

2020

Sistemas Operativos

Trabajo Práctico N°2: eXpOS

Grupo: 13

Integrantes: Nuria Delaude,
Manuel Onetto, Lucas
Maidana, Martin Rotelli,
Franco Molina

Índice

1. PALABRA.	2
a. Definición de palabra.	2
b. Cantidad de palabras que ocupa cada instrucción en la arquitectura de la maquina XSM.	2
d. Cargue y ejecute el código XSM.	3
i. Salida. Modificar la primera línea asignando un 1 (uno) al registro R0.	4
ii. Instrucción de la línea 2.	4
2. MODELO DE ESTADOS EXPOS.	5
a. Similitudes y diferencia con los modelos vistos en clase	5
b. Diagrama de estados eXpOS.	7
i. Transiciones.	7
3. DEPURACIÓN DEL CÓDIGO.	9
a. Encontrando los errores.	9
b. Solución propuesta para imprimir los números pares.	10
4. MECANISMO UTILIZADO PARA CONCRETAR OPERACIONES I/O.	12
b) INTERRUPCIONES.	12
BIBLIOGRAFÍA	15

1. Palabra.

a. Definición de palabra.

En informática, una **palabra** es la unidad natural de datos utilizada por un diseño de procesador en particular. Una palabra es un dato de tamaño fijo manejado como una unidad por el conjunto de instrucciones o el hardware del procesador. El número de bits en una palabra (el tamaño de la palabra, el ancho de la palabra o la longitud de la palabra) es una característica importante de cualquier diseño de procesador o arquitectura de computadora específicos.

El sistema de archivos lógico eXpFS consta de archivos organizados en un solo directorio llamado la root. Root también se trata conceptualmente como un archivo. Cada archivo eXpFS es una secuencia de palabras. Asociados con cada archivo eXpFS hay tres atributos: nombre, tamaño, tipo.

Cada atributo tiene una palabra de longitud. El nombre del archivo debe ser una cadena.

Cada archivo debe tener un nombre único. El tamaño del archivo será el número total de palabras almacenadas en el archivo.

Las estructuras de datos que contienen palabras de diferentes tamaños se refieren a ellas como WORD (16 bits / 2 bytes), DWORD (32 bits / 4 bytes) y QWORD (64 bits / 8 bytes) respectivamente.

b. Cantidad de palabras que ocupa cada instrucción en la arquitectura de la maquina XSM.

El XSM o eXperimental String Machine Simulator se utiliza para simular el hardware XSM. Es una máquina monoprocesador impulsada por interrupciones. La máquina maneja los datos como cadenas. Una cadena es una secuencia de caracteres terminados por \0.

La longitud de una cadena es de 16 caracteres como máximo, incluido \0. Cada una de estas cadenas se almacenan, en una palabra. La máquina interpreta un solo carácter también como una cadena.

Cada instrucción en XSM tiene dos palabras de largo:

Instrucciones de transferencia de datos

Familia de instrucciones relacionadas con el movimiento de datos entre un registro y un registro / ubicación de memoria / entero o constante de cadena. La instrucción MOV admite la transferencia de datos a través de varios modos de direccionamiento.

Instrucciones aritméticas

Las instrucciones aritméticas realizan operaciones aritméticas en registros que contienen números enteros. Si el registro contiene un valor no entero, se genera una excepción. Las instrucciones aritméticas son ADD, SUB, MUL, DIV, MOD, INR y DCR.

Instrucciones Lógicas

Las instrucciones lógicas se utilizan para comparar valores en registros. Las cadenas también se pueden comparar de acuerdo con el orden lexicográfico de ASCII. Las instrucciones lógicas son GT, LT, EQ, NE, GE y LE.

Instrucciones de bifurcación

La bifurcación se logra cambiando el valor de la IP a la dirección de palabra de la instrucción de destino especificada por target_address.

Las instrucciones de ramificación

Las instrucciones de ramificación son JZ, JNZ, JMP

Instrucciones de pila

Las instrucciones de la pila son PUSH y POP.

Instrucciones de subrutina

Las instrucciones de subrutina proporcionan un mecanismo para invocaciones de procedimientos. Las instrucciones de la subrutina son CALL y RET.

Instrucciones de depuración

La máquina, cuando se ejecuta en modo de depuración, invoca al depurador cuando se ejecuta esta instrucción. Esta instrucción se puede utilizar para depurar el código del sistema. La instrucción para ingresar al modo de depuración es BRKP.

Interrupción de software

Genera una interrupción en el kernel con n (4 a 18) como parámetro. La instrucción para activar la interrupción es INT n, donde n es el número de la rutina de interrupción.

d. Cargue y ejecute el código XSM.

