CPEN 411: Computer Architecture

Slide Set #9: Reorder Buffer, Exceptions, Multiple Instruction Issue

Original Slides: Tor Aamodt

aamodt@ece.ubc.ca

Background: PowerPC 604, one of the first microprocessors with a reorder buffer

Introduction to Slide Set 9

- Out-of-order execution with Tomasulo's algorithm allows an instruction to execute when its source operands are available. Branch prediction enables us to fetch and decode instructions after a branch before that branch is "resolved".

- To increase performance (reduce CPI) we would like to start executing the instructions fetched and decoded following the predicted branch *before* that branch is resolved. However, we have a problem: out-of-order execution (e.g., using Tomasulo's algorithm) allows an instruction to update the register file before the branch is "resolved".

- An elegant solution is a hardware structure known as the "re-order buffer". One reason for calling it an "elegant solution" is it also enables "virtual memory" (Slide Set 12) by supporting an execution model known as "precise exceptions".

Learning Objectives

After we finish this set of slides you should be able to:

- Explain the motivation for using a reorder buffer and how a reorder buffer operates.
- Define multiple instruction issue and explain the motivation for using it.
- List two approaches to achieving multiple instruction issue
- Evaluate the timing of a processor using multiple issue and Tomasulo's algorithm.
- Evaluate the timing of a processor using multiple issue, Tomasulo's algorithm and a reorder buffer.
- Define what an exception is and list several categories of exceptions.
- Explain what is meant by a precise exception.
- Explain how the reorder buffer supports precise exceptions.

Speculative Execution

```
        DIVD    R3,R1,R2  ; F:1, D:2, I:3, X:4, W:104
        BEQZ    R3,Label  ; F:2, D:3, I:4, X:105 ("not taken")
        …
Label: DMUL    R4,R4,R2  ; F:3, D:4, I:5, X:6, W:?
```

branch predicted "taken" on cycle 2

Speculative Execution

```
        DIVD    R3,R1,R2  ; F:1, D:2, I:3, X:4, W:104
        BEQZ    R3,Label  ; F:2, D:3, I:4, X:105 ("not taken")
        …                 <-- branch predicted "taken" on cycle 2
Label:  DMUL    R4,R4,R2  ; F:3, D:4, I:5, X:6, W:?
```

Consider the short program above. Assume this code runs on a pipeline using Tomasulo's algorithm as described in Slide Set 7. Also, a branch predictor is used and predicts the branch is taken. This enables the DMUL to start execution before the correct outcome of the branch is known.

"S:N" means in stage S on cycle N. The pipeline stages are: F=fetch, D=decode, I=issue, X= execute begin, W=write result. BEQZ is resolved in "X". DMUL takes 10 cycles to execute.

Speculative Execution

```
        DIVD    R3,R1,R2  ; F:1, D:2, I:3, X:4, W:104
        BEQZ    R3,Label  ; F:2, D:3, I:4, X:105 ("not taken")
        …                 <-- branch predicted "taken" on cycle 2
Label:  DMUL    R4,R4,R2  ; F:3, D:4, I:5, X:6, W:?
```

Consider the short program above. Assume this code runs on a pipeline using Tomasulo's algorithm as described in Slide Set 7. Also, a branch predictor is used and predicts the branch is taken. This enables the DMUL to start execution before the correct outcome of the branch is known.

"S:N" means in stage S on cycle N. The pipeline stages are: F=fetch, D=decode, I=issue, X= execute begin, W=write result. BEQZ is resolved in "X". DMUL takes 10 cycles to execute.

Can DMUL write to common data bus on cycle 16 or does it need to wait until the correct branch outcome is known?

A: Write on clock cycle 16
B: Wait until correct branch outcome known on clock cycle 105
C: Not sure

Speculative Execution

```
        DIVD    R3,R1,R2  ; F:1, D:2, I:3, X:4, W:104
        BEQZ    R3,Label  ; F:2, D:3, I:4, X:105 ("not taken")
        …
Label:  DMUL    R4,R4,R2  ; F:3, D:4, I:5, X:6, W:?
```

branch predicted "taken" on cycle 2

Consider the short program above. Assume this code runs on a pipeline using Tomasulo's algorithm as described in Slide Set 7. Also, a branch predictor is used and predicts the branch is taken. This enables the DMUL to start execution before the correct outcome of the branch is known.

"S:N" means in stage S on cycle N. The pipeline stages are: F=fetch, D=decode, I=issue, X= execute begin, W=write result. BEQZ is resolved in "X". DMUL takes 10 cycles to execute.

Can DMUL write to common data bus on cycle 16 or does it need to wait until the correct branch outcome is known?

A: Write on clock cycle 16
B: Wait until correct branch outcome known on clock cycle 105 ✓
C: Not sure

Speculative Execution

```
        DIVD    R3,R1,R2  ; F:1, D:2, I:3, X:4, W:104
        BEQZ    R3,Label  ; F:2, D:3, I:4, X:105 ("not taken")
        …
Label: DMUL    R4,R4,R2  ; F:3, D:4, I:5, X:6, W:?
```

branch predicted "taken" on cycle 2

Speculative Execution

```
        DIVD    R3,R1,R2  ; F:1, D:2, I:3, X:4, W:104
        BEQZ    R3,Label  ; F:2, D:3, I:4, X:105 ("not taken")
        …                 branch predicted "taken" on cycle 2
Label:  DMUL    R4,R4,R2  ; F:3, D:4, I:5, X:6, W:?
```

Goal: Support execution of instructions fetched following a branch prediction before we know if the prediction is correct. Above: Want to execute and broadcast result of DMUL "speculatively" long before branch resolved on cycle 105.

Problem: Such instructions should not update register file because branch might have been predicted incorrectly. Above: cannot let DMUL write to register file until control hazard resolved.

The Reorder Buffer (ROB)

- An elegant solution to this problem is a hardware structure known as a "Reorder Buffer" (ROB).

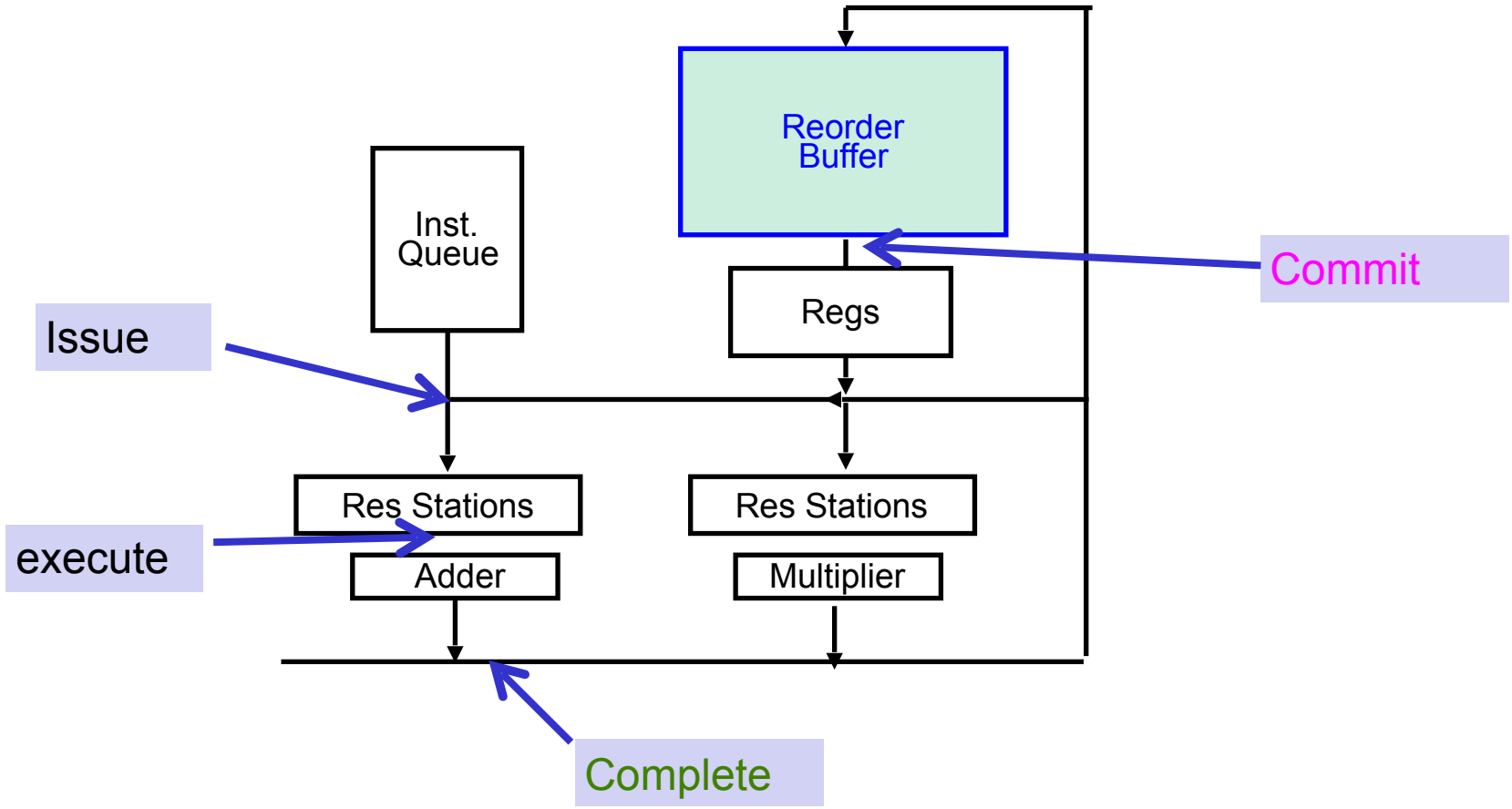
- The reorder buffer is used to divide the "writeback" step into two separate steps: "instruction completion" and "instruction commit".

- An instruction enters "instruction completion" when it finishes execution.
  - Instruction broadcasts result on the CDB so dependent instructions can begin execution.
  - However, register file not updated yet.
  - Instead, the result is "buffered" in the reorder buffer to be written into the register file later.

- An instruction enters "instruction commit" when it is the oldest instruction that has completed (and is free from "exceptions").
  - Writes results to register file

The Reorder Buffer (ROB)

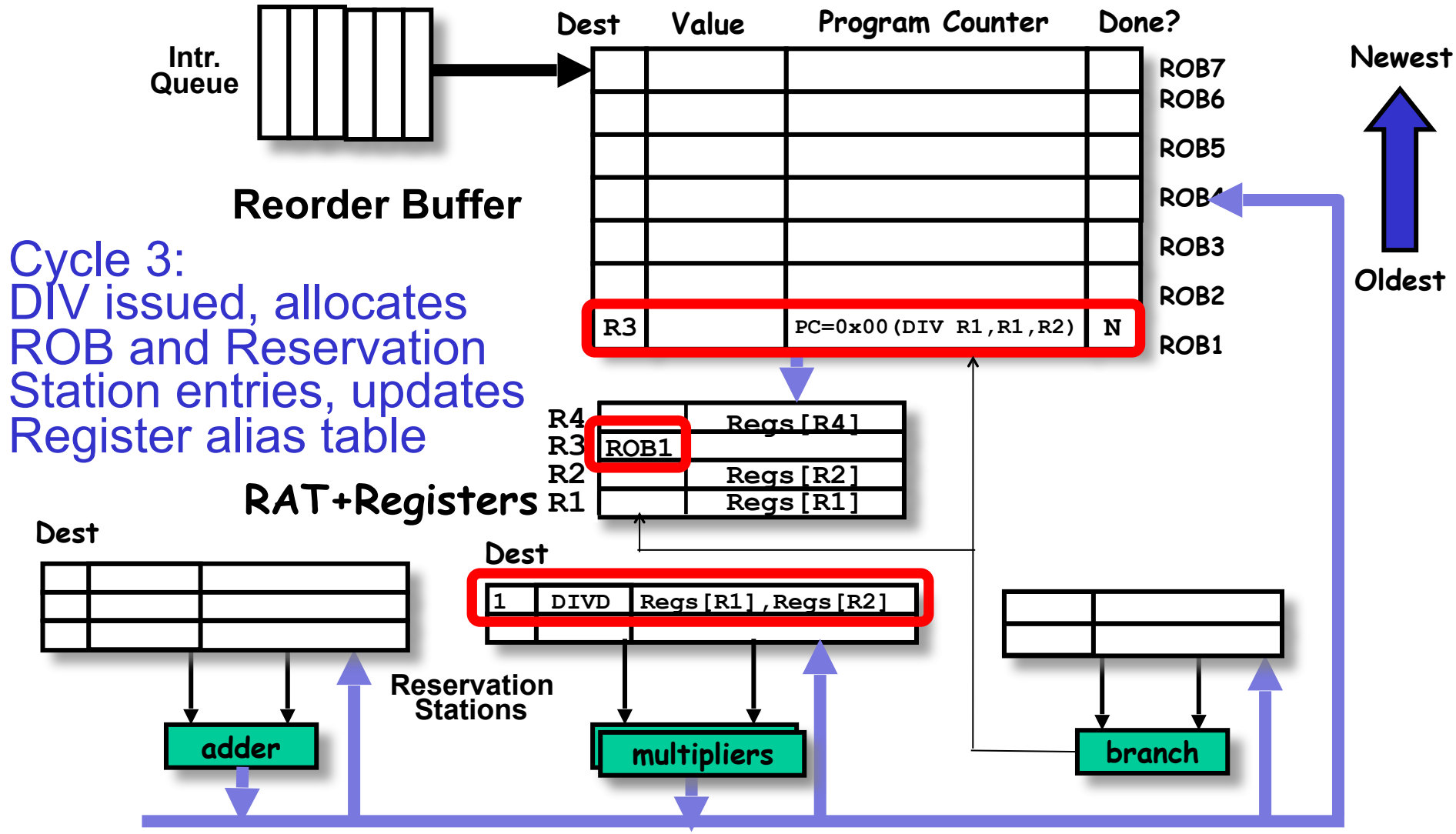
Pipeline stages: fetch, decode, issue, execute, complete, commit



The Reorder Buffer (ROB)

- The Reorder Buffer (ROB) maintains a “window” of dynamic instructions currently in the pipeline.
- The instructions are stored in “first-in first-out” order. That is, the order originally specified by the programmer). Entry allocated in ROB during issue at same time as a reservation station is allocated.
- Each ROB entry contains the instruction’s PC, destination register number, destination register value, and an execution status flag.
- To make it easier to update the ROB after an instruction executes and to enable easy lookup of results buffered in the ROB but not yet written to the register file we use the ROB entry of the instruction as the tag for renaming instead of reservation station ID.

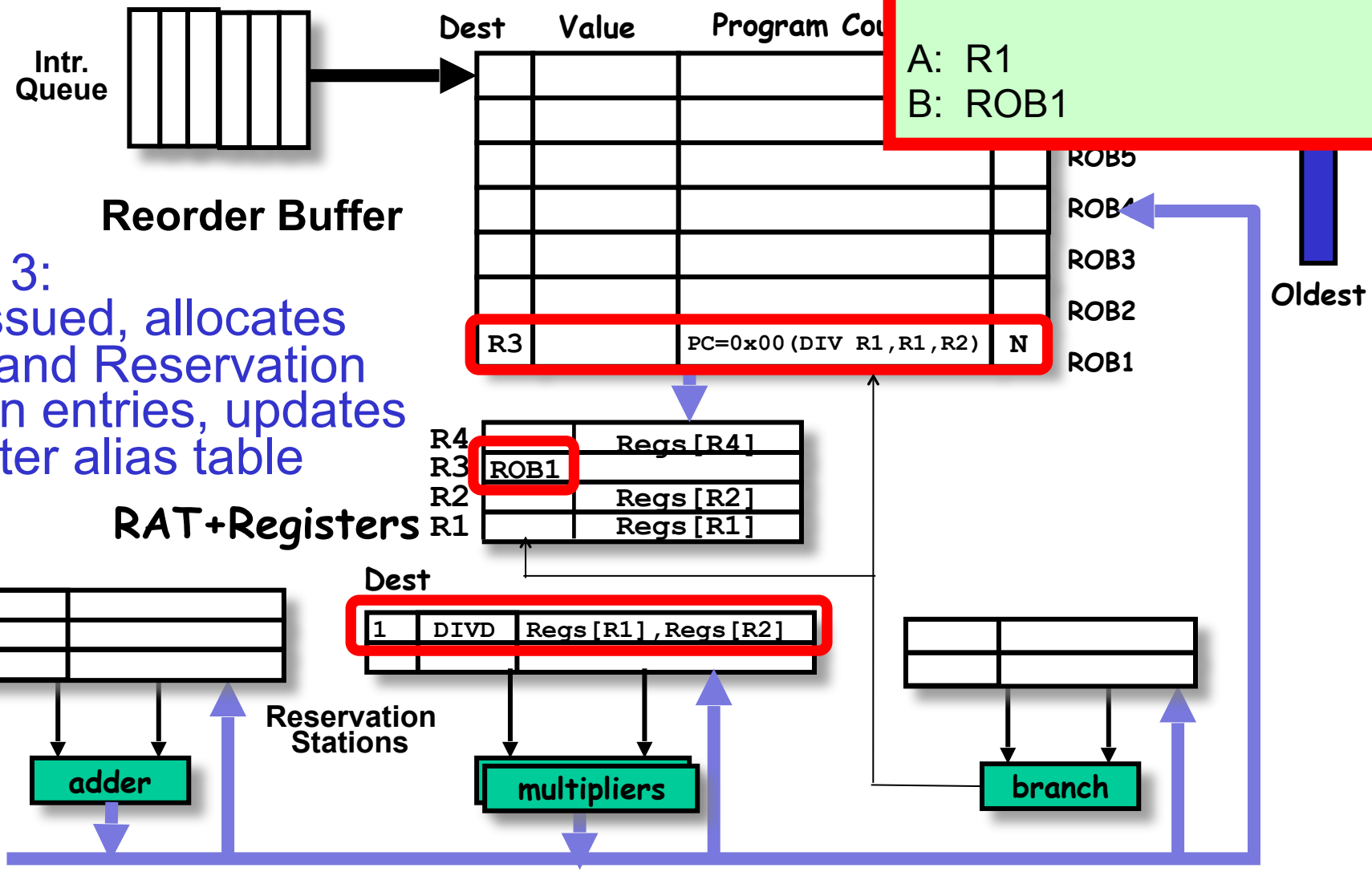
Tomasulo With Reorder Buffer



Tomasulo With Reorde

In reservation station for DIVD, what does “1” represent?

