More Assembly!

Check In 1 in class today and colloquium at HMC right after class (mask required)

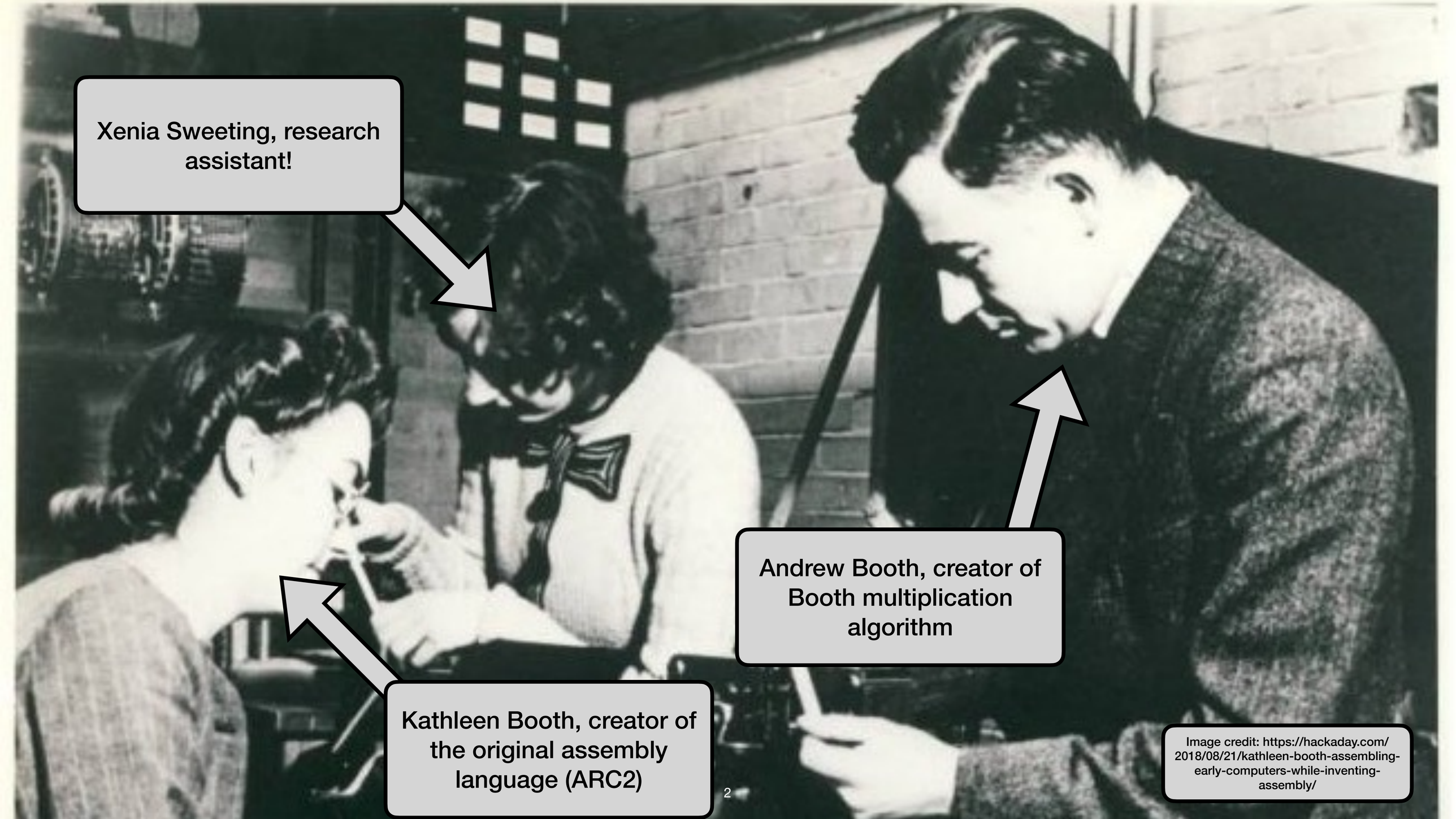


Image credit: https://hackaday.com/2018/08/21/kathleen-booth-assembling-early-computers-while-inventing-assembly/

Outline

- Revisiting Instruction Classes
- Interpreting “Immediates” in RISC-V instructions
- Check-In 1

Remember from last time…

- Opcodes dictate what instructions are used and how they should be interpreted
- Assembly instructions can generally be classified as one of…

