

《计算机组成原理》RISC-V 汇编练习 3 参考答案

一、选择

1、在 RISC-V 汇编语言中，op Regx, Regy, Regz 中，从左到右三个寄存器依次代表（ C ）

- A、源寄存器 1, 源寄存器 2, 目的寄存器
- B、目的寄存器, 源寄存器 2, 源寄存器 1
- C、目的寄存器, 源寄存器 1, 源寄存器 2
- D、源寄存器 2, 源寄存器 1, 目的寄存器

点评：通用指令格式，比如 R 型就是 op rd, rs1, rs2

2、对于一个给定的函数，以下哪种编程语言代码量最大？（ C ）

哪种编程语言代码量最小？（ B ）

- A、C 语言
- B、Java 语言
- C、RV 汇编语言

3、芯片中寄存器的数量随时间变化增长率（ B ）

- A、符合摩尔定律
- B、与新指令集架构有关系

4、RISC-V 中条件分支的字节地址范围是多少？（ D ）

- A、地址在 0 到 4K-1 之间
- B、地址在 0 到 8K-1 之间
- C、分支前后地址范围各大约 2K
- D、分支前后地址范围各大约 4K

点评：B 型指令含隐藏位共 13 位，前后各大约 2^{12} 字节=即前后各大约 4K 字节

精确来说： $-2^{12} \sim 2^{12}-2$ 字节(隐藏位始终是 0，不存在奇数)，-4096~4094 字节

5、RISC-V 中 jal 的字节地址范围是多少？（ D ）

- A、地址在 0 到 512K-1 之间
- B、地址在 0 到 1M-1 之间
- C、分支前后地址范围各大约 512K
- D、分支前后地址范围各大约 1M

点评：J 型指令含隐藏位共 21 位，前后各大约 2^{20} 字节=即前后各大约 1M 字节

精确来说： $-2^{20} \sim 2^{20}-2$ 字节 (隐藏位始终是 0，不存在奇数)，-1M~1M-2 字节

二、填空

1、R 型指令格式中，从左到右，依次为 funct7, rs2, rs1, funct3, rd, opcode。

(填空从这六项里选择：funct3、funct7、opcode、rs1、rs2、rd)

