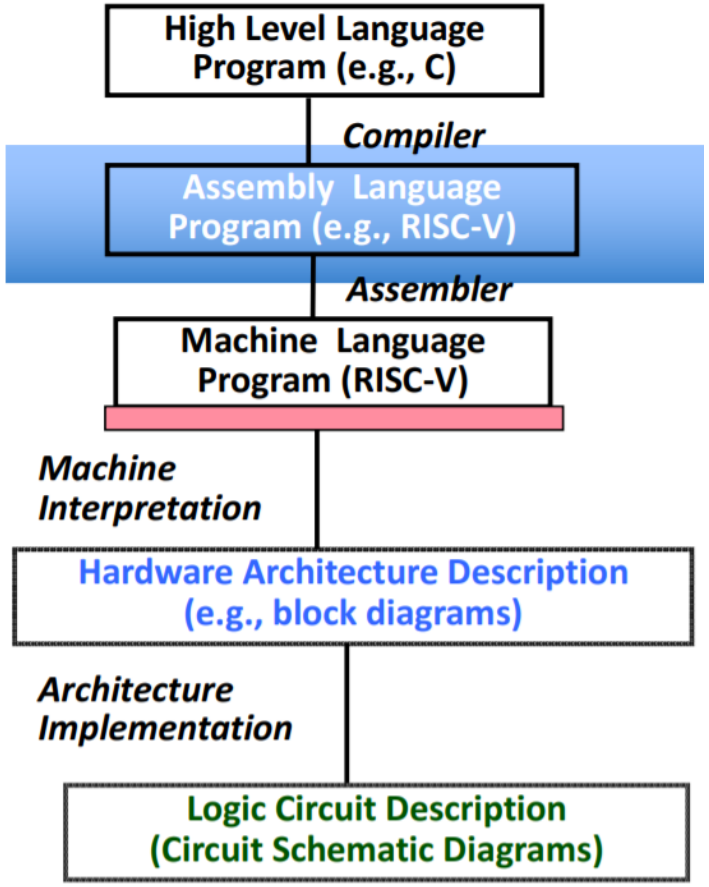
Discussion: RISC-V Intro

SONGZE@SHANGHAITECH.EDU.CN

Assembly language: RISC-V



Five

- RISC-V: the 5th generation of RISC ISA

ISA: Instruction Set Architecture

RISC: Reduced Instruction Set Computing, e.g. MIPS, RISC-V
CISC: Complex Instruction Set Computing, e.g. x86

Instructions

- Instruction is an atomic operation that a computer can do
- Most of them follow the format: A B, C, D
  - But there are also exceptions - will be covered later
- Instructions may deal with: registers; memory; immediate numbers.
- For a list of instructions, please refer to Green card.

Registers

- RISC-V has 32 **available** registers.
  - Extensions to RISC-V may enlarge that.
- Each register has 32-bit length .
  - Our textbook uses 64-bit registers along with 64-bit memory address.
- Each register has a number(e.g. x10) and a name(e.g. a1).
  - Details of names will be covered later.
  - You should use a register's name rather than its number when programming.
- Registers have no type.

Registers

- Important: registers are not variables and should **NOT** be used as variables!
  - Registers are very limited in amount
  - Recall the Great Idea #3: Principle of Locality/ Memory Hierarchy

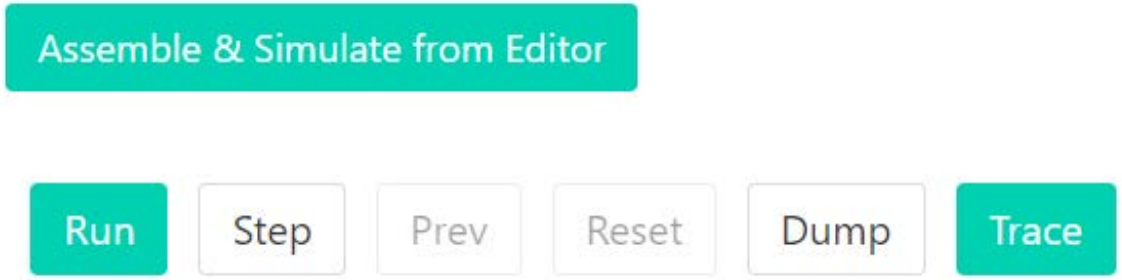
- Store data in memory
- Load data from memory before use and save data back to memory after use

