UNIVERSIDADE da MADEIRA

Relatório de Arquitetura de Computadores

Segundo projeto prático

Professores :

Dionísio Barros

Sofia Inácio

Dino Vasconcelos

Pedro Camacho

Trabalho realizado por :

Paulo Alexandre Rodrigues Alves Nº2120722

Renato Gabriel Silva Pêssego Nº2121922

Índice

1.Objetivos

Este relatório incidiu sobre o terceiro projeto prático de arquitetura de computadores. O objetivo foi realizar um sistema de perguntas e respostas de escolha múltipla, do botão A ao D, que apresenta um temporizador de cinco segundos para responder e, que regista o respectivo tempo restante ao dar a resposta, sendo estas informações mostradas nos displays de sete segmentos.

A sua implementação foi feita com recurso ao microcontrolador AT89S51 e às linguagens de programação C e Assembly no programa Keil uVision 5.

2.Descrição da solução e análise de resultados

2.1 Declaração de funções e variáveis

Antes de inicializar o programa, em C, são declaradas as constantes, variáveis e funções que serão utilizadas. Aqui são atribuídas as inicializações de cada uma das variáveis e constantes, enquanto em assembly, o mesmo é feito apenas na função principal.

2.2 Função principal (main)

Já na função principal, é chamada a função que ativa as interrupções, colocando o IE a 87h, isto é, ativando o timer 0, e as interrupções externas 0 e 1, e o IT0 é também colocado a 1 para detetar as interrupções na transição descendente do clock. Após isto, é chamada a função que define as prioridades das interrupções, colocando o botão 1 (interrupção externa 0) como a mais prioritária e as restantes com igual prioridade, ou seja, IP = 0x01. É então chamada a função ativa temporizador, que irá definir o modo do temporizador com o modo 1 de 16 bits (TMOD = 0x01) e iniciar o temporizador com o valor D8F0h (TL0 = 0xF0 e TH0 = 0xD8). Este valor foi obtido pois, sabemos que o clock vai dar overflow quando somar 1 ao seu valor máximo, que é FFFFh, e, no nosso caso, queremos contar ciclos de 10ms ou seja 10000 ciclos de relógio, então para tal, temos de subtrair a este valor de overflow o número de ciclos que queremos contar (FFFF + 1 - 2710h (10000 em decimal), obtendo então os tais D8F0h. Por fim, esta função coloca o TR0 = 1 para ativar o temporizador. Por fim, esta função main vai ficar num loop infinito de while que só será interrompido quando alguma interrupção for chamada. Em assembly, a única diferença é que as inicializações são uma rotina chamada nesta função main que atribui os valores iniciais às variáveis.

2.3 Interrupção do temporizador(overflow)

Quando ocorre overflow do temporizador esta interrupção é chamada e executada. A princípio voltamos a atribuir os mesmos valores que no ínicio do programa ao TH0 e TL0 pois vamos voltar a contar 10ms. Então decrementamos a variável de tempo inicial que verifica se já passaram 10 ciclos de 10ms, isto é 100ms, e então verificamos se a mesma é maior que 0. Se for então não fazemos nada e o programa continua até este valor ser 0. Quando isto ocorre, passaram-se 100ms, então vamos reiniciar a variável, isto é atribuir-lhe o valor 10 novamente, e então verificamos se a variável booleana/bit de esperar resposta é 1, e caso seja, então quer dizer que vamos decrementar o contador. Para tal, verificamos se os milissegundos são 0 e, se o forem, verificamos se os segundos também o são. Caso o sejam, temos que terminar o contador pois já terminou o tempo para responder. Terminar o contador é uma função que coloca a variável esperar resposta a 0, e atribui à resposta 0, isto é, não respondeu. É colocado também os segundos restantes e milissegundos restantes a 0, sendo então chamada a função de atualizar display, que recebe estes valores de tempo restante, e através de um switch verifica qual é o valor dos segundos e milissegundos, e coloca o valor dos bits necessários a ativar, na porta correspondente a cada display. Já no caso onde os milissegundos são 0 mas os segundos não, decrementamos os segundos e colocamos os milissegundos a 9. Já caso os milissegundos sejam diferentes de 0, decrementamos os milissegundos. Por fim, atualizamos o display com o valor atual dos segundos e milissegundos da mesma forma descrita anteriormente. Por outro lado, se o esperar resposta é 0, verificamos se o booleano congelar timer é verdadeiro e, se o for, atualizamos o display com os 5s e 0ms. Se ambas variáveis são falsas, isto significa que temos uma resposta dada. Neste caso vamos decrementar o ciclo de mostrar resposta que começa a 20 (2s) e vai ser decrementado até 0, se o mesmo for maior que 10, isto é, se estiver entre 1 e 2s, vamos atualizar o display com as variáveis dos segundos e milissegundos restantes, caso contrário, sabemos que o ciclo está entre 0 e 2s e, neste caso, verificamos se o ciclo é igual a 0 e, se o for, atualizamos para o valor inicial (20) de forma a continuar o ciclo de mostrar resposta/tempo restante. Depois desta verificação atualizamos o display com a resposta dada, de forma semelhante a como se atualiza o display com tempo, usando um switch para cada letra ou 0 no caso de não haver resposta.

2.4 Interrupção do botão 1(interrupção externa 0)

Quando o botão 1 é pressionado é chamada esta interrupção, que vai verificar se não se espera resposta e o timer não está congelado, e, se for verdade, então vamos reinicia-lo e congelá-lo, seguidamente reiniciamos o tempo inicial com 10, colocamos o congelar timer a 1 e reiniciamos os segundos a 5 e os milissegundos a 0. Caso não se espere resposta e o congelar resposta seja verdadeira, então vamos descongelar o timer, e, para tal, vamos reiniciar o tempo inical, colocar o esperar resposta a 1, e o congelar timer a 0. Caso o esperar resposta seja verdade, significa que o timer já está a decrementar, logo, vamos reiniciar o timer de forma a este ficar congelado a 5s. Para tal, reiniciamos o tempo inicial, os segundos a 5, os milissegundos a 0, e atualizamos o display com estes valores. Para além disto, congelamos o timer, colocando a variável a 1, e o esperar resposta a 0.

2.5 Interrupção botão de resposta(interrupção externa 1)

Quando o botão resposta é pressionado, esta interrupção é chamada. Esta vai verificar se a variável esperar resposta é 1 e, se não for, não faz nada. Caso contrário esta coloca o esperar resposta a 0, os segundos restantes e milissegundos restantes passam a ser iguais aos segundos e milissegundos atuais, respectivamente e, o ciclo mostrar resposta é reiniciado para 20, para os 2s de ciclo de mostrar a resposta. Por fim, é verificado por um conjunto de ifs a fim de saber qual dos botões de resposta foi pressionado, e é então atribuída essa letra correspondente à variável resposta.

2.6 Outras diferenças entre C e assembly

Todo o raciocínio é semelhante nas duas linguagens, sendo que em assembly, não existe a variável congelar timer, tendo em vez disso, a resposta como padrão a “0eh”, que significa que o timer está congelado, então quando o programa deteta que esta é a resposta, age como em C quando o congelar resposta está a 1. Fora isso, todo o raciocínio é o mesmo sendo feito apenas as alterações relativas a cada linguagem.