从左到右各个字段的长度分别为：7, 5, 5, 3, 5, 7。

点评：见黑书教材 P58 下面，funct3 和 funct7 叫功能码，本质也是一种操作码字段。

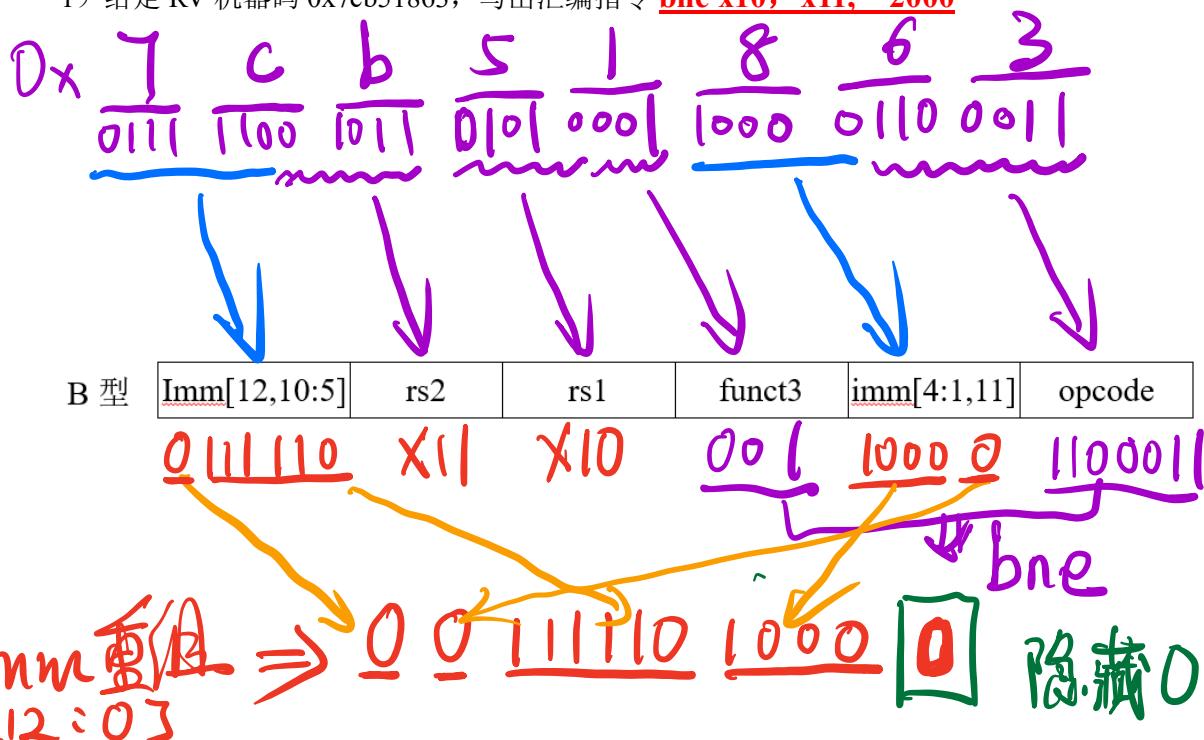
2、汇编指令解读类题型

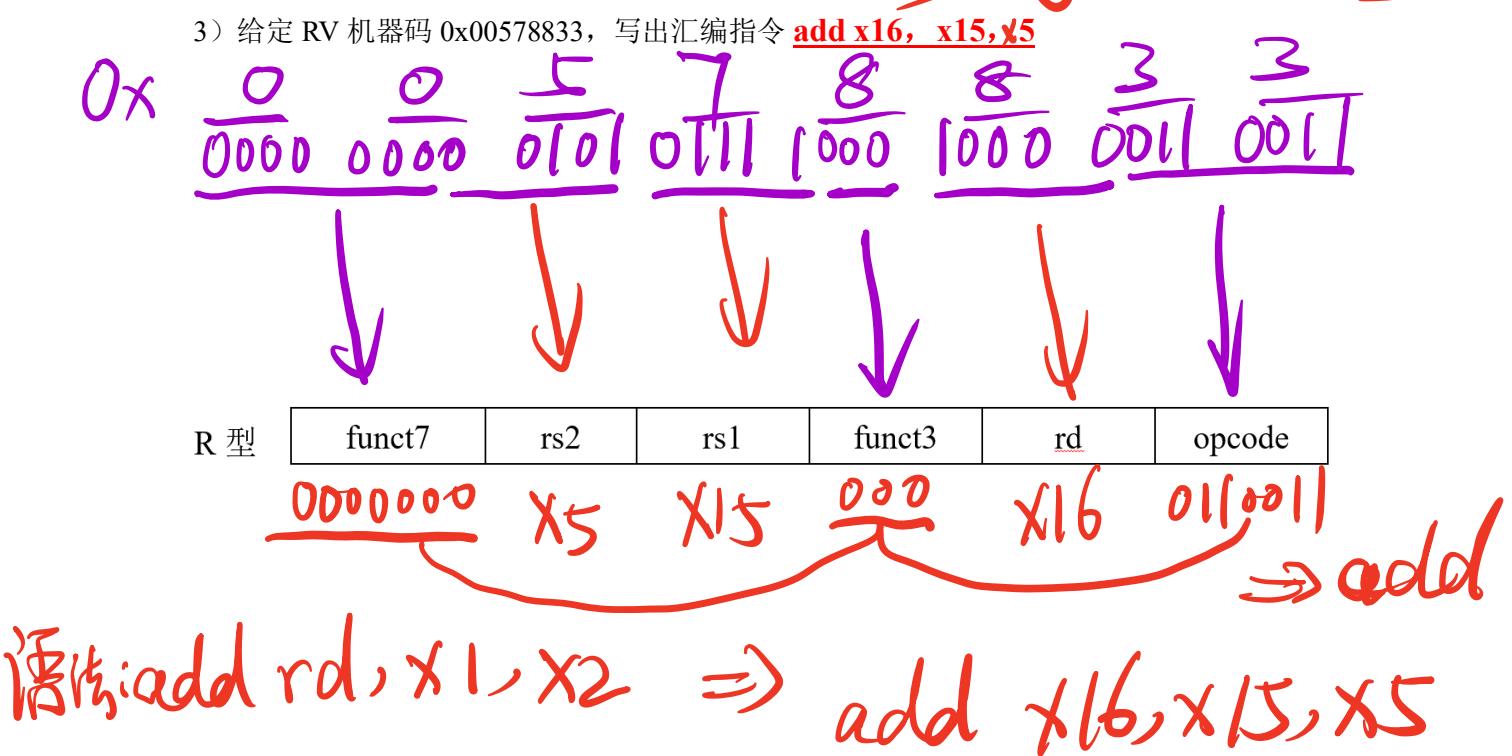
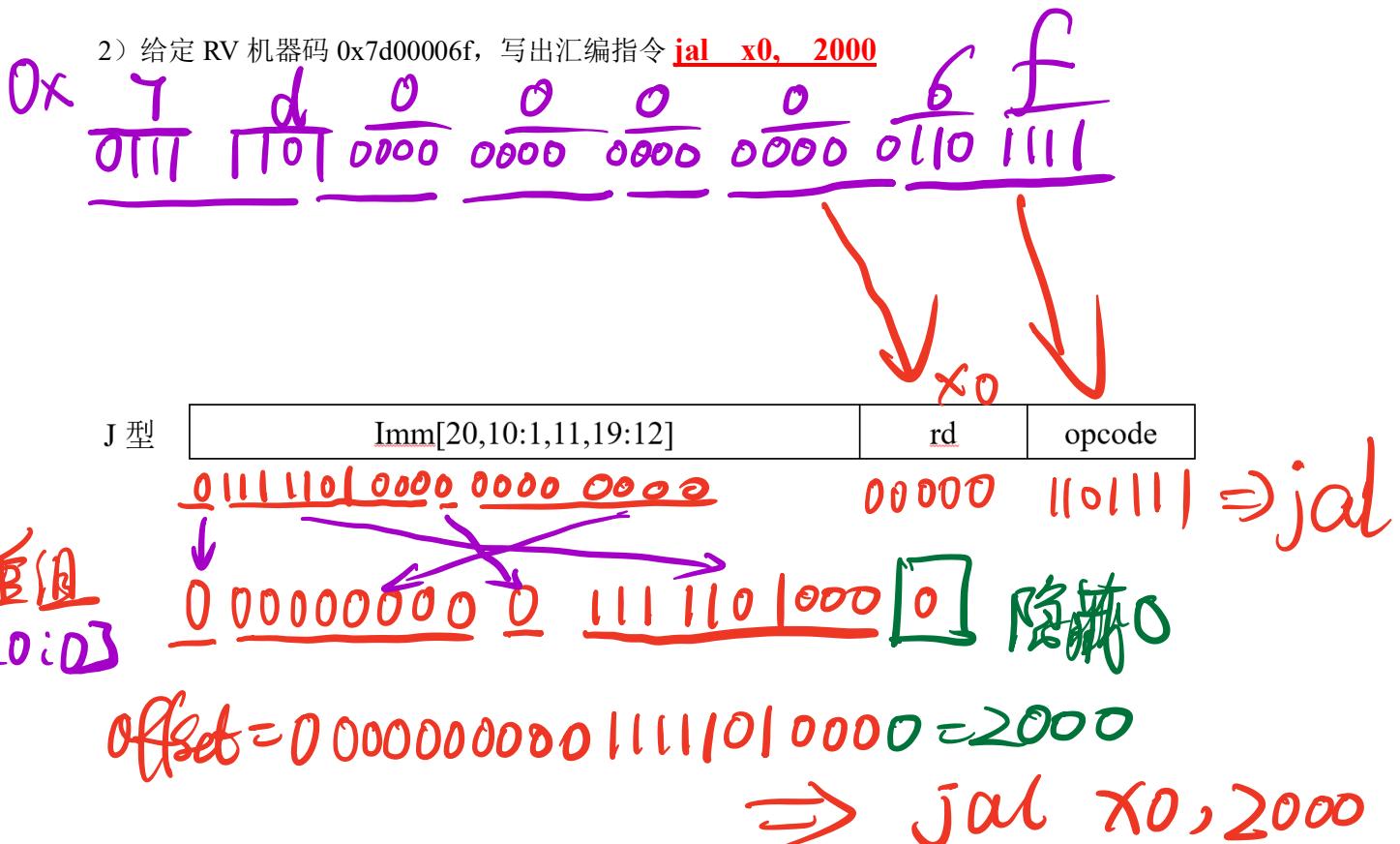
R型	funct7	rs2	rs1	funct3	<u>rd</u>	opcode
I型	imm[11:0]		rs1	funct3	<u>rd</u>	opcode
S型	Imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
B型	Imm[12,10:5]	rs2	rs1	funct3	imm[4:1,11]	opcode
J型	Imm[20,10:1,11,19:12]			<u>rd</u>	opcode	
U型	Imm[31:12]			<u>rd</u>	opcode	

指令语法

类型	原始语法	示例
R	op rd, rs1, rs2	add x5, x6, x7
I	op rd, rs1, imm (寄存器跳转) op rd, offset(rs1)	addi x5, x6, -10 jalr x1, 100(x5)
S	(load载入类) op rd, offset(rs1)	ld x5, 40(x6)
	(store存储类) op rs2, offset(rs1)	sd x5, 40(x6)
B	op rs1, rs2, offset	beq x5, x6, 100
J	op rd, offset	jal x1, 100
U	op rd, imm	lui x10, 0x87654 # x10 = 0x87654000

