Faculdade de Ciências Exatas e da Engenharia

Licenciatura em Engenharia Informático

Arquitetura de Computadores

2021/2022

PizzO – Online Pizzaria

Docentes: Dionísio Barros | Pedro Camacho | Nuno Ferreira | Sofia Inácio

Aluno:

Pedro Jaques, nº 2046919

Funchal, 29 de junho de 2022

Índice

1. Introdução

O seguinte relatório do segundo projeto prático de Arquitetura de Computadores, tem como objetivo fundamentar como foi elaborado todo o projeto tendo em conta os objetivos e requisitos definidos previamente no enunciado de modo a obter um produto final funcional.

Para o desenvolvimento deste projeto foi utilizada a linguagem A*ssemby* para o processador PEPE, posteriormente compilada e transformada para código máquina através de um simulador JAVA, que permitiu efetuar os devidos testes ao programa.

2. Objetivos

Este projeto teve como objetivo o desenvolvimento de um programa para fazer o controlo dos pedidos de pizzas online, a uma pizzaria que efetua entregas. A interface gráfica deveria simular a página web onde os clientes efetuam os seus pedidos online. Este programa guardaria os valores gastos para cada utilizador de modo a quando um certo valor fosse atingido este pudesse usufruir de um desconto.

Outro dos requisitos deste programa era que seria necessário o utilizador ter o login efetuado no sistema para poder fazer um pedido, de modo que assim o valor gasto por si pudesse ser guardado em um histórico do valor total que este já tinha gasto em pizzas, para poder ser aplicado um desconto quando um determinado valor fosse atingido (50€).

A interface gráfica do utilizador foi feita através de um *display* com dimensões 7x16 (7 linhas de 16 carateres – bytes), com dois botões de controlo, o "**OK_Button**" e "**NR_SEL_Button**" que permitiam ao utilizador escolher uma de entre as opções apresentadas no *display* e confirmar essa opção. Para a criação de um utilizador ou fazer *login* no programa, tinha também um *input* para o *username* e outro para a *password*.

A utilização da linguagem *Assembly* foi também um dos objetivos deste projeto. Sendo esta uma linguagem de baixo nível, foi necessária uma gestão cuidadosa da memória e dos registos de modo que não existisse sobreposição de dados e instruções, sendo que isto levaria a uma inconsistência e a um errado funcionamento do programa.

3. Discussão de Resultados

Como referido anteriormente, é necessário ter uma gestão equilibrada da memória para não ocorrer falhas no programa. Assim sendo, no anexo A, é apresentado um mapa da utilização da memória.

Relativamente aos periféricos de entrada, estes foram colocados da seguinte forma na memória:

- Do endereço 0150H até 0157H: Periférico de entrada "***Username"***;
- Endereço 0160H até 0167H: Periférico de entrada "***Password"***;
- Endereço 0170H: Periférico de entrada "**OK_Button**";
- Endereço 0172H: Periférico de entrada "**NR_SEL_Button**".

Em relação aos periféricos do U*sername* e da *Password* é necessário ter alguns cuidados, nomeadamente:

- O *username* introduzido tem de conter no mínimo 1 e no máximo 8 carateres;
- A *password* introduzida tem de conter no mínimo 3 e no máximo de 8 carateres.

3.1. Start Screen e Login Menu

Para inicializar o programa e ter acesso ao "*Start Screen*" o utilizador tem de ativar a entrada "**OK_Button**".

O programa inicia-se apresentando uma mensagem de boas-vindas ao utilizador. Após a validação do "**OK_Button**", é encaminhado para o "*Login Menu*" onde tem à sua disposição 3 opções: "*1. New Account*", "*2. Login*", "*3. Exit*". A escolha destas opções é feita através do periférico de entrada "**NR_SEL_Button**", tendo posteriormente de ser validado pelo periférico de entrada "**OK_Button**".

3.1.1. Register Form

Para criar um novo utilizador é necessário primeiramente verificar se ainda existe espaço disponível na base de dados para a introdução de um novo registo. Após feita esta verificação pela rotina "**CheckUsersInDBRoutine**" (anexo B).

Caso a base de dados esteja completa, é apresentada uma mensagem ao utilizador a informar que não é possível criar mais nenhum registo. A base de dados deste programa, por questões de simplicidade, apenas pode conter 5 utilizadores.

Caso contrário, são feitas as próximas verificações, sendo estas efetuadas nos periféricos de entrada do *Username* e da *Password*, nomeadamente:

- Verifica se o *username* introduzido já existe na base de dados;
- Verifica se o utilizador deixou algum dos campos em branco (*username* ou *password*);
- Verifica se a password tem no mínimo 3 carateres.

Para verificar se o *username* introduzido é igual a algum já existente na base de dados, a rotina criada para essa verificação foi a "**CheckUsername**", que faz a comparação de cada carater introduzido no periférico de entrada *Username* com cada carater da posição do *username* na base de dados e contabiliza quantos são iguais. Se este valor for igual a 8 significa que já existe um utilizador na base de dados com o mesmo *username* e é apresentada ao utilizador uma mensagem de erro.

Para verificar se algum dos campos foi deixado em branco, foi implementado a rotina "**CheckFormFieldsRoutine**", que verifica quantos carateres tem no periférico de entrada *Username* e no *Password.* Caso não apresente nenhum carater em algum dos dois, é apresentada uma mensagem de erro.

Para verificar se a password contém o número mínimo de carateres obrigatórios, a rotina implementada foi a "**CheckFormFieldsRoutine**", que verifica quantos carateres tem a *password* inserida e contabiliza os mesmos. Caso este valor seja inferior a 3, é apresentada uma mensagem de erro.

Após realizadas estas verificações, e se não houver erros, é apresentado no *display* que o registo foi efetuado com sucesso e ao clicar no "**OK_Button**", o utilizador terá o seu registo criado na base de dados e será encaminhado para o menu onde pode efetuar o pedido de pizzas.

3.1.2. Login Form

Para efetuar o *login*, a primeira verificação que é realizada é se o *username* introduzido existe na base de dados, de modo a saber se o utilizador já tem uma conta em seu nome. Esta é feito através da rotina "**CheckUsername**", onde é realizada uma comparação e contabilização de carateres iguais entre o *username* introduzido no periférico de entrada e os *usernames* existentes na base de dados. Se for retornado o valor 8 significa que existe um registo com esse *username*, logo pode avançar para a verificação da *password*, caso contrário significa que ainda não existe nenhum registo com esse *username*, sendo necessário proceder à criação de um novo registo.

Para a verificação da *password*, foi implementada a rotina "**CheckPassword**", em que o processo é idêntico à rotina referida anteriormente.

Caso esteja tudo correto, é apresentado no *display* que o login foi efetuado com sucesso e ao clicar no "**OK_Button**", o utilizador será encaminhado para o menu onde pode efetuar o pedido de pizzas.

3.2. Main Menu

Após o utilizador ter efetuado o *login*, será apresentado um menu com duas opções: "1. Order Pizza", "2. Logout". Do mesmo modo que nos menus anteriores, para proceder à escolha das opções, o utilizador terá de selecionar a opção que pretende e pressionar o botão "**OK_Button**" para que a sua escolha seja validada.

3.2.1. Pizza Menu, Pizza Size Menu e Payment Menu

Estes são os menus onde o utilizador tem a liberdade de escolher qual a pizza que pretende e qual o tamanho da pizza selecionada. Existem 2 tamanhos à escolha do utilizador, pequena ou grande, sendo o seu preço 5€ e 9€, respetivamente.

