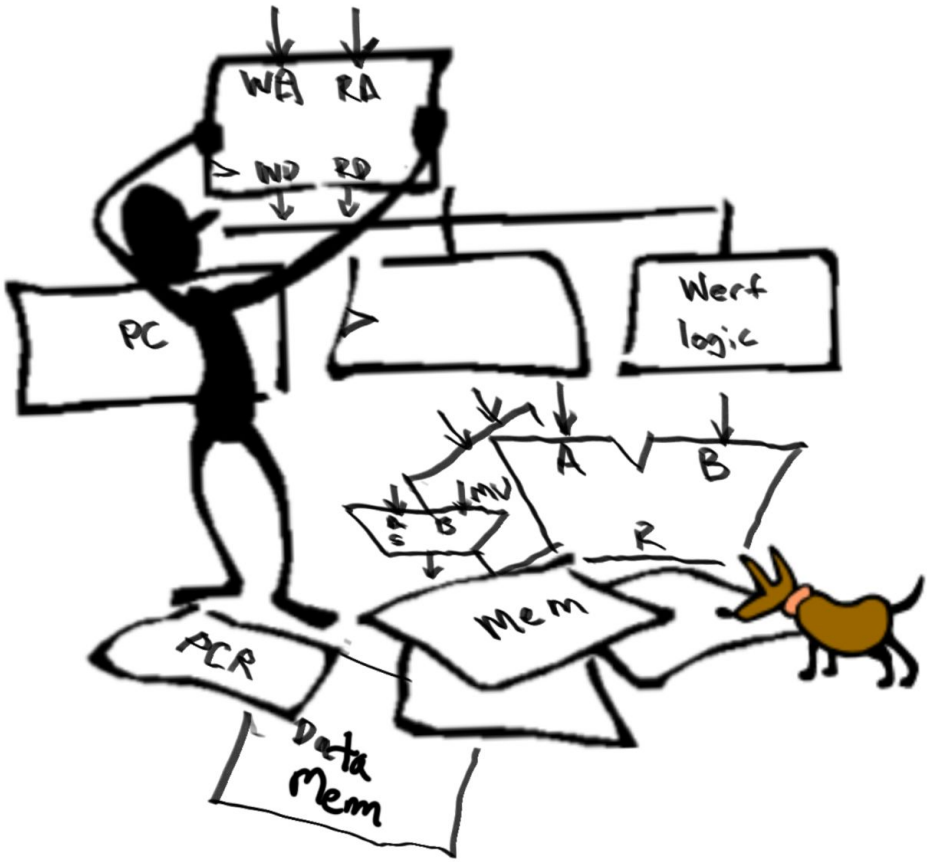
Building a Computer

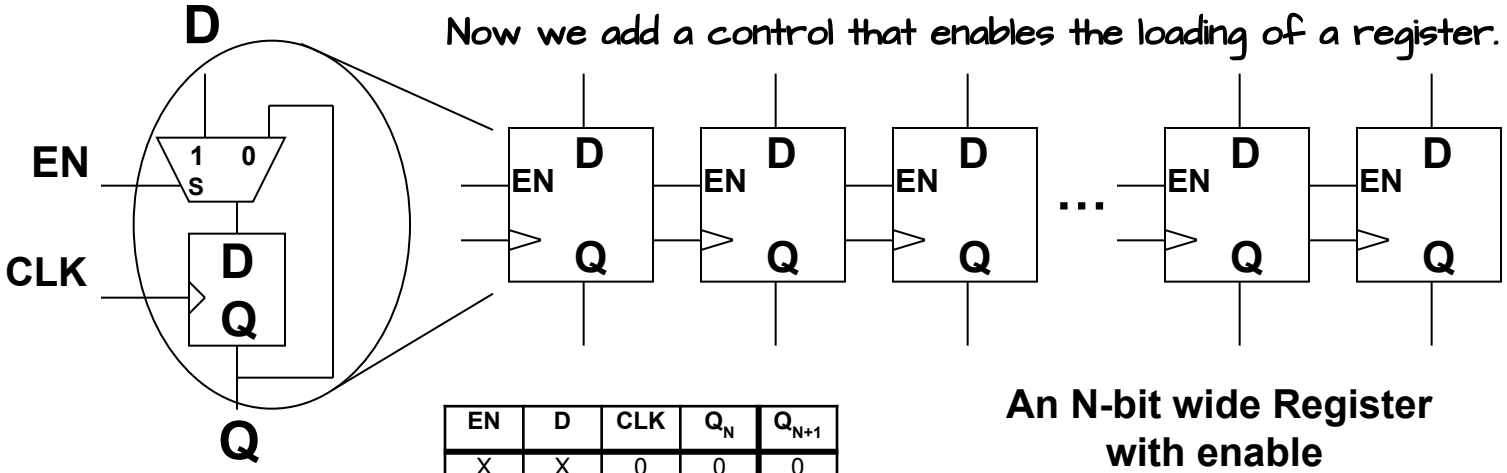


Problem Set #4 should go out tonight.

Need to move the second Midterm to 11/8

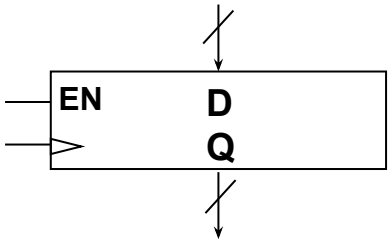
One More Functional Unit

Thus far, our building blocks units have focused on logical and arithmetic functions. We'll also need functional units for storing intermediate results. By now, we are used to the notion of building wide registers.



