Modularity through Memory Virtualization

Prof. George Candea

School of Computer & Communication Sciences

Objectives

- Understand memory virtualization
  - *the coolest form of modularization we have in operating systems*
- Understand more deeply the role played by naming in modularization
- Start developing a “system designer” mindset

Outline

- Enforced modularity
- Page tables
    - *Directory for mapping memory names (VA) to memory locations (PA)*
- Caching
- Constants in systems design

Enforced Modularity

Modularity Recap

- Names or not?
  - *Memory address names a memory location*
  - *Pointer*
    - `void*` names a memory location
    - `int*` names the location of an integer
  - *Virtual address names a physical memory location*
- Enforced modularity
  - *module boundaries provided by a mechanism external to the modules*

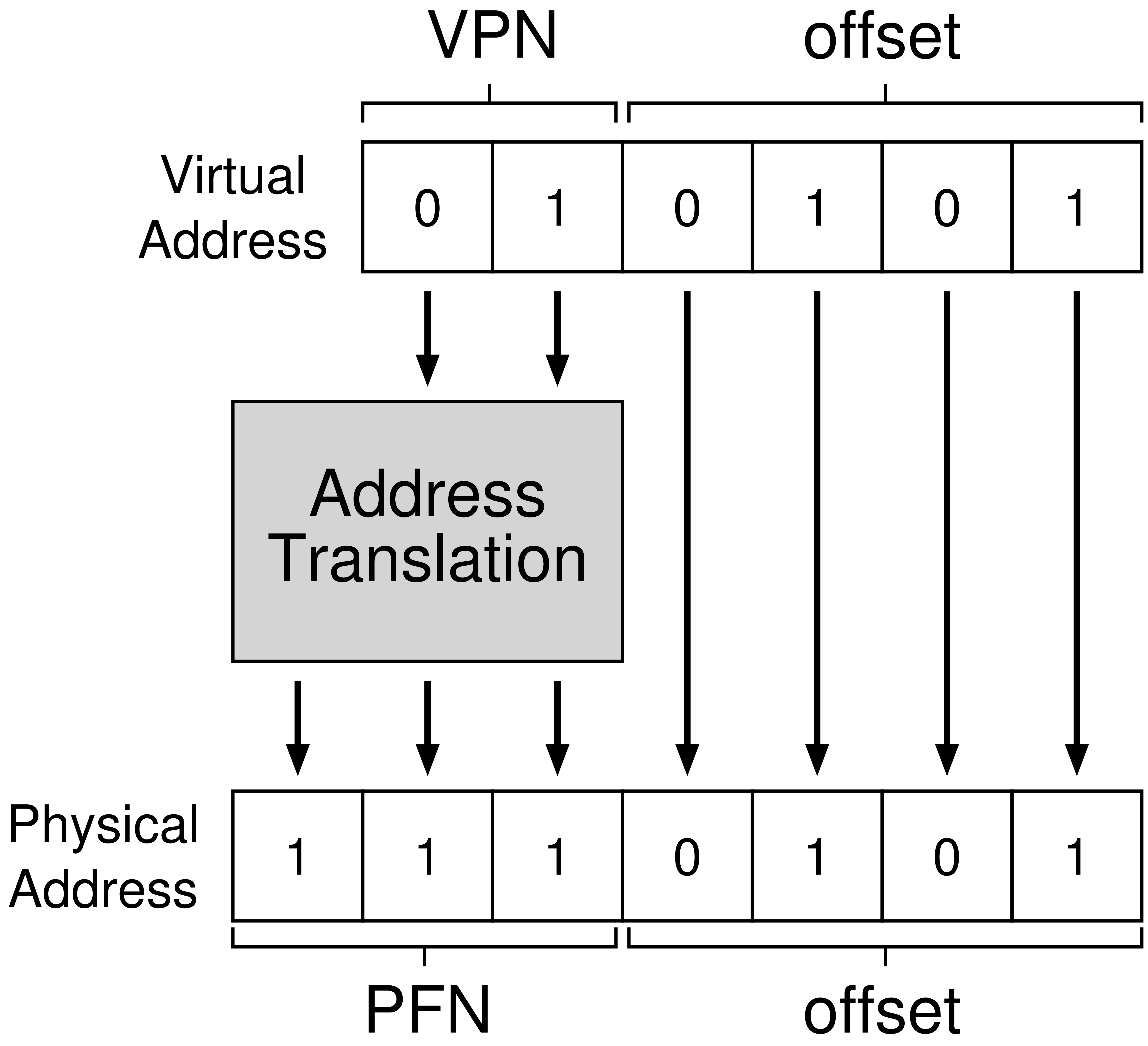
Enforced vs. Soft Modularity

- Module X interacts with module Y
  - *Encapsulation demands of X: module Y does not write to memory pages other than its own*
- Three options for enforced modularity?
  - *(a) Y is written carefully* ✘
  - *(b) The compiler used to compile Y guarantees it* ✔ *if X and Y share compiler*
  - *(c) Y's language runtime / interpreter guarantees it* ✔ *if X and Y share runtimes*

Role of Trust in Enforcing Modularity

- Further modularization options for X and Y
  - *(d) Y runs in a separate process / address space*
  - *(e) Y runs in a separate VM*
  - *(f) Y runs on a separate physical machine*
  - *(g) Y runs on an air-gapped machine*
- Enforced modularity = module boundaries provided by a mechanism external to the modules trusted by all modules

Virtual Memory & Page Tables



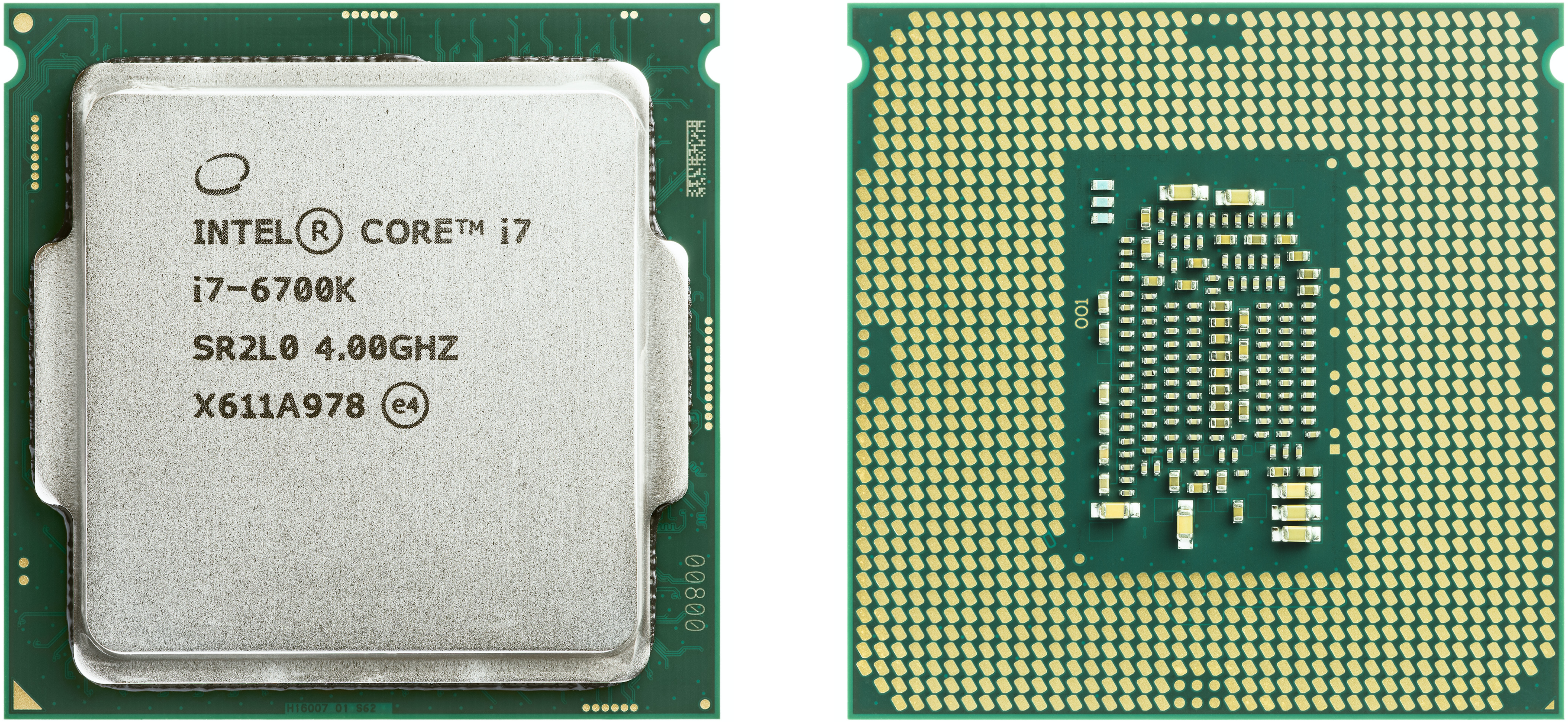
Virtual namespace

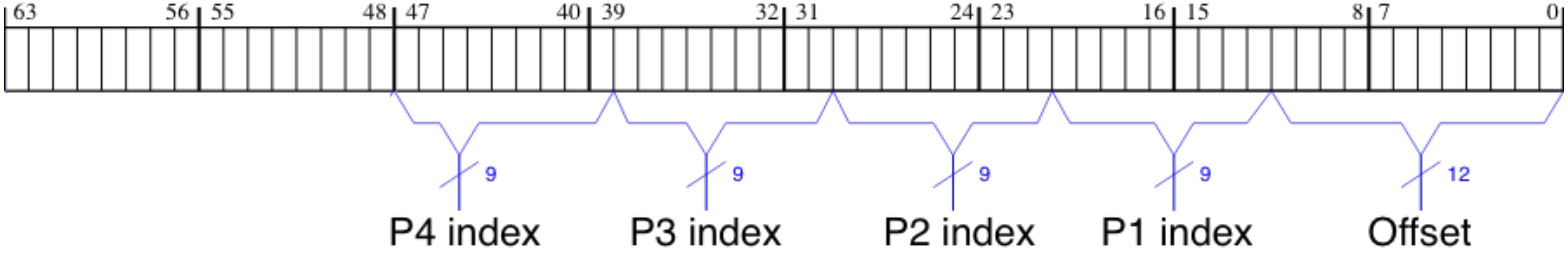
Name translation controls visibility

“if you can’t name it, you can’t use it"

