Disciplina: Arquitectura de Sistemas e Computadores
Docentes: Professor Miguel Barão e Professor Gaspar Brogueira
Curso: Engenharia Informática

Trabalho Elaborado Por:

Marlene Oliveira Nº 25999
João Aiveca Nº 26175

Ano Lectivo 2009/2010

No âmbito da disciplina de Arquitectura de Sistemas e Computadores I, foi-nos pedida a elaboração de um programa em MIPS, que permita tocar música a partir de um ficheiro de texto elaborado pelo utilizador. A melodia escrita deverá ser interpretada pelo programa e reproduzida na interface MIDI, que normalmente se encontra nas placas de som dos computadores.

Segundo o enunciado, o programa deverá satisfazer os seguintes requisitos:

I. Pedir ao utilizador a introdução de um nome de um ficheiro de música;
II. Abrir o ficheiro. Caso este não exista, o programa deverá apresentar uma mensagem de erro;
III. Ler a configuração presente na primeira linha do ficheiro (instrumento musical que se pretende usar para tocar a melodia e duração da nota mais curta que ocorre na música).

O código do nosso programa apresenta as funções main, Name, OpenFile, Read, Verify, newline, ToString, MIDI, reorder, error message e end.

A função main terá como principal objectivo controlar a execução de todas as outras funções e preservar todas as variáveis necessárias à execução das referidas funções. A função main não retorna nenhum valor nem recebe nenhum argumento. Começamos por guardar o endereço da informação que irá ser inserida pelo utilizador no registo $s0 e o valor do stack pointer actual no registo $s1. Seguidamente o programa executa a função Name. Quando volta ao main, o programa executa a função newline. Quando o programa voltar a regressar ao main, executará a função OpenFile. Seguidamente preservamos a stack actual e reservamos espaço na stack. O programa irá então executar a função Read. Quando regressar novamente ao main, o programa irá executar a função Verify. O programa irá então executar a função ToString, o que permitirá que o programa reconheça o instrumento a ser tocado. O valor que a função ToString devolve, correspondente ao instrumento musical, irá ser guardado no registo $s2. Depois disto, o programa irá executar de novo a função ToString que, desta vez, irá permitir que o programa reconheça a duração de cada nota musical. O valor correspondente à duração de cada nota será preservado no registo $s3. O programa irá ignorar o New Line e fará também o reset ao valor do registo $a1, o que fará com que este valor não gere erros nas funções reorder ou MIDI. Seguidamente o valor guardado em $s2, correspondente ao tempo de cada nota musical, será copiado para o registo $a2. O programa preservará então a stack actual e continuará a sua execução na função reorder, removendo New Lines e Carriage Returns contidos no ficheiro. Quando o programa regressa ao main, a stack actual é novamente preservada e é carregado o valor ox3C, correspondente a C, a base das notas musicais. O programa executará seguidamente a função MIDI, o que permitirá a interpretação da música contida no ficheiro de texto. Finalmente, o programa restaura a stack inicial e continua a sua execução na função end, onde irá terminar e sair.

A função Name terá como objectivo receber o input do utilizador para o caminho do ficheiro. Esta função receberá como argumentos o endereço do buffer de input (a stack), o número de caracteres a ler (sendo o máximo 128 caracteres) e o valor inteiro correspondente ao serviço a ser executado. A função retornará no endereço 0x10010000 o caminho do ficheiro que o utilizador indicou, com um newline no final. Na função Name é inicialmente carregado no registo $a0 o conteúdo do registo $s0, correspondente ao endereço de memória com o input do utilizador. Seguidamente é carregado em $a1 o número máximo de caracteres que a

String dada pelo utilizador pode possuir. Posteriormente é carregado o valor 8 no registo $v0, correspondente ao número do serviço a ser pedido ao sistema operativo (syscall 8). Finalmente, é executado o syscall 8, que lê uma String dada pelo utilizador, e o programa regressa ao main.

