UNIVERSIDADE ESTADUAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
BACHARELADO EM CIÊNCIAS
DA COMPUTAÇÃO

LUCAS DE LUCENA SIQUEIRA

DISCIPLINA: LABORATÓRIO DE ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

EXPERIMENTO 04

RELATÓRIO - DISPLAY DE 7 SEGMENTOS

CAMPINA GRANDE - PB
2021

SUMÁRIO

1. RESUMO

Neste trabalho em questão é possível observar alguns conceitos básicos e importantes sobre a programação do microcontrolador MCS51 em Assembly a partir do simulador MCU8051 IDE. Por fim, foram feitas as atividades propostas pelo professor e expostas neste relatório para a prática e conhecimento do conjunto de instrução do microcontrolador e alguns dos seus periféricos, como a utilização do display de 7 segmentos com a utilização da técnica de multiplexação e o uso do keypad.

2. INTRODUÇÃO

O presente trabalho tem como objetivo introduzir o estudo do microcontrolador MCS51 a partir do uso do simulador MCU8051 IDE. Tendo o Assembly como linguagem dominante na programação do referente microcontrolador, se faz necessário também o estudo da mesma a partir de consultas em seu *Datasheet*, que é nada mais do que uma folha com dados e especificações técnicas e de desempenho do produto levado em consideração, que no nosso caso é o microcontrolador MCS51.

3. MATERIAL E MÉTODOS

3.1. OBJETIVO

O experimento em questão tem como objetivo induzir o estudo e a prática da programação do microcontrolador MCS51 a partir do simulador MCU8051 IDE com o uso da linguagem Assembly. Tornando possível estudar o display de 7 segmentos com a técnica de multiplexação além da leitura de dados no simulador.

3.2. HARDWARE NECESSÁRIO

- Computador com sistema operacional superior ou equivalente ao Windows 7.

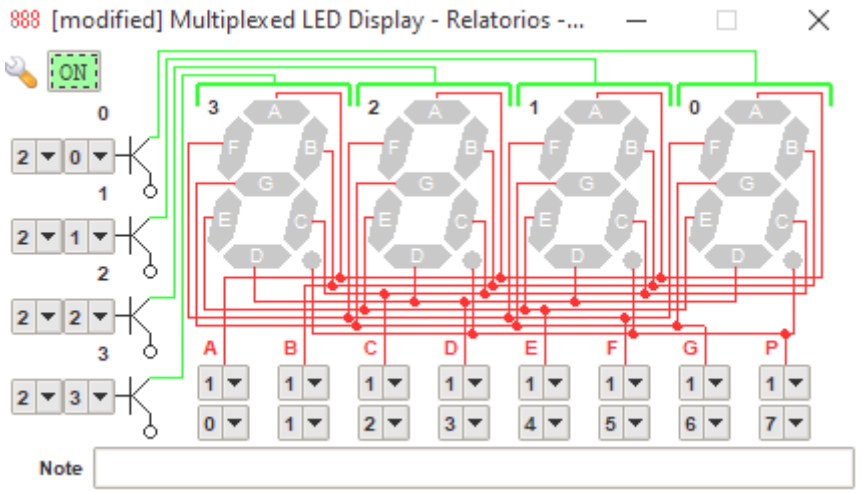
3.3. SOFTWARE NECESSÁRIO

- Simulador MCU 8051 IDE.

3.4. PROCEDIMENTOS

3.4.1. LIGAÇÕES DO DISPLAY DE 7 SEGMENTOS

Na figura abaixo estão as conexões realizadas no painel de LED “Multiplexed LED Display”. Para a ligação dos bits de 0 a 7 inferiores foi utilizada a porta P1 e para a ligação dos bits de 0 a 3 laterais foi utilizada a porta P2.



3.4.2. LISTAGEM DO PROGRAMA EM LM

- ORG 00H

É uma pseudo-instrução que define o endereço em que o programa irá começar ou continuar.

- LJMP inicio

Faz um salto para o label “inicio”.

- numeros: DB 11000000b

Instanciação de um label. Representa o número 0 em binário.

- DB 11111001b

Representa o número 1 em binário.

- DB 10100100b

Representa o número 2 em binário.

- DB 10110000b

Representa o número 3 em binário.

- DB 10011001b

Representa o número 4 em binário.

- DB 10010010b

Representa o número 5 em binário.

- DB 10000010b

Representa o número 6 em binário.

- DB 11111000b

Representa o número 7 em binário.

- DB 10000000b

Representa o número 8 em binário.

- DB 10010000b

Representa o número 9 em binário.

- inicio: MOV DPTR, #numeros

Move o valor presente no label "numeros" para o registrador DPTR.

- MOV A, #04

Move o valor 5 para o registrador acumulador A.

- MOV B, #03

Move o valor 4 para o registrador acumulador B.

- LCALL dispnum

Chama a subrotina "dispnum".

- LJMP $

Move o ponteiro relativo à execução para o endereço de memória $.

- dispnum: MOV P2, #0ffH

Instanciação da label “dispnum”. Move o valor "00ffH" para o registrador P2.

- MOV P1, #0ffH

Move o valor "00ffH" para o registrador P1.

- MOVC A, @A+DPTR

Soma o conteúdo do registrador acumulador com DPTR deixando o resultado da operação guardado no acumulador criando um endereço de memória.

- MOV P1, A

Move o conteúdo do registrador acumulador para o registrador P1.

- MOV A, B

Move o conteúdo do registrador B para o registrador acumulador.

- CJNE A, #01, disp1

Compara o conteúdo do registrador acumulador com o valor 01, caso seja diferente o programa vai para o label "disp1".

