UNIVERSIDADE da MADEIRA

Arquitetura de Computadores

2º Projeto

Discentes:

Francisco Miguel Gouveia Serrão 2040916

Ricardo Lucas Jardim 2040416

Docentes:

Dionísio Barros

Élvio Jesus

Nuno Ferreira

Sofia Inácio

INDICE

1. INTRODUCAO

Com este trabalho pretende-se realizar a implementação de um programa que permita controlar os pedidos de uma pizzaria *online* que faz entregas, em linguagem *Assembly*. Deve existir um *display* que simule a página web onde são realizados os pedidos dos clientes. Este *display* tem uma dimensão de sete linhas de dezasseis bytes e o processador utilizado será o **PEPE**.

Nesta simulação da página web os clientes têm que estar registados para poderem efetuar pedido, este registo é composto por ***Username*** e ***Password***. Inicialmente existe um menu que permite fazer *login* ou registo no caso de ser um novo cliente.

Após ser realizado o *login*, o utilizador tem a possibilidade de realizar um pedido ou fazer *logout* da sua conta.

No que toca ao pedido em si, o utilizador tem à sua disposição cinco pizzas diferentes com dois tamanhos, pequeno e grande com um custo de cinco e oito euros respetivamente. O cliente tem desconto caso o seu histórico de compras atinja os 50 euros e este desconto pode ocorrer de duas maneiras diferentes. Caso a compra atual do utilizador ultrapasse os 50 euros, a pizza de menor valor terá um desconto de 50%, caso não atinja este desconto é aplicado na pizza de menor valor da encomenda seguinte.

Após a finalização de pedido é mostrado no *display* a informação de descontos e o valor total a pagar.

2. IMPLEMENTAÇÃO EM ASSEMBLY

2.1. INICIALIZAÇÃO DE BOTÕES E DISPLAY

Inicialmente foram implementados em endereços da memória os botões **ON_OFF, OK** e **NR_SELL**. Também foram implementados dois periféricos de entrada que são utilizados para os dados do utilizador, nomeadamente **USERNAME** e **PASSWORD**.

Relativamente ao *display*, foram reservados na memória endereços de forma a que este tivesse 7 linhas de 16 caracteres.

Cada vez que é feita uma mudança de menu são utilizadas as etiquetas **LIMPA_PERIFERICOS**, **MOSTRA_DISPLAY** e **LIMPA_DISPLAY** de forma a ser alterado o *display* para esse menu seguinte.

2.2. MENU_INICIO

No inicio de todo o programa, resumidamente, em primeiro lugar verifica o estado geral do *site* (se esta desligado ou ligado) através do botão **ON_OFF** de modo a verificar o estado atual do site e se necessário atualizá-lo (transitar de desligado para ligado ou vice-versa).

Após esta verificação e caso o estado do botão **ON_OFF** esteja a 1 (indicando que o site está ligado), é realizada a verificação da opção escolhida no menu principal através do botão NR_SELL para o menu seguinte, sendo que após esta verificação de escolha, são utilizadas as funções LIMPA_PERIFÉRICOS, LIMPA_DISPLAY e LIMPA_USER_PASS para limpar o display atual e transitar para o menu seguinte.

2.3. MENU DE LOGIN

No menu de login, foram reservados espaços na memória (00E0H e 0100H) de forma a servirem de periféricos de entrada **USERNAME** e **PASSWORD** referidos anteriormente. Após isto, o programa "espera" até que o estado do botão **OK** seja alterado para 1 pelo utilizador para copiar os periféricos de entrada para o *display* e em seguida através das etiquetas OPCAO_LOGIN, ECRA_USERNAME, PROXIMOCARACTER_ECRA, ECRA_PASS, LOGIN_USER_2,**LOGIN_USER, PROXIMO_USER** e **LOGIN_PASS** e das rotinas **OPCAO_LOGIN_CICLO** e **OPCAO_PASS_CICLO, CICLO_ECRA_PASS .**

Caso o **USERNAME** e **PASSWORD** estejam corretos, o programa salta para a etiqueta de **LOGIN_SUCESSO**, caso contrário o programa salta para a etiqueta **FALHA_LOGIN** que entrará no menu de ERRO.

2.4. MENU DE REGISTO

No menu de registo, da mesma forma, o utilizador insere através dos periféricos de entrada um **USERNAME** e **PASSWORD** à sua escolha. O programa espera que o utilizador coloque o botão **OK** a 1 e o programa através das etiquetas **VERIFICA_USER** que verifica se o **USERNAME** tem pelo menos 1 caracter, **VERIFICA_PASS** que verifica se a **PASSWORD** tem entre 3 e 8 caracteres, **VERIFICA_USERNAME_IGUAL** que verifica se já existe um **USERNAME** igual e **VERIFICA_PROXIMO_USER** que verifica o utilizador seguinte e das rotinas **CICLO_VERIFICA_USER**, **CICLO_VERIFICA** e **CICLO_VERIFICA_USERNAME_IGUAL.**

Caso não encontre nenhum **USERNAME** igual nem faltar caracteres na **PASSWORD** ou **USERNAME,** é utilizada a etiqueta **CREATE** e a rotina **CICLO_CREATE** para guardar os dados desse utilizador na memória.

Caso algum dos requisitos não seja cumprido, o *display* é limpo assim como os periféricos e é mostrado no *display* o menu de erro.

2.5. MENU INICIAL

No menu inicial, é mostrado no *display* 3 opções nomeadamente**, ENCOMENDAR**, **VALOR GASTO** e **LOGOUT**. O programa lê o valor do botão **NR_SELL** de forma a verificar qual a opção que foi escolhida pelo utilizador. Caso o utilizador escolha um valor para o botão **NR_SELL** maior do que 3, o programa executa um salto para o menu de erro, caso contrário este realiza um salto para a etiqueta correspondente ao menu escolhido pelo utilizador.

2.6. MENU DE ESCOLHA

No menu escolha são atribuídos ao botão **NR_SELL** os valores de 1 a 6 para que quando o utilizador atribuir um dos desses valores ao botão **NR_SELL**, o programa salte para a etiqueta com a pizza respetiva limpando os periféricos e *display* antes de apresentar o menu seguinte no *display*.

Em seguida o programa aguarda até que o utilizador coloque o botão OK a 1 através da função ESPERA_OK e em seguida permite que seja alterado

2.7. MENU DA PIZZA

Existem 6 menus de pizza com 2 opções cada um, grande e pequeno. É apresentado à frente da opção o preço de cada uma e o utilizador mais uma vez através do valor colocado no botão **NR_SELL** escolhe a opção pretendida e o programa guarda em memória o valor monetário da pizza e se foi grande ou pequena, sendo limpo o display e periféricos e mostrado o menu seguinte, **MENU FINALIZAR.**

2.8. MENU FINALIZAR

No menu finalizar, são apresentadas no display o desconto e o valor final. O programa soma os valores das pizzas e verifica se o utilizador já ultrapassou os 50€. Caso o utilizador tenha ultrapassado os 50€ o programa verifica se a compra atual é superior a 50€. Se o valor da compra atual não ultrapassar os 50€ o programa grava em memória o valor do historial de gastos do utilizador para realizar o desconto na encomenda seguinte.

2.9. MENUS DE ERRO

Nos menus de erros, **MENU_ERRO**, **MENU_EXISTENTE**, **MENU_FALTA** e **PASS_INCORRETA** são compostos por uma mensagem que é mostrada no *display* com o erro que ocorreu no processo seja de registo, *login* ou outro e o botão **OK** para dar a possibilidade ao utilizador para voltar ao menu principal. Caso o utilizador queira voltar ao menu principal deve alterar o valor do botão **OK** para 1.

