UNIVERSIDADE da MADEIRA

Relatório de Arquitetura de Computadores

Segundo projeto prático

Professores :

Dionísio Barros
Sofia Inácio
Dino Vasconcelos
Pedro Camacho

Trabalho realizado por :

Paulo Alexandre Rodrigues Alves Nº2120722
Renato Gabriel Silva Pêssego Nº2121922

Funchal, abril de 2024

Índice

1.Introdução..... 3
2.Objetivos..... 3
3.Desenvolvimento.....3
1.Periféricos de entrada..... 3
- ON_OFF:..... 3
- SelecionarOpcao:..... 3
- Password:..... 3
- Introduzir_PEPE:..... 4

2.Funcionamento da máquina de metro:.....4
- Comprar viagens:..... 4
- Usar cartão:..... 4

4.Discussão de resultados..... 5
5.Conclusão..... 5
6.Bibliografia..... 5
7.Lista de Figuras.....6
- Figura 1 - Menu inicio.....6
- Figura 2 - PEPE gerado..... 6
- Figura 3 - Menu comprar.....6
- Figura 4 - Menu valor a pagar..... 6
- Figura 5 - Menu dinheiro que foi inserido.....7
- Figura 6 - Menu inserir dinheiro..... 7
- Figura 7 - Menu troco a devolver..... 7
- Figura 8 - Menu consultar PEPE.....7
- Figura 9 - Menu saldo do PEPE.....8
- Figura 10 - Introdução da password.....8
- Figura 11 - Menu stock.....8

8.Anexo A.....9
10.Anexo B.....13

1.Introdução

Este projeto, desenvolvido no âmbito da cadeira de Arquitetura de computadores, permitiu-nos simular uma máquina de estação de metro. O utilizador é capaz de maneira fluida e fácil comprar bilhetes PEPE, sendo também capaz de o recarregar. O administrador é capaz de gerir o stock de moedas. A implementação deste projeto é realizada utilizando a linguagem Assembly em um simulador desenvolvido em Java disponibilizado pelos docentes.

Ao longo deste relatório iremos expor as dificuldades que tivemos a realizar o projeto e de como os superamos.

2.Objetivos

Este relatório tem como objetivo detalhar como foi implementado o segundo projeto de Arquitetura de computadores. Procura abordar tanto as funcionalidades básicas, como as funcionalidades extras como a elaboração dos fluxogramas. A gestão do stock das moedas e notas a serem devolvidas ao utilizador, o botão de ligar e desligar a máquina, a introdução de notas de 10 e 20 euros,introdução de password e valores monetários diferentes para cada estação de metro.

3.Desenvolvimento

1.Periféricos de entrada

ON_OFF:

Este botão é responsável por ligar a nossa máquina de metro. Quando o seu valor está a ‘1’ é atualizado o display da máquina para o menu inicial.

SelecionarOpcao:

É um dos periféricos mais importantes, pois permite manipular quase todas as operações do projeto. O utilizador é capaz de selecionar a opção desejada desde que esta esteja disponível no display, com alguma indicação, como continuar, voltar para o menu principal, cancelar, entre outras.Permite também escolher de um menu predefinido o dinheiro que o utilizador pretende inserir na máquina.

Password:

Este periférico proporciona ao administrador acesso ao stock de moedas da máquina.

Introduzir_PEPE:

Este periférico permite ao utilizador recarregar o seu bilhete PEPE ou até comprar bilhetes para outras estações no mesmo bilhete PEPE.

2.Funcionamento da máquina de metro:

Durante este programa, iremos recorrer a várias funções auxiliares tais como, *LimparDisplay*, *Limparperifericos*, *LimparTotal*, *LimparTotalInserido*, *LimparMoedasInseridas* e outras funções semelhantes, que servem para limpar valores que podem influenciar no decorrer do programa.

Inicialmente o display encontra-se em branco. Para aparecer o menu inicial [Figura 1] basta-nos colocar '1' no endereço de memória da etiqueta ON_OFF. Aparecendo este display seguidamente.

Agora, para selecionar uma destas três opções, manipulamos a posição de memória da etiqueta SelecionarOpcao.Obtendo para a opção um:

Comprar viagens:

Ao comprar viagens é gerado um PEPE imediatamente [Figura 2].O utilizador seguidamente tem de escolher uma ou várias opções de quais estações quer ir [Figura 3].Ao finalizar esse processo é pedido para o utilizador efetuar o pagamento, aparecendo o respectivo valor a pagar [Figura 4].Para adicionar dinheiro para comprar as viagens é necessário que durante a compra seja adicionado no PEN_EN a respetiva nota que o utilizador pretende inserir, [Figura 6] sendo este avisado da moeda/nota que inseriu [Figura 5] e o valor que inseriu.Ao finalizar a inserção de dinheiro é calculado o respectivo troco sendo apresentado da seguinte maneira [Figura 7]

Usar cartão:

É necessário introduzir o PEPE que foi gerado no comprar [Figura 8]. Por exemplo, foi gerado anteriormente o PEPE 8000, então no periférico de Introduzir_PEPE colocar na primeira posição 80 e na segunda 0.Ao inserir o PEPE gerado é mostrado ao utilizador o respectivo saldo desse bilhete.[Figura 9]

Stock:

Para aceder ao stock é necessário o utilizador, ser administrador a fim de saber a password para aceder ao stock.Que neste caso é "PaRp" [Figura 10].Dentro do stock é possível ver a quantidade de moedas que a máquina possui. [Figura 11]

4.Discussão de resultados

Podemos concluir que implementamos a maioria das funcionalidades que nos foram propostas.Tanto as obrigatórias, tanto as extras.

Nas extras tal como a elaboração dos fluxogramas. A gestão do stock das moedas e notas a serem devolvidas ao utilizador.

O botão de ligar e desligar a máquina, a introdução de notas de 10 e 20 euros,introdução de password, valores monetários diferentes para cada estação de metro.

5.Conclusão

Consideramos que atingimos os resultados exigidos para este segundo projeto prático.O que nos facilitou este projeto foi a realização de fluxogramas para descrever o comportamento de diversas rotinas.

Onde sentimos mais dificuldade foi na gestão de registos disponíveis, pois algumas rotinas exigiam manipulação e cópias entre diferentes registos.Outro ponto irritante foi que, por vezes, ocorria um erro de "constant out of bounds" quando um JMP estava muito distante, sendo necessário utilizar um JMP intermediário..

Concluímos que a linguagem Assembly é uma linguagem complexa sendo mais difícil que uma linguagem de alto nível mas fundamental para perceber a manipulação de registos e de posições de memória, logo uma melhor compreensão do funcionamento interno de um computador.

6.Bibliografia

Guia Componente de Avaliação P2 de Arquitetura de Computadores
J. Delgado e C. Ribeiro, Arquitectura de Computadores, FCA, 2007
D. M. Harris and S. L. Harris, Digital Design and Computer Architecture, Elsevier MK, 2007

7.Lista de Figuras

Figura 1 - Menu inicio

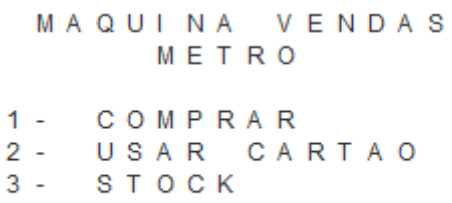


Figura 2 - PEPE gerado

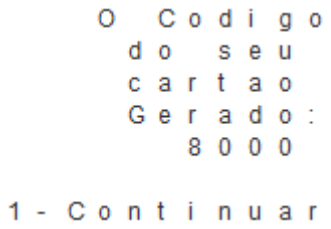


Figura 3 - Menu comprar

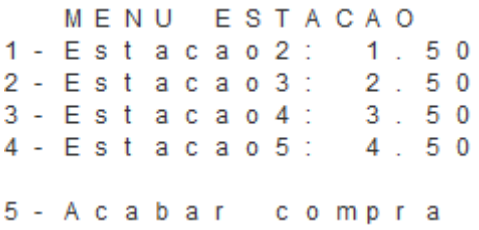


Figura 4 - Menu valor a pagar

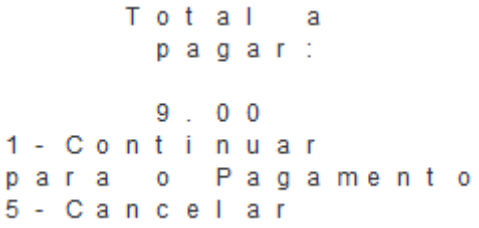


Figura 5 - Menu dinheiro que foi inserido

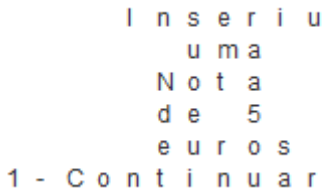


Figura 6 - Menu inserir dinheiro

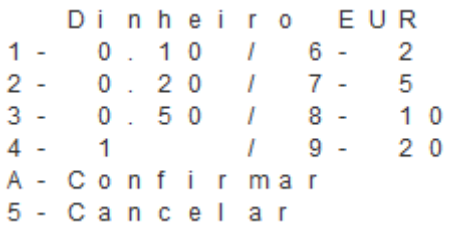


Figura 7 - Menu troco a devolver

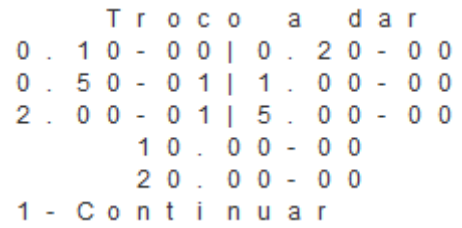


Figura 8 - Menu consultar PEPE

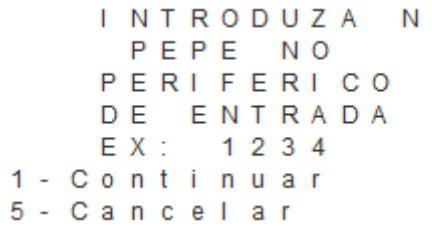


Figura 9 - Menu saldo do PEPE

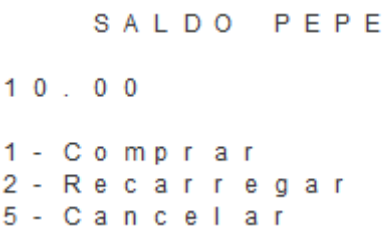


Figura 10 - Introdução da password

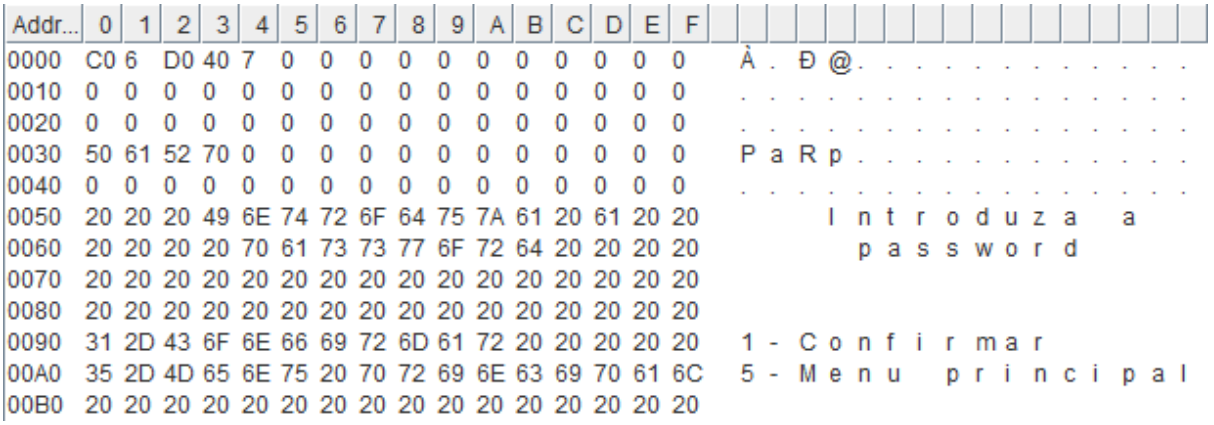
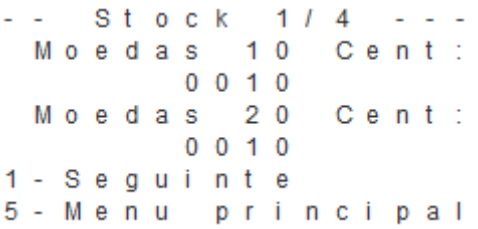
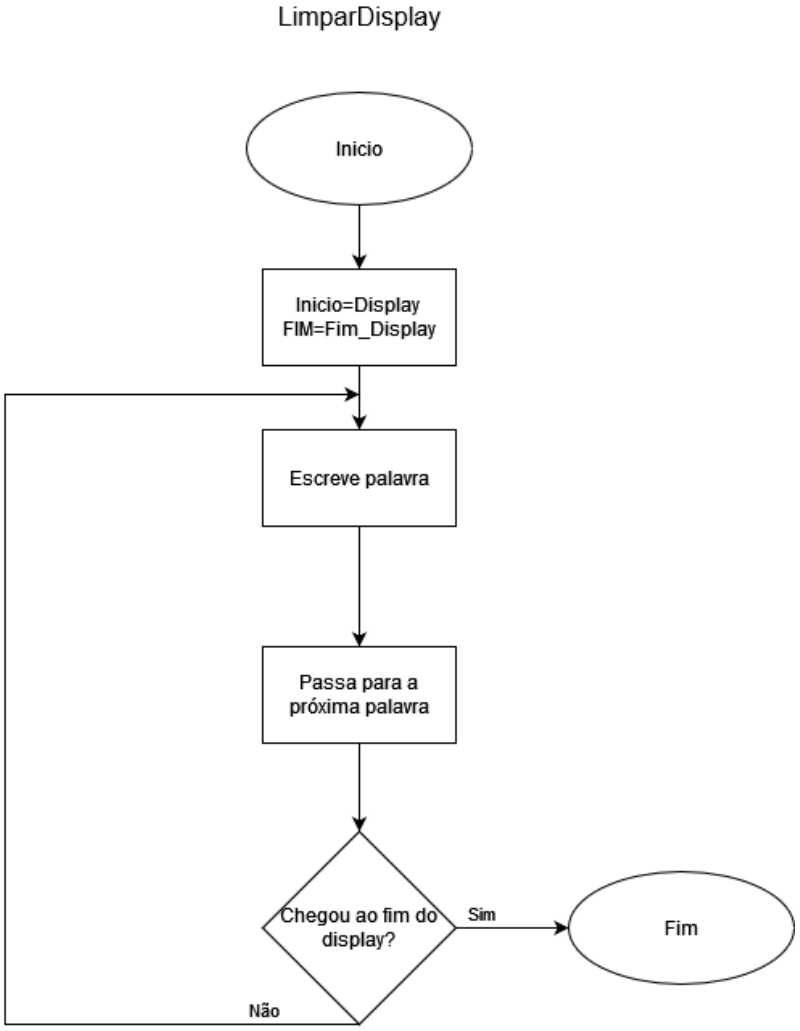