- CPL A

Inverte os bits do registrador acumulador.

- SJMP disp5

Salta para o label "disp5"

- disp1: CJNE A, #02, disp2

Instanciação do label “disp1”. Compara o conteúdo do registrador acumulador com o valor 02, caso seja diferente o programa vai para o label "disp2".

- CPL A

Inverte os bits do registrador acumulador.

- SJMP disp5

Salta para o label "disp5".

- disp2: CJNE A, #03, disp3

Compara o conteúdo do registrador acumulador com o valor 03, caso seja diferente o programa vai para o label "disp3".

- MOV A, #0FBH

Move o valor 0F7H para o registrador acumulador.

- SJMP disp5

Salta para o label "disp5".

- disp3: CJNE A, #04, disp4

Compara o conteúdo do registrador acumulador com o valor 04, caso seja diferente o programa vai para o label "disp4".

- SJMP disp5

Vai para o label "disp5".

- disp4: CLR A

Zera todo o registrador acumulador.

- disp5: MOV P2, A

Move o conteúdo do registrador acumulador para o registrador P2.

- RET

Retorna à sub-rotina.

- END

Indica o fim do programa.

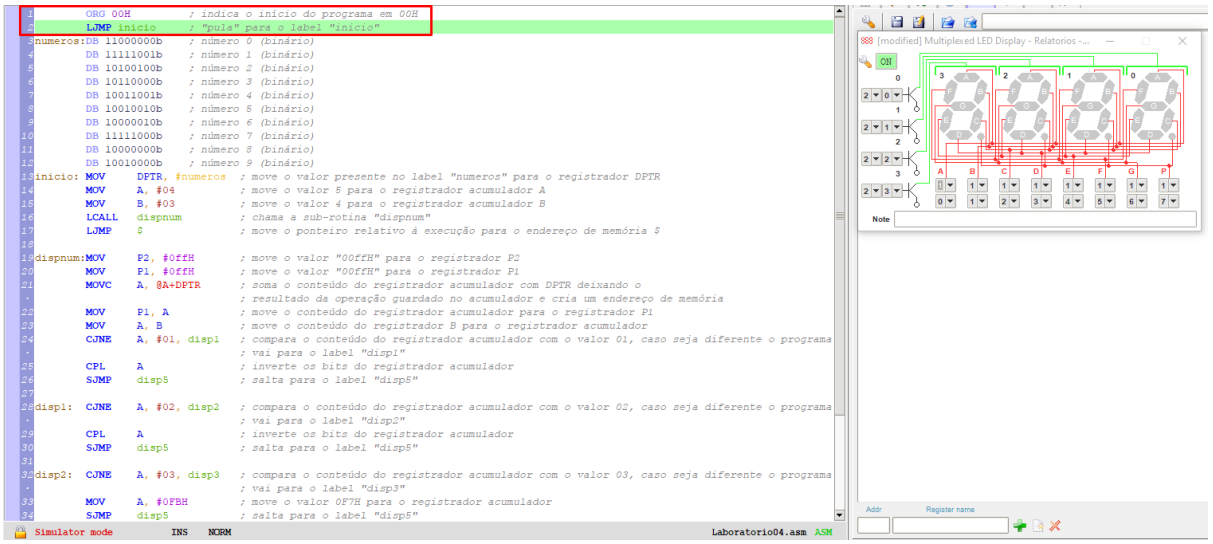
3.4.3. FUNCIONAMENTO DO PROGRAMA

Aqui estará presente toda a descrição detalhada de cada linha do programa utilizado como exemplo, apresentando em cada linha seu respectivo funcionamento comentado.

