Announcements/Reminders

- Progress report due: Tuesday, May 23
- Final Project due: Friday, June 2

- Classes this week:
  – today (only)
- Classes next week:
  – Wednesday
  – Friday

Computer Science 703
Advance Computer Architecture
2006 Semester 1
Lecture Notes
17May06
Executing Out-of-order

James Goodman

Department of Computer Science

Sources for this lecture

R.M. Tomasulo, "An efficient algorithm for exploiting multiple arithmetic units," *IBM Journal*, January 1967, pp. 25-33.

Hennessy/Patterson, Section 3.2-3.3, pp. 181-196.

Out-of-Order (OoO) Execution

- Dataflow: each instruction (procedure, method, code segment) depends on certain operands.
  – These operands have been assigned values sometime previously.
  – If they have been assigned, the instruction can be "issued"
- Instructions can be issued even if previous instructions have not been issued.
- Instructions are retired (committed) in-order
  – Necessary for handling exceptions
  – Memory ordering is easier
- Registers are loaded from main memory, or stored as output of instructions
  – If a register value is needed before it is available, this is a hazard.

Simple OoO Scheme: Busy Bits

How do we know if a register is available?

- Associate "busy bit" for each register
  - Set bit when issuing instruction with register as target
  - Clear bit when value is entered into register
  - Issue instruction only if busy bit is not set
  - Instruction can only be issued if source registers are not busy (available)

Handling Hazards

- 3 kinds of hazards:
  - RAW (True dependence)
    - Handled by busy bits
  - WAR (Anti-dependence)
    - Not handled by busy bits
  - WAW (Output dependence)
    - Not handled by busy bits
    - Why would this happen?

Handling WAR & WAW Hazards

- What if instruction has been issued with target register 4 and another instruction wants to target register 4?
  - May be pending source register instructions (WAR) or not (WAW)
  - Recognize that the register actually has two values assigned to it!
- Single assignment concept: a "virtual register" is assigned a value once
  - The virtual register is "live" until it has been read for the last time
  - Last time can be detected only when register is written
- Rename registers (reassigning registers through a level of indirection):
  - Each time register is targeted, if outstanding writes, assign a new register and remap the registers
  - Issue logic must remember which register 4 to use (trivial if in-order)

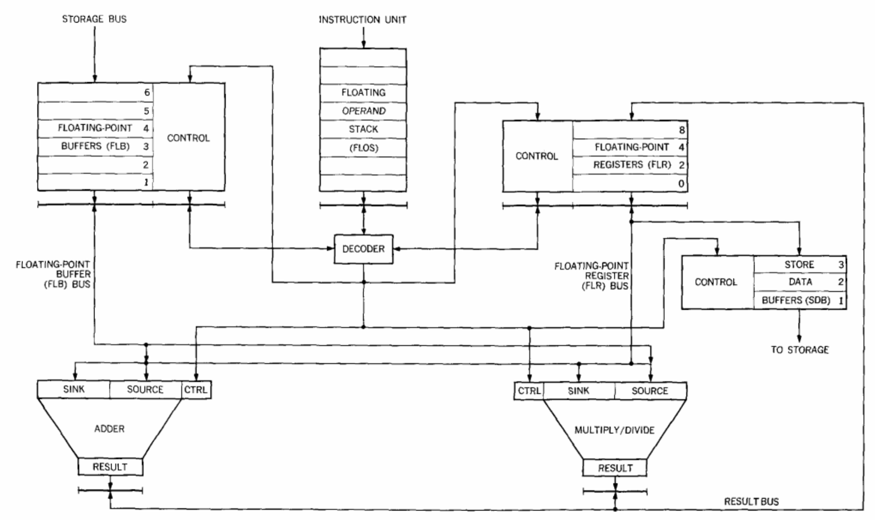

Figure 1 Data registers and transfer paths without CDB.

