Faculdade de Ciências Exatas e da Engenharia

Licenciatura em Engenharia Informática

Arquitetura de Computadores

2ºProjeto – Máquina de Vendas Automática

DOCENTES:

Dionísio Barros

Sofia Inácio

Pedro Camacho

Dino Vasconcelos

DISCENTES:

Bjorn André Costa Foss nº2048319

Joana Andrade Azevedo nº2076220

Conteúdo

1. Introdução

No ambiento da cadeira Arquitetura de Computadores, foi proposto a realização de uma máquina de vendas, com uma interface gráfica que deverá simular a seleção e venda de um produto.

Foi utilizado o processador PEPE com recurso ao simulador em Java de forma a testar o programa a ser desenvolvido.

A linguagem utilizada foi o assembly de forma a converter as operações lógicas em operações de máquinas que é interpretado pelo computador.

2. Objetivo

O objetivo deste trabalho é desenvolver um programa que execute as funções de uma máquina automática de venda de produtos. O programa simula a seleção e venda de produtos, bem como a visualização do stock.

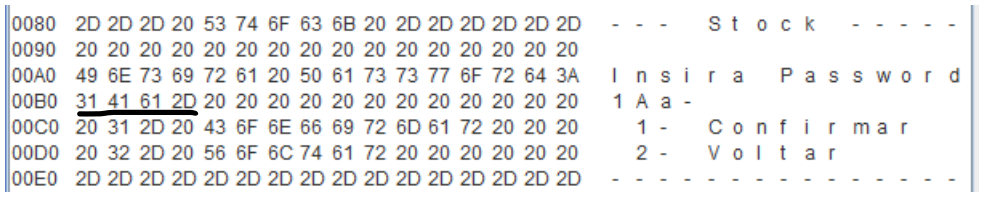
O programa deve permitir que o usuário selecione produtos de duas categorias (bebidas e snacks), visualize seus preços e faça o pagamento utilizando apenas moedas de determinado valor. Além disso, o programa deve imprimir um talão com informações sobre o produto selecionado, o valor inserido no pagamento e o troco.

O utilizador pode visualizar o stock da máquina através da autenticação.

3. Desenvolvimento

3.1. Considerações iniciais

- Cada produto tem um valor/preço fixo.
- A máquina de vendas automática tem duas categorias de produtos: snacks e bebidas.
- Cada categoria tem 3 produtos.
- Nas bebidas temos as opções Brisa, Coca-Cola, ambas com o custo de 1€ e água Luso, com um custo de 0,50€.
- Na categoria de Snacks temos Kinder Bueno, Kit Kat e Bounty, ambos com o custo de 1€.
- Na seleção e compra de um produto aparece um talão que apresenta o valor inserido e o valor a ser devolvido.
- A máquina de vendas só recebe os seguintes valores monetários: 0,10€, 0,20€, 0,50€, 1,00€ e 2,00€ e 5,00€.
- Os utilizadores podem verificar o stock através da autenticação.
- Para realizar a autenticação é necessária uma palavra-passe, esta tem pelo menos 4 carateres com certos requisitos tais como: um número, uma letra maiúscula, uma letra minúscula e um carater especial.
- A palavra-passe definida é **1Aa-**, e a sua representação em hexadecimal é 31 41 61 2D.



- O stock é composto por duas páginas, a primeira onde aparece a quantidade de moedas armazenadas e na segunda a quantidade de cada tipo de produto.

3.2. Periféricos

- ON_OFF: o seu objetivo é ligar ou desligar a máquina de vendas automáticas.
- PER_EN: opção do periférico que permite ao utilizador selecionar as opções que surgem no display. Caso o utilizador escolha uma opção que não exista, no display aparece uma mensagem de erro "Opção Errada" e volta para o menu anterior.
- OK: Este periférico confirma o estado do periférico de entrada PER_EN
- PASSWORD: O periférico password tem como função ler a palavra-passe para aceder o stock da máquina de vendas. A função VerificarPassword utiliza PASSWORD_I (posição do primeiro carater para escrever a palavra-passe), PASSWORD_F (posição do final carater possível para escrever a palavra-passe), PASSWORD_T (tamanho da palavra-passe correta) e VAZIO (representa um carater vazio) para comparar a palavra-passe introduzida com a palavra-passe correta. Se estiver certa o utilizador consegue aceder ao stock.

3.3. Funcionamento

O programa começa com o display e periféricos limpos. Para aceder a máquina de vendas automática é necessário ligar a máquina no periférico ON_OFF ao colocar '1'.

Após esta ação é apresentado o menu inicial da máquina, o utilizador pode escolher entre a seleção e compra de produtos ou aceder ao stock.

Para escolher uma destas opções é necessário colocar o valor da opção no periférico de entrada PER_EN, se colocar '1' acede aos produtos e se optar por '2' vai para o menu do stock.

Ao escolher a opção Produtos, o utilizador tem duas categorias, snacks e bebidas. Em cada categoria têm 3 produtos.

Ao selecionar um produto é apresentado um menu a confirmar a seleção do mesmo. Depois da confirmação é exibido os valores monetários que a máquina pode receber.

Na realização da compra do produto é emitido um talão com o valor inserido e o troco a devolver, caso seja necessário. Depois de realizar a compra o programa volta ao ecrã inicial.

Se optar pela opção '2' no menu inicial, será direcionado para o stock. Para aceder ao mesmo é necessário introduzir uma palavra-passe, esta ficou definida por '1Aa-'.

No stock aparece a quantidade de moedas e de produtos presentes na máquina.

3.4. Análise de Resultados

Na análise de resultados deveria ser apresentado as operações que realizadas na compra de um produto, dado que não foi implementado por completo não vai ser considerado na análise de resultados.

A autenticação do stock funciona corretamente, verificando se uma palavra-passe introduzida é igual a palavra-passe correta.

```
0080  2D 2D 2D 20 53 74 6F 63 6B 20 2D 2D 2D 2D 2D 2D   - - -  S t o c k  - - - - -
0090  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00A0  49 6E 73 69 72 61 20 50 61 73 73 77 6F 72 64 3A   I n s i r a  P a s s w o r d
00B0  31 41 61 2D 20 20 20 20 20 20 20 20 20 20 20 20   1 A a -
00C0  20 31 2D 20 43 6F 6E 66 69 72 6D 61 72 20 20 20     1 -  C o n f i r m a r
00D0  20 32 2D 20 56 6F 6C 74 61 72 20 20 20 20 20 20     2 -  V o l t a r
00E0  2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D   - - - - - - - - - - - - - - - -
```

1.PALAVRA-PASSE CORRETA

4. Conclusão

Concluindo, os objetivos do trabalho não foram alcançados devido a falta de tempo para realizar o mesmo. Os pontos que não foram implementados foi o talão e atualização do display depois de selecionar os produtos.

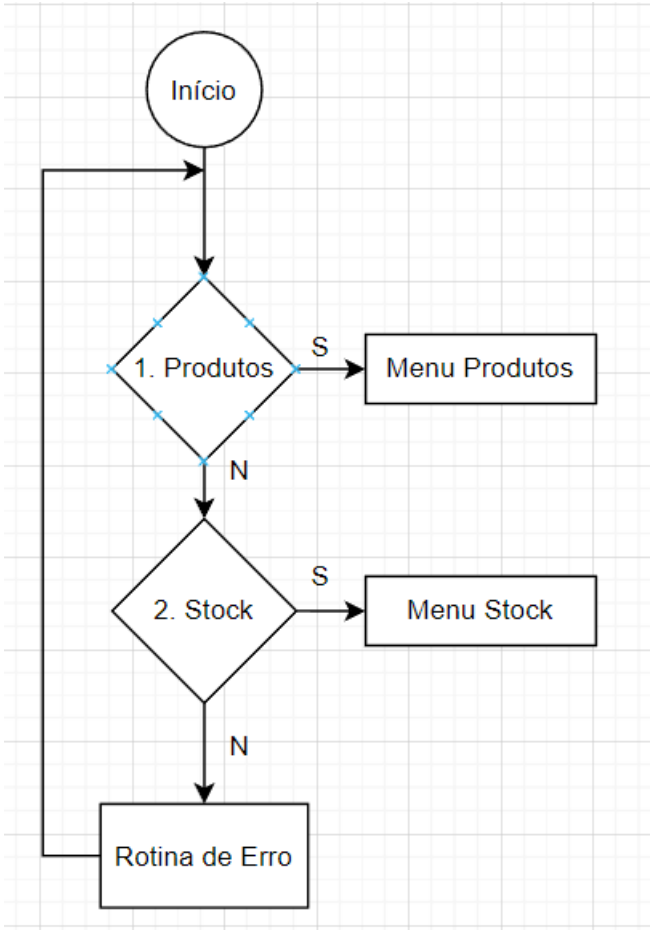
Desta forma e em geral, este trabalho foi um sucesso a nível de conhecimento de assembly e da cadeira de Arquitetura de Computadores.

