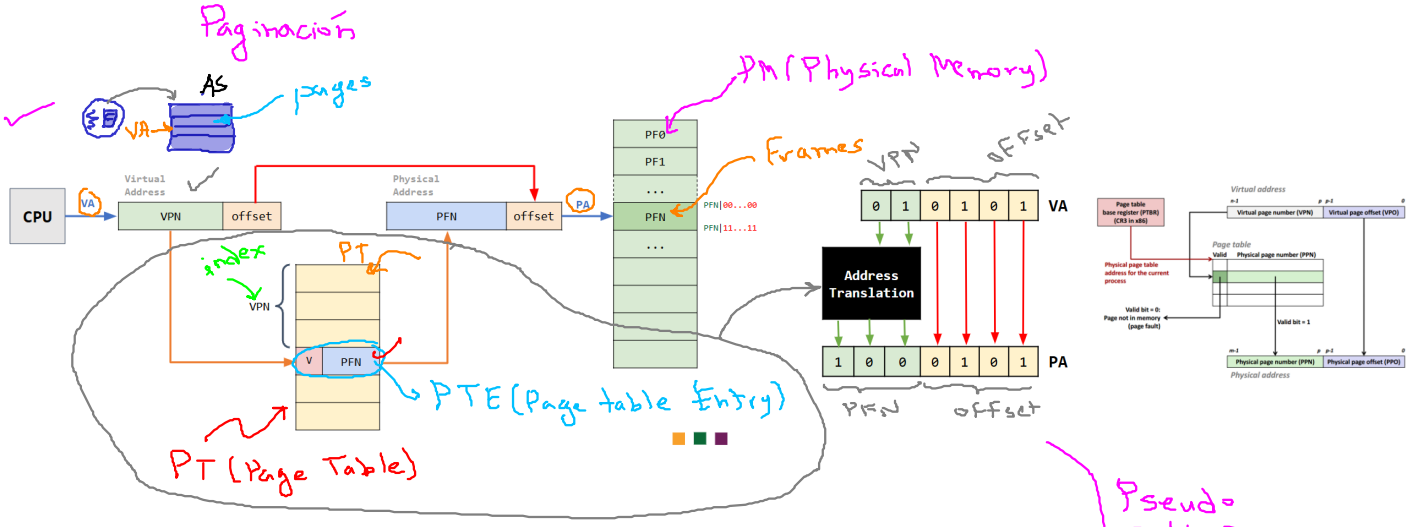
08/05/2025 - Sistemas Operativos - UdeC

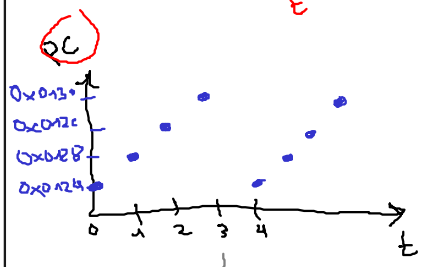
1. Repaso

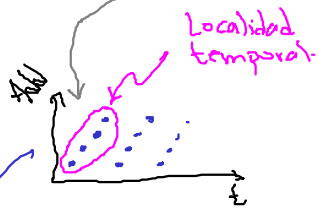
AS (Address space) = Memoria virtual



Pseudo codigo

```
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)
```





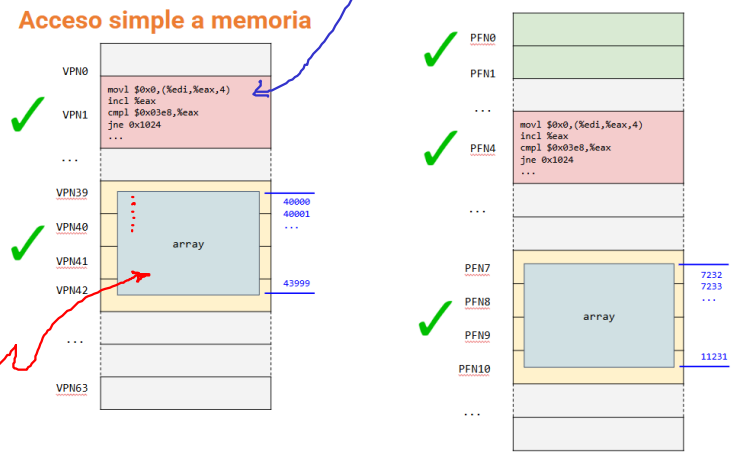
Acceso simple a memoria

Código

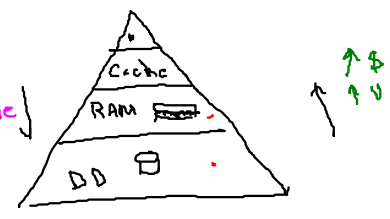
1. Espacio de direcciones de 64KB y tamaño de página de 1KB.