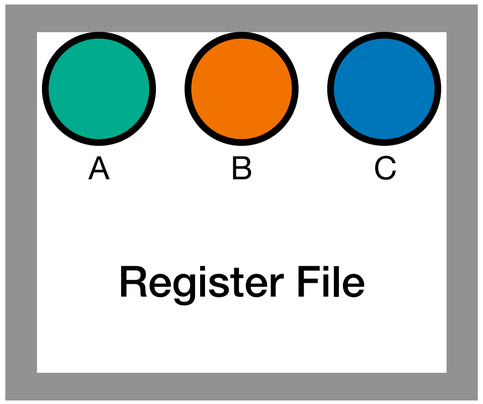
Data Transfers

Computations

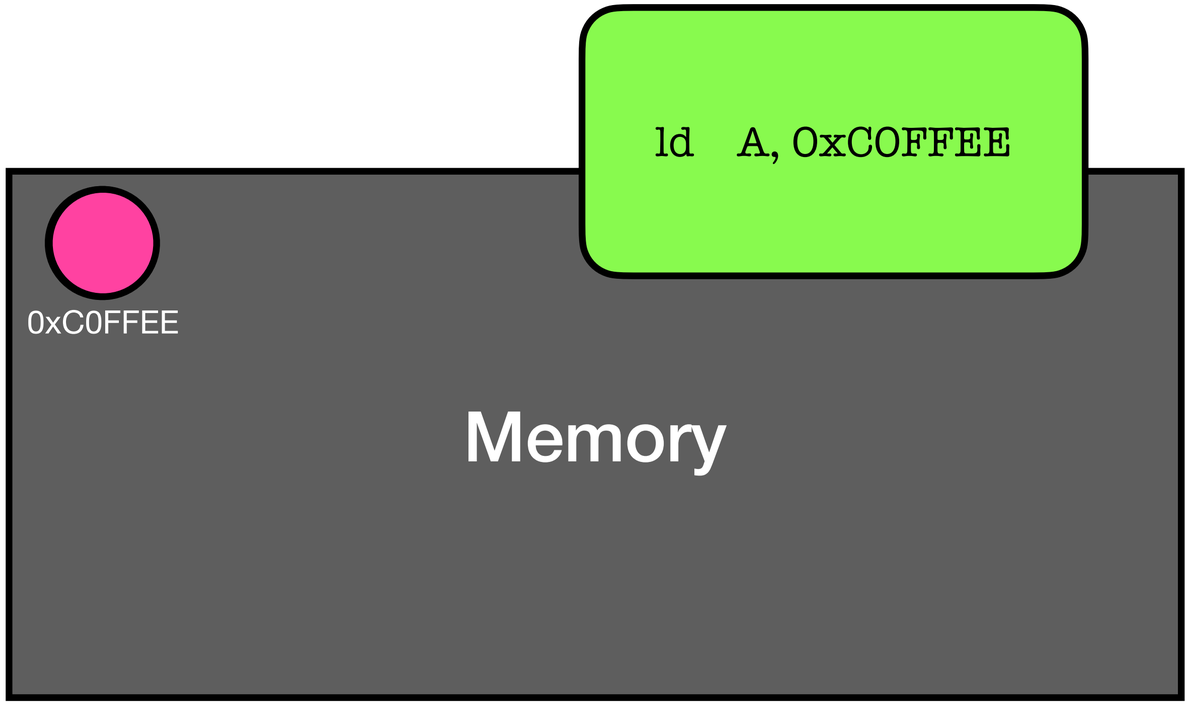
Control Logic

Data Transfers

- Moving data from memory to the register state or vice versa
- Data transfer clobbers previous state!

