arm Education

Real Time Operating Systems Design and Programming

Homework

OS Memory

Contents

1 Overview

You are expected to work out the following questions once you have gone through the lecture. Please go over the lecture slides and any relevant textbook so that you can grasp the basic ideas behind memory management. The questions are set up and designed as examples to help you understand the concepts in more details.

2 Memory system setup for Questions

Consider the following memory system setup:

- Byte addressed
- 8-bit virtual address
- 128-byte physical memory space
- Single-level paging, 32-byte page size
- TLB with 2 entries, LRU is the replacement algorithm
- Virtual memory, clock is the replacement algorithm

Table 1 TLB

Page	Frame	Use	Dirty	Valid
				0
				0

Table 2 Frame Table (Clock)

Frame	Page	Clock Hand
0	000	Pointing to this frame
1	001	
2	010	
3	011	

Table 3 Page Table

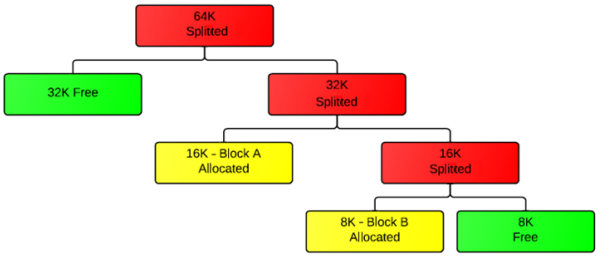
Page	Frame	Use	Dirty	Valid
0	00	1	0	1
1	01	0	1	1
2	10	0	0	1
3	11	1	0	1
4				0
5				0
6				0
7				0

3 Questions

1. What is temporal locality, give a specific example when memory hierarchy can harness temporal locality to improve performance.

2. What is spatial locality, give a specific example when memory hierarchy can harness spatial locality to improve performance.

3. The memory system applies the buddy memory allocation and you have the following memory setup.



   Try to allocate memory for the following memory request sequence:
   a. 7K, 4K, 15K
   b. 17K, 6K
   c. Release Block A, 9K, 5K, release Block B
   d. Release Block B, 9K, 14K

4. Assume the access to cache takes A time units and access to main memory takes B time units. What is the worst case memory access time for the following schemes? What is the best case memory access time for the following schemes? You can ignore the access time to base registers of page tables.
   a. Single level paging
   b. Single level paging with translation lookaside buffer
   c. Two-level paging
   d. Two-level paging with translation lookaside buffer in which the direct frame address is stored

5. What is the theoretically optimal algorithm for replacing a page? Why is it practically infeasible and what is the alternative way? To what extend does the alternative approach approximate the original algorithm?

6. Consider the memory setup given at the beginning of this document, answer the following question:
   a. What is the total addressable memory by the virtual address?
   b. What is the minimum disk capacity to support the virtual memory system?
   c. How many bits are used to address pages and why?

7. Based on the memory setup given at the beginning of this document, for each of the following successive memory accesses, show in Binary the new content of the page table, TLB, and frame table, and the position of the clock hand:
    a. write to the binary virtual address “00010000”
    b. read from the binary virtual address “01010000”
    c. write to the binary virtual address “10010000”

4 Answers

1. Currently accessed block of memory is more likely to be accessed again. A very noticeable example is the “i”, the counter variable in a loop.

2. Neighbour blocks of the currently accessed block of memory are more likely to be accessed again. For example, arrays are usually accessed in sequence. Given that they are allocated to a contiguous memory space, which is usually the case, pre-fetch part of the arrays will thus improve performance.

3. Assume the minimum size of a buddy block is 4K or smaller:
    a. Allocated 7K to the 8K free block; split the 32K free block into two 16K free blocks, split one of the 16K free blocks into two 8K free blocks, split one of the 8K free blocks into two 4K free block, allocate 4K to the one of the 4K free block; allocate 15K to the other 16K free block.
    b. Allocate 17K to the 32K free block; allocate 6K to the 8K free block.
    c. Release Block A (does not coalesce); allocate 9K to the just-released 16K block; allocate 5K to the 8K; release Block B (does not coalesce).
    d. Release Block B and coalesce with the other free 8K block to make a free 16K block; allocate 9K to the free 16K blocks; split the free 32K block into two 16K free blocks, allocate 14K to one of the free 16K block.

