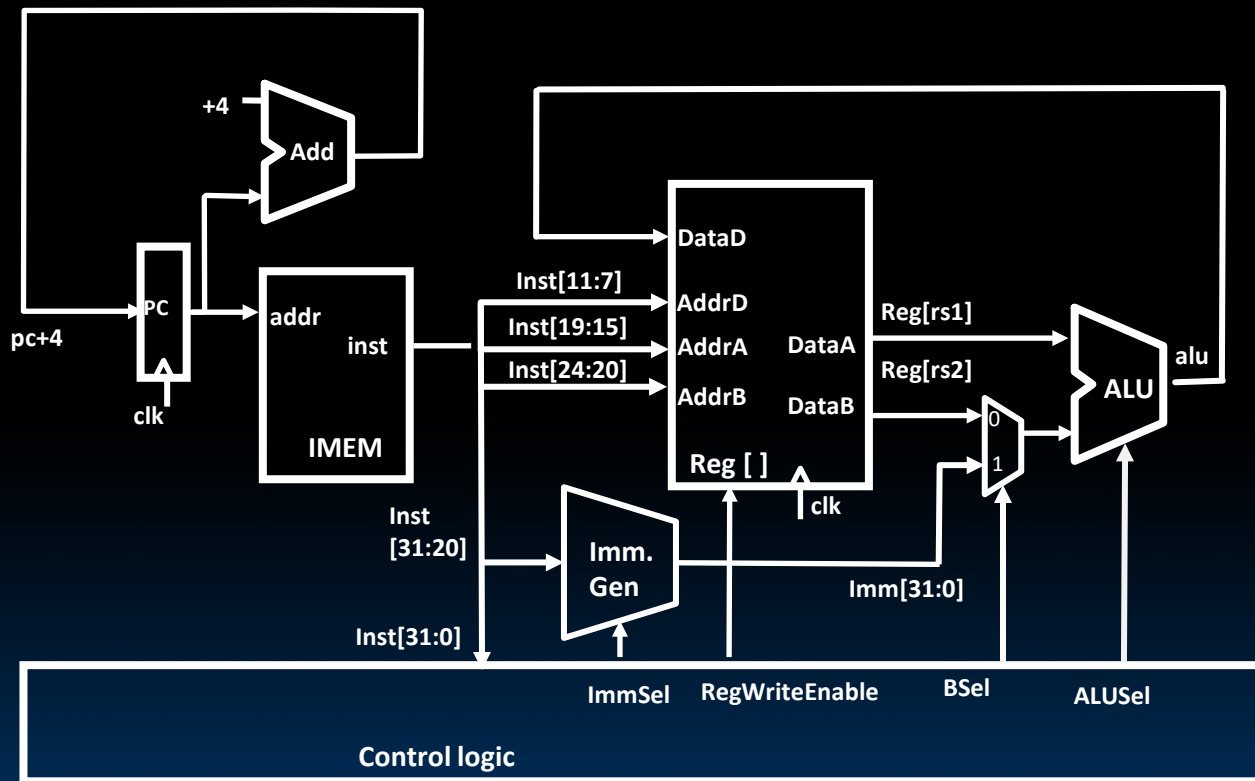


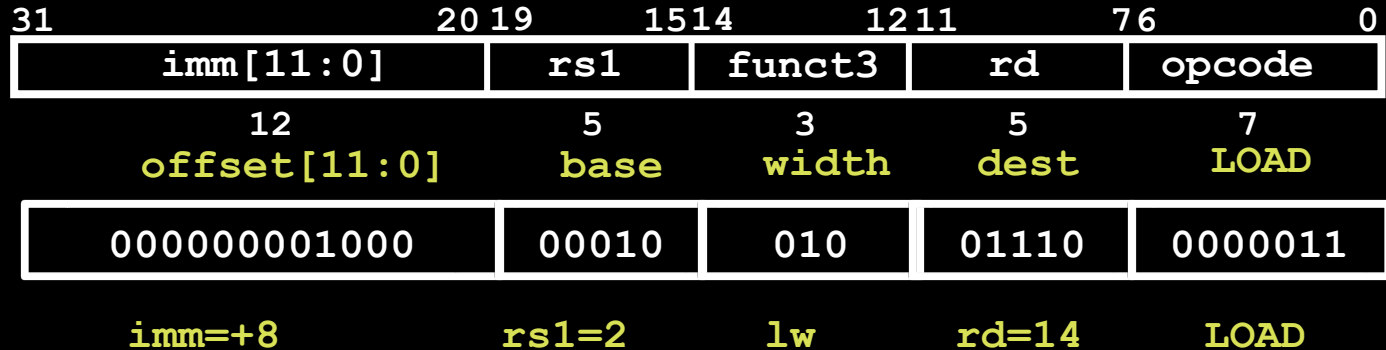
# Supporting Loads

# R+I Arithmetic/Logic Datapath



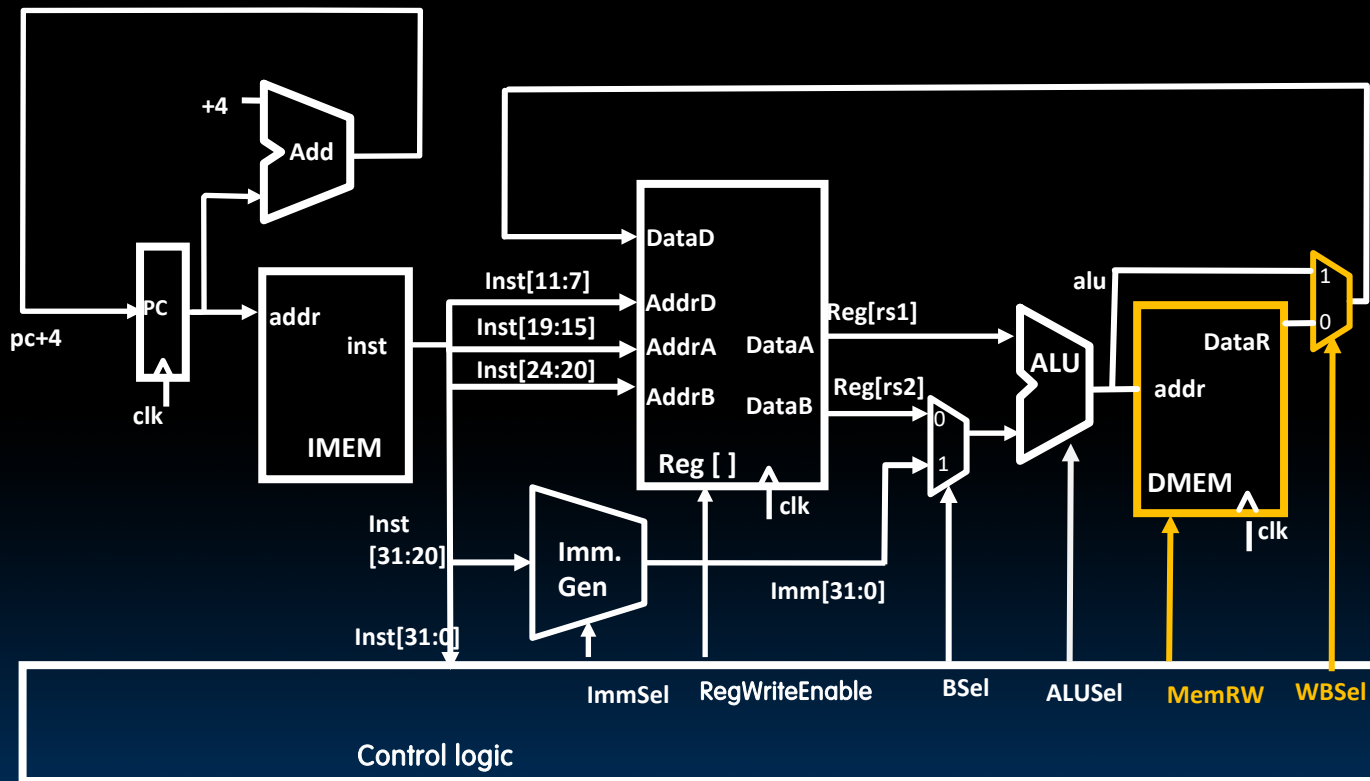
# Add lw

- RISC-V Assembly Instruction (I-type): **lw x14, 8(x2)**

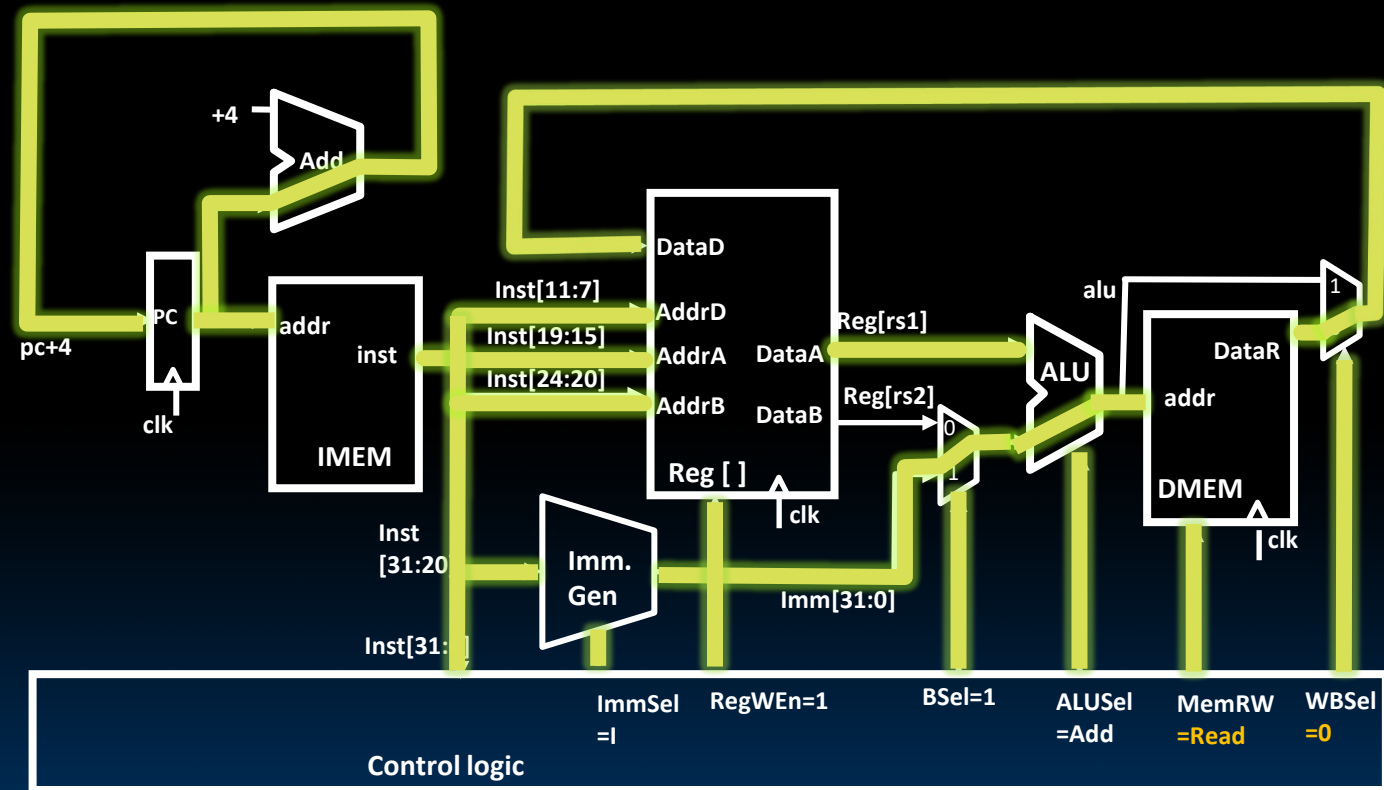


- The 12-bit signed immediate is added to the base address in register **rs1** to form the **memory** address
  - This is very similar to the add-immediate operation but used to create address not to create final result
- The value loaded from **memory** is stored in register **rd**

