厦门大学

《计算机组成原理》

课程实验报告

姓名：　庾晓萍

学院：信息学院

　系：软件工程

专业：　软件工程

学号：20420192201952

第四次实验　汇编语言以及指令系统实验

1. 实验环境

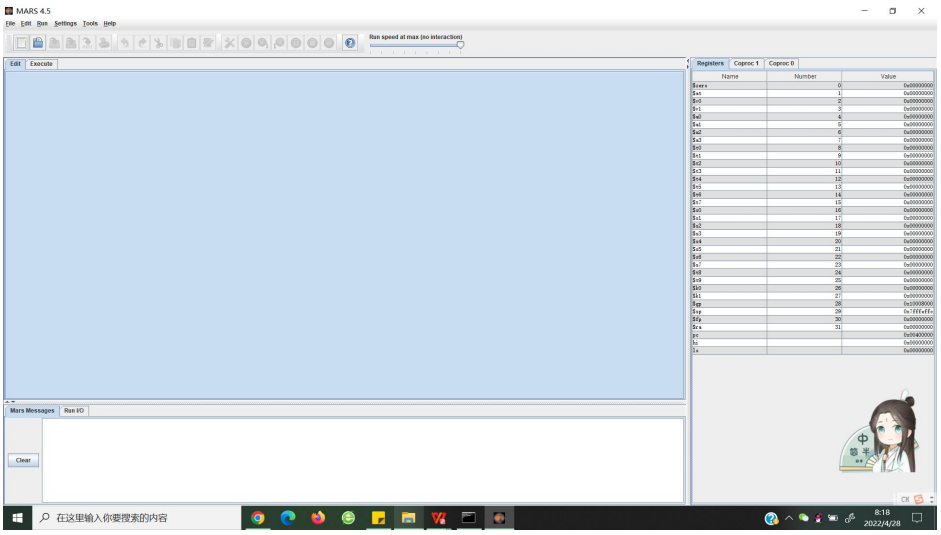
- **MIPS 汇编仿真器，RISV-V 汇编仿真器，Intel-X86(32)汇编仿真器**
- **Windows 系统或 Mac os 环境下运行 Logisim 软件**

2. 实验内容

2.1 验证实验　基于现有电路（第四次实验——CPU 设计实验（单周期 MIPS 处理器）.circ 以及汇编仿真器环境下验证实验结果并对电路和实验结果进行分析

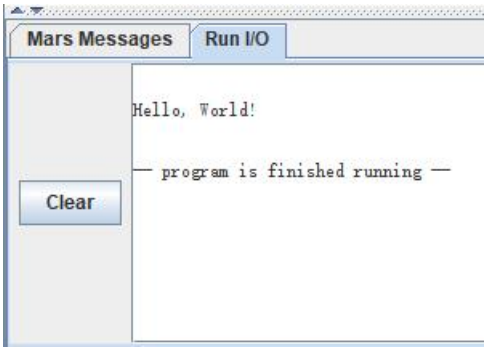
2.1.1 MIPS 汇编语言程序的运行

（1）运行 MIPS 汇编仿真器



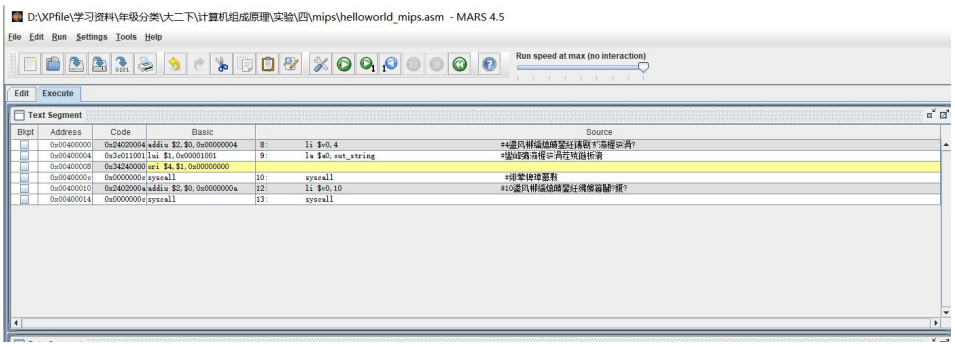
（2）运行第一个 MIPS 汇编语言程序

在 MARS 汇编仿真器中打开 helloworld_mips.asm。对 helloworld_mips.asm 源程序进行汇编（点击：Assemble）。运行汇编后的程序（点击：Go），在下面的界面中显示“Hello, World!”。



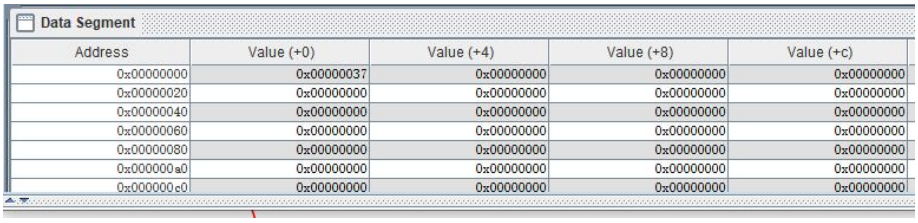
也可以单步执行程序：首先点击：Reset，此时将第一条指令背景置黄色，表示从第一条指

