centro universitário FEI Curso de Ciência da Computação
CE4411 – Arquitetura de Computadores
Relatório de Projeto

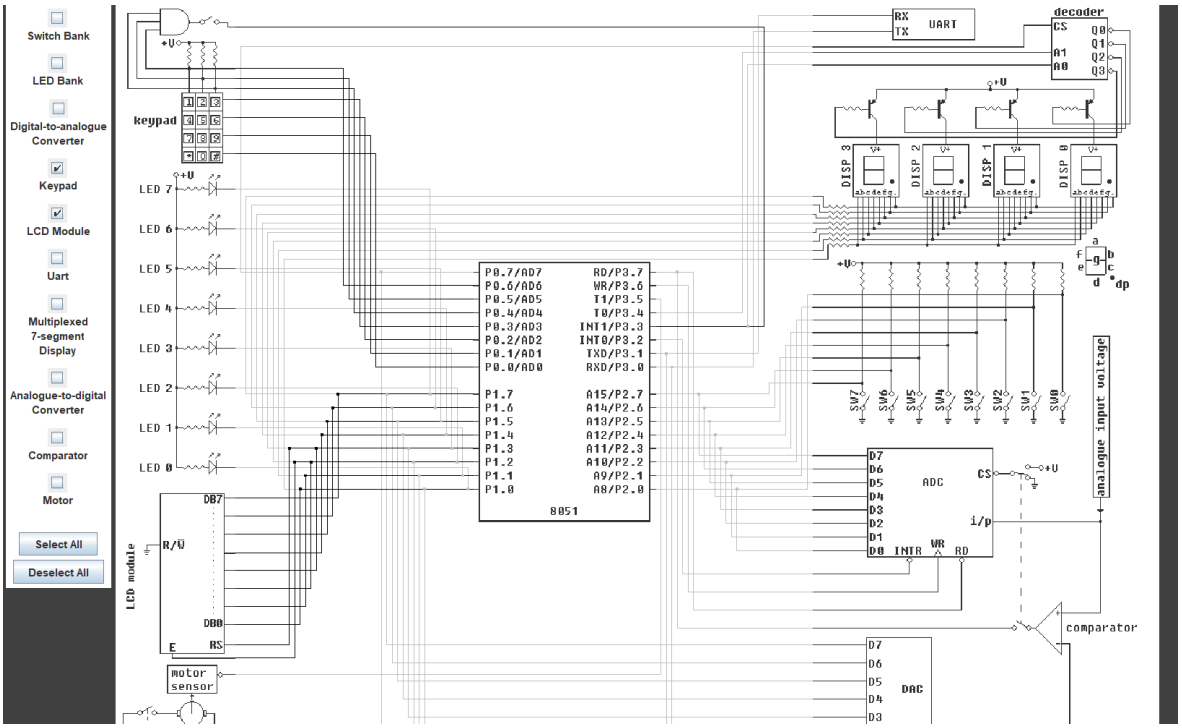
Nome: João Vitor Governatore	R.A.: 24.122.027-6
Nome: Pedro Henrique Lega Kramer Costa	R.A.: 24.122.049-0

Projeto de Arquitetura de Computadores

1. Descrição do Projeto

Projeto da máquina de refrigerante contendo 5 itens (botões 1 a 5) e 2 botões de controle para visualização no LCD(*(para cima) e #(para baixo)). Para fácil interatividade o botão subir foi implementado com uma lógica circular para demonstrar que ao clicar no botão de subir o Display irá para o último item. Já o botão de descer ele não é circular, pois o objetivo é mostrar que chegou na última opção de bebidas.
O projeto para sua fase final pretende deixar os botões de (1-5) totalmente funcionais, o que ainda está sendo implementado pelo grupo, além de uma tela que indica o quanto que falta para acabar o preparo das bebidas.Os botões de subida e descida já foram implementados corretamente

