CS252 Graduate Computer Architecture
Fall 2015
Lecture 5: Out-of-Order Processing

Krste Asanovic
krste@berkeley.edu
http://inst.eecs.berkeley.edu/~cs252/fa15

Last Time in Lecture 4

- Iron Law of processor performance
- Pipelining: reduce cycle time, try to keep CPI low
- Hazards:
  - Structural hazards: interlock or more hardware
  - Data hazards: interlocks, bypass, speculate
  - Control hazards: interlock, speculate
- Precise traps/interrupts for in-order pipeline

IBM 7030 "Stretch" (1954-1961)

- Original goal was to use new transistor technology to give 100x performance of tube-based IBM 704.
- Design based around 4 stages of "lookahead" pipelining
- More than just pipelining, a simple form of decoupled execution with indexing and branch operations performed speculatively ahead of data operations
- Also had a simple store buffer
- Very complex design for the time, difficult to explain to users performance of pipelined machine
- When finally delivered, was benchmarked at only 30x 704 and embarrassed IBM, causing withdrawal after initial deliveries

Simple vector-vector add code example

```
#     for(i=0; i<N; i++)
#           A[i]=B[i]+C[i];

loop: fld f0, 0(x2) // x2 points to B
      fld f1, 0(x3) // x3 points to C
      fadd.d f2, f0, f1
      fsd f2, 0(x1) // x1 points to A
      add x1, 8     // Bump pointer
      add x2, 8     // Bump pointer
      add x3, 8     // Bump pointer
      bne x1, x4, loop // x4 holds end
```

Simple Pipeline Scheduling

Can reschedule code to try to reduce pipeline hazards

```
loop: fld f0, 0(x2) // x2 points to B
      fld f1, 0(x3) // x3 points to C
      add x3, 8     // Bump pointer
      add x2, 8     // Bump pointer
      fadd.d f2, f0, f1
      add x1, 8     // Bump pointer
      fsd f2, -8(x1) // x1 points to A
      bne x1, x4, loop // x4 holds end
```

Long latency loads and floating-point operations limit parallelism within a single loop iteration

Loop Unrolling

Can unroll to expose more parallelism

```
loop:  fld f0, 0(x2) // x2 points to B
       fld f1, 0(x3) // x3 points to C
       fld f10, 8(x2)
       fld f11, 8(x3)
       add x3, 16 // Bump pointer
       add x2, 16 // Bump pointer
       fadd.d f2, f0, f1
       fadd.d f12, f10, f11
       add x1, 16 // Bump pointer
       fsd f2, -16(x1) // x1 points to A
       fsd f12, -8(x1)
      bne x1, x4, loop // x4 holds end
```

- Unrolling limited by number of architectural registers
- Unrolling increases instruction cache footprint
- More complex code generation for compiler, has to understand pointers
- Can also software pipeline, but has similar concerns