Após ter selecionado o seu pedido, terá um menu onde pode adicionar mais pizzas ao pedido ou efetuar o pagamento.

Nesta fase o valor do tamanho da pizza selecionado será adicionado no histórico de compras do utilizador, sendo este processo efetuado pela rotina

"**UpdateUserHistoricRoutine**", que faz a adição do valor do tamanho da pizza ao valor já existente na base de dados na posição do histórico do utilizador. O programa sabe onde se encontra esta posição, pois quando é efetuado o login guarda num registo o valor do endereço deste histórico.

3.3. Rotinas implementadas

As rotinas seguintes foram criadas para que o código ficasse mais organizado e mais acessível para a sua manutenção. Algumas rotinas já foram explicadas no decorrer do relatório, sendo assim, apenas serão explicadas as que ainda não foram referidas, tais como:

- "**ShowDisplayRoutine**" – Mostra no *display* o menu pretendido;
- "**CleanDisplayRoutine**" – Limpa o *display* utilizando o carater 20H;
- "**CleanPeripheralsRoutine**" – Limpa os periféricos "**OK_Button**" e "**NR_SEL_Button**";
- "**CleanUserPassRoutine**" – Limpa os periféricos do *Username* e da *Password*;
- "**ValidateRoutine**" – Aguarda um "clique" no "**OK_Button**";
- "**ShowUserAndPassOnDisplayRoutine**" – Copia dos periféricos *Username* e *Password* para o *display*;
- "**SaveUserInDBRoutine**" – Guarda os periféricos *Username* e *Password* na base de dados;
- "**InvalidOptionRoutine**" – Mostra a mensagem de opção introduzida inválida.

4. Conclusão

Este segundo projeto permitiu aplicar os conhecimentos adquiridos nas aulas acerca da linguagem Assembly. Sendo que esta é uma linguagem de baixo nível, tornou o processo de programação mais complexo. Foi essencial entender o funcionamento de cada instrução para uma correta implementação das mesmas, fazendo também uma planificação da memória para não ocorrerem perdas de informações importantes (instruções, menus, variáveis, etc).

5. Anexo A

5.1. Mapa de utilização de memória

		HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Inicio do Programa		0	Begin															
		10	Display															
		...																
		7F																
		...																
Periférico do Username		150	Username															
Periférico da Passord		160	Password															
		170	OK_Button		NR_SEL_Button													
		...																
		2000	Diferentes menus															
		...																
		3000																
		...																
Base de dados de utilizadores		4000	Username															
	1	4010	Password															
		4020	Histórico															
		4030	Username															
	2	4040	Password															
		4050	Histórico															
	...	...																
		40D0	Username															
	5	40E0	Password															
		40F0	Histórico															
		...																
		6000	Código do programa															
		...																
		64A0																
		...																
		9900	Stack															
		...																
		FFF0																

