COMP 4611: Design and Analysis of Computer Architectures

Homework #4

Due date: Dec. 4, 2012 at 17:00

Name: __________ Student No.: __________

Q1 [20 marks] Cache

i. (15 marks) Suppose a fully associative cache has 8 frames, each frame being able to hold one block, and applies the FIFO replacement policy. Present a memory access sequence using block numbers (e.g., 2, 3, 2, 22, 7, 9) in which the miss rate would INCREASE if the size of the cache is increased to 9 frames.

0 1 2 3 4 5 6 7 8 0 1 2 9 0 10 1 11 2 12 13 14 15 16 9 10

8 frames: ~~0 1 2 3 4 5 6 7 8 0 1 2~~ 9 10 11 12 13 14 15 16 – 12 misses (or 20 misses if the initial compulsory misses that fill the cache are counted in.)

9 frames: ~~0 1 2 3 4 5 6 7 8 9 0 10 1~~ 11 2 12 13 14 15 16 9 10 – 13 misses (or 22 misses if the initial compulsory misses that fill the cache are counted in.)

ii. (5 marks) Would the same anomaly occur with a cache using the LRU replacement policy? Please explain the reason.

No.
(Note: The anomaly shown in i. is called the Belady's anomaly. The LRU policy leads to a stack structure where the 9-frame stack always contains the 8-frame stack with the same memory access sequence. Hence, there wouldn't be a case that the 8-frame cache sees a hit on a block that is not present in the 9-frame cache. Therefore it is impossible that the 8-frame cache reports fewer misses than the 9-frame one.)

Q2 [20 marks] Simultaneous multithreading (SMT)

Consider a Simultaneous Multithreading (SMT) machine with limited hardware resources. **Circle** the following hardware constraints that will limit the total number of threads that the machine can support. For the item(s) that you circle, **briefly** describe the minimum requirement to support **N** threads.

[✓ **(A)**] Number of execution functional unit

Needs at least N execution functional units for N thread to execute concurrently.

[✓ **(B)**] Number of physical registers

The system needs sufficient physical registers to meet N threads' need. In addition, it needs N PCs.

(C) Data cache size

(D) Data cache Associativity

(E) Number of instruction caches (suppose we have one instruction cache in a non-SMT processor)

Q3 (20 marks) Memory hierarchy

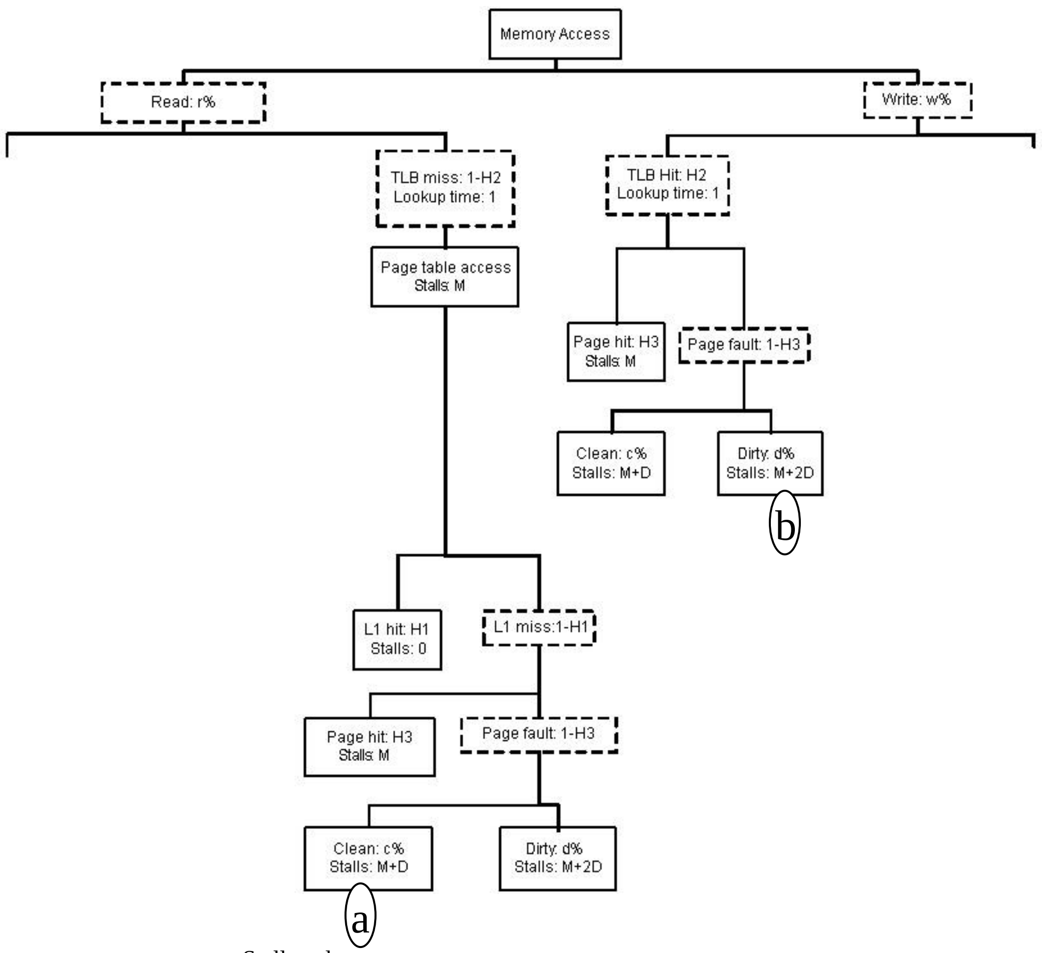
Consider a memory subsystem of hierarchical architecture. In this memory hierarchy, there are L1 cache, main memory (DRAM), and a hard disk (There is NO L2 cache). The L1 cache is a physical cache using physical addresses for indexing and tagging. The hard disk serves as the secondary storage and swapping device for the virtual memory. The whole subsystem applies following strategies.

