72.08 Arquitecturas de las Computadoras

Florencia Petrikovich

Contenidos

Introducción	1
ASM y C	3
Arquitectura del procesador	4
Interrupciones	6
Integrados	9
Periféricos	10
Modo Protegido	11
Memoria Virtual	14
Memoria Caché	18
Procesadores de 64 bits	20

Introducción

Programas y binarios

- ➢ Compilación y linkedicion en C
  - gcc primero invoca al preprocesador cpp → el mismo toma el código fuente original y realiza las macro expansión de directivas (como #include y #define)
  - gcc toma la salida y convierte el código en C en código equivalente en ASM (gcc por default usa sintaxis AT & T en vez de Intel)
    - Genera un ".S"
  - Compilador GNU de ASM "gas" se encarga de generar el código objeto (osea el binario que es interpretado por el micro)
    - Extensión ".O"
  - El linkeditor crea el EJECUTABLE → puede ser formatos ELF, A.OUT, modelo flat
    - Linkeditor valida todas las referencias a funciones y después une todos los programas objetos que voy a necesitar
    - El ejecutable es el **ARCHIVO BINARIO (código + syscalls) + HEADER**
      - Header → propio del sistema operativo y es información para el SO para saber como ejecutarlo correctamente

Nota: De más rápido a más lento → CPU, Memoria cache, RAM, Disco. Rapidez de un procesador es la velocidad con la que lee una instrucción

- ➢ Ejecución de un programa
  - Línea de comando (bash)
  - SO (execve)
  - Lectura de Disco → SO lee byte por byte
  - Asignación de espacio en memoria → Lo sube a la RAM
    - Cuando hay un programa muy grande no lo pasa todo a la RAM, en tiempo de ejecución va subiendo lo relevante y sacando lo que no
  - CALL o JMP a dirección de memoria de programa
    - SO hace el jump y va a empezar a leer instrucciones desde acá
  - Ejecución del programa

  Nota: Los registros están dentro del microprocesador y nunca estan vacios

- ➢ Programa en memoria
  - **Código** → instrucciones del programa
    - Puntero a segmento de código → CS:IP
      - CS: Code segment → apunta al comienzo del segmento de código
      - IP: Instruction Pointer → relativo al inicio del segmento de código
        - Se desplaza dentro de esa zona de memoria
  - **Datos** → variables estáticas y globales que se inician al cargar el programa
    - Puntero a segmento de datos
      - DS: Data segment
  - **Heap** → memoria dinámica que se reserva y se libera en tiempo de ejecución
  - **Pila** → argumentos y variables LOCALES a la función
    - Puntero a segmento de pila → SS:SP
      - SP: Stack Pointer y SS: Stack Segment

- ➢ Introducción a SO
  - El SO es un programa que administra los recursos, etc. El mismo administra el user y kernel space.
  - División en memoria que se hace por seguridad:
    - **User space** → Donde corren los programas. Área "desprotegida"
      - Root es el único usuario que puede hablar con el SO
    - **Kernel space** → Donde corre el SO
      - Solo el kernel space puede modificar el SO
  - User space se puede comunicar con el kernel space mediante SYSTEM CALLS

- ➢ System calls → Formas de ejecutar un system call
  - Int 80h → interrupción nro 80, consume más tiempo
  - Instrucciones SYSCALL/SY SRET (intel) y SY SENTER/SY SEXIT (amd)
    - Disponibles desde Pentium II
    - La librería de C depende de la arquitectura
  - vsyscall y VDSCO → Syscall virtuales y Virtual Dynamic Shared Object
    - Linux crea páginas de memoria en user space para acelerar tiempos:

Procesadores y Lenguaje ASM (en Intel)

- ➢ Procesador mantuvo los registros anteriores al evolucionar
  - Ventajas: Reutilización de código, compatibilidad hacia atrás, vista comercial
  - Desventajas: Sigue teniendo lo viejo → ineficiente

- ➢ Arquitectura de 80386 → primer procesador de 32 bits de Intel
  - *Concepto de diseño superescalar* → consiste en agregar más de una unidad de procesamiento para aumentar la capacidad del mismo.
  - Cumple con:
    - **Multitarea**: Tener espacio de memoria individual para cada tarea y un espacio de memoria común para varias tareas.
    - **Multiusuario**: Más de un usuario tenga acceso a la CPU
    - **Tiempo compartido:** SO asigna un tiempo para cada tarea (time slot)
    - **Tiempo real:** Conmutación de tareas viene dada por acontecimientos externos
    - **Sistema de protección:** Mínimo 2 niveles → usuario y supervisor

- ➢ Modos del procesador
  - *Protegido* y *Real* (sin restricciones para acceder a la memoria)
    - Todos los procesadores empiezan en real y se pasan a protegido

Apuntes extras:

- Mouse, teclado, disco, RAM, ROM, placas, permisos administrados por el SO
- No existe el movimiento de datos de memoria a memoria en una sola instrucción
- **Bus**: Cables de cobre donde viaja la electricidad con distintos voltajes y carga la información
- Si usamos valores absolutos en los mov va haber poca PORTABILIDAD → trabajamos con posiciones relativas
- Hay UNA PILA POR PROGRAMA que esté corriendo
  - Solo una activa a la vez dado que solo hay un SP
- Linkeditar con ld produce un programa ejecutable de menor tamaño que hacerlo con gcc

ASM en Linux

IDT (Interrupt Descriptor Table)

- ➢ *Descriptor*: puntero que accede a una posición de memoria
- ➢ Se encuentra en la RAM
- ➢ Se le llama vectores de interrupciones
  - Los mismos se cargan en la IDT al encender la compu
- ➢ Micro la usa como referencias para correr rutinas
- ➢ UNIX uso la entrada int 80h como entrada al kernel
- ➢ Es GLOBAL para todos los procesos que están corriendo y se mantiene constante
- ➢ Nunca baja a disco

ASM y C

Manejo de Pila

- ➢ *PUSH* → decrementa ESP y guarda el valor en el stack
- ➢ *POP* → toma el contenido de la dirección e incrementa el ESP
- ➢ *EBP* → se utiliza para acceder a los parámetros/variables locales de la pila y de esta manera no se modifica el registro ESP. También mantiene una referencia de donde esta la dirección de retorno
- ➢ *Registros a preservar* → RBP, RSP, RBX, R12, R13, R15
- ➢ *Registros para pasar argumentos* → RDI, RSI, RDX, R10, R8, R9, pila
- ➢ *Instrucción call* → Guarda el IP para el retorno y ejecuta un JMP al primer byte de la función

Alineamiento de palabra → and esp, -16

- ➢ -16 es FFFF FFF0h
- ➢ Al usar el and con el ESP, el mismo va a cambiar a XXXX XXX0h donde X era el valor anterior del ESP.
  - Ventaja: datos alineados y no se tiene que cambiar de fila en el stack para obtener el dato completo. Más performante el acceso físico a memoria

Canary

- ➢ gcc llama a una función que ubica un dato llamado CANARY entre el EBP y las variables locales y el ret verifica que el canary no haya sido modificado.
  - Si fue modificado termina la ejecución
- ➢ Fue creado porque se creaban programas que pisaban la dirección de retorno con su código para que el programa salte a una posición de memoria con código malicioso que querían correr → EXPLOIT.
- ➢ Utiliza la función _stack_chk_fail

Apuntes extras:

- Cuando se hace int el que responde es el microprocesador (el que busca en la IDT)
- Hypervisor crea su propio IDT
- El micro se encarga de que un programa no trate de acceder a otro si el mismo no debería
- Si cuando hago ret, el ESP no está en la posición de retorno, toma el valor que encuentra y lo interpreta como una dirección de memoria. Va a la misma e interpreta lo que hay como la siguiente instrucción a correr.

