Dynamic Scheduling Example 1

In this lecture, we covered Tomasulo's Algorithm in detail. Here's a detailed version of what we covered in class, with notes on the operations occurring in each cycle. This solution shows the state of the reservation stations, register status table, and the pipeline at every point.

Recall that the code being run is the following:

```
L.D       F6, 32(R2)
L.D       F2, 44(R3)
MUL.D     F0, F2, F4
SUB.D     F8, F6, F2
DIV.D     F10, F0, F6
ADD.D     F6, F8, F2
```

And we assume the following latencies: 3 cycles for ADD.D and SUB.D, 10 cycles for MUL.D, 40 cycles for DIV.D, 2 cycles (1 + 1) for L.D. Here's our setup. Note that we only show the register result status table for the floating point registers, as we have no integer operations in this program, and only show those registers used in this code (F0-F10).

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	N					
ALU2	N					
ALU3	N					
Mult1	N					
Mult2	N					

Register result status table (FP only)

F0	F2	F4	F6	F8	F10

Pipeline diagram

Cycle	1	2	3	4	5
L.D F6, 32(R2)					
L.D F2, 44(R3)					
MUL.D F0,F2,F4					
SUB.D F8, F6, F2					
DIV.D F10, F0, F6					
ADD.D F6, F8, F2					

Cycles 1-2:

No instructions issue until cycle 2, so we're showing the first two cycles together. Both loads are fetched during these cycles, and the first load issues during cycle 2. When we issue that load, we give it a reservation station (Load1) and rename its destination register (F6) by updating the register result status table to show that Load1 will produce the most up-to-date value of F6. Both operands for the load are ready (immediate value 32, register R2), so we can store those values in the Vj/Vk fields. Remember that the notation [R2] means "the value in register R2."

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	**Y**	**L.D**	**32**	**[R2]**		
Load2	N					
Load3	N					
ALU1	N					
ALU2	N					
ALU3	N					
Mult1	N					
Mult2	N					

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
			Load1		

Pipeline diagram

Cycle	1	2	3	4	5
`L.D F6, 32(R2)`	**IF**	**IS**			
`L.D F2, 44(R3)`		**IF**			
`MUL.D F0,F2,F4`					
`SUB.D F8, F6, F2`					
`DIV.D F10, F0, F6`					
`ADD.D F6, F8, F2`					

Cycle 3:

In this cycle, the first load reaches the EX stage, in which it calculates the address to access by adding 32 + [R2]. The second load issues in this cycle; we assign it to reservation station Load2 and rename its destination register (F2) in the register result status table. We also fetch the MUL.D operation in this cycle.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	Y	L.D	32	[R2]		
Load2	**Y**	**L.D**	**44**	**[R3]**		
Load3	N					
ALU1	N					
ALU2	N					
ALU3	N					
Mult1	N					
Mult2	N					

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
	Load2		Load1		

Pipeline diagram

Cycle	1	2	3	4	5
`L.D F6, 32(R2)`	IF	IS	**EX**		
`L.D F2, 44(R3)`		IF	**IS**		
`MUL.D F0,F2,F4`			**IF**		
`SUB.D F8, F6, F2`					
`DIV.D F10, F0, F6`					
`ADD.D F6, F8, F2`					

Cycle 4:

The first load moves to the MEM stage, actually performing its memory access. The second load calculates its address in its EX stage. Meanwhile, the MUL.D is issued to reservation station Mult1. We rename its destination (F0) as done with the loads. Note that one of its source operands, F4, is available, but the second source operand, F2, will be produced by the L.D instruction in Load2. We indicate this fact by setting the Qj field of Mult1 appropriately. The SUB.D instruction is fetched during this cycle.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	Y	L.D	32	[R2]		
Load2	Y	L.D	44	[R3]		
Load3	N					
ALU1	N					
ALU2	N					
ALU3	N					
Mult1	**Y**	**MUL.D**		**[F4]**	**Load2**	
Mult2	N					