3. ANÁLISE DE RESULTADOS

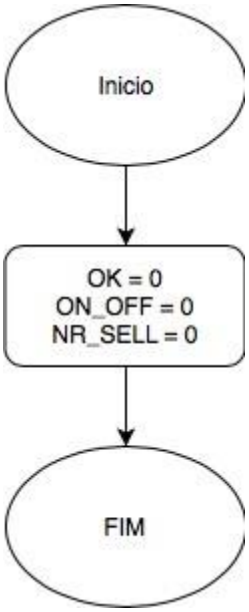
No que toca aos resultados da nossa simulação chegamos a resultados muito satisfatórios visto que o programa geral do *site* está em funcionamento faltando apenas a implementação dos descontos nas pizzas caso o valor gasto dos clientes seja superior a 50€.

4. CONCLUSÃO

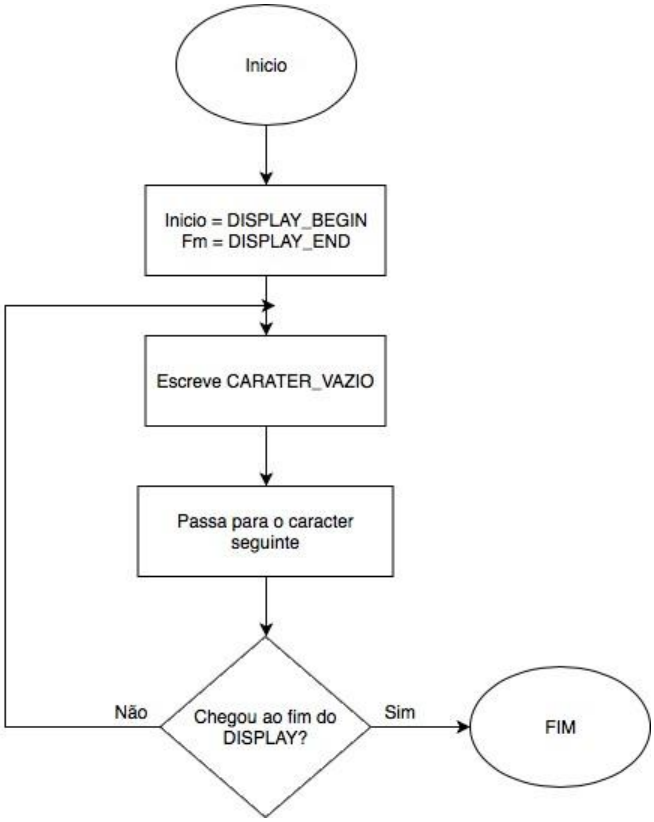
Em suma, o objetivo geral deste projeto foi cumprido, visto que foram implementados todos os requisitos da pizzaria *online* propostos pelos docentes, à exceção dos saldos das pizzas. Tivemos algumas dificuldades nesta implementação e

devido a este percalço não conseguimos realizar essa implementação dentro do prazo de entrada.

No entanto, as restantes componentes requisitadas pelos docentes, como a realização de registo de novos utilizadores, *login* dos mesmos, realização automática de descontos por parte do processador e guardar em memória os dados dos utilizados, entre outros.



Com a realização deste projeto foi-nos permitido aprofundar os nossos conhecimentos sobre, não só do **PEPE** em si, como também a implementação em linguagem *Assembly* das instruções do **PEPE** de forma a serem realizadas todas as tarefas necessárias por parte do **PEPE** na pizzaria.



5. ANEXOS A

Figura 1 - LIMPA_PERIFÉRICOS

Figura 2 - LIMPA_DISPLAY

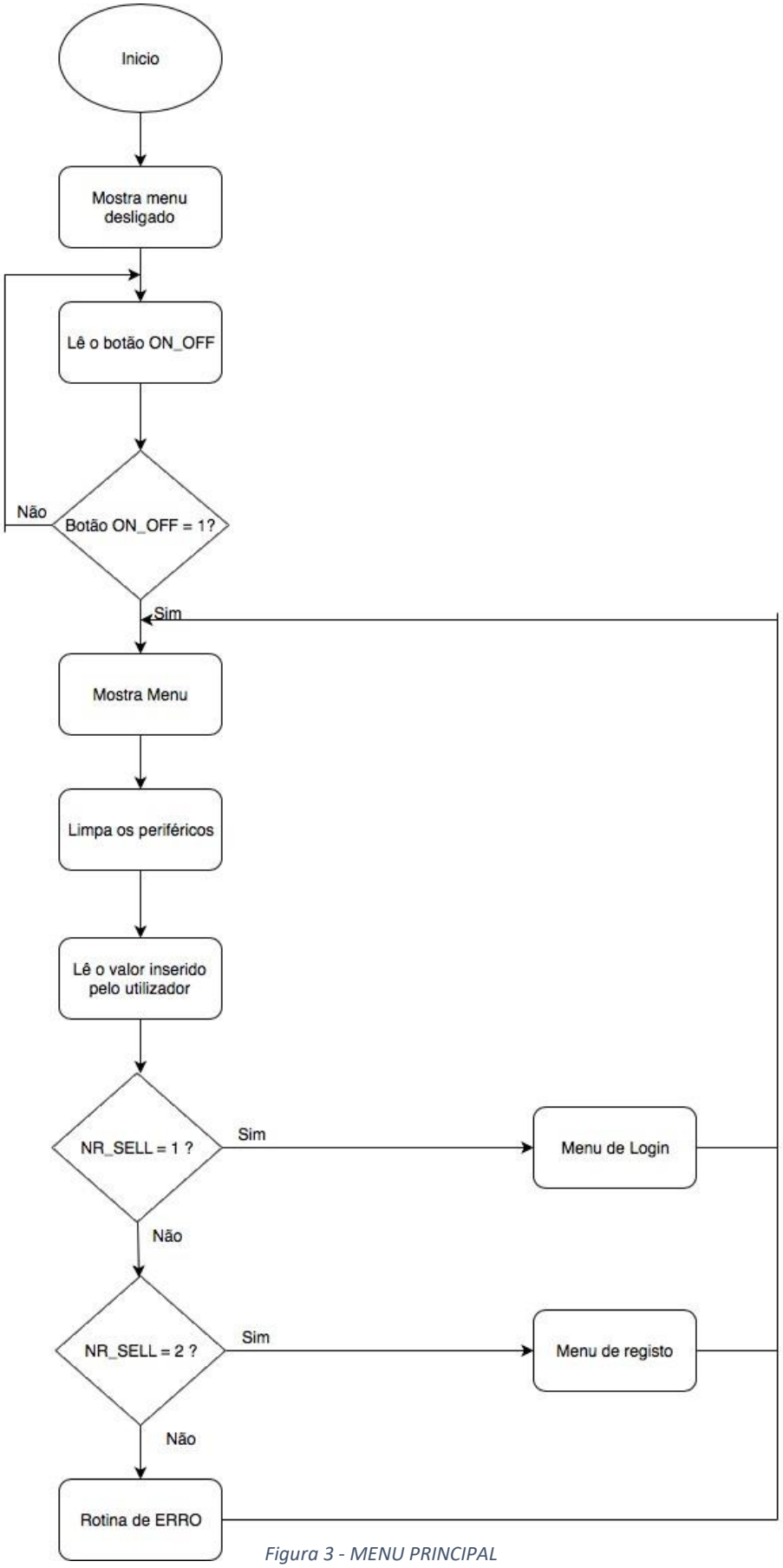


Figura 3 - MENU PRINCIPAL

NOTA: O menu principal tem o mesmo fluxograma que todos os outros sendo que a única diferença entre eles está nos valores do botão **NR_SELL** e nomes do menu seguinte caso seja essa a opção dada pelo valor deste mesmo botão