5. Bibliografia

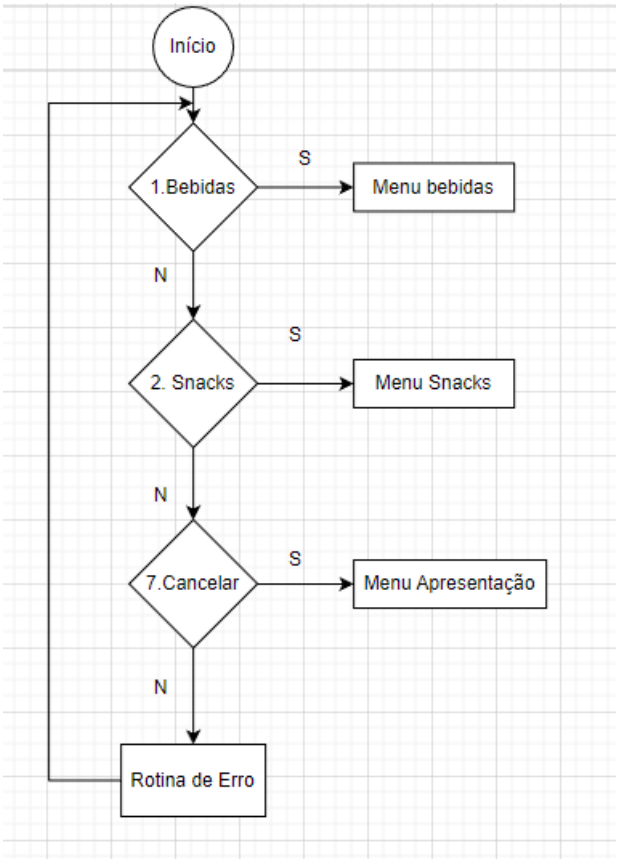
[1] J. Delgado e C. Ribeiro, Arquitetura de Computadores, FCA, 2014.