3.Conclusão

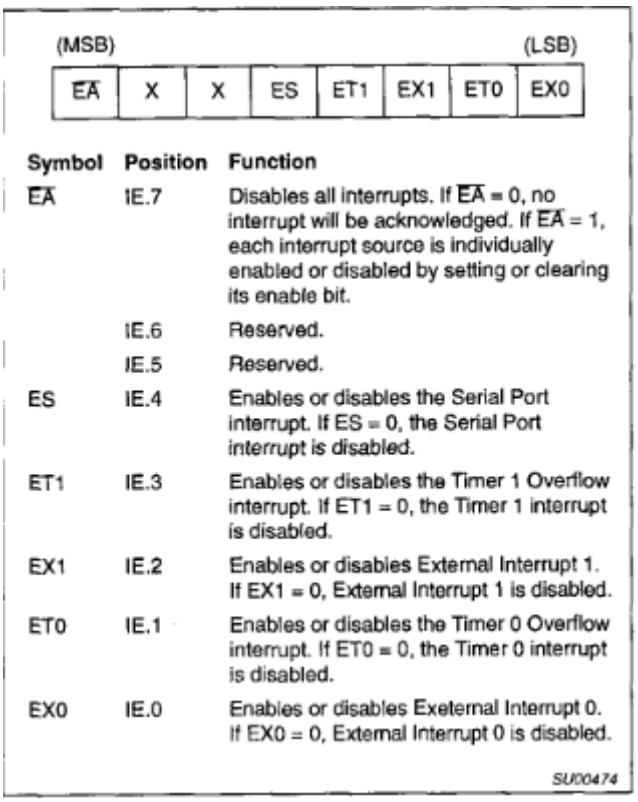
Concluindo, no fim deste projeto, percebemos que somos capazes de construir um programa 100% funcional para um microcontrolador como o AT89S51, em diferentes linguagens, tal como C e assembly. Somos também capazes de entender como funciona o tratamento de exceções de um microcontrolador, e as principais diferenças entre a linguagem de programação C e assembly.

4.Bibliografia

João Dionísio Simões Barros, Sebenta L3 2002-2003
J. Delgado e C. Ribeiro, Arquitectura de Computadores, 5ª edição, FCA, 2015.
D. A. Patterson and J. L. Hennessy, Computer Organization and Design, The hardware/software interface, Elsevier, 3 Edition, 2005.

5.Lista de figuras

Figura 1 - Registo IE



(MSB) (LSB)

$\overline{EA}$	X	X	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
$\overline{EA}$	IE.7	Disables all interrupts. If $\overline{EA}$ = 0, no interrupt will be acknowledged. If $\overline{EA}$ = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
	IE.5	Reserved.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables Exeternal Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

SU00474

Figura 2 - Registo TCON

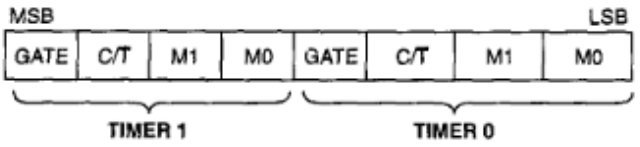
Registo TCON

MSB LSB

GATE	C/T	TF	TR	IE0	IT0	IE1	IT1

GATE	1	Temporizador/Contador está activo quando o INT0 = 1 e TR = 1.
	0	Temporizador/Contador está activo quando o TR = 1.
C/T	1	Temporizador/Contador opera a partir do pino T0.
	0	Temporizador opera a partir do *clock* interno.
TF	1	Fica activo quando há um *overflow* em T0.
	0	Fica com o valor nulo quando a interrupção é tratada ou pela operação de *reset*.
TR	1	Activa o Temporizador/Contador 0.
	0	Desactiva o Temporizador/Contador 0.
IE0	1	Foi detectada uma transição em INT0.
IT0	1	O sinal INT0 é detectado na transição.
	0	O sinal INT0 é detectado ao nível.
IE1	1	Foi detectada uma transição em INT1.
IT1	1	O sinal INT1 é detectado na transição.
	0	O sinal INT1 é detectado ao nível.

Figura 3 - Registo TMOD



	GATE	Gating control when set. Timer/Counter "x" is enabled only while "$\overline{INTx}$" pin is high and "TRx" control pin is set. when cleared Timer "x" is enabled whenever "TRx" control bit is set.
	C/T	Timer or Counter Selector cleared for Timer operation (input from in=ternal system clock.) Set for Counter operation (input from "Tx" input pin).

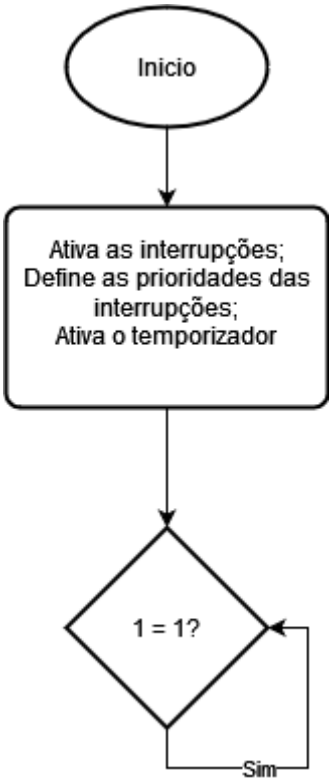
M1	M0	OPERATING
0	0	8048 Timer "TLx" serves as 5-bit prescaler.
0	1	16-bit Timer/Counter "THx" and "TLx" are cascaded; there is no prescaler.
1	0	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 stopped.

