ECE/CS 552: Introduction to Computer Architecture
ASSIGNMENT #4
Due Date: At the beginning of lecture, November 17th, 2010

This homework is to be done individually. Total 6 Questions, 100 points

1. (15 pts.)The following set of instructions is to be renamed onto a set of physical registers. The initial register map and free list are given. **We assume that a physical register is released and placed on the tail of the free list when the next definition of the same architected register is committed.**

1) (10 pts.) Show the rename mappings after each instruction has been dispatched by filling out the table below, rewriting each instruction with renamed registers and updating the rename mappings in the table appropriately. The first instruction has been done for you. ~~Assume that physical registers are allocated from the free list in increasing numerical order.~~ (This line is confusing)

Free list: (head) ~~P9~~, P10, P11, P12, P13, P14, P15 (tail)

Original Instruction	Cycle timestamp <disp,commit>	Renamed Instruction	Rename mappings after instruction dispatches							
			R1	R2	R3	R4	R5	R6	R7	R8
Initial register mappings			P1	P2	P3	P4	P5	P6	P7	P8
R5 <= R1 + R5	0, 5	P9 <= P1+P5					P9			
R1 <= mem (R3 + R1)	1,6	P10 <= mem(P3+P1)	P10							
R1 <= R3 + 4	2,7	P11 <= P3 + 4	P11							
R7 <= R1 + R2	3,8	P12 <= P11+P2							P12	
R4 <= R4 – 1	4,9	P13 <= P4 - 1				P13				
R4 <= R4 + R8	5,10	P14 <= P13+P8[1]				P14				
R7 <= R4 – R7	6,11	P15 <=P14-P12							P15	
R4 <= mem (R1 + R6)	7,12	P5 <= mem(P11+P6)				P5				
R1 <= R5 – R4	8,13	P1 <= P9-P5	P1							

1: If assuming the free list is always in increasing numerical order, the renaming physical register is different starting from here. But the architectural register being renamed should be the same.

2) (5 pts.) What is the free list of physical register after the registers of the last instruction are renamed?

P10, P12. (P13 can be counted or not).

2. (13 pts.) Consider an instruction sequence:

A B C D E F G H.

Figure 1 shows the dependency information between them. The arrow represents the dependency information between instructions. There are two computers P1 and P2. The pipeline width of both of them is 2. That is, at most 2 instructions can be issued at the same time. P1 is an in-order pipeline (no instruction can start execution before a preceding instruction starts execution) and P2 is an out-of-order pipeline computer (instructions can be executed in any order). Each instruction takes one cycle.

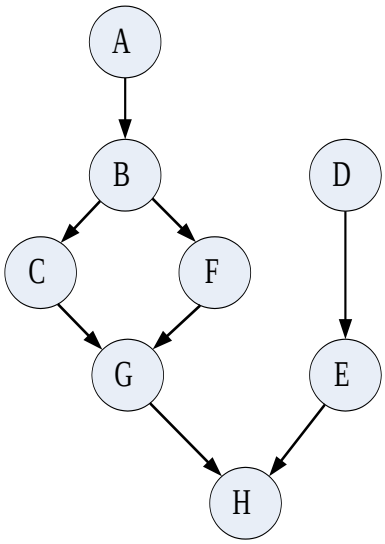

Figure 1. Dependency information between instructions

1) (10 pts.) Show the schedule of issuing the instructions to execution in both computers in table 1. The time in table 1 is represented by number of cycles that has passed. Lane 1 and Lane 2 represent the two lanes in the datapath since its width is 2.

Table 1: Schedule of instruction issuing

Time		1	2	3	4	5	6	7	8	9	10
P1	Lane1	A	B	C	E	G	H				
	Lane2			D	F						
P2	Lane1	A	B	C	G	H					
	Lane2	D	E	F							

2) (3 pts.) What is the speedup of P2 compared with P1 on this instruction sequence?

SPEEDUP = 6/5 = 1.2

3. (12 pts.)
Given a 2 Kbytes two-way set associative cache with 16 byte lines and the following code:

```
for (int i = 0; i < 1000; i++) {
    A[i] = 40 * B[i];
}
```

1) (10 pts.) Compute the overall miss rate (assume array entries require 4 bytes)

Each array contains 1000 elements. Each cache line contains 4 words. Since each array element will be accessed only once, all misses shall be compulsory misses.

