Arquitetura de Computadores
Projeto: Balanca Digital em Assembly

Docente: Dionisio Barros

Joao Filipe Correia Andrade Sousa-2133023
Lucas Araujo-2147123

1. Introdução

Este relatório descreve o desenvolvimento de uma balança eletrónica comercial implementada com recurso ao processador PEPE-16 e a uma memória RAM externa. O objetivo principal do projeto foi simular o funcionamento de uma balança que mede o peso de produtos, calcula o preço total com base no preço por quilo, e apresenta funcionalidades como histórico, confirmação, alteração e cancelamento.

2. Descrição do Projeto

A balança foi desenvolvida utilizando exclusivamente o PEPE-16 e a RAM. Foram definidos botões simulados através de endereços específicos da RAM, que controlam o comportamento do sistema. Estes botões são:

- **LIGAR/DESLIGAR** – Ativa ou desativa a balança.

Joao Filipe Correia Andrade Sousa-2133023
Lucas Araujo-2147123

- **SELECIONAR** – Permite escolher entre três produtos disponíveis, cada um com um preço por quilo específico.
- **OK** – Confirma a seleção do produto e o registo da pesagem.
- **CANCEL** – Remove o último produto registado da compra.
- **CHANGE** – Apresenta o histórico da compra e permite finalizar e limpar os registos.

O peso é simulado e introduzido através de um endereço na RAM. Após a seleção de um produto, o peso é multiplicado pelo preço por quilo, e o total é registado e acumulado no histórico da compra.

3. Análise de Resultados

A implementação do projeto foi feita exclusivamente em Assembly para o processador PEPE-16. O código inicializa os periféricos e constantes utilizadas na lógica da balança, onde se destaca a separação clara entre os endereços de entrada (botões) e os dados de controlo (como limites de peso e caracteres ASCII).

Durante a execução:

- A balança permanece inativa até o botão **ON_OFF** ser pressionado, momento em que os valores são inicializados.
- O utilizador pode navegar entre três produtos com o botão **Sel_Nr_Menu**, cada um com um preço definido internamente.
- Após selecionar um produto e o peso, o botão **OK** e armazena a informação na memória.
- O sistema multiplica automaticamente o peso pelo preço do produto, utilizando rotinas específicas para multiplicação e formatação do valor total.
- O botão **CHANGE** mostra todos os produtos com o seu codigo de produto
- O botão **CANCEL** volta atras no menu.
- O botão **OK** quando estamos no menu limpar registos vai para um ecran de confirmacao e depois e percisos clicar outravez no **OK** para eleminar, ou **CANCEL** para voltar atras

O código está estruturado com comentários úteis que tornam clara a organização da RAM e a lógica de execução. A utilização de endereços constantes e máscaras para conversão e visualização demonstra um bom aproveitamento dos recursos limitados do PEPE-16.

4. Conclusão

O projeto da balança eletrónica comercial foi desenvolvido com sucesso, cumprindo todos os requisitos propostos. A implementação em Assembly para o PEPE-16 demonstrou-se eficaz na simulação de uma balança funcional, capaz de registar produtos, calcular preços com base no peso, apresentar o histórico de compras e permitir operações de correção e finalização.

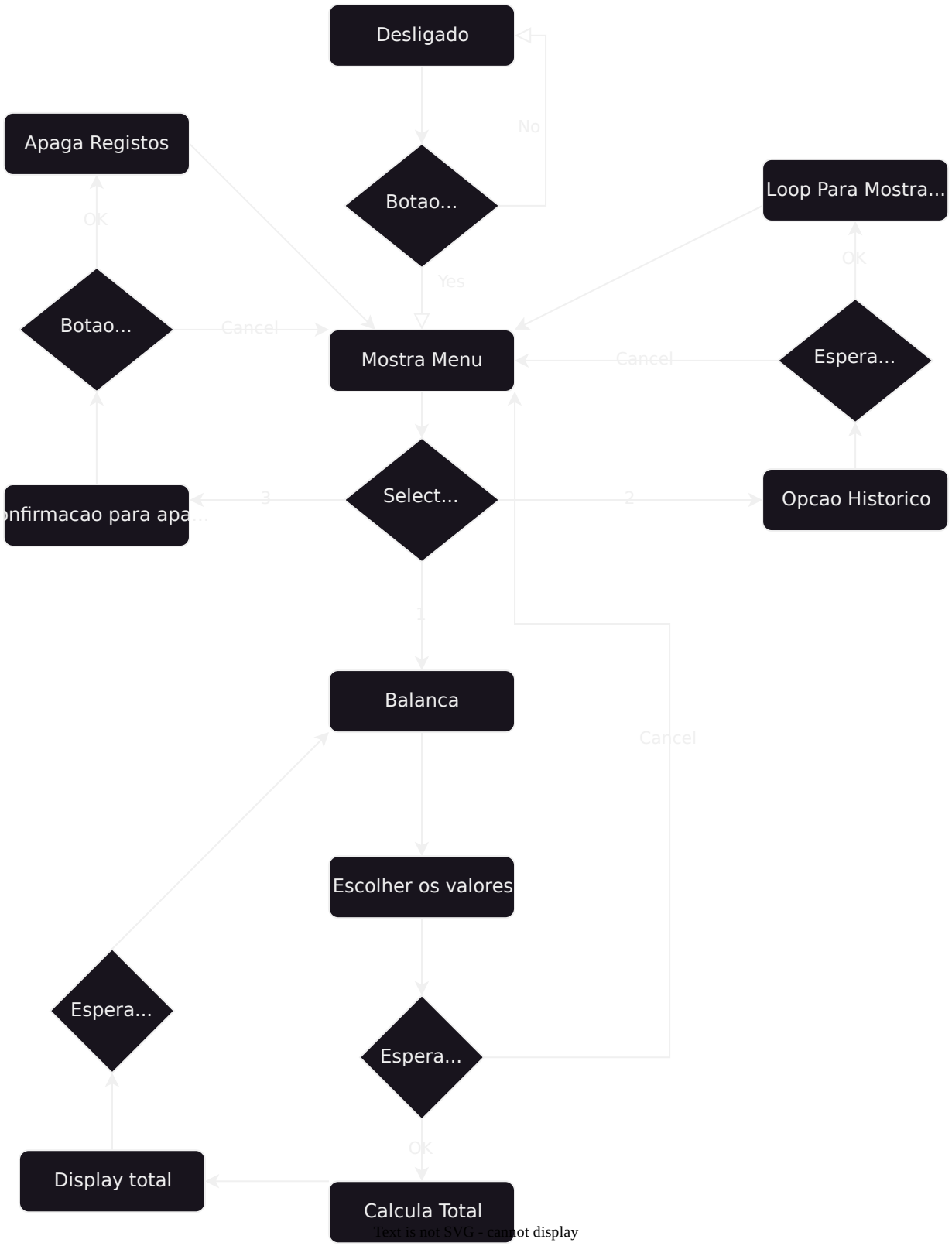
Todas as funcionalidades especificadas foram implementadas e testadas, incluindo:

- Sistema de ativação e desativação;

- Cálculo automático do valor da compra;
- Registo e visualização do histórico;
- Cancelamento e limpeza de registos.

O código foi devidamente comentado e organizado, sendo incluído como Anexo B, e os fluxogramas das principais rotinas estão apresentados no Anexo A. Através deste projeto, foi possível aplicar conceitos fundamentais de programação em Assembly, manipulação de memória, e controlo de entrada/saída, respeitando os constrangimentos do sistema e os objetivos propostos.

