Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Sistemas Operativos

Proyecto Fase I

Para esta primera fase, el grupo de Sistemas Operativos requiere que usted implemente una arquitectura virtual, que correrá sobre el sistema operativo nativo: LINUX, cuyo objetivo será implementar la capa de hardware y ofrecer servicios que permitan al *minikernel* (que usted desarrollará en la segunda fase) su manejo.

Como ud. sabe, una de las funciones del Sistema Operativo, es crear una abstracción del Hardware, ofreciendo una máquina extendida sobre la que ejecutan las aplicaciones. La idea que se engloba en esta primera fase del proyecto, es invertir este proceso, es decir, colocar una capa encima del Sistema Operativo a la que podemos denominar hardware virtual, que cree una máquina real sobre la extendida.

Esta fase del proyecto será sumamente sencilla, pero a la vez indispensable para que tenga éxito en la implementación del *minikernel*.

La especificación en detalle de la arquitectura es la siguiente:

1. El procesador tiene dos (2) modos de ejecución clásicos: modo privilegiado o núcleo, en el que se ejecuta código del sistema operativo y modo usuario, que corresponde con la ejecución del código de procesos de usuario.

2. La arquitectura no manejará registros de propósito general, en lugar de esto posee un registro acumulador (AC) el cual se emplea para la mayoría de las operaciones.

3. El tamaño de la palabra será de ocho (8) dígitos decimales.

4. Los datos serán representados en signo magnitud, el primer dígito del dato indicará el signo (el signo + se representa como 0, y el - como 1). Lógicamente, los 7 dígitos restantes especifican la magnitud.

5. Los registros de propósito especial son los siguientes:

   MAR = contiene la dirección de memoria de una palabra a buscar en la fase de búsqueda.

   MDR = contiene el dato donde apunta el MAR en la fase de búsqueda

   IR = registro que almacena la instrucción actual

   RB = registro base de la posición de memoria del proceso en ejecución

   RL = registro límite de la posición de memoria del proceso en ejecución

   RX = registro base de la pila

   SP = registro puntero al tope de la pila

   PSW = Palabra de estado del sistema

6. Descripción detallada del registro PSW:

Código de condición	***Modo de operación***	***Habilitar/deshabilitar interrupciones***	***PC***
1 dígito decimal	1 dígito decimal	1 dígito decimal	5 dígitos decimales

Código de condición: Muestra el resultado de la operación aritmética más reciente

Código de Condición = 0, el resultado de la operación =0 X=Y

Código de Condición =1, el resultado de la operación <0 X < Y

Código de Condición = 2, el resultado de la operación >0 X > Y

Código de Condición = 3, el resultado de la operación genero un desbordamiento.

Modo de operación: Como se describió en el punto 1, el procesador tiene dos modos de ejecución:

Modo usuario = 0
Modo kernel = 1

Habilitar/deshabilitar Interrupciones: Indica si las interrupciones están habilitadas o no.

Deshabilitadas = 0
Habilitadas = 1

PC: Contiene la dirección de la próxima instrucción a leer.

7. La memoria RAM deberá emularse como un arreglo de 2000 posiciones, donde cada posición alberga una palabra de la arquitectura. Las primeras 300 posiciones estarán reservadas para el Sistema Operativo, el resto debe estar disponible para el usuario.

8. El Formato de instrucción es el siguiente:

Código de operación	***Direccionamiento***	***Valor***
2 dígitos decimales	1 dígito decimal	5 dígitos decimales

Código de operación: (ver conjunto de instrucciones)

Direccionamiento: En la instrucción, hay siempre un campo que indica el tipo de direccionamiento:

0 = directo (los 5 últimos dígitos referencian a memoria)
1 = inmediato (los 5 últimos dígitos son el dato)
2 = indexado (los 5 últimos dígitos son un índice a partir del acumulador de una posición de memoria).

