ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital

Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Tecnologias de Computadores 2018/2019

Trabalho Prático nº 1

Computador básico (PEPE 8)

Elaborado em: 05/12/2018
Francisco Gabriel Fonseca Mesquita nº193025

Índice

Lista de Figuras

1 Introdução

Ao longo deste trabalho vamos fazer uso de um simulador do processador PEPE 8 e da simulação 3.4 fornecida pelo professor para correr 2 códigos diferentes em assembly para a realização dos exercícios propostos na ficha.
Na realização deste trabalho temos como objetivo aprender a trabalhar com processador de 8 bits e também trabalhar com alguns comandos em assembly que este sabe interpretar.

2 1º programa (exercício 1)

2.1 Explicação do código:

Explicação do código usado no exercício 1 proposto na ficha:

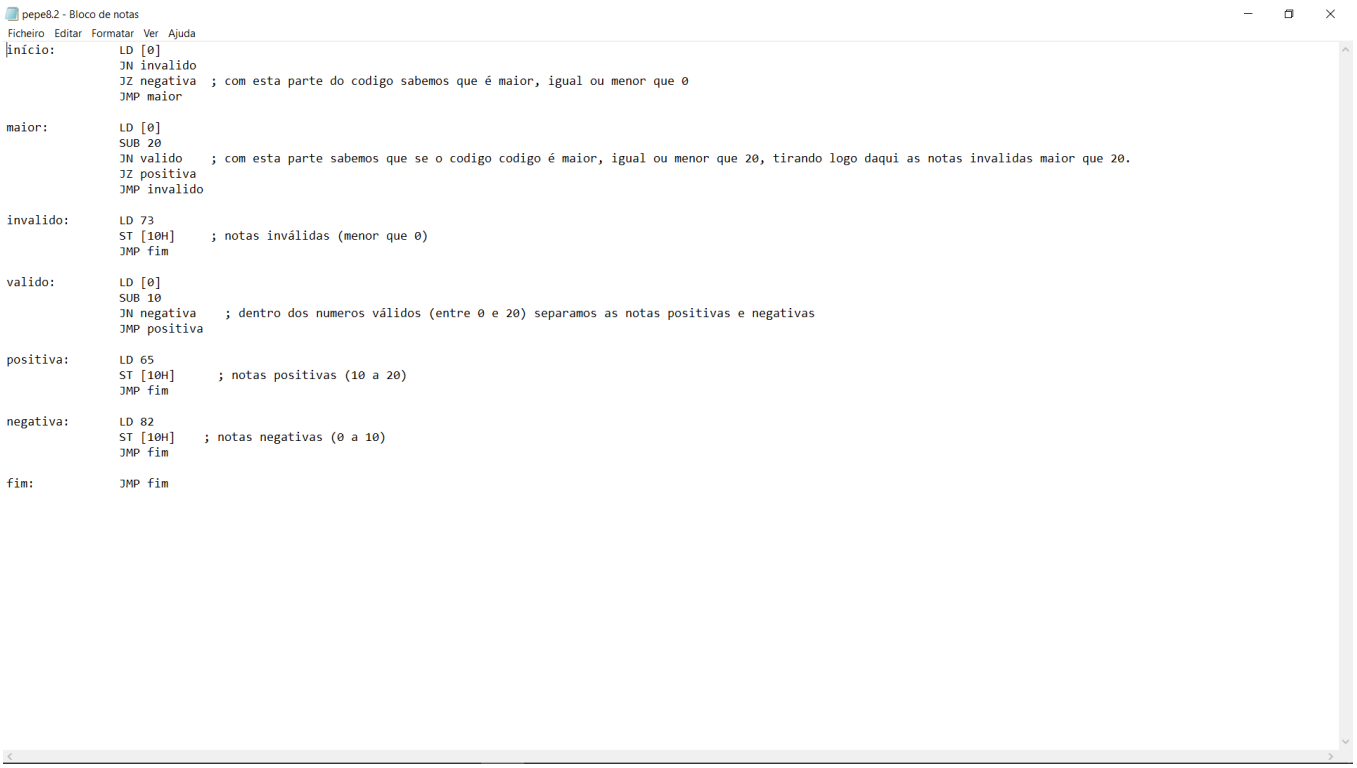


Figura1 – código assembly exercicio 1

Explicação do código:

- O código inicia na secção inicio, o LD [0] é responsável por carregar a memória e logo aí é possível através dos jump verificar se o número é negativo ou zero, se não o for passa então para a secção maior.