2. Tabla de página (tipo array) localizada en la dirección física 1KB (1024).
3. Del enunciado, el código está desde 1KB (1024), luego este se encuentra dentro de la página cuya VPN = 1
4. El tamaño del array es de 1000 enteros → 4000 bytes.
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
```

VPN	PFN
39	7
40	8
41	9
42	10

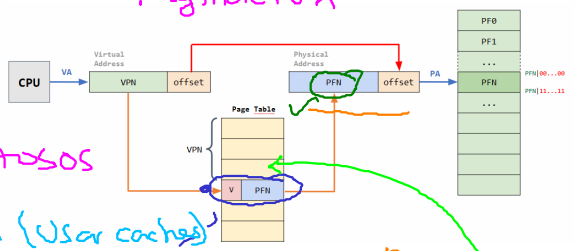


Como aprovechar las localidades espacial y temporal?



2. Que pasa cuando se hace una traduccion de direcciones al usar paginacion.

Paginación



Access a memoria → costosos

Solucion (Usar caches)

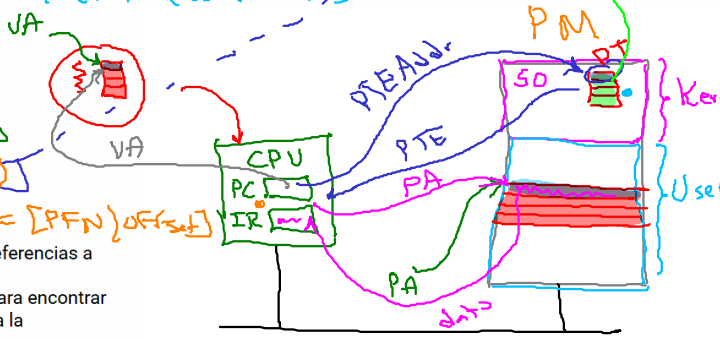
Pasos de la traducción de direcciones mediante paginación

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

VA = [VPN | offset]

PTE = [bits | PFN]

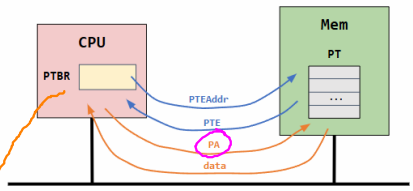
PA = [PFN | offset]



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.



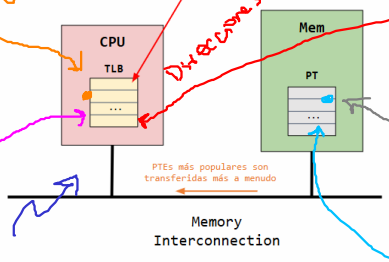
3. TLB

Usemos una cache.