令开始执行；然后点击：Step，每点击一次 Step，执行一条指令；直到所有指令执行完毕。



（3）求累加和的 MIPS 汇编语言程序

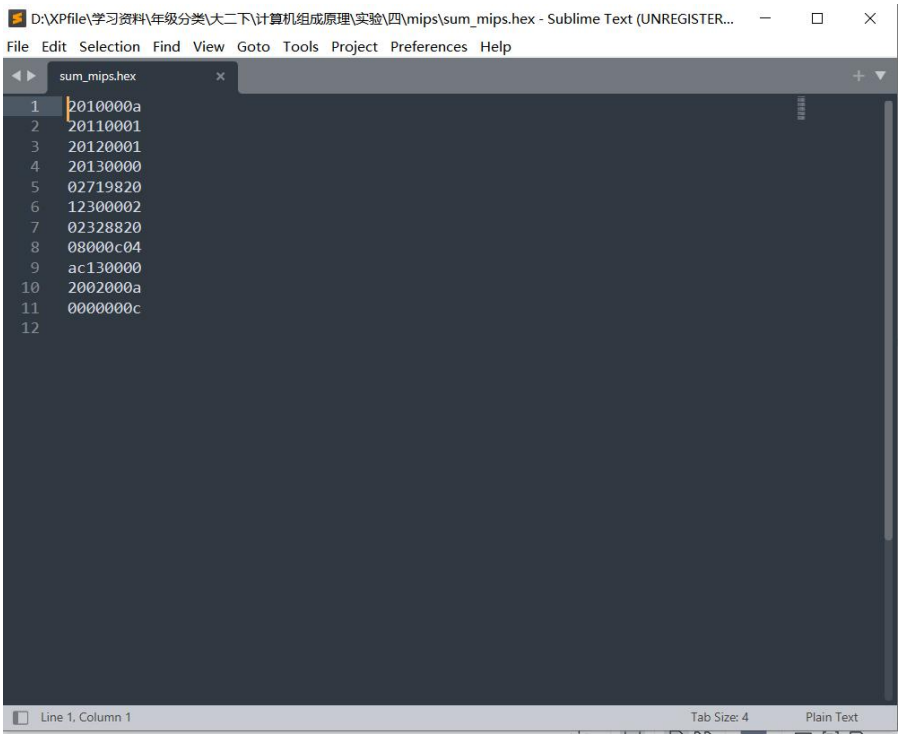
在 MARS 汇编仿真器中打开 sum_mips.asm。对 sum_mips.asm 源程序进行汇编（点击：Assemble）运行汇编后的程序（点击：Go）观看数据段中的内容。



Data Segment

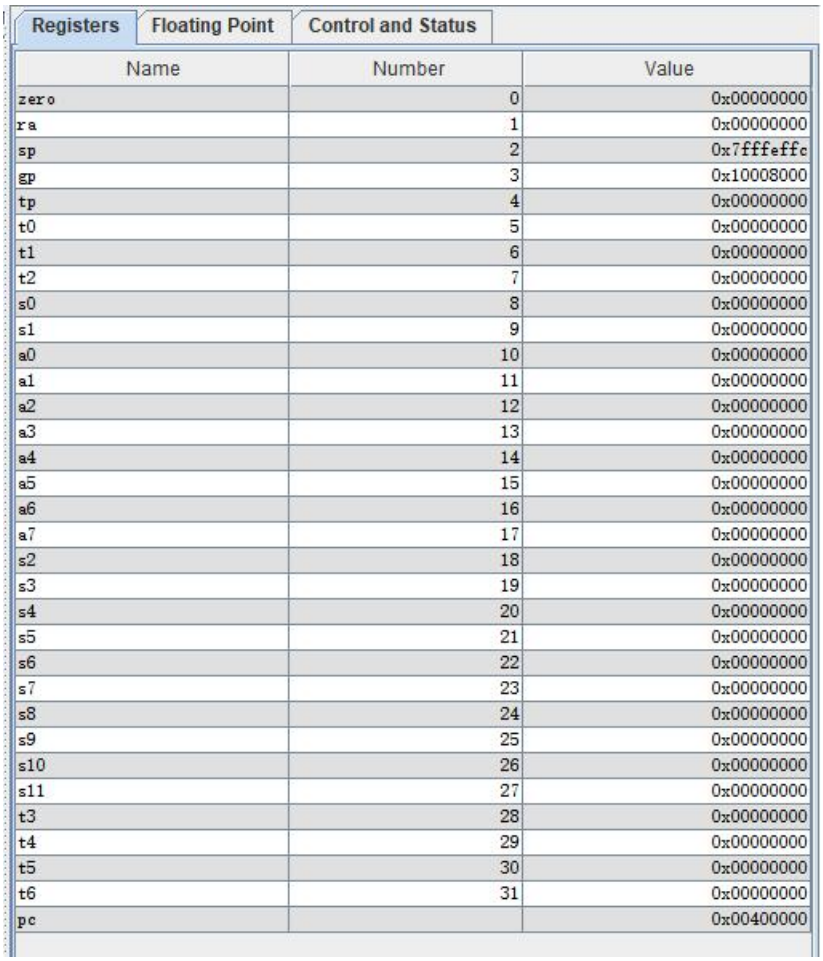
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x00000000	0x00000037	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000
0x00000060	0x00000000	0x00000000	0x00000000	0x00000000
0x00000080	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a0	0x00000000	0x00000000	0x00000000	0x00000000
0x000000c0	0x00000000	0x00000000	0x00000000	0x00000000

