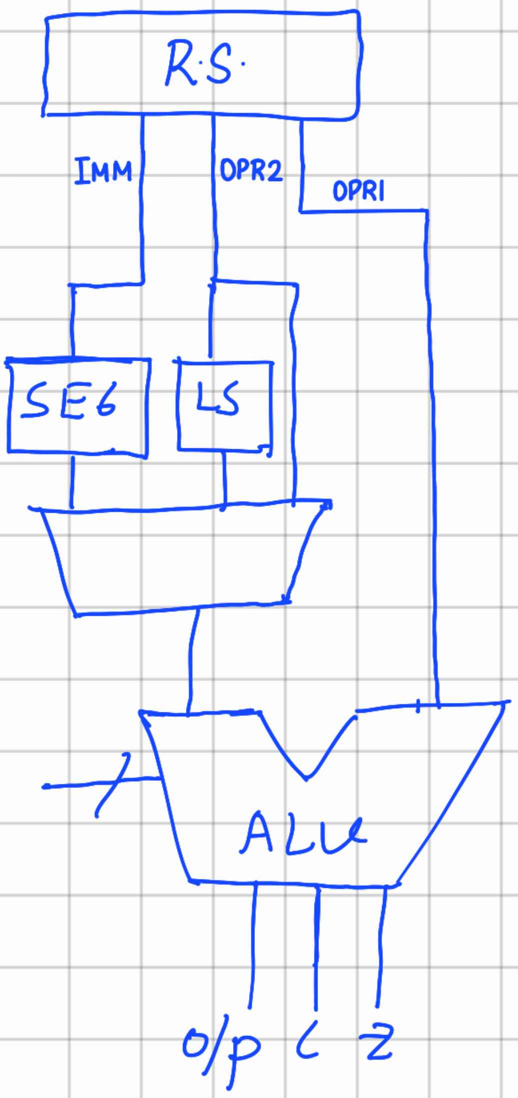
ALU EXEC



★ The ALU queue must send RB, RC, C, Z, mux control, ALU control

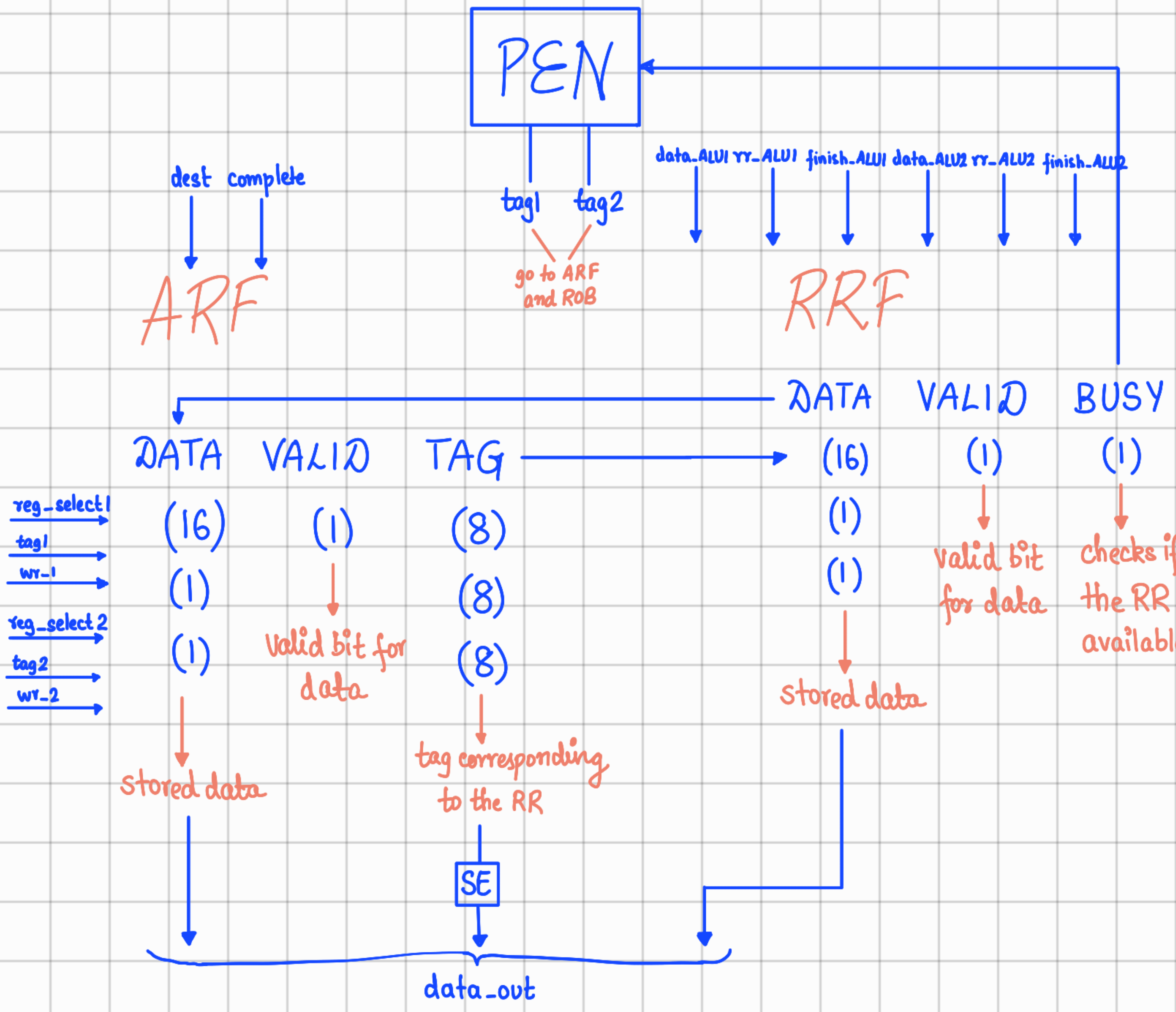
★ you also have controls for writing into C, Z & registers

RS

CONTROL	PC	OPR1	V1	OPR2	V2	IMM	C	V3	Z	V4	READY	ISSUED
(4)	(16)	(16)	(1)	(16)	(1)	(6)	(8)	(1)	(8)	(1)	(1)	(1)
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
represents the type of instr.	Program Counter	Contains the data or the sign extended RRF tag (8 bits)	valid for OPR1	same as OPR2	valid for OPR2	6-Bit Immediate Value	RRF tag or SE carry	valid for carry	RRF tag or SE zero	valid for zero	V1·V2·V3·V4	current entry can be replaced by new entry if ISSUED is set

ROB

PC	OUT	DEST	RR1	C	RR2	Z	RR3	FINISHED	COMPLETED
(16)	(16)	(5)	(8)	(1)	(8)	(1)	(8)	(1)	(1)
Program Counter	Output value from the Exec pipeline	Destination register	RR for DEST	carry bit after exec	RR for C	zero bit after exec	RR for Z	instr. has finished executing and OUT is updated ready to update RRF	instr. has written OUT, C, Z to respective RRs ready to update ARF



Operand Read : we get reg_select1 (from decode), tag1 (from PEN), wr_1 (control)

we look at the corresponding entry in ARF, if valid = '1', data_out = ARFData(i)

else we use the tag1 to look at the entry in RRF, if valid = '1', data_out = RRFData(i)

if the valid for RRF is '0', data_out = sign_extended (tag1)

we write tag into the corresponding ARF entry at the end of clock cycle and clear valid

busy bit is also set

SAME for second instruction

Instruction finishing : we get data_ALU1 (from 1st execution unit), rr_ALU1 (from ROB), finish1 (control)

when finish_ALU1 is '1', we write data_ALU1 to the RRF entry (given by rr_ALU1) and set valid

SAME for all execution units

Instruction Completion : we get dest (from ROB) and complete (control)

we look at the tag corresponding to the dest register and move data from the RRF to the ARF at the end of clock cycle. Busy bit cleared.