POLITECNICO
MILANO 1863

Architettura dei calcolatori e sistemi operativi

M4 – Sistema operativo e Tabella delle pagine

Memoria virtuale del sistema operativo

Gestita tramite paginazione

- Indirizzi virtuali del SO vengono tradotti in indirizzi fisici da MMU

Problemi fondamentali:

- Sistema operativo è anche il gestore della memoria fisica
  - Deve poter accedere alla memoria in base a indirizzi fisici
- Le Tabelle delle Pagine hanno dimensioni molto grandi
  - Accesso da parte della MMU
  - Gestione da parte del sistema operativo

Soluzione: dipendente dalla MMU (x64)

Architettura Intel x64

L'architettura x64 ha uno spazio di indirizzamento potenziale **2^{64} Byte** con indirizzi virtuali da **64** bit

Lo spazio di indirizzamento virtuale utilizzabile è limitato a **2^{48} Byte**
⇨ **256 TByte:** indirizzi virtuali da **48** bit

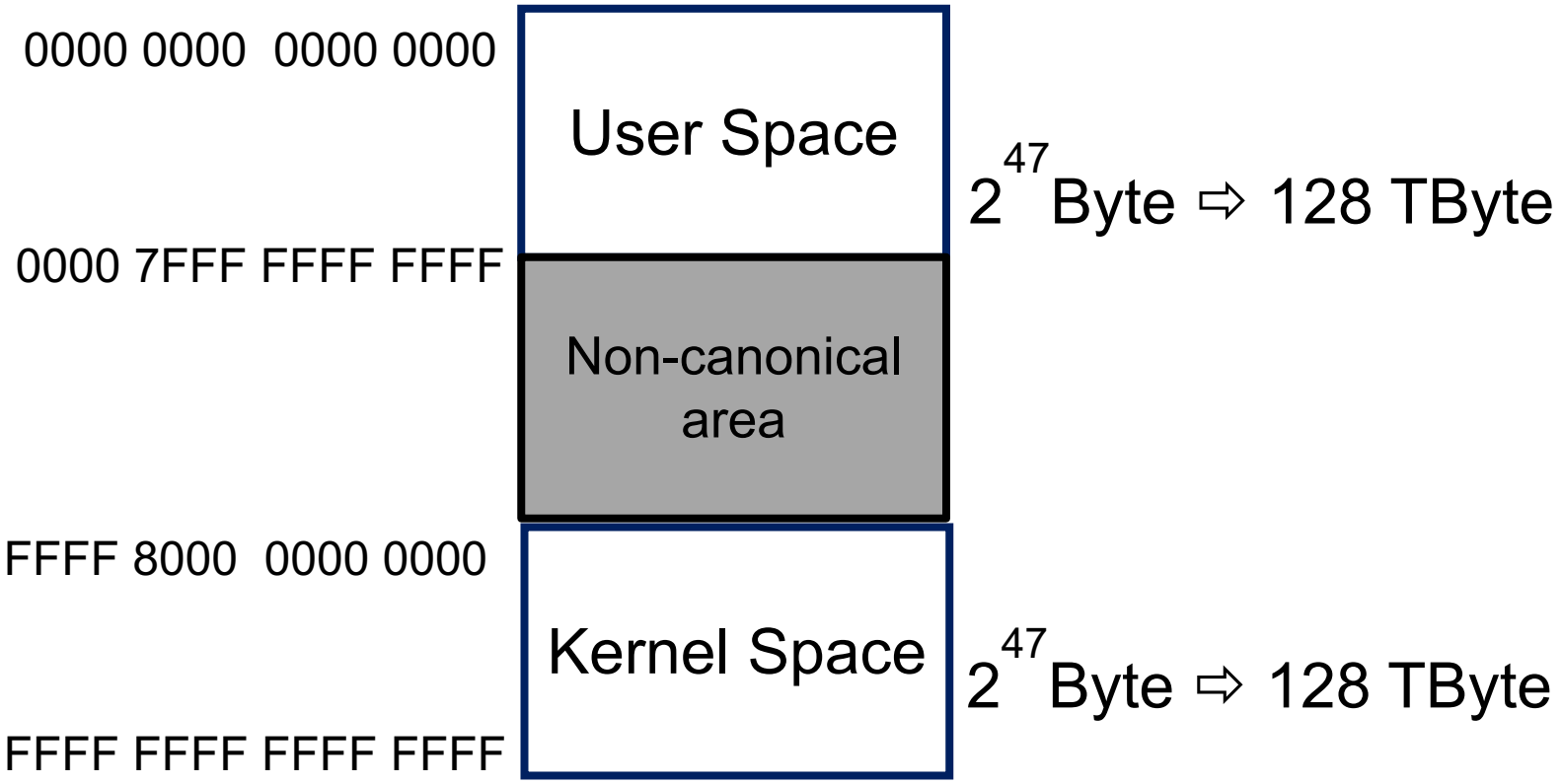
Dimensione di pagina: **4K Byte** ⇨ 2^{12} **Byte** ⇨ **12** bit per lo spiazzamento (offset)

Numero di pagine dello spazio di indirizzamento virtuale = **2^{48} / 2^{12} = 2^{36}** pagine virtuali ⇨ **36** bit per indicare il Numero di Pagina Virtuale **(NPV)**

Spazio di indirizzamento: Intel x64

Lo spazio di indirizzamento virtuale utilizzabile pari a **2^{48} Byte ⇨ 256 TByte**

suddiviso in due sottospazi di modo U ed S ambedue da 2^{47} Byte ⇨ 128 TByte



Struttura della memoria virtuale del Sistema Operativo

- Indirizzi virtuali del Kernel (2^{47} Byte)
  da FFFF 8000 0000 0000 a FFFF FFFF FFFF FFFF
- Lo spazio virtuale è suddiviso in 5 aree principali:
  - **codice e i dati** del sistema operativo: 0,5 Gb (512 Mb)
  - **moduli a caricamento dinamico:** 1,5 Gb
  - **dati dinamici del Kernel**: 32 Tb
  - **mappatura della memoria fisica:** 2^{46} byte = 64 Terabyte, (per permettere al codice del Kernel di accedere direttamente a indirizzi fisici)
  - **mappatura della memoria virtuale:** è un'area utilizzata per ottimizzare particolari configurazioni discontinue della memoria, e non verrà trattata
  - tra queste aree esistono spazi non utilizzati e lasciati per usi futuri

Memoria virtuale del kernel

Area	Costanti Simboliche	Indirizzo Iniziale (solo inizio)	Indirizzo finale (solo inizio)	Dimensione
unused space		**FFFF 8000 0000 0000**		
Mappatura memoria fisica	PAGE_OFFSET	FFFF 8800 0000 0000	FFFF C7FF 0000 0000	64 T Byte
unused space				1 T Byte
Memoria dinamica del kernel	VMALLOC_START	FFFF C900 0000 0000	FFFF E8FF 0000 0000	32 T Byte
unused space				
Mappatura memoria virtuale	VMEMMAP_START	FFFF EA00 0000		1 T Byte
unused space				
Codice e dati del SO	_START_KERNEL_MAP	FFFF FFFF 8000 0000		0,5 G Byte
Area per caricare dinamicam. i moduli	MODULES_VADDR	FFFF FFFF A000 0000		1,5 G Byte

Struttura della memoria virtuale del kernel

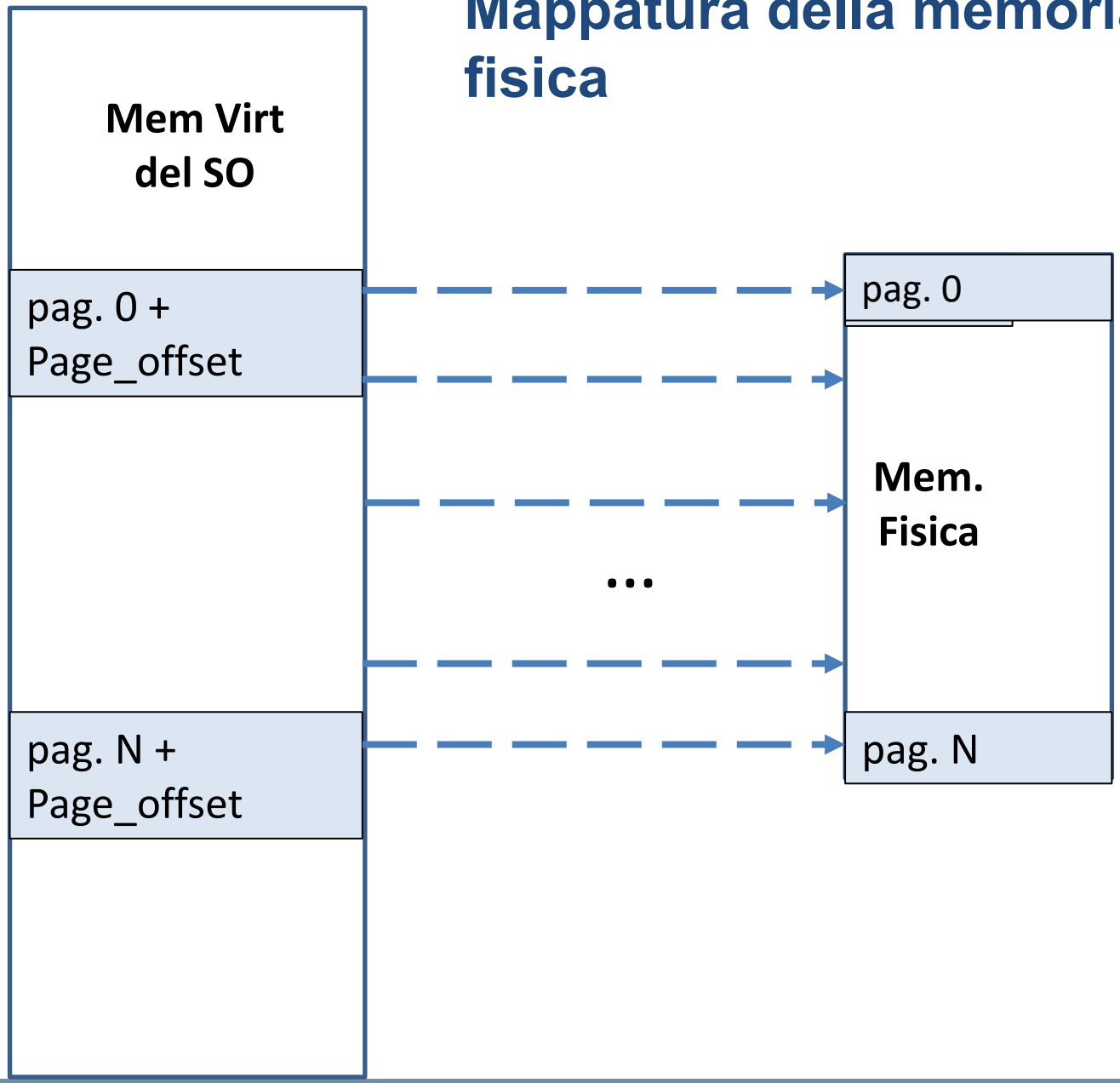
Area	Costanti simboliche per indirizzi iniziali	Indirizzo iniziale	Dimensioni
Mappatura memoria fisica	PAGE_OFFSET	`ffff 8800..`	64TB
Memoria dinamica del kernel	VMALLOC_START	`ffff c900..`	32TB
Mappatura della memoria virtuale	VMEMMAP_START	`ffff ea00..`	1TB
Codice e dati	_START_KERNEL_MAP	`ffff ffff 80..`	0,5GB
Area per caricare i moduli	MODULES_VADDR	`ffff ffff a0..`	1,5GB