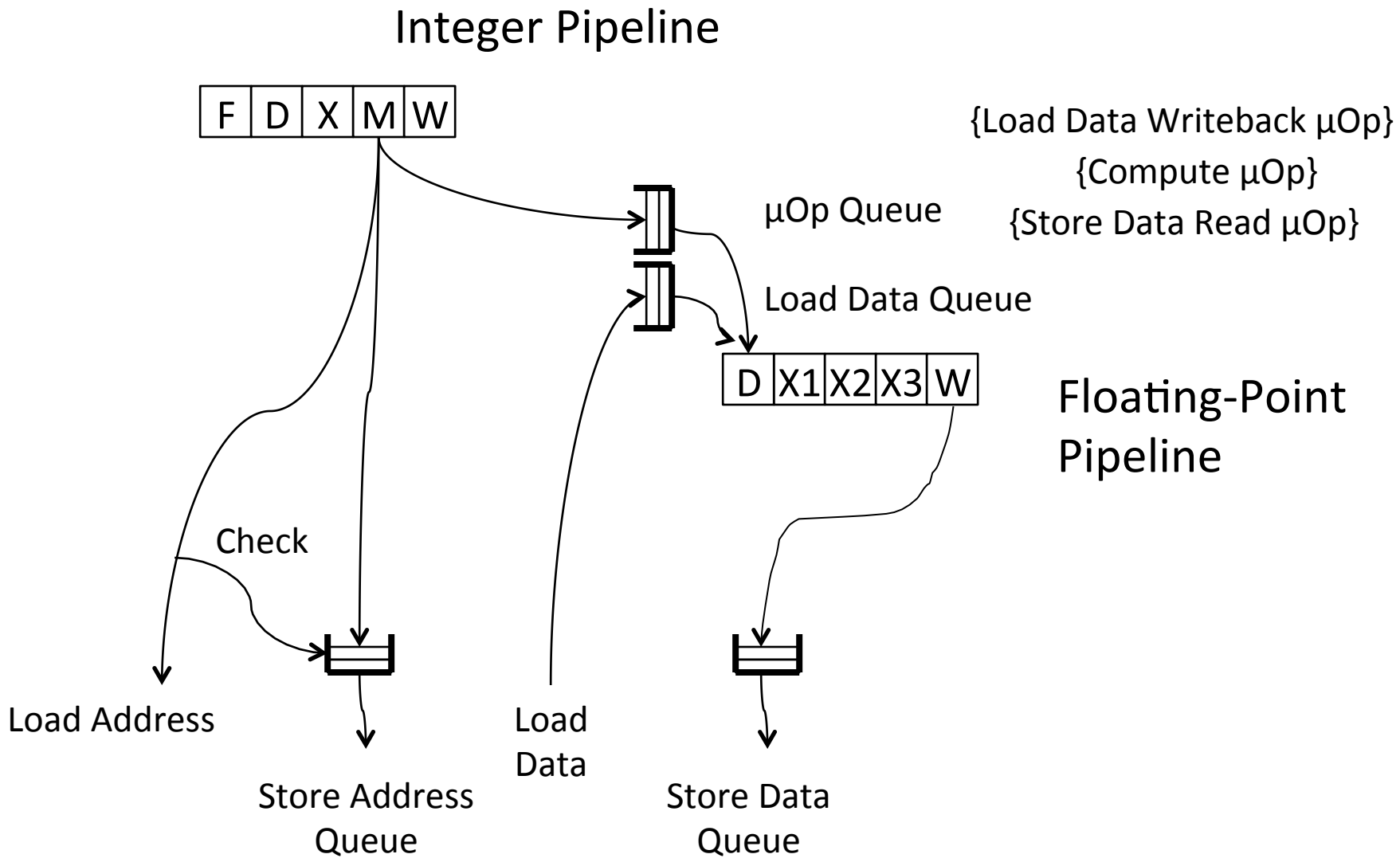
Decoupling *(lookahead, runahead)* in µarchitecture

Can separate control and memory address operations from data computations:

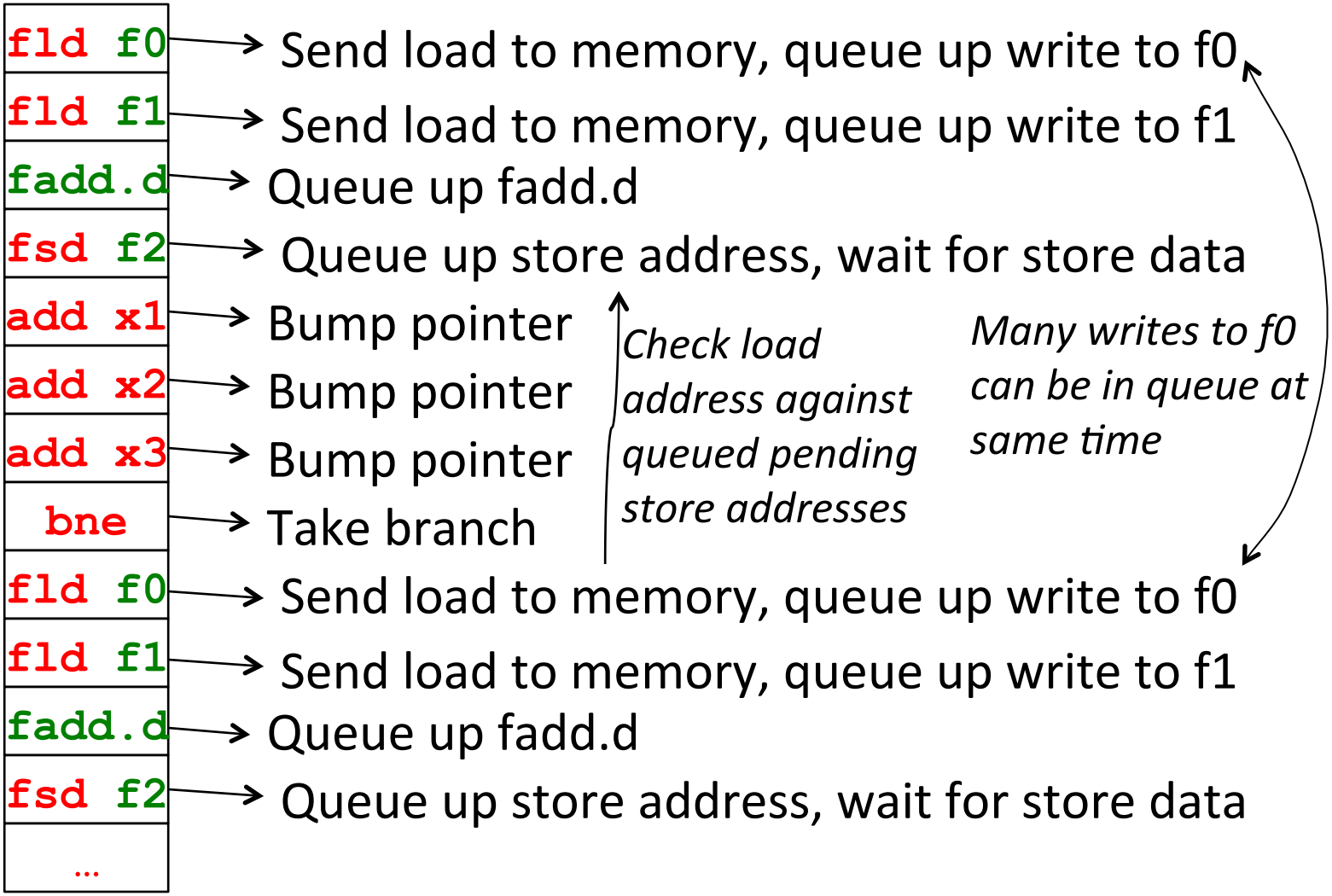
```
loop: fld f0, 0(x2) // x2 points to B
      fld f1, 0(x3) // x3 points to C
      fadd.d f2, f0, f1
      fsd f2, 0(x1) // x1 points to A
      add x1, 8     // Bump pointer
      add x2, 8     // Bump pointer
      add x3, 8     // Bump pointer
      bne x1, x4, loop // x4 holds end
```

The control and address operations do not depend on the data computations, so can be computed early relative to the data computations, which can be delayed until later.

Simple Decoupled Machine



Decoupled Execution



Supercomputers

Definitions of a supercomputer:

- Fastest machine in world at given task
- A device to turn a compute-bound problem into an I/O bound problem
- Any machine costing $30M+
- Any machine designed by Seymour Cray

- CDC6600 (Cray, 1964) regarded as first supercomputer

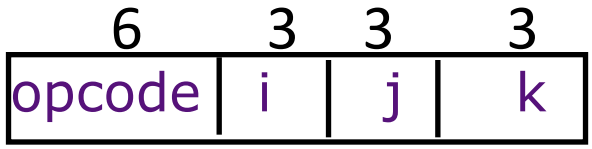
CDC 6600 *Seymour Cray, 1963*

- A fast pipelined machine with 60-bit words
  - 128 Kword main memory capacity, 32 banks
- Ten functional units (parallel, unpipelined)
  - Floating Point: adder, 2 multipliers, divider
  - Integer: adder, 2 incrementers, ...
- Hardwired control (no microcoding)
- *Scoreboard* for dynamic scheduling of instructions
- Ten Peripheral Processors for Input/Output
  - a fast multi-threaded 12-bit integer ALU
- Very fast clock, 10 MHz (FP add in 4 clocks)
- >400,000 transistors, 750 sq. ft., 5 tons, 150 kW, novel freon-based technology for cooling
- Fastest machine in world for 5 years (until 7600)
  - over 100 sold ($7-10M each)

