CMSC 411
Computer Systems Architecture
Lecture 10
Instruction Level Parallelism 4
(Dynamic Scheduling & Tomasulo Algorithm)

Outline

- ILP
- Compiler techniques to increase ILP
- Loop Unrolling
- Static Branch Prediction
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- (Start) Tomasulo Algorithm
- Conclusion

Advantages of Dynamic Scheduling

- Dynamic scheduling - hardware rearranges the instruction execution to reduce stalls while maintaining data flow and exception behavior
- Handles cases when dependences unknown at compile time
  - it allows the processor to tolerate unpredictable delays such as cache misses, by executing other code while waiting for the miss to resolve
- Allows code that compiled for one pipeline to run efficiently on a different pipeline
- Simplifies the compiler
- Leads to hardware speculation, a technique with significant performance advantages (discuss later)

HW Schemes: Instruction Parallelism

- Key idea: Allow instructions behind stall to proceed

```
DIVD  F0,F2,F4
ADDD  F10,F0,F8
SUBD  F12,F8,F14
```

- Enables out-of-order execution and allows out-of-order completion (e.g., `SUBD`)
  - In a dynamically scheduled pipeline, all instructions still pass through issue stage in order (in-order issue)
- Will distinguish when an instruction *begins execution* and when it *completes execution*; between 2 times, the instruction is *in execution*
- Note: Dynamic execution creates WAR and WAW hazards and makes handling exceptions harder

Dynamic Scheduling Step 1

- Simple pipeline had 1 stage to check both structural and data hazards: Instruction Decode (ID), also called Instruction Issue
- Split the ID pipe stage of simple 5-stage pipeline into 2 stages:
  - Issue
    - » Decode instructions, check for structural hazards
  - Read operands
    - » Wait until no data hazards, then read operands

A Dynamic Algorithm: Tomasulo's

- For IBM 360/91 (before caches!)
  - ⇒ Long memory latency
- Goal: High Performance without special compilers
- Small number of floating point registers (4 in 360) prevented interesting compiler scheduling of operations
  - This led Tomasulo to try to figure out how to get more effective registers — renaming in hardware!
- Why study a 1966 computer?
  - The descendants of this have flourished!
    - » Alpha 21264, Pentium 4, AMD Opteron, Power 5, …

Tomasulo Algorithm

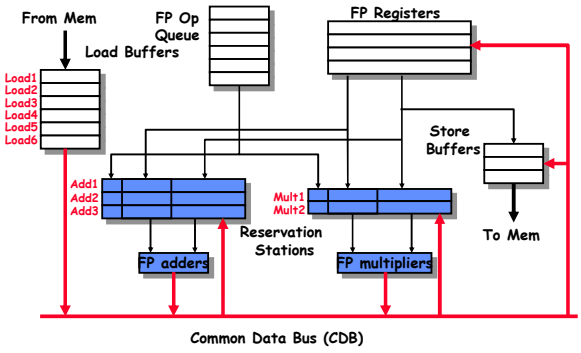
- Control & buffers distributed with Function Units (FU)
  - FU buffers called "reservation stations"; have pending operands
- Registers in instructions replaced by values or pointers to reservation stations (RS); called register renaming;
  - Renaming avoids WAR, WAW hazards
  - More reservation stations than registers, so can do optimizations compilers can't

Tomasulo Algorithm (cont.)

- Results to FU from RS, not through registers, over Common Data Bus that broadcasts results to all FUs
  - Avoids RAW hazards by executing an instruction only when its operands are available
- Load and Stores treated as FUs with RSs as well
- Integer instructions can go past branches (use branch prediction), also allow FP ops beyond basic block in FP queue

Tomasulo Organization

From H&P Figure 2.9



Reservation Station Components

- Op: Operation to perform in the unit (e.g., + or –)
- V_j, V_k: Value of Source operands
  - Store buffers have V field, result to be stored
- Q_j, Q_k: Reservation stations producing source registers (value to be written)
  - Note: Q_j,Q_k=0 => ready
  - Store buffers only have Q_i for RS producing result
- Busy: Indicates reservation station or FU is busy

In addition

- Register result status table—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions that will write that register.

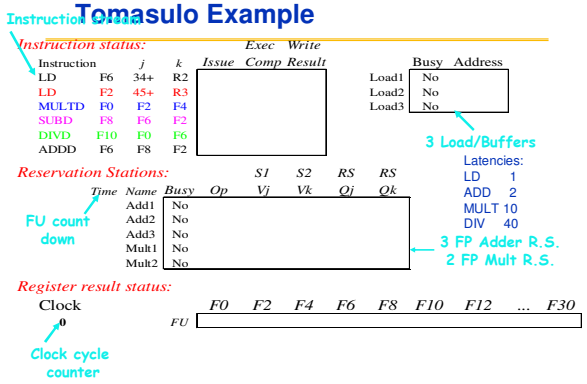
Three Stages of Tomasulo Algorithm

1. Issue—get instruction from FP Op Queue
   - If reservation station free (no structural hazard), control issues instr & sends operands (renames registers).
2. Execute—operate on operands (EX)
   - When both operands ready then execute; if not ready, watch Common Data Bus for result
3. Write result—finish execution (WB)
   - Write on Common Data Bus to all awaiting units; mark reservation station available