SU00534

Figura 4 -

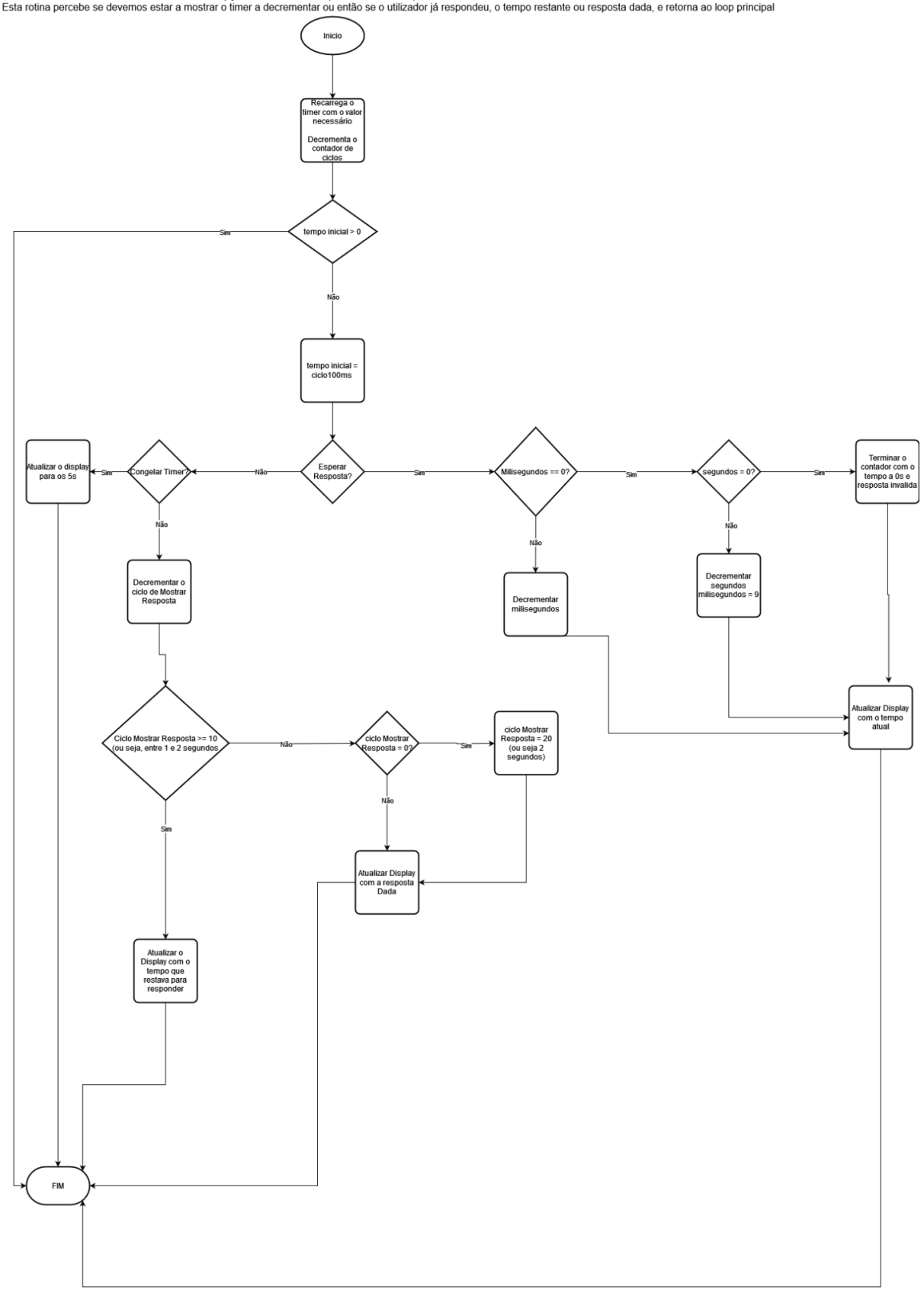
6.Anexo A

Fluxograma 1 - Rotina principal/Função main



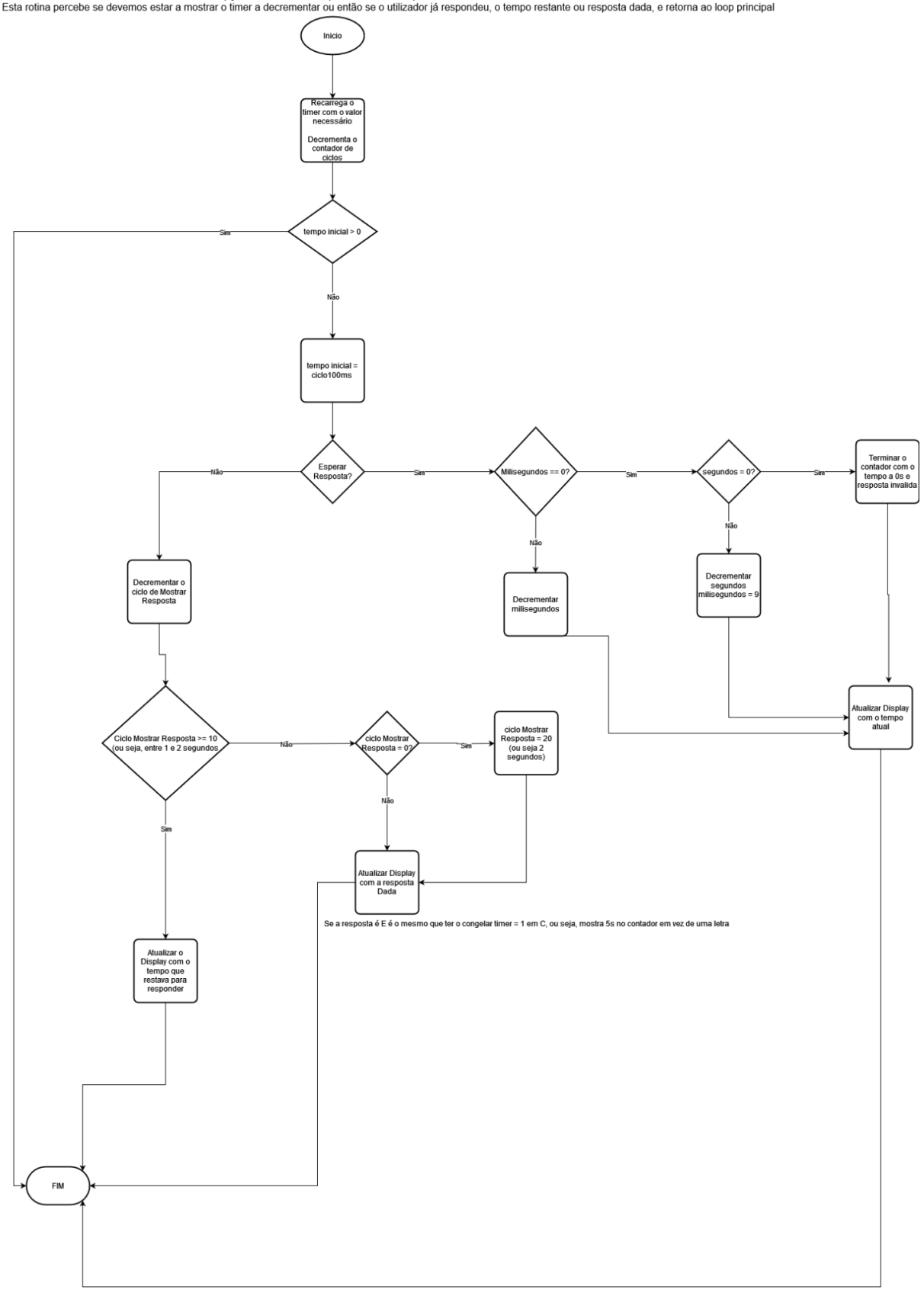
Fluxograma 2 - Rotina interrupção timer em C

NOTAS: Esta rotina só será chamada se a interrupção timer 0 acontecer(isto é der overflow).
Esta rotina percebe se devemos estar a mostrar o timer a decrementar ou então se o utilizador já respondeu, o tempo restante ou resposta dada, e retorna ao loop principal