Accesso agli indirizzi fisici da parte del SO

- Nella gestione della memoria il SO deve essere in grado di utilizzare gli indirizzi fisici, anche se, come tutto il software, esso opera su indirizzi virtuali.
- Esempio: nella Tabella delle Pagine gli indirizzi sono fisici, perché utilizzati nell'accesso alla memoria. Per operare sulla tabella delle pagine il SO deve quindi accedere alla memoria anche tramite indirizzi fisici.
- **Soluzione**:
  - dedicare una parte dello spazio virtuale del SO alla mappatura 1:1 della memoria fisica.
  - L'indirizzo iniziale di tale area virtuale è definito dalla costante **PAGE_OFFSET**, il cui valore varia nelle diverse architetture.
  - L'indirizzo virtuale PAGE_OFFSET corrisponde all'indirizzo fisico 0 e la conversione tra i 2 tipi di indirizzi è quindi

indirizzo fisico = indirizzo virtuale - PAGE_OFFSET
indirizzo virtuale = indirizzo fisico + PAGE_OFFSET

Mappatura della memoria fisica



Mappatura virtuale/fisica degli indirizzi

- Nel codice del SO esistono funzioni che eseguono la conversione tra indirizzi fisici e virtuali dell'area di rimappatura con gli opportuni controlli. Ad esempio, la funzione
  - unsigned long __phys_addr (unsigned long x)
  - esegue una serie di controlli e se tutto va bene restituisce x – PAGE_OFFSET

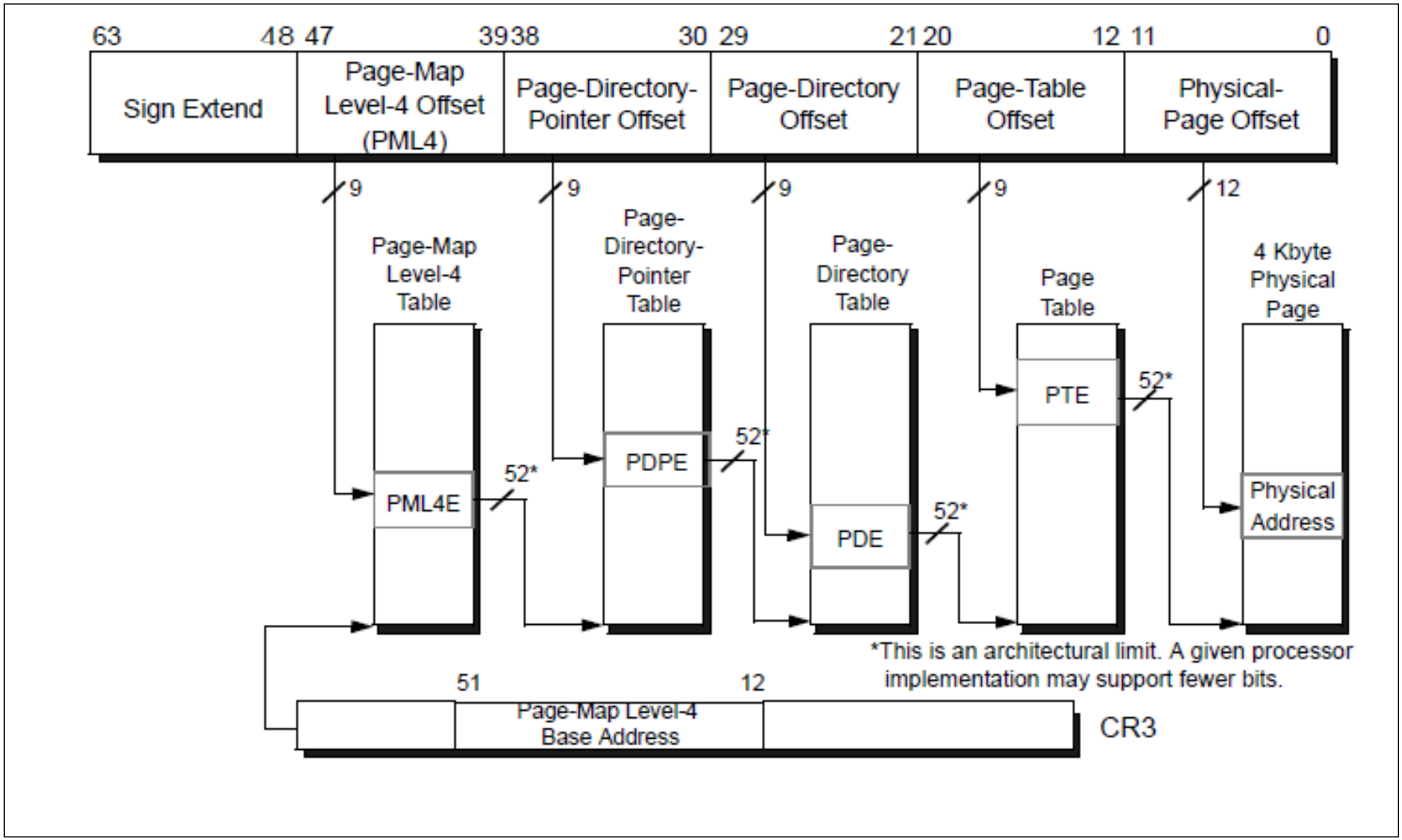
- Esempio: consideriamo gli indirizzi limite della sPila (8 K) di un processo
  - 0xFFFF 8800 5C64 **4**000 - inizio
  - 0xFFFF 8800 5C64 **6**000 - fine
- Confrontandoli con la mappa del SO vediamo che tali indirizzi si trovano nell'area di mappatura fisica del nucleo
- infatti le aree di sPila dei processi sono allocate dinamicamente dal SO e il loro indirizzo **fisico** viene ottenuto al momento di tale allocazione
  - tale indirizzo fisico viene trasformato in un indirizzo virtuale dell'area di mappatura della memoria fisica per essere utilizzato dal sistema

Paginazione nel x64

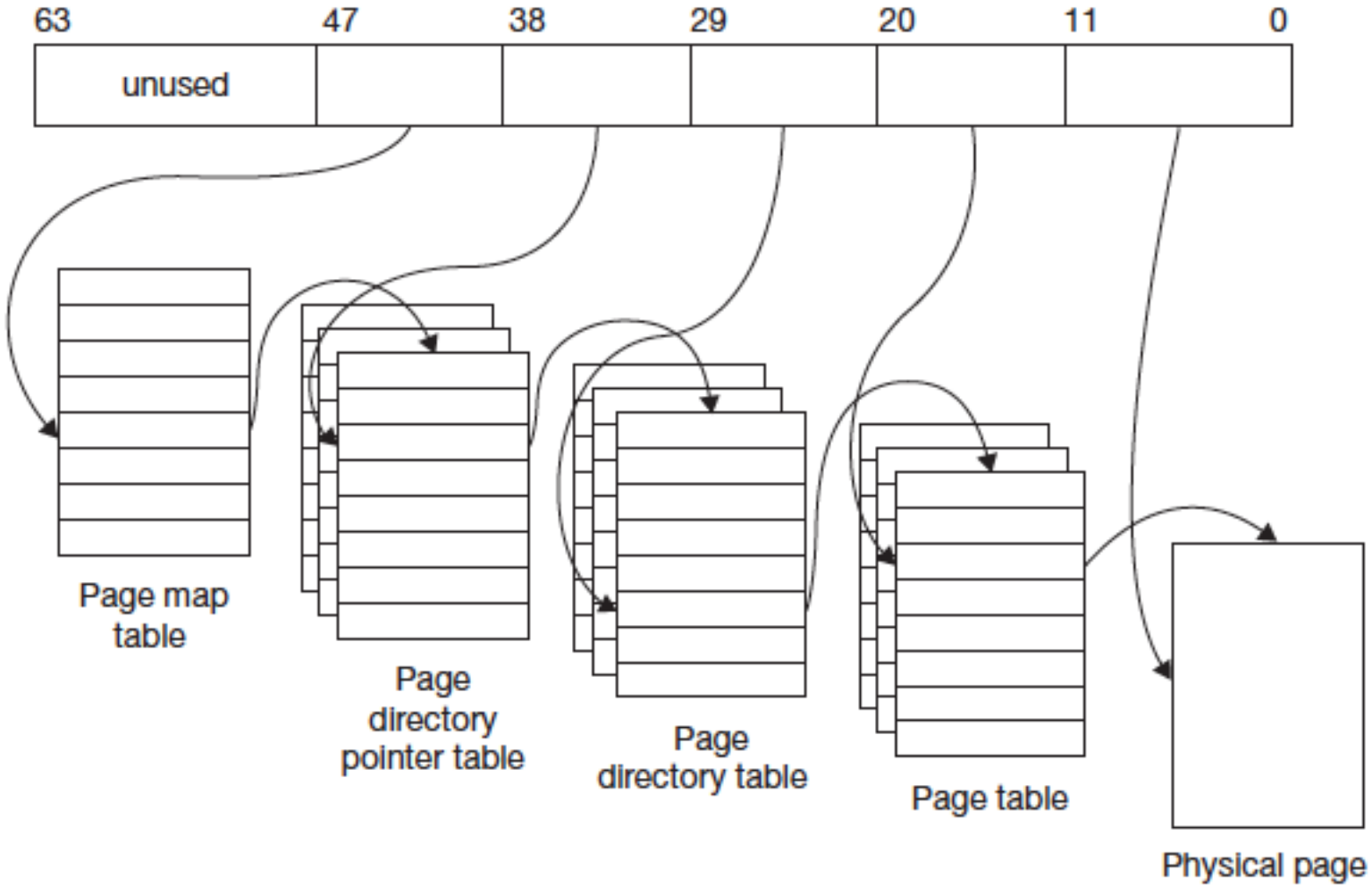
Numero di pagine virtuali 2^{36}

- Necessario evitare di allocare una tabella così grande per ogni processo ➔ Tabella organizzata ad albero su 4 livelli
  - i 36 bit del NPV sono suddivisi in 4 gruppi da 9 bit
  - ogni gruppo da 9 bit rappresenta lo spiazzamento (offset) all'interno di una tabella (**directory**) contenente 512 righe
  - Dato che ogni PTE occupa 64 bit (8 byte), la dimensione di ogni directory è di 4KB ➔ ogni directory occupa esattamente una pagina
  - L'indirizzo della directory principale è contenuto nel registro CR3 della CPU e viene caricato ogni volta che viene mandato in esecuzione un processo