CDC 6600:
A Load/Store Architecture

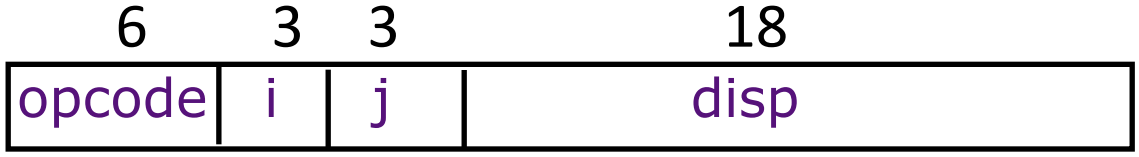
- Separate instructions to manipulate three types of reg.
  - 8x60-bit data registers (X)
  - 8x18-bit address registers (A)
  - 8x18-bit index registers (B)

- All arithmetic and logic instructions are register-to-register



Ri ← Rj op Rk

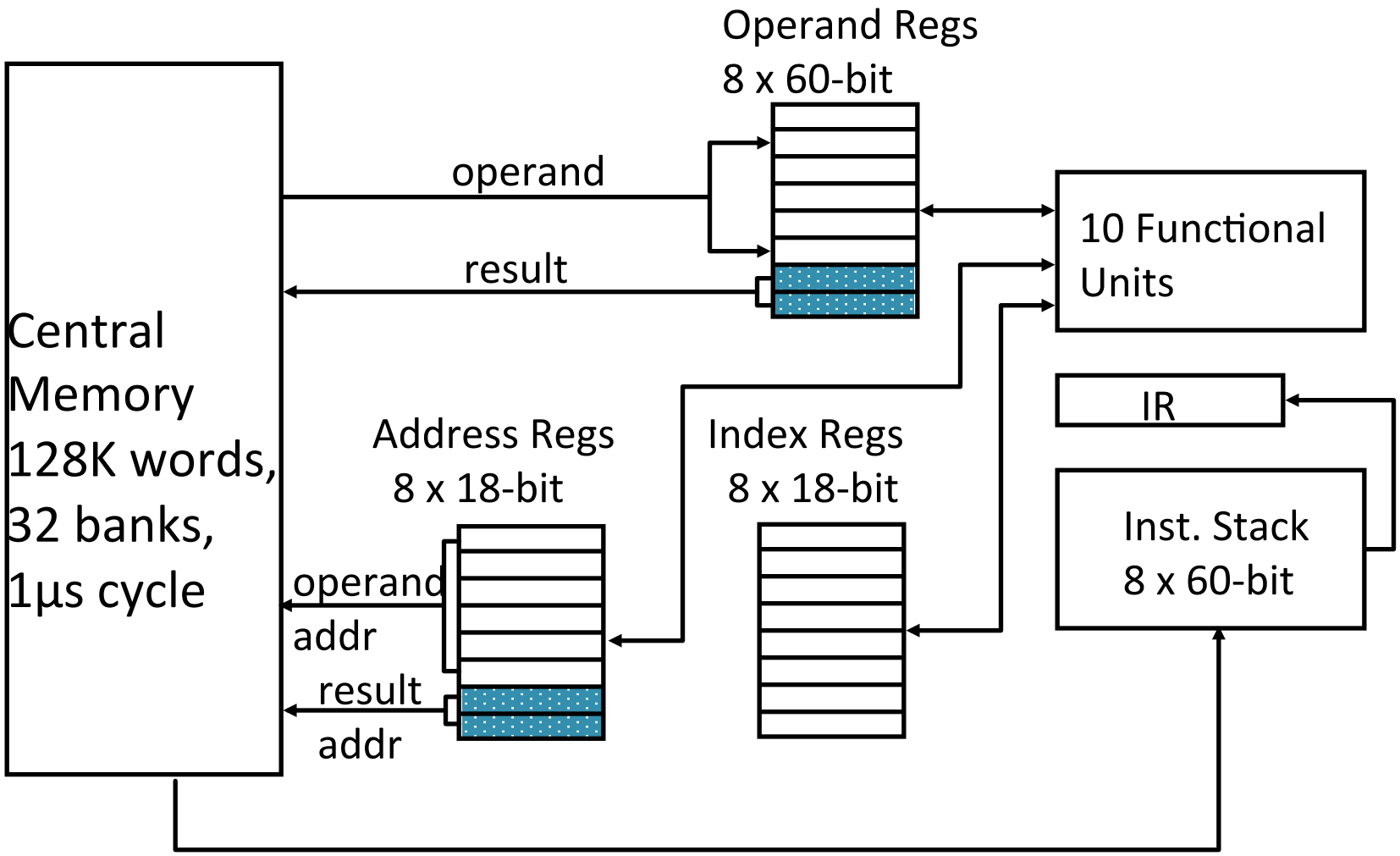
- Only Load and Store instructions refer to memory!



