Computer Architecture: Multiple Issue

Berk Sunar and Thomas Eisenbarth

ECE 505

WPI

Outline

- 5 stages of RISC
- Type of hazards
- Static and Dynamic Branch Prediction
- Pipelining with Exceptions
- Pipelining with Floating-Point Operations
- Loop Unrolling
- Correlating and Tournament Branch Prediction
- Dynamic Scheduling: Scoreboard
- Dynamic Scheduling: Tomasulo:
- Hardware Based Speculation

- Multiple Issue: VLIW 3.7
- Multiple Issue: Superscalar(speculative) 3.8
- Branch Target Buffer: Principles 3.9

Multiple Issue Processors

Last time:

- data and control stalls eliminated with dynamic scheduling and speculative execution
  - Performance close to 1 IPC
- To go beyond 1 IPC, more than 1 instruction must be issued (and completed) per cycle

Three major flavors of multiple issue:

1. Statically scheduled superscalar processors
2. VLIW (very long instruction word) processors
3. Dynamically scheduled superscalar processors

Characteristics

- Superscalar Processors issue varying number of instructions per clock
  - Either in-order execution (statically scheduled) or out-of-order execution (dynamically scheduled)
- VLIW Processors issue fixed number of instruction, formatted as one large instruction, with parallelism explicitly indicated by inst.
  - Inherently statically scheduled by compiler
  - High similarity to static superscalar!

Overview of Multiple-Issue Processors

Common Name	Issue structure	Hazard detection	Scheduling	Distinguishing characteristic	Examples
Superscalar (static)	Dynamic	Hardware	Static	In-order execution	Mostly Embedded : MIPS, ARM (e.g. Cortex-A8)
Superscalar (dynamic)	Dynamic	Hardware	Dynamic	Some out-of-order execution, but no speculation	None so far
Superscalar (speculative)	Dynamic	Hardware	Dynamic with speculation	Out-of-order execution with speculation	Intel Core iX, AMD Phenom, IBM POWER7
VLIW/LIW	Static	Primarily software	Static	All hazards determined by compiler (often implicitly)	Mostly signal processing, e.g. TI C6x
EPIC	Primarily static	Primarily software	Mostly static	All hazards determined and indicated explicitly by compiler	Itanium

Basic concept of Static VLIW

- VLIW uses multiple independent functional units (as last time), but
- VLIW packages instruction for each FU into one very large instruction.
- Overheads grow with amount of parallelism

Example:

- VLIW with 1x Integer FU, 2x Load/Store FU and 2x FP FU
- 16 to 24 bit opcode per FU: 80-120 bit inst. word

VLIW: Example

Program with loop (same as last week):

```
Loop:   L.D     F0, 0(R1)
        MUL.D   F4, F0, F2
        S.D     F4, 0(R1)
        DADDIU  R1, R1, -8
        BNE     R1, R2, Loop
```

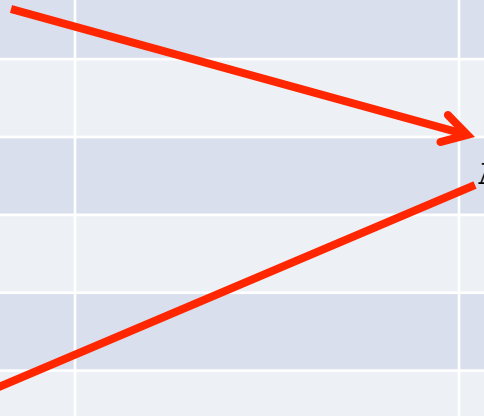
- VLIW requires heavy unrolling for being efficient

Q: How many unrolls to prevent stalls? Assuming these latencies:

Source Instruction	Destination instruction	Latency cycles
FP ALU op	FP ALU op	3
FP ALU op	Store double	2
Load double	FP ALU op	1
Load double	Store double	0

VLIW: How many unrolls?

