**UNIVERSIDADE DE SÃO PAULO**
ESCOLA DE ENGENHARIA DE SÃO CARLOS


# Cronômetro Digital em Assembly no Microprocessador 8051

SEL0614 – Aplicação de Microprocessadores

Giovanna de Freitas Velasco - 13676346
Jean Carlos Pereira Cassiano - 13864008

Prof. Pedro Oliveira

São Carlos
10/2024

# 1 Introdução

O presente projeto consiste no desenvolvimento de um contador digital de 0 a 9 utilizando um display de 7 segmentos e controlado por meio de dois botões de entrada.

Cada botão de entrada permite um ajuste distinto do delay presente entre cada incremento no contador, onde o botão 0, denotado por SW0 configura o delay para 0,25 segundos e o botão 1, denotado por SW1, configura o delay para 1 segundo. Além de exibir a contagem no display, o programa deve apenas iniciar a contagem quando um dos botões for pressionado, interrompendo a contagem caso nenhum botão esteja pressionado.

Este tipo de contador é uma aplicação básica, mas essencial, para entender como interagir com periféricos de entrada e saída, bem como gerenciar temporização utilizando o timer interno do microcontrolador.

A plataforma utilizada para desenvolver essa aplicação foi o simulador *EdSim51*, que simula um Microprocessador 8051, o que nos permitiu aprender melhor sobre o desenvolvimento de códigos na linguagem Assembly para aplicações em microprocessadores. O código criado foi disponibilizado no arquivo *Counter.asm*, presente no repositório do *GitHub* com os projetos da disciplina e um vídeo curto exemplificando seu funcionamento, presente no seguinte link: https://youtu.be/oXKW_1NEock.

# 2 Desenvolvimento do Código

O desenvolvimento do contador digital foi realizado em linguagem Assembly, utilizando o microcontrolador 8051. Em seguida, utilizamos o simulador EdSim51 para garantir que todas as rotinas funcionassem corretamente. O programa foi testado em modo de execução normal e em modo passo a passo, permitindo verificar que os atrasos estavam de acordo com as especificações e que o contador exibia os números corretamente no display. A precisão do delay foi ajustada para que o Timer 0 gerasse ciclos exatos de 50 ms, garantindo que o intervalo de contagem entre números estivesse dentro dos tempos configurados. Vamos analisar as principais funções criadas para o projeto. A seguir, detalhamos a metodologia adotada e explicamos as funções principais do código.

### 2.0.1 A função MAIN

A função MAIN é responsável pelo controle principal do fluxo do programa. Ela inicialmente chama a função CHECK_BUTTONS para verificar continuamente se um dos botões (SW0 ou SW1) está pressionado. Uma vez que o delay é configurado ( a depender do botão pressionado), a MAIN usa o valor do contador (armazenado em R0) para acessar a tabela DISPLAY_CODES. Com isso, o número correspondente é exibido no display de 7 segmentos. Após exibir o número, a MAIN chama a subrotina DELAY, que pausa o programa por um período configurado antes de avançar para o próximo número. Em seguira, o contador é incrementado e o processo inteiro se repete. Quando o contador chegar em 10, ele é reiniciado para 0, gerando o ciclo de contagem de 0 a 9.

A lógica principal do código garante que o contador funcione apenas quando um botão estiver pressionado. Se nenhum botão estiver pressionado, a função CHECK_BUTTONS força o programa a permanecer em espera.

Essas rotinas trabalham de forma coordenada para permitir que o contador exiba os números no display de 7 segmentos, ajustando o tempo entre os incrementos conforme o botão pressionado.

### 2.0.2 A função CHECK_BUTTONS

A função CHECK_BUTTONS realiza a verificação de qual botão foi pressionado e ajusta o delay correspondente àquele botão. Se o botão SW0 (conectado a P2.0) for pressionado, a função salta para SET_DELAY_250MS. Essa sub-rotina , por sua vez, configura o delay para 0,25 segundos, atribuindo 5 ciclos de 50 ms à variável DELAY_TIME. Já se o botão SW1 (conectado a P2.1) for pressionado, a função salta para SET_DELAY_1S. Essa sub-rotina configura o delay para 1 segundo, atribuindo 20 ciclos de 50 ms à variável DELAY_TIME. Caso nenhum dos botões esteja pressionado, a função repete a verificação continuamente, forçando o programa a permanecer em espera.

Portanto, essa lógica permite que o sistema ajuste o tempo de espera entre os incrementos do contador com base no botão pressionado. Quando ambos os botões não estão pressionados, o programa se mantém ocioso.

