AYUDANTÍA T2

Germán Leandro Contreras Sagredo - Raimundo José Herrera Sufán

IIC2333 - Sistemas Operativos y Redes

INTRODUCCIÓN

INTRODUCCIÓN

Los objetivos de esta ayudantía son:

- Entender el funcionamiento de las dos partes de la tarea.
- Resolver las dudas que surjan durante la explicación de lo pedido.

PARTE 1 - DOER

OBJETIVOS

1. Repasar la utilidad de `fork`, `exec` y `wait`.
2. Aclarar el funcionamiento esperado de `doer`.

FORK

- Se usa para crear nuevos procesos que se ejecutan concurrentemente con el proceso padre o creador.
- Se ejecuta el mismo código desde la línea donde se hizo fork.
- Es un proceso distinto.

```
void main() {
    int pid = fork();
    if (pid == 0) {
        printf("Nací!\n");
    }
    else {
        printf("Nuevo hijo %d!\n", pid);
    }
}
```

EXEC

- Se usa para reemplazar todo el contenido de un proceso por otro programa.
- Sirve para ejecutar distintas cosas después de haber hecho `fork`.

```
void main() {
    if (fork() == 0) {
        printf("Nací!\n");
        char cmd[10] = "ls";
        char *args[] = {cmd, NULL};
        execvp(cmd, args);
    }
}
```

WAIT

- Se usa para esperar la finalización de un proceso hijo.
- Se podría usar para reasignar recursos escasos.

```
if ((pid = fork()) == 0) {
    sleep(2);
    exit(1);
} else do {
    if ((pid = waitpid(pid, &status, 1)) == 0) {
        printf("Still running!\n");
        sleep(1);
    } else {
        printf("Exit!\n");
    }
} while (pid == 0);
```

DOER

- Las instrucciones pueden ser (y serán) otros programas compilados en C.
- ¿Qué hacer cuando $m > n$?
- Tiempo paralelo y tiempo secuencial
- La concurrencia viene dada por la cantidad máxima de procesos disponibles para ejecutarse.

PARTE 2 - MEMORY SIMULATOR

OBJETIVOS

1. Repasar cómo funciona el manejo de la memoria virtual en la práctica mediante una simulación.
2. Extender dicho repaso a partir de la paginación multinivel.

ESTRUCTURA

- Direcciones de 28 bits.
  - Primeros 20 bits de direccionamiento de páginas. ¿Sabemos cuántas páginas hay?
  - Últimos 8 bits de offset. ¿De qué tamaño debe ser una página?
- Tabla de páginas de nivel variable (de 1 a 5, inclusive).
- TLB de 64 entradas.
- Memoria física con 256 frames de 2^8B. ¿Podía ser cada frame de otro tamaño?

IMPORTANTE

Esta parte de la tarea considera dos funcionalidades distintas que deben implementar.

DIRECCIONAMIENTO EFICIENTE

Sea n el número de niveles que usaremos en la simulación.

- Si $n = 1$, simplemente tenemos 2^{20} páginas y una tabla de páginas de 2^{20} entradas. ¿Por qué?
- ¿Qué pasa si $n = 2$? ¿Es constante el número de entradas que tendremos por nivel?

DIRECCIONAMIENTO EFICIENTE

- Debemos definir el número de bits a utilizar por cada nivel.
- Lo que equivale a ver el número de tablas posible que se puede tener dentro del siguiente nivel.
- Queremos que la suma del espacio que ocupa una tabla en cada nivel sea mínimo.

Una vez definido el número de bits a utilizar por nivel, pasamos a la simulación.

SIMULACIÓN DE PAGINACIÓN

- Se leerá una lista de direcciones virtuales que va de 0 a 268435455. ¿Tiene sentido este rango?
- Por cada dirección recibida, se debe obtener el marco físico asociado y el contenido de la dirección.
- ¿Qué marco físico se asigna al principio? → Considere una asignación fully associative.
- ¿Cómo consigo el contenido en un principio? → `data.bin`, el que será nuestro 'disco'.