2. Desenhos esquemáticos

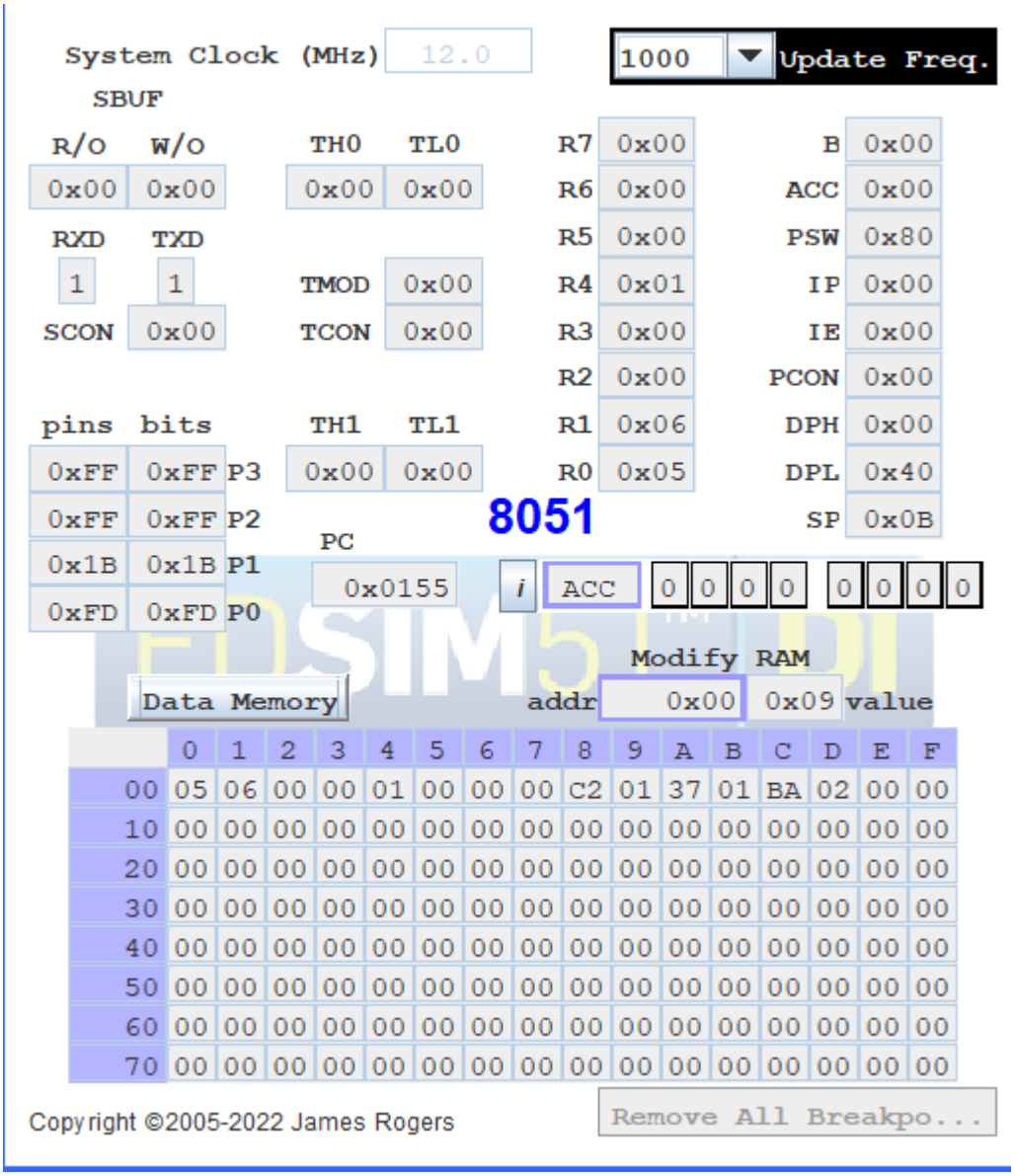


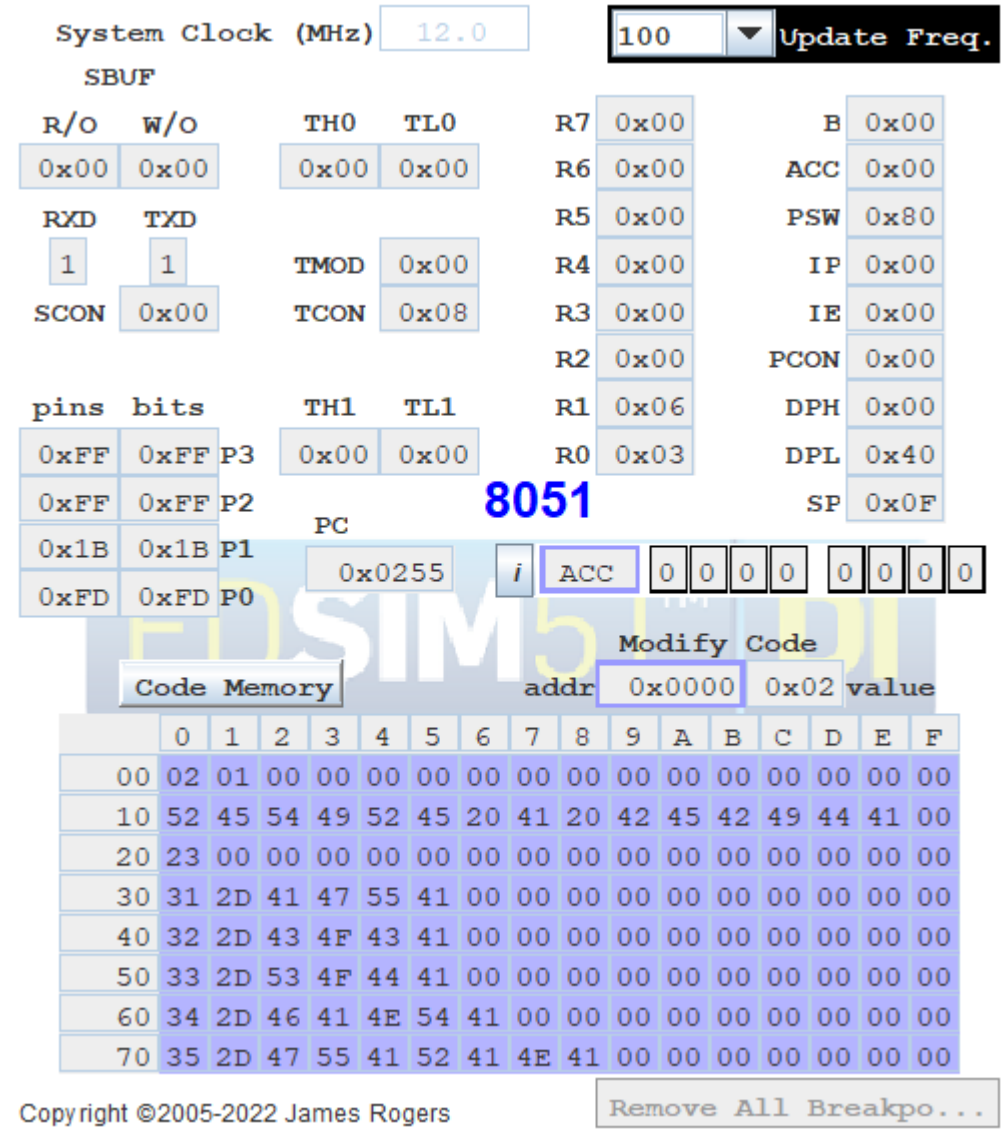
Foi utilizado o display LCD e KeyPad

