

Lecture 11

Exploring ILP with Multi-Issue (3.6)

Instructor: Jun Yang

Review

- Branch Predictor
 - 1-bit
 - 2-bit
 - Correlating predictor (m,n)
- Branch Target Buffer
- Reduce branch penalty
- Two things can go wrong
 - BTB miss (misfetch)
 - Mispredicted a branch (mispredict)
- Suppose for *taken branches*, BTB hit rate of 85% and predict accuracy of 90%, misfetch penalty of 2 cycles and mispredict penalty of 5 cycles, what is the average branch penalty?

$$2 \cdot (15\%) + 5 \cdot (85\% \cdot 10\%) = 0.725$$

CPI = ...

- $CPI = 1.0 + BSPPI + FPSPPI + LdSPI$

Branch prediction attacks this

Our-of-order execution instruction scheduling attack this

- But what is the barrier to even higher performance?

Getting CPI < 1: Issuing Multiple Instructions/Cycle

Common name	Issue structure	Hazard detection	Scheduling	Distinguishing characteristic	Examples
Superscalar (static)	Dynamic	Hardware	Static	In-order execution	Sun UltraSPARC II/III
Superscalar (dynamic)	Dynamic	Hardware	Dynamic	Some out-of-order execution	IBM Power2
Superscalar (speculative)	Dynamic	Hardware	Dynamic with speculation	Out-of-order exe. with speculation	P4, MIPS R10000, Alpha 21264, HP PA 8500
VLIW/LIWI	Static	Software	Static	No hazards between issue packets	Trimedia, i860
EPIC (explicitly parallel instruction computer)	Mostly static	Mostly software	Mostly static	Explicit dependences marked by compiler	Itanium

Statically Scheduled Superscalar

- Compiler needs to do a good job in scheduling code (rearranging code sequence) – statically scheduled
- Fetch up to *n* instructions as an *issue packet* if issue width is *n*
- Check hazards during issue stage (including decode)
 - Issue checks are too complex to perform in one clock cycle
 - Issue stage is split and pipelined
 - Needs to check hazards within a packet, between two packets, among current and all the earlier instructions in execution.
- In effect an *n*-fold pipeline with complex issue logic and large set of bypass paths.

Type	Pipe	Stages	ID	EX	MEM	WB
Int. instruction	IF	ID	EX	MEM	WB	
FP instruction	IF	ID	EX	MEM	WB	
Int. instruction	IF	ID	EX	MEM	WB	
FP instruction	IF	ID	EX	MEM	WB	
Int. instruction	IF	ID	EX	MEM	WB	
FP instruction	IF	ID	EX	MEM	WB	

Dynamically Scheduled Superscalar

- Do not rely on compiler scheduling
- Dynamically determine instructions execution order
- Multi-issue + Tomasulo
- Branches are resolved before further instructions can execute
- Example (on page 221-222)


```

Loop:  LD    F0, 0(R1)      ;F0= array element
      ADD.D F4,F0,F2      ;add scalar in F2
      S.D   F4, 0(R1)     ;store result
      DADDIU R1, R1, #-8  ;decrement pointer
                               ;8 bytes (per DW)
                               ;branch R1!=R2
      BNE   R1, R2, Loop
            
```

 - 2 issue, 2 CDB
 - 1 integer unit (ALU and effective address, 1 cycle)
 - Separate FP unit for each type (3 cycles for ADDD)
 - Perfect dynamic branch-prediction, separate branch condition evaluation unit, single issue

Example (cycle 1)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1				First issue
1	ADD.D F4,F0,F2	1				
1	S.D F4, 0(R1)					
1	DADDIU R1, R1, #-8					
1	BNE R1, R2, Loop					
2	L.D F0, 0(R1)					
2	ADD.D F4,F0,F2					
2	S.D F4, 0(R1)					
2	DADDIU R1, R1, #-8					
2	BNE R1, R2, Loop					
3	L.D F0, 0(R1)					
3	ADD.D F4,F0,F2					
3	S.D F4, 0(R1)					
3	DADDIU R1, R1, #-8					
3	BNE R1, R2, Loop					

Nov. 4, 2003

Lec. 11

7

Example (cycle 2)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2			First issue
1	ADD.D F4,F0,F2	1				Wait for L.D
1	S.D F4, 0(R1)	2				
1	DADDIU R1, R1, #-8	2				
1	BNE R1, R2, Loop					
2	L.D F0, 0(R1)					
2	ADD.D F4,F0,F2					
2	S.D F4, 0(R1)					
2	DADDIU R1, R1, #-8					
2	BNE R1, R2, Loop					
3	L.D F0, 0(R1)					
3	ADD.D F4,F0,F2					
3	S.D F4, 0(R1)					
3	DADDIU R1, R1, #-8					
3	BNE R1, R2, Loop					

