El Sistema Operativo XV6 - Capítulo 3 - Parte 1

Carlos Santiago Sandoval Casallas

csandovalc@unal.edu.co

Departamento de Ingeniería de Sistemas e Industrial
Universidad Nacional de Colombia

7 de mayo de 2025

UNIVERSIDAD NACIONAL DE COLOMBIA

Agenda

UNIVERSIDAD NACIONAL DE COLOMBIA

Descripción

Anteriormente se trató el aislamiento y la importancia de brindar al proceso la ilusión de un espacio de direcciones y memoria único. Las tablas de páginas son el mecanismo más empleado por los sistemas operativos para cumplir con este objetivo [1], estas determinan el significado de las direcciones de memoria y las partes de la memoria física que son accesibles.

Esta estrategia es ampliamente usada, esto debido, a que permite llevar a cabos “trucos” con la memoria, por ejemplo, asignar la misma página de trampolín en varios espacios de direcciones, proteger las pilas de kernel y del usuario mediante páginas sin asignar, esto último, con el fin de evitar que en caso de un desbordamiento de la memoria las pilas sean sobreescritas.

UNIVERSIDAD NACIONAL DE COLOMBIA

Hardware de paginación

Se destaca que: **las instrucciones de RISC-V manipula las direcciones virtuales**. La memoria física es indexada a través de **direcciones físicas**, el hardware de la tabla de páginas que brinda RISC-V se encarga de conectar estos dos tipos de direcciones, asignando cada dirección virtual a una dirección física.

XV6 se ejecuta sobre un Sv39 RISC-V, esto indica que solo se emplean los 39 bits menos significativos de una dirección virtual de 64 bits, los 25 bits restantes no se utilizan.

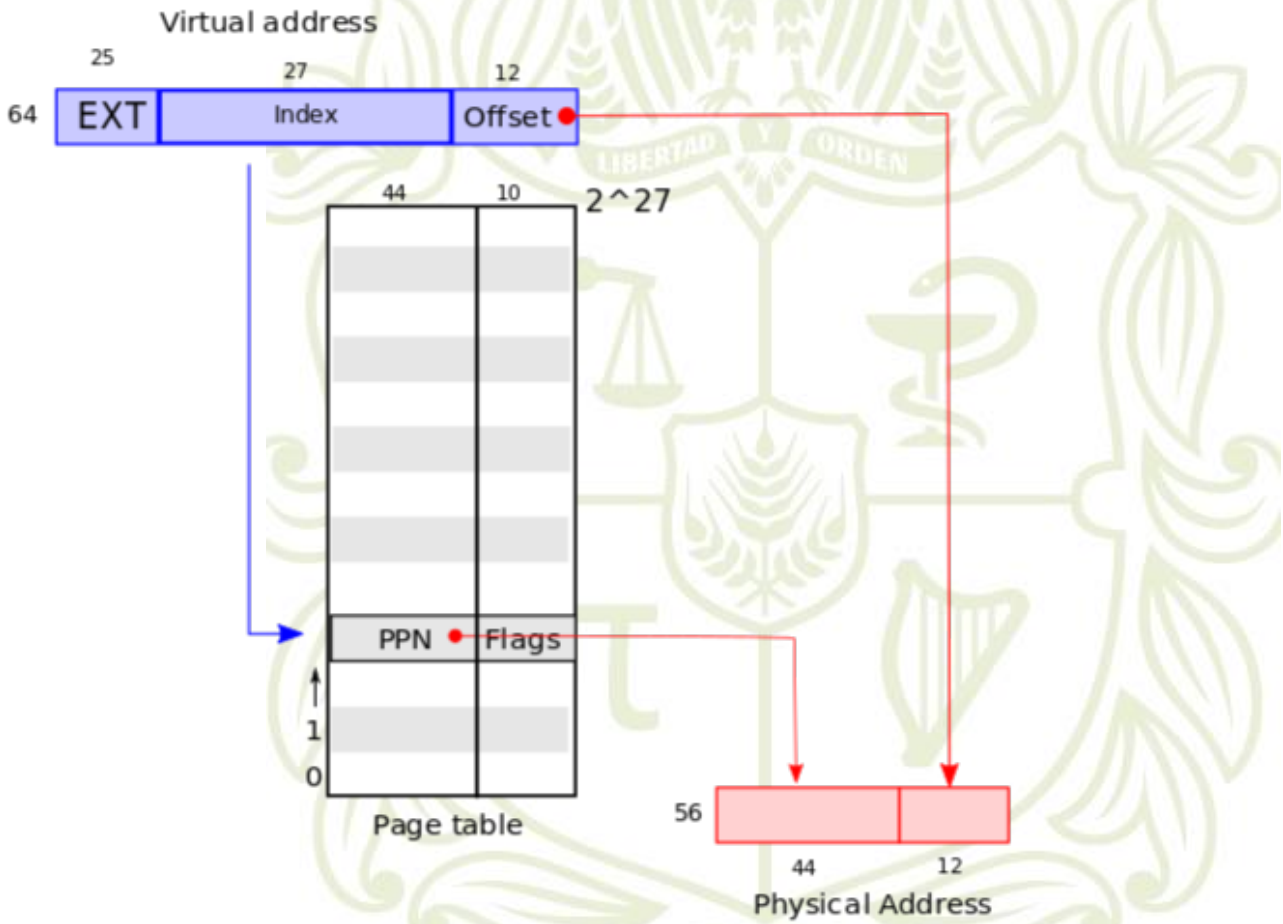
Una tabla de páginas **(figura 1)** es una matriz con 2^{27} entradas, estas reciben el nombre de PTE (**P**age **T**able **E**ntry). Cada PTE contiene un número de página física o PPN (**P**hysical **P**age **N**umber) de 44 bits, además, se emplean 10 bits adicionales para algunos indicadores.