Figura 5 - Rotina de Erro

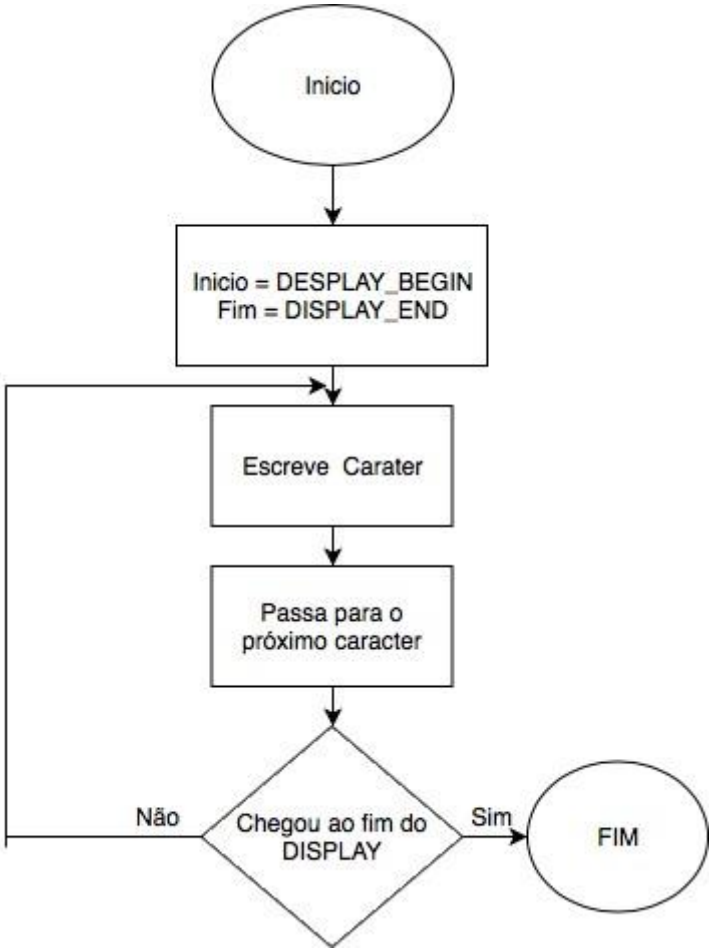


Figura 4 - MOSTRA_DISPLAY

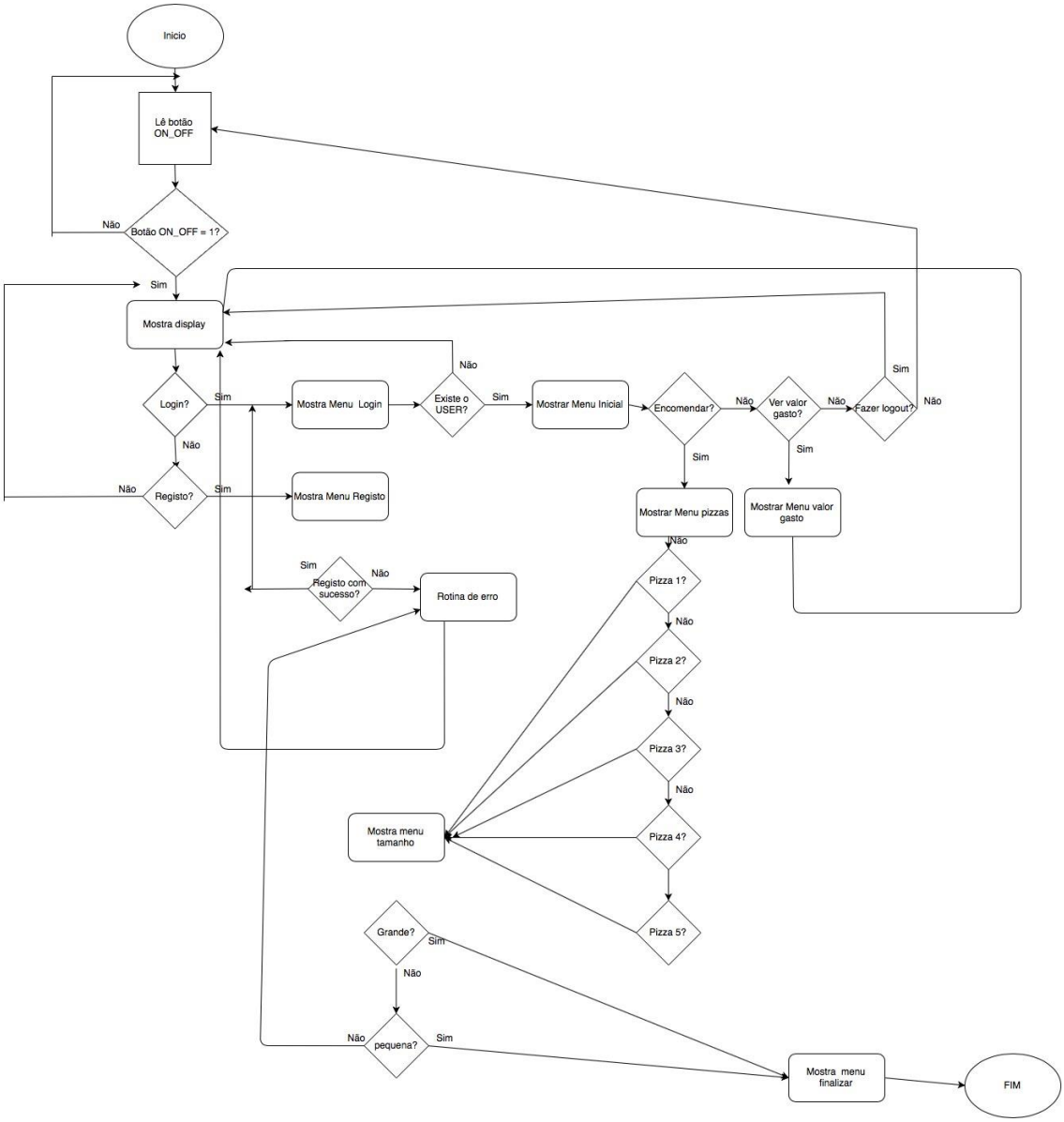


Ilustração 1 - fluxograma geral

ANEXOS B

```
;Display

DISPLAY_BEGIN EQU 00C0H ; Endereço correspondente onde o Display inicializa
DISPLAY_END EQU  012FH ; Endereço correspondente onde o Display termina
CARACTER_VAZIO EQU 20H ; CARACTER VAZIO

BASE_DADOS EQU 9070H   ;Endereço onde se inicializa a dase de dados
BASE_DADOS_FIM EQU 9600H ;Endereço onde a base de dados termina
NR_UTILIZADORES EQU 4000H ; nº de utilizador ja registados na base de dados


POSICAO_ID EQU 9000H ; Endereço onde se encontrará a posicao do ID que fez login
BASE EQU 00E6H ;Endereço onde ficara situado o valor gasto na pizzaria
BASE2 EQU 0124H ;Endereço onde ficara situado o valor a pagar

GUARDAR_SALDO EQU 8900H ; Endereço onde se encontrará em memoria o valor a
pagar

PLACE 4000H
String 0,3

PLACE 9070H       ;!SALDO
STRING "AAAAAAAA",0,50,0,0,0,0,0,0 ; USERNAME
STRING "66666666",0,0,0,0,0,0,0,0 ; PASS
STRING "BBBBBBBB",0,30,0,0,0,0,0,0
STRING "33333333",0,0,0,0,0,0,0,0
STRING "CCCCCCCC",0,20,0,0,0,0,0,0
STRING "55555555",0,0,0,0,0,0,0,0


; Periféricos e botoes

ON_OFF EQU 040H
NR_SELL EQU 050H
OK EQU 060H


USERNAME EQU 0020H

PASSWORD EQU 0030H
```

```
;Constantes para os menus
LOGIN EQU 1
NOVO_REGISTO EQU 2


ENCOMENDAR EQU 1
VALOR_GASTO EQU 2
LOG_OUT EQU 3

PIZZA_AMALFI EQU 1
PIZZA_CALZONE EQU 2
PIZZA_ATUM EQU 3
PIZZA_CAMARAO EQU 4
PIZZA_VEGAN EQU 5
FINALIZAR EQU 6

PEQUENO EQU 2
GRANDE EQU 1
VOLTAR EQU 3

VALOR_PEQUENO EQU 5
VALOR_GRANDE EQU 8

VOLTAR_ATRAS EQU 1

STACKPOINTER EQU 3600H ; inico da StackPointer

; Endereços de memoria onde estaram situadas os menus criados em Strings
PLACE 8000H
MENU_INICIO:
 String"   BEM VINDO    "
 String"1 - LOGIN      "
 String"2 - NOVO REGISTO"
 String"               "
 String"               "
 String"               "
 String"               "

PLACE 8080H
MENU_LOGIN:
   String"    LOGIN      "
   String"USERNAME       "
   String"               "
   String"PASSWORD       "
```

```
   String"              "
   String"              "
   String"              "

PLACE 8100H
MENU_REGISTO:
   String" NOVO REGISTO   "
   String"USERNAME       "
   String"              "
   String"PASSWORD       "
   String"              "
   String"              "
   String"              "


PLACE 8180H
MENU_INICIAL:
  String"  MENU INICIAL  "
  String"1 - ENCOMENDAR  "
  String"2 - VALOR GASTO "
  String"3 - LOG OUT     "
  String"              "
  String"              "
  String"              "

PLACE 8200H
MENU_ESCOLHA:
  String"ESCOLHA A PIZZA "
  String"1 - AMALFI      "
  String"2 - CALZONA     "
  String"3 - ATUM        "
  String"4 - CAMARAO     "
  String"5 - VEGAN       "
  String"6 - FINALIZAR   "

PLACE 8280H
MENU_AMALFI:
  String"AMALFI         "
  String"1 - GRANDE    8 "
  String"2 - PEQUENA   5 "
  String"3 - VOLTAR      "
  String"              "
  String"              "
  String"              "
```

```
PLACE 8300H
MENU_CALZONE:
  String"CALZONA        "
  String"1 - GRANDE    8 "
  String"2 - PEQUENA   5 "
  String"3 - VOLTAR     "
  String"               "
  String"               "
  String"               "

PLACE 8380H
MENU_ATUM:
  String"ATUM          "
  String"1 - GRANDE    8 "
  String"2 - PEQUENA   5 "
  String"3 - VOLTAR     "
  String"               "
  String"               "
  String"               "

PLACE 8400H
MENU_CAMARAO:
  String"CAMARAO        "
  String"1 - GRANDE    8 "
  String"2 - PEQUENA   5 "
  String"3 - VOLTAR     "
  String"               "
  String"               "
  String"               "

PLACE 8480H
MENU_VEGAN:
  String"VEGAN         "
  String"1 - GRANDE    8 "
  String"2 - PEQUENA   5 "
  String"3 - VOLTAR     "
  String"               "
  String"               "
  String"               "

PLACE 8500H
MENU_FINALIZAR:
  String"   FINALIZAR    "
```

```
 String"   ENCOMENDA    "
 String"               "
 String" DESCONTO      "
 String"               "
 String" VALOR FINAL   "
 String"               "

PLACE 8580H
MENU_ERRO:
 String "    ATENCAO     "
 String "                "
 String "    OPCAO       "
 String "    ERRADA      "
 String "                "
 String "   CLICK OK     "
 String "                "

PLACE 8600H
MENU_VALOR_GASTO:
 String "    HISTORIAL   "
 String "                "
 String "                " ; MOSTRAR VALOR DO HISTORIAL
 String "                "
 String "1- VOLTAR ATRAS "
 String "                "
 String "                "

PLACE 8700H
MENU_EXISTENTE:
   String "    ATENCAO     "
   String "                "
   String "   UTILIZADOR   "
   String "      JA        "
   String "   EXISTENTE    "
   String "                "
   String "   CLICK OK     "

PLACE 8780H
MENU_FALTA:
 String "    ATENCAO     "
 String "                "
 String "     FALTA      "
 String "   CARACTERES   "
 String "                "
```

```
  String "            "
  String "   CLICK OK    "

PLACE 8800H
MENU_SUCESSO:
   String "    REGISTO    "
   String "   REALIZADO   "
   String "      COM      "
   String "    SUCESSO    "
   String "            "
   String "            "
   String "   CLICK OK    "

