

**DISCIPLINA: LABORATÓRIO ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES**

**RELATÓRIO: LABORATÓRIO 03**

**(2022)**

**ALUNO: Lucas Fausto Medeiros**

**Professor.: Dr. Misael Elias de Morais**

CAMPUS CAMPINA GRANDE, julho de 2022

1. Resumo:

Neste Experimento foi possível visualizar alguns dos conceitos base e importantes na programação baixo nível dos micro-controladores (MCS51), Assembly, pelo Simulador MCY8051 IDE. Ademais, a atividade que foi proposta no Lab. em questão, foi a demonstração do conjunto de instrução do microcontrolador e seus equipamentos. Neste, fizemos o acendimento de um display de sete segmentos com a técnica de multiplexação e o uso de um kaypad.

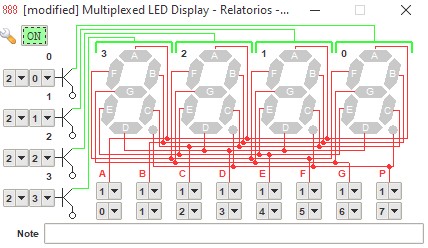
1. Introdução:

O Laboratório III tem o objetivo apresentar o estudo da linguagem Assembly a partir de consultas em seu DataSheet, que nada mais é que uma folha com dados e especificações técnicas de seu desempenho levando em conta o microcontrolador MCS51.

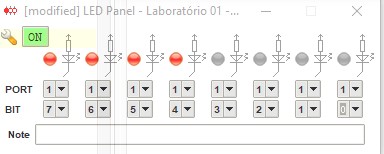
1. Métodos e Materiais:
   1. Objetivo: O experimento teve como objetivo mostrar na pratica o estudo pratico da programação em Assembly, que se fez possível estudar display de 7 segmentos com a técnica de multiplexação além da leitura de dados no simulador.

* 1. Hardware e Software necessários: Computador com o MCU8051 IDE instalado.

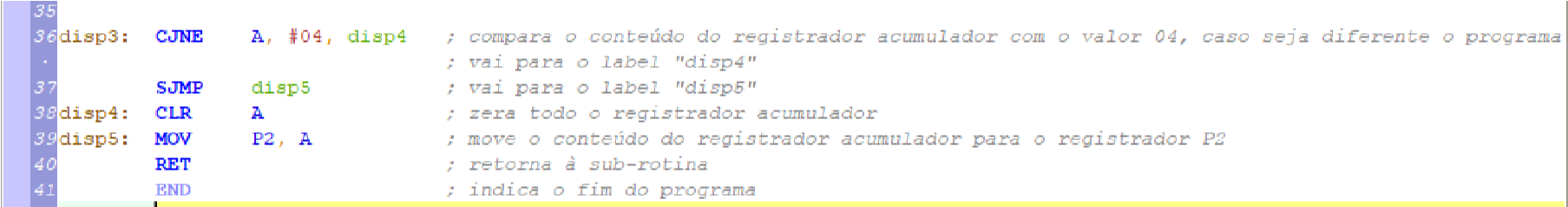
* 1. Procedimentos:
     1. Ligação do display de 7 segmentos: Para a ligação dos bits de 0 a 7 inferiores fora utilizada a porta **P1** e a ligação dos bits laterais de 0 a 3 laterais, utilizou a porta **P2**. Como mostra a imagem abaixo:



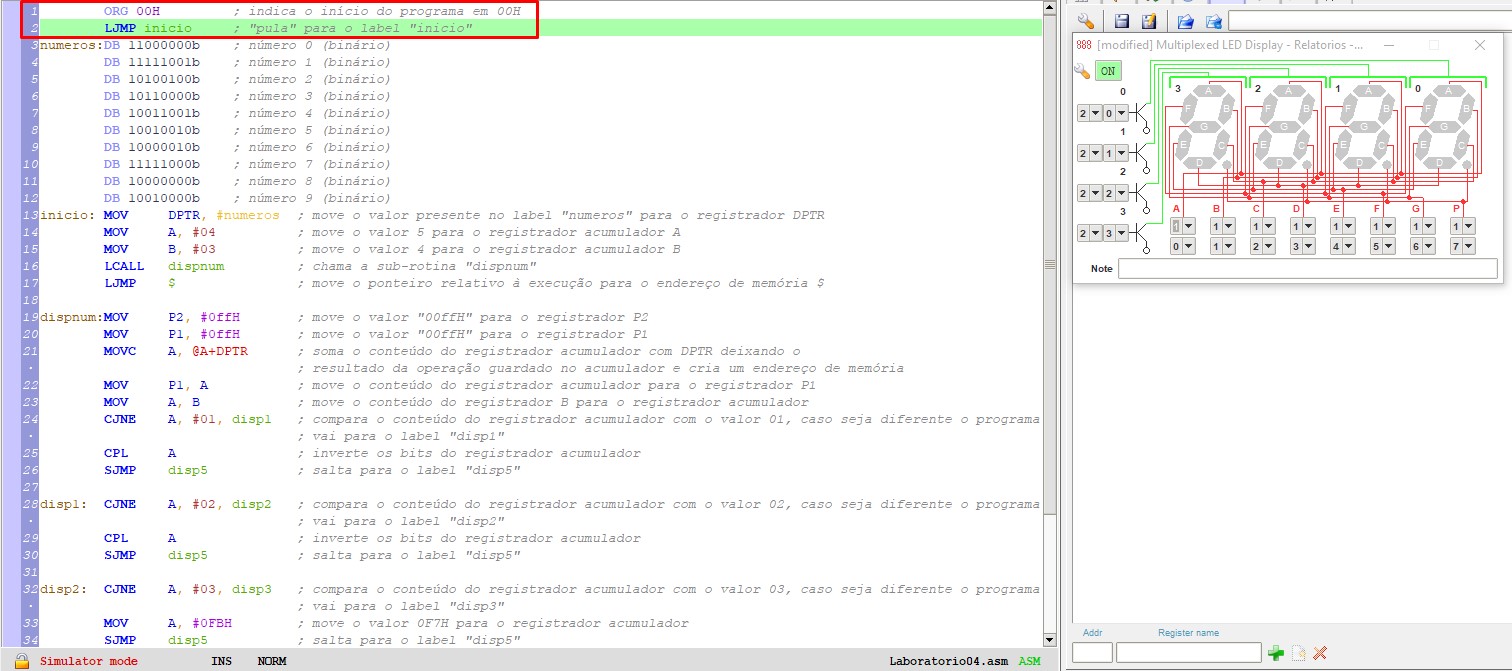
A ativação do painel é feita ativando e relacionando cada um dos oito led’s a uma porta e um bit do microcontrolador.



