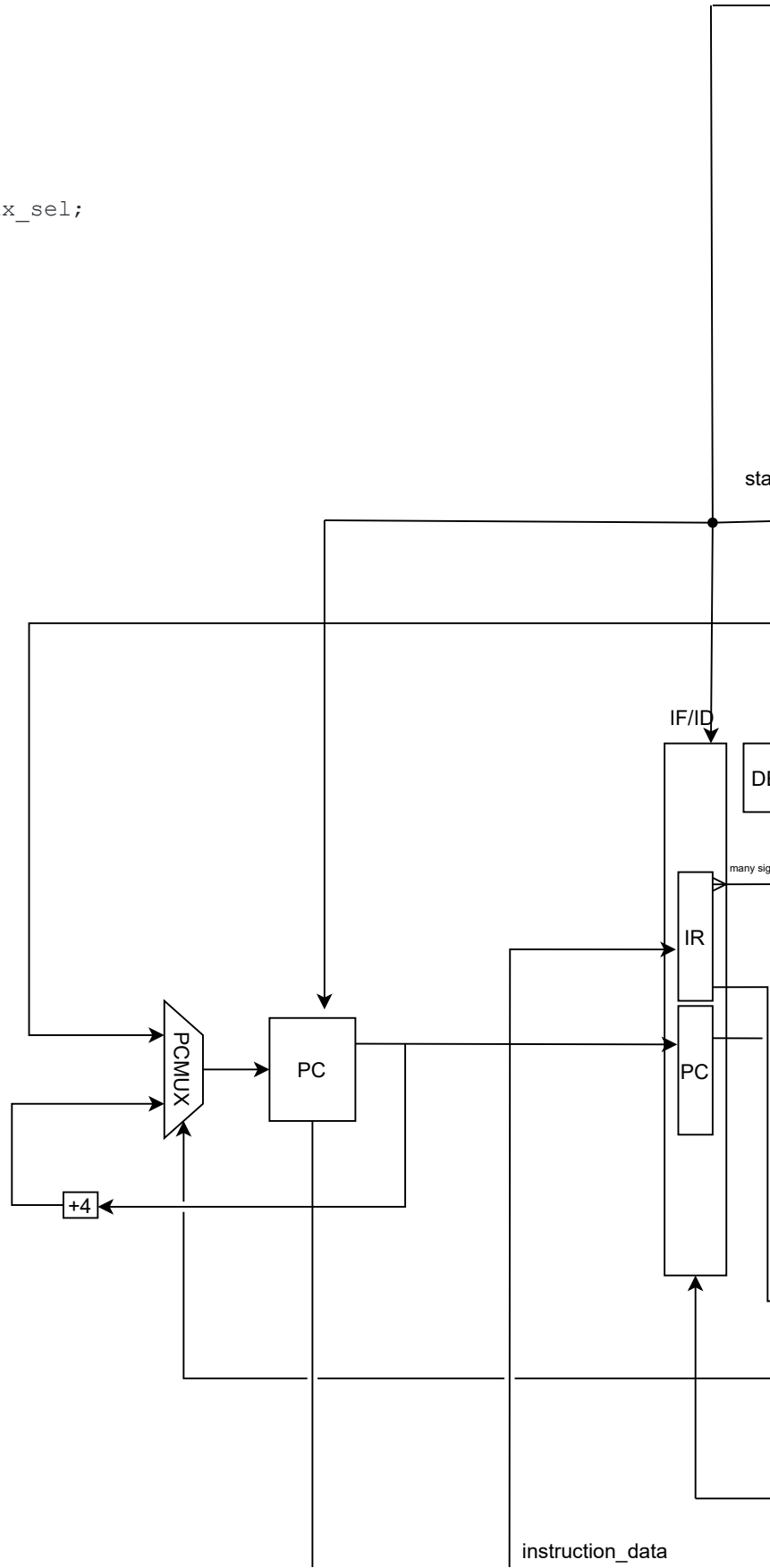
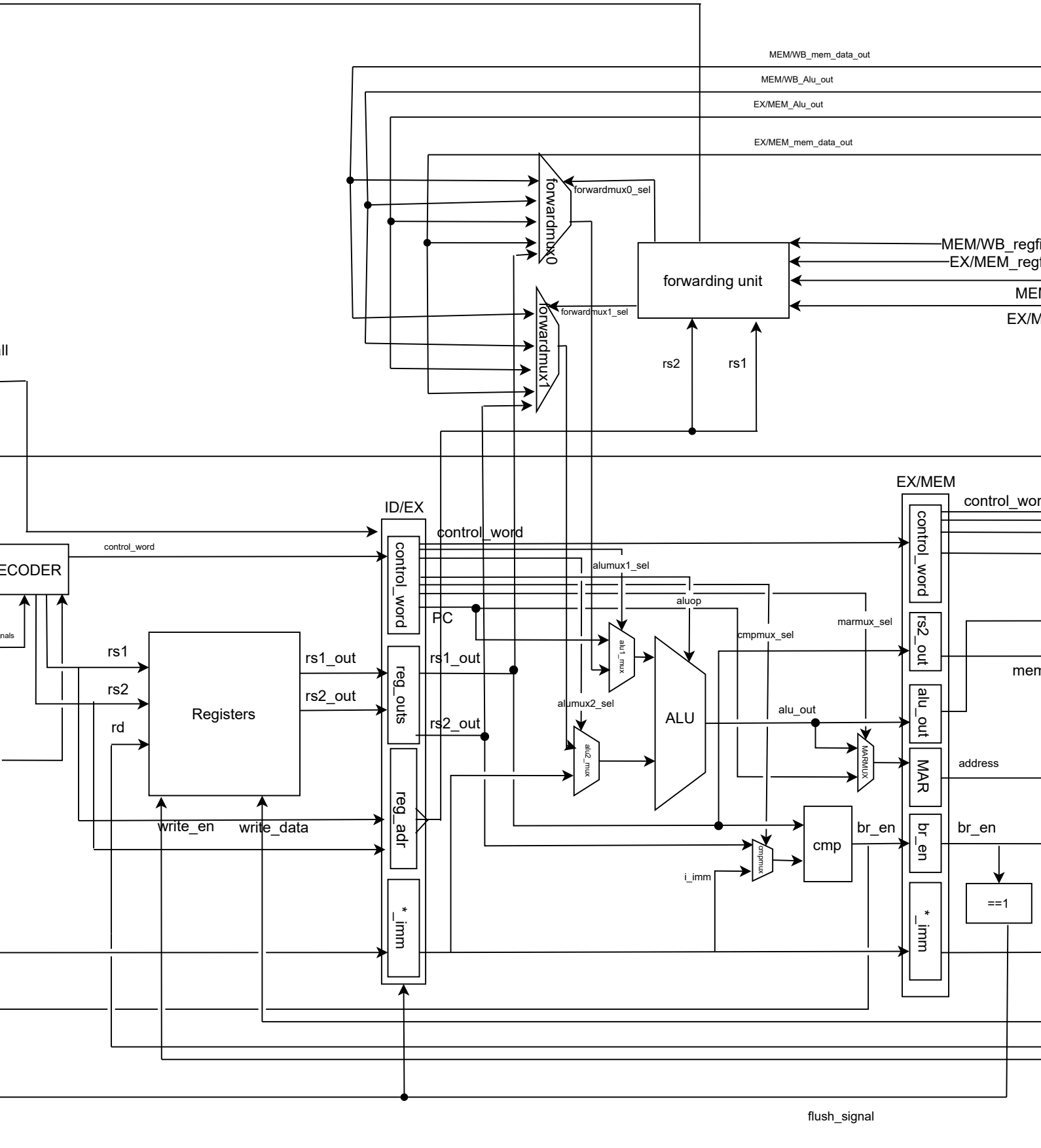
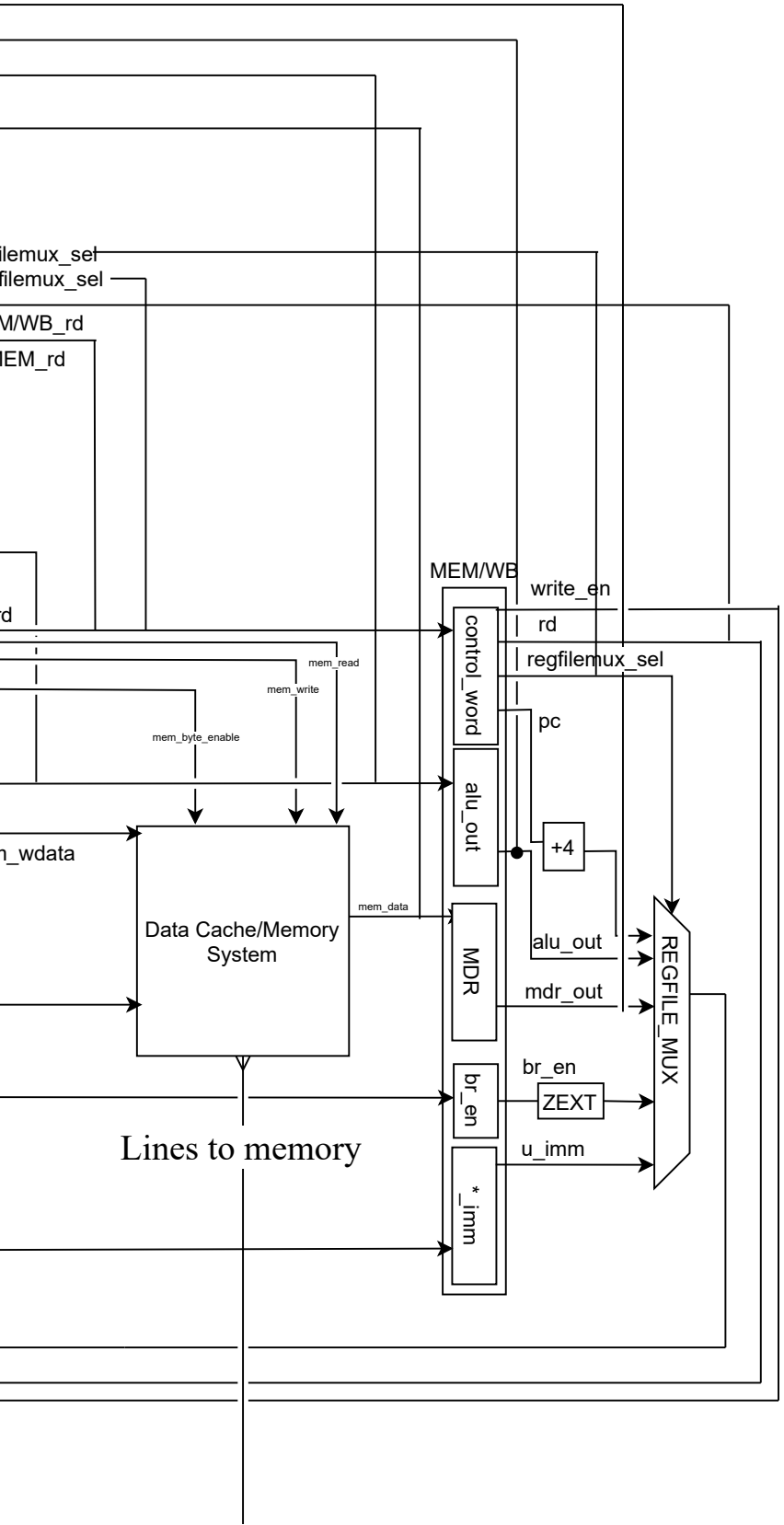
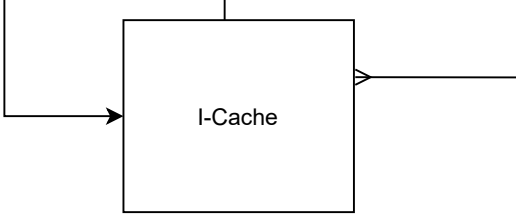
```
typedef struct packed {
    rv32i_opcode opcode;
    alu_ops aluop;
    logic load_regfile;

    logic mem_read;
    logic mem_write;


    regfilemux::refilemux_sel_t regfilemux_sel;
    pcmux::pcmux_sel_t pcmux_sel;
    alumux::alumux1_sel_t alumux1_sel;
    alumux::alumux2_sel_t alumux2_sel;
    marmux::marmux_sel_t marmux_sel;
    cmpmux::cmpmux_sel_t cmpmux_sel

    logic [3:0] mem_byte_enable;
    logic [1:0] mem_addr_bits;

    logic [5:0] rd;
    logic [2:0] funct3;
    logic [6:0] funct7;

    logic [31:0] PC


} rv32i_control_word;
```




