CS61C Spring 2026

Virtual Memory

Discussion 13

1 Page Table Walk

Assume we have 16-bit VPNs, 12-bit PPNs, 8-bit page offsets, and 32-bit page table entries (PTEs). The first six entries of the page table are shown below.

Page Table	Valid?	Dirty?	PPN
0xB61C 0483	Valid	Clean	0x483
0xFB83 A61C	Valid	Dirty	0x61C
0x8483 3F01	Valid	Clean	0xF01
0x7ABC 4103	Invalid	N/A	N/A
0xC012 F7CB	Valid	Dirty	0x7CB
0x15DA C203	Invalid	N/A	N/A
...			

...where each page table entry (PTE) is formatted as:

1 Valid Bit	1 Dirty Bit	18 Status Bits	12 PPN Bits

1.1 Of the first 6 entries in the TLB, fill out the above table for each entry. List whether the PTE is a valid mapping. If so, list translate its corresponding physical page and if the page is clean/dirty.

For each of the Page Table Entries, we can rewrite the hex as binary and decode according to the PTE format. For example, the first entry `0xB61C 0483` is `0b1011_0110...0100_1000_0011`.

- The first MSB is defined as the valid bit which is `1` so our entry is valid.
- The second MSB is `0` which means our entry is not dirty.
- The last 12 bits are for the PPN mapping which (if PTE is valid) corresponds to the physical page allocated to that virtual page. For this mapping, the `PPN= 0x483`.

1.2 For each of the following virtual addresses, answer whether accessing will result in a 1) Page Table Hit or 2) Page Fault, and translate to its corresponding physical address. Each access occurs independently, not sequentially. The next available free page has PPN 0x42D.

(a) `0x000429`

Answer: Page Table Hit, `PA = 0x7CB29`

Our virtual address is split up into `[VPN | Offset]` bits. We have 8-bit offsets so our `offset =` 0x29 `and our` VPN = `0x0004` which corresponds to entry `0xC012F7CB`. Because this is a valid mapping, our `PPN = 0x7CB` which means our physical address is `0x7CB29`.

(b) `0x00018D`

Answer: Page Table Hit, **PA = 0x61C8D**

Following the above procedure: **offset =** 0x8D **and** VPN = **0x0001** which corresponds to entry **0xFB83A61C**. Because this is a valid mapping, our **PPN = 0x61C** which means our physical address is **0x61C8D**.

(c) `0x000345`

Answer: Page Fault, **PA = 0x42D45**

Following the above procedure: **offset =** 0x45 **and** VPN = **0x0003** which corresponds to entry **0x7ABC4103**. This is an invalid mapping which means we have not yet mapped this VPN to a physical page. The next available free page has **PPN = 0x42D** which means our physical address is **0x42D45**. Although not shown, this PTE will be updated with this new mapping.

1.3 Recall that the Page Table Base Register (PTBR) stores the physical address of our page table. For this program, the PTBR = `0x10000`. What is the physical address of the page table entry `0xC012F7CB`?

0x10010. From the PTBR, we know that the first PTE is at **0x10000**. Since each PTE is 4 bytes and the required entry is at the 4th index, **Physical address of the entry = PTBR + offset = 0x10000 + (4 PTEs) = 0x10000 + 16 bytes = 0x10010**.

1.4 We want to reserve the first 10 pages of physical memory to be read-only. How can we modify our page table to accomplish this?

We can add a Read-Only status bit to the metadata which, if true, disallows writing to a specific physical page.

2 Page Table with TLB

2.1 A processor has 16-bit addresses, 256 byte pages, and an 8-entry fully associative TLB with LRU replacement (the LRU field is 3 bits and encodes the order in which pages were accessed, 0 being the most recent). The TLB for the current process is the initial state given below, and we have three free physical pages. Assume that all current page table entries are in the initial TLB. Write out the physical addresses of each location accessed and fill in the final state of the TLB according to the following access pattern. **Free Physical Pages**: `0x17`, `0x18`, `0x19`

Initial TLB

VPN	PPN	Valid	Dirty	LRU
0x01	0x11	1	1	0
0x00	0x00	0	0	7
0x10	0x13	1	1	1
0x20	0x12	1	0	5
0x00	0x00	0	0	6
0x11	0x14	1	0	4
0xac	0x15	1	1	2
0xff	0xff	1	0	3

