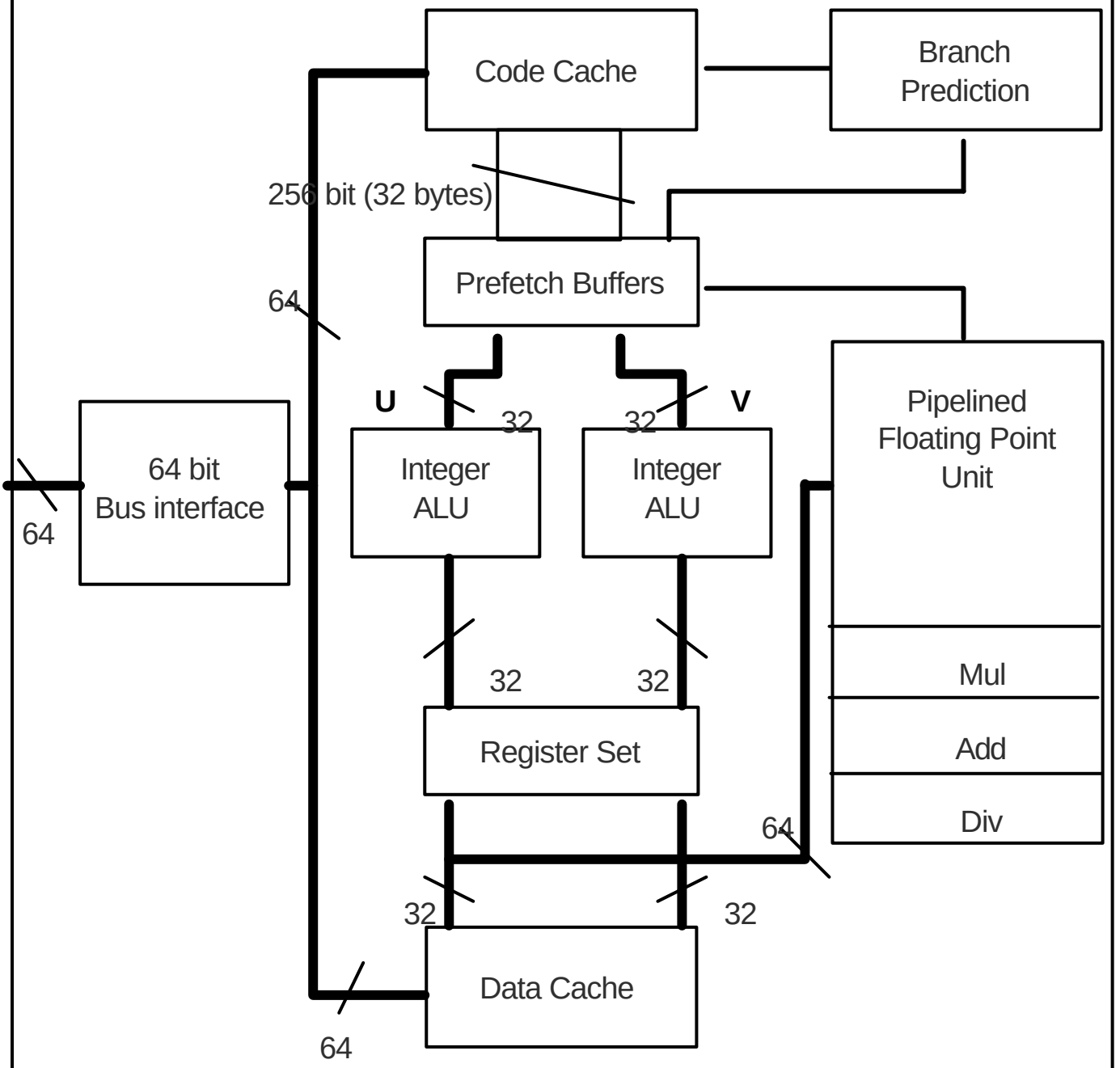
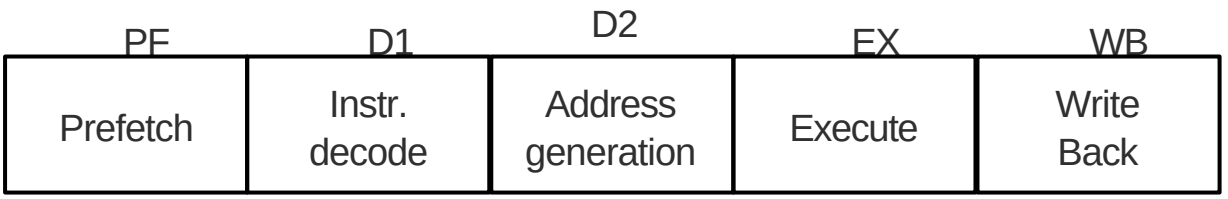
PENTIUM

Architettura interna del Pentium
Pipeline interna (U e V)
Branch Prediction Logic
Registri interni
Registri di sistema
Operating mode nel Pentium
Real Mode
Protected Mode
Livelli di privilegio
Segmentazione
Selettori e Descrittori di segmento
Call gates, Interruzioni
Task, Task State Segment
Paginazione
Indirizzi logici, lineari, fisici
Translation Lookaside Buffer

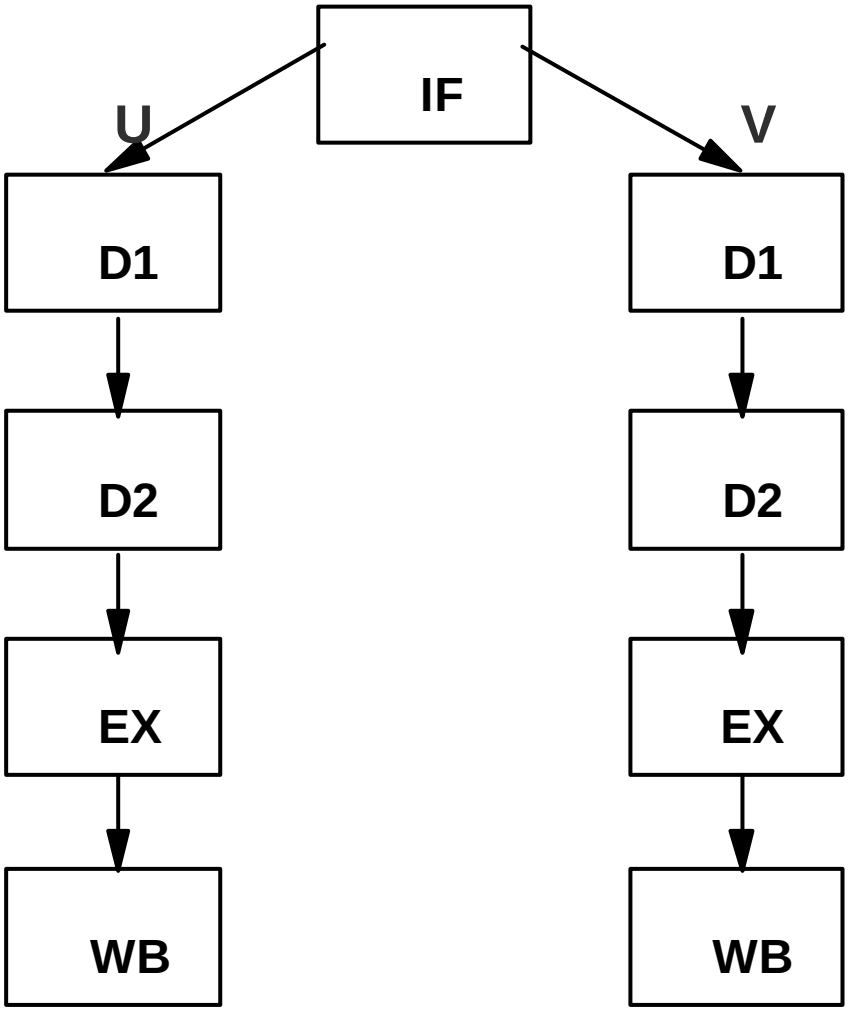
Architettura del Pentium



Pipeline



Instruction pairing (IF= instruction fetch/align)

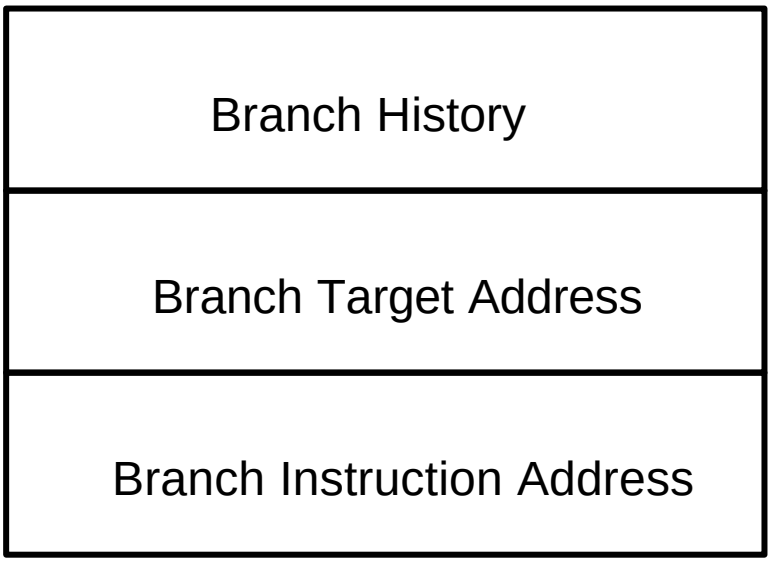


Pipeline

- La pipeline U esegue tutte le istruzioni mentre la V esegue solo "istruzioni semplici"
- Le istruzioni sulle due pipelines, in generale, non possono avere dati interdipendenti
- L'istruzione sulla V *segue* sempre quella sulla U
- Vi sono due prefecth buffers da 32 bytes, alternantisi, che operano in parallelo al Branch Target Buffer
- Le alee sono verificate nello stadio D1 (all'interno della stessa istruzione)
- Le due pipeline operano sincronamente (gli stadi D1 e D2 sono iniziati e terminati sincronamente; se la pipe v e' bloccata in EX u puo' avanzare. Nessuna istruzione puo' entrare in EX finche' u e v non entrano in WB))
- Alcune istruzioni possono essere eseguite solo da una delle due pipelines
- A causa della complessità di alcune istruzioni, la fase execute può durare >> 1 CK

