Computación Paralela y Distribuida

2da Sesión

John Corredor, PhD
ECEI - IS

Objetivo

- Comprender la Jerarquía de Memoria

Jerarquía de Memoria

La memoria está construida jerárquicamente, en diferentes capas. Esto se debe a que los objetivos finales del diseño de la memoria son:

- Tener muchos (gigabytes, terabytes, etc., suficientes para contener todo el espacio de direcciones),
- Hacerlo rápido (tan rápido como los registros de la CPU),
- Que sea accesible (no demasiado costoso).

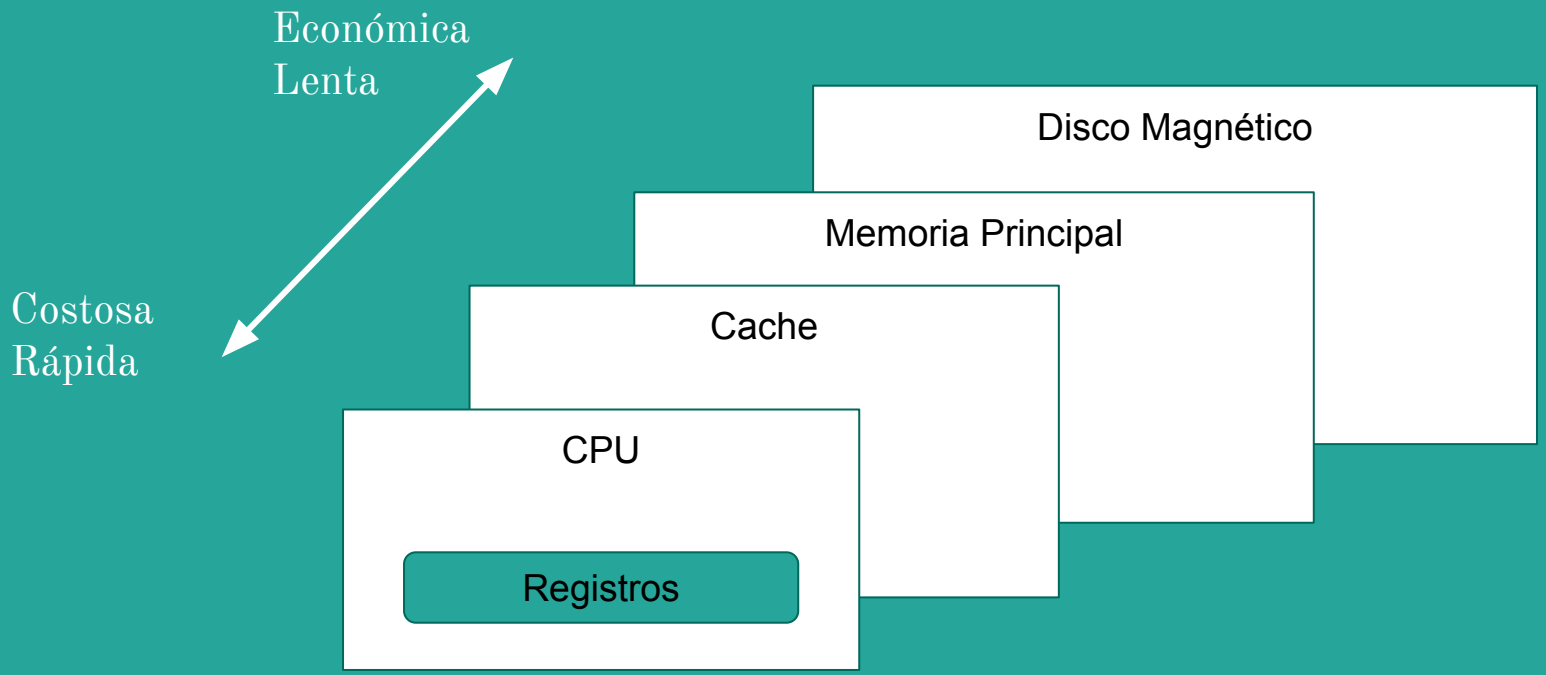
Para crear la ilusión de mucha memoria rápida, se crea una estructura jerárquica de memoria, con múltiples niveles.

