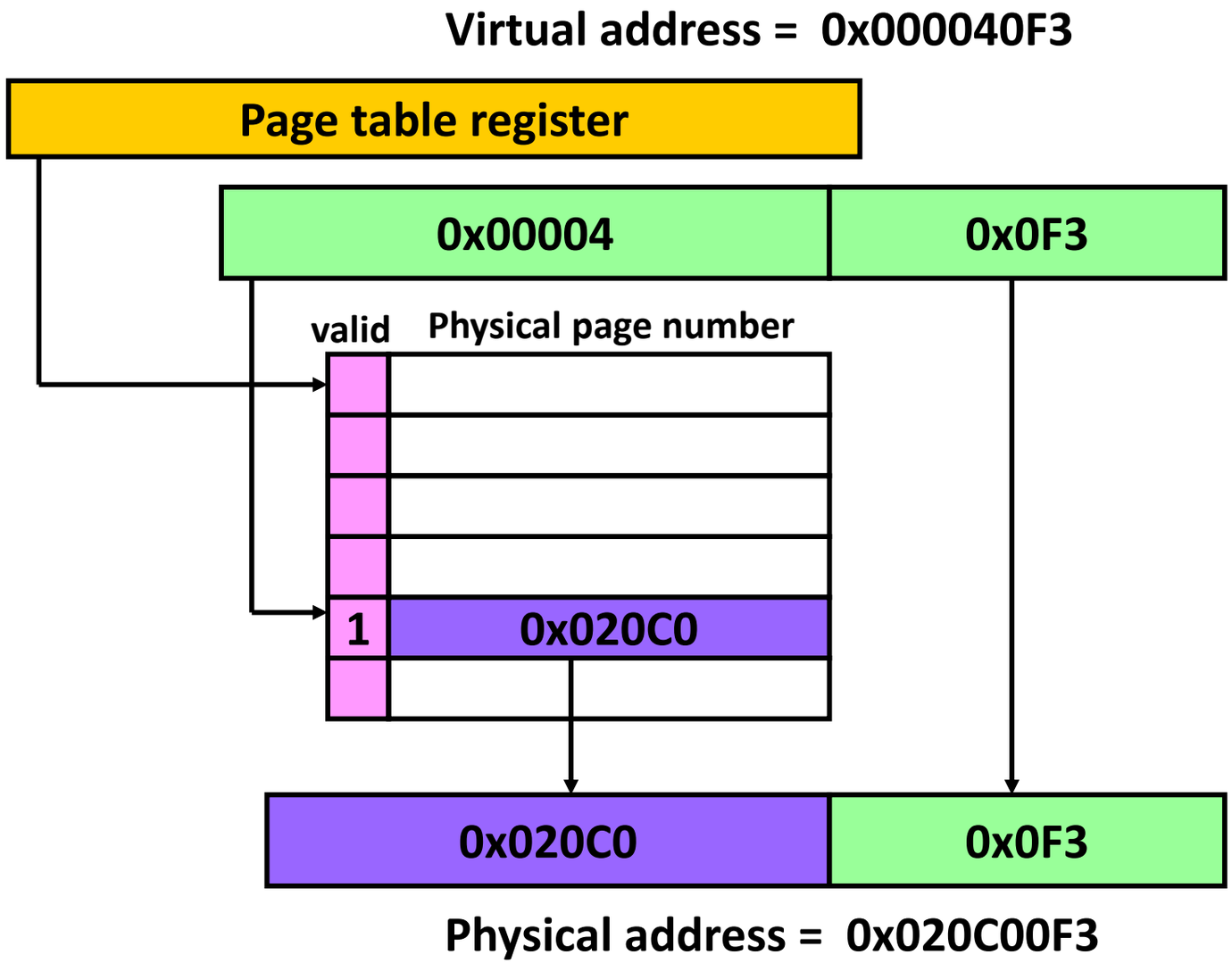
EECS 370

Multi-Level Page Tables

Announcements

- Lab
  - Assignment due tonight
  - Pre-lab quiz due tomorrow
- Project 4
  - Out now, due after Thanksgiving
- Final exam
  - Previous exams posted

Reminder: Page tables



Class Problem

- Given the following:
  - 4KB page size, physical memory of 16KB, page table stored in physical page 0 and can never be evicted, 20 bit, byte-addressable virtual address space.
  - The page table initially has virtual page 0 in physical page 1, virtual page 1 in physical page 2 and no valid data in other physical pages.
- Fill in the table on the next slide for each reference
  - Note: like caches we'll use LRU when we need to replace a page.