MEM/WB_mem_data_out
MEM/WB_Alu_out
EX/MEM_Alu_out
EX/MEM_mem_data_out
forwardmux0
forwardmux0_sel
forwardmux1
forwardmux1_sel
forwarding unit
rs2
rs1
ID/EX
EX/MEM
control_word
PC
rs1
rs2
rd
Registers
rs1_out
rs2_out
reg_outs
reg_adr
* _imm
write_en
write_data
alumux1_sel
alumux2_sel
alu1_mux
alu2_mux
ALU
aluop
cmpmux_sel
marmux_sel
alu_out
MARMUX
address
MAR
cmp
cmpmux
br_en
i_imm
==1
flush_signal

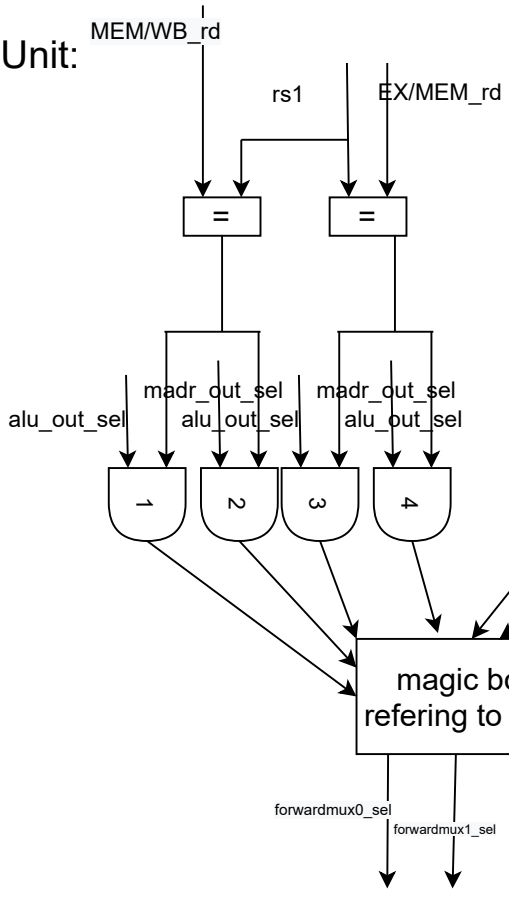

ilemux_sel
filemux_sel
M/WB_rd
IEM_rd
rd
mem_read
mem_write
mem_byte_enable
n_wdata
Data Cache/Memory System
mem_data
Lines to memory
MEM/WB
write_en
rd
regfilemux_sel
pc
control_word
alu_out
+4
MDR
br_en
*_imm
alu_out
mdr_out
br_en
ZEXT
u_imm
REGFILE_MUX