El hardware de paginación traduce una dirección virtual, emplea 27 bits, **los más significativos**, de los 39 disponibles, estos bits permiten indexar cada PTE, del cual se extrae el PPN almacenado, los 44 bits del PPN serán los 44 bits más significativos de una dirección física de 56 bits, donde los 12 bits restantes son extraídos de los bits menos significativos de la dirección virtual original.

Que es una página?

En XV6 se define una página como un segmento de memoria alineado, cuyo tamaño es de 2^{12} (4096) bytes. [1]

Figura: Sistema de direcciones virtuales y físicas de RISC-V, con una tabla simplificada. [1]



Fuente: xv6: a simple, Unix-like teaching operating system [1]

En Sv39 RISV-V, los 25 bits superiores **no se utilizan para la traducción**. Por su parte, la dirección física tiene espacio para crecer, dado que el PTE emplea 54 de los 64 bits disponibles, **44 para el PPN y 10 para los indicadores**, por lo que puede crecer estos 10 bits, esta fue una decisión de los diseñadores de RISC-V con base en el futuro.

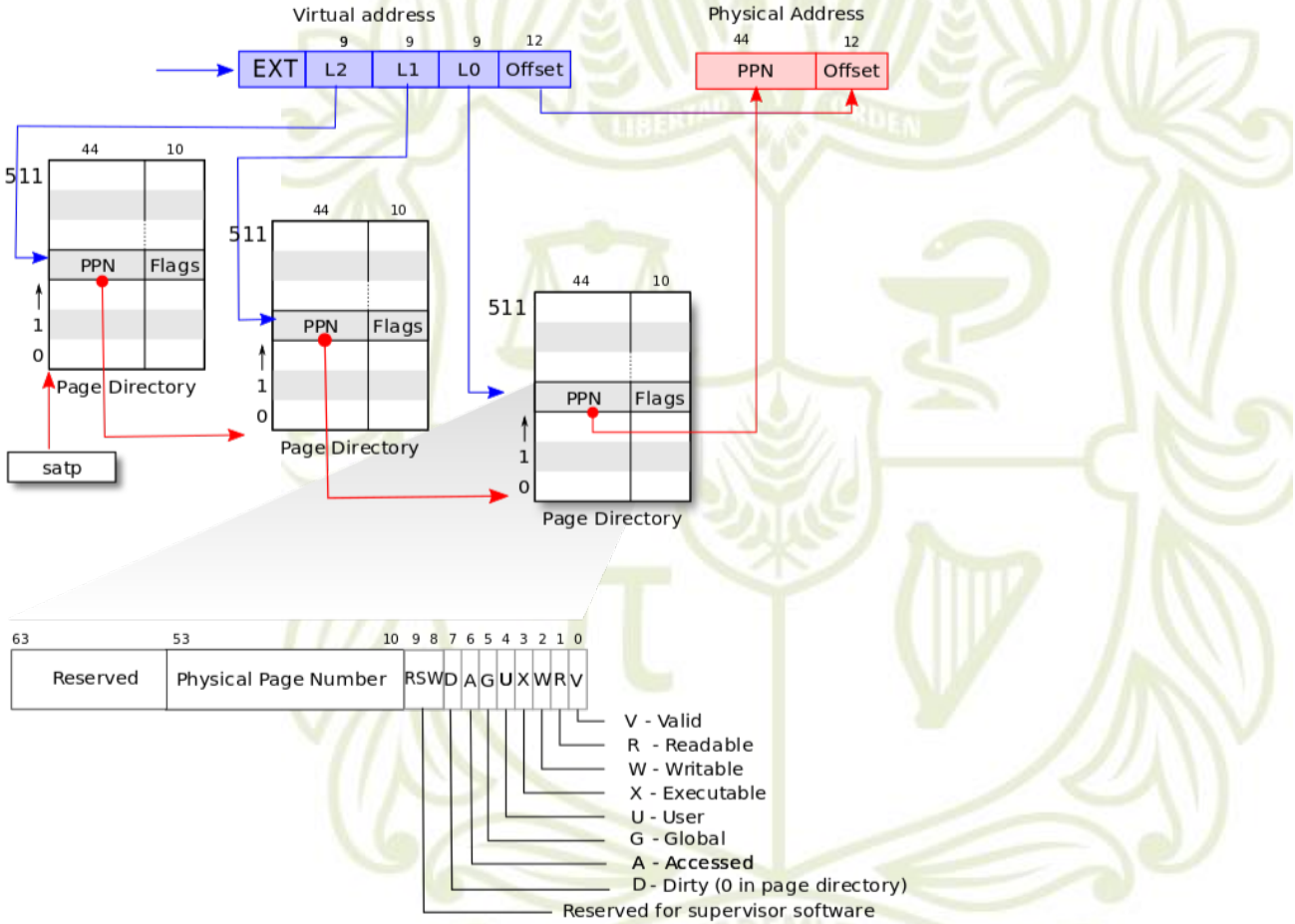
Si consideramos el sistema lineal ya mostrado, se pueden representar un total de 2^{39} bytes en la memoria virtual, lo que sería aproximadamente 512 GB, esto se considera como suficiente espacio para que las aplicaciones se puedan ejecutar sobre la computadora RISC-V, por su parte, se pueden representar 2^{56} bytes en la memoria física, permite al sistema abordar y adaptar múltiples dispositivos. [1]

En caso de que se requiera de mas capacidad, ya se ha definido Sv48, el cual posee direcciones virtuales de 48 bits. [1]

UNIVERSIDAD NACIONAL DE COLOMBIA

Traducción en tres pasos

Figura: Sistema de traducción de direcciones. [1]



Fuente: xv6: a simple, Unix-like teaching operating system [1]

Una CPU RISC-V traduce una dirección virtual a una física en tres pasos. La tabla de páginas se almacena en la memoria física como un **árbol de tres niveles de profundidad**.

