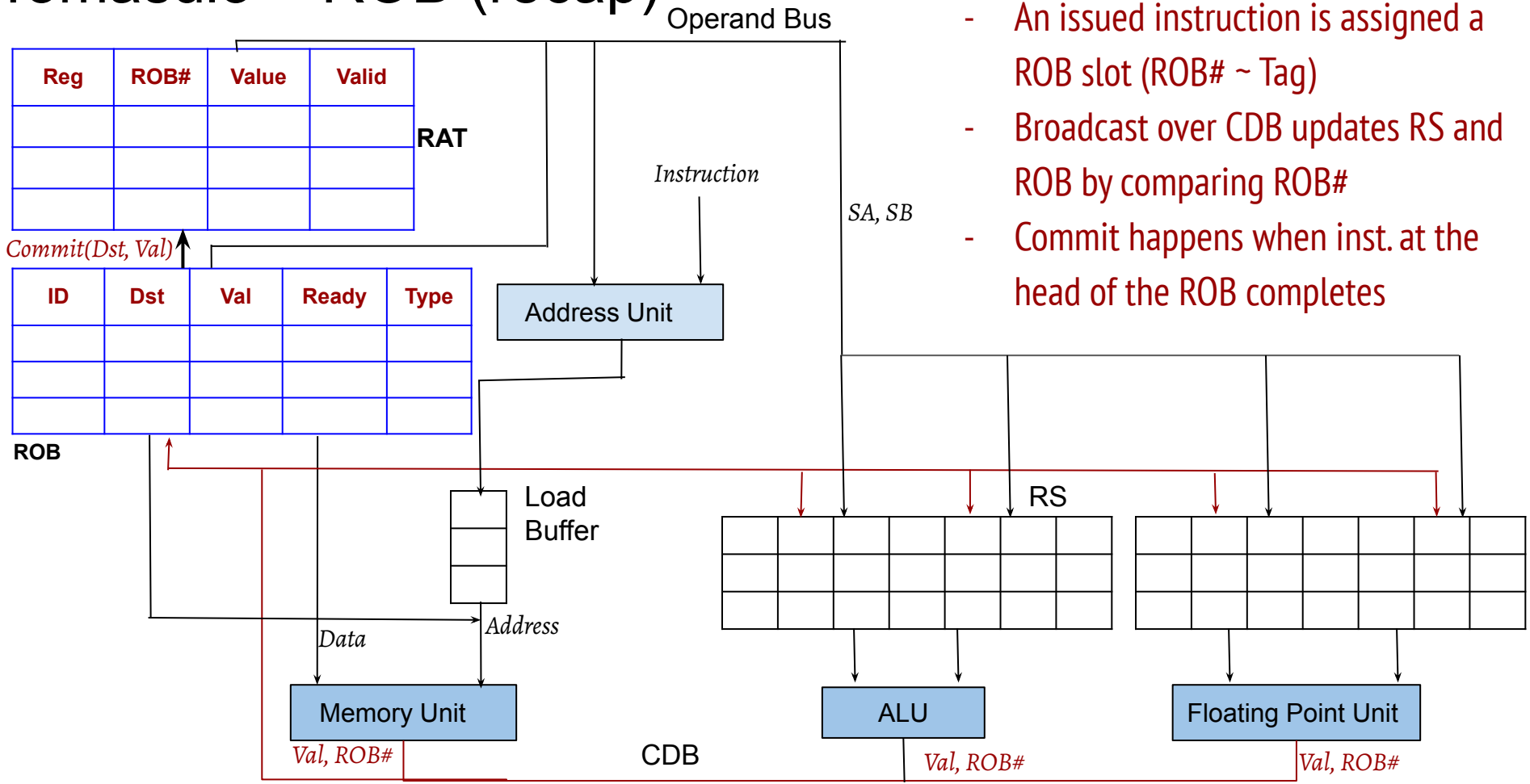
Computer Architecture

Superscalar Processors

