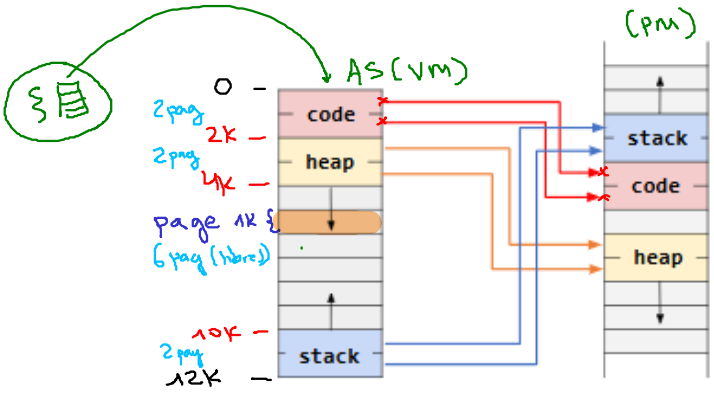
18/09/2025 - Sistemas Operativos (Vde@)

Paginación

1. Idea basica

Aspectos claves

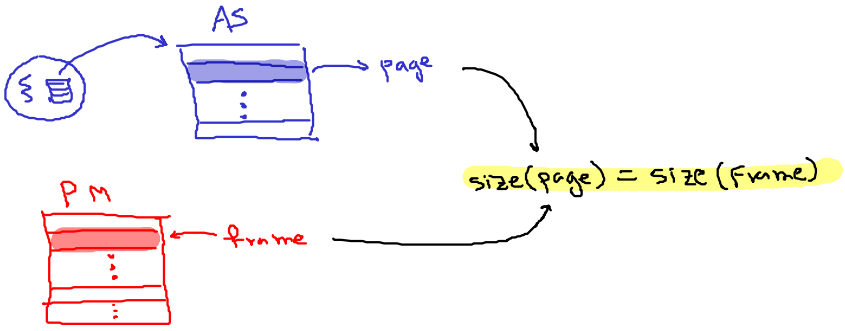
- Dividir el espacio de direcciones (**address space**) en unidades de tamaño fijo conocidas como páginas.
- Cada página es independientemente mapeada en memoria física.
- Elimina la necesidad de que los segmentos del espacio de direcciones sea continuo.



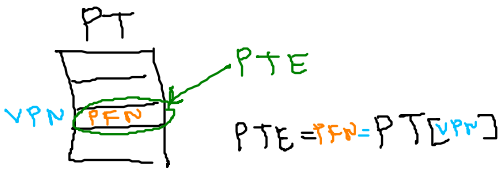
2. Conceptos importantes

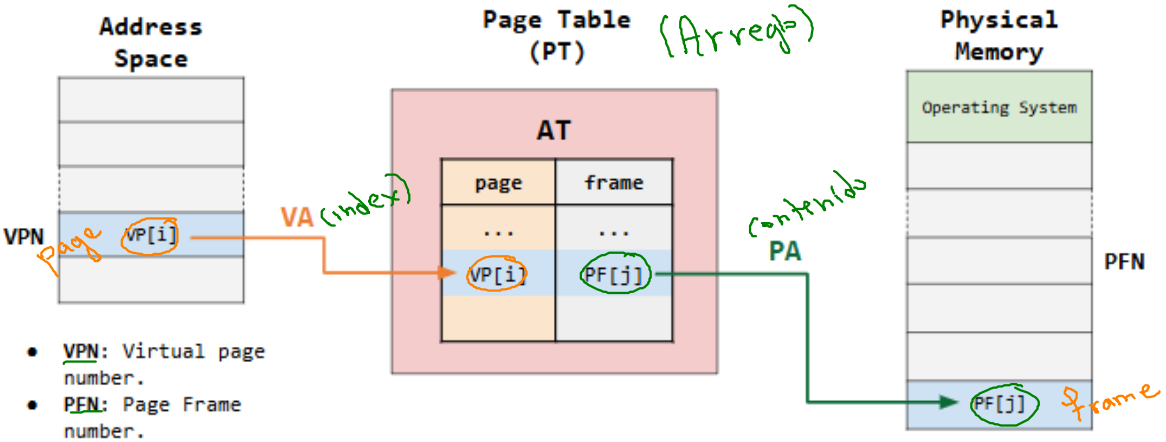
a. Page (Pagina)
↓
Memoria virtual

b. Frame (Marco)
↓
Memoria física



c. Page Table (Tabla de pagina)
↳ Page Table Entry (PTE)





- VPN: Virtual page number.
- PFN: Page Frame number.

Ejemplo ideal:

Ejemplo (Muy ideal)

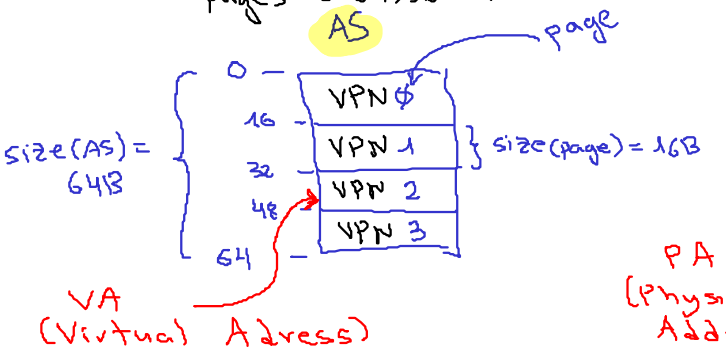
- Memoria física de 128 bytes con frames (marcos de página) de 16 bytes.
- Espacio de direcciones (Address Space) de 64 bytes con páginas de 16 bytes.

size(PM) = 128B VM size(AS) = 64B size(Frame) = 16B size(page) = 16B

Se pide: Bosqueje la memoria virtual y la Física y responda.

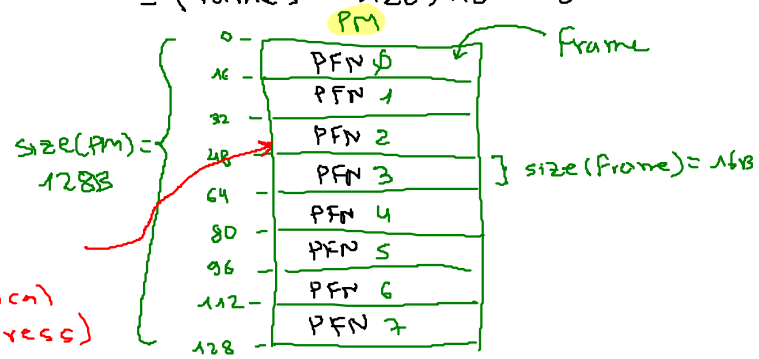
Virtual Memory

- size(AS) = 64B
- size(page) = 16
- pages = 64/16 = 4



Physical Memory

- size(PM) = 128B
- size(Frame) = 16B
- Frames = 128/16 = 8



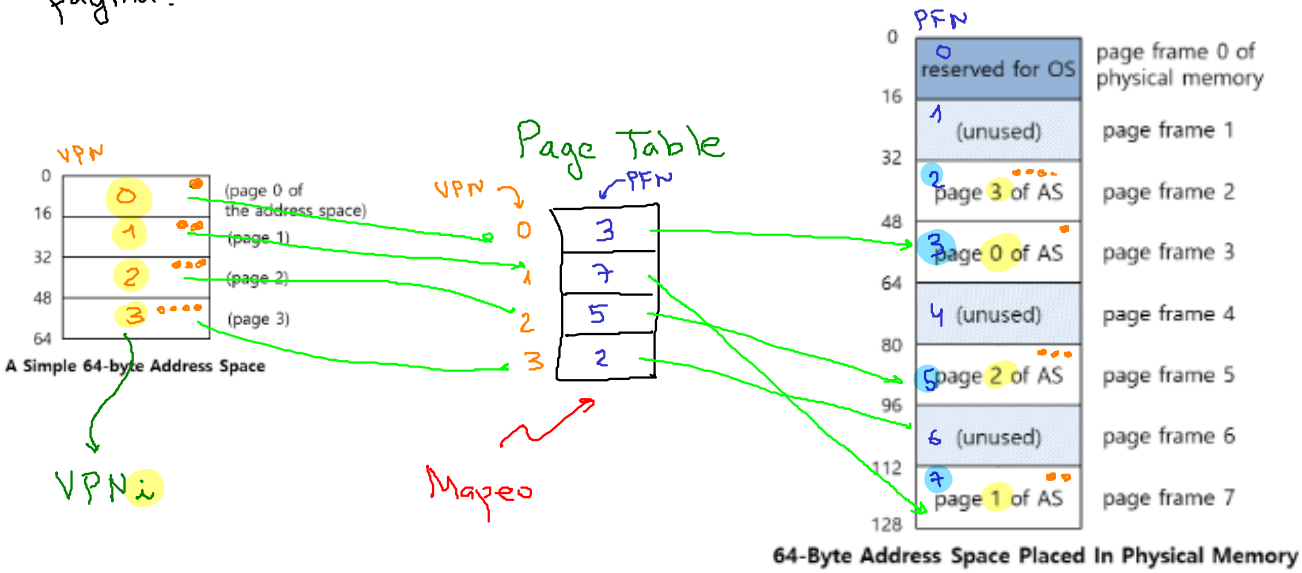
a. ¿Cual es el numero de paginas en que se divide la MV?

