

Virtual Memory Examples¹

This example problem will help demonstrate how **virtual addresses** are translated into **physical addresses**.

Imagine a system with the following parameters:

- **Virtual addresses:** 20 bits
- **Physical addresses:** 18 bits
- **Page size:** 1 KB
- **TLB:** 2-way set associative, 16 total entries
- **Single-level page table**

The contents of the TLB and **the first 32 entries** of the page table are shown as follows. **All numbers are in hexadecimal.**

Acronyms and Terminology used in this handout:

- **TLB** – Translation Lookaside Buffer
- **TLBI** – TLB index
- **TLBT** – TLB tag
- **PPN** – Physical page number
- **PPO** – Physical page offset
- **VPN** – Virtual page number
- **VPO** – Virtual page offset
- **PTE** – Page table entry
- **PDE** – Page directory entry
- **CT** – Cache tag
- **CI** – Cache index
- **CO** – Cache offset

¹ Adapted from an exercise used at Carnegie Mellon University
Page 1 of 8

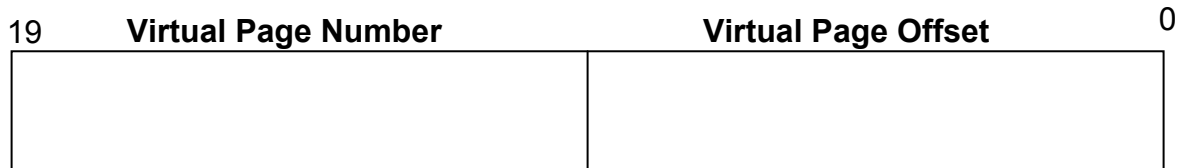
TLB						
Index	Set 0			Set 1		
	Tag	PPN	Valid	Tag	PPN	Valid
0	03	C3	1	01	71	0
1	00	28	1	01	35	1
2	02	68	1	3A	F1	0
3	03	12	1	02	30	1
4	7F	05	0	01	A1	0
5	00	53	1	03	4E	1
6	1B	34	0	00	1F	1
7	03	38	1	32	09	0

Page Table		
VPN	PPN	Valid
000	71	1
001	28	1
002	93	1
003	AB	0
004	D6	0
005	53	1
006	1F	1
007	80	1
008	02	0
009	35	1
00A	41	0
00B	86	1
00C	A1	1
00D	D5	1
00E	8E	0
00F	D4	0
010	60	0
011	57	0
012	68	1
013	30	1
014	0D	0
015	2B	0
016	9F	0
017	62	0
018	C3	1
019	04	0
01A	F1	1
01B	12	1
01C	30	0
01D	4E	1
01E	57	1
01F	38	1

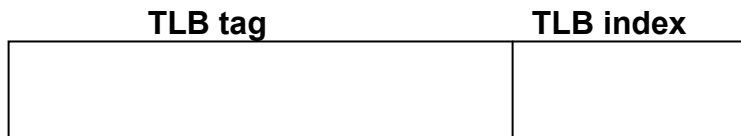
Part 1: Components of Virtual and Physical Addresses

1. **Virtual Address Fields.** The virtual address is 20 bits.

- a. Show the number of bits used for the **virtual page number** and the **virtual page offset** below.



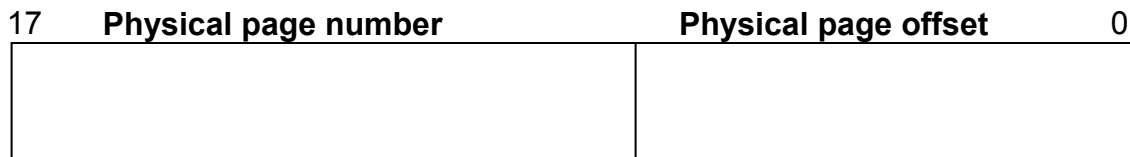
- b. Show the number of bits used for the TLB tag and TLB index. **Label the bit numbers to denote the bit range.**



- c. Assuming a single-level paging system, how many entries are in the page table?

2. **Physical Address Fields.** The physical address is 18 bits.

- a. Show the number of bits used for the **physical page number** and the **physical page offset**. Label the bit numbers to denote the bit range.



Part 2: Virtual Addresses, Physical Addresses, TLBs, and Page faults

For the given virtual addresses, indicate:

1. Which TLB entry is accessed
2. The physical address which is formed
3. Whether the TLB access was a hit or a miss
4. Whether a page fault occurs
5. The physical page number

1. **Virtual address:** 0x078E6
 - a. **Virtual address in binary** (one bit per box)

19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

2. **Address translation**

Parameter	Value
Virtual Page Number	0x
TLB Index	0x
TLB Tag	0x
TLB Hit? (Hit/Miss)	
Page Fault? (yes/no)	
Physical page number	0x

3. **Physical address** (one bit per box)

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1. **Virtual address:** 0x04AA4
 - a. **Virtual address in binary** (one bit per box)

19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

2. **Address translation**

Parameter	Value
Virtual Page Number	0x
TLB Index	0x
TLB Tag	0x
TLB Hit? (Hit/Miss)	
Page Fault? (yes/no)	
Physical page number	0x

3. **Physical address** (one bit per box)

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Part 3: An Example: Pentium III Memory System

