The x86 VM System

- Topics
  - Hardware support for virtual memory
  - X86 (lots of helpful hardware)
- Learning Objectives:
  - Explain what MMU hardware has to do
  - Describe the x86 MMU support

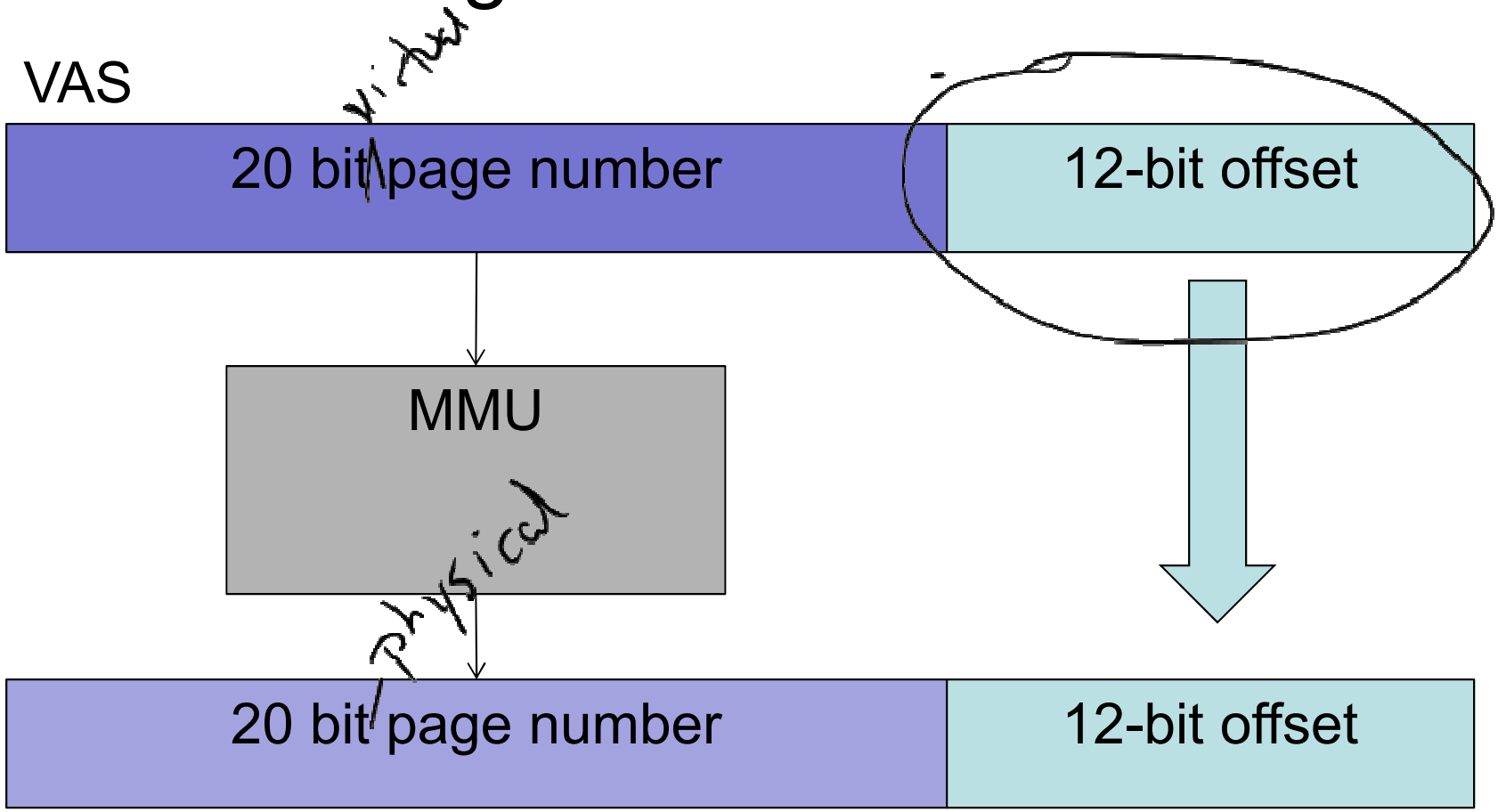
A Mapping Table

- Conceptually, we need a map from triples to physical addresses (or faults)