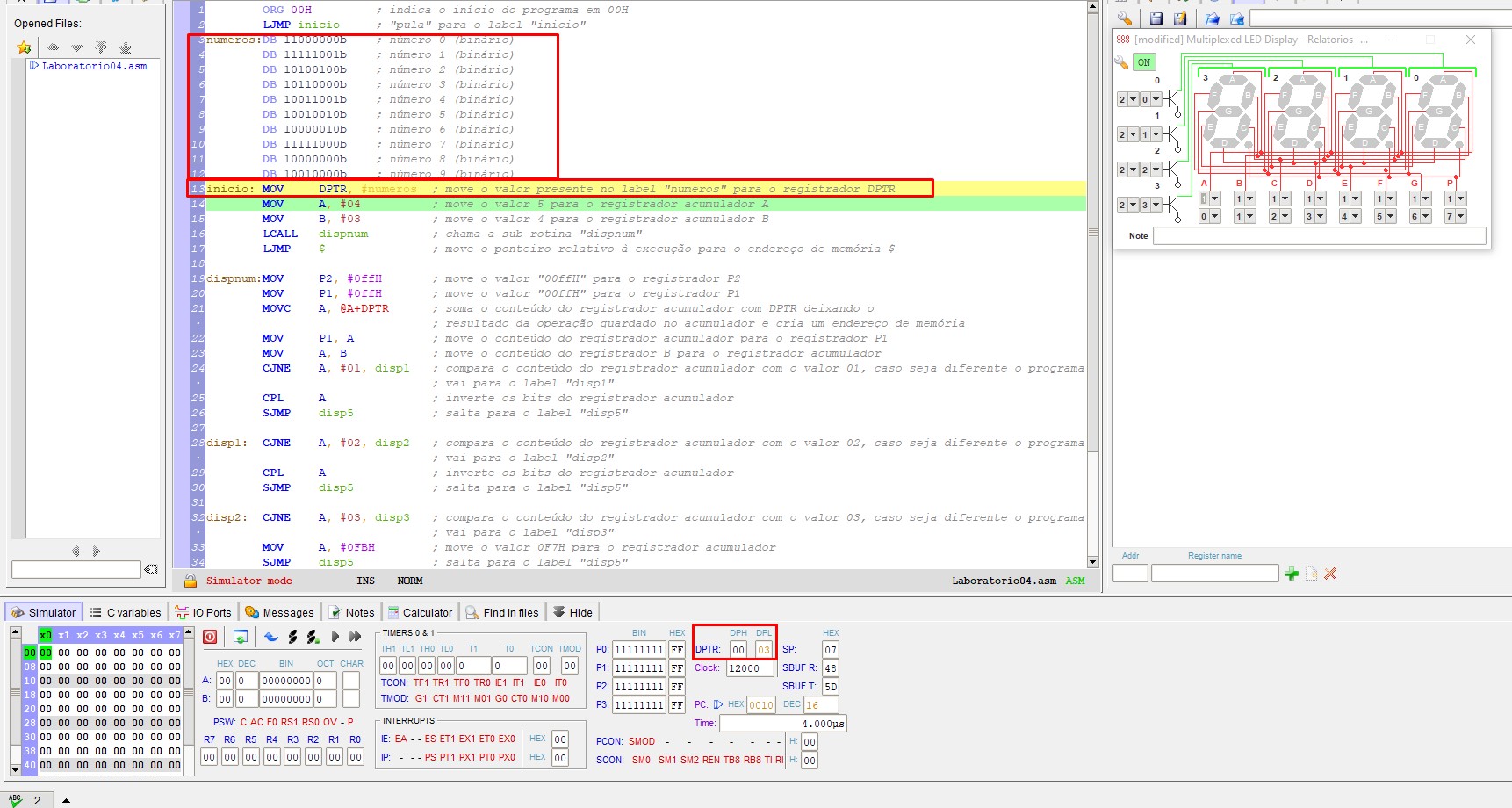
* + 1. Listagem do programa em LM:
       1. **ORH 00H:** Pseudo-instrução que define o endereço que o programa irá começar e continuará.
       2. **LJMP INICIO:** Faz um salto para o label inicio.
       3. **NUMEROS: DB 11000000 b:** Instancia um label. Representando o número 0 em binário.
       4. **DB 11111001 b:** Representa o número 1 em binário.
       5. **DB 10100100 b:** Representa o número 2 em binário.
       6. **DB 10110000 b:** Representa o número 3 em binário.
       7. **DB 10011001 b:** Representa o número 4 em binário.
       8. **DB 10010010 b:** Representa o número 5 em binário.
       9. **DB 10000010 b:** Representa o número 6 em binário.
       10. **DB 11111000 b:** Representa o número 7 em binário.
       11. **DB 10000000 b:** Representa o número 8 em binário.
       12. **DB 10010000 b:** Representa o número 9 em binário.
       13. **INICIO: MOV DPTR, #NUMEROS:** Move o valor do label NUMEROS para o registrador DPTR.
       14. **MOV A, #04:** Move o valor 5 para o registrador Acumulador A.
       15. **MOV B, #05:** Move o valor 4 para o registrador Acumulador B.
       16. **LCALL DISPNUM:** Chama a sub-rotina DISPNUM.
       17. **LJMP $:** Move o ponteiro relativo da execução para o endereço de memória $.
       18. **DISPNUM: MOV P2, #0FFH:** Instancia o label DISPNUM. Move para o valor #0FFH para o registrador P2.
       19. **MOV P1, #0FFH:** Move o valor #0FFH para o registrador P1.
       20. **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.
       21. **MOV P1, A:** Move o valor de registrador Acumulador para p1.
       22. **MOV A, B:** Move o valor do registrador B para o registrador Acumulador.
       23. **CJNE A, #01, DISP1:** Compara o conteúdo do registrador Acumulador com o valor 01, caso sejam diferentes o programa vai até o label DISP1.
       24. **CPL A:** Inverte os bits do registrador Acumulador.
       25. **DISP1: CJNE A, #02, DISP2:** Instancia o label DISP1. Compara o conteúdo do registrador Acumulador com o valor 02, caso sejam diferentes o programa vai para o label DISP2.
       26. **CPL A:** Inverte os bits do registrador Acumulador.
       27. **SJMP DISP5:** Solta o label DISP5.
       28. **DISP2: CJNE A, #03, DISP3:** Compara o conteúdo do registrador Acumulador com o valor 03, caso sejam diferentes o programa vai até o label DISP3.
       29. **MOV A, #0FBH:** Move o valor 0FBH para o registrador Acumulador.
       30. **DISP3: CJNE A, #04, DISP4:** Compara o conteúdo do registrador acumulador com o valor 04, caso sejam diferentes o programa vai para o label DISP4.
       31. **SJMP DISP5:** Salta até o label DISP5.
       32. **DISP4: CLR A:** Zera o contudo do registrador Acumulador.
       33. **DISP5: MOV P2, A:** Move o conteúdo do registrador Acumulador para o registrador P2.
       34. **RET:** Retorna a sub-rotina.
       35. **END:** Fim do Programa.
    2. Funcionamento do programa: Descrição com detalhes de cada linha do programa que foi utilizado no LAB. Com comentário de cada linha.



