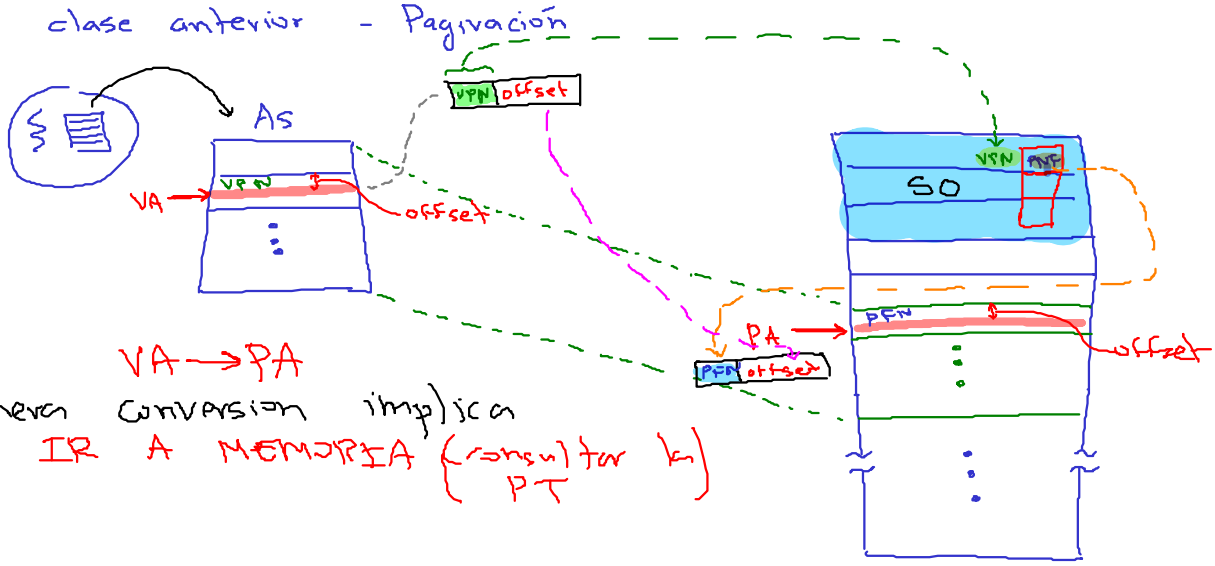
25/09/2025 - Sistemas Operativos (Ude@)
30

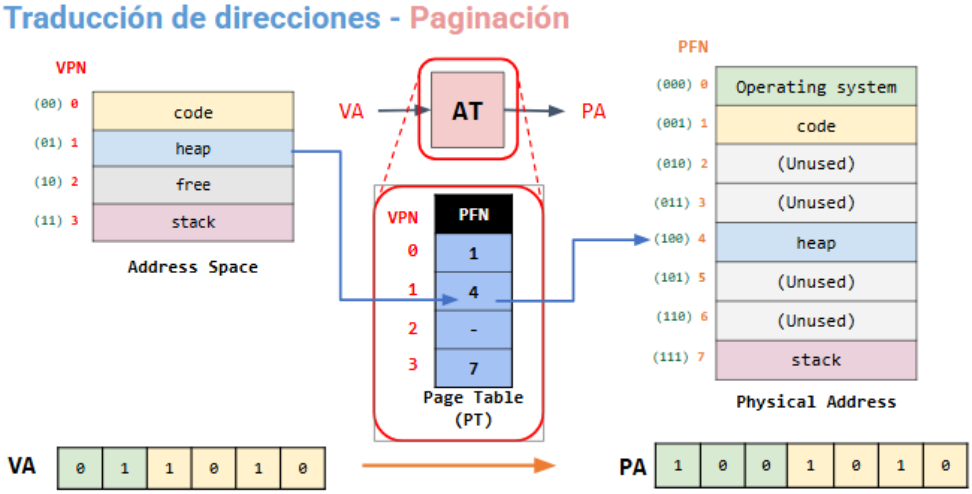
Avisos:
1. Quiz semana 5 (cierra 07/10/2025)
2. Taller seguimiento opcional (0.4 parcial)

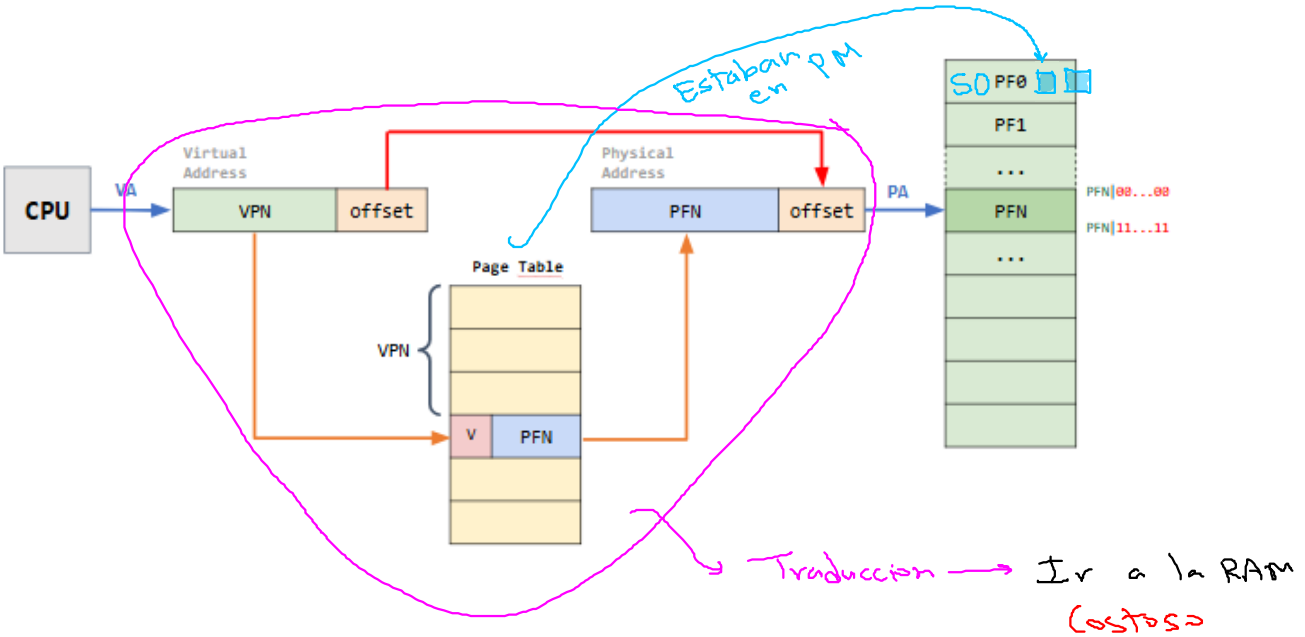
1. Repaso clase anterior - Paginación



VA → PA

La mera conversion implica IR A MEMORIA (consultar la PT)





