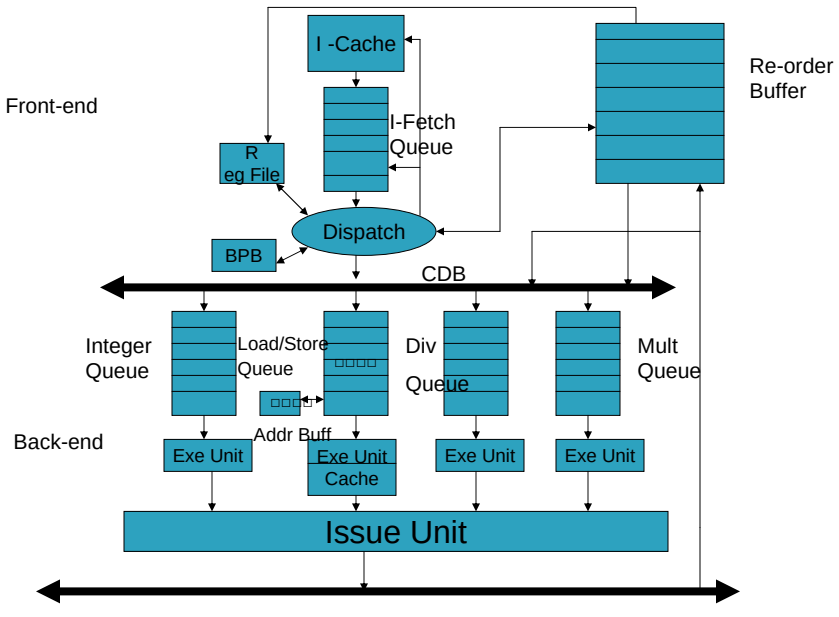
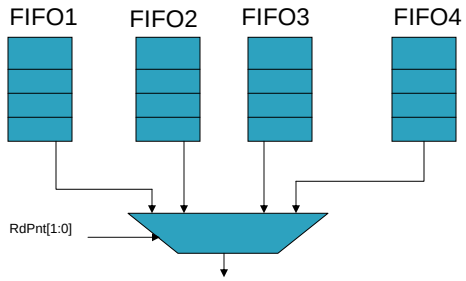
Tomasulo with Speculative Execution

Architecture



Instruction Fetch Queue



- It is arranged as four , 4 location FIFO
- Always an entire block (4 words) is fetch from Cache
- Four Way Interleaved Fetching is done
- IFQ is Flushed every time a Jump Instruction comes in Dispatch or a Branch is taken (successful branch).