Memory Unit 1	Memory Unit 2	FP unit 1	FP unit 2	Int unit
L.D F0,0(R1)				
		ADD.D F4,F0,F2		
S.D F4,0(R1)				
				DADDUI R1,R1,#-56
				BNE R1,R2,Loop



VLIW: Book proposes 7

Memory Unit 1	Memory Unit 2	FP unit 1	FP unit 2	Int unit
L.D F0,0(R1)	L.D F6,-8(R1)			
L.D F10,-16(R1)	L.D F14,-24(R1)			
L.D F18,-32(R1)	L.D F22,-40(R1)	ADD.D F4,F0,F2	ADD.D F8,F6,F2	
L.D F18,-48(R1)		ADD.D F12,F10,F2	ADD.D F16,F14,F2	
		ADD.D F20,F18,F2	ADD.D F24,F22,F2	
S.D F4,0(R1)	S.D F8,-8(R1)	ADD.D F28,F26,F2		
S.D F12,-16(R1)	S.D F16,-24(R1)			DADDUI R1,R1,#-56
S.D F24,+24(R1)	S.D F20,+16(R1)			
S.D F28,+8(R1)				BNE R1,R2,Loop

- 23 instructions in 9 cycles: 2.5 IPC

VLIW: Disadvantages

- Increase in code size
  - Non-fully filled instructions result in excess code
  - VLIW requires heavy unrolling (increasing code)
- Operation in lockstep: components like caches can cause entire processor to stall
- Code compatibility: if number/delay of functional units varies across processor families, code needs to be recompiled for each machine

Outline

- 5 stages of RISC
- Type of hazards
- Static and Dynamic Branch Prediction
- Pipelining with Exceptions
- Pipelining with Floating-Point Operations
- Loop Unrolling
- Correlating and Tournament Branch Prediction
- Dynamic Scheduling: Scoreboard
- Dynamic Scheduling: Tomasulo:
- Hardware Based Speculation
- Multiple Issue: VLIW 3.7

- Multiple Issue: Superscalar(speculative) 3.8
- Branch Target Buffer: Principles 3.9

Putting it all together:

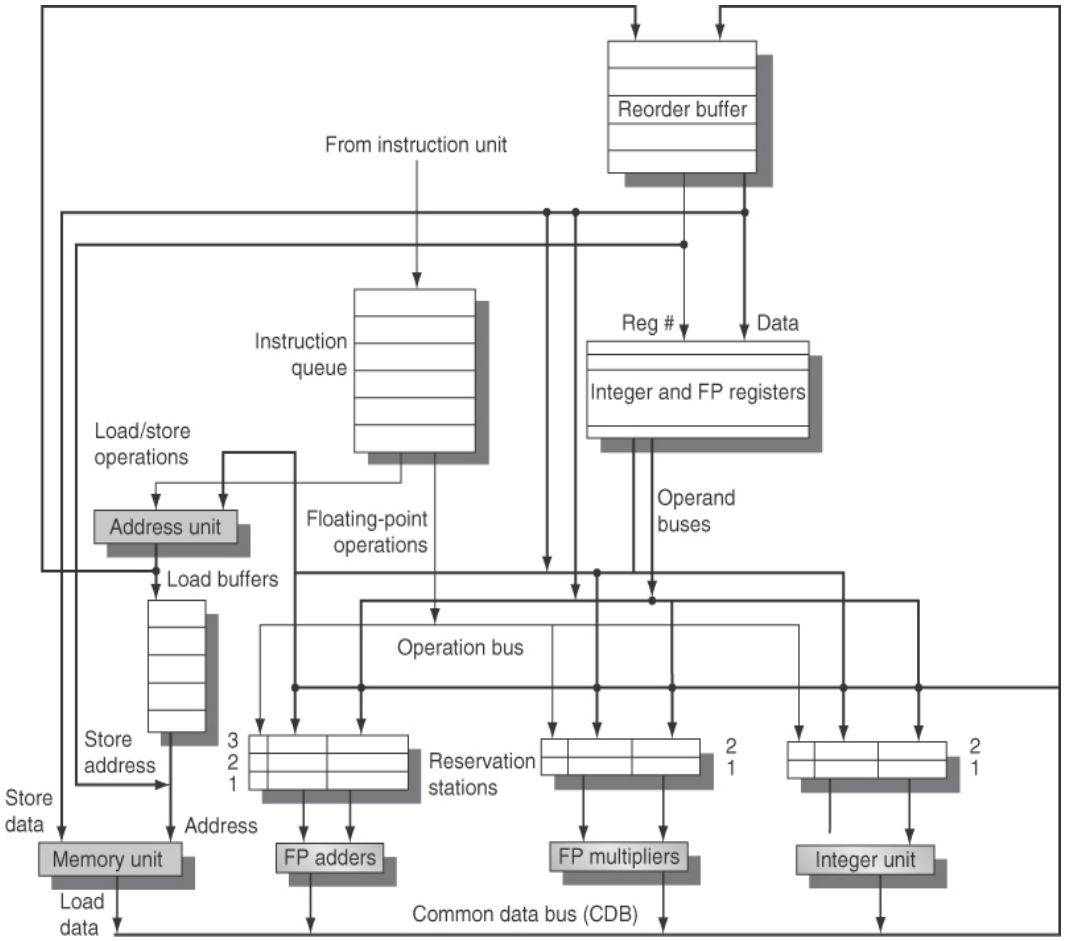
Goal: combine Multiple Issue with Dynamic Scheduling and Speculation