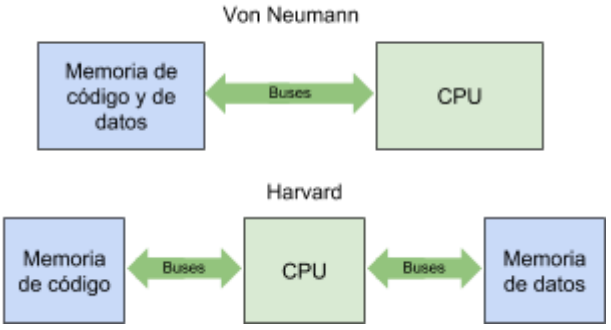
Arquitectura de procesador

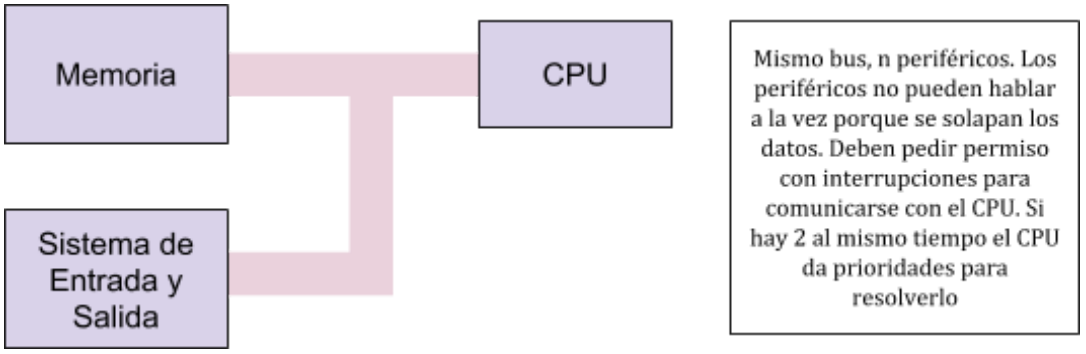
Transmisión digital

- Codificación en línea: Sender → Encoder → Decoder → Receiver
  - El encoder convierte data digital a eléctrica y el decoder pasa la eléctrica a digital
- Transmisión en serie: Un cable mandando 1 bit a la vez
- Transmisión paralela: muchos cables (BUS) y cada cable manda en paralelo 1 bit.
  - A misma velocidad de transmisión (⅔ de la velocidad de la luz), transmisión paralela es mas rapida que transmisión en serie.
- Transistores: Funcionan como represa para generar la transmisión de 1's y 0's
  - Compuerta por la cual circula la corriente y la compuerta es controlada por una llave que está controlada eléctricamente.
  - Se utilizan para amplificar señales

Tipos de Arquitectura

- Von Neumann
  - Primero busca la instrucción y luego la ejecuta
  - Desventaja: Más lenta
  - Ventaja: Más barata
- Harvard
  - Al mismo tiempo podemos leer las instrucciones y buscar/modificar los datos
  - Desventaja: Más cara
  - Ventaja: Más performante





CPU

- Unidad de control: Recupera instrucciones de memoria, las decodifica, escribe en memoria
- Unidad de ejecución: Lleva a cabo la ejecución de la instrucción
- Registros: Memoria interna utilizada como variable
- Flags: Indican eventos luego de ejecutar las instrucciones
- *Otros componentes*: Arithmetic logic unit, data cache, instrucción cache, bus unit

Nota: Primera instrucción que ejecuta el micro al prenderse → fabricante setea el valor de IP, donde la misma es la primera instrucción a ejecutarse (la cual está en la ROM). El valor del IP sale por el bus de direcciones y lo que se encuentre en esa posición se ejecuta.

Con n líneas de bus de address puedo ver 2^n posiciones de memoria

Memorias

Según la velocidad y la frecuencia con la que se quiere acceder a los datos es el repositorio donde se guardan los mismo.

- ➢ Clasificación
  - Por el **modo** en que se accede a los datos
  - Por las **operaciones** que aceptan
  - Por la **duración** de los datos
- ➢ ROM
  - PROM, EPROM, EEPROM, Flash
  - No volátil: Mantiene su información sin energía
  - Escritura más lenta que la RAM
- ➢ RAM
  - Volátil: pierde su información sin energía
  - DRAM
    - Necesita refresco de valores cada n milisegundos
    - Menos compleja, más económica
    - Más lenta
  - SRAM (caché)
    - No necesita refresco
    - Más compleja, más costosa
    - Más rápida

- ➢ Tiempo de Acceso: tiempo que le toma a una memoria RAM para completar un acceso después de otro. Se compone de
  - **Latencia**: Tiempo que tarda en devolver el valor la memoria
  - **Transferencia**: Cuanto tarda en leer/escribir el dato en el disco

- ➢ Memoria comercial
  - Vcc (+) y GND (-) → alimentación
  - G → read enable y W → write enable
  - Chip select → un apagado lógico

Decodificación

- ➢ El decodificador decide cuando se va a activar cada periférico, no el procesador
  - El procesador solo manda el puntero y el CD resuelve todo
- ➢ Mapa de memoria es la cantidad de memoria que el micro puede ver
- ➢ Más recomendado tener mayor cantidad de decos pequeños que tener uno grande

Mapa E/S

- ➢ Se agrego un mapa para los periféricos de entrada y salida porque no todo entraba en el mapa de memoria.
  - No es necesario si todo entra en el mapa de memoria
- ➢ Los buses de datos/address son los mismos para el mapa de memoria y E/S
- ➢ Cómo distinguir a cual se entra → **IO/M**
  - IO/M = 1 → Mapa E/S
    - Instrucciones IN, OUT

        - IO/M = 0 → Mapa de memoria
            - Instrucción mov
- ➢ Porque no simplemente se agregó una patita mas del bus de address?
    - Rompe retrocompatibilidad y la lógica de potencias de 2
    - No tengo registros existentes con dicho tamaño
- ➢ IO/M debe participar en el circuito decodificador → se le agrega la condición a todos los periféricos.

Sistema de Entrada y Salida

- ➢ Interrupción: una señal externa interrumpe al micro para requerir un servicio de atención
- ➢ E/S aislada: señal especial del micro indica la ejecución de una operación de E/S
- ➢ Acceso directo a memoria (***DMA***)
    - La información se transfiere directamente a la memoria, no requiere de intervención del CPU
        - Periférico recibe acceso directo
    - Usan los mismos buses de datos y address entonces el micro no puede hacer nada
    - Únicamente READ/WRITE a memoria
    - Deja que el disco rígido acceda directamente a la memoria
    - Ubicado entre el CPU y los periféricos

Apuntes extras

- **Drivers**: software que le permite a la computadora reconocer cómo funcionan los dispositivos externos.
- Placa de video en modo externo mapeada en B8000h
    - Cuando ya booteo el SO pasa a modo gráfico

Interrupciones

Importante: Nunca hay 2 códigos ejecutándose al mismo tiempo → Ejecución **ÚNICA**

- ➢ Detienen la ejecución del programa, corren la rutina de la interrupción adecuada, y vuelve a correr el programa principal.
- ➢ La rutina de atención se llama con un CALL y en la rutina se devuelve con IRET
- ➢ Patitas **INTR** y **NMI** para detener la ejecución del código

Interrupciones de Hardware

- ➢ Mediante INTR y NMI del micro
- ➢ Interrupciones enmascarables
    - Flag IF indica si se debe atender las interrupciones externas
        - IF = 1 → HABILITADO → STI
        - IF = 0 → DESHABILITADO → CLI
- ➢ Int NO enmascarables
    - Aquellas que ingresan por la patita NMI
    - Siempre ejecuta la rutina en la posición 2h de la IDT

Interrupciones de Software

- ➢ Mediante la instrucción INT

- ➢ Son programables → pueden suceder cuando el desarrollador quiera, mientras que las de hardware no se pueden determinar.
- ➢ Funcionamiento:
    1. Micro detiene la ejecución del programa
    2. Va a la IDT, busca la posición dicha y busca su puntero
    3. Corre esa rutina y al terminar, retorna y sigue ejecutando

PIC

- Integrado fuera del procesador (no lo creo Intel)
- Tiene cola de interrupciones, estado de interrupciones, etc.

