School of Computer Science
Georgia Institute of Technology

CS4290/CS6290, Fall 2010
Hyesoon Kim, Instructor

Exam 1, Oct. 5, 2010

Name :_____

GT Account:_____

Problem 1 (10 points):_____

Problem 2 (15 points):_____

Problem 3 (10 points):_____

Problem 4 (15 points):_____

Problem 5 (10 points):_____

Total (60 points):_____

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

Name:_____

Problem 1 (10 points):

Part a. (5 points) Out of order scheduling processors support in-order retirements using ROB, history buffer, check point mechanisms etc. Without in-order retirement, an out of order scheduling processor can still execute programs correctly. Why is supporting in-order retirement important?

To support debugging, precise interrupt/exceptions

Part b. (5 points) Compiler performs data and control dependence analysis. Not all data dependence analysis can be done by the compiler. Why? Provide a code example also. Memory data dependence.

```
STORE R1, 0[R2]
LOAD R5, 24[R8]
STORE R3, -8[R9]
```

RAW exists if (R2+2) == (R8+24) WAR exists if (R8+24) == (R9-8) WAW exists if (R2+0) == (R9-8)

Name:_____

Problem 2 (15 points):

Part a. (5 points) What kind of information is stored in the BTB? (provide at least two information) Branch

target address, branch types (indirect, return, conditional), branch taken/not-taken (for some architectures)
Part b. (5 points) What is the size of a g-share branch predictor, which has a 12-bit history register? Do you think the size is appropriate? Explain your answer.

2$\hat{1}$2*2/8 = 1KB one 1KB branch predictor is not so big in today's modern processor's standards. (You can claim that 1KB is big. but you need to explain why you think it is big and what are downsides.) Large branch predictors take longer to warm up and also consume too much power, increase branch predictor access time. Smaller branch predictor has lower prediction accuracy.
Part c. (5 points) There are several mechanisms to reduce branch misprediction penalty other than a branch predictor. Discuss at least two mechanisms. Explain pros and cons of each mechanism. Predication: (+)

avoid branch misprediction penalty (-) both paths are always fetched/executed

Delay branch slot: (+) reduce pipeline bubbles (-) deeper pipeline cannot find many instructions to fill out delay branch slot. Providing ISA compatability is very hard. combining with branch predictor becomes too much complicated.

Dual/multi-path execution: (+) avoid branch misprediction penalty. (-) many useless instructions are fetched/executed

Name:_____

Problem 3 (10 points):

Part a. (5 points) Provide an example of loop interchange. What kind of cache misses can be reduced using loop interchange?

Capacity misses can be reduced.

```
for (j=0;j<10,000;j++) {
  for (i=0; i<40,000;i++) {
    c[i][j] = a[i][j]+b[i][j];
  }
}

 original code

for (i=0; i<40,000;i++) {
   for (j=0;j<10,000;j++) {
     c[i][j] = a[i][j]+b[i][j];
  }
}

  after loop interchange
```

Part b.(5 points) What is MSHR? What kind of information is stored in the MSHR?

Miss Status/Information Holding Register. In a non-blocking cache, MSHR stores unresolved miss information for each miss. Memory address, data, valid, # of outstanding memory requests,

Name:_____

Problem 4 (15 points):

```
        0x00 ADD R1, R2, R3
        0x04 MUL R4, R1, R3
        0x08 BR  R4, TARGET
        0x10 MUL R2, R2, -1
        0x14 ADD R1, R7, R3
TARGET  0x18 ADD R5, R2, R3
        0x20 ADD R6, R2, -1
```

The execution time of all instruction is one cycle except for multiplication (which takes 2 cycles) and division (which takes 4 cycles.) We assume that the register file can be written and read at the same cycle. How many cycles to complete the following sequence of instructions? The branch at 0x08 is taken in this problem.

Calculate the total execution time of the above code for the two architectures in Figure 1. Calculate IPC also. The execution time of each hardware unit is as follow.

unit name	the unit location	delay
MUX	all stages	0.1 ns
I-cache access cycle	FE_stage	0.8 ns
Branch execution unit	EX_stage	0.5 ns
+size unit	FE_stage	0.2 ns
Register read&write	ID_stage	0.8 ns
SIGN EXT	ID_stage	0.1 ns
ALU	EX_stage	0.7 ns
Mem access cycle	MEM_stage	0.8 ns
Latch write time	all stages	0.05ns
Latch read time	all stages	0.05ns

A critical length is determined by the longest path. To simplify the problem, you assume that all the hardware units are shown in these diagrams. All the pipeline latches require latch read/write time including the PC register. In these diagrams, FE_stage, ID_stage, EX_stage, and MEM_stage have latches.

