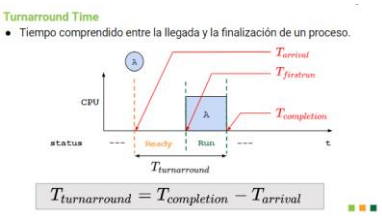
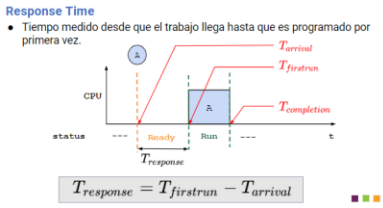
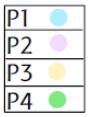
Políticas

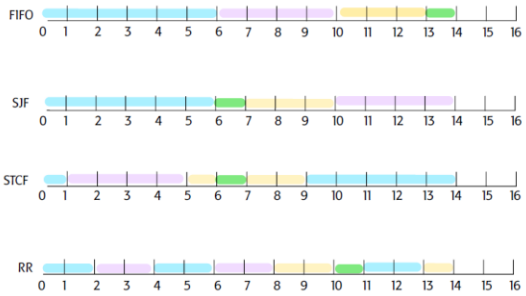




- SJF: Es óptimo si todos los procesos llegan al mismo tiempo.
- STCF: Es óptimo s i no se considera el tiempo de respuesta e interactividad.
- RR: optimiza el tiempo de respuesta.
- SJF y STCF no tienen interrupciones como RR, por lo que ejecutan un programa hasta que finalice.
- SJF y STCF optimizan tiempo de ejecución.

Process	Arrival time	Execution time
P1	0	6
P2	1	4
P3	5	3
P4	6	1





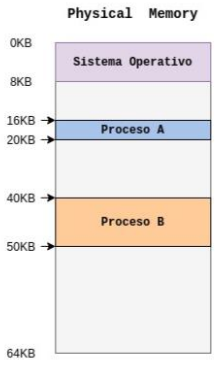
MLFQ

- Regla 1: si prioridad(A) > prioridad(B), entonces se ejecuta A.
- Regla 2: Si prioridad(A) = prioridad(B), ambas se ejecutan en RR.
- Asume que todo proceso que llega tiene un corto tiempo de ejecución (es de alta prioridad) y a medida que pasa el tiempo, se reduce su nivel de prioridad en las colas para evitar que si es un proceso que consume gran tiempo de la CPU, se aproveche de ella.
- Si un proceso ejecuta operaciones I/O, permanece en el mismo nivel de prioridad mientras sucede la interrupción, siempre y cuando no haya completado su tiempo de asignación.
- Después de un período de tiempo s, rodos los procesos del sistema se llevan a la cola de más alta prioridad. Evitar efecto de STARVATION.

Procesos tipo Batch: Alto consumo de CPU. Pausas cortas. Poca interacción con el usuario.

Procesos interactivos: Bajo consumo de CPU. Largas pausas (bloqueo I/O). Interacción directa con el usuario

Virtualización de memoria



1. ¿Cuál es el tamaño del Address Space para cada uno de los procesos mostrados?

Segun la figura el tamaño es:

Proceso	Tamaño
A	(20K - 16K) = 4K = 4(2^10) = 4096
B	(50K - 40K) = 10K = 10(2^10) = 10240

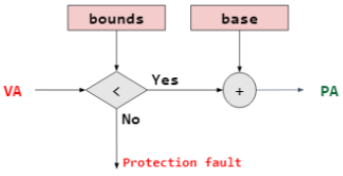
Proceso	Registro Base	Registro Bound
A	16K = 16(2^10) = 16384	4K = 4096
B	40K = 40(2^10) = 40960	10K = 10240

Verificación de límites

```
0 <= VA < bounds
```

Traducción de direcciones

```
PA = VA + base
```



Proceso	Virtual Address	Physical Address
A	3200	19584
B	6400	47360
A	0	16384
A	4096	Protection Fault
B	2K	43008
B	5120	45K
A	120	16504