A: R1
B: ROB1

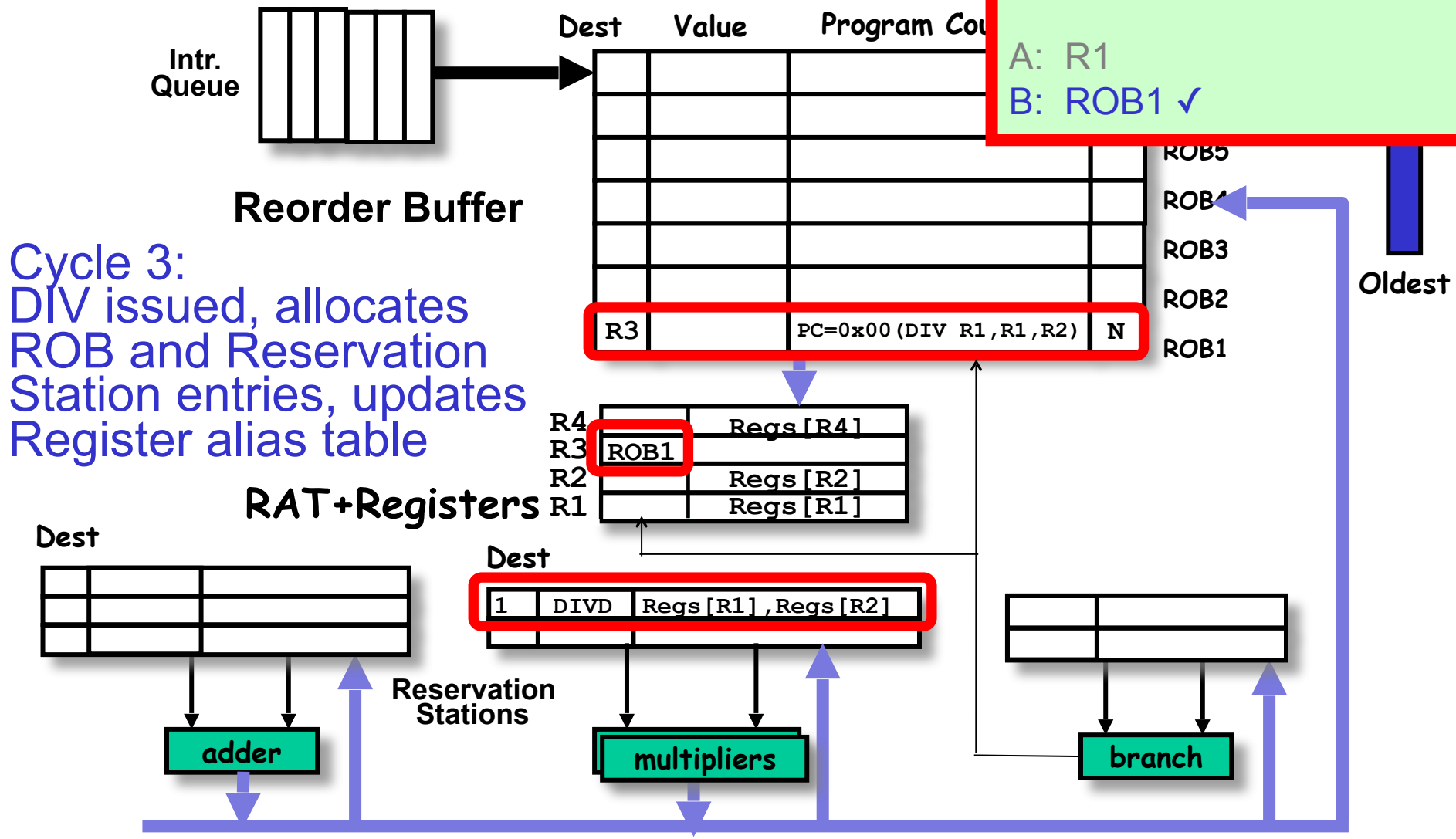


Cycle 3:
DIV issued, allocates ROB and Reservation Station entries, updates Register alias table

Tomasulo With Reorde

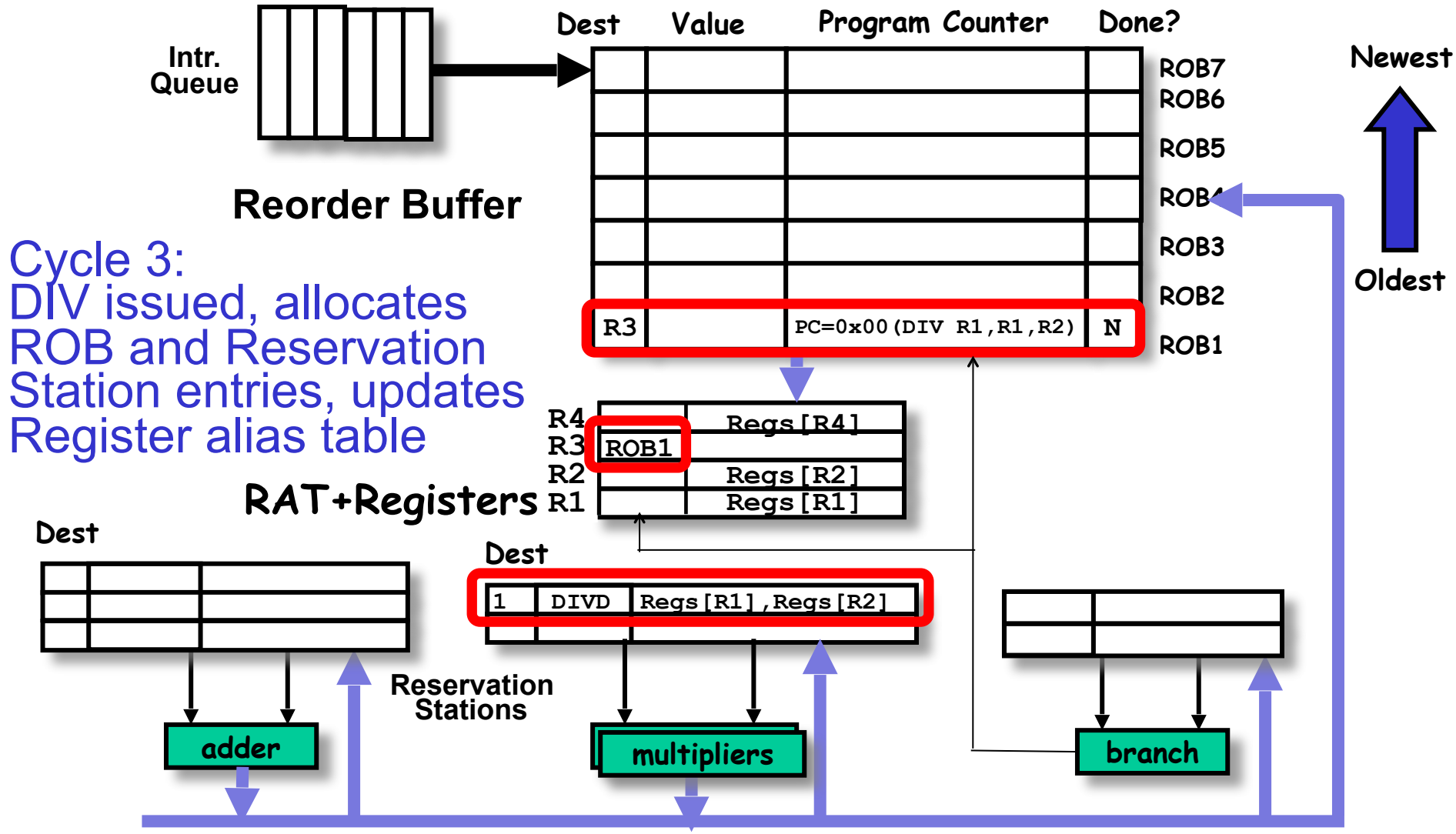
In reservation station for DIVD, what does “1” represent?

A: R1
B: ROB1 ✓

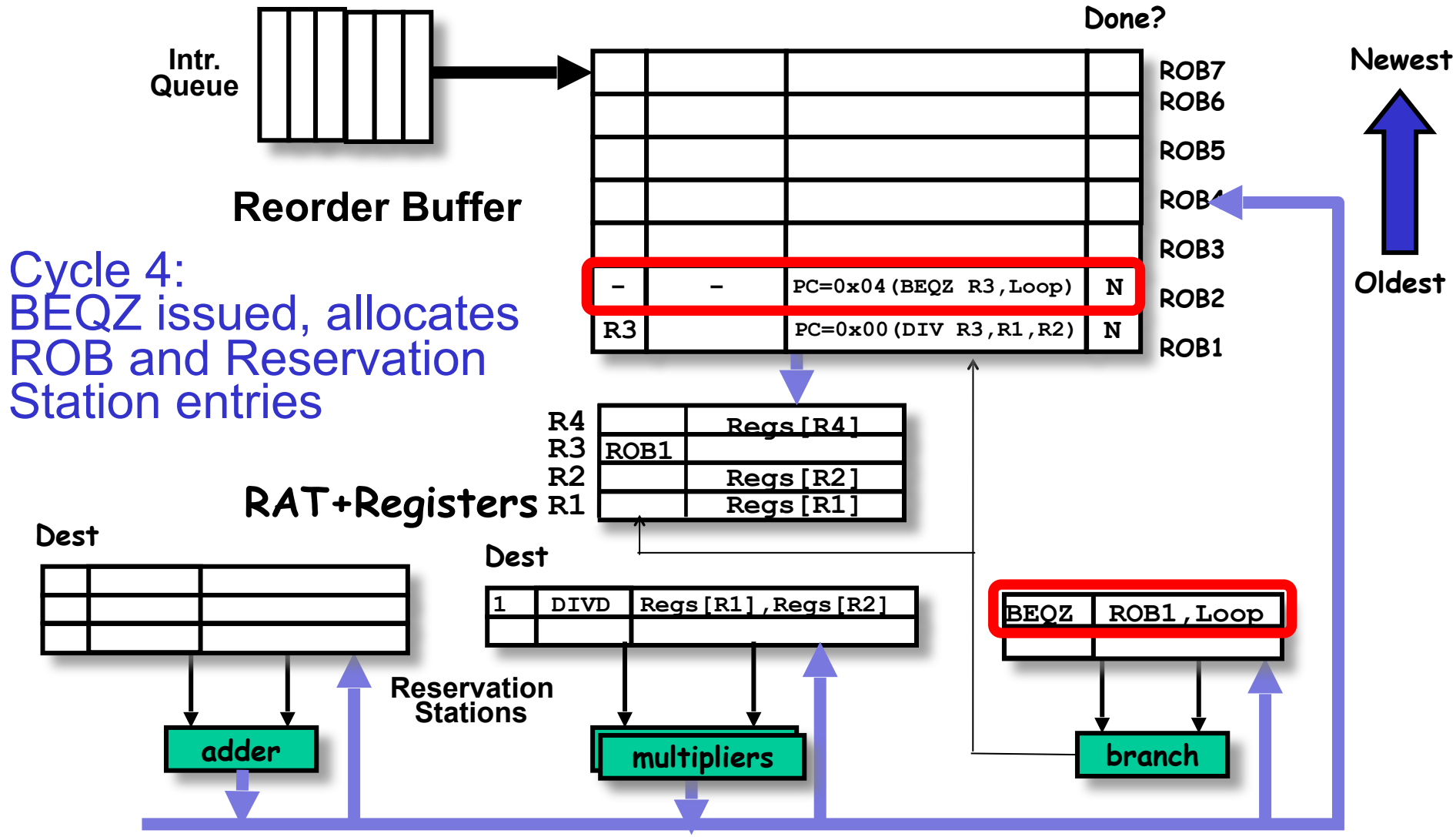


Cycle 3:
DIV issued, allocates ROB and Reservation Station entries, updates Register alias table