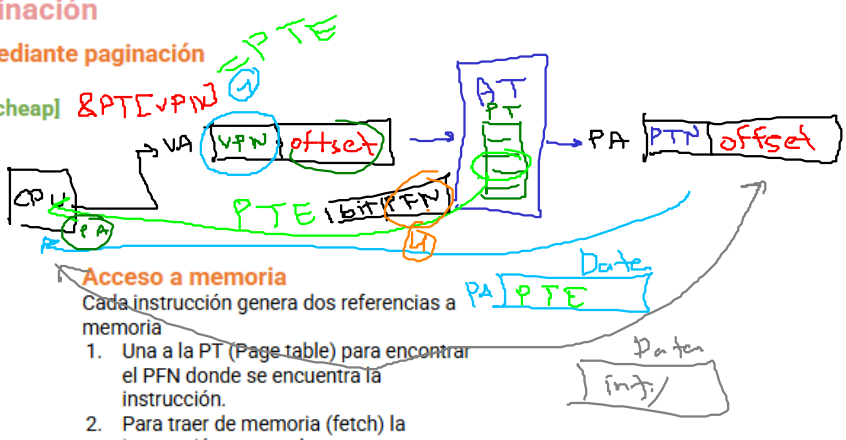
Traduccion → Ir a la RAM

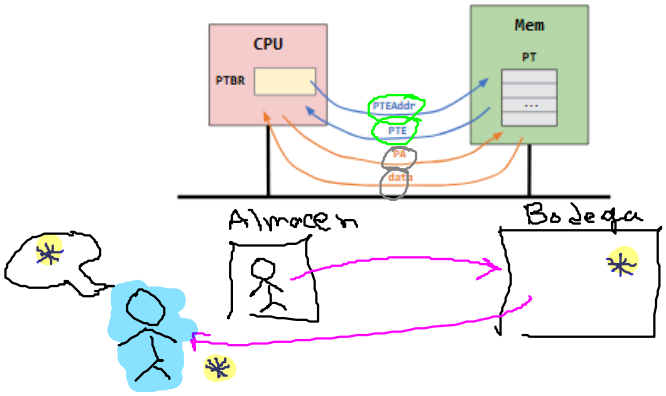
Costoso

Traducción de direcciones - Paginación

Pasos de la traducción de direcciones mediante paginación

1. Extraer el VPN de la VA [cheap]
2. Calcular la dirección del PTE (PTEAddr) [cheap]
3. Obtener (Fetch) el PTE [cost]
4. Extraer el PFN [cheap]
5. Construir la PA [cheap]
6. Traer el PA a un registro [cost]





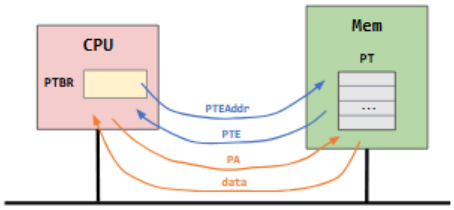
Acceso a memoria

Cada instrucción genera dos referencias a memoria

1. Una a la PT (Page table) para encontrar el PFN donde se encuentra la instrucción.
2. Para traer de memoria (fetch) la instrucción como tal.

Traducción de direcciones - Problemas de la paginación

1. Lentitud en el proceso debido a los accesos a la tabla de página.
2. Gasto de memoria (cada proceso tiene su propia tabla de página).



Código

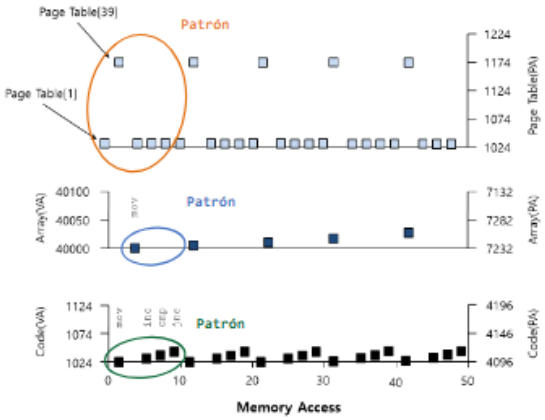
```
int array[1000];
...
for (i = 0; i < 1000; i++)
  array[i] = 0;
```

Compilación y ejecución

```
prompt> gcc -o array array.c -Wall -o
prompt>./array
```

Código ensamblador resultante

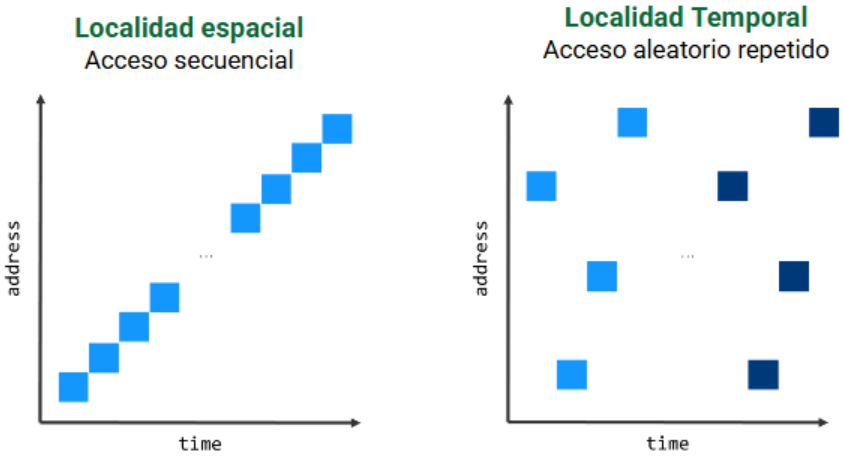
```
0x1024 movl $0x0,(%edi,%eax,4)
0x1028 incl %eax
0x102c cmpl $0x03e8,%eax
0x1030 jne 0x1024
```



5 ciclos * 10 accesos/1 ciclo = **50 accesos**

Instrucción	Fetch inst	Mem Access
`movl $0x0,(%edi,%eax,4)`	2	2
`incl %eax`	2	
`cmpl $0x03e8,%eax`	2	
`jne 0x1024`	2	

Traducción de direcciones - Trazas de memoria

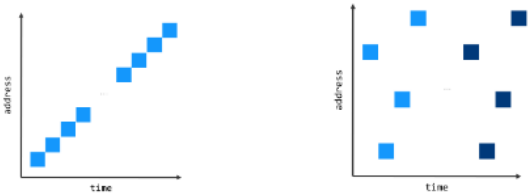


2. TLB – Contextualización

Ejemplo: Bodegas.

