Apuntes de Arquitectura de Sistemas

Daniel Monjas Miguélez

March 7, 2022

Contents

1 Tema 1: Soporte Hardware

1.1 Motivación

matriz

```
int matriz[10000][10000];
```

inicialización a cero: bien

```
for (unsigned i = 0; i < 10000; ++i)
  for (unsigned j = 0; j < 10000; ++j)
    matriz[i][j] = 0;
```

inicialización a cero: mal

```
for (unsigned i = 0; i < 10000; ++i)
  for (unsigned j = 0; j < 10000; ++j)
    matriz[j][i] = 0;
```

Para una matriz de tamaño 10000×10000:

	matriz[i][j]		matriz[j][i]	
Mac OS X	user	0.45	user	17.413
1.25GHz PPC G4	system	1.15	system	2.037
512MB RAM	real	1.84	real	20.097
Linux	user	0.42	user	12.25
2 x 3GHz P4 Xeon	system	0.73	system	0.733
4 GB RAM	real	1.2	real	12.99
Sun OS	user	1.645	user	45.495
2 x 1GHz UltraSparc	system	0.89	system	0.885
8 GB RAM	real	2.87	real	47.725
Windows XP	user	0.843	user	9.937
2 x 3GHz P4 Xeon	system	0.25	system	0.250
4 GB RAM	real	1.125	real	10.344
Linux (casa)	user	0.224	user	13.317
2GHz AMD Athlon64	system	0.524	system	0.660
1 GB RAM	real	0.799	real	14.554

La causa de este fenómeno es la forma en que C gestiona la memoria, pues C/C++ almacenan las matrices por filas. Ejemplo: *int* $m[4][4]$

m[0][0]	m[0][1]	m[0][2]	m[0][3]
m[1][0]	**m[1][1]**	m[1][2]	m[1][3]
m[2][0]	m[2][1]	**m[2][2]**	m[2][3]
m[3][0]	m[3][1]	m[3][2]	**m[3][3]**

Memoria Virtual

Es una utilidad que permite a los programas direccionar la memoria desde un punto de vista lógico, sin importar la cantidad de memoria principal física disponible. Se concebió como método para tener múltiples trabajos de usuario residiendo en memoria principal de forma concurrente, de forma que no exista un intervalo de tiempo de espera entre la ejecución de procesos sucesivos, es decir, mientras un proceso se escribe en almacenamiento secundario y se lee el sucesor. Se introdujeron los sistemas de paginación, que permiten que los procesos se compriman en un número determinado de bloques de tamaño fijo, denominados páginas. Un programa referencia a una palabra por medio de una

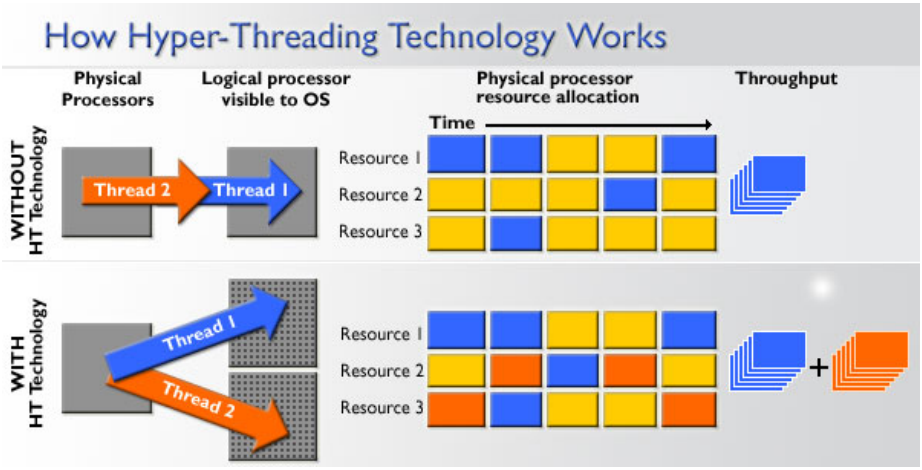
dirección virtual, que consiste en un número de página y un desplazamiento dentro de la página.

Todas las páginas de un proceso se mantienen en disco. Cuando un proceso está en ejecución, algunas de sus páginas se encuentran en memoria principal, y si se referencia a una página que no está en memoria principal el hardware de gestión de memoria lo detecta y permite que la página que falta se cargue (carga bajo demanda).

1.2 Clasificación

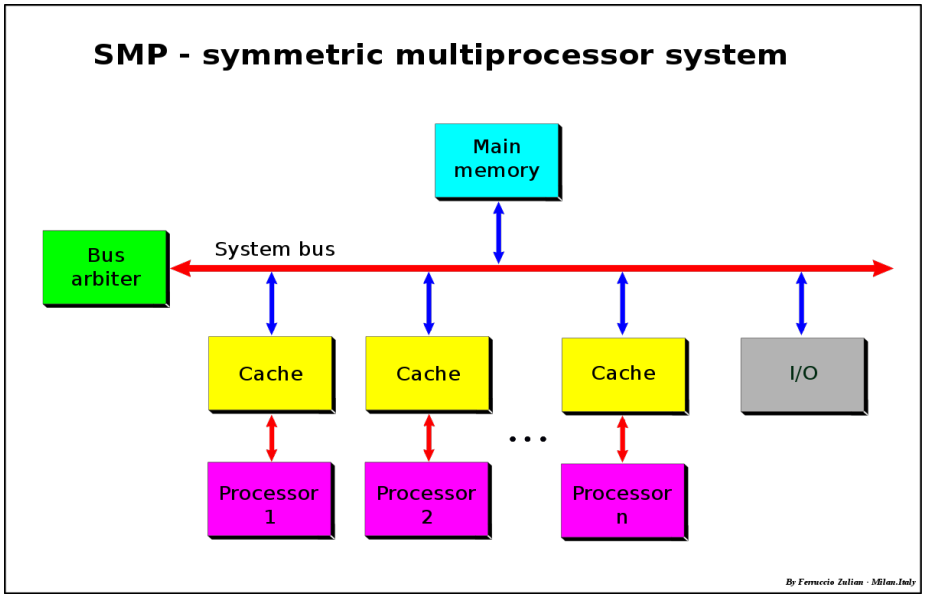
1.2.1 Clasificación "práctica" de arquitecturas paralelas

- Multiprocesadores de memoria compartida:
  - SMT(Simultaneos Multithreading/Hyperthreading): permite a una única CPU ejecutar varios flujos de control. Esto requiere tener múltiples copias de algunos componentes hardware de la CPU, como contadores de programa y registros de archivo, mientras otras partes siguen siendo únicas como las unidades que realizan aritmética con punto flotante. Cuando un procesador tiene Hyper-Threading puede tener de 2 a 64 hebras (puede tener más, veanse procesadores de servidor), dependiendo del número de núcleos físicos del mismo.



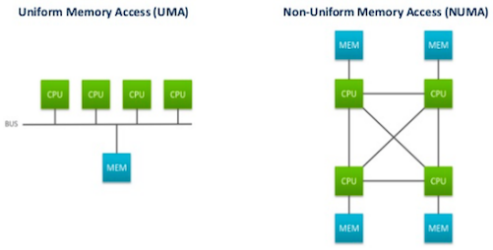
  - SMP(Symmetric Multi-Processing): el núcleo puede ejecutar en cualquier procesador, y normalmente cada procesador realiza su propia planificación del conjunto disponible de procesos e hilos. El núcleo puede construirse como múltiples procesos o múltiples hilos, permitiéndose la ejecución de partes del núcleo en paralelo. El enfoque SMP complica el sistema operativo, ya que debe asegurar que dos procesadores no seleccionan un mismo proceso y que no se pierde ningún proceso

de la cola. Se deben emplear técnicas para resolver y sincronizar el uso de los recursos. Suelen tener entre 2 y 256 procesadores.

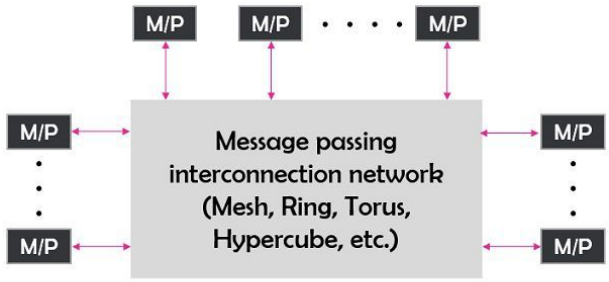


  - UMA/ccNUMA (Uniforma Memory Access/Cache Coherent UMA): Se define como la situación en la cual el acceso a cualquier RAM desde CPU toma siempre la misma cantidad de tiempo. Suele tener entre 2 y 4096 procesadores.

