**Problem 1** An (m,n) correlating branch predictor uses the behavior of the most recent m executed branches to choose from $2^m$ predictors, each of which is an n-bit predictor. A two-level local predictor works in a similar fashion, but only keeps track of the past behavior of each individual branch to predict future behavior.

There is a design trade-off involved with such predictors: correlating predictors require little memory for history, which allows them to maintain 2-bit predictors for a large number of individual branches (reducing the probability of branch instructions reusing the same predictor), while local predictors require substantially more memory to keep history and are thus limited to tracking a relatively small number of branch instructions.

For this exercise, consider a (1,2) correlating predictor that can track four branches (requiring 16 bits) versus a (1,2) local predictor that can track two branches using the same amount of memory. For the following branch outcomes, provide each prediction, the table entry used to make the prediction, any updates to the table as a result of the prediction, and the final misprediction rate of each predictor. Assume that all branches up to this point have been taken. Initialize each predictor to the following:

Table 1: branch outcomes

| Branch PC (word address) | Outcome |
|---|---|
| 454 | T |
| 543 | NT |
| 777 | NT |
| 543 | NT |
| 777 | NT |
| 454 | T |
| 777 | NT |
| 454 | T |
| 543 | T |

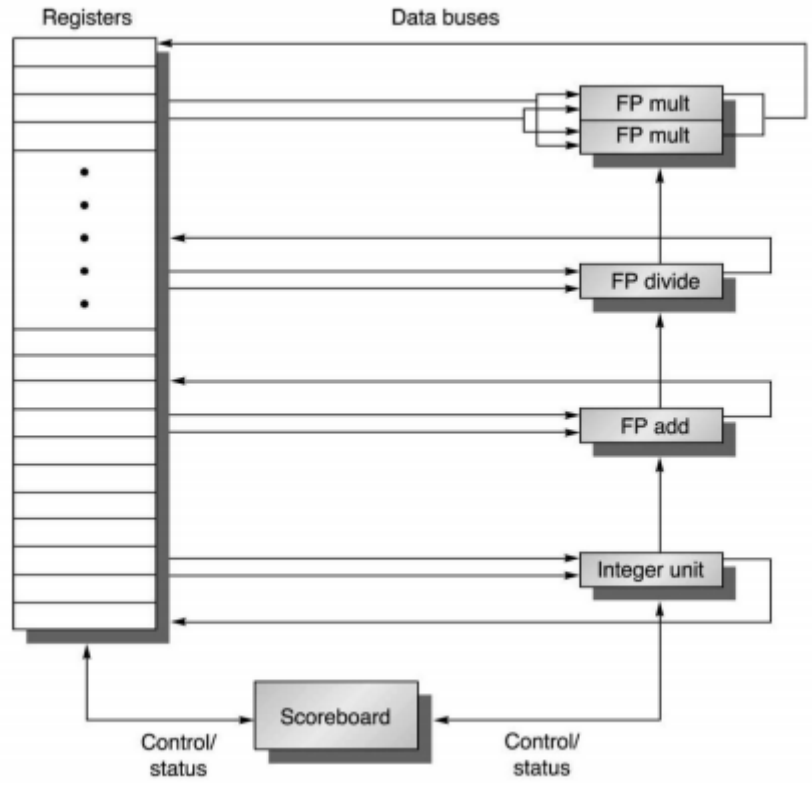
Table 2: Correlating predictor

| Correlating predictor | | | |
|---|---|---|---|
| Entry | Branch | Last outcome | Prediction |
| 0 | 0 | T | T with one misprediction |
| 1 | 0 | NT | NT |
| 2 | 1 | T | NT |
| 3 | 1 | NT | T |
| 4 | 2 | T | T |
| 5 | 2 | NT | T |
| 6 | 3 | T | NT with one misprediction |
| 7 | 3 | NT | NT |

Table 3: Local predictor

| Local predictor | | | |
|---|---|---|---|
| Entry | Branch | Last 2 outcomes (right is most recent) | Prediction |
| 0 | 0 | T,T | T with one misprediction |
| 1 | 0 | T,NT | NT |
| 2 | 0 | NT, T | NT |
| 3 | 0 | NT | T |
| 4 | 1 | T, T | T |
| 5 | 1 | T,NT | T with one misprediction |
| 6 | 1 | NT,T | NT |
| 7 | 1 | NT,NT | NT |

**Problem 2** The scoreboard algorithm is a disorderly execution algorithm proposed by CDC company in the last century. Taking CDC 6600 as an example, its structure is shown in the following figure, which includes two floating-point multiplication modules, one floating-point division module, one floating-point addition module, and one integer module (used to calculate the addresses of **lw** and **sw** instructions)




The diagram about CDC 6600

An instruction is executed in four stages, including "**transmit**, **read**, **execute**, and **write back**" . Note that we assume that the LD instruction execution takes 1 clock cycle, floating-point multiplication takes 10 clock cycles, and floating-point add takes 2 clock cycles.

(a) Please write the cycles that each stage belongs to when executing the instructions shown in the following table (starting from 1).

| Op | dest | j | k | Issue | Read Oper | Exec Comp | Write Result |
|---|---|---|---|---|---|---|---|
| LD | F6 | 34 | R2 | | | | |
| LD | F2 | 45 | R3 | | | | |
| MULD | F0 | F5 | F2 | | | | |
| MULD | F7 | F2 | F6 | | | | |
| ADDD | F6 | F8 | F7 | | | | |

(b) When in the ninth clock cycle, try filling in the following Functional Unit status chart.

| Time | Name | Busy | Op | $F_i$ | $F_j$ | $F_k$ | $Q_j$ | $Q_k$ | $R_j$ | $R_k$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | Integer | | | | | | | | | |
| | Mult1 | | | | | | | | | |
| | Mult2 | | | | | | | | | |
| | Add | | | | | | | | | |

**Problem 3** Suppose we have a deeply pipelined processor,for which we implement a branch-target buffer for the conditional branches only.Assume that the misprediction penalty is always four cycles and the buffer miss penalty is always three cycles.Assume an 85% hit rate, 90% accuracy, and 20% branch frequency.How much faster is the processor with the branch-target buffer versus a processor that has a fixed two-cycle branch penalty?Assume a base clock cycle perinstruction(CPI) without branch stalls of one.