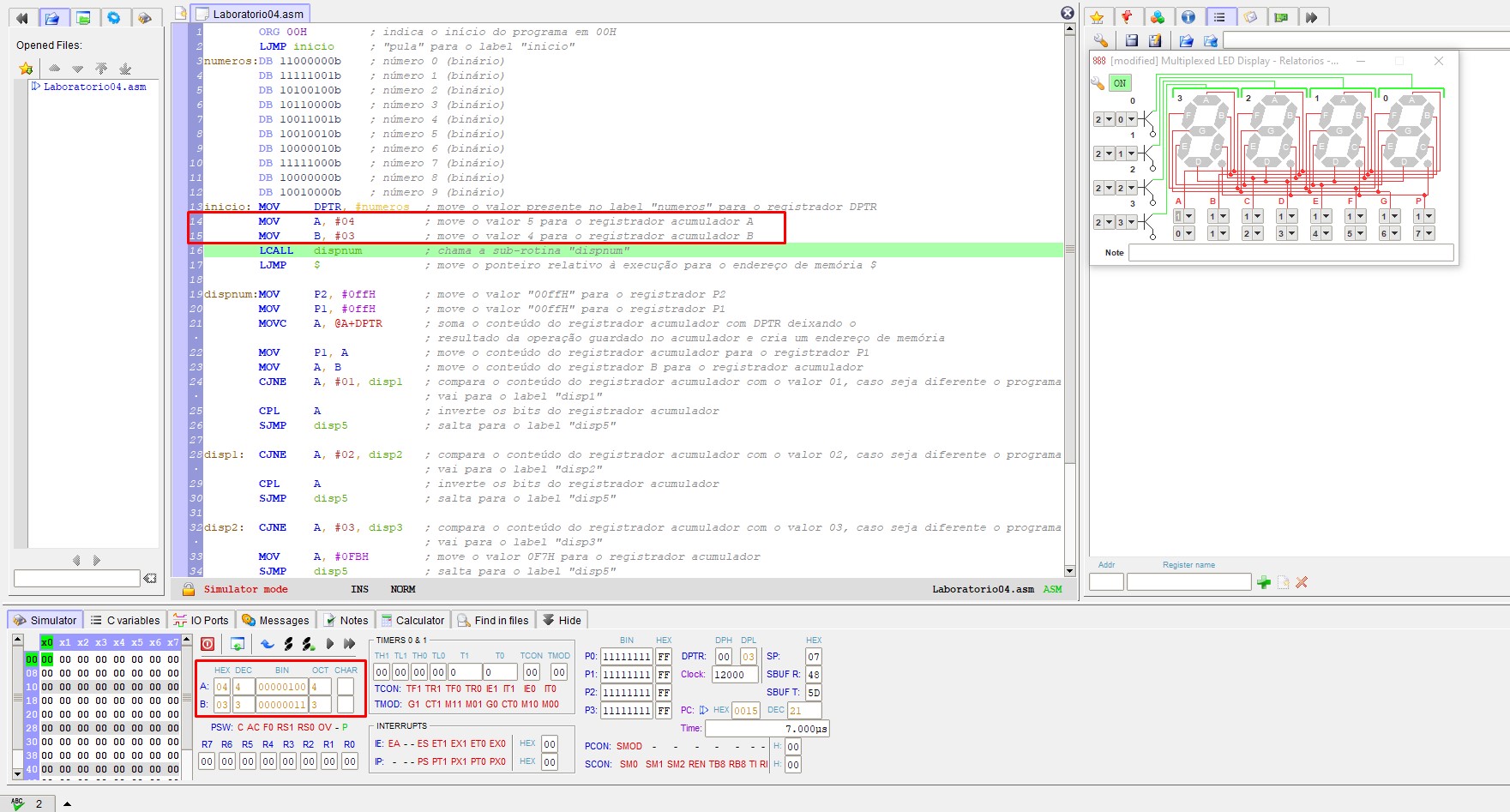
No primeiro passo, o programa irá executar as duas primeiras linhas, o programa irá iniciar em 00 em hexadecimal, o comando **LJMP INICIO** vai desviar o ponteiro de execução do programa para o endereço de memória definido pelo label **INICIO.**



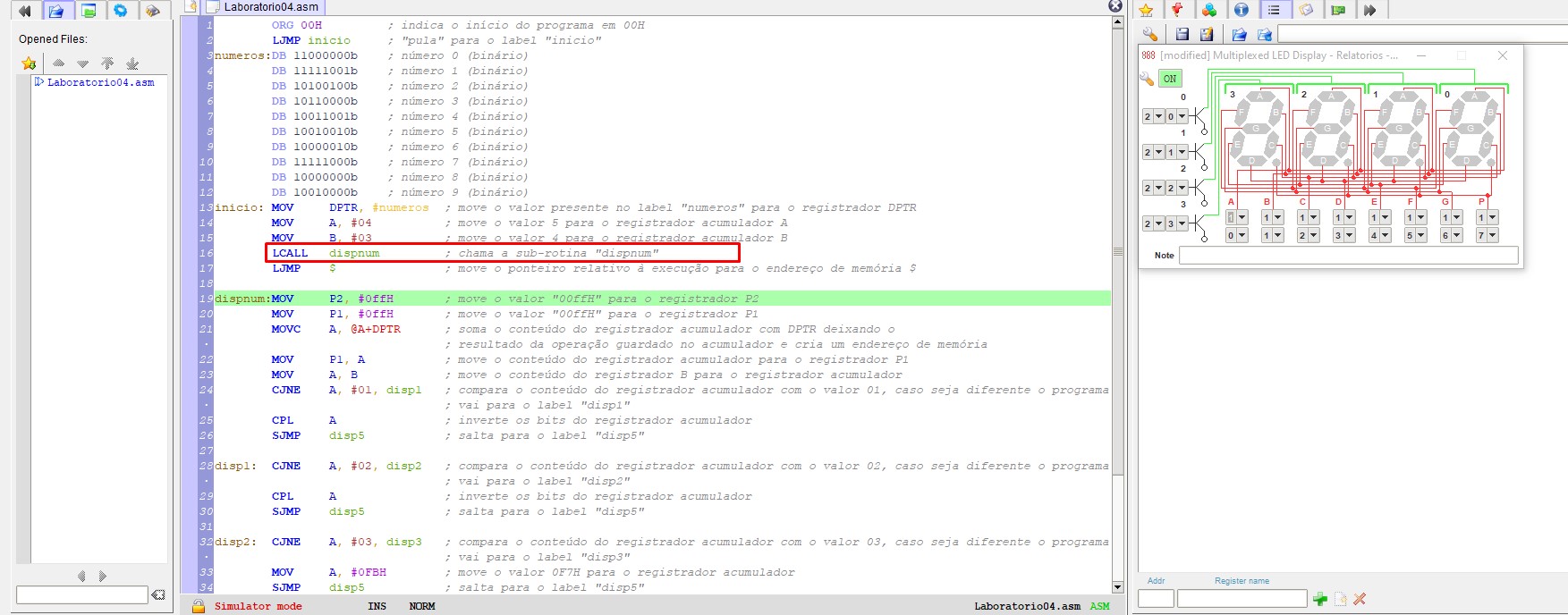
No segundo passo, o programa ao chegar no label **INICIO**, ele vai se tornar o protagonista, executando o comando **MOV DPTR #NUMEROS**, que vai mover o valor presente em **#NUMEROS**, label do programa com a definição dos dígitos de 0 – 9 em binário.

