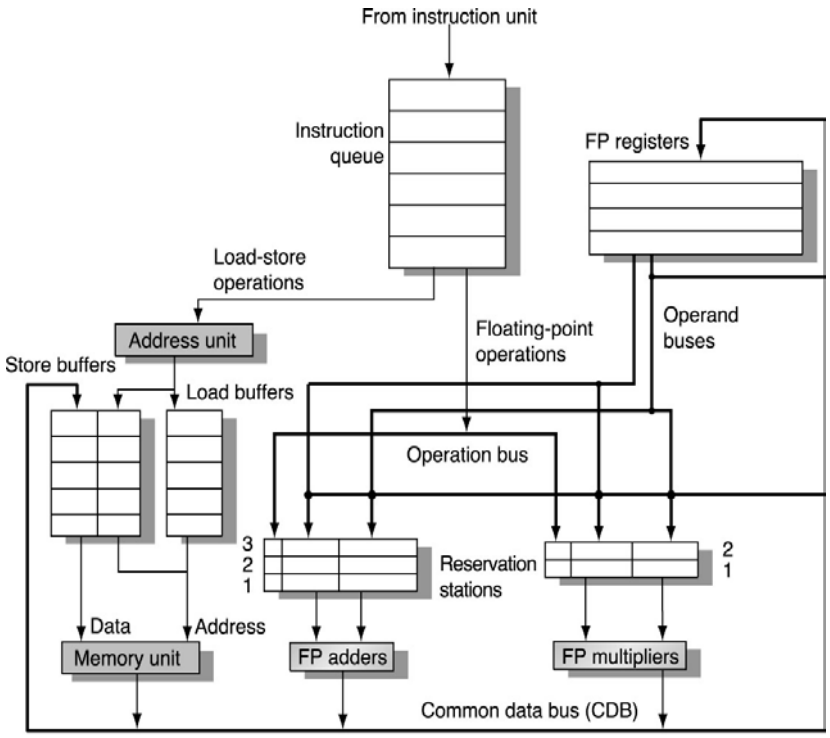
Hardware Based Scheduling

1. Consider a dynamically scheduled machine using Tomasulo's algorithm and the datapath shown below. The functional unit latencies are as follows: load/store – 2 cycles. FP ADD – 3 cycles, FP Mult – 5 cycles, Integer/Branch operations – 1 cycle. There is one each of a load/store unit, FP ADD, FPMULT, and integer unit. All units have one reservation station and all units are pipelined. If an operand is written on one cycle, dependent instructions will start executing on the next cycle.



   a. Using Tomasulo's algorithm fill in the table below to show the clock cycle on which each instruction progresses through the corresponding functional unit. Start at clock cycle 0.

The tag generated for an operand is unique to the reservation station and therefore the reservation station cannot be de-allocated when an instruction in the reservation is issued and begins executing. The presence of one reservation station per unit affects the load and store instructions which have to stall on a structural hazard

Solution I	Issue	Execute	WB
Code	**Issue**	**Execute**	**WB**
L.D F2, 0(R1)	0	1-2	3
MUL.D F4, F2, F0	1	4-8	9
L.D F6, 0(R2)	4	5-6	7
ADD.D F6, F4, F6	5	10-12	13
S.D F6, 0(R2)	8	14-15	
DADDUI R1, R1, #8	9	10	11
DADDIU R2, R2, #-8	12	13	14
BGT R1, #800	15	16	

b. Now imagine the datapath has included 4 entry ROB as shown below. Fill in the state of the ROB when the ADD.D instruction is ready to commit (but before it does). The column for **Complete?** Records whether execution has completed.

Entry	Instruction	Destination	Complete?
1	S.D F6, 0(R2)	M[R2 + 0]	NO
2	DADDUI R1, R1, #8	R1	YES
3	DADDIU R2, R2, #-8	R2	YES
4	ADD.D F6, F4, F6	F6	YES

c. With the four entry ROB, on what cycle could the bgt instruction issue? Briefly explain.

As early as cycle 14, after the ADD.D commits.

2. Consider the following code sequence.

```
1.      LOOP: LD F2, 0(R1)
2.            LD F4, 0(R2)
3.            MULT F6, F2, F4
4.            SUBD F4, F6, F10
5.            ADDD F8, F8, F4
6.            DADDIU R1, R1, #-8
7.            DADDIU R2, R2, #-8
8.            BNEZ R1, R4 LOOP
```

a. Using Tomasulo's algorithm fill in the table below to show the clock cycle on which each instruction progresses through the corresponding functional unit. Assume the first cycle is numbered 0. Use the architecture parameters from the previous problem with two reservation stations per functional unit. Note execution begins on the cycle following the issue of an instruction to the reservation station.