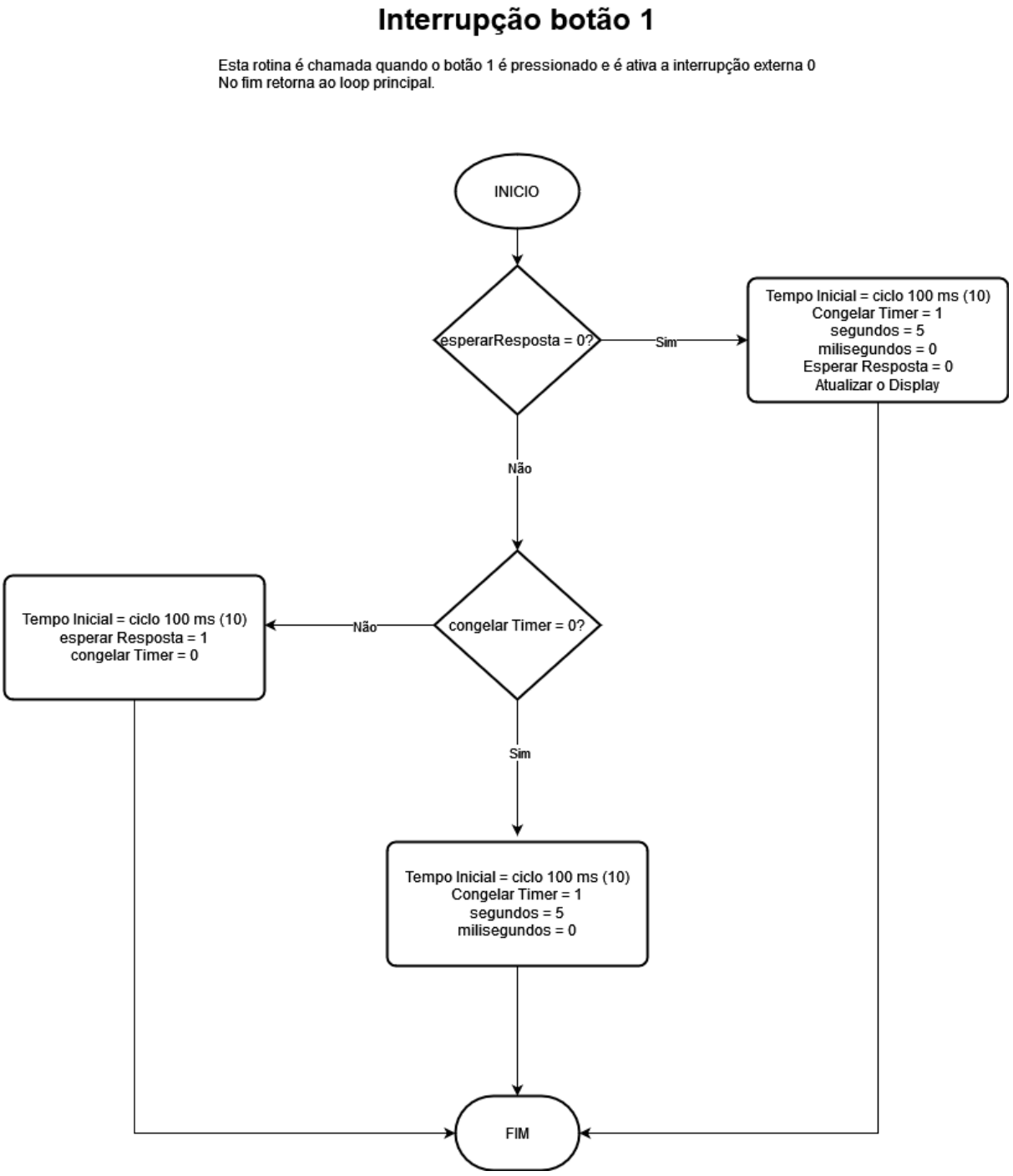


Fluxograma 3 - Rotina interrupção timer em Assembly

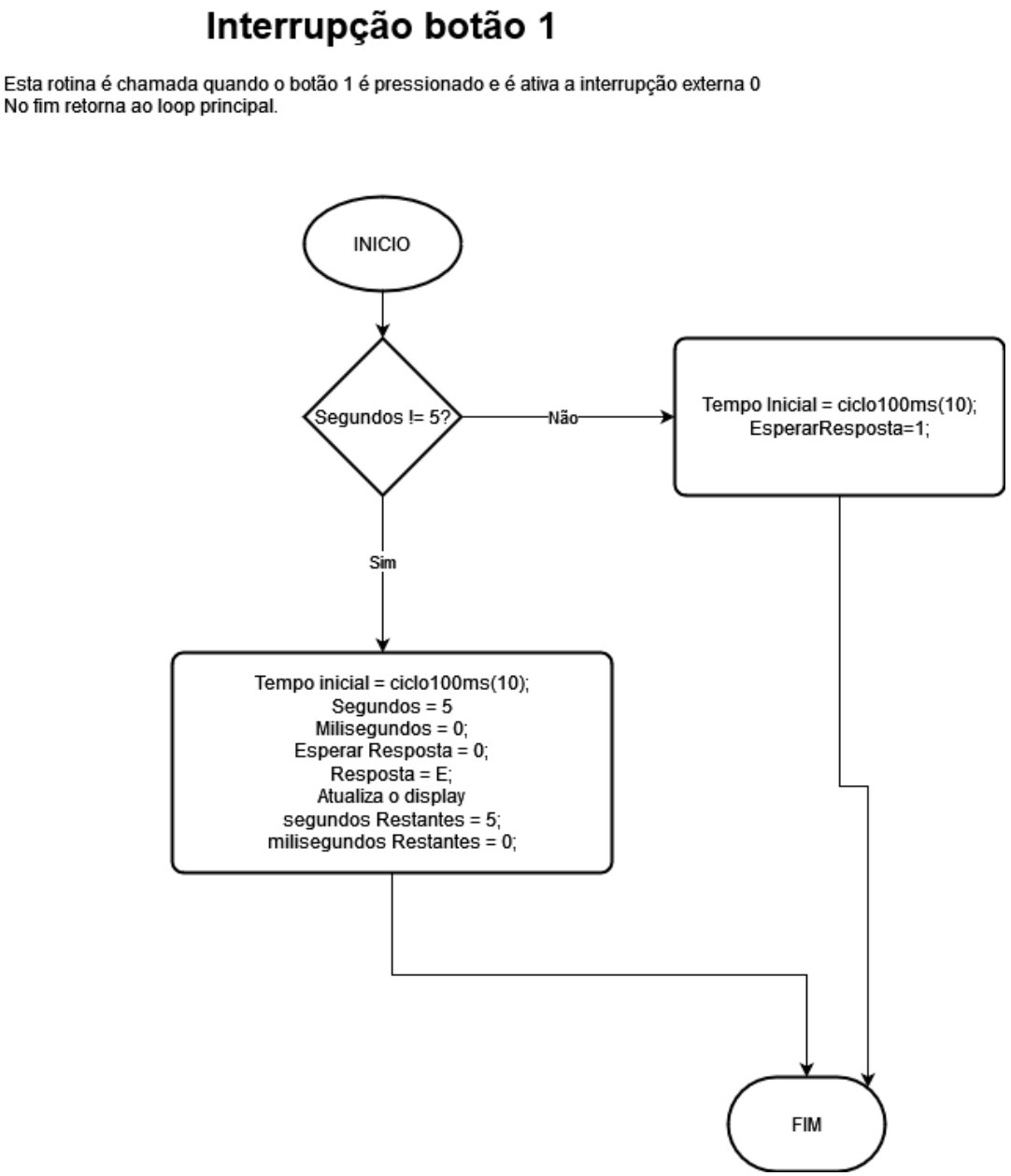
NOTAS: Esta rotina só será chamada se a interrupção timer 0 acontecer(isto é der overflow).
Esta rotina percebe se devemos estar a mostrar o timer a decrementar ou então se o utilizador já respondeu, o tempo restante ou resposta dada, e retorna ao loop principal