Cache

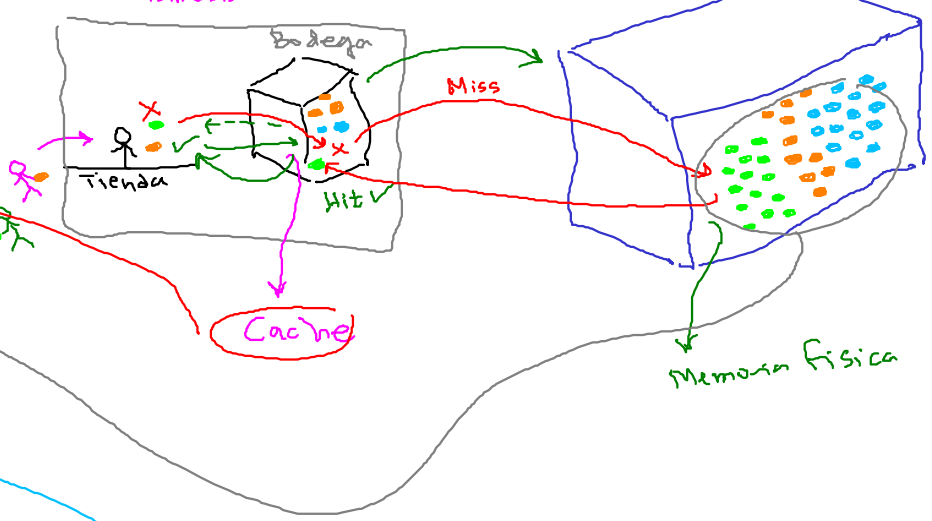
Direcciones

mas usadas



Tareas

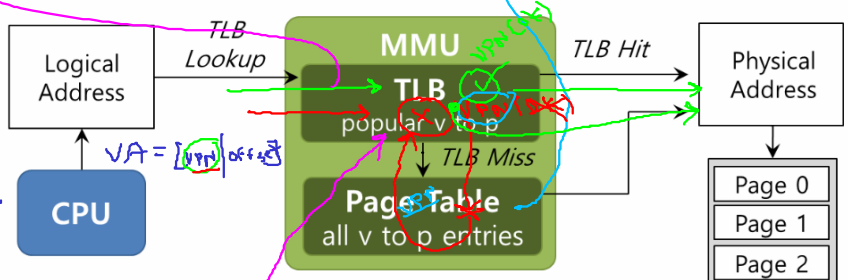
Medellin (Bodega Grande)



Hit → No acceso a PT

Miss → Acceso a PT (costoso)

PA = [PFN | offset]



(Figura 1)

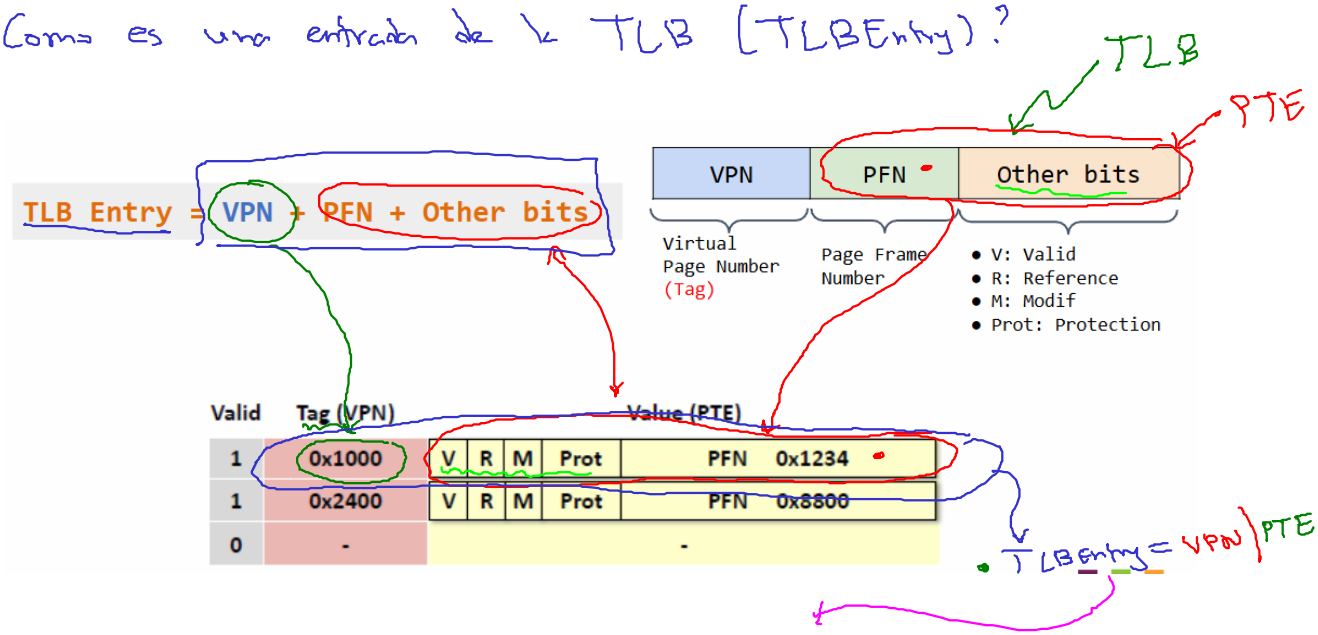
Address Translation with MMU

TLB = Tabla

TLBEntry

¿Cual es el formato de una TLB Entry?

4. Como es una entrada de la TLB (TLBEntry)?

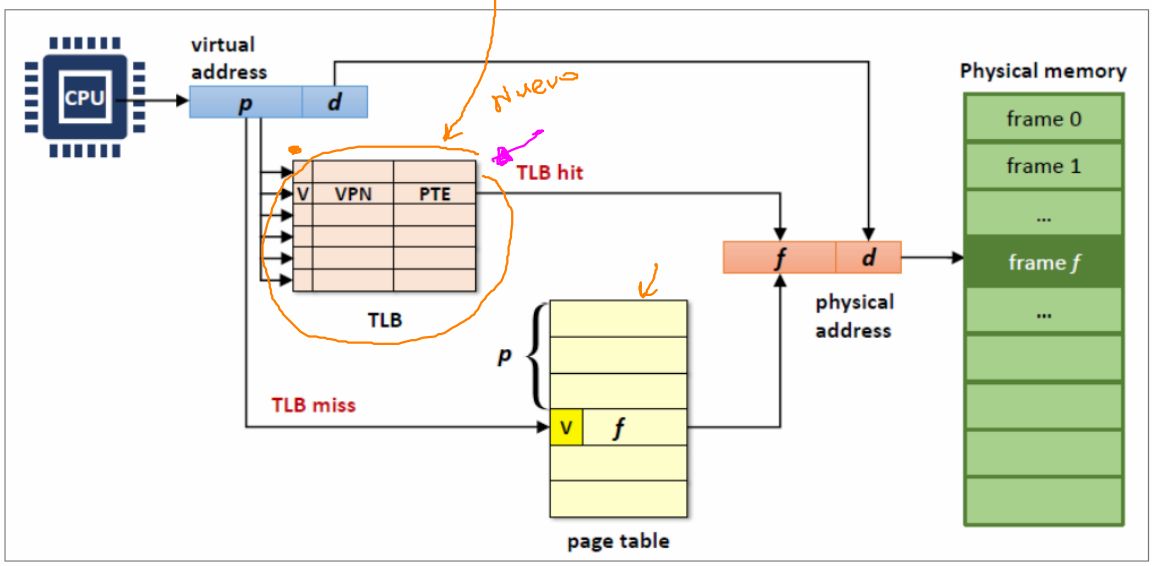