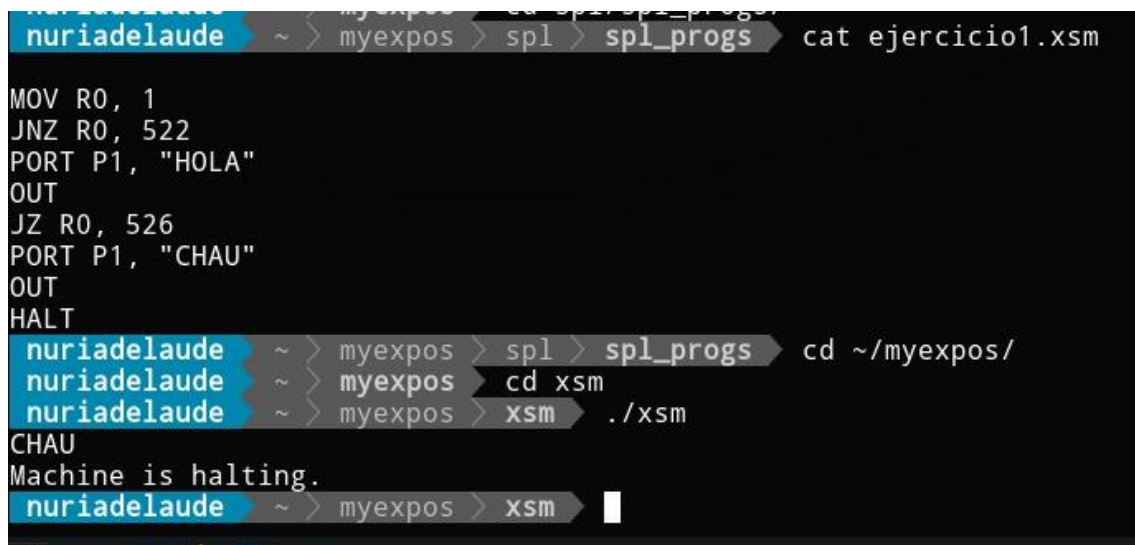
```
nuriadelaide ~ > myexpos > spl > spl_progs > cat ejercicio1.xsm
MOV R0, 0
JNZ R0, 522
PORT P1, "HOLA"
OUT
JZ R0, 526
PORT P1, "CHAU"
OUT
HALT
nuriadelaide ~ > myexpos > spl > spl_progs > cd ..
nuriadelaide ~ > myexpos > spl > cd ..
nuriadelaide ~ > myexpos > cd xfs-interface/
nuriadelaide ~ > myexpos > xfs-interface > ./xfs-interface
Unix-XFS Interface Version 2.0.
Type "help" for getting a list of commands.
# load --os $HOME/myexpos/spl/spl_progs/ejercicio1.xsm
# exit
nuriadelaide ~ > myexpos > xfs-interface > cd ..
nuriadelaide ~ > myexpos > cd xsm
nuriadelaide ~ > myexpos > xsm > ./xsm
HOLA
Machine is halting.
nuriadelaide ~ > myexpos > xsm > █
```

Para completar este punto, cargamos el código en la carpeta “spl_progs” en un file. xsm que luego cargamos en la xfs-interface utilizando el comando:

load --os \$HOME/myexpos/spl_progs/ejercicio1.xsm

Luego, desde ./xsm ejecutamos el código y obtuvimos la respuesta que se puede observar en imagen anterior.

- i. Salida. Modificar la primera línea asignando un 1 (uno) al registro R0.



```
nuriadelaude ~ > myexpos > spl > spl_progs > cat ejercicio1.xsm
MOV R0, 1
JNZ R0, 522
PORT P1, "HOLA"
OUT
JZ R0, 526
PORT P1, "CHAU"
OUT
HALT
nuriadelaude ~ > myexpos > spl > spl_progs > cd ~/myexpos/
nuriadelaude ~ > myexpos > cd xsm
nuriadelaude ~ > myexpos > xsm > ./xsm
CHAU
Machine is halting.
nuriadelaude ~ > myexpos > xsm > 
```

En esta imagen se puede observar que cambiamos el 0 por un 1 cuando el código asigna un valor al registro R0.

Lo que sucedió fue que recibimos un “CHAU” ¿por qué? Reconoció que al asignarle un 1, no cumplía con la condición anterior.

- ii. Instrucción de la línea 2.

JNZ: es una instrucción de salto, ‘JUMP SHORT IF NOT ZERO’ (ZF=0, flag)

R0: registro procesador

522: valor de IP, ‘instruction pointer’ (contador del programa).

