

Arquitectura de Computadores 2

Obligatorio II – 2005

Facultad de Ingeniería
Instituto de Computación
Departamento de Arquitectura de Sistemas

1. Documentos y Archivos Necesarios

En la página web del curso están disponibles todos los documentos y archivos necesarios. Por favor, asegúrese de contar con los mismos:

- a. Letra del obligatorio (*arqsis2_oblig2_2005.doc*)
- b. Referencia de laboratorio de Tklsem (*labman.pdf*)
- c. Tarjeta de referencia assembler SPARC (*refcard.pdf*)
- d. Referencia SPARC del curso
- e. RPM del Tklsem
- f. RPM de las “binutils” para SPARC
- g. Archivo para recompilar ROM, ejemplos y archivo para inclusión “gx.h” (*isem_rom.tar.gz*)

2. Objetivo del Trabajo

El presente trabajo obligatorio tiene como objetivo introducir al estudiante al conocimiento práctico de la arquitectura SPARC. Se utiliza un ambiente simulado como plataforma de programación ya que el hardware SPARC normalmente es inaccesible debido a las protecciones del Sistema Operativo.

El simulador (Tklsem) contiene un módulo de software que se ejecuta en el momento de iniciar el simulador. Este módulo de software se ejecuta en modo supervisor y realiza tareas de inicialización antes de ceder el control al software de modo usuario.

Este código supervisor también brinda algunos servicios básicos de tipo entrada/salida (gráfica y de consola) e implementa un timer.

En este trabajo trataremos de desarrollar un despachador de tareas, el cual podría ser utilizado, por ejemplo, como parte de una biblioteca de threading o como parte de un sistema operativo.

El trabajo constará de dos módulos; en primer lugar se extenderá la funcionalidad del código supervisor, y posteriormente se utilizará esta extensión para desarrollar alguna aplicación.

a) *Extensión del Código Supervisor*

El despachador de tareas se ejecutará en modo supervisor, por lo cual habrá que modificar el código de supervisor provisto junto con el simulador. Se sugiere leer detenidamente la documentación y los ejemplos para ver cómo trabajar con el código de supervisor.

En el módulo de supervisor entonces se deberán realizar todas las inicializaciones necesarias, y proporcionar distintos servicios al código de usuario, los cuales implementarán en forma de siete nuevos traps que se agregarán al código de supervisor:

1. **init (trap 100)**

La trap 100 realizará todas las tareas de inicialización necesarias, como se comentó anteriormente (inicialización de estructuras, timer, etc.)

2. **agregarTarea (trap 101)**

La trap 101 brindará el servicio de “instalar” una nueva rutina a ser ejecutada por el despachador de tareas. Esta rutina necesita recibir algunos parámetros, los cuales recibirá en los registros de entrada:

- i. %i0: dirección de la primera instrucción ejecutable de la rutina a instalar
- ii. %i1: un nivel de prioridad de la tarea. Este nivel le permitirá al despachador darle más tiempo de CPU a una tarea que a las demás.

Y retornará en el registro %i2 el identificador de la tarea.

3. **arrancarTarea (trap 102)**

La trap 102 permite modificar el estado de una tarea de forma que el despachador la considere al momento de realizar la asignación de CPU. Esta rutina recibirá en %i0 el identificador de la tarea.

4. **detenerTarea (trap 103)**

La trap 103 permite modificar el estado de una tarea de forma que el despachador, por el momento, no lo considere como despachable. Recibe en %i0 el identificador de la tarea

5. **arrancarTodasTareas (trap 104)**

Idem a la trap 102 pero arrancando todas las tareas registradas para despacharse.

6. **cederCPU (trap 105)**

Cede el tiempo de CPU, de forma que el despachador asigne otra tarea a ejecutar.

7. eliminarTarea (trap 106)

La trap 106 le brinda a las tareas registradas una forma de terminar. Una tarea que ejecuta la trap 106 será quitada del proceso de despacho. Recibe en %i0 el identificador de la tarea.

Todas las tareas a ser ejecutadas por el despachador son leaf. El despachador deberá asegurar que todas las tareas se ejecuten adecuadamente, respetando sus respectivos contextos y las prioridades configuradas. Para esto se utilizará un algoritmo de despacho que se base en Round Robin, asignando stots de tiempo mayor a las tareas más prioritarias.

b) Aplicación

Para poder demostrar el funcionamiento del despachador se programarán algunas tareas para ejecutar en paralelo.

