UNIVERSIDADE da MADEIRA

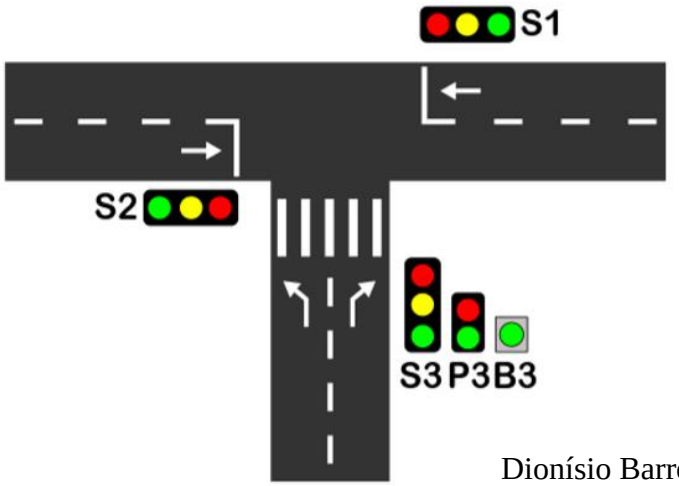
**Faculdade de Ciências Exatas e da Engenharia**

**2019/2020**

**Arquitetura de Computadores**

Licenciatura em Engenharia Informática

# 3º Projeto – Gestão de semáforos numa interseção rodoviária




**Docentes:**

Dionísio Barros, Nuno Ferreira

Sofia Inácio, Pedro Camacho

**Trabalho realizado por:**

Diego Briceño (nº 2043818)

Rúben Rodrigues (nº 2046018)

Funchal, 24 de maio de 2020

# Índice

# 1. Introdução

Este relatório apresentará os objetivos relacionados ao terceiro trabalho prático da unidade curricular de Arquitetura de Computadores, assim como o seu desenvolvimento, discussão de resultados e a conclusão a que os alunos chegaram.

A linguagem C é uma linguagem de alto nível independente da arquitetura do computador onde os programas correm. O compilador consegue gerar o código-máquina adequado para o processador desse computador, a partir da linguagem de alto nível.

A implementação prévia de um programa numa linguagem de alto nível permite facilitar a implementação do mesmo em linguagem Assembly.

O programa desenvolvido foi criado em linguagem C e linguagem Assembly para o processador 8051 e, para efetuar a simulação do mesmo, utilizou-se o programa Keil uVision e a ferramenta de simulação Multisim.

# 2. Objetivos

Este trabalho prático tem como objetivos elaborar fluxogramas que permitem ser o ponto de partida para a criação do programa e estudar as linguagens C e Assembly para o processador 8051 e a configuração e programação de interrupções no mesmo.

Neste trabalho pretendeu-se desenvolver um programa em linguagem Assembly e C para o processador 8051, capaz de realizar a gestão de quatro semáforos numa interseção rodoviária.

Os quatro semáforos correspondem a três semáforos para automóveis denominados por S1, S2 e S3, e a um semáforo P3 correspondente a um semáforo para peões. Cada semáforo tem três cores (verde, amarelo e vermelho), com exceção do semáforo para peões que apenas tem duas cores (verde e vermelho). Os semáforos para automóveis permanecem com a luz verde ligada durante 10 segundos, luz amarela durante 5 segundos e luz vermelha durante 15 segundos. O semáforo P3 está verde quando S3 está vermelho, e está vermelho quando S3 está verde ou amarelo. Também, antes de P3 mudar para vermelho, o seu estado fica intermitente de 1 em 1 segundos, durante 5 segundos. Cada semáforo tem apenas uma luz de uma cor ligada de cada vez.

O botão B3 corresponde a um botão que permite aos peões solicitar a colocação de P3 a verde. Isso só pode acontecer quando S3 está a verde e o botão é pressionado, colocando o semáforo S3 a amarelo durante 5 segundos, antes de ficar vermelho.

# 3. Desenvolvimento

Para desenvolver o programa descrito anteriormente, decidiu-se começar por implementá-lo em linguagem C, visto que a linguagem C se aproxima mais à nossa linguagem e é mais fácil para compreender. De seguida, efetuou-se o mapeamento da linguagem C para a linguagem Assembly, pois esta linguagem permite reduzir o tempo de execução do programa, permitindo obter uma melhor eficácia em relação ao processamento dos dados pelo processador.

