
rcall  set_display1
rcall  set_display2
rcall  set_display3
rcall  set_display4
////////////////////////////////////////////////////////
set_display1:
       sbr           R16, 0x01
       out           PORTB, R16
       add           R17, R16
       out           PORTD, R24
       pause_i_us    4
       sbr           R16, 0x00
       out           PORTB, R16

       ret
set_display2:
       sbr           R16, 0x02
       out           PORTB, R16
       add           R17, R16
       out           PORTD, R25
       pause_i_us    4
       sbr           R16, 0x00
       out           PORTB, R16
       ret
set_display3:
       sbr           R16, 0x04
       out           PORTB, R16
       add           R17, R16
       out           PORTD, R26
       pause_i_us    4
       sbr           R16, 0x00
       out           PORTB, R16
       ret
set_display4:
       push   R16
       sbr           R16, 0x08
       out           PORTB, R16
       sbr           R17, 0x01
       out           PORTD, R27
       pause_i_us    4
       sbr           R16, 0x00
       out           PORTB, R16
       pop           R16
       ret
```

- Código da função principal e de outras funções auxiliares

```
.org 0
	rjmp config
////////////////////////////
.include "minhaLib.inc"
////////////////////////////

config:
	rcall  WDT_off
	insere_IO		DDRD,  0xFF ; Porta D em pull-up
	insere_IO		DDRB,  0xCF ; Saida B com algumas portas não funcionando como saida
somente (2 botoes)
	insere_IO		PORTB, 0xFF ; Porta B em pull-up
	insere_IO		PINB,  0xFF
	insere_IO		TCCR0A, 0x00 ; Clock na forma normal
	insere_IO		TCCR0B, 0X03 ; Clock com divisão de 64 para 1 ms
	insere_IO		TCNT0, 0x06 ; Começa em 6 na contagem até 256
	insere_SRAM		TIMSK0, 0x01 ; Liga a interrupção por overflow de tempo (Ligada a
chave da parede)
	sei                                             ; Liga a chave de interrupção global no
SREG
	/*
	R20 - decimal [0~9]
	R21 - unidade [0~9]
	R22 - dezena  [0~5]
	R23 - minuto  [0~9]
	*/

loop_eterno_inicio:
	insere_IO		TCCR0B, 0X00
loop_eterno:
	in	R19, PINB
	sbrc	R19, 4
	rjmp loop_inicio
	rjmp	loop_eterno
loop_inicio:
	insere_IO		TCCR0B, 0X03
loop:
	in	R19, PINB
	sbrc	R19, 5
	rjmp	loop_eterno_inicio
	play	R20, R24
	play	R21, R25
	play	R22, R26
	play	R23, R27
	rcall	set_display1
	rcall	set_display2
	rcall	set_display3
	rcall	set_display4
	rjmp	loop
```