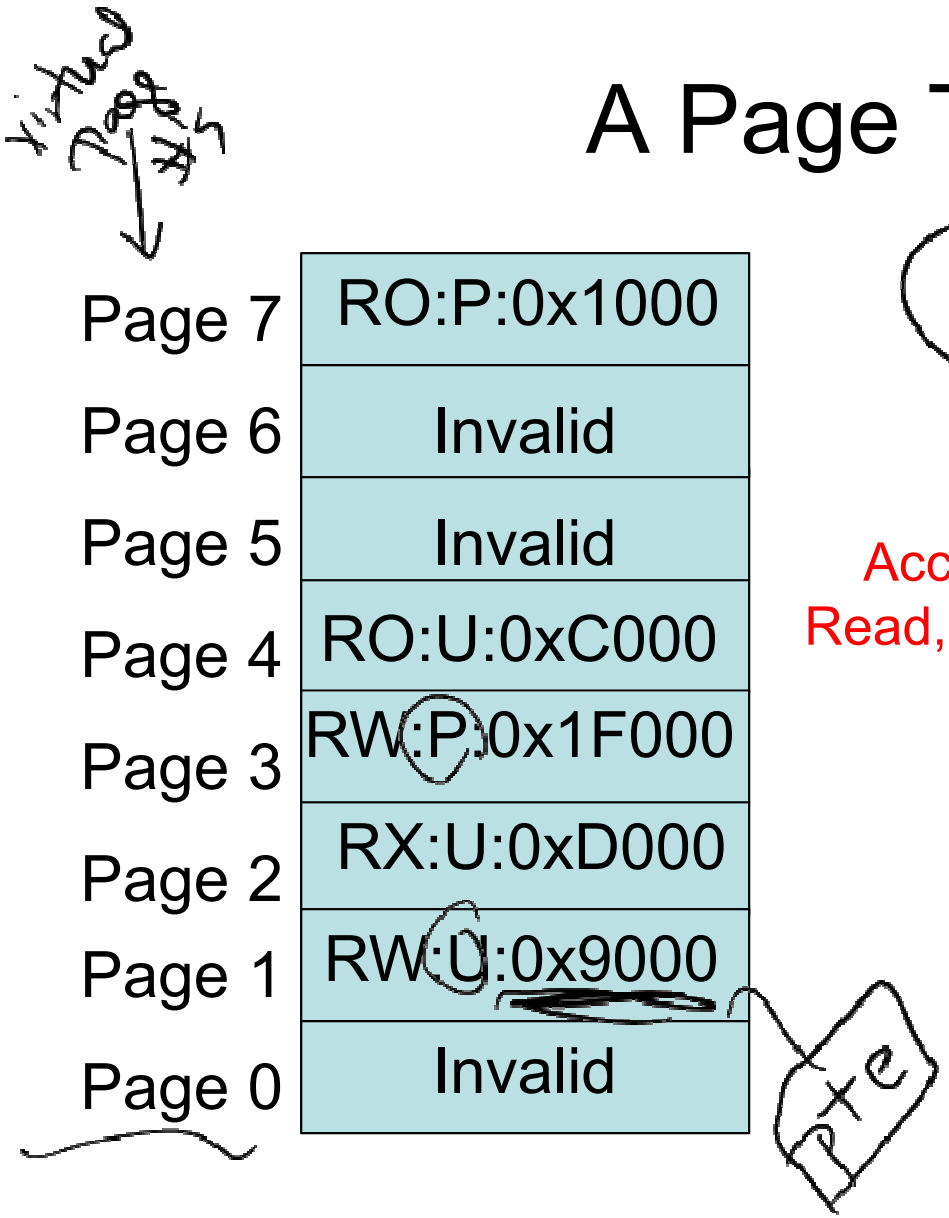
  (VA, access, privilege) => (PA/fault)

