

Lecture 10

Exploring ILP with Multi-Issue (3.6)

Instructor: Jun Yang

CPI = ...

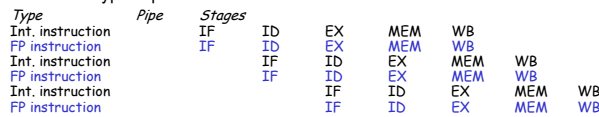
- $CPI = 1.0 + BSPI + FPSPI + LdSPI$
 - Branch prediction attacks this
 - Our-of-order execution instruction scheduling attack this
- But what is the barrier to even higher performance?

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	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.



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

- An integer example:

```

Loop:   LD    R2, 0(R1)    ;R2= array element
        DADDIU R2, R2, #1 ;increment R2
        SD    R2, 0(R1)  ;store result
        DADDIU R1, R1, #-4 ;decrement pointer
        BNE   R2, R3, Loop ;branch if not last element
    
```

- Separate integer FU for effective address, ALU op. and branch condition evaluation
- 2-issue, 2 CDB

Multi-issue + Tomasulo

Iteration	Instructions	Issue	Execution	Memory	Write CDB	Comment
1	LD R2,0(R1)	1	2	3	4	
1	DADDIU R2, R2, #1	1	5		6	Wait for LD
1	SD R2,0(R1)	2	3	7		Wait for DADDIU
1	DADDIU R1, R1, #4	2	3		4	
1	BNE R2, R3, LOOP	3	7			Wait for DADDIU
2	LD R2,0(R1)	4	8	9	10	Wait for BNE
2	DADDIU R2, R2, #1	4	11		12	Wait for LD
2	SD R2,0(R1)	5	9	13		Wait for DADDIU
2	DADDIU R1, R1, #4	5	8		9	Wait for BNE
2	BNE R2, R3, LOOP	6	13			Wait for DADDIU
3	LD R2,0(R1)	7	14	15	16	Wait for BNE
3	DADDIU R2, R2, #1	7	17		18	Wait for LD
3	SD R2,0(R1)	8	15	19		Wait for DADDIU
3	DADDIU R1, R1, #4	8	14		15	Wait for BNE
3	BNE R2, R3, LOOP	9	19			Wait for DADDIU

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
 - 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

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

Additional Functionalities of ROB

- Dynamically execute instructions while maintaining precise interrupt model.
 - In-order commit allows handling interrupts in-order at commit time
- Undo speculative actions when a branch is mispredicted
 - In reality, misprediction is expected to be handled as soon as possible. Flushing all the entries that appear after the branch, allowing those preceding instructions to continue.
 - Performance is very sensitive to branch-prediction mechanism
 - ❖ Prediction accuracy, misprediction detection and recovery
- Avoids hazards through memory (memory disambiguation)
 - WAW and WAR are removed since updating memory is done in order
 - RAW hazards are maintained by 2 restrictions:
 - ❖ A load's effective address is computed after all earlier stores
 - ❖ A load can not read from memory if there is an earlier store in ROB having the same effective address (some machine simply bypass the value from store to the load)

Nov. 24, 2004

Lec. 10

9

Compare Tomasulo with Speculation

Iteration	Instructions	Issue	Execution	Memory	Write CDB	Commit	Comment
1	LD R2,0(R1)	1	2	3	4	5	
1	DADDIU R2, R2, #1	1	5		6	7	Wait for LD
1	SD R2,0(R1)	2	3	XXXX		7	Wait for DADDIU
1	DADDIU R1, R1, #4	2	3		4	8	Commit in order
1	BNE R2, R3, LOOP	3	7			8	Wait for DADDIU
2	LD R2,0(R1)	4	5	6	7	9	No execution delay
2	DADDIU R2, R2, #1	4	8		9	10	Wait for LD
2	SD R2,0(R1)	5	6	XXXX		10	Wait for DADDIU
2	DADDIU R1, R1, #4	5	6		7	11	Commit in order
2	BNE R2, R3, LOOP	6	10			11	Wait for DADDIU
3	LD R2,0(R1)	7	8	9	10	12	Earliest possible
3	DADDIU R2, R2, #1	7	11		12	13	Wait for LD
3	SD R2,0(R1)	8	9	XXXX		13	Wait for DADDIU
3	DADDIU R1, R1, #4	8	9		10	14	Commit in order
3	BNE R2, R3, LOOP	9	13			14	Wait for DADDIU

Nov. 24, 2004

Lec. 10

10

Brief Comparison of Techniques

Techniques	Execution order	Degree of exploring ILP
Pipeline	In-order	None
Scoreboard	Out-of-order	Some
Tomasulo	Out-of-order	More
Multi-issue	Some in-order Some out-of-order	Many. Within basic block
Hardware speculation	Out-of-order	Beyond basic block

Nov. 24, 2004

Lec. 10

11