

## Arquitetura de Computadores

Projeto 2 – Pizzaria

#### **Docentes:**

- Dionísio Barros
- Pedro Camacho
- Nuno Ferreira
- Sofia Isabel Silva Inácio

#### **Alunos:**

- Juan Silva, nº 2082516
- Joel Camacho, nº 2083516

# Índice

Índice	2
Introdução	3
Resultados	4
Login	4
Menu de Cliente	
Menu de Pizzas	
Menu de Tamanho	4
Menu de Pagamento	4
Bibliografia	5
Conclusão	6
Anexo A – Fluxogramas Gerais	7
Anexo B – Código em Linguagem Assembly	

## Introdução

Neste projeto foi desenvolvido um programa em *assembly* que permite controlar os pedidos de comida online de uma pizzaria que faz entregas. A interface gráfica (ou display) simula a página web onde os clientes efetuam os seus pedidos online.

### Resultados

Ao iniciar a simulação teremos um menu onde permite-nos escolher entre fazer um *login* ou fazer um novo registo.

### Login

Ao efetuar o *login* o cliente deverá introduzir um *username* e uma *password*, caso estes dados não correspondam com algum dos dados na base de dados é apresentada uma mensagem de erro. Caso os dados introduzidos nos periféricos coincidam com aos dados na base de dados o cliente é redirecionado para o menu de cliente.

#### Menu de Cliente

No menu de cliente são apresentadas duas escolhas, uma para efetuar uma encomenda e outra para fazer *logout*, caso o cliente escolha fazer *logout* é redirecionado para o menu inicial, caso escolha encomendar uma pizza é apresentado um menu com cinco pizzas.

#### Menu de Pizzas

No menu de pizzas estão disponíveis cinco pizzas para escolha, em qualquer uma das cinco escolhas o cliente é redirecionado para o menu de tamanho.

#### Menu de Tamanho

No menu de tamanho o cliente pode escolher se pretende uma pizza de tamanho grande, com o custo de oito euros, ou se pretende uma pizza de tamanho pequeno, com o custo de cinco euros. Após a escolha do tamanho o cliente é redirecionado para o menu de pagamento.

### Menu de Pagamento

Neste menu pretendia-se mostrar ao cliente o valor total gasto na pizzaria e, caso este valor seja maior que cinquenta euros, mostrar que teve desconto de dois euros caso seja uma pizza pequena, ou desconto de quatro euros no caso de ter encomendado uma pizza grande. Infelizmente, este menu não foi bem sucedido.

# Bibliografia

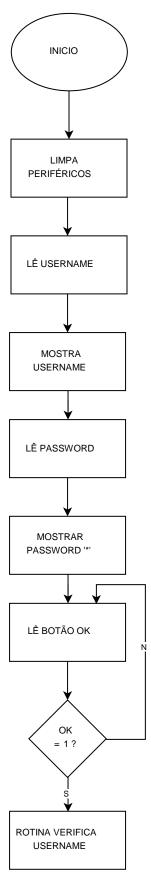
J. Delgado e C. Ribeiro, Arquitetura de Computadores, FCA, 2007.