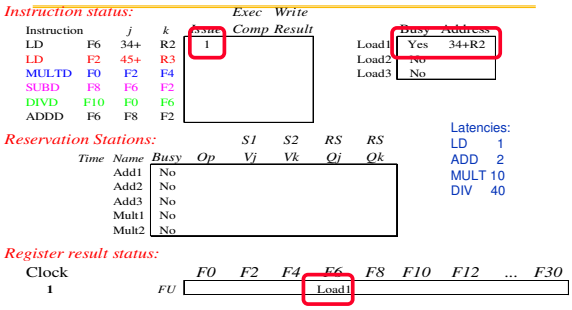
Common Data Bus

- Normal data bus: data + destination ("go to" bus)
- Common data bus: data + source ("come from" bus)
  - 64 bits of data + 4 bits of Functional Unit source address
  - Write if matches expected Functional Unit (produces result)
  - Does the broadcast

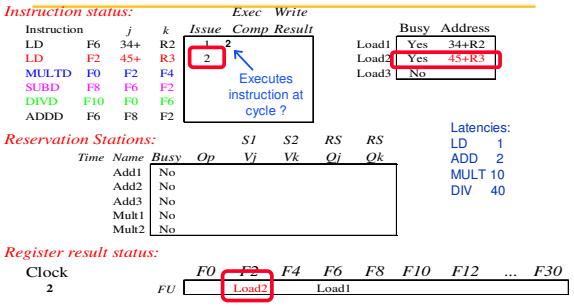
Tomasulo Example



Tomasulo Example Cycle 1

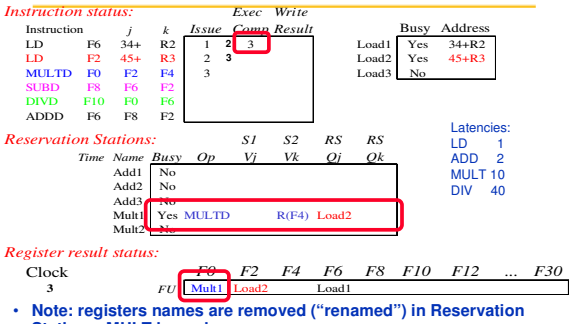


Tomasulo Example Cycle 2



Note: Can have multiple loads outstanding

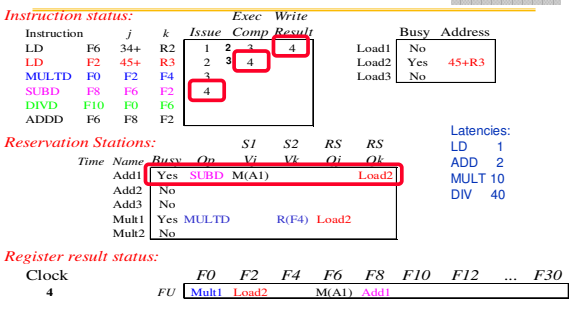
Tomasulo Example Cycle 3



- **Note: registers names are removed ("renamed") in Reservation Stations; MULT issued**
- **Load1 completing; what is waiting for Load1?**

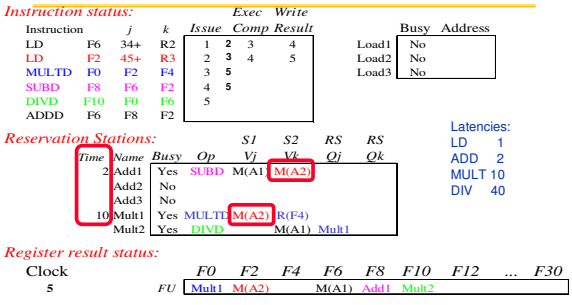
Tomasulo Example Cycle 4

M(A1) result of first load...



- **Load2 completing; what is waiting for Load2?**

Tomasulo Example Cycle 5



- **Timer starts down for Add1, Mult1**

Tomasulo Example Cycle 6

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5		
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies: LD 1, ADD 2, MULT 10, DIV 40

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
6	FU	Mult1	M(A2)			Add2	Add1	Mult2			

- **Issue ADDD here despite name dependency on F6?**

Tomasulo Example Cycle 7

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies: LD 1, ADD 2, MULT 10, DIV 40

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
7	FU	Mult1	M(A2)			Add2	Add1	Mult2			

- **Add1 (SUBD) completing; what is waiting for it?**

Tomasulo Example Cycle 8

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies: LD 1, ADD 2, MULT 10, DIV 40

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Mult1	M(A2)			Add2	(M-M)	Mult2			