Example: Tomasulo+Speculation machine extended to issue 2 instructions per cycle

- Separate load/store unit, integer unit and FP unit, each can initiate 1 instruction per cycle
- Instructions issued *in-order* to prevent program semantics violations

Multiple Issue Tomasulo + Speculation

- Architecture essentially the same
- However, components must be redundant
  - E.g. CDB must broadcast up to 2 (N) results per cycle
  - Issue and completion logic become more complex



Multiple Issue with Speculation

Problem: multiple instructions scheduled in parallel may depend on each other

- Tables to be updated in parallel
  - Either by pipelining table updates (issuing logic) or
  - By widening issue logic (or both)

  →Issuing step becomes bottleneck, as complexity grows with $N \uparrow 2$ (for N IPC)
- Back-end of pipeline must be able to complete and commit multiple instructions per clock
  - Easier since dependences were resolved during issue

Instruction Issue: Steps

All steps must be performed in one clock cycle:

1. Assign reservation station and ROB for *every* instruction to be issued next. (Possible without knowing instruction for ROB and for RS by limiting the number of instructions per unit class). Instructions that cannot be issued (no FU available) are delayed in-order.
2. Analyze dependences of instructions in the issued *instruction bundle*.
3. If instruction in bundle depends on earlier one in bundle, use ROB number to update reservation table of dependent instruction. Otherwise just as before.

Multiple Issue+ Speculation: Example

Example Program:

```
LOOP:  LD     R2,0(R1)
       DADDIU R2,R2,#1
       SD     R2,0(R1)
       DADDIU R1,R1,#8
       BNE    R2,R3,LOOP
```

Q: Multiple issue performance with and without Speculation?

- Without speculation: 2nd LD must wait for BNE to execute
- With speculation: 2nd LD can execute as soon as R1 is updated
- →Data –dependent branches limit performance without speculation

Example: without speculation: cycle 1

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1			
1	DADDIU R2,R2,#1	1			
1	SD R2,0(R1)				
1	DADDIU R1,R1,#8				
1	BNE R2,R3,LOOP				
2	LD R2,0(R1)				
2	DADDIU R2,R2,#1				
2	SD R2,0(R1)				
2	DADDIU R1,R1,#8				
2	BNE R2,R3,LOOP				
3	LD R2,0(R1)				
3	DADDIU R2,R2,#1				
3	SD R2,0(R1)				
3	DADDIU R1,R1,#8				
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 2

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2		
1	DADDIU R2,R2,#1	1			
1	SD R2,0(R1)	2			
1	DADDIU R1,R1,#8	2			
1	BNE R2,R3,LOOP				
2	LD R2,0(R1)				
2	DADDIU R2,R2,#1				
2	SD R2,0(R1)				
2	DADDIU R1,R1,#8				
2	BNE R2,R3,LOOP				
3	LD R2,0(R1)				
3	DADDIU R2,R2,#1				
3	SD R2,0(R1)				
3	DADDIU R1,R1,#8				
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 3

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	
1	DADDIU R2,R2,#1	1			
1	SD R2,0(R1)	2	3		
1	DADDIU R1,R1,#8	2	3		
1	BNE R2,R3,LOOP	3			
2	LD R2,0(R1)				
2	DADDIU R2,R2,#1				
2	SD R2,0(R1)				
2	DADDIU R1,R1,#8				
2	BNE R2,R3,LOOP				
3	LD R2,0(R1)				
3	DADDIU R2,R2,#1				
3	SD R2,0(R1)				
3	DADDIU R1,R1,#8				
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 4