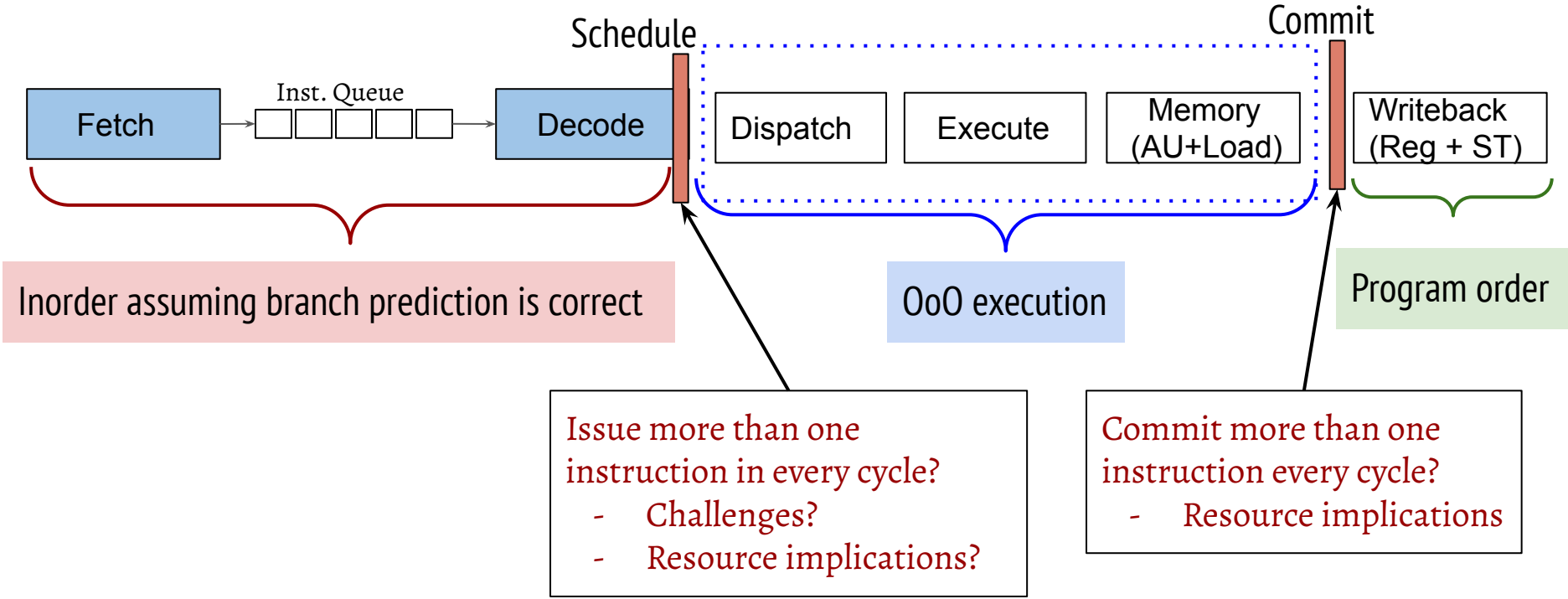
Debadatta Mishra, CSE, IITK

Tomasulo + ROB (recap)

