Uma imagem com sentado, preto, computador

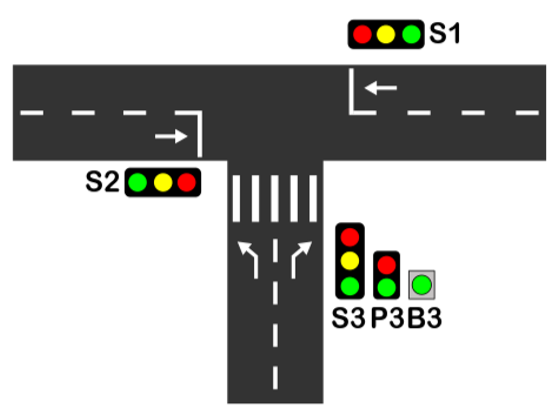
Descrição gerada automaticamente

**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 3](#_Toc41209280)

[2. Objetivos 3](#_Toc41209281)

[3. Desenvolvimento 4](#_Toc41209282)

[4. Discussão de Resultados 6](#_Toc41209283)

[5. Conclusão 6](#_Toc41209284)

[6. Bibliografia 6](#_Toc41209285)

[7. Anexos 7](#_Toc41209286)

[7.1. Fluxograma do programa principal 7](#_Toc41209287)

[7.2. Fluxograma da interrupção do timer 8](#_Toc41209288)

[7.3. Fluxograma da interrupção externa 8](#_Toc41209289)

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.

1. **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 microcontrolador 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.

1. **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 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 do microcontrolador.

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, e 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, 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 P3 é pressionado (com S3 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 transformar 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 ao LED’s, sendo R0 posto a 0 até que o acumulador altere de valor novamente.

1. **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 P3 para pôr o semáforo S3 a amarelo de modo a P3 ficar verde 5 segundos quando o botão é pressionado também funcionou e tornou-se simples com o nosso raciocínio.

1. **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 microcontrolador 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.