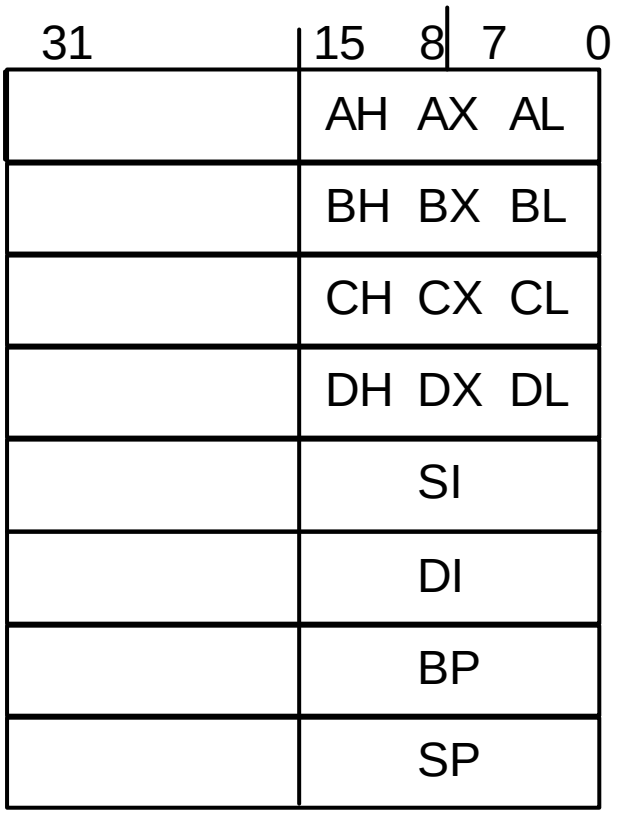
Branch Prediction

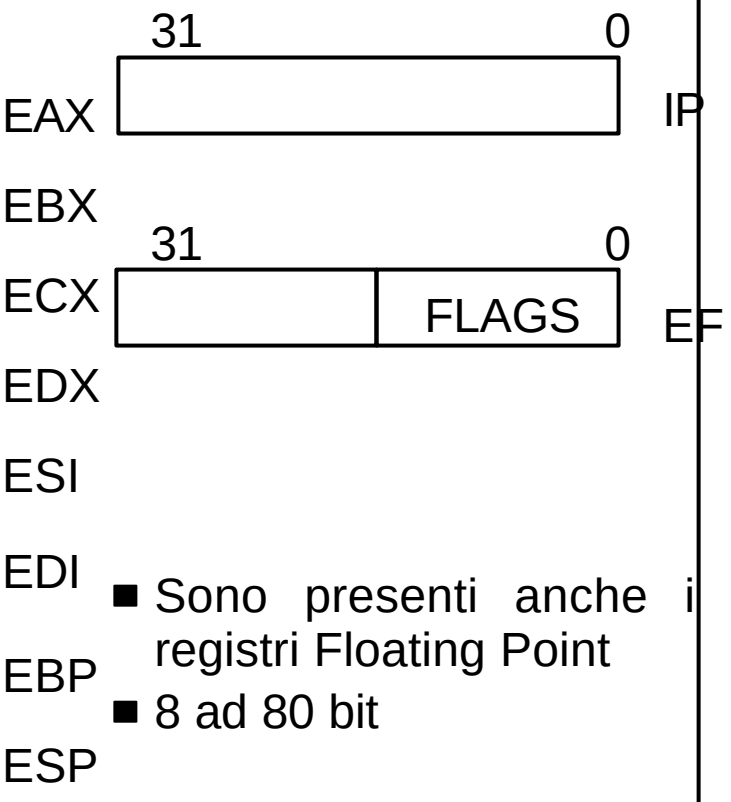
BRANCH TARGET BUFFER



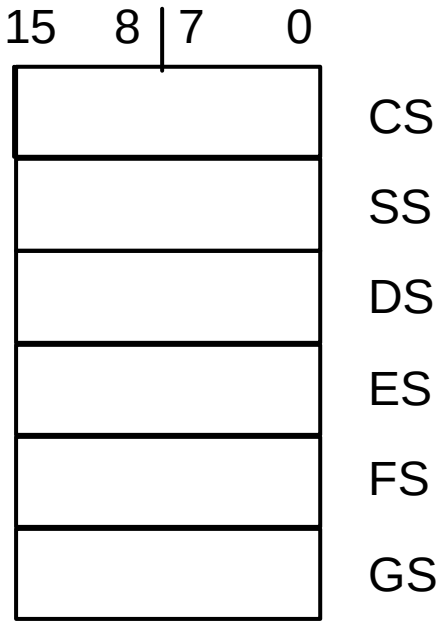
- Lo stadio di prefetch ha due buffer da 32 bytes riempiti alternativamente e sequenzialmente fino a un branch. Se la predizione e' TAKEN l'altro buffer viene riempito con il nuovo codice, e l'esecuzione prosegue di li'. In caso di errore le pipelines delle istruzioni sono azzerate
- Se il BTB predice correttamente non si hanno penalizzazioni; in caso contrario si hanno 6 o 7 clock aggiuntivi (a seconda delle due pipelines)

Registri

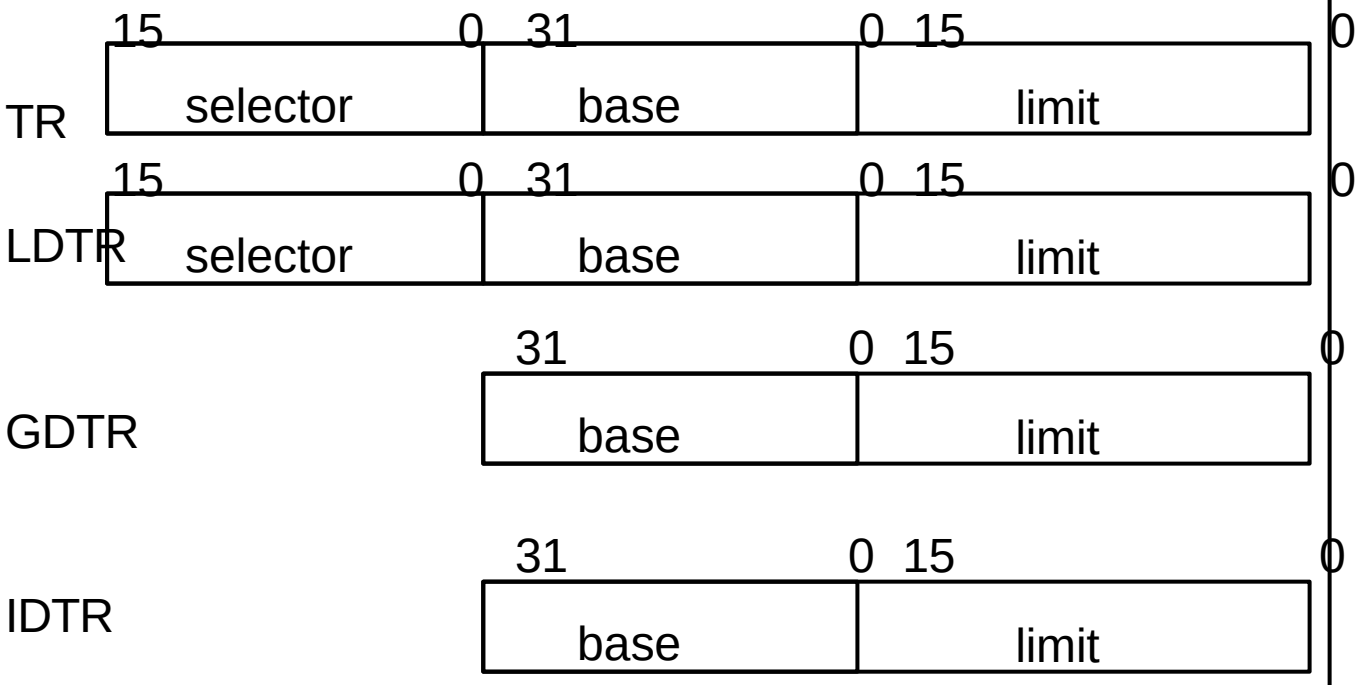


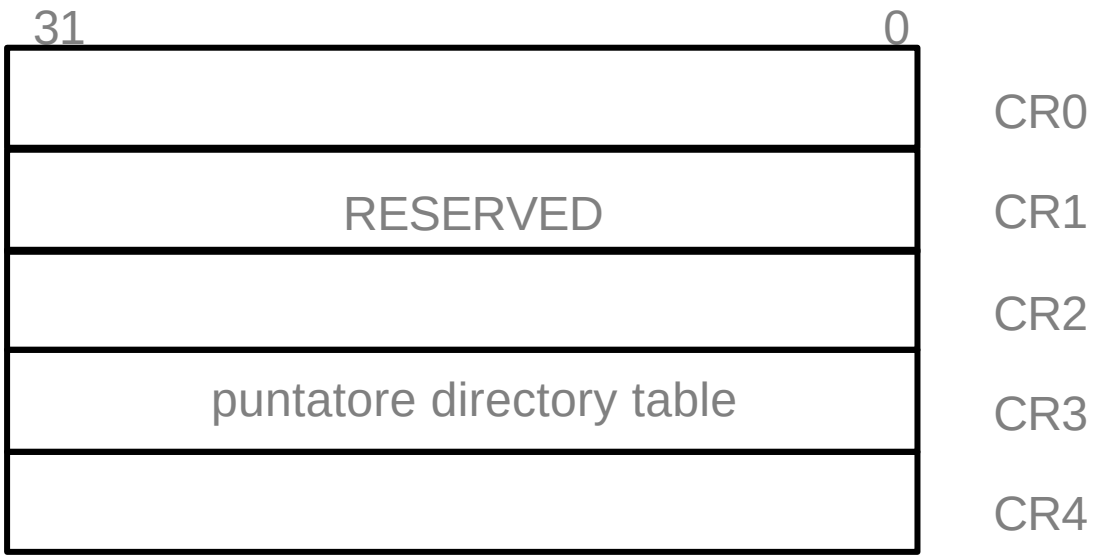


- Sono presenti anche i registri Floating Point
- 8 ad 80 bit



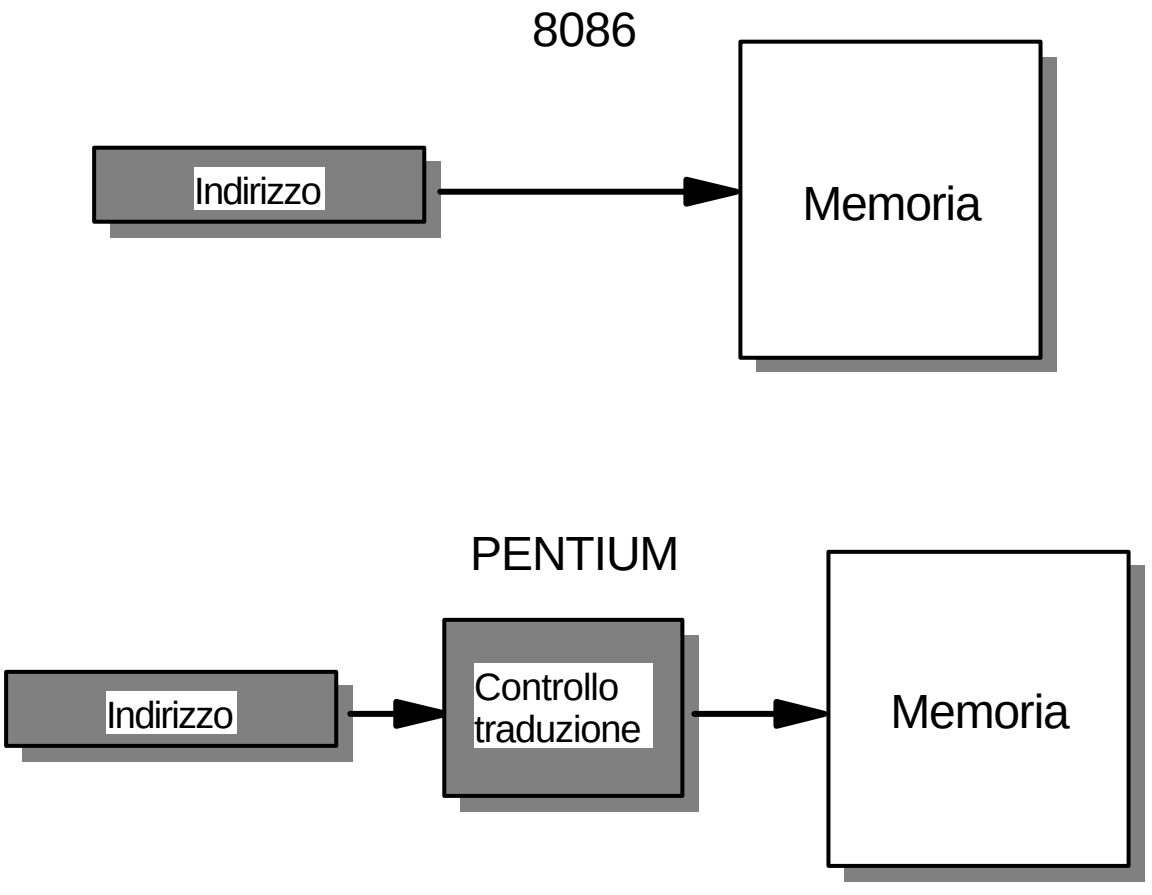
Registri


15
0
31
0
15
0
TR
selector
base
limit
15
0
31
0
15
0
LDTR
selector
base
limit
31
0
15
0
GDTR
base
limit
31
0
15
0
IDTR
base
limit


31
0
RESERVED
puntatore directory table
CR0
CR1
CR2
CR3
CR4

Gestione della memoria

- Protezione e controllo nell'accesso ai vari segmenti per verificare sia il diritto di accedere sia la correttezza dell'indirizzo
- Vari "environment" ciascuno dei quali deve risultare separato e protetto dagli altri (più processi in multitasking)
- Necessità di riferire molti segmenti per i diversi processi



Operating modes

REAL MODE	emula il funzionamento dell'8086 indirizzamento con segmento x 16 + offset anche se a 32 bit indirizzo lineare di 4gbyte gestisce le interruzioni come l'8086 a partire dall'indirizzo 0
PROTECTED MODE	gestisce la protezione in 4 livelli di privilegio, il multitasking con descrittori di task, la segmentazione con selettori e descrittori di segmento la paginazione con tabelle delle pagine le interruzioni con descrittore di interruzione
VIRTUAL 8086 MODE	in Protected Mode, i task operanti in Virtual 8086 Mode sono configurati per emulare l'8086 (es., una shell DOS in ambiente multitasking)