### 2.0.3 A função DELAY

A sub-rotina DELAY usa o Timer 0 para gerar uma pausa precisa no programa. Esse atraso é configurado conforme a variável DELAY_TIME, que define o número de ciclos de 50 ms que devem ser executados. A função configura o Timer 0 no modo 16 bits, e carrega os valores em TH0 e TL0 para produzir um atraso de aproximadamente 50 ms a cada ciclo. Com o Timer 0 configurado, o programa entra em um loop, onde aguarda o overflow do Timer 0. Quando o timer atinge o valor máximo e transborda, o bit TF0 é acionado. A sub-rotina então decrementa R2 (que armazena o número de ciclos de delay definidos por DELAY_TIME) e repete o ciclo até que o número desejado de ciclos de 50 ms seja completado. Após atingir o total de ciclos especificado, o programa retorna ao loop principal, permitindo a atualização do contador.

A função DELAY garante que o tempo de espera entre os números no display seja preciso, e sua duração é controlada dinamicamente pela função CHECK_BUTTONS.

### 2.0.4 Configuração do Display de 7 segmentos

A tabela DISPLAY_CODES contém os códigos binários que mapeiam cada número de 0 a 9 para os segmentos específicos do display de 7 segmentos. Cada código na tabela corresponde a uma combinação de segmentos que devem ser ativados para formar o número. Assim, 0C0H exibe o número 0, 0F9H exibe o número 1, 0A4H exibe o número 2 e assim por diante.

Ao carregar o valor correspondente em P1, o display de 7 segmentos exibe o número desejado. Isso permite que o contador altere rapidamente os números conforme o valor do contador R0 sem necessidade de operações complexas.

# 3 Análise no Simulador

Vamos agora realizar uma análise do funcionamento do cógido no simulador EdSim51. Ao iniciar, o simulador, além de fazer as preparações para as rotinas funcionarem, ele é levado para a rotina MAIN, que, por sua vez, redireciona para a rotina CHECK_BUTTONS.

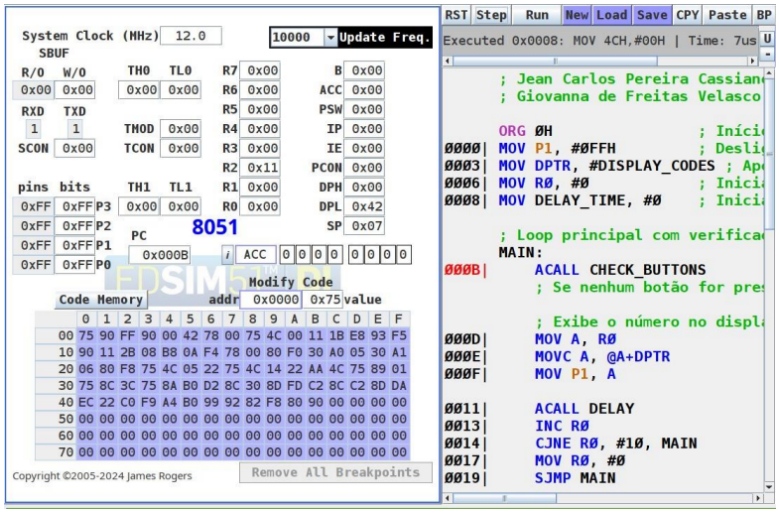


Figura 1: Rotina MAIN chama a rotina CHECK_BUTTONS

Na função CHECK_BUTTONS, a rotina fica eternamente no loop até o momento em que um dos botões SW0 ou SW1 seja apertado, fazendo a rotina retornar para a MAIN.

