

Arquitetura de Computadores

Trabalho Prático 3

Data de entrega e discussão: 24-05-2024

Faculdade de Ciências Exatas e da Engenharia

Ano letivo 2023/2024

2º Ano 2º Semestre

Pedro Brito 2022622

Índice

[Introdução 2](#_Toc167101761)

[Objetivos 2](#_Toc167101762)

[Desenvolvimento 2](#_Toc167101763)

[Mapa de Utilização da Memória 3](#_Toc167101764)

[Inicialização do Programa 3](#_Toc167101765)

[Comprar um PEPE 4](#_Toc167101766)

[Utilizar um PEPE 5](#_Toc167101767)

[Ver Stock 5](#_Toc167101768)

[Encerramento do Programa 5](#_Toc167101769)

[Tratamento de Erros 5](#_Toc167101770)

[Funções Extra 6](#_Toc167101771)

[Conclusão 6](#_Toc167101772)

[Anexo A – Fluxogramas 6](#_Toc167101773)

[Anexo B: Código em linguagem Assembly 6](#_Toc167101774)

# Introdução

Este relatório foi realizado no âmbito de descrever o trabalho realizado para o 2º trabalho prático de Arquitetura de Computadores. Este trabalho teve como principal objetivo implementar uma máquina de venda de bilhetes de metro em linguagem Assembly utilizando o processador PEPE de 16 bits – *figura 1*.

Para isto foi utilizado um simulador de arquitetura de computadores que é providenciado [aqui](https://grupos.ist.utl.pt/~livro-ac.daemon/). Este simulador é executado em ambiente JAVA.

# Objetivos

* Breve estudo dos requisitos de software para a elaboração do projeto
* Desenho dos principais fluxogramas do controlo e das rotinas secundárias
* Programação em linguagem assembly
* Verificação experimental do programa

# Desenvolvimento

Antes de iniciar a programação foi necessário mapear a utilização da memória. Foi decidido também como os dados das estações, dos pepes e dos menus deveriam ser guardados.

Para as estações foi decidido que o início dos endereços seria localizado no endereço 4512 ou 11A0 em hexadecimal. Cada estação utiliza duas linhas de 16 bytes, ou seja, 32 bytes no total. A primeira linha guarda o nome da estação e a segunda linha guarda nos primeiros 8 bytes os valores das moedas e das notas de forma seguida.

Para os Pepes decidiu-se guardar de forma sequencial em que cada PEPE utiliza 2 bytes, 1 byte para o índice e 1 byte para o valor do saldo. Isto quer dizer que o intervalo de valores, tanto de índices quanto de saldos, é de 1-255.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Endereço | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 11A0 | Nome da Estação 1 | | | | | | | | | | | | | | | |
| 11B0 | 0,1 | 0,2 | 0,5 | 1 | 2 | 5 | 10 | 20 |  |  |  |  |  |  |  |  |
| … | … | | | | | | | | | | | | | | | |
| 1200 | Nome da Estação 4 | | | | | | | | | | | | | | | |
| 1210 | 0,1 | 0,2 | 0,5 | 1 | 2 | 5 | 10 | 20 |  |  |  |  |  |  |  |  |

Tabela 1 - Endereçamento das Estações

Tabela 2 - Endereçamento dos PEPEs

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Endereço | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 3000 | | ID1 | B1 | ID2 | B2 | ID3 | B3 | ID4 | B4 | ID5 | B5 | ID6 | B6 | ID7 | B7 | ID8 | B8 |
| … | … | | | | | | | | | | | | | | | | |
| 31F0 | | ID248 | B248 | ID249 | B249 | ID250 | B250 | ID251 | B251 | ID252 | B252 | ID253 | B253 | ID254 | B254 | ID255 | B255 |

## Mapa de Utilização da Memória

|  |  |  |
| --- | --- | --- |
| Decimal | Hexadecimal | Utilização |
| 16 | 10 | Endereço para Troco |
| 32 | 20 | OK\_BUTTON |
| 33 | 21 | Periférico de entrada PER\_EN |
| 48 – 159 | 30 – 9F | Display |
| 176 - 1663 | B0 – 67F | Instruções |
| 1792 - 4367 | 700 – 110F | Menus |
| 4512 - 4639 | 11A0 – 121F | Estações |
| 8192 | 2000 | Stack Pointer |
| 12288 - 12784 | 3000 – 31F0 | Pepes |

