

## Homework 7

Chinese Name: \_\_\_\_\_

Pinyin Name: \_\_\_\_\_

Student ID: \_\_\_\_\_

E-Mail (omit "@shanghaitech.edu.cn"): \_\_\_\_\_

**10** 1. Put T (True) or F (False) for each of following statement. [2 points each]

- (1) ( ) The size of the virtual address space accessible to the program cannot be larger than the size of the physical address space.
- (2) ( ) There is no fragmentation of physical memory while using virtual memory.
- (3) ( ) Freeing applications from having to manage a shared memory space, granting the ability to share memory used by libraries between processes, and increased security due to memory isolation are all benefits of using virtual memory.
- (4) ( ) The two users will not use any same part of physical memory, as they have separate virtual memory spaces.
- (5) ( ) Page table walk refers to the behavior of the TLB miss handler of the MMU, system firmware, or operating system to look up the address mapping in the page table to see if there is a mapping when the TLB misses.

**23** 2. **Page Table Calculations [23 points]**

Assume we have a computer with 16KB pages, 32-bit virtual addresses, and 32-bit PTEs (8 bits are reserved for protection and valid bit). We use two-level hierarchical page tables to manage virtual address and the machine is byte-addressable.

- 7** (a) For this computer, how many virtual pages can be addressed per process? [7 points]
- 8** (b) What is the maximum size of the physical memory that can be supported by this computer? Tips: the length of physical address is not restricted to 32 bits. [8 points]
- 8** (c) Suppose that a running program is currently using 300 MB of memory. What is the smallest possible number of PTEs and PTPs that must be valid in the page table(s) of this program? [8 points]



22 4. **Virtual Memory and TLB [22 points]**

Li Hua creates a machine which is byte-addressed with 20-bit virtual address and 16-bit physical address. The processor manual only specifies that the machine uses a 3-level page table with the following virtual-address breakdown.

L1 Index	L2 Index	L3 Index	Page Offset
4 bits	4 bits	4 bits	8 bits

8 (a) **Physical Address [8 points]**

What is the page size of Li Hua's machine?

\_\_\_\_\_

How many bits do physical page number (PPN) and page offset need in physical address, respectively?

PPN: \_\_\_\_\_ bits

Page offset: \_\_\_\_\_ bits

14 (b) **TLB [14 points]**

Li Hua executes the following snippet of code on his new processor. Assume `sizeof(int)` is 4, and the array elements are mapped to virtual addresses 0x6000 through physical address 0x1FFC. Assume array and sum have been suitably initialized.

```

1 int List[4096] = {0};
2 for (i = 0; i < 2; i++) {
3     for (j = 0; j < 8; j++) {
4         sum += List[j * 512]
5     }
6 }
```

The processor manual states this machine has a TLB with 16 entries. Assume that variables `i`, `j` and `sum` are stored in registers, and ignore address translation for instruction fetches; only accesses to array require address translation.

In the end, how many misses from the TLB and total memory accesses will Li Hua observe (Consider only the presence of the TLB, ignore other data cache such as L1D or L2 cache. Disregarding the effect of the initialization of the array in the line 1 on the TLB):

1. The TLB is direct-mapped

Misses from the TLB: \_\_\_\_\_

Total memory accesses: \_\_\_\_\_

2. The TLB is fully-associative (assume LRU replacement policy)

Misses from the TLB: \_\_\_\_\_

Total memory accesses: \_\_\_\_\_

**25** 5. **Page Table Walk [25 points]**

Consider a system which uses a two-level page-based virtual memory system.

- Page size is 16 bytes
- PTE size is 4 bytes
- Memory is byte-addressable
- The system is initialized with only the base page table allocated
- Physical pages are allocated from lower to higher PPNs incrementally. Note: all allocation follows this rule, including but not limited to user data and PTE.
- The base page table is architecturally mandated to be at physical address 0x00, so a PTE containing value 0x00 is effectively an “invalid” PTE (no valid bit is necessary)
- The PTE is entirely reserved for a PPN (no valid, status, or permission bits)

**6**

- (a) 1. Fill in the blank b with the corresponding index size and offset. Show your intermediate steps.[6 points]

L1 Index	L2 Index	Page Offset
bits	bits	bits

19

- (b) First write the value 0x2E to the virtual address 0x64, then write the value 0x94 to the virtual address 0xC8 and fill in the contents of the physical memory. [19 points]

Table 4: Memory State

Address (PA)	Value (From Lower Address to Higher Address)
0x00	
0x04	
0x08	
0x0c	
0x10	
0x14	
0x18	
0x1c	
0x20	
0x24	
0x28	
0x2c	
0x30	
0x34	
0x38	
0x3c	
0x40	
0x44	
0x48	
0x4c	

Example for *From Lower Address to Higher Address*: If you were to write 0x00 to 0x50, 0x00 to 0x51, 0xe1 to 0x52, and 0x00 to 0x53, then what we expect to see is 0x0000e100 (case insensitive). In another word, keep leading and trailing 0.