6. Anexo B

```
;-------------------------------------------------------------------------------------------------------------;
;                                Trabalho 2 AC - PizzO - Pizzaria Online                                      ;
;                                                                                                             ;
;                                     Joel Freitas - 2031915                                                  ;
;                                     Pedro Jaques - 2046919                                                  ;
;-------------------------------------------------------------------------------------------------------------;


;Peripherals

 Username_Start    EQU  0150H          ;Address of the input peripheral Username
        Username_End      EQU  0157H  ;Address of the input peripheral Username
 Password_Start    EQU  0160H ;Address of the input peripheral Password
        Password_End                  EQU  0167H      ;Address of the input peripheral
Password
```

```
OK_Button               EQU  0170H      ;OK button address
NR_SEL_Button           EQU  0172H      ;NR_SEL button address

;Display Adrresses

Display_Start                   EQU     0010H   ;Display start address
    Display_End                         EQU     007FH   ;Display end address
    Username_Start_Display      EQU     0034H   ;Start address to display the Username
    Username_End_Display        EQU     003BH   ;End address to display the Username
    Password_Start_Display      EQU     0054H   ;Start address to display the Password
    Password_End_Display        EQU     005BH   ;End address to display the Password
    Total_Euros_Start                   EQU     0046H
    ;Start address to display the total price
    Total_Euros_End                     EQU     0048H
    ;End address to display the total price
    Total_Cent_Start                    EQU     004AH
    ;Start address to display cents
    Total_Cent_End                      EQU     004BH
    ;End address to display cents
    Pizza_Name_Title_End        EQU     001FH   ;End address to display Pizza Name

;Constants

    StackPointer                EQU     9900H   ;Start address of the stack
    CleaningCharacter           EQU     20H     ;Cleaning character for the display
```

```
Asterisk                                EQU             2AH
;Asterisk character for the presentation of Password


SmallPrice                              EQU             5
;Small Pizza Price
LargePrice                              EQU             9
;Large Pizza Price


OptNewAccount                           EQU     1
;Create New Account Option
OptLogin                                        EQU     2
;Login Option
OptExitLoginMenu        EQU             3                               ;Exit
Login Menu Option


OptOrderPizza                   EQU             1
;Order Pizza Option
OptLogout                                       EQU             2
        ;Logout Option


OptPesto                EQU             1                               ;Pesto
Pizza Option
OptFourCheese                   EQU     2                               ;Four
Cheese Pizza Option
OptChicken              EQU     3                               ;Chicken Pizza
Option
OptShrimp       EQU     4                               ;Shrimp Pizza Option
OptHawaii                                       EQU             5
        ;Hawaii Pizza Option


OptSmallPizza                   EQU             1
;Small Pizza Option
OptLargePizza                   EQU     2                               ;Large
Pizza Option
```

```
OptOrderMore                    EQU     1
;Order More Option
OptPayment                      EQU     2
;Payment Option

OptYes                          EQU     1
        ;Yes Option
OptNo                           EQU     2
                ;No Option

MaxUsersInDB                    EQU     5
;Maximum number of users allowed in DB
Iterate_User                    EQU     30H             ;Value
to be able to iterate between users in Database
Iterate_User_Info       EQU     10H             ;Value to be
able to iterate between user info (Username, Password, historic)
UserPassMaxSize                 EQU     8
;Maximum character size for Username and Password

;Database (Username, Password e purchase history)

DB_Start        EQU     4000H   ;Start address for the Database
DB_End          EQU     40F0H   ;End address for the Database

;-------------------------------------------------------------------------------------------------------;
;
Screens/Menus                                                   ;
;-------------------------------------------------------------------------------------------------------;

PLACE 2000H

Start_Screen:
```

```
                STRING "    Welcome     "
                STRING "      to        "
                STRING "     PizzO      "
                STRING "       -        "
                STRING "Online Pizzaria "
                STRING "                "
                STRING "            Press OK    "

PLACE 2080H

        Login_Menu:
                STRING "     PizzO      "
                STRING "                "
                STRING "1. New Account  "
                STRING "2. Login        "
                STRING "3. Exit         "
                STRING "                "
                STRING " OK to select   "

PLACE 2100H

        Login_Form:
                STRING "     Login      "
                STRING "    Username:   "
                STRING "                "
                STRING "    Password:   "
                STRING "                "
                STRING "                "
                STRING " OK to continue "
```

```
PLACE 2180H

        Account_Created_Screen:
                STRING "     PizzO      "
                STRING "                "
                STRING "Account Created "
                STRING " Successfully!  "
                STRING "                "
                STRING "                "
                STRING " OK to continue "

PLACE 2200H

        Existing_Username_Screen:
                STRING "     PizzO      "
                STRING "                "
                STRING "  This Username "
                STRING "     is not     "
                STRING "   available!   "
                STRING "                "
                STRING " OK to continue "

PLACE 2280H

        Missing_Field_Screen:
                STRING "     PizzO      "
                STRING "                "
                STRING " One field was  "
                STRING " not filled in! "
                STRING "                "
```

```
                STRING "             "
                STRING " OK to continue "

PLACE 2300H

        Incomplete_Password_Screen:
                STRING "     PizzO     "
                STRING "               "
                STRING "     Invalid    "
                STRING "    Password!   "
                STRING "At least 3 chars"
                STRING "               "
                STRING " OK to continue "

PLACE 2380H

        DB_Full_Screen:
                STRING "     PizzO     "
                STRING "               "
                STRING " Can not create "
                STRING "    Account!    "
                STRING " Full Database! "
                STRING "               "
                STRING " OK to continue "

PLACE 2400H

        Invalid_Option_Screen:
                STRING "     PizzO     "
                STRING "               "
```

```
                STRING " Invalid Option!"
                STRING " Choose a valid "
                STRING "     option     "
                STRING "                "
                STRING " OK to continue "

PLACE 2480H

        Account_Not_Created_Screen:
                STRING "     PizzO      "
                STRING "                "
                STRING "  This Account  "
                STRING "     is not     "
                STRING "    created!    "
                STRING "                "
                STRING " OK to continue "

PLACE 2500H

        Invalid_Password_Screen:
                STRING "     PizzO      "
                STRING "                "
                STRING "     Invalid    "
                STRING "    Password!   "
                STRING "                "
                STRING "                "
                STRING " OK to continue "

PLACE 2580H
```

```
Valid_Login_Screen:
        STRING "     PizzO      "
        STRING "                "
        STRING "                "
        STRING "  Valid Login!  "
        STRING "                "
        STRING "                "
        STRING " OK to continue "

PLACE 2600H

Confirm_Exit_Menu:
        STRING "     PizzO      "
        STRING "                "
        STRING "  Are you sure? "
        STRING "                "
        STRING "1. Yes    2. No "
        STRING "                "
        STRING " OK to select   "

PLACE 2680H

Exit_Screen:
        STRING "     PizzO      "
        STRING "                "
        STRING "   Thanks for   "
        STRING "   preference!  "
        STRING "Come back always"
        STRING "                "
        STRING " OK to continue "
```

```
PLACE 2700H


        Main_Menu:
                STRING "     PizzO      "
                STRING "                "
                STRING "1. Order Pizza  "
                STRING "2. Logout       "
                STRING "                "
                STRING "                "
                STRING " OK to select   "

PLACE 2780H


        Pizza_Menu:
                STRING "     PizzO      "
                STRING "1. Pesto        "
                STRING "2. Four Cheese  "
                STRING "3. Chicken      "
                STRING "4. Shrimp       "
                STRING "5. Hawaii       "
                STRING " OK to select   "

PLACE 2800H


        Pizza_Size_Menu:
                STRING "                "
                STRING "                "
                STRING "     Sizes:     "
                STRING "1. Small (5 EUR)"
```

```
                STRING "2. Large (9 EUR)"
                STRING "            "
                STRING " OK to select   "

PLACE 2880H

        Payment_Menu:
                STRING "    PizzO     "
                STRING "            "