Paginación

$$\#pages = \frac{size(AS)}{size(page)}$$

$$\#pages = \frac{size(AS)}{size(frame)}$$

$$formato\ direcciones\ virtuales = \log_2(size(AS))$$

$$formato\ direcciones\ físicas = \log_2(size(PM))$$

$$offset = \log_2 size((page))$$

$$offset = \log_2 size((frame))$$

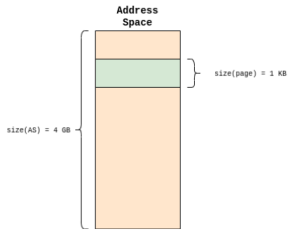
$$VPN = \log_2(\#pages)$$

TLB

Un sistema de cómputo cuenta con una memoria física de 256MB (2^{28} bytes) y un espacio de direccionamiento virtual de 4 GB (2^{32} bytes). Este sistema implementa un manejo de memoria por paginación con páginas de 1 KB (2^{10} bytes) y una TLB totalmente asociativa con 32 entradas.

a. A continuación se muestra el formato de las direcciones físicas y virtuales según los datos dados:

Dirección virtual:



Datos:
- size(AS) = 4 GB = 2^32 B
- size(page) = 1 KB = 2^10 B

De la información dada tenemos que:

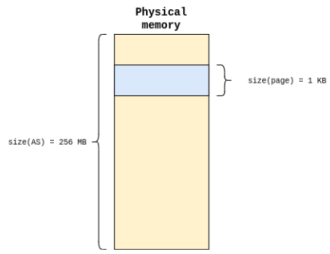
- n = log2(size(AS))
  = log2(2^32) =
  = 32
- n(offset) = log2(size(page))
  = log2(2^10)
  = 10
- n(VPN) = n - n(offset)
  = 32 - 10
  = 22

Teniendo en cuenta la información anterior, tenemos que:

```
pages = size(AS)/size(page) = 2^32/2^10 = 2^22 pages
```

De este modo la forma del address space es el siguiente:

Dirección física

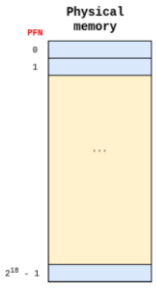


Datos:
- size(PM) = 256 MB = 2^28 B
- size(frame) = 1 KB = 2^10 B

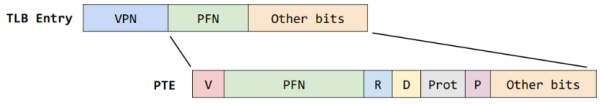
De la información dada tenemos que:

- n = log2(size(PM))
  = log2(2^28) =
  = 28
- n(offset) = log2(size(page))
  = log2(2^10)
  = 10
- n(PFN) = n - n(offset)
  = 28 - 10
  = 18

```
frames = size(PM)/size(frame) = 2^28/2^10 = 2^18 frames
```



b. ¿Cuál sería el tamaño mínimo de la TLB (en bytes) si cada entrada incluye 8 bits adicionales para las banderas (validez, ASID, ...)?



Teniendo en cuenta la información dada, el tamaño de una entrada de la TLB está dada por:

```
n = n(VPN) + n(PFN) + n(other) = 22 + 18 + 8 = 48 bits = 6B
```

c. ¿Cuál sería el tamaño mínimo de la tabla de página lineal si cada entrada de la tabla de página tiene un tamaño de 4 bytes?

```
size(PT) = pages*size(PTE) = (2^22)(4) = (2^22)(2^2) = 2^24 = (2^4)(2^20) B = 16 MB
```

$$size(PT) = pages * size(PTE)$$

Traducción de direcciones usando TLB

VPN y offset equivalentes en hexadecimal

En el siguiente caso, la VPN es de 20 BITS y el Offset es de 12 BITS. Al pasar a hexadecimal, los primeros 5 BITS son la VPN y los siguientes 3 BITS el Offset.

Caso	Virtual Address
i	0000 0001 0000 1111 1111 0101 0000 1111 = 0x010FF50F
j	0000 0001 0000 1111 1111 0101 0000 1111 = 0x010FF50F

Teniendo en cuenta una VPN de 20 BITS, un Offset de 12 Bits y un PFN de 20 Bits, se tiene la siguiente TLB (en hexadecimal)

VPN	PFN	PID
00000	00FFF	00
00000	00AAB	01
00010	F000A	00
010FF	00ABC	01

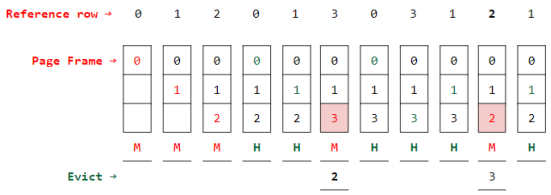
Luego, se pueden establecer los HITS y MISS que se tendrán para los siguientes procesos.