6. Anexo A - Fluxogramas

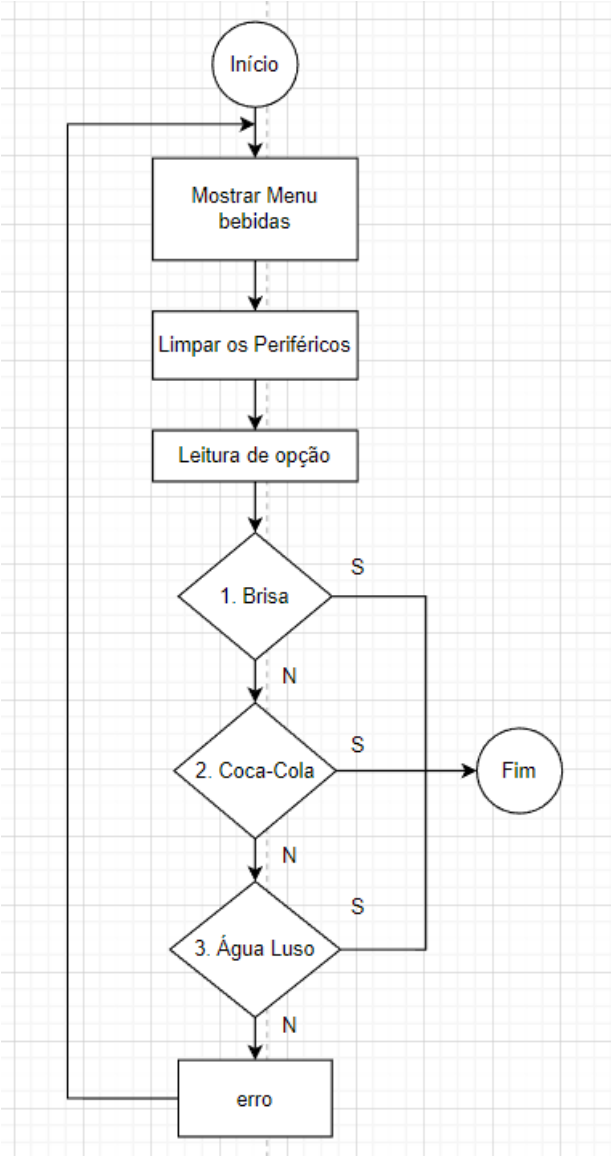
MENU – APRESENTAÇÃO



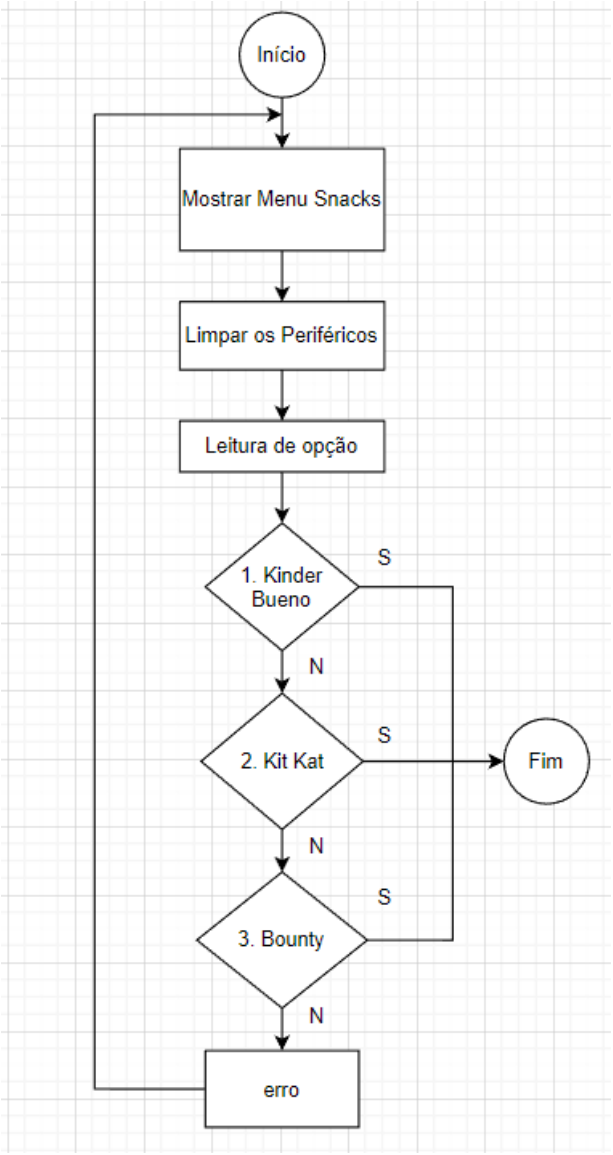
MENU – CATEGORIAS



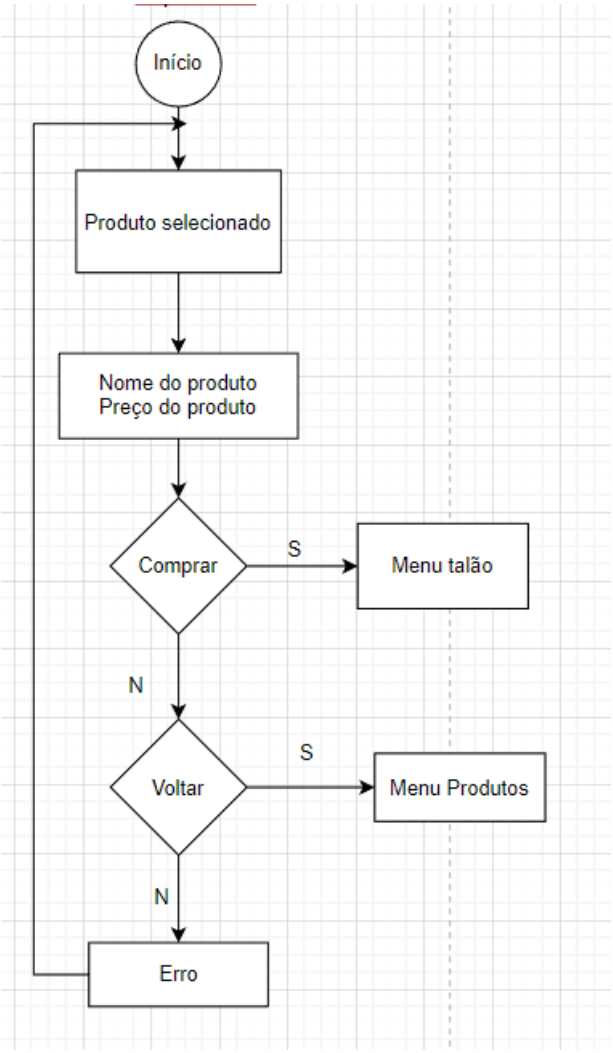
MENU – BEBIDAS



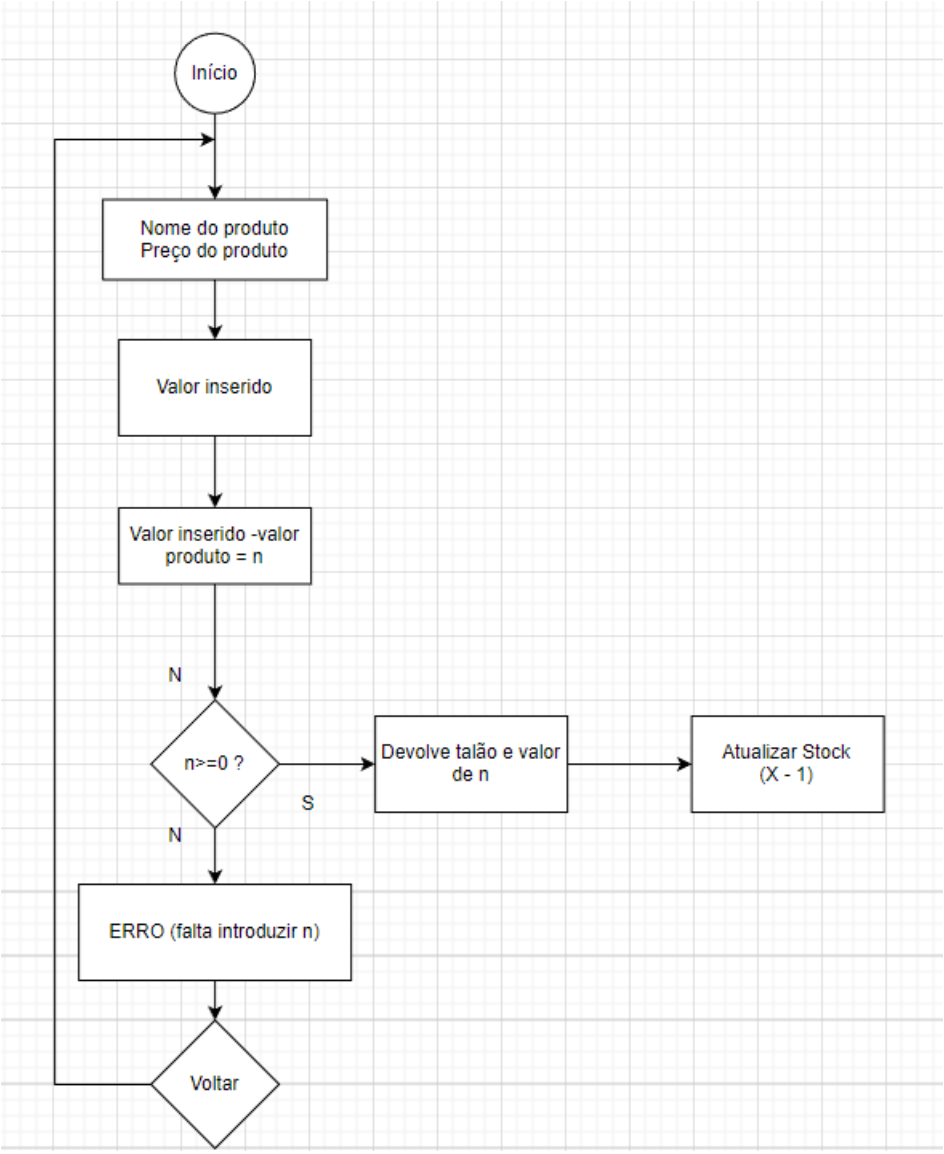
MENU – SNACKS


Início
Mostrar Menu Snacks
Limpar os Periféricos
Leitura de opção
1. Kinder Bueno
S
N
2. Kit Kat
S
Fim
N
3. Bounty
S
N
erro

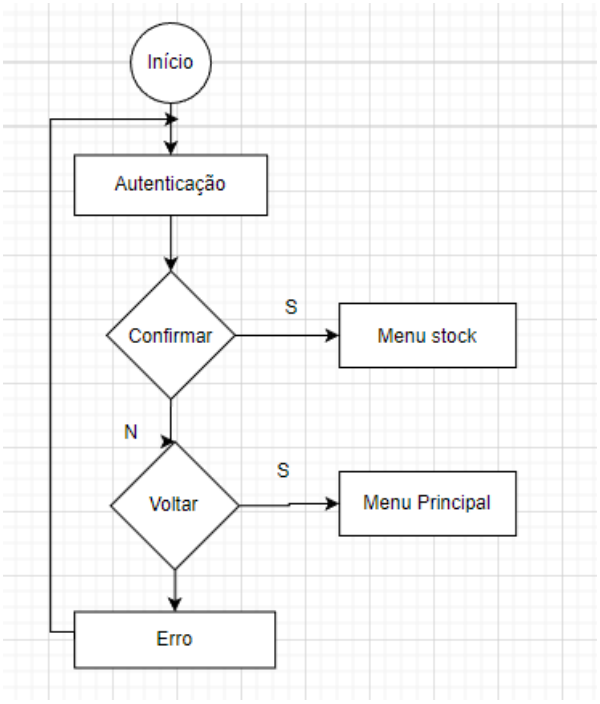
MENU - APRESENTAÇÃO DE PRODUTO



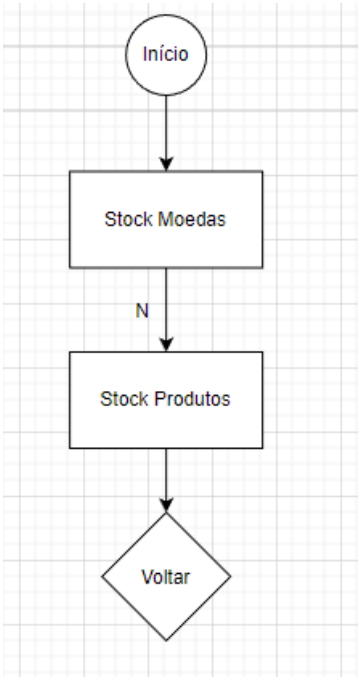
MENU – TALÃO


Início
Nome do produto
Preço do produto
Valor inserido
Valor inserido -valor produto = n
N
n>=0 ?
S
Devolve talão e valor de n
Atualizar Stock (X - 1)
N
ERRO (falta introduzir n)
Voltar