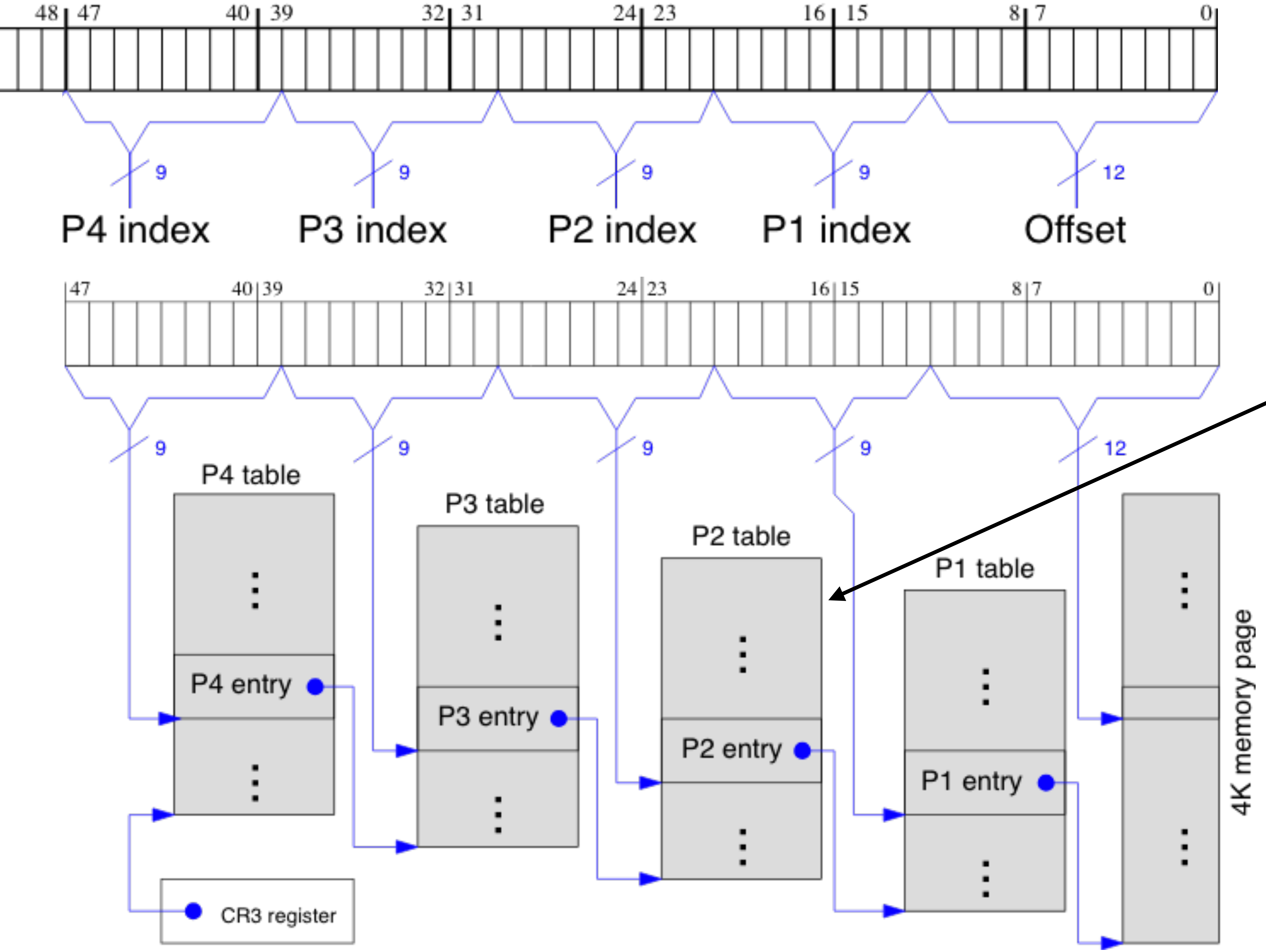
Physical namespace

Diagram courtesy of http://pages.cs.wisc.edu/~remzi/OSTEP


INTEL® CORE™ i7
i7-6700K
SR2L0 4.00GHZ
X611A978 e4
00800
H16007 01 S62
001



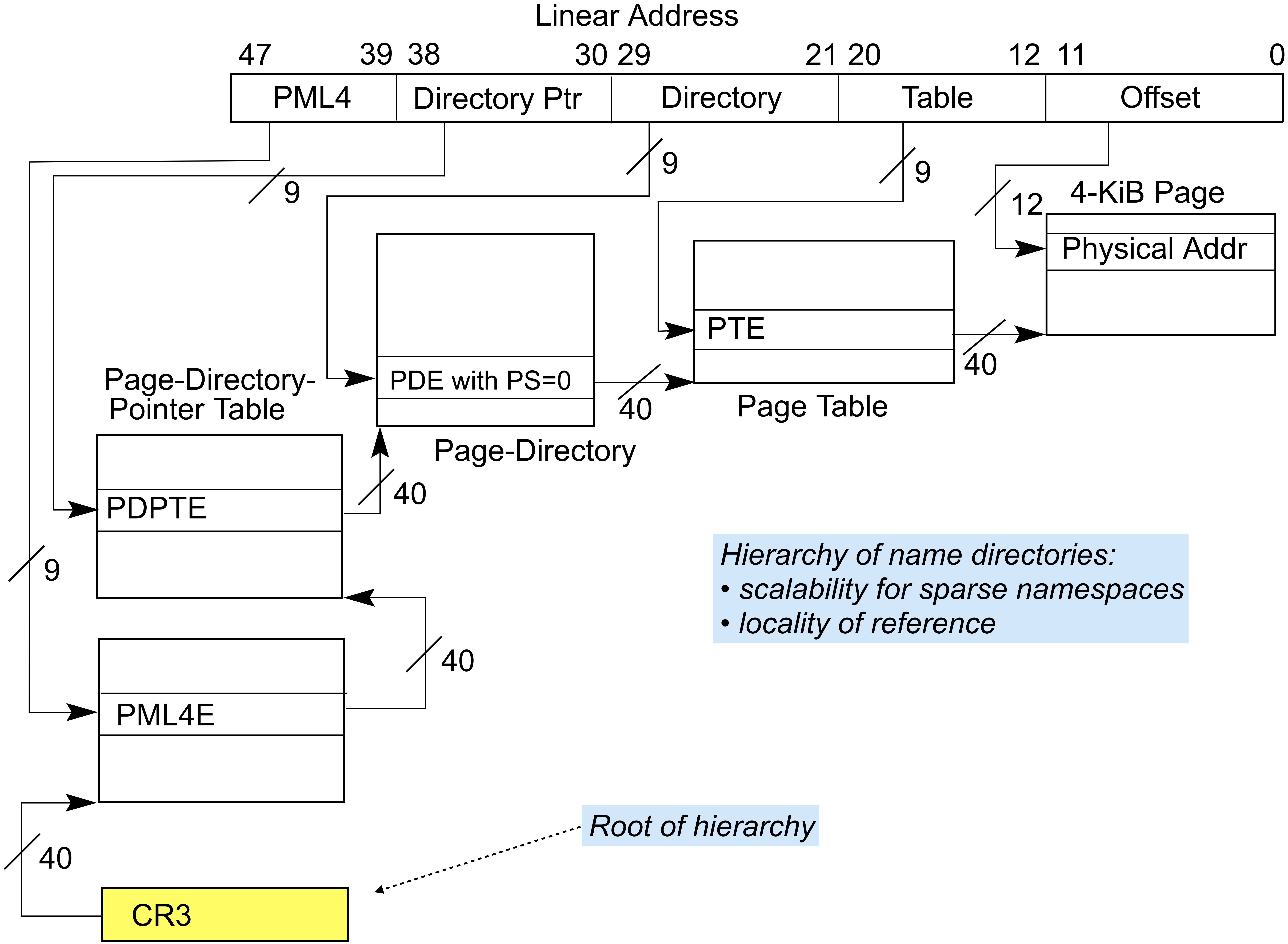
Diagrams courtesy of https://os.phil-opp.com/page-tables/



2^9=512 entries
8 bytes / entry

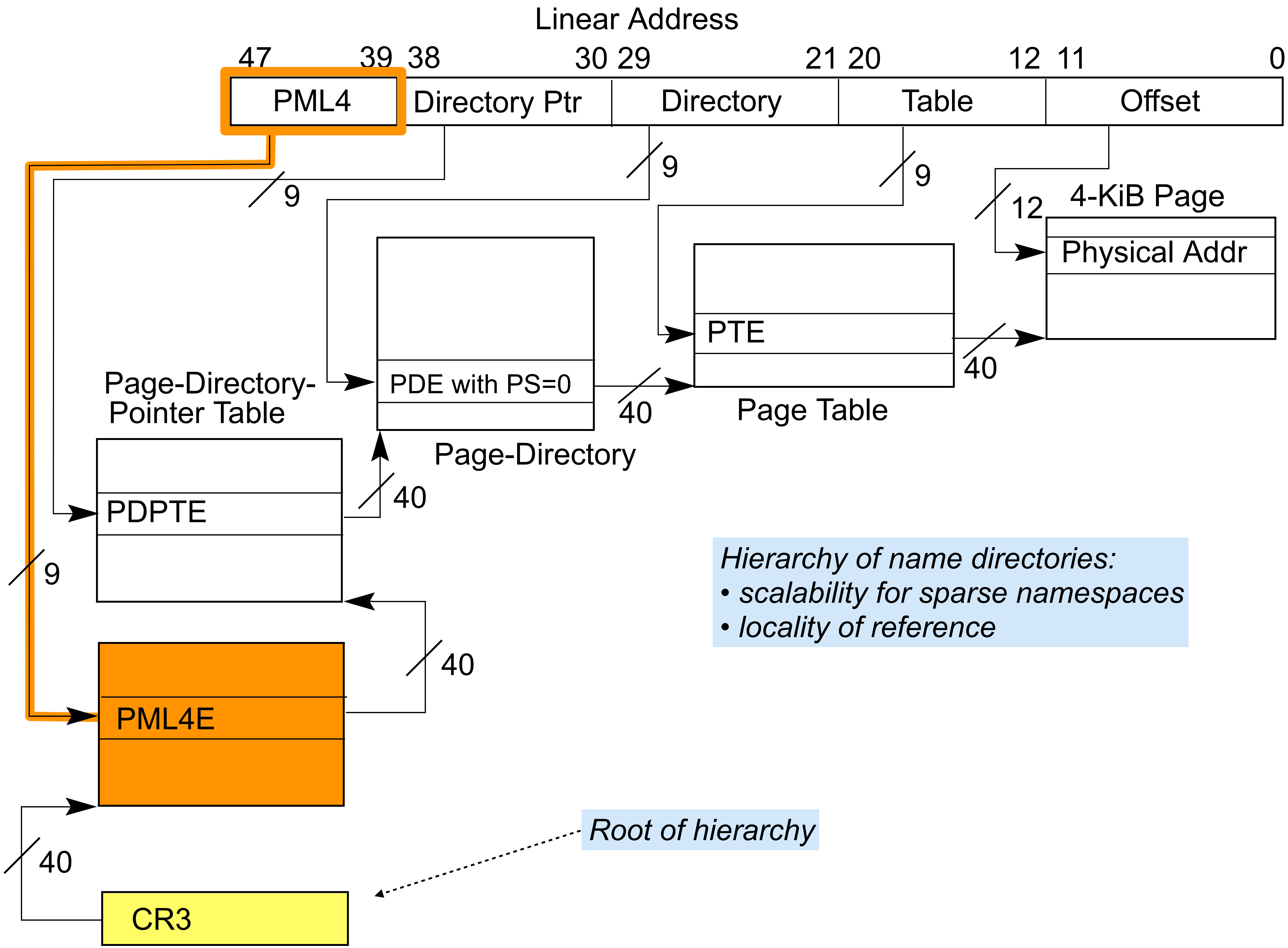
Hierarchy of name directories:
- *scalability for sparse namespaces*
- *locality of reference*

Diagrams courtesy of https://os.phil-opp.com/page-tables/



Hierarchy of name directories:
- *scalability for sparse namespaces*
- *locality of reference*

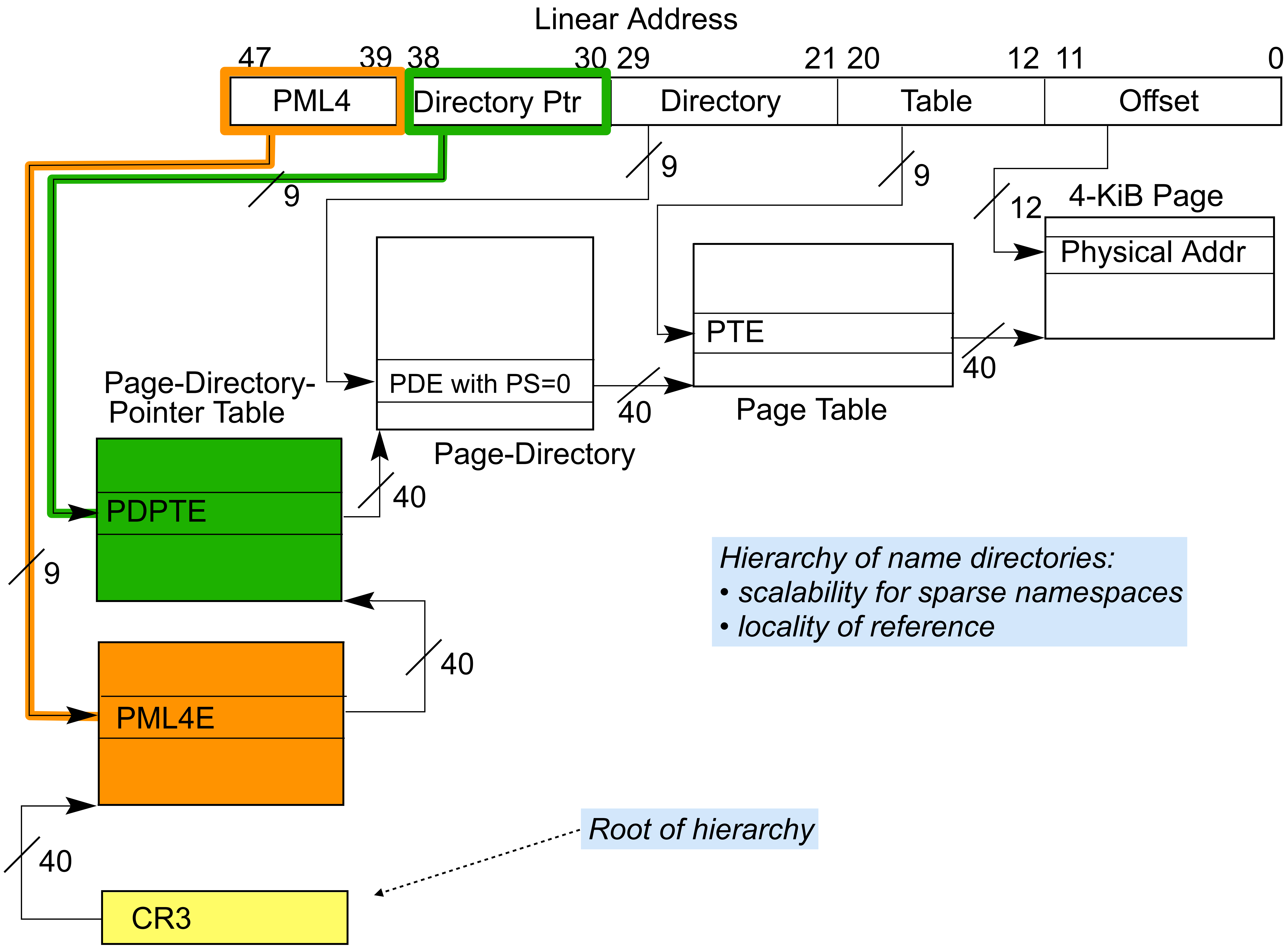
Root of hierarchy



Hierarchy of name directories:

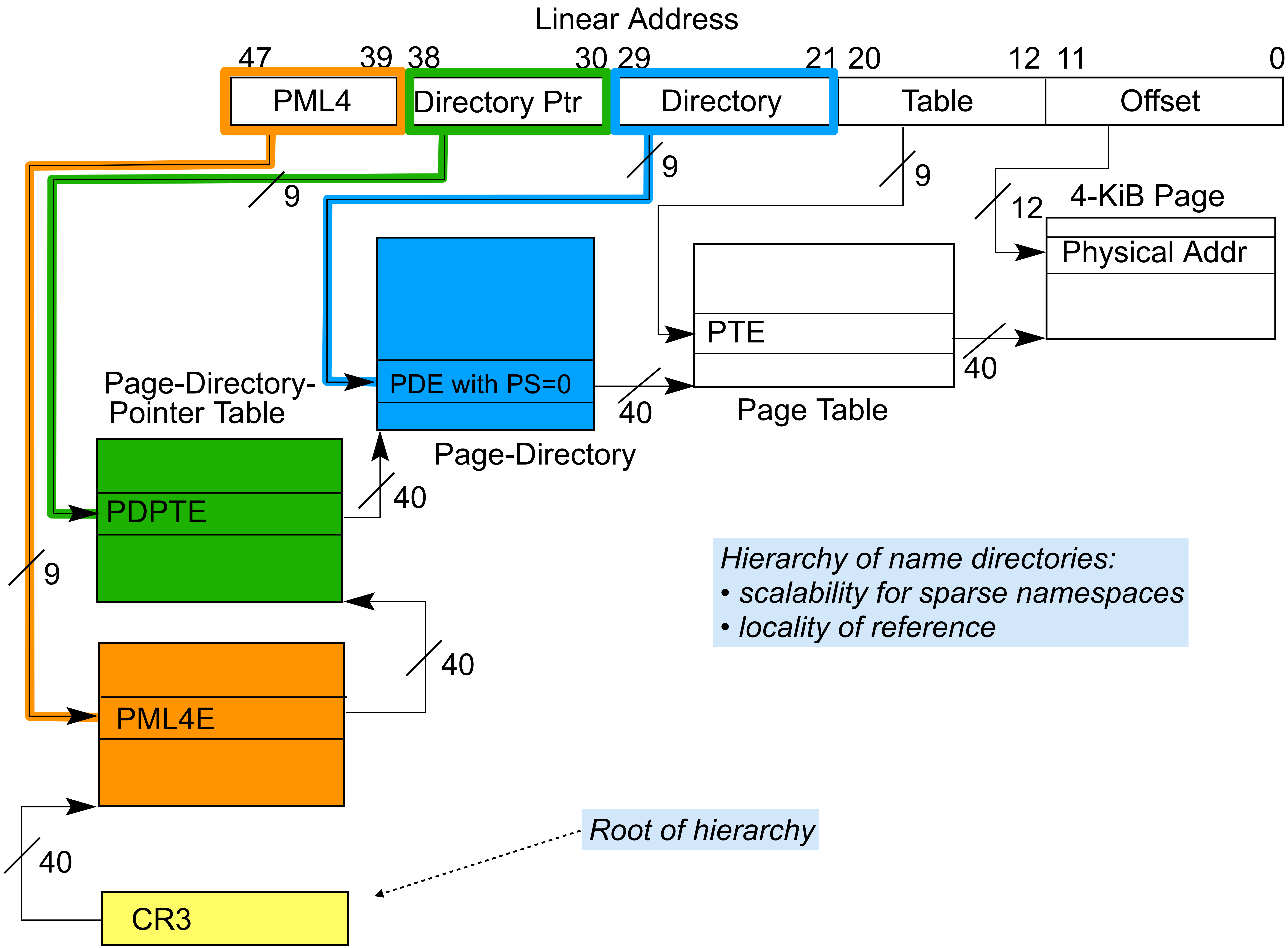
- *scalability for sparse namespaces*
- *locality of reference*

Root of hierarchy



Hierarchy of name directories:
- *scalability for sparse namespaces*
- *locality of reference*

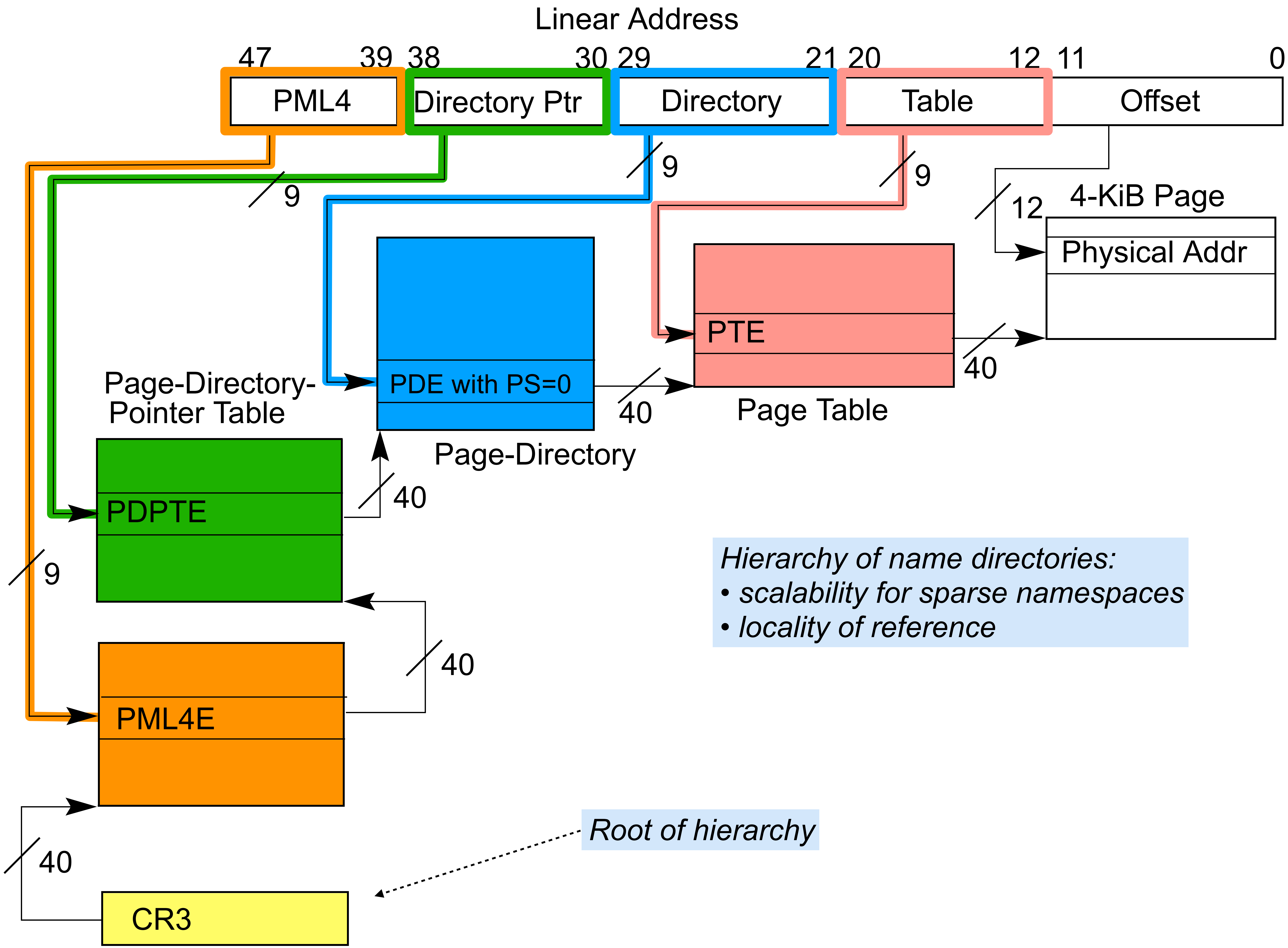
Root of hierarchy



Hierarchy of name directories:

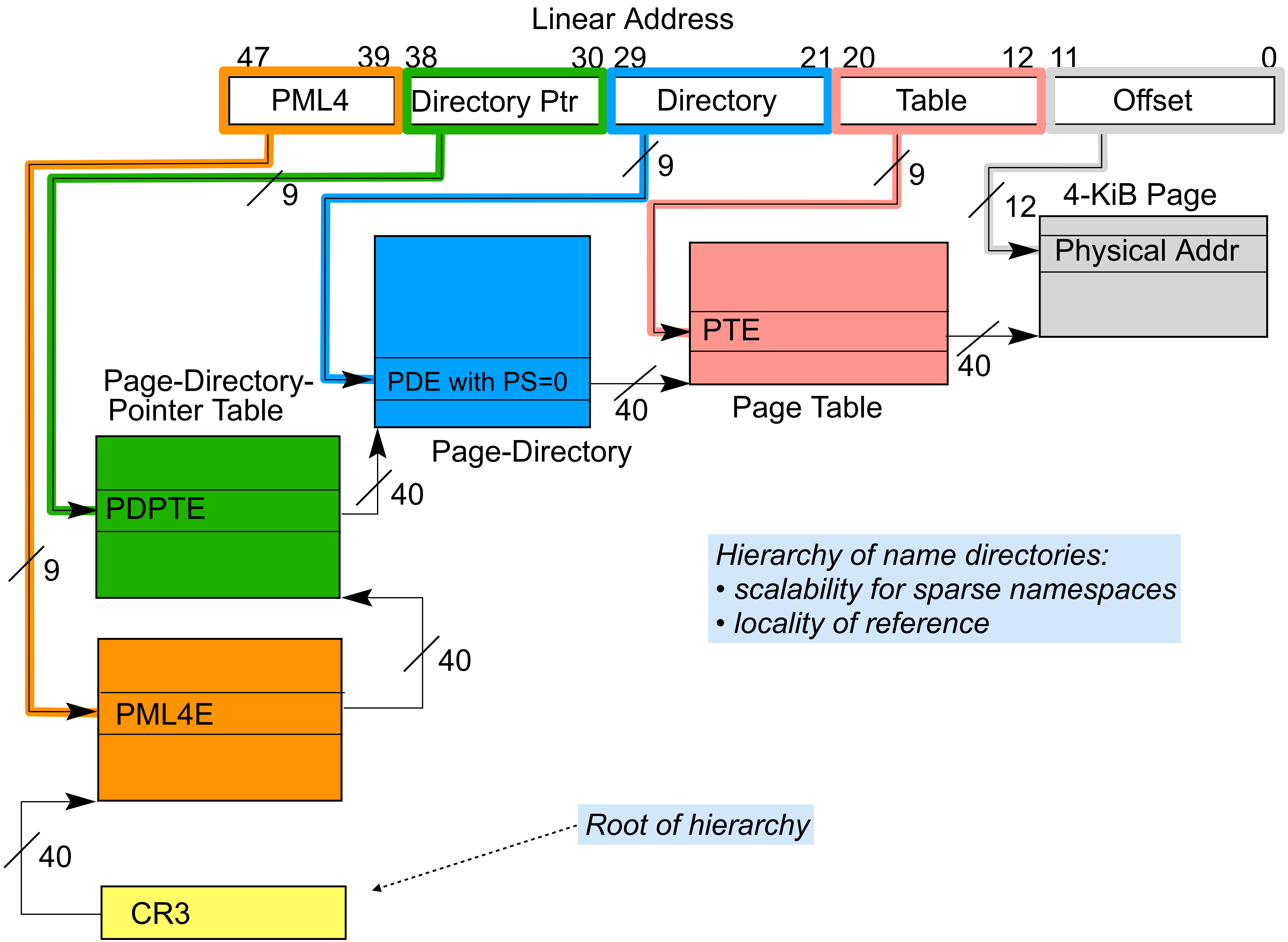
- *scalability for sparse namespaces*
- *locality of reference*

Root of hierarchy



Hierarchy of name directories:
- *scalability for sparse namespaces*
- *locality of reference*

