

"GENIUS" EM ASSEMBLY



< INTEGRANTES >



Name DB Bruna Pereira Paz

RA DB 22121020-6

Name DB Lorena Cardoso Sanches

RA DB 22121060-2

01001100010111101001100010111101001100
00111010111001000111010111001000111010
11010101001010011010101001010011010101

< DESCRIÇÃO >



1.DESCRICÃO DO PROJETO

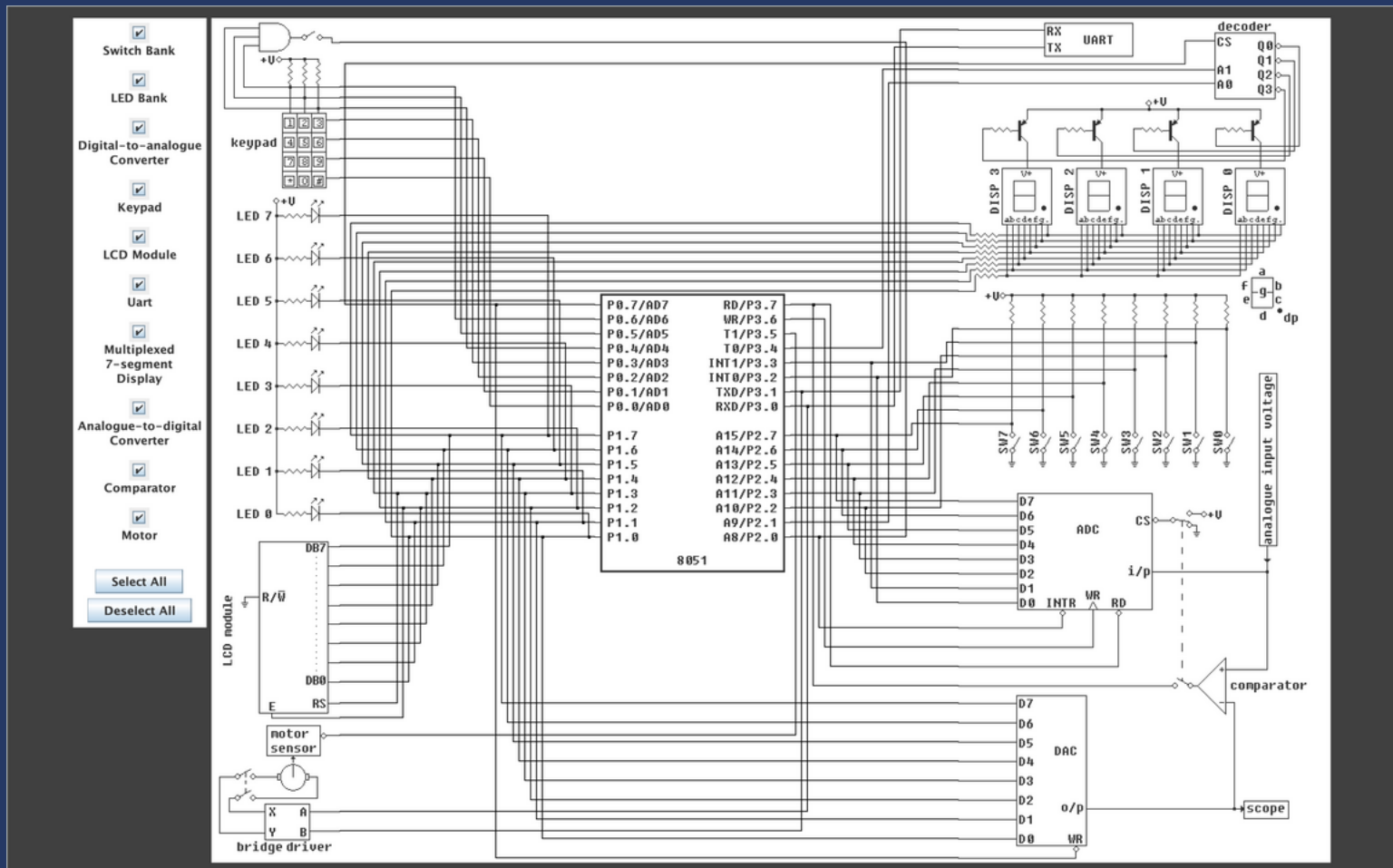
Para o Projeto de Arquitetura de Computadores, decidimos realizar uma versão do brinquedo infantil "Genius" no simulador EdSim51.