L1 cache: L1 cache is on-chip unified cache. It uses write-through policy with write-allocate. The write-through from L1 cache goes to the main memory directly. The hit rate is H1 and the hit time is 1 cycle, meaning that there is no stall when the cache is hit. The percentage of read is r% and the write is w%.

Main memory and TLB: Main memory applies the virtual memory. It has a single level page table with on-chip TLB to speed up the page table lookup. Suppose the hit rate of TLB is H2 and the page hit rate is H3 (i.e., page fault rate is 1-H3). The hit time of TLB is one cycle. If we have to access the main memory for reading or writing a block or a page table entry, we need M stall cycles to complete reading/writing the physical memory. Assume that we have designed the physical memory to allow a form of interleaving so that reading a block and writing a word can be executed "in parallel" and complete in M stall cycles.

Disk: If we have to go to the disk to swap the page from the disk to the physical memory or from the physical memory to the disk, we need D cycles including the cycles needed for memory and disk operations. The percentage of clean pages is c% and that of dirty pages is d%. Assume that, with single level page table, all the page tables are in the main memory. The page tables are in the fixed positions and will not be paged out to the disk.

Fig. 1 gives certain parts of the Memory Access Tree.



Stall cycles $= (M + D + M + 1) \times c\% \times (1 - H3)(1 - H1) \times (1 - H2) \times r\%$

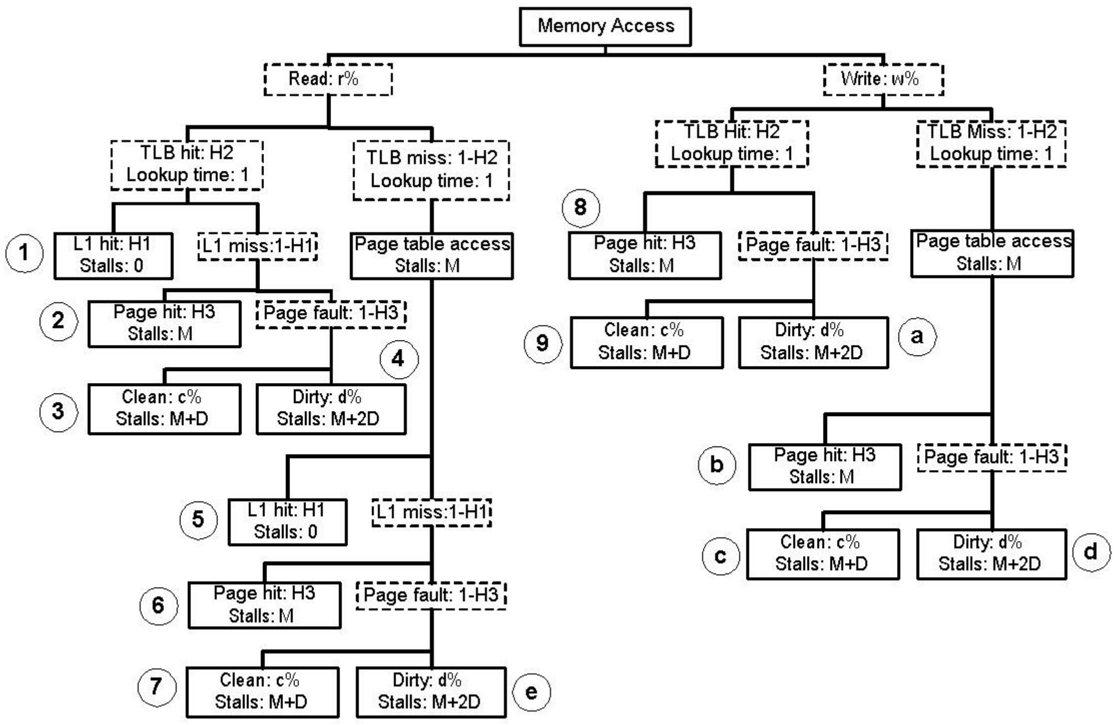
Fig. 1: The memory access tree with the subtrees for the scenarios shown: memory read with TLB miss, memory write with a TLB hit