Problema: INTR es solo 1 patita pero tenemos más de un dispositivo que quiere interrumpir al micro
Solución: PIC con 8 patitas IRQ en las cuales se puede conectar 1 dispositivo por IRQ

Funcionamiento

1. Disp genera una interrupción a traves de una patita IRQ
2. PIC alerta al micro a través de su patita INTR
3. El micro acepta la interrupción a través de su patita INTA
4. Al aceptar la interrupción, a través de BUS DE DATOS el PIC le dice al micro cuál de sus 8 patitas fue la que interrumpio
5. Sabiendo ese dato, el micro va a buscar en la tabla IDT el puntero que corresponde al dispositivo que debe ser atendido
6. Salta a la rutina de atención de interrupción una vez hallado el puntero

Nota: Como el vector de int. es el MISMO, lo puedo acceder a través de una interrupción de software o de hardware

IMR

- ➢ Los puertos de entrada y salida en la PC son el 20h y el 21h
- ➢ 20h se utiliza para programar el PIC
    - Lo utiliza el BIOS al arrancar el sistema
- ➢ 21h podemos acceder al registro **IMR** (interrupt mask register) donde podemos setear qué interrupciones llegan al micro y cuáles no

Acceso al PIC

```
in al, 21h          ; leo máscara del PIC
                    ; 0 = máscara deshabilitada, por lo tanto
                    ; pasa la señal al micro

mov al, 0FEh        ; ej: solo habilitó la int del teclado
out 21h, al
```

- ➢ **EOI** (End of Interrupt)
    - Cada rutina de atención de interrupción, luego de correr, debe avisar al PIC que terminó su ejecución
    - Se puede
        - Especificar la IRQ que terminó

        - Enviar un código que indica que finalizó la atención de la última interrupción que llegó
    - Esa palabra se envía al puerto 20h y el valor que se envía para indicar que finalizó la atención de la int es el valor 20h

➢ PIC en cascada
- Colocando 2 PICs en cascada se amplifica la cantidad de interrupciones de hardware
- Se utiliza el IRQ2 del Máster para conectar el Slave

Servicios de BIOS

➢ Al iniciar la PC, memoria ROM llamada BIOS ya viene cargada con rutinas para poder empezar a operar.

INT 10h	Rutinas de video (BIOS)
INT 13h	Rutinas de disco (BIOS)
INT 14h	Rutinas de puerto Serie (BIOS)
INT 19h	Rutina para **bootloader** (BIOS)
INT 1Ah	Rutinas para el RTC (BIOS)

➢ Cuando el SO se carga a memoria pisa estos punteros para poner sus propios drivers
➢ Rutina Bootloader
- Va al disco a buscar el SO
- Bootea el disco, puertos, etc
- Última que se ejecuta

Primero que hace la PC al prender? **POST** (power on self test)
Código de testing para verificar que ande bien el procesador. Pruebas de:
1. Registros
2. Operaciones
3. Buses

Primero que hace es buscar las extensiones de ROM. Realiza una búsqueda por los slots para buscar placas que posean memoria ROM o EPROM. Estas tienen la función de ampliar o sustituir las funciones del BIOS

Interrupciones en Multicore
Cada CPU tiene su PIC y afuera existe el APIC donde se conectan los periféricos y dicho PIC decide a que core le tira las interrupciones.

Timer Tick
- Integrado que genera una señal cada 55 milisegundos que interrumpe y llama a una rutina.
- Estándar → IRQ0
- Rutina de atención de interrupción → SCHEDULER
    - Interrumpe y le sa la ejecución durante 55 ms a otro programa (SLOT TIME)
    - Permite que tengamos la sensación de que varios programas están corriendo al mismo tiempo

- SO permite darle prioridad a ciertos programas dandole mas “tickets”

Excepciones

Similares a interrupciones en cuestión de procedimiento, pero no es un evento externo, es generado por el propio procesador que se interrumpe a sí mismo cuando encuentra alguna condición anormal.

- ➢ Primeros 32 vectores de la IDT son para las excepciones
  - La número 13 es SEGMENTATION FAULT
    - En lo que se basa el modo protegido
- ➢ Las excepciones en la IDT no las pisa el SO
- ➢ El SO no puede chequear las instrucciones porque mientras se ejecuta, el programa que está corriendo no es el SO (ejecución única!) → le enseña al micro que “está mal” y el micro se encarga de chequearlo.
  - El SO es el que carga las tablas y las tiene que cargar correctamente para garantizar la seguridad.
- ➢ **IDTR** → Registro que indica donde está la IDT
- ➢ Tipos
  - **Faults**: Excepción que puede corregirse. Dirección de la instrucción que produjo la excepción se guarda en la pila
  - **Trap**: Se utilizan para realizar accesos al SO
  - **Abort**: No siempre se puede obtener la instrucción que causó la excepción

Integrados

Familias Lógicas

Dos grandes familias de circuito integrados:

1. **TTL** → Transistor - Transistor Logic
2. **CMOS** → Complementary Metal-Oxide Semiconductor

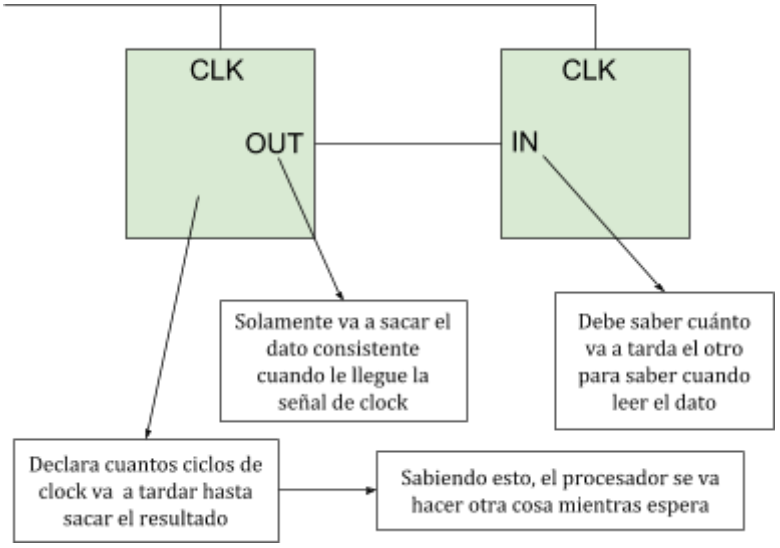
No se pueden combinar porque tienen distintos tipos de voltaje, tensión, velocidad de conmutación y tiempo de propagación

- ➢ Velocidad de conmutación: Velocidad en la cual se realiza el cambio de estado en la salida y entrada de una compuerta
  - Cuando dos procesadores se comunican deben tener la misma velocidad de conmutación o sino hay problemas
- ➢ Tiempo de propagación: Cuánto tiempo tarda en producirse el resultado
  - Mientras más complejo, más va a tardar en refrescarse la salida dado una entrada
  - Problema: Momentos de inconsistencia en la salida (ejemplo el tiempo que tardó para que el AND reciba el segundo resultado al haber recibido el primero)
  - Solución: CLOCK

Clock

- ➢ Se tiene un Generador de Clock y a través de CLK Out manda la señal del clock
  - Va conectado a TODOS los periféricos
  - Marca el ritmo en el que trabajan todos los integrados de la PC
  - Se mide en Hz → ciclos por segundo

- ➢ Se logra que no haya tiempos de inconsistencia desde lo que se puso en la entrada hasta que se propaga a la salida.
- ➢ Dos tipos de <u>flancos</u>: Ascendente y Descendente
- ➢ <u>Habilitación</u>
    - Los integrados suelen tener una entrada (EN) la cual al activarla, es posible utilizar el dispositivo
        - Integrados pueden tener más de una de estas entradas



Periféricos

PIT

- ➢ Chip que puede ser empleado como reloj de tiempo real, contador de sucesos, generador de ritmo programable, etc.