Desta maneira temos o Jogo Genius, onde é necessário decorar a sequência de cores apresentada (gerada aleatoriamente), tendo como objetivo final conseguir a maior série de vitórias.



< DESENHO ESQUEMATICO >

2. DESENHOS ESQUEMÁTICOS



<FLUXOGRAMA OU DIAGRAMA>

3. FLUXOGRAMA OU DIAGRAMA

Salvando na memória os Leds que em ordem aleatória aparecem e os Numeros clicados correspondentes a ordem dos Leds inseridas pelo usuário.

The screenshot displays the Proteus ISIS51 simulator interface. At the top, the 'System Clock (MHz)' is set to 12.0, and a dropdown menu shows '8' with an 'Update Freq.' button. Below this, the 'SBUF' register is shown with 'R/O' and 'W/O' both set to 0x00. The 'RXD' and 'TXD' pins are both set to 1, and the 'SCON' register is 0x00. The 'TH0' and 'TL0' registers are 0x00 and 0x4D respectively. The 'TMOD' register is 0x02, and the 'TCON' register is 0x37. The 'pins' and 'bits' section shows P3, P2, P1, and P0 with values 0xFF, 0xFF, 0x1B, and 0xFD respectively. The 'TH1' and 'TL1' registers are 0x00 and 0x00. The 'PC' register is 0x0278. The 'R7' through 'R0' registers are 0x00, 0x3C, 0x00, 0x00, 0x15, 0x05, 0x54, and 0x64. The 'B' register is 0x00, 'ACC' is 0x21, 'PSW' is 0x80, 'IP' is 0x00, 'IE' is 0x07, 'PCON' is 0x00, 'DPH' is 0x00, 'DPL' is 0x00, and 'SP' is 0x0B. The 'PSW' register is highlighted with a blue box and the number '8051' next to it. The 'Data Memory' section shows a table of memory addresses and values. The 'Modify RAM' section shows 'addr' as 0x00 and 'value' as 0x00. The 'Remove All Breakpoints' button is at the bottom right.

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	64	54	05	15	00	00	3C	00	9E	00	10	02	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	EF	BF	FB	EF	FB	00	00	00	00	00	00	00	00	00	00	00
60	F7	EF	FB	DF	EF	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