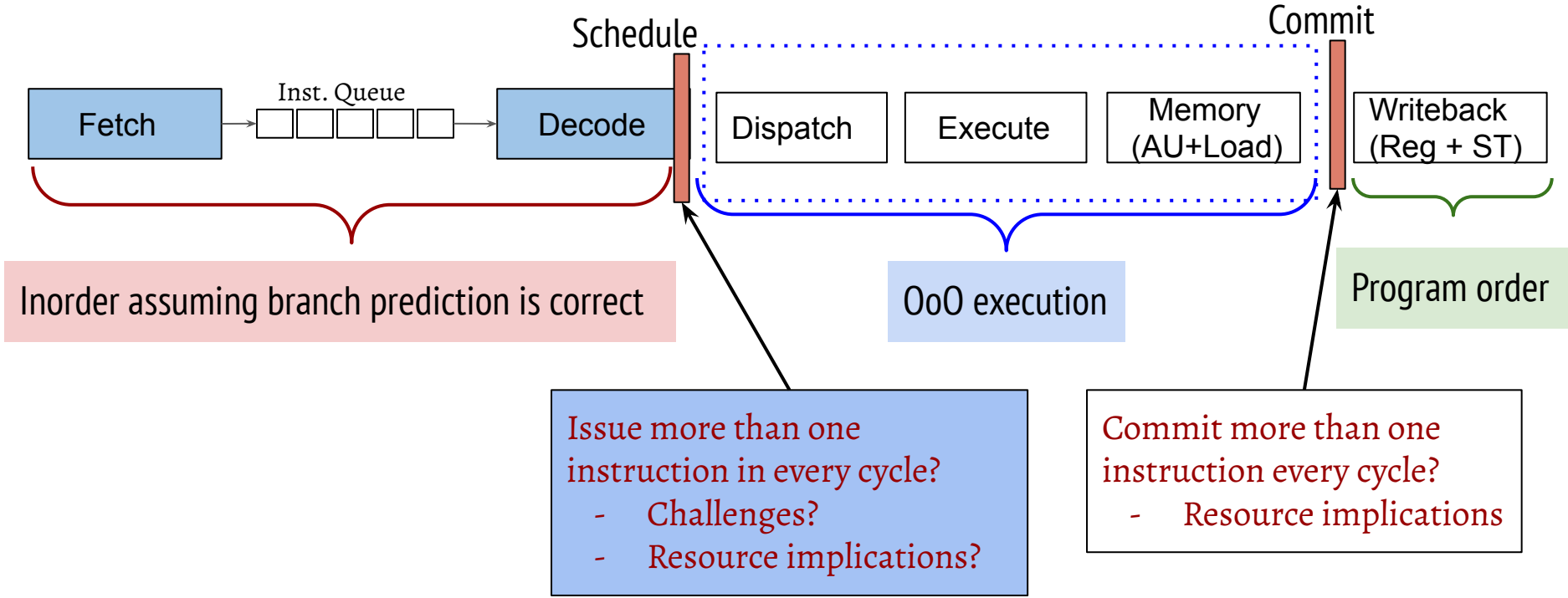


- An issued instruction is assigned a ROB slot (ROB# ~ Tag)
- Broadcast over CDB updates RS and ROB by comparing ROB#
- Commit happens when inst. at the head of the ROB completes

Speculative execution with multi-issue

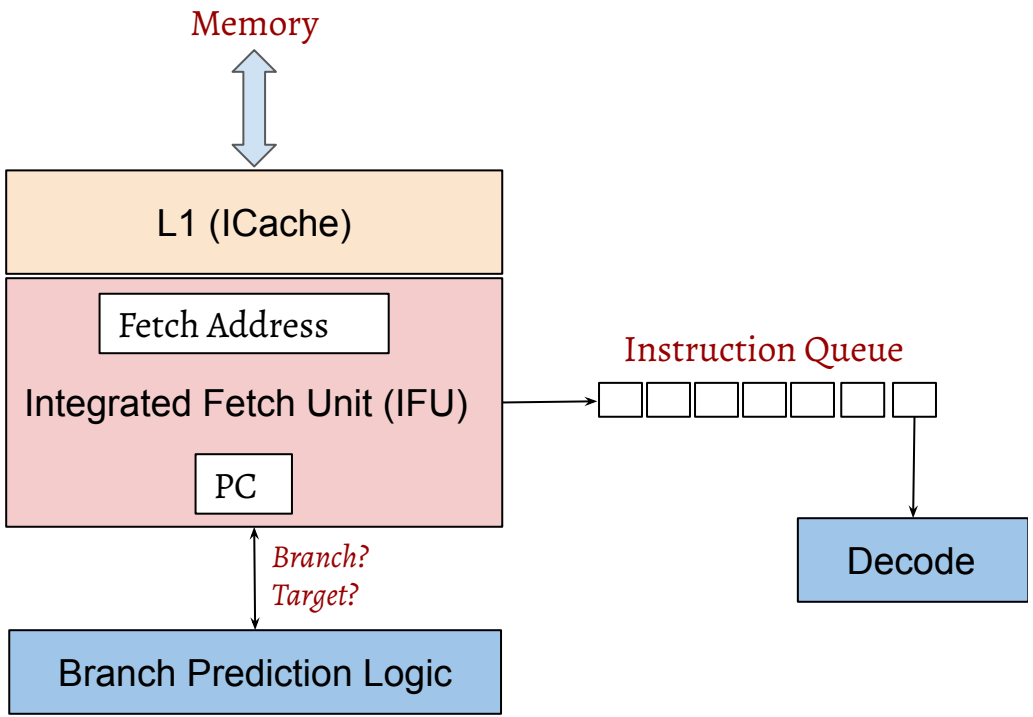


Speculative execution with multi-issue



- Instruction fetch should not become a bottleneck, increased fetch width + branch aware inst. fetch
- Complex issue logic to accomplish multiple instruction issue in the same cycle
- Issue independent instructions (catch: program execution need to progress i.e., in-order commit)