i. (5 marks) The expression of the stall cycles per memory access for case $\textcircled{a}$ is shown in the Fig. 1. Based on the memory access tree, give the expression of the stall cycles per memory access for the case $\textcircled{b}$, the scenario of a memory write with TLB hit but page fault on a dirty page.

$$(M + 2D + 1) \times d\% \times (1 - H3) \times H2 \times w\%$$

ii. (15 marks) Complete the Memory Access Tree. Either complete Fig. 1 or draw on this page.

The complete tree:



A VSD file showing the details:

Quiz2Q5.2002.vsd

Q4 [18 marks] Amdahl’s Law and Gustafson’s Law

Suppose program P takes 200 seconds to execute on a uniprocessor system. Of this time, 30% execution time is spent on the sequential part of the program, and 70% time is spent on the parallel part. The execution of the parallel part can be accelerated by using more processors, and the speedup for the parallel part is the same as the number of processors. For example, if it takes T seconds to execute the parallel part with one processor, using k processors would reduce the amount of time to T/k seconds for that part.

i. [4 marks] On a multiprocessor system with 5 processors, how much time would it take to execute program P?

200*0.3 + 200*0.7/5 = 60+140/5 = 60+28 = 88s

ii. [6 marks] Using the Amdahl’s Law, find limit of speedup we can get by using more processors to execute program P without scaling up the problem size.

Speedup = 1/(0.3+0.7/n) = 1/0.3 = 3.3

iii. [8 marks] Suppose we can scale up the problem size (e.g., giving the program a larger input data) when adding more processors into the system. Specifically, we scale up the problem so that the parallel part of P takes 8 times as much time as before to execute on a uniprocessor system. The scaled-up problem does not change the execution time of the sequential part. Compute the speedup of program P when we run it to solve the scaled-up problem on an 8-processor system.

Speedup = 0.3+0.7*8 = 5.9

Q5 [20 marks] Hard drive and RAID-5

Consider a group of 8 hard drives forming a RAID-5 system. Each drive has one single-sided platter and 20000 tracks on the surface. Each track has 5000 sectors, and each sector contains 512 bytes. The drive's rotational speed is 9600 rpm. The configuration of the RAID-5 system is illustrated in Figure 2.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5	Disk 6	Disk 7
0	1	2	3	4	5	6	P0
7	8	9	10	11	12	P1	13
14	15	16	17	18	P2	19	20
21	22	23	24	P3	25	26	27
28	29	30	P4	31	32	33	34
35	36	P5	37	38	39	40	41
42	P6	43	44	45	46	47	48
P7	49	50	51	52	53	54	55
56	57	58	59	60	61	62	P8
63	64	65	66	67	68	P9	69

Figure 2: One example of the stripe configuration of RAID 5 using 8 disks as one group.

i. [5 marks] How much useful storage capacity, excluding redundant information, can be provided by this set of 8 disks?

20000tracks * 5000sectors/track * 512B/sector * 7 = 5.12*10^10 bytes * 7

= 3.58*10^11 bytes

ii. [6 marks] Suppose disk 2 fails at certain time, and a new replacement disk is installed. Which disk/disks will be accessed to reconstruct the missing data? Specify the disk numbers and the operations (read or write) on them.

Disk 0, 1, 3—7: read

Disk 2: write

iii. [9 points] When updating the information in one disk block, the RAID-5 system applies small writes to update the data block as well as the parity block. Suppose at any moment the disk array can serve TWO write requests at most. At certain time instance, there are several write requests on block 0, 6, 12, 23, 27, 28, 66, 50, and 58. Block 0 is currently being written. The other requests are waiting to be scheduled, and they are not ordered. The RAID-5 system can schedule any one to be serviced in parallel with the write of Block 0, if the request does not conflict with the write to Block 0. Among these requests, which **ones** can potentially be serviced in parallel with the write on block 0? Explain the reasons why these ones can potentially be serviced in parallel and why others cannot.
(Hint: We do not need to consider the correlation or dependence among the waiting requests.)

6: conflict on P0 (disk 7)
12. can be serviced
23. can be serviced
27. conflict on disk 7
28. conflict on disk 0
66. can be serviced
50. conflict on disk 0
58. conflict on disk 7