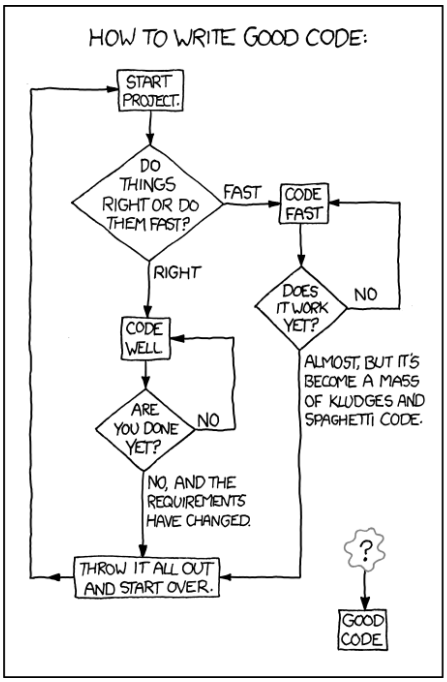
UNIVERSIDADE da MADEIRA

Arquitetura de Computadores

Engenharia Informática

Ano Lectivo: 2017 – 2018



Relatório

Projeto 2 – Pizzaria Online Assembly

Docentes:

Dionísio Barros
Nuno Ferreira
Pedro Camacho
Sofia Inácio

Discentes:

Lisandro Marote - 2030315
Pedro Rodrigues - 2091412

Índice

(Pág.)

Introdução

No âmbito da cadeira de Arquitetura de Computadores foi-nos proposta a realização de um projeto, tendo em vista a implementação de um programa que visa o funcionamento de uma pizzaria online utilizando a linguagem de baixo nível *assembly*.

Para o desenvolvimento do programa utilizou-se o *software* disponibilizado pelos docentes, mais concretamente o simulador para o processador PEPE *SIMAC* (***Sim**ulador de **A**rquitetura de **C**omputadores*).

Para projeção deste projeto irá ser realizado uma análise detalhada, onde iremos recorrer a diversos fluxogramas de maneira a nos poder orientar melhor para o alcance dos objetivos inicialmente propostos.

O processador PEPE (*Processador Especial Para Ensino*) implementa um microprocessador de 16 bits. O processador PEPE distingue-se do processador anteriormente estudado (*PEPE-8*), pois, as diferenças mais significativas são: o PEPE tem apenas uma memória (*partilhada para dados e instruções*) enquanto que o PEPE-8 tem duas (*uma para dados e uma para instruções*); o PEPE tem 16 registos disponíveis, enquanto que o PEPE-8 só tem um; e o PEPE é de 16 bits, enquanto que o PEPE-8 é de 8 bits.

Assembly é uma linguagem de baixo nível que permite fácil comunicação com o processador diretamente. *Assembly* é convertido em código máquina através de um compilador.

A Pizzaria Online terá como função principal a encomenda de pizzas. O utilizador tem ao seu dispor um menu, bem como diferentes tamanhos de pizza e a possibilidade de poder encomendar mais do que uma.

Terá um sistema de desconto, que deixa nos dados do utilizador um histórico com valor total de pizzas já encomendas, caso esse montante ultrapasse os 50 euros, este irá efetuar um desconto de 50% na pizza de menor valor.

Programa foi projetado para diferentes opções inválidas que um utilizador poderá cometer tais como: utilizador inexistente, password incorreta, opção inválida, não cumpriu requisitos de registo etc.

Objetivos:

Principal objetivo deste projeto é dominar e adquirir conhecimentos sobre a linguagem de baixo nível *Assembly*. Temos também como objetivo consolidar os conhecimentos adquiridos quer nas aulas teóricas, teórico-práticas como também nas aulas práticas laboratoriais da cadeira acerca do processador PEPE.

Desenvolvimento

A interface da pizzaria online é representada através de um display (periférico de saída) de dimensões 7*16 (7 linhas de 16 caracteres).

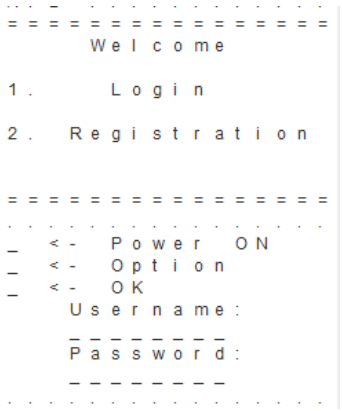


Figura 1 – Display Menu Inicial Pizzaria Online.

A interação com o utilizador é feita através dos botões e um campo de introdução de caracteres localizado abaixo do display da "pizzaria online".

Como botões temos os seguintes, ON e NR_SEL (Option) e botão OK.

O Botão ON é utilizado para activar o programa da pizzaria. O botão NR_SEL é utilizado para inserir a opção numéria para as diferentes opções que lhe poderão surgir no display. Botão OK é utilizado para validar as opções numéricas inseridas pelo utilizador.

Os campos que permitem a introdução de caracateres foram projectados para com que o utilizador possa entrar com a sua conta no Login, ou caso, contrário criar a sua conta.

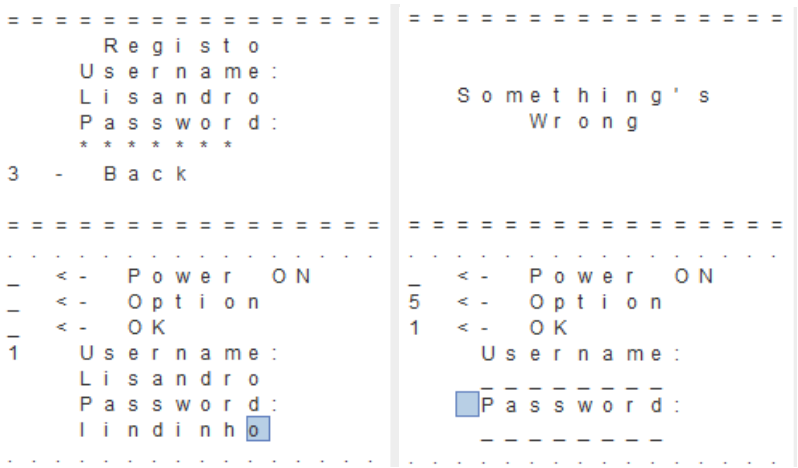


Figura 2 – Menu Registo e Display com Mensagem de Erro.

Após entrar na sua conta, é apresentado um Menu de opções ao utilizador (*Menu Pizzas ou efectuar a saída da sua conta*).

Caso seja selecionado a primeira opção, este irá fazer output com o menu das pizzas disponíveis onde de seguida irá apresentar os tamanhos pretendidos para essa mesma encomenda.

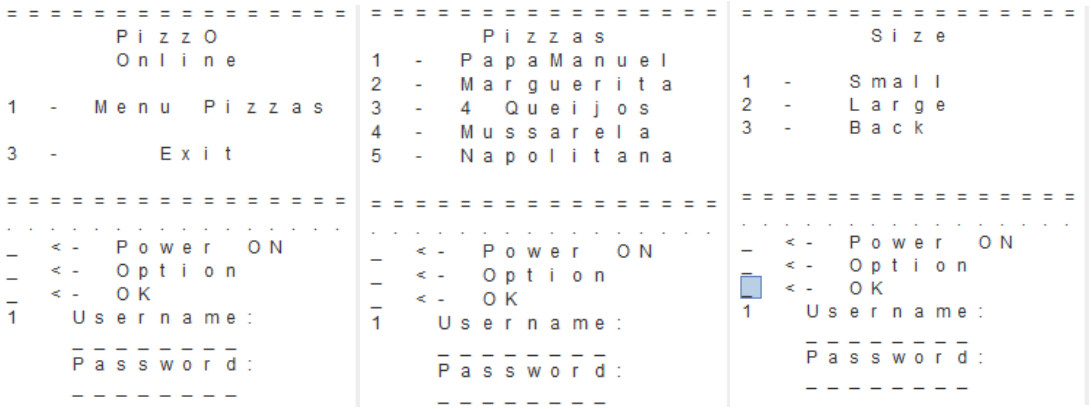


Figura 3 – Display´s Menu Pizzo, Pizzas e Tamanhos.

