Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

CI-1310 Sistemas Operativos
Grupo 02
I Semestre

I Tarea programada: Nachos Virtual Memory

Profesor:
Francisco Arroyo

Estudiantes:
Pablo Esteban Aguilar Castro | B30105
Ana Cristina Soto ROjas | B26607

día de Mes del año

Índice

1. Introducción **3**

2. Objetivos **3**

3. Descripción **4**

4. Diseño **6**

5. Desarrollo **7**

Implantar el TLB administrado por software 10

Para ello es necesario crear un sistema de traducción manipulado por software que maneje los fallos TLB 10

Note que utilizando la bandera "DUSE_TLB.en tiempo de compilación, el hardware no utiliza las tablas de páginas; por lo que es necesario asegurarse de que el estado de la TLB es colocado correctamente en los cambios de contexto 11

Muchos de los sistemas lo que hacen es invalidar al TLB completamente cada vez que ocurre un cambio de contexto; las entradas se van recargando conforme las páginas son referenciadas nuevamente 12

Para el ítem 2, su esquema de traducción de páginas debe llevar cuentas de las páginas sucias y utilizar las banderas que el hardware coloca en las entradas del TLB 12

Implantar memoria virtual 12

Evalúe el desempeño de su sistema. Los fallos de cache (en este caso fallos TLB y fallos de página) pueden ser divididos en tres categoría: 12

Escriba un conjunto de programas de usuario "útiles"para demostrar fallos de TLB y páginas (pocos y muchos fallos) 12

6. Pruebas **13**

7. Manual de usuario **14**

Requerimientos de Software 14

1. Introducción

El proyecto consiste en implementar una memoria virtual para el sistema operativo Nachos con el fin de que se pueda correr programas de usuario que llenen la memoria física.

2. Objetivos

- Detectar los page Fault en NachOS.
- Manejar los Page Fault de NachOS para que la ejecución de los programas de usuario que presentan page faults se puedan terminar de ejecutar
- Implementar algoritmos de reemplazo de páginas
- Implementar swapping en NachOS