Root of hierarchy



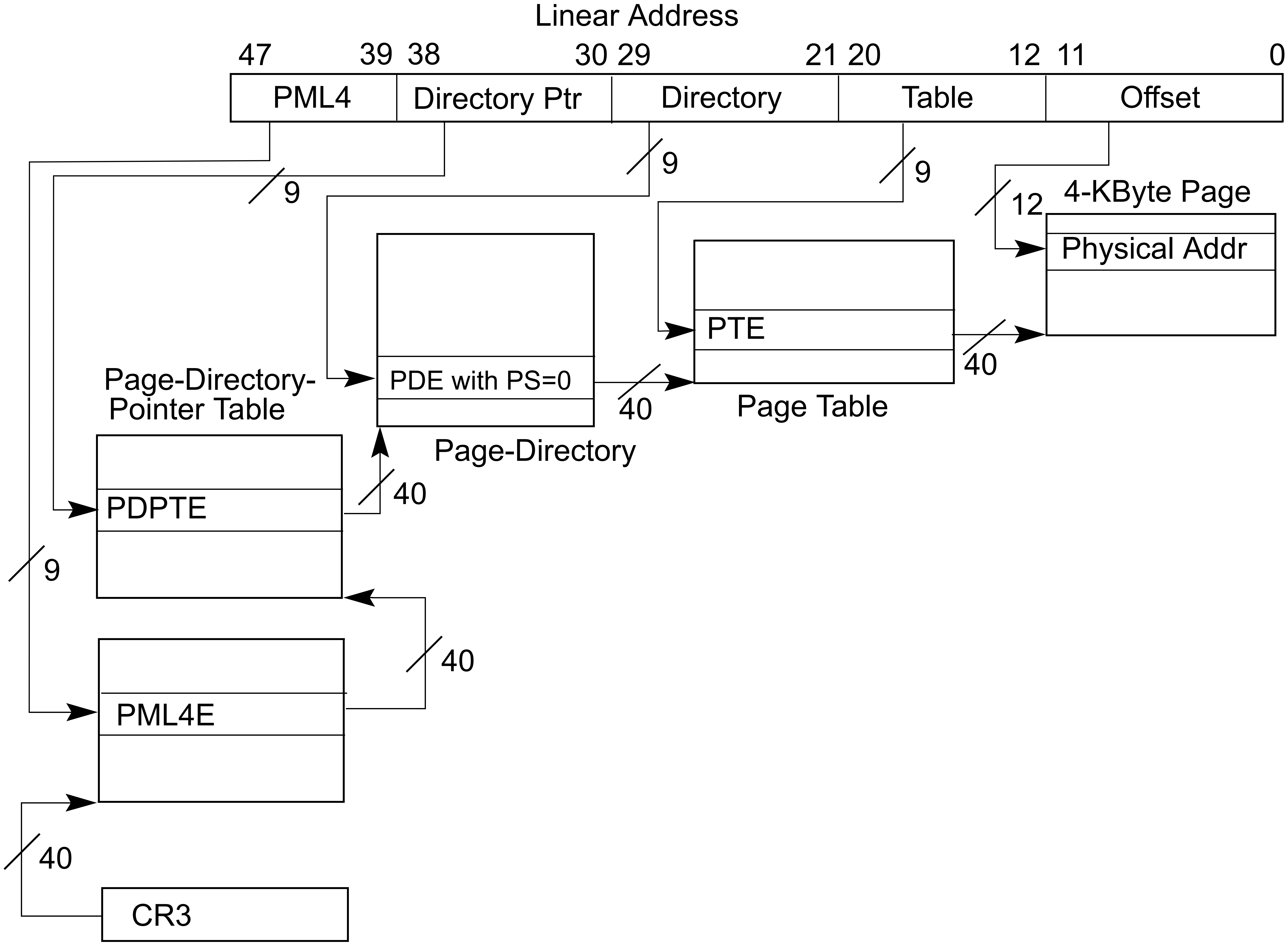
Hierarchy of name directories:

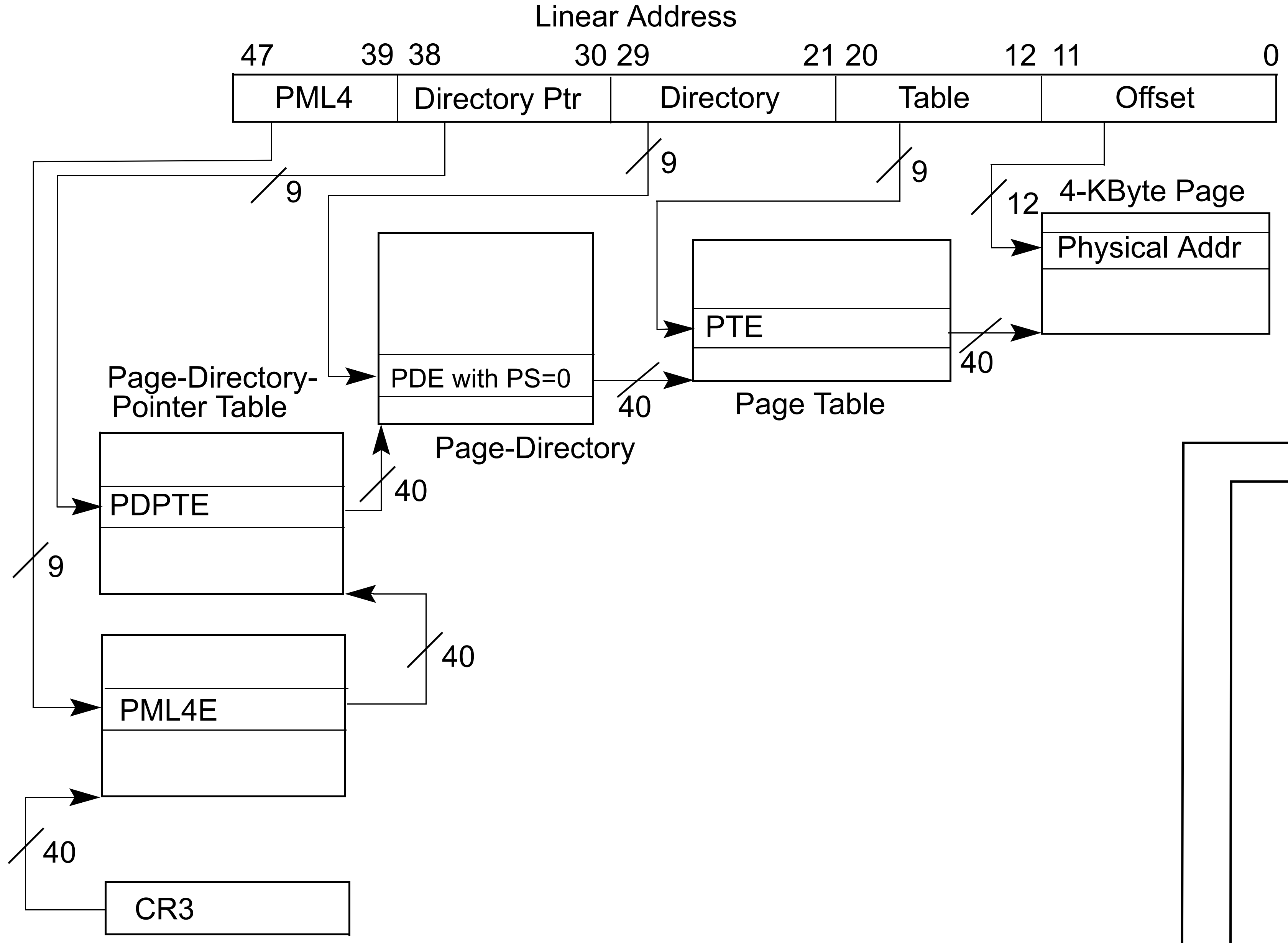
- *scalability for sparse namespaces*
- *locality of reference*