Explicación en profundidad: Corrobora el estado de uno o más de los indicadores de estado en el registro EFLAGS (CF, OF, PF, SF y ZF) y, si los indicadores están en el estado especificado (condición), realiza un salto a la instrucción de destino especificada por el operando de destino. Se asocia un código de condición (cc) con cada instrucción para indicar la condición que se está probando. Si no se cumple la condición, no se realiza el salto y la ejecución continúa con la instrucción que sigue a la instrucción Jcc.

La instrucción de destino se especifica con un desplazamiento relativo (un desplazamiento con signo relativo al valor actual del puntero de instrucción en el registro EIP). Un desplazamiento relativo (rel8, rel16 o rel32) generalmente se especifica como

una etiqueta en el código ensamblador, pero a nivel de código de máquina, se codifica como un valor inmediato firmado, de 8 bits o de 32 bits, que se agrega al puntero de instrucción. La codificación de instrucciones es más eficaz para compensaciones de -128 a +127. Si el atributo de tamaño de operando es 16, los dos bytes superiores del registro EIP se borran, lo que da como resultado un tamaño máximo de puntero de instrucción de 16 bits.

Debido a que un estado particular de los indicadores de estado a veces se puede interpretar de dos maneras, se definen dos mnemónicos para algunos códigos de operación. Por ejemplo, la instrucción JA (saltar si es superior) y la instrucción JNBE (saltar si no es inferior o igual) son mnemónicos alternativos para el código de operación 77H.

La instrucción JCC no admite saltos lejanos (saltos a otros segmentos de código). Cuando el objetivo para el salto condicional está en un segmento diferente, utilice la condición opuesta a la condición que se está probando para la instrucción Jcc y luego acceda al objetivo con un salto lejano incondicional (instrucción JMP) al otro segmento. Por ejemplo, el siguiente salto lejano condicional es ilegal: JZ FARLABEL; Para lograr este salto lejano, use las siguientes dos instrucciones: JNZ BEYOND; JMP FARLABEL; BEYOND: Las instrucciones JECXZ y JCXZ difieren de las otras instrucciones Jcc porque no verifican los indicadores de estado. En su lugar, comprueban el contenido de los registros ECX y CX, respectivamente, para 0. El registro CX o ECX se elige de acuerdo con el atributo de tamaño de dirección.

Estas instrucciones son útiles al comienzo de un ciclo condicional que termina con una instrucción de ciclo condicional (como LOOPNE). Evitan ingresar al bucle cuando el registro ECX o CX es igual a 0, lo que provocaría que el bucle se ejecute 232 o 64K veces, respectivamente, en lugar de cero.

Todos los saltos condicionales se convierten en recuperaciones de código de una o dos líneas de caché, independientemente de la dirección de salto o la capacidad de caché.

2. Modelo de estados eXpOS.

a. Similitudes y diferencia con los modelos vistos en clase

A la hora de controlar la ejecución de procesos, eXpOs tiene un modelo de estado que asimila al modelo de 7 estados visto en clase. Comparten los estados de bloqueo(wait) y los swapping (swapped=listo suspendido y swappedwait=bloqueado suspendido).

El swapping es una operación de E/S que implica mover una parte o todo el proceso de memoria principal a disco. Cuando ninguno de los procesos que se encuentra en memoria principal, se encuentra en estado Listo, el sistema operativo mueve uno de los procesos en estado Bloqueado a un nuevo estado que será el de Suspendido (disco). De esta manera se obtiene espacio para admitir un nuevo proceso en el sistema.

Además de que también comparten otras transiciones, como por ejemplo algunas de las más destacadas del modelo de 7 estados, que son:

Listo/Suspendido → Listo: cuando no hay procesos en estado listo el sistema tomará un proceso del estado Listo/Suspendido. Otro motivo puede ser que un proceso Listo/Suspendido tenga más prioridad que un proceso Listo. (punto F en las transiciones).

Nuevo → Listo/Suspendido: puede ocurrir que no haya espacio suficiente en memoria para el nuevo proceso y por eso el sistema operativo decide suspenderlo. (punto C transiciones RUNNING! WAIT MEM).