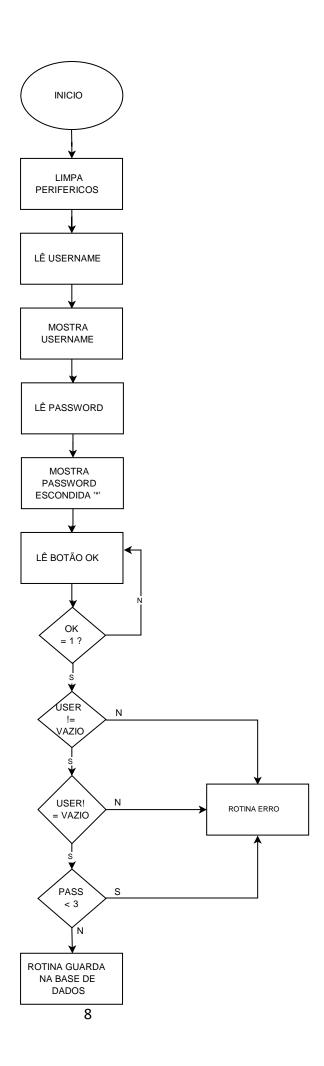
### Conclusão

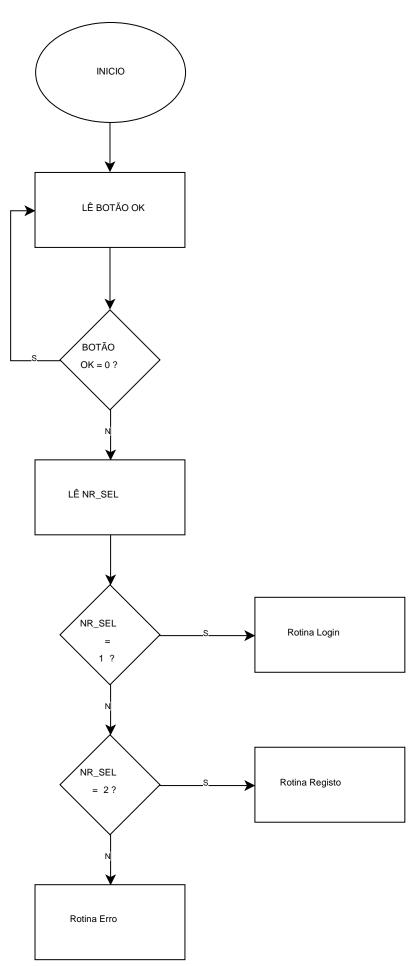
Infelizmente o projeto não está totalmente funcional, mas a maior parte deste funciona como esperado. Devido à falta de tempo, não nos foi possível terminar o pagamento e o desconto, mas o restante funciona devidamente.

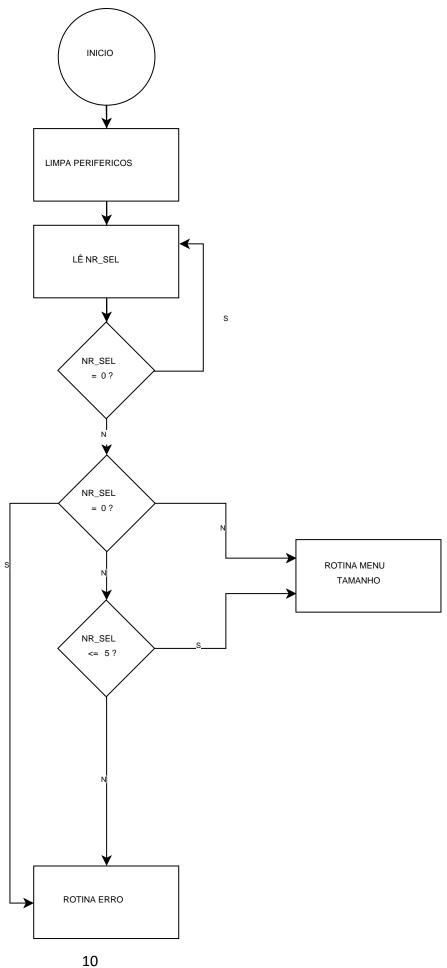
Com este projeto concluiu-se que é possível, e complexo, criar aplicações ou programas apenas em linguagem *assembly* pois esta é de baixo nível, concluiu-se também que o projeto foi gratificante pois adquirimos conhecimentos da linguagem *assembly*.