Class Problem (continued)

4KB page size, physical memory of 16KB, page table stored in physical page 0 and can never be evicted, 20 bit, byte-addressable virtual address space.

The page table initially has virtual page 0 in physical page 1, virtual page 1 in physical page 2 and no valid data in other physical pages.

Virt addr	Virt page	Page fault?	Phys addr
0x00F0C			
0x01F0C			
0x20F0C			
0x00100			
0x00200			
0x30000			
0x01FFF			
0x00200			

Poll: How many hex digits should the page number be?

Class Problem (continued)

Virt addr	Virt page	Page fault?	Phys addr
0x00F0C	0x0	N	0x1F0C
0x01F0C	0x1	N	0x2F0C
0x20F0C	0x20	Y (into 3)	0x3F0C
0x00100	0x0	N	0x1100
0x00200	0x0	N	0x1200
0x30000	0x30	Y (into 2)	0x2000
0x01FFF	0x1	Y (into 3)	0x3FFF
0x00200	0x0	N	0x1200

4KB page size, physical memory of 16KB, page table stored in physical page 0 and can never be evicted, 20 bit, byte-addressable virtual address space.

The page table initially has virtual page 0 in physical page 1, virtual page 1 in physical page 2 and no valid data in other physical pages.

Performance of System

- Every memory access requires an additional access to memory
  - Page table must be read
- Can really slow down our system (potentially multiple accesses to slow DRAM)
- Solution: create a cache of previous translations
  - Called a "Translation-Lookaside Buffer" (TLB)
  - Will be discussed in lab 12 and lecture next week

Size of the page table

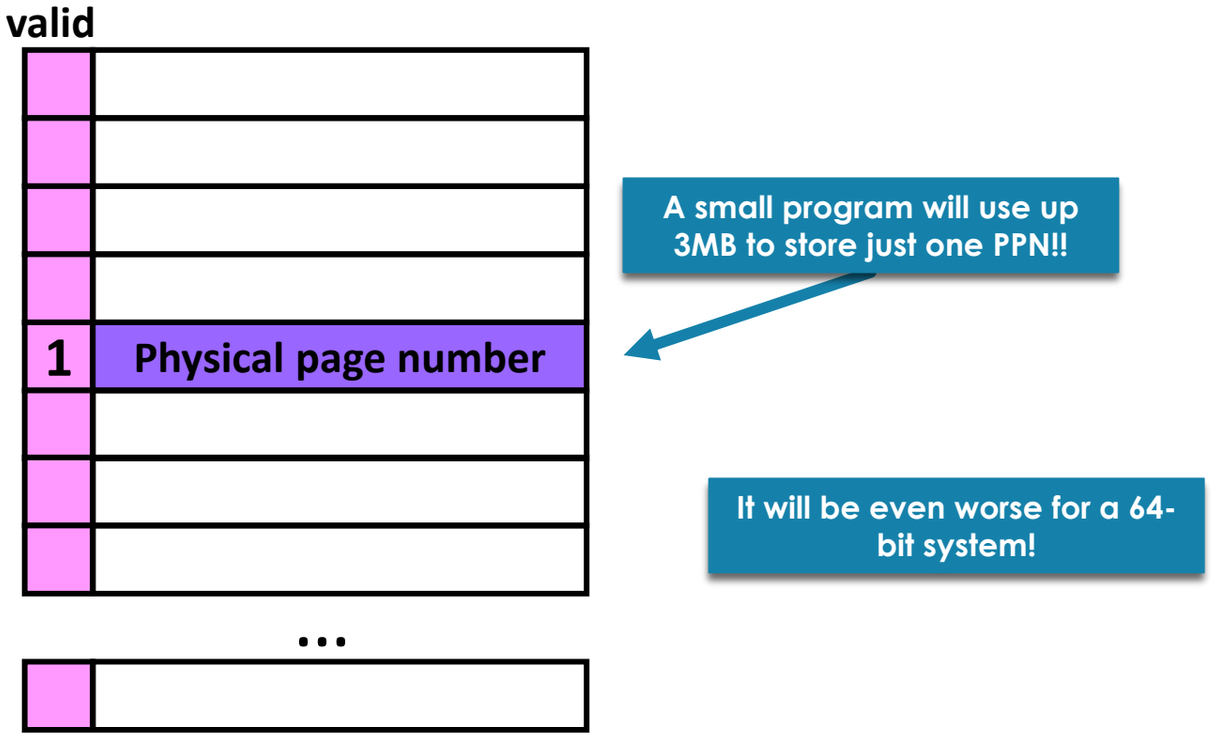
- How big is a page table entry?
  - For 32-bit virtual address:
    - If the machine can support 1GB = 2^{30} bytes of physical memory and we use pages of size 4KB = 2^{12},
    - then the physical page number is 30-12 = 18 bits.
      Plus another valid bit + other useful stuff (read only, dirty, etc.)
    - Let say about 3 bytes.
- How many entries in the page table?
  - 1 entry per virtual page
  - ARM virtual address is 32 bits – 12 bit page offset = 20
  - Total number of virtual pages = 2^{20}
- Total size of page table = Number of virtual pages
  * Size of each page table entry
  = 2^{20} x 3 bytes ~ 3 MB

