Arquitetura de Computadores

Relatório do Trabalho Prático 2

Uma imagem com Gráficos, captura de ecrã, Tipo de letra, design gráfico

Descrição gerada automaticamente

Docentes:

Dino Vasconcelos

Sofia Inácio

Dionísio Barros

Pedro Camacho

Alunos:

André Caires nº 2076620

Afonso Nunes nº 2078821

# Índice

[1 Índice 2](#_Toc165037855)

[2 Introdução 3](#_Toc165037856)

[3 Desenvolvimento 3](#_Toc165037857)

[3.1 Periféricos 3](#_Toc165037858)

[3.1.1 Periféricos de Entrada 3](#_Toc165037859)

[3.1.2 Periférico de Saída 3](#_Toc165037860)

[3.2 Menu Principal 4](#_Toc165037861)

[3.3 Menu de Compra 4](#_Toc165037862)

[3.4 Menu de Pagamento 4](#_Toc165037863)

[3.5 Menu Talão 4](#_Toc165037864)

[3.5.1 Guardando dados do cartão 5](#_Toc165037865)

[3.6 Menu Cartão 5](#_Toc165037866)

[3.7 Menu Stock 5](#_Toc165037867)

[4 Discussão de Resultados 5](#_Toc165037868)

[5 Conclusão 5](#_Toc165037869)

[6 Bibliografia 6](#_Toc165037870)

[7 Anexo A (Fluxogramas) 7](#_Toc165037871)

[8 Anexo B (Funções Extra) 24](#_Toc165037872)

[9 Anexo C (Código Assembly) 24](#_Toc165037873)

# Introdução

Para este trabalho prático foi estabelecido como meta desenvolver um programa que gerasse bilhetes, designados de PEPE’s, de código único quando efetuada pelo menos uma compra de uma viagem de metro e que fosse permitido carregamentos monetários para poderem ser reutilizados para comprar uma ou mais viagens de metro em linguagem Assembly para o processador PEPE.

# Desenvolvimento

Antes do desenvolvimento do programa foram feitos fluxogramas de forma a representar o comportamento desejado e a facilitar a compreensão do mesmo. Após os fluxogramas, foram definidos os periféricos, constantes e interfaces (menus) que seriam necessárias para o funcionamento do programa.

## Periféricos

Os periféricos são os responsáveis pela comunicação entre o utilizador e o programa.

### Periféricos de Entrada

No enunciado trabalho estava explicito que um sistema de seleção deveria ser efetuado através de um único periférico que foi definido para a posição 1B0H, sendo que para ligar, selecionar uma opção e sair de uma mensagem de erro o utilizador tem de introduzir algum valor nessa posição da memória. Quanto à introdução de códigos foi dada escolha livre sendo que a introdução do código PEPE foi definida na posição 1C0H. Foi também criada uma palavra-passe para entrar no stock, onde a posição onde o utilizador deve introduzi-la foi definida na posição 1D0H.

### Periférico de Saída

No enunciado do trabalho que o display (periférico de saída) devia ter uma dimensão de 7x16 (7 linhas 16 caracteres - bytes) que foi definido entre os endereços 200H e 26FH

## Menu Principal

Para aceder ao menu principal o utilizador primeiro tem que introduzir o valor 1 no PEN de forma a “ligar” sendo depois fornecidas as opções que o utilizador escolher. Caso o utilizador queira realizar uma compra, ele deve introduzir o valor 1 no PEN, sendo depois redirecionado para a rotina que vai ser responsável por realizar a compra de um bilhete. Caso o utilizador queira aceder ao cartão, deve introduzir o valor 2 no PEN, sendo depois redirecionado para a rotina que vai ser responsável por analisar o código do PEPE introduzido. Caso o utilizador queira ver o stock, deve introduzir o valor 3 no PEN, sendo redirecionado para a rotina que é responsável por verificar o código de acesso ao stock.

## Menu de Compra

Quando o utilizador acede ao menu de compra é-lhe disponibilizado quatro opções de estações para comprar um bilhete. Sempre que este escolhe uma opção também é-lhe dada a opção de decidir se quer adicionar mais uma estação à mesma compra. Quando o utilizador já está satisfeito com a seleção de estações ele é redirecionado para o menu de pagamento.

## Menu de Pagamento

Depois de estar satisfeito com a seleção das estações o utilizador é direcionado para os menus de pagamento onde ele pode escolher entre introduzir moedas de 0,10€, 0,20€, 0,50€, 1€, 2€, 5€, 10€ e 20€ através de três menus. Quando o valor introduzido for igual ou superior ao valor do bilhete o programa passa automaticamente para o talão.

## Menu Talão

Após concluir a compra, aparece no display um talão com o troco e o código único do PEPE gerado. Para sair do talão o utilizador tem de colocar o valor 1 no PEN e ao fazê-lo é criado um cartão cujo código é o PEPE gerado e é adicionado como saldo desse cartão o preço do bilhete da primeira viagem, pois a primeira viagem é grátis.

### Guardando dados do cartão

Após o utilizador introduzir o valor 1 no PEN, antes de passar para o menu principal, o PEPE e o saldo são guardados numa “base de dados” para depois ser acedido quando o utilizador tentar utilizar o cartão

## Menu Cartão

Quando o utilizador acede ao menu do cartão é-lhe pedido para introduzir um código sendo este depois analisa o código introduzido com todos os PEPE’s gerados procurando um compatível para aceder a mais opções. Caso o código que o utilizador introduzir não exista é mostrada uma mensagem de erro em que para sair dessa mensagem o utilizador deve introduzir o valor 1 no PEPE. Caso o código introduzido exista o utilizador tem acesso às funções de comprar e recarregar onde ele pode comprar passagens e recarregar o cartão respetivamente. Caso o utilizador escolher comprar passagem ele é redirecionado para o menu de compra e caso ele escolha recarregar o cartão ele é redirecionado para o menu de recarregar o PEPE, que funciona de forma muito semelhante ao menu de pagamento

## Menu Stock

Quando o utilizador escolhe entrar no menu do stock é-lhe pedido para introduzir um código de forma a aceder aos dados do stock. Caso o utilizador introduza o código corretamente ele ganha a acesso aos dados do stock. Caso contrário aparece uma mensagem de erro indicando que o código introduzido é inválido

# Discussão de Resultados

Neste projeto foi possível realizar todos objetivos estabelecidos para este trabalho e todas as funções extra, tendo havido alguns percalços como má/dificuldade de interpretação do enunciado e alguma dificuldade no desenvolvimento e debugging de algumas funcionalidades.

# Conclusão

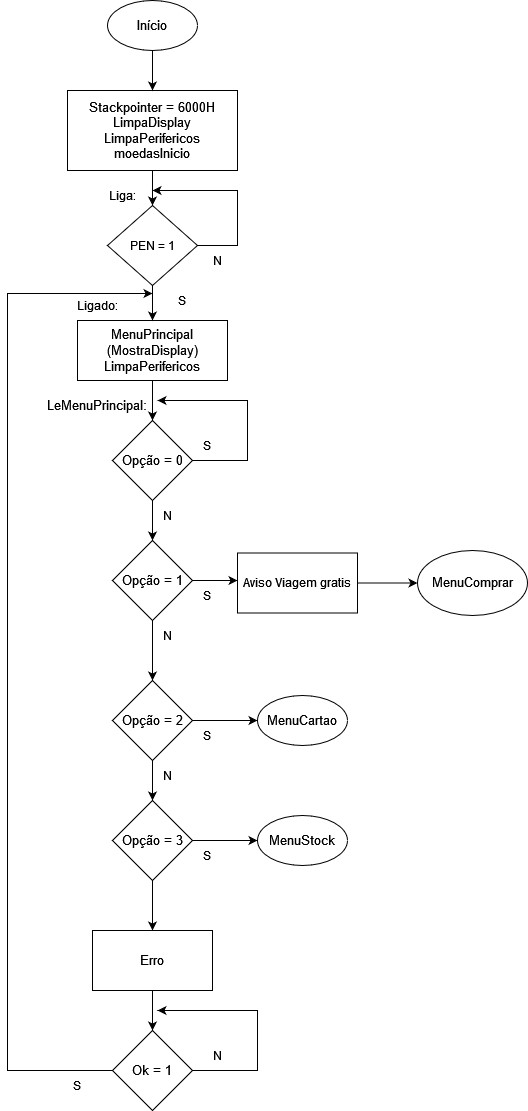
Em suma, o trabalho foi concluído de maneira satisfatória, atendendo a todos os requisitos estabelecidos e demonstrando a aplicação dos conceitos aprendidos durante as aulas teóricas e práticas

# Bibliografia

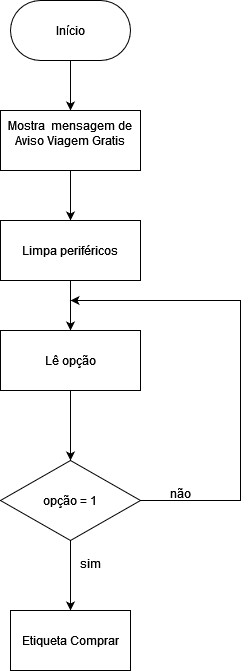
Apontamentos tirados nas aulas teóricas e teórico-práticas e algumas funções desenvolvidas nas aulas práticas laboratoriais.

