



**《计算机组成原理》**

**课程实验报告**

姓名：陈澄

学院：信息学院

系：软件工程

专业：软件工程

学号：32420212202930

2023年4月20日

**第4次实验 指令系统**

1. **实验目的**
2. **掌握MIPS汇编仿真器（MARS 4.5）、RISC-V汇编仿真器（RARS 1.5）、Intel x86汇编工具（masm32）、ARMv7在线汇编工具的使用方法。**
3. **在Logisim实现的单周期MIPS处理器、Logisim实现的单周期RISC-V处理器上运行程序。**
4. **分别使用MIPS指令、RISC-V指令、x86指令、ARMv7指令编写排序程序。**
5. **实验环境**

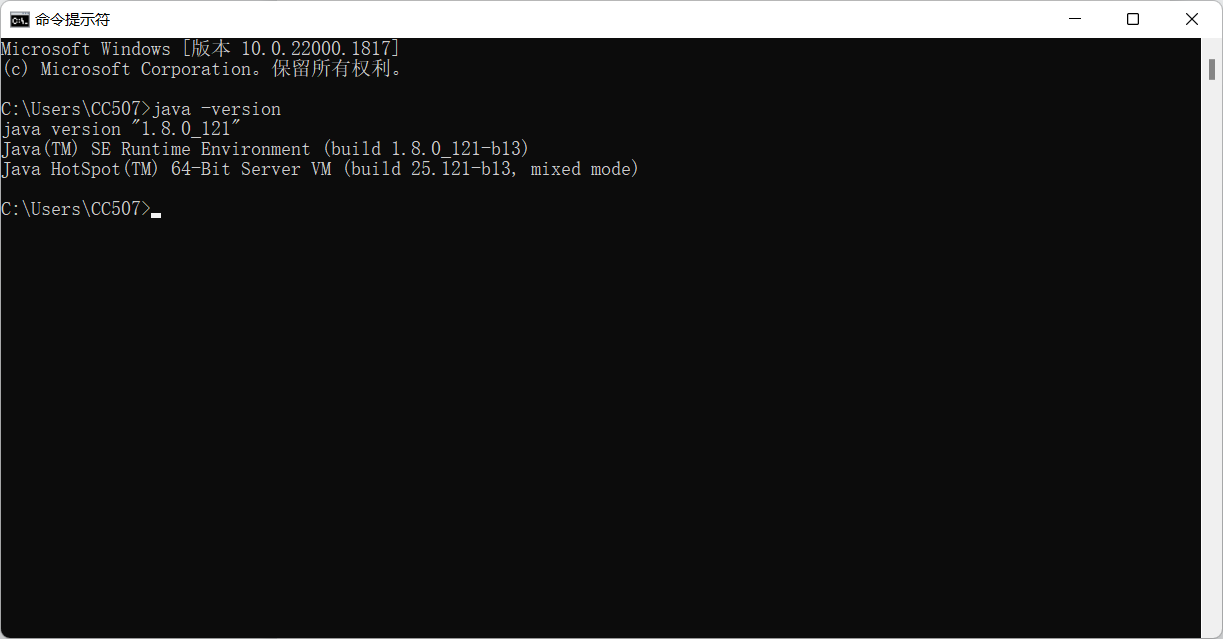
**（1）MARS 4.5、RARS 1.5、masm32、ARMv7在线汇编工具。**