**Problem 4** Consider a branch-target buffer that has penalties of zero, two, and two clockcycles for correctconditional 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 unconditionalbranch.

(a) What is the penalty in clock cycles when an unconditional branch is foundin the buffer?

(b) Determine the improvement from branch folding for unconditional branches. Assumea 80% hit rate, an unconditional branch frequency of 10%, and a two-cycle penalty for a buffer miss.How much improvement is gained by this enhancement?

**Problem 5** In this exercise,we will look at how variations on Tomasulo's algorithm perform when running the DAXPY loop shown in Loop 1.

```
        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
```

Loop 1: DAXPY(double-precision aX plus Y) loop

The functional units (FUs) are described in the table below.

| FU type | Cycles in EX | Number of FUs | Number of reservation stations |
|---|---|---|---|
| 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.

For this problem use the single-issue Tomasulo MIPS pipe-line of Figure 3.6 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 two iterations of the loop in your table.** You may ignore the first instruction.

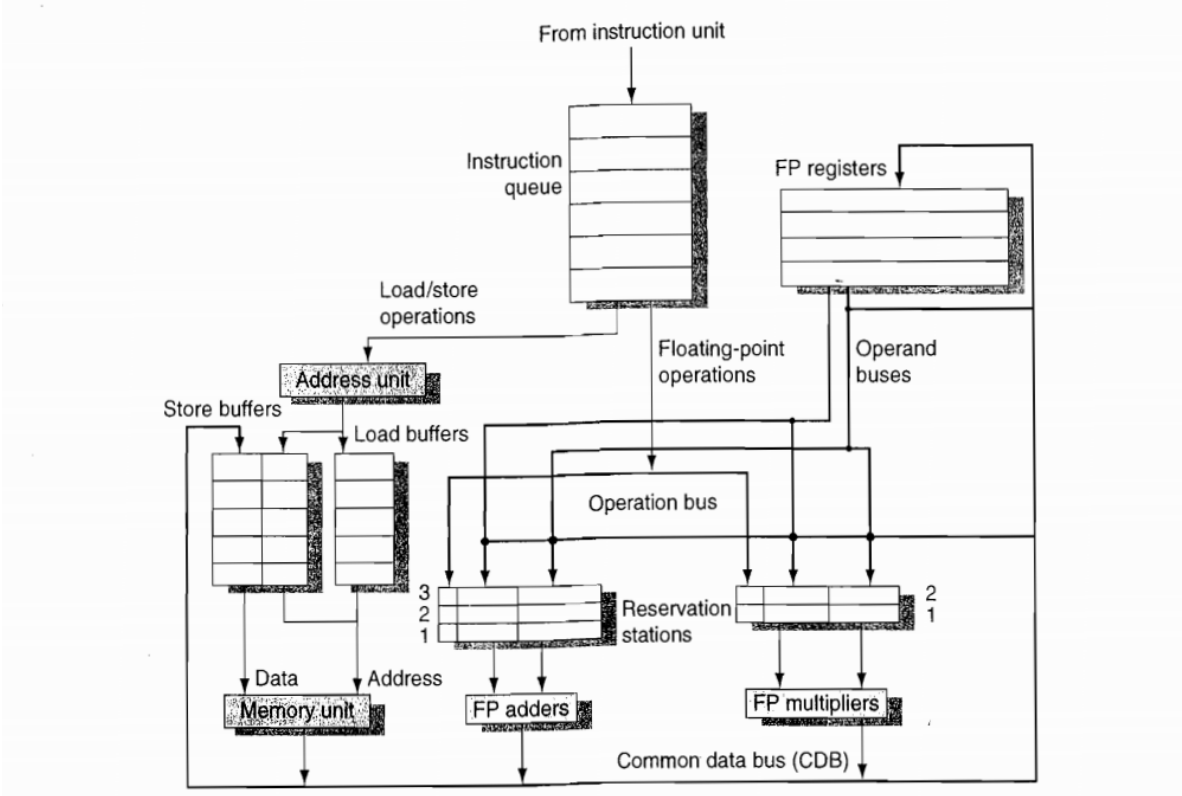



**Figure 3.6 The basic structure of a MIPS floating-point unit using Tomasulo's algorithm.** Instructions are sent from the instruction unit into the instruction queue from which they are issued in first-in, first-out (FIFO) order. The reservation stations include the operation and the actual operands, as well as information used for detecting and resolving hazards. Load buffers have three functions: (1) hold the components of the effective address until it is computed, (2) track outstanding loads that are waiting on the memory, and (3) hold the results of completed loads that are waiting for the CDB. Similarly, store buffers have three functions: (1) hold the components of the effective address until it is computed, (2) hold the destination memory addresses of outstanding stores that are waiting for the data value to store, and (3) hold the address and value to store until the memory unit is available. All results from either the FP units or the load unit are put on the CDB, which goes to the FP register file as well as to the reservation stations and store buffers. The FP adders implement addition and subtraction, and the FP multipliers do multiplication and division.