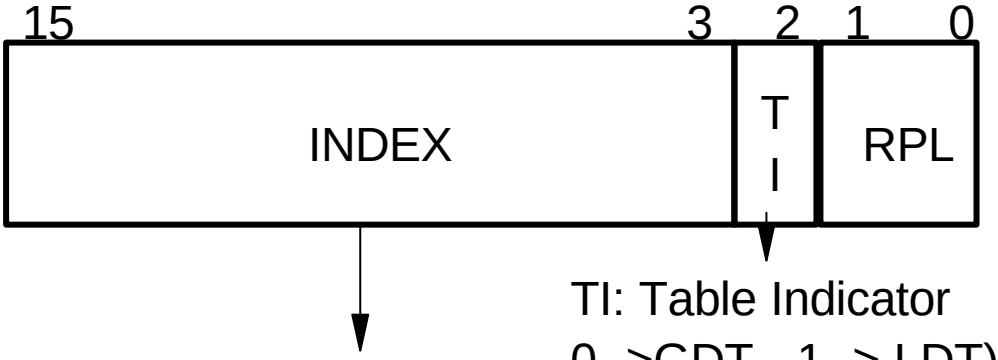
Real Mode

- E' il modo al RESET e permette di configurare il sistema accedendo a tutte le locazioni senza protezione
- La modalità di indirizzamento è identica a quella dell'8086 e permette di accedere alla memoria del modo reale. Il primo accesso e' all'indirizzo FFFFFFF0h ma appena il primo JUMP o CALL intersegment e' effettuato i bit di indirizzo A20-A31 sono posti a zero per CS.
- I segmenti sono di lunghezza 64K
- Ogni segmento contiene l'indirizzo base del segmento reale
- Si passa in modo protetto ponendo a 1 il bit PE nel registro CR0

Protected mode

L'accesso alla memoria avviene tramite i SEGMENTI che in modo protetto sono SELETTORI di DESCRITTORI di SEGMENTO

SELETTORE DI SEGMENTO



TI: Table Indicator
0 ->GDT - 1 -> LDT)

indice che punta ad un descrittore di 8 byte

13 bit -> 8192 selettori

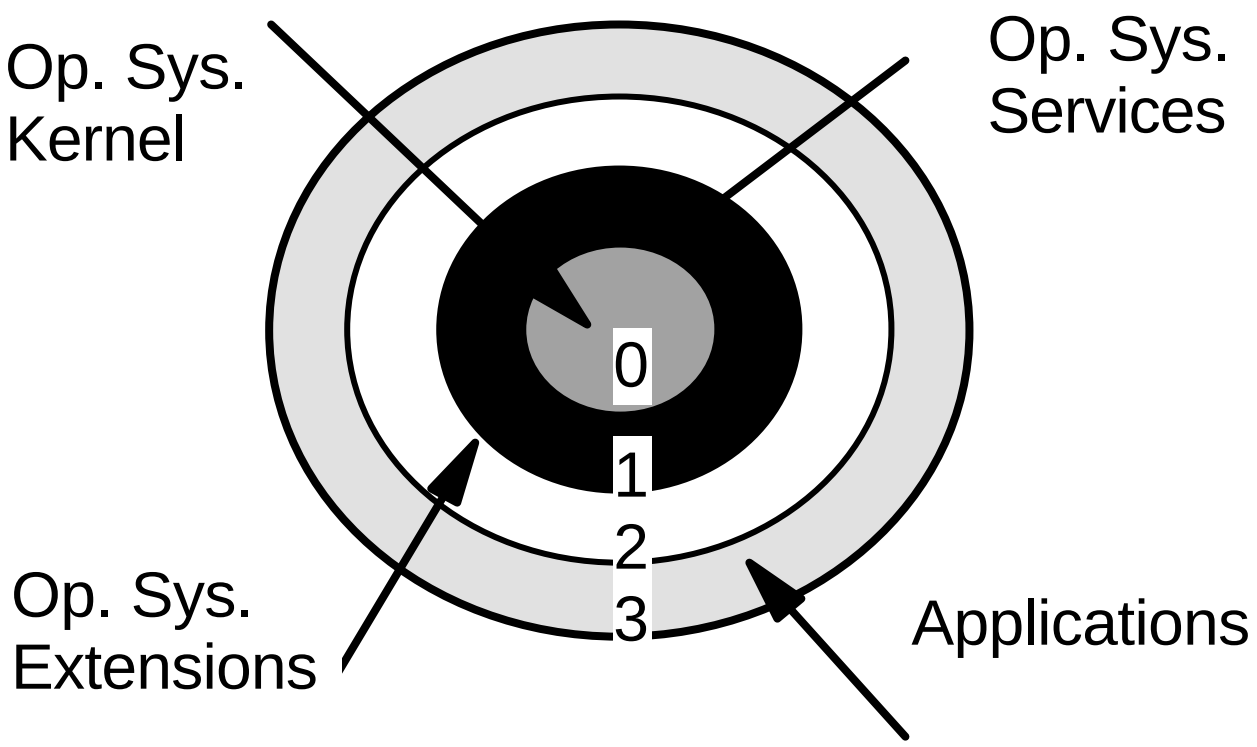
Due possibili tabelle di descrittori

GLOBAL DESCRIPTION TABLE (descrive i segmenti comuni a tutti i task)

LOCAL DESCRIPTION TABLE (descrive i segmenti privati di un task)

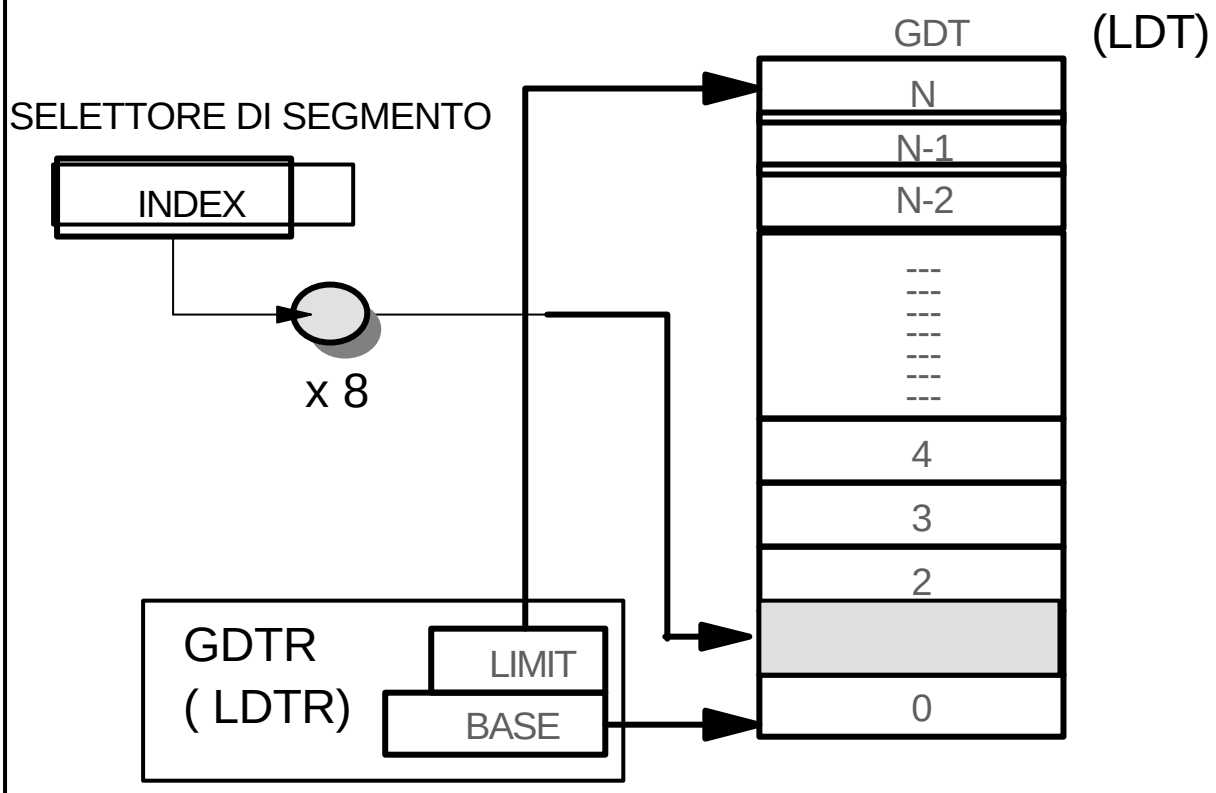
RPL: Request Privilege Level (00 -> Most privileged - 11 Least privileged):

Livelli di privilegio



CPL: Current PL, livello di privilegio corrente,
coincidente con il livello di privilegio del segmento di codice in corso di esecuzione.
Viene definito anche:
EPL (effective PL) = max (CPL, RPL)

Tabelle dei descrittori

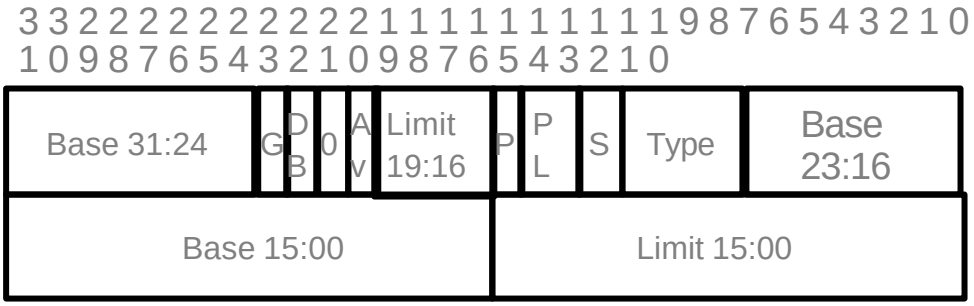


Stesso funzionamento per GDT, LDT e IDT
LDT fa parte di GDT
I descrittori sono caricati in una cache quando indirizzati
segment descriptor cache register

15 0 63 0

CS	access	base	limit
DS	access	base	limit
SS	access	base	limit
ES	access	base	limit
FS	access	base	limit
GS	access	base	limit

Descrittore di Segmento



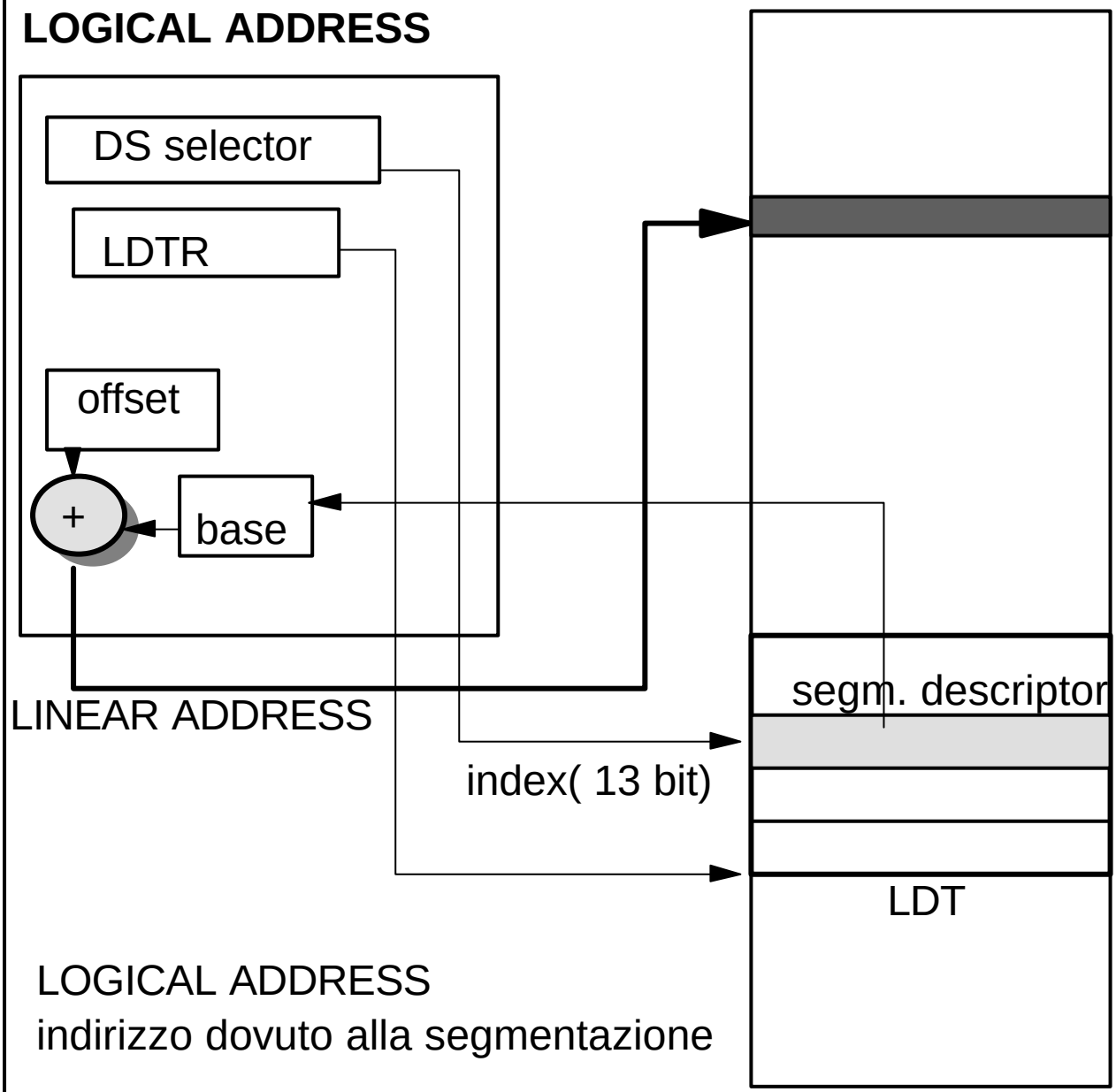
64 bit

- AV: disponibile per il software
- BASE: indirizzo base del segmento 32 bit
- DB: parallelismo del segmento (16/32 bit)
- PL: livello di privilegio (00....11)
- G: granularità (limite come multiplo di 1 o 4096 bytes)
- LIMIT: limite del segmento (lunghezza) 20 bit
- P: segmento presente in memoria (paginazione)
- S: tipo di descrittore (sistema/applicazione)
- TYPE: tipo di segmento(dato/codice- R/W - expand down/up)
- Riservato