Foi dado a opção de o utilizador poder voltar atrás na escolha da pizza quando este está na opção de escolher seu tamanho, devido a este ter podido se enganar ao efectuar a sua escolha da pizza e desejar voltar a reformular a sua encomenda, daí no display dos tamanhos aparecer a funcionaldiade de voltar atrás, que neste caso em concreto volta ao menu das pizzas.

Apôs a escolha da pizza e seu tamanho irá ser divulgado no display a possibilidade de o utilizador poder efectuar mais do que uma encomenda, ou então efectuar o pagamento. Foi também dado a possibilidade de o utilizador efectuar o logout sem efectuar qualquer tipo de encomenda, uma vez mais é uma funcinalidade que enquanto grupo entendemos que houvesse sentido existir.

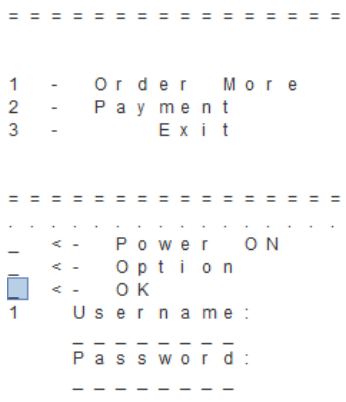


Figura 4 – Display Finalization.

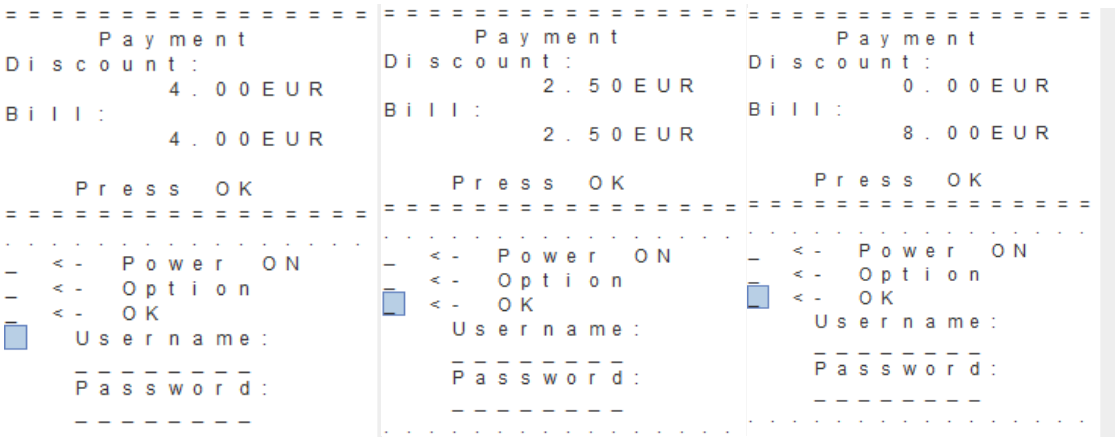


Figura 5 – Display Payment, com diferentes tipos de situações.

Para o *display* pagamento, apôs as encomendas pré-realizadas pelo utilizador é feito um cruzamento com seu histórico de encomendas efectuadas anteriormente, mais concretamente o valor total de encomendas até o momento que este utiliza a aplicação, e caso esse valor ultrapasse o montante de 50 euros é realizado um desconto de 50% na pizza de menor valor.

No caso de o utilizador ultrapassar o montante de 50 euros, mas este efectuar apenas uma encomenda de uma pizza grande é feito o desconto de 50% nessa mesma pizza, caso contrário programa foi projectado de maneira a atribuir o desconto à pizza de menor valor.

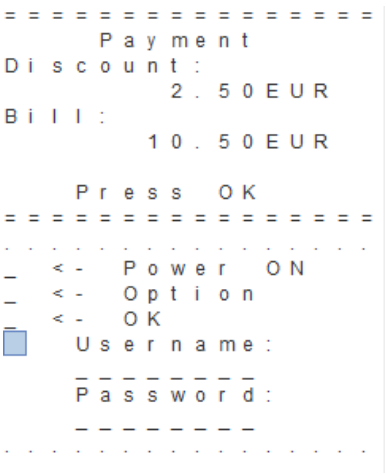


Figura 6 – Display Payment onde programa atribui desconto à pizza de menor valor.

Login:

Para efetuar login é pedido ao utilizador que introduza o seu username e a sua password. Posto isto é feita uma pesquisa na base de dados, se as credenciais introduzidas forem encontradas, é guardado no registo 9 o endereço de memória deste utilizador. Este endereço é mais tarde utilizado para verificar se o cliente tem direito a desconto bem como atualizar o histórico do mesmo.

Registo:

Para efetuar um novo registo, é pedido ao utilizador um username e uma password, quando o utilizador pressionar o botão OK, o programa vai fazer uma consulta à base de dados e verificar a existência do username, se este existir já na base de dados, deverá ser apresentada uma mensagem de erro, caso este ainda não conste na base de dados, será feita a confirmação de que a password tem pelo menos três caracteres. Se todos os requisitos forem cumpridos, será adicionado um novo registo à base de dados, este terá um saldo acumulado inicial de 0€ e a cada compra será adicionado o total.

Pagamentos:

Ao entrar no menu para efetuar o pagamento, é mostrado no display o valor total da compra já com os descontos e também é indicado qual o valor do desconto da mesma. Para calcular o total a pagar, temos duas constantes com o valor de cada uma das pizzas. Durante a encomenda é guardado nos registos R8 e R10, a quantidade de pizzas pequenas ou grandes respetivamente. Se o cliente tiver direito a desconto e este tiver pelo menos uma pizza pequena, é feito um desconto de 2.50€ caso contrário o desconto será de 5.00€. Após o cálculo do total a pagar e dos descontos, é feita uma atualização no histórico. O histórico é atualizado sempre que há uma compra, quando este excede os 50.00€ é lhe subtraído os 50.00€ e é dado um desconto ao utilizador.

Conclusão

Após a realização deste projeto, podemos concluir que todos os objetivos inicialmente propostos foram alcançados, ou seja, conseguiu-se desenvolver um programa que permite o funcionamento integral de uma "Pizzaria Online" com todas as funcionalidades pretendidas.

No decorrer deste projeto tivemos algumas dificuldades, tal com a compreensão da linguagem *assembly*, que após alguma prática foi ultrapassada, como também o funcionamento do simulador, que "*crashava"* muito frequentemente. Uma outra dificuldade sentida no grupo, foi a implementação do sistema de desconto, visto este em alguns casos ter de apresentar valores decimais com vírgula, tendo esta sido a maior dificuldade e mais demorosa. Apesar das dificuldades foi possível alcançar o objetivo final.

Concluindo, com este projeto adquirimos mais experiência e um maior à vontade com a linguagem de baixo nível assembly, como também mais conhecimentos sobre o funcionamento do processador PEPE. Tivemos oportunidade de por em prática os conhecimentos adquiridos nas aulas teóricas, teórico-práticas e práticas.