¿Cómo mejorar la traducción de direcciones?

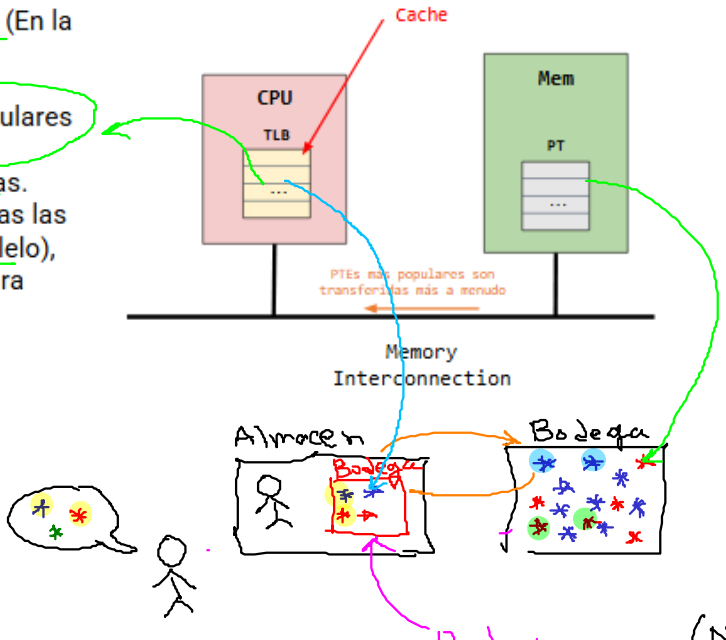
- ¿Cómo podemos acelerar la traducción de direcciones evitando la referencia adicional a memoria necesaria cuando se usa paginación?
- ¿Que soporte de hardware es necesario?
- ¿Qué es lo que tiene que hacer el sistema operativo en este caso?

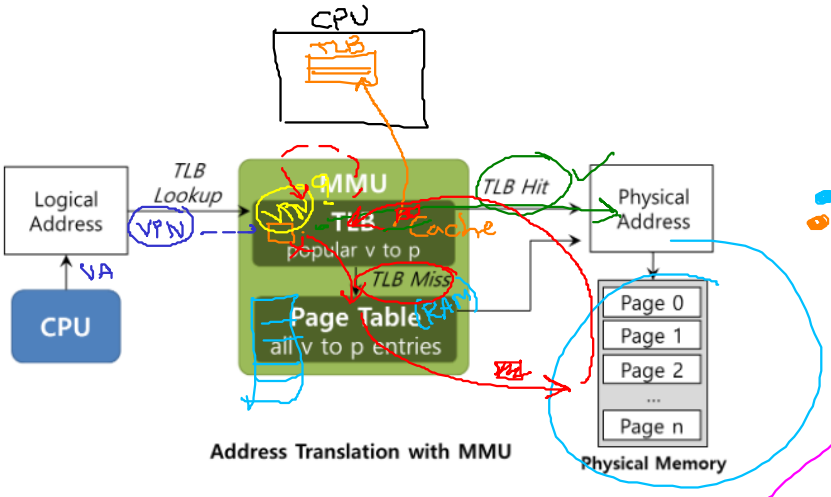


a. Que es una TLB?

Aspectos claves

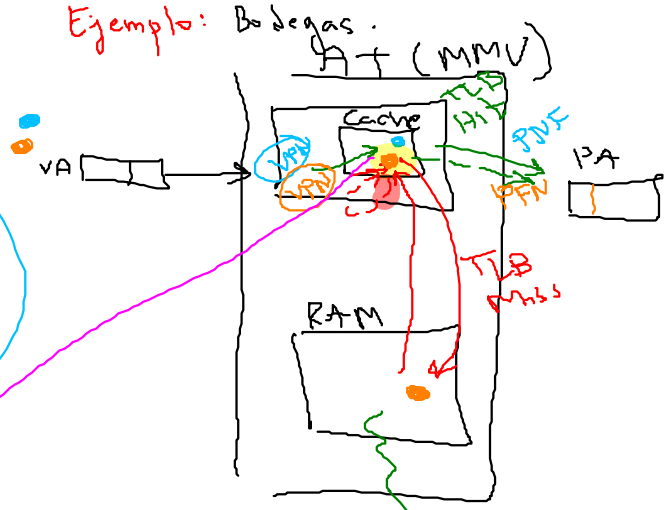
- La **TLB** es una memoria cache on chip (En la CPU).
  - Pequeña y rápida.
  - Mantiene las direcciones más populares de la tabla de página (PT).
  - Tamaños típicos: 16 ~ 256 entradas.
  - Usualmente es full asociativo (todas las entradas son comparadas en paralelo), pero puede haber asociatividad para reducir la latencia.





Address Translation with MMU

Ejemplo: Bodegas.



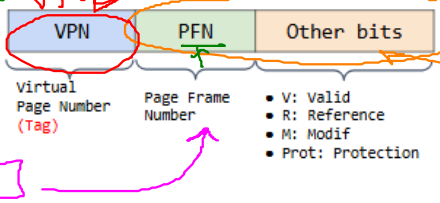
b. Formato de una TLB Entry

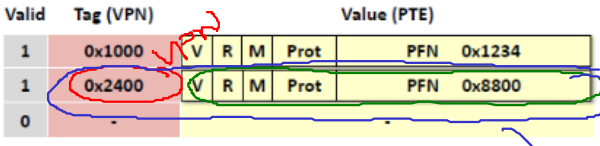
Como es el formato de la TLB Entry

Contenido de una entrada TLB

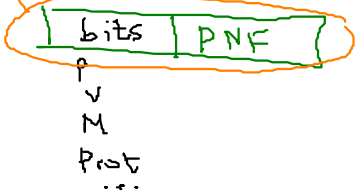
- A diferencia de una entrada de una tabla de página (**PTE = PFN + Other bits**), una entrada de una TLB (**TLB Entry**) contiene la siguiente forma:

TLB Entry = VPN + PFN + Other bits



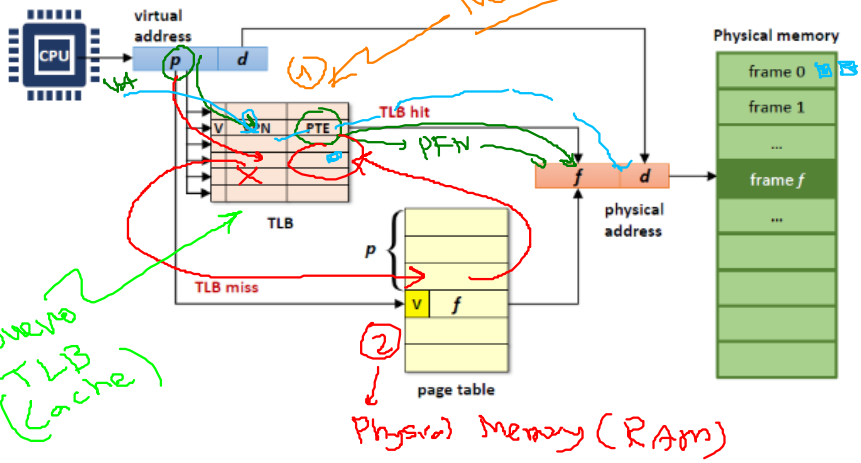


Valid	Tag (VPN)	Value (PTE)					
1	0x1000	V	R	M	Prot	PFN	0x1234
1	0x2400	V	R	M	Prot	PFN	0x8800
0	-						



TLBEntry = [VPN | PTE]

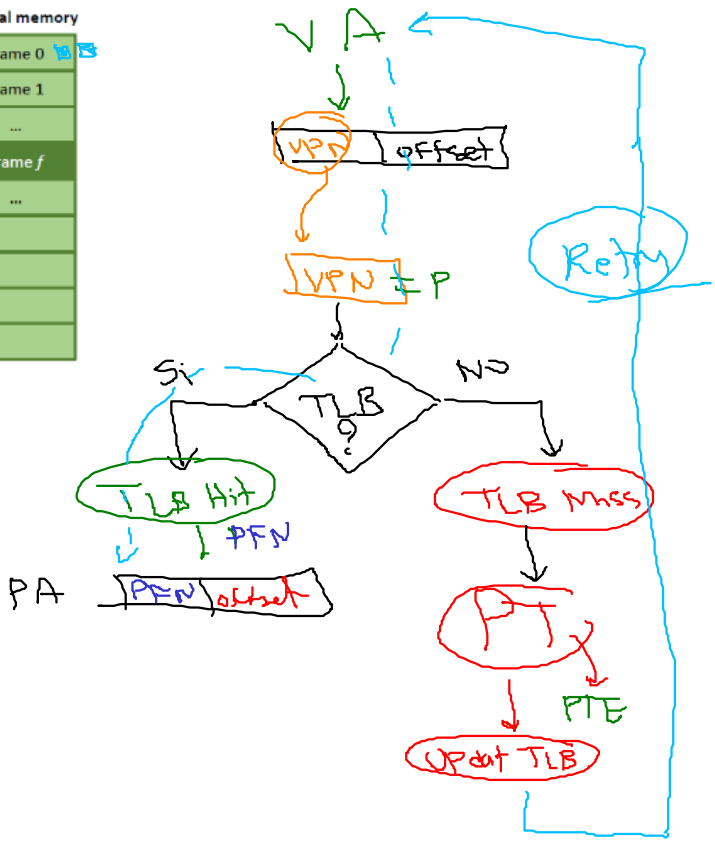
c. Traduccion de direcciones usando TLB



Nuevo TLB (Cache)

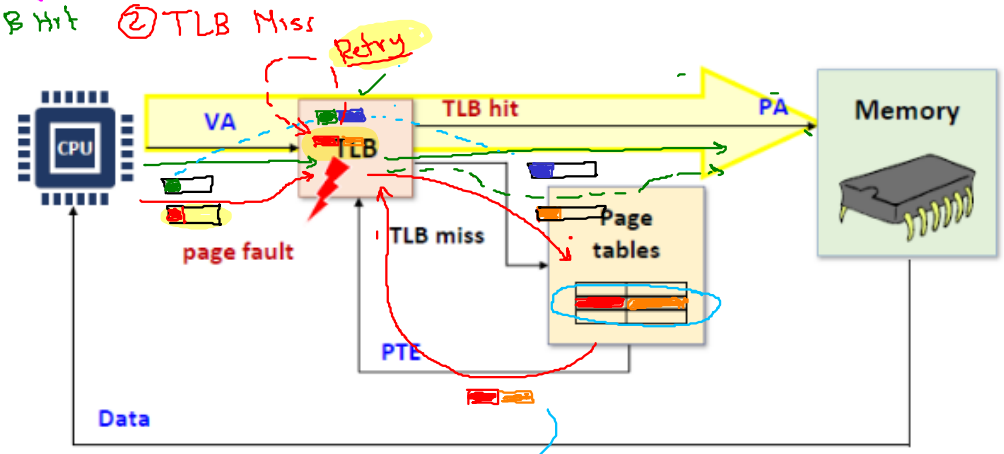
Physical Memory (RAM)

Ejemplo: Bodega



d. Hit y Miss en TLB

① TLB Hit ② TLB Miss



e. Algoritmo TLB

como es este código en

① VA (Formato) → VPN
② PA (Formato)
③ TBL, TLBEntry (Formato)
④ PT → PTE (Formato)

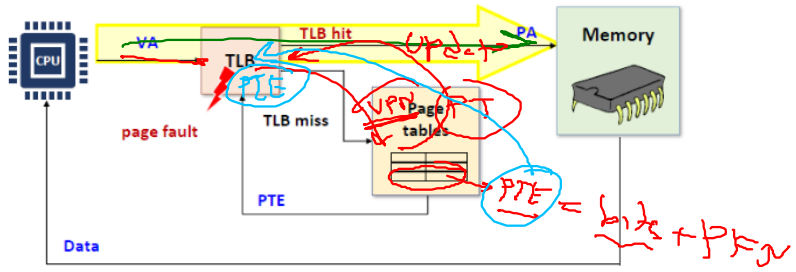
```
1  VPN = (VirtualAddress & VPN_MASK) >> SHIFT
2  (Success, TlbEntry) = TLB_Lookup(VPN)
3  if (Success == True) // TLB Hit
4    if (CanAccess(TlbEntry.ProtectBits) == True)
5      Offset = VirtualAddress & OFFSET_MASK
6      PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7      Register = AccessMemory(PhysAddr)
8    else
9      RaiseException(PROTECTION_FAULT)
10 else // TLB Miss
11   PTEAddr = PTBR + (VPN * sizeof(PTE))
12   PTE = AccessMemory(PTEAddr)
13   if (PTE.Valid == False)
14       RaiseException(SEGMENTATION_FAULT)
15   else if (CanAccess(PTE.ProtectBits) == False)
16     RaiseException(PROTECTION_FAULT)
17   else
18     TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
19     RetryInstruction()
```