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
Mult1	Load2		Load1		

Pipeline diagram

Cycle	1	2	3	4	5
`L.D F6, 32(R2)`	IF	IS	EX	**MEM**	
`L.D F2, 44(R3)`		IF	IS	**EX**	
`MUL.D F0,F2,F4`			IF	**IS**	
`SUB.D F8, F6, F2`				**IF**	
`DIV.D F10, F0, F6`					
`ADD.D F6, F8, F2`					

Cycle 5:

We have our first instruction complete during this cycle—the first L.D operation, which is using reservation station Load1. This instruction broadcasts its result on the common data bus (CDB). The one dependent instruction on this load is the SUB.D that's issuing during this cycle, so that instruction is able to read the value of F6 from the CDB. That broadcasts also leads to Load1 being cleared and reset to "not busy," and the entry for F6 in the register result status table being cleared as well.

Also during this cycle, the second L.D performs its memory access, the SUB.D is issued, and the DIV.D is fetched. The SUB.D is dependent on F2, which will come from the instruction in Load2, so it only has the value of one operand. Note that the MUL.D does nothing during this cycle, as it is dependent on the same load and must therefore stall until that instruction broadcasts its result on the CDB.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	**N**					
Load2	Y	L.D	44	[R3]		
Load3	N					
ALU1	**Y**	**SUB.D**	**[F6]**			**Load2**
ALU2	N					
ALU3	N					
Mult1	Y	MUL.D		[F4]	Load2	
Mult2	N					

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
Mult1	Load2			**ALU1**	

Pipeline diagram

Cycle	1	2	3	4	5
`L.D F6, 32(R2)`	IF	IS	EX	MEM	**WB**
`L.D F2, 44(R3)`		IF	IS	EX	**MEM**
`MUL.D F0,F2,F4`			IF	IS	**S**
`SUB.D F8, F6, F2`				IF	**IS**
`DIV.D F10, F0, F6`					**IF**
`ADD.D F6, F8, F2`					

Cycle 6:

The second load completes during this cycle. Two instructions—the SUB.D in ALU1 and the MUL.D in Mult1—depend on this instruction for a source operand. Since these two instructions occupy different functional units, both can start execution during this cycle. As with the first load, the CDB broadcast leads to the clearing of both the appropriate reservation station (Load2) and the field for the destination register (F2).

The DIV.D issues during this cycle, using reservation station Mult2. This instruction is dependent on the multiply, which will take 10 cycles to execute, so it will stall for several cycles after issue. We also fetch the ADD.D, the final instruction in this sequence.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	**N**					
Load3	N					
ALU1	Y	SUB.D	[F6]	**[F2]**		
ALU2	N					
ALU3	N					
Mult1	Y	MUL.D	**[F2]**	[F4]		
Mult2	**Y**	**DIV.D**		**[F6]**	**Mult1**	

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
Mult1				ALU1	**Mult2**

Pipeline diagram

Cycle	1	2	3	4	5	6
`L.D F6, 32(R2)`	IF	IS	EX	MEM	WB	
`L.D F2, 44(R3)`		IF	IS	EX	MEM	**WB**
`MUL.D F0,F2,F4`			IF	IS	S	**EX1**
`SUB.D F8, F6, F2`				IF	IS	**EX1**
`DIV.D F10, F0, F6`					IF	**IS**
`ADD.D F6, F8, F2`						**IF**

Cycle 7:

Both the MUL.D and SUB.D are executing during this cycle. The DIV.D stalls, as it is waiting for the multiply to finish. We issue our final instruction, the ADD.D, during this cycle. This instruction is dependent on the SUB.D in ALU1 and therefore cannot begin execution until the SUB.D finishes.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	Y	SUB.D	[F6]	[F2]		
ALU2	**Y**	**ADD.D**		**[F2]**	**ALU1**	
ALU3	N					
Mult1	Y	MUL.D	[F2]	[F4]		
Mult2	Y	DIV.D		[F6]	Mult1	