Indirizzamento Tabella delle Pagine x64



Struttura della tabella delle pagine

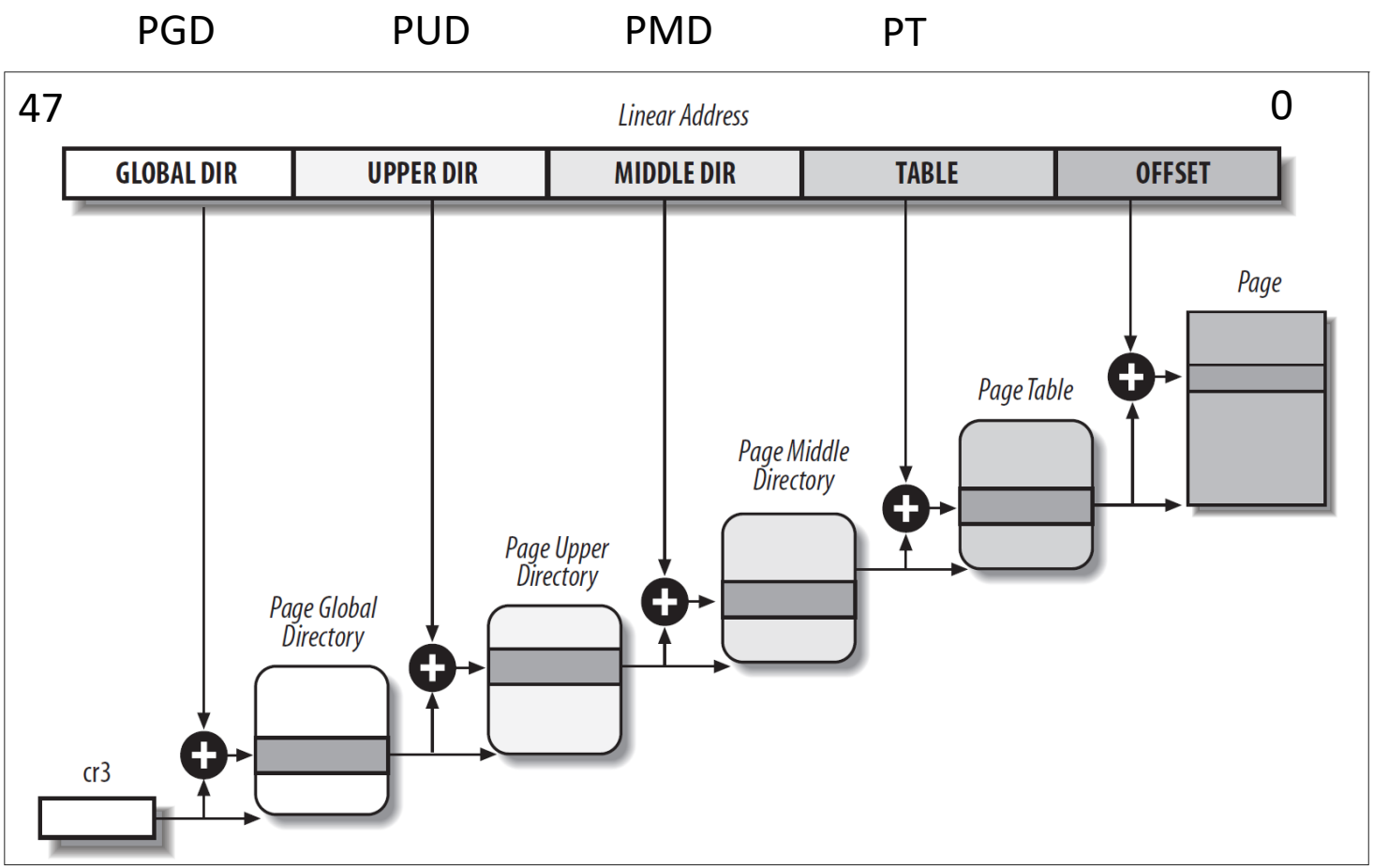


Traduzione di un indirizzo virtuale in indirizzo fisico in Linux

- Accedi alla directory di primo livello tramite CR3 (**global directory - PGD**)
- Trova la PTE indicata dall'offset di primo livello (bit 47-39) di NPV
- Leggi a tale offset l'indirizzo di base della directory di livello inferiore (**page upper directory - PUD**)
- Trova la PTE indicata dall'offset di secondo livello di NPV (bit 38-30)
- Leggi a tale offset l'indirizzo di base della directory di livello inferiore (**page middle directory - PMD**)
- Trova la PTE indicata dall'offset di terzo livello di NPV (bit 29-21)
- Leggi a tale offset l'indirizzo di base della directory di livello inferiore (**page table - PT**)
- Trova la PTE indicata dall'offset leggi l'indirizzo della pagina fisica (bit 20-12)
- I 12 bit meno significativi rappresentano 'offset all'interno della pagina fisica per individuare la parola cercata

- Questo schema richiedere di eseguire 4 accessi aggiuntivi a memoria per ogni accesso utile.
- Gli indirizzi contenuti nei diversi livelli di directory sono ***indirizzi fisici***, perché la MMU li utilizza direttamente per accedere la memoria fisica

Attraversamento delle pagine (Page Walk)



Attraversamento delle pagine

- Le PTE delle tabelle attraversate non utilizzano I 12 bit meno significativi per indirizzare la tabella del livello successivo (come non utilizzano i 16 bit più significativi dell'indirizzo da 64 bit)
  - i 12 bit meno significativi vengono utilizzati per **flag** che rappresentano proprietà di pagina
  - Significato di alcuni bit della tabella delle pagine:
    - Bit 0: P: present - PTE ha contenuto valido
    - Bit 1: R/W (1: pagina scrivibile, 0: pagina read only)
    - Bit 2: la pagina appartiene allo spazio User (1), Kernel (0)
    - Bit 5: A: accessed - la pagina è stata acceduta (azzerabile da sw)
    - Bit 6: D: dirty – la pagina è stata modificata (azzerabile da sw)
    - Bit 8: G: global page
    - Bit 63: NX: no execute Il bit più significativo è utilizzato per indicare se la pagina è eseguibile

Translation Lookaside Buffer (TLB)

- Quando il processore deve accedere un indirizzo fisico in base a un indirizzo virtuale, deve attraversare tutta la gerarchia della Tabella delle Pagine per trovare la PTE
  - ***Page Walk***, richiede 5 accessi a memoria per accedere a una sola cella utile di memoria.
- Per evitare questo inaccettabile numero di accessi l'architettura x64 possiede il TLB, che contiene le corrispondenze NPV-NPF più utilizzate recentemente
- Quando viene modificato il contenuto del registro CR3 la MMU invalida tutte le PTE del TLB, escluse quelle marcate come globali (flag G – bit 8), ossia condivise da più processi
- Esistono delle istruzioni privilegiate per controllare il TLB, ad esempio per invalidare una singola PTE.

Page fault e TLB Miss

- La MMU cerca autonomamente nella TP le PTE di pagine non presenti; quando viene richiesto un indirizzo virtuale, la MMU verifica se la pagina P richiesta è presente nel TLB
- Se la pagina è presente, la accede immediatamente
- Se la pagina non è presente si verifica un errore di TLB, o **TLB miss**.
- In questo caso la MMU deve:
  - eseguire il Page Walk (5 accessi a memoria) sulla TP per recuperare la PTE relativa alla pagina P
  - se questa è valida, il contenuto della PTE viene copiato nel TLB e la pagina viene acceduta
  - se questa non è valida (pagina non presente) si verifica un **page fault**

Paginazione in Linux

- La gestione della paginazione è una funzione che dipende in grande misura dall'architettura Hardware.
- Linux utilizza un modello parametrizzato per adattarsi alle diverse architetture.
- Linux caratterizza il comportamento dell'HW tramite una serie di parametri contenuti nei file di architettura, che indica per esempio:
  - La dimensione delle pagine
  - La struttura degli indirizzi
  - Il numero di livelli della tabella delle pagine e la lunghezza dei diversi offset
  - ecc…
- Il modello della memoria del x64 è aderente al modello Linux
- La Tabella delle Pagine è sempre residente in memoria fisica e mappa tutto lo spazio di indirizzamento del processo, user e kernel.