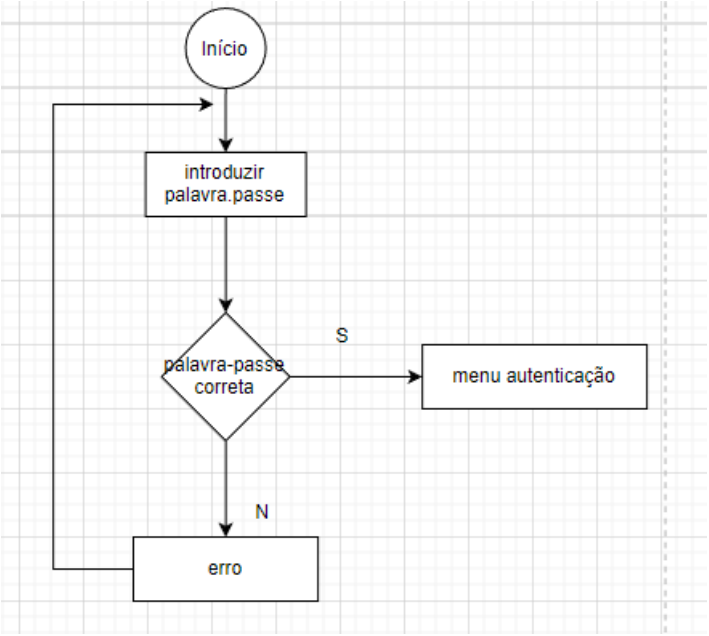
MENU – STOCK AUTENTICAÇÃO



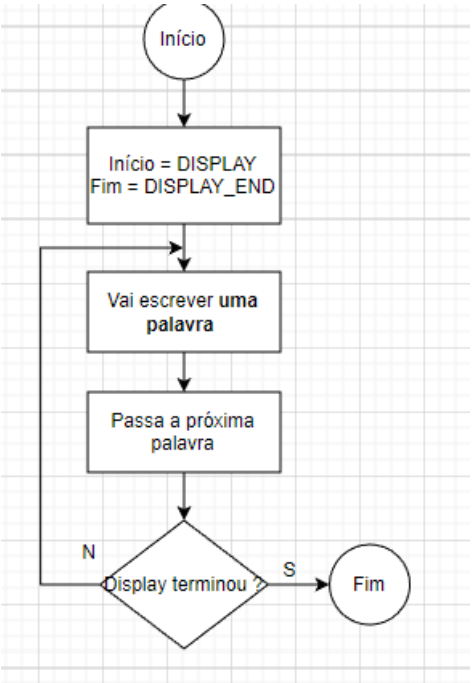
MENU – STOCK



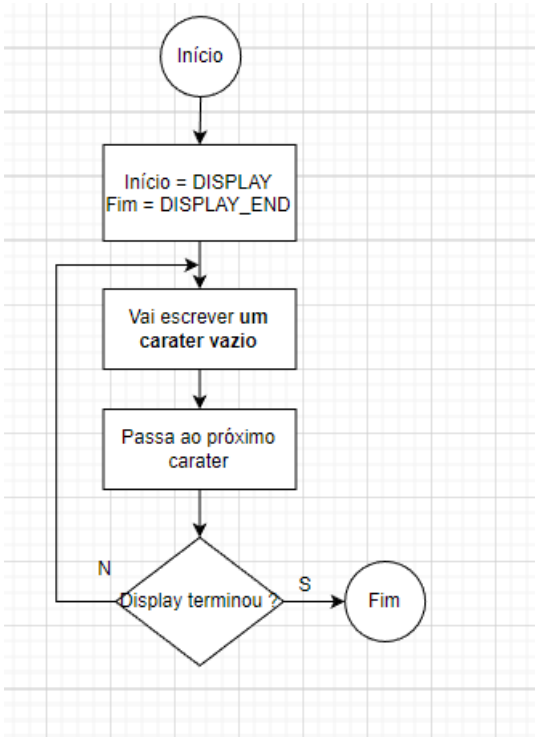
MENU – PALAVRA-PASSE



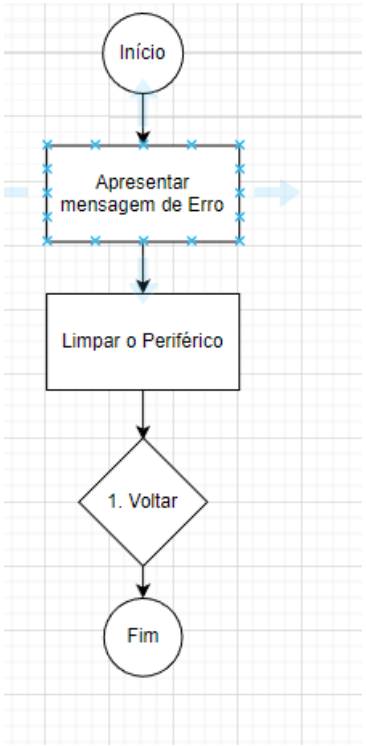
MENU – MOSTRAR DISPLAY



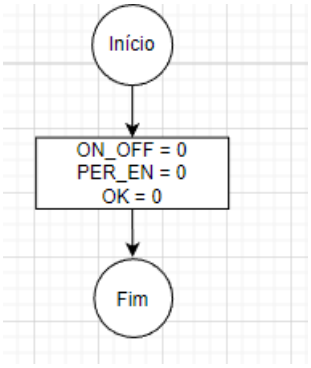
MENU – LIMPAR DISPLAY



MENU – ERRO



MENU – LIMPAR PERIFERICOS



7. Anexo B - Código

```
; P2 -> Arquitetura de Computadores
; MáquinaV - Máquina Automática de Vendas

PLACE 0000H                     ; 1º Endereço de Memória
Begin:
    JMP         Startup         ; Salto Incondicional para inicializar a Máquina
;_____________________________________________________________________________

PLACE 0070H
    ;STRING  "ASCII converter"
DISPLAY_NUM1    EQU 009CH       ; Display que representa um valor convertido em código ASCII
DISPLAY_NUM2    EQU 00ACH       ; Display que representa um valor convertido em código ASCII
DISPLAY_NUM3    EQU 00BCH       ; Display que representa um valor convertido em código ASCII
DISPLAY_NUM4    EQU 00CCH       ; Display que representa um valor convertido em código ASCII
DISPLAY_NUM5    EQU 00DCH       ; Display que representa um valor convertido em código ASCII
NR_TOT_CHAR     EQU 5           ; Número total de carateres
PASSWORD_I      EQU 00B0H       ; Inicio da palavra-passe
PASSWORD_F      EQU 00BFH       ; Fim da palavra-passe
VAZIO           EQU 0020H       ; Representa o carater vazio
PASSWORD_T      EQU 4           ; Tamanho da palavra-passe
;Periféricos de Entrada
;_____________________________________________________________________________
ON_OFF          EQU 0020H       ; Periférico de Entrada que permite Ligar / Desligar o Display
PER_EN          EQU 0021H       ; Periférico de Entrada
OK              EQU 0022H       ; Botão OK  -> Confirma o estado do Periférico de Entrada PER_EN
;PASSWORD       EQU 01D0H       ; Posição de Memória onde inserimos a nossa Password para Autenticar ao MenuStock
;_____________________________________________________________________________

;--------------------------
; O Ecrã Principal é composto por 7x16 (7 Linhas e 16 Carateres - Bytes)
DISPLAY         EQU 0080H       ; Inicio do Display
DISPLAY_END     EQU 00EFH       ; Fim do Display
EMPTY_BYTE      EQU 0020H       ; Carater Vazio (equivalente a 1 Byte) -> Limpa o Carater do Display
;--------------------------

DISPLAYSTR      EQU 00A0H       ; Endereço onde a STRING vai ser mostrada no DISPLAY (Bebida Brisa)
DISPLAYSTR1     EQU 00A0H       ; Endereço onde a STRING vai ser mostrada no DISPLAY (Bebida Coca-Cola)
DISPLAYSTR2     EQU 00A0H       ; Endereço onde a STRING vai ser mostrada no DISPLAY (Bebida Água Luso)

DISPLAYSTR3     EQU 00A0H       ; Endereço onde a STRING vai ser mostrada no DISPLAY (Snack Kinder Bueno)
DISPLAYSTR4     EQU 00A0H       ; Endereço onde a STRING vai ser mostrada no DISPLAY (Snack Kit-Kat)
DISPLAYSTR5     EQU 00A0H       ; Endereço onde a STRING vai ser mostrada no DISPLAY (Snack Bounty)
```

```
43
44  ;---------------------------
45  StackPointer    EQU 1FFEH       ; Endereço do Apontador de Stack
46  ;---------------------------
47
48  PLACE 0200H
49
50  ; Rotina para Limpar o Display
51  EraseDisplay:
52      PUSH R0                     ; É armazenado o valor de R0 na stack
53      PUSH R1                     ; É armazenado o valor de R1 na stack
54      PUSH R3                     ; É armazenado o valor de R3 na stack
55      MOV R0, DISPLAY             ; R0 guarda o endereço do DISPLAY
56      MOV R1, DISPLAY_END         ; R1 guarda o endereço do DISPLAY_END
57  Erase_Cycle:
58      MOV R2, EMPTY_BYTE          ; R2 guarda o endereço EMPTY_BYTE
59      MOVB [R0], R2               ; RTL: M[R0] <- R2
60      ADD R0, 1                   ; Incrementa uma unidade a R0
61      CMP R0, R1                  ; Compara o valor de R0 com o valor de R1
62      JLE Erase_Cycle             ; Salto Condicional para a etiqueta Erase_Cycle
63      ; POP -> repõe o valor de PUSH
64      POP R3                      ; É retirado o valor de R3 da stack
65      POP R1                      ; É retirado o valor de R1 da stack
66      POP R0                      ; É retirado o valor de R0 da stack
67  RET                             ; Retorno da Rotina EraseDisplay
68
69  ; Rotina para Limpar os Periféricos
70  ErasePeripherals:
71      PUSH R0                     ; É armazenado o valor de R0 na stack
72      PUSH R1                     ; É armazenado o valor de R1 na stack
73      PUSH R2                     ; É armazenado o valor de R2 na stack
74      MOV R0, ON_OFF              ; R0 guarda o valor de ON/OFF
75      MOV R1, PER_EN              ; R1 guarda o valor do Periférico de Entrada
76      MOV R2, OK                  ; R2 guarda o valor do Botão OK
77      MOV R3, 0                   ; RTL: R3 <- 0
78      MOVB [R0], R3               ; RTL: M[R0] <- R3
79      MOVB [R1], R3               ; RTL: M[R1] <- R3
80      MOVB [R2], R3               ; RTL: M[R2] <- R3
81      ; POP -> repõe o valor de PUSH
82      POP R2                      ; É retirado o valor de R2 da stack
83      POP R1                      ; É retirado o valor de R1 da stack
84      POP R0                      ; É retirado o valor de R0 da stack
```

```
RET                             ; Retorno da Rotina ErasePeripherals

; Rotina para Mostrar o Display
ShowDisplay:
    PUSH R0                     ; É armazenado o valor de R0 na stack
    PUSH R1                     ; É armazenado o valor de R1 na stack
    PUSH R3                     ; É armazenado o valor de R3 na stack
    MOV R0, DISPLAY             ; R0 guarda o endereço DISPLAY
    MOV R1, DISPLAY_END         ; R1 guarda o endereço DISPLAY_END
Cycle:                          ; Etiqueta que representa o Ciclo para Mostrar o Display
    MOV R3, [R2]                ; O endereço de Memória em R2 é guardado em R3
    MOV [R0], R3                ; RTL: [R0] <- R3
    ADD R2, 2                   ; Incrementa duas unidades em R2
    ADD R0, 2                   ; Incrementa duas unidades em R0
    CMP R0, R1                  ; Compara o valor de R0 com o valor de R1
    JLE Cycle                   ; Salto Condicional para a etiqueta Cycle
    ; POP -> repõe o valor de PUSH
    POP R3                      ; É retirado o valor de R3 na stack
    POP R1                      ; É retirado o valor de R1 na stack
    POP R0                      ; É retirado o valor de R0 na stack
RET                             ; Retorno da Rotina ShowDisplay

; Rotina de ERRO
RotinaERRO:
    PUSH R0                     ; É armazenado o valor de R0 na stack
    PUSH R1                     ; É armazenado o valor de R1 na stack
    PUSH R2                     ; É armazenado o valor de R2 na stack
    MOV R2, MenuERRO            ; R2 guarda o Endereço de MenuERRO
    CALL ShowDisplay            ; Chamada a Rotina para Mostrar o DISPLAY
    CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
    MOV R0, PER_EN              ; R0 guarda o valor do Periférico de Entrada
ERRO:
    MOVB R1, [R0]               ; RTL: R1 <- M[R0]
    CMP R1, 1                   ; Compara se R1 = 1
    JNE ERRO                    ; Salto Condicional para a etiqueta ERRO
    ; POP -> repõe o valor de PUSH
    POP R2                      ; É retirado o valor de R2 na stack
    POP R1                      ; É retirado o valor de R1 na stack
    POP R0                      ; É retirado o valor de R0 na stack
RET
```

```
;-------------------------------------------------
; Rotina para converter um valor em código ASCII
;-------------------------------------------------
converter:
    PUSH R0                 ; É armazenado o valor de R0 na stack
    ;PUSH R1                ; não realizamos PUSH em R1 como é o registo com o valor lido do P_IN
    PUSH R2                 ; É armazenado o valor de R2 na stack
    PUSH R3                 ; É armazenado o valor de R3 na stack
    PUSH R4                 ; É armazenado o valor de R4 na stack
    PUSH R5                 ; É armazenado o valor de R5 na stack

    MOV R0, 10
    MOV R2, DISPLAY_NUM1    ;R2 fica com a base do display
    ADD R2, 2               ;posição do carater a preencher
    MOV R3, 0               ;R3 tem o nº de carateres já preenchidos
proximoCarater:
    MOV R4, R1              ;R4 fica com uma cópia do valor lido de X
    MOD R4, R0              ;Calcula o D - resto da divisão inteira por 10
    DIV R1, R0              ;Atualiza X - quociente da divisão inteira por 10

    MOV R5, 48              ;Passamos para R5 o valor 48 devido a ser um valor "grande"
    ADD R5, R4              ;Calcula C = D + 48
    MOVB [R2], R5           ;escreve o carater - BYTE no display a preencher

    SUB R2, 1               ;próxima posição do display a preencher
    ADD R3, 1               ;incrementa o nº de carateres preenchidos

    CMP R1, 0               ; RTL: IF R1 = 0 ? (PER_EN = 0)
    JNE proximoCarater      ; Salto Jump if not Equal para a Etiqueta proximoCarater
cicloVazios:
    CMP R3, NR_TOT_CHAR     ; RTL: R3 <- guarda o endereço de NR_TOT_CHAR
    JEQ fimrotina           ; Salto Jump if equal para a Etiqueta fimrotina
    MOV R5, EMPTY_BYTE      ; RTL: R5 <- guarda o endereço de EMPTY_BYTE
    MOVB [R2], R5           ; RTL:
    SUB R2, 1               ;próxima posição do display a preencher