# Anexo A (Fluxogramas)

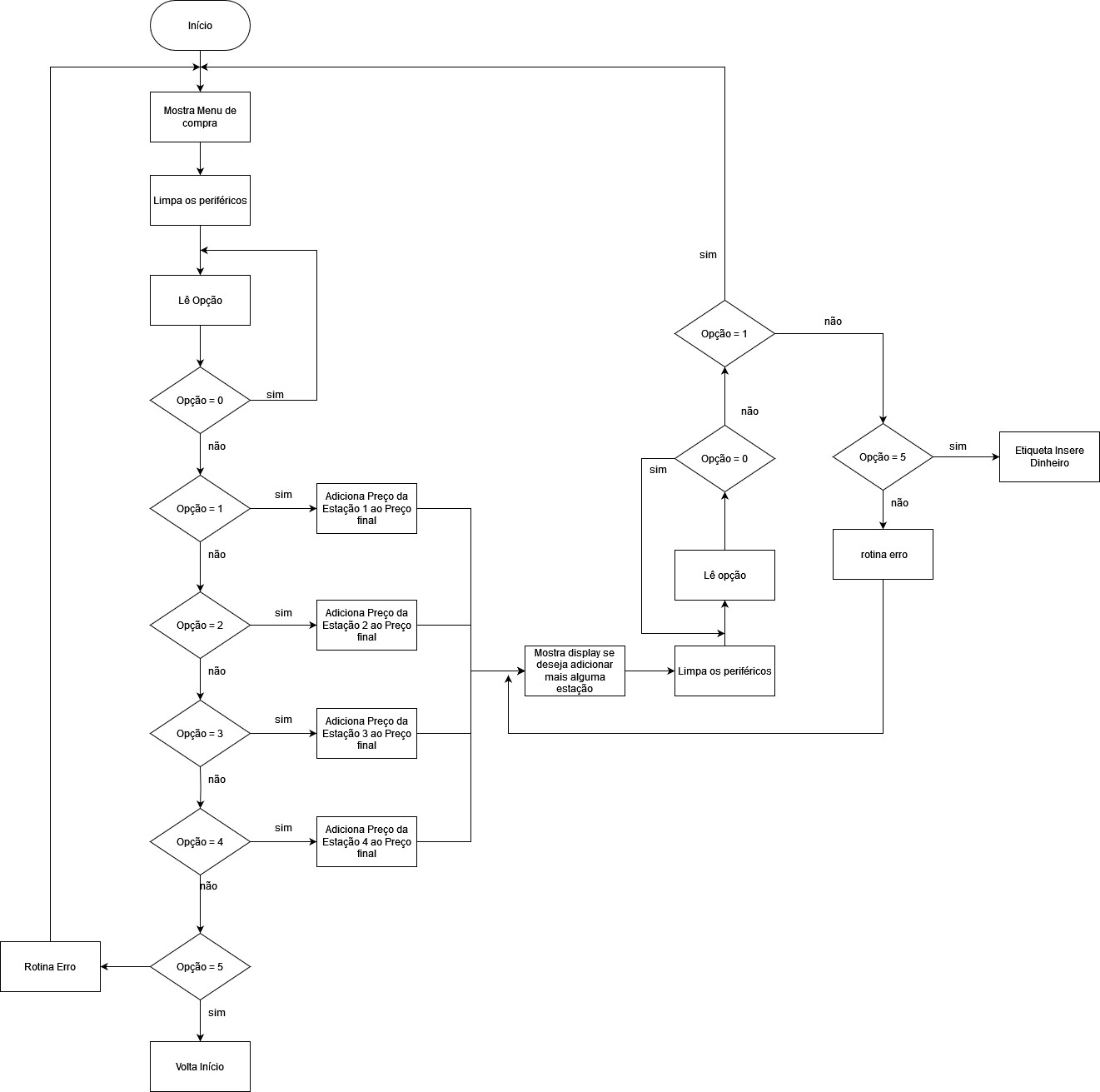
## – Inicio



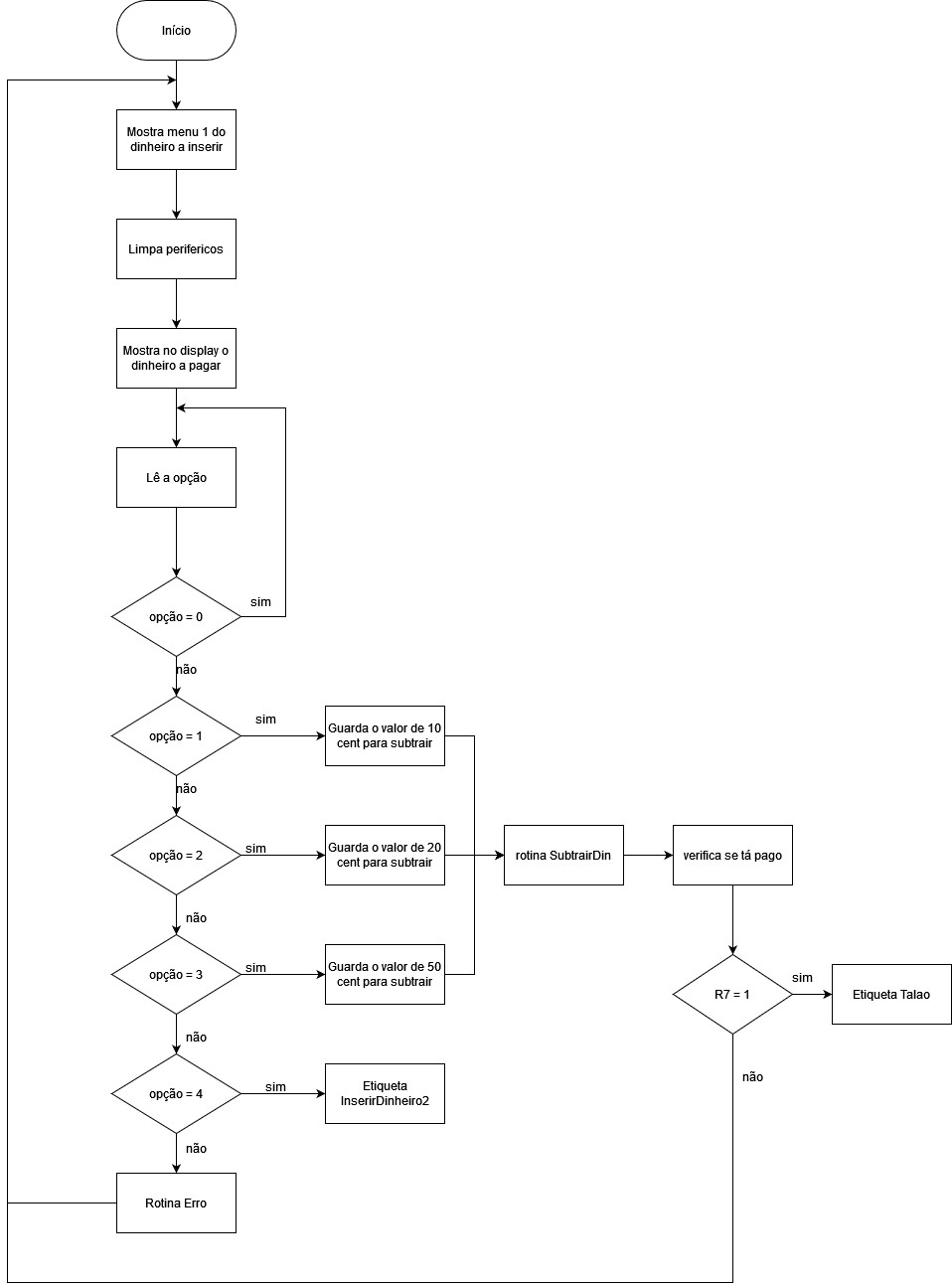
## – AvisoGratis



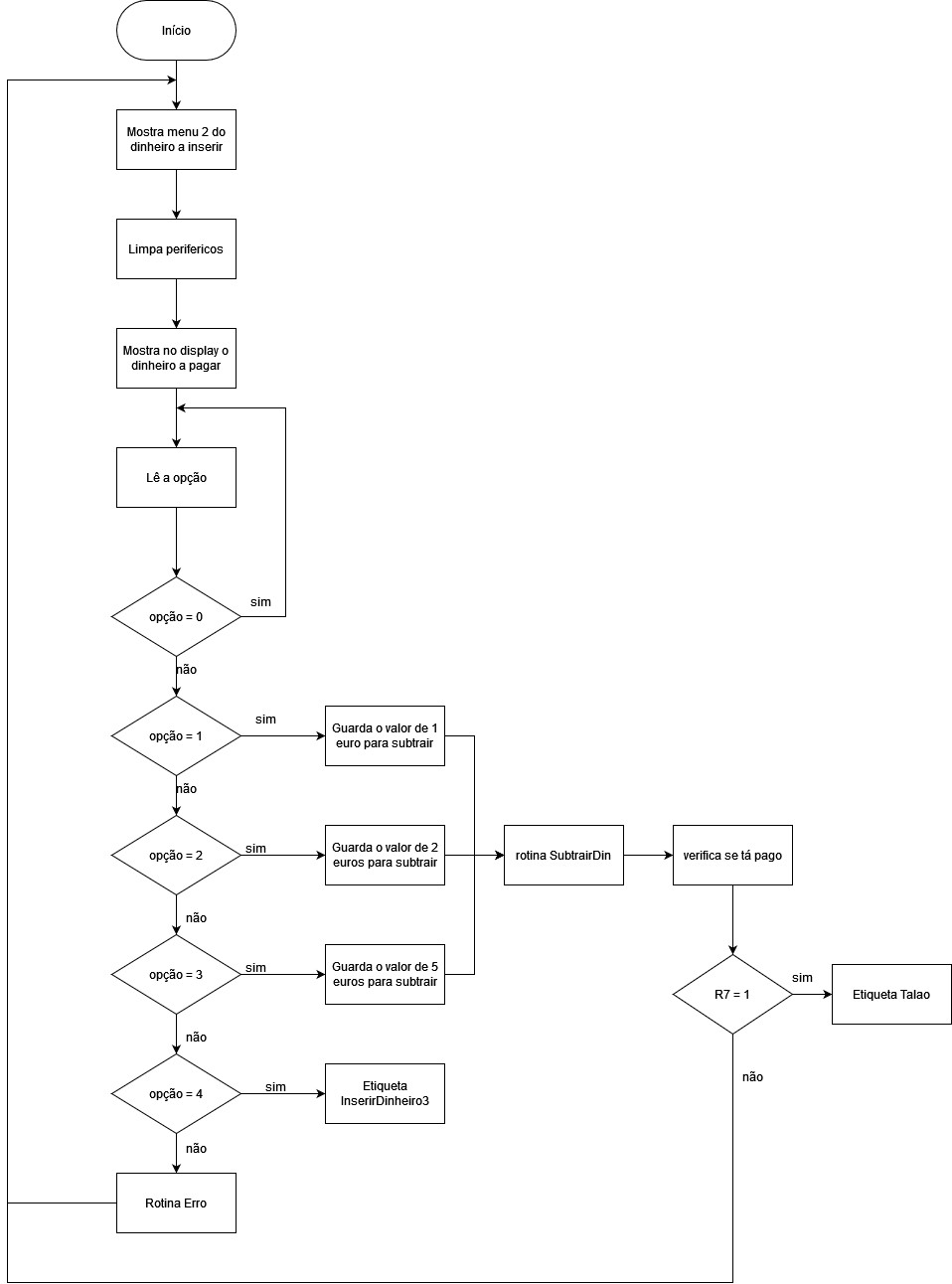
## – Comprar



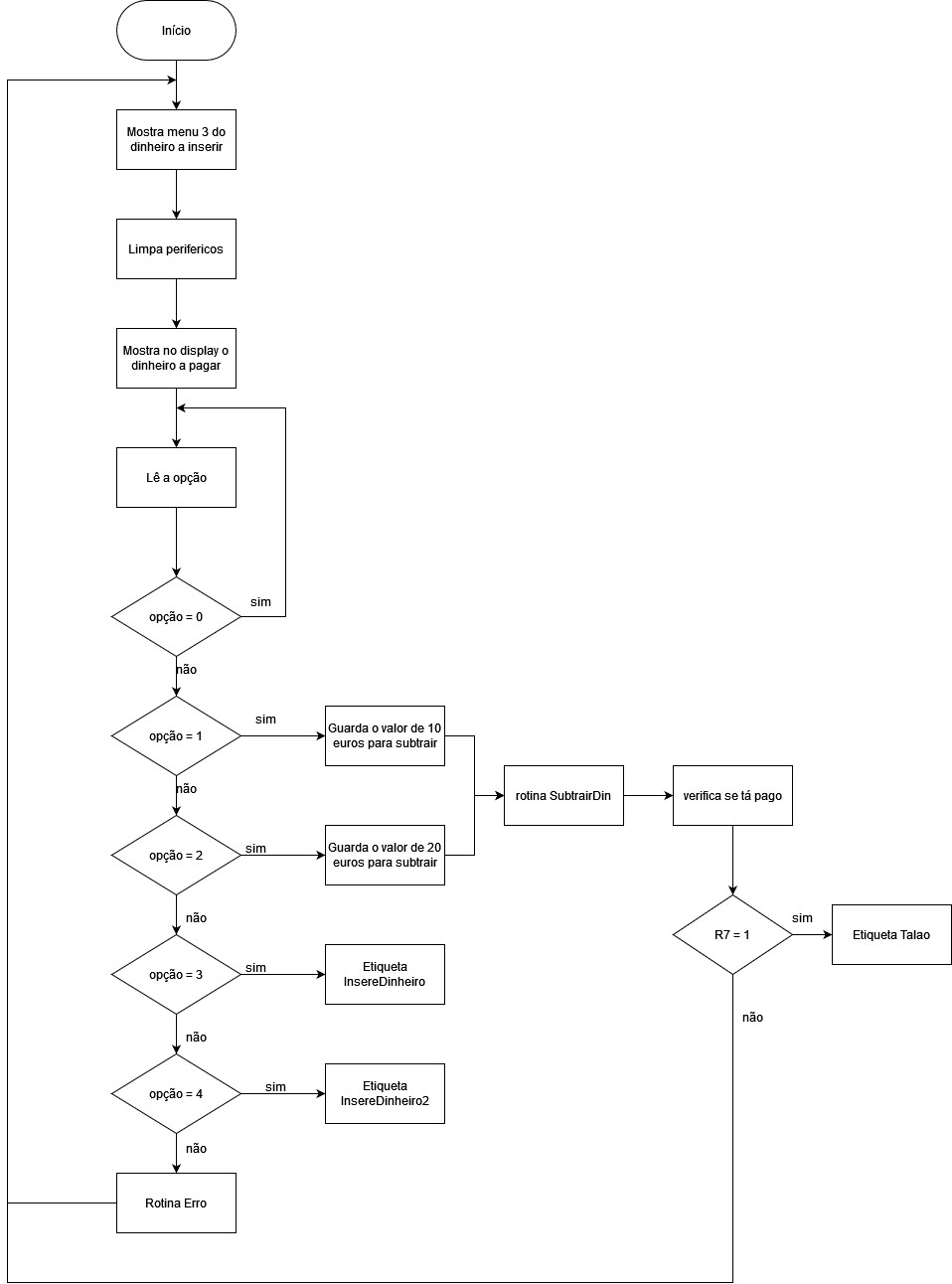
## – InsereDinheiro



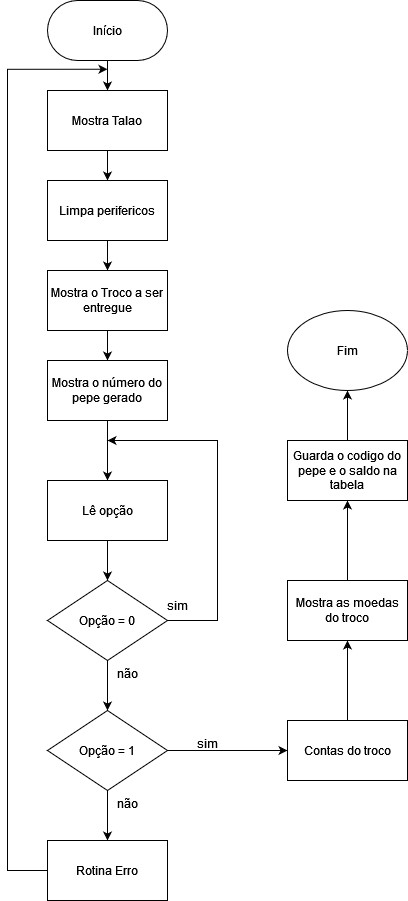
## – InsereDinheiro2



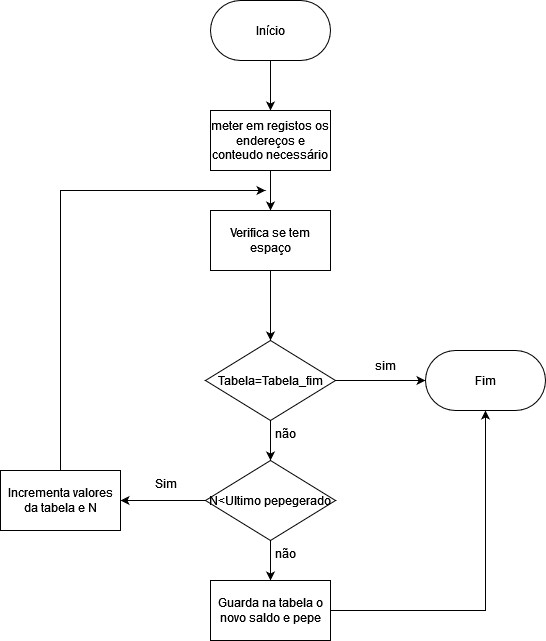
## – InsereDinheiro3



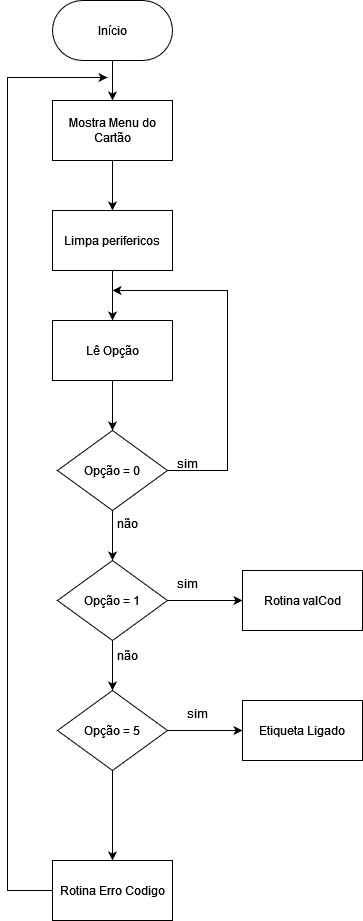
## – Talão



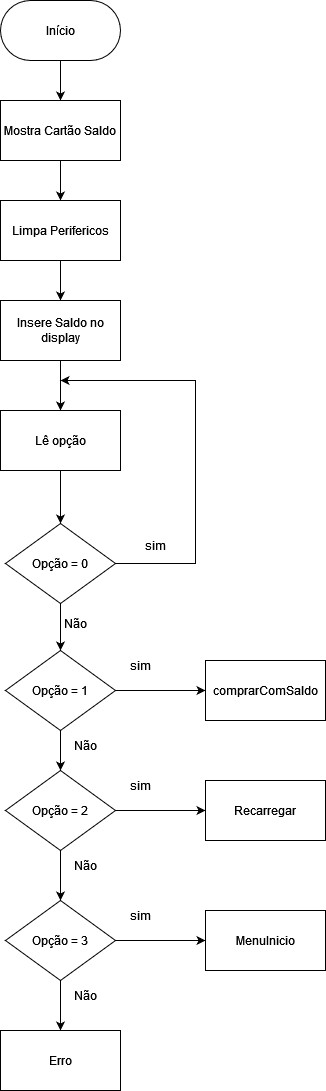
## – Guarda Dados



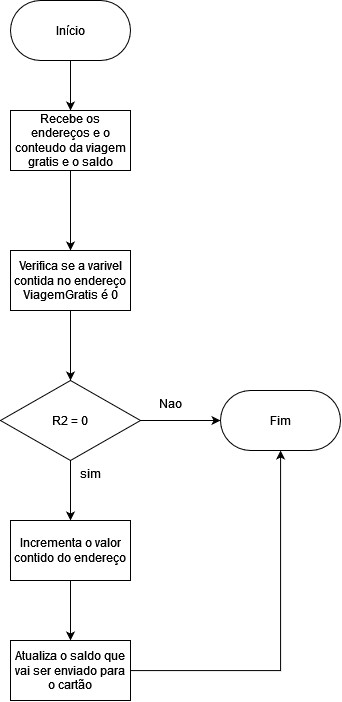
## – Cartao



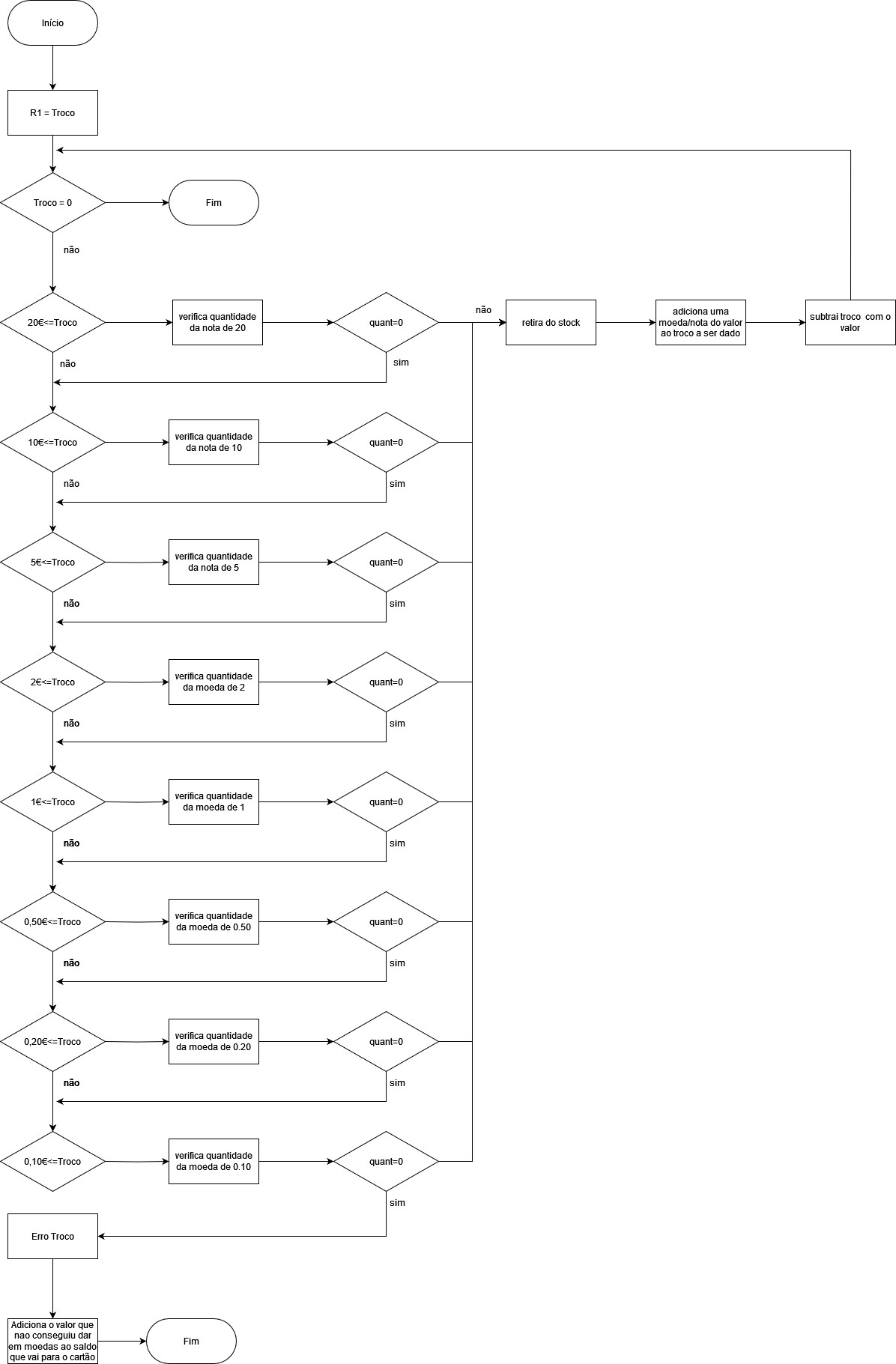
## – EntraCartão



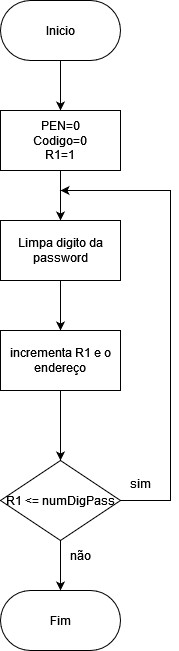
## – Verifica Viagem Gratis



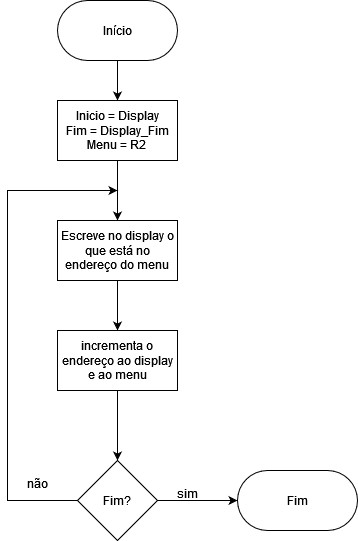
## – Conta do troco(DarTroco)



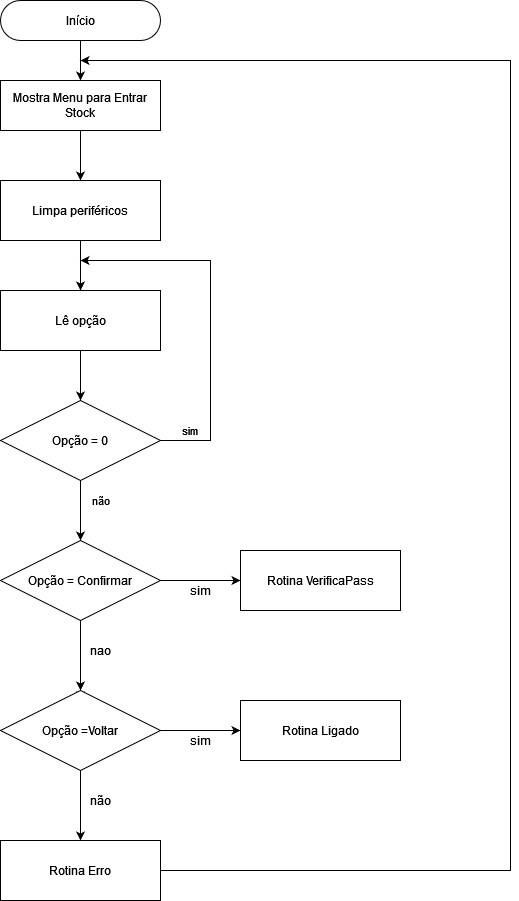
## – Limpa periféricos



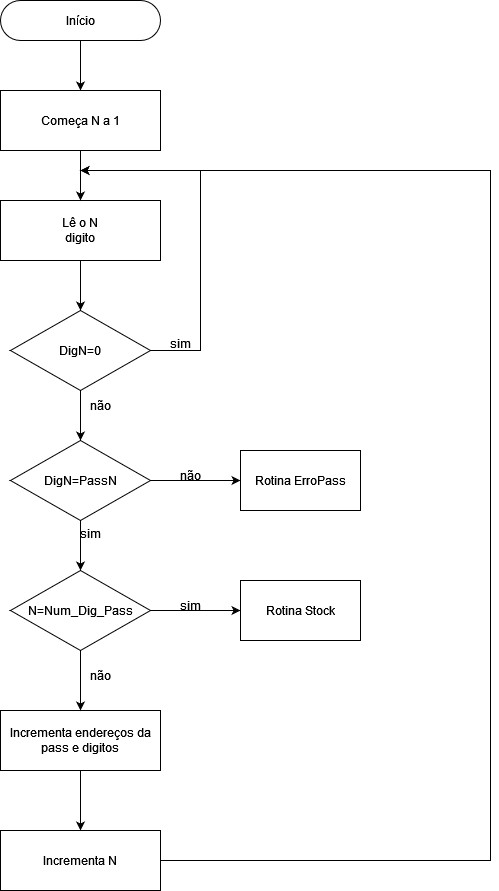
## – Mostra Display



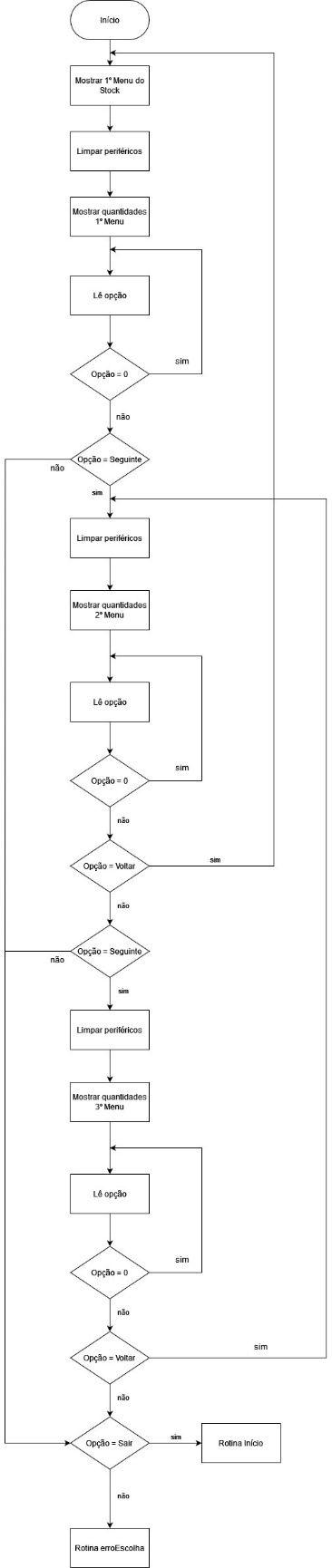