Tipos de Memoria			
Tecnología	Velocidad	Capacidad	Costo
SRAM en CPU	La más rápida	La más pequeña	El más alto
DRAM en MotherBoard			
Memoria Flash			
Disco Magnético	El más lento	La más grande	El más bajo

Tecnologías		
Tecnología	Tiempo de Acceso	Precio por GB
SRAM	0,5 - 2,5 ns	$300
DRAM	50 - 70 ns	6$
Memoria Flash	5.000 - 50.000 ns	0,40$
Disco Magnético	5’000.000 - 20’000.000 ns	0,02$

Precios del 2018

CPU ⊂ Cache ⊂ Memoria Principal ⊂ Disco



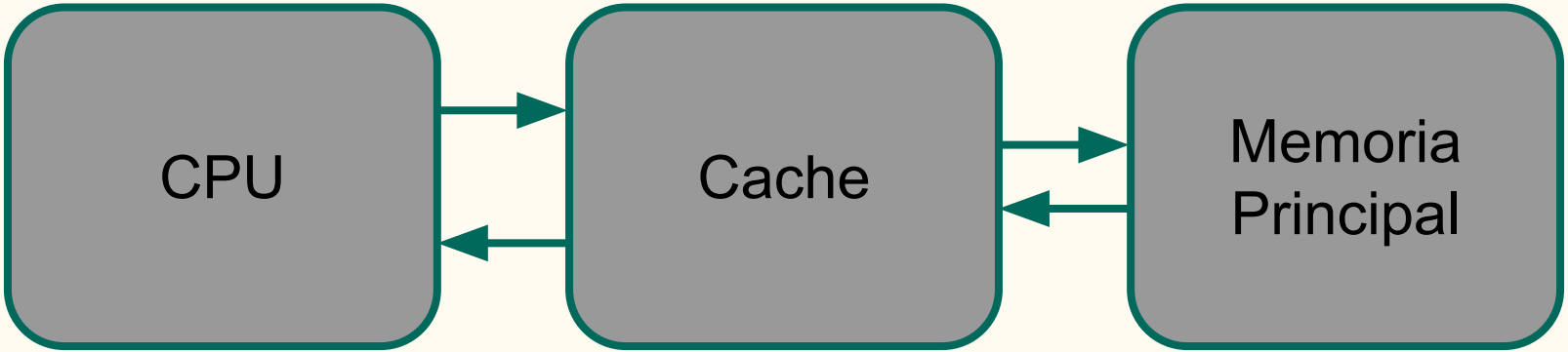
Localidad Temporal

El contenido de la memoria recientemente accedido tiende a ser accedido de nuevo en poco tiempo.

```
for (i=0; i<100; i++)
```

Localidad Espacial

El contenido de la memoria que se encuentra cerca del contenido de la memoria recientemente accedida tiende a ser accedida también, dentro de los próximos 10-100 ciclos de reloj.



- Solicitud de memoria de la CPU
- Los datos no se encuentran en el caché
- Petición de memoria de la caché a la memoria principal
- Enviar datos de la memoria a la caché
- Almacenar los datos en la caché
- Enviar datos a la CPU

Conceptos

La memoria tiene que ser transferida desde la memoria más grande para ser usada	
Cache	Memoria pequeña conectada al procesador
Block	Unidad de memoria transferida
Hit rate	fracción de las búsquedas de memoria servidas por los datos ya en caché
Miss rate	fracción de las búsquedas de memoria que requieren transferencias de memoria
Hit time	tiempo para procesar un acierto de la memoria caché
Miss penalty	tiempo para procesar un fallo en la memoria caché

```
#define size 32768
int matrix[size][size];

int main(void) {
   for(int i = 0; i<size; i++) {
      for(int j = 0; j<size; j++) {
         matrix[i][j] = 47;
      }
   }
return 0;
}
```

```
#define size 4096
int matrix[size][size];

int main(void) {
   for(int i = 0; i<size; i++) {
      for(int j = 0; j<size; j++) {
         matrix[i][j] = 47;
      }
   }
return 0;
}
```