Resumen Algoritmo:

- Instrucciones comunes ( )
- Instrucciones hit ( )
- Instrucciones miss ( )
- Instrucciones interrupción ( )

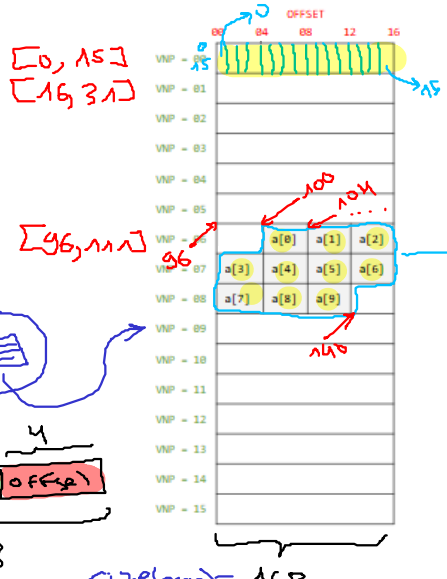
Operaciones: Miss TLB, Hit TLB, Access invalid

VA → PA



02/10/2025 – Sistemas Operativos (UdeC)

Ejemplo: Mejora en el desempeño gracias al TLB



[0, 15]
[16, 31]
[96, 111]

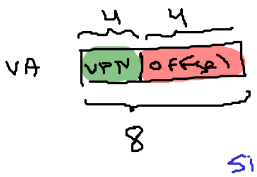
Ejemplo - Accediendo a un arreglo

A continuación se muestra como una TLB puede mejorar el desempeño en la traducción de direcciones

```
1  int sum = 0;
2  for (i = 0; i < 10; i++) {
3    sum += a[i];
4  }
```

Suposiciones (continuación):

1. Se tiene un array de diez elementos tipo int:
   - size(int) = 4B → size(array) = 40 B
2. La dirección base del arreglo a es 100: &a[0] = 100
3. Address Space:
   - Número de bits de direcciones: n(VA) = 8
   - Tamaño de la memoria: 2^8 = 256
4. Páginas:
   - size(page) = 16 B
   - num_paginas = size(VA)/size(page) = 256/16 = 16



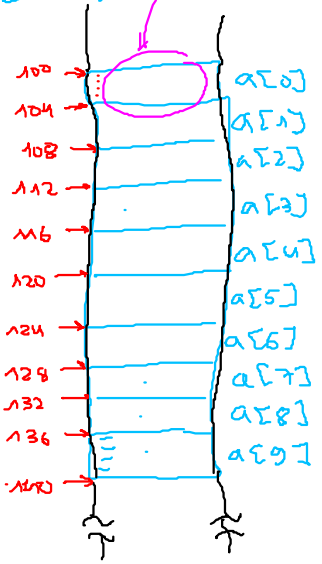
size(page) = 16B

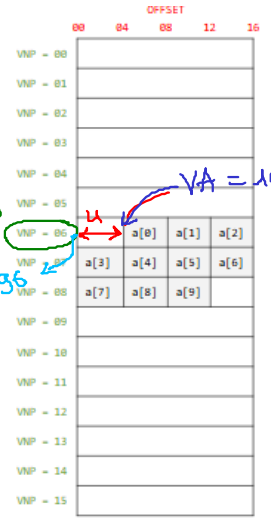
pages = 16 = $2^{n(VPN)}$ = 2^4

Size(AS) = page(size(page)) = 16 × 16 = 256 = $2^{8=n}$

size(int) = 4

[100, 139]





Ejemplo - Accediendo a un arreglo

Pregunta:

- ¿Cual es el offset asociado a la dirección de 100?

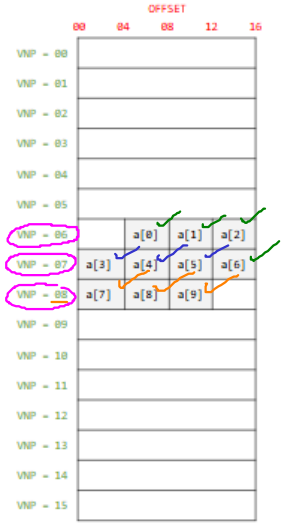
VA = 100 = &a[0] = a;

VA | VPN | offset

n = 8

VPN = 6 offset = 4

VA = 100_{10} = $0x64$ = [0110 | 0100]



Ejemplo - Accediendo a un arreglo

Pregunta:

- Asumiendo una TLB de una sola entrada, ¿Cuantos **TLB miss** y **TLB hits** se presentan en este ejemplo?

TLB

VPN	PFN
9	1
6	PFN_x
7	PFN_y
8	PFN_z

PT

VPN	PFN
...	...
6	PFN_x
7	PFN_y
8	PFN_z
...	...

Dato	VPN	Resultado (M/H)
a[0]	6	M
a[1]	6	H
a[2]	6	H
a[3]	7	M
a[4]	7	H
a[5]	7	H
a[6]	7	H
a[7]	8	M
a[8]	8	H
a[9]	8	H

200 ns

a[0] → &a[0]

100

[6 | 4]

VPN offset

Pregunta:

- Asumiendo una TLB de una sola entrada, ¿Cuantos **TLB miss** y **TLB hits** se presentan en este ejemplo?

Acceso a[i]	0	1	2	3	4	5	6	7	8	9
H/M	M	H	H	M	H	H	H	M	H	H
VPN	6			7				8		

TLB

VPN	PFN	
~~9~~	~~1~~	M
6	~~PFN_x~~	HHM
~~7~~	~~PFN_y~~	HHHM
8	PFN_z	HH

PT

VPN	PFN
...	...
6	PFN_x
7	PFN_y
8	PFN_z
...	...