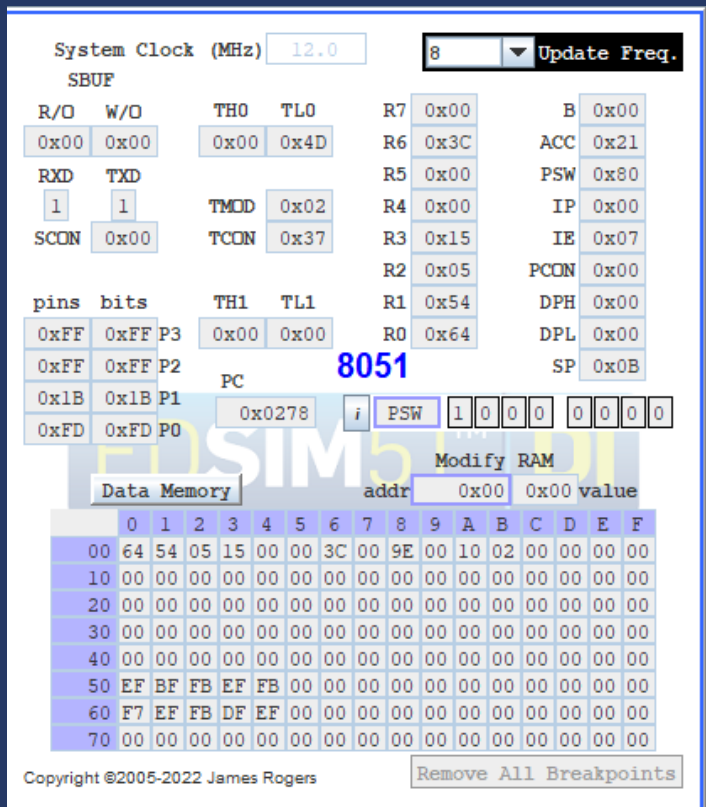
Ordem dos LEDs gerada aleatoriamente

Ordem dos LEDs inserida pelo usuário

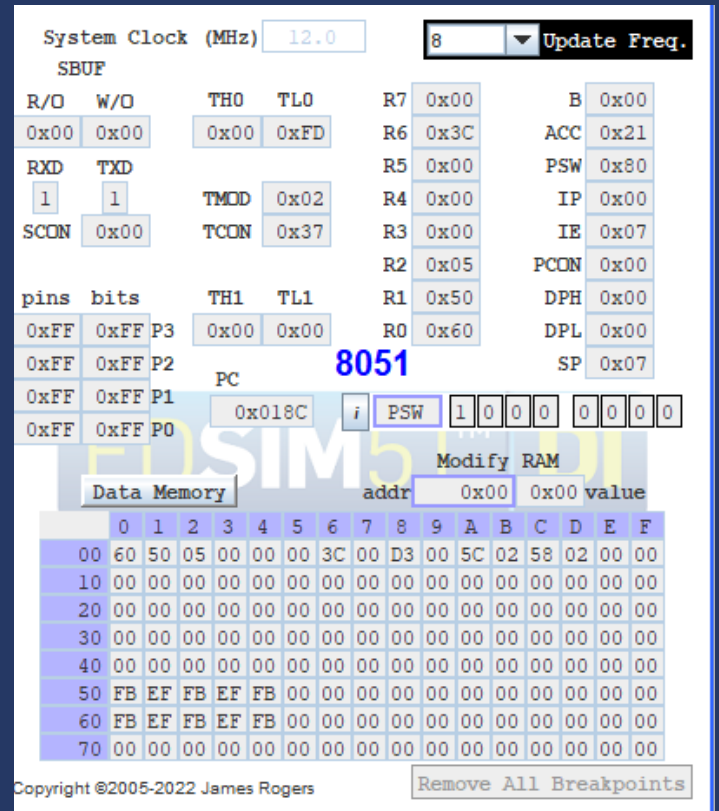
<Imagens da simulação realizada na IDE >

4. IMAGENS DA SIMULAÇÃO

Exemplo de telas de acerto e erro de sequencia das cores do jogo Genius



Quando Erra! -
Erra a sequencia de
Genius



Quando Ganha! -
Acerta a sequencia de
Genius

<Discussões e conclusões >

O projeto foi muito desafiador, uma vez que a linguagem requer muito conhecimento de inserção de memória e é uma linguagem de baixo nível, no início do projeto houve uma necessidade de trocar o tema do projeto pela dificuldade encontrada e pelo débito técnico das integrantes. O jogo Genius foi desenvolvido e a maior dificuldade encontrada foi na criação da sequencia de Leds aleatórios com a comparação das entradas do usuário com os Leds. Para este desenvolvimento foi necessário conhecimento sólido da linguagem de maquina e suas instruções.