Para a elaboração do programa em linguagem C, foi implementada a função **Inicializar**, responsável por ativar as interrupções globais, da interrupção do timer 0 e da interrupção externa 0, configurar o modo 2 (8 bits - autoreload) no timer e o tempo de contagem para 250 microssegundos, iniciar o timer 0 e definir a interrupção externa 0 para ser acionada na *falling edge*.

Ao iniciar o programa, é feita a chamada do método Inicializar. A variável **contaSegundos** corresponde ao número de segundos que passaram desde o início do ciclo. A variável **auxContaSegundos**, por sua vez, corresponde ao número de vezes que ocorre overflow no timer. O overflow no timer ocorre quando passam 250 microssegundos. Desta forma, foi possível realizar a contagem do tempo em segundos. Cada vez que a variável auxContaSegundos chegasse a 4000 (1000000 microssegundos), a variável contaSegundos era incrementada uma unidade, correspondente a um segundo passado.

Para realizar a mudança das cores das luzes dos semáforos, pensou-se numa espécie de máquina de estados finita, em que cada estado correspondia a cada possibilidade de luzes ativas ao mesmo tempo, sendo a variável contaSegundos a responsável por decidir o estado seguinte.

Quando a variável **contaSegundos** é:

- **0:** as luzes verdes dos semáforos S1, S2 e P3 e a luz vermelha do semáforo S3 são ligadas (estado inicial).
- **10:** as luzes amarelas dos semáforos S1 e S2 são ligadas.
- **Entre 10 e 14**: a luz verde do semáforo P3 está intermitente.
- **15**: as luzes vermelhas dos semáforos S1, S2 e P3 e a luz verde do semáforo S3 são ligadas.
- **25**: a luz amarela do semáforo S3 é ligada.
- **30:** é feito o reset da variável a “0“para recomeçar o ciclo, ou seja, voltar ao estado inicial (0 segundos).

A variável **auxMudarSemaforos** permite que, quando está a "1", sejam realizadas estas mudanças uma única vez, até o valor de contaSegundos ser alterado, sendo posta a "0" após as verificações dos estados.

Quando o botão B3 é pressionado, se S3 está a verde, a variável contaSegundos é colocada a 25, pois corresponde ao estado em que a luz amarela do semáforo S3 é ligada.

A implementação em Assembly não foi muito difícil aplicando os conhecimentos adquiridos sobre mapear elementos da linguagem C para linguagem Assembly e tendo em conta a simplicidade do programa desenvolvido em linguagem C. Para implementar o programa criou-se 3 rotinas: **Inicializacoes**, **AtivaInterrupcao** e **AtivaTemporizador**.

A rotina **Inicializacoes** é responsável por fazer o "reset" do acumulador, dos registos usados ao longo do programa e dos periféricos de saída. A rotina **AtivaInterrupcao**, como o nome indica, inicializa as interrupções em geral e as interrupções do timer 0 e externa 0, sendo esta última ativa na *falling edge*.

A rotina **AtivaTemporizador** define o timer 0 no modo de funcionamento 2 (8 bit auto-reload) de modo a contar-se 250 microssegundos entre overflows do timer, e inicializa os registos auxiliares à contagem do tempo (R0, que representa a variável **auxMudarSemaforos** do programa em C, R1 que conta o número de overflows no timer para se perfazer 10 milissegundos e R2 que conta quantas vezes R1 diminui a 0, perfazendo assim um segundo, para incrementar-se o acumulador).

Como já se referiu, o programa em Assembly é extremamente semelhante ao de C, no ciclo Principal verificamos se R0 está a 1, o que indica que o acumulador (que representa a variável **contaSegundos** do programa em C) sofreu alteração, e se tal se verificar então procede-se a verificar em que parte do ciclo o programa encontra-se (através do valor do acumulador) e efetuar as necessárias alterações aos LED's, sendo R0 posto a 0 até que o acumulador altere de valor novamente.

