Computer Architecture Homework Assignment #2

TA in charge: Jinkwon Kim
E-mail: coco@kaist.ac.kr
Office: N1 - #922

I. Submission and grading

- ✓ Due date: April, 12th (Fri.) 23:59:59.
- ✓ Submit your homework as a hardcopy into HW box prepared near the office room #922 in N1 building. Do not submit by KLMS or TA's e-mail.
- ✓ Late submissions will not be accepted. Please keep the submission due date.
- ✓ Each score has been written at the end of each question. The total score is 100.
- ✓ You will be given 0 point for any kind of cheating.
- ✓ Please explicitly denote number of each question and its answer. Otherwise, it can be considered as not be done.
- ✓ Please give detailed process how you solve questions. Otherwise, no point will be charged.

II. Questions

✓ Please solve every questions as below **(Total 100 points)**:

1. We look at how software techniques can extract instruction-level parallelism (ILP) in a common vector loop. The following loop is the so-called DAXPY loop (double- precision aX plus Y) and is the central operation in Gaussian elimination. The following code implements the DAXPY operation, $Y = aX + Y$, for a vector length 100. Initially, R1 is set to the base address of array X and R2 is set to the base address of Y:

```
        DADDIU   R4,R1,#800    ; R1 = upper bound for X
foo:    L.D      F2,0(R1)      ; (F2) = X(i)
        MUL.D    F4,F2,F0      ; (F4) = a*X(i)
        L.D      F6,0(R2)      ; (F6) = Y(i)
        ADD.D    F6,F4,F6      ; (F6) = a*X(i) + Y(i)
        S.D      F6,0(R2)      ; Y(i) = a*X(i) + Y(i)
        DADDIU   R1,R1,#8      ; increment X index
        DADDIU   R2,R2,#8      ; increment Y index
        DSLTU    R3,R1,R4      ; test: continue loop?
        BNEZ     R3,foo        ; loop if needed
```

Assume the functional unit latencies as shown in the table below. Assume a one-cycle delayed branch that resolves in the ID stage. Assume that results are fully bypassed.

Instruction Producing Result	Instruction using result	Latency in clock cycles
FP Multiply	FP ALU op	6
FP Add	FP ALU op	4
FP Multiply	FP Store	5
FP Add	FP Store	4
Integer operations and all loads	Any	2

A. Assume a single-issue pipeline. Show how the loop would look both unscheduled by the compiler and after compiler scheduling for both floating-point operation and branch delays, including any stalls or idle clock cycles. What is the execution time (in cycles) per element of the result vector, Y, unscheduled and scheduled? How much faster must the clock be for processor hardware alone to match the performance improvement achieved by the scheduling compiler? (Neglect any possible effects of increased clock speed on memory system performance.) (**25 points**)

B. Assume a single-issue pipeline. Unroll the loop as many times as necessary to schedule it without any stalls, collapsing the loop overhead instructions. How many times must the loop be unrolled? Show the instruction schedule. What is the execution time per element of the result? (**25 points**)

2. We will look at how variations on Tomasulo's algorithm perform when running the loop from Question 1. The functional units (FUs) are described in the table below.

FU Type	Cycles in EX	Number of FUs	Number of reservation station
Integer	1	1	5
FP adder	10	1	3
FP Multiplier	15	1	2

Assume the following:

- Functional units are not pipelined.
- There is no forwarding between functional units; results are communicated by the common data bus (CDB).
- The execution stage (EX) does both the effective address calculation and the memory access for loads and stores. Thus, the pipeline is IF/ID/IS/EX/WB.
- Loads require one clock cycle.
- The issue (IS) and write-back (WB) result stages each require one clock cycle.
- There are five load buffer slots and five store buffer slots.
  Assume that the Branch on Not Equal to Zero (BNEZ) instruction requires one clock cycle.

A. For this problem use the single-issue Tomasulo MIPS pipeline of Figure 3.6 (*in the textbook*) with the pipeline latencies from the table above. Show the number of stall cycles for each instruction and what clock cycle each instruction begins execution (i.e., enters its first EX cycle) for three iterations of the loop. How many cycles does each loop iteration take? Report your answer in the form of a table with the following column headers:
   - Iteration (loop iteration number)
   - Instruction
   - Issues (cycle when instruction issues)
   - Executes (cycle when instruction executes)
   - Memory access (cycle when memory is accessed)
   - Write CDB (cycle when result is written to the CDB)
   - Comment (description of any event on which the instruction is waiting)

   Show three iterations of the loop in your table. You may ignore the first instruction (**30 points**)

3. Consider a branch-target buffer that has penalties of zero, two, and three clock cycles for correct conditional branch prediction, incorrect prediction, and a buffer miss, respectively. Consider a branch-target buffer design that distinguishes conditional and unconditional branches, storing the target address for a conditional branch and the target instruction for an unconditional branch.

   A. What is the penalty in clock cycles when an unconditional branch is found in the buffer? (**10 points**)

   B. Determine the improvement from branch folding for unconditional branches. Assume a 90% hit rate, an unconditional branch frequency of 5%, and a two-cycle penalty for a buffer miss. How much improvement is gained by this enhancement? How high must the hit rate be for this enhancement to provide a performance gain**?** (**10 points**)