# R+I Arithmetic/Logic Datapath



# R+I Arithmetic/Logic Datapath



# All RV32 Load Instructions

|                        |                  |                  |                 |                      |            |
|------------------------|------------------|------------------|-----------------|----------------------|------------|
| <code>imm[11:0]</code> | <code>rs1</code> | <code>000</code> | <code>rd</code> | <code>0000011</code> | <b>lb</b>  |
| <code>imm[11:0]</code> | <code>rs1</code> | <code>001</code> | <code>rd</code> | <code>0000011</code> | <b>lh</b>  |
| <code>imm[11:0]</code> | <code>rs1</code> | <code>010</code> | <code>rd</code> | <code>0000011</code> | <b>lw</b>  |
| <code>imm[11:0]</code> | <code>rs1</code> | <code>100</code> | <code>rd</code> | <code>0000011</code> | <b>lbu</b> |
| <code>imm[11:0]</code> | <code>rs1</code> | <code>101</code> | <code>rd</code> | <code>0000011</code> | <b>lhu</b> |

↑ funct3 field encodes size and 'signedness' of load data

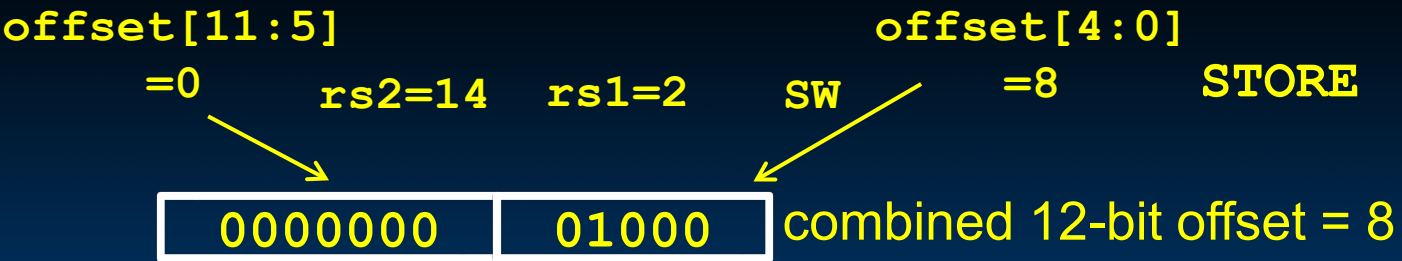
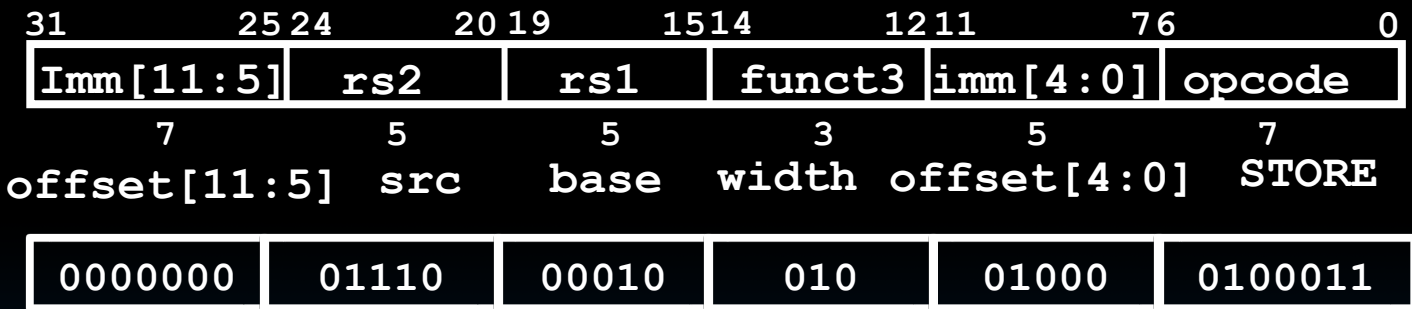
- Supporting the narrower loads requires additional logic to extract the correct byte/halfword from the value loaded from memory, and sign- or zero-extend the result to 32 bits before writing back to register file.
  - It is just a mux + a few gates

# Datapath for Stores

# Adding sw instruction

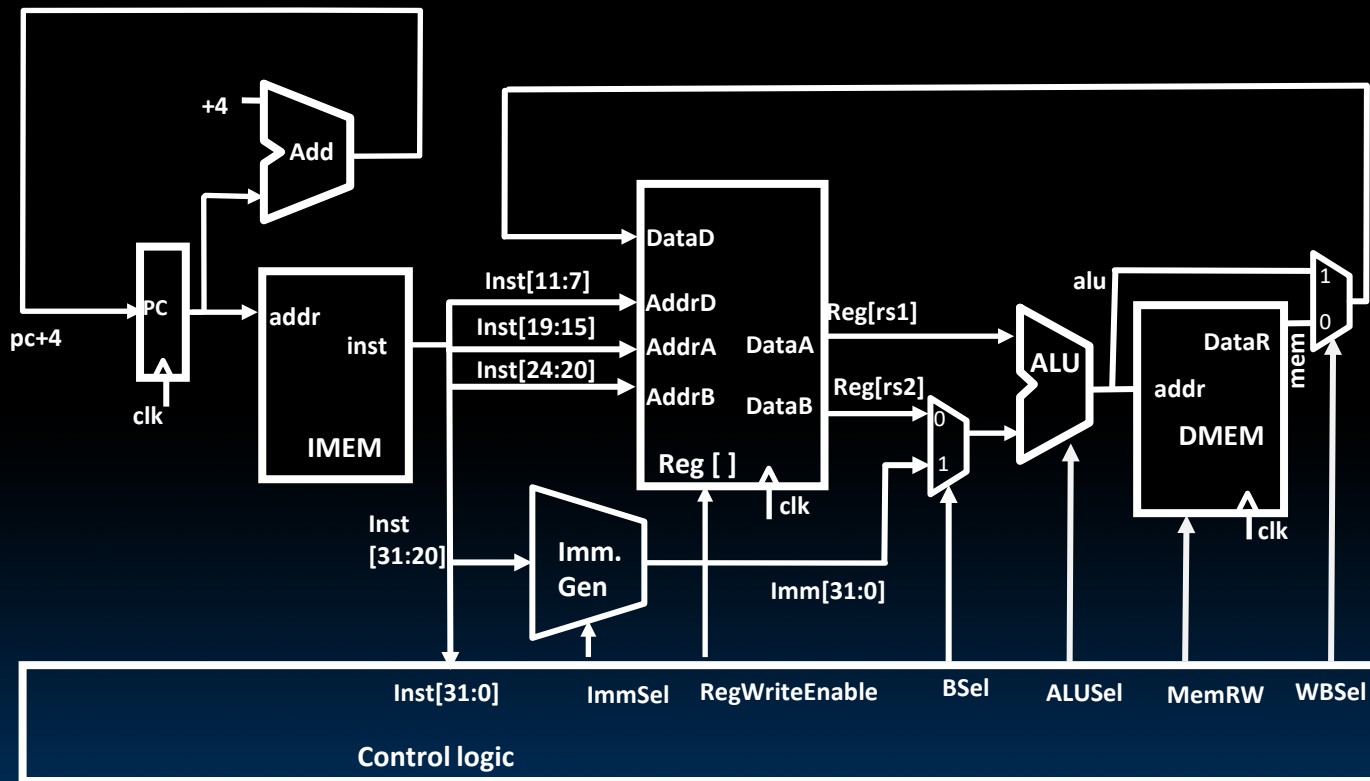
- **sw**: Reads two registers, rs1 for base memory address, and rs2 for data to be stored, as well immediate offset!