（4）导出汇编后的机器码



2.1.2 RISC-V 汇编语言程序的运行

（1）运行 RISC-V 汇编仿真器

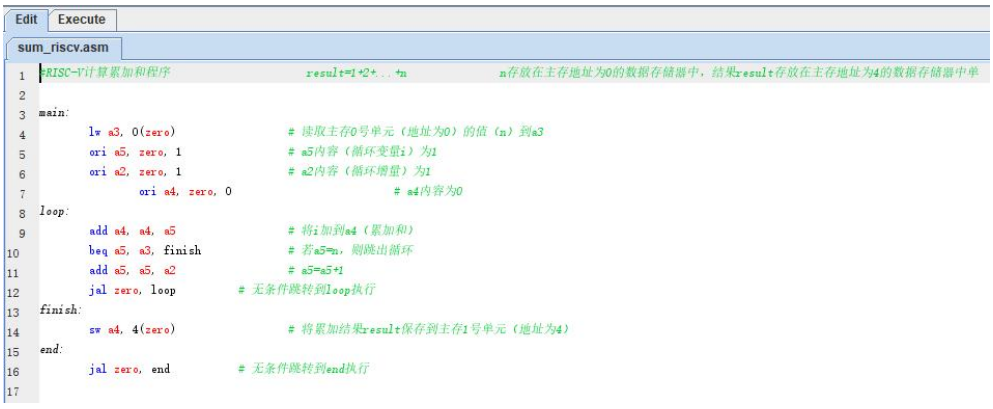


Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400000

（2）RISC-V 求累加和程序

利用前面的 RISC-V 的 9 条指令，编写一个计算 1+2+…+n 的累加和程序，从数据存储器地址为 0 的存储单元中读入参数 n，通过循环累加的算法计算累加和，结果保存到数据存储器地址为 4 的存储单元中。

