66:20 Organización de Computadoras
Trabajo práctico 1: conjunto de instrucciones MIPS

1. Objetivos

Familiarizarse con el conjunto de instrucciones MIPS32 y el concepto de ABI[1], escribiendo un programa portable que resuelva el problema descripto en la sección 5.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 8), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo, usando la herramienta TEX/ LATEX.

4. Recursos

Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [2].

[1] Application binary interface

5. Programa

Se trata de una versión en lenguaje C de un programa que toma una matriz de números enteros y devuelve su traspuesta. El programa recibirá como argumento el nombre del archivo donde está especificada la matriz, y dará el resultado por `stdout` o lo escribirá en un archivo, según las opciones de entrada. De haber errores, los mensajes de error deberán salir exclusivamente por `stderr`. El formato de las matrices a trasponer será el siguiente: En la primera línea del archivo, el número de filas, seguido por un espacio, seguido por el número de columnas. En las siguientes líneas, el valor de cada columna, separados por espacios.

5.1. Comportamiento deseado

Primero, usamos la opción `-h` para ver el mensaje de ayuda:

```
$ traspuesta -h
Usage:
  traspuesta -h
  traspuesta -V
  traspuesta [options] filename
Options:
  -h, --help     Prints usage information.
  -V, --version  Prints version information.
  -o, --output   Path to output file.

Examples:
  traspuesta -o - mymatrix
```

Ejemplo por `stdout`:

```
$ cat mymatrix
4 3
1 2 3
4 5 6
7 8 9
10 11 12

$ ./traspuesta mymatrix
3 4
1 4 7 10
2 5 8 11
3 6 9 12

$
```

El programa deberá retornar un error si la matriz no cumple con el formato (por ejemplo tiene más o menos filas o columnas de las que dice el archivo), o si algún valor que lee está fuera del rango de los enteros. En ningún caso debe terminar por señales como `SIGSEGV`.

6. Implementación

El programa a implementar deberá satisfacer algunos requerimientos mínimos, que detallamos a continuación.

6.1. API

Gran parte del programa estará implementada en lenguaje C. Sin embargo, la función que traspone la matriz estará implementada en assembler MIPS32, para proveer soporte específico en nuestra plataforma principal de desarrollo, NetBSD/pmax.

La forma de la función que traspone la matriz será la siguiente:

```
int trasponer(unsigned int filas, unsigned int columnas,
              long long *entrada, long long *salida);
```

Es decir, la función toma como entrada el número de filas, el número de columnas y dos punteros a `long long`, de los cuales el primero es la matriz de entrada y el segundo es la matriz de salida, y tiene que copiar las filas de la matriz de entrada en las columnas de la matriz de salida.

El programa en C deberá procesar los argumentos de entrada, leer la matriz del archivo, asignar memoria a los punteros donde se alojarán las matricez, y escribir en `stdout` o un archivo el resultado.

6.2. Portabilidad

Pese a contenter fragmentos en assembler MIPS32, es necesario que la implementación desarrollada provea un grado mínimo de portabilidad.

Para satisfacer esto, el programa deberá proveer dos versiones de `trasponer`, incluyendo la versión MIPS32, pero también una versión C, pensada para dar soporte genérico a aquellos entornos que carezcan de una versión más específica.

6.3. ABI

El pasaje de parámetros entre el código C (`main()`, etc) y la función `trasponer`, en assembler, deberá hacerse usando la ABI explicada en clase: los argumentos correspondientes a los registros `$a0`-`$a3` serán almacenados por el *callee*, siempre, en los 16 bytes dedicados de la sección "function call argument area" [3].

7. Proceso de Compilación

En este trabajo, el desarrollo se hará parte en C y parte en lenguaje Assembler. Los programas escritos serán compilados o ensamblados según el caso, y posteriormente enlazados, utilizando las herramientas de GNU disponibles en el sistema NetBSD utilizado. Como resultado del enlace, se genera la aplicación ejecutable.

8. Informe

El informe deberá incluir:

- Este enunciado;
- Documentación relevante al diseño e implementación del programa, incluyendo un diagrama del stack;
- Instrucciones de compilación;
- Corridas de prueba para los archivos `matrix1`, `matrix2`, y `matrix3`;
- Diagramas del stack de las funciones `main` y `trasponer`;
- El código fuente completo del programa y del informe, en formato digital.

9. Fecha de entrega

La última fecha de entrega es el jueves 26 de abril de 2018.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] The NetBSD project, `http://www.netbsd.org/`.

[3] System V application binary interface, MIPS RISC processor supplement (third edition). Santa Cruz Operations, Inc.