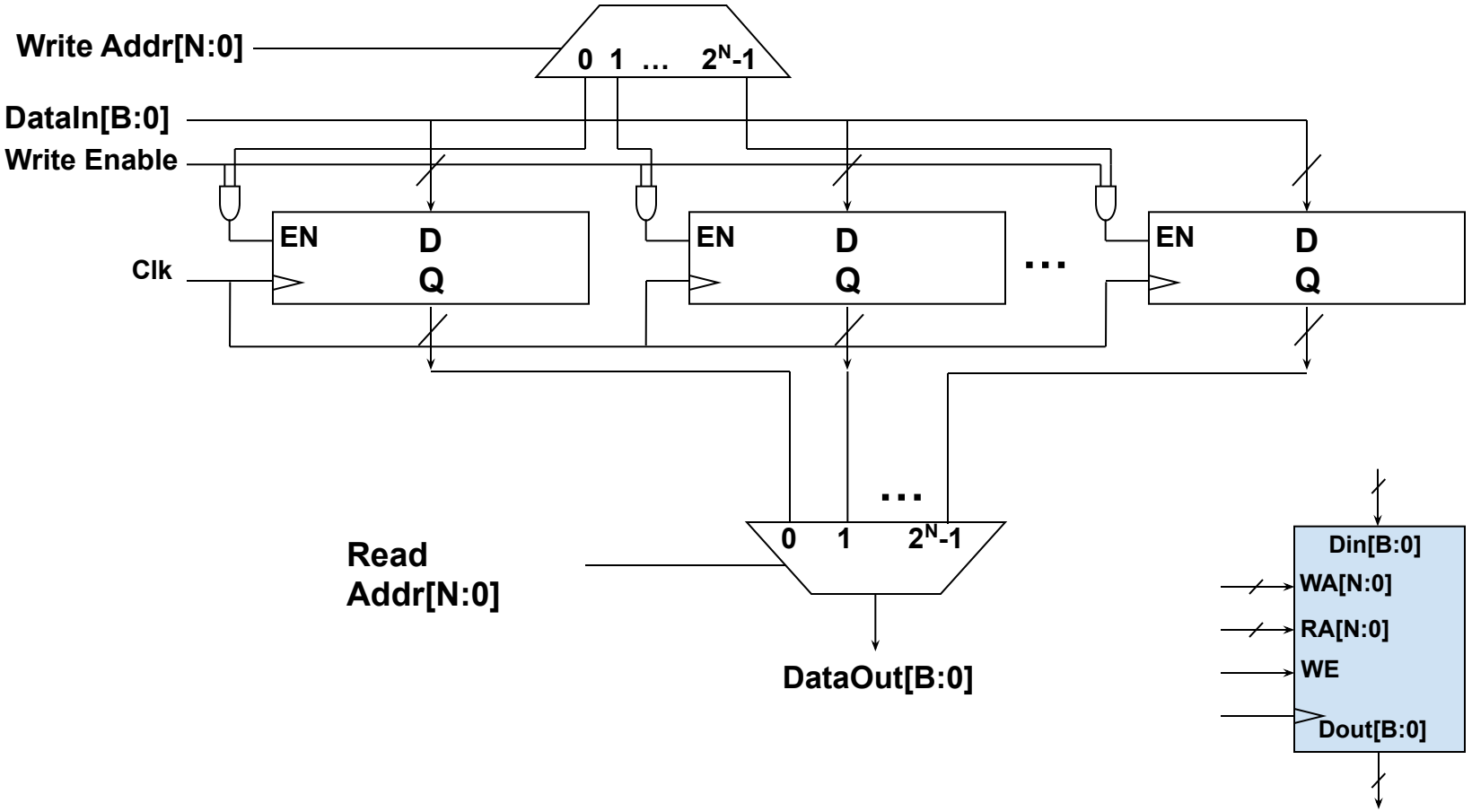
EN	D	CLK	Q_N	Q_{N+1}
X	X	0	0	0
X	X	0	1	1
X	X	1	0	0
X	X	1	1	1
0	X	↑	0	0
0	X	↑	1	1
1	0	↑	X	0
1	1	↑	X	1