## 4. Discussão de Resultados

Os resultados corresponderam com o esperado, e o que era pedido como objetivo do programa. Os semáforos S1 e S2 permanecem ligados a verde durante 10 segundos, depois dos quais ficam amarelos durante 5 e por fim ficam vermelhos. Os semáforos S1 e S2 encontram-se sincronizados entre si e o semáforo S3 não entra em conflito com estes, isto é, se S3 está verde ou amarelo S1 e S2 estão vermelhos e vice-versa.

O semáforo dos peões também não entra em conflito com o semáforo S3 (se o semáforo dos peões está verde o semáforo S3 está vermelho e se S3 está verde ou amarelo, P3 está vermelho) e implementar a intermitência do semáforo dos peões quando S1 e S2 estão a amarelo tornou-se simples tendo em conta o raciocínio explicado no desenvolvimento do programa.

Por fim, a implementação do botão B3 para pôr o semáforo S3 a amarelo de modo a P3 ficar verde 5 segundos após o botão ser pressionado também funcionou e tornou-se simples com o raciocínio explicado no desenvolvimento.

## 5. Conclusão

Concluindo, acreditamos que os objetivos do trabalho foram atingidos, em ambas as linguagens de programação e, como o programa é muito visual, a ferramenta MultiSim ajudou muito a visualizar o correto funcionamento do programa.

A realização do trabalho não provou ser muito complexa visto que ambos os alunos compreendiam bem o funcionamento do processador e a elaboração prévia dos fluxogramas facilitou a compreensão do programa e como este deveria funcionar. Os alunos ficaram satisfeitos com o programa desenvolvido, que se considera simples e eficiente.

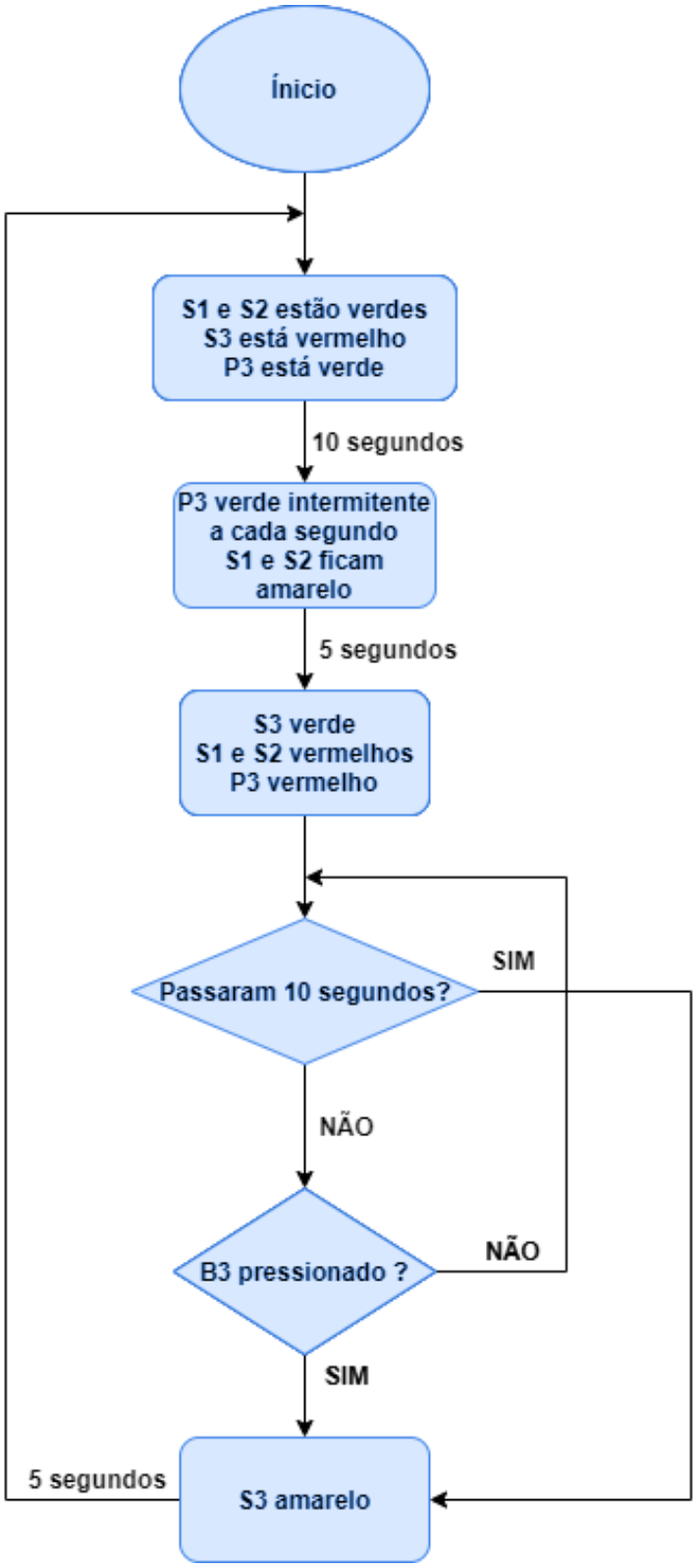
O mapeamento para a linguagem Assembly a partir do programa em C foi a parte mais complicada, mas dada a simplicidade do programa em C o mapeamento ficou facilitado graças aos conhecimentos sobre mapeamento de instruções em C para Assembly fornecidos aos alunos.

