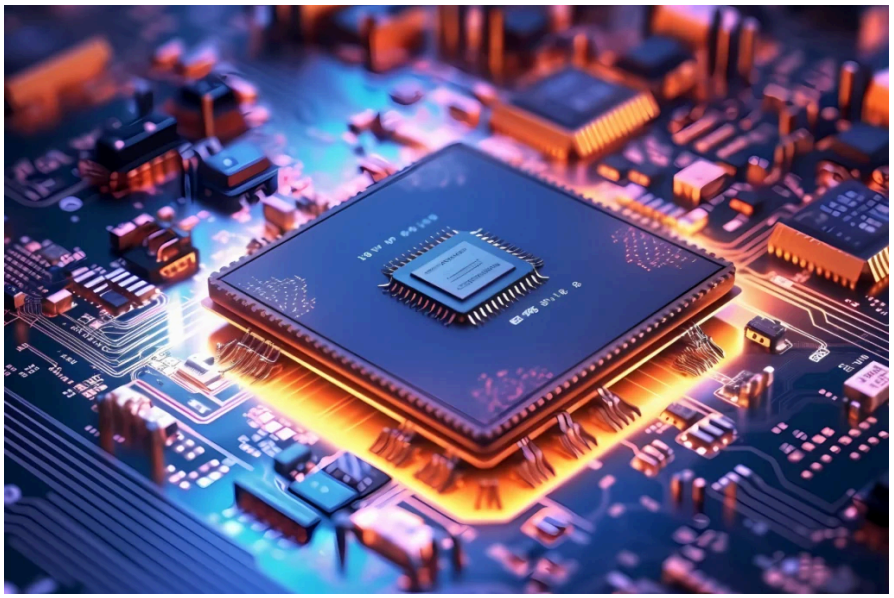
UNIVERSIDADE da MADEIRA

FCEE
04/2025

RELATÓRIO

Arquitetura de Computadores

Projeto 2



Discentes

Diogo Franco (2019112)
Cláudio Fernandes (2186622)

Docentes

José Gonçalves
Élvio Jesus
Daniel Parada
João Dionísio Simões Barros

Índice

Introdução

Neste segundo trabalho prático da disciplina de Arquitetura de Computadores foi nos pedido para implementar um programa que simule o funcionamento de uma balança eletrónica comercial. A linguagem de programação usada para a elaboração deste trabalho foi *Assembly*, sendo o processador o PEPE-16 que executa instruções/dados guardados numa memória principal (*Ram*).
A aplicação usada para a simulação foi a "*Arquitecture Simulator*", que basicamente serve para executar variados programas em diferentes processadores.

Temos alguns aspetos de funcionamento e características operacionais da balança a ter em conta: o *display* (periférico de saída) com dimensões 7x16 (7 linhas de 16 carateres); a balança com precisão de 0,1 kg, ou seja só conseguir pesar valores múltiplos de 100 gramas; no registo de produtos(opção balança) ter que aparecer obrigatoriamente o nome do produto, o preço por kg, o peso do mesmo e o preço total.

A informação de todos os produtos está alocada num intervalo de endereços de memória (tabela que constam as informações e propriedades dos produtos) em que o utilizador, quando precisar de certo produto, será desse espaço reservado que serão extraídas as informações do artigo selecionado.

A balança tem um menu principal constituído por 3 opções: a primeira (modo balança) é onde a complexidade e a criatividade estão mais presentes sendo assim a parte mais fulcral do programa.
A segunda opção (modo registos) serve para visualizar os registos que foram gravados no modo balança, tendo cada um a informação do nome do produto, o peso e o preço total.
A terceira opção (modo limpar registos) serve para limpar/apagar todos os registos guardados na balança.

Para a interação utilizador-balança foi nos proposto a utilização de 6 periféricos de entrada (Botão *On/Off*, Entrada *Sel_nr_menu*, Botão *Ok*, Botão *Change*, Botão *Cancel* e Peso), que darão uma maior dinâmica e personalização ao programa.
Para todas estas funcionalidades do programa, foram guardados espaços na memória, devidamente alocados e de forma estratégica para que não ocorra qualquer erro/contrariedade na execução do programa, como por exemplo as instruções ocuparem espaços de memória reservados à alocação de registos.

Objetivos

O principal objetivo deste trabalho prático foi o desenvolvimento de um programa funcional, elaborado em linguagem *Assembly*, direcionado para o processador PEPE-16, que simulasse o funcionamento de uma balança eletrónica comercial. Pretendeu-se aplicar os conhecimentos adquiridos no trabalho anterior da disciplina sobre a organização interna do processador, bem como o acesso à memória, sendo que neste segundo trabalho foi necessário incidir sobre a gestão da memória, definindo o mapa de endereçamento, para uma organizada distribuição das gamas de endereços associados tanto a periféricos como para a utilização da memória, onde nesta última era esperado uma separação entre a gama para as instruções e a gama para os dados, de forma intercalada para que não existissem sobreposições, e possibilitasse futuras expansões de código e/ou dados.

Considerou-se também como objetivo o controlo do fluxo do programa e a gestão de dados através do uso da *Stack Pointer*, aquando das várias chamadas a rotinas com *CALL/RET* que vão sendo realizadas no decorrer do programa, e da necessidade de preservar valores nos diversos cálculos que vão sendo realizados.

Os objetivos concretos para a realização deste trabalho foram os seguintes:

- Desenvolver uma interface principal com três modos de operação(balança, visualizar registos e limpar registos);
- Garantir a correta leitura das escolhas do utilizador através de botões e do sensor(*switches* - 16 bits) de peso;
- Implementar os cálculos para o preço total com tratamento de *overflow*;
- Apresentar as informações no *display* 7x16 de forma clara e personalizada;
- Armazenar os dados relevantes durante a simulação, mais propriamente aquando dos registos realizados no modo balança.

Desenvolvimento

Organização da Memória

Numa fase inicial do planeamento do programa, como neste trabalho lida-se com uma linguagem tão próxima do *hardware*(*Assembly*), existe a necessidade de realizar o mapeamento dos endereços na memória, começando por atribuir os endereços aos respectivos dispositivos, que neste caso ficaram definidos como se encontra na tabela 1 no Anexo A. A comunicação com os dispositivos é assim feita ao nível interno através da *RAM.*

Estando atribuídas as zonas reservadas para as memórias, procedeu-se à realização do mapa de utilização das mesmas, que se encontra especificado na tabela 2 no Anexo A.

No código utiliza-se a diretiva "*PLACE*" para definir onde começa cada bloco de instruções ou de dados, sendo que para os menus e para a tabela dos produtos optou-se por inseri-los diretamente nos endereços, já pré-definidos, através de um ficheiro "*.dat*" que deverá ser carregado na memória. Evitou-se assim o sobrecarregamento do ficheiro *assembly* com o código do programa.

Para a definição da *Stack Pointer* utilizou-se a diretiva mencionada para definir onde termina a pilha, e posteriormente é usada a *keyword* "*Stack*" para definir em qual endereço começa, ficando assim à partida bem definido o seu espaço de ação. Caso a *Stack* saia fora desta gama de endereços, o programa pára e é emitido um aviso alusivo, sendo esta uma das vantagens de atribuir um intervalo de memória para a *Stack Pointer*.

Outras zonas da memória encontram-se reservadas para a base de dados, de onde se obtém por exemplo o acesso à tabela de produtos, cuja estrutura se encontra exemplificada na tabela 3 no Anexo A, os menus do programa para o display, bem como mensagens informativas. Também endereços para a manipulação de dados como é o caso da zona de registos e de variáveis globais.

Estrutura do Programa

- **Menu Principal**
  O programa começa no menu principal onde é esperado que se introduza um valor no periférico de entrada *SEL_NR_MENU*, compreendido entre 1 e 3, que possibilitará avançar para um dos três modos de ação desta balança eletrónica, através de chamadas às rotinas correspondentes.
- **Modo Balança**
  Este modo dá acesso à balança propriamente dita, onde é efetuada uma leitura constante do peso introduzido em centenas de gramas(Hectogramas), pois como esta balança possui uma precisão de 0,1, torna-se mais fácil realizar os cálculos neste processador com esta ordem de grandeza, sendo que não trabalha com vírgula flutuante.

  Para a seleção dos produtos, a mesma é conseguida utilizando o periférico *SEL_NR_MENU*, onde deverá ser introduzido o código de um produto. Caso o utilizador deseje ter acesso à lista de produtos para consultar os respectivos códigos basta pressionar o botão *CHANGE*.
  A leitura do produto dá acesso ao cálculo do total(em cêntimos) resultante da multiplicação entre o peso do produto(em hectogramas) e o preço por quilograma (em cêntimos). O valor resultante desta operação é armazenado em cêntimos para maior precisão. Tendo em conta que o PEPE-16 é um processador de 16 *bits* com registos *signed*, os valores que podem ser representados diretamente vão de -32 768 (8000H) até 32 767 (7FFFH). Assim, qualquer resultado de multiplicação que ultrapasse esse intervalo causa um *overflow*, sendo necessário detetar este evento.

Por exemplo, para um preço de 19 EUR/Kg e um peso de 1,9 Kg (introduzido como 19 hectogramas), o cálculo seria:

$$1900 * 19 = 36100 \text{ cênt/hg}$$

o que já ultrapassa o limite suportado para valores inteiros com sinal neste sistema. Assim, para resultados superiores a 32,76 euros, o programa apresenta uma mensagem de *overflow* no display e impede que o registo seja efetuado, garantindo a integridade dos dados armazenados.

Quanto ao arredondamento que é feito no cálculo do total, este é executado ao nível dos cêntimos, quando na casa mais à direita(que não surge no display) ocorre um número igual ou superior a cinco.

É também neste modo que existe a possibilidade de efetuar um registo através da confirmação utilizando o botão *OK*, estando o número de registos limitado ao espaço de endereçamento associado.

- **Modo Registos**
  Permite efetuar a leitura dos registos guardados na memória, sendo que a sua apresentação no display é feita de forma sequencial, apresentando no máximo dois registos de cada vez, devido à limitação das dimensões do *display,* tendo o botão *CHANGE* a função de carregar mais registos para a visualização. Cada registo contém a informação referente ao nome do produto, o peso e o valor total no momento do registo.
- **Modo Limpar Registos**
  Neste modo, existe a possibilidade de eliminar o conteúdo da zona de memória reservada aos registos, após realizada a confirmação que surge no *display*.

Em todos os modos/menus existe a possibilidade de retroceder(botão *CANCEL*) e de desligar a balança(botão *ON/OFF*).

Uso da Stack e das Rotinas

Sempre que possível utilizou-se os registos disponíveis para satisfazer as necessidades de armazenamento do programa. As rotinas que não chamam outras rotinas, e que não alteram valores necessários guardados nos registos, são chamadas com *CALLF/RETF*, dando uso ao registo R11 que guarda o endereço de retorno, e evitando recorrer ao uso da memória. Na impossibilidade deste procedimento recorreu-se à *Stack* e ao *CALL/RET* para segmentar o programa. A salvaguarda do conteúdo dos registos é obtida com *PUSH* e *POP*. Desta forma criou-se rotinas independentes que permitem organizar o código, tornando-o mais legível e acessível para as constantes retificações/alterações que surgem no desenvolvimento do trabalho.

Discussão de Resultados

Durante os testes, o programa comportou-se da forma esperada na generalidade das operações. A utilização dos botões permitiu uma navegação intuitiva entre os modos, e o display revelou-se suficiente para mostrar a informação relevante ao utilizador.

Um dos principais desafios encontrados foi o tratamento do overflow nas multiplicações, já que o processador PEPE-16 lida com registos de 16 bits com sinal, e não opera com vírgula flutuante, o que acabou por limitar as possibilidades de obter um correto resultado no valor total, diminuindo consideravelmente o peso aceitável para produtos com um preço mais elevado.

O uso da Stack revelou-se essencial para a reutilização de código em diferentes rotinas. O uso de PUSH/POP garantiu a integridade dos registos.
O programa foi testado com múltiplas simulações, demonstrando robustez nas operações aritméticas, como no caso dos arredondamentos, bem como na gestão da memória, de registos e periféricos. A verificação de erros e o tratamento de entradas inválidas, como produtos não selecionados, foram tratados adequadamente.

Conclusão

Este trabalho prático de avaliação permitiu consolidar conhecimentos sobre a arquitetura de computadores, que foram sendo adquiridos nas aulas da disciplina, nomeadamente o funcionamento interno de um processador e a comunicação com a memória e periféricos, entendendo a importância e proximidade ao hardware que tem a linguagem *Assembly*, que apesar de trabalhosa comparando com linguagens de mais alto nível, permite um controlo mais pormenorizado dos componentes afetos aos programas realizados. Esta implementação em *Assembly* exigiu uma atenção rigorosa à alocação de memória, ao controlo de fluxo e à manipulação de registos, sendo uma boa aproximação prática à realidade dos sistemas embebidos.
Assim ficou-se com uma melhor noção de todo o trabalho que é realizado pelo processador para a execução de um programa relativamente simples como o que é aqui apresentado.

Bibliografia

[1] Apontamentos das aulas teóricas de Arquitetura de Computadores, docente João Dionísio Simões Barros.
[2] Apontamentos das aulas prático-laboratoriais de Arquitetura de Computadores.
[3] José Delgado e Carlos Ribeiro, “Arquitetura de Computadores”, FCA Editora de Informática, 2014.

Anexo A

Dispositivo	Gama de Endereços
Memória 1	0H – 50H
Botão – *ON_OFF*	1A0H
SEL_NR_MENU	1B0H
Botão – *OK*	1C0H
Botão – *CHANGE*	1D0H
Botão – *CANCEL*	1E0H
PESO	1F0H – 1F1H
***Display* – 7 linhas x 16 colunas(*Bytes*)**	200H – 26FH
Memória 2	1000H – FFFFH

Tabela 1 - Mapa de endereçamento

Memória	ENDEREÇOS	DADOS/INSTRUÇÕES
Memória 1	0H – 50H	Instruções
Memória 2	1000h – 1800h	Instruções
	2000h – 2400h	Dados – *Stack*
	2500h – 2800h	Dados – Histórico dos registos
	3000h – 4000h	Dados – Menus
	5000h – 6000h	Dados – Tabela de produtos
	7000H – 7500H	Outros Dados(por ex.: variáveis globais)

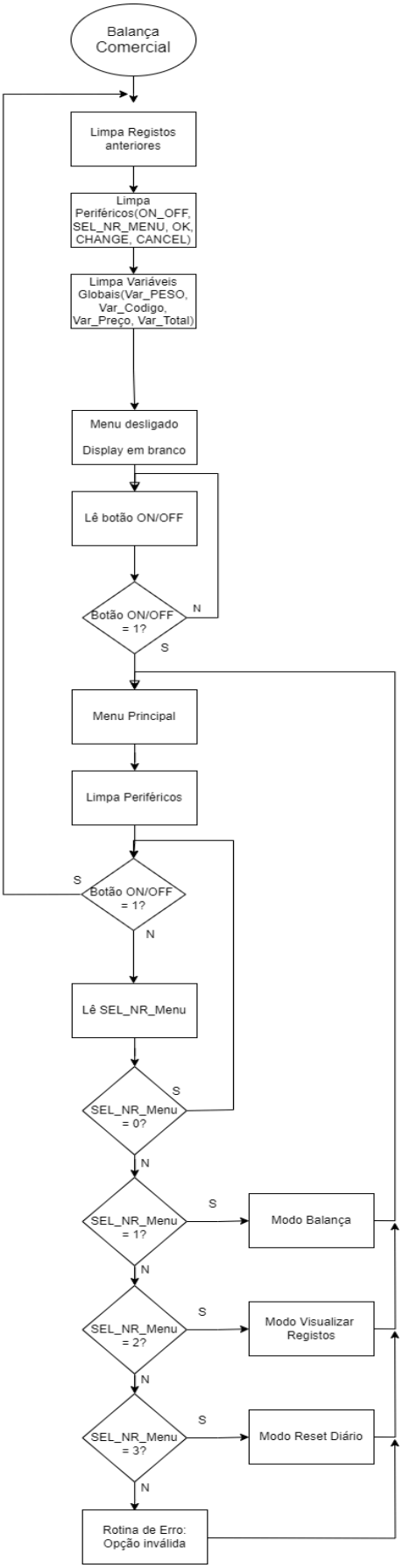
Tabela 2 - Mapa de utilização da memória

Índice	Endereço	Atributo
0	5000H	Código(2 Bytes)
	5001H	
	5002H	Preço(2 Bytes)
	5003H	
	5004H	Nome(10 Bytes)
	…	
	500DH	

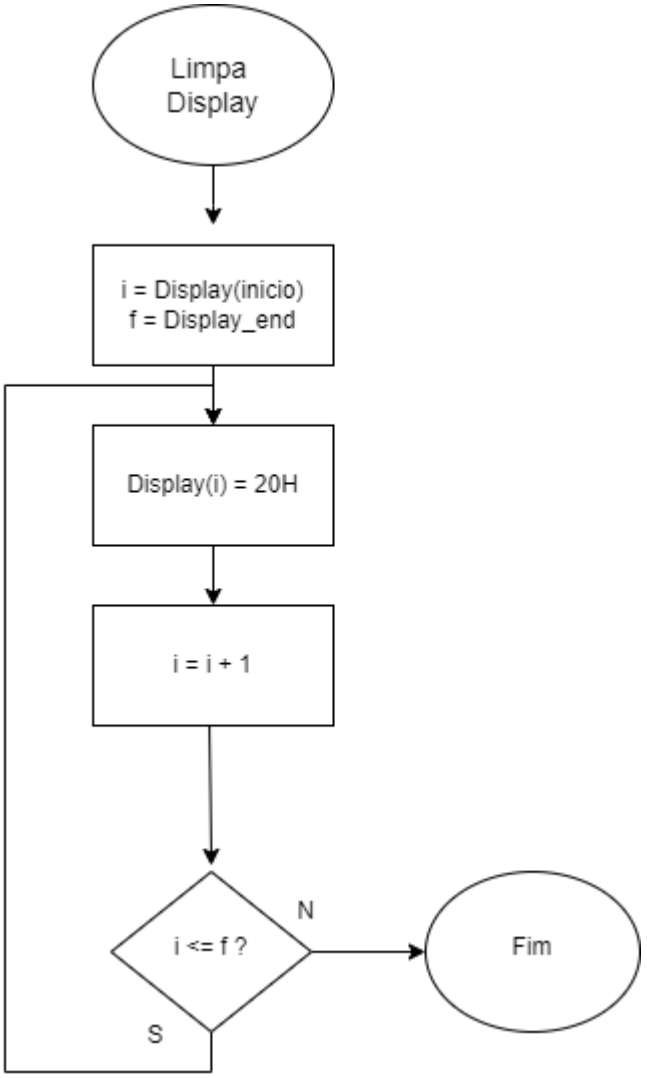
Tabela 3 - Estrutura da disposição da informação dos produtos na tabela(memória)