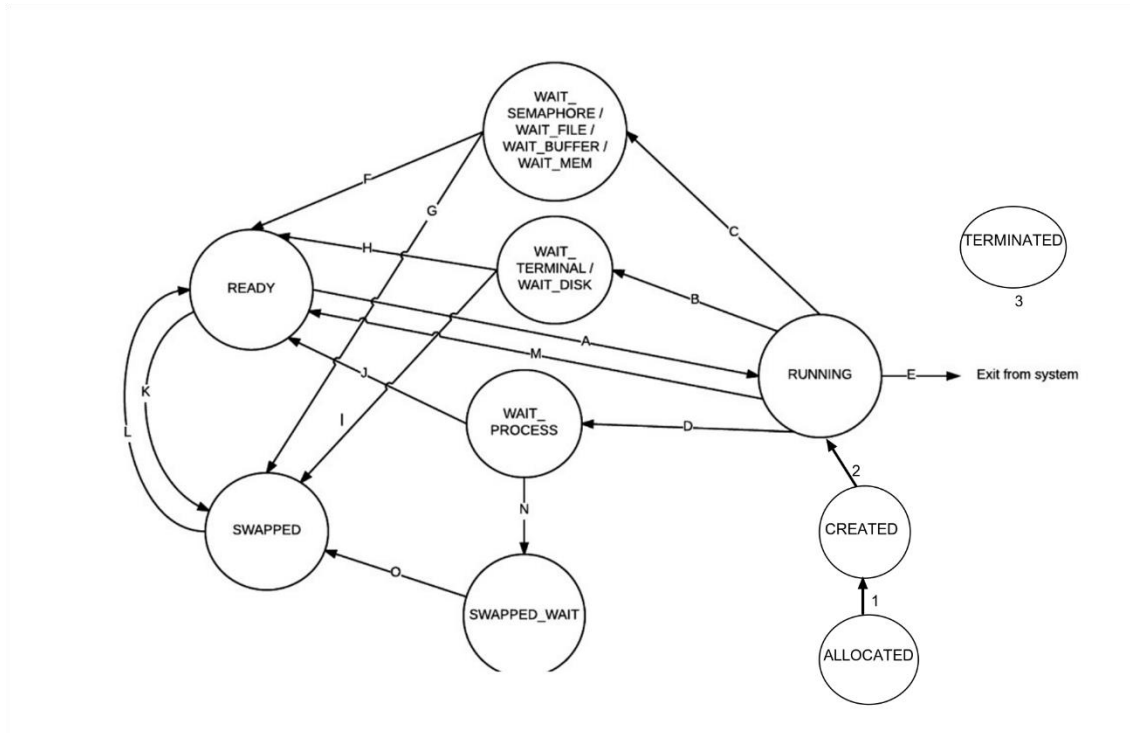
Tanto en eXpOS como en el modelo de 7 estados el procesador posee dos modos de ejecución, el modo núcleo asociado con el sistema operativo y el modo usuario asociado con los procesos del usuario.

El modo usuario en eXpOS funciona de manera tal que XSM proporciona un conjunto de instrucciones sin privilegios, que son las únicas instrucciones disponibles cuando la máquina se ejecuta en el modo de usuario. La máquina también puede ejecutar instrucciones sin privilegios en modo kernel, sin embargo, la semántica de las instrucciones será diferente. Las instrucciones sin privilegios son instrucciones de transferencia de datos, instrucciones aritméticas, instrucciones lógicas, instrucciones de pila, instrucciones de subrutina, instrucciones de depuración e interrupciones de software. Los registros disponibles en modo usuario son R0-R19, SP, BP e IP.

Para realizar el cambio de modo hay que generar una interrupción en el kernel con $n(4 \text{ a } 18)$ como parámetro. Esto también deshabilita las interrupciones. Primero se incrementa el Stack Pointer, se calcula la dirección física de SP y se almacena la dirección (virtual) de la siguiente instrucción después de que el valor actual de IP se almacena en esa ubicación. Después de esto, el modo de ejecución se cambia al modo Kernel. Tenga en cuenta que INT solo se puede invocar desde el modo de usuario. El valor de IP debe contener una dirección virtual y, por lo tanto, el valor introducido en la pila es una dirección virtual y no una dirección física. Finalmente, el valor de IP se establece de acuerdo con el valor de n como se especifica aquí.

Las instrucciones privilegiadas se pueden ejecutar solo en el modo kernel (tanto las instrucciones privilegiadas como las no privilegiadas se pueden ejecutar en el modo kernel). Cabe señalar que todas las direcciones en el modo kernel deben ser direcciones físicas, mientras que el modo de usuario utiliza direcciones lógicas, y la traducción de direcciones se realiza mediante el hardware Paging.

b. Diagrama de estados eXpOS.



i. Transiciones.