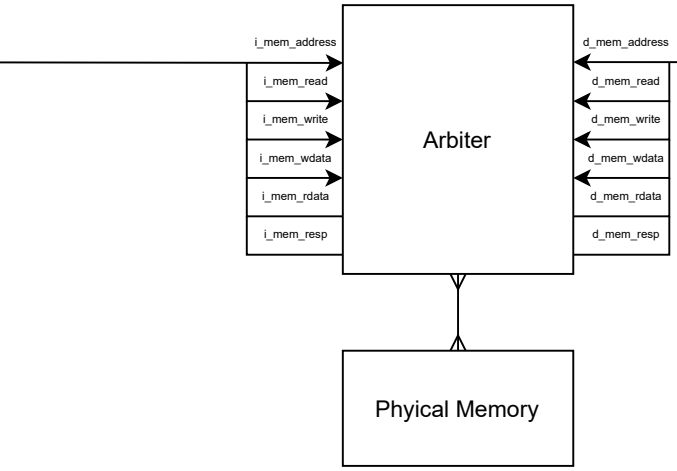
Arbiter:

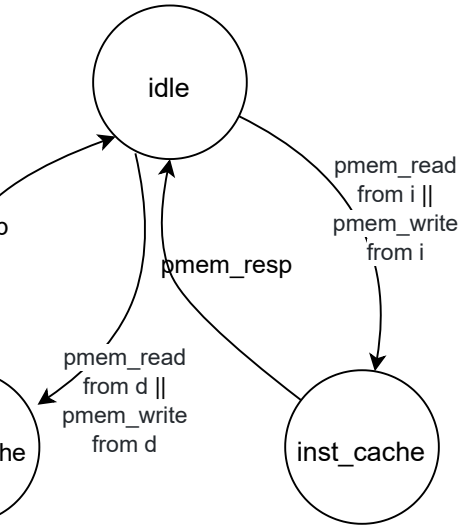
states: idle,
data_cache,
iinstrcution_cache

Forwarding Unit:

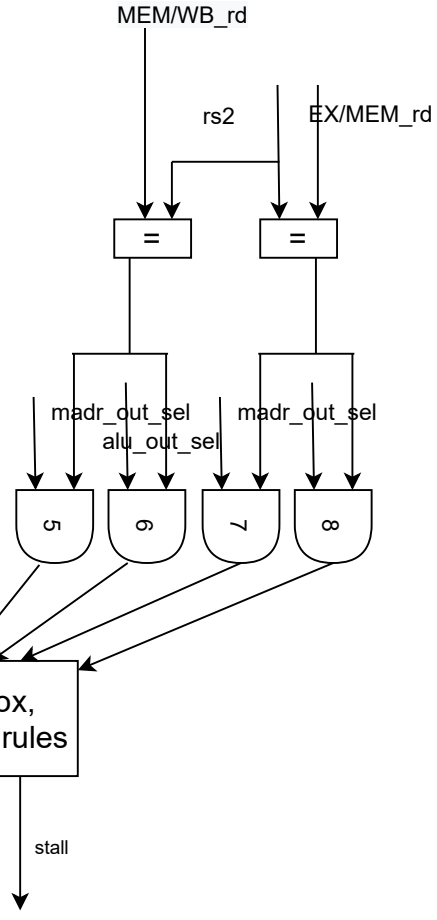


000: rs1/rs2 out
001:MEM/WB_mem_data_out
010:MEM/WB_Alu_out
011:EX/MEM_Alu_out
100: EX/MEM_mem_data_out





state assignm
idle: if (pmem_read_c_i || pmem_write_c_i) begin next_state = instruction_cache; end else if
data_cache: if (pmem_resp_m) begin next_state = idle; end
inst_cache: if (pmem_resp_m) begin next_state = idle; end



RULES

EX/MEM has priority then MEM/WB, for example, if MEM/WB_rd == rs1 and E
value from EX/MEM_rd to alu.

if MEM/WB_rd == rs1/rs2 and regfile_mux_sel == alu_out_sel, we should feed a
== mdr_out, we should feed mdr_out to alu. Similar f

if EX/MEM_rd == rs1/rs2 and regfile_mux_sel == mdr_out_sel, we

ent

(pmem_read_c_d || pmem_write_c_d) begin next_state = data_cache; end

X/MEM_rd == rs1, we should feed the

lu_out to the alu, and if regfile_mux_sel
or EX/MEM.

need to stall for one cycle.