Anexo A: Fluxograma



Text is not SVG - cannot display

Joao Filipe Correia Andrade Sousa-2133023
Lucas Araujo-2147123

ANEXO B: Codigo em Assebmly

```
; Perifericos
ON_OFF                          EQU             190H;Endereco do botao ON_OFF
Sel_Nr_Menu             EQU             1A0H;Endereco do botao de select
OK                              EQU             1B0H;Endereco do botao Ok
CHANGE                          EQU             1C0H;Endereco do botao CHANGE
CANCEL                          EQU             1D0H;Endereco do botao CANCEL
PESO                    EQU             1E0H;Endereco do botao PESO
PRODUTO                         EQU         1F0H;Endereco do botao PRODUTO

;Constantes
LIMITEPESO              EQU             1000H; Endereco da contante LIMITEPESO

PONTOASCII              EQU             1002H; Endereco da contante PONTOASCII

MASCARAANTESVIRGULA     EQU             1004H; Endereco da contante
MASCARAANTESVIRGULA
MASCARADPSVIRGULA EQU               1006H; Endereco da contante
MASCARADPSVIRGULA
NUMERO0ASCII            EQU             1008H; Endereco da contante
NUMERO0ASCII
NUMERO9ASCII            EQU             100AH; Endereco da contante
NUMERO9ASCII
NUMERO100               EQU             100CH; Endereco da contante NUMERO100

NUMERO10                EQU             100EH; Endereco da contante NUMERO10

NUMEROPRODUTOS          EQU             1010H; Endereco da contante
NUMEROPRODUTOS
ESPACOASCII             EQU             1012H; Endereco da contante
ESPACOASCII
NUMERO10K               EQU         1014H; Endereco da contante NUMERO10K
NUMERO1K                EQU         1016H; Endereco da contante NUMERO1K
PESODISPLAY             EQU         1018H; Endereco da contante PESODISPLAY
LETRAEASCII             EQU             101AH; Endereco da contante
LETRAEASCII


;Memoria
INICIOPRODUTOS          EQU             0300H; Endereco de onde esta o inicio
dos produtos
INCERMENTOPRODUTOS      EQU             0060H; Endereco ond esta distancia
entre cada produto
MUDANCACPM              EQU             0062H; isto e simplemente para mudar
de
;codigo de produto para um valor na memoria
DISTANCIAPESO           EQU             0064H; Enderenco onde estaa distancia
que o peso esta do inicio do produto
DISTANCIAPRECO          EQU             0066H; Enderenco onde estaa distancia
que o preco esta do inicio do produto
DISTANCIATOTAL          EQU             0068H; Enderenco onde estaa distancia
que o total esta do inicio do produto
```

```
; Display
Display                         EQU             210H; Endereco de onde esta o
inicio do display
Display_end             EQU             27FH; Endereco de onde esta o fim do
display
CaracterVazio           EQU         20H             ; Caracter para limpar o
ecra

MBalanca                EQU         1               ; Opcao de Balanca
MRegistos               EQU         2               ; Opcao de Registos
OLimpa                          EQU         3               ; Opcao de Limpar

;StackPointer          EQU         6000H

Place 0060H
incrementos:
      WORD 112 ; Incrementeo em decimal dos produtos
      WORD 100;  Para tirar 100 do codigo do produto
      WORD 36; a distancia em decimal do peso
      WORD 68; a distancia em decimal do preco
      WORD 102; a distancia em decimal do total
; Strings para ajudar a ler as posicoes memoria
Place 0180H
MostraBotoes:
      String "Botoes em baixo "
Place 0192H
BOnOff:
      String "On_off        "

Place 01A2H
BSel_Nr_Menu:
      String "Sel_Nr_Menu   "

Place 01B2H
BOk:
      String "Ok            "

Place 01C2H
BChange:
      String "Change        "

Place 01D2H
BCancel:
      String "Cancel        "

Place 01E2H
BPeso:
      String "Peso          "

Place 01F2H
BProduto:
      String "Produto       "
```