Paginazione del Sistema operativo

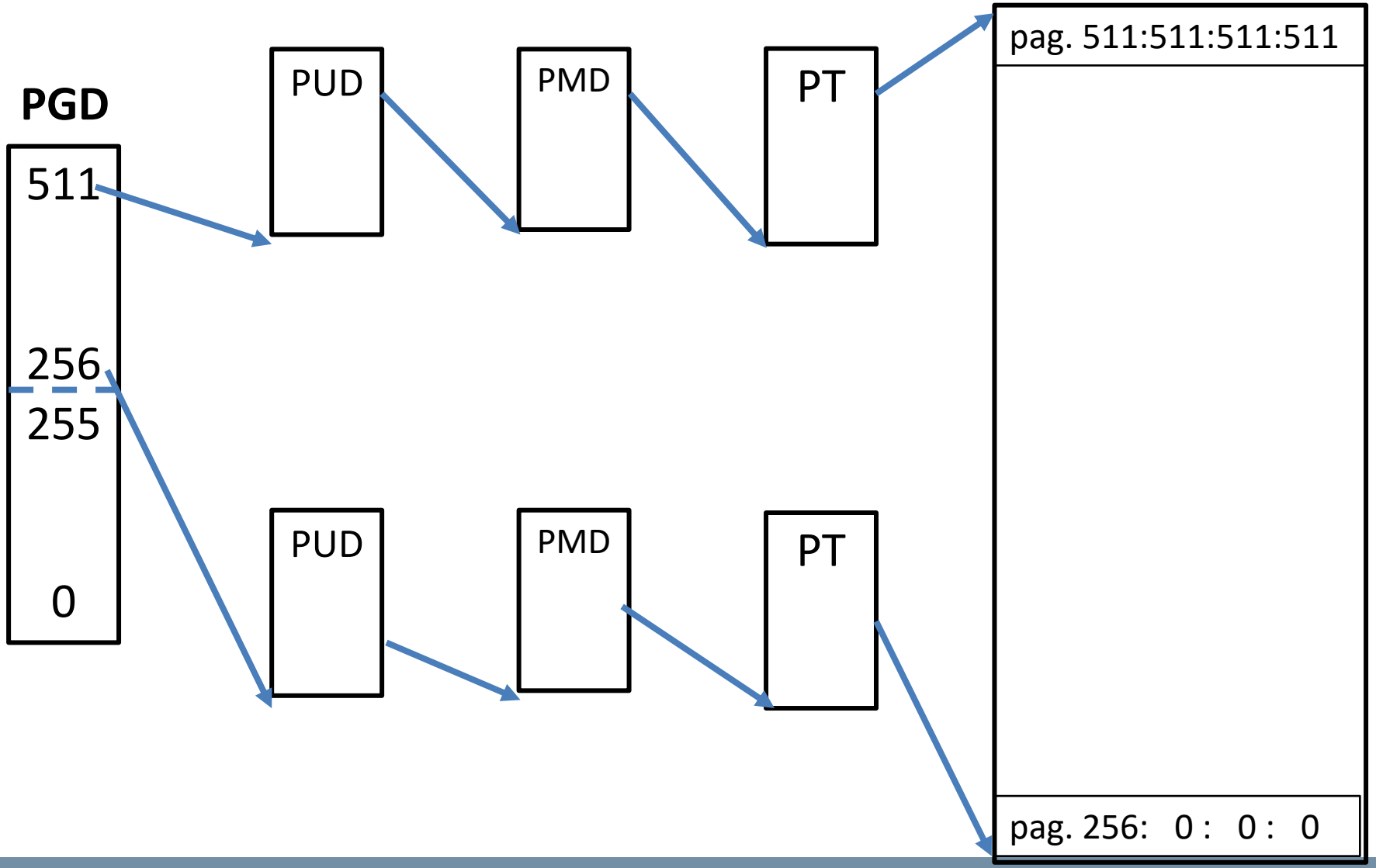
- Nel x64 tutta la memoria è paginata, indipendentemente dal modo di funzionamento della CPU, quindi anche il SO e la stessa Tabella delle Pagine sono paginati
- All'avviamento del sistema la Tabella delle Pagine non è ancora inizializzata, quindi deve esistere un meccanismo di avviamento che permetta di arrivare a caricare tale tabella per far partire il sistema:
  - all'avviamento del sistema x64 la paginazione non è attiva
  - Le funzioni di caricamento iniziale funzionano accedendo direttamente la memoria fisica, senza rilocazione
  - Quando è stata caricata una porzione della tabella delle pagine adeguata a far funzionare almeno una parte del SO, il meccanismo di paginazione viene attivato e il caricamento completo del Kernel viene terminato

Tabella Pagine del Kernel

- La tabella delle pagine attiva è quella puntata dal registro CR3, che è unico, quindi non può esistere una TP separata per il SO
- Dato che non esiste una TP del SO ma solamente quella dei processi, ***il SO viene mappato dalla TP di ogni processo***
- Lo spazio virtuale è suddiviso esattamente a metà tra modo U e modo S, la metà superiore della TP (righe 256 – 511 di ogni directory) di ogni processo è dedicata a mappare il kernel
- Questo fatto non genera ridondanza, perché tutte le metà superiori delle TP di ogni processo puntano alla stessa (**unica**) struttura di sottodirectory relativi al SO

Tabella delle Pagine

Sistema operativo



Dimensione della tabella delle pagine

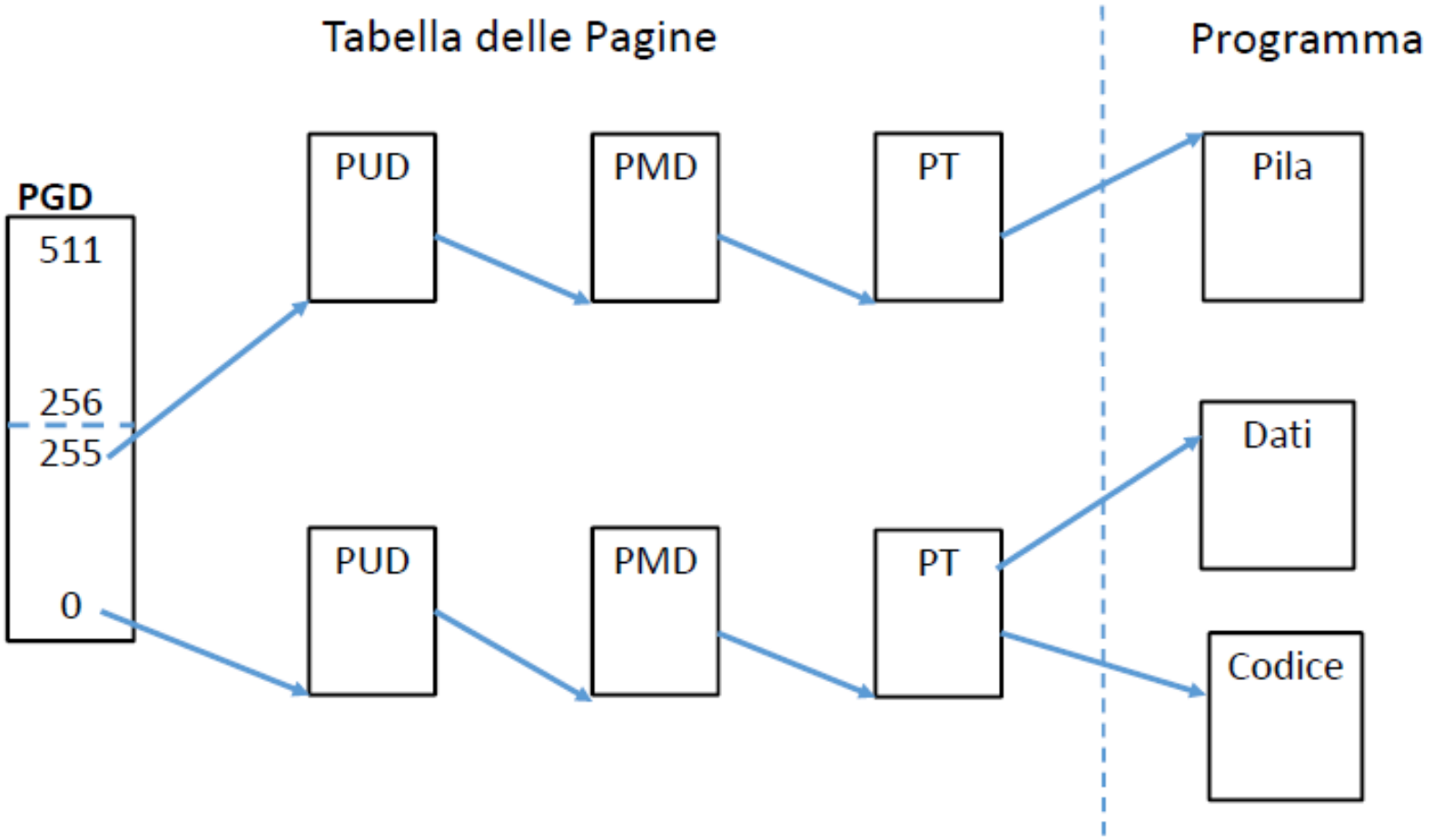
Dimensione di memoria resa accessibile da una singola PTE ai diversi livelli della gerarchia :

- una PTE di PMD accede una pagina di PT, quindi 512 x 4K = 2M
- una PTE di PUD accede una pagina di PMD, quindi 512 x 2M = 1G
- una PTE di PGD accede una pagina di PUD, quindi 512 x 1G = 512G = 0,5T

Esempio 1

- programma molto piccolo, costituito da una sola pagina di codice, una di dati e una pagina di pila, con una dimensione complessiva di 3 pagine:
- nel PGD, che occupa una sola pagina, sono sufficienti 2 PTE, una posta all'indice 0 per mappare codice e dati e l'altra all'indice 255 per mappare la pila (le PTE da 256 a 511 sono dedicate a mappare il SO)
- Ai livelli inferiori, fino alla PT, sono sufficienti 2 pagine per livello, ognuna contenente una PTE, quindi in totale la TP occupa 7 pagine
- il programma occupa 3 pagine
- Il rapporto tra le dimensioni della TP e quelle del programma risulta 7/3, cioè la TP occupa addirittura molto più spazio del programma