Bibliografia

. DELGADO, José; RIBEIRO, Carlos; *Arquitetura de Computadores,* FCA, 2007

. Manual do PEPE,
http://moodle.cee.uma.pt/pluginfile.php/35055/mod_resource/content/1/Manual_Pepe.pdf

Anexos

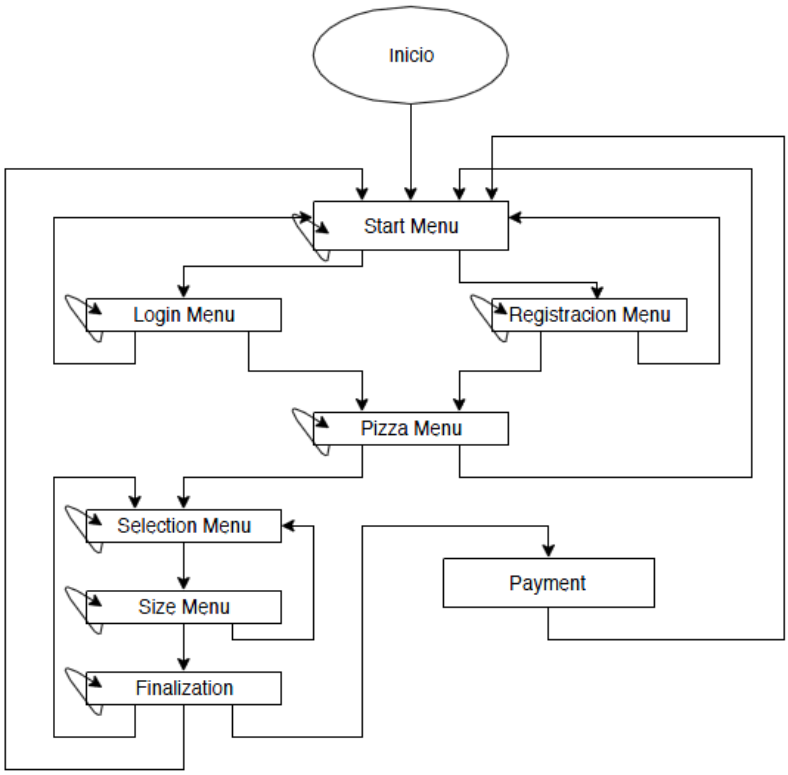


Figura 7 – Fluxograma Principal.

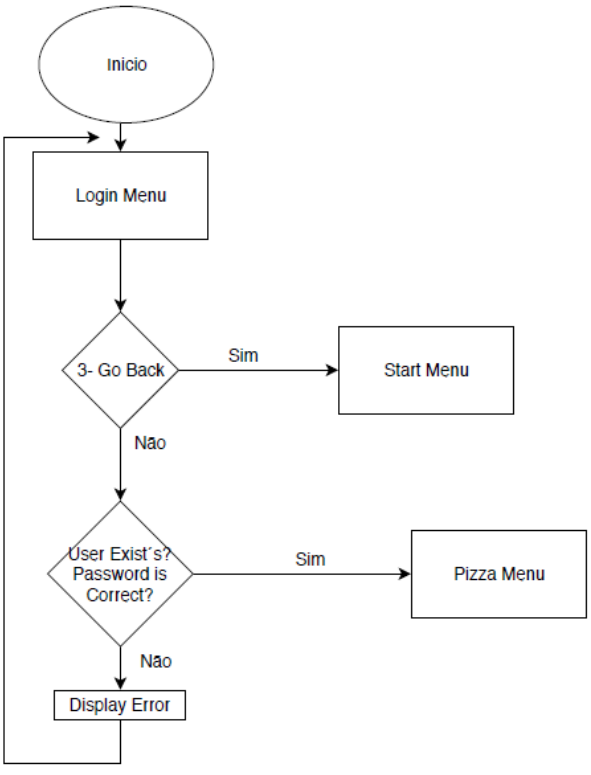


Figura 8 – Fluxograma Login Menu.

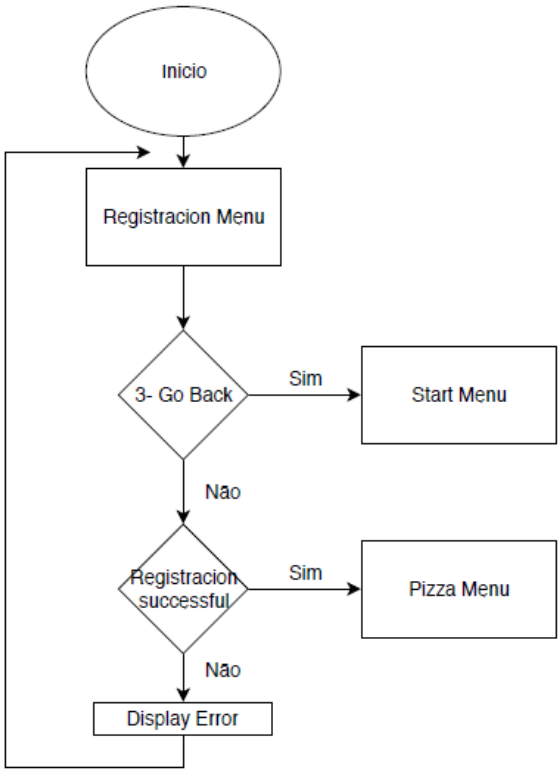


Figura 9 – Fluxograma Registracion Menu.

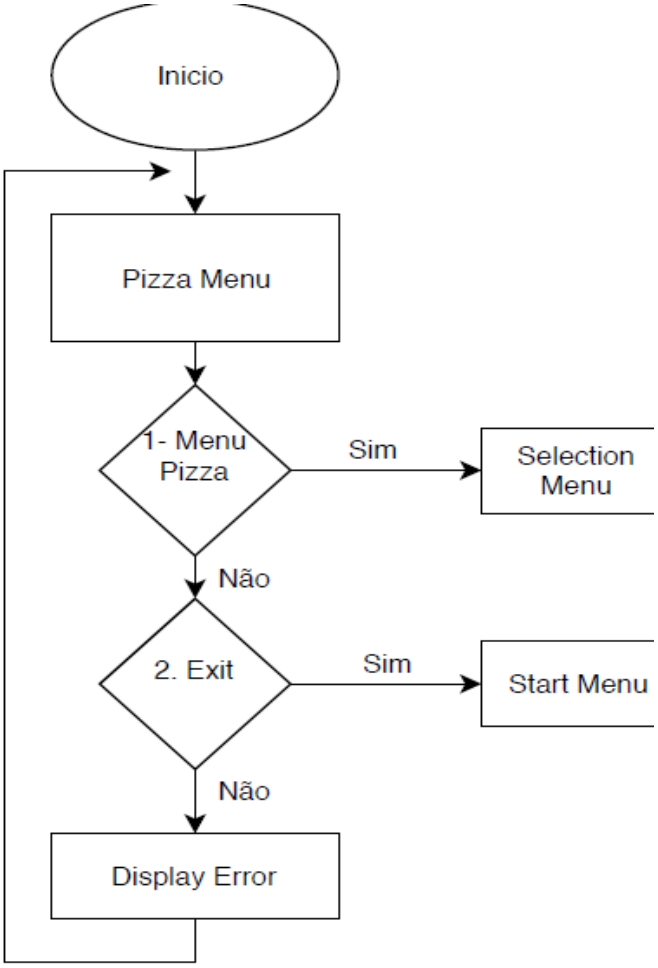


Figura 10 – Fluxograma Pizza Menu.