Caso	PID	Virtual Address (VA)	Hit/Miss	VPN	PFN	Physical Address (PA)
a	00	0x00000000	H	0x00000	0x00FFF	0x00FFF000
b	01	0x00000000	H	0x00000	0x00AAB	0x00AAB000
c	00	0xFF00FFAA	M	0xFF00F	---	---
d	00	0x0010FFAA	M	0x0010F	---	---
e	01	0x0010FFAA	M	0x0010F	---	---
f	00	0x000000FF	H	0x00000	0x00FFF	0x00FFF0FF
g	01	0x00000FAB	H	0x00000	0x00AAB	0x00AABFAB
h	00	0x010FFFFF	M	0x010FF	---	---
i	01	0x010FF50F	H	0x010FF	0x00ABC	0x00ABC50F
j	00	0x010FF50F	M	0x010FF	---	---
k	02	0x00000000	M	0x00000	---	---

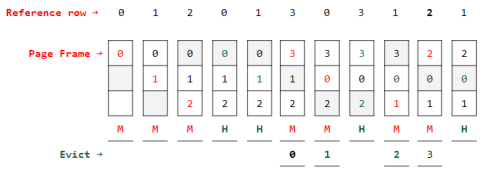
Políticas de reemplazo

Definen los criterios para seleccionar cuál página debe salir de memoria.

- **Política de reemplazo óptima: reemplaza la página que se accederá más lejos en el futuro.**



- **Política de reemplazo FIFO: Reemplaza la página que ha entrado primero. Ocurre la anomalía de Belady (Se espera que el número de MISS aumente cuando se aumenta la memoria).**



- **Política de reemplazo aleatoria: Reemplaza de manera aleatoria la página.**
- **Política LRU: Se elimina la página que ha sido menos recientemente usada en el pasado.**

La política aleatoria es útil en ciclos secuenciales.

Conceptos

- Heap: Memoria asignada dinámicamente (cuando usamos new)
- Stack: Guarda registro de direcciones, declaración de variables locales.
- Page: Bloque de memoria virtual de tamaño fijo.
- Frame: Bloque de memoria física de tamaño fijo, el cual se encuentra asociado a una página de memoria virtual
- Page table: Permite la traducción de direcciones virtuales a direcciones físicas.
- Offset: Desplazamiento dentro de la página seleccionada

$$2^{10} \rightarrow KB$$

$$2^{20} \rightarrow MB$$

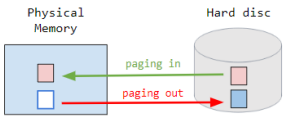
$$2^{30} \rightarrow GB$$

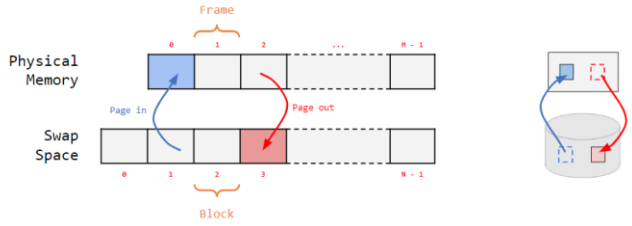
- Base: Almacena la dirección física más pequeña.
- Bounds: Almacena el tamaño del segmento de manera virtual asociada al proceso.
- TLB: Memoria caché para acelerar el proceso de traducción de direcciones. Mantiene los valores más usados de la traducción de direcciones.
- Si se encuentra en la dirección lógica, ocurre un **TLB HIT.**
- **Si no se encuentra en la dirección lógica ocurre un TLB MISS.**
- Cada vez que se cambia de VPN, se ejecuta un TLB MISS inicialmente.
- ASID: Se utiliza para gestionar el cambio de contexto de la TLB y diferenciar el VPN correspondiente del proceso que se está ejecutando.
- Bit PRESET (P): Indica si el dato está en memoria física. Si se encuentra es un HIT, de lo contrario, es un MISS.
- Si un proceso es interrumpido, se tiene que guardar el contexto de dónde quedó PCB (Process control Block, almacena el estado del proceso).
- Modo usuario: las aplicaciones no tienen acceso total a los recursos de HW.
- Modo kernel: El SO tiene acceso a todos los recursos, acceso a instrucciones privilegiadas.