M: Miss (3)

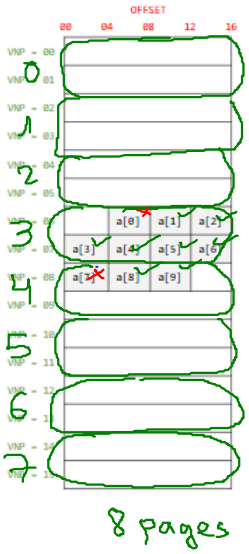
H: Hit (7)

Hits	7 (70%)	Estadísticas
Miss	3 (30%)	
Total	10 (100%)	TLB Hit rate = 70%

↓ Demora.

Conclusión

Las TLB mejoran el desempeño debido a los principios de localidad espacial y temporal



Importancia del tamaño de la página

- A mayor tamaño de la página, menor cantidad de TLB misses.

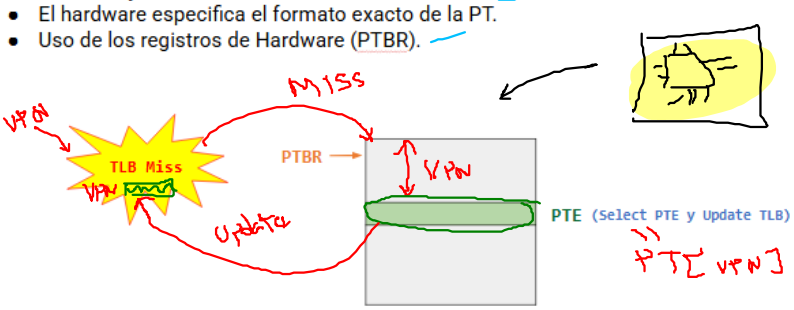
Page Size	TLB Misses	Hit Rate
16	3	70 %
32	2	80 %

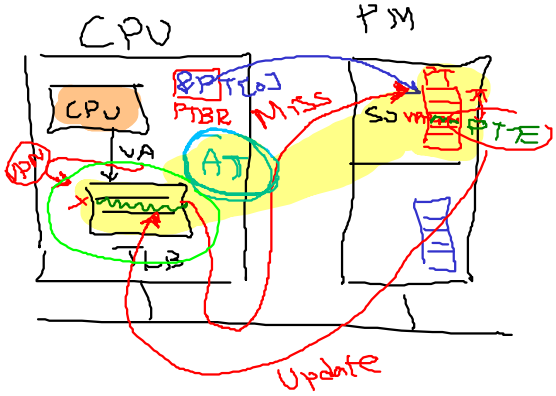
- Con un tamaño de página grande (por ejemplo 4KB) la tasa de Hits mejora y por lo tanto, también el desempeño. (Hit rate → 100%).

3. Manejo de la TLB Miss:

Opción 1 - Hardware-managed TLB (x86, ARM)

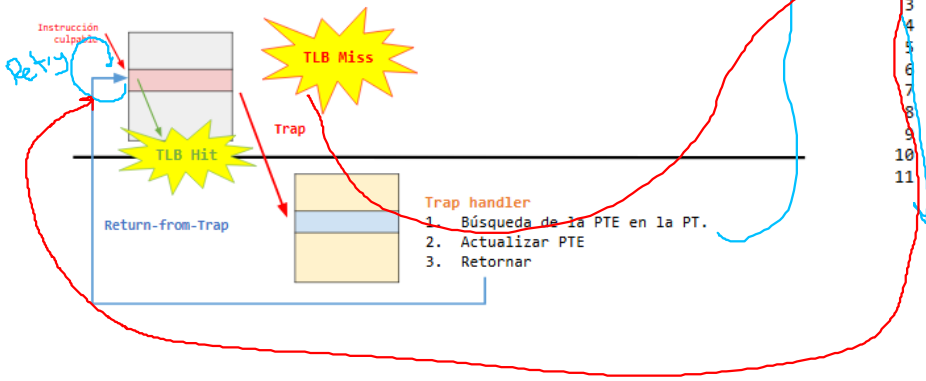
- El hardware conoce donde se encuentran las Page tables (PT) en memoria.
- Cuando sucede un TLB Miss, el Hardware se **mueve** a través de la PT, **encuentra** la PTE correcta y **extrae** la traducción deseada para luego **actualizar** la TLB. Luego la instrucción se vuelve a ejecutar.
- El hardware especifica el formato exacto de la PT.
- Uso de los registros de Hardware (PTBR).





Opción 2 - Software-managed TLB (MIPS,...)

- Cuando sucede un TLB miss, el hardware lanza la excepción.
- La excepción es manejada por el Trap handler.
- En este caso la instrucción **return to trap** es un poco **diferente** al caso tradicional ya que en este caso el hardware debe continuar la ejecución en la instrucción causante del trap (y no después).




```
1  VPN = (VirtualAddress & VPN_MASK) >> SHIFT
2  (Success, TlbEntry) = TLB_Lookup(VPN)
3  if (Success == True) // TLB Hit
4    if (CanAccess(TlbEntry.ProtectBits) == True)
5      Offset = VirtualAddress & OFFSET_MASK
6      PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7      Register = AccessMemory(PhysAddr)
8    else
9      RaiseException(PROTECTION_FAULT)
10 else // TLB Miss
11   RaiseException(TLB_MISS)
```

TLB Control Flow Algorithm (OS Handled)

