实验 6：CPU 设计综合

一、实验目的

1、 掌握不同指令数据通路的实现方式。
2、 掌握 CPU 基本结构，并学习不同部件级联调试方法。
3、 掌握 RISC-V 汇编程序设计，并转换成机器代码的方法。

二、实验环境

Logisim-ITA V2.16.1.0。
RARS：RISC-V 模拟器工具 https://github.com/thethirdone/rars

三、实验内容

指令存储器加载.hex文件，数据存储器加载.dat文件

图 1 是支持 RISC-V 基本整数指令模块 RV32I 中 9 条指令的单周期处理器原理图。

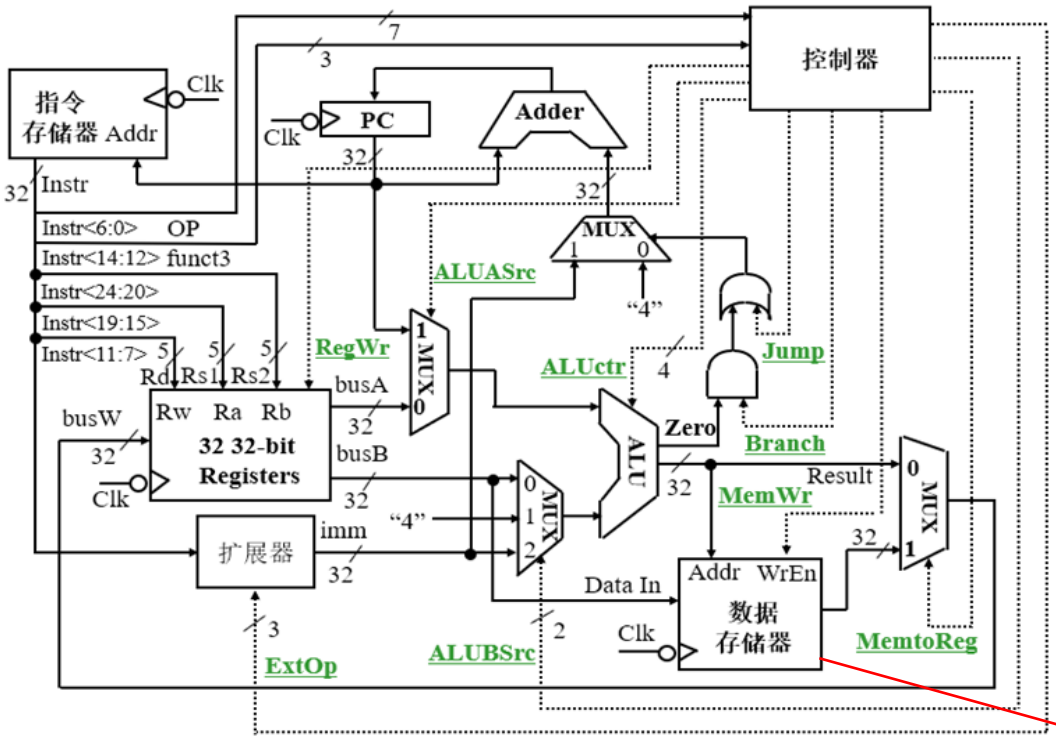

图 1　RV32I 单周期处理器原理图

用RAM，取2-25位做地址,高电平

图 1 中支持的 9 条指令的指令格式及编码如表 1 所示。根据图 1 所示的原理图，设计实现一个支持表 1 中 9 条指令的单周期处理器，并通过在该处理器上运行一个特定的程序来验证处理器设计的正确性。

表 1　RV32I 中 9 条指令的编码格式

指　令	funct7 (31-25)	rs2 (24-20)	rs1 (19-15)	funct3 (14-12)	Rd (11-7)	Op (6-0)
add rd, rs1, rs2	0000 000	rs2	rs1	000	rd	011 0011
slt rd, rs1, rs2	0000 000	rs2	rs1	010	rd	011 0011
sltu rd, rs1, rs2	0000 000	rs2	rs1	011	rd	011 0011
ori rt, rs1, imm12	imm{11:0}		rs1	110	rd	001 0011
lui rd, imm20	imm[31:12]				rd	011 0111
lw rd, rs1, imm12	imm[11:0]		rs1	010	rd	000 0011
sw rs1, rs2, imm12	imm[11:5]	rs2	rs1	010	imm[4:0]	010 0011
beq rs1, rs2, imm12	imm[12][10:5]	rs2	rs1	000	imm[4:1][11]	110 0011
jal rd, imm20	imm[20][10:1][11[19:12]				rd	110 1111

