Trabalho de Arquitectura de Sistemas e Computadores I

Licenciatura em Engenharia Informática

2009–2010

Resumo

Pretende-se com este trabalho desenvolver um programa em Assembly MIPS para tocar música lida de um ficheiro de texto. A melodia, descrita em texto ASCII, é interpretada pelo programa e tocada na interface MIDI habitualmente presente nas placas de som dos computadores pessoais.

1 Descrição do problema

A execução do programa deverá seguir os seguintes passos:

1. Pedir ao utilizador para introduzir o nome de um ficheiro de música.
2. Abrir o ficheiro, caso exista, ou apresentar mensagem de erro.
3. Ler a configuração existente na primeira linha do ficheiro.
4. Interpretar e tocar a música escrita nas restantes linhas do ficheiro.

Inicialmente o programa deve pedir ao utilizador que introduza o nome de um ficheiro. Caso o ficheiro não exista, deve ser apresentada uma mensagem de erro e é novamente solicitado ao utilizador que introduza o nome de um ficheiro.

Tendo aberto o ficheiro de texto, a primeira linha é lida e interpretada. Esta linha é formada por dois números inteiros separados por um espaço. O primeiro número permite seleccionar o instrumento musical que se pretende tocar. Alguns dos instrumentos musicais disponíveis estão indicados na tabela seguinte[1]:

0	Piano de cauda
4	Piano electrónico
12	Marimba
19	Órgão de igreja
40	Violino
43	Contrabaixo
53	Voz "Oohs"

O segundo número indica qual o intervalo de tempo correspondente à duração da nota mais curta que ocorre na música. A unidade de tempo usada é o milisegundo. As linhas seguintes descrevem as notas musicais e as respectivas durações.

Cada nota musical é representada por uma letra A a G seguida de uma sequência de um ou mais traços "`-`" correspontes à duração da nota. As letras A a G seguem a notação anglo-saxónica. A tabela seguinte mostra a correspondência com a notação usada em Portugal:

Notação portuguesa:	Dó	Ré	Mi	Fá	Sol	Lá	Si
Notação anglo-saxónica:	C	D	E	F	G	A	B

Todos os *newlines* a partir da segunda linha devem ser ignorados.

Para ilustrar o formato, considere o seguinte ficheiro:

[1] A lista completa de instrumentos suportados na interface MIDI pode ser consultada em: `http://www.midi.org/about-midi/gm/gm1sound.shtml`. Note que os códigos da tabela MIDI começam em um, enquanto os do MARS começam em zero.

```
0 200
C-C-D-C-F-E--
C-C-D-C-G-F--
```

A primeira linha "`0 200`" indica que o instrumento é um piano de cauda, e que cada traço corresponde a uma duração de 200ms.

As linhas seguintes são tocadas em sucessão. A segunda linha começa com um Dó com duração de 200ms, seguido de novo Dó com duração de 200ms, Ré 200ms, *etc.* A última nota "`E--`" é um Mi com duração de 400ms. A terceira linha segue o mesmo esquema e é tocada imediatamente a seguir à anterior.

1.1 Escala cromática

Além das sete notas apresentadas anteriormente existem mais cinco perfazendo um total de doze. As cinco notas adicionais situam-se entre algumas das restantes e não têm um nome próprio. Para as representar usa-se uma das notas com nome atribuído, Dó a Si ou C a B, e um símbolo que permita subir ou descer meio tom relativamente a estas. Para o efeito existem dois símbolos: o sustenido ♯ que permite subir meio-tom, e o bemol ♭ que permite baixar meio-tom. Por exemplo, a nota entre C e D pode ser representada por C♯ ou D♭.

A interface MIDI permite tocar todas estas notas. Para seleccionar a nota pretendida, é atribuído um código numérico a cada uma. A tabela seguinte apresenta os códigos da escala cromática:

Nota	Código
C	60
C♯, D♭	61
D	62
D♯, E♭	63
E	64
F	65
F♯, G♭	66
G	67
G♯, A♭	68
A	69
A♯, B♭	70
B	71

Para representar um sustenido ou bemol no ficheiro é usado um cardinal "`#`" ou a letra "`b`" minúscula. Por exemplo, a escala cromática pode ser tocada com o seguinte ficheiro:

```
0 500
C-C#-D-D#-E-F-F#-G-G#-A-A#-B-
```

ou equivalentemente

```
0 500
C-Db-D-Eb-E-F-Gb-G-Ab-A-Bb-B-
```

2 Implementação em MIPS

2.1 Interface MIDI

A interface MIDI pode ser usada a partir do simulador MARS usando o syscall 33. Este solicita que seja tocada uma nota particular, não podendo ser tocada outra até que a anterior termine. Os argumentos deste syscall são os seguintes:

- `$a0` código numérico da nota que se pretende tocar (0–127).
- `$a1` duração em milisegundos.
- `$a2` código do instrumento (0–127).
- `$a3` volume (0–127).

