Arquitetura e Sistemas de Computadores

Relatório do Trabalho Prático

Cifra de Vigenerè

2ºsemestre 2015/2016

Docente Prof. Teresa Gonçalves

Alunos: João Dias nº 35476

Eduardo Romão nº 35477

Índice

Introdução

No âmbito da unidade curricular de Arquitetura e Sistemas de Computadores, incorporada no programa de Licenciatura em Engenharia Informática, primeiro ano, semestre par, foi-nos pedido que elaborássemos um trabalho prático como instrumento de avaliação. Este trabalho consiste na implementação de um programa em linguagem Assembly Mips, que execute encriptações e desencriptações de um ficheiro utilizando a cifra de Vigenère.

No programa será aberto um ficheiro à escolha do utilizador, caso seja .txt será encriptado e caso seja .vig será desencriptado. A encriptação e desencriptação serão executadas letra a letra e através de uma palavra chave também introduzida pelo utilizador.

1. Carregamento de itens para memoria

No inicio do código escreveu-se .data para poder guardar os seguintes itens no segmento DATA em memoria.

Foi criado um espaço de 64 bytes para guardar os caracteres correspondentes ao nome do ficheiro. De seguida guardou-se cada extensão (.txt e .vig) de modo a ocuparem 4 bytes ou seja uma word, e criou-se mais um espaço de 4 bytes para guardar a extensão do ficheiro a abrir pelo utilizador.

No segmento de data em seguida foram criados três espaços com 512 bytes cada onde irão ser guardados a palavra chave, a frase presente no ficheiro e a frase já encriptada ou desencriptada que vai ser escrita no novo ficheiro.

Por ultimo foi guardado em memoria duas frases asciiz que vão receber o que vai aparecer antes dos inputs do utilizador, e mais duas frases asciiz que vão conter o nome de dois ficheiros, criados apos a encriptação ou desencriptação. Também foi guardado em memoria uma frase que representa uma mensagem de erro.

2. Inputs utilizador

O programa começa com duas funções que tem como objetivo receber inputs por parte do utilizador, o nome do ficheiro e a respetiva palavra chave.

2.1 *Input_file_name*

Argumentos: O primeiro argumento será o endereço de memoria onde irá ser guardado o nome do ficheiro introduzido pelo utilizador, e o segundo argumento será o numero de caracteres a ler no input do utilizador.

Objetivo: Esta função terá como objetivo receber para o espaço de memoria nome_ficheiro, o nome do ficheiro a ser aberto, introduzido pelo utilizador. Isto é feito através de uma instrução Syscall.

Retorno: Como retorno vamos ter em $v0 o valor 8 que faz com que a instrução Syscall recebe um input.

2.2 *Input_palavra_chave*

Argumentos: Nesta função os argumentos serão os mesmos que a função anterior exceptuando o facto de o input ser guardado no espaço de memoria correspondente a palavra chave.

Objetivo: Como a função anterior o objetivo é receber um input que neste caso será a palavra chave que executara a encriptação ou desencriptação.

3. Modificação do nome do ficheiro

Nesta parte do código será retirado o caracter "\n" do fim da string que contem o nome do ficheiro e a palavra chave. Caso \n não seja retirado do fim do nome do ficheiro, este não será aberto e o programa terminara em erro. Este caracter é predefinidamente colocado para indicar o fim da string.

3.1 *fim_de_frase*

Argumentos: A função *fim_de_frase* recebe como argumentos o primeiro endereço de memoria onde esta guardado o nome do ficheiro.

Objetivo: Esta função executa um loop carregando o caracter contido em cada endereço até esse caracter ser \n. Quando este caracter for detetado salta-se para uma parte da função que o substitui para um zero, logo eliminando-o. Não tem por isso nenhum valor de retorno.

4. Abrir ficheiro em modo de leitura

Já com o nome do ficheiro corretamente formatado e guardado em memoria, o próximo passo do programa será abrir o respetivo ficheiro mas em modo de leitura para poder ler o seu conteúdo. Para isso foram criadas três funções.

4.1 *abrir_ficheiro*

Argumentos: Os argumentos serão o endereço do nome do ficheiro e o modo de abertura (0 = leitura e 1 = escrita).

Objetivo: O único objetivo será a abertura do ficheiro em modo de leitura.

Retorno: Esta função retornará o numero 13 que é o correspondente á abertura de um ficheiro com a instrução Syscall e o descritor do ficheiro será guardado em $s6.

