CS330: Operating Systems

Virtual memory: Paging

Recap: Segmentation

0

Code

1KB

Heap

3KB

Stack

7KB

8KB

Address space

Code segment

Base = 28KB
Limit = 1KB

Data segment

Base = 20KB
Limit = 2KB

Stack segment

Base = 32KB
Limit = 1KB

0

20KB

22KB

28KB

29KB

31KB

32KB

48KB

RAM
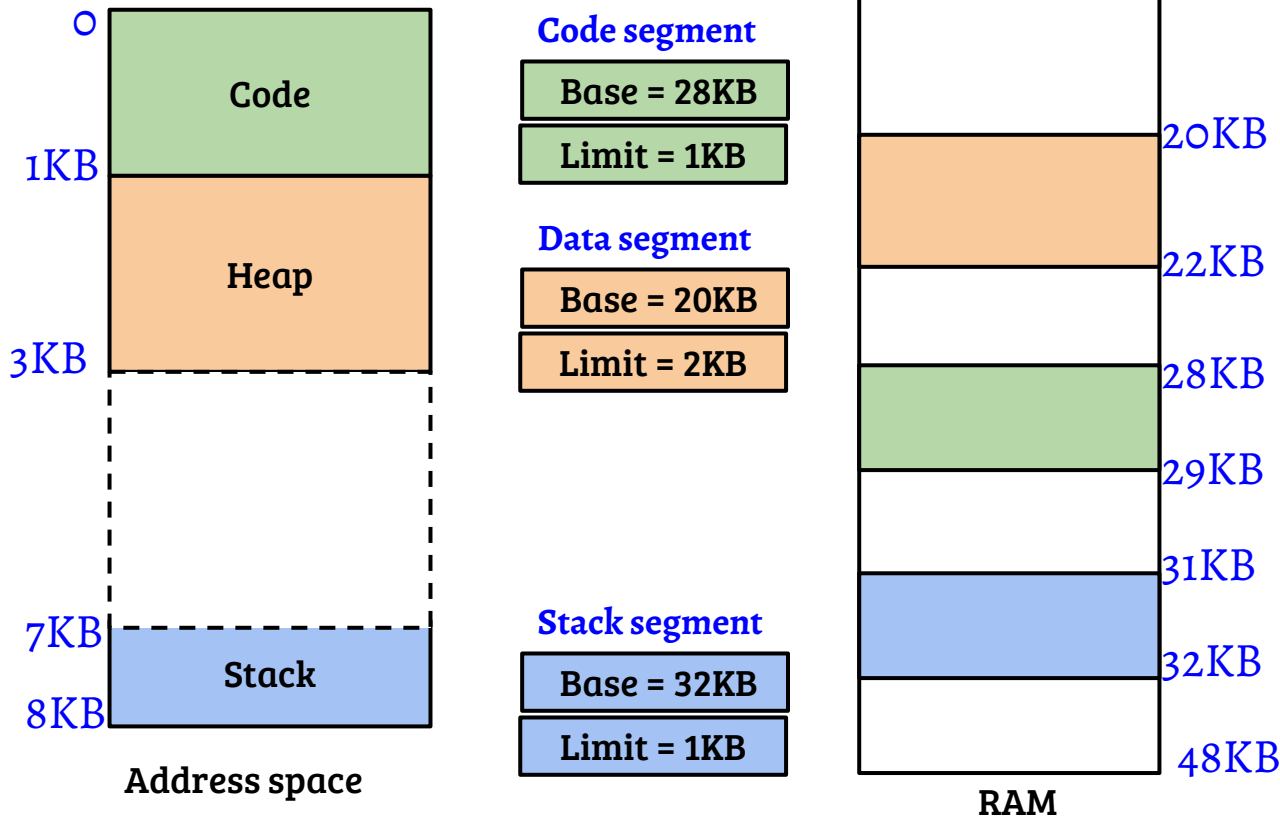
- Extension of the scheme for translation ar address space granularity
- Base-limit register pairs per segment

Recap: Segmentation in reality

Descriptor Table

Flags

D — Direction (+ or -)