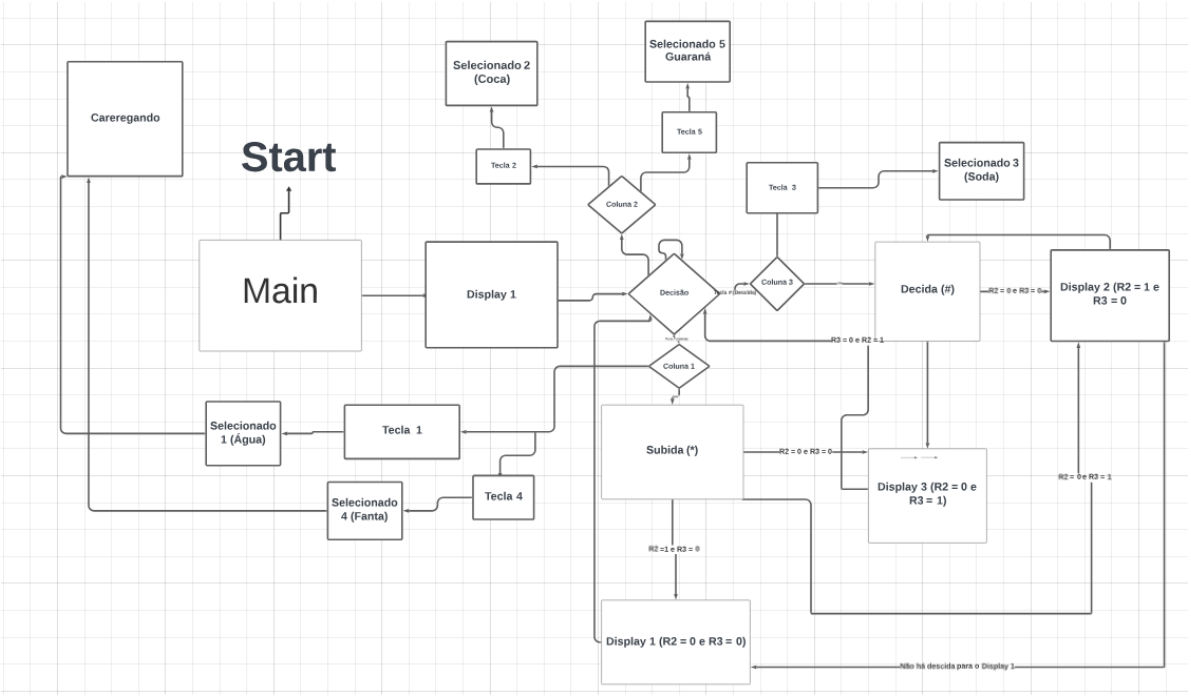
3. Fluxograma ou Diagrama

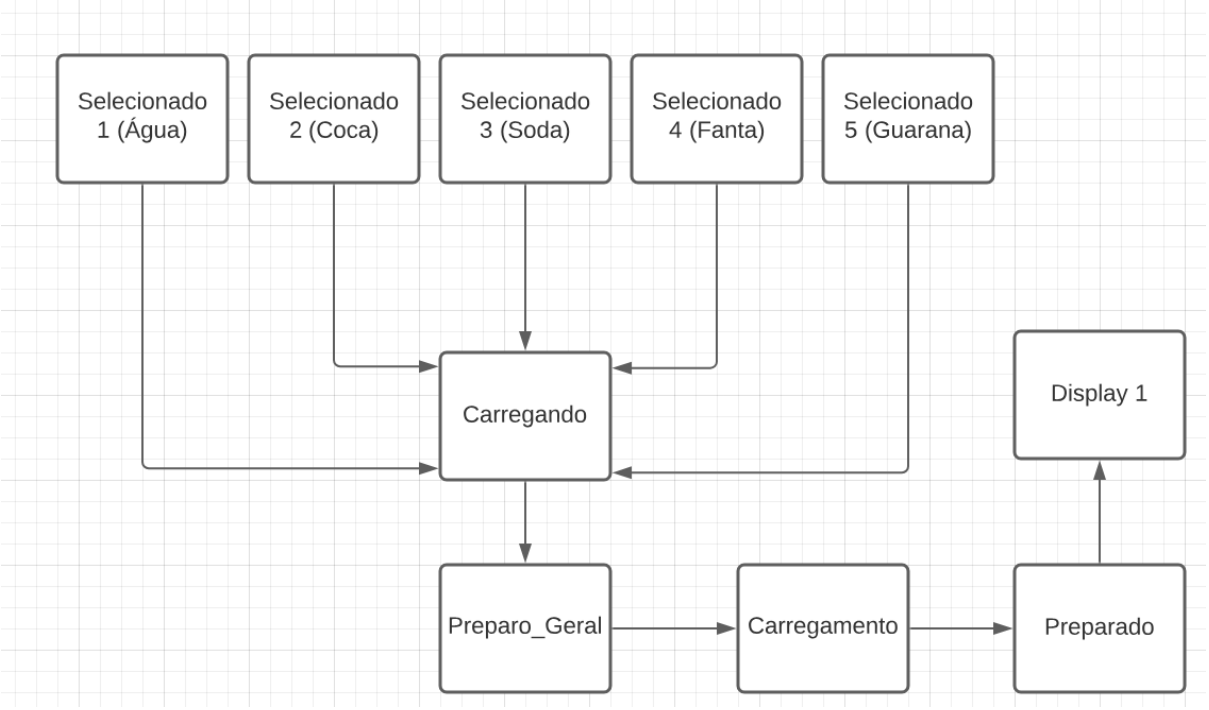
Fluxograma ou desenho descrevendo as posições de memória, sub-rotinas, etc.;

