Dynamic ILP with O-O-O Execution

Soumyajit Dey, Associate Professor,
CSE, IIT Kharagpur

February 15, 2021

ROB

Reorder instrs to handle exceptions

instr	IS	DIS	WR
div f10, f0, f6	1	2	42
l.d f2, 45(r3)	2	3	13
mul f0, f2, f4	3	14	19
sub f8, f2,f6	4	14	15

assume: div takes 40 cycles
div by zero detected in cycle 40
by this time o/p of instr 2,3,4 all written

- after exception handling instr 1 will re-execute with wrong arguments
- say in exception handling f6 is given a small value $\neq 0$
- same issue, if some l.d has a page-fault

–imprecise handling

Branch misprediction due to OOO

div r1, r3, r4
beq r1, r2, label
add r3, r4, r5
sub f8, f2,f6
⋮
div

assume: beq mispredicts
by this time 2nd DIV executes and generates exception-> *phantom exception*
By the time branch result is known unnecessary exception handling may be going on

assume: beq mispredicts
by this time o/p of ADD written in r3
r3 changed wrongly

Solution ?

- execute, broadcast O-O-O
- deposit values in regs in-order

ROB

	REG	VAL	DONE
1			
2			
3			
4			
⋮	⋮	⋮	⋮

VAL : store instr o/p for target REG
DONE : bit to say validity : fwd to dependent instr
Two pointers:–>
COMMIT : from where to commit
ISSUE: from where to write in ROB

DIY: w/o ROB

LD R1, 0(R1)
BNE R1, R2, label
ADD R2, R1, R1
MUL R3, R3, R4
–LD dependent branch

ROB usage

IQ

⋮
r1=r2+r3

RS1	+		
RS2			
RS3			

ADD

		REG	VAL	DONE
commit	1 2			
	3			
	4			
⋮	⋮	⋮	⋮	
issue	n	r1		0

r1	
r2	
r3	

Register file

- Put instr from IQ to RS and update ROB
- update issue pointer in ROB
- RAT for r1 point to ROB entry of instr

r1	ROB:n
r2	
r3	

RAT

ROB usage

RS1	+		
RS2			
RS3			

IQ

⋮
r1=r2+r3

ADD
o/p: <ROB:n> result

Can empty RS after DISPATCH; earlier RS was kept occupied until broadcast
RS served as name tag
This is now ROB entry
All RS tags are ROB entries

r1	
r2	
r3	

Register file

		REG	VAL	DONE
commit	1			
	2			
	3			
	4			
⋮	⋮	⋮	⋮	
issue	n	r1		0

r1	ROB:n
r2	
r3	

RAT

ROB usage

IQ

⋮
r1=r2+r3

RS1			
RS2			
RS3			

ADD

BROADCAST:
All RS tags are ROB entries
RS tags updated with result
W/O ROB : result went to
REG, updated RAT
With ROB : result goes to
ROB

		REG	VAL	DONE
commit	1			
	2			
	3			
	4			
⋮	⋮	⋮	⋮	
issue	n	r1	result	1

r1	
r2	
r3	

Register file

r1	ROB:n
r2	
r3	

RAT

ROB usage

IQ

⋮
r1=r2+r3

RS1			
RS2			
RS3			

ADD

COMMIT:
write result from ROB to REG
update RAT entry
update commit point

r1	result
r2	
r3	

Register file

		REG	VAL	DONE
	1 2			
	3			
	4			
⋮	⋮	⋮	⋮	
issue, commit	n	r1	result	1

r1	r1
r2	
r3	

RAT

Branch misprediction handling

l.d r1 $\phi(r1)$
bne r1 r2 label
add r2 r1 r1
mul r3 r3 r4
div r2 r3 r7
mispredicted instrs
marked in red

r1	
r2	
r3	

Register file

r1	
r2	
r3	

RAT

		REG	VAL	DONE
i/c–>	1			
	2			
	3			
	4			
⋮	⋮	⋮	⋮	
	n			

ROB

Branch misprediction handling

l.d r1 $\phi(r1)$
bne r1 r2 label
add r2 r1 r1
mul r3 r3 r4
div r2 r3 r7
mispredicted instrs
marked in red

r1	
r2	
r3	

Register file

r1	ROB1
r2	ROB3
r3	ROB4

RAT

		REG	VAL	DONE
c–>	1	r1		
	2	ϕ		
	3	r2		
i–>	4	r3		
⋮	⋮	⋮	⋮	
	n			

ROB

Branch misprediction handling

The updates go to ROB
instead of REG
Hence no damage

l.d r1 $\phi(r1)$
bne r1 r2 label
add r2 r1 r1
mul r3 r3 r4
div r2 r3 r7

r1	
r2	
r3	

Register file

r1	ROB1
r2	ROB5
r3	ROB4

RAT

		REG	VAL	DONE
c–>	1	r1		
	2	ϕ		
	3	r2		
	4	r3	15	1
i–>	5	r2	2	1
⋮	⋮	⋮	⋮	
	n			

ROB

Branch misprediction handling

l.d r1 $\phi(r1)$
bne r1 r2 label
add r2 r1 r1
mul r3 r3 r4
div r2 r3 r7
mispredicted instrs marked in red
Assume : l.d has a cache miss
bne, add delayed
mul : o/p = 15
div o/p = 2