Realizar una aplicación que desplace horizontalmente una pequeña víbora de dos cuadros de largo sobre el dispositivo gráfico GX (ver documentación).

La idea de esta aplicación es correr dos o más instancias de la misma con prioridades iguales y/o diferentes y poder comparar el avance de la víbora en cada caso.

Hay que tener en cuenta las posibles condiciones que puedan darse sobre el dispositivo gráfico GX, ya que si varias tareas mezclan comandos sobre el dispositivo, el resultado a obtener en la pantalla es impredecible.

3. El Simulador “TkIsem”

En este obligatorio utilizaremos el emulador “TkIsem” (*Tk - based Instructional SPARC Emulator*). El emulador provee una simulación razonablemente completa de las arquitecturas SPARC versión 7 y 8, incluyendo la emulación casi completa de la unidad de punto flotante y de la unidad de manejo de memoria (MMU)

TkIsem también provee la simulación de hardware mínimo como ser una consola de I/O orientada a carácter (que utilizaremos para ver la salida de nuestros programas), un dispositivo gráfico (GX), un puerto serial (UART) y un temporizador (timer).

De la página del curso podrán descargarse paquetes RPM listos para instalar en Linux Redhat 8 y 9. Es posible que los mismos puedan funcionar en versiones anteriores de Redhat así como en otras distribuciones de Linux, pero no ha sido probado.

Es posible también descargar el código fuente y recompilarlo. Sin embargo, debe tenerse en cuenta que para que la versión 4.3 del TkIsem

que es el que vamos a utilizar compile correctamente con el compilador **gcc** versión 3.x se debe introducir algunos retoques al código fuente. Se recomienda recompilar solo en caso de ser estrictamente necesario y en ese caso hacerlo a partir de los SRPM que también estarán disponibles en la página del curso.

Además del emulador son necesarias las “*binutils*” de GNU, compiladas para cross-ensamblar arquitectura SPARC (sun4). Las *binutils* proveen el ensamblador y el linker SPARC que necesitamos para crear binarios ejecutables de arquitectura SPARC. En la página del curso también está disponible un RPM con estas herramientas.

Tklsem permite cargar un binario SPARC y ejecutarlo en forma continua o paso a paso permitiendo visualizar el conjunto de registros (una ventana a la vez), las áreas de memoria (user text, user data, supervisor text y supervisor data) y también permite configurar *breakpoints* (puntos de parada de la ejecución). En el momento de desarrollar la tarea propuesta, se recomienda utilizar estas facilidades para poder comprender mejor el código y poder depurar el mismo.

4. Forma de Entrega

La entrega de la tarea debe realizarse a través del mismo formulario Web utilizado en la primera entrega.

La entrega del obligatorio debe ser un archivo empaquetado de nombre **obligatorio2.tar.gz** que debe contener los siguientes archivos:

- un archivo fuente de nombre “**handler.s**”, donde se considerarán los comentarios que aparezcan con el fin de facilitar la comprensión del mismo. Este archivo fuente contendrá el código del handler de las traps introducidas. (cambio de rutinas de servicios).
- un archivo fuente de nombre “**tareas.s**” conteniendo la implementación de la aplicación de ejemplo descrita en la Sección 2.b.
- un archivo fuente de nombre “**isem_vec.s**” conteniendo el código del vector de traps.
- dos archivos “**makefile**” para facilitar compilación y linkedición: uno para la aplicación y uno para la ROM.
- un documento “**oblig2.pdf**” con la documentación sobre la implementación (pseudo-código, decisiones de implementación, etc).

Una clara, concisa y descriptiva documentación es clave para la evaluación de los trabajos entregados. No deje la documentación para el final.

La entrega se realizará a través del sitio Web del curso, en la página <http://www.fing.edu.uy/cgi-bin/arqsis2/entrega.pl>

5. Fecha de Entrega y Consideraciones Finales

Los trabajos deberán ser entregados indefectiblemente antes del lunes 21 de noviembre a las 23.30 horas. No se aceptará ningún trabajo pasada la citada fecha y hora.

