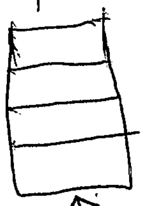
Lab 4

fetch buffer.



See A-70 Edition3 P1

pipeline: 5 stages ⇐ You do not have to implement a pipeline.

$\overbrace{\text{IF ID EX MEM WB}}$

easy ↗ IF (fetch) → Instruction ID (issue)

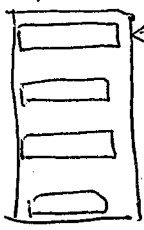
⇑ Control

Scoreboard — operand ready?

EX ⇓

☆ Instructions are issued to FUs before obtaining operands (from scoreboard)

4 Functional Units { FU1, FU2, FU3, FU4



← single reservation station but data is from registers.

fetch buffer

IF → ID → One FU
fetch issue

2 steps { Step 1: fetch
Step 2: add more FUs.

2 parts { ① IF issue ② ID oprand read ③ EX | MEM ④ WB

struct, WAW part 1 part 2

check RAW (how?) check WAR

Scoreboard

floating-point { FADD, FSUB, FMUL

2 register files { integer, floating-points (16)

(forwarding through reg. files)

3 data structures / 3 items in scoreboard

- Instruction status: in which of the 4 steps.
- FU status: { Busy, op (not op code, but FU's name); Fi destination reg.; Fj, Fk Soure reg. #; Qj, Qk FUs producing Fj, Fk
- register result status:

Rj, Rk: flags – Fj, Fk are ready and not yet read (set to no after operands are read).

Use NOPs for branches. 2 nops after each branch. If taken, flush the fetch buffer

. load/store unit : 2-cycle delay ~~issue~~

issue load $\xrightarrow{\hspace{3em}}$ write ~~to~~ reg. file.

both floating-point & integer

☆ check RAW (see Lec. 13a ~~P7 cycle 6~~)

LD. F2 45+ R3
MULT F0 F2 F4

how to detect this RAW?

```
Instruction_issue (ir) {
  ~~op~~ get op from ir;
  if (op is load) {
    get ~~F2~~ destination_register# from ir;
    register_result_status[destination_register#] = load/store unit;
  }
  ⋮
}
```