Agenda

- **Motivation for Multi-level Page Tables**
- Example architecture
- Class Problem: Multi-Level VM Design
- VM Miscellanea

How can you organize the page table?

1. Single-level page table occupies continuous region in physical memory
   - Previous example always takes 3MB regardless of how much virtual memory is used



How can you organize the page table?

2. Option 2: Use a multi-level page table
   - Split up single-level page into multiple chunks
   - Only allocate space for that chunk if it's non-empty (i.e. program uses some of those physical pages)
   - Maintain a top-level page table which contains pointers to each of these chunks

valid	PPN
0	
0	
0	
0	
1	0x1
1	0x2
1	0x3
0	
0	
0	
0	

Single-level: Tons of wasted space!

valid	2nd level page table
0	
1	0x1000
0	
0	

valid	PPN
1	0x1
1	0x2
1	0x3
0	

Multi-level: No need to allocate 2nd level entries if all invalid

Multi-Level Page Table

- Only allocate second (and later) page tables when needed
- Program starts: everything is invalid, only first level is allocated

valid	2nd level page table
0	
0	
0	
0	

- As we access more, second level page tables are allocated

valid	2nd level page table
1	0x1000
1	0x3500

valid	PPN
1	0x1
1	0x6
1	0x2
1	0x1f

valid	PPN
1	0x9a
1	0x3
0	
1	0xff

Multi-level: Size is proportional to amount of memory used

Common case: most programs use small portion of virtual memory space

Multi-level page tables

- If program is using much of the virtual address space, multi-level page table could be about the same or larger size than single-level