40h	Counter 1 (conectado al IRQ0)
41h	Counter 2
42h	Counter 3
43h	Registro de control para programar al integrado

Counter 1 es el TIMER TICK → señal cada 55 ms (lo hace cada vez que finaliza una cuenta de 65536)

RTC

- ➢ Cuenta horas, minutos, segundos.
- ➢ Sabe la fecha, si tengo conectados discos duros a la compu, etc.
- ➢ Único integrado que tiene pila

Para acceder se utiliza el modo de acceso INDIRECTO → evitar consumir direcciones de E/S

READ:
```
mov al, memory_location
out 70h, al
in al, 71h
```

WRITE:
```
mov al, memory_location
out 70h, al
mov al, new_contents
out 71h, al
```

Teclado

Interrupciones:

- ➢ MAKE CODE → al presionar una tecla
- ➢ BREAK CODE → al liberar una tecla

Leer el scan code en el driver → IN al, 60h

> Nota: El driver de teclado recibe siempre siempre los mismos códigos y lo que se muestra en pantalla depende del driver instalado. Puerto 60h de E/S (ESTÁNDAR) esta el teclado (IRQ1)

Disco

- ➢ Mini imanes girados para un lado representan un 0 y para el otro lado representan un 1
- ➢ Más de un disco (es uno encima del otro)
- ➢ Se leen datos de arriba y de abajo
- ➢ Archivos se guardan en forma circular (track) dado que tarda más en mover la cabeza

Tablas de particiones

- ➢ **MBR** → Master Boot Record (512 bytes)
  - Tabla de particiones → info de como tenemos particionado el disco
  - Boot code → código de nuestro SO
  - Rutina 19 que levanta el SO va a buscar el MBR
- ➢ **VBR** → Volume Boot Record
  - Los 512 bytes de cada partición que se asemeja al MBR

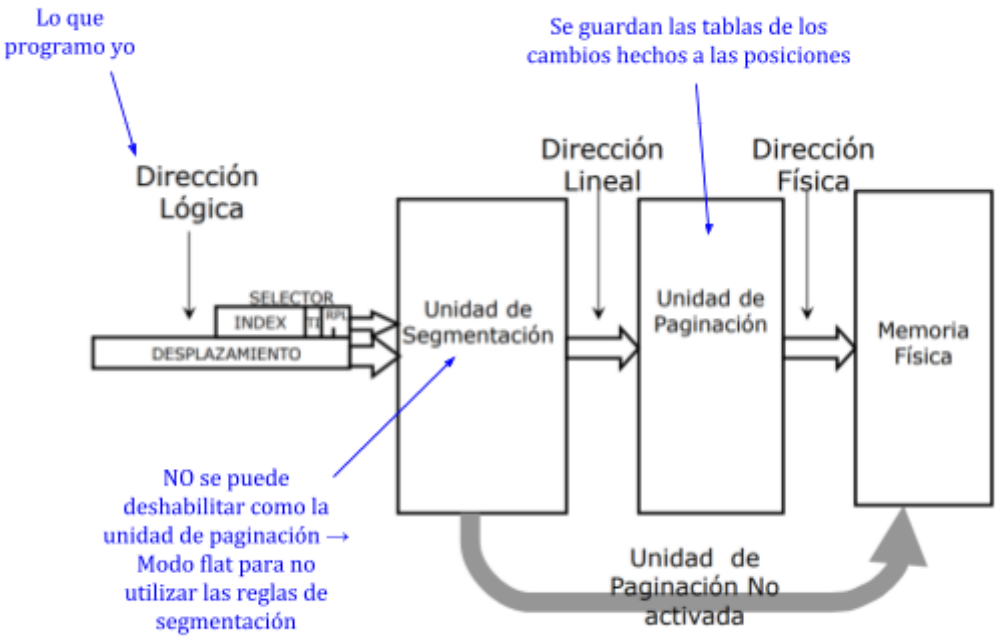
Modo Protegido

Conmutación de una tarea

- ➢ Cuando el timer tick para un programa, para que luego pueda seguir ejecutando el mismo necesita una "FOTO" del estado de todo
  - FOTO = Contexto de la tarea → Guardar todo el contexto de la tarea en la RAM
    - Scheduler es el que "backupea" la tarea actual y busca en la RAM la que sigue
- ➢ SO es una TAREA → Timer tick decide **cuando corre**:
  - Excepciones
  - Interrupciones o syscalls
  - Cuando no hay nada mas corriendo
- ➢ **Virtualización de memoria** → cuando algunos contextos bajan a disco porque no hay más espacio en la RAM
- ➢ Si una tarea trata de cambiar el contexto de otra → EXCEPCIÓN
  - Procesador hallo esta anomalía porque el SO no está corriendo

Memory Management Unit

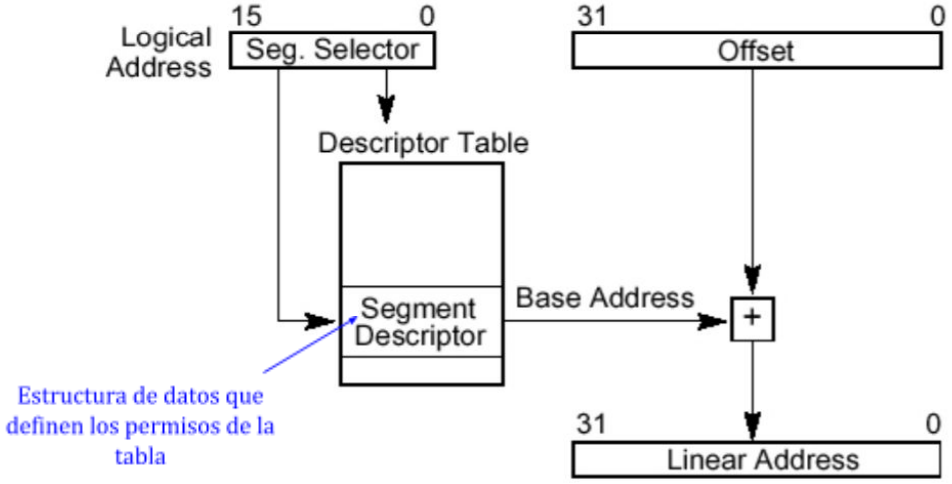
- ➢ Dirección lógica
  - SELECTOR → selecciona el segmento en el que se va a trabajar
    - Registro de segmento (CS, DS, ES, SS, FS, GS)
    - Contiene un bit para TI y 2 para RPL → para la tabla de permisos
  - DESPLAZAMIENTO dentro del segmento
  - Ejemplo → mov DS:[1000h], EAX



➢ Unidad de paginación divide la memoria en bloques de igual tamaño

- Si son muy GRANDES pierdo mucha memoria
- Si son muy CHICOS le tengo que dar muchas páginas a un programa y tener registros de cuales son → menos performante