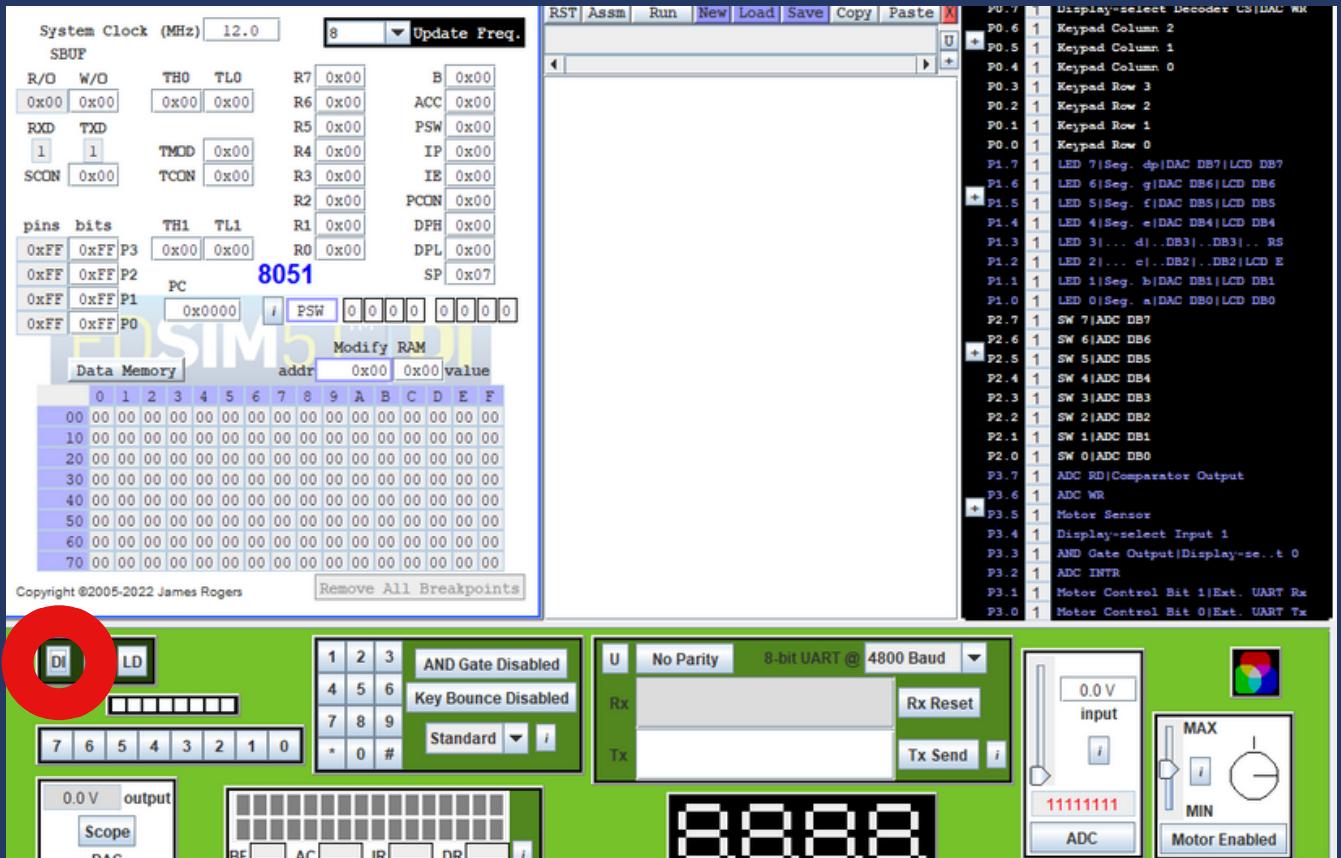
Obrigada!



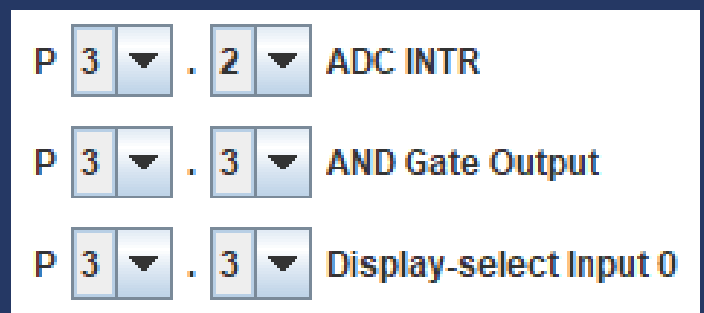
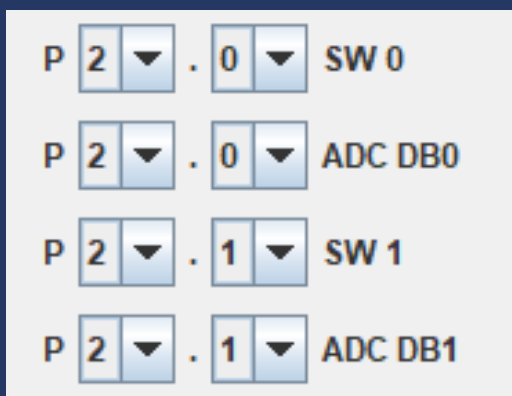
◀ Instruções para jogar ▶