Evaluación del rendimiento de la memoria

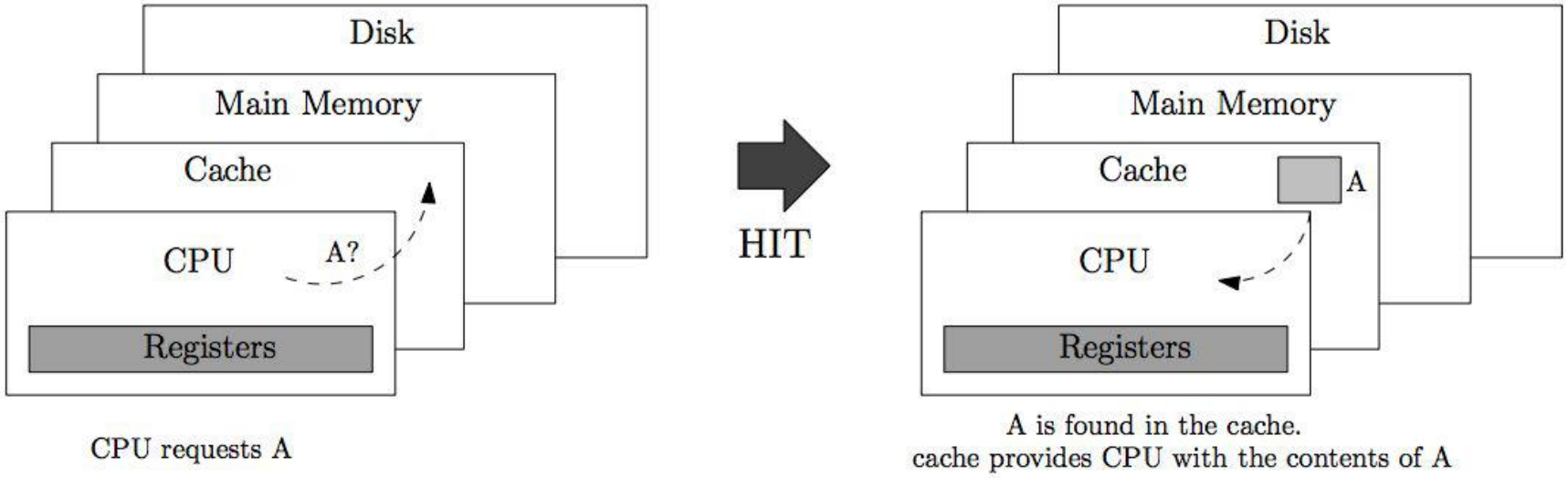
- Hit rate (***h***): (# referencias de memoria encontradas en el nivel superior) / (# referencias de memoria);
- Miss rate: 1 - ***h***
- Hit time (***Th***): tiempo para acceder al nivel superior (incluido el tiempo para determinar si el acceso es un **hit** o un **miss**);
- Miss penalty (***Tm***): tiempo para sustituir un bloque del nivel superior por el bloque correspondiente, recuperado de un nivel inferior, más el tiempo que el nivel superior tarda en entregar la información solicitada (que se encuentra en algún lugar del bloque) a la CPU.

$$\text{AMAT} = \boldsymbol{h} \times \boldsymbol{Th} + (1 - \boldsymbol{h}) \times \boldsymbol{Tm}$$

¿Preguntas?