Joao Filipe Correia Andrade Sousa-2133023
Lucas Araujo-2147123

```
;Tabelas dos Precos e Nomes
Place 0300H
Uvas:
      String "100   Uvas      "
      String "Peso :           "
      String "              KG"
      String "Preco:           "
      String "      5.34EUR/KG"
      String "Total:           "
      String "              UR"
Place 0370H
Melancia:
      String "101 Melancia    "
      String "Peso :           "
      String "              KG"
      String "Preco:           "
      String "      1.87EUR/KG"
      String "Total:           "
      String "              UR"

Place 03E0H
Ananas:
      String "102  Ananas     "
      String "Peso :           "
      String "              KG"
      String "Preco:           "
      String "      1.87EUR/KG"
      String "Total:           "
      String "              UR"
Place 0450H
Kiwi:
      String "103   Kiwi      "
      String "Peso :           "
      String "              KG"
      String "Preco:           "
      String "      3.56EUR/KG"
      String "Total:           "
      String "              UR"
Place 04C0H
Pessego:
      String "104  Pessego    "
      String "Peso :           "
      String "              KG"
      String "Preco:           "
      String "      4.46EUR/KG"
      String "Total:           "
      String "              UR"
Place 0530H
Banana:
      String "105  Banana     "
      String "Peso :           "
      String "              KG"
      String "Preco:           "
      String "     2.58EUR/KG"
      String "Total:           "
```

```
      String "              UR"

Place 05A0H
Morango:
      String "106  Morango    "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "     4.46EUR/KG"
      String "Total:          "
      String "              UR"

Place 0610H
Framboesa:
      String "107 Framboesa   "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "     17.81EUR/KG"
      String "Total:          "
      String "              UR"

Place 0680H
Laranja:
      String "108  Laranja    "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      1.60EUR/KG"
      String "Total:          "
      String "              UR"

Place 06F0H
Tangerina:
      String "109 Tangerina   "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      2.22EUR/KG"
      String "Total:          "
      String "              UR"

Place 0760H
Cenoura:
      String "110  Cenoura    "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      1.04EUR/KG"
      String "Total:          "
      String "              UR"

Place 07D0H
Batata:
      String "111  Batata     "
```

```
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      1.14EUR/KG"
      String "Total:          "
      String "              UR"

Place 0840H
Nabo:
      String "112   Nabo      "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      2.28EUR/KG"
      String "Total:          "
      String "              UR"

Place 08B0H
Beterraba:
      String "113 Beterraba   "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      5.23EUR/KG"
      String "Total:          "
      String "              UR"

Place 0920H
Alho:
      String "114   Alho      "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      6.19EUR/KG"
      String "Total:          "
      String "              UR"

Place 0990H
Cebola:
      String "115  Cebola     "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "     1.43EUR/KG"
      String "Total:          "
      String "              UR"

Place 0A00H
Ervilha:
      String "116  Ervilha    "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      1.42EUR/KG"
      String "Total:          "
```

```
      String "              UR"

Place 0A70H
Lentilhas:
      String "117 Lentilhas   "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      2.19EUR/KG"
      String "Total:          "
      String "              UR"


Place 0AE0H
Trigo:
      String "118   Trigo     "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      0.95EUR/KG"
      String "Total:          "
      String "              UR"

Place 0B50H
Milho:
      String "119   Milho     "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      3.62EUR/KG"
      String "Total:          "
      String "              UR"

Place 0BC0H
Favas:
      String "120   Favas     "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      4.07EUR/KG"
      String "Total:          "
      String "              UR"

Place 0C30H
Castanhas:
      String "121 Castanhas   "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      8.92EUR/KG"
      String "Total:          "
      String "              UR"

Place 0CA0H
Noz:
```

```
      String "122    Noz      "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "     18.39EUR/KG"
      String "Total:          "
      String "              UR"



Place 0D10H
Amendoim:
      String "123 Amendoim    "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "      8.03EUR/KG"
      String "Total:          "
      String "              UR"

Place 0D80H
Cafe:
      String "124   Cafe      "
      String "Peso :          "
      String "              KG"
      String "Preco:          "
      String "     20.25EUR/KG"
      String "Total:          "
      String "              UR"


;Aqui estao guardadas as constantes de antes
Place 1000H
Constantes:
      WORD 30000; maior que 30 kg
      WORD 46     ; PONTO EM ASCII
      WORD 255; 00FFH
      WORD 65280; FF00H
      WORD 48     ;NUMERO 0 EM ASCII
      WORD 57 ;NUMERO 9 EM ASCII
      WORD 100; Valor a ser comparado para as decimas
      WORD 10 ; Valor a ser Multiplicado p
      WORD 26 ;Numero de produtos +1
      WORD 32 ; escpaco em ASCII
      WORD 10000; 10k em decimal para fazer comparacoes
      WORD 1000; 1k em decimal para fazer comparacoes
      WORD 106; distancia para o display do peso
      WORD 69; Letra E em ASCII
;Aqui estao os menus
Place 2000H
MenuInicio:
      String " MENU PRINCIPAL "
      String "1 - BALANCA     "
      String "2 - REGISTOS    "
      string "----------------"
```

```
      String "3 - LIMPAR      "
      String "    REGISTOS    "

Place 2080H
MenuBalanca:
      String "  MENU BALANCA  "
      String "   POR FAVOR    "
      String "    INSIRA      "
      String "   O PESO E O   "
      String "    PRODUTO     "
      String "O Peso Atual e :"
      String "              KG"

Place 2100H
MenuErro:
      String "    ATENCAO     "
      String "                "
      String "    OPCAO       "
      string "    ERRADA      "
      String "                "
      String "                "
      String "                "

Place 2180H
MenuDadosErrados:
      String "                "
      String "    ATENCAO     "
      String "    VERIFIQUE   "
      String "    OS DADOS    "
      String "    INSERIDOS   "
      String "                "
Place 21F0H
MenuConfirmacaoClear:
      String "                "