General Operation

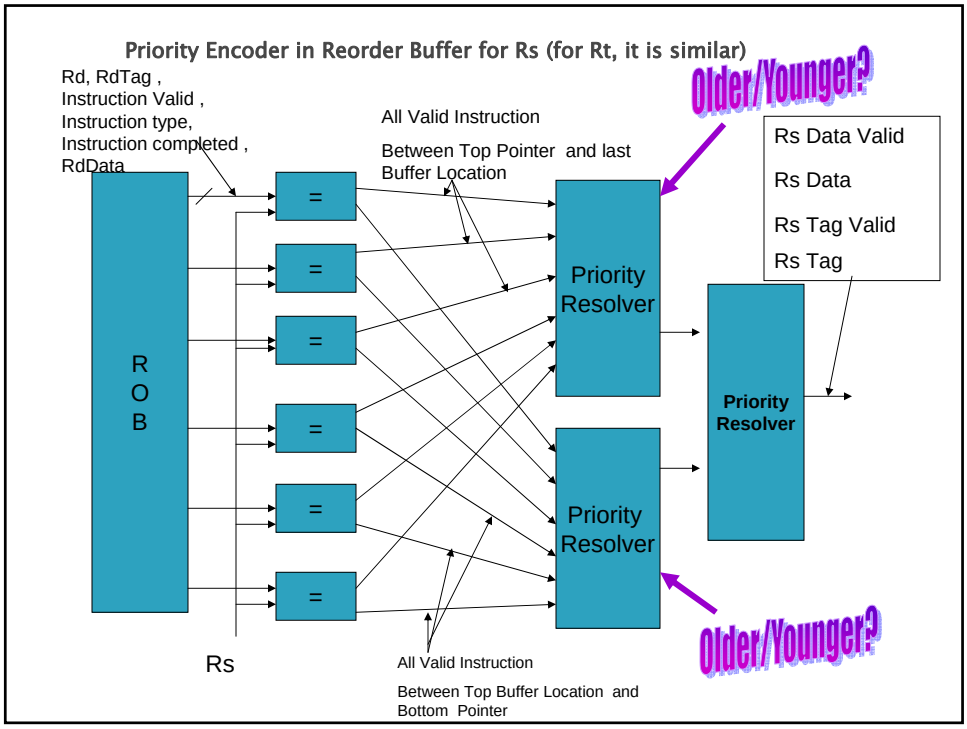
Dispatching of Instruction

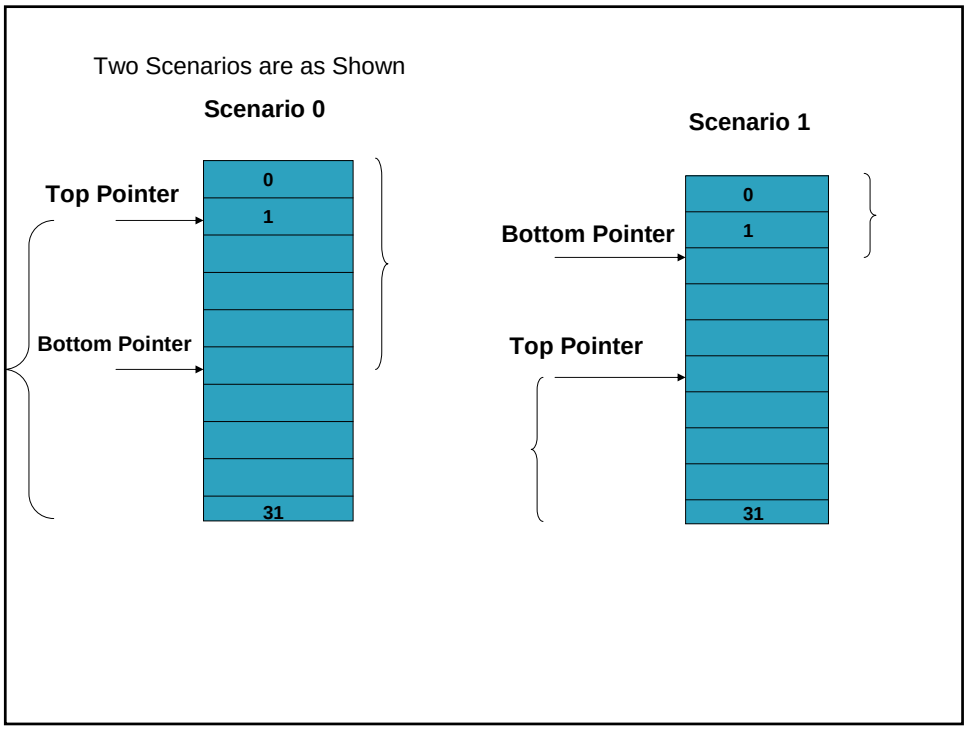
- At a given time an Instruction can be
  - Ø Incomplete - Pending in the Backend
  - Ø Complete - Instruction is in ROB but not Committed
  - Ø Committed

- The instruction is given a ROB slot regardless of it type , except for a jump or an invalid instruction

- For register reading instructions, “Rs” or “Rs and Rt” Data is/are searched in ROB .
  In case a match is found Corresponding Data is forwarded else Data from Register is forwarded. Corresponding Data means “ROB Tag if its pending in backend or “ROB Data if its complete”.
- If there is no match, then data from register file is taken.

- Ø Since more than one instruction in ROB can have the same destination register ID, one needs a Priority Resolver.