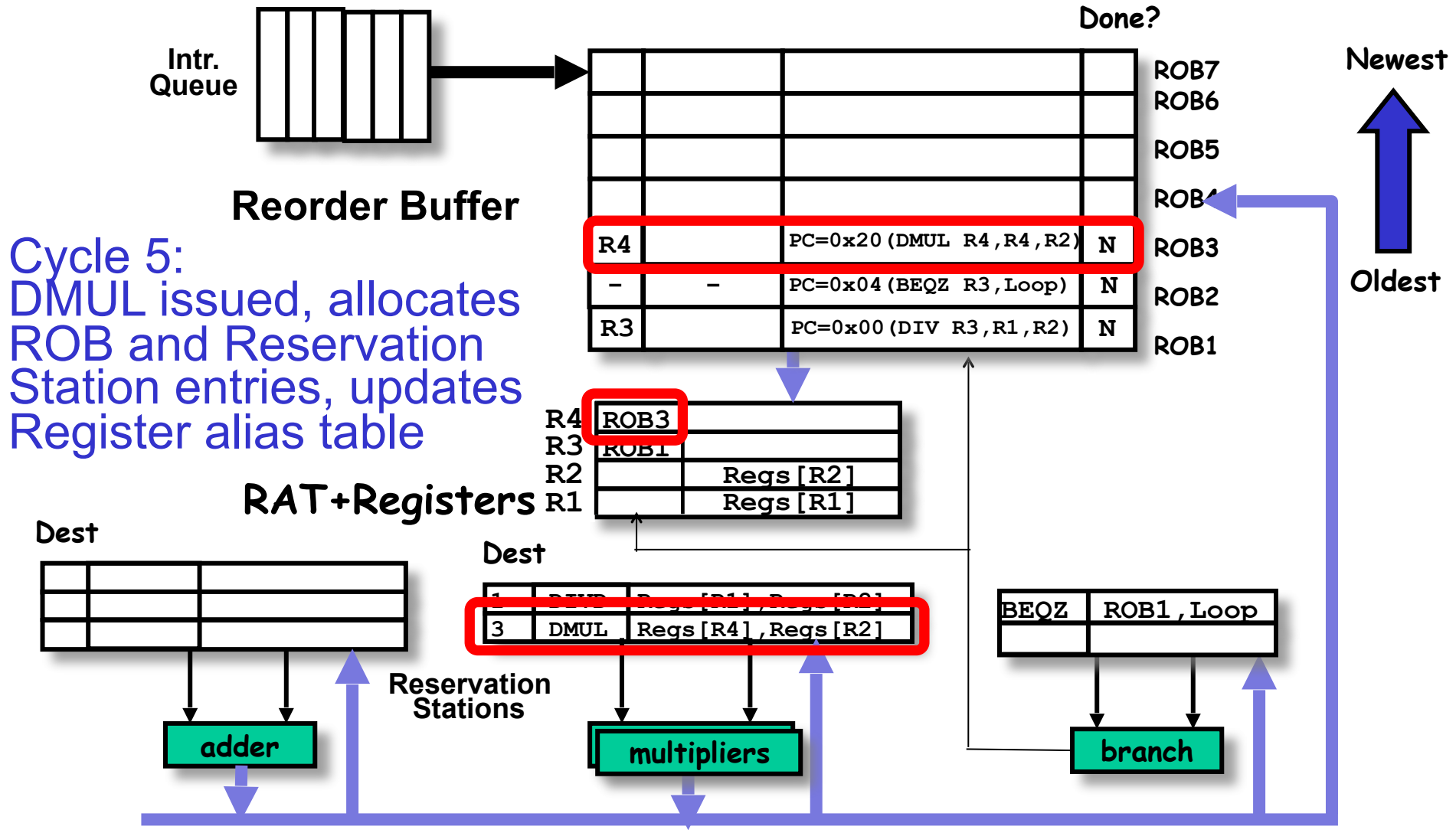
Tomasulo With Reorder Buffer



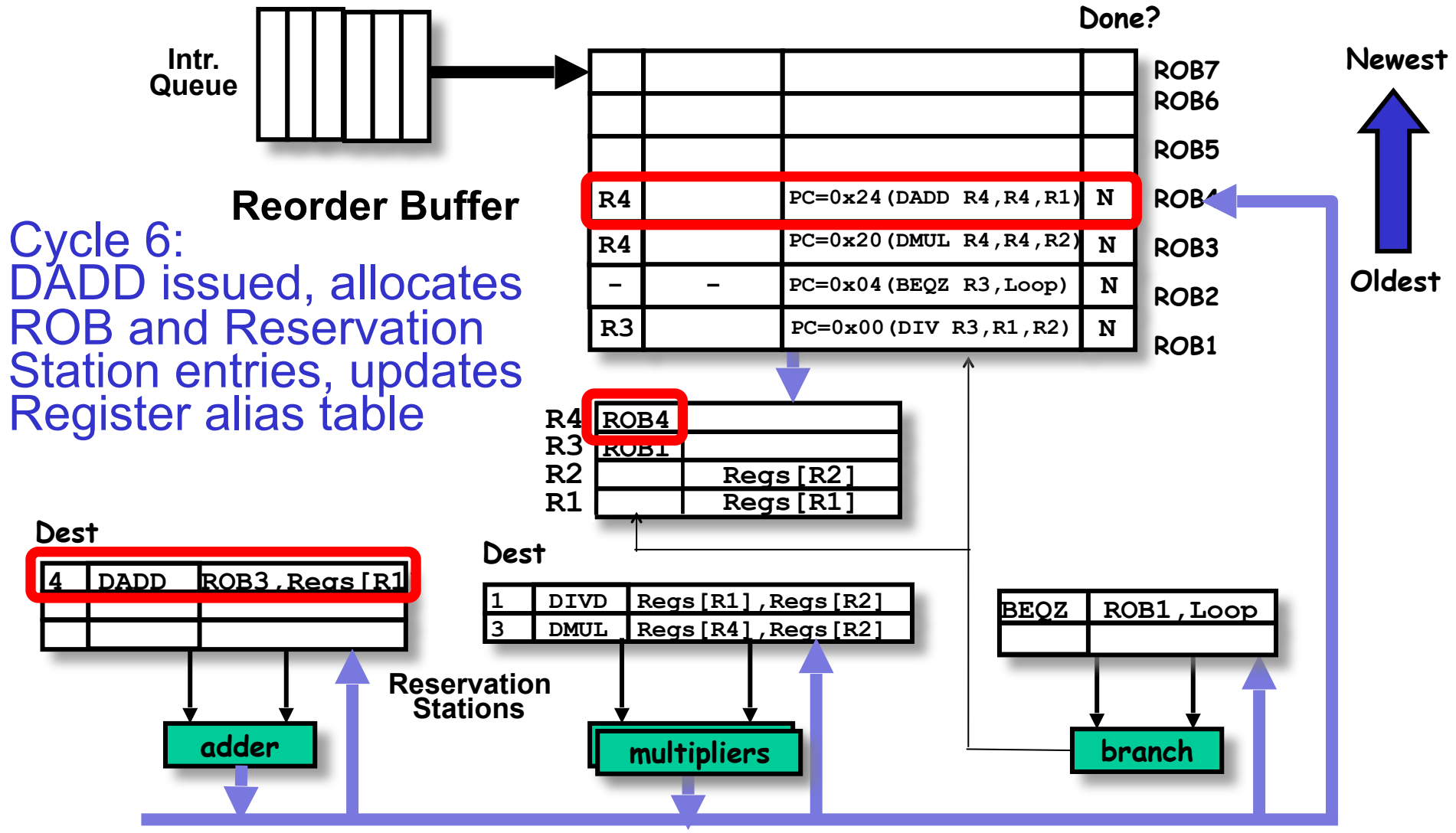
Tomasulo With Reorder Buffer



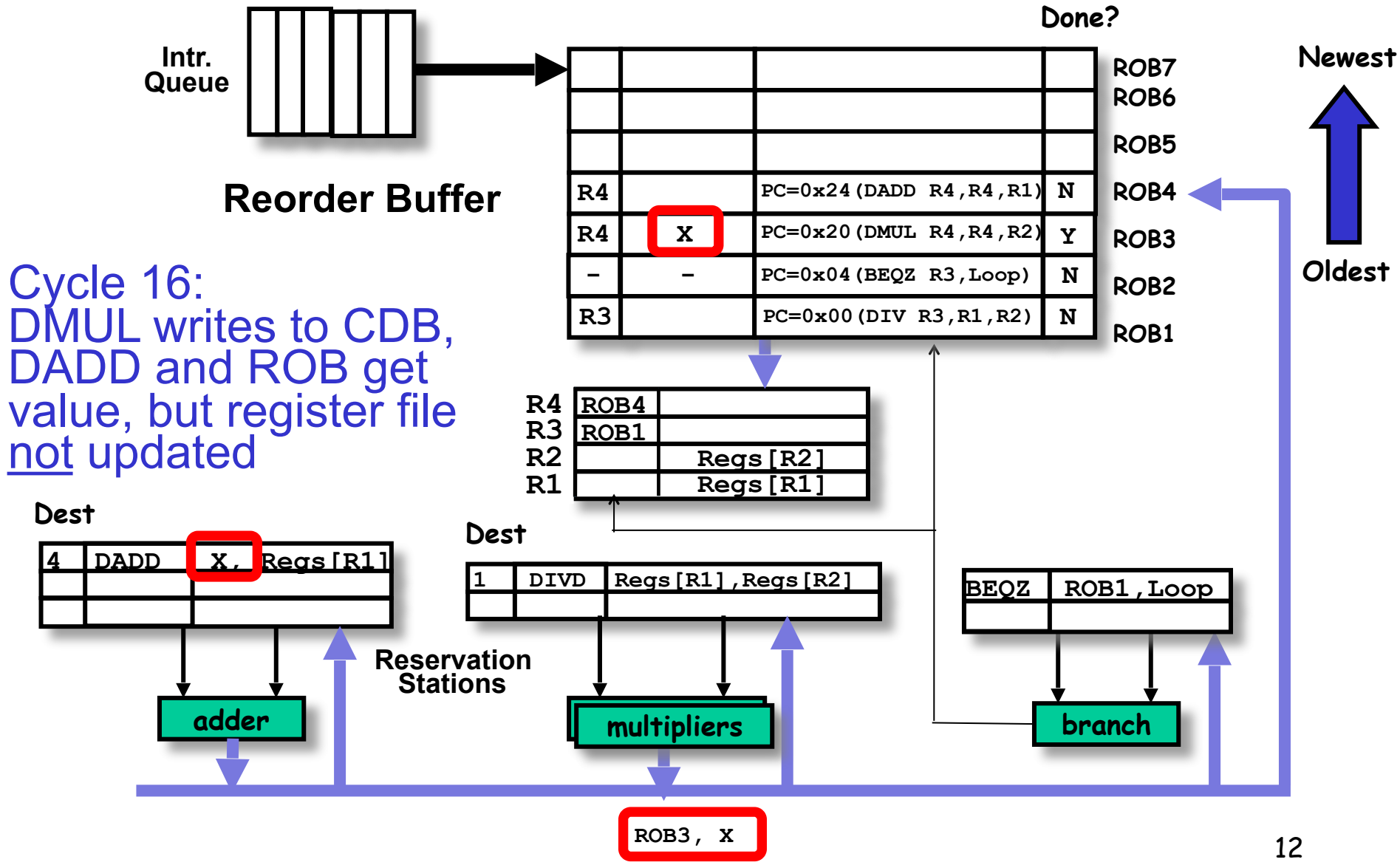
Tomasulo With Reorder Buffer



Tomasulo With Reorder Buffer

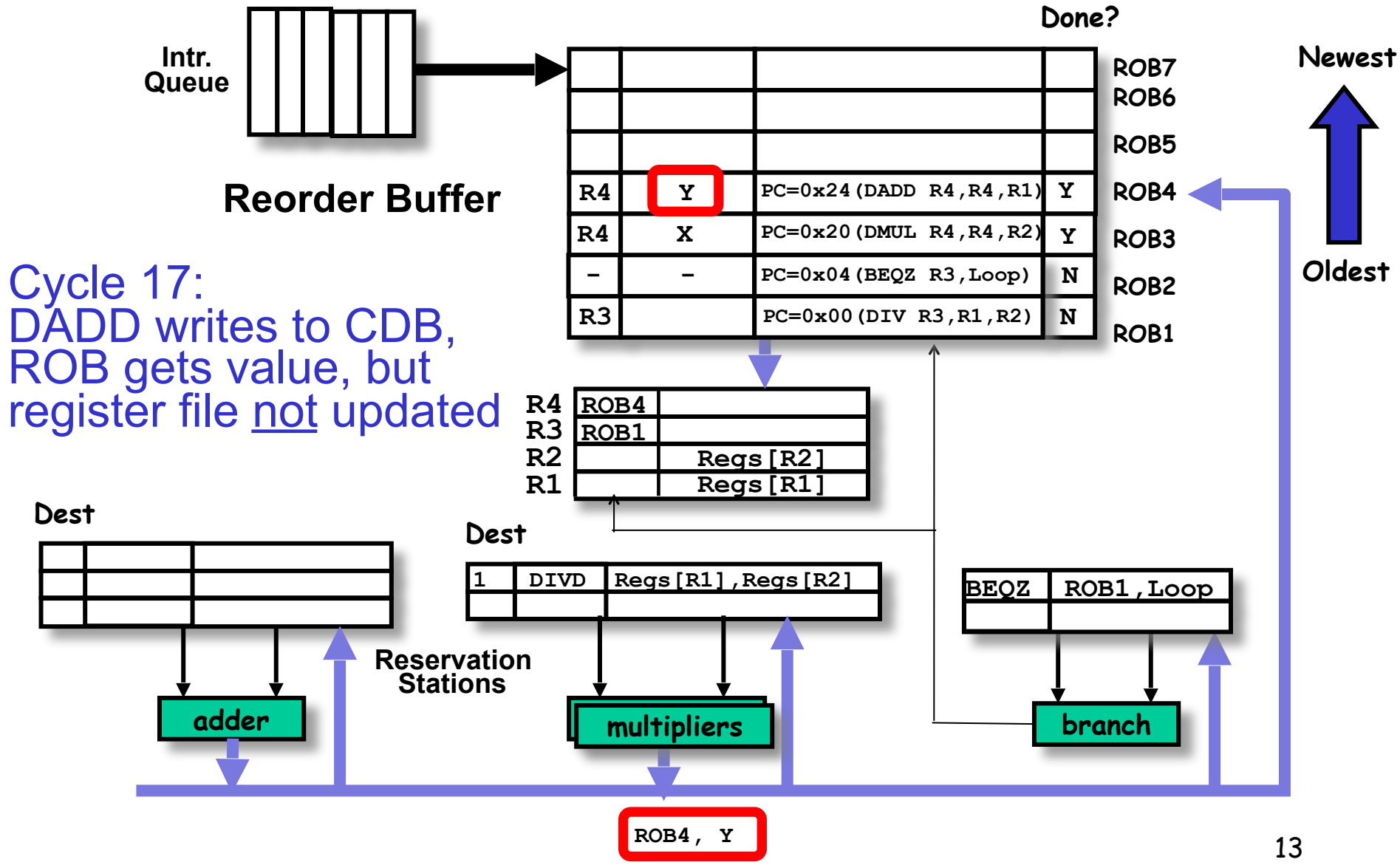


Tomasulo With Reorder Buffer

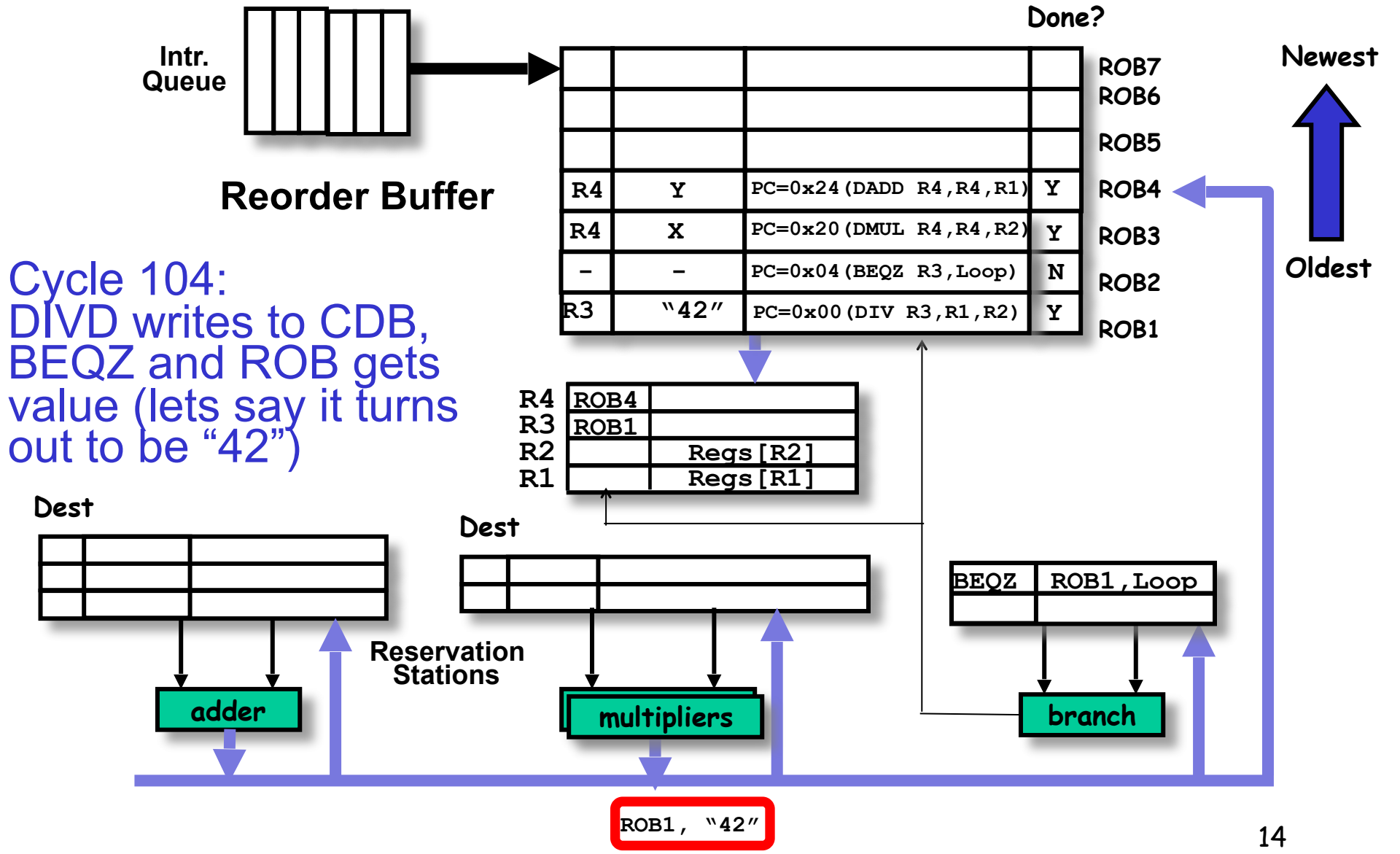


Cycle 16:
DMUL writes to CDB, DADD and ROB get value, but register file not updated

Tomasulo With Reorder Buffer

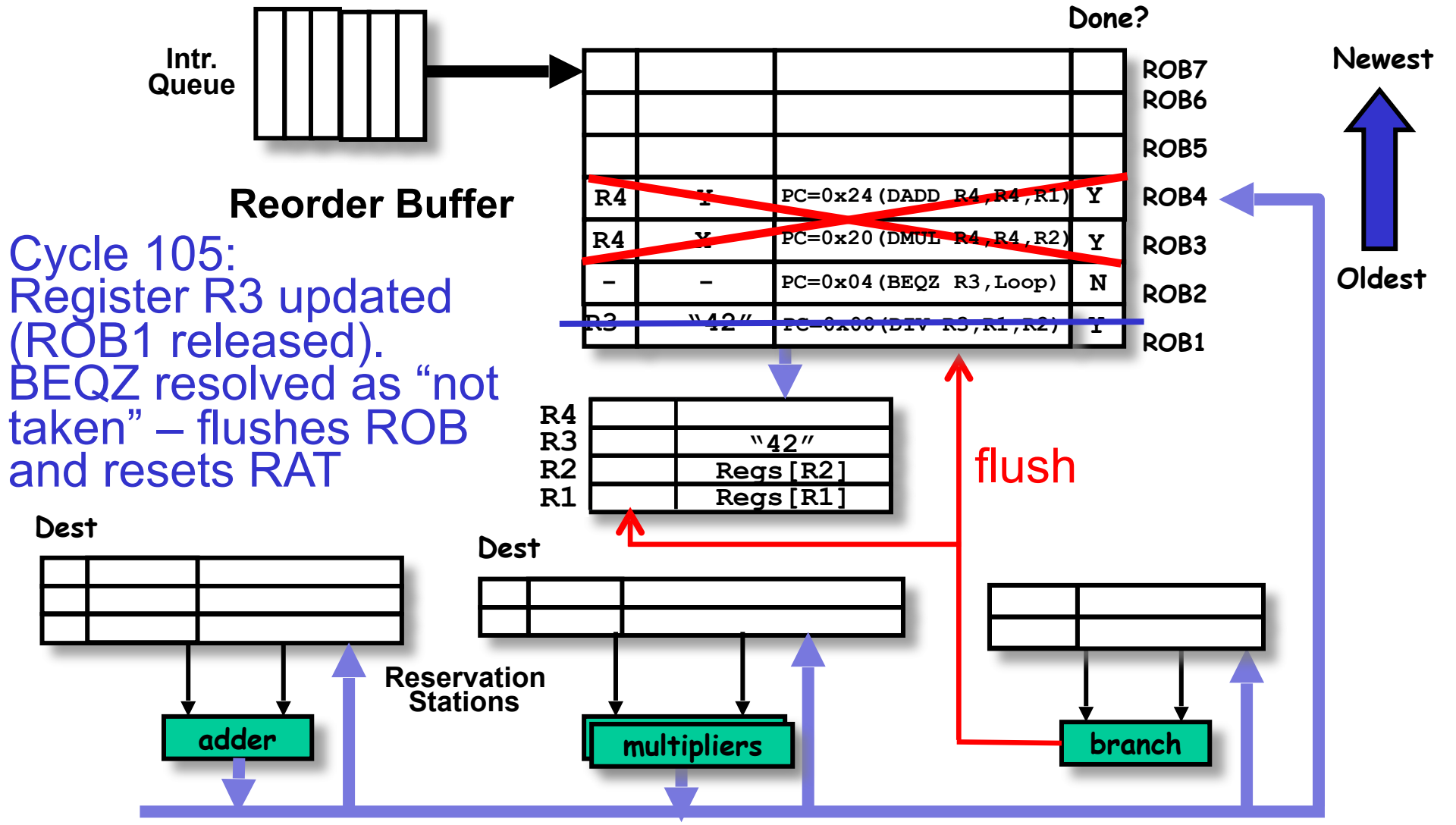


Tomasulo With Reorder Buffer



Cycle 104:
DIVD writes to CDB, BEQZ and ROB gets value (lets say it turns out to be "42")

Tomasulo With Reorder Buffer

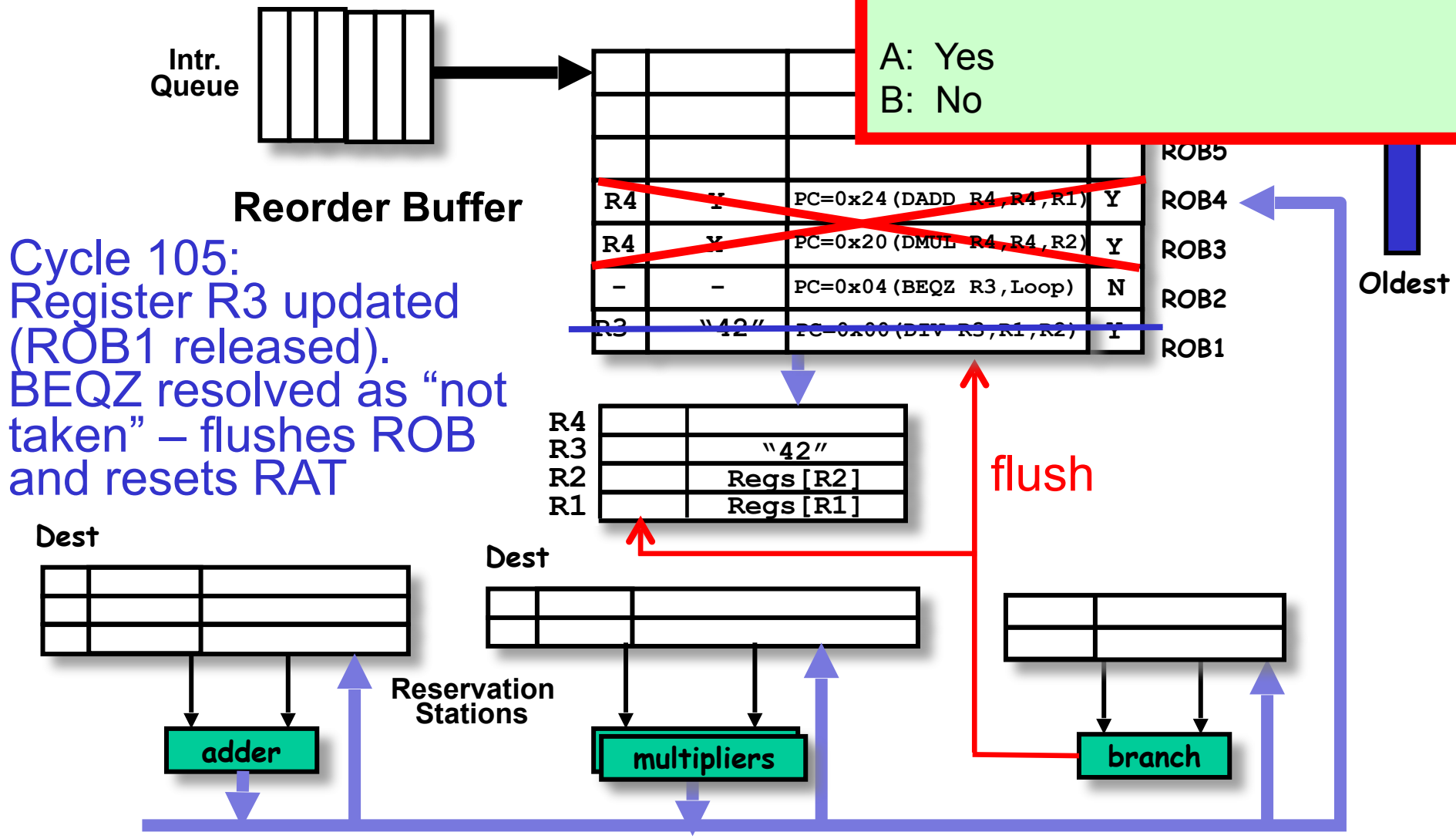


Cycle 105:
Register R3 updated (ROB1 released).
BEQZ resolved as "not taken" – flushes ROB and resets RAT

Tomasulo With Re

Since the branch was mispredicted should we flush it and execute it again too?

