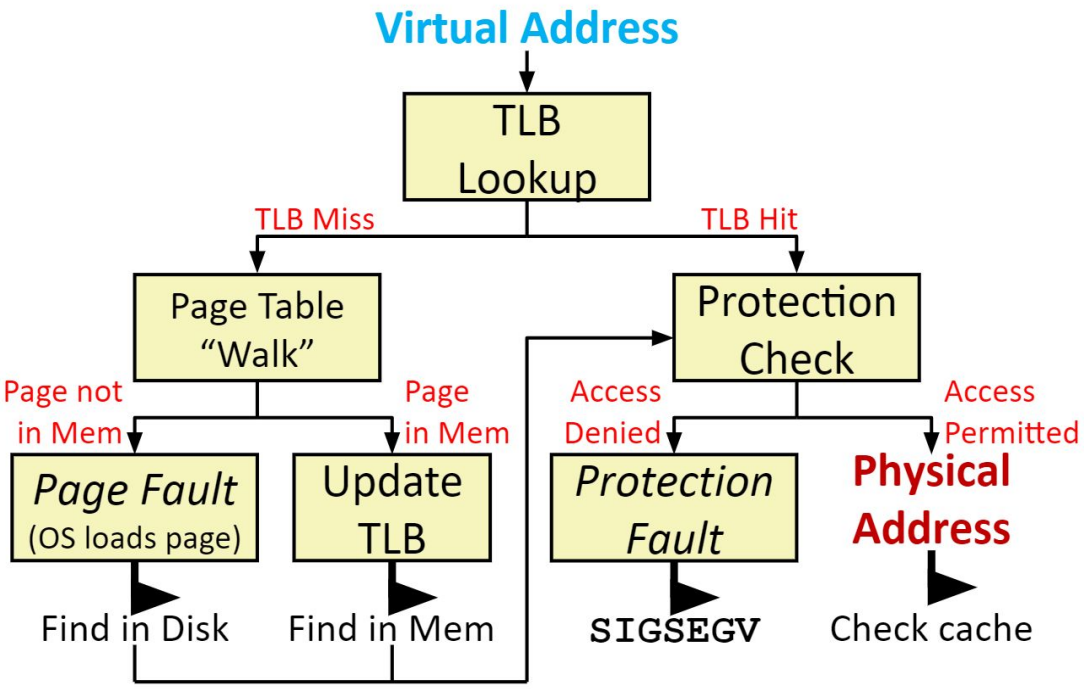
CSE 351 Section 8 – Virtual Memory

Hi there! Welcome back to section, we're happy that you're here ☺

Address Translation



Address Translation Symbols

Basic Parameters:

$N = 2^n$	Number of addresses in virtual address space
$M = 2^m$	Number of addresses in physical address space
$P = 2^p$	Page size (bytes)

Components of Virtual Address (VA):

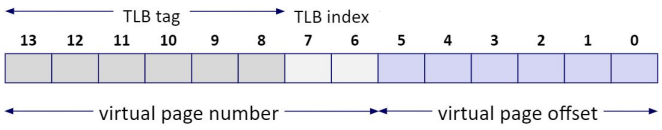
VPO	Virtual Page Offset
VPN	Virtual Page Number
TLBI	TLB Index
TLBT	TLB Tag

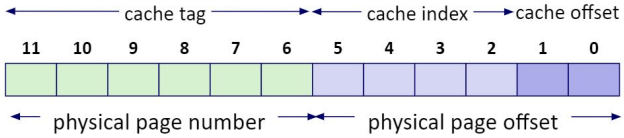
Components of Physical Address (PA):

PPO	Physical Page Offset (Same as VPO)
PPN	Physical Page Number

A Small Example

Suppose we have a simple memory system with **14-bit** virtual addresses, **12-bit** physical addresses, and a page size of **64 bytes**. The TLB has **16 entries** in total and is **4-way** set associative. The cache is direct-mapped with **16 sets** and a block size of **4 bytes**.





Current State of Memory System:

TLB:

Set	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Page table (partial):

VPN	PPN	V	VPN	PPN	V
0	28	1	8	13	1
1	–	0	9	17	1
2	33	1	A	09	1
3	02	1	B	–	0
4	–	0	C	–	0
5	16	1	D	2D	1
6	–	0	E	–	0
7	–	0	F	0D	1

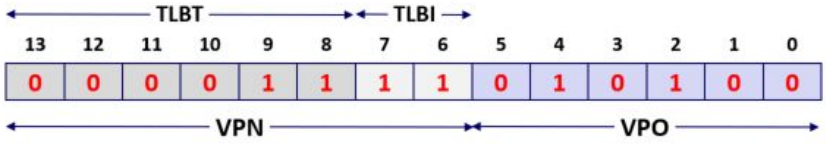
Cache:

Index	Tag	V	B0	B1	B2	B3	Index	Tag	V	B0	B1	B2	B3
0	19	1	99	11	23	11	8	24	1	3A	00	51	89
1	15	0	–	–	–	–	9	2D	0	–	–	–	–
2	1B	1	00	02	04	08	A	2D	1	93	15	DA	3B
3	36	0	–	–	–	–	B	0B	0	–	–	–	–
4	32	1	43	6D	8F	09	C	12	0	–	–	–	–
5	0D	1	36	72	F0	1D	D	16	1	04	96	34	15
6	31	0	–	–	–	–	E	13	1	83	77	1B	D3
7	16	1	11	C2	DF	03	F	14	0	–	–	–	–

Memory Requests

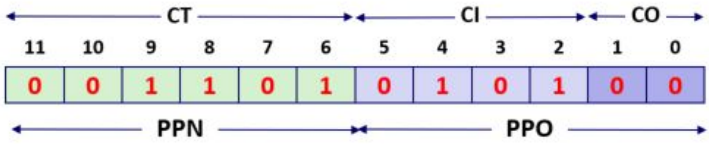
1.

