CSE 2431 **HOMEWORK 3 SOLUTION** **SPRING 2025**

Instructor: Luan Duong, Ph.D.

Electronic Submission: 11:59 pm, Monday Mar 24, 2025

Point: 25 points (5% of total grade)

Notes:

K = 2^{10}; M=2^{20}; G=2^{30}; index always starts from 0 (e.g. page 0 is the first page)
All memory is byte addressable (unless otherwise stated in the question)
If a number has no special mark (i.e. 3251), it is a decimal number.
If a number ends with 'b' (i.e. 01100000b), it is a binary number.
If a number starts with '0x' (i.e. 0xFF10), it is a hexadecimal number.

Question 1: Paging [11 points]

Let's do paging again, agents! Suppose 16-bit addresses are used for both virtual address and physical address. Suppose page/frame size is 256 bytes.

1) How many bits are used for page number and how many bits are used for the offset? [0.5 point] **8 bits (Because $2^8 = 256$)**

2) What is the maximum number of pages a process can have? [0.5 point] Also **256**.

3) Suppose each entry in the page table takes 4 bytes (including frame number, valid bit, and some other bits). Suppose an OS uses an array to store the page table. What is the size of a page table? [0.5 point] **4 x 256 = 1KB**

4) Suppose the first 3 pages of a process are mapped to frames 100 to 102, and the last 3 pages of the process are mapped to frames 5 to 7. All other pages are invalid. Draw the page table (include valid bit and frame number). [2 points]

VPN	Valid	Frame
0 or 0x00	1	100 or 0x64
1 or 0x01	1	101 or 0x65
2 or 0x02	1	102 or 0x66
3 or 0x03	0	---
…	0	---
252 or 0xFC	0	---
253 or 0xFD	1	5 or 0x05
254 or 0xFE	1	6 or 0x06
255 or 0xFF	1	7 or 0x07

5) Translate the following virtual addresses to physical addresses (show how you get the answers): [2 points] Note: same as your mid-journey test, you do not need to convert them to decimal numbers. Use the simplified calculation.

a) 0x01CC:

Page number **0x01** → Frame number **0x65**, offset **0xCC**, so PA = **0X65CC**

b) 0x02FF

Page number **0x02** → Frame number **0x66**, offset **0xFF**, so PA = **0X66FF**

c) 0x0301

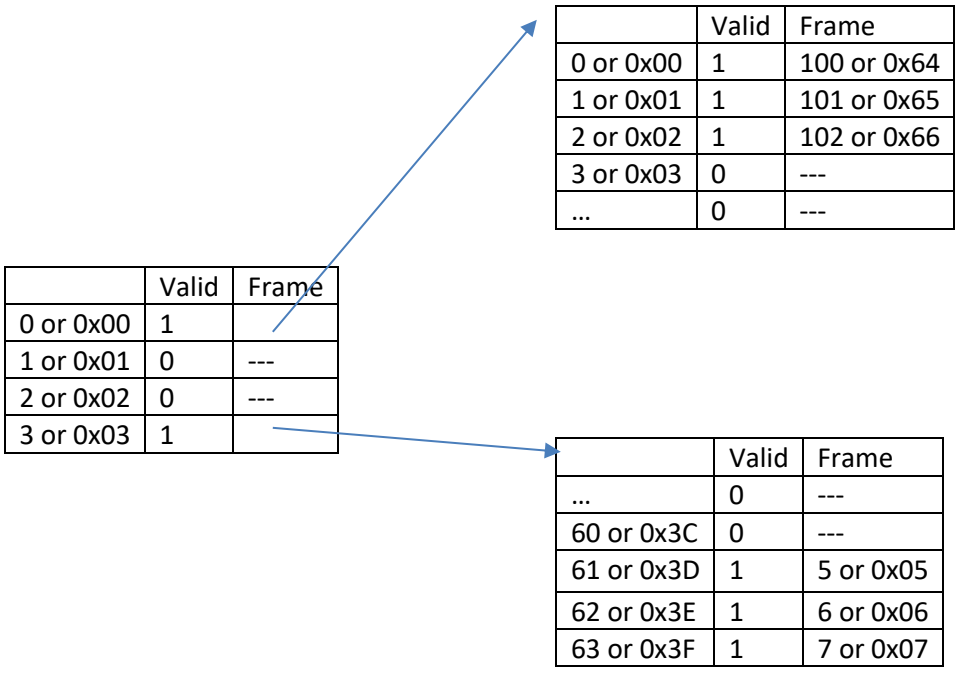
Page number **0x03** → Invalid. Report error.