      String "    ATENCAO     "
      String "    ISTO IRA    "
      String "    APAGAR TODOS"
      String "    OS DADOS    "
      String "    INSERIDOS   "
Place 2270H
MenuOverFlow:
      String "                "
      String "    ATENCAO     "
      String "    COM O PESO  "
      String "   ATUAL OCORREU"
      String "    UM OVERFLOW "
      String "                "
;isto e para conseguir ir para o principio do programa sem erros
Place 0000H
Inicio:
            MOV R0, Principio
            JMP R0
;onde o programa comeca
Place 3000H
;inicializar o stackpoint para o Call funcuonar
```

```
pilha:
	STACK 50H;
StackPointer:

Principio:
	MOV SP, StackPointer; para inicializar o stackpointer(nao usado depois
retirar)
	CALL LimpaDisplay;Apagar tudo o que esta escrito
	CALL LimpaPerifericos; apagar tudo dos perifericos
	MOV R0, ON_OFF; Coloca o botao de on Em R0
Liga:
	MOVB R1, [R0]; coloca o valor em R1
	CMP R1, 1; verifica se esta presionado
	JNE Liga; se nao estiver entao continua no liga
ligado:
	MOV R2, MenuInicio ;coloca em R2 o menu de inicio
	CALL MostraDisplay; mostra esse menu
	CALL LimpaPerifericos; limpa os perifericos
Le_Opcao:
	MOV R0, ON_OFF; Coloca o endereco do On off em R0
	MOVB R1, [R0]; Colocao valor do botao em R1
	CMP R1, 1; verifica se esta ligado
	JEQ Principio; se estiver entao desliga
	MOV R0, Sel_Nr_Menu; coloca em R0 o endereco do select
	MOV R3, [OK]; Coloca em R3 o valor do ok
	MOVB R1, [R0]; Coloca em R1 o valor do on off
	CMP R3,0; Caso o OK esteja a 0 entao volta atras
	JLE Le_Opcao;volta atras
	CMP R1, 0;caso nada seja selecionado volta atras
	JEQ Le_Opcao; volta atras
	CMP R1, MBalanca;OPCAO 1
	JEQ BufferBalanca;vai para o buffer da balanca
	CMP R1, MRegistos;OPCAO 2
	JEQ BufferRegistos; vai para o buffer de registos
	CMP R1, OLimpa; Opcao 3
	CALL ConfirmacaoClear; vai para a confirmacao do clera
	CMP R1,1; se o R1 for a 1 quer dizer que foi recebido o valor para mostrar
o menu inicial
	JEQ ligado; ; vai para o inicio para mostrar o menu
	CALL RotinaERRO;caso nada disso de certo etnao vai para a rotina de erro
	JMP ligado; vai para o menu inicial

BufferRegistos:
	CALL LimpaPerifericos; Limpa os perifericos
	CALL MostraRegistos; vai para a rotina de mostra regisots
	MOV R2, MenuInicio; depois mostra o menu inicial
	CALL MostraDisplay; da display do menu
	CALL LimpaPerifericos; limpa os perifericos outraves
	CMP R1,1 ;Compara com 1, caso seja 1 desliga; para verificar se e para
desligar
	JEQ Principio; Volta ao desligado
	JMP Le_Opcao; caso nao seja entao volta so para a opvao

BufferBalanca:
	MOV R2, MenuBalanca; coloca o endereco da balanca em R2
```

```
      CALL MostraDisplay ;mostra o menu
      CALL LimpaPerifericos; limpa os perifericos
      JMP OBalanca; volta para a balanca

BufferMostraDisplayProdutos:
      CALL MostraDisplayProdutos; Chama a funcao para mostrar os produtos
      MOV R2, MenuBalanca;Guarda a posicao def moemoria do menu inicial
      CAll MostraDisplay;Mostra o menu de escolha do peso e do Produto
      CALL LimpaPerifericos; limpa os perifericos
      JMP BufferVoltaBalanca; Volta para a balanca

;------------------
;   Rotinas Erro
;------------------
RotinaERRO:
      PUSH R0
      PUSH R1
      PUSH R2
      MOV R2, MenuErro
      CALL MostraDisplay
      CALL LimpaPerifericos
      MOV R0, OK
ERRO:
      MOVB R1, [R0]
      CMP R1, 1
      JNE ERRO
      POP R2
      POP R1
      POP R0
      RET

RotinaERROBalanca:
      PUSH R8
      PUSH R2
      MOV R8,0
      MOV R2, MenuDadosErrados
      CALL MostraDisplay
      CALL LimpaPerifericos

      POP R2;
      POP R8;
      RET
; para mostrar o que esta no enderco R2
MostraDisplay:
      PUSH R0
      PUSH R1
      PUSH R3
      MOV R0, Display
      MOV R1, Display_end
Ciclo:
      MOV R3, [R2]
      MOV [R0], R3
      ADD R2, 2
      ADD R0, 2
      CMP R0, R1
```

```
      JLE Ciclo
      POP R3
      POP R2
      POP R1
      RET


;----------------------
;MostraDisplay Produtos
;----------------------
MostraDisplayProdutos:
      PUSH R0
      PUSH R1
      PUSH R2
      PUSH R3
      PUSH R4
      PUSH R5
      PUSH R6
      PUSH R7
      PUSH R8
      PUSH R9
      PUSH R10
      PUSH R11
      PUSH TEMP
      MOV R0, OK;Coloca temporariamente o endereco do OK em R0
      MOV R1,0; Coloca temporariamente o 0 em R1
      MOV [R0], R1 ;Volta a colocar o Ok a 0
      MOV R0, Display; Coloca em R0 o  apontador para o inicio do display
      MOV R1, Display_end; coloca em R1 o apontador para o fim do disply
      MOV R2, INICIOPRODUTOS; Coloca o apontador de R2 para o inicio dos
produtos
      MOV R6, 0; inicia o Contador para saber em qual produto estamos
      MOV R7, [MUDANCACPM]; Coloca em R7 100  por causa do codigo do produto
      MOV R8, [INCERMENTOPRODUTOS]; Coloca em R8 o valor para passar ao proximo
produto
      MOV R9, [NUMEROPRODUTOS]; Coloca o numero de produtos em R9
      MOV R10, 9; Coloca 8 em R10 para verificar se ja acabou a linha
      MOV TEMP, 0; Coloca a variavel temporaria a 0 para ser o contador
      JMP LoopLinhaDisplay; Carrega a primeira parte
BufferLoopDisplay:
      MOV R11,0;COloca o R11 a 0
      MOV [OK],R11; Coloca O botao de OK a 0
      SUB R2, 4; Volta as primeiras 4 casas
      SUB R2, 4; Volta as primeiras 4 casas
      SUB R2, 4; Volta as primeiras 4 casas
      SUB R2, 4; Volta ao inicio do priduto
      ADD R2, R8; Passa ao proximo produto
LoopDisplay:
      MOV R0, Display; Coloca em R0 o  apontador para o inicio do display
      MOV TEMP,CANCEL ;COloca a variavel temporaria a apontar para o cancel
      MOVB R5, [TEMP]; Coloca em R5 o registo que esta em CANCEL
      MOV TEMP, OK; COloca o apontador par ok em TEMP
      MOVB R4, [TEMP]; COloca em R4 o valor do enderenco Temp
      MOV TEMP, 0; Coloca a variavel temporaria a 0
      CMP R5, 1; Verifica se o utilizador quer cancelar
```

```
      JEQ AcabaDisplay; AcabR2a o display e volta a aparecer o menu
      CMP R4, 0; Verifica se o utilizador quer continuar
      JLE LoopDisplay; Volta atras caso nao quira
      CALL LimpaDisplay; para limpar o display
LoopLinhaDisplay:
      ADD R6, 1; Incrementa 1 nos produtos
      CMP R9, R6; Verifica se ja acabou os produtos
      JLE BufferAcabaDisplay; Se ja estiver ultrapassado o numero de produtos
acaba
BufferLinhaDisplay:
      ADD TEMP, 1; Adiciona 1 a variavel temporaria
      CMP TEMP, R10; verifica se a primeira linha ja foi escrita
      JEQ BufferDisplayLinha; Vai incrementar para o poximo produto e depois
voltar atras;
      MOV R4, [R2]; Coloca em R4 o valor que esta em R2, estou a reutilizar para
poupar memoria
      MOV [R0], R4; Coloca no Display o valor de R2
      ADD R2, 2; avanca na casa do prodito
      ADD R0,2; avanca no display
      CMP R0, R1; Se o display ja acabou entao volta atras
      JLE BufferLinhaDisplay; Repete o processo para dar print em tudo
      JMP BufferLoopDisplay; Volta tras para a opcao de prosseguir