- Nesta parte do código vamos verificar se o número é maior que 20, através da subtração do número introduzido na memória por 20, se o resultado for 0 sabemos que a nota introduzida foi 20 ou seja positiva, se o resultado for menor que 0 sabemos que a nota introduzida está entre 0 e 20, se o resultado for maior que 0 então sabemos que a nota é maior que 20 ou seja é uma nota inválida.

- Caso o número introduzido pelo utilizador seja menor que 0 ou acima de 20, então o código vai para a secção invalido, depois carrega um valor para o registro e guarda esse valor na memória 10H fazendo logo de seguida o jump para o fim onde vai terminar o programa. No simulador o valor que foi carregado para o registro, neste caso foi o 73, vai mostrar o caracter I no simulador pois 73 é o valor decimal para I.

- Depois de testado se o número é maior ou menor que 0 falta apenas saber se a nota é positiva ou negativa, sendo assim dentro da secção valido foi feita a separação da nota positiva e da nota negativa, para isso subtrai 10 ao valor introduzido na memoria e se este der negativo sabemos que a nota é menor que 10 pois para x-10=-y temos que x tem de ser menor que 10. Se o valor de x (x é o valor introduzido pelo utilizador na memória) for 10 ou acima nunca vai dar negativo, ou seja, depois de chegar a esta conclusão usei o jump negativo para separar o negativo do positivo enviando a respetiva nota para uma secção positiva caso esta fosse 10 ou acima ou negativa caso esta fosse menor que 10

- Caso a nota introduzida for maior ou igual a 10 o código vai para a secção positiva, onde aí vai carregar o valor 65 no registro e guardar esse mesmo valor na memoria 10H, fazendo depois o jump final e terminando o programa. O valor 65 decimal corresponde ao caracter "A" de acordo com a tabela de ascii.

- Caso a nota introduzida for menor que 10, o código vai para a secção negativa, onde aí vai carregar o valor 82 e guardar esse mesmo valor na memória 10H, fazendo depois o jump final e terminando o programa. O valor 82 decimal corresponde ao caracter "R" de acordo com a tabela de ascii.

2.2 Demonstração do código:

Demonstração do código:

- Nota inválida (menor que 0 ou maior que 20):