I campi possono essere differentemente interpretati a seconda del TIPO e di altri parametri

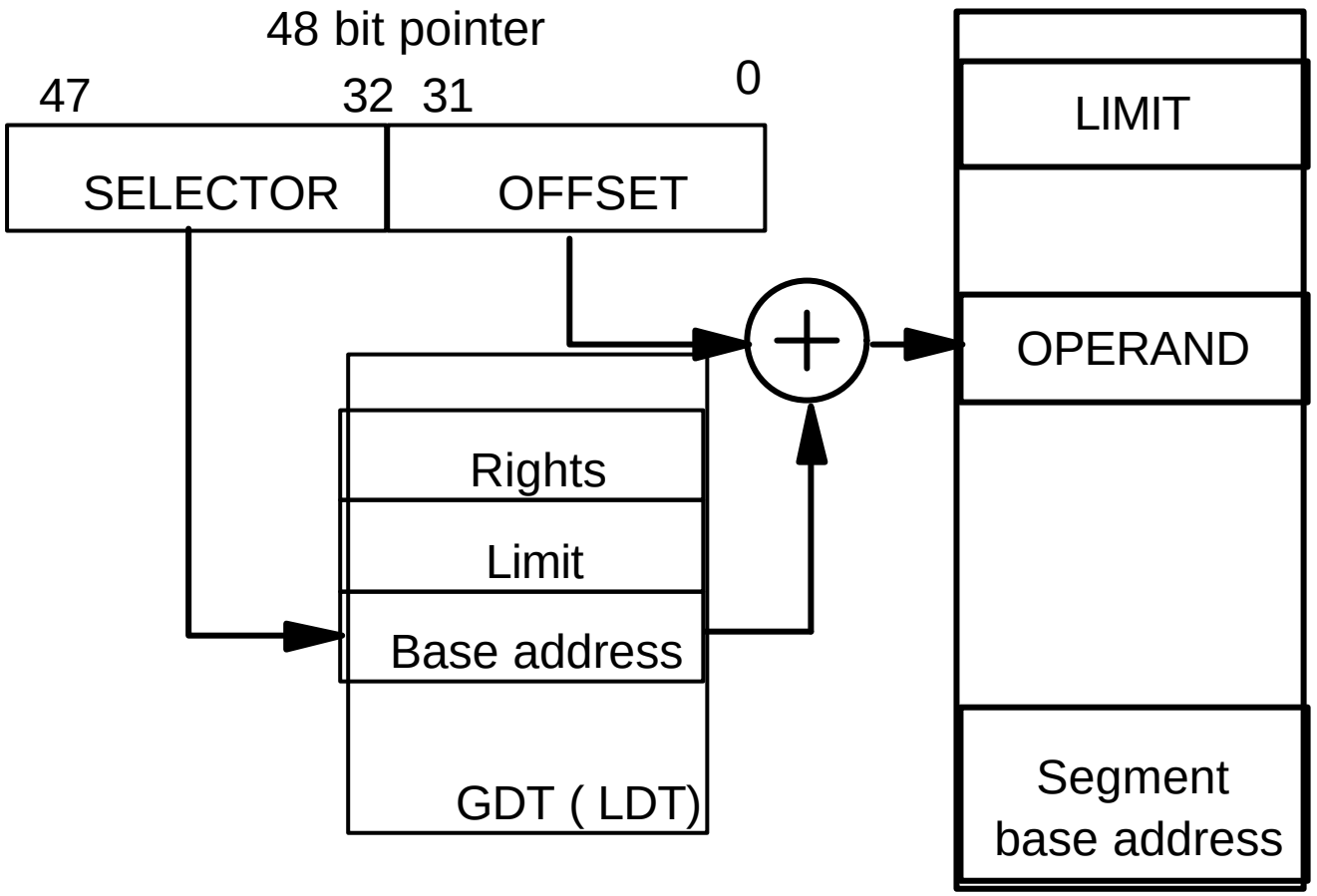
La dimensione di un segmento è multipla di byte (max 1 Mbytes) o di 4 Kbytes (max 4 Gbytes)

Indirizzamento con Segmentazione



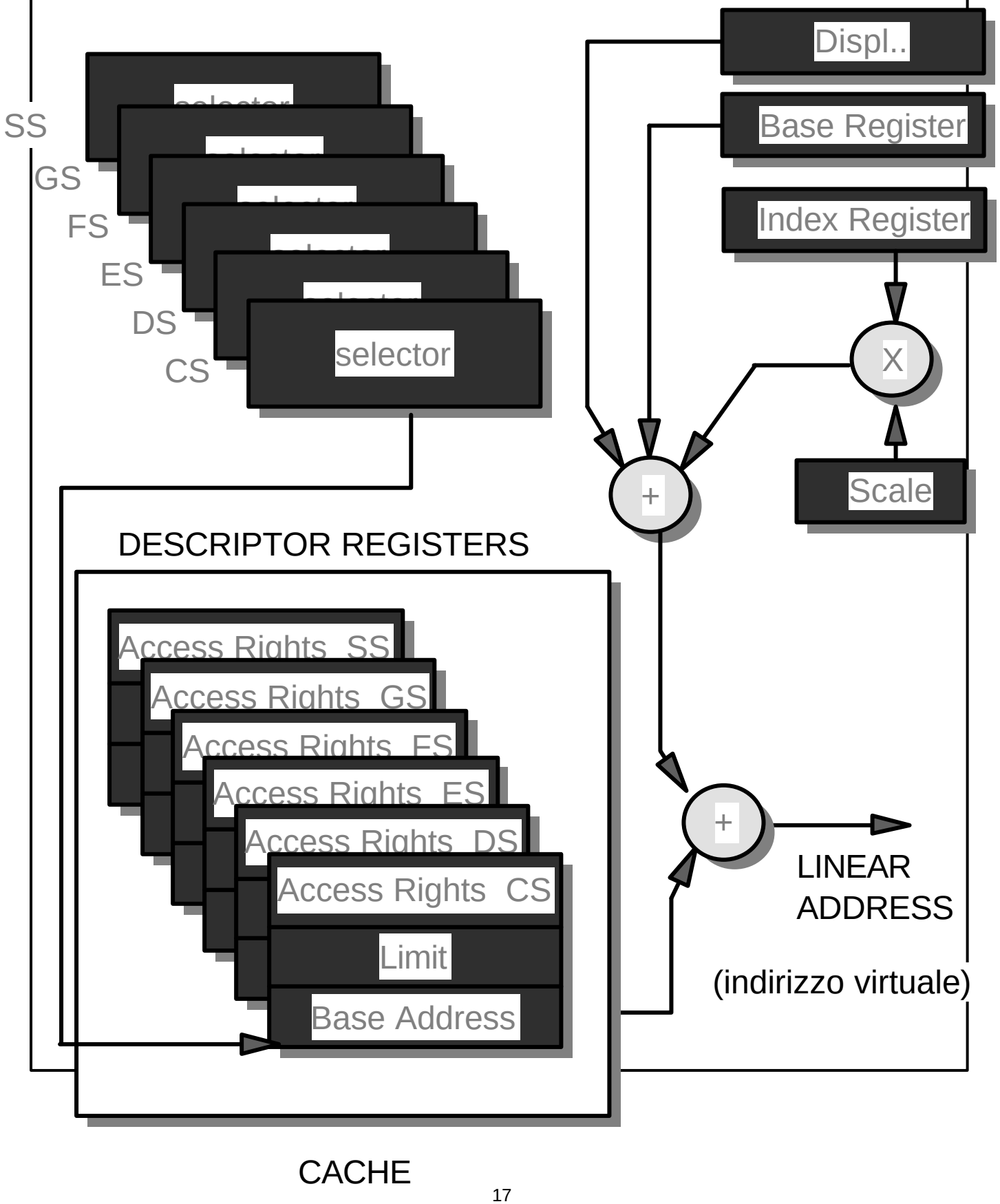
LOGICAL ADDRESS
indirizzo dovuto alla segmentazione

Indirizzamento



- L'indirizzo ottenuto è un INDIRIZZO VIRTUALE che, se il PAGING è attivo, deve essere tradotto in indirizzo fisico
- I descrittori sono in una cache interna alla CPU e sono caricati ogni qualvolta ad essi si fa riferimento nelle due tabelle che contengono tutti i descrittori

Indirizzamento



System segment- CALL GATES

ci sono tre tipi di descrittori di segmenti di sistema

- ➡ TASK STATE SEGMENT descrive un task
- ➡ LOCAL DESCRIPTION TABLE punta ad una ldt
- ➡ GATE

GATE e' usato per passare ad un livello di privilegio piu' alto

Accesso a dati:
CPL pl del CS corrente
RPL pl del segmento dell'operando
EPL= max(RPL, CPL)
DPL pl del descrittore puntato dal segmento dell'operando
si accede al dato solo se EPL <= DPL in senso numerico, cioè se chi accede ha privilegio maggiore uguale del segmento dati

call far, interrupt e trap gates permettono di eseguire una routine a privilegio diverso

Accesso a segmenti di codice

Accesso a codice in un altro segmento:

E' possibile accedere a un altro segmento di codice con le istruzioni JMP, CALL, RET

Il trasferimento è consentito in modo diretto solo se:

1) DPL = CPL, oppure

2) il segmento è conforming ("fidato"), e CPL >= DPL (in senso numerico). L'attributo conforming è un bit del descrittore. CPL rimane invariato.