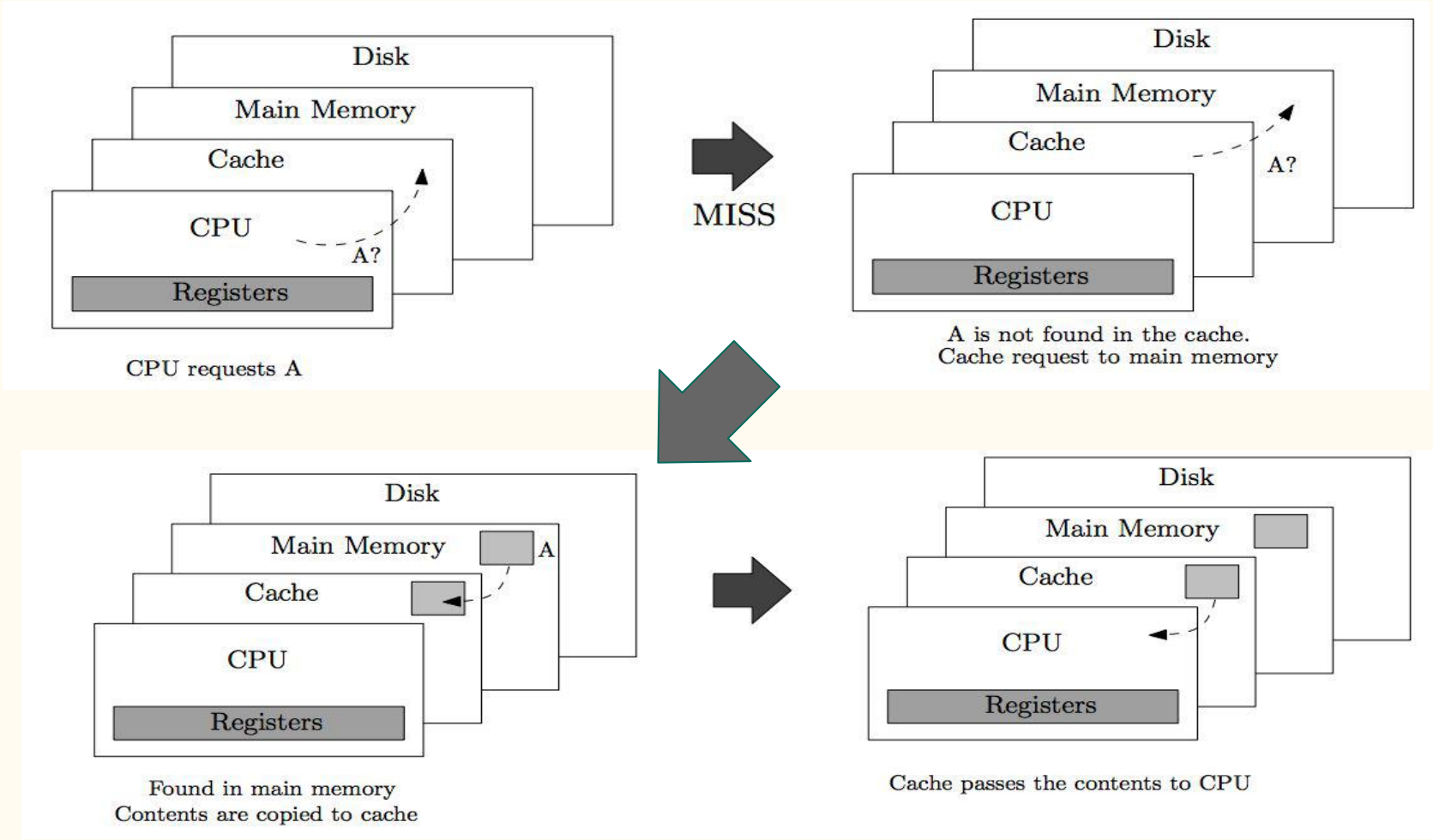
Cache

Cache es el nombre que se eligió para representar el nivel de la jerarquía de la memoria entre la CPU y la memoria principal. Cuando se diseña un cache, hay que tomar varias decisiones, incluyendo el tamaño del bloque, cómo el cache almacena la información, cómo manejar las escrituras, etc.



CPU requests A

A is found in the cache.
cache provides CPU with the contents of A


Disk
Main Memory
Cache
CPU
A?
Registers
CPU requests A
MISS
Disk
Main Memory
Cache
A?
CPU
Registers
A is not found in the cache.
Cache request to main memory
Disk
Main Memory
A
Cache
CPU
Registers
Found in main memory
Contents are copied to cache
Disk
Main Memory
Cache
CPU
Registers
Cache passes the contents to CPU

Compromisos del tamaño del bloque

Un tamaño de bloque más grande

- menos fallos de caché debido a la localización espacial
- tiempos de transferencia más largos de bloqueo
- menos bloques en el caché → más competencia por el caché

