

Less  
is  
More

Less is More

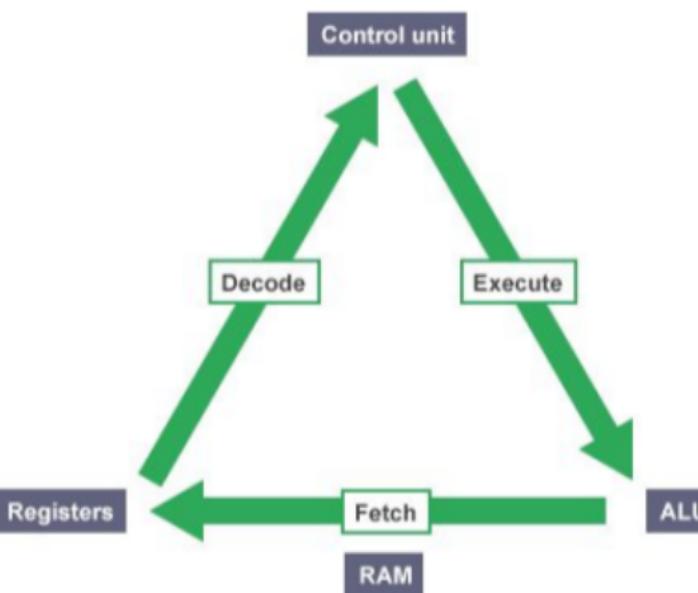
Fetch - obtiene instrucción

funciones del S.O

Decode - lee la instrucción

Exec - Ejecuta la operación

Jump → salta a la siguiente  
instrucción



1. administra eficientemente  
los recursos.

2. Controla la ejecución  
de programas

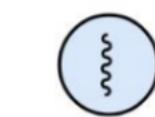
↓  
Previene errores

↓  
Controla permisos y  
accesos

3. Permite la virtualización  
de recursos.

Virtualización de recursos

Recursos Virtual



Proceso

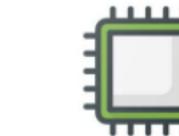


Espacio de  
direcciónamiento



Archivo

Recursos físicos



Procesador



Memoria



Disco

modos del SO

modo usuario

- No privilegiado

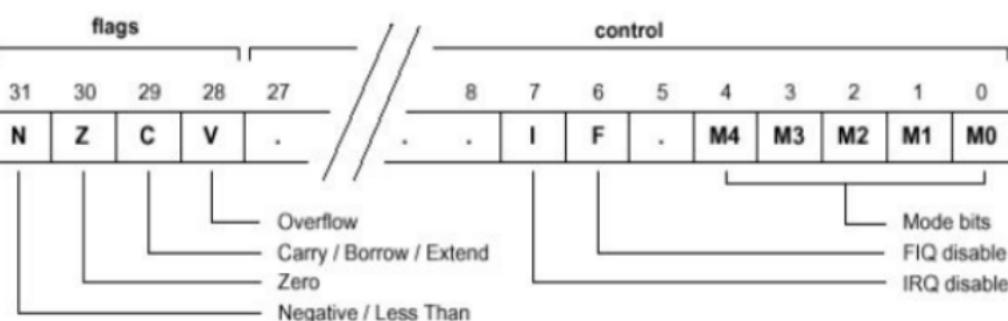
- Aplicaciones

Pasa de uno a otro

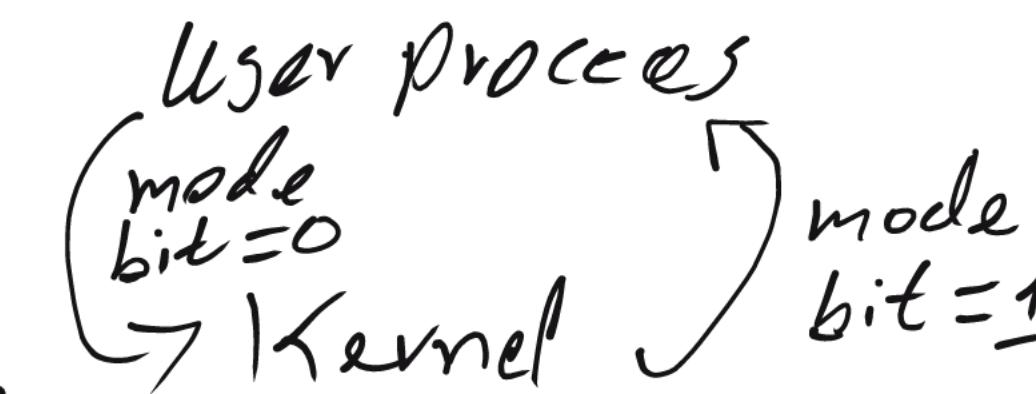
modo kernel Con bit de modo

- Privilegiado

- acceso al hardware



Los bits M0, M1, M2, M3 y M4 (M[4:0]) son los bits de modo y determinan el modo en el cual el procesador opera.



Proceso:

Un Programa en ejecución

Componentes

proceso = CPU + memoria + I/O info.

1. Estado del procesador

2. Espacio de direcciones

3. Información E/S

como se crea un proceso

1. Carga del Código

2. Asignación de memoria en Pila.

3. Creación del espacio de memoria Head

4. Tareas de inicialización.

5. Inicio del Programa ejecutando  
el punto de entrada.

modelo de los 3 estados

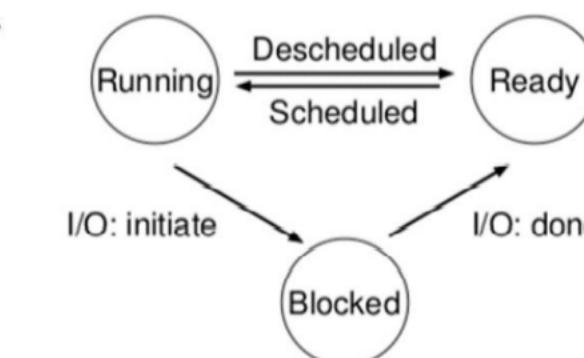
Ready → El Proceso está listo

Running → El proceso se está ejecutando

Blocked → El proceso desarrolla  
alguna operación (I/O disco)

Time	Process <sub>0</sub>	Process <sub>1</sub>	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	
4	Running	Ready	
5	-	Running	Process <sub>0</sub> now done
6	-	Running	
7	-	Running	
8	-	Running	Process <sub>1</sub> now done