## 6. Bibliografia

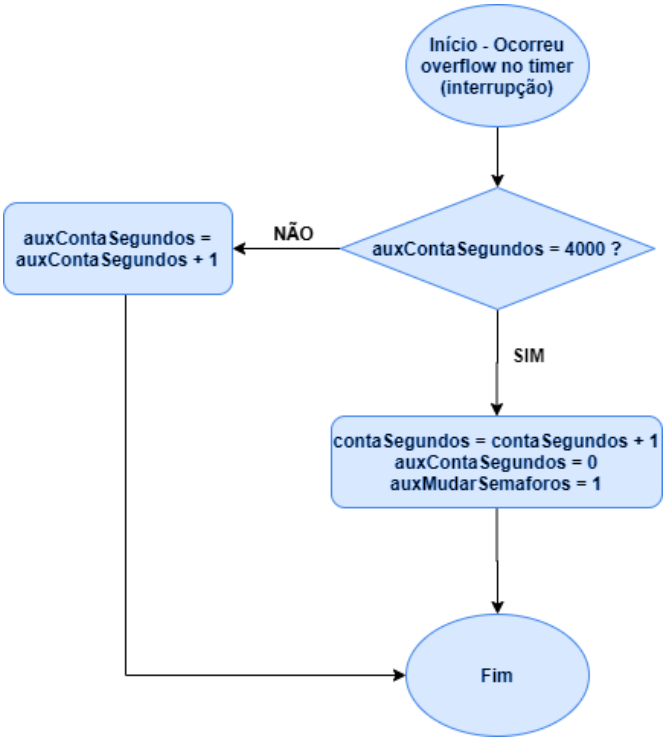
J. Delgado e C. Ribeiro, Arquitetura de Computadores, FCA - Editora de Informática, 2010.

# 7. Anexo A - Fluxogramas

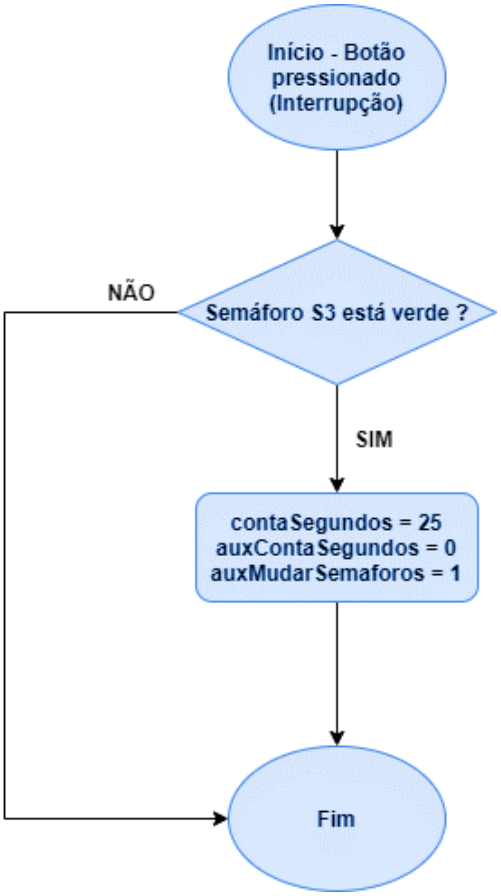
## 7.1. Fluxograma do programa principal



