Week 6

CS110P.07

Suting Chen

Part 1

- Basic ideas of RISC-V
- RISC-V instructions

Suting Chen

Basic ideas of RISC-V

程序是状态机（State machine）

- 有限状态机的必要组件有： State, Transition, Entrance, Input

Suting Chen

Basic ideas of RISC-V

程序是状态机

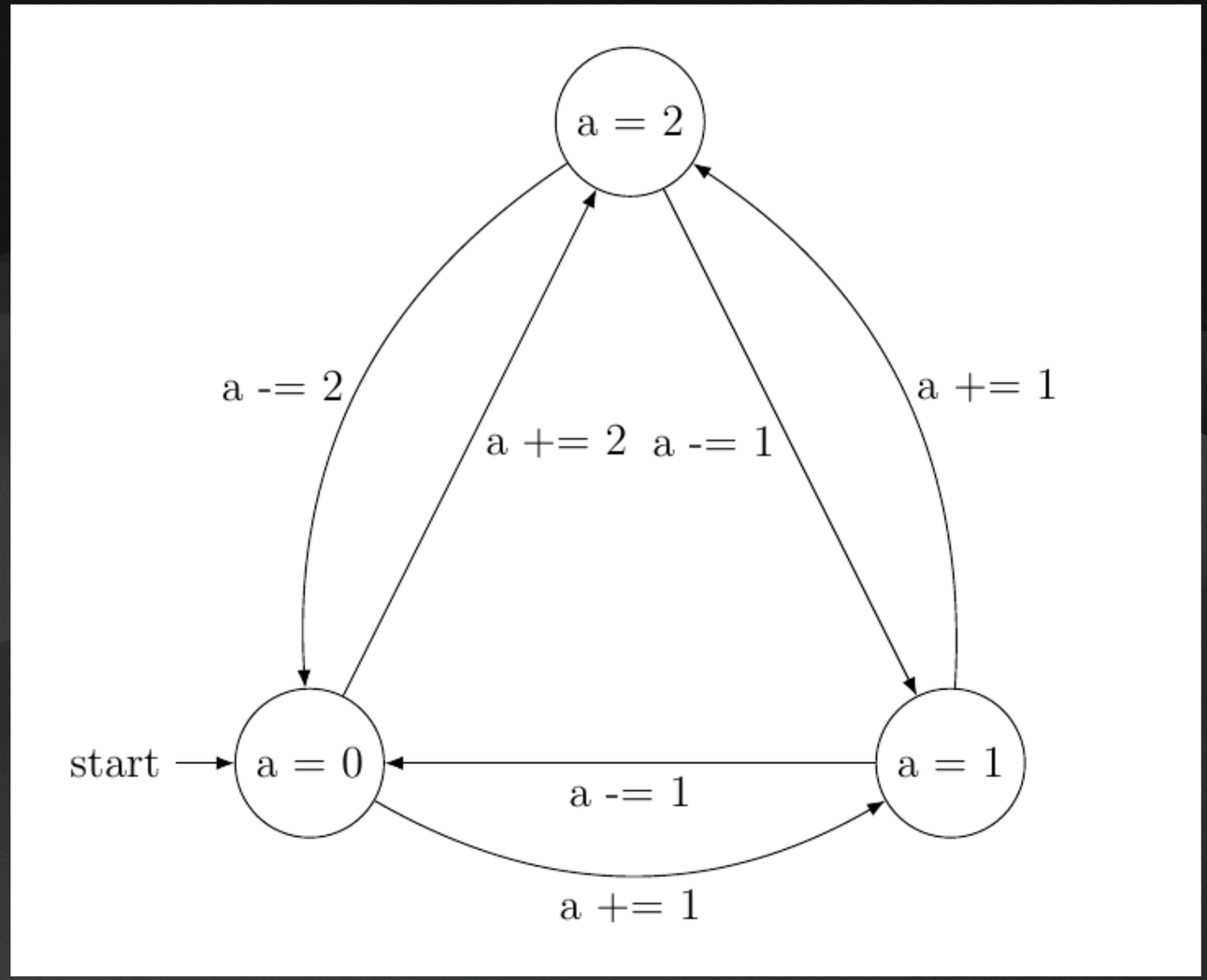
- 32个32位寄存器 (Register)
- PC (Program counter)
- 内存 (Memory)

Suting Chen

Basic ideas of RISC-V

程序是状态机

- 32个32位寄存器 (Register)
- PC (Program counter)
- 内存 (Memory)