En particular, no se aceptarán trabajos enviados por mail a los docentes del curso, ni entregados en medios físicos en el Instituto.

Reiteramos que el sistema de entregas soporta múltiples entregas por grupo, llevando un histórico de las mismas. **Intente realizar una entrega vacía a los efectos de verificar que su sistema le permite entregar correctamente.**

Apéndice A – Instalación y uso del Tklsem

Para instalar el Tklsem a partir de los RPM's que se pueden encontrar en la página del curso se deben seguir los siguientes pasos:

- a. Verificar que se cuenta con los paquetes de Tcl/Tk instalados. Para ello alcanza con verificar la existencia del binario "wish" o similar en el directorio `/usr/bin`. De no existir, se debe instalar a partir de los CD's de Instalación o bajándolos de Internet.
- b. Instalar el RPM "tkisem-inco.xx.y.rpm" mediante el siguiente comando: `rpm -ivh -nodeps tkisem-inco.xx.y.rpm`. Luego se debe crear un link simbólico desde `/usr/bin/wish` o `/usr/local/bin/wish8.x` a `/usr/local/bin/wish`.
- c. Instalar el RPM "binutils-sparc-intel.xx.y.rpm" mediante un comando similar al anterior, en este caso no hace falta la bandera "nodeps".

Las utilidades quedan instaladas en los directorios `/usr/local/tkisem` y `/usr/local/tkisem/binutils`.

Para compilar un programa en ensamblador SPARC se deben ejecutar dos pasos, primero el ensamblado y luego el linkeditado, con comandos similares a los siguientes:

```
/usr/local/tkisem/binutils/bin/sun4-as -o prg.o prg.s  
/usr/local/tkisem/binutils/bin/sun4-ld -o prg.sparc prg.o
```

El simulador reconoce la existencia de dos variables de ambiente, a saber TKISEM_ROM (archivo a usar como código supervisor) y TKISEM_USER (archivo a usar como código de usuario o de aplicación). Se recomienda utilizar estas variables para agilizar el proceso de desarrollo.

Apéndice B - Makefile

Se sugiere la utilización de un "Makefile" similar al que se indica en la Figura B.1 a los efectos de simplificar la compilación y el linkeditado.

```
# MAKEFILE para ensamblar programas SPARC
# (c) INCO, Facultad de Ingeniería
# Curso "Arquitectura de Sistemas 2"

SPARCUTILS_DIR = /usr/local/tkisem/binutils/bin

###
# Defino dos pattern rules
###
%.o : %.s
    $(SPARCUTILS_DIR)/sun4-as -o $@ $<

%.sparc : %.o
    $(SPARCUTILS_DIR)/sun4-ld -o $@ $<

###
# Aqui ensamblo mis programas
###

all: fib.sparc count.sparc xtable.sparc

clean:
    rm *.sparc;
    rm *.o;

# Ejemplo 1
fib.o : fib.s
fib.sparc : fib.o

# Ejemplo 2
count.o : count.s
count.sparc : count.o

# Ejemplo 3
xtable.o : xtable.s
xtable.sparc : xtable.o

###
# FIN del Makefile
```

Figura B.1

Apéndice C – Ejemplos para comenzar

Como los handlers de traps deben ejecutarse asociados al código de modo supervisor del Tklsem, debemos compilar nuestros handlers junto con el resto de este código supervisor en uno solo.

A estos efectos estará disponible en la página web del curso el archivo “*isem_rom.tar.gz*”, conteniendo dos archivos objeto (*isem_rom.o* e *isem_vec.o*), un makefile y un archivo fuente (*handler.s*).

Este makefile linkedita un archivo llamado “*isem_rom*” que luego debemos cargar en el simulador. El handler a crear para este obligatorio se crea en el archivo *handler.s*.

```
!-----
! Conteo hasta un numero fijo
! SPARC Ejemplo I

.set      valor,0
.set      max,64

.data
.align    4
pesp: .space 128, 0

.text
.align    4
.global   main

main:
    or %r0, valor,%l1
    sethi %hi(pesp), %l2
    or %l2, %lo(pesp), %l2
    or %r0, max, %l3

loop:
    stb    %l1, [%l2]
    addcc  %l1, 1, %l1
    addcc  %l2, 1, %l2
    subxcc %l3, 1, %l3
    bne    loop
    nop
    ta 0

!-----
```