## 7.2. Fluxograma da interrupção do timer




## 7.3. Fluxograma da interrupção externa



# 8. Anexo B – Código

## 8.1. Código em linguagem C

```
#include <reg51.h>

//sbit S1_Vermelho = P1^0; //porta de saída para a luz vermelha do semáforo S1
//sbit S1_Amarelo = P1^1; //porta de saída para a luz amarela do semáforo S1
//sbit S1_Verde = P1^2; //porta de saída para a luz verde do semáforo S1

//sbit S2_Vermelho = P1^3; //porta de saída para a luz vermelha do semáforo S2
//sbit S2_Amarelo = P1^4; //porta de saída para a luz amarela do semáforo S2
//sbit S2_Verde = P1^5;//porta de saída para a luz verde do semáforo S2

//sbit S3_Vermelho = P2^2; //porta de saída para a luz vermelha do semáforo S3
//sbit S3_Amarelo = P2^1; //porta de saída para a luz amarela do semáforo S3
//sbit S3_Verde = P2^0; //porta de saída para a luz verde do semáforo S3

//sbit P3_Vermelho = P2^4; //porta de saída para a luz vermelha do semáforo P3
sbit P3_Verde = P2^3; //porta de saída para a luz verde do semáforo P3

sbit B3= P3^2; //porta de entrada para pressionar o botão B3

int contaSegundos = 0; //esta variável é igual ao número de segundos que passaram desde o início do ciclo
int auxContaSegundos = 0; //esta variável conta o número de vezes que ocorre overflow no timer (que ocorre quando passam 250 microssegundos)
bit auxMudarSemaforos = 1; //variável "booleana" para evitar estar sempre a definir os semáforos

//****************IMPORTANTE**********************
//0 - LED ligado
//1 - LED desligado
int S1_S2_Verdes=0x1B; // 0001 1011 em binário
int S3_Vermelho_P3_Verde=0x13; // 0001 0011 em binário
int S1_S2_Amarelos=0x2D; // 0010 1101 em binário
int S1_S2_Vermelhos=0x36; // 0011 0110 em binário
int S3_Verde_P3_Vermelho=0x0E; // 0000 1110 em binário
int S3_Amarelo_P3_Vermelho=0x0D; // 0000 1101 em binário

void Inicializar(void){
  //Ativar as interrupções globais, do timer 0 e da interrupção externa 0
  EA = 1;
  ET0 = 1;
  EX0 = 1;
  //Configurar o timer no modo 2 (8 bits autoreload)
  TMOD &= 0xF0;
  TMOD |= 0x02;
  //Configurar o tempo de contagem para 250 microssegundos
  TH0 = 0x06;
  TL0 = 0x06;
  //Iniciar timer 0 e definir a interrupção para ser acionada na falling edge
  TR0 = 1;
  IT0 = 1;
}
void Timer0_ISR(void) interrupt 1{
  //auxContaSegundos = 1 -> 250 microssegundos
  //auxContaSegundos = 4 -> 1 milisegundo
  //auxContaSegundos = 4000 -> 1 segundo
  if(auxContaSegundos == 4000){ //quando esta variável chegar a 4000, significa que fez-se um segundo
    contaSegundos++;
    auxContaSegundos = 0; //reset da variável
    auxMudarSemaforos = 1; //como o contaSegundos aumentou, definimos isto a 1 para efetuar as verificações
  }else{ //se não chegou, incrementa-se a variável
    auxContaSegundos++;
  }
}
```

```
61
62  void External0_ISR(void) interrupt 0 {
63    if (P2 == S3_Verde_P3_Vermelho){ //verifica-se se o sinal verde do semáforo S3 está ligado
64      contaSegundos = 25; //se estiver define-se o contasegundos a 25 pois é neste momento que se põe
    o semáforo 3 a amarelo como pretendido
65      auxContaSegundos = 0; //reset da variável que conta os overflows do timer, de modo a passar 5
    segundos a partir do botão ser clicado
66      auxMudarSemaforos = 1; //como o contaSegundos mudou, definimos isto a 1 para efetuar as
    verificações
67    }
68  }
69
70  void main(void){
71    Inicializar();
72    for(;;){
73      if (auxMudarSemaforos==1){
74        if(contaSegundos == 0){ //parte inicial do ciclo, S1 e S2 verdes, S3 vermelho, P3 verde
75          P1 = S1_S2_Verdes;
76          P2 = S3_Vermelho_P3_Verde;
77        }
78        if(contaSegundos == 10){ //depois de 10 segundos, S1 e S2 ficam amarelos
79          P1=S1_S2_Amarelos;
80        }
81        if(contaSegundos > 10 &&contaSegundos <15){ //entre os 10 e 15 segundos, o semáforo P3 deve
    ficar verde intermitente, a cada segundo
82          P3_Verde = ~P3_Verde;
83        }
84        if(contaSegundos == 15){ //depois de 15 segundos, S1 e S2 ficam vermelhos, S3 fica verde e P3
    fica vermelho
85          P1=S1_S2_Vermelhos;
86          P2=S3_Verde_P3_Vermelho;
87        }
88        if(contaSegundos == 25){ //depois de 25 segundos, S3 fica amarelo
89          P2=S3_Amarelo_P3_Vermelho;
90        }
91        auxMudarSemaforos=0; //mudar isto para 0 para não fazer as verificações até o contaSegundos
    incrementar
92        if(contaSegundos == 30){ //depois dos 30 segundos, faz-se reset da variável para voltarmos ao
    início do ciclo e tratar dos semáforos
93          contaSegundos = 0;
94          auxMudarSemaforos = 1;
95        }
96      }
97    }
98  }
```