- Multiprocesadores masivamente paralelos:
  - NUMA/ccNUMA (Non Uniform Memory Access/ Cache Coherent NUMA): A diferencia de UMA, algunas partes de la memoria pueden tomar más tiempo de acceso que otras, creando una penalización en el rendimiento. Esta penalización se puede minimizar por medio de la administración de recursos.

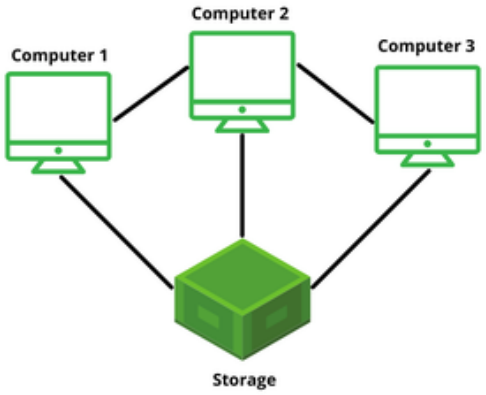


- Paso de mensajes/NoRMA (No Remote Memory Access): en las arquitecturas NoRMA, el espacio de direcciones global no es único y la memoria no es globalmente accesible desde todos los procesadores. El acceso a modulos de memoria remotos es solo posible indirectamente a través del paso de mensajes por medio de la red de interconexión a otros procesadores, lo que en respuesta recibirá los datos buscados en un mensaje de respuesta.



NORMA (No-Remote Memory Access)

- Cluster→+10M procesadores. Al igual que los sistemas multiprocesadores, los sistemas clústes juntan múltiples CPUs para conseguir un trabajo computacional. La diferencia respecto a los sistemas multiprocesadores en que los clústers se componen de dos o más sistemas individuales unidos juntos, a los que se denominan nodos.
  - GPUs



1.2.2 Clasificación Arquitectura de Computadores

- Sistemas monoprocesador

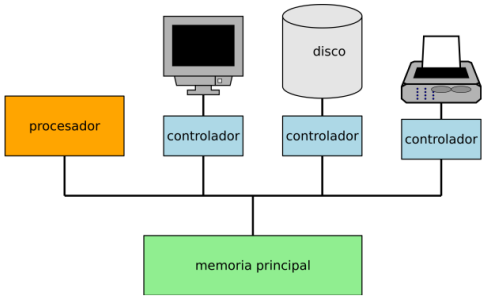
  - Bus único
  - Buses separados/especializados

- Sistemas multiprocesador
  - Multiproceso simétrico (SMP)
  - Multihebra simultánea (SMT)
  - Multinúcleo (SMP)

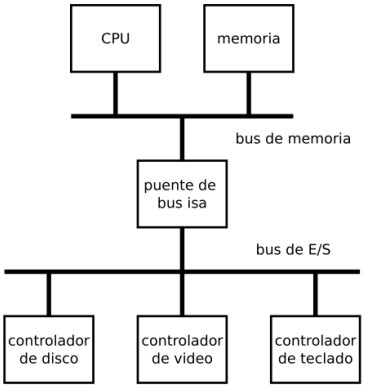
- Sistemas distribuidos

Sistema monoprocesador: Es el modelo más simple, pues conecta todo en un bus común.

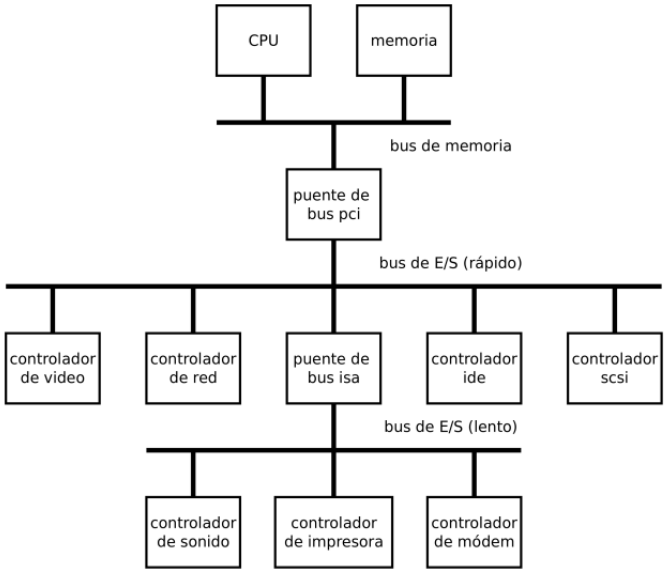
- Ventaja→ precio.
- Inconveniente → infrautilización de componentes por la diferencia de velocidad.



Una posible solución para la diferencia de velocidad es aislar los componentes por velocidad y conectarlos por medio de un puente

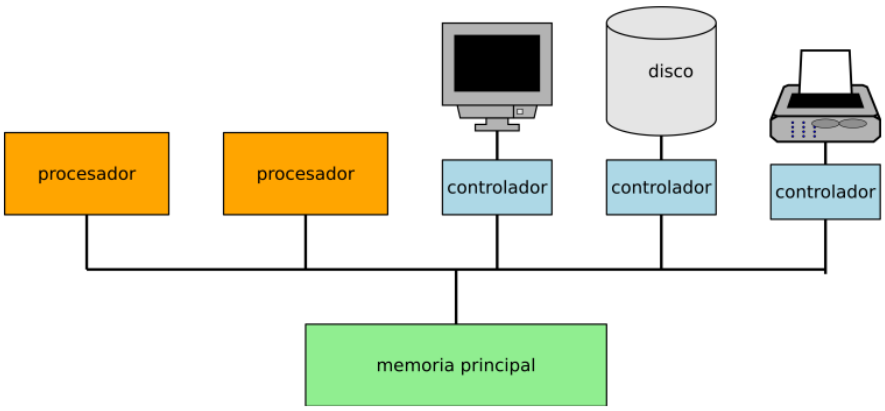


Otra posible solución incluso mejor es separar el bus de E/S en dos buses en función de los dispositivos E/S más rápidos y más lentos, y conectar ambos buses por medio de un puente isa.

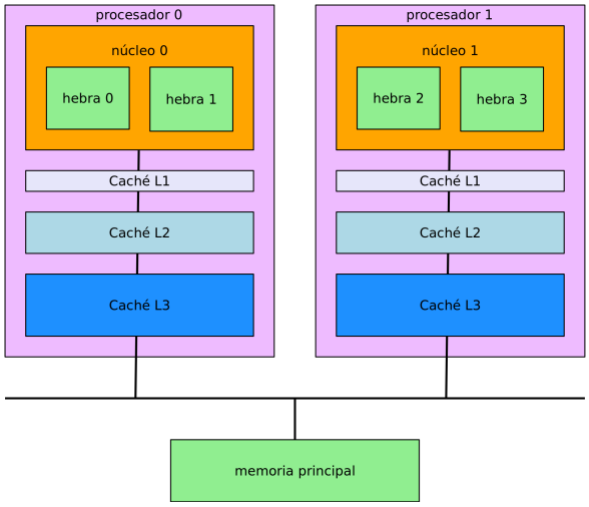


Sistema multiprocesador: multiproceso simétrico. Lo más simple es conectar todos los elementos a un bus común.

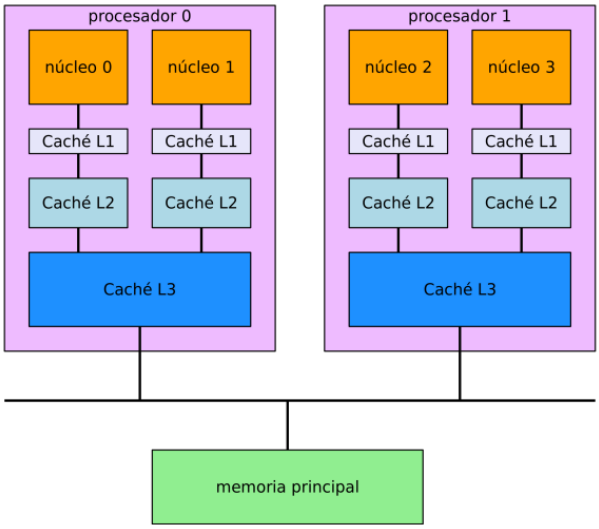
- Ventaja → precio.
- Inconveniente → se agrava la infrautilización de componentes.