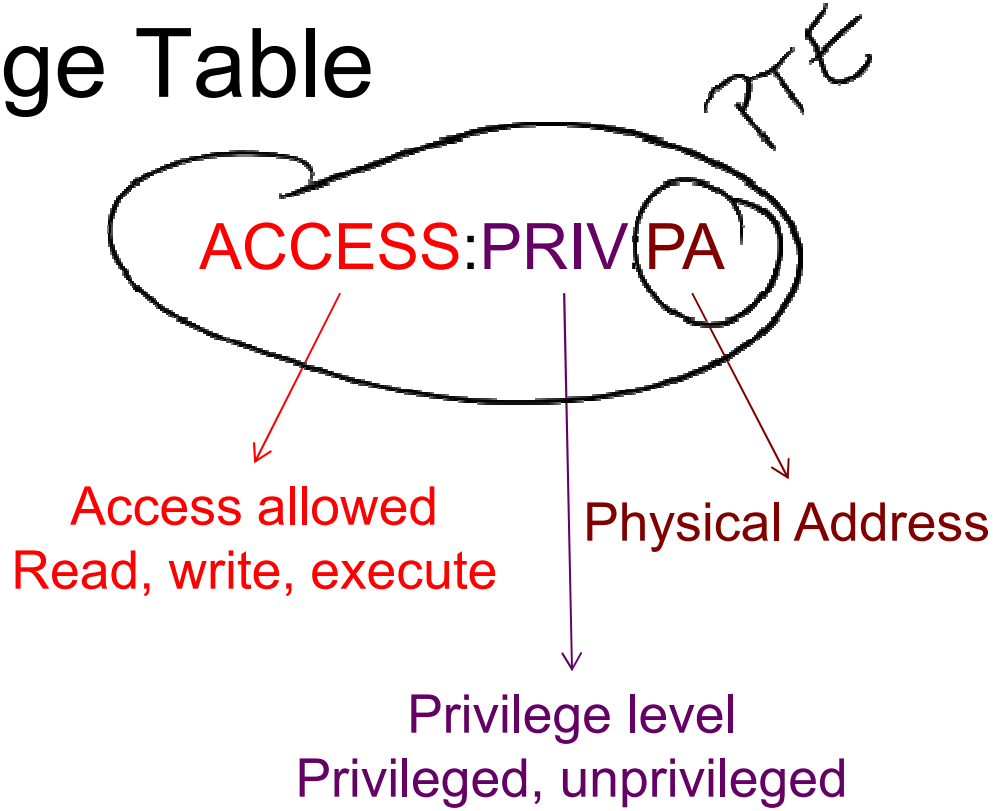
- While we might imagine storing a mapping for every byte in the virtual address space, that would get large pretty rapidly!
- Instead, let's divide memory into fixed-size units called pages.
- We'll store one mapping for each page.
- On the x86, pages are 4 KB.

Dividing the bits in an address



A Page Table


Page 7
RO:P:0x1000
Page 6
Invalid
Page 5
Invalid
Page 4
RO:U:0xC000
Page 3
RW:P:0x1F000
Page 2
RX:U:0xD000
Page 1
RW:U:0x9000
Page 0
Invalid

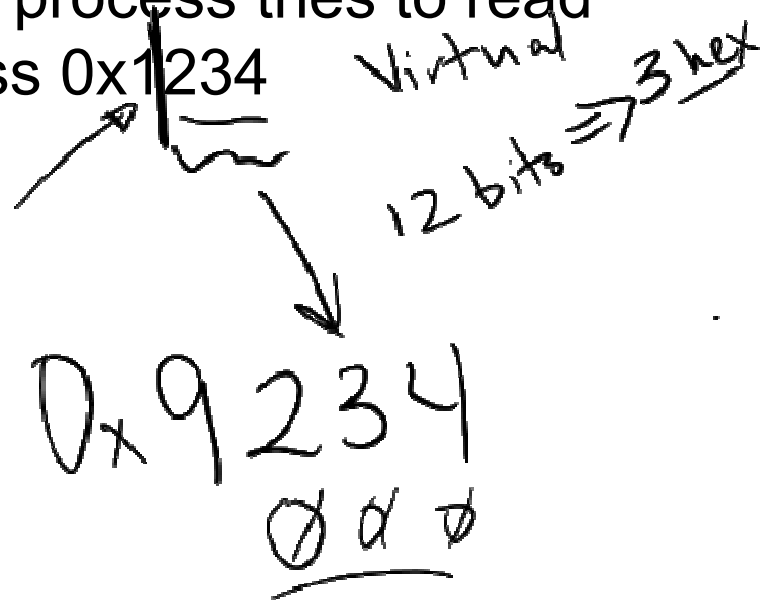

ACCESS:PRIV:PA
Access allowed
Read, write, execute
Privilege level
Privileged, unprivileged
Physical Address