S — Privilege

R W X — Read Write Execute
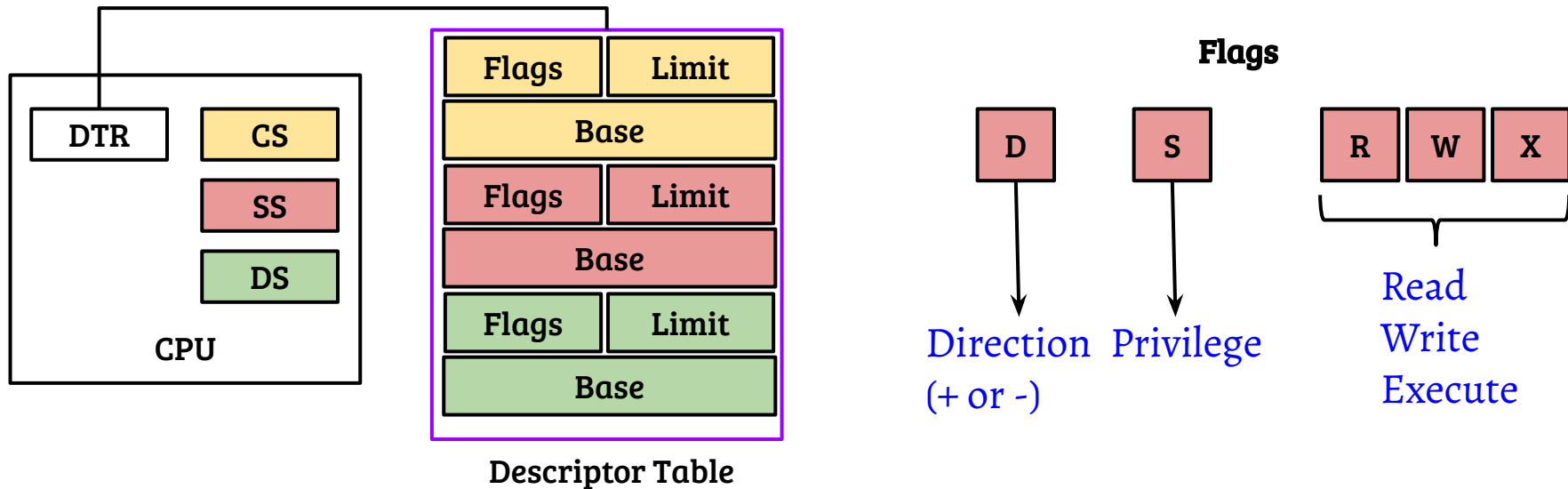
- Descriptor table register (DTR) is used to access the descriptor table
- # of descriptors depends on architecture
- Separate descriptors used for user and kernel mode

Paging

- Paging addresses the following issues with segmentation
    - External fragmentation caused due to variable sized segments
    - No support for discontinuous/sparse mapping
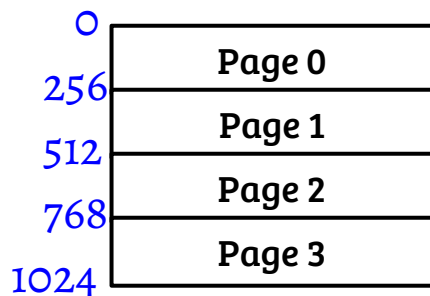
Paging

- Paging addresses the following issues with segmentation
    - External fragmentation caused due to variable sized segments
    - No support for discontinuous/sparse mapping
- The idea of paging
    - Partition the address space into fixed sized blocks (call it page)
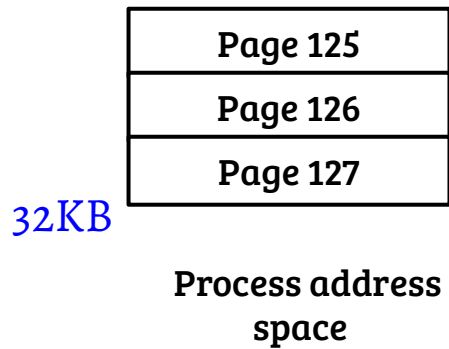   - Physical memory partitioned in a similar way (call it page frame)