为了便于调试，建议增加复位按钮，用于复位存储器中的数据。根据指令格式将汇编程序转换为机器代码表示，并写入指令存储器中，启动程序执行，然后写出执行每条指令后寄存器中保存的值，并验证执行结果的正确性。参照以下累加程序的验证过程，使用冒泡排序程序进行正确性验证。累加程序的验证过程如下。

1、编写验证程序的汇编代码

编写一个计算 1+2+…+n 的累加和程序，从第 0 单元中读入参数 n，通过循环累加的算法计算累加和，结果保存到第 4 单元（假定 Logisim 的 RAM 容量为 64K*32b=256KB，则相当于其中的第 0x0001 单元）。该程序的汇编代码如下：

```
        lw x3, 0x0(x0)      # 读取主存地址 0x0000 处的 n 到 x3
        ori x5, x0, 0x1     # x5 内容（循环变量 i）为 1
        ori x2, x0, 0x1     # x2 内容（循环增量）为 1
loop:
        add x4, x4, x5      # 将 i 加到 x4（累加和）
        beq x5, x3, finish  # 若 x5=n，则跳出循环
        add x5, x5, x2      # x5=x5+1
        jal x0, loop        # 无条件跳转到 loop 执行
finish:
        sw x4, 0x4(x0)      # 将累加结果保存到 0x0001 单元
end:    jal x0, end         # 无条件跳转到 end 执行
```

2、将汇编代码转换为机器代码

将编写好的汇编程序读入 RARS 中进行汇编调试，通过仿真执行后，把汇编程序转换成机器代码，写入到一个机器代码文件中。具体步骤如下。

1）存储模式设置

由于单周期 CPU 设计中没有考虑内存管理单元，且指令存储器和数据存储器分离，所有地址为主存物理地址，并且起始地址都是 0，因此，为了使测试程序能在 RARS 中仿真运行，需配置 RARS 模拟器中的 RISC-V 虚拟存储模式。可以将菜单 Setting 中的 Memory Configuration 选项设置为 Compact，Data at Address 0。这样数据段的起始地址就从 0 开始，如图 2 所示。

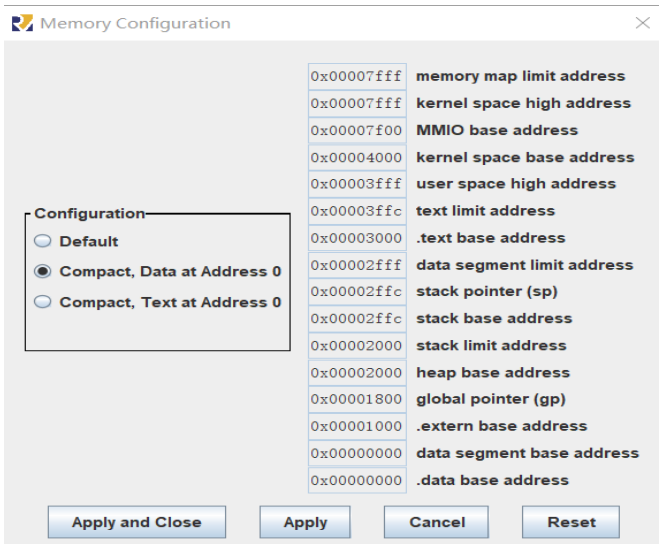

图 2　Memory Configuration 选项设置

2）　仿真执行并生成机器代码

在 RARS 中读入或编写汇编程序并保存后，通过执行 Aseemble（F3）命令，将汇编代码编转换成机器代码。将初始地址设置为 0，进行仿真运行。仿真过程参见图 3 和图 4。