- **2 level paging system**
 - **Page directory**
 - contains pointers to page tables
 - one page directory per process
 - **Page table**
 - contains pointers to pages
- Page table entries (PTE) and page directory entries (PDE) are 32 bits wide. (2-level page table)

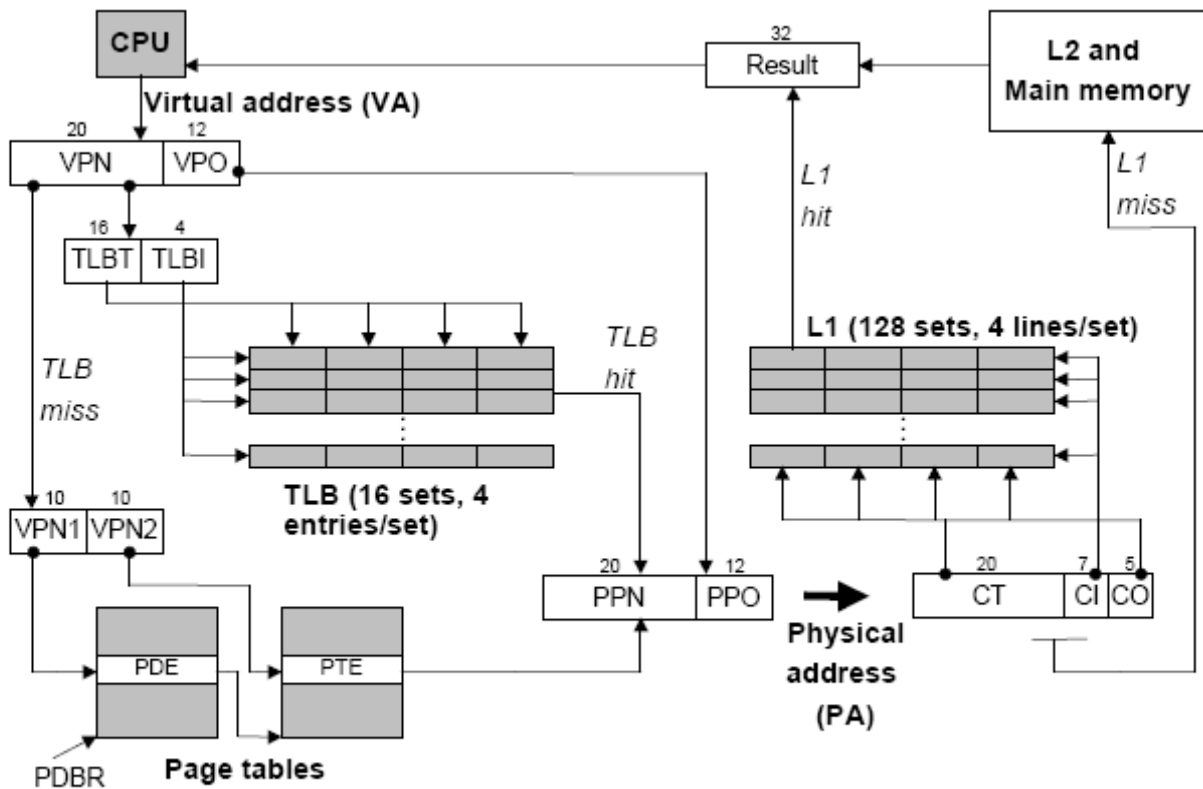


Figure 1: Summary of Pentium address translation).

1. How many entries can the level-1 page table hold?

2. Assume there is a single task running on the system.
 - The task's heap area is allocated in the *virtual* address range 0x660000 – 0x666600.
 - The task's stack area is allocated in *virtual* address range 0x7999400 – 0x8000000
 - The task's text area is allocated in the *virtual* address range 0x1000 – 0x1400.
 - The task has no other sections
 - a. How many valid **page table entries** are there?
 - b. How many valid **page directory entries** are there?
 - c. How much memory is in use strictly by the **page directory** and the **page tables**?
3. If the Pentium III used a flat page table, how much space would that page table take up?
4. How can you change the system to double the cache size? What needs to change for your suggested solution?

If you are curious about the contents of the Page Directory and Page Tables:

- **Page Directory Entry**

3	1	1	9	8	7	6	5	4	3	2	1	0
Page table physical base addr		Avail	G	PS		A	CD	WT	U/S	R/W	P=1	

Page table physical base address: 20 most significant bits of physical page table address (forces page tables to be 4KB aligned)

Avail: available for system programmers

G: global page (don't evict from TLB on task switch)

PS: page size 4K (0) or 4M (1)

A: accessed (set by MMU on reads and writes, cleared by software)

CD: cache disabled (1) or enabled (0)

WT: write-through or write-back cache policy for this page table

U/S: user or supervisor mode access

R/W: read-only or read-write access

P: page table is present in memory (1) or not (0)

- **Page Table Entry**

31	1	1	9	8	7	6	5	4	3	2	1	0
Page physical base address		Avail	G	0	D	A	CD	WT	U/S	R/W	P	

Page base address: 20 most significant bits of physical page address (forces pages to be 4 KB aligned)

Avail: available for system programmers

G: global page (don't evict from TLB on task switch)

D: dirty (set by MMU on writes)

A: accessed (set by MMU on reads and writes)

CD: cache disabled or enabled

WT: write-through or write-back cache policy for this page

U/S: user/supervisor

R/W: read/write

P: page is present in physical memory (1) or not (0)