Paging

- Paging addresses the following issues with segmentation
    - External fragmentation caused due to variable sized segments
    - No support for discontinuous/sparse mapping
- The idea of paging
    - Partition the address space into fixed sized blocks (call it pages)
    - Physical memory partitioned in a similar way (call it page frames)
    - OS creates a mapping between *page* to *page frame*
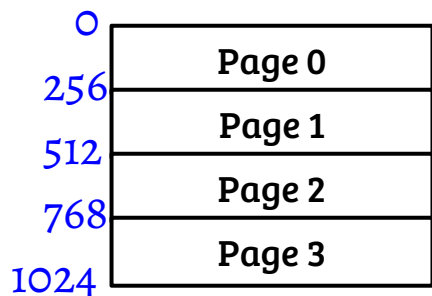   - H/W uses the mapping to translate VA to PA

Paging example (pages)

0	
	Page 0
256	
	Page 1
512	
	Page 2
768	
	Page 3
1024	

Page 125
Page 126
Page 127

32KB

Process address space

- Virtual address size = 32KB, Page size = 256 bytes

- Address length = 15 bits {0x0 - 0x7FFF}

- # of pages = 128

Paging example (pages)

Process address space

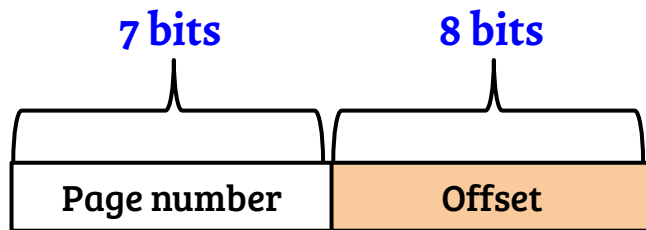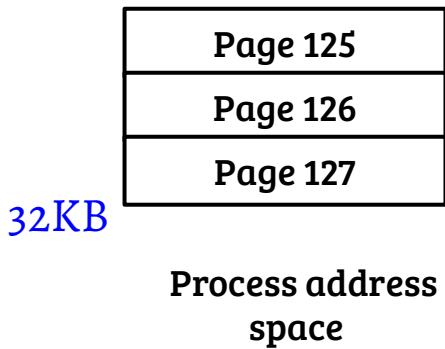- Virtual address size = 32KB, Page size = 256 bytes

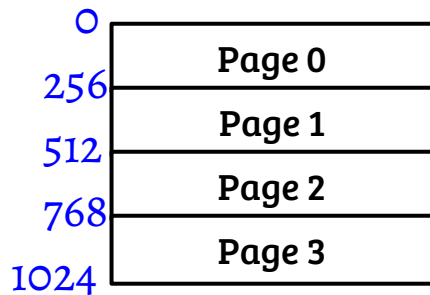- Address length = 15 bits {0x0 - 0x7FFF}

- # of pages = 128
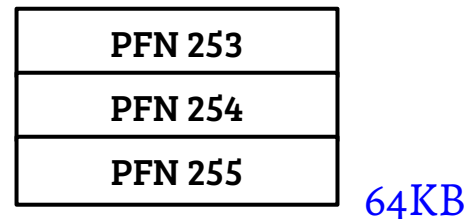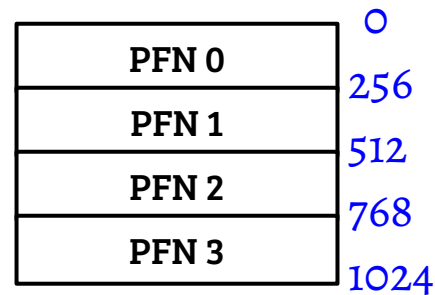

Virtual address

- Example: For Virtual address *0x0510*, Page number = 5, offset = 16

Paging example (page frames)

- Physical address size = 64KB

- Address length = 16 bits {0x0 - 0xFFFF}

- # of page frames = 256

Process address space:

```
0      ┌──────────────┐
       │    Page 0    │
256    ├──────────────┤
       │    Page 1    │
512    ├──────────────┤
       │    Page 2    │
768    ├──────────────┤
       │    Page 3    │
1024   └──────────────┘
```

```
       ┌──────────────┐
       │   Page 125   │
       ├──────────────┤
       │   Page 126   │
       ├──────────────┤
       │   Page 127   │
32KB   └──────────────┘
```