A função OpenFile terá como principal objectivo abrir o ficheiro indicado pela função Name. Esta função recebe como argumentos o endereço onde se encontra o caminho do ficheiro, a flag, o mode e o valor inteiro correspondente ao código do serviço a ser executado. A função retornará o descritor do ficheiro. Inicialmente, na função OpenFile, é carregado no registo $a0 o caminho do ficheiro de texto com a música introduzido pelo utilizador na função Name. Seguidamente carrega-se o valor inteiro zero no registo $a1 (correspondente às flags do syscall 13 – open file), o que faz com que o programa leia o conteúdo do ficheiro. É carregado também o valor inteiro zero no registo $a2 (correspondente ao mode do syscall 13 – open file). Posteriormente carrega-se no registo $v0 o valor inteiro 13 (syscall 13). Finalmente, executa-se o syscall 13, que abre o ficheiro cujo caminho foi previamente disponibilizado pelo utilizador. Seguidamente o programa regressa ao main.

A função Read tem por objectivo permitir que o ficheiro de texto que contém a música seja lido. Esta função recebe como argumentos o descritor do ficheiro, um buffer (com o tamanho de 1000 B), o número máximo de caracteres a ler (1000) e o valor inteiro correspondente ao código do serviço a ser executado. A função retornará o número de caracteres lidos. Inicialmente é copiado o descritor do ficheiro, inicialmente contido em $v0, para o registo $a0. O buffer contido no registo $sp é também copiado para um registo à parte, o registo $a1. No registo $a2 será carregado o número máximo de caracteres que podem constar no ficheiro de texto que contém a música, neste caso 51200 caracteres. O programa regressa então ao main.

A função Verify irá verificar se o ficheiro aberto é, na realidade, um ficheiro válido. Esta função terá como argumentos o primeiro carácter lido do ficheiro, o valor correspondente ao zero na tabela ASCII e o valor correspondente ao nove na tabela ASCII. Esta função não retornará nenhum valor. Inicialmente é carregado em $a0 o primeiro byte do ficheiro. Seguidamente a função verifica se o valor lido é nulo, significa que ocorreram erros ao abrir o programa. Quando isto se verifica, podemos concluir que o ficheiro não é válido. Assim, o programa passa para a função ErrorMessage, que devolve ao utilizador uma mensagem de erro e encerra o programa. Se o valor não for nulo, carregamos nos registos $t1 e $t3 os valores da tabela ASCII correspondentes, respectivamente, a 0 e a 9. Caso o valor carregado em $a0 seja menor que o valor contido em $a1, ou seja, menor que zero (tendo em conta a tabela ASCII), $t2 tomará o valor inteiro 1. Se o resultado anterior for verdadeiro significa que o carácter lido não é um número. Neste caso será apresentada uma mensagem de erro, uma vez que não estamos perante um ficheiro de texto correctamente formatado para ser lido por este programa. Caso o valor carregado em $a0 seja menor que o valor carregado em $a2,ou seja, menor que nove, $t2 tomará então o valor inteiro 1. Se o resultado anterior for falso, o carácter lido não é um número. Será então apresentada uma mensagem de erro, uma vez que não estamos perante um ficheiro de texto correctamente formatado para ser lido por este programa. Se o programa passar por todos estes testes e não apresentar uma mensagem de

erro, estamos perante um ficheiro de texto com a formatação correcta. De seguida, o programa regressa ao main.

A função newline tem como principal objectivo eliminar o New Line que foi carregado pela função read. Esta função recebe como argumentos o valor correspondente ao New Line na tabela ASCII (carregado no registo $a0) e o byte carregado na memória (carregado em $a1). A função newline não retornará nenhum valor. Começamos por preservar o valor do registo $s0, guardando-o temporariamente em $t7. Inicialmente carregamos o valor 0xa (New Line) no registo $a0 e o byte relevante no registo $a1. Seguidamente incrementamos a stack e verificamos se o byte contido em $a1 corresponde ao newline. Caso não se verifique esta relação, o ciclo repete-se. Caso o valor contido em $a1 corresponda ao New Line, o programa apaga aquele valor, colocando o valor null no local anteriormente ocupado pelo New Line. Finalmente, o programa restaura em $s0 o seu valor inicial e regressa ao main.