Suting Chen

Basic ideas of RISC-V

程序是状态机

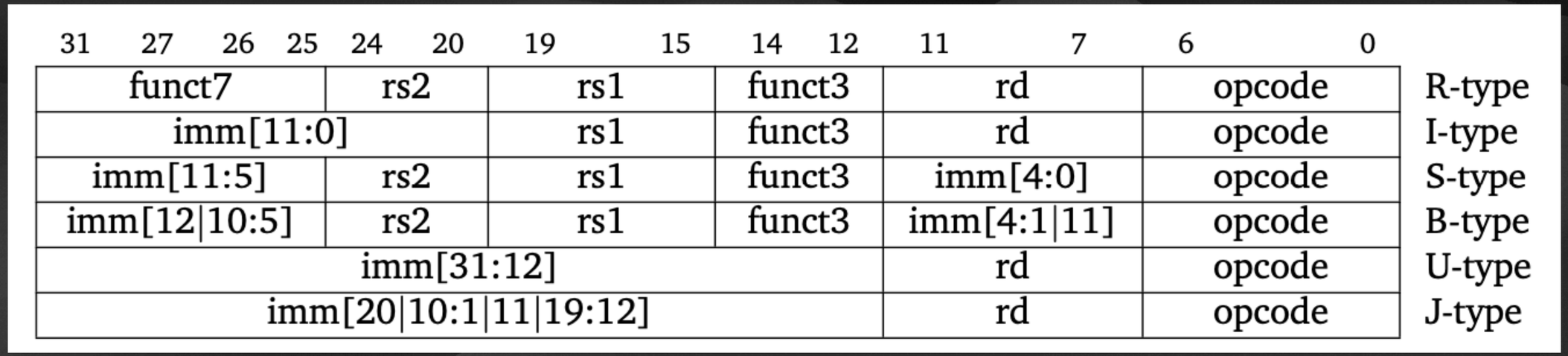
- 寄存器、内存中的数据等 — State
- 每一条汇编指令 — Transition
- 程序入口 — Entrance

- 内存中的“代码” — Input

Suting Chen

RISC-V instructions

指令根据功能分为不同类型



31 27	26 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

Suting Chen

RISC-V instructions

刚才的那些都是什么

- func7, func3, opcode — 帮助CPU识别是哪条指令
- rd — 这条指令将要“写入”的寄存器
- rs1, rs2 — 这条指令将要“读取”的寄存器
- imm — “立即”数（但不一定长得像数）

Suting Chen

RISC-V instructions

R-type

- add, sub, xor, or, and — 加减、简单位运算
- sll, srl, sra — 移位
- slt, sltu — bool a = …

Suting Chen

RISC-V instructions

R-type — add, sub, xor, or, and

```
add rd rs1 rs2
sub rd rs1 rs2
xor rd rs1 rs2
or  rd rs1 rs2
and rd rs1 rs2
```

Suting Chen

RISC-V instructions

R-type — add, sub, xor, or, and

```
t3 = t1 + t2

a1 = a6 - a0

a7 += a4
```

```
add t3 t1 t2

sub a1 a6 a0

add a7 a7 a4
```

Suting Chen

RISC-V instructions

R-type — sll, srl, sra

- shift left logical
- shift right logical
- shift right arithmetical

Suting Chen

RISC-V instructions

R-type — What are “Logical” and “arithmetical”

- 假设我们有一个四位寄存器: 0b1001
- 左移: 0b0010
- 右移: 0b0100
- 还有其他可能的右移吗

Suting Chen

RISC-V instructions

Most Significant Bit (We apply little endian here)

- 0b11001010 — 1
- 0b01100011 — 0

- 这一位经常有特殊含义（正负性等）
- 与之相对的 Least Significant Bit

Suting Chen

RISC-V instructions

R-type — What are “Logical” and “arithmetical”

- 假设我们有一个四位寄存器: 0b1001
- 左移: 0b0010
- 右移: 0b0100
- “Arithmetical”: 0b1100

Suting Chen

RISC-V instructions

R-type — sll, srl, sra

```
# t1 = 0xFFFFFFFF
# t2 = 4

sll t3 t1 t2
srl t4 t1 t2
sra t5 t1 t2
```

Suting Chen

RISC-V instructions

R-type — slt, sltu

- Set less than

- Set less than (unsigned)

Suting Chen

RISC-V instructions

R-type — Set less than

```
int32_t a = 0x1234;
int32_t b = 0x0040;

c = (a < b) ? 1 : 0;
```

```
# t1 = 0x1234
# t2 = 0x0040

slt t3 t1 t2
```

Suting Chen

RISC-V instructions

R-type — Set less than

```
int32_t a = 0xFFFFFFFF;
int32_t b = 0x02b66240;

c = (a < b) ? 1 : 0;
```

```
# t1 = 0xFFFFFFFF
# t2 = 0x02b66240

slt t3 t1 t2
```

Suting Chen

RISC-V instructions

R-type — Set less than (unsigned)

```
unsigned a = 0x1234;
unsigned b = 0x0040;

c = (a < b) ? 1 : 0;
```

```
# t1 = 0x1234
# t2 = 0x0040

sltu t3 t1 t2
```