- Llamadas al sistema: Las aplicaciones le solicitan al SO realizar alguna a ctividad (debida al software).
- Interrupciones: por ejemplo, acciones del teclado (debido al HW).
- Instrucción Trap: Permite al usuario cambiar al modo kernel.
- Instrucción return from Trap: Permite retornar al modo usuario.
- Bit de validez: indica si una PTE (Page Table Entry) puede ser usada.
- **Propuesta cooperativa**: Los procesos liberan la CPU periódicamente. El cambio de contexto se da empleando llamadas al sistema
- **Propuesta no cooperativa:** El SO toma el control de la CPU mediante interrupciones por timer.
- **Acceso limitado:** Los procesos deben estar aislados entre sí. El kernel del SO debe estar aislado e los procesos.
- **Ejecución directa:** Las aplicaciones deben ejecutarse directamente en el procesador. El SO no interviene.

Operaciones:

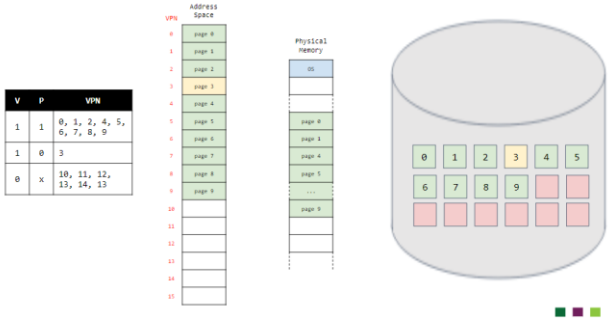
1. swap in = page in
2. swap out = page out



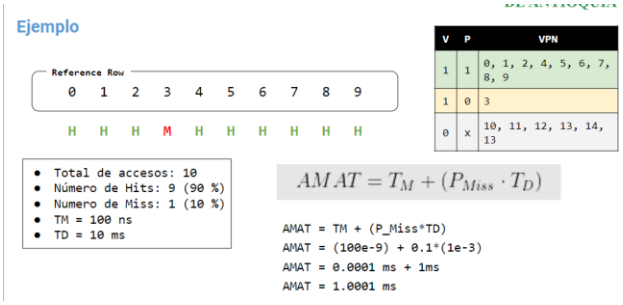


De modo que la expresión para el **tiempo promedio de acceso a memoria** quedará como:

$$AMAT = T_M + (P_{Miss} \cdot T_D)$$



Ejemplo



- Total de accesos: 10
- Número de Hits: 9 (90 %)
- Numero de Miss: 1 (10 %)
- TM = 100 ns
- TD = 10 ms

$$AMAT = T_M + (P_{Miss} \cdot T_D)$$

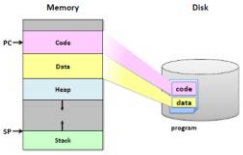
```
AMAT = TM + (P_Miss*TD)
AMAT = (100e-9) + 0.1*(1e-3)
AMAT = 0.0001 ms + 1ms
AMAT = 1.0001 ms
```

¿Como se crea un proceso?

UNIVERSIDAD DE ANTIOQUIA

Pasos

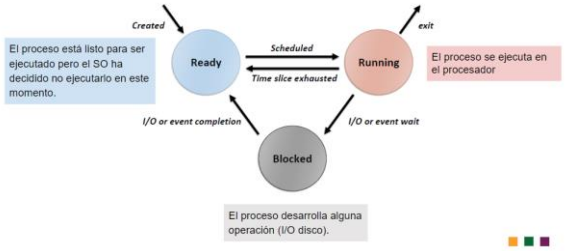
1. Carga (**load**) del código del programa a memoria en forma diferida (lazy loading).
2. Asignación de memoria en la pila en tiempo de ejecución:
   - **Pila**: variables locales, parámetros de funciones, direcciones de retorno.
   - **Inicializa la pila con argumentos**: argc y argv de la función main.
3. Creación del espacio de memoria **heap**:
   - Memoria dinámica (C: malloc/free; Java: new).



Estados de un proceso

UNIVERSIDAD DE ANTIOQUIA

- Modelo de tres estados



3. Teniendo en cuenta la tabla de página mostrada.
   - ¿Cual es el tamaño de una de sus entradas?
   - ¿Cual es el tamaño de la tabla de página entera?

Page Table (PT)

	V	PFN
0	1	1
1	1	4
2	0	-
3	1	7

4 bits

```
size(PTE) = size(bits) + size(PFN)
size(PTE) = size(V) + size(PFN)
size(PTE) = 1 + 3
size(PTE) = 4 bits

size(PT) = num_pages*size(PTE)
size(PTE) = 4*4 bits
size(PTE) = 16 bits
```

Address Translation usando TLB

UNIVERSIDAD DE ANTIOQUIA