d) 0xFF01

Page number **0xFF** → Frame number **0x07**, offset **0x01**, so PA = **0X0701**

6) Now suppose the OS is using two-level page table. Draw the page table. Suppose frames 8 to 99 are free so that you can allocate space for the page table there. Also suppose the page table directory always takes several consecutive full frames (e.g. If it is 2 bytes, it takes one frame; if it is 257 bytes, it takes 2 consecutive frames.) [3 points]

Each page of page table can hold 256/4 = 64 entries. The first page of page table holds entries for pages 0-63, so it must be allocated. The second page (64-127) and the third page (128-191) do not have valid entries, so they don't have to be allocated. The final page (192-255) must be allocated.



	Valid	Frame
0 or 0x00	1	
1 or 0x01	0	---
2 or 0x02	0	---
3 or 0x03	1	

	Valid	Frame
0 or 0x00	1	100 or 0x64
1 or 0x01	1	101 or 0x65
2 or 0x02	1	102 or 0x66
3 or 0x03	0	---
…	0	---

	Valid	Frame
…	0	---
60 or 0x3C	0	---
61 or 0x3D	1	5 or 0x05
62 or 0x3E	1	6 or 0x06
63 or 0x3F	1	7 or 0x07

Then let's actually put this page table in memory. The directory takes 4x4 bytes = 16 bytes, which is smaller than a frame, so it will take one frame. We can put directory in frame 8, two other pages in frames 9 and 10. Of course other allocation is also possible. Now the memory layout will look like:

	Valid	Frame
0 or 0x00	1	9 or 0x09
1 or 0x01	0	---
2 or 0x02	0	---
3 or 0x03	1	10 or 0x0A

Frame 8

Empty space

	Valid	Frame
0 or 0x00	1	100 or 0x64
1 or 0x01	1	101 or 0x65
2 or 0x02	1	102 or 0x66
3 or 0x03	0	---
…	0	---

Frame 9

	Valid	Frame
…	0	---
60 or 0x3C	0	---
61 or 0x3D	1	5 or 0x05
62 or 0x3E	1	6 or 0x06
63 or 0x3F	1	7 or 0x07

Frame 10

7) What is the size of this two-level page table? [0.5 point]

Since we have to fit a page directory into a whole frame, so just do not care how much space left for that frame. Thus: **Size = 3 pages x 256bytes/pages = 768 bytes**

8) Repeat 5) on the two-level page table and show how you get the answers. [2 points]

For the 16-bit address, last 8 bits are offset, first 2 bits are PDIndex, and middle 6 bits are PTIndex

a) 0x01CC: offset=0xCC, PDIndex = 0, PTIndex = 1, so will go to the second entry of page 9, so frame number is 0x65. VA = 0x65CC.

b) 0x02FF: offset=0xFF, PDIndex = 0, PTIndex = 2, so will go to the third entry of page 9, so frame number is 0x66. VA = 0x66FF.

c) 0x0301: offset=0x01, PDIndex = 0, PTIndex = 3, so will go to the fourth entry of page 9, which is invalid. Report error.

d) 0xFF01: offset=0x01, PDIndex = 11b = 3, PTIndex = 111111b = 63, so will go to the last entry of page 10, so frame number is 0x07. VA = 0x0701.