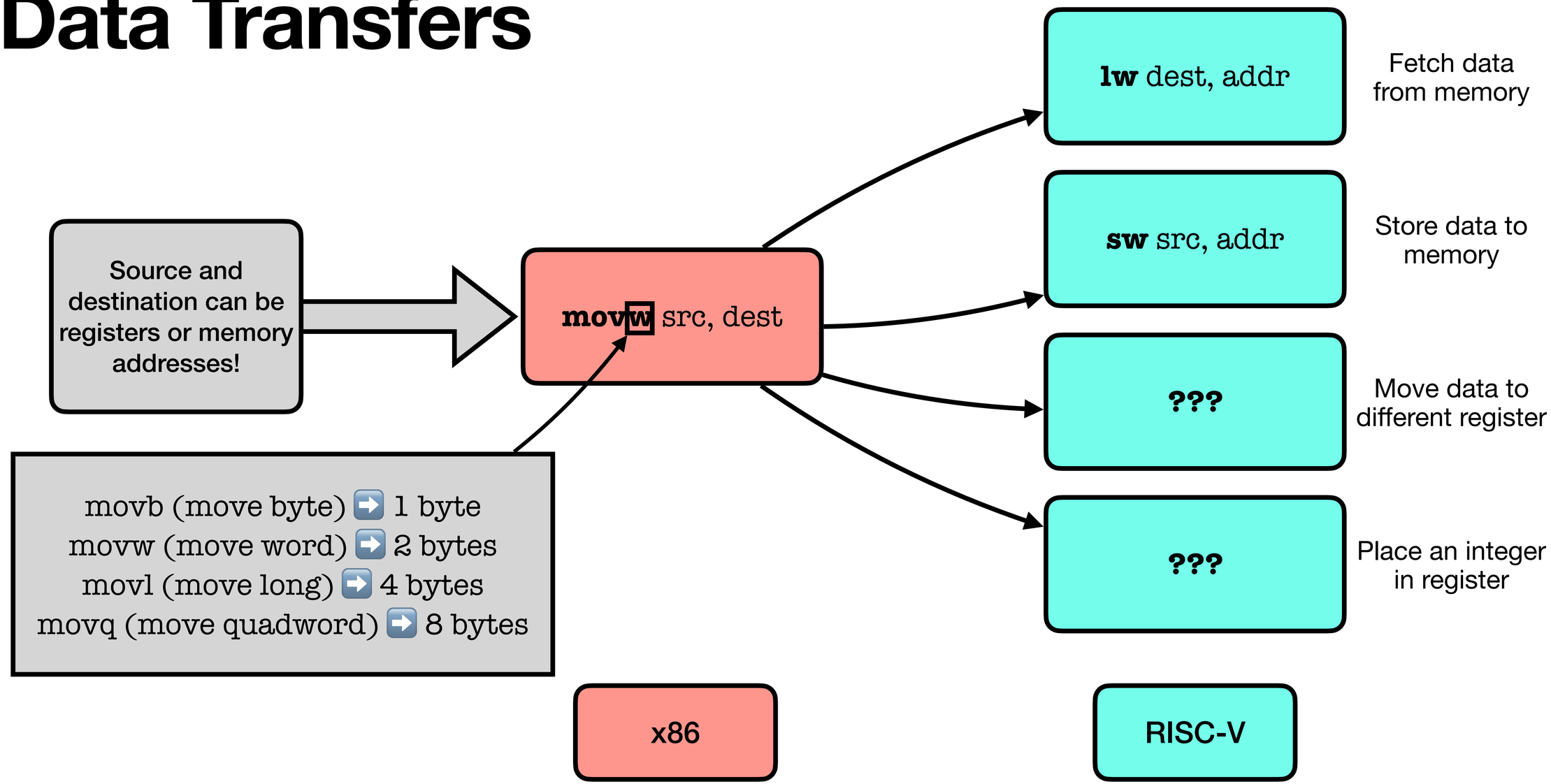


fast, limited capacity



slow, large capacity

Data Transfers



Computations

Integer in the raw instruction bytes

	x86	RISC-V (register-register)	RISC-V (register-immediate)
arithmetic	add, sub, ...	add, sub, ...	addi, ...
logical	and, or, xor	and, or, xor	andi, ori, xori
shift	shl, shr, sar, sal	sll, srl, sra	slli, srli, srai
comparison	cmp	slt[u]	slti[u]

Why no subi?

In RISC-V, `movw imm, dest` is expressed as `addi dest, x0, imm`!

Case Study: Immediates in RISC-V

31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

0xE5B18393

0b111001011011 x3 add x7 I-type

0b111001011011 00011 000 00111 0010011

Convert to binary!

Expand immediate to base 10!

Negative? **0b111001011011**

Flip! **0b000110100100**

Add 1! **0b000110100101**

addi x7, x3, -421

Case Study: Immediates in RISC-V

31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

imm [12|10:5] **0x52338263** imm [4:1|11]

x3 x7 beq I-type

0b01010010001100111000001001100011

Convert to binary!

Expand immediate to base 10!

TODO: as a class!

Control Logic

Jump to a certain part of the code if…

	x86	RISC-V (register-register)
unconditional	jmp	jal, jalr
Conditional	je, jne, js, jns, …	beq, bne, blt, …

Control Logic

- Conditional jumps: if the condition is true, then go to the destination specified by the instruction
  - For RISC-V, this is the current PC + the immediate
  - For x86, this is the absolute address in the instruction
- Otherwise, go to the current PC + the size of the instruction
  - For RISC-V, this is PC + 4 (B-type instructions are 4 bytes long)
  - For x86, this is dependent on the instruction size

Takeaways

- High level language tokens can be described as “data transfers,” “computations,” or “control logic”
- The processor uses expected formats to interpret instruction bytes in the same way as the decoded opcode