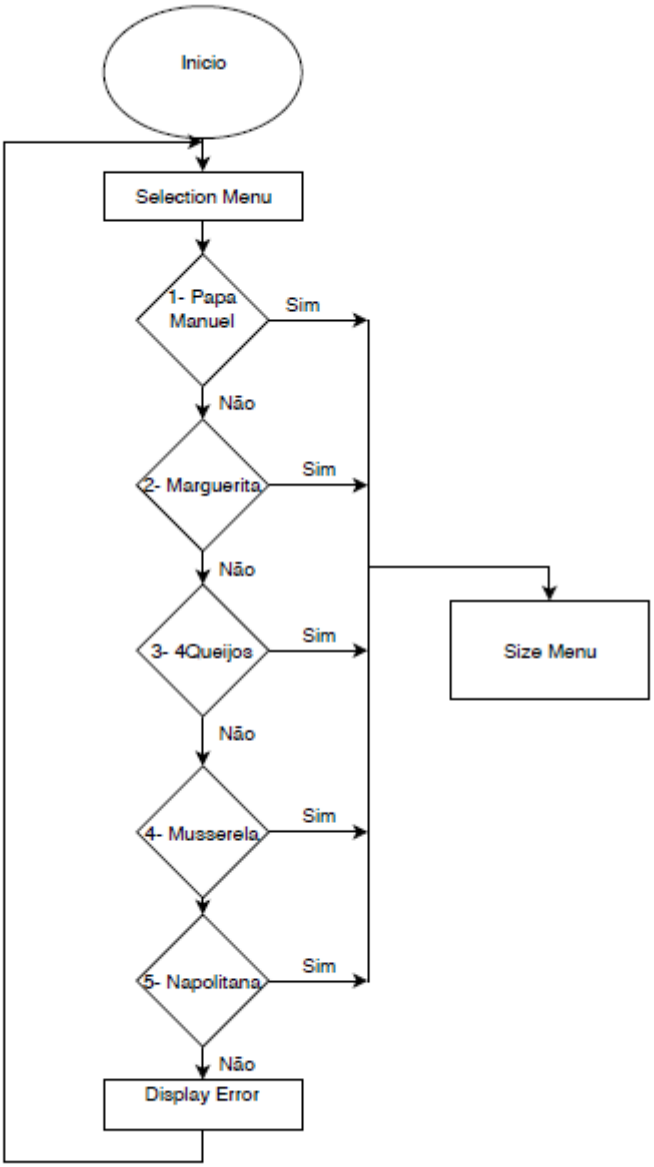


Figura 11 – Fluxograma Selection Menu.

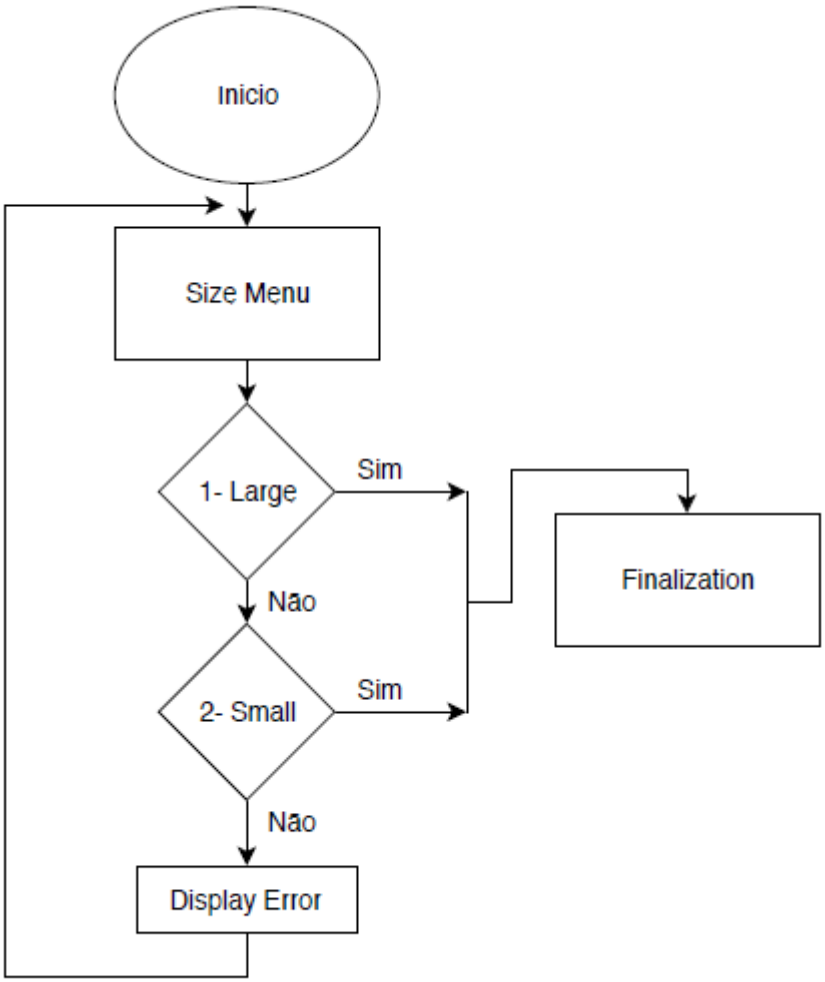


Figura 12 – Fluxograma Size Menu.

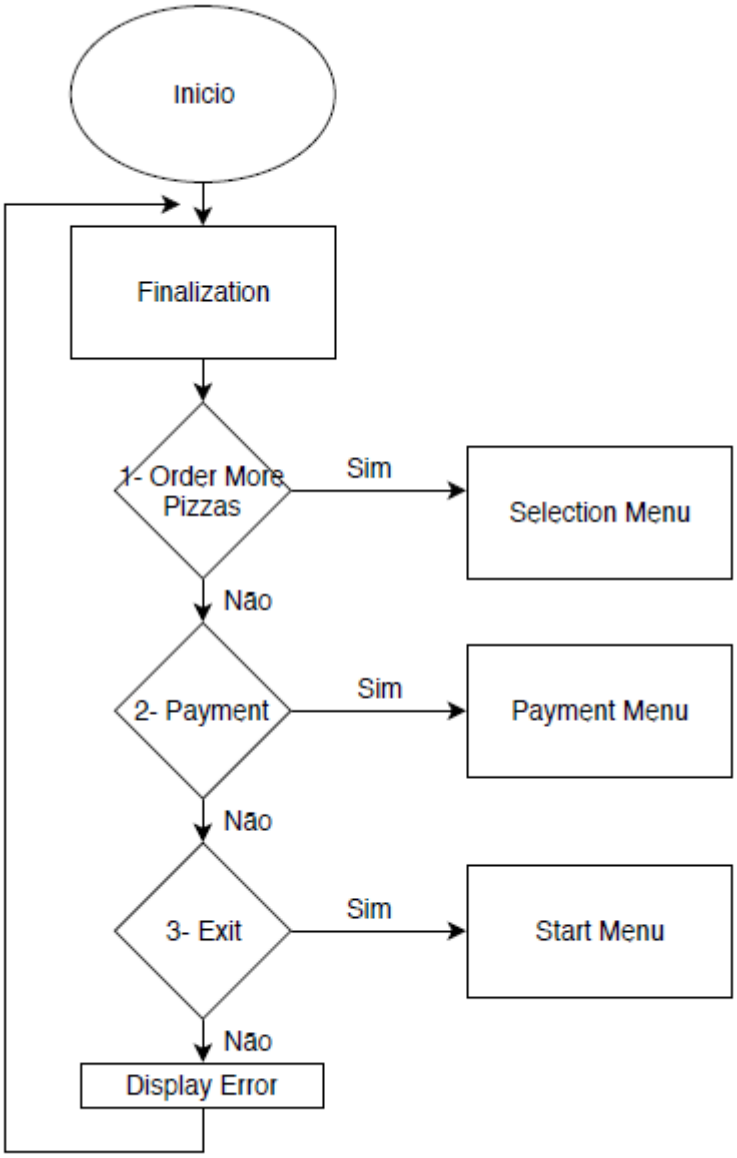


Figura 13 – Fluxograma Finalization.