La raíz del árbol es una tabla de páginas de 4096 bytes que contiene 512 PTE de 64 bits cada uno, cada PTE contiene la dirección física de la tabla de páginas del siguiente nivel del árbol, el nivel más profundo del árbol, contiene los 44 bits superiores de la dirección física a la que se hace referencia, los 12 bits restantes provienen de los 12 bits menos significativos de la dirección virtual original. [1]

El hardware de paginación utiliza los 9 bits superiores de los 27 bits más significativos **(de los 39 utilizables)** para indexar un PTE en la tabla de páginas raíz, cuya dirección física es almacenada en el registro **satp**, los siguientes 9 para un PTE del nivel medio y los 9 inferiores para indexar un PTE del nivel final del árbol. [1]

Sistema en Sv48

En Sv48 RISC-V, una tabla de páginas tiene cuatro niveles y los bits 39 al 47 indexan las direcciones virtuales en el nivel más alto. [1]

UNIVERSIDAD NACIONAL DE COLOMBIA

Fallo de página

Si alguno de los tres PTE's necesarios para traducir una dirección no está presente, el hardware de paginación lanza una excepción de fallo de página, dejando que el kernel maneje la situación (se profundizara en el capítulo 4 del libro). [1]

El sistema propuesto en la figura 2, permite manejar direcciones virtuales con alta eficiencia de la memoria, en comparación con el sistema línea de la figura 1. En el caso de que grandes rangos de direcciones virtuales no tengan una asignación, la estructura de tres niveles puede omitir directorios de página completos.

Esto quiere decir que si una aplicación usa pocas páginas y estas empiezan desde la dirección cero, en el directorio raíz, las entradas de la 1 a la 511 del directorio de páginas de nivel superior no son válidas, **y el kernel no tiene que asignar directorios a sus páginas intermedias** y a su vez **no requieren de los directorios del nivel más bajo para cada uno de esos 511 directorios intermedios**, por lo que este sistema puede ahorrar 511 páginas intermedias y 511 * 512 páginas de directorios del nivel más profundo.

A diferencia del sistema lineal, donde dicha optimización no es posible. [1]

Desventajas del sistema de árbol

Cuando una CPU ejecuta una instrucción de carga o almacenamiento, es necesario que la CPU recorra toda la estructura, **y el sistema debe cargar tres entradas del sistema de árbol** desde la memoria para realizar la traducción de la dirección virtual para ejecutar la instrucción de carga/almacenamiento sobre una dirección física. Para evitar este costo adicional de cargar un PTE desde la memoria física, la CPU puede almacenar en caché las entradas de la tabla de páginas en un búfer, conocido como TLB (**T**ranslation **L**ook-aside **B**uffer). [1]

Sistema de banderas o indicadores

Cada PTE contiene unos bits indicadores, que definen el comportamiento permitido para el hardware sobre la dirección virtual asociada, estas banderas están en la figura 2 y se profundiza en la lista:

- **PTE_V:** Indica si el PTE está presente, en caso de que no lo esté, el hacer referencia a esta página provocará una excepción dado que no está permitido su acceso.
- **PTE_R:** Controla si las instrucciones pueden leer en la página.
- **PTE_W:** Controla si las instrucciones pueden escribir en la página.
- **PTE_X:** Controla si la CPU puede interpretar el contenido de la página como instrucciones a ejecutar.

- **PTE_U:** Controla si las instrucciones ejecutadas en modo usuario pueden acceder a la página, en caso de que este atributo no este configurado, el PTE solo puede ser empleado en modo supervisor (kernel).

Las banderas y las estructuras relacionadas se encuentran en: ***(kernel/riscv.h)*** [1] [2]

Uso de la tabla de páginas

Para que una CPU use una tabla de páginas, el kernel debe escribir la dirección física de la página que contiene la tabla de páginas raíz en el registro **satp**. Una CPU se encargará de traducir todas las direcciones generadas por instrucciones posteriores empleando la tabla a la que apunta su propio **satp**. [1]

Cada CPU tiene su propio satp, para que diferentes CPU puedan ejecutar diferentes procesos, cada uno con su espacio de direcciones privado, el cual es descrito por la tabla de páginas referenciada. [1]

