Part C:
How big is the page table?

- Page table can actually become pretty big…
- Example #1:
    - 4 KB pages
        - Therefore need 2^{12} (or 12 bits of offset)
        - (Offset does same thing that it does in cache block – just picks page entry)
    - 32-bit virtual address
        - 32 – 12 = 20 bits of VPN
    - 4 Byte / page table entry
        - Holds LRU status, valid, dirty, PFN (~32 bits)
- Therefore, 2^{20} entries in page table, each ~ 4 bytes each → 4 Mbytes
    - Not as big as memory, but what about cache?

Another Example…

- Assume
    - Virtual address = 64 bits
    - 4 KB pages
    - 4 bytes/page

VPN (52 bits)	Offset (12 bits)

- PT would be:
    - $4.5 \times 10^{15} \times 4 \sim 10^{16}$ bytes → 10 *petabytes!*
- Soluton(s):
    - Multi-level, inverted page tables – you'll learn about in OS

Part D:
Is page table / virtual address translation slow?

- It can be → have maybe 2 references / translation
- Solution: TLB = "Translation Lookaside Buffer"
    - Fast cache for page table

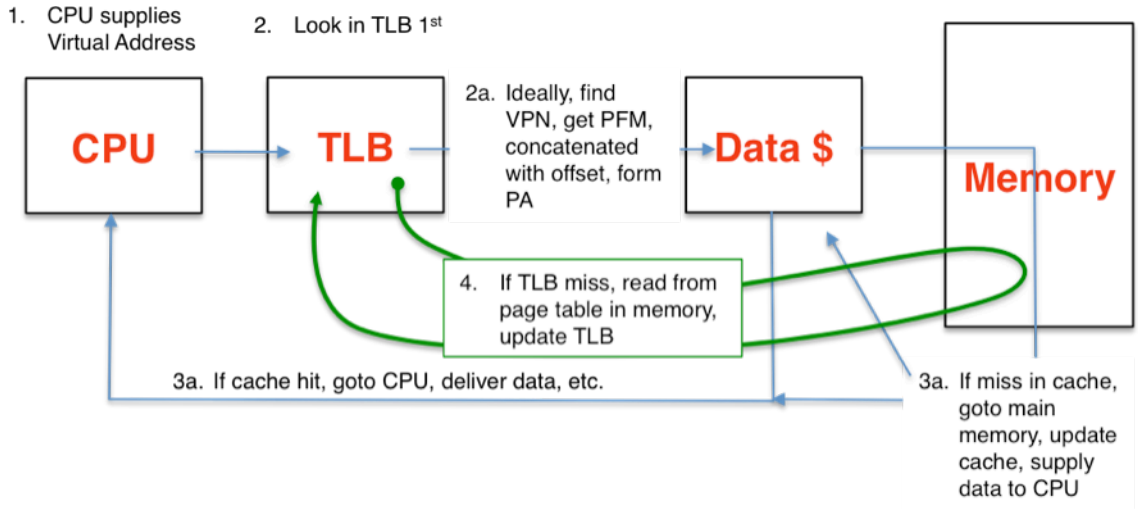
What does the TLB look like?

- It's a really small, fully associative cache

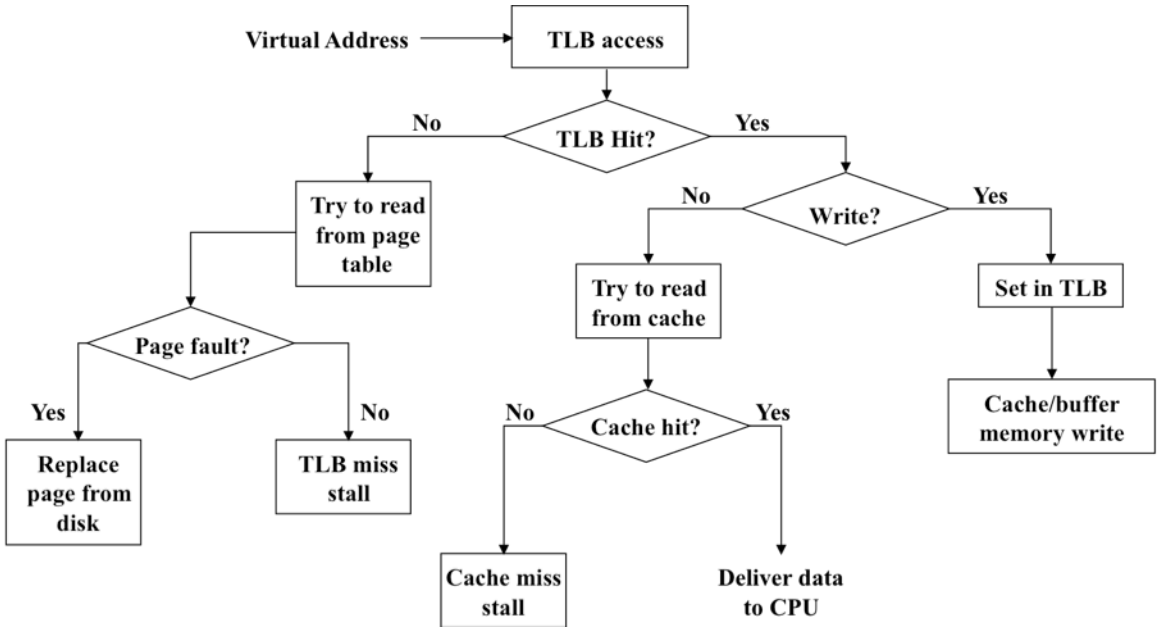
Virtual Page #	Physical Frame #	Dirty	LRU	Valid

1. All of the virtual page numbers would be searched for a match
2. The physical frame number is the data that is supplied
3. The physical frame number is concatenated with an offset to form a physical address

Where is the TLB on the critical path?