Exemplo mostrando os caracteres **a, b, c, d, e** na memória:



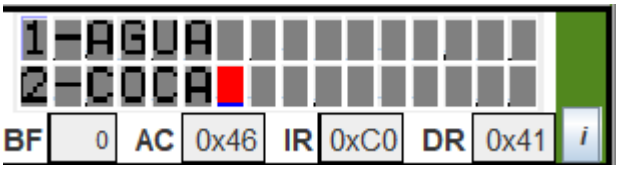

System Clock (MHz) 12.0
100 Update Freq.
SBUF
R/O W/O
0x00 0x00
RXD TXD
1 1
SCON 0x00
TH0 TL0
0x00 0x00
TMOD 0x00
TCON 0x08
TH1 TL1
0x00 0x00
R7 0x00
R6 0x00
R5 0x00
R4 0x00
R3 0x00
R2 0x00
R1 0x06
R0 0x03
B 0x00
ACC 0x00
PSW 0x80
IP 0x00
IE 0x00
PCON 0x00
DPH 0x00
DPL 0x40
SP 0x0F
pins bits
0xFF 0xFF P3
0xFF 0xFF P2
0x1B 0x1B P1
0xFD 0xFD P0
8051
PC
0x0255
i
ACC
0 0 0 0 0 0 0 0
Modify Code
Code Memory
addr 0x0000 0x02 value
0 1 2 3 4 5 6 7 8 9 A B C D E F
00 02 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 52 45 54 49 52 45 20 41 20 42 45 42 49 44 41 00
20 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 31 2D 41 47 55 41 00 00 00 00 00 00 00 00 00 00
40 32 2D 43 4F 43 41 00 00 00 00 00 00 00 00 00 00
50 33 2D 53 4F 44 41 00 00 00 00 00 00 00 00 00 00
60 34 2D 46 41 4E 54 41 00 00 00 00 00 00 00 00 00
70 35 2D 47 55 41 52 41 4E 41 00 00 00 00 00 00 00
Copyright ©2005-2022 James Rogers
Remove All Breakpo...



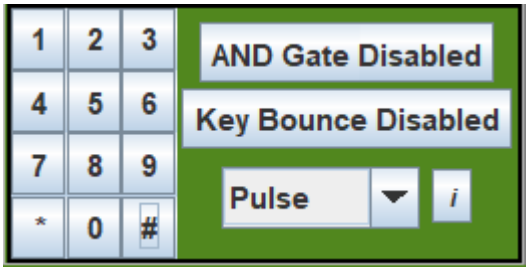


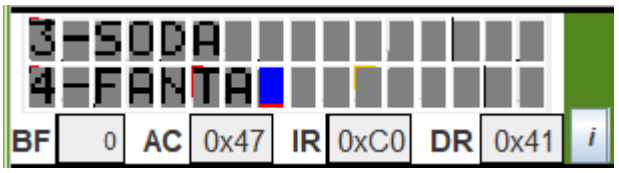
4. Imagens da simulação realizada na IDE

1) Ao rodar o código

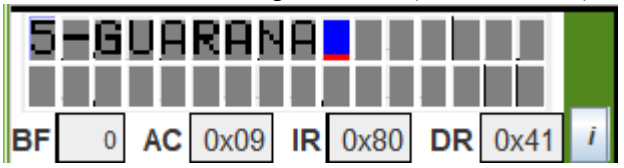


2) Ao clicar o botão para baixo(# no teclado)

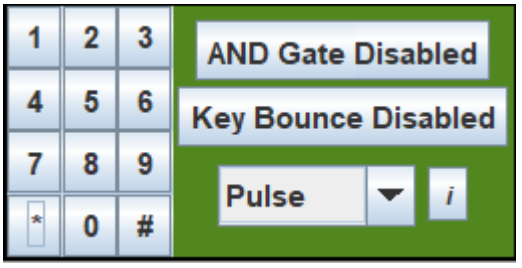


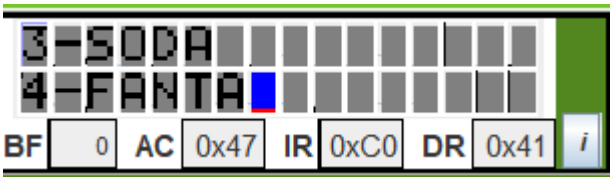


Ao clicar novamente para baixo(# no teclado)