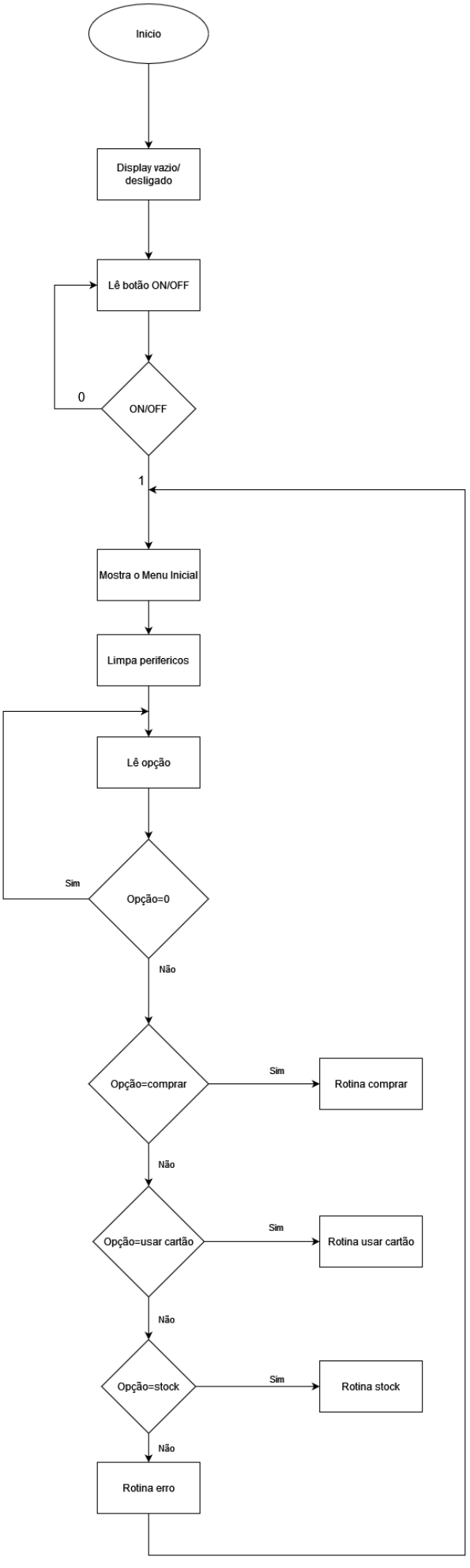
Figura 11 - Menu stock



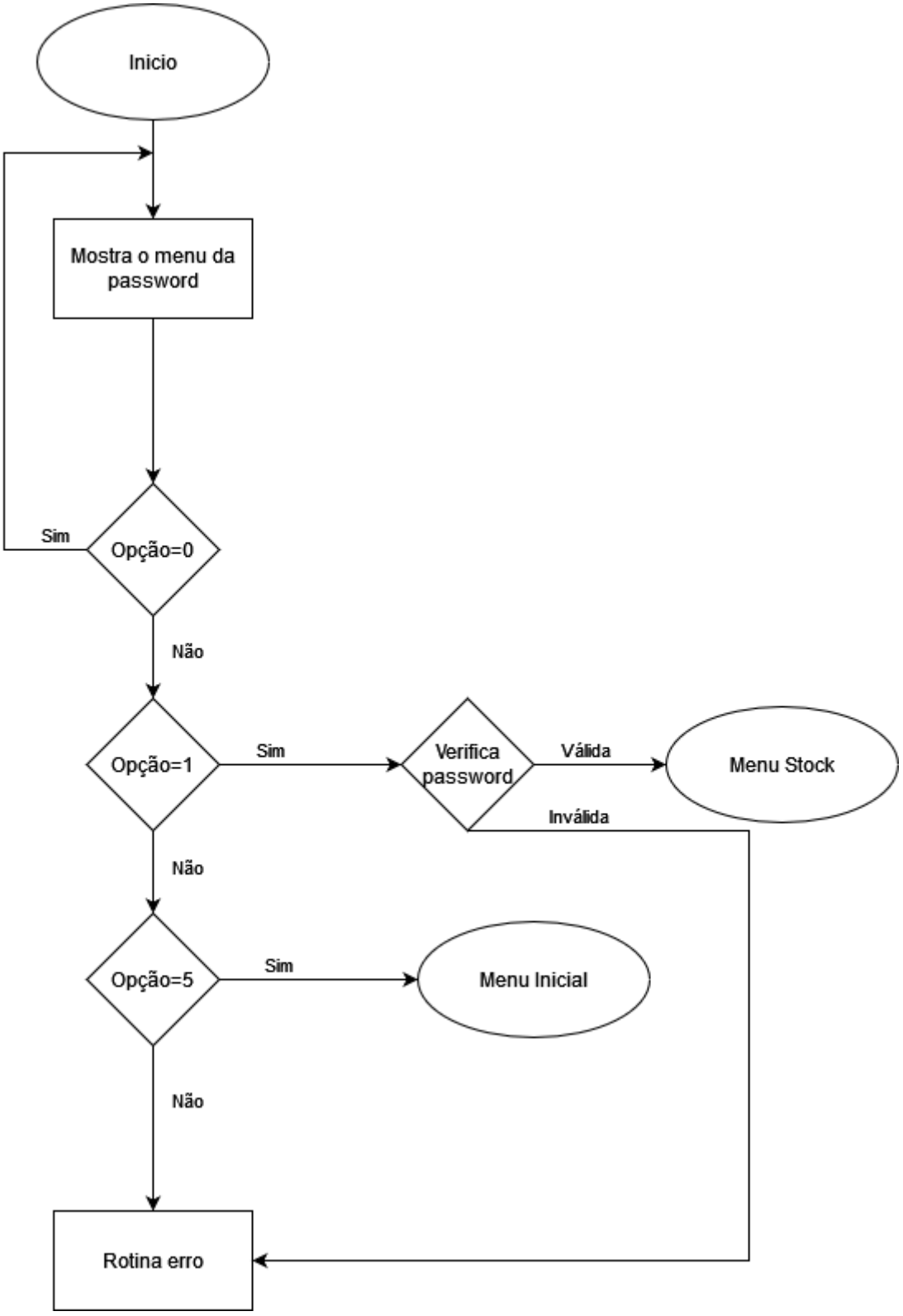
8.Anexo A

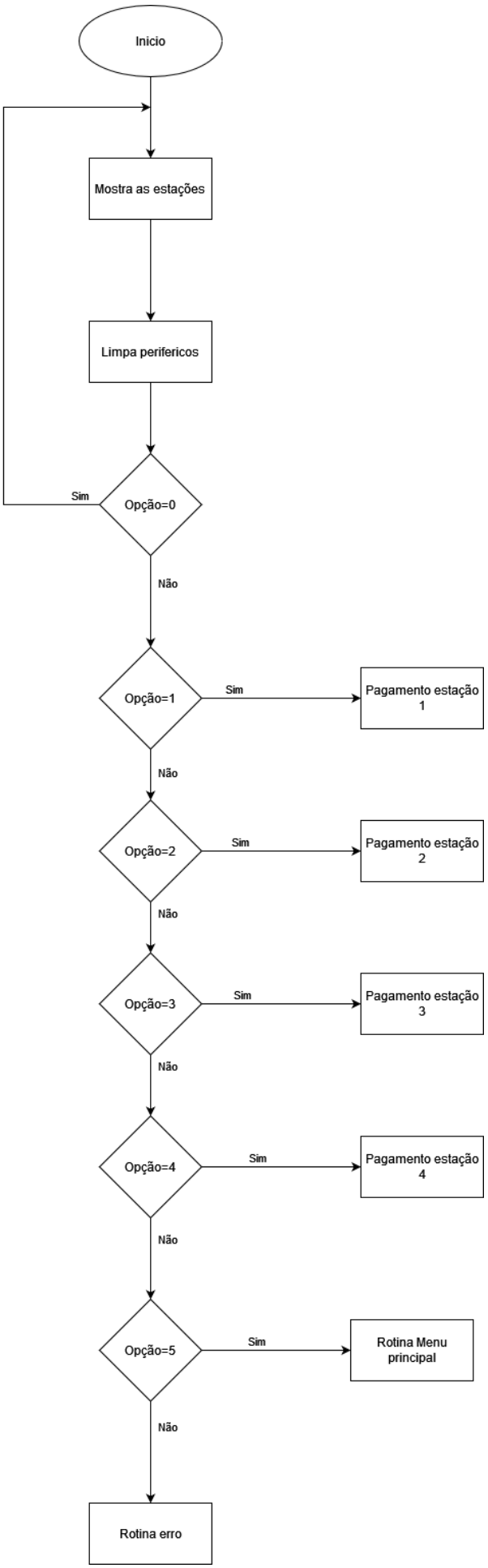

LimparDisplay
Inicio
Inicio=Display
FIM=Fim_Display
Escreve palavra
Passa para a
próxima palavra
Chegou ao fim do
display?
Sim
Fim
Não

INICIO

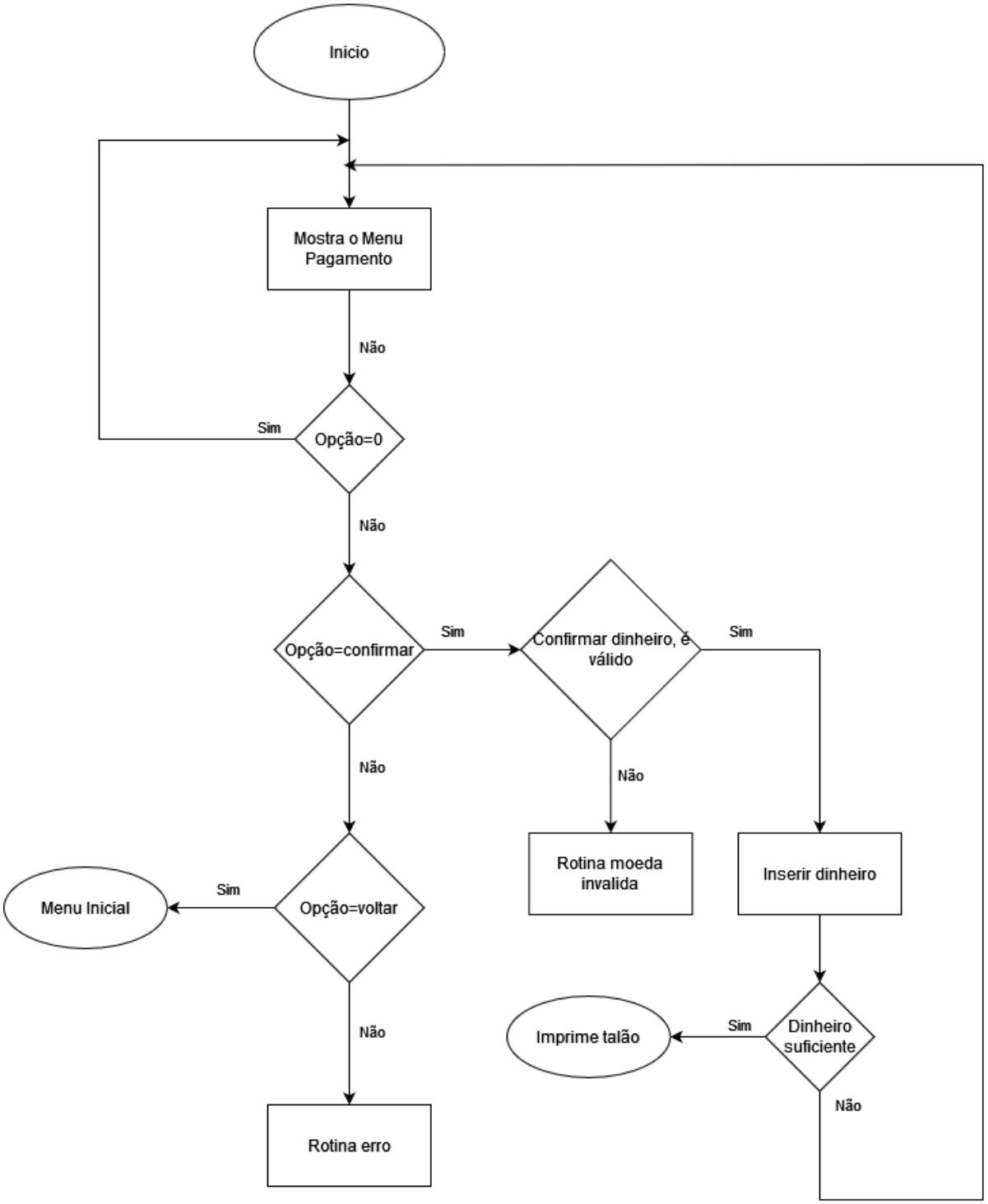

Inicio
Display vazio/ desligado
Lê botão ON/OFF
ON/OFF
0
1
Mostra o Menu Inicial
Limpa perifericos
Lê opção
Opção=0
Sim
Não
Opção=comprar
Sim
Rotina comprar
Não
Opção=usar cartão
Sim
Rotina usar cartão
Não
Opção=stock
Sim
Rotina stock
Não
Rotina erro

PasswordStock




Inicio
Mostra as estações
Limpa perifericos
Opção=0
Sim
Não
Opção=1
Sim
Pagamento estação 1
Não
Opção=2
Sim
Pagamento estação 2
Não
Opção=3
Sim
Pagamento estação 3
Não
Opção=4
Sim
Pagamento estação 4
Não
Opção=5
Sim
Rotina Menu principal
Não
Rotina erro

Pagamento


Inicio
Mostra o Menu Pagamento
Não
Sim
Opção=0
Não
Opção=confirmar
Sim
Confirmar dinheiro, é válido
Sim
Não
Não
Rotina moeda invalida
Inserir dinheiro
Sim
Opção=voltar
Menu Inicial
Não
Sim
Dinheiro suficiente
Imprime talão
Não
Rotina erro