Tomasulo Example Cycle 9

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies: LD 1, ADD 2, MULT 10, DIV 40

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
9	FU	Mult1	M(A2)			Add2	(M-M)	Mult2			

Tomasulo Example Cycle 10

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8	10	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies: LD 1, ADD 2, MULT 10, DIV 40

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
10	FU	Mult1	M(A2)			Add2	(M-M)	Mult2			

- **Add2 (ADDD) completing; what is waiting for it?**

Tomasulo Example Cycle 11

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies: LD 1, ADD 2, MULT 10, DIV 40

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

- **Write result of ADDD here?**
- **All quick instructions complete in this cycle!**

Tomasulo Example Cycle 12

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
12	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 13

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
13	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 14

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5		
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
14	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 15

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5	15	
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5			
ADDD F6	F8	F2	6	8	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

- **Mult1 (MULTD) completing; what is waiting for it?**

Tomasulo Example Cycle 16

Instruction status:

Instruction	j	k	Issue	Comp	Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	2	3	4	5
MULTD F0	F2	F4	3	5	15	16
SUBD F8	F6	F2	4	5	7	8
DIVD F10	F0	F6	5	16		
ADDD F6	F8	F2	6	8	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(A1)		

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

- **Just waiting for Mult2 (DIVD) to complete**

Faster than light computation (skip a few cycles)

Tomasulo Example Cycle 55

Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result
LD F6	34+	R2	1	2 3	4
LD F2	45+	R3	2	3 4	5
MULTD F0	F2	F4	3	5 15	16
SUBD F8	F6	F2	4	5 7	8
DIVD F10	F0	F6	5	16	
ADDD F6	F8	F2	6	8 10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(A1)		

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
55	FU	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 56

Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result
LD F6	34+	R2	1	2 3	4
LD F2	45+	R3	2	3 4	5
MULTD F0	F2	F4	3	5 15	16
SUBD F8	F6	F2	4	5 7	8
DIVD F10	F0	F6	5	16 56	
ADDD F6	F8	F2	6	8 10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	M*F4	M(A1)		

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

- **Mult2 (DIVD) is completing; what is waiting for it?**

Tomasulo Example Cycle 57

Instruction status:

Instruction	j	k	Issue	Exec Comp	Write Result
LD F6	34+	R2	1	2 3	4
LD F2	45+	R3	2	3 4	5
MULTD F0	F2	F4	3	5 15	16
SUBD F8	F6	F2	4	5 7	8
DIVD F10	F0	F6	5	16 56	57
ADDD F6	F8	F2	6	8 10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	Yes	DIVD	M*F4	M(A1)		

Latencies:
LD 1
ADD 2
MULT 10
DIV 40

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(A2)		(M-M+M	(M-M)	Result			

- **Once again: In-order issue, out-of-order execution and out-of-order completion.**

Why can Tomasulo overlap loop iterations?

- Register renaming
  - Multiple iterations use different physical destinations for registers (dynamic loop unrolling).
- Reservation stations
  - Permit instruction issue to advance past integer control flow operations
  - Also buffer old values of registers - totally avoiding WAR stalls
- Other perspective
  - Tomasulo building data flow dependency graph on the fly

2 Main Advantages w/ Tomasulo

- Distribution of the hazard detection logic
  - distributed reservation stations and the CDB
  - If multiple instructions waiting on single result, & each instruction already has other operand, then instructions can be released simultaneously by broadcast on CDB
  - If a centralized register file were used, the units would have to read their results from the registers when register buses are available
- Elimination of stalls for WAW and WAR hazards

Tomasulo Drawbacks

- Complexity
  - delays of 360/91, MIPS 10000, Alpha 21264
- Many associative stores (CDB) at high speed
- Performance limited by Common Data Bus
  - Each CDB must go to multiple functional units ⇒high capacitance, high wiring density
  - Number of functional units that can complete per cycle limited to one!
    - » Solution is to use multiple CDBs ⇒ more FU logic for parallel associative stores
- Non-precise interrupts!
  - We will address this later

And In Conclusion … #1

- Leverage Implicit Parallelism for Performance: Instruction Level Parallelism
- Loop unrolling by compiler to increase ILP
- Branch prediction to increase ILP
- Dynamic HW exploiting ILP
  - Works when can't know dependence at compile time
  - Can hide L1 cache misses
  - Code for one machine runs well on another

And In Conclusion … #2

- Reservations stations: renaming to larger set of registers + buffering source operands
  - Prevents registers becoming bottleneck
  - Avoids WAR, WAW hazards
  - Allows loop unrolling in HW
- Not limited to basic blocks (integer units get ahead, beyond branches)
- Helps ameliorate cost of cache misses as well

Tomasulo

- Lasting Contributions
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- 360/91 descendants include
  - Intel Pentium 4
  - IBM Power 5/6
  - AMD Athlon/Opteron
  - DEC Alpha 21264