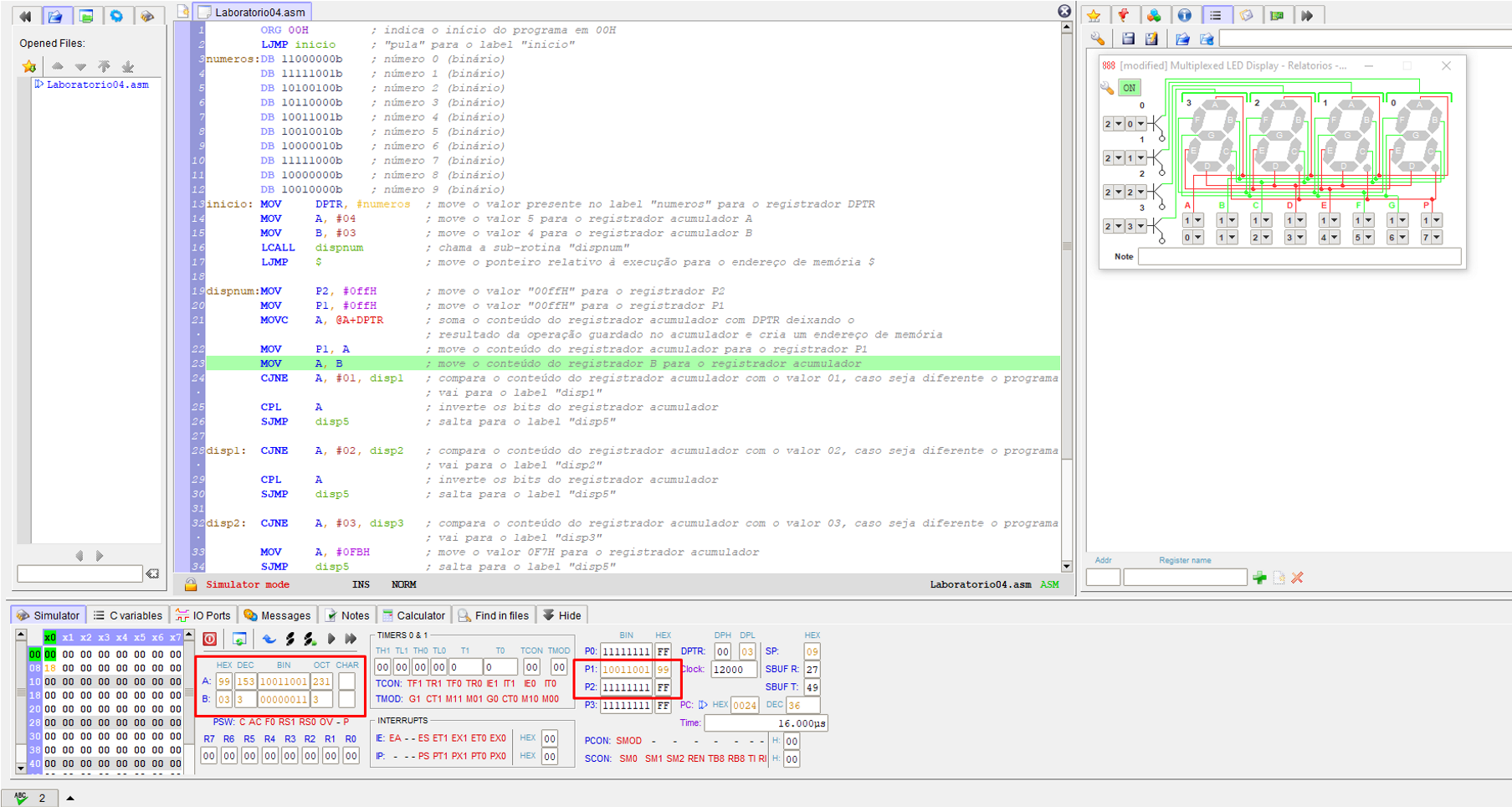


No terceiro passo, o valor 4 é motivo imediato para o registrador Acumulador A e o valor 3 para o Acumulador B.