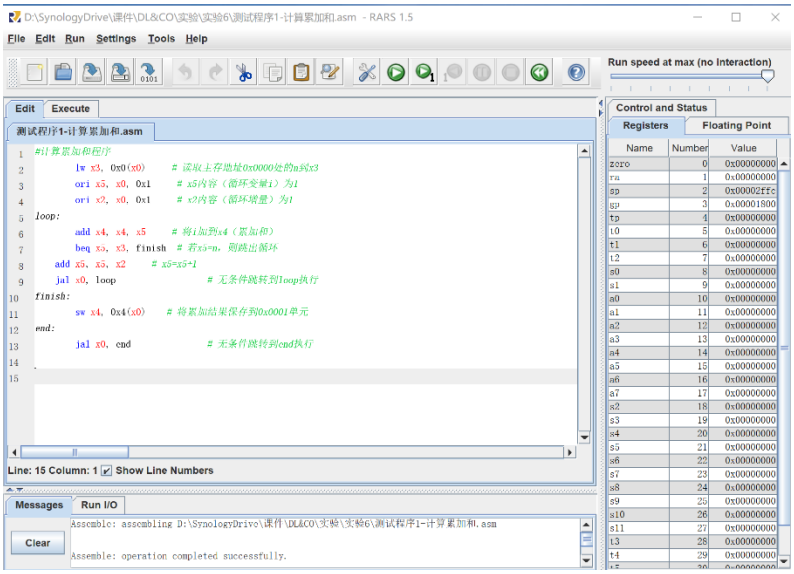

图 3 汇编程序编辑窗口查看

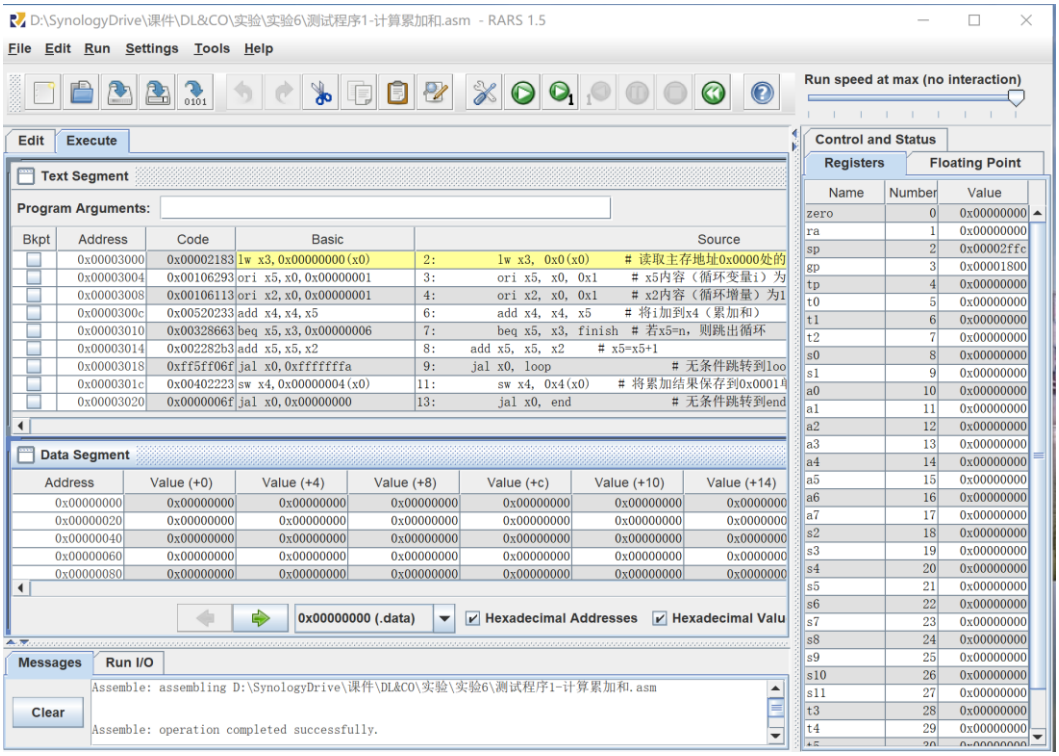

图 4 程序编译后指令地址、机器码及执行时寄存器内容查看

在地址 0x0000 处设置参数 n（如 64），点击单步执行 Step(F7)，可以在每一条指令执行后，查看各寄存器中的内容；选择连续运行 Go(F5)，由于程序中使用死循环作为结束语句，需单击暂停按钮 Pause(F9)或停止按钮 Stop(F11)，终止死循环，查看执行结果。观察到数据存储器的 0x0004 处，数据为 0x00013ba。十进制数为 5050。验证了累加程序的正确性。如图 5 所示。

