COMP 4611: Design and Analysis of Computer Architectures

Homework #3

Due date: Nov. 22, 2013 at 17:00

Name: __________   Student No.: __________

Q1 [16 marks] Cache

i. (8 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. (8 marks) Would the same anomaly occur with a cache using the LRU replacement policy? Please explain the reason.

No.
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. (Note: The anomaly shown in i. is called the Belady's anomaly.)

Q2 [8 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

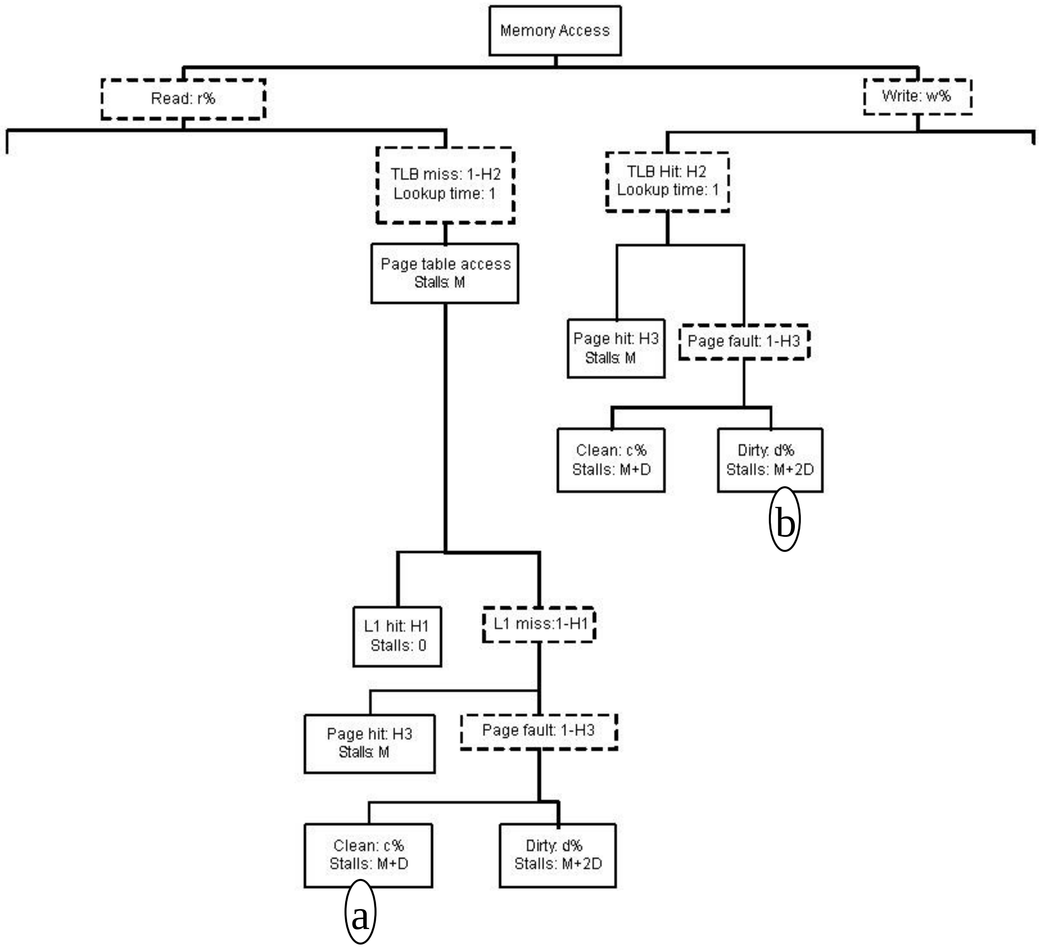
Consider a memory subsystem of hierarchical architecture. In this memory hierarchy, there are L1 cache, main memory (DRAM), and a hard disk (There is NO L2 cache). The L1 cache is a physical cache using physical addresses for indexing and tagging. The hard disk serves as the secondary storage and swapping 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 $= (M + D + M + 1) \times c\% \times (1 - H3)(1 - H1) \times (1 - H2) \times r\%$