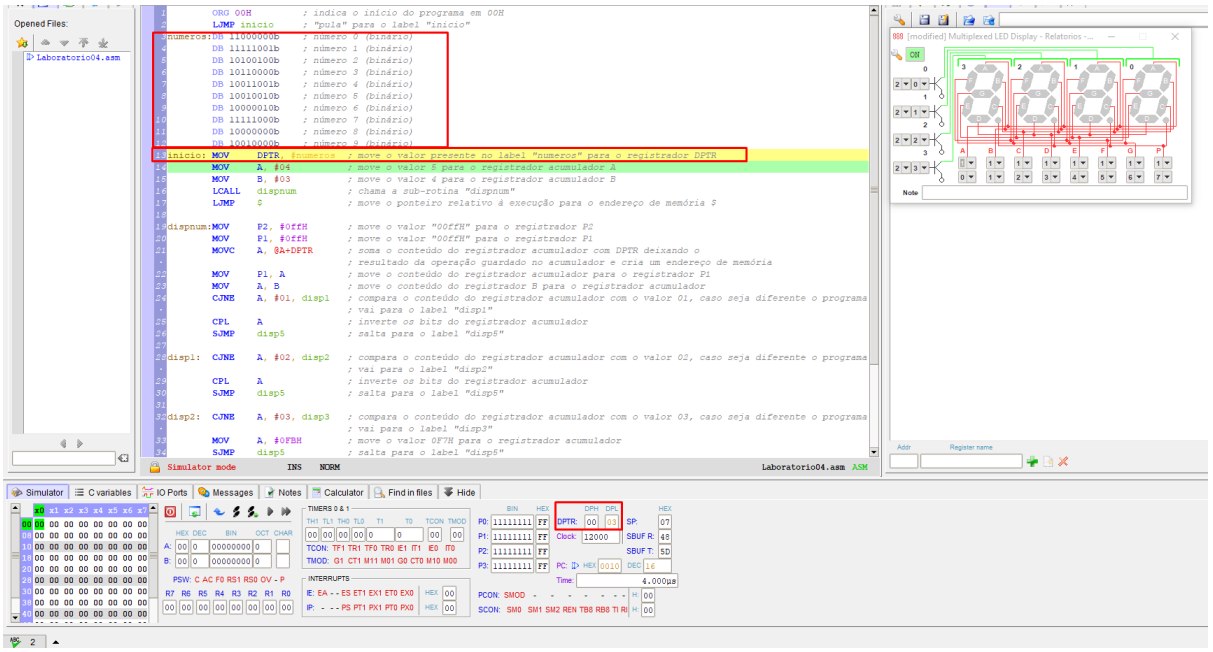
```
        ORG 00H          ; indica o início do programa em 00H
        LJMP inicio      ; "pula" para o label "inicio"
numeros:DB 11000000b     ; número 0 (binário)
        DB 11111001b     ; número 1 (binário)
        DB 10100100b     ; número 2 (binário)
        DB 10110000b     ; número 3 (binário)
        DB 10011001b     ; número 4 (binário)
        DB 10010010b     ; número 5 (binário)
        DB 10000010b     ; número 6 (binário)
        DB 11111000b     ; número 7 (binário)
        DB 10000000b     ; número 8 (binário)
        DB 10010000b     ; número 9 (binário)
inicio: MOV     DPTR, #numeros  ; move o valor presente no label "numeros" para o registrador DPTR
        MOV     A, #04          ; move o valor 5 para o registrador acumulador A
        MOV     B, #03          ; move o valor 4 para o registrador acumulador B
        LCALL   dispnum         ; chama a sub-rotina "dispnum"
        LJMP    $               ; move o ponteiro relativo à execução para o endereço de memória $

dispnum:MOV     P2, #0ffH       ; move o valor "00ffH" para o registrador P2
        MOV     P1, #0ffH       ; move o valor "00ffH" para o registrador P1
        MOVC    A, @A+DPTR      ; soma o conteúdo do registrador acumulador com DPTR deixando o
                                ; resultado da operação guardado no acumulador e cria um endereço de memória
        MOV     P1, A           ; move o conteúdo do registrador acumulador para o registrador P1
        MOV     A, B            ; move o conteúdo do registrador B para o registrador acumulador
        CJNE    A, #01, disp1   ; compara o conteúdo do registrador acumulador com o valor 01, caso seja diferente o programa
                                ; vai para o label "disp1"
        CPL     A               ; inverte os bits do registrador acumulador
        SJMP    disp5           ; salta para o label "disp5"

disp1:  CJNE    A, #02, disp2   ; compara o conteúdo do registrador acumulador com o valor 02, caso seja diferente o programa
                                ; vai para o label "disp2"
        CPL     A               ; inverte os bits do registrador acumulador
        SJMP    disp5           ; salta para o label "disp5"

disp2:  CJNE    A, #03, disp3   ; compara o conteúdo do registrador acumulador com o valor 03, caso seja diferente o programa
                                ; vai para o label "disp3"
        MOV     A, #0FBH        ; move o valor 0F7H para o registrador acumulador
        SJMP    disp5           ; salta para o label "disp5"

disp3:  CJNE    A, #04, disp4   ; compara o conteúdo do registrador acumulador com o valor 04, caso seja diferente o programa
                                ; vai para o label "disp4"
        SJMP    disp5           ; vai para o label "disp5"
disp4:  CLR     A               ; zera todo o registrador acumulador
disp5:  MOV     P2, A           ; move o conteúdo do registrador acumulador para o registrador P2
        RET                     ; retorna à sub-rotina
        END                     ; indica o fim do programa
```

Comentários e execução do código acima:

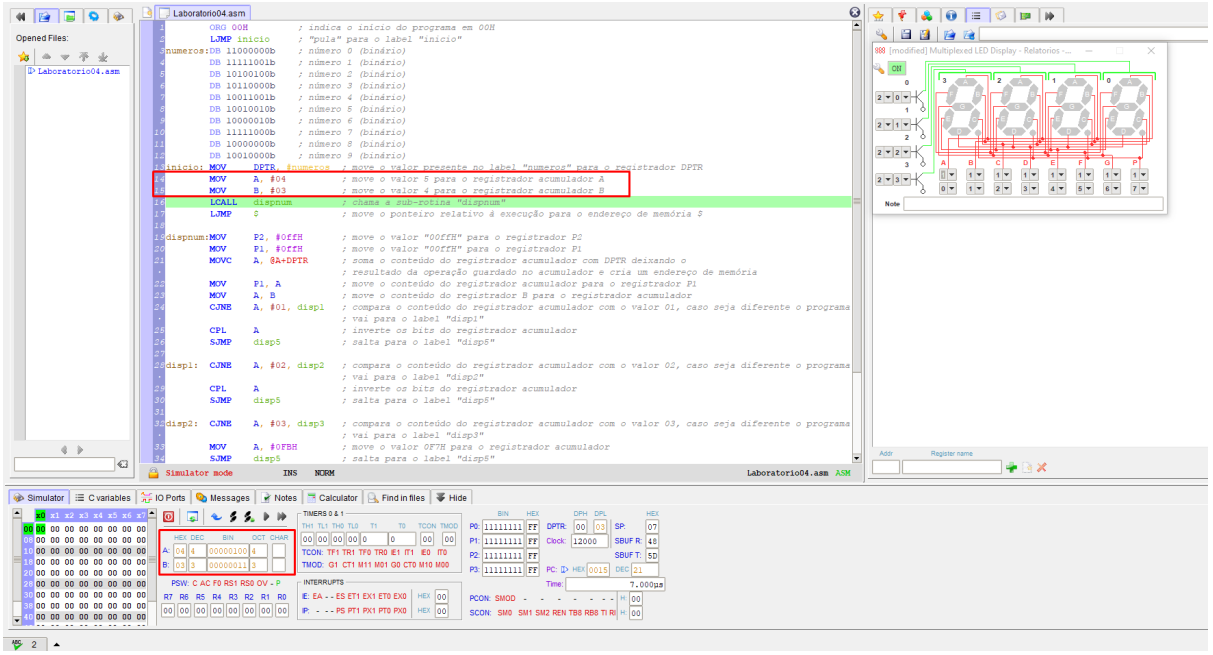
No primeiro momento do programa, ao executar as duas primeiras linhas, o programa será iniciado em 00 em hexadecimal e o comando “LJMP inicio” vai desviar o ponteiro de execução do programa para o endereço de memória definido pela label inicio.