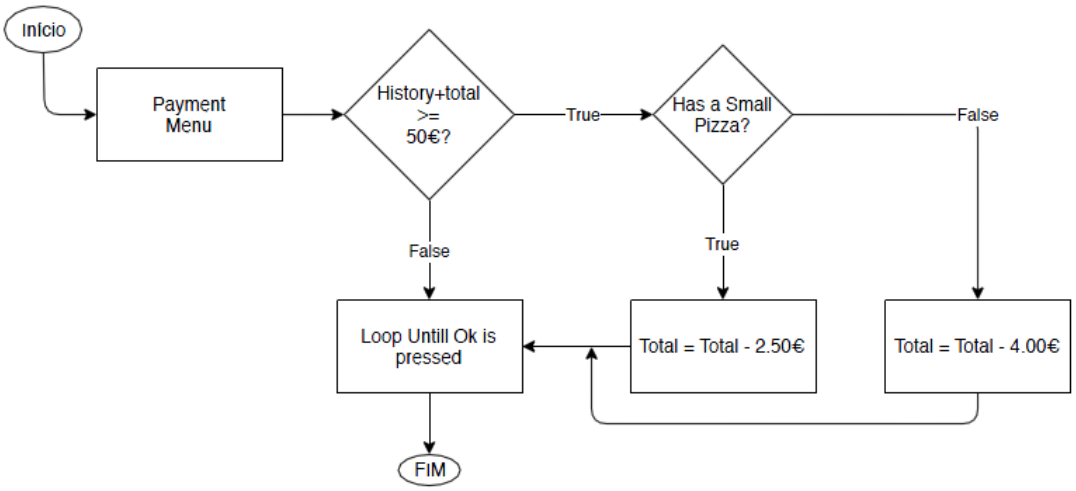


Figura 14 – Fluxograma Payment Menu.

