



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

EDILSON DO NASCIMENTO COSTA  
JÚNIOR

DISCIPLINA: LABORATÓRIO DE ORGANIZAÇÃO E ARQUITETURA  
DE COMPUTADORES

EXPERIMENTO 02

**RELATÓRIO - ROTAÇÃO DE LED'S**

CAMPINA GRANDE -  
PB 2022

## SUMÁRIO

1. RESUMO.....	3
2. INTRODUÇÃO.....	3
3. MATERIAL E MÉTODOS.....	3
3.1. OBJETIVO.....	3
3.2. SOFTWARE NECESSÁRIO.....	3
3.3. HARDWARE NECESSÁRIO.....	3
3.4. PROCEDIMENTOS.....	3
3.4.1. LIGAÇÕES DO PAINEL DE LED.....	4
3.4.2. LISTAGEM DO PROGRAMA EM LM.....	5
3.4.3. FUNCIONAMENTO DO PROGRAMA.....	6
3.4.4. FUNCIONAMENTO DO PAINEL DE LED’S.....	11
3.4.5. CONTEÚDO DO ARQUIVO “.lst”.....	13
4. RESULTADOS E DISCUSSÃO.....	15
5. CONCLUSÕES.....	15
6. REFERÊNCIAS BIBLIOGRÁFICAS.....	15

## **1. RESUMO**

Neste trabalho, você pode observar alguns conceitos básicos e importantes sobre a programação do microcontrolador MCS51 em assembler a partir do simulador MCU8051 IDE. Alguns conceitos sobre a rotação de LEDs foram aprendidos com o auxílio de um simulador e comandos em linguagem assembly.

## **2. INTRODUÇÃO**

O presente trabalho tem como objetivo apresentar o estudo do microcontrolador MCS51 utilizando o simulador IDE MCU8051. Tendo o Assembler como linguagem dominante na programação de microcontroladores, também é necessário estudá-lo a pedido em seu Datasheet, que nada mais é do que uma planilha com dados, características técnicas e operacionais do produto levados em consideração, que é no nosso caso é o microcontrolador MCS51.

## **3. MATERIAL E MÉTODOS**

### **3.1. OBJETIVO**

O objetivo do experimento é o de 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 a rotação de led 's no simulador.

### **3.2. HARDWARE NECESSÁRIO**

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

### **3.3. SOFTWARE NECESSÁRIO**

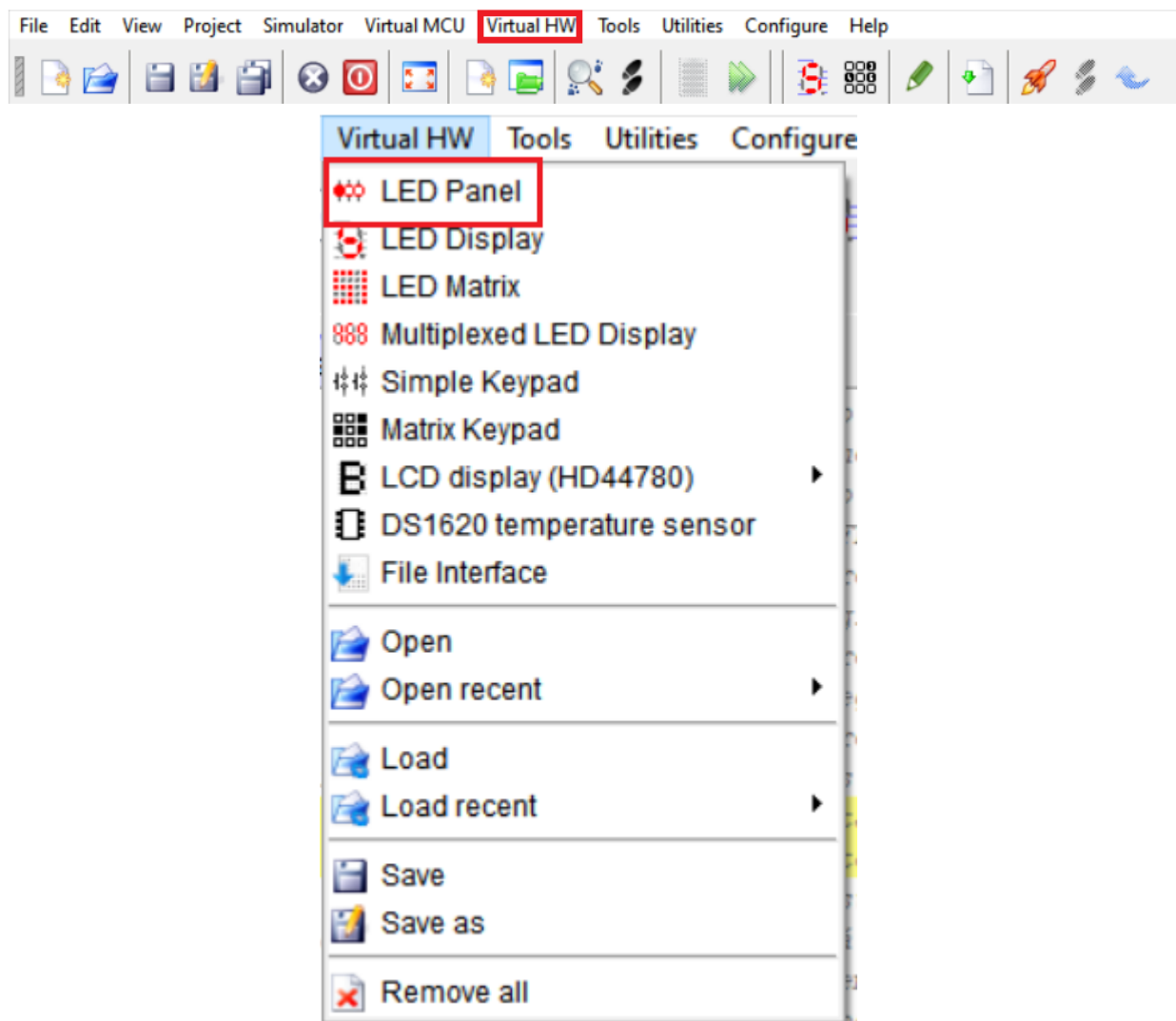
- Simulador MCU 8051 IDE.