Alterar portas no DI

Entrar no Dynamic Interface



Portas Padrão que devem ser alteradas



< Instruções para jogar >

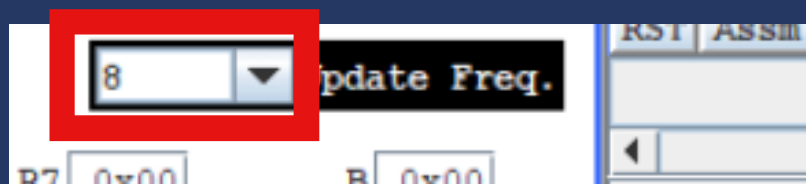
Alterar portas no DI

Portas devem ficar assim:

P	3	▼	.	2	▼	SW 0
P	3	▼	.	2	▼	ADC DB0
P	3	▼	.	3	▼	SW 1
P	3	▼	.	3	▼	ADC DB1

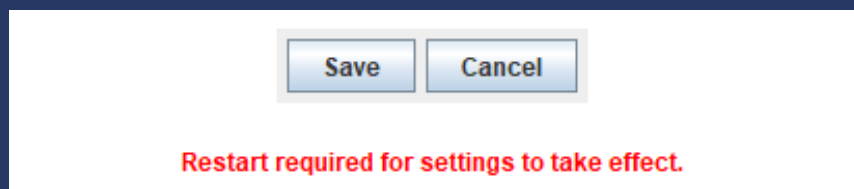
P	2	▼	.	0	▼	ADC INTR
P	2	▼	.	0	▼	AND Gate Output
P	2	▼	.	1	▼	Display-select Input 0

É necessário atualizar o valor da Frequência para 8.



8 ▼ Update Freq.

Apertar o "Save"
E reiniciar o EdSim51



Save Cancel

Restart required for settings to take effect.

<CODIGO FONTE>

1

ORG 0000h

;Arrumar - tela LCD - Loren

RS equ P1.3

EN equ P1.2

LJMP CONFIG

; Inicia Jogo

INT_EXT0:

AJMP INICIAR_GAME

RETI

ORG 000Bh

INT_TEMPO:

MOV TH0, #0

MOV TL0, #0

RETI

ORG 0080h

RETORNA_MSG_ERROU:

acall lcd_inicio

MOV A, #0

ACALL posicionaCursor

MOV A, #'E'

ACALL sendCharacter

MOV A, #'R'

ACALL sendCharacter

MOV A, #'R'

ACALL sendCharacter

MOV A, #'O'

ACALL sendCharacter

MOV A, #'U'

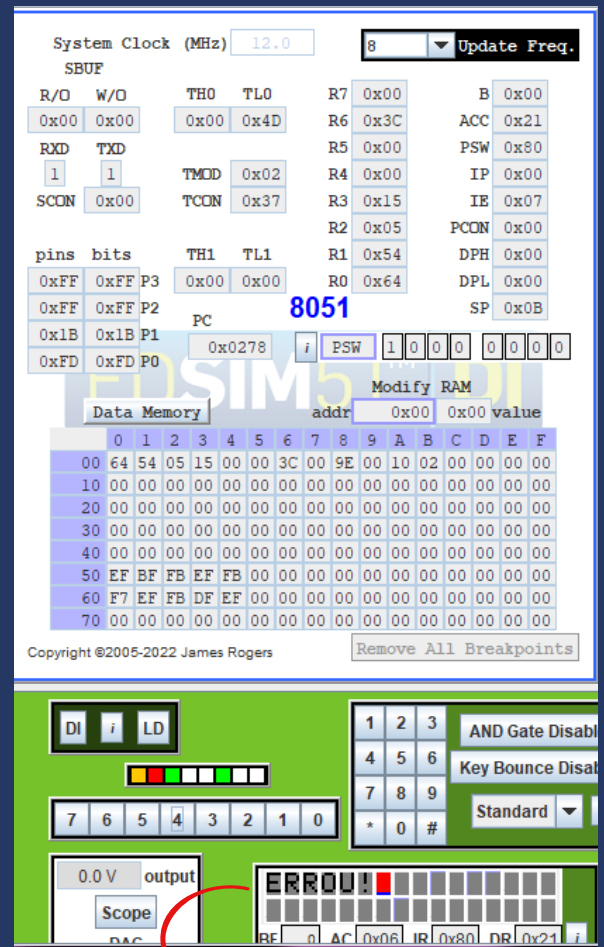
ACALL sendCharacter

MOV A, #'I'

ACALL sendCharacter

ACALL retornaCursor

MOV P1, #11111111b



Mensagem quando
perde!

