|  |  |  |  |
| --- | --- | --- | --- |
| Nome: | Lucas da Silva Carrassco | R.A.: | 22.120.053-8 |
| Nome: | William Yang | R.A.: | 22.121.043-8 |

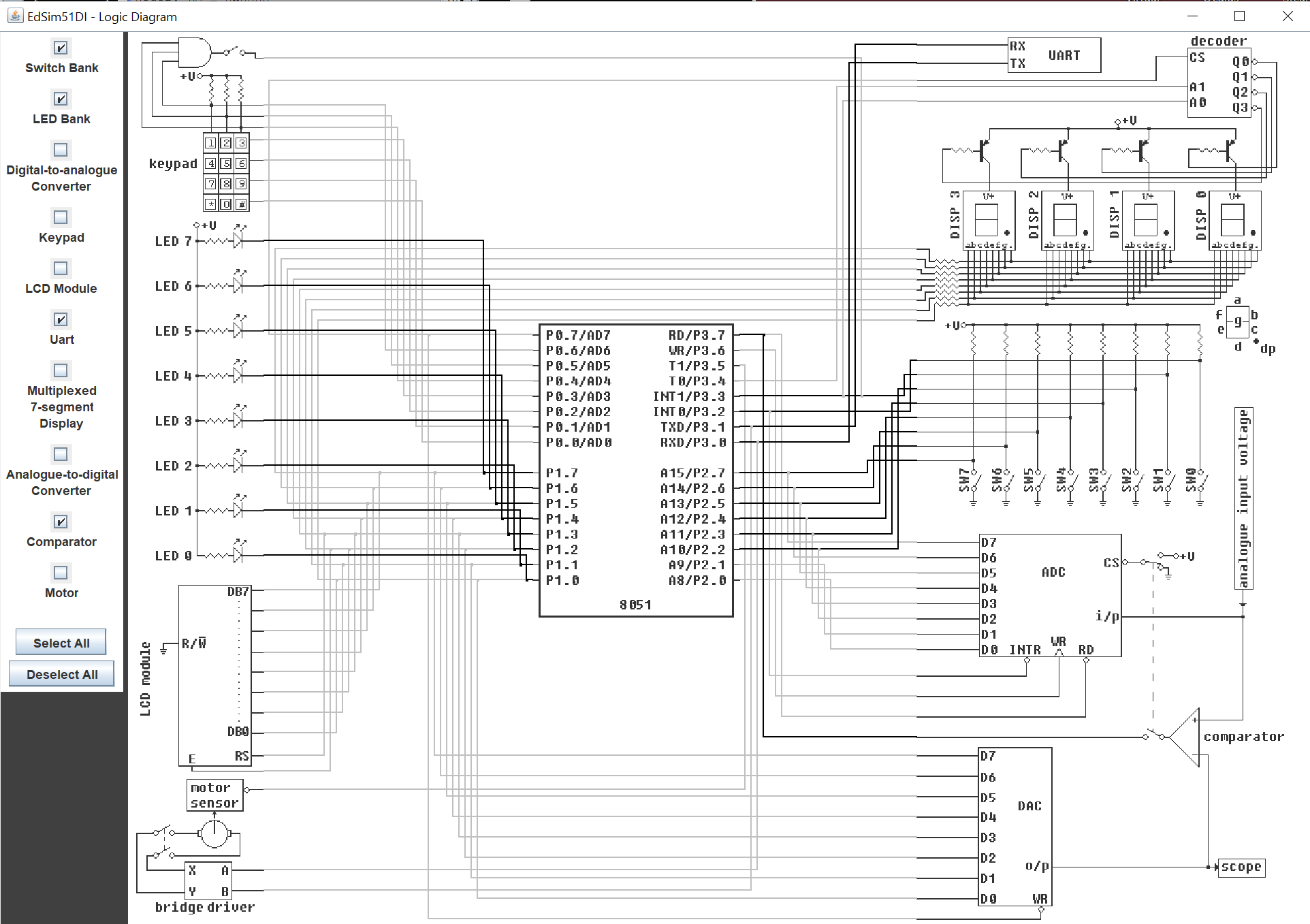
Projeto de Arquitetura de Computadores

# Descrição do Projeto

O projeto realizado pelo grupo foi de uma fechadura eletrônico utilizando o microcontrolador 8051 simulado no EdSim51 Onde o usuário digitaria uma combinação alfanumérica de 4 dígitos na porta serial RS-232 e o programa ira escrever cada byte no endereço de memória, assim registrando a senha e “travando” a fechadura (indicado pelo pino p2.0 e p2.1 onde 1 = travado e 0 = aberto). Apos o registro da senha o usuário