Process address space

DRAM:

```
0      ┌──────────────┐
       │    PFN 0     │
256    ├──────────────┤
       │    PFN 1     │
512    ├──────────────┤
       │    PFN 2     │
768    ├──────────────┤
       │    PFN 3     │
1024   └──────────────┘
```

```
       ┌──────────────┐
       │   PFN 253    │
       ├──────────────┤
       │   PFN 254    │
       ├──────────────┤
       │   PFN 255    │
       └──────────────┘
64KB
```
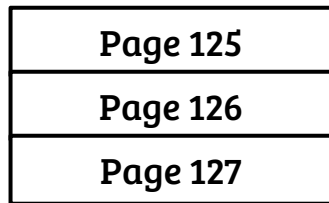
DRAM

Paging example (page frames)

- Physical address size = 64KB
- Address length = 16 bits {0x0 - 0xFFFF}
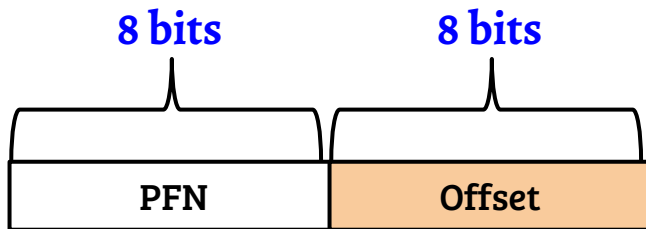- # of page frames = 256

Process address space

0	Page 0
256	Page 1
512	Page 2
768	Page 3
1024	

	Page 125
	Page 126
	Page 127
32KB	

DRAM

0	PFN 0
256	PFN 1
512	PFN 2
768	PFN 3
1024	

	PFN 253
	PFN 254
	PFN 255
64KB	

8 bits   **8 bits**

PFN	Offset

Physical address

- Example: For physical address *0x1F51*, PFN = 31, offset = 81

Paging example (page table mapping)

Process address space

0	Page 0
256	Page 1
512	Page 2
768	Page 3
1024	

Page 125
Page 126
Page 127

32KB

Page table

1
-
2
4
-

128 entries

-
3

DRAM

PFN 0	0
PFN 1	256
PFN 2	512
PFN 3	768
PFN 4	1024
	1280

PFN 253
PFN 254
PFN 255

64KB
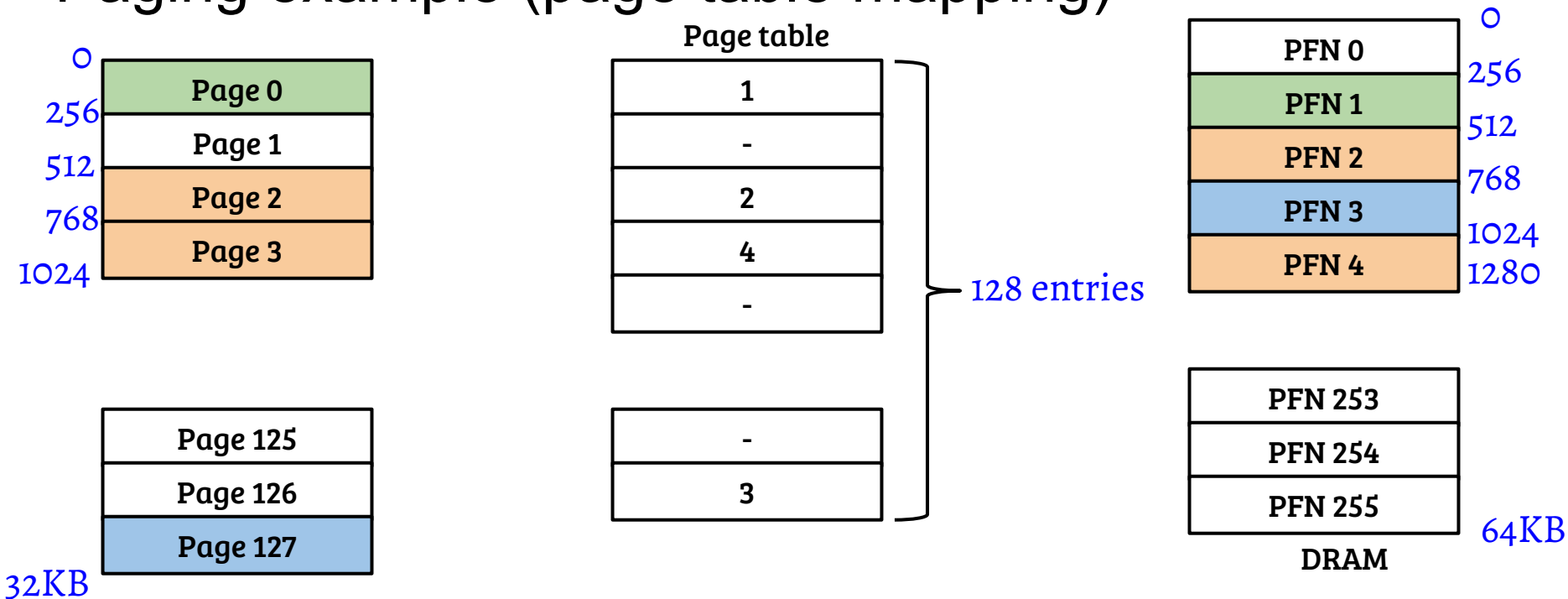
- Each entry in page table is called page table entry (PTE)
- Example mapping: page 0 ⇒ PFN 1, page 2 ⇒ PFN 2 and so on