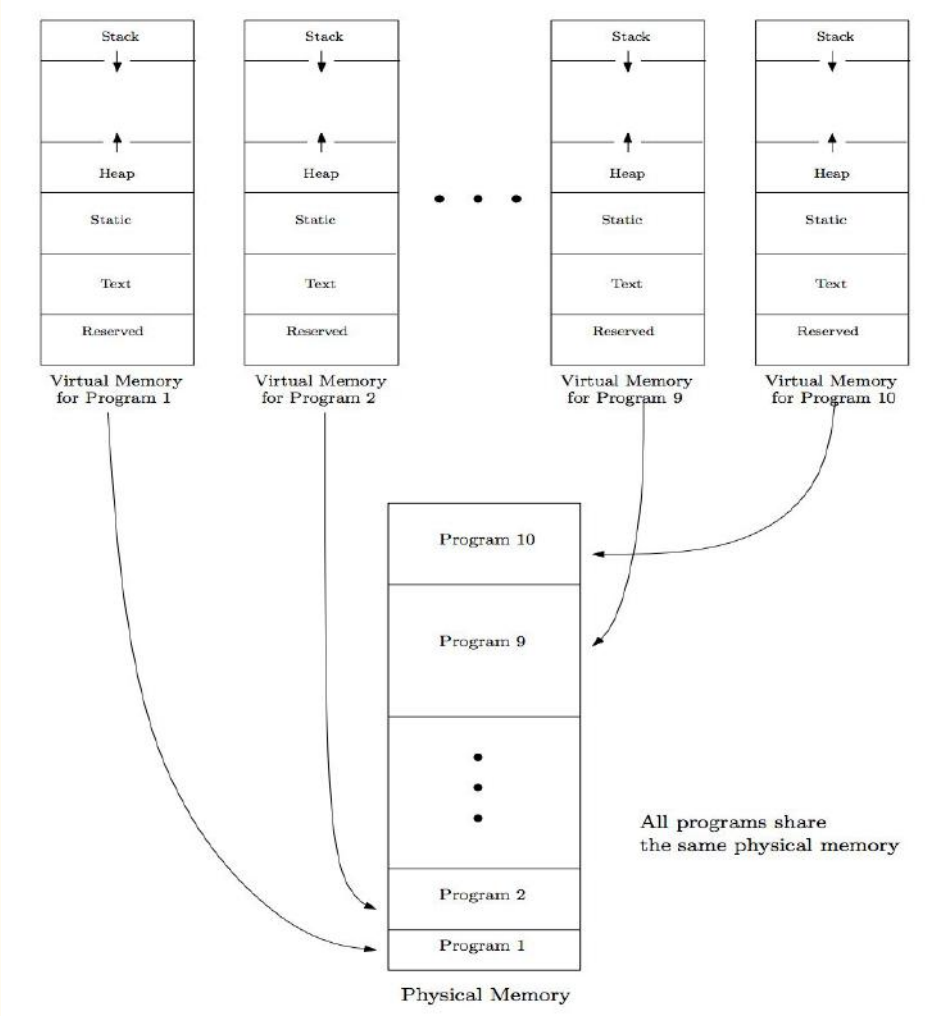
En la práctica

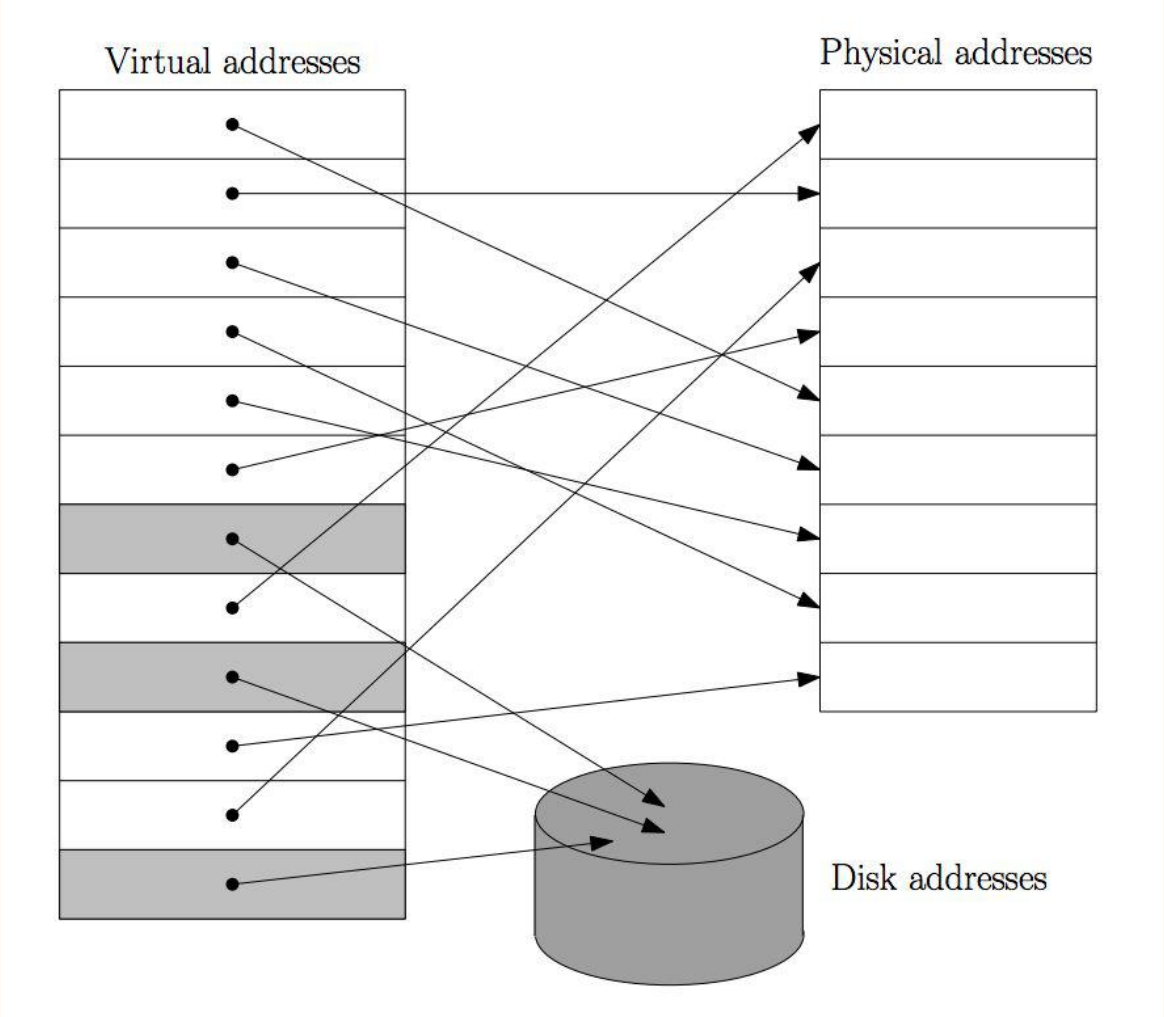
- valor óptimo en algún lugar del medio
- depende del proceso de ejecución