valid	PPN
1	0x4
1	0x5
0	
0	
1	0x1
1	0x2
1	0x3
0	
0	
1	0x6
0	
0	
1	0x7

In this case, 2 level is no better

valid	2nd level page table
1	0x2000
1	0x1000
1	0x3000
1	0x4000

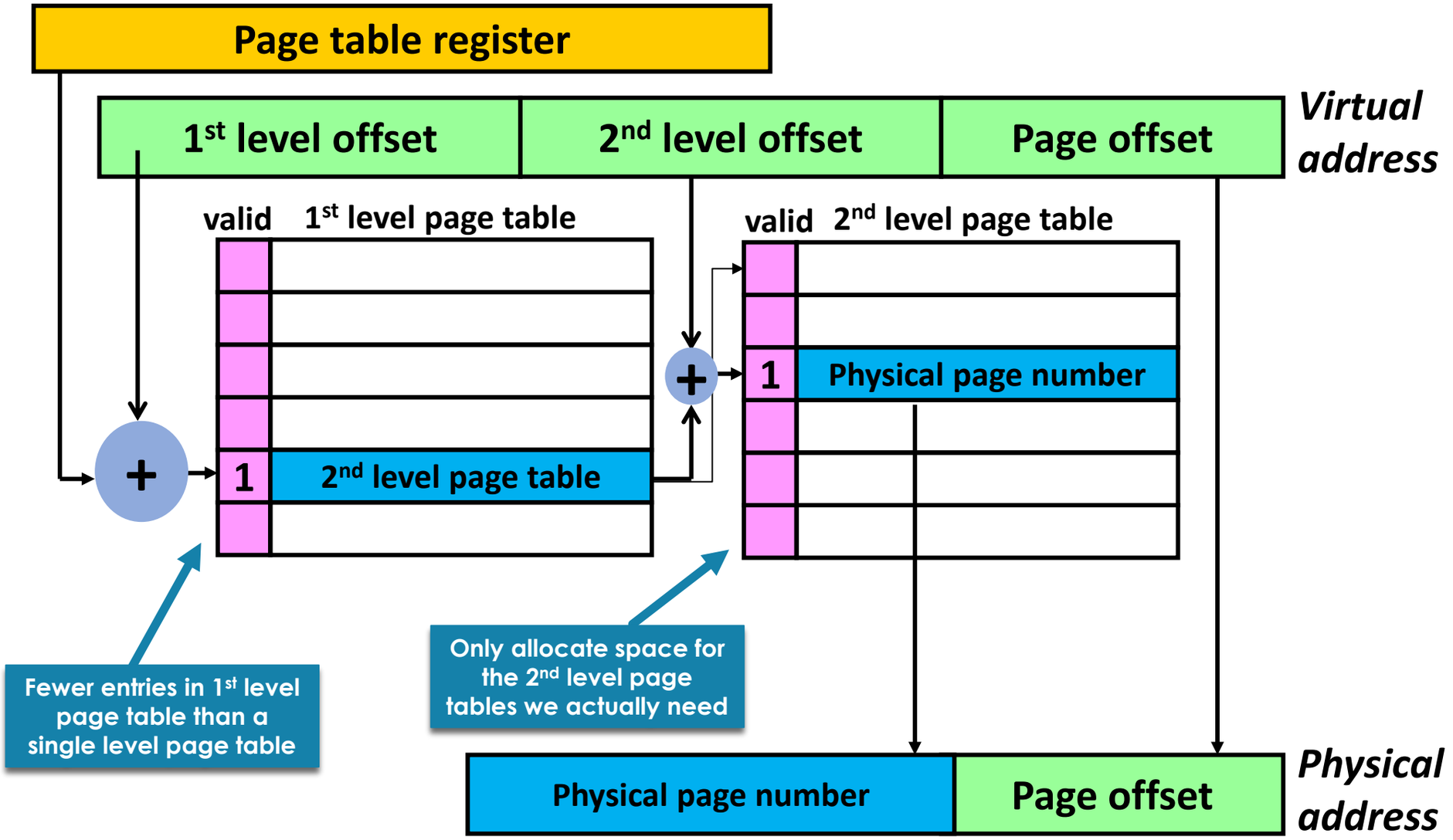
valid	PPN
1	0x4
1	0x5
0	
0	

valid	PPN
1	0x1
1	0x2
1	0x3
0	

valid	PPN
0	
1	0x6
0	
0	

valid	PPN
1	0x7
0	
0	
0	

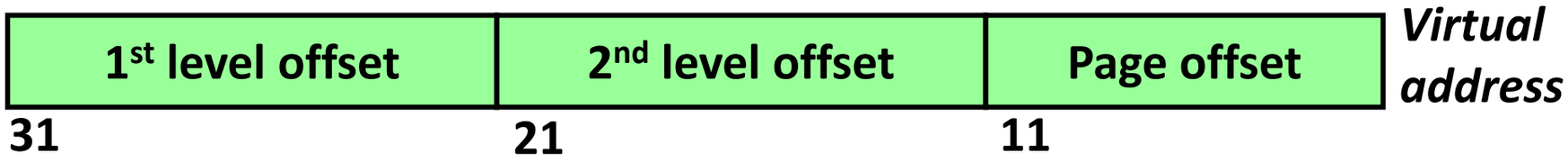
Hierarchical page table



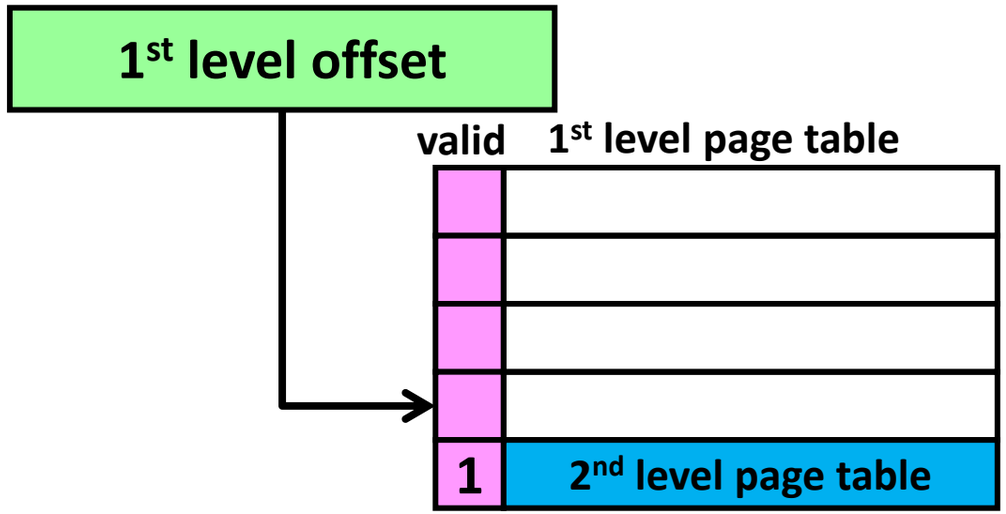
Agenda