1) 给定 RV 机器码 0x7cb51863, 写出汇编指令 bne x10, x11, 2000





4) RV64 中, 将 x3 作为基地址偏移 100 个字节的双字长数据装载到 x6 里面, 写出汇编指令 ld x6, 100(x3), 该指令格式 rs1 对应寄存器 x3, rd 对应寄存器 x6。

5) 写出将 x5 里的一个字长数据存储到内存里, 其中内存基地址存储在 x7 里, 同时偏移量是 40, 写出汇编指令 sw x5, 40(x7)。该指令格式 rs1 对应寄存器 x7, rs2 对应寄存器 x5。

注意: load 和 store 类型指令中的 rs1 都对应内存基地址, 放在最后。

三、主观大题：

1、已知分支语句 **beq x10, x0, L1**

为了提供更大的分支距离，请给出替换代码（共3行）：

bne x10, x0, L2

jal x0, L1

L2:

点评：黑书 P82 例题，考察 jal 比 B 型指令跳转更远。

2、在 RV64 中，假设 long long int A[]首地址存在 x10 中、long long int h 存放在 x21 中，针对 C 语言代码：**A[30]=h+A[30]-1**，修改下面的汇编代码，将正确的完整代码填表：

错误代码	正确代码	十六进制机器码
lw x9, 120(x10)	ld x9, 240(x10)	0x0f053483
addi x9, x21, x9	add x9, x21, x9	0x009a84b3
subi x9, x9, 1	addi x9, x9, -1	0xffff48493
sw x10, 120(x9)	sd x9,240(x10)	0xe953823

0x0f053483	immediate	rs1	funct3	rd	opcode
	000011110000	01010	011	01001	0000011
0x009a84b3	funct7	rs2	rs1	funct3	rd
	0000000	01001	10101	000	01001
0xffff48493	immediate	rs1	funct3	rd	opcode
	111111111111	01001	000	01001	0010011
0xe953823	immediate[11:5]	rs2	rs1	funct3	immediate[4:0] opcode
	0000111	01001	01010	011	10000 0100011

点评：注意数组中的数据类型是 **int**，还是 **long long int**，决定了指令及偏移量不同。

不存在 **subi** 指令，只有 **addi** 并且 **addi** 的立即数可以用负数。

3、在 RV64 下，针对下面伪指令写出对应的两条硬件指令及十六进制机器码（填表）

li x19, 0x00000000003d0500

硬件指令	十六进制机器码
lui x19, 0x3d0	0x003d09b7
addi x19, x19, 0x500	0x50098993

① lui 是 U型指令

x19

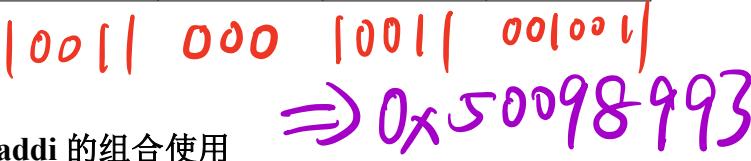
U型

Imm[31:12]	rd	opcode
0000 0000 0011 110 0000 1001 0110 111		
003d0		

查卡片

$\Rightarrow 0x003d09b7$

I型	imm[11:0]	rs1	funct3	rd	opcode
----	-----------	-----	--------	----	--------

② addi 是 I 型

 4、深入探讨 lui 与 addi 的组合使用

我们的 PPT 中都是以 RV32 进行的讨论

如何创建 0xDEADBEEF?

代码: LUI t2, 0xDEADC # t2 = 0xDEADC000

ADDI t2, t2, -0x111 # t2 = 0xDEADBEEF

-0x111 在 RV32 通用寄存器里的补码表示是 0xFFFFFEFF

1) 上述代码用 RARS 软件执行, 在 RV32 下执行, t2=0xDEADBEEF

2) 上述代码用 RARS 软件执行, 在 RV64 下执行, t2=0xFFFFFFFFDEADBEEF

3) addi 一般规则要求立即数范围在 $-2^{11} \sim 2^{11}-1$

其他班的 PPT 中 RV32

LUI t2, 0xDEADC # t2 = 0xDEADC000

ADDI t2, t2, 0xFFFFFEFF # t2 = 0xDEADBEEF

4) 试用 RARS 验证其他班代码, 看是否得到一样的结果? **能**

在 RV32 和 RV64, 与我们的代码是等价的。

在 RV32 下执行, t2=0xDEADBEEF。

在 RV64 下执行, t2=0xFFFFFFFFDEADBEEF

修改程序

LUI t2, 0xDEADC # t2 = 0xDEADC000

ADDI t2, t2, 0xFFFFFFFFFFFFFEFF # t2 = 0xDEADBEEF

5) 试用 RARS 验证其他班代码, 看是否得到正确结果? **能**

修改程序

LUI t2, 0xDEADC # t2 = 0xDEADC000

ADDI t2, t2, 0xEEF # t2 = 0xDEADBEEF

6) 试用 RARS 验证其他班代码, 看是否得到一样的结果?为什么?