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1			
1	SD R2,0(R1)	2	3		
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3			
2	LD R2,0(R1)	4			
2	DADDIU R2,R2,#1	4			
2	SD R2,0(R1)				
2	DADDIU R1,R1,#8				
2	BNE R2,R3,LOOP				
3	LD R2,0(R1)				
3	DADDIU R2,R2,#1				
3	SD R2,0(R1)				
3	DADDIU R1,R1,#8				
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 5

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		
1	SD R2,0(R1)	2	3		
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3			
2	LD R2,0(R1)	4			
2	DADDIU R2,R2,#1	4			
2	SD R2,0(R1)	5			
2	DADDIU R1,R1,#8	5			
2	BNE R2,R3,LOOP				
3	LD R2,0(R1)				
3	DADDIU R2,R2,#1				
3	SD R2,0(R1)				
3	DADDIU R1,R1,#8				
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 6

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3		
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3			
2	LD R2,0(R1)	4			
2	DADDIU R2,R2,#1	4			
2	SD R2,0(R1)	5			
2	DADDIU R1,R1,#8	5			
2	BNE R2,R3,LOOP	6			
3	LD R2,0(R1)				
3	DADDIU R2,R2,#1				
3	SD R2,0(R1)				
3	DADDIU R1,R1,#8				
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 7

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4			
2	DADDIU R2,R2,#1	4			
2	SD R2,0(R1)	5			
2	DADDIU R1,R1,#8	5			
2	BNE R2,R3,LOOP	6			
3	LD R2,0(R1)	7			
3	DADDIU R2,R2,#1	7			
3	SD R2,0(R1)				
3	DADDIU R1,R1,#8				
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 8

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8		
2	DADDIU R2,R2,#1	4			
2	SD R2,0(R1)	5			
2	DADDIU R1,R1,#8	5	8		
2	BNE R2,R3,LOOP	6			
3	LD R2,0(R1)	7			
3	DADDIU R2,R2,#1	7			
3	SD R2,0(R1)	8			
3	DADDIU R1,R1,#8	8			
3	BNE R2,R3,LOOP				

Example: without speculation: cycle 9+10

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4			
2	SD R2,0(R1)	5	9		
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6			
3	LD R2,0(R1)	7			
3	DADDIU R2,R2,#1	7			
3	SD R2,0(R1)	8			
3	DADDIU R1,R1,#8	8			
3	BNE R2,R3,LOOP	9			

Example: without speculation: cycle 11+12

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9		
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6			
3	LD R2,0(R1)	7			
3	DADDIU R2,R2,#1	7			
3	SD R2,0(R1)	8			
3	DADDIU R1,R1,#8	8			
3	BNE R2,R3,LOOP	9			

Example: without speculation: cycle 13

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9	13	
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6	13		
3	LD R2,0(R1)	7			
3	DADDIU R2,R2,#1	7			
3	SD R2,0(R1)	8			
3	DADDIU R1,R1,#8	8			
3	BNE R2,R3,LOOP	9			

Example: without speculation: cycle 14

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9	13	
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6	13		
3	LD R2,0(R1)	7	14		
3	DADDIU R2,R2,#1	7			
3	SD R2,0(R1)	8			
3	DADDIU R1,R1,#8	8	14		
3	BNE R2,R3,LOOP	9			

Example: without speculation: cycle 15

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9	13	
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6	13		
3	LD R2,0(R1)	7	14	15	
3	DADDIU R2,R2,#1	7			
3	SD R2,0(R1)	8	15		
3	DADDIU R1,R1,#8	8	14		15
3	BNE R2,R3,LOOP	9			

Example: without speculation: cycle 16+17

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9	13	
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6	13		
3	LD R2,0(R1)	7	14	15	16
3	DADDIU R2,R2,#1	7	17		
3	SD R2,0(R1)	8	15		
3	DADDIU R1,R1,#8	8	14		15
3	BNE R2,R3,LOOP	9			

Example: without speculation: cycle 18

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9	13	
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6	13		
3	LD R2,0(R1)	7	14	15	16
3	DADDIU R2,R2,#1	7	17		18
3	SD R2,0(R1)	8	15		
3	DADDIU R1,R1,#8	8	14		15
3	BNE R2,R3,LOOP	9			