Paging example (page table walk)
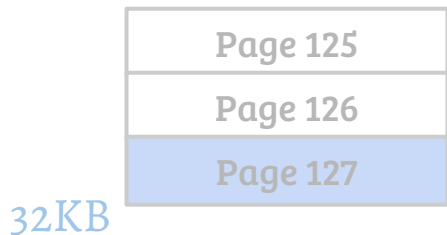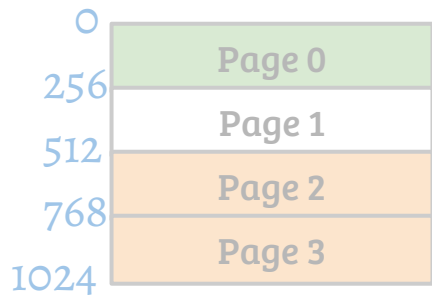
```
PTW (vaddr V, PTable P)
// Input: Virtual address, Page table
// Returns physical address
{
    Entry = P[V >> 8];
    if (Entry.present)
        return (Entry.PFN << 8) + (V & 0xFF);
    Raise PageFault;
}
```
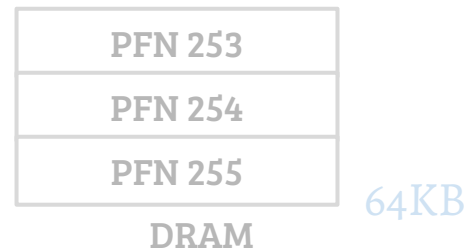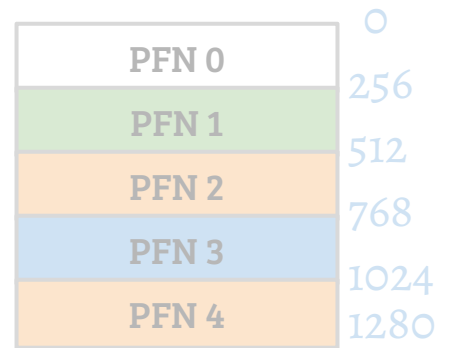
Process address space