Pages (AS) = 4

b. ¿Cual es el numero de Frames en que se divide la PM?

Frames (PM) = 8

c. Teniendo en cuenta la siguiente figura como seria la Tabla de pagina?



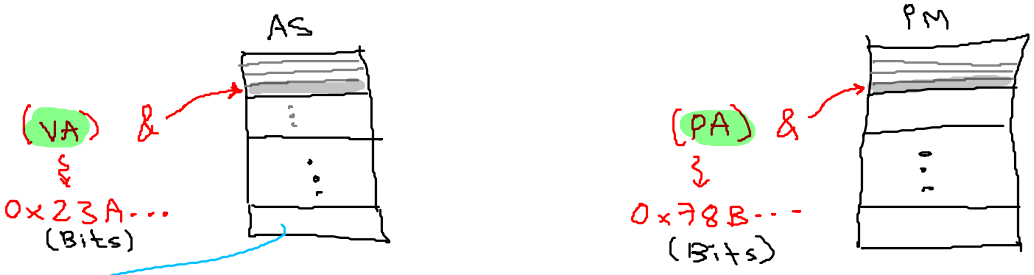
PFN j

Conceptos vistos:

1. Memoria Virtual (Address space)
   - Paginas
2. Memoria Fisica
   - Frames
3. Tabla de pagina (PT)

23/09/2025 - Sistemas Operativos (UdeC)

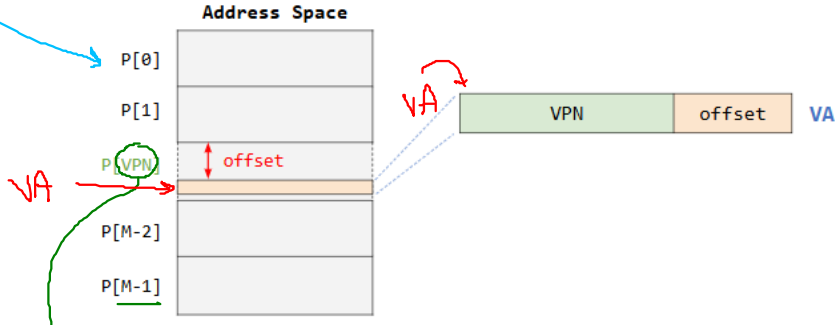
2. Traducción de direcciones



Direcciones virtuales (Address Space)

Una dirección virtual está compuesta por dos partes:

- **VPN (Virtual Page number):** Índice asociado a una página en particular.
- **Offset:** Desplazamiento dentro de la página seleccionada.

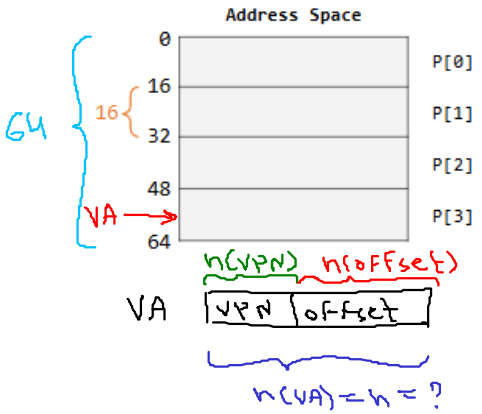


VPN: Virtual page Number (page index)

Ejemplo: Formato de direcciones Virtuales.

Ejemplo - Enunciado del ejemplo original

- Memoria física de 128 bytes con frames (marcos de página) de 16 bytes. PM
- Espacio de direcciones (Address Space) de 64 bytes con páginas de 16 bytes. AS(VM)



Datos:

- size(AS) = 64 B
- size(page) = 16 B
- Número de páginas = 4 (Obtenido previamente)

Preguntas:

1. ¿Cuántos bits se necesitan para direccionar la memoria virtual? n(VA)=? = n
2. ¿Cuántos bits se necesitan para referirse a cada página? n(VPN)=?
3. ¿Cuántos bits se necesitan para acceder a cada las direcciones de cada página? n(offset)=?
4. ¿Cuál es el VPN y el offset de la dirección virtual 21 en el AS de 64 B?

1. n(VA) = n = ?

Rango n bits: $0 \leq VA < 2^n - 1$ → Total de direcciones: 2^n (size)

$$size(AS) = 64 = 2^n$$

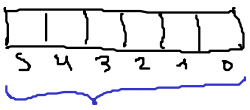
$$2^6 = 2^n$$

$$n = n(VA) = 6$$

$size(AS) = 2^n$

$n = 6$

VA



$n = 6$

$VA \in [0, 63]$

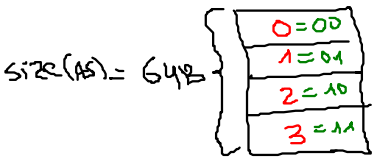
$0 = 000000$

$1 = 000001$

$63 = 111111$

2. $n(VPN) = ?$

AS



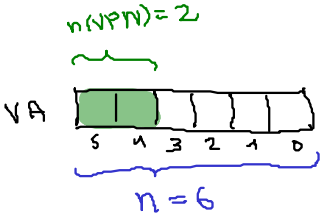
pages = 4

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

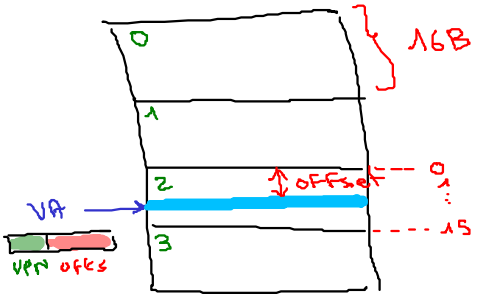
$4 = 2^{n(VPN)}$

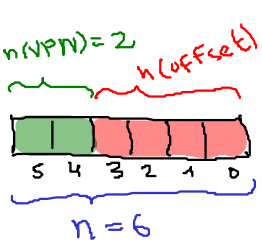
$2^2 = 2^{n(VPN)}$

$n(VPN) = 2$



3. $n(offset) = ?$





$n(offset) = n - n(VPN)$

$n(offset) = 6 - 2$

$n(offset) = 4$

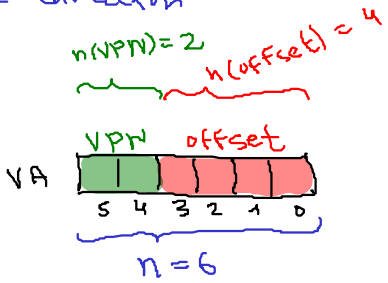
$size(page) = 2^{n(offset)}$

$16 = 2^{n(offset)}$

$2^4 = 2^{n(offset)}$

$4 = n(offset)$

* Formato de dirección



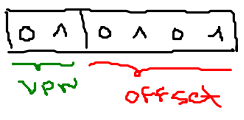
4. $VA = 21$

- VPN = ?
- offset = ?

$VA = 21 =$ 010101_2 ($n = 6$)

$offset = 0101_2 = 5$

$VPN = 01_2 = 1$



VPN = 1

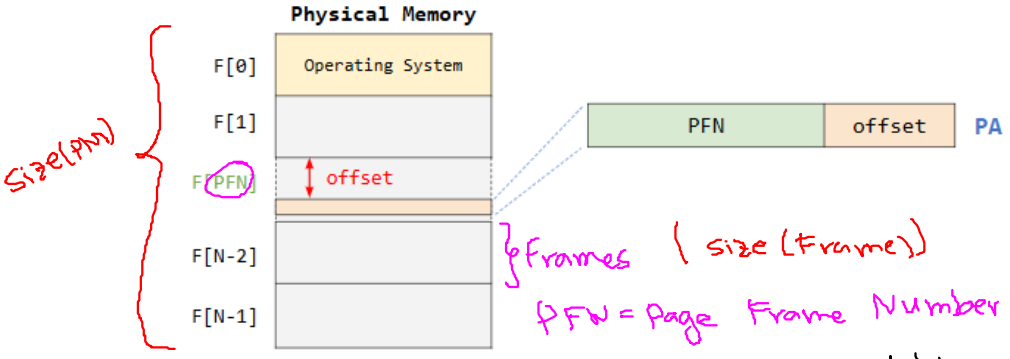
offset = 5

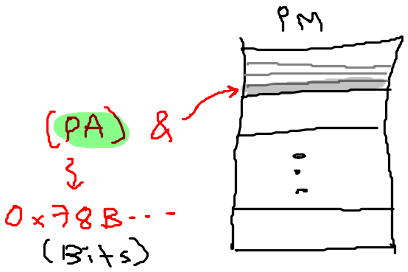
Como sería en términos de ubicación?

Direcciones físicas

Una dirección física está compuesta por dos partes:

- **PFN (Page Frame number)**: Frame dentro del que se encuentra la dirección física.
- **Offset**: Desplazamiento dentro del frame seleccionado.





1. n(PA): size(PA) = 2^n
2. n(PFN): Frames = $2^{n(PFN)}$
3. n(offset) = size(Frame) = $2^{n(offset)}$ = size(page)

Ejemplo: Formato de direcciones físicas

Ejemplo - Enunciado del ejemplo original

- Memoria física de 128 bytes con frames (marcos de página) de 16 bytes.
- Espacio de direcciones (Address Space) de 64 bytes con páginas de 16 bytes.