BufferDisplayLinha:
      SUB R2, 4; Volta as primeiras 4 casas
      SUB R2, 4; Volta as primeiras 4 casas
      SUB R2, 4; Volta as primeiras 4 casas
      SUB R2, 4; Volta ao inicio do priduto
      ADD R2, R8; Passa ao proximo produto
      MOV TEMP,0;Voolta a colocar o contador a 0
      JMP LoopLinhaDisplay; passa a proxima linha do display
BufferAcabaDisplay:
      MOV R11,0;Coloca o R11 a 0
      MOV [OK],R11; Volta a colocar o Ok a 0 para esperar o input do utilizador
LoopEsperaPorInput:
      MOV TEMP, OK; Le o ok
      MOVB R0,[TEMP];Coloca o valor do OK em R0
      CMP R0, 0;Se o R0 estiver a 1 acaba o display
      JEQ LoopEsperaPorInput; Caso contrario volta atras
AcabaDisplay:
      POP TEMP
      POP R11
    POP R10
      POP R9
      POP R8
      POP R7
      POP R6
      POP R5
      POP R4
      POP R3
      POP R2
      POP R1
      POP R0
      CALL LimpaPerifericos; para limpar os perifericos
      RET
;--------------------
```

```
; Limpa Perifericos
;--------------------
LimpaPerifericos:
      PUSH R0
      PUSH R1
      PUSH R2
      PUSH R3
      PUSH R4
      PUSH R5
      PUSH R6
      MOV R0, ON_OFF; coloca em R0 o endereco de ON_OFF
      MOV R1, Sel_Nr_Menu; coloca em R1 o endereco de Sel_Nr_Menu
      MOV R2, OK; coloca em R2 o endereco de OK
      MOV R3, CHANGE; coloca em R3 o endereco de CHANGE
      MOV R4, CANCEL; coloca em R4 o endereco de CANCEL
      MOV R5, PESO ; coloca em R5 o endereco de PESO
      MOV R6, PRODUTO; coloca em R6 o endereco de PRODUTO
      MOV R7, 0; coloca em R7 0
      MOV R8, 0; coloca em R8 0
CicloLimpaPerifericos:; Para limpar os 2 brimeiros bits
      ADD R0,R8; Isto e para limpar o peimeiro depois e inrementado para limpar
o segundo
      ADD R1,R8; Isto e para limpar o peimeiro depois e inrementado para limpar
o segundo
      ADD R2,R8; Isto e para limpar o peimeiro depois e inrementado para limpar
o segundo
      ADD R3,R8; Isto e para limpar o peimeiro depois e inrementado para limpar
o segundo
      ADD R4,R8; Isto e para limpar o peimeiro depois e inrementado para limpar
o segundo
      ADD R5,R8; Isto e para limpar o peimeiro depois e inrementado para limpar
o segundo
      ADD R6,R8; Isto e para limpar o peimeiro depois e inrementado para limpar
o segundo
      MOVB [R0], R7; para limpa o primeiro bit
      MOVB [R1], R7; para limpa o primeiro bit
      MOVB [R2], R7; para limpa o primeiro bit
      MOVB [R3], R7; para limpa o primeiro bit
      MOVB [R4], R7; para limpa o primeiro bit
      MOVB [R5], R7; para limpa o primeiro bit
      MOVB [R6], R7; para limpa o primeiro bit
      ADD R8, 1 ; passa para o segundo
      CMP R8, 1; verifica se ja limpou os 2
      JEQ CicloLimpaPerifericos; se nao volta a fazer
      POP R6
      POP R5
      POP R4
      POP R3
      POP R2
      POP R1
      POP R0
      RET
;------------------
;  Limpa Display
;------------------
```

```
LimpaDisplay:
      PUSH R0
      PUSH R1
      PUSH R2
      MOV R0, Display; Guarda o inicio do display e R0
      MOV R1, Display_end; Guarda o fim do display e R1
CicloLimpa:
      MOV R2, CaracterVazio; coloca 20H em R2
      MOVB [R0], R2; Coloca no enderco R0 o caracgter vazio
      ADD R0, 1;anda 1 para a direita
      CMP R0, R1; enquanto nao acabou vai limpando
      JLE CicloLimpa; voltar a limpar
      POP R2
      POP R1
      POP R0
      RET

;---------------------------------------------------------------
;Buffers para voltar ao inicio
;---------------------------------------------------------------
BufferCancel:JMP ligado ; buffer porque nao da para saltar mais que 100H na
memoria
BufferDesliga: JMP Principio; buffer porque nao da para saltar mais que 100H na
memoria
BufferMostraDisplay: JMP BufferMostraDisplayProdutos; buffer porque nao da para
saltar mais que 100H na memoria
BufferVoltaBalanca: JMP OBalanca; buffer porque nao da para saltar mais que 100H
na memoria

;-----------------------------------------------------
;Isto e simplemente para dar clear nos registos pois
;estamos a assumir que so pode ter uma vez cada fruta
;e o Total e o registo que vai ficar
;-----------------------------------------------------

ConfirmacaoClear:
      MOV R2, MenuConfirmacaoClear ;coloca o apontador para o inicio do menu em
R2
      CALL MostraDisplay; Mostra o menu
      CALL LimpaPerifericos; limpa os perifericos
LoopClear:
      MOV TEMP,OK; COloca em TEMP o endereco de OK
      MOVB R3,[TEMP]; Le o botao de OK
      CMP R3, 1; Verifica se esta a 1
      JEQ ClearRegistos;Apaga todos os registos
      MOV TEMP, CANCEL; COloca o endereco de Cancel EM TEMP
      MOVB R3, [TEMP]; le o botao de cancelar em r3
      CMP R3, 1; Verifica se o botao de cancelar esta ativo
      JEQ BufferClear; Volta para o menu anterior
      JMP LoopClear; Volta atras caso o utilizador nao escolha
ClearRegistos:
      MOV R0, [INCERMENTOPRODUTOS]; Coloca em R0, o incremento para o proximo
produto
      MOV R10, [NUMEROPRODUTOS]; Coloca o numero de produtos em R10
      MOV R11, 0; Coloca em R11 o contador
```

```
      MOV R2, INICIOPRODUTOS; coloca o apontador para o incio em R2
LoopRemover:
      ADD R11,1; Adiciona 1 ao contador
      CMP R11,R10; COmpara para ver se ja cheou ao final
      JEQ BufferClear; ACaba
      MOV R5,R2; Copia tambem para o R5
      MOV R6, [DISTANCIAPESO]; Coloca em R6 a disancia ao peso
      MOV R7,CaracterVazio ; Colloca 20H em R7
      SHL R7, 8; Vai 2 casas para a esquerda
      MOV TEMP, CaracterVazio; Coloca o caracter vazio em TEMP
      ADD R7,TEMP;Coloca em R7 2020H
      ADD R5, R6; Vai para o Peso
      MOV [R5], R7; Apaga a primeira casa do peso
      ADD R5, 2;avanca
      MOV [R5], R7;apaga a segunda casa do peso
      ADD R5, 2;avanca
      MOV [R5], R7;apaga a terceira casa do peso
      ADD R5, 2;avanca
      MOV [R5], R7;apaga a Quarta casa do peso
      ADD R5, 2;avanca
      MOV [R5], R7;apaga a quinta casa do peso
      MOV R6, [DISTANCIATOTAL]
      MOV R5,R2; Coloca em R5 o o produto atual
      ADD R5,R6; vai para o total
      MOV [R5], R7; Apaga o total
      ADD R5, 2;avanca
      MOV [R5], R7; Apaga o total
      ADD R5, 2;avanca
      MOV [R5], R7; Apaga o total
      ADD R5, 2;avanca
      MOV [R5], R7; Apaga o total
      ADD R5, 2;avanca
      MOV [R5], R7;
      ADD R2,R0; passa para o proximo produto
      JMP LoopRemover; volta a fazer ate acababar os produtos
BufferClear:
      MOV R1,1; coloca a 1 para mostrar o menu inicial
      RET

;------------------
;      Balanca
;------------------
BufferErro:
      CALL RotinaERROBalanca; Para chamar a rotina de erro
      CALL Espera; esperar paar o input de ok
      MOV R2, MenuBalanca; coloca o endereco do menu da balanca em R2
      CALL MostraDisplay; mostra o menu
      CALL LimpaPerifericos; limpa os perifericos
      JMP OBalanca; vai para a balanca
OBalanca:
      ;carrega o valor dos perifericos