Since every 4 words will be loaded or written back simultaneously in a cache, the miss rate is 25% since every 4^{th} access misses.

2) (2 pts.) What kind of cache locality is being exploited?

Spatial locality.

4. (10 pts.) Consider a cache with the following characteristics:

32-byte blocks
8-way set associative
256 sets
32-bit addresses
writeback policy
LRU replacement policy

1) (2 pts.) How many bytes of data storage are there?

$256 \times 8 \times 32 = 2^{18} = 64$ KB

2) (2 pts.) How many tag bits per set?

$32 - \log_2 256 - \log_2 32 = 32 - 8 - 5 = 19$ bits per set

3) (3 pts.) What operation is needed upon a read-miss (the program wants to read from a memory location that is not in the cache)?

a. Find the LRU cache line to replace. If it is dirty, write the block to next level cache / memory.
b. Fetch 32-byte memory data from next level cache/memory of that memory address and other data from the same block/line;
c. Update the tag bit of that cache line.

4) (3 pts.) What operation is needed upon a write-miss (the program wants to write to a memory location that is not in the cache)?

a. Find the LRU cache line to replace. If it is dirty, write the block to next level cache / memory.
b. Fetch 32-byte memory data from next level cache/memory of that memory address and other data from the same block/line;
c. Write to the memory location in that cache line. Mark the cache line as dirty. Update the tag bit of that cache line.

5. (20 pts.) Consider a 3-way set associative cache. A, B, C, D are memory addresses

that have the same index bits but different tag bits from each other. In a program, the reference sequence is as follows:
A, B, A, C, D, A, D, C, A, C
1) (5 pts.) What is the miss rate if the cache is using LRU replacement policy?

40%

2) (5 pts.) What is the miss rate if the cache is using MRU replacement policy?

50%

3) (10 pts.) Assuming the memory addresses being accessed are still A, B, C and D, provide a case of memory reference sequence with length=10, in which MRU performs better than LRU. Show the reference sequence and the miss rate of LRU and MRU policy.

For example: A,B,C,D,A,B,C,D,A,B

6. (30 points) *Simplescalar Simulation*
This problem will introduce you to the Simplescalar simulator (documentation is available from http://www.simplescalar.com/. Note that you will need a Unix account at CAE for this problem (all students in this class are eligible for accounts at CAE). The SimpleScalar 3.0 simulator is available at CAE in "**/home/vhosts/ece552.ece.wisc.edu/simplesim-3.0**".
You will use a script called "RUNgcc" to run the gcc benchmark from the SPECint95 benchmark suite. Copy the files in "**/home/vhosts/ece552.ece.wisc.edu/hw4**" to somewhere that you have write access.

1) (15 pts.) "sim-cache" is a cache simulator in SimpleScalar. Report the L2 unified cache, L1 instruction cache, and L1 data cache miss rates, using the following command (run from your hw4 directory):

./RUNgcc /home/vhosts/ece552.ece.wisc.edu/simplesim-3.0/sim-cache ./gcc.ss -cache:il1 il1:128:64:1:l -cache:il2 dl2 -cache:dl1 dl1:256:32:1:l -cache:dl2 ul2:1024:64:2:l >& outfile

This will run the gcc benchmark and simulate a two-level cache hierarchy with an 8K direct-mapped L1 instruction cache with 64 byte lines (128 sets, 64 byte lines, 1-way set associative, 'l' for LRU replacement policy), an 8K direct mapped L1 data cache with 32 byte lines (256 sets, 32 byte lines, 1-way set associative, 'l' for LRU replacement policy), and a 128K 2-way set associative unified L2 cache with 64 byte lines. The output of the simulation will be placed in "outfile".

Miss rate of the three caches should be close to the values given below:

IL1: 0.0118; DL1: 0.0521 UL2: 0.4852

2) (15 pts.) Using the RUNgcc script and the sim-cache tool, find the sizes of each of the three caches that will reduce the miss rates of each of the three caches by 50%. Report the configuration as well as the miss rate reported by sim-cache.

Any configuration that decreases the miss rate of each cache by no less than 50% are acceptable.