Administración de memoria del kernel

El kernel **mapea toda la memoria física en su propia página de tablas**, esto le permite leer y escribir en cualquier ubicación de la memoria física empleando las instrucciones de carga y almacenamiento. Dado que tiene acceso a todos los directorios de páginas, el kernel puede alterar el contenido de un PTE de un directorio de páginas escribiendo en la dirección virtual del PTE mediante instrucciones de almacenamiento estándar. **En resumen, puede alterar el valor de un PTE escribiendo directamente sobre él. [1]**

UNIVERSIDAD NACIONAL DE COLOMBIA

Revisión de conceptos

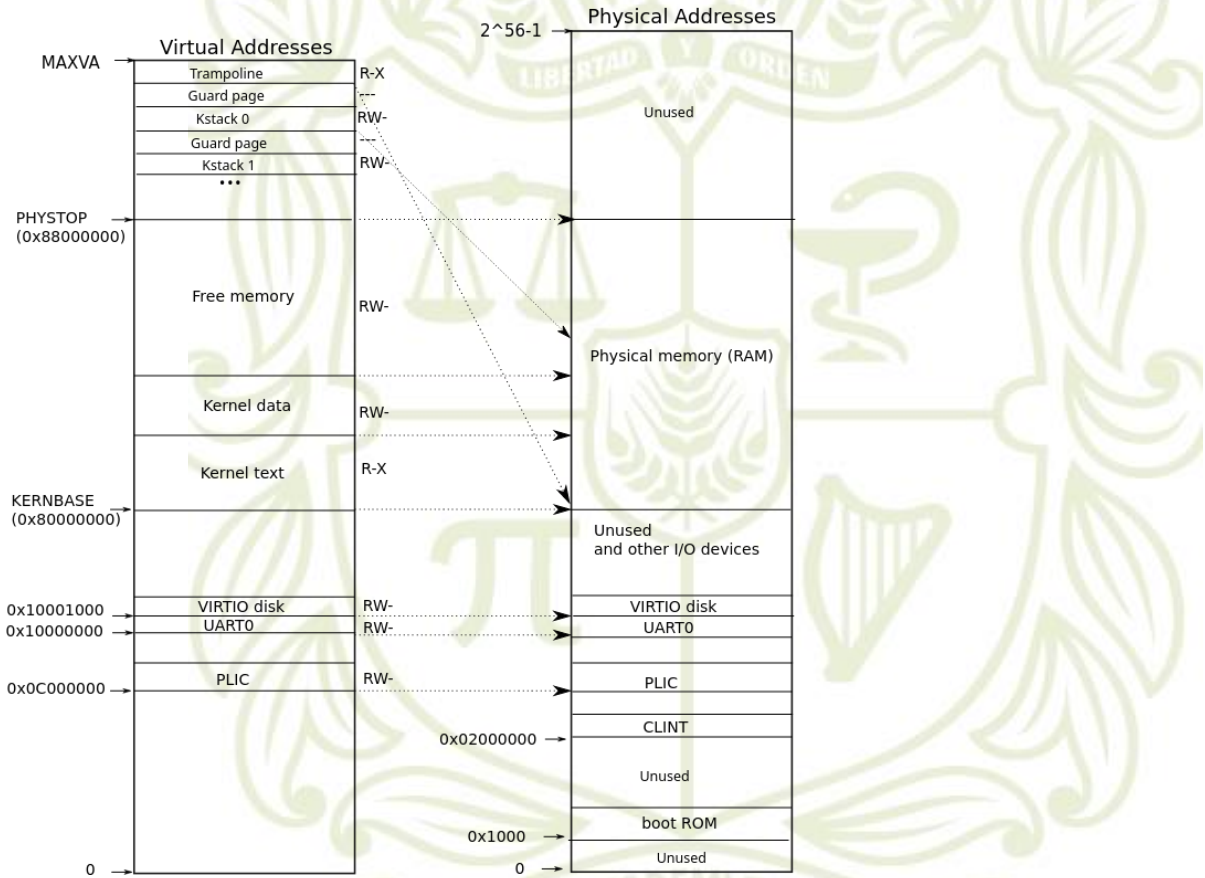
Cuando se habla de memoria física, se hace referencia a las celdas de almacenamiento en DRAM. Un byte de memoria física tiene una dirección, esta se llama dirección física. **Las instrucciones solo usan direcciones virtuales**, que es traducida a direcciones físicas por el hardware de paginación y **luego es enviado al hardware DRAM** para leer o escribir en la unidad de almacenamiento.