Two-level page table does not change translation result, which is as expected.

Question 2: Page Replacement Policy [7.2 points]

Agents, please fill in the blank by tracing the following Paging Algorithms and determine if the page fault happens for each page reference. Suppose you only have 3 frames in your main memory. If there is a tie, choose the **lowest-numbered page**.

Page Reference Stream: 4, 2, 3, 4, 1, 3, 2, 4, 5, 4, 3, 2

FIFO Algorithm [2.4 points]												
	4	2	3	4	1	3	2	4	5	4	3	2
Frame 1	**4**	4	4	4	**1**	1	1	1	1	1	**3**	3
Frame 2	-	**2**	2	2	2	2	2	**4**	4	4	4	**2**
Frame 3	-	-	**3**	3	3	3	3	3	**5**	5	5	5
Page Faults?	Y	Y	Y	N	Y	N	N	Y	Y	N	Y	Y

Optimal Algorithm [2.4 points]												
	4	2	3	4	1	3	2	4	5	4	3	2
Frame 1	**4**	4	4	4	**1**	1	1	**4**	4	4	4	4
Frame 2	-	**2**	2	2	2	2	2	2	**5**	5	5	5
Frame 3	-	-	**3**	3	3	3	3	3	3	3	3	**2**
Page Faults	Y	Y	Y	N	Y	N	N	Y	Y	N	N	Y

LRU (Least Recently Used) Algorithm [2.4 points]												
	4	2	3	4	1	3	2	4	5	4	3	2
Frame 1	4	4	4	4	4	4	2	2	2	2	3	3

Frame 2	-	2	2	2	1	1	1	4	4	4	4	4
Frame 3	-	-	3	3	3	3	3	3	5	5	5	2
Page Faults	Y	Y	Y	N	Y	N	Y	Y	Y	N	Y	Y

Question 3: **Disk Writing** [6.8 points]

For a **6,000-RPM** disk, assume its **average seek time** is 5 msec, its **average data transfer speed** is 100 MB/sec, and the **sector size** is 512 bytes. (In this problem, **assume** that 1 MB = 1 × 10^6 bytes; 1 msec = 0.001 sec; throughput means the amount of data transferred divided by time.) [Ref: Lecture 10, Slide 47]

$$T_{total} = T_{seek} + \frac{1}{2} \times T_{rot} + T_{transfer}$$

1) How long does it take to access **1MB** of contiguous data on average? What is the average throughput (bytes read per second) if the user accesses 1MB for each I/O read operation? Assume the data are perfectly aligned on the disk [3.4 points]

$$T_{rot} = \frac{min}{6000rev} \times \frac{60s}{1\,min} \times \frac{1000ms}{1s} = 10ms$$

$$T_{transfer} = \frac{1 \times 10^6 bytes.sec}{100 \times 10^6 bytes} \times \frac{1000ms}{1s} = 10ms$$

$$T_{total} = 5 + \frac{1}{2} \times 10 + 10 = 5 + 5 + 10 = 20ms$$

$$Throughput = \frac{1 \times 10^6}{20ms} \times \frac{1000ms}{1s} = 50MB/s$$

2) How long does it take to access **100 MB** of contiguous data (on average)? What is the average throughput (bytes read per second) if the user accesses 100 MB for each I/O read operation? Assume the data are perfectly aligned on the disk. [3.4 points]

$$T_{rot} = \frac{min}{6000rev} \times \frac{60s}{1\ min} \times \frac{1000ms}{1s} = 10ms$$

$$T_{transfer} = \frac{100 \times 10^6 bytes.sec}{100 \times 10^6 bytes} \times \frac{1000ms}{1s} = 1000ms$$

$$T_{total} = 5 + \frac{1}{2} \times\ 10 + 1000 = 5 + 5 + 1000 = 1010ms$$

$$Throughput\ = \frac{100 \times 10^6}{1010ms} \times \frac{1000ms}{1s} = 99.0099MB/s$$