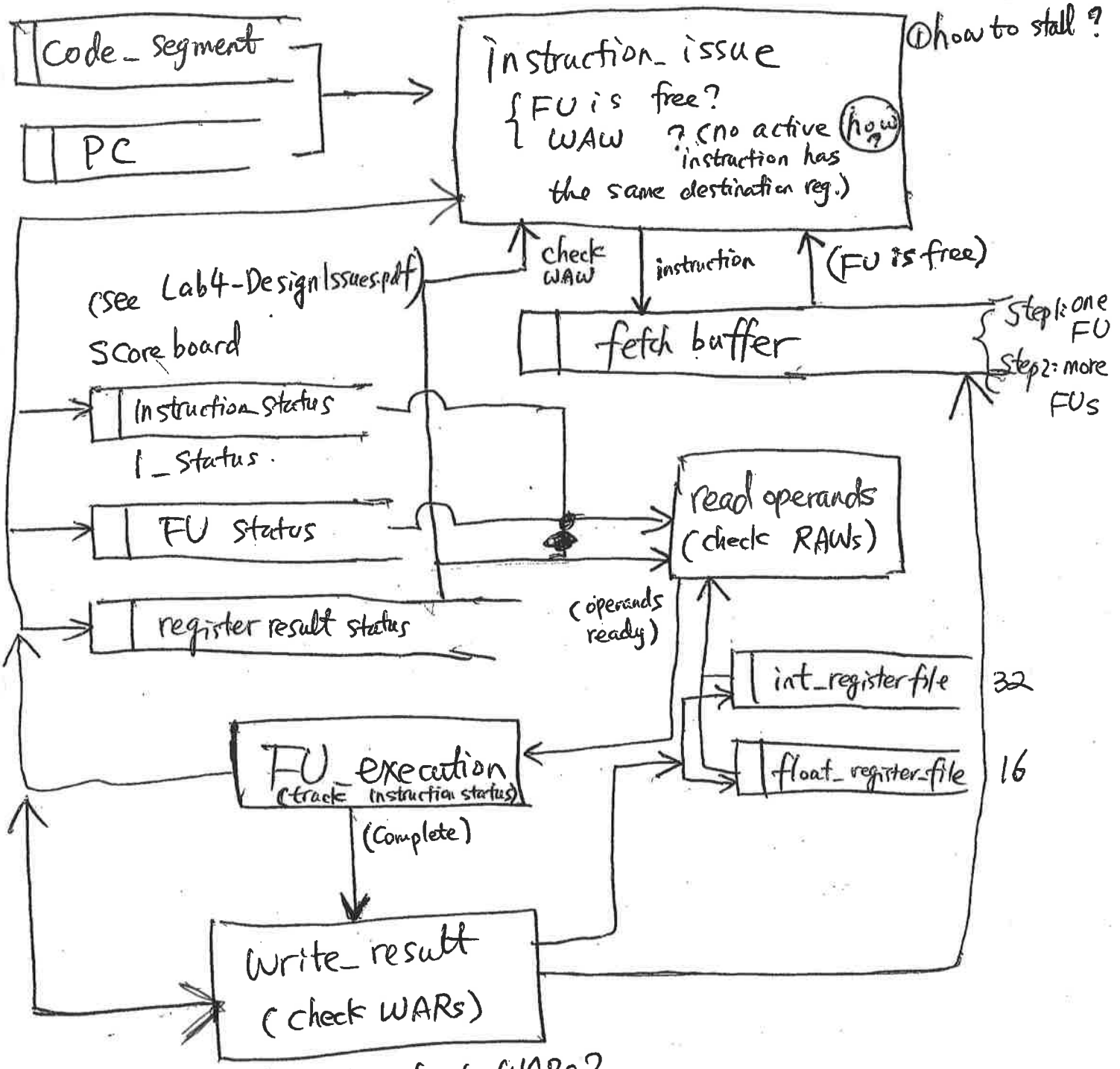
# Lab 4 Data Flow Diagram






4 FU { Int, FP_ADD, FP_MULT, FP_MEM

4 status { ISSUE, READ_OPS, EXE_COMPLETE, WRITE_RESULT

```
instruction_issue( code, PC, scob, &fetch_buf ) {

    instruction ← get instr. from code_seg ;
op ← decode ( instruction ) ;  /* op_a_reg, op_b_reg *
                                  dest_reg */
    if ( check_fu_busy ( scob.FU_status, op ) == TRUE ) {
        printf output ("stall") ;
            return STALL ;
    }
    if ( check_waw ( scob.R_status, dest_reg ) == YES ) {
            output "stall" ;
            return stall ;
    }
    set_fetch_buffer( &fetch_buf , instruction ) ;
     PC ++ ; ←
     return  non_STALL ;
}
```

```
main ( ) {
    input program_name ;
    code_segment ← load_program( program_name);
PC ← Simulator_init ( ) ;
    Simulator_run ( code_segment, PC )
    return 0 ;
}
```

```
Simulator_run( code_segment, PC) {
int int_reg_file [NUM_INT_REGS] ;
int fp_reg_file [NUM_FP_REGS] ;
    init two register files ;
    init scob ;
    running ← TRUE;
    while (running ) {
        print_scob (scob) ; /* for debuging */
Stall?
fetch_buf ← instruction_issue ( code_segment, PC, scob, *fetch_buf) ;
        read_operands ( scob , int_reg_file, fp_reg_file, fetch_buf) ;
        fu_execution ( int_reg_file, fp_reg_file, *scob, fetch_buf) ;
        fetch_buf ← write_result ( * scob ) ;
        clock_cycles ++ ;
        running ← check finished ? ;
    }
```

```
scob {
    int clockcycles
        ↑
        0 ;
    I_status[NUM_FU]
    FU_status[NUM_FUS]
    : Busy, op, Fi Fj Fk
                Qj Qk
                Rj Rk
    R_status[NUM_REGS]
        ↑ may need two arrays here.
        one for int, one for float ;
}
```

*int_reg_file
*fp_reg_file