UASLP
Universidad Autónoma
de San Luis Potosí

FACULTAD DE
INGENIERÍA

Práctica 0. Administrador de Memoria en Nachos.

Sistemas operativos B.
Profesora: M.I. Ortiz Hernández Marcela.
grupo: 2402-03.
Semestre: 2019-2020/II.

Cantu Olivares Pedro de Jesus.
Martinez Murillo Raul Antonio.

26/Febrero/2020.

1.- Resumen:

Para realizar esta práctica debíamos tener conocimiento previo del funcionamiento y el código de nachos, el cual se pedía en las actividades 1 y 2. Estos conocimientos abarcaban las características de memoria de la arquitectura, la técnica de asignación de memoria, traducción de direcciones y como se prueba en administrador de memoria.

2.- Instrucciones:

Modificar el código de Nachos para que al ejecutar un programa realice lo siguiente (en el orden en que se especifica a continuación):
a) Mostrar en pantalla el tamaño del proceso en bytes (en decimal).
b) Imprimir en pantalla la cantidad de páginas n que forman el proceso (que representan la cantidad de marcos que se requieren para su carga y ejecución).
c) Imprimir el contenido de la tabla de páginas (índice, marco y bit de validez).
d) Mostrar el mapeo de memoria conforme se vaya ejecutando el proceso. Con el siguiente formato:
Dirección Lógica No. De Página(p) Desplazamiento(d) Dirección Física.

3.- Descripción de la técnica de asignación de memoria implementada en Nachos.

a) Características de memoria en la arquitectura:

• Especificar cómo se determina la cantidad la memoria física total (indicando cálculos y archivos necesarios):
La cantidad de memoria física se calcula multiplicando la cantidad de marcos(paginas físicas en nachos) que están asignados a la constante **NumPhysPages** multiplicados por la constante **PageSize** que corresponde al tamaño de las páginas y el cual es asignado al tamaño del segmentos del disco (**SectorSize**).

-En /nachos/machine/machine.h:

```
#define PageSize      SectorSize     // set the page size equal to
                       // the disk sector size, for
                       // simplicity

#define NumPhysPages  32
#define MemorySize     (NumPhysPages * PageSize)
```

-En /nachos/machine/disk.h:

```
#define SectorSize          128    // number of bytes per disk
sector
```

• Describa cómo se organiza la memoria física:
La memoria física se organiza en marcos de página(en el código de nachos se utiliza la variable **NumPhysPages**), cada uno de estos es un sector de la memoria, el tamaño está definido por la variable **SectorSize** en **disk.h**, todos los sectores deben de ser del mismo tamaño.

b) Descripción de la técnica de asignación de memoria implementada:

La asignación de memoria para los procesos se hace mediante paginación. Para ello se tiene una clase llamada **TranslationEntry.** La cual define los campos de una página. La tabla de páginas consiste en un arreglo de estas.

c) Estructuras de datos definidas para la técnica de asignación de memoria (indicando nombres de variables, de clases, métodos y archivos utilizados):

-En /nachos/userprog/addrspace.h (declaración de la tabla de páginas dentro de la definición de la clase AddrSpace):

```
private:
     TranslationEntry *pageTable;    // Assume linear page table
translation
```

-En /nachos/userprog/addrspace.cc en el constructor de la clase AddrSpace (instancia de la tabla de paginas con el numero de paginas calculado):

```
// how big is address space?
     size = noffH.code.size + noffH.initData.size +
noffH.uninitData.size
           + UserStackSize;    // we need to increase the size
                           // to leave room for the stack

     numPages = divRoundUp(size, PageSize);

pageTable = new TranslationEntry[numPages];
```

d) Descripción del procedimiento de asignación de memoria al proceso (indicando nombres de variables, de archivos y métodos utilizados):

-En /nachos/userprog/addrspace.cc
La asignación de memoria se lleva a cabo en el constructor de **AddrSpace** por lo que se describe dicho constructor a continuación:

El constructor de **AddrSpace** recibe como parámetro un Openfile (declarado en /nachos/filesystem/openfile.h) el cual es un archivo ejecutable que el sistema de archivos de nachos permite leer y escribir. se llama al método **ReadAt()** del ejecutable,para saber su

tamaño. Se llevan a cabo condicionales de control para la correcta lectura del ejecutable y se calcula el tamaño total del mismo. Así con el tamaño calculado se calcula posteriormente el número de páginas necesarias para cargar el programa en memoria.
Se instancia la tabla de páginas con el número de páginas calculado y se asigna a cada una el valor de sus atributos concernientes, mediante un ciclo for. Finalmente se carga el ejecutable en la memoria.

e) Descripción del procedimiento para el mapeo de direcciones lógicas a físicas (indicando nombres de variables, de archivos y métodos utilizados):

-En /nachos/machine/translate.cc

En este caso primero se define un método para la clase **Machine** en el archivo **tranlate.cc**.

El método **Machine::Translate** recibe como parámetro una dirección virtual(**int virtAddr**), una dirección fisica por referencia para retornarla despues de la traducción(**int* physAddr**), el tamaño del proceso para checar que la dirección virtual no acceda a una dirección mayor a la que corresponde(**int size**) y una variable booleana para indicar si se está haciendo una escritura o no(**bool writing**).

Lo primero es declarar una variable vpn que seria el numero de pagina al que se quiere acceder dentro de la tabla de páginas, offset que es el desplazamiento dentro de la memoria física, y otra pageFrame.
Se declara un apuntador a una página (***entry**).

Se determina si el proceso está queriendo acceder a una locación de memoria que no le pertenece, si es así lanza una excepción del tipo **AddressErrorException**.

Se determina si hay un TLB o solo tabla de páginas(no ambos).

Se calcula el vpn con la fórmula:
```
vpn = (unsigned) virtAddr / PageSize;
```

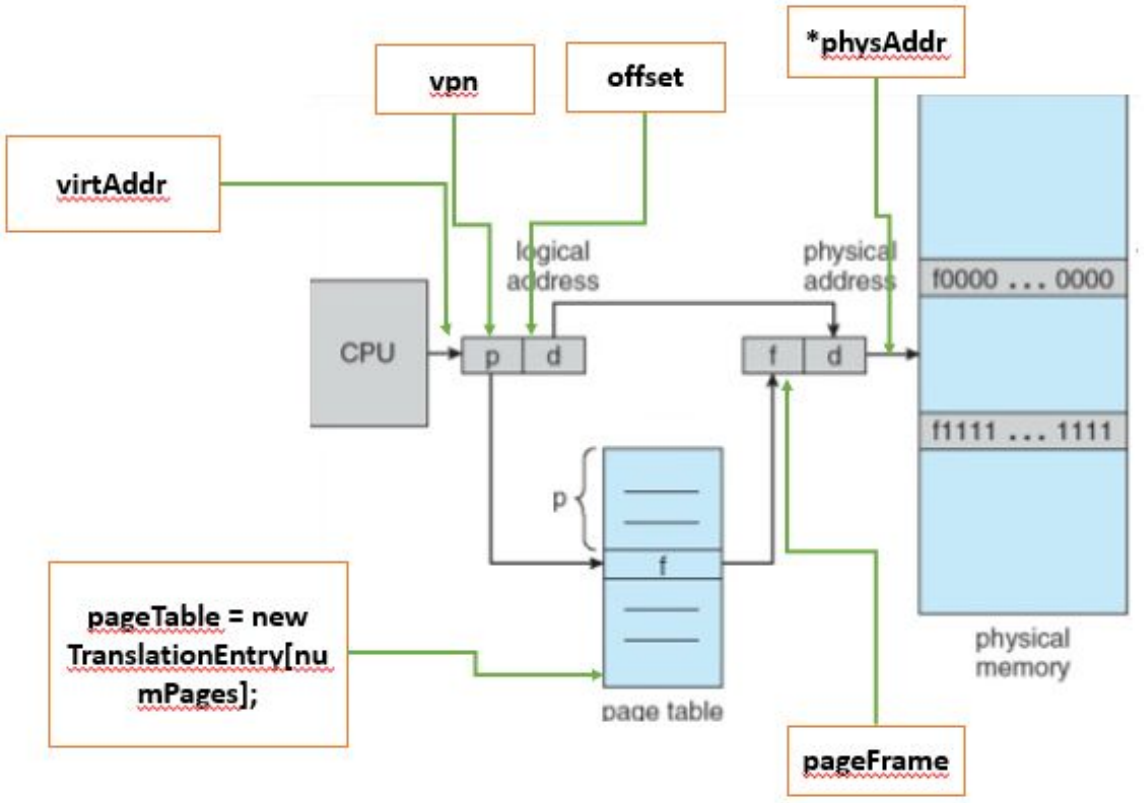
y el offset con la fórmula:
```
offset = (unsigned) virtAddr % PageSize;
```