## – Entrar Stock



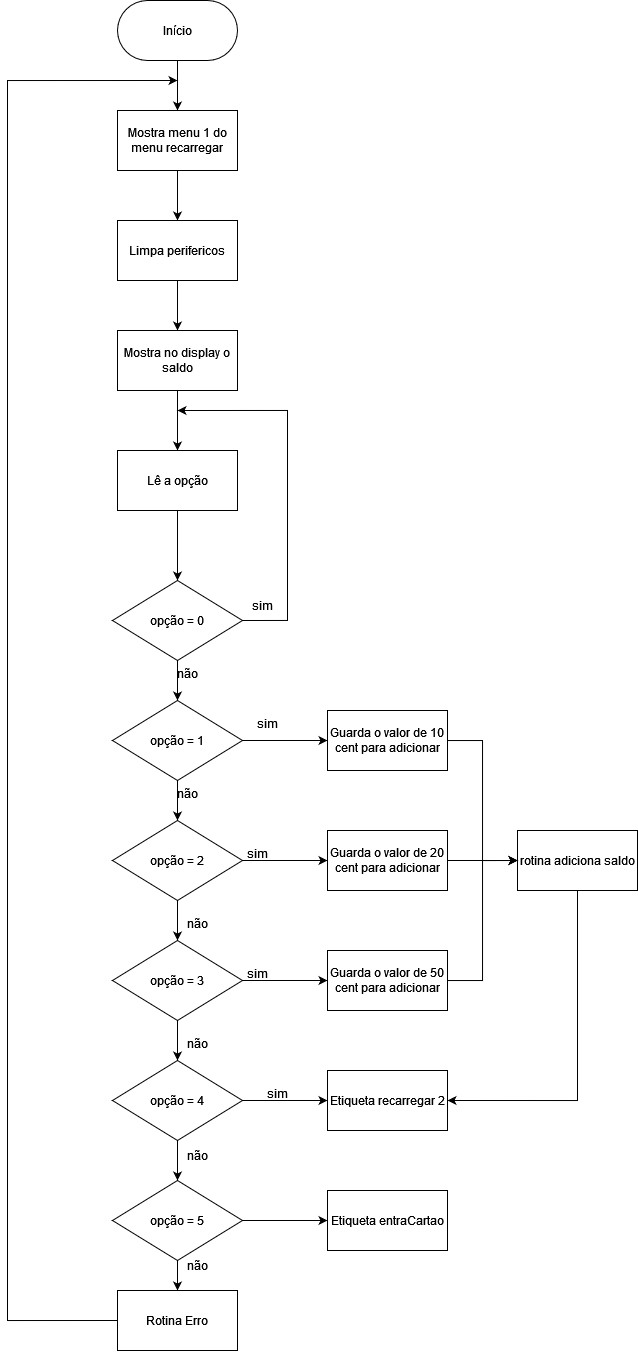
## – Verifica Passe



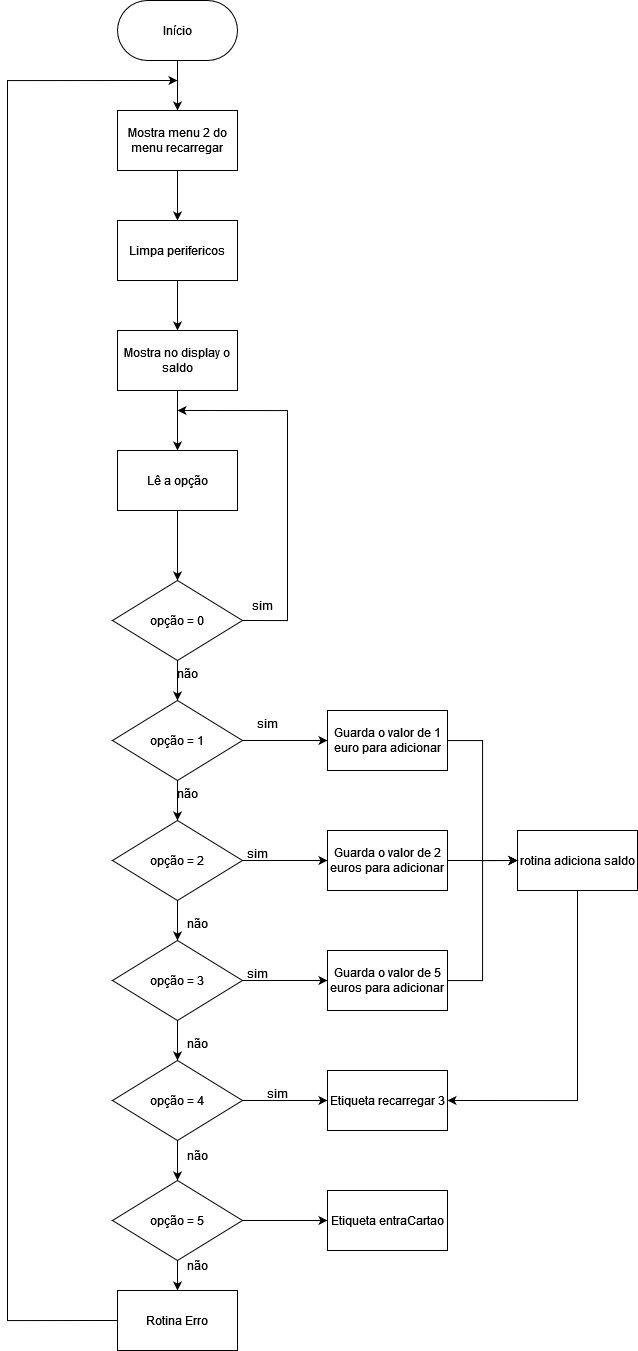
## – Stock



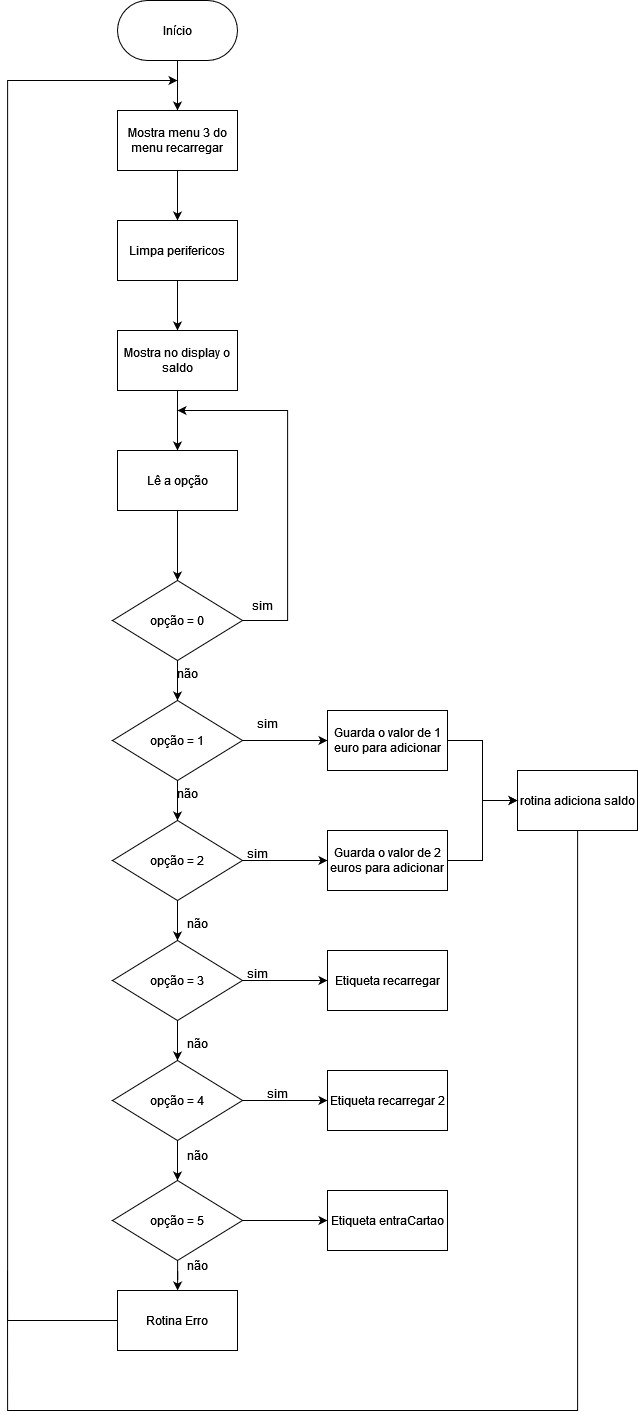
## – recarregar saldo



## – recarregar2



## – recarregar3



# Anexo B (Funções Extra)

De acordo com o que é interpretado com funções extra no enunciado, foram implementadas a gestão do stock das moedas, processamento automático da compra caso o valor monetário introduzido seja igual ou superior ao valor da passagem, valores monetários diferentes para cada estação, botão ON/OFF, promoção da primeira viagem grátis sendo o valor da passagem adicionado ao saldo do PEPE, introdução de valor monetário de 10€ e 20€ e introdução de password para aceder ao stock da máquina. O gerenciamento do Stock também foi feito. Quando não há troco suficiente para fornecer o valor total, o restante será adicionado ao saldo do cartão, pois achamos o mais justo a fazer.