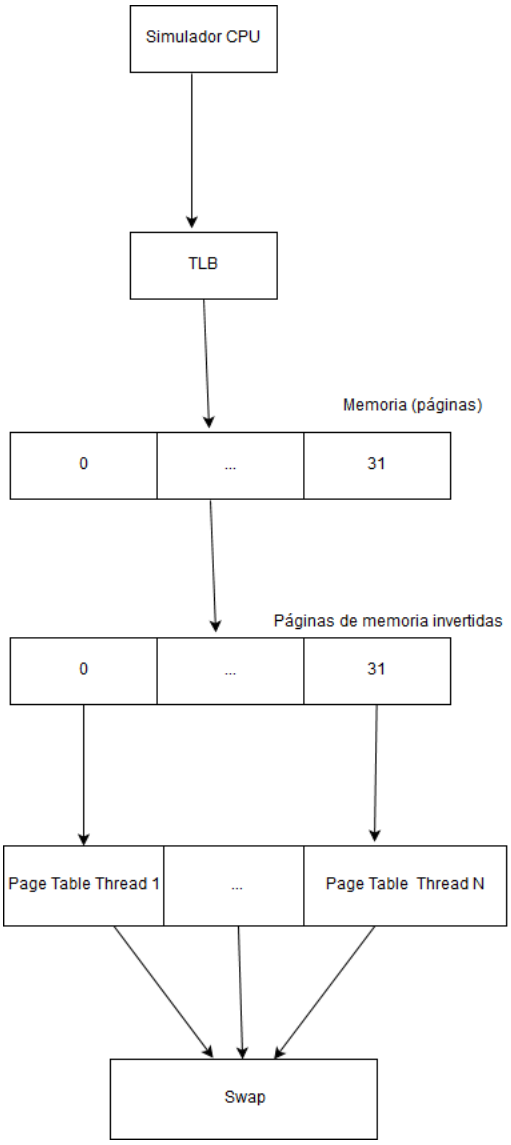
3. Descripción

- En la tercera fase de Nachos se investigará el uso de caching que se utilizará en esta tarea con dos propósitos:
  - Primero, se va a emplear un traductor de direcciones basado en software (Translation Lookaside Buffer [TLB]) como una memoria cache para la tabla de páginas, con el fin de dar la ilusión de un acceso rápido al traductor de direcciones virtuales que funcionaría en un espacio de direcciones muy grande
  - Además, se va a utilizar el disco como parte de la memoria principal, con el fin de proveer un espacio de direccionamiento (casi) ilimitado, con el desempeño cercano al de la memoria física
  - Para esta asignación no se provee código nuevo, el único cambio necesario de compilar el código ya existente (threads y userprog) con las banderas DVM y DUSE_TLB), incluidas dentro del Makefile del proyecto "vm"(-DVM y -DUSE_TLB); su tarea será escribir el código para manejar el TLB e implantar la memoria virtual
- En el proyecto 2 se utilizaron tablas de páginas para simplificar la asignación de memoria y aislar los fallos a un espacio de direcciones, de manera que no afecten a otros programas
- Para esta fase, el hardware no sabe nada de las tablas de páginas
- En su lugar, solo trabaja con un cache, cargado por software (el sistema operativo, alias usted) de las entradas de las tablas de páginas, denominado TLB
- En casi todas las arquitecturas modernas se incorpora un TLB a fin de aligerar las traducciones del espacio de direcciones
  - Dada una dirección de memoria (una instrucción que se desea buscar o un dato para cargar o guardar), el procesador primero busca en el TLB para determinar si el mapeo de la dirección virtual a la dirección física ya se conoce
  - Si es así (acierto TLB) entonces la traducción puede efectuarse rápidamente
  - Pero si el mapeo no se encuentra en el TLB (fallo TLB) entonces se deben acceder las tablas de páginas y/o de segmentos
  - En muchas arquitecturas, entre ellas Nachos, DEC MIPS y las HP Snakes, un fallo TLB causa una trampa al kernel del sistema operativo, que se encarga de efectuar la traducción, carga la traducción en el TLB y regresa al programa que efectuó el fallo
  - Esto permite al kernel del sistema operativo elegir entre cualquier combinación de tabla de páginas, tabla de segmentos, tabla invertida, etc., necesaria para llevar a cabo la traducción de la dirección
  - En sistemas que no utilicen TLB basado en software, entonces el hardware realiza la misma labor, pero en este caso el hardware debe especificar el formato exacto para las tablas de páginas y segmentos
  - Por esta razón las TLB manipuladas por software son más flexibles, al costo de ser un poco más lentas para manejar los fallos TLB. Si estos fallos son muy infrecuentes, el impacto en el desempeño de un sistema TLB manejado por software es mínimo
- La ilusión de tener memoria ilimitada es provista por el sistema operativo, utilizando la memoria principal como un cache para el disco (swap)
- Para esta asignación, la traducción de las direcciones de las páginas, permite la flexibilidad de traer páginas del disco en el momento en que se necesiten

- Cada entrada en el TLB posee un bit de validez:
  - Si el bit está encendido entonces la página virtual está en memoria, solo se debe actualizar el TLB
  - Si está apagado o la página virtual no está en el TLB, entonces se necesita una tabla de páginas administrada por software (SO, alias usted) para saber si la página está en memoria (cargando el TLB cuando se efectúe la traducción) o si la página debe ser traída del disco (swap)
- Adicionalmente, el hardware coloca el bit de uso cada vez que la página se utiliza y el bit de suciedad si el contenido fue modificado.
- Cuando el programa hace referencia a una página que no se encuentra en el TLB, el hardware genera una excepción de falta de página (PageFault exception), llevando el control al kernel (ExceptionHandler)
- El kernel, verifica en la tabla de páginas del hilo/proceso, si la página no está en memoria, entonces lee la página del disco, corrige la entrada en la tabla para apuntar a la nueva página y luego pasa el control al programa que produjo la excepción
- Por supuesto, el kernel debe encontrar primero espacio en la memoria principal para esta nueva página que debe ser traída, probablemente desocupando una casilla y escribiendo esa página removida en el disco (swap), si ésta fue modificada antes (dirty)
- Como en cualquier sistema de caching, el desempeño depende de la estrategia para determinar cuáles de las páginas permanecen en memoria y cuáles se mantienen en el disco En un fallo de página, el kernel debe decidir cuál de las páginas va a reemplazar; idealmente debe reemplazar aquella que no se va a utilizar por un largo tiempo, manteniendo en memoria solamente las páginas que van a ser referenciadas pronto Otra consideración importante, es que si la página que se reemplaza había sido modificada, ésta debe ser guardada en el disco antes de traer la nueva
- En muchos sistemas de memoria virtual (como UNIX) se evita ese gasto de tiempo extra, escribiendo las páginas al disco por adelantado, de manera que cualquier fallo de página posterior podrá ser completado más rápidamente