4. Actually the access time depends on a lot of factors, but in a simplified model, the worst and best memory access times for the scheme outlined in question 4 are (respectively) :
    a. 2B, one access to the page table (memory), then the frame (memory), access time for both cases are the same.
    b. 2B+A, one access to the TLB (cache), and one access to the page table (memory), then the frame (memory); A+B, one access to the TLB and then to the frame.
    c. 3B, one access to the L1 PT (memory), one access to the L2 PT (memory), then the frame (memory); access time for both cases are the same.
    d. 3B+A,one extra access to cache compared to c; A+B, one access to the TLB and then to the frame.

5. Replacing the least recently used (LRU) page is close to optimal. But the problem is that the OS will have to record the sequence of every access. A commonly used alternative way is the clock-ike algorithm, which periodically clears the use bits of the page. This means, only pages that have not been used since the last period will be cleared. Although it does not guarantee the page to be swapped out is the least recently used one, it will not swap out the most recently used page.

6. Related information: byte addressed, 8-bit virtual address, 128-byte physical memory, 32-byte page size:
    a. 8-bit can address up to 2^8 (256) bytes.
    b. 256 bytes – 128 bytes of physical memory = 128 bytes.
    c. Page size is 32(2^5) byte, 8-5=3, so the highest three bits are used to address the page.

7. As explained in Q6, the highest three bits are related to pages, so the access sequence is page 0 (a) then page 2 (b) and finally page 4 (c):
    a. The address 00010000 corresponds to page 0. Since the TLB is empty, the frame number should be retrieved from the page table. According to the page table, page 0 is resident in the memory (valid bit is one), so there is no need for page replacement, i.e., the frame table is left as it is. On the other hand, the TLB should be loaded with the indexed entry page/frame. The protection bits in the page table and the TLB should be updated accordingly. The following figure shows the new states of the tables.

Page	Frame	Use	Dirty	Valid
000	00	1	1	1
				0

Page	Frame	Use	Dirty	Valid
0	00	1	1	1
1	01	0	1	1
2	10	0	0	1
3	11	1	0	1
4				0
5				0
6				0
7				0

    b. The address (“01010000”) corresponds to page 2. Page 2 is not in the TLB. But it is resident in the memory (see page table valid bit). The tables are then updated like explained above in case (a) and are shown in the following.

Frame	Page	Clock Hand

0	000	
1	001	Pointing to this frame
2	010	
3	011	

Frame	Page	Clock Hand
0	000	
1	001	
2	010	Pointing to this frame
3	011	

Page	Frame	Use	Dirty	Valid
000	00	1	1	1
010	10	1	0	1

Page	Frame	Use	Dirty	Valid
0	00	1	1	1
1	01	0	1	1
2	10	1	0	1
3	11	1	0	1
4				0
5				0
6				0
7				0

Frame	Page	Clock Hand
0	000	
1	001	
2	010	
3	011	Pointing to this frame

c. The address (“10010000”) corresponds to page 4. No corresponding entry in the TLB. The page table should be checked instead. The entry with index 4 has a valid bit equal to zero, i.e. page 4 is NOT resident in the main memory. The clock algorithm routine is called to replace a resident page and load page 4 into the memory. The states of the table go through the following intermediary steps:

Page	Frame	Use	Dirty	Valid
0	00	1	1	1

arm Education

1	01	0	1	1
2	10	1	0	1
3	11	1	0	1
4				0
5				0
6				0
7				0

Frame	Page	Clock Hand
0	000	Pointing to this frame
1	001	
2	010	
3	011	

Page	Frame	Use	Dirty	Valid
0	00	1	1	1
1	01	0	1	1
2	10	0	0	1
3	11	0	0	1
4				0
5				0
6				0
7				0

Frame	Page	Clock Hand
0	000	
1	001	Pointing to this frame
2	010	
3	011	

Page	Frame	Use	Dirty	Valid
0	00	0	1	1
1	01	0	1	1
2	10	0	0	1
3	11	0	0	1
4				0
5				0
6				0
7				0

Frame	Page	Clock Hand
0	000	

1	100	
2	010	Pointing to this frame
3	011	

Page	Frame	Use	Dirty	Valid
0	00	0	1	1
1		0 (First 0)		0
2	10	0	0	1
3	11	0	0	1
4	01	1	1	1
5				0
6				0
7				0

Page	Frame	Use	Dirty	Valid
100	10	1	1	1
010	01	1	0	1