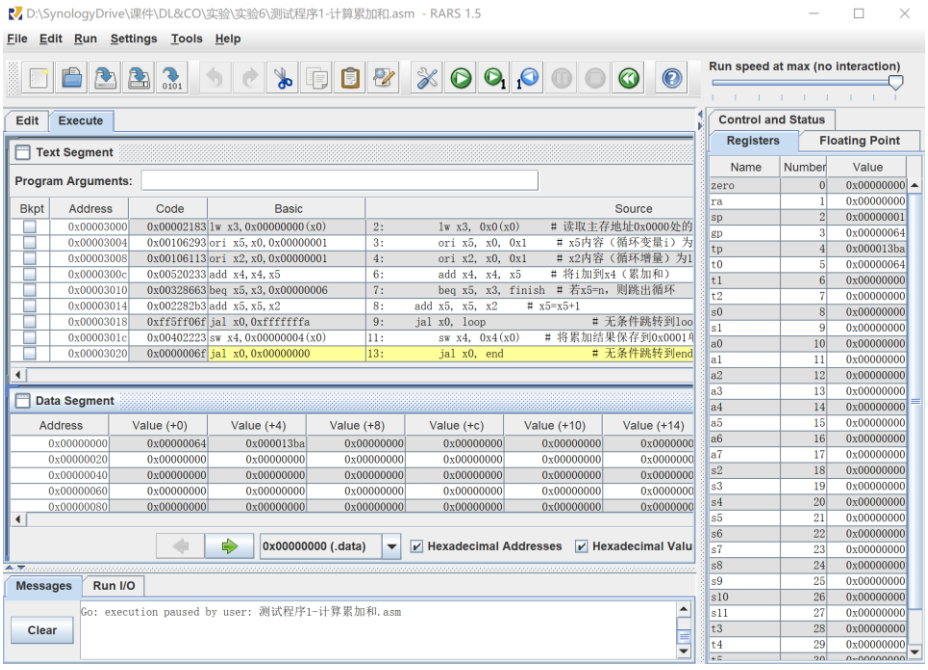

图 5 程序仿真时指令地址和机器码查看

3）机器代码导出

利用图 6 所示的 RARS 的 File 菜单中的 Dump Memeory To File 功能，可以将汇编程序对应的机器指令和数据段中的数据导出，选择导出类型为 16 进制文本格式。

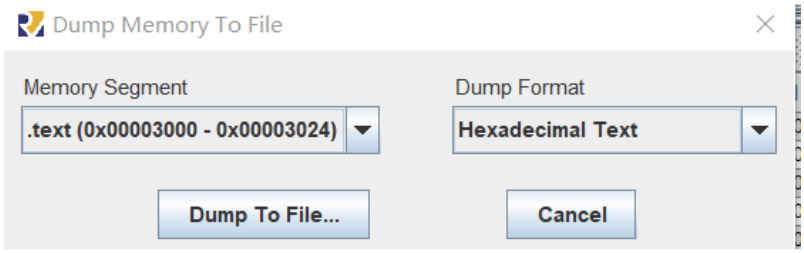

图 6　Dump Memeory To File 设置

将指令和数据导出到某个文本文件中后，插入首行内容为 V2.0 raw。这样，可以通过在 Logisim 中选择 Load Image 功能，将导出的文件直接加载到 ROM 组件中。假设将上述图 5 中的机器代码导出到文件 test.o 中，则 test.o 中的内容如下：

v2.0 raw

00002183

00106293

00106113

00520233

00328663

002282b3

ff5ff06f

00402223

0000006f

3、将机器代码写入 ROM 并启动执行

将上述导出的二进制机器编码文件（如 test.o）装入 Logisim 的 ROM 中启动执行，主要包含以下步骤：

1）在 Logisim 中，打开 CPU 设计电路图，用鼠标右键点击 IFU 模块，选择 View IFU；在 IFU 的 ROM 模块（作为指令存储器）上用鼠标右键点击，选择装载文件 load image，装入机器代码文件。如图 7 所示。