2

;reinicia jogo

CPL P0.1

MOV R1, #80

MOV R0, #96

JNB P3.2, \$

LJMP INICIAR_GAME

RETORNA_MSG_GANHOU:

acall lcd_inicio

MOV A, #0

ACALL posicionaCursor

MOV A, #'G'

ACALL sendCharacter

MOV A, #'A'

ACALL sendCharacter

MOV A, #'N'

ACALL sendCharacter

MOV A, #'H'

ACALL sendCharacter

MOV A, #'O'

ACALL sendCharacter

MOV A, #'U'

ACALL sendCharacter

ACALL retornaCursor

MOV P1, #11111111b

;Reinicia novamente

CPL P0.1

MOV R1, #80

MOV R0, #96

JNB P3.2, \$

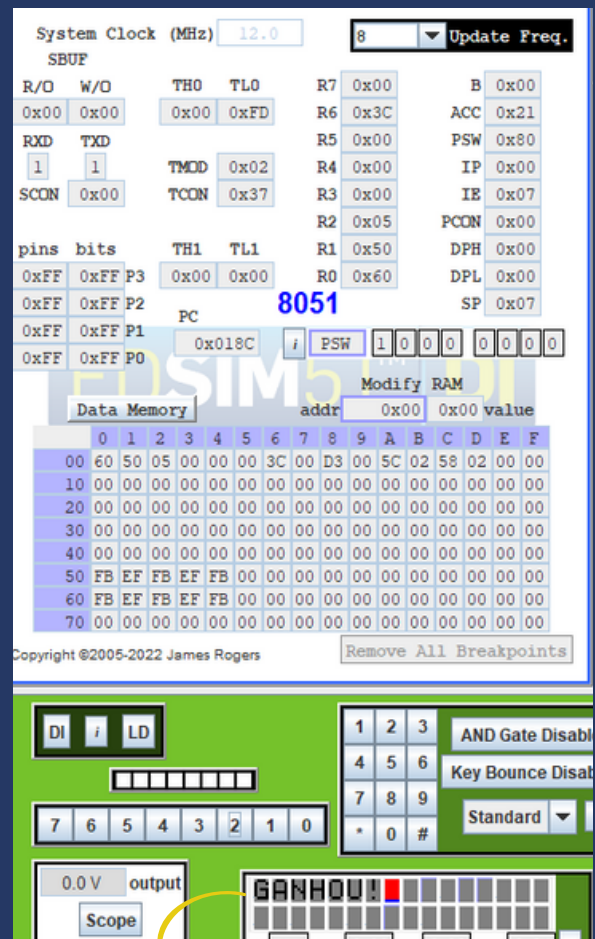
LJMP ENTRADA_PRA_INICIAR

DELAY_ARMAZENAMENTO:

DJNZ R6, DELAY_ARMAZENAMENTO

MOV R6, #60

RET



Mensagem quando
ganha!

ROTATE:

```
RR A
DJNZ B, ROTATE
MOV P1, A
RET
```

SALVA_SEQ:

```
MOV @R1, P1
INC R1
RET
```

SALVA_USR:

```
MOV @R0, P2
INC R0
CALL DELAY_ARMAZENAMENTO ;Torna led mais lento - delay
MOV P1, #11111111b ;Desliga led
LJMP INICIAR_GAME ;inicia o jogo
```

GERA_SEED:

```
MOV A, TL0
MOV B, #17
MUL AB
RLC A
ADD A, B
MOV TL0, A
RET
```

;Gera numero aleatorio

RANDOM:

```
CALL GERA_SEED
MOV P1, #11111111b
MOV A, TL0
MOV B, #6h
DIV AB
MOV A, #01111111b
MOV R2, B
CJNE R2, #0h, ROTATE
LJMP INICIAR_GAME
```

;Salva o led aceso no P1

SALVA_RANDOM:

```
CJNE R1, #84, SALVA_SEQ  
MOV @R1, P1  
CPL P0.0  
RET
```

;Aqui armazena os numeros digitados

ARMAZENA_ENTRADA:

```
MOV P1, P2 ;Mostra qual o botão o usuário apertou  
CJNE R0, #100, SALVA_USR  
MOV @R0, P2  
CPL P0.0  
CPL P0.1;Flag  
LJMP INICIAR_GAME
```