Tracing Process State: CPU Only



# Estructuras de datos del S.O.

## Process list

listado de procesos en cada estado

## Register context

Contenido de los registros del Programa

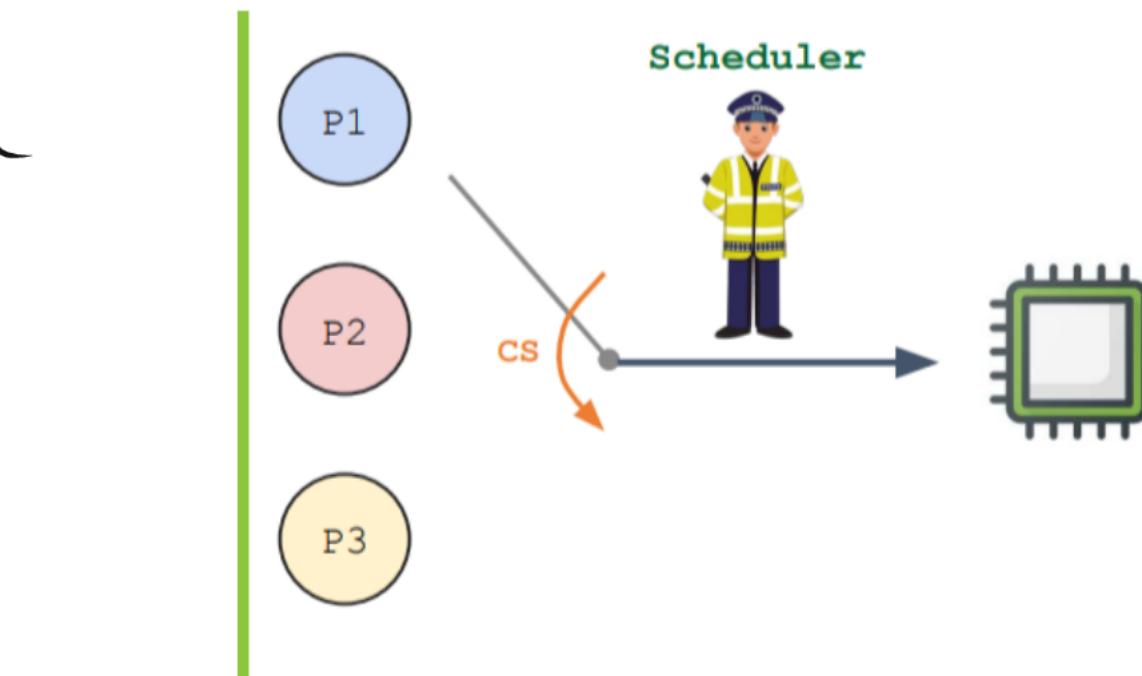
## Process Control Block

Información de cada Proceso.

# Multiprogramación

## Time sharing

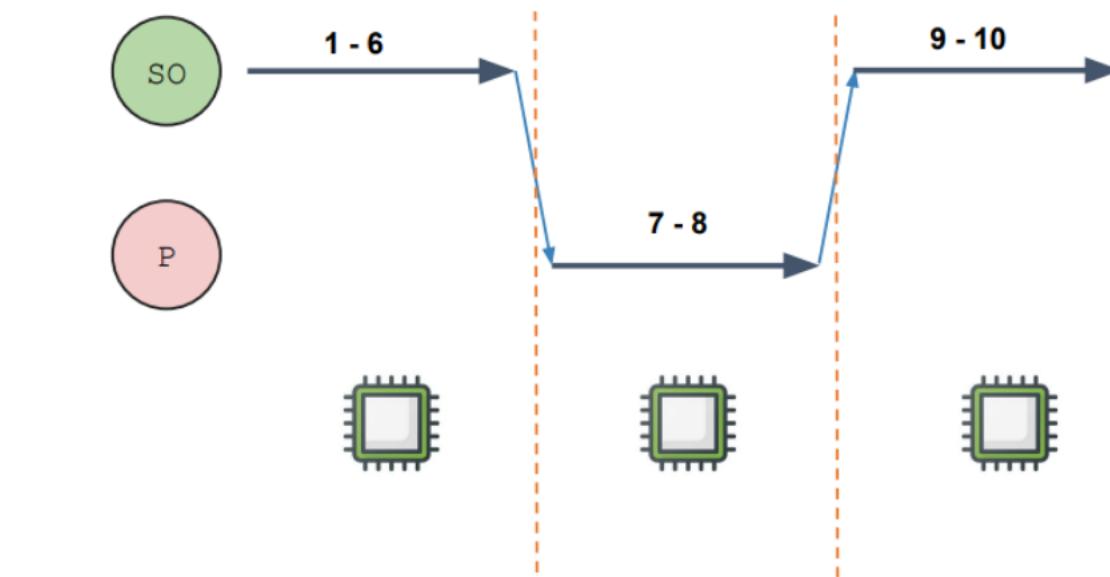
Tecnica empleada por los SO para Compartir CPU entre varios procesos



## Ejecución directa limitada

- Los Procesos deben estar aislados entre sí.
  - El Kernel debe estar aislado de los procesos
  - Los I/O deben estar aislados de los procesos.
  - Las aplicaciones deben ejecutarse en el procesador
- Ejecución directa
- Limitada

## Ejecución directa



Problema 1. Operación restringida

Transferencia de control

m. Kernel ↑  
↓ m. Usuario

## Problema 2: cambio entre procesos

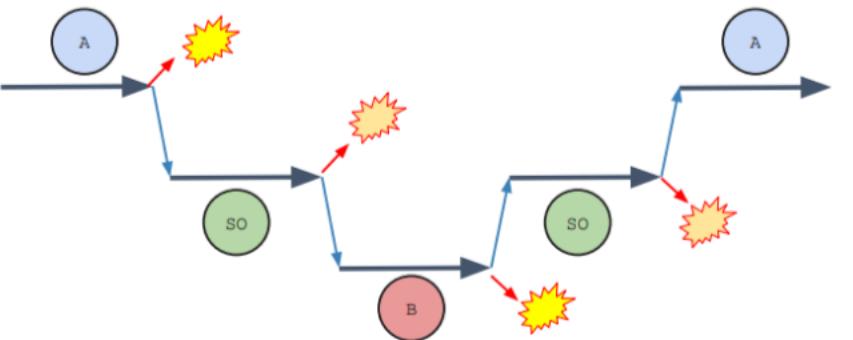
Propuesta cooperativa

↳ se cede el control

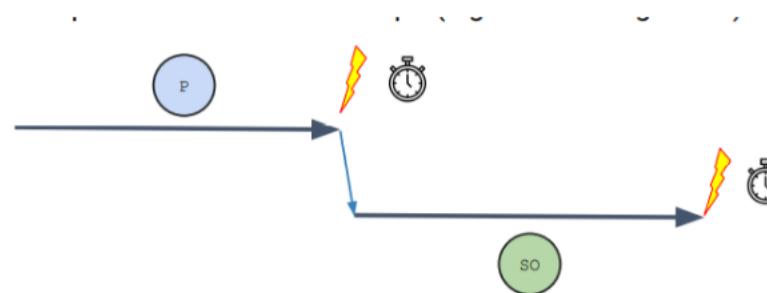
Propuesta no-cooperativa

↳ El SO toma el control.

Cooperativa



No-cooperativa



## Suposiciones de carga de trabajo

- se ejecutan por la misma cantidad de tiempo
- inician al mismo tiempo
- cada trabajo al iniciarse se ejecuta hasta finalizar
- se usa solo CPU y no I/O
- el runtime siempre es conocido.

Conceptos.