Código Assembly:

```
;-----------------  Periféricos    ----------------------------------
-----------------
POWER                       EQU 0B0H    ;'Begins' the Program
NR_SEL                      EQU 0C0H    ;Input every choise
OK                          EQU 0D0H    ;Inputs 'OK' to end the choise
USERB                       EQU 0F3H    ;Beginning of the user input
segment
USERE                       EQU 0FAH    ;End of User input segment
PASSB                       EQU 0113H   ;Beginning Password input
segment
PASSE                       EQU 011AH   ;End of password input segment
;-----------------   Display     ----------------------------------
--------------
Display                     EQU 20H     ;Memory position to start the
Display
Display_End                 EQU 8FH     ;Memory position to end the
Display
;-----------------   Constantes  ----------------------------------
--------------
EmptyCharacter              EQU 20H     ;To place the " " character
NR_CHARACTERS               EQU 4       ;Número de caracters
SMALLPIZZA                  EQU 5       ;Price small pizza
LARGEPIZZA                  EQU 8       ;Price large pizza
DISCOUNT                    EQU 47H     ;To place the discount ammount
on the payment display
PRICE                       EQU 67H     ;To place the final ordder
price
USERSCREEN                  EQU 43H     ;Saves the position where the
inputed user will apear
PASSSCREEN                  EQU 63H     ;Saves the position place '*'
where the password should be
BDDATACHECK                 EQU 3000H   ;Saves inputed username and
password to check its existance in the DataBase
BD                          EQU 3010H   ;Position of the actual
DataBase
InputUnderScore             EQU 5FH     ;Saves the character "_" to
use as a input Checker
StackPointer                EQU 2FF0H   ;Position for the StackPointer
;-------------------------------------------------------------------
--------------
Place 2000H                             ;Layout from all possible
screens
StartMenu:
  String "    Welcome     "
  String "                "
  String "1 - Login       "
  String "2 - Registration"
  String "                "
  String "                "
  String "                "
LoginMenu:
  String "      Login     "
  String "   Username:    "
  String "                "
  String "   Password:    "
  String "                "
  String "3 - Back        "
```

```
  String "                "
RegistrationMenu:
  String "    Registo     "
  String "   Username:    "
  String "                "
  String "   Password:    "
  String "                "
  String "3 - Back        "
  String "                "
ErrorMenu:
  String "                "
  String "                "
  String "  Something's   "
  String "     Wrong      "
  String "                "
  String "                "
  String "                "
PizzOMenu:
  String "     PizzO      "
  String "     Online     "
  String "                "
  String "1 - Menu Pizzas "
  String "                "
  String "3 -    Exit     "
  String "                "
FinalizationMenu:
    String "                  "
    String "                  "
  String "1 - Order More  "
  String "2 - Payment     "
  String "3 -    Exit     "
    String "                  "
    String "                  "
SelectionMenu:
    String "     Pizzas     "
  String "1 - PapaManuel  "
  String "2 - Marguerita  "
  String "3 - 4 Queijos   "
  String "4 - Mussarela   "
  String "5 - Napolitana  "
    String "                  "
PaymentMenu:
  String "    Payment     "
  String "Discount:       "
  String "         .  EUR  "
  String "Bill:           "
  String "         .  EUR  "
  String "                "
  String "   Press OK     "
SizeMenu:
  String "      Size      "
    String "                  "
    String "1 -  Small      "
  String "2 -  Large      "
  String "3 -  Back       "
    String "                  "
    String "                  "

;----------------------Screen Frame ---------------------------------
----------
```

```
place 10H
  String "================"
place 90H
  String "================"

;-----------------Inputs-----------------------------------------------
---------
place 0B0H
  String "_ <- Power ON    "
  String "_ <- Option      "
  String "_ <- OK          "
  String "   Username:     "
  String "   ________      "
  String "   Password:     "
  String "   ________      "

;------------- DataBase ---------------------------------------
place 3010H ;Placing an example user account
String "qwerty__qwerty__"


;---------------------------Instructions----------------------------
place 0000H
  Begin:
    MOV R0, Beginning
    JMP R0                          ;Jumps to the Beginning of the
code

place 6000H
Beginning:
  MOV R9, 2DH
  MOV R10, 3020H
  MOV [R10], R9
  CALL CleanDisplay
  CALL CleanPerif
  CALL IS_ON
  ON:
    MOV SP, StackPointer            ;Sets the position of the
StackPointer
    MOV R2, StartMenu              ;Sets the memory position of the
screen to sho
    CALL ShowDisplay               ;Turns on the screen with starting
Menu
    CALL CleanPerif                ;Cleans all the priferics
    MOV R8, 0
    MOV R9, 0
    MOV R10, 0
  FirstMenu:
    MOV R0, OK                     ;Places the priferic memory position
in R0
    MOVB R0, [R0]                  ;Gets the actual value of the
priferic "OK"
    MOV R2, 30H
    SUB R0, R2
    CMP R0, 1
    JNZ FirstMenu                  ;Keeps the FirstMenu on display
untill the "OK" is pressed
    MOV R0, NR_SEL                 ;Sets the memory position for the
NR_SEL priferic
    MOVB R0, [R0]                  ;Reads the input NR_SEL
    SUB R0, R2
```

```
    CMP R0, 1                       ;Switch case for the FirstMenu
NR_SELs
    JZ DisplayLogin                 ;If no NR_SEL selected, Display
Error
    CMP R0, 2                       ;1 for Login
    JZ DisplayRegistrationAUX       ;1 for Registration
    CALL DisplayError
    JMP ON
;------------- POWER ON the Display ---------------------------------
--------
IS_ON:
  MOV R0, POWER                 ;Move the priferic memory code to R0
  MOVB R1, [R0]                 ;Gets the value of the Memory
  MOV R2, 30H                   ;This will be used to convert ASCII
String to Hexadecimal
  SUB R1, R2
  CMP R1, 1                     ;Checks if the Machine is ON
  JNZ IS_ON                     ;Repeats untill the Priferic is set to
1 (ON)s
  RET

;------------------------- Cleans the Scrren ------------------------
----------
CleanDisplay:
  PUSH R0
  PUSH R1
  PUSH R2
  MOV R0, Display               ;Saves the memory position where the
Display begins
  MOV R1, Display_End           ;Saves where the Display ends
  MOV R2, EmptyCharacter        ;Saves the EmptyCharacter 20H wich
is a " "
  CleaningCycle:                ;Cleans the menu from Display
position
    MOVB [R0], R2
    ADD R0, 1
    CMP R0, R1
    JLE CleaningCycle           ;Keeps the loop untill all the
Display is filled with " "
  POP R2
  POP R1
  POP R0
  RET
;--------------------------- Cleans Every Input ---------------------
---------
                    ;Resets every input to its initial value
CleanPerif:
  PUSH R3
  PUSH R2
  PUSH R1
  PUSH R0
  MOV R0, POWER
  MOV R1, NR_SEL
  MOV R2, OK
  MOV R3, InputUnderScore
  MOVB [R0], R3
  MOVB [R1], R3
  MOVB [R2], R3
  MOV R0, USERB
  MOV R1, USERE
  CleaningUser:               ;Cleans the menu from USER priferic
```

```
    MOVB [R0], R3
    ADD R0, 1
    CMP R0, R1
    JLE CleaningUser
    MOV R0, PASSB
    MOV R1, PASSE
  CleaningPass:                ;Cleans the menu from PASS priferic
    MOVB [R0], R3
    ADD R0, 1
    CMP R0, R1
    JLE CleaningPass
  POP R0
  POP R1
  POP R2
  POP R3
  RET
;-------------------------------Auxiliary Jumps----------------------
--
DisplayRegistrationAUX:
  CALL DisplayRegistration
  JMP ON
;-------------------------------Prints out the Display---------------
---------
ShowDisplay:                       ;Sets the Display for the menu that
needs to be shown
  PUSH R3
  PUSH R2
  PUSH R1
  PUSH R0
  MOV R0, Display
  MOV R1, Display_End
  Cycle:                           ;Places the Display char by char
    MOV R3, [R2]
    MOV [R0], R3
    ADD R2, 2
    ADD R0, 2
    CMP R0, R1
    JLE Cycle
  POP R0
  POP R1
  POP R2
  POP R3
  RET
;----------------------------- Displays an Error --------------------
----------
DisplayError:           ;Places the Error mensage on the Display
  PUSH R0
  PUSH R2
  MOV R2, ErrorMenu       ;Gets the position where the ErroMenu is
saved
  CALL ShowDisplay
  CALL CleanPerif