An N-bit wide Register with enable



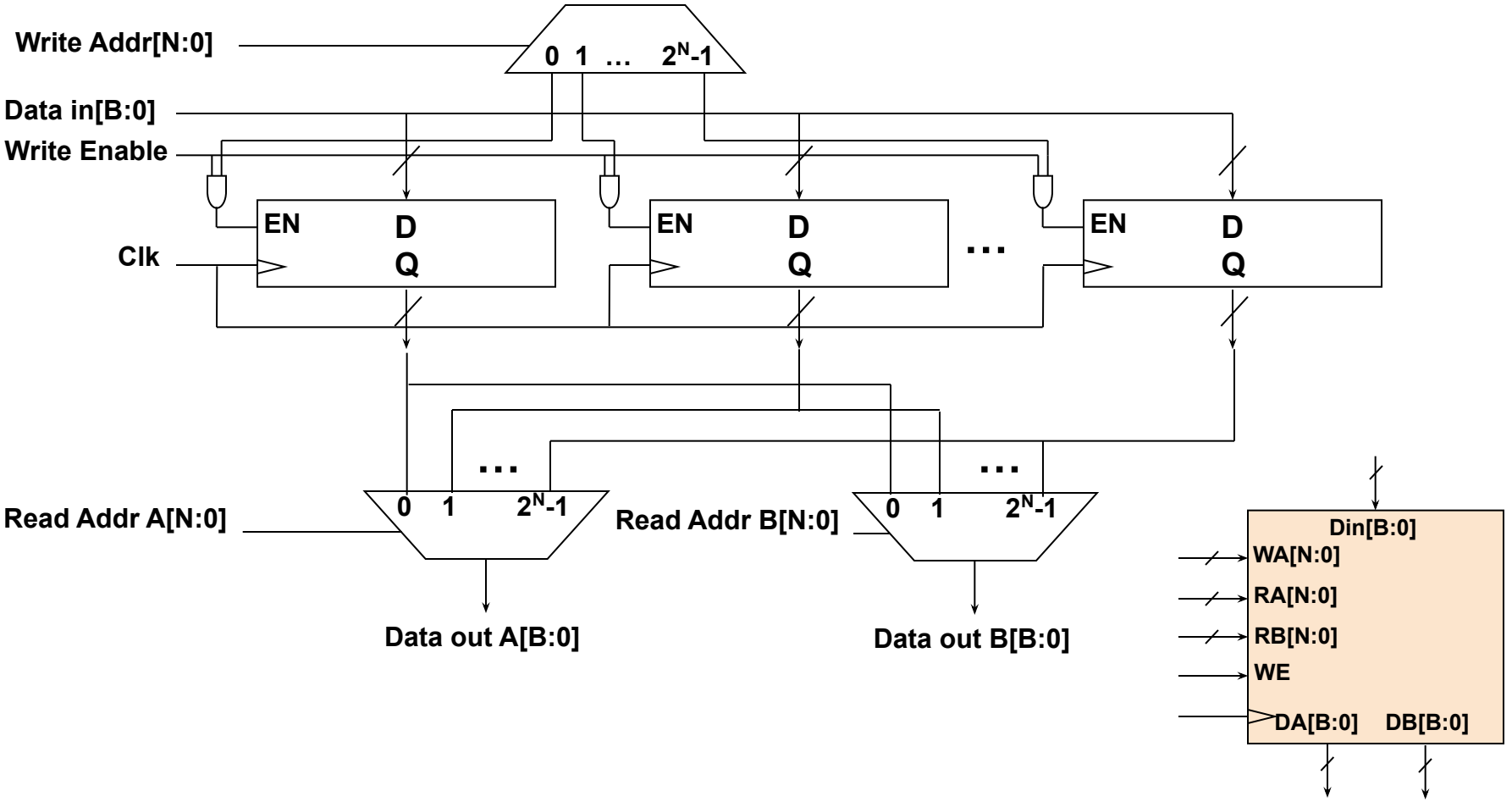
A Register File

We can also construct an addressable array of registers

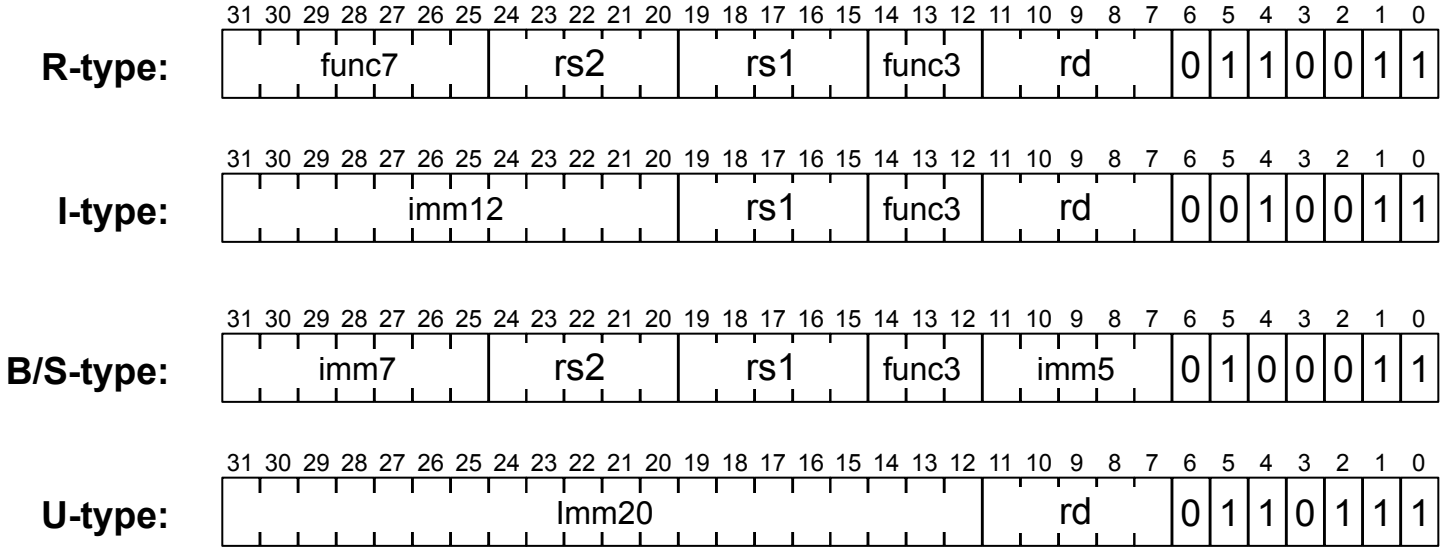


A Multi-Ported Register File

Multiple read ports by simply adding more output MUXs



The MiniRISC-V ISA



Four key instruction formats:

0) ALU with two register operands
1) ALU with a register and an immediate operand
2) Stores and Branches
5) Jumps and large constants (LUI, AUIPC)

R-Type Data Processing

ALU instructions with register operands

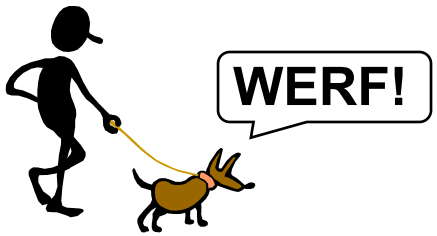
Rd - register file write address
Rn, Rm - register source operands
Shift or Rs - Optional shift of Rm