      CALL MostraPeso; Mostra o peso no display
      MOV R0, CANCEL; Carrega o botao de Cancel em R0
      MOVB R1, [R0]; Carrega o valor do botao cancel
      CMP R1,1 ;verifica se esta pressionao
```

```
	JEQ BufferCancel;
	MOV R0, ON_OFF; Carrega o botao de Onoff em R0
	MOVB R1, [R0]; Carrega o valor do botao cancel
	CMP R1,1 ;verifica se esta pressionao
	JEQ BufferDesliga;
	MOV R0,CHANGE; CARREGA o botao change em R0
	MOVB R1, [R0]; carrega o que esta gravado no endereco
	CMP R1, 1; Verifica se esta a 1
	JEQ BufferMostraDisplay;
	MOV R0, PRODUTO; Coloca em R0 o apontador para o produto
	MOV R1, PESO ;Coloca em R1 o apontador para o peso
	MOV R7, OK; Coloca o em R7 o apontador parabotao de ok
	MOV R9, [NUMEROPRODUTOS]; Coloca em R9 24, que e o numero de produtos
	MOVB R8, [R7]     ; R8 = BOTAO DE OK
	MOVB R2, [R0]     ; R2 = PRODUTO
	MOV R3, [R1]      ; R3 = PESO
	MOV R4, [LIMITEPESO]
	CMP R8, 0; verifica se o botao de ok esta a 1
	JLE OBalanca
	CMP R2, 0; VERIFICAR SE O PRODUTO ESTA A 00
	JLE OBalanca;volta atras
	CMP R3,0; VERIFICAR SE O PESO ESTA A 00
	JLE OBalanca;Volta atras
	MOV R4, [MUDANCACPM]; Guarda em R4, a mudanca de Codifo para produto
	SUB R2,R4; aqui fica guardado no r2 qual e o produto
	MOV TEMP, [LIMITEPESO]; guarda o limite de peso
	CMP TEMP, R3      ;VERIFICAR SE O PESO ULTRAPASSA 30000 QUE E 7530H
	JLE BufferErro;Vai para erro se o peso for maior que 30000
	CMP R9, R2; Se o valor recebido em R0 for maior que R9 quer dizer que o
produto nao existe
	JLE BufferErro ;Da erro caso isso aconteca
	MOV R5, INICIOPRODUTOS ; onde vamos guardar onde estamos nos menus
	MOV R6, [INCERMENTOPRODUTOS]  ; o valor que vai incrementar nos menus

CicloEncontraFruta:
	CMP R2,0; Verifica se ja encontrou a fruta
	JLE BufferDisplay; se encontrar mostra
	ADD R5,R6; se nao entao passa para o proximo produto
	SUB R2, 1; tira um para passar ao proximo produto
	JMP CicloEncontraFruta; volta a fazer o ciclo
BufferDisplay:
	MOV R2, R5; Coloca o produto em R2
	CALL EditarPrint; aqui faz as contas e escreve o resultado
	CALL MostraDisplay; mostra o resultado
	CALL LimpaPerifericos;limpa os perifericos
	CALL Espera; espera o ok

	MOV R2, MenuBalanca; coloca o endereco do menu em R2
	CALL MostraDisplay; mostra o menu
	CALL LimpaPerifericos;limpa os perifericos outravez para evitar erros
	JMP OBalanca; volta a balanca

ERROOVERFLOW:
	POP TEMP
	POP R11
```

```
      POP R10
      POP R9
      POP R8
      POP R7
      POP R6
      POP R2
      MOV R2, MenuOverFlow; acaba o editar print e mostra o menu de erro
Overflow
      RET
EditarPrint:
      PUSH R2
      PUSH R6
      PUSH R7
      PUSH R8
      PUSH R9
      PUSH R10
      PUSH R11
      PUSH TEMP

      MOV R6, [DISTANCIAPESO]; Coloca a distancia para escrever o peso
      ADD R6,4; Para colocar no fim da linha
      MOV R7, R5; coloca no produto certo
      MOV R8, [MASCARAANTESVIRGULA];colocaar em R8 00FF
      MOV R9, [MASCARADPSVIRGULA]; colocar em R9 FF00
      MOV R10,[NUMERO0ASCII] ; COLOCAR O NUMERO 0 ASCII EM R10 MUDAR ISTO PARA
DINAMICO DEPOIS
      MOV TEMP, [PONTOASCII];coloca o Ponto em ASCII na variavel temporaria
      AND R8, R3 ; Vai buscar os digitos antes da virgula
      AND R9, R3 ;Vai buscar os numeros depois da virgula
      SHR R9,8    ;coloca os bits na parte menos significativa
      ADD R7,R6   ;passa para a casa para escrever o peso

      MOV TEMP, [PESODISPLAY]; coloca o temp a distancia que o peso esta escrito
      MOV R1, Display; coloca em R1 onde esta o dispalay
      ADD R1, TEMP; Coloca o temp onde esta escrito

      MOV R6, [DISTANCIAPESO]; mete em r6 a distancia ao peso
      MOV TEMP,[R1];Coloca em TEMP o que esta em R1
      MOV [R7],TEMP; Copia de antes
      ADD R7,2; avanca
      ADD R1,2;avaca
      MOV TEMP,[R1];Coloca em TEMP o que esta em R1
      MOV [R7],TEMP; Copia de antes
      ADD R7,2; avanca
      ADD R1,2;avaca
      MOV TEMP,[R1];Coloca em TEMP o que esta em R1
      MOV [R7],TEMP; Copia de antes
      ADD R7,2; avanca
      ADD R1,2;avaca
      MOV TEMP,[R1];Coloca em TEMP o que esta em R1
      MOV [R7],TEMP; Copia de antes
      ADD R7,2; avanca
      ADD R1,2;avaca

      MOV R6, [DISTANCIAPRECO] ; coloca a distancia ao preco para ir buscalo
```

```
      MOV R7, R5  ; volta ao inicio para o incremento estar certo
      ADD R7,R6   ;POE O APONTADOR R7 PARA O PRECO
      MOVB R8, [R7] ; Coloca a parte decimal do preco em R8
      ADD R7, 2; avanca para a casa das unidades
      MOV R11, [R7]     ;Colocar o valor da casa das unidades e o "ponto" para
R11
      MOV TEMP, [MASCARADPSVIRGULA]; Colocar a mascara em TEMP
      AND R11, TEMP; Fazer colocar os ultimo bit a 0 Pois e o "ponto"
      SHR R11,8; Mover 8 bits para a direita para ficar os bits na casa unidade
      CMP R8, R10;verificar se o valor recebito em R8 era 0
      JLE R8NaoASCII ; salta para o SomaUnidades se o falor recebido for 0
      SUB R8, R10 ; passa o valor R8 de ASCII para unidade
      MOV TEMP, [NUMERO10]; Coloca o numero 10 em
      MUL R8, TEMP; Multiplica R8 por 10
      JV ERROOVERFLOW; Caso haja overflow entao da erro
      JMP SomaUnidades; Salta para o SomaUnidades
R8NaoASCII:
      MOV R8,0; Caso o R8 nao for um valor ASCII
SomaUnidades:
      SUB R11, R10; Transforma o R11 de ASCII para unidades
      ADD R8,R11; Colocar o Valor da soma no R8
      MUL R8,R3 ; Multiplica o peso pelo preco
      ADD R7, 2; Coloca o apontador R7 a apontar para as casas decimais
      MOV R11, [R7];Coloca Ambos os digitos em R11
      MOV R9, [MASCARADPSVIRGULA]; Coloca em R9, FF00
      AND R9, R11; Coloca em R9 o valor primeiro digito decimal
      SHR R9, 8; Coloca os bits na parte menos significativa
      SUB R9, R10; Tansforma de ASCII para unidades
      MOV TEMP, [MASCARAANTESVIRGULA]; Coloca em R10 00FF
      AND R11, TEMP; coloca o R11 SO com o segundo digito
      SUB R11, R10; Transforma o R11 de ASCII para unidades
      MOV TEMP, [NUMERO10]; Coloca o numero 10 em
      MUL R9, TEMP; Multiplica R9 por 10
      ADD R9,R11; Coloca em R9, o valor total das decimas
      MUL R9, R3; Multiplica a parte decimal pelo peso
      MOV R11, 0;Coloca o R11 a 0 para ser o valor adicionado caso as decimas
sejam superiores a 100
LoopDecimas:
      MOV R1, [NUMERO100]; Coloca em R1 o numero 100
      SUB R1,1 ; Para quando for 99
      CMP R9,R1 ;Verifica se o R9 e maior que 100
      JLE HexaToASCIIToMemory;salta para o proximo passo se for
      ADD R1,1;Volta a colocar antes de subtrair
      JV ERROOVERFLOW; Caso haja overflow entao da erro
      SUB R9, R1; Subtrai 100
      ADD R11, 1; Adiciona 1 para as unidades
      JMP LoopDecimas; Volta ao inicio do loop

HexaToASCIIToMemory:
      ADD R8, R11; Adiciona o valor obtido antes para R8
      MOV R3,0; carrega a casa das 1k
      MOV R4,0; Carrega a casa das 100
      MOV R5,0; Carrega a casa das 10
      MOV R9,0; Carrega a casa das 1