Priority Encoder in Reorder Buffer for Rs (for Rt, it is similar)
Rd, RdTag ,
Instruction Valid ,
Instruction type,
Instruction completed ,
RdData
All Valid Instruction
Between Top Pointer and last
Buffer Location
Older/Younger?
Rs Data Valid
Rs Data
Rs Tag Valid
Rs Tag
R
O
B
=
=
=
=
=
=
Priority
Resolver
Priority
Resolver
Priority
Resolver
Older/Younger?
Rs
All Valid Instruction
Between Top Buffer Location and
Bottom Pointer


Two Scenarios are as Shown
Scenario 0
Top Pointer
Bottom Pointer
0
1
31
Scenario 1
Bottom Pointer
Top Pointer
0
1
31

$2 is needed by the dispatch. How would you (the ROB) respond?

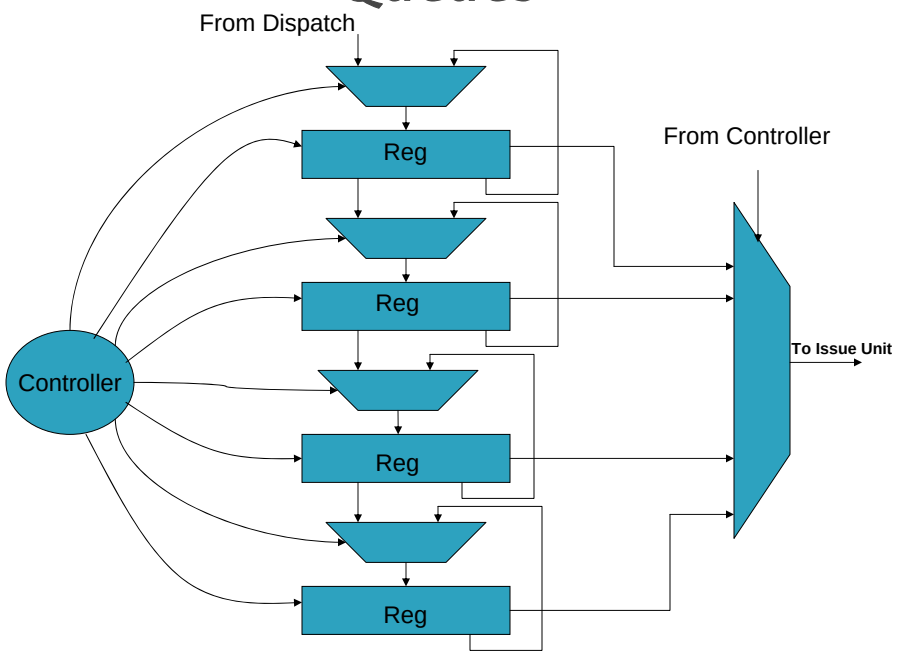
Scenario 0

Address	VALID	RD_ID	Reg_Write
0	0	0	1
	0	$2	1
	0	$0	0
Top (RP)	1	$1	1
	1	$2	1
	1	$15	1
	1	$2	1
	1	$12	1
	1	$2	0
	1	$7	0
Bottom (WP)	0	$13	1
	0	$0	1
	0	$4	0
	0	$2	1
31	0	$0	1

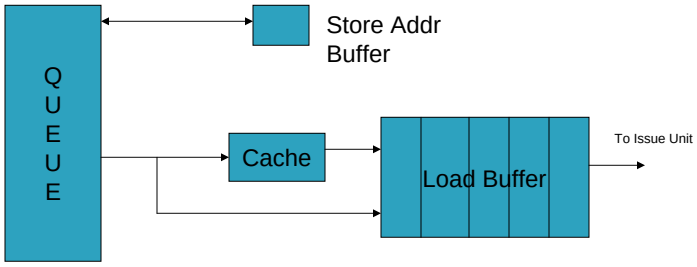
Scenario 1

Address	VALID	RD_ID	Reg_Write
0	1	$0	1
	1	$2	1
	1	$10	1
Bottom (WP)	0	$1	0
	0	$21	1
	0	$12	1
	0	$2	0
	0	$15	1
	0	$22	1
Top (RP)	1	$7	1
	1	$13	0
	1	$2	1
	1	$1	1
	1	$2	0
31	1	$3	1