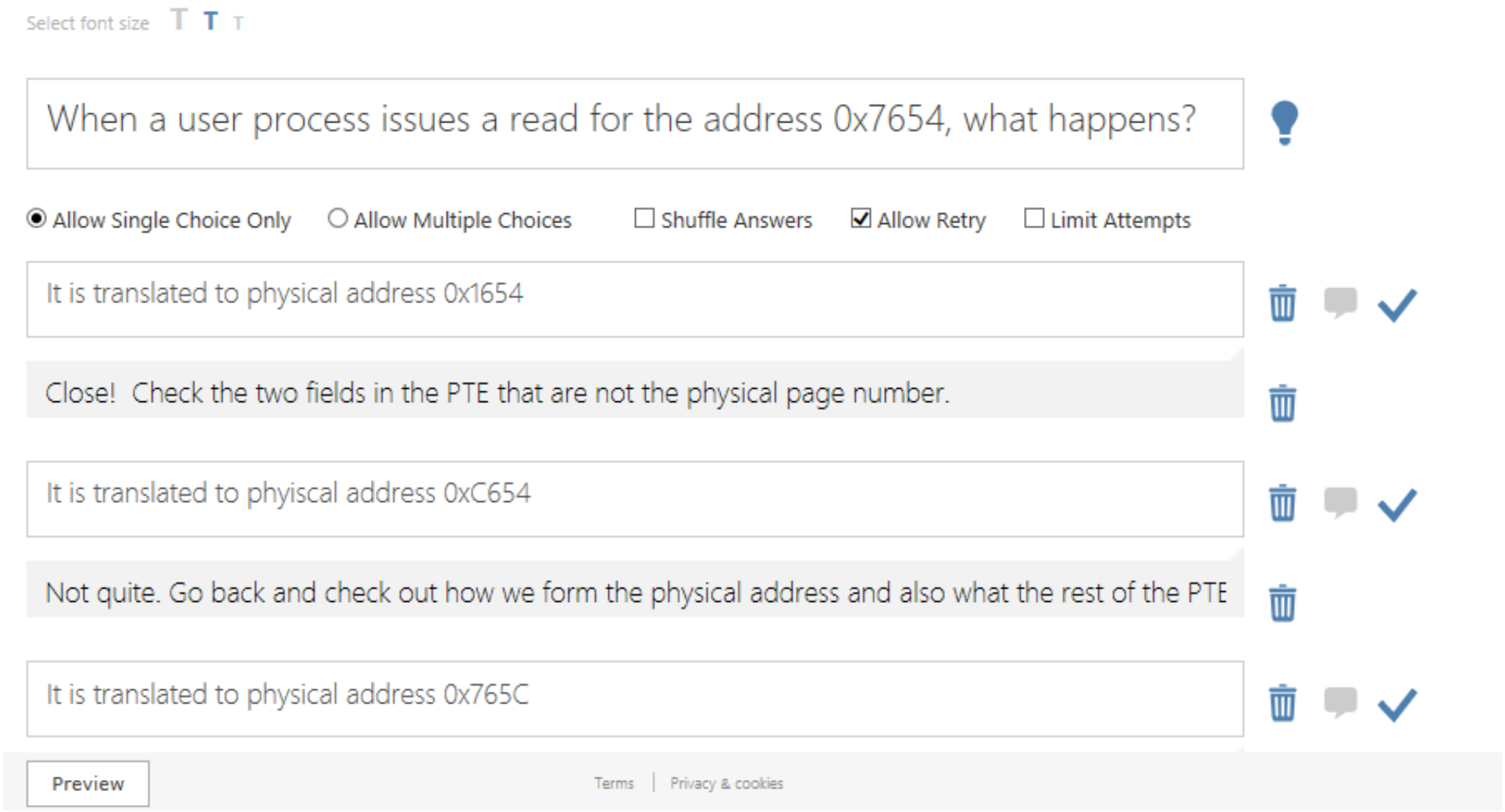
A Page Table

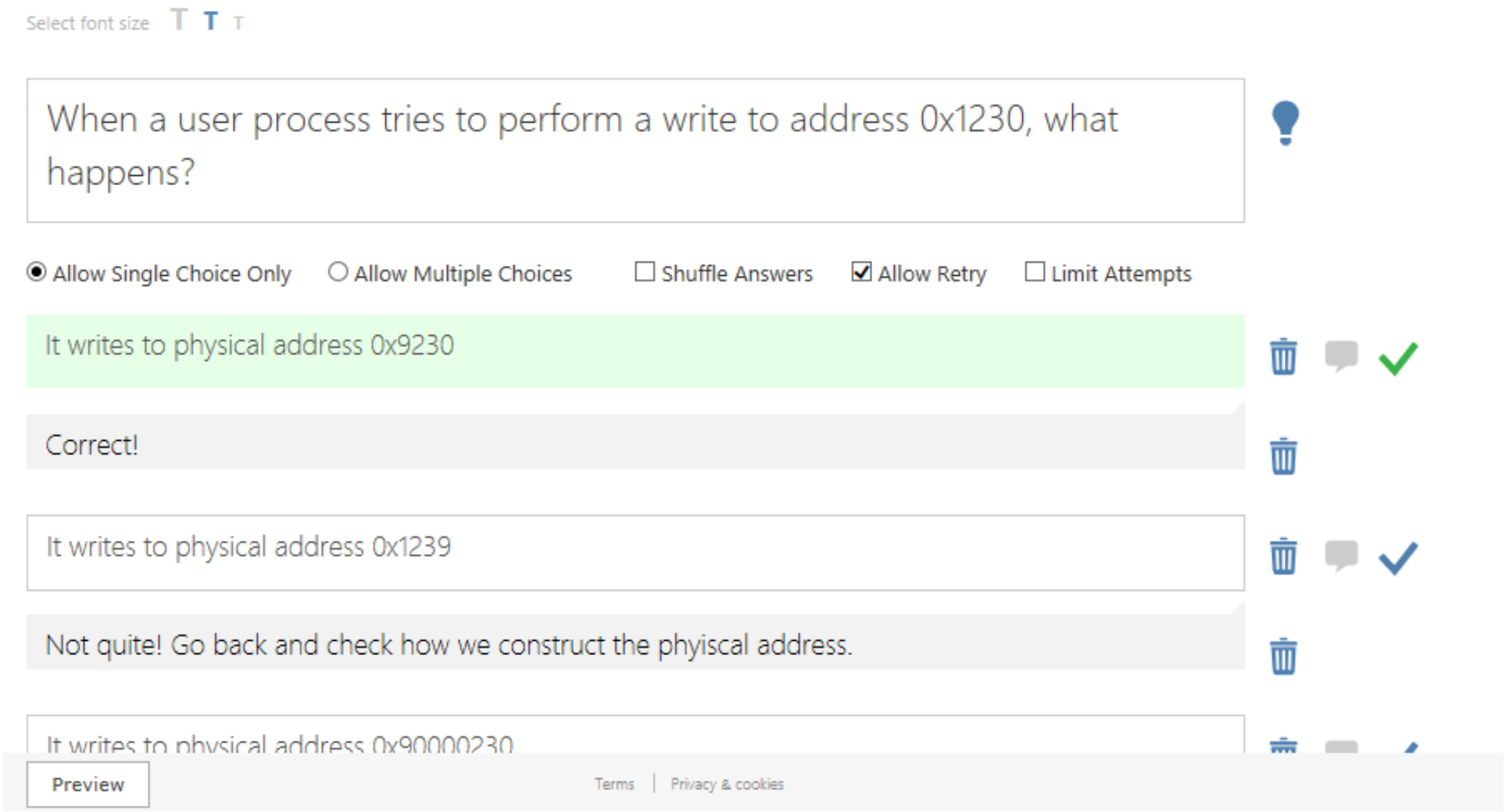
Page 7	RO:P:0x1000
Page 6	Invalid
Page 5	Invalid
Page 4	RO:U:0xC000
Page 3	RW:P:0x1F000
Page 2	RX:U:0xD000
Page 1	RW:U:0x9000
Page 0	Invalid

Let's translate an address!

