Computer Architecture Homework 8

Spring 2022, May

Problem 1

Choose True / False:

1.A virtual memory system that uses paging is vulnerable to external fragmentation._____
2.One way to solve Compulsory miss is to increase the block size._____
3.In a bare system, addresses issued with loads/stores are real physical addresses other than virtual address._____
4.The size of the virtual address space accessible to the program cannot be larger than the size of the physical address space._____

Solution: F T T F

Problem 2

This question refers to an architecture using segmentation with paging. In this architecture, the 32-bit virtual address is divided into fields as follows:

3 bit segment number	13 bit page number	16 bit offset

Here is the relevant table (all values in hexadecimal):

Segment Table		Page Table A		Page Table B	
0	Page Table A	0	CAEF	0	C001
1	Page Table B	1	DEAB	1	D5AA
X	(rest invalid)	2	BFFE	2	A000
		3	AF11	3	BA09
		X	(rest invalid)	X	(rest invalid)

Find the physical address corresponding to each of the following virtual addresses (answer "bad virtual address" if the virtual address is invalid):

1.0x00000000 _____
2.0x20032003 _____
3.0x100205BD _____

Solution:
1.0xCAEF0000
2.0xBA092003
3.bad virtual address

Problem 3

In a 34-bit machine we subdivide the virtual address into 4 segments as follows:

8 bit	7 bit	7 bit	12 bit

We use a 3-level page table, such that the first 8-bit are for the first level and so on. Assume the size of each page table is equal to one page size. (Ignore the fragments and treat it roughly as one page)

Question 1. What is the page size in such a system?

Question 2. For a process with 256KiB memory starting from address 0, what is the size of the page table (3 layers in total)?

Question 3. What is the size of the page table(3 layers in total) for a process that has a code segment of 48KiB starting at address 0x1000000, a data segment of 600KiB starting at address 0x80000000 and a stack segment of 64KiB starting at address 0xf0000000 and growing upward ?

Solution:
1.4096 byte
2.
1 level_1 entry$\rightarrow 2^7 = 128$ level_2 entries
1 level_2 entry $\rightarrow 2^7 = 128$ level_3 entries
$256\text{k}/(2^{12}) = 64$ level_3 entries
1 level_2 entry, 1 level_1 entry
each page table is 4kb aligned
(1+1+1)*4kb = 12 kb
3.
$48\text{k}/(2^{12}) = 12$ level_3 entries
$600\text{k}/(2^{12}) = 150$ level_3 entries $\rightarrow$ 2 level_3 page $\rightarrow$ 1 level_2 page
$64\text{k}/(2^{12}) = 16$ level_3 entries
for example: 0x08000 0000 + 96000(600KiB) = 0x08009 6000
If the first 8 bits of the two addresses are different, which means that two leve2 tables are required. If the next 7 digits are different, which means that two leve3 tables are required.
$\text{level}_1 + \text{level}_2 + \text{level}_3 = (1 + 3 + 4)$*4kb = 32kb

Problem 4

A processor has 16-bit addresses, 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 being the most recent). 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. Assume also that all pages can be read from and written to. Fill in the final state of the TLB according to the access pattern below Free physical pages: 0x17, 0x18, 0x19.

1	write 0x776e
2	read 0x9796
3	write 0x9a0f
4	read 0x5a82
5	write 0x035b
6	read 0x0365

VPN	PPN	Valid	Dirty	LRU
0x6f	0x48	1	0	0
0x03	0x97	1	1	5
0x77	0x56	1	0	6
0x1f	0x2d	1	1	1
0x9a	0x9a	1	0	3
0x00	0x00	0	0	7
0xea	0x6d	1	1	2
0xc8	0x21	1	0	4

VPN	PPN	Valid	Dirty	LRU

Solution:

Table 1: 1

VPN	PPN	Valid	Dirty	LRU
0x6f	0x48	1	0	1
0x03	0x97	1	1	6
0x77	0x56	1	1	0
0x1f	0x2d	1	1	2
0x9a	0x9a	1	0	4
0x00	0x00	0	0	7
0xea	0x6d	1	1	3
0xc8	0x21	1	0	5

Table 2: 2

VPN	PPN	Valid	Dirty	LRU
0x6f	0x48	1	0	2
0x03	0x97	1	1	7
0x77	0x56	1	1	1
0x1f	0x2d	1	1	3
0x9a	0x9a	1	0	5
0x97	0x17	1	0	0
0xea	0x6d	1	1	4
0xc8	0x21	1	0	6

Table 3: 3

VPN	PPN	Valid	Dirty	LRU
0x6f	0x48	1	0	3
0x03	0x97	1	1	7
0x77	0x56	1	1	2
0x1f	0x2d	1	1	4
0x9a	0x9a	1	1	0
0x97	0x17	1	0	1
0xea	0x6d	1	1	5
0xc8	0x21	1	0	6

Table 4: 4

VPN	PPN	Valid	Dirty	LRU
0x6f	0x48	1	0	4
0x5a	0x18	1	0	0
0x77	0x56	1	1	3
0x1f	0x2d	1	1	5
0x9a	0x9a	1	1	1
0x97	0x17	1	0	2
0xea	0x6d	1	1	6
0xc8	0x21	1	0	7

Table 5: 6

VPN	PPN	Valid	Dirty	LRU
0x6f	0x48	1	0	5
0x5a	0x18	1	0	1
0x77	0x56	1	1	4
0x1f	0x2d	1	1	6
0x9a	0x9a	1	1	2
0x97	0x17	1	0	3
0xea	0x6d	1	1	7
0x03	0x19/0x97	1	1	0

Problem 5

Assume a computer has 32-bit addresses, 4KB pages, and the physical memory space is 4GB. The computer uses two-level paging, each page table entry consists of a next-level address index and seven additional control bits.

Question 1. What is the minimum number of bits per secondary page table entry? And justify your ans.

Question 2. What is the minimum number of bits per level 1 page table entry? And justify your ans.

Question 3. Assuming that each page table entry is 8 bytes in size (In addition to the next level address index and seven additional control bits, we have added some new things to expand its size to 8 bytes) and each page table is exactly 1 page in size, how many bits of virtual address does the program actually use? (Hint: It means that not all 32 bits are valid virtual addresses, and some bits may be useless.)

Question 4. According to question 3, how many bytes is the virtual address space of an application?

Solution:
1. 4GB/4KB= 2^{20} bits PPN. There are 4GB bytes of memory and 4KB bytes/page, which means there are 2^{20} pages in the machine. That means that the second-level page table has to have at least 20 bits (to be able to select the physical page). Then there are 7 further bits per page, so we need at least 20 + 7 = 27 bits per entry.

2. 27 bits, again. The reason is the same. The level 1 page table entries need to be able to "resolve" 2^{20} bits since a page table could conceivably live on any physical page.

3. 30 bits. A page table entry is 8 bytes, and a page table is 4KB (because it needs to fit on a page). Thus, a page table can have no more than 512 entries, which means that no more than 9 bits are used to index into the page table. Finally, since a page is 4KB, 12 bits of the address determine the offset into the page. Altogether, this is a maximum of 9 + 9 + 12 = 30 bits of address in use.

4. 2^{30}= 1GB