Fluxogramas

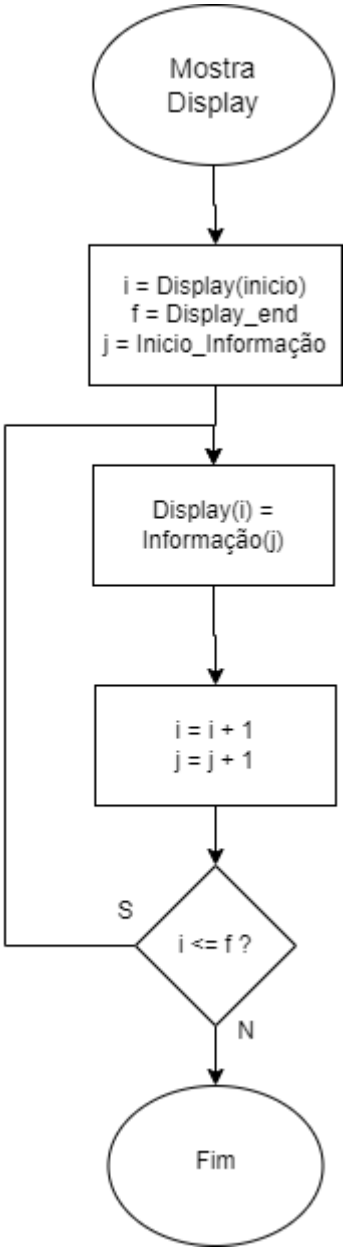
1.Programa principal



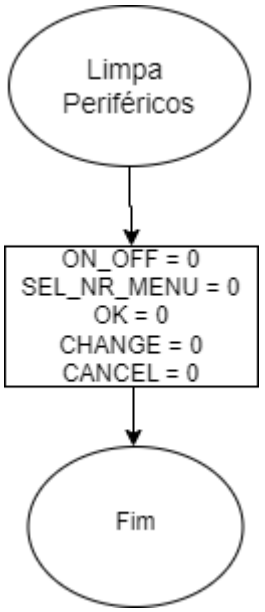
2.Limpa Display



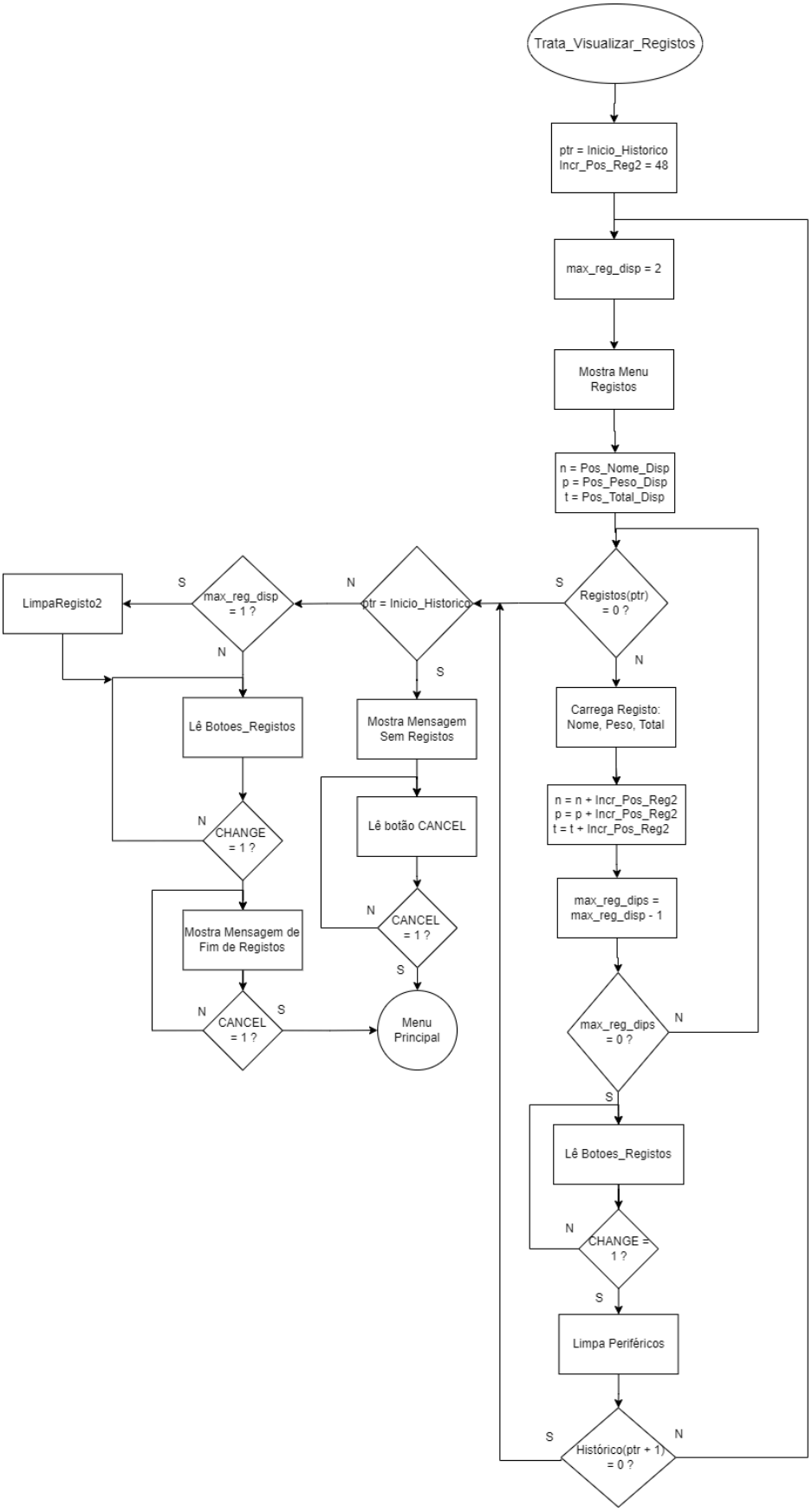
3.Mostra Display



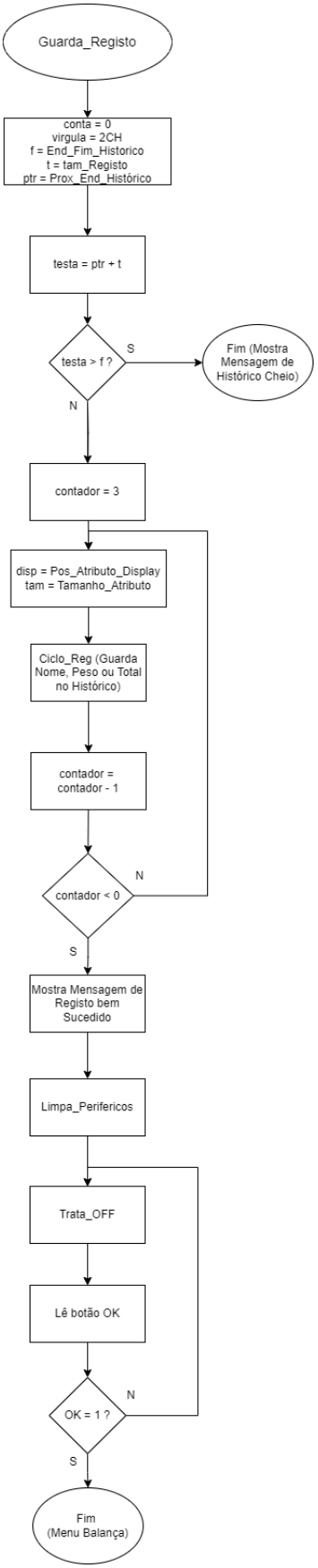
4.Limpa Periféricos



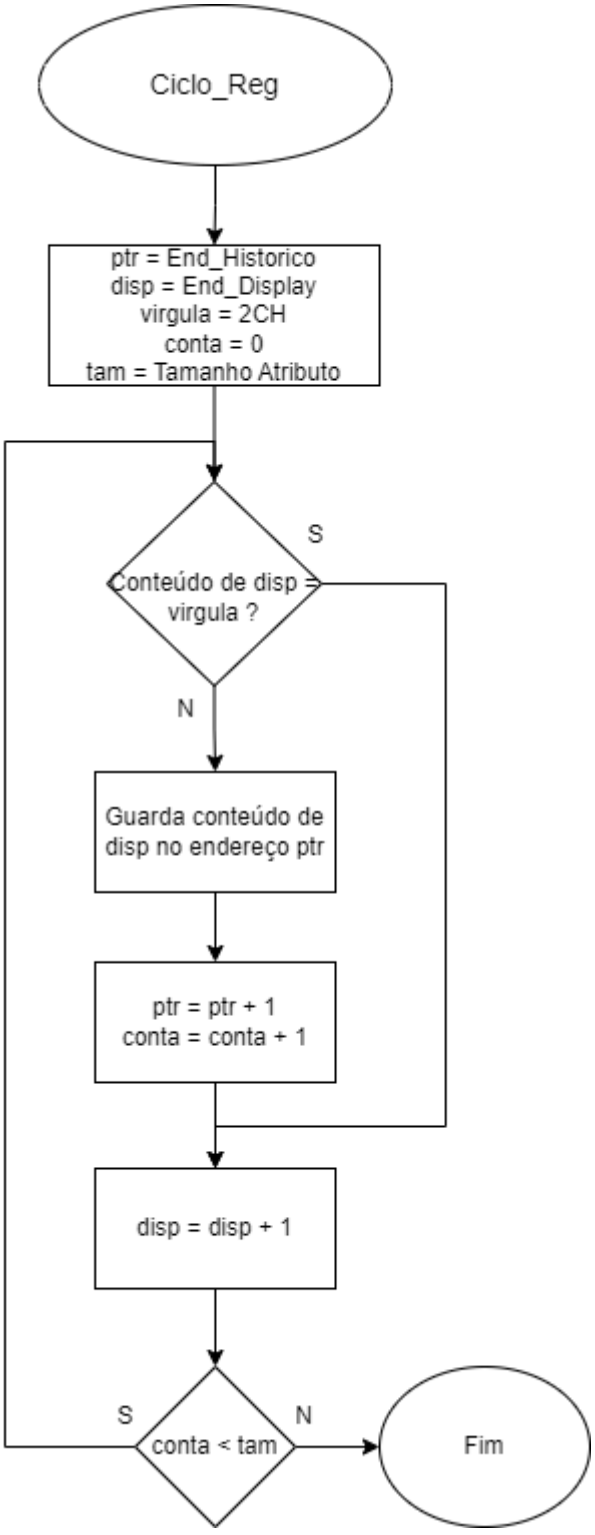
5.Trata Visualizar Registos



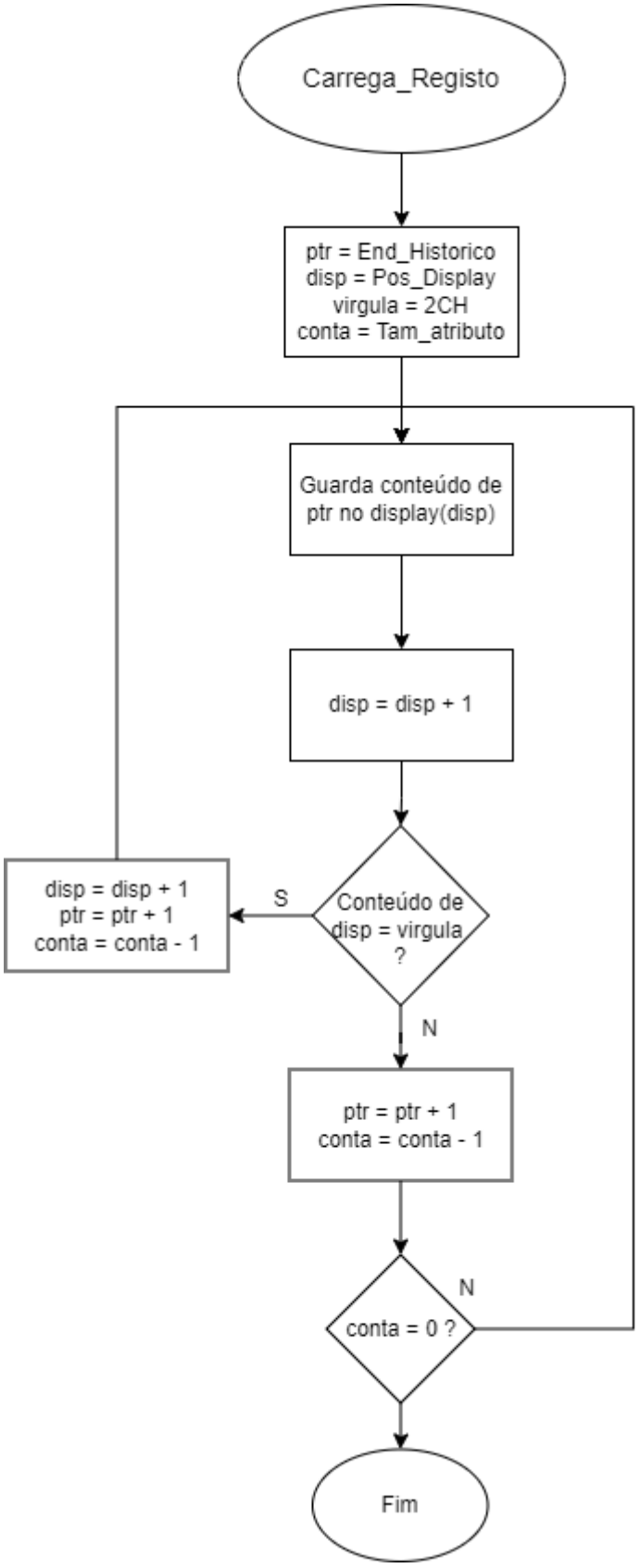
6. Guarda Registo



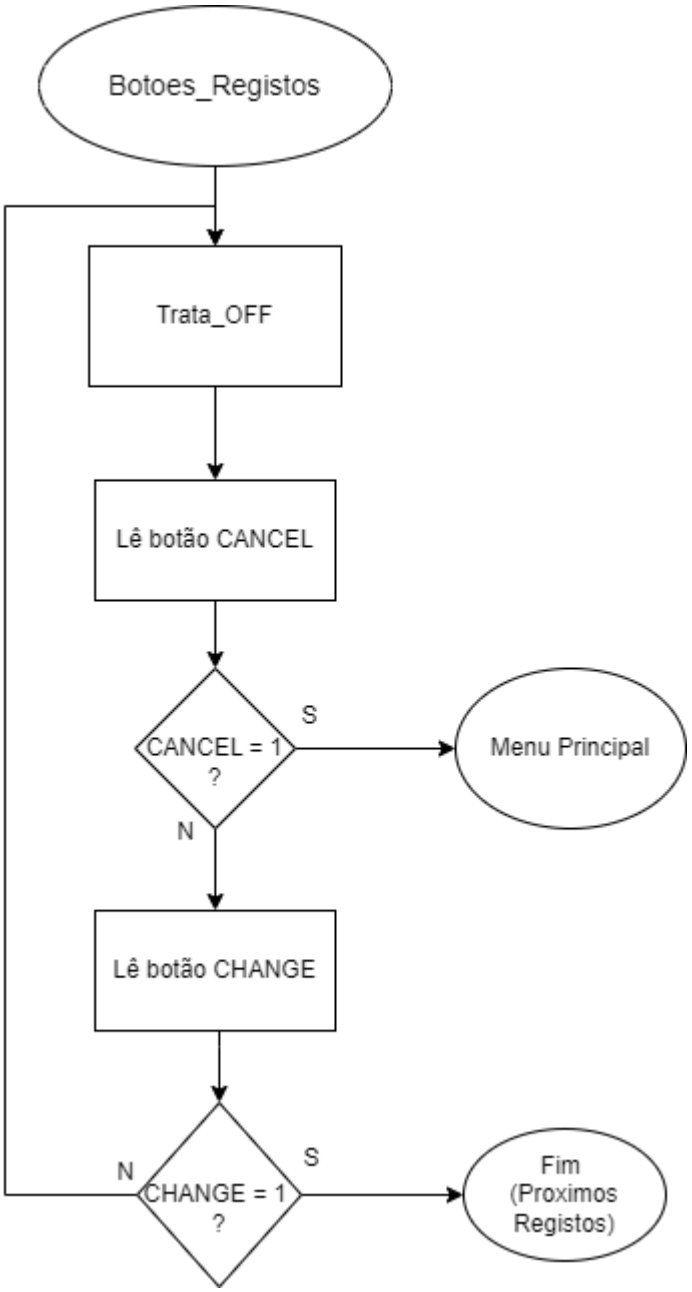
7. Ciclo Reg


Ciclo_Reg
ptr = End_Historico
disp = End_Display
virgula = 2CH
conta = 0
tam = Tamanho Atributo
Conteúdo de disp = virgula ?
S
N
Guarda conteúdo de disp no endereço ptr
ptr = ptr + 1
conta = conta + 1
disp = disp + 1
conta < tam
S
N
Fim

