



# Lecture 3-2: “li” Pseudo Instruction

## **CS10014 Computer Organization**

Department of Computer Science

Tsung Tai Yeh

Thursday: 1:20 pm– 3:10 pm

Classroom: EC-022



# Acknowledgements and Disclaimer

- Slides were developed in the reference with
  - CS 61C at UC Berkeley
    - <https://inst.eecs.berkeley.edu/~cs61c/sp23/>
  - CS 252 at UC Berkeley
    - <https://people.eecs.berkeley.edu/~culler/courses/cs252-s05/>
  - CSCE 513 at University of South Carolina
    - <https://passlab.github.io/CSCE513/>



# Outline

- U-Format
  - “li” pseudo instruction



# “li” pseudo instruction

- **li rd, Immediate**

- Load 32-bit values or address in the destination register
- How to translate “li t0 0x12345678” to instructions?
- One instruction is impossible since no 32-bit object can encode all  $2^{32}$  possible immediates AND all 32 possible destination registers

- **Solution: lui instruction**

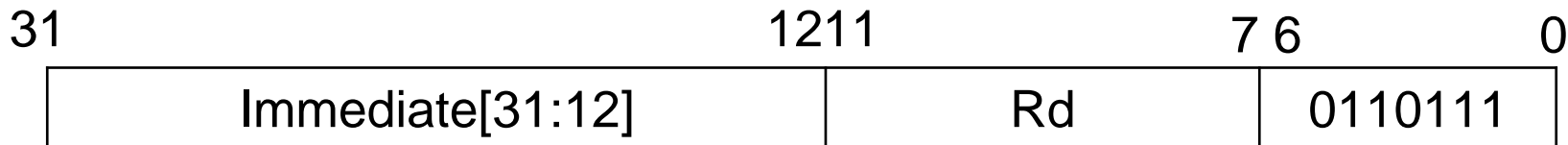
- In the above example, we can do
  - lui t0 0x12345
  - addi t0 t0 0x678



# “li” pseudo instruction

- **lui rd, immediate**

- Load Upper Immediate, U-type
- Writes the signed-extended 20-bit immediate, left-shifted by 12 bits, to x[rd], zeroing the lower 12 bits
- $x[rd] = \text{sext}(\text{immediate}[31:12]) \ll 12$

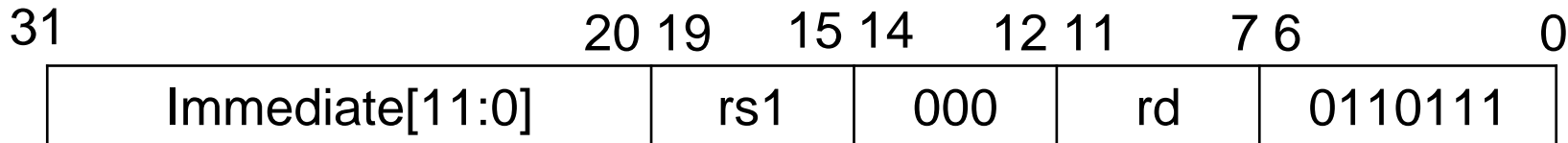




# “li” pseudo instruction

- **addi rd, rs1 immediate**

- Add immediate, I-type
- Adds the signed-extended immediate to register x[rs1] and write the result to x[rd]
- $x[rd] = x[rs1] + sext(\text{immediate})$





# “li” pseudo instruction

- **li rd, immediate**
  - **If the range of the immediate is 0 ~ 4096 (12-bit)**
    - li rd, immediate => addi rd, x0, immediate
  - **else**
    - lui rd (immediate << 12)
    - addi rd, rd, (immediate & 0xFFF)

0



# “li” pseudo instruction

- In case
  - lui a0, immediate0 << 12
  - addi a0, a0, (immediate1 & 0xFFF)
- **When 11<sup>th</sup> immediate1 = 0**
  - lui a0, immediate0 << 12
  - addi a0, a0, (immediate1 & 0xFFF)
- **When 11<sup>th</sup> immediate1 = 1**
  - lui a0, ((immediate0 + 1) << 12)
  - addi a0, a0, (immediate1 & 0xFFF); then a0 – 2<sup>12</sup>





# “li” pseudo instruction

- In case: `li a0, 0xABCEDFFF`
  - `lui a0, 0xABCDE`
  - `addi a0, a0, 0xFFF`
- Problem: `0xFFF` isn't 4095; it's -1 (2' complement)
  - `0xFFF = 1111 1111 1111` (11<sup>th</sup> immediate = 1)
    - `lui a0, (immediate0+1) << 12`
    - `add a0, a0, (immediate1 & 0xFFF); then a0 - 2^12`
  - `lui t0 0xABCDE #t0 stores 0xABCDF000`
  - `addi t0 t0 0xFFF #t0 = (0xABCDF000 & 0xFFF)-2^12`  
`#t0 = 0xABCDEF000`