r1	700
r2	
r3	

Register file

r1	~~ROB1~~
r2	ROB5
r3	ROB4

RAT

l.d returns
commit, write REG
bne still pending
add returns

		REG	VAL	DONE
c–>	1	r1	700	
	2	ϕ		
	3	r2	3	1
	4	r3	15	1
i–>	5	r2	2	1
⋮	⋮	⋮	⋮	
	n			

ROB

Branch misprediction handling

bne returns and come to commit
roll back issue
ROB invalidated
undo RAT entries
RS, ALU -> empty

l.d r1 $\phi(r1)$
bne r1 r2 label
add r2 r1 r1
mul r3 r3 r4
div r2 r3 r7
mispredicted instrs marked in red

r1	700
r2	
r3	

Register file

r1	~~ROB1~~
r2	~~ROB5~~
r3	~~ROB4~~

RAT

		REG	VAL	DONE
	1	r1	700	
i, c–>	2	ϕ		
	3	r2	3	1
	4	r3	15	1
	5	r2	2	1
⋮	⋮	⋮	⋮	
	n			

Exception handling

div r1 r2 r3
add ...
1. If add is not in RAW, it will finish and write in ROB
2. div : when encounters div-by-0 exception will need to invoke exception handler
3. Exception marked in ROB for r1
4. When DIV reaches commit, future ROB updates are flushed & exception handler invoked
5. Similar issue with an ld suffering from a page-fault
6. Not committing to REG ensures consistent architectural state

Phantom exception

beq r1 r2 label
div r0 r0 r5
1. branch not taken, div executed
2. div : when encounters div-by-0 exception, branch still not resolved
3. Exception marked in ROB for r0
4. Commit reaches branch and waits till branch is resolved; If resolution is to take branch ->
5. The DIV and other executed ROB entries are flushed
6. Since *exception handling is delayed till triggering instr commits* –> consistent state

ROB update with commit

1	r1	r2+r3
2	r3	r5+r6
3	r1	ROB1$*r7$
4	r1	r4+r8
5	r2	r9 +ROB2

ROB

r1	ROB4
r2	ROB5
r3	ROB2

RAT

r1	
r2	
r3	
r4	

REG

Original Tomasulo

1. No commit phase, With every broadcast, RAT was checked
2. Since r2+r3 is not latest update to r1 as par RAT, so it will not update REG

ROB Scheme

1. In-order commit
2. Write r2+r3 to REG, But do not change RAT as it is also correct that the write is not latest update to r1

ROB update with commit

1		
2	r3	r5+r6
3	r1	ROB1$*r7$
4	r1	r4+r8
5	r2	r9 +ROB2

ROB

r1	ROB4
r2	ROB5
r3	ROB2

RAT

r1	r2+r3
r2	
r3	r5+r6
r4	

REG

1. delete entry from ROB1 with REG update
2. move to next entry ROB2

1. follow ROB2: write r5+r6 to r3
2. check RAT : ROB2 is latest rename of r3
3. empty ROB2, RAT: ROB2–>r3

ROB update with commit

1		
2		
3	~~r1~~	~~ROB1 $*$ r7~~
4	r1	r4+r8
5	r2	r9 +ROB2

ROB

r1	ROB4
r2	ROB5
r3	r3

RAT

r1	ROB1$*$ r7
r2	
r3	r5+r6
r4	

REG

1. Follow ROB3: write ROB1$*$ r7 to r1
1. Delete entry from ROB3 with REG update
2. Move to next entry ROB4

1. Check RAT : ROB3 is NOT latest rename of r1
2. Keep RAT intact

1. At this point if an exception is detected: Flush ROB
2. Make all RAT entries point to REG: consistent arch state

ROB update with commit

1		
2		
3		
4	~~r1~~	~~r4+r8~~
5	r2	r9 +ROB2

ROB

r1	r1
r2	ROB5
r3	r3

RAT

r1	r4+r8
r2	
r3	r5+r6
r4	

REG

1. W/O exception : follow usual course

1. Check RAT : ROB4 is latest rename of r1
2. let RAT entry for r1 point to r1

Unified RS

- Making RS specific to exec units is not optimal
- have a common RS file for all exec units
- Dispatch logic is complex, but we have same on #RS as they are expensive

Super scalar Processor

Requirements

- fetch $>$ 1 instr /cycle
- decode $>$ 1 instr /cycle
- issue $>$ 1 instr /cycle
- dispatch $>$ 1 instr /cycle
- broadcast $>$ 1 instr /cycle
- commit $>$ 1 instr /cycle

The chain works as fast as the weakest link !

Thank You[1]

[1]Most of the material are taken from the famous book on Comp Arch by Hen/Pat, Comp Arch course by Milos Prvulovic for teaching purposes