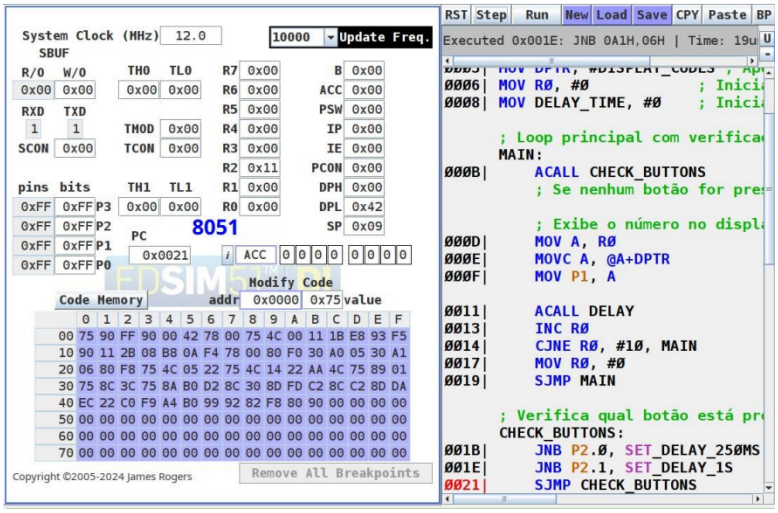


Figura 2: Rotina CHECK_BUTTONS

Quando o SW1 é acionado, por exemplo, o delay é configurado para 1 segundo e a rotina retorna para a MAIN. Isso é mostrado na Figura 3.

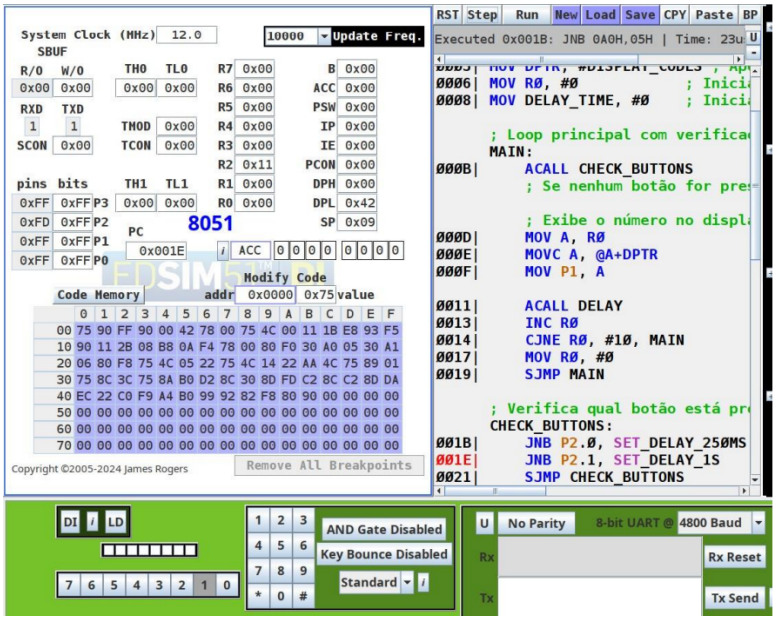


Figura 3: Exemplo do SW1 sendo acionado

Ao retornar para a função MAIN o simulador mapeia o valor de R0 (atualmente zero) no display de 7 segmentos. Assim, o valor zero é mapeado pela tabela de código de números, declarada no início do código, e o devido código é passado para o display, como visto na Figura 4.

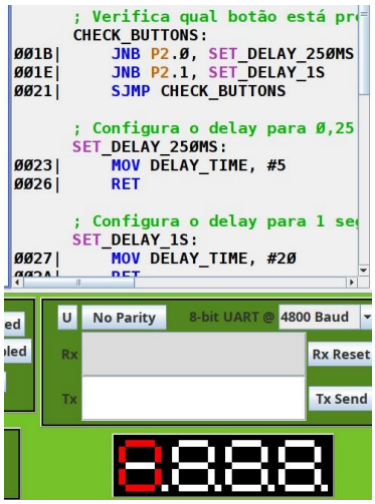


Figura 4: Display com valor zero

Após isso, é chamada a rotina de Delay. Nesta rotina, chamamos uma sub rotina que demora cerca de 50ms. Foi utilizado o Timer 0 para seu funcionamento. O código verifica continuamente a flag TF0 com JNB TF0, WAIT_DELAY. Esta linha mantém o programa em um loop de espera até que o Timer 0 sofra um overflow, momento em que TF0 é setado para 1.

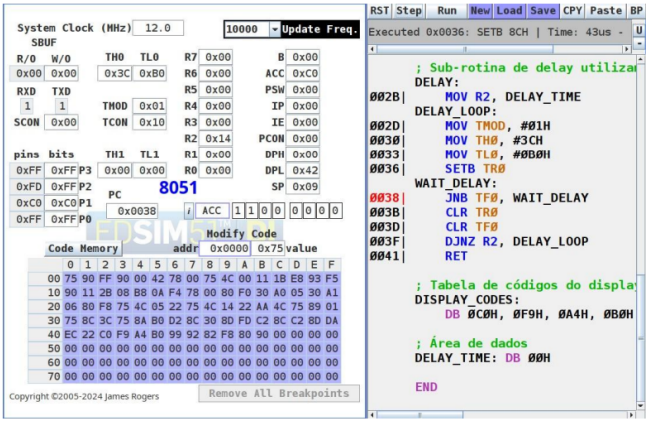


Figura 5: Função Delay esperando o Overflow do TF0

Quando o TF0 é setado para 1, a função continua. Tal rotina demora cerca de 50ms, e, então, ela retorna para o começo, zerando os registradores e recomeçando. Dado o botão apertado, o código seguirá esse loop conforme o tempo esperado. Neste caso, como o SW1 foi apertado, serão cerca de 20 iterações, até o tempo correr 1 segundo. Esse comportamento é evidenciado na Figura 6


