

UNIVERSIDADE DE SÃO PAULO – USP
ESCOLA DE ENGENHARIA DE SÃO CARLOS – EESC

Projeto 1 - Aplicação de Microprocessadores

Alunos:

Gustavo Curado Ribeiro - 14576732

Luís Filipe Silva Forti - 14592348

Miguel Rodrigues Fonseca - 14682196

Professor:

Ricardo Fernandes

São Carlos - SP
2025

Sumário

1	Funcionamento do Código em Assembly	1
2	Desafio do projeto	2
2.1	Tempo da instrução DJNZ no 8051	3
2.2	Códigos de delay e cálculos adicionais	4
2.3	Tempos obtidos para contar de 0 a 9 para cada delay	5

1 Funcionamento do Código em Assembly

O programa implementa um contador em display de 7 segmentos utilizando a família 8051. O funcionamento baseia-se em dois modos de operação: contagem crescente e contagem decrescente, controlados por dois botões conectados à porta P2.0 e P2.1.

Inicialização

Na inicialização, o ponteiro de dados (DPTR) é carregado com o endereço da tabela LUT (Look-Up Table), que contém os padrões dos dígitos de 0 a 9, e o registrador R0 é definido como índice da contagem. O display (P1) liga, mostrando o número em R0.

Seleção de Modo

O programa permanece em espera verificando os botões:

- Se P2.0 for pressionado, inicia-se a contagem crescente.
- Se P2.1 for pressionado, inicia-se a contagem decrescente.

Contagem Crescente

No modo crescente:

1. O bit PSW.5 é setado, definindo o *delay* rápido.
2. Após o *delay*, verifica-se se o botão de contagem decrescente foi pressionado; caso positivo, ocorre a troca de modo.
3. Caso contrário, R0 é incrementado. Quando o valor atinge 10 (0Ah), R0 retorna a zero, reiniciando a contagem.
4. O valor de R0 é usado como índice para acessar a tabela LUT.
5. O valor correspondente é enviado ao display (P1).
6. Reinicia-se o loop.

Contagem Decrescente

No modo decrescente, a lógica é análoga, mas com diferenças importantes:

1. O bit PSW.5 é limpo, definindo o *delay* lento.
2. Após o *delay*, verifica-se se o botão de contagem crescente foi pressionado; caso positivo, ocorre a troca de modo.

3. Caso contrário, R0 é decrementado. Quando o valor atinge -1 (0FFh), R0 retorna a 9, reiniciando a contagem.
4. O valor de R0 é usado como índice para acessar a tabela LUT.
5. O valor correspondente é enviado ao display (P1).
6. Reinicia-se o loop.

Rotina de Delay

O programa utiliza duas rotinas de atraso diferentes:

- **Delay lento:** três contagens aninhadas (R1 = 10, R2 = 200 e R3 = 250), resultando em um atraso mais longo.
- **Delay rápido:** contagens menores (R1 = 5, R2 = 200, R3 = 250), resultando em um atraso mais curto.

A seleção entre os atrasos é controlada pelo bit PSW.5, que funciona como um *flag* indicando o modo atual de contagem.

Tabela (LUT)

A LUT, localizada a partir do endereço 0200h, contém os valores hexadecimais que representam os dígitos de 0 a 9 no display de 7 segmentos. Esses padrões são acessados indiretamente utilizando DPTR e o índice R0.

Explicação Importante

Devido ao comportamento dos botões no *Edsim*, que funcionam mais como *switches* (chaves) do que como *push-buttons* (botões de pulso), se ambos os botões (P2.0 e P2.1) forem pressionados ao mesmo tempo, o contador ficará em estado de espera até que apenas um botão volte ao nível lógico baixo (0). Além disso, a contagem é mantida na forma atual quando seu botão é apertado, já que o programa verifica apenas a ativação do botão referente ao modo oposto do atual.

2 Desafio do projeto

Aqui estaremos explicando como é possível calcular os tempos de delay de forma precisa com base no clock do 8051 e nas instruções utilizadas no delay.