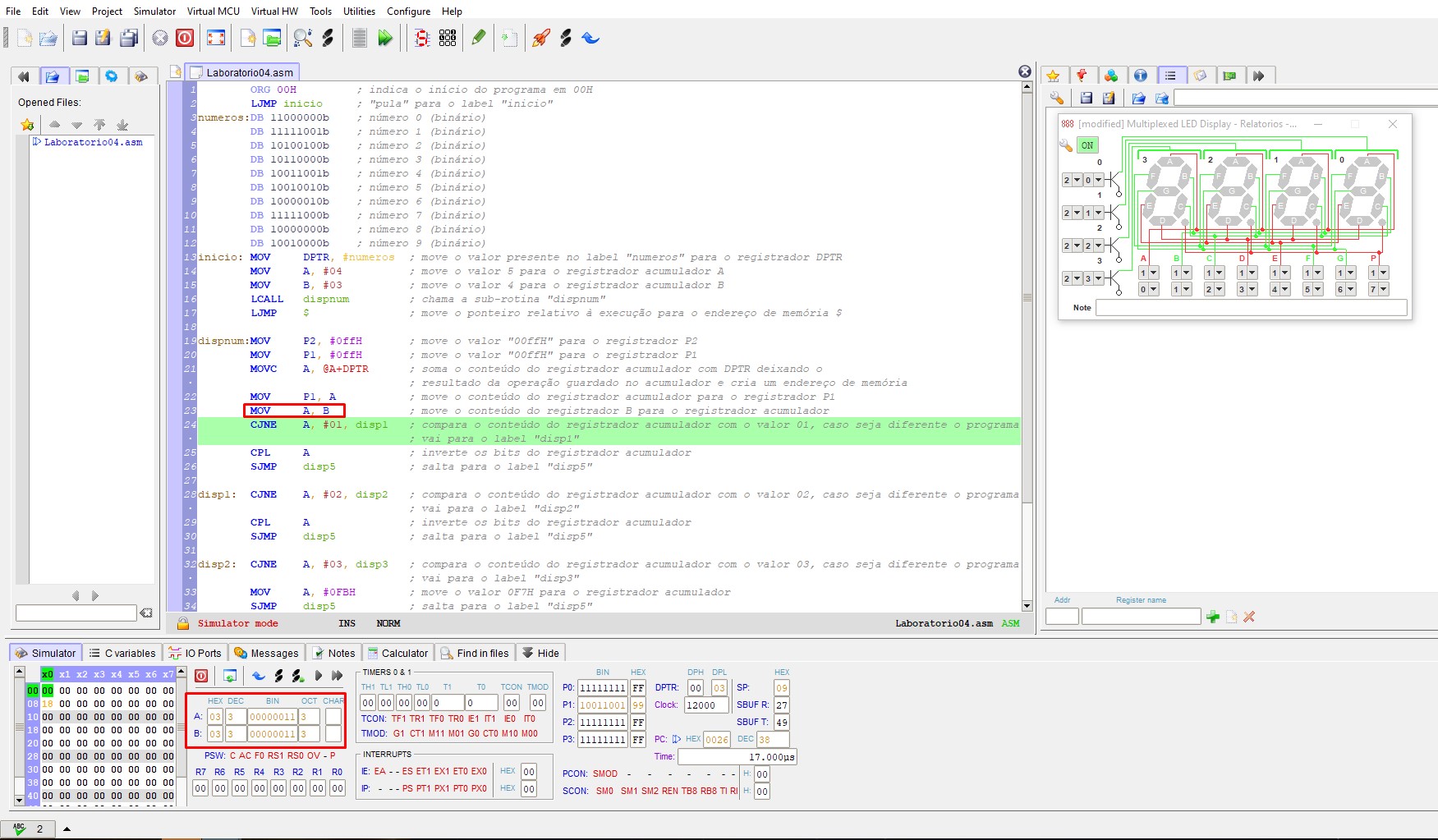


No quarto momento, o programa vai para a seção definida pelo label **DISPNUM** a partir do comando **LCALL DISPNUM.** Ademais, o valor 0FF é movido para os registradores **P1** e **P2**, enquanto que o comando **MOVC A, @A+DPTR** vai somar o conteúdo presente no registrador Acumulador com o conteúdo de **DPTRN** movendo o resultado para o acumulador A.