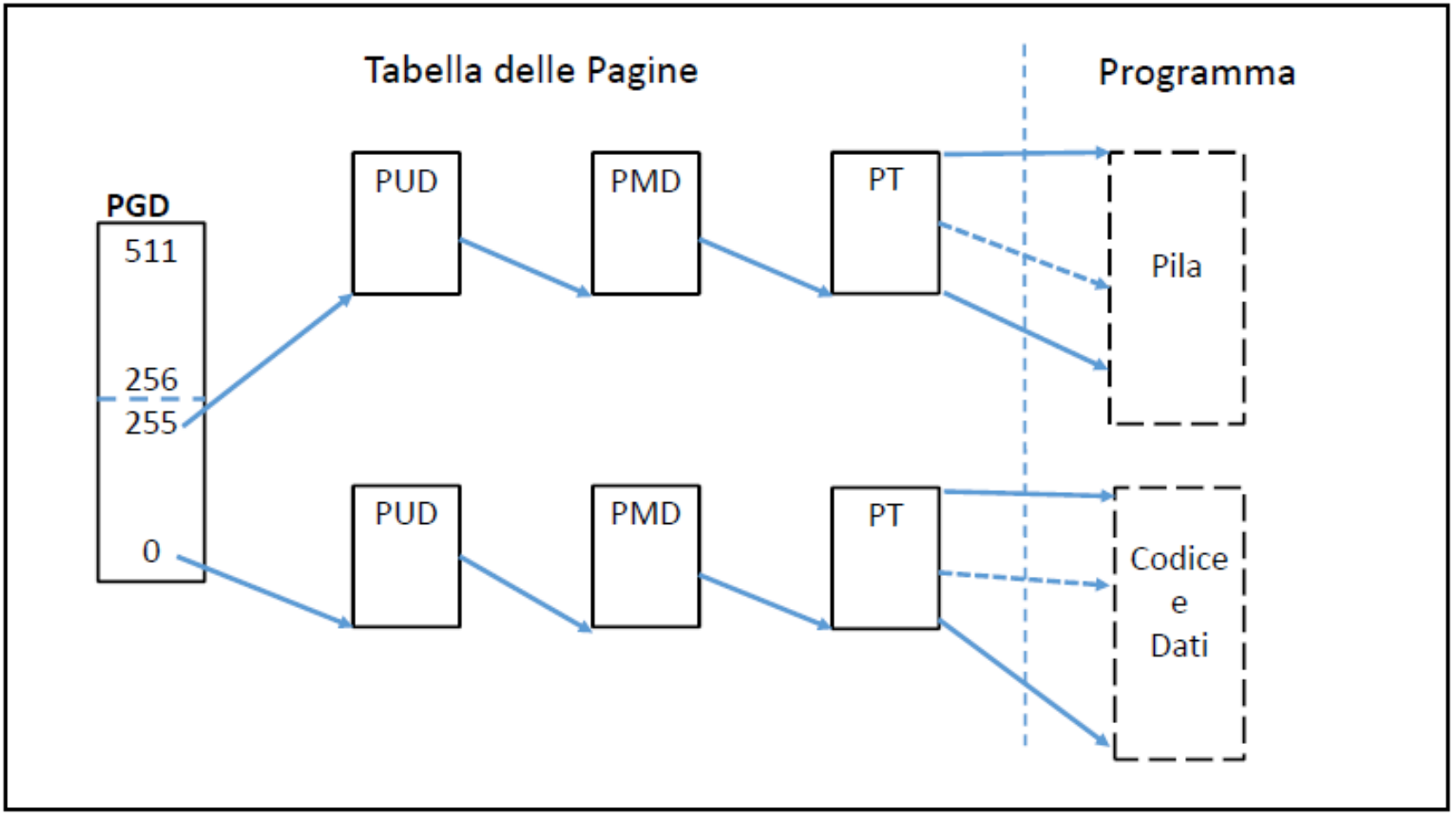
Page walk: esempio 1



Esempio 2

- Se si aumentano le dimensioni del programma dell'esempio precedente fino al limite massimo possibile senza aumentare le dimensioni della Tabella delle Pagine:
  - Le due aree del programma possono aumentare fino a richiedere l'uso di tutte le PTE (512) di ognuna delle due pagine di PT già allocate
  - La dimensione del programma può quindi crescere fino a 2x512 = 1024 pagine senza aumentare il numero di pagine dedicate alla Tabella delle Pagine
- Il rapporto tra le dimensioni della TP e quelle del programma risulta ora 7/1024 = 0,0068, cioè 0,68%.

Page walk: esempio 2



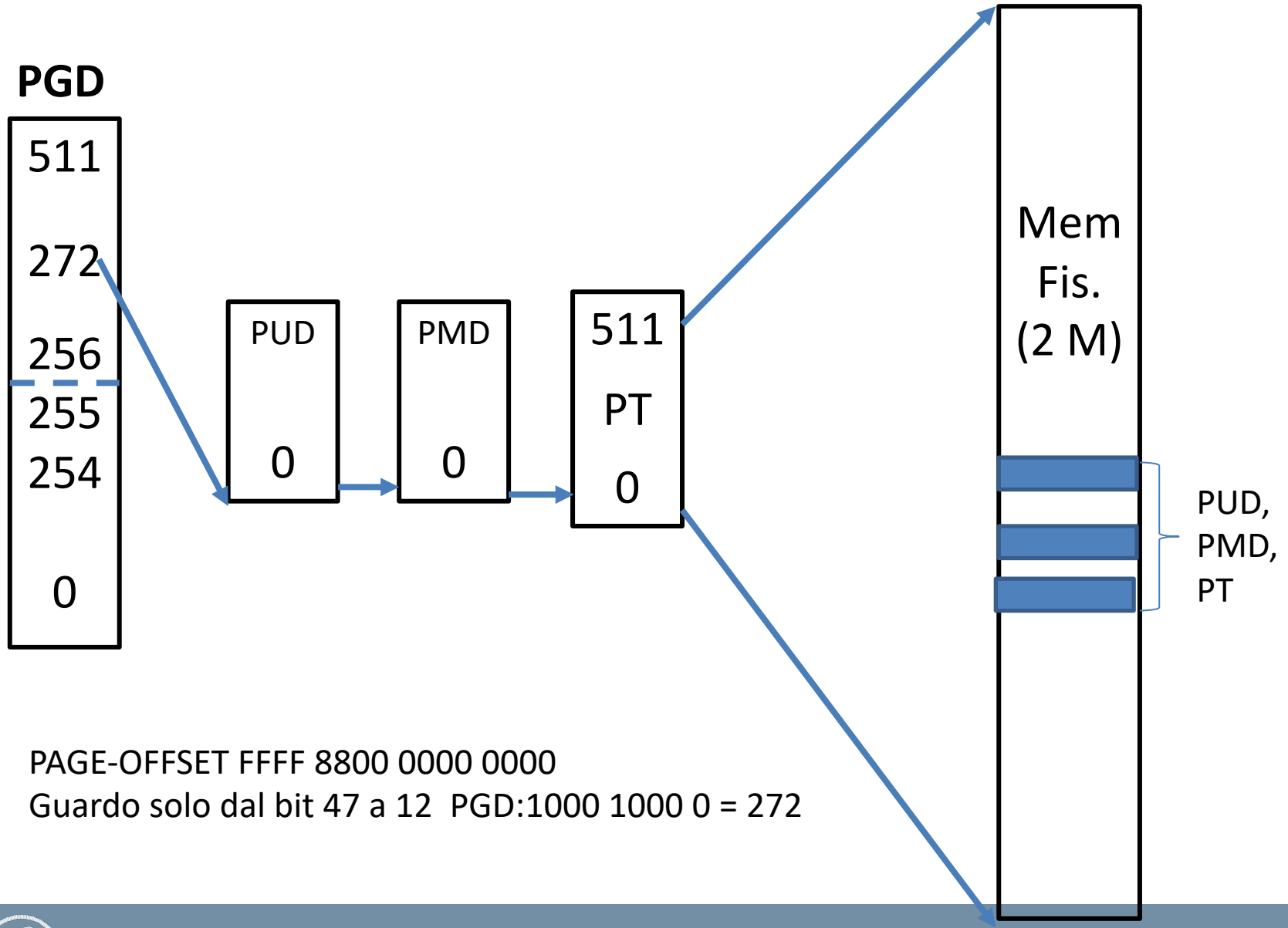
Considerazioni sugli esempi

- I due esempi mostrano che l'occupazione percentuale di memoria dovuta alla Tabella delle Pagine rispetto al programma diminuisce per programmi grandi fino a scendere nettamente sotto 1% già per un programma che satura solamente l'ultimo livello di directory.
- Aumentando ulteriormente la dimensione del programma il peso delle pagine che occupano i livelli superiori di directory diventa percentualmente sempre meno rilevante,
- Il rapporto dimensionale tra Tabella delle Pagine e programma tende a quello fondamentale tra una pagina di PT e la sua area indirizzabile, cioè 1/512 = 0,00195, cioè circa lo 0,2%.
- Si osservi che questo rapporto è lo stesso che esiste tra la dimensione di una PTE necessaria ad indirizzare una pagina e la dimensione della pagina, cioè 8 byte/4 Kbyte = 1/512.

Occupazione della Tabella delle Pagine relativa alla mappatura della memoria fisica

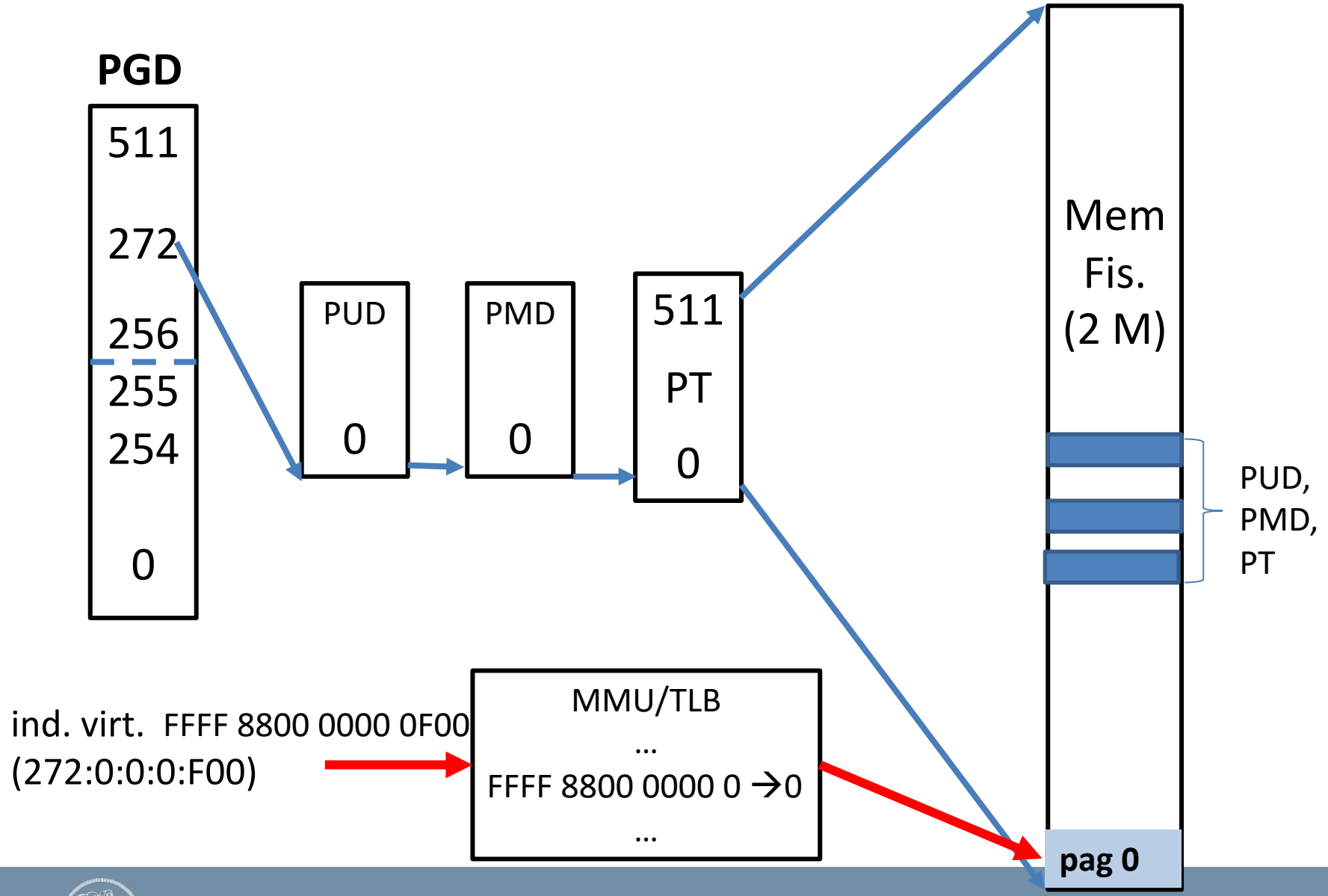
- Nell'area virtuale del kernel una parte è utilizzata per la mappatura della memoria fisica, che consente al SO di accedere direttamente a indirizzi fisici
  - la dimensione virtuale di quest'area del SO è 2^{46} byte = 64 Terabyte (PAGE_OFFSET da ffff 8800 …. a ffff c7ff .....
- La dimensione effettivamente mappata è determinata dalla dimensione effettiva della memoria fisica.
- Il rapporto tra la dimensione della porzione di Tabella delle Pagine interessata e la dimensione della memoria fisica tende a 1/512.
  - per esempio, su una macchina con 4Gbyte di memoria (1M di pagine), la TP occupa 8 Mbyte (2K pagine).
  - la Tabella delle Pagine occupa uno spazio significativo, ma sempre percentualmente accettabile rispetto allo spazio disponibile.