precisara inserir a combinação correta de 4 dígitos para abrir a fechadura, caso erro o pino p1.0 ira ficar piscando indicando que a senha esta incorreta e apos de um tempo quando parar de piscar o usuário vai poder tentar inserir a senha de novo. Quando a combinação inserida for a correta, os pinos p2.0 e p2.1 iram trocar seu estado para 0 indicando a abertura e todos os pinos p1 iram acender por um tempo e apagar.

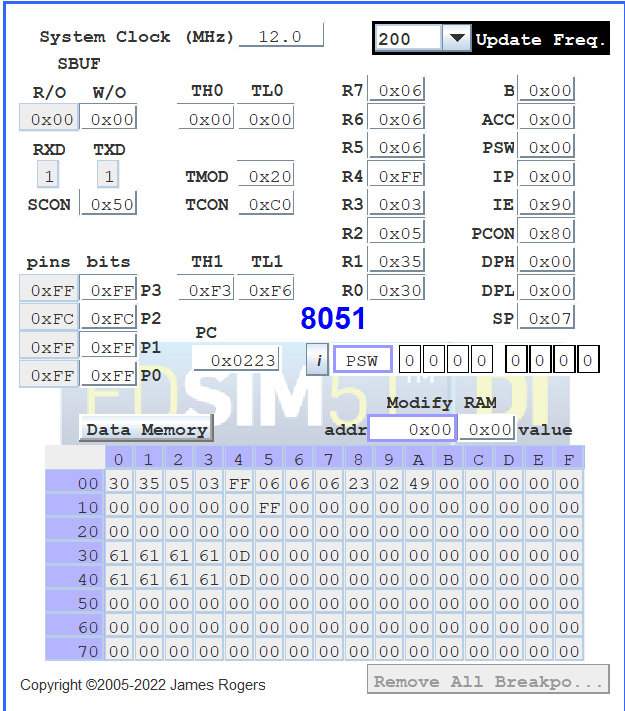
# Desenhos esquemáticos

O desenho esquemático do edsim51, mostrando as partes que estão sendo utilizadas.



# Fluxograma ou Diagrama

Fluxograma ou desenho descrevendo as posições de memória, sub-rotinas, etc.;



