|  |  |
| --- | --- |
|  | Pontifícia Universidade Católica do Paraná **MICROPROCESSADORES I**  Prof.: Vilson Rodrigo Mognon/Afonso Ferreira Miguel - 2o Semestre/2021  ALUNO: Gabriel Vitor Cezário Data: 19/11/2021 |

**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 | PC0 |
| S2 | PC1 |

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** |  | **HEXA** |
| **DDRD** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |  | **0xFF** |
| **DDRB** | **1** | **1** | **0** | **0** | **1** | **1** | **1** | **1** |  | **0xCF** |
| **PORTB** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |  | **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** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0xBF** |
| **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0x86** |
| **2** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0xDB** |
| **3** | **1** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0xCF** |
| **4** | **1** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0xE6** |
| **5** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **0xED** |
| **6** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0xFD** |
| **7** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **0x87** |
| **8** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0xFF** |
| **9** | **1** | **1** | **1** | **0** | **0** | **1** | **1** | **1** | **0xE7** |

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** | **0xCF** | **0x01** |
| **Apagado** | **4** | **Apagado** | **Apagado** | **0xE6** | **0x02** |
| **Apagado** | **Apagado** | **A** | **Apagado** | **0x77** | **0x04** |
| **Apagado** | **Apagado** | **Apagado** | **8** | **0xFF** | **0x08** |

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

R: Todos os números vão aparecer ao mesmo tempo em cada display na visão do observador

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

;----------------------------------------------------------------------------

O compare serve para que quando seja a centésima (99 de 0 a 99) repetição de 1 ms, ele vá para a atualização de valores, ou seja, para uma sequência de sub-rotinas guiadas por centésimo

Retorno\_inte é o ponto para qual o código pula após a sub rotina centésimo independente do que aconteça

TIMER0\_OVERFLOW:

PUSH R16

IN R16, SREG

PUSH R16

LDI R16, 6

OUT TCNT0, R16

inc R20

cpi R20,99

breq centesimo // valores puros

retorno\_inte:

POP R16

OUT SREG,R16

POP R16

RETI

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:

|  |
| --- |
| // Set-up da tabela  tabela\_R18:  ldi ZL,LOW(tabela\*2)  ldi ZH,HIGH(tabela\*2)  add ZL,R18  ldi R18,0  adc ZH,R18  lpm R18,Z  ret  // Tabela de conversao numero-display  tabela:  .db 0xBF, 0x86, 0xDB, 0xCF, 0xE6, 0xED, 0xFD, 0x87, 0xFF, 0xE7  //////////////////////////////////////////////////////////////////////////////////  /////////////////////////////////////////////////////////////////////////////////  .macro set\_display  push R17  ldi R17,0  out PORTB,R17 // reseta o display (apaga)  ; Busca na tabela em memória flash os dados  mov R18, @0  // Coleta o número usado, para buscar semelhante a um vetor o número na tabela  call tabela\_R18  // Consulta a tabela em memória flash como um vetor e o valor estará em R18 para  // ser mostrado no display  ldi R17, @1  out PORTB,R17  out PORTD,R18  pop R17  .endm |

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

|  |
| --- |
| loop\_config\_inicial: // inicia configuracoes de clock  insere\_IO TCCR0A, 0x00  insere\_IO TCCR0B, 0X03  insere\_IO TCNT0, 0x06  insere\_SRAM TIMSK0, 0x01  sei // Liga a chave de interrupção global no SREG  rjmp loop    //////////////////////////////////////////////////////////////////////////////////  /////////////////////////////////////////////////////////////////////////////////  loop\_clock: // inicio/reinicio do timer  insere\_IO TCCR0B, 0X03  insere\_IO TCCR0A, 0x00 |

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

|  |
| --- |
| .org 0x20  push R16  in R16, SREG  push R16  sbr R16, 0x06  out TCNT0, R16  inc R20  cpi R20,99  breq centesimo // valores puros  retorno\_inte:  pop R16  out SREG, R16  pop R16  reti |