➢ Dirección lógica a lineal

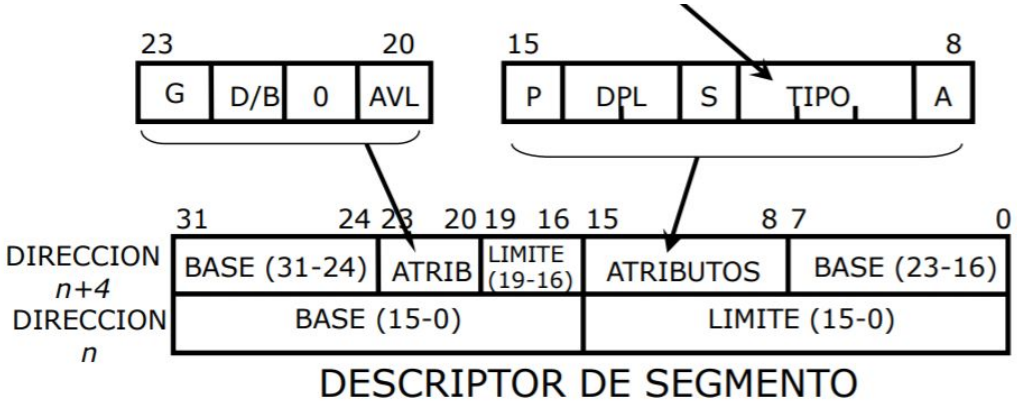


➢ Existen dos tablas de descriptores

1. **GDT** (Global Descriptor Table)
   a. TI = 0
   b. Apuntada por el registro GDTR
   c. Primer descriptor no se utiliza
2. **LDT** (Local Descriptor Table)
   a. TI = 1
   b. Apuntada por el registro LDTR

Descriptores

Descriptor de Segmento



- **P** (Bit de presencia)
  - P = 1 el segmento está cargado en la memoria física
  - P = 0 el segmento está ausente

Nota: Hay que tener un equilibro porque no es bueno tener muchos segmentos con P = 0 porque se tiene que indicar donde están todos esos.

- **DPL** (Nivel de privilegio)
  - Indica el nivel de privilegio del segmento
  - 0 = máximo privilegio (Kernel), 3 = mínimo (User)
- **S** (Tipo de segmento)
  - S = 1 → segmento normal (código, datos, pila)
  - S = 0 → segmento de sistema (puerta de llamada, TSS, etc)
- **Tipo**: Campo de 3 bits → en el caso de segmento normal
  - **E** (Ejecutable) = 1 → segmento ejecutable
    - **C** (Ajustable o conforming) = 1 define si el segmento al ser accedido cambia su nivel de privilegio a aquel del segmento que lo llamó
    - **R** (Readable) = 1 → si el segmento de código puede ser leído
  - **E** = 0 → segmento de datos
    - **ED** (Expansión decreciente) = 0 → segmento de datos normal, ED = 1 se trata de segmento de pila
    - **W** (Writable) = 1 → el segmento puede escribir, W = 0 solo puede leer

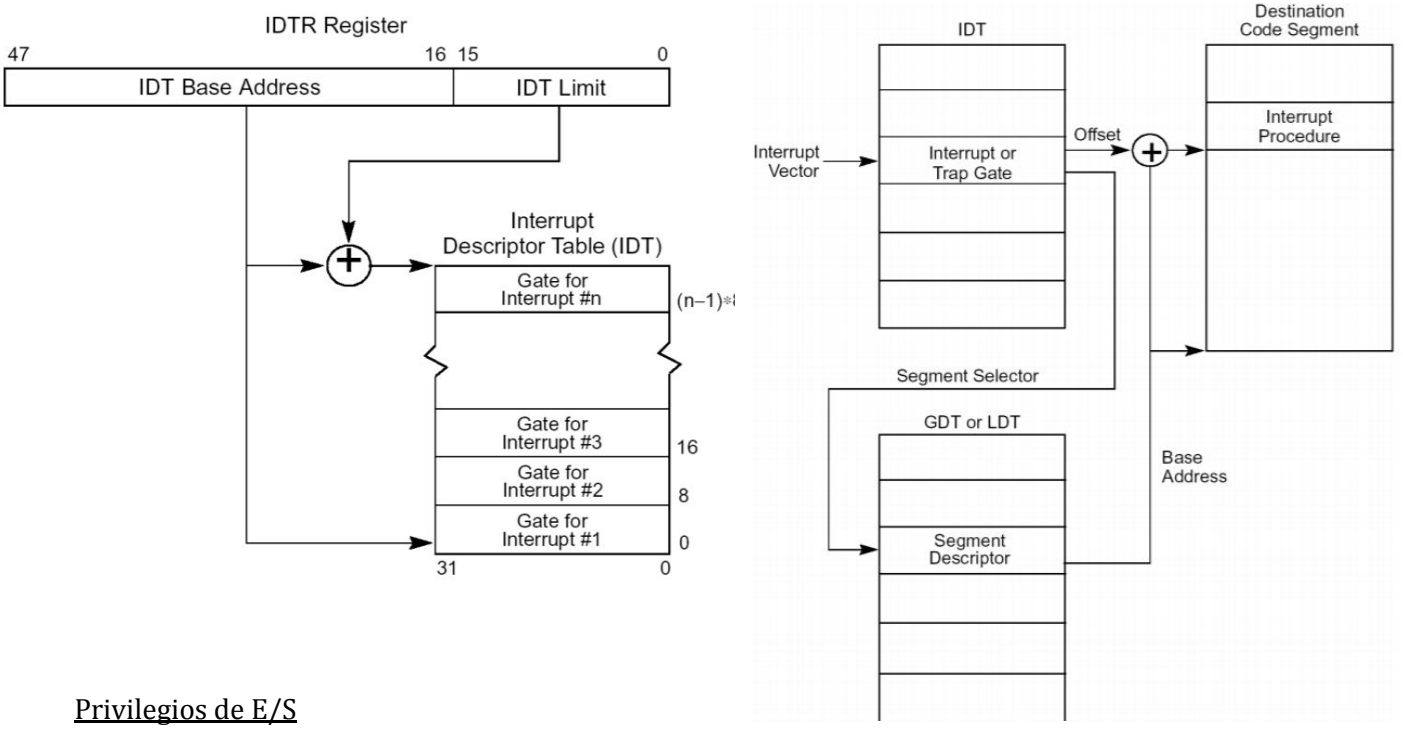
Descriptor de Sistema

Los más usuales:

1. Trap Gate (Puerta de excepción)
2. Interrupt Gate (Puerta de interrupción)
3. Task Gate
   a. El procesador de Intel nos da una herramienta para guardar el contexto de ejecución. Se denomina TSS (Task State Segment)

Procedimiento: Los descriptores de sistema tienen los segment selector. Al encontrar el Gate que se desee, el segment selector nos dice que posición de la GDT queremos acceder. Se accede al descriptor del segmeto y se chequean los permisos. Si todo coincide y es válido, el base address del descriptor de segmento se une al offset del descriptor de sistema y con eso se crea la dirección

LINEAL. Si la unidad de paginación está apagada, esa dirección es la misma que la física y se va a buscar la rutina de atención deseada.



Privilegios de E/S

El procesador provee dos mecanismos

1. Campo de 2 bits **IOPL** en los EFLAGS → Especifica el mínimo nivel de privilegio que se debe tener para poder usar las instrucciones de Entrada/Salida
2. Mapa de bits de E/S en el TSS

Nota: Cuando se guarda una tarea en la RAM se actualiza la tabla de permisos (la cual también está en la RAM). Chmod modifica la tabla de permisos de archivos.