**sw x14, 8(x2)**

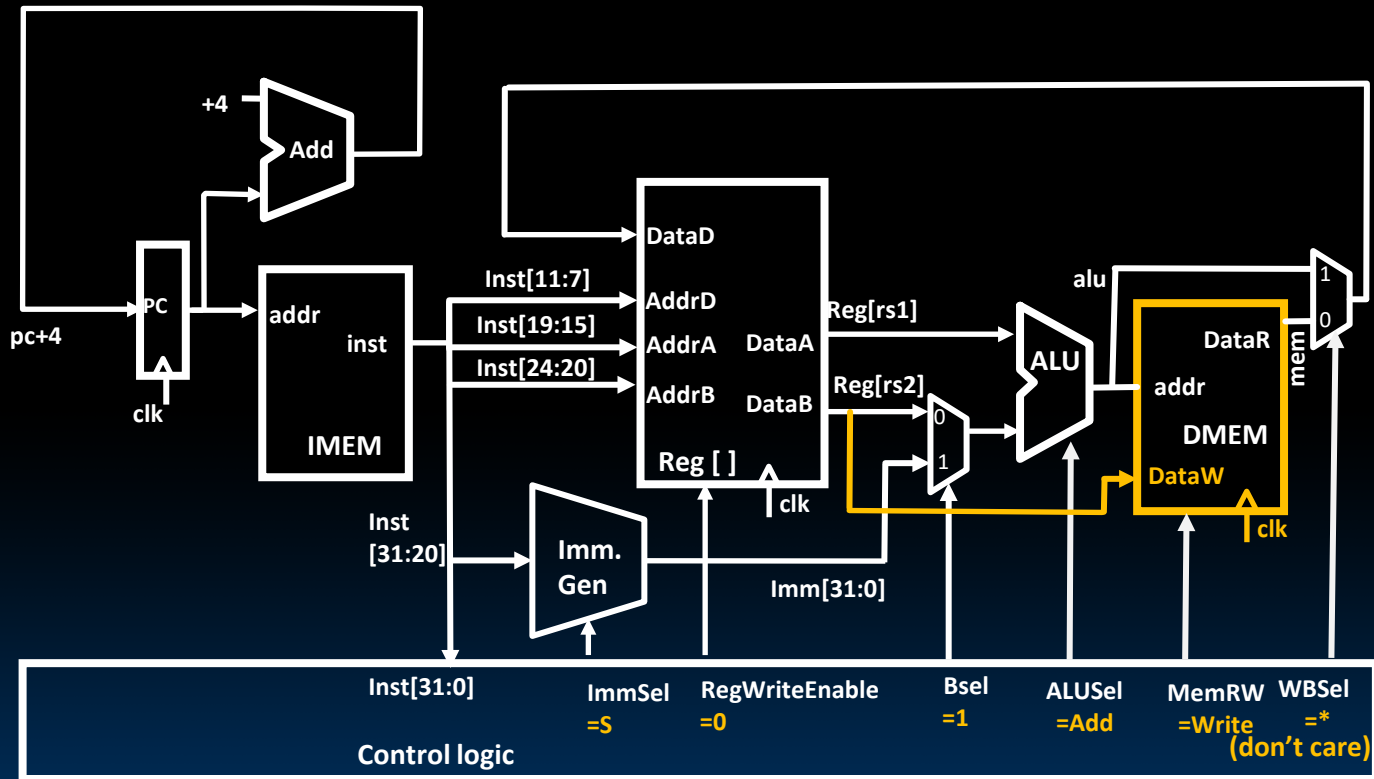




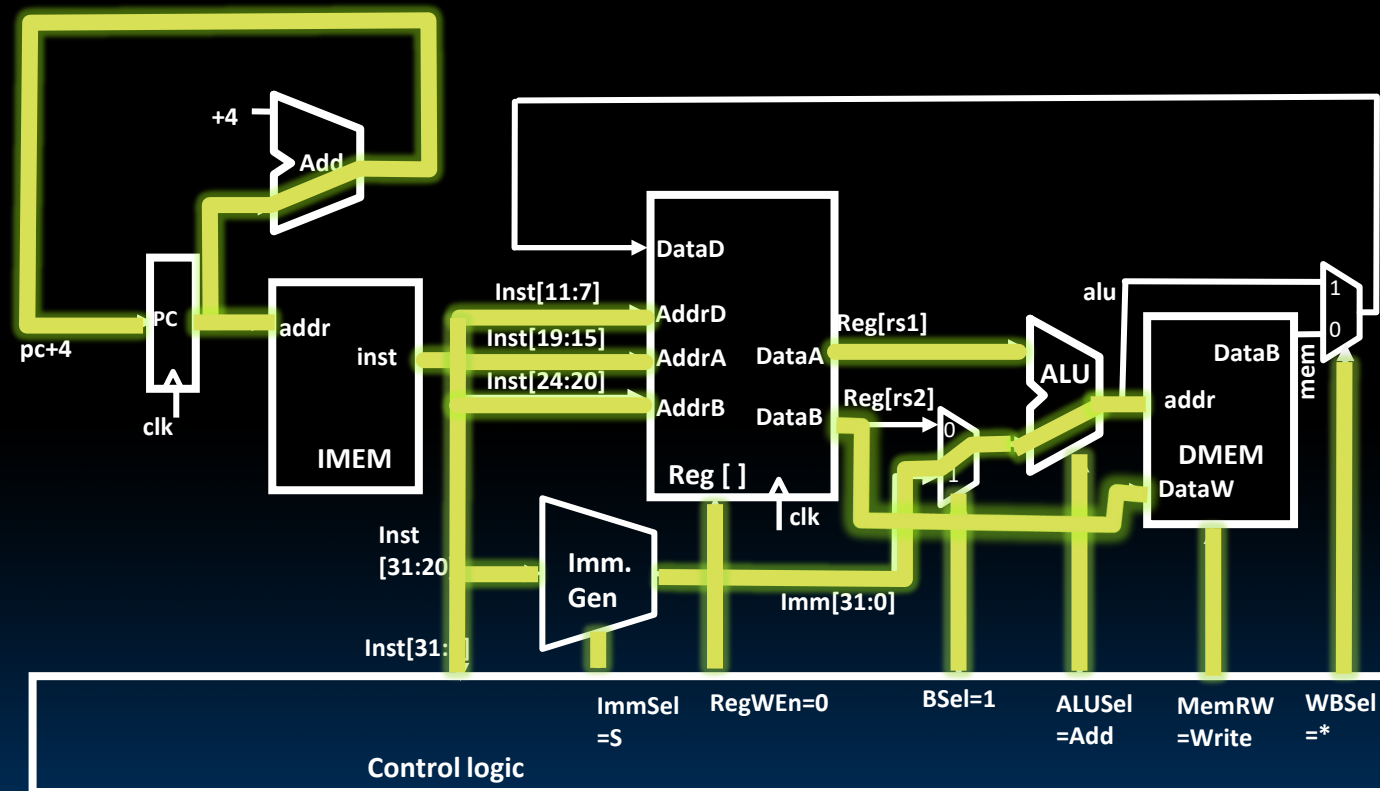
# Datapath with 1w



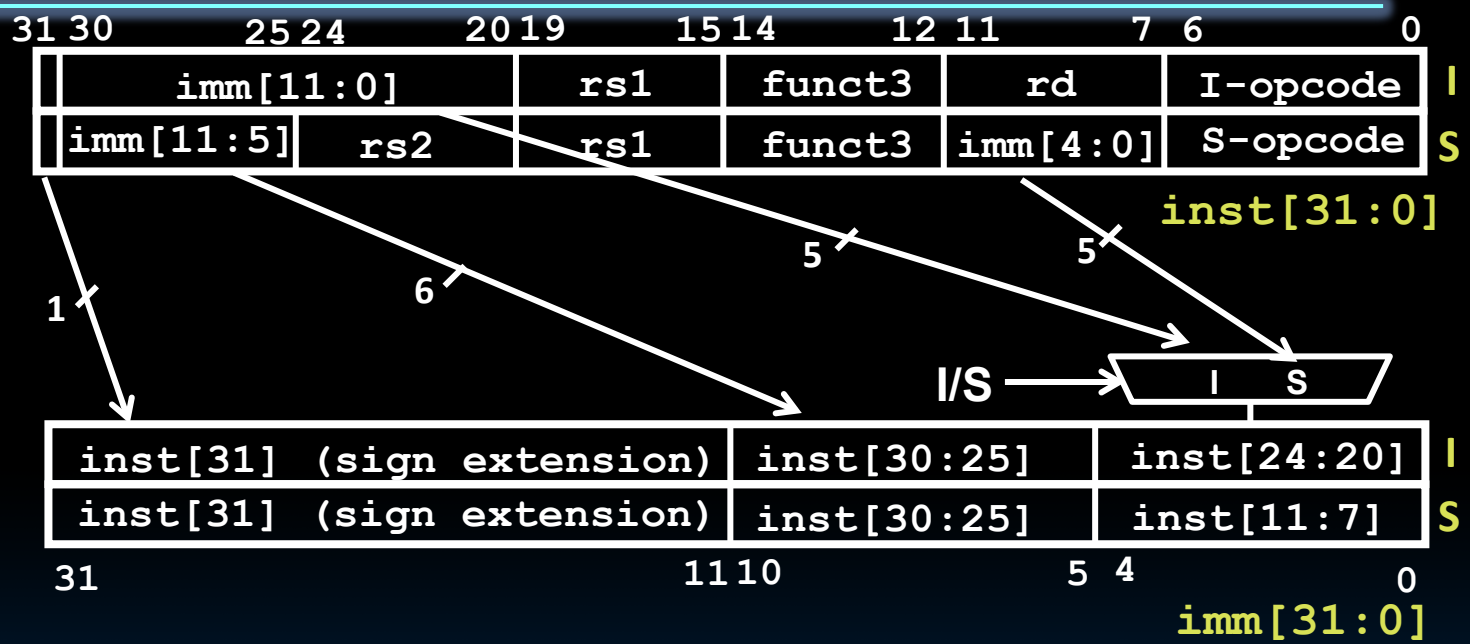
# Adding sw to Datapath



# Adding sw to Datapath



# I+S Immediate Generation



- Just need a 5-bit mux to select between two positions where low five bits of immediate can reside in instruction
- Other bits in immediate are wired to fixed positions in instruction

# All RV32 Store Instructions

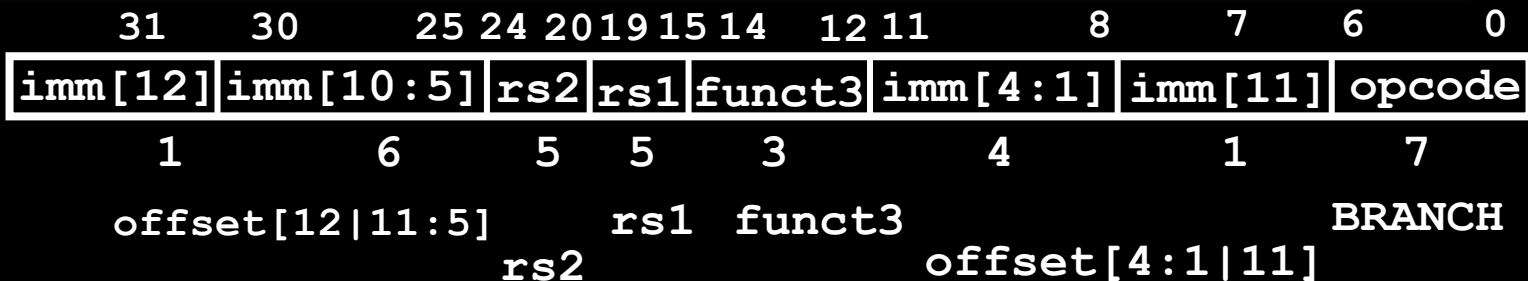
- Store byte writes the low byte to memory
- Store halfword writes the lower two bytes to memory

|           |     |     |     |          |         |           |
|-----------|-----|-----|-----|----------|---------|-----------|
| Imm[11:5] | rs2 | rs1 | 000 | imm[4:0] | 0100011 | <b>sb</b> |
| Imm[11:5] | rs2 | rs1 | 001 | imm[4:0] | 0100011 | <b>sh</b> |
| Imm[11:5] | rs2 | rs1 | 010 | imm[4:0] | 0100011 | <b>sw</b> |