Ri ← M[Rj + disp]

Touching address registers 1 to 5 initiates a load
6 to 7 initiates a store
- very useful for vector operations

CDC 6600: Datapath



CDC6600 ISA designed to simplify high-performance implementation

- Use of three-address, register-register ALU instructions simplifies pipelined implementation
  - Only 3-bit register specifier fields checked for dependencies
  - No implicit dependencies between inputs and outputs
- Decoupling setting of address register (Ar) from retrieving value from data register (Xr) simplifies providing multiple outstanding memory accesses
  - Software can schedule load of address register before use of value
  - Can interleave independent instructions inbetween
- CDC6600 has multiple parallel but unpipelined functional units
  - E.g., 2 separate multipliers
- Follow-on machine CDC7600 used pipelined functional units
  - Foreshadows later RISC designs

CDC6600: Vector Addition

```
        B0 ← - n
loop:   JZE  B0, exit
        A0 ← B0 + a0        load X0
        A1 ← B0 + b0        load X1
        X6 ← X0 + X1
        A6 ← B0 + c0        store X6
        B0 ← B0 + 1
        jump loop
```

Ai = address register
Bi = index register
Xi = data register

CDC6600 Scoreboard

- Instructions dispatched in-order to functional units provided no structural hazard or WAW
  - Stall on structural hazard, no functional units available
  - Only one pending write to any register
- Instructions wait for input operands (RAW hazards) before execution
  - Can execute out-of-order
- Instructions wait for output register to be read by preceding instructions (WAR)
  - Result held in functional unit until register free

MEMORANDUM

August 28, 1963

Memorandum To: Messrs. A. L. Williams
T. V. Learson
H. W. Miller, Jr.
E. R. Piore
O. M. Scott
M. B. Smith
A. K. Watson

Last week CDC had a press conference during which they officially announced their 6600 system. I understand that in the laboratory developing this system there are only 34 people, "including the janitor." Of these, 14 are engineers and 4 are programmers, and only one person has a Ph. D., a relatively junior programmer. To the outsider, the laboratory appeared to be cost conscious, hard working and highly motivated.

Contrasting this modest effort with our own vast development activities, I fail to understand why we have lost our industry leadership position by letting someone else offer the world's most powerful computer. At Jenny Lake, I think top priority should be given to a discussion as to what we are doing wrong and how we should go about changing it immediately.