    ADD R3, 1               ;incrementa o nº de carateres preenchidos
    JMP cicloVazios
```

```
163
164 fimrotina:                     ;etiqueta que indica o fim da rotina
165     POP R5                      ; É retirado o valor de R5 da stack
166     POP R4                      ; É retirado o valor de R4 da stack
167     POP R3                      ; É retirado o valor de R3 da stack
168     POP R2                      ; É retirado o valor de R2 da stack
169     ;POP R1
170     POP R0                      ; É retirado o valor de R0 da stack
171 RET                             ;retorno da rotina
172
173 ; Programa Principal
174 ;-----------------------------  ; Iniciamos o Programa Principal no Endereço de Memória 3000
175 Startup:
176     MOV SP, StackPointer        ; Instrução de Transferência de Dados que Guarda o Endereço do Apontador da Pilha
177     CALL EraseDisplay           ; Chamada a Rotina para Limpar o Display
178     CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
179     MOV R0, ON_OFF              ; R0 guarda o valor de ON_OFF
180 SwitchON:
181     MOVB R1, [R0]               ; Leitura onde R1 guarda o endereço de memória de R0
182     CMP R1, 1                   ; Compara se ON_OFF = 1
183     JNE SwitchON                ; Salto Condicional para a etiqueta SwitchON
184 ON:
185     MOV R2, MenuPrincipal       ; RTL: R2 <- guarda o endereço do Menu Principal
186     CALL ShowDisplay            ; Chamada a Rotina para Mostrar o Display
187     CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
188 Read_Option:
189     MOV R0, PER_EN              ; R0 guarda o endereço corresponde à opção
190     MOVB R1, [R0]               ; R1 guarda o valor lido da respetiva opção escolhida
191     CMP R1, 0                   ; Compara se R1 = 0
192     JEQ Read_Option             ; Salto Condicional para a etiqueta Read_Option
193     ;-------------------
194     CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
195     JEQ OProdutos               ; Salto Condicional (Jump if equal) para a etiqueta OProdutos
196     CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
197     JEQ OStockAutenticar1       ; Salto Condicional (Jump if equal) para a etiqueta OStockAutenticar
198     ;-------------------
199     CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
200     JMP ON                      ; Salto Incondicional para a etiqueta ON
201
202 ; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
203 OStockAutenticar1:
204     JMP OStockAutenticar        ; Salto Incondicional para a etiqueta OStockAutenticar
```

```
205
206 ;-----------------------------
207 ; Rotina para o Menu Produtos
208 ;-----------------------------
209 OProdutos:
210     MOV R2, MenuProdutos        ; R2 guarda o endereço MenuProdutos
211     CALL ShowDisplay            ; Chamada a Rotina para Mostrar o Display
212     CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
213     MOV R0, PER_EN              ; Compara o valor do Periférico de Entrada com R0
214 Cycle_Produtos:
215     MOVB R1, [R0]               ; RTL: R1 <- M[R0]
216     CMP R1, 0                   ; Compara se R1 = 0
217     JEQ Cycle_Produtos          ; Salto Condicional para a etiqueta Cycle_Produtos
218     ;----------------------
219     CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
220     JEQ OBebidas                ; Salto Condicional para a etiqueta OBebidas
221     CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
222     JEQ OSnacks                 ; Salto Condicional para a etiqueta OSnacks
223     CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
224     JEQ ON                      ; Salto Condicional Jump if equal para a etiqueta ON -> retrocede para o Menu Principal
225     ;----------------------
226     CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
227     JMP OProdutos               ; Salto Incondicional para a etiqueta OProdutos
228
229 ; Rotina para o Menu das Bebidas
230 ;-------------------------------------------------
231 OBebidas:
232     MOV R2, MenuBebidas         ; R2 guarda o endereço MenuBebidas
233     CALL ShowDisplay            ; Chamada a Rotina para Mostrar o Ecrã
234     CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
235     MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
236 Cycle_Bebidas:                  ; Etiqueta que representa o Ciclo da Rotina OBebidas
237     MOVB R1, [R0]               ; RTL: R1 <- [R0]
238     CMP R1, 0                   ; Compara se R1 = 0
239     JEQ Cycle_Bebidas           ; Salto Condicional para a etiqueta Cycle_Bebidas
240     ;----------------------
241     CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
242     JEQ OBrisa                  ; Salto Condicional Jump if equal para a Opção Brisa
243     CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
244     JEQ OCocaCola               ; Salto Condicional Jump if equal para a Opção CocaCola
245     CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
246     JEQ OAguaLuso               ; Salto Condicional Jump fi equal para a Opção AguaLuso
```

```
    CMP R1, 4                   ; RTL: IF R1 = 4 ? (PER_EN = 4)
    JEQ OProdutos               ; Salto Condicional Jump if equal para retroceder ao Menu dos Produtos
    ;----------------------
    CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
    JMP OBebidas                ; Salto Incondicional para a etiqueta OBebidas

; Rotina para a Opção Brisa
OBrisa:
    MOV R2, MenuEscolhaBebida   ; R2 guarda o endereço do MenuEscolhaBebida
    CALL ShowDisplay            ; Chamada a Rotina para Mostrar o Display
    CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
    MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
Cycle_Brisa:
    MOVB R1, [R0]               ; RTL: R1 <- M[R0]
    CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
    JEQ Cycle_Brisa             ; Salto Condicional para a etiqueta Cycle_Brisa
    CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
    JEQ OPT_Pagamento1          ; Salto Condicional Jump if equal para a Opção de Pagamento
    CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
    JEQ OBebidas                ; Salto Condicional Jump if equal para retroceder ao Menu das Bebidas
    CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
    JEQ StringDisplayBrisa      ; Salto Jump if equal para a Etiqueta StringDisplayBrisa
    CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
    JMP OBrisa                  ; Salto Incondicional para a etiqueta OBrisa

; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
OPT_Pagamento1:
    JMP OPT_Pagamento           ; Salto Incondicional para a Etiqueta OPT_Pagamento

;TrocoBrisa:
    ;display do menu
    ;meter R6 em ascii na posição de memória inserido caso diferente de 0 substituir o menu default
    ;SUB R6, 100
    ;meter R6 em ascii na posição de memória troco
    ;ler periférico botão ok e voltar ao menu principal
    ;JMP TrocoBrisa

; Rotina para Apresentar a informação da bebida escolhida (Brisa)
;-------------------------------------------------------------------------
StringDisplayBrisa:
    MOV R6, TableBrisa          ; RTL: R6 <- guarda o endereço de origem TableBrisa
    MOV R7, DISPLAYSTR          ; RTL: R7 <- guarda o endereço de Destino DISPLAYSTR
```

```
    MOV R8, 32                  ; RTL: R8 <- guarda o número total de carateres da String
cyclestring:
    MOVB R9, [R6]               ; Guarda o Byte de menor peso de R6 em R9
    MOVB [R7], R9               ; RTL: M[R7] <- R9
    ADD R6, 1                   ; RTL: R6 <- R6 + 1 (Incrementa uma unidade em R6)
    ADD R7, 1                   ; RTL: R7 <- R7 + 1 (Incrementa uma unidade em R7)
    SUB R8, 1                   ; RTL: R8 <- R8 - 1 (Decrementa uma unidade em R8)
    CMP R8, 0                   ; RTL: IF R8 = 0 ? (PER_EN = 0)
    JNZ cyclestring             ; Salto Jump if not Zero para a etiqueta de ciclo cyclestring
fora:
    MOV R9, PER_EN              ; RTL: R9 <- guarda o valor do Periférico de Entrada PER_EN
    MOVB R10, [R9]              ; RTL: R10 <- M[R9]
    CMP R10, 0                  ; RTL: IF R10 = 0 ? (PER_EN = 0)
    JZ OPT_Pagamento1           ; Salto Jump if Zero para a opção OPT_Pagamento

    JMP fora                    ; Salto Incondicional para a Etiqueta fora

; Rotina para a Opção Coca-Cola
OCocaCola:
    ;MOV R6, 0
    ;MOV R7, 100
    MOV R2, MenuEscolhaBebida   ; R2 guarda o endereço do MenuEscolhaBebida
    CALL ShowDisplay            ; Chamada a Rotina para Mostrar o Display
    CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
    MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
Cycle_CocaCola:
    MOVB R1, [R0]               ; RTL: R1 <- [R0]
    CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
    JEQ Cycle_CocaCola          ; Salto Condicional para a etiqueta Cycle_Brisa
    CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
    JEQ OPT_Pagamento2          ; Salto Condicional Jump if equal para a Opção de Pagamento
    CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
    JEQ OBebidas                ; Salto Condicional Jump if equal para retroceder ao Menu das Bebidas
    CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
    JEQ StringDisplayCocaCola   ; Salto Jump if equal para a Etiqueta StringDisplayCocaCola
    CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