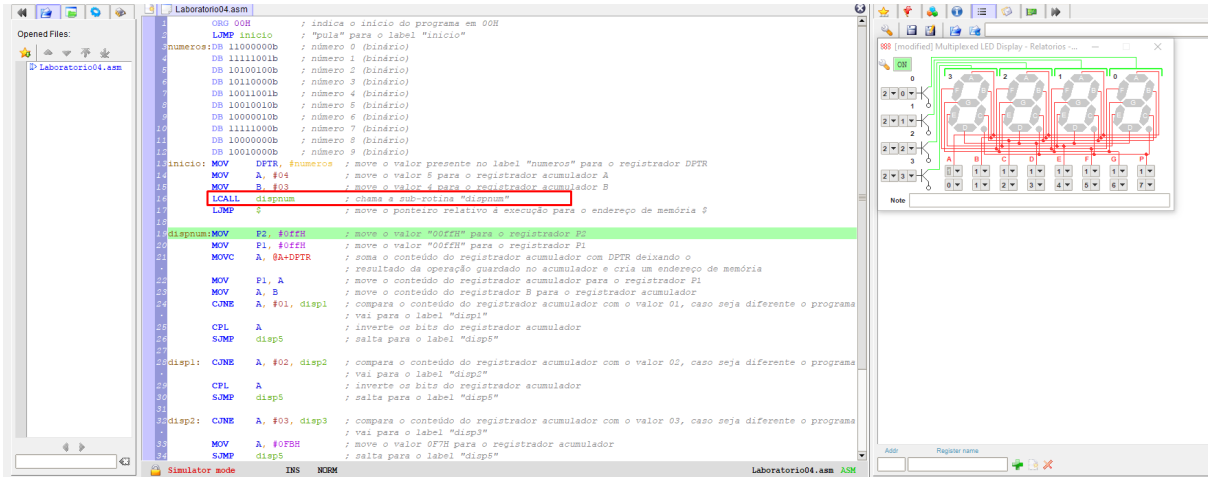
No segundo momento do programa, o label “inicio” vai se tornar o protagonista, executando o comando “MOV DPTR #numeros”, que vai mover o valor presente em #numeros (label do programa com a definição dos dígitos de 0 a 9 em binário).

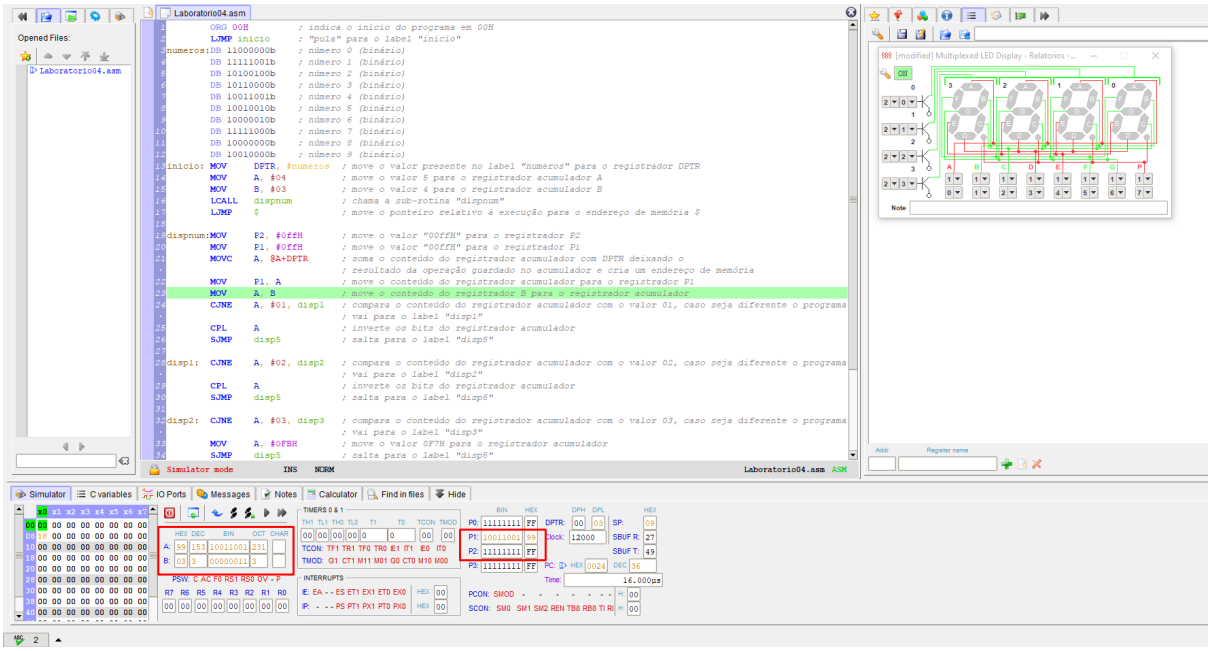


No terceiro momento, o valor 4 é motivo imediatamente para o registrador acumulador A e o valor 3 para o acumulador B.