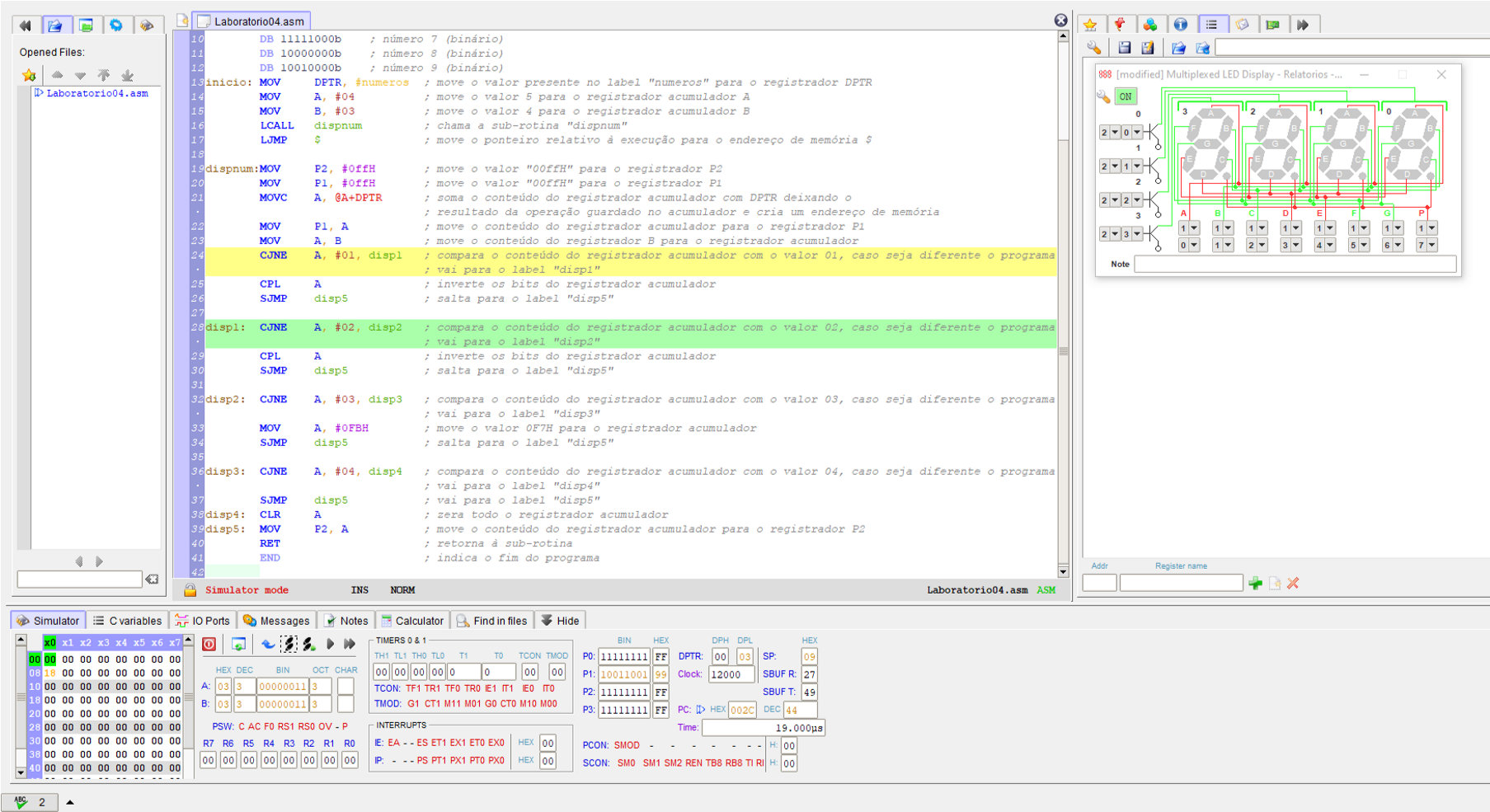




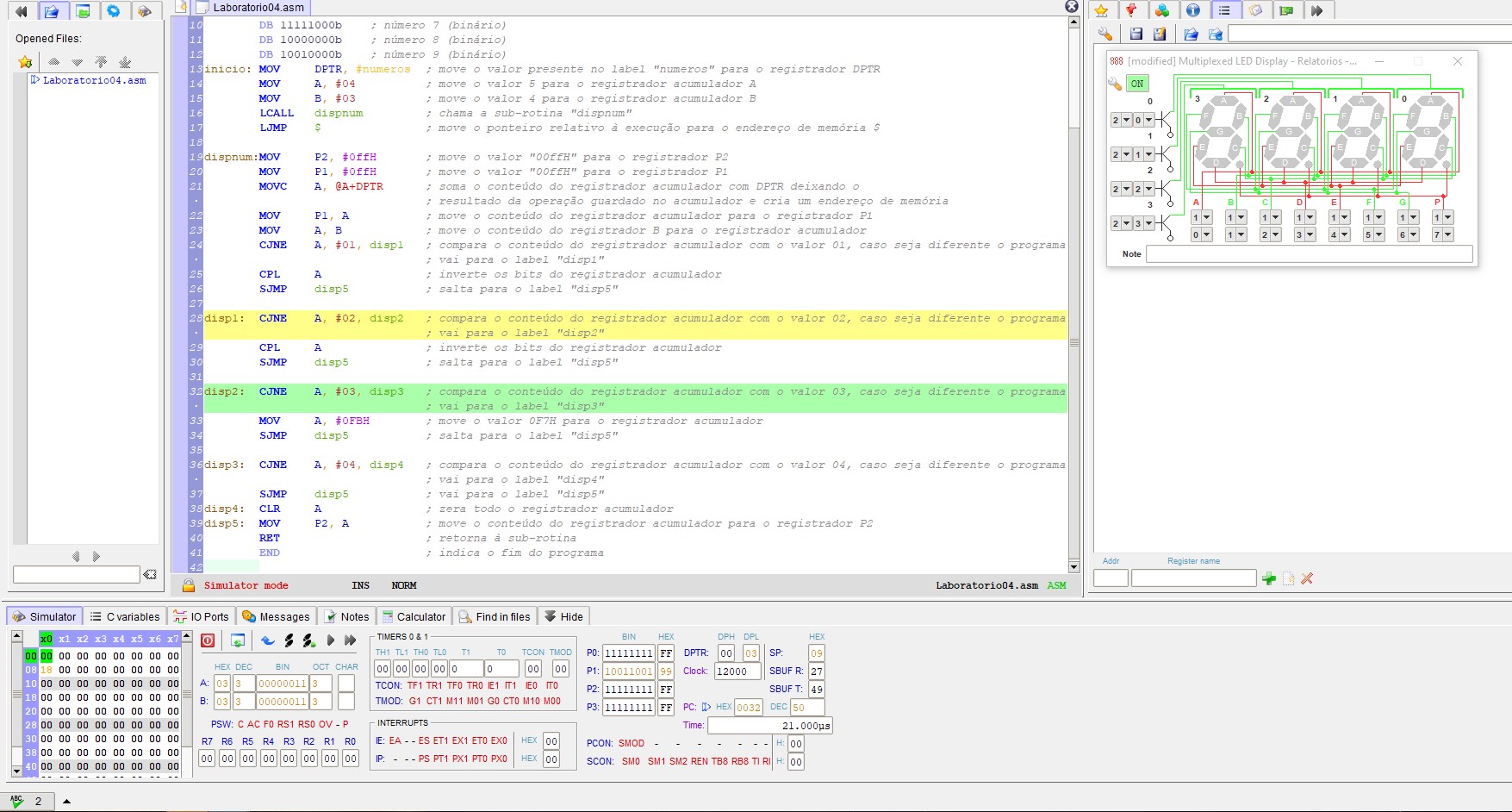
Logo após, o conteúdo do registrador Acumulador B é movido para o registrador Acumulador A.



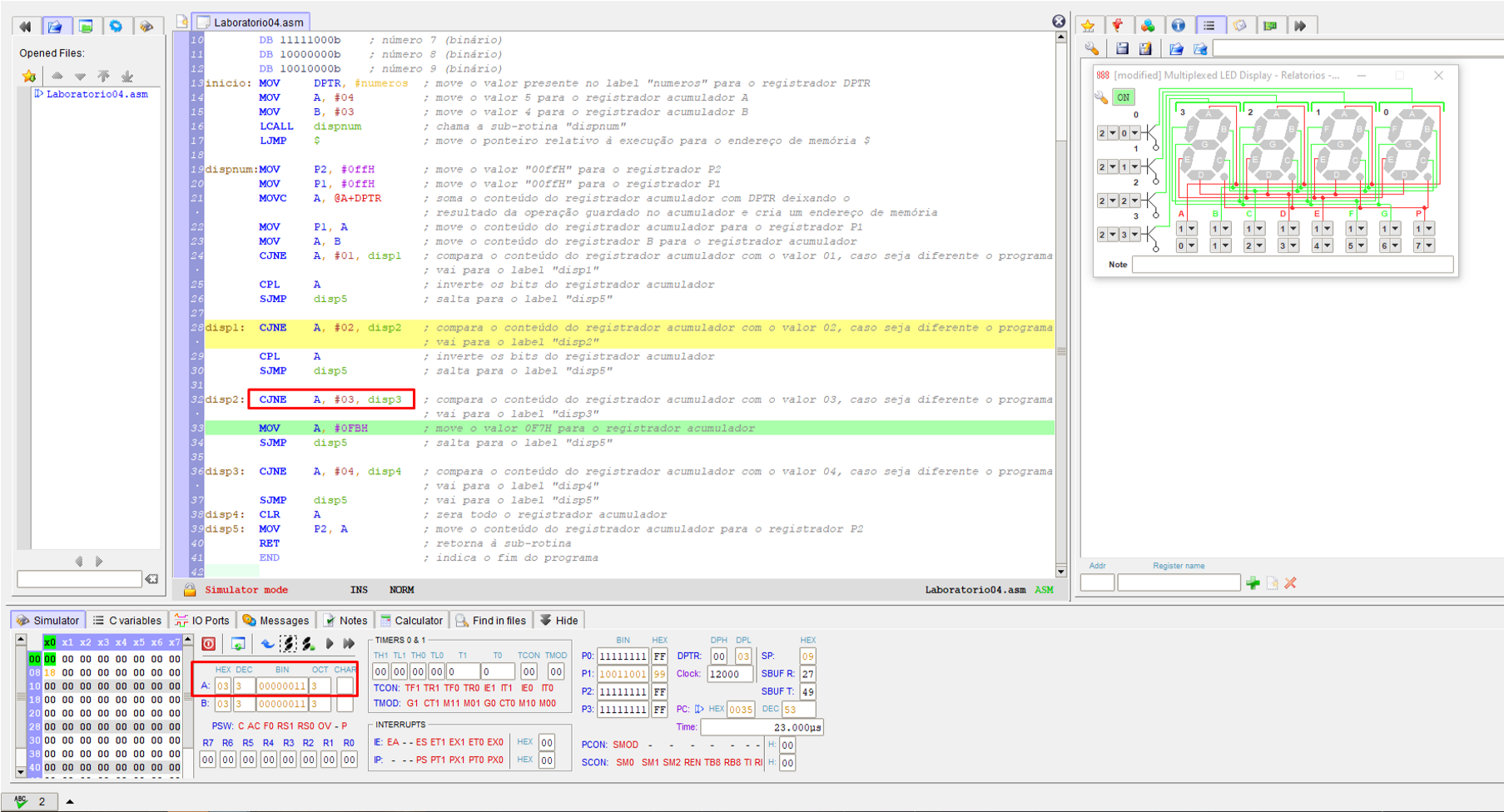
Após, o comando **CJNE A, #01, DISP1** compara o conteúdo do Acumulador A com o número 01, que caso sejam diferentes o programa vai ser direcionado para o label **DISP1N.**