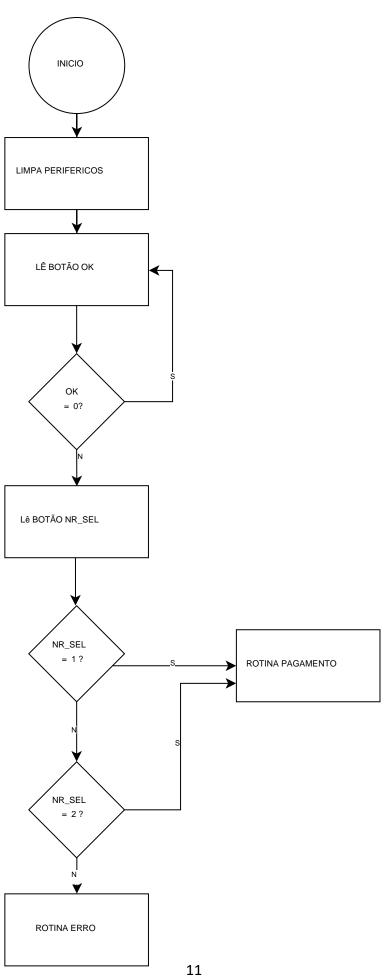
# Anexo A – Fluxogramas Gerais

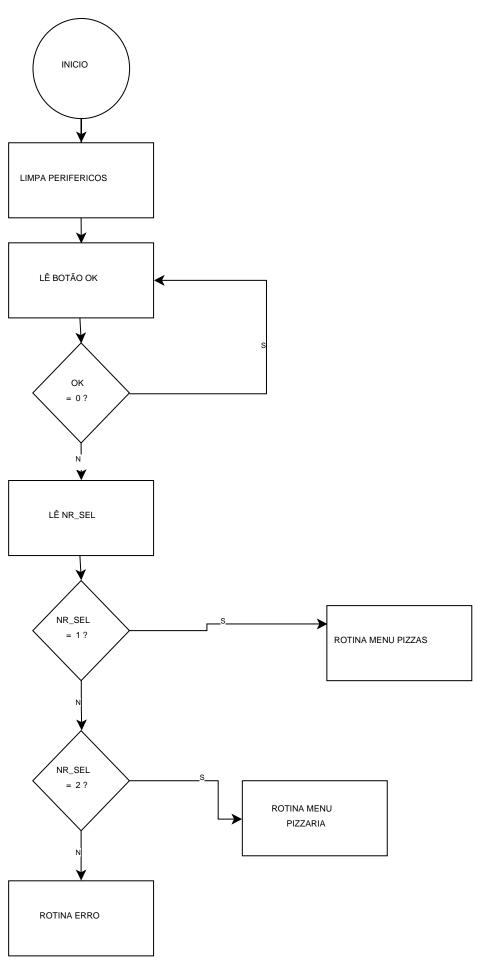












## Anexo B – Código em Linguagem Assembly

; PERIFÉRICOS

```
Username EQU 01A0H ; ENDEREÇO DO PERIFERICO DO USERNAME
Password EQU 01B0H ; ENDEREÇO DO PERIFERICO DA PASSWORD
NR SEL EQU 1C0H ; ENDEREÇO DO SELETOR NR SEL
Ok EQU 1D0H ; ENDEREÇO DO BOTÃO OK
;BASE DE DADOS
BasedeDados EQU 7000H ; ENDEREÇO DE INICIO DA BASE DE DADOS
numClientes EQU 10 ; NUMERO MAXIMO DE CLIENTES DA PIZZARIA
bd user EQU 0 ; INDICE DO USERNAME DENTRO DA FICHA DE CLIENTE
bd pass EQU 16 ; INDICE DA PASSWORD DENTRO DA FICHA DE CLIENTE
bd valor EQU 32 ;INDICE DO VALOR DENTRO DA FICHA DE CLIENTE
tam EQU 48 ; TAMANHO DE CADA FICHA DE CLIENTE
; DISPLAY
Display EQU 200H ; ENDEREÇO ONDE COMEÇA E É APRESENTADO O DISPLAY
DisplayEnd EQU 26FH ; ENDEREÇO ONDE ACABA O DISPLAY
UserNoDisplay EQU 0230H ; ENDEREÇO ONDE É MOSTRADO O USERNAME
INTRODUZIDO
PassNoDisplay EQU 0250H ; ENDEREÇO ONDE É MOSTRADO A PASSWORD
INTRODUZIDO
posicaoValorgasto EQU 0235H ; ENDEREÇO ONDE É MOSTRADO O VALOR GASTO
NO PAGAMENTO
posicaoDesconto EQU 0255H ; ENDEREÇO ONDE É MOSTRADO O DESCONTO NO
PAGAMENTO
; CONSTANTES
CaracterVazio EQU 20H ; ASCII 20H É O ESPAÇO
CaracterAsterisco EOU 2AH ; ASCII 20H É O ASTERISCO
Nr Caracteres EQU 8 ; NUMERO DE CARACTERES DO USERNAME / PASSWORD
MLogin EQU 1
MRegistar EQU 2
OCliente pizza EQU 1
OCliente sair EQU 2
OTamanho_peq EQU 1
OTamanho grande EQU 2
Terminador EQU 00H
valorDesconto EQU 32H ; VALOR 50
StackPointer EQU 6000H
;UTILIZADOR JÁ REGISTADO POR DEFEITO
Place 7000H
     String "Juan"
Place 7010H
     String "1234"
Place 7020H
     String 20
MENUS DO PROJETO
```

```
Place 2000H ; A PARTIR DO ENDEREÇO 2000 DA MEMORIA VAI ESTAR O MENU
ABAIXO
MenuPizzaria:
     String " MENU PIZZARIA "
     String "
     String "1. LOGIN
     String "2. REGISTAR
     String "
     String "
     String "
                             11
Place 2080H
MenuLogin:
     String " MENU LOGIN
     String "
     String " USERNAME:
     String "
     String " PASSWORD:
     String "
     String "
Place 2160H
MenuRegisto:
     String " MENU REGISTO "
     String "
     String " USERNAME:
     String "
     String " PASSWORD:
     String "
     String "
Place 2240H
MenuCliente:
     String " MENU CLIENTE "
     String "
     String "1. PIZZAS
     String "2. SAIR
     String "
     String "
     String "
Place 2320H
MenuPizzas:
     String " MENU PIZZAS
     String "
     String "1. PIZZA 1
     String "2. PIZZA 2
     String "3. PIZZA 3
     String "4. PIZZA 4
     String "5. PIZZA 5
Place 2400H
MenuTamanho:
     String " MENU TAMANHO "
     String "
     String "1. PEQUENA 5 EUR"
     String "2. GRANDE 8 EUR "
     String "
     String "
                             11
     String "
Place 2480H
MenuErro:
```