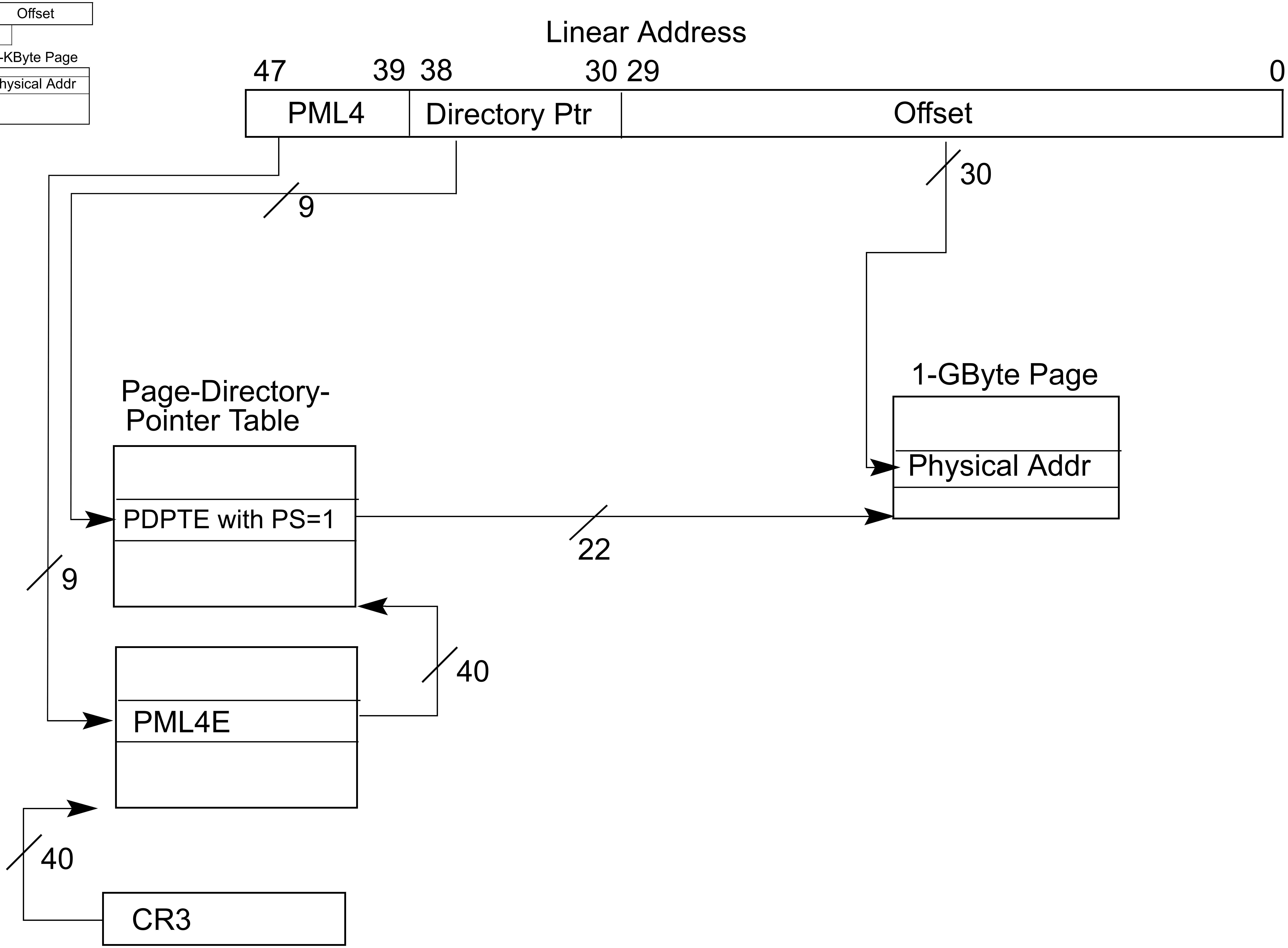
Root of hierarchy

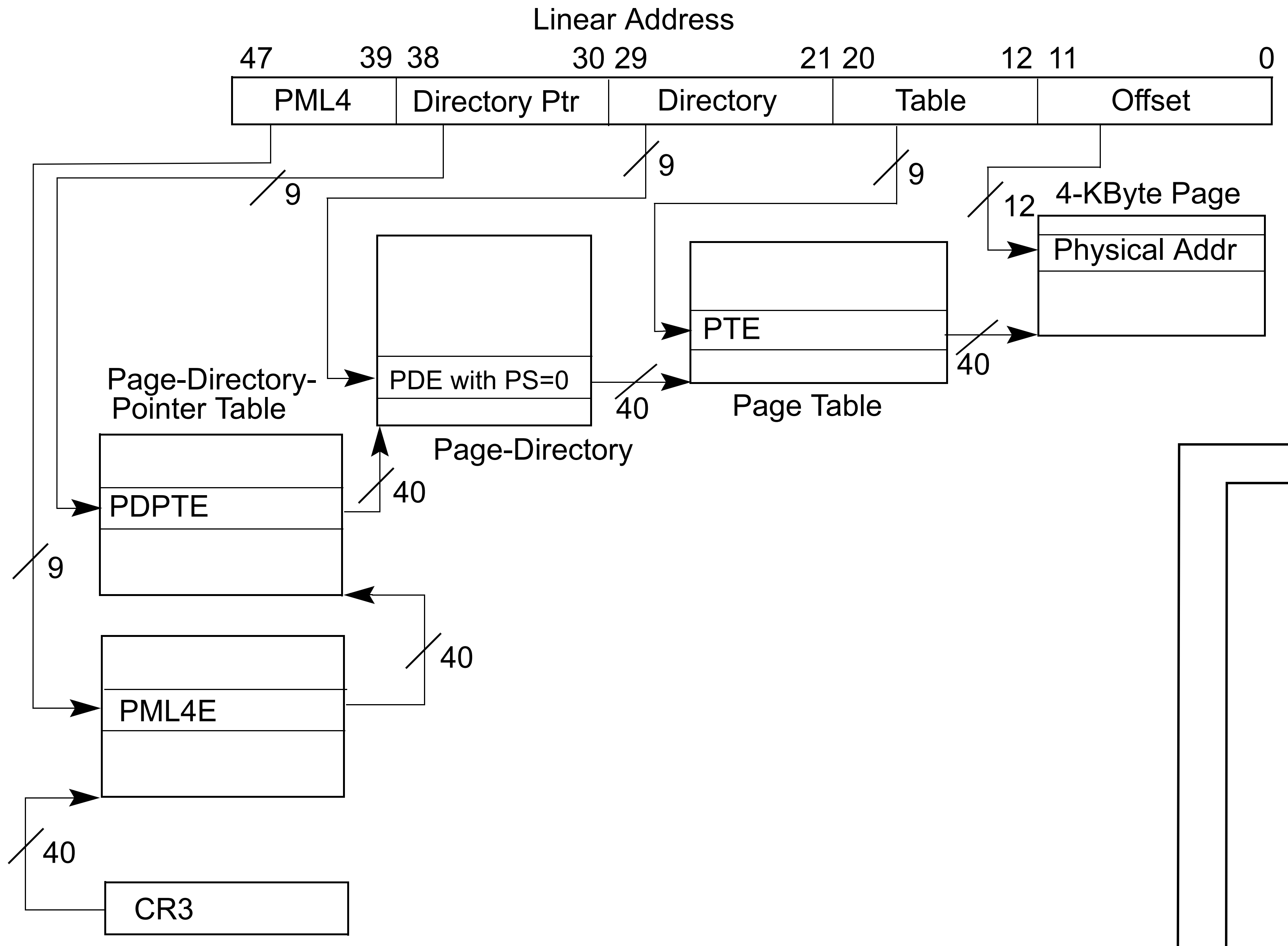
<table>
<tr><th>Execute disabled</th><th colspan="43"></th><th>Global</th><th>Page attribute type</th><th>Dirty</th><th>Accessed</th><th>Caching disabled</th><th>Write-through caching</th><th>User / Supervisor</th><th>Write / Read-only</th><th>Present</th><th></th></tr>
<tr><th>63</th><th>62</th><th>61</th><th>60</th><th>59</th><th>58</th><th>57</th><th>56</th><th>55</th><th>54</th><th>53</th><th>52</th><th>51</th><th></th><th>M[1]</th><th>M-1</th><th></th><th></th><th>32</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th></th></tr>
<tr><td colspan="15">Reserved</td><td colspan="24">Address of PML4 table</td><td colspan="7">Ignored</td><td>PCD</td><td>PWT</td><td colspan="3">Ign.</td><td><b>CR3</b></td></tr>
<tr><td>XD[3]</td><td colspan="11">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of page-directory-pointer table</td><td colspan="4">Ign.</td><td>Rsvd</td><td>Ign</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td><b>1</b></td><td><b>PML4E: present</b></td></tr>
<tr><td colspan="50">Ignored</td><td><b>0</b></td><td><b>PML4E: not present</b></td></tr>
<tr><td>XD</td><td colspan="4">Prot. Key</td><td colspan="7">Ignored</td><td colspan="3">Rsvd.</td><td colspan="6">Address of 1GB page frame</td><td colspan="17">Reserved</td><td>PAT</td><td colspan="3">Ign.</td><td>G</td><td><b>1</b></td><td>D</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td><b>1</b></td><td><b>PDPTE: 1GB page</b></td></tr>
<tr><td>XD</td><td colspan="11">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of page directory</td><td colspan="4">Ign.</td><td><b>0</b></td><td>Ign</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td><b>1</b></td><td><b>PDPTE: page directory</b></td></tr>
<tr><td colspan="50">Ignored</td><td><b>0</b></td><td><b>PDTPE: not present</b></td></tr>
<tr><td>XD</td><td colspan="4">Prot. Key</td><td colspan="7">Ignored</td><td colspan="3">Rsvd.</td><td colspan="15">Address of 2MB page frame</td><td colspan="8">Reserved</td><td>PAT</td><td colspan="3">Ign.</td><td>G</td><td><b>1</b></td><td>D</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td><b>1</b></td><td><b>PDE: 2MB page</b></td></tr>
<tr><td>XD</td><td colspan="11">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of page table</td><td colspan="4">Ign.</td><td><b>0</b></td><td>Ign</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td><b>1</b></td><td><b>PDE: page table</b></td></tr>
<tr><td colspan="50">Ignored</td><td><b>0</b></td><td><b>PDE: not present</b></td></tr>
<tr><td>XD</td><td colspan="4">Prot. Key</td><td colspan="7">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of 4KB page frame</td><td colspan="3">Ign.</td><td>G</td><td>PAT</td><td>D</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td><b>1</b></td><td><b>PTE: 4KB page</b></td></tr>
<tr><td colspan="50">Ignored</td><td><b>0</b></td><td><b>PTE: not present</b></td></tr>
</table>

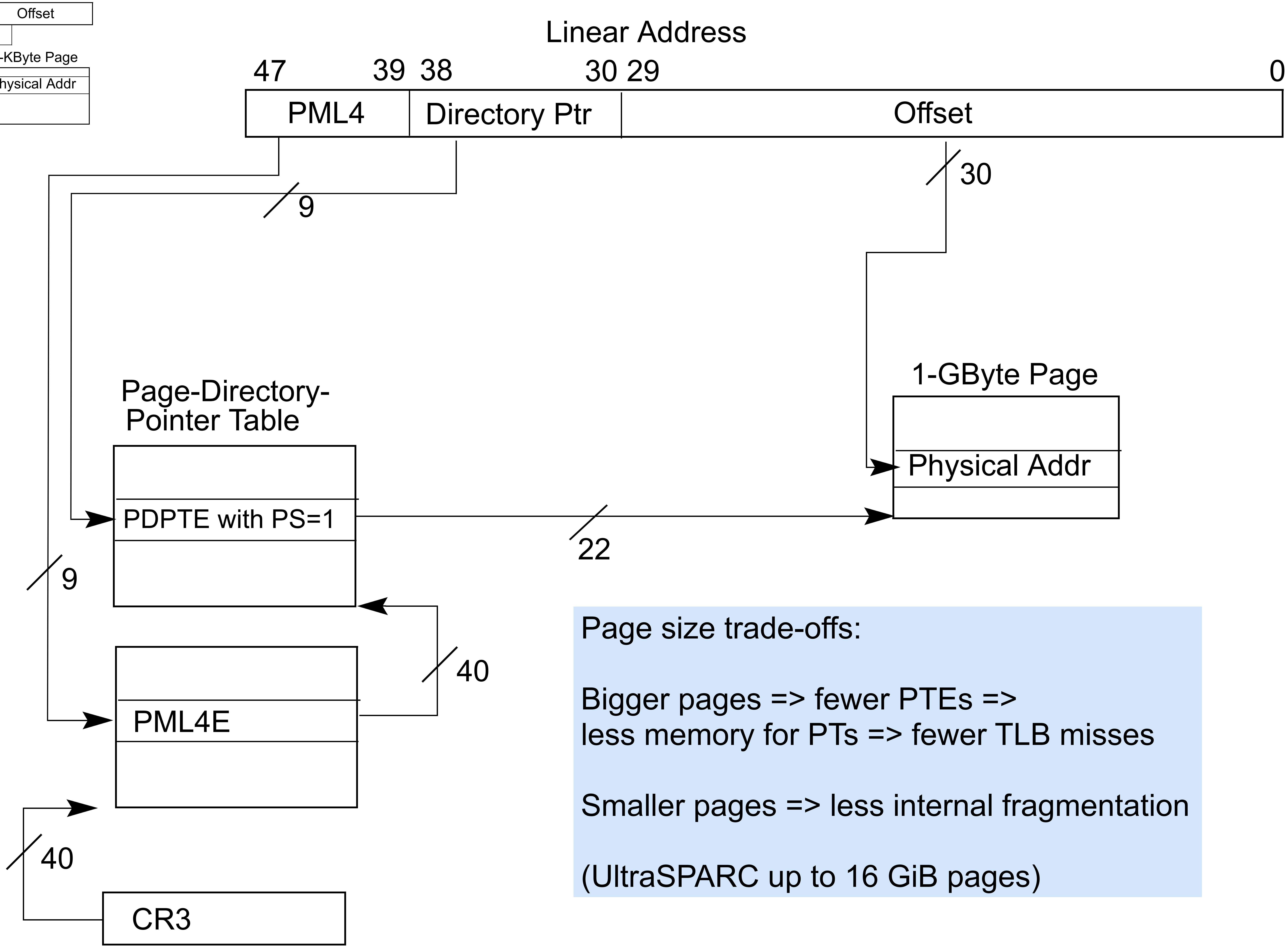
Intel® 64 and IA-32 Architectures Software Developer's Manual . Volume 3: System Programming Guide, May 2018


Linear Address
47
39
38
30
29
21
20
12
11
0
PML4
Directory Ptr
Directory
Table
Offset
9
9
9
12
4-KByte Page
Physical Addr
PDE with PS=0
40
PTE
40
Page Table
Page-Directory-
Pointer Table
Page-Directory
PDPTE
40
9
40
PML4E
40
CR3


Linear Address
47
39
38
30
29
21
20
12
11
0
PML4
Directory Ptr
Directory
Table
Offset
9
9
9
12
4-KByte Page
Physical Addr
PTE
40
PDE with PS=0
40
Page Table
Page-Directory-
Pointer Table
Page-Directory
PDPTE
40
9
40
PML4E
40
CR3


Linear Address
47
39
38
30
29
0
PML4
Directory Ptr
Offset
9
30
1-GByte Page
Physical Addr
Page-Directory-
Pointer Table
PDPTE with PS=1
22
9
40
PML4E
40
CR3



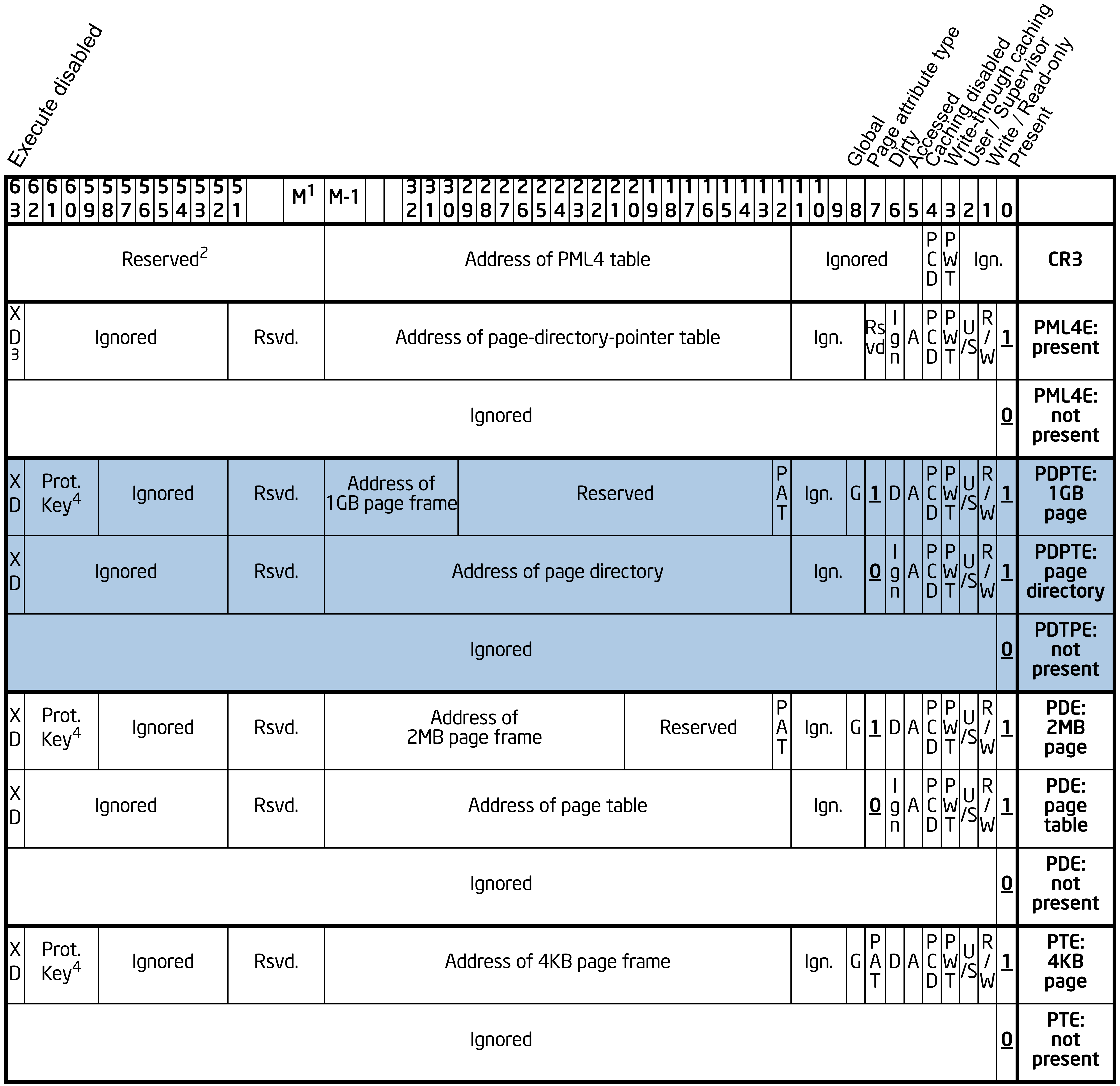


Page size trade-offs:

Bigger pages => fewer PTEs =>
less memory for PTs => fewer TLB misses

Smaller pages => less internal fragmentation

(UltraSPARC up to 16 GiB pages)