Ejemplo: T1 se encuentra en las posiciones 1000h a 2000h y T2 tiene la instrucción

mov AX, [1023 h]

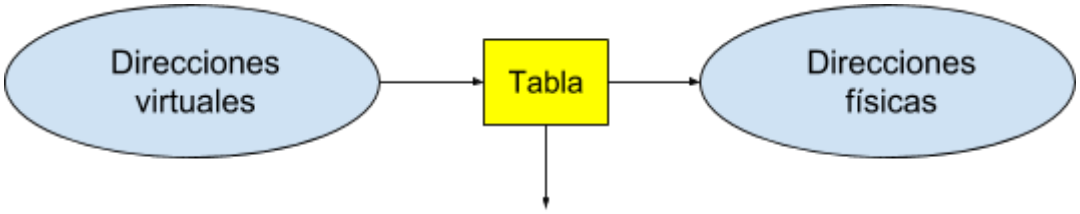
Cada vez que se hace un mov a memoria el procesador se fija en la tabla de permisos que fue hecha por el SO si está permitido → Si **NO** lo está, se corre una rutina de excepción que fue escrita por el SO. El SO es llamado y el mismo reacciona.

Memoria Virtual

Problemática: La cantidad de memoria virtual es distinta en algunos caso a la cantidad de memoria física. SO tiene que arreglarse con las direcciones para correr procesos en PCs con distintos tamaños de memoria. SO se tiene que fijar si un proceso va a entrar en la memoria física y si no, lo tiene que mandar a un medio alternativo (ej disco)

Solución: Indirección

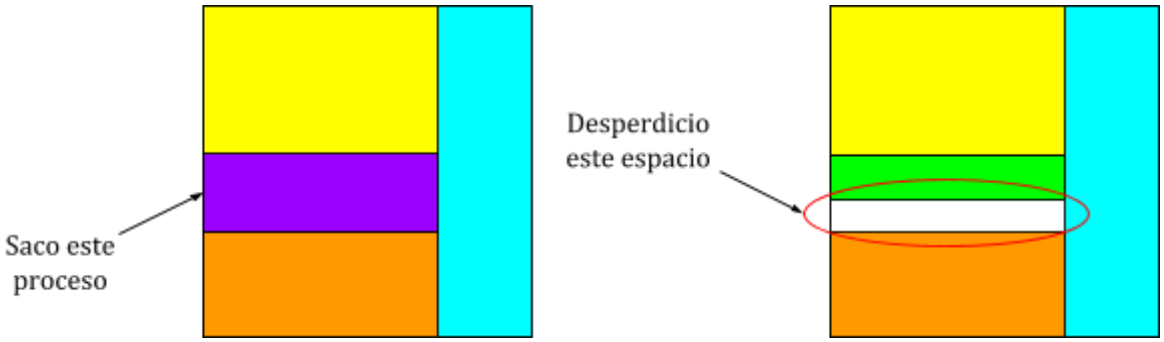
SO con ayuda del micro le da a los procesos un conjunto de direcciones virtuales. CADA PROCESO ES DUEÑO DE SU ESPACIO VIRTUAL! Las direcciones luego se traducen mediante tablas a las reales.



Traducción de direcciones
Ordenadas y administradas por el SO. Van a asignar el espacio REAL en memoria física donde va ir cado uno de los procesos. La tabla se guarda en memoria.

MV para múltiples procesos

- **Páginas**: Bloques de memoria de tamaño constante. Procesador las va a asignar para dividir la memoria en partes iguales
- Porque se utilizan las páginas?

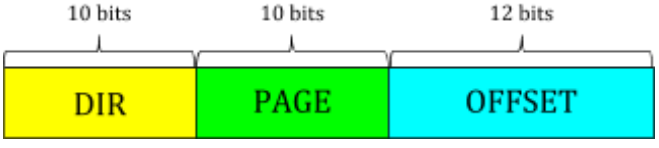


- Proceso no puede ocupar menos de 1 página pero puede ocupar hasta n páginas
- El que maneja las páginas es el procesador NO el SO!
- Tamaño de las páginas
  - Si las paginas son muy **grandes** pierdo espacio y tengo pocos procesos posibles
  - Si son muy **chicas** la tabla del mapping va a ocupar mucho espacio al se muy grande
  - Que tamaño entonces? Intel → 4k
  - El mismo se puede definir → cuando el SO bootea se decide el tamaño de las mismas
- Cantidad de páginas
  - Cantidad de memoria VIRTUAL divido tamaño de páginas
- Si se acaban las páginas en la memoria física → empiezo a mandar algunas a DISCO
  - Los procesos que se mandan a disco NO corren
- SO ubica la misma info en la misma página física (ejemplo librería estándar de C)
  - Proceso cree que tiene una página única

Nota: Todas las páginas tienen dirección virtual pero no todas tienen una página física (las de disco no la tienen)

Paginación en Intel (32 bits)

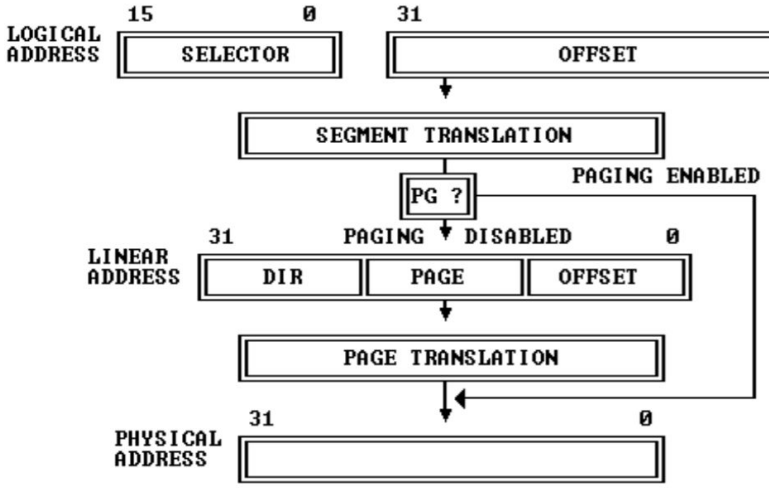
- *Dirección lógica* → SELECTOR + OFFSET
  - Pasa por la tabla de descriptores y obtenemos *dirección lineal*
    - La lineal entra en la unidad de paginación donde es interpretada como:



DIR	PAGE	OFFSET
Sirve como índice para ir a buscar a la tabla *directorio de páginas* una entrada. La entrada me lleva a una tabla de páginas, una estructura de datos que me da info de las páginas que se encuentran en memoria física	Entrada que tengo que tomar de la *tabla de páginas* la cual fue referenciada por la entrada DIR. La entrada me referencia a una página de 4k.	Para entrar a cualquier byte dentro de la *página*. 12 bits → $2^{12} = 4k =$ tamaño de la página. El offset nunca es alterado en la paginación.

