Fall, 2022
Week 16
2022.12.26

組別：＿＿＿＿＿　簽名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

[group14]

1.

For each combination of cache, TLB, and Page Table hit/miss, denote if it is possible. If not, explain why.

Cache	TLB	Page Table	Possible? Why not?
Miss	Miss	Miss	TLB miss and is followed by a page fault; after retry, data miss in cache.
Miss	Miss	Hit	TLB miss but entry found in page table; after retry, data miss in cache.
Miss	Hit	Miss	impossible; not in TLB if page not in memory.
Miss	Hit	Hit	Yes, but page table never checked if TLB hits
Hit	Miss	Miss	impossible; not in cache if page not in memory.
Hit	Miss	Hit	TLB miss, but entry found in page table; after retry, data in cache.
Hit	Hit	Miss	impossible; not in TLB if page not in memory.
Hit	Hit	Hit	Yes

[group1]

2.

Please select the correct options and explain the incorrect ones.

- ✓ a. When a page fault happens, hardware cannot do anything, but software can handle the fault instead.
- ✗ b. Higher associativity reduces access time but also increases complexity, cost, and miss rate.
- ✓ c. If a page is not in memory, it is impossible to find its data in the cache or TLB.
- ✗ d. TLB can only use fully associative page replacement.
- ✗ e. In a virtual memory system, page's write policy uses write-through to write data.
- ✗ f. Page table entries store the virtual page number and are indexed by the physical frame number.
- ✓ g. We can utilize bound register to limit the size of page table
- ✗ h. Page fault means that page is not resident in cache.

[group9]

3.

Memory 中常見的三種 miss 來源與其原因該如何配對?

(1): Compulsory misses — B: First access to a block
(2): Capacity misses — C: Due to finite cache size
(3): Conflict misses — A: Due to competition for entries in a set

[group4]

4.

Please choose the right answer.

a. It is possible that page table miss but TLB hit.

√ b. It is called compulsory miss that we first access a block in cache.

dirty

√ c. When page is written, we need to note it for page replacement. The bit we use is called reference bit.

√ d. We have the same offset in virtual address and physical address, it means that the page size and the frame size is the same.

√ e. Because the page table size is too large, we can use inverted page table、multilevel page table to improve this problem.

f. If TLB is miss, the cache must be miss, too. TLB miss 有可能 memory 或 cache 有资料

g. We use write through when a page is written.

write-back

[group10]

5.

Choose the correct options and explain it if it is wrong.

√ 1. The access to the page table is slow. So, using bounds registers is one of the solutions to limit table size.

2. Principle of locality: Programs use a whole part of their memory space frequently.

√ 3. Virtual memory technique treats the main memory as a fully-set associative write-back cache.

4. Only load/store operations need to access the page table.

fetch

5. In virtual memory, only write-through is feasible, given the latency of disk write.

6. To reduce page fault rate, we prefer least-recently used (LRU) replacement. We could know that the page has not been used recently by the value of its dirty bit (= 0).

reference

[group11]

6.

Select the correct options, and provide your reason for why it is right or wrong.

1. Hardware can detect the page fault and remedy the situation.
✓ 2. It is possible that TLB is missed but the page table is hit. However, it is impossible that TLB is hit and the page table is missed.
3. Page fault has a small miss penalty, so we don't have to reduce page fault rate.
4. Page table stores placement information in disk. in memory
✓ 5. On a TLB miss, the target data might be in the cache.
✓ 6. When we implement LRU replacement, we should clear the reference bit to 0 periodically by OS.

[group13]

7.

The memory architecture of a machine X is summarized in the table.

Virtual Address	54 bits
Page Size	16 K bytes
PTE Size	4 bytes

entry 记录 page 资讯

(a) Assume that there are 8 bits reserved for the operating system function (protection, replacement, valid, modified,…) other than required by the hardware translation algorithm. Derive the largest physical memory size (in bytes) allowed by this PTE format. Make sure you consider all the fields required by the translation algorithm.

(b) How large (in bytes) is the page table?

(c) Assuming 1 application exists in the system and the maximum physical memory is devoted to the process, how much physical space (in bytes) is there for the application's data and code? physical memory

page size = $2^{10} \times 16 = 2^{10} \times 2^4 = 2^{14}$ bytes

(a) Physical page number = 32-8 = 24 bits. The largest physical memory size = $2^{24} \times 16K$ bytes = 256GB

(b) The virtual page number has 54-14 = 40 bits. The number of page table entries are 2^{40}. Each PTE has 4 bytes. So the total size of the page table is 2^{42} bytes which is 4 terrabytes.

(c) The application's page table has an entry for every physical page that exists on the system, which means the page table size is is $2^{24} * 4$ bytes. This leaves the remaining physical space to the process: $2^{24} * 16K - 2^{26}$ bytes = $2^{38} - 2^{26}$ bytes.