Fluxograma 4 - Rotina botão um em C

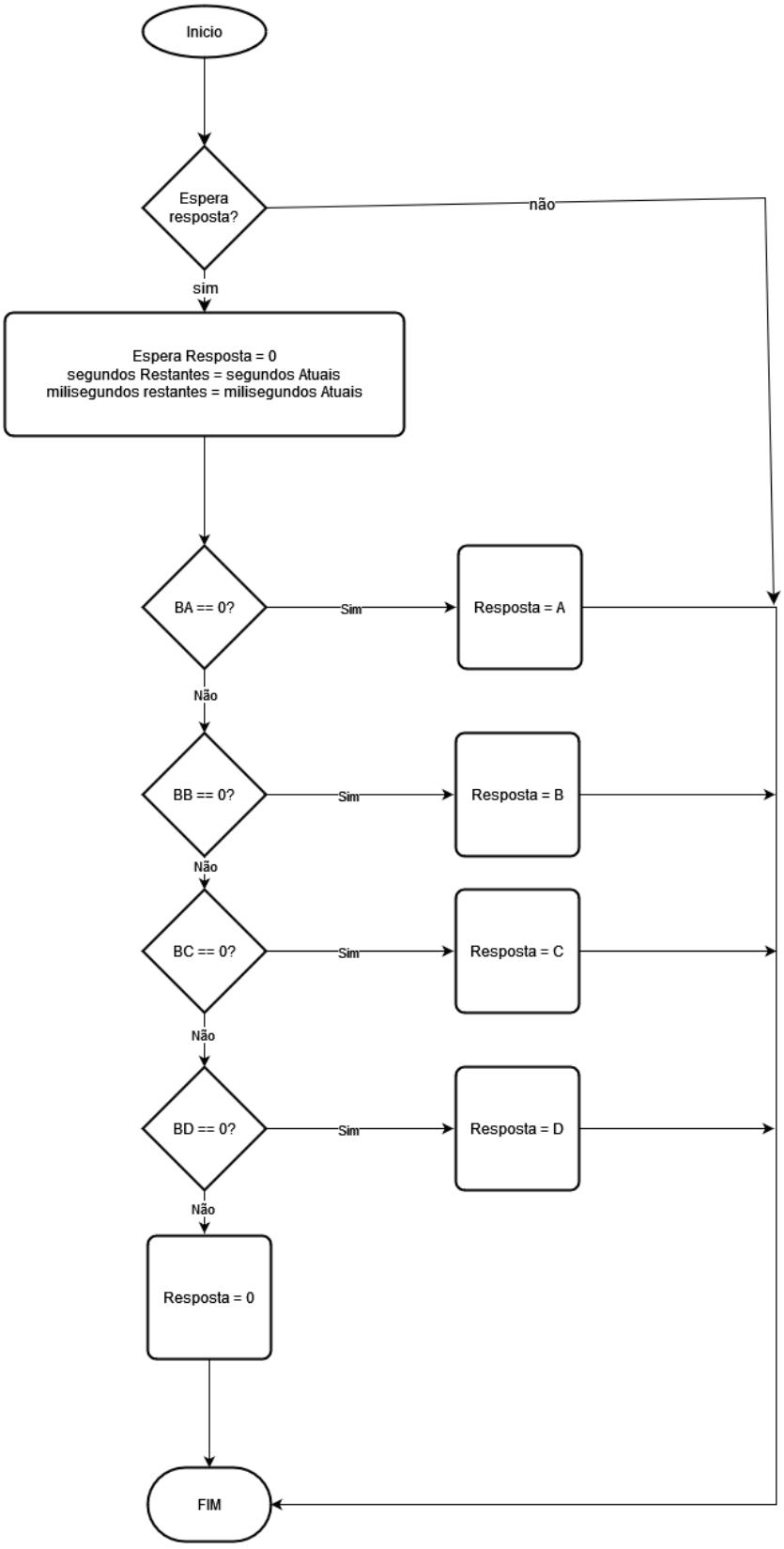


Fluxograma 5 - Rotina botão um em Assembly



Fluxograma 6 - Rotina recebeu resposta

NOTAS: Esta rotina só será chamada se a interrupção de p3.3 acontecer(interrupcao externa1). E retorna ao loop principal