; Aqui Armazena qual entrada o usuario colocou - Lorena

LOOP_INSERT:

```
MOV P1, #01111111b  
MOV P1, #11111111b  
JNB P2.7, ARMAZENA_ENTRADA  
JNB P2.6, ARMAZENA_ENTRADA  
JNB P2.5, ARMAZENA_ENTRADA  
JNB P2.4, ARMAZENA_ENTRADA  
JNB P2.3, ARMAZENA_ENTRADA  
JNB P2.2, ARMAZENA_ENTRADA  
SJMP LOOP_INSERT
```

;aqui compara os dados armazenados

COMPARA_JOGO:

```
MOV P1, #11111111b  
MOV A, 80  
CJNE A, 96, ERRO  
MOV A, 81  
CJNE A, 97, ERRO  
MOV A, 82  
CJNE A, 98, ERRO  
MOV A, 83  
CJNE A, 99, ERRO
```

```
MOV A, 84
CJNE A, 100, ERRO
LJMP RETORNA_MSG_GANHOU
```

ERRO:

```
LJMP RETORNA_MSG_ERROU
```

CONFIG:

```
MOV R6, #60
MOV R1, #80
MOV R0, #96
SETB ITO
SETB EX0
SETB IT1
SETB EX1
MOV TMOD, #2
MOV TH0, #0 ;Move para o valor de recarga do contador o valor 0.
MOV TL0, #0
SETB ET0
SETB TR0 ;Contador 0
```