**（2）Windows系统下运行Logisim软件（需安装JDK）。**

1. **实验内容**
   1. **课堂完成部分（验证实验的内容）**
2. MIPS汇编语言程序的运行

（1）运行MIPS汇编仿真器

查看电脑上是否已有JDK？



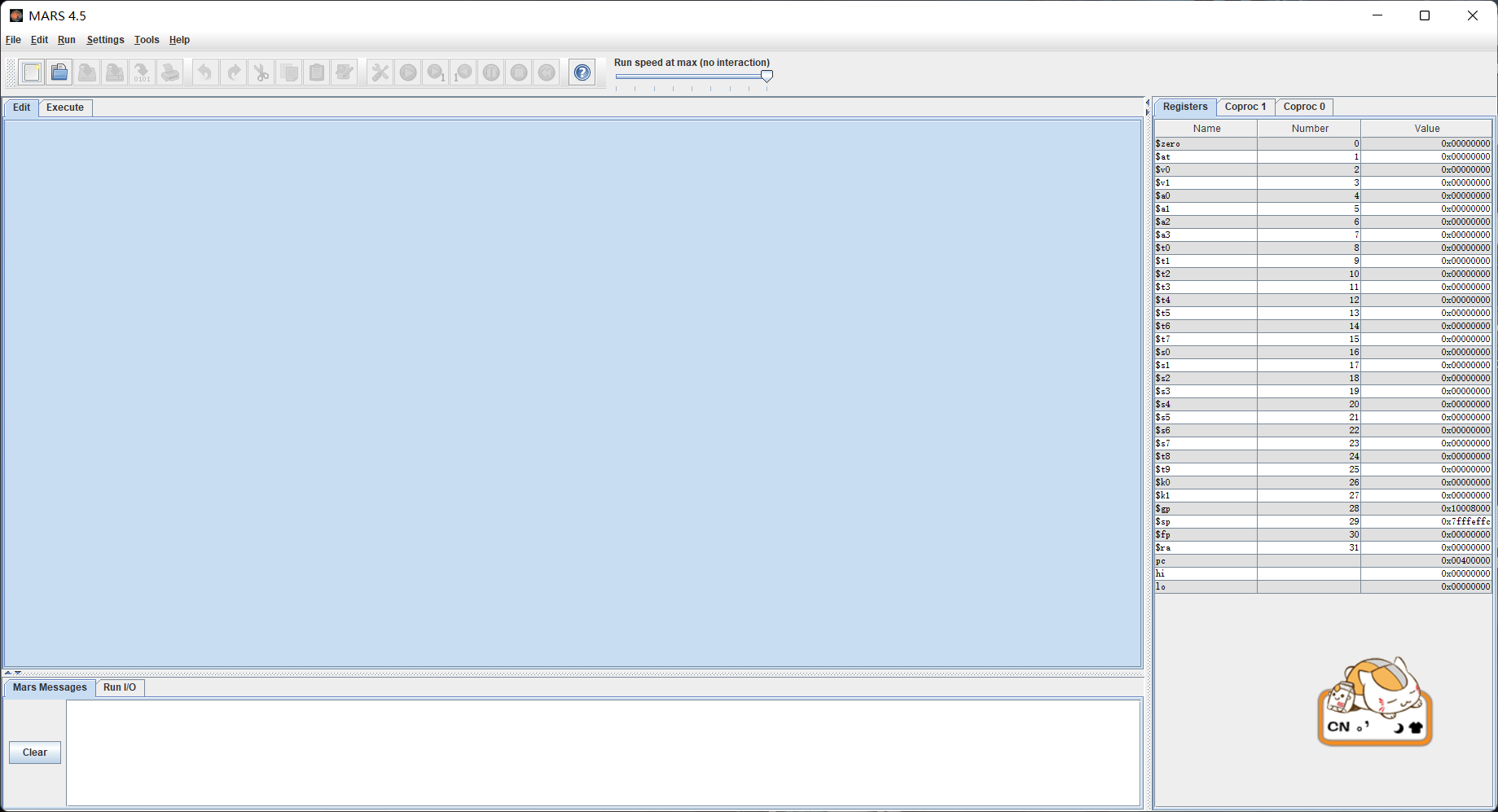
运行MIPS汇编仿真器Mars4\_5.jar