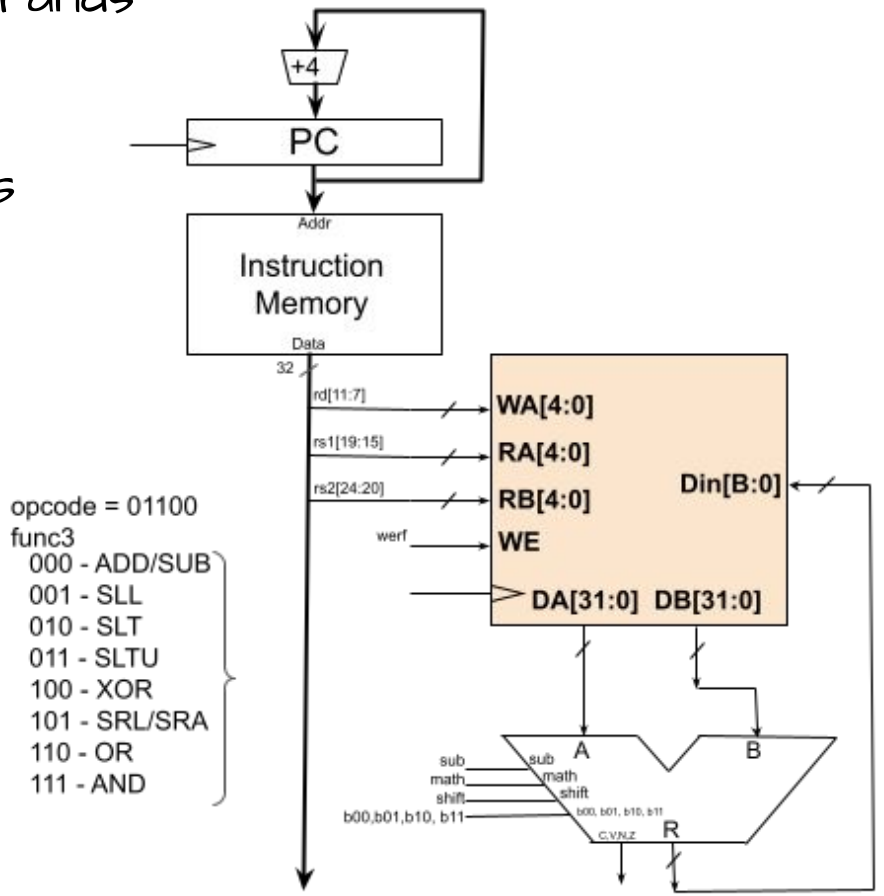
LA - direction and type of shift

S-bit - controls update of PSR

Func decoding from ALU lecture

Register write back controlled by WERF logic





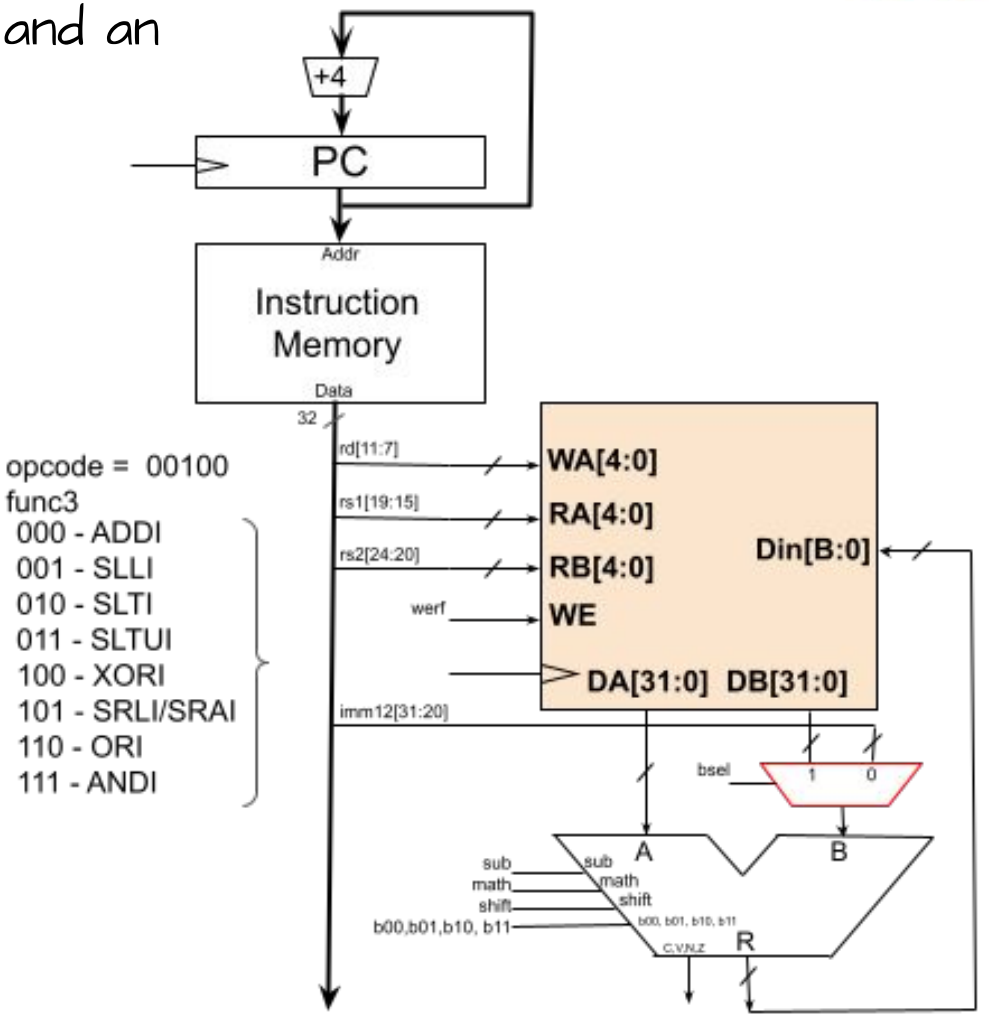
I-type Data Processing

ALU instructions with a register and an immediate operand

Rd - register file write address
Rs1 - register source operand
Imm12 - 12-bit immediate operand