TP per la mappatura di una memoria di 2 M



PAGE-OFFSET FFFF 8800 0000 0000
Guardo solo dal bit 47 a 12 PGD:1000 1000 0 = 272

Accesso a un indirizzo di memoria fisica (di 2 M)



Esplorazione della Page Global directory

- Si consideri un processo utente che crea una pagina di codice e una di pila
- E' possibile scrivere un kernel module che esplora la Page Global Directory (PGD) e stampa i suoi contenuti validi (bit Present = 1 – bit meno significativo della PTE)
- Il risultato sarà dato da:
    - Riga della PGD che consente di indirizzare la pagina di codice del processo (modo U: righe tra 0 e 255)
    - Riga della PGD che consente di indirizzare la pagina di pila del processo (modo U: righe tra 0 e 255)
    - Pagine del sistema operativo allocate (nella parte della PGD di kernel tra riga 256 e 511)

Aree di memoria User

- Pagina di codice inizia a 0x0000 0000 0040 0000
- Pagina di dati inizia a 0x0000 0000 0060 0000

- Per individuare la riga di PGD corrispondente si considerano i 9 bit da 47 a 39 (si trascurano le prime 4 cifre esadecimali):
  - sia codice che dati hanno 0000 0000 0: riga 0 della PGD

- La pagina di pila inizia all'indirizzo 0x0000 7fff ffff ffff

- La riga di PGD corrispondente: bit 0111 1111 1: riga 255 della PGD

Aree di memoria del kernel

- Le 5 aree del kernel sono sempre allocate e sono sempre nello spazio del processo
- Si considerano i 9 bit che rappresentano lo spiazzamento nel PGD, bit 47 – 39 (si scartano le prime 4 cifre esadecimali)

Costanti Simboliche indirizzi iniziali	Indirizzo Iniziale	9 bit 47-39	valore decimale
PAGE_OFFSET	ffff **880**0..	1000 1000 0	272
VMALLOC_START	ffff **c90**0..	1100 1001 0	402
VMEMMAP_START	ffff **ea0**0..	1110 1010 0	468
_START_KERNEL_MAP	ffff **ffff** 80..	1111 1111 1	511
MODULES_VADDR	ffff **ffff** a0..	1111 1111 1	511

Decomposizione dell'indirizzo virtuale

Indirizzo virtuale: **0x00007fffb0d42118**

Indirizzo virtuale in binario (48 bit meno significativi):
0111 1111 1111 1111 1011 0000 1101 0100 0010 0001 0001 1000

PGD	0111 1111 1	255
PUD	111 1111 10	510
PMD	11 0000 110	390 = 2+4+128+256
PT	1 0100 0010	322 = 2 + 64 +256
OFFSET	0001 0001 1000	280 = 8 + 16 + 256

Notazione:
255:510:390:322

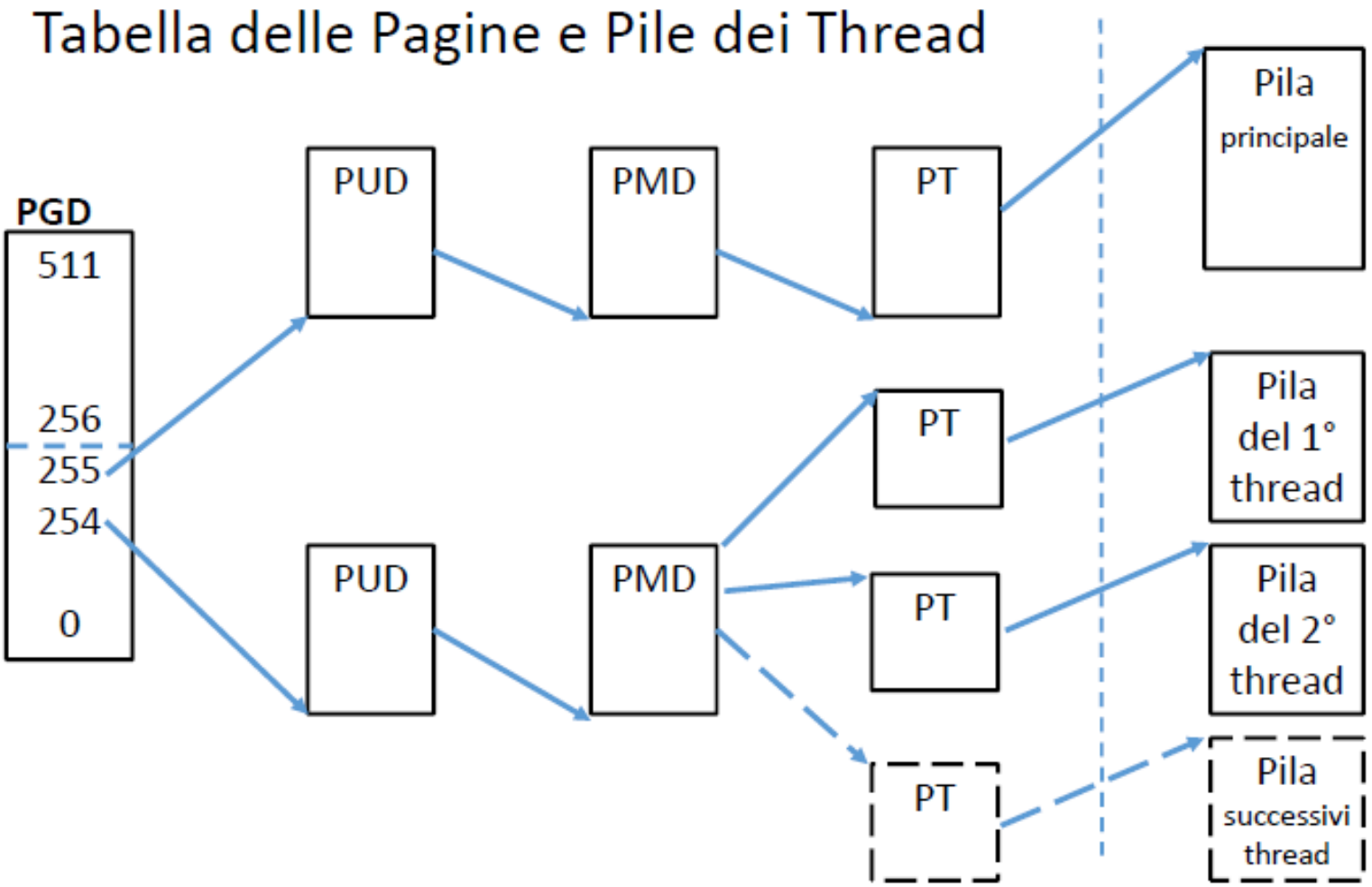
Tabella delle pagine di processi e thread

I thread condividono la stessa tabella delle pagine del processo padre (thread principale)

Struttura della tabella delle pagine per un processo:

- indirizzo iniziale del codice (..0000 0040 0000) → 0:0:2:0
- indirizzo iniziale dei dati (..0000 0060 0000) → 0:0:3:0
- Indirizzo iniziale della pila del processo (7fff ffff ffff) → 255:511:511:511
- indirizzo iniziale delle pile degli eventuali thread (7f7ffff ffff) → 254:511:511:511
    - Dimensione della pila per un thread: 8 MB
        - richiedono 2^{11} (2048) pagine da 4 KB, che sono indirizzate da 4 PT (ognuna da 512 elementi) + 1
        - Quindi richiedono 4 PT + 1 pagina ognuno
        - Le PTE del PMD per le pile di 2 thread consecutivi sono distanziate di 4 posizioni

Le pile dei thread



Esempio

Data la struttura di VMA di un processo P mostrare la struttura della Tabella delle Pagine a livello delle singole directory, considerando che per ogni pila dei thread si allocano solo 2 PTE

VMA	Start Address	Dim	R/W	P/S	M/A	mapping
C	000000400000	3	R	P	M	<X,0>
K	000000600000	1	R	P	M	<X,3>
S	000000601000	4	W	P	M	<X,4>
D	000000605000	2	W	P	A	<-1,0>
M0	000010000000	2	W	S	M	<G,2>
M1	000020000000	1	R	S	M	<G,4>
M2	000030000000	1	W	P	M	<F,2>
M3	000040000000	1	W	P	A	<-1,0>
T1	7FFFF77FBFFF	2	W	P	A	<-1,0>
T0	7FFFF77FEFFF	2	W	P	A	<-1,0>
P	7FFFFFFFCFFF	3	W	P	A	<-1,0>

Struttura della Tabella delle Pagine

VMA C: 3 pagine a partire da 0000 0040 0000 (48 bit hex)

PGD: 0000 0000 0 = 0

PUD: 000 0000 00 = 0

PMD: 00 0000 010 = 2

PT: 0 0000 0000 = 0

Prima pagina 0: 0: 2: 0

Seconda 0: 0: 2: 1

Terza: 0: 0: 2: 2

Struttura della Tabella delle Pagine

VMA K: 1 pagina a partire da 0000 0060 0000 (48 bit)

PGD: 0000 0000 0 = 0

PUD: 000 0000 00 = 0

PMD: 00 0000 011 = 3

PT: 0 0000 0000 = 0

Prima pagina 0: 0: 3: 0

VMA S: 4 pagine a partire da 0000 0060 1000 (48 bit)

PGD: 0000 0000 0 = 0

PUD: 000 0000 00 = 0

PMD: 00 0000 011 = 3

PT: 0 0000 0001 = 1

Prima pagina	0: 0: 3: 1
Seconda	0: 0: 3: 2
Terza	0: 0: 3: 3
Quarta	0: 0: 3: 4

Struttura della Tabella delle Pagine

VMA D: 2 pagine a partire da 0000 0060 5000 (48 bit)