### **3.4. PROCEDIMENTOS**

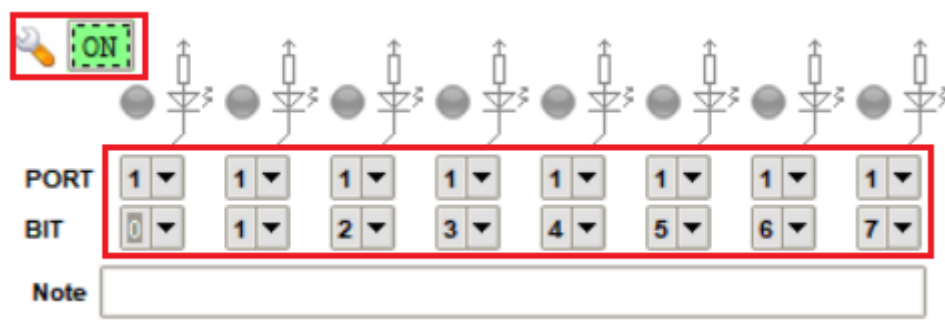
### 3.4.1. LIGAÇÕES DO PAINEL DE LED

Para iniciar o experimento é necessário fazer a ligação das portas e seus respectivos bits no painel de led 's do simulador.

Localização do painel de led's:



Para configurar o painel, basta ativá-lo e relacionar cada um dos oito led's a uma porta e a um bit do microcontrolador como mostra a imagem a seguir:



### 3.4.2. FUNCIONAMENTO DO PROGRAMA

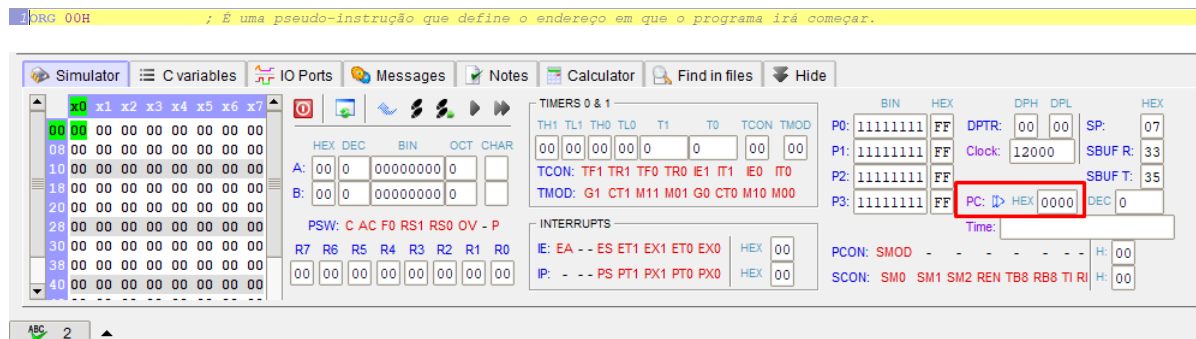
Descrição detalhada de cada linha do programa utilizado como exemplo, apresentando em cada linha seu respectivo funcionamento comentado.

```
1 ORG 00H ; Define o endereço de início em que o programa irá iniciar ou continuar.
2 LJMP INICIO ; É um label que tem como função apontar para um determinado trecho de código.
3 ORG 30H ; Define o endereço de início em que o programa irá iniciar ou continuar.
4 INICIO: MOV SP, #2FH ; Move para 2FH para o registrador SP.
5 MOV A, #01H ; Move para o registrador acumulador o valor 01H.
6 V1: MOV P1, A ; Move para o registrador P1 o conteúdo do registrador Acumulador.
7 MOV R0, #10 ; Move para o registrador R0 o valor 10.
8 V3: MOV R1, #3 ; Move para o registrador R1 o valor 3.
9 DJNZ R1, $ ; Mantém um decremento no registrador R1 até que ele chegue a zero, só assim o programa dará
; procedência para as próximas linhas de comando.
10 DJNZ R0, V3 ; Mantém, constantemente, um decremento no registrador R0 a ida até a localização do label V3 até que
; o registrador R0 fique com seu conteúdo zerado.
11 RL A ; Desloca o registrador Acumulador à esquerda, movendo todos os bits do Acumulador uma casa à esquerda
; e adicionando um zero no bit mais à direita.
12 LJMP V1 ; É um label que tem como função apontar para um determinado trecho do código.
13 END ; Define o final do programa.
```

Comentários e execução do código acima:

#### ORG 00H

**Função:** Indica que o programa vai começar no endereço 0H (PC = 0).



## LJMP INICIO

**Função:** o programa cai para a linha referente ao label INICIO.

2 LJMP INICIO ; Vai para a linha referente ao label INICIO.

The screenshot shows the Proteus simulator interface. On the left, the assembly code is displayed: `2 LJMP INICIO ; Vai para a linha referente ao label INICIO.`. The main window shows the register status. The PC register is highlighted with a red box, showing the value 0030. The PSW register shows the value C AC F0 RS1 RS0 OV - P. The TIMERS 0 & 1 section shows the TCON and TMOD registers. The INTERRUPTS section shows the IE and IP registers. The I/O Ports section shows the P0, P1, P2, and P3 registers. The SBUF register shows the value F7. The PCON register shows the value SMOD. The SCON register shows the value SM0 SM1 SM2 REN TB8 RB8 TI RI.

## ORG 30H

**Função:** É uma pseudo-instrução que define o que o programa irá continuar no endereço e memória 30H.

3 ORG 30H ; É uma pseudo-instrução que define o endereço em que o programa irá continuar.

The screenshot shows the Proteus simulator interface. On the left, the assembly code is displayed: `3 ORG 30H ; É uma pseudo-instrução que define o endereço em que o programa irá continuar.`. The main window shows the register status. The PC register is highlighted with a red box, showing the value 0030. The PSW register shows the value C AC F0 RS1 RS0 OV - P. The TIMERS 0 & 1 section shows the TCON and TMOD registers. The INTERRUPTS section shows the IE and IP registers. The I/O Ports section shows the P0, P1, P2, and P3 registers. The SBUF register shows the value F7. The PCON register shows the value SMOD. The SCON register shows the value SM0 SM1 SM2 REN TB8 RB8 TI RI.