String " ATENCAO: "

```
String "
                           11
     String "
               OPCAO
     String "
                ERRADA
     String "
     String "
     String "
Place 2560H
MenuPassErrada:
     String "
              ATENCAO:
     String "
                           11
     String "
             PASSWORD
     String "
               INVALIDA
     String "
     String "
     String "
Place 2640H
MenuUserEmUso:
    String "
                           **
              ATENCAO:
     String "
     String "
              USERNAME
     String "
                 EM
     String "
              UTILIZACAO
                           **
     String "
     String "
Place 2720H
MenuUserErrado:
     String "
                           "
               ATENCAO:
     String "
     String "
               USERNAME
     String "
     String "
                           **
               INVALIDO
     String "
     String "
Place 2800H
MenuUserRegistado:
     String " ATENCAO:
     String "
     String "
              CLIENTE
     String "
             REGISTADO
     String "
               COM
     String "
               SUCESSO!
     String "
Place 2880H
MenuPagamento:
     String "
              PAGAMENTO:
     String "
     String " VALOR GASTO:
     String "
     String " DESCONTO:
     String "
                           11
     String "
Place 0000H
Inicio:
     MOV RO, Prinpicio
```

```
JMP R0
Place 4000H
Prinpicio:
    MOV SP, StackPointer
    CALL LimpaPerifericos
    CALL RotinaPizzaria ; APRESENTA DESDE INICIO O MENU PRINCIPAL
PIZZARIA
ROTINA PIZZARIA
RotinaPizzaria:
    PUSH RO
    PUSH R1
    PUSH R2
    PUSH R3
    CALL LimpaPerifericos
    MOV R2, MenuPizzaria ; MOSTRA MENU
    CALL MostraDisplay
pizzaria_Le_ok:
    MOV RO, Ok
    MOVB R1, [R0]
    CMP R1, 0
    JEQ pizzaria Le ok ; CICLO ATÉ O OK PASSAR A '1'
    MOV R2, NR SEL
    MOVB R3, [R2]
    CMP R3, MLogin ; OPCAO = OPÇÃO DE LOGIN?
    JEQ ChamaRotinaLogin ; ESCOLHIDA A OPÇÃO DE LOGIN
    CMP R3, MRegistar ;OPCAO = OPÇÃO DE REGISTO?
    JEQ ChamaRotinaRegisto ; ESCOLHIDA A OPÇÃO DE REGISTO
    MOV R2, MenuErro ; NENHUMA OPÇÃO ESCOLHIDA
    CALL RotinaErro ; ERRO
ChamaRotinaRegisto:
    CALL RotinaRegisto
ChamaRotinaLogin:
    CALL RotinaLogin
    POP R3
    POP R2
    POP R1
    POP R0
    RET
ROTINA DE MENU CLIENTE
RotinaCliente:
    PUSH RO
     PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    CALL LimpaPerifericos
    MOV R2, MenuCliente
    CALL MostraDisplay
cliente Le ok:
```

MOV RO, Ok

```
MOVB R1, [R0]
     CMP R1, 0
     JEQ cliente Le ok ; CICLO ATÉ O OK PASSAR A '1'
     MOV R3, NR SEL
     MOVB R4, [R3]
     CMP R4, OCliente_pizza ;OPCAO = OPÇÃO DE PIZZAS?
     JEQ ChamaRotinaPizzas
     CMP R4, OCliente sair ; OPCAO = OPÇÃO DE LOGOUT?
     JEQ ChamaRotinaPizzaria
     MOV R2, MenuErro
     CALL RotinaErro
ChamaRotinaPizzas:
     CALL RotinaPizzas
ChamaRotinaPizzaria:
     CALL RotinaPizzaria
     POP R4
     POP R3
     POP R2
     POP R1
     POP RO
     RET
ROTINA DE MENU DE PIZZAS
RotinaPizzas:
     PUSH RO
     PUSH R1
     PUSH R2
     PUSH R3
     PUSH R4
     CALL LimpaPerifericos
     MOV R2, MenuPizzas
     CALL MostraDisplay
pizzas Le ok:
     MOV RO, Ok
     MOVB R1, [R0]
     CMP R1, 0
     JEQ pizzas Le ok ; CICLO ATÉ O OK PASSAR A '1'
     MOV R3, NR SEL
     MOVB R4, [R3]
     CMP R4, 0
     JEQ PizzaCompradaErro ;OPCAO = 0, ERRO
     CMP R4, 5
     JLE RotinaPizzaEscolhida ; OPCAO = UMA DAS CINCO PIZZAS (1 -
5)
PizzaCompradaErro:
     MOV R2, MenuErro
     CALL RotinaErro
RotinaPizzaEscolhida:
     CALL RotinaTamanho
     POP R4
     POP R3
     POP R2
     POP R1
```

```
RET
ROTINA DE TAMANHO
RotinaTamanho:
    PUSH RO
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    CALL LimpaPerifericos
    MOV R2, MenuTamanho
    CALL MostraDisplay
tamanho Le ok:
    MOV RO, Ok
    MOVB R1, [R0]
    CMP R1, 0
    JEQ tamanho Le ok ; CICLO ATÉ O OK PASSAR A '1'
    MOV R3, NR SEL
    MOVB R4, [R3]
    CMP R4, OTamanho peq
    JEQ PizzaComprada ; TAMANHO = PEQUENA
    CMP R4, OTamanho grande
    JEQ PizzaComprada ;TAMANHO = GRANDE
    MOV R2, MenuErro
    CALL RotinaErro
PizzaComprada:
    CALL RotinaPagamento
    POP R4
    POP R3
    POP R2
    POP R1
    POP RO
    RET
ROTINA DE PAGAMENTO / APLICA DESCONTO
RotinaPagamento:
     PUSH RO
     PUSH R1
    PUSH R2
    PUSH R3
    PUSH R5
    PUSH R7
     ;R4 - TAMANHO DA PIZZA
     ;R6 - FICHA DO CLIENTE ATUAL
    MOV R3, bd valor ; COLOCA EM R3 O INDICE DO VALOR NO HISTORICO
DA BASE DE DADOS
    ADD R3, R6 ; COLOCA EM R3 O INDICE DO VALOR NO HISTORICO DA
BASE DE DADOS DA FICHA DE CLIENTE PRETENDIDA
    MOV R7, valorDesconto
    CMP R4, OTamanho peq
```