答: 出错 (不能), 因为超出 addi 的 12 位立即数 (含符号位) 的整数范围, 其中最高位是符号位。RARS 提示 "0xEEF": Unsigned value is too large to fit into a sign-extended immediate"

对于不带符号的 12 位数, 比如 0xEEF, 按正数+0xEEF 判断, 超出了正数范围。

负数则必须带负号, 而且 12 位 (含符号位) 的表示范围里面。

修改程序

LUI t2, 0xDEADC # t2 = 0xDEADC000

ADDI t2, t2, 0x00000EEF # t2 = 0xDEADBEEF

7) 试用 RARS 验证其他班代码, 看是否得到一样的结果?为什么?

答: 出错, 因为 0xEEF 和 0x00000EEF 默认作为正数识别超出了 addi 的最大正数范围, 最高位是符号位。RARS 提示 "0x00000EEF": Unsigned value is too large to fit into a sign-extended immediate

修改程序

LUI t2, 0xDEADC # t2 = 0xDEADC000

ADDI t2, t2, 0xFFFFFEFF # t2 = 0xDEADBEEF

8) 试用 RARS 验证其他班代码, 看是否得到一样的结果?

答: 出错, "0xFFFFFEFF": operand is out of range

修改程序

```
LUI t2, 0xDEADC      # t2 = 0xDEADC000
ADDI t2, t2, 0xFFFFFEEF # t2 = 0xDEADBEEF
```

9) 试用 RARS 验证其他班代码, 看是否得到一样的结果?

答: 出错, "**0xFFFFFFFFEEF**": operand is of incorrect type

10) 思考: 为什么 LUI 后面的立即数没有 ADDI 这种范围限制出错?

答案: 因为指令格式的立即数直接对应了立即数的高 20 位表示, 低 12 位为 0。不受范围约束, 其取值只要刚好填满 20 位即可, 最高代表符号位。对于 RV64, 需要符号扩展。

经验性总结:

1、addi: 一般立即数字段要符合正负数范围, 输入带正负号的真值。

2、addi 的立即数直接被识别成补码需要同时满足条件:

1) 立即数是负数并且是以字或双字形式输入

2) 除了后面的低 12 位, 字或双字的高位必须都是 1

(RARS 会认为是处理好的负数补码, 不必继续转换成补码。在 RV64 中, 32 位的负立即数还会继续进行符号扩展, 前面有 FFFFFFFF。)

例如: **0xFFFFFEEF** 和 **0xFFFFFFFFFFFFEEF** 直接被识别成补码, 等价于 -0x111
所以上面有两组代码和我们的代码等价。

其他班的 PPT 中

```
LUI t2, 0xDEADC      # t2 = 0xDEADC000
ADDI t2, t2, 0xFFFFFEEF # t2 = 0xDEADBEEF
```

4) 试用 RARS 验证其他班代码, 看是否得到正确结果? 能

在 RV32 和 RV64, 与我们的代码是等价的。在 RV32 下执行, t2=0xDEADBEEF。

在 RV64 下执行, t2=0xFFFFFFFFDEADBEEF

修改程序

```
LUI t2, 0xDEADC      # t2 = 0xDEADC000
ADDI t2, t2, 0xFFFFFFFFFFFFEEF # t2 = 0xDEADBEEF
```

5) 试用 RARS 验证其他班代码, 看是否得到正确结果? 能

11) 基于上述经验性总结, 写出产生 64 位常量 0x1122334455667B88 的 RISC-V 汇编代码, 并将该值存储到寄存器 x10 中, 并用 RARS 软件验证是否正确。

其他解法:

lui x10, 0x11223	#x10=0x0000 0000 1122 3000	lui x10,0x11223
addi x10, x10, 0x344	#x10=0x0000 0000 1122 3344	slli x10,x10,20
slli x10, x10, 32	#x10=0x1122 3344 0000 0000	lui x11,0x34455
lui x5, 0x55668	#x5 = 0x0000 0000 5566 8000	add x10,x10,x11
addi x5, x5, 0xFFFFFB88	#x5 = 0x0000 0000 5566 7B88	addi x10,x10,0xB667
add x10, x10, x5	#x10=0x1122 3344 5566 7B88	slli x10,x10,8
		addi x10,x10,0xB8
		slli x10,x10,4
		addi x10,x10,0x8