A função ToString terá como principal objectivo ler os números do ficheiro de música e convertê-los de Strings de caracteres para os seus correspondentes valores inteiros. Esta função recebe como argumentos o valor correspondente ao zero na tabela ASCII, o valor base de conversão para inteiro e o multiplicador da casa decimal. Esta função retornará o número convertido de String para inteiro. Inicialmente nesta função é efectuado o reset ao valor contido no registo $v0, ou seja, $v0 passará a ser igual a zero. Seguidamente é carregado em $a0 o valor correspondente ao zero na tabela ASCII, em $a1 é carregado o valor actual convertido em inteiro e em $a2 é carregado o multiplicador da casa decimal. Posteriormente o programa carrega o carácter do ficheiro em $t0 e incrementa a stack. Se o carácter lido não for um número, admitindo que o ficheiro é válido após ser submetido na função Verify, estamos perante um espaço ou um New Line, pois estes encontram-se abaixo dos números de 0 a 9 pela tabela ASCII. Neste caso, se o valor carregado é menor que 0x30 (correspondente ao espaço na tabela ASCII), a função termina e o programa regressa ao main. Caso contrário, passamos a numcicle, que procura qual foi o carácter lido. Ao incrementar o comparador (em $a0) e o valor (em $a1) até que a comparação dê verdadeira, podemos representar qualquer algarismo, e obter directamente o seu valor em inteiro em $a1. Ao encontrar o equivalente a um inteiro do algarismo carregado, passamos a cal. Esta label permite obter o resultado acumulado dos vários algarismos lidos, tento em conta que um algarismo á esquerda tem um valor 10 vezes superior. Assim, sempre que um novo algarismo é lido e o programa atinge este ponto, o valor de $v0 é decuplicado, para este assumir a sua casa decimal correcta. Após a multiplicação, é então somado o valor do algarismo lido. Então, é repetido o ciclo para um novo carácter lido, até que seja encontrado o valor nulo ou um newline.

A função MIDI tem como principal objectivo reproduzir a música contida no ficheiro de texto. Esta função recebe como argumentos a nota musical a tocar (guardada no registo $a0) que varia de acordo com o que está no ficheiro lido, a duração da nota musical (carregada em $a1 e em milissegundos) que varia com a nota lida do ficheiro, o instrumento musical a utilizar (guardado em $a2 e que depende do ficheiro lido), o volume (carregado no registo $a3), o " – " (guardado no registo $t1 e que corresponde ao valor 0x2d na tabela ASCII), o "b" (guardado no registo $t2 e que corresponde ao valor 0x62 na tabela ASCII), o "#" (guardado no registo $t3 e que corresponde ao valor 0x23 na tabela ASCII) e, finalmente, o valor 0x3a (guardado no registo $t4 e que corresponde ao valor acima de 9 na tabela ASCII). Esta função não retorna