POP RO

### JNE pizzagrande pizzapequena: MOV R5, 2 ; DESCONTO DA PIZZA PEQUENA ADD R3, 5 CMP R3, R7 JGE AplicaDesconto MOV R5, 0 ; NAO DESCONTA CALL RotinaMostraPagamento JMP fim\_pag AplicaDesconto: SUB R3, R7 SUB R3, R5 ; DESCONTA CALL RotinaMostraPagamento JMP fim pag pizzagrande: MOV R5, 4 ; DESCONTO DA PIZZA GRANDE ADD R3, 5 CMP R3, R7 JGE AplicaDesconto MOV R5, 0 ; NAO DESCONTA CALL RotinaMostraPagamento JMP fim pag fim pag: POP R7 POP R5 POP R3 POP R2 POP R1 POP RO ROTINA MOSTRA VALOR E MOSTRA DESCONTO RotinaMostraPagamento: PUSH RO PUSH R1 PUSH R2 PUSH R4 PUSH R5 PUSH R6 ;R5 - DESCONTO ;R3 - VALOR GASTO CALL LimpaPerifericos ;LIMPA PERIFÉRICOS MOV R2, MenuPagamento ; COLOCA MENU DE LOGIN CALL MostraDisplay ; MOSTRA NO DISPLAY O MENU MOV R6, R3 MOV R4, posicaoValorgasto

CALL CONVERTE NUM CHAR

MOV R4, posicaoDesconto CALL CONVERTE NUM CHAR

MOV R6, R5

19

```
MOV RO, Ok
ciclo pagamento:
    MOVB R1, [R0]
     CMP R1, 1
     JNE ciclo_pagamento
     CALL RotinaCliente
     POP R6
     POP R5
     POP R4
     POP R2
     POP R1
     POP R0
     RET
ROTINA CONVERTE (por corrigir)
CONVERTE NUM CHAR:
     ;R6 - VALOR A CONVERTER
     ;R4 - POSICAO A APRESENTAR
     PUSH RO
     PUSH R1
    PUSH R2
     PUSH R3
     PUSH R5
     MOV R0, 10
     MOV R2, R4
     MOV R3, 0; CONTADOR = 0
    MOV R1, R6
PROXIMOCARACTER:
    MOV R4, R1
     MOD R4, R0 ;OBTEM O RESTO DA DIVISAO POR 10
     DIV R1, R0 ; OBTEM QUOCIENTE
     MOV R5, 48
     ADD R5, R4 ;OBTEM "RESTO+48"
     MOVB [R2], R5; COPIA ALGARISMO PARA DISPLAY
     ADD R3, 1 ; INCREMENTA O CONTADOR
     SUB R2, 1 ; DECREMENTA A POSIÇÃO
     CMP R1, 0 ; QUOCIENTE = 0?
     JNE PROXIMOCARACTER
FIM ROTINA CONVERTE:
     POP R5
     POP R3
     POP R2
     POP R1
     POP R0
    RET
ROTINA DE LOGIN
;»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»
RotinaLogin:
     PUSH RO
     PUSH R1
```

PUSH R2

PUSH R3

PUSH R4

PUSH R5

PUSH R6

PUSH R7

CALL LimpaPerifericos ; LIMPA PERIFÉRICOS

MOV R2, MenuLogin ; COLOCA MENU DE LOGIN

CALL MostraDisplay ; MOSTRA NO DISPLAY O MENU

#### login Le ok:

CALL RotinaUser ; COLOCA O USERNAME DO PERIFERICO NO DISPLAY CARACTER A CARACTER

CALL RotinaPass ; COLOCA A PASSWORD DO PERIFERICO NO DISPLAY CARACTER A CARACTER, CONVERTENDO PARA '\*'

MOV RO, Ok

MOVB R1, [R0]

CMP R1, 0

JEQ login Le ok ; CICLO ATÉ O OK PASSAR A '1'