## INICIO: MOV SP,#2FH

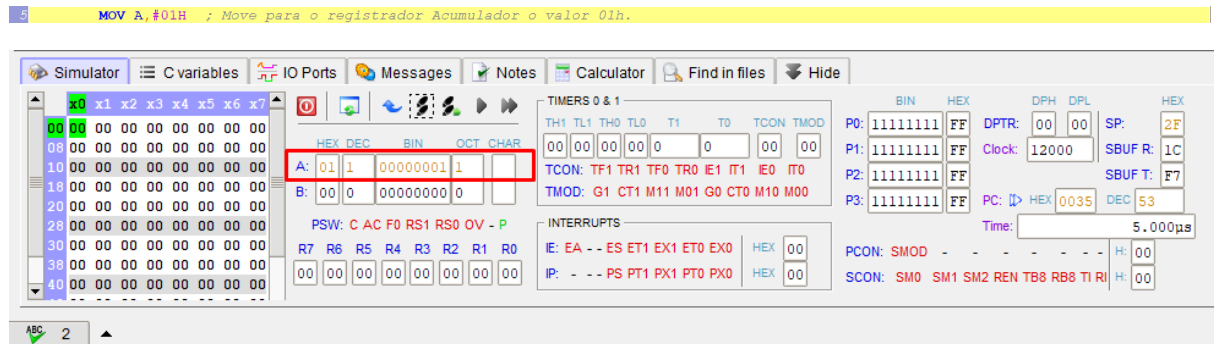
**Função:** Move para o registrador Stack Pointer (SP) o valor 2Fh.

INICIO: MOV SP,#2FH ; Move para o registrador Stack Pointer (SP) o valor 2Fh.

The screenshot shows the Proteus simulator interface. On the left, the assembly code is displayed: `INICIO: MOV SP,#2FH ; Move para o registrador Stack Pointer (SP) o valor 2Fh.`. The main window shows the register status. The SP register is highlighted with a red box, showing the value 2F. The PSW register shows the value C AC F0 RS1 RS0 OV - P. The TIMERS 0 & 1 section shows the TCON and TMOD registers. The INTERRUPTS section shows the IE and IP registers. The I/O Ports section shows the P0, P1, P2, and P3 registers. The SBUF register shows the value F7. The PCON register shows the value SMOD. The SCON register shows the value SM0 SM1 SM2 REN TB8 RB8 TI RI.