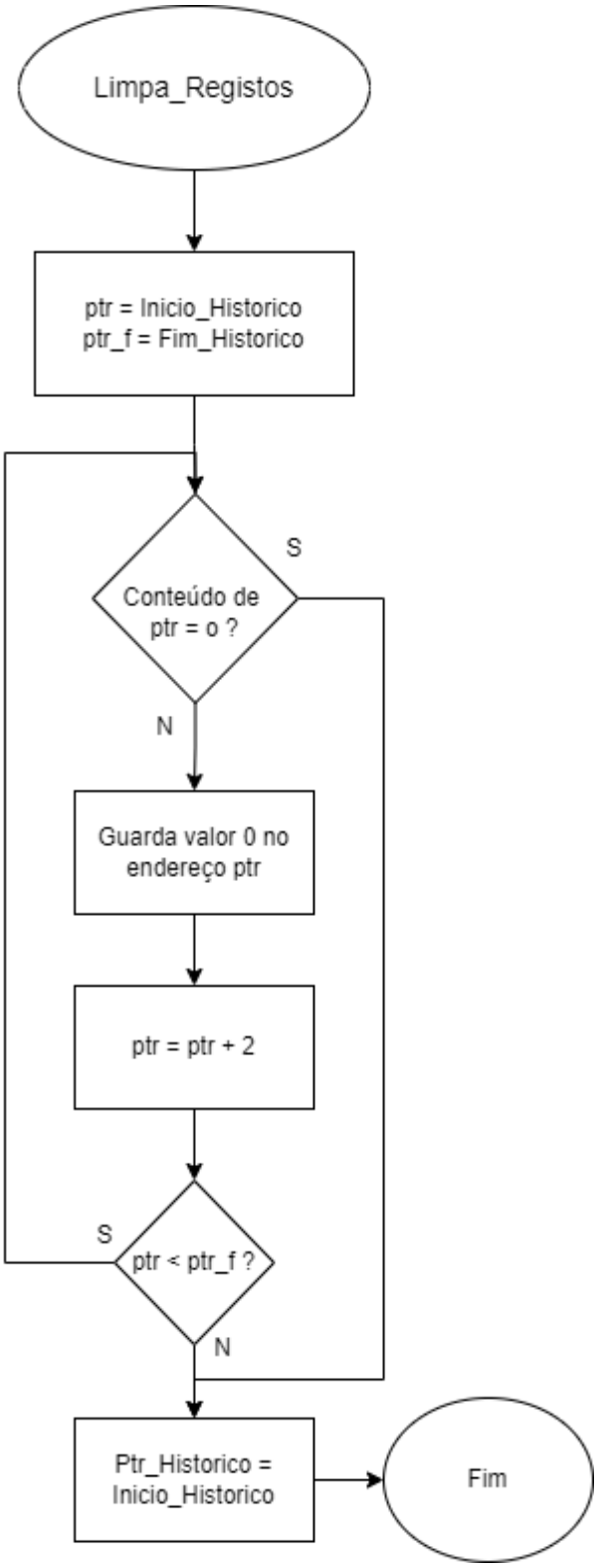
8.Carrega Registo



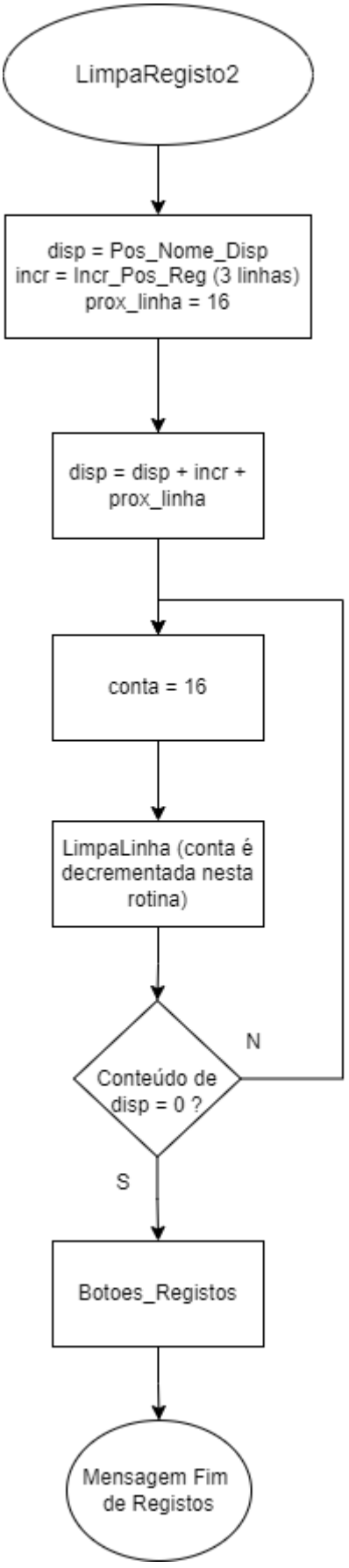
9. Botões Registo



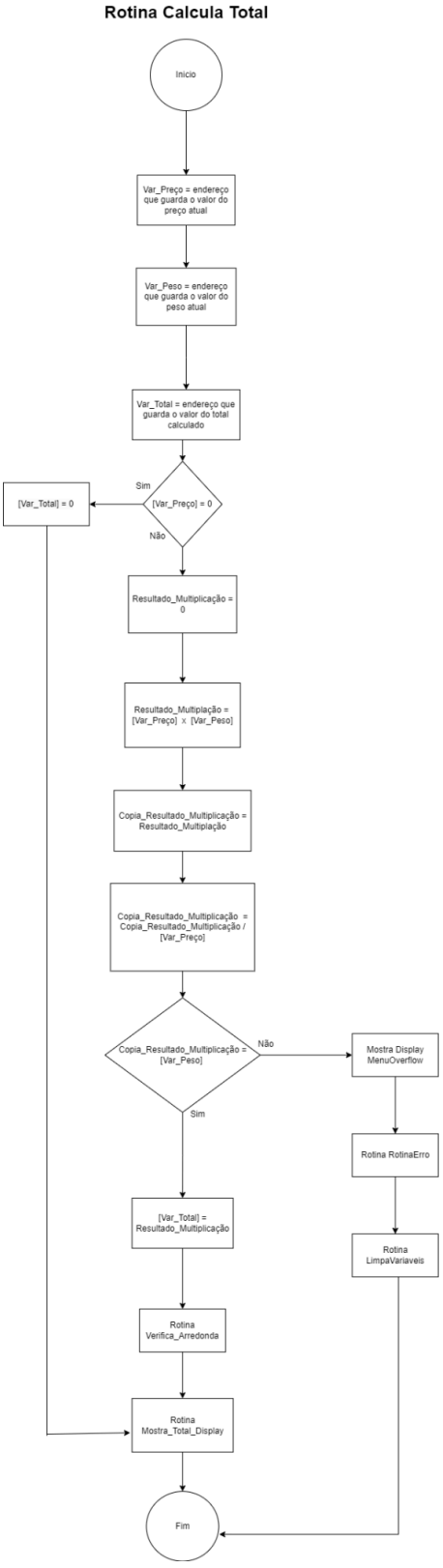
10. Limpa Registos



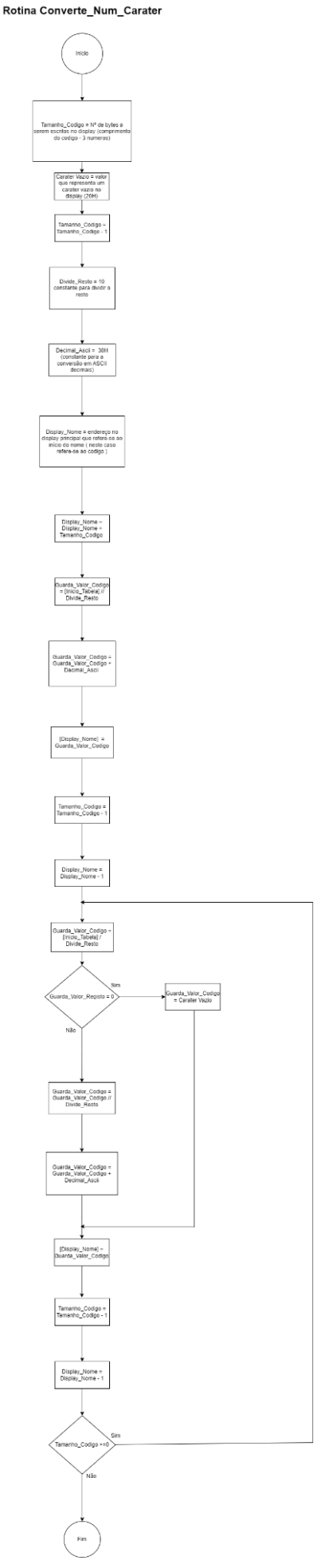
11. Limpa Registo2



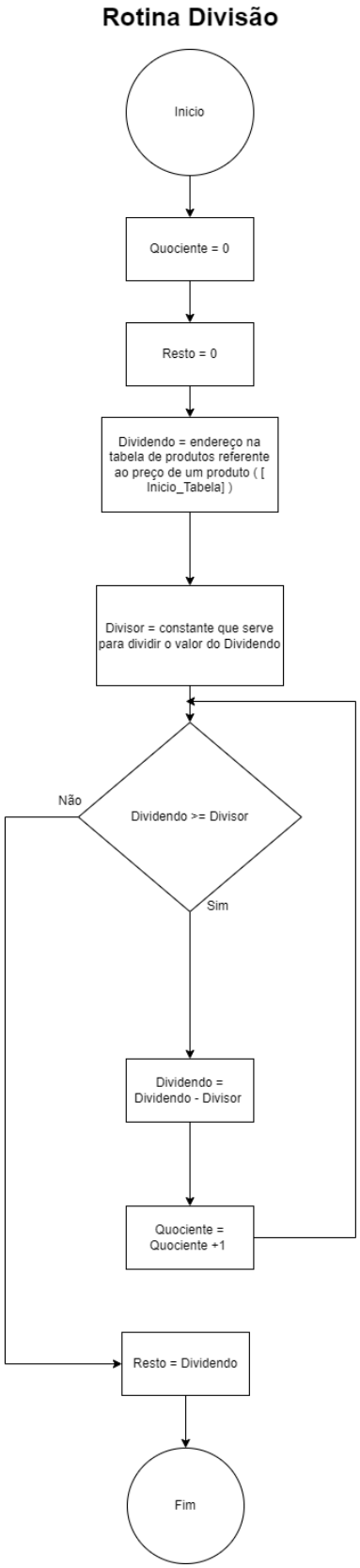
12.Calcula total



13. Converte num carater

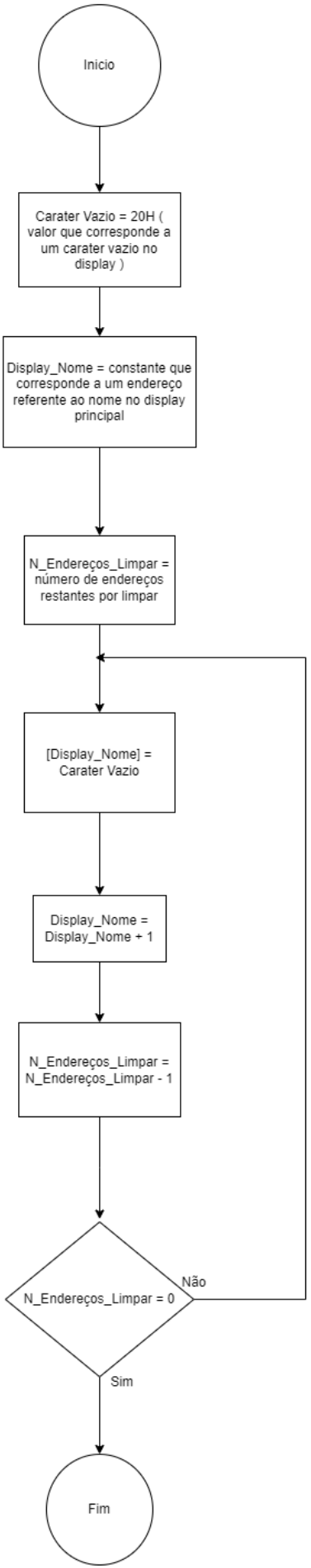


14. Divisão

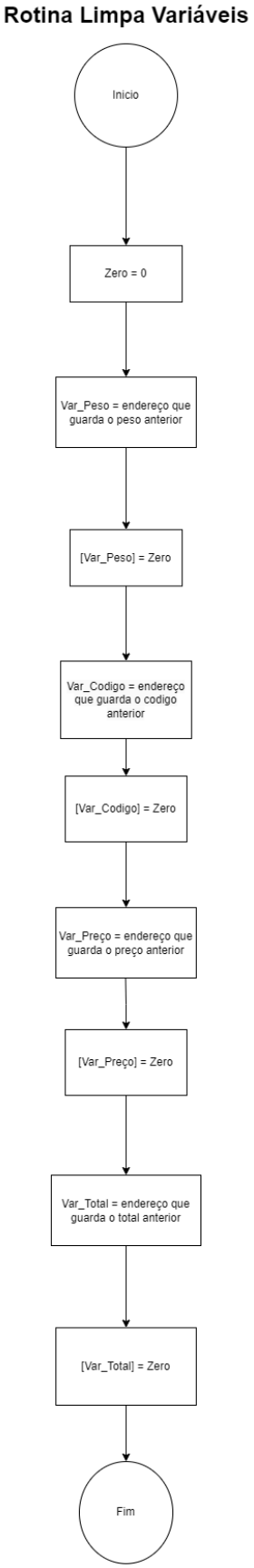

Rotina Divisão
Inicio
Quociente = 0
Resto = 0
Dividendo = endereço na tabela de produtos referente ao preço de um produto ([Inicio_Tabela])
Divisor = constante que serve para dividir o valor do Dividendo
Dividendo >= Divisor
Não
Sim
Dividendo = Dividendo - Divisor
Quociente = Quociente +1
Resto = Dividendo
Fim

