Computer Architecture Homework 8

Spring 2022, May

Problem 1 (10 points)

Choose True / False:

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

Problem 2 (10 points)

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 __0XCAEF0000__
2.0x20032003 __0XBA092003__
3.0x100205BD __bad virtual address__

Problem 3 (30 points)

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

offset

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. What is the size of a page table for a process that has 256K of memory starting at address 0?

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

① the page offset is 12bits, then the page size is

2^{12} bytes = 4KB

② The process needs 256K Byte of memory, And that is

256K/4K = 64 pages of memory.

and a single third level page can handle 2^7 = 128 pages,

So we need 1 first level page table,

1 second level page table

1 third level page table.

So the size is (1+1+1) × 4KB = 12KB

(2) • Code segment 48K. need 48K/4K = 12 pages.
So we need 1 third level page table for it.

• data segment 600K, need 600K/4K = 150 pages.
So we need 2 third level page table for it, since
1 third level page table can refer to 2^7 = 128 pages.

• Stack segment 64K. need 64K/4K = 16 pages.
So we need 1 third level page table for it.

• totally we need |1+2+1| = 4 third-level page table,
3 second level page table, 1 for each segment.
and 1 first level page table.

So the size is (1+3+1) × 4KB = 32KB

Problem 4 (20 points)

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
0x5a	0x97	1	1	5
0x77	0x56	1	0	6
0x1f	0x2d	1	1	1
0x9a	0x9a	1	0	3
0x97	0x00	0	0	7
0xea	0x6d	1	1	2
0x03	0x21	1	0	4

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	1	1	0

Problem 5 (30 points)

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?

① 4KB pages. So offset is 12 bits.

and the address is 32 bits.

So it needs $32-12=20$ bits index referring to pages

by adding control bits, we get the minimum PTE of L2 is

$20+7=27$ bits

② one PTE of L1 contains address referring to a L2 page table.

which is 32 bits, by adding control bits, we get the

minimum PTE of L1 is $32+7=39$ bits

Solution is 27 bits,

③ L1: 4KB/8B = 512 entrys. needs $\log_2 512 = 9$ bits.

L2: also 4KB/8B = 512 entrys. needs $\log_2 512 = 9$ bits.

So. virtual address size is $9+9+12=30$ bits.

④ $2^9 \times 2^9 \times 4KB = 2^{18} \times 2^2 KB = 2^{20} KB = 1GB$