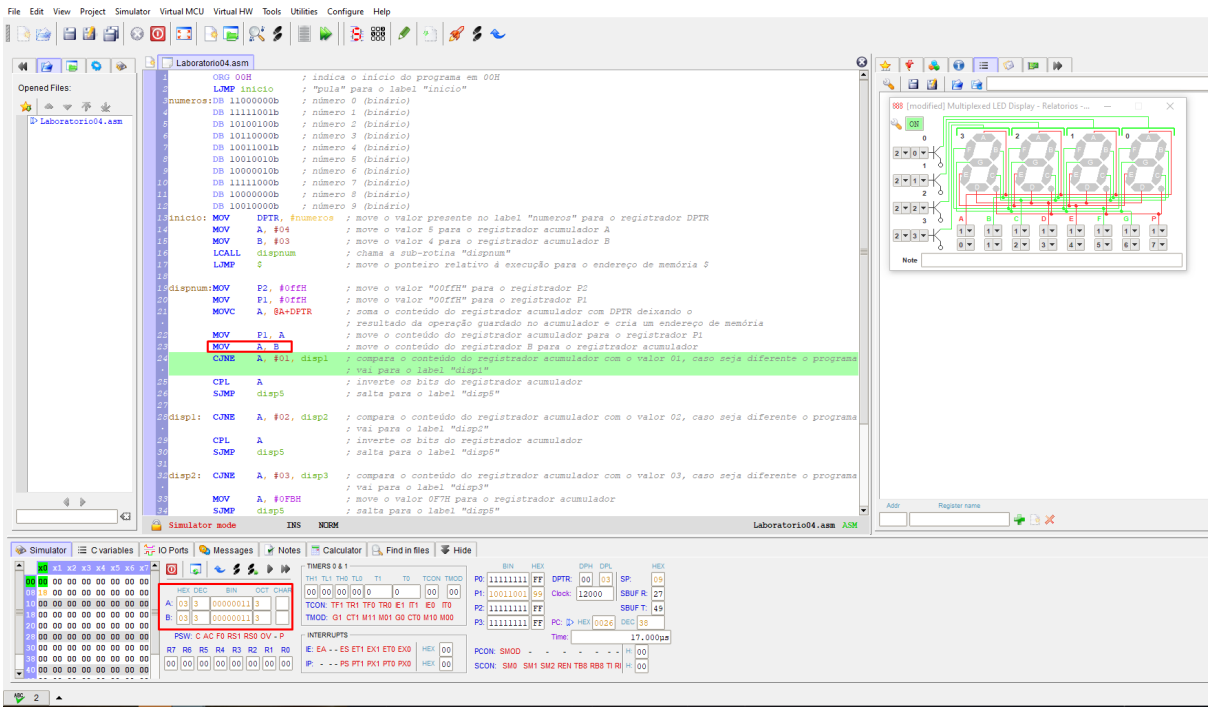


No quarto momento, o programa vai para a seção definida pelo label “dispnum” a partir do comando “LCALL dispnum”. Além disso o valor 0ff é movido para os registradores P1 e P2, enquanto que o comando “MOVC A, @A+DPTR” vai somar o conteúdo presente no registrador acumulador com o conteúdo de DPTR movendo o resultado para o acumulador A.