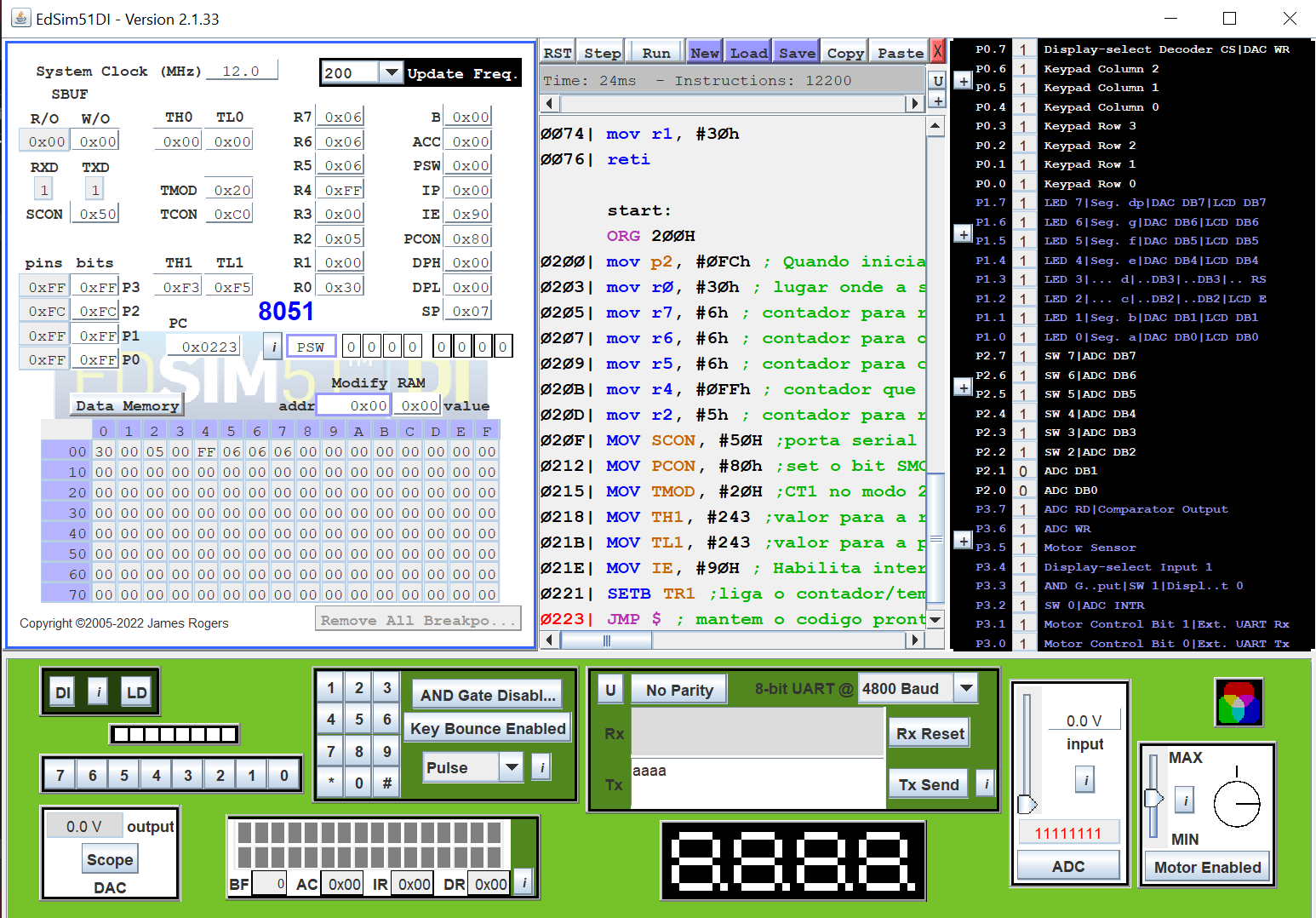
R0 ~ R7 = contadores para o programa

Endereço 30 ~ 33 da memória = registro da senha

Endereço 40~ 43 = tentativas

