DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Computer Organization and Architecture (CS2600)

Total Time : 50 minutes **Quiz 2**
Max Marks : 25 **20-03-2024**

---

1. Answer all questions
2. Unless otherwise specified, all questions are related to RISC V architecture.
3. State clearly any assumptions that you make.
4. 1-mark for neatness.

---

1. Consider a 32-bit CPU with a direct-mapped, byte-addressable write-through cache memory of 32 KByte, having a cache line size of 64 bytes. **(9 marks)**

   (a) How many cache lines are present in this cache memory? (1 mark)
   Answer: Cache lines $= 32KB/64 = 2^{15-6} = 2^9 = 512$
   Grading: 0 or 1

   (b) What is the length of the tag bits in the 32-bit address? (1 mark)
   #bits in tag per cache line $= 32 - (9 + 6) = 17$
   Grading: 0 or 1

   (c) What are the total number of bits present in the cache? (2 marks)
   Answer: #bits of data in cache line $= 64 \times 8 = 2^{6+3} = 2^9$
   #valid bit $= 1$
   #Total number of bits $= 512 \times (512 + 17 + 1) = 2,71,360$ bits
   Grading: 1 for the formula; 1 mark for the answer

   (d) A program defines an array as follows `int T0[1024]` and then immediately reads each element of the array as follows:

   ```
   for(int i=0; i<1024; ++i) x ^= T0[i];
   ```

   If a cache hit takes 1ns and a cache miss takes 10ns to complete a load/store operation, what is the minimum time the program takes to access the entire array? (3 marks)
   Answer: #Number of elements per cache line = 64/sizeof(int) = 64/4 = 16
   Minimum execution time occurs when the array is aligned to 64 bytes.
   #Number of cache misses to read the entire array into cache is sizeof(T0)/16 = 4096/16 = 64
   #Thus, we have 64 cache misses and the remaining 1024 - 64 = 960 cache hits. Total memory access time is $64 \times 10 + 960 \times 1 = 1600$ns
   Grading: 1 mark for number of cache misses; 1 mark for cache hits; 1 mark for the answer.

   (e) The program then executes the function:

   ```
   memset(T0, 0, sizeof(T0));
   ```

   which sets all elements of `T0` to 0. How much time does the `memset` function take to execute memory operations? Assume no write buffer is present. (1 mark)
   Answer: Every store would result in a write back to RAM. Thus the `memset` would take $1024 \times 10 =$ 10240ns.
   Grading: 0 or 1

   (f) How would a write buffer improve the performance of the processor? (1 mark)
   Answer:With a write buffer, the CPU does not stall while the memory block updates the memory. This boots performance. ⟨⟨optional: For example, ideally, the entire memset function executes in 1024ns.⟩⟩
   Grading: part marks possible

2. Consider a 32-bit CPU with a 2-level paging unit with each page table entry is of 4Byte, and each page is of 4KByte. **(8 marks)**

   (a) How many bits of address are needed to index into a second-level page table? (1 mark) Number of bits to index into a second level page is 4KB/4 = 1024 -> 10-bits

   (b) What is the maximum amount of memory (in bytes) needed to store the page tables for a single program? (1 mark)
   First level page table has 2^{10} entires. Thus, there are 2^{10} second level pages. The total memory required to store all page tables is $4096 \times (2^{10} + 1)$

(c) A program defines an array as follows `int T0[16384]` and then reads each element of the array as follows:

```
for(int i=0; i<15360; ++i) x ^= T0[i] ^ T0[i+1024];
```

i. What is the maximum number of memory accesses needed for the array accesses during the execution of the `for` loop?
Each iteration makes two accesses to the array. If there is in no TLB, each access would require 3 memory operations. Two for converting from virtual address to physical address, the third for the actual memory operation. Thus, the number of memory accesses is 3x2x15360 = 92,160
Grading: 2 marks. 1/2 mark for TLB assumption. 1/2 mark for 3 memory operations, 1 mark for calculation.

ii. What is the maximum number of these memory accesses that can result in page faults? Its a 2-level page table. 2 page faults to load the page directory and a second level page table.
Every element of the array is loaded. There are 16384 elements. If the array is aligned to 4KB, it spans across 16 pages. Thus, a maximum of 16+2=18 page faults are obtained.
If the array is not-aligned to 4KB, it spans across 17 pages. Thus, a maximum of 17+2=19 page faults are obtained.
Grading: 3 marks for 1 mark for page dir and page table faults; 2 mark for non-alignment; 1 mark if only alignment is mentioned;

(5(2+3) marks)

(d) Suppose the CPU designer wants to introduce a TLB in the processor. What is the minimum size of the TLB that best optimizes the `for` loop described above? Ignore all other memory accesses. (1 marks)
At any given iteration, the program accesses at-most 2 pages. Thus, a TLB of size 2 would provide optimal result.

3. Match the elements in Table 1, based on what the LHS is **most related to**. Each entry on the LHS uniquely matches an entry in the RHS. **(5 marks)**

Table 1: Match the Following

(A) `mpp`	(a) is most related where a `program is interrupted`
(B) `mepc`	(b) is most related to `privilege modes`
(C) `mtvec`	(c) is most related to the `source of the interrupt`
(D) `mcause`	(d) most related to `interrupt service routines`
(E) `mret`	(e) most related to `resuming execution`

(A) → (b)
(B) → (a)
(C) → (d)
(D) → (c)
(E) → (e)

4. An OS has a system call defined as follows:

```
int my_read(int fd, char *buf, int nbytes);
```

Write a RISC V assembly code snippet to explain how this system call is realized in assembly in the user program and explain. **(3 marks)**

0.5 marks for fd, buf, nbytes in a0, a1, a2 resp. 0.5 marks for a7 holding system call number; 0.5 marks for ecall; 0.5 marks for return; 1 marks for clarity in explaining.