Venus: A RISC-V simulator

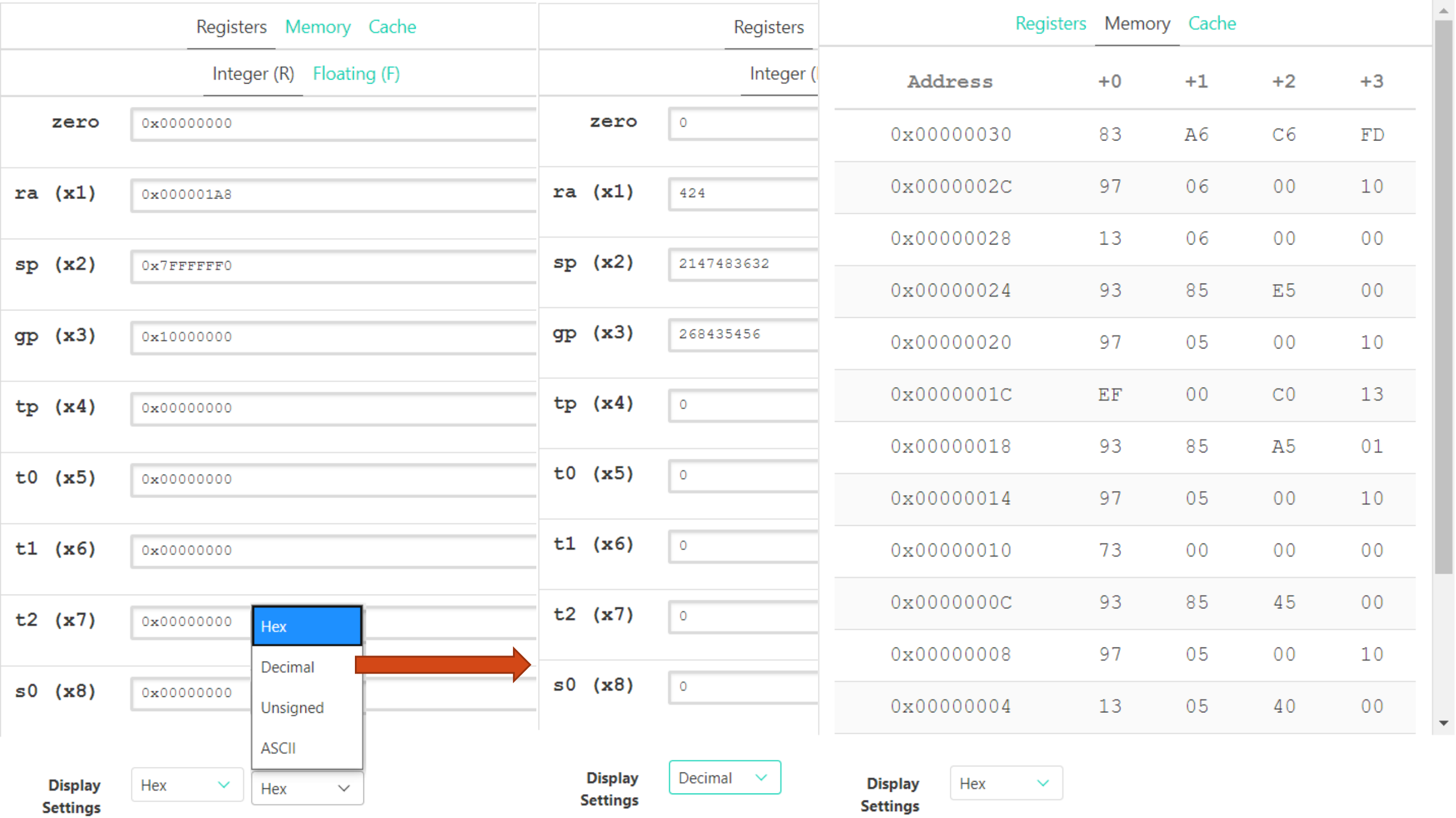
- Webpage: http://autolab.sist.shanghaitech.edu.cn/venus/

- With its GUI and tools, you can debug your assembly program very easily.
  - Supports single step, previous step, breakpoints.
  - You can see memory and register status real time.

How to use Venus



PC	Machine Code	Basic Code		Original Code
0x0	0x0040006F	jal x0 4		j receive_values_end # print the testing array
0x4	0x00400513	addi x10 x0 4	Breakpoint	li a0, 4 # 4 = print_string ecall.
0x8	0x10000597	auipc x11 65536	Current line	la a1, inputstring


Registers
Memory
Cache
Integer (R)
Floating (F)
zero 0x00000000
ra (x1) 0x000001A8
sp (x2) 0x7FFFFFF0
gp (x3) 0x10000000
tp (x4) 0x00000000
t0 (x5) 0x00000000
t1 (x6) 0x00000000
t2 (x7) 0x00000000
s0 (x8) 0x00000000
Hex
Decimal
Unsigned
ASCII
Display Settings
Hex
Hex
Registers
Integer (
zero 0
ra (x1) 424
sp (x2) 2147483632
gp (x3) 268435456
tp (x4) 0
t0 (x5) 0
t1 (x6) 0
t2 (x7) 0
s0 (x8) 0
Display Settings
Decimal
Registers
Memory
Cache
Address +0 +1 +2 +3
0x00000030 83 A6 C6 FD
0x0000002C 97 06 00 10
0x00000028 13 06 00 00
0x00000024 93 85 E5 00
0x00000020 97 05 00 10
0x0000001C EF 00 C0 13
0x00000018 93 85 A5 01
0x00000014 97 05 00 10
0x00000010 73 00 00 00
0x0000000C 93 85 45 00
0x00000008 97 05 00 10
0x00000004 13 05 40 00
Display Settings
Hex

Assembler directives

Directive	Effect
`.data`	Store subsequent items in the static segment at the next available address.
`.text`	Store subsequent instructions in the text segment at the next available address.
`.byte`	Store listed values as 8-bit bytes.
`.asciiz`	Store subsequent string in the data segment and add null-terminator.
`.word`	Store listed values as unaligned 32-bit words.

Access memory in RISC-V

- Use load/store instructions to access memory.
  - Details will be covered in later lectures & discussions.
  - Convention: **lw** t0, sth ==========>

```
.data
    sth: .word 0x1234ABCD
```

t0 (x5) 0x1234ABCD

- Little Endian

Address	+0	+1	+2	+3
0x10000000	CD	AB	34	12

- RISC-V do not require mandatory memory alignment
  - But you'd better follow this convention

```
1 .data
2     a: .word 0x1234ABCD
3     b: .byte 0xFF
4     c: .word 0xABCD1234
```

0x10000008	AB	00	00	00
0x10000004	FF	34	12	CD
0x10000000	CD	AB	34	12

Suggestions on programming in RISC-V

- Write some pseudo code / C code first, then translate into assembly.
- Write comments for every line - or you will forgot what you have done very easily!
  - Comment can be the C code / pseudo code that you refer to.
- Use register names rather than numbers! (will be covered more later)