;espera entrada pra começar o jogo

ENTRADA_PRA_INICIAR:

```
JB P3.2, ENTRADA_PRA_INICIAR
```

INICIAR_GAME:

```
JNB P0.0, LOOP_INSERT
JNB P0.1, COMPARA_JOGO
CALL RANDOM
CALL SALVA_RANDOM
SJMP INICIAR_GAME
```

lcd_inicio:

```
CLR RS
CLR P1.7
CLR P1.6
SETB P1.5
CLR P1.4
SETB EN
CLR EN
```


CALL delay 6

SETB EN

CLR EN

SETB P1.7

SETB EN

CLR EN

CALL delay

CLR P1.7

CLR P1.6

CLR P1.5

CLR P1.4

SETB EN

CLR EN

SETB P1.6

SETB P1.5

SETB EN

CLR EN

CALL delay

CLR P1.7

CLR P1.6

CLR P1.5

CLR P1.4

SETB EN

CLR EN

SETB P1.7

SETB P1.6

SETB P1.5

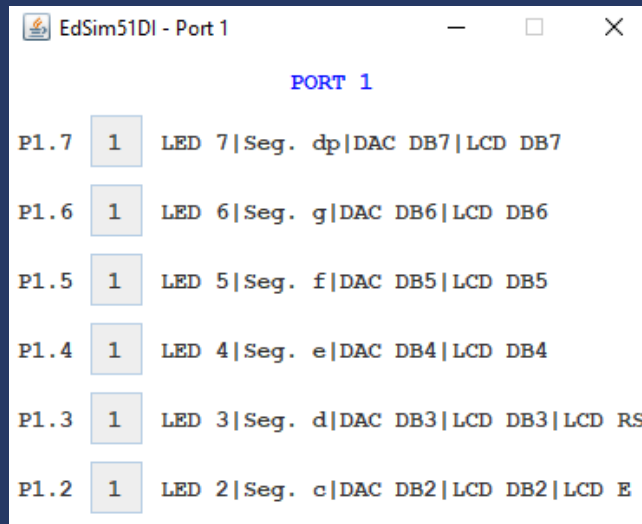
SETB P1.4

SETB EN

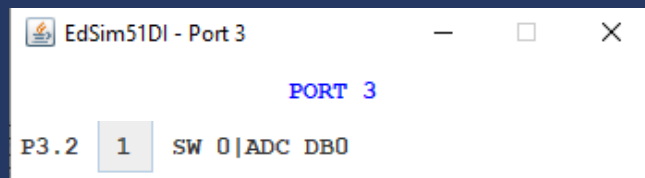
CLR EN

CALL delay

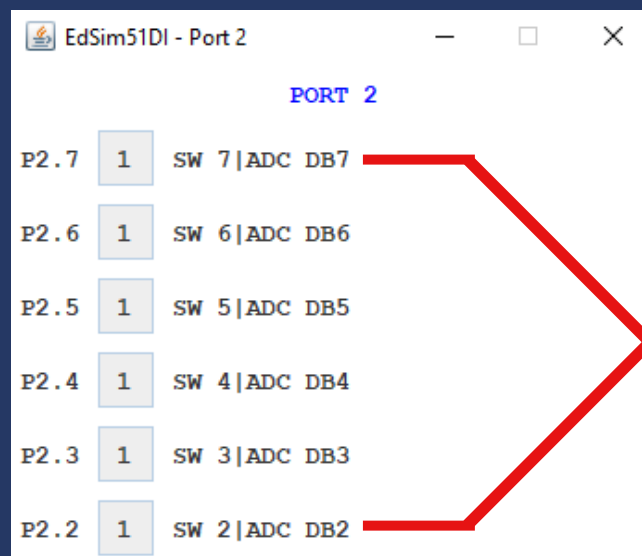
RET



Leds
do
Genius



Inicia o Jogo



Botões
para o
jogador
acender os
LEDs

sendCharacter: 7

```
SETB RS  
MOV C, ACC.7  
MOV P1.7, C  
MOV C, ACC.6  
MOV P1.6, C  
MOV C, ACC.5  
MOV P1.5, C  
MOV C, ACC.4  
MOV P1.4, C  
SETB EN  
CLR EN
```

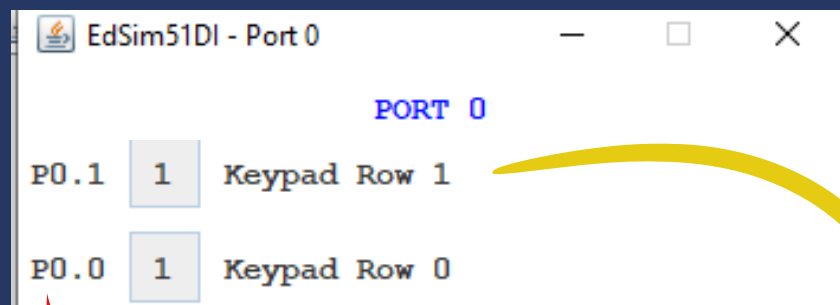
```
MOV C, ACC.3  
MOV P1.7, C  
MOV C, ACC.2  
MOV P1.6, C  
MOV C, ACC.1  
MOV P1.5, C  
MOV C, ACC.0  
MOV P1.4, C  
SETB EN  
CLR EN
```

```
CALL delay
```

```
RET
```

posicionaCursor:

```
CLR RS  
SETB P1.7  
MOV C, ACC.6  
MOV P1.6, C  
MOV C, ACC.5  
MOV P1.5, C  
MOV C, ACC.4  
MOV P1.4, C  
SETB EN  
CLR EN
```



Confirma que o Genius gerou a sequência de cores

Confirma que o jogador colocou a sequência e faz a comparação

8

```
MOV C, ACC.3
MOV P1.7, C
MOV C, ACC.2
MOV P1.6, C
MOV C, ACC.1
MOV P1.5, C
MOV C, ACC.0
MOV P1.4, C
SETB EN
CLR EN
```

CALL delay

RET

;reinicia o display

retornaCursor:

```
CLR RS
CLR P1.7
CLR P1.6
CLR P1.5
CLR P1.4
SETB EN
CLR EN
```

```
CLR P1.7
CLR P1.6
SETB P1.5
SETB P1.4
SETB EN
CLR EN
CALL delay
```

RET

Limpa o display
clearDisplay:
CLR RS
CLR P1.7

9

```
CLR P1.6
CLR P1.5
CLR P1.4
```

```
SETB EN
CLR EN
```

```
CLR P1.7
CLR P1.6
CLR P1.5
SETB P1.4
```

```
SETB EN
CLR EN
```

```
CALL delay
RET
```

```
CLR P1.6
CLR P1.5
CLR P1.4
SETB EN
CLR EN
```

```
CLR P1.7
CLR P1.6
CLR P1.5
SETB P1.4
SETB EN
CLR EN
```

CALL delay

RET

delay:

```
MOV R3, #50
DJNZ R3, $
RET
```

FIM