10.Anexo B

```
; Entrada
ON_OFF EQU 0010H                ;endereço do botao ON/OFF da MAQUINA
SelecionarOpcao EQU 0020H   ;Endereco do Seletor de Opcao
password EQU 0030H              ;Endereço onde será introduzida a Password
Introduzir_PEPE EQU 0040H   ;Endereco onde será introduzido o N do PEPE

; Saida
Display EQU 50H                                         ;Endereço de onde
começa o display
Fim_Display EQU 00BFH                                   ;Endereço de onde termina o
display
Display_primeiro_valor_stock EQU 70H        ;Linha da memoria onde será escrito o
numeros de moedas em stock(3 linha do display)
fim_display_primeiro_valor_stock EQU 7FH    ;Ultimo Byte da linha acima descrita
Display_segundo_valor_stock EQU 90H             ;Mesmo que acima mas da proxima
linha de moedas (linha 5 no caso)
fim_display_segundo_valor_stock EQU 9FH         ;Fim da linha acima
vazio EQU 20H                                           ;Caractere Vazio em
ascii

;Display de troco
DisplayM10 EQU 65H          ;Onde fica armazenado o numero de moedas de 10  a serem
dadas como Troco
DisplayM20 EQU 6DH          ;Onde fica as moedas de 20 centimos
DisplayM50 EQU 75H          ;Onde fica As de 50 centimos
DisplayM1 EQU 7DH           ;Onde fica as de 1 Euro
DisplayM2 EQU 85H           ;Onde fica as de 2 euros
DisplayN5 EQU 8DH           ;Onde fica as notas de 5 euros
DisplayN10 EQU 9AH          ;Onde fica as notas de 10 euros
DisplayN20 EQU 0AAH         ;Onde fica as notas de 20 euros


; Opcoes Menu Inicial
comprar EQU 1               ;Opcao comprar
cartao EQU 2                ;Opcao de CARTAO
stock EQU 3                 ;Opcao de stock

; Opcoes gerais
Continuar EQU 1             ;Opcao de continuar
Recarregar EQU 2        ;Opcao de Recarregar
```

```
Cancelar EQU 5                ;Opcao de Cancelar
Voltar EQU 5                  ;Opcao de Voltar
StackPointer EQU 6000H     ;Se nao for definido é usada a ultima linha da memoria, neste caso escreve para cima deste

; Comprar-Estacoes
E2 EQU 1                      ;Opcao da Estacao 2
E3 EQU 2                      ;Opcao da Estacao 3
E4 EQU 3                      ;Opcao de Estacao 4
E5 EQU 4                      ;Opcao da estacao 5

;Stock de moedas e Notas -- O stock funciona de forma semelhante ao display, onde serão guardados dois bytes para cada tipo de moeda/nota aceite sendo armazenados pela seguinte orderm (moedas de 10 centimos, 20 centimos, 50, 1 euro, 2 euros, notas de 5, notas de 10, notas de 20
M_10 EQU 00D0H      ;comeco da memoria onde se guarda o numero de moedas de 10
F_M_10 EQU 00D3H     ;fim da memoria das moedas de 10 centimos
M_20 EQU 00E0H      ;O mesmo de cima so que pras de 20
F_M_20 EQU 00E3H
M_50 EQU 00F0H
F_M_50 EQU 00F3H
M_01 EQU 0100H      ;O mesmo para todas as moedas e notas
F_M_01 EQU 0103H
M_02 EQU 0110H
F_M_02 EQU 0113H
N_05 EQU 0120H
F_N_05 EQU 0123H
N_10 EQU 0130H
F_N_10 EQU 0133H
N_20 EQU 0140H
F_N_20 EQU 0143H


;Moedas Inseridas
M_10_inseridas EQU 01E0H                ;Endereco onde esta as Moedas de 10 inseridas
M_20_inseridas EQU 01F0H                ;Endero onde esta as Moedas de 20 inseridas
M_50_inseridas EQU 0200H
M_1_inseridas EQU 0210H
M_2_inseridas EQU 0220H                 ;O mesmo para todas as moedas e notas
N_5_inseridas EQU 0230H
N_10_inseridas EQU 0240H
N_20_inseridas EQU 0250H

;Moedas para dar de troco
```

```
M_10_troco EQU 0280H                    ;O mesmo que acima mas estes enderecos guardam as moedas a serem dadas como troco
M_20_troco EQU 0290H
M_50_troco EQU 02A0H
M_1_troco EQU 02B0H
M_2_troco EQU 02C0H
N_5_troco EQU 02D0H
N_10_troco EQU 02E0H
N_20_troco EQU 02F0H

;Valores de cada Moeda em decimal

M_10_HEX EQU 000AH                      ;Aqui esta cada valor das moedas em Hexadecimal para facilitar as contas
M_20_HEX EQU 0014H
M_50_HEX EQU 0032H
M_01_HEX EQU 0064H
M_02_HEX EQU 00C8H
N_5_HEX EQU 01F4H
N_10_HEX EQU 03E8H
N_20_HEX EQU 7D0H


;compras
Total  EQU 0190H                        ;endereco onde se guarda o total a pagar para ser mais facil de guardar sem ter de  ocupar um registo constantemente
Total_inserido EQU 01A0H                ;Aqui guarda-se o Total que ja foi inserido pelo utilizador
Diferenca_troco EQU 01B0H               ;Aqui guarda-se a diferença entre o TOtal inserido e o Total, ou seja, o troco a ser dado

Stock_padrao EQU 0AH                    ;por padrão iremos comecar com 10 de cada tipo de moeda e nota (ascii)

;Mascara
Mascara_primeiro_digito EQU 0F0H        ;Mascara que isola o primeiro digito de um byte usando um AND
Mascara_segundo_digito EQU 0FH          ;Igual mas isola o segundo digito de um byte
Mascara_primeiro_byte EQU 0FF00H        ;Mascara que isola o primeiro byte de uma palavra
Mascara_segundo_byte EQU 00FFH          ;Mascara que isola o segundo Byte de uma  palavra

End_Mascara_primeiro_digito EQU 0300H   ;Endereco onde se guarda a Mascara de primeiro digito
End_Mascara_segundo_digito EQU 0310H    ;Endereço onde se guarda a mascara do segundo digito
```

```
End_Mascara_primeiro_byte EQU 0320H     ;endereço onde se guarda a mascara do primeiro Byte
End_Mascara_segundo_byte EQU 0330H     ;Endereço onde se guarda a mascara do segundo byte

;Conversores
End_conversor EQU 0360H                                 ;Enderecos auxiliares usados para realizar as conversoes de HEX para ASCII e Decimal
End_conversor_aux EQU 0370H
End_conversor_aux2 EQU 0380H

Conversor_Hex_para_dec EQU 0400H        ;Endereço onde se guarda o valor convertido

;PEPEs
Saldo_PEPE EQU 06H ;O Saldo do PEPE é guardado nos bytes 6 e 7 de cada um (basta somar 6H em cada um pra verificar o saldo)

place 2000H
Menu_Inicial: ; Escrever Menu inicial
   String " MAQUINA VENDAS "
   String "       METRO          "
   String "                      "
   String "1- COMPRAR         "
   String "2- USAR CARTAO   "
   String "3- STOCK              "
   String "                      "

place 2500H
Menu_comprar: ; Escrever Menu comprar
   String "  MENU ESTACAO  "
   String "1-Estacao2: 1.50"
   String "2-Estacao3: 2.50"
   String "3-Estacao4: 3.50"
   String "4-Estacao5: 4.50"
   String "                      "
   String "5-Acabar compra "

place 2580H
Menu_troco: ; Escreve o Menu de mostrar o troco
   String "  PEPE GERADO   "
   String "                      "
   String "                      " ;O espaço será preenchido posteriormente para alterar
   String "                      "
   String "                      "
   String "5- Voltar      "
   String "                      "

place 2600H
```

```
Menu_usarcartao: ; Escreve o Menu usar cartao
   String "   INTRODUZA N  "
   String "       PEPE NO       "
   String "   PERIFERICO   "
   String "   DE ENTRADA   "
   String "   EX: 1234  "
   String "1-Continuar           "
   String "5-Cancelar  "

place 2680H
Menu_saldoPepe: ; Escreve menu do saldo do pepe
   String "   SALDO PEPE   "
   String "                "
   String "                "
   String "                "
   String "1-Comprar             "
   String "2-Recarregar          "
   String "5-Cancelar  "

place 2700H
Menu_Stock1: ; Escreve Menu de Stock
   String "-- Stock 1/4 ---"
   String " Moedas 10 Cent:"
   String "                "
   String " Moedas 20 Cent:"
   String "                "
   String "1-Seguinte  "
   String "5-Menu principal"

place 2780H
Menu_Stock2: ; Escreve Menu de Stock
   String "-- Stock 2/4 ---"
   String " Moedas 50 Cent:"
   String "                "
   String " Moedas 1  Euro:"
   String "                "
   String "1-Seguinte  "
   String "5-Menu principal"

place 2800H
Menu_Stock3: ; Escreve Menu de Stock
   String "-- Stock 3/4 ---"
   String " Moedas  2 Euro:"
   String "                "
   String " Notas  5 Euros:"
   String "                "
   String "1-Seguinte  "
   String "5-Menu principal"
```

```
place 2880H
Menu_Stock4: ; Escreve Menu de Stock
   String "-- Stock 4/4 ---"
   String " Notas 10 Euros:"
   String "                "
   String " Notas 20 Euros:"
   String "                "
   String "1-Seguinte   "
   String "5-Menu principal"


place 2900H
Menu_erro: ; Menu mostrado quando é selecionada uma opção errada
   String "       Atencao          "
   String "                "
   String " Opcao Invalida "
   String "                "
   String "                "
   String "5-Voltar        "
   String "                "

place 2980H
Menu_passErrada: ;Menu mostrado quando é selecionada uma opção errada
   String "       Password    "
   String "       Errada "
   String "                "
   String "                "
   String "                "
   String "1-Menu principal"
   String "                "

place 2A00H
Menu_pass: ; Menu mostrado quando é selecionada uma opção errada
   String "   Introduza a  "
   String "       password        "
   String "                "
   String "                "
   String "1-Confirmar           "
   String "5-Menu principal"
   String "                "

Place 2A80H
continuar_compra:
   String " Vai adicionar  "
   String "   a Estacao "
   String "                "  ;substituir por estação (Display + 30)
```

```
    String " ao seu carrinho"
    String " Quer Continuar?"
    String "1- Continuar          "
    String "5- VOltar     "

Place 2B10H
;Menu quando carrinho ta vazio
carrinho_vazio:
    String "   O Carrinho   "
    String "       esta     "
    String "       Vazio    "
    String "                "
    String "                "
    String " 1- Continuar   "
    String "                "

Place 2B90H
cartao_gerado: ;menu quando o cartao
    String "   O Codigo  "
    String "       do seu           "
    String "       cartao           "
    String "       Gerado:          "
    String "                "
    String "                "
    String "1-Continuar           "

PLACE 2C10H
checkout_menu_sem_saldo: ;Menu de checkout quando nao se utiliza cartao
    String "       Total a "
    String "       pagar:  "
    String "                "
    String "                "
    String "1-Continuar           "
    String "para o Pagamento"
    String "5-Cancelar  "

PLACE 2C90H
InserirMoedas: ;Menu para inserir moedas
    String "  Dinheiro EUR  "
    String "1- 0.10 / 6- 2  "
    String "2- 0.20 / 7- 5  "
    String "3- 0.50 / 8- 10 "
    String "4- 1  / 9- 20 "
    String "A-Confirmar           "
    String "5-Cancelar  "

PLACE 2D10H
Inseriu_Moeda10:  ;Menu depois de inserir moeda de 10 centimos
```

```
   String "                "
   String "       Inseriu          "
   String "       uma    "
   String "       Moeda            "
   String "       de 10   "
   String "       centimos         "
   String "1-Continuar             "
PLACE 2D90H
Inseriu_Moeda20:   ;Menu depois de inserir moeda de 20 centimos
   String "                "
   String "       Inseriu          "
   String "       uma    "
   String "       Moeda            "
   String "       de 20   "
   String "       centimos         "
   String "1-Continuar             "
PLACE 2E10H
Inseriu_Moeda50:   ;Menu Depois de inserir uma moeda de 50 centimos
   String "                "
   String "       Inseriu          "
   String "       uma    "
   String "       Moeda            "
   String "       de 50   "
   String "       centimos         "
   String "1-Continuar             "
PLACE 2E90H
Inseriu_Moeda1:         ;igual para os proximos menus so que para as proximas moedas e
notas
   String "                "
   String "       Inseriu          "
   String "       uma    "
   String "       Moeda            "
   String "       de 1    "
   String "       euro    "
   String "1-Continuar             "
PLACE 2F10H
Inseriu_Moeda2:
   String "                "
   String "       Inseriu          "
   String "       uma    "
   String "       Moeda            "
   String "       de 2    "
   String "       euros   "
   String "1-Continuar             "
PLACE 2F90H
Inseriu_Nota5:
   String "                "
   String "       Inseriu          "
```

```
   String "       uma     "
   String "       Nota    "
   String "       de 5    "
   String "       euros   "
   String "1-Continuar           "
PLACE 3010H
Inseriu_Nota10:
   String "                "
   String "       Inseriu          "
   String "       uma     "
   String "       Nota    "
   String "       de 10   "
   String "       euros   "
   String "1-Continuar           "
PLACE 3090H
Inseriu_Nota20:
   String "                "
   String "       Inseriu          "
   String "       uma     "
   String "       Nota    "
   String "       de 20   "
   String "       euros   "
   String "1-Continuar           "

Place 3190H
Menu_saldo_ins: ;Menu quando o saldo do cartao nao é suficiente para concluir uma
compra
   String "       Saldo   "
   String "   insuficiente "
   String "                "
   String "                "
   String "                "
   String "                "
   String "1-Continuar pag "

Place 3210H
Menu_troco_a_dar: ;Menu com o troco a dar ao utilizador
   String "   Troco a dar  "
   String "0.10-XX|0.20-XX "
   String "0.50-XX|1.00-XX "
   String "2.00-XX|5.00-XX "
   String "       10.00-XX        "
   String "       20.00-XX        "
   String "1-Continuar           "

Place 3290H
Menu_PEPE_invalido:   ;Menu quando o Numero dado pelo utilizador nao e um PEPE
valido
```

```
    String "        ATENCAO       "
    String "    O NUMERO          "
    String " DE PEPE NAO E  "
    String "        VALIDO        "
    String "                      "
    String "1-CONTINUAR       "
    String "                      "

Place 3310H
Menu_SEM_troco:    ;Menu quando nao ha troco suficiente
    String " Nao tem troco  "
    String "  Suficiente  "
    String "  Contacte um    "
    String "   Responsavel  "
    String "        Faltou: "
    String "                      "
    String "1-CONTINUAR       "

Place 3410H
Menu_compra_cartao: ;Menu quando é realizada um compra com o cartao
    String "        Compra        "
    String "        Realizada    "
    String "        com o  "
