Arquitetutra de Computadores
Relatório Fase II

Licenciaatura em Engenharia Informática

June 2021

Relatório de :
Diogo Mestre 48973
Rodrigo Alves 48681

1 Introdução

Na segunda parte do trabalho é feita a conversão do editor de texto feito em C, para assembly. Nesta última parte foram implementados todos os comandos do editor de texto.
Começamos por criar uma variável(palavra) que vai receber do utilizador o respectivo comando inserido pelo mesmo. Posteriormente será chamada a função caracter que irá identificar de que comando se trata. A função caracter irá devolver 1 se o comando corresponder à string que entra como argumento na função, a string foi inicializada na secção .data. No caso de retornar 1 é chamada de seguida a função que corresponde ao respetivo comando inserido.
Main-No inicio da main é feito um load address(la) do Array que vai guardar o endereço do Array na variável a0, depois é feito um jump and link(jal) para a função finalf que vai encontrar o final. De seguida é feito um move(mv) de a0 para a nova variavel s2 que vai representar a posiçao final, e a variavel s1 vai ser igual à posiçao inicial.
É feito um load address(la) da variável palavra que recebe o comando introduzido pelo utilizador na consola,este endereço é guardado na variável a0. A variável a1 vai representar o tamanho máximo da variavel palavra(100). De seguida, fazemos um jump and link(jal) para a função gets que permite obter a string a partir da consola. Guardamos na pilha a variavel a0 fazendo um store word.
Depois o programa vai comparar o conteúdo guardado em a0 com o conteúdo definido em a1, que representa um endereço de uma das strings definidas no .data, esta parte vai ser feita pela função caracter por isso fazemos um jump and link(jal) para a mesma que retorna a0 com o valor igual a 1 ou 0. A seguir é adicionada uma variável auxiliar t1 que vai conter o valor 1 fazendo assim um load 32 immediate(li), esta vai ser comparada com a variável a0 utilizando um branch not equal(bne), caso forem iguais vai ser feito um load address(la) do Array e vai ser guardado em a0, a variável s1 foi copiada e passada para a1 com um move(mv), a mesma situação foi feita entre a2 e s2. Caso forem diferentes passa para a instrução que se encontra na label do branch not equal(bne).
Logo a seguir foi feito um jump and link da respetiva função e depois fizemos um move(mv) de a0 para s1 e por último um salto incondicional para I (j I) quando for terminada a operação chamada.

2 Funções

No início do programa o array encontra-se vazio e portanto o valor da posição atual corresponde ao endereço do array e a final corresponde também ao início do array. A posição atual será guardada no registo s1 e a posição final será guardada no registo s2, assim, desta forma sempre que se chammar qualquer função estes valores não serão alterados, uma vez, que se tratam de registos que são respostos pelas mesmas após a execução. Algumas funções estão um pouco diferentes do programa inicialmente implementado em C.

Insert- Esta função permite inserir o texto antes da linha atual. É executada quando é inserido o comando i. Esta recebe como argumentos no a0 o endereço do array, no a1 o endereço da posição atual e no a2 o endereço da posição final do array. Começa por receber uma string e verificar se corresponde ao ponto, caso afirmativo ela sai da função. Para esta verificação é utilizada a função check. Caso a string inserida pelo utilizador seja diferente do ".", então quer dizer que estamos perante uma frase e neste caso a função irá calcular o tamanho da string inserida através de outra função auxiliar: a tamanho. Este tamanho será utilizado para mover todos os cracteres a começar no fim. Este deslocamento baseia-se em alterar o endereço final, ou seja, somar-lhe o tamanho da string e colocar nesse endereço o caracter correspondente ao endereço antes do deslocamento. Este processo é repetido até ao endereço da posição inicial. Desta forma, criámos o espaço necessário para inserir a string no nosso array. No fim irá retornar o novo endereço da posição final, uma vez que não altera o endereço da posição inicial.

Append- Esta função acrescenta o texto depois da linha actual. É executada quando é inserido no terminal o comando a. Esta recebe como argumentos no a0 o endereço do array, no a1 o endereço da posição atual e no a2 o endereço da posição final do array.

É muito similar à função Insert com a excepção que a posição inicial é alterada após a inserção de uma nova frase no array. Posição inicial esta que será a soma do endereço inicial que a função recebeu mais o tamanho da string inserida. A posição final é depois alterada quando retornar à main.

Change- Substitui a linha atual por um texto inserido pelo utilizador. Esta é executada quando é inserido na consola o comando c. Recebe como argumentos no a0 o endereço do array e no a1 o endereço da posição atual.