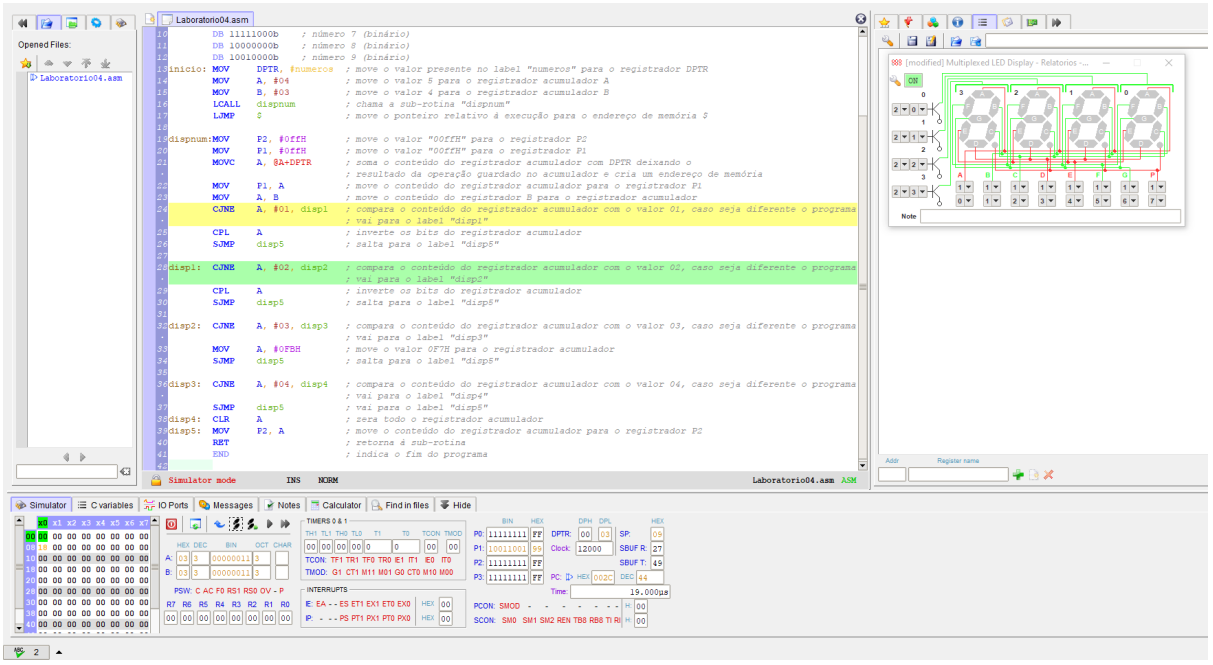




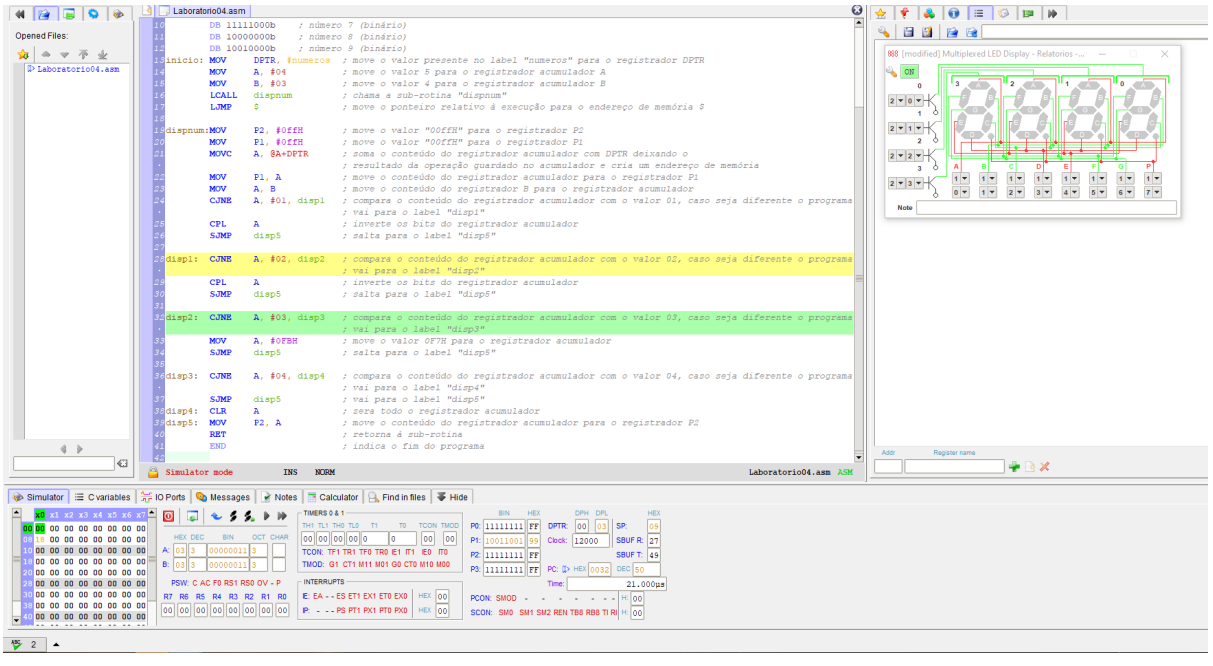
Logo após, o conteúdo do registrador acumulador B é movido para o registrador acumulador A.



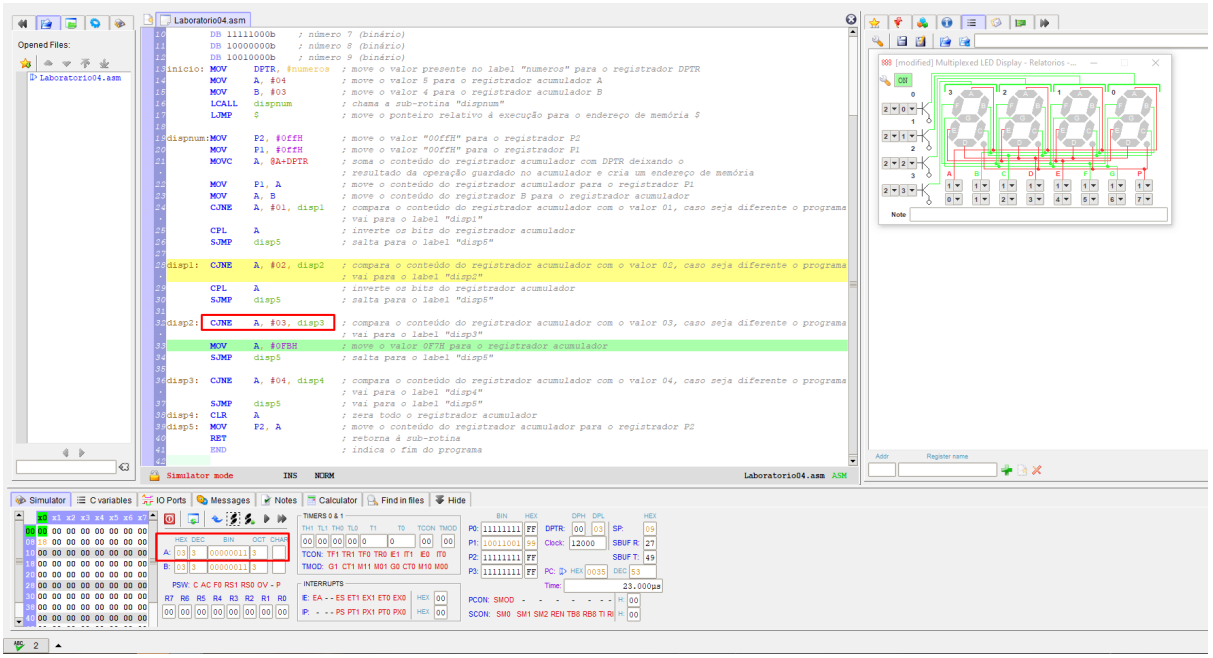
Então o comando “CJNE A, #01, disp1” compara o conteúdo do acumulador A com o número 01, que caso seja diferente o programa vai ser redirecionado para o label “disp1”, acontecimento que é presente.