                STRING "            "
                STRING "1. Order more   "
                STRING "2. Payment      "
                STRING "            "
                STRING " OK to select   "

PLACE 2900H

        Payment_Screen:
                STRING "    PizzO     "
                STRING "            "
                STRING "            "
                STRING "TOTAL:   ,   EUR"
                STRING "Disc.:   ,   EUR"
                STRING "            "
                STRING " OK to pay      "

PLACE 2980H

        Register_Form:
                STRING "          New Account  "
```

```
                STRING "    Username:   "
                STRING "                "
                STRING "    Password:   "
                STRING "                "
                STRING "                "
                STRING " OK to continue "


PLACE 2A00H


        Pesto:
                STRING "     Pesto      "
        Four_Cheese:
                STRING "   Four Cheese  "
        Chicken:
                STRING "     Chicken    "
        Shrimp:
                STRING "     Shrimp     "
        Hawaii:
                STRING "     Hawaii     "


;-------------------------------------------------------------------------------------------------------;
;
Main Program                                                                    ;
;-------------------------------------------------------------------------------------------------------;


PLACE 0000H


        Begin:
                MOV R0, Begining
        ;Moves the address of the tag Start to R0
```

```
                JMP R0


PLACE 6000H


        Begining:

                MOV SP, StackPointer                    ;Base of stack
at 9900H

                CALL CleanDisplayRoutine                ;Routine to clean the
display

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL CleanUserPassRoutine               ;Routine to clean
Username and Password

                CALL ValidateRoutine                    ;Validation to
enter


        Start:

                MOV R2, Start_Screen                    ;Move to R2
the address of the "Start_Screen"

                CALL ShowDisplayRoutine                 ;Routine to
display the "Start_Screen"

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL CleanUserPassRoutine               ;Routine to clean
Username and Password

                CALL ValidateRoutine                    ;Validation to
enter


        LoginMenu:

                MOV R2, Login_Menu                      ;Moves
to R2 the address of the "Login_Menu"

                CALL ShowDisplayRoutine                 ;Routine to
display the "Login_Menu"

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL ValidateRoutine                    ;Routine to
validate the selected options
```

```
ReadLoginMenuOptions:

        MOV R0, NR_SEL_Button                                   ;Moves
to R0 the address of the peripheral "NR_SEL_Button"

        MOVB R1, [R0]
;Moves to R1 the value of the address "NR_SEL_Button"

        CMP R1, OptNewAccount
;Compares the value of R1 with the constant "OptNewAccount"

        JEQ NewAccountForm                                      ;If the
values are equal, jumps to the address "NewAccountForm"

        CMP R1, OptLogin
;Compares the value of R1 with the constant "OptLogin"

        JEQ LoginForm
;If the values are equal, jumps to the address "LoginForm"

        CMP R1, OptExitLoginMenu                ;Compares the value of
R1 with the constant "OptExitLoginMenu"

        JEQ ConfirmExitMenu                                     ;If
equal, jumps to "ConfirmExitMenu"

        CALL InvalidOptionRoutine               ;Routine to display
invalid option pop up

        JMP LoginMenu
        ;Goes back to menu if none where selected


ConfirmExitMenu:

        MOV R8, ConfirmExitMenu

        MOV R2, Confirm_Exit_Menu               ;Moves to R2 the
address of "Confirm_Exit_Menu"

        CALL ShowDisplayRoutine                         ;Routine to
display the "Confirm_Exit_Menu"

        CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

        CALL ValidateRoutine                            ;Routine to
validate the selected options


ReadExitOptions:
```

```
                MOV R0, NR_SEL_Button                                   ;Moves
to R0 the address of the peripheral "NR_SEL_Button"

                MOVB R1, [R0]
        ;Moves to R1 the content of the address "NR_SEL_Button"

                CMP R1, OptYes
        ;Compares the value of R1 with the constant "OptYes"

                JEQ     ExitProgram
        ;If true jumps to "ExitProgram"

                CMP R1, OptNo
        ;Compares the value of R1 with the constant "OptNo"

                JEQ     LoginMenu
                ;If true jumps back to de "LoginMenu"

                CALL InvalidOptionRoutine                       ;If the option selected
does not exist calls routine to display invalid option pop up

                JMP LoginMenu
                ;Goes back to menu if none where selected


        ExitProgram:

                MOV R2, Exit_Screen                                     ;Moves
to R2 the address of "Exit_Screen"

                CALL ShowDisplayRoutine                                 ;Routine to
display the "Exit_Screen"

                CALL CleanPeripheralsRoutine            ;Routine to clean the input peripherals

                CALL ValidateRoutine                                    ;Routine to
validate the selected options

                JMP Begining
        ;Goes back to the begining of the program


        NewAccountForm:

                CALL CheckUsersInDBRoutine              ;Routine to check if more users
can be added(R4 keeps with number of users in DB)

                CMP R4, MaxUsersInDB                                    ;Compares R4
with R1 to check if there is space available in DB

                JEQ FullDataBasePopUp                                   ;If no
space available jumps to full DB screen
```

```
MOV R2, Register_Form                                   ;Moves to R2 the address of "Register_Form"

CALL ShowDisplayRoutine                                 ;Routine to display the "Register_Form"

CALL CleanPeripheralsRoutine            ;Routine to clean the input peripherals

CALL CleanUserPassRoutine                               ;Routine to clean the Username and Password inputs

CALL ValidateRoutine                                    ;Routine to validate the selected options

CALL ShowUserAndPassOnDisplayRoutine            ;Routine to Show Username and Password on the display

CALL CleanPeripheralsRoutine            ;Routine to clean the input peripherals

CALL ValidateRoutine                                    ;Routine to validate the selected options

CALL CheckUsername
        ;Routine to check if the Username inserted is already taken

MOV R9, UserPassMaxSize                                 ;Moves to R9 the constant "UserPassMaxSize"

CMP R7, R9
        ;Compares R7 with UserPassMaxSize constant to check if characters are equal to the one in DB

JEQ     ExistingUsernamePopUp                           ;If equal jumps to "ExistingUsernamePopUp"

CALL CheckFormFieldsRoutine             ;Routine to check if any field was left empty

CMP R5, 0
                ;Compares the value of R5 (amount of characters that Username has) with 0

JLE MissingFieldPopUp                                   ;If equal, no Username was inserted, jumps to "MissingFieldPopUp"

CMP R6, 0
                ;Compares the value of R6 (amount of characters that Password has) with 0

JLE MissingFieldPopUp                                   ;If equal, no Password was inserted, jumps to "MissingFieldPopUp"
```

```
                CMP R6, 3
                        ;Compares the value of R6 (amount of characters that Password has)
with 3

                JLT IncompletePasswordPopUp             ;If less, the Password was
incomplete(at least 3 characters), jumps to "IncompletePasswordPopUp"

                CALL SaveUserInDBRoutine                        ;Routine to save new
user to the Database

                JMP     AccountCreatedWithSuccess       ;After creating account jumps
to "AccountCreatedWithSuccess"


        FullDataBasePopUp:

                MOV R2, DB_Full_Screen                          ;Moves to R2
the address of "DB_Full_Screen"

                CALL ShowDisplayRoutine                         ;Routine to
display the "DB_Full_Screen"

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL CleanUserPassRoutine                       ;Routine to clean the
Username and Password inputs

                CALL ValidateRoutine                            ;Routine to
validate the selected options

                JMP LoginMenu
                ;Jumps back to "LoginMenu"


        ExistingUsernamePopUp:

                MOV R2, Existing_Username_Screen ;Moves to R2 the address of
"Existing_Username_Screen"

                CALL ShowDisplayRoutine                         ;Routine to
display the "Existing_Username_Screen"

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL CleanUserPassRoutine                       ;Routine to clean the
Username and Password inputs

                CALL ValidateRoutine                            ;Routine to
validate the selected options

                JMP LoginMenu
                ;Jumps back to "LoginMenu"
```

```
MissingFieldPopUp:

        MOV R2, Missing_Field_Screen        ;Moves to R2 the address of "Missing_Field_Screen"

        CALL ShowDisplayRoutine                     ;Routine to display the "Missing_Field_Screen"

        CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals

        CALL CleanUserPassRoutine                   ;Routine to clean the Username and Password inputs

        CALL ValidateRoutine                        ;Routine to validate the selected options

        JMP LoginMenu
        ;Jumps back to "LoginMenu"


IncompletePasswordPopUp:

        MOV R2, Incomplete_Password_Screen;Moves to R2 the address of "Incomplete_Password_Screen"

        CALL ShowDisplayRoutine                     ;Routine to display the "Incomplete_Password_Screen"

        CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals

        CALL CleanUserPassRoutine                   ;Routine to clean the Username and Password inputs