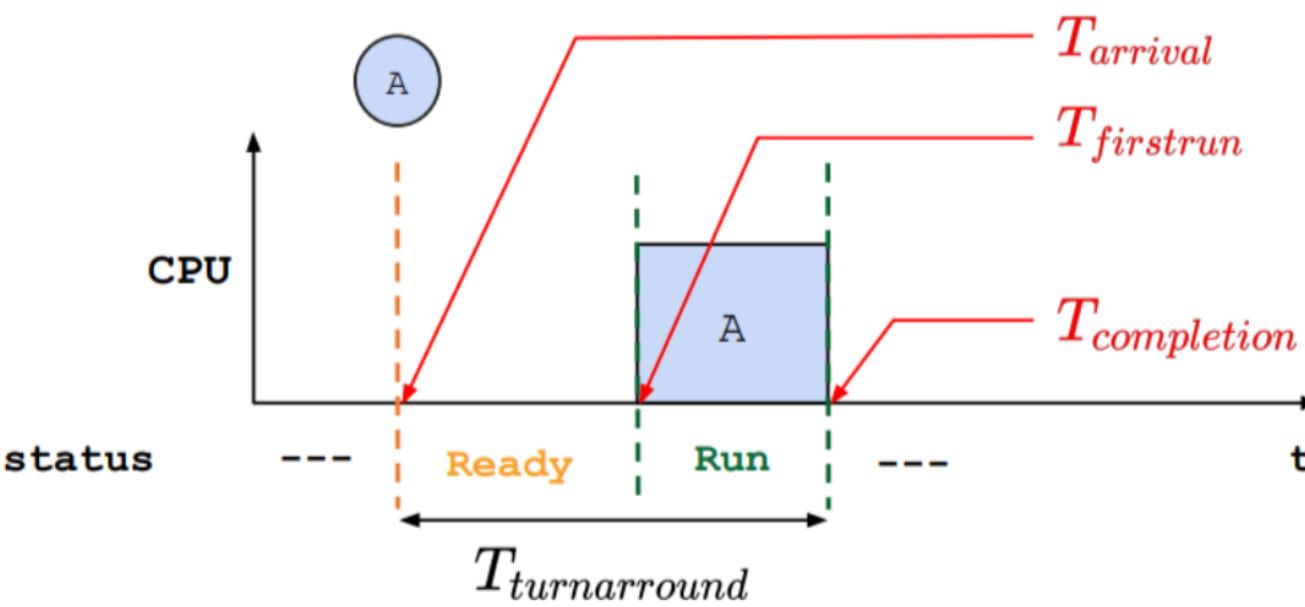
sheduler: decide como y cuando se usa la cpu.

workload: procesos en ejecución en el sistema

# Métricas usadas para scheduling

Turnaround Time:

Tiempo entre la llegada y la salida



$$T_{turnaround} = T_{completion} - T_{arrival}$$

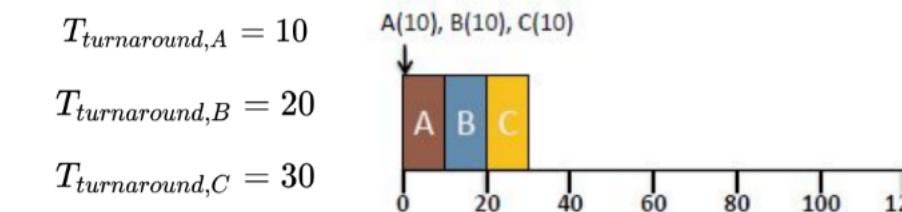
# Algoritmos de scheduling

FCFS

- Primero en llegar, primero en atenderse.
- Cola FIFO.
- solo se selecciona un nuevo proceso cuando el anterior acaba.

Ejemplo: Suponga que se tiene una situación en la que:

- o Tres procesos llegan a un sistema (A, B y C)
- o Llegan en el tiempo 0 en el orden: A - B - C
- o Cada proceso se ejecuta por 10 segundos.



$$T_{turnaround,avg} = \frac{10+20+30}{3} = 20$$

## Efecto Convoy

- se da cuando un trabajo con mucho tiempo de ejecución va por delante de procesos más ligeros



## SJF

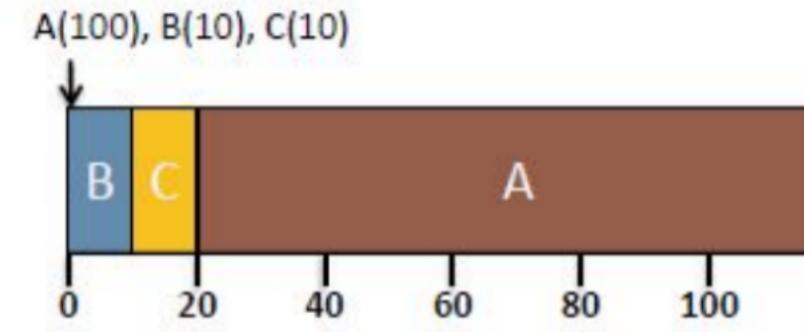
- El scheduler ejecuta primero los procesos mas cortos

- **Ejemplo:** Suponga que se tiene una situación en la que:
  - Tres procesos llegan a un sistema (A, B y C)
  - Llegan en el tiempo 0 en el orden: A - B - C
  - A se ejecuta por 100 seg, B y C por 10 seg cada uno.

$$T_{turnaround,A} = 10$$

$$T_{turnaround,B} = 20$$

$$T_{turnaround,C} = 120$$



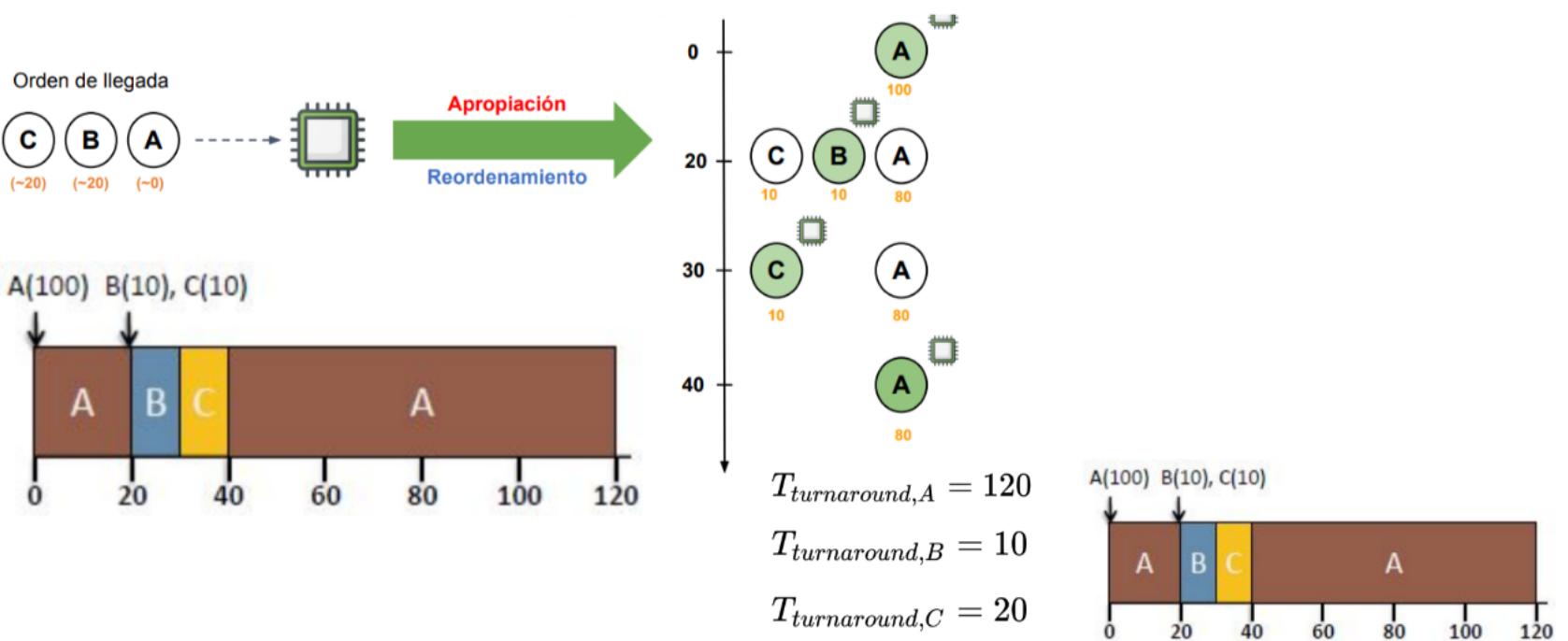
$$T_{turnaround,avg} = \frac{10+20+120}{3} = 50$$