<table>
<tr><td>Execute disabled</td><td colspan="40"></td><td>Global</td><td>Page attribute type</td><td>Dirty</td><td>Accessed</td><td>Caching disabled</td><td>Write-through caching</td><td>User / Supervisor</td><td>Write / Read-only</td><td>Present</td><td></td></tr>
<tr><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td></td><td>M[1]</td><td>M-1</td><td></td><td></td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
<tr><td colspan="15">Reserved[2]</td><td colspan="24">Address of PML4 table</td><td colspan="7">Ignored</td><td>PCD</td><td>PWT</td><td colspan="3">Ign.</td><td>CR3</td></tr>
<tr><td>XD[3]</td><td colspan="11">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of page-directory-pointer table</td><td colspan="4">Ign.</td><td>Rsvd</td><td>Ign</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td>1</td><td>PML4E: present</td></tr>
<tr><td colspan="50">Ignored</td><td>0</td><td>PML4E: not present</td></tr>
<tr><td>XD</td><td colspan="4">Prot. Key[4]</td><td colspan="7">Ignored</td><td colspan="3">Rsvd.</td><td colspan="6">Address of 1GB page frame</td><td colspan="17">Reserved</td><td>PAT</td><td colspan="3">Ign.</td><td>G</td><td>1</td><td>D</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td>1</td><td>PDPTE: 1GB page</td></tr>
<tr><td>XD</td><td colspan="11">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of page directory</td><td colspan="4">Ign.</td><td>0</td><td>Ign</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td>1</td><td>PDPTE: page directory</td></tr>
<tr><td colspan="50">Ignored</td><td>0</td><td>PDTPE: not present</td></tr>
<tr><td>XD</td><td colspan="4">Prot. Key[4]</td><td colspan="7">Ignored</td><td colspan="3">Rsvd.</td><td colspan="15">Address of 2MB page frame</td><td colspan="8">Reserved</td><td>PAT</td><td colspan="3">Ign.</td><td>G</td><td>1</td><td>D</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td>1</td><td>PDE: 2MB page</td></tr>
<tr><td>XD</td><td colspan="11">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of page table</td><td colspan="4">Ign.</td><td>0</td><td>Ign</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td>1</td><td>PDE: page table</td></tr>
<tr><td colspan="50">Ignored</td><td>0</td><td>PDE: not present</td></tr>
<tr><td>XD</td><td colspan="4">Prot. Key[4]</td><td colspan="7">Ignored</td><td colspan="3">Rsvd.</td><td colspan="24">Address of 4KB page frame</td><td colspan="3">Ign.</td><td>G</td><td>PAT</td><td>D</td><td>A</td><td>PCD</td><td>PWT</td><td>U/S</td><td>R/W</td><td>1</td><td>PTE: 4KB page</td></tr>
<tr><td colspan="50">Ignored</td><td>0</td><td>PTE: not present</td></tr>
</table>

Intel® 64 and IA-32 Architectures Software Developer's Manual .
Volume 3: System Programming Guide, May 2018

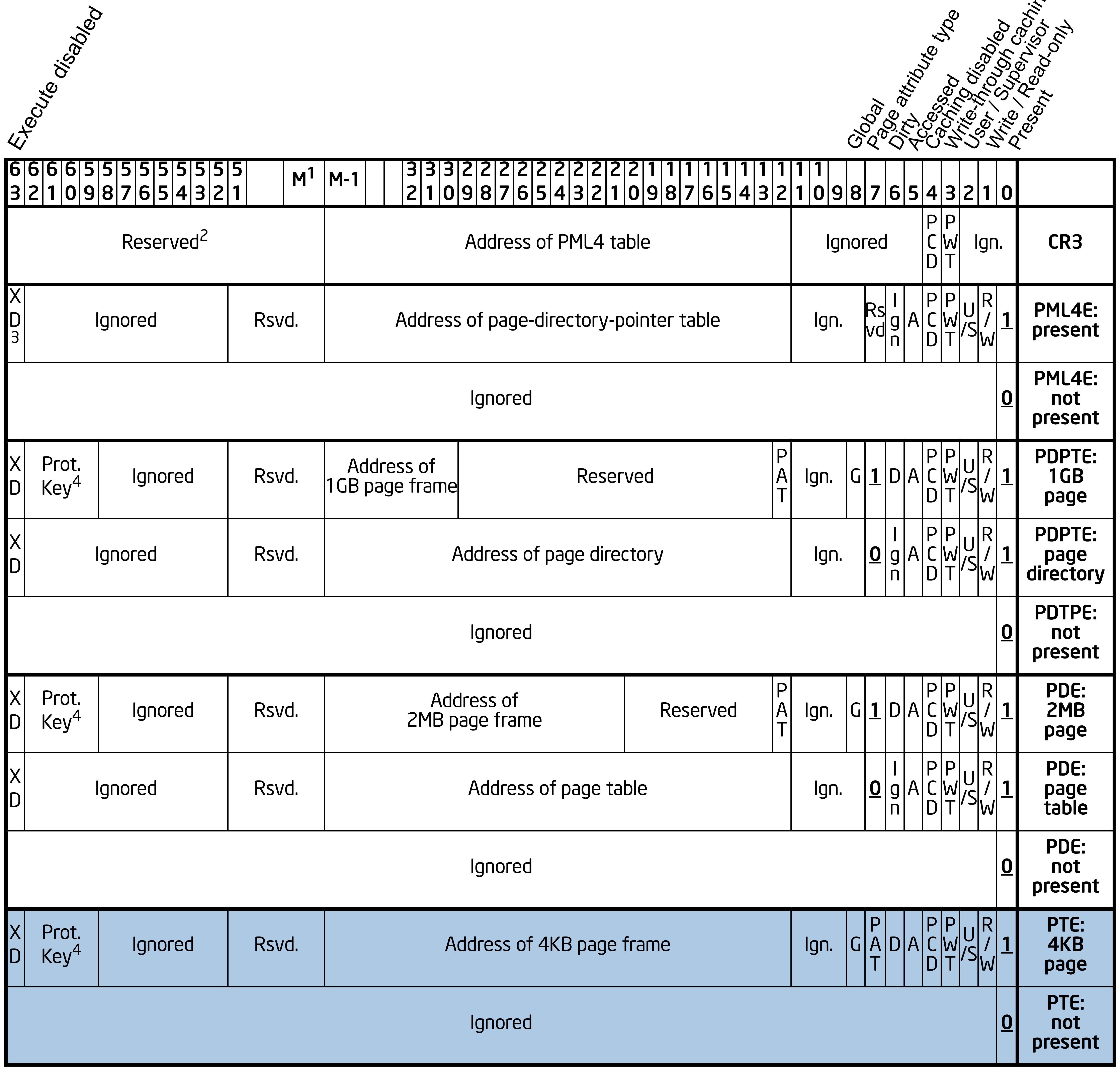
Execute disabled

Global
Page attribute type
Dirty
Accessed
Caching disabled
Write-through caching
User / Supervisor
Write / Read-only
Present

63 62 61 60 59 58 57 56 55 54 53 52 51 M[1] M-1 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

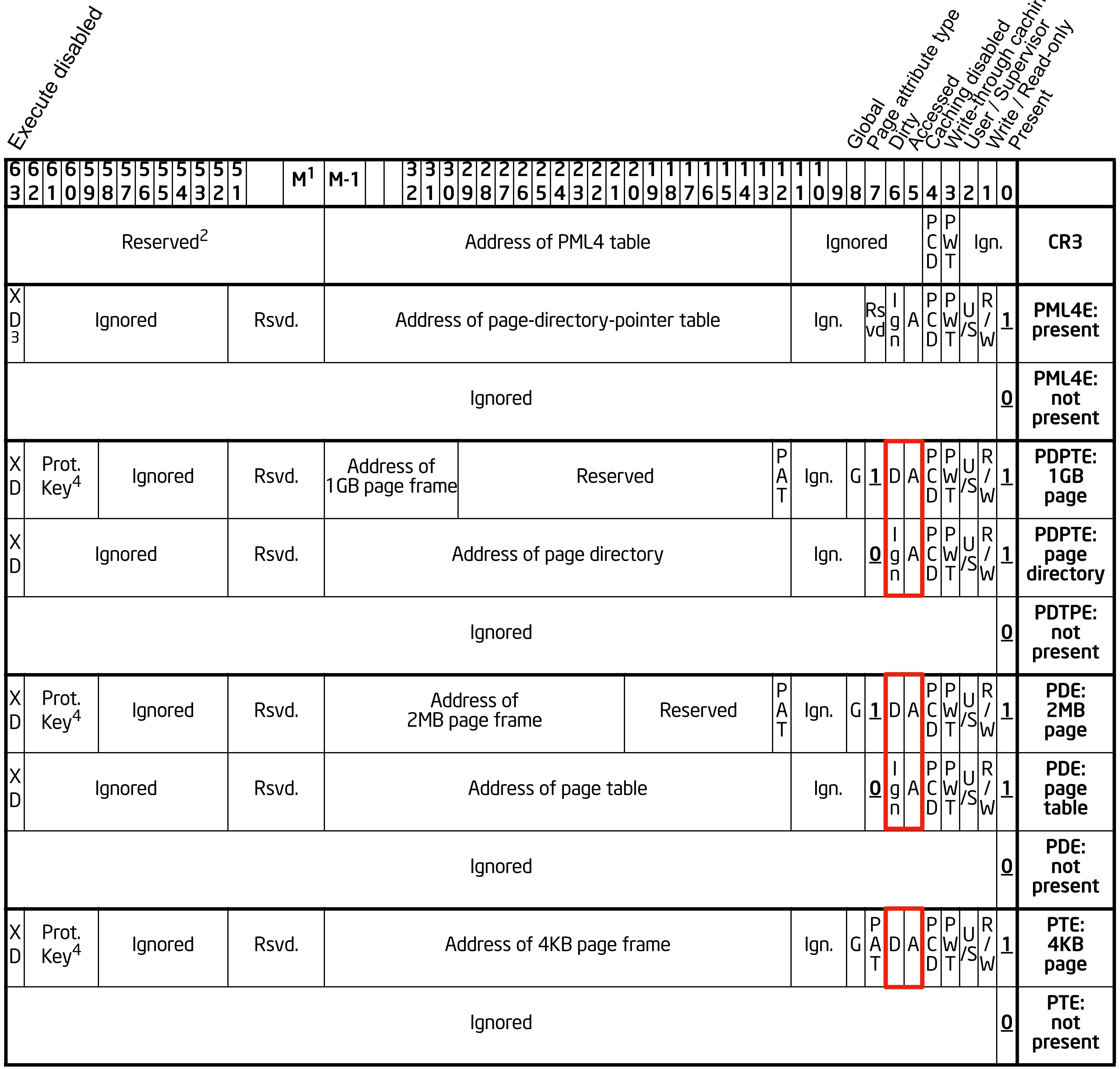
CR3	Reserved[2]	Address of PML4 table	Ignored	PCD	PWT	Ign.											
PML4E: present	XD[3]	Ignored	Rsvd.	Address of page-directory-pointer table	Ign.	Rsvd	Ign	A	PCD	PWT	U/S	R/W	**1**				
PML4E: not present	Ignored	**0**															
PDPTE: 1GB page	XD	Prot. Key[4]	Ignored	Rsvd.	Address of 1GB page frame	Reserved	PAT	Ign.	G	**1**	D	A	PCD	PWT	U/S	R/W	**1**
PDPTE: page directory	XD	Ignored	Rsvd.	Address of page directory	Ign.	**0**	Ign	A	PCD	PWT	U/S	R/W	**1**				
PDTPE: not present	Ignored	**0**															
PDE: 2MB page	XD	Prot. Key[4]	Ignored	Rsvd.	Address of 2MB page frame	Reserved	PAT	Ign.	G	**1**	D	A	PCD	PWT	U/S	R/W	**1**
PDE: page table	XD	Ignored	Rsvd.	Address of page table	Ign.	**0**	Ign	A	PCD	PWT	U/S	R/W	**1**				
PDE: not present	Ignored	**0**															
PTE: 4KB page	XD	Prot. Key[4]	Ignored	Rsvd.	Address of 4KB page frame	Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	**1**		
PTE: not present	Ignored	**0**															

Intel® 64 and IA-32 Architectures Software Developer's Manual . Volume 3: System Programming Guide, May 2018



Execute disabled										Global	Page attribute type	Dirty	Accessed	Caching disabled	Write-through caching	User / Supervisor	Write / Read-only	Present	
63	62 61 60 59	58 57 56 55 54 53 52	51 … M[1]	M-1 … 32 31 30	29 28 27 26 25 24 23 22 21	20 19 18 17 16 15 14 13	12	11 10 9	8	7	6	5	4	3	2	1	0		
Reserved[2]				Address of PML4 table				Ignored					PCD	PWT	Ign.				**CR3**
XD[3]	Ignored		Rsvd.	Address of page-directory-pointer table				Ign.		Rsvd	Ign	A	PCD	PWT	U/S	R/W	1		**PML4E: present**
Ignored																	0		**PML4E: not present**
XD	Prot. Key[4]	Ignored	Rsvd.	Address of 1GB page frame	Reserved		PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1		**PDPTE: 1GB page**
XD	Ignored		Rsvd.	Address of page directory				Ign.		0	Ign	A	PCD	PWT	U/S	R/W	1		**PDPTE: page directory**
Ignored																	0		**PDTPE: not present**
XD	Prot. Key[4]	Ignored	Rsvd.	Address of 2MB page frame		Reserved	PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1		**PDE: 2MB page**
XD	Ignored		Rsvd.	Address of page table				Ign.		0	Ign	A	PCD	PWT	U/S	R/W	1		**PDE: page table**
Ignored																	0		**PDE: not present**
XD	Prot. Key[4]	Ignored	Rsvd.	Address of 4KB page frame				Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	1		**PTE: 4KB page**
Ignored																	0		**PTE: not present**

Intel® 64 and IA-32 Architectures Software Developer's Manual .
Volume 3: System Programming Guide, May 2018


Execute disabled
Global
Page attribute type
Dirty
Accessed
Caching disabled
Write-through caching
User / Supervisor
Write / Read-only
Present

	63 62 61 60 59 58 57 56 55 54 53 52 51	M[1]	M-1	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0													
CR3	Reserved[2]	Address of PML4 table	Ignored	PCD	PWT	Ign.											
PML4E: present	XD[3]	Ignored	Rsvd.	Address of page-directory-pointer table	Ign.	Rsvd	Ign	A	PCD	PWT	U/S	R/W	1				
PML4E: not present	Ignored	0															
PDPTE: 1GB page	XD	Prot. Key[4]	Ignored	Rsvd.	Address of 1GB page frame	Reserved	PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1
PDPTE: page directory	XD	Ignored	Rsvd.	Address of page directory	Ign.	0	Ign	A	PCD	PWT	U/S	R/W	1				
PDTPE: not present	Ignored	0															
PDE: 2MB page	XD	Prot. Key[4]	Ignored	Rsvd.	Address of 2MB page frame	Reserved	PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1
PDE: page table	XD	Ignored	Rsvd.	Address of page table	Ign.	0	Ign	A	PCD	PWT	U/S	R/W	1				
PDE: not present	Ignored	0															
PTE: 4KB page	XD	Prot. Key[4]	Ignored	Rsvd.	Address of 4KB page frame	Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	1		
PTE: not present	Ignored	0															

Intel® 64 and IA-32 Architectures Software Developer's Manual .
Volume 3: System Programming Guide, May 2018

4 x 100 ns table lookup +
1 x 100 ns access =
500 ns =
~1,500 instructions @ 3 GHz clock

I could be doing a lot of useful work while waiting to read memory !