Datos:

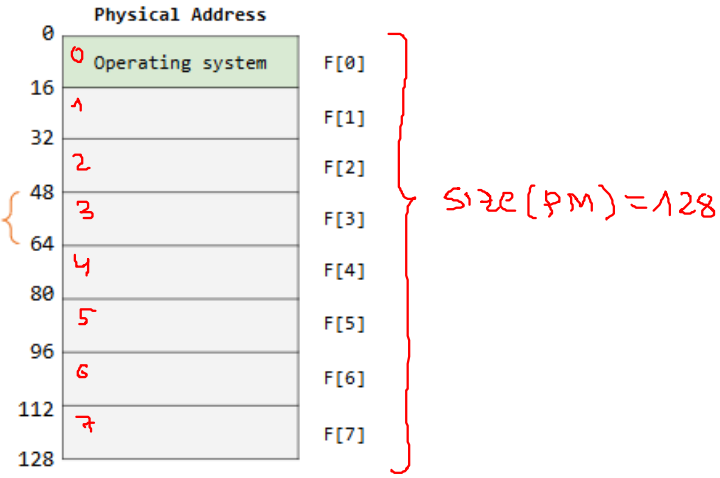
- size(PM) = 128 B
- size(page) = 16 B
- Número de frames = 8 (Obtenido previamente)

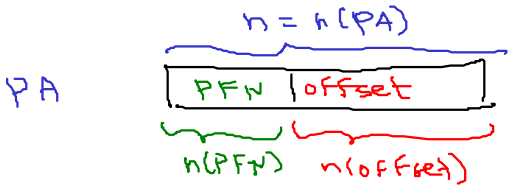
size(Frame)=16

Frames = 8

Preguntas:

1. ¿Cuántos bits se necesitan para direccionar la memoria física?
2. ¿Cuántos bits se necesitan para hacer referencia a los frames?
3. ¿Cuántos bits se necesitan para el offset?
4. ¿Cómo es el formato de la dirección para este ejemplo?





1. $n = n(PA) = ?$

$size(PM) = 2^n$

$128 = 2^n$

$2^7 = 2^n$

$n = 7$

2. $n(PFN) = ?$

$frames = 2^{n(PFN)}$

$8 = 2^{n(PFN)}$

$2^3 = 2^{n(PFN)}$

$n(PFN) = 3$

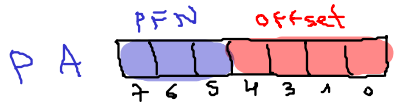
3. $n(offset) = ?$

$n(offset) = n - n(PFN)$

$n(offset) = 7 - 3$

$n(offset) = 4$

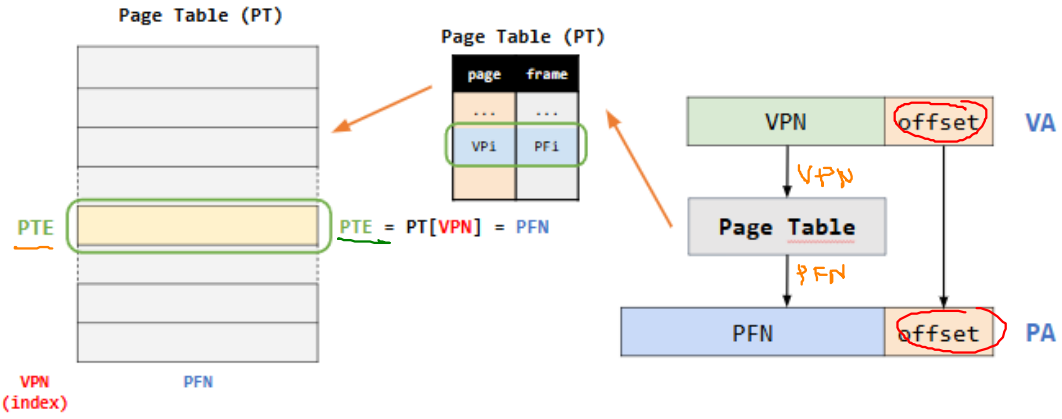
4. Formato de la dirección física



3. Traducción de Direcciones → Necesito la PT (Page Table)

Tabla de página (Page Table)

Array que traduce direcciones virtuales a físicas.



PTE (Page Table Entry): Hay una por cada página en el AS.

Hasta el momento

$VA = [0\ 1 \mid 0\ 1\ 0\ 1]$

$VPN = 1$ $\quad offset = 5$

$VA = 21 \rightarrow PT \rightarrow PA = ?$

$PA = [\quad \mid 0\ 1\ 0\ 1]$

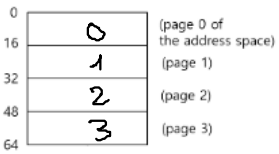
Ejemplo:

Ejemplo - Enunciado del ejemplo original

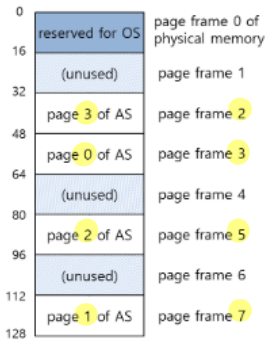
- Memoria física de 128 bytes con frames (marcos de página) de 16 bytes.
- Espacio de direcciones (Address Space) de 64 bytes con páginas de 16 bytes.

Preguntas:

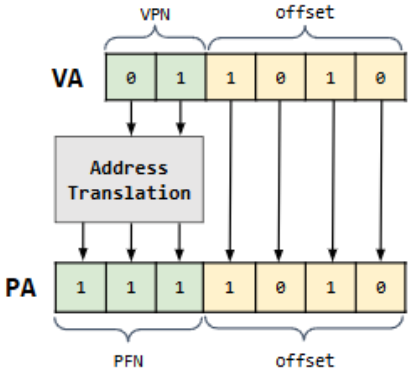
1. ¿Bosqueje la tabla de página?
2. ¿Cuál es la dirección física asociada a la dirección virtual 21?



A Simple 64-byte Address Space



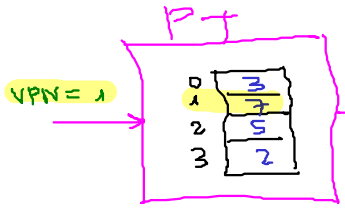
64-Byte Address Space Placed In Physical Memory



PTEs = pages = 4

VA = [0 1 | 0 1 0 1]

VPN = 1 offset = 5



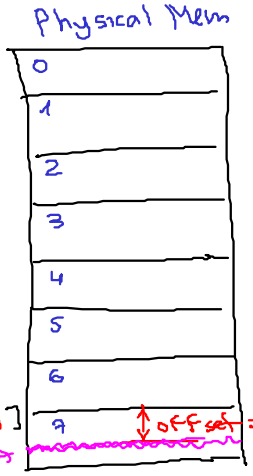
PA = [1 1 1 | 0 1 0 1]

PA = [1 1 1 | 0 1 0 1] = 117

PFN = 7 offset (5)

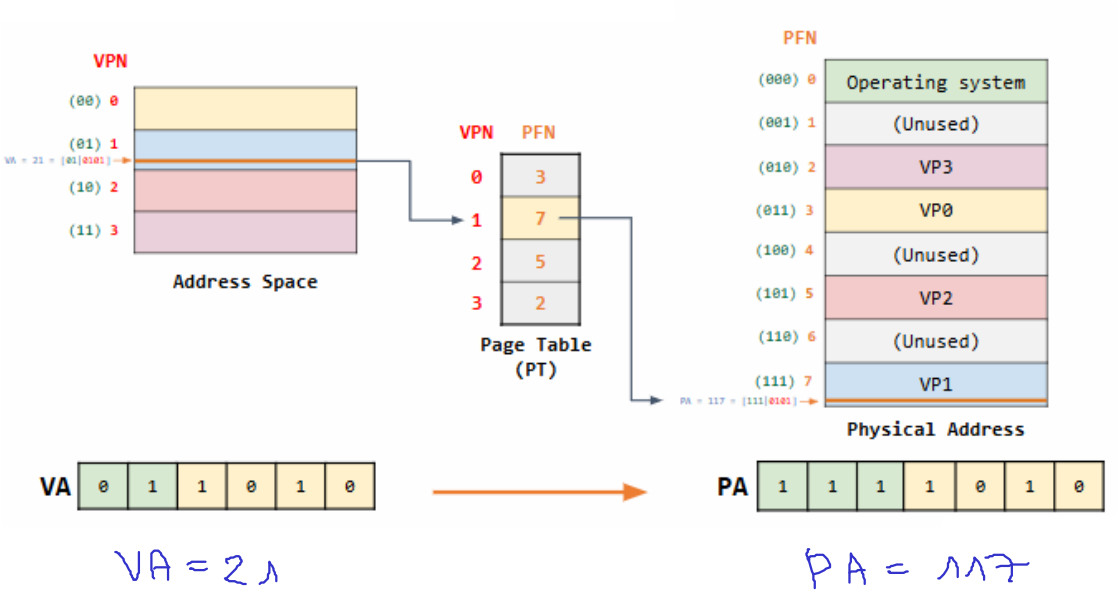
7 x size(page) + offset = 7(16) + 5 = 112 + 5 = 117