- 1) Asignado-Nuevo: Un proceso es creado por la llamada al sistema Fork y se le asigna una entrada del BCP, su estado actual es "asignado". Cambia a estado "creado" o "nuevo" cuando el Fork completa la creación del mismo.
 - 2) Nuevo-Ejecutando: Se programa el proceso para su primera ejecución.
 - 3) Un proceso puede terminar repentinamente por diversas razones como excepciones, apagado del equipo, cierre de sesión, etc. En este caso el proceso queda en estado TERMINADO.
- A) Listo-Ejecutando: El SO planifica al proceso para su ejecución.
- B) Ejecutando-Esperando terminal: El proceso aguarda por el acceso al terminal o algún input a través del mismo.
- Ejecutando-Esperando disco: Se aguarda acceso al disco o a que termine alguna operación con el mismo.
- C) Ejecutando-Esperando Semáforo: El semáforo por el cual el proceso intenta pasar se encuentra bloqueado por lo que lo mantiene en espera.
- Ejecutando-Esperando Archivo: Al igual que el semáforo, el archivo al cual quiere acceder se encuentra bloqueado y mantiene en espera al proceso.

Ejecutando-Esperando Buffer: El buffer se encuentra bloqueado, por lo que mantiene en espera al proceso.

Ejecutando-Esperando Memoria: No se encuentra ninguna página de memoria libre para el proceso, se mantiene en espera.

D) Ejecutando-Proceso: Se aguarda otro proceso para completar una llamada o simplemente para terminar el proceso.

E) Ejecutando-Saliente: Dentro del proceso se invoca a la salida de sistema o se finaliza su ejecución.

F) Esperando Semáforo-Listo: Se desbloquea el semáforo por el cual el proceso estaba en espera.

Esperando Archivo-Listo: Se desbloquea el acceso al archivo por el cual el proceso estaba en modo de espera.

Esperando Buffer-Listo: El buffer es desbloqueado permitiendo al proceso salir de modo de espera.

Esperando Memoria-Listo: Se desbloquea el proceso ya que se liberan páginas de memoria.

G) Esperando Semáforo/Archivo/Buffer/Memoria - Intercambio: Se realiza el swapping (intercambio entre memoria y disco) en el proceso posponiendo el mismo.

H) Esperando Terminal - Listo: Los datos ingresados por terminal son leídos correctamente y la misma se libera para algún otro proceso.

Esperando Disco - Listo: Se termina la operación en la cual estaba involucrado el disco.

I) Esperando Terminal/Disco - Intercambio: Se realiza el swap (intercambio entre memoria y disco) en el proceso posponiendo el mismo.

J) Esperando Proceso - Listo: El proceso por el cual se esperaba sale del sistema o el que aguardaba por él recibe una señal.

K) Listo - Intercambio: Se realiza el swap (intercambio entre memoria y disco) en el proceso, posponiendo el mismo.

L) Intercambio - Listo: Al contrario, aquí se reanuda el proceso mediante el swap. Pasa de disco a memoria.

M) Ejecutando - Listo: Debido a una interrupción de temporizador se cambia el contexto, causando que vuelva un proceso a estado Listo.

N) Esperando Proceso- Intercambio Bloqueado: Se realiza el swapping pero en un estado "bloqueado" mientras espera otros procesos.

O) Intercambio Bloqueado - Intercambio: El proceso bloqueado recibe una señal del que estaba esperando o éste termina.

3. Depuración del código.

a. Encontrando los errores.

Colocamos breakpoints en el código para poder realizar el debug, lanzamos el comando reg para ver el estado y obtenemos la siguiente imagen con R0=0.

```
Next instruction at IP = 540, Page No. = 1: MOV R16,R0
debug> reg
R0: 0   R1:      R2:      R3:      R4:
R5:      R6:      R7:      R8:      R9:
R10:     R11:     R12:     R13:     R14:
R15:     R16: 0   R17: 0   R18:      R19:
P0:      P1: 0   P2:      P3:
BP:      SP:      IP: 540 PTBR:     PTLR:
EIP:     EC:      EPN:     EMA:
debug> █
```

Avanzamos en el debug con el comando s hasta que imprima y nos encontramos con R0=1.

```
debug> s
Previous instruction at IP = 532: MOV R16,R0
Mode: KERNEL      PID: -1
Next instruction at IP = 534, Page No. = 1: PORT P1,R16
debug> s
Previous instruction at IP = 534: PORT P1,R16
Mode: KERNEL      PID: -1
Next instruction at IP = 536, Page No. = 1: OUT
debug> reg
R0: 1   R1:      R2:      R3:      R4:
R5:      R6:      R7:      R8:      R9:
R10:     R11:     R12:     R13:     R14:
R15:     R16: 1   R17: 0   R18:      R19:
P0:      P1: 1   P2:      P3:
BP:      SP:      IP: 536 PTBR:     PTLR:
EIP:     EC:      EPN:     EMA:
debug> █
```