What to do ?

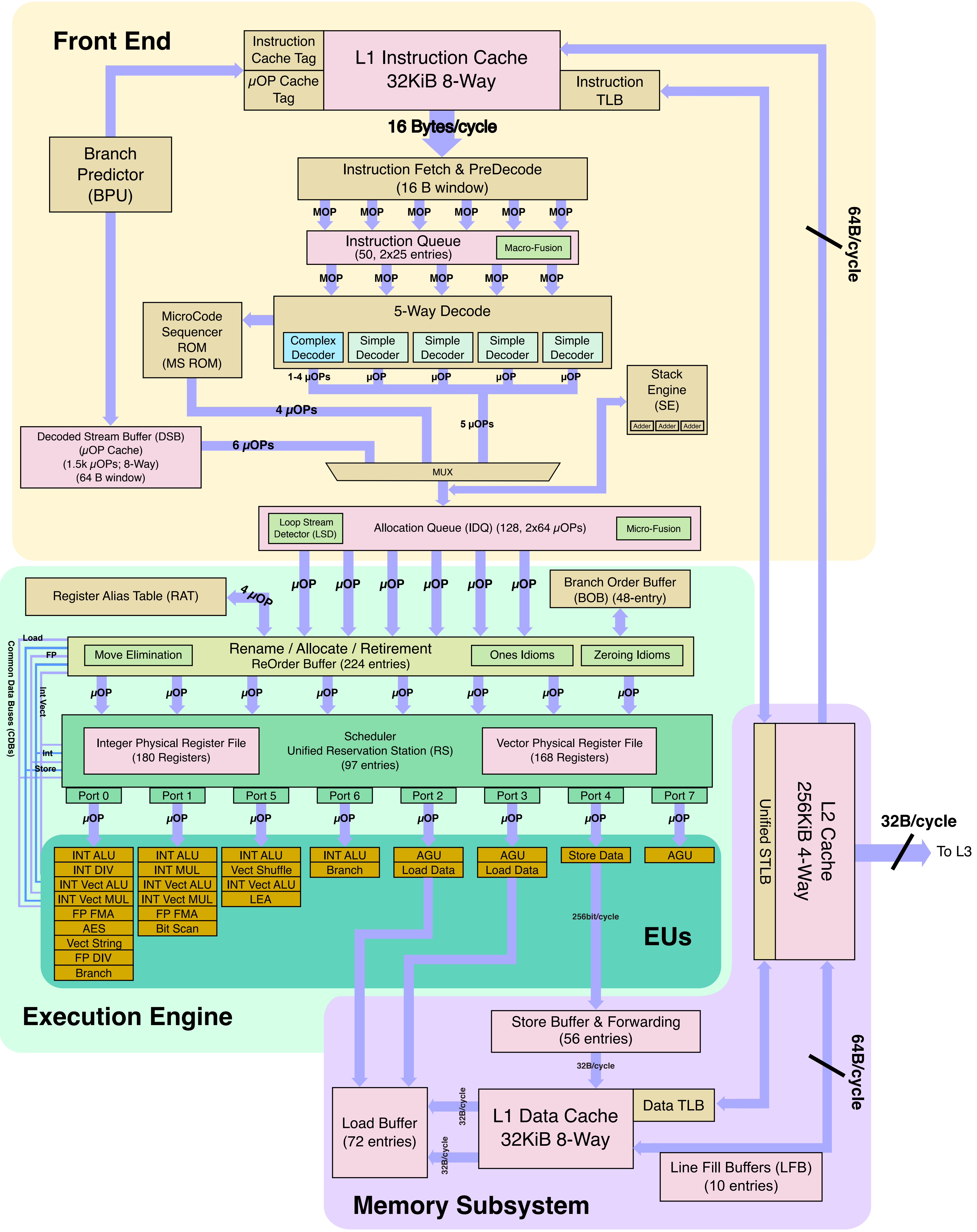
Caching

Key Issues in Caching

- Locality
  - *temporal (reuse within short amount of time)*
  - *spatial (use shortly physically nearby data)*
- Replacement algorithm
  - *LRU, Time-aware LRU in CDNs, Least-frequent LRU in CDNs, MRU in scanning big data*
- Write-through vs. write-back
- Working set
- Direct-mapped vs. partially associative vs. fully associative
- Size vs. speed trade-off

Intel Skylake Microarchitecture

Gen9
Core
Core
L3$ Slice
L3$ Slice
L3$ Slice
L3$ Slice
Ring
Core
Core
System Agent
Display Controller
PCIe
eDRAM Controller
(optional)
Memory Controller


Front End
Instruction Cache Tag
μOP Cache Tag
L1 Instruction Cache 32KiB 8-Way
Instruction TLB
16 Bytes/cycle
Branch Predictor (BPU)
Instruction Fetch & PreDecode (16 B window)
MOP
MOP
MOP
MOP
MOP
MOP
Instruction Queue (50, 2x25 entries)
Macro-Fusion
MOP
MOP
MOP
MOP
MOP
MicroCode Sequencer ROM (MS ROM)
5-Way Decode
Complex Decoder
Simple Decoder
Simple Decoder
Simple Decoder
Simple Decoder
1-4 μOPs
μOP
μOP
μOP
μOP
Stack Engine (SE)
Adder
Adder
Adder
4 μOPs
5 μOPs
Decoded Stream Buffer (DSB) (μOP Cache) (1.5k μOPs; 8-Way) (64 B window)
6 μOPs
MUX
Loop Stream Detector (LSD)
Allocation Queue (IDQ) (128, 2x64 μOPs)
Micro-Fusion
64B/cycle
Register Alias Table (RAT)
4 μOP
μOP
μOP
μOP
μOP
μOP
μOP
Branch Order Buffer (BOB) (48-entry)
Common Data Buses (CDBs)
Load
FP
Int Vect
Int
Store
Move Elimination
Rename / Allocate / Retirement ReOrder Buffer (224 entries)
Ones Idioms
Zeroing Idioms
μOP
μOP
μOP
μOP
μOP
μOP
μOP
μOP
Integer Physical Register File (180 Registers)
Scheduler Unified Reservation Station (RS) (97 entries)
Vector Physical Register File (168 Registers)
Port 0
Port 1
Port 5
Port 6
Port 2
Port 3
Port 4
Port 7
μOP
μOP
μOP
μOP
μOP
μOP
μOP
μOP
INT ALU
INT DIV
INT Vect ALU
INT Vect MUL
FP FMA
AES
Vect String
FP DIV
Branch
INT ALU
INT MUL
INT Vect ALU
INT Vect MUL
FP FMA
Bit Scan
INT ALU
Vect Shuffle
INT Vect ALU
LEA
INT ALU
Branch
AGU
Load Data
AGU
Load Data
Store Data
AGU
256bit/cycle
EUs
Execution Engine
Unified STLB
L2 Cache 256KiB 4-Way
32B/cycle
To L3
Store Buffer & Forwarding (56 entries)
32B/cycle
64B/cycle
Load Buffer (72 entries)
32B/cycle
32B/cycle
L1 Data Cache 32KiB 8-Way
Data TLB
Line Fill Buffers (LFB) (10 entries)
Memory Subsystem

Front End

Instruction Cache Tag

μOP Cache Tag

L1 Instruction Cache
32KiB 8-Way

Instruction TLB

16 Bytes/cycle

Branch Predictor (BPU)

Instruction Fetch & PreDecode
(16 B window)

MOP MOP MOP MOP MOP MOP

Instruction Queue
(50, 2x25 entries)

Macro-Fusion

MOP MOP MOP MOP MOP

MicroCode Sequencer ROM (MS ROM)

5-Way Decode

Complex Decoder

Simple Decoder

Simple Decoder

Simple Decoder

Simple Decoder

1-4 μOPs μOP μOP μOP μOP

4 μOPs

5 μOPs

Stack Engine (SE)

Adder Adder Adder

Decoded Stream Buffer (DSB)
(μOP Cache)
(1.5k μOPs; 8-Way)
(64 B window)

6 μOPs

MUX

Loop Stream Detector (LSD)

Allocation Queue (IDQ) (128, 2x64 μOPs)

Micro-Fusion

μOP μOP μOP μOP μOP

Branch Order Buffer (BOB) (48-entry)

ate / Retirement
r (224 entries)

Ones Idioms

Zeroing Idioms

P μOP μOP μOP μOP

Scheduler
Reservation Station (RS)
(97 entries)

Vector Physical Register File
(168 Registers)

Port 6 Port 2 Port 3 Port 4 Port 7

μOP μOP μOP μOP μOP

NT ALU
Branch

AGU
Load Data

AGU
Load Data

Store Data

AGU

FP FMA
AES
Vect String
FP DIV
Branch

FP FMA
Bit Scan

256bit/cycle

EUs

Execution Engine

Unified STLB

L2 Cache
256KiB 4-Way

32B/cycle

L3 cache

RAM

64B/cycle

Store Buffer & Forwarding
(56 entries)

32B/cycle

Load Buffer
(72 entries)

32B/cycle

32B/cycle

L1 Data Cache
32KiB 8-Way

Data TLB

Line Fill Buffers (LFB)
(10 entries)