En direcciones

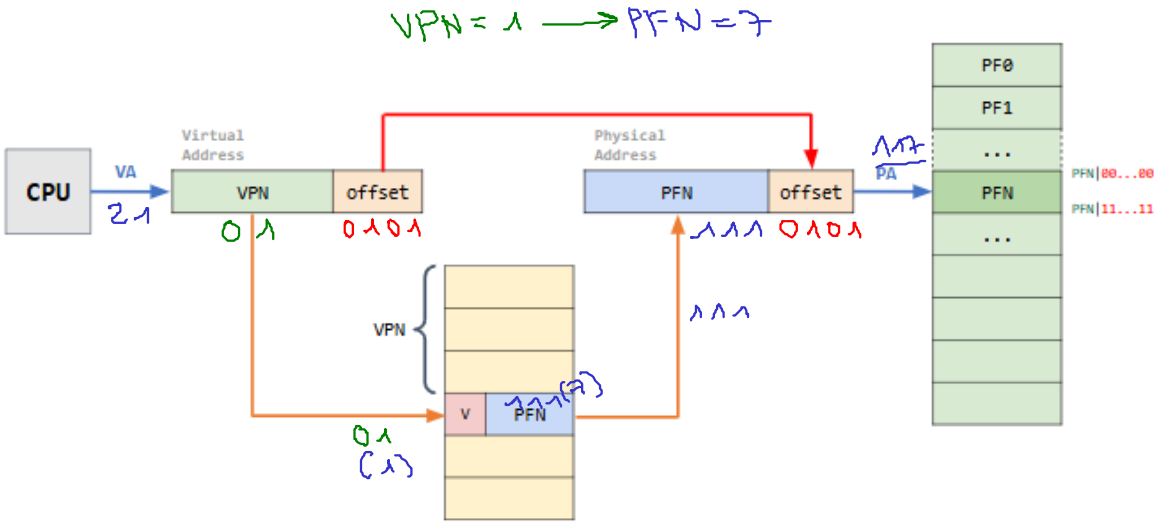


PA = 117 = [1 1 1 0 1 0 1]

PA = 117



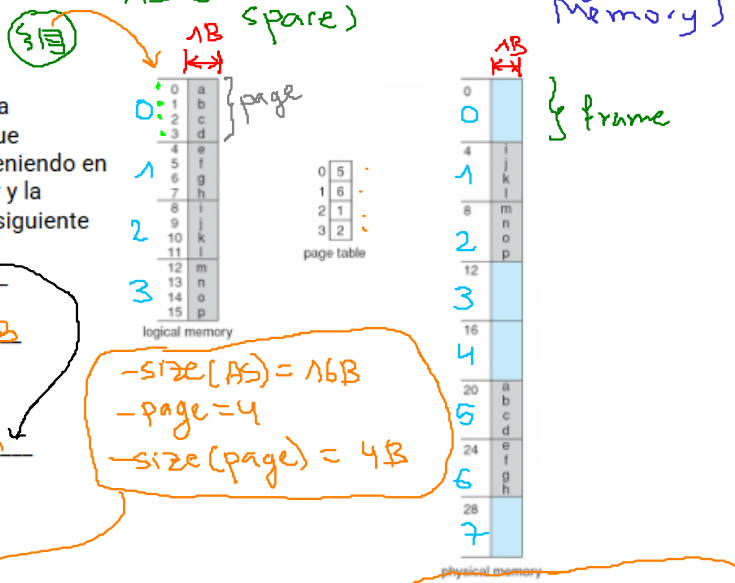
Hardware para la traducción



Ejemplo:

Ejemplo

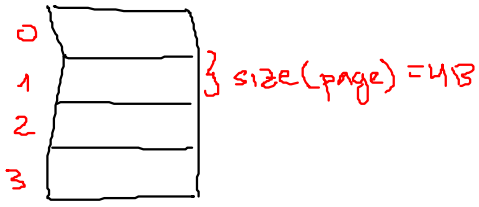
1. A continuación se muestra una memoria virtual y una física que almacenan datos de 1 byte. Teniendo en cuenta la información anterior y la siguiente figura, completar la siguiente información:
   a. Número de páginas: 4
   b. Número de frames: 8
   c. Tamaño de la página: 4B
   d. Offset: 2 bits
   e. VPN: 2 bits
   f. PFN: 3 bits
   g. Número total de PTEs: 4



- size(AS) = 16B
- page = 4
- size(page) = 4B

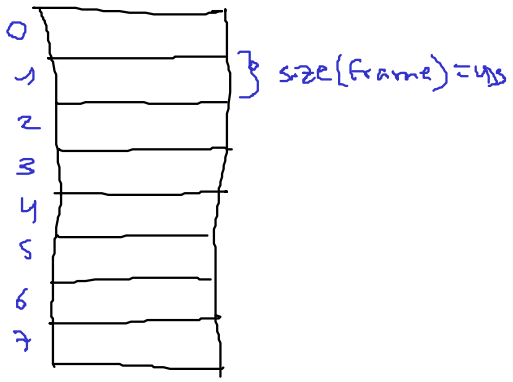
- size(PM) = 32B
- frames = 8
- size(frame) = 4B

AS (Memoria Virtual)

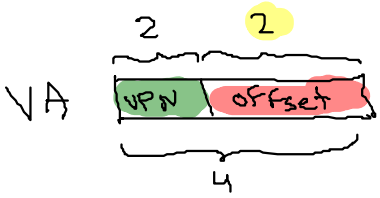


size(AS) = 16B

PM (Memoria Física)



Formato de una VA (Virtual Adress)



* n = ?

$2^n = 16 \rightarrow 2^n = 2^4 \rightarrow n = 4$ bits

* n(VPN) = ?

$2^{n(VPN)} = \text{pages}$

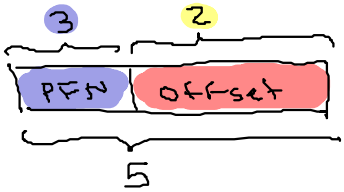
$2^{n(VPN)} = 4 = 2^2$

n(VPN) = 2

* n(offset) = n − n(VPN) = 4 − 2 = 2

size(PM) = 32B

Formato de una PA (Physical Adress)



* n = ?

$2^n = 32 \rightarrow 2^n = 2^5 \rightarrow n = 5$

* PTEs = ?

PTEs = pages = 4

Ejemplo

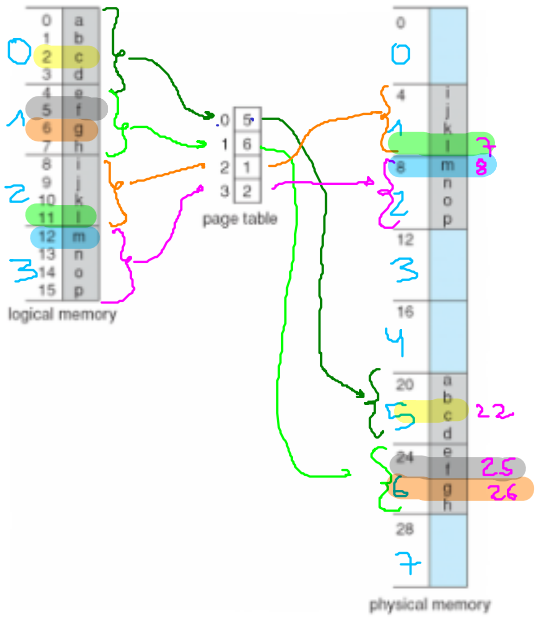
2. Teniendo en cuenta la tabla de página (page table). Llenar la siguiente tabla:

VA [VPN | offset] PA [PFN | offset]

Virtual Address	Physical Address
2 [0010]	[101 | 10] = 22
5 [0101]	[110 | 01] = 25
6 [0110]	[110 | 10] = 26
11 [1011]	[001 | 11] = 7
12 [1100]	[010 | 00] = 8

```
PA = PFN*size(page) + offset
```

4+3



Ejemplo

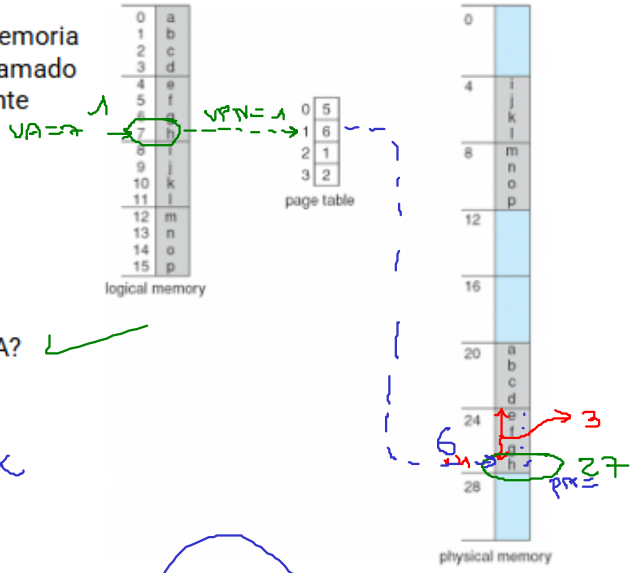
3. Suponga que se va a obtener de memoria la letra **h** a un registro de la CPU (llamado %eax) para lo cual se usa la siguiente instrucción:

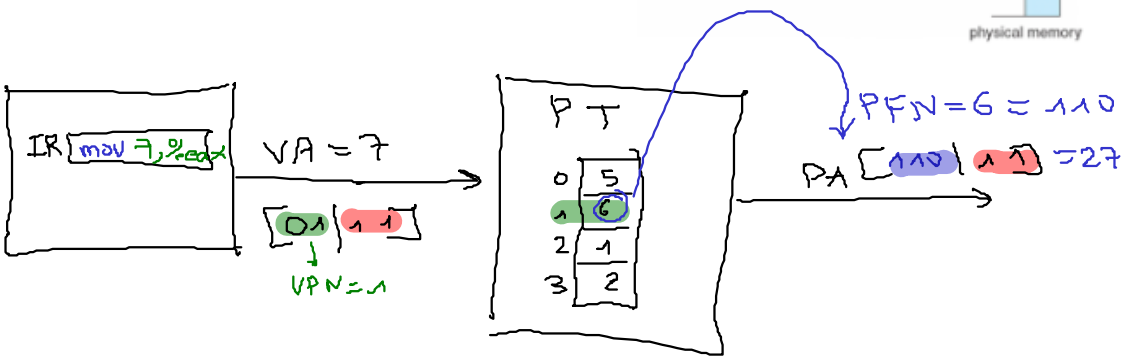
```
mov VA, %eax
```

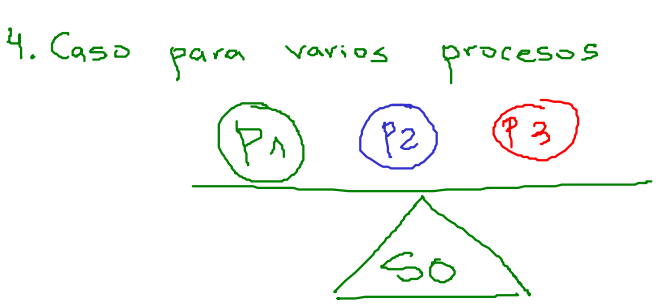
Donde **VA** es la dirección virtual donde se encuentra **h**. ¿Cual es el formato correspondiente a VA y PA?

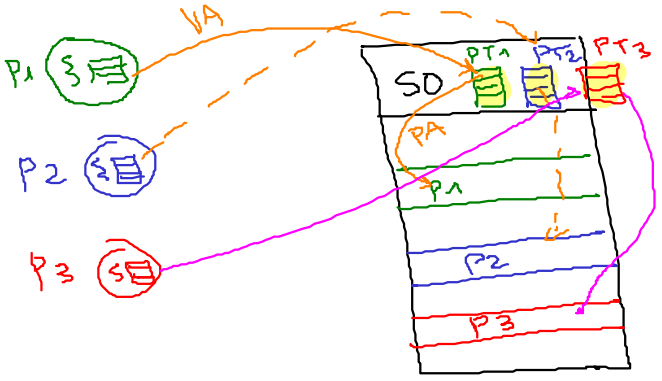
h → VA = 7

mov VA, %eax



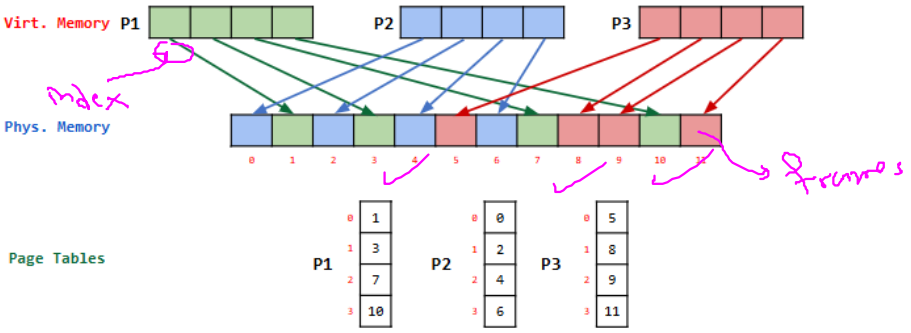






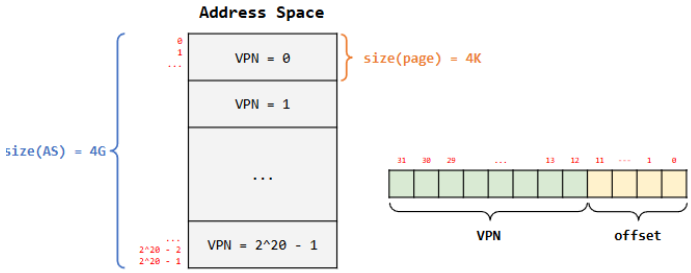
¿Que pasa cuando se ejecutan varios procesos?

- Se crea una tabla de página separada para cada proceso.
  - No hay manera de acceder a la memoria física de otro proceso.
  - En un context switch, el registro del MMU es apuntado a la dirección base de la tabla de página del proceso que se encuentra en ejecución.



¿Que tan grandes pueden ser las páginas?

Las tablas de página **pueden ser muy grandes**. El siguiente ejemplo muestra esto



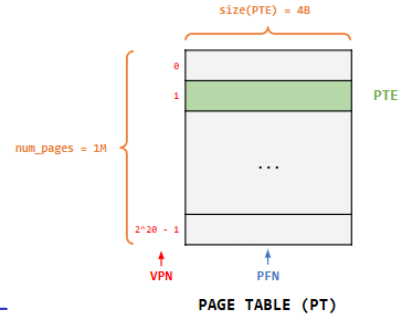
Ejemplo:

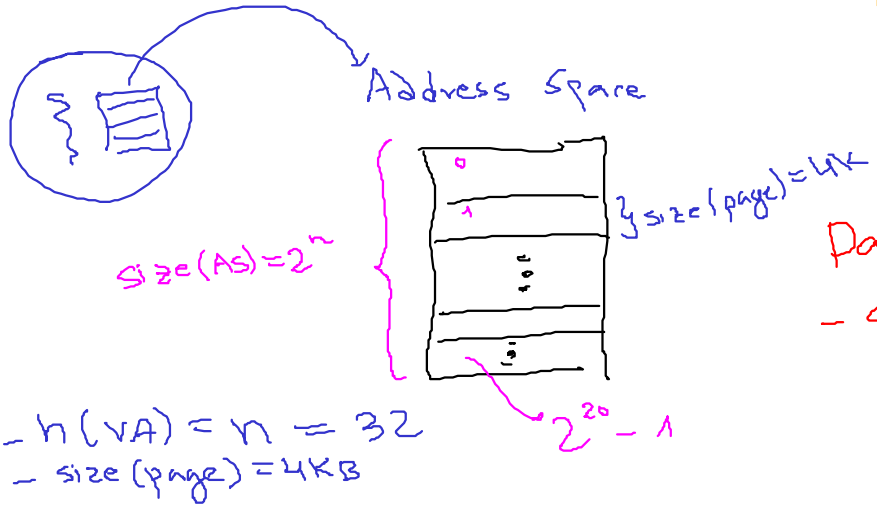
Suponga que se tiene una memoria virtual (Address Space) con las siguientes características:

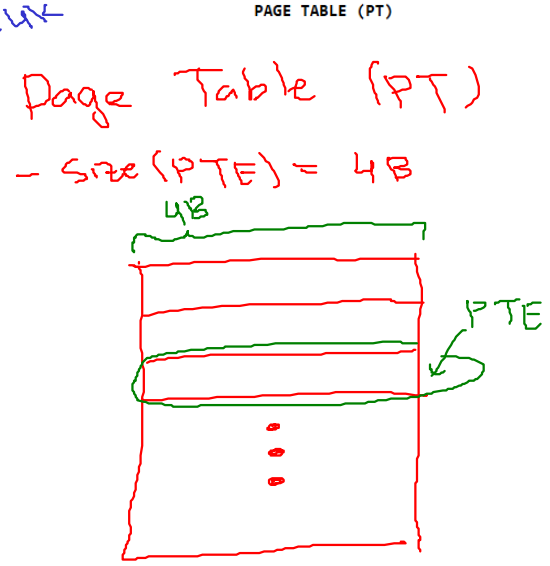
- 32 bits de direcciones.
- Páginas de 4KB
- Asuma que cada PTE tiene un tamaño de 4B

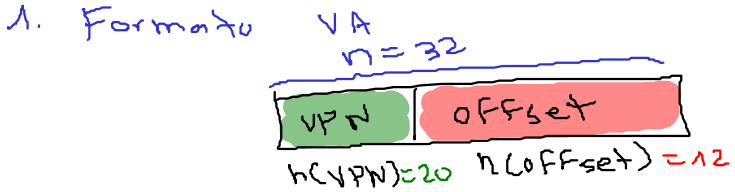
Responda las siguientes preguntas:

1. ¿Cual es el formato de las VA en este caso?
2. ¿Cual es el tamaño de la tabla de página (PT)?