PLACE 8880H
PASS_INCORRETA:
   String "    ATENCAO    "
   String "            "
   String "     ERRO      "
   String "      NO       "
   String "    LOGIN      "
   String "            "
   String "   CLICK OK    "

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;INICIO;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PLACE 0000H
INICIO:
  MOV R0, PRINCIPIO  ; Inicialização  da aplicaçao
  JMP R0

PLACE 2000H
PRINCIPIO:
  MOV SP, STACKPOINTER ; Copia para SP o endereço da StackPointer
  CALL LIMPA_DISPLAY   ; Funcao limpa o display
  CALL LIMPA_PERIFERICOS  ; Funcao limpa os periféricos
  CALL LIMPA_USER_PASS  ; Funcao limpa perifericos
  MOV R1, ON_OFF ; Copia para R1 o endereço on botao ON/ON_OFF

LIGA: ; Verifica se o programa vai começar atraves da atribuicao de R0 ao botao ON_OFF
  MOVB R0,[R1]; ; Copia o byte mais significativo de R1 para R0
  CMP R0,1 ; Caso se encontre a 1 o programa inicia-se
  JNE LIGA

LIGADO:; Cria o primeiro menu a ser exposto ao utilizador
```

```
  MOV R2, MENU_INICIO;
  CALL MOSTRA_DISPLAY;
  CALL LIMPA_PERIFERICOS;
  CALL LIMPA_USER_PASS

LE_OPCAO:; Ciclo para a escolha das opcoes expostas no MENU
  MOV R1, ON_OFF;
  MOVB R1, [R1] ;
  CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
  JEQ PRINCIPIO ; Caso isto aconteça salta para o principio

  MOV R0,NR_SELL ; atribuicao do botao NR_SELL que ira selecionar varias opçoes ao
longo da aplicaçao
  MOVB R0,[R0]
  CMP R0,0
  JEQ LE_OPCAO ; Jump if Equal, enquanto o botao NR_SELL estiver a 0 continua no
ciclo

  CALL ESPERA_OK ; Funcao para que o utilizador click no ok para verificar a opcao
escolhida

  CMP R0, 3 ; Menu exposto contem apenas 2 opcoes caso o utilizador insira um
  JGE OPCAO_INVALIDA; numero maior ou igual a 3 ira mostrar uma mensagem de erro

  CMP R0, LOGIN ; verifica quando o utilizador escolhe fazer o logIN
  JEQ OPCAO_LOGIN

  CMP R0, NOVO_REGISTO ; verifica quando o utilizador escolhe fazer um novo registo
  JEQ SALTA_REG
  JMP LIGADO

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SALTA_REG: ; etiqueta criada para poder saltar para Novo registo, atravez de JMP,
como JEQ nao tem muito alcance
   JMP OPCAO_NOVREG

; opcao invalida para quando o utilizador insira um valor que nao seja possivel,
mostrando uma mensagem de erro
  OPCAO_INVALIDA:
        CALL ROTINA_ERRO
        CALL ESPERA_OK
        JMP LIGADO

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;LOGIN;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
; Mostra o menu login e  limpa os perifericos, mas contem sempre a opcao em toda a aplicao para sair desta
OPCAO_LOGIN:
    MOV R2,MENU_LOGIN
    CALL MOSTRA_DISPLAY
    CALL LIMPA_PERIFERICOS
    CALL LIMPA_USER_PASS

    MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
    MOVB R1, [R1] ;    Move para R1 o valor do botao ON_OFF
    CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
    JEQ PRINCIPIO

    CALL ESPERA_OK
; Estas instruçoes fazem com que ao escrever nos perifericos seja possivel, depois de dar ok, mostra no display o nome que inserio e a password em *
ECRA_USERNAME: ; atrbuiçoes de endereços a registos, inicializado de um ciclo while com indice=0
   MOV R2, USERNAME
   MOV R3, PASSWORD
   MOV R4, 00E0H
   MOV R6, 0 ; Total Carateres = 0
   MOV R7, 2AH
   MOV R8, 0100H
   MOV R10, 7
PROXIMOCARACTER_ECRA:; incrementa o indice "indice = indice + 1" verificando se encontra um caracter 0 ou se chega ao maximo numero possivel isto é 8 caracteres e se isto for verdade acaba o ciclo
   MOVB R9,[R2]
   CMP R9, 0H ; verifica se encontrou algum 0 assim sabe-se que o username a acabou
   JEQ ECRA_PASS ;
   MOVB [R4], R9
   CMP R6, R10 ; compara se ja chegou aos 8 caracteres
   JEQ ECRA_PASS
   ADD R6,1 ;indice = indice +1 para verificar se ja chegou aos 8 caractere
   ADD R2,1 ; incrementa o endereço do username
   ADD R4,1 ; incrementa o endereço do lugar onde ira aparecer no display
   JMP PROXIMOCARACTER_ECRA ; caso nao acabou ou encontrou um 0 continua no ciclo
ECRA_PASS: ; neste caso o mesmo procedimento para a PASSWORD
   MOV R6,0 ; inicializar outravez o indice a 0
CICLO_ECRA_PASS:
   MOVB R5,[R3]
   CMP R5, 0H
```

```
   JEQ LOGIN_USER_2
   MOVB [R8], R7
   CMP R6, R10
   JEQ LOGIN_USER_2
   ADD R3,1
   ADD R8,1
   ADD R6,1
   JMP CICLO_ECRA_PASS
; login do username
LOGIN_USER_2:
    CALL LIMPA_PERIFERICOS
    CALL ESPERA_OK
    JMP LOGIN_USER
LOGIN_USER: ; depois de dar click no ok a aplicacao inicia a procura de um user e pass ja existenes
    MOV R0,0
    MOV R10,NR_UTILIZADORES ; Endereço com o nr de utilizadores ja criados com o inicio  de 3 users
    MOV R1,[R10] ; conteudo do endereco dos nr de utilizadoes
    MOV R3,USERNAME
    MOV R4,BASE_DADOS
    MOV R5,0 ; N TOTAL DE CARACTERES
OPCAO_LOGIN_CICLO: ; ciclo que verifica se existe algum username na base de dados
    MOV R6, [R4+R5]
    MOV R7, [R3+R5]
    CMP R6,R7
    JNE PROXIMO_USER ;se nao encontrar o utilizador no inicio, salta para o proximo utilizador
    ADD R5,2 ; incrementa o R5 2 em 2, isto é verifica  2 em 2 bytes
    MOV R10,8
    CMP R5, R10; compara com 8 caso este seja verdade salta para a verificaçao da password
    JEQ LOGIN_PASS