# STCF

- si un nuevo trabajo entra y tiene menor tiempo de ejecución entonces expropia al que está ejecutando y corre el de menor tiempo

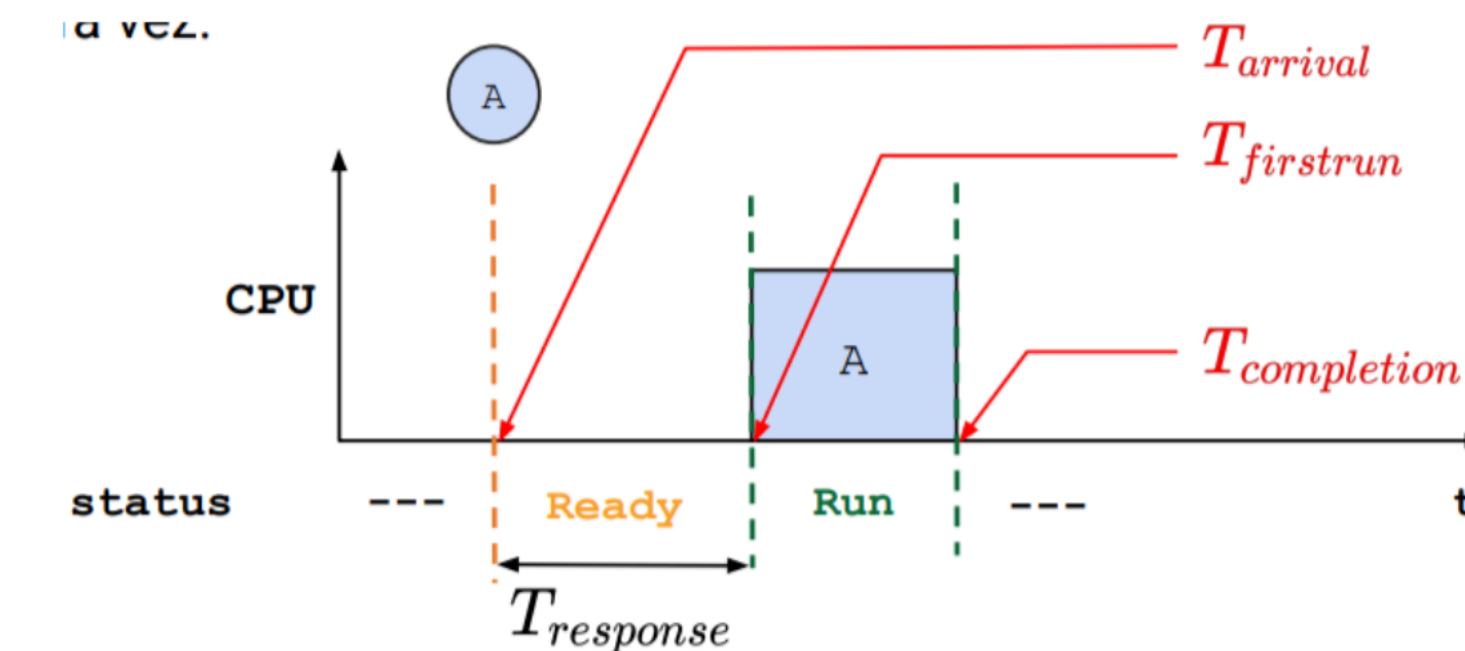
- **Ejemplo:** Suponga que se tiene una situación en la que:

- Tres procesos llegan a un sistema (A, B y C)
- A llega en  $t = 0$  y B y C llegan en  $t = 20$ .
- A se ejecuta por 100 seg, B y C por 10 seg cada uno.



# Response Time

Tiempo medido desde que el trabajo llega hasta que se programa por primera vez



$$T_{response} = T_{firstrun} - T_{arrival}$$

## Round Robin

Consiste en ejecutar procesos en porciones de tiempo (**Quantum**) e ir intercalandolos hasta que finalicen.

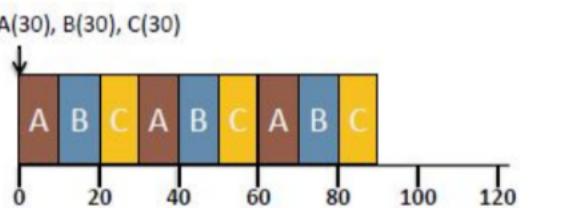
Suponga que se tiene una situación en la que:

- Tres procesos llegan a un sistema (A, B y C)
- Llegan en el tiempo 0 en el orden: A - B - C
- El tiempo de ejecución de cada uno de los procesos es de 30 seg.

Calcule el turnaround time y el response time empleando el **Algoritmo RR** con un quantum de 10 seg.

**Algoritmo RR (quantum = 10)**

Proceso	Arrival time	Run-time
A	0	30
B	0	30
C	0	30

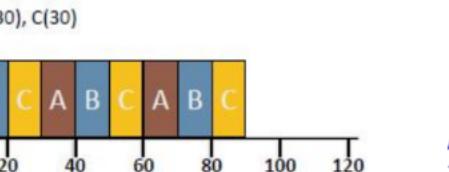


Proceso	Arrival time	Run-time
A	0	30
B	0	30
C	0	30

Tiempo	Ready queue	CPU
0	C	A
10	A C	B
20	B A	C
30	C B	A
40	A C	B
50	B A	C
60	C B	A
70	---	B
80	---	C
90	---	---

**Algoritmo RR (quantum = 10)**

Proceso	Arrival time	Run-time
A	0	30
B	0	30
C	0	30



$$T_{turnaround,A} = 70$$

$$T_{turnaround,B} = 80$$

$$T_{turnaround,C} = 90$$

$$T_{turnaround,avg} = \frac{70+80+90}{3} = 80$$

$$T_{response,A} = 0$$

$$T_{response,B} = 10$$

$$T_{response,C} = 20$$

$$T_{response,avg} = \frac{0+10+20}{3} = 10$$

## Multilevel feedback queue

si prioridad  $a > b$  se ejecuta a.

si Prioridad  $a=b$ , RR Para ambos

Nuevo proceso  $\rightarrow$  cola más alta

después de un tiempo los procesos

Pasan a cola de alta prioridad

Cuando un proceso termina

su control pasa a cola de

baja prioridad.

# Hardware Address Translation (AT)

Mecanismo donde se transforma la memoria virtual por la física

Virtual address → AT → Physical address

## Address Space

abstracción de la memoria

FÍSICA.