    String "        Cartao        "
    String "   com sucesso  "
    String "                      "
    String "1-CONTINUAR       "


place 0000H                     ;No começo do programa saltar para onde temos as nossas
instrucoes (4000H)
Inicio:
    MOV R0, comeco
    JMP R0

place 4000H
;Todas as opcoes int, são intermedios para evitar o erro de Constant out of Bound, ou seja
salta para um intermedio para conseguir alcançar o objetivo final
;-----------------------------------Opcao comprar int-----------------------------------------------
Opcao_comprar_int:
    JMP Opcao_comprar
;-----------------------------------Opcao cartao int------------------------------------------------
Opcao_cartao_int:
    JMP Opcao_cartao
;-----------------------------------Opcao password int----------------------------------------------
Menu_Password_int:
    JMP Menu_Password
```

```
comeco:
   MOV SP, StackPointer   ;Move O stack Pointer para o seu endereço definido
   CALL LimparDisplay_int    ;Limpa o Display logo no começo do programa
   CALL Limparperifericos_int    ;Limpa os perifericos logo no começo do programa
   MOV R0, ON_OFF    ;copia o endereço do botao ON_OFF
Ligar:
   MOVB R1, [R0] ; Copia apenas um byte de R0, ou seja o Valor de ON_OFF (1/0)
   CMP R1, 1 ; Compara com 1
   JNE Ligar ; Caso nao seja 1 é 0, logo volta ao inicio desta rotina e fica neste ciclo até ser
1
Ligado:
   CALL Alocar_mascaras     ;Chama a rotina que aloca as mascaras ao seu endereço
correspondente
   CALL Definir_stock_padrao_int    ;No começo do programa vamos repor as moedas e
notas da maquina para um valor Padrao, neste caso 10
Ligado_e_stockado: ;Depois de repormos o stock prosseguimos
   CALL Limpa_moedas_inseridas          ;Limpamos todos os endereços de moedas que
nao sejam do stock para evitar valores a mais
   CALL LimparMoedasTroco_int
   CALL Limpa_total                     ;Reinicia o total a 0
   CALL Limpa_total_inserido            ;Reinicia o total inserido a 0
   CALL LimparDisplay_int               ;Limpa o Display
   MOV R2, Menu_Inicial                 ;Move para o R2 o Menu inicial. R2 é o registo
usado pela função mostrar display para escrever o menu
   CALL MostrarDisplay_int              ;Mostra o Menu inicial
   CALL Limparperifericos_int           ;Limpa os perifericos
Ler_Opcao:
   MOV R0, SelecionarOpcao              ;Atribui a  R0 o endereço do selecionar
Opcao
   MOVB R1, [R0]                        ;Atribui a R1, o valor no endereço R0
   CMP R1, 0                                ;Se este valor de R1 continuar a
0, permanecer no Let Opcao
   JEQ Ler_Opcao                        ; Fica infinitamente nesta rotina até ser
inserido um valor no endereço de SelecionarOpcao
   CMP R1, comprar                          ;Vai comparar se o input do
selecionar opcao é igual a constante comprar (1)
   JEQ Opcao_comprar_inicio             ;Se for, salta para a opcao_comprar_incio
   CMP R1, cartao                       ;Compara o R1 com a constante cartao
   JEQ Opcao_cartao_int                 ;Se for igual salta para a opcao cartao
   CMP R1, stock                        ;Por fim compara com o Stock
   JEQ Menu_Password_int                ;Se for salta para o menu para
introduzir a password para poder aceder ao Stock
   CALL Rotina_erro                     ;Se não for nenhuma destas opcoes, foi
introduzido uma opcao invalida entao chamamos a rotina de erro
   JMP Ligado_e_stockado                ;Se de alguma maneira, passar por todos
estes, volta para a rotina anterior(Nao é suposto atingir este passo, só se ocorrer algum erro
inesperado, como da aplicação por exemplo)
```

```
;-----------------------------------------------Colocar Mascaras nos Endereços----------------------------------
Alocar_mascaras:                ;Nesta rotina iremos alocar as mascaras aos seus respetivos endereços
   PUSH R8                      ;usamos Push e POP para evitar o problema de faltar registos para as rotinas
   PUSH R9
   MOV R8, Mascara_primeiro_digito          ;Move para R8 o valor da mascara de primeiro digito
   MOV R9, End_Mascara_primeiro_digito ;E move para R9, o endereço onde esta deve ser guardada
   MOVB [R9], R8                            ;COloca o valor da mascara(presente em R8) no endereço R9
   MOV R8, Mascara_segundo_digito           ;Repete o mesmo para as 4 mascaras que temos
   MOV R9, End_Mascara_segundo_digito
   MOVB [R9], R8
   MOV R8, Mascara_primeiro_byte
   MOV R9, End_Mascara_primeiro_byte
   MOV [R9], R8
   MOV R8, Mascara_segundo_byte
   MOV R9, End_Mascara_segundo_byte
   MOV [R9], R8
   POP R9                       ;Fazemos o POP de forma invertida ao PUSH (First In last out)
   POP R8
   RET                          ;Retorna para onde esta rotina foi chamada
; -----------------------------------------------Mostrar Menu Erro-----------------------------------------------
Rotina_erro:                    ;Em caso de erro vamos escrever o MENU erro e esperar o input do utilizador para prosseguir
   MOV R2, Menu_erro            ;Movemos para o R2 o MEnu que desejamos escrever
   CALL MostrarDisplay          ;Mostramos o Display
   CALL Limparperifericos_int   ;Limpamos os perifericos
CicloErro:
   MOV R0, SelecionarOpcao      ;Copiamos para R0 o endereço do periferico de entrada onde vamos receber o seleciona opcao
   MOVB R1, [R0]                ;Guarda no reg 1 o valor no endereço da opção guardado no reg 0, ou seja a opcao introduzida pelo utilizador
   CMP R1, Voltar               ;Compara R1 com Voltar
   JEQ Ligado_e_stockado        ;Se for igual volta ao menu incial depois de ter atribuido o stock. Depois de o fazermos para evitar reiniciar o stock ao padrao
   JMP CicloErro                ;Caso nao seja o valor voltar, continua no ciclo
```

```
; ------------------------------------------------Mostrar Menu Erro no carrinho------------------------------------------------
Rotina_erro_carrinho:                       ;Caso de erro, no carrinho voltamos pro menu comprar e nao pro inicio
   CALL LimparDisplay_int                   ;Tudo igual ao anterior so muda o JMP Final
   MOV R2, Menu_erro
   CALL MostrarDisplay
   CALL Limparperifericos
CicloErro_carrinho:
   MOV R0, SelecionarOpcao
   MOVB R1, [R0]                            ; Guarda no reg 1 o valor no endereço da opção guardado no reg 0
   CMP R1, Voltar       ;Caso nao seja o valor voltar continua no ciclo
   JEQ Opcao_comprar    ;Manda para o menu de compra
   JMP CicloErro_carrinho
; ------------------------------------------------Mostrar Menu Comprar------------------------------------------------
Opcao_comprar_inicio:                       ;começa a rotina de compra
   Call LimparDisplay_int                   ;Limpa o display e perifericos como sempre
   CALL Limparperifericos
   MOV R2, cartao_gerado                    ;Move o Menu desejado pro R2, Cartao gerado neste caso
   CALL MostrarDisplay                      ;Mostra o Display
   MOV R6, Display_segundo_valor_stock    ;Move o endereço do display onde queremos escrever o codigo do PEPE gerado
   MOV R0, 0                                ;Inicia o R0 que sera alterado na rotina Gerar_PEPE
   CALL Gerar_PEPE                          ;o ENDEREÇO DO PEPE GERADO ESTARA EM R0 E SERA O NOSSO CODIGO DO cartao
   MOV R7, 2020H                            ;Mov para o R7 o codigo de dois bytes com o codigo ascci do caracter vazio
   MOV[R6], R7                              ;Mete vazio no endereço do display
   ADD R6, 2                                ;Avança em 2 bytes
   MOV[R6], R7                              ;poe Vazio
   ADD R6, 2                                ;repete
   MOV[R6],R7
   ADD R6, 2                                ;avanca em 2 bytes
   MOV R9, End_conversor_aux                ;Mov para R9 o endereço usado para a conversao do valor HEX(endereço do PEPE gerado) para ASCII (Para mostrar ao utilizador o seu codigo)
   MOV R10, Mascara_primeiro_byte           ;Atribui a R10 a mascara de primeiro byte
   AND R10, R0                              ;isola o primeiro Byte do endereço do PEPE Gerado e passsa para o R10
   MOV [R9], R10                            ;Move para o endereço R9 o valor do primeiro byte do PEPE Gerado
```

```
   MOVB R2, [R9]                          ;Move a R2 esse valor para usar
no conversor
   CALL Hex_para_Ascii                    ;chama a rotina que vai
converter esse valor para ASCII
   MOV [R6], R2                           ;MOve para o endereço R6 o
valor convertido
   ADD R6, 2                                    ;Fazemos o mesmo para
o segundo byte
   MOV R10, Mascara_segundo_byte
   AND R10, R0
   MOV R8, 100H                           ;Multiplicamos por 100H para
mover para o primeiro byte
   MUL R10, R8
   MOV [R9], R10                          ;Repetimos o processo
   MOVB R2, [R9]
   CALL Hex_para_Ascii
   MOV [R6], R2
   ADD R6, 2
   MOV[R6], R7
   ADD R6, 2
   MOV[R6],R7
   ADD R6, 2
   MOV [R6],R7
   CALL N_PEPE_talao                      ;Coloca o N-pepe no talao
ciclo_cartao_gerado:                      ;ciclo para esperar a resposta do
utilizador
   MOV R7, 0000H
   MOV R8, SelecionarOpcao
   MOVB R7, [R8]
   CMP R7, 0
   JEQ ciclo_cartao_gerado
   CMP R7, Continuar
   JEQ Continuar_cartao_gerado

Continuar_cartao_gerado:          ;Se o utilizador colocar continuar limpa o total,
para confirmar que esta vazio
   CALL Limpa_total
   MOV R4, 0 ;Limpa o total a pagar
   JMP Opcao_comprar                      ;Salta para a opcao_comprar
Opcao_comprar:
   CALL LimparDisplay                           ;Mostra o Menu_comprar
   MOV R2, Menu_comprar
   CALL MostrarDisplay
   CALL Limparperifericos                 ;Limpa perifericos
CicloComprar:
   MOV R5, SelecionarOpcao                ;Espera o input do utilizador
   MOVB R1, [R5]
```

```
   CMP R1, 0                    ; Guarda no reg 1 o valor no endereço da opção guardado no
reg 0
   JZ CicloComprar                                    ; Caso a opção continue igual a 0
continua o ciclo ate o estado mudar
   CMP R1, E2                                          ;Se for o valor de alguma das
estacoes vai para o menu de compra da estacao
   JEQ Pagamento_E2_int
   CMP R1, E3
   JEQ Pagamento_E3_int
   CMP R1, E4
   JEQ Pagamento_E4_int2
   CMP R1, E5
   JEQ Pagamento_E5_int2
   CMP R1, Cancelar
   JEQ Finalizar_compra_int2
   CALL Rotina_erro
   JMP CicloComprar
;-------------------------------------------------Converter de Hex para
ascii-----------------------------------
;R2 terá 2 digitos a ser convertidos (MOVB)
Hex_para_Ascii:
   PUSH R4
   PUSH R5
   PUSH R1
   PUSH R3
   MOV R3, 10H ;10 é a base pela qual vamos dividir
   MOV R1, 30H ;30 é a base que vamos somar para conveter de decimal para ASCII
   MOV R5, End_conversor_aux2
   MOV R4, Mascara_primeiro_digito
   AND R4, R2 ;Agora R4 tem apenas o primeiro digito
   DIV R4, R3 ;Passa do penultimo digito do registo pro ultimo para converter corretamente
(de 8 para 38H se nao ia tentar fazer 80H + 30H)
   ADD R4, R1 ;Converte esse digito em ascii
   MOVB [R5], R4 ;Move para o primeiro Byte do R5
   MOV R4, Mascara_segundo_digito
   AND R4, R2 ;Agora R4 tem o segundo digito
   ADD R4, R1 ;Converte o segundo digito em ascii
   ADD R5, 1
   MOVB [R5], R4 ;Move para o segundo Byte do R5
   SUB R5, 1
   MOV R2, [R5] ;O R2 que inicialmente tinha 2 digitos recebe os digitos convertidos (4)
   POP R3
   POP R1
   POP R5
   POP R4
   RET
;-------------------------------------------------Gerar
PEPE-----------------------------------------------------
```

```
Gerar_PEPE:
   MOV R0, 8000H     ;8000 é onde comecam os espacos para PEPEs
   MOVB R1, [R0]    ;Vamos verificar se o primeiro digito da linha de memoria é 1. Isto significa que este pepe ja foi gerado anteriormente e temos de procurar outro espaco
   CMP R1, 1
   JEQ verificar_proximo_PEPE
   MOV R2, 1
   MOVB [R0], R2 ;primeiro bit do PEPE Mostra se este espaço já está definido ou não --- atualmente R0 é o numero de PEPE
   ;O proximo byte fica vazio por conveniencia e facilidade de leitura no byte 6 e 7 de cada linha temos o saldo fo PEPE em HEX (Ex: 8006)
   ADD R0, 2              ;Os proximos bytes ficam com 0000 em ascii que é o saldo incial do PEPE mas acabou por nao ser usado logo nao era necessario
   MOV R4, 3030H
   MOV [R0], R4
   ADD R0,2
   MOV R4,3030H
   MOV [R0], R4 ;Assim vai determinar o saldo de cada PEPE gerado como 0000 centimos (Para mostrar na tela posteriormente, assumimos que os ultimos 2 digitos são os centimos(depois da virgula)) Nos proximos 2 bytes fica o saldo de cada PEPE em hexadecimal para facilitar o acesso
   SUB R0, 2
   SUB R0, 2
   RET
verificar_proximo_PEPE:
   MOV R3, 10H                ;Soma para passar para a proxima linha e faz a mesma verificaçao
   ADD R0, R3
   MOVB R1, [R0]
   CMP R1, 1
   JEQ verificar_proximo_PEPE
   MOV R2, 1
   MOVB [R0], R2
   ADD R0, 2
   MOV R4, 3030H
   MOV [R0], R4
   ADD R0,2
   MOV R4,3030H
   MOV [R0], R4 ;Assim vai determinar o saldo de cada PEPE gerado como 0000 centimos (Para mostrar na tela posteriormente, assumimos que os ultimos 2 digitos são os centimos(depois da virgula))
   SUB R0, 2
   SUB R0, 2
   RET

;-----------------------Definir_stock_padrao int-----------------
Definir_stock_padrao_int:
   CALL Definir_stock_padrao
```

```
   RET
;------------------------------Limpar perifericos int´----------------------
Limparperifericos_int:
   CALL Limparperifericos
   RET
;-------------------------------------------------Pagamento_E2
int-----------------------------------------------
Pagamento_E2_int:
   JMP Pagamento_E2
;-------------------------------------------------Pagamento_E3
int-----------------------------------------------
Pagamento_E3_int:
   JMP Pagamento_E3
;-------------------------------------------------Pagamento_E4
int-----------------------------------------------
Pagamento_E4_int2:
   JMP Pagamento_E4_int
;-------------------------------------------------PAgamento E5 int-----------------------------------------------