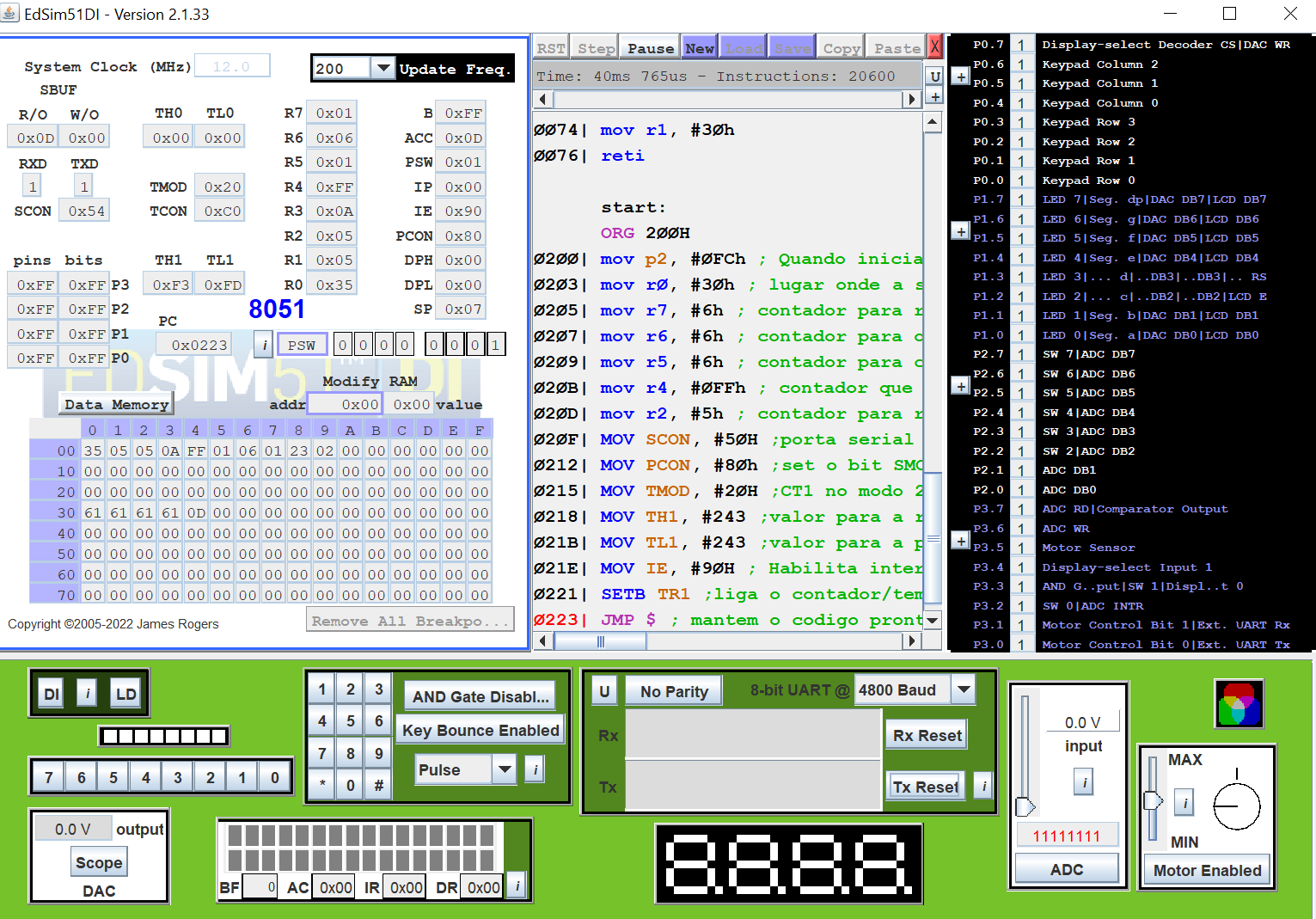
# Imagens da simulação realizada na IDE

Algumas Imagens da simulação realizada na IDE, com telas apresentando os resultados obtidos.