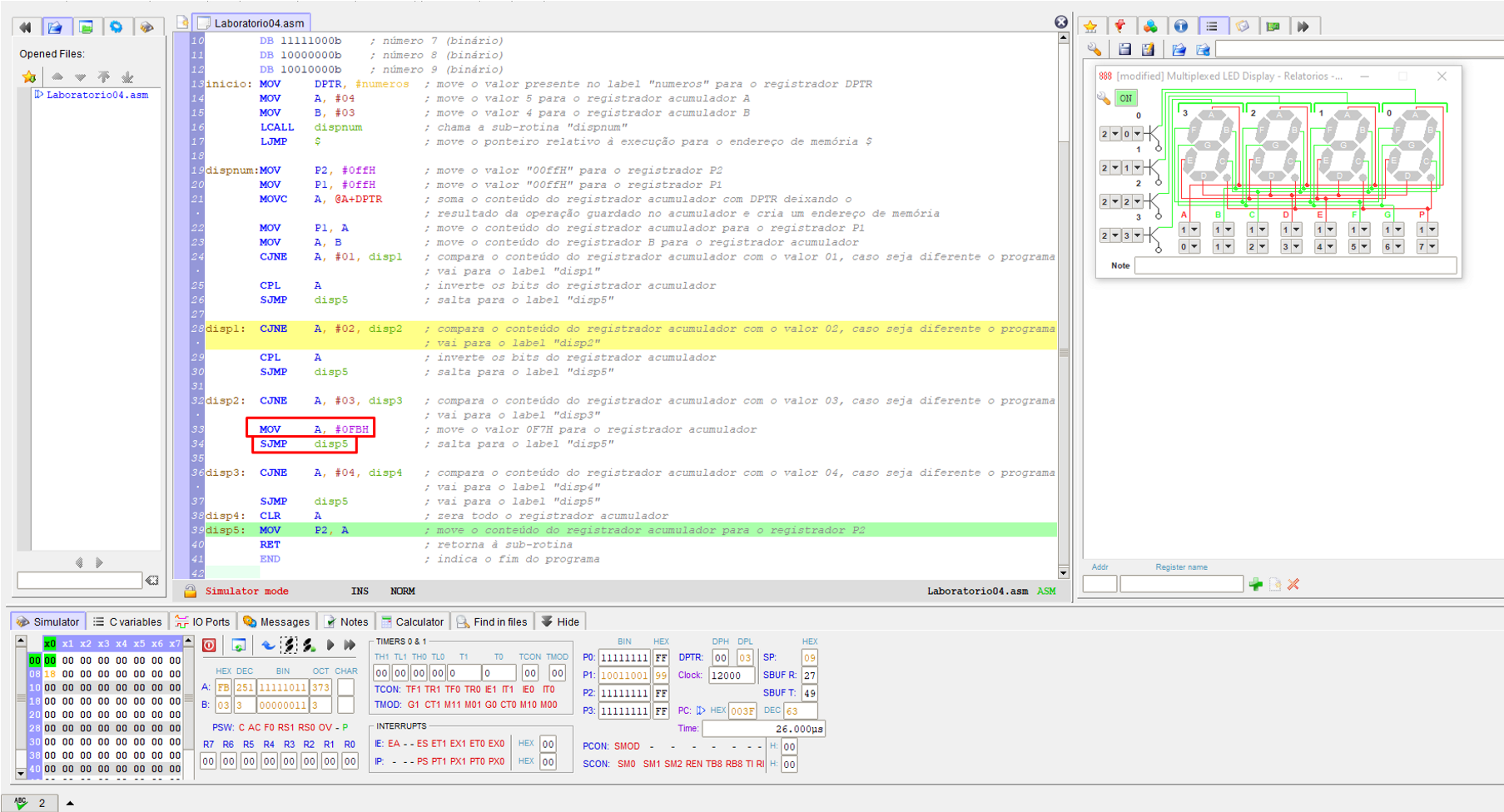
Logo após a comparação, o mesmo vai acontecer no segundo trecho do código, sendo que a comparação que será realizada é com o valor 02, com isso, o conteúdo do registrador Acumulador A é diferente de 02, então o programa vai pular para o label **DISP2,** devido ao comando **CJNE A, #02, DISP2.**



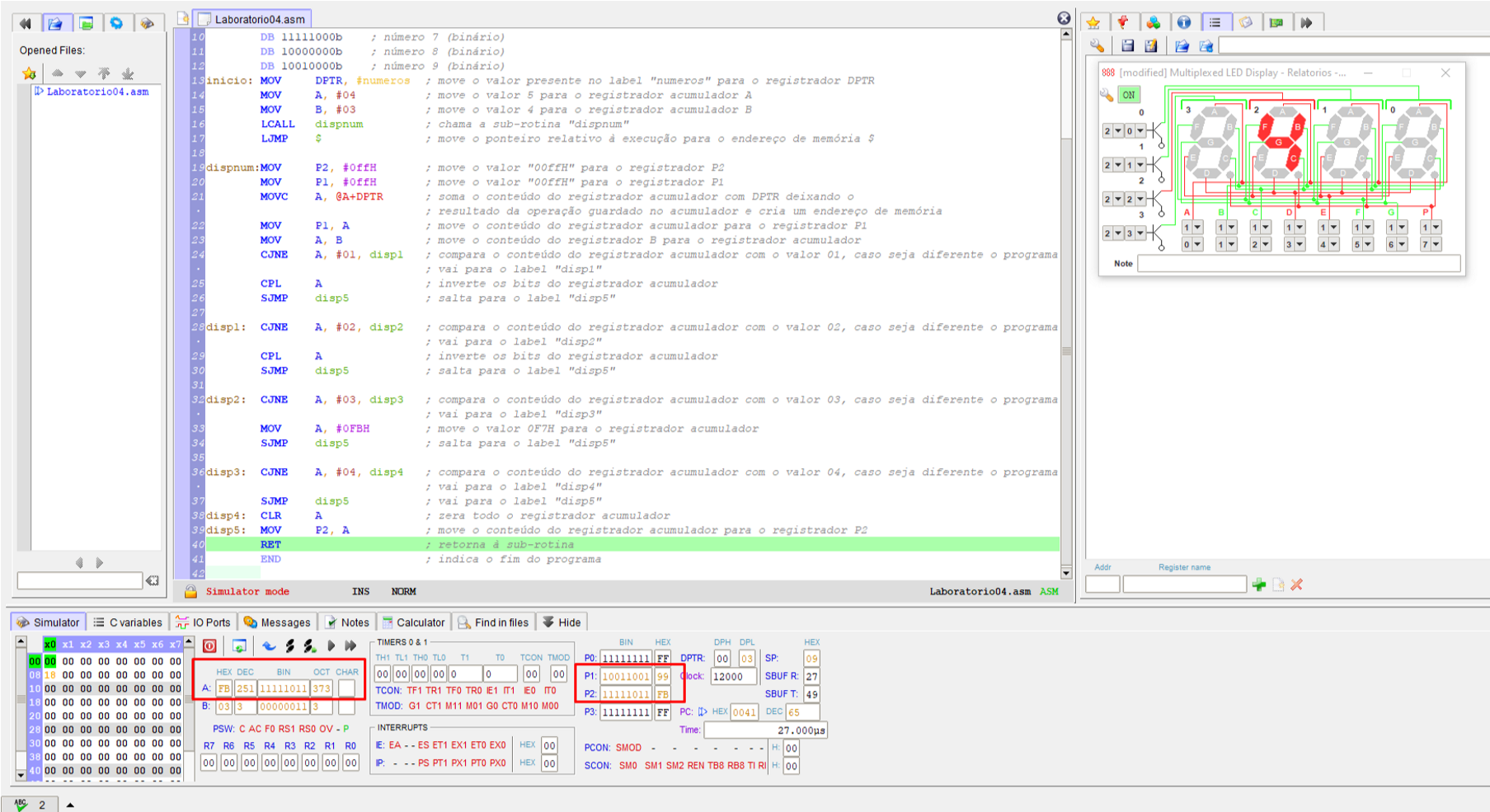
Com fim, no presente trecho do código, a partir do comando **CJNE A, #03, DISP3** é realizado que o conteúdo do registrador Acumulador A é comum com o valor 3, então o programa irá dar continuidade no conteúdo guardado no label **DISP2.**