- Motivation for Multi-level Page Tables
- **Example architecture**
- Class Problem: Multi-Level VM Design
- VM Miscellanea

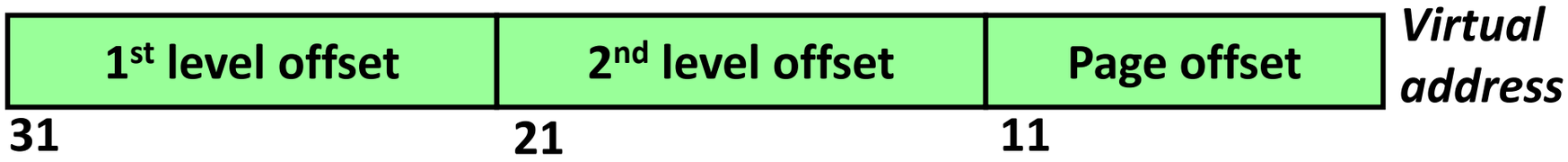
Hierarchical page table – 32bit Intel x86



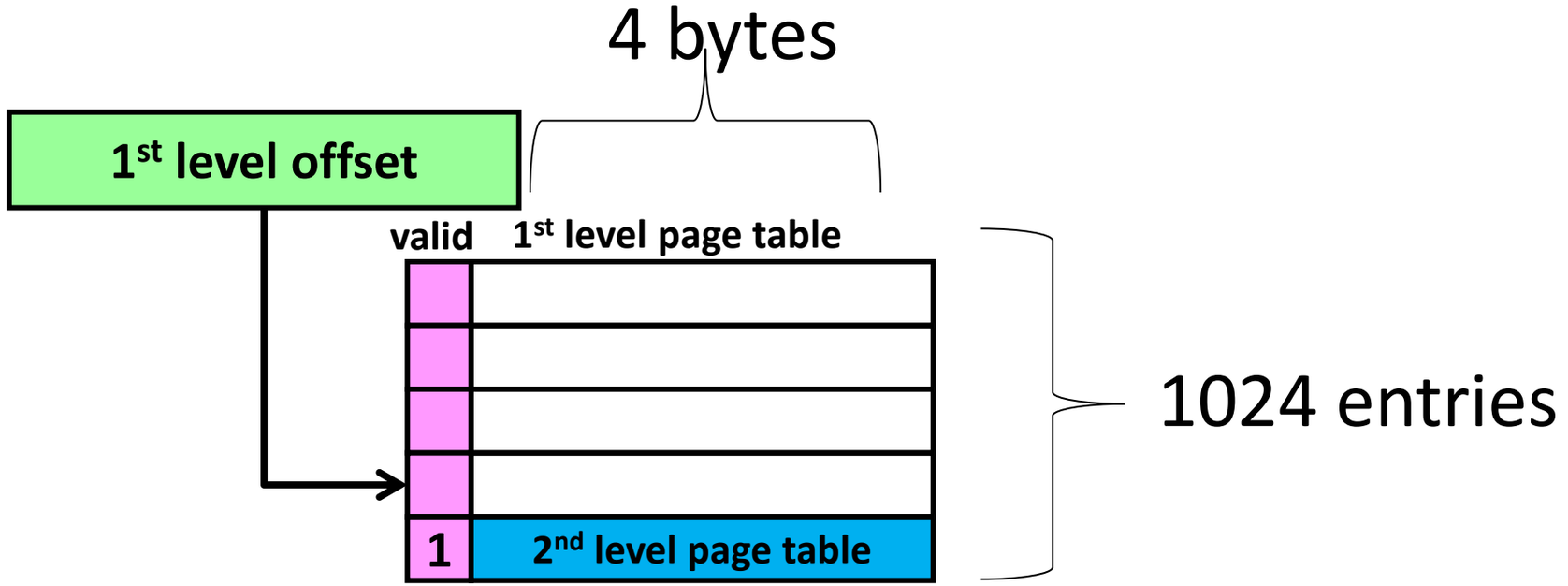
- How many bits in the virtual 1st level offset field? 10
- How many bits in the virtual 2nd level offset field? 10
- How many bits in the page offset? 12
- How many entries in the 1st level page table? 2^{10}=1024