打开“命令提示符”，执行：

• cd D:\

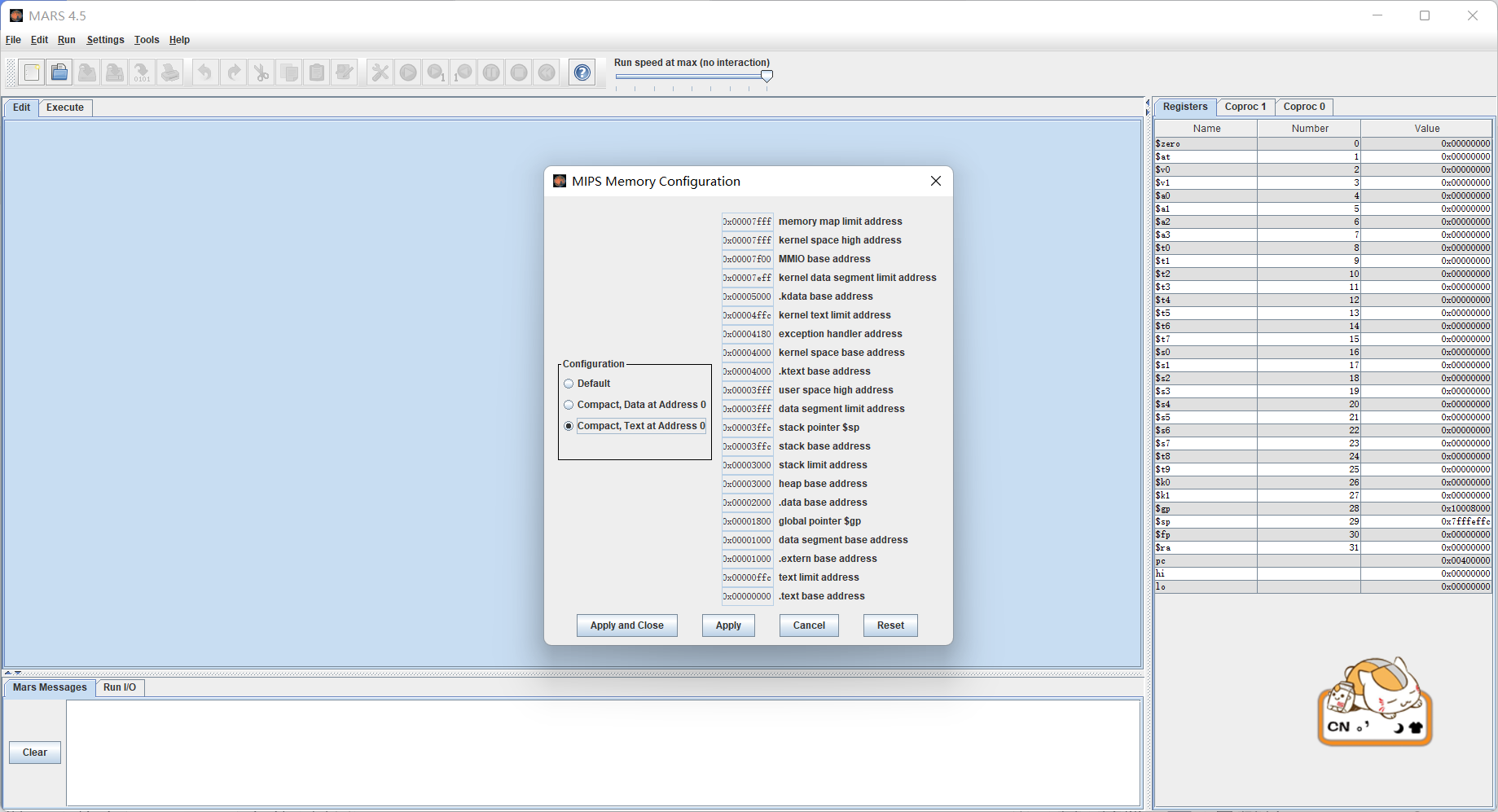
• cd \mips

• java -jar Mars4\_5.jar

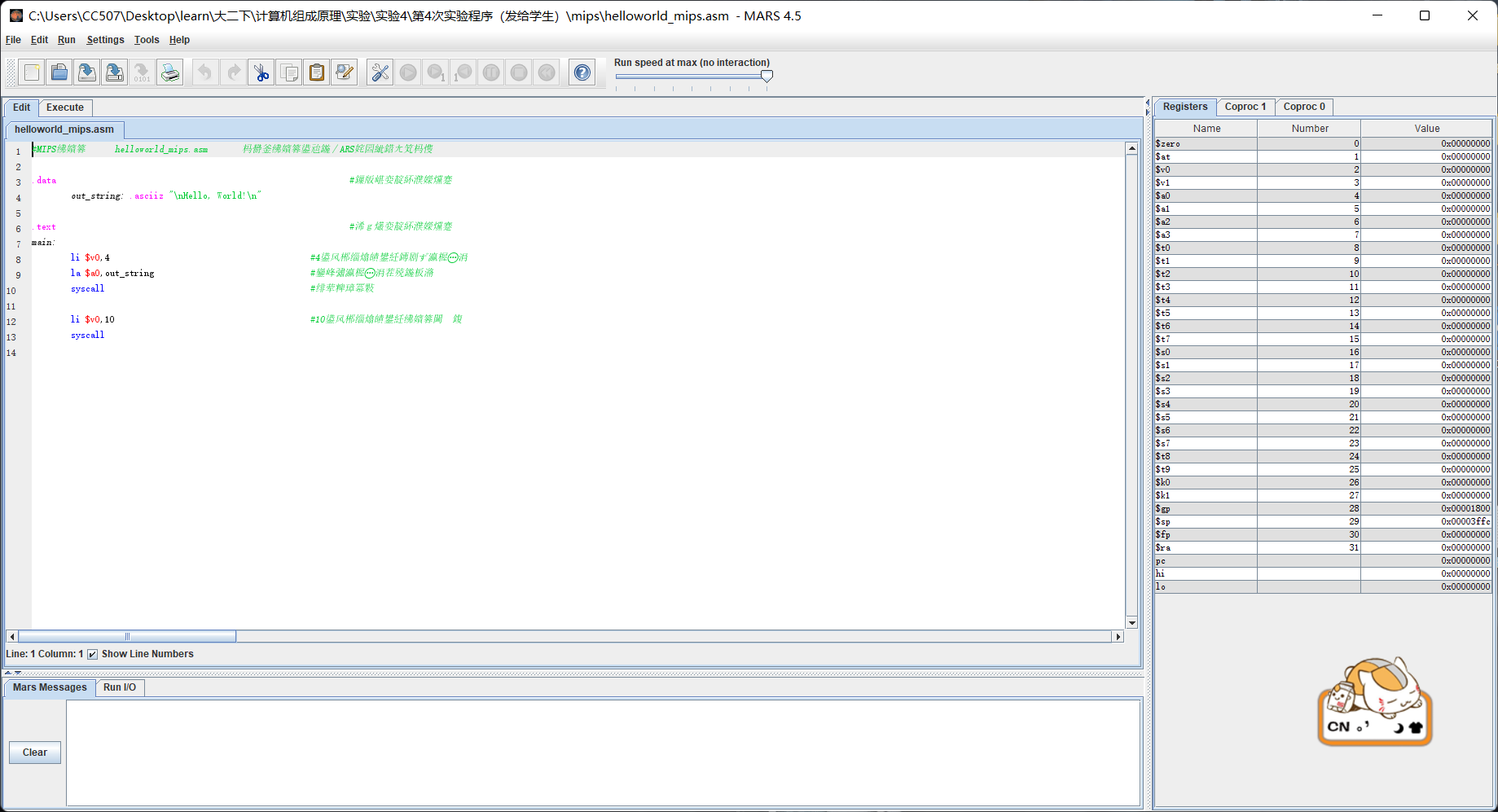


1. 运行第一个MIPS汇编语言程序

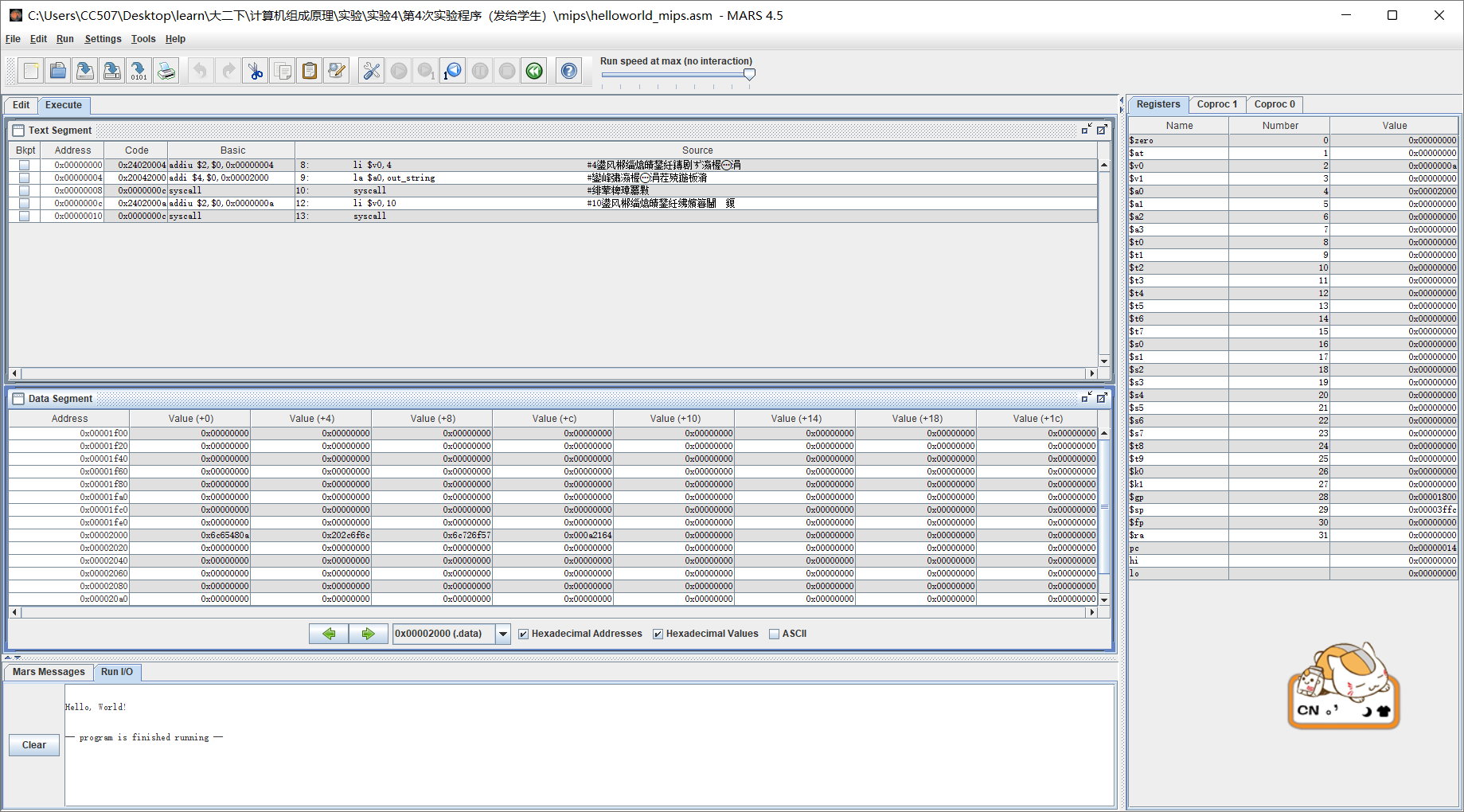