## Inicialização do Programa

O programa só começa a sua inicialização se o OK\_BUTTON tiver o valor 1. Enquanto não estiver o valor 1 no endereço de memória dado para o OK\_BUTTON o programa não inicia. Após inserido o valor 1, o programa dá reset nos periféricos de entrada e o troco e invoca o menu inicial para o display.

Após mostrar o menu inicial, o programa fica à espera que o utilizador insira um valor para selecionar o que pretende fazer: comprar um PEPE (1), utilizar um PEPE (2), ver o Stock (3) ou sair do programa (4). Se o valor inserido não for nenhum destes, o programa continua no ciclo até que o utilizador introduza um valor válido.

## Comprar um PEPE

Se o utilizador colocar o valor 1, quer dizer que pretende comprar um novo cartão PEPE. Após isto, é mostrado o menu de seleção de estações. Criou-se 4 estações. Após seleção da estação, o programa guarda os valores do endereço da estação (onde estão guardados o stock de moedas e notas daquela estação), o preço do bilhete da estação e o endereço do menu dessa estação. Como as próximas rotinas são semelhantes utilizou-se um registo (R3) para diferenciar quando o utilizador quer usar um PEPE ou criar um PEPE. Se R3 for 0 significa que o programa deve criar um PEPE e, caso contrário, o programa deve avançar para usar um pepe. Isto foi realizado de modo a ter um código mais limpo e reutilizável. Depois disto é mostrado o menu dessa estação e o utilizador terá de selecionar o valor monetário a introduzir para pagar o bilhete e a criação do cartão. De seguida basta fazer a subtração entre o valor monetário e o valor do preço do bilhete e guardar esse valor na criação do pepe.

De seguida é necessário atualizar o número de moedas na estação. Como foi guardado os valores de dados da estação, basta atualizar, utilizando o endereço da estação, chegar ao endereço da moeda em específico (incrementar o endereço da estação com o número guardado da moeda) e incrementar o seu valor em 1.

Na criação de um novo pepe, o programa percorre todos os pepes criados (se existirem) ao somar 2 ao endereço do pepe anterior. Se o iterador que percorre os pepes, encontrar um índice com valor 0, sabe que é nesse endereço que deve ser guardado o novo valor pepe, visto que não há pepes com índices 0 (o primeiro PEPE tem índice 1). Após encontrar o endereço correto, guarda o valor do índice nesse endereço e no endereço seguinte guarda o valor da subtração realizada.

Por último é necessário mostrar ao utilizar o valor do troco realizado na operação. Para mostrar ao utilizar o valor do troco é necessário converter os valores hexadecimais em decimais. Para isto utilizou-se um ciclo para subtrair o valor hexadecimal em unidades, dezenas e centenas. Ou seja, guardou-se 4 registos a 0 para os valores das unidades decimais, unidades, dezenas e centenas que têm o valor máximo de 9. Se o valor em hexadecimal for superior a 999, incrementa-se o registo das centenas com 1 e de seguida subtrai-se 1000 ao valor hexadecimal. Executa-se o mesmo processo para as dezenas (se for superior a 99, diminui-se 100). Este processo é executado continuamente até o valor hexadecimal for inferior a 9. Se isto acontecer simplesmente atribui-se o valor hexadecimal ao registo das unidades decimais. De seguida, basta utilizar os endereços do display para colocar os valores dos quatro registos no display utilizando o menu do troco.

## Utilizar um PEPE

Na utilização do PEPE, o utilizador necessita de escolher um PEPE. A escolha do PEPE é realizada quando o utilizador escreve um índice e o programa procura, através de um ciclo, esse índice na base de dados dos PEPES. Quando o programa encontra um PEPE com o mesmo índice, retorna esse valor e mostra o valor monetário que esse PEPE tem guardado. Neste ciclo foi necessário colocar um ponto de paragem, visto que se não for feito, o programa continua a procurar pela memória toda até encontrar um valor e entra num ciclo possivelmente infinito. Assim, como se sabe que não há PEPEs com índice 0, pede-se ao programa que assim que chegar a um endereço de memória em que o seu valor é 0, deve retornar um erro que informa que não há PEPEs com esse endereço.