    JMP OCocaCola               ; Salto Incondicional para a etiqueta OCocaCola

;TrocoCocaCola:
    ; Display do Menu
    ; Colocar R6 em ASCII na posição de Memória inserido caso diferente de 0 substituir o menu default
    ;JMP TrocoCocaCola
```

```
331
332 ; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
333 OPT_Pagamento2:
334     JMP OPT_Pagamento           ; Salto Incondicional para a Etiqueta OPT_Pagamento
335
336 ; Rotina para Apresentar a informação da bebida escolhida (Coca-Cola)
337 ;---------------------------------------------------------------------
338 StringDisplayCocaCola:
339     MOV R6, TableCocaCola       ; RTL: R6 <- guarda o endereço de origem TableCocaCola
340     MOV R7, DISPLAYSTR1         ; RTL: R7 <- guarda o endereço de Destino DISPLAYSTR1
341     MOV R8, 32                  ; RTL: R8 <- guarda o número total de carateres da String
342 cyclestring1:
343     MOVB R9, [R6]               ; Guarda o Byte de menor peso de R6 em R9
344     MOVB [R7], R9               ; RTL: M[R7] <- R9
345     ADD R6, 1                   ; RTL: R6 <- R6 + 1 (Incrementa uma unidade em R6)
346     ADD R7, 1                   ; RTL: R7 <- R7 + 1 (Incrementa uma unidade em R7)
347     SUB R8, 1                   ; RTL: R8 <- R8 - 1 (Decrementa uma unidade em R8)
348     CMP R8, 0                   ; RTL: IF R8 = 0 ? (PER_EN = 0)
349     JNZ cyclestring1            ; Salto Jump if not Zero para a etiqueta de ciclo cyclestring
350 fora1:
351     MOV R9, PER_EN              ; RTL: R9 <- Guarda o valor do Periférico de Entrada PER_EN
352     MOVB R10, [R9]              ; RTL: R10 <- M[R9]
353     CMP R10, 0                  ; RTL: IF R10 = 0 ? (PER_EN = 0)
354     JZ OPT_Pagamento2           ; Salto Jump if Zero para a opção OPT_Pagamento
355
356     JMP fora1                   ; Salto Incondicional para a Etiqueta fora1
357
358 ; Rotina para a Opção Água Luso
359 OAguaLuso:
360     MOV R2, MenuEscolhaBebida   ; R2 guarda o endereço do MenuEscolhaBebida
361     CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
362     CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
363     MOV R0, PER_EN              ; R0 guarda o valor do Periférico de Entrada
364 Cycle_AguaLuso:
365     MOVB R1, [R0]               ; RTL: R1 <- [R0]
366     CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
367     JEQ Cycle_AguaLuso          ; Salto Condicional para a etiqueta Cycle_AguaLuso
368     CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
369     JEQ OPT_Pagamento3          ; Salto Condicional Jump if equal para a Opção de Pagamento
370     CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
371     JEQ OBebidas                ; Salto Condicional Jump if equal para retroceder ao Menu das Bebidas
```

```
372     CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
373     JEQ StringDisplayAguaLuso   ; Salto Jump if equal para a Etiqueta StringDisplayAguaLuso
374     CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
375     JMP OAguaLuso               ; Salto Incondicional para a etiqueta OAguaLuso
376
377 ; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
378 OPT_Pagamento3:
379     JMP OPT_Pagamento           ; Salto Incondicional para a Etiqueta OPT_Pagamento
380
381 ; Rotina para Apresentar a informação da bebida escolhida (Água Luso)
382 ;---------------------------------------------------------------------
383 StringDisplayAguaLuso:
384     MOV R6, TableLuso           ; RTL: R6 <- guarda o endereço de origem TableLuso
385     MOV R7, DISPLAYSTR2         ; RTL: R7 <- guarda o endereço de Destino DISPLAYSTR2
386     MOV R8, 32                  ; RTL: R8 <- guarda o número total de carateres da String
387 cyclestring2:
388     MOVB R9, [R6]               ; Guarda o Byte de menor peso de R6 em R9
389     MOVB [R7], R9               ; RTL: M[R7] <- R9
390     ADD R6, 1                   ; RTL: R6 <- R6 + 1 (Incrementa uma unidade em R6)
391     ADD R7, 1                   ; RTL: R7 <- R7 + 1 (Incrementa uma unidade em R7)
392     SUB R8, 1                   ; RTL: R8 <- R8 - 1 (Decrementa uma unidade em R8)
393     CMP R8, 0                   ; RTL: IF R8 = 0 ? (PER_EN = 0)
394     JNZ cyclestring2            ; Salto Jump if not Zero para a etiqueta de ciclo cyclestring
395 fora2:
396     MOV R9, PER_EN              ; RTL: R9 <- Guarda o valor do Periférico de Entrada PER_EN
397     MOVB R10, [R9]              ; RTL: R10 <- M[R9]
398     CMP R10, 0                  ; RTL: IF R10 = 0 ? (PER_EN = 0)
399     JZ OPT_Pagamento3           ; Salto Jump if Zero para a opção OPT_Pagamento
400
401     JMP fora2                   ; Salto Incondicional para a Etiqueta fora2
402
403 ;---------------------------------------------------
404 ; Rotina para o Menu da Categoria dos Produtos Snacks
405 ;---------------------------------------------------
406 OSnacks:
407     MOV R2, MenuSnacks          ; R2 guarda o endereço MenuSnacks
408     CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
409     CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
410     MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
```

```
Cycle_Snacks:                   ; Etiqueta que representa o Ciclo da Rotina OSnacks
    MOVB R1, [R0]               ; RTL: R1 <- M[R0]
    CMP R1, 0                   ; Compara se R1 = 0
    JEQ Cycle_Snacks            ; Salto Condicional para a etiqueta Cycle_Snacks
    ;---------------------
    CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
    JEQ OKinderBueno            ; Salto Condicional Jump if equal para a etiqueta OKinderBueno
    CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
    JEQ OKitKat                 ; Salto Condicional Jump if equal para a etiqueta OKitKat
    CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
    JEQ OBounty                 ; Salto Condicional Jump if equal para a etiqueta OBounty
    CMP R1, 4                   ; RTL: IF R1 = 4 ? (PER_EN = 4)
    JEQ OProdutos1              ; Salto Condicional Jump if equal para a etiqueta OProdutos
    ;---------------------
    CALL RotinaERRO             ; Chamada à Rotina de Erro caso a opção selecionada seja incorreta
    JMP OSnacks                 ; Salto Incondicional para a etiqueta OSnacks

; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
OProdutos1:
    JMP OProdutos               ; Salto Incondicional para a Etiqueta Produtos

; Rotina para a Opção Kinder Bueno
OKinderBueno:
    MOV R2, MenuEscolhaSnack    ; R2 guarda o endereço do MenuEscolhaSnack
    CALL ShowDisplay            ; Chamada a Rotina para Mostrar o Display
    CALL ErasePeripherals       ; Chamada a Rotina para Limpar os Periféricos
    MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
Cycle_KinderBueno:
    MOVB R1, [R0]               ; RTL: R1 <- M[R0]
    CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
    JEQ Cycle_KinderBueno       ; Salto Condicional para a etiqueta Cycle_KinderBueno
    CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
    JEQ OPT_Pagamento4          ; Salto Condicional Jump if equal para a Opção de Pagamento
    CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
    JEQ OSnacks                 ; Salto Condicional Jump if equal para retroceder ao Menu dos Snacks
    CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
    JEQ StringDisplayKinderBueno    ; Salto Jump if equal para a Etiqueta StringDisplayKinderBueno
    CALL RotinaERRO             ; Chamada à Rotina de Erro caso a opção selecionada seja incorreta
    JMP OKinderBueno            ; Salto Incondicional para a etiqueta OKinderBueno
```

```
; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
OPT_Pagamento4:
    JMP OPT_Pagamento           ; Salto Incondicional para a etiqueta OPT_Pagamento

; Rotina para Apresentar a informação do snack escolhido (Kinder Bueno)
;-------------------------------------------------------------------------
StringDisplayKinderBueno:
    MOV R6, TableKinderBueno    ; RTL: R6 <- guarda o endereço de origem TableKinderBueno
    MOV R7, DISPLAYSTR3         ; RTL: R7 <- guarda o endereço de Destino DISPLAYSTR
    MOV R8, 32                  ; RTL: R8 <- guarda o número total de carateres da String
cyclestring3:
    MOVB R9, [R6]               ; Guarda o Byte de menor peso de R6 em R9
    MOVB [R7], R9               ; RTL: M[R7] <- R9
    ADD R6, 1                   ; RTL: R6 <- R6 + 1 (Incrementa uma unidade em R6)
    ADD R7, 1                   ; RTL: R7 <- R7 + 1 (Incrementa uma unidade em R7)
    SUB R8, 1                   ; RTL: R8 <- R8 - 1 (Decrementa uma unidade em R8)
    CMP R8, 0                   ; RTL: IF R8 = 0 ? (PER_EN = 0)
    JNZ cyclestring3            ; Salto Jump if not Zero para a etiqueta de ciclo cyclestring
fora3:
    MOV R9, PER_EN              ; RTL: R9 <- guarda o valor do Periférico de Entrada PER_EN
    MOVB R10, [R9]              ; RTL: R10 <- M[R9]
    CMP R10, 0                  ; RTL: IF R10 = 0 ? (PER_EN = 0)
    JZ OPT_Pagamento4           ; Salto Jump if Zero para a opção OPT_Pagamento

    JMP fora3                   ; Salto Incondicional para a Etiqueta fora3

; Rotina para a Opção Kit-Kat
OKitKat:
    MOV R2, MenuEscolhaSnack    ; R2 guarda o endereço do MenuEscolhaSnack
    CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
    CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
    MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
