Jong Park
10/26/2018
CS 472 – Computer Architecture
HW3

1. For a 4KiB page, and a 32 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 12 bytes. Show all calculations.

   Number of pages in a table = Total possible Logical Address Entries / Page size
   Total size of page table = Page Table Entry * Number of pages

   Page = 4KiB = 4,096 bytes = 2^{12} bytes
   Address space = 32bit
   PTE = 12 bytes
   max # of page = $2^{(32-12)}$ = 2^{20} = 1Mi pages
   Total size of a page table = 12 bytes * 2^{20} = **12 MiB**

2. For a 4KiB page, and a 64 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 12 bytes. Show all calculations.

   Page = 4KiB = 4,096 bytes = 2^{12} bytes
   Address space = 64bit
   PTE = 12 bytes
   max # of page = $2^{(64-12)}$ = 2^{52} = 4 Pi pages
   Total size of a page table = 12 bytes * 2^{52} = **48 PiB**

3. For a 16KiB page, and a 32 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 12 bytes. Show all calculations.
   Page = 16KiB = 16,384 bytes = 2^{14} bytes
   Address space = 32bit
   PTE = 12 bytes
   max # of page = $2^{(32-14)}$ = 2^{18} = 256 Ki pages
   Total size of a page table = 12 bytes * 2^{18} = 3,072KiB = **3 MiB**

4. For a 16KiB page, and a 64 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 12 bytes. Show all calculations.
   Page = 16KiB = 16,384 bytes = 2^{14} bytes
   Address space = 64bit
   PTE = 12 bytes
   max # of page = $2^{(64-14)}$ = 2^{50} = 1Pi pages
   Total size of a page table = 12 bytes * 2^{50} = **12 PiB**

5. Suppose the existence of an associative cache design with N = 2. Assume that the main memory uses a 32-bit address (and that each address maps to a single byte). If the cache scheme uses the following breakdown of the address bits:

Tag	Index	Offset
31-6 (26bits)	5-3 (3bits)	2-0 (3bits)

a) What is the cache block size (in words)?

$$3\,offsets = \frac{2^3\,bytes}{block} * \frac{word}{4\,byes} = \frac{8}{4}\,words$$ = **2 words**.

b) How many block indices does the cache have?

$$3\,bit\,index = \frac{2^3}{2\,sets}$$ = **4 rows of indices with 2 blocks on each indices**.

(4 rows * 2 sets) * 8 bytes = 64 possible entries with 32 entries maximum at given point.

c) Including space for the valid bits, tags, and actual block data, how many bits would be required to actually implement this hypothetical cache?

$$Total\,Cache = (no.\,of\,rows * no.\,of\,sets) * (valid\,bit + tag\,bits + data\,bits)$$

$$Total\,Cache = (4*2) * (1 + 26 + 8\,bytes * \frac{8\,bits}{1\,byte})$$ = **728 bits**.

6. Using the cache design from the previous problem, assume the following byte-addressed cache references are recorded. Assume that the cache is initially empty. Also assume that accesses occurred from left to right (e.g. address 0 was requested, then address 4, followed by address 16, etc).

Byte Address											
0	4	16	132	232	160	1024	30	140	3100	180	2180

a) What was the number of cache hits?

Address (binary)	Address	Block #	Index (mod4)	H/M
000 000 000	0	0	0	Miss
000 000 100	4	0	0	Hit
000 010 000	16	2	2	Miss
010 000 100	132	16	0	Miss*
011 101 000	232	29	1	Miss
010 100 000	160	20	0	Hit
10000 000 000	1024	128	0	Miss*
000 011 110	30	3	3	Miss
010 001 100	**140**	**17**	**1**	**Miss***
110000 011 100	**3100**	**387**	**3**	**Miss***
010 110 100	**180**	**22**	**2**	**Miss***
100010 000 100	**2180**	**272**	**0**	**Miss***

* means replaced

2 Hits
(Bold indicates the final state of the index)

b) Create a table to show the final state of the cache including the value of each index, the valid bits, and the tags.

Block Index	**Valid Bit**	**Least Recently Used**	**Set 0**		**Set 1**	
			Tag (Leading 0's are not shown)	**Data**	**Tag** (Leading 0's are not shown)	**Data**
000 (0)	Y	0	10000	Mem[1024]	**100010**	**Mem[2180]**
001 (1)	Y	0	11	Mem[232]	**10**	**Mem[140]**
010 (2)	Y	0	0	Mem[16]	**10**	**Mem[180]**
011 (3)	Y	0	0	Mem[30]	**110000**	**Mem[3100]**