Nov. 4, 2003

Lec. 11

8

Example (cycle 3)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3		First issue
1	ADD.D F4,F0,F2	1				Wait for L.D
1	S.D F4, 0(R1)	2	3			
1	DADDIU R1, R1, #-8	2				Wait for ALU
1	BNE R1, R2, Loop	3				
2	L.D F0, 0(R1)					
2	ADD.D F4,F0,F2					
2	S.D F4, 0(R1)					
2	DADDIU R1, R1, #-8					
2	BNE R1, R2, Loop					
3	L.D F0, 0(R1)					
3	ADD.D F4,F0,F2					
3	S.D F4, 0(R1)					
3	DADDIU R1, R1, #-8					
3	BNE R1, R2, Loop					

Nov. 4, 2003

Lec. 11

9

Example (cycle 4)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1				Wait for L.D
1	S.D F4, 0(R1)	2	3			Wait for ADD.D
1	DADDIU R1, R1, #-8	2	4			
1	BNE R1, R2, Loop	3				Wait for DADDIU
2	L.D F0, 0(R1)		4			
2	ADD.D F4,F0,F2		4			
2	S.D F4, 0(R1)					
2	DADDIU R1, R1, #-8					
2	BNE R1, R2, Loop					
3	L.D F0, 0(R1)					
3	ADD.D F4,F0,F2					
3	S.D F4, 0(R1)					
3	DADDIU R1, R1, #-8					
3	BNE R1, R2, Loop					

Nov. 4, 2003

Lec. 11

10

Example (cycle 5)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5			
1	S.D F4, 0(R1)	2	3			Wait for ADD.D
1	DADDIU R1, R1, #-8	2	4		5	
1	BNE R1, R2, Loop	3				Wait for DADDIU
2	L.D F0, 0(R1)	4				Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for BNE complete
2	S.D F4, 0(R1)	5				
2	DADDIU R1, R1, #-8	5				
2	BNE R1, R2, Loop					
3	L.D F0, 0(R1)					
3	ADD.D F4,F0,F2					
3	S.D F4, 0(R1)					
3	DADDIU R1, R1, #-8					
3	BNE R1, R2, Loop					

Nov. 4, 2003

Lec. 11

11

Example (cycle 6)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5			
1	S.D F4, 0(R1)	2	3			Wait for ADD.D
1	DADDIU R1, R1, #-8	2	4		5	
1	BNE R1, R2, Loop	3	6			
2	L.D F0, 0(R1)	4				Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for BNE complete
2	S.D F4, 0(R1)	5				Wait for BNE complete
2	DADDIU R1, R1, #-8	5				Wait for BNE complete
2	BNE R1, R2, Loop	6				
3	L.D F0, 0(R1)					
3	ADD.D F4,F0,F2					
3	S.D F4, 0(R1)					
3	DADDIU R1, R1, #-8					
3	BNE R1, R2, Loop					

Nov. 4, 2003

Lec. 11

12

Example (cycle 13)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5		8	
1	S.D F4, 0(R1)	2	3	9		
1	DADDIU R1, R1, #-8	2	4		5	
1	BNE R1, R2, Loop	3	6			
2	L.D F0, 0(R1)	4	7	8	9	
2	ADD.D F4,F0,F2	4	10		13	
2	S.D F4, 0(R1)	5	8			Wait for ADD.D
2	DADDIU R1, R1, #-8	5	9		10	
2	BNE R1, R2, Loop	6	11			
3	L.D F0, 0(R1)	7	12	13		
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4, 0(R1)	8	13			
3	DADDIU R1, R1, #-8	8				Wait for ALU
3	BNE R1, R2, Loop	9				Wait for DADDIU

Nov. 4, 2003

Lec. 11

19

Example (cycle 14)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5		8	
1	S.D F4, 0(R1)	2	3	9		
1	DADDIU R1, R1, #-8	2	4		5	
1	BNE R1, R2, Loop	3	6			
2	L.D F0, 0(R1)	4	7	8	9	
2	ADD.D F4,F0,F2	4	10		13	
2	S.D F4, 0(R1)	5	8	14		
2	DADDIU R1, R1, #-8	5	9		10	
2	BNE R1, R2, Loop	6	11			
3	L.D F0, 0(R1)	7	12	13	14	
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4, 0(R1)	8	13			Wait for ADD.D
3	DADDIU R1, R1, #-8	8	14			
3	BNE R1, R2, Loop	9				Wait for DADDIU