# Anexo C (Código Assembly)

;Constantes simbólicas

PEN EQU 1B0H ;define o endereço do periférico de entrada

Codigo EQU 1C0H ;define o endereço do código do cartão

Tabela EQU 4000H ;define o endereço do início da tabela

Tabela\_fim EQU 4026H ;define o endereço do fim da Tabela

Dig EQU 1D0H ;define o endereço do input do inicio dos digitos da password para aceder ao stock

;Periférico de saída (Display)

Display EQU 200H ;endereço do início do display

Display\_fim EQU 26FH ;endereço do fim do display

StackPointer EQU 6000H ;define endereço da StackPointer para 9000H

CaracterVazio EQU 20H ;define CaracterVazio como " "

;Preço da estações

PrecEst1 EQU 130 ;constante relativa ao preço da estação 1

PrecEst2 EQU 270 ;constante relativa ao preço da estação 2

PrecEst3 EQU 450 ;constante relativa ao preço da estação 3

PrecEst4 EQU 320 ;constante relativa ao preço da estação 4

;Endereços relativos à compra de um bilhete

PEPE EQU 1F20H ;endereço onde se encontra o último endereço PEPE gerado

PrecFinal EQU 1F30H ;endereço onde fica o preço total da compra do bilhete

PrecViagemGratis EQU 1F32H ;endereço onde fica o saldo referente à primeira viagem grátis

ViagemGratis EQU 1F34H ;endereço onde vai ficar 0 ou 1 caso a viagem gratis tenha sido aplicada

DinheiroInserido EQU 1F40H ;endereço onde fica o dinheiro inserido pelo utilizador

Troco EQU 1F50H ;endereço onde fica o troco que tem de ser devolvido

;Endereços para colocar no Display

PrecPagamento EQU 21AH ;endereço do display para começar a colocar os caracteres referente ao Pagamento

SaldoCartao EQU 21AH ;endereço do display para começar a colocar os caracteres referente ao Saldo do PEPE

TrocoTalao EQU 23EH ;endereço do display para começar a colocar os caracteres referente ao Troco do Talão

PosPEPE EQU 228H ;endereço do display para começar a colocar os caracteres referente ao número do PEPE

;Constante para fazer as contas com as moedas

V10cent EQU 10 ;constante com valor 10 para representa 10 cent

V20cent EQU 20 ;constante com valor 20 para representa 20 cent

V50cent EQU 50 ;constante com valor 50 para representa 50 cent

V1euro EQU 100 ;constante com valor 100 para representa 1 eur

V2euro EQU 200 ;constante com valor 200 para representa 2 eur

V5euro EQU 500 ;constante com valor 500 para representa 5 eur

V10euro EQU 1000 ;constante com valor 1000 para representa 10 eur

V20euro EQU 2000 ;constante com valor 2000 para representa 20 eur

;Endereços da quantidade de moedas e notas

q10cent EQU 1F60H ;endereço onde está guardado a quantidade de moedas de 10 centimos

q20cent EQU 1F61H ;endereço onde está guardado a quantidade de moedas de 20 centimos

q50cent EQU 1F62H ;endereço onde está guardado a quantidade de moedas de 50 centimos

q1eur EQU 1F63H ;endereço onde está guardado a quantidade de moedas de 1 euro

q2eur EQU 1F64H ;endereço onde está guardado a quantidade de moedas de 2 euros

q5eur EQU 1F65H ;endereço onde está guardado a quantidade de moedas de 5 euros

q10eur EQU 1F66H ;endereço onde está guardado a quantidade de moedas de 10 euros

q20eur EQU 1F67H ;endereço onde está guardado a quantidade de moedas de 20 euros

;Endereços da quantidade de moedas e notas para troco

t10cent EQU 1F68H ;endereço onde está guardado a quantidade de moedas de 10 centimos para troco

t20cent EQU 1F69H ;endereço onde está guardado a quantidade de moedas de 20 centimos para troco

t50cent EQU 1F6AH ;endereço onde está guardado a quantidade de moedas de 50 centimos para troco

t1eur EQU 1F6BH ;endereço onde está guardado a quantidade de moedas de 1 euro para troco

t2eur EQU 1F6CH ;endereço onde está guardado a quantidade de moedas de 2 euros para troco

t5eur EQU 1F6DH ;endereço onde está guardado a quantidade de moedas de 5 euros para troco

t10eur EQU 1F6EH ;endereço onde está guardado a quantidade de moedas de 10 euros para troco

t20eur EQU 1F6FH ;endereço onde está guardado a quantidade de moedas de 20 euros para troco

;Endereços do display para aparecer as moedas no stock

d10c1e5e EQU 21FH ;endereço para as moedas de 10 cent 1 euro e 5 euros

d20c2e10e EQU 22FH ;endereço para as moedas de 20 cent 2 euros e 10 euros

d50c20e EQU 23FH ;endereço para as moedas de 50 cent 20 euros

;Password

PLACE 1F80H

Password:

String "ST.AC" ;password para entrar no stock

Num\_dig\_pass EQU 5 ;número de digitos da password

;-------------------------------------------------------------------------

; MENUS PARA O DISPLAY

;-------------------------------------------------------------------------

PLACE 2000H

;--------------

;Menu Principal

;--------------

MenuInicio:

String " MAQUINA VENDAS "

String " METRO "

String " "

String "1- Comprar "

String "2- Usar Cartão "

String "3- Stock "

String " "

PLACE 2080H

MenuComprar:

String " Menu Estação "

String "1-Estação1- 1.30"

String "2-Estação2- 2.70"

String "3-Estação3- 4.50"

String "4-Estação4- 3.20"

String " "

String "5- Cancelar "

PLACE 2780H

AvisoComprar:

String " DESEJA "

String " ADICONAR "

String " MAIS "

String " ESTAÇÕES? "

String " "

String "1-Sim "

String "5-Não "

PLACE 2100H

MenuPagar1:

String " Menu Pagamento "

String " . "

String "1- 0.10 "

String "2- 0.20 "

String "3- 0.50 "

String " "

String "4- Seguinte "

PLACE 2180H

MenuPagar2:

String " Menu Pagamento "

String " . "

String "1- 1.00 "

String "2- 2.00 "

String "3- 5.00 "

String " "

String "4- Seguinte "

PLACE 2200H

MenuPagar3:

String " Menu Pagamento "

String " . "

String "1- 10.00 € "

String "2- 20.00 € "

String " "

String "3- Voltar Inicio"

String "4- Voltar "

PLACE 2280H

Talao:

String " "

String " PEPE GERADO "

String " "

String " TROCO: . "

String " "

String "1- Sair "

String " "

PLACE 2300H

MenuCartao:

String " INTRUDUZA Nº "

String " PEPE "

String " "

String " XXXXX "

String " "

String "1- Continuar "

String "5- Cancelar "

PLACE 2380H

CartaoSaldo:

String " SALDO PEPE "

String " . "

String " "

String " "

String "1- Comprar "

String "2- Recarregar "

String "3- Sair "

PLACE 2400H

MenuStock1:

String "-- Stock 1/3 ---"

String "10 cent....... "

String "20 cent....... "

String "50 cent....... "

String "----------------"

String "1- Seguinte "

String "2- Sair "

PLACE 2480H

MenuStock2:

String "-- Stock 2/3 ---"

String "1 euro........ "

String "2 euros....... "

String "----------------"

String "1- Seguinte "

String "2- Voltar "

String "3- Sair "

Place 2500H

MenuStock3:

String "-- Stock 3/3 ---"

String "5 euros....... "

String "10 euros...... "

String "20 euros...... "

String "----------------"

String "1- Voltar "

String "2- Sair "

PLACE 2580H

MenuErro:

String " ERRO "

String " "

String " VALOR "

String " INTRODUZIDO "

String " INVÁLIDO "

String " "

String "1- Ok "

PLACE 2600H

NPEPEErro:

String " "

String " "

String " Nº PEPE "

String " INCORRETO "

String " "

String " "

String "1- Ok "

PLACE 2680H

MenuPassword:

String " INTRODUZA "

String " PASSWORD "

String " "

String " XXXXX "

String " "

String "1- Confirmar "

String "5- Voltar "

PLACE 2700H

ErroPassword:

String " "

String " "

String " PASSWORD "

String " INCORRETA "

String " "

String " "

String "1- Ok "

PLACE 2800H

MenuRecarregar1:

String "Recarregar Pepe "

String " . "

String "1- 0.10 "

String "2- 0.20 "

String "3- 0.50 "

String "4- Seguinte "

String "5- Sair "

PLACE 2880H

MenuRecarregar2:

String "Recarregar Pepe "

String " . "

String "1- 1.00 "

String "2- 2.00 "

String "3- 5.00 "

String "4- Seguinte "

String "5- Sair "

PLACE 2900H

MenuRecarregar3:

String "Recarregar Pepe "

String " . "

String "1- 10.00 € "

String "2- 20.00 € "

String "3- Voltar Inicio"

String "4- Voltar "

String "5- Sair "

PLACE 2980H

MenuSucesso:

String " "

String " Viagem "

String " Comprada "

String " Com "

String " Sucesso "

String " "

String "1- OK "

PLACE 2A00H

MenuPrecRecarregar:

String " "

String " "

String " SALDO "

String " INSUFECIENTE "

String " "

String " "

String "1- OK "

PLACE 2A80H

MenuPrecTrocIns:

String " TROCO "

String " INSUFECIENTE "

String " "

String " restante "

String " adicionado "

String " ao saldo "

String "1- OK "

PLACE 2B00H

MenuTROC1:

String "-- TROCO 1/3 ---"

String "10 cent....... "

String "20 cent....... "

String "50 cent....... "

String "----------------"

String " "

String "1- Seguinte "

PLACE 2B80H

MenuTROC2:

String "-- TROCO 2/3 ---"

String "1 euro........ "

String "2 euros....... "

String "----------------"

String " "

String " "

String "1- Seguinte "

Place 2C00H

MenuTROC3:

String "-- TROCO 3/3 ---"

String "5 euros....... "

String "10 euros...... "

String "20 euros...... "

String "----------------"

String " "

String "1- Sair "

PLACE 2C80H

Menu1ViagemGrat:

String " Atenção "

String " 1ª Viagem "

String " gratis "

String "(preço da vigem)"

String "(fica no saldo )"

String " "

String "1- OK "

PLACE 0000H

MOV R0, Principio ;atribui a R0 o endereço da etiqueta Principio

JMP R0 ;salta para o endereço guardado em R0(Principio)

PLACE 3000H

Principio:

MOV SP, StackPointer ;atribui o endereço da StackPointer

CALL LimpaDisplay ;limpa o display

CALL LimpaPerifericos ;limpa os periféricos

CALL moedasInicio ;adiciona as moedas na maquina

MOV R0, PEN ;atribui a R0 o endereço do periferico de entrada

Liga:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 1 ;compara R1 com o valor 1

JNE Liga ;caso R1 não seja 1 salta para etiqueta Liga

Ligado:

MOV R2, MenuInicio ;atribui a R2 o endereço do MenuInicio

CALL MostraDisplay ;coloca no display o MenuInicio

CALL LimpaPerifericos ;limpa os periféricos

LeMenuPrincipal:

MOV R3, PEN ;atribui a R3 o endereço da opção

MOVB R4, [R3] ;atribui a R4 o 1º byte do valor da opção

CMP R4, 0 ;compara opção com 0

JEQ LeMenuPrincipal ;caso seja 0 salta para etiqueta LeMenuPrincipal

CMP R4, 1 ;compara o valor de opção com 1

JEQ AvisoGratis ;caso opção seja 1 salta para etiqueta AvisoGratis

CMP R4, 2 ;compara o valor de opção com 2

JEQ CartaoIntermedio ;caso opção seja 1 salta para etiqueta Cartão

CMP R4, 3 ;compara o valor de opção com 3

JEQ EntrarStockIntermedio ;caso opção seja 1 salta para etiqueta Stock

CALL Erro ;chama rotina Erro

JMP Ligado ;salta para a etiqueta Ligado

CartaoIntermedio:

JMP Cartao ;Salto para Etiqueta Cartao

EntrarStockIntermedio:

JMP EntrarStock ;Salto para Etiqueta EntrarStock

;----------------

;Etiqueta Comprar

;----------------

AvisoGratis:

MOV R2, Menu1ViagemGrat ;atribui a R2 endereço do Menu1ViagemGrat

CALL MostraDisplay ;coloca no display o Menu1ViagemGrat

CALL LimpaPerifericos ;limpa os periféricos

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

leAvisoGrat:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 1 ;compara opção com 1

JNE leAvisoGrat ;enquanto que nao carrega 1 aparece o aviso

JMP Comprar ;quando carregar passa para a etiqueta Comprar

;----------------

;Etiqueta Comprar

;----------------

Comprar:

MOV R2, MenuComprar ;atribui a R2 endereço do MenuComprar

CALL MostraDisplay ;coloca no display o MenuComprar

CALL LimpaPerifericos ;limpa os periféricos

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoCompra:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara opção com 0

JEQ LeOpcaoCompra ;caso seja 0 salta para etiqueta LeOpcaoCompra

CMP R1, 1 ;compara opção com 1

JEQ adicPrec1 ;caso seja igual salta para a etiqueta adicPrec1

CMP R1, 2 ;compara opção com 2

JEQ adicPrec2 ;caso seja igual salta para a etiqueta adicPrec2

CMP R1, 3 ;compara opção com 3

JEQ adicPrec3 ;caso seja igual salta para a etiqueta adicPrec3

CMP R1, 4 ;compara opção com 4

JEQ adicPrec4 ;caso seja igual salta para a etiqueta adicPrec4

CMP R1, 5 ;compara opção com 5

JEQ Ligado ;caso opção seja 5 salta para etiqueta Ligado

CALL Erro ;chama rotina Erro

JMP Comprar ;salta para etiqueta Comprar

adicPrec1:

MOV R3, PrecEst1 ;atribui a R3 o preço da estação 1

CALL adicionaPre ;chama a rotina adicionaPre para adicionar o valor da estação ao preço final

JMP PerguntaMaisEstacoes ;salta para a Eiqueta PerguntaMaisEstacoes

adicPrec2:

MOV R3, PrecEst2 ;atribui a R3 o preço da estação 2

CALL adicionaPre ;chama a rotina adicionaPre para adicionar o valor da estação ao preço final

JMP PerguntaMaisEstacoes ;salta para a etiqueta PerguntaMaisEstacoes pois pode querer comprar mais viagens

adicPrec3:

MOV R3, PrecEst3 ;atribui a R3 o preço da estação 3

CALL adicionaPre ;chama a rotina adicionaPre para adicionar o valor da estação ao preço final

JMP PerguntaMaisEstacoes ;salta para a etiqueta PerguntaMaisEstacoes pois pode querer comprar mais viagens

adicPrec4:

MOV R3, PrecEst4 ;atribui a R3 o preço da estação 4

CALL adicionaPre ;chama a rotina adicionaPre para adicionar o valor da estação ao preço final

JMP PerguntaMaisEstacoes ;salta para a etiqueta PerguntaMaisEstacoes pois pode querer comprar mais viagens

PerguntaMaisEstacoes:

MOV R2, AvisoComprar ;atribui a R2 endereço do AvisoComprar

CALL MostraDisplay ;coloca no display o AvisoComprar

CALL LimpaPerifericos ;limpa os periféricos

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeAviso:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara R1 com 0