Example: without speculation: cycle 19

Iteration	Instruction	Issue	Execute	Mem access	Write CDB
1	LD R2,0(R1)	1	2	3	4
1	DADDIU R2,R2,#1	1	5		6
1	SD R2,0(R1)	2	3	7	
1	DADDIU R1,R1,#8	2	3		4
1	BNE R2,R3,LOOP	3	7		
2	LD R2,0(R1)	4	8	9	10
2	DADDIU R2,R2,#1	4	11		12
2	SD R2,0(R1)	5	9	13	
2	DADDIU R1,R1,#8	5	8		9
2	BNE R2,R3,LOOP	6	13		
3	LD R2,0(R1)	7	14	15	16
3	DADDIU R2,R2,#1	7	17		18
3	SD R2,0(R1)	8	15	19	
3	DADDIU R1,R1,#8	8	14		15
3	BNE R2,R3,LOOP	9	19		

Example: with speculation

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4				
2	DADDIU R2,R2,#1	4				
2	SD R2,0(R1)	5				
2	DADDIU R1,R1,#8	5				
2	BNE R2,R3,LOOP	6				
3	LD R2,0(R1)	7				
3	DADDIU R2,R2,#1	7				
3	SD R2,0(R1)	8				
3	DADDIU R1,R1,#8	8				
3	BNE R2,R3,LOOP	9				

Nothing changes up to here

Example: with speculation cycle 5

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	
1	SD R2,0(R1)	2	3			
1	DADDIU R1,R1,#8	2	3		4	
1	BNE R2,R3,LOOP	3	7			
2	LD R2,0(R1)	4	5	Speculative Execution starts in cycle 5		
2	DADDIU R2,R2,#1	4				
2	SD R2,0(R1)	5				
2	DADDIU R1,R1,#8	5				
2	BNE R2,R3,LOOP	6				
3	LD R2,0(R1)	7				
3	DADDIU R2,R2,#1	7				
3	SD R2,0(R1)	8				
3	DADDIU R1,R1,#8	8				
3	BNE R2,R3,LOOP					

Speculative Execution➔ ROB needed

Example: with speculation cycle 6

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	
1	SD R2,0(R1)	2	3			
1	DADDIU R1,R1,#8	2	3		4	
1	BNE R2,R3,LOOP	3	7			
2	LD R2,0(R1)	4	5	6		
2	DADDIU R2,R2,#1	4				
2	SD R2,0(R1)	5	6			
2	DADDIU R1,R1,#8	5	6			
2	BNE R2,R3,LOOP	6				
3	LD R2,0(R1)	7				
3	DADDIU R2,R2,#1	7				
3	SD R2,0(R1)	8				
3	DADDIU R1,R1,#8	8				
3	BNE R2,R3,LOOP					

Ok because of renaming

Example: with speculation cycle 7

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	
1	BNE R2,R3,LOOP	3	7			
2	LD R2,0(R1)	4	5	6	7	
2	DADDIU R2,R2,#1	4			Write only on "commit"	
2	SD R2,0(R1)	5	6			
2	DADDIU R1,R1,#8	5	6		7	
2	BNE R2,R3,LOOP	6				
3	LD R2,0(R1)	7				
3	DADDIU R2,R2,#1	7				
3	SD R2,0(R1)	8				
3	DADDIU R1,R1,#8	8				
3	BNE R2,R3,LOOP					

Example: with speculation cycle 8

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	
2	DADDIU R2,R2,#1	4	8			
2	SD R2,0(R1)	5	6			
2	DADDIU R1,R1,#8	5	6		7	
2	BNE R2,R3,LOOP	6				
3	LD R2,0(R1)	7	8			
3	DADDIU R2,R2,#1	7				
3	SD R2,0(R1)	8				
3	DADDIU R1,R1,#8	8				
3	BNE R2,R3,LOOP					

Example: with speculation cycle 9

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	9
2	DADDIU R2,R2,#1	4	8		9	
2	SD R2,0(R1)	5	6			
2	DADDIU R1,R1,#8	5	6		7	
2	BNE R2,R3,LOOP	6				
3	LD R2,0(R1)	7	8	9		
3	DADDIU R2,R2,#1	7				
3	SD R2,0(R1)	8	9			
3	DADDIU R1,R1,#8	8	9			
3	BNE R2,R3,LOOP	9				