A user process tries to read address 0x1234

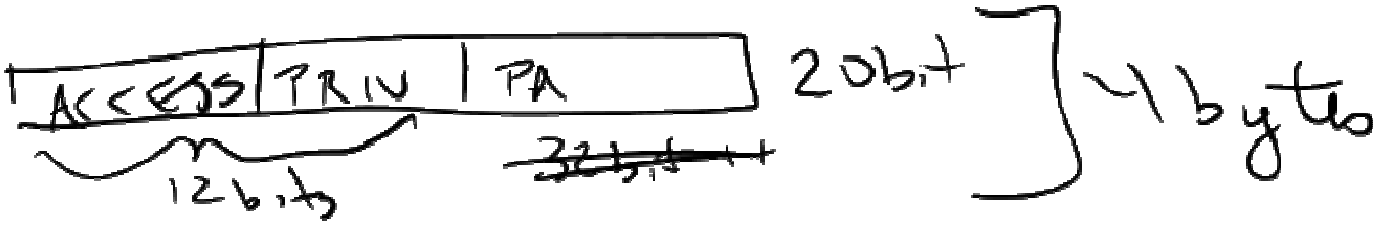



Select font size T T T
When a user process issues a read for the address 0x7654, what happens?
Allow Single Choice Only
Allow Multiple Choices
Shuffle Answers
Allow Retry
Limit Attempts
It is translated to physical address 0x1654
Close! Check the two fields in the PTE that are not the physical page number.
It is translated to phyiscal address 0xC654
Not quite. Go back and check out how we form the physical address and also what the rest of the PTE
It is translated to physical address 0x765C
Preview
Terms
Privacy & cookies


Select font size T T T
When a user process tries to perform a write to address 0x1230, what happens?
Allow Single Choice Only
Allow Multiple Choices
Shuffle Answers
Allow Retry
Limit Attempts
It writes to physical address 0x9230
Correct!
It writes to physical address 0x1239
Not quite! Go back and check how we construct the phyiscal address.
It writes to physical address 0x90000230
Preview
Terms
Privacy & cookies

How big are page tables?

- So, let's see how large our page tables are:
- How many 4 KB pages are there in a 32-bit address space?