PGD: 0000 0000 0 = 0

PUD: 000 0000 00 = 0

PMD: 00 0000 011 = 3

PT: 0 0000 0101 = 5

Prima pagina 0: 0: 3: 5

Seconda: 0: 0: 3: 6

VMA M0: 2 pagine a partire da 0000 1000 0000 (48 bit)

PGD: 0000 0000 0 = 0

PUD: 000 0000 00 = 0

PMD: 01 0000 000 = 128

PT: 0 0000 0000 = 0

Prima pagina 0: 0: 128: 0

Seconda 0: 0: 128: 1

Struttura della Tabella delle Pagine

VMA M1: 1 pagina a partire da 0000 2000 0000 (48 bit)
PGD: 0000 0000 0 = 0
PUD: 000 0000 00 = 0
PMD: 10 0000 000 = 256
PT: 0 0000 0000 = 0
Prima pagina 0: 0: 256: 0

VMA M2: 1 pagina a partire da 0000 3000 0000 (48 bit)
PGD: 0000 0000 0 = 0
PUD: 000 0000 00 = 0
PMD: 11 0000 000 = 384
PT: 0 0000 0000 = 0
Prima pagina 0: 0: 384: 0

VMA M3: 1 pagina a partire da 0000 4000 0000 (48 bit)
PGD: 0000 0000 0 = 0 PUD: 000 0000 01 = 1
PMD: 00 0000 000 = 0 PT: 0 0000 0000 = 0
Pagina :0: 1: 0: 0

Struttura della Tabella delle Pagine

VMA T1: 2 pagine a partire da 7FFF F77F BFFF (48 bit)

PGD: 0111 1111 1 = 255

PUD: 111 1111 11 = 511

PMD: 11 0111 011 = 443

PT: 1 1111 1011 = 507

Prima pagina: 255: 511: 443; 508

Seconda: 255: 511: 443: 507

VMA T0: 2 pagine a partire da 7FFF F77F EFFF (48 bit)

PGD: 0111 1111 1 = 255

PUD: 111 1111 11 = 511

PMD: 11 0111 011 = 443

PT: 1 1111 1110 = 510

Prima pagina: 255: 511: 443: 511

Seconda: 255: 511: 443: 510

Struttura della Tabella delle Pagine

VMA P: 3 pagine a partire da 7FFF FFFF CFFF (48 bit hex)

PGD: 0111 1111 1 = 255

PUD: 111 1111 11 = 511

PMD: 11 1111 111 = 511

PT: 1 1111 1100 = 508

Prima pagina 255: 511: 511: 510

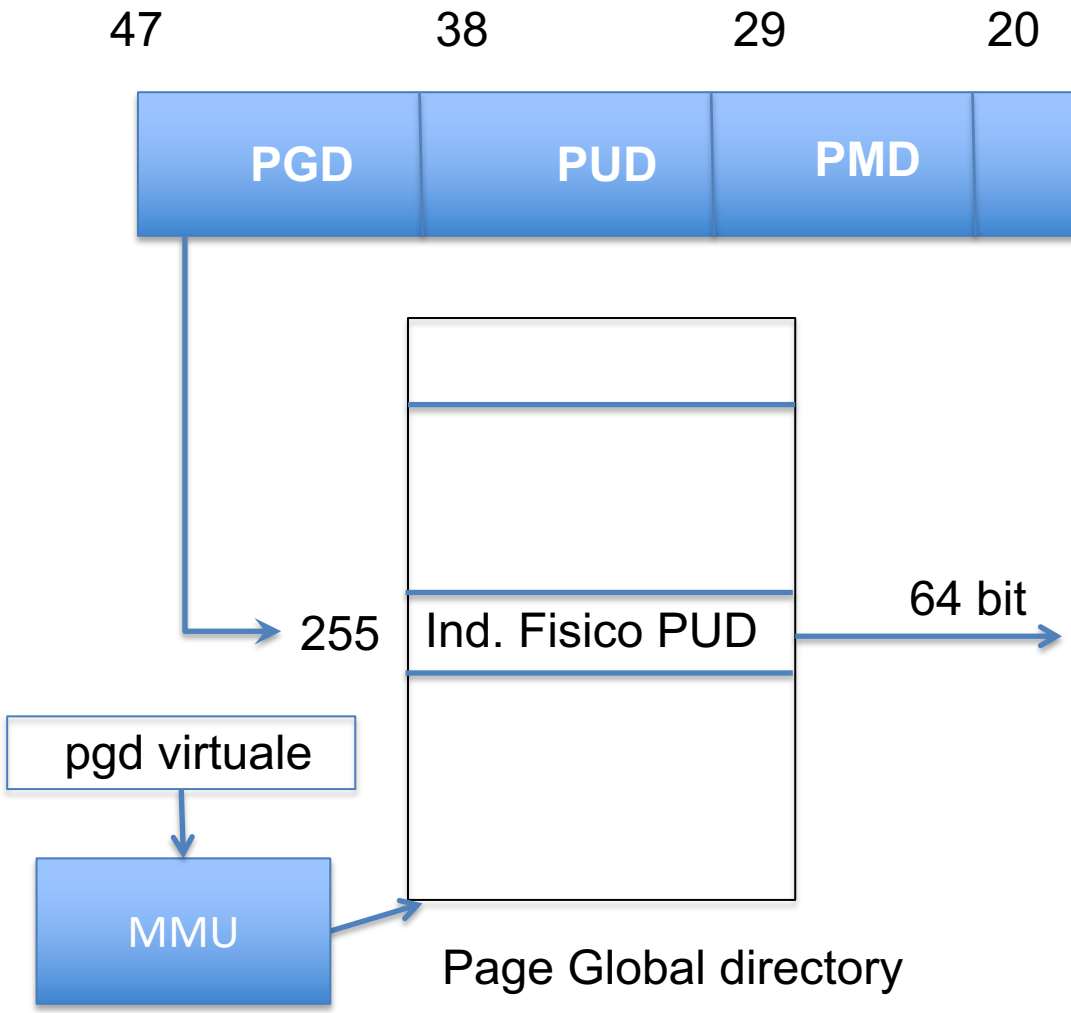
Seconda 255: 511: 511: 509

Terza: 255: 511: 511: 508

Page walk software – Lettura dei directory

- E' possibile simulare il page walk eseguito dalla MMU via software
- Il SO è in grado di utilizzare gli indirizzi fisici grazie alla mappatura virtuale/fisica del sistema operativo (la costante PAGE_OFFSET consente la rilocazione)
- La funzione get_PT esegue le stesse operazioni svolte dalla MMU quando accede alla Tabella delle Pagine interpretando le diverse componenti di un indirizzo virtuale
- La funzione ripete le stesse operazioni per attraversare PGD, PUD, PMD e PT
    - Esempio: indirizzo **0x00007fffb0d42118**

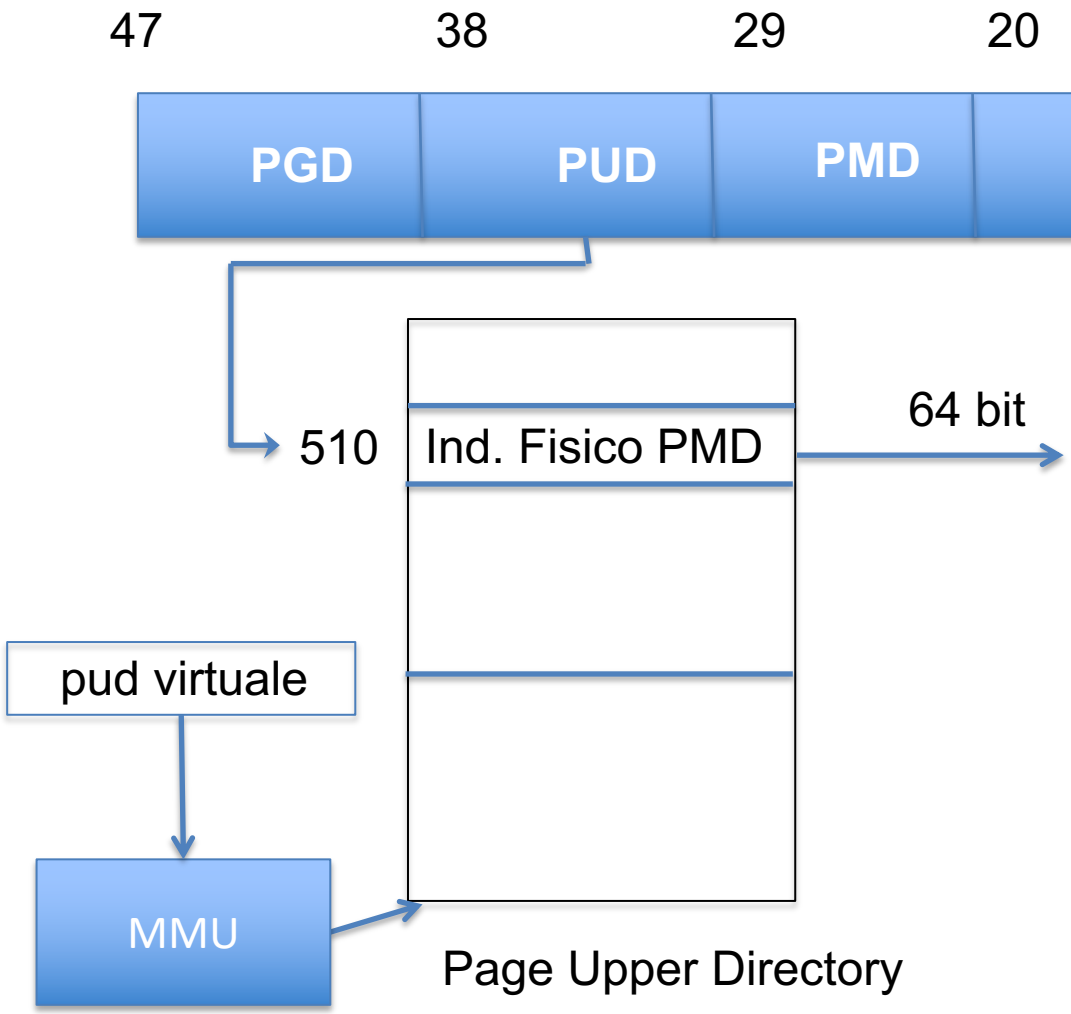
Calcolo dell'indirizzo virtuale della Page Upper Directory - 0x00007fffb0d42118



Rimappa indirizzo fisico in indirizzo virtuale:

- Azzera offset (12 bit meno significativi = 0)
- **Indirizzo virtuale pud = indirizzo fisico pud + PAGE_OFFSET/8**
- La divisione per 8 necessaria per allineare a 8 byte essendo la memoria indirizzata a byte