Il trasferimento è consentito tramite una gate se:

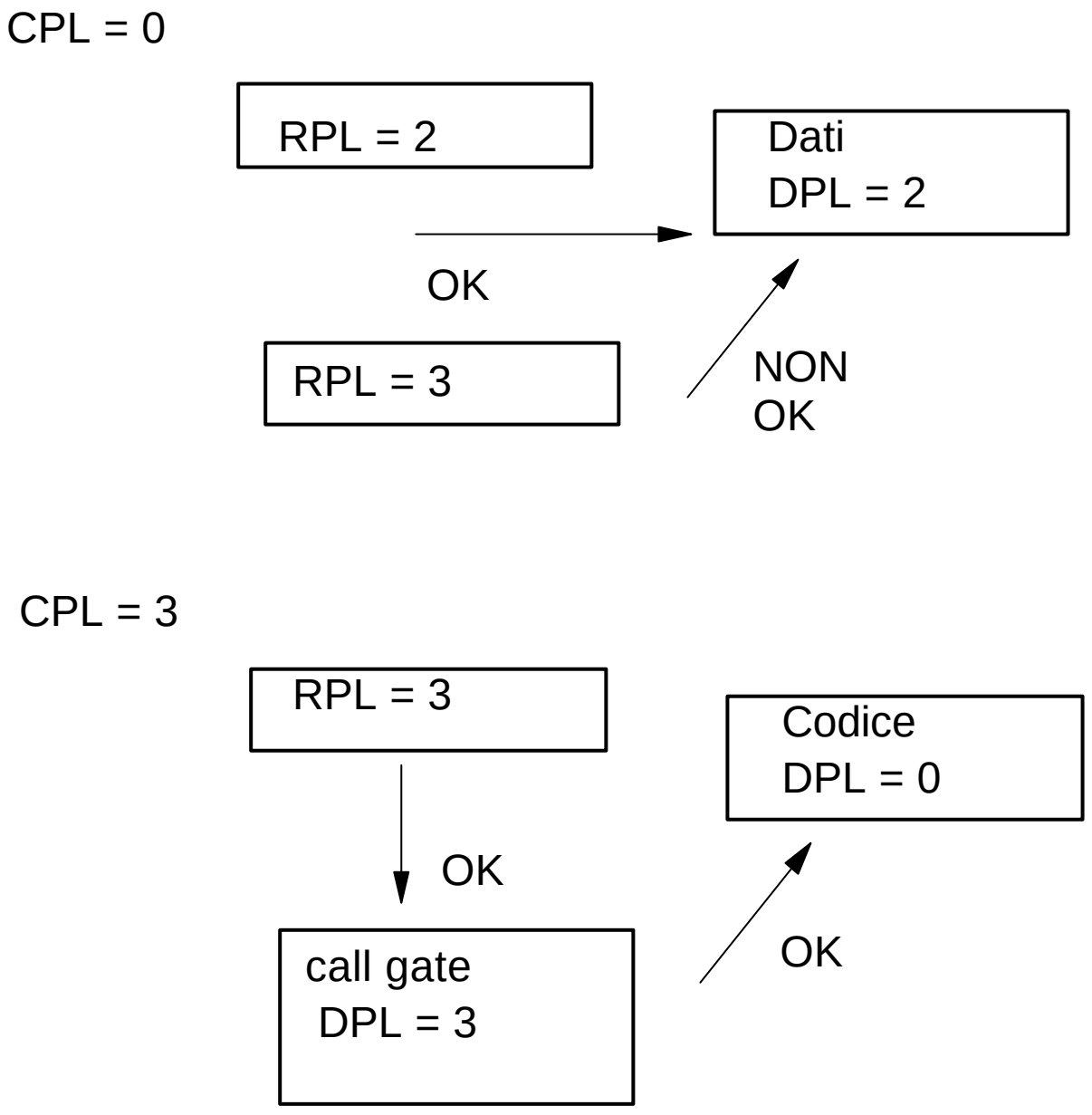
1) EPL <= DPL gate (il privilegio del richiedente è sufficiente per la gate), e

2) CPL >= DPL (il privilegio del segmento richiesto è maggiore uguale di quello corrente)

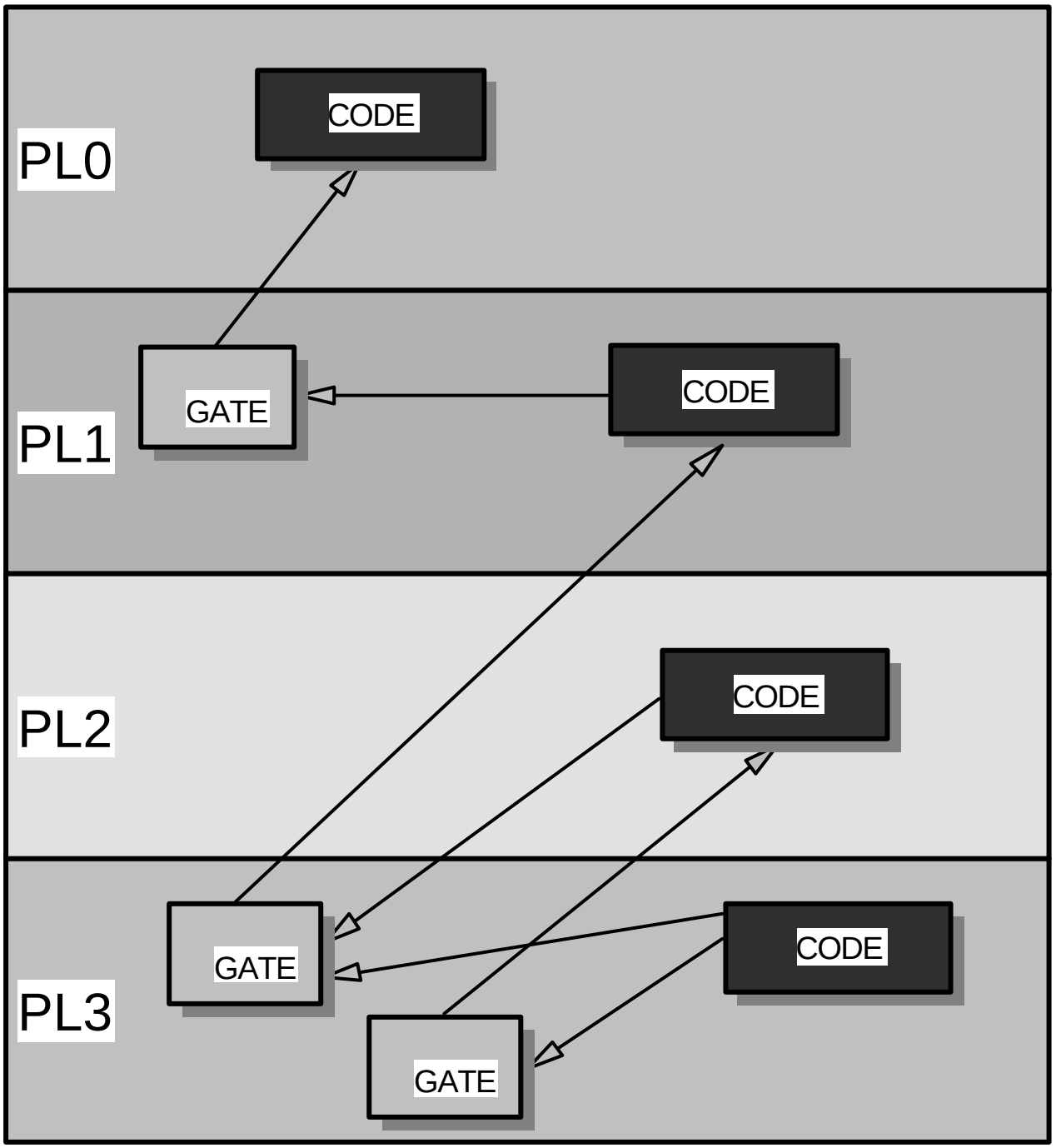
CPL diventa uguale a DPL.

Il motivo per cui non è consentito il trasferimento a segmenti di livello di privilegio minore è che da questi si rientrerebbe con una RET che alzerebbe il livello di privilegio: potrebbe essere usata indiscriminatamente

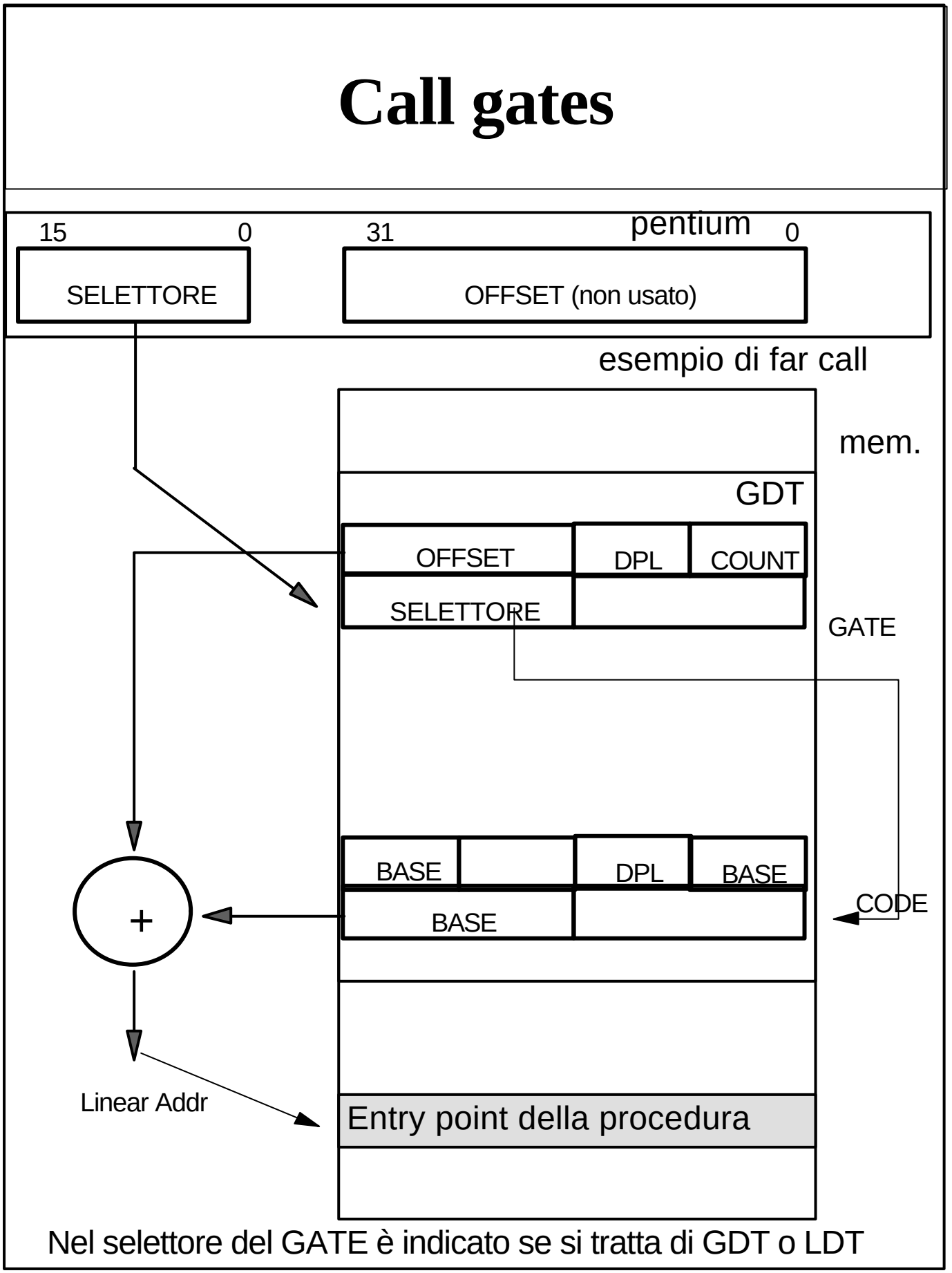
Esempi


CPL = 0
RPL = 2
Dati
DPL = 2
OK
RPL = 3
NON
OK
CPL = 3
RPL = 3
Codice
DPL = 0
OK
call gate
DPL = 3
OK