**width**

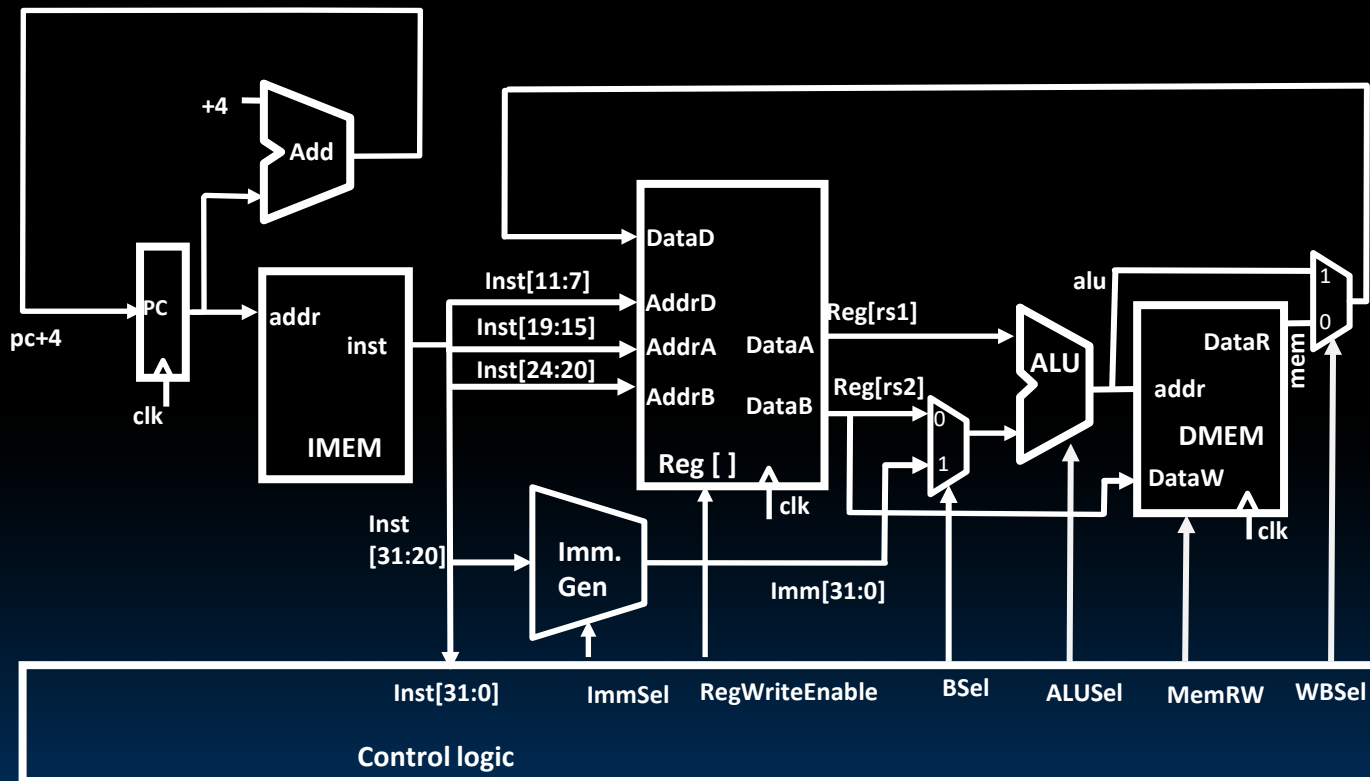
# Implementing Branches

# RISC-V B-Format for Branches



- B-format is mostly same as S-Format, with two register sources (**rs1/rs2**) and a 12-bit immediate **imm[12:1]**
- But now immediate represents values -4096 to +4094 in 2-byte increments
- The 12 immediate bits encode **even** 13-bit signed byte offsets (lowest bit of offset is always zero, so no need to store it)

# Datapath So Far

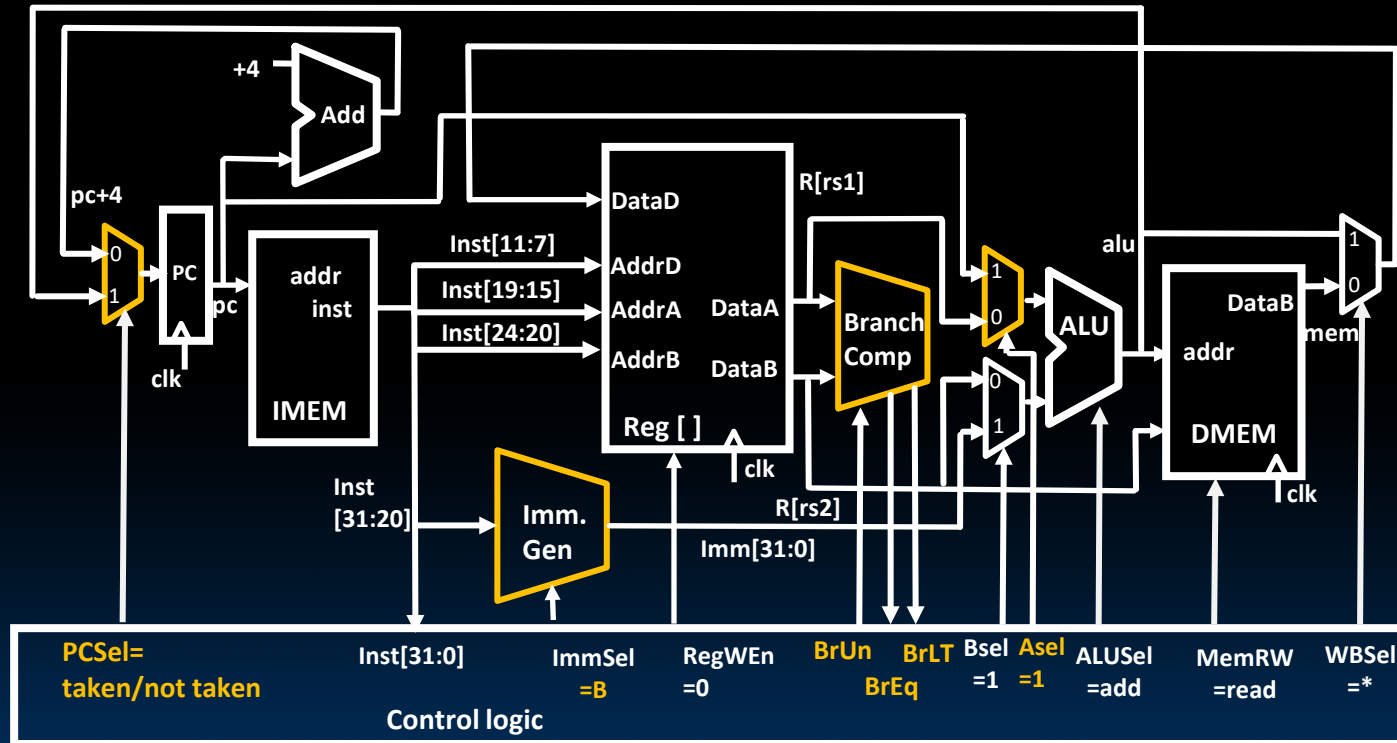




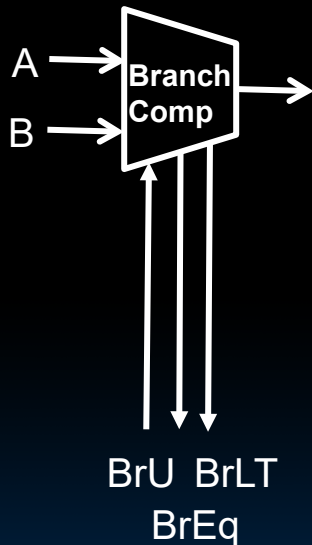
# To Add Branches

- Different change to the state:
 
$$PC = \begin{cases} PC + 4, & \text{branch not taken} \\ PC + \text{immediate}, & \text{branch taken} \end{cases}$$
- Six branch instructions: **beq**, **bne**, **blt**, **bge**, **bltu**, **bgeu**
- Need to compute **PC + immediate** and to compare values of **rs1** and **rs2**
  - But have only one ALU – need more hardware

# Adding Branches



# Branch Comparator



$\text{BrEq} = 1, \text{ if } A=B$

$\text{BrLT} = 1, \text{ if } A<B$

$\text{BrUn} = 1$  selects unsigned comparison for  $\text{BrLT}$ ,  
0=signed

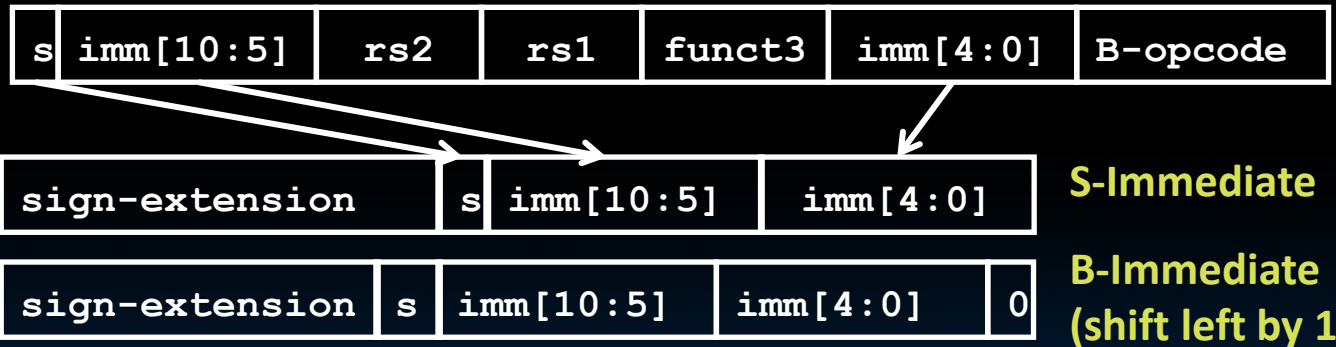
$\overline{\text{BGE}}$  branch:  $A \geq B$ , if  $\overline{A<B}$

$\overline{A<B} = \neg(A<B)$

# Branch Immediates (In Other ISAs)

12-bit immediate encodes PC-relative offset of -4096 to +4094 bytes in multiples of 2 bytes

Standard approach: Treat immediate as in range -2048..+2047, then shift left by 1 bit to multiply by 2 for branches



Each instruction immediate bit can appear in one of two places in output immediate value – so need one 2-way mux per bit

# Branch Immediates (In Other ISAs)

12-bit immediate encodes PC-relative offset of -4096 to +4094 bytes in multiples of 2 bytes

RISC-V approach: keep 11 immediate bits in fixed position in output value

|              |           |          |
|--------------|-----------|----------|
| sign=imm[11] | imm[10:5] | imm[4:0] |
|--------------|-----------|----------|

**S-Immediate**

|              |         |           |          |   |
|--------------|---------|-----------|----------|---|
| sign=imm[12] | imm[11] | imm[10:5] | imm[4:1] | 0 |
|--------------|---------|-----------|----------|---|

**B-Immediate**  
(shift left by 1)

Only one bit changes position between S and B, so only need a single-bit 2-way mux

# RISC-V Immediate Encoding

## Instruction encodings, inst[31:0]

| 31           | 30 | 25 | 24 | 20  | 19  | 15  | 14     | 12     | 11 | 8           | 7 | 6      | 0      |        |        |
|--------------|----|----|----|-----|-----|-----|--------|--------|----|-------------|---|--------|--------|--------|--------|
| 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 |