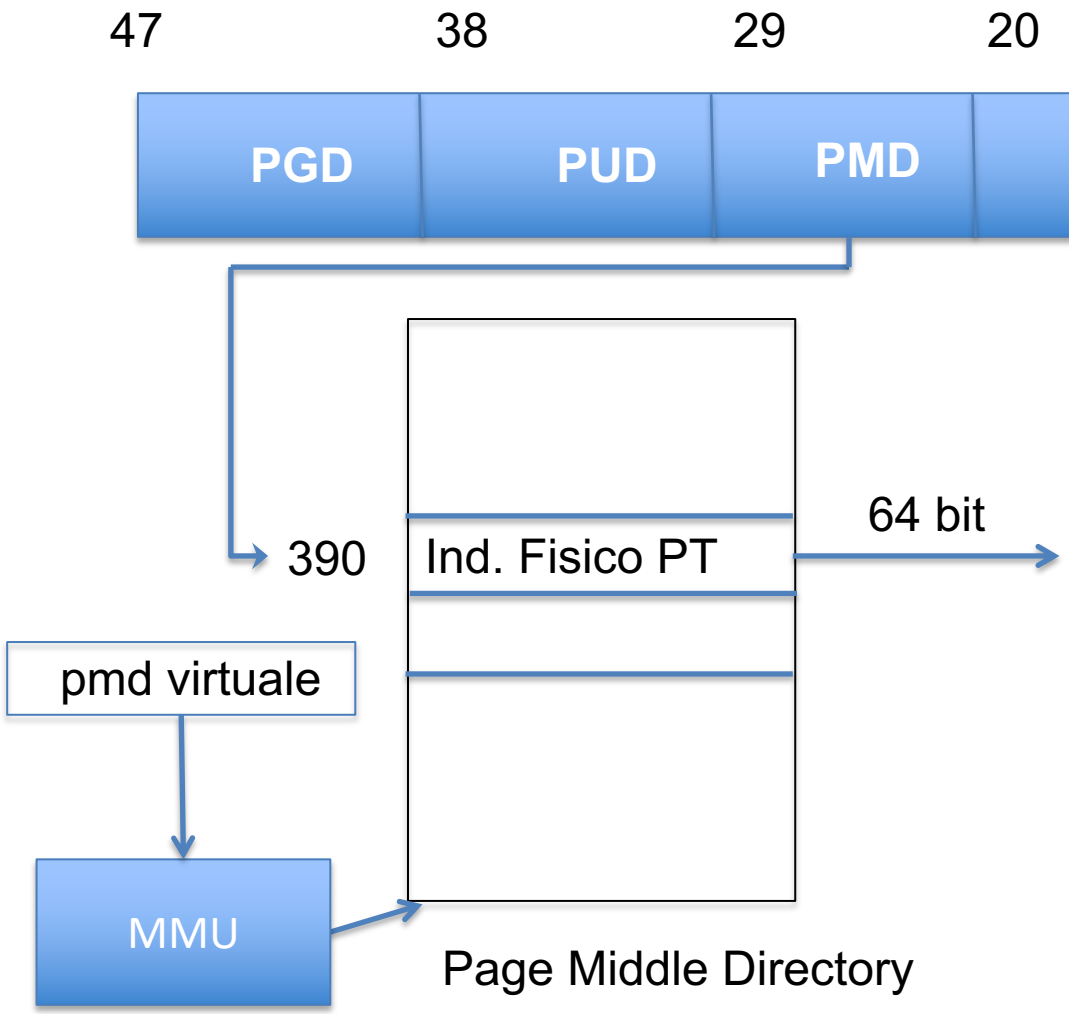
Calcolo dell'indirizzo virtuale della Page Middle Directory



Rimappa indirizzo fisico in indirizzo virtuale:

- Azzera offset (12 bit meno significativi = 0)
- **Indirizzo virtuale pmd = indirizzo fisico pmd + PAGE_OFFSET/8**
- La divisione per 8 necessaria per allineare a 8 byte essendo la memoria indirizzata a byte

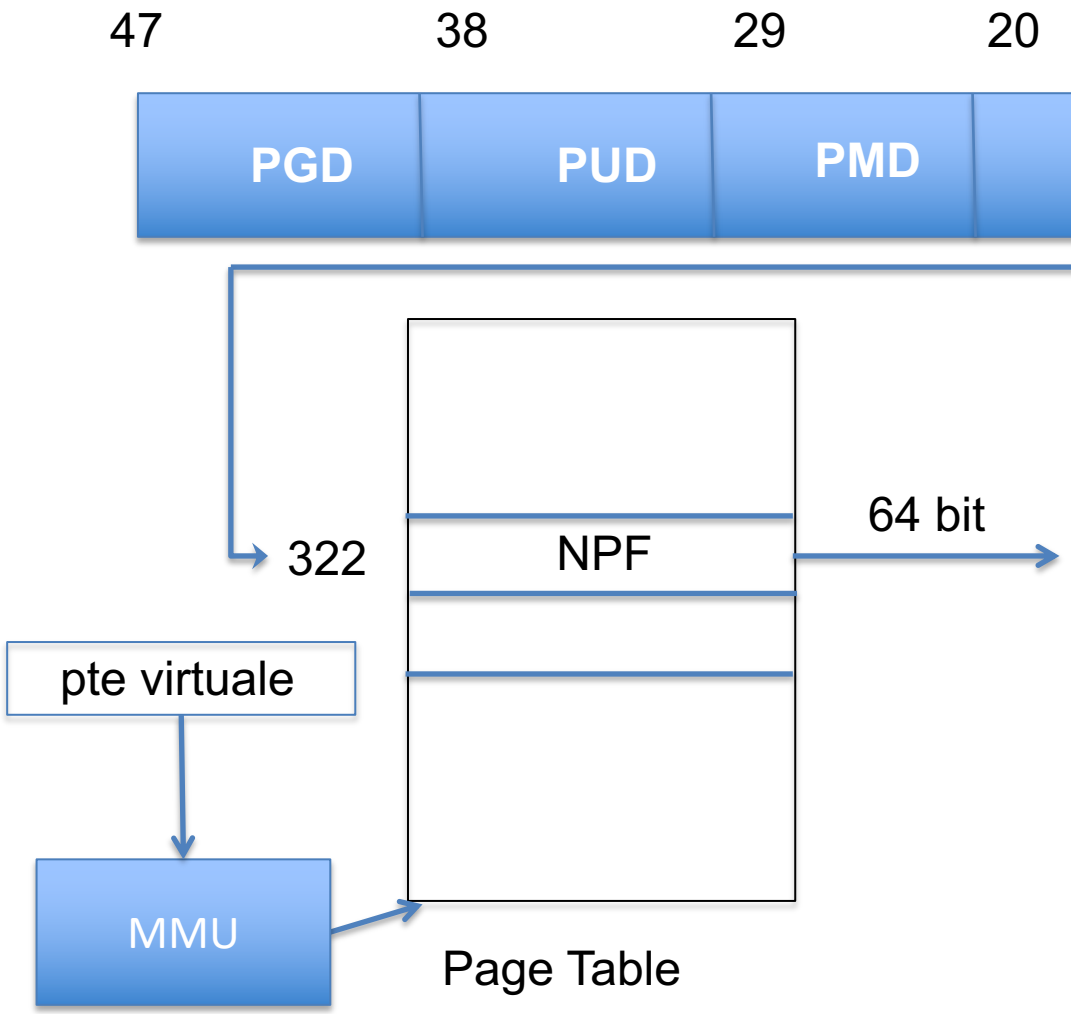
Calcolo dell'indirizzo virtuale della Page Table



Rimappa indirizzo fisico in indirizzo virtuale:

- Azzera offset (12 bit meno significativi = 0)
- **Indirizzo virtuale pte = indirizzo fisico pte + PAGE_OFFSET/8**
- La divisione per 8 necessaria per allineare a 8 byte essendo la memoria indirizzata a byte

Calcolo dell'indirizzo parola



Rimappa indirizzo fisico in indirizzo virtuale:

- Azzera offset (12 bit meno significativi = 0) e bit 63
- **Indirizzo virtuale NPV = NPF + PAGE_OFFSET/8**
  È l'indirizzo virtuale che si mappa sull'indirizzo fisico NPF

Indirizzo della parola = NPV + OFFSET
Utilizzo di questo indirizzo per leggere la parola

Esecuzione del Context Switch relativamente alla memoria

- La funzione schedule( ), prima di invocare la la macro assembler switch_to(prev, next) che esegue la commutazione delle sPile dei 2 processi, invoca la seguente funzione:

  switch_mm (oldmm, mm, next);

- Il punto centrale di questa funzione è costituito dall'invocazione della macro assembler **load_cr3** che assegna al registro CR3 il valore della variabile **pgd** presa dal descrittore del nuovo (next) task da eseguire.

Esercizio: decomposizione dell'indirizzo virtuale

Indirizzo virtuale: **0x00FABC00FFFF**

Indirizzo virtuale in binario:
0000 0000 1111 1010 1011 1100 0000 0000 1111 1111 1111 1111

PGD	0000 0000 1	1
PUD	111 1010 10	490 = 2+8+32+64+128+256
PMD	11 1100 000	480 = 32+64+128+256
PT	0 0000 1111	15 = 1+2 + 4 +8
OFFSET	1111 1111 1111	4095

Notazione:
1:490:480:15

Esercizio: tabella delle pagine

Dato un processo che ha la seguente struttura di memoria fisica:

- 10 KB codice
- 2 KB costanti
- 4 KB dati statici
- 4 KB pila del thread 1
- 6 KB pila

Si assuma che tutto il codice è allocato in memoria fisica

Definire la struttura delle VMA del processo

  - vengono allocate inizialmente 3 pagine di pila, l'ultima lasciata libera per la crescita
  - Vengono allocate 2 pagine di pila per i thread

VMA del Processo

VMA	Start Address	Dim	R/W	P/S	M/A	mapping
C	000000400000	3	R	P	M	<X,0>
K	000000600000	1	R	P	M	<X,3>
S	000000601000	1	W	P	M	<X,4>
T0	7FFFF77FEFFF	2	W	P	A	<-1,0>
P	7FFFFFFFCFFF	3	W	P	A	<-1,0>

Tabella delle pagine

Indicare il contenuto della tabella delle pagine, assumendo una allocazione in ordine delle aree virtuali, partendo dall'indirizzo esadecimale bb6

PGD	PUD	PMD	PT	NPF	
0	0	2	0	bb6	C0
0	0	2	1	bb7	C1
0	0	2	2	bb8	C2
0	0	3	0	bb9	K0
0	0	3	1	bba	S0
255	511	443	511	bbb	T00
255	511	443	510	bbc	T01
255	511	511	511	bbd	P0
255	511	511	510	bbe	P1
255	511	511	509	bbf	P2