注意: addi x5,x5,0xFFFFFB88 在RARS汇编器中等价于

addi x5,x5,-0x478

5、对于以下 C 语句，编写相应的 RISC-V 汇编代码。

B[8] = A[i - j]; //long long int A[1024], B[1024];

假设：为变量 i 和 j 分配寄存器 x28 和 x29, A 和 B 的基地址分别在寄存器 x10 和 x11 中。

```
sub x30, x28, x29    #i-j
slli x30, x30, 3      #8*(i-j)
add x30,x10,x30      #&A[i-j]-->x30
ld x30, 0(x30)        #A[i-j]-->x30
sd x30, 64(x11)       #x30-->B[8]
```

点评：数组元素之间赋值，需要先 load 后 store，这个题容易错。

6、RV64 中，给定 C 语言，翻译成 RV 语句。按照第一组例子完成后续代码。

注意：a 保存在 x22, b 保存在 x23

1) if (a > b) a += 1; bge x23, x22, Exit addi x22, x22, 1 Exit:	2) if (a < b) a += 1; bge x22, x23, Exit addi x22, x22, 1 Exit:
3) if (a == b) a += 1; bne x22, x23, Exit addi x22, x22, 1 Exit:	4) if (a != b) a += 1; beq x22, x23, Exit addi x22, x22, 1 Exit:
5) if (a >= b) a += 1; blt x22, x23, Exit addi x22, x22, 1 Exit:	6) if (a <= b) a += 1; blt x23, x22, Exit addi x22, x22, 1 Exit:

7、写出下列 RV 汇编代码的机器码，用十六进制描述

RV 代码	指令地址（十进制）	机器码（十六进制）
Loop: slli x10, x22, 3 add x10, x10, x25 ld x9, 0(x10) bne x9, x24, Exit addi x22, x22, 1 beq x0, x0, Loop Exit:	80000	0x003b1513
	80004	0x01950533
	80008	0x00053483
	80012	0x01849663
	80016	0x001b0b13
	80020	0xfe0006e3
	80024	-----

上述代码完成了什么功能？将上述代码补充完整并在 RARS 上测试（数组自己定义就行）。

答：在数组中查找第一个不等于特定值 k 的元素就跳出，**while(A[i]==k) i+=1;**

数组定义参考 https://blog.csdn.net/weixin_44126785/article/details/120450066

到

B 型	Imm[12,10:5]	rs2	rs1	funct3	imm[4:1,11]	opcode
-----	--------------	-----	-----	--------	-------------	--------

对于bne, 分析如下:

B型指令格式 $OP, rs1, rs2, offset$
 $bne x9, x24, Exit \Rightarrow bne x9, x24, 12$
 $offset = 12 = 6 \times 2\text{byte}$
 $Imm[12:1] = \underline{\underline{0000}} \underline{\underline{0000}} \underline{\underline{0110}}$, 拆开
 $rs2=x24$
 $rs1=x9$
 $0x \frac{0000000}{0} \frac{1}{1} \frac{000}{8} \frac{0100}{4} \frac{00}{9} \frac{01100}{6} \frac{10001}{6} \frac{1}{3}$

代码和运行结果(寄存器值截图)贴进来

对于beq, 分析如下:

B型指令格式 $OP, rs1, rs2, offset$
 $beq x0, x0, Loop \Rightarrow beq x0, x0, -20$
 $offset = -20 = -10 \times 2\text{byte}$
 $Imm[12:1] = \underline{\underline{[-10]_2}} = \underline{\underline{111111110110}}$

$rs1 = x0$
 $rs2 = x0$
 $0x \frac{111111}{f} \frac{000000000000}{e} \frac{0110}{000} \frac{1100011}{6} \frac{e3}{3}$

.data

v:

.dword 2, 2, 6, 7, 2, 2

.text

la x25, v #x25 里面存储数组的首地址

addi x24, x0, 2 # k=2

addi x22, x0, 0 #下标, i从0开始

Loop: slli x10, x22, 3

add x10, x10, x25

ld x9, 0(x10)

bne x9, x24, Exit

addi x22, x22, 1

beq x0, x0, Loop

Exit:

F:\riscv2.asm - RARS 1.5

File Edit Run Settings Tools Help

Edit Execute

riscv2.asm

```

1 .data
2 v:
3 .dword 2, 2, 6, 7, 2, 2
4 .text
5
6 la x25, v #x25 里面存储数组的首地址
7 addi x24, x0, 2 # 初始的下标 k=2
8 addi x22, x0, 0 # 下标, i从0开始
9
10 Loop: slli x10, x22, 3
11 add x10, x10, x25
12 ld x9, 0(x10)
13 bne x9, x24, Exit

```

Line: 1 Column: 1 Show Line Numbers

Registers	Floating Point	Control and Status
Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x0000000000000000
sp	2	0x000000007fffffc
gp	3	0x0000000010008000
tp	4	0x0000000000000000
t0	5	0x0000000000000000
t1	6	0x0000000000000000
t2	7	0x0000000000000000
s0	8	0x0000000000000000
s1	9	0x0000000000000006
a0	10	0x0000000010010010
a1	11	0x0000000000000000
a2	12	0x0000000000000000
a3	13	0x0000000000000000
a4	14	0x0000000000000000
a5	15	0x0000000000000000
a6	16	0x0000000000000000
a7	17	0x0000000000000000
s2	18	0x0000000000000000
s3	19	0x0000000000000000
s4	20	0x0000000000000000
s5	21	0x0000000000000000
s6	22	0x0000000000000002
s7	23	0x0000000000000000
s8	24	0x0000000000000002
s9	25	0x0000000010010000
s10	26	0x0000000000000000
s11	27	0x0000000000000000
t3	28	0x0000000000000000
t4	29	0x0000000000000000
t5	30	0x0000000000000000
t6	31	0x0000000000000000
pc		0x00000000040002c

$x9=6$

$x22=2$

$x22$ 是下标*i*

点评: 单纯补充 RV 代码并不难, 但其实 while 语句里面缺少了对数组下标越界的判断, 实际编程的时候需要补充这一段, 以免访问内存出错。

8、写一个 SUM 函数，对 1 到 100 进行加和，并用 RARS 运行完整程序，注意要有函数调用，将代码和 RARS 运行结果贴图。

```

addi x10,x0,100 #常数 100, 作为 Sum 参数
addi x11,x0,0   #初始化和为 s=0, 也是返回值
jal x1, Sum     #函数调用
jal x0, Exit    #一般这里是子程序调用继续执行的一条语句,
                  #本题程序结束, 所以直接跳到 Exit 即可

```

Sum:

```

      addi x5,x0,1      #初始化 i=1
      Loop:             #循环开始标签
      blt x10, x5, Finish #初始化 i>100, 跳出循环
      add x11,x11,x5    #s=s+i
      addi x5,x5,1       #i=i+1
      jal x0, Loop       #继续循环
      Finish:            #循环结束标签
      jalr x0, 0(x1)     #函数调用返回

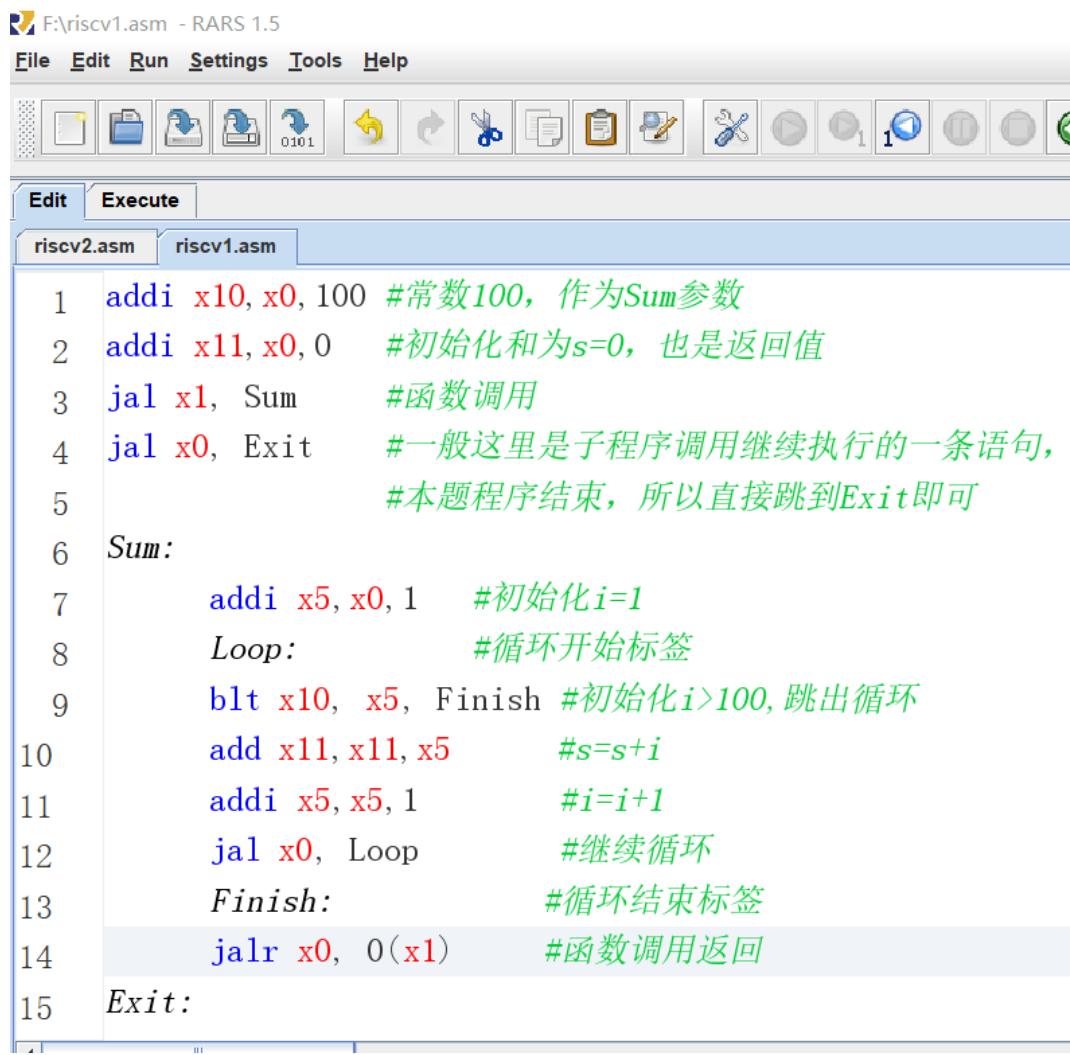
```