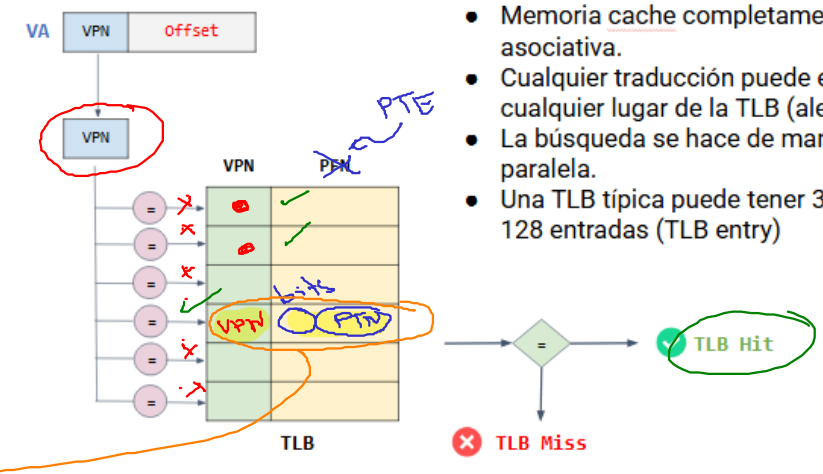
4. Sobre la TLB Entry

TLB Entry = VPN + PTE

PTE = bits + PFN

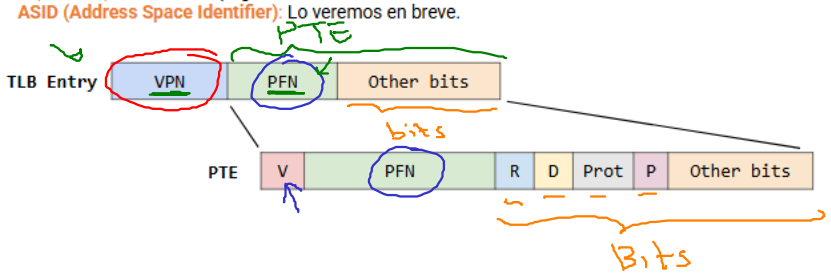
Sobre las TLB

- Memoria cache completamente asociativa.
- Cualquier traducción puede estar en cualquier lugar de la TLB (aleatoria).
- La búsqueda se hace de manera paralela.
- Una TLB típica puede tener 32, 64 0 128 entradas (TLB entry)



Entrada TLB

- V (Valid): bit que dice si un PTE puede o no ser usada. Se evalúa cada vez que una dirección virtual es usada.
- D (Dirt): Indica si la salida ha sido modificada (Una operación write ocurre sobre esta).
- R (Reference): Indica si la página ha sido accedida. Es llevada a 1 (set) cuando la página ha sido leída o escrita. Es útil ya que:
  - Rastrea el acceso a la página.
  - Determina la popularidad de la página.
- P (Protection): Bits que controlan las operaciones que son permitidas (R, W, X, User/Kernel, etc).
- P (Present): Indica si la página se encuentra en memoria física o en el disco.
- ASID (Address Space Identifier): Lo veremos en breve.



5. TLB y Context switch

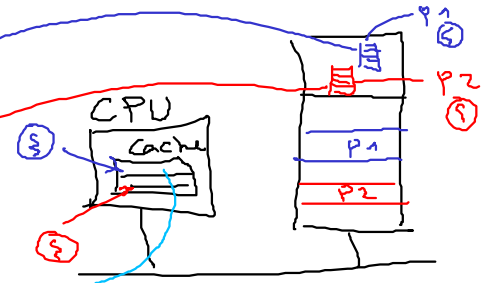
- El TLB es una estructura de hardware (memoria cache) **compartida por todos los procesos**.
- Las traducciones para el proceso en ejecución, **no tienen sentido** para los demás procesos.
- Es necesario que cuando se cambie de un proceso a otro; el hardware, el software o ambos, tengan en cuenta este detalle para asegurarse de que el proceso que está a punto de ejecutarse, no acceda de manera accidental la información de algún proceso ejecutado previamente.

Ejemplo:

Se tienen dos procesos (P1 y P2) con las siguientes PT y la siguiente TLB.

¿Que pasa cuando se da un cambio de contexto?

Problema: Se tiene la misma tabla para todos los procesos; sin embargo, no se puede distinguir qué entrada se asocia a cada proceso.



P1

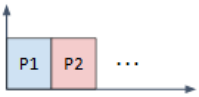
PT P1

VPN	PFN
10	100

P2

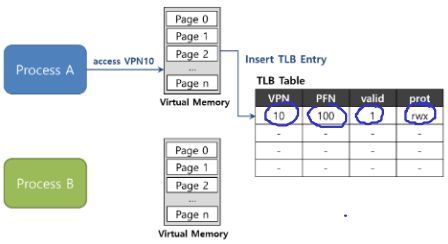
PT P2

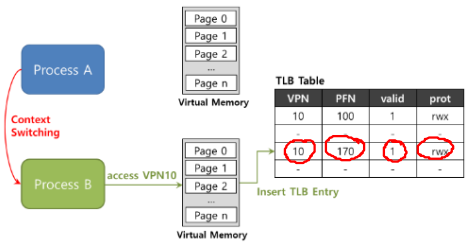
VPN	PFN
10	170

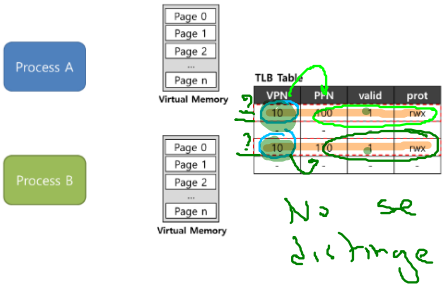


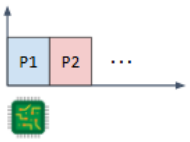
VPN	PFN	valid	Prot
10	100	1	rwx
---	---	0	---
10	170	1	rwx
---	---	0	---

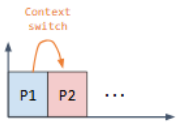
TLB Entry











Problema: Ambos procesos tienen entradas en la TLB pero no se puede distinguir cuál entrada está asociada a cada proceso.

Solucionando el problema

Solución 1 - Vaciar (flush) la TLB en los cambios de contexto

- **Flush**: Establecer todos los bits de validez (**V**) a **cero** (0).

Solución 1 - Vaciado (vaciado) de la TLB - Consecuencias del vaciado

1. Un proceso nunca encontrará accidentalmente traducciones incorrectas en la TLB.
2. El **flushing** es costoso pues se pierden todas las traducciones recientemente cargadas a la TLB:

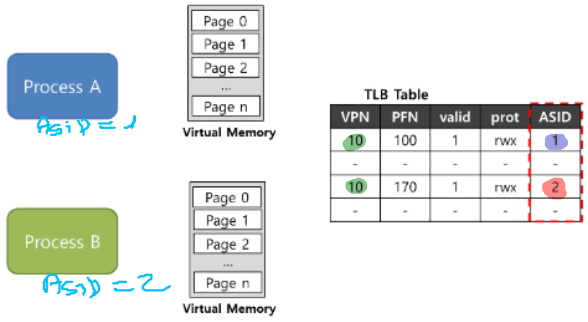
`context switch [up] → TLB miss [up] → Costo [up]`

Solución 2 - Proveer un campo Address Space Identifier (ASID) en el TLB

- **ASID (Address Space Identifier)**: Campo de 8 bits en la TLB.
- Permite diferenciar un proceso de otro en la TLB.
- Gracias a este campo, varios procesos pueden compartir la TLB (manteniendo sus instrucciones) sin ninguna confusión.
- Soluciona el problema de overhead debido a la técnica de vaciado (flushing) de la TLB.