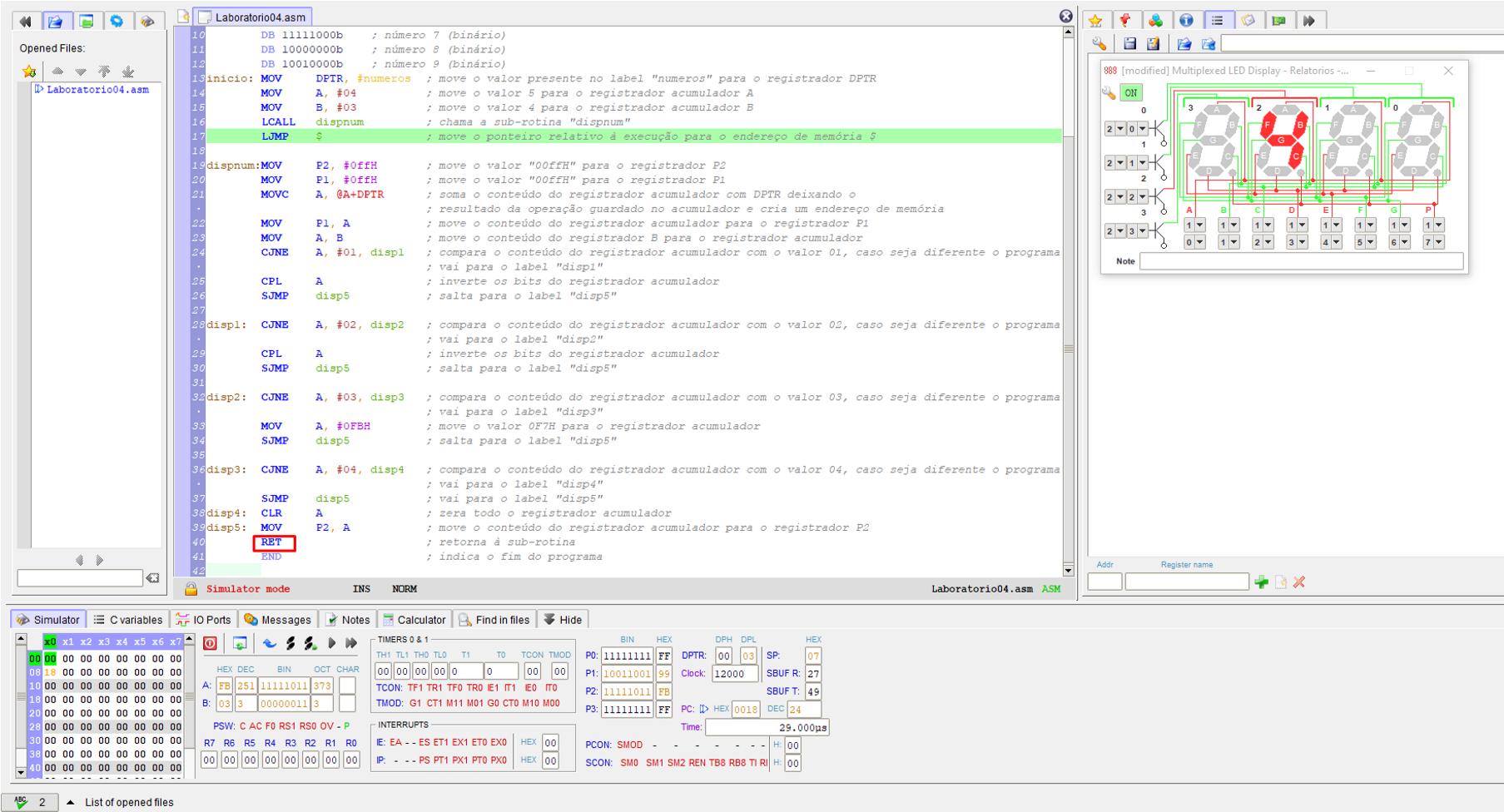
Logo a execução do trecho de código contido no label **DISP2** que move o valor **0FB** para o registrador Acumulador a partir do comando **MOV a, #0FBH**, o comando **SJMP DISP5** vai fazer o programa ir para o label **DISP5**.



Contudo, depois do comando **MOV P2, A**, o conteúdo do registrador acumulador é movido para o registrador **P2** que vai mostrar o número 04 a partir do acendimento dos led’s. Com isso, 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** vai retornar o programa para a sub-rotina que irá iniciar com o comando **LJMP $**, no qual, vai fazer com que o número 4 continue sendo apresentado no display.



1. Resultados e Discussão: Ao terminar todo o experimento, é possível compreender de forma pratica o melhor funcionamento da linguagem Assembly e, usar periféricos disponíveis no simulador MCU 5081 IDE, e suas interações.

1. 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 com o exemplo do painel de LED 's de 7 segmentos.
2. Referencias:
   1. MORAIS, Misael. Organização e arquitetura de computadores. [ S. l.], . 2021. Disponível em: https://drive.google.com/drive/folders/0Bwjlecok7TpyfnAtQmx6bE4yZ043amNsbnRxMkF4UFlpWVZhWmRfeVBSeHRRVi1xRzNOZnM?resourcekey=0-WmQ6S1i6hLYyCoGBLbMeGw. Acesso em: 12 jun. 2022