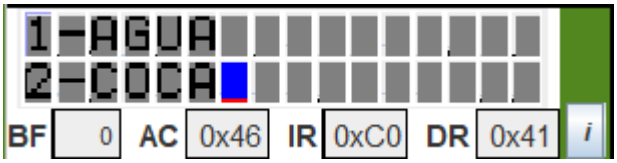


3) Ao clicar o botão para cima(* no teclado)

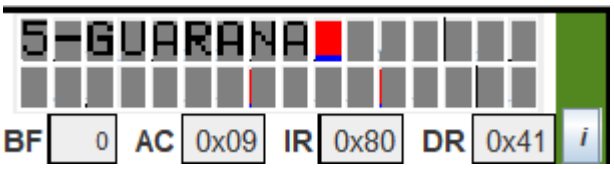




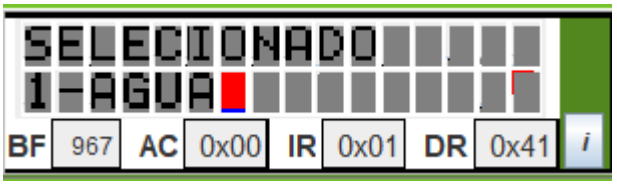
Ao clicar novamente para cima(* no teclado)

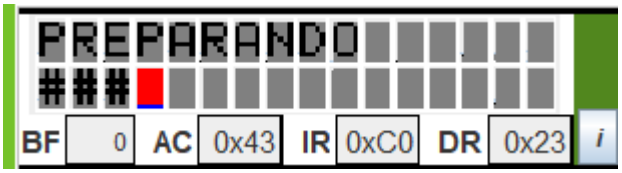


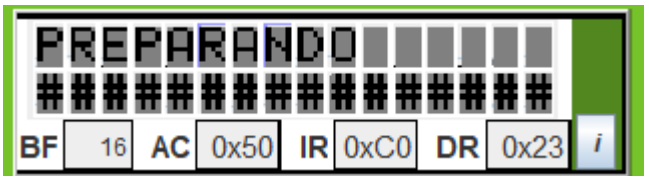
Ao clicar novamente para baixo(# no teclado)

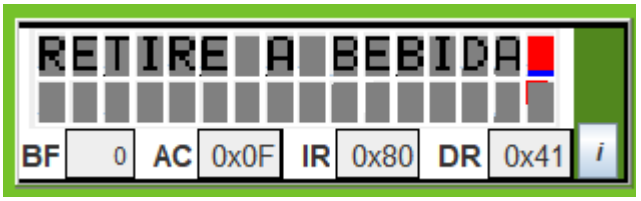


4) Ao clicar em qualquer botão de 1-5(possíveis opções) será mostrado a seguir o pedido solicitado e irá carregá-lo, em seguida aparecerá para retirar o pedido. Por fim irá voltar novamente ao menu (Exemplo com o item 1-Água)









5. Discussões e conclusões

Descreva sobre o processo de desenvolvimento do projeto, tais como:

- Descreva os desafios encontrados no projeto, as dificuldades.
- O que você aprendeu de novo com o projeto (extraclasse, que não foi ensinado em aula).
- Qual a sua visão em relação à programação em linguagem assembly (houve a necessidade de conhecer bem o hardware?) em comparação com outras linguagens.

1. Um dos desafios que surgiram durante o desenvolvimento do projeto foi em relação ao reconhecimento da tecla pressionada pelo botão, no qual apresentamos dificuldades em direcionar corretamente para a sub - rotina desejada ao clicar nesses botões. Além disso, tivemos dificuldade em implementar a lógica para apresentar as bebidas conforme o descer e subir dos botões. Porém essas dificuldades foram superadas à medida em que aprofundamos nosso entendimento dos botões do microcontrolador 8051, permitindo alcançar um resultado satisfatório.

2. Houve sucesso em compreender como funciona a pinagem para o KeyPad e a escrita no LCD a partir do código apresentado em aula que foi adaptado para o projeto

3.
    a. A linguagem assembly apresentou-se durante o desenvolvimento do código com muitas restrições o que causou dificuldade, acabando sendo necessário refazer grande parte do trabalho já implementado a fim de descobrir como corrigir o erro.
    b. Fora as dificuldades apresentadas houveram muitos ensinamentos, como por exemplo a reaproveitação de variáveis e a reescrita das mesmas (registradores, acumuladores, entre outros..). Também aprimoramos o nosso entendimento sobre o funcionamento da memória, melhorando nossa capacidade de otimizar o uso dos recursos de hardware.