Após a escolha do PEPE, o utilizador deve escolher se pretende utilizar esse PEPE para comprar novas viagens ou se pretende carregar monetariamente o PEPE. Se pretende comprar novas viagens o utilizador terá de escolher a estação de metro e realiza a compra nessa estação e tem um processa semelhante ao comprar um novo PEPE. Nesta situação o R3, como referido anteriormente, serve para comprar viagens ao utilizar um cartão. Desta forma o programa sabe que não deve criar um PEPE, mas sim utilizar um. Se o utilizador pretende carregar monetariamente o PEPE, é apresentado um menu com os valores monetários disponíveis para carregamento. Após escolhido o valor, o programa soma ao saldo do PEPE o valor escolhido. No fim, o programa mostra o novo saldo do PEPE.

## Ver Stock

Na seleção de ver stock, é pedido ao utilizador para inserir uma *password* para ter acesso ao stock das estações de metro. Esta *password* é guardada como uma constante (9F) para efeitos de demonstração. O utilizador deve inserir no periférico de entrada a password. Se não coincidir, é apresentado um menu de erro. Se coincidir, o programa avança para a seleção das estações de forma semelhante ao que é feito nas situações anteriores. Após esta seleção é apresentado 4 páginas em que cada página mostra 2 moedas/notas. De notar que os valores guardados estão em hexadecimal. Assim, é necessário converter os valores para decimal, tal como já foi referido anteriormente só que neste caso, como são valores inteiros, criou-se uma rotina de modo a ter apenas 3 registos para unidades, dezenas e centenas (o processo é igual).

## Encerramento do Programa

Para encerrar, o programa limpa todos os valores dos periféricos e coloca o display em branco com o valor 2020H, que em ASCII é o valor em branco.

## Tratamento de Erros

Como em todos os programas é necessário existir o tratamento de erros.

O primeiro erro que foi necessário tratar foi no caso de se criar um pepe e o valor monetário escolhido é menor que o preço do bilhete da estação. Como isto, não é possível, é necessário invocar um erro de fundos insuficientes.

Outro erro que se tratou foi na seleção do PEPE. Aqui temos duas possibilidades: ou não há PEPEs criados ou o índice inserido é demasiado grande para os PEPEs criados. Para o primeiro caso basta comparar o endereço constante dos PEPEs (o primeiro endereço) com o valor 0, e se esta comparação for verdadeira deve-se propagar o erro de que não há PEPEs criados. No segundo caso, o primeiro caso já foi averiguado. Assim, ou há um PEPE com o índice inserido ou o índice inserido é grande demais. Isto pode ser averiguado com um ciclo em que se incrementa o endereço do PEPE em 2 e se, caso, encontre um valor 0, deve ser propagado o erro de que o índice do PEPE é inválido.

Por último temos o erro da *password* inválida que já foi falado anteriormente.

Após os erros serem mostrados ao utilizador, o utilizador deve selecionar a opção para voltar ao menu.

## Funções Extra

De acordo com as funções extra pedidas foram introduzidas as seguintes rotinas:

* Caso o valor monetário do pagamento seja igual, ou superior, ao valor do produto a compra é processada automaticamente, e caso seja inferior é gerado um erro;
* Botão ON/OFF para ligar a máquina;
* Promoção da primeira viagem ser grátis e o valor adicionado no saldo do PEPE;
* Introdução do valor monetário de 10 e 20 euros;
* Introdução de uma *password* para aceder ao stock;

# Conclusão

Com este trabalho foi possível atingir os objetivos de forma clara. O Assembly, sendo uma linguagem de baixíssimo nível, obrigou uma aprendizagem de manipulação de memória eficiente.

# Anexo A – Fluxogramas

# Anexo B: Código em linguagem Assembly