15. Limpa Linha

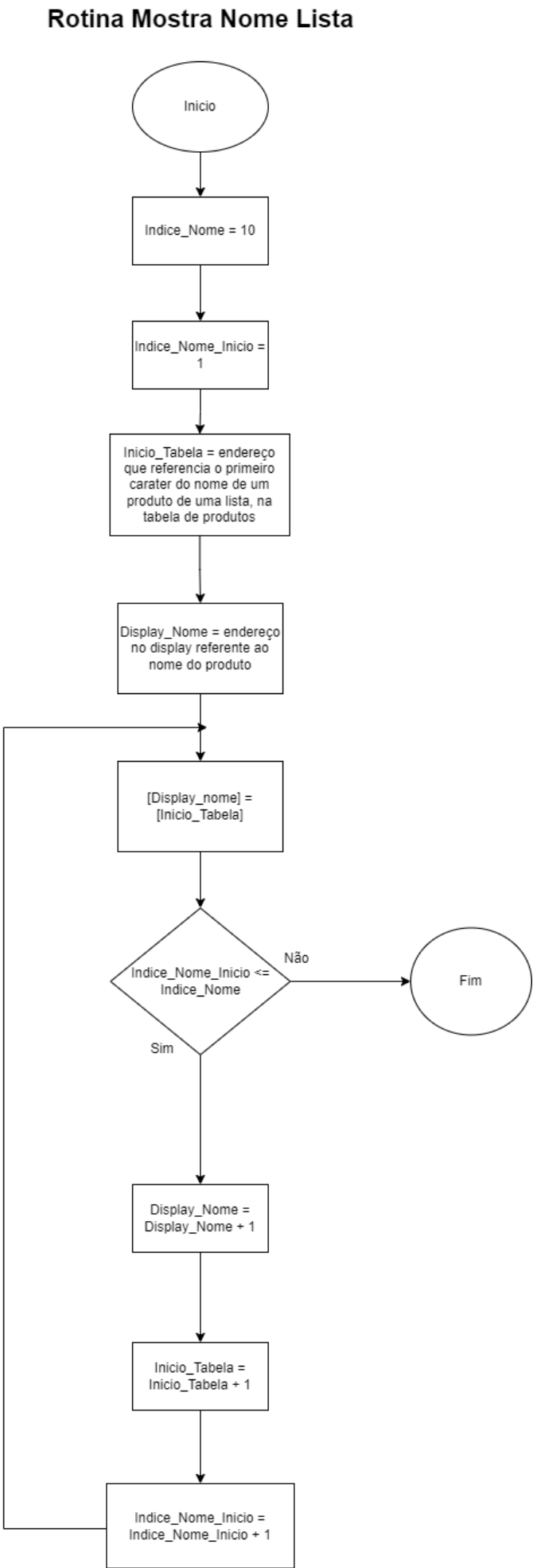
Rotina LimpaLinha



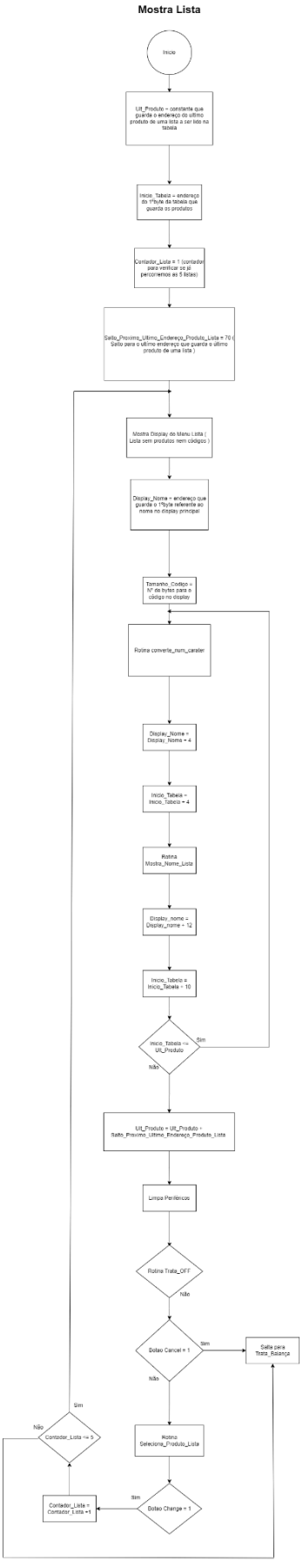
16. Limpa Variáveis



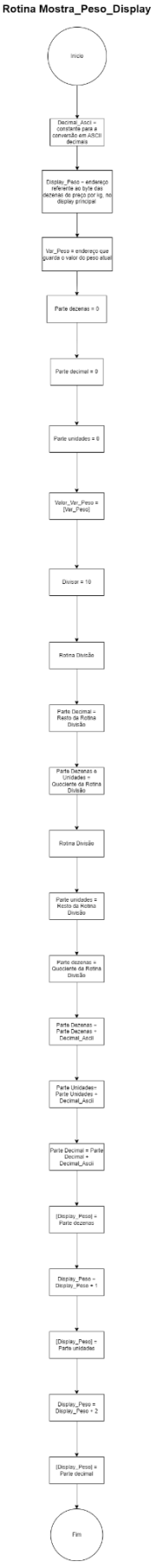
17. Mostra Nome Lista



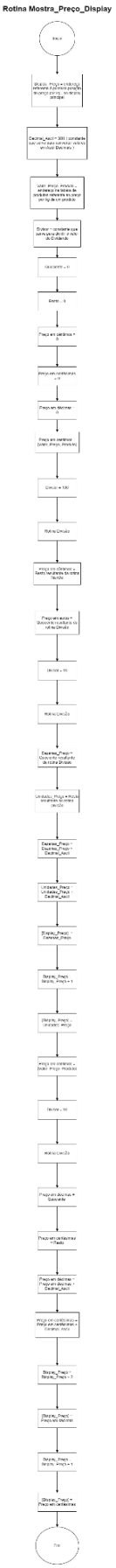
18. Mostra Lista



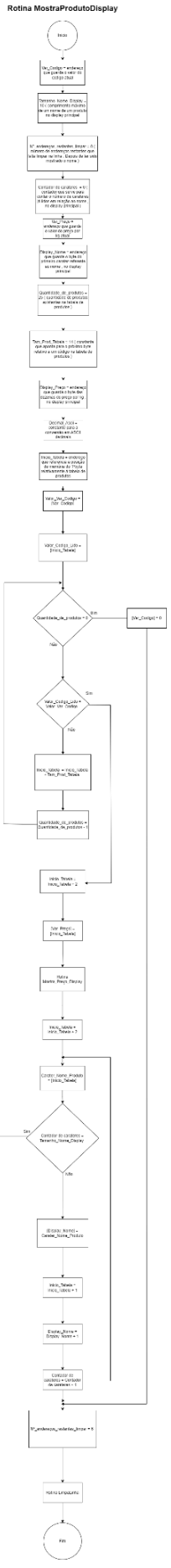
19. Mostra Peso Display



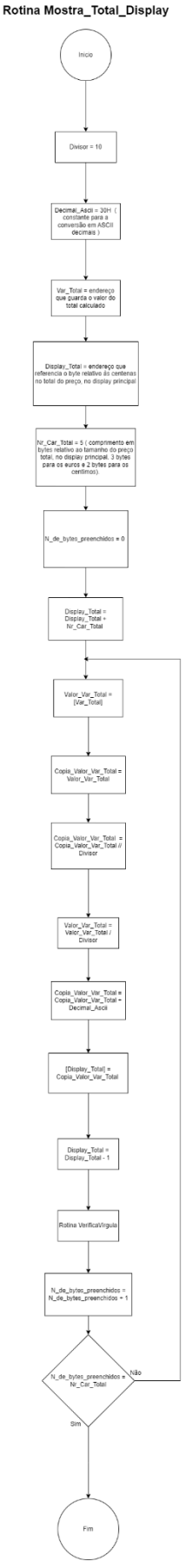
20.Mostra Preço Display



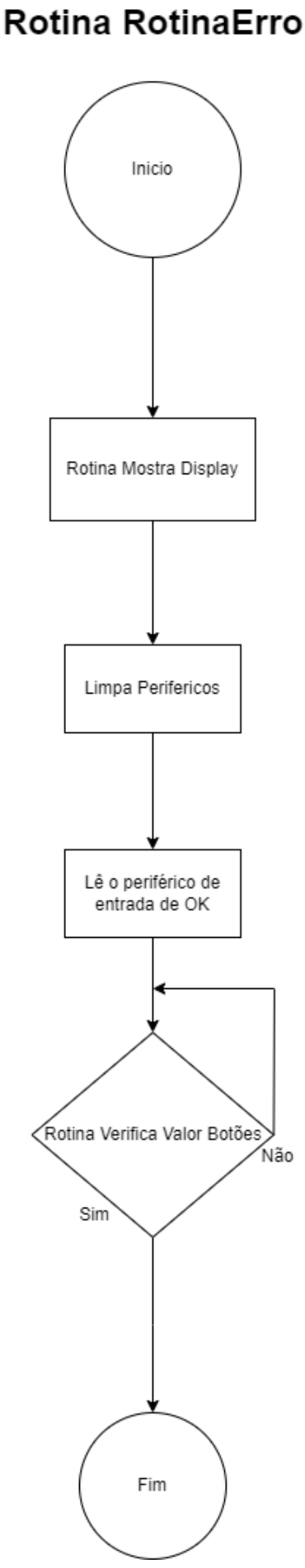
21.Mostra Produto Display



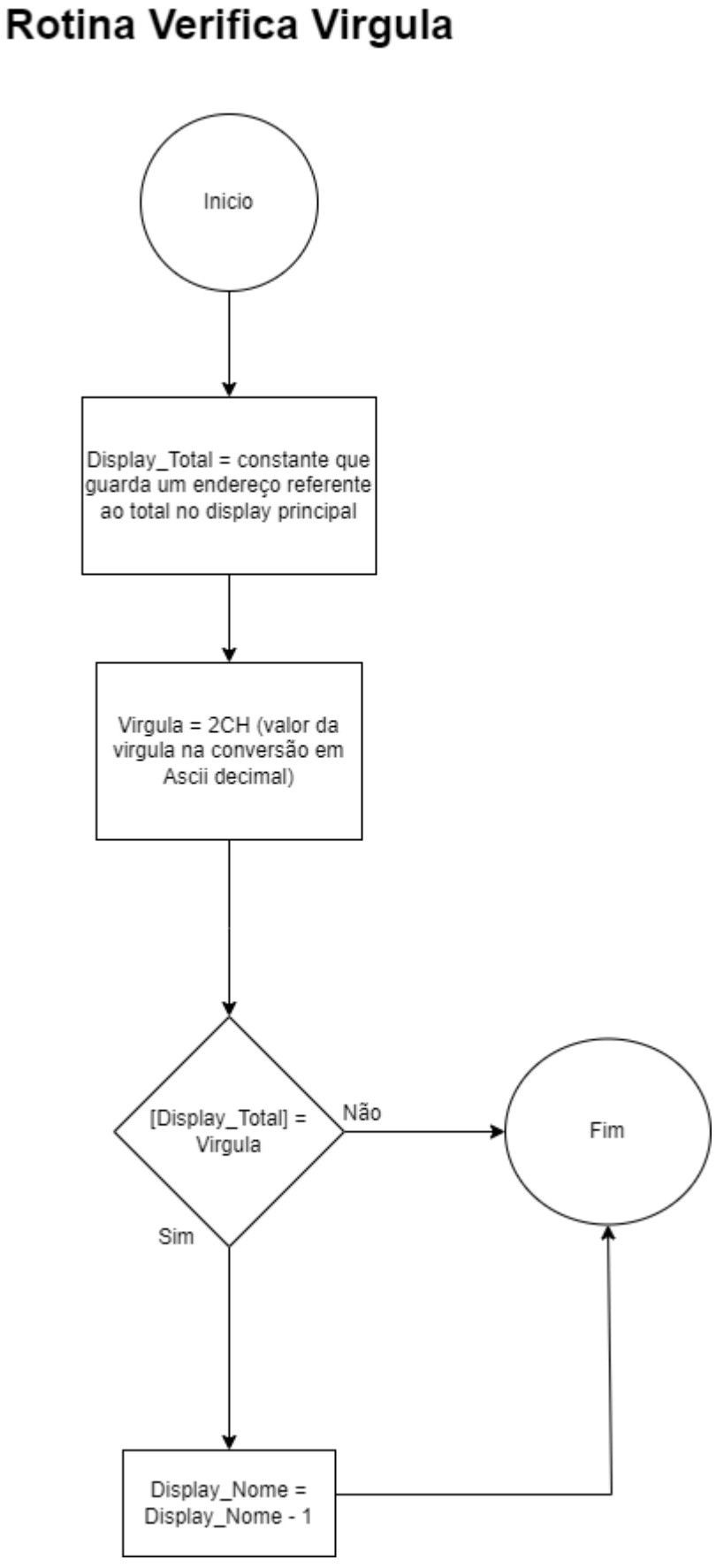
22.Mostra Total Display



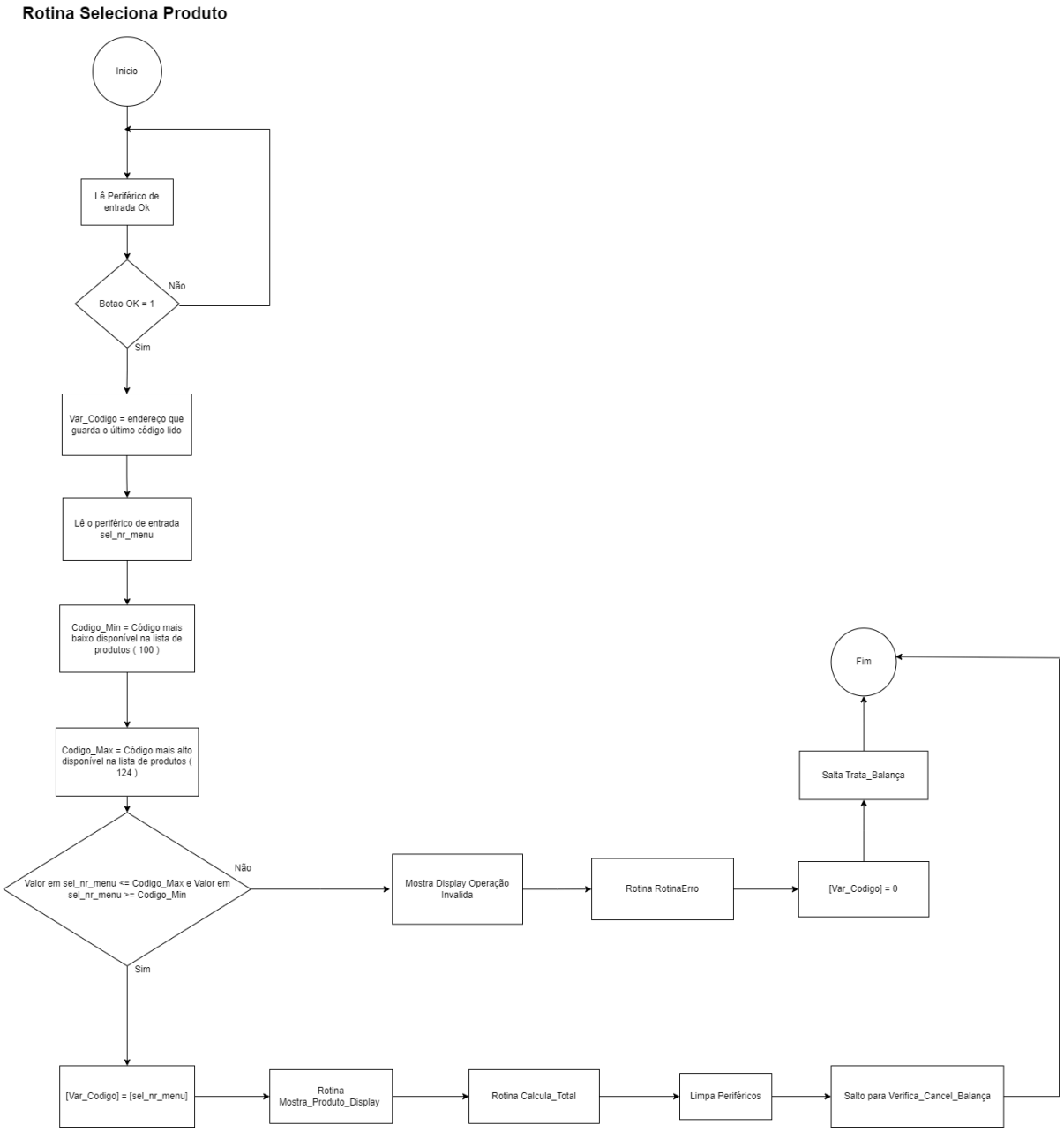
23. Rotina Erro



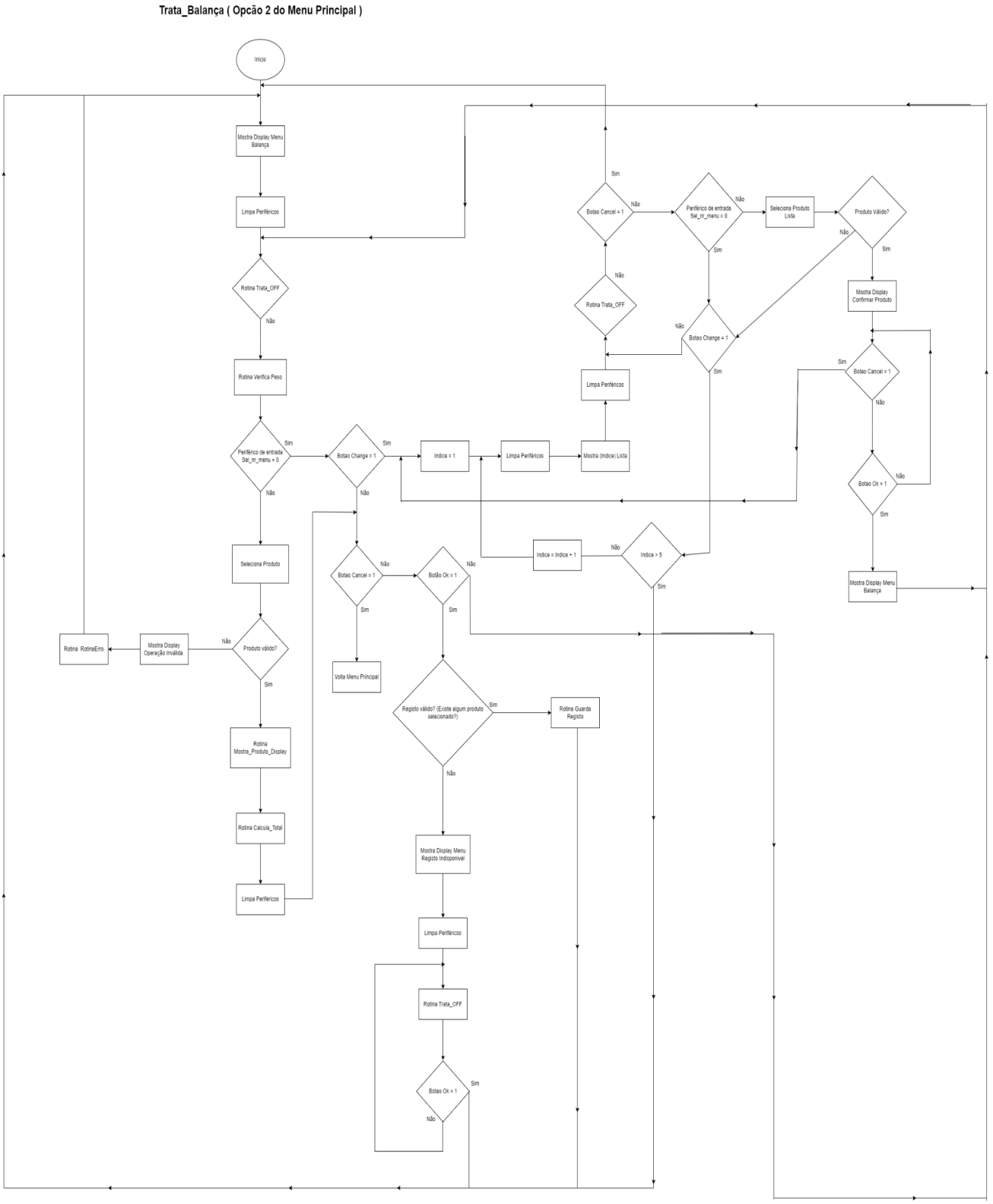
24. Verifica Virgula



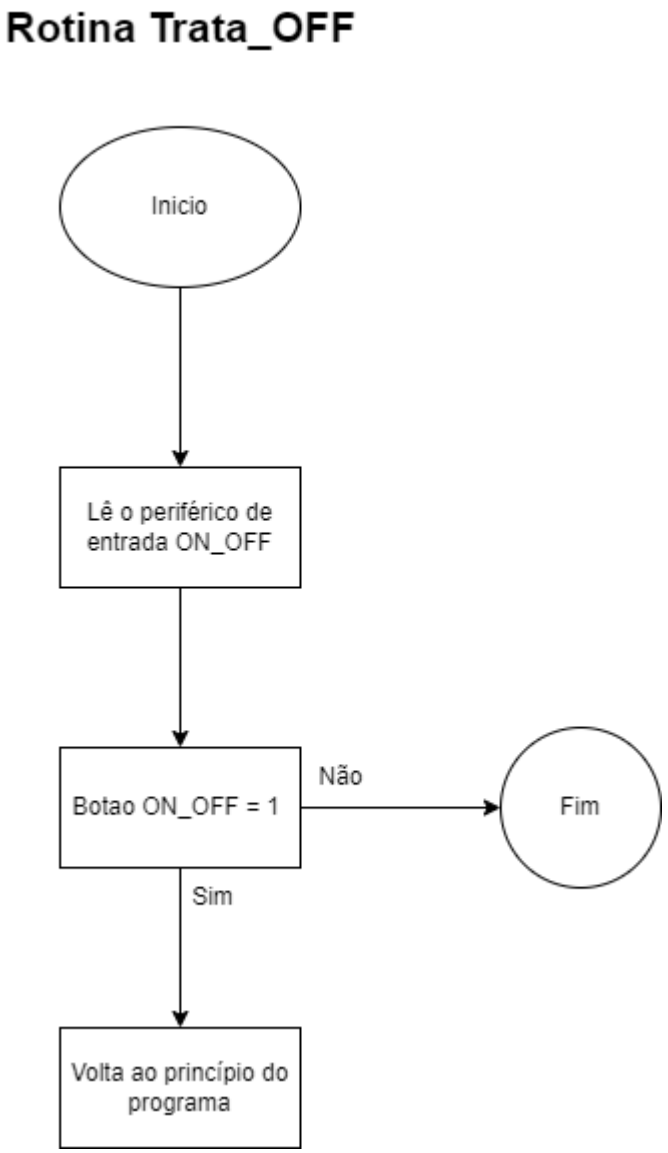
25. Seleciona Produto



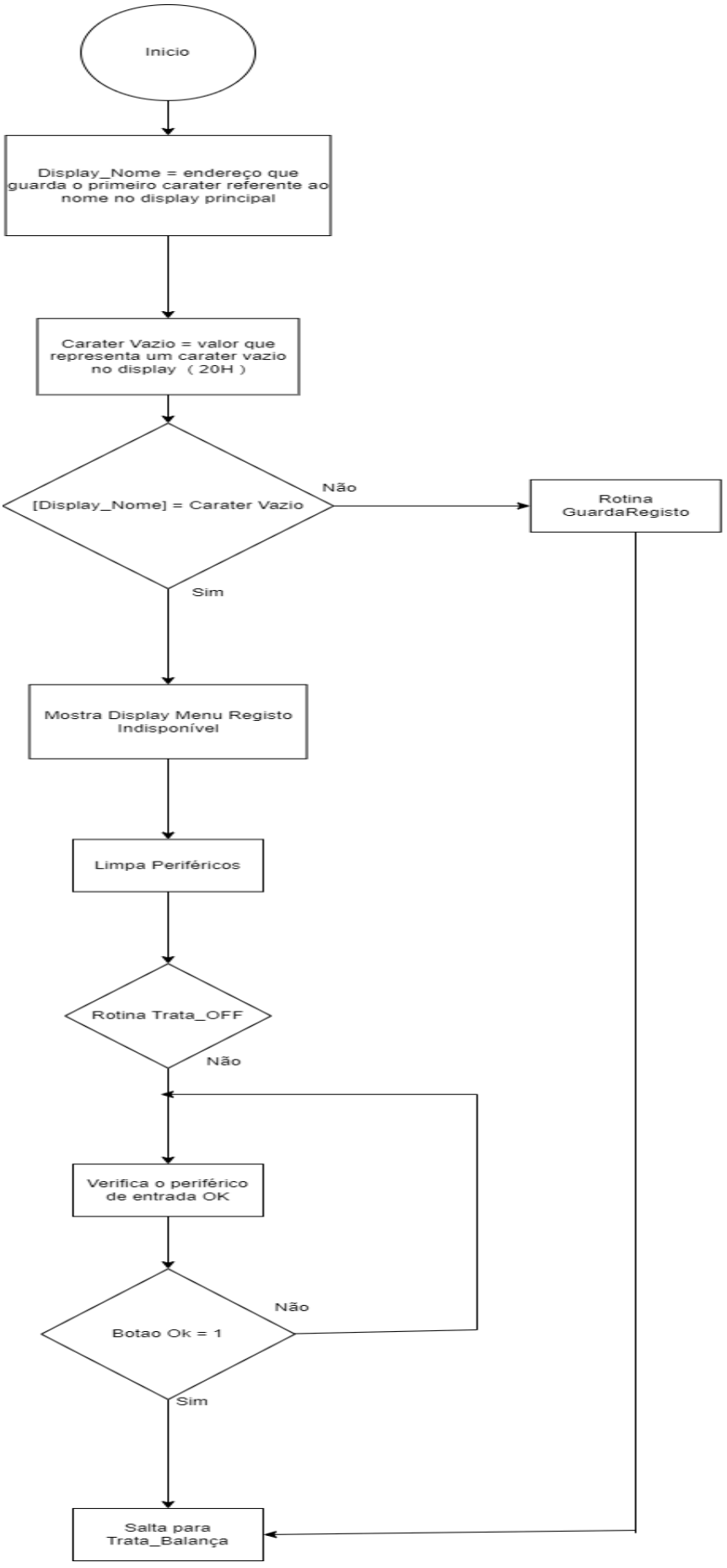
26.Trata Balança



27. Trata_OFF

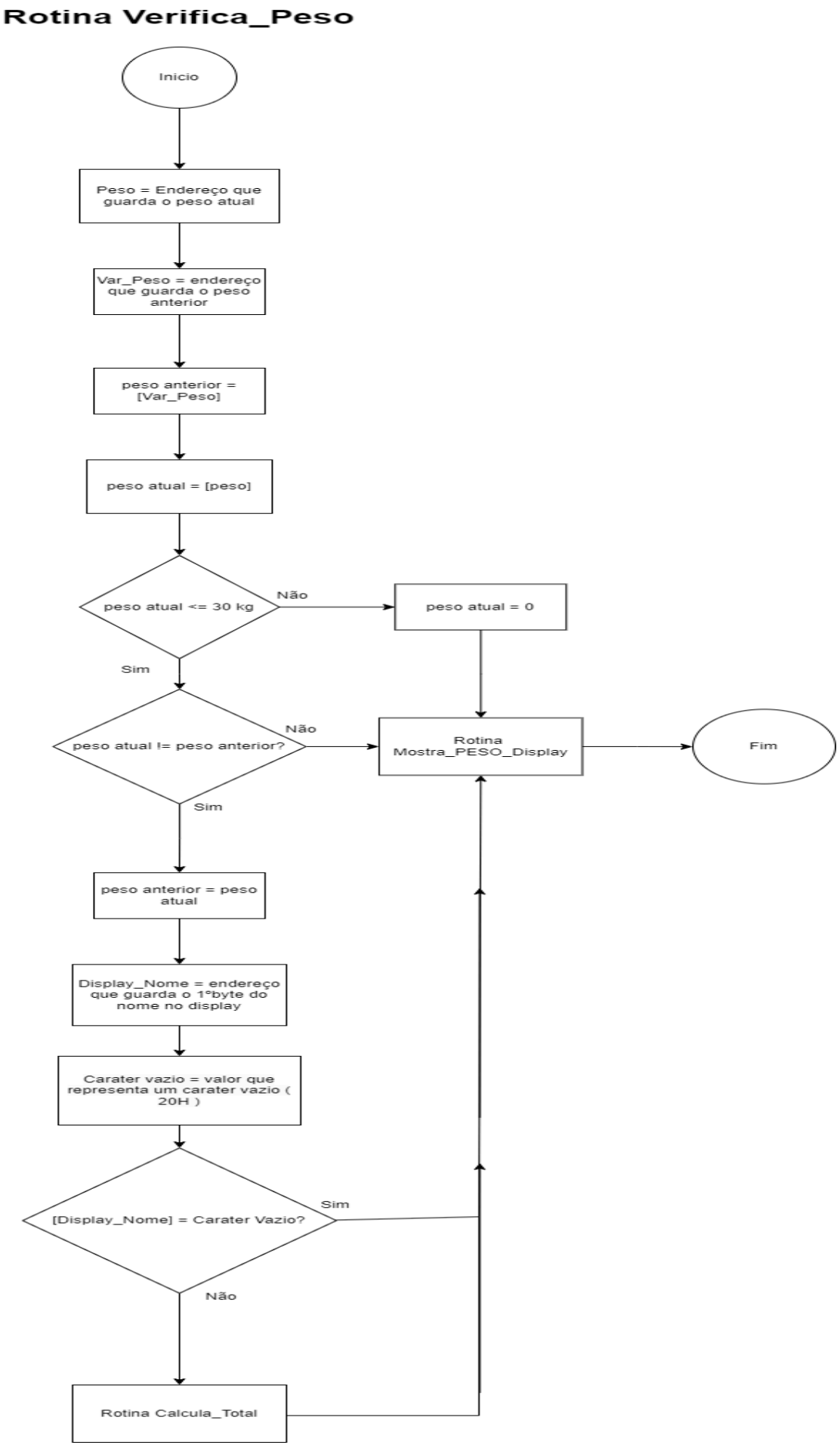

Rotina Trata_OFF
Inicio
Lê o periférico de entrada ON_OFF
Botao ON_OFF = 1
Não
Fim
Sim
Volta ao princípio do programa

