Homework 7

Chinese Name: 张钰晴

Pinyin Name: zhangyuqing

Student ID: 2022533170

E-Mail (omit "@shanghaitech.edu.cn"): zhangyq2022

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

(1) (F) The size of the virtual address space accessible to the program cannot be larger than the size of the physical address space.

(2) (F) There is no fragmentation of physical memory while using virtual memory.

(3) (T) 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) (F) The two users will not use any same part of physical memory, as they have separate virtual memory spaces.

(5) (T) 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]

page size = $16 \times 1024 = 2^{14}$ bytes

$32 - 14 = 18$

2^{18} virtual pages

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]

$32 - 8 = 24$ bits for physical page number.

each page of size 16kB

$2^{24} \times 16 \times 1024 = 2^{38}$ bytes. $= 256$TB

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]

300MB $= 300 \times 2^{20}$ bytes

number of pages $= \frac{300 \times 2^{20}}{2^{14}} = 19200$

$32 - 14 = 18$ bits used for 2 levels of page table index

∴ number of valid second-level PTEs ≥ 19200

Suppose in the 18 bits, 9 bits for first-level and 9 bits for second-level. Each page table 2^9 entries.

∴ number of first-level PTPs $\geq \left\lceil \frac{19200}{512} \right\rceil = 38$

∴ 19200 PTEs, 38 PTPs

20 3. **TLB Replacement [20 points]**

A processor has 16-bit address, 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 is the most recent). At some time instant, the TLB for the current process is the initial state given in the Table 2. Assume that all current page table entries are in the initial TLB. Assume also that all pages can be read from and written to.

Fill in the final state of the TLB in Table 3 according to the access pattern in Table 1. When needed, the page fault handler will allocate free physical pages using the following order: 0x17, 0x18, 0x19.

Table 1: Access Pattern for Memory

No.	Access Pattern
1	Write 0x2132
2	Read 0x12F0
3	Write 0x2032
4	Write 0x1104
5	Read 0x20AC
6	Write 0x1016
7	Read 0xAC08
8	Write 0x1216

Table 2: Initial TLB

VPN	PPN	Valid	Dirty	LRU	
0x01	0x11	1	1	0	x
0x00	0x00	0	0	6	x
0x10	0x13	1	1	1	✓
0x20	0x12	1	0	5	✓
0x00	0x00	0	0	7	x
0x11	0x14	1	0	4	✓
0xAC	0x15	1	1	2	✓
0xFF	0x16	1	0	3	x

Table 3: Final TLB

VPN	PPN	Valid	Dirty	LRU
0x21	0x17	1	1	5
0x12	0x18	1	1	0
0x20	0x12	1	1	3
0x11	0x14	1	1	4
0x10	0x13	1	1	2
0xAC	0x15	1	1	1
0xFF	0x01	1	0	7
0x01	0x16	1	1	6

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?

$2^8 = 256$ bytes

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

PPN: 8 bits

Page offset: 8 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 }
```

List 0, 512, 1024 ...

0, 512 ...

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: 8

   Total memory accesses: 29

60 M M M
68 H M
70 M M
78 H M
80 M M
88 H M
90 M M
98 H M

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

   Misses from the TLB: 8

   Total memory accesses: 29

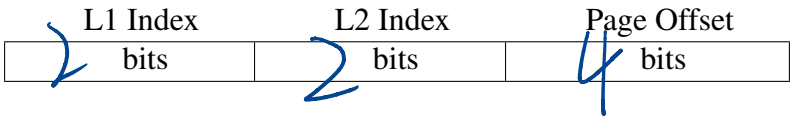
60
68
70
78
80
88
90
98

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 page table at 0x00

L2 0x10 0x30.

Data. 0x20 0x40

0x2E 0x24 0x48

0x24 0x48

0110 1100

1 2

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	0x00000010
0x04	0x00000000
0x08	0x00000000
0x0c	0x00000020
0x10	0x00000000
0x14	0x00000000
0x18	0x00000020
0x1c	0x00000000
0x20	0x00000000
0x24	0x2E000000
0x28	0x00000000
0x2c	0x00000000
0x30	0x00000040
0x34	0x00000000
0x38	0x00000000
0x3c	0x00000000
0x40	0x00000000
0x44	0x00000000
0x48	0x94000000
0x4c	0x00000000

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.