        CALL ValidateRoutine                        ;Routine to validate the selected options

        JMP LoginMenu
        ;Jumps back to "LoginMenu"


AccountCreatedWithSuccess:

        MOV R2, Account_Created_Screen      ;Moves to R2 the address of "Account_Created_Screen"

        CALL ShowDisplayRoutine                     ;Routine to display the "Account_Created_Screen"

        CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals
```

```
        CALL CleanUserPassRoutine                   ;Routine to clean the
Username and Password inputs

        CALL ValidateRoutine                        ;Routine to
validate the selected options


    LoginForm:

        MOV R2, Login_Form                          ;Moves
to R2 the address of "Login_Form"

        CALL ShowDisplayRoutine                     ;Routine to
display the "Login_Form"

        CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

        CALL CleanUserPassRoutine                   ;Routine to clean the
Username and Password inputs

        CALL ValidateRoutine                        ;Routine to
validate the selected options

        CALL ShowUserAndPassOnDisplayRoutine    ;Routine to Show
Username and Password on the display

        CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

        CALL ValidateRoutine                        ;Routine to
validate the selected options

        CALL CheckUsername
    ;Routine to check if the Username inserted is equal to any on the DB

        MOV R3, UserPassMaxSize                     ;Moves to R9 the
constant "UserPassMaxSize"

        CMP R7, R3
        ;Compares R7 with UserPassMaxSize constant to check if Username exists in
DB

        JLT     UsernameNotFound                    ;If less
than UserPassMaxSize, it means that the Username inserted is not in the Database, so jumps
to "UsernameNotFound"

        CALL CheckPassword
    ;Routine to check the Username and Password

        CMP R9, R3
        ;Compares R9 with UserPassMaxSize constant to check if Username exists in
DB
```

```
                JLT     InvalidPassword
        ;If less than 6, it means that the Password inserted is not correct Database, so jumps to
"InvalidPassword"

                JMP     ValidLogin


        JumpToExit:

                JMP ConfirmExitMenu


        UsernameNotFound:

                MOV R2, Account_Not_Created_Screen      ;Moves to R2 the address of
"Account_Not_Created_Screen"

                CALL ShowDisplayRoutine                 ;Routine to
display the "Account_Not_Created_Screen"

                CALL CleanUserPassRoutine               ;Routine to clean the
Username and Password inputs

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL ValidateRoutine                    ;Routine to
validate the selected options

                JMP LoginMenu
                ;Jumps back to the LoginMenu


        InvalidPassword:

                MOV R2, Invalid_Password_Screen ;Moves to R2 the address of
"Invalid_Password_Screen"

                CALL ShowDisplayRoutine                 ;Routine to
display the "Invalid_Password_Screen"

                CALL CleanUserPassRoutine               ;Routine to clean the
Username and Password inputs

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL ValidateRoutine                    ;Routine to
validate the selected options

                JMP LoginForm
        ;Jumps back to the LoginForm
```

```
ValidLogin:
        MOV R2, Valid_Login_Screen          ;Moves to R2 the address of "Valid_Login_Screen"
        CALL ShowDisplayRoutine             ;Routine to display the "Valid_Login_Screen"
        CALL CleanUserPassRoutine           ;Routine to clean the Username and Password inputs
        CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals
        CALL ValidateRoutine                ;Routine to validate the selected options

MainMenu:
        MOV R2, Main_Menu
        ;Moves to R2 the address of "Main_Menu"
        CALL ShowDisplayRoutine             ;Routine to display the "Main_Menu"
        CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals
        CALL ValidateRoutine                ;Routine to validate the selected options

ReadMainMenuOptions:
        MOV R0, NR_SEL_Button               ;Moves to R0 the address of the peripheral "NR_SEL_Button"
        MOVB R1, [R0]
        ;Moves to R1 the content of the address "NR_SEL_Button"
        CMP R1, OptOrderPizza               ;Compares the value of R1 with the constant "OptOrderPizza"
        JEQ     PizzaMenu
        ;If true jumps to "ExitProgram"
        CMP R1, OptLogout
        ;Compares the value of R1 with the constant "OptLogout"
        JEQ     JumpToExit
        ;If true jumps to "ConfirmExitMenu"
        CALL InvalidOptionRoutine           ;If the option selected does not exist calls routine to display invalid option pop up
```

```
        JMP MainMenu
    ;Goes back to menu if none where selected


        PizzaMenu:

            MOV R2, Pizza_Menu                                  ;Moves
to R2 the address of "Pizza_Menu"

            CALL ShowDisplayRoutine                             ;Routine to
display the "Pizza_Menu"

            CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals

            CALL ValidateRoutine                                ;Routine to
validate the selected options


        ReadPizzaMenuOptions:

            MOV R0, NR_SEL_Button                               ;Moves
to R0 the address of the peripheral "NR_SEL_Button"

            MOVB R1, [R0]
    ;Moves to R1 the content of the address "NR_SEL_Button"

            CMP R1, OptPesto
    ;Compares the value of R1 with the constant "OptPesto"

            MOV R3, Pesto
    ;Moves to R2 the address of "Pesto"

            JEQ     PizzaSizeMenu
    ;If true jumps to "PizzaSizeMenu"

            CMP R1, OptFourCheese
    ;Compares the value of R1 with the constant "OptFourCheese"

            MOV R3, Four_Cheese                                 ;Moves
to R2 the address of "Four_Cheese"

            JEQ     PizzaSizeMenu
    ;If true jumps to "PizzaSizeMenu"

            CMP R1, OptChicken
    ;Compares the value of R1 with the constant "OptChicken"

            MOV R3, Chicken
    ;Moves to R2 the address of "Chicken"

            JEQ     PizzaSizeMenu
    ;If true jumps to "PizzaSizeMenu"
```

```
        CMP R1, OptShrimp
;Compares the value of R1 with the constant "OptShrimp"

        MOV R3, Shrimp
;Moves to R2 the address of "Shrimp"

        JEQ     PizzaSizeMenu
;If true jumps to "PizzaSizeMenu"

        CMP R1, OptHawaii
;Compares the value of R1 with the constant "OptHawaii"

        MOV R3, Hawaii
;Moves to R2 the address of "Hawaii"

        JEQ     PizzaSizeMenu
;If true jumps to "PizzaSizeMenu"

        CALL InvalidOptionRoutine                   ;If the option selected
does not exist calls routine to display invalid option pop up

        JMP PizzaMenu
;Goes back to menu if none where selected


    PizzaSizeMenu:

        MOV R2, Pizza_Size_Menu                     ;Moves to R2
the address of "Pizza_Size_Menu"

        CALL ShowDisplayRoutine                     ;Routine to
display the "Pizza_Size_Menu"

        CALL ShowPizzaNameRoutine                   ;Routine to display
pizza name on title

        CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals

        CALL ValidateRoutine                        ;Routine to
validate the selected options


    ReadPizzaSizeOptions:

        MOV R0, NR_SEL_Button                               ;Moves
to R0 the address of the peripheral "NR_SEL_Button"

        MOVB R1, [R0]
;Moves to R1 the value of "NR_SEL_Button"

        CMP R1, OptSmallPizza                       ;Compares the
value of R1 with the constant "OptSmallPizza" (1)
```

```
                MOV R3, SmallPrice                                  ;Saves
the small pizza price in R3 (5€)

                JEQ     PaymentMenu
        ;If true jumps to "PaymentMenu"

                CMP R1, OptLargePizza                               ;Compares the
value of R1 with the constant "OptLargePizza" (2)

                MOV R3, LargePrice                                  ;Saves
the large pizza price in R3 (9€)

                JEQ     PaymentMenu
        ;If true jumps to "PaymentMenu"

                CALL InvalidOptionRoutine                           ;If the option selected
does not exist calls routine to display invalid option pop up

                JMP PizzaSizeMenu
        ;Goes back to menu if none where selected


        PaymentMenu:

                MOV R0, Iterate_User_Info

                ADD R10, R0

                CALL UpdateUserHistoricRoutine

                MOV R2, Payment_Menu                                ;Moves
to R2 the address of "Payment_Menu"

                CALL ShowDisplayRoutine                             ;Routine to
display the "Payment_Menu"

                CALL CleanPeripheralsRoutine        ;Routine to clean the input peripherals

                CALL ValidateRoutine                                ;Routine to
validate the selected options


        ReadPaymentMenuOptions:

                MOV R0, NR_SEL_Button                               ;Moves
to R0 the address of the peripheral "NR_SEL_Button"

                MOVB R1, [R0]
        ;Moves to R1 the value of "NR_SEL_Button"

                CMP R1, OptOrderMore
        ;Compares the value of R1 with the constant "OptOrderMore" (1)
```

```
                JEQ     PizzaMenu
                ;If true jumps to "PizzaMenu"

                CMP R1, OptPayment
        ;Compares the value of R1 with the constant "OptPayment" (2)

                JEQ     PaymentScreen
        ;If true jumps to "PaymentScreen"

                CALL InvalidOptionRoutine                       ;If the option selected
does not exist calls routine to display invalid option pop up

                JMP PaymentMenu
        ;Goes back to menu if none where selected


        PaymentScreen:

                MOV R2, Payment_Screen                          ;Moves to R2
the address of "Payment_Screen"

                CALL ShowDisplayRoutine                         ;Routine to
display the "Payment_Screen"

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL ValidateRoutine                            ;Routine to
validate the selected options

                JMP     MainMenu


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

;

        End Main Program

