EN164: Design of Computing Systems
Lecture 23: Processor / ILP 4

Professor Sherief Reda
http://scale.engin.brown.edu
Electrical Sciences and Computer Engineering
School of Engineering
Brown University
Spring 2011

[ material from Patterson & Hennessy, 4th ed]
Why do dynamic scheduling?

- Why not just let the compiler schedule code?
- Can’t always schedule around branches
  - Branch outcome is dynamically determined
- Not all stalls are predictable
  - e.g., cache memory misses
- Alias analysis
- Different implementations of an ISA have different latencies and hazards
Scoreboard dynamic scheduling limitations

- Out of order commit -> limited to basic blocks (cannot handle branches)
- Stalled issue on WAW
- Stalled issue on structural hazard
- Stalled WB on WAR

Need to do better if we want to improve IPC

```
div.d f1, f2, f3
beq f1, $0, exit
add r1, r2, r3
sub r4, r1, r4
......
exit:
```
Register renaming

• Reservation stations and reorder buffer effectively provide register renaming

• On instruction issue to reservation station
  – If operand is available in register file or reorder buffer
    • Copied to reservation station
    • No longer required in the register; can be overwritten
  – If operand is not yet available
    • It will be provided to the reservation station by a function unit
    • Register update may not be required
Speculation

- Predict branch and continue issuing and executing from the branch target direction
  - Don’t commit until branch outcome is determined and instructions are no longer speculative
- Load speculation
  - Speculate on load addresses
  - Enable reordering of load, store instructions
  - Don’t commit load until speculation cleared

```
div.d  f1, f2, f3
beq   f1, $0, exit
add   r1, r2, r3
sub   r4, r1, r4
.....
```

```
exit:
mult  r1, r2, r3
sw    r2, 0(r1)
lw    r6, 0(r5)
...
```
How can we do better?

• Register renaming eliminates WAR and WAW

• In-order commit can enable (1) *speculation* (where instructions from predicted targets are executed but not committed until branch is resolved) and (2) elimination of WAW hazards

We will study dynamic scheduling on x86 family of processors

```
div r1, r2, r3
add r2, r4, r5
add r3, r2, r6

...  
```

```
div.d f1, f2, f3
beq f1, $0, exit
add r1, r2, r3
sub r4, r1, r4
.....
```

```
exit:  
```

```
divd.d r1, r2, r3
...  
add r1, r2, r6
...  
```