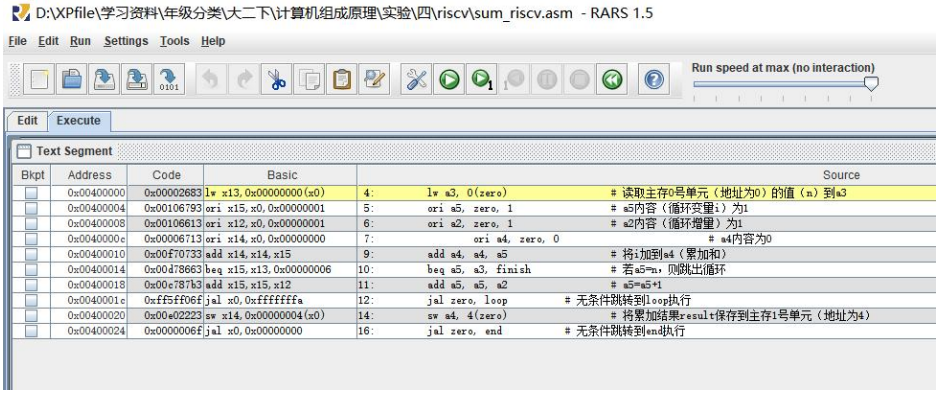


Edit | Execute

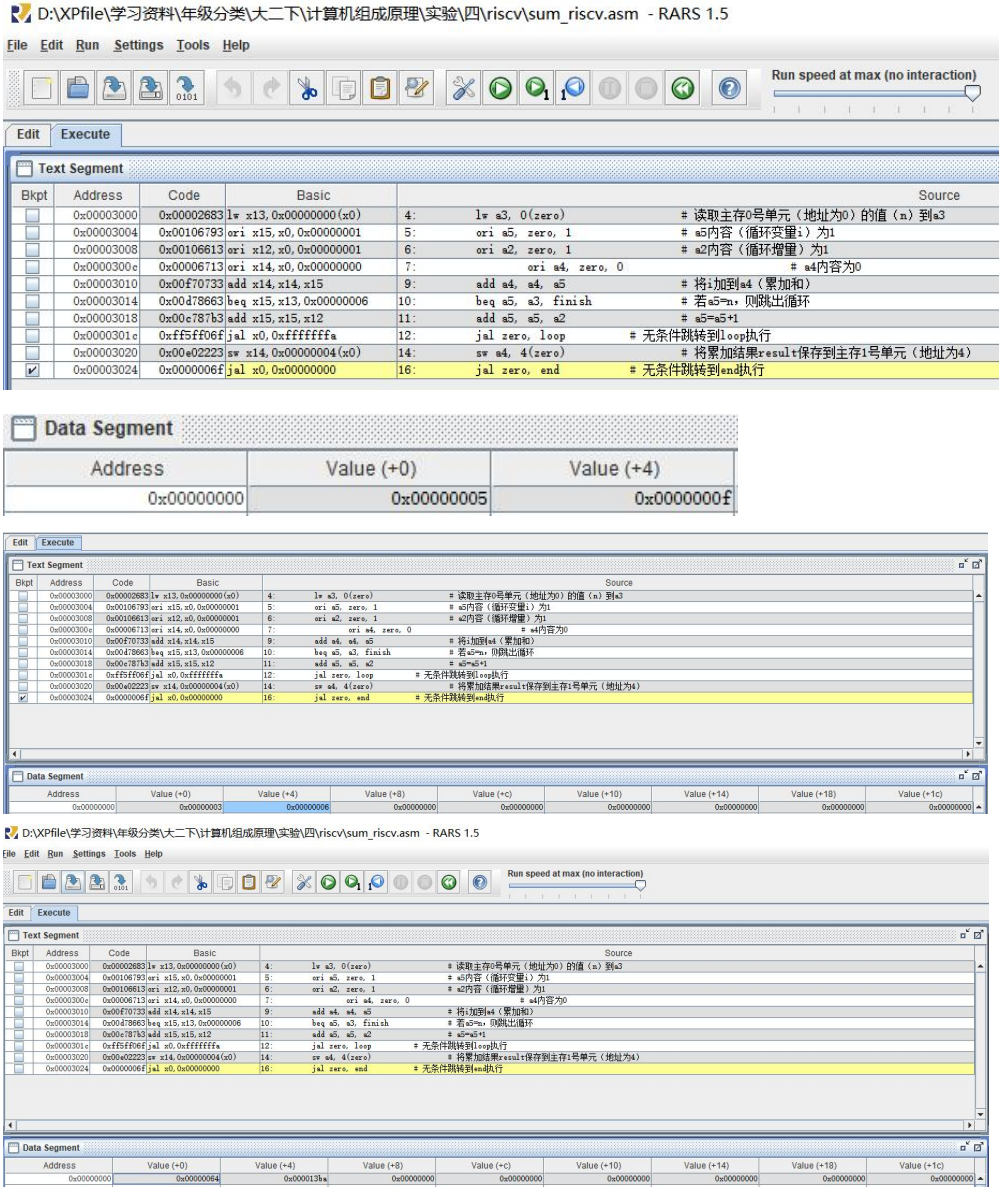
sum_riscv.asm

```
#RISC-V计算累加和程序                    result=1+2+...+n          n存放在主存地址为0的数据存储器中，结果result存放在主存地址为4的数据存储器中单
main:
        lw a3, 0(zero)              # 读取主存0号单元（地址为0）的值（n）到a3
        ori a5, zero, 1             # a5内容（循环变量i）为1
        ori a2, zero, 1             # a2内容（循环增量）为1
                ori a4, zero, 0                     # a4内容为0
loop:
        add a4, a4, a5              # 将i加到a4（累加和）
        beq a5, a3, finish          # 若a5=n，则跳出循环
        add a5, a5, a2              # a5=a5+1
        jal zero, loop          # 无条件跳转到loop执行
finish:
        sw a4, 4(zero)              # 将累加结果result保存到主存1号单元（地址为4）
end:
        jal zero, end           # 无条件跳转到end执行
```

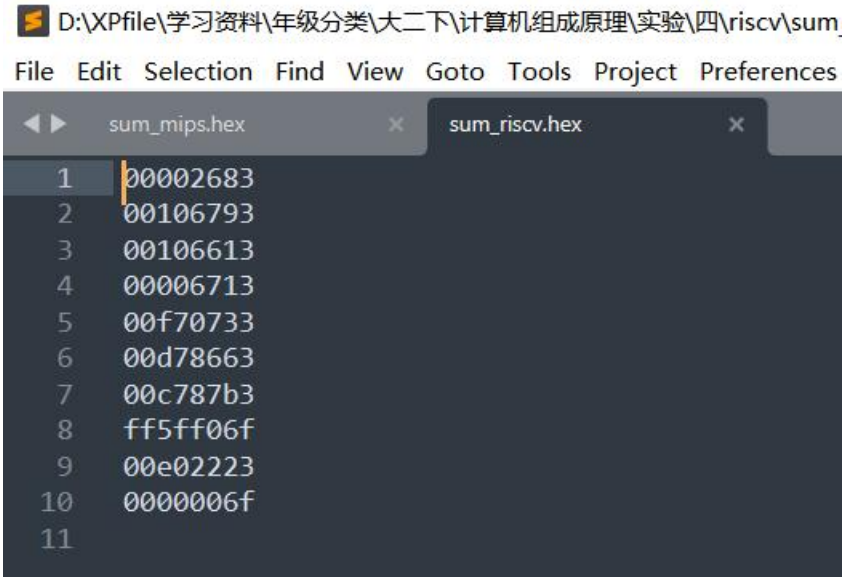
（3）在 RARS 中汇编“sum_riscv.asm”



（4）在 RARS 中运行“sum_riscv.asm”