Register result status table (FP only)

F0	F2	F4	**F6**	F8	F10
Mult1			**ALU2**	ALU1	Mult2

Pipeline diagram

Cycle	1	2	3	4	5	6	7
`L.D F6, 32(R2)`	IF	IS	EX	MEM	WB		
`L.D F2, 44(R3)`		IF	IS	EX	MEM	WB	
`MUL.D F0,F2,F4`			IF	IS	S	EX1	**EX2**
`SUB.D F8, F6, F2`				IF	IS	EX1	**EX2**
`DIV.D F10, F0, F6`					IF	IS	**S**
`ADD.D F6, F8, F2`						IF	**IS**

Cycle 8:

Since both the MUL.D and SUB.D continue to execute in this cycle, the dependent instructions that follow them (DIV.D and ADD.D, respectively) continue to stall. Note that this is the last execution cycle for the SUB.D, which will complete in Cycle 9, thus allowing the ADD.D to proceed.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	Y	SUB.D	[F6]	[F2]		
ALU2	Y	ADD.D		[F2]	ALU1	
ALU3	N					
Mult1	Y	MUL.D	[F2]	[F4]		
Mult2	Y	DIV.D		[F6]	Mult1	

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
Mult1			ALU2	ALU1	Mult2

Pipeline diagram

Cycle	1	2	3	4	5	6	7	8
L.D F6, 32(R2)	IF	IS	EX	MEM	WB			
L.D F2, 44(R3)		IF	IS	EX	MEM	WB		
MUL.D F0,F2,F4			IF	IS	S	EX1	EX2	**EX3**
SUB.D F8,F6,F2				IF	IS	EX1	EX2	**EX3**
DIV.D F10,F0,F6					IF	IS	S	**S**
ADD.D F6,F8,F2						IF	IS	**S**

Cycle 9:

The SUB.D completes, clearing the appropriate reservation station (ALU1) and register result status table entry (F8). The ADD.D, which is dependent on this instruction, can now begin execution. The DIV.D continues to stall as the MUL.D continues execution.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	**N**					
ALU2	Y	ADD.D	**[F8]**	[F2]		
ALU3	N					
Mult1	Y	MUL.D	[F2]	[F4]		
Mult2	Y	DIV.D		[F6]	Mult1	

Register result status table (FP only)

F0	F2	F4	F6	**F8**	F10
Mult1			ALU2		Mult2

Pipeline diagram

Cycle	1	2	3	4	5	6	7	8	9
`L.D F6, 32(R2)`	IF	IS	EX	MEM	WB				
`L.D F2, 44(R3)`		IF	IS	EX	MEM	WB			
`MUL.D F0,F2,F4`			IF	IS	S	EX1	EX2	EX3	**EX4**
`SUB.D F8,F6,F2`				IF	IS	EX1	EX2	EX3	**WB**
`DIV.D F10,F0,F6`					IF	IS	S	S	**S**
`ADD.D F6,F8,F2`						IF	IS	S	**EX1**

Cycles 10-11:

Both the MUL.D and ADD.D continue execution during these cycles. The DIV.D remains stalled.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	N					
ALU2	Y	ADD.D	[F8]	[F2]		
ALU3	N					
Mult1	Y	MUL.D	[F2]	[F4]		
Mult2	Y	DIV.D		[F6]	Mult1	

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
Mult1			ALU2		Mult2

Pipeline diagram

Cycle	1	2	3	4	5	6	7	8	9	10	11
`L.D F6, 32(R2)`	IF	IS	EX	MEM	WB						
`L.D F2, 44(R3)`		IF	IS	EX	MEM	WB					
`MUL.D F0,F2,F4`			IF	IS	S	EX1	EX2	EX3	EX4	**EX5**	**EX6**
`SUB.D F8,F6,F2`				IF	IS	EX1	EX2	EX3	WB		
`DIV.D F10,F0,F6`					IF	IS	S	S	S	**S**	**S**
`ADD.D F6,F8,F2`						IF	IS	S	EX1	**EX2**	**EX3**