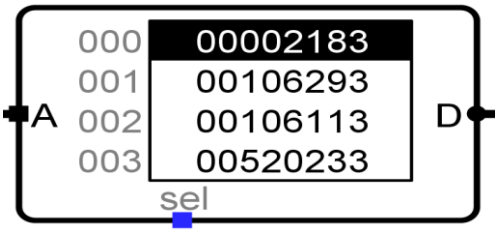

图 7　将指令装入指令存储器

2）在 CPU 电路图中，选中数据存储器 RAM，用鼠标右键点击，选择 Edit Content，在 0x0000 位置处，写入参数值 n（如 0xa）。如图 8 所示。

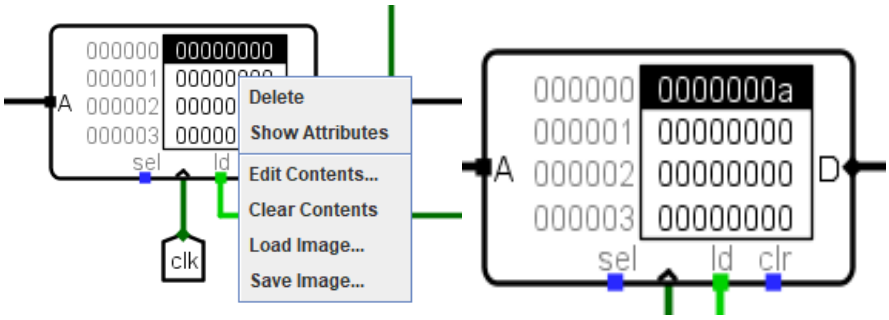

图 8　在第 0 单元设置参数

3）在 Logisim 的仿真菜单下，选择合适的频率，如 1kHz，选中 Ticks Enable，开始自动执行机器代码，程序执行结束后查看 RAM 中 0x0001 处的结果。如图 9 所示。

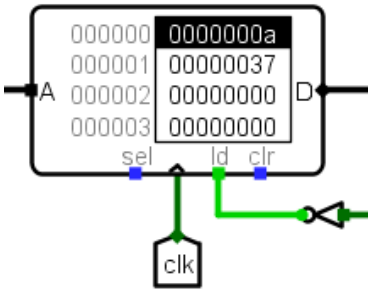

图 9 查看结果为 0x37

4）修改参数 n 为 0x64 时，得到结果为 0x13ba。如图 10 所示。

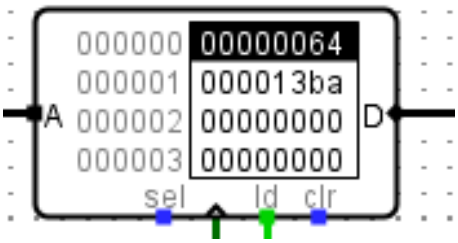

图 10 查看结果为 0x13ba

附：冒泡排序算法及其验证数据加载

1、冒泡排序算法基本思路

冒泡排序算法要点是：对所有相邻记录的关键字值进行比效，如果是逆序（a[j]>a[j+1]），则将其交换，最终达到有序化。其基本处理思想为：（1）将整个待排序的记录序列划分成有序区和无序区，初始状态有序区为空，无序区包括所有待排序的记录。（2）对无序区从前向后依次将相邻记录的关键字进行比较，若逆序将其交换，从而使得关键字值小的记录向上“冒”（左移），关键字值大的记录向下“落”（右移）。 每经过一趟冒泡排序，都使无序区

（左边区域）中关键字值最大的记录进入有序区（右边区域），对于由 n 个记录组成的记录序列，最多经过 n-1 趟冒泡排序，就可以将这 n 个记录按关键字从小到大的顺序排列。

2、冒泡排序算法代码

完整的冒泡排序算法代码如下：

```
for (i=n; i>1; i--）{
    for (j=1; j<=i-1; j++) {
        if（a[j].key>a.[j+1].key）{
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
        }
    }
}
```