Los 20 bits de DIR y PAGE coinciden con la cantidad de páginas que hay. Se hizo la división porque era complicado trabajar con 20 bits a la vez y de esta manera se puede trabajar con grupos de páginas (ej mandar un grupo de páginas a disco)

Nota: Unidad de paginación se puede DESHABILITAR con el bit PG dentro del registro **CRO**



Nota: La dirección lineal al pasar el módulo de paginación se convierte en dirección física. El registro **CR3** de 32 bits indica la ubicación de Directorio de páginas

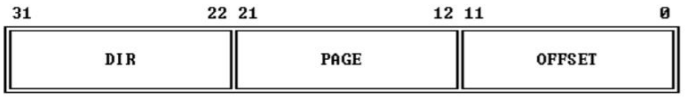
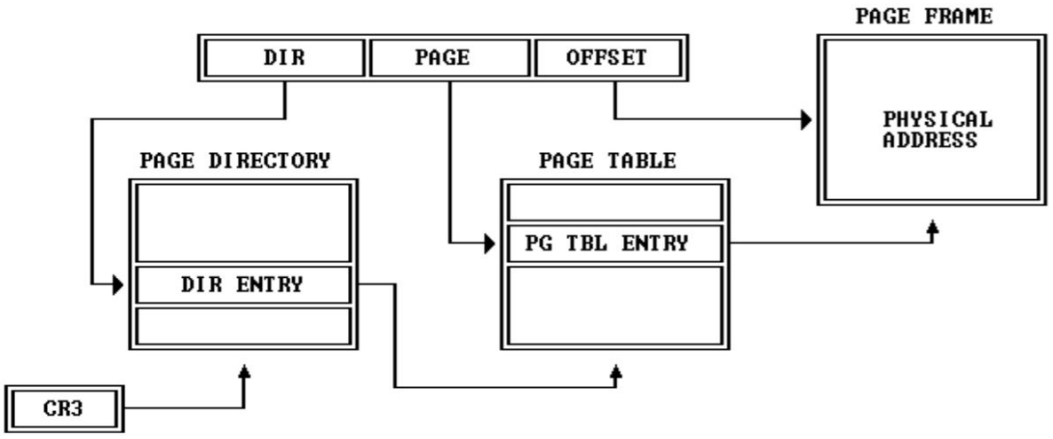
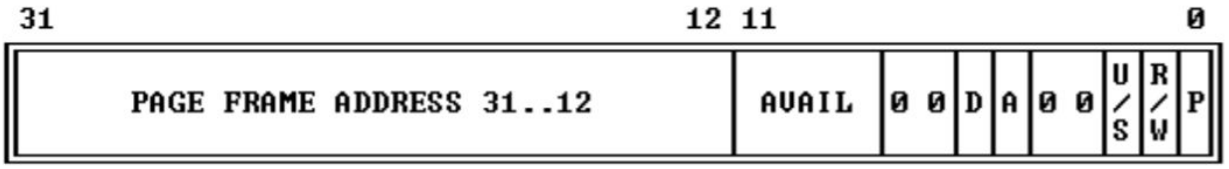


Figure 5-9. Page Translation



Estructura de las entradas

- El directorio de páginas puede referenciar a 1024 (2^{10}) Tablas de Páginas.
- Cada una de estas 1024 tablas de páginas puede referenciar a 1024 Páginas en memoria física.
- Los elementos de la tabla tienen el MISMO formato



Page Frame Address → 20 bits porque hay 2^{20} posibles páginas. Me dice donde empieza cada una.
Privilegios

1. **AVAIL** → Available for systems programmer use
2. **D** → Dirty
3. **U/S** → User/Supervisor
    a. Divide la páginas en 2 niveles de privilegios (Usuario y Supervisor)
    b. Si no se respetan los accesos se produce una excepción. Si el procesador está ejecutando en niveles 0, 1 y 2 es Supervisor, 3 es Usuario.
4. **R/W** → Read/Write
5. **P** → Present
    a. P = 1 → Esa tabla se puede utilizar para obtener una dirección de memoria
    b. P = 0 → No se puede utilizar
        i. Al encontrar P = 0, el **procesador genera una excepción**. Los sistemas operativos que tiene soporte de memoria virtual utilizan esta excepción para *disparar una rutina que lleve a memoria la página faltante* (que seguramente se encuentra en disco).

Nota: En una entrada de DIR puedo agrupar privilegios de 1024 páginas. En una entrada de tabla de páginas puedo poner privilegios distintos para una página específica.

Nota: Cuando se manda algo a disco no se borra la entrada de la página en la tabla de páginas. Va a ver una entrada idéntica pero una con el P = 0 y otra con P = 1 (porque son las misma páginas FÍSICAS)

Protección combinada

- Cuando la paginación está HABILITADA, el procesador primero chequea los permisos en la segmentación y luego en la paginación.
- Por ejemplo se puede definir un segmento de datos, que está compuesto por páginas donde algunas son de escritura y otras son de lectura.
- Debido a este manejo de memoria, algunos SO (como UNIX) eligen utilizar el modelo flat de memoria

Modelo de memoria Flat

- Unix declara dos segmentos de código y de datos que referencian a toda la memoria
    - Esto le permite un manejo más simple de la memoria y a su vez más portabilidad ya que muchos procesadores no tienen unidad de segmentación pero si de paginación.

Linux

- ➢ Hasta la versión 2.2 de Kernel, Linux utilizaba los TSS para realizar conmutación de tareas
- ➢ A partir de la versión 2.4, realiza la conmutación de tareas mediante funciones
- ➢ Linux utiliza la paginación de hardware
- ➢ En microprocesadores de 64 bits, agrega un nivel extra de páginas (en el medio de las de hardware) llamado "middle directory"

Paging and swapping

1. Cuando se necesita una página en memoria y la memoria está completa, Linux elige una página que no ha sido accedida últimamente y realiza un "page out" que consiste en guardar esa página en disco.
    a. Generalmente en una zona especial destinada para ello, puede ser una partición o un archivo en el filesystem.
2. Luego setea en la página "vieja" el bit de Presencia en 0, y guarda en su índice la dirección donde la debe encontrar en el disco rígido.
    a. A este concepto se lo denomina "paging"
3. El concepto de swapping se refiere a guardar en disco TODAS las páginas de un proceso
    a. En Linux, existe un thread llamado "kswapd" que se encarga de esto

Nota: TLB → Buffer de traducción adelantada. Es una memoria caché asociativa que guarda los marcos de páginas donde están almacenadas las páginas más recientemente referenciadas. Su función es la de reducir el tiempo de búsqueda de la dirección real a partir de una dirección virtual, ahorrando el acceso a la tabla de páginas.

Memoria Caché

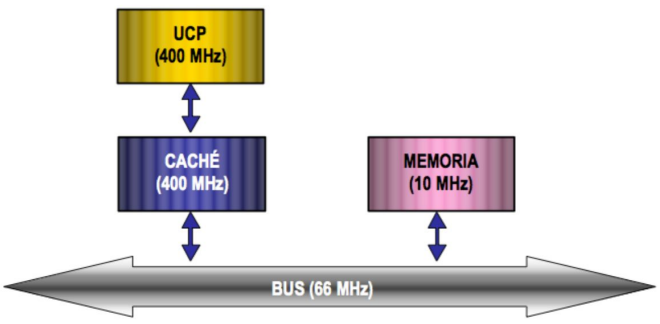
Definición: Área de memoria rápida (SRAM) ubicada dentro del procesador. El caché inteligente Intel se refiere a la arquitectura que permite a todos los núcleos compartir dinámicamente el acceso al caché de alto nivel. Es transparente al programador y el procesador.

Por qué esta?

- ➢ Para cada instrucción que se ejecuta, el microprocesador debe ir a buscarla a memoria y traerla de vuelta (todo mediante el bus de datos)
    - ○ Ir a buscar a memoria es más lento que trabajar dentro del procesador
- ➢ Si el código es un ciclo con n iteraciones, debe ir a buscar las mismas instrucciones n veces?
    - ○ **NO!** → guardamos el bloque en la caché DENTRO del procesador

Nota: La memoria caché ve a la memoria como bloques. Estos son de tamaño **FIJO** y no hay inteligencia de la memoria caché sobre paginación. Bloques pueden ser partes de distintas páginas.

