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:

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?

Answer:
Page size = 2^{12} Bytes = 4 KB.

Question 2. What is the size of a page table for a process that has 256K of memory starting at address 0?

Answer:
Number of pages = $256 \times 2^{10} \div 2^{12} = 64$.
By the subdivided virtual address, one L1 PT entry indicates 2^7 L2 PT entries, one L2 PT entry indicates 2^7 L3 PT entries and one L3 PT entry indicates one memory page.
Therefore, 64 pages indicates 64 L3 PT entries, which indicates one L2 PT entry and thus indicates one L1 PT entry.
In conclusion, only one page table in each page level is needed, so the size is $3 \times 4 = 12$ KB.

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 ?

Answer:
As for code segment: Number of pages = $48 \times 2^{10} \div 2^{12} = 12$, indicating 1 L3 PT.
As for data segment: Number of pages = $600 \times 2^{10} \div 2^{12} = 150$, indicating 2 L3 PT.
As for stack segment: Number of pages = $64 \times 2^{10} \div 2^{12} = 16$, indicating 1 L3 PT.
Since the most significant 8 bits of base addresses are different, $2^{7+7+12} > 2^{32-8}$ and $2^{7+12} < 2^{32-8}$, the mapping range of a single L2 PT can only cover one segment, but that of a single L1 PT can cover all of the three segments. Therefore, at least 3 L2 PTs are required and only one L1 PT is required.
Considering the requirement of L3 PTs, there should be one L2 PT corresponding to the respective segment.
In conclusion, there are 1 L1 PT, 3 L2 PTs and 4 L3 PTs required for the given segments. So the size is $8 \times 4 = 32$ KB.

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
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
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	0x97	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..

Answer:
Number of pages $= 4 \times 2^{30} \div (4 \times 2^{10}) = 2^{20}$, so the address index in each L2 PT is 20-bits.
Therefore, the minimum number of bits per L2 PT entry is $20 + 7 = 27$ bits.

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

Answer:
The logic is the same as question 1. Each L1 PT also needs 20 bits to retrieve index, so the minimum number of bits per L1 PT entry is $20 + 7 = 27$ bits.

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.)

Answer:
Number of PT entries in one page $= 4 \times 2^{10} \div 8 = 2^9$.
Since there are two virtual page levels, totally $9 + 9 + 12 = 30$ bits of virtual addresses are actually used.

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

Answer:
Since the number of virtual pages is $2^9 \times 2^9 = 2^{18}$ and each virtual page is 4 KB in size, the overall virtual address space is $2^{18} \times (4 \times 2^{10}) = 2^{30}$ Bytes = 1 GB.