

- 1) (POINTS 14/40) Consider the following snippet of code running a processor that uses the Tomasulo's algorithm to perform the dynamic scheduling of instructions. The program performs the operation $Y=aX/Y$ on a vector of 100 elements. Initially, $R1 = 0$ and $F0$ contains the value of the constant 'a'.

```

etic:  L.D    F2, 0(R1)      ; read Xi
        MUL.D F4, F2, F0    ; multiply a*Xi
        L.D    F6, 400(R1)  ; load Yi
        DIV.D  F6, F4, F6    ; a*Xi/Yi
        S.D    F6, 400(R1)  ; store Yi
        ADDI   R1, R1, 8     ; update R1
        SGTI   R3, R1, 800   ; R1 >? 800, result in R3
        BEQ    R3, R0, etic  ; continue to loop if false
    
```

Working hypothesis:

- the pipeline implements a single-dispatch policy
- the instructions after a branch are executed speculatively and predicted 'taken'
- the issue stage (I) calculates the address of the actual reads and writes;
- reads require 1 clock cycle; writes require 0 clock cycles (write buffer + split-cache);
- there's only one CDB
- dispatch stage (D) and complete stage (C) require 1 clock cycle
- there are separated integer units for the calculation of the actual address, for arithmetic and logical operations, for the evaluation of the branch condition
- the functional units do not take advantage of pipelining techniques internally (reservation stations are busy until the end of CDB-write, except for Stores)
- the load buffer has 5 slots
- the store buffer has 5 slots (writes wait for the operand in the store buffer, i.e. in the execution stage)
- the rest of the processor and has the following characteristics

Type of Functional Unit	No. of Functional Units	Cycles for stage I	No. of reservation stations
Integer (effective addr.)	1	1	2
Integer (op. A-L)	1	1	2
Integer (branch calc.)	1	1	2
FP Adder	1	4	3
FP Multiplier	1	8	3
FP Divider	1	15	2

Complete the following chart until the end of the third iteration of the code fragment above in the case of simple dynamic scheduling.

Iter.	Instruction	P disPatch (start-stop)	I Issue (start-stop)	M MEM ACCESS (clock)	W CDB-Write (Complete) (clock)	C Commit (clock)	Comments
1	L.D F2, 0(R1)	1-4	2	3	4	5	
...	...						
...	...						

- 2) (POINTS 10/40) For the same fragment of code of exercise 1, let's assume a single-pipeline processor such that the branch condition is solved in the decode stage, so that we have only 1 cycle for the delay slot.

Moreover, let's assume that:

- The dispatch and complete stage requires 1 cycle
- There are the following latencies between operations:

Producer Instruction	Consumer Instruction	Latency (clock cycles)
FP operation	FP operation	4
FP operation	Store double	2
Load double	FP operation	2
Load double	Store double	1

The pipeline is single-dispatch: calculate the execution time (in cycles) of a single loop and show where there are stalls with and without static scheduling of the instructions (without unrolling techniques).

- 3) (POINTS 8/40) Explain the operation and draw a diagram of a PAg branch 2-level predictor with a 12-bit BSHR and size $2^{12} \times 2$ bit for the PHT.
- 4) Given the sequence P1: R, P2: R, P3: R, P1: W, P2: W, P3: W ($P_x:R$ = read by the processor P_x , $P_x:W$ write by the processor P_x), with respect to a certain variable 'a', show for each processor the sequence of states, and transactions on the bus that occur in a multiprocessor UMA with write-back caches for each processor and DRAGON coherence protocol.