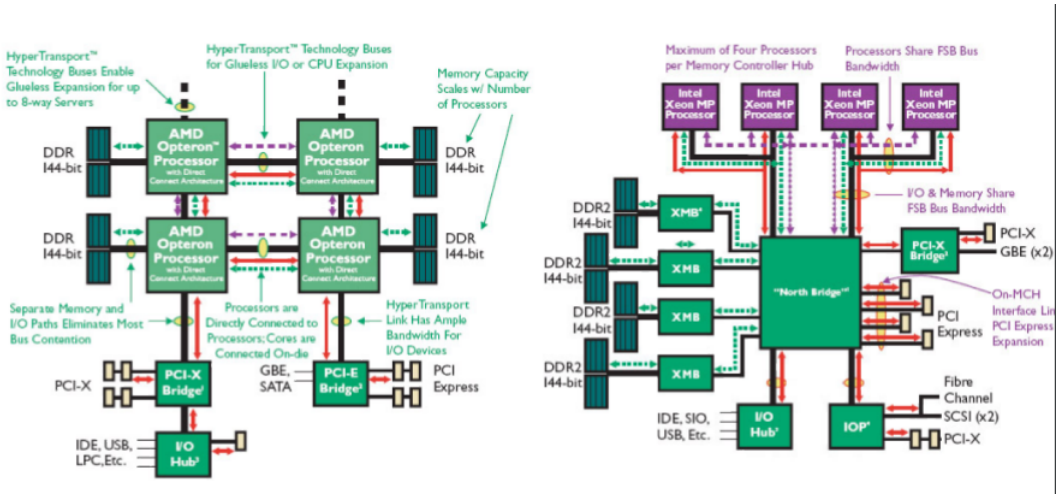
Sistemas multiprocesador: multihebra simultánea



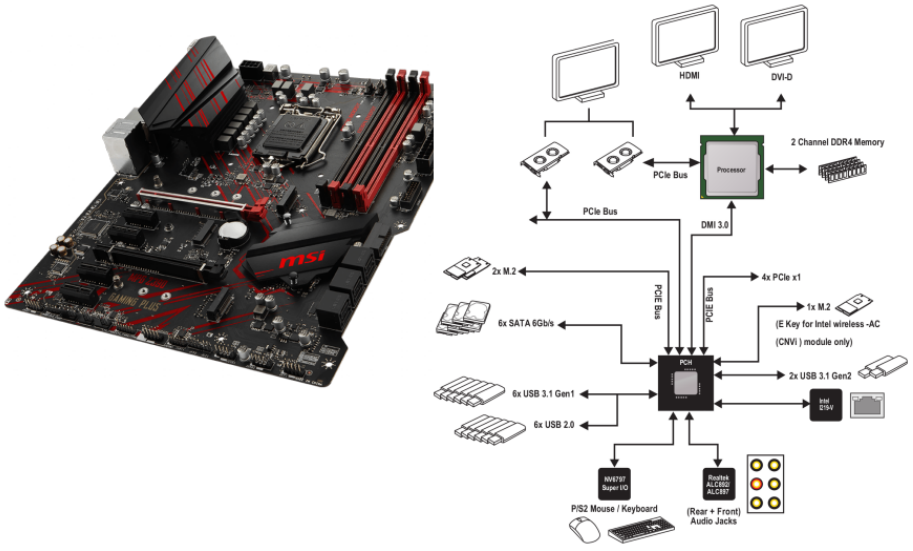
Sistemas multiprocesador: multiproceso simétrico



Sistemas multiprocesador actuales



Arquitecturas de un sistema actual



1.2.3 Componentes

Componentes básicos

- Procesadores
- Jerarquía de memoria.

Surge el problema de que cuanto menor es el tiempo de acceso mayor es el coste por bit, y que cuanto mayor es la capacidad menor la velocidad de acceso. Para lidiar con este dilema surge la jerarquía de memoria. En la jerarquía de memoria según se desciende disminuye el coste por bit, aumenta la capacidad, aumenta el tiempo de acceso y se reduce la frecuencia de acceso a ese nivel de la jerarquí por parte del procesador.

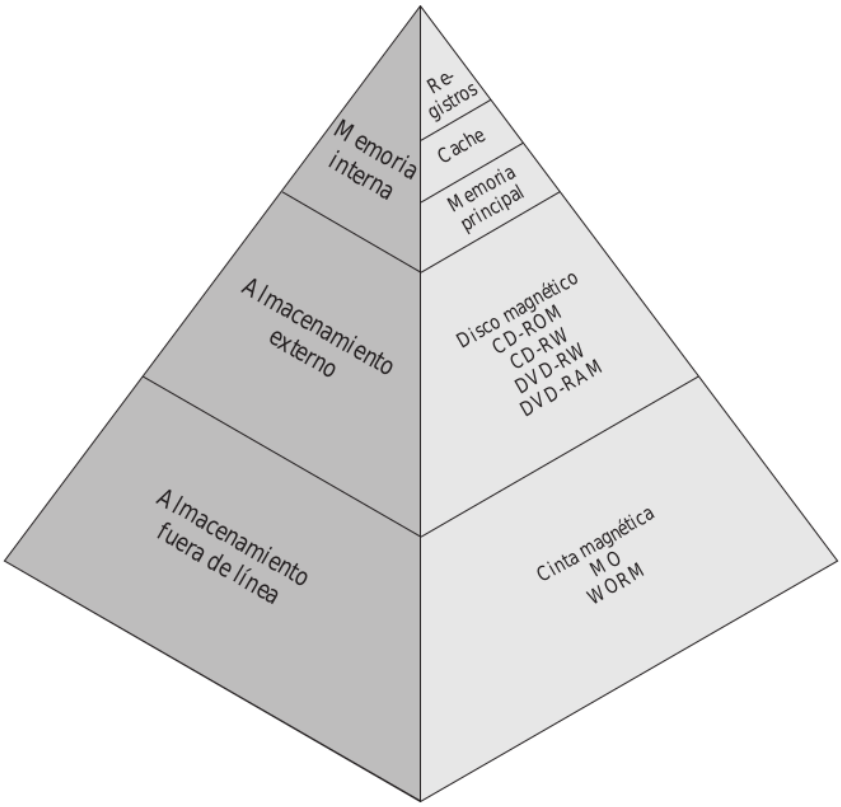


Figura 1.14. La jerarquía de memoria.

- Buses de interconexión: AGP, Hypertransport, IDE, IEEE 1394, ISA, M.2, PCI, PCIe, SATA, SCSI, USB, ...
- Entrada/Salida: controladores, canales de DMA, procesadores de E/S,...
- Periféricos: altavoz, disco, impresora, micrófono, monitor, ratón, teclado, ...

1.2.4 Procesador

Interfaz del procesador

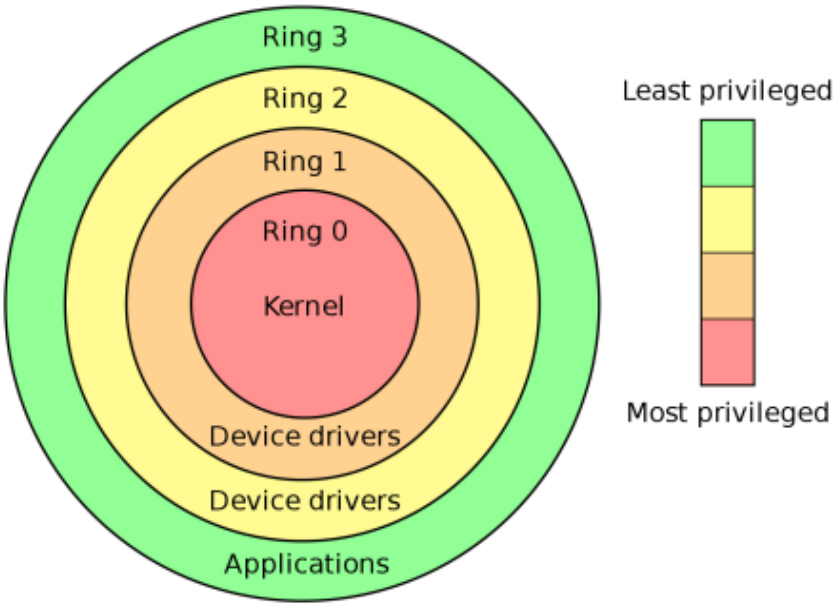
- Conjunto de instrucciones
  - transferencia:

    `in, mov, out,...`
  - modificación:

    `add, and, div, mul, or, sub,...`
  - control:

    `cli, sti,...`

    Chuleta instrucciones 8086
- Registros generales y especiales
- Al menos dos modos de ejecución con diferentes privilegios:
  - privilegiado: acceso completo
  - no privilegiado: acceso limitado ⇒ excepción



Registros del procesador: familia x86-64

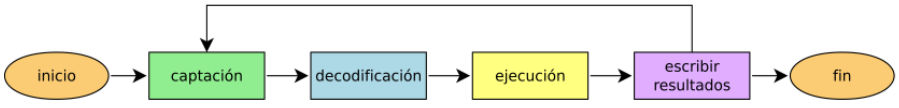
En el modo núcleo (modo privilegiado, modo kernel, ...) se pueden ejecutar instrucciones privilegiadas y se puede accesder a áreas de memoria protegidas. Sóla mente el núcleo o kernel del sistema operativo puede ejecutar instrucciones en modo privilegiado. Algunos ejemplos de instrucciones privilegiadas son:

- Acceso a los dispositivos de E/S: consultar el estado de los dispositivos de E/S, llevar a cabo DMA (Direct Memory Access), atrapar interrupciones.
- Manipular la unidad de gestión de memoria (MMU): manipualar las tablas de segmento y páginas, cargar y vaciar el búfer de traducción anticipada (TLB)
- Configurar varios modos de funcionamiento: nivel de prioridad de interrupciones, alterar el vector de interrupción.
- Utilizar la instrucción *halt* para activar el modo de ahorro de energía. Esta instrucción detiene abruptamente la CPU hasta que la siguiente interrupción externa ha sido tratada. También es común que se ejecute cuando no hay trabajo inmediato que ejecutar, de forma que se pone al procesador en un estado ocioso.