9. Conjunto de instrucciones:

Código de operación	*Instrucción*	*Descripción*
00	sum	Suma dato a AC
01	res	Resta dato a AC
02	mult	Multiplica dato a AC
03	divi	Divide dato a AC
04	load	Carga el contenido de memoria en AC
05	str	Almacena el AC en memoria
06	loadrx	Carga el RX en AC
07	strrx	Almacena AC en RX
08	comp	Compara dato con AC
09	jmpe	Salta si AC=M[SP] a la dirección especificada Ejemplo: 09000469 Si (AC = M[SP]) ent PC= M[469] fsi
10	jmpne	Salta si AC!=M[SP] a La dirección especificada
11	jmplt	Salta si AC<M[SP] a la dirección especificada
12	jmplgt	Salta si AC>M[SP] a La dirección especificada
13	svc	Llamada al sistema, el código se pasa en AC y parámetros en la pila
14	retrn	Retorno de una subrutina
15	hab	Habilita interrupciones
16	dhab	Deshabilita interrupciones
17	tti	Establece tiempo en el reloj Ejemplo: 17000012 El reloj interrumpe cada 12 ciclos
18	chmod	Cambia de modo
19	loadrb	Carga el RB en AC
20	strrb	Almacena AC en RB
21	loadrl	Carga el RL en AC
22	strrl	Almacena AC en RL
23	loadsp	Carga el SP en AC
24	strsp	Almacena AC en SP
25	psh	Apila
26	pop	Desafila
27	j	salto incondicional
28	sdmap	Establece la pista a acceder
29	sdmac	Establece el cilindro a Acceder
30	sdmas	Establece el sector a acceder
31	sdmaio	Establece si es I/O
32	sdmam	Establece la posición de memoria a ser accedida
33	sdmaon	Inicia DMA

10. La máquina debe contar con 1 procesador, 1 disco magnético de almacenamiento masivo, 1 DMA y un bus compartido entre el procesador y el DMA para ir a memoria, es decir, debe haber algún tipo de arbitraje de modo que cuando el procesador esté accediendo a memoria no haya inconsistencia cuando el DMA lo esté haciendo, igual para el caso que el DMA intente acceder a memoria.

11. Los registros base y límite, RB y RL respectivamente, se usarán para cumplir con el requerimiento de protección de memoria. Recuerde que toda dirección de memoria es una dirección relativa al inicio del proceso. Para verificar que cada acceso a memoria sea válido debe sumarse la dirección relativa con RB, si ese resultado es mayor que RL o menor que RB entonces se generará una interrupción por direccionamiento invalido. En modo privilegiado se omite esta comprobación, por tanto un código ejecutando en este modo puede direccionar toda la memoria.

12. Manejo deE/S:

- Las operaciones de *sdma(x)* son las indicadas para una operación de entrada salida, el procesador, con cada una de estas instrucciones va indicando los parámetros al DMA, y eventualmente con la instrucción *sdmaon* el DMA comienza a trabajar

- En particular, la instrucción *sdmam* le dice al DMA la dirección de memoria de donde leer los datos ó escribir los datos.

  La decisión de escribir o leer depende de la instrucción *sdmaio*:

  0=leer
  1=escribir

- A continuación, el DMA comienza su trabajo comunicándose con el disco. Esta comunicación no requiere el uso del bus del sistema, pero si la comunicación del DMA con memoria.

- La comunicación del DMA con el disco se deja al diseñador.

- El disco tiene que ser un arreglo de al menos 3 dimensiones con las que se pueda simular la pista, cilindro y sector. Se puede asumir que cada sector tiene un dato de 9 caracteres. El disco debe ser de al menos 10 pistas, 10 cilindros y 100 sectores por cilindro.

- El disco envía datos desde un sector hacia el DMA o viceversa.

- Cuando la operación de entrada salida ha finalizado, DMA establece en el registro ESTADOdma del DMA el resultado de la operación:

  0= éxito
  1= error

- Luego, interrumpe al procesador.

13. Manejo de interrupciones:

- El manejo de interrupciones en este sistema se realizara mediante un vector de interrupciones. Este vector será indexado por el código de interrupción, justo antes de ir al manejador de interrupciones, es decir, indexar el vector con el código de interrupción, la maquina debe salvaguardar todos los registros y el código de tratamiento de la interrupción

debe encargarse, si ese es el caso, de restaurar los registros para seguir con la rutina interrumpida.

