# ECE3700JFA23 RC1

TA: Xu Weiqing

## About 370 & RC

1. Review Slides
2. HW before
3. Q&A
4. Lecture-RC-HW

## Assembly Programming

```
gcc -o hello hello.c
```

1. high level language (HLL) (hello.c) ( → preprocessor (cpp) → hello.i)
2. complier (ccl)
3. **Assembly language (hello.s) ← 370 focus**
4. assembler (as)
5. Machine language (hello.o) (→ linker (ld) → hello)

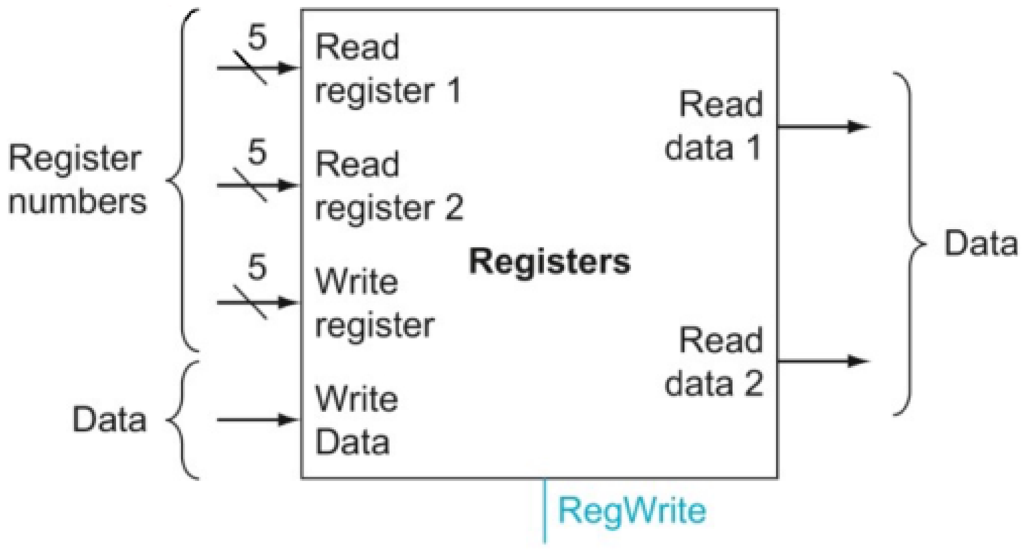
## Instruction Set Architecture (ISA)

1. a collection of instructions that a computer understands
2. RISC-V 32bits https://jicanvas.com/files/166996
3. data bus

## Design Principle

1. simplicity favors regularity P8
2. smaller is faster P11
3. make the common case P27

## Register Operands



1. RV32: 32×32bits register file    RV64: ________
2. operands

- **x0 : the constant value 0    move & clear**
- **x1 (ra) : return address**
- **x2 (sp) : stack pointer**
- x3 (gp) : global pointer
- x4 (tp) : thread pointer
- x5 – x7 , x28 – x31 : temporaries
- x8 : frame pointer
- x9 , x18 – x27 : saved registers

- x10 – x11 : function arguments/results
- x12 – x17 : function arguments

3. basic usage: **add**

   a: x5 b: x6 c: x7

   ```
   a = b + c;
   ```

## Memory Operands

1. load: mem → reg;    store: reg→mem;
2. load-perform-store
3. **byte addressable**  bit/word address?

| | 0xffff_0000 | 0xffff_0001 | 0xffff_0002 | 0xffff_0003 |
|---|---|---|---|---|
| 0xffff_0000 | | | | |

4. Big & **Little Endian samllest-least**

   0x1020A0B0

| | 0xffff_0000 | 0xffff_0001 | 0xffff_0002 | 0xffff_0003 |
|---|---|---|---|---|
| 0xffff_0000 | B0 | A0 | 20 | 10 |

5. Integer Array

   Address of Array = Base Address + Offset = Base Address + (index × 4)

   &A[n] = &A[0] + 4n

6. basic usage: **lw P23**

## Immediate Operands

1. immediate: constant data
2. **signed**
3. basic usage: **addi, slli…**

```
addi x22, x22, -1
```

Exercise: Assume i, j are in x5 , x6 respectively, and base address of array A and B are in x7 , x28 respectively.

```
A[i] = B[j] + 5;
```

```

```

## Logical Operations

1. shift: **sll, slli, srl, srli, sra, srai**

   Fill vacated bits with 0/signed bits.

   ```
   sll x5 x6 x7
   slli x5 x5 3
   srai x5 x6 3
   ```

2. **and, or, xor, andi, ori, xori**

   how to write NOT?

## Conditional Operations

1. beq, bne, blt, bge

   ```
   beq rs1 rs2 L1  # if (rs1 == rs2) goto L1
   bne rs1 rs2 L1  # if (rs1 != rs2) goto L1
   blt rs1 rs2 L1  # if (rs1 < rs2) goto L1
   bge rs1 rs2 L1  # if (rs1 >= rs2) goto L1
   ```

    how to write unconditional branch?

2. signed?

## Load upper immediate

**lui** rd, constant

load+**clear**

## Load / Store

1. lw lh lb
2. lbu lhu

## Advanced questions

1. RV64
2. 64 regs
3. design principle
4. signed offset
5. sltiu

### References

1. ECE3700JFA23 Slides T2
2. Computer Systems: A Programmer's Perspective, Third edition