将MARS汇编仿真器的代码段的开始地址设置为0



在MARS汇编仿真器中打开helloworld\_mips.asm

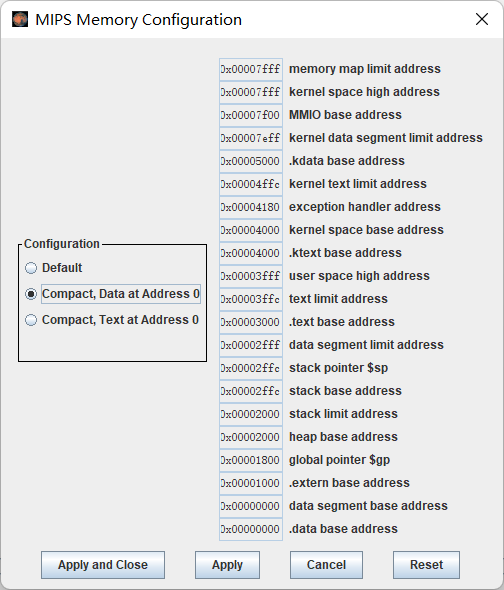


对helloworld\_mips.asm源程序进行汇编（点击：Assemble）



1. 求累加和的MIPS汇编语言程序

将MARS汇编仿真器的数据段的开始地址设置为0