Pagamento_E5_int2:
   JMP Pagamento_E5_int
;-------------------------------------------------Finalizar compra int 2-----------------
Finalizar_compra_int2:
   JMP Finalizar_compra_int
   ;-------------------------------------------------Checkout int-----------------
Checkout_int:
   JMP Checkout
         ;-------------------------------------------------Ligado e stockado 3 int-----------------
Ligado_e_stockado_int3:
   JMP Ligado_e_stockado_int2

;-------------------------------------------------Limpa total-----------------------
Limpa_total:
PUSH R5
PUSH R6
MOV R5, Total          ;Muda todos os bytes do total para 0
MOV R6, 0
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
```

```
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
POP R6
POP R5
RET
    ;-----------------------------------------------Limpa Total_inserido-----------------------
Limpa_total_inserido:
PUSH R5
PUSH R6
MOV R5, Total_inserido                          ;Igual ao de cima mas do total_inserido
MOV R6, 0
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
```

```
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
ADD R5, 1
MOVB [R5], R6
POP R6
POP R5
RET

;---------------------------------------Limpar moedas troco int-------------------
LimparMoedasTroco_int:
        CALL LimparMoedasTroco
   RET
;-----------------------------------------------Limpa moedas
inseridas-----------------------------------------
Limpa_moedas_inseridas:
   MOV R0, M_10_inseridas                    ;vai ao endereço de cada moeda inserida e
muda por 0, limpando assim as moedas_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   MOV R0, M_20_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   MOV R0, M_50_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   MOV R0, M_1_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   MOV R0, M_2_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   MOV R0, N_5_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   MOV R0, N_10_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   MOV R0, N_20_inseridas
   MOV R1, 0000H
   MOV [R0], R1
   RET
```

```
;-----------------------------------------------Opcao Comprar
int-----------------------------------------------
opcao_comprar_int:
        JMP Opcao_comprar
   ;---------------------------------------MostrarDisplay_int ----------------------
MostrarDisplay_int:
   CALL MostrarDisplay
   RET
;-----------------------------------------------ROTINA
PAGAMENTO-----------------------------------------------
Pagamento_E2:
   ;Vou usar R4 para armazenar o total a pagar
   CALL LimparDisplay                              ;rotina para adicionar E2 ao carrinho

   CALL Limparperifericos
   MOV R2, continuar_compra              ;é preciso alterar para mostrar a estaçao atual
   CALL MostrarDisplay
   MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
   MOV R7, 2020H
   MOV [R6], R7                                    ;Move caracteres vazios
   ADD R6, 2
   MOV [R6], R7
   ADD R6, 2
   MOV [R6], R7
   ADD R6, 2
   MOV R8, 4532H                                   ;Ecreve E2, no Menu
   MOV [R6], R8                                    ;45 é E em Hexadecimal
   ADD R6, 2
   MOV R6, R7
   ADD R6, 2
   MOV R6, R7
   ADD R6, 2
   MOV R6, R7
   ADD R6, 2
   MOV R6, R7

ciclo_pagamento_2:                         ;Espera o input do utilizador
   MOV R5, SelecionarOpcao
   MOVB R1, [R5]
   CMP R1, 0
   JZ ciclo_pagamento_2       ;Se for 0 fica em ciclo
   MOV R7, Continuar
   CMP R1, R7                                      ;Se for 1 salta para concluir pagamento
   JEQ concluir_pagamento_2
   MOV R7, Voltar
   CMP R1, R7
   JEQ opcao_comprar_int      ;Se for voltar volta ao menu comprar
```

```
    CALL Rotina_erro_carrinho

concluir_pagamento_2:
    MOV R5, 312EH                   ;Valor do preco em ascii 1.50 (31 = 1, 2E = . , 35 = 5, 0
    MOV R10, 350AH
    MOV R4, Total
    MOV R3, [R4]
    MOV R7, 150
    ADD R3, R7
    MOV R8, 270FH                   ;Valor maximo (9999)
    CMP R3, R8
    ;JGT total_invalido
    MOV [R4], R3
    MOV R4, 32H
    JMP Adicionar_talao
    JMP Opcao_comprar_int2


;----------------------------------------Converter_palavra de HEX PARA ASCII-------------------------
HEx_para_dec:
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    ;R0 tem o valor a converter
    MOV R1, 10
    MOV R2, Conversor_Hex_para_dec  ;onde vai ficar guardado o valor convertido
    ADD R2, 3                                   ;Caracter menos significativo
    MOV R3, 0                                   ;numeor de caracteres preenchidos
caracterseguinte:
    MOV R4, R0                                  ;Copia do valor a Converter
    MOD R4, R1                                  ;Resto da divisao inteira por 10
    DIV R0, R1                                  ;Atualiza o valor desejado - quociente
da divisao por 10
    MOV R5, 48                                  ;valor usado para converter ascii
    ADD R5, R4                                  ;Resto + 48
    MOVB [R2], R5                           ;Escrevemos na memeoria o carater
    SUB R2, 1                                   ;mudamos de caracter
    ADD R3, 1                                   ;Ja preenchemos um caracter logo
incrementamos um
    CMP R0, 0
    JNE caracterseguinte
ciclo_vazio:
    CMP R3, 4                                   ;Verifica se ja foram convertidos os
caracteres necessarios e, se sim termina a rotina
    JEQ  terminaconversao
    MOV R5, 30H
```

```
   MOVB [R2], R5
   SUB R2, 1                                   ;proxima posicao de memoria a
preencher
   ADD R3, 1                                   ;soma 1 ao numero de caracteres ja
convertidos
   JMP ciclo_vazio
terminaconversao:
   POP R5
   POP R4
   POP R3
   POP R2
   POP R1
   RET
;-----------------------------------------Pagamento_E4_int2----------------------------
Pagamento_E4_int:
   JMP Pagamento_E4
;-----------------------------------------Finalizar_compra_int-------------------------
Finalizar_compra_int:
   JMP Finalizar_compra

;-----------------------------------------Adicionar ao talão---------------------------
N_PEPE_talao:                  ;vai escrever o Numero do Pepe que foi usado no inicio
   MOV R6, 9000H
   MOV [R6], R0         ;N_PEPE esta no R0
   MOV R0, [R6]
   ADD R6, 2
   ADD R6, 2
   MOV R1, End_conversor_aux
   MOV R10, Mascara_primeiro_byte
   AND R10, R0                                 ;vamos separar os bytes para
poderem ser convertidos em ascii em ostrarmos na tela  o codigo corretamente
   MOV [R1], R10
   MOVB R2, [R1]
   CALL Hex_para_Ascii
   MOV [R6], R2
   ADD R6, 2
   MOV R10, Mascara_segundo_byte               ;Fazer o mesmo para o segundo byte
   AND R10, R0
   MOV R8, 100H
   MUL R10, R8
   MOV R8, 0000H
   MOV [R1], R10
   MOVB R2, [R1]
   CALL Hex_para_Ascii
   MOV [R6], R2
   RET
Adicionar_talao:
;R5 ja tem o valor do preco do bilhetes
```

```
;R4 tem o numero da estacao
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    PUSH R1
    MOV R6, 9010H ;Onde começa o talao
    MOV R7, 45H ;45 é E em Ascii, Se tiver E nesse espaco de memoria sabemos que esse
espaço do talão ta ocupado
    MOVB R8, [R6]
    CMP R8, R7 ;Se for E passamos à frente
    JEQ Proximo_item
    JMP escrever_talao
Proximo_item:
    MOV R0, 10H
    ADD R6, R0 ;passamos para a proxima linha de memoria do talao
    MOVB R8, [R6] ; passamos para R8 o valor nessa memoria
    CMP R8, R7 ; Se ja tiver E esta ocupado e passamos pro proximo
    JEQ Proximo_item ;vai para a rotina proximo_item
escrever_talao:
    MOV R9, 45H ;O R9 sera o nosso E
    MOVB [R6], R9 ;escrevemos o E que indica que esta cheio (E de estacao)
    ADD R6, 1 ;passamos para o proximo caracter que será a estacao a ser escrita no talao,
o valor da estacao esta escrito em ascii no registo R4, e o seu custo em R5
    MOVB [R6], R4
    ADD R6, 3
    MOV [R6], R5    ;Escreve o Preço
    ADD R6, 2
    MOV [R6], R10
    POP R1
    POP R9
    POP R8
    POP R7
    POP R6
    JMP Opcao_comprar_int

Adicionar_talao_cartao:
;R5 ja tem o valor do preco do bilhetes
;R4 tem o numero da estacao
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    PUSH R1
    MOV R6, 9010H ;Onde começa o talao
    MOV R7, 45H ;45 é E em Ascii, Se tiver E nesse espaco de memoria sabemos que esse
espaço do talão ta ocupado
    MOVB R8, [R6]
```

```
    CMP R8, R7 ;Se for E passamos à frente
    JEQ Proximo_item_cartao
    JMP escrever_talao_cartao
Proximo_item_cartao:
    MOV R0, 10H
    ADD R6, R0 ;passamos para a proxima linha de memoria do talao
    MOVB R8, [R6] ; passamos para R8 o valor nessa memoria
    CMP R8, R7 ; Se ja tiver E esta ocupado e passamos pro proximo
    JEQ Proximo_item_cartao ;vai para a rotina proximo_item
escrever_talao_cartao:
    MOV R9, 45H ;O R9 sera o nosso E
    MOVB [R6], R9 ;escrevemos o E que indica que esta cheio (E de estacao)
    ADD R6, 1 ;passamos para o proximo caracter que será a estacao a ser escrita no talao,
o valor da estacao esta escrito em ascii no registo R4, e o seu custo em R5
    MOVB [R6], R4
    ADD R6, 3
    MOV [R6], R5    ;Escreve o Preço
    ADD R6, 2
    MOV [R6], R10
    POP R1
    POP R9
    POP R8
    POP R7
    POP R6
    RET

;-----------------Limpar Display intermedio--------------------
LimparDisplay_int:
    CALL LimparDisplay
    RET
;-------------------------------------------------PAgamento E5 int-----------------------------------------------
Pagamento_E5_int:
    JMP Pagamento_E5


Pagamento_E3:
    ;Vou usar R4 para armazenar o total a pagar
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, continuar_compra ;é preciso alterar para mostrar a estaçao atual
    CALL MostrarDisplay
    MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
    MOV R7, 2020H
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
```

```
    ADD R6, 2
    MOV R8, 4533H
    MOV [R6], R8 ;45 é E em Hexadecimal
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7

ciclo_pagamento_3:
    MOV R5, SelecionarOpcao          ;igual as outras estacoes
    MOVB R1, [R5]
    CMP R1, 0
    JZ ciclo_pagamento_3
    MOV R7, Continuar
    CMP R1, R7
    JEQ concluir_pagamento_3
    MOV R7, Voltar
    CMP R1, R7
    JEQ Opcao_comprar_int2
    CALL Rotina_erro_carrinho

concluir_pagamento_3:
    MOV R5, 322EH
    MOV R10, 350AH
    MOV R4, Total ;Memoria onde se encontra o valor total a pagar do carrinho atual
    MOV R3, [R4]
    MOV R7, 250
    ADD R3, R7
    MOV R8, 270FH  ;9999 em decimal
    CMP R3, R8
    ;JGT total_invalido_int
    MOV [R4], R3
    MOV R4, 33H ;3 em Ascii
    CALL Adicionar_talao
    JMP Opcao_comprar_int2



Pagamento_E4:
    ;Vou usar R4 para armazenar o total a pagar
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, continuar_compra ;é preciso alterar para mostrar a estaçao atual
    CALL MostrarDisplay
```

```
    MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
    MOV R7, 2020H
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV R8, 4534H
    MOV [R6], R8 ;45 é E em Hexadecimal
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
ciclo_pagamento_4:
    MOV R5, SelecionarOpcao
    MOVB R1, [R5]
    CMP R1, 0
    JZ ciclo_pagamento_4
    MOV R7, Continuar
    CMP R1, R7
    JEQ concluir_pagamento_4
    MOV R7, Voltar
    CMP R1, R7
    JEQ Opcao_comprar_int2
    CALL Rotina_erro_carrinho

concluir_pagamento_4:
    MOV R5, 332EH ;3.50 em ASCII
    MOV R10, 350AH
    MOV R4, Total ;Memoria onde se encontra o valor total a pagar do carrinho atual
    MOV R3, [R4]
    MOV R7, 350
    ADD R3, R7
    MOV R8, 270FH  ;9999 em decimal
    CMP R3, R8
    ;JGT total_invalido_int
    MOV [R4], R3
    MOV R4, 34H ;4 em Ascii
    CALL Adicionar_talao
    JMP opcao_comprar_int

;-----------------------------------------------total_invalido_int ----------------
;total_invalido_int:
```

```
    ;JMP total_invalido
;---------------------------------------------- Opcao_comprar_int----------------------
Opcao_comprar_int2:
    JMP Opcao_comprar_int

;---------------------------------------------- Ligado_e_stockado_int4----------------------
Ligado_e_stockado_int4:
    JMP Ligado_e_stockado_int3

Pagamento_E5:
    ;Vou usar R4 para armazenar o total a pagar
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, continuar_compra ;é preciso alterar para mostrar a estaçao atual
    CALL MostrarDisplay
    MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
    MOV R7, 2020H
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV R8, 4535H
    MOV [R6], R8 ;45 E em Hexadecimal
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7



ciclo_pagamento_5:
    MOV R5, SelecionarOpcao
    MOVB R1, [R5]
    CMP R1, 0
    JZ ciclo_pagamento_5
    MOV R7, Continuar
    CMP R1, R7
    JEQ concluir_pagamento_5
    MOV R7, Voltar
    CMP R1, R7
    JEQ Opcao_comprar_int2
    CALL Rotina_erro_carrinho
```

```
concluir_pagamento_5:
   MOV R5, 342EH ;450 em ASCII
   MOV R10, 350AH
   MOV R4, Total ;Memoria onde se encontra o valor total a pagar do carrinho atual
   MOV R3, [R4]
   MOV R7, 450
   ADD R3, R7
   MOV R8, 270FH  ;9999 em decimal
   CMP R3, R8
   ;JGT total_invalido_int
   MOV [R4], R3
   MOV R4, 35H ;5 em Ascii
   CALL Adicionar_talao
   JMP opcao_comprar_int



Finalizar_compra:
   MOV R4, Total
   MOV R3, [R4]         ;Verificamos se o valor TOTAl é 0, se for vai para carrinho vazio, se
nao vai para o checkout
   CMP R3, 0
   JEQ Carrinho_vazio
   JMP Checkout
Carrinho_vazio:
   CALL LimparDisplay
   CALL Limparperifericos
   CALL Limpar_talao
   MOV R2, carrinho_vazio              ;Mostra tela carrinho vazio
   CALL MostrarDisplay
Ciclo_Carrinho_vazio:
   MOV R5, SelecionarOpcao
   MOVB R1, [R5]
   CMP R1, Continuar
   JEQ Ligado_e_stockado_int4         ;Volta ao inicio
   JMP Ciclo_Carrinho_vazio

Checkout:
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, checkout_menu_sem_saldo       ;Mostra menu checkout
   CALL MostrarDisplay
   MOV R3, Display_primeiro_valor_stock ;mesmo valor do Display