#### VerificarUserLogin:

MOV R1, BasedeDados ; INICIALIZA REGISTO DE BASE DA 1ª FICHA

MOV R2, numClientes ; NÚMERO DE FICHAS DE UTILIZADORES DA

PTZZARTA

MOV R3, tam ; NÚMERO DE BYTES DE CADA FICHA DE UTILIZADOR

MOV R4, Username ; USERNAME DO PERIFERICO

MOV R7, Nr Caracteres ; NUMERO DE CARACTERES A VERIFICAR

MOV R6, BasedeDados ; CÓPIA DO REGISTO DE BASE DA 1ª FICHA

#### login verifica prox caracter user:

MOVB R5, [R4] ; CARACTER DO USERNAME DO PERIFERICO

MOVB RO, [R1] ; CARACTER DO USERNAME DA BASE DE DADOS DE DETERMINADA FICHA

CMP R5, R0 ; COMPARA CARACTER DE AMBOS

JNE login procura basededados next ; PROXIMA FICHA DE CLIENTE

ADD R4, 1 ; PASSA A APONTAR PARA O ENDEREÇO DO PROXIMO

CARACTER DO USERNAME DO PERIFERICO

ADD R1, 1 ; PASSA A APONTAR PARA O ENDEREÇO DO PROXIMO CARACTER DO USERNAME DA BASE DE DADOS DE DETERMINADA FICHA

SUB R7, 1 ; MENOS UM CARACTER A VERIFICAR

CMP R7, 0 ; CHEGOU AO FIM DO USERNAME (8 caracteres)

JEQ VerificarPassLogin ; TODOS OS CARACTERES IGUAIS, USERNAME EXISTE, VERIFICA A PASS

JMP login verifica prox caracter user ; CICLO PARA VERIFICAR CARACTER A CARACTER

#### login procura basededados next:

ADD R6, R3 ; PASSA PARA A PROXIMA FICHA DE CLIENTE

MOV R1, R6 ; COLOCA A APONTAR PARA A PROXIMA FICHA DE CLIENTE

MOV R7, Nr Caracteres ; REPOE NUMERO DE CARACTERES A VERIFICAR

MOV R4, Username ; USERNAME DO PERIFERICO

SUB R2, 1 ; MENOS UM CLIENTE A VERIFICAR

CMP R2, 0 ; CHEGOU AO FIM DOS CLIENTES?

JNE login\_verifica\_prox\_caracter\_user ;NAO, PASSA VOLTA A COMPARAR OS CARACTERES

MOV R2, MenuUserErrado ;SIM, CHEGOU AO FIM E NAO ENCONTROU ESSE USERNAME

CALL RotinaErro ; ERRO E MOSTRA MENSAGEM DE USERNAME INVALIDO

#### VerificarPassLogin:

MOV R1, Password ; VOLTA A COLOCAR A APONTAR PARA O INICIO DA PASSWORD DO PERIFERICO

MOV R3, bd\_pass ; COLOCA EM R3 O INDICE DA PASSWORD DA BASE DE DADOS

ADD R3, R6 ; COLOCA EM R3 O INDICE DA PASSWORD DA BASE DE DADOS DA FICHA DE CLIENTE PRETENDIDA

MOV R7, Nr Caracteres ; REPOE NUMERO DE CARACTERES A VERIFICAR

#### login verifica prox caracter pass:

MOVB RO, [R1] ; CARACTER DA PASS DO PERIFERICO

MOVB R4, [R3] ; CARACTER DA PASSWORD DA BASE DE DADOS DE DETERMINADA FICHA

CMP R4, R0 ; COMPARA CARACTER DE AMBOS

JNE login pass errada; PASSWORD ERRADA

ADD R1, 1 ; PASSA A APONTAR PARA O ENDEREÇO DO PROXIMO CARACTER DA PASSWORD DO PERIFERICO

ADD R3, 1 ; PASSA A APONTAR PARA O ENDEREÇO DO PROXIMO CARACTER DA PASSWORD DA BASE DE DADOS DE DETERMINADA FICHA

SUB R7, 1 ; MENOS UM CARACTER A VERIFICAR

CMP R7, 0 ; CHEGOU AO FIM DA PASSWORD (8 caracteres)

JEQ ChamaRotinaCliente ; TODOS OS CARACTERES IGUAIS, PASSWORD CORRETA

JMP login\_verifica\_prox\_caracter\_pass ;CICLO PARA VERIFICAR
CARACTER A CARACTER

#### login pass errada:

MOV R2, MenuPassErrada

CALL RotinaErro ; MOSTRA MENSAGEM DE PASS ERRADA

#### ChamaRotinaCliente:

CALL RotinaCliente

POP R7

POP R6

POP R5

POP R4

POP R3

POP R2

POP R1

POP RO

RET

#### 

ROTINA REGISTO

RotinaRegisto:

CALL LimpaPerifericos ;LIMPA PERIFERICOS

MOV R2, MenuRegisto

CALL MostraDisplay ; MOSTRA MENU DE REGISTO

registo\_Le\_ok:

CALL RotinaUser ; COLOCA O USERNAME DO PERIFERICO NO DISPLAY CARACTER A CARACTER

CALL RotinaPass ; COLOCA A PASSWORD DO PERIFERICO NO DISPLAY CARACTER A CARACTER, CONVERTENDO PARA '\*'

MOV RO, Ok

MOVB R1, [R0]

CMP R1, 0

JEQ registo Le ok ; CICLO ATÉ O OK PASSAR A '1'