Integrated fetch unit



- Instruction fetch can be a multi-cycle operation
- Fetch address is determined by looking up the current PC in the BTB
- If the PC is present in the BTB and branch is predicted (taken), IF address becomes the predicted PC value
- IFU can be part of the L1 inst. cache
- Additional techniques like instruction prefetching can be incorporated so that IF does not become a bottleneck

Multiple issue challenges

- Issuing multiple instructions in one clock cycle
- Checking dependencies across instructions in the same batch of issue

Multiple issue challenges

- Issuing multiple instructions in one clock cycle
- Possible solutions
  - Performing all operations to issue more than one instruction in one cycle is challenging
  - Widening the issue logic can address this to a certain extent
  - Pipelining the issue logic (more than one stage)
- Checking dependencies across instructions in the same batch of issue

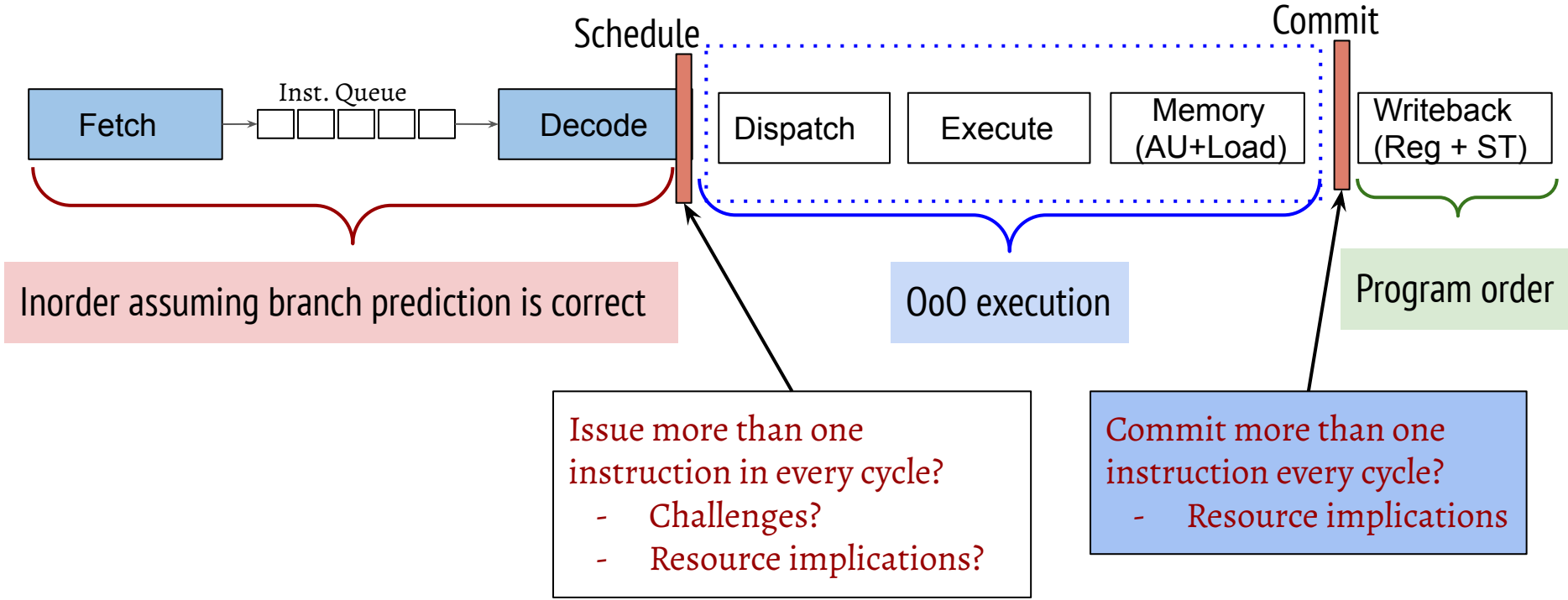
Multiple issue challenges

- Issuing multiple instructions in one clock cycle
- Possible solutions
  - Performing all operations to issue more than one instruction in one cycle is challenging
  - Widening the issue logic can address this to a certain extent
  - Pipelining the issue logic (more than one stage)
- Checking dependencies across instructions in the same batch of issue
- Possible solution (1)
  - Assign resources (ROB and RS) to instructions in the issue bundle (till possible)
  - Check and update the RS operands to the ROB entry for each dependent instruction