6. Código-fonte

```
; --- Mapeamento de Hardware (8051) ---
   RS      equ     P1.3    ;Reg Select ligado em P1.3
   EN      equ     P1.2    ;Enable ligado em P1.2



;especificações do projeto: rodar o código com o keypad no pulse
;usar a frequencia 100hz como recomendada para o código



org 0000h
  LJMP START

;Alocando os Textos na memória
ORG 0010H
TERMINADO:
```

```
  DB "RETIRE A BEBIDA"
  DB 00H

ORG 0020H

JOGO_VELHA:
  DB "#"
  DB 00H

ORG 0030H

  AGUA:
    DB "1-AGUA"
    DB 00h

ORG 0040H

  COCA:
    DB "2-COCA"
    DB 00h

org 0050h
  SODA:
    DB "3-SODA"
    DB 00H

ORG 0060H
  FANTA:
    DB "4-FANTA"
    DB 00H

ORG 0070H

  GUARANA:
    DB "5-GUARANA"
    DB 00H

ORG 0080H

  SELECIONADO:
    DB "SELECIONADO "
```

```
   DB 00H

ORG 0090H

  PREPARANDO:
  DB "PREPARANDO "
  DB 00H

ORG 0100H

;começo do programa
START:

  ;Registradores responsáveis pelo controle de subida e descida (inicialização)
  MOV R2,#0H
  MOV R3,#0H

  ACALL lcd_init ;iniciar o LCD
  JMP DISPLAY1 ; chamo o display 1 (onde começa o Programa)

DISPLAY1:;Responsável por mostrar no LCD as 2 primeiras opções(agua e coca)

  ACALL clearDisplay;limpar o displya anterior

  ;CONTROLES DE SUBIDA E DECIDA:
  MOV R2,#0H ;->CONTROLADOR PARA O DISPLAY1
  MOV R3,#0H;->CONTROLADOR PARA O DISPLAY1

  ;Escrita no LCD
  MOV A, #00h; posiciono o cursor na primeira linha do display
  ACALL posicionaCursor
  MOV DPTR,#AGUA;endereço inicial de memória da agua
  ACALL escreveStringROM ; escrevo
  MOV A, #40h; posiciono novamente
  ACALL posicionaCursor
  MOV DPTR,#COCA;endereço inicial de memória da String Display LCD
  ACALL escreveStringROM
  JMP CONTROLLER; chamo o controller que é onde fara a verificação de qual tecla foi
pressionada

DISPLAY2:;Responsável por mostrar no LCD as 2 segundas opções(soda e fanta)
```

```
  ACALL clearDisplay;limpar o display anterior

  ;CONTROLES DE SUBIDA E DECIDA:
  MOV R2,#1H
  MOV R3,#0H

  ;Escrita no LCD
  MOV A , #00H
  ACALL posicionaCursor
  MOV DPTR, #SODA
  ACALL escreveStringROM
  MOV A , #40H
  ACALL posicionaCursor
  MOV DPTR, #FANTA
  ACALL escreveStringROM
  JMP CONTROLLER;chamo o controller que é onde fara a verificação de qual tecla foi
pressionada

DISPLAY3: ;Responsável por mostrar no LCD a ultima opção(guarana)

  ACALL clearDisplay;limpar o display anterior

  ;CONTROLES DE SUBIDA E DECIDA:
  MOV R2,#0H
  MOV R3,#1H

  ;Escrita no LCD
  MOV A , #00H
  ACALL posicionaCursor
  MOV DPTR, #GUARANA
  ACALL escreveStringROM
  JMP CONTROLLER;chamo o controller que é onde fara a verificação de qual tecla foi
pressionada

DESCIDA:; lógica para descer os displays ao pressionar a tecla #
  MOV A,R2
  MOV B,R3

  ;DISPLAY3->DISPLAY3
  CJNE A,B ,DISPLAY3; se o A for diferente do B vou para o Display 3
```

```
  ;DISPLAY1->DISPLAY2
  JZ DISPLAY2  ; Vai para o Display 2 se o A for 0
  ;DISPLAY2->DISPLAY3
  JNZ DISPLAY1 ; Vai para o Display 1 se o A não for 0

SUBIDA: ; lógica para subir os displays ao pressionar a tecla *
  MOV A,R2
  MOV B,R3
  JNZ DISPLAY1; Vai para o Display 1 se o A não for 0
  CJNE A,B ,DISPLAY2 ; se o A for diferente do B vou para o Display 2
  JZ DISPLAY3 ; Vai para o Display 3 se o A for 0



CONTROLLER: ; controla todos os botões pressionados
  CLR F0; limpa o lixo de memoria do botão
  ACALL LEITURA_TECLADO ;chama a leitura de teclado para verificar se algo foi
pressionado
  JNB F0, CONTROLLER   ;Se nenhum botão não for pressionado, volta pro controller
  JNB P0.4, COLUNA3; Pino referente a coluna 3 (3 e #) se um botão dessa coluna for
pessionado vai para a coluna 3
  JNB P0.5,COLUNA2; Pino referente a coluna 2 (2 e 5) se um botão dessa coluna for
pessionado vai para a coluna 2
  JNB P0.6,COLUNA1; Pino referente a coluna 1 (1 , 4 e *) se um botão dessa coluna for
pessionado vai para a coluna 1
  JMP CONTROLLER; se nada for pressionado retorna para a rotina novamente



COLUNA1: ;realizar uma outra leitura para saber a linha que o botão foi presionado
;Primeiro, ela verificará a coluna, como mencionado anteriormente. Quando a linha atual for
pressionada, ela identificará o botão específico que foi pressionado.
  JNB P0.3 , CARREGANDO_AGUA ; ENTRAR NO 1
  JNB P0.2, CARREGANDO_FANTA ;ENTRA NO 4
  JNB P0.0, SUBIDA ;ENTRA NO *
  SJMP CONTROLLER; se não clicar nesses botões volta para o Controller



COLUNA2: ;realizar uma outra leitura para saber a linha que o botão foi presionado
  ;Primeiro, ela verificará a coluna, como mencionado anteriormente. Quando a linha atual for
pressionada, ela identificará o botão específico que foi pressionado.
  JNB P0.3 , CARREGANDO_COCA ; ENTRAR NO 2
  JNB P0.2, CARREGANDO_GUARANA ;ENTRAR NO 5
```

```
  SJMP CONTROLLER; se não clicar nesses botões volta para o Controller

COLUNA3: ;realizar uma outra leitura para saber a linha que o botão foi presionado