    JMP OPCAO_LOGIN_CICLO
PROXIMO_USER: ; proximo username, isto é, verifica com outros usernames e verifica se encontra
   ADD R0,1 ; contador do nr de utilizadores
   MOV R5,0
   MOV R6,32 ; copia 32 para o R3 para poder verificar  o proximo username na base de dados
   ADD R4,R6 ; DA NOS O ENDERECO DA POSIÇAO DO ID
   CMP R0,R1
   JGT FALHA_LOGIN
   JMP OPCAO_LOGIN_CICLO
```

```
; Para a verificaçao da password foi realidado pelo mesmo procedimento que o anterior
LOGIN_PASS:
    MOV R1, POSICAO_ID ; compia para um endereco qual foi o endere que foi
sucedido o log in
    MOV [R1],R4

    MOV R3,PASSWORD
    MOV R2, POSICAO_ID
    MOV R9,[R2]

    MOV R10, 16
    ADD R9, R10
    MOV R5,0
OPCAO_PASS_CICLO:
    MOV R6, [R9+R5]
    MOV R7, [R3+R5]
    CMP R6,R7
    JNE FALHA_LOGIN
    ADD R5,2
    MOV R10,8
    CMP R5, R10
    JEQ LOGIN_SUCESSO
    JMP OPCAO_PASS_CICLO

;Login efectuado com sucesso
LOGIN_SUCESSO:
   CALL LIMPA_PERIFERICOS

   JMP LOGIN_INICIAL
; Inicialização do menu inicial
LOGIN_INICIAL:
   MOV R2,MENU_INICIAL
   CALL MOSTRA_DISPLAY
   CALL LIMPA_PERIFERICOS
   CALL LIMPA_USER_PASS
   JMP MENU_PRINCIPAL ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; caso ocorra uma falha no login  ou a aplicacao nao encontre um username e a
password mostra um disply de erro
FALHA_LOGIN:
      MOV R2,PASS_INCORRETA
      CALL MOSTRA_DISPLAY
```

```
      CALL LIMPA_PERIFERICOS
      CALL LIMPA_USER_PASS
      CALL ESPERA_OK
      JMP  LIGADO

; funcao espera ok que faz com que o utilizador tenha que por 1 no endereço do ok antes de processeder na aplicaçao
ESPERA_OK:
     MOV R2,OK
          MOV R2,[R2]
          CMP R2,0
          JEQ ESPERA_OK
     RET
;Menu que mostra que ja existe um utilizador com o mesmo username
UTILIZADOR_EXISTENTE:
      CALL ROTINA_EXISTE
      CALL LIMPA_USER_PASS
      CALL ESPERA_OK
      JMP  LIGADO
;Menu que mostra que falta caracteres necessarios para criar um novo registo
FALTA_CARACTERES:
      MOV R2,MENU_FALTA
      CALL MOSTRA_DISPLAY
      CALL LIMPA_PERIFERICOS
      CALL LIMPA_USER_PASS
      CALL ESPERA_OK
      JMP  LIGADO

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SALTA2:
   JMP PRINCIPIO
SALTAR_PARA:
   JMP LOGIN_INICIAL
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;NOVO REGISTO;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Opcao novo registo
OPCAO_NOVREG:
     MOV R2,MENU_REGISTO
     CALL MOSTRA_DISPLAY
     CALL LIMPA_PERIFERICOS
     CALL LIMPA_USER_PASS


     MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
```

```
    MOVB R1, [R1] ;    Move para R1 o valor do botao ON_OFF
    CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
    JEQ SALTA2

    CALL ESPERA_OK

; Tal como anteriormente feito, atraves do mesmo raciocinio
ECRA_USERNAME2:
    MOV R2, USERNAME
    MOV R3, PASSWORD
    MOV R4, 00E0H
    MOV R6, 0 ; Total Carateres = 0
    MOV R7, 2AH
    MOV R8, 0100H
    MOV R10, 7
PROXIMO_CARACTER_ECRA2:
    MOVB R9,[R2]
    CMP R9, 0H
    JEQ ECRA_PASSWORD
    MOVB [R4], R9
    CMP R6, R10
    JEQ ECRA_PASSWORD
    ADD R2,1
    ADD R4,1
    ADD R6,1
    JMP PROXIMO_CARACTER_ECRA2


ECRA_PASSWORD:
    MOV R6,0
CICLO_ECRA_PASS2:
    MOVB R5,[R3]
    CMP R5, 0H
    JEQ VERIFICA_USER
    MOVB [R8], R7
    CMP R6, R10
    JEQ VERIFICA_USER
    ADD R3,1
    ADD R8,1
    ADD R6,1
    JMP CICLO_ECRA_PASS2

VERIFICA_USER:
    CALL LIMPA_PERIFERICOS
```

```
    CALL ESPERA_OK
    JMP VERIFICA_USER2

VERIFICA_USER2:
    MOV R0,0 ; INDICE I = 0
    MOV R1,8 ; INDICE I < 8
    MOV R2,0 ; N DE CARACTERES
    MOV R3,USERNAME ; ATRIBUIÇÃO DO ENDEREÇO DA PASSWORD AO R3
CICLO_VERIFICA_USER:
    MOV R7, R3
    ADD R7, R0
    MOVB R4,[R7]
    MOV R5, 0H
    CMP R4, R5
    JEQ FALTA_CARACTERES
    ADD R0,1
    CMP R0,1
    JEQ VERIFICA_PASS
    JMP CICLO_VERIFICA_USER

;VERIFICA SE A PASSWORD TEM ENTRE 3 A 8 CARACTERES
VERIFICA_PASS:
    CALL ESPERA_OK
    MOV R0,0 ; INDICE I = 0
    MOV R1,8 ; INDICE I < 8
    MOV R2,0 ; N DE CARACTERES
    MOV R3,PASSWORD ; ATRIBUIÇÃO DO ENDEREÇO DA PASSWORD AO R3
CICLO_VERIFICA:
    MOV R7, R3
    ADD R7, R0
    MOVB R4,[R7]
    MOV R5, 0H
    CMP R4, R5
    JEQ FALTA_CARACTERES
    ADD R0,1
    CMP R0,3
    JEQ VERIFICA_USERNAME_IGUAL
    JMP CICLO_VERIFICA
;VERIFICA SE EXISTE ALGUM USERNAME COM O MESMO NOME
VERIFICA_USERNAME_IGUAL:
    MOV R0,0
    MOV R10,NR_UTILIZADORES
    MOV R1,[R10]
    MOV R3,USERNAME
```

```
    MOV R4,BASE_DADOS
    MOV R5,0
CICLO_VERIFICA_USERNAME_IGUAL:
    MOV R6, [R4+R5]
    MOV R7, [R3+R5]
    CMP R6,R7
    JNE VERIFICA_PROXIMO_USER
    ADD R5,2
    MOV R10,8
    CMP R5, R10
    JEQ UTILIZADOR_EXISTENTE
    JMP CICLO_VERIFICA_USERNAME_IGUAL
VERIFICA_PROXIMO_USER:
   ADD R0,1
   MOV R5,0
   MOV R6,32
   ADD R4,R6
   CMP R0,R1
   JEQ CREATE
   JMP CICLO_VERIFICA_USERNAME_IGUAL
;CRIA O NOVO REGISTO
CREATE:
    MOV R3,USERNAME
    MOV R4,PASSWORD
    MOV R5,BASE_DADOS
    MOV R8,NR_UTILIZADORES
    MOV R7,[R8] ; nº de utilizadores registados
    MOV R6, 32
    MUL R7,R6 ; 3 * 32
    ADD R5,R7  ; APONTA PARA A POSIÇAO USERNAME [base_dados]+3*32
    MOV R8,R5
    MOV R7, 16
    ADD R8,R7 ; APONTA PARA A POSSIÇAO DA PASSWORD  [base_dados]+3*32 + 16
    MOV R0, 0