      MOV R10,0; Carrega a casa das 0.1
```

```
      MOV R11,0;Carrega a Casa das 0.01
loop10k:
      MOV TEMP, [NUMERO10K];Coloca 100 em TEMP
    SUB TEMP,1; Colocao  temp a 99 pois se deixarmos a 100 vai dar erros
      CMP R8, TEMP; Compara o R3 e o TEMP(Que e 100)
      JLE Buffer10k;Se for menor que 99 entao vai para as desena
      ADD TEMP,1 ;Caso nao for volta a por a 100
      SUB R8, TEMP; tira esses 100
      ADD R4,1; Coloca 1 na casa das 100k
      JMP loop10k;volta atras
Buffer10k:
      MOV TEMP, [NUMERO10]; Coloca o numero 10 em temp
    SUB TEMP,1; Colocao  temp a 99 pois se deixarmos a 100 vai dar erros
      CMP R4, TEMP; Compara o R3 e o TEMP(Que e 100)
      JLE loop1k;Se for menor que 99 entao vai para as desena
      ADD TEMP,1 ;Caso nao for volta a por a 100
      SUB R4, TEMP; tira esses 100
      ADD R3,1; Coloca 1 na casa das 100k
      JMP Buffer10k;volta atras
loop1k:
      MOV TEMP, [NUMERO1K];Coloca 100 em TEMP
    SUB TEMP,1; Colocao  temp a 99 pois se deixarmos a 100 vai dar erros
      CMP R8, TEMP; Compara o R3 e o TEMP(Que e 100)
      JLE loop100;Se for menor que 99 entao vai para as desena
      ADD TEMP,1 ;Caso nao for volta a por a 100
      SUB R8, TEMP; tira esses 100
      ADD R5,1; Coloca 1 na casa das 100k
      JMP loop1k;volta atras
loop100:
      MOV TEMP, [NUMERO100];Coloca 100 em TEMP
    SUB TEMP,1; Colocao  temp a 99 pois se deixarmos a 100 vai dar erros
      CMP R8, TEMP; Compara o R3 e o TEMP(Que e 100)
      JLE loop10;Se for menor que 99 entao vai para as desena
      ADD TEMP,1 ;Caso nao for volta a por a 100
      SUB R8, TEMP; tira esses 100
      ADD R9,1; Coloca 1 na casa das 100k
      JMP loop100;volta atras
loop10:
      MOV TEMP, [NUMERO10];Coloca 100 em TEMP
    SUB TEMP,1; Colocao  temp a 99 pois se deixarmos a 100 vai dar erros
      CMP R8, TEMP; Compara o R3 e o TEMP(Que e 100)
      JLE BufferColocaASCII;Se for menor que 99 entao vai para as desena
      ADD TEMP,1 ;Caso nao for volta a por a 100
      SUB R8, TEMP; tira esses 100
      ADD R10,1; Coloca 1 na casa das 100k
      JMP loop10;volta atras

BufferColocaASCII:
      CMP R8, 4;Caso for 4 ou menos entao nao faz nada
      JLE ColocaASCII; Salta para a frente
      ADD R10,1 ;caso for para arredondar entao mete em R10
      MOV TEMP,[NUMERO10];Coloca o numero 10 em TEMP
      SUB TEMP,1; Coloca em 9 pois eu quero verificar se e 9 ou menos
      CMP R10, TEMP;Se for menor ou igual a 9 continua com o programa
      JLE ColocaASCII; salta para continuar
```

```
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB R10,TEMP;tira 10 do R10 para nao dar erros de display
      ADD R9,1; adiciona 1 no R9 (carry)
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB TEMP,1;Quero comparar se for 9
      CMP R9, TEMP; se for 9 ou menos continua
      JLE ColocaASCII; continua o programa
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB R9, TEMP;tira 10 do R9 para nao dar erros de display
      ADD R5,1; Coloca 1 no R5(Carry)
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB TEMP, 1;Para comparar com 9
      CMP R5, TEMP; Verifica se e 9 ou menos
      JLE ColocaASCII;continua o programa
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB R5,TEMP; Tira 10 do R5, para daar carry
      ADD R4,1; Adiciona 1 no R4 (carry)
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB TEMP, 1; Para Comparar com 9 e nao 10
      CMP R4, TEMP; verifica se e 9 ou menos
      JLE ColocaASCII; se for entao continua
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB R4, TEMP; Tira 10 de R4
      ADD R3,1; Adiciona 1 no R3, (carry)
      MOV TEMP,[NUMERO10];Volta a colocar 10 no TEMP
      SUB TEMP, 1; Para comparar com 9
      CMP R3, TEMP; verifica se e 9 ou menos
      JLE ColocaASCII; se for continua
      JMP ERROOVERFLOW; caso seja maior entao da overflow
ColocaASCII:
      MOV R11, R8; coloca o resto em R11(nao utilzado depois tirar)
      MOV R7, R2; Volta ao produto escolhido
      MOV R6, [DISTANCIATOTAL]; Mete a distancia para o total  em R6
      ADD R7, R6; coloca O apontador (R7) para o total
      MOV TEMP, [NUMERO0ASCII]; Coloca 30H em TEMP para transformar os numeros
em ASCII
      ADD R3,TEMP; Coloca o numero em ASCII
      ADD R4,TEMP; Coloca o numero em ASCII
      ADD R5,TEMP; Coloca o numero em ASCII
      ADD R9,TEMP; Coloca o numero em ASCII
      ADD R10,TEMP; Coloca o numero em ASCII
      ADD R11,TEMP; Coloca o numero em ASCII
      MOV [R7], R3; Coloca o valor De r3 em R7
      ADD R7,2; Avanca 2 casas
      SHL R4,8; Coloca o valor 8 casas para a esquerda(Casa das centenas)
      ADD R4,R5; Coloca o valor de R4 na segunda parte(casa das desenas)
      MOV [R7],R4; Coloca isso na memoria
      ADD R7, 2; Avanca 2 casas
      MOV TEMP, [PONTOASCII]; Coloca o pornto em ascii no TEMP
      SHL TEMP, 8 ;Coloca o valor de R9 2 casas para a esquerda
      ADD R9,TEMP; Coloca o POnto depois das unidades
      MOV [R7], R9; Coloca isso na memoria
      ADD R7,2 ;Avanca 2 casas na memoria
      SHL R10,8 ;Coloca o valor 8 casas a esquerda ( decimas)
      MOV TEMP, [LETRAEASCII]; coloca a letra E
```

```
	ADD R10,TEMP;Coloca o valor das centsimas
	MOV [R7],R10; Coloca isso na memoria
	POP TEMP
	POP R11
	POP R10
	POP R9
	POP R8
	POP R7
	POP R6
	POP R2
	RET
BufferColoca0ASCII:



BufferDesligaRegistos:
	POP R11
	POP R10
	POP R9
	POP R8
	POP R7
	POP R6
	POP R5
	POP R4
	POP R3
	POP R2
	POP R1
	POP R0
	MOV R1,1;e para desligar
	RET
BufferVoltaRegistos:
	POP R11
	POP R10
	POP R9
	POP R8
	POP R7
	POP R6
	POP R5
	POP R4
	POP R3
	POP R2
	POP R1
	POP R0
	MOV R1,0;nao e para desligar
	RET
BufferAcabaMostraProdutos:
	MOV R0, CANCEL; Carrega o botao de Cancel em R0
	MOVB R1, [R0]; Carrega o valor do botao cancel
	CMP R1,1 ;verifica se esta pressionao
	JEQ BufferVoltaRegistos;
	MOV R0, OK; coloca botao de ok no R0
	MOVB R1,[R0]; coloca o valor do botao ok no R1
	CMP R1, 1;verifica se esta a pressionado
	JLE BufferVoltaRegistos; se estiver a 0 volta atras
```

```
      JMP BufferAcabaMostraProdutos; enquanto nao ha opcoes continua no loop
BufferBuffferDesliga:
      JMP BufferDesligaRegistos; para conseguir voltar atras
MostraRegistos:
      PUSH R0
      PUSH R1
      PUSH R2
      PUSH R3
      PUSH R4
      PUSH R5
      PUSH R6
      PUSH R7
      PUSH R8
      PUSH R9
      PUSH R10
      PUSH R11
      MOV R0, INICIOPRODUTOS; Coloca o inicio do produto para mostrar
      MOV R1, [INCERMENTOPRODUTOS]; Coloca no R1 a distancia entre produtos
      MOV R2, 0 ; Coloca no R2 onde o contador de espacos de memoria
      MOV R3, [NUMEROPRODUTOS]; quantidade de produtos mais 1
      MOV R4, 0; contador dos produtos
      MOV R5, Display;coloca o inicio do display no R5
      MOV R6, Display_end; coloca o fim do display no R5
      MOV R7,0; Variavel para ficar como temporaria
      MOV R11, [NUMERO0ASCII]; coloca 0 em R11