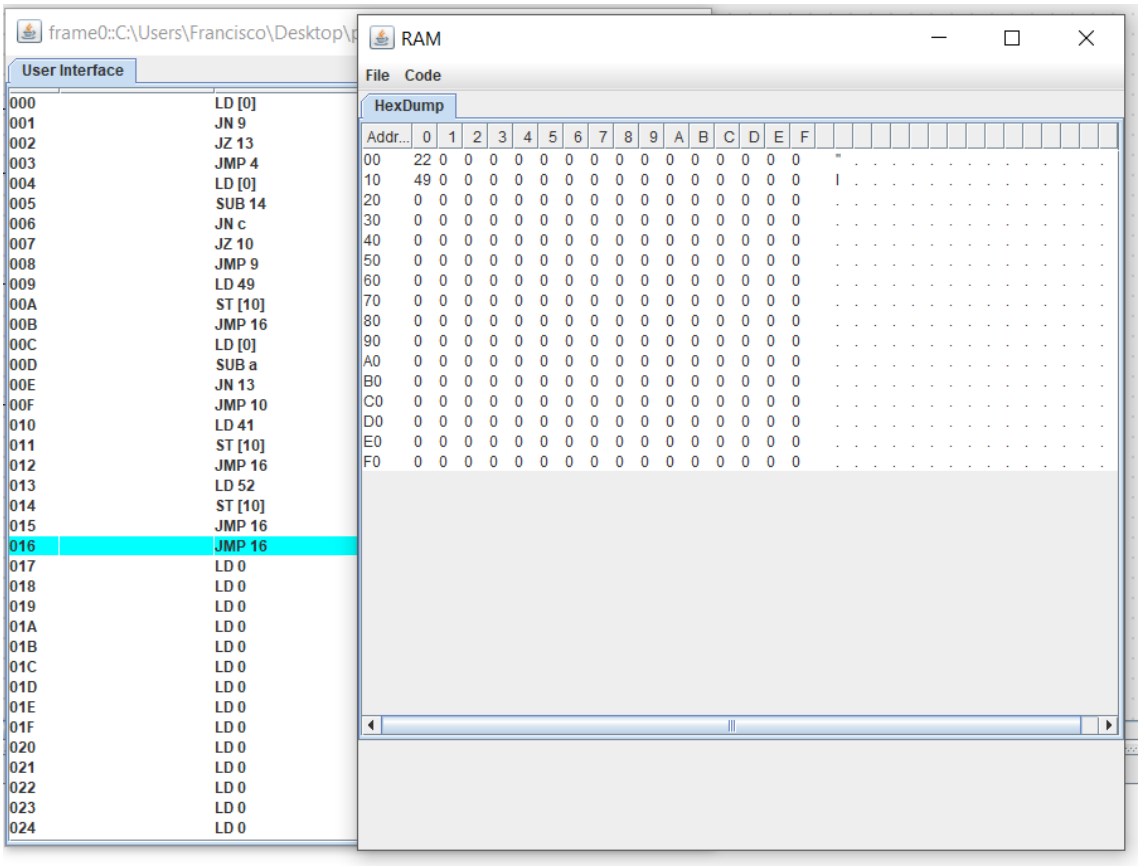


Figura11 – demonstração exercicio 1 nota inválida

- Nota negativa (maior ou igual a 0 e menor que 10):