4. Diseño

El siguiente diagrama muestra el diseño para el proyecto del funcionamieneto de la memoria virtual de nachos:



Text

5. Desarrollo

Los archivos que se modificaron para realizar este proyecto Primero se modificó el Exception Handler para que cuando ocurría una excepción del sistema diferente a syscall se indique cuál es. Además en caso de que haya una PageFault Exception se llama a un metodo de la clase addrspace para cargar la página:

```
        void ExceptionHandler(ExceptionType which)
{
    int type = machine->ReadRegister(2);
    unsigned int dirLogica; //se usa para almacenar la direccion logica de la pagina en
         la que ocurrio el page fault
    unsigned int vpn; // se usa para averiguar el numero de pagina virtual en caso de
        que haya una page fault exception
/*if((type >= 0) && (type <= 13)){
        which = SyscallException;
        }*/
    DEBUG ('a', "Tipo de syscall %d y ExceptionType %d\n", type, which);
    switch ( which ) {
       case SyscallException:
          switch (type) {
             case SC_Halt:
                Nachos_Halt();             // System call # 0
                break;
             case SC_Exit:
             // System call # 1
                 Nachos_Exit();
                 break;
             case SC_Exec:
             // System call # 2
                  Nachos_Exec();
                  break;
             case SC_Join:
             // System call # 3
                  Nachos_Join();
                  break;
             case SC_Create:
             // System call # 4
                  break;
             case SC_Open:
                Nachos_Open();
             // System call # 5
                break;
             case SC_Read:
              //system Call #6
                  break;
             case SC_Write:
                Nachos_Write();
            // System call # 7
                break;
             case SC_Close:
             //System Call # 8
                  break;
             case SC_Fork:
                 Nachos_Fork();
             //System Call # 9
             break;
```

```
49                case SC_Yield:
50                    Nachos_Yield();
51                //System Call #10
52                break;
53                case SC_SemCreate:
54                    Nachos_SemCreate();
55                //System Call #11
56                break;
57                case SC_SemDestroy:
58                //System Call #12
59                break;
60                case SC_SemSignal:
61                //System Call # 13
62                break;
63                case SC_SemWait:
64                //System Call # 14
65                break;
66                default:
67                   printf("Unexpected syscall exception %d\n", type);
68                   ASSERT(false);
69                   break;
70            }break;
71                 case PageFaultException:
72                      printf("Excepcion de page fault \n");   // No valid translation
                          found
73                      dirLogica = machine->ReadRegister(39);
74                      vpn = dirLogica/PageSize;
75                      printf("Ocurre en la direccion: %d \n", dirLogica);
76                      printf(" En la pagina : %d \n", vpn);
77                    currentThread->space->Load(vpn);
78                      break;
79                 case  ReadOnlyException:     // Write attempted to page marked
80                       printf("Excepcion de read only");                      // "read-
                             only"
81                       ASSERT(false);
82                       break;
83                  case  BusErrorException:     // Translation resulted in an
84                        printf("Execepcion de Bus Error");
85                      // invalid physical address
86                         ASSERT(false);
87                         break;
88                  case AddressErrorException: // Unaligned reference or one that
89                                              // was beyond the end of the
90                                              // address space
91                       printf("Excepcion de Adress Error");
92                       ASSERT(false);
93                         break;
94                  case OverflowException:     // Integer overflow in add or sub.
95                       printf("Excepcion de over flow");
96                       ASSERT(false);
97                       break;
98                  case IllegalInstrException:
99                       printf("Excepcion de instruccion ilegal");
100                      ASSERT(false);
101                      break;
102                 default:
```

```
103                             printf( "Unexpected exception %d\n", which );
104                             ASSERT(false);
105                             break;
106         }
```

Además se modificó el constructor de Addrspace

Implantar el TLB administrado por software

Para ello es necesario crear un sistema de traducción manipulado por software que maneje los fallos TLB

- Revise la traducción de direcciones lógicas a físicas que ocurren en el método "Translate.[en] el archivo "translate.cc"del directorio "machine"

  Al revisar dicho método podemos observar que lo que hace es tomar una dirección virtual que ingresa y la convierte en una dirección física, considerando para ello una tabla de páginas o una TLB. Recordemos que este proceso de trasladar de lógico a físico se debe realizar ya que las páginas no están acomodadas de la misma forma física que a como se manejan físicamente por el sistema operativo. El método mencionado se encarga de revisar errores que pueden existir, como de alineamienteo entre otros y en caso de que exista alguno devuelve el tipo de la excepción.