See flow chart below / in notes:



Please make note of approximate times for each of 5 “critical paths”
i.e. from left-to-right: worst case (10^6 CCs), 1 memory reference needed (~100 CCs), ~10-100 CCs, ideal (1-2 CCs), (1-2 CCs)

Part E: Example 1:

Assume a machine with the following characteristics:

- The CPU supplies a 64 bit virtual address.
- The 64 bit virtual address must be translated into a 64 bit physical address.
- This single core machine has 2 levels of cache.
- The clock rate is 1 GHz.

Question A:

The page table has 2^{32} entries. Determine the physical address associated with the virtual address using portions of the memory state of the machine provided below. (Addresses hex.)

Page Table Register: AAAA 0000 0000 0000
Virtual Address: 0000 BBBB 0000 AAAA

	Address (LSB at right, MSB at left)	Data
1	0000, 0000, 0000, 0000	FEDC, BA98, 7654, 3210
2	0000, 0000, 1111, 1111	0123, 4567, 89AB, CDEF
3	0000, 0000, AAAA, 0000	0000, 0000, 3333, 7777
4	0000, 0000, DDDD, 9988	2222, 7777, 8888, 4444
5	0000, 000F, 0000, 0000	E1E1, 0E1E, 10E1, E100
6	0000, BBBB, 0000, AAAA	0000, 0000, EEEE, FFFF
7	0000, BBBB, AAAA, AAAA	0000, 0000, 9999, 2222
8	000A, AAA0, 0000, 0000	0000, 0000, 2222, 2222
9	0BBB, AAAA, 0000, AAAA	0000, 0000, 5555, 4444
10	AAAA, 0000, 0000, 0000	0000, 0000, BBBB, CCCC
11	AAAA, 0000, 0000, BBBB	0000, 0000, FFFF, EEEE
12	AAAA, BBBB, BBBB, AAAA	1111, 0000 AAAA, 0000
13	BBBB, AAAA, 0000, 0000	0000, 0000, 9999, 1111
14	BBBB, BBBB, BBBB, AAAA	0000, 0000, 9999, 8888
15	CCCC, CCCC, DDDD, DDDD	8888, 9999, 0000, 1111
16	FFFF, CCCC, DDDD, AAAA	1111, 1111, 0000, AAAA

Goto memory location AAAA 0000 0000 0000
\+ 0000 BBBB
AAAA 0000 0000 BBBB
Get: 0000 0000 FFFF EEEE
Physical address is: FFFF EEEE 0000 AAAA

Question B:

The physical address generated above is then sent to a direct mapped, L1 cache. The L1 cache has the following characteristics:

- The cache has 4096 blocks.
- There are 256, 32-bit words in each block. Addresses are to the word.
- The cache can hold 4 MB (i.e. 4,194,304 bytes) of data.

Using the physical address found in Part A, fill in the following table:

Index	0AA
Offset	AA
Tag	FFFF EEEE 000

Physical address: FFFF EEEE 0000 AAAA

256 words/block → 2^8 words/block → 8 bits of offset → AA

4096 Blocks → 2^{12} → 12 bits of index → 0AA

Therefore there are 64 – 12 – 8 = 44 bits of tag → FFFF EEEE 000

Question C:

Assume that you have a sequence of 3 virtual addresses that you need to convert to physical addresses. The first virtual address takes 3 nanoseconds to translate to a physical address. The second virtual address takes 300,000 nanoseconds to translate to a physical address. The third virtual address takes 100 nanoseconds to translate to a physical address. For each virtual address, *briefly* comment on the critical path of translation.

3 ns	TLB hit + cache hit
300000 ns	TLB miss + page fault
100 ns	(1) TLB Miss + Page Table Hit OR (2) TLB Hit + Cache Miss

Part E: Example 2:

Assume that the 1st three lines of code for Microsoft Word are as follows – really!

Address 1:	10000	# static variable 10000
Address 2:	20000	# static variable 20000
Address 3:	Load R17 Address 1	# Load 10000 into R17

Thus, we want to load the data at address 1 of the Microsoft Word assembly code into register #17.

Question A:

Using a modern superscalar machine's datapath as context (with two levels of on-chip cache), list all of the steps involved with loading this initial value stored in the program code written by Microsoft into physical register #17. You have just turned your machine on and are loading Microsoft Word for the first time.

TLB Miss, Page Table Lookup, Page Fault, Page Table update, TLB Update, L1 $ Miss, L2 $ Miss, Memory Reference, L2 Update, L1 Update, data into R17.

Question B:

Provide a rough estimate of the amount of time that this would take.

Seconds!