Multiple issue challenges

- Issuing multiple instructions in one clock cycle
- Possible solutions
  - Performing all operations to issue more than one instruction in one cycle is challenging
  - Widening the issue logic can address this to a certain extent
  - Pipelining the issue logic (more than one stage)
- Checking dependencies across instructions in the same batch of issue
- Possible solution (1)
  - Assign resources (ROB and RS) to instructions in the issue bundle (till possible)
  - Check and update the RS operands to the ROB entry for each dependent instruction
- Possible solution (2)
  - Use a dispatch queue to hold decoded instructions (ROB booked)
  - Issue instructions based on dependency and FU availability

Speculative execution with multi-issue



- More write ports are required in the physical register file

Example

```
loop:   lw r2,0(r1)
        addiu r2,r2,#1
        sw r2,0(r1)
        addiu r1,r1,#8
        bne r2,r3,loop
```

- Simple program to increment an array (r1 = base, r3=end)
- Assume each FU consumes one cycle
- What is the execution behavior with a speculative superscalar processor with issue width = 2 and commit width = 2

Example

Iteration	Instruction	Issue	Execution	Memory	WriteRes	Commit
1	`lw r2,0(r1)`	1	2	3	4	5
1	`addiu r2,r2,#1`	1	5		6	7

Wait for lw to write result

Example

Iteration	Instruction	Issue	Execution	Memory	WriteRes	Commit
1	`lw r2,0(r1)`	1	2	3	4	5
1	`addiu r2,r2,#1`	1	5		6	7
1	`sw r2,0(r1)`	2	3			7
1	`addiu r1,r1,#8`	2	3		4	8

Wait for lw to write result

Wait for r2 (addiu)

Can commit after sw

Example

Iteration	Instruction	Issue	Execution	Memory	WriteRes	Commit	
1	`lw r2,0(r1)`	1	2	3	4	5	
1	`addiu r2,r2,#1`	1	5		6	7	*Wait for lw to write result*
1	`sw r2,0(r1)`	2	3			7	*Wait for r2 (addiu)*
1	`addiu r1,r1,#8`	2	3		4	8	*Can commit after sw*
1	`bne r2,r3,loop`	3	7			8	*Issue a single branch*

Example

Iteration	Instruction	Issue	Execution	Memory	WriteRes	Commit	
1	`lw r2,0(r1)`	1	2	3	4	5	
1	`addiu r2,r2,#1`	1	5		6	7	*Wait for lw to write result*
1	`sw r2,0(r1)`	2	3			7	*Wait for r2 (addiu)*
1	`addiu r1,r1,#8`	2	3		4	8	*Can commit after sw*
1	`bne r2,r3,loop`	3	7			8	*Issue a single branch*
2	`lw r2,0(r1)`	4	5	6	7	9	*CC5, r1 available*
2	`addiu r2,r2,#1`	4	8		9	10	*Wait for lw to write result*

Example

Iteration	Instruction	Issue	Execution	Memory	WriteRes	Commit	
1	`lw r2,0(r1)`	1	2	3	4	5	
1	`addiu r2,r2,#1`	1	5		6	7	*Wait for lw to write result*
1	`sw r2,0(r1)`	2	3			7	*Wait for r2 (addiu)*
1	`addiu r1,r1,#8`	2	3		4	8	*Can commit after sw*
1	`bne r2,r3,loop`	3	7			8	*Issue a single branch*
2	`lw r2,0(r1)`	4	5	6	7	9	*CC5, r1 available*
2	`addiu r2,r2,#1`	4	8		9	10	*Wait for lw to write result*
2	`sw r2,0(r1)`	5	6			10	*Wait for r2 (addiu)*
3	`addiu r1,r1,#8`	5	6		7	11	*Can commit after sw*