0	Page 0
256	Page 1
512	Page 2
768	Page 3
1024	

	Page 125
	Page 126
	Page 127
32KB	

0	PFN 0
256	PFN 1
512	PFN 2
768	PFN 3
1024	PFN 4
1280	

	PFN 253
	PFN 254
	PFN 255
DRAM	64KB

Paging example (example translation)
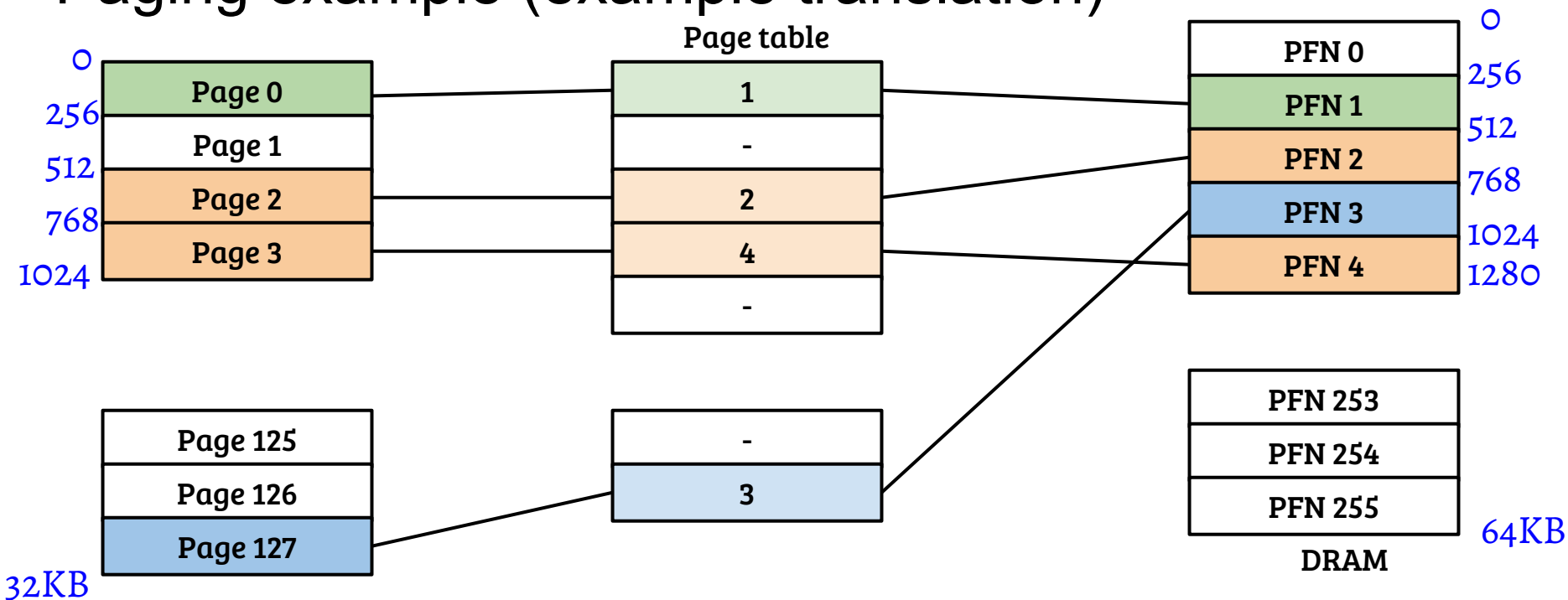
- Virtual address 0x10 translates to physical address 0x110
- Virtual address 0x7FF0 translates to physical address 0x3F0

Paging example (page table walk)

Page table

0	Page 0
256	

1

PFN 0	0
	256

- Where is the page table stored?
- What is the structure of the PTE?
- What is the maximum physical memory size supported?

Page 125
Page 126
Page 127

3

PFN 253
PFN 254
PFN 255

32KB

64KB

DRAM

Process address space

Paging example (page table walk)

Page table

Page 0	1

PFN 0

0 256 256

- Where is the page table stored?
- Page table is stored in RAM. Page table base register (CR3 in X86) contains the address
- What is the structure of the PTE?
- What is the maximum physical memory size supported?

Page 127

PFN 255

32KB 64KB

DRAM

Process address space

Paging example (structure of an example PTE)

8 bits

PFN			X	D	A	S	W	P

- PFN occupies a significant portion of PTE entry (8 bits in this example)

- **P** — Present bit, 1 ⇒ entry is valid

- **W** — Write bit, 1 ⇒ Write allowed

- **S** — Privilege bit, 0 ⇒ only kernel mode access is allowed

- **A** — Accessed bit, 1 ⇒ Address accessed (set by H/W during walk)

- **D** — Dirty bit, 1 ⇒ Address written (set by H/W during walk)

- **X** — Execute bit, 1 ⇒ Instruction fetch allowed for this page

- Reserved/unused bits

Paging example (Page table entries)

Page table

Process address space	Page table	DRAM
0		0
Page 0	0x125	PFN 0
256		256
Page 1	0x0	PFN 1
512		512
Page 2	0x207	PFN 2
768		768
Page 3	0x407	PFN 3
1024	0x0	1024
		PFN 4
		1280