Suting Chen

RISC-V instructions

R-type — Set less than (unsigned)

```
unsigned a = 0xFFFFFFFF;
unsigned b = 0x02b66240;

c = (a < b) ? 1 : 0;
```

```
# t1 = 0xFFFFFFFF
# t2 = 0x02b66240

sltu t3 t1 t2
```

Suting Chen

RISC-V instructions

R-type — Conclusion

add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2	
sub	SUB	R	0110011	0x0	0x20	rd = rs1 - rs2	
xor	XOR	R	0110011	0x4	0x00	rd = rs1 ^ rs2	
or	OR	R	0110011	0x6	0x00	rd = rs1 | rs2	
and	AND	R	0110011	0x7	0x00	rd = rs1 & rs2	
sll	Shift Left Logical	R	0110011	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	R	0110011	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith*	R	0110011	0x5	0x20	rd = rs1 >> rs2	msb-extends
slt	Set Less Than	R	0110011	0x2	0x00	rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	R	0110011	0x3	0x00	rd = (rs1 < rs2)?1:0	zero-extends

Suting Chen

RISC-V instructions

I-type

- addi, xori, ori, andi, slli, srli, srai — 计算
- slti, sltiu — Set less than
- lb, lw, lu, lbu, lhu — 从内存加载数据

- jalr, ecall, ebreak

Suting Chen

RISC-V instructions

I-type — addi, xori, ori, andi, slli, srli, srai

```
addi rd rs1 imm
xori rd rs1 imm
ori  rd rs1 imm
andi rd rs1 imm
slli rd rs1 imm
srli rd rs1 imm
srai rd rs1 imm
```

```
rd = rs1 addi imm
rd = rs1 xori imm
rd = rs1 ori  imm
rd = rs1 andi imm
rd = rs1 slli imm
rd = rs1 srli imm
rd = rs1 srai imm
```

Suting Chen

Exercise

I-type — addi, xori, ori, andi, slli, srli, srai

Suting Chen

RISC-V instructions

I-type — slti, sltiu

- Set less than imm

- Set less than imm (unsigned)

Suting Chen

RISC-V instructions

I-type — slti, sltiu

```
int32_t a = 0x1234;

c = (a < 0x0040) ? 1 : 0;
```

```
# t1 = 0x1234

slti t3 t1 0x0040
```

Suting Chen

RISC-V instructions

I-type — slti, sltiu

```
int32_t a = 0xFFFFFFFF;


c = (a < 0x6240) ? 1 : 0;
```

```
# t1 = 0xFFFFFFFF


slti t3 t1 0x6240
```

Suting Chen

RISC-V instructions

I-type — Set less than imm (unsigned)

```
unsigned a = 0x1234;

c = (a < 0x0040) ? 1 : 0;
```

```
# t1 = 0x1234

sltiu t3 t1 0x0040
```

Suting Chen

RISC-V instructions

I-type — Set less than imm (unsigned)

```
unsigned a = 0xFFFFFFFF;


c = (a < 0x6240) ? 1 : 0;
```

```
# t1 = 0xFFFFFFFF


sltiu t3 t1 0x6240
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
lb  rd imm(rs1)
lw  rd imm(rs1)
lu  rd imm(rs1)
lbu rd imm(rs1)
lhu rd imm(rs1)
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
load  byte                 rd imm(rs1)
load  word                 rd imm(rs1)
load            unsigned   rd imm(rs1)
load  byte      unsigned   rd imm(rs1)
load  halfword  unsigned   rd imm(rs1)
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
.data
number:
.word 0x01234567 0x89ABCDEF

.text
la t0 number
lw t1 0(t0)
lb t1 0(t0)
lb t1 3(t0)
lh t1 1(t0)
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
.text
la t0 number
lw t1 0(t0)
lb t1 0(t0)
lb t1 3(t0)
lh t1 1(t0)
```

89	AB	CD	EF
01	23	45	67

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
.text
la t0 number
lw t1 0(t0)
lb t1 0(t0)
lb t1 3(t0)
lh t1 1(t0)
```

89	AB	CD	EF
01	23	45	67

```
0x01234567
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
.text
la t0 number
lw t1 0(t0)
lb t1 0(t0)
lb t1 3(t0)
lh t1 1(t0)
```

89	AB	CD	EF
01	23	45	67

```
0x00000067
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
.text
la t0 number
lw t1 0(t0)
lb t1 0(t0)
lb t1 3(t0)
lh t1 1(t0)
```

89	AB	CD	EF
01	23	45	67

```
0x00000001
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu

```
.text
la t0 number
lw t1 0(t0)
lb t1 0(t0)
lb t1 3(t0)
lh t1 1(t0)
```