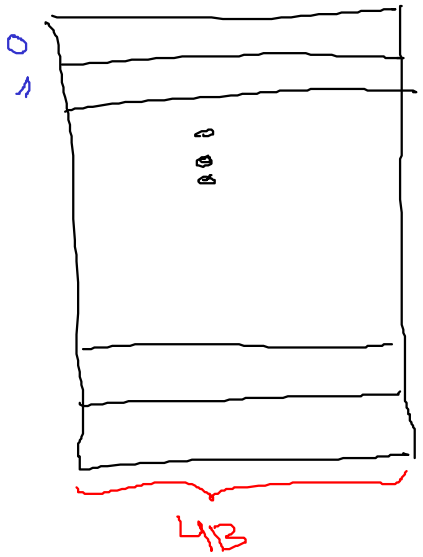


$$pages = \frac{size(AS)}{size(page)} = \frac{2^{32}}{4K} = \frac{2^{32}}{4(2^{10})} = \frac{2^{32}}{2^{12}} = 2^{20} = 1M$$

$$2^{h(VPN)} = pages \rightarrow 2^{h(VPN)} = 2^{20} \rightarrow h(VPN) = 20$$

$$h(offset) = h - h(VPN) = 32 - 20 = 12$$

2. Size(PT) = ?



$2^{20} - 2$

$2^{20} - 1$

$2^{20} = [0, 2^{20} - 1]$

PTEs

$size(PT) = (PTE_s)(size(PTE))$

$= 2^{20} \times 4B$

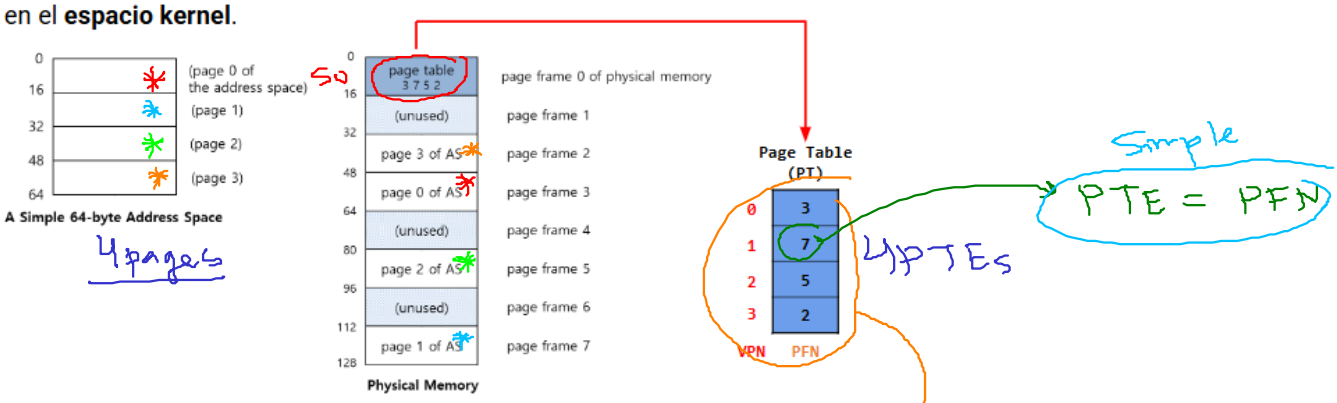
$size(PT) = 4MB$

n Procesos	$\sum size(PT_i)$
1	4MB
3	3 (4MB) = 12MB
1000	1000 (4MB) = 4000 MB = 4000×2^{20}

Gasto de Memoria.

¿Donde se guardan las tablas de página?

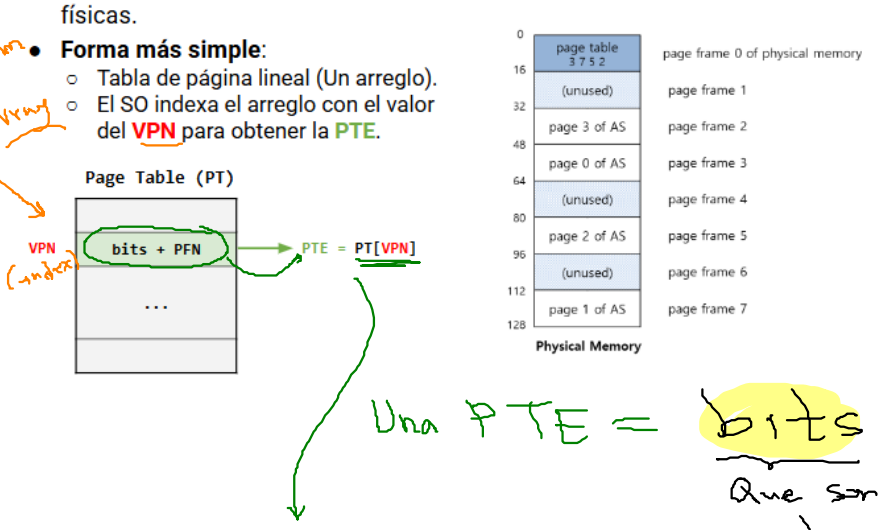
Las tablas de página son estructuras que se almacenan en memoria física en el **espacio kernel**.



5. Estructura de una tabla de pagina

Tabla de página (PT)

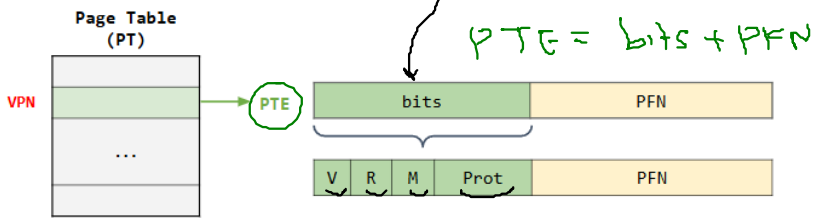
- **Estructura de datos** usada para mapear direcciones virtuales a direcciones físicas.
- **Forma más simple**:
  - Tabla de página lineal (Un arreglo).
  - El SO indexa el arreglo con el valor del **VPN** para obtener la **PTE**.



Una PTE = bits + PFN

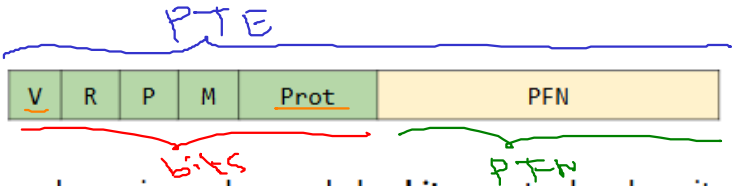
Que son?

Estructura de una PTE (Page Table Entry)

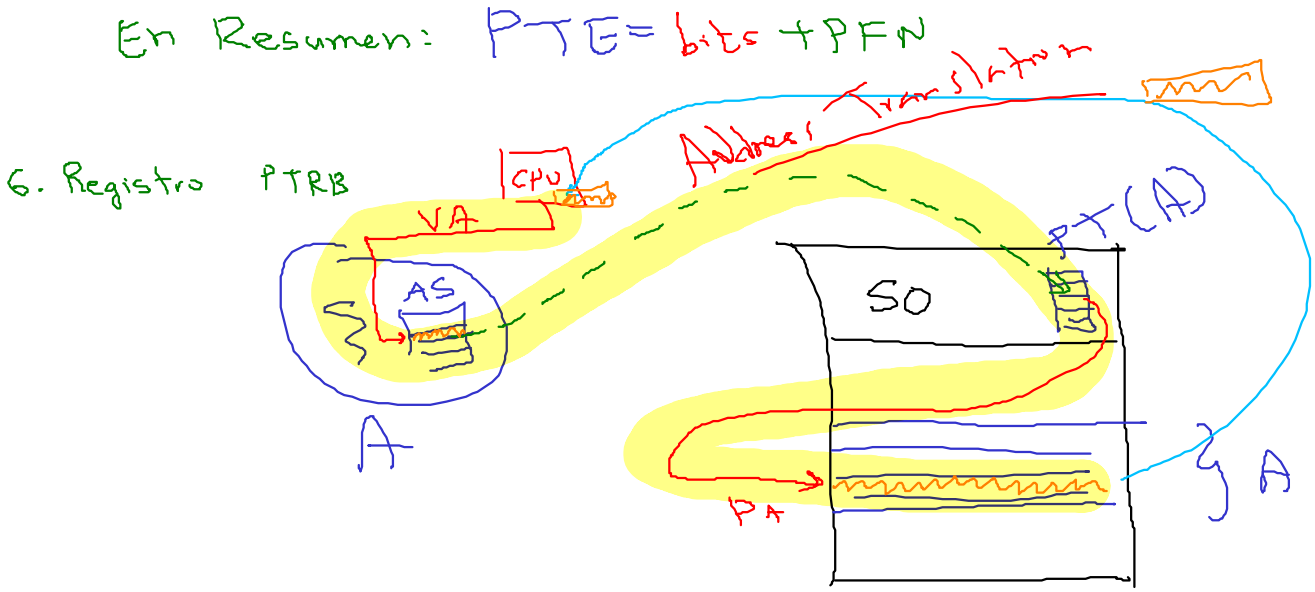


Estructura de una PTE (Page Table Entry)

- **Bit de validez (V):** Indica si la traducción es válida.
- **Bits de protección (Prot):** read - write - execute.
- **Bit de presencia (P):** Indica si la página está en la mem. física.
- **Bit sucio (M):** Indica si la página ha sido modificada.
- **Bit de referencia (R):** Indica si la página ha sido accedida.