Exit:

Registers	Floating Point	Control and Status
Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x00000000400000c
sp	2	0x000000007fffffc
gp	3	0x0000000010008000
tp	4	0x0000000000000000
t0	5	0x0000000000000065
t1	6	0x0000000000000000
t2	7	0x0000000000000000
s0	8	0x0000000000000000
s1	9	0x0000000000000000
a0	10	0x0000000000000064
a1	11	0x00000000000013ba

0x13ba 就是十进制 5050

1 + 2 + 3 + ... + 100
= 5050



The screenshot shows the RISC-V assembly editor interface. The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains various icons for file operations and assembly editing. The tabs at the top are labeled riscv2.asm and riscv1.asm, with riscv1.asm currently selected. The assembly code in the editor window is identical to the one provided above, with line numbers 1 through 15. The code implements a recursive sum function with a base case of 100 and a loop that adds the current value of x5 to the result in x11.

```

1 addi x10, x0, 100 #常数 100, 作为 Sum 参数
2 addi x11, x0, 0   #初始化和为 s=0, 也是返回值
3 jal x1, Sum      #函数调用
4 jal x0, Exit     #一般这里是子程序调用继续执行的一条语句,
                  #本题程序结束, 所以直接跳到 Exit 即可
5
6 Sum:
7      addi x5, x0, 1      #初始化 i=1
8      Loop:             #循环开始标签
9      blt x10, x5, Finish #初始化 i>100, 跳出循环
10     add x11, x11, x5    #s=s+i
11     addi x5, x5, 1       #i=i+1
12     jal x0, Loop       #继续循环
13     Finish:            #循环结束标签
14     jalr x0, 0(x1)     #函数调用返回
15 Exit:

```

9、PPT 中 swap 函数调用例子存在无限循环的 BUG，如何修改成只执行一次调用就结束？

PPT 程序： addi x10,x0,21 addi x11,x0,20 jal x1,swap swap: add x6,x0,x10 add x10,x0,x11 add x11,x0,x6 jalr x0,0(x1)	第 1 种修改： addi x10,x0,21 addi x11,x0,20 jal x1,swap swap: add x6,x0,x10 add x10,x0,x11 add x11,x0,x6 jalr x0, 16 (x1)	第 2 种修改： addi x10,x0,21 addi x11,x0,20 jal x1,swap jal x0, exit (或 beq x0, x0, exit) swap: add x6,x0,x10 add x10,x0,x11 add x11,x0,x6 jalr x0,0(x1) exit:
--	---	---

鼓励大家用 RARS 软件亲自去验证各种答案😊😊😊

会用卡片 1，记忆卡片 2，尤其是几类指令的语法

卡片的使用，从大进制查指令
先看 opcode，再看 funct3，再看 funct6/7