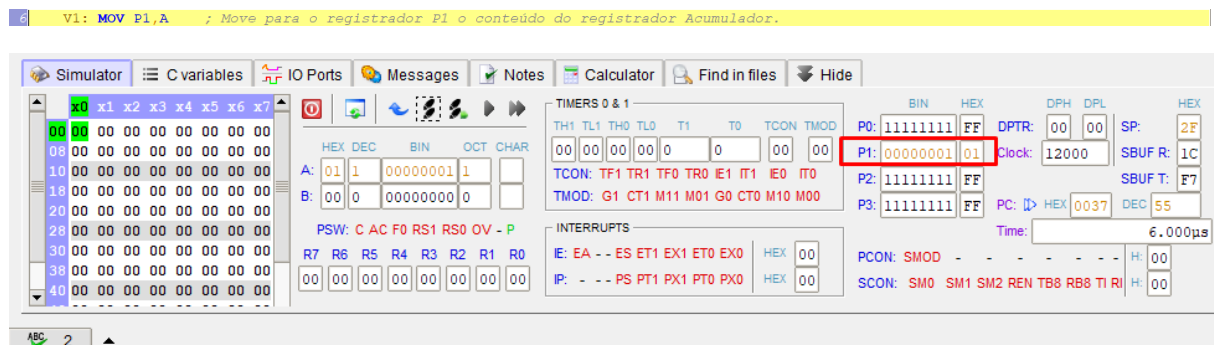
**MOV A,#01H**

**Função:** Move para o registrador Acumulador o valor 01H.

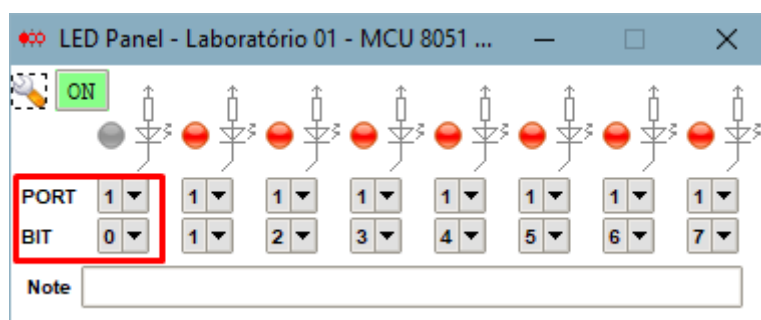


**V1: MOV P1,A**

**Função:** Move para o registrador P1 o conteúdo do registrador Acumulador. V1 é um label

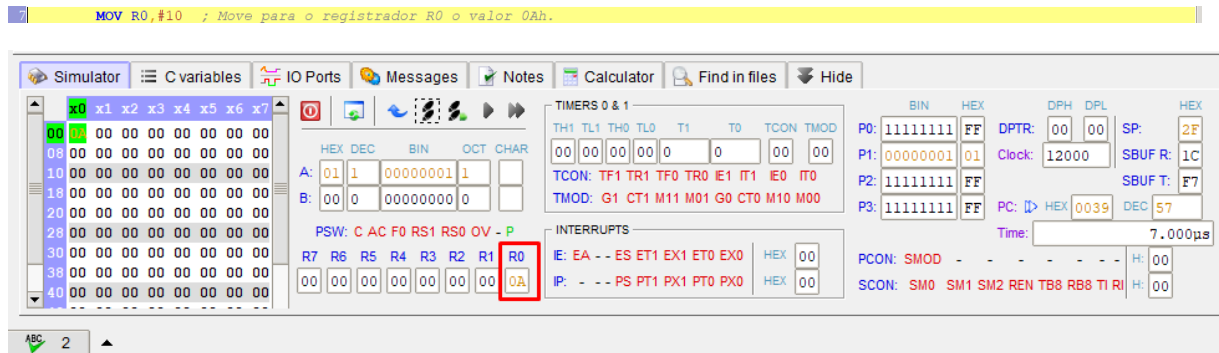


**Painel de LED 's:** Como o bit 0 do registrador P1 foi alterado para 1, o led associado desligou.

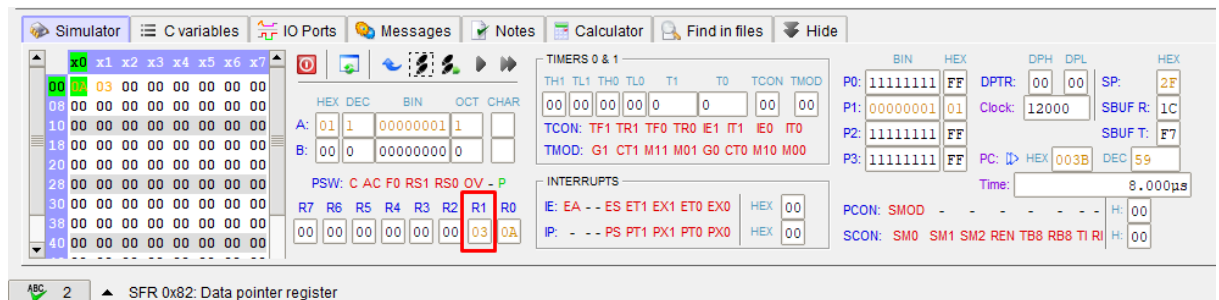


**MOV R0,#10**

**Função:** Move para o registrador R0 o valor 0Ah.

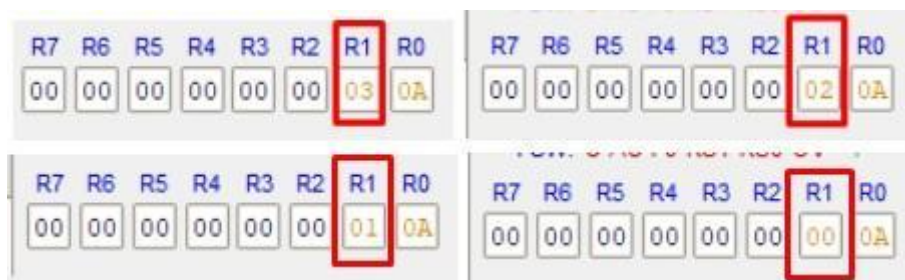
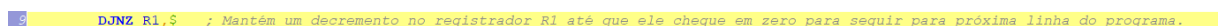
**V3: MOV R1,#3**

**Função:** Move para o registrador R1 o valor 3.



**DJNZ R1,\$**

**Função:** Mantém um decremento no registrador R1 até que ele chegue em zero para seguir para a próxima linha do programa.

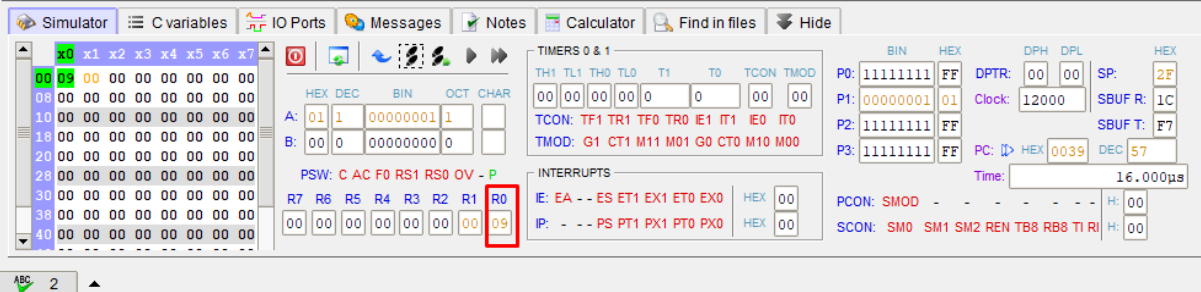




## DJNZ R0,V3

**Função:** Mantém, constantemente, um decremento no registrador R0 e ida até a localização do label V3 até que o registrador R0 fique com o seu conteúdo zerado.

10 DJNZ R0,V3 ; Mantém, constantemente, um decremento no registrador R0 e ida até a localização do label V3 até que o registrador R0 fique com o seu conteúdo zerado.

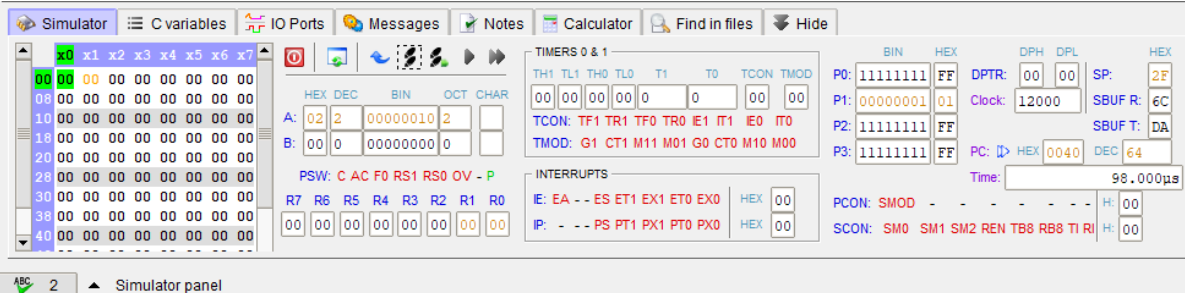


The screenshot shows the AVR simulator interface. The assembly code line 10 is `DJNZ R0,V3 ; Mantém, constantemente, um decremento no registrador R0 e ida até a localização do label V3 até que o registrador R0 fique com o seu conteúdo zerado.`. The register window shows R0 with a value of 09, highlighted with a red box. The PC register shows 0039. The status bar at the bottom indicates 'ABC 2'.

## RLA

**Função:** Desloca o registrador Acumulador à esquerda, movendo todos os bits do Acumulador uma casa à esquerda e adicionando um zero no bit mais à direita.

11 RLA ; Desloca o registrador Acumulador à esquerda, movendo todos os bits do Acumulador uma casa à esquerda e adicionando um zero no bit mais à direita.

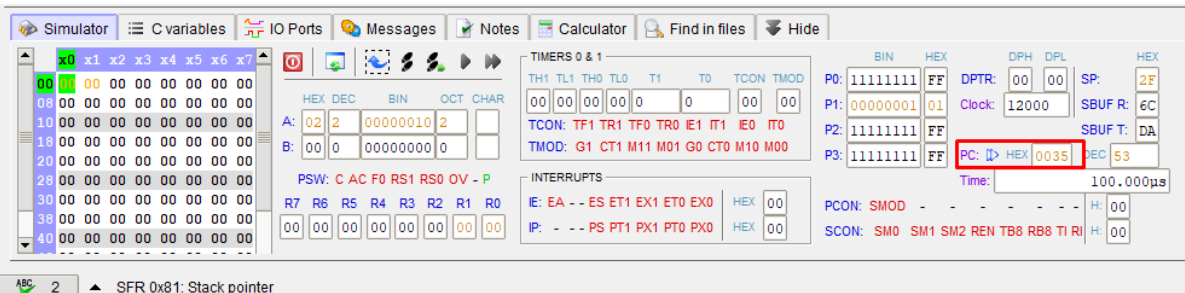


The screenshot shows the AVR simulator interface. The assembly code line 11 is `RLA ; Desloca o registrador Acumulador à esquerda, movendo todos os bits do Acumulador uma casa à esquerda e adicionando um zero no bit mais à direita.`. The register window shows R0 (the Accumulator) with a value of 02, highlighted with a red box. The PC register shows 0040. The status bar at the bottom indicates 'ABC 2' and 'Simulator panel'.

