Trabajo Práctico #1:

Conjunto de instrucciones MIPS

Julian Ferres, *Padrón Nro. 101483*
julianferres@gmail.com

Cecilia María Hortas, *Padrón Nro. 100687*
ceci.hortas@gmail.com

Matías Ezequiel Scakosky, *Padrón Nro. 99627*
scakosky@hotmail.com

2do. Cuatrimestre de 2018
66.20 Organización de Computadoras – Práctica Martes
Facultad de Ingeniería, Universidad de Buenos Aires

Abstract

Se propone como objetivo del Trabajo Práctico realizar un programa que utilice código en C y en Assembly MIPS para codificar y decodificar información en base64. Se utiliza el programa GXemul para simular el entorno de desarrollo, una máquina MIPS que corre una versión de NetBSD.

Contents

1 Introducción

Se propone escribir un programa cuya función principal se realice en C y desde allí se llame a subrutinas programadas en código MIPS32 que se encarguen de codificar o decodificar información de base64. Se cuentan con una serie de comandos básicos para el desarrollo del programa que serán detallados en el subtítulo de Implementacion. El objetivo principal del programa consiste en realizar una acción que puede ser codificar o decodificar a partir de un archivo de entrada y generar un archivo de salida. En caso de no recibir los nombres de archivos se utiliza por defecto los streams standar stdin y stdout. Esto mismo ya fue realizado en el Trabajo Práctico 0 pero la novedad radica en implementar dichas funciones en MIPS32.

2 Documentación relevante al diseño e implementación del programa

A partir de la consigna se determina que los comandos que debe incluir el programa son: `-h, --help Despliega el menú de ayuda -V, --version Imprime la versión y cierra el programa -i, --input Determina la ubicación del archivo de entrada -o, --output Determina la ubicación del archivo de salida -a, --action Determina la acción que ejecuta el programa: codificar (por defecto) o decodificar`

2.1 Diseño

El programa principal se encuentra desarrollado en la función `main`. Se detalla todo lo relativo al manejo de los comandos y se utiliza la libería `getopt.h`. Se guarda en el archivo `main.c` Las acciones de codificar y decodificar se separan en subrutinas distintas y se implementan siguiendo una lógica similar. Ambas están en los archivos `code.S`
y `decode.S`
respectivamente. Así mismo se desarrollan otras subrutinas para modularizar el código y que sea más claro. Se tiene entonces los archivos `write_c.S` para escribir un carácter, `read_c.S` para leer un carácter y `getb64index.S` Pueden encontrar el codigo de nuestro programa en el repositorio: `https://github.com/chortas/Orga-de-Compus-6620`[1]

[1] Allí se encuentran tres carpetas: TP0, TP1 e Información. Todo lo detallado en este informe se encuentra en la carpeta TP1.

2.2 Detalle de implementación

2.2.1 Subrutina encode

2.2.2 Subrutina decode

2.2.3 Subrutina write_c

2.2.4 Subrutina read_c

2.2.5 Subrutina geti64_c

2.2.6 Función principal main

3 Comandos para compilar el programa

En esta sección se detallan los pasos para compilar el programa en NetBSD a partir del entorno proporcionado por GXemul.

- Desde el directorio donde se instaló GXemul se corre el siguiente comando para bootear la imagen del disco patrón: `hostOS# ./gxemul -e 3max -d netbsd-pmax.img`
- Desde otra consola de linux se crea en el host OS con el usuario root un alias para la interfaz loopback (lo:0) con la IP 172.20.0.1 con el siguiente comando: `hostOS# ifconfig lo:0 172.20.0.1`
- Luego se ejecutan los siguientes comandos para la conexión contra la interfaz creada:

  ```
  hostOS# export TERM=xterm
  hostOS# ssh -p 2222 root@127.0.0.1
  ```

- Se transfieren los archivos a compilar a NetBSD con los siguientes comandos:

  ```
  scp -P2222 -r TP1 root@127.0.0.1:/root/TP0NetBSD
  ```

- Luego se ejecutan los siguientes comandos para realizar la compilación y extraer el código MIPS generado por el compilador en el sistema operativo que corre sobre GXemul:

  ```
  root@: # ls
  root@: # pwd
  root@: # mkdir TP1NetBSD
  root@: # cd TP1NetBSD
  root@: /TP1NetBSD/TP1# gcc -Wall -O0 -S -mrnames *.c
  ```

  - `-s` para detener el compilador luego de generar el código assembly.
  - `-mrnames` para cambiar los números de los registros por sus nombres de convención

- Luego se transfiere el archivo .s para el sistema operativo sobre el cual corre GXemul:

  ```
  hostOS# scp -P2222 root@127.0.0.1:/root/TP1NetBSD/TP1/*.s /home/user
  ```

4 Corridas de prueba

Las pruebas realizadas se basaron en los ejemplos del enunciado. Se probaron los comandos básicos como `-h` y `-V` para probar que muestren el resultado esperado. Luego, para probar los comandos `-a action -i input -o output` se realizó lo siguiente:

- Se probó que de omitir esos comandos la acción por default sea la ejecución de `encode` con la entrada por `stdin` y la salida generada por `stdout`.
- Se probó que de omitir nombres de archivos de `input` y `output` los archivos tomados por default eran los *stream* stándar.
- Se probó que de recibir los nombres de archivos de entrada y salida las acciones esperadas se concretaban. Se tomaron los ejemplos mencionados en el enunciado.
- Finalmente se ejecutó el siguiente comando en la terminal para verificar que archivos de tamaño creciente codificaban y decodificaban correctamente: `n=1;while :; do head -c n </dev/urandom >/tmp/in.bin; ./main -a encode -i /tmp/in.bin -o /tmp/out.b64; ./main -a decode -i /tmp/out.b64 -o /tmp/out.bin; if diff /tmp/in.bin /tmp/out.bin; then :; else echo ERROR: n; break; fi; echo ok:  n; n=((n+1)); rm -f /tmp/in.bin /tmp/out.b64 /tmp/out.bin; done`

5 Código fuente en C

5.1 main.c

6 Código implementado en MIPS32

6.1 code.s

6.2 decode.s

6.3 write_c.s

6.4 read_c.s

6.5 getb64index.s

7 Conclusiones

En conclusión, se cumplió el objetivo del Trabajo Práctico ya que se desarrolló el programa detallado por el enunciado. Los comandos del programa se ejecutan como detalla el comando `-h` y tras ser sometidos a distintas pruebas se concluye que funcionan según lo esperado. Así mismo fue posible la implementación del código en MIPS32 y se pudo utilizar el programa GXemul para simular un entorno de desarrollo de una maquina MIPS corriendo el sistema operativo NetBSD para ejecutarlo.

References

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

[2] J. L. Hennessy and D. A. Patterson, "Computer Architecture. A Quantitative Approach," 3ra Edición, Morgan Kaufmann Publishers, 2000.

[3] J. Larus and T. Ball, "Rewriting Executable Files to Mesure Program Behavior," Tech. Report 1083, Univ. of Wisconsin, 1992.

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

[5] Base64 (Wikipedia), http://en.wikipedia.org/wiki/Base64

[6] Base64 Enconde and Decode - Online, https://www.base64encode.org/

[7] Getopt Long Option Example (The GNU C library), https://www.gnu.org/

[8] MIPS RISC Processor, http://www.sco.com/developers/devspecs/mipsabi.pdf