89	AB	CD	EF
01	23	45	67

```
0x00002345
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu (Some strange cases)

```
.text
la  t0 number
lw  t1 3(t0)
lhu t1 5(t0)
lh  t1 5(t0)
lbu t1 7(t0)
lb  t1 7(t0)
```

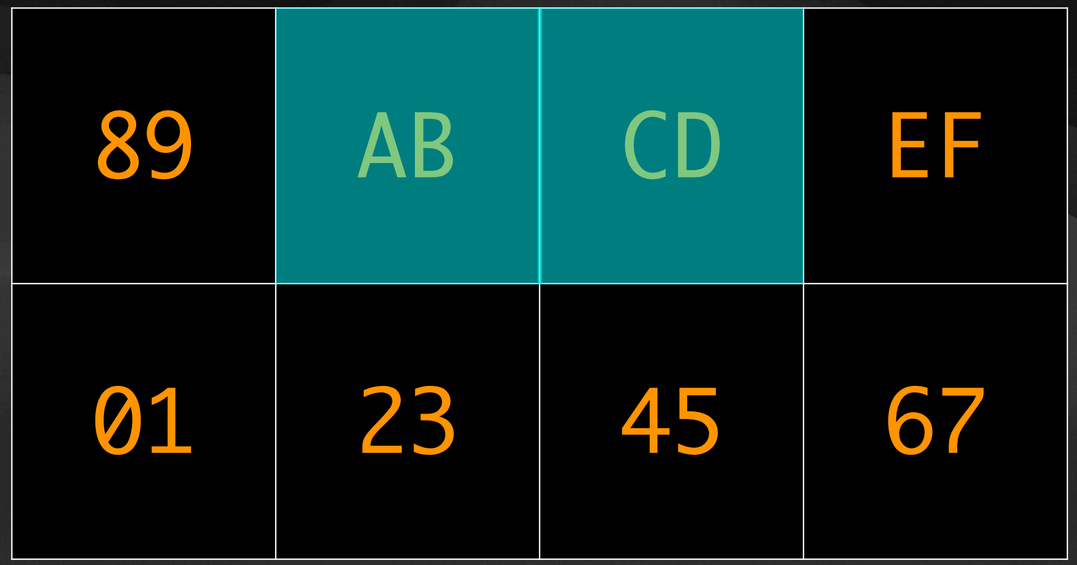

```
0xABCDEF01
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu (Some strange cases)

```
.text
la   t0 number
lw   t1 3(t0)
lhu  t1 5(t0)
lh   t1 5(t0)
lbu  t1 7(t0)
lb   t1 7(t0)
```




```
0x0000ABCD
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu (Some strange cases)

```
.text
la   t0 number
lw   t1 3(t0)
lhu  t1 5(t0)
lh   t1 5(t0)
lbu  t1 7(t0)
lb   t1 7(t0)
```

89	AB	CD	EF
01	23	45	67

```
0xFFFFABCD
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu (Some strange cases)

```
.text
la   t0 number
lw   t1 3(t0)
lhu  t1 5(t0)
lh   t1 5(t0)
lbu  t1 7(t0)
lb   t1 7(t0)
```

89	AB	CD	EF
01	23	45	67

```
0x00000089
```

Suting Chen

RISC-V instructions

I-type — lb, lw, lu, lbu, lhu (Some strange cases)

```
.text
la  t0 number
lw  t1 3(t0)
lhu t1 5(t0)
lh  t1 5(t0)
lbu t1 7(t0)
lb  t1 7(t0)
```

89	AB	CD	EF
01	23	45	67

0xFFFFFF89

Suting Chen

RISC-V instructions

I-type — Conclusion (jalr, ecall will be covered later)

addi	ADD Immediate	I	0010011	0x0		rd = rs1 + imm	
xori	XOR Immediate	I	0010011	0x4		rd = rs1 ^ imm	
ori	OR Immediate	I	0010011	0x6		rd = rs1 | imm	
andi	AND Immediate	I	0010011	0x7		rd = rs1 & imm	
slli	Shift Left Logical Imm	I	0010011	0x1	imm[5:11]=0x00	rd = rs1 << imm[0:4]	
srli	Shift Right Logical Imm	I	0010011	0x5	imm[5:11]=0x00	rd = rs1 >> imm[0:4]	
srai	Shift Right Arith Imm	I	0010011	0x5	imm[5:11]=0x20	rd = rs1 >> imm[0:4]	msb-extends
slti	Set Less Than Imm	I	0010011	0x2		rd = (rs1 < imm)?1:0	
sltiu	Set Less Than Imm (U)	I	0010011	0x3		rd = (rs1 < imm)?1:0	zero-extends
lb	Load Byte	I	0000011	0x0		rd = M[rs1+imm][0:7]	
lh	Load Half	I	0000011	0x1		rd = M[rs1+imm][0:15]	
lw	Load Word	I	0000011	0x2		rd = M[rs1+imm][0:31]	
lbu	Load Byte (U)	I	0000011	0x4		rd = M[rs1+imm][0:7]	zero-extends
lhu	Load Half (U)	I	0000011	0x5		rd = M[rs1+imm][0:15]	zero-extends