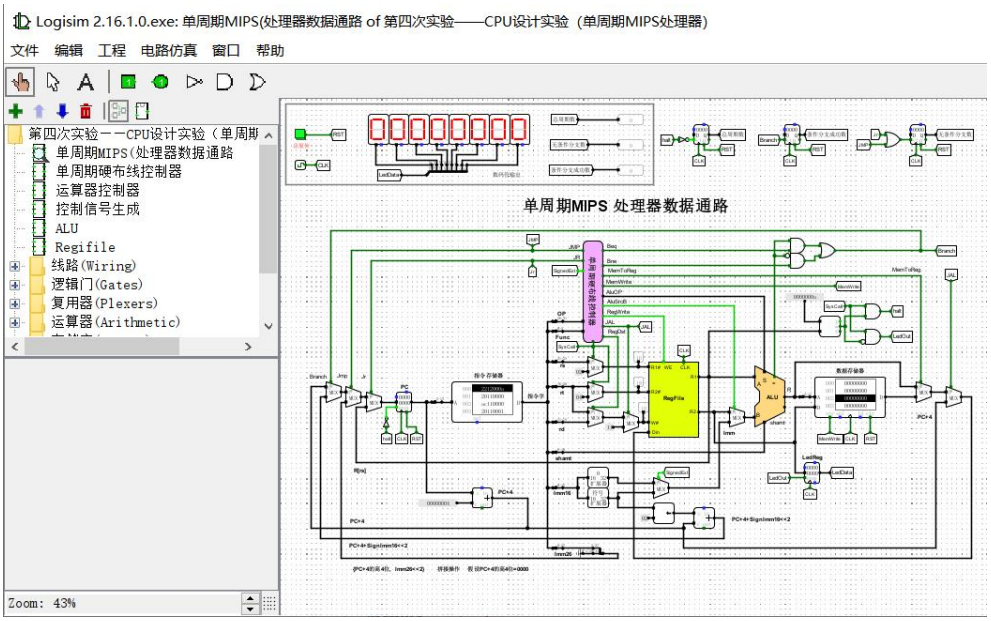
（5）将汇编后的机器码，导出到一个文本文件中



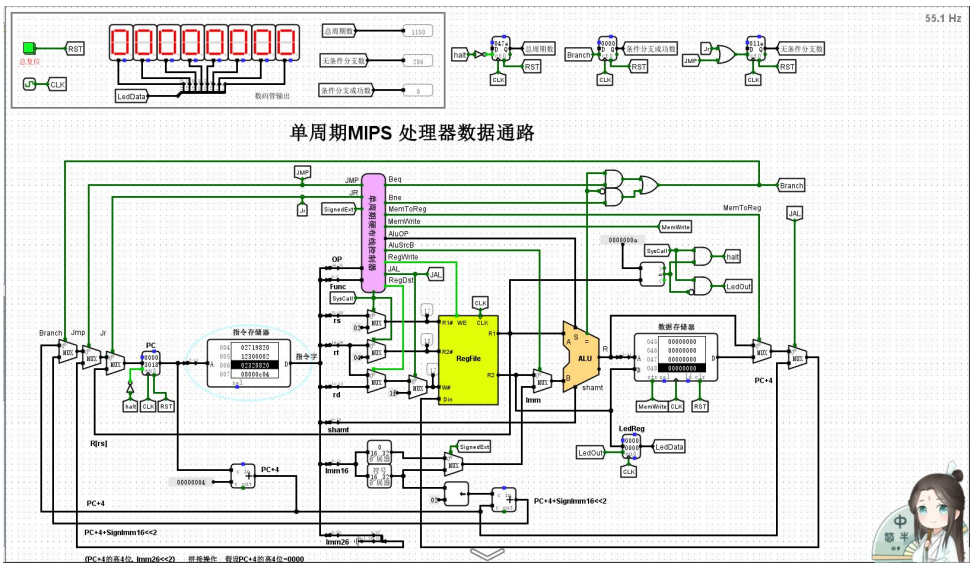
2.1.3 MIPS 单周期处理器数据通路

（1）在单周期 MIPS 处理器的数据通路上运行程序

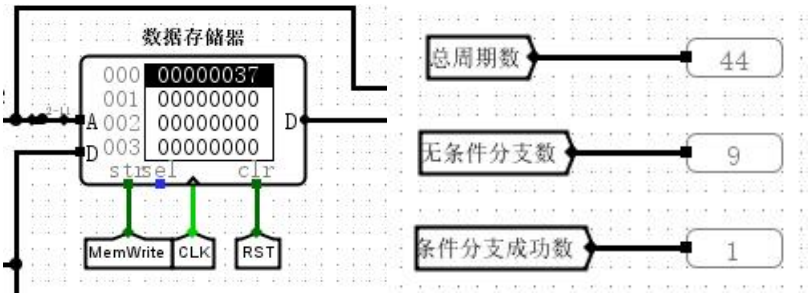
修改前面导出的机器码文件 sum_mips.hex，增加：v2.0 raw。打开 Logisim，在 Logisim 中增加寄存器文件库“cs3410.jar”打开设计文件“第四次实验——CPU 设计实验（单周期 MIPS 处理器）.circ。



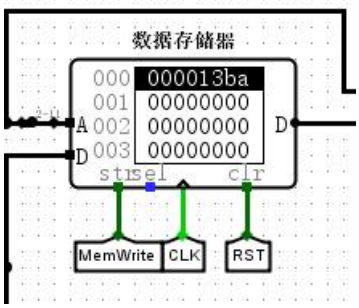
在设计文件的指令存储器中装入机器码文件 sum_mips.hex。设置时钟频率=1KHz，按 Ctrl+K 启动时钟（Ticks Enabled），程序开始执行。

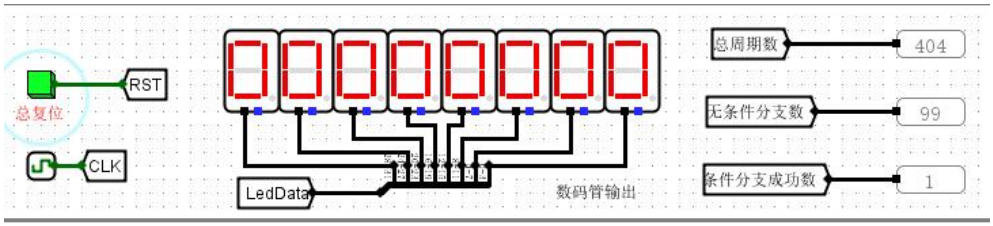


按总复位，程序重新执行。等程序执行完毕，观察数据存储器第 0 号单元的值（累加和，正确值为 37h=55），以及程序运行的总周期数、无条件分支数、条件分支成功数。