- Investigue cómo se realizan las traducciones y los tipos de excepciones que se pueden generar, ponga especial atención a "PageFaultException"

  Dicho método recibe como parámetros la dirección física, la variable en la que se guardará la dirección dísica, el tamaño de memoria que se leerá y una variable booleana que nos indica el bit de solo léctura de la TLB, que recordemos que lo que nos dice es si el estado de esa página es solo de lectura y por lo tanto no podemos escribir en ella.

  La realización de la traducción consiste en revisar primero si existen problemas de alineación, es decir si la dirección lógica que ingresamos cabe en la página(en caso de que sea de tamaño 4 revisamos los últimos dos bits para ver que quepa, y en caso de que sea tamaño 2 revisamos el último bit), en caso de que esto no suceda entonces vamos a reportar el error AddressErrorException.

  Luego revisamos que exisa una TLB o una tabla de páginas, pero no ambas. Continuamos calculando la página física correspondiente a la lógica y a qué parte de esa página nos debemos mover (offset). Una vez que tenemos esto calculado revisamos si el número de página obtenido es válido dentro de los valores de la tabla, en caso contrario devuelve el error AddressErroException. Con esos mismos valores calculados revisamos si el valor de página obtenido es válido dentro de la tabla y en caso contrario devolvemos una excepción PageFaultException.

  Ahora suponiendo que no tenemos ninguno de los casos anteriores entonces procedemos a buscar si dicha página se encontraba en la tabla y era válida, entonces en caso de que la página no se encuentre en la TLB devuelve el error de PageFaultException. Pero, en caso de que sí se encuentre entonces entonces revisamos si la página es solo de escritura y queremos escribir en ella, en caso de ser así se devuelve la excepción ReadOnlyException.

  Luego en caso de que el número de página ingresado sea muy grande en comparación con los valores de físicos posibles es porque se cargó a la tabla un valor inválido a la tabla y entonces mostramos la excepción BusErrorException.

  En caso de que no ocurra ninguna de las excepciones anteriores entonces en la tabla indicamos decimos que esta página está en uso y si escribimos en ella entonces modificamos también el valor de si está sucia o no. Luego de esto guardamos la dirección física, que sería en número de página por el tamaño de la página y a eso le añadimos el extra- Luego revisamos que ese valor sea mayor a cero y menor al tamaño de la memoria. Una vez termiando esto entonces se hizo correctamente la conversión de la dirección.

- Determine qué cosas debe cambiar para que esa excepción no ocurra de nuevo cuando se re-ejecuta la instrucción

  Se debe cambiar que cada vez que se desee acceder a la tabla para consultar por una página se debe contemplar que el estado de la tabla tome en cuenta los cambios de contexto y no lo comparta entre ellos. Manteniendo así la diferencia entre la primera ejecución y las sigueintes ejecuciones de la instrucción y así no se tendría problema con que ocurra de nuevo la excepción

Se debe considerar en el constructor del addrSpace si la tabla de entrada de entrada es válidad entonces en la tabla de páginas se colocan todas las entradas de la tabla como inválidas y página física -1 para de esta manera controlar el estado de la tabla al iniciar el adress space, ya que en este caso no tendríamos el problema de que al ejecutar la instrucción de nuevo se continue y ocurra la excepción de nuevo.

Note que utilizando la bandera "DUSE_TLB.en tiempo de compilación, el hardware no utiliza las tablas de páginas; por lo que es necesario asegurarse de que el estado de la TLB es colocado correctamente en los cambios de contexto

- Revise el método RestoreState"de la clase .AddrSpace.El método hasta el momento solo asigna el número de página en la tabla y el número de páginas.
- Revise los .ASSERT"que se encuentran al principio del método "Translate"Los ASSERT son los siguientes:

```
ASSERT(tlb == NULL || pageTable == NULL);
ASSERT(tlb != NULL || pageTable != NULL);
```

  Lo cual significa que podemos tener solo una de las dos opciones propuestas, un tlb o un pageTable, entonces en esta parte nos aseguramos de que en efecto solo exista una.