A diferencia de la memoria física y las direcciones virtuales, la memoria virtual no es un objeto físico, sino más bien, un **conjunto de abstracciones y mecanismos** que emplea el kernel para administrar memoria física y direcciones virtuales. [1]

UNIVERSIDAD NACIONAL DE COLOMBIA

Disposición de memoria en XV6

Figura: Izq: El espacio de direcciones del kernel, Der:Espacio de direcciones físicas de RISC-V [1]



Fuente: xv6: a simple, Unix-like teaching operating system [1]

Espacio de direcciones del kernel

XV6 emplea una tabla de páginas por proceso, esta tabla describe el espacio de direcciones de usuario de cada proceso, pero adicionalmente, implementa una tabla de una **única página** que describe el espacio de direcciones del kernel.

El kernel configura su propio espacio de direcciones para poder acceder a la memoria física y los recursos de hardware en direcciones virtuales predecibles, esta disposición se puede apreciar en la figura 3 y en el archivo ***(kernel/memlayout.h)*** [2], este último es donde se declaran las constantes para el diseño de la memoria del kernel de XV6. [1]

QEMU

QEMU simula una computadora que incluye RAM, la cual comienza en la dirección física **0x80000000** y continua hasta al menos la dirección **0x88000000**, esta última recibe el nombre de **PHYSTOP**. Adicionalmente, QEMU también simula dispositivos de E/S, **por ej. una interfaz de disco**. QEMU expone al software las interfaces de los dispositivos como registros de control asignados por debajo de la dirección física **0x80000000**.

El kernel puede interactuar con los dispositivos leyendo/escribiendo estas direcciones físicas especiales, las operaciones sobre estas direcciones se ejecutan directamente en el hardware del dispositivo en vez de la RAM. [1]

UNIVERSIDAD NACIONAL DE COLOMBIA

El kernel accede a la RAM y a los registros de dispositivos que son mapeados en memoria usando el mapeo directo; es decir, que **la dirección virtual es igual a la dirección física**. Un ejemplo evidente es la ubicación del kernel, este se encuentra en **KERNBASE** o **(0x80000000)**, esto tanto en la dirección virtual como en la memoria física. Esta implementación **simplifica el código del kernel** cuando es necesario leer o escribir en la memoria física. [1].

Sin embargo, este método no es aplicado por el kernel en su totalidad, existen direcciones virtuales las cuales no están asignadas directamente.

Asignaciones no directas del kernel

Estas direcciones virtuales no asignadas directamente son:

- **La página del trampolín:** Esta página es asignada en la parte superior del espacio de direcciones virtuales, al igual que las direcciones altas de las tablas de página de usuario. La página que contiene el código del trampolín, es mapeada dos veces en las direcciones virtuales, una mediante mapeo directo **(dado que esta se encuentra dentro del código del kernel)** y otra en las direcciones más altas de las direcciones virtuales. [1]

UNIVERSIDAD NACIONAL DE COLOMBIA

- **La página de la pila del kernel:** Cada proceso tiene su propia pila del kernel, esta se encuentra mapeada en direcciones altas y con una página de protección debajo de esta, la página de protección no tiene un PTE válido, es decir el indicador **PTE_V** no está establecido, por lo que si la pila se desborda, provocará que el kernel entre en pánico. **Es mejor generar un estado de pánico al kernel, que correr el riesgo de que el desbordamiento de pila pueda sobrescribir otras direcciones de memoria**. [1]

Aunque el kernel asigna sus pilas a las direcciones de memoria altas, estas también pueden ser accedidas por el kernel a través del mapeo directo.

