CSSE 332 – Operating Systems: Virtual Memory Exercise

Name:_____ Grade:_____

How to convert a *logical* address to a *physical* address in a paging system

In a ***paging*** system, the process' space is divided into consecutive ***pages*** of fixed length:

Page 0, Page 1, Page 2, etc.

As such, a ***logical address*** (also called a ***virtual address***) consists of:

- A page number
- An offset within that page

Virtual address

Page #	**Offset**

In a paging system, main memory is divided into consecutive ***frames*** of fixed length (same length as the pages):

Frame 0, Frame 1, Frame 2, etc.

Some of the logical pages are placed in the physical frames. For example, pages 3, 8 and 10 might be placed in frames 104, 87 and 378, respectively, with the other pages of the process in virtual memory, *i.e.*, on disk (or other form of secondary memory).

The goal of this exercise is to help you understand 4 approaches to converting a logical address (page number + offset) to a physical address (frame number + offset, or a ***page fault*** if the page is not in main memory):

- ***Simple indexing***
- ***Multi-level indexing***
- ***Hashing*** into an ***inverted page table***
- ***Associative lookup*** in a ***cache*** called the ***translation lookaside buffer*** (***TLB***)

Real operating systems will often use a combination of these 4 approaches. In addition, there is an interaction with the main memory cache (see Figure 8.10 in your text).

Throughout this exercise, logical addresses and physical addresses are written as pairs of numbers (page number offset for logical addresses, frame number/offset for physical addresses). These are physically manifested as the high and low bits of a binary number.

1. (7 points) **Paging with simple indexing**: Connect the pieces of hardware below to show how each logical address is translated to a physical address or a page fault is detected.

Virtual address

Page #	**Offset**

+

Register

Page table pointer

Address	Main memory: Present bit	Main memory: Frame number
0	104	
1	334	
2	45	
3	891	
4	115	
…	…	…
256	0	333
257	0	228
258	0	610
259	0	200
260	1	324
261	1	900
262	1	1005
263	0	820
264	1	20
265	0	5
266	0	1005
267	0	220
268	1	4
269	1	303
270	1	689
271	0	446
272	1	848
273	0	666
274	1	111
275	1	229

Physical address

Frame #	**Offset**

Register

Interrupt indicator

Assumptions: Assume the value in the register for the page table pointer is 256.

Answer the following questions based on the above (completed) diagram.

Logical **2 105** causes a page fault? Yes or No (circle one). If no, converts to physical: _____

Logical **5 640** causes a page fault? Yes or No (circle one). If no, converts to physical: _____

Logical **12 320** causes a page fault? Yes or No (circle one). If no, converts to physical: _____

Consider a 32-bit addressing scheme with 18 bits for the page number and 14 for the offset.
How many entries in the page table? _____ What is the size of each page? _____

2. (7 points) **Multi-level indexing**: Connect the pieces of hardware below to show how each logical address is translated to a physical address or a page fault is detected.

Virtual address

Page #	**Offset**

Register

Root page table ptr

Address	Main memory: Present bit	Main memory: Pointer to page table block
0	1	104
1	1	334
2	0	45
3	1	1
4	1	115
5	1	713
6	1	2
7	0	228
…	….	….
	Present bit	Frame #
256	1	900
257	1	1005
258	0	820
259	1	20
260	0	5
261	0	1005
262	0	220
…	…	….
512	1	303
513	1	689
514	0	446
515	1	848
516	0	666
517	1	111
518	1	229

Physical address

Frame #	**Offset**

Register

Interrupt indicator

Assumptions: Assume the value in the register for the root page table pointer is 0. Suppose that the page # is divided into 4 bits + 8 bits for a two-level page table. The page number for this exercise is written in binary. Also, assume that each page (or page table block) has 256 bytes.

Answer the following questions based on the above (completed) diagram.

Logical **0010 00000010 105** causes a page fault? First lookup or Second lookup or Neither (circle one). If neither, converts to physical: _____