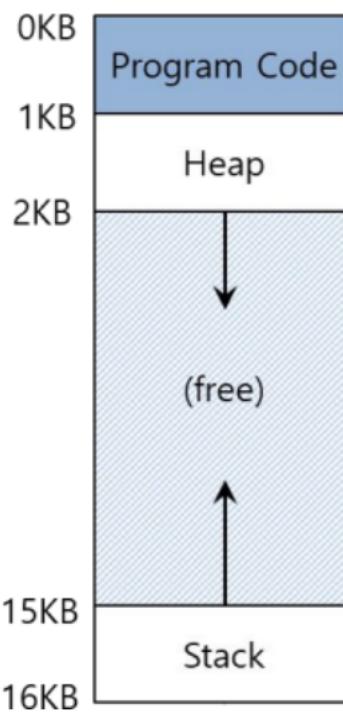
se compone:

Program code

Data

Heap

Stack



### Segmentos del espacio de direcciones

Program Code	<ul style="list-style-type: none"><li>Instrucciones</li><li>Variables globales</li></ul>
Heap	<ul style="list-style-type: none"><li>Memoria <b>asignada dinámicamente</b><ul style="list-style-type: none"><li>malloc en C</li><li>new en Java o C++</li></ul></li></ul>
Stack	<ul style="list-style-type: none"><li>Memoria <b>automática</b></li><li>Direcciones y valores de retorno</li><li>Variables locales y argumentos pasados a rutinas</li></ul>

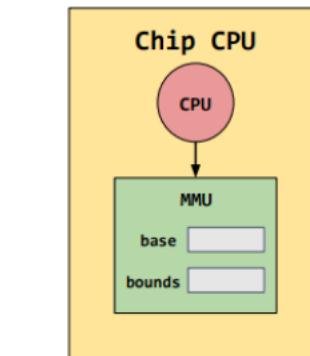
## Dynamic Reallocation

hace uso de los registros:

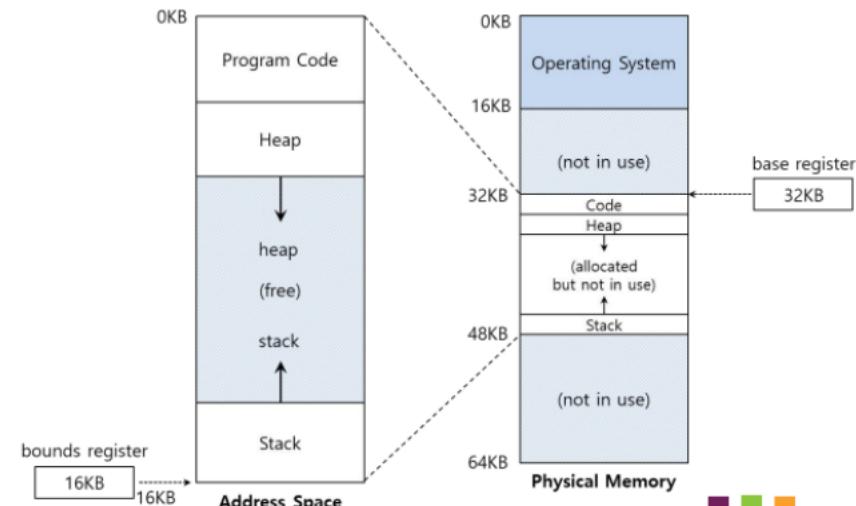
**base:** Almacena la dirección física más pequeña.

**bounds:** Almacena el tamaño del segmento de memoria virtual asociada al proceso.

### MMU (Memory Management Unit)



**MMU:** Hardware encargado de realizar la traducción de direcciones lógicas a físicas.



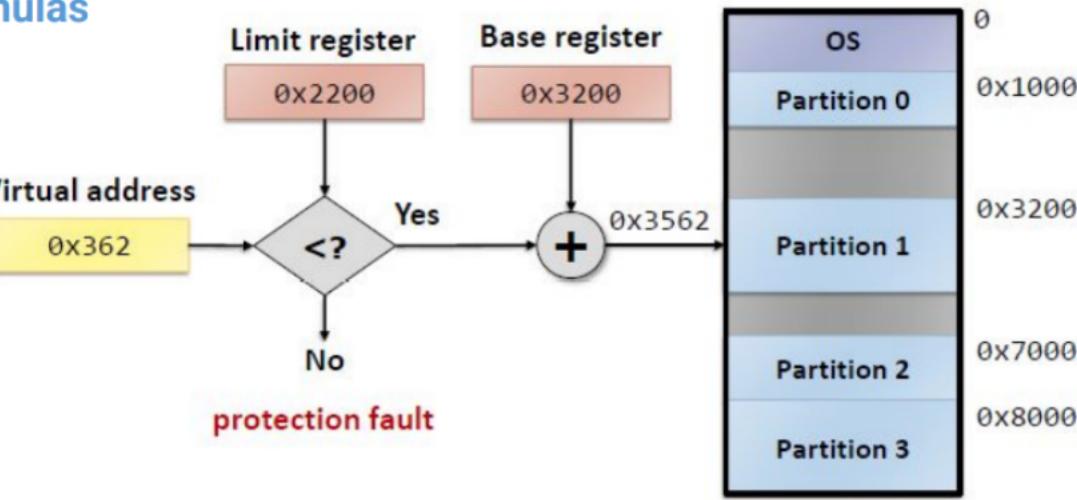
cuando el proceso es creado, el SO decide donde sera cargada.

$$PA = VA + \text{base}$$

Las direcciones virtuales no deben ser mayores al valor de registros bounds ni negativa.

$$0 \leq VA < \text{bounds}$$

Fórmulas



$$0 \leq VA < \text{bounds}$$

$$PA = VA + \text{base}$$

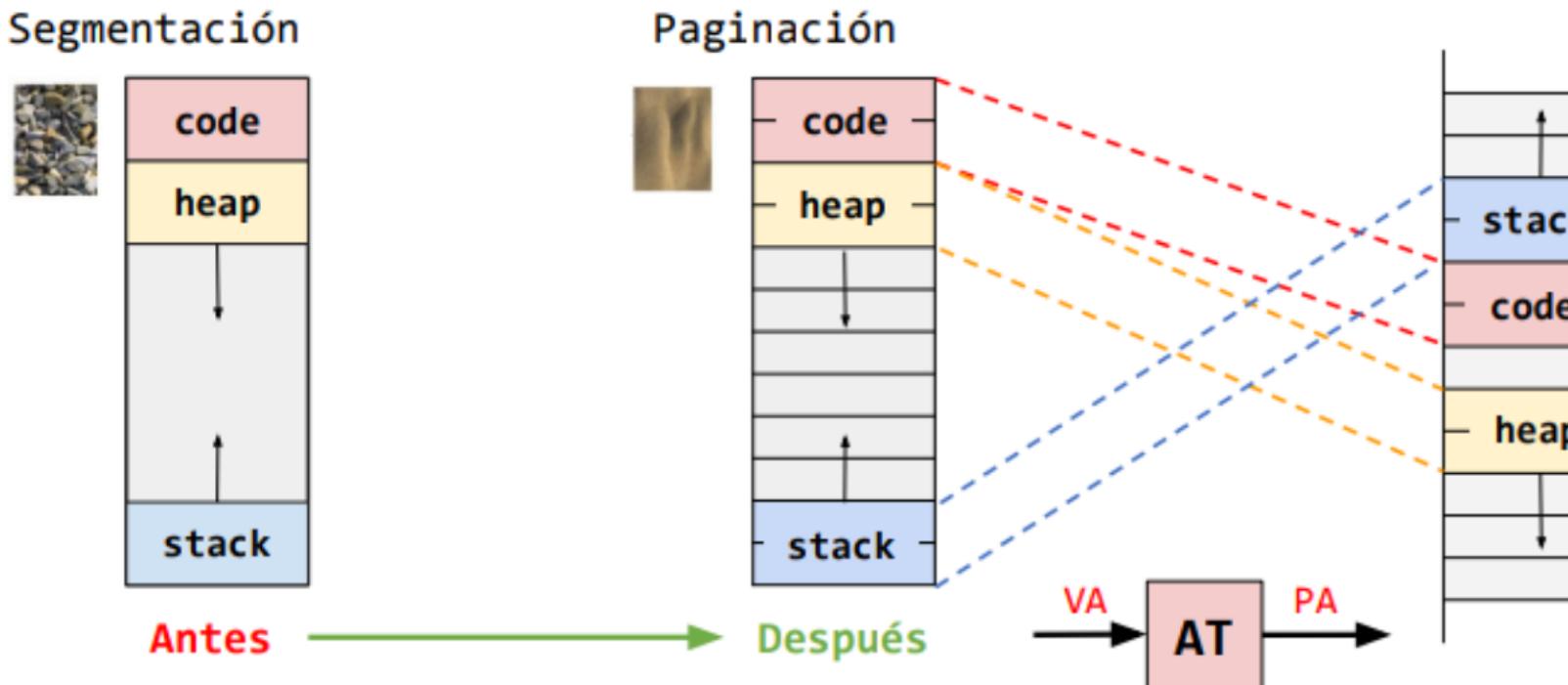
El SO interviene en el base y bound

- un proceso inicia  $\rightarrow$  busca espacio
- un proceso termina  $\rightarrow$  Recupera espacio
- Ocurre un cambio almacena y restaura de contexto.  $\rightarrow$  los registros en el PCB

Free list: lista de rangos de espacio libre en memoria.