  ;Primeiro, ela verificará a coluna, como mencionado anteriormente. Quando a linha atual for pressionada, ela identificará o botão específico que foi pressionado.

  JNB P0.0 , DESCIDA ; ENTRAR NO #
  JNB P0.3, CARREGANDO_SODA;ENTRAR NO 3
  SJMP CONTROLLER; se não clicar nesses botões volta para o Controller

CARREGANDO_AGUA: ; escreve o Display referente a agua
  ACALL clearDisplay
  MOV A, #00H
  ACALL posicionaCursor
  MOV DPTR, #SELECIONADO
  ACALL escreveStringROM
  MOV A,#40H
  ACALL posicionaCursor
  MOV DPTR, #AGUA
  ACALL escreveStringROM
  ACALL PREPARO_GERAL ; depois de escrever vai para o preparo geral

CARREGANDO_COCA:; escreve o Display referente a coca
  ACALL clearDisplay
  MOV A, #00H
  ACALL posicionaCursor
  MOV DPTR, #SELECIONADO
  ACALL escreveStringROM
  MOV A,#40H
  ACALL posicionaCursor
  MOV DPTR, #COCA
  ACALL escreveStringROM
  ACALL PREPARO_GERAL ; depois de escrever vai para o preparo geral

CARREGANDO_SODA:; escreve o Display referente a soda
  ACALL clearDisplay
  MOV A, #00H
  ACALL posicionaCursor
  MOV DPTR, #SELECIONADO
  ACALL escreveStringROM
  MOV A,#40H
```

```
  ACALL posicionaCursor
  MOV DPTR, #SODA
  ACALL escreveStringROM
  ACALL PREPARO_GERAL ; depois de escrever vai para o preparo geral


CARREGANDO_FANTA:; escreve o Display referente a fanta
  ACALL clearDisplay
  MOV A, #00H
  ACALL posicionaCursor
  MOV DPTR, #SELECIONADO
  ACALL escreveStringROM
  MOV A,#40H
  ACALL posicionaCursor
  MOV DPTR, #FANTA
  ACALL escreveStringROM
  ACALL PREPARO_GERAL ; depois de escrever vai para o preparo geral


CARREGANDO_GUARANA:; escreve o Display referente ao guaraná
  ACALL clearDisplay
  MOV A, #00H
  ACALL posicionaCursor
  MOV DPTR, #SELECIONADO
  ACALL escreveStringROM
  MOV A,#40H
  ACALL posicionaCursor
  MOV DPTR, #GUARANA
  ACALL escreveStringROM
  ACALL PREPARO_GERAL ; depois de escrever vai para o preparo geral



  PREPARO_GERAL:; Nessa Sub-rotina vai escrever no display preparando e depois vai para
a subrotina de Carregamento
  ACALL clearDisplay
  MOV A, #00H
  ACALL posicionaCursor
  MOV DPTR,#PREPARANDO
  ACALL escreveStringROM
  MOV A, #40H
```

```
  MOV R4, #15h
  ACALL posicionaCursor
  ACALL CARREGAMENTO
  JMP CONTROLLER

CARREGAMENTO:; Nessa Subrotina será responsável por escrever # na segunda linha do
display, que é referente ao preparo da bebida. Simulando um carregamento
  MOV DPTR, #JOGO_VELHA
  ACALL escreveStringROM
  ACALL DELAY
  INC A
  DJNZ R4, CARREGAMENTO
  JMP PREPARADO; ao acabar de escrever vai para a subrotina de preparado

PREPARADO:; responsavel por limpar o display e escrerver Retire a Sua Bebida, depois disso
volta para o display 1 para que possa pedir outra bebida
  ACALL clearDisplay
  MOV A, #00H
  ACALL posicionaCursor
  MOV DPTR,#TERMINADO
  ACALL escreveStringROM
  JMP DISPLAY1; Menu Principal



;funções abaixo foram mostradas em sala para leitura do keypad e escrita no LCD

LEITURA_TECLADO:
  MOV R0, #0                    ; clear R0 - the first key is key0

  ; SCANEA A LINHA 0
  MOV P0, #0FFh
  CLR P0.0                      ; clear row0
  CALL LER_COLUNAS              ; call column-scan subroutine
  JB F0, ACHOU                  ; | if F0 is set, jump to end of program
          ; | (because the pressed key was found and its number is in  R0)
  ; SCANEA A LINHA 1
  SETB P0.0                     ; set row0
  CLR P0.1                      ; clear row1
  CALL LER_COLUNAS      ; call column-scan subroutine
  JB F0, ACHOU                  ; | if F0 is set, jump to end of program
```

```
        ; | (because the pressed key was found and its number is in  R0)
  ; SCANEA A LINHA 2
  SETB P0.1                     ; set row1
  CLR P0.2                      ; clear row2
  CALL LER_COLUNAS              ; call column-scan subroutine
  JB F0, ACHOU                  ; | if F0 is set, jump to end of program
        ; | (because the pressed key was found and its number is in  R0)
  ; SCANEA A LINHA 3
  SETB P0.2                     ; set row2
  CLR P0.3                      ; clear row3
  CALL LER_COLUNAS              ; call column-scan subroutine
  JB F0, ACHOU                  ; | if F0 is set, jump to end of program
        ; | (because the pressed key was found and its number is in  R0)
ACHOU:
  RET

LER_COLUNAS:
  JNB P0.4, TECLA    ; if col0 is cleared - key found
  INC R0                        ; otherwise move to next key
  JNB P0.5, TECLA    ; if col1 is cleared - key found
  INC R0                        ; otherwise move to next key
  JNB P0.6, TECLA    ; if col2 is cleared - key found
  INC R0                        ; otherwise move to next key
  RET                           ; return from subroutine - key not found

TECLA:
  SETB F0                       ; key found - set F0