Adds a mux to the B input of the ALU

- 12-bit immediate value is sign-extended 20-bits

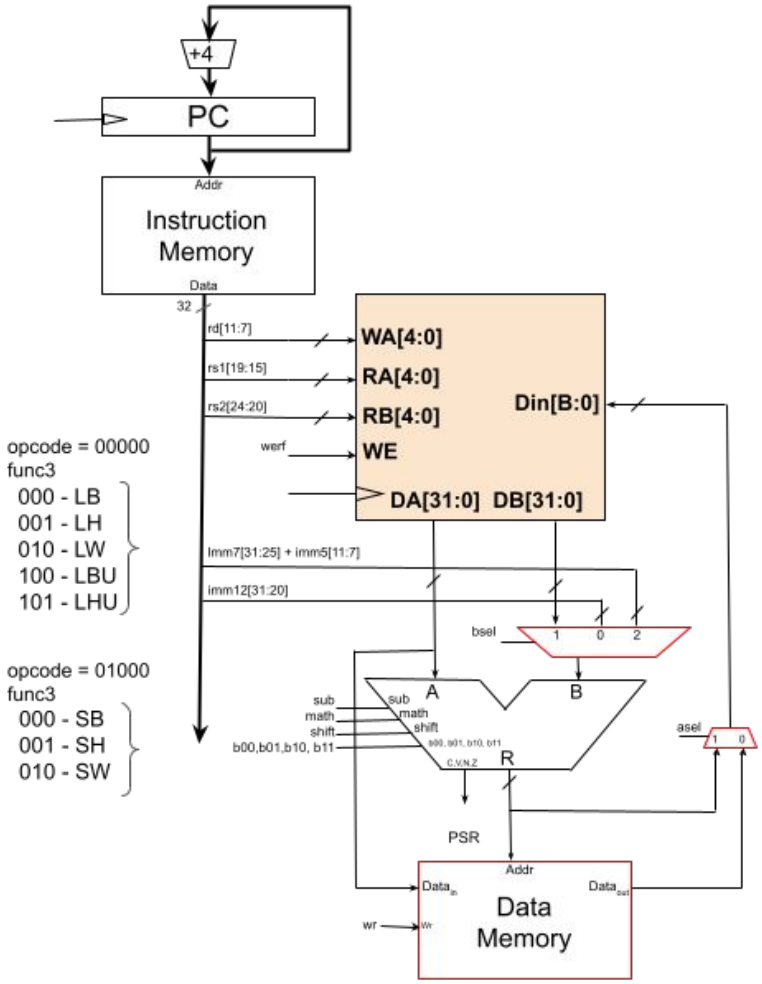


Data Transfer

Load/Store instructions using a base register and an immediate offset

Rd - loaded register
Rs1 - stored register
Rs2 - base register
Imm12 - 12-bit immediate load offset
Imm7+ Imm5 - 12-bit immediate store offset

Widens mux to the ALU's B input
- 12-bit immediate value is zero-extended 20-bits



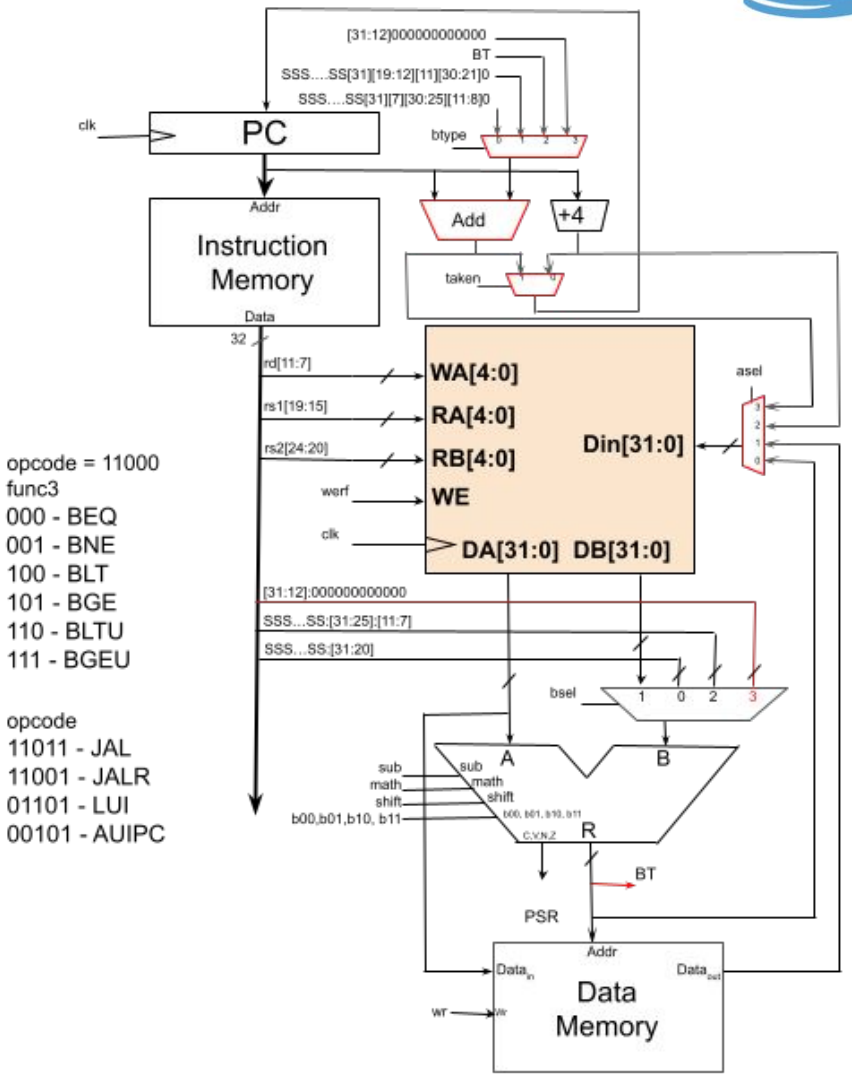
Branch, Jump, LUI and AUIPC

Branches add a 12-bit "sign-extended" immediate value (multiplied by 2) to the current PC if taken.

Jumps always add a 20-bit sign-extended immediate value and support saving PC+4.

We also need to be able to compute the PC-relative offsets used by the AUIPC instruction, and large immediates of LUI and store them into the register file

.