Cycle_KitKat:
    MOVB R1, [R0]               ; RTL: R1 <- [R0]
    CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
    JEQ Cycle_KitKat            ; Salto Condicional para a etiqueta Cycle_KitKat
    CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
    JEQ OPT_Pagamento5          ; Salto Condicional Jump if equal para a Opção de Pagamento
    CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
    JEQ OSnacks                 ; Salto Condicional Jump if equal para retroceder ao Menu dos Snacks
    CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
    JEQ StringDisplayKitKat     ; Salto Jump if equal para a Etiqueta StringDisplayKitKat
```

```
493         CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
494         JMP OKitKat                 ; Salto Incondicional para a etiqueta OKitKat
495
496     ; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
497     OPT_Pagamento5:
498         JMP OPT_Pagamento           ; Salto Incondicional para a Etiqueta OPT_Pagamento
499
500     ; Rotina para Apresentar a informação do snack escolhido (KitKat)
501     ;-------------------------------------------------------------------------
502     StringDisplayKitKat:
503         MOV R6, TableKitKat         ; RTL: R6 <- guarda o endereço de origem TableKitKat
504         MOV R7, DISPLAYSTR4         ; RTL: R7 <- guarda o endereço de Destino DISPLAYSTR
505         MOV R8, 32                  ; RTL: R8 <- guarda o número total de carateres da String
506     cyclestring4:
507         MOVB R9, [R6]               ; Guarda o Byte de menor peso de R6 em R9
508         MOVB [R7], R9               ; RTL: M[R7] <- R9
509         ADD R6, 1                   ; RTL: R6 <- R6 + 1 (Incrementa uma unidade em R6)
510         ADD R7, 1                   ; RTL: R7 <- R7 + 1 (Incrementa uma unidade em R7)
511         SUB R8, 1                   ; RTL: R8 <- R8 - 1 (Decrementa uma unidade em R8)
512         CMP R8, 0                   ; RTL: IF R8 = 0 ? (PER_EN = 0)
513         JNZ cyclestring4            ; Salto Jump if not Zero para a etiqueta de ciclo cyclestring
514     fora4:
515         MOV R9, PER_EN              ; RTL: R9 <- guarda o valor do Periférico de Entrada PER_EN
516         MOVB R10, [R9]              ; RTL: R10 <- M[R9]
517         CMP R10, 0                  ; RTL: IF R10 = 0 ? (PER_EN = 0)
518         JZ OPT_Pagamento5           ; Salto Jump if Zero para a opção OPT_Pagamento
519
520         JMP fora4                   ; Salto Incondicional para a Etiqueta fora4
521
522     ; Rotina para a Opção Bounty
523     OBounty:
524         MOV R2, MenuEscolhaSnack    ; RTL: R2 <- guarda o endereço do MenuEscolhaSnack
525         CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
526         CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
527         MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
528     Cycle_Bounty:
529         MOVB R1, [R0]               ; RTL: R1 <- [R0]
530         CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
531         JEQ Cycle_Bounty            ; Salto Condicional para a etiqueta Cycle_Bounty
532         CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
533         JEQ OPT_Pagamento6          ; Salto Condicional Jump if equal para a Opção de Pagamento
534         CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
```

```
535         JEQ OSnacks                 ; Salto Condicional Jump if equal para retroceder ao Menu dos Snacks
536         CMP R1, 3                   ; RTL: IF R1 = 3 ? (PER_EN = 3)
537         JEQ StringDisplayBounty     ; Salto Jump if equal para a Etiqueta StringDisplayBounty
538         CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
539         JMP OBounty                 ; Salto Incondicional para a etiqueta OBounty
540
541     ; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
542     OPT_Pagamento6:
543         JMP OPT_Pagamento           ; Salto Incondicional para a Etiqueta OPT_Pagamento
544
545     ; Rotina para Apresentar a informação do snack escolhido (Bounty)
546     ;-------------------------------------------------------------------------
547     StringDisplayBounty:
548         MOV R6, TableBounty         ; RTL: R6 <- guarda o endereço de origem TableBounty
549         MOV R7, DISPLAYSTR5         ; RTL: R7 <- guarda o endereço de Destino DISPLAYSTR
550         MOV R8, 32                  ; RTL: R8 <- guarda o número total de carateres da String
551     cyclestring5:
552         MOVB R9, [R6]               ; Guarda o Byte de menor peso de R6 em R9
553         MOVB [R7], R9               ; RTL: M[R7] <- R9
554         ADD R6, 1                   ; RTL: R6 <- R6 + 1 (Incrementa uma unidade em R6)
555         ADD R7, 1                   ; RTL: R7 <- R7 + 1 (Incrementa uma unidade em R7)
556         SUB R8, 1                   ; RTL: R8 <- R8 - 1 (Decrementa uma unidade em R8)
557         CMP R8, 0                   ; RTL: IF R8 = 0 ? (PER_EN = 0)
558         JNZ cyclestring5            ; Salto Jump if not Zero para a etiqueta de ciclo cyclestring
559     fora5:
560         MOV R9, PER_EN              ; RTL: R9 <- guarda o valor do Periférico de Entrada PER_EN
561         MOVB R10, [R9]              ; RTL: R10 <- M[R9]
562         CMP R10, 0                  ; RTL: IF R10 = 0 ? (PER_EN = 0)
563         JZ OPT_Pagamento6           ; Salto Jump if Zero para a opção OPT_Pagamento
564
565         JMP fora5                   ; Salto Incondicional para a Etiqueta fora
566
567     ; Rotina de Pagamento
568     ;------------------------
569     OPT_Pagamento:
570         MOV R2, MenuPagamento       ; RTL: R2 <- guarda o endereço do MenuPagamento
571         CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
572         CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
573         MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
574     Cycle_Pagamento:
575         MOVB R1, [R0]               ; RTL: R1 <- M[R0]
576         CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
```

```
577     JEQ Cycle_Pagamento         ; Salto Condicional Jump if equal para a Etiqueta Cycle_Pagamento
578     CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
579     JEQ OPT_Talao               ; Salto Jump if equal para a opção OPT_Talao
580     CALL RotinaERRO             ; Chamada à Rotina de Erro caso a opção selecionada seja incorreta
581     JMP OPT_Pagamento           ; Salto Incondicional para a etiqueta OPT_Pagamento
582
583 ; Rotina para Mostrar o Talão da Compra efetuada
584 ;-----------------------------------------------
585 OPT_Talao:
586     MOV R2, MenuTalao           ; RTL: R2 <- guarda o endereço de MenuTalao
587     CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
588     CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
589     MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
590 Cycle_Talao:
591     MOVB R1, [R0]               ; RTL: R1 <- M[R0]
592     CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
593     JEQ Cycle_Talao             ; Salto Jump if equal para a opção de ciclo Cycle_Talao
594     CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
595     JEQ ON2                     ; Salto Jump if equal para a Etiqueta ON2
596     CALL RotinaERRO             ; Chamada à Rotina de Erro caso a opção selecionada seja incorreta
597     JMP OPT_Talao               ; Salto Incondicional para a etiqueta OPT_Talao
598 ON2:                            ; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
599     JMP ON                      ; Salto Incondicional para a etiqueta ON onde vai retornar para o Menu Principal da Máquina
600
601 ;-----------------------------------------
602 ; Rotina para o Menu de Autenticação do Stock
603 ;-----------------------------------------
604 OStockAutenticar:
605     MOV R2, MenuStockAutenticar ; RTL: R2 <- guarda o endereço de MenuStockAutenticar
606     CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
607     CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
608     MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
609 Cycle_StockAutenticar:
610     MOVB R1, [R0]               ; RTL: R1 <- M[R0]
611     CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
612     JEQ Cycle_StockAutenticar   ; Salto Jump if equal para a etiqueta Cycle_StockAutenticar
613     ;----------------------------
614     CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
615     JEQ VerificarPassword       ; confirmação password
616     JEQ OStockMoedas            ; Salto Jump if equal para a etiqueta OStockMoedas
617     CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
618     JEQ ON1                     ; Salto Jump if equal para a Etiqueta ON1
```

```
619     ;----------------------------
620     CALL RotinaERRO             ; Chamada a Rotina de Erro caso a opção selecionada seja incorreta
621     JMP OStockAutenticar        ; Salto Incondicional para a etiqueta OStockAutenticar
622 ON1:                            ; executamos este salto pois o PEPE possui uma limitação apenas conseguindo saltar de 128 a 128 instruções
623     JMP ON                      ; Salto Incondicional para a etiqueta ON onde vai retornar para o Menu Principal da Máquina
624
625 ; Rotina para Verificar Password inserida
626 ;---------------------------------------
627 VerificarPassword:
628     MOV R1, PASSWORD_I          ; Guarda a posição inicial da palavra-passe
629     MOV R2, PASSWORD_F          ; Guarda a posição final da palavra-passe
630     MOV R3, Password            ; Guarda o valor da password do registo
631     MOV R6, PASSWORD_T          ; Guarda o tamanho  da palavra-passe
632 CicloDeComparacao:
633     MOVB R4, [R1]               ; Guarda o byte de menor peso de R1 em R4
634     MOVB R5, [R3]               ; Guarda o byte de menor peso de R3 em R5
635     CMP R4, R5                  ; Compara se o início da palavra-passe está vazio
636     JNE PasswordIncorreta       ; Salto condicional para a label PasswordIncorreta
637     ADD R1, 1                   ; RTL: R1 <- R1 + 1
638     SUB R6, 1                   ; RTL: R6 <- R6 - 1
639     JZ Confirmacao              ; Salto Condicional Jump if Zero para confirmar a inserção da Password
640     ADD R3, 1                   ; RTL: R3 <- R3 + 1
641     JMP CicloDeComparacao       ; Salto incondicional para a label CicloDeComparacao
642
643 PasswordIncorreta:
644     MOV R2,MenuStockPasswordIncorreta   ; RTL: R2 <- guarda o endereço de MenuStockPasswordIncorreta
645     CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
646     CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
647 TentarNovamente:
648     MOV R0, PER_EN              ; R0 guarda o endereço corresponde à opção
649     MOVB R1, [R0]               ; R1 guarda o valor lido da respetiva opção escolhida
650     CMP R1, 1                   ; Compara se R1 = 1
651     JEQ OStockAutenticar        ; Salto Jump if equal para a etiqueta OStockAutenticar
652     JMP TentarNovamente         ; Salto Jump if not equal para retroceder ao menu de Autenticação
653 Confirmacao:
654     MOVB R4, [R1]               ; Guarda o byte de menor peso de R1 em R4
655     MOV R2,VAZIO                ; RTL: R2 <- guarda o endereço VAZIO
656     CMP R2, R4                  ; RTL: IF R2 = R4 ? (Compara o valor de R2 com R4)
657     JNE PasswordIncorreta       ; Salto Jump if not equal caso a Password inserida seja incorreta
658
659 ; Rotina para Mostrar o Stock das Moedas
660 ;---------------------------------------
```

```
OStockMoedas:
    MOV R2, MenuStockMoedas     ; RTL: R2 <- guarda o endereço de MenuStockMoedas
    CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
    CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
    CALL converter              ; Chamada à Rotina para converter um valor em código ASCII
    MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
CycleStockMoedas:
    MOVB R1, [R0]               ; RTL: R1 <- M[R0]
    CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
    JEQ CycleStockMoedas        ; Salto Jump if equal para a etiqueta CycleStockMoedas
    CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
    JEQ OPT_StockProdutos       ; Salto Jump if equal para a etiqueta OPT_StockProdutos
    CMP R1, 2                   ; RTL: IF R1 = 2 ? (PER_EN = 2)
    JEQ OStockAutenticar        ; Salto Jump if equal para a etiqueta OStockAutenticar
    CALL RotinaERRO             ; Chamada à Rotina de ERRO caso a opção selecionada no PER_EN seja incorreta
    JMP OStockMoedas            ; Salto Incondicional para a etiqueta OStockMoedas

; Rotina para Mostrar o Stock dos Produtos
;--------------------------------------
OPT_StockProdutos:
    MOV R2, MenuStockProdutos   ; RTL: R2 <- guarda o endereço de MenuStockProdutos
    CALL ShowDisplay            ; Chamada à Rotina para Mostrar o Ecrã
    CALL ErasePeripherals       ; Chamada à Rotina para Limpar os Periféricos
    CALL converter              ; Chamada à Rotina para converter um valor em código ASCII
    MOV R0, PER_EN              ; RTL: R0 <- guarda o valor do Periférico de Seleção PER_EN
CycleStockProdutos:
    MOVB R1, [R0]               ; RTL: R1 <- M[R0]
    CMP R1, 0                   ; RTL: IF R1 = 0 ? (PER_EN = 0)
    JEQ CycleStockProdutos      ; Salto Jump if equal para a etiqueta CycleStockProdutos
    CMP R1, 1                   ; RTL: IF R1 = 1 ? (PER_EN = 1)
    JEQ OStockMoedas            ; Salto Jump if equal para a etiqueta OStockMoedas
    CALL RotinaERRO             ; Chamada à Rotina de ERRO caso a opção selecionada no PER_EN seja incorreta
    JMP OPT_StockProdutos       ; Salto Incondicional para a etiqueta OPT_StockProdutos
```

```
; Ecras

;password
;DB
PLACE 1000H
    ; Bebidas
    TableBrisa:
        STRING  " Opcao: Brisa   "
        STRING  " Preco: 1.00    "

PLACE 1030H
    TableCocaCola:
        STRING  " Opcao:Coca-Cola"
        STRING  " Preco: 1.00    "

PLACE 1060H
    TableLuso:
        STRING  " Opcao: Luso    "
        STRING  " Preco: 0.50    "

PLACE 1090H
    ; Snacks
    TableKinderBueno:
        STRING  " Opcao: KBueno  "
        STRING  " Preco: 1.00    "

PLACE 1120H
    TableKitKat:
        STRING  " Opcao: Kit-Kat "
        STRING  " Preco: 1.00    "

PLACE 1150H
    TableBounty:
        STRING  " Opcao: Bounty  "
        STRING  " Preco: 1.00    "
```

```
730
731 ; MENUS
732 PLACE 2000H
733
734     MenuPrincipal:
735         STRING  " -------------- "
736         STRING  " MAQUINA MADEIRA"
737         STRING  " BEM-VINDO      "
738         STRING  " -------------- "
739         STRING  " 1)Produtos     "
740         STRING  " 2)Stock        "
741         STRING  " ---------------"
742
743     MenuProdutos:
744         STRING  " -------------- "
745         STRING  " Produtos       "
746         STRING  " -------------- "
747         STRING  " 1)Bebidas      "
748         STRING  " 2)Snacks       "
749         STRING  " 7)Cancelar     "
750         STRING  " -------------- "
751
752     MenuERRO:
753         STRING  " -- ATENCAO --- "
754         STRING  " -------------- "
755         STRING  "    OPCAO       "
756         STRING  "    ERRADA      "
757         STRING  " -------------- "
758         STRING  "Tente  Novamente"
759         STRING  " -------------- "
760
761     MenuBebidas:
762         STRING  " Bebidas:       "
763         STRING  " -------------- "
764         STRING  " 1)Brisa        "
765         STRING  " 2)Coca-Cola    "
766         STRING  " 3)Agua Luso    "
767         STRING  " 7)Voltar       "
768         STRING  " ---------------"
769
```

```
769
770     MenuEscolhaBebida:
771         STRING  "Bebida escolhida"
772         STRING  " ------------- "
773         STRING  "               "  ; Aparece a informação da tabela para a respetiva bebida escolhida
774         STRING  "               "
775         STRING  " 1- Confirmar  "
776         STRING  " 2- Voltar     "
777         STRING  " --------------"
778
779     MenuSnacks:
780         STRING  " Snacks:       "
781         STRING  " ------------- "
782         STRING  " 1)Kinder Bueno "
783         STRING  " 2)Kit-Kat     "
784         STRING  " 3)Bounty      "
785         STRING  " 7)Voltar      "
786         STRING  " ------------- "
787
788     MenuEscolhaSnack:
789         STRING  "Snack escolhido:"
790         STRING  " ------------- "
791         STRING  "               "  ; Aparece a informação da tabela para o respetivo snack escolhido
792         STRING  "               "
793         STRING  " 1- Confirmar  "
794         STRING  " 2- Voltar     "
795         STRING  " --------------"
796
797     MenuPagamento:
798         STRING  " Val.Monetarios "
799         STRING  " 0.10          "
800         STRING  " 0.20          "
801         STRING  " 0.50          "
802         STRING  " 1.00          "
803         STRING  " 2.00          "
804         STRING  " 5.00          "
805
```

```
805
806     MenuTalao:
807         STRING  "----------------"
808         STRING  "     TALAO      "
809         STRING  "----------------"
810         STRING  "                "  ; Inserir nome do Produto
811         STRING  "Inserido:       "  ; Valor inserido
812         STRING  "Troco:          "  ; Inserir Troco (Valor Inserido - Preço Produto)
813         STRING  "----------------"
814
815 ;---------------------------------
816     MenuStockAutenticar:
817         STRING  "--- Stock ------"
818         STRING  "                "
819         STRING  "Insira Password:"
820         STRING  "                "  ; 00B0H -> Inserir Password em código ASCII
821         STRING  " 1- Confirmar   "
822         STRING  " 2- Voltar      "
823         STRING  "----------------"
824
825     MenuStockPasswordIncorreta:
826         STRING  " ---- Stock --- "
827         STRING  " -------------- "
828         STRING  "    Password    "
829         STRING  "   Incorreta!   "
830         STRING  " -------------- "
831         STRING  " 1- Voltar      "
832         STRING  " -------------- "
833
834     MenuStockMoedas:
835         STRING  "Stock-Moedas 1/2"
836         STRING  " 0.10:          "
837         STRING  " 0.20:          "
838         STRING  " 0.50:          "
839         STRING  " 1.00:          "
840         STRING  " 2.00:          "
841         STRING  " 5.00:          "
842
```

```
842
843     MenuStockProdutos:
844         STRING  "  Produtos   2/2"
845         STRING  " Brisa:          "
846         STRING  " CocaCola:       "
847         STRING  " Luso:           "
848         STRING  " KBueno:         "
849         STRING  " Kit-Kat:        "
850         STRING  " Bounty:         "
851
852     Password:
853         STRING "1Aa-"   ; Palavra-passe - hexa 31 41 61 2D
```