7.Anexo B

Em C

```
#include <reg51.h>
// Definição das constantes  /Definimos estas constantes para uma melhor manutencao do programa permitindo-nos mudar o funcionamento do mesmo apenas alterando estes valores
#define ciclo100ms 10
#define numeroCiclosMostrarResposta 20 //Por exemplo se quisermos fazer ciclos de 4s(2s para cada estado do display) mudamos de 20 para 40 ou se quisermos mudar a contagem do relogio tbm o podemos fazer alterando apenas as nossas constantes

// Timer
#define TempoL 0xF0
#define TempoH 0xD8

//pins usados
sbit B1 = P3^2;     // Botão 1
sbit BResposta = P3^3; // PIN de qualquer resposta
sbit BA = P3^4;     // Botão A
sbit BB = P3^5;     // Botão B
sbit BC = P3^6;     // Botão C
sbit BD = P3^7;     // Botão D


//Funcoes usadas
void ativaInterrupcoes(void);
void prioridadeInterrupcoes(void);
void ativaTemporizador(void);
void terminarcontador(void);
void atualizarDisplays(int, int);
void atualizarDisplayResposta(char);

//Variaveis
int tempoInicial = ciclo100ms;            //Variavel que conta o numero de ciclos de 10 ms  //10 ciclos de 10ms para dar 100 ms
int segundos = 5;                                    //Segundos do contador //segundos iniciais
int miliSegundos = 0;               //milisegundos do contador  //milisegundos iniciais
bit congelarTimer = 1;      //variavel booleana que nos diz se o timer esta parado a mostrar 5s //Inicialmente o timer esta "congelado"
```

```
bit esperarResposta = 0;     //variavel booleana que nos diz se esperamos uma
resposta do utilizador ou nao  //Inicialmente nao esperamos uma resposta
apenas um input do botao 1
int segundosRestantes = 5;    //Segundos restantes quando o utilizador
respondeu  //Inicialmente restam 5 segundos para responder
int miliSegundosRestantes = 0;  //milisegundos restantes quando o utilizador
respondeu  //Inicialmente restam 0 milisegundos para responder
int cicloMostrarResposta = numeroCiclosMostrarResposta; //contador que serve
para realizar o ciclo de 1 em 1s  //inicalmente temos 20 ciclos de 10ms para dar
2s, 1 para cada estado do display
char Resposta = '0';    //Inicialmente nao temos uma resposta

//Funcao primaria
void main(){
   ativaInterrupcoes();     //Ativar as interrupcoes
   prioridadeInterrupcoes();       //Atribuir as prioridades a cada interrupcao
   ativaTemporizador();    //ativar o timer
   while(1);                  //ciclo da funcao principal, esperando pelas
interrupcoes para reagir
}

void ativaInterrupcoes(){
   IE = 0x87; // Ativa interrupções timer 0 e externas 1 (P3.3) e 0 (P3.2)
  IT0 = 1; // A interrupção externa será detectada na transição descendente do
clk
}
void prioridadeInterrupcoes() {
        IP = 0x01; // A interrupcao externa 0 tem a maior prioridade
}

void ativaTemporizador() {
        TMOD = 0x01; // Ativa o temporizador no modo de leitura de 16 bits
        TL0 = 0xF0; // Valor do byte menos significativo
        TH0 = 0xD8; // Valor do byte mais significativo
        TR0 = 1; // Ativa o temporizador 0, para fazer contagens de 10ms
   return;
}

void Timer(void) interrupt 1 {
     TL0 = 0xF0; // Valor do byte menos significativo
        TH0 = 0xD8; // Valor do byte mais significativo
   tempoInicial--;
   if(tempoInicial > 0){  //Se o tempo inicial nao e 0 entao so decrementamos o
tempo inicial e continuamos
         return;
```

```
    }
    else{                                                    //Se nao
resetamos o tempo inicial e verificamos se estamos a espera de uma resposta
        tempoInicial = ciclo100ms;
        if(esperarResposta){            //Se sim decrementamos o tempo
conforme possivel
        if(miliSegundos == 0){    //Se os milisegundos ja forem 0
decrementamos os segundos
            if(segundos==0){                //Se tanto os milisegundos como
os segundos sao 0 terminamos o contador
                terminarcontador();
            }
            else{                                                //Caso
contrario decrementamos os segundos e colocamos os milisegundos a 9
                segundos--;
                miliSegundos = 9;
            }
        }
        else{                                                    //Mas caso os
milisegundos nao sejam 0 decrementamos milisegundos
            miliSegundos--;
        }
        atualizarDisplays(segundos, miliSegundos);    //Chamamos a funcao que
atualiza o display conforme o tempo dado
    }else if(congelarTimer){            //Se estamos no estado de congelar o
timer a 5s entao mantemos o display com 5 segundos e 0 milisegundos
        atualizarDisplays(5, 0);
    }
    else{                                                    //Caso
nao se espere uma resposta quer dizer que ja a temos logo vamos fazer o ciclo
onde mostramos a resposta e o tempo restante para responder
        cicloMostrarResposta--;
        if(cicloMostrarResposta >= (numeroCiclosMostrarResposta/2)){  //Se o
ciclo estiver entre 10 e 20 (1 e 2 segundos) mostramos o tempo restante
            atualizarDisplays(segundosRestantes, miliSegundosRestantes);
        }
        else{  //Caso contrario mostramos a resposta dada atraves da funcao
encarregue disso
            if(cicloMostrarResposta == 0){
                cicloMostrarResposta = numeroCiclosMostrarResposta;
            }
            atualizarDisplayResposta(Resposta); //Esta funcao ira colocar no
display -. e a resposta dada pelo utilizador
        }
    }
```

```
}
}

void atualizarDisplays(int segundos, int miliSegundos){  //Para colocar o tempo no display esta funcao recebee os segundos e os milisegundos e faz um switch para cada um deles
   switch(segundos){                                     //No caso dos segundos verifica que valor tem e coloca-o com o bit do . ativo para ficar 0. por exemplo.
        case 0:
              P1 = 0x40;                                 //Estes valores são apenas os equivalentes em hexadecimal dos codigos em binario nas tabelas do enunciado para cada valor
              break;
        case 1:
              P1 = 0x79;
              break;
        case 2:
              P1 = 0x24;
              break;
        case 3:
              P1 = 0x30;
              break;
        case 4:
              P1 = 0x19;
              break;
        case 5:
              P1 = 0x12;
              break;
        case 6:
              P1 = 0x02;
              break;
        case 7:
              P1 = 0x78;
              break;
        case 8:
              P1 = 0x00;
              break;
        case 9:
              P1 = 0x10;
              break;
        default:
              break;
   }
```

```
   switch(miliSegundos){                    //Os milisegundos sao iguais aos
segundos mas com o bit do . desativado para ficar com o resultado de 1.0 por
exemplo
        case 0:
                P2 = 0xC0;
                break;
        case 1:
                P2 = 0xF9;
                break;
        case 2:
                P2 = 0xA4;
                break;
        case 3:
                P2 = 0xB0;
                break;
        case 4:
                P2 = 0x99;
                break;
        case 5:
                P2 = 0x92;
                break;
        case 6:
                P2 = 0x82;
                break;
        case 7:
                P2 = 0xF8;
                break;
        case 8:
                P2 = 0x80;
                break;
        case 9:
                P2 = 0x90;
                break;
        default:
                break;
   }
   return;
}
void terminarcontador(){          //Esta funcao e chamada quando o tempo
terminou e neste caso vamos...
   esperarResposta = 0;           //Deixar de esperar uma resposta visto que ja
acabaou o tempo
   Resposta = '0';                                  //Atribuir uma resposta nula a
resposta
   segundosRestantes = 0;    //Deixar os segundos e milisegundos restantes a 0
```

```
   miliSegundosRestantes = 0;
   atualizarDisplays(segundosRestantes, miliSegundosRestantes);    //Atualizar o
display com estes valores
   return;
}

void atualizarDisplayResposta(char Resposta){        //PAra atualizar o display
com a resposta e muito semelhante a atualizar com os numeros
   P1 = 0x3F;        //Equivalente a -. que e comum a todas as respostas logo
fica fora do switch por conveniencia
   switch(Resposta){                              //VAmos fazer um switch com a
resposta
        case 'A':                                          //E vamos colocar o valor
correspondente a letra respondida no display 2, tal como acontecia na funcao de
mostrar o tempo
                P2 = 0x88;
                break;
        case 'B':
                P2 = 0x83;
                break;
        case 'C':
                P2 = 0xC6;
                break;
        case 'D':
                P2 = 0xA1;
                break;
        case '0':
                P2 = 0xBF;
                break;
        default:
                break;
   }
   return;
}

void botao1(void) interrupt 0{            //Quando o botao 1 e pressionado esta
interrupcao e ativa
   if(esperarResposta == 0 && congelarTimer == 0){       //Se nao esperamos
uma resposta  e o temporizador nao esta congelado entao vamos congelar o
timer e resetar o contador do ciclo de 100ms pois vamos resetar o timer
        tempoInicial = ciclo100ms;
        congelarTimer = 1;
        segundos = 5;
        miliSegundos = 0;
   }
```

```
   else if(esperarResposta == 0 && congelarTimer ==1){    //Se nao esperamos uma resposta mas o tempo ja esta congelado entao vamos comecar a esperar uma resposta e vamos descongelar o timer de forma a permitir o temporizador decrescer
        tempoInicial = ciclo100ms;       //Reiniciamos sempre o tempoInicial pois estamos a resetar o timer
        esperarResposta = 1;
        congelarTimer = 0;
   }
   else if(esperarResposta == 1){       //Se esperamos uma resposta entao ao clicar no botao 1 o temporizador reseta para os 5 segundos e atualiza o display e ja nao esperamos uma resposta, apenas input do botao 1
        tempoInicial = ciclo100ms;
        segundos = 5;
        miliSegundos = 0;
        atualizarDisplays(segundos, miliSegundos);
        congelarTimer = 1;
        esperarResposta = 0;
   }
}

void botaoRepostas(void) interrupt 2{ //Se o botao com as portas and para todos os botoes de resposta for ativo quer dizer que alguma resposta foi pressionada
   if(esperarResposta){    //Neste caso se esperamos uma resposta entao deixamos de o fazer uma vez que acabamos de receber uma
        esperarResposta = 0;
        segundosRestantes = segundos;        //o tempo restante e o atual na hora que o botao foi clicado
        miliSegundosRestantes = miliSegundos;
        cicloMostrarResposta = numeroCiclosMostrarResposta;

        if(BA == 0){                         //Verificamos qual dos botoes e que esta ativo e atribuimos a resposta na variavel correspondente
            Resposta = 'A';    //Nota: Caso dois botoes sejam clicados simultanemante, no exato mesmo instante, o que e extremamente improvavel e irrealista, a resposta dada sera por preferencia de ordem alfabetica
            return;
        }
        else if(BB == 0){
            Resposta = 'B';
            return;
        }
        else if(BC == 0){
            Resposta = 'C';
```

```
                return;
        }
        else if(BD == 0){
                Resposta = 'D';
                return;
        }
        else{
                Resposta = '0';
                return;
        }
   }
   else{
        return;
   }
}
```