El procesador comprueba el nivel de privilegio en la ejecución de cada instrucción. Los posibles cambios de privilegio son:

- Usuario ⇒ Núcleo: ganar privilegios
  - Al arrancar.
  - Llamada al sistema.
  - Interrupción hardware.
  - Excepción.
- Núcleo ⇒ Usuario: perder privilegios
  - El sistema operativo prepara el entorno necesario para que la aplicación comience su ejecución.
  - El sistema operativo termina alguna de sus actividades y devuelve el control a la aplicación.

Ciclo de instrucción: se denomina ciclo de instrucción al procesamiento requerido por una única instrucción. El ciclo de instrucción básco es:



- El procesador capta una instrucción desde memoria.
- La instrucción debe ser decodificada para averiguar su tipo.
- Conocido el tipo puede ser necesario la captación de nuevos operandos.
- Se ejecuta la instrucción.

- Se almacenan los resultados de la ejecución.
- El proceso se repite instrucción a instrucción hasta que el programa termina.

Tendencias en el diseño de procesadores

- CISC (Complex Instruction Set Computing)⇒ RISC (Reduced Instruction Set Computing) ⇒ VLIW (Very Long Instruction Word).

  CISC: Es un tipo de diseño de microprocesador. Este contiene un conjunto de instrucciones muy grande que van desde instrucciones muy simples a instrucciones muy especializadas. Se introducen instrucciones que en un diseño RISC se requieren de varias instrucciones, reduciendo así el número de instrucciones de los programas, pero al ser instrucciones más complejas acaban requiriendo más ciclos de reloj para su ejecución.

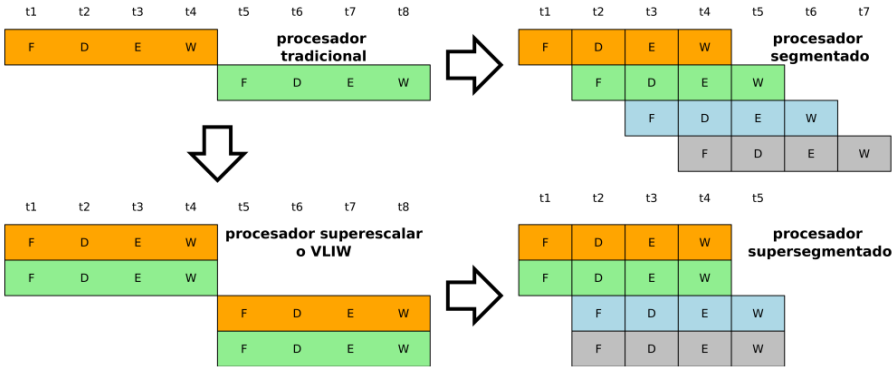
  RISC: Como su nombre dice se trata de un diseño de microprocesador. Este diseño contiene instrucciones muy simples y con la combinación de ellas se puede obtener cualquier programa. Si bien su ejecución es rápida, pues son instrucciones muy simples, su tamaño en memoria puede llegar a ser muy grande, pues la ejecución de un programa simple puede requerir de muchas instrucciones RISC.

  VLIW: se trata de un procesador segmentado que puede terminar más de una operación por ciclo en el que el compilador es el principal responsable de agrupar operaciones que pueden procesarse en paralelo para definir instrucciones que, de esta forma, se codifican a través de las denominadas palabras de instrucción larga (LIW) o muy larga (VLIW)

- Ejecución concurrente sobre procesadores: intento de explotación del paralelismo entre instrucciones (ILP). ILP es una mediad de cuántas operaciones pueden ejecutarse simultáneamente sin afectar al resultado.

Técnicas de explotación de ILP

- **Segmentación de cauce:** la ejecución de múltiples instrucciones puede solaparse total o parcialmente.
- **Ejecución superescalar:** múltiples unidades de ejecución se utilizan para ejecutar múltiples instrucciones en paralelo. Un procesador superescalar es un procesador segmentado que puede finalizar más de una instrucción por ciclo y que posee recursos hardware para extraer el paralelismo entre instrucciones. Para aprovechar al máximo el procesamiento de instrucciones en paralelo que proporcionan las distintas estapas, el procesador incluye una serie de elementos como ventanas de instruccioens o estaciones de buffers, buffers de renombramiento, buffers de reordenamiento, etc.



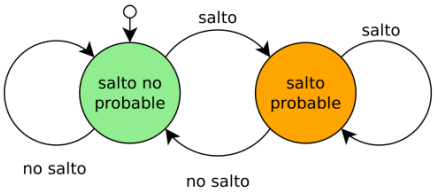
- **Computación con instrucciones explícitamente paralelas (EPIC):** uso del compilador en lugar de complejos circuitos para identificar y explotar el ILP. La intención era permiter un escalado simple del rendimiento sin disparar las frecuencias del reloj. Tiene su base en VLIW.

- **Ejecución fuera de orden:** ejecución de instrucciones en cualquier orden que no viole las dependencias entre instrucciones. El orden de ejecución depende de la disponibilidad de los datos de entrada y las unidades de ejecución, no del orden original del programa.

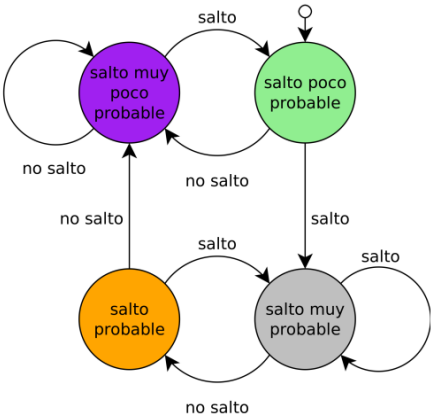
- **Renombrado de registros:** técnica para evitar la innecesaria serialización de instrucciones por la reutilización de registros. Puede ser aplicado por el propio compilador al asignar los registros de la arquitectura, pero también puede implementarse en hardware. Esto es lo usual en procesadores superescalares, donde se incluyen estructuras de buffers con una serie de campos específicos (por ejemplo, buffers de renombramiento).

- **Ejecución especulativa:** permitir la ejecución de instrucciones completas, o partes, antes de conocer con seguridad si su ejecución debe tener lugar. De esta forma se previene el retraso que habría, de tener que hacer el procesamiento después de saber que es necesario. Si resulta que el procesamiento no era necesario, los cambios realizados se revierten y los resultados se ignoran.

- **Predicción de salto:** se utiliza para evitar quedar parado antes de que se resuelvan las dependencias de control. Se utiliza en conjuntción con la ejecución especulativa. Se basa en determinar la alternativa más probable, y continuar el procesamiento, tras las instrucción de salto, con la secuencia de instrucciones que corresponde a dicha opción más probable. Cuando la condición de salto se evalúa, se comprueba si la predicción que se había hecho era correcta o no. Si no lo era habrá que retomar el procesamiento a partir de la primera instrucción de la alternativa que no se tomó, es decir, de la menos probable.



(a) Algoritmo de predicción de dos estados: funciona bien en bucles salvo entrada y salida.



(b) Algoritmo de predicción de cuatro estados: permite recordar estado previo de un bucle.

- **Multihebra simultánea (SMT):** técnica que permite la ejecución de múltiples hebras de ejecución para aprovechar mejor las unidades funcionales de los procesadores superescalares.