Final TLB

VPN	PPN	Valid	Dirty	LRU
0x01	0x11	1	1	5
0x13	0x17	1	1	3
0x10	0x13	1	1	6
0x20	0x12	1	1	1
0x23	0x18	1	1	2
0x11	0x14	1	0	4
0xac	0x15	1	1	7
0x34	0x19	1	1	0

Access Pattern:

1. `0x11f0` (Read)

   Hit. PA: `0x14f0`, LRUs: `1, 7, 2, 5, 6, 0, 3, 4`

2. `0x1301` (Write)

   Miss. Map VPN `0x13` to next available free page which is PPN `0x17`. PA: `0x1701` and set valid and dirty bits. LRUs: `2, 0, 3, 6, 7, 1, 4, 5`

3. `0x20ae` (Write)

   Hit. Set dirty bit because of the write. PA: `0x12ae`. LRUs: `3, 1, 4, 0, 7, 2, 5, 6`

4. `0x2332` (Write))

   Miss. Map VPN `0x23` to next available free page which is PPN `0x18` and set valid and dirty. PA: `0x1832`. LRUs: `4, 2, 5, 1, 0, 3, 6, 7`

5. `0x20ff` (Read)

   Hit. PA: `0x12ff`. LRUs: `4, 2, 5, 0, 1, 3, 6, 7`

6. `0x3415` (Write)

   Miss and replace last entry. Map VPN `0x34` to the last free page which is `0x19` and set valid and dirty bit. PA: `0x1915`. LRUs `5, 3, 6, 1, 2, 4, 7, 0`

3 Past Exam Question (SU25 Final Q8)

3.1 For the following system:

- 64 GiB physical memory
- 128 GiB virtual memory
- 1 MiB pages
- 32-bit page table entries

Answer the following questions.

(a) How many physical addresses are there? Express as a power of 2.

2^{36}

64 GiB = $64c \cdot 2^{30} = 2^6 c \cdot 2^{30} = 2^{36}$ addresses.

(b) How many bits are in the VPN, PPN, and page offset?

VPN: 17 bits

PPN: 16 bits

Offset: 20 bits

Page size = 1 MiB = $2^{20} \longrightarrow$ offset = 20 bits

Virtual memory = $128c \cdot 2^{30} = 2^7 c \cdot 2^{30} = 2^{37}$ VPN = $37 - 20 = 17$ bits

Physical memory = $64c \cdot 2^{30} = 2^{36}$ PPN = $36 - 20 = 16$ bits

(c) How many entries are in the page table?

2^{17} entries

Number of entries = number of VPNs = 2^{17}.

(d) How many pages are needed to store the page table?

1 page

Each PTE = 4 bytes $\longrightarrow$ page holds $\frac{2^{20}}{2^2} = 2^{18}$ entries.

We need 2^{17} entries $\longrightarrow$ fits within one page.

3.2 For the remaining questions, assume:

- 24-bit VPN
- 20-bit PPN
- 12-bit offset
- 64-bit PTEs

The TLB and first few page table entries are shown below.

PTE Format (64 bits)

63	62–20	19–0
Valid	**Status Bits**	**PPN**

TLB

Valid	VPN	PPN
1	0x000000	0x0061C
0	0x000001	0x00700
0	0x000002	0x10502
0	0x000003	0x01620

Page Table

0x BE51 C1A5 5150 061C
0x C41B B100 0000 1620
0x 061C 051A FF04 A111
0x C41B B100 0001 5100
...

For each virtual address below:

- Translate it to a physical address
- Indicate whether it is a:
  - TLB Hit
  - TLB Miss & Page Table Hit
  - Page Fault

Assume each access is independent. You can also assume that there is an implicit leading 0 for virtual address.

(a) `0x00000C16`

Physical Address: `0x0061CC16`

Result: TLB Hit

VPN matches valid TLB entry ⟶ direct translation.

(b) `0x0000200F`

Physical Address: `0x0B1A400F`

Result: Page Fault

Not in TLB and invalid in page table ⟶ page fault.

(c) `0x00003505`

Physical Address: `0x15100505`

Result: TLB Miss & Page Table Hit

Miss in TLB but valid entry in page table.