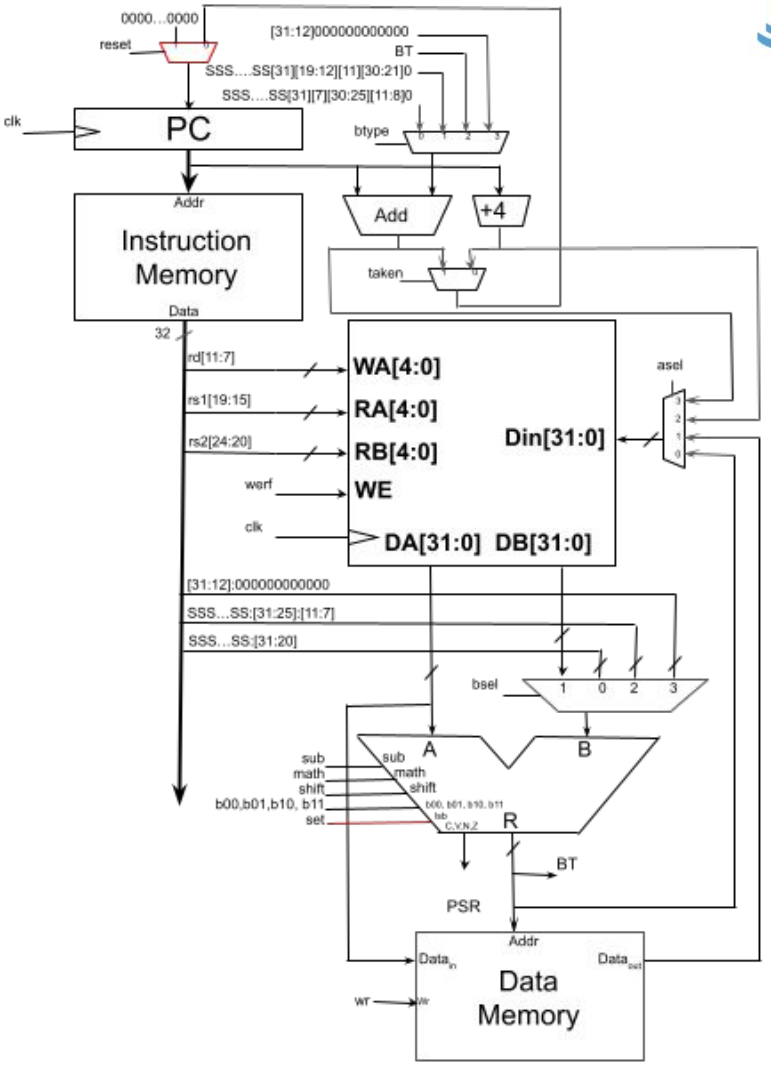


A FEW LOOSE ENDS

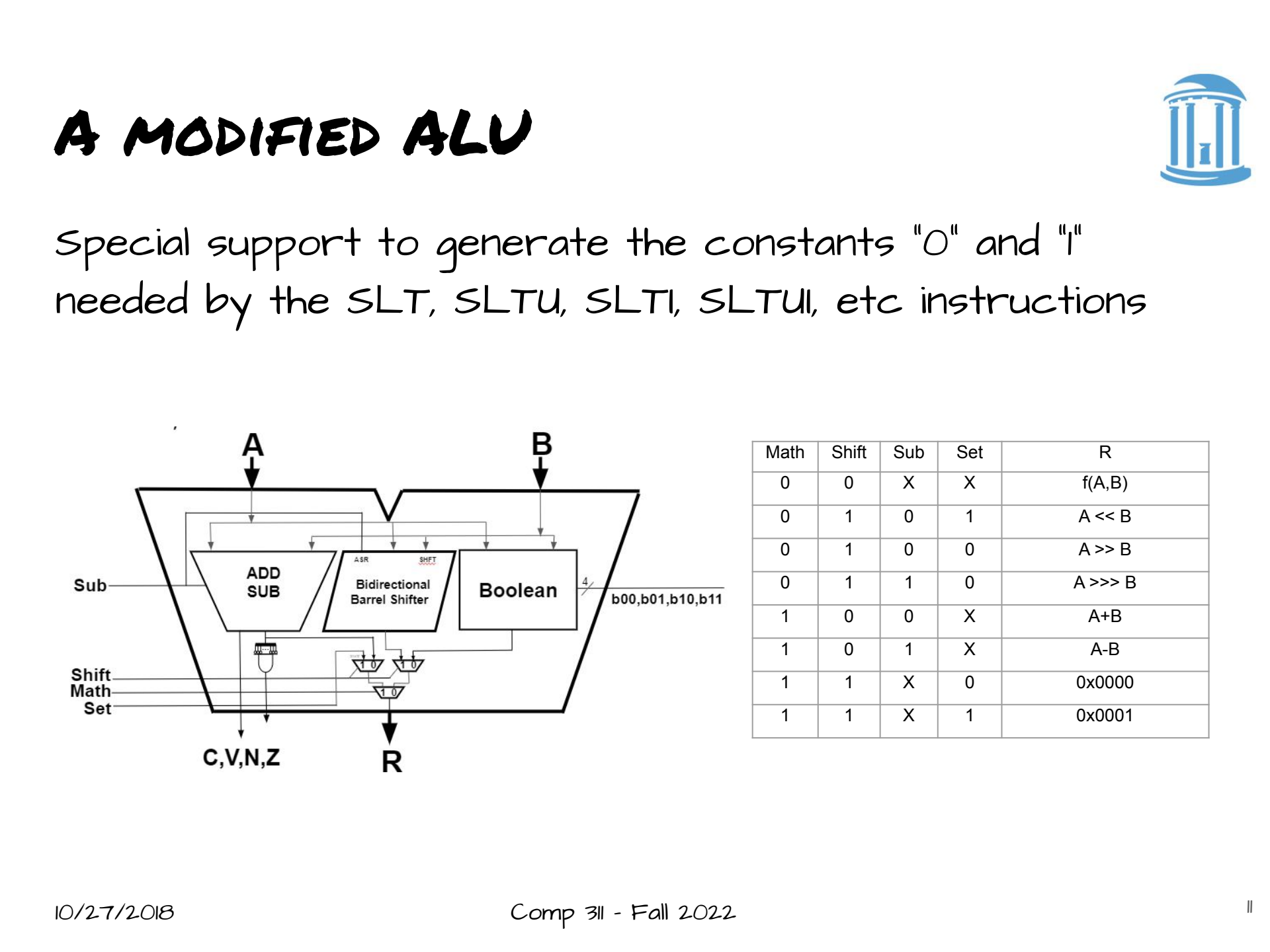
Next we will add a "reset" signal that starts execution at a known address by forcing a value into the PC. Like miniRISCV our cpu starts execution at location 0.