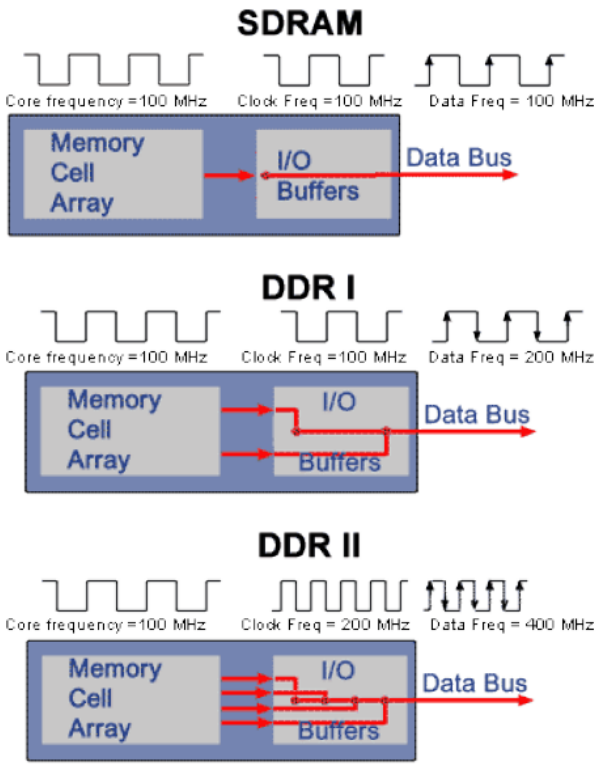
1.2.5 Memoria

Jerarquía de memoria: Se requiere de jerarquía de memoria entre otras cosas por la gran diferencia de velocidad entre procesador y memoria. La jerarquía de memoria puede aliviar este problema gracias a **los principios de localidad** y a la **regla 90/10**:

- Localidad **espacial:** la información a la que se accede suelen estar próxima a la que ha sido accedida con anterioridad.
- Localidad **temporal:** la información a la que se accede una vez suele volver a ser utilizada.
- Regla 90/10: el 10% del código realiza el 90% del trabajo.

Tipos de memoria RAM:

- SRAM (Statica random-access memory): Es un tipo de memoria que utiliza circuitería flip-flop para almacenar cada bit. La diferencia con una DRAM es que la segunda debe de refrescarse periódicamente. La SRAM es más rápida y más cara que la DRAM. Se utiliza típicamente para la caché y los registros internos de la CPU mientras que la DRAM se utiliza para la memoria principal.
- DDR SDRAM I (Double Data Rate Synchronous RAM): Entre otras cosas las memorias DDR como su nombre indica tienen dos ciclos de reloj, es decir se hace un envío de información cuando el reloj sube y otro cuando



el reloj baja. Con esto hace que una memoria DDR con una frecuencia de reloj x duplique el ancho de banda de una SDR SDRAM con igual frecuencia de reloj. Este tipo de memoria se ha visto superada por las versiones 2, 3, 4 y 5 de la misma. Ninguno de sus sucesores tienen compatibilidad ni con su predecesor ni con su sucesor, lo que quiere decir que una RAM DDR1 SDRAM no es compatible con DDR2, DDR3, ...

- DDRAM II

Cantidad de memoria en registros

- Tipos de registros:
- RISC:
  - 32 de propósito general (32 ó 64 bits)
  - 32 de punto flotante (64 bits IEEE 754)
  - multimedia (64,...,256 bits)
- CISC:
  - IA32: 8 de propósito general, 8 de punto flotante, 8 multimeida.

  - IA64: 128 de propósito general, 128 de punto flotante.

- Algunos procesadores tienen varios conjuntos de estos registros (ventanas de registros)

Análisis de una jerarquía de dos niveles

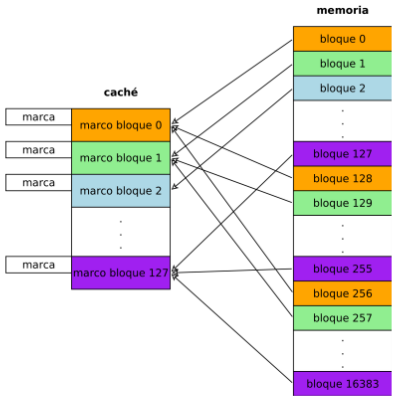
$$\overline{T}_{acceso} = (1 - T_{fallos}) \times T_{cache} + T_{fallos} \times (T_{cache} + T_{ram})$$

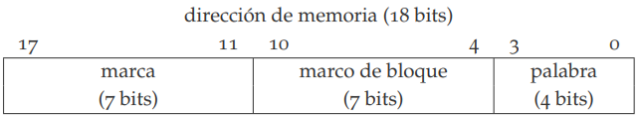
Parámetros de diseño de memorias caché

- Tamaño:
  - L1:8,...,256KB
  - L2:1,...,16MB
  - L3:4,...,128MB
- Tamaño de bloque: 32,...,128B
- Tiempo de acceso: 1,...,10ns
- Política de búsqueda:
  - bajo demanda
  - anticipativas
- Política de colocación:
  - correspondencia directa: el bloque B_j de memoria principal se puede ubicar sólo en el marco de bloque que cumple la siguiente relación $i = j \mod m$, donde m es el número total de líneas que tiene la cache.
  - asociativa por conjuntos: En la corresponcencia asociativa por conjuntos las líneas de memoria caché se agrupan en $v = 2^d$ conjunto con k líneas/conjuto o vías. Se cumple que el número total de marcos de bloque que tiene la caché $m = v * k$. Un bloque B_j de memoria principal se puede ubicar sólo en el conjunto C_i de memoria caché que cumple la siguiente relación $i = j \mod v$.
  - completamente asociativa: la caché se organiza en un único conjunto de caché con varias líneas de caché. Un bloque de memoria puede ocupar cualquiera de las lineas de caché. La organización de la caché se puede enmarcar como una matriz de filas (1*m).
- Política de reemplazo:
  - LRU
  - FIFO
  - aleatoria

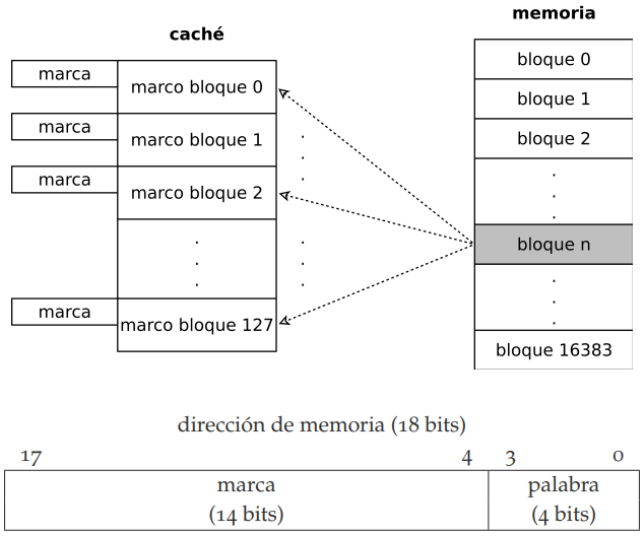
- Política de actualización:
  - escritura directa
  - post-escritura
- Otras características importantes:
  - caché de víctimas
  - inclusiva/exclusiva
  - unificada/separada

Políticas de colocación: correspondencia directa

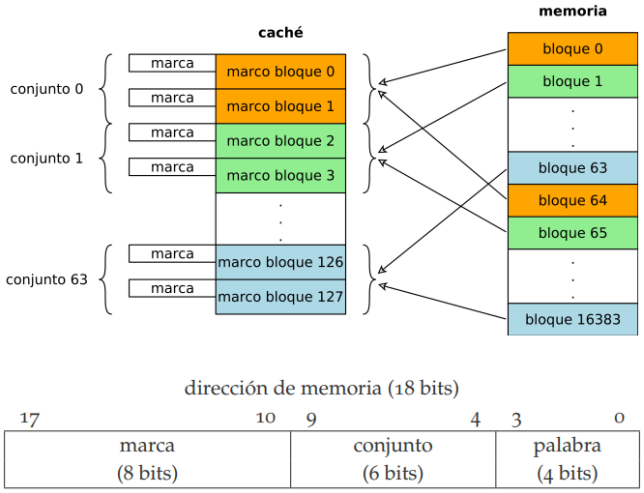




Políticas de colocación: correspondencia totalmente asociativa



Políticas de colocación: correspondencia asociativa por conjuntos (2 M/C)



Protección de memoria.

El sistema operativo debe proteger a los programas entre sí y a él mismo de programas maliciosos o erróneos. Este problema se soluciona utilizando los llamados espacios de direcciones (AS, Address Space). El espacio de direcciones de un computador corresponde al rango de direcciones disponible para un programa de computador.

El procesador dispone de recursos para establecer límites al espacio de direcciones de memoria accesibles por los programas: memoria virtual.

Tabla 8.1. Características de la paginación y la segmentación.