1. **Bibliografia**

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

1. **Anexo A - Fluxogramas**
   1. **Fluxograma do programa principal**

**Uma imagem com texto

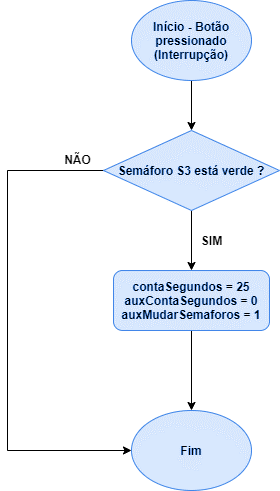
Descrição gerada automaticamente**

* 1. **Fluxograma da interrupção do timer**

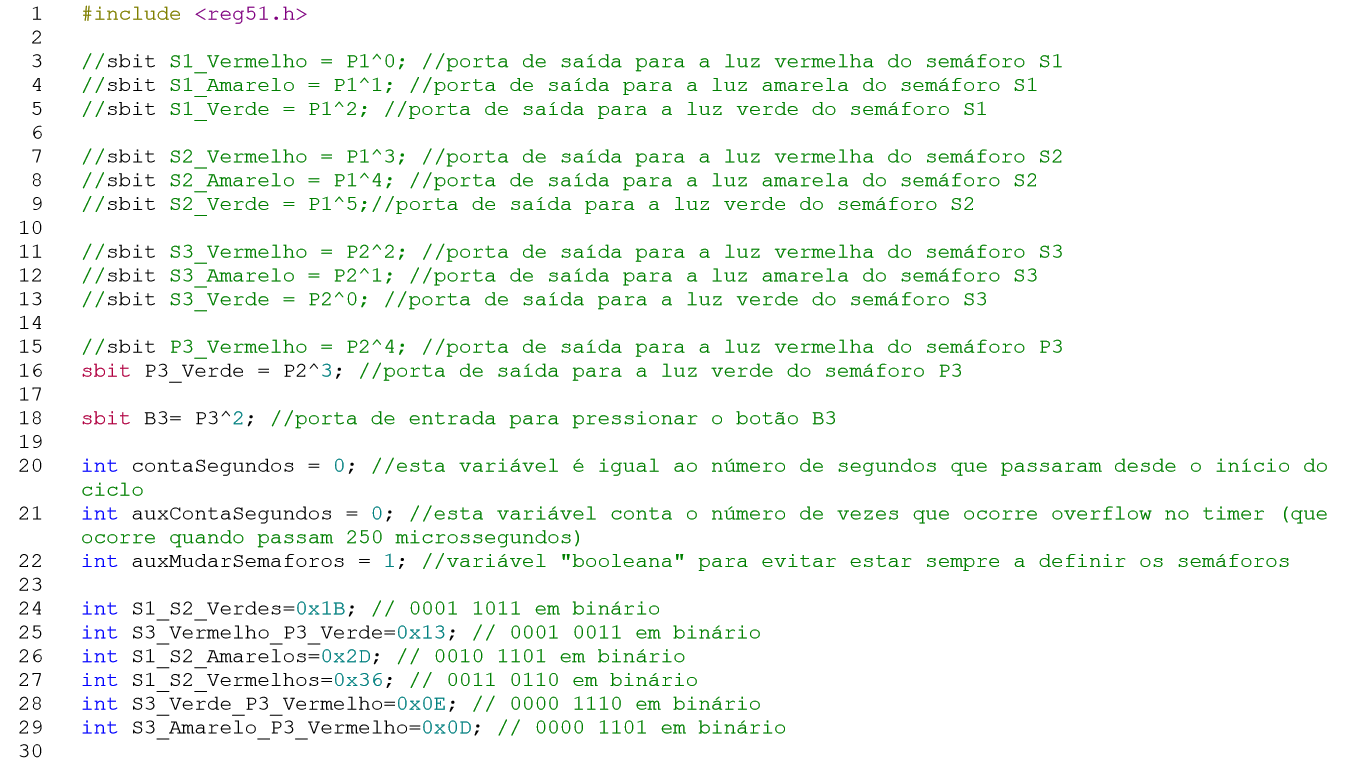
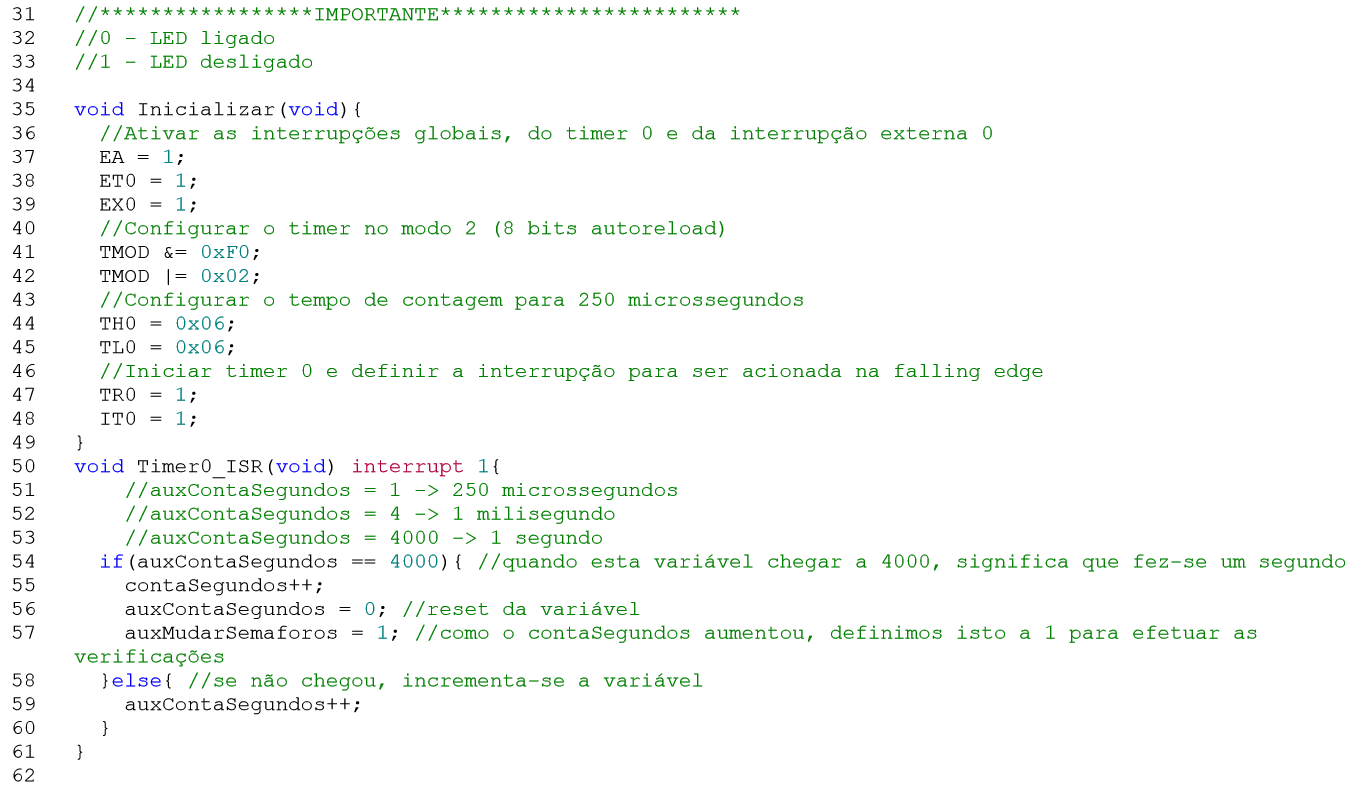
**Uma imagem com texto