Instruction	Issue	Execute	WB
LD F2, 0(R1)	0	1-2	3
LD F4, 0(R2)	1	2-3	4
MULT F6, F2, F4	2	5-9	10
SUBD F4, F6, F10	3	11-13	14
ADDD F8, F8, F4	4	15-17	18
DADDIU R1, R1, #-8	5	6	7
DADDIU R2, R2, #-8	6	7	8
BNEZ R1, R4 LOOP	8	9	

b. For which instructions are the results not written back to the register file?

LD F4, 0(R2)

The WAW on F4 ensures that the status of F4 is updated to wait on the write result from SUBD F4, F6, F10. On cycle 4 when the LD instruction writes back the result, the register does not load the value since it is now expecting an instruction from the SUBD instruction.

c. Now imagine this loop body is scheduled with a scoreboard. On what cycle will the ADDD instruction be issued?

7 – Note that WAW hazard between instructions 2 & 4 which prevents 4 from being issued until the hazard clears on cycle 5. Thus instruction 4 can issue on cycle 6 and instruction 5 can then issue on cycle 7. (Note in the scoreboard there is an additional RO stage in the table!).

d. Now imagine the unrolled loop body is statically scheduled on a two way superscalar machine where issue restrictions permit a memory operation, integer operation or a branch operation to be paired with a floating point operation. Devise the best schedule you can for the body of one iteration of the loop.

Only the LD or DADDIU instructions can be paired with any of the floating point instructions. Concurrency between remaining pairs of instructions is limited by RAW and issue constraints.

3. For this question consider a machine with the following properties. The functional unit latencies are as follows: FP load/store – 2 cycles. FP ADD – 3 cycles, FP Mult – 5 cycles, Integer operations (including branches) – 1 cycle. There is one of each type of functional unit. Except for the integer unit, all of the other units have two reservation stations. Reservation station 1 is always allocated first. The re-order buffer has 4 entries addressed R0, R1, R2, R3 with R0 being the first buffer allocated after execution starts buffers are allocated sequentially. Consider the following code sequence scheduled on a processor using Tomasulo's algorithm. Show the contents of the re-order buffer when the MULTD instruction from the first iteration is ready to commit. Assume an instruction in a reservation station will start execution on the cycle after the instruction supplying data writes back the value over the CDB. Indicate in the table below the instruction occupying that re-order buffer slot and whether it has completed execution and waiting to commit.

```
FOO:    LD      F0, 0(R1)
        MULT    F2, F0, F4
        LD      F4, 4(R1)
        ADD     F2, F2, F4
        SUBI    R1, R1, 8
        SW      0(R1), F2
        BNZ     R1, FOO
        SW      8(R1), F2
```

	Done?	*Instruction*
R0	Y	SUBI R1, R1, 8
R1	N	MULTD F2, F0, F4
R2	Y	LD F4, 0(R1)
R3	N	ADD F2, F2, F4

4. Consider the following code and scheduling on a superscalar processor using Tomasulo's algorithm.

```
FOO:    1.    L.D       F0, 0(R1)
        2.    MUL.D     F2, F0, F6
        3.    L.D       F4, 0(R2)
        4.    ADD.D     F2, F2, F4
        5.    S.D       F2, 0(R2)
        6.    DADDIU    R2, R2, #-8
        7.    DADDIU    R1, R1, #-8
        8.    BNE       R1, R3, FOO
```

   a. Can instruction 4 be issued before instruction 2 completes? If so how is the WAW hazard avoided?

The tag in register F2 will be updated when instruction 4 is issued. The update will record the tag value of the reservation station holding instruction 4. Thus even I instruction 2 completes after instruction 4, it will not errorneously update F2, and the correct order of completion of instructions 2 and 4 will be preserved. Note that instructions dependent on instruction 2 will have the tag of the reservation station that was allocated to instruction 2.

   b. What is the maximum ILP within the loop body that software pipelining can produce?

The number of instructions in the loop body.

5. Explain
   a. how WAW hazard is avoided when using an implementation of Tomasulo's algorithm. Use the following code sequence as an example.

```
ADD.D    F0, F2, F4
..
..
SUB.D    F0, F6, F8
```

When the SUB.D instruction is issued the tag stored with register F0 is updated to reflect the source producing the value of F0. Thus when instructions are issued in order, WAW hazards are avoided during execution since completion of the ADD.D instruction after the completion of the SUB.D is ignored.

   b. how WAR hazards are avoided when using the scoreboard. Use the following code sequence as an example.

```
ADD.D    F0, F2, F4
..
..
MUL.D    F2, F6, F8
```

The MUL.D instruction can be issued. All write operations are first checked during the write-back stage to ensure that they can complete. The scoreboard prevents the write to F2 from occurring prior to the read by the ADD.D instruction.