   MOV R4, Total
   MOV R5, [R4] ;total a ser pago
   MOV R0, R5
   CALL HEx_para_dec
```

```
    MOV R1, Conversor_Hex_para_dec
    MOVB R5, [R1]                   ;total convertido
    MOV R6, 2020H
    MOV [R3], R6
    ADD R3, 2
    MOV [R3], R6
    ADD R3, 2
    MOV [R3], R6
    ADD R3, 2
    MOVB [R3], R5
    ADD R3,1
    ADD R1, 1                                   ;Mostra o valor a pagar
    MOVB R5, [R1]
    MOVB [R3], R5
    MOV R0, 2EH
    ADD R3, 1
    MOVB[R3], R0
    ADD R3, 1
    ADD R1, 1
    MOVB R5,[R1]
    MOVB [R3], R5
    ADD R3, 1
    ADD R1, 1
    MOVB R5,[R1]
    MOVB [R3], R5
Ciclo_checkout:
    MOV R7, SelecionarOpcao
    MOVB R8, [R7]
    CMP R8, Continuar
    JEQ Pagar
    CMP R8, Cancelar
    JEQ Cancelar_pagar
    JMP Ciclo_checkout

Pagar:
    CALL inserir_dinheiro               ;chama a rotina de inserir dinheiro
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R7, Total_inserido
    MOV R8, [R7]
    MOV R7, Total
    MOV R9, [R7]                                ;Depois Calcula o troco a dar  de volta
    CMP R8, R9
    JGE Calcular_troco_sem_cartao_int2
    CALL LimparDisplay
    CAll Limparperifericos
    MOV R2, Menu_saldo_ins
    CALL MostrarDisplay
```

```
ciclo_falta_de_dinheiro:
   MOV R0, SelecionarOpcao
   MOVB R1, [R0]
   CMP R1, 1                                   ;Espera input
   JEQ Pagar
   JMP ciclo_falta_de_dinheiro
Cancelar_pagar:
   CALL Limpa_total                    ;Apaga o que era preciso pagar e o talao
   CALL Limpar_talao
   JMP Ligado_e_stockado_int3


;Fazer toda a soma em Hexadecimal, depois fazer um conversor para decimal, os ultimos
dois digitos sao sempre de virgula
;-------------------------------------------------Calcular_troco_sem_cartao_int-------------------------
Calcular_troco_sem_cartao_int2:
   JMP Calcular_troco_sem_cartao_int

;-------------------------------------------------Limpar_talao-----------------------------------------------------
Limpar_talao:
   MOV R0, 9000H ;onde comeca o talao
   MOV R1, 915FH    ;onde termina o talao
   MOV R2, 2020H     ;palavra de caracteres vazio
Ciclo_limpar_talao:
   MOV [R0], R2         ;Limpa dois bytes
   ADD R0, 2                  ;avança 2 bytes
   CMP R0, R1                 ;Compara o inicio do talao com o fim para saber se acabou de
limpar
   JLE Ciclo_limpar_talao    ;Se for menor ou igual ainda nao terminou de limpar logo volta
ao comeco do ciclo
   RET


; ------------------------------------------------Mostrar Menu
Cartao-----------------------------------------------
Opcao_cartao:
   CALL LimparDisplay
   MOV R2, Menu_usarcartao
   CALL MostrarDisplay
   CALL Limparperifericos
CicloCartao:
   MOV R0, SelecionarOpcao
   MOVB R1, [R0]                                 ; Guarda no reg 1 o valor no endereço
da opção guardado no reg 0
   CMP R1, 0
   JEQ CicloCartao                           ; Caso a opção continue igual a 0 continua o
ciclo ate o estado mudar
```

```
   ;TODO ler o input do utilizador..percorrer os PEPE se existir mostrar info se nao existe
mostrar erro
   CMP R1, Cancelar
   JEQ Ligado_e_stockado_int7
   CMP R1, Continuar
   JEQ Verificar_PEPE_introduzido
   CALL Rotina_erro
   JMP CicloCartao
;----------------------Ligado_e_stockado_int7-------------------
Ligado_e_stockado_int7:
   JMP Ligado_e_stockado_int4

Verificar_PEPE_introduzido:
   MOV R3, Introduzir_PEPE  ;periferico de entrada para escrever o Pepe
   MOV R4, [R3]                        ;Valor que o utilizador escreveu
   CALL Limparperifericos
   MOV R5, 0F000H           ;Mascara para isolar o primeiro digito. Se o primeiro digito nao
for 8 ja se sabe que o PEPE é invalido
   AND R5, R4                          ;R5 tem o primeiro digito inserido pelo utilizador
   MOV R7, 8000H
   CMP R5, R7
   JNE PEPE_invalido_int2
   MOVB R6, [R4]                       ;Primeiro byte do espaco de memoria introduzido pelo
utilizador esta em R6
   CMP R6, 1                ;Se nao for 1, O pepe nao é valido já que quando um PEPE é
gerado temos que o primeiro byte da memoria é 1
   JNE PEPE_invalido_int2
PEPE_valido:
   MOV R8, R4               ;copia do Numero do PEPE
   ADD R8, Saldo_PEPE
   MOV R5, [R8]                        ;R5 tem o saldo do PEPE em Hexadecimal
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, Menu_saldoPepe
   CALL MostrarDisplay
   MOV R0, R5               ;R0 tem o saldo do PEPE em Hex
   CALL HEx_para_dec        ;vai converter R0 para decimal, guardando o ascii equivalente
no espaco de memoria conversor_Hex_para_dec
   MOV R5, Conversor_Hex_para_dec ;R5 tem o espaco de memoria onde esta o numero
convertido
   MOV R3, Display_primeiro_valor_stock ;Editar terceira linha do Display
   MOV R9, [R5]             ; R8 tem o valor em ASCII (primeiro byte) do saldo do PEPE
   MOV R2, vazio            ; R2 tem o caracter vazio
   MOV [R3], R9
   ADD R3, 2
   ADD R5, 2           ;2 byte de memoria do valor convertido
   MOVB R9, [R5]            ;R8 tem o segundo byte de valores convertidos
   MOV R2, 2EH              ;R2 é o ponto em ascii
```

```
    MOVB [R3], R2
    ADD R3, 1
    MOVB [R3], R9               ;copiar os dois bytes de uma vez da erro por ser um endereço
impar por isso faz se byte a byte
    ADD R5, 1
    MOVB R9, [R5]
    ADD R3, 1
    MOVB [R3], R9      ;O Saldo ja foi escrito no menu
    JMP Ciclo_Pepe_valido
Ciclo_Pepe_valido:
    MOV R2, SelecionarOpcao
    MOVB R1, [R2]
    CMP R1, 0
    SUB R0, Saldo_PEPE
    JEQ Ciclo_Pepe_valido
    CMP R1, Continuar
    JEQ Comprar_com_cartao
    CMP R1, Recarregar
    JEQ Recarregar_cartao_int
    CMP R1, Cancelar
    JEQ Ligado_e_stockado_int5
    JMP Ciclo_Pepe_valido
Comprar_com_cartao:
    ;R0 continua a ter o saldo do PEPE em Hex
    ;R8 continua a ter o numero do PEPE
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, Menu_comprar
    CALL MostrarDisplay
    MOV R2, R0          ;O R2 passa a ter o saldo do PEPE
    MOV R0, R8          ;O R0 passa a ter o numero do PEPE para podermos reutilizar a
funcao N_PEPE_talao
    CALL N_PEPE_talao
    MOV R0, R2   ;Voltamos a por o saldo do PEPE em R0
CicloComprar_com_cartao:
    MOV R6, SelecionarOpcao
    MOVB R1, [R6]
    CMP R1, 0                   ; Guarda no reg 1 o valor no endereço da opção guardado no
reg 0
    JZ CicloComprar_com_cartao                                  ; Caso a opção continue
igual a 0 continua o ciclo ate o estado mudar
    CMP R1, E2
    JEQ Pagamento_E2_com_cartao
    CMP R1, E3
    JEQ Pagamento_E3_com_cartao
    CMP R1, E4
    JEQ Pagamento_E4_com_cartao_int
    CMP R1, E5
```

```
    JEQ Pagamento_E5_com_cartao_int
    CMP R1, Cancelar
    JEQ Terminar_compra_com_cartao
    JMP CicloComprar_com_cartao
;-------------------------------------------------Recarregar Cartao
int-----------------------------------------------
Recarregar_cartao_int:
    JMP Recarregar_cartao
;-------------------------------------PEPE_invalido_int2--------------------
PEPE_invalido_int2:
    JMP PEPE_invalido_int
    ;-------------------------------------------------------------------
Terminar_compra_com_cartao:                                     ;semelhante ao sem
cartao mas usamos o saldo do PEPE em vez do valor inserido
    MOV R1,Total
    MOV R2, [R1]
    ;R04 é o endereco do PEPE
    MOV R5, R4
    ADD R5, Saldo_PEPE
    MOV R0, [R5]
    CMP R0, R2
    JGE concluir_pagamento_com_cartao
    JMP saldo_insuficiente
concluir_pagamento_com_cartao:
    SUB R0, R2
    MOV [R5], R0
    MOV R2, Menu_compra_cartao
    Call MostrarDisplay
    CALL LimparMoedasTroco
    CALL Limpa_moedas_inseridas  ;Limpamos tudo
    CAll Limpa_total
    CALL Limpa_total_inserido
    CALL Limpa_diferenca
ciclo_concluir_pagamento_cartao:
    MOV R1, SelecionarOpcao
    MOVB R2, [R1]
    CMP R2, 0
    JEQ ciclo_concluir_pagamento_cartao
    CMP R2, Continuar
    JEQ terminar_pagamento_cartao
    JMP ciclo_concluir_pagamento_cartao
terminar_pagamento_cartao:
    CALL LimparMoedasTroco
    CALL Limpa_moedas_inseridas  ;Limpamos tudo
    CAll Limpa_total
    CALL Limpa_total_inserido
    CALL Limpa_diferenca
    JMP Ligado_e_stockado_int4
```

```
saldo_insuficiente:
   CALL LimparDisplay
   CALL Limparperifericos                    ;Se o saldo for insuficiente saltamos
para o caso do saldo insuficiente
   MOV R2, Menu_saldo_ins
   CALL MostrarDisplay
   JMP ciclo_concluir_pagamento_cartao



Recarregar_cartao:
   CALL inserir_dinheiro                ;Usa a mesma rotina inserir dinheiro do compra
   MOV R1, R8   ;copia  do endereco do PEPE
         ;endereço do saldo do PEPE
   MOV R4, [R1] ;Saldo do PEPE em Hex
   MOV R5, Total_inserido
   MOV R6, [R5]   ;Total inserido no R6
   ADD R4, R6
   MOV [R1], R4
   CALL LimparDisplay
   CALL Limparperifericos
   CALL LimparMoedasTroco
   CALL Limpa_moedas_inseridas  ;Limpamos tudo
   CAll Limpa_total
   CALL Limpa_total_inserido
   CALL Limpa_diferenca
   JMP Ligado_e_stockado_int4

;---------------------------------------Ligado_e_stockado_int5 -------------------------
Ligado_e_stockado_int5:
   JMP Ligado_e_stockado_int4

;-------------------------------------------------PEPE invalido int´-----------------------------
PEPE_invalido_int:
   JMP PEPE_invalido


Pagamento_E2_com_cartao:
   ;Vou usar R9 para armazenar o total a pagar
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, continuar_compra ;é preciso alterar para mostrar a estaçao atual
   CALL MostrarDisplay
   MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
   MOV R7, 2020H
   MOV [R6], R7
   ADD R6, 2
   MOV [R6], R7
```

```
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV R8, 4532H
    MOV [R6], R8 ;45 é E em Hexadecimal
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7

ciclo_pagamento_2_com_cartao:
    MOV R5, SelecionarOpcao
    MOVB R1, [R5]
    CMP R1, 0
    JZ ciclo_pagamento_2_com_cartao                    ;Basicamente igual ao sem
cartao
    MOV R7, Continuar
    CMP R1, R7
    JEQ concluir_pagamento_2_com_cartao
    MOV R7, Voltar
    CMP R1, R7
    JEQ Comprar_com_cartao
    JMP ciclo_pagamento_3_com_cartao

concluir_pagamento_2_com_cartao:
    MOV R5, 312EH ;1.50 em ASCII
    MOV R10, 350AH
    MOV R9, Total ;Memoria onde se encontra o valor total a pagar do carrinho atual
    MOV R3, [R9]
    MOV R7, 150
    ADD R3, R7
    ;MOV R8, 270FH  ;9999 em decimal
    ;CMP R3, R8
    ;JGT total_invalido_int
    MOV [R9], R3
    MOV R9, 32H ;2 em Ascii
    CALL Adicionar_talao_cartao
    JMP Comprar_com_cartao

        ;-------------------------------------------------Pagamento E4 com cartao int----------------
Pagamento_E4_com_cartao_int:
        JMP Pagamento_E4_com_cartao
        ;-------------------------------------------------Pagamento E5 com cartao int----------------
Pagamento_E5_com_cartao_int:
```

```
        JMP Pagamento_E5_com_cartao



;-------------------------------------------------Comprar com Cartao int----------------
Comprar_com_cartao_int:
    JMP Comprar_com_cartao

Pagamento_E3_com_cartao:
    ;Vou usar R9 para armazenar o total a pagar
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, continuar_compra ;é preciso alterar para mostrar a estaçao atual
    CALL MostrarDisplay
    MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
    MOV R7, 2020H
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV R8, 4533H
    MOV [R6], R8 ;45 é E em Hexadecimal
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7

ciclo_pagamento_3_com_cartao:
    MOV R5, SelecionarOpcao
    MOVB R1, [R5]
    CMP R1, 0
    JZ ciclo_pagamento_3_com_cartao
    MOV R7, Continuar
    CMP R1, R7
    JEQ concluir_pagamento_3_com_cartao                                    ;Igual as outras
estacoes mudando apenas o valor ASCCi da propria Estacao e o seu valor
    MOV R7, Voltar
    CMP R1, R7
    JEQ Comprar_com_cartao_int
    JMP ciclo_pagamento_3_com_cartao

concluir_pagamento_3_com_cartao:
```

```
    MOV R5, 322EH ;2.50 em ASCII
    MOV R10, 350AH
    MOV R9, Total ;Memoria onde se encontra o valor total a pagar do carrinho atual
    MOV R3, [R9]
    MOV R7, 250                 ; O seu valor em centimos
    ADD R3, R7
    MOV R8, 270FH  ;9999 em decimal
    CMP R3, R8
    ;JGT total_invalido_int
    MOV [R9], R3
    MOV R9, 33H ;3 em Ascii
    CALL Adicionar_talao_cartao
    JMP Comprar_com_cartao

;-------------------------------------------------Calcular_troco_sem_cartao_int--------------------------
Calcular_troco_sem_cartao_int:
    JMP Calcular_troco_sem_cartao

Pagamento_E4_com_cartao:
    ;Vou usar R9 para armazenar o total a pagar
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, continuar_compra ;é preciso alterar para mostrar a estaçao atual
    CALL MostrarDisplay
    MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
    MOV R7, 2020H
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV R8, 4534H
    MOV [R6], R8 ;45 é E em Hexadecimal
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
ciclo_pagamento_4_com_cartao:
    MOV R5, SelecionarOpcao
    MOVB R1, [R5]
    CMP R1, 0
    JZ ciclo_pagamento_4_com_cartao
    MOV R7, Continuar
```

```
    CMP R1, R7
    JEQ concluir_pagamento_4_com_cartao
    CMP R1, Voltar
    JEQ Comprar_com_cartao_int
    JMP ciclo_pagamento_3_com_cartao

concluir_pagamento_4_com_cartao:
    MOV R5, 332EH ;3.50 em ASCII
    MOV R10, 350AH
    MOV R9, Total ;Memoria onde se encontra o valor total a pagar do carrinho atual
    MOV R3, [R9]
    MOV R7, 350
    ADD R3, R7
    MOV R8, 270FH  ;9999 em decimal
    CMP R3, R8
    ;JGT total_invalido_int
    MOV [R9], R3
    MOV R9, 34H ;4 em Ascii
    CALL Adicionar_talao_cartao
    JMP Comprar_com_cartao
;----------------------------------------Comprar com cartao int 2-----------------
Comprar_com_cartao_int2:
        JMP Comprar_com_cartao_int

Pagamento_E5_com_cartao:
    ;Vou usar R9 para armazenar o total a pagar
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, continuar_compra ;é preciso alterar para mostrar a estaçao atual
    CALL MostrarDisplay