28. Trata Registo Indisponível

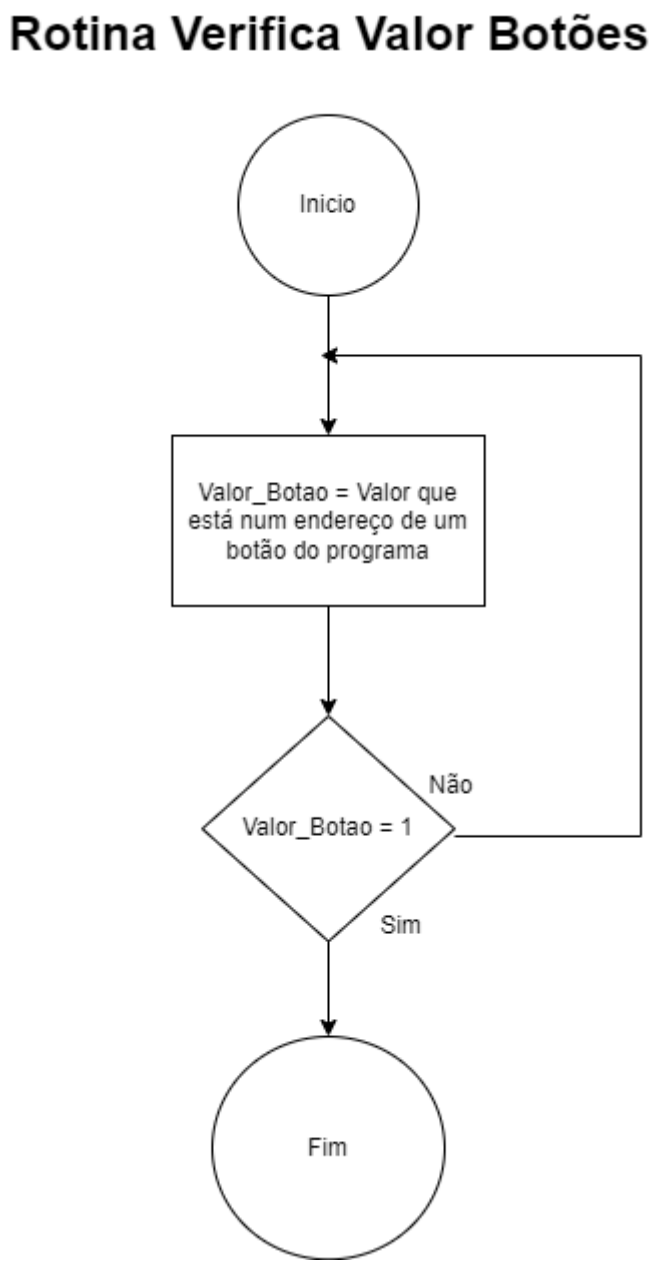


29.Trata Limpar

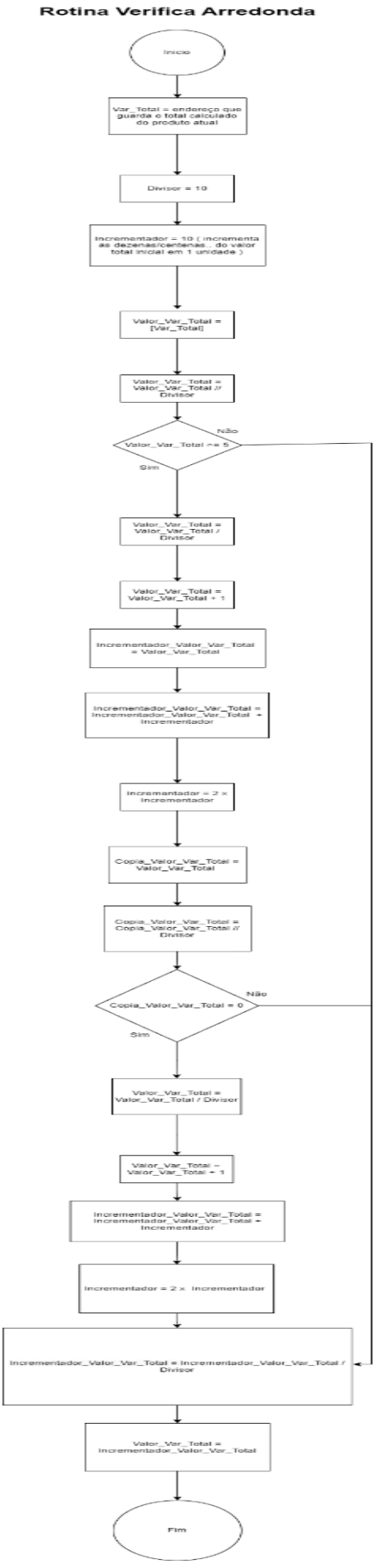
30. Verifica Peso



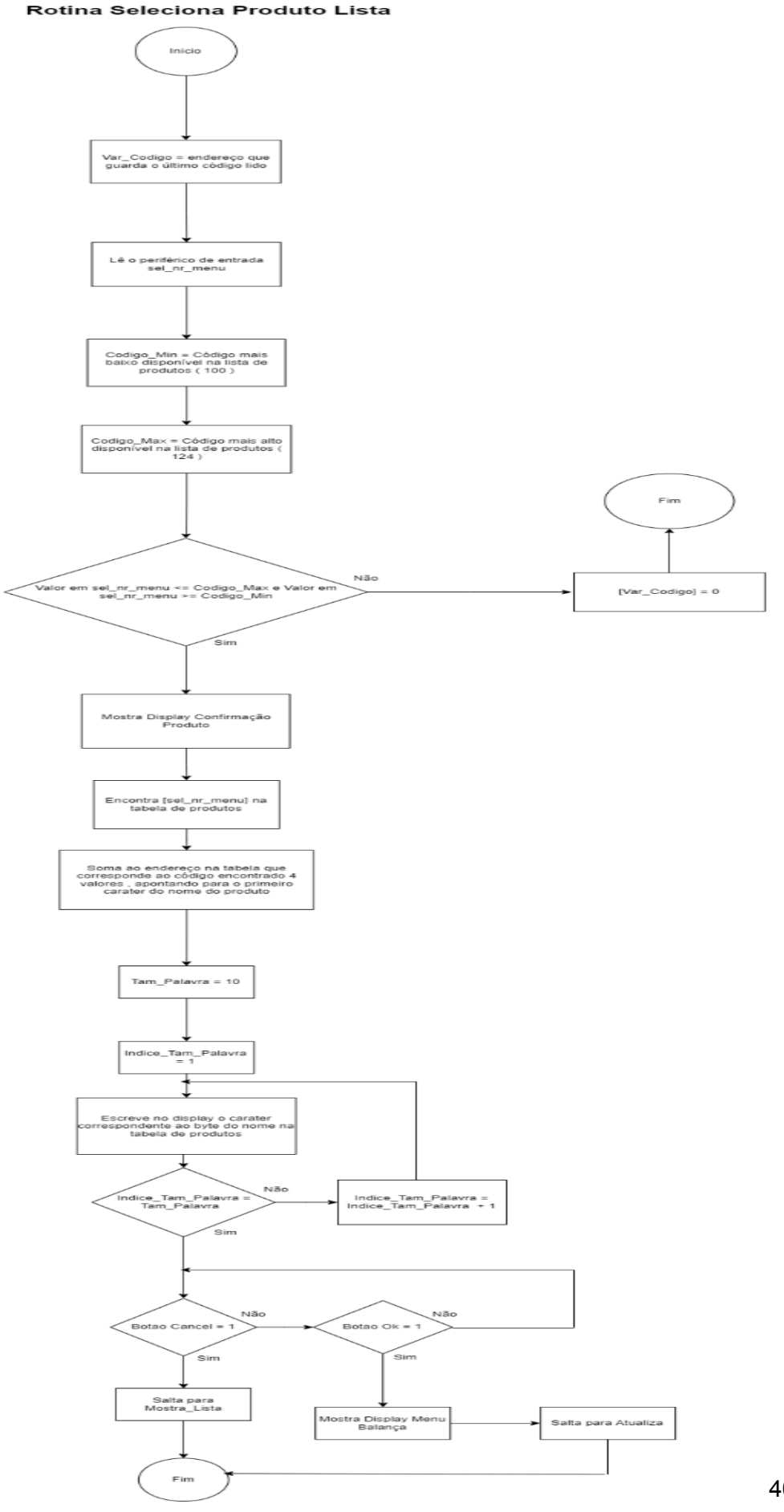
31.Verifica Valor Botões



32. Verifica Arrendonda



33. Seleciona Produto Lista



Anexo B

```
; Periféricos de entrada
ON_OFF                  EQU             1A0H            ;Localização do botão ON/OFF
SEL_NR_MENU             EQU             1B0H            ;Localização dos switches Opção(8
BITS)
OK                      EQU             1C0H            ;Localização do botão
OK
CHANGE                  EQU             1D0H            ;Localização do botão CHANGE
CANCEL                  EQU             1E0H            ;Localização do botão CANCEL
PESO                    EQU             1F0H            ;Localização dos switches para
introduzir o peso(16 BITS)

; Periféricos de saída
Display                 EQU             200H            ; Endereço onde começa o display
Display_end     EQU             26FH            ; Endereço onde termina o display

; Opções do Menu Principal
MBalança        EQU     1               ; Opção do Modo Balança
MRegistos               EQU     2               ; Opção do Modo Visuzlizar
Registos
MLimpar                 EQU     3               ; Opção do Modo Limpar Registos

; Posições dos valores no Display
Display_Nome    EQU             210H            ; Posição do primeiro caracter referente ao
nome no display
Display_Nome_Fim EQU            21FH            ; Posição do último caracter referente ao nome no
display
Display_Peso    EQU             23AH            ; Posição do primeiro caracter referente ao
peso no display
Display_Preço   EQU             255H            ; Posição(ímpar) do primeiro caracter
referente ao preço no display
Display_Total   EQU             267H            ; Posição do primeiro caracter referente ao
total no display
Display_Peso_Reg  EQU           22AH            ; Posição do primeiro caracter referente ao peso no
menu Registos
Display_Total_Reg  EQU          237H            ; Posição do primeiro caracter referente ao total no
menu Registos
Display_Confirmacao_Produto EQU 0215H           ; Posição do primeiro caracter referente ao nome no menu
Confirmação de produto

; Constantes
Max_PESO                EQU             12CH            ; Máximo peso permitido na
balança = 30.000 gramas
Qtdd_Produtos   EQU             25              ; Quantidade total de produtos na
tabela de produtos
Codigo_Min              EQU             100             ; Valor do primeiro código na tabela
de produtos
Codigo_Max              EQU             124             ; Valor do último código na tabela
de produtos
Decimal_Ascii   EQU             30H             ; Constante para a conversão em
ASCII decimais
Ponto                   EQU             00H                     ; Constante referente ao
ponto em Hexadecimal
Virgula                 EQU             2CH                     ; Constante referente à
vírgula em Hexadecimal
CaracterVazio   EQU     20H             ; Caracter para limpar o ecra(vazio)
Nr_Car_Total    EQU             5               ; Nr de carateres do total a
preencher
Tam_Registos    EQU             18              ; Nr de bytes ocupado na memória
por cada Registo
Tam_Nome_DisplayEQU     10              ; Espaço ocupado pelo nome no display
Tam_Peso_Display  EQU   3               ; Espaço ocupado pelo Peso no display
```

```
Tam_Total_Display  EQU  5                    ; Espaço ocupado pelo Total no display
Incr_Pos_Registo   EQU  48                   ; Constante para somar às posições no display do
menu Registos para o segundo registo
Tamanho_Codigo     EQU  3                    ; Tamanho do código em bytes, a ser escrito no display
Tam_Prod_Tabela    EQU  14                   ; Espaço de memória ocupado por cada
produto na tabela

; Outros endereços
Var_PESO           EQU  7000H                ; Espaço na memória para guardar
o último valor do peso na balança
Var_Codigo         EQU  7100H                ; Endereço onde se guarda o código do produto atual
Var_Preço          EQU  7200H                ; Endereço onde se guarda o preço
do produto atual
Var_Total          EQU  7300H                ; Endereço onde se guarda o total
Ptr_Historico      EQU  7400H                ; Variável que guarda o próximo endereço
livre para guardar registos(Ponteiro)
Inicio_Tabela      EQU  5000H                ; Endereço onde começa a tabela de
produtos inserida na memória
Ult_Produto        EQU  5039H                ; Endereço que guarda o último
produto na tabela de produtos
Inicio_Historico   EQU  2500H                ; Posição do primeiro caracter referente ao display dos registos
guardados
Fim_Historico      EQU  2800H                ; Posição do ultimo endereço referente aos registos guardados

; Configuração da Stack
PLACE 2000H
Fim_Stack:                                   ; Fim da Stack - 2000h
Stack 200H
Inicio_Stack:                                ; Inicio da Stack - 2000h + 200h*2 =
2400h

;-------------------------------------------------------------;
;          Os Menus encontram-se inseridos no ficheiro Memória.dat   ;
;-------------------------------------------------------------;
MenuPrincipal                  EQU  3000H
MenuBalança                    EQU  3100H
MenuRegistos                   EQU  3200H
MenuReset                      EQU  3300H
MenuOverflow                   EQU  3400H
MenuOpInvalida                 EQU  3500H
MenuLista                      EQU  3600H
MenuSucessoApagarHistorico     EQU  3700H
MenuRegistoVazio               EQU  3800H
MenuSucessoRegisto             EQU  3900H
MenuLimiteRegistos             EQU  3A00H
MenuRegistoIndisponivel        EQU  3B00H
MenuConfirmacaoProduto         EQU  3C00H
MenuFimRegistos                EQU  3D00H

Place 0000H                                  ; Memoria 1 - Depois do
Reset
Inicio:
        MOV R0, Principio
        JMP R0                               ; Salta para o programa
Principal

Place 1000H                                  ; Memoria 2
Principio:
        MOV SP, Inicio_Stack                 ; Inicio da Stack - 2400H
        CALL Limpa_Registos                  ; Limpa os registos guardados na
base de dados
        CALLF Inicia_Ponteiro_Registos       ; Inicia o apontador para o endereço do primeiro registo
na base de dados
        CALLF LimpaDisplay                   ; Coloca o display em branco
        CALL LimpaPerifericos                ; Atribui o valor zero aos periféricos de
entrada
```

```
                CALLF LimpaVariaveis                                    ; Limpa endereços que contêem as variáveis,
excepto o peso
                MOV R0, ON_OFF                                                  ; R0 -> End. ON_OFF
                CALLF Trata_ON                                                  ; Fica à espera que a balança seja
ligada ([1A0H] = 1)
Inicio_Menus:
                MOV R2, MenuPrincipal                                   ; R2 -> primeiro endereço com layout do
Menu Principal
                CALL MostraDisplay                                              ; Mostra Menu no Display
                CALL LimpaPerifericos                                   ; Atribui o valor zero aos periféricos de
entrada
Le_Opcao:
                CALL Trata_OFF                                                  ; Verifica se foi pressionado o botão
ON_OFF(para desligar a balança)
                MOV R0, SEL_NR_MENU                                             ; Endereço do Seletor
                MOVB R1, [R0]                                                   ; Ler Opção no Menu
                CMP R1, 0                                                               ; Opção = 0?
                JEQ Le_Opcao                                                    ; Volta a ler a Opção enquanto não
for inserido um valor diferente de zero
                CMP R1, MBalança                                        ; Opção = Modo Balança(1)?
                JEQ Trata_Balança                                       ; Abre o menu do modo Balança, que
possibilita pesar e registar produtos
                CMP R1, MRegistos                                               ; Opção = Modo Registos(2)?
                JEQ Salto_Trata_Visualiza_Registos  ; Abre o menu do modo Registos, que possibilita visuzlizar os
registos efetuados
                CMP R1, MLimpar                                                 ; Opção = Modo Limpar
Histórico(3)?
                JEQ Salto_Trata_Limpar                                  ; Abre o menu do modo Reset, que possibilita
limpar os registos realizados desde que a balança foi ligada
                MOV R2, MenuOpInvalida                                  ; R2 -> primeiro endereço com layout do
Menu Opção Inválida
                CALL RotinaERRO                                                 ; Trata opção diferente de 0, 1, 2 ou
3
                JMP Inicio_Menus                                        ; Volta ao Menu Principal

;--------------------------------------------------------------------------------------------
;   Inicia Ponteiro de Registos - Rotina que coloca o apontador no endereço do primeiro registo
;--------------------------------------------------------------------------------------------
Inicia_Ponteiro_Registos:
                MOV R0, Inicio_Historico                ; Primeiro endereço da zona de memória que guarda o
histórico de registos
                MOV R1, Ptr_Historico                                   ; Variável que guarda o endereço seguinte
disponível para guardar um registo
        MOV [R1], R0                                            ; Ptr_Historico = [Inicio_Historico]
                RETF

;--------------------------------------------------------------------------------------------
;   Rotina Liga - Rotina para verificar se foi pressionado o botão ON_OFF quando balança está desligada
; R0 = Endereço do botão
;--------------------------------------------------------------------------------------------
Trata_ON:
                MOVB R1, [R0]                                                   ; Obtem valor do botão
                CMP R1, 1                                                               ; Botão = 1?
                JNE Trata_ON                                                    ; Se botão diferente de 1, volta a ler
o valor
            RETF

;--------------------------------------------------------------------------------------------
;   Rotina Desliga - Rotina para verificar se foi pressionado o botão ON_OFF no decorrer do programa
;--------------------------------------------------------------------------------------------
Trata_OFF:
                PUSH R0
                PUSH R1
                MOV R0, ON_OFF                                                  ; R0 -> End. ON_OFF
                MOVB R1, [R0]                                                   ; Ler botão ON_OFF
                CMP R1, 1                                                               ; ON_OFF = 1?
```

```
                JEQ Principio                                   ; Caso seja 1 volta ao Principio e
limpa Display
                POP R1
                POP R0
                RET

;---------------
;   Modo Balança
;---------------
Trata_Balança:
                MOV R2, MenuBalança                             ; R2 -> primeiro endereço com
layout do Menu Balança
                CALL MostraDisplay                              ; Mostra Menu Balança no Display
                CALL LimpaPerifericos                   ; Atribui o valor zero aos periféricos de
entrada
Atualiza:
                CALL Trata_OFF                                  ; Verifica se foi pressionado botão
para desligar a balança
                CALL Verifica_PESO                              ; Atualiza peso no Display
                MOV R0, SEL_NR_MENU                             ; R0 -> Periférico SEL_NR_MENU
                MOV R1, [R0]                                    ; R1 = [SEL_NR_MENU]
                CMP R1, 0                                               ; [SEL_NR_MENU] = 0?
                JNE Salto_Seleciona_Produto             ; Se é diferente de 0 verifica o produto selecionado
                CALLF Verifica_CHANGE_Balança   ; Verifica se foi pressionado o botão CHANGE para
mostrar a lista de produtos e os seus respetivos códigos
                JMP Verifica_CANCEL_Balança             ; Verifica se foi pressionado o botão CANCEL
para voltar ao Menu Principal

;-------------------------------------------------------------------------------
; Verifica se o botão Change foi pressionado no modo balança para mostrar a lista de produtos
;-------------------------------------------------------------------------------
Verifica_CHANGE_Balança:
                MOV R0,CHANGE                                   ; R0 -> Periférico CHANGE
                MOVB R1,[R0]                                    ; R1 = [CHANGE]
                CMP R1,1                                                ; [CHANGE] = 1?
                JEQ Mostra_Lista                                ; Se foi pressionado o botão CHANGE mostra
a lista de produtos
                RETF

;-------------------------------------------------------------------------------
; Verifica se o botão Cancel poi pressionado no modo balança para voltar ao M. Principal
;-------------------------------------------------------------------------------
Verifica_CANCEL_Balança:
                MOV R0, CANCEL                                  ; R0 -> Periférico CANCEL
                MOVB R1, [R0]                                   ; R1 = [CANCEL]
                CMP R1, 1                                               ; [CANCEL] = 1?
                JEQ Inicio_Menus                                ; Se o botão cancel foi premido volta ao menu
principal
                JMP Verifica_Registo                    ; Verifica se houve solicitação para registar
um produto

;-------------------------------------------------
; Verifica confirmação de registo no modo balança
;-------------------------------------------------
Verifica_Registo:
        MOV R0,OK                                               ; R0 -> Periférico OK
        MOVB R1,[R0]                                    ; R1 = [OK]
        CMP R1,1                                                ; [OK] = 1?
        JNE Atualiza                                    ; se nem o botao ok nem o cancel foram premidos atualiza o
display com possíveis novos valores
                JMP Trata_Registo_Indisponivel  ; Verifica se é válido o pedido de registo
Guarda_registo:
                CALL GuardaRegisto                              ; quando o botao ok for premido
avança para a rotina GuardaRegisto
                JMP Trata_Balança                               ; Volta para o início do modo Balança
```

```
;-------------------------------
; Verifica se é válido o registo
;-------------------------------
Trata_Registo_Indisponivel:
                MOV R3, Display_Nome                    ; R3 -> Endereço da primeira posição do
nome no menu Balança
                MOVB R4, [R3]                           ; R4 = [R3], suposto primeiro
caracter do nome
                MOV R5, CaracterVazio                   ; R5 = 20H(corresponde ao caracter
vazio(ASCII))
                CMP R5, R4                              ; Verifica se existe algum
caracter não vazio na primeira posição do nome no display
                JNE Guarda_registo                      ; Se existir significa que foi
selecionado um produto e procede para guardar o registo
                MOV R2, MenuRegistoIndisponivel         ; Se não tiver sido selecionado nenhum produto mostra
mensagem de registo indisponível
                CALL MostraDisplay                      ; Mostra mensagem no display
                CALL LimpaPerifericos                   ; Linmpa os periféricos excepto o PESO
                CALL Trata_OFF                          ; Verifica se foi pressionado
ON/OFF para desligar a balança
                MOV R0, OK                              ; R0 -> Periférico OK
                CALLF Verifica_Valor_Botoes             ; Espera que seja pressionado o botão OK para dar
confirmação e limpar a mensagem
                JMP Trata_Balança                       ; Volta para o início do modo Balança


;------------------------
; Salta para Trata_Limpar
;------------------------
Salto_Trata_Limpar:
                JMP Trata_Limpar                        ; Salta para a função de limpar o histórico de
registos(salto anterior estava fora do alcance -255 - 254)

;--------------------------------------
; Salta para Trata_Visualiza_Registos
;--------------------------------------
Salto_Trata_Visualiza_Registos:
                JMP Trata_Visualiza_Registos            ; Salta para a função de visualizar os registos(salto
anterior estava fora do alcance -255 - 254)

;-------------------------------
; Salta para Seleciona_Produto
;-------------------------------
Salto_Seleciona_Produto:
                JMP Seleciona_Produto                   ; Salta para a função de seleção de
produto(salto anterior estava fora do alcance -255 - 254)

;-----------------------------------------------------------------------
; Mostra lista de produtos disponíves na tabela de produtos na base de dados
;-----------------------------------------------------------------------
Mostra_Lista:
                CALL LimpaPerifericos                   ; Limpa os periféricos excepto o PESO
                MOV R7, Ult_Produto                     ; constante que guarda o endereço do ultimo
produto a ser lido na tabela
                MOV R8, Inicio_Tabela                   ; R8 -> Endereço do inicio da tabela de
produtos
                MOV R5, 1                               ; contador para verificar se ja
percorremos as 5 páginas referentes à lista de produtos no display
                MOV R10, 70                             ; constante que guarda o
salto para o proximo ultimo produto a ser escrito numa página da lista
Ciclo_Proxima_Lista:
                MOV R2,MenuLista                        ; endereço do menuLista
                CALL MostraDisplay                      ; Mostra o layout do MenuLista no
Display
                MOV R1,Display_Nome                     ; R1 -> Endereço do primeiro
caracter do nome no display
```

```
                MOV R2,Tamanho_Codigo                   ; R2 = Nº de bytes para o código no display
;constantes a usar na Lista
                MOV R3,12
                MOV R4,10
                MOV R9,4
        Ciclo:
                CALL Converte_Num_Caracter              ; chamada para converter o codigo em 3
caracteres(bytes)
                ADD R8,R9                                                       ; aponta para o proximo
endereco da tabela de produtos a ser lido (nome)
                ADD R1,R9                                               ; aponta para a posicao do display
para escrita do nome do produto
                CALL Mostra_Nome_Lista                          ; Escreve o nome do produto no
display(MenuLista)
                ADD R1,R3                                               ; Faz com que o endereço do
display fique a apontar para o comeco da linha seguinte, pronto a receber o novo codigo
                ADD R8,R4                                                       ; Faz com que o
endereço da tabela de produtos aponte para o proximo codigo
                CMP R8,R7                                                       ; verifica se ja chegámos
ao ultimo produto da lista
                JLE Ciclo                                               ; se não chegámos continua na
mesma lista e avança para o proximo produto
                ADD R7,R10                                                      ; se ja chegámos guarda
em R7 o valor do proximo ultimo byte a ser escrito na lista ( produto )
                CALL LimpaPerifericos                           ; Limpa os periféricos excepto o PESO
        Ciclo_Verifica_Codigo_Produto:          ; verifica se numa lista o utilizador introduzirá um produto ou
avançará para a proxima lista
                CALL Trata_OFF                                          ; Verifica se foi pressionado
ON/OFF para desligar a balança
                MOV R0,CANCEL                                           ; R0 -> Periférico CANCEL
                MOVB R1,[R0]                                            ; R1 = [CANCEL]
                CMP R1,1                                                        ; [CANCEL] = 1?
                JEQ Trata_Balança                               ; Se foi pressionado o botão CANCEL volta
para o menu Balança
                MOV R0,SEL_NR_MENU                                      ; R0 -> Periférico SEL_NR_MENU
                CALLF Seleciona_Produto_Lista           ; Trata da seleção do produto através da introdução do
código na entrada SEL_NR_MENU
                MOV R0,CHANGE                                           ; R0 -> Periférico CHANGE
                MOVB R1,[R0]                                            ; R1 = [CHANGE]
                CMP R1,1                                                        ; [CHANGE] = 1?
                JNE Ciclo_Verifica_Codigo_Produto       ; se o botao change nao foi pressionado e o valor em
sel_nr_menu é inválido repete o ciclo, até que um seja pressionado/válido
                ADD R5,1                                                ; incrementa o valor do contador ou
seja mais uma lista de 5 produtos foi mostrada(5 produtos da tabela percorridos)
                CMP R5,5                                                        ; verifica se já chegámos
ao fim das 5 listas.  25 produtos / 5 listas = 5 produtos p/lista
                JGT Acaba_Lista                                 ; se já , então retorna ao
menuBalança
                JMP Ciclo_Proxima_Lista                 ; se change foi pressionado , avança para a proxima
lista