Luego seguimos avanzando con varias s pero no logramos aumentar más de 2 a R0.

```
Previous instruction at IP = 554: MOV R0,R16
Mode: KERNEL      PID: -1
Next instruction at IP = 556, Page No. = 1: BRKP
debug> reg
R0: 2   R1:      R2:      R3:      R4:
R5:      R6:      R7:      R8:      R9:
R10:     R11:     R12:     R13:     R14:
R15:     R16: 2   R17: 0   R18:      R19:
P0:      P1: 1   P2:      P3:
BP:      SP:      IP: 556 PTBR:     PTLR:
EIP:     EC:      EPN:     EMA:
debug> s
```

Queda atrapado en un loop infinito con R0=2.

b. Solución propuesta para imprimir los números pares.

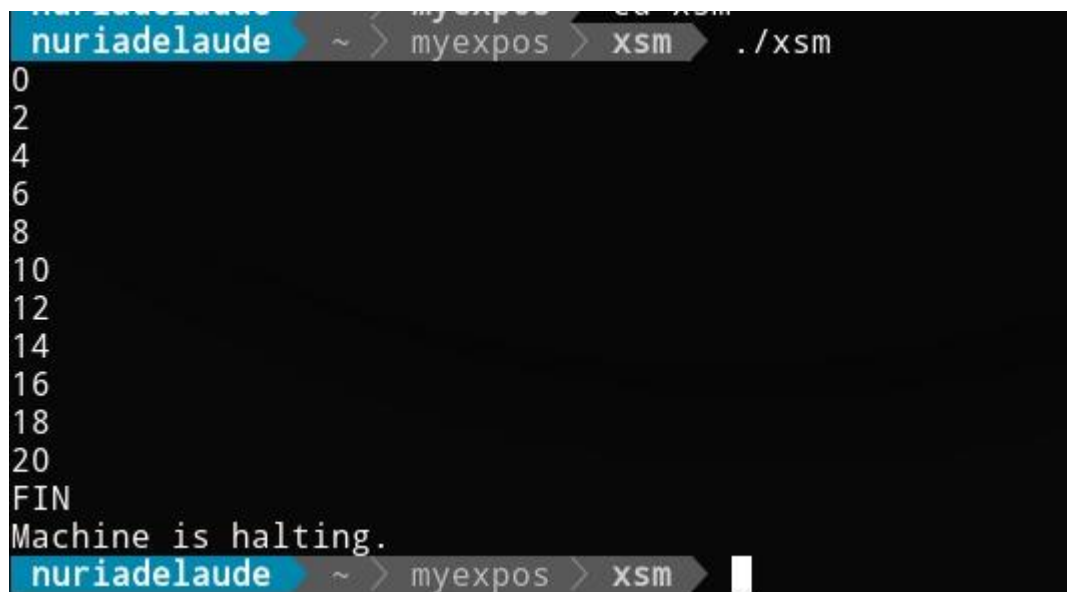
En la siguiente imagen podemos observar la solución propuesta. Decidimos cambiar:

if (i / 2 == 0) then por if (i % 2 == 0)

El primer if lo que nos devuelve el cociente de la división entre dos números enteros, mientras que el segundo if nos devuelve el resto entre 2 números. Si el resto de hacer un número dividido 2 nos da 0, quiere decir que este es un número par.

El primer if solo nos devuelve como resultado 0 y 1. Luego al hacer $i = i + 1$ a partir de $i=2$ entramos en un loop infinito donde i siempre va a valer 2. Por lo que decidimos cambiarlo a $i = i + 1$. De esta forma, cuando no cumpliera con la condición podemos aumentar la variable en 1 y que la condición pruebe con el siguiente número, siguiendo con el ejemplo de $i=2$, el próximo número sería $i=3$ y así sucesivamente. Entonces tanto si cumple o no con la condición, se aumenta i de tal forma que puede continuar la ejecución y no entre en loop.

El segundo if con las correcciones antes mencionadas, nos devuelve todos los números pares menores a 21.