## LJMP V1

**Função:** Vai para a linha referente ao label V1.

12 LJMP V1 ; Vai para a linha referente ao label V1.



The screenshot shows the AVR simulator interface. The assembly code line 12 is `LJMP V1 ; Vai para a linha referente ao label V1.`. The register window shows the PC register with a value of 0035, highlighted with a red box. The status bar at the bottom indicates 'ABC 2' and 'SFR 0x81: Stack pointer'.

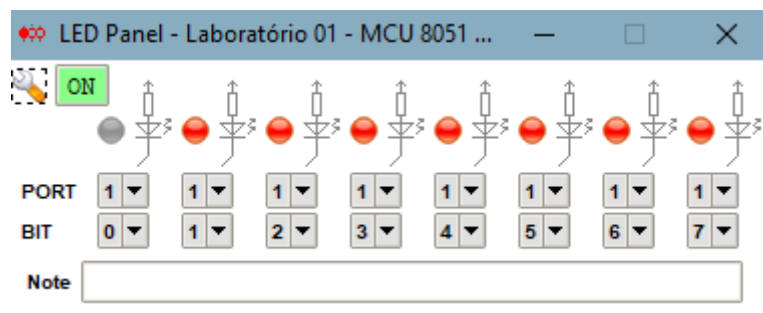
**END**

**Função:** Indica o final do programa.

```
15 END ; Final do programa.
```

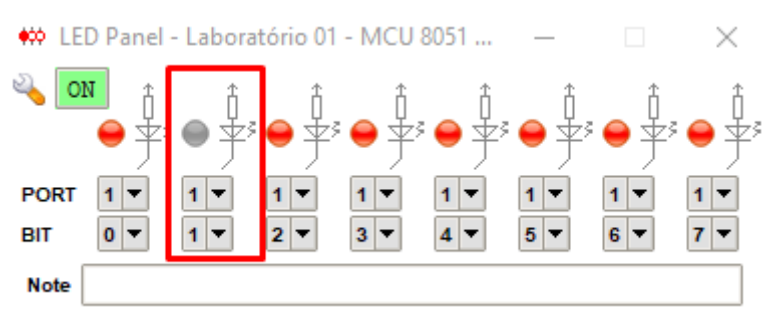
### 3.4.3. FUNCIONAMENTO DO PAINEL DE LED'S

Inicialmente, o painel tem a primeira alteração quando o programa chega na linha 6 do programa, ficando da seguinte forma:



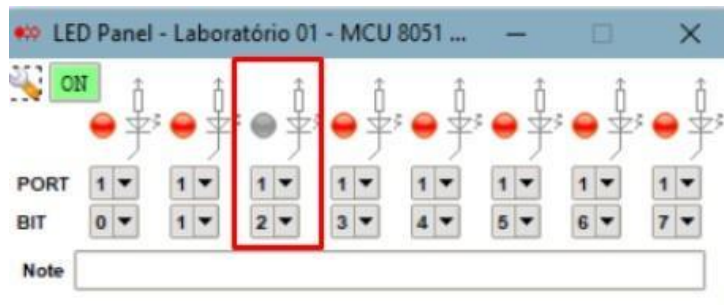
(1º Execução)

Após várias iterações, quando o programa atinge a linha 12, que executa uma instrução relacionada ao retorno à linha 6, que, ao ser executada, altera sequencialmente o conteúdo do registrador P1, que permanece com o mesmo conteúdo do registrador acumulador (que a princípio era equivalente a 01H, porém, devido ao comando "RL A" na linha 11, foi alterado para 02H) há uma alteração na barra de LEDs pois o conteúdo do registrador P1 foi alterado para 00000010 conforme mostrado abaixo:

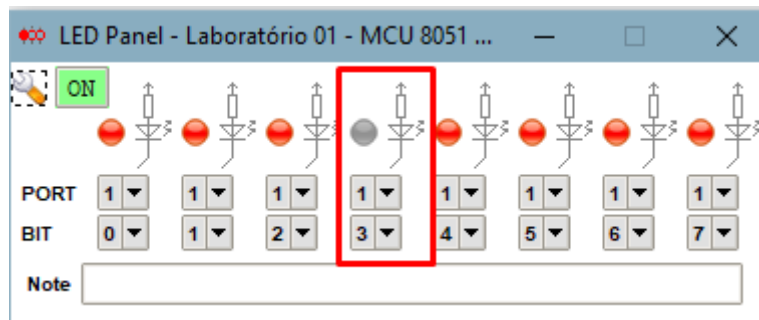


(2º Execução)

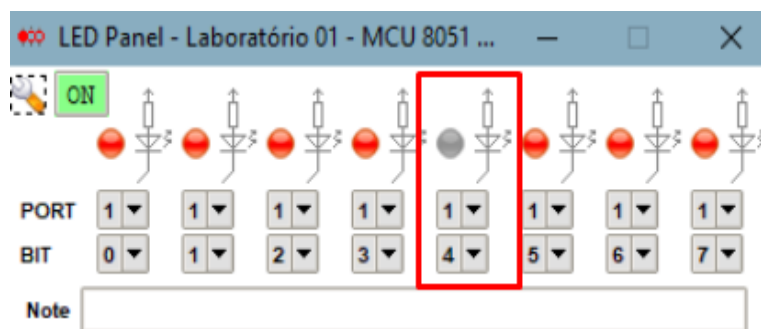
Nota-se que como consequência da listagem do programa em Assembly, sempre haverá essa mudança padronizada no painel de LED 's, em que o led que antes estava desligado ascende e o led referente ao bit seguinte (led à direita) irá desligar, como é mostrado a seguir:



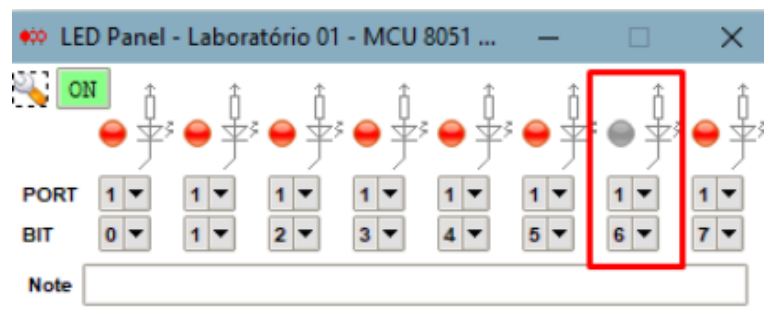
(3º Execução)



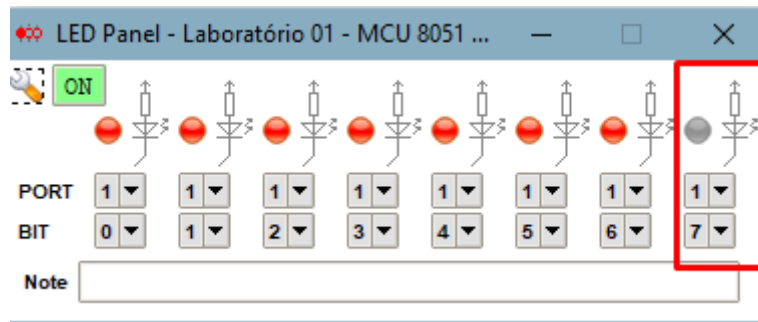
(4º Execução)



(5º Execução)

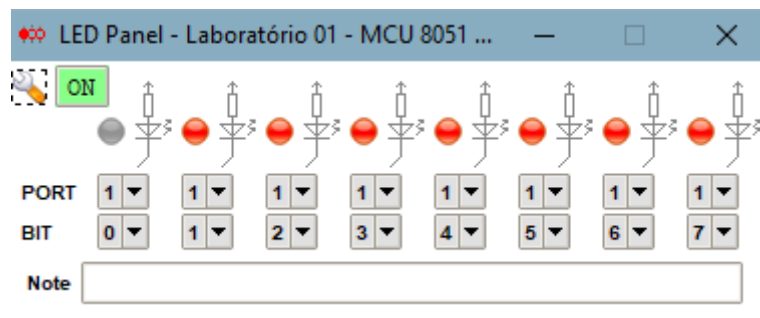


(6º Execução)



(7º Execução)

Após esta última execução, o painel voltará a ter o LED referente ao bit 0 apagado novamente, já que o conteúdo do registrador P1 voltou a ser 00000001 e o ciclo será reiniciado novamente, como mostra a seguir:



(8º Execução)

### 3.4.4. CONTEÚDO DO ARQUIVO “.lst”

Conteúdo do arquivo “.lst” criado após salvar o programa em assembly, contendo diversas informações relativas à compilação do programa em Assembly, como é mostrado a seguir:

```

Laboratório 2 - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
Laboratório 2
0000 020030      1      ORG 00H          ; É uma pseudo-instrução que define o endereço em que o programa irá começar.
                                2      LMP INICIO      ; Vai para a linha referente ao label INICIO.
                                3      ORG 30H          ; É uma pseudo-instrução que define o endereço em que o programa irá continuar.
0030 75812F      4      INICIO: MOV SP,#2FH      ; Move para o registrador Stack Pointer (SP) o valor 2FH.
0033 7481        5      MOV A,R0H          ; Move para o registrador Stack Pointer (SP) o valor 2FH.
0035 F590        6      VI: MOV P1,A          ; Move para o registrador P1 o conteúdo do registrador Acumulador.
0037 78A0        7      MOV R0,#0H        ; Move para o registrador R0 o valor 00h.
0039 7903        8      V3: MOV R1,#3      ; Move para o registrador R1 o valor 3.
003B D9FE        9      DJNZ R1,$          ; Mantém um decremento no registrador R1 até que ele chegue em zero para seguir p
003D 08FA       10      DJNZ R0,V3          ; Mantém, constantemente, um decremento no registrador R0 e irá até à localizacão
003F 23         11      RL A              ; Desloca o registrador Acumulador à esquerda, movendo todos os bits do Acumulado
0040 020035     12      LMP V1              ; Vai para a linha referente ao label V1.
                                13      END          ; Final do programa.
ASSEMBLY COMPLETE, NO ERRORS FOUND, NO WARNINGS

SYMBOL TABLE:
??MCU_8051_IDE . . . . . N NUMB 0051H NOT USED
??VERSION . . . . . N NUMB 014AH NOT USED
AC . . . . . B ADDR 0006H NOT USED
ACC . . . . . D ADDR 000EH NOT USED
ACSR . . . . . D ADDR 0097H NOT USED
ADCF . . . . . D ADDR 00F6H NOT USED
ADCLK . . . . . D ADDR 00F2H NOT USED
ADCON . . . . . D ADDR 00F3H NOT USED
ADCON1 . . . . . D ADDR 00F5H NOT USED
ADOL . . . . . D ADDR 00F4H NOT USED
AUXR . . . . . D ADDR 00EEH NOT USED
AUXR1 . . . . . D ADDR 00A2H NOT USED
B . . . . . D ADDR 00F8H NOT USED
BDONCON . . . . . D ADDR 009BH NOT USED
BDONCON1 . . . . . D ADDR 009CH NOT USED
BRL . . . . . D ADDR 009AH NOT USED
CCAP0H . . . . . D ADDR 00FAH NOT USED
CCAP0L . . . . . D ADDR 00E9H NOT USED
CCAP1H . . . . . D ADDR 00FBH NOT USED
CCAP1L . . . . . D ADDR 00EBH NOT USED
CCAP2H . . . . . D ADDR 00FCH NOT USED
CCAP3H . . . . . D ADDR 00FDH NOT USED
CCAP4H . . . . . D ADDR 00FEH NOT USED
CCAP12H . . . . . D ADDR 00FCH NOT USED
CCAP12L . . . . . D ADDR 00ECH NOT USED
CCAP13H . . . . . D ADDR 00FCH NOT USED
CCAP13L . . . . . D ADDR 00EDH NOT USED
CCAP14H . . . . . D ADDR 00FEH NOT USED
CCAP14L . . . . . D ADDR 00EEH NOT USED
CCAP0H . . . . . D ADDR 00FAH NOT USED
CCAP0L . . . . . D ADDR 00E9H NOT USED
CCAP1H . . . . . D ADDR 00FBH NOT USED
CCAP1L . . . . . D ADDR 00EBH NOT USED
CCAP2H . . . . . D ADDR 00FCH NOT USED
CCAP3H . . . . . D ADDR 00FDH NOT USED
CCAP4H . . . . . D ADDR 00FEH NOT USED
CCAP12H . . . . . D ADDR 00FCH NOT USED
CCAP12L . . . . . D ADDR 00ECH NOT USED
CCAP13H . . . . . D ADDR 00FCH NOT USED
CCAP13L . . . . . D ADDR 00EDH NOT USED
CCAP14H . . . . . D ADDR 00FEH NOT USED
CCAP14L . . . . . D ADDR 00EEH NOT USED
CCAP0H . . . . . D ADDR 00FAH NOT USED
CCAP0L . . . . . D ADDR 00E9H NOT USED
CCAP1H . . . . . D ADDR 00FBH NOT USED
CCAP1L . . . . . D ADDR 00EBH NOT USED
CCAP2H . . . . . D ADDR 00FCH NOT USED
CCAP3H . . . . . D ADDR 00FDH NOT USED
CCAP4H . . . . . D ADDR 00FEH NOT USED
CCAP12H . . . . . D ADDR 00FCH NOT USED
CCAP12L . . . . . D ADDR 00ECH NOT USED
CCAP13H . . . . . D ADDR 00FCH NOT USED
CCAP13L . . . . . D ADDR 00EDH NOT USED
CCAP14H . . . . . D ADDR 00FEH NOT USED
CCAP14L . . . . . D ADDR 00EEH NOT USED

