Multiple-issue processors

M. Sonza Reorda

Politecnico di Torino
Dipartimento di Automatica e Informatica

INTRODUCTION

The techniques for reducing the effects of data and control dependencies can be further exploited to obtain a CPI less than one: this requires issuing more than one instruction per clock cycle.

There are two kinds of processors able to do so:

- ***superscalar*** **processors, either statically or dynamically scheduling instructions**
- ***Very Long Instruction Word*** **(VLIW) processors.**

MULTIPLE-ISSUE STATIC SCHEDULING

Implementing a superscalar processor able to possibly issue several instructions according to the static order defined by the compiler is rather easy and inexpensive.

Statically scheduled superscalar MIPS version

Two instructions can be issued per clock cycle if:

- **one is a load, store, branch, or integer ALU operation**
- **the other is any FP operation (but load and store, which belong to the first category).**

Two instructions (64 bits) are fetched and decoded at every clock cycle. The two instructions are aligned on a 64-bit boundary and constitute an *issue packet*.

Instruction order within the packet

Old superscalar processors required a fixed structure for issue packets (e.g., that the integer instruction is always the first one).

Current processors normally do not have this limitation.

Ideal situation

FP instructions are all assumed to last for 3 clock cycles

Instruction type	Pipe stages							
Integer instruction	IF	ID	EX	MEM	WB			
FP instruction	IF	ID	EX	EX	EX	WB		
Integer instruction		IF	ID	EX	MEM	WB		
FP instruction		IF	ID	EX	EX	EX	WB	
Integer instruction			IF	ID	EX	MEM	WB	
FP instruction			IF	ID	EX	EX	EX	WB
Integer instruction				IF	ID	EX	MEM	WB
FP instruction				IF	ID	EX	EX	EX

Instruction fetching

At each clock cycle 2 instructions (64 bits) must be read from the instruction memory (i.e., from the cache).

If the two instructions belong to different cache blocks, several processors fetch one instruction, only.

FP Units

In order to obtain a real benefit, the floating point units should be either pipelined, or multiple and independent.

FP Register Contention

When the first instruction is a FP load, store, or move, there is a possible contention for a FP register port.

Possible solutions are:

- **forcing the first instruction to be executed by itself**
- **giving the FP register file a second port.**

Possible RAW Hazard

When the first instruction is a FP load, store, or move, and the second reads its result, a RAW hazard is also possible.

The second instruction must then be delayed by one clock cycle.

Data and Branch Delay

In the MIPS pipeline, load has a latency of one clock cycle, which means that in the superscalar pipeline the result of a load instruction can not be used on the same clock cycle or on the next one.

Therefore, in the MIPS superscalar version the load delay slot (as well as the branch delay slot) becomes equal to *three* instructions.

Exceptions

Some technique is required to guarantee precise exceptions.

MULTIPLE-ISSUE DYNAMIC SCHEDULING

It can be obtained by adopting a scheme similar to the Tomasulo one.

To make the implementation easier, instructions are never issued to the reservation stations out-of-order.

Once a branch is fetched, no other instructions are fetched until the branch is completed.

Example

```
Loop:   L.D       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)
        BNE       R1,R2,LOOP    ;branch R1!=R2
```

Let us study how this code is executed by a MIPS architecture implementing Tomasulo's algorithm.

Assumptions

At each clock cycle an integer and a FP instruction can be issued together.

The Write Result stage takes one extra clock cycle (unless for load and store instructions).

Execution latencies are

- **1 clock cycle for integer instructions**
- **2 clock cycles for load instructions**
- **3 clock cycles for FP add instructions.**

Branch prediction is always perfect.

The ALU is used for both integer and address computations.

Processor behavior

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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	4		5	Wait for ALU
1	BNE	R1,R2,Loop	3	6			Wait for DADDIU
2	L.D	F0,0(R1)	4	7	8	9	Wait for BNE complete
2	ADD.D	F4,F0,F2	4	10		13	Wait for L.D
2	S.D	F4,0(R1)	5	8	14		Wait for ADD.D
2	DADDIU	R1,R1,#-8	5	9		10	Wait for ALU
2	BNE	R1,R2,Loop	6	11			Wait for DADDIU
3	L.D	F0,0(R1)	7	12	13	14	Wait for BNE complete
3	ADD.D	F4,F0,F2	7	15		18	Wait for L.D
3	S.D	F4,0(R1)	8	13	19		Wait for ADD.D
3	DAADIU	R1,R1,#-8	8	14		15	Wait for ALU
3	BNE	R1,R2,Loop	9	16			Wait for DADDIU

Resource usage

Clock number	Integer ALU	FP ALU	Data cache	CDB
2	1/L.D			
3	1/S.D		1/L.D	
4	1/DADDIU			1/L.D
5		1/ADD.D		1/DADDIU
6				
7	2/L.D			
8	2/S.D		2/L.D	1/ADD.D
9	2/DADDIU		1/S.D	2/L.D
10		2/ADD.D		2/DADDIU
11				
12	3/L.D			
13	3/S.D		3/L.D	2/ADD.D
14	3/DADDIU		2/S.D	3/L.D
15		3/ADD.D		3/DADDIU
16				
17				
18				3/ADD.D
19			3/S.D	
20				

Performance evaluation

By looking at the execution of the first 3 iterations we observe that 15 instructions are completed in 16 clock cycles.

The sustained instruction completion rate is thus

15/16=0.94

Possible improvements

A further integer functional unit is added for effective address calculation.

A second CDB is added.

Processor behavior

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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	Executes 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	Executes 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	Executes earlier
3	BNE	R1,R2,Loop	9	11			Wait for DADDIU

Resource usage

Clock number	Integer ALU	Address adder	FP ALU	Data cache	CDB #1	CDB #2
2		1/L.D				
3	1/DADDIU	1/S.D		1/L.D		
4					1/L.D	1/DADDIU
5			1/ADD.D			
6	2/DADDIU	2/L.D				
7		2/S.D		2/L.D	2/DADDIU	
8					1/ADD.D	2/L.D
9	3/DADDIU	3/L.D	2/ADD.D	1/S.D		
10		3/S.D		3/L.D	3/DADDIU	
11					3/L.D	
12			3/ADD.D		2/ADD.D	
13				2/S.D		
14						
15						3/ADD.D
16				3/S.D		

Performance evaluation

The same number of instructions are completed in 5 clock cycles less (11 instead of 16), leading to a sustained instruction completion rate equal to

15/11=1.36