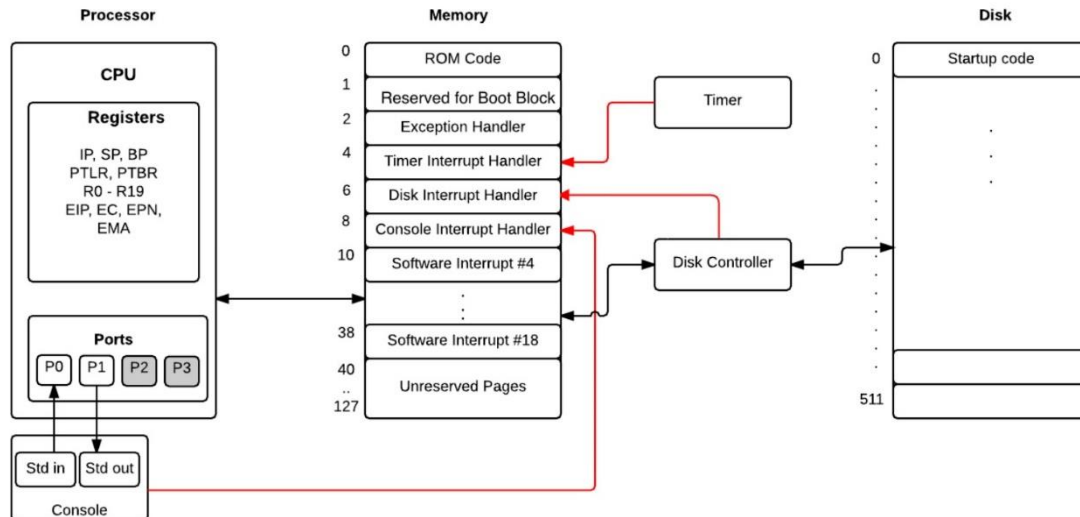
```
nuriadelaude ~ > myexpos > xsm > ./xsm
0
2
4
6
8
10
12
14
16
18
20
FIN
Machine is halting.
nuriadelaude ~ > myexpos > xsm >
```

```
nuriadelaide ~ > myexpos > spl > spl_progs cat punto3corregido.spl
alias i R0;
i = 0;
while (i < 21) do
if (i % 2 == 0) then
print i;
i = i + 1;
endif;
i = i + 1;
endwhile;
print "FIN";
nuriadelaide ~ > myexpos > spl > spl_progs
```

```
nuriadelaide ~ > myexpos > spl > spl_progs cat punto3corregido.xsm
MOV R0, 0
_L1:
MOV R16, 21
GT R16, R0
JZ R16, _L2
MOV R16, R0
MOD R16, 2
MOV R17, 0
EQ R16, R17
JZ R16, _L3
MOV R16, R0
PORT P1, R16
OUT
MOV R16, R0
ADD R16, 1
MOV R0, R16
JMP _L4
_L3:
_L4:
MOV R16, R0
ADD R16, 1
MOV R0, R16
JMP _L1
_L2:
MOV R16, "FIN"
PORT P1, R16
OUT
HALT
nuriadelaide ~ > myexpos > spl > spl_progs
```

4. Mecanismo utilizado para concretar operaciones I/O.

b) INTERRUPCIONES.



Cuando la máquina se enciende, el Bootstrap Loader carga el primer bloque del disco en un área predefinida de la memoria y transfiere el control al nuevo código cargado llamado código de inicio del sistema operativo. El código de inicio del SO carga el sistema con rutinas en la memoria. Además de estas, el timer de interrupciones del temporizador, el manejador de excepciones, el manejador de interrupción de disco y el manejador de terminal son también cargadas.

- Manejador de interrupciones con temporizador

La especificación de los requisitos de hardware para el eXpOS asume que la máquina está equipada con un dispositivo temporizador que envía periódicamente interrupciones de hardware.

El sistema operativo es invocado por el hardware manejador de interrupciones del temporizador.

- Handler de tiempo de interrupción

Sugieren que se emplea una programación cooperativa de tareas múltiples. Esto significa que se emplea un algoritmo de planificación "Round Robin", pero un proceso puede suspenderse dentro de una llamada de sistema cuando:

- El recurso al que el proceso está tratando de acceder (como un _le o un semáforo) está bloqueado por otro proceso (o incluso internamente bloqueado por otro sistema operativo en ejecución simultánea).
- Hay un disco o un dispositivo de E/S de acceso en una llamada al sistema que es lento. Si el 17 espera el acceso al dispositivo y es evitado, debe haber soporte de hardware del dispositivo para enviar una interrupción de hardware cuando el funcionamiento del dispositivo es terminado. Esto permite que el sistema operativo ponga el proceso en suspenso por ahora, para continuar programando

los procesos restantes en forma de "Round Robin" y luego despertar en el proceso en suspenso cuando el dispositivo envía la interrupción.