        Acaba_Lista:                                                    ; etiqueta para simbolizar que ja
foram percorridas todas as listas . Se chegou aqui nenhuma opcao de sel_nr_menu foi valida
                JMP Trata_Balança                               ; salto para o menu Balança

;-------------------------------------------------------------------------------------------------------------------------
; Seleciona produto da lista - Rotina para verificar se o valor em sel_nr_menu em alguma lista é valido e confirmar a escolha
do produto
; R0 = Endereço do periférico de entrada SEL_NR_MENU
;-------------------------------------------------------------------------------------------------------------------------
Seleciona_Produto_Lista:
                MOVB R9, [R0]                                           ; R9 = [SEL_NR_MENU]
                MOV R3, Codigo_Min                                      ; R3 = min(100)
    CMP R9, R3                                                  ; [SEL_NR_MENU] < min
    JLT Produto_Invalido_Lista                  ; Se código introduzido for menor que o menor dos códigos na
tabela, irá corresponder a uma opção de produto inválida
                MOV R3, Codigo_Max                                      ; R3 = max(124)
```

```
    CMP R9, R3                                          ; [SEL_NR_MENU] > max
    JGT Produto_Invalido_Lista                  ; Se código introduzido for maior que o maior dos códigos na
tabela, irá corresponder a uma opção de produto inválida
                ;Produto Válido
    MOV R2, MenuConfirmacaoProduto          ; Menu que serve para confirmarmos o produto
                CALL MostraDisplay                                  ; Mostra o Menu de confirmação no
display
                MOV R5, Inicio_Tabela                       ; R5 -> Inicio da tabela de produtos
                MOV R7, Display_Confirmacao_Produto ; endereço da posição no MenuConfirmacaoProduto da 1ºletra do
nome do produto
                ; constantes para percorrermos toda as letras do nome do produto
                MOV R8, 1
                MOV R4, 10
                MOV R10, 14                                         ; constante que guarda o valor do
salto para o proximo codigo disponivel na tabela de produtos
Ciclo_Obter_Codigo:                                         ; ciclo para identificarmos na tabela
de produtos o produto escolhido
                MOV R6, [R5]                                        ; R6 = [Endereço código Tabela]
                CMP R9, R6                                              ; [SEL_NR_MENU] =
[Endereço código Tabela]?
                JEQ IdentificaProduto                       ; Se o código introduzido no periférico
corresponder ao código no endereço atual da tabela identifica o produto
                ADD R5, R10                                             ; salto para o proximo
codigo disponivel na tabela de produtos
                JMP Ciclo_Obter_Codigo                      ; Passa para o próximo código na tabela
IdentificaProduto:
                ADD R5, 4                                           ; soma 4 ao endereço que
inicialmente aponta para o codigo do produto escolhido , para que no final aponte para o endereço da 1ºletra do nome do
produto escolhido
Ciclo_Escreve_Nome_Produto:
                MOVB R6,[R5]                                        ; R6 = Caracter do Nome do
produto
                MOVB [R7],R6                                        ; escreve no display e na posicao
correta o nome do produto
                CMP R8,R4                                               ; verifica se já
percorremos todas as letras do produto(10)
                JEQ Verifica_Opcoes_Botoes          ; se sim verifica se queremos avançar com o produto ou
cancelar
                ADD R8,1
                ADD R5,1
                ADD R7,1                                            ; incrementa quer o numero da letra
, quer o end. da tabela de produtos e do display
                JMP Ciclo_Escreve_Nome_Produto      ; Continua a escrever o nome do produto
Verifica_Opcoes_Botoes:
                MOV R0,CANCEL                                       ; R0 -> Periférico CANCEL
                MOVB R1,[R0]                                        ; R1 = [CANCEL]
                CMP R1,1                                                ; [CANCEL] = 1?
                JEQ Mostra_Lista                            ; se cancel estiver pressionado , volta para a
primeira página da lista no display
                MOV R0,OK                                               ; R0 -> Periférico OK
                MOVB R1,[R0]                                        ; R1 = [OK]
                CMP R1,1                                                ; [OK] = 1?
                JEQ Mostra_Produto_Balança          ; se ok estiver pressionado avança para o menuBalança
com o produto escolhido
                JMP Verifica_Opcoes_Botoes          ; so sai do ciclo quando algum deles for pressioando
Mostra_Produto_Balança:
                MOV R2,MenuBalança                                  ; R2 -> Endereço MenuBalança
                CALL MostraDisplay                                  ; Mostra MenuBalança no Display
                JMP Atualiza                                        ; avança para atualiza. O valor a ter
em conta será o valor valido escolhido pelo utilizador depois do visionamento do mesmo na lista
Produto_Invalido_Lista:
                MOV R0, Var_Codigo                                  ; R0 -> Endereço da variável que
guarda o último código lido
                MOV R9, 0                                               ; R9 = 0
                MOV [R0], R9                                        ; guarda na variavel que guarda o
codigo atual o valor 0, pois o codigo inserido foi invalido
```

```
                RETF

;------------------------
; Salta para Trata_Balança
;------------------------
Salto_Trata_Balança:
                JMP Trata_Balança                                       ; Salta para a função que trata o Modo
Balança(salto anterior estava fora do alcance -255 - 254)

;--------------------------------------------------------------------------------
; Mostra Nome Lista - Rotina para mostrar o nome de cada produto correspondente no display
; R1 = Endereço do display
; R8 = Endereço da tabela de produtos
;--------------------------------------------------------------------------------
        Mostra_Nome_Lista:
                PUSH R4                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R1
                PUSH R8
                PUSH R5
                PUSH R6
                MOV R5, 1                                               ; indice para verificarmos
se ja percorremos todas as letras (bytes) do nome do produto, na tabela de produtos
                MOV R6, 9                                               ; 9 bytes
        Ciclo_Mostra_Nome_Lista:
                MOVB R4, [R8]                                   ; R8 = Endereço da tabela de
produtos
                MOVB [R1], R4                                   ; R1 = Display_Nome
                CMP R5, R6                                              ; Indice = 10?

                JGT Salta_Fim_Mostra_Nome_Lista         ; se já chegou ao fim do nome então saimos da rotina

                ADD R1,1                                        ; incrementa o endereço do display
                ADD R8,1                                                ; incrementa o endereço da tabela
de produtos
                ADD R5,1                                                ; avança para a proxima letra a ser
escrita no display(indice = indice + 1)
                JMP Ciclo_Mostra_Nome_Lista                     ; Continua a escrever o nome no Display
        Salta_Fim_Mostra_Nome_Lista:
                POP R6                                                  ; Repor os valores
presentes nos registos antes desta chamada
                POP R5
                POP R8
                POP R1
                POP R4
                RET

;--------------------------------------------------------------------------------------------------------------
; Converte num caracter - rotina para converter o codigo dos produtos numa palavra de 3 bytes representada no display
; R1 = Posição no display(Nome do produto)
; R2 = 3(nº de bytes para mostrar o código no display)
; R8 = Endereço da tabela de produtos
;--------------------------------------------------------------------------------------------------------------
        Converte_Num_Caracter:
                PUSH R1                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R2
                PUSH R3
                PUSH R4
                PUSH R5
                PUSH R6
                PUSH R7
                PUSH R8
                MOV R3, R2                                              ; R2 = Nº de bytes a serem escritos
no display (comprimento do codigo - 3 numeros)
                SUB R3, 1                                               ; i = 2;
```

```
                MOV R5, 10                                              ; constante que servirá
para dividirmos o resto
                MOV R7, Decimal_Ascii                   ; Constante para a conversão em ASCII
decimais
                ADD R1, R3                                      ; R1 = Endereço do Display + i
                ;Leitura de x
                MOV R4, [R8]                                    ; R4 = valor x -> valor periferico de
entrada
                MOV R6, R4                                      ; R6 = x
                MOD R6, R5                                      ; R6 = Resto (R6/10) = Resto(x/10)
                ADD R6, R7                                              ; R6 = C + 30H
                MOVB [R1], R6                                   ; Display(i) = C
                SUB R3, 1                                       ; decrementa o indice
                SUB R1, 1                                               ; decrementa o endereço
do display
        Ciclo_Converte_Num_Caracter:
                DIV R4, R5                                              ; R4 = Parte inteira
(R4/10) = Inteiro (x/10)
                CMP R4, 0                                               ; R4 = 0
                JEQ Caracter_Vazio_Converte_Num_Caracter
                MOV R6, R4                                              ; R6 = R4(Parte inteira da
divisão anterior)
                MOD R6, R5                                              ; R6 = Resto (R6/10)
                ADD R6, R7                                              ; R6 = C + 30H
                JMP Escreve_Display_Converte_Num_Caracter
        Caracter_Vazio_Converte_Num_Caracter:
                MOV R6, CaracterVazio                   ; R6 = 20H(Caracter Vazio)
        Escreve_Display_Converte_Num_Caracter:
                MOVB [R1], R6                                   ; Display(i) = C
                SUB R3, 1                                               ; Decrementa o indice
                SUB R1, 1                                               ; Decrementa o endereço
do display
                CMP R3, 0                                               ; i >= 0?
                JGE Ciclo_Converte_Num_Caracter         ; Continua a converter cada dígito num caracter
                POP R8                                                  ; Repor os valores
presentes nos registos antes desta chamada
                POP R7
                POP R6
                POP R5
                POP R4
                POP R3
                POP R2
                POP R1
                RET

;-------------------------------------------------------------------------------
; Seleciona Produto - Rotina para Selecionar Produto (Códigos válidos de 100 a 124)
;-------------------------------------------------------------------------------
Seleciona_Produto:
        MOV R0, OK                                              ; R0 -> Periférico OK
        CALLF Verifica_Valor_Botoes             ; Aguarda que seja pressionado o botão OK para validar o código
inserido
        ; Lê valor inserido nos switches
        MOV R0, SEL_NR_MENU                             ; R0 -> Periférico SEL_NR_MENU
        MOVB R1, [R0]                                   ; R1 = [SEL_NR_MENU]
        ; Verifica se está no intervalo válido (100 a 124)
                MOV R3, Codigo_Min                              ; R3 = min
        CMP R1, R3                                      ; R1 < min
        JLT Produto_Invalido                    ; Se código introduzido for menor que o menor dos códigos na
tabela, irá corresponder a uma opção de produto inválida
                MOV R3, Codigo_Max                              ; R3 = max
        CMP R1, R3                                      ; R1 > max
        JGT Produto_Invalido                    ; Se código introduzido for maior que o maoir dos códigos na
tabela, irá corresponder a uma opção de produto inválida
        ; Produto válido, guardar na variável Var_Codigo
```

```
    MOV R2, Var_Codigo                          ; R2 -> Endereco da variável que guarda o último
código lido
    MOVB [R2], R1                               ; [Var_Codigo] = [SEL_NR_MENU]
                CALL Mostra_produto_Display     ; Mostra produto no display
CalculoTotal:
                CALL Calcula_Total              ; Calcula o total com base no preço do
produto selecionado e no PESO introduzido
                MOV R0, Var_Codigo                  ; R0 -> Endereco da variável que
guarda o último código lido
                MOV R1, [R0]                        ; R1 = [Var_Codigo]
                CMP R1, 0                               ; [Var_Codigo] = 0?
                JEQ Salto_Trata_Balança         ; Se código guardado for igual a 0 reinicia o
modo balança
    JMP Fim_Seleciona_Produto           ; Caso contrário termina a seleção do produto
Produto_Invalido:                               ; Produto inválido, mostra menu de erro
                MOV R2, MenuOpInvalida          ; R2 -> primeiro endereço com layout do
Menu Opção Inválida
    CALL RotinaERRO                             ; Chama a rotina que trata erro
                MOV R0, Var_Codigo                  ; R0 -> Endereco da variável que
guarda o último código lido
                MOV R1, 0                               ; R1 = 0
                MOV [R0], R1                        ; [Var_Codigo] = 0
                JMP Trata_Balança               ; Volta para o MenuBalança
Fim_Seleciona_Produto:
                CALL LimpaPerifericos           ; Limpa Periféricos excepto o PESO
                JMP Verifica_CANCEL_Balança     ; Verifica se foi pressionado o botão CANCEL
no menu Balança

;--------------------------------------------------------------------------------
;   Verifica Peso - Rotina para verificar se houve alteração no PESO introduzido na balança
;--------------------------------------------------------------------------------
Verifica_PESO:
                PUSH R0                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                MOV R0, PESO                        ; R0 -> End. PESO
                MOV R2, [R0]                        ; R2 = PESO(atual)
                MOV R1, Var_PESO                    ; R1 -> End. PESO(guardado)
                MOV R3, [R1]                        ; R3 = PESO(anterior)
                MOV R4, Max_PESO                    ; R4 = 30Kg
                CMP R2, R4                              ; R2 <= 30Kg?
                JLE Atualiza_PESO               ; Se sim atualiza o PESO caso tenha alterado
                MOV R3, 0                               ; Para valores >30Kg o
peso fica a 0
PESO_Atualizado:
                MOV R2, R3                              ; R2 = PESO atual
                CALLF Mostra_PESO_Display       ; Mostra o peso atual no Display
                POP R4                                  ; Repor os valores
presentes nos registos antes desta chamada
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;-----------------------------------------------
;   Atualiza o PESO com novo valor na memória
;-----------------------------------------------
Atualiza_PESO:
                CMP R2, R3                              ; Se o peso foi alterado
guarda e atualiza no display
                JEQ Atualizado                          ; Se o peso não foi alterado
mantém o mesmo valor na memória
```

```
                MOV [R1], R2                                    ; Se o peso foi alterado guarda
novo valor
                MOV R3, [R1]                                    ; R3 = Peso guardado
                MOV R4, Display_Nome                    ; R4 -> 1º Endereço do nome no Display
                MOVB R6, [R4]                                   ; R6 = 1º Caracter do nome no
display
                MOV R5, CaracterVazio                   ; R5 = 20H(caracter vazio)
                CMP R6, R5                                              ; Campo do nome no
display está vazio?
                JEQ Atualizado                                  ; Peso é atualizado sem calcular o
Total, pois não existe um produto selecionado
                CALL Calcula_Total                      ; Se tiver sido selecionado um produto é
calculado o valor total
Atualizado:
                JMP PESO_Atualizado

;-------------------------
; Salta para Menu Principal
;-------------------------
Salto_Inicio_Menus:
                JMP Inicio_Menus                        ; Salta para o Menu Principal(salto anterior
estava fora do alcance -255 - 254)

;-----------------------------------------------------
; Trata_Visualiza_Registos - Mostra os registos guardados
;-----------------------------------------------------
Trata_Visualiza_Registos:
                MOV R0, Inicio_Historico        ; R0 -> Primeiro endereço do histórico de Registos
                MOV R7, 0                                               ; R7 = 0
                MOV R9, Incr_Pos_Registo                ; R9 = Constante a somar para a escrita do segundo
registo no Menu de Registos
Proximo_Visualiza:
                MOV R10, 2                      ; Mostrar até 2 registos por vez
                MOV R2, MenuRegistos                    ; R2 -> MenuRegistos
                CALL MostraDisplay                              ; Mostra MenuRegistos no Display
                MOV R5, Display_Nome                    ; R5 -> Posição do nome no display
                MOV R6, Display_Peso_Reg                ; R6 -> Posição do Peso no Menu de Registos
                MOV R8, Display_Total_Reg               ; R8 -> Posição do Total no Menu de Registos
Loop_Visualiza:
                MOVB R3, [R0]                   ; Lê primeiro byte do registo
                CMP R3, R7
                JEQ Sem_Registos                ; Se for 0, não há mais registos
                MOV R1, R5                                              ; R1 -> Posição do nome
no display
                MOV R4, Tam_Nome_Display                ; Contador de bytes copiados(começa com 10,
contagem decrescente)
                CALL Carrega_Registo                    ; Carrega Registo no Display
                ADD R5, R9                                              ; R5 -> Posição do nome
do segundo registo no display
                MOV R1, R6                                              ; R1 -> Posição do Peso
no Menu de Registos
                MOV R4, Tam_Peso_Display                ; Contador de bytes copiados(começa com 3, contagem
decrescente)
                CALL Carrega_Registo                    ; Carrega Registo no Display
                ADD R6, R9                                              ; R6 -> Posição do peso
do segundo registo no display
                MOV R1, R8                                              ; R1 -> Posição do Total
no Menu de Registos
                MOV R4, Tam_Total_Display               ; Contador de bytes copiados(começa com 5, contagem
decrescente)
                CALL Carrega_Registo                    ; Carrega Registo no Display
                ADD R8, R9                                              ; R8 -> Posição do total
do segundo registo no display
                SUB R10, 1                                              ; Subtrai numero de
registos a carregar no display
                CMP R10, 0                                              ; Já carregou 2 registos?
```

```
                JNE Loop_Visualiza                  ; Verifica se pode mostrar outro registo
                CALLF Botoes_Registos               ; Espera que o botão CHANGE ou CANCEL seja premido
                CALL LimpaPerifericos               ; Limpa Periféricos excepto o PESO
                MOVB R3, [R0]                       ; Lê primeiro byte do próximo registo
                CMP R3, R7                                  ; Verifica se ainda
existem mais registos guardados
                JEQ Sem_Registos                    ; Se for 0, não há mais registos
                JMP Proximo_Visualiza               ; Recomeça para mostrar os próximos 2
registos
Sem_Registos:
                MOV R8, Inicio_Historico            ; R8 -> 1º Endereço do Historicode Registos
                CMP R0, R8                                  ; Verifica se nenhum registo foi
mostrado
                JNE Fim_Visualiza                   ; Se existir pelo menos 1 registo guardado
não mostra mensagem
                MOV R2, MenuRegistoVazio            ; R2 -> Mensagem: Sem registos para visuzlizar
                CALL MostraDisplay                  ; Mostra mensagem no display
                MOV R0, CANCEL                      ; R0 -> CANCEL
                CALLF Verifica_Valor_Botoes         ; Aguarda que seja primido o botão CANCEL
                JMP Salto_Inicio_Menus              ; Volta ao Menu Principal
Fim_Visualiza:
                MOV R5, 1                                   ; R5 = 1
                CMP R5, R10                                 ; Ultimo carregamento de
Registos corresponde a um registo no display