Em Assembly

```
; Definição das constantes

; Display 1
D1 EQU P1

; Display 2
D2 EQU P2


; Inputs
Inputs EQU P3
B1 EQU P3.2 ; Botão 1
BResposta EQU P3.3 ; PIN de qualquer resposta
BA EQU P3.4 ; Botão A
BB EQU P3.5 ; Botão B
BC EQU P3.6 ; Botão C
BD EQU P3.7 ; Botão D

; Timer
TempoL EQU 0xF0 ;F0D8 é 61656 em decimal que é quando da overflow no timer
TempoH EQU 0xD8
TempoInicial EQU 0x0A ;

; Função principal
ORG 0000h
JMP Inicio
```

```
; Tratar interrupção externa 0
CSEG AT 0003h
JMP InterrupcaoBotao1

; Tratar interrupção externa 1
CSEG AT 0013h
JMP InterrupcaoBotaoResposta

; Tratar a interrupção de temporização 0, para contar 10ms
CSEG AT 000Bh
JMP InterrupcaoTemp0

;Para guiar vou definir que as variaveis sao as seguintes:
;R7 sera o contador de 1 segundo que muda entre a resposta e o tempo restante
;R3 sera uma variavel booleana que nos diz se estamos a espera de uma resposta do utilizador ou nao
;R4 sera a variavel dos milisegundo a mostrar
;R5 sera o mesmo que R4 mas para os segundos
;R1 sera o valor dos segundos que faltavam para o utilizador dar a resposta
;R2 sera os milisegundos que faltavam para acabar o timer
;R0 sera o tempo inicial
;R6 sera a resposta dada pelo utilizador
;O A sera variavel assistente nas funcoes

CSEG AT 0050h
Inicio:
        MOV SP, #07h ; Endereço inicial do stack pointer
        CALL Inicializacoes ; Chama a rotina de inicializações
        CALL PrioridadeInterrupcoes ; Chama a rotina que define as prioridades das interrupções
        CALL AtivaInterrupcoes ; Chama a rotina que ativa as interrupções necessárias
        CALL AtivaTemporizador ; Chama a rotina que ativa o temporizador
        SJMP Principal ; Vai para a função principal

Principal:
        SJMP Principal ; Loop principal infinito

Inicializacoes:
        MOV Inputs, #0FFh ; Configurar P3 como input
        MOV D1, #0FDh ; Colocar os displays como "-". FD é 11111101 em binário, valor que coloca o display no valor pretendido
        CLR C ; Limpar o carry
```

```
    MOV R4, #0
    MOV R1, #05h
    MOV R2, #00h
    MOV R3, #0
    MOV R5, #0
    MOV R6, #0eh
    MOV R7, #0

        RET

AtivaInterrupcoes:
        MOV IE, #10000111b ; Ativa interrupções timer 0 e externas 1 (P3.3) e 0 (P3.2)
        SETB IT0 ; A interrupção externa será detectada na transição descendente do clk
        RET

PrioridadeInterrupcoes:
        MOV IP, #00000001b ; O timer 0 tem a maior prioridade
        RET

AtivaTemporizador:
        MOV TMOD, #00000001b ; Ativa o temporizador no modo de leitura de 16 bits
        MOV TL0, #TempoL ; Valor do byte menos significativo
        MOV TH0, #TempoH ; Valor do byte mais significativo
        MOV R0, #TempoInicial ; Indica o número de contagens de 10ms que terão de ser realizadas para fazer alguma alteração, neste caso 10, 100 milissegundos, ou 0.1s
        SETB TR0 ; Ativa o temporizador 0, para fazer contagens de 10ms
        RET

; Tratamento da interrupção externa 1, ou seja, quando algum dos botões de resposta foi apertado
InterrupcaoBotaoResposta:
    MOV A, R3
        JNZ ObterResposta ; Se estamos à espera de uma resposta e o utilizador pressiona uma resposta, chamamos a rotina que obtém a resposta
        ; Se não estamos à espera de uma resposta não fazemos nada
        RETI

ObterResposta:
        ; Vamos comparar com todos os pinos de resposta para ver qual botão de resposta foi pressionado
        JNB BA, RespostaA
```

```
        JNB BB, RespostaB
        JNB BC, RespostaC
        JNB BD, RespostaD
        RETI

RespostaA:  ;Se a resposta A for seleciona, coloca-se na variavel relacionada a resposta dada o valor A para conveniencia
        MOV R6, #0Ah
        MOV R3, #0          ;Deixamos de esperar uma resposta
    MOV A, R5          ;Copiamos o tempo restante para as suas variáveis relacionadads
    MOV R1, A
    MOV A, R4
    MOV R2, A
     MOV R7, #13h    ;Resetamos o timer de ciclo entre mostrar resposta e tempo restante
        RETI

RespostaB:                  ;Repetimos o mesmo processo para cada uma das diferentes respostas
        MOV R6, #0Bh
        MOV R3, #0
    MOV A, R5
    MOV R1, A
    MOV A, R4
    MOV R2, A
    MOV R7, #13h
        RETI

RespostaC:
        MOV R6, #0Ch
        MOV R3, #0
    MOV A, R5
    MOV R1, A
    MOV A, R4
    MOV R2, A
    MOV R7, #13h

        RETI

RespostaD:
        MOV R6, #0Dh
        MOV R3, #0
    MOV A, R5
    MOV R1, A
```

```
    MOV A, R4
    MOV R2, A
    MOV R7, #13h
        RETI

; Tratamento da interrupção externa 0, ou seja, do botão 1
InterrupcaoBotao1:
    MOV A, R5
        CJNE A, #05, IniciarContador ; Se não estamos à espera de uma resposta então temos de iniciar o contador
        JMP     ComecarContador;Se estamos à espera de uma resposta não faz nada
        RETI

IniciarContador:     ;Quando iniciamos o contador colocamos os segundos a 5 e os milisegundos a 0 e alteramos a variavel de esperar resposta para true
        MOV R0, TempoInicial
    MOV R4, #00h
        MOV R5, #05h
    MOV R3, #00h
    MOV R6, #0eh
    CALL MostrarTimer
    MOV R1, #05h
    MOV R2, #00h
        RETI
ComecarContador:          ;Quando iniciamos o contador colocamos os segundos a 5 e  os milisegundos a 0 e alteramos a variavel de esperar resposta para true
    MOV R0, #TempoInicial
        MOV R3, #01h

        RETI

; Tratamento da Interrupção de overflow na contagem do timer0
InterrupcaoTemp0:
        MOV TL0, #TempoL ; Inicia a contagem de 10ms
        MOV TH0, #TempoH ; TL0 e TH0 guardam o número para iniciar contagens
        DJNZ R0, FimIT0 ; Decrementa a variável para fazer contagens de 0.1s e se não for 0 continua a contagem
        MOV R0, #TempoInicial ; Se for 0 significa que já passaram 100ms então fazemos o que é necessário