4.2 *ler_ficheiro*

Argumentos: Semelhante á primeira função o primeiro argumento será o descritor do ficheiro atualmente guardado em $s6, o segundo argumento será o endereço onde se começara a guardar o conteúdo do ficheiro e por ultimo será o numero de caracteres que se vai ler no ficheiro anteriormente aberto.

Objetivo: Como objetivo temos a leitura do conteúdo do ficheiro aberto, leitura essa que será guardada em memoria .DATA.

Retorno: Será o numero 14 que corresponde á função para ler na instrução Syscall.

4.3 *fechar_ficheiro*

Argumentos: A ultima função deste bloco recebe como argumentos apenas o descritor do ficheiro guardado em $s6.

Objetivo: O objetivo será o fecho do ficheiro em questão de modo a não sofrer alterações indesejadas.

Retorno: Como retorno teremos o numero 16 que representa a função que encerra um ficheiro com a instrução Syscall.

5. Modificação da palavra chave

Apos a leitura do ficheiro, é carregado o primeiro endereço da frase, da chave e do buffer "escrever" para as variáveis $t4, $t5 e $t6 respetivamente. Confirmando que a variável $t1 está a 0 o programa ira continuar saltando para a função comprimento, que usará a respetiva variável como um contador.

5.1 *Comprimento*

Argumentos: Esta função irá ter como argumento ($a0) o valor da variável $t4, que contem o primeiro endereço do local onde esta armazenada a frase lida no ficheiro.

Objetivo: A função tem como objetivo calcular o comprimento da frase lida no ficheiro. Isto é feito carregando o byte (caracter) correspondente ao endereço, até que o mesmo seja igual a 0x0a (\n). Até este caracter ser alcançado vai se somando uma unidade a variável $t1 que servirá como contador.

Retorno: Após a conclusão desta função, será retornado o valor do contador $t1 que representará o número de caracteres presentes no espaço de memoria correspondente a frase lida.

5.2 *Aumentar*

Argumentos: Como argumentos será recebido para esta função o primeiro endereço de memoria da palavra chave em duas variáveis diferentes ($a0 e $a1).

Objetivo: O objetivo será repetir a palavra chave até ser atingido um comprimento igual ao da frase lida. Usando $s3 que tem guardado o comprimento a atingir pela palavra chave guardado, irá ser feita uma comparação com o contador da função ($t1), de modo a saber se a palavra chave já tem o tamanho pretendido. O primeiro argumento servirá para fazer load do byte no endereço, enquanto que o segundo argumento será onde vai ser guardado o novo caracter. A diferença destes dois argumentos permanece no facto de ser feito um reset a $a0 para se ter sempre um caracter para fazer load, enquanto que ao segundo argumento irá ser sempre adicionado uma unidade para ir fazendo store do caracter, aumentado assim o comprimento da palavra chave.

Quando quer a palavra chave e a frase lida tiverem o mesmo numero de caracteres tem de ser certificado que todas as letras são maiúsculas ou minúsculas de modo a não haver erro quando for feita a encriptação ou desencriptação caracter a caracter. Neste programa foi optado por colocar todas as letras em maiúsculas, usando a função *maiúsculas* para a conversão.

5.3 *Maiúsculas*

Argumentos: Será recebido o primeiro endereço do espaço de memoria onde estará colocado a frase lida ou a palavra chave que se converterá de modo a conter apenas letras maiúsculas

Objetivo: Como foi enunciado acima a finalidade da função é converter letras minúsculas em maiúsculas. Para isto é carregado o byte corresponde ao endereço no argumento para a variável $t1. Se este byte for 0 significa que a frase chegou ao fim logo sair-se-á da função. Caso o byte seja maior que 0x61(a) significa que se estará na presença de uma letra minúscula, sendo por isso retirado 32 unidades ao caracter e fazendo se um novo store no mesmo endereço. O ciclo repete-se até se chegar ao fim da frase, não havendo por isso valores de retorno desta função.

6. Tipo de ficheiro

De modo a saber se o programa deveria executar uma encriptação ou desencriptação, foi decidido usar uma comparação de words. Esta comparação é feita entre a extensão do ficheiro e dois exemplos de extensões previamente guardadas em memoria.

Em primeiro lugar tem-se a função ponto que irá procurar no nome do ficheiro a localização de um ponto que indicará onde começa a extensão do respetivo ficheiro. Para isso a função repete-se até o byte carregado corresponder a 0x2e (.), passando de imediato para a função extensão, mas guardando em $a0 o endereço onde esta guardado o caracter ponto.