The Microarchitecture of the Pentium 4 Processor, Intel Technology Journal Q1, 2001

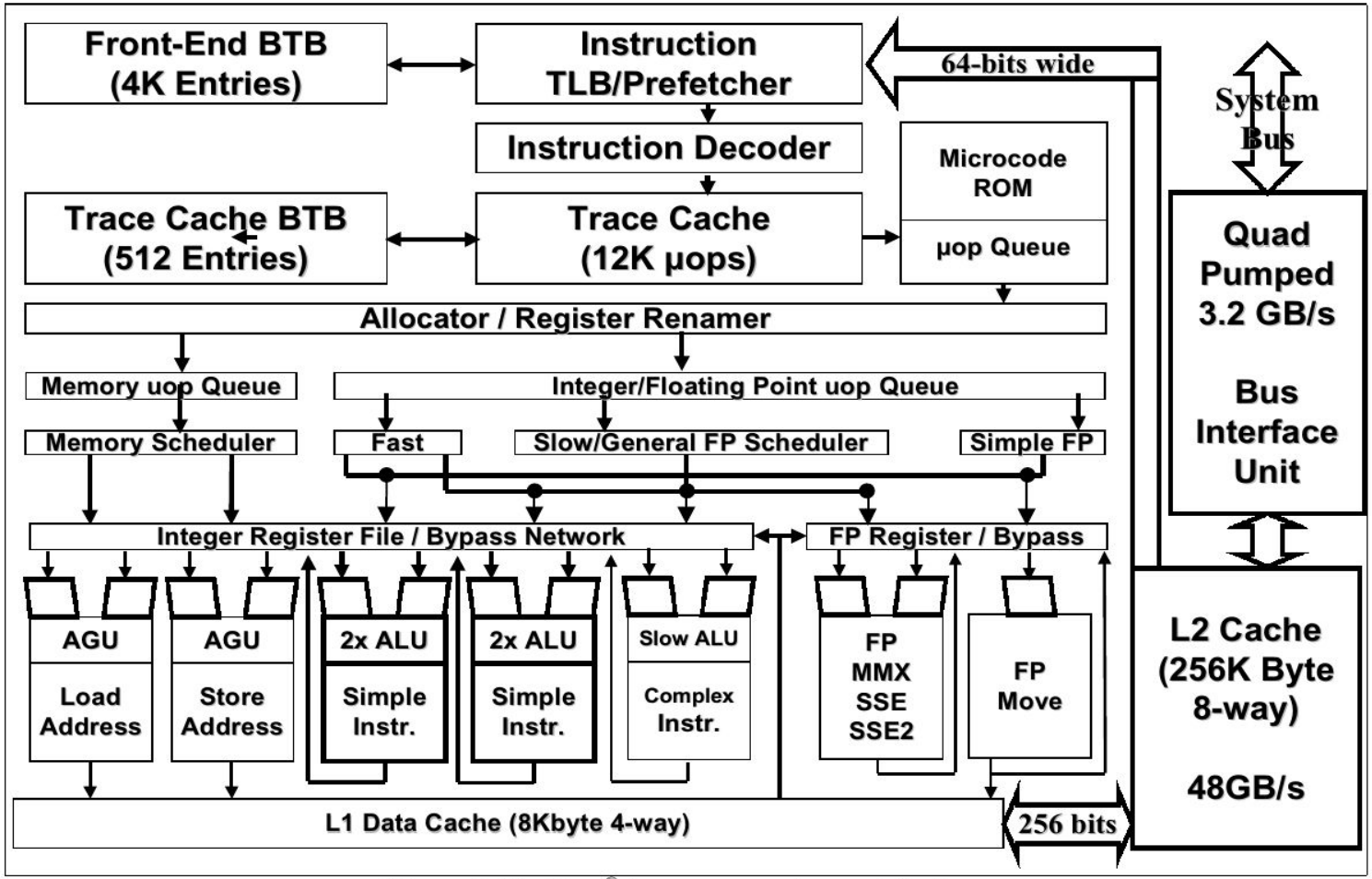


Figure 4: Pentium® 4 processor microarchitecture