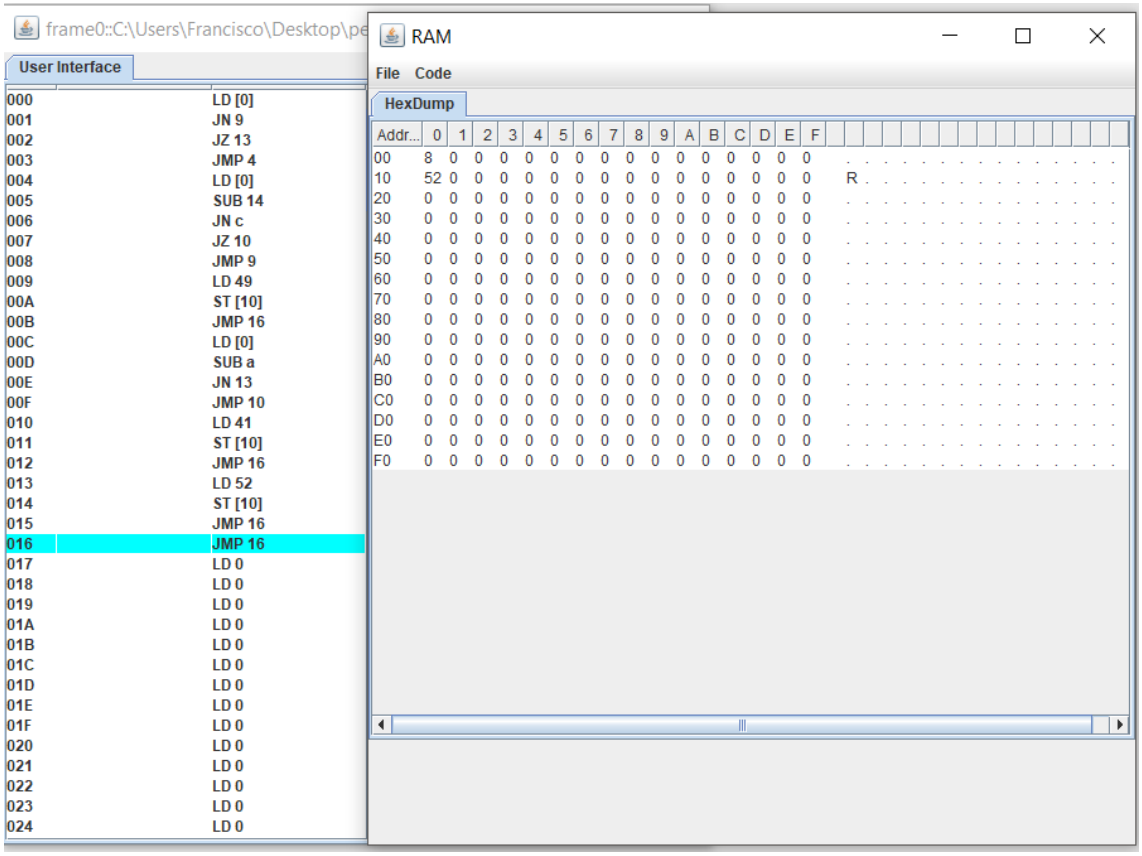


Figura111 – demonstração exercicio 1 nota negativa

- Nota positiva (maior ou igual a 10 e menor ou igual a 20)

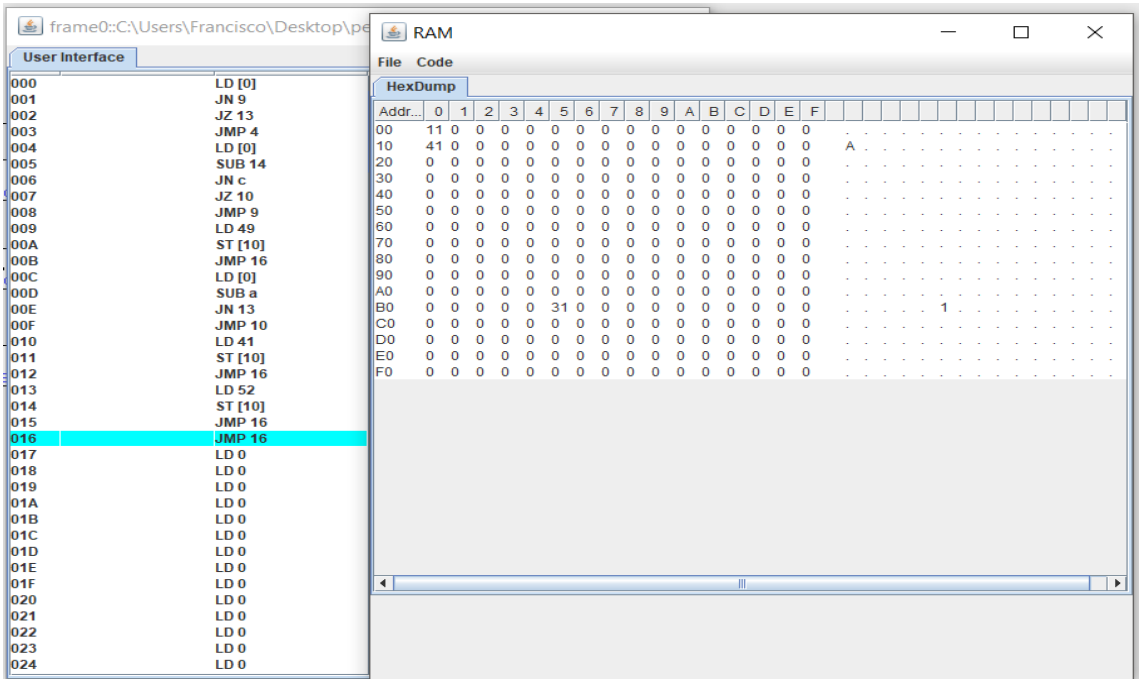
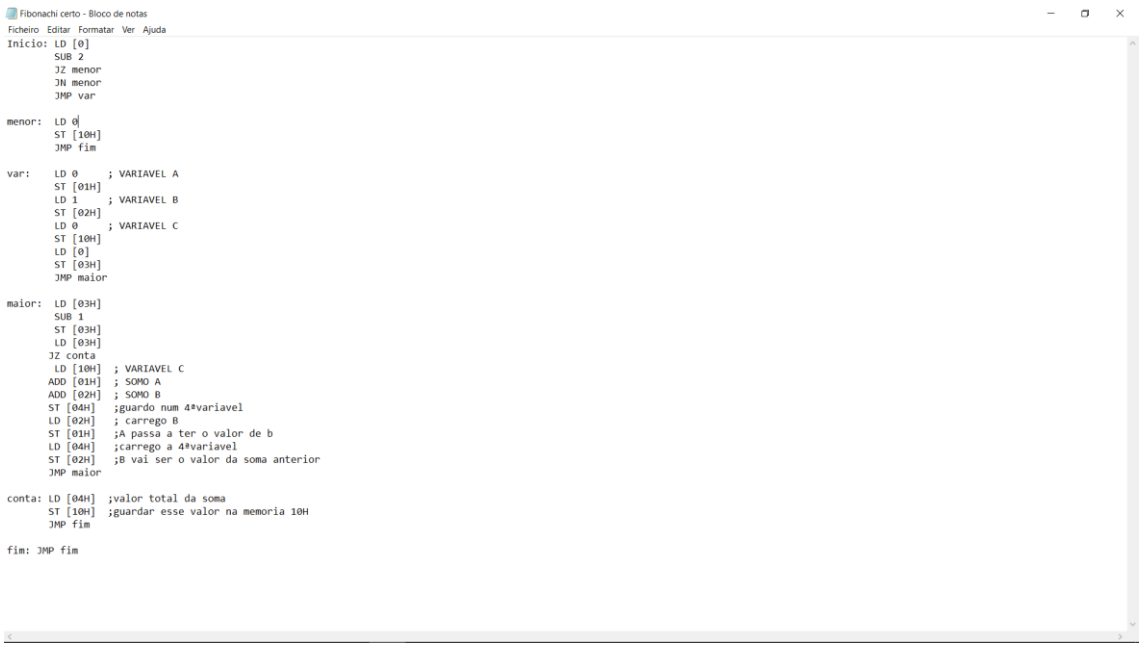