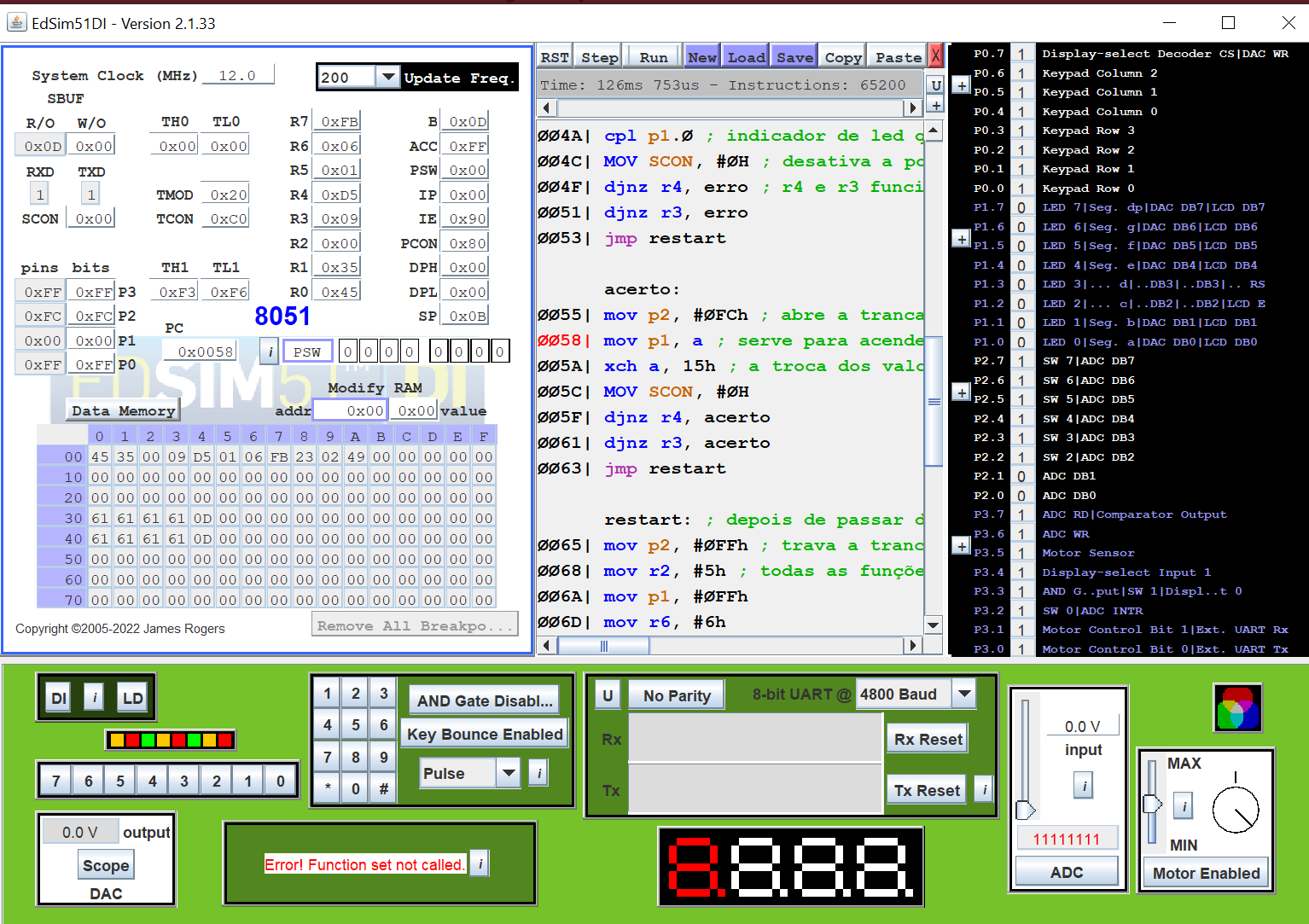
Primeiro instante onde o programa esta esperando o registro da senha e com as portas p2.0 e p2.1 no estado 0 (Indicados pelos retângulos vermelho).

O programa registrando a senha nos endereço de memória de 30 – 33, e mudando o estado dos pinos p2.0 e p2.1 para 1.