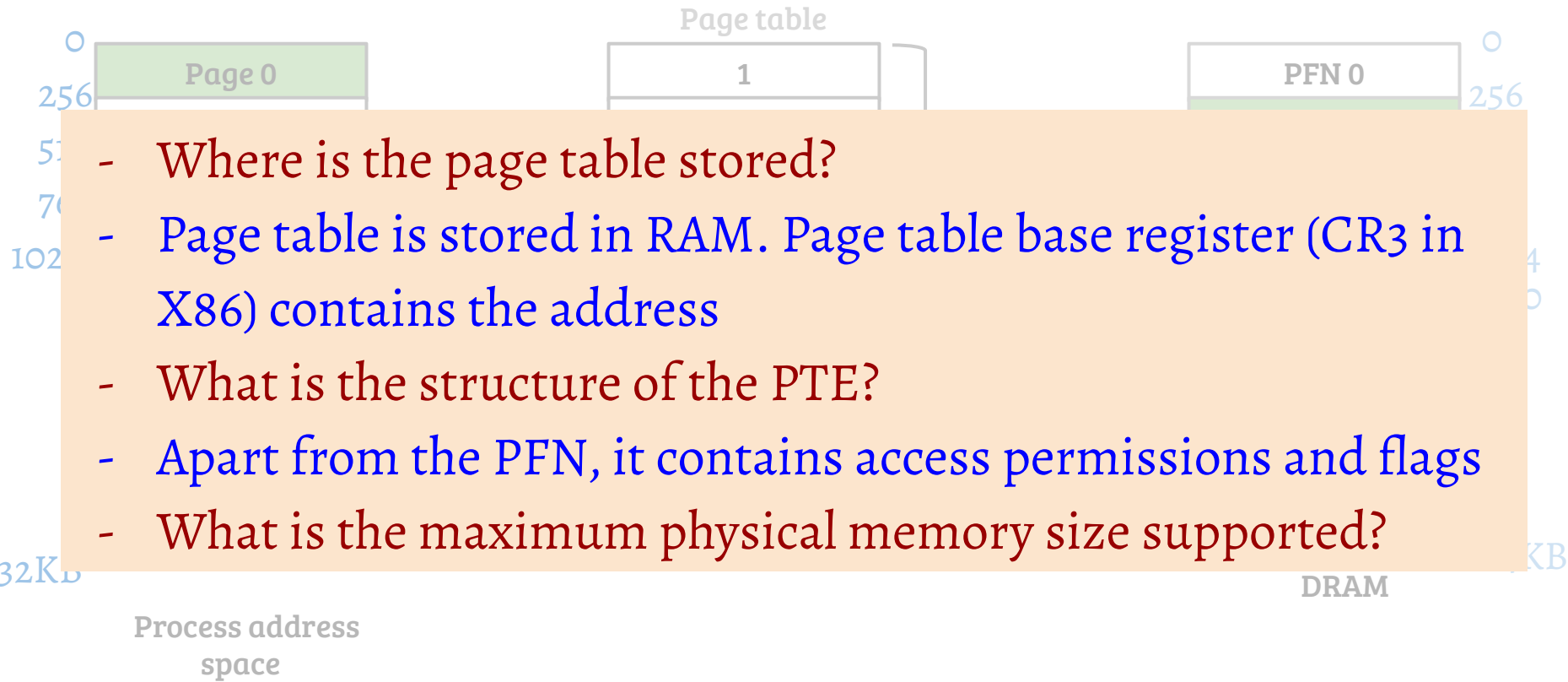|		
Page 125	0x0	PFN 253
Page 126	0x307	PFN 254
Page 127		PFN 255
32KB		64KB

Process address space

DRAM

- Code: Page 0 (Read and Execute)
- Data: Page 2 and Page 3 (Read and Write)
- Stack: Page 127 (Read and Write)

Paging example (page table walk)

Page table

Page 0	

	1

PFN 0	

0   256

256   512

- **Where is the page table stored?**
- Page table is stored in RAM. Page table base register (CR3 in X86) contains the address
- **What is the structure of the PTE?**
- Apart from the PFN, it contains access permissions and flags
- **What is the maximum physical memory size supported?**

32KB

Process address
space

DRAM

Paging example (page table walk)

- Where is the page table stored?
- Page table is stored in RAM. Page table base register (CR3 in X86) contains the address
- What is the structure of the PTE?
- Apart from the PFN, it contains access permissions and flags
- What is the maximum physical memory size supported?
- For this example, 8-bits can be used to specify 256 page frames. Maximum RAM size = 256 * 256 = 64KB

Paging: one level of page table may not be feasible!