Call gates

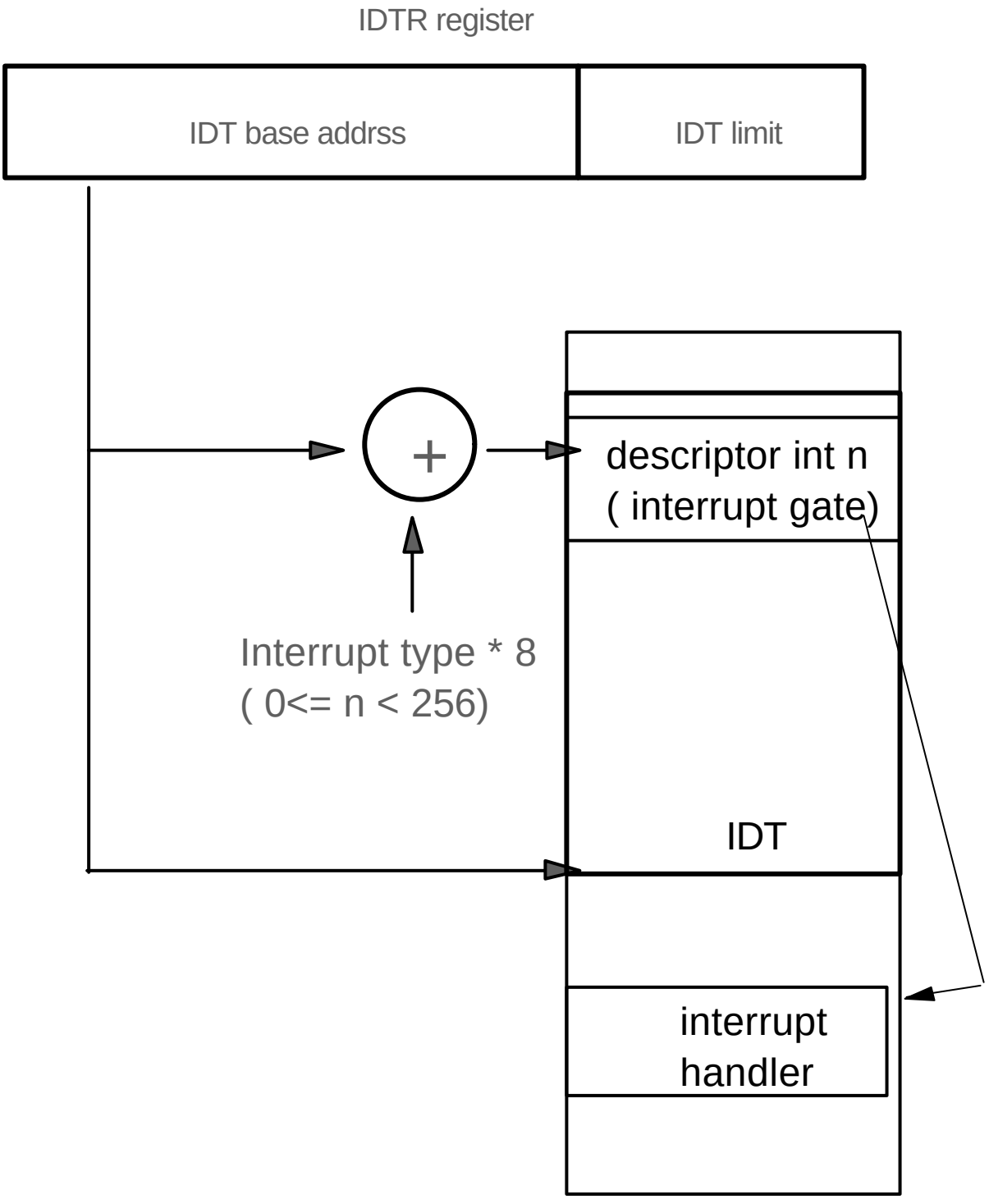


Call gates

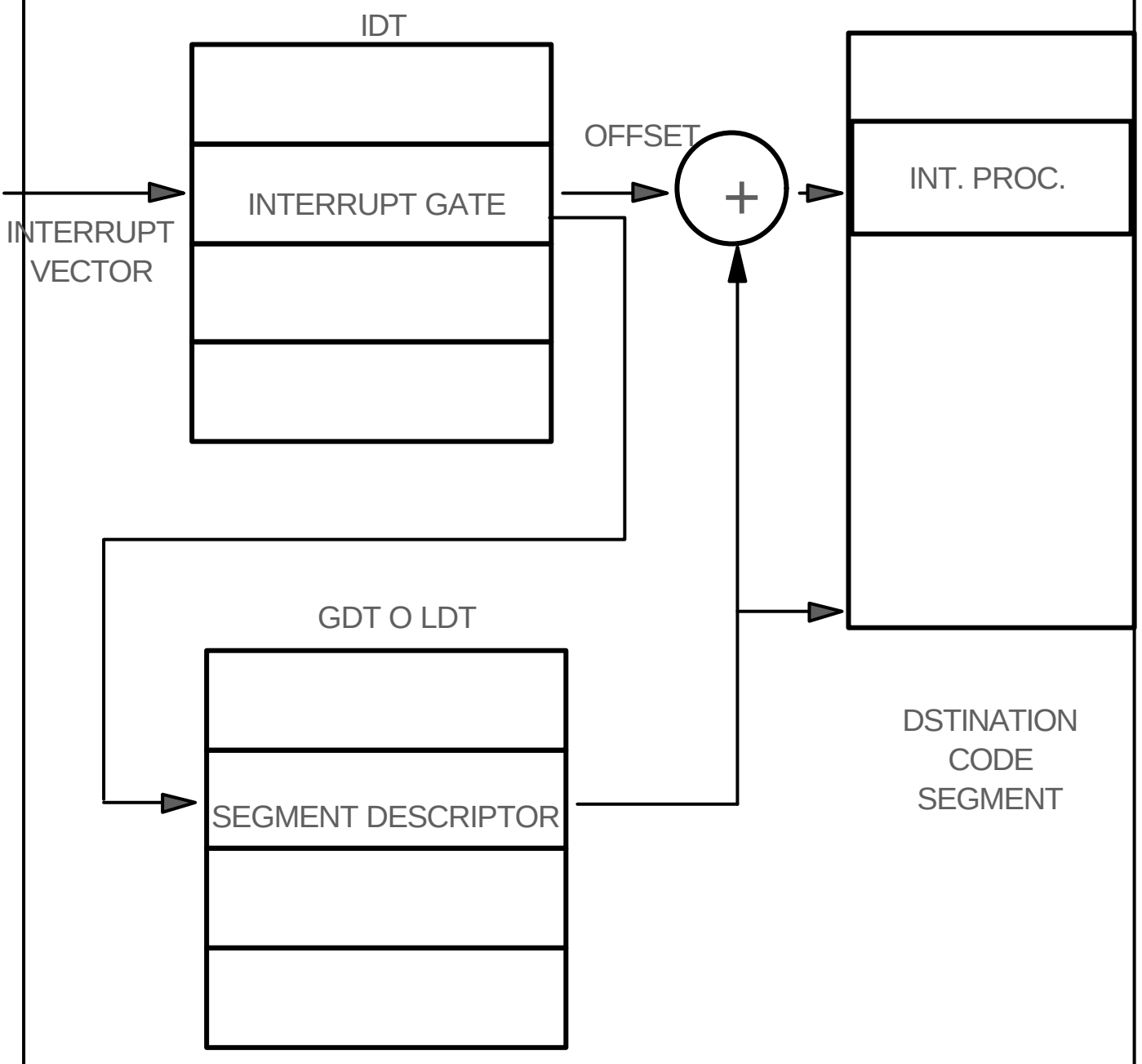


Nel selettore del GATE è indicato se si tratta di GDT o LDT

Interruzioni



Interruzioni



- Interruzioni
- Faults, Traps

Task

Task

e' il processo in esecuzione ed e' definito da un TASK STATE SEGMENT che ne definisce il CONTESTO e dotato di un insieme di segmenti indirizzabili dalla propria LDT (oltre che dalla GDT)

Attivare un task significa inizializzare i registri del processore con i registri contenuti nel TSS

Commutare un task (con una attivazione da parte del task stesso o da parte di una interruzione) significa caricare il TSS corrente salvando il contesto e attivare un nuovo task

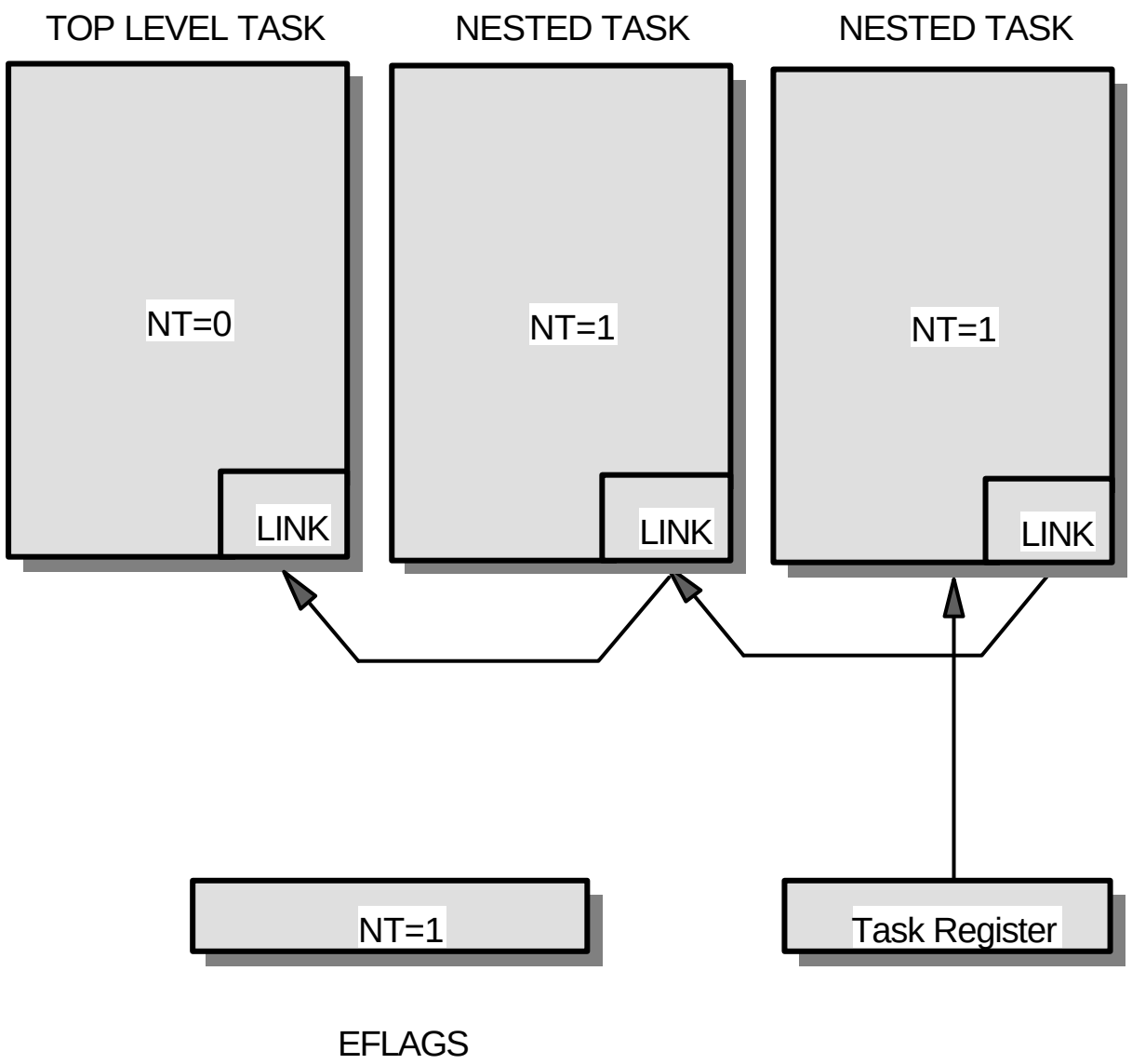
Task state segment

31 — 15	0	
Indirizzo di base Mappa I/O	T	64
Selettore LDT del processo		60
GS		5C
FS		58
DS		54
SS		50
CS		4C
ES		48
EDI		44
ESI		40
EBP		3C
ESP		38
EBX		34
EDX		30
ECX		2C
EAX		28
EFLAGS		24
EIP		20
CR3 (PDBR)		1C
SS2		18
ESP2		14
SS1		10
ESP1		C
SS0		8
ESP0		4
LINK (Selettore TSS precedente)		0