Como traducir direcciones

TLB + PT

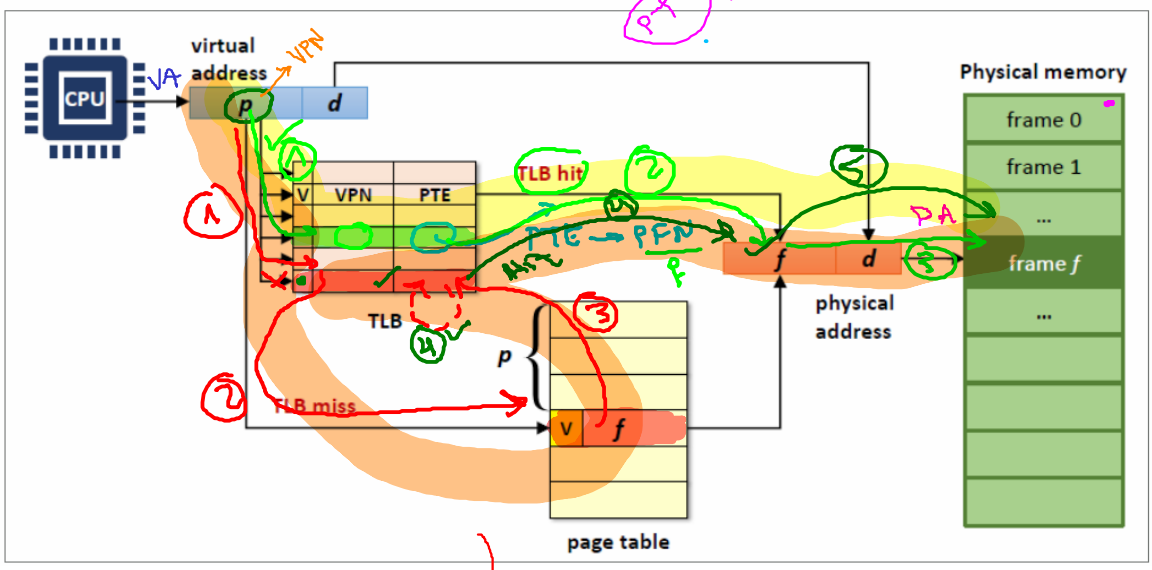
Nuevo

5. Como haga la traducción de direcciones teniendo en cuenta la TLB



Veamos el proceso de traducción -

1 Hit (VPN está en TLB)
2 Miss (VPN no está en TLB)



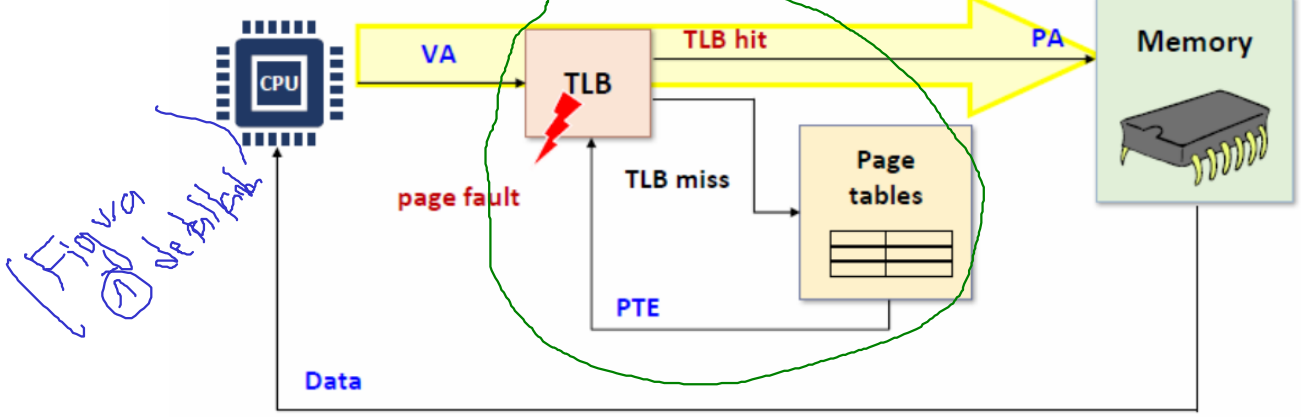
Pseudocódico que hace la Traducción de direcciones usando TLB

```
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 ( )

6. Resumen



Próxima clase → Cómo se aprecia la mejora gracias a la TLB?