# Segmentación : división grande

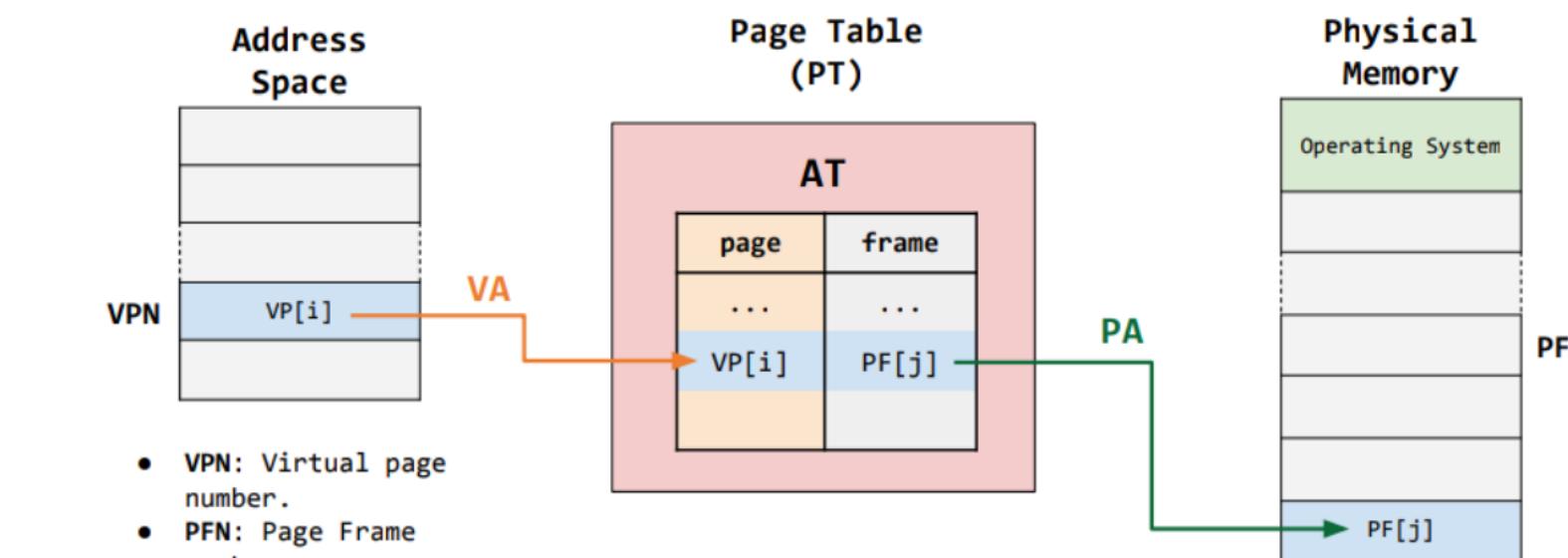
Paginación: división final y fija  
por páginas



Page: bloque de memoria V.  
de tamaño fijo

Frame: bloque de memoria f. asociado a una page.

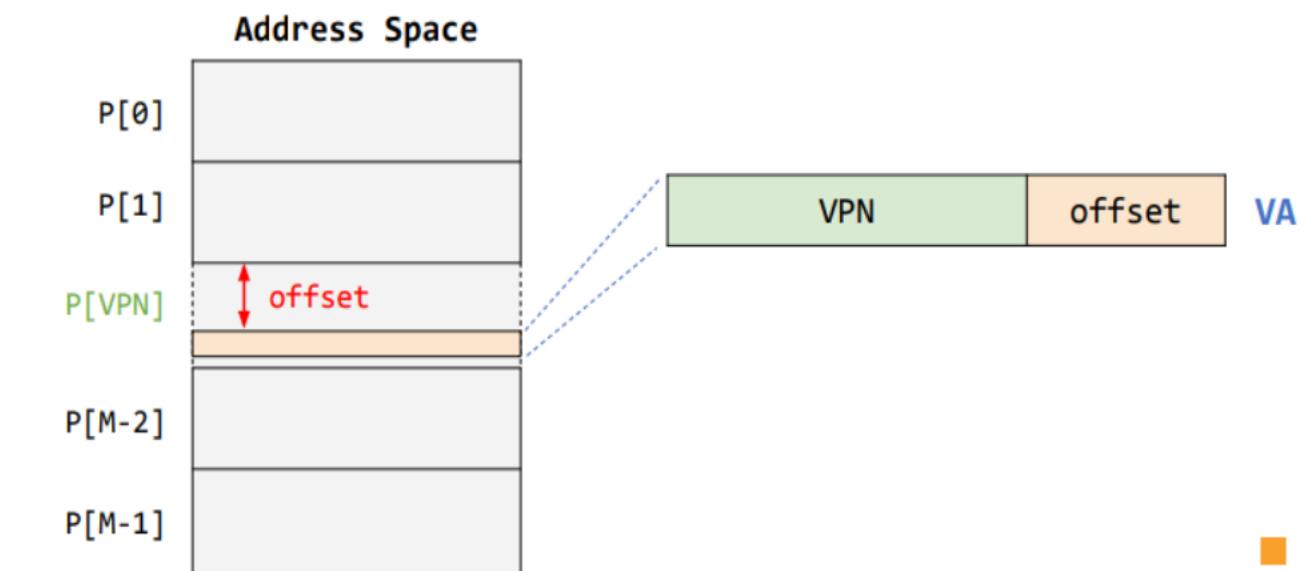
Table. permite la AT haciendo un mapping en M.V. de la M.F.



# Direcciones Virtuales

VPN: Índice asociado a una página en particular.

offset: Desplazamiento dentro de la página seleccionada.