## 8.2. Código em linguagem Assembly

```
;constantes
TempoL EQU 0x06
TempoH EQU 0x06

;S1_Vermelho EQU P1.0
;S1_Amarelo EQU P1.1
;S1_Verde EQU P1.2

;S2_Vermelho EQU P1.3
;S2_Amarelo EQU P1.4
;S2_Verde EQU P1.5

;S3_Vermelho EQU P2.2
;S3_Amarelo EQU P2.1
S3_Verde EQU P2.0

;P3_Vermelho EQU P2.4
P3_Verde EQU P2.3

B3 EQU P3.2

S1_S2 EQU P1
S3_P3 EQU P2

S1_S2_Verdes EQU 00011011b
S3_Vermelho_P3_Verde EQU 00010011b
S1_S2_Amarelos EQU 00101101b
S1_S2_Vermelhos EQU 00110110b
S3_Verde_P3_Vermelho EQU 00001110b
S3_Amarelo_P3_Vermelho EQU 00001101b

Zero EQU 0 ;para os semáforos indica que o semáforo está ligado
Um EQU 1 ;para os semáforos indica que o semáforo está desligado
Dez EQU 10
Quinze EQU 15
VinteECinco EQU 25
Quarenta EQU 40
Cem EQU 100

; A - contador de tempo em segundos dos semaforos para os carros
; R0 - variavel auxiliar para só efetuar mudanças ao semáforos quando A mudar
; R1 - numero de contagens de 250 microsegundos para fazer 10 milisegundos (40)
; R2 - numero de contagens de 10 milisegundos para fazer 1 segundo (100)

CSEG AT 0000h
    JMP Inicio ;depois de um reset efetuar as inicializações

CSEG AT 0003h
    JMP InterrupcaoExterior0

CSEG AT 000Bh
    JMP InterrupcaoTempo0

CSEG AT 0020h
Inicio:
    MOV SP, #7
    CALL Inicializacoes
    CALL AtivaInterrupcao
    CALL AtivaTemporizador