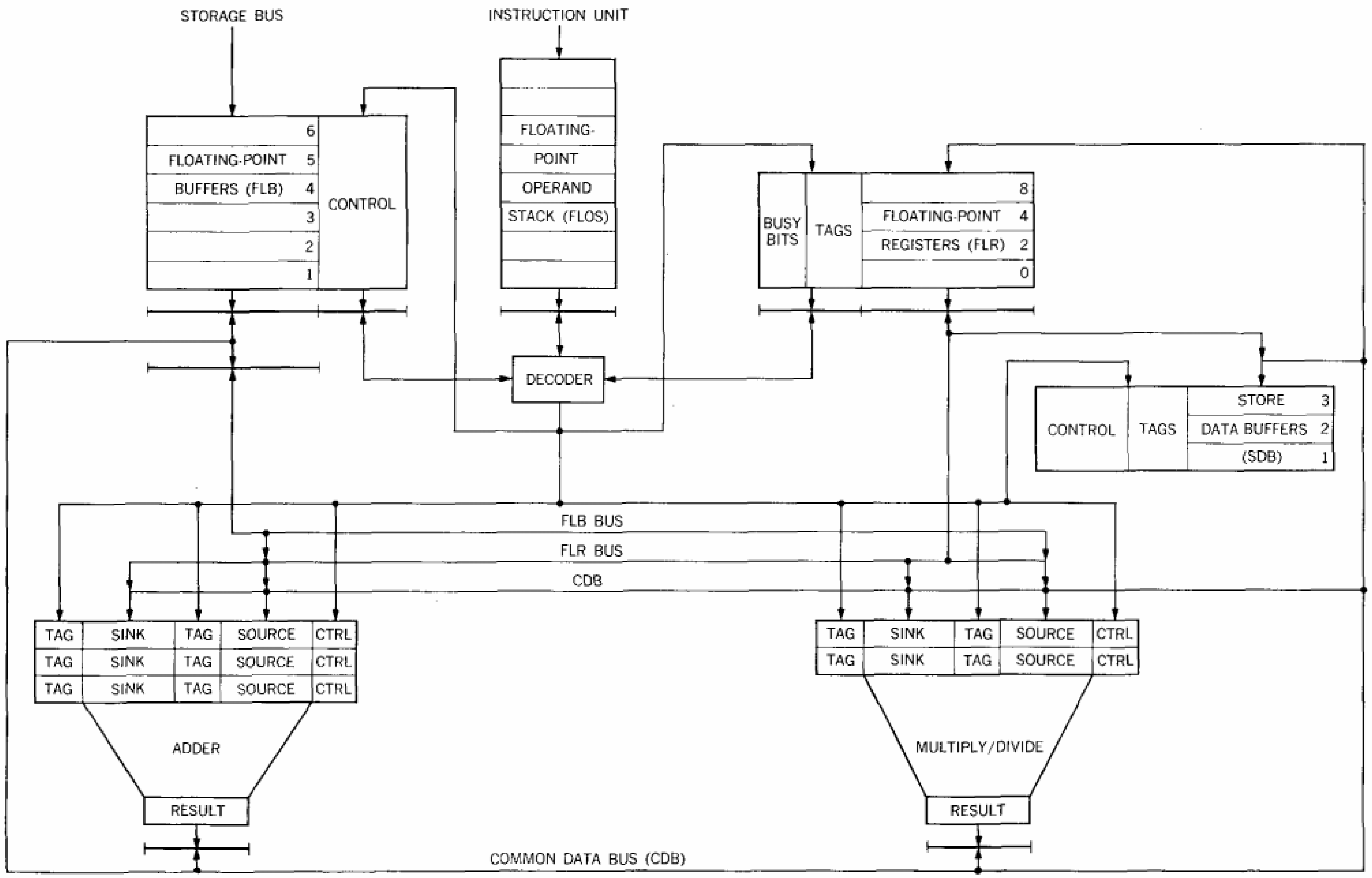

Figure 4 Data registers and transfer paths, including CDB and reservation stations.