    MOV R6, Display_primeiro_valor_stock ;é o mesmo que o do STOCK
    MOV R7, 2020H
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV [R6], R7
    ADD R6, 2
    MOV R8, 4535H
    MOV [R6], R8 ;45 é E em Hexadecimal
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
    ADD R6, 2
    MOV R6, R7
```

```
ciclo_pagamento_5_com_cartao:
    MOV R5, SelecionarOpcao
    MOVB R1, [R5]
    CMP R1, 0
    JZ ciclo_pagamento_5_com_cartao
    MOV R7, Continuar
    CMP R1, R7
    JEQ concluir_pagamento_5_com_cartao
    MOV R7, Voltar
    CMP R1, R7
    JEQ Comprar_com_cartao_int2
    JMP ciclo_pagamento_3_com_cartao

concluir_pagamento_5_com_cartao:
    MOV R5, 342EH ;450 em ASCII
    MOV R10, 350AH
    MOV R9, Total ;Memoria onde se encontra o valor total a pagar do carrinho atual
    MOV R3, [R9]
    MOV R7, 450
    ADD R3, R7
    MOV R8, 270FH  ;9999 em decimal
    CMP R3, R8
    ;JGT total_invalido_int
    MOV [R9], R3
    MOV R9, 35H ;5 em Ascii
    CALL Adicionar_talao_cartao
    JMP Comprar_com_cartao




PEPE_invalido:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, Menu_PEPE_invalido          ;Vem para se o Valor inserido nao é valido
    CALL MostrarDisplay

ciclo_PEPE_invalido:
    MOV R0, SelecionarOpcao
    MOVb R1, [R0]
    CMP R1, 0
    JEQ ciclo_PEPE_invalido          ;Espera input
    CMP R1, Continuar
    JEQ Ligado_e_stockado_int
```

```
    POP R8
    POP R7
    POP R6
    POP R5
    POP R4
    POP R3
    POP R0
    CALL Rotina_erro

;--------------------------------------------------Ligado_e_stockado intermedio (erro de out of bounds)----------------
Ligado_e_stockado_int:
    JMP Ligado_e_stockado

; ------------------------------------------------Rotina password-----------------------------------------------
Menu_Password:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, Menu_pass
    CALL MostrarDisplay
CicloMenu_Password:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]                                   ; Guarda no reg 1 o valor no endereço da opção guardado no reg 0
    CMP R1, 0
    JEQ CicloMenu_Password                              ; Caso a opção continue igual a 0 continua o ciclo ate o estado mudar
    CMP R1,Continuar
    JEQ VerificaPass
    CMP R1, Voltar
    JEQ Ligado_e_stockado_int
    CALL Rotina_erro
    JMP CicloMenu_Password
PassErro:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, Menu_passErrada
    CALL MostrarDisplay
CicloPassErro:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]                                   ; Guarda no reg 1 o valor no endereço da opção guardado no reg 0
    CMP R1, 1           ;Caso nao seja o valor voltar continua no ciclo
    JEQ Ligado_e_stockado_int
    JMP CicloPassErro
VerificaPass:
        MOV R0, password
        MOV R1, PassStock
```

```
        MOV R4, 0
Caracter_a_caracter:
        MOVB R2, [R0]
        MOVB R3, [R1]                   ;Compara caracter a caracter o input do
utilizador e a password
        CMP R2, R3
        JNE PassErro
        ADD R0,1
        ADD R1,1
        ADD R4,1
        CMP R4, 4
        JNE Caracter_a_caracter
        JMP Opcao_stock1
;-------------------------------------------------Ligado_e_stockado_int2-------------------------------------
-----
Ligado_e_stockado_int2:
   JMP Ligado_e_stockado_int
; -----------------------------------------------Mostrar Menus
Stock-----------------------------------------------
Opcao_stock1:
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, Menu_Stock1
   CALL MostrarDisplay
   MOV R1, Display_primeiro_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, M_10
   MOV R0, [R3]
   CALL Alterar_stock
   MOV R1, Display_segundo_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, M_20
   MOV R0, [R3]
   CALL Alterar_stock
   CALL Limparperifericos
CicloStock1:
   MOV R0, SelecionarOpcao
   MOVB R1, [R0]                        ; Guarda no reg 1 o valor no endereço
da opção guardado no reg 0
   CMP R1, 0
   JEQ CicloStock1                      ; Caso a opção continue igual a 0 continua o
ciclo ate o estado mudar
   CMP R1, Voltar
   JEQ Ligado_e_stockado_int
   CMP R1, 1
   JEQ Opcao_stock2
   CALL Rotina_erro
   JMP CicloStock1
```

```
Opcao_stock2:
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, Menu_Stock2
   CALL MostrarDisplay
   MOV R1, Display_primeiro_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, M_50
   MOV R0, [R3]
   CALL Alterar_stock
   MOV R1, Display_segundo_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, M_01
   MOV R0, [R3]
   CALL Alterar_stock
   CALL Limparperifericos
CicloStock2:
   MOV R0, SelecionarOpcao
   MOVB R1, [R0]                              ; Guarda no reg 1 o valor no endereço
da opção guardado no reg 0
   CMP R1, 0
   JEQ CicloStock2                            ; Caso a opção continue igual a 0 continua o
ciclo ate o estado mudar
   CMP R1, Voltar
   JEQ Ligado_e_stockado_int
   CMP R1, 1
   JEQ Opcao_stock3
   CALL Rotina_erro
   JMP CicloStock2
Opcao_stock3:
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, Menu_Stock3
   CALL MostrarDisplay
   MOV R1, Display_primeiro_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, M_02
   MOV R0, [R3]
   CALL Alterar_stock
   MOV R1, Display_segundo_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, N_05
   MOV R0, [R3]
   CALL Alterar_stock
   CALL Limparperifericos
CicloStock3:
   MOV R0, SelecionarOpcao
```

```
   MOVB R1, [R0]                                ; Guarda no reg 1 o valor no endereço
da opção guardado no reg 0
   CMP R1, 0
   JEQ CicloStock3                              ; Caso a opção continue igual a 0 continua o
ciclo ate o estado mudar
   CMP R1, Voltar
   JEQ Ligado_e_stockado_int
   CMP R1, 1
   JEQ Opcao_stock4
   CALL Rotina_erro
   JMP CicloStock3
Opcao_stock4:
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, Menu_Stock4
   CALL MostrarDisplay
   MOV R1, Display_primeiro_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, N_10
   MOV R0, [R3]
   CALL Alterar_stock
   MOV R1, Display_segundo_valor_stock ;Vamos alterar o valor das moedas conforme o
stock
   MOV R3, N_20
   MOV R0, [R3]
   CALL Alterar_stock
   CALL Limparperifericos
CicloStock4:
   MOV R0, SelecionarOpcao
   MOVB R1, [R0]                                ; Guarda no reg 1 o valor no endereço
da opção guardado no reg 0
   CMP R1, 0
   JEQ CicloStock4                              ; Caso a opção continue igual a 0 continua o
ciclo ate o estado mudar
   CMP R1, Voltar
   JEQ Ligado_e_stockado_int2
   CMP R1, 1
   JEQ Opcao_stock1
   CALL Rotina_erro
   JMP CicloStock4
;-------------------------------------------------Inserir
Dinheiro-------------------------------------------------------------------------
inserir_dinheiro:
   PUSH R0
   PUSH R1
   PUSH R3
   PUSH R4
   PUSH R5
```

```
colocar_mais_moedas:
ciclo_inserir_dinheiro:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R2, InserirMoedas           ;Mostra o Menu de inserir moedas
    CALL MostrarDisplay
    MOV R0, SelecionarOpcao
ciclo_inserir_dinheiro_pos_limpar:
    MOVB R1, [R0]
    CMP R1, 0                                   ;Verifica qual valor sera inserido
    JEQ ciclo_inserir_dinheiro_pos_limpar
    MOV R3, 1H
    CMP R1, R3
    JEQ inseriu_M10
    MOV R3, 2H
    CMP R1, R3
    JEQ inseriu_M20
    MOV R3, 3H
    CMP R1, R3
    JEQ inseriu_M50
    MOV R3, 4H
    CMP R1, R3
    JEQ inseriu_M01
    MOV R3, Cancelar
    CMP R1, R3
    JEQ Cancelar_carregamento_int
    MOV R3, 6H
    CMP R1, R3
    JEQ inseriu_M02
    MOV R3, 7H
    CMP R1, R3
    JEQ inseriu_N5_int
    MOV R3, 8H
    CMP R1, R3
    JEQ inseriu_N10_int
    MOV R3, 9H
    CMP R1, R3
    JEQ inseriu_N20_int
    MOV R3, 0AH
    CMP R1, R3
    JEQ Confirmar_carregamento_int
    JMP ciclo_inserir_dinheiro_pos_limpar
inseriu_M10:                                    ;Para inserir soma se o
valor ao total inserido e soma se em um a moeda ou nota inserida para posteriormente ir
para o stock
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R3, Total_inserido          ;Se inseriu _10 vem para ca
```

```
    MOV R4, [R3]
    MOV R6, M_10_HEX
    ADD R4, R6                                  ;soma o valor da moeda ao
total_inserido em HEX
    MOV [R3], R4
    MOV R4, M_10_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Moeda10                     ;Mostra menu de confirmacao do que inseriu
    CALL MostrarDisplay
ciclo_inseriu_M10:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0                                   ;Espera input
    JEQ ciclo_inseriu_M10
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro
    JMP ciclo_inseriu_M10
inseriu_N5_int:
    JMP inseriu_N5
inseriu_N20_int:
    JMP inseriu_N20
inseriu_M20:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R3, Total_inserido                  ;Igual para todas as outras moedas e notas so muda o
valor a somar ao total_inserido
    MOV R4, [R3]
    MOV R6, M_20_HEX
    ADD R4, R6
    MOV [R3], R4
    MOV R4, M_20_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Moeda20
    CALL MostrarDisplay
ciclo_inseriu_M20:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0
    JEQ ciclo_inseriu_M20
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro
    JMP ciclo_inseriu_M20
Cancelar_carregamento_int:
    JMP Cancelar_carregamento
```

```
inseriu_M50:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R3, Total_inserido
    MOV R4, [R3]
    MOV R6, M_50_HEX
    ADD R4, R6
    MOV [R3], R4
    MOV R4, M_50_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Moeda50
    CALL MostrarDisplay
ciclo_inseriu_M50:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0
    JEQ ciclo_inseriu_M50
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro
    JMP ciclo_inseriu_M50
ciclo_inserir_dinheiro_int:
    JMP ciclo_inserir_dinheiro
inseriu_M01:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R3, Total_inserido
    MOV R4, [R3]
    MOV R6, M_01_HEX
    ADD R4, R6
    MOV [R3], R4
    MOV R4, M_1_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Moeda1
    CALL MostrarDisplay
ciclo_inseriu_M01:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0
    JEQ ciclo_inseriu_M01
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro_int
    JMP ciclo_inseriu_M01
inseriu_M02:
    CALL LimparDisplay
```

```
    CALL Limparperifericos
    MOV R3, Total_inserido
    MOV R4, [R3]
    MOV R6, M_02_HEX
    ADD R4, R6
    MOV [R3], R4
    MOV R4, M_2_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Moeda2
    CALL MostrarDisplay
ciclo_inseriu_M02:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0
    JEQ ciclo_inseriu_M02
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro_int
    JMP ciclo_inseriu_M02
inseriu_N10_int:
    JMP inseriu_N10
Confirmar_carregamento_int:
    JMP Confirmar_carregamento
inseriu_N5:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R3, Total_inserido
    MOV R4, [R3]
    MOV R6, N_5_HEX
    ADD R4, R6
    MOV [R3], R4
    MOV R4, N_5_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Nota5
    CALL MostrarDisplay

ciclo_inseriu_N5:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0
    JEQ ciclo_inseriu_N5
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro_int
    JMP ciclo_inseriu_N5
inseriu_N10:
```

```
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R3, Total_inserido
    MOV R4, [R3]
    MOV R6, N_10_HEX
    ADD R4, R6
    MOV [R3], R4
    MOV R4, N_10_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Nota10
    CALL MostrarDisplay
ciclo_inseriu_N10:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0
    JEQ ciclo_inseriu_N10
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro_int
    JMP ciclo_inseriu_N10
inseriu_N20:
    CALL LimparDisplay
    CALL Limparperifericos
    MOV R3, Total_inserido
    MOV R4, [R3]
    MOV R6, N_20_HEX
    ADD R4, R6
    MOV [R3], R4
    MOV R4, N_5_inseridas
    MOV R5, [R4]
    ADD R5, 1
    MOV [R4], R5
    MOV R2, Inseriu_Nota20
    CALL MostrarDisplay
ciclo_inseriu_N20:
    MOV R0, SelecionarOpcao
    MOVB R1, [R0]
    CMP R1, 0
    JEQ ciclo_inseriu_N20
    CMP R1, Continuar
    JEQ ciclo_inserir_dinheiro_int
    JMP ciclo_inseriu_N20
Confirmar_carregamento:
    ;Neste momento ainda nao foram atualizadas as moedas do stock. Isso sera feito depois
de ver se o dinheiro inserido é suficiente para pagar
    POP R5
    POP R4
```

```
   POP R3
   POP R1
   POP R0
   RET
Cancelar_carregamento:
   CALL Limpa_total_inserido
   CALL Limpa_moedas_inseridas
   CALL Limpa_total                    ;Se cancelar devolve tudo
   CALL Ligado_e_stockado
   POP R5
   POP R4
   POP R3
   POP R1
   POP R0
   RET
;--------------------------------------------------Adicionar moedas inseridas ao
Stock----------------------------------------------
Adicionar_moedas_stock:
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        MOV R0, M_10_inseridas          ;Adicionar as moedas ou notas inseridas no
Stock
        MOV R1, [R0]
        MOV R2, M_10
        MOV R3, [R2]                    ;So pega nas moedas e notas inseridas e soma
aos do stock e por fim apaga as inseridsa, para nao ficarem duplicadas
        ADD R3, R1
        MOV [R2], R3
        MOV R0, M_20_inseridas
        MOV R1, [R0]
        MOV R2, M_20
        MOV R3, [R2]
        ADD R3, R1
        MOV [R2], R3
        MOV R0, M_50_inseridas
        MOV R1, [R0]
        MOV R2, M_50
        MOV R3, [R2]
        ADD R3, R1
        MOV [R2], R3
        MOV R0, M_1_inseridas
        MOV R1, [R0]
        MOV R2, M_01
        MOV R3, [R2]
        ADD R3, R1
        MOV [R2], R3
```

```
        MOV R0, M_2_inseridas
        MOV R1, [R0]
        MOV R2, M_02
        MOV R3, [R2]
        ADD R3, R1
        MOV [R2], R3
        MOV R0, N_5_inseridas
        MOV R1, [R0]
        MOV R2, N_05
        MOV R3, [R2]
        ADD R3, R1
        MOV [R2], R3
        MOV R0, N_10_inseridas
        MOV R1, [R0]
        MOV R2, N_10
        MOV R3, [R2]
        ADD R3, R1
        MOV [R2], R3
        MOV R0, N_20_inseridas
        MOV R1, [R0]
