Advanced Computer Architecture CMSC 611
Extra Credit HW2

Due at 1.05pm, Dec 3rd, 2012

(If you wish to **go green,** then you can submit the entire Homework electronically. Make sure you include the string "CMSC 611 Homework" in your subject line. Deadline remains the same) Please **DO NOT** email your homework to Dr. Olano!! **DO NOT** include him in the CC either!! There is a strong chance it won't be graded if you do!! **Send it only to <abhay1@umbc.edu>**

1) (30 points)
   a) What is seek time in a fixed disk? What is rotational latency?
      Seek time is the time required to position the head over the correct track. Rotational latency is the time required for the correct sector to rotate under the head.
   b) How does RAID increase data *availability* in a disk IO system? How does RAID increase data *reliability* in a disk IO system?
      Data availability is increased due to redundancy of data storage, either via parity bits or mirroring. This allows hot swapping which means that a single failed disk can be replaced without taking down the entire array.
      Data reliability is increased due to the detection of errors via the Parity bits.
      Performance is increased via parallel access to multiple disks.
   c) What is a Vector operation? Give an example
      A vector operation is the same operation applied to multiple sets of data. The Pentium 4 allows a floating point operation to be applied to 4 single precision FP numbers (each 32 bits) that are contained in a 128 bit registers.
   d) What is the primary advantage of adding a translation lookaside buffer (TLB)?
      A specially tailored hardware designed for accelerating address translation from virtual address space to physical address space
   e) What is the difference between a conflict miss and a compulsory miss for caches? How would you reduce each type?
      Compulsory misses are those misses caused by the first reference to a data. Increasing the block size reduces the number of those misses.
      Conflict misses occur when multiple memory locations map to the same cache location. Increasing the cache size and associativity reduce those misses.
   f) Define
      i. Out-of-order execution
      ii. Very long instruction word (VLIW)
      Out-of-order execution – a superscalar architecture in which the order of instruction execution depends on data and resource availability. This order may be different from that in the program.

VLIW – a static ILP architecture in which the compiler, based on independence, selects sets of instructions for parallel execution. These sets are packed in words of a wide (e.g., 512 bits) instruction memory.

2) (30 points)
Assume you have a processor with an ideal CPI without memory stalls for each instruction type as follows: ALU=1, Load/Store=1.5, Branch=1.5, Jump=1. Consider an application which has an instruction mix of 30% ALU and logical operations, 40% load and store, 20% branch and 10% jump instructions.
(a) Assume a 4-way set associative 1-level separate data and instruction cache with a miss rate of 20% (0.20) for data accesses and miss rate of 10% (0.10) for instructions, and a miss penalty of 40 cycles for both instruction and data caches (and assume a cache hit takes 1 cycle). What is the effective CPU time (or effective CPI with memory stalls) and the average memory access time for this application with this Cache organization?
First compute ideal CPI without memory stalls:
CPIideal=(0.3*1)+(0.4*1.5)+(0.2*1.5)+(0.1*1) = 1.3

The effective CPI, and therefore execution time, includes the memory stalls. This is given by
CPIactual = CPIideal + Memory Stalls per instruction.

The Memory stalls per instruction, Stalls/Inst is given by
Stalls/Inst= (stalls-data/data)+(stalls-inst/inst)
Stalls-data (stalls due to data)= (data miss rate * data miss-penalty) * data accesses/inst
Stalls-inst (stalls due to inst) = (inst. Miss rate * miss penalty) * inst.accesses/inst

The data accesses take place during the 40% Load/Store operations.
The instruction accesses take place once for each instruction – i.e., 100% of the program.
Therefore, stalls/inst = ((0.2*40)*0.4) + (0.1*40)*1) = 7.2

Now compute the Average Memory Access time (AMAT). This is given by:
AMAT = percentage of data accesses (hit time + data miss rate* miss penalty) +
percentage of inst. Accesses (hit time + inst.miss rate*miss penalty).

The percentage data accesses is 0.4/1.4 (i.e., total of 1.4 accesses to memory during entire program execution, of which 0.4 are for data), and for inst it is 1/1.4
Therefore AMAT = (0.4/1.4)(1 + 0.2*40) + (1/1.4)(1+ 0.1*40) = 6.14

Note that another way to derive stalls per instruction is to multiply the average number of memory accesses per instruction by the AMAT minus hit time. In this example, the average number of memory accesses per instruction is 1.4 (i.e., 1 for instruction and 0.4 for data due to Load/Store). Therefore the average stall cycles per instruction can be derived as:

1.4*(AMAT-1) = 1.4*[(0.4/1.4)(1 + 0.2*40) + (1/1.4)(1+ 0.1*40) - 1] = 7.2 which is identical to what we got earlier.

(b) Now consider a 2 level 4-way unified cache with a level l (L1) miss rate of 25% (0.25) and a level 2 (L2) local miss rate of 30% (0.30). Assume hit time in L1 is 1 cycle, assume miss penalty is 15 cycles if you miss in L1 and hit in L2 (i.e., hit time in L2 is 15 cycles), and assume miss penalty is 50 cycles if you miss in L2 (i.e., miss penalty in L2 is 50 cycles). Derive the equation for the effective CPU time (or effective CPI) and the average memory access time for the same instruction mix as part (a) for this cache organization.
First note that the miss rate for L2 is given as the local miss rate. Average memory accesses per instruction = 1.4 as noted earlier (0.4 for data and 1 for inst).