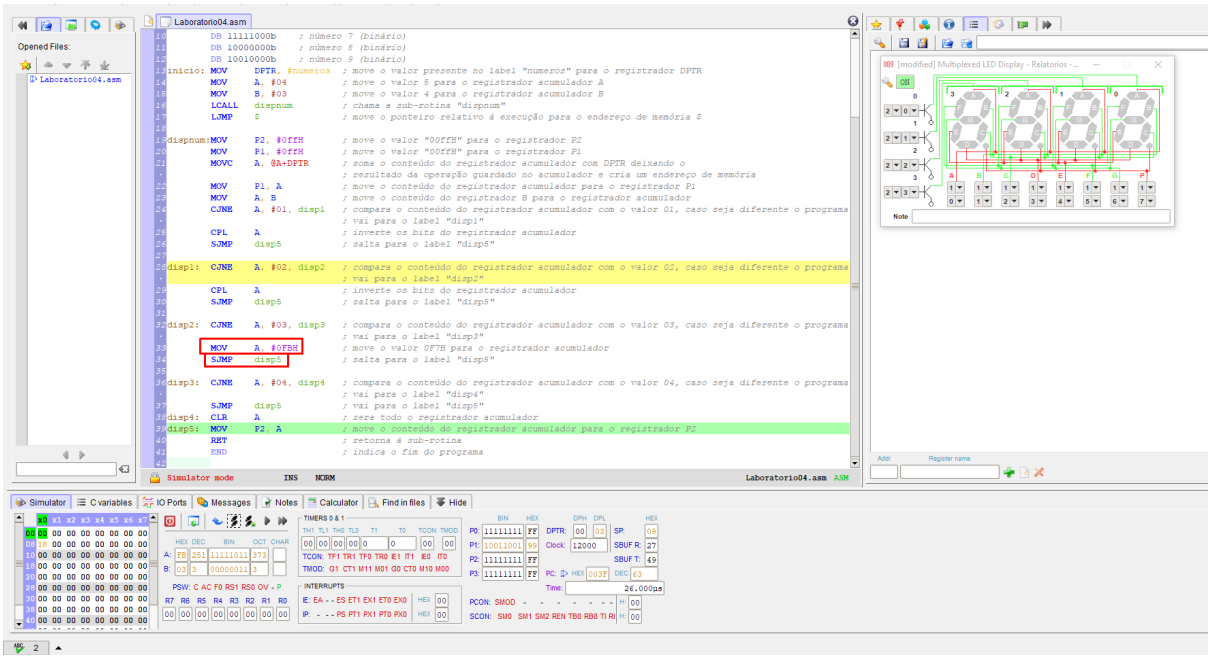
Após a comparação, o mesmo irá ocorrer no segundo trecho do código, sendo que a comparação que será realizada é com o valor 2, contudo o conteúdo do registrador acumulador A é diferente de 2, por isso o programa irá pular para o label “disp2”, devido ao comando “CJNE A, #02, disp2”.



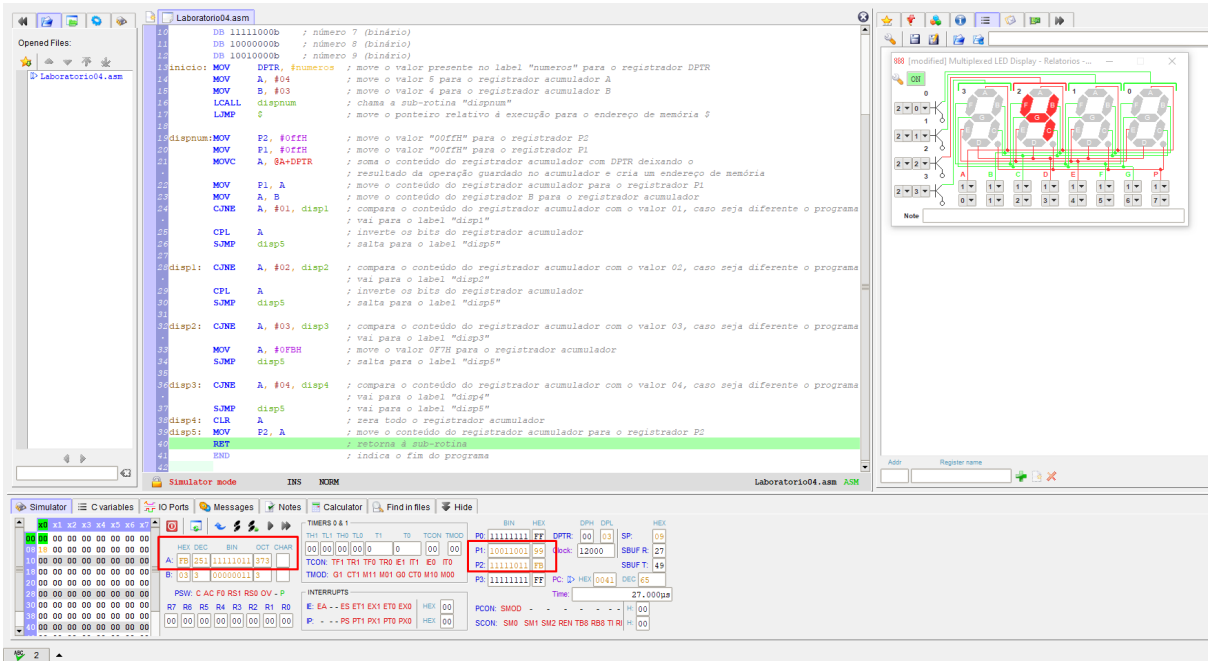
Então finalmente, o nesse trecho da execução, a partir do comando “CJNE A, #03, disp3” é realizado que o conteúdo do registrador acumulador A é comum com o valor 3, então o programa irá dar continuidade no conteúdo guardado no label “disp2”.