Paginación sencilla	Paginación con memoria virtual	Segmentación sencilla	Segmentación con memoria virtual
Memoria principal particionada en fragmentos pequeños de un tamaño fijo llamados marcos	Memoria principal particionada en fragmentos pequeños de un tamaño fijo llamados marcos	Memoria principal no particionada	Memoria principal no particionada
Programa dividido en páginas por el compilador o el sistema de gestión de la memoria	Programa dividido en páginas por el compilador o el sistema de gestión de la memoria	Los segmentos de programa se especifican por el programador al compilador (por ejemplo, la decisión se toma por parte el programador)	Los segmentos de programa se especifican por el programador al compilador (por ejemplo, la decisión se toma por parte el programador)
Fragmentación interna dentro de los marcos	Fragmentación interna dentro de los marcos	Sin fragmentación interna	Sin fragmentación interna
Sin fragmentación externa	Sin fragmentación externa	Fragmentación externa	Fragmentación externa
El sistema operativo debe mantener una tabla de páginas por cada proceso mostrando en el marco que se encuentra cada página ocupada	El sistema operativo debe mantener una tabla de páginas por cada proceso mostrando en el marco que se encuentra cada página ocupada	El sistema operativo debe mantener una tabla de segmentos por cada proceso mostrando la dirección de carga y la longitud de cada segmento	El sistema operativo debe mantener una tabla de segmentos por cada proceso mostrando la dirección de carga y la longitud de cada segmento
El sistema operativo debe mantener una lista de marcos libres	El sistema operativo debe mantener una lista de marcos libres	El sistema operativo debe mantener una lista de huecos en la memoria principal	El sistema operativo debe mantener una lista de huecos en la memoria principal
El procesador utiliza el número de página, desplazamiento para calcular direcciones absolutas	El procesador utiliza el número de página, desplazamiento para calcular direcciones absolutas	El procesador utiliza el número de segmento, desplazamiento para calcular direcciones absolutas	El procesador utiliza el número de segmento, desplazamiento para calcular direcciones absolutas
Todas las páginas del proceso deben encontrarse en la memoria principal para que el proceso se pueda ejecutar, salvo que se utilicen solapamientos (*overlays*)	No se necesita mantener todas las páginas del proceso en los marcos de la memoria principal para que el proceso se ejecute. Las páginas se pueden leer bajo demanda	Todos los segmentos del proceso deben encontrarse en la memoria principal para que el proceso se pueda ejecutar, salvo que se utilicen solapamientos (*overlays*)	No se necesitan mantener todos los segmentos del proceso en la memoria principal para que el proceso se ejecute. Los segmentos se pueden leer bajo demanda
	La lectura de una página a memoria principal puede requerir la escritura de una página a disco		La lectura de un segmento a memoria principal puede requerir la escritura de uno o más segmentos a disco

Hay varias implementaciones:

- Segmentaciones: tamaño variable y arbitrario.
  - registro base: principio del espacio de direcciones.
  - registro límite: tamaño del espacio de direcciones.
- Paginación: tamaño variable en múltiplos de página.
  - tablas de páginas: lista de páginas de un proceso.

La mayoría de los SO asocian un espacio de direcciones diferente para cada proceso (salvo SASOS, Single Adrress Space Operating System):

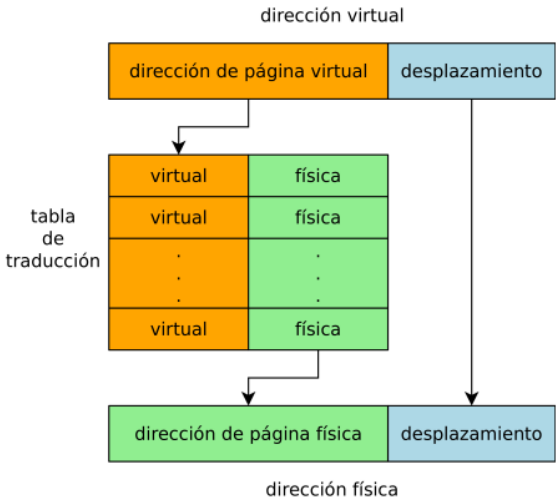
- Ventaja ⇒ protección automática.
- Inconveniente ⇒ dificulta la compartición

Las partes de un espacio de direcciones pueden colocarse en cualquier parte de la memoria física. Algunos espacios de direcciones deben ocupar lugares específicos de la memoria física, ej: dispositivos de E/S.

El procesador debe tener una unidad de gestión de memoria (MMU, Memory Management Unit) extremadamente eficiente porque la traducción entre direcciones virtual-física puede realizarse más de una vez por instrucción. Se requiere una traducción por cada acceso a memoria.

Traducción de dirección virutal a dirección física

1. La ejecución de una instrucción puede requerir varias traducciones, por ejemplo:
   - Captaciño de la instrucción.
   - Captación del dato.
   - Escritura del resultado.
2. Cada traducción puede requerir varios accesos a memoria:
   - Segmentación + paginación.
   - Tablas de página multinivel.
3. Se necesita una tabla de traducción por cada espacio de direcciones.
4. Dependerá del tamaño de las direcciones virtuales y físicas, pero habrán al menos tantas entradas como páginas tenga el proceso.



Fucnionamiento de la segmentación (x86)

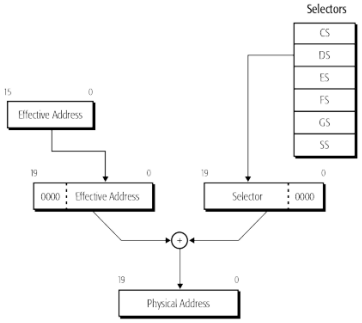


Figura: Cálculo de la dirección física.

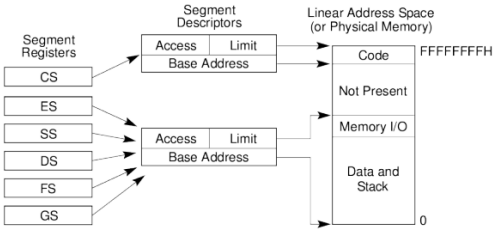


Figura: División del espacio de direcciones de un proceso.

Segmentación + paginación (x86_64)

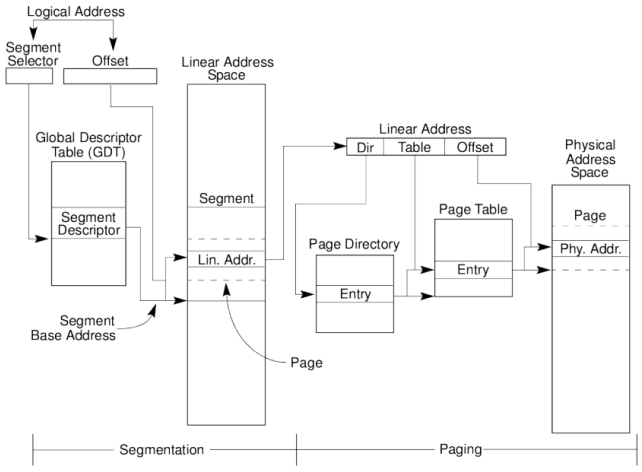
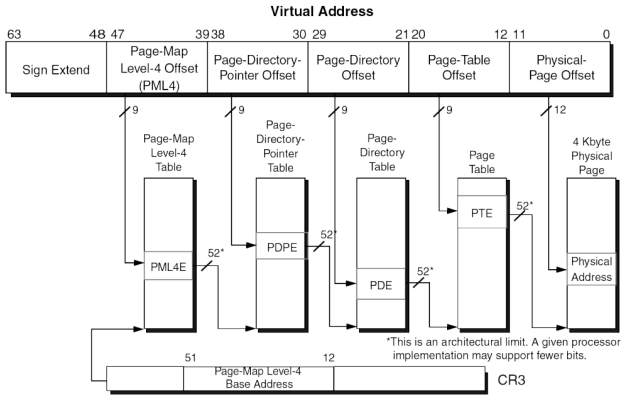


Tabla de páginas multinivel (x86_64)