SIMULACIÓN DE PAGINACIÓN

Los pasos a seguir, por cada dirección, son:

1. Buscar en la TLB la dirección. ¿Cuántos bits poseerá cada entrada de esta?
2. TLB-*hit* → El frame se obtiene de la TLB.
3. TLB-*miss* → Se baja hasta el último nivel de paginación para obtener el frame, o un page-fault.
4. El *frame* está → Se almacena la dirección en la TLB (usar LRU si está llena).
5. *Page-fault* → Se busca el dato en `data.bin` y se carga en un frame (usar LRU si no quedan disponibles). Consiguientemente se almacena la dirección en la TLB. Importante acceder desde la primera palabra del frame, es decir, offset igual a cero.

SIMULACIÓN DE PAGINACIÓN

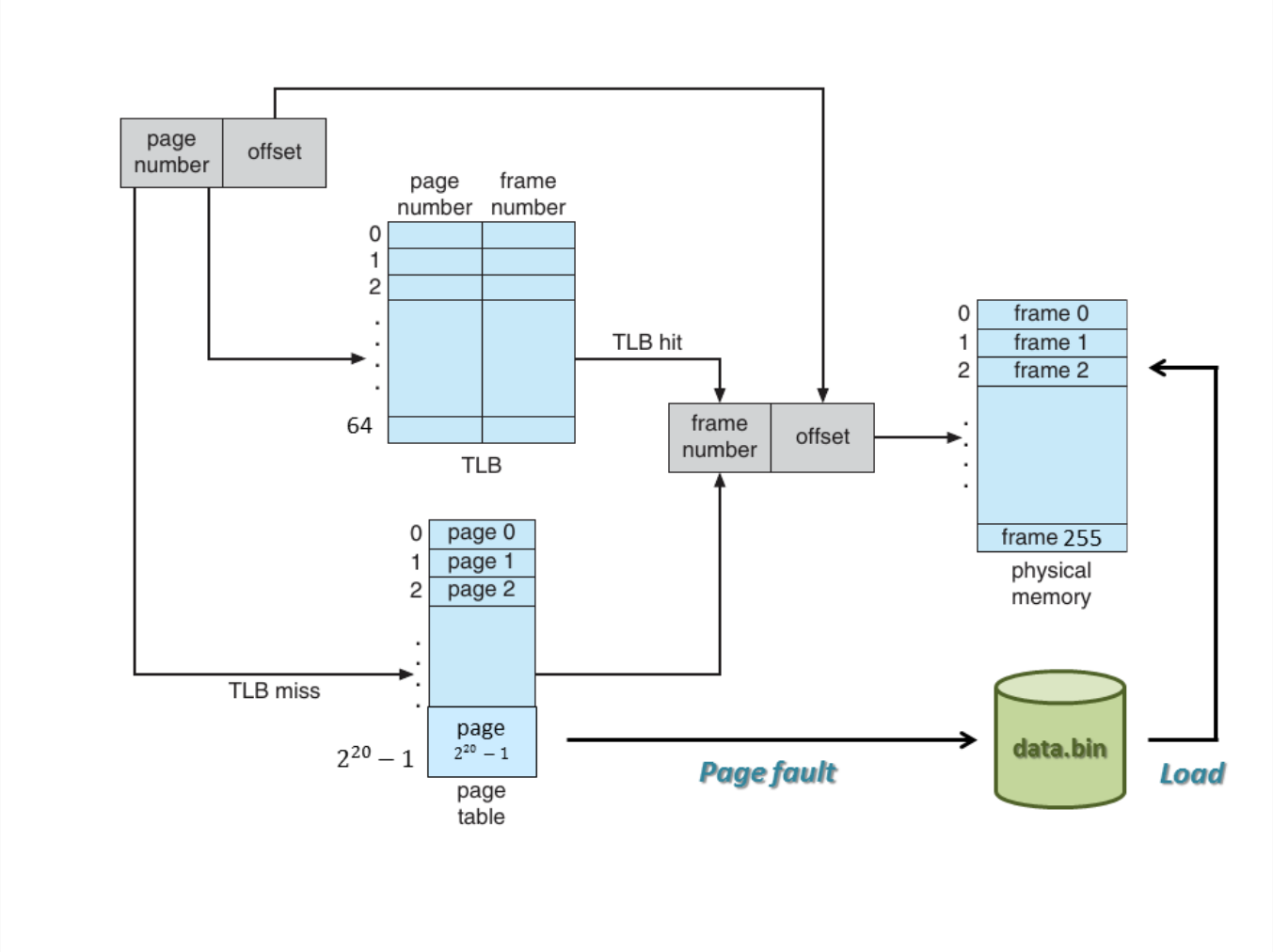

Figura: Diagrama del funcionamiento general de la simulación para n = 1.