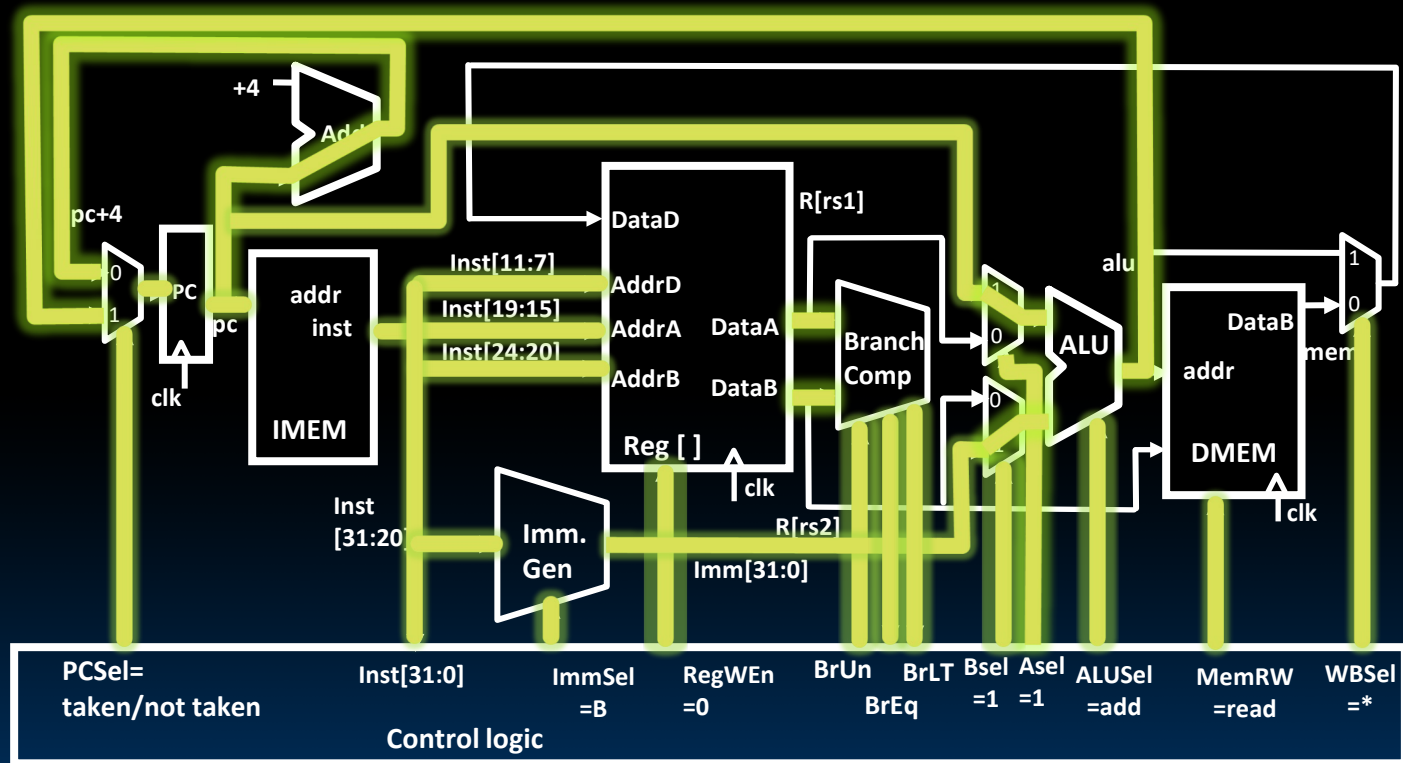
## 32-bit immediates produced, imm[31:0]

| 31         | 25 | 24 | 12 | 11      | 10          | 5 | 4           | 1 | 0        |        |
|------------|----|----|----|---------|-------------|---|-------------|---|----------|--------|
| -inst[31]- |    |    |    |         | inst[30:25] |   | inst[24:21] |   | inst[20] | I-imm. |
| -inst[31]- |    |    |    |         | inst[30:25] |   | inst[11:8]  |   | inst[7]  | S-imm. |
| -inst[31]- |    |    |    | inst[7] | inst[30:25] |   | inst[11:8]  |   | 0        | B-imm. |

Upper bits sign-extended from inst[31] always

Only bit 7 of instruction changes role in immediate between S and B

# Lighting Up Branch Path

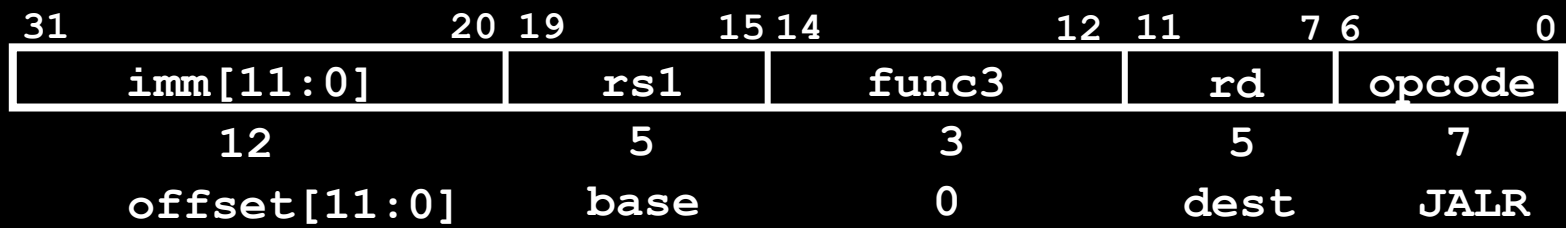


# Adding JALR to Datapath



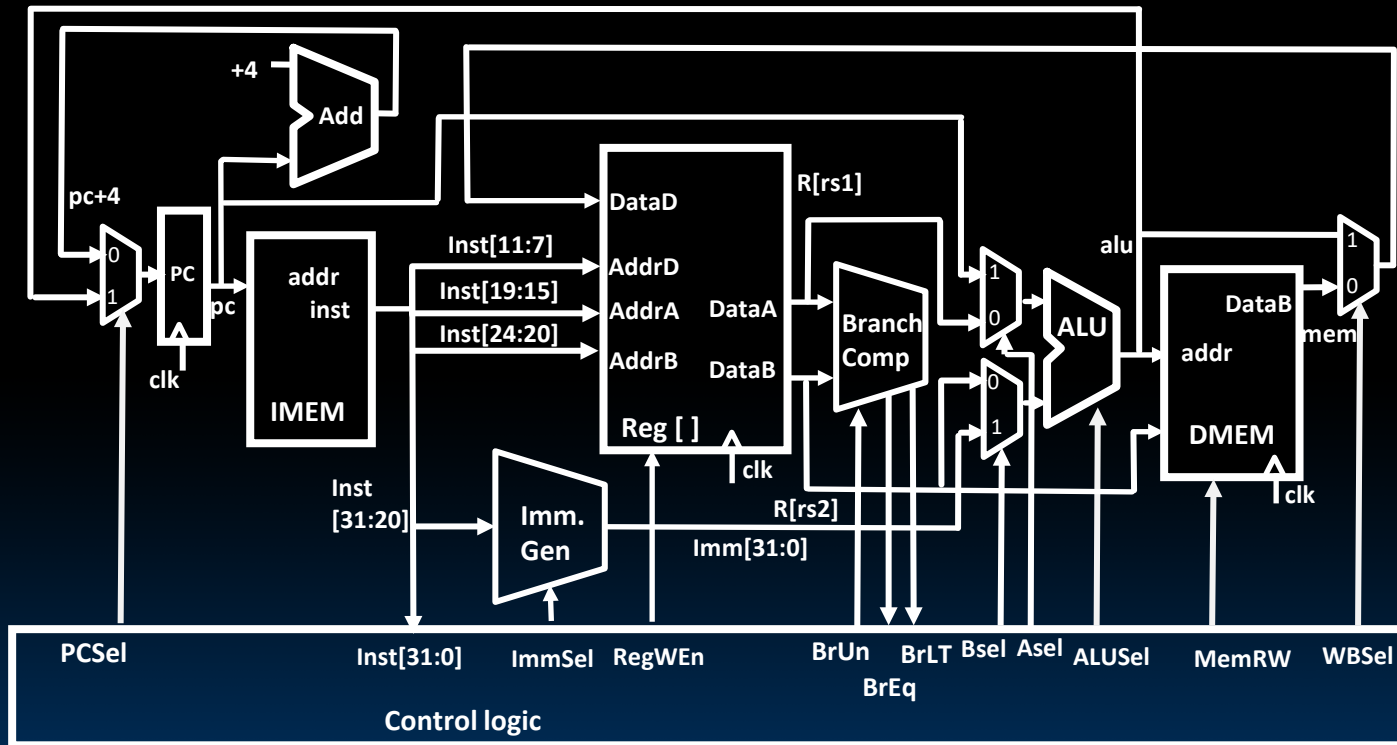


# Let's Add JALR (I-Format)

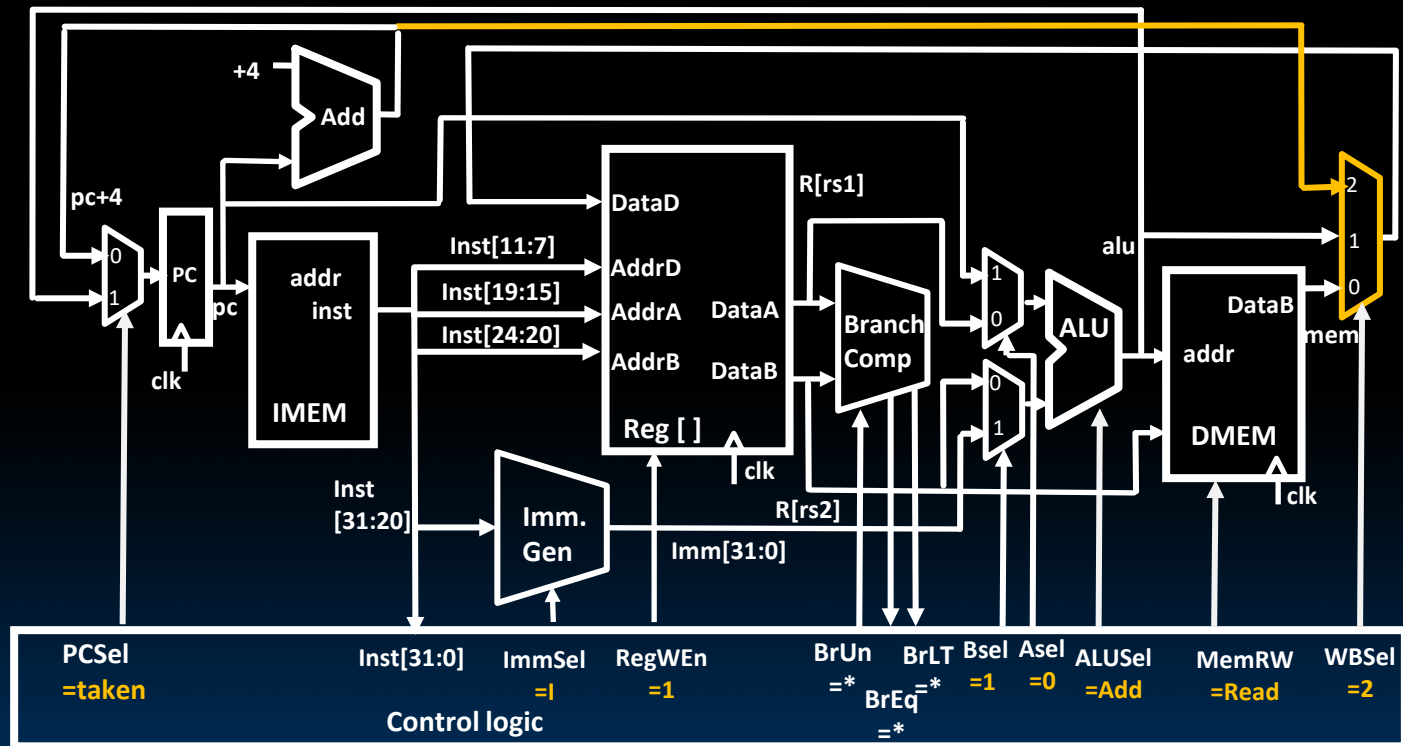


- JALR rd, rs, immediate
- Two changes to the state
  - Writes PC+4 to **rd** (return address)
  - Sets PC = **rs1** + **immediate**
  - Uses same immediates as arithmetic and loads
    - **no** multiplication by 2 bytes
    - LSB is ignored