JEQ LeAviso ;caso seja igual salta para a etiqueta LeAviso pois não inseriu uma das opções

CMP R1, 1 ;compara R1 com 1

JEQ Comprar ;caso seja igual salta para a Etiqueta comprar

CMP R1, 5 ;compara R1 com 5

JEQ InserirDinheiro ;caso seja igual salta para a Etiqueta InserirDinheiro

CALL Erro ;caso não seja nenhuma opção válida chama a rotina Erro

JMP PerguntaMaisEstacoes ;Salto para a Etiqueta PerguntaMaisEstacoes

;-------------------------

;Etiqueta Inserir dinheiro

;-------------------------

InserirDinheiro:

MOV R2, MenuPagar1 ;atribui a R2 endereço do MenuPagar1

CALL MostraDisplay ;coloca no display o MenuPagar1

CALL LimpaPerifericos ;limpa os periféricos

CALL DinheiroAPagar ;mostra no display o dinheiro a Pagar atualizado

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoPagamento:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara a opção com 0

JEQ LeOpcaoPagamento ;caso seja igual salta para a Etiqueta LeOpcaoPagamento pois não inseriu uma das opções

CMP R1, 1 ;compara a opção com 1

JEQ Sub10c ;caso seja igual salta para a etiqueta Sub10c

CMP R1, 2 ;compara a opção com 2

JEQ Sub20c ;caso seja igual salta para a etiqueta Sub20c

CMP R1, 3 ;compara a opção com 3

JEQ Sub50c ;caso seja igual salta para a etiqueta Sub50c

CMP R1, 4 ;compara a opção com 4

JEQ InserirDinheiro2 ;caso seja igual passa para o proximo menu (InserirDinheiro2)

CALL Erro ;caso nao seja nenhuma opção chama a rotina Erro

JMP InserirDinheiro ;salta para a etiqueta InserirDinheiro

Sub10c:

MOV R0, V10cent ;atribui a R0 a constante V10cent

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago1 ;salta para VerificaPago1

Sub20c:

MOV R0, V20cent ;atribui a R0 a constante V20cent

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago1 ;salta para VerificaPago1

Sub50c:

MOV R0, V50cent ;atribui a R0 a constante V50cent

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago1 ;salta para VerificaPago1

VerificaPago1:

CMP R7,1 ;compara R7 com 1

JEQ talao ;caso seja igual salta para etiqueta talao

JMP InserirDinheiro ;caso não seja salta para a etiqueta InserirDinheiro

InserirDinheiro2:

MOV R2, MenuPagar2 ;atribui a R2 endereço do MenuPagar2

CALL MostraDisplay ;coloca no display o MenuPagar2

CALL LimpaPerifericos ;limpa os periféricos

CALL DinheiroAPagar ;mostra no display o dinheiro a Pagar atualizado

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoPagamento2:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara R1 com 0

JEQ LeOpcaoPagamento2 ;caso seja igual salta para a Etiqueta LeOpcaoPagamento2 pois não inseriu uma das opções

CMP R1, 1 ;Compara R1 com 1

JEQ Sub1euro ;caso seja igual salta para a etiqueta Sub1euro

CMP R1, 2 ;Compara R1 com 2

JEQ Sub2euro ;caso seja igual salta para a etiqueta Sub2euro

CMP R1, 3 ;Compara R1 com 3

JEQ Sub5euro ;caso seja igual salta para a etiqueta Sub5euro

CMP R1, 4 ;compara R1 com 4

JEQ InserirDinheiro3 ;caso seja igual passa para o proximo menu (InserirDinheiro2)

CALL Erro ;caso nao seja nenhuma opção chama a rotina Erro

JMP InserirDinheiro2 ;salta para a etiqueta InserirDinheiro2

Sub1euro:

MOV R0, V1euro ;atribui a R0 a constante V1euro

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago2 ;salta para VerificaPago2

Sub2euro:

MOV R0, V2euro ;atribui a R0 a constante V2euro

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago2 ;salta para VerificaPago2

Sub5euro:

MOV R0, V5euro ;atribui a R0 a constante V5euro

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago2 ;salta para VerificaPago2

VerificaPago2:

CMP R7,1 ;compara R7 com 1

JEQ talao ;caso seja igual salta para etiqueta talao

JMP InserirDinheiro2 ;caso não seja salta para a etiqueta InserirDinheiro2

InserirDinheiro3:

MOV R2, MenuPagar3 ;atribui a R2 endereço do MenuPagar3

CALL MostraDisplay ;coloca no display o MenuPagar3

CALL LimpaPerifericos ;limpa os periféricos

CALL DinheiroAPagar ;mostra no display o dinheiro a Pagar atualizado

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoPagamento3:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara R1 com 0

JEQ LeOpcaoPagamento3 ;caso seja igual salta para a Etiqueta LeOpcaoPagamento3 pois não inseriu uma das opções

CMP R1, 1 ;Compara R1 com 1

JEQ Sub10euros ;caso seja igual salta para a etiqueta Sub10euros

CMP R1, 2 ;Compara R1 com 2

JEQ SUB20euros ;caso seja igual salta para a etiqueta Sub20euros

CMP R1, 3 ;Compara R1 com 3

JEQ InserirDinheiro ;caso seja igual salta para a etiqueta InserirDinheiro

CMP R1, 4 ;Compara R1 com 4

JEQ InserirDinheiro2 ;caso seja igual salta para a etiqueta InserirDinheiro2

CALL Erro ;caso nao seja nenhuma opção chama a rotina Erro

JMP InserirDinheiro3 ;salta para a etiqueta InserirDinheiro3

Sub10euros:

MOV R0, V10euro ;atribui a R0 a constante V10euro

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago3 ;salta para VerificaPago3

SUB20euros:

MOV R0, V20euro ;atribui a R0 a constante V10euro

CALL SubtrairDin ;chama a rotina SubtrairDin para subtrair o valor a pagar (se o valor inserido der para pagar R7 ficará a 1)

JMP VerificaPago3 ;salta para VerificaPago3

VerificaPago3:

CMP R7,1 ;compara R7 com 1

JEQ talao ;caso seja igual salta para etiqueta talao

JMP InserirDinheiro3 ;caso não seja salta para a etiqueta InserirDinheiro3

;---------------

;Etiqueta Talão

;---------------

talao:

MOV R7, 0 ;reseta o valor de R7 para 0

MOV R2, Talao ;atribui a R2 endereço do Talao

CALL MostraDisplay ;coloca no display o Talao

CALL LimpaPerifericos ;limpa os periféricos

CALL DinheiroTroco ;mostra no display o troco

CALL PepeTalao ;mostra no Display o numero do PEPE gerado

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeSairTalao:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara R1 com 0

JEQ LeSairTalao ;caso seja igual salta para a etiqueta LeSairTalao

CMP R1, 1 ;compara R1 com 1

JEQ LigadoIntermedio2 ;caso seja igual salta para a etiqueta LigadoIntermedio2 pois o endereço ligado está muito longe para um salto condicional

CALL Erro ;caso não seja nenhuma opção chama a rotina Erro

JMP talao ;salta para a etiqueta talao

LigadoIntermedio2:

CALL DarTroco ;faz as contas para troco

CALL MostraTroco ;mostra o troco a receber

CALL GuardaDados ;chama a rotina GuardaDados para guarda na base de dados o saldo da viagem gratis e o codigo do Pepe

CALL limpaEndCompra ;chama a rotina limpaEndCompra para limpar os endereços de troco, preço final, Dinheiro Inserido, etc

JMP Ligado ;salta para a etiqueta Ligado

;----------------------------------------------------

;Rotina para limpar os endereços utilizados na compra

;----------------------------------------------------

limpaEndCompra:

PUSH R0 ;armazena o valor de R0 na StackPointer

PUSH R1 ;armazena o valor de R1 na StackPointer

PUSH R2 ;armazena o valor de R2 na StackPointer

PUSH R3 ;armazena o valor de R3 na StackPointer

PUSH R4 ;armazena o valor de R4 na StackPointer

PUSH R5 ;armazena o valor de R5 na StackPointer

PUSH R6 ;armazena o valor de R6 na StackPointer

MOV R0, PrecFinal ;atribui a R0 o endereço do Preço Final

MOV R1, DinheiroInserido ;atribui a R1 o endereço do Dinheiro Inserido

MOV R2, Troco ;atribui a R2 o endereço do Troco

MOV R3, 0 ;atribui a R3 o valor 0

MOV R4, PrecViagemGratis ;atribui a R4 o endereço do preco da viagem gratis

MOV R5, ViagemGratis ;atribui a R5 o endereço da variavel da viagem gratis

MOV R6, t10cent ;atribui a r6 o endereço das moedas do troco

MOV [R0], R3 ;atribui ao conteudo de R0 zero

MOV [R1], R3 ;atribui ao conteudo de R1 zero

MOV [R2], R3 ;atribui ao conteudo de R2 zero

MOV [R4], R3 ;atribui ao conteudo de R4 zero

MOV [R5], R3 ;atribui ao conteudo de R5 zero

MOV R1, 8 ;são 8 categorias de moeda

cicloMoedasTroco:

MOVB [R6], R3 ;mete o conteudo do endereço a 0

SUB R1, 1 ;passa ao proximo

ADD R6, 1 ;aumenta o endereço passando ao proximo

CMP R1, 0 ;compara com 0

JGT cicloMoedasTroco ;enquanto nao for volta ao ciclo

POP R6 ;recupera o valor de R6

POP R5 ;recupera o valor de R5

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;-------------------

;Rotina Guarda Dados

;-------------------

GuardaDados:

PUSH R0 ;armazena o valor de R0 na StackPointer

PUSH R1 ;armazena o valor de R1 na StackPointer

PUSH R2 ;armazena o valor de R2 na StackPointer

PUSH R3 ;armazena o valor de R3 na StackPointer

PUSH R4 ;armazena o valor de R4 na StackPointer

PUSH R5 ;armazena o valor de R5 na StackPointer

PUSH R6 ;armazena o valor de R6 na StackPointer

MOV R6, 1 ;coloca 1 em R6

MOV R0, Tabela ;atribui a R0 o endereço da Tabela

MOV R1, Tabela\_fim ;atribui a R1 o endereço do fim da Tabela

MOV R2, PEPE ;atribui a R2 o endereço do último valor do PEPE gerado

MOV R3, PrecViagemGratis ;atribui a R3 o endereço do saldo a adicionar associado ao PEPE

MOV R4, [R2] ;atribui a R4 o último PEPE gerado

MOV R5, [R3] ;atribui a R5 o saldo a adicionar associado ao PEPE

verDisponivel:

CMP R0, R1 ;Compara se o próximo endereço livre da tabela é igual ao endereço do fim da tabela

JEQ SemEspaco ;ainda não sei o que fazer para esta situação

CMP R6, R4 ;compara R6 com PEPE

JLT incrementaPosTabela ;caso seja menor salta para etiqueta incrementaPosTabela

MOV [R0], R4 ;guarda o PEPE na tabela

ADD R0, 2 ;soma 2 ao próximo endereço livre da tabela

MOV [R0], R5 ;guarda o saldo associado ao PEPE

ADD R0, 2 ;soma 2 ao próximo endereço livre da tabela

fimGuarda:

POP R6 ;recupera o valor de R6

POP R5 ;recupera o valor de R5

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

SemEspaco:

JMP fimGuarda ;salto para a etiqueta Ligado

incrementaPosTabela:

ADD R0, 4 ;soma 4 ao endereço da tabela

ADD R6, 1 ;incrementa R6

JMP verDisponivel ;salta para etiqueta verDisponivel

;---------------

;Etiqueta Cartão

;---------------

Cartao:

MOV R2, MenuCartao ;atribui a R2 o endereço do MenuCartao

CALL MostraDisplay ;coloca no display o MenuCartao

CALL LimpaPerifericos ;limpa os periféricos

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeMenuCartao:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara o valor de R1 com 0

JEQ LeMenuCartao ;caso seja igual volta à etiqueta LeMenuCartao

CMP R1, 1 ;compara o valor de R1 com 1

JEQ valCod ;caso seja igual ele vai validar o codigo saltando para a etiqueta valCod

CMP R1, 5 ;compara R1 com 5

JEQ LigadoIntermedio3 ; caso seja igual salta para a etiqueta LigadoIntermedio3 pois o endereço ligado fica muito longe para um salto condicional

CALL Erro ;caso não seja nenhuma opção chama a rotina Erro

JMP Cartao ;salta para a Etiqueta Cartao

valCod:

MOV R0, Codigo ;Atribui a R0 o endereço do codigo

MOV R1, [R0] ;Atribui a R1 o conteudo do endereço de R0

CMP R1,0 ;Compara R1 com 0

JEQ valCod ;caso seja igual ainda não foi introduzido um codigo e vai voltar saltando para a Etiqueta valCod

CALL ValidaCodigo ;chama a rotina ValidaCodigo (Se R7 for 1 quer dizer que o codigo está certo)

CMP R7, 1 ;compara R7 com 1

JEQ entraCartao ;caso seja igual salta para a etiq entraCartao

CALL ErroCodPepe ;caso nao seja chama a rotina de erro do codigo

JMP Cartao ;salta para a etiqueta Cartao

;----------------------

;Etiqueta entra Cartao

;----------------------

entraCartao:

MOV R2, CartaoSaldo ;atribui a R2 o endereço do CartaoSaldo

CALL MostraDisplay ;coloca no display o CartaoSaldo

CALL LimpaPerifericos ;limpa os periféricos

CALL InsereSaldo ;mostra no display o Saldo do Cartao

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoCartao:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara o valor de R1 com 0

JEQ LeOpcaoCartao ;caso seja igual salta para a etiqueta LeOpcaoCartao

CMP R1, 1 ;compara o valor de R1 com 1

JEQ comprarComSaldo ;caso seja igual salta para a etiqueta comprarComSaldo para comprar viagens

CMP R1, 2 ;compara o valor de R1 com 2

JEQ recarregar ;caso seja igual salta para a etiqueta recarregar para adicionar saldo

CMP R1, 3 ;compara o valor de R1 com 3

JEQ LigadoIntermedio3 ;caso seja igual salta para a etiqueta LigadoIntermedio3 pois o endereço ligado fica muito longe para um salto condicional

CALL Erro ;caso não seja nenhuma opção chama a rotina Erro

JMP entraCartao ;salta para a Etiqueta entraCartao

LigadoIntermedio3:

JMP Ligado ;salta para a etiqueta Ligado

;----------------------------------

;Etiqueta comprar bilhete com SALDO

;----------------------------------

comprarComSaldo:

MOV R2, MenuComprar ;atribui a R2 o endereço do MenuComprar

CALL MostraDisplay ;coloca no display o MenuComprar

CALL LimpaPerifericos ;limpa os periféricos

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

cicloComprarSaldo:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara o valor de R1 com 1

JEQ cicloComprarSaldo ;caso seja igual salta para a etiqueta cicloComprarSaldo

CMP R1, 1 ;compara o valor de R1 com 1

JEQ compraEst1 ;caso seja igual salta para a etiqueta compraEst1

CMP R1, 2 ;compara o valor de R1 com 2

JEQ compraEst2 ;caso seja igual salta para a etiqueta compraEst2

CMP R1, 3 ;compara o valor de R1 com 3

JEQ compraEst3 ;caso seja igual salta para a etiqueta compraEst3

CMP R1, 4 ;compara o valor de R1 com 4

JEQ compraEst4 ;caso seja igual salta para a etiqueta compraEst4

CMP R1, 5 ;compara o valor de R1 com 5

JEQ entraCartao ;caso seja igual salta para a etiqueta entraCartao

CALL Erro ;caso não seja nenhuma opção chama a rotina Erro

JMP comprarComSaldo ;salta para a etiqueta comprarComSaldo

compraEst1:

MOV R7, PrecEst1 ;Atribui a R7 o valor da estação 1

CALL verificaSaldo ;chama a rotina verificaSaldo(R7 se for 1 é possivel efetuar a compra)