Figura 6: A rotina termina cerca de 1 segundo após seu início

Assim, o código retorna para a MAIN, onde o valor de R0 é incrementado, como demonstrado a seguir:

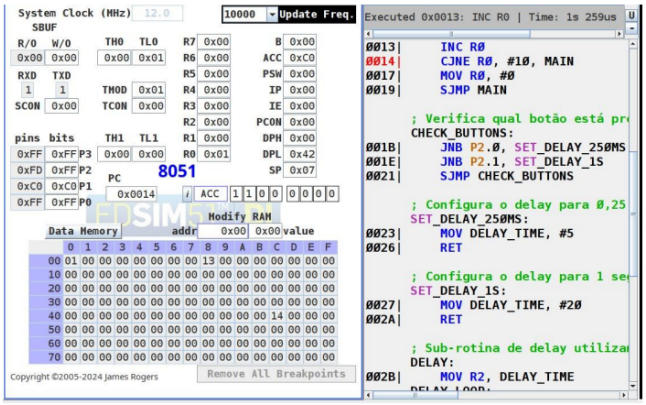


Figura 7: Valor de R0 é incrementado em 1 (atualmente em 1 após a primeira iteração)

Novamente, o código mapeia o valor de R0 de acordo com o código dos números do display, e este código é passado para o display de 7 segmentos. Atente-se ao tempo demorado: cerca de 1 segundo.

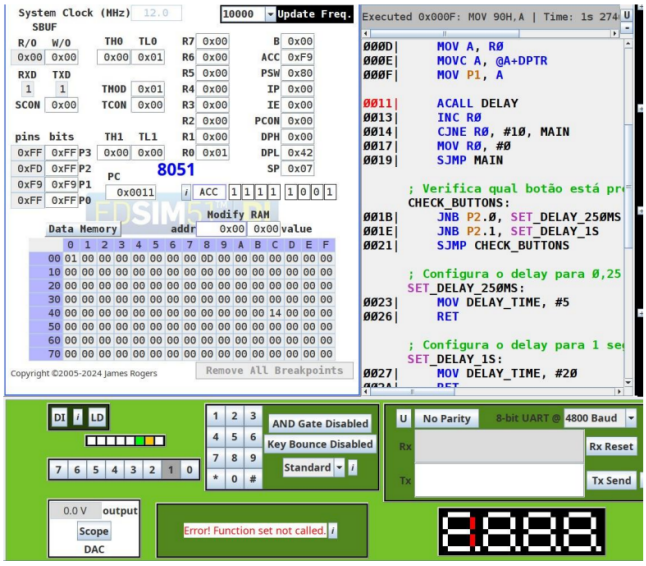


Figura 8: Registrador A com o valor mapeado para o display de 7 segmentos e número aparecendo

Desta forma, o simulador continua seguindo todos os passos até agora, verificando novamente os botões que foram ativados ou não. As rotinas voltam a ser chamadas continuamente até que um dos botões seja desativado.

# 4 Conclusão

O desenvolvimento deste projeto proporcionou uma compreensão prática do uso do microcontrolador 8051 para controlar um display de 7 segmentos, utilizando entradas de botões e temporização baseada em hardware.

Sua realização permitiu um estudo de diversos tópicos ligados às possíveis aplicações de microprocessadores. O desenvolvimento do código permitiu uma compreensão mais profunda da programação em Assembly 8051, além de necessitar do planejamento da organização das chamadas de sub-rotinas e da verificação constante do acionamento dos botões SW1 e SW2. Assim, o projeto também contribuiu para aprendermos a realizar uma integração mais eficientes entre as entradas e saídas de um sistema embarcado.

Além disso, a utilização de delays distintos possibilitou uma maior compreensão de como variar o tempo entre as atividades a serem realizadas pelo microprocessador, o que também contribui para um melhor aprendizado sobre o funcionamento do microcontrolador 8051 e de sistemas embarcados em geral.