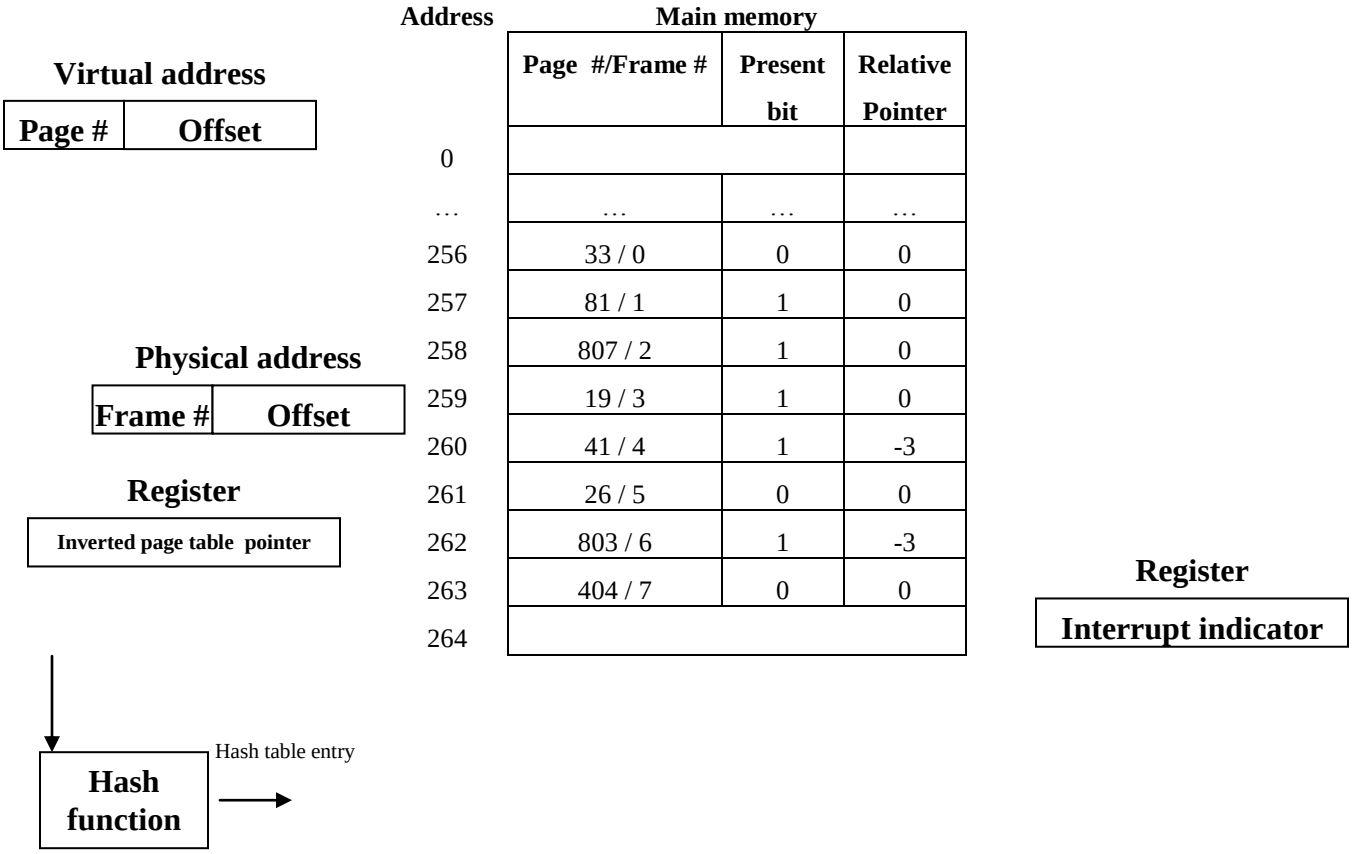
Logical **0110 00000010 240** causes a page fault? First lookup or Second lookup or Neither (circle one). If neither, converts to physical: _____

Logical **0011 00000011 120** causes a page fault? First lookup or Second lookup or Neither (circle one). If neither, converts to physical: _____

Consider a 32-bit addressing scheme with 6 + 12 bits for the page number and 14 for the offset. Assume a page table entry is 4 bytes. Assume byte-addressing :
# of entries in the root page table? ____Total entries in the page table? _____ Page size in bytes? _____

What is the primary advantage of using layers in an indexing scheme? Primary disadvantage?

3. (7 points) **Hashing into an inverted page table**: Connect the pieces of hardware below to show how each logical address is translated to a physical address or a page fault is detected. In this example, the hash function generates the **index to the frame table i.e. the frame number.**



Address	Page #/Frame #	Present bit	Relative Pointer
0			
…	…	…	…
256	33 / 0	0	0
257	81 / 1	1	0
258	807 / 2	1	0
259	19 / 3	1	0
260	41 / 4	1	-3
261	26 / 5	0	0
262	803 / 6	1	-3
263	404 / 7	0	0
264			

Main memory

Assumptions: Assume the hashing function is h(x) = 3 + (x % 8) and the inverted page table pointer value is 256.

Answer the following questions based on the above (completed) diagram.

Logical **81 105** causes a page fault? Yes or **No** (circle one). If no, converts to physical: __1, 105_

Logical **41 640** causes a page fault? Yes or No (circle one). If no, converts to physical: _____

Logical **25 320** causes a page fault? Yes or No (circle one). If no, converts to physical: _____

Logical **28 204** causes a page fault? Yes or No (circle one). If no, converts to physical: _____

Logical **19 880** causes a page fault? Yes or No (circle one). If no, converts to physical: _____

In the above table, what is the longest chain? ____ Shortest chain? _____

4. (7 points) **<u>Associative lookup in a cache called the translation lookaside buffer(TLB)</u>**:
Connect the pieces of hardware below to show how each logical address is translated to a physical address or a page fault is detected.

Virtual address

Page #	**Offset**

+

Register

Page table pointer

Address	**Main memory**	
	Present bit	Frame #
0	1	104
1	1	334
2	0	45
3	1	891
4	1	115
…	…	…
256	0	333
257	0	228
258	0	610
259	0	200
260	1	324
261	1	900
262	1	1005
263	0	820
264	1	20
265	0	5
266	0	1005
267	0	220
268	1	4
269	1	303
270	1	689
271	0	446
272	1	848
273	0	666
274	1	111
275	1	229

Translation lookaside buffer (TLB)	
16	848
53	100
4	324
80	226
5	900

Physical address

Frame #	**Offset**

Register

Interrupt indicator

<u>Assumptions</u>: Assume the value in the register for the page table pointer is 256.

Answer the following questions based on the above (completed) diagram.

Logical **4 105** is in TLB? Yes or No? Causes a page fault? Yes or No (circle one).
If no, converts to physical: _____

Logical **18 640** is in TLB? Yes or No? Causes a page fault? Yes or No (circle one).
If no, converts to physical: _____

Logical **16 105** is in TLB? Yes or No? Causes a page fault? Yes or No (circle one).
If no, converts to physical: _____

If there are *n* items in the TLB, is the TLB lookup time O(*n*) or O(1)? How much silicon is needed?

What is the primary advantage of this scheme?

5. (8 points) Summary (after doing the exercises on the following pages):

	Advantages	Disadvantages
Indexing		
Indexing with layers		
Hashing into an **inverted page table**		
Associative lookup in a cache called the **translation lookaside buffer (TLB)**		