MS108: Computer System 1

Spring 2014

Homework #2

Due: Two Weeks from Assignment

TA: Ran Ye

Email: 叶冉 [happyinglife@sjtu.edu.cn]

Collaboration Policy

These homework sets will be extremely valuable as tools for learning the material and for doing well on the midterm and final. You are required to obey the following rules:
(a) Each student should write out their solution independently and in their own words.
(b) Same applies to programming assignments – you should do your own coding.
Above all, make sure that you understand the solution to these homework problems. They really are assigned to help you understand the material and be prepared for the types of problems on the midterm and final!

Q1. Scoreboarding

In a scoreboarding system, instructions go through 4 steps in the pipeline—issue (ID1), read operands (ID2), execution (EX) and write result (WB).

i. In which step does the scoreboard check for WAR hazards?

ii. In which step does the scoreboard check for structural hazards?

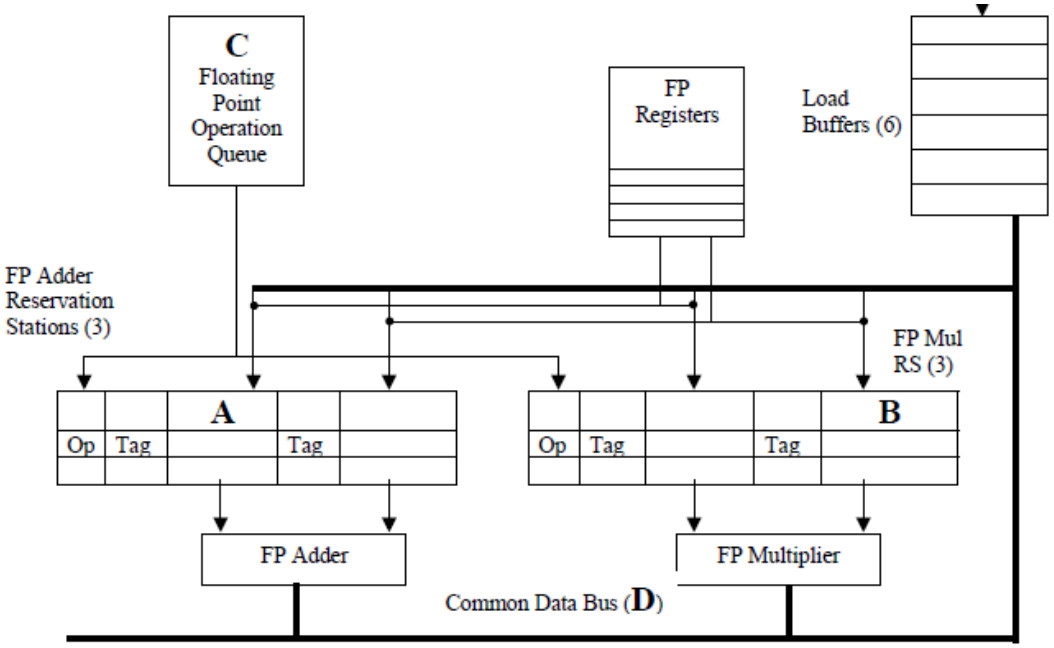
iii. The scoreboard checks for the WAW hazard in the issue step. Can it postpone the checking for WAW hazard to the write result step? Why?

Q2. Tomasulo's algorithm

The drawing below depicts the basic structure of Tomasulo's algorithm as described in the textbook and during class. This is the version without a reorder buffer – all renaming occurs in the reservation stations.

i. Where indicated by the letters A, B, C, and D determine the width of the fields in the reservation stations, the width of the buses, etc. Please write your answers in the box below.

A=		C=	
B=		D=	



ii. Now consider the three reservation stations associated with the FP adder. How many (and what size) comparators are needed in order to determine when relevant values are being broadcast on the Common Data Bus? Why?

Q3. Scoreboarding vs. Tomasulo's Algorithm

A shortcoming of the scoreboard approach occurs when multiple functional units that share input buses are waiting for a single result. The units cannot start simultaneously, but must serialize. This is not true in Tomasulo's algorithm. Give a code sequence that uses no more than 10 instructions and shows this problem. Assume the same hardware configuration as in the lectures. Indicate where Tomasulo's algorithm can continue, but the scoreboard approach must stall. Assume the following latencies.

Instruction Producing Result	Instruction Using Result	Latency in Clock Cycles
FP ALU op	Another FP Alu Op	3
FP ALU op	Store double	2
Load Double	FP ALU Op	1
Load Double	Store Double	1

Q4. Instruction set design

A given processor has 56 instructions (corresponding to 56 operation codes) and 128 registers, and uses 16-bit immediates in its ISA. In the instruction set, there are four types of

instructions as listed below:
Type A: takes one source register, uses one destination register,
Type B: takes two source registers, uses one destination register,
Type C: takes one source register, one immediate, uses one destination register,
Type D: takes one immediate, uses one destination register.
Assume that the ISA requires that all instructions be a multiple of 8 bits (1 byte) in length, and the operation codes (opcodes) are of fixed length (i.e., the ISA does not use shorter opcodes for some instructions and longer opcodes for others).

i. How many bits do we need to use to encode the opcodes?

ii. How many **bytes** do we need to use to encode the Type-B instruction?

iii. How many **bytes** do we need to use to encode the Type-D instruction?

Q5. Branch Prediction

Suppose we have a program with the following sequence of statements. It has three branches as indicated by B1, B2 and B3.