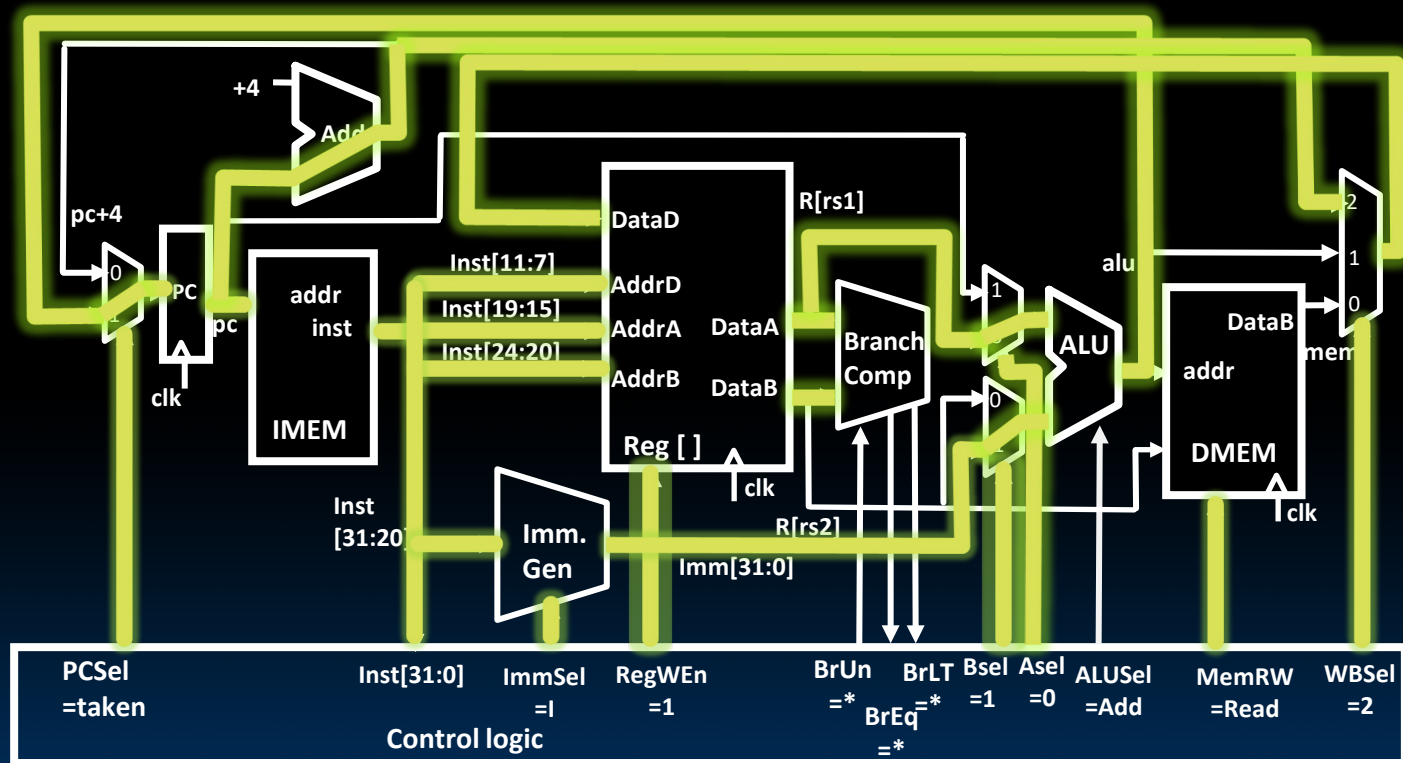
# Datapath So Far, With Branches



# Datapath With JALR



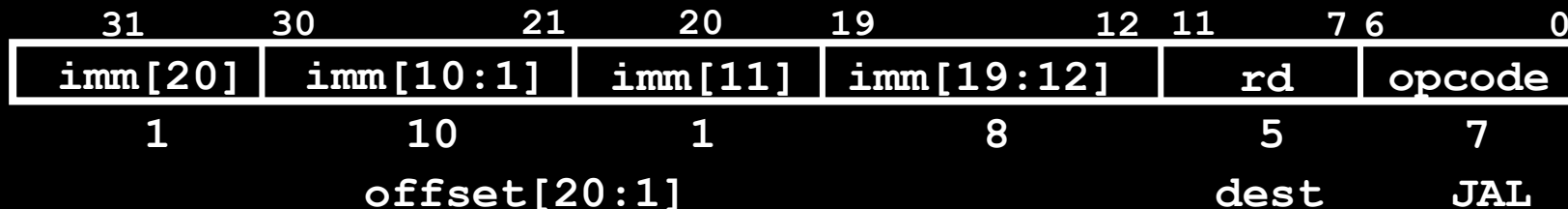
# Datapath With JALR



Adding JAI

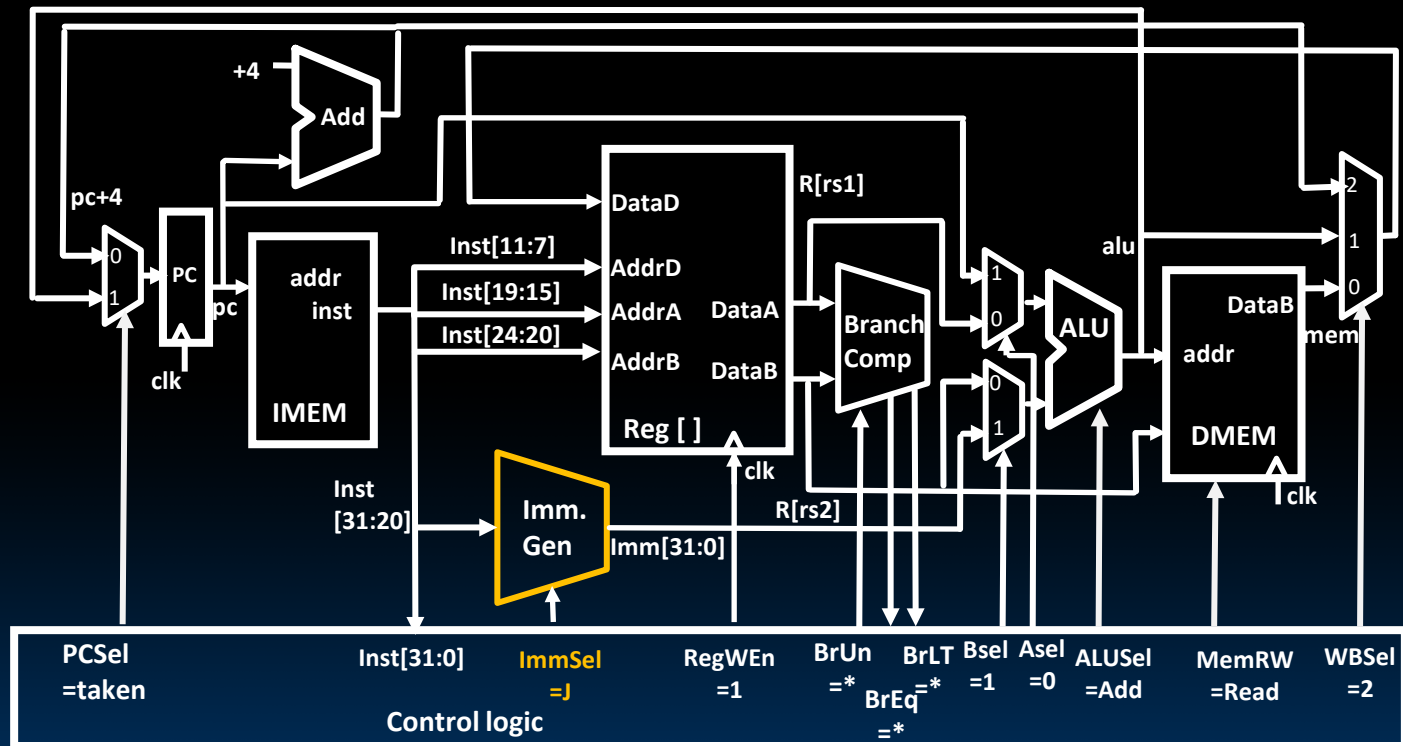


# J-Format for Jump Instructions

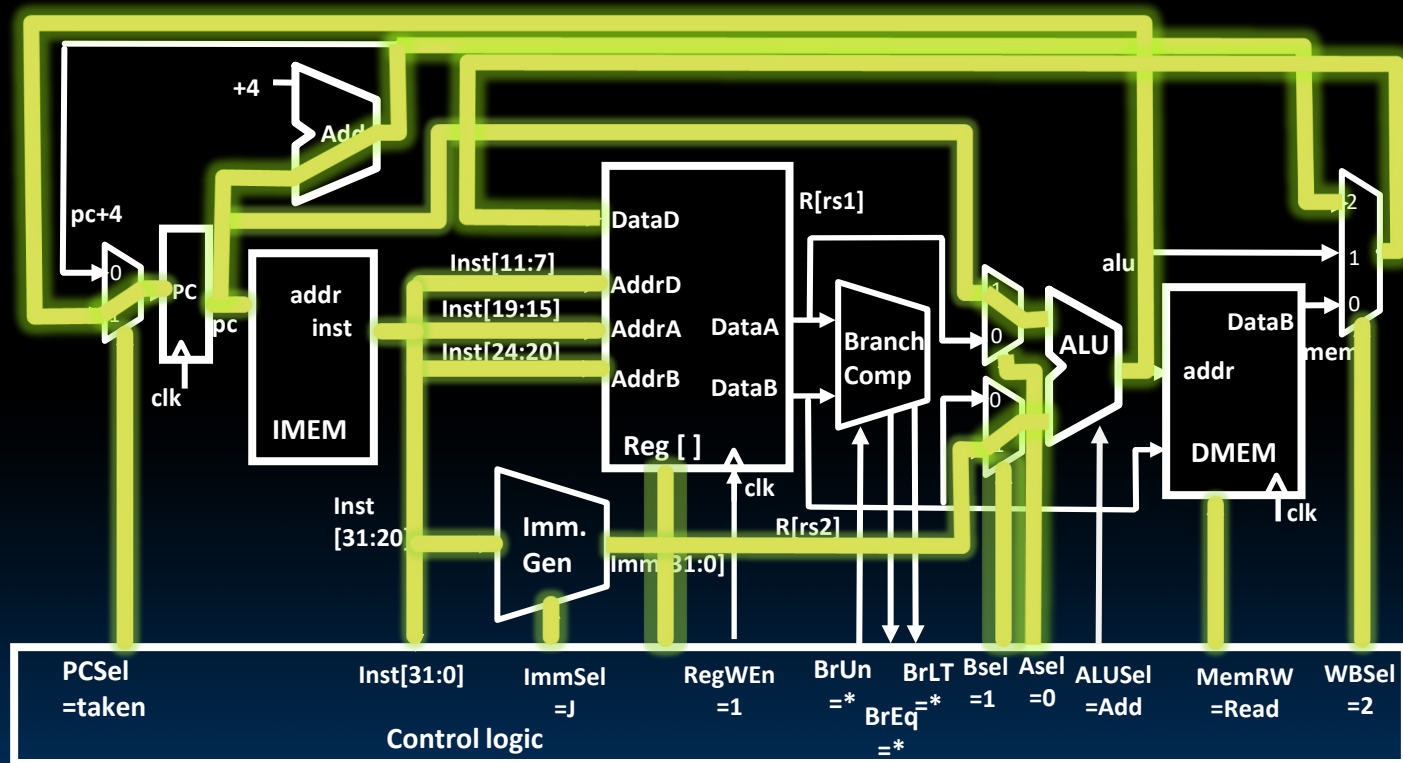


- Two changes to the state
  - **j<sub>al</sub>** saves PC+4 in register **rd** (the return address)
  - Set PC = PC + offset (PC-relative jump)
- Target somewhere within  $\pm 2^{19}$  locations, 2 bytes apart
  - $\pm 2^{18}$  32-bit instructions