TJW, Jr:jmc

T. J. Watson, Jr.

cc: Mr. W. B. McWhirter

IBM Memo on CDC6600

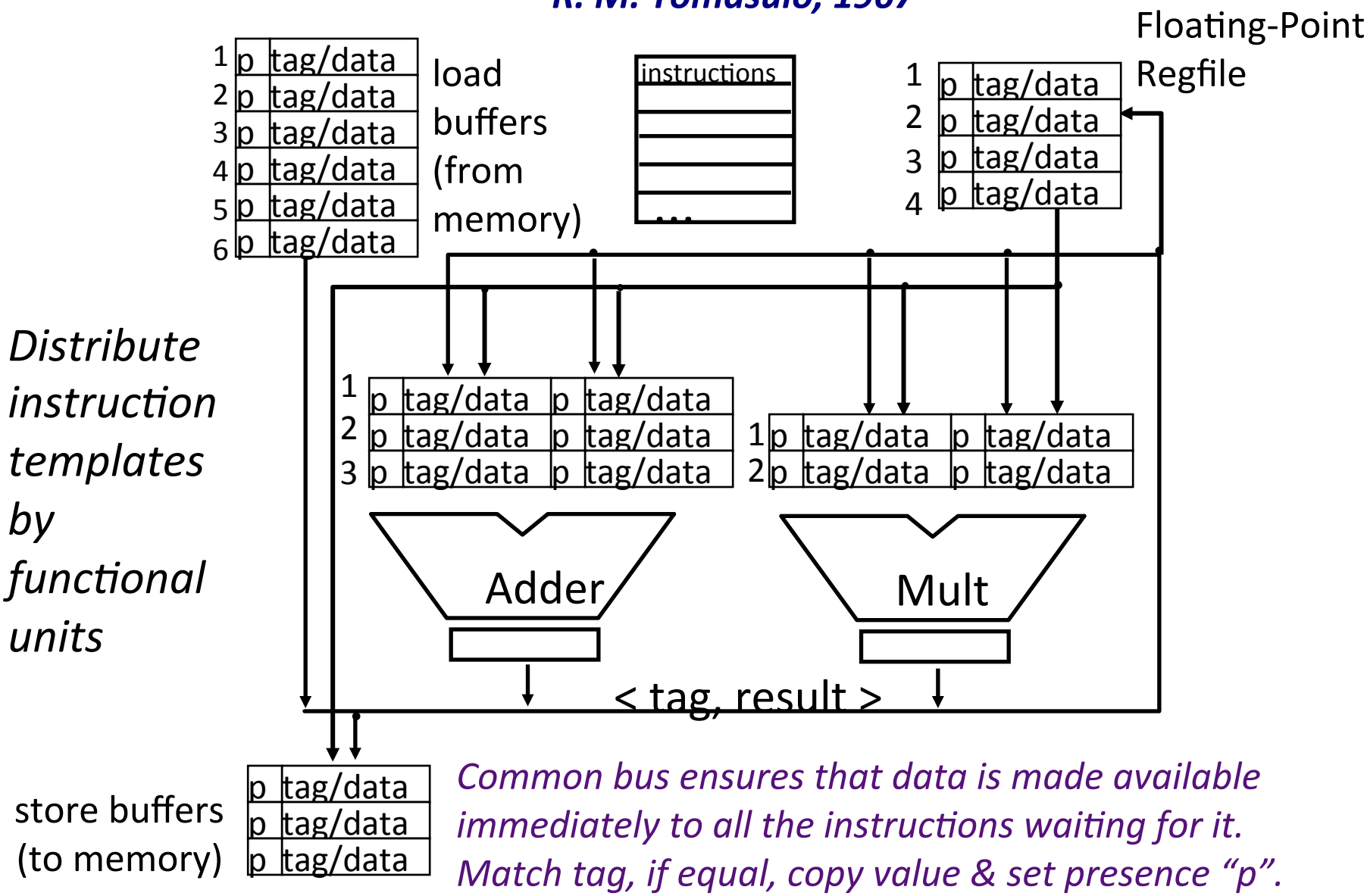
Thomas Watson Jr., IBM CEO, August 1963:

> *"Last week, Control Data ... announced the 6600 system. I understand that in the laboratory developing the system there are only 34 people including the janitor. Of these, 14 are engineers and 4 are programmers... Contrasting this modest effort with our vast development activities, I fail to understand why we have lost our industry leadership position by letting someone else offer the world's most powerful computer."*

To which Cray replied: *"It seems like Mr. Watson has answered his own question."*

IBM 360/91 Floating-Point Unit

R. M. Tomasulo, 1967



Common bus ensures that data is made available immediately to all the instructions waiting for it. Match tag, if equal, copy value & set presence "p".

IBM ACS

- Second supercomputer project (Y) started at IBM in response to CDC6600
- Multiple Dynamic instruction Scheduling invented by Lynn Conway for ACS
  - Used unary encoding of register specifiers and wired-OR logic to detect any hazards (similar design used in Alpha 21264 in 1995!)
- Seven-issue, out-of-order processor
  - Two decoupled streams, each with DIS
- Cancelled in favor of IBM360-compatible machines

Precise Traps and Interrupts

- This was the remaining challenge for early out-of-order machines
- Technology scaling meant plenty of performance improvement with simple in-order pipelining and cache improvements
- Out-of-order machines disappeared from 60s until 90s

Acknowledgements

- This course is partly inspired by previous MIT 6.823 and Berkeley CS252 computer architecture courses created by my collaborators and colleagues:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)