```

## Laboratorio 2 - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

CCF0	B	ADDR	0008H	NOT USED
CCF1	B	ADDR	0009H	NOT USED
CCF2	B	ADDR	000AH	NOT USED
CCF3	B	ADDR	000BH	NOT USED
CCF4	B	ADDR	000CH	NOT USED
CCON	D	ADDR	0008H	NOT USED
CFINT	C	ADDR	0033H	NOT USED
CH	D	ADDR	00F9H	NOT USED
CKCON	D	ADDR	008FH	NOT USED
CKCON0	D	ADDR	008FH	NOT USED
CKRL	D	ADDR	0097H	NOT USED
CKSEL	D	ADDR	0085H	NOT USED
CL	D	ADDR	00E9H	NOT USED
CLKREG	D	ADDR	008FH	NOT USED
CMD0	D	ADDR	00D9H	NOT USED
CPRL2	B	ADDR	00C8H	NOT USED
CR	B	ADDR	00DEH	NOT USED
CT2	B	ADDR	00C9H	NOT USED
CY	B	ADDR	00D7H	NOT USED
DP0H	D	ADDR	0083H	NOT USED
DP0L	D	ADDR	0082H	NOT USED
DP1H	D	ADDR	0085H	NOT USED
DP1L	D	ADDR	0084H	NOT USED
DPH	D	ADDR	0083H	NOT USED
DPL	D	ADDR	0082H	NOT USED
EA	B	ADDR	00AFH	NOT USED
EC	B	ADDR	00AEH	NOT USED
EECON	D	ADDR	0096H	NOT USED
ES	B	ADDR	00ACH	NOT USED
ET0	B	ADDR	00A9H	NOT USED
ET1	B	ADDR	00ABH	NOT USED
ET2	B	ADDR	00ADH	NOT USED
EX0	B	ADDR	00A8H	NOT USED
EX1	B	ADDR	00AAH	NOT USED
EXEN2	B	ADDR	00CBH	NOT USED
EXF2	B	ADDR	00CEH	NOT USED
EXTI0	C	ADDR	0003H	NOT USED
EXTI1	C	ADDR	0013H	NOT USED
F0	B	ADDR	00D3H	NOT USED
FE	B	ADDR	009FH	NOT USED
IE	D	ADDR	00A8H	NOT USED
IE0	B	ADDR	0089H	NOT USED
IE1	B	ADDR	0088H	NOT USED
INICIO	C	ADDR	0030H	
INT0	B	ADDR	0082H	NOT USED
INT1	B	ADDR	0083H	NOT USED
IP	D	ADDR	0088H	NOT USED
IPH	D	ADDR	0087H	NOT USED
IPH0	D	ADDR	0087H	NOT USED
IPH1	D	ADDR	0083H	NOT USED
IPL0	D	ADDR	0088H	NOT USED
IPL1	D	ADDR	0082H	NOT USED
IT0	B	ADDR	0088H	NOT USED

&lt;

## Laboratorio 2 - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

RS0	B	ADDR	00D3H	NOT USED
RS1	B	ADDR	00D4H	NOT USED
RXD	B	ADDR	00B0H	NOT USED
SADDR	D	ADDR	00A9H	NOT USED
SADDR_0	D	ADDR	00A9H	NOT USED
SADDR_1	D	ADDR	00AAH	NOT USED
SADEN	D	ADDR	00B9H	NOT USED
SADEN_0	D	ADDR	00B9H	NOT USED
SADEN_1	D	ADDR	00BAH	NOT USED
SBUF	D	ADDR	0099H	NOT USED
SCON	D	ADDR	0098H	NOT USED
SINT	C	ADDR	0023H	NOT USED
SM0	B	ADDR	009FH	NOT USED
SM1	B	ADDR	009EH	NOT USED
SM2	B	ADDR	009DH	NOT USED
SP	D	ADDR	0081H	
SPCON	D	ADDR	00C3H	NOT USED
SPCR	D	ADDR	00D5H	NOT USED
SPDAT	D	ADDR	00C5H	NOT USED
SPDR	D	ADDR	00B6H	NOT USED
SPSR	D	ADDR	00AAH	NOT USED
SPSTA	D	ADDR	00C4H	NOT USED
T0	B	ADDR	00B4H	NOT USED
T1	B	ADDR	00B5H	NOT USED
TZCON	D	ADDR	00C8H	NOT USED
TZMOD	D	ADDR	00C9H	NOT USED
TB8	B	ADDR	009BH	NOT USED
TCLK	B	ADDR	00CCH	NOT USED
TCON	D	ADDR	008BH	NOT USED
TF0	B	ADDR	00BDH	NOT USED
TF1	B	ADDR	00BFH	NOT USED
TF2	B	ADDR	00CFH	NOT USED
TH0	D	ADDR	00BCH	NOT USED
TH1	D	ADDR	00BDH	NOT USED
TH2	D	ADDR	00CDH	NOT USED
TI	B	ADDR	0099H	NOT USED
TIMER0	C	ADDR	000BH	NOT USED
TIMER1	C	ADDR	001BH	NOT USED
TIMER2	C	ADDR	002BH	NOT USED
TL0	D	ADDR	00BAH	NOT USED
TL1	D	ADDR	00BBH	NOT USED
TL2	D	ADDR	00CCH	NOT USED
TMOD	D	ADDR	0089H	NOT USED
TR0	B	ADDR	00BCH	NOT USED
TR1	B	ADDR	00BEH	NOT USED
TR2	B	ADDR	00CAH	NOT USED
TXD	B	ADDR	00B1H	NOT USED
V1	C	ADDR	0035H	
V3	C	ADDR	0039H	
WDTCON	D	ADDR	00A7H	NOT USED
WDTPRG	D	ADDR	00A7H	NOT USED
WDTTRST	D	ADDR	00A6H	NOT USED
WR	B	ADDR	00B6H	NOT USED

&lt;

Observando seu conteúdo, você pode ver que está em sua estrutura, nomes de símbolos, tipos correspondentes e todos os outros atributos, como endereços de memória e seus valores associados.