SIMULACIÓN DE PAGINACIÓN

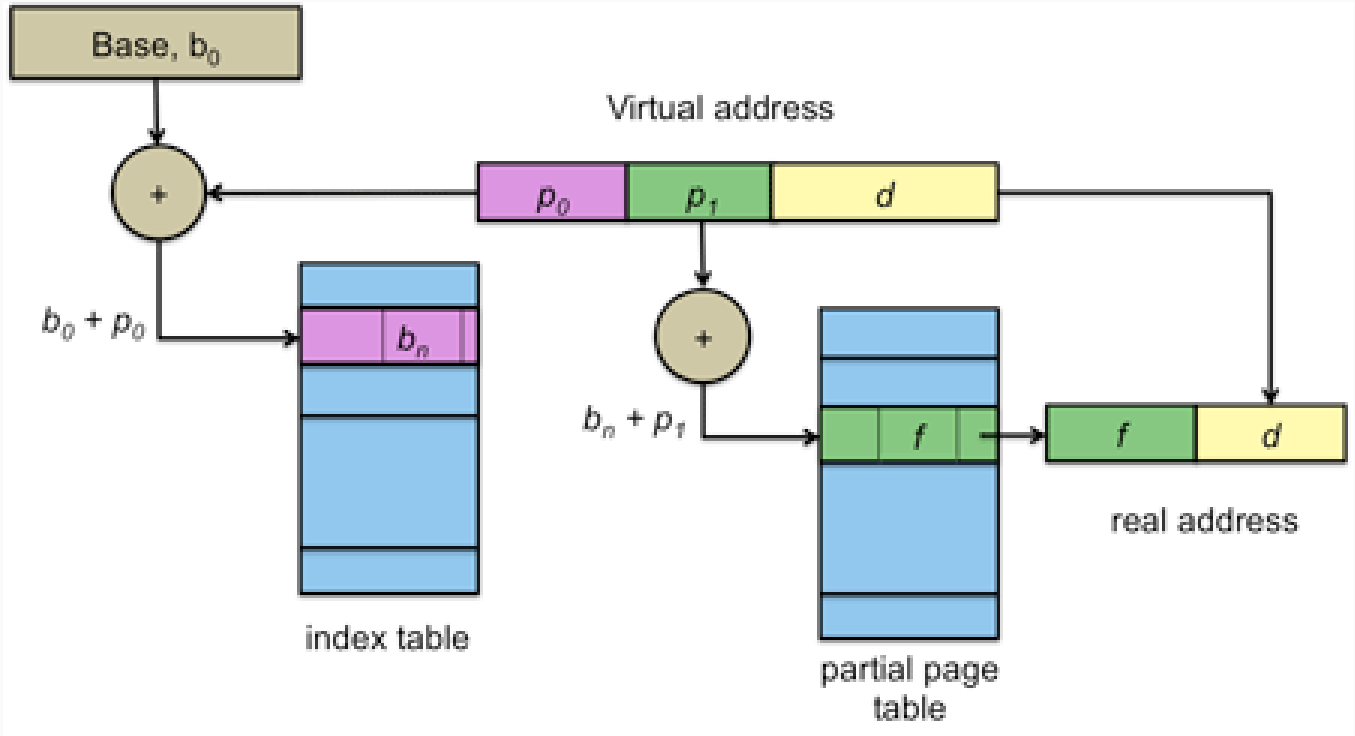

Figura: Diagrama de cómo se podría ver **solo** el direccionamiento para n = 2.