AMAT= (hit time L1) + (miss rate L1)*(Hit time L2 + (Local miss rate L2 * Miss Penalty))

The global miss rate for L2 is not the same as L1 – but you can derive the global miss rate from the local miss rate of L1 and L2. Note that the local and global miss rate of L1 are the same.

AMAT = (1 + (0.25)*(15 + (0.3*50))) = 8.5

(THERE WAS A TYPO HERE. IF YOU HAD TAKEN 10 as miss rate, the answer is 7.25)

Effective CPI = Ideal CPI + average memory stalls per instruction.
Average memory stalls per instruction = misses per instruction-L1 * Hit-time L2 + Misses-per-instruction L2*miss-penalty

This is equivalent to: Average mem.stalls per instruction = (mem.accesses/inst)*(miss rate L1 * Hit time L2 + Miss rate L2 * miss penalty).

However, note that the Miss rate L2 in the above equation refers to the global miss rate of L2.

Alternately, we can use our earlier observation that average mem.stalls per instruction can be derived as

Mem.accesses per instruction * (AMAT -1).

This gives us

Avg.Mem.stalls/inst = 1.4*(AMAT -1) = 1.4*(8.5 -1) = 10.5 cycles.

Which of the two designs (between part a and part b) gives a better performance? Explain your answer.

The **first design**, gave us lower AMAT and avg. mem. Stalls/inst and therefore it is a better design.

3) (40 points)

Consider a system with the following processor components and policies:

- A direct-mapped L1 data cache of size 4KB and block size of 16 bytes, indexed and tagged using physical addresses, and using a write-allocate, write-back policy
- A fully-associative data TLB with 4 entries and an LRU replacement policy
- Physical addresses of 32 bits, and virtual addresses of 40 bits
- Byte addressable memory
- Page size of 1MB

Part A

Which bits of the virtual address are used to obtain a virtual to physical translation from the TLB? Explain exactly how these bits are used to make the translation, assuming there is a TLB hit.

The virtual address is 40 bits long. Because the virtual page size is 1MB = 2^20 bytes, and memory is byte addressable, the virtual page offset is 20 bits. Thus, the first 40-20=20 bits are used for address translation at the TLB. Since the TLB is fully associative, all of these bits are used for the tag; i.e., there are no index bits.

When a virtual address is presented for translation, the hardware first checks to see if the 20 bit tag is present in the TLB by comparing it to all other entries simultaneously. If a valid match is found (i.e., a TLB hit) and no protection violation occurs, the page frame number is read directly from the TLB.

Part B

Which bits of the virtual or physical address are used as the tag, index, and block offset bits for accessing the L1 data cache? Explicitly specify which of these bits can be used directly from the virtual address without any translation.

Since the cache is physically indexed and physically tagged, all of the bits from accessing the cache must come from the physical address. However, since the lowest 20 bits of the virtual address form the page offset and are therefore not translated, these 20 bits can be used directly from the virtual address. The remaining 12 bits (of the total of 32 bits in the physical address) must be used after translation.

Since the block size is 16 bytes = 2^4 bytes, and memory is byte addressable, the lowest 4 bits are used as block offset.

Since the cache is direct mapped, the number of sets is 4KB/16 bytes = 2^8. Therefore, 8 bits are needed for the index.

The remaining 32-8-4 = 20 bits are needed for the tag.

Part C
The following lists part of the page table entries corresponding to a few virtual addresses (using hexadecimal notation). Protection bits of 01 imply read-only access and 11 implies read/write access. Dirty bit of 0 implies the page is not dirty. Assume the valid bits of all the following entries are set to 1.

Virtual page number		Physical page number	Protection bits	Dirty bits
1	FFFFF	CFC	11	0
2	FFFFE	CAC	11	0
3	FFFFD	CFC	11	0
4	FFFFC	CBA	11	0
5	FFFFB	CAA	11	0
6	FFFFA	CCA	01	0

The following table lists a stream of eight data loads and stores to virtual addresses by the processor (all addresses are in hexadecimal). Complete the rest of the entries in the table corresponding to these loads and stores using the above information and your solutions to parts A and B. For the data TLB hit, data cache hit, and protection violation columns, specify "yes" or "no." Assume initially the data TLB and data cache are both empty.

	Processor load/store to virtual address	Corresponding physical address	Part of the physical address used to index the data cache	Data TLB hit?	Data cache hit?	Protection violation?	Dirty bit
1	Store FFFFF ABAC1	CFCABAC1	AC	No	No	No	1
2	Store FFFFC ECAB1	CBAECAB1	AB	No	No	No	1
3	Load FFFFF BAAE3	CFCBAAE3	AE	Yes	No	No	0
4	Load FFFFB CEBC3	CAACEBC3	BC	No	No	No	0

5	Store FFFFE AAFA1	CACAAFA1	FA	No	No	No	1
6	Store FFFFC AABC9	CBAAABC9	BC	Yes	No	No	1
7	Load FFFFD BAAE2	CFCBAAE2	AE	No	Yes	No	0
8	Store FFFFA ABAC4	CCAABAC4	AC	No	No	Yes	0