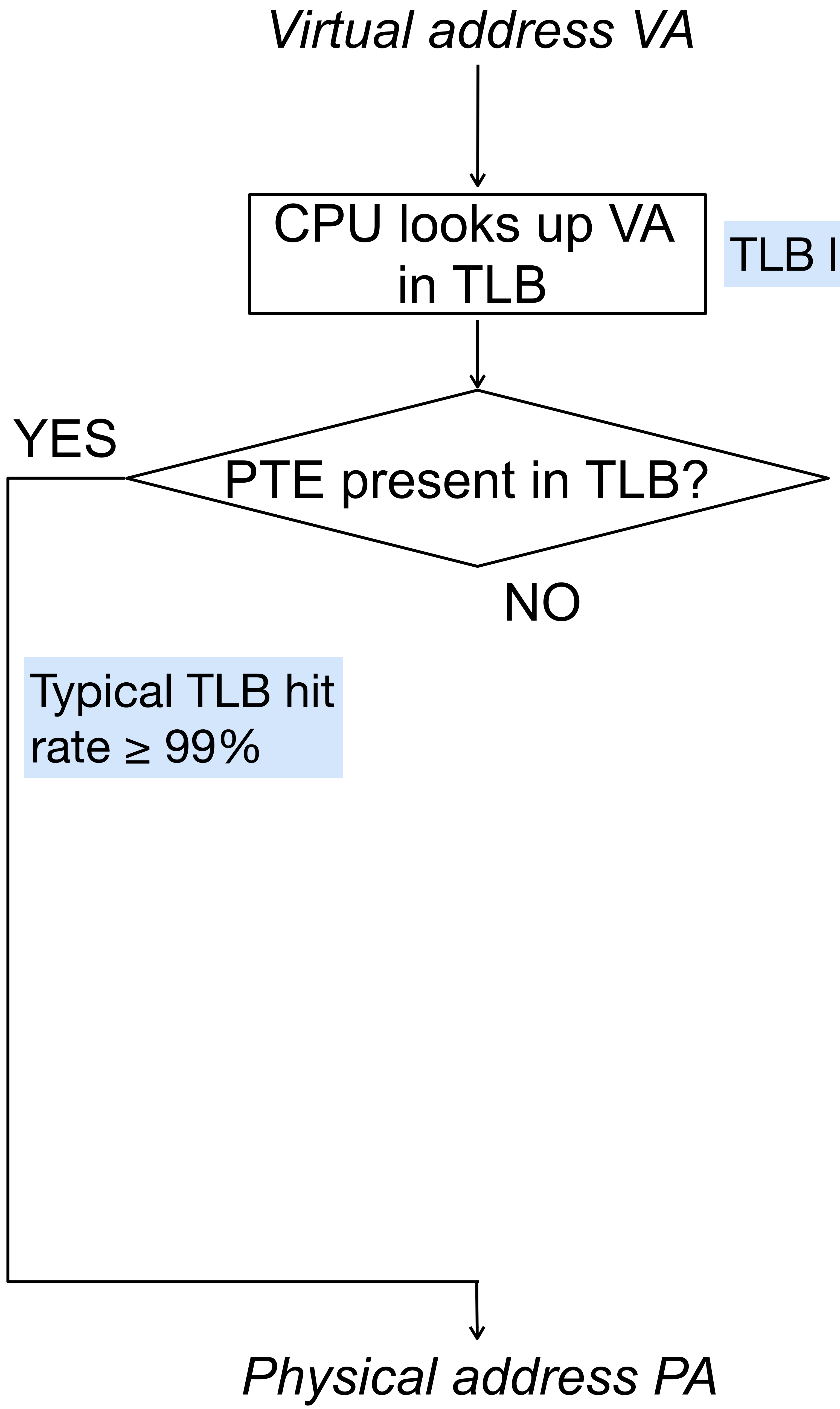
Memory Subsystem

Front End
Instruction Cache Tag
μOP Cache Tag
L1 Instruction Cache
32KiB 8-Way
Instruction TLB
16 Bytes/cycle
Branch Predictor (BPU)
Instruction Fetch & PreDecode
(16 B window)
MOP
Instruction Queue
(50, 2x25 entries)
Macro-Fusion
5-Way Decode
Complex Decoder
Simple Decoder
MicroCode Sequencer ROM (MS ROM)
1-4 μOPs
μOP
4 μOPs
5 μOPs
6 μOPs
Stack Engine (SE)
Adder
Decoded Stream Buffer (DSB)
(μOP Cache)
(1.5k μOPs; 8-Way)
(64 B window)
MUX
Loop Stream Detector (LSD)
Allocation Queue (IDQ) (128, 2x64 μOPs)
Micro-Fusion
Branch Order Buffer
(BOB) (48-entry)
ate / Retirement
r (224 entries)
Ones Idioms
Zeroing Idioms
Scheduler
Reservation Station (RS)
(97 entries)
Vector Physical Register File
(168 Registers)
Port 6
Port 2
Port 3
Port 4
Port 7
NT ALU
Branch
AGU
Load Data
Store Data
AGU
FP FMA
AES
Vect String
FP DIV
Branch
FP FMA
Bit Scan
256bit/cycle
EUs
Execution Engine
Unified STLB
L2 Cache
256KiB 4-Way
32B/cycle
L3 cache
RAM
64B/cycle
Store Buffer & Forwarding
(56 entries)
32B/cycle
Load Buffer
(72 entries)
L1 Data Cache
32KiB 8-Way
Data TLB
Line Fill Buffers (LFB)
(10 entries)
Memory Subsystem

Cache the translations of names using TLB

Split TLB to exploit access locality and on-chip location

Technology trade-offs: speed vs. cost



Virtual address VA

CPU looks up VA in TLB

PTE present in TLB?

YES

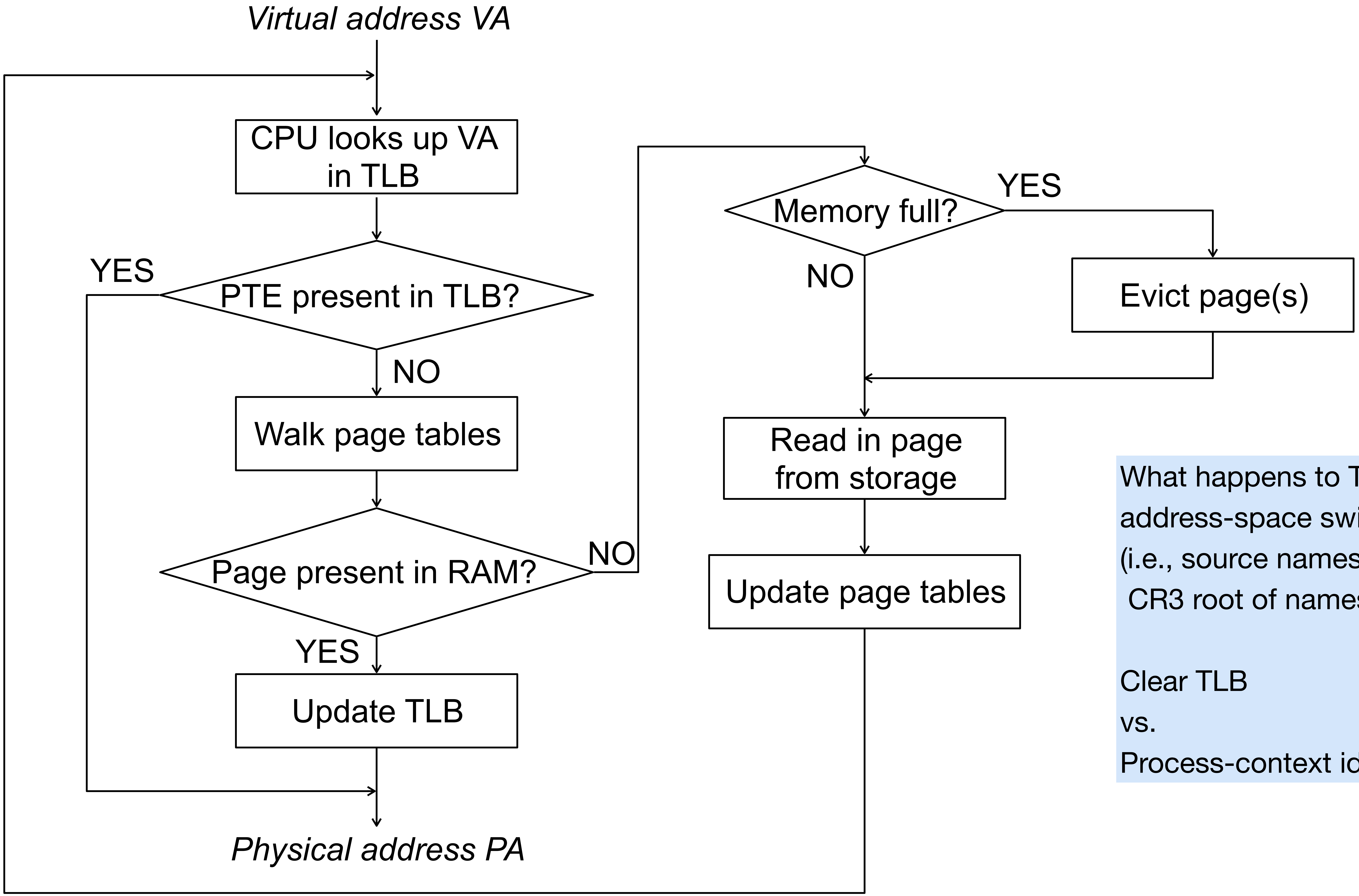
NO

Physical address PA

TLB lookup ≤ 1 cycle

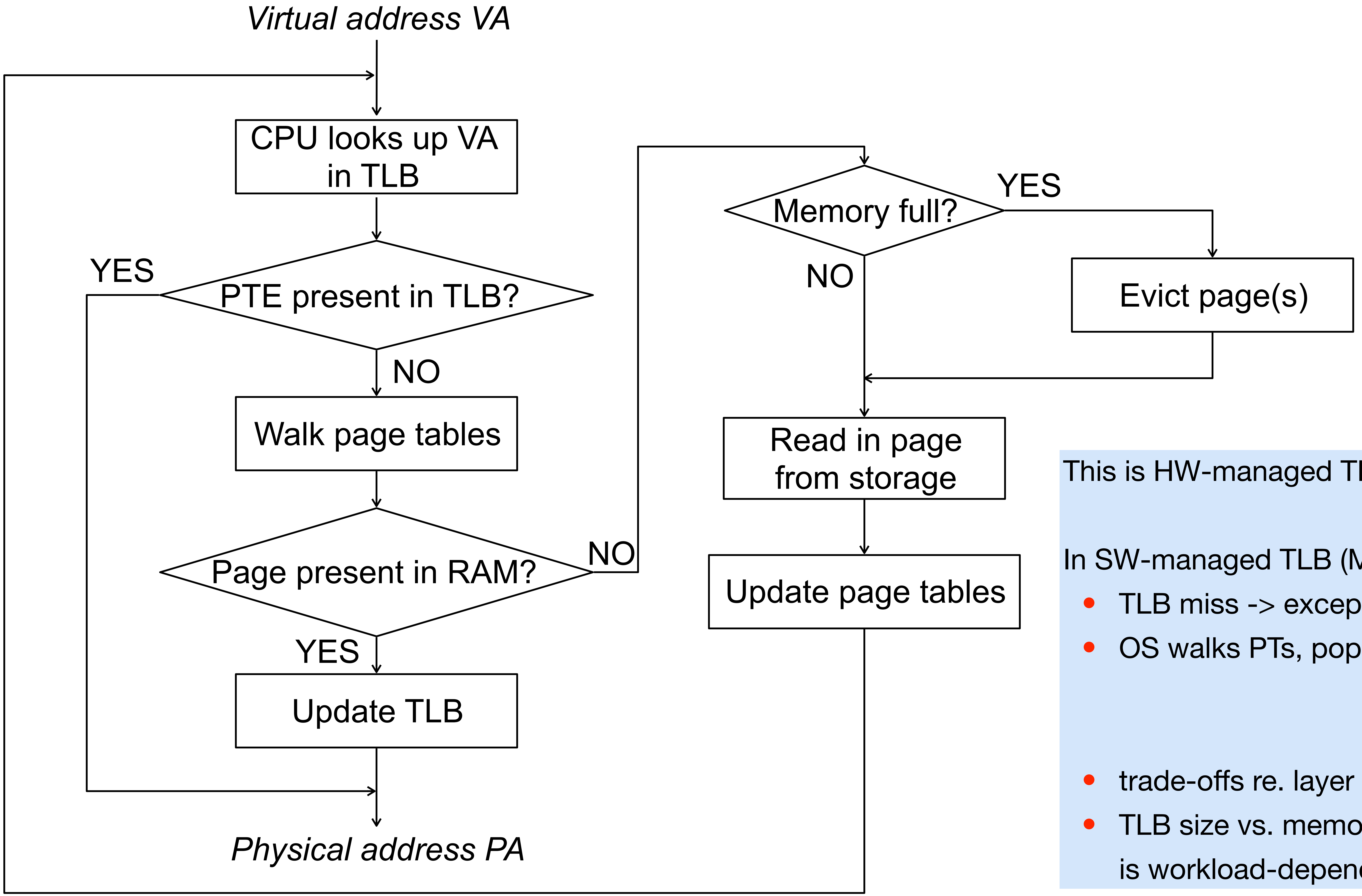
Typical TLB hit rate ≥ 99%

Read from L1 = 3-4 cycles
Read from L2 = 10-12 cycles
Read from L3 = 30-80 cycles
Read from RAM = 100-150 cycles



What happens to TLB on an address-space switch?
(i.e., source namespace differs => CR3 root of namespace changes)

Clear TLB
vs.
Process-context identifiers



This is HW-managed TLB !

In SW-managed TLB (MIPS, SPARC, …)

- TLB miss -> exception to OS
- OS walks PTs, populates TLB

- trade-offs re. layer to delegate to
- TLB size vs. memory size trade-off is workload-dependent => hard!

Indexing and Tagging in Caches

- Index into cache → check the tag
- Types of caches
  - *PIPT (Physically indexed / Physically tagged)*
  - *VIVT (Virtually indexed / Virtually tagged)*
  - *VIPT (Virtually indexed / Physically tagged)*
  - ~~*PIVT (Physically indexed / Virtually tagged)*~~
- Caches on x86
  - *L1 is VIPT*
  - *L2 and L3 are PIPT*

Constants in System Design

Systems “Constants” (CPU cycles)

- Register-register ADD/OR/etc. < 1 CPU cycle
- Memory write ~1 cycle
- Correctly / incorrectly predicted "if" branch = 1 cycle / 10-20 cycles
- L1 / L2 / L3 / main RAM read = 3-4 cycles / 10-12 cycles / 30-80 cycles / 100-150 cycles
- TLB hit / miss = 0.5 - 1 cycle / 7-21 cycles
- CAS = 15-30 cycles
- C function direct / indirect call = 15-30 cycles / 20-50 cycles
- Kernel syscall = 1,000 - 1,500 cycles
- Thread context switch (direct costs) = 2,000 cycles
- On NUMA, different-socket mem hierarchy access is 3 - 10x that of non-NUMA

Systems “Constants" (absolute time)

- L1 / L2 / main RAM reference = 1 / 4 / 100 ns
- Mutex lock or unlock = 17 ns
- Branch misprediction = 3 ns
- Send 2KB (2,000 bytes) over commodity network = 88 ns
- Compress 1 KB with Snappy = 2,000 ns = 2 microsec
- SSD random read = 16,000 ns = 16 microsec
- Read 1 MB sequentially from RAM = 5,000 ns = 5 microsec
- Packet roundtrip in same datacenter < 50 microsec
- Read 1 MB sequentially from SSD = 78 microsec
- Seek on magnetic disk = 3 millisec
- Read 1 MB sequentially from magnetic disk = 1 millisec
- Packet roundtrip CA -> Amsterdam -> CA = 150 millisec