2.1 Tempo da instrução DJNZ no 8051

Para o microcontrolador 8051, com **clock de 12 MHz**, cada ciclo de máquina dura:

$$T_{ciclo} = \frac{12}{12\text{MHz}} = 1\mu s$$

A instrução DJNZ consome, em média, 2 ciclos de máquina, ou seja:

$$T_{DJNZ} \approx 2\mu s$$

Para um delay de 1 segundo, seriam necessários:

$$\frac{1}{T_{DJNZ}} \approx \frac{1}{2 * 10^{-6}} = 5 * 10^5 = 250 * 200 * 10 \text{ ciclos}$$

Assim, para um delay de 1 segundo, pode-se utilizar de múltiplos loops aninhados avaliando diferentes DJNZ, um com 10 iterações, outro com 200 e um último com 250. Pela mesma lógica, um delay mais rápido, de 0,5 segundos, teria o primeiro loop com apenas 5 iterações.

2.2 Códigos de delay e cálculos adicionais

```

1 delay:
2     JNB PSW.5, delay_fast ; Se F0 valer zero, vai para delay rapido
3
4     ; ----- delay lento -----
5     ; delay = 1s
6     MOV R1, #10
7 d1s:  MOV R2, #200
8 d2s:  MOV R3, #250
9 d3s:  DJNZ R3, d3s
10     DJNZ R2, d2s
11     DJNZ R1, d1s
12     RET
13
14 delay_fast:
15     ; ----- delay rapido -----
16     ; delay = 0.5s
17     MOV R1, #5
18 d1f:  MOV R2, #200
19 d2f:  MOV R3, #250
20 d3f:  DJNZ R3, d3f
21     DJNZ R2, d2f
22     DJNZ R1, d1f
23     RET

```

Listing 1 – Códigos de delay

Tempo para contar de 0 até 9 (10 números):

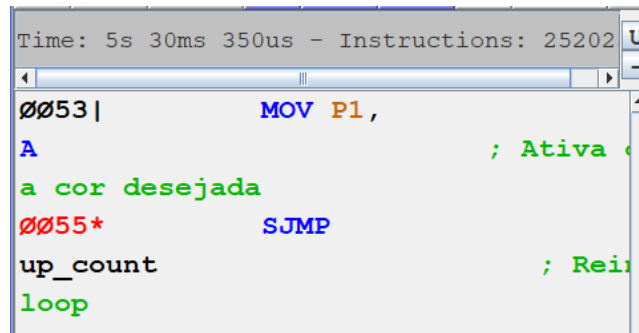
$$T_{total_rapido} = 10 \times 0,5 s = 5 s$$

$$T_{total_lento} = 10 \times 1 s = 10 s$$

Por fim, sabemos que esse será mais ou menos o tempo necessário para ir de 0 a 9 no contador. Entretanto, é claro que o tempo será um pouco maior, pois as trocas de número do display e as operações entre delays aumentam levemente o tempo necessário.

2.3 Tempos obtidos para contar de 0 a 9 para cada delay

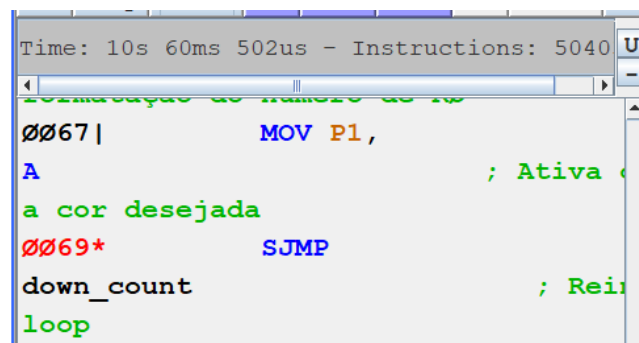
Para medir o tempo de uma contagem completa, compilamos o código e adicionamos um breakpoint na linha imediatamente após a linha de atualização do display. Então permitimos que o código execute até a contagem retornar a 0. Com o breakpoint ativando, tiramos uma print do tempo total utilizado, obtendo as imagens abaixo:



The screenshot shows a debugger window with the following assembly code:

```
Time: 5s 30ms 350us - Instructions: 25202
0053|          MOV P1,
A          ; Ativa c
a cor desejada
0055*          SJMP
up_count    ; Rein
loop
```

Figura 1 – Tempo de contagem crescente: delay rápido



The screenshot shows a debugger window with the following assembly code:

```
Time: 10s 60ms 502us - Instructions: 5040
0067|          MOV P1,
A          ; Ativa c
a cor desejada
0069*          SJMP
down_count  ; Rein
loop
```

Figura 2 – Tempo de contagem decrescente: delay lento

Pode-se perceber que os valores são coerentes com o esperado, comprovando o código.