VPN	PFN	valid	Prot	ASID
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---

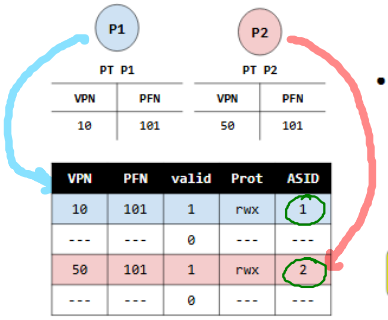
Proveer un campo Address Space Identifier (ASID) en el TLB - Ejemplo



TLB Table

VPN	PFN	valid	prot	ASID
10	100	1	rwx	1
-	-	-	-	-
10	170	1	rwx	2
-	-	-	-	-

Proveer un campo Address Space Identifier (ASID) en el TLB - Ejemplo



VPN	PFN	valid	Prot	ASID
10	101	1	rwx	1
---	---	0	---	---
50	101	1	rwx	2
---	---	0	---	---

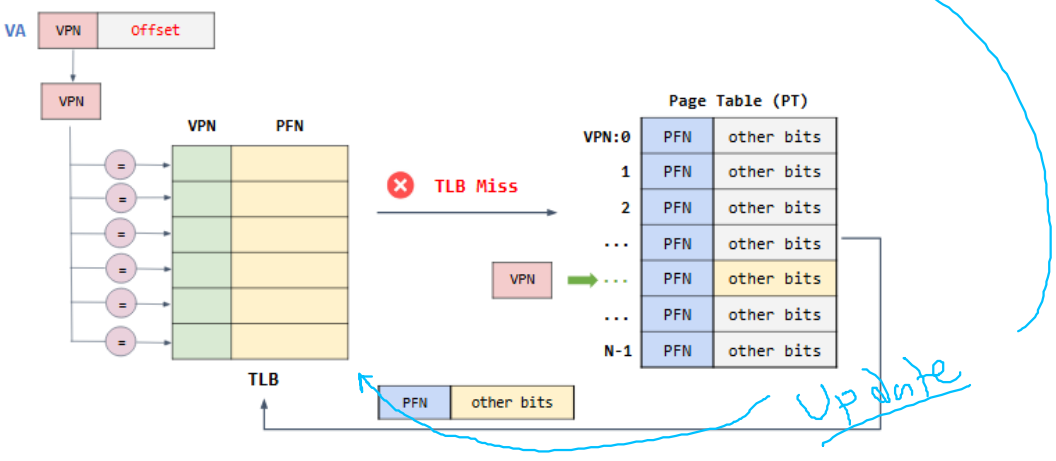
- Es común que procesos distintos compartan segmentos de código (binaries, shared libraries).

¿Qué pasa cuando dos procesos apuntan a un mismo frame (PFN)?

Compartir páginas es útil porque reduce el número de frames en uso.

6. Políticas de reemplazo.

Una política de reemplazo en este caso, consiste en el procedimiento llevado a cabo para reemplazar una entrada antigua en la TLB (cache) por una nueva.



Objetivo de la política de reemplazo

Reducir la tasa de miss aumentando por consiguiente la tasa de hits.

¿Cómo hacer el reemplazo?

Hay diferentes políticas, algunas son:

1. Last Recently Used (LRU).
2. Random (aleatorio).

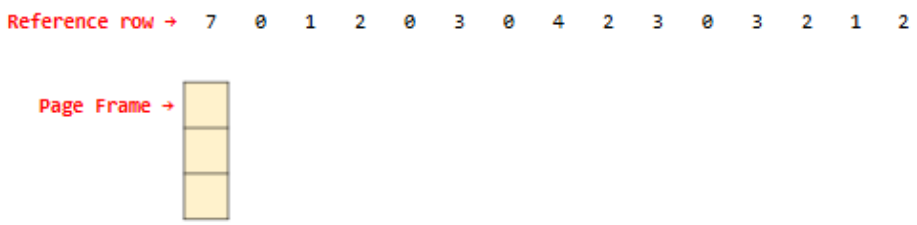
LRU (Last Recently Used)

Reemplaza la entrada de la TLB que no haya sido usada recientemente (la que lleva más tiempo sin ser accedida).

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, 2
```



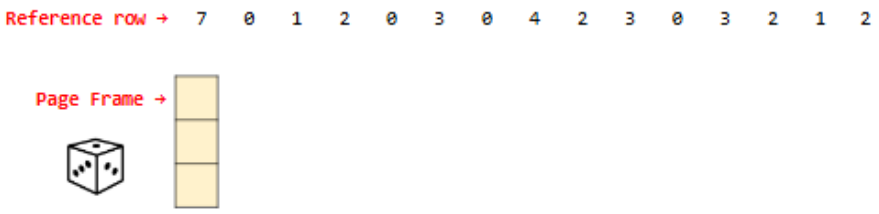
Random

Realiza un reemplazo seleccionando una entrada al azar de la TLB.

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:

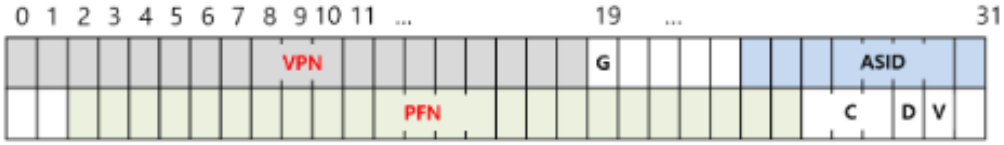
```
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2
```



7. Caso real

Forma de la entrada TLB

- TLB de un MIPS R4000.
- El sistema operativo maneja la TLB por software.



All 64 bits of this TLB entry (example of MIPS R4000)

Flag	Content
19-bit VPN	The rest reserved for the kernel.
24-bit PFN	Systems can support with up to 64GB of main memory(2^24*4KB pages).
Global bit (G)	Used for pages that are globally-shared among processes.
ASID	OS can use to distinguish between address spaces.
Coherence bit (C)	Determine how a page is cached by the hardware.
Dirty bit (D)	Marking when the page has been written.
Valid bit (V)	Tells the hardware if there is a valid translation present in the entry.