6. Attached is a figure that contains all of the data structures for Tomasulo's algorithm and a code sequence. Show the contents of the data structures when the first MUL.D has completed execution but has not yet written its result over the CDB. The functional unit latencies are as follows: FP load/store – 2 cycles. FP ADD – 3 cycles, FP Mult – 5 cycles, Integer operations – 1 cycle. You have as many functional units as you need, i.e., there will be no structural hazards on functional units. The ADD.DI (FP ADD using an immediate operand) behaves the same as an ADD.D instruction. **Assume all reservation stations are de-allocated only after the instruction completes!** An extra copy of the data structures below is provided as an attached worksheet. Please only enter your final answer on this page.

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	0	R1	0	1-2	3
MUL.D	F8	F6	F0	1	4-8	
MUL.D	F10	F4	F2	2	3-7	8
ADD.D	F8	F8	F10	3		
ADD.DI	F6	F4	0	4	5-7	
S.D	F8	0	R2	5		
DADDUI	R1	8	R1			
DADDUI	R2	8	R2			
DADDUI	R3	-1	R3			
BNEZ	R3	Loop				

	Busy	Address	
Load1	Y	R2+0	(for SD)
Load2			
Load3			

Note potential conflicts on the CDB at cycle 8

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	Y	Add	-	[mult2]	mult1	
	Add2	Y	Add	F4	0	-	-
	Int2						
	Int1	Y	Add	8	R2		
	Mult1	Y	Add	8	R1		
	Mult2						

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
8	*FU*				ADD2	ADD1	MULT2			

7. Consider a machine that uses Tomasulo's algorithm to achieve out-of-order execution. This machine has the following *functional units*:

   Load: 2 reservation stations, latency = 2 cycles

   MUL: 1 reservation station, latency = 5 cycles

   DIV: 1 reservation station, latency = 44 cycles

   ADD/SUB: 1 reservation station, latency = 3 cycles

   Assume that writeback on the CDB happens in the cycle after the functional unit completes execution.

   The following sequence of instructions is executed on this machine:

1	L.D	F6, 44(R1)
2	MUL.D	F4, F6, F2
3	DIV.D	F8, F4, F9
4	ADD.D	F10, F8, F14
5	ADD.D	F8, F22, F24
6	SUB.D	F4, F20, F26

a) Show the dependence arcs between the instructions (identify: flow, anti, output), and identify the registers involved.

1->2 flow F6

2->3 flow F4

2->6 output F4

3->6 anti F4

3->4 flow F8

3->5 output F8

4->5 anti F8

b) Fill in the following table. *The first cycle is cycle-0.*

			Issue	Execute	Writeback
1	L.D	F6, 44(R1)	0	1-2	3
2	MUL.D	F4, F6, F2	1	4-8	9
3	DIV.D	F8, F4, F9	2	10-53	54
4	ADD.D	F10, F8, F14	3	55-57	58
5	ADD.D	F8, F22, F24	59	60-62	63
6	SUB.D	F4, F20, F26	64	65-67	68

c) If **scoreboarding** were used on this machine, in what cycle would *instruction 5* issue?

			Issue	ReadOp	Execute	Writeback
1	L.D	F6, 44(R1)	0	1	2-3	4
2	MUL.D	F4, F6, F2	1	5	6-10	11
3	DIV.D	F8, F4, F9	2	12	13-56	57
4	ADD.D	F10, F8, F14	3	58	59-61	62
5	ADD.D	F8, F22, F24	**63**	X	X	X
6	SUB.D	F4, F20, F26	X	X	X	X

Instr-5 issues in cycle-63 and not cycle-58 (after DIV writes-back, WAW hazard ends) due to structural hazard on Add Functional Unit with Instr-4.

d) Fill in the following data structures at the conclusion of cycle 14.

Reservation Stations

	Busy	*Address*
Load 1		

Register Result Status

	Vj	*Qj*
F0		
F1		

	Busy	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
MUL	***No***					
	Yes	***DIV***	***[F4]***	***[F9]***		
DIV						

Note: In the (DIV, Vj) entry, it is also correct if ***[result value from previous MUL (Instr-2)]*** is used instead of ***[F4]***

Note: The provided Register Result Status table does not explicitly show the required entries to reflect the state of all relevant registers. It is enough to specify the required registers. The set would normally appear as follows.

Reg#	*Vj*	*Qj*
R1	[R1]	
F4	[F4]	
F6	[F6]	
F8		[DIV]
F10		[ADD]
F14	[F14]	

F20 [F20]

F22 [F22]

F24 [F24]

e) When instruction 4 is issued, what tags are assigned to operands in the assigned reservation station? How does the issue logic determine the tags?