3、冒泡排序算法流程图

假设数组 a 中存放的是关键字序列，对数组 a 的元素按照从小到大的顺序排序，其完整的冒泡排序算法流程图如图 11 所示。

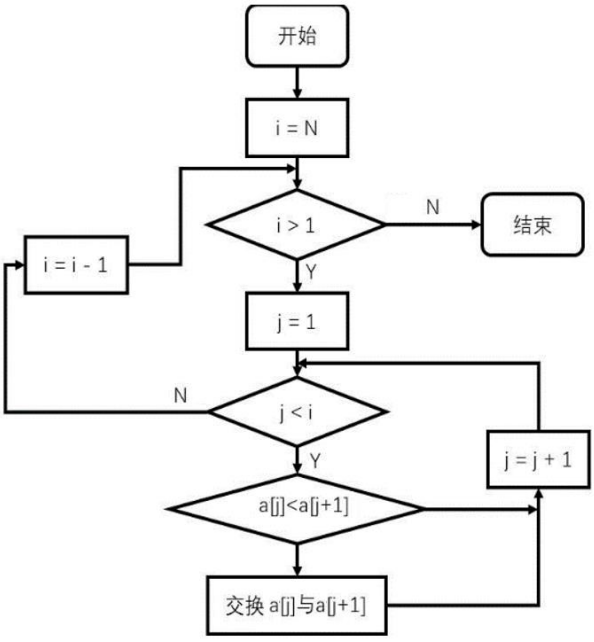

图 11 冒泡排序算法流程图

4、冒泡排序算法汇编代码设计

假设所有参数存放在从 0x0000 开始的一块存储区。首先存储的是待排序数据个数 n，然后存储 n 个待排序数据元素。

将数据个数 n 读到寄存器 x1，外循环变量 i 分配在 x2，内循环变量 j 分配在 x3，常量 1 和 4 分别存放在 x4 和 x5，常量-1 存放在 x6，第 j 个元素 a[j]的地址存放 x7，第 j+1 个元

素 a[j+1]的地址存放在 x8。第 j 个元素 a[j]读入 x9，第 j+1 个元素 a[j+1]读入 x10。

汇编语言源程序代码如下。

```
    lw    x1,0(x0)          #待排序的数字个数 n 存在 0x0000 处，
    add   x2,x1,x0          #i=n
    ori   x4,x0,1           #x4=1
    ori   x5,x0,4           #x5=4
    ori   x6,x0,0xffffffff  #x6=-1
L1:
    beq   x2,x4,finish      #if i=1  则结束
    ori   x3,x0,1           #j=1
    ori   x7,x0,4
    ori   x8,x0,8
L2:
    sltu   x11, x3,x2       # if j<i then  读取两个元素比较
    beq   x11,x0,L3
    lw x9,0(x7)             #读取第 j 个元素
    lw x10,0(x8)            #读取第 j+1 个元素
    sltu x11,x9,x10
    beq x11,x4,L4
    sw x10,0(x7)            #交换存储
    sw x9,0(x8)             #交换存储
    jal x0, L4
L3:
    add x2,x2,x6
    jal x0,   L1
L4:
    add x3,x3,x4              # j=j+1
    add x7,x7,x5
    add x8,x8,x5
    jal x0,   L2
finish:
    jal x0, finish
```

5、冒泡排序程序数据和机器代码的预置

（1）将测试用的排序程序数据文件加载到数据存储器（RAM）中，数据文件内容如下：

```
v2.0 raw
a 8 4 2 12 3 6 b 5 9 7
```

（2）导出排序程序仿真执行后的机器代码并加载到指令存储器（ROM）中。

RV32I 数据通路设计提示：**在寄存器堆设计中，要确保 0 号寄存器的数值始终为 0。**

四、思考题

1、 如果增加 R 型 and 指令和 I 型 srli 指令，则需要对单周期处理器进行哪些修改？
2、 在计算累加和程序中，参数设置为何值时，运算结果可能不对？为什么？
3、 你能否编写出仅用给出的 9 条指令实现的其它排序算法程序，并在设计的 CPU 中进行验证。

五、实验报告

1、 根据本次实验要求，写出实验操作步骤，包括：仿真检测图、错误现象及原因分析、思考题等内容。以 word 或 PDF 格式提交
2、 将实验报告和电路图.circ 文件打包上传到教学支撑平台的网站中。