Figura1V – demonstração exercicio 1 nota positiva

3 2º programa (exercício 2)

3.1 Explicação do código:

Explicação do código usado no exercício 2 proposto na ficha:



FiguraV – código assembly exercicio 2

- O código começa por interpretar o número introduzido pelo utilizador para verificar se é maior que 2, para isso subtrai por 2 e se o numero for positivo então sabemos que o valor introduzido é maior que 2.

- Se o numero introduzido for menos que 2 então o codigo direciona-nos para a secção menor através do jump negativo e zero, ai é carregado o valor 0 e guardado na memoria 10H.

- Depois de verificado se o número é maior que 2 e se este realmente for maior que 2 então o código passa á secção var onde ai vai guardar valores em outras partes de memória, criando assim uma variável, são então criadas 4 variaveis nas memórias [01H], [02H],[03H] e [10H], fazendo de seguida jump para maior

- Já dentro da secção maior, começo por carregar o valor de [03H] que corresponde ao valor introduzido pelo utilizador pela igualdade feita acima, subtraio 1 a esse valor e volto a guardar na memória fazendo assim um contador em assembly que só irá parar quando [03H] for igual a zero e ativar o jump para zero, enquanto isso não acontecer o simulador não executa o jump zero e carrega assim o valor de [10H] que é 0 pela igualdade feita acima, depois a esse valor soma-se os valores guardados em [01H] e [O2H] que são 0 e 1 respetivamente (valores necessários para começar a executar a sequência de

