Computer Architecture I

Homework 8 Virtual Memory

Name (Pinyin): Chen Yucong
Email (Prefix): chenyc

Instructions:

Homework 8 covers the content of virtual memory, please refer to the lecture slides. You can print
it out, write on it and scan it into a pdf, or you can edit the PDF directly, just remember: you must
create a **PDF** and upload it to the **Gradescope**.

Please assign the questions properly on Gradescope,otherwise you will lose 25% of points.

Question Set 1. Short answer questions [10 points]

(a) What are 3 specific benefits of using virtual memory? [6 points]

1. Adding Disks to Hirachy
2. Simplify Memory for Apps
3. Protection between Processes

(b) True / False: [4 points]

1. The virtual and physical page number must be the same size. False
2. The virtual and physical page must have the same page size. True
3. Page tables make it possible to store the pages of a program non-contiguously. True
4. Page tables should be kept in CPU registers. False

Question Set 2. Calculation I [30 points]

(Show progress, worth 50% pts)

The virtual memory system is single-processor, single-core computer with

1. 4 KiB pages
2. 2 MB virtual address space
3. 2 GB physical address space.

The computer has a single-level TLB that can store 4 entries.
You may assume that the TLB is fully-associative with LRU replacement policy.

(a) Given a virtual address, how many bits are the Virtual Page Number and Offset? (Hint: Think of virtual address space) [15 points]

$\begin{cases} 2MB = 2^{21} \text{ bytes} \Rightarrow \#\text{bits of VA} : 21 \\ 4KiB = 2^{12} \text{ bytes} \\ 21-12 = 9 \end{cases}$

$\Rightarrow$ #bits of virtual page number in VA = 9

$\Rightarrow$ #bits of offset in VA =12

(b) Given a physical address, how many bits are the Physical Page Number and Offset? (Hint: Think of physical address space) [15 points]

$\begin{cases} 2GB = 2^{31} \text{ bytes} \Rightarrow \#\text{bits of PA} : 31 \\ \text{the offset in PA is with the same size as VA} \\ 31-12 = 19 \end{cases}$

$\Rightarrow$ #bits of physical page number in PA = 19

$\Rightarrow$ #bits of offset in PA =12

Question Set 3. Calculation II [30 points]

(Show progress, worth 50% pts)

The virtual memory system is single-processor, single-core computer with

1. 4 KiB pages
2. 28 bits virtual address
3. 16 MB physical memory

The computer has a single-level TLB that can store 16 entries.
You may assume that the TLB is fully-associative with LRU replacement policy.

(a) Given a virtual address, how many bits are the Virtual Page Number and Offset? [15 points]

#bits of VA: 28 bits
4KiB = 2^{12} bytes
28 − 12 = 16

⇒ #bits of virtual page number in VA = 16

⇒ #bits of offset in VA =12

(b) Given a physical address, how many bits are the Physical Page Number and Offset?
[15 points]

16MB = 2^{24} bytes ⇒ #bits of PA: 24
the offset in PA is with the same size as VA
24 − 12 = 12

⇒ #bits of physical page number in PA = 12

⇒ #bits of offset in PA =12

Question Set 4. TLB [30 points]

The virtual memory system is single-processor, single-core computer with

1. 256 byte pages
2. 16 bits addresses
3. an 4-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, and 7 being the least recent. We use decimal to represent it.

At some time instant, the TLB for the current process is the initial state given in the table below. Assume that all current page table entries are in the initial TLB and all pages can be read from and written to.

(a) Fill in the final state of the TLB according to the access pattern below. [20 points]

Free Physical Page: 0×17, 0×19

Access:

1. 0×01f0 (Read) hit, 1, 0 ; (0, 3, 1, 2)
2. 0×1301 (Write) miss, 1, 1 ; (1, 0, 2, 3), Allocate 0x17
3. 0×21ae (Write) miss, 1, 1 ; (2, 1, 3, 0), Allocate 0x19
4. 0×20ff (Read) miss, 1, 0 ; (3, 2, 0, 1)
5. 0×20ff (Write) hit, 1, 1 ; (3, 2, 0, 1)

Initial TLB

VPN	PPN	Valid	Dirty	LRU
0x01	0x11	1	0	0
0x00	0x00	0	0	3
0x10	0x13	1	1	1
0x20	0x12	1	0	2

Final State of TLB

VPN	PPN	Valid	Dirty	LRU
0x01	0x11	1	0	3
0x13	0x17	1	1	2
0x20	0x12	1	1	0
0x21	0x19	1	1	1

(b) Short answer questions [10 points]

```
//Let src, dst be char*
//10 < strlen(dst) <= strlen(src)
int random[10];
```

```
for(int i=0;i < 10; i++){
random[i] = rand();
}// rand is initialized with random unsigned ints.
for(int j = 0; j < 10; j++){
dst[rand[j]] = src[rand[j]];
}
```

Assuming all of code fits in 1 page, TLB currently has a pointer to the code, the strings are page-aligned (starting on a page memory). You do not need an TLB entry for the address translation of **random**.

How many page faults would occur?

1. In the best case. 0
2. In the worst case. 20