Estructura habitual tabla de páginas

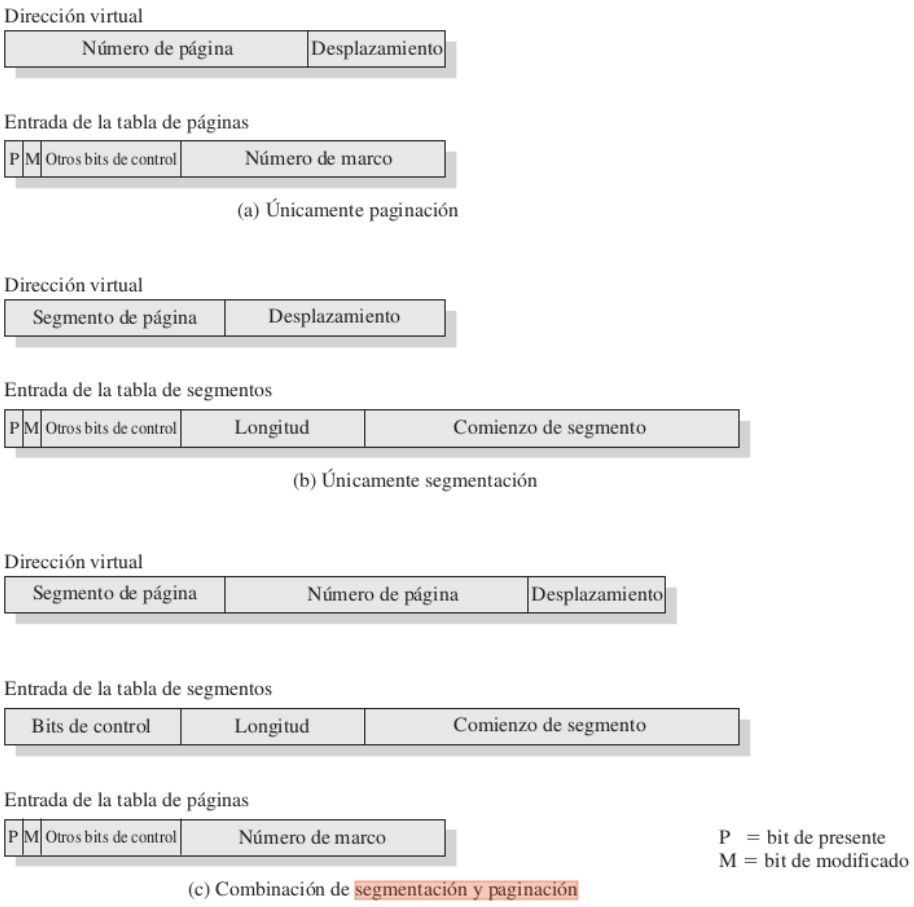


Figura 8.2. Formatos típicos de gestión de memoria.

Buffer de traducción anticipada (TLB)

Traducir direcciones por software es muy lento, mientras que mediante hardware puede hacerse más rápido y además podemos añadir una caché de traducciones (TLB). En principio, toda referencia a la memoria virtual puede causar dos accesos a memoria física, uno para buscar la entrada en la tabla de página apropiada y otra para buscar los datos solicitados. De esa forma, el esquema de memoria virtual básico causaría el efecto de duplicar el tiempo de acceso a la memoria. Para solventar este problema, la mayoría de esquemas de la memoria virtual utilizan una *cache* especial de alta velocidad para las entradas de la tabla de páginas, habitualmente denominada buffer de traducción anticipada. TLB:

- Conenido: pares de direcciones virtual/física.
- Implementado mediante memorias asociativas.

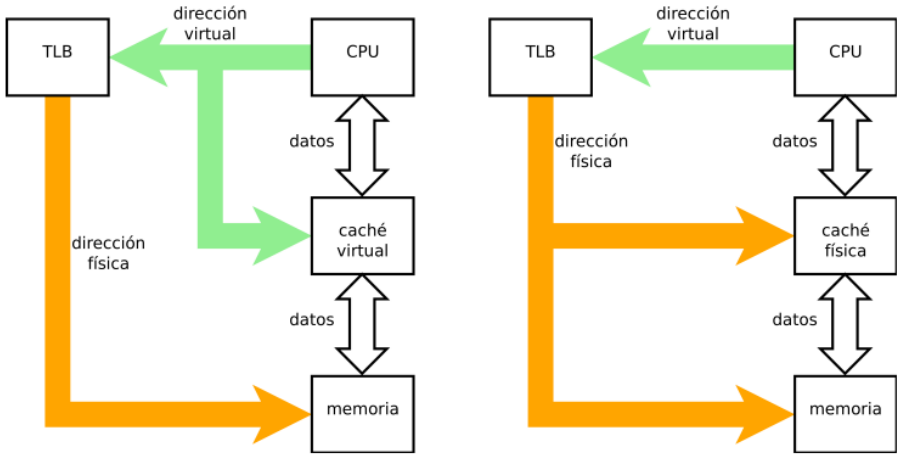
- Tamaño típico: de 32 a 128 entradas.

Hay dos tipos fundamentales:

- Etiquetadas: añaden una marca del espacio de direcciones al que pertenece.
- No etiquetadas: no poseen la capacidad anterior, la mayoría.

Funcionamiento:

- Si se encuentra la dirección en el TLB se devuelve su traducción.
- Si no se encuentra es necesario calcular la traducción.
  - Hardware: se devuelve la traducción correcta (x86).
  - Software: más lento pero más flexible (PowerPC).
- Es fundamental conocer la interacción TLB/caché.

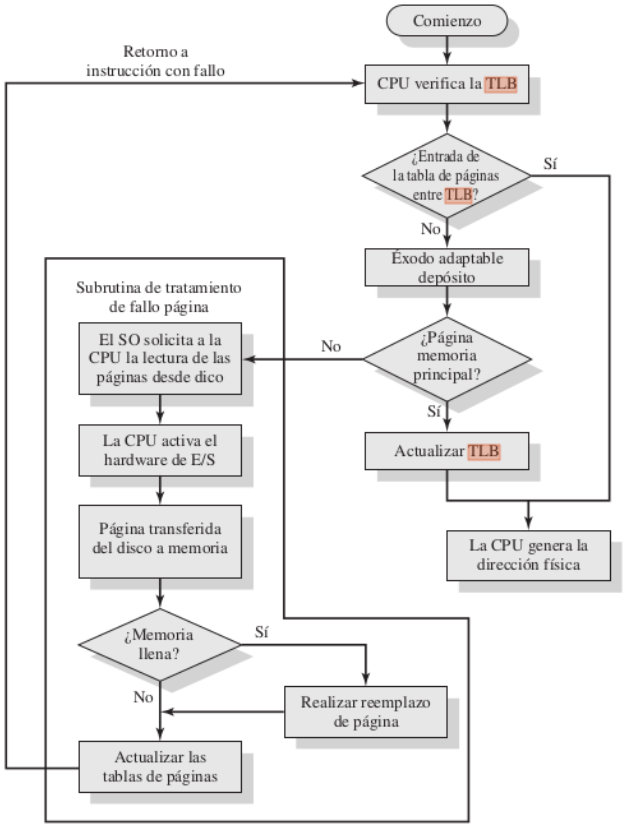


El proceso que sigue generalmente es el siguiente:

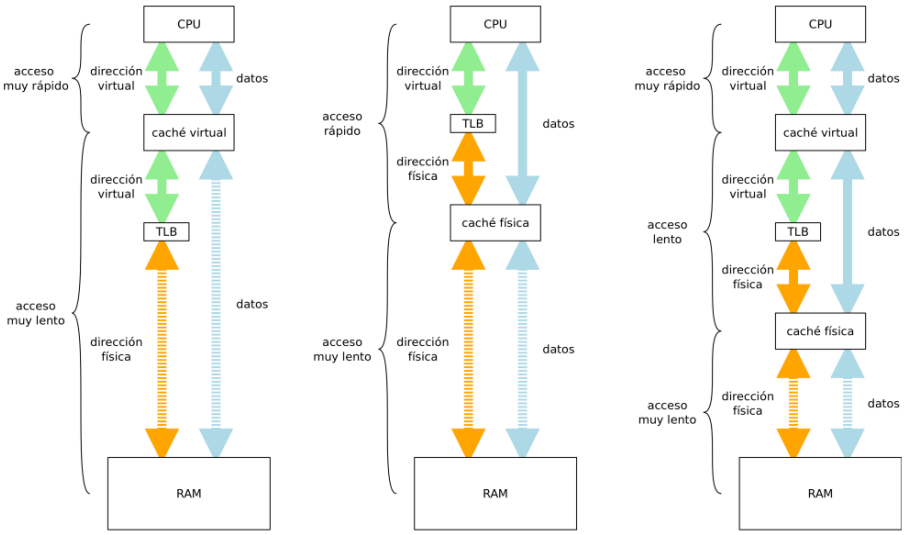
1. Dada una dirección virtual, el procesador primero examina la TLB.
2. Si la entrada de la tabla de páginas solicitada está presente (acierto en la TLB), entonces se recupera el número de marco y se construye la dirección real.
3. Si la entrada de la tabla de páginas no se encuentra (fallo en la TLB), el procesador utiliza el número de página para indexar la tabla de páginas del proceso y examinar la correspondiente entrada de la tabla de páginas.

(a) Si el bit de presente está puesto a 1, entonces la página se encuentra en memoria principal, y el procesador puede recuperar el número de marco desde la entrada de la tabla de páginas para construir la dirección real. El procesador también autorizará la TLB para incluir este nueva entrada de tabla de páginas.

(b) Si el bit presente no está puesto a 1, entonces la página solicitada no se encuentra en memoria principal y se produce un fallo de acceso memoria, llamado **fallo de página**.



Esquemas de direccionamiento de caché



Control de Entrada/Salida
El sistema operativo inicia una operación E/S:

- Instrucciones especiales: **in/out** (x86)
- E/S en memoria: **mov** (x86)
  - Acceso al hardware como direcciones de memoria.
  - Requiere de una MMU capaz de traducir el bus de E/S.