TR nel Pentium → task gate descriptor

104 byte

Nested Tasks

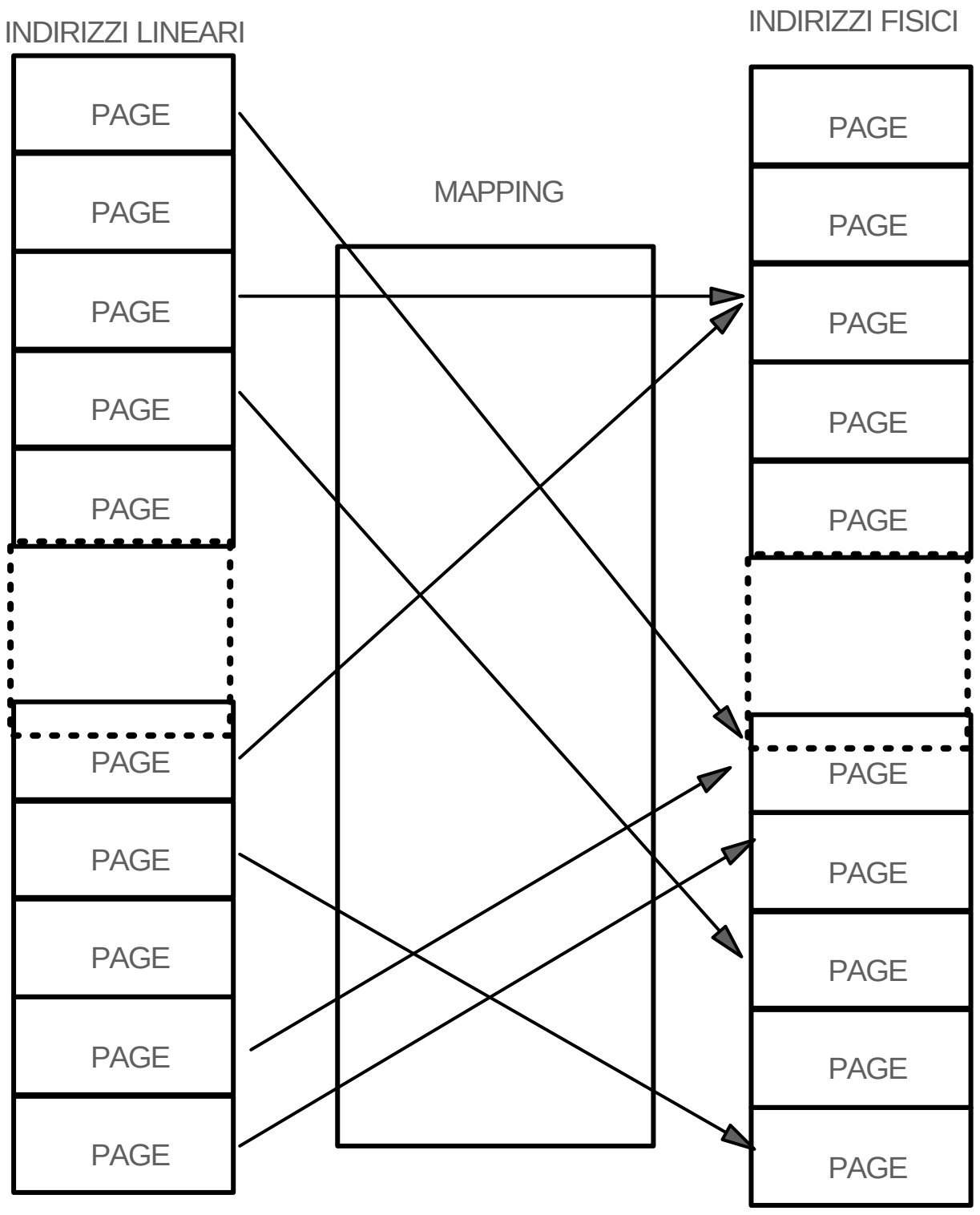


Il Task Register contiene il selettore del TSS descriptor

I/O

- La possibilità di effettuare I/O dipende dal bit di IOPL nei flags
- I bit della mappa di I/O nel TSS controllano l'accesso alle porte di I/O
- L'accesso alle porte di I/O è possibile se:
  - CPL $\leq$ IOPL oppure
  - I/O PERMISSION MAP OK
- La mappa di I/O contiene un bit per ogni porta accessibile
  - EX: il bit di controllo per la porta 41(decimale) si trova nel 41-esimo bit della mappa ovvero nel PRIMO bit del sesto byte.

Paging


INDIRIZZI LINEARI
INDIRIZZI FISICI
MAPPING
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE
PAGE

Spazi di indirizzamento

LOGICAL ADDRESS SPACE

14 bit (selector) + 32 bit offset = 46bit

64 Tbyte

LINEAR ADDRESS

dopo la segmentazione in protected

indirizzo di 32 bit (BASE nel descrittore + OFFSET)

per indirizzare 4GBYTE

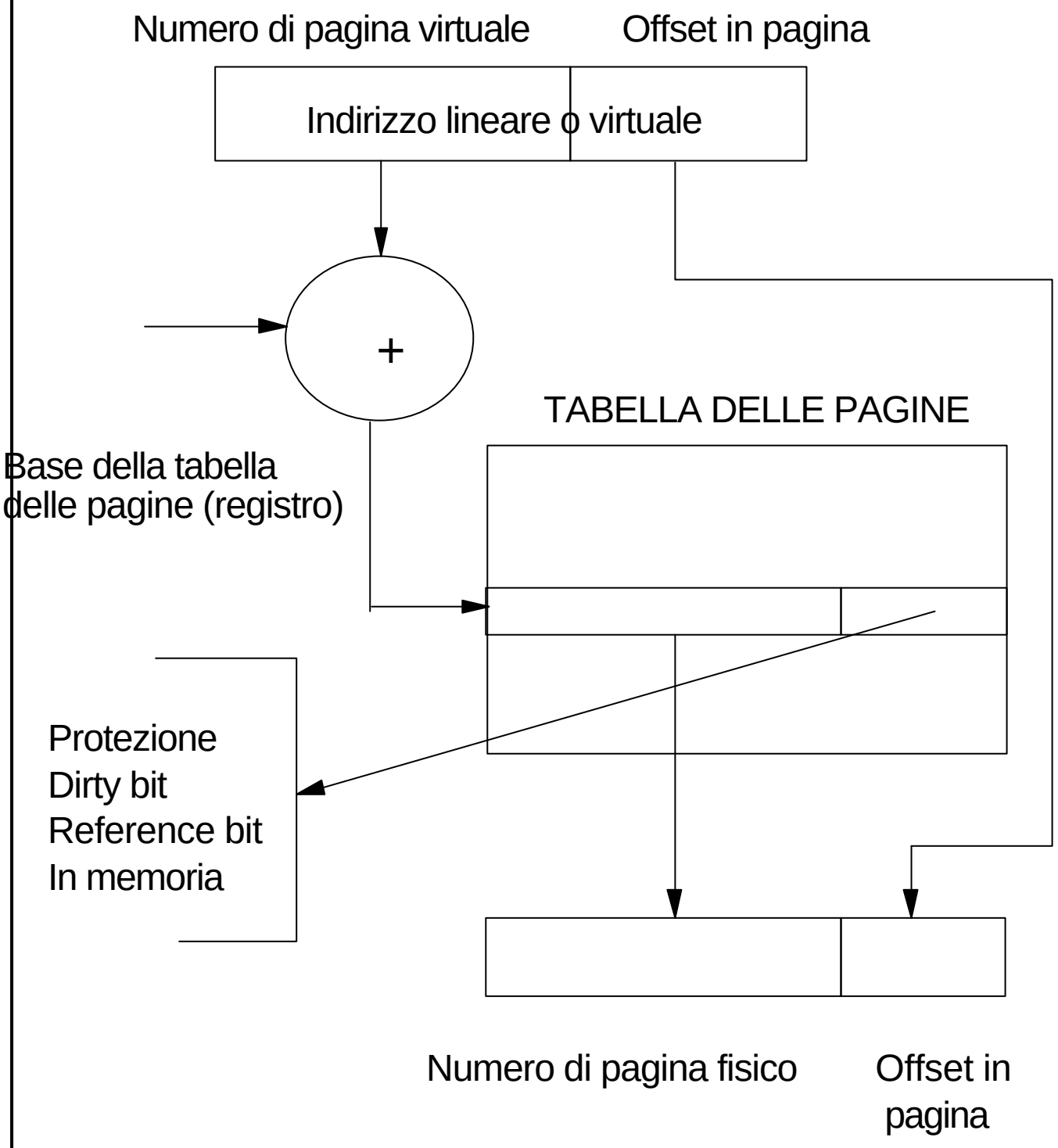
PHYSICAL ADDRESS 32 bit

spazio di indirizzamento **reale**: 4, 16,..., 256 Mbyte...

PAGINAZIONE permette di allocare pagine di 4K o 4 Mbyte indirizzate da un indirizzo VIRTUALE nell'indirizzo fisico disponibile

PAGINAZIONE e' indipendente dalla SEGMENTAZIONE

Paging


Numero di pagina virtuale
Offset in pagina
Indirizzo lineare o virtuale
+
TABELLA DELLE PAGINE
Base della tabella
delle pagine (registro)
Protezione
Dirty bit
Reference bit
In memoria
Numero di pagina fisico
Offset in
pagina

Paging nel Pentium

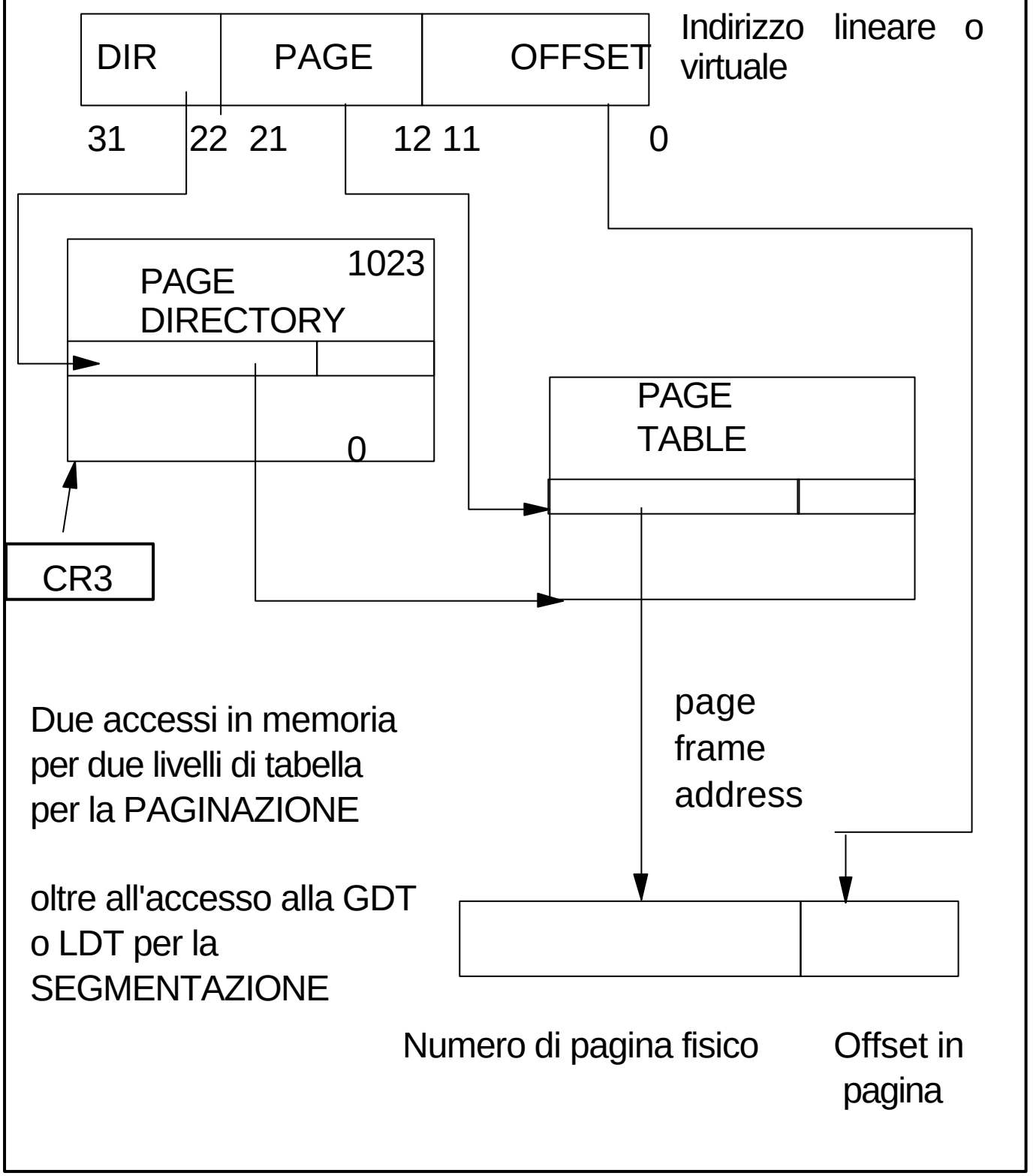
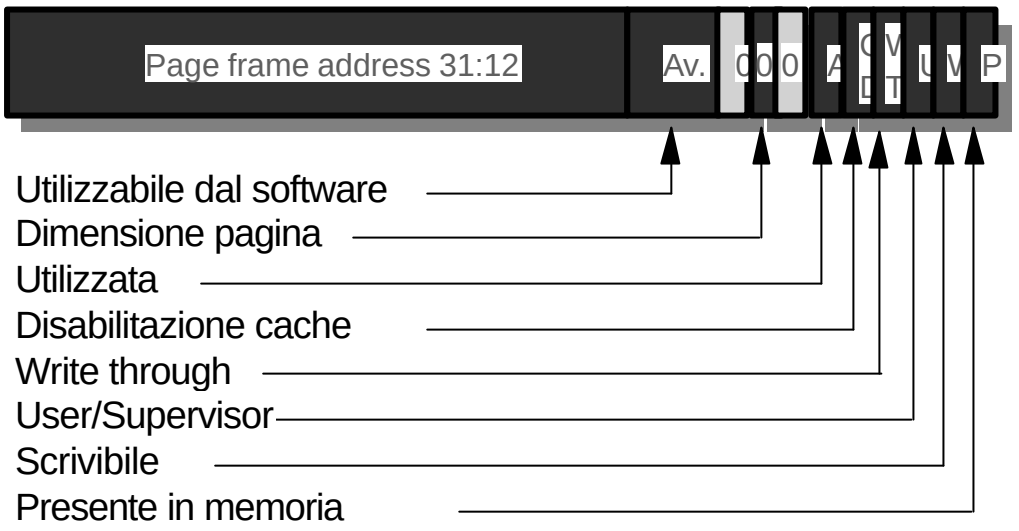
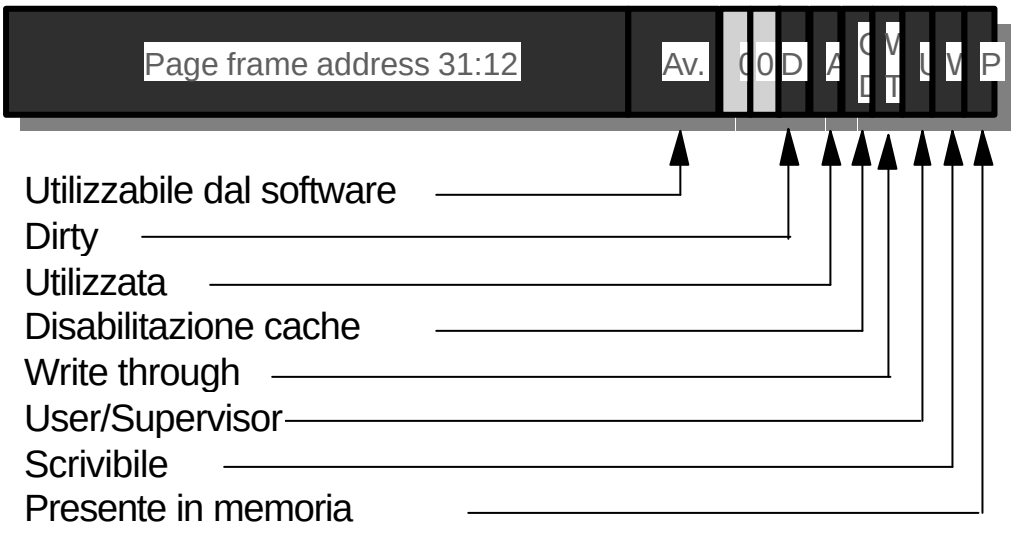