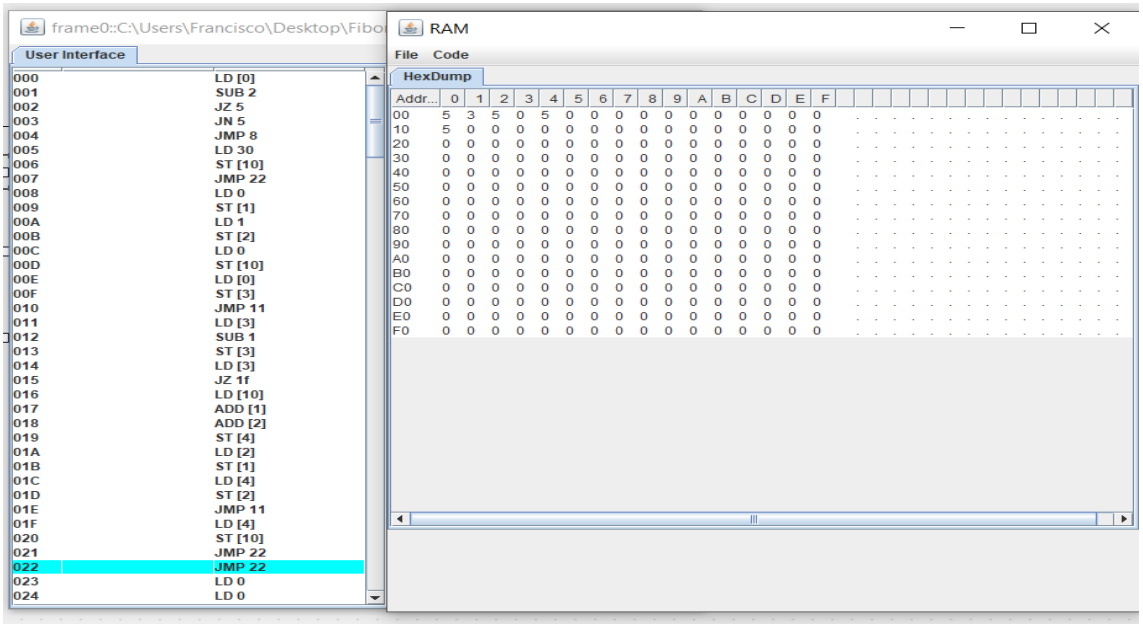
fibonacci), essa soma corresponde á soma dos dois termos anteriores que nos vai dar o termo seguinte na sequência de fibonacci, depois de feita a soma, o valor resultante é guardado em outra espaço da memória ainda não utilizado para facilitar o manuseamento desse valor. Depois disto só tenho que fazer com que os termos anteriores sejam o valor guardado em [02H] e o valor da soma que está guardado em [04H] para isso carrego o valor de [02H] e guardo-o em [01H] (passando este a ser o primeiro termo da soma) e carrego o valor guardado em [04H] e guardo-o em [02H] (passando este a ser o segundo termo da soma), depois disto fecho o loop com um jump de volta á secção maior. Enquanto o valor guardado em [03H] (que foi o valor introduzido pelo utilizador) não chegar a 0 então o loop estará sempre a correr e a gerar os numeros da sequência de fibonacci que serão armazenados em [04H].

- Quando o valor de [03H] chegar a 0 o loop termina e o programa passa então para a secção conta onde irá carregar o valor guardado em [04H] e guardá-lo em [10H] gerando assim o valor na sequência de fibonacci que corresponde ao valor introduzido pelo utilizador. Depois disso dá jump para o final onde finaliza o programa.

3.2 Demonstração do código:

Para este código irei então fazer a demonstração para F5 na sequência de fibonacci que corresponde ao termo 5.

Sequencia de fibonacci: 0,1, 1, 2, 3, **5(F5)**, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, ...



FiguraVI – demonstração exercicio 2 para F5

4 Conclusão

A realização deste trabalho foi fundamental para uma maior compreensão do funcionamento do pepe8 e serviu como forma de introdução á linguagem de programação assembly, que é uma linguagem utilizada para programar essencialmente para dispositivos computacionais como microprocessadores e microcontroladores.
Foram cumpridos todos os objetivos na realização do trabalho com algumas dificuldades no exercício 2, sendo este trabalho muito útil para aprofundar o conhecimento no processador PEPE8 e em assembly através dos exercícios propostos.

5 Referências

https://pt.wikipedia.org/wiki/Sequência_de_Fibonacci

https://pt.wikipedia.org/wiki/Assembly