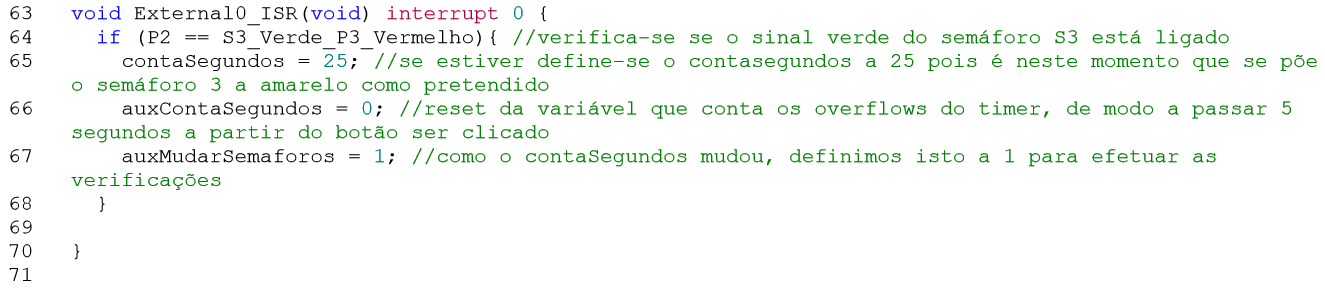
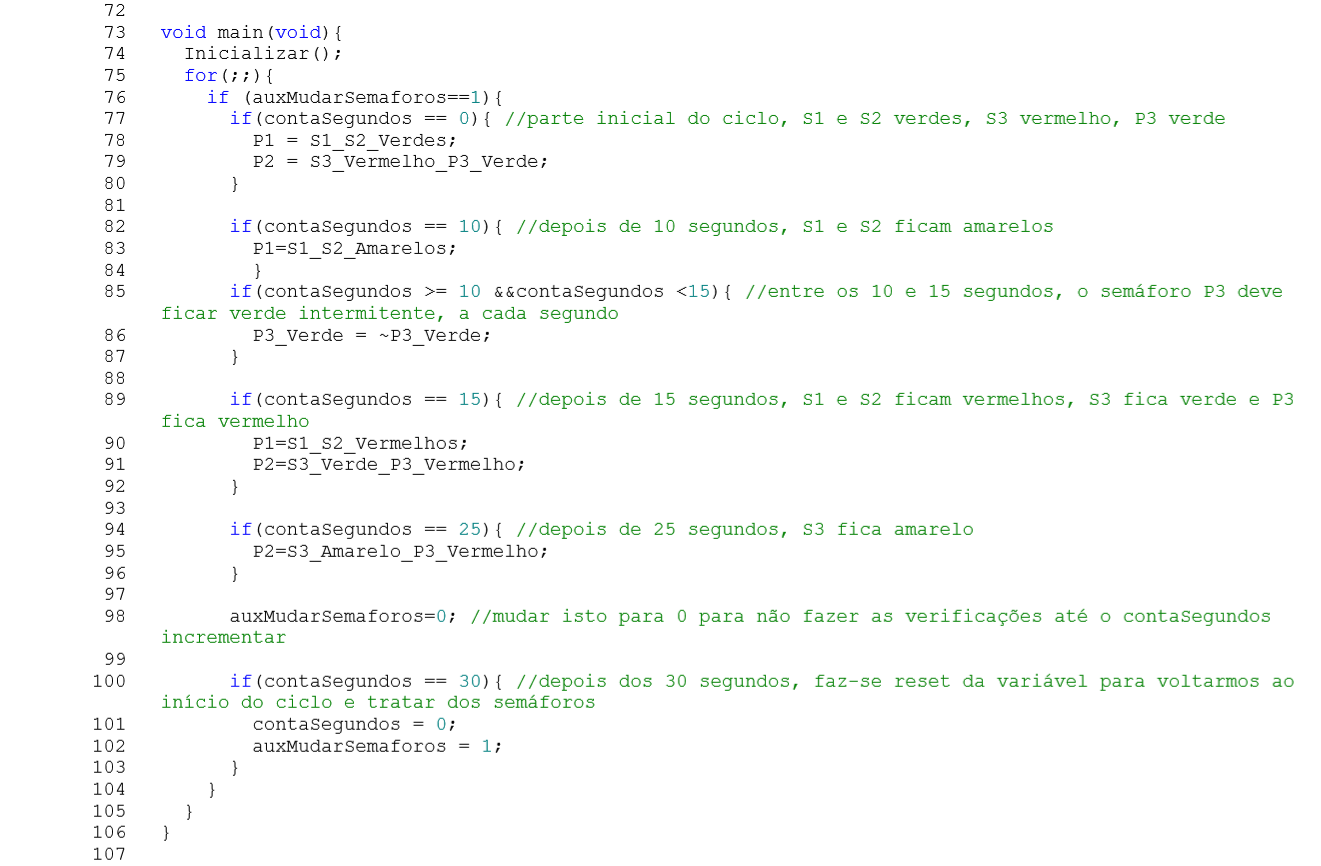
Descrição gerada automaticamente**