  DisplayErrorLoop:     ;Keeps showing the Error untill "OK" is
pressed
    MOV R0, OK
    MOVB R0,[R0]
    MOV R2, 30H
    SUB R0, R2
    CMP R0, 1
    JNZ DisplayErrorLoop  ;Keeps the loop going
  POP R2
```

```
  POP R0
  RET


;-------------------------- Menu Login ------------------------------
----------
DisplayLogin:                   ;Displays a Menu for Login
  PUSH R4
  PUSH R3
  PUSH R2
  PUSH R1
  PUSH R0
  LoginBeginning:
    MOV R2, LoginMenu             ;Sets the position of the Login
screen
    CALL ShowDisplay              ;Displays the Screen
    CALL CleanPerif               ;Cleans all Perif
    MOV R3, OK
  LoginCycle:
    CALL GetUser
    CALL GetPass
    MOVB R0, [R3]               ;Places in R0 the value of the input
OK
    MOV R1, 30H
    SUB R0, R1                  ;Converts the input from a string
number to an integer number
    CMP R0, 1
    JNE LoginCycle                ;Keeps looping untill OK is
"pressed"
    MOV R4, NR_SEL
    MOVB R0, [R4]
    SUB R0, R1
    CMP R0, 3
    JZ CancelLogin
  CALL BDCONSULT                ;Calls the consultation to the
DataBase
  CMP R9, 0
  JZ LoginBeginning
  CancelLogin:
  POP R0
  POP R1
  POP R2
  POP R3
  POP R4
  JMP ON

;------------------- Recieve the Data form user input ---------------
----------
      ;Will take care of reading the User data form the priferic
GetUser:
  PUSH R0
  PUSH R1
  PUSH R4
  PUSH R5
  PUSH R7
  MOV R0, USERB               ;Gets where the user input starts
  MOV R4, USERE               ;Gets where the user input ends
  MOV R2, USERSCREEN          ;Gets where in the screen is the user
space
  ADD R4, 1                   ;Adds 1 to the password end position to
make comparation easyer
```

```
  MOV R7, InputUnderScore      ;This has the ASCII code for "_"
  MOV R5, BDDATACHECK          ;Memory for a data comparation
  GetUserLoop:
  MOVB R1, [R0]
  CMP R1, R7
  JZ NEXTCHARU                 ;Jumps if he finds "_" meaning that user
has not inserted nothing
  MOVB [R2], R1
  NEXTCHARU:
    MOVB [R5], R1              ;Copy the input of username to the
DBDATACHECK
    ADD R0, 1
    ADD R2, 1
    ADD R5, 1
    CMP R0, R4
    JNZ GetUserLoop            ;Keeps the loop going untill we have all
characters
  POP R7
  POP R5
  POP R4
  POP R1
  POP R0
  RET

;------------------- Recieve the Pass form user input ---------------
----------
            ;Will take care of reading the password priferic
GetPass:
  PUSH R0
  PUSH R1
  PUSH R2
  PUSH R4
  PUSH R5
  PUSH R7
  PUSH R8
  MOV R5, BDDATACHECK          ;Memory for a data comparation
  MOV R0, 8
  ADD R5, R0
  MOV R0, PASSB                ;Gets where the password input starts
  MOV R4, PASSE                ;Gets where the password input ends
  ADD R4, 1                    ;Adds 1 to the password end position to
make comparation easyer
  MOV R2, PASSSCREEN           ;Gets where in the screen the password
should be
  MOV R8, 2AH                  ;Saves the '*' to be placed as password
in the screen
  MOV R7, InputUnderScore          ;This is the ASCII code for "_"
  GetPassLoop:
    MOVB R1, [R0]
    CMP R1, R7
    JZ NEXTCHARP
    MOVB [R2], R8
  NEXTCHARP:                 ;Skips the screen input if there's nothing
in the password
    MOVB [R5], R1
    ADD R0, 1
    ADD R2, 1
    ADD R5, 1
    CMP R0, R4
    JNZ GetPassLoop          ;Keeps the loop going untill we have all
characters
```

```
  POP R8
  POP R7
  POP R5
  POP R4
  POP R2
  POP R1
  POP R0
  RET

;----------Search the DataBase for the credentials and attempts to
login--------
BDCONSULT:                    ;Consults the DataBase
  PUSH R0
  PUSH R1
  PUSH R2
  PUSH R3
  PUSH R4
  PUSH R5
  PUSH R6
  MOV R0, BDDATACHECK        ;Gets the initial position for the string
it should compare
  MOV R1, BD                 ;Gets the first position of the DataBase
  MOV R4, BD                 ;Gets the first position of the DataBase
to compare afterwards
  MOV R5, 0                  ;Places a 0 on R5 to use as a counter
  TRYLOGIN:
    MOVB R2, [R1]              ;Gets the 1st character from the
DataBase to compare
    CMP R2, 0                  ;Compares with '0' if it is zero we are
at the end of the DataBase
    JZ BadLogin                ;Leaves the DataBase
    MOVB R3, [R0]              ;Gets 1st character to compare with the
one from the DataBase
    CMP R2, R3
    JNZ NEXTUSER               ;If the 1st not equal should jump to the
NEXT USER
    ADD R0, 1
    ADD R1, 1
    ADD R5, 1
    MOV R6, 16                 ;Sets the end of comparation this is
here so I can use R6 again
    CMP R5, R6
    JZ LOGGED                  ;If we get here it means that all 8
bytes compared mached, the login is sucessfull
    JMP TRYLOGIN
  NEXTUSER:
    MOV R0, BDDATACHECK        ;Resets the initial position for the
comparation String
    MOV R6, 20H                ;Saves 20 to be incremented in memory
    ADD R4, R6                 ;Increments the memory moves to the Next
place in the DataBase
    MOV R1, R4                 ;Gets the new Data to compare
    JMP TRYLOGIN               ;Jums to try login with the new Data
  BadLogin:
    MOV R2, ErrorMenu
    CALL ShowDisplay
    CALL CleanPerif
    BadLoginLoop:
      MOV R3, OK
      MOVB R1, [R3]
      MOV R2, 30H
```

```
      SUB R1, R2
      CMP R1, 1
      JNZ BadLoginLoop
      MOV R9, 0
      JMP ENDLOGIN
  LOGGED:
    MOV R9, R4              ;Places the memory position from the
macthed user so we can use it later on the program
    CALL DisplayPizzOMenuAUX
  ENDLOGIN:
  POP R6
  POP R5
  POP R4
  POP R3
  POP R2
  POP R1
  POP R0
  RET

;----------------------Displays the RegistrationMenu----------------
----------
DisplayRegistration:          ;Behaviour of the Menu Registration
  PUSH R0
  PUSH R2
  PUSH R3
  RegisterStart:
    MOV R3, BDDATACHECK         ;Memory for a data comparation
    MOV R2, RegistrationMenu    ;Screen to Display
    CALL ShowDisplay
    CALL CleanPerif
  RegistrationCycle:
    CALL GetUser
    CALL GetPass
    MOV R0, OK
    MOVB R0, [R0]
    MOV R2, 30H
    SUB R0, R2
    CMP R0, 1
    JNZ RegistrationCycle
  Call Register               ;Will attempt to save into the DataBase
if theres no user with the same name
  CMP R9, 0
  JZ RegisterStart
  POP R5
  POP R2
  POP R0
  RET

;---------------Checks if the password has at least 3 characters------
----------
PassCharacterCounter:
  PUSH R0
  PUSH R1
  PUSH R2
  PUSH R3
  MOV R6, 0
  MOV R3, InputUnderScore
  MOV R0, BDDATACHECK
  MOV R1, 8
  ADD R0, R1
  MOV R1, 0
```

```
  Counting:
    MOVB R2, [R0]
    CMP R2, R3
    JZ PassBad          ;Did not meet the requirements, has less than 3
characters
    ADD R1, 1
    CMP R1, 3
    JZ PassGood         ;Password is good enough
    JMP Counting        ;Keeps the loop untill it breaks
  PassBad:
    MOV R6, 1
  PassGood:
  POP R3
  POP R2
  POP R1
  POP R0
  RET

DisplayPizzOMenuAUX:
  CALL DisplayPizzOMenu
  RET
;-------------------------- Consults the DataBase --------------------
----------
;         if the username is not taken and the password meets the
requirements
;                   records a new user into the DataBase
Register:
  PUSH R0
  PUSH R1
  PUSH R2
  PUSH R3
  PUSH R4
  PUSH R5
  PUSH R6
  CALL PassCharacterCounter   ;Will check if the password has 3
characters at least
  CMP R6, 1
  JZ UserFound
  MOV R0, BDDATACHECK   ;Checker
  MOV R1, BD
  MOV R2, BD
  MOV R3, 0             ;Counter
  MOV R4, 8             ;User length
  UserCheckLoop:
    MOVB R5, [R0]          ;First character
    MOVB R6, [R1]
    CMP R6, 0
    JZ Registering        ;The DataBase
    CMP R5, R6
    JNZ NextRegisterLoop  ;This user is not the same
    ADD R0, 1             ;increments checker character
    ADD R1, 1
    ADD R3, 1
    CMP R3, R4
    JZ UserFound          ;User already exists cannot Register
    JMP UserCheckLoop     ;Keeps looping untill one of the breaks
verify
  NextRegisterLoop:
    MOV R0, BDDATACHECK
    MOV R3, 0
    MOV R6, 20H
```

```
    ADD R2, R6            ; increments 20 to get to the next spot in
the data base
    MOV R1, R2
    JMP UserCheckLoop
  Registering:
    MOV R0, BDDATACHECK
    MOV R3, 0
    MOV R4, 16
    MOV R9, R2
    RegisteringLoop:
      MOVB R1, [R0]
      MOVB [R2], R1
      ADD R0, 1
      ADD R2, 1
      ADD R3, 1
      CMP R3, R4
      JZ RegisterEndSucess
      JMP RegisteringLoop
  UserFound:
    MOV R2, ErrorMenu
    CALL ShowDisplay
    CALL CleanPerif
    MOV R3, OK
    MOV R9, 0
    UserFoundLoop:
      MOV R2, 20H
      MOV R1, [R3]
      SUB R1, R2
      CMP R1, 1
      JNZ UserFoundLoop
      JMP RegisterEnd
  RegisterEndSucess:
    ;MOV R1, 3030H
    ;MOV [R2], R1
    ;ADD R2, 2
    ;MOV [R2], R1
    CALL DisplayPizzOMenu
  RegisterEnd:
  POP R6
  POP R5
  POP R4
  POP R3
  POP R2
  POP R1
  POP R0
  RET
;------------------PizzOMenu-------------------------------------

DisplayPizzOMenu:
  PUSH R0
  PUSH R1
  PUSH R2
  MOV R2, PizzOMenu             ;Plaes PizzOMenu in R2 to be displayed
when needed
  PizzOBeginning:                ;Starts to show the menu and clean
Inputs
    CALL ShowDisplay            ;We'll be back here if something goes
wrong
    CALL CleanPerif
    MOV R1, 30H
    PizzOLoop:
```

```
      MOV R0, OK
      MOVB R0, [R0]
      SUB R0, R1
      CMP R0, 1                ;Checks if OK is pressed
      JNZ PizzOLoop            ;Loops Untill the OK is pressed
    MOV R0, NR_SEL
    MOVB R0, [R0]
    SUB R0, R1
    CMP R0, 1
    JNZ NotPizzO1
    CALL DisplaySelectionMenu   ;Calls the Pizza SelectionMenu
    NotPizzO1:
    CMP R0, 3
    JZ PizzOLoopEnd             ;Goes back to the main meny
    Call DisplayError
    JMP PizzOBeginning           ;We get here if we did something
wrong on this menu and pressed ok on error
  PizzOLoopEnd:
  POP R2
  POP R1
  POP R0
  RET

  ;-------------------------------SelectionMenu-----------------------
---------
              ;Selection Menu, allows to choose wich pizza we want
  DisplaySelectionMenu:
  PUSH R0
  PUSH R1
  PUSH R2
  MOV R2, SelectionMenu       ;Places the SelectionMenu in R2 to be
displayed
    SelectionBeginning:
      CALL ShowDisplay
      CALL CleanPerif
      MOV R1, 30H
      SelectionLoop:
        MOV R0, OK
        MOVB R0, [R0]
        SUB R0, R1
        CMP R0, 1
        JNZ SelectionLoop   ;Keeps looping untill OK is pressed
      MOV R0, NR_SEL
      MOVB R0, [R0]
      SUB R0, R1
      CMP R0, 1
      JNZ NotOpt1
      CALL DisplaySizeMenu              ;Pizza PapaManuel Choosen
      JMP SelectionBeginning
      NotOpt1:
        CMP R0, 2
        JNZ NotOpt2
        CALL DisplaySizeMenu            ;Pizza Marguerita Choosen
        JMP SelectionBeginning
        NotOpt2:
          CMP R0, 3
          JNZ NotOpt3
          CALL DisplaySizeMenu          ;Pizza 4 Queijos Choosen
          JMP SelectionBeginning
          NotOpt3:
            CMP R0, 4
```

```
            JNZ NotOpt4
            CALL DisplaySizeMenu        ;Pizza Mussarela Choosen
            JMP SelectionBeginning
            NotOpt4:
              CMP R0, 5
              JNZ NotOpt5
              CALL DisplaySizeMenu     ;Pizza Napolitana Choosen
              JMP SelectionBeginning
              NotOpt5:      ;Else
                CALL DisplayError      ;Displays an Error nothing was
selected
                JMP SelectionBeginning  ;Restarts the menu from the
beginning
      ExitSelectionMenu:
      POP R2
      POP R1
      POP R0
      RET
;-------------------------------SizeMenu-------------------------------
--
                  ;Here we shall have the option between
                        ;Small  or Large
DisplaySizeMenu:
  PUSH R0
  PUSH R1
  PUSH R2
  MOV R1, 30H
  SizeMenuBegin:
    MOV R2, SizeMenu
    CALL ShowDisplay
    CALL CleanPerif
    SizeLoop:
      MOV R0, OK
      MOVB R0, [R0]
      SUB R0, R1
      CMP R0, 1
      JNZ SizeLoop                  ;Waits for OK to be Pressed
    MOV R0, NR_SEL
    MOVB R0, [R0]
    SUB R0, R1
    CMP R0, 1                       ;Small
    JNZ SizeMenu2
    ADD R8, 1                      ;R8 Saves the number of small
pizzas in the current order
    CALL DisplayFinalizationMenu   ;Calls the finalization Menu
    JMP SizeEnd
    SizeMenu2:
      CMP R0, 2                     ;Large
      JNZ SizeMenu3
      ADD R10, 1                   ;R10 Saves the number of Large
pizzas in the current Order
      CALL DisplayFinalizationMenu ;Calls the finalization Menu
      JMP SizeEnd
      SizeMenu3:
        CMP R0, 3                   ;Back
        JNZ SizeError
        JMP SizeEnd
    SizeError:                     ;Not a valid option
      CALL DisplayError
      JMP SizeMenuBegin
  SizeEnd:
```

```
  POP R2
  POP R1
  POP R0
  RET
;--------------------------------FinalizationMenu----------------------
---------
DisplayFinalizationMenu:
PUSH R0
PUSH R1
PUSH R2
MOV R1, 30H
FinalizationMenuBegin:
MOV R2, FinalizationMenu
CALL ShowDisplay
CALL CleanPerif
  FinalizationLoop:
    MOV R0, OK
    MOVB R0, [R0]
    SUB R0, R1
    CMP R0, 1
    JNZ FinalizationLoop    ;Waits for OK to be Pressed
  MOV R0, NR_SEL
  MOVB R0, [R0]
  SUB R0, R1
  CMP R0, 1                 ;Order More
  JNZ Finalization2
  JMP FinalizationEnd
  Finalization2:
    CMP R0, 2               ;Payment
    JNZ Finalization3
    CALL DisplayPaymentMenu
    Finalization3:
      CMP R0, 3
      JNZ Finalization4
      JMP ON                ;Exit
      Finalization4:
        CALL DisplayError
        JMP FinalizationMenuBegin
FinalizationEnd:
POP R2
POP R1
POP R0
RET


;--------------------------------PaymentMenu--------------------------
--------

DisplayPaymentMenu:
PUSH R0
PUSH R1
PUSH R2
PUSH R3
MOV R1, 30H
PaymentMenuBegin:
  MOV R2, PaymentMenu
  CALL ShowDisplay
  CALL CleanPerif
  CALL PaymentCalculation
  PaymentLoop:
    MOV R0, OK
```

```
    MOVB R0, [R0]
    SUB R0, R1
    CMP R0, 1
    JNZ PaymentLoop
  JMP ON
POP R3
POP R2
POP R1
POP R0
RET

PaymentCalculation:
PUSH R0
PUSH R1
PUSH R2                     ;History Total
PUSH R3
PUSH R4                     ;Total
MOV R3, R9                  ;Gets the user memory position
MOV R1, 16
ADD R3, R1                  ;Gets the memory position for the payment
history
MOV R2, [R3]
MOV R1, SMALLPIZZA          ;Gets the price for a Small pizza
MUL R1, R8                  ;R8 cointains how many small pizzas user is
buying
ADD R2, R1                  ;Updates history
MOV R4, R1                  ;Starts counting the Total
MOV R1, LARGEPIZZA          ;Gets the price for the Large pizza
MUL R1, R10                 ;R10 contains the amount of large pizzas
user is buying
ADD R2, R1                  ;Updates history
ADD R4, R1                  ;Updates Total
MOV [R3], R2                ;Saves history
MOV R1, 50
CMP R2, R1                  ;Checks if the user is able to get a
discount
JLT NoDiscount              ;Historic plus actual dont reach 50 EUR no
discount this time
SUB R2, R1                  ;Removes 50 eur from historic
MOV [R3], R2                ;Updates DataBase
CMP R8, 0
JZ LargeDiscount            ;If the user has the right to a discount and
is buying atleast one small pizza
SUB R4, 3                   ;Discount = Total - 3EUR + 0.50EUR
MOV R2, PRICE               ;Sets the decimal number to display total to
pay
MOV R0, R4                  ;Moves total to R0
CALL CONVERTER              ;Places total on the display
ADD R2, 2                   ;\
MOV R3, 35H                 ;|
MOVB [R2], R3               ; > Adds 50 after the comma
ADD R2, 1                   ;|
MOV R3, 30H                 ;|
MOVB [R2], R3               ;/
MOV R2, DISCOUNT            ;Sets the decimal number to display discount
MOV R0, 2
CALL CONVERTER
ADD R2, 2
MOV R3, 35H
MOVB [R2], R3
ADD R2,1
```

```
MOV R3, 30H
MOVB[R2], R3
JMP CalculationEnd      ;If the user has the right to a discount and
is buying only large pizzas
LargeDiscount:
SUB R4, 4
MOV R2, PRICE
MOV R0, R4
CALL CONVERTER
ADD R2, 2
MOV R3,30H
MOVB [R2], R3
ADD R2, 1
MOVB [R2], R3
MOV R2, DISCOUNT
MOV R0, 4H
CALL CONVERTER
ADD R2, 2
MOVB [R2], R3
ADD R2, 1
MOVB [R2], R3
JMP CalculationEnd
  NoDiscount:
    MOV R3, 30H
    MOV R2, PRICE
    MOV R0, R4
    CALL CONVERTER
    ADD R2, 2
    MOVB [R2], R3
    ADD R2, 1
    MOVB [R2], R3
    MOV R0, 0
    MOV R2, DISCOUNT
    CALL CONVERTER
    ADD R2, 2
    MOVB[R2], R3
    ADD R2, 1
    MOVB [R2], R3
CalculationEnd:
POP R4
POP R3
POP R2
POP R1
POP R0
RET


CONVERTER:  ;ROTINA DE CONVERSÃO NUM->CARACTER
  PUSH R0
  PUSH R1
  PUSH R2
  PUSH R3
  PUSH R4
  PUSH R5
  MOV R1, R0
  MOV R0, 10
  MOV R3, 0                         ;Starts counter at zero
  NextChar:
    MOV R4, R1
    MOD R4, R0                      ;Gets rest of the division by 10
    DIV R1, R0                      ;Divides by 10
```

```
    MOV R5, 30H
    ADD R5, R4                      ;Converts to ASCII
    MOV R4, R2
    MOVB [R4], R5                   ;Displays the converted number
    SUB R2, 1                       ;Updates the Display value
    ADD R3, 1                       ;Counter
    CMP R3, 2
    CMP R1, 0                       ;If zero the conversion is over
    JNZ NextChar                    ;Loops untill all nubers are
placed
  POP R5
  POP R4
  POP R3
  POP R2
  POP R1
  POP R0
  RET
```