qualquer valor. Inicialmente são carregados nos respectivos registos todos os argumentos que a função recebe, bem como o primeiro byte do ficheiro (carregado no registo $t0) e o valor correspondente ao código do serviço a ser executado (syscall 33 – MIDI out). É incrementada a stack. De seguida o programa verifica se o valor de $t0 é igual ao valor contido em $t1, ou seja, se o valor contido em $t0 é um traço (" – "). Se o valor não for um traço o programa executa o syscall. Se o valor de $t0 for um número, estamos perante uma oitava negativa. O programa continuará a sua execução na label "numcasenegative". Esta notação deve-se ao facto de desconhecermos que o Dó lido pelo utilizador (60 em ASCII) é, na verdade, o C3. Assim, este foi o método que encontrámos para representar escalas mais graves. Como só percebemos mais tarde que o Dó lido correspondia ao terceiro da escala, decidimos manter o programa intacto, e abordar este senão no relatório. O programa irá então retirar o valor correspondente ao número de oitavas que deverá descer, de acordo com o valor lido do ficheiro. Seguidamente o programa volta a executar a função MIDI. Se o valor não for um número, o programa restaura a duração contida no registo $s3 e, se esta duração for igual à contida no registo $a1, o programa volta a executar a função MIDI. Caso contrário, o programa reproduz a nota. Tal continua em ciclo até que seja encontrado o valor nulo. Ao encontrá-lo, o programa toca a última nota necessária, e regressa ao main.

A função reorder tem como principal objectivo retirar New Lines (0xa) e Carriage Returns (0xd) do ficheiro lido. Esta função recebe como argumentos o valor carregado do ficheiro (guardado em $t0), 0xa que corresponde ao valor do New Line na tabela ASCII (guardado em $t1), 0xd que corresponde ao valor do Carriage Return na tabela ASCII (guardado em $t2), o valor 0 que corresponde ao valor null na tabela ASCII (guardado em $t3) e a stack a preservar (guardada em $t4). A função reorder não retorna nenhum valor. Inicialmente é carregado um byte do ficheiro no registo $t0 e são também carregados todos os argumentos que a função recebe nos respectivos registos. Seguidamente comparamos o valor carregado em $t0 com o valor carregado em $t1 (New Line). Se o valor carregado for o New Line, ou seja, se $t0 é igual a $t1, o programa carrega outro byte do ficheiro e coloca-o no local da stack onde se encontrava o anterior e de seguida compara o valor carregado com o valor null (contido no registo $t3). Se o valor carregado corresponder ao null, o programa restaura a stack e volta a executar a função reorder. Caso contrário, o programa incrementa a stack e volta a executar o ciclo. Se o valor inicialmente carregado em $t0 corresponde ao carriage return, ou seja, se $t0 é igual a $t2, o programa carrega outro byte do ficheiro e coloca-o no local da stack onde se encontrava o anterior e de seguida compara o valor carregado com o valor null (contido no registo $t3). Se o valor carregado corresponder ao null, o programa restaura a stack e volta a executar a função reorder. Caso contrário, o programa incrementa a stack e volta a executar o ciclo. Se o valor inicialmente carregado em $t0 é igual ao valor null (carregado em $t3), ou seja, se $t0 é igual a $t3, o programa termina e volta ao main. Caso contrário, o programa incrementa a stack e volta a executar a função reorder.

A função ErrorMessage tem como objectivo principal retornar uma mensagem de erro. Esta função recebe como argumentos o valor correspondente ao serviço a ser executado (carregado no registo $v0) e a String que contém a mensagem a ser apresentada, que se encontra guardada em memória. A função ErrorMessage não retorna nenhum valor. Inicialmente é carregada uma mensagem de erro na String que se encontra na label "Erro". De seguida, é carregado em $v0 o valor inteiro 4 (correspondente ao código do serviço a ser

executado de seguida – Print String) e é carregado no registo $a0 o endereço da que contém a String com a mensagem de erro. Finalmente, o programa gera uma excepção e retorna a String e continua para a função End.

A função End tem como principal objectivo terminar a execução do programa. Esta função recebe como argumentos o valor do serviço a ser executado (carregado em $v0) e o descritor do ficheiro (carregado em $a1). A função End não retorna nenhum valor. Nesta função começamos por recuperar a stack do início do programa. Seguidamente carregamos o valor 16 no registo $v0 (correspondente ao código do serviço a ser executado seguidamente – Close File) e colocamos o descritor do ficheiro em $a0. Finalmente, gera-se a excepção e o programa termina.