                JEQ LimpaRegisto2                   ; Se sim limpa campos referentes
ao carregamento do segundo registo
Fim_Registos:
                MOV R2, MenuFimRegistos             ; R2 -> Mensagem: não existem mais
Regisos para mostrar
                CALL MostraDisplay                  ; Mostra mensagem no Display
                MOV R0, CANCEL                      ; R0 -> Periférico CANCEL
                CALLF Verifica_Valor_Botoes         ; Aguarda que seja primido o botão CANCEL
                JMP Salto_Inicio_Menus              ; Volta ao Menu Principal

;-----------------------------------------------------
;  Limpa Campos no segundo registo do menu de registos
;-----------------------------------------------------
LimpaRegisto2:
                MOV R4, Display_Nome                ; R4 -> 1º Endereço do nome no display
                ADD R4, R9                                  ; R4 -> 1º Endereço do
nome no segundo registo do menu Registos
                MOV R5, 16                                  ; R5 = 16
                ADD R4, R5                                  ; R4 -> 1º Endereço da
linha referente ao peso no segundo registo do menu Registos
                MOV R10, 0                                  ; R10 = 0
Ciclo_LimpaRegisto2:
                CALLF LimpaLinha                    ; Limpa os 16 bytes da linha correspondente
nos campos do segundo registo
                MOVB R9, [R4]                       ; R9 = [1º endereço da linha
seguinte]
                CMP R9, R10                                 ; [1º endereço da linha
seguinte] = 0?
                JNE Ciclo_LimpaRegisto2             ; Se for diferente de 0 limpa a linha atual
                CALLF Botoes_Registos               ; Espera que o botão CHANGE ou CANCEL
seja premido
                JMP Fim_Registos                    ; Página de registos concluída

;-------------------------------
; Rotina que carrega atributos do Registo no display
; R0 = Endereço do histórico de registos
; R1 = Endereço do display
; R4 = Contador(tamanho do atributo)
; R7 = 0
;-------------------------------
Carrega_Registo:
```

```
                PUSH R2                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R3
                PUSH R4
                PUSH R6
                PUSH R7
                MOV R2, Virgula                                 ; R2 = 2CH(Vírgula ASCII)
                JMP Proximo_Byte                        ; Carrega proximo byte no menu dos registos
Virgula_Registos:
                ADD R1, 1                                               ; Salta posição a
carregar, pois a posição atual está reservada para a virgula no menu
                ADD R0, 1                                               ; R0 -> próximo byte dos
registos guardados a carregar no display
                SUB R4, 1                                               ; Decrementa contador
Proximo_Byte:
                MOVB R6, [R0]                                   ; R6 = [End. registo guardado]
                MOVB [R1], R6                                   ; Carrega [End. registo guardado]
no display
                ADD R1, 1                                               ; Proxima posição no
display
                MOVB R3, [R1]                                   ; R3 = [End. no display a ser
carregado]
                CMP R3, R2                                              ; [End. no display a ser
carregado] = 2CH(Virgula ASCII)?
                JEQ Virgula_Registos                    ; É vírgula não carrega
                ADD R0, 1                                               ; R0 -> próximo byte dos
registos guardados a carregar no display
                SUB R4, 1                                               ; Decrementa contador
                CMP R4, R7                                              ; Contador > 0?
                JGT Proximo_Byte                        ; Se sim ainda não acabou de carregar o
valor, carrega proximo byte
                POP R7                                                  ; Repor os valores
presentes nos registos antes desta chamada
                POP R6
                POP R4
                POP R3
                POP R2
                RET

;--------------------------------------------------------------------------------
; Botoes_Registos  - Rotina para verificar os valores nos botoes no Menu dos Registos
;--------------------------------------------------------------------------------
Botoes_Registos:
                MOV R5, CANCEL                                  ; R5 -> Periferico CANCEL
                MOV R6, CHANGE                                  ; R6 -> Periferico CHANGE
Ciclo_Botoes_Registos:
                CALL Trata_OFF                                  ; Verifica se foi pressionado
ON/OFF para desligar a balança
                MOVB R1, [R5]                                   ; Obtem valor do botão CANCEL
                CMP R1, 1                                               ; Botão = 1?
                JEQ Salto_Inicio_Menus                  ; Se botão igual a 1 volta ao Menu Principal
                MOVB R1, [R6]                                   ; Obtem valor do botão CHANGE
                CMP R1, 1                                               ; Botão = 1?
                JEQ Termina_Botoes_Registos             ; Se botão igual a 1 passa para o proximo
carregamento de registos no display
                JMP Ciclo_Botoes_Registos               ; Volta a ler os botões enquanto nenhum for
pressionado
Termina_Botoes_Registos:
                RETF

;--------------------------------------------------------------------------------
;  Modo Limpa Histórico - exibe o menu que permite ao utilizador limpar o histórico dos registos
;--------------------------------------------------------------------------------
Trata_Limpar:
        MOV R2, MenuReset                                       ; R2 -> MenuReset
```

```
    CALL MostraDisplay                                  ; Mostra o Menu que permite limpar o histórico no
display
    CALL LimpaPerifericos                               ; Atribui o valor zero aos periféricos de entrada
Ciclo_Trata_Limpar:
    CALL Trata_OFF                                      ; Verifica se foi pressionado ON/OFF para
desligar a balança
            MOV R0,CANCEL                                       ; R0 -> Periferico CANCEL
    MOVB R1,[R0]                                        ; Obtem valor do botão CANCEL
    CMP R1,1                                            ; Botão = 1?
    JEQ Salto_Inicio_Menus                              ; Se botão igual a 1 volta ao Menu Principal
Verifica_Botao_Ok:
    MOV R0,OK                                           ; R0 -> Periferico OK
    MOVB R1,[R0]                                        ; Obtem valor do botão OK
    CMP R1,1                                            ; Botão = 1?
    JNE Ciclo_Trata_Limpar                              ; Se for diferente de um volta a ler os botões
    CALL Limpa_Registos                                 ; Caso contrário limpa os registos guardados em 2500H - 2800H
    MOV R2,MenuSucessoApagarHistorico       ;exibe o menu que o historico de registos foi apagado com sucesso
    CALL MostraDisplay                                  ; mostra menu no Display
    CALL LimpaPerifericos                               ; Atribui o valor zero aos periféricos de entrada
    MOV R0,OK                                           ; R0 -> End. OK
    CALLF Verifica_Valor_Botoes             ; quando é pressionado o botao ok o volta para o menu principal
    JMP Inicio_Menus                                    ; Salta para o menu principal

;---------------------------------
; Salta para Verifica_Valor_Botoes
;---------------------------------
Salta_Verifica_Botoes:
            CALLF Verifica_Valor_Botoes             ; Salta para Verifica_Valor_Botoes(salto anterior estava
fora do alcance -255 - 254)
            RET
;-------------------------------------------------------------------------
; Calcula Total - Faz a multiplicação entre o Peso e o preço do produto
;-------------------------------------------------------------------------
        Calcula_Total:
            PUSH R0                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
            PUSH R1
            PUSH R2
            PUSH R3
            PUSH R4
            PUSH R5
            MOV R0, Var_Preço                                   ; R0 -> End. da variável que guarda
o valor do preço do produto selecionado
            MOV R1, [R0]                                ; R1 = [End. da variável que guarda o valor
do preço]
            CMP R1, 0                                               ; R1 = 0?
            JEQ Total_Zero                                  ; Se não tiver sido selecionado
nenhum produto não é calculado o valor total
            MOV R3, Var_PESO                                    ; R3 -> End. da variável que guarda
o valor do peso atual
            MOV R2, [R3]                                ; R2 = Peso centenas de gramas
            MOV R4, R2                                              ; R4 = peso
            ; --- Multiplicação: Preço × Peso ---
            MUL R2, R1                                          ; R2 = peso x preço
            MOV R5, R2                                  ; Cópia do resultado
            DIV R5, R1                                  ; Retira o valor do preço ao resultado da
multiplicação(obtendo o valor do peso)
            CMP R5, R4                                              ; Verifica se bate certo o
valor do peso após a multiplicação
            JNE Erro_Overflow                           ; Se não bate certo, ocorreu
Overflow(resultado obtido não corresponde ao valor correto) para valores >32767
            MOV R3, Var_Total                           ; R3 -> End. da variável que guarda o valor
do total calculado
            MOV [R3], R2                                        ; [End. variável total] = Resultado
da multiplicação
```

```
                CALL Verifica_Arredonda                 ; Verifica se é necessário arredondar o
resultado
Mostra_Total:
                CALL Mostra_Total_Display       ; Mostra o total calculado no Display
                JMP Sem_Overflow                        ; Salta para o tratamento sem Overflow
Erro_Overflow:
                MOV R2, MenuOverflow                    ; R2 -> MenuOverflow
                CALL RotinaERRO                                 ; Chama rotina para tratamento de
erro
                CALL LimpaVariaveis                             ; Limpa as variáveis globais
Sem_Overflow:
                POP R5                                                  ; Repor os valores
presentes nos registos antes desta chamada
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;--------------------------------
; Coloca a zero a variável total
;--------------------------------
Total_Zero:
                MOV R0, Var_Total                       ; R0 -> End. da variável que guarda o valor
do total calculado
                MOV R1, 0                                               ; R1 = 0
                MOV [R0], R1                                    ; [End. variável total] = 0
                JMP Mostra_Total                        ; Mostra total no display

;-------------------------------------------------------------------------------
; Verifica Arredondamento - Rotina para verificar se é para arredondar o número
;-------------------------------------------------------------------------------
Verifica_Arredonda:
                PUSH R0                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                PUSH R5
                MOV R5, Var_Total                       ; R5 -> End. da variável que guarda o valor
do total calculado
                MOV R1, [R5]                                    ; R1 = [End. variável total]
                MOV R0, 10                                              ; Divisor
                MOV R4, 10                                              ; Incrementador
                MOV R3, R1                                              ; R3 = [End. variável
total]
        ; Verifica se é para arredondar o valor
                MOV R2, R3                                              ; R2 = [End. variável
total]
                MOD R2, R0                                              ; R2 = Resto(Total/10)
isola digito mais à direita
                CMP R2, 5                                               ; R2 = 5?
                JLT Nao_Arredonda                       ; Arredonda apenas quando o ultimo digito é
igual ou superior a 5
                DIV R3, R0                                              ; R3 = Parte inteira
restante após ser removido o digito das unidades
                ADD R3, 1                                               ; Sendo para arredondar
incrementa parte inteira em 1 unidade
                ADD R1, R4                                              ; Incrementa as dezenas
do valor total inicial em 1 unidade
                MUL R4, R4                                              ; Incrementador para a
proximo digito do valor total
Ciclo_Arredonda:
```

```
                MOV R2, R3                                              ; R2 = [End. variável
total]
                MOD R2, R0                                              ; R2 = Resto(Total/10)
isola digito mais à direita
                CMP R2, 0                                               ; R2 = 0?
                JNE Nao_Arredonda                                       ; Arredonda apenas quando o
ultimo digito é igual a 0
                DIV R3, R0                                              ; R3 = Parte inteira
restante após ser removido o digito das unidades
                ADD R3, 1                                               ; Sendo para arredondar
incrementa parte inteira em 1 unidade
                ADD R1, R4                                              ; Incrementa as
dezenas/centenas.. do valor total inicial em 1 unidade
                MUL R4, R4                                              ; Incrementador para a
proximo digito do valor total
Nao_Arredonda:
                DIV R1, R0                                              ; Remove digito menos
significativo
                MOV [R5], R1                                            ; Guarda na variável o valor total
                POP R5                                                  ; Repor os valores
presentes nos registos antes desta chamada
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;-------------------------------------------------------------------------------------------
;   Mostra peso no Display - Rotina para colocar o valor do peso em caracteres decimais no display
; R2 = valor do peso a mostrar no display
;-------------------------------------------------------------------------------------------
Mostra_PESO_Display:
                MOV R0, R2                                      ; R0 = PESO em centenas de gramas
        ; Converter para kg com 1 casa decimal: dividir por 10
                MOV R1, 10                                              ; R1 = 10
                CALL Divisao                                    ; Rotina obter quociente e resto
entre PESO/10(R0 e R1 respetivamente)
                MOV R4, R1                                      ; R4 = resto -> parte decimal
        ; Separar parte inteira em dezenas e unidades
                MOV R1, 10                                              ; R1 = 10
                CALL Divisao                                    ; R0 = dezena, R1 = unidade
                MOV R2, R0                                      ; R2 = dezena
                MOV R3, R1                                      ; R3 = unidade
        ; Converter para ASCII
                MOV R6, Decimal_Ascii                   ; 30h
                ADD R2, R6                                      ; dezena
                ADD R3, R6                                      ; unidade
                ADD R4, R6                                      ; decimal
        ; Enviar para o display nas posições certas
                MOV R5, Display_Peso                            ; R5 -> 1º Endereço para escrever o peso no
display
                MOVB [R5], R2                                   ; dezena
                ADD R5, 1                                               ; Próximo endereço(byte)
                MOVB [R5], R3                                   ; unidade
                ADD R5, 2                                               ; Salta 2 Bytes de
endereço, pois o endereço seguinte já contém o caracter ','
                MOVB [R5], R4                                   ; decimal (1 casa)
                RETF

;-------------------------------------------------------------------------------
;   Mostra produto no Display - Rotina para colocar o nome e o preço do produto no display
;-------------------------------------------------------------------------------
Mostra_produto_Display:
                PUSH R0                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
```

```
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                PUSH R5
                PUSH R6
                MOV R0, Var_Codigo                      ; Endereço onde está guardado o código
                MOVB R1, [R0]           ; R1 = Código selecionado
                MOV R2, Qtdd_Produtos                   ; Número total de produtos
                MOV R3, Inicio_Tabela   ; R3 = Início da tabela de produtos
Proximo_Produto:
                CMP R2, 0                                                       ; contador = 0?
                JEQ Produto_Nao_Encontrado      ; Para quando não existem produtos na tabela
                MOV R4, [R3]            ; Lê código do produto atual
                CMP R1, R4                                                      ; código inserido =
[indice]?
                JEQ Produto_Encontrado                  ; Se sim encontrou
                MOV R5, Tam_Prod_Tabela                 ; R5 = 14
                ADD R3, R5              ; Avança para o próximo produto (assumindo 14 bytes por
produto)
                SUB R2, 1                                                       ; Contador = Contador - 1
                JMP Proximo_Produto                             ; Verifica se o codigo do proximo
produto na tabela coincide
Produto_Encontrado:
                ADD R3, 2               ; Avança para o preço (após 2 bytes de código)
                MOV R7, [R3]                                    ; R7 = preço do produto encontrado
                MOV R8, Var_Preço                               ; R8 -> End. da variável que guarda
o preço
                MOV [R8], R7                                    ; [End. variável preço] = preço do
produto selecionado
                CALL Mostra_Preço_Display       ; Carrega preço no display
                ADD R3, 2                                                       ; R3 agora está na
posição do nome (após preço de 2 bytes)
                MOV R4, Display_Nome                    ; R4 -> 1º Endereço do nome no display
                MOV R5, Tam_Nome_Display        ; R5 = tamanho do nome no display
                MOV R6, 0                                                       ; R6 = 0
Mostra_Nome:
                MOVB R0, [R3]                                   ; R0 = caracter do nome
                CMP R6, R5                                              ; Verifica se chegou ao
fim do nome
                JEQ Fim_Produto                                 ; Se carregou o nome todo termina
a rotina
                MOVB [R4], R0                                   ; Carrega caracter no display
                ADD R3, 1                                               ; Proximo endereço na
tabela
                ADD R4, 1                                               ; Proximo endereço no
display
                ADD R6, 1                                               ; Incrementa contador
                JMP Mostra_Nome
Fim_Produto:
                MOV R5, 6                                                       ; Nº de endereços
restantes na linha do nome para limpar
                CALL Chama_LimpaLinha                   ; Limpa o restante da linha a seguir ao nome
estar escrito
                POP R6                                                          ; Repor os valores
presentes nos registos antes desta chamada
                POP R5
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;--------------------------------------------------------------------------------------------
; Chama a rotina que coloca a linha no display em branco, a partir da Rotina Mostra_produto_Display
```

```
;-------------------------------------------------------------------------------------------
Chama_LimpaLinha:
                CALLF LimpaLinha                        ; Salta para a rotina LimpaLinha(salto anterior
estava fora do alcance -255 - 254)
                RET

;-----------------------------------------------------------------
;   Mostra Total no Display - Rotina para colocar Total no display
;-----------------------------------------------------------------
Mostra_Total_Display:
                PUSH R0                                         ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                PUSH R5
                MOV R0, 10                                      ; R0 = 10 Divisor
                MOV R1, Var_Total                       ; R1 -> End. Variavel Total
                MOV R1, [R1]                                    ; R1 = [End. Variavel Total]
                MOV R2, Display_Total                   ; R2 fica com a base do total no display
                ADD R2, Nr_Car_Total                    ; posição do carater a preencher, do menos
significativo para o mais significativo
                MOV R3, 0                                       ; R3 tem o nr de
carateres já preenchido