Nesta função o programa começa por verificar se a string inserida pelo utilizador é ou não um ponto, como faz nas anteriores. Caso não se trate de um ponto então o programa vai contar o numero de caracteres que existem nessa posição, ou seja, o número de caracteres do endereço atual(que corresponde ao 0 desse endereço) até ao próximo 0. De seguida vai ver se esse número é ou não inferior ao tamanho da string inserida. Se não houver espaço suficiente iremos deslocar para a direita a partir do endereço do fim até ao enderço atual, deslocamento este que vai ter de tamanho: caracteres da string-caracteres da posição atual. Após o deslocamento é feito o load no array da string. No caso em que há espaço na posição atual então a string será sobreposta ao que se situa nesses endereços.

Delete- Apaga a linha atual, ou seja, retira do array a última linha escrita

pelo utilizador. A função é executada quando é inserido na consola o comando d. Recebe como argumentos no a0 o endereço do array e no a1 o endereço da posição atual.
Esta começa por ver o número de caracteres do endereço da posição atual até ao primeiro 0 que encontrar, guardando o número de caracteres que essa posição contém, em seguida vai guardar o endereço do primeiro 0 depois do 0 do endereço atual e vai deslocar todos os carcateres até ao final para a esquerda com um deslocamento (número de endereços entres a posição atual e final) de: número de caracteres da string que vai ser removida.
Write- Escreve ficheiro no sistema de ficheiros, ou seja, passa todo o conteúdo do array para um ficheiro pré-existente. Esta função é executada quando é inserido no terminal o comando w.
Esta função começa por abrir o ficheiro já criado e abre-o em modo de escrita "w" pois o objetivo é passar tudo o que foi inserido pelo utilizador para esse mesmo documento. Depois de terminada a passagem do conteúdo, fecha o ficheiro, concluindo assim este processo. Além disso, esta função mostra o número de caracteres passados no terminal após o processo anterior ter terminado.
File- Guarda o nome do ficheiro no qual serão ou não escritos os nossos textos nas suas posições. Esta só é executada quando o comando for f.
Ao ser chamada esta função depois é necessário escrever o nome do ficheiro para que o programa o receba, pois este nome vai ser guardado. Se o utilizador chamar o comando w(write), esta função ja explicada anteriormente, vai escrever o conteuúdo que o utilizador escreveu na consola num ficheiro com o nome que está guardado na função file.
Edit-Abre um ficheiro ja existente para edição.Esta só é executada quando o comando for "e" seguido do nome do ficheiro, que será guardado em File-name. Esta função quando é chamada recebe o nome do ficheiro ja criado, ou seja, documentado. O objetivo da edit é substituir o conteúdo do ficheiro existente por tudo que foi inserido pelo utilizador na consola.
Print-Total- Esta função permite-nos imprimir para o terminal todos os textos já inseridos pelo utilizador nas suas respetivas posições. Recebe como argumento o endereço do array.
É executada quando é inserido no terminal o comando %p. Esta função utiliza a syscall PrintString e começa por imprimir a primeira string, depois vai percorrer char a char todos os caracteres até encontrar um 0, de seguida a função vai guardar o endereço desse 0 e imprimir a string que é iniciada pelo caracter que vem depois desse 0. Cada string é escrita no terminal em linhas consecutivas separadas. A função vai estar sempre a imprimir as strings até encontrar dois zeros seguidos.
Print- Imprime a linha atual, ou seja , a última linha inserida pelo utilizador no terminal. Esta é executada quando for inserido na consola o comando p.
Esta recebe como argumento o endereço da posição inicial e através da syscall PrintString, devolve a string a que esse endereço corresponde.
Quit- Esta apenas sai do programa sem guardar os dados. É executada quando é inserido o comando Q no terminal. A função ao ser chamada na consola encerra

o programa sem ter guardado o conteúdo introduzido pelo utilizador.
quit- Esta sai do programa e guarda os textos e as suas respetivas posições em linhas consecutivas do ficheiro, de nome igual ao File-name, o qual foi previamente definido. Esta só é executada quando é inserido o comando q na consola. Se for introduzido na consola o comando "q", este só encerra o programa quando todo o conteúdo escrito pelo utilizador estiver guardado num ficheiro,caso nao esteja guardado o programa não encerra, pois não pode ser efetuada enquanto o conteúdo não for guardado.
Numero-Esta indica a posição atual a partir de um número inserido na consola, recebendo como argumento esse mesmo número.
Em primeiro lugar é feito uma leitura do endereço, depois a função vai lendo posição a posicão até ser encontrado um 0, ao encontrar é somado 1 à variavel t1, quando t1 for igual ao número inserido o endereço dessa string é guardado e é dado como a posição atual.
Funções auxiliares:

Check- Função auxiliar que verifica se dois char são ou não iguais. Esta função é utilizada para averiguar se o texto que o utilizador inseriu é ou não o caracter(".").
A função recebe o endereço de memória da string e também o endereço de memória do ponto final definido numa string no ".data". Ao percorrer a string a função averigua posição a posiçao cada elemento da string, se o valor da string for igual a um ponto final, esta termina e devolve 1, caso contrário devolve 0.
Caracter- Serve para identificar que tipo de comando é chamado no terminal. Trata-se de uma função auxiliar que permite identificar que tipo de comando é chamado pelo utilizador na consola. Esta função recebe dois endereços de strings que vão ser lidas posição a posição, simultaneamente, até ao 0.Se forem diferentes esta função retorna 0, se forem iguais retorna 1.
Tamanho- Esta função auxiliar serve para encontrar o número de caracteres de uma determinada string. Recebe como argumentos um endereço de memória de uma string e só retorna à main quando encontrar o 0(fim da string).
Cifrão- Esta é muito similar á função Append (inserção de texto depois da linha atual), mas neste caso esta vai inserir depois da última linha de texto e não depois da atual. Apenas é executada quando é inserido o comando $a.
finalf-Função que encontra o final, ou seja, esta função vai percorrer a string ou o array com o intuito de encontrar o endereço final. Recebe como argumento o endereço ou de uma string ou do nosso Array. Termina quando encontrar dois zeros seguidos.
Gets-Esta recebe uma string a partir da consola.
Vai receber como argumentos: a0 é o endereço onde se quer guardar o que for introduzido na consola e o a1 o respetivo tamanho reservado.
Puts-Esta faz print de uma string.
Recebe como argumento o endereço da string e imprime-a na consola.

Ainda nos comandos que apresentam números foi necessário descodificá-los, para tal, decidimos ler caracter a caracter a string inserida pelo utilizador e guardá-los

em registos diferentes, de modo a que conseguissemos chamar a função correta com o endereço atual já atualizado.

3 Conclusão

Com este trabalho aprofundámos os conhecimentos que nos foram ensinados em aula. Na primeira fase do trabalho foi nos pedido que realizássemos o editor de texto em liguagem C, neste conseguimos realizar um protótipo do programa pedido apesar de não possuir todas as funções e funcionalidades, funcionava e não possuía nenhum erro. Nesta fase aceitámos a sugestão do professor utilizando um array de apontadores para guardar o texto inserido pelo utilizador nas respetivas posições.
Quando passámos para a segunda fase do trabalho, foi nos requerido a conversão do nosso protótipo, juntamente com todas as aplicabilidades obrigatórias. Foi nesta parte que começámos a ter mais problemas, nomeadamente, algumas funções apresentam erros que levam a que estas nunca párem de correr, ficando assim num loop infinito. Nós tentámos resolvê-las através de ebreaks mas não conseguimos perceber de onde provinham os erros. As funções que os apresentam são: Append, Insert, Change e a Delete. Nós suspeitamos que o erro poderá ter sido o facto de termos armazenado os valores em espaços na secção .data em vez de termos usado a syscall Sbrk para alocar o espaço. Como muitas destas funções são essenciais ao funcionamento do projeto este não corre na sua totalidade, apenas pede para inserirmos o comando, de seguida chama a função correspondente mas depois estas ficam em loops e não retornam à main, por esta razão decidimos criar outros ficheiros auxiliares para testarmos se as funçoes estavam a funcionar ou não. Estes ficheiros irão seguir em anexo.
Outro problema foi ainda o uso de muitos registos auxiliares que provocaram alguma confusão, o que levou a certos problemas, como: a passagem dos argumentos para as funções, a perda de informção possívelmente útil e dificuldade de leitura do código.
Uma vez que não tivémos tempo suficiente para fazer o debug ao programa suspeitamos que existam erros menores na main e noutras funções que podem provocar alguns problemas no desenrolar do programa.
Neste trabalho realizámos todas as funçoes requeridas, testando-as sem terem que depender daquelas que possuiam os erros e assim desta maneira tentar dimninuir o número de erros.
Outra dificuldade sentida foi tentar perceber o funcionamento em Risc-V dos dados dinâmicos que eram um fator importante para o desenrolar do programa.
É com muita pena nossa que não conseguimos realizar o trabalho de modo que este não possuísse erros e estivesse a funcionar perfeitamente. Devido à falta de tempo e ás tentativas falhadas para a resolução de certos problemas não foi possivél entregarmos o trabalho como perspetivámos inicialmente.