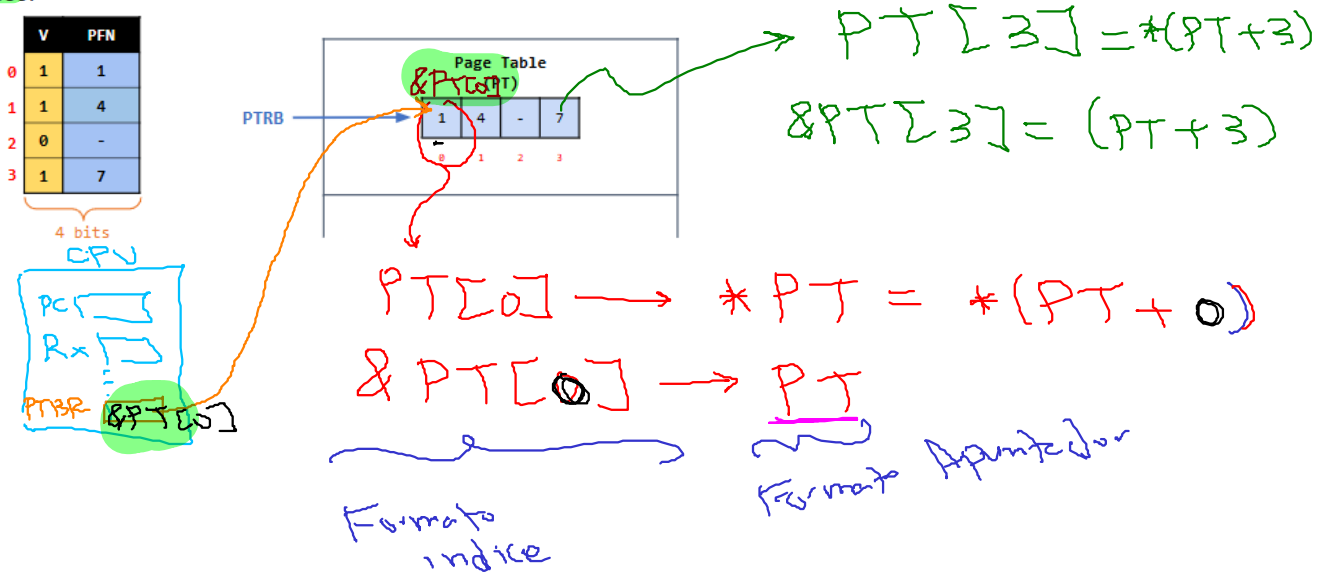
Para comprender mejor cada uno de los **bits** mostrados descritos anteriormente, vamos a analizar el siguiente ejemplo tratando de explicar de manera gradual el rol de cada uno de los **bits** previamente mencionados.



Registro PTRB

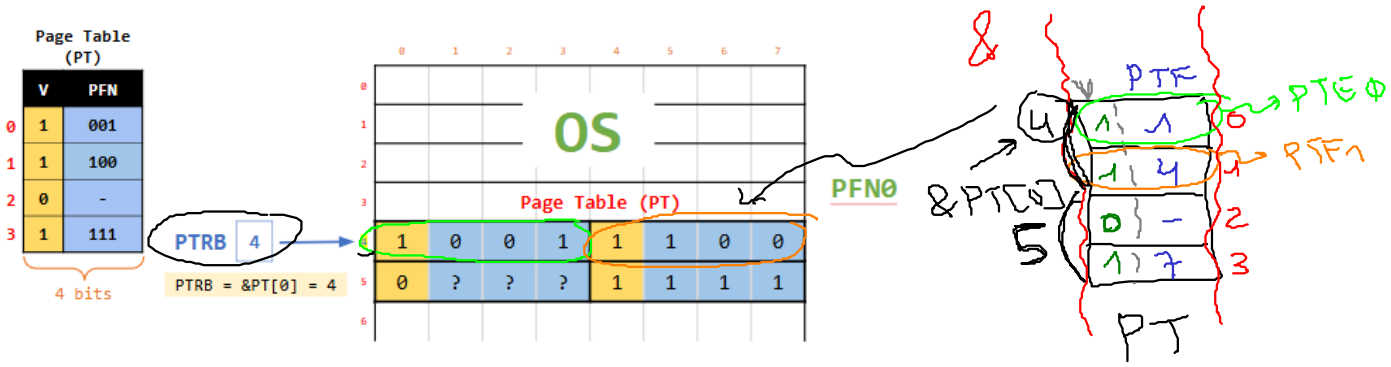
Problema: El hardware debe hacer la traslación pero ¿Dónde está la tabla asociada al proceso?

- Las **tablas de página (PT)** se encuentran dentro del espacio Kernel en la memoria física.
- Para conocer la ubicación de estas se emplea el **PTRB (Page Table Base Register)**.
- El registro PTRB mantiene la dirección física de la tabla de página asociada al proceso.



Registro PTRB

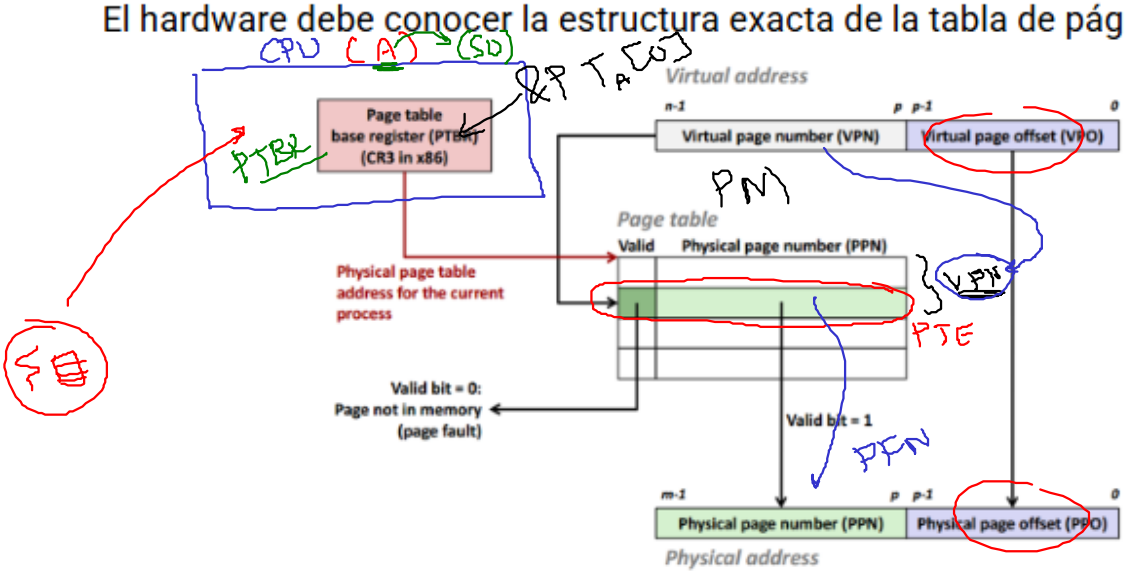
Suponiendo que PTRB = 4, entonces tenemos:



7. Traduccion de direcciones

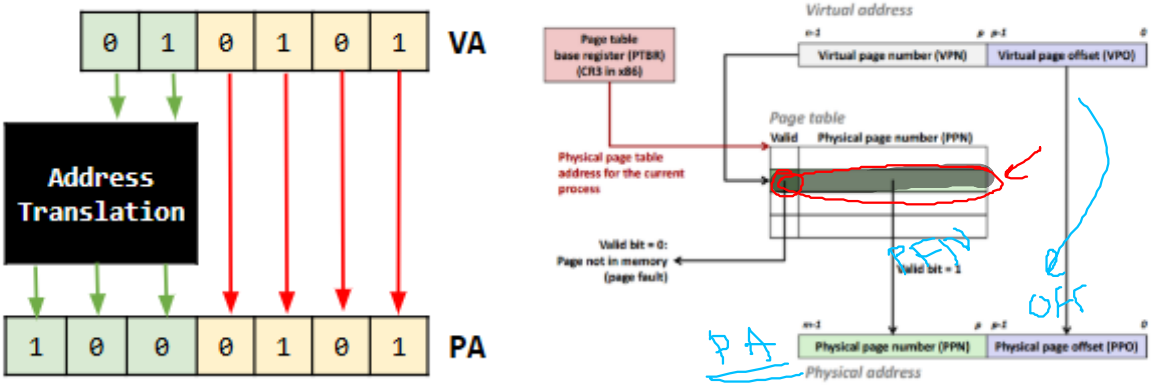
Hardware asociado

El hardware debe conocer la estructura exacta de la tabla de página (PT)