      SUB R11, 1; para comparar com o 0 tambem
      JMP loopMostraProdutos; vai para o loop
BufferMostraProdutos:
      MOV R2, 0; volta a colocar o R2 a 0
      MOV TEMP, [DISTANCIATOTAL]; Coloca a distancia ao total em TEMP
      SUB R0,TEMP; Vai para o titolo do produto
      ADD R0,R1; vai para o proximo produto
loopMostraProdutos:
      CMP R3,R4; verifica se ja acabou os produtos;
      JLE BufferAcabaMostraProdutos;
      ADD R4,1; Adiciona 1 ao contador
      MOV TEMP, [DISTANCIATOTAL]; Coloca a distancia ao total em TEMP
      ADD R0, TEMP; Vai para onde esta o total
      MOV R7, [R0]; Mete o valor de R0 em 0
      MOV TEMP, [MASCARAANTESVIRGULA]; COLOCA 00FF EM TEMP
      AND R7,TEMP; para obter so o segundo valor
      CMP R7,R11; Verifica se nao tem nada, se nao tiver entao nao ha registos
      JLE BufferMostraProdutos; volta a fazer o loop
      MOV TEMP, [DISTANCIATOTAL]; Coloca a distancia ao total em TEMP
      SUB R0,TEMP; Vai para o titolo do produto
BufferLoopDisplayMostra:
      MOV R10, [R0];coloca em R10 o que esta nos produtos
      MOV [R5], R10; Grava isso na memoria
      ADD R5,2; vai para a poroxima posicao de memoria
      ADD R0,2; vai para a poroxima posicao de memoria
      ADD R2,2; incremetna o contador
      CMP R5,R6; verifica se o ecran ja acabou
      JLE BufferLoopDisplayMostra; se nao vola atras
      CMP R3,R4;verifica se ja acabou o produto
      JLE EsperaOK; espera o ok para acabar
```

```
      MOV R5, Display; Volta a colocar o display no incio
      SUB R0, R2; Volta a colocar no inicio do produto
      ADD R0, R1; Vai para o proximo produto
      CALL LimpaPerifericos; Limpa o OK
Continua:
      MOV R8, OK; coloca OK em R8
      MOVB R9, [R8]; Coloca o valor do botao em R9
      CMP R9,1; verifica se esta a 1;
      JEQ loopMostraProdutos; vai para o  loop mostra
      MOV R8, CANCEL; Coloca o botao cancel em R8
      MOVB R9, [R8];Coloca o valor do botao em R9
      CMP R9, 1;Verifica se esta ativo
      JEQ BufferVoltaRegistos;Se estiver entao volta atras
      JMP Continua; esper pelo ok ou cancel para a proxima acao
EsperaOK:
      MOV R8, OK; coloca OK em R8
      MOVB R9, [R8]; Coloca o valor do botao em R9
      CMP R9,1; verifica se esta a 1;
      JEQ BufferVoltaRegistos; vai para o  loop mostra
      MOV R8, CANCEL; Coloca o botao cancel em R8
      MOVB R9, [R8];Coloca o valor do botao em R9
      CMP R9, 1;Verifica se esta ativo
      JEQ BufferVoltaRegistos;Se estiver entao volta atras
      MOV R8, ON_OFF;COlocao o botao em R8
      MOVB R9, [R8];COloca o valor do R9
      CMP R9,1; Verifica se esta ativo
      JEQ BufferBuffferDesliga; Se estiver entao desliga
      JMP EsperaOK; enquanto nao ha input volta ao loop
MostraPeso:
      PUSH R0
      PUSH R1
      PUSH R2
      PUSH R3
      PUSH R4
      PUSH R5
      PUSH R6
      PUSH R7

      MOV R0, PESO; coloca o endereco do peso em R0
      MOV R1, [R0]; coloca o peso em R1
      MOV R2, 0; Casa de 10K
      MOV R3, 0; Casa de 1K
      MOV R4, 0; Casa de 100
      MOV R5, 0; Casa de 10
      MOV R6, 0; casa de 1
loopPeso10k:
      MOV TEMP, [NUMERO10K]; Coloca 10000 em TEMP
      SUB TEMP, 1; para comparar com 9999
      CMP R1,TEMP;Verifica se o numero ja e menor que 10k
      JLE loopPeso1k; se o valor for menor passa para 1k
      ADD R2, 1; adiciona 1 na casa de 10k
      ADD TEMP, 1; volta a colocar 10k
      SUB R1, TEMP; tira 10k
      JMP loopPeso10k; volta atras
loopPeso1k:
```

```
      MOV TEMP, [NUMERO1K]; Coloca 1000 em TEMP
      SUB TEMP, 1; para comparar com 999
      CMP R1,TEMP;Verifica se o numero ja e menor que 1k
      JLE loopPeso100; se o valor for menor passa para 1k
      ADD R3, 1; adiciona 1 na casa de 1k
      ADD TEMP, 1; volta a colocar 1k
      SUB R1, TEMP; tira 1k
      JMP loopPeso1k; volta atras
loopPeso100:
      MOV TEMP, [NUMERO100]; Coloca 100 em TEMP
      SUB TEMP, 1; para comparar com 99
      CMP R1,TEMP;Verifica se o numero ja e menor que 100
      JLE loopPeso10; se o valor for menor passa para 100
      ADD R4, 1; adiciona 1 na casa de 100
      ADD TEMP, 1; volta a colocar 100
      SUB R1, TEMP; tira 100
      JMP loopPeso100; volta atras
loopPeso10:
      MOV TEMP, [NUMERO10]; Coloca 10 em TEMP
      SUB TEMP, 1; para comparar com 9
      CMP R1,TEMP;Verifica se o numero ja e menor que 10
      JLE BufferAcabaLoopPeso; se o valor for menor passa para 10
      ADD R5, 1; adiciona 1 na casa de 10k
      ADD TEMP, 1; volta a colocar 10
      SUB R1, TEMP; tira 10
      JMP loopPeso10; volta atras
BufferAcabaLoopPeso:
      CMP R5, 4; verifica se e perciso arredondar
      JLE AcabaLoopPeso;se nao entao continua
      ADD R4,1; se for perciso entao adiciona 1 na proxima casa
      MOV TEMP, [NUMERO10]; coloca o TEMP a 10
      SUB TEMP,1; Tira 1 para comparar com 9
      CMP R4, TEMP; verifica se e 9 ou menos
      JLE AcabaLoopPeso; se for entao continua
      MOV TEMP, [NUMERO10]; Volta a colocar o temp a 10
      SUB R4, TEMP; tira 10 do R4
      ADD R3,1; e coloca no R3 1 (carry)
      MOV TEMP, [NUMERO10]; volta a colocar o TEMP a 10
      SUB TEMP,1; tira 1 para comparar com 9
      CMP R3, TEMP; se for menor que 9
      JLE AcabaLoopPeso; entao continua
      MOV TEMP, [NUMERO10];volta a colocar o TEMP a 10
      SUB R3, TEMP; tira 10 do R3
      ADD R2,1; coloca em R2
      CMP R2,2; verifica se e 2 ou menos
      JLE AcabaLoopPeso; se for continua
      JMP ERROOVERFLOW; se nao isso quer dizer que ja ultrapassou o limite de
peso
AcabaLoopPeso:
      MOV R6, R1; coloca o restante em R6
      MOV TEMP, [NUMERO0ASCII]; Coloca em TEMP o 0 em ascii
      ADD R2, TEMP; Coloca o numero em ASCII
      ADD R3, TEMP; Coloca o numero em ASCII
      ADD R4, TEMP; Coloca o numero em ASCII
      ADD R5, TEMP; Coloca o numero em ASCII
```

```
      ADD R6, TEMP; Coloca o numero em ASCII

      SHL R2,8; anda 8 casas para a esquerda
      ADD R2, R3; coloca o outro numero
      MOV TEMP, [PONTOASCII]; coloca o Ponto
      SHL TEMP,8 ; Coloca o potno a esquerda
      ADD TEMP, R4; coloca o outro numero
      MOV R4, TEMP; guarda esse valor em R4
      SHL R5, 8; anda 8 casas para a esquerda
      ADD R5, R6; coloca o outro numero

      MOV TEMP, [PESODISPLAY]; coloca onde a distancia do display no TEMP
      MOV R7, Display; Coloca o inicio do display em R7
      ADD R7, TEMP; vai para onde e para escrever
      MOV [R7], R2; coloca os numeros na memoria
      ADD R7, 2;avanca 2 casas na memoria
      MOV [R7], R4; coloca os numeros na memoria

      POP R7
      POP R6
      POP R5
      POP R4
      POP R3
      POP R2
      POP R1
      POP R0
      RET

Espera:
      PUSH R0
      PUSH R1
loopEspera:
      MOV R0, OK;Coloca no R0 o endereco de ol
      MOVB R1, [R0];Coloca o valor OK Em R1
      CMP R1, 0; Verifica se ainda esta a 0
      JEQ loopEspera; Acaba o loop
Acabaloop:
      POP R1
      POP R0
      RET
```