Example: with speculation cycle 10

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	9
2	DADDIU R2,R2,#1	4	8		9	10
2	SD R2,0(R1)	5	6			10
2	DADDIU R1,R1,#8	5	6		7	
2	BNE R2,R3,LOOP	6	10			
3	LD R2,0(R1)	7	8	9	10	
3	DADDIU R2,R2,#1	7				
3	SD R2,0(R1)	8	9			
3	DADDIU R1,R1,#8	8	9		10	
3	BNE R2,R3,LOOP	9				

Example: with speculation cycle 11

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	9
2	DADDIU R2,R2,#1	4	8		9	10
2	SD R2,0(R1)	5	6			10
2	DADDIU R1,R1,#8	5	6		7	11
2	BNE R2,R3,LOOP	6	10			11
3	LD R2,0(R1)	7	8	9	10	
3	DADDIU R2,R2,#1	7	11			
3	SD R2,0(R1)	8	9			
3	DADDIU R1,R1,#8	8	9		10	
3	BNE R2,R3,LOOP	9				

Example: with speculation cycle 12

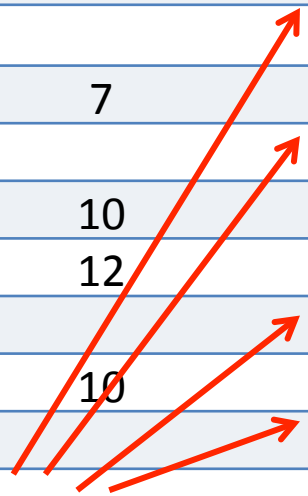
Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	9
2	DADDIU R2,R2,#1	4	8		9	10
2	SD R2,0(R1)	5	6			10
2	DADDIU R1,R1,#8	5	6		7	11
2	BNE R2,R3,LOOP	6	10			11
3	LD R2,0(R1)	7	8	9	10	12
3	DADDIU R2,R2,#1	7	11		12	
3	SD R2,0(R1)	8	9			
3	DADDIU R1,R1,#8	8	9		10	
3	BNE R2,R3,LOOP	9				

Example: with speculation cycle 13

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	9
2	DADDIU R2,R2,#1	4	8		9	10
2	SD R2,0(R1)	5	6			10
2	DADDIU R1,R1,#8	5	6		7	11
2	BNE R2,R3,LOOP	6	10			11
3	LD R2,0(R1)	7	8	9	10	12
3	DADDIU R2,R2,#1	7	11		12	13
3	SD R2,0(R1)	8	9			13
3	DADDIU R1,R1,#8	8	9		10	
3	BNE R2,R3,LOOP	9	13			

Example: with speculation cycle 14

Iteration	Instruction	Issue	Execute	Mem access	Write CDB	Commit
1	LD R2,0(R1)	1	2	3	4	5
1	DADDIU R2,R2,#1	1	5		6	7
1	SD R2,0(R1)	2	3			7
1	DADDIU R1,R1,#8	2	3		4	8
1	BNE R2,R3,LOOP	3	7			8
2	LD R2,0(R1)	4	5	6	7	9
2	DADDIU R2,R2,#1	4	8		9	10
2	SD R2,0(R1)	5	6			10
2	DADDIU R1,R1,#8	5	6		7	11
2	BNE R2,R3,LOOP	6	10			11
3	LD R2,0(R1)	7	8	9	10	12
3	DADDIU R2,R2,#1	7	11		12	13
3	SD R2,0(R1)	8	9			13
3	DADDIU R1,R1,#8	8	9		10	14
3	BNE R2,R3,LOOP	9	13			14



Commits almost 2 IPC after 1st iteration

Example: with speculation

Example with speculation: code completes after 14 cycles instead of 19 cycles.

→Advantage for data-dependent branches

- However, mispredictions can result in worse performance and potentially much higher power consumption.