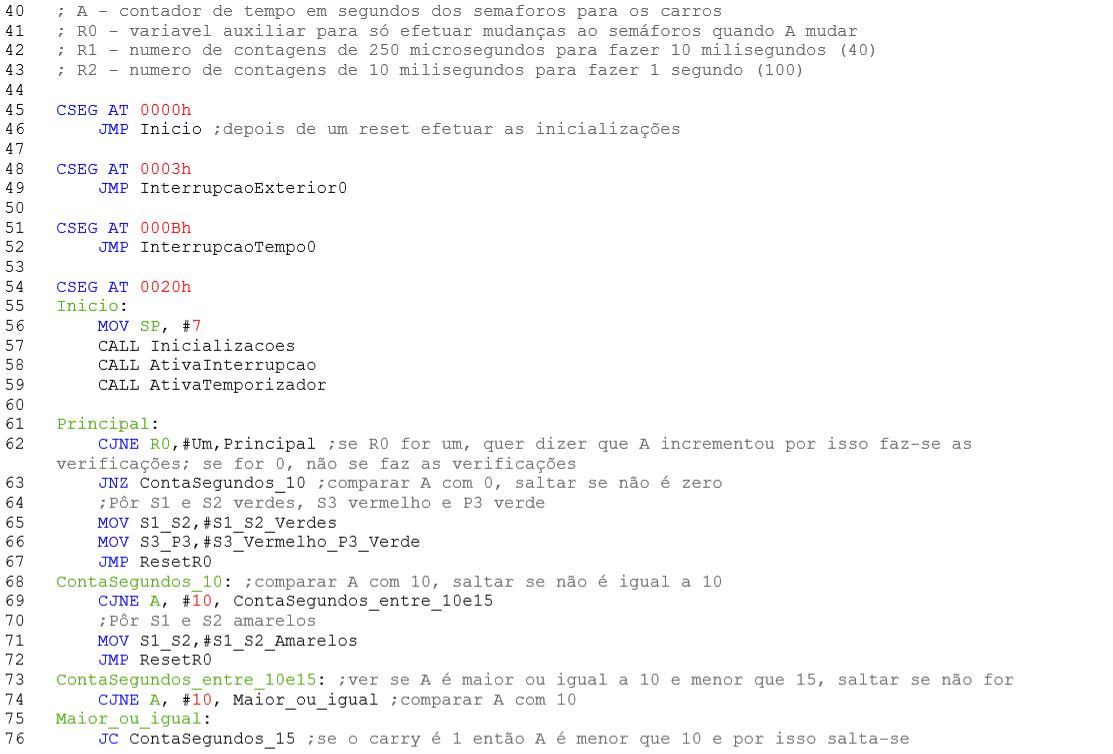
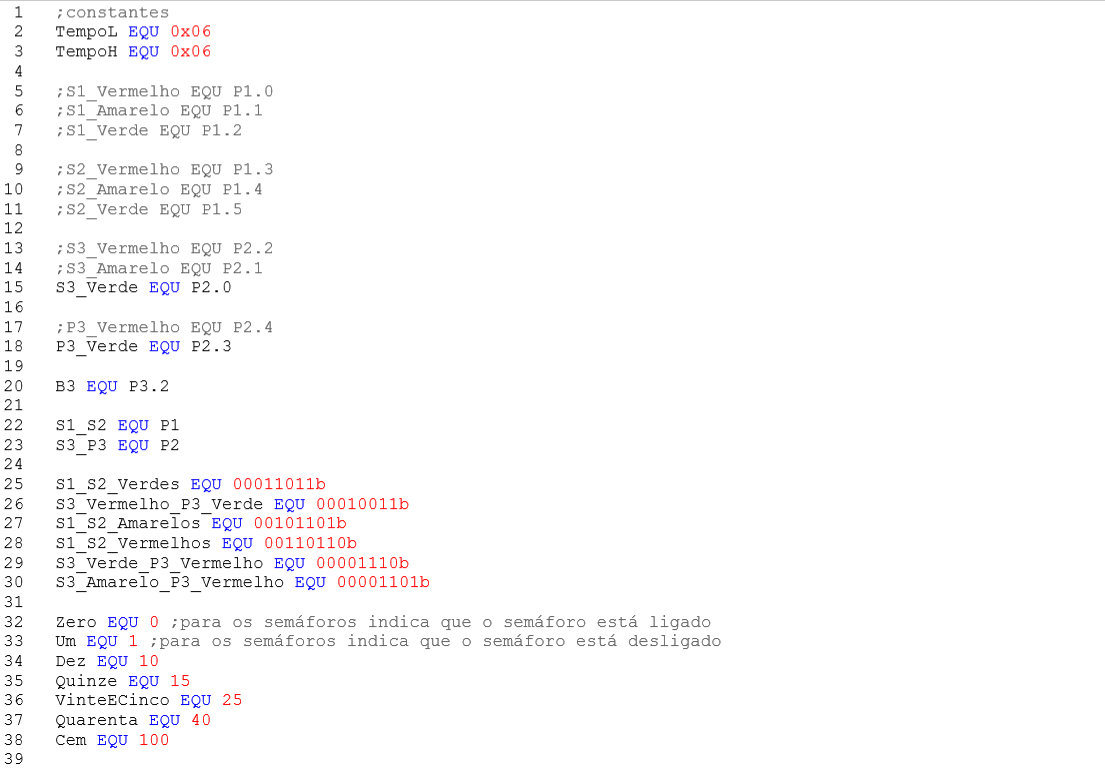
* 1. **Fluxograma da interrupção externa**

****

1. **Anexo B – Código**
   1. **Código em linguagem C**





* 1. **Código em linguagem Assembly**