6.1 *Extensao*

Argumentos: Esta função terá como argumentos o endereço do ponto que estará guardado em $a0 devido à função anterior

Objetivo: Como objetivo esta função irá guardar o caracter ponto e os três seguintes caracteres do nome do ficheiro num espaço de memoria antes pré-designado (fich), não tendo por isso nenhum valor de retorno.

Estando a extensão do ficheiro guardada em memoria o programa passara para a sua comparação através de uma comparação de words, pois cada extensão terá o comprimento de 4 bytes logo 1 word

6.2 *Compara_extensao*

Argumentos: Como argumentos serão carregados para $a1, $a2 e $a3 os três primeiros endereços dos locais onde estão guardadas as três extensões.

Objetivo: A finalidade desta função será apenas o carregamento de uma word para as variáveis $t1, $t2 e $t3 dos endereços disponibilizados nos argumentos. Em seguida, através de dois branches irá se verificar se a extensão é igual a .txt ou .vig. Se for .txt o programa salta para a encriptação, caso contrario saltara para desencriptação.

7. Encriptação

Nesta parte do código será usada uma função principal (main_encripta) que se irá repetindo fazendo a encriptação da frase lida caracter a carácter. A função que fará a encriptação de um carácter será a *encripta.*

7.1 *Encripta*

Argumentos: Dois argumentos serão recebidos, que serão um byte da frase ($a0) e o correspondente byte da palavra chave ($a1).

Objetivo: A função irá encriptar o byte recebido de acordo com a cifra de vigenére. Para tal os bytes recebidos nos argumentos serão somados, e subtrai-se 65 que corresponde á posição decimal da tabela ASCII da letra A. Se a letra encriptada resultante for maior que 90 (Z) significara que é um overflow e as letras terão que se reiniciar, subtrai-se por isso 26 (tamanho do alfabeto).

Retorno: Nesta função ficara o valor de retorno o valor da deslocação da letra da frase lida com a letra gerada pela função encripta.

O valor retornado pela função encripta será, já na função main_desencripta, somado à letra da frase e guardado no endereço de memoria onde se irá guardar os caracteres encriptados da frase.

Depois de já ter todos os caracteres armazenados em memoria no buffer “escreve”, o respetivo conteúdo do buffer será colocado num novo ficheiro chamado “Encriptado.vig”. Para tal é aberto um ficheiro com o nome antes mencionado, e caso ele não exista é criado um novo. De seguida através de uma instrução Syscall com um valor de retorno 15, é escrito o conteúdo do buffer no novo ficheiro.

8. Desencriptação

Para executar a desencriptação de um ficheiro o método utilizado foi igual ao da parte da encriptação. Foi criado uma função principal chamada main_desencripta que também faz um loop para ir desencriptando carácter a caracter. Esta função é em tudo semelhante à função main_encriptado tirando o facto de chamar uma função para desencriptar (desencripta).

8.1 *Desencripta*

Argumentos: Esta função tem como argumentos dois bytes correspondentes a frase a desencriptada e á palavra chave.

Objetivo: A finalidade desta função como enunciado anteriormente, será descodificar a frase lida do ficheiro .vig, através da cifra de vigenere. Para isto é subtraído ao carácter da frase o valor do caracter da palavra chave. A este valor é adicionado o tamanho do alfabeto de modo a ser certificado que não temos uma letra "negativa" e também adicionado 65 que é o valor da primeira letra (A). Se a letra desencriptada não for maior que 90 (Z) não será necessário retirar 26 unidades, podendo por isso calcular o valor da deslocação da letra desencriptada.

Retorno: Este valor da deslocaçao da letra desencriptada pela letra encriptada será o valor de retorno desta função.

Para finalizado é também criado um ficheiro novo onde vão ser escritos todos os novos caracteres desencriptados. Este novo ficheiro vai se chamar "Desencriptado.txt".

Conclusão

Finalizando, apos vários testes ao nosso programa podemos concluir que ele executa todas as tarefas pedidas no enunciado do trabalho, quer seja encriptar um ficheiro .txt ou desencriptar um ficheiro .vig.

Este trabalho permitiu-nos aplicar os conceitos aprendidos na sala de aula e aprofundar o conhecimento da linguagem assembly enriquecendo assim o nosso conhecimento da unidade curricular de Arquitetura e Sistemas de Computadores.