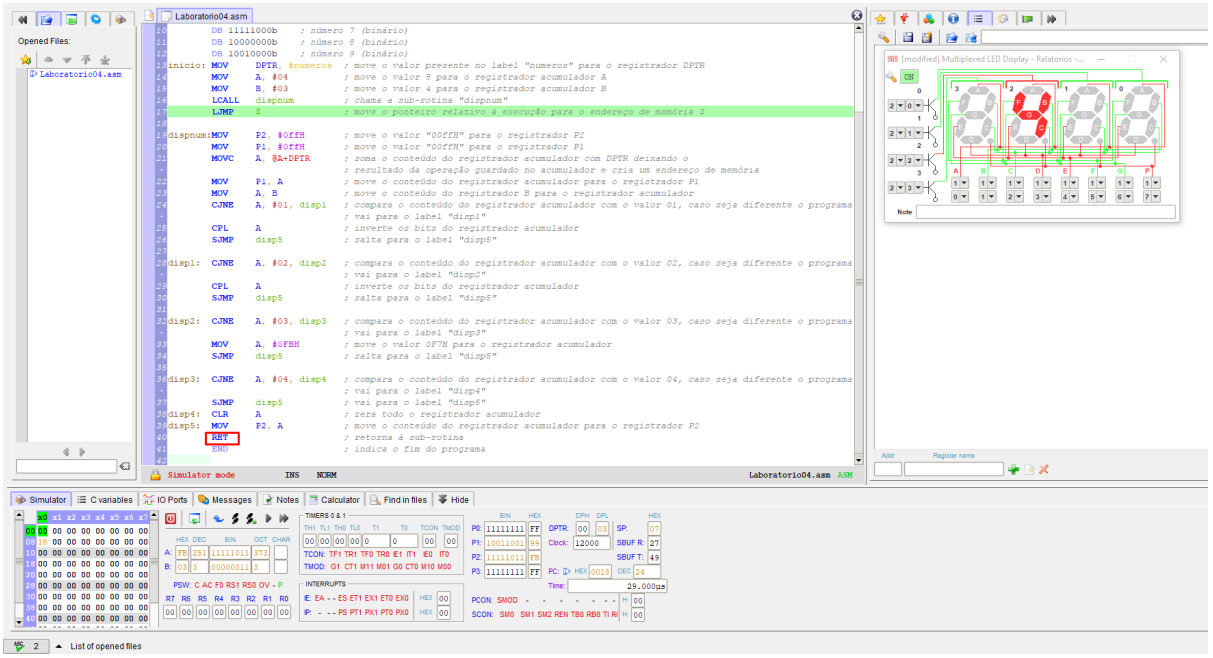
Após a execução do trecho de código contido no labem “disp2” que irá mover o valor OFb para o registrador acumulador a partir do comando “MOV A, #0FBH, o comando “SJMP disp5” vai fazer o programa ir para o label “disp5”.



Então, após o comando “MOV P2, A”, o conteúdo do registrador acumulador é movido até o registrador P2 que irá mostrar o número 4 a partir do acendimento dos leds. Já no registrador P1 o número 10011001 vai indicar que os bits com 0 estarão ligados e os bits com 1 estarão desligados.



Por fim, o comando “RET” irá retornar o programa para a sub rotina que irá iniciar com o comando “LJMP $” que fará com que o número 4 continue sendo apresentado no display.



4. RESULTADOS E DISCUSSÃO

Após todo o experimento foi possível compreender algumas peculiaridades da linguagem a nível de máquina “Assembly”. Após a execução do código listado foi possível compreender a relação entre as operações realizadas e o painel de LED 's de 7 segmentos.

5. CONCLUSÕES

Foi possível concluir que a linguagem Assembly em atuação ao microcontrolador MCU 8051 pode dar uma vasta biblioteca de opções e atuações diante de seus registradores e portas, assim como demonstrado no decorrer do relatório com o exemplo do painel de LED 's de 7 segmentos.

6. REFERÊNCIAS BIBLIOGRÁFICAS

NETO, Hugo Vieira. MICROCONTROLADORES MCS51. Curitiba, 2002. Disponível em: https://pessoal.dainf.ct.utfpr.edu.br/hvieir/download/mcs51.pdf. Acesso em: 24 jul. 2021.

MORAIS, Misael. Organização e arquitetura de computadores. [S. l.], . 2021. Disponível em: https://drive.google.com/drive/folders/0Bwjlecok7TpyfnAtQmx6bE4yZ043amNsbnRxMkF4UFlpWVZhWmRfeVBSeHRRVi1xRzNOZnM?resourcekey=0-WmQ6S1i6hLYyCoGBLbMeGw. Acesso em: 01 ago. 2021