$$\frac{2^{32}}{2^{12}} = 2^{20} = 1M \Rightarrow \boxed{4MB}$$

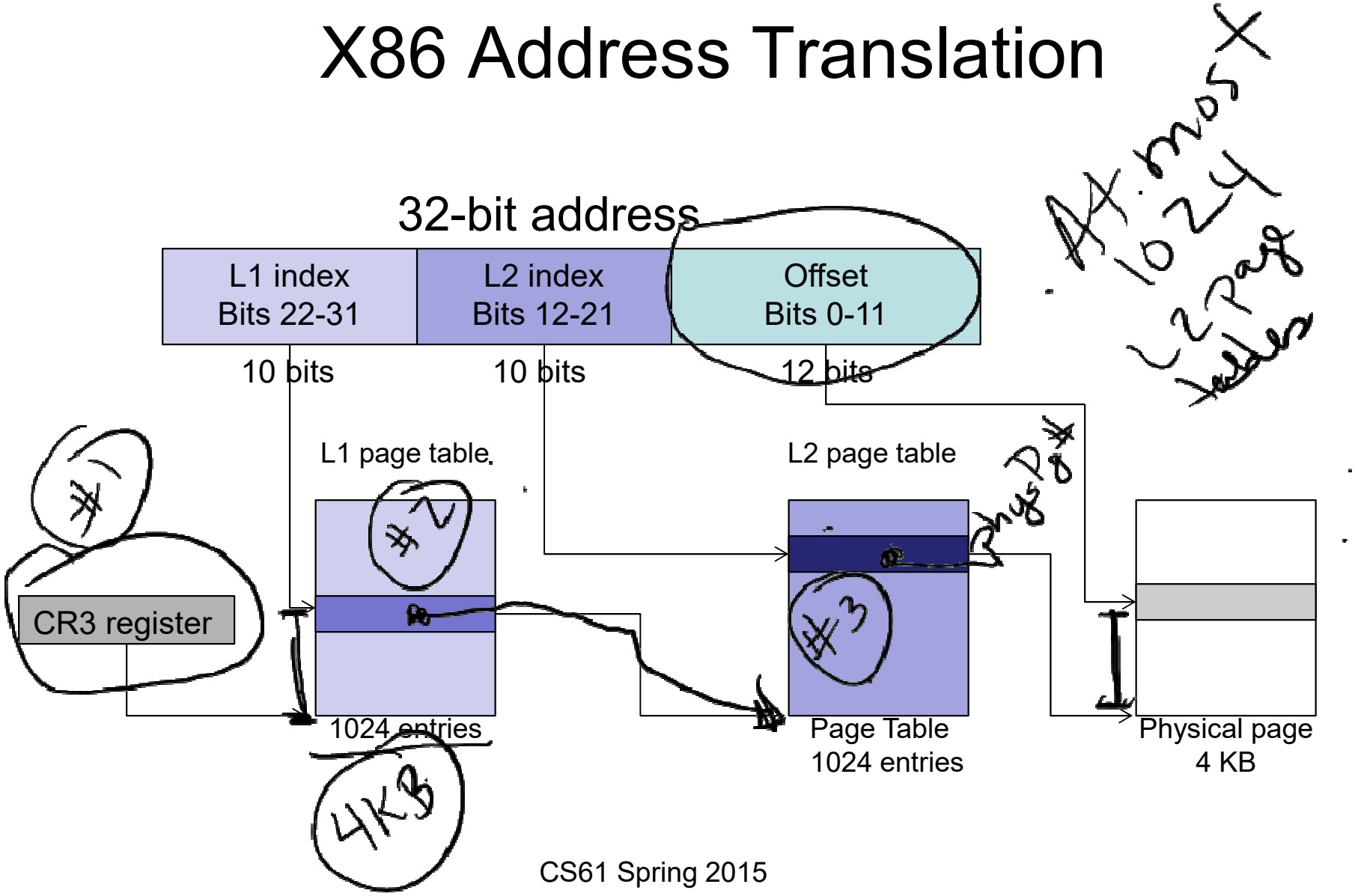
Intel x86 VM System

- The Intel x86 Virtual Memory Architecture is a reflection of many major revisions that have occurred over various generations of Intel microprocessors.
- While this makes the system a bit more complicated than some others, it is the most widely used platform today, so understanding it will serve you well.
- Note:
  - This presentation does not cover x86 in its full glory.
  - We cover it sufficiently so that you can tackle assignment 6 and so that should you ever need to dig into the details, they will make sense.

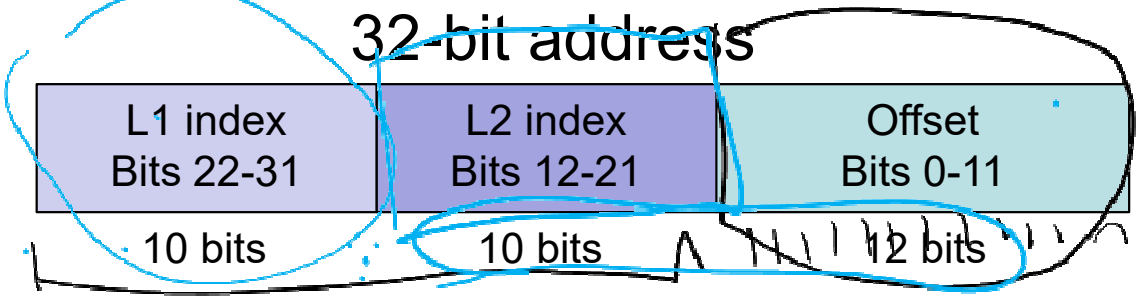
X86 Historical Summary

	Introduced	N Instr	Phys. Mem	VAS size
8008	1972	66	16 KB	None
8080	1974	111	64 KB	None
8086	1978	133	1 MB	None
8088	1981	133	1 MB	None
80286	1982	169	16 MB	16 MB
80386	1985	187	4 GB	64 TB
80486	1989	193	4 GB	64 TB
Pentium	1993	198	4 GB	64 TB
Pentium Pro	1995	234	64 GB	64 TB
Pentium 4	2004		1 TB	16 EB
Core i7	2008		16 TB	16 EB

X86 Address Translation



Manipulating Addresses



- Some handy macros! First, for pages:

```
#define PAGESHIFT 12
#define PAGESIZE (1 << PAGESHIFT)
#define PAGEOFFMASK (PAGESIZE – 1)
#define PAGENUMBER(VA) ((uintptr_t)VA >> PAGESHIFT)
```

- Next for 2-level page tables:

```
#define PAGETABLE_ENTRIES (1 << 10)
#define L1PAGEINDEX(VA) ((uintptr_t)VA >> 22)
#define L2PAGEINDEX(VA) \
    ((uintptr_t)VA >> 12) & (PAGETABLE_NENTRIES – 1)
```

Reading Addresses

32-bit address

- Let's practice extracting fields from hex addresses.
  - How many hex digits for the offset?
  - How many for the L2 index?
  - How many for the L1 index?
- 0xDEADBEEF
  - Offset = 0xEEF
  - L2 Index = 0x2DB
  - L1 Index = 0x37A

Wrapping Up

- Page tables are the data structure that maps from virtual addresses to physical addresses.
- The x86 implements 2-level page tables to conserve memory.
- While a complete flat page table would require 4 MB of memory, a tiny process can make do with far less using 2-level page tables:
  - 1 L1 page table (4 KB)
  - 1 L2 page table (4 KB)
  - Total: 8 KB