Hierarchical page table – 32bit Intel x86

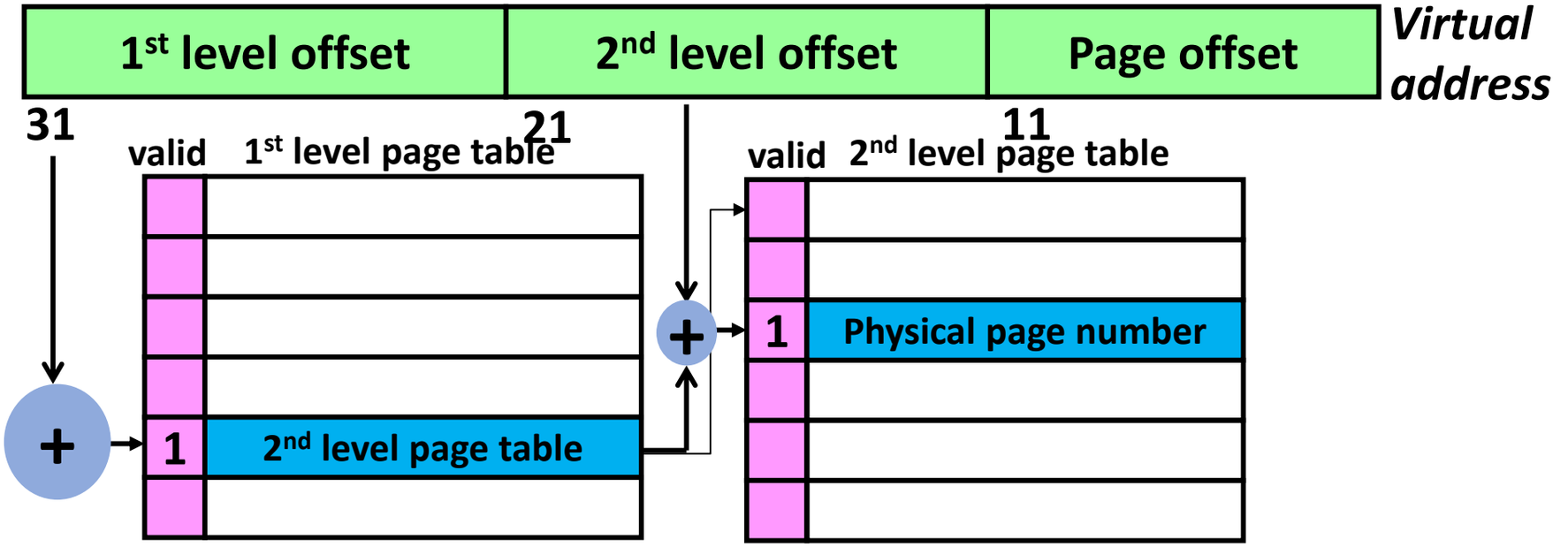


- Let's say physical address size + overhead bits is 4 bytes per entry
- Total size of 1st level page table
  - 4 bytes * 1024 entries = 4 KB



Hierarchical page table

- How many entries in the 2nd level of the page table?
  - $2^{10} = 1024$
- How many bytes for each VPN in a 2nd level table?
  - Let's round up to 4 bytes



Hierarchical page table – 32bit Intel x86

Virtual address

• How many bits in the virtual 1st level offset field?	10
• How many bits in the virtual 2nd level offset field?	10
• How many bits in the page offset?	12
• How many entries in the 1st level page table?	2^{10}=1024
• How many bytes for each entry in the 1st level page table?	4
• How many entries in the 2nd level of the page table?	2^{10}=1024
• How many bytes for each entry in a 2nd level table?	~4
• **What is the total size of the page table?** **(here *n* is number of valid entries in the 1st level page table)**	**4K+n*4K**