        MOV A, R3     ;Vemos se estamos a espera de uma resposta, pois se nao entao estamos a mostrar a ultima resposta dada
    JZ RespostaDada
```

```
    CALL MostrarTimer ;Se não foi dada uma resposta metemos no display o tempo atual
   MOV A, R4          ;Mas se estamos a espera de resposta vemos se os milisegundo sao 0
        JZ SubSeg
        DEC R4
        RETI

SubSeg:                    ;Se os milisegundos forem 0 entao subtraimos um segundo e colocamos os milisegundos como 9
        MOV A, R5    ;Se os segundo tbm forem 0 entao terminamos o contador
        JZ TerminarContador
        DEC R5              ;Caso contrario fazemos o esperado
        MOV R4, #09h
        RETI

TerminarContador:
   MOV R6, #0       ;A resposta dada é nenhuma, de forma a mostrar -.- No display
        MOV R3, #00h     ;Neste caso ja nao estamos a espera de uma resposta pois acabou o tempo
   MOV R1, #0       ;os segundo e milisegundos restantes sao colocados a 0
   MOV R2, #0
        RETI
RespostaDada:
   MOV A, R7         ;Se não esperamos resposta então temos alguma resposta dada ou 0.
   CLR C
   SUBB A, #0Ah
   JNC MostrarTempo;Se R7 for maior que 10 mostrar o tempo
   JMP MostrarResposta ;Se nao mostrar REsposta

MostrarResposta:
   CJNE R7, #00h, continuarMostrarResposta
   JMP resetarCiclo
   continuarMostrarResposta:
   DEC R7                  ;Deccrementamos o contador
   MOV A, R6               ;Vamos buscar o valor de resposta guardado
     CJNE A, #0ah, naoA      ;Se for A mostramos -A no display. Os bits ficam invertidos dos dados no enunciado uma vez que aqui comeca do bit mais significativo para o menos, o contrario do dado na tabela
   MOV D1, #00111111b
   MOV D2, #10001000b
   naoA:
   CJNE A, #0bh, naoB    ;Se for B mostramos -B
```

```
   MOV D1, #00111111b
   MOV D2, #10000011b
   naoB:
   CJNE A, #0ch, naoC    ;Se for C mostramos -C
   MOV D1, #00111111b
   MOV D2, #11000110b
   naoC:
   CJNE A, #0dh, naoD    ;Se for D mostramos -D
   MOV D1, #00111111b
   MOV D2, #10100001b
   naoD:
    CJNE A, #0eh, naoe ;Se a resposta é e, isto significa que o timer foi resetado então continuamos a mostrar os 5 segundos restantes até o utilizador alterar R3 e começar a decrementar o timer
   MOV D1, #00010010b
   MOV D2, #11000000b
   naoe:
   CJNE A, #0, nao0_Resposta ;Se nao for nenhum mostramos -.-
   MOV D1, #00111111b
   MOV D2, #10111111b
   nao0_Resposta:
   RETI
   resetarCiclo:
   MOV R7, #13h
   RETI
FimIT0:
   RETI

MostrarTempo:
   DEC R7                    ;Deccrementamos o contador
   MOV A, R2                 ;Vemos os milisegundos guardados quando o utilizador deu a resposta
   CJNE A, #0, nao0    ;Se nao for 0 passa a frente
   MOV D2, #11000000b    ;Se for colocamos no segundo display o 0
   nao0:
     CJNE A, #1, nao1      ;Fazemos o mesmo ate descobrirmos o valor dos milisegundos a colocar no display2
   MOV D2, #11111001b
   nao1:
   CJNE A, #2, nao2
   MOV D2, #10100100b
   nao2:
   CJNE A, #3, nao3
   MOV D2, #10110000b
   nao3:
```

```
    CJNE A, #4, nao4
    MOV D2, #10011001b
    nao4:
    CJNE A, #5, nao5
    MOV D2, #10010010b
    nao5:
    CJNE A, #6, nao6
    MOV D2, #10000010b
    nao6:
    CJNE A, #7, nao7
    MOV D2, #11111000b
    nao7:
    CJNE A, #8, nao8
    MOV D2, #10000000b
    nao8:
    CJNE A, #9, nao9
    MOV D2, #10010000b
    nao9:
    MOV A, R1                    ;Agora fazemos o mesmo para os segundos so
que metemos o numero com o ponto a frente. Exemplo: 1.
    CJNE A, #0, nao0_2
    MOV D1, #01000000b
    nao0_2:
    CJNE A, #1, nao1_2
    MOV D1, #01111001b
    nao1_2:
    CJNE A, #2, nao2_2
    MOV D1, #00100100b
    nao2_2:
    CJNE A, #3, nao3_2
    MOV D1, #00110000b
    nao3_2:
    CJNE A, #4, nao4_2
    MOV D1, #00011001b
    nao4_2:
    CJNE A, #5, nao5_2
    MOV D1, #00010010b
    nao5_2:
    CJNE A, #6, nao6_2
    MOV D1, #00000010b
    nao6_2:
    CJNE A, #7, nao7_2
    MOV D1, #01111000b
    nao7_2:
    CJNE A, #8, nao8_2
```

```
MOV D1, #00000000b
nao8_2:
CJNE A, #9, nao9_2
MOV D1, #00010000b
nao9_2:
RETI

MostrarTimer:
MOV A, R4                ;Vemos os milisegundos guardados quando o utilizador deu a resposta
CJNE A, #0, nao0Timer    ;Se nao for 0 passa a frente
MOV D2, #11000000b       ;Se for colocamos no segundo display o 0
nao0Timer:
 CJNE A, #1, nao1Timer   ;Fazemos o mesmo ate descobrirmos o valor dos milisegundos a colocar no display2
MOV D2, #11111001b
nao1Timer:
CJNE A, #2, nao2Timer
MOV D2, #10100100b
nao2Timer:
CJNE A, #3, nao3Timer
MOV D2, #10110000b
nao3Timer:
CJNE A, #4, nao4Timer
MOV D2, #10011001b
nao4Timer:
CJNE A, #5, nao5Timer
MOV D2, #10010010b
nao5Timer:
CJNE A, #6, nao6Timer
MOV D2, #10000010b
nao6Timer:
CJNE A, #7, nao7Timer
MOV D2, #11111000b
nao7Timer:
CJNE A, #8, nao8Timer
MOV D2, #10000000b
nao8Timer:
CJNE A, #9, nao9Timer
MOV D2, #10010000b
nao9Timer:
MOV A, R5                ;Agora fazemos o mesmo para os segundos so que metemos o numero com o ponto a frente. Exemplo: 1.
CJNE A, #0, nao0_2TImer
MOV D1, #01000000b
```

```
    nao0_2Timer:
    CJNE A, #1, nao1_2Timer
    MOV D1, #01111001b
    nao1_2Timer:
    CJNE A, #2, nao2_2Timer
    MOV D1, #00100100b
    nao2_2Timer:
    CJNE A, #3, nao3_2Timer
    MOV D1, #00110000b
    nao3_2Timer:
    CJNE A, #4, nao4_2Timer
    MOV D1, #00011001b
    nao4_2Timer:
    CJNE A, #5, nao5_2Timer
    MOV D1, #00010010b
    nao5_2Timer:
    CJNE A, #6, nao6_2Timer
    MOV D1, #00000010b
    nao6_2Timer:
    CJNE A, #7, nao7_2Timer
    MOV D1, #01111000b
    nao7_2Timer:
    CJNE A, #8, nao8_2Timer
    MOV D1, #00000000b
    nao8_2Timer:
    CJNE A, #9, nao9_2Timer
    MOV D1, #00010000b
    nao9_2Timer:
    RET

END ; Final do programa
```