                                ;

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


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

;
                                                                Routine
to clean the display
                                                                ;

;-----------------------------------------------------------------------------------------------------;
```

```
CleanDisplayRoutine:

        PUSH R0
                ;Saves in Stack the records that are changed during the routine

        PUSH R1

        PUSH R2

        MOV R0, Display_Start                   ;Moves to R0 the address "Display_Start"

        MOV R1, Display_End                     ;Moves to R1 the address "Display_End"

        MOV R2, CleaningCharacter               ;Moves to R2 the "CleaningCharacter" (20H)


CleanDisplayCicle:

        MOVB [R0], R2
;Moves to the current Display address the CleaningCharacter to clean the Display

        CMP R0, R1
        ;Compares the address of the start of the Display with the address of the end of the display

        JGE EndOfCleaningRoutine                ;If equal, the display is clean and terminates the routine

        ADD R0, 1
                ;Adds 1 to the current address of the Display to move to the next position on the display

        JMP CleanDisplayCicle                   ;If the end of the display hasn't been reached yet, repeat the cycle


EndOfCleaningRoutine:

        POP R2
                ;Removes from Stacks the records

        POP R1

        POP R0

        RET
```

```
;--------------------------------------------------------------------------------------------------------;
;                                                Routine to clean the Peripherals
                                                ;
;--------------------------------------------------------------------------------------------------------;


        CleanPeripheralsRoutine:

                PUSH R0
                                ;Saves in Stack the records that are changed during the routine

                PUSH R1

                PUSH R2

                MOV R0, OK_Button
        ;Moves to R0 the address of the peripheral "OK_Button"

                MOV R1, NR_SEL_Button                                   ;Moves
to R1 the address of the peripheral "NR_SEL_Button"

                MOV R2, 0
                        ;Moves to R2 the constant 0

                MOVB [R0], R2
        ;Puts 0 in the peripheral "OK_Button"

                MOVB [R1], R2
        ;Puts 0 in the peripheral "NR_SEL_Button"


        EndOfCleanPeripheralsRoutine:

                POP R2
                        ;Removes from Stack the records

                POP R1

                POP R0

                RET


;--------------------------------------------------------------------------------------------------------;
;                                                Routine to show the Display
                                                ;
;
                The Menu/Screen to display comes in the register R2
```

```
                ;
;-------------------------------------------------------------------------------------------------------------;


        ShowDisplayRoutine:

                PUSH R0
                                ;Saves in Stack the records that are changed during the routine

                PUSH R1

                PUSH R3

                MOV R0, Display_Start                                   ;Moves to R0
the address of "Display_Start"

                MOV R1, Display_End                                             ;Moves
to R1 the address of "Display_End"


        ShowDisplayCycle:

                MOV R3, [R2]
        ;Moves from R2(the menu/screen to display) to R3

                MOV [R0], R3
        ;Moves to the display all the information on the menu/screen

                CMP R0, R1
                ;Compares the address of the start of the Display with the address of the end
of the display

                JGE EndOfShowDisplayRoutine             ;If equal, everythig has been
displayed, so terminates the routine

                ADD R2, 2
                        ;Increments 2 to R2, to move to the next byte to be displayed

                ADD R0, 2
                        ;Increments 2 to the value of the display memory to display the next
byte

                JMP ShowDisplayCycle                                    ;Repeats the
cycle


        EndOfShowDisplayRoutine:

                POP R3
                        ;Removes from Stacks the records
```

```
                POP R1
                POP R0
                RET

;-------------------------------------------------------------------------------------------------------;
;                                Routine to Show Pizza Name(R3) on the title of Pizza Size Menu          ;
;-------------------------------------------------------------------------------------------------------;

        ShowPizzaNameRoutine:
                PUSH R0
                                ;Saves in Stack the records that are changed during the routine

                PUSH R1
                PUSH R2
                MOV R0, Display_Start                                   ;Moves to R0 the address of "Display_Start"
                MOV R1, Pizza_Name_Title_End            ;Moves to R1 the address of "Pizza_Name_Title_End"

        CopyToDisplayPizzaNameCycle:
                MOV R2, [R3]
        ;Moves from R2(the pizza name to display) to R3
                MOV [R0], R2
        ;Moves to the display the pizza name
                CMP R0, R1
                ;Compares the address of "Display_Start" with the address "Pizza_Name_Title_End"
                JGE EnfOfShowPizzaNameRoutine           ;If equal, everythig has been displayed, so terminates the routine
                ADD R3, 2
                        ;Increments 2 to R3, to move to the next byte to be displayed
```

```
                ADD R0, 2
                        ;Increments 2 to the value of the display memory to display the next
byte

                JMP CopyToDisplayPizzaNameCycle      ;Jump to the address
"CopyToDisplayPizzaNameCycle"


        EnfOfShowPizzaNameRoutine:

                POP R2
                        ;Removes from Stacks the records

                POP R1

                POP R0

                RET


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

;                                                Routine to clean the Username and Password
                                                ;

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


        CleanUserPassRoutine:

                PUSH R0
                                ;Saves in Stack the records that are changed during the routine


                PUSH R1

                PUSH R2

                MOV R0, Username_Start                                  ;Moves to R0
the address of "Username_Start"(address of the starting input of Username)

                MOV R1, OK_Button
        ;Moves to R1 the address of "OK_Button"(before this address stays the address of the
end of de Password input peripheral)

                MOV R2, 0
                        ;Moves to R2 the constant 0


        CleanUserPassCycle:
```

```
                MOVB [R0], R2
        ;Moves the value of R2(0) to R0 (position between the beginning of the peripheral
"Username_Start" and the peripheral "OK_Button")

                CMP R0, R1
                ;Compares the address of R0 with R1 to check if both peripherals are clean

                JEQ EndOfCleanUserPassRoutine           ;If the values are equal, the
Username and Password peripherals are clean

                ADD R0, 1
                        ;Increases 1 to the value of the R0 to avance 1 position of memory

                JMP CleanUserPassCycle                                  ;Repeats the
cycle until the peripherals are clean


        EndOfCleanUserPassRoutine:

                POP R2
                        ;Removes from Stack the records

                POP R1

                POP R0

                RET


;-------------------------------------------------------------------------------------------------------;
;                                                       Routine to validate the Button OK
                                                ;
;-------------------------------------------------------------------------------------------------------;


        ValidateRoutine:

                PUSH R0
                                ;Saves in Stack the records that are changed during the routine

                PUSH R1

                MOV R0, OK_Button
        ;Moves to R0 the address of "OK_Button"


        Validate:

                MOVB R1, [R0]
        ;Moves to R1 the value of "OK_Button"
```

```
                CMP R1, 1
                        ;Compares the value of R1 with 1

                JLT Validate
        ;If not equal, jump to  "Validate" address until it does the validation, creating a loop


        EndOfValidateRoutine:

                POP R1
                        ;Remove the records from the Stack

                POP R0

                RET


;-------------------------------------------------------------------------------------------------------;
;                                                Routine to display wrong option
                                                ;
;-------------------------------------------------------------------------------------------------------;


        InvalidOptionRoutine:

                MOV R2, Invalid_Option_Screen           ;Moves to R2 the address of
"Invalid_Option_Screen"

                CALL ShowDisplayRoutine                 ;Routine to
display the "Invalid_Option_Screen"

                CALL CleanPeripheralsRoutine    ;Routine to clean the input peripherals

                CALL ValidateRoutine                    ;Routine to
validate the selected options

                RET


;-------------------------------------------------------------------------------------------------------;
;                                Show Username And Password On Display Routine
                                                ;
;-------------------------------------------------------------------------------------------------------;


        ShowUserAndPassOnDisplayRoutine:
```

```
        PUSH R0
                ;Saves in Stack the records that are changed during the routine
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R7
        MOV R0, Username_Start                  ;Moves to R0 the address of the start of the peripheral Username
        MOV R1, Password_Start                  ;Move to R1 the address of the start of the peripheral Password
        MOV R2, Asterisk
    ;Move to R2 the constant "Asterisk"
        MOV R3, Username_Start_Display  ;Move to R3 the address of the display where Username will start to be displayed
        MOV R4, Username_End_Display    ;Move to R4 the address of the display where Username will end to be displayed
        MOV R5, Password_Start_Display  ;Move to R5 the address of the display where Password will start to be displayed
        MOV R6, Password_End_Display    ;Move to R6 the address of the display where Password will end to be displayed

    CopyUserToDisplayCycle:
        MOVB R7, [R0]
    ;Moves to R7 the value of R0(first character of Username)
        CMP R7, 0
            ;Compares the value of R3 with 0, to check if the end of inserted Username was reached
        JEQ CopyPassToDisplayCycle      ;If equal, jump to "CopyPassToDisplayCycle"
        MOVB [R3], R7
    ;Shows in the display the character from Username peripheral
```

```
                CMP R3, R4
                ;Compares the beginning of the display for the peripheral of Username with
the end

                JEQ CopyPassToDisplayCycle                ;If the end of the display was
reached, it jumps to "CopyPassToDisplayCycle"

                ADD R0, 1
                        ;Increment 1 to the input peripheral for Username

                ADD R3, 1
                        ;Increment 1 to the display for the introduction of Username

                JMP CopyUserToDisplayCycle                ;Repeats the cycle if the
Username isn't all copied


        CopyPassToDisplayCycle:

                MOVB R7, [R1]
        ;Move to R3 the value of R1

                CMP R7, 0
                        ;Compares the value of R3 with 0, to check if the end of inserted
Password was reached

                JEQ EndOfShowUserAndPassOnDisplayRoutine ;If equal, it jump to
"EndOfShowUserAndPassOnDisplayRoutine"

                MOVB [R5], R2
        ;Shows in the display the character from Password peripheral

                CMP R5, R6
                ;Compares the beginning of the display for the peripheral of Password with the
end

                JEQ EndOfShowUserAndPassOnDisplayRoutine ;If you have reached the end of
the display, it jumps to the address "EndOfShowUserAndPassOnDisplayRoutine"

                ADD R1, 1
                        ;Increment 1 to the input peripheral referring to Password

                ADD R5, 1
                        ;Increment 1 to the display for the introduction of Password

                JMP CopyPassToDisplayCycle                ;Repeats the cycle if the
Password isn't all copied


        EndOfShowUserAndPassOnDisplayRoutine:
```

```
                POP R7
                        ;Remove the records from the Stack
                POP R6
                POP R5
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET


;-------------------------------------------------------------------------------------------------------;
;                       Routine to check how many users are there in the Database
                                ;
;-------------------------------------------------------------------------------------------------------;


        CheckUsersInDBRoutine:
                PUSH R0
                                ;Saves in Stack the records that are changed during the routine
                PUSH R1
                PUSH R2
                PUSH R3
                MOV R0, DB_Start
        ;Move to R0 the address of the start of the Database
                MOV R1, DB_End
        ;Move to R1 the address of the end of the Database
                MOV R2, Iterate_User                                    ;Value that
serves to iterate a user in user in memory
                MOV R4, 0
                        ;Moves to R4 the value 0 (variable to count number of users)


        CheckCycle:
```

```
                CMP R0, R1
                ;Compares R0 with R1 to check if the end of DB was reached

                JGT EndOfCheckUsersInDBRoutine        ;If greater, the end of Database was
reached

                MOVB R3, [R0]
        ;Moves to R3 the value of R0

                CMP R3, 0
                        ;Compares R3 with 0

                JEQ EndOfCheckUsersInDBRoutine        ;If equal, all users in the memory have
been checked, jumps to "EndOfCheckUsersInDBRoutine"

                ADD R4, 1
                        ;Increments 1 to R4 (variable to count number of users)

                ADD R0, R2
                ;Increments the address in R0 to iterate to the next user

                JMP CheckCycle
        ;Repeats until all users are checked


        EndOfCheckUsersInDBRoutine:

                POP R3
                        ;Remove the records from the stack

                POP R2

                POP R1

                POP R0

                RET


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

;                               Routine to check if a Username already exists in the Database
                               ;

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


        CheckUsername:

                PUSH R0
                                ;Saves in Stack the records that are changed during the routine

                PUSH R1
```

```
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R8
        MOV R0, Username_Start                  ;Moves to R0 the address of the start of the Username peripheral
        MOV R1, Username_End                    ;Moves to R1 the address of the end of the Username peripheral
        MOV R2, DB_Start
    ;Moves to R2 the address of the start of the Database
        MOV R3, DB_End
    ;Moves to R3 the address of the end of the Database
        MOV R4, Iterate_User                    ;Moves to R4 the value to iterate between users
        MOV R7, 0
            ;Variable to count how many equal characters
        MOV R8, R2
        ;Moves to R8 the address of the start of the Database

    VerificationCycle:
        CMP R2, R3
        ;Check if the end of the database as been reached
        JGT     EndOfCheckUsername              ;If yes, jumps to the end of the routine
        CMP R0, R1
        ;Check if the end of the Username peripheral was reached
        JGT     NextUserCycle
    ;If yes, Username all checked, jumps to "NextUserCycle"
        MOVB R5, [R0]
    ;Moves to R5 the value of the first character in the input
        MOVB R6, [R2]
    ;Moves to R6 the value of the first character in the Database
```

```
                CMP R5, R6
                ;Compares it with the first character of the first Username in the database

                JEQ     NextCharCycle
        ;If equal, jumps to "NextCharCycle"


        NextUserCycle:

                MOV R0, Username_Start                          ;Resets R0 to
the begin os Username peripheral

                ADD R8, R4
                ;Iterates to the next username in the database

                MOV R2, R8
                ;Moves to R2 that address to the next check

                JMP     VerificationCycle                       ;Goes
back to "VerificationCycle"


        NextCharCycle:

                ADD R7, 1
                        ;Adds 1 to the variable that saves the number of equal characters

                ADD R0, 1
                        ;Adds 1 to move to next character on the Username

                ADD R2, 1
                        ;Adds 1 to move to next character on the database

                JMP VerificationCycle                           ;Repeats the
cycle until all Username is checked


        EndOfCheckUsername:

                POP R8
                        ;Remove the records from the stack

                POP R6

                POP R5

                POP R4

                POP R3

                POP R2

                POP R1
```

```
                POP R0
                RET

;-------------------------------------------------------------------------------------------------------------;
;                                                                                   Routine to
check the login
                                ;
;-------------------------------------------------------------------------------------------------------------;

        CheckPassword:
                PUSH R0
                                ;Saves in Stack the records that are changed during the routine
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                PUSH R5
                PUSH R6
                PUSH R7
                PUSH R8
                MOV R0, Password_Start                          ;Moves to R0
the address of the start of the Password peripheral
                MOV R1, Password_End                                    ;Moves
to R1 the address of the end of the Password peripheral
                MOV R2, DB_Start
        ;Moves to R2 the address of the start of the Database
                MOV R3, DB_End
        ;Moves to R3 the address of the end of the Database
                MOV     R4, Iterate_User
                MOV R5, Iterate_User_Info
                MOV R9, 0
                        ;Variable to count equal chars
```

```
                ADD R2, R5
                ;Jumps to the address where the password is saved in database

                MOV R6, R2


        VerificatePasswordCycle:

                CMP R2, R3
                ;Check if the end of the database as been reached

                JGT     EndOfCheckPassword                      ;If yes, jumps to
the end of the routine

                MOV R5, UserPassMaxSize

                MOV R10, R6

                CMP R9, R5

                JGE EndOfCheckPassword

                CMP R0, R1
                ;Compares start of Password input peripheral with the end of it

                JGT     NextPasswordCycle                       ;If
equal, Password all checked, jumps to "NextPasswordCycle"

                MOVB R7, [R0]
        ;Moves to R7 the value of the first character in the input