- Handler de excepciones

Si un proceso genera una instrucción ilegal, una dirección inválida (fuera de su espacio de direcciones virtuales) o hace una división por cero (u otras condiciones defectuosas que son dependientes de la máquina), la máquina generará una excepción.

Cuando el manejador encuentra una excepción debe terminar el proceso, despertar a todos los procesos que lo esperan (o recursos bloqueados por él) e invoca al programador para continuar la programación del round robin de los procesos restantes. El manejador de excepciones también se invoca cuando un fallo de página ocurre. El módulo que maneja la demanda de paginación (si el hardware de la máquina soporta demanda de paginación) es invocado por el manejador de excepciones cuando una página falla.

- Controlador del disco

eXpOS trata al disco como un dispositivo de bloqueo especial y asume que el disco proporciona rutinas de transferencia de bloques de bajo nivel para transferir bloques de disco a la memoria

(páginas) y volver. Las rutinas de transferencia de bloque contienen instrucciones para iniciar el bloque de transferencia de memoria por el hardware del controlador del disco. Después de iniciar la transferencia de memoria del disco, la rutina de transferencia de bloque normalmente vuelve al programa de llamada, que suspende para que la operación del disco se complete. Cuando la transferencia de memoria del disco está completa, el controlador del disco levanta una interrupción de hardware. El controlador de la interrupción del disco es responsable de los procesos de vigilia que se suspendieron esperando la finalización de la operación de disco.

- Handler de terminales y otros dispositivos

Todos los demás dispositivos de manejo de datos (excepto el disco) se tratan como dispositivos de corriente. Se supone que para cada dispositivo hay rutinas de bajo nivel asociadas que pueden ser invocadas por el sistema operativo para transferir datos e instrucciones de control. Algunos de estos dispositivos pueden provocar una interrupción de hardware cuando la transferencia se completa. Por lo tanto, para cada dispositivo que provoca una interrupción, debe haber un manejador de interrupciones correspondiente.

La entrada estándar y la salida estándar son dos dispositivos especiales de flujo con identificadores predefinidos STDIN = -1 y STDOUT = -2. La salida estándar sólo permite la escritura y la entrada estándar sólo permite la lectura. La operación de lectura típicamente pone el proceso que ejecuta la operación en suspenso para la entrada de la consola de el usuario. Cuando el usuario introduce datos, el dispositivo de consola debe enviar una interrupción de hardware. La rutina correspondiente del manejador se denomina manejador de terminal.

El manipulador de la terminal es responsable de despertar los procesos que están bloqueados para la entrada de consola.

- Interrupciones de software

Las interrupciones de software (trampas) son los mecanismos por los cuales los programas del modo de usuario pueden transferir el control al código que se ejecuta en el modo de kernel. Las rutinas de servicio de interrupción de software típicamente contienen el código del sistema operativo para varias llamadas al sistema. Al regresar de una interrupción de software, la ejecución se reanuda a partir de la siguiente instrucción en el programa en modo de usuario. Un total de 15 interrupciones de software están disponibles para un programa en modo de usuario

La arquitectura XSM tiene un dispositivo de consola que actúa como interfaz de entrada/salida. El mecanismo que utiliza la XSM para estas operaciones (input/output) son las I/O por interrupciones ya que la manera de realizar estas operaciones es a través de la terminal. Son dos dispositivos especiales de flujo con identificadores predefinidos STDIN y STDOUT donde el primero se encarga únicamente de la lectura y otro de la escritura.

Las macros 'IN' 'INI' 'OUT' son las utilizadas a la hora de requerir interactuar con el usuario e invocar así al dispositivo de consola.

Cuando se ejecuta una instrucción de entrada por consola (IN) éste genera un contador que marca a cada instrucción (en modo usuario) luego de la instrucción IN, una vez alcanzado el umbral fijado por la XSM, el contador y simulador de máquina aguardan al ingreso de datos se inicia la entrada sin suspender lo que es la ejecución de la máquina hasta que se lea la misma entrada, se utiliza el puerto P0 para esto (ver figura) y cuando se ingresan los datos el contador se reinicia y se activa la interrupción de la consola dando como resultado que la ejecución de la máquina se transfiera al manipulador de interrupciones de la consola.

El mismo manipulador de terminal es el encargado de reactivar aquellos procesos que se encuentran en un estado bloqueado, esperando la entrada.

En cuanto a la salida (STDOUT) es un proceso sincrónico que trabaja de manera inmediata. Se utiliza el identificador OUT para la invocación al dispositivo consola. El puerto utilizado en este caso es el P1 (ver figura).

BIBLIOGRAFÍA

- Documentacion de eXpOS.