  RET                           ; and return from subroutine



escreveStringROM:
  MOV R1, #00h
  ; Inicia a escrita da String no Display LCD
loop:
  MOV A, R1
  MOVC A,@A+DPTR          ;lê da memória de programa
  JZ finish                 ; if A is 0, then end of data has been reached - jump out of loop
  ACALL sendCharacter       ; send data in A to LCD module
  INC R1                        ; point to next piece of data
   MOV A, R1
```

```
  JMP loop              ; repeat
finish:
  RET

; initialise the display
; see instruction set for details
lcd_init:

  CLR RS                ; clear RS - indicates that instructions are being sent to the module

; function set
  CLR P1.7              ; |
  CLR P1.6              ; |
  SETB P1.5             ; |
  CLR P1.4              ; | high nibble set

  SETB EN               ; |
  CLR EN                ; | negative edge on E

  CALL delay            ; wait for BF to clear
        ; function set sent for first time - tells module to go into 4-bit mode
; Why is function set high nibble sent twice? See 4-bit operation on pages 39 and 42 of
HD44780.pdf.

  SETB EN               ; |
  CLR EN                ; | negative edge on E
        ; same function set high nibble sent a second time

  SETB P1.7             ; low nibble set (only P1.7 needed to be changed)

  SETB EN               ; |
  CLR EN                ; | negative edge on E
      ; function set low nibble sent
  CALL delay            ; wait for BF to clear


; entry mode set
; set to increment with no shift
  CLR P1.7              ; |
  CLR P1.6              ; |
  CLR P1.5              ; |
```

```
  CLR P1.4          ; | high nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  SETB P1.6         ; |
  SETB P1.5         ; |low nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  CALL delay        ; wait for BF to clear


; display on/off control
; the display is turned on, the cursor is turned on and blinking is turned on
  CLR P1.7          ; |
  CLR P1.6          ; |
  CLR P1.5          ; |
  CLR P1.4          ; | high nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  SETB P1.7         ; |
  SETB P1.6         ; |
  SETB P1.5         ; |
  SETB P1.4         ; | low nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  CALL delay        ; wait for BF to clear
  RET


sendCharacter:
  SETB RS           ; setb RS - indicates that data is being sent to module
  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                   ; | high nibble set

  SETB EN               ; |
  CLR EN                ; | negative edge on E

  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                   ; | low nibble set

  SETB EN               ; |
  CLR EN                ; | negative edge on E

  CALL delay            ; wait for BF to clear
  CALL delay            ; wait for BF to clear
  RET

;Posiciona o cursor na linha e coluna desejada.
;Escreva no Acumulador o valor de endereço da linha e coluna.
;|-----------------------------------------------------------------------------------|
;|linha 1 | 00 | 01 | 02 | 03 | 04 |05 | 06 | 07 | 08 | 09 |0A | 0B | 0C | 0D | 0E | 0F |
;|linha 2 | 40 | 41 | 42 | 43 | 44 |45 | 46 | 47 | 48 | 49 |4A | 4B | 4C | 4D | 4E | 4F |
;|-----------------------------------------------------------------------------------|
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                   ; | high nibble set
```

```
  SETB EN                 ; |
  CLR EN                  ; | negative edge on E

  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                     ; | low nibble set

  SETB EN                 ; |
  CLR EN                  ; | negative edge on E

  CALL delay              ; wait for BF to clear
  CALL delay              ; wait for BF to clear
  RET


;Retorna o cursor para primeira posição sem limpar o display
retornaCursor:
  CLR RS
  CLR P1.7          ; |
  CLR P1.6          ; |
  CLR P1.5          ; |
  CLR P1.4          ; | high nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  CLR P1.7          ; |
  CLR P1.6          ; |
  SETB P1.5         ; |
  SETB P1.4         ; | low nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  CALL delay        ; wait for BF to clear
  RET
```

```
;Limpa o display
clearDisplay:
  CLR RS
  CLR P1.7          ; |
  CLR P1.6          ; |
  CLR P1.5          ; |
  CLR P1.4          ; | high nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  CLR P1.7          ; |
  CLR P1.6          ; |
  CLR P1.5          ; |
  SETB P1.4         ; | low nibble set

  SETB EN           ; |
  CLR EN            ; | negative edge on E

  MOV R6, #40
  rotC:
  CALL delay        ; wait for BF to clear
  DJNZ R6, rotC
  RET


delay:
  MOV R0, #20
  DJNZ R0, $
  RET
```

Anexo referente ao .asm será enviado na página de entrega também