A: Yes
B: No

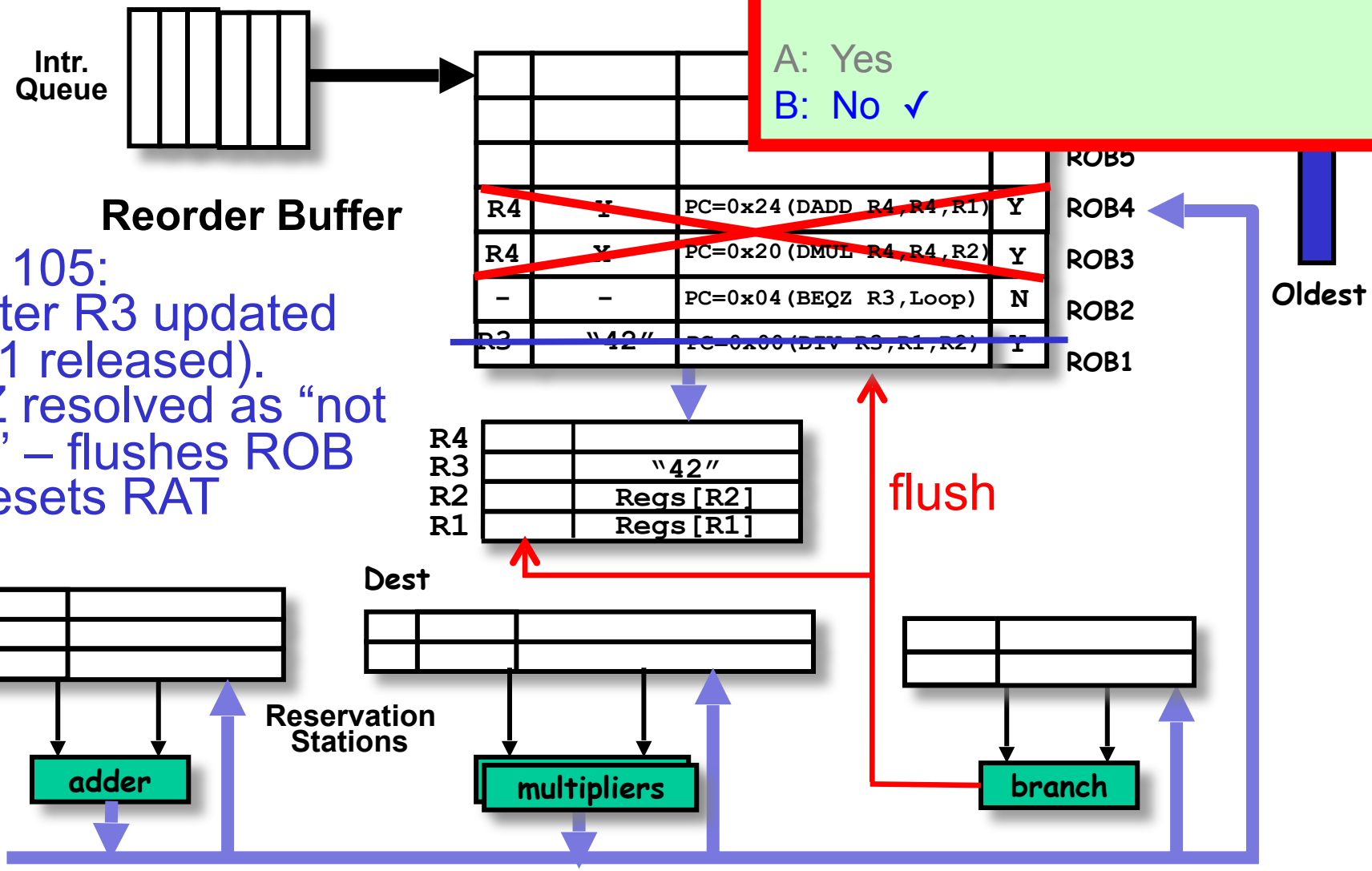


Cycle 105:
Register R3 updated (ROB1 released).
BEQZ resolved as "not taken" – flushes ROB and resets RAT

Tomasulo With Re

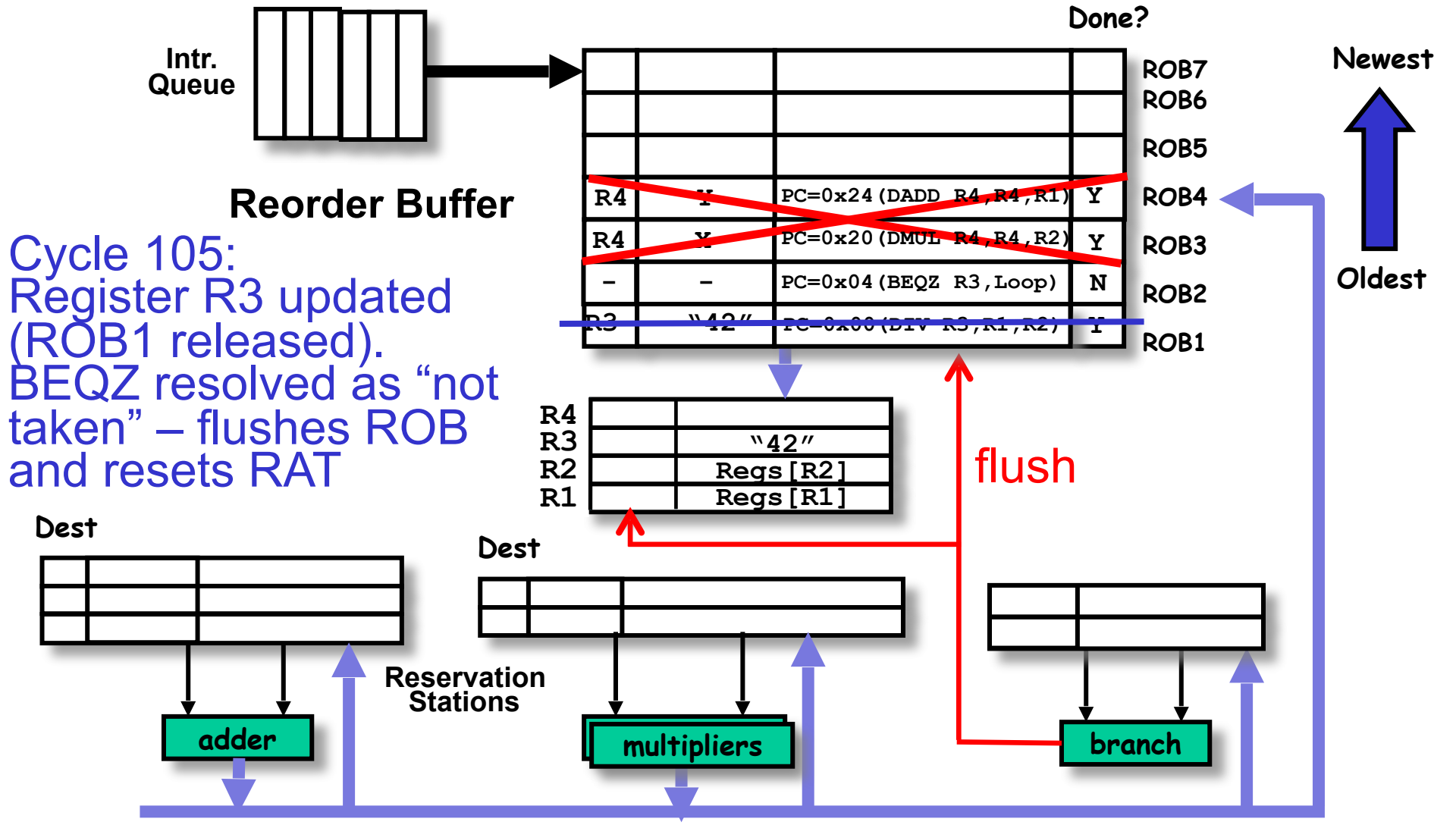
Since the branch was mispredicted should we flush it and execute it again too?

A: Yes
B: No ✓



Cycle 105:
Register R3 updated (ROB1 released).
BEQZ resolved as "not taken" – flushes ROB and resets RAT

Tomasulo With Reorder Buffer



Cycle 105:
Register R3 updated (ROB1 released).
BEQZ resolved as "not taken" – flushes ROB and resets RAT

Exceptions

- So far, we have considered how to design hardware to support higher performance execution of a program, but we assumed we can always execute an instruction.

- What happens if an instruction cannot complete or the program needs to ask the operating system to do something?
- If there is a bug, the program might tell the hardware to do something that is impossible. For example, divide by zero, or read/write the wrong location in memory. Another reason an instruction might not be able to complete is related to virtual memory.
- We call these “special cases” exceptions.

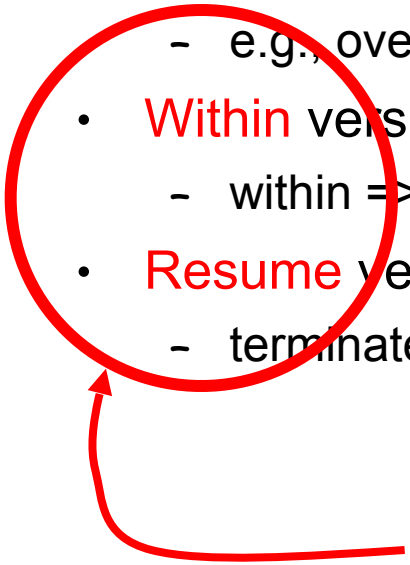
- The hardware detects exceptions and has a special program run that either handles the exception, or passes control to the operating system. The OS may handle the exception or end the program.

Exceptions

- Exceptions are also sometimes referred to “interrupts”, or “faults”. We’ll just call them all “exceptions”.
- Generally they require transfer of control to the operating system (OS) or other software “handler” routine.
- Causes:
  - I/O device request
  - Invoking an OS service from a user program
  - Page fault
  - Memory protection violation
  - FP arithmetic anomaly
  - Integer arithmetic overflow
  - etc...

Categories of Exceptions

- Synchronous versus Asynchronous
  - Async: external device or hardware failure.
- User requested versus coerced
  - user requested => handle after instruction
- User maskable versus user nonmaskable
  - e.g., overflow can be masked (ignored) by user.
- Within versus between instructions
  - within => hard
- Resume versus terminate
  - terminate => easier

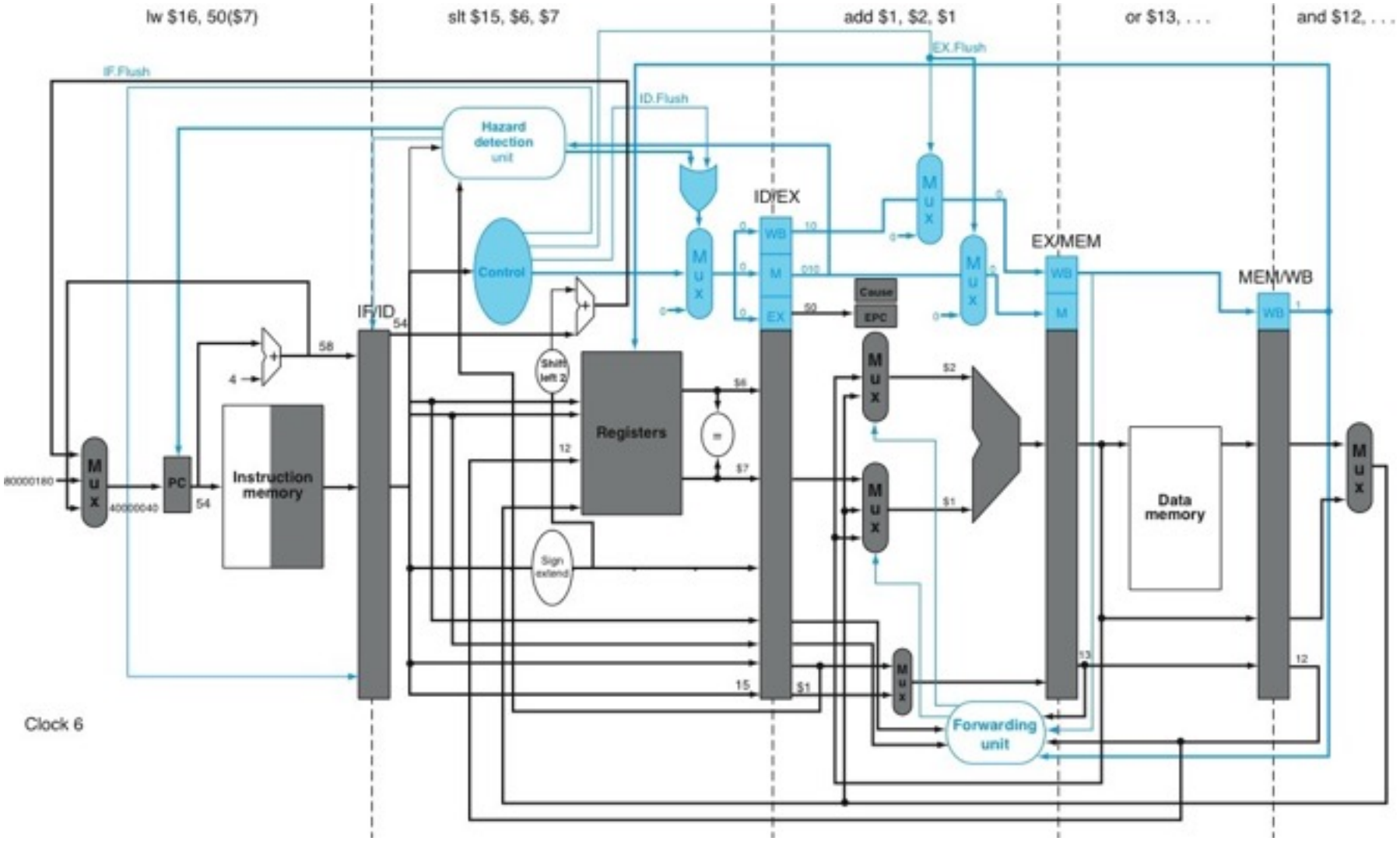


Most challenging, but very important => required for supporting “virtual memory” (which we will talk about later).