Outline

- 5 stages of RISC
- Type of hazards
- Static and Dynamic Branch Prediction
- Pipelining with Exceptions
- Pipelining with Floating-Point Operations
- Loop Unrolling
- Correlating and Tournament Branch Prediction
- Dynamic Scheduling: Scoreboard
- Dynamic Scheduling: Tomasulo:
- Hardware Based Speculation
- Multiple Issue: VLIW 3.7
- Multiple Issue: Superscalar(speculative) 3.8

- Branch Target Buffer: Principles 3.9

Branch Target Buffer (BTB)

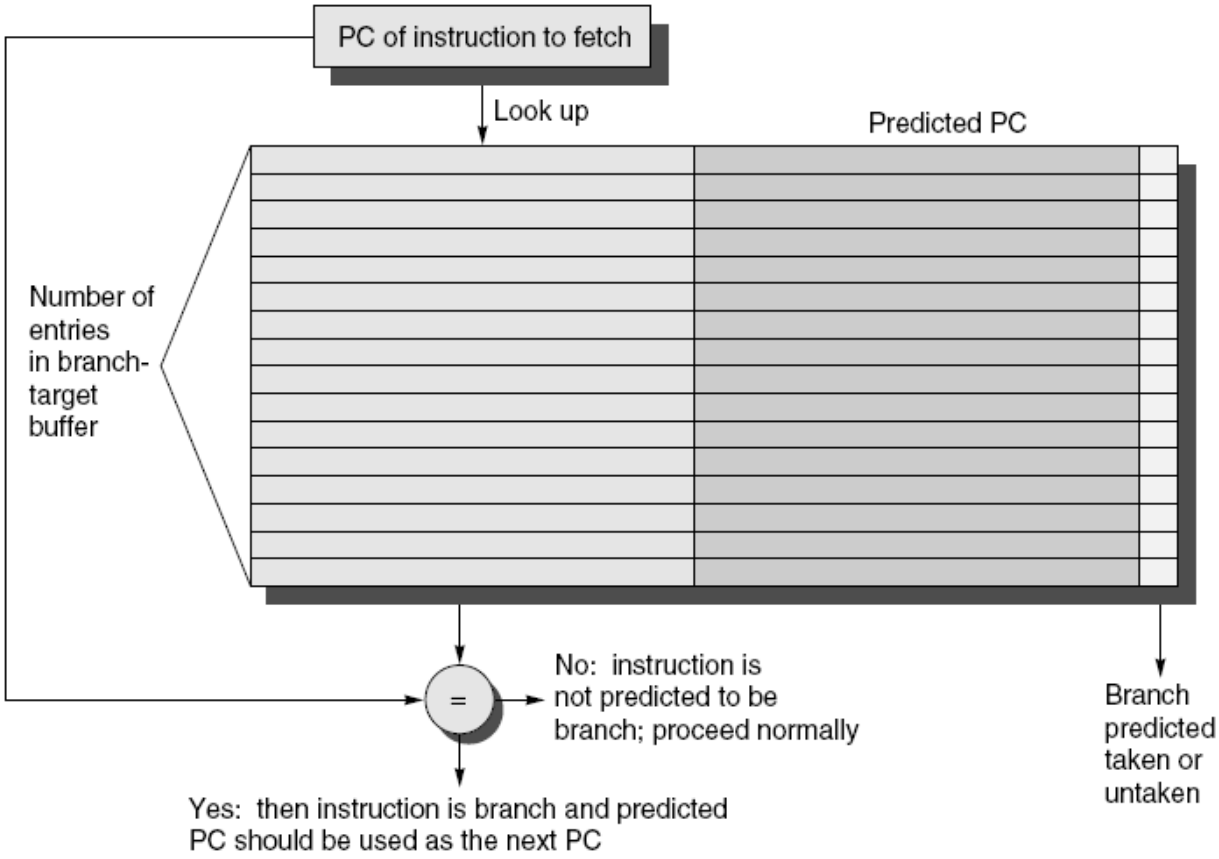
- Multiple Issue requires high IF bandwidth

Branch prediction:

- predicts branch outcome only
- Even with best possible prediction of branch outcome, still have to wait for branch target address to be determined for IF

Branch Target Buffer (BTB)

- BTB decides whether *undecoded* instruction is branch, and if so, predicts following PC



Branch Target Buffer (BTB)

- BTB decides whether *undecoded* instruction is branch, and if so, predicts following PC
- BTB only contains info for control instructions (Jumps and Branches)
- For all other instructions (and predict not-taken branches), next PC is PC+4
- How to update BTB?

BTB updating explained