    MOV R1,NR_UTILIZADORES ; incrementa + 1
    MOV R7,[R1]
    ADD R7,1
    MOV [R1],R7
CICLO_CREATE:
    MOV R9,[R3+R0]
    MOV [R5+R0], R9
    MOV R9, [R4+R0]
    MOV [R8 + R0], R9
    ADD R0, 2
```

```
    MOV R7, 8
    CMP R0,R7
    JEQ SALTA_LIGADO
    JMP CICLO_CREATE
;JUMP PARA LIGADO
SALTA_LIGADO:
   MOV R2,MENU_SUCESSO
   CALL MOSTRA_DISPLAY
   CALL LIMPA_PERIFERICOS
   CALL LIMPA_USER_PASS
   CALL ESPERA_OK
   JMP LIGADO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; mostra ao utilizador que ocorreu um erro
OPCAO_INVALIDA_2:
  CALL ROTINA_ERRO
  CALL ESPERA_OK
  JMP  LOGIN_INICIAL

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;MENU_PRINCIPAL;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; menu principal que mostra o valor gasto, a opcao encomendar ou logout
MENU_PRINCIPAL:
   MOV R3, GUARDAR_SALDO
   MOV R4,0
   MOV [R3],R4

   MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
   MOVB R1, [R1] ;      Move para R1 o valor do botao ON_OFF
   CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
   JEQ SALTA

   MOV R0,NR_SELL
   MOVB R0,[R0]

   CMP R0,0
   JEQ MENU_PRINCIPAL

   CALL ESPERA_OK

   CMP R0, 4
   JGE OPCAO_INVALIDA_2

   CMP R0,ENCOMENDAR
   JEQ OPCAO_ENCOMENDAR
```

```
   CMP R0, VALOR_GASTO
   JEQ OPCAO_VALOR_GASTO

   CMP R0, LOG_OUT
   JEQ SALTA
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SALTA:
   JMP LIGADO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; opcao que mostra um menu com o valor gasto
OPCAO_VALOR_GASTO:
   MOV R2, MENU_VALOR_GASTO
   CALL MOSTRA_DISPLAY
   CALL LIMPA_PERIFERICOS
   CALL LIMPA_USER_PASS
   MOV R2, POSICAO_ID
   MOV R7,8
   MOV R2, [R2]
   MOV R1, [R2 + R7]
   CALL CONVERTE_TO_CHAR
VALOR_GASTO_CICLO:
   MOV R2, ON_OFF; Move para R1 o endereço do botao ON_OFF
   MOVB R2, [R2] ;      Move para R1 o valor do botao ON_OFF
   CMP R2, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
   JEQ SALTA

   MOV R0,NR_SELL
   MOVB R0,[R0]

   CMP R0,0
   JEQ VALOR_GASTO_CICLO

   CALL ESPERA_OK

   CMP R0, 2
   JGE OPCAO_ERRADA

   CMP R0,VOLTAR_ATRAS
   JEQ SALTO

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Etiquetas usas por causa que o JEQ nao abranje um salto com muitas linhas
SALTO:
```

```
 JMP LOGIN_INICIAL

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Este menu avisa ao utilizador que ocorreu um erro
OPCAO_ERRADA:
   CALL ROTINA_ERRO
   CALL ESPERA_OK
   JMP OPCAO_VALOR_GASTO

;;;;;;;;;;;;;;;;;;;;;;;;;;;;MENU ENCOMENDAR E ESCOLHA DA PIZZA;;;;;;;;;;;;;;;;;;;;;;
; Menu encomendar, com as opcoes de escolher as pizzas desejadas pelo utilizador,
sempre com a opçao de sair da aplicacao e finalizar a encomenda
OPCAO_ENCOMENDAR:
   MOV R2,MENU_ESCOLHA
   CALL MOSTRA_DISPLAY
   CALL LIMPA_PERIFERICOS
   CALL LIMPA_USER_PASS
CICLO_ENCOMENDAR:
   MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
   MOVB R1, [R1] ;      Move para R1 o valor do botao ON_OFF
   CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
   JEQ SALTA

   MOV R0,NR_SELL
   MOVB R0,[R0]

   CMP R0,0
   JEQ CICLO_ENCOMENDAR

   CALL ESPERA_OK

   CMP R0, 7
   JGE OPCAO_INVALIDA_3

   CMP R0, PIZZA_AMALFI
   JEQ MENU_PIZZA_AMALFI

   CMP R0, PIZZA_CALZONE
   JEQ MENU_PIZZA_CALZONE

   CMP R0, PIZZA_ATUM
   JEQ MENU_PIZZA_ATUM

   CMP R0, PIZZA_CAMARAO
```

```
    JEQ MENU_PIZZA_CAMARAO

    CMP R0, PIZZA_VEGAN
    JEQ MENU_PIZZA_VEGAN

    CMP R0, FINALIZAR
    JEQ SALTAR_FINAL
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Etiquetas usadas por causa que o JEQ nao abranje um salto com muitas linhas
SALTAR_FINAL:
    JMP FINALIZAR_ENCOMENDA

SALTAR_EN:
    JMP OPCAO_ENCOMENDAR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;MENU_PIZZA_AMALFI;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Menu da pizza AMALFI, com as opcoes de escolher a pizza grande, pequena ou voltar
atras
MENU_PIZZA_AMALFI:
    MOV R2,MENU_AMALFI
    CALL MOSTRA_DISPLAY
    CALL LIMPA_PERIFERICOS
    CALL LIMPA_USER_PASS

CICLO_AMALFI:
    MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
    MOVB R1, [R1] ;      Move para R1 o valor do botao ON_OFF
    CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
    JEQ SALTA

    MOV R0,NR_SELL
    MOVB R0,[R0]

    CMP R0,0
    JEQ CICLO_AMALFI

    CALL ESPERA_OK

    CMP R0, 4
    JGE OPCAO_INVALIDA_3

    CMP R0, GRANDE
    JEQ PIZZA_MAIOR_VALOR
```

```
   CMP R0, PEQUENO
   JEQ PIZZA_MENOR_VALOR

   CMP R0, VOLTAR
   JEQ SALTAR_EN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; este menu avisa ao utilizador que ocorreu um erro
OPCAO_INVALIDA_3:
  CALL ROTINA_ERRO
  CALL ESPERA_OK
  JMP OPCAO_ENCOMENDAR

;;;;;;;;;;;;;;;;;;;;;;;;;;;MENU_PIZZA_CALZONE;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Menu da pizza calzone, com as opcoes de escolher a pizza grande, pequena ou voltar atras
MENU_PIZZA_CALZONE:
   MOV R2,MENU_CALZONE
   CALL MOSTRA_DISPLAY
   CALL LIMPA_PERIFERICOS
   CALL LIMPA_USER_PASS

CICLO_CALZONE:
   MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
   MOVB R1, [R1] ;      Move para R1 o valor do botao ON_OFF
   CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
   JEQ SALTA

   MOV R0,NR_SELL
   MOVB R0,[R0]

   CMP R0,0