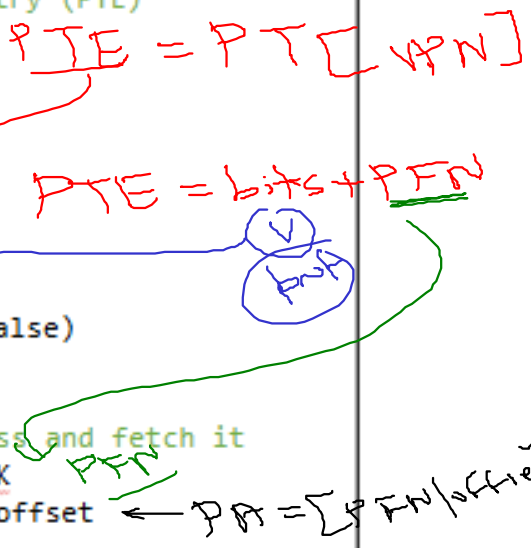
Lógica implementada resumida

```
index into PT: PTRB + VPN*size(PTE)
Fetch PTE for VPN;              // Traer la PTE
if (VALID == 0)
  fault();
PA = PFN << PAGESHIFT(offset)   // Realizar la traducción
Fetch PA → R1;                  // Cargar los datos desde la memoria física
```



Resumen protocolo completo

```
1  // Extract the VPN from the virtual address
2  VPN = (VirtualAddress & VPN_MASK) >> SHIFT
3
4  // Form the address of the page-table entry (PTE)
5  PTEAddr = PTBR + (VPN * sizeof(PTE))
6
7  // Fetch the PTE
8  PTE = AccessMemory(PTEAddr)
9
10 // Check if process can access the page
11 if (PTE.Valid == False)
12     RaiseException(SEGMENTATION_FAULT)
13 else if (CanAccess(PTE.ProtectBits) == False)
14     RaiseException(PROTECTION_FAULT)
15 else
16     // Access is OK: form physical address and fetch it
17     offset = VirtualAddress & OFFSET_MASK
18     PhysAddr = (PTE.PFN << PFN_SHIFT) | offset
19     Register = AccessMemory(PhysAddr)
```



30/09/2025 - Sistemas Operativos (UdeC)

8. Problemas de la paginación

Acceso simple a memoria

Código

array.c

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

gcc

Compilación y ejecución

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

En Asm

sizeof(int) = 4B

4B × 1000 = 4000B

K = 2^{10}B

M = 2^{20}B

Código ensamblador resultante

A

```
0x1024 movl $0x0,(%edi,%eax,4)   # %edi = &array; %eax = i → array[i] = 0
0x1028 incl %eax                 # %eax = %eax + i      → i = i + 1
0x102c cmpl $0x03e8,%eax         # 0x03e8 = 1000        → i - 1000 < 0 → sf = 1
0x1030 jne 0x1024                # Si no se llega a i = 1000 vaya al principio
```

1000

Acceso simple a memoria

Como el SO asigna la memoria para este proceso usando paginación

A

Código

1. Espacio de direcciones de 64KB y tamaño de página de 1KB. pages = 64
2. Tabla de página (tipo array) localizada en la dirección física 1KB (1024). PTBR = &PT[0] = 1K = 1024
3. Del enunciado, el código está desde 1KB (1024), luego este se encuentra dentro de la página cuya VPN = 1
4. El tamaño del array es de 1000 enteros → 4000 bytes.
5. Se asume que el rango de direcciones del array es: 40000 <= dir < 44000 ocupando las páginas virtuales 39, 40, 41 y 42.
6. Se asume el mapeo mostrado en la tabla a continuación.

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

Memory Map

code

array

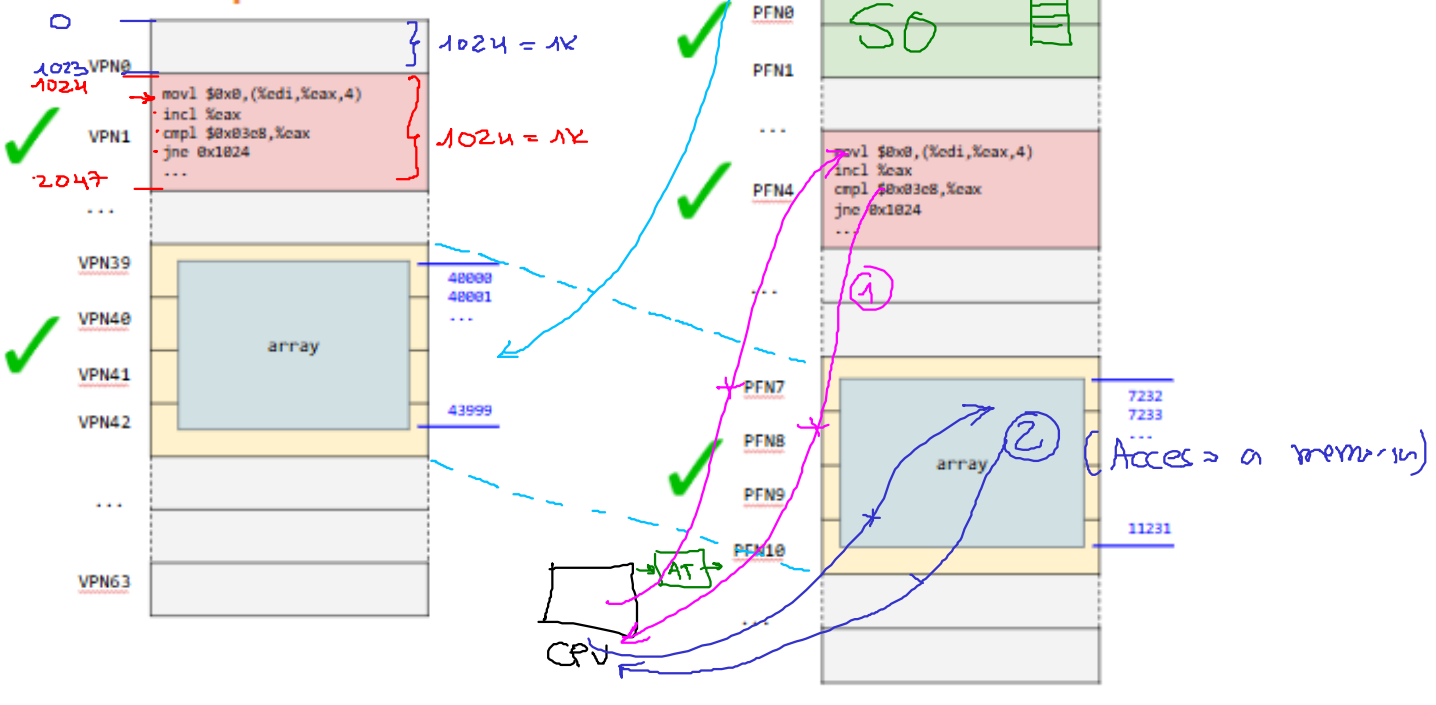
PT (Page table)

VPN	PFN
39	7
40	8
41	9
42	10

PT

PT[A]

Acceso simple a memoria



Acceso simple a memoria

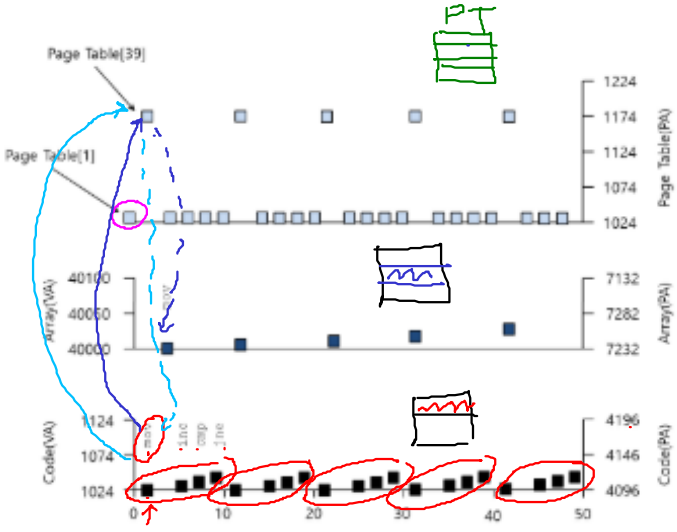
Análisis de la traza de memoria

Para la ejecución del código del programa, cada instruction fetch genera dos referencias a memoria:

- Acceso a la tabla de página (PT) para obtener la dirección física en la que reside la instrucción.
- A la dirección física en la que se encuentra la instrucción para poder llevarla a la CPU para su ejecución

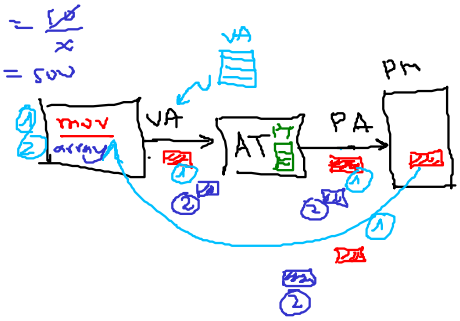
Instrucción	Accesos a memoria
`movl $0x0,(%edi,%eax,4)`	2 + 2 = 4
`incl %eax`	2
`cmpl $0x03e8,%eax`	2
`jne 0x1024`	2

5 primeros ciclos



5 → 50
100 → ?
5 = 50
x
x = 500

50



Solución Costosa

↓

Mucho viaje a memoria

↓

Solución: TLB