        MOV R2, N_20
        MOV R3, [R2]
        ADD R3, R1
        MOV [R2], R3
        CALL Limpa_moedas_inseridas
        POP R3
        POP R2
        POP R1
        POP R0
        RET
;-----------------------------------------------------Calcular troco pago com
moedas-------------------------------------------------------------
Calcular_troco_sem_cartao:
    ;So entra nesta rotina se o Total_inserido for maior Total
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    PUSH R10
    CALL Adicionar_moedas_stock      ;Adiciona as moedas ao stock
    MOV R1, Total
    MOV R2, Total_inserido
    MOV R3, Diferenca_troco
```

```
    MOV R4, [R1]
    MOV R5, [R2]
    ;R4 = Total a pagar
    ;R5 = Total_inserido
    MOV R6, R5   ;Copia do total inserido no R6
    SUB R6, R4  ;R6 é o valor da diferenca a pagar ao cliente
    MOV [R3], R6
Subtrair_troco:
    CMP R6, 0
    JEQ dar_troco
    MOV R7, N_20_HEX
    CMP R6, R7                ;vou comparar o valor do troco a dar com as notas mais
valiosas e tentar da-las se tiverem um valor maior que o do troco
    JGE Dar_N_20_int                    ;Se tiverem dou a nota e diminuo do troco e
volto a repetir as comparacoes
SEM_N_20:                               ;Se nao houver notas de um determinado valor
ou for muito para o troco desejado passo pra proxima nota ou moeda
    MOV R7, N_10_HEX
    CMP R6, R7
    JGE Dar_N_10_int
SEM_N_10:
    MOV R7, N_5_HEX
    CMP R6, R7
    JGE Dar_N_5_int
SEM_N_5:
    MOV R7, M_02_HEX
    CMP R6, R7
    JGE Dar_M_2_int
SEM_M_2:
    MOV R7, M_01_HEX
    CMP R6, R7
    JGE Dar_M_1_int
SEM_M_1:
    MOV R7, M_50_HEX
    CMP R6, R7
    JGE Dar_M_50_int
SEM_M_50:
    MOV R7, M_20_HEX
    CMP R6, R7
    JGE Dar_M_20_int
SEM_M_20:
    MOV R7, M_10_HEX
    CMP R6, R7
    JGE Dar_M_10_int
Dar_N_20_int:
JMP Dar_N_20
Dar_N_10_int:
JMP Dar_N_10
```

```
Dar_N_5_int:
JMP Dar_N_5



Dar_M_2_int:
JMP Dar_M_2

Dar_M_1_int:
JMP Dar_M_1

Dar_M_50_int:
JMP Dar_M_50

Dar_M_20_int:
JMP Dar_M_20

Dar_M_10_int:
JMP Dar_M_10
SEM_TROCO:                  ;Caso nao haja troco mostra outro menu a informar que nao
ha troco para o valor para contactar um responsavel
   CALL LimparDisplay
   CALL Limparperifericos
   MOV R2, Menu_SEM_troco
   CALL MostrarDisplay
   MOV R2, 00A0H
   MOV R0, R6
   CALL HEx_para_dec
   MOV R0, Conversor_Hex_para_dec
   MOV R3, [R0]
   MOV [R2], R3
ciclo_sem_troco:
   MOV R0, SelecionarOpcao
   MOV R1, [R0]                     ;espera input
   CMP R1, 1
   JEQ voltar_ao_inicio
   JMP ciclo_sem_troco
voltar_ao_inicio:
   CALL Limpa_total
   CALL Limpa_total_inserido
   CALL LimparMoedasTroco
   CALL Limpa_diferenca                ;limpa tudo e volta ao inicio
   CALL Limpa_moedas_inseridas
   CALL Ligado_e_stockado
dar_troco:
   MOV R2, Menu_troco_a_dar
   CALL MostrarDisplay
```

```
    ;Mostrar moedas de 10 a dar de troco
    MOV R2, M_10_troco
    MOV R0, [R2]
    CALL HEx_para_dec
    MOV R3, Conversor_Hex_para_dec                ;QuANDO TERMINAR DE CALCULAR o
TROCO as moedas de troco (ex: M_10_troco) tem as moedas a dar de troco
    ADD R3, 2                                                                ;COnverte Para
decimal e mostra num MENU
    MOVB R2, [R3]
    MOV R0, DisplayM10
    MOVB [R0], R2
    ADD R3,1
    MOVB R2, [R3]
    ADD R0, 1
    MOVB [R0], R2
    ;--------------
    MOV R2, M_20_troco
    MOV R0, [R2]
    CALL HEx_para_dec
    MOV R3, Conversor_Hex_para_dec
    ADD R3, 2
    MOVB R2, [R3]
    MOV R0, DisplayM20
    MOVB [R0], R2
    ADD R3,1
    MOVB R2, [R3]
    ADD R0, 1
    MOVB [R0], R2
    ;--------------
    MOV R2, M_50_troco
    MOV R0, [R2]
    CALL HEx_para_dec
    MOV R3, Conversor_Hex_para_dec
    ADD R3, 2
    MOVB R2, [R3]
    MOV R0, DisplayM50
    MOVB [R0], R2
    ADD R3,1
    MOVB R2, [R3]
    ADD R0, 1
    MOVB [R0], R2
    ;--------------
    MOV R2, M_1_troco
    MOV R0, [R2]
    CALL HEx_para_dec
    MOV R3, Conversor_Hex_para_dec
    ADD R3, 2
    MOVB R2, [R3]
```

```
MOV R0, DisplayM1
MOVB [R0], R2
ADD R3,1
MOVB R2, [R3]
ADD R0, 1
MOVB [R0], R2
;--------------
MOV R2, M_2_troco
MOV R0, [R2]
CALL HEx_para_dec
MOV R3, Conversor_Hex_para_dec
ADD R3, 2
MOVB R2, [R3]
MOV R0, DisplayM2
MOVB [R0], R2
ADD R3,1
MOVB R2, [R3]
ADD R0, 1
MOVB [R0], R2
;--------------
MOV R2, N_5_troco
MOV R0, [R2]
CALL HEx_para_dec
MOV R3, Conversor_Hex_para_dec
ADD R3, 2
MOVB R2, [R3]
MOV R0, DisplayN5
MOVB [R0], R2
ADD R3,1
MOVB R2, [R3]
ADD R0, 1
MOVB [R0], R2
;--------------
    MOV R2, N_10_troco
MOV R0, [R2]
CALL HEx_para_dec
MOV R3, Conversor_Hex_para_dec
ADD R3, 2
MOVB R2, [R3]
MOV R0, DisplayN10
MOVB [R0], R2
ADD R3,1
MOVB R2, [R3]
ADD R0, 1
MOVB [R0], R2
;--------------
    MOV R2, N_20_troco
MOV R0, [R2]
```

```
    CALL HEx_para_dec
    MOV R3, Conversor_Hex_para_dec
    ADD R3, 2
    MOVB R2, [R3]
    MOV R0, DisplayN20
    MOVB [R0], R2
    ADD R3,1
    MOVB R2, [R3]
    ADD R0, 1
    MOVB [R0], R2
    ;--------------
    CALL Limpa_total
    CALL Limpa_total_inserido
    CALL LimparMoedasTroco
    CALL Limpa_diferenca
    CALL Limpa_moedas_inseridas
ciclo_sair_troco:
        MOV R0, SelecionarOpcao
        MOVB R1, [R0]
        CMP R1, Continuar
        JEQ Ligado_e_stockado_int6          ;Espera input para continuar
        JMP ciclo_sair_troco

Ligado_e_stockado_int6:
JMP Ligado_e_stockado_int5

Dar_N_20:
    MOV R1, N_20 ;Verifica se ha pelo menos um em stock
    MOV R8, [R1]
    CMP R8 , 1
    JGE sair_N_20  ;Se houver a maquina deixa sair esse valor
    JMP SEM_N_20      ;Se nao passa pro proximo
sair_N_20:
    MOV R9, N_20_troco          ;Adiciona um deste valor ao valor dado de troco
    MOV R10, [R9]
    ADD R10, 1
    MOV [R9], R10
    SUB R8, 1                   ;E subtrai do stock
    MOV [R1], R8
    MOV R8, N_20_HEX
    SUB R6, R8                  ; e subtrai do que ainda falta pagar
    JMP Subtrair_troco
Dar_N_10:
    MOV R1, N_10 ;Verifica se ha pelo menos um em stock
    MOV R8, [R1]
    CMP R8 , 1
    JGE sair_N_10  ;Se houver a maquina deixa sair esse valor
    JMP SEM_N_10      ;Se nao passa pro proximo
```

```
sair_N_10:
   MOV R9, N_10_troco         ;Adiciona um deste valor ao valor dado de troco
   MOV R10, [R9]
   ADD R10, 1
   MOV [R9], R10
   SUB R8, 1                  ;E subtrai do stock
   MOV [R1], R8
   MOV R8, N_10_HEX
   SUB R6, R8
   JMP Subtrair_troco
Dar_N_5:
   MOV R1, N_05 ;Verifica se ha pelo menos um em stock
   MOV R8, [R1]
   CMP R8 , 1
   JGE sair_N_5  ;Se houver a maquina deixa sair esse valor
   JMP SEM_N_5       ;Se nao passa pro proximo
sair_N_5:
   MOV R9, N_5_troco          ;Adiciona um deste valor ao valor dado de troco
   MOV R10, [R9]
   ADD R10, 1
   MOV [R9], R10
   SUB R8, 1                  ;E subtrai do stock
   MOV [R1], R8
         MOV R8, N_5_HEX
   SUB R6, R8
   JMP Subtrair_troco
Dar_M_2:
   MOV R1, M_02 ;Verifica se ha pelo menos um em stock
   MOV R8, [R1]
   CMP R8 , 1
   JGE sair_M_2  ;Se houver a maquina deixa sair esse valor
   JMP SEM_M_2       ;Se nao passa pro proximo
sair_M_2:
   MOV R9, M_2_troco          ;Adiciona um deste valor ao valor dado de troco
   MOV R10, [R9]
   ADD R10, 1
   MOV [R9], R10
   SUB R8, 1                  ;E subtrai do stock
   MOV [R1], R8
         MOV R8, M_02_HEX
   SUB R6, R8
   JMP Subtrair_troco
Dar_M_1:
   MOV R1, M_01 ;Verifica se ha pelo menos um em stock
   MOV R8, [R1]
   CMP R8 , 1
   JGE sair_M_1  ;Se houver a maquina deixa sair esse valor
   JMP SEM_M_1       ;Se nao passa pro proximo
```

```
sair_M_1:
   MOV R9, M_1_troco          ;Adiciona um deste valor ao valor dado de troco
   MOV R10, [R9]
   ADD R10, 1
   MOV [R9], R10
   SUB R8, 1                  ;E subtrai do stock
   MOV [R1], R8
   MOV R8, M_01_HEX
   SUB R6, R8
   JMP Subtrair_troco
Dar_M_50:
   MOV R1, M_50;Verifica se ha pelo menos um em stock
   MOV R8, [R1]
   CMP R8 , 1
   JGE sair_M_50  ;Se houver a maquina deixa sair esse valor
   JMP SEM_M_50     ;Se nao passa pro proximo
sair_M_50:
   MOV R9, M_50_troco         ;Adiciona um deste valor ao valor dado de troco
   MOV R10, [R9]
   ADD R10, 1
   MOV [R9], R10
   SUB R8, 1                  ;E subtrai do stock
   MOV [R1], R8
   MOV R8, M_50_HEX
   SUB R6, R8
   JMP Subtrair_troco
Dar_M_20:
   MOV R1, M_20 ;Verifica se ha pelo menos um em stock
   MOV R8, [R1]
   CMP R8 , 1
   JGE sair_M_20  ;Se houver a maquina deixa sair esse valor
   JMP SEM_M_20     ;Se nao passa pro proximo
sair_M_20:
   MOV R9, M_20_troco         ;Adiciona um deste valor ao valor dado de troco
   MOV R10, [R9]
   ADD R10, 1
   MOV [R9], R10
   SUB R8, 1                  ;E subtrai do stock
   MOV [R1], R8
   MOV R8, M_20_HEX
   SUB R6, R8
   JMP Subtrair_troco
Dar_M_10:
   MOV R1, M_10 ;Verifica se ha pelo menos um em stock
   MOV R8, [R1]
   CMP R8 , 1
   JGE sair_M_10  ;Se houver a maquina deixa sair esse valor
   JMP SEM_TROCO              ;Se nao passa pro proximo
```

```
sair_M_10:
   MOV R9, M_10_troco        ;Adiciona um deste valor ao valor dado de troco
   MOV R10, [R9]
   ADD R10, 1
   MOV [R9], R10
   SUB R8, 1                 ;E subtrai do stock
   MOV [R1], R8
   MOV R8, M_10_HEX
   SUB R6, R8
   JMP Subtrair_troco
;-----------------------------------------------------Alterar moedas conforme
Stock-----------------------------------------------
Alterar_stock:
   PUSH R2
   PUSH R3
   PUSH R4
   CALL HEx_para_dec    ;Converte o R0 dado e coloca no espaco de memoria proprio
   MOV R2, Conversor_Hex_para_dec
   MOV R3, [R2]         ; R3 contem o valor convertido (primeiros 2bytes)
   MOV R4, 2020H        ;Palavra vazia
   MOV [R1], R4
   ADD R1, 2            ;AVANca 2 bytes
   MOV [R1], R4
   ADD R1, 2
   MOV [R1], R4
   ADD R1, 2
   MOV [R1], R3
   ADD R1, 2
   ADD R2, 2            ;Avanco 2 na memoria tbm
   MOV R3, [R2]
   MOV [R1], R3
   ADD R1, 2
   MOV [R1], R4
   ADD R1, 2            ;AVANca 2 bytes
   MOV [R1], R4
   ADD R1, 2
   MOV [R1], R4
   POP R4
   POP R3
   POP R2
   RET
;-----------------------------------------------------Definir stock
padrao----------------------------------------------
Definir_stock_padrao:
        PUSH R0
        PUSH R1
        MOV R0, M_10
```

```
        MOV R1, Stock_padrao                    ;Copia para o Stock o valor padrao de
moedas e notas
        MOV [R0], R1
        MOV R0, M_20
        MOV [R0], R1
        MOV R0, M_50
        MOV [R0], R1
        MOV R0, M_01
        MOV [R0], R1
        MOV R0, M_02
        MOV [R0], R1
        MOV R0, N_05
        MOV [R0], R1
        MOV R0, N_10
        MOV [R0], R1
        MOV R0, N_20
        MOV [R0], R1
        POP R1
        POP R0
        CALL Ligado_e_stockado_int7

; ------------------------------------------------------Mostrar
Display----------------------------------------------
MostrarDisplay:
   PUSH R0
   PUSH R1 ; R2 guarda sempre o Menu que queremos mostar logo nao precisa de push
   PUSH R3 ; Usamos o Push para que os valores que tivessem anteriormente nos registos
nao interfiram
   MOV R0, Display
   MOV R1, Fim_Display
Ciclo:
   MOV R3, [R2] ; Guarda em R3 o Menu no endereço de R2 (Menu_inicial por exemplo)
   MOV [R0], R3
   ADD R2, 2 ; Avançamos de 2 em 2 posições tanto onde se escreve como onde se le
   ADD R0, 2 ; Aqui escreve-se
   CMP R0, R1 ; Se o endereço de R0 for igual ao de R1(fim do display) chegou ao fim
   JLE Ciclo
   POP R3
   POP R1
   POP R0
   RET ; Retorna

; ------------------------------------------------------Limpar
Perifericos----------------------------------------------
Limparperifericos:
   PUSH R0
   PUSH R1
   PUSH R2
```

```
    PUSH R3
    PUSH R4
    PUSH R5
    MOV R0, ON_OFF              ;Move byte a byte e muda para o caracter vazio
    MOV R1, SelecionarOpcao
    MOV R2, password
    MOV R3, 0
    MOV R4, 3FH
    MOV R5, Introduzir_PEPE
    MOVB [R0], R3 ; MOVER UM BYTE é SEMPRE COM REGISTOS E NAO CONSTANTES
    MOVB [R1], R3
    MOVB [R2], R3
    MOVB [R5], R3
    ADD R5, 1
    MOVB [R5], R3
    CALL ciclo_limpar_password
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET
ciclo_limpar_password:
    ADD R2, 1
    MOVB [R2], R3
    CMP R2, R4
    JNE ciclo_limpar_password
    RET
;--------------------------------------------------------------------Limpa diferenca----------
Limpa_diferenca:
    PUSH R0
    PUSH R1
;Mete a diferença (troco) a 0s
    MOV R0, Diferenca_troco
    MOV R1, 0000H
    MOV [R0], R1
    POP R1
    POP R0
    RET

; ----------------------------------------------------LIMPAR DISPLAY
-----------------------------------------------------
LimparDisplay:
    PUSH R0
    PUSH R1
    PUSH R2 ; Usamos o Push para que os valores que tivessem anteriormente nos registos
nao interfiram
```

```
    MOV R0, Display
    MOV R1, Fim_Display
CicloLimpa:
    MOV R2, vazio ; Copia o caracter vazio para R2
    MovB [R0], R2 ; Como so copiamos um caracter usamos MOVB
    ADD R0, 1
    CMP R0, R1 ; Se o endereço de R0 for igual ao de R1(fim do display) chegou ao fim
    JLE CicloLimpa
    POP R2
    POP R1
    POP R0
    RET ; Retorna

;-----------------------------------------------------------Limpar Moedas de Troco----------
LimparMoedasTroco:
    PUSH R1
    PUSH R2
    MOV R2, 0000H
    MOV R1, M_10_troco              ;endereco das moedas de troco
    MOV [R1], R2                            ;pomos a 0 cada uma
    MOV R1, M_20_troco
    MOV [R1], R2
    MOV R1, M_50_troco
    MOV [R1], R2
    MOV R1, M_1_troco
    MOV [R1], R2
    MOV R1, M_2_troco
    MOV [R1], R2
    MOV R1, N_5_troco
    MOV [R1], R2
    MOV R1, N_10_troco
    MOV [R1], R2
    MOV R1, N_20_troco
    MOV [R1], R2
    POP R2
    POP R1
    RET



PLACE 7980H
PassStock: STRING "PaRp"  ;Password para o stock

place 8000H
PEPEs:
Table 200 ; 200 espaços para Pepes gerados Cada Pepe irá comecar com o valor 8 e os
proximos 3 digitos serão os gerados por cada pepe
```

place 9000H
Talao:
Table 21;20 espacos para guardar as estaçoes que foram compradas nesta sessão, para no fim imprimir o talao. O primeiro espaco sera o id do PEPE usado para comprar os bilhetes