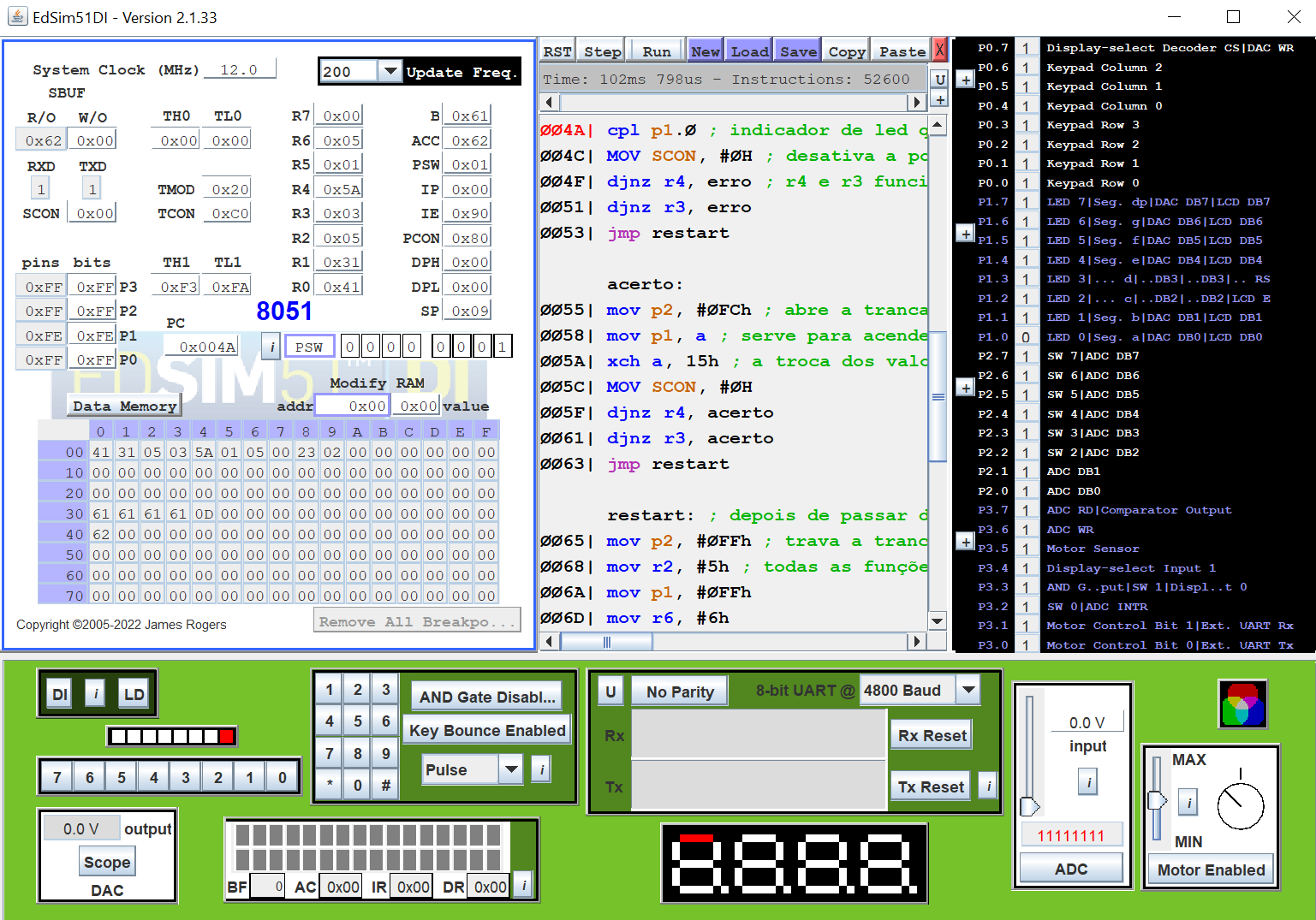
Obs. A letra “a” minuscula equivale a 61 na tabela ASCII.



Quando a senha, registrada no endereço de 40 – 43, for correta, os acendera os pinos p1 por um período e mudara o estado dos pinos p2.0 e p2.1 para 0 indicando a abertura .



Em caso de erro o programa ira piscar o pino p1.0 um tempo para depois poder receber o input de novo.