- La arquitectura debe lanzar una excepción en varios casos, la lista de códigos de interrupción es la siguiente:

Código	***Interrupción***
8	*Overflow*
7	*Underflow*
6	*Direccionamiento invalido*
5	*Instrucción invalida*
4	*Finalización de operación de E/S*
3	*Reloj*
2	*Llamada al sistema*
1	*Código de interrupción invalido*
0	*Código de llamada al sistema invalido*

15. Los archivos de entrada se cargarán en memoria en la posición que indique el usuario, y tienen el siguiente formato:

```
_start <linea donde comienza la ejeucion>
.NumeroPalabras <numero de palabras que contiene el archivo de entrada>
.NombreProg <nombre del programa>
//palabras
```

Ejemplo detallado:

Entrada

```
_start 1
.NumeroPalabras 7
.NombreProg 5mas5
04100005            //load 5
00100005            //sum 5
25000000            //psh
04100007            //load VALOR
13000000            //svc
04100007            //load VALOR
13000000            //svc
```

NOTA: VALOR representa cualquier número que no es necesario que se especifique en esta fase.

14. Inicio del sistema y modos de trabajo:

Cuando inicia el sistema, debe levantar una consola con una línea de órdenes clara y sencilla en la que se le pueda ordenar al sistema ejecutar un archivo en uno de los dos modos de trabajo: modo normal y modo *debugger*.

Dependiendo del modo de trabajo que se haya especificado en la línea de órdenes al ejecutar un programa, el sistema va a comportarse distinto:

- En caso que sea en modo normal, el sistema ejecuta completamente el programa y retorna el control a la consola.

- En caso que sea en modo *debugger*, va a ejecutar cada instrucción del programa y va a retornar el control a la consola y a arrojar información de lo que acaba de ejecutar, información como:

  - Dirección de memoria de la instrucción
  - La instrucción como tal
  - El resultado de esa instrucción

Además debe preguntar al usuario si desea seguir con la depuración del programa o desea ver el valor de algún registro en especial.

Requisitos Obligatorios:

1. El sistema debe ejecutar en el sistema operativo Linux y debe ser escrito en C.
2. El sistema debe tener un log, es decir, toda acción que haga en el sistema debe quedar registrado en un archivo 'log' el cual se usará para evaluar la correctitud de su arquitectura.
3. Cuando ocurra una interrupción debe imprimirse un mensaje tanto en el log, como en la salida estándar, donde se indique el código y la descripción de la interrupción.
4. Usted debería implantar el ciclo de instrucción al igual que como se hace en la realidad, así como la fase de búsqueda utilizando los registros MAR, MDR, IR y PC para ello, y la fase de ejecución.
5. Deben controlarse las distintas condiciones de competencia que presenta la implementación de la arquitectura.
6. Para el manejo de concurrencia utilice hilos o procesos o ambos.
7. Usted debe implementar un sistema de arbitraje que provea el acceso correcto al bus del sistema.
8. Es indispensable para que su proyecto tenga validez, que respete la arquitectura en el sentido de códigos de operación formato de los campos y orden de los mismos.
9. El código fuente debe ser extensamente comentado y estructurado apropiadamente. Un código correctamente comentado y presentado es mucho más fácil de interpretar y de corregir.
10.La entrega del código fuente solamente puede contener archivos de extensión .c , .h y un archivo makefile. No se debe incluir ningún programa ejecutable. El makefile reconstruirá automáticamente su programa a partir del código fuente proporcionado. Si el código enviado no compila, no se corrige. Es obligatorio el uso de cabeceras .h.
11.Un documento PDF con una explicación detallada de su solución que deberá incluir: Una planificación de tareas para la realización del proyecto en su totalidad. Esta planificación deberá realizarla con ayuda de sus compañeros de equipo. Una explicación exhaustiva de su código y lógica para llegar a la solución.
12.La entrega debe estar comprimida en un archivo (tar o zip) y que incluirá todo el código fuente y el makefile. Si el código es enviado sin el Makefile o el Makefile no funciona, el código no será evaluado.
13.Anexar los archivos de sus casos de prueba con resultados reales.

Fecha de entrega Sabado 17/01/26 antes de las 11:59 PM. El proyecto debe ser enviado al correo laboratoriosisop@gmail.com con el asunto [SO]_[Proy1]_[CI1]_[CI2]_[CI3].

Proyecto enviado que no cumpla con el formato o se entregue después de la fecha y hora establecida no será corregido.

La evaluación puede realizarse como máximo en grupos de 2 o 3 estudiantes, no se acepta trabajos individuales, los grupos pueden estar conformados por estudiantes de diferentes secciones de laboratorio, copia y/o plagio (IA) serán severamente penalizados con puntuación 0.

La distribución de puntos queda de la siguiente manera:

- ✓ **Informe - 4 pts**
- ✓ **Defensa individual - 2 pts**
- ✓ **Código/Documentación/Comentarios - 14 pts**