Computación

Año 2015

Prof: T. S. Grigera — JTP: C. Grunfeld — AD: G. Sieben

Práctica 6 — Arquitectura y organización de computadoras II

Esta práctica abarca los siguientes temas:

- La unidad central de procesamiento (CPU) y la arquitectura de Von Neumann. Buses: comunicación con memoria y periféricos. Organización de una CPU: ALU, registros, control y bus interno. El ciclo de ejecución y las interrupciones. Lenguaje de máquina: frontera software/hardware (Instruction Set Arquitecture). Microcódigo. Instrucciones típicas de un ISA.
- El sistema operativo. Funciones del sistema operativo. Memoria virtual: páginas y fallas de página. Entrada/salida y control de periféricos ("drivers"). Procesos e hilos: contexto, alternancia de procesos, estados de un proceso. El cargador ("bootstrap loader") y el BIOS.
- Lenguaje ensamblador. La traducción o compilación. División de un programa ejecutable en módulos. Reubicación de módulos en memoria: el proceso de encadenado ("linking") de módulos. Bibliotecas. Lenguajes de alto nivel.

Bibliografía: Null y Lobur (2003, caps. 4 y 8), Tanenbaum (2006, caps. 3, 6 y 7).

Problema 1. Deténgase brevemente en la descripción de los buses PCI y USB de Tanenbaum (2006, secs. 3.6.2 y 3.6.4). Diga si se trata de buses en serie o paralelo. ¿Cuántos conectores utiliza cada bus? ¿Cuántos de ellos son para datos y cuántos para control?

Problema 2. La arquitectura Marie. MARIE es una arquitectura de computadoras inventada con fines pedagógicos por Null y Lobur (2003, cap. 4). Considere brevemente esta arquitectura, como ejemplo concreto de los conceptos discutidos en clase y responda las siguientes preguntas:

a) ¿Por qué el registro AC tiene 16 bits mientras que el MAR sólo cuenta con 12 bits? (vea las secs. 4.2.1 y 4.2.2)

b) Estudie el ciclo de ejecución y explique la función de las interrupciones (4.2.3).

c) Determine las operaciones de las siguientes instrucciones en el lenguaje de máquina de MARIE:

   i) 0010000000000111

   ii) 1001000000001011

   iii) 0011000000001001

Problema 3. Cierta computadora tiene una unidad de memoria con 24 bits por palabra y un conjunto de instrucciones de 150 operaciones. Todas las instrucciones tienen un campo para el código de operación y un código para la dirección (sólo se permite una dirección por instrucción). Cada instrucción ocupa una palabra.

a) ¿Cuántos bits se necesitan para el código de operación?

b) ¿Cuántos bits quedan para la dirección?

c) ¿Cuál es la cantidad de memoria que se puede direccionar?

d) ¿Cuál es el mayor número binario (sin signo) que se puede almacenar en una palabra?

Problema 4. Sistemas operativos, memoria virtual y procesos.

a) Cierta computadora tiene un espacio de direcciones de 32 bits y paginas de 4kB. ¿Cuántas páginas de memoria virtual hay?

b) Considerando los tres estados posibles de un proceso, a priori podría haber seis transiciones (dos salidas de cada estado). ¿Por qué hemos considerado sólo cuatro?

c) Averigüe cuáles son los posibles estados de un proceso en Linux (utilice Google o la página de manual del comando `ps`, accesible mediante `man ps`). Observará que son 7 en lugar de los 3 que hemos considerado nosotros. Notando que algunos de nuestro estados básicos estan subdivididos, haga una correspondencia entre los 7 estados de Linux y los 3 que hemos definido. Haga un diagrama indicando las transiciones que le parecen razonables; controle su respuesta vía Google.

Problema 5.

a) Considere un sistema operativo con una interfase gráfica (Windows, Gnome o KDE) y una interfase basada en línea de comandos (DOS o Linux shell). Explique cómo realizar las siguientes operaciones con cada una de las interfases consideradas: listar un directorio (carpeta), acceder a otro directorio, copiar un archivo, borrar un archivo. ¿Cuál de las dos interfases le parece más conveniente para un uso ocasional? ¿Y para un uso repetitivo?

b) Suponga que se tiene un proceso que produce diariamente cierta información (por ejemplo, facturas emitidas en el día) en un archivo guardado en cierto directorio. Todos los días es necesario mover ese archivo a otro directorio, cuyo nombre contenga la fecha correspondiente, y dejar el directorio original vacío para recibir el archivo del día siguiente. Indique cómo hacer esta operacion con las dos interfases consideradas. ¿Se le ocurre un modo de automatizar ese proceso?

Problema 6. En general, ¿es preferible un ejecutable reubicable en memoria, o no reubicable? ¿En qué circunstancias podría requerirse un ejecutable no reubicable?

Problema 7. Un "linker" podría proceder de la siguiente forma: antes de recorrer la biblioteca, el linker arma una lista de los módulos requeridos, examinando los símbolos "EXTERN" de los módulos objeto. Luego hace una sola pasada secuencial de la biblioteca, extrayendo los módulos solicitados a medida que aparecen. ¿Funciona siempre este procedimiento? ¿Qué requerimiento hay que imponer a la biblioteca para que el linker pueda funcionar con una sola pasada a la biblioteca?