…
if (a<b) then a=2*a; # branch B1
if (c>b) then c=c-b; # branch B2
if (a>c) then a=a-b; # branch B3
…

The instruction sequence corresponding to the above statements is shown in Fig. 1 in assembly language. In Fig. 1, register R1 is used for the variable a, R2 for b and R3 for c. R4 is a register to store temporary results. We maintain a (m,n) predictor for each branch and the predictor for the branch B3 is illustrated in the following table (Fig. 2).

```
        …
S1:     SUB   R4, R1, R2  ;   R4=R1-R2
B1:     BGE   R4, S2      ;   if R4≥0, then branch to S2 (B1 branch)
        ADD   R1, R1, R1  ;   R1=R1+R1
S2:     SUB   R4, R3, R2  ;   R4=R3-R2
B2:     BLE   R4, S3      ;   if R4≤0, then branch to S3 (B2 branch)
        SUB   R3, R3, R2  ;   R3=R3-R2
S3:     SUB   R4, R1, R3  ;   R4=R1-R3
B3:     BLE   R4, S4      ;   if R4≤0, then branch to S4 (B3 branch)
        SUB   R1, R1, R2  ;   R1=R1-R2
S4:     …
```

Fig. 1: The sequence of instructions

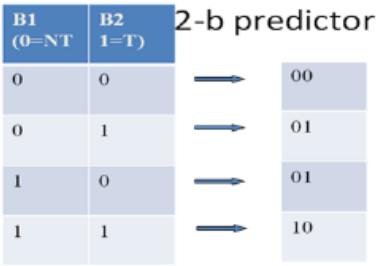


Fig. 2: The (m,n) predictor for branch B3

a) For the (m, n) predictor, what is the parameter m and what is the parameter n?

b) Suppose at certain time instance, the variables a=26, b=50 and c=46. The program counter (PC) points the first instruction at label S1 (SUB R4, R1, R2). Suppose the state of predictor of B3 at the time instance is shown in Fig. 2. According to this predictor, what prediction will be made for the branch B3 (TAKEN or NOT TAKEN)? Explain the reason.

c) Follow the conditions of the question b. When the program has just finished the execution of the branch B3 (i.e., PC becomes more than B3), what will be the state of the predictor of B3?

d) Suppose we can use up to 10000 bits for dynamic branch prediction using this (m,n) predictor scheme. How many entries can we hold in the cache at most? Assume the number of entries is a power of 2, and each entry corresponds to a different instruction address. (Hint: m and n are determined in question a)

Q6. Pipeline performance with stalls

Assume the following MIPS instruction mix:

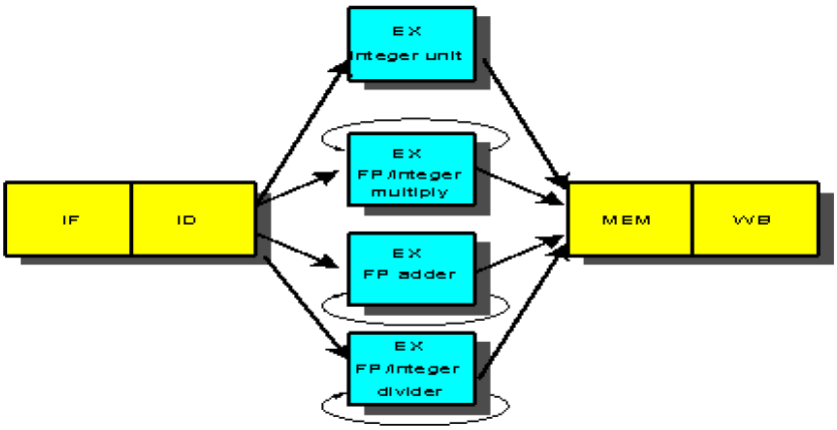
Type	Frequency	
Arith/Logic	50%	
Load	30%	of which 25% are followed immediately by an ALU instruction using the loaded value
Store	5%	
Branch	15%	of which 30% are taken

What is the resulting CPI for the pipelined MIPS with forwarding and branch address calculation in ID stage when using a predict branch not-taken scheme?

Q7. Pipelining without/with Forwarding

A multiple-execution pipelined processor with multiple execution functional units is shown below. In the execution stage, the Integer Unit takes 1 cycle, the FP/Integer Multiply Unit is pipelined and takes 4 cycles, the FP Adder Unit is pipelined and takes 3 cycles and the FP/Integer Divider is NOT pipelined and takes 5 cycles. Assume the data memory (data cache) can service one read or write per clock cycle. The instruction memory (instruction cache) works independently of the data memory. Assume the register file supports first half cycle write and second half cycle read. Note this is a single-issue and in-order issue pipeline.

In this question, you can use short forms like F, D, E, M1, M2, M3, M4, D1, D5, M, W to represent IF, ID, EX, Multiply stage 1, Multiply stage 2, Multiply stage 3, Multiply stage 4, Divide cycle 1, Divide cycle 5, MEM, WB, and so on.



i. Fill in the pipeline timing chart for the code segment WITHOUT forwarding.

Instruction	Clock Cycles																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
LD F2, 1000(R0) (example)	F	D	E	M	W																				
LD F3, 1004(R0)																									
DIVF F0, F2, F3																									
ADDF F1, F0,45																									
SUBF F2, F2, F3																									
LD R4, 1008(R0)																									
MULTF F3, F1, F2																									

ii. Fill in the pipeline timing chart for the code segment WITH forwarding. The forwarding logic can forward data from any execution functional unit's final output and the MEM/WB latch to any execution functional unit's initial input.

Instruction	Clock Cycles																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
LD F2, 1000(R0) (example)	F	D	E	M	W																				
LD F3, 1004(R0)																									
DIVF F0, F2, F3																									
ADDF F1, F0,45																									
SUBF F2, F2, F3																									
LD R4, 1008(R0)																									
MULTF F3, F1, F2																									