Project:

Macros:

#define LD 0

#define ST 1

#define ADD 2

#define SUB 3

#define MULT 4

#define DIV 5

#define HALT 6

#define WAIT\_FOR\_OPERANDS = 0

#define EXECUTING = 1

#define WAIT\_FOR\_WRITE\_RESULT = 2

Structs:

1. inst\_queue (fifo – because issue is in order)
   * int num\_inst\_in\_queue
   * inst\* inst\_list -> linked list / instruction[16]
2. inst:
   * int pc
   * int imm -> from parsing
   * int src\_0 -> from parsing
   * int src\_1 -> from parsing
   * int dst -> from parsing
   * int opcode -> from parsing
   * int cycle\_issue -> at parsing set to -1
   * int cycle\_read\_operands -> at parsing set to -1
   * int cycle\_execute\_end -> at parsing set to -1
   * int cycle\_write\_result -> at parsing set to -1
3. func\_unit:
   * char\* unit -> SUB0 SUB1 SUB2

name given in parsing by the create\_func\_unit\_status\_table

* + int type -> SUB (macro). Given an parsing.
  + int delay -> Given at parsing
  + int state (WAIT\_FOR\_OPERANDS, EXECUTING, WAIT\_FOR\_WRITE\_RESULT)
  + int remaining\_time -> reset\_func\_unit changes back to delay
  + int busy
  + dst\_reg (Fi)
  + src\_reg\_1 (Fj)
  + src\_reg\_2 (Fk)
  + …

Globals:

int clk\_cycle = 0 - clock cycle

int pc = 0 - next instruction to fetch

int mem[4096] - memory image

float regs[16]   - processor float registers.

// Scoreboard

func\_unit\* func\_unit\_status\_table -> point to a dynamically allocated array

char\* reg\_result\_status\_table [16] -> point to NULL of name of unit

// Status

char\* trace\_unit\_name - unit to be traced (according to cfg.txt)

inst next\_inst\_to\_be\_issued

inst finished\_inst[4096] - array of finished instructions

in order to prevent “glitches”:

for each data base the main.c will hold an “updated” version

and in end of loop :

struct\_1 = updated\_struct\_1

struct\_2 = updated\_struct\_2

Functions:

// Read memory file

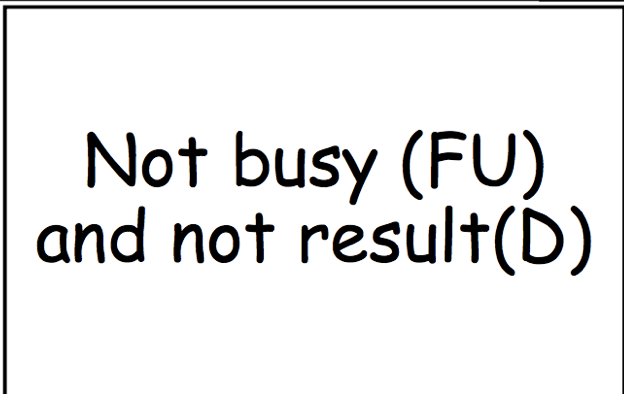
1. read\_memory\_into\_mem\_array

// parsing

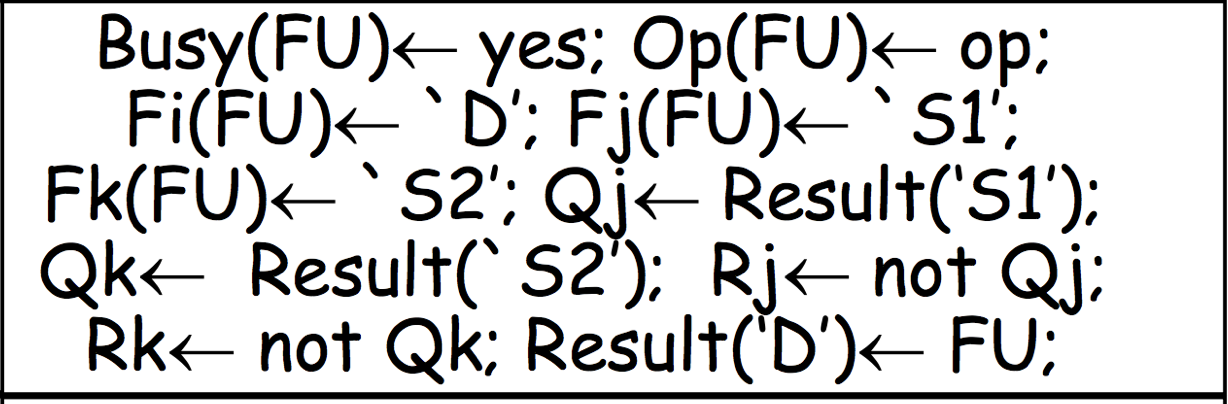
1. create\_func\_unit\_status\_table
2. set\_trace\_unit\_name
3. create\_inst -> create inst struct from line in file

// scoreboard flow (based on the “green” table from the lecture)