Name:_____

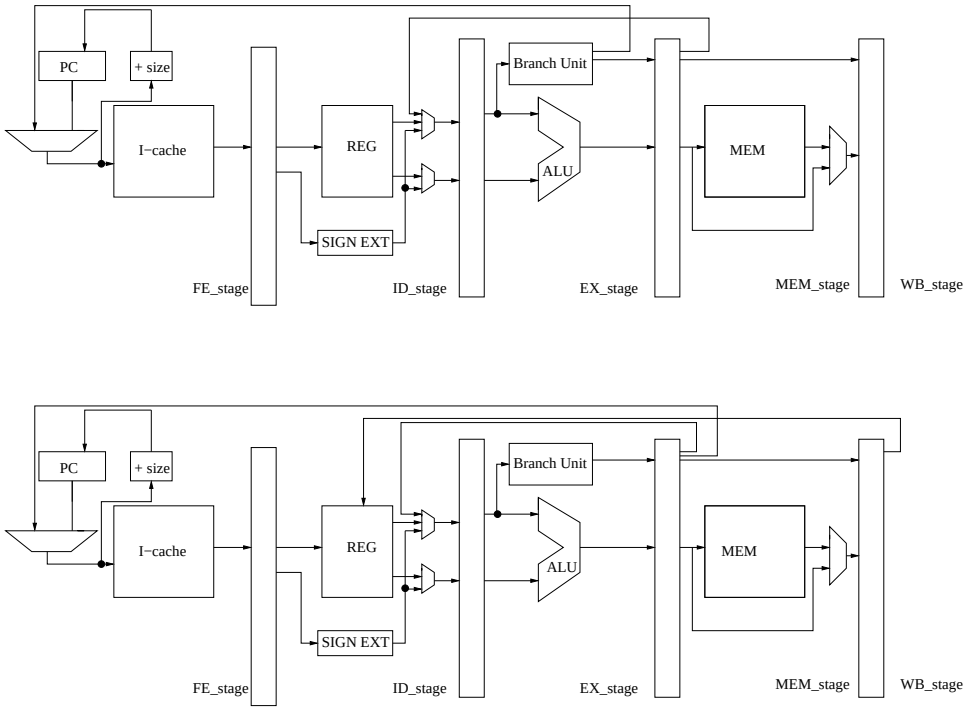

Figure 1: Top: case (a), Bottom case (b)

	Case (a)	Case (b)
# of cycles to complete the code	13	14
Cycle time	1.5 ns	1 ns
Total execution time	19.5	14
IPC	0.38	0.35

case (a) : the critical path is: latch read (0.05)+ branch unit (0.5) + mux (0.1) + I-cache (0.8) + latch write (0.05) = 1.5 ns

case (b) : the critical path is: latch read (0.05)+ max (0.1) + I-cache (0.8) or Register access or MEM + latch write (0.05) = 1.0 ns (FE)

Name:_____

Problem 5 (10 points):

Part a. (7 points) A computer has a 32KB write through cache and 1MB physical memory. Each cache block is 8B, the cache is 4-way set associative and uses the true LRU replacement policy. Assume 26-bit virtual address space and byte-addressable memory. It uses a 8KB page size. The cache uses a virtual tag and physical index. How big (in bits) is the tag store?

# of sets: 32KB/(8B*4) = 1K, # of index bits 10 bits # of physical address bits in the virtual address: 8KB 13 bits. LSB 13 bits in the virtual address bits is the same as the physical address. So we can use the LSB 13 bits to index physical cache. # of tag bits: 26 - 10 - 3 = 13.

(13 + valid (1) + 2 (LRU)) * 1K * 4 = 64Kbits = 8KB

Part b.(3 points) What are benefits of a virtual tag and physical index cache compared to a physical tag and physical index cache?

If virtual index and physical index are the same: there is no need to translate memory addresses to index the cache. so it can save the translation time. cons: virtual tag, whenever a process is changed, the entire cache has to be flushed. Permission has to be check separately.

If virtual index and physical index are not the same, the cache system will not provide any benfits over physical index physica tag. before accessing the cache, the address has to be translated just like physical index physical tag. Having virtual tag does not help at all in terms of saving translation time.