

## Scoreboard Summary

- Speedup 1.7 from compiler; 2.5 by hand  
BUT slow memory (no cache) limits benefit
- Limitations of *CDC 6600* scoreboard:
  - No forwarding hardware
  - Limited to instructions in single iteration (small *window*)
    - why?
  - Small number of functional units (structural hazards)
    - insts to same fu cannot be reordered
  - Wait for WAR hazards (after EX, before WB)
  - Prevent WAW hazards (in ID)

CSE 240A

Dean Tullsen

## Another Dynamic Algorithm: Tomasulo Algorithm

- For IBM 360/91 about 3 years after CDC 6600
- Goal: High Performance without special compilers
- Differences between IBM 360 & CDC 6600 ISA
  - IBM has only 2 register specifiers/instr vs. 3 in CDC 6600
  - IBM has 4 FP registers vs. 8 in CDC 6600

CSE 240A

Dean Tullsen

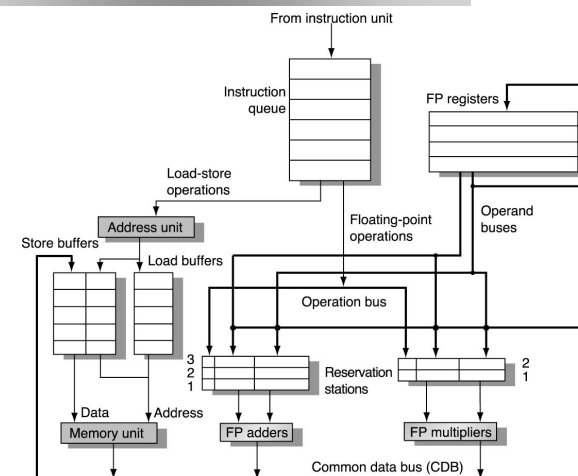
## Differences between Tomasulo Algorithm & Scoreboard

- Control & buffers distributed with Function Units vs. centralized in scoreboard; called “reservation stations”  
=> instrs schedule themselves
- Registers in instructions replaced by pointers to reservation station buffer scoreboard => registers primary operand storage  
Tomasulo => reservation stations as operand storage
- HW renaming of registers to avoid WAR, WAW hazards  
Scoreboard => both source registers read together (thus one could not be overwritten while we wait for the other).  
Tomasulo => each register read as soon as available.
- Common Data Bus broadcasts results to all FUs  
RS's (FU's), registers, etc. responsible for collecting own data off CDB
- Load and Store Queues treated as FUs as well

CSE 240A

Dean Tullsen

## Tomasulo Organization



CSE 240A

Dean Tullsen

## Reservation Station Components

---

Op—Operation to perform in the unit (e.g., + or –)  
Qj, Qk—Reservation stations producing source registers  
Vj, Vk—Value of Source operands  
Rj, Rk—Flags indicating when Vj, Vk are ready  
Busy—Indicates reservation station and FU is busy

Register result status—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions that will write that register.

## Three Stages of Tomasulo Algorithm

---

1. **Issue**—get instruction from FP Op Queue  
If reservation station free, the scoreboard issues instr & sends operands (renames registers).
2. **Execution**—operate on operands (EX)  
When both operands ready then execute;  
if not ready, watch CDB for result
3. **Write result**—finish execution (WB)  
Write on Common Data Bus to all waiting units;  
mark reservation station available.

## Tomasulo Example

---

```
ADDD F4, F2, F0
MULD F8, F4, F2
ADDD F6, F8, F6
SUBD F8, F2, F0
ADDD F2, F8, F0
```

Add, Sub 4 cycle latency  
Multiply 10 cycle latency

## Loop Example

---

```
loop: LD    F0, 0(R1)
      MULD  F4, F0, F2
      SD    F4, 100(R1)
      ADDI  R1, R1, #8
      BNEZ  R1, loop
```

## Tomasulo Summary

- Avoids WAR, WAW hazards of Scoreboard
- Allows loop unrolling in HW
- Not limited to basic blocks (provided branch prediction)
- Lasting Contributions
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation

CSE 240A

Dean Tullsen

## Scoreboard vs. Tomasulo, the score

	<u>Scoreboard</u>	<u>Tomasulo</u>
issue	when FU free	when RS free
read operands	from reg file	from reg file, CDB
write operands	to reg file	to CDB
structural hazards	functional units	reservation stations
WAW, WAR hazards	problem	no problem
register renaming	no	yes
instructions completing	no limit	1 per cycle (per CDB)
instructions beginning exec	1 (per set of read ports)	no limit

CSE 240A

Dean Tullsen

## Modern Architectures

- Alpha 21264+, MIPS R10K+, Pentium 4 use an *instruction queue*.
- Uses explicit register renaming. Registers are not read until instruction issues (begins execution). Register renaming ensures no conflicts.

Div R5, R4, R2  
 Add R7, R5, R1  
 Sub R5, R3, R2  
 Lw R7, 1000(R5)



R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	PR13
R6	PR20
R7	PR30
...	

CSE 240A

Dean Tullsen

## Dynamic Scheduling Key Points

- Dynamic scheduling is code motion in HW.
- Dynamic scheduling can do things SW scheduling (static scheduling) cannot.
- Scoreboard, Tomasulo have various tradeoffs
- Register renaming eliminates WAW, WAR dependencies.
- To get cross-iteration parallelism, we need to eliminate WAW, WAR dependencies.

CSE 240A

Dean Tullsen