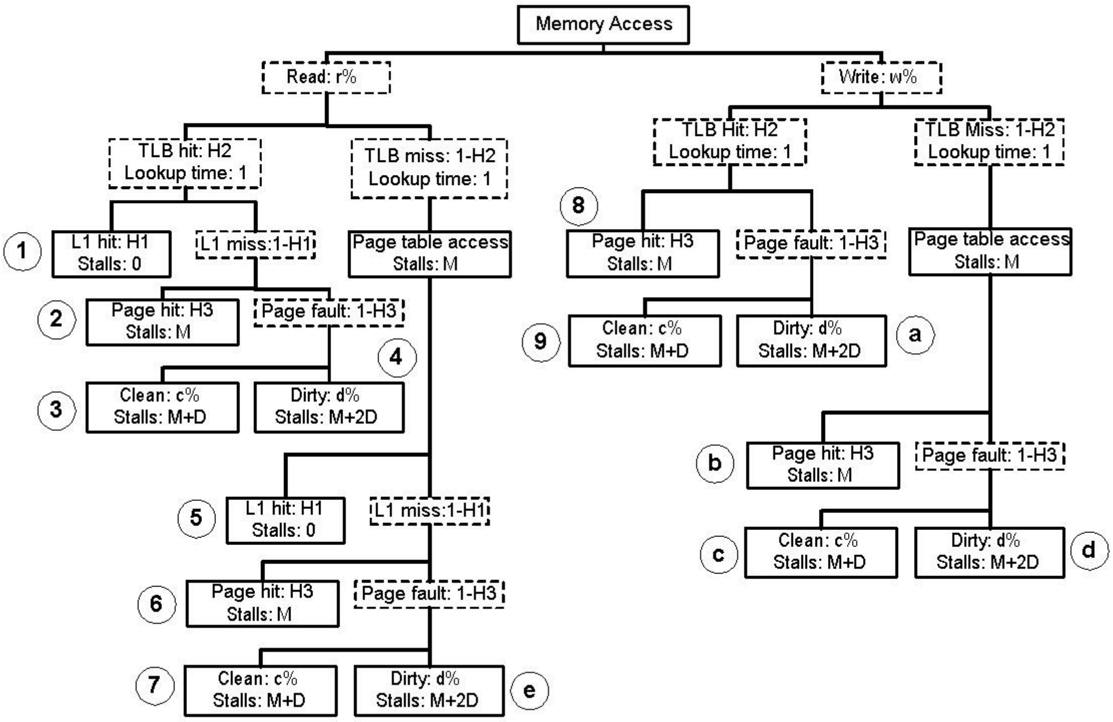
Fig. 1: The memory access tree with the subtrees for the scenarios shown: memory read with TLB miss, memory write with a TLB hit

i. (5 marks) The expression of the stall cycles per memory access for case 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 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:



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 [12 marks] Hard drive and RAID-6

Consider a group of 6 hard drives forming a RAID-6 system as shown below. There are 4 data disks, A-D, one row parity disk, R, and one diagonal parity disk, G. Blocks are labeled with the disk name and the diagonal number as shown in the following diagram. For example, the first block on Disk B is labeled B1 (shown with an underline in the diagram) because it belongs to Disk B and diagonal 1.

Data Disk A	Data Disk B	Data Disk C	Data Disk D	Row Parity Disk R	Diagonal Parity Disk G
A0	**B1**	**C2**	**D3**	**R4**	**G0**
A1	**B2**	**C3**	**D4**	**R0**	**G1**
A2	**B3**	**C4**	**D0**	**R1**	**G2**
A3	**B4**	**C0**	**D1**	**R2**	**G3**

Suppose Data Disk A and the row parity disk R are destroyed. Show how the RAID-6 system can recover data. Make sure you list the recovered blocks in the order of recovery, and show the other blocks used for recovering the block. Use the block numbers as shown in the following diagram to refer to the pertinent blocks. You can use the diagram below to help you develop the answer and write the answer in the numbered space provided below.

Data Disk A	Data Disk B	Data Disk C	Data Disk D	Row Parity Disk R	Diagonal Parity Disk G
	B1	**C2**	**D3**		**G0**
	B2	**C3**	**D4**		**G1**
	B3	**C4**	**D0**		**G2**
	B4	**C0**	**D1**		**G3**

1: recover block _A3__ using blocks _B3 C3 D3 G3_________________

2: recover block _R2__ using blocks _A3 B4 C0 D1_________________

3: recover block _A2__ using blocks _B2 C2 R2 G2_________________

4: recover block _R1__ using blocks _A2 B3 C4 D0_________________

5: recover block _A1__ using blocks _B1 D1 R1 G1_________________

6: recover block _R0__ using blocks _A1 B2 C3 D4_________________

7: recover block _A0__ using blocks _C0 D0 R0 G0_________________

8: recover block _R4__ using blocks _A0 B1 C2 D3_________________

Q6 [12 marks] Average memory access time (AMAT)

Suppose we have a processor with a 2ns clock cycle time and a unified cache. The cache has a miss rate of 0.06, a miss penalty of 18 clock cycles, and a hit time of 1 clock cycle. We assume that the read and write miss penalties are the same.

a) (3 marks) what is the average memory access time in nanoseconds?