Agenda

- Motivation for Multi-level Page Tables
- Example architecture
- **Class Problem: 32bit Intel x86**
- VM Miscellanea

Poll

Class Problem (32 bit x86)

- What is the least amount of memory that could be used? When would this happen?
- What is the most memory that could be used? When would this happen?
- How much memory is used for this memory access pattern:
  0x00000ABC
  0x00000ABD
  0x10000ABC
  0x20000ABC

Class Problem (32 bit x86)

- What is the least amount of memory that could be used? When would this happen?
  - 4KB for 1st level page table. Occurs when no memory has been accessed (before program runs)

- What is the most memory that could be used? When would this happen?
  - 4KB for 1st level page table
    + 1024*4KB for all possible 2nd level page tables
    = 4100KB (which slightly greater than 4096KB)
  - Occurs when program uses all virtual pages (= 2^{20} pages)
    - But it can also happen under other circumstances. When?

Class Problem (32 bit x86)

- How much memory is used for this memory access pattern:

```
0x00000ABC  // Page fault
0x00000ABD
0x10000ABC  // Page fault
0x20000ABC  // Page fault
```

- 4KB for 1st level page table + 3*4KB for each 2L page table = 16 KB

Agenda

- Motivation for Multi-level Page Tables
- Example architecture
- Class Problem: 32bit Intel x86
- **Class Problem: Multi-Level VM Design**
- VM Miscellanea

Class Problem – Multi-level VM

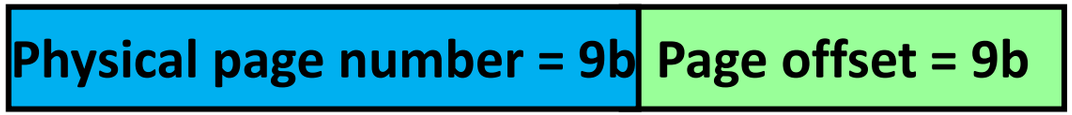
- Design a two-level virtual memory system of a byte addressable processor with ***24-bit long addresses***. No cache in the system. **256Kbytes of memory** installed, and no additional memory can be added.
  - **Virtual memory page: 512 Bytes**. Each page table entry must be an integer number of bytes, and must be the smallest size required to fit the physical page number + 1 bit to mark valid-entry
  - We want each second-level page table to fit exactly in one memory page, and 1st level page table entries are 3 bytes each (a memory address pointer to a 2L page table).
- Compute:
  - Number of entries in each 2nd level page table;
  - Number of virtual address bits used to index the 2nd level page table;
  - Number of virtual address bits used to index the 1st level page table;
  - Size of the 1st level page table.

Class Problem – Multi-level VM

Page Offset: 9 bits (512B page size)

Physical address = 18b (256KB Mem size)

Physical page number = 18b (256KB mem size) – 9b (offset)



2nd level page table entry size: 9b (physical page number) + 1b =~ 2 bytes

2nd level page table **fits exactly in 1 page**

#entries in 2nd level page table is 512 bytes / 2 bytes = 256

#entries in 2nd level page table = 256 ➜Virtual page bits = 8b

Virtual 1st level page bits = 24 – 8 – 9 = 7b

1st level page table size = 2^7 * 3 bytes = 384B

Virtual address = 24b

Agenda

- Motivation for Multi-level Page Tables
- Example architecture
- Class Problem: Multi-Level VM Design
- **VM Miscellanea**

Page Replacement Strategies

- Page table indirection enables a fully associative mapping between virtual and physical pages.

- How do we implement LRU in OS?
  - True LRU is expensive, but LRU is a heuristic anyway, so approximating LRU is fine
  - Keep a "***accessed" bit per page***, cleared occasionally by the operating system. Then pick any "unaccessed" page to evict

Other VM Translation Functions

- Page data location
  - Physical memory, disk, uninitialized data
- Access permissions
  - Read only pages for instructions
  - **This is how your system detects segmentation faults**
- Gathering access information
  - Identifying dirty pages by tracking stores
  - Identifying accesses to help determine LRU candidate

Next time

- Speeding up virtual memory