- Intente correr el programa de usuario "haltz determine que ocurre En este caso al correr el programa halt vamos a tener un error ya que al intentar detener el error tendremos un problema con las condiciones de la TLB.
- Corrija el método RestoreState"para que no inicialice la variable "machine->pageTable"ni "machine->pageTableSize", utilice compilación condicional para que el proyecto de üserprog"no se estropee

  Para esto se corrigió de la siguiente manera:

```
            void AddrSpace::RestoreState()
{
#ifndef VM
    machine->pageTable = pageTable;
    machine->pageTableSize = numPages;
#else
DEBUG('a', "Restaurando el estado de la maquina ");
//se devuelven los indices de second chance
indiceSWAPSecondChance = 0;
indiceTLBSecondChance = 0;
//se hace una nueva TLB
machine->tlb = new TranslationEntry[TLBSize];
for (int i = 0; i < TLBSize; ++i)
{
machine->tlb[i].valid = false;
        machine->tlb[i].physicalPage = 0;
}
#endif
}
```

  Entonces ahora cuando se restaure un estado vamos a tener que guardaremos las entradas de la TLB e invalidamos las que tenía la TLB para no interferir con la próxima vez que se use por otro programa.

Muchos de los sistemas lo que hacen es invalidar al TLB completamente cada vez que ocurre un cambio de contexto; las entradas se van recargando conforme las páginas son referenciadas nuevamente

Para el ítem 2, su esquema de traducción de páginas debe llevar cuentas de las páginas sucias y utilizar las banderas que el hardware coloca en las entradas del TLB

Implantar memoria virtual

- Para ello se necesitan rutinas para mover páginas del disco a la memoria y viceversa
- Se recomienda que utilice el sistema de archivos de Nachos como área de respaldo (backing-store), de esta manera, cuando se implante el sistema de archivos en la asignación 4, es posible utilizar el sistema de memoria virtual como un caso de prueba
- A fin de encontrar las páginas que no se referencian y sacarlas cuando ocurre un fallo de página, es necesario llevar pista de todas las páginas que están siendo utilizadas en el sistema
- Una manera simple de hacer esto, es empleando un çore map", que es básicamente una tabla de páginas invertida, en lugar de traducir números de página virtual en números de páginas físicas (marcos), un çore map"traduce de número de página física al número de página virtual del hilo/proceso que está utilizando esa página física

Esto se hace con ayuda del archivo swap. Además para el reemplazo se utilizará el algoritmo de second chance, implementado en el archivo addrspace.cc

Evalúe el desempeño de su sistema. Los fallos de cache (en este caso fallos TLB y fallos de página) pueden ser divididos en tres categoría:

- Çompulsory misses", aquellos que ocurren debido a la primera referencia de un item; siempre la página debe ser traída del disco y colocada en la memoria y en el TLB
- Çapacity misses", aquellos debidos al tamaño del cache; si el "working set"del programa de mayor que la memoria principal o que el número de entradas en el TLB, el programa incurre en fallos. Estos ocurren en un cache que no es infinito
- Çonflict misses", aquellos debidos a la política de reemplazo del cache. No ocurrirían si el cache utilizara una política óptima para efectuar el reemplazo de sus páginas, para el mismo programa en un cache del mismo tamaño

Escriba un conjunto de programas de usuario "útiles"para demostrar fallos de TLB y páginas (pocos y muchos fallos)