1. issue\_possible



1. issue

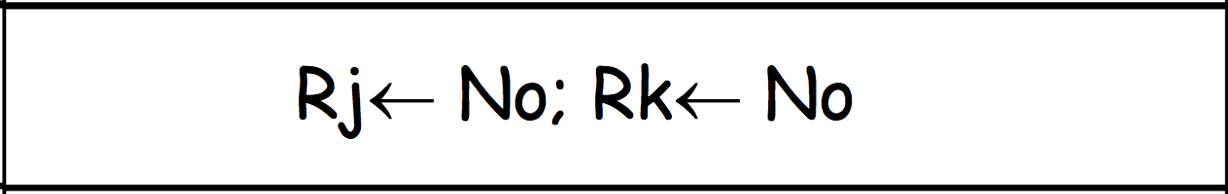


+ inst.issue\_cycle == clk\_cycle

1. operands\_ready

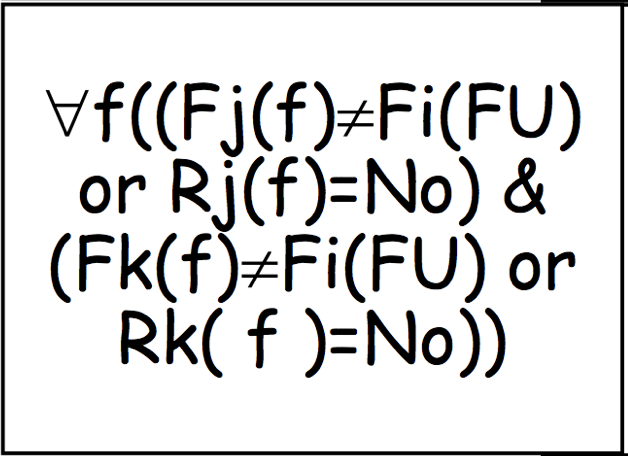


1. read\_operands

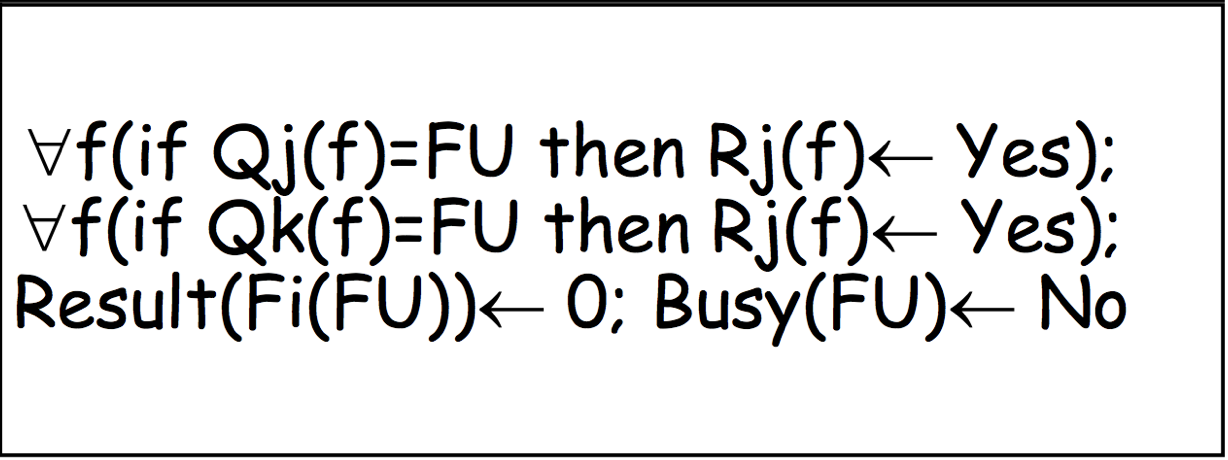


+ inst.read\_operands\_cycle == clk\_cycle

1. func\_unit\_done
2. write\_result\_possible



1. write\_result

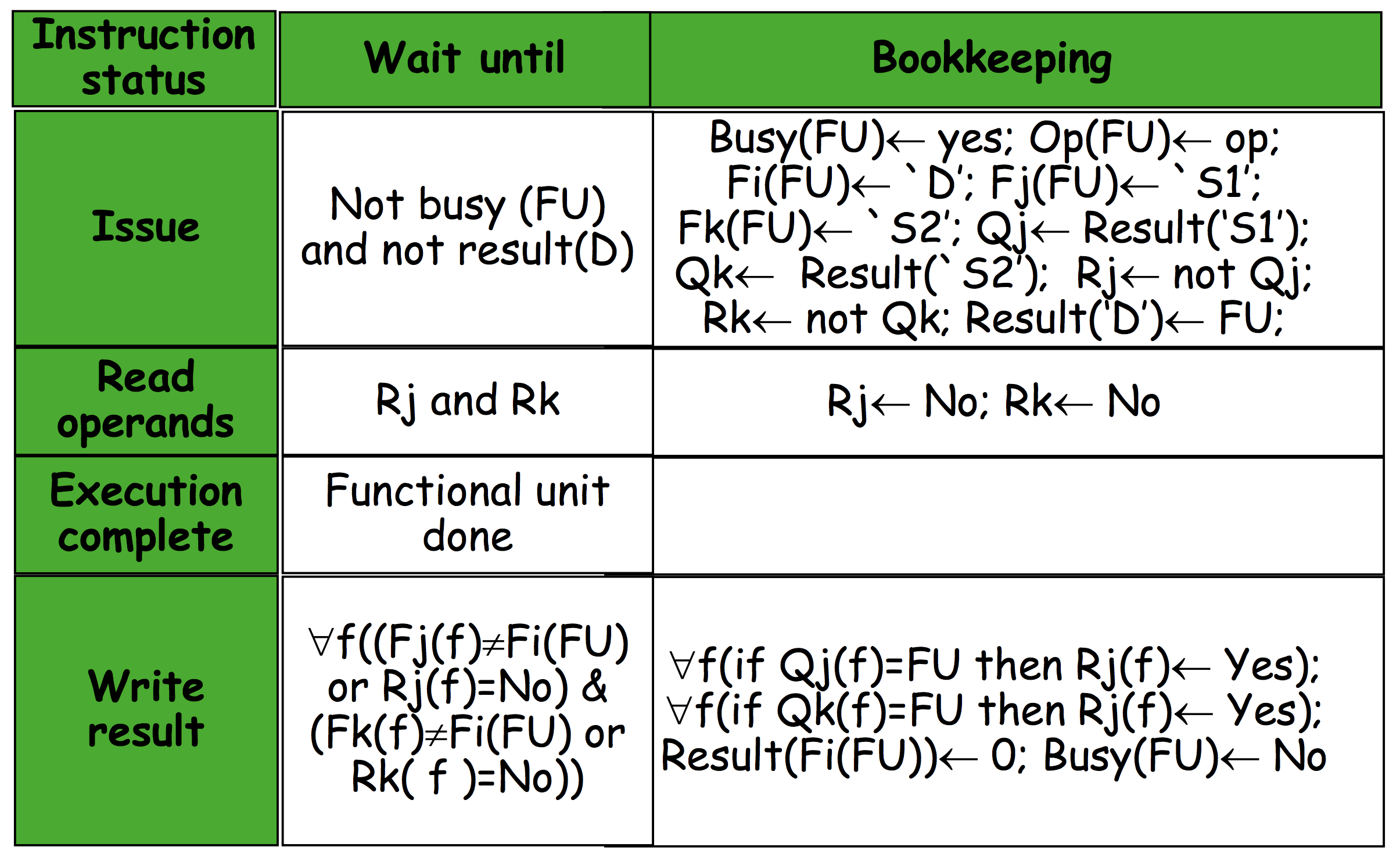


+ inst.write\_result\_cycle == clk\_cycle

1. reset\_func\_unit -> return the unit to reset state. Resets all fields except unit, type and delay

// write final memory

1. write\_mem\_image\_to\_file
2. write\_reg\_image\_to\_file



**flow – pseudo code :**

// Read Memory

open memin.txt

read\_memory\_into\_mem\_array

close memin.txt

// Read Configuration

open cfg.txt

create\_func\_unit\_status\_table

set\_trace\_unit\_name

close cfg.txt

\*\* open traceunit.txt -> file written on the fly

while { (the set operations executed in each clock cycle)

// FETCH (to instruction\_queue)

if (inst\_queue.num\_inst\_in\_queue < 16) // 16 defined in project

inst\_queue.push(fetch\_inst(pc,updated\_inst\_queue))

pc += 1

