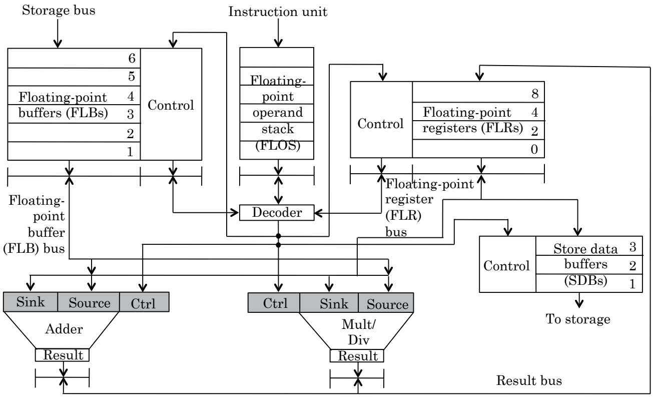
CS4100: Computer System Design

Tomasulo Algorithm

Madhu Mutyam
PACE Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Oct 12, 2015

Original Design of the IBM 360 FP Unit


Storage bus
Instruction unit
6
5
Floating-point buffers (FLBs)
4
3
2
1
Control
Floating-point operand stack (FLOS)
Control
8
Floating-point registers (FLRs)
4
2
0
Floating-point buffer (FLB) bus
Decoder
Floating-point register (FLR) bus
Control
Store data buffers (SDBs)
3
2
1
To storage
Sink
Source
Ctrl
Adder
Result
Ctrl
Sink
Source
Mult/ Div
Result
Result bus

IBM 360/91 FP Unit with Tomasulo Algorithm

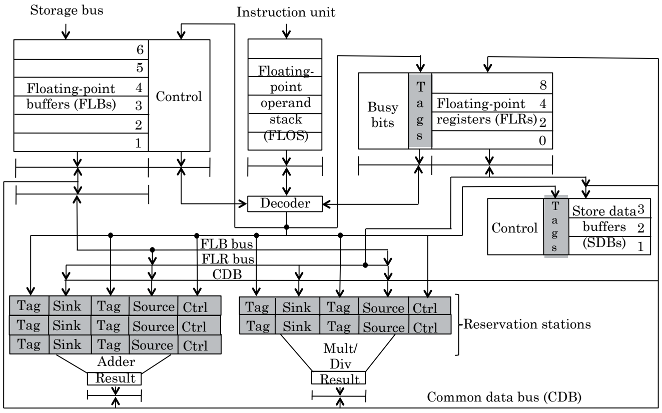


Illustration of Tomasulo Algorithm

w: ADD F4, F0, F8
x: MUL F2, F0, F4
y: ADD F4, F4, F8
z: MUL F8, F4, F2

- Assumptions:
  - FLOS to dispatch up to two instructions in every cycle
  - Instructions can begin execution in the same cycle that is dispatched to a RS
  - Functional units are not pipelined
  - ADD takes 2 cycles whereas MUL takes 3 cycles

Illustration of Tomasulo Algorithm (Contd)

	Reservation Station			
	Tag1	S1	Tag2	S2
1				
2				
3				
	Adder			

	Reservation Station			
	Tag1	S1	Tag2	S2
4				
5				
	Multiplier/Divide			

	FLR		
	Busy	Tag	Data
0			6.0
2			3.5
4			10.0
8			7.8

[CYCLE #1] Dispatched instructions: *w* and *x* (in order)

w: ADD F4, F0, F8

		Reservation Station			
		Tag1	S1	Tag2	S2
w	1	0	6.0	0	7.8
	2				
	3				
		Adder: *w*			

	Reservation Station			
	Tag1	S1	Tag2	S2
4				
5				
	Multiplier/Divide			

	FLR		
	Busy	Tag	Data
0			6.0
2			3.5
4	Yes	1	10.0
8			7.8

x: MUL F2, F0, F4

		Reservation Station			
		Tag1	S1	Tag2	S2
w	1	0	6.0	0	7.8
	2				
	3				
		Adder: *w*			