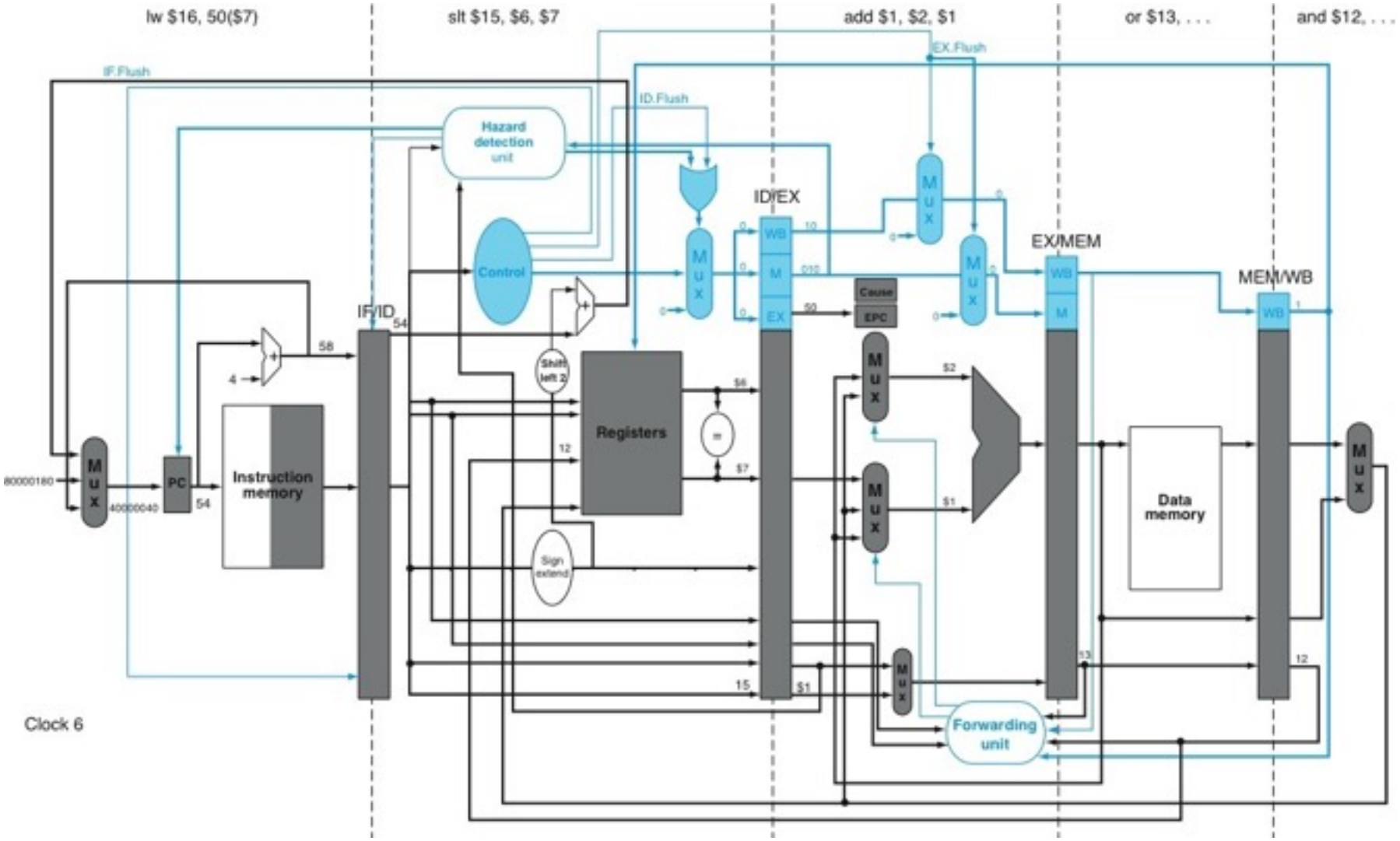
Exceptions in an In-order Pipeline

- Upon detection of exception:
  1. Force first instruction of an exception handler upon next IF
  2. Turn off control signals for all instructions after and including “faulting instruction” (i.e., convert them to “nops” / “squash” them)
  3. After exception handling routine gets control, save PC of faulting instruction (or N+1 PCs if delayed branch with N delay slots)

Exception Handling Support

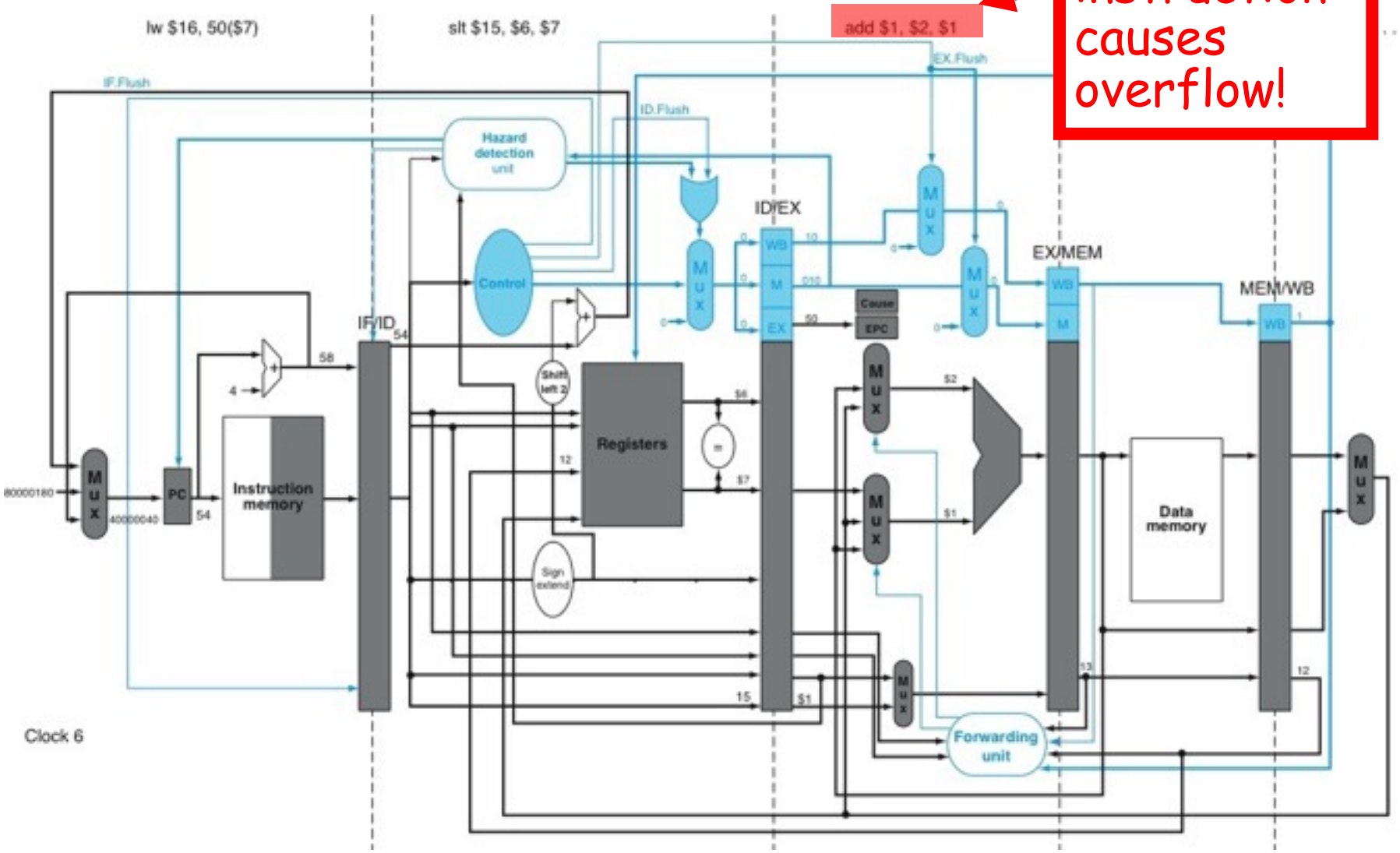


Exception Handling Support



Exception Handling Sup

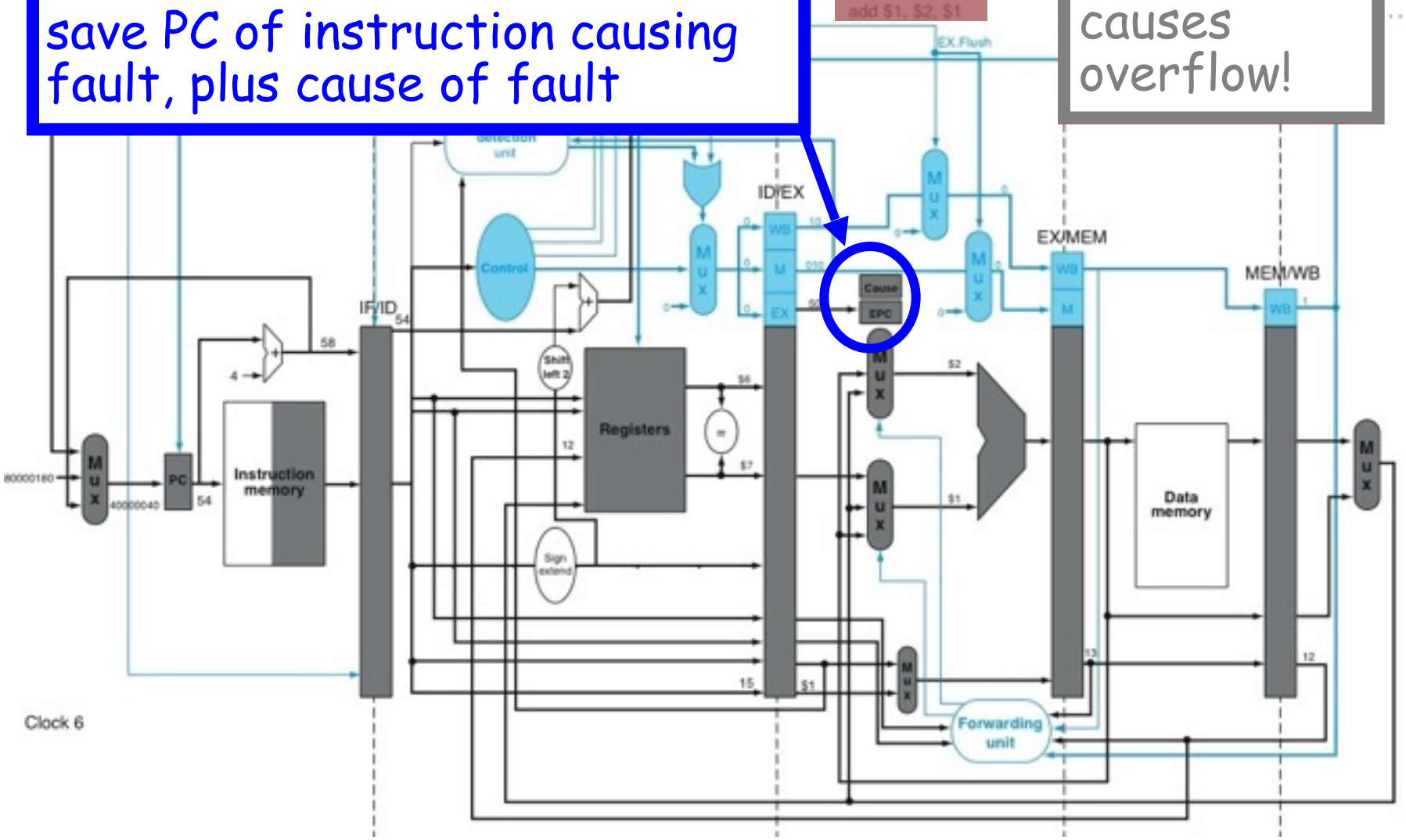
add instruction causes overflow!



Exception Handling Sup

save PC of instruction causing fault, plus cause of fault

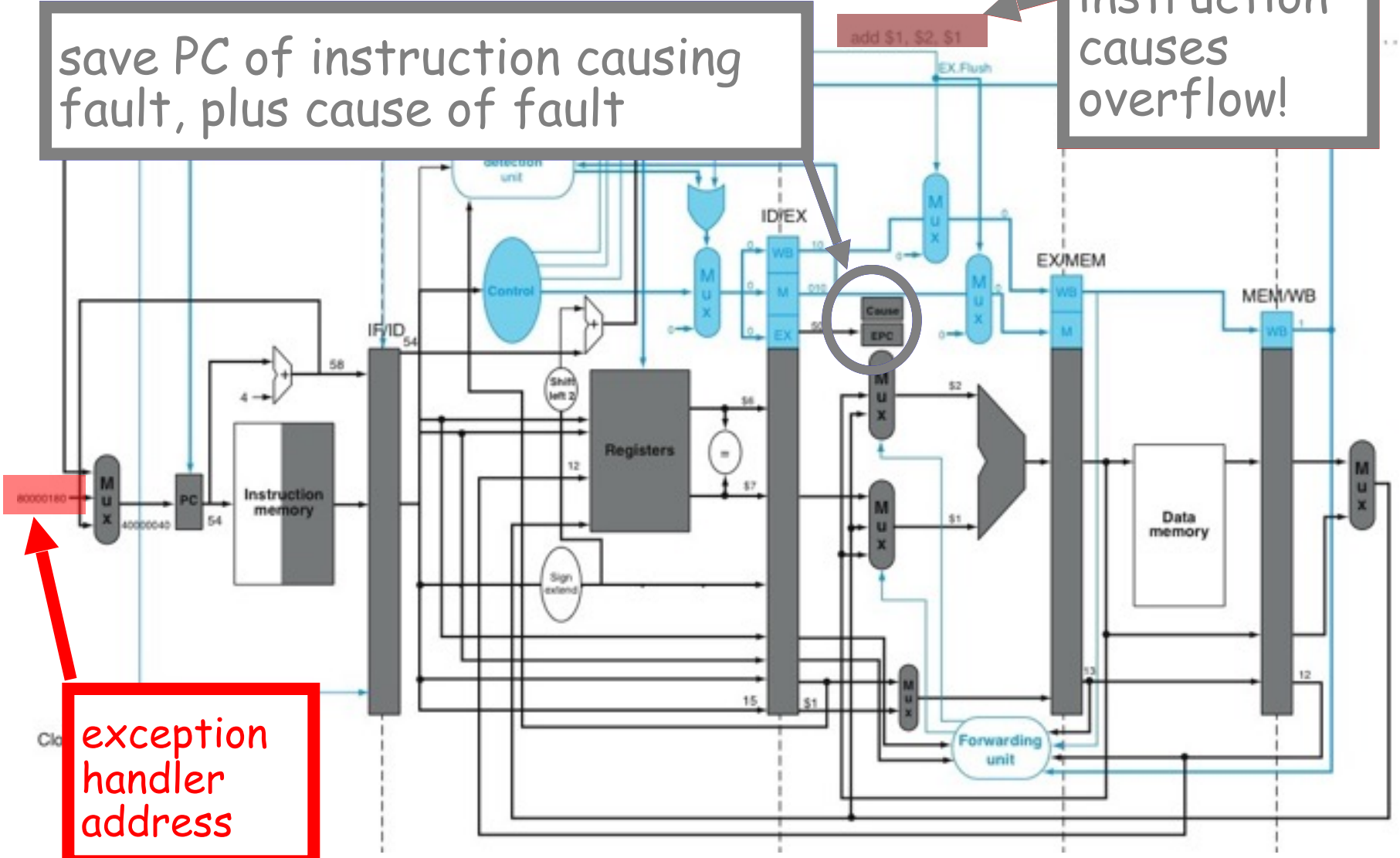
add instruction causes overflow!



Exception Handling Sup

save PC of instruction causing fault, plus cause of fault

add instruction causes overflow!

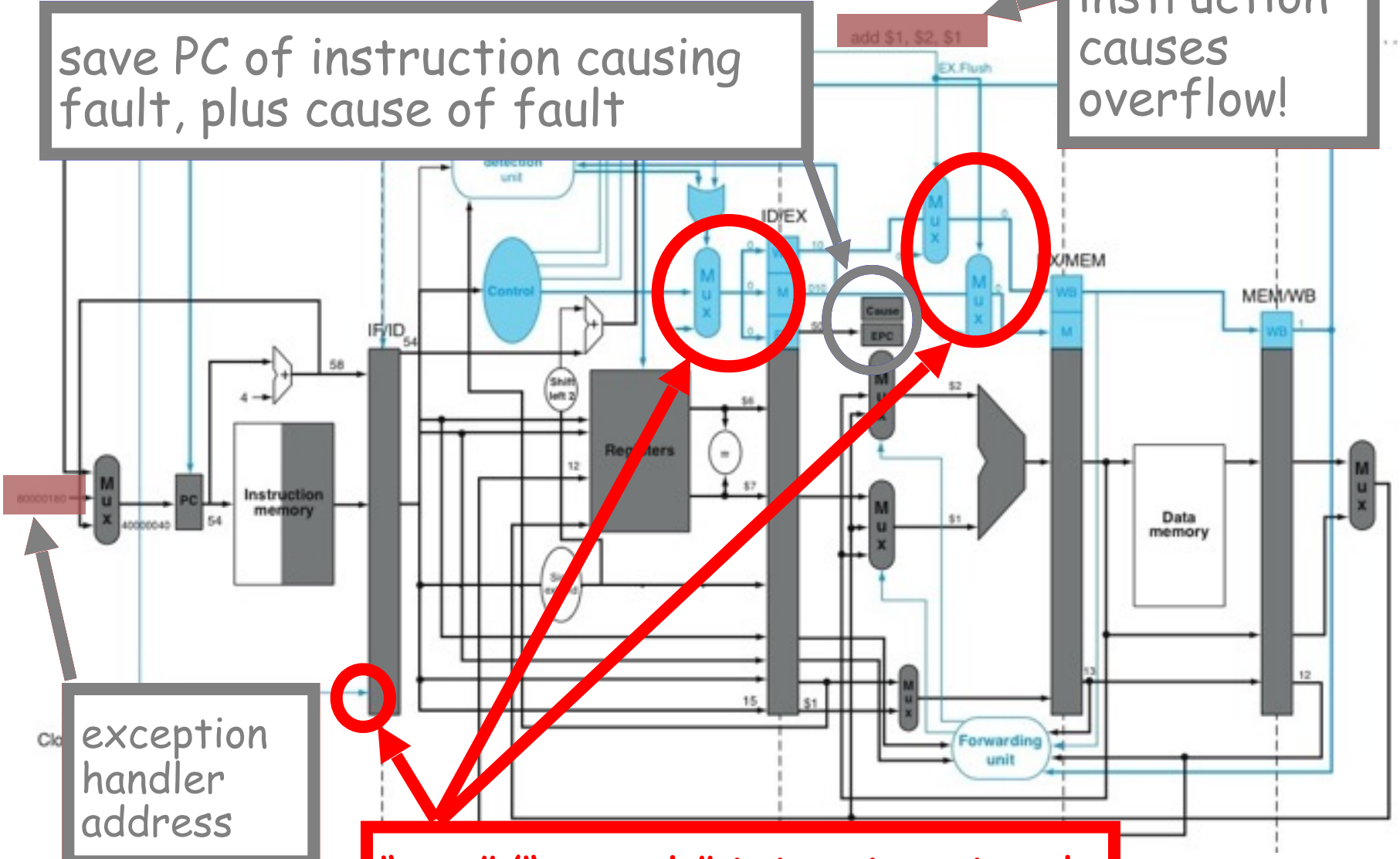


exception handler address

Exception Handling Sup

save PC of instruction causing fault, plus cause of fault

add instruction causes overflow!