si no hay TLB y todo es correcto con respecto a las direcciones, se calcula la tabla a la que hay que acceder:
```
entry = &pageTable[vpn];
```

Se calcula la página física(marco) accediendo al valor que tiene la página(entry) como marco correspondiente:
```
pageFrame = entry->physicalPage;
```

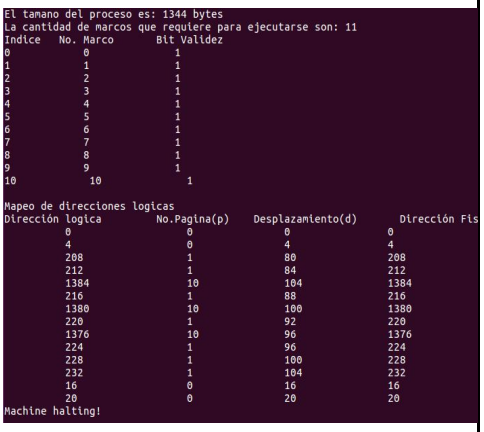
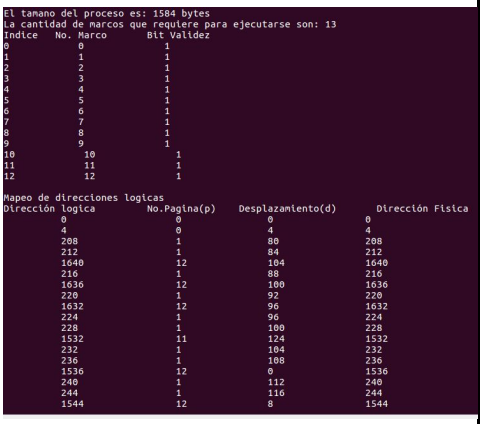
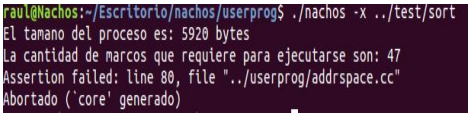
finalmente se cambia el valor de la dirección física con la fórmula siguiente:
```
*physAddr = pageFrame * PageSize + offset;
```

f) Dibuje el esquema de traducción de direcciones usando paginación y especifique la equivalencia con las estructuras de datos, fórmulas y variables de Nachos:



g) Indicar la orden de ejecución desde la línea de comandos para probar el administrador de memoria:
userprog$: ./nachos -x <Programa_a_ejecutar>

4.- Llene la siguiente tabla ejecutando para uno de los programas de prueba

Programa de prueba	Orden de ejecución	Tamaño (en bytes)	Número de marcos requeridos para su ejecución	Salida
halt	./nachos -x ../test/halt	1344	11	
shell	./nachos -x ../test/shell	1584	13	 Unexpected user mode exception 1 7 Assertion failed: line 61, file "../userprog/exception.cc" Aborted (core dumped)
sort	./nachos -x ../test/sort	5920	47	

matmult	./nachos -x ../test/matmult	6880	54	

5.- Problemas presentados en la práctica y soluciones.

En general no tuvimos ningún problema realizando esta práctica, puesto que la Actividad 1 y 2 de nachos nos dieron las herramientas necesarias para poder hacerla sin ningún problema.