Virtual Address: 0x03D4



VPN 0xF TLBT 0x03 TLBI 3 TLB Hit? Y Page Fault? N PPN 0x0D

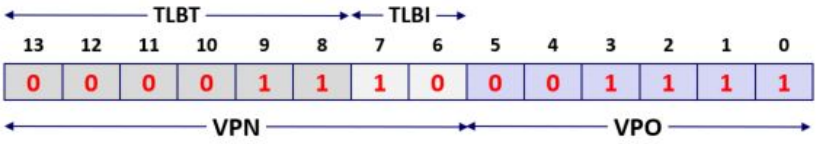
Physical Address:



CT 0x0D CI 5 CO 0 Cache Hit? Y Data (byte) 0x36

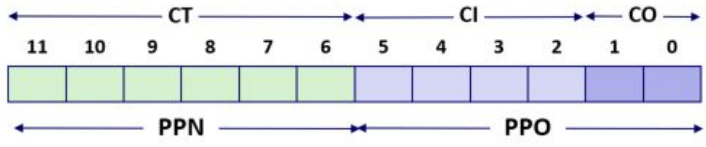
2.

Virtual Address: 0x038F



VPN **0x0E** TLBT **0x03** TLBI **2** TLB Hit? **N** Page Fault? **Y** PPN **N/A**

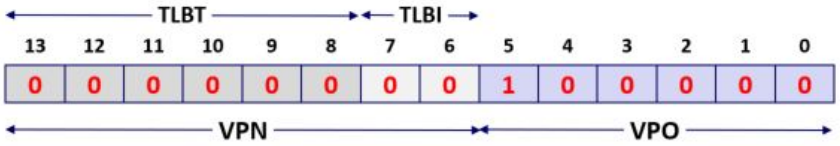
Physical Address:



CT N/A CI N/A CO N/A Cache Hit? N/A Data (byte) N/A

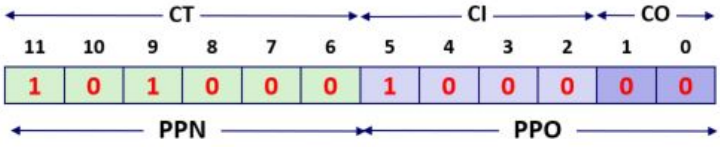
3.

Virtual Address: 0x0020



VPN **0x00** TLBT **0x00** TLBI **0** TLB Hit? **N** Page Fault? **N** PPN **0x28**

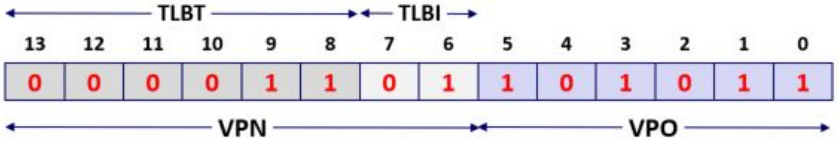
Physical Address:



CT **0x28** CI **8** CO **0** Cache Hit? **N** Data (byte) **N/A**

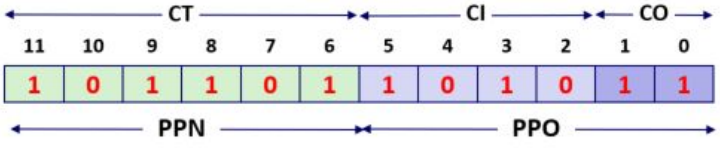
4.

Virtual Address: 0x036B



VPN **0x0D** TLBT **0x03** TLBI **1** TLB Hit? **Y** Page Fault? **N** PPN **0x2D**

Physical Address:



CT **0x2D** CI **A** CO **3** Cache Hit? **Y** Data (byte) **0x3B**

Virtual Memory Table

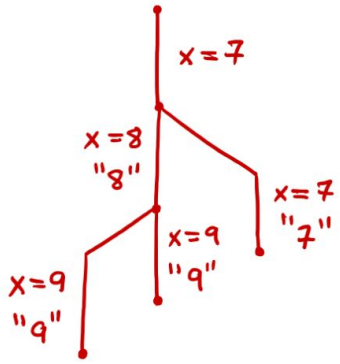
VA width (n)	PA width (m)	Page size (P)	VPN width	PPN width	Bits in PTE (assume V, D, R, W, X)
32	32	16 KiB	18	18	23
32	26	8 KiB	19	13	18
36	32	32 KiB	21	17	22
40	36	32 KiB	25	21	26
64	40	64 KiB	48	24	29

$p = \log_2 P$, VPN width $= n - p$, PPN width $= m - p$, bits in PTE $=$ PPN width $+ 5$

Fork and Concurrency

Consider this code using Linux's `fork`:

```
int x = 7;
if( fork() ) {
    x++;
    printf(" %d ", x);
    fork();
    x++;
    printf(" %d ", x);
} else {
    printf(" %d ", x);
}
```



What are *all* the different possible outputs (i.e. order of things printed) for this code?

(Hint: there are four of them.)

7 8 9 9
8 7 9 9
8 9 7 9
8 9 9 7