JMP ResultadoOpSaldo ;salta para a etiqueta ResultadoOpSaldo

compraEst2:

MOV R7, PrecEst2 ;Atribui a R7 o valor da estação 2

CALL verificaSaldo ;chama a rotina verificaSaldo(R7 se for 1 é possivel efetuar a compra)

JMP ResultadoOpSaldo ;salta para a etiqueta ResultadoOpSaldo

compraEst3:

MOV R7, PrecEst3 ;Atribui a R7 o valor da estação 3

CALL verificaSaldo ;chama a rotina verificaSaldo(R7 se for 1 é possivel efetuar a compra)

JMP ResultadoOpSaldo ;salta para a etiqueta ResultadoOpSaldo

compraEst4:

MOV R7, PrecEst4 ;Atribui a R7 o valor da estação 4

CALL verificaSaldo ;chama a rotina verificaSaldo(R7 se for 1 é possivel efetuar a compra)

JMP ResultadoOpSaldo ;salta para a etiqueta ResultadoOpSaldo

ResultadoOpSaldo:

CMP R7, 1 ;Compara R7 com 1

JEQ pagoSucess ;caso seja igual salta para a etiqueta pagoSucess

JMP faltaSaldo ;caso não seja salta para a etiqueta faltaSaldo

pagoSucess:

CALL menSucesso ;chama a rotina para apresentar a mensagem de sucesso

JMP entraCartao ;salta para a etiqueta entraCartao

faltaSaldo:

CALL menErroInsuf ;caso o saldo seja insufeciente será chamada a rotina de erro

JMP entraCartao ;salta para a etiqueta entraCartao

;­---------------------------

;Etiqueta recarregar bilhete

;­---------------------------

recarregar:

MOV R2, MenuRecarregar1 ;atribui a R2 o endereço do MenuRecarregar1

CALL MostraDisplay ;coloca no display o MenuRecarregar1

CALL LimpaPerifericos ;limpa os periferico

CALL InsereSaldo ;mostra no display o saldo do PEPE

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoRecar:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara o valor de R1 com 0

JEQ LeOpcaoRecar ;caso seja igual salta para a etiqueta LeOpcaoRecar

CMP R1, 1 ;compara o valor de R1 com 1

JEQ adicSaldo10c ;caso seja igual salta para a etiqueta adicSaldo10c

CMP R1, 2 ;compara o valor de R1 com 2

JEQ adicSaldo20c ;caso seja igual salta para a etiqueta adicSaldo20c

CMP R1, 3 ;compara o valor de R1 com 3

JEQ adicSaldo50c ;caso seja igual salta para a etiqueta adicSaldo50c

CMP R1, 4 ;compara o valor de R1 com 4

JEQ recarregar2 ;caso seja igual salta para a etiqueta recarregar2 (proximo menu)

CMP R1, 5 ;compara o valor de R1 com 5

JEQ entraCartao ;caso seja igual salta para a etiqueta entraCartao

CALL Erro ;caso não seja nenhuma opção chama a rotina Erro

JMP recarregar ;salto para a Etiqueta recarregar

adicSaldo10c:

MOV R7, V10cent ;atribui a R7 o valor de V10cent

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar ;salta para o endereço recarregar

adicSaldo20c:

MOV R7, V20cent ;atribui a R7 o valor de V20cent

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar ;salta para o endereço recarregar

adicSaldo50c:

MOV R7, V50cent ;atribui a R7 o valor de V50cent

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar ;salta para o endereço recarregar

;­---------------------------

recarregar2:

MOV R2, MenuRecarregar2 ;atribui a R2 o endereço do MenuRecarregar2

CALL MostraDisplay ;coloca no display o MenuRecarregar2

CALL LimpaPerifericos ;limpa os Periféricos

CALL InsereSaldo ;mostra no display o saldo do PEPE

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoRecar2:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara o valor de R1 com 0

JEQ LeOpcaoRecar2 ;caso seja igual salta para a etiqueta LeOpcaoRecar2

CMP R1, 1 ;compara o valor de R1 com 1

JEQ adicSaldo1e ;caso seja igual salta para a etiqueta adicSaldo1e

CMP R1, 2 ;compara o valor de R1 com 2

JEQ adicSaldo2e ;caso seja igual salta para a etiqueta adicSaldo2e

CMP R1, 3 ;compara o valor de R1 com 3

JEQ adicSaldo5e ;caso seja igual salta para a etiqueta adicSaldo5e

CMP R1, 4 ;compara o valor de R1 com 4

JEQ recarregar3 ;caso seja igual salta para a etiqueta recarregar3 (proximo menu)

CMP R1, 5 ;compara o valor de R1 com 5

JEQ entraCartao ;caso seja igual salta para a etiqueta entraCartao

CALL Erro ;caso não seja nenhuma opção chama a rotina Erro

JMP recarregar2 ;salto para a etiqueta recarregar2

adicSaldo1e:

MOV R7, V1euro ;atribui a R7 o valor de V1euro

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar2 ;salta para o endereço recarregar2

adicSaldo2e:

MOV R7, V2euro ;atribui a R7 o valor de V2euro

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar2 ;salta para o endereço recarregar2

adicSaldo5e:

MOV R7, V5euro ;atribui a R7 o valor de V5euro

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar2 ;salta para o endereço recarregar2

;­---------------------------

recarregar3:

MOV R2, MenuRecarregar3 ;atribui a R2 o endereço do MenuRecarregar3

CALL MostraDisplay ;coloca no display o MenuRecarregar3

CALL LimpaPerifericos ;limpa os Periféricos

CALL InsereSaldo ;mostra no display o saldo do PEPE

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoRecar3:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara o valor de R1 com 0

JEQ LeOpcaoRecar3 ;caso seja igual salta para a etiqueta LeOpcaoRecar3

CMP R1, 1 ;compara o valor de R1 com 1

JEQ adicSaldo10e ;caso seja igual salta para a etiqueta adicSaldo10e

CMP R1, 2 ;compara o valor de R1 com 2

JEQ adicSaldo20e ;caso seja igual salta para a etiqueta adicSaldo20e

CMP R1, 3 ;compara o valor de R1 com 3

JEQ recarregar ;caso seja igual salta para a etiqueta recarregar

CMP R1, 4 ;compara o valor de R1 com 4

JEQ recarregar2 ;caso seja igual salta para a etiqueta recarregar2

CMP R1, 5 ;compara o valor de R1 com 5

JEQ entraCartaoIntermedio ;caso seja igual salta para a etiqueta entraCartaoIntermedio pois entraCartao é um endereço muito longe para um salto condicional

CALL Erro ;caso não seja nenhuma opção chama a rotina Erro

JMP recarregar3 ;salta para a etiqueta recarregar3

entraCartaoIntermedio:

JMP entraCartao ;salta para a etiqueta entrarCartao

adicSaldo10e:

MOV R7, V10euro ;atribui a R7 o valor de V10euro

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar3 ;salta para a etiqueta recarregar2

adicSaldo20e:

MOV R7, V20euro ;atribui a R7 o valor de V20euro

CALL adicionaSaldo ;chama a rotina adicionaSaldo

JMP recarregar3 ;salta para a etiqueta recarregar2

;--------------------

;Etiqueta EntrarStock

;--------------------

EntrarStock:

MOV R2, MenuPassword ;atribui a R2 endereço do MenuPassword

CALL MostraDisplay ;coloca no display o MenuPassword

CALL LimpaPerifericos ;limpa os periféricos

MOV R0, PEN ;atribui a R0 endereço do periferico de entrada

LeOpcaoPass:

MOVB R1, [R0] ;atribui a R1 o 1ºbyte do valor do periferico de entrada

CMP R1, 0 ;compara R1 com 0

JEQ LeOpcaoPass ;caso seja igual salta para a etiqueta LeOpcaoPass

CMP R1, 1 ;compara R1 com 1

JEQ VerificaPass ;caso seja igual salta para a etiqueta VerificaPass

CMP R1, 5 ;compara R1 com 5

JEQ LigadoIntermedio1 ;caso seja igual salta para a etiqueta Ligado

CALL Erro ;chama a rotina erro

JMP EntrarStock ;salta para a etiqueta EntrarStock

LigadoIntermedio1:

JMP Ligado

;----------------------------------------------

;Rotina Insere SALDO (R6 é o endereço do saldo)

;----------------------------------------------

InsereSaldo:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

PUSH R5 ;armazena valor de R5 na StackPointer

MOV R0, SaldoCartao ;atribui a R0 o endereço do para aprecer no display o Saldo do cartao

MOV R1, [R6] ;Atribui a R1 o conteudo de R6(saldo do cartao)

MOV R2, 5 ;Atribui a R2 o valor 5 pois são os caracteres a serem usados no display da direita para a esqueda, sendo que a 3ª posição é o ponto para diferenciar os euros dos centimos

MOV R3, 48 ;atribui a R3 o valor 48 pois em hexadecimal 48 é 30 e é a partir de 30 é 0 es ascii e 31 é 1, etc até 9

MOV R4, 10 ;atribui a R4 o valor 10

cicloPreencheSaldo:

MOV R5, R1 ;atribui a R5 o valor de R1

DIV R1, R4 ;faz a divisão de R1 por 10, retirando o ultimo caracter

MOD R5, R4 ;faz o resto da divisao de R5 por 10 para ficar com o ultimo digito

ADD R5, R3 ;soma a R5 o 48 para traduzir o numero para ascii e aparecer direito no display

MOVB [R0], R5 ;atribui o conteudo de R0(endereço do display) o digito a inserir

SUB R2, 1 ;subtrai 1 a R2 (numero de caracteres que faltam)

SUB R0, 1 ;subtrai 1 a R0 (passa ao proximo endereço da direita para a esquerda)

CMP R2, 3 ;compara R2 com 3(posição do ponto)

JEQ pontoSaldo ;caso seja igual salta para a etiqueta pontoSaldo

CMP R2, 0 ;compara R2 com 0(acabou)

JEQ fimPreencheSaldo ;caso seja igual salta para a etiqueta fimPreencheSaldo

JMP cicloPreencheSaldo ;salto para a etiqueta cicloPreencheSaldo

pontoSaldo:

SUB R2, 1 ;subtrai 1 a R2 (numero de caracteres que faltam)

SUB R0, 1 ;subtrai 1 a R0 (passa ao proximo endereço da direita para a esquerda)

JMP cicloPreencheSaldo ;salto para a etiqueta cicloPreencheSaldo

fimPreencheSaldo:

POP R5 ;recupera o valor a R5

POP R4 ;recupera o valor a R4

POP R3 ;recupera o valor a R3

POP R2 ;recupera o valor a R2

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

RET

;---------------------------------------------------------------------

;Rotina verifica SALDO (R6 é o endereço do saldo, R7 valor da estação)

;---------------------------------------------------------------------

verificaSaldo:

PUSH R0 ;armazena valor de R0 na StackPointer

MOV R0, [R6] ;Atribui a R0 o conteudo de R6 (Saldo)

CMP R0, R7 ;Compara o valor da estação com o saldo do cartão

JGE realizaCompra ;Caso o valor do saldo seja mairo ou igual ao valor da estação faz um salto para a Etiqueta realizaCompra

MOV R7, 0 ;caso o saldo seja insufeciente R7 ficara com o valor 0

JMP fimVerifSaldo ; salta para a etiqueta fimVerifSaldo

realizaCompra:

SUB R0, R7 ;Subtrar o saldo com o preço da estação

MOV [R6], R0 ;atualiza o conteudo de R6(atualiza o saldo)

MOV R7, 1 ;saldo é sufeciente R7 ficar com o valor 1

fimVerifSaldo:

POP R0 ;recupera o valor de R0

RET

;---------------------------------------------------------------------

;Rotina adiciona SALDO (R6 é o endereço do saldo, R7 valor a inserir)

;---------------------------------------------------------------------

adicionaSaldo:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R0, R7 ;guarda em R0 o valor inserido

CALL VerificaMoeda ;adiciona a moeda ao stock

MOV R1, [R6] ;Atribui a R0 o conteudo de R6(saldo)

ADD R1, R7 ;soma R0 com R7(saldo atual mais o valor a adicionar)

MOV [R6], R1 ;Atribui ao conteudo de R6 o valor de R0(atualiza o saldo)

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;---------------------------------------------------------------------

;Rotina mensagem sucesso compra

;---------------------------------------------------------------------

menSucesso:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R2, MenuSucesso ;atribui a R2 o endereço de MenuSucesso

CALL MostraDisplay ;coloca no display o MenuSucesso

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeOkMenSucess:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeOkMenSucess ;caso valor de R1 não seja 1 salta para etiqueta LeOk

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

POP R2 ;recupera o valor a R2

RET

;---------------------------------------------------------------------

;Rotina mensagem saldo insufeciente compra

;---------------------------------------------------------------------

menErroInsuf:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R2, MenuPrecRecarregar ;atribui a R2 o endereço de MenuPrecRecarregar

CALL MostraDisplay ;coloca no display o MenuSucesso

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeOkMenIns:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeOkMenIns ;caso valor de R1 não seja 1 salta para etiqueta LeOk

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

POP R2 ;recupera o valor a R2

RET

;---------------------------------------

;Rotina ValidaCodigo (R1=Codigo do PEPE)

;---------------------------------------

ValidaCodigo:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

MOV R2, Tabela ;Atribui a R2 o endereço Tabela

MOV R3, Tabela\_fim ;Atribui a R3 o endereço Tabela\_fim

CicloValida:

CMP R2, R3 ;compara R2 com R3(verifica se chegou ao fim)

JGE CodigoNaoExiste ;nao encontrou um codigo valido

MOV R4, [R2] ;atribui a R4 o conteudo de R2(codigo do pepe)

CMP R4, R1 ;compara se é o mesmo codigo que foi inserido

JEQ CodigoValido ;caso seja igual salta para etiqueta CodigoValido

ADD R2, 4 ;soma ao endereço o valor 4, ou seja, passa para o proximo codigo pepe

JMP CicloValida ;salta para a Etiqueta CicloValida

CodigoNaoExiste:

MOV R7, 0 ;caso o codigo nao exista R7 ficará a 0

JMP fimValidaCodigo ;salta para a etiqueta fimValidaCodigo

CodigoValido:

MOV R7, 1 ;caso o codigo exista R7 ficará a 1

MOV R6, R2 ;em R6 ficará guardado o enderço do codigo pepe ecolhido

ADD R6, 2 ;adicionamos 2 ao endereço para que em R6 fique o endereço do saldo do cartão pepe

fimValidaCodigo:

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

RET

;-------------------

;Etiqueta VerificaPass

;-------------------

VerificaPass:

MOV R4, 1 ;Atribui a R4 o valor n que vai incrementar até a 5(numero de digitos da pass)

MOV R0, Password ;atribui a R0 o endereço da password

MOV R2, Dig ;atribui a R2 o endereço do input do inicio dos digitos da password

VerifDig:

MOVB R1, [R0] ;atribui a R1 o n digito da password

MOVB R3, [R2] ;atribui a R3 o input do n digito da password

CMP R3, 0 ;compara R1 com 0

JEQ VerifDig ;caso seja igual salta para a etiqueta VerifDig

CMP R1, R3 ;compara R1 com R3

JEQ fimVerif ;caso seja igual salta para a etiqueta fimVerif

CALL ErroPass ;chama a rotina ErroPass

JMP EntrarStock ;salta para a etiqueta EntrarStock

fimVerif:

CMP R4, Num\_dig\_pass ;compara R4 com o número de digitos da password

JEQ Stock1 ;caso seja igual salta para a etiqueta Stock1

JMP IncrementaEnderecos ;salta para a etiqueta IncrementaEnderecos

IncrementaEnderecos:

ADD R4, 1 ;incrementa R4

ADD R0, 1 ;incrementa o endereço da password

ADD R2, 1 ;incrementa o endereço do input dos digitos da password

JMP VerifDig ;salta para a etiqueta VerifDig

;-------------

;Eiqueta Stock

;-------------

Stock1:

MOV R2, MenuStock1 ;atribui a R2 o endereço do MenuStock1

CALL MostraDisplay ;coloca no display o MenuStock1

CALL LimpaPerifericos ;limpa os periféricos

MOV R7, 1 ;R7 fica a 1 para usar na rotina MStock

CALL MStock ;chama a rotina MStock

MOV R0, PEN ;atribui a R0 o endereço da opção

LeOpcaoStock1:

MOVB R1, [R0] ;atribui a R1 o 1º byte da opção

CMP R1, 0 ;compara opção com 0

JEQ LeOpcaoStock1 ;caso seja 0 salta para etiqueta LeOpcaoStock1

CMP R1, 1 ;compara opção com 1

JEQ Stock2 ;caso seja 1 salta para etiqueta Stock2

CMP R1, 2 ;compara opção com 2

JEQ LigadoIntermedio ;caso seja 2 salta para etiqueta Ligado

CALL Erro ;chama rotina Erro

JMP LeOpcaoStock1 ;salta para etiqueta LeOpcaoStock1

Stock2:

MOV R2, MenuStock2 ;atribui a R2 o endereço do MenuStock2

CALL MostraDisplay ;coloca no display o MenuStock2

CALL LimpaPerifericos ;limpa os periféricos

MOV R7, 2 ;R7 fica a 2 para usar na rotina MStock

CALL MStock

MOV R0, PEN ;atribui a R0 o endereço da opção

LeOpcaoStock2:

MOVB R1, [R0] ;atribui a R1 o 1º byte da opção

CMP R1, 0 ;compara opção com 0

JEQ LeOpcaoStock2 ;caso seja 0 salta para etiqueta LeOpcaoStock2

CMP R1, 1 ;compara opção com 1

JEQ Stock3 ;caso seja 1 salta para etiqueta Stock3

CMP R1, 2 ;compara opção com 2

JEQ Stock1 ;caso seja 2 salra para a etiqueta Stock1

CMP R1, 3 ;compara a opção com 3

JEQ LigadoIntermedio ;caso seja 3 salta para etiqueta Ligado

CALL Erro ;chama rotina Erro

JMP LeOpcaoStock2 ;salta para etiqueta LeOpcaoStock2

LigadoIntermedio:

JMP Ligado

Stock3:

MOV R2, MenuStock3 ;atribui a R2 o endereço do MenuStock3

CALL MostraDisplay ;coloca no display o MenuStock3

CALL LimpaPerifericos ;limpa os periféricos

MOV R7, 3 ;R7 fica a 3 para usar na rotina MStock

CALL MStock

MOV R0, PEN ;atribui a R0 o endereço da opção

LeOpcaoStock3:

MOVB R1, [R0] ;atribui a R1 o 1º byte da opção

CMP R1, 0 ;compara opção com 0

JEQ LeOpcaoStock3 ;caso seja 0 salta para etiqueta LeOpcaoStock3

CMP R1, 1 ;compara opção com 1

JEQ Stock2 ;caso seja 1 salta para etiqueta Stock2

CMP R1, 2 ;compara opção com 2

JEQ LigadoIntermedio ;caso seja 2 salta para etiqueta Ligado

CALL Erro ;chama rotina Erro

JMP LeOpcaoStock3 ;salta para etiqueta LeOpcaoStock3

;-----------

;Rotina Erro

;-----------

Erro:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R2, MenuErro ;atribui a R2 o endereço de MenuErro

CALL MostraDisplay ;coloca no display o MenuErro

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeOk:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeOk ;caso valor de R1 não seja 1 salta para etiqueta LeOk

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

POP R2 ;recupera o valor a R2

RET

;-----------------

;Rotina Erro Troco

;-----------------

ErroTroco:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R2, MenuPrecTrocIns ;atribui a R2 o endereço de MenuPrecTrocIns

CALL MostraDisplay ;coloca no display o MenuPrecTrocIns

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeOkTroco:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeOkTroco ;caso valor de R1 não seja 1 salta para etiqueta LeOkTroco

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

POP R2 ;recupera o valor a R2

RET

;--------------------

;Rotina Erro password

;--------------------

ErroPass:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R2, ErroPassword ;atribui a R2 o endereço de ErroPassword

CALL MostraDisplay ;coloca no display o ErroPassword

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeOkPass:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeOkPass ;caso valor de R1 não seja 1 salta para etiqueta LeOk

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

POP R2 ;recupera o valor a R2

RET

;-----------------------

;Rotina Erro codigo Pepe

;-----------------------

ErroCodPepe:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R2, NPEPEErro ;atribui a R2 o endereço de NPEPEErro

CALL MostraDisplay ;coloca no display o NPEPEErro

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeOkCodPepe:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeOkCodPepe ;caso valor de R1 não seja 1 salta para etiqueta LeOkCodPepe

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

POP R2 ;recupera o valor a R2

RET

;--------------------

;Rotina MostraDisplay

;--------------------

MostraDisplay:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R3 ;armazena valor de R1 na StackPointer

MOV R0, Display ;atribui a R0 o endereço do inicio do display

MOV R1, Display\_fim ;atribui a R1 o endereço do fim do display

CicloEscreve:

MOV R3, [R2] ;atribui a R3 o conteudo de R2 (endereço do menu a exibir)

MOV [R0], R3 ;atribui ao conteudo de R0 a palavra em R3

ADD R0, 2 ;adiciona 2 ao valor de R0

ADD R2, 2 ;adiciona 2 ao valor de R2

CMP R0, R1 ;compara R0 com R1 (inicio do display com fim do display)

JLE CicloEscreve ;caso os endereços do display não sejam iguais (display não está preenchido) salta para etiqueta CicloEscreve

POP R3 ;recupera o valor de R3

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;----------------------------------------

;Rotina soma Preços (R3 valor da estação)

;----------------------------------------

adicionaPre:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R0, PrecFinal ;atribui a R0 o endereço onde vai ficar o preço final da compra

MOV R1, [R0] ;atribui a R1 o conteudo do R0

ADD R1, R3 ;adiciona o atual preco final ao valor da estação que está guardado em R3

MOV [R0], R1 ;atribui ao conteudo de R0 o valor de R1

CALL VerifViagemGratis ;chama a rotina VerifViagemGratis

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;----------------------------------------------------

;Rotina saldo para viagem gratis(R3 valor da estação)

;----------------------------------------------------

VerifViagemGratis:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

MOV R0, PrecViagemGratis ;atribui a R0 o endereço do saldo a enviar para pepe da viagem gratis

MOV R1, ViagemGratis ;atribui a R1 o endereço ondem contem 0(viagem gratis não atribuída) ou 1(viagem gratis já atribuída)

MOV R2, [R1] ;atribui a R2 o conteudo de R1

CMP R2, 0 ;compara R2 com 0

JEQ adicionaSaldoVigGr ;caso seja igual salta para a etiqueta adicionaSaldoVigGr

JMP fimVerifVigGr ;caso não seja(viagem gratis já atribuída) salta para fimVerifVigGr

adicionaSaldoVigGr:

MOV R4, 1 ;atribui a R4 o valor 1

MOV [R0], R3 ;atribui ao conteudo de R0 o valor de R3

MOV [R1], R4 ;atribui ao conteudo de R1 o valor de R4

JMP fimVerifVigGr ;salta para fimVerifVigGr

fimVerifVigGr:

POP R4 ;recupera o valor de R4

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;-------------------

;Rotina LimpaDisplay

;-------------------

LimpaDisplay:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

MOV R0, Display ;atribui a R0 o endereço do inicio do display

MOV R1, Display\_fim ;atribui a R1 o endereço do fim do display

MOV R2, CaracterVazio ;atribui a R2 o valor de CaracterVazio

CicloLimpa:

MOVB [R0], R2 ;escreve no 1º byte do display o valor de R2

ADD R0, 1 ;incrementa R0 (endereço do display)

CMP R0, R1 ;compara R0 com R1 (posição atual do display com posição final do display)

JLE CicloLimpa ;caso R0 seja menor salta para etiqueta CicloLimpa

POP R2 ;recupera valor de R2

POP R1 ;recupera valor de R1

POP R0 ;recupera valor de R0

RET

;-----------------------

;Rotina LimpaPerifericos

;-----------------------

LimpaPerifericos:

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

PUSH R5 ;armazena valor de R5 na StackPointer

MOV R1, PEN ;atribui a R1 o endereço do periferico de entrada

MOV R2, Codigo ;atribui a R1 o endereço do codigo do pepe

MOV R4, Dig ;atribui a R4 o endereço do inicio dos 5 digitos da password

MOV R5, 1 ;atribui a R5 o valor 1

MOV R3, 0 ;atribui a R3 o valor 0

MOV [R2], R3 ;coloca no conteudo do endereço R2 o valor de R3

MOV [R1], R3 ;coloca no conteudo do endereço R1 o valor de R3

cicloLimpaDigitos:

MOVB [R4], R3 ;coloca no conteudo do 1ºbyte do endereço R4 o valor de R3

ADD R5, 1 ;adiciona 1 ao R5(incrementa o numero de digitos limpos)

ADD R4, 1 ;adiciona 1 a R4(incrementa o endereço dos digitos)

CMP R5, Num\_dig\_pass ;compara R5 com Num\_dig\_pass

JLE cicloLimpaDigitos ;caso seja menor ou igual vai continuar a limpar, salto para a etiqueta cicloLimpaDigitos

POP R5 ;recuper o valor de R5

POP R4 ;recuper o valor de R4

POP R3 ;recuper o valor de R3

POP R2 ;recuper o valor de R2

POP R1 ;recuper o valor de R1

RET

;-----------------------

;Rotina troco talão

;-----------------------

DinheiroTroco:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

PUSH R5 ;armazena valor de R5 na StackPointer

PUSH R6 ;armazena valor de R6 na StackPointer

MOV R0, TrocoTalao ;Atribui a R0 o endereço onde será escrito o troco no Display

MOV R1, Troco ;Atribui a R1 o endereço onde está o Troco

MOV R2, [R1] ;Atribui a R2 o conteudo de R1(Troco)

MOV R3, 10 ;Atribui a R3 o valor 10

MOV R4, 5 ;Atribui a R4 o valor 5 pois são os caracteres a serem usados no display da direita para a esqueda, sendo que a 3ª posição é o ponto para diferenciar os euros dos centimos

MOV R5, 48 ;atribui a R5 o valor 48 pois em hexadecimal 48 é 30 e é a partir de 30 é 0 es ascii e 31 é 1, etc até 9

cicloPreencheTalao:

MOV R6, R2 ;Atribui a R6 o valor de R2

DIV R2, R3 ;Faz a divisao de R2 por 10, retirando assim o ultimo caracter

MOD R6, R3 ;Faz o resto da divisao de R6 por 10, ficando assim com o ultimo caracter

ADD R6, R5 ;adiciona ao ultimo caracter 48 para traduzir o numero para ascii para aparecer direito no display

MOVB [R0], R6 ;adiciona no 1ºbyte do endereço display o caracter traduzido

SUB R4, 1 ;subtrai 1 a R4 para dizer que já preenchemos mais um digito

SUB R0, 1 ;subtrai o endereço na posição do display pois escrevemos o numero da direita para a esquerda

CMP R4, 3 ;compara R4 com 3 pois 3 é a posição do ponto

JEQ pontoTalao ;caso seja igual faz um salto para pontoTalao

CMP R4, 0 ;compara r4 com 0 para ver se já acabou

JGT cicloPreencheTalao ;caso seja maior vai continuar

JEQ fimPreencheTalao ;caso acabe salta para a etiqueta fimPreencheTalao

pontoTalao:

SUB R4, 1 ;subtrai 1 a R4 para dizer que já preenchemos mais um digito

SUB R0, 1 ;subtrai o endereço na posição do display pois escrevemos o numero da direita para a esquerda

JMP cicloPreencheTalao ;salta para a etiqueta cicloPreencheTalao

fimPreencheTalao:

POP R6 ;recupera o valor de R6

POP R5 ;recupera o valor de R5

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;-----------------

;Rotina PEPE Talão

;-----------------

PepeTalao:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

PUSH R5 ;armazena valor de R5 na StackPointer

PUSH R6 ;armazena valor de R6 na StackPointer

MOV R0, PosPEPE ;Atribui a R0 o endereço da posição do nº do pepe no display

MOV R1, PEPE ;Atribui a R1 o endereço do nº do ultimo pepe gerado

MOV R2, [R1] ;Atribui a R2 o conteudo de R1(nº do ultimo pepe gerado)

ADD R2, 1 ;Soma 1 a R2 para o proximos nº pepe

MOV [R1], R2 ;atualiza o conteudo de r1 para um novo nº pepe

MOV R3, 10 ;Atribui a R3 o valor 10

MOV R4, 48 ;atribui a R4 o valor 48 pois em hexadecimal 48 é 30 e é a partir de 30 é 0 es ascii e 31 é 1, etc até 9

MOV R5, 3 ;Atribui a R5 o valor 3 pois são os caracteres a serem escritos no display da direita para a esqueda

CicloEscrevePEPETalao:

MOV R6, R2 ;Atribui o valor de R2 a R6

DIV R2, R3 ;Faz a divisao de R2 por 10, retirando assim o ultimo caracter

MOD R6, R3 ;Faz o resto da divisao de R6 por 10, ficando assim com o ultimo caracter

ADD R6, R4 ;adiciona ao ultimo caracter 48 para traduzir o numero para ascii para aparecer direito no display

MOVB [R0], R6 ;adiciona no 1ºbyte do endereço display o caracter traduzido

SUB R5, 1 ;subtrai 1 a R5 para dizer que já preenchemos mais um digito

SUB R0, 1 ;subtrai o endereço na posição do display pois escrevemos o numero da direita para a esquerda

CMP R5, 0 ;compara R5 com 0

JGT CicloEscrevePEPETalao ;caso seja maior vai escrever mais um digito

POP R6 ;recupera o valor de R6

POP R5 ;recupera o valor de R5

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;-----------------------

;Rotina dinheiro a Pagar

;-----------------------

DinheiroAPagar:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

PUSH R5 ;armazena valor de R5 na StackPointer

PUSH R6 ;armazena valor de R6 na StackPointer

MOV R0, PrecFinal ;Atribui a R0 o endereço do preço final

MOV R6, DinheiroInserido ;Atribui a R6 o endereço do dinheiro inserido

MOV R1, [R0] ;Atribui a r1 a conta a pagar

MOV R0, [R6] ;atribui a R0 o dinheiro inserido

SUB R1, R0 ;conta a pagar menos dinheiro inserido

MOV R2, PrecPagamento ;Atribui a R2 o endereço onde vai aparecer no display o dinehiro que falta pagar

MOV R3, 10 ;Atribui a R3 o valor 10

MOV R4, 5 ;Atribui a R4 o valor 5 pois são os caracteres a serem usados no display da direita para a esqueda, sendo que a 3ª posição é o ponto para diferenciar os euros dos centimos

MOV R5, 48 ;48 em decimal é 30 em hexadecimal(onde começa a escrever 0 no display)

cicloPreenche:

MOV R0, R1 ;Atribui a R0 o valor de R1

DIV R1, R3 ;Faz a divisao de R1 por 10, retirando assim o ultimo caracter

MOD R0, R3 ;Faz o resto da divisao de R0 por 10, ficando assim com o ultimo caracter

ADD R0, R5 ;adiciona ao ultimo caracter 48 para traduzir o numero para ascii para aparecer direito no display

MOVB [R2], R0 ;adiciona no 1ºbyte do endereço display o caracter traduzido

SUB R2, 1 ;subtrai o endereço na posição do display pois escrevemos o numero da direita para a esquerda

SUB R4, 1 ;subtrai 1 a R4 para dizer que já preenchemos mais um digito

CMP R4, 3 ;compara R4 com 3 pois 3 é a posição do ponto

JEQ ponto ;caso seja igual salta para a etiqueta ponto

CMP R4, 0 ;compara R4 com 0 a ver se chegou ao final

JGT cicloPreenche ;caso seja maior que 0 salta para a etiqueta cicloPreenche

JEQ fimDinAPagar ;caso chegue ao final salta para a etiqueta fimDinAPagar

ponto:

SUB R2, 1 ;subtrai o endereço na posição do display pois escrevemos o numero da direita para a esquerda

SUB R4, 1 ;subtrai 1 a R4 para dizer que já preenchemos mais um digito

JMP cicloPreenche ;salta para a etiqueta cicloPreenche

fimDinAPagar:

Pop R6 ;recupera o valor de R6

POP R5 ;recupera o valor de R5

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;------------------------

;Rotina Subtrair Dinheiro (R0 valor inserido)

;------------------------

SubtrairDin:

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

PUSH R5 ;armazena valor de R5 na StackPointer

CALL VerificaMoeda ;Chama a rotina para verificar a moeda

MOV R1, PrecFinal ;atribui a R1 o endereço do preço final

MOV R2, DinheiroInserido ;atribui a R2 o endereço do dinheiro inserido

MOV R3, [R1] ;atribui a R3 o conteudo de R1 (Preço a pagar)

MOV R4, [R2] ;atribui a R4 o conteudo de R2 (dinheiro inserido)

ADD R4, R0 ;Soma o valor de R4 com o de R0(atualiza dinherio inserido)

MOV [R2], R4 ;Atribui ao conteudo de R2 o valor de R4(guarda no endereço)

CMP R4, R3 ;Compara R4 com R3

JGE taPago ;caso o dinheiro inserido seja maior ou igual ao que tem de pagar salta para a etiqueta taPago

JMP fimSubtrairDin ;caso contrário salta para a etiqueta fimSubtrairDin

taPago:

MOV R5, Troco ;atribui a R5 o endereço do troco

SUB R4, R3 ;subtrair o dinheiro inserido com o total a pagar

MOV [R5], R4 ;atribui ao conteudo de R5 o valor de R4

MOV R7, 1 ;R7 fica com o valor 1 para processos de automatizaçao da compra

fimSubtrairDin:

POP R5 ;recupera o valor de R5

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

RET

;------------------------

;Rotina Introduzir moedas

;------------------------

moedasInicio:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

MOV R0, q10cent ;Atribui o endereço da quantidade de moedas de 10 cent

MOV R1, 8 ;quantidade de tipos de moedas(10 cent, 20 cent, 50 ...)