SIMULACIÓN DE PAGINACIÓN

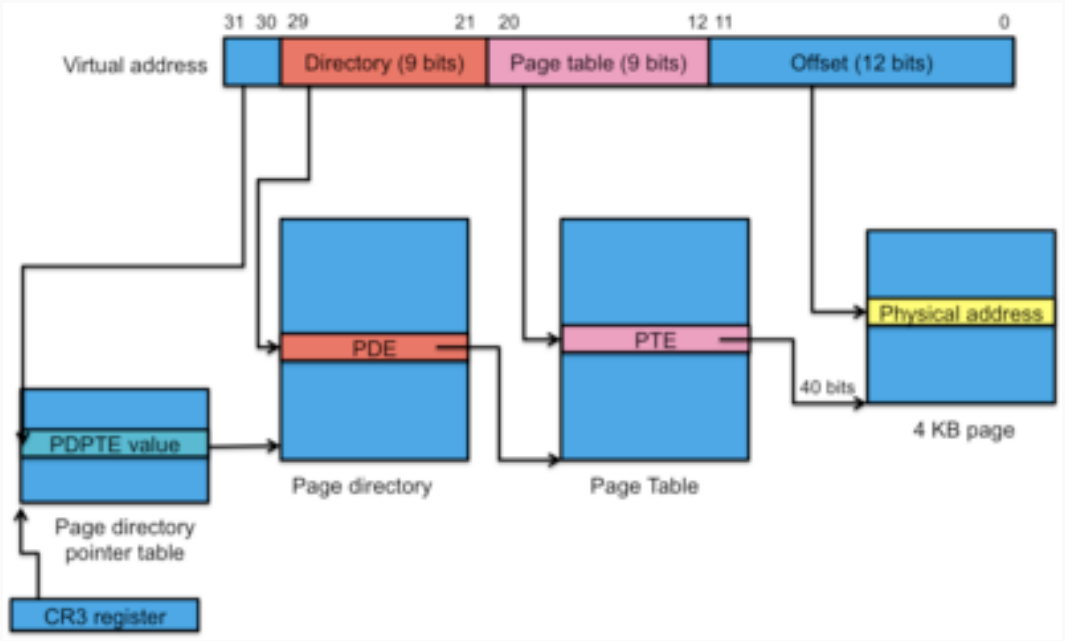

Figura: Diagrama de cómo se podría ver **solo** el direccionamiento para $n = 3$.

SIMULACIÓN DE PAGINACIÓN

A medida que aumenta n, el diagrama se complejiza más... no obstante, los pasos a seguir son los mismos.

No existe un formato fijo a seguir, pero se recomienda:

- **TLB:** Arreglo de 64 posiciones.
- **Tabla de páginas:** Arreglo de x posiciones (dependiendo de los bits). Si es de un nivel mayor, que posea punteros a los arreglos correspondientes de más bajo nivel.
- **Memoria física:** Arreglo de 256 posiciones con punteros a `char*` de 256 bytes.

A gusto del consumidor.

Direccionamiento eficiente

- Asegúrese de obtener la mejor combinación de bits por nivel antes de comenzar con la simulación.
- Para esto, al no conocer la cantidad bits necesarios para tener la dirección de la tabla del siguiente nivel, **debe asumir** que esta es igual a la cantidad de bits de direccionamiento de dicho nivel (tal como se hace en el ejemplo del enunciado). Esto estandarizará los cálculos que haga.
- **No se asignará puntaje si su programa no hace el cálculo del óptimo.** Es decir, no se tomarán en cuenta las tareas que posean los valores según nivel almacenados como constantes u obtenidos desde programas en otros lenguajes de programación (aunque estén correctos).

Simulación de paginación

- Como se podrá dar cuenta, no es necesario que inicialice **todas** las tablas de **todos** los niveles. Puede hacerlo a medida que la simulación lo amerite.
- No mantengan `data.bin` en memoria, tendrá menos puntaje y su programa **no correrá bien**. Las funciones `fopen`, `fread`, `fseek` y `fclose` lo ayudarán.
- Por el formato del 'disco', deben hacer lectura binaria ('`rb`').
- Trabaje con `unsigned_int` (entero sin signo).
- En general se hará reemplazo a partir de **LRU**, pero no hay limitación en cuanto a la forma en la que puede medir el tiempo del último acceso por frame o entrada en la **TLB**.

FIN