- Consider a 32-bit address space (=4GB)
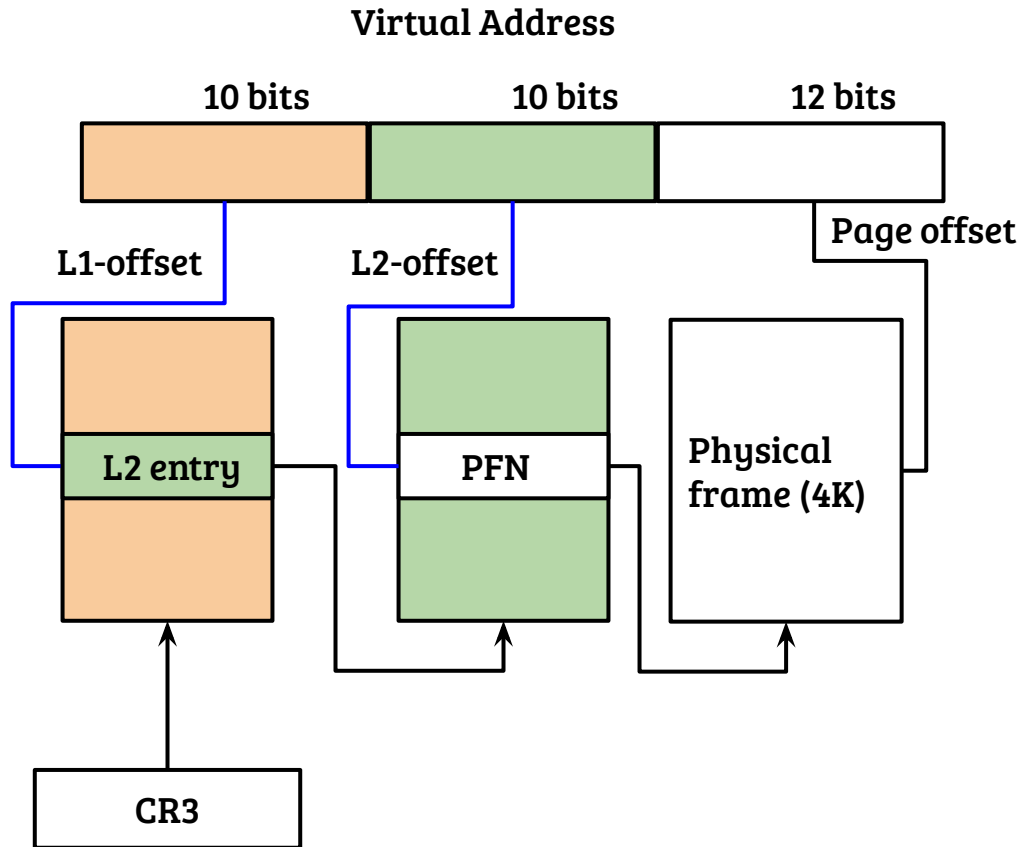- What should be the page size for this system?

Paging: one level of page table may not be feasible!

- Consider a 32-bit address space (=4GB)
- What should be the page size for this system?
- Large page size results in *internal fragmentation*
- Assuming page size = 4KB, How many entries are required in a one-level paging system?

Paging: one level of page table may not be feasible!

- Consider a 32-bit address space (=4GB)
- What should be the page size for this system?
- Large page size results in *internal fragmentation*
- Assuming page size = 4KB, How many entries are required in a one-level paging system? (2^{20} entries)
- Not possible to hold 2^{20} entries in a single page
- Therefore, multi-level page tables are used in modern systems

Two-level page tables (32-bit virtual address)

Virtual Address

10 bits	10 bits	12 bits

L1-offset   L2-offset   Page offset

L2 entry

PFN

Physical frame (4K)

CR3

- Two-level page table
- Level-1 page table contains entries pointing to Level-2 page table structures
- Level-2 entry contains PFN along with flags