We will also make a small change to the ALU design discussed in Lecture 14

.



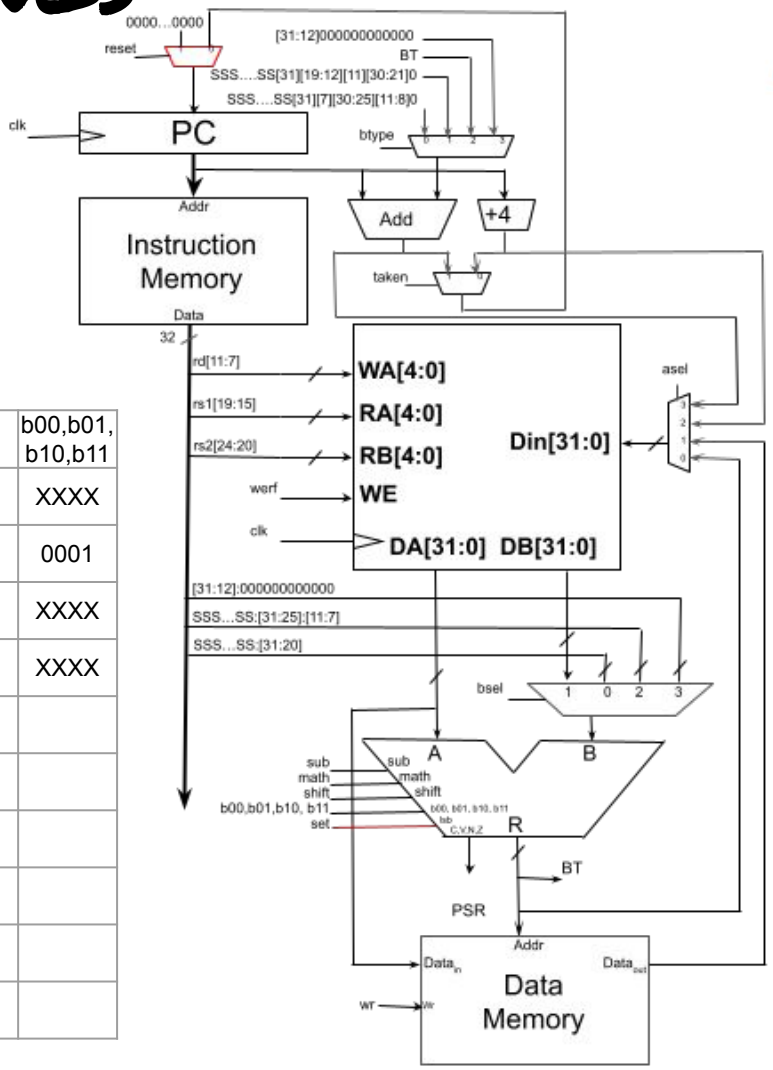
A MODIFIED ALU



Special support to generate the constants "0" and "1" needed by the SLT, SLTU, SLTI, SLTUI, etc instructions

Math	Shift	Sub	Set	R
0	0	X	X	f(A,B)
0	1	0	1	A << B
0	1	0	0	A >> B
0	1	1	0	A >>> B
1	0	0	X	A+B
1	0	1	X	A-B
1	1	X	0	0x0000
1	1	X	1	0x0001