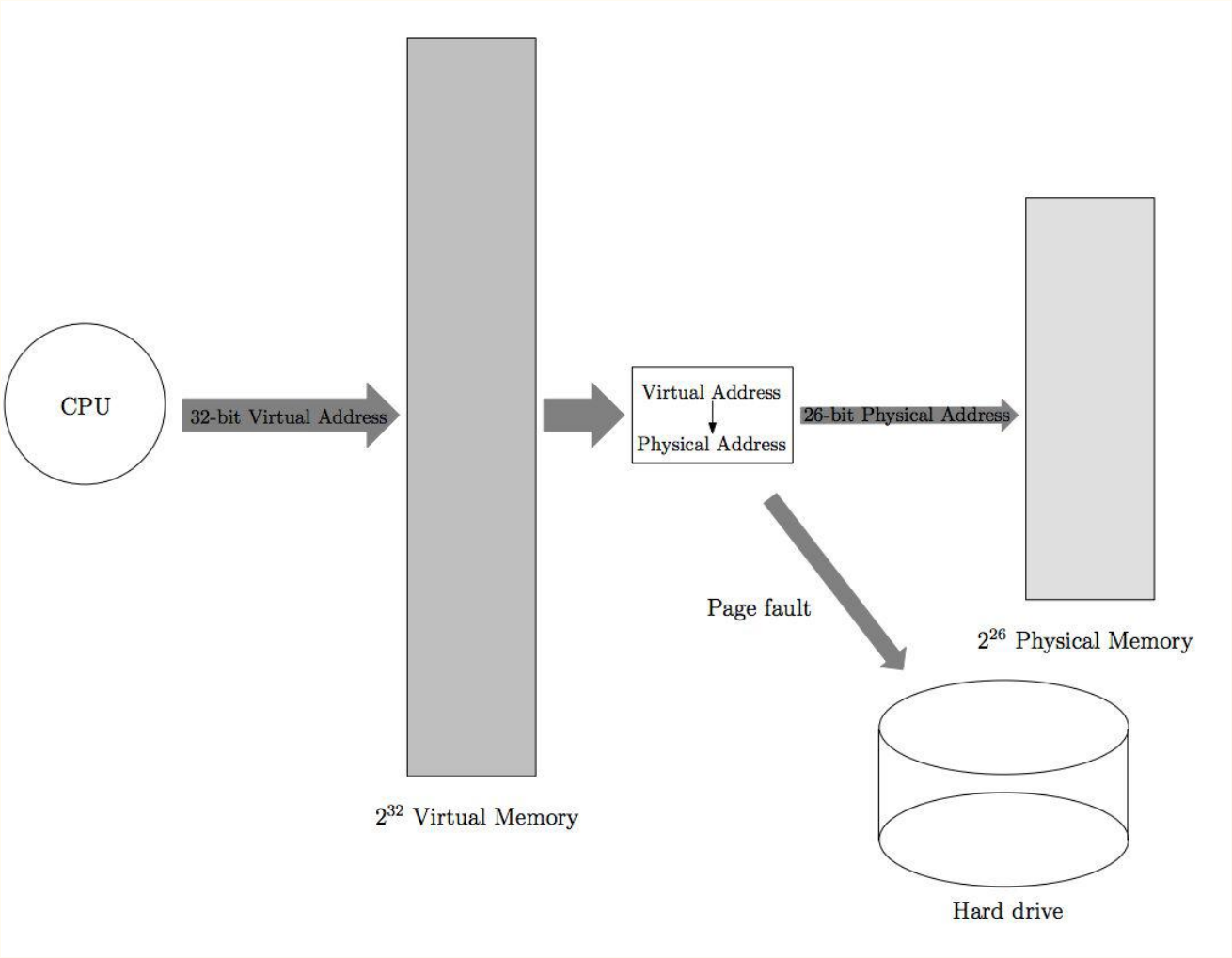
Memoria Virtual

- Eliminar las cargas de programación que surgen de tener sólo un pequeño y limitado cantidad de memoria principal disponible.

- Permitir un eficiente y seguro intercambio de memoria entre múltiples programas, cuando muchos de ellos se ejecutan simultáneamente.


Stack
Heap
Static
Text
Reserved
Virtual Memory for Program 1
Virtual Memory for Program 2
Virtual Memory for Program 9
Virtual Memory for Program 10
Program 10
Program 9
Program 2
Program 1
Physical Memory
All programs share the same physical memory


Virtual addresses
Physical addresses
Disk addresses


CPU
32-bit Virtual Address
2^{32} Virtual Memory
Virtual Address
Physical Address
26-bit Physical Address
2^{26} Physical Memory
Page fault
Hard drive

Referencias

- Hennessy J L and Patterson D. 2011. Computer Architecture, Fifth Edition: A Quantitative Approach 5th ed (San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.) ISBN 012383872X, 9780123838728.
- David B. Kirk and Wen-mei W. Hwu. 2010. Programming Massively Parallel Processors: A Hands-on Approach, Third Edition. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ©2010 ISBN:0123814723 9780123814722.
- Michael J. Quinn. 2003. Parallel Programming in C with MPI and OpenMP. McGraw-Hill Education Group.
- Victor Eijkhout. 2012. Introduction to High Performance Scientific Computing. Lulu.com.
- Michael McCool, James Reinders, and Arch Robison. 2012. Structured Parallel Programming: Patterns for Efficient Computation (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Tutorial online: https://www.tutorialspoint.com/parallel_computer_architecture/index.htm