2 ns + 0.06 × 18 × 2 ns = 4.16 ns

b) (3 marks) If we design a new cache that doubles the cache size and reduces the miss rate to 0.04. However, the clock cycle time increases by 20%. The access time to the physical main memory (DRAM) remains the same as before. What is the average memory access time in nanoseconds?

1.2*2ns + 0.04*18*2ns = 2.4ns + 1.44ns = 3.84ns

c) (6 marks) Suppose a program's instruction count is 1 billion and it has 1.5 references per instruction. The CPI with ideal memory is 2 cycles. The miss penalty is 36ns for both processors. What are the CPU execution times for the systems in questions a) and b), respectively?

CPU time(a): 10^9 *(2ns *2+1.5*36ns*0.06) = 7.24s
CPU time(b): 10^9*(2.4ns*2 + 1.5*36*0.04) = 6.96s

Q7 (8 marks) Memory consistency model

Examine the memory consistency model of a real-world processor (refer to Chapter 5.6 of the textbook).

a) (3 marks) Choose your favorite multicore CPU, and state its memory consistency model.

MIPS R10000 using sequential consistency.

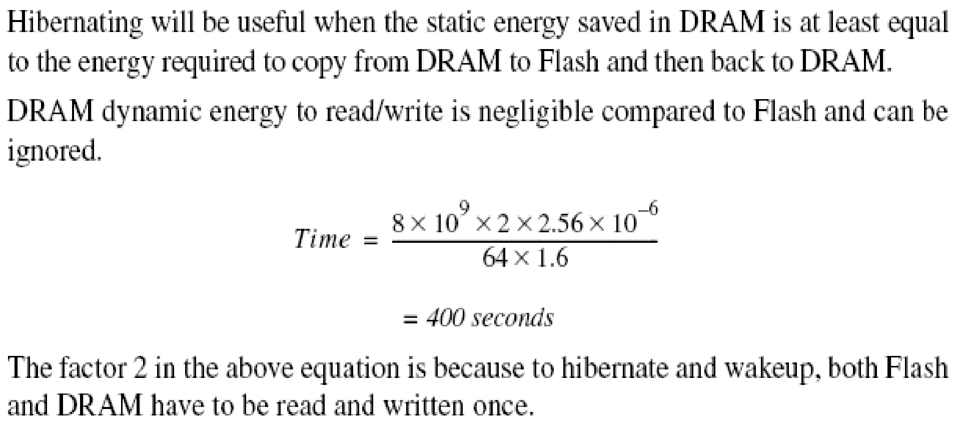
b) (5 marks) Explain the memory consistency model stated in a)

The memory operations are handled in a way so that the outcome of the computation is the same as if

1. all memory accesses on all processors were in the program order for the instructions executed on the particular processors; and
2. the memory accesses on all processors were interleaved and all processors saw the same interleaved order.

Q8 (6 marks) Hibernation

Whenever a computer is idle, we can either put it in stand by (where DRAM is still active) or we can let it hibernate. Assume that, to hibernate, we copy the content of the whole DRAM to a nonvolatile medium such as Flash. If reading or writing a cacheline (block) of 64 bytes to Flash requires 2.56uJ and reading or writing a cacheline to DRAM requires 0.5nJ. Suppose idle power consumption for DRAM is 1.6W (for 8GB). How long should a system be idle to benefit from hibernating? Assume a main memory of size 8GB.



Hibernating will be useful when the static energy saved in DRAM is at least equal to the energy required to copy from DRAM to Flash and then back to DRAM.

DRAM dynamic energy to read/write is negligible compared to Flash and can be ignored.

$$Time = \frac{8\times10^{9}\times2\times2.56\times10^{-6}}{64\times1.6}$$

$$= 400\ seconds$$

The factor 2 in the above equation is because to hibernate and wakeup, both Flash and DRAM have to be read and written once.