		Reservation Station			
		Tag1	S1	Tag2	S2
x	4	0	6.0	1	—
	5				
		Multiplier/Divide: *x*			

	FLR		
	Busy	Tag	Data
0			6.0
2	Yes	4	3.5
4	Yes	1	10.0
8			7.8

Illustration of Tomasulo Algorithm (Contd)

[CYCLE #2] Dispatched instructions: *y* and *z* (in order)

y: ADD F4, F4, F8

		Reservation Station			
		Tag1	S1	Tag2	S2
w	1	0	6.0	0	7.8
y	2	1	—	0	7.8
	3				
		Adder: *w*			

		Reservation Station			
		Tag1	S1	Tag2	S2
x	4	0	6.0	1	—
	5				
		Multiplier/Divide: *x*			

	FLR		
	Busy	Tag	Data
0			6.0
2	Yes	4	3.5
4	Yes	2	10.0
8			7.8

z: MUL F8, F4, F2

		Reservation Station			
		Tag1	S1	Tag2	S2
w	1	0	6.0	0	7.8
y	2	1	—	0	7.8
	3				
		Adder: *w*			

		Reservation Station			
		Tag1	S1	Tag2	S2
x	4	0	6.0	1	—
z	5	2	—	4	—
		Multiplier/Divide: *x*			

	FLR		
	Busy	Tag	Data
0			6.0
2	Yes	4	3.5
4	Yes	2	10.0
8	Yes	5	7.8

[CYCLE #3] Dispatched instructions: ________________

		Reservation Station			
		Tag1	S1	Tag2	S2
	1				
y	2	0	13.8	0	7.8
	3				
		Adder: *y*			

		Reservation Station			
		Tag1	S1	Tag2	S2
x	4	0	6.0	0	13.8
z	5	2	—	4	—
		Multiplier/Divide: *x*			

	FLR		
	Busy	Tag	Data
0			6.0
2	Yes	4	3.5
4	Yes	2	10.0
8	Yes	5	7.8

Illustration of Tomasulo Algorithm (Contd)

[CYCLE #4] Dispatched instructions: ______________

		Reservation Station			
		Tag1	S1	Tag2	S2
	1				
y	2	0	13.8	0	7.8
	3				
		Adder: *y*			

		Reservation Station			
		Tag1	S1	Tag2	S2
x	4	0	6.0	0	13.8
z	5	2	—	4	—
		Multiplier/Divide: *x*			

	FLR		
	Busy	Tag	Data
0			6.0
2	Yes	4	3.5
4	Yes	2	10.0
8	Yes	5	7.8

[CYCLE #5] Dispatched instructions: ______________

	Reservation Station			
	Tag1	S1	Tag2	S2
1				
2				
3				
	Adder			

		Reservation Station			
		Tag1	S1	Tag2	S2
x	4	0	6.0	0	13.8
z	5	0	21.6	4	—
		Multiplier/Divide: *x*			

	FLR		
	Busy	Tag	Data
0			6.0
2	Yes	4	3.5
4			21.6
8	Yes	5	7.8

[CYCLE #6] Dispatched instructions: ______________

	Reservation Station			
	Tag1	S1	Tag2	S2
1				
2				
3				
	Adder			

		Reservation Station			
		Tag1	S1	Tag2	S2
z	5	0	21.6	0	82.8
		Multiplier/Divide: *z*			

	FLR		
	Busy	Tag	Data
0			6.0
2			82.8
4			21.6
8	Yes	5	7.8

Tomasulo Algorithm

- Issue
  - Get the next instruction from the instruction queue in FIFO
  - If an RS entry is available, issue the instruction to RS either with operand value or register tag
  - Stall the instruction if there is no empty RS entry
- Execute
  - If all the operands are available, execute the instruction
  - Loads and stores are maintained in program order through effective address calculation
  - No instruction is allowed to initiate execution until all branches that precede the instruction in program order
- Write-back
  - Write the result on the CDB into registers and RS entries

Thank You