Permisos asignados durante el mapeo

El kernel asigna a las páginas trampolín y el código del kernel los permisos de **PTE_R** y **PTE_X**. El kernel lee y ejecuta las instrucciones desde estas páginas. Mientras que las otras páginas son mapeadas con los permisos **PTE_R** y **PTE_W**, para que el kernel pueda leer y escribir en la memoria de las páginas mapeadas, **exceptuando las páginas de protección, las cuales no son páginas válidas**. [1]

Creando un espacio de direcciones

Gran parte del código para manipular espacio de direcciones y tablas de páginas se encuentra en ***(kernel/vm.c)***. Allí se define la estructura de datos principal **pagetable_t**, esta en realidad es un puntero a la raíz de una página la cual contiene una tabla de páginas de RISC-V, esta estructura puede representar una tabla de páginas del kernel o de un proceso.

Las funciones principales son ***walk (kernel/vm.c:86)***, el cual retorna la dirección del PTE correspondiente a una dirección virtual, y ***mappages (kernel/vm.c:143)*** el cual crea los PTE's para nuevos mapeos de una dirección física. [1] [2]

UNIVERSIDAD NACIONAL DE COLOMBIA

Convenciones y algunas funciones

Las funciones que comienzan con **kvm** manipulan la tabla de páginas del kernel, mientras que las funciones que comienzan con **uvm** manipulan una tabla de páginas de usuario, en otro caso, las funciones pueden ser empleadas para ambos casos.

copyout y ***copyin*** permiten copiar datos hacia y desde direcciones virtuales de usuario proporcionadas como argumentos de llamada al sistema, estas funciones están en el archivo ***(kernel/vm.c)*** porque necesitan traducir esas direcciones para encontrar la memoria física correspondiente. [1] [2]

Arranque del sistema de memoria

Al inicio del arranque del sistema, la función ***main*** llama a ***kvminit (kernel/vm:54)*** para crear la tabla de páginas del kernel mediante la función ***kvmmake (kernel/vm.c:20)***, esta llamada ocurre antes de que XV6 habilite la paginación en el RISC-V, por lo que las direcciones hacen referencia directamente a la memoria física.

kvmmake primero asigna una página en la memoria física, para almacenar la página raíz de la tabla de páginas. Luego llama a ***kvmmap*** para instalar las traducciones que el kernel necesita, traducciones como: los datos e instrucciones del kernel, la memoria física hasta **PHYSTOP** y los rangos de memoria que representan dispositivos. [1] [2]

UNIVERSIDAD NACIONAL DE COLOMBIA

proc_mapstacks (kernel/proc.c:33) asigna una pila de kernel para cada proceso, para esto, llama a ***kvmmap***, para mapear cada pila en la dirección virtual generada por ***KSTACK***, lo que deja espacio para las páginas de protección de pila no válidas. ‘

kvmmap llama a ***mappages*** para asignar en una tabla de páginas un rango de direcciones virtuales a un rango de direcciones físicas. Este proceso se realiza por separado para cada intervalo de página de las direcciones virtuales en el rango, para realizar el mapeo emplea ***walk***, con el fin de encontrar la dirección del PTE de dicha dirección, luego inicializa el PTE para contener el PPN y los permisos deseados (RXW) y este es marcado como válido (V) por defecto ***(kernel/vm.c:158)***. [1] [2]

UNIVERSIDAD NACIONAL DE COLOMBIA

Walk

walk (kernel/vm.c:86) imita el comportamiento del hardware de paginación de RISC-V mientras busca el PTE en una dirección virtual (Figura 2). La función recorre la tabla de páginas de tres niveles, empleando los bits de dirección virtual de cada nivel para encontrar el PTE de la tabla de páginas del siguiente nivel o de la página final ***(kernel/vm.c:92)***.