Control Line Settings

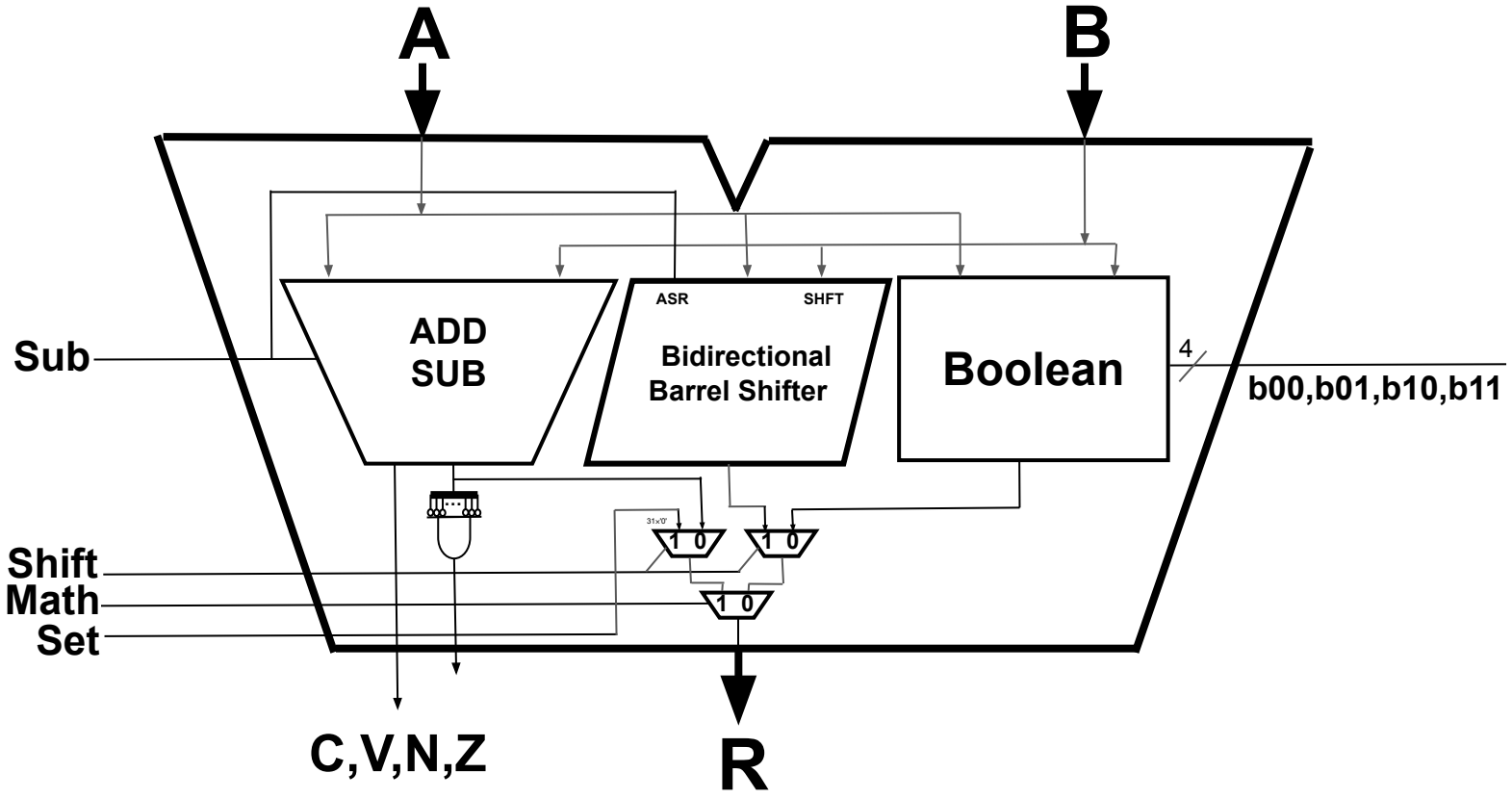


Instr	btype	taken	bsel	asel	werf	wr	math	shift	sub	set	b00,b01, b10,b11
add	X	0	1	0	1	0	1	0	0	X	XXXX
andi	X	0	0	0	1	0	0	0	X	X	0001
lw	X	0	0	1	1	0	1	0	0	X	XXXX
sw	X	0	2	X	0	1	1	0	0	X	XXXX
lui											
slti											
beq											
jalr											
auipc											
xor											

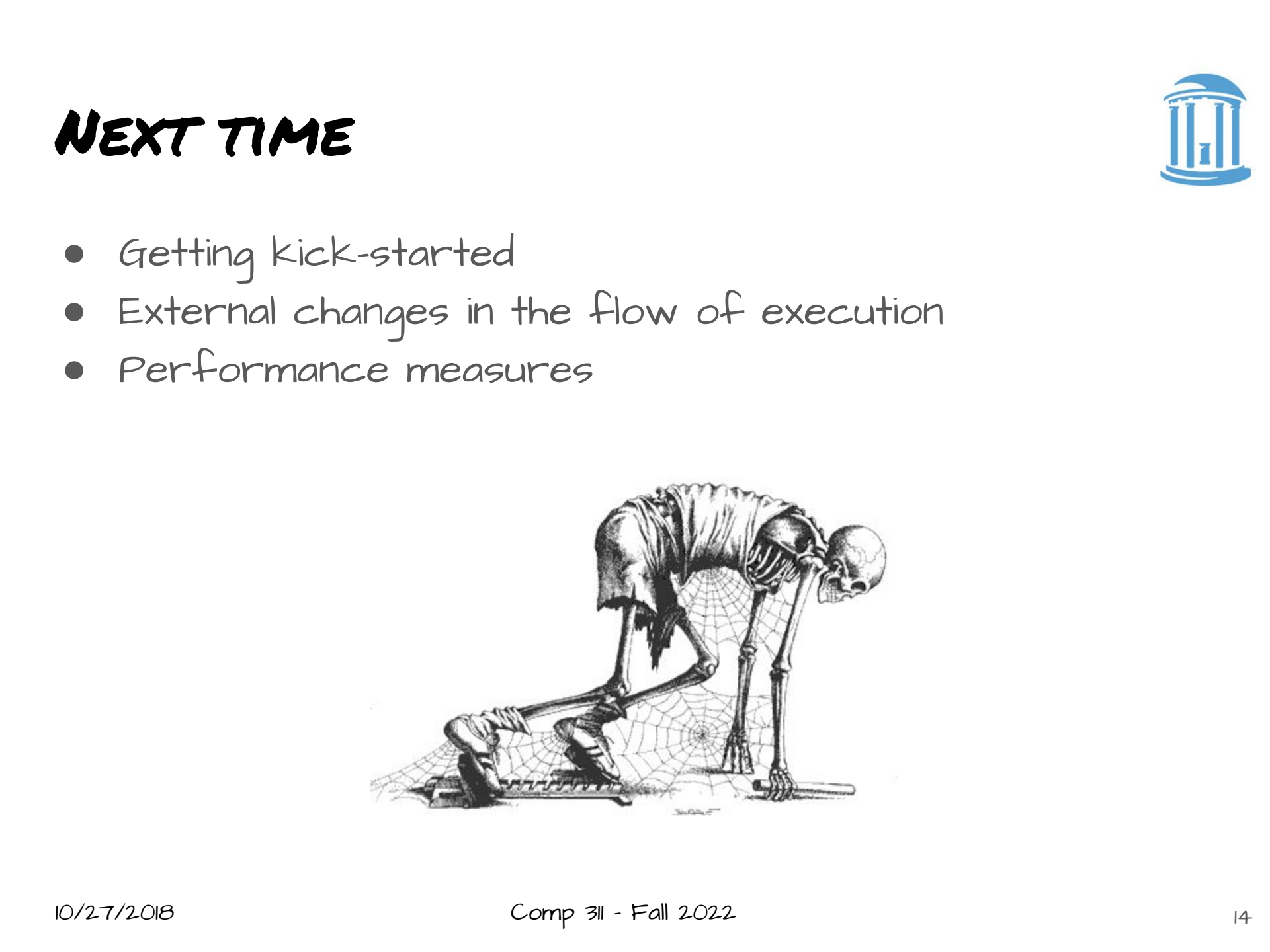
A MODIFIED ALU

Math	Shift	Sub	Set	R
0	0	X	X	f(A,B)
0	1	0	1	A << B
0	1	0	0	A >> B
0	1	1	0	A >>> B

Special support to generate the constants "0" and "1" needed by the SLT, SLTU, SLTI, SLTUI, etc instructions



Next time



- Getting kick-started
- External changes in the flow of execution
- Performance measures