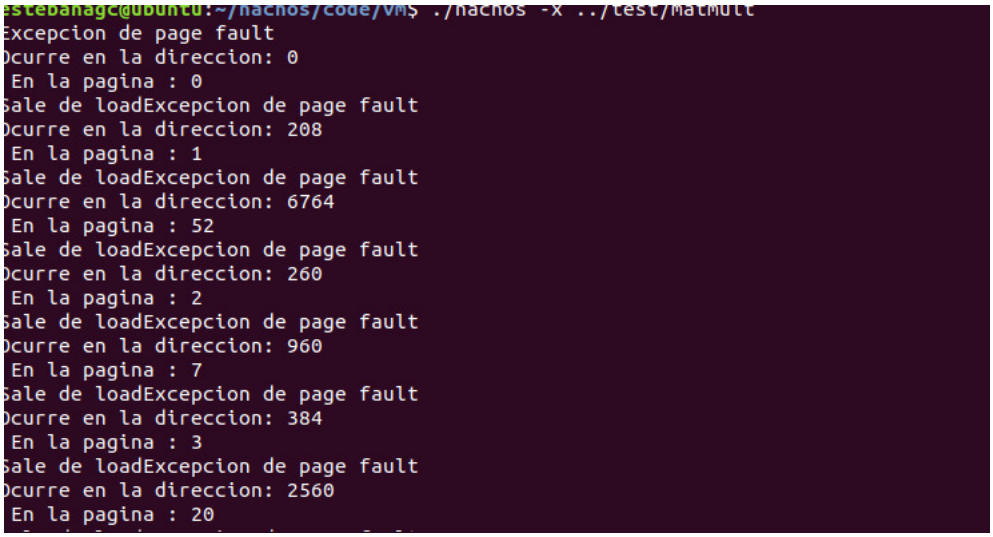
En otras palabras, escriba un programa de usuario de prueba que muestre cuando ocurren pocos fallos en el TLB; otro que muestre cuando ocurren pocos fallos de página; otro que muestre cuando ocurren muchos fallos de TLB; etc. Como ejemplo, se presentan los programas "sort.cz "matmult.c.en el directorio "test"que presentan un gran número de çonflict misses"para la mayoría de políticas de administración Para cada caso de prueba, explique el desempeño de su sistema e indique como se podría mejorar

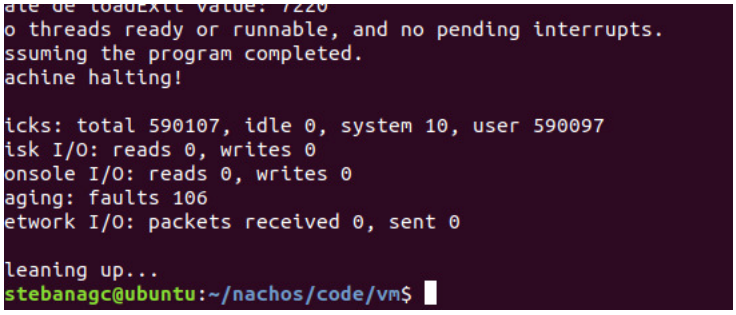
6. Pruebas

```
estebanagc@ubuntu:~/nachos/code/vm$ ./nachos -x ../test/sort
Excepcion de page fault
Ocurre en la direccion: 0
 En la pagina : 0
Sale de loadExcepcion de page fault
Ocurre en la direccion: 208
 En la pagina : 1
Sale de loadExcepcion de page fault
Ocurre en la direccion: 5868
 En la pagina : 45
Sale de loadExcepcion de page fault
Ocurre en la direccion: 260
 En la pagina : 2
Sale de loadExcepcion de page fault
Ocurre en la direccion: 704
 En la pagina : 5
Sale de loadExcepcion de page fault
Ocurre en la direccion: 768
 En la pagina : 6
Sale de loadExcepcion de page fault
Ocurre en la direccion: 896
 En la pagina : 7
```

```
 En la pagina : 34
Sale de loadExcepcion de page fault
Ocurre en la direccion: 32
 En la pagina : 0
Sale de loadExit value: 1023
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 17351562, idle 0, system 10, user 17351552
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 2938
Network I/O: packets received 0, sent 0

Cleaning up...
```

```
estebanagc@ubuntu:~/nachos/code/vm$ ./nachos -x ../test/halt.lab
Excepcion de page fault
Ocurre en la direccion: 0
 En la pagina : 0
Sale de loadExcepcion de page fault
Ocurre en la direccion: 208
 En la pagina : 1
Sale de loadExcepcion de page fault
Ocurre en la direccion: 1260
 En la pagina : 9
Sale de loadMachine halting!

Ticks: total 25, idle 0, system 10, user 15
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 3
Network I/O: packets received 0, sent 0

Cleaning up...
estebanagc@ubuntu:~/nachos/code/vm$
```





7. Manual de usuario

Requerimientos de Software

- **Sistema Operativo:** Linux (también funciona en Windows 10 si se tiene el subsistema Linux de Windows).
- **Arquitectura:** [32 bits, 64 bits]
- **Ambiente:** [Code::Blocks, NetBeans,...]

Referencias

[1] SILBERSCHATZ, A., AND PETERSON, J. L. *Operating system concepts*. Addison-Wesley Reading, MA, 2012.