Cycle 12:

The ADD.D completes during this cycle, broadcasting its result on the CDB and clearing the appropriate reservation station (ALU2) and register result status table entry (F6). With no dependent instructions, this broadcast has no other effects. Meanwhile, the MUL.D continues to execute and the DIV.D continues to stall.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	N					
ALU2	**N**					
ALU3	N					
Mult1	Y	MUL.D	[F2]	[F4]		
Mult2	Y	DIV.D		[F6]	Mult1	

Register result status table (FP only)

F0	F2	F4	**F6**	F8	F10
Mult1					Mult2

Pipeline diagram

Cycle	1	2	3	4	5	6	7	8	9	10	11	**12**
`L.D F6, 32(R2)`	IF	IS	EX	MEM	WB							
`L.D F2, 44(R3)`		IF	IS	EX	MEM	WB						
`MUL.D F0,F2,F4`			IF	IS	S	EX1	EX2	EX3	EX4	EX5	EX6	**EX7**
`SUB.D F8,F6,F2`				IF	IS	EX1	EX2	EX3	WB			
`DIV.D F10,F0,F6`					IF	IS	S	S	S	S	S	**S**
`ADD.D F6,F8,F2`						IF	IS	S	EX1	EX2	EX3	**WB**

Cycles 13-16:

The MUL.D executes for two additional cycles, finishing in cycle 15. In cycle 16, this instruction broadcasts its result on the CDB, which allows the DIV.D to begin execution. The CDB broadcast also clears reservation station Mult1 and register result status table entry F0.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	N					
ALU2	N					
ALU3	N					
Mult1	**N**					
Mult2	Y	DIV.D	**[F0]**	[F6]		

Register result status table (FP only)

F0	F2	F4	F6	F8	F10
					Mult2

Pipeline diagram

Cycle	1	2	3	4	5	6	7	8	9	10	11	12	...	15	16
`L.D F6, 32(R2)`	IF	IS	EX	MEM	WB										
`L.D F2, 44(R3)`		IF	IS	EX	MEM	WB									
`MUL.D F0,F2,F4`			IF	IS	S	EX1	EX2	EX3	EX4	EX5	EX6	EX7	…	**EX10**	**WB**
`SUB.D F8,F6,F2`				IF	IS	EX1	EX2	EX3	WB						
`DIV.D F10,F0,F6`					IF	IS	S	S	S	S	S	S	…	**S**	**EX1**
`ADD.D F6,F8,F2`						IF	IS	S	EX1	EX2	EX3	WB			

Cycles 17-56:

The DIV.D has 40 execution cycles, running from cycle 16 to cycle 55. In cycle 56, this instruction finally broadcasts its result on the CDB, concluding the execution of this section of code.

Reservation stations:

Name	Busy?	Op	Vj	Vk	Qj	Qk
Load1	N					
Load2	N					
Load3	N					
ALU1	N					
ALU2	N					
ALU3	N					
Mult1	N					
Mult2	**N**					

Register result status table (FP only)

F0	F2	F4	F6	F8	**F10**

Pipeline diagram

Cycle	1	2	3	4	5	6	7	8	9	10	11	12	...	15	16	...	55	56
`L.D F6, 32(R2)`	IF	IS	EX	MEM	WB													
`L.D F2, 44(R3)`		IF	IS	EX	MEM	WB												
`MUL.D F0,F2,F4`			IF	IS	S	EX1	EX2	EX3	EX4	EX5	EX6	EX7	…	EX10	WB			
`SUB.D F8,F6,F2`				IF	IS	EX1	EX2	EX3	WB									
`DIV.D F10,F0,F6`					IF	IS	S	S	S	S	S	S	…	S	EX1	…	**EX40**	**WB**
`ADD.D F6,F8,F2`						IF	IS	S	EX1	EX2	EX3	WB						