# Discussões e conclusões

Encontramos várias dificuldades durante o projeto durante a implementação da lógica, por exemplo, na parte de como realizar o registro da primeira leitura e pular par outra linha para registrar as tentativas e sobrescrever elas em caso de erro, realizar a rotina para ele somente só fazer a verifica-o das tentativas sem afetar a senha registrada e muitos mais.

Percebemos que a programação em linguagem assembly realmente é uma coisa diferente, precisando de resolver coisas de um jeito que não funciona em outra linguagem.

# Código-fonte

org 000h

jmp start

org 023H ; PONTEIRO DA INTERRUPCAO PARA CANAL SERIAL

mov p2, #0FFh ; quando entrar na interrupção a tranca é fechada

djnz r7,escrita ; registro da senha

mov r1, #30h ; lugar que vai registrar a senha

mov r0, #40h ; indicador usado para comparação de senhas

djnz r6, escrita ; confirmacao entra aqui para ele entrar de novo na func de escrever

reti

escrita: ; funciona tanto para registro de senha quanto para a comparação

mov r3, #0Ah ; um timer para os acertos e erros

MOV A,SBUF ; REALIZA A LEITURA DO BYTE RECEBIDO

MOV @r0, A ; ESCREVE O VALOR NO ENDEREÇO 30H

mov a, @r0 ; na primeira passagem r0 registra a senha, na segunda é usada para a comparação tambem

mov b, @r1 ; r1 utilizado para a comparação de senhas

inc r0 ; os inc's fazem parte tanto do registro quanto da comparação

inc r1

CLR RI ; RESETA RI PARA RECEBER NOVO BYTE

djnz r5, return ; utilizado para fazer o registro e travar o codigo, para o mesmo não chegar na comparação no primeiro input

inc r5 ; mantem o r5 em 1 para n quebrar o codigo.

cjne a, b, erro ; se a senha digitada for diferente da senha registrada cjne chama a subrotina erro

djnz r2, return ; r2 conta os acertos até o fim da senha

mov 15h, #0ffh ; serve para o led do indicador de acertos

clr a

acall acerto

return:

reti

erro:

cpl p1.0 ; indicador de led que mostra o erro, piscando apenas o primeiro Led

MOV SCON, #0H ; desativa a porta serial para n receber nada até o fim do tempo de espera

djnz r4, erro ; r4 e r3 funcionam como timers

djnz r3, erro

jmp restart

acerto:

mov p2, #0FCh ; abre a tranca, localizada no P2.0 e P2.1

mov p1, a ; serve para acender/apagar todos os Leds

xch a, 15h ; a troca dos valores dentro da função faz os leds piscarem

MOV SCON, #0H

djnz r4, acerto

djnz r3, acerto

jmp restart

restart: ; depois de passar do acerto ou erro, o restart prepara o programa para receber outra tentativa de acesso

mov p2, #0FFh ; trava a tranca caso tenha passado pelo acerto

mov r2, #5h ; todas as funções até o reti resetam o programa para receber outra senha

mov p1, #0FFh

mov r6, #6h

MOV SCON, #50H

mov r0, #40h

mov r1, #30h

reti

start:

ORG 200H

mov p2, #0FCh ; Quando iniciado a tranca fica aberta até o registro da senha

mov r0, #30h ; lugar onde a senha vai ser salvo

mov r7, #6h ; contador para registrar os 4 numeros

mov r6, #6h ; contador para os input

mov r5, #6h ; contador para conferir a senha

mov r4, #0FFh ; contador que funciona como um "Timer"

mov r2, #5h ; contador para registrar os acertos

MOV SCON, #50H ;porta serial no modo 1 e habilita a recepção

MOV PCON, #80h ;set o bit SMOD

MOV TMOD, #20H ;CT1 no modo 2

MOV TH1, #243 ;valor para a recarga

MOV TL1, #243 ;valor para a primeira contagem

MOV IE, #90H ; Habilita interrupção serial

SETB TR1 ;liga o contador/temporizador 1

JMP $ ; mantem o codigo pronto para o input