Suting Chen

RISC-V instructions

S-type

- sb, sh, sw — 向内存写入数据

Suting Chen

RISC-V instructions

S-type

```
sb rs2 imm(rs1)
sh rs2 imm(rs1)
sw rs2 imm(rs1)
```

Suting Chen

RISC-V instructions

S-type

```
Store  byte     rs2 imm(rs1)
Store  word     rs2 imm(rs1)
Store  halfword rs2 imm(rs1)
```

Suting Chen

RISC-V instructions

S-type

Suting Chen

RISC-V instructions

S-type

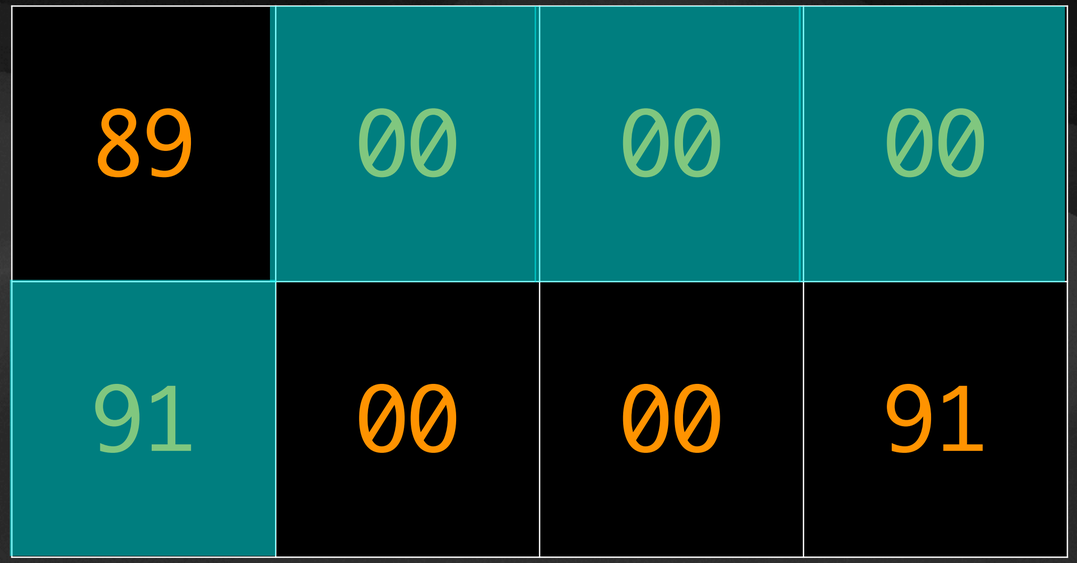
```
.text
la t0 number
li t1 0x91
sw t1 0(t0)
sw t1 3(t0)
sb t1 2(t0)
sh t1 6(t0)
```

Suting Chen

RISC-V instructions

S-type

```
.text
la t0 number
li t1 0x91
sw t1 0(t0)
sw t1 3(t0)
sb t1 2(t0)
sh t1 6(t0)
```



Suting Chen

RISC-V instructions

S-type

```
.text
la t0 number
li t1 0x91
sw t1 0(t0)
sw t1 3(t0)
sb t1 2(t0)
sh t1 6(t0)
```

89	00	00	00
91	91	00	91

Suting Chen

RISC-V instructions

S-type

```
.text
la t0 number
li t1 0x91
sw t1 0(t0)
sw t1 3(t0)
sb t1 2(t0)
sh t1 6(t0)
```

00	91	00	00
91	91	00	91

Suting Chen

RISC-V instructions

S-type

sb	Store Byte	S	0100011	0x0		M[rs1+imm][0:7] = rs2[0:7]	
sh	Store Half	S	0100011	0x1		M[rs1+imm][0:15] = rs2[0:15]	
sw	Store Word	S	0100011	0x2		M[rs1+imm][0:31] = rs2[0:31]	

Suting Chen

RISC-V instructions

简单总结一下

- RISC-V提倡内存操作向32位对齐，但不强制。
- 向内存写入时 覆盖 内存中已有的内容（初始化或未初始化）
- 从内存读取时，如果长度不及32位，根据指令选择 补0或MSB

Suting Chen

RISC-V instructions

U-type

- lui — Load Upper Immediate

- auipc — Add Upper Immediate to PC

Suting Chen

RISC-V instructions

U-type — Load Upper Immediate

```
lui t0 0x3ab85
```

```
t0 = 0x3ab85000
```

Suting Chen

RISC-V instructions

U-type — Add Upper Immediate to PC

```
auipc t0 0x3

rd = PC + imm << 12

t0 = PC + 0x3 << 12
```

Suting Chen

RISC-V instructions

U-type

lui	Load Upper Imm	U	0110111			rd = imm << 12	
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)	

Suting Chen

RISC-V instructions

B-type

- beq, bne, blt, bge, bltu, bgeu

- How label becomes immediates?