Figura C.1

Al desarrollar en assembler para Tklsem debemos tener en cuenta lo siguiente:

- a. La forma correcta de terminar el programa consiste en invocar la trap 0. En este caso, el Tklsem muestra el mensaje "Program terminated normally" en la consola de I/O y para correr nuevamente el programa debemos volver a cargarlo con el botón "Load".
- b. Para imprimir o leer caracteres individuales de la consola de I/O disponemos de las traps 1 y 2 respectivamente. El trap 1 espera el código ASCII del carácter en el registro %i0.
- c. Para imprimir cadenas terminadas en cero disponemos de la trap 6. Esta trap espera recibir en el registro %i0 la dirección de memoria a partir de la cual se encuentra la cadena.
- d. La función de entrada no-bloqueante esta disponible en la trap 3 (función getc_nb), devuelve en %o0 el carácter disponible. Esta función trabaja a nivel de caracteres individuales.

En la figura C.1 se muestra un "esqueleto" de programa assembler SPARC que puede ser utilizado como punto de partida para el obligatorio.

Apéndice D : El dispositivo gráfico GX

El GX es un dispositivo grafico mapeado en memoria que representa un display de 512 x 512 píxeles (aunque no todos tienen porque estar visibles al mismo tiempo). El píxel de coordenadas (0,0) corresponde a la esquina superior izquierda, y el píxel de coordenadas (511,511) corresponde a la esquina inferior derecha.

El dispositivo GX tiene mapeados en memoria tres ítems:

- a- una palabra de estado
- b- un registro de comandos
- c- 254 registros de parámetros

El ciclo de operación se describe en alto nivel de la siguiente forma:

1. open(gx device)
2. store (argumentos en los registros de parámetros)
3. store (comando en el registro de comandos)
4. ... (repetir 2 y 3 tanto como haga falta) ...
5. close(gx device)

El orden en que se realizan los pasos 2 y 3 es crítico, el comando se ejecuta en el instante en que se hace la instrucción STORE en el registro de comandos.

Los comandos disponibles son:

- a. open
- b. close
- c. color
- d. gx_op
- e. line
- f. fill
- g. blt

(d) define como se modifican los bits (or, xor, otros...)

Para la definición de constantes, ver el archivo "gx.h". Como referencia de los comandos disponibles, así como también para examinar algunos ejemplos de uso, ver el capítulo "*Laboratory 8*" en el documento "**labman.pdf**" adjunto. Se adjuntan también algunos archivos fuente ejemplo.

Apéndice E : El Timer

El timer también es un dispositivo mapeado en memoria. El dispositivo se compone de dos registros mapeados en memoria, **PERIOD** y **COUNT**. Solo PERIOD es accesible por el usuario, COUNT se utiliza internamente por el dispositivo. Hay un bit de "Interrupt" también, el cual indica que se ha producido una interrupción.

El registro PERIOD está mapeado en memoria en la dirección 0x130000, y la interrupción generada es la 0x11.

El funcionamiento es el siguiente:

1. se escribe un entero en PERIOD
2. el dispositivo pone a cero el registro COUNT
3. el dispositivo incrementa en uno a COUNT cada ciclo de reloj
4. cuando COUNT alcanza a PERIOD:
 - se genera una interrupción
 - se pone a cero COUNT
 - se pone a uno el bit "**Interrupt**"

Si se escribe CERO en PERIOD, se deshabilita el Timer. Desde el handler de la interrupción, alcanza con leer del registro PERIOD para resetear la el bit de interrupción (importante para evitar loops de interrupciones).

Referencias

1. Página Web del curso: <http://www.fing.edu.uy/inco/cursos/arqsis2/>
2. Manual de la arquitectura SPARC v8:
<http://www.sparc.com/standards/V8.pdf>.
3. Referencia SPARC de la Universidad Rice:
<http://www.owl.net.rice.edu/~comp320/2004/notes/sparc/>
4. Tklsem Home Page: <http://www.cs.unm.edu/~maccabe/tksem/>
5. Manual del GNU Assembler (GAS): http://www.gnu.org/manual/gas-2.9.1/html_mono/as.html