Si el PTE hallado no es válido, la página solicitada no se ha asignado, con base en sí el argumento alloc es distinto de cero, walk podrá asignar una nueva página en la tabla de páginas y ubicar allí la dirección física del PTE. La función devuelve la dirección del PTE en la capa mas profunda del árbol ***(kernel/vm.c:102)***. [1] [2]

UNIVERSIDAD NACIONAL DE COLOMBIA

El correcto funcionamiento del código anterior, depende de que la memoria física sea mapeada de forma directa en el espacio de direcciones virtuales del kernel.

Dado que a medida que walk desciende en el árbol, este extraerá la dirección **(física)** de la tabla de páginas del siguiente nivel del árbol ***(kernel/vm.c:94)***, y luego usará dicha dirección física como una **dirección virtual**, para buscar el PTE en el siguiente nivel ***(kernel/vm.c:92)***. [1] [2]

Instalación y caché

main llama a ***kvminithart (kernel/vm.c:62)*** para que instale la tabla de páginas del kernel, esto a través de, escribir la dirección física de la página raíz de la tabla de páginas en el registro **satp**. Después de esto, la CPU podrá traducir direcciones usando la tabla de páginas del kernel. [1] [2]
El sistema cuenta con paginación!!!

Cada CPU almacena en caché las entradas de la tabla de páginas en un búfer de búsqueda de traducción (TLB), pero cuando XV6 cambia una tabla de páginas, **se debe indicar a la CPU que invalide las entradas TLB almacenadas en caché**, esto debido a que, la TLB podría emplear un mapeo antiguo, que apunte a una página física que haya sido asignada a otro proceso, permitiendo que un proceso modifique la memoria de otro proceso. [1] [2]

UNIVERSIDAD NACIONAL DE COLOMBIA

RISC-V tiene la instrucción **SFENCE.VMA**, la cual vacía el TLB de la CPU actual. XV6 ejecuta *SFENCE.VMA* en **kvminithart** después de recargar el contenido del registro *satp* y en el código trampolín, que cambia a una tabla de páginas de usuario antes de volver al espacio de usuario ***(kernel/trampoline.S:89)***. [3]

También es necesario ejecutar **SFENCE.VMA** antes de cambiar el registro *satp*, para esperar que todas las instrucciones pendientes de carga y almacenamiento, esto garantiza que todas las actualizaciones pendientes se realicen sobre la tabla de páginas antigua y no sobre la nueva. [1] [2]

Flush Parcial en RISC-V

Para evitar vaciar el TLB completo, las CPU RISC-V permiten identificadores del espacio de direcciones (ASID) [1] [3]. Por lo que el kernel podría vaciar solo las entradas de TLB para un espacio de direcciones en particular.

XV6 no emplea esta característica

Asignación y seguimiento de memoria física

El kernel debe asignar y liberar memoria física en tiempo de ejecución para: las tablas de páginas, la memoria de usuario, las pilas del kernel y los búfers de las tuberías, etc. Para esto, XV6 emplea la memoria física entre el **KERNBASE** y el **PHYSTOP** (Figura 3) para realizar estas asignaciones en tiempo de ejecución. [1]

XV6 realiza un seguimiento de las páginas libres mediante el uso de una lista enlazada empleando las propias páginas libres. La asignación consiste en **eliminar una página de la lista enlazada**, liberar consiste en **agregar la página a la lista enlazada**. [1]

Referencias I

[1] Cox, Kaashoek, Morris. **xv6: a simple, Unix-like teaching operating system.** 2022. URL: https://github.com/mit-pdos/xv6-riscv-book.

[2] MIT PDOS. **xv6-riscv**. 25 de ago. de 2022. URL: https://github.com/mit-pdos/xv6-riscv.

[3] Andrew Waterman, Krste Asanovic y John Hauser, eds. **The RISC-V instruction set manual Volume II: privileged architecture**. https://github.com/riscv/riscv-isa-manual/releases/download/Priv-v1.12/riscv-privileged-20211203.pdf. 2021.

UNIVERSIDAD NACIONAL DE COLOMBIA

Gracias por la atención

Contacto:

csandovalc@unal.edu.co

UNIVERSIDAD NACIONAL DE COLOMBIA