Suting Chen

RISC-V instructions

B-type — beq, bne, blt, bge, bltu, bgeu

```
beq  rs1 rs2 imm
bne  rs1 rs2 imm
blt  rs1 rs2 imm
bge  rs1 rs2 imm
bltu rs1 rs2 imm
bgeu rs1 rs2 imm
```

```
if (rs1 beq  rs2) goto label;
if (rs1 bne  rs2) goto label;
if (rs1 blt  rs2) goto label;
if (rs1 bge  rs2) goto label;
if (rs1 bltu rs2) goto label;
if (rs1 bgeu rs2) goto label;
```

Exercise

B-type — Jump or not?

Suting Chen

RISC-V instructions

B-type

beq	Branch ==	B	1100011	0x0		if(rs1 == rs2) PC += imm	
bne	Branch !=	B	1100011	0x1		if(rs1 != rs2) PC += imm	
blt	Branch <	B	1100011	0x4		if(rs1 < rs2) PC += imm	
bge	Branch ≥	B	1100011	0x5		if(rs1 >= rs2) PC += imm	
bltu	Branch < (U)	B	1100011	0x6		if(rs1 < rs2) PC += imm	zero-extends
bgeu	Branch ≥ (U)	B	1100011	0x7		if(rs1 >= rs2) PC += imm	zero-extends

Suting Chen

How label becomes immediates?

Suting Chen

RISC-V instructions

How label becomes immediates?

- Calculate the offset.

```
0 nop
1 blt x0 x0 label_1
2 nop
  label_1:
3 nop
```

```
0x0 addi x0 x0 0
0x4 blt  x0 x0 8
0x8 addi x0 x0 0
label_1:
0xC addi x0 x0 0
```

RISC-V instructions

How label becomes immediates?

- Calculate the offset.

```
0x0 addi x0 x0 0
0x4 blt  x0 x0 8
0x8 addi x0 x0 0
    label_1:
0xC addi x0 x0 0
```

```
0x0 0x00000013
0x4 0x00004463
0x8 0x00000013
0xC 0x00000013
```

RISC-V instructions

How label becomes immediates?

- Wait … What is the imm field?

```
0x0 addi x0 x0 0
0x4 blt  x0 x0 8
0x8 addi x0 x0 0
    label_1:
0xC addi x0 x0 0
```

```
0x0 0x00000013
0x4 0x00004463
0x8 0x00000013
0xC 0x00000013
```

- Extract immediate field -> 0x004

Exercise

B-type — Calculate offset

Suting Chen

RISC-V instructions

Jal, Jalr

- Jump And Link

- Jump And Link Register

Suting Chen

RISC-V instructions

Jump And Link

```
0 nop
1 jal sp label_1
2 nop
3 nop
      label_1:
4 nop
```

```
0x0  addi x0 x0 0
0x4  jal  x2 12
0x8  addi x0 x0 0
0xC  addi x0 x0 0
      label_1:
0x10 addi x0 x0 0
```

Suting Chen

RISC-V instructions

Jump And Link

- The same thing about imm field happened again

```
0x0   addi x0 x0 0
0x4   jal  x2 12
0x8   addi x0 x0 0
0xC   addi x0 x0 0
      label_1:
0x10 addi x0 x0 0
```

```
0x0   0x00000013
0x4   0x00C0016F
0x8   0x00000013
0xC   0x00000013
0x10  0x00000013
```

- Extract immediate field -> 0x00006

RISC-V instructions

Jump And Link Register

```
0 la t0 label_2
1 jalr sp t0 4
2 nop
3 nop
  label_2:
4 nop
5 nop
```

```
# load address
0x8  jalr x2 x5 4
0xC  addi x0 x0 0
0x10 addi x0 x0 0
     label_2:
0x14 addi x0 x0 0
```

Suting Chen

RISC-V instructions

Jump And Link Register

- This time we have the corresponding offset

```
# load address
0x8  jalr x2 x5 4
0xC  addi x0 x0 0
0x10 addi x0 x0 0
     label_2:
0x14 addi x0 x0 0
```

```
0x8  0x00428167
0xC  0x00000013
0x10 0x00000013
0x14 0x00000013
```

- Extract immediate field -> 0x004

Exercise

Jal, Jalr

Suting Chen

RISC-V instructions

Immediates — Conclusion

- 所有计算机算出来的都要除以二
- 所有手写的都不变

Suting Chen

RISC-V instructions

ecall (optional)

- Similar to syscall in real life.