Os caracteres nas duas primeiras letras representam o endereço da memória e seu conteúdo.

### Caracteres presentes na segunda coluna:

- B:** Refere-se à memória “bit”.
- I:** Refere-se à memória “idata”.
- C:** Refere-se à memória “code”.
- D:** Refere-se à memória de “dados”.
- X:** Refere-se à memória “xdata”.

### Abreviações presentes na terceira coluna:

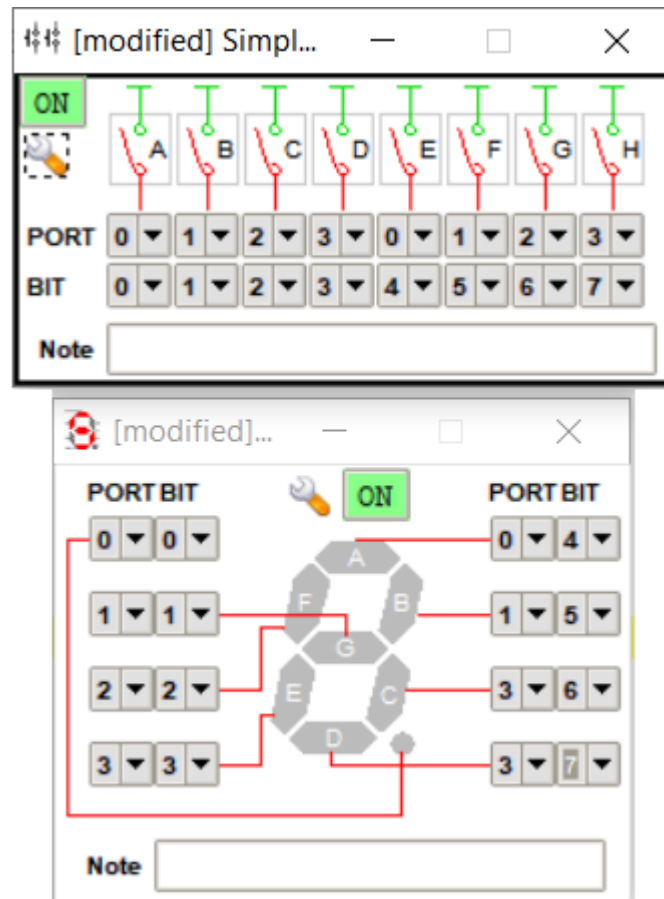
- ADDR:** Se refere a um endereço de memória.
- NUMB:** Se refere a um número.
- SEG:** Se refere a um segmento.

### Em Relação ao item B:

```
1      ORG      00H ; Define o endereço de início em que o programa irá iniciar ou continuar.
2      SJMP     INICIO ; É um label que tem como função apontar para um determinado trecho de código.
3
4      ORG      30H ; Define o endereço de início em que o programa irá iniciar ou continuar.
5 INICIO: MOV     R0, #30H ; Move para 2FH para o registrador SP.
6      MOV     R3, #00 ; Move para o registrador acumulador um determinado valor.
7 SALT01: MOV     A, @R0 ; Move para o registrador acumulador o valor de R0.
8      MOV     B, A ; Move para o registrador acumulador o valor de A.
9      MOV     A, R3 ; Move o registro 3 para o acumulador.
10     CLR     C ; Zera o acumulador
11     SUBB    A, B ; Subtrai o conteúdo do Acumulador B eo Carry do Acumulador.
12     JNC     SALT02 ; Salta se o carry for 0. o jump é relativo.
13     MOV     R3, B ; Move para R3 o conteúdo do acumulador.
14 SALT02: INC     R0 ; Incrementa o Registro 0.
15     MOV     B, R0 ; Move para o acumulador o Registro 0.
16     MOV     A, #40 ; Move para A o valor de 40.
17     CLR     C ; Zera o acumulador.
18     SUBB    A, B ; Subtrai o conteúdo do Acumulador B eo Carry do Acumulador.
19     JNC     SALT01
20     SJMP    $
21     END     ; Define o final do programa.
```

Na segunda parte, o programa inicia realizando um salto curto (SJMP) para START, então após mover os dados #30h e #00 para os registradores correspondentes R0 e R3, inicia-se o ciclo entre JUMP1 e JUMP2 utilizando JNC. O registrador 0 (R0) é incrementado em 1 bit a cada ciclo completo.

### Em relação ao item C:



Por fim, na terceira parte, foi feita uma configuração simples utilizando um simulador de display de LED de teclado simples, que, em resumo, ao pressionar uma tecla em um teclado simples, o LED referente ao módulo de LED virtual.

#### 4. RESULTADOS E DISCUSSÃO

Após o teste, conseguimos entender as funcionalidades da linguagem da plataforma "Assembly". Após executar o código acima, conseguimos entender a relação entre as funções, função dos painéis de LED e registradores.

#### 5. CONCLUSÕES

Pode-se concluir que a linguagem assembly executada no microcontrolador MCU 8051 pode fornecer uma grande biblioteca de opções e ações em frente aos seus registradores e portas, conforme demonstrado durante a sessão no exemplo do painel de LEDs.

#### 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: 10 jul. 2022.

MORAIS, Misael. Organização e arquitetura de computadores. [S. l.], . 2021. Disponível em:<https://drive.google.com/drive/folders/0Bwjlecok7TpyfnAtOmx6bE4yZ43amNsbnRxMkF4UFlpWVZhWmRfeVBSeHRRVi1xRzNOZnM?resourcekey=0-WmQ6S1i6hLYyCoGBLbMeGw>. Acesso em: 10 ago. 2022