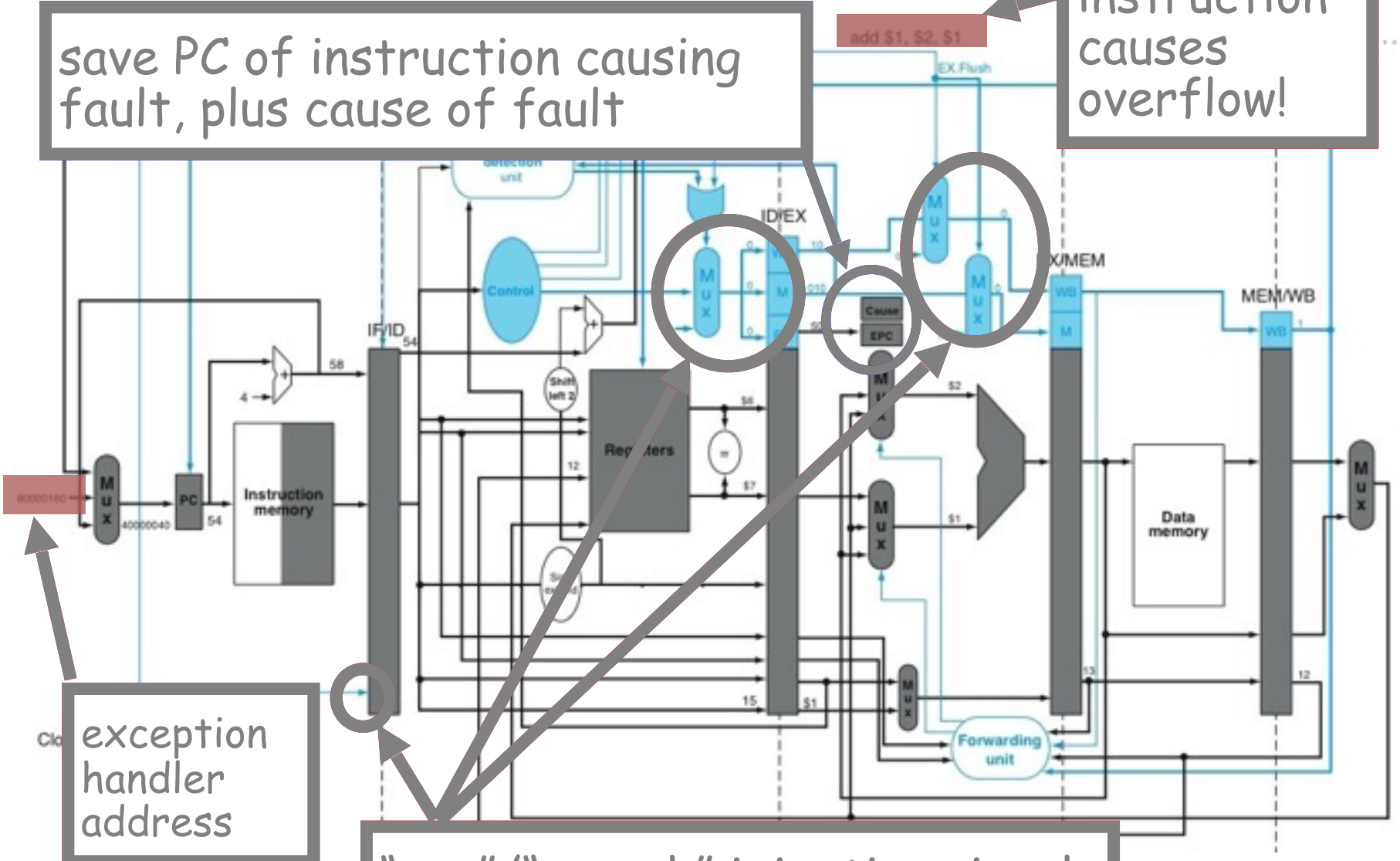
exception handler address

"nop"/"squash" injection signals

Exception Handling Sup

save PC of instruction causing fault, plus cause of fault

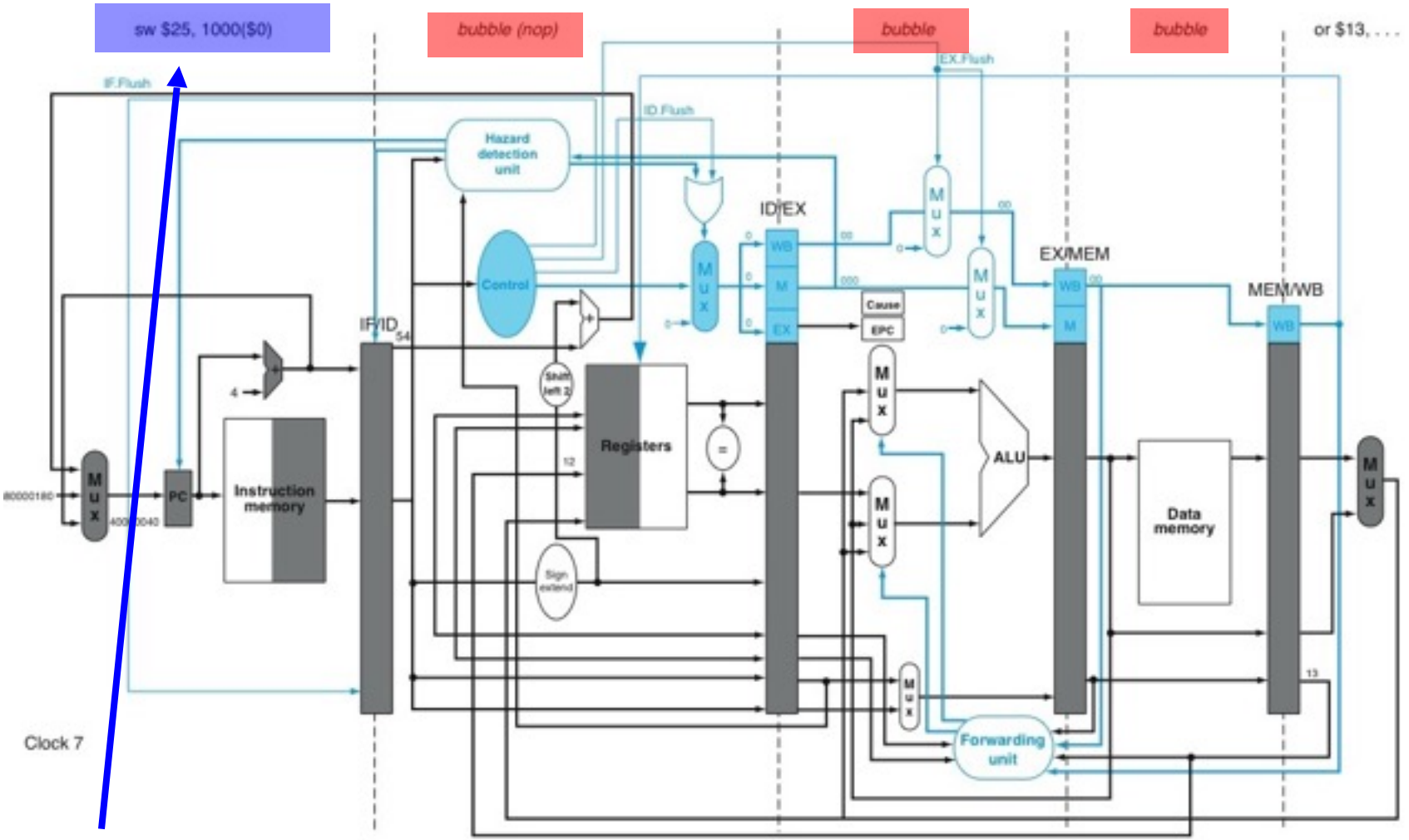
add instruction causes overflow!



exception handler address

"nop"/"squash" injection signals

Exception Handling Support



First instruction of handler

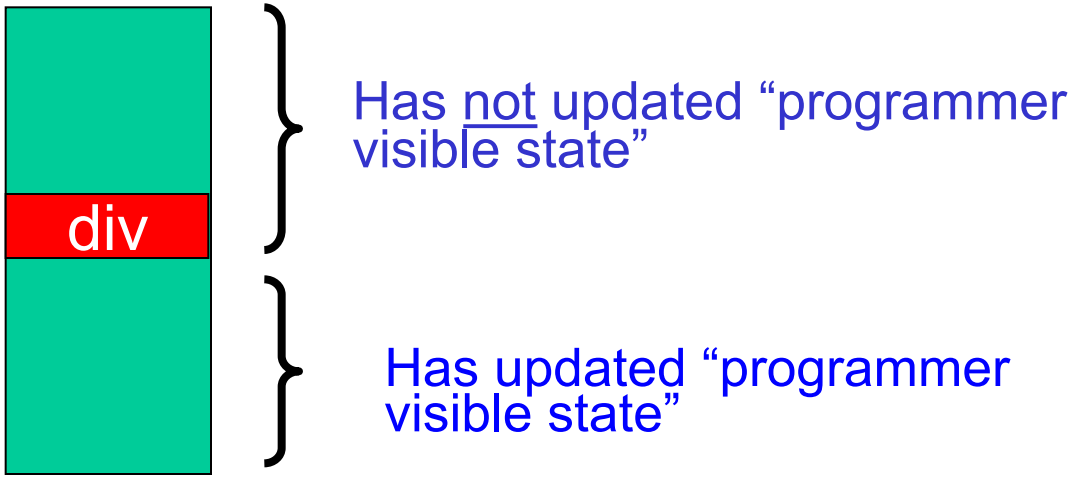
PAT06F43.eps

After exception handled...

- RFE instruction restarts pipeline (MIPS)

Precise Exceptions

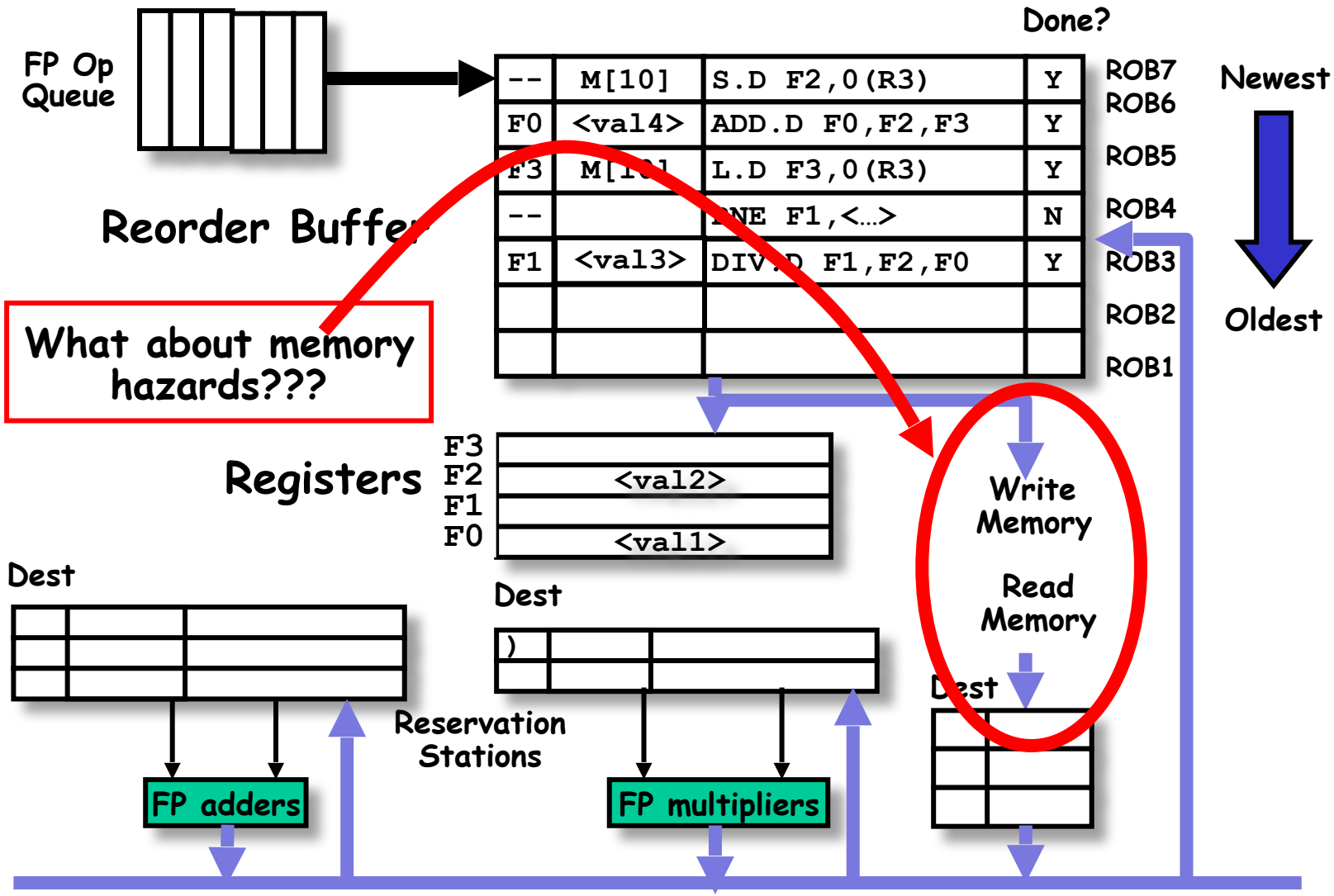
- If hardware guarantees that ALL instructions before faulting instruction have updated programmer visible state (i.e., R1...R31, memory) and NONE after (and including) faulting instruction have updated programmer visible state, then pipeline is said to support **precise exceptions**.



Precise Exceptions?

- Both Scoreboard and Tomasulo have:
  - In-order issue, out-of-order execution, out-of-order completion
- Out-of-order completion means that interrupts are not precise
- Need way to “resynchronize” execution with instruction stream
  - Easiest way is with reorder buffer

Memory Dependencies



Memory Disambiguation

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

Memory Disamb

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

Can “LD R6, 0(R3)” read memory before “SD R5,0(R2)” writes memory in this example?

A: Yes, definitely it can
B: Maybe
C: No, definitely not
D: Not sure

Memory Disamb

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

Can “LD R6, 0(R3)” read memory before “SD R5,0(R2)” writes memory in this example?

A: Yes, definitely it can
B: Maybe ✓
C: No, definitely not
D: Not sure

Memory Disambiguation

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

Memory Disambiguation

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

- If LD wants to read from a different location than the SD wants to write to, it is safe for the LD to read memory *before* the SD writes memory?

Memory Disambiguation

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

- If LD wants to read from a different location than the SD wants to write to, it is safe for the LD to read memory *before* the SD writes memory?

- Is this useful? Yes! Turns out to be very useful in out-of-order processors (e.g., Core i7 tries to do this).

Memory Disambiguation

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

- If LD wants to read from a different location than the SD wants to write to, it is safe for the LD to read memory *before* the SD writes memory?

- Is this useful? Yes! Turns out to be very useful in out-of-order processors (e.g., Core i7 tries to do this).

- When can LD read memory? Naïve answer is that we are not allowed to start load until we know that address 0(R2) ≠ 0(R3). More details next slide.

Memory Disambiguation

- Example:

```
SD R5,0(R2)
LD R6,0(R3)
```

- If LD wants to read from a different location than the SD wants to write to, it is safe for the LD to read memory *before* the SD writes memory?

- Is this useful? Yes! Turns out to be very useful in out-of-order processors (e.g., Core i7 tries to do this).

- When can LD read memory? Naïve answer is that we are not allowed to start load until we know that address 0(R2) ≠ 0(R3). More details next slide.

- More sophisticated answer (Andreas Moshovos, University of Toronto): "Predict" whether or not they are dependent (called "dependence prediction") and use reorder buffer to fixup if we are wrong. We won't talk more about this (I do talk about it in EECE 527).

Hardware Support for Memory Disambiguation

- Add “store queue” to keep track of all outstanding stores to memory, in program order.

Hardware Support for Memory Disambiguation

- Add “store queue” to keep track of all outstanding stores to memory, in program order.
  - Keep track of address (when becomes available) and value (when becomes available)
  - FIFO ordering: retire stores from this buffer in program order

Hardware Support for Memory Disambiguation

- Add “store queue” to keep track of all outstanding stores to memory, in program order.
  - Keep track of address (when becomes available) and value (when becomes available)
  - FIFO ordering: retire stores from this buffer in program order
- When issuing a load, record current head of store queue (know which stores are ahead of you).

Hardware Support for Memory Disambiguation

- Add “store queue” to keep track of all outstanding stores to memory, in program order.
  - Keep track of address (when becomes available) and value (when becomes available)
  - FIFO ordering: retire stores from this buffer in program order
- When issuing a load, record current head of store queue (know which stores are ahead of you).
- When have address for load, check store queue:
  - If *any* store prior to load is waiting for its address, stall load.
  - If load address matches earlier store address (associative lookup), then we have a *memory-induced RAW hazard*:
    - store value available ⇒ return value
    - store value not available ⇒ return ROB number of source
  - Otherwise, send out request to memory

Hardware Support for Memory Disambiguation

- Add “store queue” to keep track of all outstanding stores to memory, in program order.
  - Keep track of address (when becomes available) and value (when becomes available)
  - FIFO ordering: retire stores from this buffer in program order
- When issuing a load, record current head of store queue (know which stores are ahead of you).
- When have address for load, check store queue:
  - If *any* store prior to load is waiting for its address, stall load.
  - If load address matches earlier store address (associative lookup), then we have a *memory-induced RAW hazard*:
    - store value available ⇒ return value
    - store value not available ⇒ return ROB number of source
  - Otherwise, send out request to memory
- Stores modify memory during commit, which is “in order”, and this prevents WAR/WAW hazards through memory.

Limits to Performance

Pipeline CPI = Ideal CPI
+ Structural stalls
+ Data hazard stalls
+ Control hazard stalls

Have looked at how to remove/reduce stalls: pipelining, forwarding, out-of-order execution, branch prediction, speculative execution.

Can we reduce "Ideal CPI"?

Limits to Performance

Pipeline CPI = Ideal CPI
+ Structural stalls
+ Data hazard stalls
+ Control hazard stalls

Have looked at how to remove/reduce stalls: pipelining, forwarding, out-of-order execution, branch prediction, speculative execution.

Can we reduce "Ideal CPI"?

- Ideal CPI limited by rate at which instructions issued: 1 per cycle.

Limits to Performance

Pipeline CPI = Ideal CPI
+ Structural stalls
+ Data hazard stalls
+ Control hazard stalls

Have looked at how to remove/reduce stalls: pipelining, forwarding, out-of-order execution, branch prediction, speculative execution.

Can we reduce "Ideal CPI"?

- Ideal CPI limited by rate at which instructions issued: 1 per cycle.
- Modern microprocessors can achieve CPI < 1 by issuing more than one instruction per cycle.

Limits to Performance

Pipeline CPI = Ideal CPI
+ Structural stalls
+ Data hazard stalls
+ Control hazard stalls

Have looked at how to remove/reduce stalls: pipelining, forwarding, out-of-order execution, branch prediction, speculative execution.

Can we reduce "Ideal CPI"?

- Ideal CPI limited by rate at which instructions issued: 1 per cycle.
- Modern microprocessors can achieve CPI < 1 by issuing more than one instruction per cycle.
- We will start to speak about average *instructions per cycle (IPC)*

Limits to Performance

Pipeline CPI = Ideal CPI
+ Structural stalls
+ Data hazard stalls
+ Control hazard stalls

Have looked at how to remove/reduce stalls: pipelining, forwarding, out-of-order execution, branch prediction, speculative execution.