* Código da função principal e de outras funções auxiliares

|  |
| --- |
| .org 0  rjmp config  centesimo:  clr R20  inc R21  cpi R21, 10  breq unidade  rjmp retorno\_inte  unidade:  clr R21  inc R22  cpi R22, 10  breq dezena  rjmp retorno\_inte  dezena:  clr R22  inc R23  cpi R23, 6  breq minuto  rjmp retorno\_inte  minuto:  clr R23  inc R24  cpi R24, 10  breq reseta  rjmp retorno\_inte  reseta:  ldi R20,0  ldi R21,0  ldi R22,0  ldi R23,0  ldi R24,0  rjmp retorno\_inte  .org 0x20  push R16  in R16, SREG  push R16  sbr R16, 0x06  out TCNT0, R16  inc R20  cpi R20,99  breq centesimo // valores puros  retorno\_inte:  pop R16  out SREG, R16  pop R16  reti  .include "minhaLib.inc"  config:  insere\_IO DDRD, 0xFF // Porta D como saida  insere\_IO DDRB, 0xFF // Saida B com algumas portas não funcionando como saida somente (2 botoes)  cbi DDRC,0  cbi DDRC,1  sbi PORTC,0  sbi PORTC,1  insere\_SRAM ADMUX, 0x2F // coloca o pino do analógico longe do PINC 0 e 1  /\*  R19 - interrupção [0~99]  R20 - decimal [0~9]  R21 - unidade [0~9]  R22 - dezena [0~5]  R23 - minuto [0~9]  \*/    ldi R19,0  ldi R20,0  ldi R21,0  ldi R22,0  ldi R23,0 // todos os numeros são 0  // inicio  comeco:  set\_display R21,0x01  pause\_i\_ms 4  set\_display R22,0x02  pause\_i\_ms 4  set\_display R23,0x04  pause\_i\_ms 4  set\_display R24,0x08  pause\_i\_ms 4  SBIC PINC, 0 // botao S1  rjmp comeco  rjmp loop\_config\_inicial  loop\_config\_inicial: // inicia configuracoes de clock  insere\_IO TCCR0A, 0x00  insere\_IO TCCR0B, 0X03  insere\_IO TCNT0, 0x06  insere\_SRAM TIMSK0, 0x01  sei // Liga a chave de interrupção global no SREG  rjmp loop    loop\_clock: // inicio/reinicio do timer  insere\_IO TCCR0B, 0X03  insere\_IO TCCR0A, 0x00    loop:  SBIS PINC,1 // botao S2  rjmp pausa  set\_display R21,0x01  pause\_i\_ms 4  set\_display R22,0x02  pause\_i\_ms 4  set\_display R23,0x04  pause\_i\_ms 4  set\_display R24,0x08  pause\_i\_ms 4  rjmp loop  pausa:  insere\_IO TCCR0B, 0X00 // para o clock  pause\_loop:    set\_display R21,0x01  pause\_i\_ms 4  set\_display R22,0x02  pause\_i\_ms 4  set\_display R23,0x04  pause\_i\_ms 4  set\_display R24,0x08  pause\_i\_ms 4  SBIC PINC,0 // Se S1 for pressionado, reinicia o clock,  // senão mostra os mesmos valores no display  jmp pause\_loop  jmp loop\_clock  // Set-up da tabela  tabela\_R18:  ldi ZL,LOW(tabela\*2)  ldi ZH,HIGH(tabela\*2)  add ZL,R18  ldi R18,0  adc ZH,R18  lpm R18,Z  ret  // Tabela de conversao numero-display  tabela:  .db 0xBF, 0x86, 0xDB, 0xCF, 0xE6, 0xED, 0xFD, 0x87, 0xFF, 0xE7  //////////////////////////////////////////////////////////////////////////////////  /////////////////////////////////////////////////////////////////////////////////  .macro set\_display  push R17  ldi R17,0  out PORTB,R17 // reseta o display (apaga)  ; Busca na tabela em memória flash os dados  mov R18, @0 // Coleta o número usado, para buscar semelhante a um vetor o número na tabela  call tabela\_R18 // Consulta a tabela em memória flash como um vetor e o valor estará em R18 para  // ser mostrado no display  ldi R17, @1  out PORTB,R17  out PORTD,R18  pop R17  .endm |