// ISSUE (only one instruction can be waiting to be issued)

inst = create\_inst(inst\_queue.get\_head)

if(inst.opcode == HALT)

break -> exit while loop

if(issue\_possible)

inst\_queue.pop

issue

// READ\_OPERANDS + EXEC + WRITE RESULT

for unit in func\_unit\_status\_table:

if (unit.unit == trace\_unit\_name)

print to traceunit.txt the necessary data on the unit

if(unit.state == WAIT\_FOR\_OPERANDS)

if(operands\_ready)

read\_operands(inst)

unit.state = EXECUTING

if(inst.state == EXECUTING)

if(func\_unit\_done)

unit.state = WAIT\_FOR\_WRITE\_RESULT

else

unit.remaning\_time -= 1

if(unit.state == WAIT\_FOR\_WRITE\_BACK)

if(write\_result\_possible)

write\_result

reset\_func\_unit

finished\_inst[inst.pc] = inst -> pass inst to finished list

clk\_cycle += 1

}

\*\* close traceunit.txt

// Printing Stage

open memout.txt

write\_mem\_image\_to\_file

close memout.txt

open regout.txt

write\_reg\_image\_to\_file

close regout.txt

open traceinst.txt

print traceinst.txt -> according to finished\_inst array

close traceinst.txt

// Termination

… free all dynamically allocated memory …