Can we reduce "Ideal CPI"?

- Ideal CPI limited by rate at which instructions issued: 1 per cycle.
- Modern microprocessors can achieve CPI < 1 by issuing more than one instruction per cycle.
- We will start to speak about average *instructions per cycle (IPC)*
- IPC = (1/CPI)

Multiple Instruction Issue

- We will just call it “multiple issue”

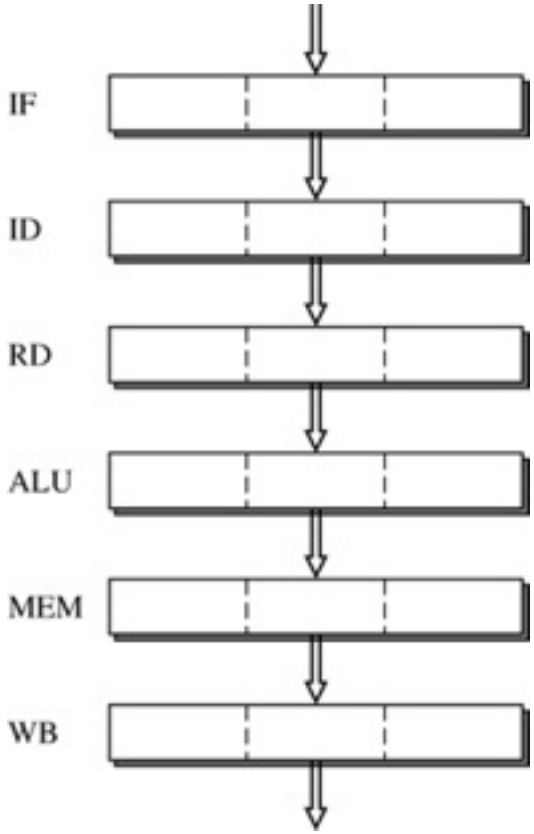
Multiple Instruction Issue

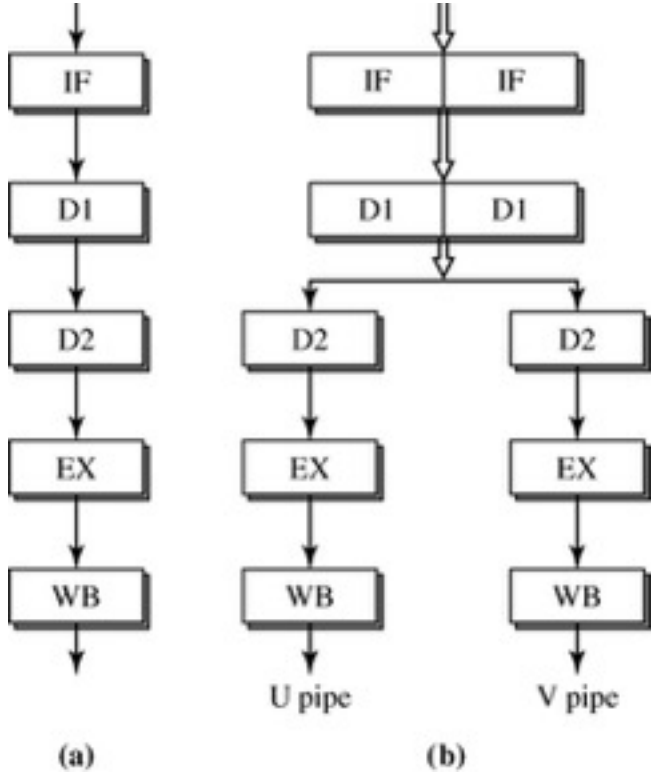
- We will just call it “multiple issue”
- Two types of multiple issue processor
  - Superscalar
    - Dynamic issue - varying number of instructions per cycle.
    - Statically or dynamically scheduled execution order
      - i.e., in order, out of order w/ or w/o speculative execution

Multiple Instruction Issue

- We will just call it “multiple issue”
- Two types of multiple issue processor
  - Superscalar
    - Dynamic issue - varying number of instructions per cycle.
    - Statically or dynamically scheduled execution order
      - i.e., in order, out of order w/ or w/o speculative execution
  - VLIW (very long instruction word)
    - Static issue - fixed number of instructions per cycle.
    - Statically scheduled execution

Inorder Superscalar Pipeline



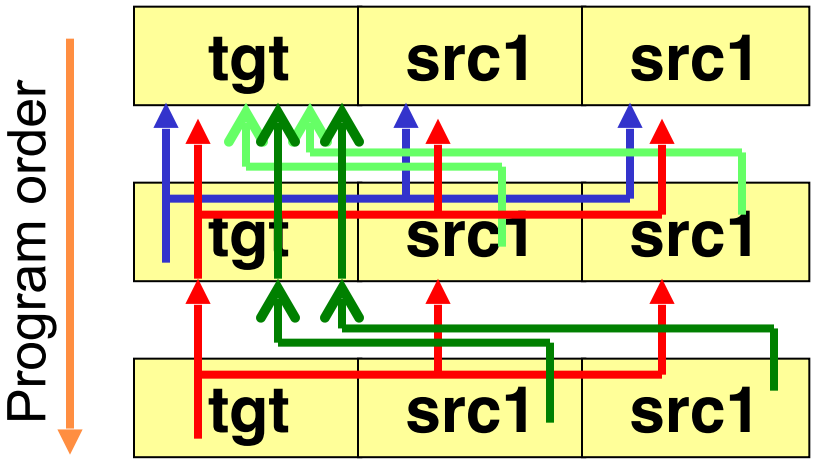


3-wide Superscalar 486 Pentium

(2-wide superscalar)

Statically Scheduled Superscalar

- Issue up to N instructions per cycle (where N is the “issue width” of the processor).
- Issue: Check each instruction for hazards with earlier instructions in program order issued this cycle, as well as against all earlier instructions still in execution.



Above: Comparisons to avoid RAW, WAR, and WAW hazards in a 3-wide in-order superscalar.

2-Issue In-order Superscalar

	Clock Number							
	1	2	3	4	5	6	7	8
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

- Issue restriction: At most one Integer and one Floating Point operation can issue per cycle (this organization was used in the HP 7100 processor).
- Challenge maintaining peak throughput: Next 3 instructions cannot use result of load; branch delay 4-5 instructions (if resolve in execute).

2-Issue In-order S

Assuming branch resolved in execute, what is the branch penalty measured in number of instructions squashed/flushed on a branch misprediction?

A: 1 instruction
B: 2-3 instructions
C: 4-5 instructions
D: 6-7 instructions
E: Not sure

	Cloc							
	1	2	3	4				
integer instruction	IF	ID	EX	MEM				
FP instruction	IF	ID	EX	EX				
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

- Issue restriction: At most one Integer and one Floating Point operation can issue per cycle (this organization was used in the HP 7100 processor).
- Challenge maintaining peak throughput: Next 3 instructions cannot use result of load; branch delay 4-5 instructions (if resolve in execute).

2-Issue In-order S

Assuming branch resolved in execute, what is the branch penalty measured in number of instructions squashed/flushed on a branch misprediction?

A: 1 instruction
B: 2-3 instructions
C: 4-5 instructions ✓
D: 6-7 instructions
E: Not sure

	Cloc							
	1	2	3	4				
integer instruction	IF	ID	EX	MEM				
FP instruction	IF	ID	EX	EX				
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

- Issue restriction: At most one Integer and one Floating Point operation can issue per cycle (this organization was used in the HP 7100 processor).
- Challenge maintaining peak throughput: Next 3 instructions cannot use result of load; branch delay 4-5 instructions (if resolve in execute).

2-Issue In-order Superscalar

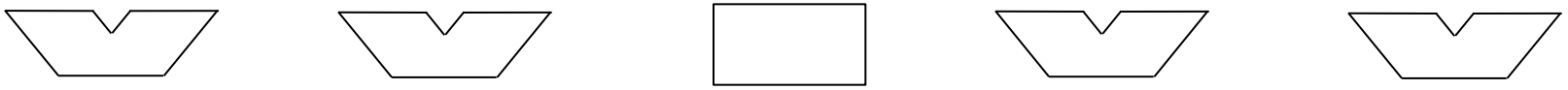
	Clock Number							
	1	2	3	4	5	6	7	8
integer instruction	IF	ID	EX	MEM	WB			
FP instruction	IF	ID	EX	EX	EX	WB		
integer instruction		IF	ID	EX	MEM	WB		
FP instruction		IF	ID	EX	EX	EX	WB	
integer instruction			IF	ID	EX	MEM	WB	
FP instruction			IF	ID	EX	EX	EX	WB

- Issue restriction: At most one Integer and one Floating Point operation can issue per cycle (this organization was used in the HP 7100 processor).
- Challenge maintaining peak throughput: Next 3 instructions cannot use result of load; branch delay 4-5 instructions (if resolve in execute).

Very Long Instruction Word (VLIW)

- In a superscalar processor, we need to check for dependencies between instructions before they can be issued in parallel.
- Each time we fetch and decode the same group of instructions we will find the same dependencies (through registers).
- Why not let the compiler search for these dependencies and groups of instructions that are known to be independent? This is how a VLIW processor works
- Drawbacks:
  - Need to explicitly encode “no ops” for units that do not have an operation.
  - Hard to predict which memory access instructions miss in cache
  - Need to use profile feedback to find “hot traces” in program to extract instruction level parallelism
  - Memory dependencies hard to determine in compiler

Inst N:	ADD.D F1,F1,F4	**NOP**	L.D F7, 0(R1)	DSUBI R2,R2,#1	**NOP**
Inst N+1:	ADD.D F1,F1,F7	MUL.D F4,F5,F6	**NOP**	DSUBI R2,R2,#1	BEQZ R2, Loop

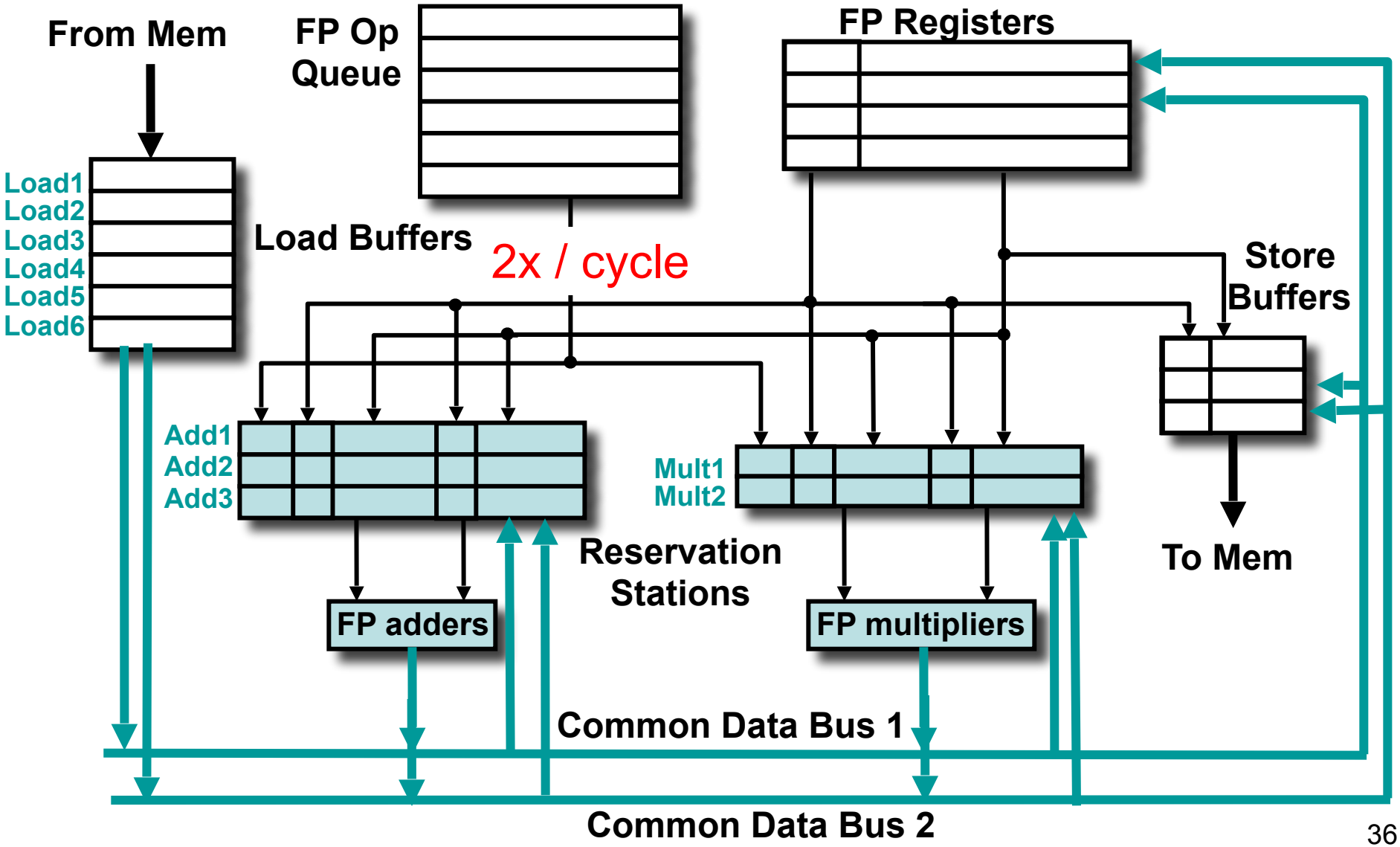


Multiple Issue w/ Dynamic Scheduling

- Use extension of Tomasulo's algorithm.
- Benefits:
  - Overcome data hazards
  - Overcome issue restrictions

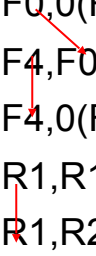
- Key challenge: for each instruction in issue packet, in program order
  - Assign reservation station and update pipeline control tables
- Two approaches to overcome this challenge:
  - Run this step at higher clock frequency (i.e., do in half a clock cycle for 2-wide superscalar)
  - Add logic to perform operation on multiple (e.g., 2) instructions at once.

Tomasulo Organization (with two-wide issue)



Example

```
Loop:   L.D     F0,0(R1)    ; F0-array element
        ADD.D   F4,F0,F2    ; add scalar in F2
        S.D     F4,0(F1)    ; store result
        DADDIU  R1,R1,#-8   ; decrement pointer 8 bytes (per DW)
        BNE     R1,R2,Loop      ; branch R1 != R2
```



Assumptions:

(1) 2-wide issue

(2) Infinite number of reservation stations

(3) Perfect branch prediction (let's do three iterations of loop); separate branch unit.

(4) Issue instruction from target of taken branch 1 cycle after branch (due to fetch restrictions)

(5) One Integer Unit (handles load/store effective address calculation)

(6) Separate FP Function Unit for each type of FP operation

(7) Issue and Write Results take one cycle

(8) Latencies: One cycle for integer ALU; two cycles for loads; three cycles for FP adds

(9) Two CDBs

(10) Load/Store effective address calculation "decoupled" from memory access (e.g., compute effective address as soon as possible, even if data to store to memory is not available)

(11) NO Reorder Buffer

Example: 2-issue w/ Tomasulo

Clock Cycle: 1

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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					

Clock number	Integer ALU	FP ALU	Data cache	CDB
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 2

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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					

Clock number	Integer ALU	FP ALU	Data cache	CDB
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 2

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	Comment
1	L.D	F0,0(R1)	1	2			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					

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

Example: 2-issue w/ To

Execute ADD.D on cycle 2?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 2

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	Comment
1	L.D	F0,0(R1)	1	2			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					

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

Example: 2-issue w/ To

Execute ADD.D on cycle 2?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure ✓

Clock Cycle: 2

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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)					
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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 2

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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				
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					

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

Examp asulo

Execute S.D on cycle 3?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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				
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					

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

Examp asulo

Execute S.D on cycle 3?

A: Yes, very sure ✓
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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				
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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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				
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					

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

Example: 2-issue w/ T

Execute DADDIU on cycle 3?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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				
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					

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

Example: 2-issue w/ T

Execute DADDIU on cycle 3?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure ✓

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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					
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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 3

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 4

Iteration number	Instructions		Issues at	Executes	Memory access at	Write CDB at	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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 4

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				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					

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

Example: 2-issue w/

S.D access memory on cycle 4?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 4

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				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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 4

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				Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
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					

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

Examp asulo

Execute DADDUI on cycle 4?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 4

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				Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
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					

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

Examp asulo

Execute DADDUI on cycle 4?

A: Yes, very sure ✓
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 4

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				Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4			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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 4

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				Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4			Wait for ALU
1	BNE	R1,R2,Loop	3				Wait for DADDIU
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					

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				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 4

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				Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4			Wait for ALU
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					

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

Example: 2-issue w/ Tomasulo

Clock Cycle: 5

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				Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4			Wait for ALU
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					

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				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4			Wait for ALU
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					

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		
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ To

Execute 2/L.D on cycle 5?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ To

Execute 2/L.D on cycle 5?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure ✓

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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				
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ To

Execute 2/L.D on cycle 5?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure ✓

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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				
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ To

Execute 2/L.D on cycle 5?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure ✓

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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 L.D
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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 L.D
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 5

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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 L.D
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Example: 2-issue w/ Tomasulo

Clock Cycle: 5

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			Wait for L.D
1	S.D F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU R1,R1,#-8	2	4		5	Wait for ALU
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 L.D
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 6

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU	R1,R1,#-8	2	4		5	Wait for ALU
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 L.D
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					

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

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 6

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			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				Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ T

Execute 2/L.2 on cycle 6?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 6

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			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				Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
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					

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

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ To

Execute 2/L.2 on cycle 6?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure ✓

Clock Cycle: 6

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			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				Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 6

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			Wait for L.D
1	S.D F4,0(R1)	2	3			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				Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5				Wait for ADD.D
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 6

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			Wait for L.D
1	S.D F4,0(R1)	2	3			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				Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5				Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 6

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			Wait for L.D
1	S.D F4,0(R1)	2	3			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				Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5				Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 7

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			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				Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5				Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
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					

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				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 7

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			Wait for L.D
1	S.D F4,0(R1)	2	3			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			Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5				Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
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					

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				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 7

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			Wait for L.D
1	S.D F4,0(R1)	2	3			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			Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5				Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
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					

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				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 7

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			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			Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5				Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
2	BNE	R1,R2,Loop	6				Wait for DADDIU
3	L.D	F0,0(R1)	7				
3	ADD.D	F4,F0,F2	7				
3	S.D	F4,0(R1)					
3	DADDIU	R1,R1,#-8					
3	BNE	R1,R2,Loop					

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				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 8

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			Wait for L.D
1	S.D	F4,0(R1)	2	3			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			Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5				Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
2	BNE	R1,R2,Loop	6				Wait for DADDIU
3	L.D	F0,0(R1)	7				
3	ADD.D	F4,F0,F2	7				
3	S.D	F4,0(R1)					
3	DADDIU	R1,R1,#-8					
3	BNE	R1,R2,Loop					

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				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 8

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			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			Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5				Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
2	BNE	R1,R2,Loop	6				Wait for DADDIU
3	L.D	F0,0(R1)	7				
3	ADD.D	F4,F0,F2	7				
3	S.D	F4,0(R1)					
3	DADDIU	R1,R1,#-8					
3	BNE	R1,R2,Loop					

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				1/ADD.D
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Exam asulo

2/L.D access memory on cycle 8 (before 1/S.D)?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 8

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			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			Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5				Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
2	BNE	R1,R2,Loop	6				Wait for DADDIU
3	L.D	F0,0(R1)	7				
3	ADD.D	F4,F0,F2	7				
3	S.D	F4,0(R1)					
3	DADDIU	R1,R1,#-8					
3	BNE	R1,R2,Loop					

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				1/ADD.D
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Exam asulo

2/L.D access memory on cycle 8 (before 1/S.D)?