修改 sum_mips.asm 程序，计算 1+2+…+100=5050，并在单周期 MIPS 处理器数据通路上运行，观察结果是不是 13bah（5050）？

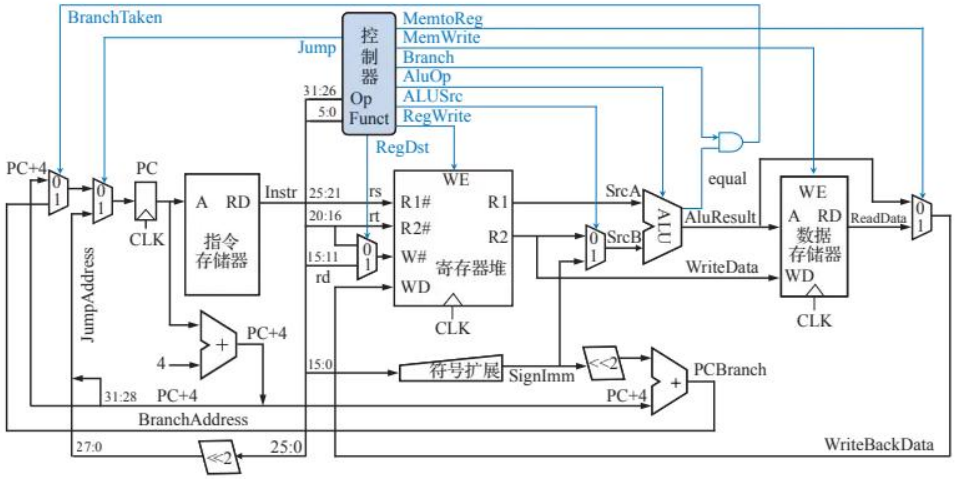




（4）要求：

① 分析单周期 MIPS 处理器数据通路电路原理：与教材图 6.25 相比，该单周期

MIPS 处理器数据通路增加了哪些部件？为什么要增加这些部件？与教材表 6.9 相比，该单周期 MIPS 处理器数据通路的控制器增加了哪几个控制信号？为什么要增加这些控制信号？



答：实现构建 CPU 所需的主要功能部件包括：程序计数器 PC、指令存储器 IM、数据存储器 DM，立即数扩展器、地址转移逻辑 NPC、运算器 ALU。同时增加了其他的数据通路、功能模块、多路选择器、控制信号。控制信号包含 ADD、SUB、AND、OR、sll、srl、sra、slt、lui、RegWriteBack、Alusrc、RegDst、disp、JumBranch、beq、bne、SingnedIMM、MemRead、Memwrite。

②累加和程序（1+2+…+10）运行后，总周期数、无条件分支数、条件分支成功数分别是 44、9、 1，请分析为什么是这个值？
答：总周期数是 main 中的 4 次，fin 中的 2 次，loop 中 36+2 次，一共 44 次。无条件分支数是之前进入循环的 9 次。条件分支成功数为 1 时，即 s1=s0，转到 finish，将 s3 存储到地址为 0 的存储单元中。10 号系统调用程序退出。

③累加和程序（1+2+…+100）运行后，总周期数、无条件分支数、条件分支成功数分别是多少？并分析为什么是这个值？
答：总周期数为 404，无条件分支数为 99，条件分支成功数为 1。总周期数是 main 和 fin 中的 6 次，loop 中的 99*4+2=398 次，一共 404 次。无条件分支数是之前进入循环的 99 次。条件分支成功数为 1 时，即 s1=s0，转到 finish，将 s3 存储到地址为 0 的存储单元中。10 号系统调用程序退出。

2.2 设计实验

在不同 ISA 指令架构（MIPS，RISC-V，Intel-X86）编译环境下完成对应程序设计，将对应的程序分别命名为:

a) **MIPS:Fib-mips.asm 和 Sort-mips.asm,**

b) **RISC-V: Fib-riscv.asm 和 Sort-riscv.asm**

c) **Intel-X86: Sort-x86.asm**

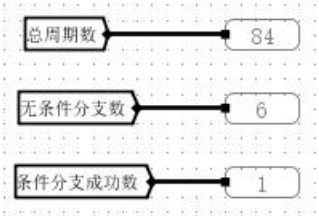
2.2.1 MIPS 汇编语言程序设计

a) 计算费波那契数列的程序

① 核心算法

```
.text
main:
        li $t0,7 #斐波那契的n=7
        li $t1,0 #初始化f(0)=0
        li $t2,1 #初始化f(1)=1
        li $t4,1 #初始化f(2)=1
        li $t3,2 #int i=2
    sw $t1,0($zero)  #将f(0)置于缓存中
    blt $t0,2,out    #如果输入n<2，直接跳转out
    sw $t2,4($zero)  #将f(1)置于缓存中
    blt $t0,3,out    #如果输入n<3，直接跳转out
    sw $t2,8($zero)  #将f(2)置于缓存中
    blt $t0,4,out    #如果输入n<4，直接跳转out
loop:
    bgt $t3,7,out    #如果i>7，直接跳转out
        move $t1,$t2        #f(n-2)=f(n-1)
        move $t2,$t4        #f(n-1)=f(n)
        add $t4,$t1,$t2     #f(n)=f(n-1)+f(n-2),n从3开始
    addi $t3,$t3,1      #i++,i第一次变成3
    sll $t0, $t3, 2     #偏移量 $t0 =i*4 sll是左移
    add $t0,$zero,$t0  #实际地址 = 偏移量$t0 + 基准地址
        sw $t4,0($t0)       #将f(n)写入到实际地址
    b loop
out:     #=========结果=========#
        li $v0,10 #结束程序
        syscall  #执行
```