Suting Chen

Part 2

- Heap, Stack, Static data
- Memory issue

Suting Chen

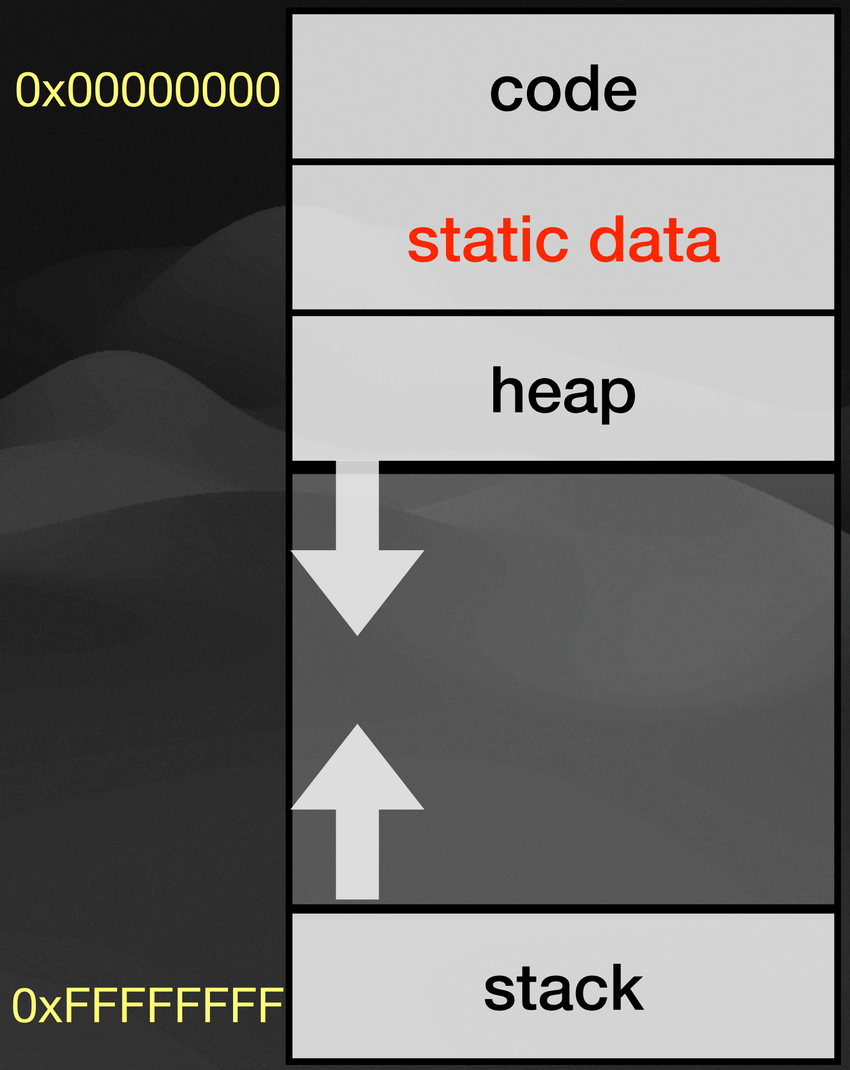
Heap, Stack, Static data

Static data

- 为什么叫static？位置在一开始就确定了，永远不会改变。
- 在什么位置？heap和code之间

```
int foo_0 = 123;

void func(void) {
    static int foo_1 = 258;
}
```