MOV R2, 5 ;no início teremos 5 de cada coisa(5 moedas de 10 cent, 5 moedas de 20 cent )

cicloAdicionaM:

MOVB [R0], R2 ;adiciona no 1ºbyte do endereço o valor 5

SUB R1, 1 ;subtrai a quantidade de moedas que ainda falta

ADD R0, 1 ;passa para o proximo endereço que é o da prox moeda

CMP R1, 0 ;compara com 0 pois se for maior vai continuar no ciclo

JGT cicloAdicionaM ;salto etiqueta cicloAdicionaM

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;-----------------------------------------------------

;Rotina verificar moeda/nota (R0 valor inserido)

;-----------------------------------------------------

VerificaMoeda:

PUSH R1 ;armazena valor de R1 na StackPointer

MOV R1, V10cent ;atribui a R1 o valor da constante de 10 centimos

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona10C ;se for igual salta para a etiqueta adiciona10C

MOV R1, V20cent ;atribui a R1 o valor da constante de 20 centimos

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona20C ;se for igual salta para a etiqueta adiciona20C

MOV R1, V50cent ;atribui a R1 o valor da constante de 50 centimos

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona50C ;se for igual salta para a etiqueta adiciona50C

MOV R1, V1euro ;atribui a R1 o valor da constante de 1 euro

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona1e ;se for igual salta para a etiqueta adiciona1e

MOV R1, V2euro ;atribui a R1 o valor da constante de 2 euros

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona2e ;se for igual salta para a etiqueta adiciona2e

MOV R1, V5euro ;atribui a R1 o valor da constante de 5 euros

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona5e ;se for igual salta para a etiqueta adiciona5e

MOV R1, V10euro ;atribui a R1 o valor da constante de 10 euros

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona10e ;se for igual salta para a etiqueta adiciona10e

MOV R1, V20euro ;atribui a R1 o valor da constante de 20 euros

CMP R1, R0 ;compara se é igual ao valor inserido

JEQ adiciona20e ;se for igual salta para a etiqueta adiciona20e

adiciona10C:

MOV R1, q10cent ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

JMP fimAdiciona ;salta para a etiqueta fimAdiciona

adiciona20C:

MOV R1, q20cent ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

JMP fimAdiciona ;salta para a etiqueta fimAdiciona

adiciona50C:

MOV R1, q50cent ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

JMP fimAdiciona ;salta para a etiqueta fimAdiciona

adiciona1e:

MOV R1, q1eur ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

JMP fimAdiciona ;salta para a etiqueta fimAdiciona

adiciona2e:

MOV R1, q2eur ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

JMP fimAdiciona ;salta para a etiqueta fimAdiciona

adiciona5e:

MOV R1, q5eur ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

JMP fimAdiciona ;salta para a etiqueta fimAdiciona

adiciona10e:

MOV R1, q10eur ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

JMP fimAdiciona ;salta para a etiqueta fimAdiciona

adiciona20e:

MOV R1, q20eur ;Atribui a R1 o endereço da quantidade de moedas no stock

CALL AdicionaStock ;chama a rotina para adicionar ao stock

fimAdiciona:

POP R1 ;recupera o valor de R1

RET

;------------------------------------------------------------------

;Rotina adicionar moeda/nota stock (R1 endereço do stock da moeda)

;-----------------------------------------------------------------

AdicionaStock:

PUSH R0 ;armazena valor de R0 na StackPointer

MOVB R0, [R1] ;R0 fica com a quantidade de moedas/notas

ADD R0, 1 ;Adicinoa 1 à quantidade

MOVB [R1], R0 ;Escreve no endereço a quantidade atualizada

POP R0 ;recupera o valor de R0

RET

;-----------------------------------------------------------------

;Rotina retira moeda/nota stock (R3 endereço do stock da moeda)

;-----------------------------------------------------------------

retiraStock:

PUSH R0 ;armazena valor de R0 na StackPointer

MOVB R0, [R3] ;R0 fica com a quantidade de moedas/notas

SUB R0, 1 ;Adicinoa 1 à quantidade

MOVB [R3], R0 ;Escreve no endereço a quantidade atualizada

POP R0 ;recupera o valor de R0

RET

;-----------------------------------------------------------------

;Rotina adicionar moeda/nota troco (R3 endereço do stock da moeda)

;-----------------------------------------------------------------

AdicionaTroco:

PUSH R0 ;armazena valor de R0 na StackPointer

MOVB R0, [R3] ;R0 fica com a quantidade de moedas/notas

ADD R0, 1 ;Adicinoa 1 à quantidade

MOVB [R3], R0 ;Escreve no endereço a quantidade atualizada

POP R0 ;recupera o valor de R0

RET

;------------------------------------------------------------

;Rotina para mostrar stock1(R7, num 1 a 3 referente à pagina)

;------------------------------------------------------------

MStock:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

CMP R7, 1 ;verifica se é a 1ª pagina

JEQ Sto1 ;caso o for salta para a etiqueta Sto1

CMP R7, 2 ;verifica se é a 2ª pagina

JEQ Sto2 ;caso o for salta para a etiqueta Sto2

CMP R7, 3 ;verifica se é a 3ª pagina

JEQ Sto3 ;caso o for salta para a etiqueta Sto3

Sto1:

MOV R0, d10c1e5e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q10cent ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d20c2e10e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q20cent ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d50c20e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q50cent ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

JMP fimMStock ;salta para a etiqueta fimMStock

Sto2:

MOV R0, d10c1e5e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q10eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d20c2e10e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q2eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

JMP fimMStock ;salta para a etiqueta fimMStock

Sto3:

MOV R0, d10c1e5e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q5eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d20c2e10e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q10eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d50c20e ;endereço do display a meter a quantidade moedas/notas

MOV R1, q20eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

JMP fimMStock ;salta para a etiqueta fimMStock

fimMStock:

POP R1

POP R0

RET

;------------------------------

;Rotina para por no endereço display(R0 endereço a colocar, R1 endereço da quantida)

;------------------------------

StoDisplay:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

PUSH R5 ;armazena valor de R5 na StackPointer

PUSH R6 ;armazena valor de R6 na StackPointer

MOVB R2, [R1] ;quantidade da moeda/nota

MOV R3, 2 ;serão 2 digitos para apresentar a quantidade

MOV R4, 48 ;utilizado para traduzir o nº para ascii

MOV R5, 10 ;usar para divisão e resto da divisão

cicloPreenceSt:

MOV R6, R2 ;R6 fica com o valor de R2

DIV R2, R5 ;R2 fica sem o ultimo digito

MOD R6, R5 ;R6 fica só com o ultimo digito

ADD R6, R4 ;traduz para ascii

MOVB [R0], R6 ;insere no display o digito

SUB R3, 1 ;decrementa uma posição

SUB R0, 1 ;posição vem uma para a esquerda porque escrevemos o numero da direita pa esquerda

CMP R3, 0 ;compara o R3 com 0

JGT cicloPreenceSt ;caso ainda haja digitos salta para a etiqueta cicloPreenceSt

POP R6 ;recupera o valor de R6

POP R5 ;recupera o valor de R5

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

RET

;------------------------------

;Rotina para dar troco

;------------------------------

DarTroco:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R3 ;armazena valor de R3 na StackPointer

PUSH R4 ;armazena valor de R4 na StackPointer

MOV R0, Troco ;endereço do troco a ser devolvido

MOV R1, [R0] ;R1 é o valor do Troco

CicloTroco:

CMP R1, 0 ;Caso não haja troco já sai

JEQ fimTroco

ret20e:

MOV R2, V20euro ;R2 tem o valor de 20 euros

CMP R2, R1 ;Compara o valor com o troco

JLE retira20e ;caso seja menor ou igual salta

ret10e:

MOV R2, V10euro ;R2 tem o valor de 10 euros

CMP R2, R1 ;Compara o valor com o troco

JLE retira10e ;caso seja menor ou igual salta

ret5e:

MOV R2, V5euro ;R2 tem o valor de 5 euros

CMP R2, R1 ;Compara o valor com o troco

JLE retira5e ;caso seja menor ou igual salta

ret2e:

MOV R2, V2euro ;R2 tem o valor de 2 euros

CMP R2, R1 ;Compara o valor com o troco

JLE retira2e ;caso seja menor ou igual salta

ret1e:

MOV R2, V1euro ;R2 tem o valor de 1 euros

CMP R2, R1 ;Compara o valor com o troco

JLE retira1e ;caso seja menor ou igual salta

ret50c:

MOV R2, V50cent ;R2 tem o valor de 50 cent

CMP R2, R1 ;Compara o valor com o troco

JLE retira50c ;caso seja menor ou igual salta

ret20c:

MOV R2, V20cent ;R2 tem o valor de 20 cent

CMP R2, R1 ;Compara o valor com o troco

JLE retira20c ;caso seja menor ou igual salta

ret10c:

MOV R2, V10cent ;R2 tem o valor de 10 cent

CMP R2, R1 ;Compara o valor com o troco

JLE retira10c ;caso seja menor ou igual salta

erroTro:

CALL ErroTroco ;Chama a mensagem erro pois não há troco sufeciente

MOV R2, PrecViagemGratis ;caso não haja moedas sufecientes é posto o restante do troco no saldo do cartão

MOV R3, [R2] ;R3 tem o valor do saldo que vai para o cartão

ADD R3, R1 ;adiciona o restante do troco que nao dá para dar

MOV [R2], R3 ;adiciona no conteudo do endereço o saldo + troco que nao deu

JMP fimTroco

retira20e:

MOV R3, q20eur ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ ret10e ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t20eur ; atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

retira10e:

MOV R3, q10eur ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ ret5e ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t10eur ; atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

retira5e:

MOV R3, q5eur ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ ret2e ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t5eur ; atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

retira2e:

MOV R3, q2eur ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ ret1e ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t2eur ; atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

retira1e:

MOV R3, q1eur ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ ret50c ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t1eur ; atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

retira50c:

MOV R3, q50cent ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ ret20c ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t50cent ;atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

retira20c:

MOV R3, q20cent ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ ret10c ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t20cent ;atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

retira10c:

MOV R3, q10cent ;atribui a R3 o endereço da quantidade de moedas em stock

MOVB R4, [R3] ;quantida de moeda/nota

CMP R4, 0 ;compara com zero

JEQ erroTro ;caso não haja passa prox

CALL retiraStock ;atualiza o stock

MOV R3, t10cent ;atribui a R3 o endereço da quantidade de moedas para troco

CALL AdicionaTroco ;atualiza o troco

SUB R1, R2 ;subtrai o valor ao troco

JMP CicloTroco

fimTroco:

POP R4 ;recupera o valor de R4

POP R3 ;recupera o valor de R3

POP R2 ;recupera o valor de R2

POP R1 ;recupera o valor de R1

POP R0 ;recupera o valor de R0

RET

;------------------------------

;Rotina para mostrar TROCO(R7, num 1 a 3 referente à pagina)

;------------------------------

MTroco:

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

CMP R7, 1 ;verifica se é a 1ª pagina

JEQ to1 ;caso o for salta para a etiqueta Sto1

CMP R7, 2 ;verifica se é a 2ª pagina

JEQ to2 ;caso o for salta para a etiqueta Sto2

CMP R7, 3 ;verifica se é a 3ª pagina

JEQ to3 ;caso o for salta para a etiqueta Sto3

to1:

MOV R0, d10c1e5e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t10cent ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d20c2e10e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t20cent ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d50c20e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t50cent ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

JMP fimMTroc ;salta para a etiqueta fimMStock

to2:

MOV R0, d10c1e5e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t1eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d20c2e10e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t2eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

JMP fimMStock ;salta para a etiqueta fimMStock

to3:

MOV R0, d10c1e5e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t5eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d20c2e10e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t10eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

MOV R0, d50c20e ;endereço do display a meter a quantidade moedas/notas

MOV R1, t20eur ;atribui a R1 o endereço da quantidade respetiva

CALL StoDisplay ;insere a quantidade no display

JMP fimMStock ;salta para a etiqueta fimMStock

fimMTroc:

POP R1

POP R0

RET

;---------------------------

;Rotina para mostrar o troco

;---------------------------

MostraTroco:

PUSH R2 ;armazena valor de R2 na StackPointer

PUSH R0 ;armazena valor de R0 na StackPointer

PUSH R1 ;armazena valor de R1 na StackPointer

PUSH R7 ;armazena valor de R7 na StackPointer

MOV R7, 1

MOV R2, MenuTROC1 ;atribui a R2 o endereço de MenuTROC1

CALL MostraDisplay ;coloca no display o MenuTROC1

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

CALL MTroco ;chama a rotina para aparecer as moedas a dar

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeSeguinteT:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeSeguinteT ;caso valor de R1 não seja 1 salta para etiqueta LeSeguinteT

MostraTroco2:

MOV R7, 2

MOV R2, MenuTROC2 ;atribui a R2 o endereço de MenuTROC2

CALL MostraDisplay ;coloca no display o MenuTROC2

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

CALL MTroco ;chama a rotina para aparecer as moedas a dar

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeSeguinteT2:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeSeguinteT2 ;caso valor de R1 não seja 1 salta para etiqueta LeSeguinteT

MostraTroco3:

MOV R7, 3

MOV R2, MenuTROC3 ;atribui a R2 o endereço de MenuTROC3

CALL MostraDisplay ;coloca no display o MenuTROC3

CALL LimpaPerifericos ;chama rotina LimpaPerifericos

CALL MTroco ;chama a rotina para aparecer as moedas a dar

MOV R0, PEN ;atribui a R0 o endereço do botão Ok

LeSeguinteT3:

MOVB R1, [R0] ;atribui a R1 o 1º byte do valor de R0

CMP R1, 1 ;compara R1 com 1

JNE LeSeguinteT3 ;caso valor de R1 não seja 1 salta para etiqueta LeSeguinteT

POP R7 ;recupera o valor a R7

POP R1 ;recupera o valor a R1

POP R0 ;recupera o valor a R0

POP R2 ;recupera o valor a R2

RET