② 运行结果（n=7）




```
000  00000000 00000001 00000001 00000002  00000003 00000005 00000008 0000000d
008  00000015 00000000 00000000 00000000  00000000 00000000 00000000 00000000
```

Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x00000001	0x00000001	0x00000002	0x00000003	0x00000005	0x00000008	0x0000000d
0x00000015	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

b) 冒泡排序算法的程序

①从大到小排序

1.核心代码

```
.text
main:
        li $t0,7 #斐波那契的n=7
        li $t1,0 #初始化f(0)=0
        li $t2,1 #初始化f(1)=1
        li $t4,1 #初始化f(2)=1
        li $t3,2 #int i=2
        sw $t1,0($zero)   #将f(0)置于缓存中
        blt $t0,2,out     #如果输入n<2，直接跳转out
        sw $t2,4($zero)   #将f(1)置于缓存中
        blt $t0,3,out     #如果输入n<3，直接跳转out
        sw $t2,8($zero)   #将f(2)置于缓存中
        blt $t0,4,out     #如果输入n<4，直接跳转out
loop:
        bgt $t3,7,out     #如果i>7，直接跳转out
        move $t1,$t2        #f(n-2)=f(n-1)
        move $t2,$t4        #f(n-1)=f(n)
        add $t4,$t1,$t2     #f(n)=f(n-1)+f(n-2),n从3开始
        addi $t3,$t3,1      #i++,i第一次变成3
        sll $t0, $t3, 2     #偏移量 $t0 =i*4 sll是左移
        add $t0,$zero,$t0   #实际地址 = 偏移量$t0 + 基准地址
        sw $t4,0($t0)       #将f(n)写入到实际地址
        j loop
out:     #=========结果=========#
        li $v0,10 #结束程序
        syscall  #执行
```

2.运行结果

000	0000000a	0000000a	00000009	00000008	00000007	00000006	00000005	00000004
008	00000003	00000002	00000001	00000000	00000000	00000000	00000000	00000000

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x0000000a	0x0000000a	0x00000009	0x00000008	0x00000007	0x00000006	0x00000005	0x00000004
0x00000020	0x00000003	0x00000002	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

②从小到大排序

1.核心代码

```
oLop:
                                #外层循环开始
      addi $s2, $zero, 10        #$s2 就是 int j = 10
iLop:
                                   #内层循环开始
          sll $t0, $s2, 2          #偏移量j*4，初始是40
          add $t1, $s1, $t0        #A[j]的实际内存地址
          addi $t2, $t1, -4        #A[j-1]的实际内存地址
          lw $t3, 0($t1)           #$t3=A[j]的值
          lw $t4, 0($t2)           #$t4=A[j-1]的值
          slt $t5, $t3, $t4        #若A[j-1] < A[j]
          beq $t5, $zero, afterSwap #为真交换，否则跳过
#swap
          lw $t6, 0($t1)          # tmp=A[j]
          sw $t4, 0($t1)           # A[j] = A[j-1]
          sw $t6, 0($t2)           # A[j-1]=tmp
afterSwap:
          addi $s2, $s2, -1         # j = j - 1
          slt $t0, $s0, $s2        #若i < j
          bne $t0, $zero, iLop     #继续内层循环
          addi $s0, $s0, 1         # i = i + 1
          slti $t0, $s0, 9        #若i<10
          bne  $t0, $zero, oLop    #则继续外层循环
                                  #外层循环结束
          li $v0, 10               #停
          syscall                  #机
```

2.运行结果


```
000  0000000a 00000001 00000002 00000003  00000004 00000005 00000006 00000007
008  00000008 00000009 0000000a 00000000  00000000 00000000 00000000 00000000
```

Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x0000000a	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005	0x00000006	0x00000007
0x00000008	0x00000009	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

2.2.2 RISC-V 汇编语言程序设计

a) 计算费波那契数列的程序

```
        ori a0,zero,10
        ori a1,zero,0
        ori a2,zero,1
        ori a3,zero,0
        ori a4,zero,0
        sw a1,0(zero)
        sw a2,4(zero)
        ori a5,zero,8
        ori a6,zero,4
        ori s2,zero,1
loop:

        add a3,a1,a2
        sw a3 0(a5)
        add a5,a5,a6
        add a1,a2,zero
        add a2,a3,zero
        add a4,a4,s2
        beq a4,a0,finish
        jal zero,loop
finish:
        jal zero,finish
```