- ➢ Velocidad de respuesta acorde al CPU
- ➢ Guarda copias de la DRAM



Si la caché **no** estuviera habría diferencias de velocidades:

- ➢ Cuello de botella
- ➢ Pérdida de performance

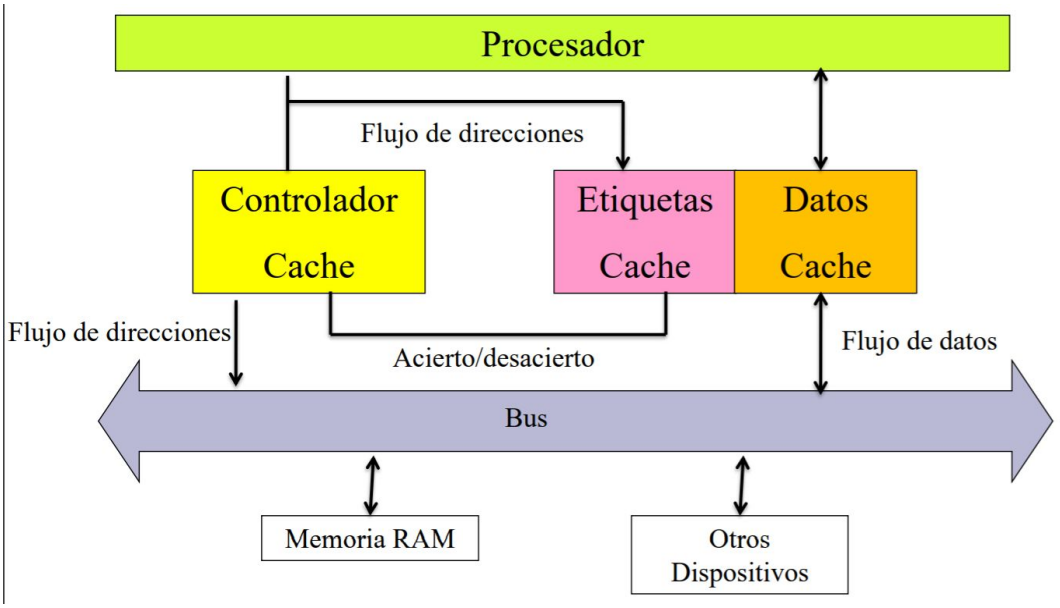
Estructura

La caché se compone de

1. Memoria de datos
    a. El bloque que se guarda de la DRAM
2. Memoria de etiquetas
    a. Dice de donde saque el bloque que se guardó
3. Controlador
    a. Selecciona cuantos y cuales bytes se copian a la memoria de datos.

Nota: Para calcular la cantidad de bloques en la caché → cantidad de memoria caché dividido por el tamaño del bloque. Cantidad de etiquetas → cantidad de memoria física / tamaño del bloque.

Conexión en serie



Funcionamiento: Controlador caché se fija en las etiquetas a ver si lo 15 bits (en este caso) más significativos matchean. En caso de que lo hagan → ACIERTO, el bloque buscado está en la memoria caché entonces lo va a buscar a la memoria caché. En caso de que no matchean, va a memoria trae el bloque deseado a la caché mediante el bus de datos.

Mapeos

- ➢ Mapeo Directo
    - Un bloque de memoria solo se puede mapear a una única ranura en el caché
    - Sencilla, poco utilizada
- ➢ Mapeo Asociativo
    - Un bloque de memoria se puede mapear a cualquier ranura del caché
    - Compleja, más utilizada

Politicas de sustitucion

Se actualiza la caché al haber fallo o ausencia de palabra buscada

- ➢ First In First Out (FIFO)
- ➢ Least Recently Used (LRU)
    - Se necesita un flag que indique cuántas veces se utilizó el bloque
- ➢ Random

Actualización de RAM

- ➢ Escritura inmediata (write through)
  - Se actualizan ambas memorias juntas
  - Más lento
  - Más económico
- ➢ Escritura obligada (write back)
  - Solo se actualiza lo estrictamente necesario
  - Más rápido
  - Menos económica
  - Problemas con dispositivos DMA
  - Problemas con Multi-cores
  - Más utilizada

Nota: Cuando se escribe en un bloque en caché, el que está en memoria se marca como DIRTY y no se puede utilizar porque está desactualizado. Se actualiza (pisa) cuando se baja el bloque a memoria.

Procesadores de 64 bits

- ➢ **IA-32**: Tecnología Intel de 32 bits
- ➢ **Intel 64/EM64T**: Extensión Intel de 64 bits
  - Compatible con 32 bits
- ➢ **AMD64**: Tecnología AMD de extensión 64 bits
  - Compatible con 32 bits
- ➢ **IA-64**: Tecnología Intel de 64 bits (Itanium)
  - NO compatible con 32 bits
  - Fracaso en el mercado

Definiciones:

- ➢ **Arquitectura**: Definido por el set de instrucciones y por lo tanto define la compatibilidad
- ➢ **Microarquitectura**: Define cómo se implementa el hardware dado una arquitectura
- ➢ **Procesadores**: Implementación comercial de la microarquitectura

Set de instrucciones

- ➢ IA-32: Para los Pentium, Celeron, Core Duo, Xeon y Core 2 Duo
- ➢ Intel Xscale: Para los tipo ARM
- ➢ IA-64: Es el ser de instrucciones para los Itanium (high end)

Micro-Arquitecturas

Nuevas tecnologías:

- ➢ Multi-núcleos
- ➢ Consumo de energía
- ➢ Virtualización
- ➢ Caché
- ➢ FSB
- ➢ Pipelines

Micro-Arquitecturas y Procesadores

- IA-32
  - P5
    - Pentium
  - P6
    - Pentium Pro, Pentium II, Celeron, Pentium III
  - NetBurst
    - Pentium IV, Xeon, Pentium IV HT, Pentium M
  - Core
    - Core2Duo, Core2Quad Xeon, Quad Core
  - Nehalem
    - Core i7

Pentium

- Tiene 64 bits de bus de datos
- 32 bits de bus de direcciones
- Arquitectura Superescalar
  - Tiene 3 unidades de ejecución
    - Punto flotante
    - U-pipe
    - V-pipe
  - Podría ejecutar al mismo tiempo (porque son instrucciones independiente):
    - FAD ST, ST(2)
    - MOV EAX, 12h
    - MOV EBX, 13h
- Paginación
  - Puede manejar dos niveles
  - Página de 4 MB
  - Se setea con el bit PSE en el CR0

Pentium Pro

- 36 lineas de bus de direcciones (64G)
- 64 líneas de bus de datos
- Para acceder a +4GM utiliza PAE (Physical Address Extension)
- Las tareas siguen viendo un máximo de 4GB
- El sistema operativo tiene que proveer de un manejo de memoria para lograr utilizar los 64 Gb de direcciones físicas.

Physical Address Extension

- Se habilita con el bit 5 del CR4 (PAE)
- El micro llega a 64 GB de memoria física
  - Pero las direcciones virtuales siguen siendo de 32 bits
- Se agregan en las entradas del directorio y las tablas de páginas 24 bits para direccionar las páginas
  - 16 M de paginas
  - 16 M * 4 KB = 64 GB

Intel 64 / EM64T

- Extended Memory 64 Technology
- Compite con AMD64
- Direcciones virtuales de 64 bits
- Direcciona 16 EB (exabytes)
- Bus de direcciones puede llegar a 40 líneas (1 TB)
- Registros de 64 bits (RAX, RCX, RIP)
- Instrucción de direccionamiento a memoria de 64 bits
- ALU de 64 bits
- Esta tecnología introduce un nuevo modo de funcionamiento: IA-32e, que tiene dos sub-modos:
    - **Modo compatibilidad con 32 bits** (Legacy Mode)
        - Similar a modo protegido de 32 bits
        - Para acceder +4GB usa PAE (Physical Address Extension)
    - **Modo 64 bits** (Long Mode)
        - Utiliza direcciones de lógicas de 64 bits
        - Los operandos por default son de 32 bits (salvo que tengan prefijo REX)