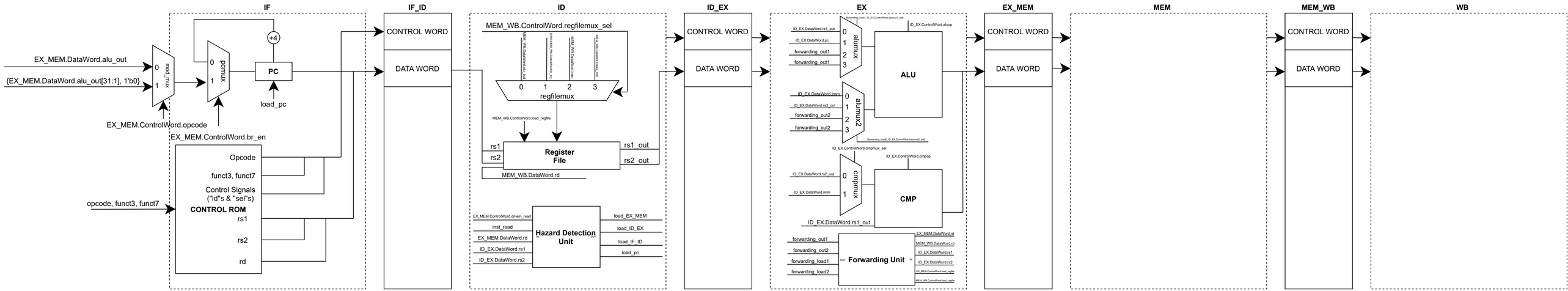



**Control Word**

Loads:
1. load_regfile

Muxes:
1. regfilemux_sel
2. cmpmux_sel
3. alumux_sel
4. modmux_sel

Operations:
1. cmpop
2. aluop

Memory:
1. dmem_read
2. dmem_write
3. mem_byte_enable

Miscellaneous:
1. opcode
2. br_en

**Data Word**

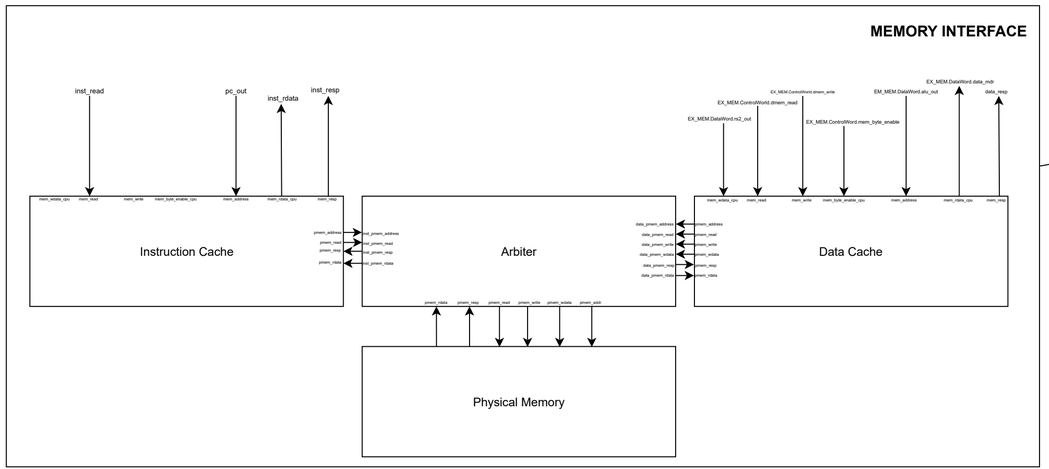
PC:
1. pc

Register Index:
1. rs1
2. rs2
3. rd

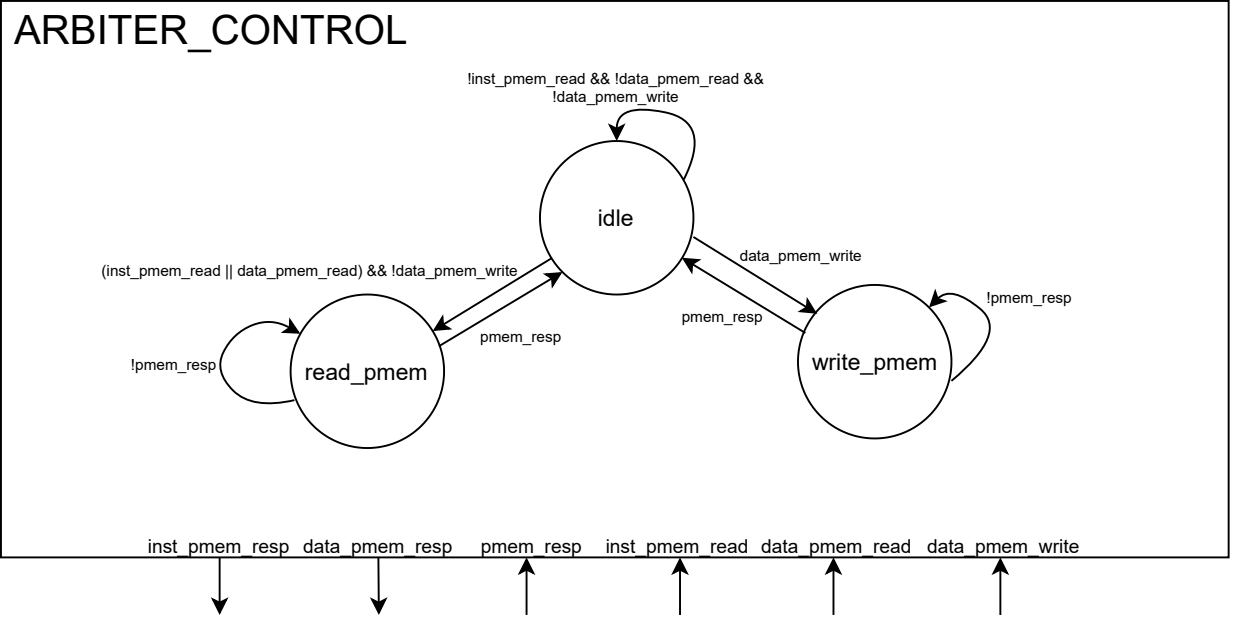
Outputs & Addresses:
1. rs1_out
2. rs2_out
3. alu_out

Immediates:
1. imm

Data:
1. data_mdr



# ARBITER_CONTROL

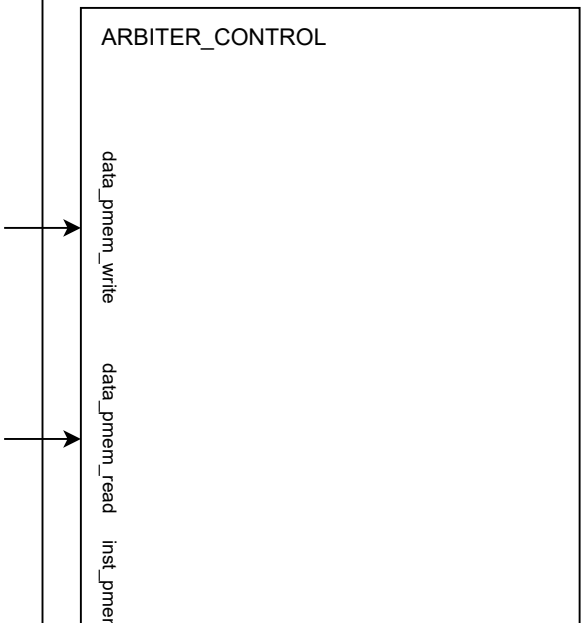



**IDLE**: does nothing (waits for a cache miss in either instruction or data cache)

**read_pmem**: waits for either inst_pmem_read or data_pmem_read then retrieves from physical memory. It will service the instruction memory request before data memory request if both read signals are high. Then when pmem_resp is 1, it will set either inst_mem_resp or data_mem_resp to 1 depending on what it just read.

**write_pmem**: waits for data_pmem_write then writes to physical memory. Once pmem_resp is 1, it will set data_pmem_resp to 1

# ARBITER_DATAPATH




Write logic:

```
if(state == write_pmem) {
        if(pmem_resp){
                data_pmem_resp = 1
        }else{
                pmem_addr = data_pmem_address
                pmem_write = 1
                pmem_wdata = data_pmem_wdata
        }
}
```

Read logic:

```
if(state == read_pmem) {
        if(inst_pmem_read){
                pmem_addr = inst_pmem_addr
                pmem_read = 1
        }else{
                pmem_addr = data_pmem_addr
                pmem_read = 1
        }
        if(pmem_resp){
                next_state = idle
                if(inst_pmem_read){
                        inst_pmem_rdata = pmem_rdata
                        inst_pmem_resp = 1
                }else{
                        data_pmem_rdata = pmem_rdata
                        data_pmem_resp 1
                }
        }
}
```

Idle Logic:

```
if(state == idle) {
        if(inst_pmem_read || data_pmem_read) && !data_pmem_write){
        next_state = read_pmem
        }else if(data_pmem_write){
                next_state = write_pmem
        }else{
                next_state = idle
        }
}
```


}
m_read
pmem_resp
inst_pmem_resp
data_pmem_resp