- Immediate encoding optimized similarly to branch instruction to reduce hardware cost

# Datapath with JAL



# Light Up JAL Path

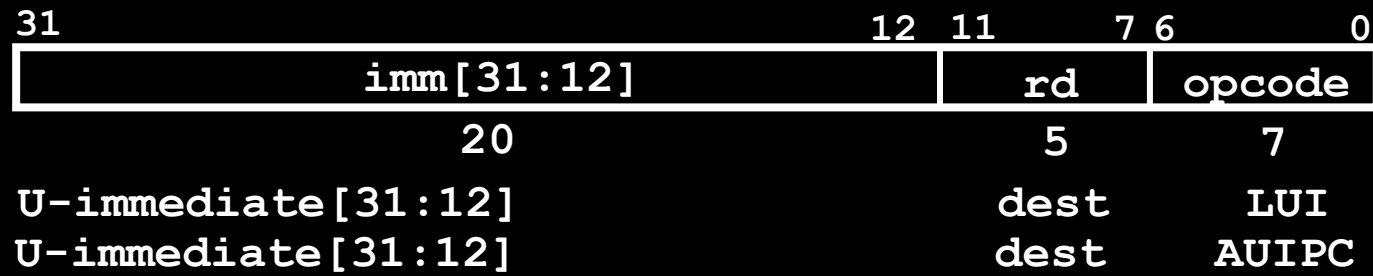




# Adding U-Types

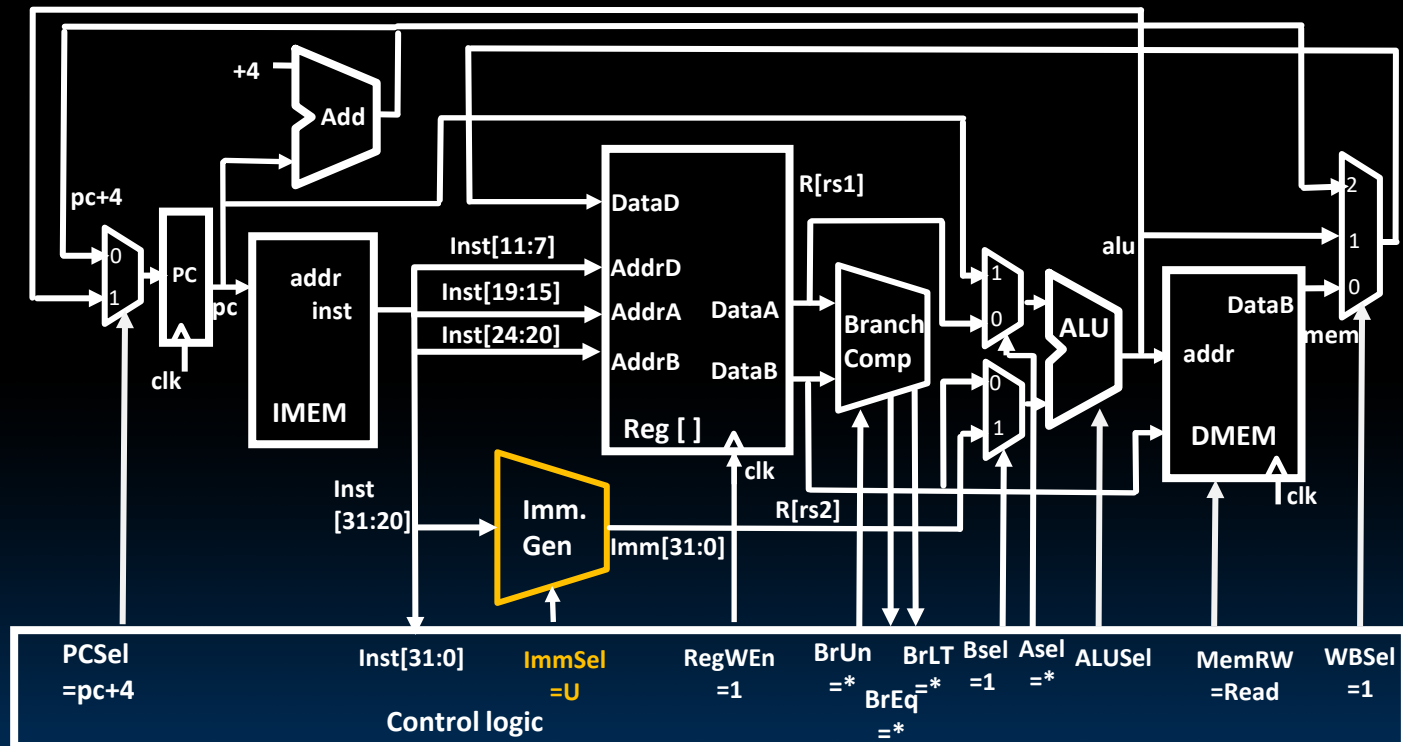


# U-Format for “Upper Immediate” Instructions

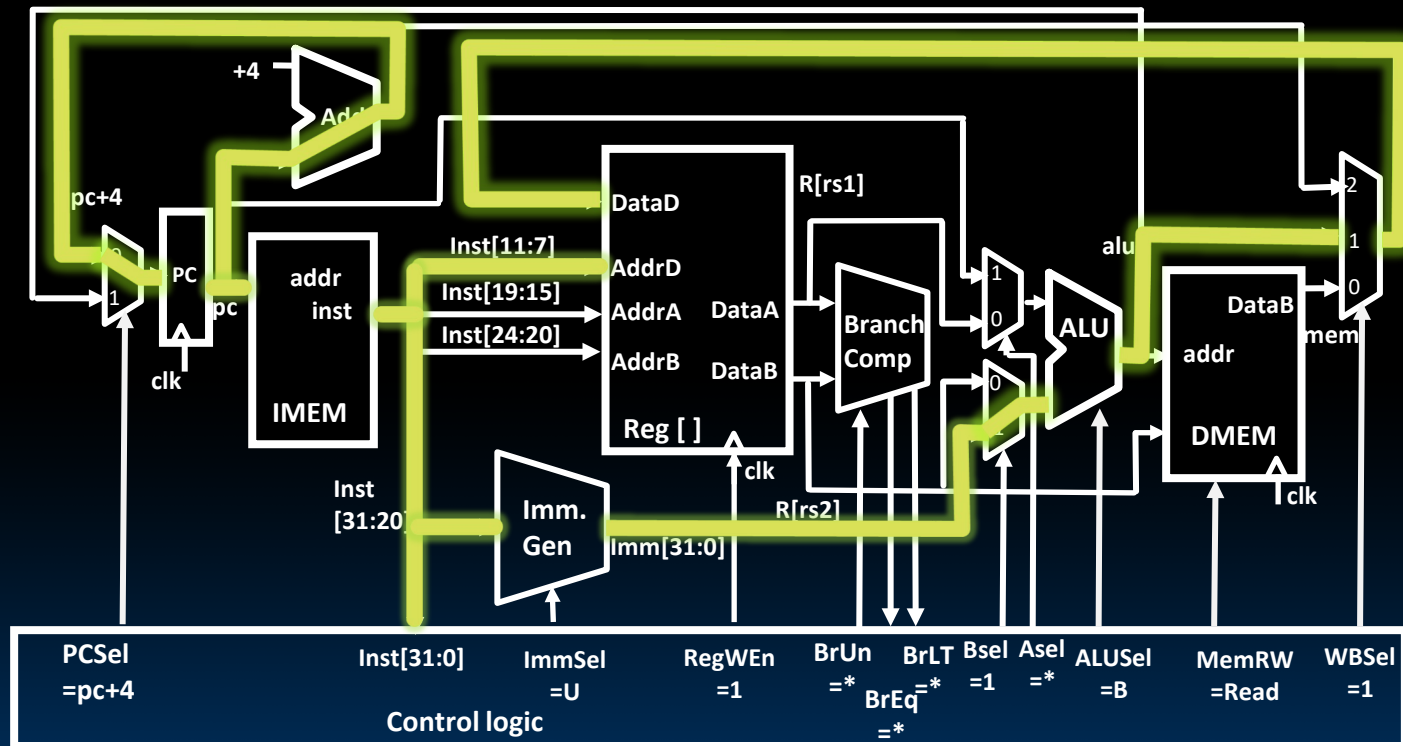


- Has 20-bit immediate in upper 20 bits of 32-bit instruction word
- One destination register, **rd**
- Used for two instructions
  - **lui** – Load Upper Immediate
  - **auipc** – Add Upper Immediate to PC