RS	Busy	Op	Vj	Vk	Qj	Qk
ADD	Yes	ADD		[F14]	[DIV]	

The issue logic determines this by examining the status of registers F8 and F14 (the sources of Instr-4) in the Register File. The value of F14 is available in the reg-file, while F8 is waiting to get its value from the previous DIV instruction that has not completed yet. This is indicated by a tag for DIV in the status for F8.

f) The described machine dispatches upto one instruction per cycle from the instruction queue to the reservation stations. Can the instructions be dispatched out-of-order? Justify briefly.

NO. The Register File maintains either the most up-to-date values for each register (field Vj), or else, a tag identifying the Reservation-Station that corresponds to the instruction that will produce a value for that register. When an instruction is being issued to a Reservation Station, dispatch uses this information from the Register File to assign available operand values, or, tags identifying Reservation Stations holding instructions that will produce the operand values. Hence, in-order issue is required to ensure that no flow dependences are violated. For example, if there is a flow-dependence between two adjacent instructions, then issuing the consumer instruction earlier would lead its operands to get either old-values or old-tags from the register file, instead of the result-value or tag for the producer instruction.

Note:

Note that out-of-order issue need not violate output or anti-dependences (name-dependences), so long as the dispatch logic is augmented to check for whether values and tags in the register-file were produced by an instruction that occurs later in program order than the one currently being issued. And, the register-file is capable of holding tags and values from all instructions currently dispatched out-of-order (that is multiple entries per register). But there is still no way around the flow-dependence (true-dependence) constraint, so out-of-order issue is not permitted dues to flow dependencies.

8. Consider a dynamically scheduled machine. The functional unit latencies are as follows: load/store – 3 cycles. FP ADD – 5 cycles, FP Mult – 10 cycles, Integer/Branch operations – 2 cycles. There is one load/store unit, two FP ADD/SUB units, and one FPMULT. If an operand is written on one cycle, dependent instructions will start executing on the next cycle.

Consider the following code sequence.

```
L.D F2, 20(R1)
L.D F4, 30(R2)
MUL.D F6, F2, F4
SUB.D F4, F2, F4
L.D F8,40(R2)
ADD.D F4, F2, F2
SUB.D F8, F4, F8
S.D F8,20(R2)
S.D F6,30(R1)
```

a) Assuming the use of a scoreboard, fill in the table below to show the clock cycle on which each instruction progresses through the corresponding execution stage. Assume the first cycle is numbered 1. Also assume reading operands takes one cycle to complete and operands become available one cycle following write backs on previous instructions.

Code	Issue	Read Ops	Execute	WB
L.D F2, 20(R1)	1	2	3-5	6
L.D F4, 30(R2)	7	8	9-11	12
MUL.D F6, F2, F4	8	13	14-23	24
SUB.D F4, F2, F4	13	14	15-19	20
L.D F8, 40(R2)	14	15	16-18	19
ADD.D F4, F2, F2	21	22	23-27	28
SUB.D F8, F4, F8	22	29	30-34	35
S.D F8, 20(R2)	23	36	37-39	
S.D F6, 30(R1)	40	41	42-44	

Note structural hazards due the presence of a single functional unit of a specific type as well as WAW hazards. Also note that there are 2 FP ADD/SUB functional units.

b) Rewrite the above code sequence using a minimal number of additional registers to eliminate all WAR and WAW hazards.

```
L.D F2, 20(R1)
L.D F4, 30(R2)
MUL.D F6, F2, F4
SUB.D F10, F2, F4
L.D F8,40(R2)
ADD.D F12, F2, F2
SUB.D F14, F12, F8
S.D F14,20(R2)
S.D F6,30(R1)
```

Three new reigsters F10,F12 and F14 are introduced to eliminate all WAR and WAW hazards.

c) Now consider using Tomasulo's algorithm. Each functionalunit has one reservation station and all units are pipelined. Note that execution begins on the cycle following the issue of an instruction to the reservation station. When can the first ADD.D instruction begin executing and what is (functionally) the value of the tag stored in register F4 after this instruction has been issued? How does this issue time compare to the issue time when using a scoreboard. Explain any differences.

Code	Issue	Execute	WB
L.D F2, 20(R1)	1	2-4	5
L.D F4, 30(R2)	6	7-9	10
MUL.D F6, F2, F4	7	11-20	21
SUB.D F4, F2, F4	8	11-15	16
L.D F8, 40(R2)	11	12-14	15
ADD.D F4, F2, F2	12	13-17	18
SUB.D F8, F4, F8	17	19-23	24
S.D F8, 20(R2)	18	25-27	
S.D F6, 30(R1)	28	29-31	

The destination register has a tag whose value is set to reservation station value of the second ADD.D instruction. Thus, a WAW hazard is avoided with the preceding SUB.D instruction and it can be issued earlier that when using a scoreboard scheme.

Note the presence of structural hazards on some functional units due to the presence of 1 reseravtion station. Also note that there are 2 FP ADD/SUB functional units.