Por otro lado es SO puede averiguar cuando una orden E/S finaliza de las siguientes formas:

- Sondeo: leer el puerto de estado hasta encontrar el valor adecuado. La CPU continuamente rueba todos y cada uno de los dispositivos conectados a él para detectar si algún dispositivo necesita atención de la CPU. Algoritmo de sondeo:
  1. Cuando un dispositivo tiene algún comando para ser ejecutado por la CPU, comprueba continuamente el bit de ocupado de la CPU hasta que se borra (0).
  2. Cuando se borra el bit de ocupado, el dispositivo establece el bit de escritura en su registro de comando y escribe un byte en el registro de salida de datos.
  3. Ahora el dispositivo establece (1) el bit de comando listo.
  4. Cuando la CPU verifica el bit de comando listo y lo encuentra establecido (1), establece (1) su bit de ocupado.

5. Luego, la CPU lee el registro de comando del dispositivo y ejecuta el comando del dispositivo.
6. Después de la ejecución del comando, la CPU borra (0) el bit de comando listo, el bit de error del dispositivo para indicar la ejecución exitosa del comando del dispositivo y además borra (0) su bit de ocupado para indicar que la CPU está libre para ejecutar el comando de algún otro dispositivo.

- Interrupción: existe una línea física mediante la cual se avisa del fin de la operación y se cede el control al SO. Procesamiento de interrupciones:
  1. El dispositivo genera una señal de interrupción hacia el procesador.
  2. El procesador termina la ejecución de la instrucción actual antes de responder a la interrupción.
  3. El procesador comprueba si hay petición de interrupción pendiente, determina que hay una y manda y una señal de reconocimiento al dispositivo que produjo la interrupción. Este reconocimiento permite que el dispositivo elimine su señal de interrupción.
  4. En ese momento, el procesador necesita prepararse para transferir el control a la rutina de interrupción. Para comenzar, necesita salvar la información requerida para reanudar el programa actual en el momento de la interrupción. La información mínima requerida es la palabra de estado del programa (PSW) y la posición de la siguiente instrucción que se va a ejecutar, que está contenida en el contador de programa. Esta información se peude apilar en la pila de control de sistema.
  5. A continuación, el procesador carga del contador del programa con la posición del punto de entrada de la rutina de interrupción que responderá a esta interrupción. Dependiendo de la arquitectura de computador y del diseño del sistema operativo, puede haber un único programa, uno por cada tipo de interrupción o uno por cada dispositivo y tipo de interrupción. Si hay más de una rutina de manejo de interrupción, el procesador debe determinar cuál invocar. Esta información puede estar incluida en la señal de interrupción original o el procesador puede tener que realizar una petición al dispositivo que generó la interrupción para obtener una respuesta que contiene la información requerida.
  6. En este momento, el contador del programa y la PSW vinculados con el programa interrumpido se han almacenado en la pila del sistema. Sin embargo, hay otra información que se considera parte del estado del programa en ejecución. En concreto, se necesita salvar el contenido de los registros del procesador, puesto que estos registros los podría utilizar el manejador de interrupciones. Por tanto, se deben salvar todos estos valores, así como cualquier otra información de

estado. Generalmente, el manejador de interrupción comenzará salvando el contenido de todos los registros en pila.

7. El manejador de interrupción puede en este momento comenzar a procesar la interrupción. Esto incluirá un examen de la información de estado relacionada con la operación de E/S o con otro evento distinto que haya causado la interrupción. Asimismo, puede implicar el envío de mandtos adicionales o reconocimientos al dispositivo de E/S.
8. Cuando se completa el procesamiento de la interrupción, se recuperarn los valores de los registros salvados en la pila y se restituyen los registros.
9. La última acción consiste en restituir de la pila los valores de la PSW y del contador del programa. Como resultado, la siguiente instrucción que se va a ejecutar corresponderá al programa previamente interrumpido.

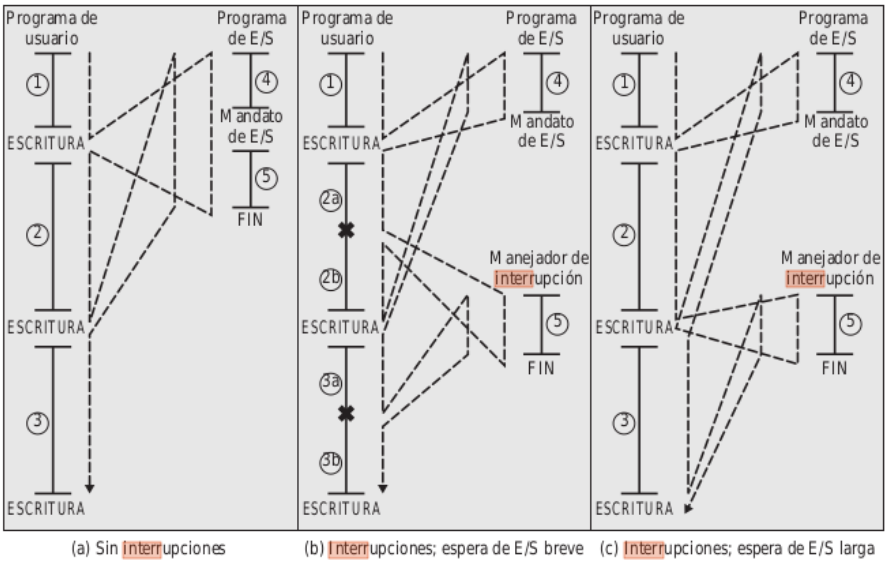
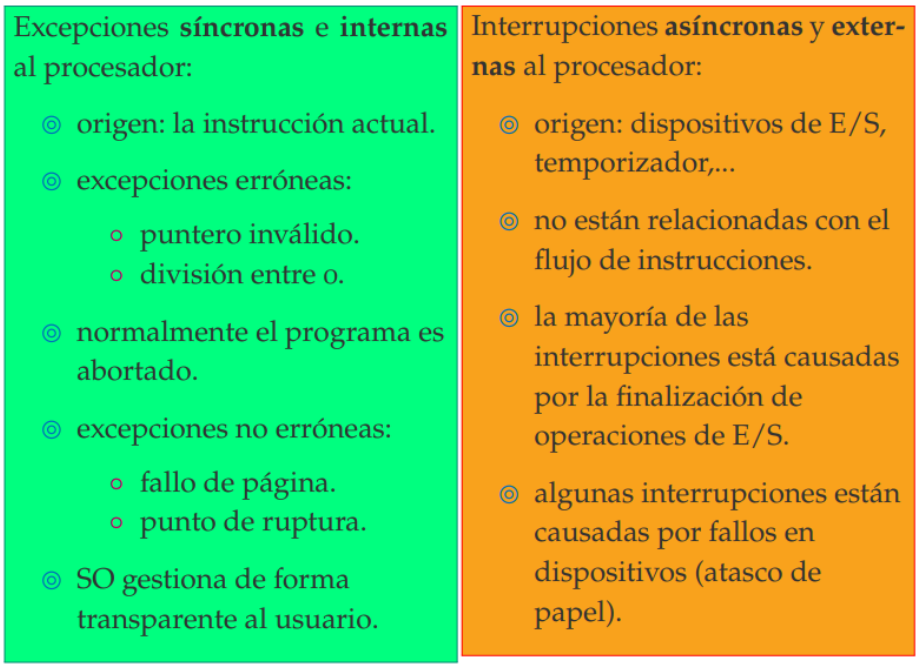


Figura 1.5. Flujo de programa del control sin interrupciones y con ellas.

Eventos:excepciones e interrupciones
Eventos o "situaciones especiales":

- Excepciones (síncronas): llamadas al sistema.
- Interrupciones (asíncronas): fin de operación de DMA.

Hay otras diferencias como el origen, la predictibilidad o la reproducibilidad. El manejo de excepciones e interrupciones debe producirse a tiempo y es específico para cada tipo de procesador.



Interrupciones software (int/syscall - swi)

Se tratan de un mecanismo para ceder el contro al SO elevando el nivel de privilegio. Son síncroncas, predecibles y reproducibles, luego aunque se llamen interrupciones son excepciones.

Un mismo programa aislado siempre provoca las mismas:

- Instrucción desconocida.
- Instrucción errónea (divisón entre 0).
- Modo de direccionamiento erróneo.
- Violación del espacio de direcciones.
- Llamada al sistema.
- Fallo de página (solo políticas de paginación locales).

Deben atenderse, sino se daría un comportamiento erróneo. Producen una sobrecarge en espacio y tiempo fuera del programa que la solicita.

Interrupciones

Se trata de un mecanismo para solicitar la atención de la CPU. Son asíncronas, no predecibles y no reproducibles. Un periférico puede solicitar una interrupción independientemente del estado de la CPU o el proceso actual.

Ejemplos:

- Eventos externos: sensores
- Final de una operación DMA.
- Final de una operación E/S.