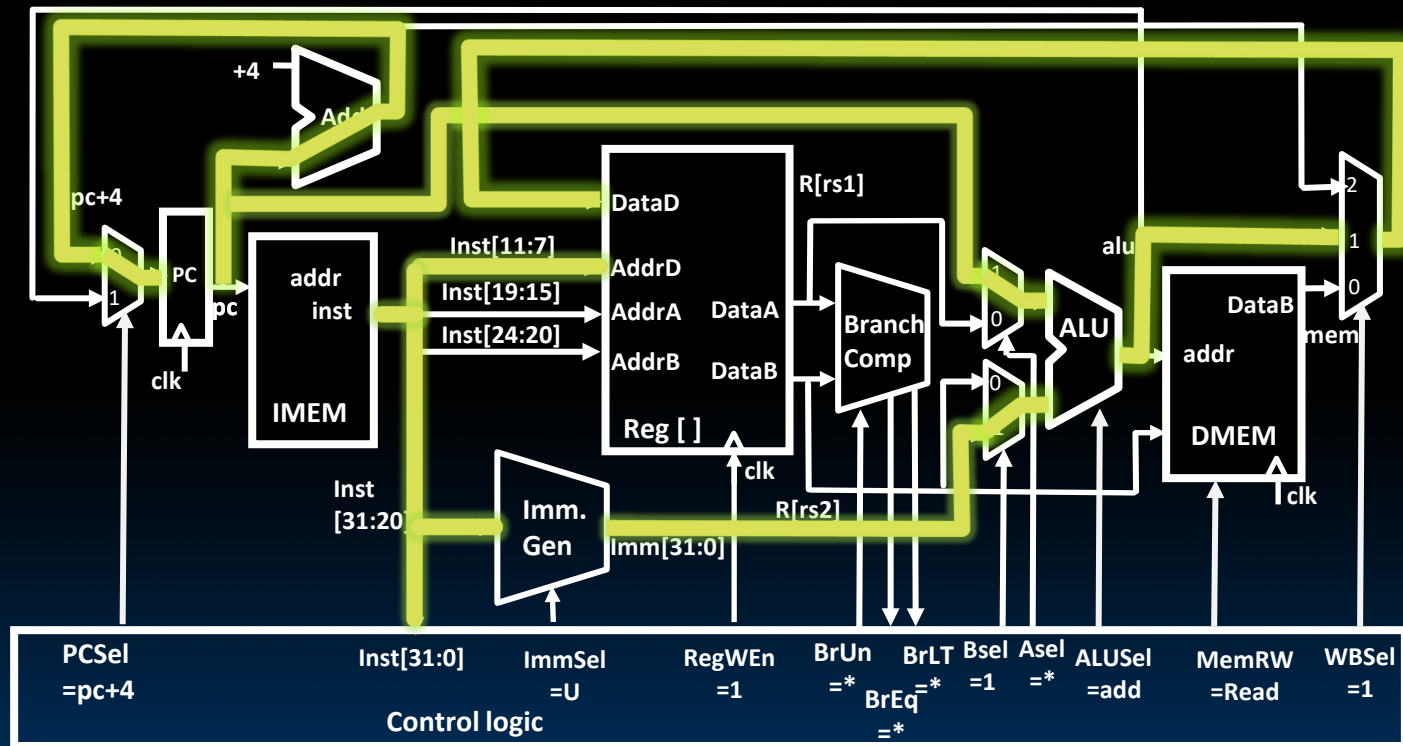
# Datapath With LUI, AUIPC



# Lighting Up LUI

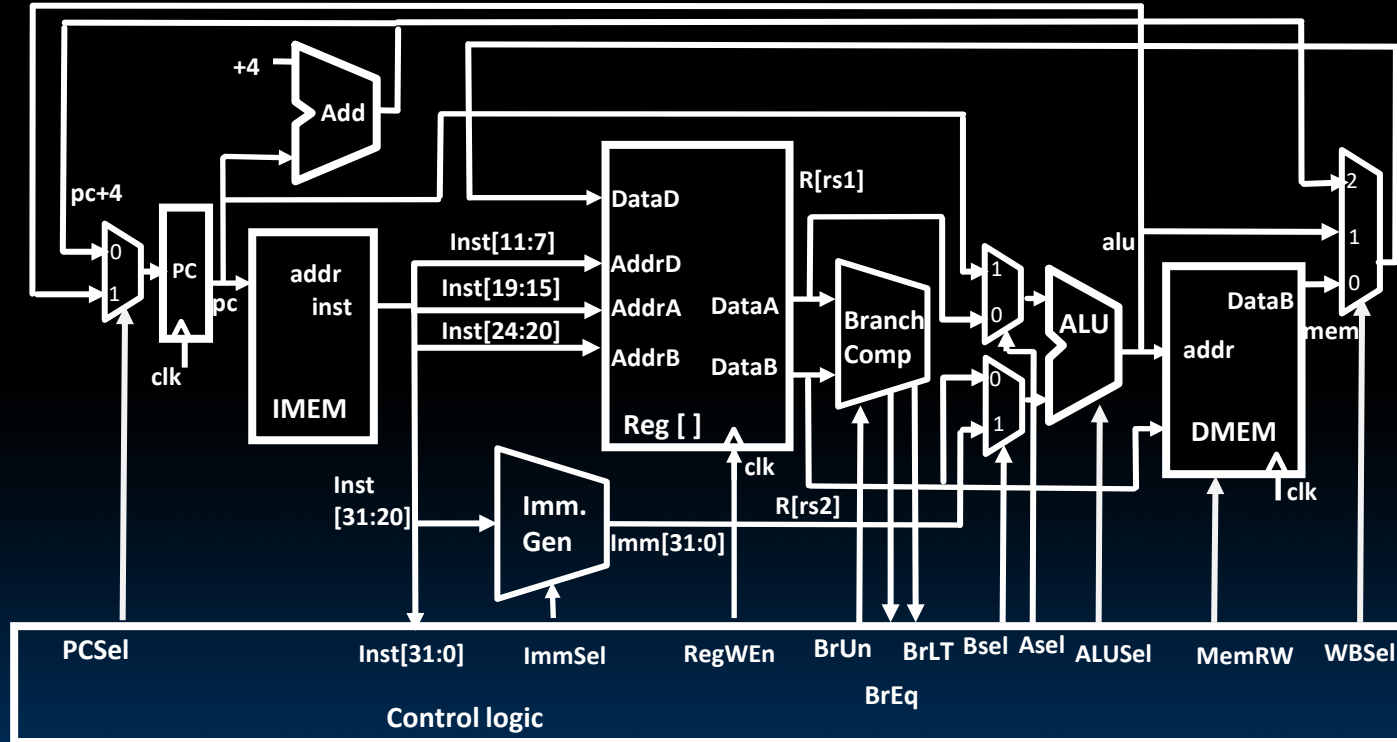


# Lighting Up AUIPC



**“And In  
Conclusion...”**

# Complete RV32I Datapath!





# Complete RV32I ISA!

## Open RISC-V Reference Card

| Base Integer Instructions: RV32I |                         |     |                   |                      |                    |                  |                  |                 |
|----------------------------------|-------------------------|-----|-------------------|----------------------|--------------------|------------------|------------------|-----------------|
| Category                         | Name                    | Fmt | RV32I Base        | Category             | Name               | Fmt              | RV32I Base       |                 |
| Shifts                           | Shift Left Logical      | R   | SLL rd,rs1,rs2    | Loads                | Load Byte          | I                | LB rd,rs1,imm    |                 |
|                                  | Shift Left Log. Imm.    | I   | SLLI rd,rs1,shamt |                      | Load Halfword      | I                | LH rd,rs1,imm    |                 |
|                                  | Shift Right Logical     | R   | SRL rd,rs1,rs2    |                      | Load Byte Unsigned | I                | LBU rd,rs1,imm   |                 |
|                                  | Shift Right Log. Imm.   | I   | SRLI rd,rs1,shamt |                      | Load Half Unsigned | I                | LHU rd,rs1,imm   |                 |
|                                  | Shift Right Arithmetic  | R   | SRA rd,rs1,rs2    |                      | Load Word          | I                | LW rd,rs1,imm    |                 |
|                                  | Shift Right Arith. Imm. | I   | SRAI rd,rs1,shamt | Stores               | Store Byte         | S                | SB rs1,rs2,imm   |                 |
| Arithmetic                       | ADD                     | R   | ADD rd,rs1,rs2    |                      | Store Halfword     | S                | SH rs1,rs2,imm   |                 |
|                                  | ADD Immediate           | I   | ADDI rd,rs1,imm   |                      | Store Word         | S                | SW rs1,rs2,imm   |                 |
|                                  | SUBtract                | R   | SUB rd,rs1,rs2    |                      | Branches           | Branch =         | B                | BEQ rs1,rs2,imm |
|                                  | Load Upper Imm          | U   | LUI rd,imm        |                      |                    | Branch ≠         | B                | BNE rs1,rs2,imm |
|                                  | Add Upper Imm to PC     | U   | AUIPC rd,imm      | Branch <             |                    | B                | BLT rs1,rs2,imm  |                 |
| Logical                          | XOR                     | R   | XOR rd,rs1,rs2    | Branch ≥             |                    | B                | BGE rs1,rs2,imm  |                 |
|                                  | XOR Immediate           | I   | XORI rd,rs1,imm   | Branch < Unsigned    |                    | B                | BLTU rs1,rs2,imm |                 |
|                                  | OR                      | R   | OR rd,rs1,rs2     | Branch ≥ Unsigned    | B                  | BGEU rs1,rs2,imm |                  |                 |
|                                  | OR Immediate            | I   | ORI rd,rs1,imm    | Jump & Link J&L      | J                  | JAL rd,imm       |                  |                 |
|                                  | AND                     | R   | AND rd,rs1,rs2    | Jump & Link Register | I                  | JALR rd,rs1,imm  |                  |                 |
|                                  | AND Immediate           | I   | ANDI rd,rs1,imm   |                      |                    |                  |                  |                 |
| Compare                          | Set <                   | R   | SLT rd,rs1,rs2    | Synch                | Synch thread       | I                | FENCE            |                 |
|                                  | Set < Immediate         | I   | SLTI rd,rs1,imm   |                      |                    |                  |                  |                 |
|                                  | Set < Unsigned          | R   | SLTU rd,rs1,rs2   |                      |                    |                  |                  |                 |
|                                  | Set < Imm Unsigned      | I   | SLTIU rd,rs1,imm  |                      |                    |                  |                  |                 |
|                                  |                         |     |                   |                      |                    |                  |                  |                 |

Environment

CALL

BREAK

ECALL

EBREAK

Not in 61C



- We have designed a complete datapath
  - Capable of executing all RISC-V instructions in one cycle each
  - Not all units (hardware) used by all instructions
- 5 Phases of execution
  - IF, ID, EX, MEM, WB
  - Not all instructions are active in all phases
- Controller specifies how to execute instructions
  - We still need to design it