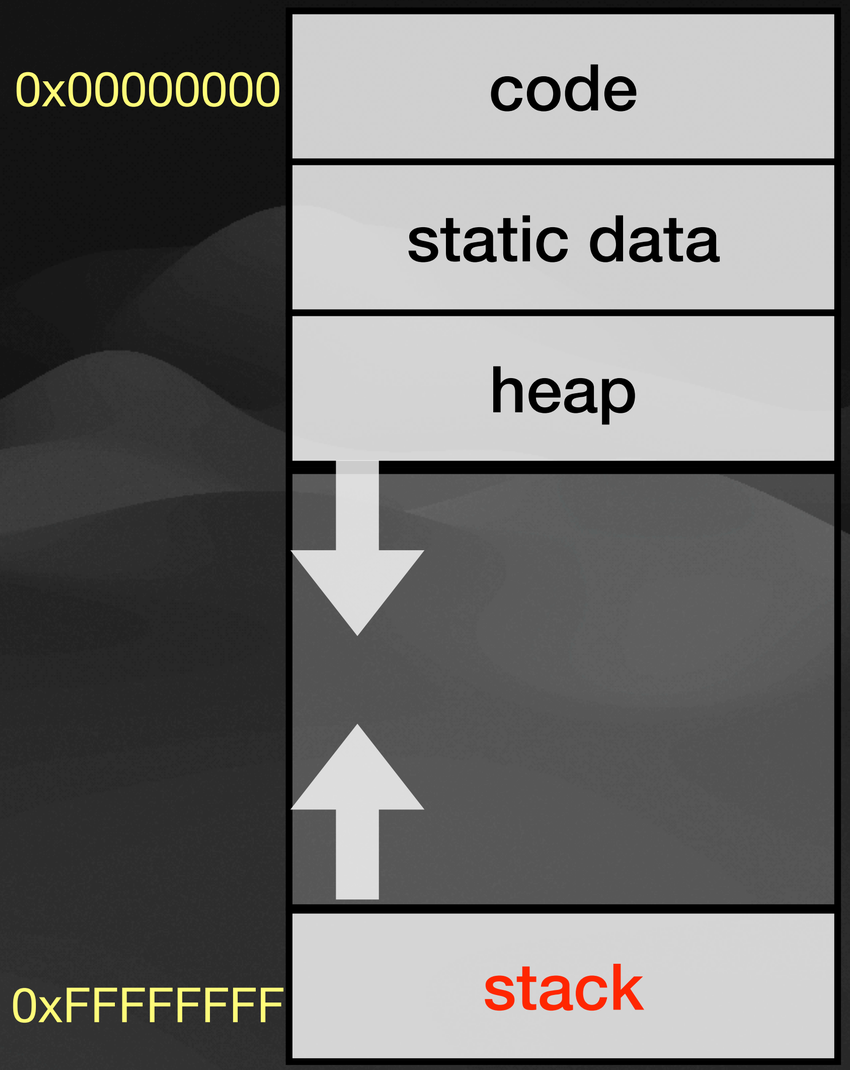


Heap, Stack, Static data

Stack

- 形象一些的记忆：函数栈
- 符合FILO (First In Last Out)

```
void func(void) {
    double foo_0 = 0.0;
    int foo_1 = 258;
}
```

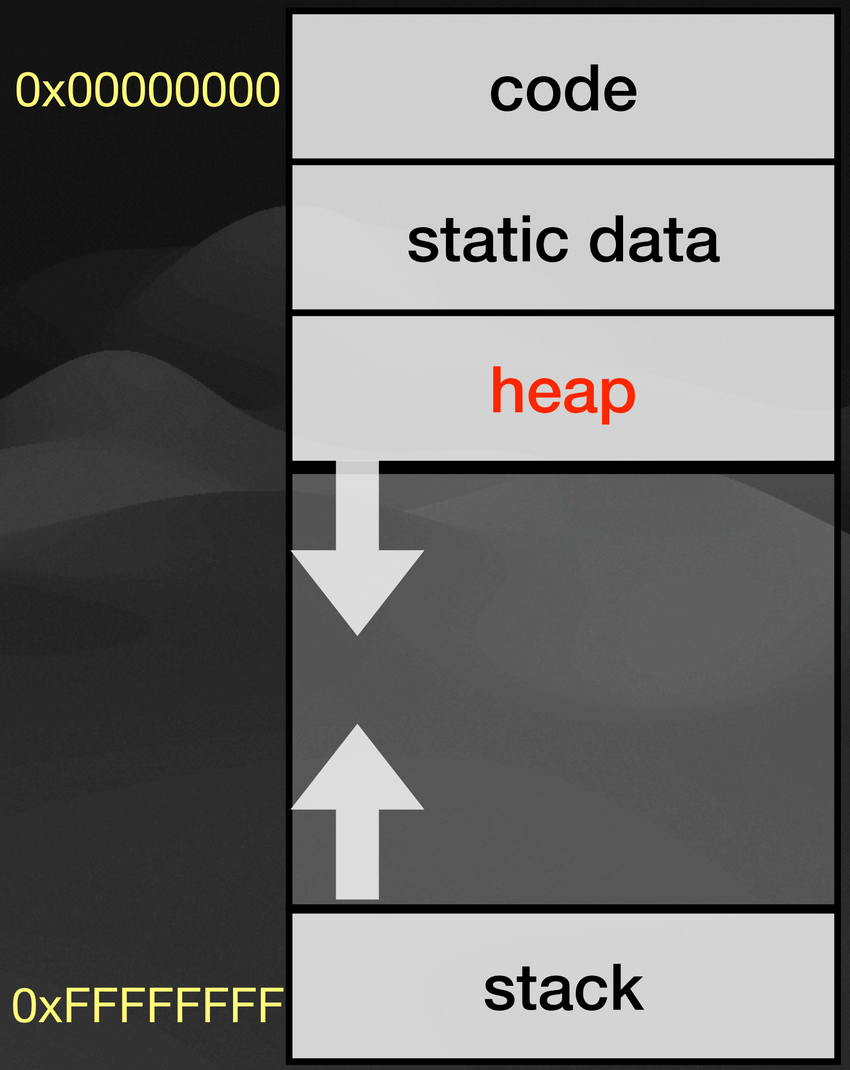


Heap, Stack, Static data

Heap

- 为什么叫heap？堆
- 动态分配，99%的内存泄露来源

```
void func(void) {
    char *string = malloc(10);
    free(string);
}
```



Exercise

Part 2

Suting Chen

Start early!

... or fail the course

Recommended readings

1. Why JALR encodes the LSB?
2. All my slides are here.

Reference

1. Pictures on RISC-V green card at [https://github.com/jameslzhu/riscv-card]