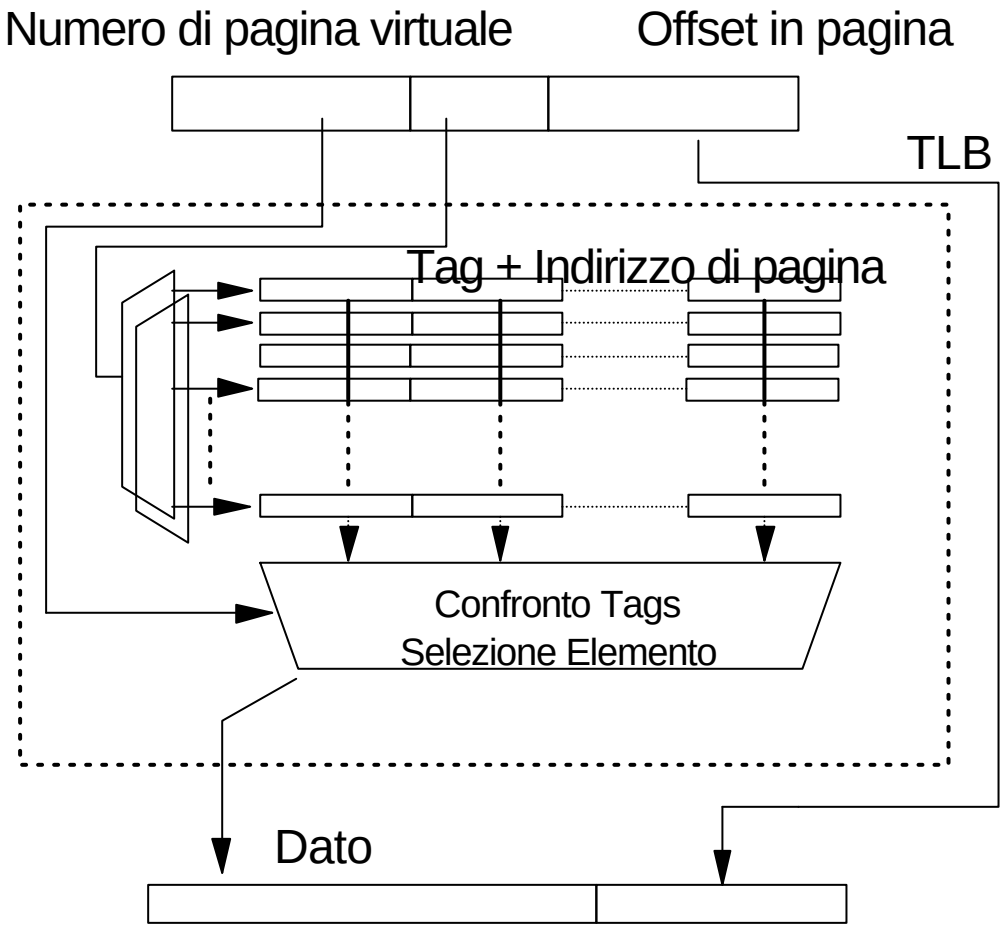
Tabelle delle Pagine da 4K





TLB

Translation lookaside buffer (TLB)
cache che memorizza le page entries piu' recentemente usate (Intel garantisce 98% hit rate)

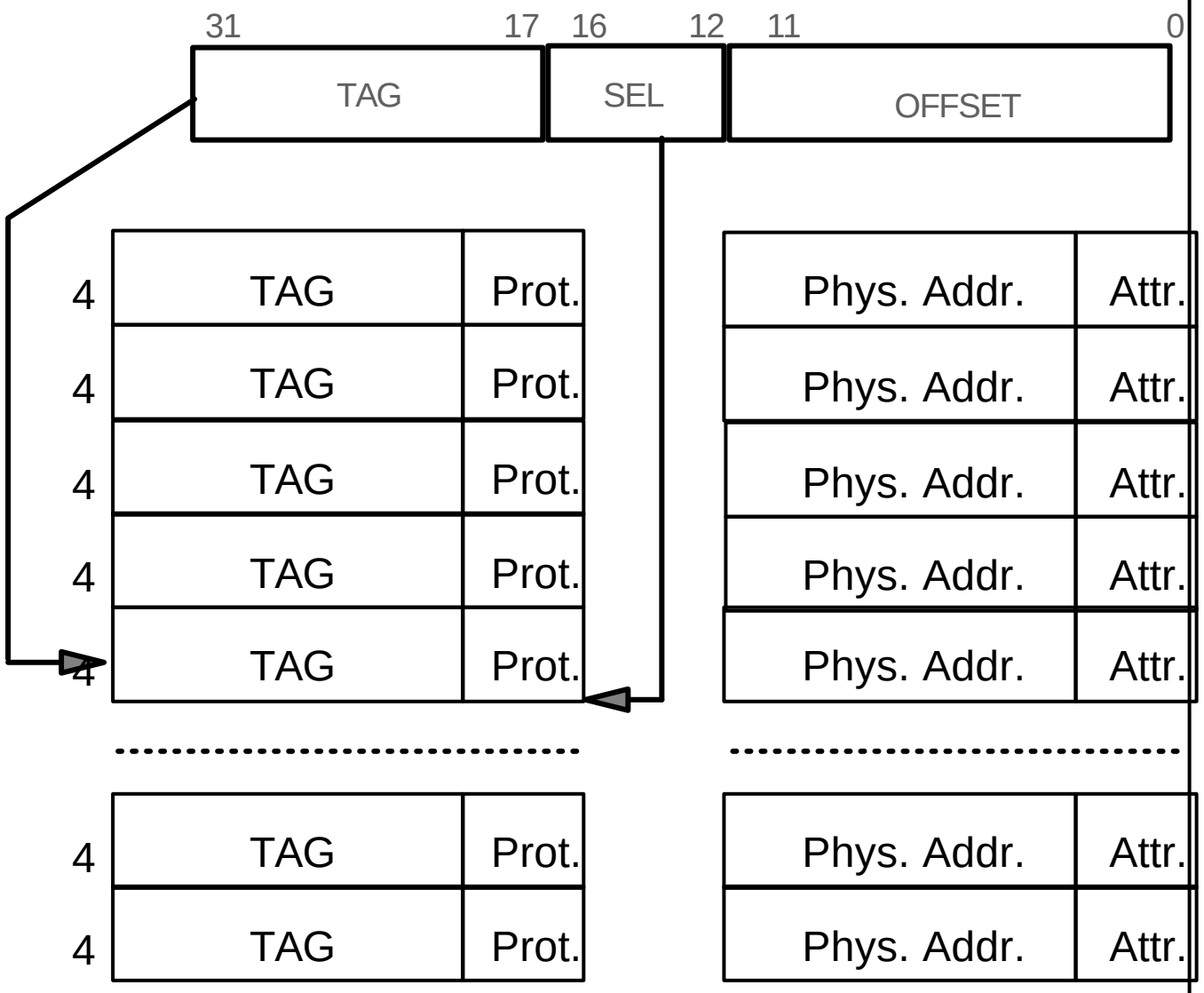


PENTIUM TLB

- TLB del PENTIUM: due per DATI (4 vie-64 elementi per pagine da 4 K oppure 4 vie 8 elementi per pagine da 4 MB) e uno per ISTRUZIONI (4 vie 32 elementi per entrambi i tipi di pagine)
- I TLB dei dati sono doppia porta (due pipelines)
- I TLB dei dati e delle istruzioni sono protetti da bit di parita'
- Il rimpiazzamento avviene tramite un algoritmo LRU che richiede 3 bit per set

PENTIUM TLB

CODE TLB (pagine da 4 KB)
4-way, 32 entries



Esiste un secondo TLB (8 elementi - 4 vie)
per le pagine da 4 MB

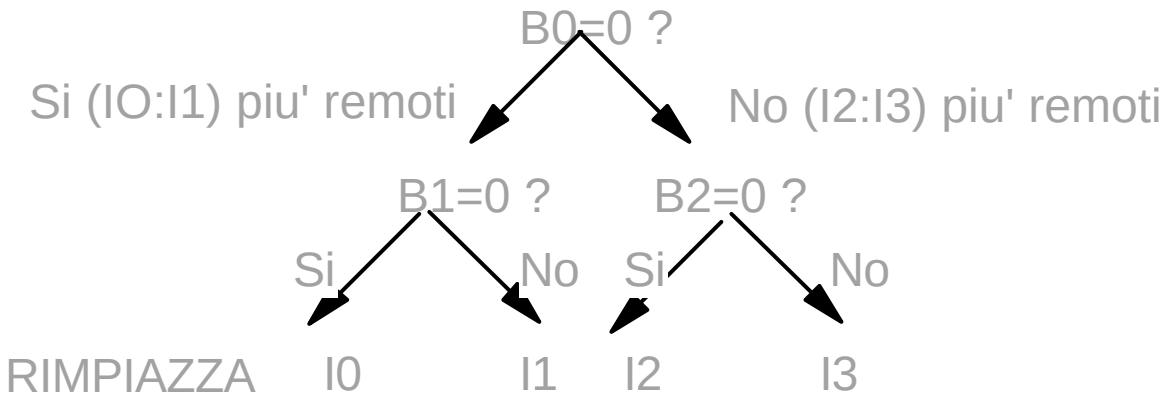
Pseudo LRU

- Meccanismo di rimpiazzamento (per l TLB, analogo per le caches che sono pero' a due vie):
  - I 4 elementi di un set sono indicati con I0, I1, I2 e I3
  - Se una linea e' non valida viene rimpiazzata.
  - Vi sono tre bit (B0, B1 e B2) per set

- Se l'ultimo accesso al set e' stato a I0 o a I1 allora B0=1 altrimenti B0=0
- Se l'ultimo accesso alla coppia I0:I1 e' stato a I0 allora B1=1 altrimenti B1=0
- Se l'utlimo accesso alla coppia I2:I3 e' stato a I2 allora B2=1 altrimenti B2=0

All'atto del rimpiazzamento:

- La cache prima seleziona quale fra I0:I1 e I2:I3 ha avuto l'accesso meno recente (B0) e poi seleziona all'interno della coppia



Indirizzamento completo