Queues

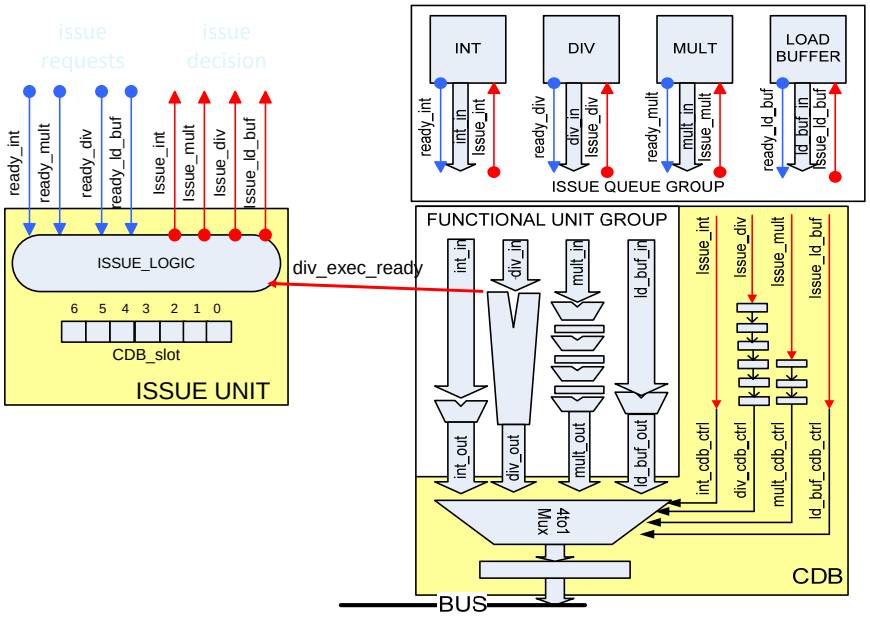


Load Store Queue



- Load Word accesses the memory and result is written into the Load/Store Buffer
- Store Word Does not access memory while getting issued from Queue and goes to Load/Store Buffer directly
- Whenever Store Word is issued Store Address Buffer is updated with Store Word Memory address and ROB Tag
- Once a Store Word is committed from the Top of ROB corresponding entry in Store Address Buffer is invalidated .

Issue Unit



Branch Instruction Handling

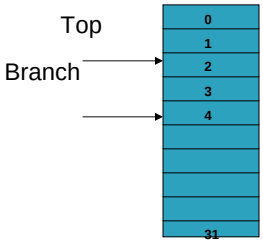
- Once a Branch Instruction arrives in Dispatch , Branch Prediction is done using BPB (Branch Prediction Buffer).
- Last 3 Bits of PC are used by the Branch Predictor
- Branches are handled aggressively. They are executed as soon as they arrive on CDB without waiting for branch Instruction to become the top instruction in ROB.
- Selective flushing mechanism is used to flush the instructions in backend in case of a mispredicted branch instruction.

Flushing Mechanism for Branch Instruction

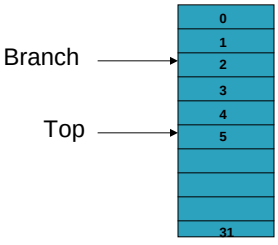
- In order to flush instruction in backend a Flush signal along with the following are conveyed to the backend.

1. Current Top Location of ROB
2. Depth of the Branch Instruction
3. All instructions in the backend (as well as in ROB) with depth greater than the successful branches depth need to leave (need to be flushed

Consider Two Scenarios shown below



Depth = 2 (= 4-2)



Depth = 29 (= 2-5 mod 32)

Exception Handling

- Exception once Generated are carried through Queues
- Once a Instruction with Exception is announced on CDB Program Jumps to a pre-specified Address
- Since Program Jumps to a Address flushing needs to be done which is similar to a branch Instruction or a jump instruction handling
- Since Store word Instructions are executed once they are at the top of buffer exception handling for them are also done once the cache (virtual memory) returns an exception (such as page fault).