proximocarater:
                MOV R4, R1                                      ; R4 fica com uma cópia
do valor lido
                MOD R4, R0                                      ; R4 = resto da divisão
inteira por 10
                DIV R1, R0                                      ; Atualiza R1 = quociente
da divisão inteira por 10
                MOV R5, Decimal_Ascii                   ; R5 = 30H

                ADD R5, R4                                      ; Calcula Caracter =
Hexadecimal + 30H
                MOVB [R2], R5                                   ; escreve o carater no display
                SUB R2, 1                                       ; proxima posição do
display a preencher
                CALL VerificaVirgula                    ; Verifica se posição atual contém uma vírgula
                ADD R3, 1                                       ; incrementa o nr de
carateres preenchidos
                CMP R3, Nr_Car_Total                    ; Valor a preencher chegou ao fim?
                JNE proximocarater                      ; Se não preencheu o valor total salta para o
proximo digito a carregar
fimrotina:
                POP R5                                          ; Repor os valores
presentes nos registos antes desta chamada
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;-------------------------------------------------------------------------------------
; Verifica Vírgula - Rotina para verificar se a posição atual no display contém uma vírgula
; R2 = Endereço do display
;-------------------------------------------------------------------------------------
VerificaVirgula:
                PUSH R0                                         ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R1
                MOVB R0, [R2]                                   ; R0 = [End. Display]
                MOV R1, Virgula                                 ; R1 = 2CH(Virgula)
                CMP R0, R1                                      ; R0 = 2CH?
```

```
                JNE Sem_Virgula                                         ; Se não for igual, não é virgula
                SUB R2, 1                                               ; Passa para o próximo
endereço
Sem_Virgula:
                POP R1                                                  ; Repor os valores
presentes nos registos antes desta chamada
                POP R0
                RET

;-----------------------------------
;Coloca a linha no display em branco
; R4 = Endereço do display
; R5 = 16(bytes a limpar na linha)
;-----------------------------------
LimpaLinha:
                PUSH R5                                                 ; R5 = Nº de endereços a limpar numa linha
                MOV R0, CaracterVazio       ; Carácter vazio (ex: espaço)
Ciclo_LimpaLinha:
                MOVB [R4], R0           ; Escreve carácter vazio no display
                ADD R4, 1               ; Avança para o próximo endereço
                SUB R5, 1                                               ; Decrementa contador de endereços a limpar
                CMP R5, 0                                               ; Contador = 0?
                JGT Ciclo_LimpaLinha        ; Continua até R5 == 0
                POP R5
                RETF

;---------------------------------------------------------------------
;Produto não encontrado na tabela - limpa a variável global do código
;---------------------------------------------------------------------
Produto_Nao_Encontrado:
                MOV R0, Var_Codigo                      ; R0 -> End. Variável código
                MOV R1, 0                                       ; R1 = 0
                MOV [R0], R1                            ; [End. Variável código] = 0
                JMP Fim_Produto

;---------------------------------------------------------------------
; Mostra_PRECO_Display - Mostra preço no display no formato XX,YY€
; R3 = Endereço da tabela de produtos(preço do produto)
;---------------------------------------------------------------------
Mostra_Preço_Display:
                PUSH R0                                                 ; Guarda na stack os valores que constam
nos registos a serem usados na rotina
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                PUSH R5
                PUSH R6
                PUSH R7
                MOV R7, Display_Preço           ; Endereço base para mostrar o preço
                MOV R0, R3              ; R3 tem o endereço do preço na tabela
                MOV R1, [R0]            ; R1 = preço em cêntimos (ex: 1299)
                ; Separar euros e cêntimos usando Divisão
                MOV R0, R1              ; R0 = preço
                MOV R1, 100             ; divisor
                CALL Divisao            ; R0 = euros, R1 = cêntimos
                MOV R2, R0              ; R2 = euros
                MOV R3, R1              ; R3 = cêntimos
                ; --- Parte inteira (euros) ---
                MOV R0, R2              ; euros
                MOV R1, 10                                              ; R1 = 10
                CALL Divisao            ; R0 = dezenas, R1 = unidades
                MOV R4, R0              ; R4 = dezena dos euros
                MOV R5, R1              ; R5 = unidade dos euros
                ; Converter para ASCII
                MOV R6, Decimal_Ascii           ; R6 = 30H
```

```
                ADD R4, R6                                      ; Soma 30H para obter o caracter decimal
                ADD R5, R6                                      ; Soma 30H para obter o caracter decimal
                ; Escrever euros no display
                MOVB [R7], R4                           ; Escreve dezena no display
                ADD R7, 1                                       ; Avança para a próxima posição no display
                MOVB [R7], R5                           ; Escreve unidade no display
                ; --- Parte decimal (cêntimos) ---
                MOV R0, R3              ; cêntimos
                MOV R1, 10                                      ; R1 = 10
                CALL Divisao            ; R0 = decimas, R1 = centesimas
                MOV R4, R0                                      ; R4 = décimas
                MOV R5, R1                                      ; R5 = centésimas
                ; Converter para ASCII
                ADD R4, R6                                      ; Soma 30H para obter o caracter decimal
                ADD R5, R6                                      ; Soma 30H para obter o caracter decimal
                ; Escrever cêntimos no display (após vírgula)
                ADD R7, 2                               ; Salta para depois da vírgula (posição 3)
                MOVB [R7], R4                           ; Escreve decimas no display
                ADD R7, 1                                       ; Avança para a próxima posição no display
                MOVB [R7], R5                           ; Escreve centesima no display
                POP R7                                          ; Repor os valores presentes nos registos
antes desta chamada
                POP R6
                POP R5
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;-------------------------------------------------------------------------------------------------------
;   Divisão - Rotina com algoritmo de divisão para obter o quociente e o resto na divisão entre dois números
; R0 = valor a dividir
; R1 = divisor(10)
;-------------------------------------------------------------------------------------------------------
        Divisao:
                PUSH R2                                                 ; Guarda valor em R2 na
Stack
                MOV R2, 0                                       ; Quociente = 0
        DivideLoop:
                CMP R0, R1
                JLT FimDivisao                                  ; Se dividendo(R0) < divisor(R1), acabou a
divisão
                SUB R0, R1                                              ; Caso contrário
dividendo(R0) = dividendo(R0) - divisor(R1)
                ADD R2, 1                                               ; R2(Quociente) = R2 + 1
                JMP DivideLoop                                  ; Continua até que não seja
possível mais nenhuma divisão inteira
        FimDivisao:
                MOV R1, R0                                      ; R1 agora é o resto
                MOV R0, R2                                      ; R0 é o quociente
                POP R2                                                  ; Repõe valor anterior em
R2
                RET

;-------------------------------------------------------------------------
;   Rotina Erro - Rotina para indicar ao utilizador que escolheu uma opção errada
;-------------------------------------------------------------------------
RotinaERRO:
                CALL MostraDisplay                                      ; Mostra Mensagem de erro no
Display
                CALL LimpaPerifericos                           ; Atribui o valor zero aos periféricos de
entrada
                MOV R0, OK                                              ; R0 -> End. OK
                CALLF Verifica_Valor_Botoes             ; Aguarda que seja pressionado o botão OK
```

```
                    RET

;--------------------------------------------------------------------
; Mostra Display - Rotina para mostrar no Display o que está na memória
; R2 = Endereço do menu a mostrar
;--------------------------------------------------------------------
MostraDisplay:
                    PUSH R0                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                    PUSH R1
                    PUSH R2
                    PUSH R3
                    MOV R0, Display                                         ; R0 -> primeiro endereço(par) do
display
                    MOV R1, Display_end                                     ; R1 -> último endereço(par) do
display
CicloCarrega:
                    MOVB R3, [R2]                                           ; R3 = Informação(j)
                    MOVB [R0], R3                                           ; Display(i) = Informação(j)
                    ADD R2, 1                                               ; j = j + 1
                    ADD R0, 1                                               ; i = i + 1
                    CMP R0, R1                                              ; i <= Fim_Display?
                    JLE CicloCarrega                                ; Carrega informação no Display até
preencher o espaço reservado para o mesmo
                    POP R3                                                  ; Repor os valores
presentes nos registos antes desta chamada
                    POP R2
                    POP R1
                    POP R0
                    RET

;-----------------------------------------------------
;   Aviso - Histórico de registos encontra-se cheio
;-----------------------------------------------------
Mostra_Display_Historico_Cheio:
                    MOV R2, MenuLimiteRegistos          ; R2 -> MenuLimiteRegistos
                    CALL RotinaERRO                                 ; Chama rotina para informar ao
utilizador acerca desta irregularidade
                    JMP Trata_Balança                               ; Volta para o menu Balança

;--------------------------------------------------------------------
; GuardaRegisto - Rotina para adicionar um registo à tabela de registos
;--------------------------------------------------------------------
GuardaRegisto:
                    PUSH R0                                                 ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
      PUSH R1
      PUSH R2
      PUSH R3
                    PUSH R4
                    PUSH R5
                    PUSH R6
                    MOV R4, 0                                               ; Contador = 0
                    MOV R6, Virgula                                 ; R6 = 2CH(Virgula)
      ; Verifica se há espaço no histórico
      MOV R0, Ptr_Historico                         ; R0 -> End. que contém proximo endereço disponivel para
guardar um registo
      MOV R1, [R0]                  ; R1 = endereço atual disponível para escrita
      MOV R2, Fim_Historico                         ; R2 -> Ultimo endereço reservado para guardar registos
                    MOV R3, Tam_Registos                                    ; tamanho de cada registo
      ADD R1, R3                                    ; Verifica se é possivel realizar mais um registo

      CMP R1, R2
      JGT Mostra_Display_Historico_Cheio      ; sem espaço, salta para display aviso
      MOV R0, Ptr_Historico                         ; R0 -> End. que contém proximo endereço disponivel para
guardar um registo
```

```
    MOV R1, [R0]                    ; R1 = endereço onde começa o registo
    ; Guarda nome do produto
                MOV R5, Tam_Nome_Display            ; R5 = 10(Tamanho do nome no display)
    MOV R2, Display_Nome            ; R2 -> 1º End. reservado para o nome no display
                CALLF Ciclo_Reg                     ; Guarda nome do produto
    ; Guarda peso
                ADD R5, Tam_Peso_Display            ; R5 = 3(Tamanho do peso no display)
    MOV R2, Display_Peso            ; R2 -> 1º End. reservado para o peso no display
                CALLF Ciclo_Reg                     ; Guarda peso do registo
    ; Guarda Total
    ADD R5, Tam_Total_Display       ; R5 = 5(Tamanho do total no display)
    MOV R2, Display_Total           ; R2 -> 1º End. reservado para o total no display
                CALLF Ciclo_Reg                     ; Guarda total do registo
    ; Atualiza ponteiro para próximo registo
                MOV [R0], R1        ; Atualiza Ptr_Historico para novo endereço
                MOV R2, MenuSucessoRegisto          ; R2 -> MenuSucessoRegisto
                CALL MostraDisplay                  ; Mostra mensagem de registo bem
sucedido no display
                CALL LimpaPerifericos               ; Limpa os periféricos excepto o Peso
                MOV R0, OK                          ; R0 -> OK
                CALLF Verifica_Valor_Botoes         ; Aguarda que seja pressionado o botão OK
                POP R6                              ; Repor os valores
presentes nos registos antes desta chamada
                POP R5
                POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

;-------------------------------------------------------------------------
; Ciclo Regista - Rotina para registar na memória os atributos dos registos
; R1 = Endereço do Histórico de registos
; R2 = Endereço do display com conteúdo do registo a ser guardado
; R4 = contador
; R5 = tamanho do conteúdo a ser guardado
;-------------------------------------------------------------------------
Ciclo_Reg:
    MOVB R3, [R2]                   ; R3 = [end. display]
                CMP R3, R6                          ; R3 = 2CH(virgula)?
                JEQ Proximo_Registo                 ; Encontrou vírgula, não regista a
virgula
    MOVB [R1], R3                   ; Guarda valor na posição de memória
associada
    ADD R1, 1                       ; Proximo endereço para guardar o registo
                ADD R4, 1                           ; Incrementa Contador
Proximo_Registo:
                ADD R2, 1                           ; Proxima posição no
display
                CMP R4, R5                          ; Contador < Tamanho do
atributo a guardar
                JLT Ciclo_Reg                       ; Se for menor continua para o
próximo byte
                RETF

;-----------------------------------------------------------------
; Limpa_Registos - Rotina para limpar os registos guardados na memória
;-----------------------------------------------------------------
Limpa_Registos:
                PUSH R0                             ; Guarda na stack os
valores que constam nos registos a serem usados na rotina
                PUSH R1
                PUSH R2
                PUSH R3
                MOV R0, Inicio_Historico    ; Endereço inicial dos registos (2500H)
```

```
                MOV R1, Fim_Historico        ; Endereço final dos registos (2800H)
                MOV R3, 0                                                    ; R3 = 0
LimpaLoop:
                MOV R2, [R0]                 ; Lê a palavra atual da memória
                CMP R2, 0                                                    ; Verifica se a posição
atual já se encontra nula
                JEQ LimpaFim            ; Já está a 0, parar limpeza
                MOV [R0], R3            ; Escreve 0 nos endereços atuais
                ADD R0, 2               ; Passa ao próximo edereço(par)
                CMP R0, R1                                                   ; Verifica se chegou ao
fim da gama reservada para os registos
                JLT LimpaLoop           ; Continua se ainda dentro do intervalo
LimpaFim:
                MOV R1, Ptr_Historico                        ; R1 -> End. que guarda proxio endereço
disponivel para guardar um registo
                MOV R0, Inicio_Historico    ; Endereço inicial dos registos (2500H)
                MOV [R1], R0                                             ; Coloca o ponteiro do histórico de
registos no Inicio
                POP R3                                                       ; Repor os valores
presentes nos registos antes desta chamada
                POP R2
                POP R1
                POP R0
                RET

;--------------------------------------------------------------------------------
; Limpa Perifericos - Rotina para limpar os periféricos de entrada, excepto o PESO
;--------------------------------------------------------------------------------

LimpaPerifericos:
                PUSH R0                                                      ; Repor os valores
presentes nos registos antes desta chamada
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                PUSH R6
                MOV R0, ON_OFF                                   ; R0 -> End. ON_OFF
                MOV R1, SEL_NR_MENU                              ; R1 -> End. SEL_NR_MENU
                MOV R2, OK                                               ; R2 -> End. OK
                MOV R3, CHANGE                                   ; R3 -> End. CHANGE
                MOV R4, CANCEL                                   ; R4 -> End. CANCEL
                MOV R6, 0                                                ; R6 = 0, Pois o
processador Não realiza MOVB [], Constante!
                MOVB [R0], R6                                    ; Botão ON_OFF = 0
                MOVB [R1], R6                                    ; SEL_NR_MENU = 0
                MOVB [R2], R6                                    ; Botão OK = 0
                MOVB [R3], R6                                    ; Botão CHANGE = 0
                MOVB [R4], R6                                    ; Botão CANCEL = 0
                POP R6                                                   ; Repor os valores
presentes nos registos antes desta chamada
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;---------------------------------------------------------------
; Limpa Variáveis - Rotina para limpar os endereços das variáveis
;---------------------------------------------------------------
LimpaVariaveis:
                MOV R0, 0                                                ; R0 = 0
                MOV R1, Var_PESO                                 ; R1 -> End. Variavel peso
                MOV [R1], R0                                     ; [End. Variavel peso] = 0
                MOV R1, Var_Codigo                               ; R1 -> End. Variavel codigo
```

```
                MOV [R1], R0                                    ; [End. Variavel codigo] = 0
                MOV R1, Var_Preço                               ; R1 -> End. Variavel preço
                MOV [R1], R0                                    ; [End. Variavel preço] = 0
                MOV R1, Var_Total                               ; R1 -> End. Variavel total
                MOV [R1], R0                                    ; [End. Variavel total] = 0
                RETF

;-------------------------------------------------------------------------
;  Limpa Display - Rotina que limpa o display (Modo OFF - Balança Desligada)
;-------------------------------------------------------------------------
LimpaDisplay:
                MOV R0, Display                                 ; R0 -> primeiro endereço(par) do
display
                MOV R1, Display_end                             ; R1 -> último endereço(par) do
display
                MOV R2, CaracterVazio                           ; R2 = 20H(caracter vazio em hexadecimal)
CicloLimpa:
                MOVB [R0], R2                                   ; Dsiplay(i) = Carater Vazio
                ADD R0, 1                                               ; i = i + 1 (endereço do
próximo byte)
                CMP R0, R1                                              ; i = Fim_Display?
                JLE CicloLimpa                                  ; Continua a "limpar" o display até
ao último byte associado
                RETF

;-------------------------------------------------------------------------
;  Verifica_Valor_Botoes  - Rotina para verificar os valores nos botoes
; R0 = Endereço do botão a ser verificado
;-------------------------------------------------------------------------
Verifica_Valor_Botoes:
                CALL Trata_OFF                                  ; Verifica se foi pressionado
ON/OFF para desligar a balança
                MOVB R1, [R0]                                   ; Obtem valor do botão
                CMP R1, 1                                               ; Botão = 1?
                JNE Verifica_Valor_Botoes               ; Se botão diferente de 1, volta a ler o valor
        RETF
```