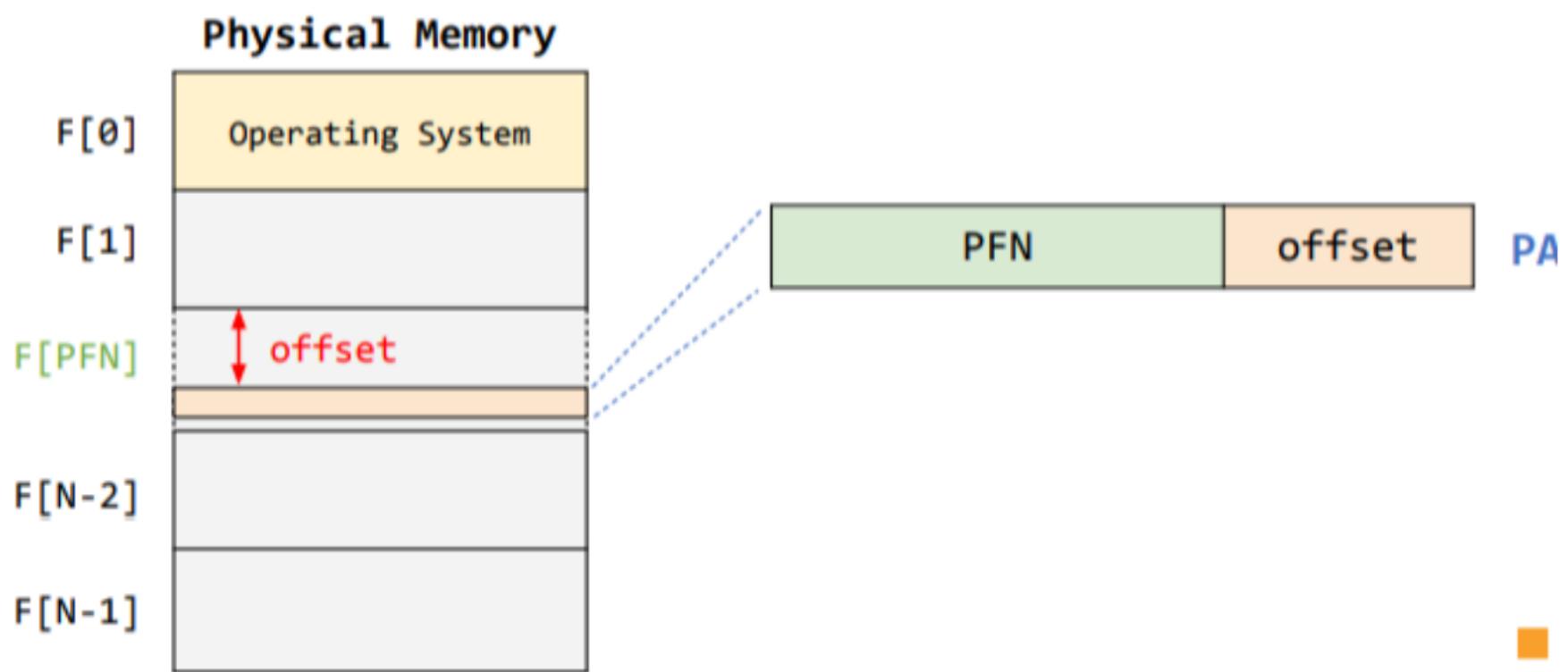


## Direcciones físicas

PFN: frame dentro del que se encuentra la F.A.

offset: Desplazamiento dentro del frame.

offset. Desplazamiento dentro del frame seleccionado.



Pages,frames =  $\frac{\text{numero de Paginas} \times \text{frames}}{\text{size(address space)} / \text{size(page)}}$

bits dirección virtual

$\log_2$  (posiciones de memoria) = m

bits necesarios del offset

$\log_2$  (bytes de las Páginas) = n

bits del numero de páginas

V.A - offset = m-n

# Tabla de páginas:

Usada para el mapeo de VA a FA

Bit de validez → indica si la traducción es válida ✓

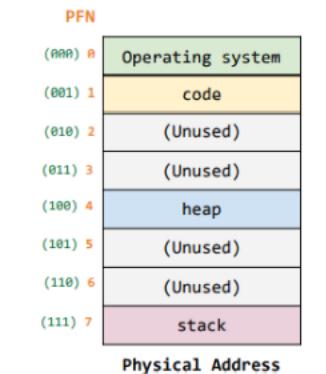
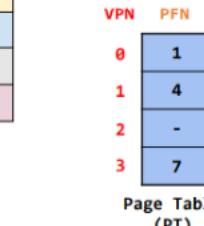
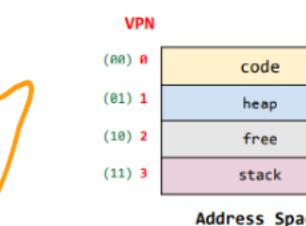
Bit de protección → read-write-execute <sup>prot</sup>