   JEQ CICLO_CALZONE

   CALL ESPERA_OK

   CMP R0, 4
   JGE OPCAO_INVALIDA_3

   CMP R0, GRANDE
   JEQ PIZZA_MAIOR_VALOR

   CMP R0, PEQUENO
   JEQ PIZZA_MENOR_VALOR
```

```
   CMP R0, VOLTAR
   JEQ SALTAR_EN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; No caso que a pizza grande, isto é, de maior valor seja escolhida,a aplicaçao guarda no
endereco GUARDAR_SALDO e soma no seu conteudo a constante 8
PIZZA_MAIOR_VALOR:
  MOV R5,GUARDAR_SALDO ;
  MOV R3,[R5]
  MOV R4, 8 ; CONSTANTE
  ADD R3,R4 ; adiciona 8 ao valor esta no endereco GUARDAR_SALDO e adiciona 8
  MOV [R5],R3 ; volta a colocar depois de fazer a soma o valor que foi somado
  JMP OPCAO_ENCOMENDAR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; No caso que a pizza pequena, isto é, de menor valor seja escolhida,a aplicaçao guarda
no endereco GUARDAR_SALDO e soma no seu conteudo a constante 5
PIZZA_MENOR_VALOR:
  MOV R5,GUARDAR_SALDO
  MOV R3, [R5]
  MOV R4, 5 ; CONSTANTE
  ADD R3,R4
  MOV [R5],R3
  JMP OPCAO_ENCOMENDAR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SALTA_PRINCIPIO:
   JMP PRINCIPIO

;;;;;;;;;;;;;;;;;;;;;;;;;;;MENU_PIZZA_ATUM;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Menu da pizza atum, com as opcoes de escolher a pizza grande, pequena ou voltar
atras

MENU_PIZZA_ATUM:
      MOV R2,MENU_ATUM
      CALL MOSTRA_DISPLAY
      CALL LIMPA_PERIFERICOS
      CALL LIMPA_USER_PASS

CICLO_ATUM:
      MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
      MOVB R1, [R1] ;  Move para R1 o valor do botao ON_OFF
      CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
      JEQ SALTA_PRINCIPIO
```

```
    MOV R0,NR_SELL
    MOVB R0,[R0]

    CMP R0,0
    JEQ CICLO_ATUM

    CALL ESPERA_OK

    CMP R0, 4
    JGE OPCAO_INVALIDA_3

    CMP R0, GRANDE
    JEQ PIZZA_MAIOR_VALOR

    CMP R0, PEQUENO
    JEQ PIZZA_MENOR_VALOR

    CMP R0, VOLTAR
    JEQ SALTAR_EN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;MENU_PIZZA_CAMARAO;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Menu da pizza CAMARAO, com as opcoes de escolher a pizza grande, pequena ou
voltar atras

MENU_PIZZA_CAMARAO:
    MOV R2,MENU_CAMARAO
    CALL MOSTRA_DISPLAY
    CALL LIMPA_PERIFERICOS
    CALL LIMPA_USER_PASS

CICLO_CAMARAO:
    MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
    MOVB R1, [R1] ;  Move para R1 o valor do botao ON_OFF
    CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
    JEQ SALTA_PRINCIPIO

    MOV R0,NR_SELL
    MOVB R0,[R0]

    CMP R0,0
    JEQ CICLO_CAMARAO

    CALL ESPERA_OK
```

```
    CMP R0, 4
    JGE OPCAO_INVALIDA_3

    CMP R0, GRANDE
    JEQ PIZZA_MAIOR_VALOR

    CMP R0, PEQUENO
    JEQ PIZZA_MENOR_VALOR

    CMP R0, VOLTAR
    JEQ SALTAR_EN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;MENU_PIZZA_VEGAN;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Menu da pizza Vegan, com as opcoes de escolher a pizza grande, pequena ou voltar
atras
MENU_PIZZA_VEGAN:
    MOV R2,MENU_VEGAN
    CALL MOSTRA_DISPLAY
    CALL LIMPA_PERIFERICOS
    CALL LIMPA_USER_PASS

CICLO_VEGAN:
    MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
    MOVB R1, [R1] ;    Move para R1 o valor do botao ON_OFF
    CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
    JEQ SALTA_PRINCIPIO

    MOV R0,NR_SELL
    MOVB R0,[R0]

    CMP R0,0
    JEQ CICLO_VEGAN

    CALL ESPERA_OK

    CMP R0, 4
    JGE OPCAO_INVALIDA_3

    CMP R0, GRANDE
    JEQ PIZZA_MAIOR_VALOR

    CMP R0, PEQUENO
    JEQ PIZZA_MENOR_VALOR
```

```
    CMP R0, VOLTAR
    JEQ SALTAR_ENCOMENDAR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SALTAR_ENCOMENDAR:
   JMP OPCAO_ENCOMENDAR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;FINALIZAR_ENCOMENDA;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Finalizar a encomenda, é neste menu que a aplicação finaliza que mostra ao utilizador
quando ira ter que pagar,
;depois de dar ok a aplicaçao soma o valor  que o utilizador ja tinha gasto  e soma o
com o valor a pagar, de seguida
;mete o endereco onde se encontra o saldo e mete o seu conteudo a 0
FINALIZAR_ENCOMENDA:
   MOV R2,MENU_FINALIZAR
   CALL MOSTRA_DISPLAY
   CALL LIMPA_PERIFERICOS
   CALL LIMPA_USER_PASS
   MOV R2, POSICAO_ID ; endereço da posicao iD
   MOV R3,8
   MOV R7, GUARDAR_SALDO
   MOV R10,[R7]

   MOV R1, [R7]
   CALL CONVERTE_VALOR_FINAL

   MOV R2, [R2]
   MOV R8, [R2 + R3]
   ADD R8, R10
   MOV[R2+R3],R8

   CALL ESPERA_OK


CICLO_ENC_FIM:
   MOV R1, ON_OFF; Move para R1 o endereço do botao ON_OFF
   MOVB R1, [R1] ;      Move para R1 o valor do botao ON_OFF
   CMP R1, 1 ; Verifica se é para desligar a aplicacao (1 desligar )
   JEQ SALTA_PRINCIPIO

   CALL ESPERA_OK

   JMP  SALTAR_PARA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;Funcao CONVERTE que converte o valor final a pagar

CONVERTE_VALOR_FINAL:
    PUSH R0
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    MOV R0, 10
    MOV R2, BASE2
    ADD R2, 2; Posição do primeiro carater
    MOV R3, 0 ; Total Carateres = 0
PROXIMO_CARACTERS:
    MOV R4, R1
    MOD R4, R0 ;Resto da divisao
    DIV R1, R0 ;Divisão inteira por 10
    MOV R5, 48
    ADD R5, R4
    MOV R4, R2
    MOVB [R4], R5
    SUB R2, 1
    ADD R3, 1
    CMP R1, 0