Por exemplo, o código assembly seguinte toca as notas C e D com durações de 500ms:

```
li $a0, 60     # selecciona nota C
li $a1, 500    # duracao de 500ms
li $a2, 0      # piano de cauda
li $a3, 100    # volume = 100 (max 127)
li $v0, 33     # syscall para MIDI out sincrono
syscall        # TOCA!!!
li $a0, 62     # selecciona nota D
syscall        # TOCA!!!
```

2.2 Leitura de Ficheiros

A leitura de ficheiros é efectuada da mesma maneira que em sistemas UNIX/LINUX com as funções `open`, `read` e `close`. Em particular, as barras existentes nos caminhos dos ficheiros são a "/" da tecla 7 (ex: `/home/joao/asc1/xpto.S`).

Para abrir um ficheiro para leitura pode usar-se o código seguinte:

```
la $a0, FILE         # string com o nome do ficheiro
li $a1, 0
li $a2, 0            # read only
li $v0, 13           # open
syscall              # v0 = descritor do ficheiro
```

No final, o registo `$v0` contém o descritor do ficheiro que será usado para as operações de leitura seguintes. A leitura do ficheiro é efectuada com um `read` da seguinte maneira:

```
# a0 = descritor do ficheiro
# a1 = buffer onde vai ser escrita a informacao lida
# a2 = numero de bytes a ler
li $v0, 14           # read
syscall
```

No final todos os ficheiros abertos devem ser fechados com a função `close`:

```
# a0 = descritor do ficheiro
li $v0, 16           # close
syscall
```

O código seguinte abre um ficheiro "teste.txt", lê o primeiro carácter e em seguida fecha o ficheiro:

```
        .data
NOME:   .asciiz "teste.txt"

        .text
main:   addiu $sp, $sp, -8
        ...

        la $a0, NOME
        li $a1, 0
        li $a2, 0
        li $v0, 13
        syscall        # ABRE FICHEIRO. v0 = descritor do ficheiro

        # Aqui deveria testar-se se ocorreu erro!

        move $a0, $v0  # descritor do ficheiro
        move $a1, $sp  # buffer de 4 bytes na pilha
        li $a2, 1      # ler 1 byte
        li $v0, 14     # read
        syscall        # LE 1 BYTE E COLOCA NA PILHA
```

```
li $v0, 16     # fecha ficheiro com descritor em $a0
...
```

3 Desenvolvimento do trabalho

Para o desenvolvimento deste trabalho deve seguir as seguintes indicações:

- O trabalho deve ser realizado no simulador MARS 3.8.
- O programa deve estar estruturado em funções com especificação clara dos argumentos e valor de retorno de cada função.
- Cada função deve vir acompanhada de um fluxograma que descreva os principais passos executados dentro de cada função.
- O código deve vir comentado. Em particular, devem estar marcadas as correspondências entre os blocos de código assembly e os blocos do fluxograma. O exemplo seguinte exemplifica como devem ser apresentados os comentários

```
#########################################################
# xpto - Esta funcao calcula a area de um retangulo
#
# Argumentos:
#  a0 - largura
#  a1 - comprimento
#
# Retorna:
#  v0 - area calculada
#########################################################
xpto:
    addiu $sp, $sp, -12
    ...

    # [Fluxograma] Multiplica largura e comprimento
    mult $a0, $a1
    ...
```

Para além da função `main`, aconselha-se a implementação das seguintes funções:

- Função que converte uma string num número inteiro.
- Função que pede o nome de um ficheiro ao utilizador e tenta abrir o ficheiro indicado. Deve devolver o descritor do ficheiro aberto. Em caso de erro, a função deve apresentar mensagem de erro e sair.
- Função que lê a configuração existente na primeira linha do ficheiro aberto.
- Função que interpreta e toca a música.

Para além destas, podem naturalmente ser desenvolvidas outras funções conforme apropriado.

4 Entrega do trabalho

O trabalho consiste no código a executar no simulador acompanhado de um relatório em formato PDF onde são indicadas todas as funções implementadas, os seus argumentos e valores de retorno, e ainda um fluxograma que mostre os principais passos executados em cada função. Os ficheiros deverão ser submetidos no moodle e comprimidos num dos formatos `tar.gz`, `tar.bz2`, `zip` ou `rar`.

Docentes: Miguel Barão e Gaspar Brogueira