A: Yes, very sure ✓
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 8

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			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		Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5				Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
2	BNE	R1,R2,Loop	6				Wait for DADDIU
3	L.D	F0,0(R1)	7				
3	ADD.D	F4,F0,F2	7				
3	S.D	F4,0(R1)					
3	DADDIU	R1,R1,#-8					
3	BNE	R1,R2,Loop					

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/L.D	1/ADD.D
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 8

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			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		Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
2	BNE	R1,R2,Loop	6				Wait for DADDIU
3	L.D	F0,0(R1)	7				
3	ADD.D	F4,F0,F2	7				
3	S.D	F4,0(R1)					
3	DADDIU	R1,R1,#-8					
3	BNE	R1,R2,Loop					

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				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 8

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			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		Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				
3	S.D F4,0(R1)					
3	DADDIU R1,R1,#-8					
3	BNE R1,R2,Loop					

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				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 8

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			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		Wait for BNE complete
2	ADD.D	F4,F0,F2	4				Wait for L.D
2	S.D	F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU	R1,R1,#-8	5				Wait for ALU
2	BNE	R1,R2,Loop	6				Wait for DADDIU
3	L.D	F0,0(R1)	7				Wait for BNE complete
3	ADD.D	F4,F0,F2	7				Wait for L.D
3	S.D	F4,0(R1)					
3	DADDIU	R1,R1,#-8					
3	BNE	R1,R2,Loop					

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				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 8

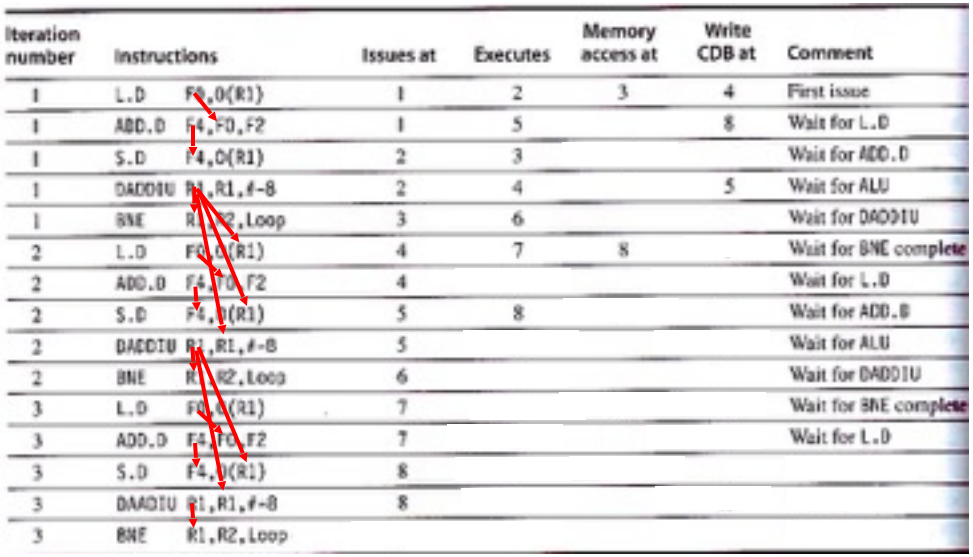
Iteration number	Instructions	Issues at	Executes	Memory access at	Write CDB at	Comment
1	L.D F0,0(R1)	1	2	3	4	First issue
1	ADD.D F4,F0,F2	1	5		8	Wait for L.D
1	S.D F4,0(R1)	2	3			Wait for ADD.D
1	DADDIU R1,R1,#-8	2	4		5	Wait for ALU
1	BNE R1,R2,Loop	3	6			Wait for DADDIU
2	L.D F0,0(R1)	4	7	8		Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				
3	DADDIU R1,R1,#-8	8				
3	BNE R1,R2,Loop					

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				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 9



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			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		Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				
3	DADDIU R1,R1,#-8	8				
3	BNE R1,R2,Loop					

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				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 9

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		Wait for BNE complete
2	ADD.D F4,F0,F2	4				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				
3	DADDIU R1,R1,#-8	8				
3	BNE R1,R2,Loop					

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			1/S.D	
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 9

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				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5				Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				
3	DADDIU R1,R1,#-8	8				
3	BNE R1,R2,Loop					

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			1/S.D	2/L.D
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 9

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				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9			Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				
3	DADDIU R1,R1,#-8	8				
3	BNE R1,R2,Loop					

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				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 9

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				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9			Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				
3	BNE R1,R2,Loop					

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				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 9

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				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9			Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop					

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				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle: 9

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				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9			Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				

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				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:10

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				Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9			Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				

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				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:10

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			Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9			Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				

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		
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:10

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			Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9		10	Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				

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				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:10

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			Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9		10	Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:11

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			Wait for L.D
2	S.D F4,0(R1)	5	8			Wait for ADD.D
2	DADDIU R1,R1,#-8	5	9		10	Wait for ALU
2	BNE R1,R2,Loop	6				Wait for DADDIU
3	L.D F0,0(R1)	7				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:11

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			Wait for L.D
2	S.D F4,0(R1)	5	8			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				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:12

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			Wait for L.D
2	S.D F4,0(R1)	5	8			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				Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
13				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:12

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			Wait for L.D
2	S.D F4,0(R1)	5	8			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			Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:13

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			Wait for L.D
2	S.D F4,0(R1)	5	8			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			Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:13

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			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			Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				2/ADD.D
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:13

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			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		Wait for BNE complete
3	ADD.D F4,F0,F2	7				Wait for L.D
3	S.D F4,0(R1)	8				Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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/L.D	2/ADD.D
14				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:13

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			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		Wait for BNE complete
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				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:14

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			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		Wait for BNE complete
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				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:14

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		Wait for BNE complete
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				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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			2/S.D	
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:14

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				Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8				Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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			2/S.D	3/L.D
15				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:14

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				Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14			Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:15

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				Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14			Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:15

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			Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14			Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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		
16				
17				
18				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:15

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			Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:16

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			Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9				Wait for DADDIU

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				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:16

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			Wait for L.D
3	S.D	F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU	R1,R1,#-8	8	14		15	Wait for ALU
3	BNE	R1,R2,Loop	9	16			Wait for DADDIU

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				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:17

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			Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DADDIU

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				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:18

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			Wait for L.D
3	S.D F4,0(R1)	8	13			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DADDIU

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				
19				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:18

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			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DADDIU

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				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:19

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			Wait for ADD.D
3	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DADDIU

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				
20				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:19

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	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DADDIU

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				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

Example: 2-issue w/ Tomasulo

Clock Cycle:19

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	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DADDIU

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				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

- Fetch Issue 15 instructions/9 cycles = 1.67 IPC

Example: 2-issue w/ Tomasulo

Clock Cycle:19

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	DADDIU R1,R1,#-8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DADDIU

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				

1. Mark relevant dependencies (true deps)
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply scheduling algorithm + assumptions

- Fetch Issue 15 instructions/9 cycles = 1.67 IPC
- Execution Completion rate = 15 inst./16 cycles = 0.94 IPC (much less than 2)

Modified Example (extra Int ALU)

```
Loop:   L.D     F0,0(R1)      ; F0-array element
        ADD.D   F4,F0,F2      ; add scalar in F2
        S.D     F4,0(F1)      ; store result
        DADDIU  R1,R1,#-8     ; decrement pointer 8 bytes (per DW)
        BNE     R1,R2,Loop    ; branch R1 != R2
```

Changed Assumptions:

…

(5) Address Adder & Integer Unit

…

Modified Example, cont'd...

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

- Execution completion rate: 3 cycles less (3/S.D vs 1/L.D: 16-4+1 = 13 vs. 16 cycles before) IPC = 15/13 = 1.15 (vs. 0.94)
- **Performance limited by waiting for branch to resolve.**

Modified Example, cont'd...

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		

- Execution completion rate: 3 cycles less (3/S.D vs 1/L.D: 16-4+1 = 13 vs. 16 cycles before) IPC = 15/13 = 1.15 (vs. 0.94)
- **Performance limited by waiting for branch to resolve.**

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle:

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)						
1	DADDIU R2,R2,#1						
1	SD R2,0(R1)						
1	DADDIU R1,R1,						
1	BNE R2,R3,LOOP						
2	LD R2,0(R1)						
2	DADDIU R2,R2,#1						
2	SD R2,0(R1)						
2	DADDIU R1,R1,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle:

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)						
1	DADDIU R2,R2,#1						
1	SD R2,0(R1)						
1	DADDIU R1,R1,						
1	BNE R2,R3,LOOP						
2	LD R2,0(R1)						
2	DADDIU R2,R2,#1						
2	SD R2,0(R1)						
2	DADDIU R1,R1,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 1

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1					First issue
1	DADDIU R2,R2,#1	1					
1	SD R2,0(R1)						
1	DADDIU R1,R1,						
1	BNE R2,R3,LOOP						
2	LD R2,0(R1)						
2	DADDIU R2,R2,#1						
2	SD R2,0(R1)						
2	DADDIU R1,R1,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 2

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2				First issue
1	DADDIU R2,R2,#1	1					Wait for LW
1	SD R2,0(R1)	2					
1	DADDIU R1,R1,	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,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 3

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3			First issue
1	DADDIU R2,R2,#1	1					Wait for LW
1	SD R2,0(R1)	2	3				
1	DADDIU R1,R1,	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,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 4

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4		First issue
1	DADDIU R2,R2,#1	1					Wait for LW
1	SD R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU R1,R1,	2	3		4		
1	BNE R2,R3,LOOP	3					Wait for DADDIU
2	LD R2,0(R1)	4					
2	DADDIU R2,R2,#1	4					
2	SD R2,0(R1)						
2	DADDIU R1,R1,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 5

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5				Wait for LW
1	SD R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU R1,R1,	2	3		4		
1	BNE R2,R3,LOOP	3					Wait for DADDIU
2	LD R2,0(R1)	4					
2	DADDIU R2,R2,#1	4					
2	SD R2,0(R1)						
2	DADDIU R1,R1,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 5

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5				Wait for LW
1	SD R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU R1,R1,	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,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

Was 1/DADDIU able to start executing on clock cycle 5 only because 1/LD committed on clock cycle 5.

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 5

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5				Wait for LW
1	SD R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU R1,R1,	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,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

Was 1/DADDIU able to start executing on clock cycle 5 only because 1/LD committed on clock cycle 5.

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure ✓

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo

Execute 2/LD on clock cycle 5 before 1/BNE has completed executing?

A: Yes, very sure
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 5

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Comm at clo numb	
1	LD R2,0(R1)	1	2	3	4	5	
1	DADDIU R2,R2,#1	1	5				Wait for LW
1	SD R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU R1,R1,	2	3		4		
1	BNE R2,R3,LOOP	3					Wait for DADDIU
2	LD R2,0(R1)	4					
2	DADDIU R2,R2,#1	4					
2	SD R2,0(R1)						
2	DADDIU R1,R1,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo

Execute 2/LD on clock cycle 5 before 1/BNE has completed executing?

A: Yes, very sure ✓
B: Yes, but not sure
C: Not sure
D: No, but not sure
E: No, very sure

Clock Cycle: 5

Iteration number	Instructions		Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Comm at clo numb	
1	LD	R2,0(R1)	1	2	3	4	5	
1	DADDIU	R2,R2,#1	1	5				Wait for LW
1	SD	R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU	R1,R1,	2	3		4		
1	BNE	R2,R3,LOOP	3					Wait for DADDIU
2	LD	R2,0(R1)	4					
2	DADDIU	R2,R2,#1	4					
2	SD	R2,0(R1)						
2	DADDIU	R1,R1,						
2	BNE	R2,R3,LOOP						
3	LD	R2,0(R1)						
3	DADDIU	R2,R2,#1						
3	SD	R2,0(R1)						
3	DADDIU	R1,R1,						
3	BNE	R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 5

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5				Wait for LW
1	SD R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU R1,R1,	2	3		4		
1	BNE R2,R3,LOOP	3					Wait for DADDIU
2	LD R2,0(R1)	4	5				No execute delay
2	DADDIU R2,R2,#1	4					
2	SD R2,0(R1)						
2	DADDIU R1,R1,						
2	BNE R2,R3,LOOP						
3	LD R2,0(R1)						
3	DADDIU R2,R2,#1						
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 5

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5				Wait for LW
1	SD R2,0(R1)	2	3				Wait for DADDIU
1	DADDIU R1,R1,	2	3		4		
1	BNE R2,R3,LOOP	3					Wait for DADDIU
2	LD R2,0(R1)	4	5				No execute delay
2	DADDIU R2,R2,#1	4					Wait for LW
2	SD R2,0(R1)	5					
2	DADDIU R1,R1,	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,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 7

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	2	3		4		
1	BNE R2,R3,LOOP	3	7				Wait for DADDIU
2	LD R2,0(R1)	4	5	6	7		No execute delay
2	DADDIU R2,R2,#1	4					Wait for LW
2	SD R2,0(R1)	5	6				Wait for DADDIU
2	DADDIU R1,R1,	5	6		7		
2	BNE R2,R3,LOOP	6					Wait for DADDIU
3	LD R2,0(R1)	7					
3	DADDIU R2,R2,#1	7					
3	SD R2,0(R1)						
3	DADDIU R1,R1,						
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 8

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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		No execute delay
2	DADDIU R2,R2,#1	4	8				Wait for LW
2	SD R2,0(R1)	5	6				Wait for DADDIU
2	DADDIU R1,R1,	5	6		7		
2	BNE R2,R3,LOOP	6					Wait for DADDIU
3	LD R2,0(R1)	7	8				
3	DADDIU R2,R2,#1	7					Wait for LW
3	SD R2,0(R1)	8					
3	DADDIU R1,R1,	8					
3	BNE R2,R3,LOOP						

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 9

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9		Wait for LW
2	SD R2,0(R1)	5	6				Wait for DADDIU
2	DADDIU R1,R1,	5	6		7		
2	BNE R2,R3,LOOP	6					Wait for DADDIU
3	LD R2,0(R1)	7	8	9			
3	DADDIU R2,R2,#1	7					Wait for LW
3	SD R2,0(R1)	8	9				
3	DADDIU R1,R1,	8	9				Executes earlier
3	BNE R2,R3,LOOP	9					

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 10

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,	5	6		7		
2	BNE R2,R3,LOOP	6	10				Wait for DADDIU
3	LD R2,0(R1)	7	8	9	10		
3	DADDIU R2,R2,#1	7					Wait for LW
3	SD R2,0(R1)	8	9				Wait for DADDIU
3	DADDIU R1,R1,	8	9		10		Executes earlier
3	BNE R2,R3,LOOP	9					Wait for DADDIU

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 11

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,	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		
3	DADDIU R2,R2,#1	7	11				Wait for LW
3	SD R2,0(R1)	8	9				Wait for DADDIU
3	DADDIU R1,R1,	8	9		10		Executes earlier
3	BNE R2,R3,LOOP	9					Wait for DADDIU

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 12

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,	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		Wait for LW
3	SD R2,0(R1)	8	9				Wait for DADDIU
3	DADDIU R1,R1,	8	9		10		Executes earlier
3	BNE R2,R3,LOOP	9					Wait for DADDIU

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 13

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,	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 LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,	8	9		10		Executes earlier
3	BNE R2,R3,LOOP	9	13				Wait for DADDIU

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle: 14

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,	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 LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle:

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,	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 LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

Example of Multiple Issue, Tomasulo & Reorder Buffer

Clock Cycle:

Iteration number	Instructions	Issues at clock number	Executes at clock number	Read access at clock number	Write CDB at clock number	Commits at clock number	Comment
1	LD R2,0(R1)	1	2	3	4	5	First issue
1	DADDIU R2,R2,#1	1	5		6	7	Wait for LW
1	SD R2,0(R1)	2	3			7	Wait for DADDIU
1	DADDIU R1,R1,	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 execute delay
2	DADDIU R2,R2,#1	4	8		9	10	Wait for LW
2	SD R2,0(R1)	5	6			10	Wait for DADDIU
2	DADDIU R1,R1,	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 LW
3	SD R2,0(R1)	8	9			13	Wait for DADDIU
3	DADDIU R1,R1,	8	9		10	14	Executes earlier
3	BNE R2,R3,LOOP	9	13			14	Wait for DADDIU

1. Mark true dependencies
2. Each clock cycle: Start from oldest instruction, working toward younger instructions and attempt to apply <whichever OoO> algorithm rules

- Commit rate: 15 instructions in 14-5+1 = 10 cycle
- Speculative execution helped use commit 1.5 instructions per cycle

Summary of Slide Set 9

In this slide set, we learned about the following:

- Support for precise exceptions and speculative execution with a single mechanism: The reorder buffer.
- Multiple issue lowers the “ideal CPI” to below 1.
- Two forms of multiple issue: Superscalar, VLIW

We now know how a single processor “core” works. In the next slide set we will talk about software approaches to instruction scheduling, take a look at the Pentium 4 processor, and talk about the limits of instruction level parallelism.

After this we will shift focus to multicore processors. Note that while the current focus is on increasing the number of cores per chip, extracting greater levels of instruction level parallelism will remain important due to Amdahl’s Law.