    JNE PROXIMO_CARACTERS
    CMP R3, 1
    JGT ROTINA_FIM
ROTINA_FIM:
    POP R5
    POP R4
    POP R3
    POP R2
    POP R0
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Funcao CONVERTE que converte o valor gasto na PIZZARIA
CONVERTE_TO_CHAR:
  PUSH R0
  PUSH R2
  PUSH R3
  PUSH R4
  PUSH R5
  MOV R0, 10
  MOV R2, BASE
  ADD R2, 2; Posição do primeiro carater
```

```
  MOV R3, 0 ; Total Carateres = 0
PROXIMOCARACTER:
  MOV R4, R1
  MOD R4, R0 ;Resto da divisao
  DIV R1, R0 ;Divisão inteira por 10
  MOV R5, 48
  ADD R5, R4
  MOV R4, R2
  MOVB [R4], R5
  SUB R2, 1
  ADD R3, 1
  CMP R1, 0
  JNE PROXIMOCARACTER
  CMP R3, 3
  JGT FIM_ROTINA

FIM_ROTINA:
  POP R5
  POP R4
  POP R3
  POP R2
  POP R0
  RET

;Mostra o display
; Funcao mostra display
MOSTRA_DISPLAY:
          PUSH R0
          PUSH R1
          PUSH R3
  MOV R0, DISPLAY_BEGIN     ; Guarda o endereço onde o Display começa
  MOV R1, DISPLAY_END       ; Guarda o endereço onde o Display acaba
CICLO_MOSTRA:
  MOV R3, [R2]          ; Copia a parte do display a copiar para o registo R3
  MOV [R0], R3                  ; Coloca no Display a parte que copiou para R3
  ADD R2,2              ; Incrementa o endereço do display para obter o proximo a ser
copiado
  ADD R0,2      ; Incrementa o endereço para o qual e copiado o display
  CMP R0,R1             ; Compara para ver se se ja chegou ao fim do display
  JLE CICLO_MOSTRA
  POP R3
  POP R1
  POP R0
  RET
```

```
;LIMPA_PERIFERICOS
; funcao que limpa os botoes
LIMPA_PERIFERICOS:
   PUSH R0
   PUSH R1
   PUSH R2
   PUSH R4
   PUSH R5
   PUSH R6
   MOV R0,ON_OFF          ; copia para R0 o endereço do botão ON_OFF
   MOV R1,NR_SELL         ; Copia para R1 o endereço do botao NR_SELL
   MOV R2, OK             ; copia para R2 o endereço do botão ok
   MOV R4, 0              ; coloca no registo R4 a constante 0
   MOVB [R0], R4          ; coloca no valor do botao ON_OFF o valor zero
   MOVB [R1], R4          ; coloca no valor do botao NR_SELL o valor zero
   MOVB [R2], R4          ; coloca no valor do botao OK o valor zero
   POP R6
   POP R5
   POP R4
   POP R2
   POP R1
   POP R0
   RET

; Funcao que mostra ao utilizador que ocorreu um erro
ROTINA_ERRO:
         PUSH R0
         PUSH R1
      PUSH R2
         MOV R2, MENU_ERRO        ; copia para R2 o endereço do menu de erro
         CALL MOSTRA_DISPLAY      ; chama a função para mostrar no display o
menu_erro
      CALL LIMPA_PERIFERICOS    ; limpa os perifericos e termina
      POP R2
      POP R1
      POP R0
      RET

; Funcao que mostra ao utilizador que ja existe um utilizador
ROTINA_EXISTE:
         PUSH R0
         PUSH R1
     PUSH R2
```

```
        MOV R2, MENU_EXISTENTE      ; copia para R2 o endereço do menu existente
        CALL MOSTRA_DISPLAY         ; chama a função para mostrar no display o
menu_existente
        CALL LIMPA_PERIFERICOS      ; limpa os perifericos e termina
    POP R2
    POP R1
    POP R0
    RET

;funcao que limpa o display
LIMPA_DISPLAY:
  PUSH R0
  PUSH R1
  PUSH R2
  MOV R0, DISPLAY_BEGIN ; Copia para R0 o endereço onde começa o Display
        MOV R1, DISPLAY_END ; Copia para R1 o endereço onde acaba o Display
CICLO_LIMPA:
  MOV R2, CARACTER_VAZIO  ; Copia para R2 o endereço do CARACTER_VAZIO
  MOVB [R0],R2            ; Copia o carater vazio para o valor de R0
  ADD R0,1
  CMP R0,R1               ; compara o endereço do inicio do display com o endereço do fim
do display
  JLE CICLO_LIMPA         ; caso nao sejam iguais os endereços o programa volta ao inicio
do ciclo
  POP R2
  POP R1
  POP R0
  RET

;;;;;;;;;;;;;;;;;;;;;;;;;;LIMPA PERIFERICOS U/P;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;funcao que limpa os perifer username e password

LIMPA_USER_PASS:
   PUSH R0
   PUSH R1
   PUSH R2
   PUSH R3
   PUSH R4
   MOV R0, 0020H  ;  Copia para R0 o endereço 0020H ( endereço inicio periferico
USERNAME)
        MOV R1, 002FH  ;  Copia para R1 o endereço 002FH ( endereço fim periferico
USERNAME)
   MOV R3, 0030H  ;  Copia para R3 o endereço 0030H ( endereço inicio periferico
PASSWORD)
```

```
   MOV R4, 003FH  ;  Copia para R4 o endereço 003FH ( endereço fim periferico
PASSWORD)
CICLO_LIMPA_USER:
   MOV R2, 0     ;
   MOVB [R0],R2   ;  copia para o valor do inicio do periferico USERNAME o endereço de
R2
   ADD R0,1      ;  adiciona 1 ao endereço do inicio do periferico do USERNAME
   CMP R0,R1     ;   compara o endereço do fim do USERNAME com a posição em que se
encontra o carater que esta em R0
   JLE CICLO_LIMPA_USER    ; Salta para o ciclo caso o endereço de R0 ainda não seja
igual ao de R1 indicando que ainda
                    ; não chegou ao fim do periferico
CICLO_LIMPA_PASS:
   MOV R2, 0
   MOVB [R3],R2          ; copia para o valor do inicio do perferico PASSWORD o
endereço de R2
   ADD R3,1              ; adiciona 1 a R3
   CMP R3,R4             ;   compara o endereço do fim da PASSWORD com a posição em
que se encontra o carater que esta em R0
   JLE CICLO_LIMPA_USER    ; Salta para o ciclo caso o endereço de R3 ainda não seja
igual ao de R4 indicando que ainda                    ; não chegou ao fim do periferico
   POP R4
   POP R3
   POP R2
   POP R1
   POP R0
   RET
```