#### RotinaVerificarUserRegisto:

MOV R1, BasedeDados ; INICIALIZA REGISTO DE BASE DA 1ª FICHA

MOV R2, numClientes ; NÚMERO DE FICHAS DE UTILIZADORES DA PIZZARIA

MOV R3, tam ; NÚMERO DE BYTES DE CADA FICHA DE UTILIZADOR

MOV R4, Username ; USERNAME DO PERIFERICO

MOV R7, Nr Caracteres

MOV R6, R1 ; COPIA DO REGISTO DE BASE DA 1ª FICHA

#### registo\_verifica\_prox\_caracter\_user:

MOVB R5, [R4] ; CARACTER DO USERNAME DO PERIFERICO

MOVB RO, [R1] ; CARACTER DO USERNAME DA BASE DE DADOS DE DETERMINADA FICHA

CMP R5, R0 ; COMPARA CARACTER DE AMBOS

JNE registo\_procura\_basededados\_next ; VERIFICA PROXIMO CLIENTE

ADD R4, 1 ; PASSA A APONTAR PARA O ENDEREÇO DO PROXIMO CARACTER DO USERNAME DO PERIFERICO

ADD R1, 1 ; PASSA A APONTAR PARA O ENDEREÇO DO PROXIMO CARACTER DO USERNAME DA BASE DE DADOS DE DETERMINADA FICHA

SUB R7, 1 ; MENOS UM CARACTER A VERIFICAR

CMP R7, 0

JEQ registo\_user\_em\_uso ;USERNAME TOTALMENTE IGUAL A BASE DE DADOS, LOGO ESTA EM USO

JMP registo\_verifica\_prox\_caracter\_user ;CICLO PARA VERIFICAR
SE HA UM USERNAME IGUAL

#### registo\_user\_em\_uso:

MOV R2, MenuUserEmUso

CALL RotinaErro ; COLOCA NO DISPLAY A MENSAGEM DE UTILIZADOR EM USO

#### registo procura basededados next:

MOV R4, Username ; VOLTA A COLOCAR A APONTAR PARA O INICIO DO USERNAME DO PERIFERICO

MOV R7, Nr\_Caracteres ; REPOE

ADD R6, R3; PROXIMA FICHA DA BASE DE DADOS

MOV R1, R6 ; COLOCA EM R1 PARA AVALIAR EM CIMA

SUB R2, 1 ; MENOS UM CLIENTE A VERIFICAR

CMP R2, 0

JNE registo\_verifica\_prox\_caracter\_user ; CICLO DE VERIFICAR CARACTER A CARACTER

#### ; VERIFICA PASSWORD

MOV R1, Password ; PASSWORD DO PERIFERICO

MOV R2, 0 ; CONTADOR DE CARACTERES

#### registo verifica prox caracter pass:

MOVB R0, [R1] ; CARACTER DA PASSWORD DO PERIFERICO CMP R0, Terminador

JEQ registo pass invalida

ADD R2, 1 ; INCREMENTA CONTADOR DE CARACTERES

ADD R1, 1 ; PASSA A APONTAR PARA O PROXIMO CARACTER

CMP R2, 3 ; VERIFICA SE A PASSWORD TEM 3 CARACTERES

JNE registo\_verifica\_prox\_caracter\_pass ; CICLO PARA VERIFICAR SE TEM PELO MENOS 3 CARACTERES

#### AdicionaBasedeDados:

MOV R1, Username ; VOLTA A COLOCAR A APONTAR PARA O INICIO DO USERNAME DO PERIFERICO

MOV R3, BasedeDados ; COLOCA NOVAMENTE NO INICIO DA BASE DE DADOS

MOV R7, BasedeDados ; COPIA DO INICIO DA BASE DE DADOS MOV R6, tam ; TAMANHO DE CADA FICHA

#### proxima ficha basededados:

MOVB R5, [R3]

CMP R5, Terminador ; VERIFICA SE ESTA FICHA NAO ESTA OCUPADA

JEQ add prox caracter user ; ESTÁ LIVRE, PODE PREENCHER

ADD R3, R6 ; PASSA PARA A PROXIMA FICHA

ADD R7, R6 ; COPIA PARA USAR NO REGISTO

JMP proxima ficha basededados

#### add prox caracter user:

MOVB R0, [R1]

CMP RO, Terminador ; VERIFICA SE O USERNAME ACABOU

JEQ AdicionaPassBasedeDados ;SE ACABOU O USERNAME, PASSA PARA A PASSWORD

MOVB [R3], R0 ; COLOCA O PRIMEIRO CARACTER NA BASE DE DADOS

ADD R1, 1 ; PASSA AO PROXIMO CARACTER DO PERIFERICO

ADD R3, 1 ; PASSA A PROXIMA POSIÇÃO DA BASE DE DADOS

JMP add prox caracter user

#### registo pass invalida:

MOV R2, MenuPassErrada ; PASSWORD INVALIDA

CALL RotinaErro ; APRESENTA MENSAGEM DE PASSWORD INVALIDA

#### AdicionaPassBasedeDados:

MOV R1, Password ; VOLTA A COLOCAR A APONTAR PARA O INICIO DA PASSWORD DO PERIFERICO

MOV R3, bd\_pass ; COLOCA EM R3 O INDICE DA PASSWORD DA BASE DE DADOS

ADD R3, R7 ; COLOCA EM R3 O INDICE DA PASSWORD DA BASE DE DADOS DA FICHA DE CLIENTE PRETENDIDA

#### add\_prox\_caracter\_pass:

MOVB R0, [R1]