Nov. 4, 2003

Lec. 11

20

Example (cycle 15)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5		8	
1	S.D F4, 0(R1)	2	3	9		
1	DADDIU R1, R1, #-8	2	4		5	
1	BNE R1, R2, Loop	3	6			
2	L.D F0, 0(R1)	4	7	8	9	
2	ADD.D F4,F0,F2	4	10		13	
2	S.D F4, 0(R1)	5	8	14		
2	DADDIU R1, R1, #-8	5	9		10	
2	BNE R1, R2, Loop	6	11			
3	L.D F0, 0(R1)	7	12	13	14	
3	ADD.D F4,F0,F2	7	15			
3	S.D F4, 0(R1)	8	13			Wait for ADD.D
3	DADDIU R1, R1, #-8	8	14		15	
3	BNE R1, R2, Loop	9				Wait for DADDIU

Nov. 4, 2003

Lec. 11

21

Example (cycle 16)

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5		8	
1	S.D F4, 0(R1)	2	3	9		
1	DADDIU R1, R1, #-8	2	4		5	
1	BNE R1, R2, Loop	3	6			
2	L.D F0, 0(R1)	4	7	8	9	
2	ADD.D F4,F0,F2	4	10		13	
2	S.D F4, 0(R1)	5	8	14		
2	DADDIU R1, R1, #-8	5	9		10	
2	BNE R1, R2, Loop	6	11			
3	L.D F0, 0(R1)	7	12	13	14	
3	ADD.D F4,F0,F2	7	15			
3	S.D F4, 0(R1)	8	13			Wait for ADD.D
3	DADDIU R1, R1, #-8	8	14		15	
3	BNE R1, R2, Loop	9	16			

Nov. 4, 2003

Lec. 11

22

Analysis

- Instruction throughput higher than single-issue pipeline
- Still not significant since there is only 1 FP operation per iteration, and thus, the integer pipeline becomes a bottleneck
- What will happen if we separate integer units for effective address calculation and for ALU operations?

Nov. 4, 2003

Lec. 11

23

Example

Iter.no.	Instructions	Issues at	Executes	Mem at	Write CDB	Comment
1	L.D F0, 0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5		8	Wait for L.D
1	S.D F4, 0(R1)	2	3	9		Wait for ADD.D
1	DADDIU R1, R1, #-8	2	3	4		Execute earlier
1	BNE R1, R2, Loop	3	5			Wait for DADDIU
2	L.D F0, 0(R1)	4	6	7	8	Wait for BNE complete
2	ADD.D F4,F0,F2	4	9		12	Wait for L.D
2	S.D F4, 0(R1)	5	7	13		Wait for ADD.D
2	DADDIU R1, R1, #-8	5	6		7	Execute earlier
2	BNE R1, R2, Loop	6	8			Wait for DADDIU
3	L.D F0, 0(R1)	7	9	10	11	Wait for BNE complete
3	ADD.D F4,F0,F2	7	12		15	Wait for L.D
3	S.D F4, 0(R1)	8	10	16		Wait for ADD.D
3	DADDIU R1, R1, #-8	8	9		10	Execute earlier
3	BNE R1, R2, Loop	9	11			Wait for DADDIU

Nov. 4, 2003

Lec. 11

24

Superscalar with Speculation

- Speculative execution – *execute* control dependent instructions even when we are not sure if they should be executed
- With branch prediction, we speculate on the outcome of the branches and execute the program as if our guesses were correct. Misprediction? Hardware undo
 - Recall in the dynamic superscalar example, instructions after the branch can be fetched and issued, but can not execute before the branch is resolved
 - Speculation allows them to execute with care.
- Multi-issue + branch prediction + Tomasulo
- Implemented in a number of processors:
PowerPC 603/604/G3/G4, Pentium II/III/4, Alpha 21264, AMD K5/K6/Athlon, MIPS R10k/R12k

Nov. 4, 2003

Lec. 11

25

Hardware Modifications

- Speculated instructions execute and generate results. Should they be written into register file? Should they be passed onto dependent instructions (in reservation stations)?
- Separate the bypassing paths from actual completion of an instruction. Do not allow speculated instructions to perform any updates that cannot be undone.
- When instructions are no longer speculative, allow them to update register or memory – *instruction commit*.
 - Out-of-order execution, in-order commit (provide precise exception handling)
- Then where are the instructions and their results between execution completion and instruction commit? Instructions may finish considerably before their commit.
- *Reorder buffer (ROB)* holds the results of instructions that have finished execution but have not committed.
 - ROB is a source of operands for instructions, much like the store buffer

Nov. 4, 2003

Lec. 11

26