```

```
61  Principal:
62      CJNE R0,#Um,Principal ;se R0 for um, quer dizer que A incrementou por isso faz-se as
    verificações; se for 0, não se faz as verificações
63      JNZ ContaSegundos_10 ;comparar A com 0, saltar se não é zero
64      ;Pôr S1 e S2 verdes, S3 vermelho e P3 verde
65      MOV S1_S2,#S1_S2_Verdes
66      MOV S3_P3,#S3_Vermelho_P3_Verde
67      JMP ResetR0
68  ContaSegundos_10: ;comparar A com 10, saltar se não é igual a 10
69      CJNE A, #10, ContaSegundos_entre_10e15
70      ;Pôr S1 e S2 amarelos
71      MOV S1_S2,#S1_S2_Amarelos
72      JMP ResetR0
73  ContaSegundos_entre_10e15: ;ver se A é maior a 10 e menor que 15, saltar se não for
74      CJNE A, #10, Maior ;comparar A com 10
75  Maior:
76      JC ContaSegundos_15 ;se o carry é 1 então A é menor que 10 e por isso salta-se
77      CJNE A, #15, Menorque  ;comparar A com 15
78  Menorque:
79      JNC ContaSegundos_15 ;se o carry é 0 então A é maior ou igual a 15 e por isso salta-se
80      CPL P3_Verde ;muda o estado de P3, para pôr P3 intermitente
81      JMP ResetR0
82  ContaSegundos_15:
83      CJNE A, #15, ContaSegundos_25
84      ;Pôr S1 e S2 vermelhos, S3 verde e P3 vermelho
85      MOV S1_S2,#S1_S2_Vermelhos
86      MOV S3_P3,#S3_Verde_P3_Vermelho
87      JMP ResetR0
88  ContaSegundos_25:
89      CJNE A, #25, ResetR0
90      ;Pôr S3 amarelo
91      MOV S3_P3,#S3_Amarelo_P3_Vermelho
92  ResetR0:
93      MOV R0,#Zero ;reset de R0, para não se efetuar as verificações sem A ser incrementado
94  ContaSegundos_30:
95      CJNE A, #30, Principal
96      CLR A ;fazer A=0 para voltar-se ao início do ciclo
97      MOV R0, #Um
98      JMP Principal
99
100 Inicializacoes:
101     CLR A ;fazer reset de A
102     MOV R0,#Um ;meter R0 a um para definir os semáforos no estado inicial
103     MOV R1, #Quarenta ;meter em R1 o número de contagens necessárias para perfazer 10 milisegundos
    (40)
104     MOV R2, #Cem ;meter em R2 o número de contagens de 10 milisegundos para perfazer 1 segundo (100)
105     MOV S1_S2, #S1_S2_Verdes
106     MOV S3_P3, #S3_Vermelho_P3_Verde
107     RET
108
109 AtivaInterrupcao:
110     MOV IE, #10000011b ;ativar as interrupções em geral, a do timer0 e a externa 0
111     SETB IT0 ;definir a interrupção externa para ser ativa no falling edge
112     RET
113
114 AtivaTemporizador:
115     MOV TMOD, #00000010b ;definir o timer 0 no modo 2 (8 bit - auto reload)
116     MOV TL0, #TempoL
117     MOV TH0, #TempoH
118     SETB TR0 ;iniciar o timer
119     RET
120
121 InterrupcaoTempo0:
122     DJNZ R1, FimIT0 ;decrementar R1 e verificar se chegou a 0, saltar se não chegou a 0
123     MOV R1, #Quarenta ;reset de R1
124     DJNZ R2, FimIT0 ;como passou 10 milisegundos, decrementar R2 e verificar se chegou a 0, saltar
    se não
125     INC A ;se R2 chegou a 0, passou um segundo, incrementar A
126     MOV R0,#Um ;meter R0 a um, para efetuar as verificações de A em Principal pois A aumentou
127     MOV R2, #Cem ;reset de R2
128 FimIT0:
129     RETI
130
131 InterrupcaoExterior0:
132     JB S3_Verde, FimIE0 ;verificar se S3_Verde está a 1 (desligado) e saltar se isso ocorrer
133     MOV A, #VinteECinco ;se S3_Verde está a 0 (ligado) metemos em A vinte e cinco para mudar S3 para
    amarelo
134     MOV R0,#Um ;meter R0 a um, para efetuar as verificações de A em Principal pois A foi alterado
135     MOV R1, #Quarenta ;reset de R1
136     MOV R2, #Cem ;reset de R2
137 FimIE0:
138     RETI
139
140 end
```