CMP RO, Terminador ; VERIFICA SE A PASS ACABOU

JEQ AdicionaSucess ; SE ACABOU A PASS, MENSAGEM DE

SUCESSO

MOVB [R3], R0 ; COLOCA O PRIMEIRO CARACTER DA PASS NA BASE DE DADOS

ADD R1, 1 ; PASSA AO PROXIMO CARACTER DA PASS

ADD R3, 1 ; PASSA A PROXIMA POSIÇÃO DA PASS DA BASE DE

DADOS

JMP add\_prox\_caracter\_pass ;CICLO DE INTRODUZIR A
PASSWORD NA BASE DE DADOS

```
AdicionaSucess:
    Call RotinaRegistoSucesso
ROTINA MOSTRA USERNAME NO DISPLAY
RotinaUser:
    PUSH RO
    PUSH R1
    PUSH R2
    MOV R1, Username
    MOV R2, UserNoDisplay
    registo_prox_caracter_user:
         MOVB RO, [R1]
         CMP R0, Terminador
         JEQ registo acaba user
         MOVB [R2], R0; coloca no display
         ADD R1, 1 ;passa a apontar para o endereço do proximo
caracter
         ADD R2, 1 ; passa a apontar para onde por o proximo
caracter
         JMP registo prox caracter user
    registo acaba user:
    POP R2
    POP R1
    POP RO
    RET
ROTINA MOSTRA PASSWORD NO DISPLAY COMO '*'
RotinaPass:
    PUSH RO
    PUSH R1
    PUSH R2
    PUSH R3
    MOV R1, Password
    MOV R2, PassNoDisplay
    registo prox caracter pass:
         \overline{MOVB} R0, [R1]
         CMP R0, Terminador
         JEQ registo acaba pass
         MOV R3, CaracterAsterisco
         MOVB [R2], R3; coloca no display o \star
         ADD R1, 1 ; passa a apontar para o endereço do proximo
caracter
         ADD R2, 1 ;passa a apontar para onde por o proximo
caracter
         JMP registo prox caracter pass
    registo acaba pass:
    POP R3
    POP R2
    POP R1
    POP RO
    RET
ROTINA MOSTRA DISPLAY
```

```
MostraDisplay:
    PUSH RO
    PUSH R1
    PUSH R2
    PUSH R3
    MOV RO, Display
    MOV R1, DisplayEnd
Ciclo:
    MOV R3, [R2]
    MOV [R0], R3
    ADD RO, 2
    ADD R2, 2
    CMP R0, R1
    JLE Ciclo
    POP R3
    POP R2
    POP R1
    POP RO
    RET
ROTINA LIMPA PERIFÉRICOS
LimpaPerifericos:
    PUSH RO
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    MOV R0, Username
    MOV R1, Password
    MOV R2, NR SEL
    MOV R3, Ok
    MOV R4, 0
    MOV R5, Nr_Caracteres
limpa proximo:
    MOVB [R0], R4
    MOVB [R1], R4
    SUB R5, 1
    ADD RO, 1
    ADD R1, 1
    CMP R5, 0
    JNE limpa proximo ; CICLO PARA LIMPAR USERNAME E PASSWORD
    MOVB [R2], R4
    MOVB [R3], R4
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET
ROTINA LIMPA DISPLAY
LimpaDisplay:
```

```
PUSH RO
    PUSH R1
    PUSH R2
    MOV RO, Display
    MOV R1, DisplayEnd
    MOV R2, CaracterVazio
CicloLimpa:
    MOVB [R0], R2
    ADD RO, 1
    CMP R0, R1
    JLE CicloLimpa
    POP R2
    POP R1
    POP R0
    RET
ROTINA ERRO OPCAO
RotinaErro:
    PUSH RO
    PUSH R1
    PUSH R2
    CALL MostraDisplay
    CALL LimpaPerifericos
    MOV RO, Ok
ERRO:
    MOVB R1, [R0]
    CMP R1, 1
    JNE ERRO
    POP R2
    POP R1
    POP R0
    JMP Prinpicio
REGISTO SUCESSO
RotinaRegistoSucesso:
    PUSH RO
    PUSH R1
    PUSH R2
    MOV R2, MenuUserRegistado
    CALL MostraDisplay
    CALL LimpaPerifericos
    MOV RO, Ok
SUCESSO:
    MOVB R1, [R0]
    CMP R1, 1
    JNE SUCESSO
    POP R2
    POP R1
    POP RO
    JMP RotinaCliente
    RET
```