                MOVB R8, [R2]
        ;Moves to R8 the value of the first character in the Database

                CMP R7, R8
                ;Compares it with the first character of the first Password in the database

                JEQ NextPassCharCycle                           ;If equal, jumps
to the next char


        NextPasswordCycle:

                MOV R0, Password_Start                          ;Resets R0 to
the begin os Password peripheral

                ADD R6, R4
                ;Adds R4 to R6 to jump to the next password in the database

                MOV R2, R6
                ;Updates R2 to the address of the next password
```

```
                JMP     VerificatePasswordCycle                 ;Goes back to
"VerificatePasswordCycle"


        NextPassCharCycle:

                ADD R9, 1
                        ;Adds 1 to the variable that saves the number of equal characters

                ADD R0, 1
                        ;Adds 1 to move to next character on the Password

                ADD R2, 1
                        ;Adds 1 to move to next character on the database

                JMP VerificatePasswordCycle             ;Repeats the cycle until all
Password is checked


        EndOfCheckPassword:

                POP R8
                        ;Remove the records from the stack

                POP R7

                POP R6

                POP R5

                POP R4

                POP R3

                POP R2

                POP R1

                POP R0

                RET


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

;                                       Routine to check if any field on the form was left
empty                                   ;

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


        CheckFormFieldsRoutine:
```

```
                PUSH R0
                        ;Saves in Stack the records that are changed during the routine
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                MOV R0, Username_Start                          ;Moves to R0
the address of the start of the input peripheral Username
                MOV R1, Username_End                                    ;Moves
to R1 the address of the end of the input peripheral Username
                MOV R2, Password_Start                          ;Moves to R2
the address of the start of the input peripheral Password
                MOV R3, Password_End                                    ;Moves
to R3 the address of the start of the input peripheral Password
                MOV R5, 0
                        ;Flag: 0 - no Username, >1 - has Username
                MOV R6, 0
                        ;Flag: 0 - no Password, >1 - has Password


        CheckUserFieldCycle:
                MOVB R4, [R0]
        ;Moves to R4 the value of the R0 address
                CMP R4, 0
                        ;Compares the value of R4 (character of the Username peripheral) with
0
                JGT UsernameFound
        ;If greatter, the Username peripheral contains written characters, jumps to the
"UsernameFound"
                JMP CheckPassFieldCycle                         ;Else, jumps to
CheckPassFieldCycle


        UsernameFound:
                ADD R5, 1
```

```
CheckPassFieldCycle:

        CMP R2, R3
        ;Compares the start of the Password peripheral with the end

        JGT EndOFCheckFormFieldsRoutine      ;If greater, the peripheral was all checked, jumps to "EndOFCheckFormFieldsRoutine"

        MOVB R4, [R2]
    ;Moves to R4 the content of the address of R2

        CMP R4, 0
            ;Compares the value of R4 (character of the Password peripheral) with 0

        JGT PasswordCharFound                                   ;If greatter, the Password input peripheral contains written characters, jumps to "PasswordCharFound"

        JMP EndOFCheckFormFieldsRoutine             ;Else jumps to terminate the routine


    PasswordCharFound:

        ADD R6, 1
            ;Increments the flag that tells if there are characters

        ADD R2, 1
            ;Increments 1 to the value of the R2 (next Password peripheral character)

        JMP CheckPassFieldCycle


    EndOFCheckFormFieldsRoutine:

        POP R4
            ;Remove the records from the stack

        POP R3

        POP R2

        POP R1

        POP R0

        RET


;-------------------------------------------------------------------------------------------------------------;
```

```
;                    Routine to save in the Database the Username and Password
                                ;

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


        SaveUserInDBRoutine:

                PUSH R0
                                ;Saves in Stack the records that are changed during the routine

                PUSH R1

                PUSH R2

                PUSH R3

                PUSH R4

                PUSH R5

                PUSH R6

                PUSH R7

                PUSH R8

                MOV R0, DB_Start
        ;Move to R0 the address of the start of the Database

                MOV R1, Iterate_User                                    ;Move to R1
the constant "Iterate_User" to iterate between users

                MOV R2, Iterate_User_Info                       ;Move to R2 the
constant "Iterate_User_Info" to iterate between the user info (Username, Password and
Historic of Purchase)

                MOV R3, Username_Start                                  ;Move to R3
the address of the start of the peripheral Username

                MOV R4, Username_End                                            ;Move
to R4 the address of the end of the peripheral Username

                MOV R5, Password_Start                                  ;Move to R5
the address of the start of the peripheral Password

                MOV R6, Password_End                                            ;Move
to R6 the address of the end of the peripheral Password


        CheckPositionToSaveNewUser:

                MOV R7, R0
                ;Moves to R7 the address of the start of the Database
```

```
        MOVB R8, [R0]
    ;Moves to R8 the value of the first byte of the Username on Database

        CMP R8, 0
            ;Compares R8 with the constant 0

        JEQ SaveUsernameCycle                                   ;If
equal, there is no user in that position of the Database, jump to "SaveUsernameCycle"

        ADD R0, R1
        ;Adds R1 to R0 to iterate to next user position

        JMP CheckPositionToSaveNewUser      ;Repeats untill an empty position is
found in the Database


    SaveUsernameCycle:

        MOVB R8, [R3]
    ;Moves to R8 the character in the first position of the Username peripheral

        CMP R8, 0
            ;Compares the character with 0

        JEQ JumpToPasswordPosition              ;If equal, Username's saving is
over and jumps to "JumpToPasswordPosition"

        MOVB [R0], R8
    ;Writes character in Database position

        CMP R3, R4
        ;Compares the start of Username peripheral with the end

        JEQ JumpToPasswordPosition              ;If equal, jumps to the
"JumpToPasswordPosition"

        ADD R0, 1
            ;Increments 1 to the actual Database address to write the next
Username character

        ADD R3, 1
            ;Increments 1 to the address of Username peripheral to check the next
character

        JMP SaveUsernameCycle                                   ;If the
Username haven't been all written in the Database, repeats the cycle


    JumpToPasswordPosition:
```

```
                ADD R7, R2
                ;Adds R2 to R7 to place the start of the Password position in the Database in
R7

                MOV R0, R7
                ;Moves to R0 the place to begin writing the Password in the Database


        SavePasswordCycle:

                MOVB R8, [R5]
        ;Moves to R8 the character of the peripheral Password

                CMP R8, 0
                        ;Compares the value R8 with 0

                JEQ EndOfSaveUserInDBRoutine            ;If equal, the Password writing
is over and jumps to "EndOfSaveUserInDBRoutine"

                MOVB [R0], R8
        ;Writes Password character in Database

                CMP R5, R6
                ;Compares the start of Password peripheral with the end

                JEQ EndOfSaveUserInDBRoutine            ;If equal, jumps to
"EndOfSaveUserInDBRoutine"

                ADD R0, 1
                        ;Increments 1 to the actual Database address to write the next
Password character

                ADD R5, 1
                        ;Increments 1 to the address of Password peripheral to check the next
character

                JMP SavePasswordCycle                                   ;If the
Password haven't been all written in the Database, repeats the cycle


        EndOfSaveUserInDBRoutine:

                POP R8
                        ;Remove the records from the stack

                POP R7

                POP R6

                POP R5

                POP R4
```

```
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;-------------------------------------------------------------------------------------------------------;
;                                                   Routine to update user historic of pizzas                                                   ;
;-------------------------------------------------------------------------------------------------------;

        UpdateUserHistoricRoutine:
                PUSH R0
                                ;Saves in Stack the records that are changed during the routine
                PUSH R1
                MOV R0, R10
                ;Moves to R0 the value of R10 (address where the purchase history is located)

        UpdateHistoricCycle:
                MOVB R1, [R0]
        ;Moves to R1 the value already stored in the user's history
                ADD R1, R3
                ;Increments the value of the history with the selected pizza value
                MOVB [R0], R1
        ;Puts the value of the sum of the current purchase with the history in the memory position where the history is located
                JMP EndOfUpdateUserHistoricRoutine              ;Ends the routine

        EndOfUpdateUserHistoricRoutine:
                POP R1
                POP R0
                RET
```