Bit de Presencia → indica si la página está en M. f

Bit sucio → indica si la pag se ha modificado <sup>M</sup>

Bit Referencia → indica si la pag ha sido accedida. <sup>R</sup>

VR | PI | M | Prot | PFN



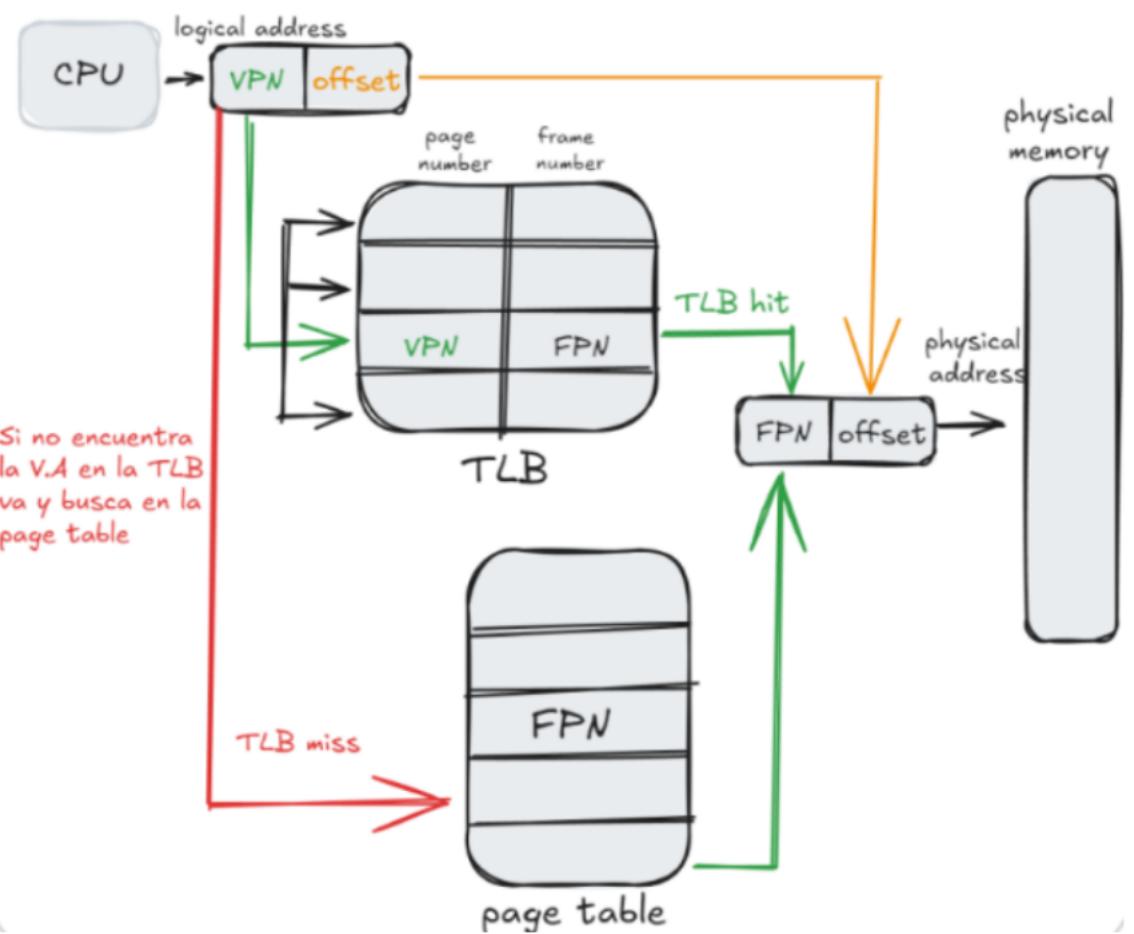
VA 0 1 1 0 1 0



PA 1 0 0 1 0 1 0

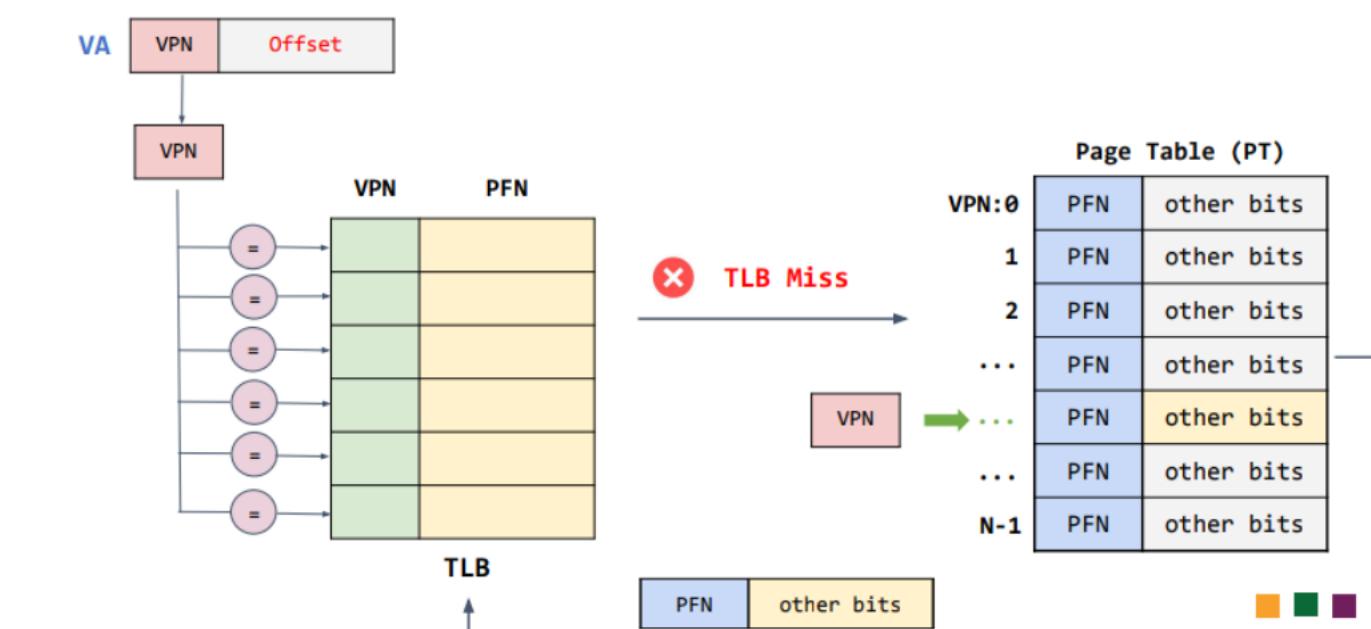
# TLB

Es una memoria cache para acelerar el proceso de AT y hace parte del MMU. Tamaños típicos de 16-256 entradas.



# Política de reemplazo

Consiste en el proceso por el cual se cambia una entrada en la fib por una nueva.



LRU

## Ejemplo:

Asuma que se tiene una TLB de tres entradas, y que el acceso a memoria (principal) se está realizando de acuerdo al orden presentado en la siguiente traza:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1

Reference row →													
7	0	1	2	0	3	0	4	2	3	0	3	2	1
Page Frame →													
7	7	7	2	2	2	2	4	4	4	0	0	0	1
0	0	0	0	0	0	0	0	0	3	3	3	3	3
		1	1	1	3	3	3	2	2	2	2	2	2
M	M	M	M	H	M	H	M	M	M	M	H	H	M

# Random

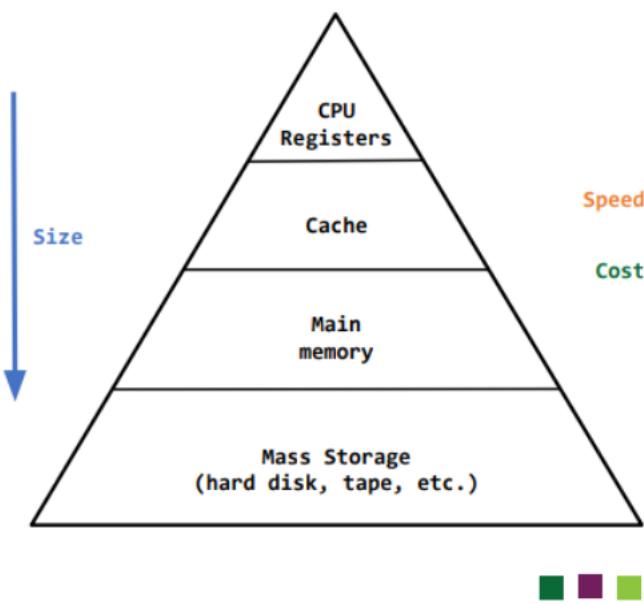
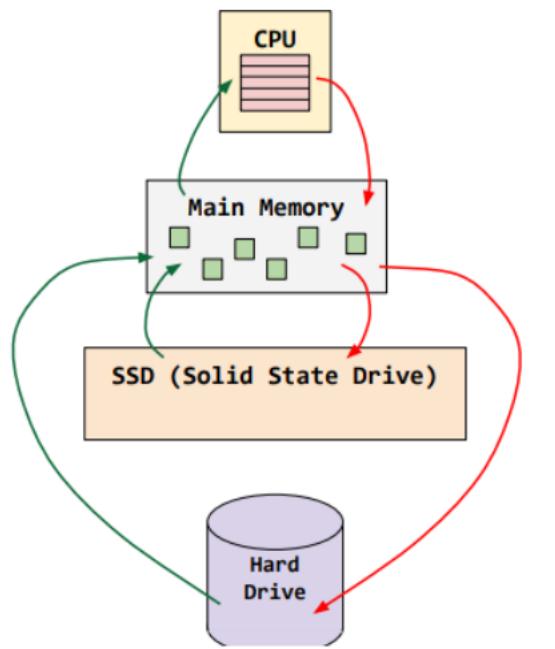
## Ejemplo

Asuma que se tiene una TLB de tres entradas, y que el acceso a memoria (principal) se está realizando de acuerdo al orden presentado en la siguiente traza:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1

Page Frame →														
7	0	1	2	0	3	0	4	2	3	0	3	2	1	
Reference row →														
7	7	7	7	7	3	3	4	4	4	4	4	4	4	
0	0	1	2	2	2	2	0	2	3	3	3	3	2	
M	M	M	M	H	M	H	M	H	M	H	H	H	M	
0	2	1	1	0		0	0	1		0	0	0	0	2

# SWAPPING



Cada copia aplica Backing store

1. Ejecución del programa
2. Se crean las referencias
3. se accede a la librería
4. se mueve o copia la paga la m. ppal.

se necesita un bit adicional,  
bit de Presencia.

- 1: la página esta en M.F.
- 0: la página no esta en M.F  
pero si en disco.

Métrica AMAT = P<sub>hit</sub>\*T<sub>M</sub> + P<sub>miss</sub>\*T<sub>D</sub>

OPT → Caso Ideal

FIFO → Reemplaza la Primer Página en entrar a M.

RANDOM → Reemplaza aleatoriamente la Página.

LRU → Reemplaza la pgj menos reciente usada.