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x00000000	0x00000000	0x00000001	0x00000001	0x00000002	0x00000003	0x00000005	0x00000008
0x00000020	0x00000015	0x00000022	0x00000037	0x00000059	0x00000000	0x00000000	0x00000000

b) 冒泡排序算法的程序

（1）从小到大

```
loopout:
        beq s0,s1,finish        #外层循环退出条件s0=s1
        ori s2,zero,1           #每次外层循环刚开始，设置s2=1
        ori s3,zero,0           #每次外层循环刚开始，设置s3=0，从一对元素开始判断
        loopin:
                lw s4,0(s3)
                lw s5,4(s3)
                slt s6,s5,s4    #如果s5<s4，那么s6=1，否则s6=0，为1需要交换值
                beq s6,zero,next        #如果s6=0，不需要交换，直接跳转向next，否则顺序执行
                #交换值的部分
                add a1,s4,zero  #a1=s4
                add s4,s5,zero  #s4=s5
                add s5,a1,zero  #s5=a1
        next:
                #送回内存
                sw s4,0(s3)
                sw s5,4(s3)
                add s3,s3,t0            #地址加4
                add s2,s2,a0            #变量加1
                add  s7,s0,s2           #s7=s0+s2     s7=j+i
                #j+i>n,退出
                slt t1,s1,s7            #s1<s7,t1=1,否则t1=0，t1=1退出循环
                beq t1,zero,loopin      #等于1，继续内层循环，否则外层变量加1，继续外层循环
                add s0,s0,a0
                jal zero,loopout
finish:
        jal zero,finish
```

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x00000000	0x00000001	0x00000002	0x00000004	0x00000006	0x00000007	0x00000008	0x00000009
0x00000020	0x00000004	0x0000000f	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

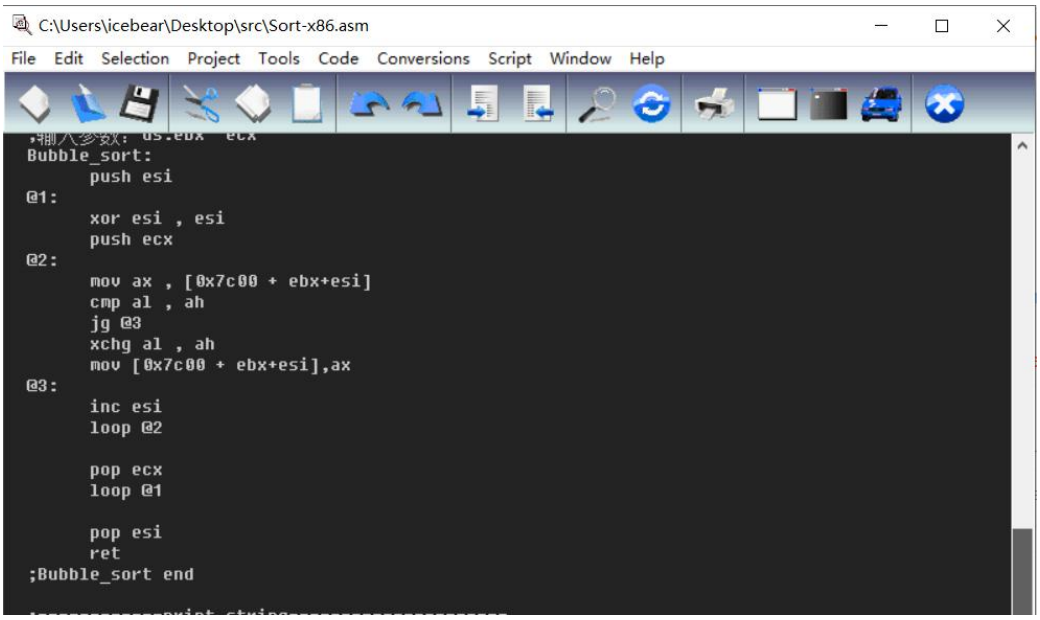
（2）从大到小

```
loopout:
        beq s0,s1,finish        #外层循环退出条件s0=s1
        ori s2,zero,1           #每次外层循环刚开始，设置s2=1
        ori s3,zero,0           #每次外层循环刚开始，设置s3=0，从一对元素开始判断
        loopin:
                lw s4,0(s3)
                lw s5,4(s3)
                slt s6,s5,s4    #如果s5<s4，那么s6=1，否则s6=0，为1需要交换值
                bne s6,zero,next        #如果s6=0，不需要交换，直接跳转向next，否则顺序执行
                #交换值的部分
                add a1,s4,zero  #a1=s4
                add s4,s5,zero  #s4=s5
                add s5,a1,zero  #s5=a1
        next:
                #送回内存
                sw s4,0(s3)
                sw s5,4(s3)
                add s3,s3,t0            #地址加4
                add s2,s2,a0            #变量加1
                add  s7,s0,s2           #s7=s0+s2     s7=j+i
                #j+i>n,退出
                slt t1,s1,s7            #s1<s7,t1=1,否则t1=0，t1=1退出循环
                beq t1,zero,loopin      #等于1，继续内层循环，否则外层变量加1，继续外层循环
                add s0,s0,a0
                jal zero,loopout
finish:
        jal zero,finish
```

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x00000000	0x0000000f	0x0000000d	0x0000000b	0x00000009	0x00000008	0x00000007	0x00000006
0x00000020	0x00000002	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

2.2.3 Intel x86 汇编语言程序设计

冒泡排序算法的程序



3. 实验提交

(1). 实验报告命名方式：例如：**30620192203840+孙明策-4pdf**

(2). 请将所有实验相关文件—实验报告（Word 文档）+三种 ISA 的汇编源程序打包至.zip 等压缩包，命名：学号+姓名-4.Zip.

(3). 实验报告上传路径：

- **计组（1）（2）请上传至曾文华老师对应 FTP 路径。**
- **卓越班请上传至张海英老师对应 FTP 路径。**

(4). 报告提交时间：deadline: 2022/05/05 晚 12：00