Assignment # 3

1. Consider a single-issue, 5-stage pipeline design. For all instructions, fetching (F), decoding (D), memory (M), and write-back (W) take 1 cycle each. The execution time is 3 cycles for multiplication (E_MUL), 6 cycles for division (E_DIV), and 1 cycle for any other executions (E). The processor has an integer ALU, a multiplier and a divider. Furthermore, the memory (M) and writeback (W) stages (and only those) can accommodate multiple instructions without a structural hazard, as long as only one instructions needs memory access or access to registers, respectively. If more than one instruction requires access to memory or registers, preference is given to instructions in issuing order (while the other(s) stall(s)).

   (a) Show the forwarding and stalls for the following MIPS code for one loop. Indicate stalls by (S) and forwarding by clearly visible arrows between the appropriate states.

   (b) After heavy optimization, multiplication and division are merged into one unit that completes execution in 2 cycles. The processor now only has an integer ALU and one combined MUL/DIV engine.
   Show the forwarding and stalls for the following MIPS code for one loop. Compare the result to the previous engine. Assuming the second engine to be clocked 10% faster, express the performance gain for the given loop. Assume the execution time of the loop to end after the fetch of the last instruction.

```
Loop:  LD    F2, 0(R1)
       DIV   F8, F2, F0
       MULT  F2, F6, F2
       LD    F4, 0(R2)
       ADD   F4, F0, F4
       ADD   F10, F8, F2
       ADDI  R1, R1, 8
       ADDI  R2, R2, 8
       SD    F4, 0(R2)
       SUB   R20, R4, R1
       BNZ   R20, Loop
```

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
LD F2, 0(R1)																			
DIV F8, F2, F0																			
MULT F2, F6, F2																			
LD F4, 0(R2)																			
ADD F4, F0, F4																			
ADD F10, F8, F2																			
ADDI R1, R1, 8																			
ADDI R2, R2, 8																			
SD F4, 0(R2)																			
SUB R20, R4, R1																			
BNZ R20, Loop																			

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
LD F2, 0(R1)																			
DIV F8, F2, F0																			
MULT F2, F6, F2																			
LD F4, 0(R2)																			
ADD F4, F0, F4																			
ADD F10, F8, F2																			
ADDI R1, R1, 8																			
ADDI R2, R2, 8																			
SD F4, 0(R2)																			
SUB R20, R4, R1																			
BNZ R20, Loop																			

2. For this problem use the single-issue Tomasulo MIPS pipeline with the number of functional units and reservation stations listed in the table below

FU type	Execution Time	Number of FUs	Number of reservation stations
Integer (LD, SD, ADDI)	1	1	3
FP adder	3	1	2
FP multiplier	6	1	2

When applying Tomasulos algorithm, consider the following arbitration rules:

- *Only one instruction can be issued per clock cycle.*
- Function units are not pipelined.
- The execution stage (EX) does both the effective address calculation and the memory access for loads and stores. Thus the pipeline is IS / EX / WB.
- All stages (IS, WB) except EX take 1 cycle to complete.
- There is no forwarding between functional units. Both integer and floating point results are communicated through the CDB.
- Memory accesses use the integer functional unit to perform effective address calculation. All loads and stores access memory during the EX stage.
- If instruction A has a read-after-write dependency on instruction B, instruction A can begin execution only AFTER instruction B was in WB stage.
- Only one instruction can write to the CDB in a clock cycle.
- Branches and stores do not need the CDB.
- Whenever there is a conflict for a functional unit or the CDB, assume program order.
- When an instruction is done executing in its functional unit and is waiting for the CDB, it is still occupying the functional unit and its reservation station (meaning no other instruction may enter).

Using Tomasulos algorithm, fill in the table below to indicate the cycle number at each stage for all instructions.

The reservation station table is provided for reference only.

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Integer 1							
Integer 2							
Integer 3							
FP Add 1							
FP Add 2							
FP Mul 1							
FP Mul 2							

Instruction	Issue	Execution	Write-back	Reason to Stall
LD F0, 0(R0)				
MUL.D F1, F0, F1				
ADD.D F0, F0, F2				
SD F1, 0(R1)				
LD F2, 4(R0)				
MUL.D F3, F2, F1				
ADD.D F4, F1, F2				
SD F3, 4(R3)				
ADDI F2, F2, 1				
ADD.D F0, F4, F1				