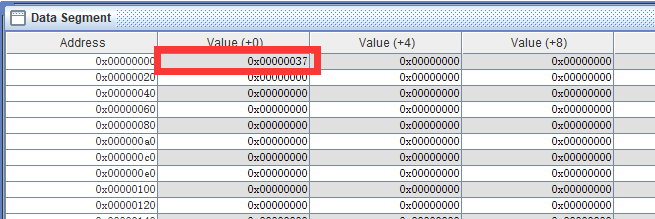


在MARS汇编仿真器中打开sum\_mips.asm

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（地址=0）：

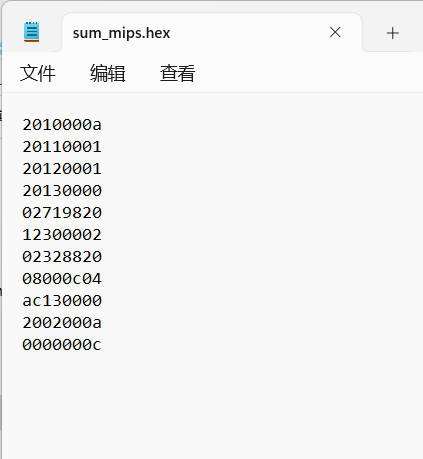


导出汇编后的机器码：

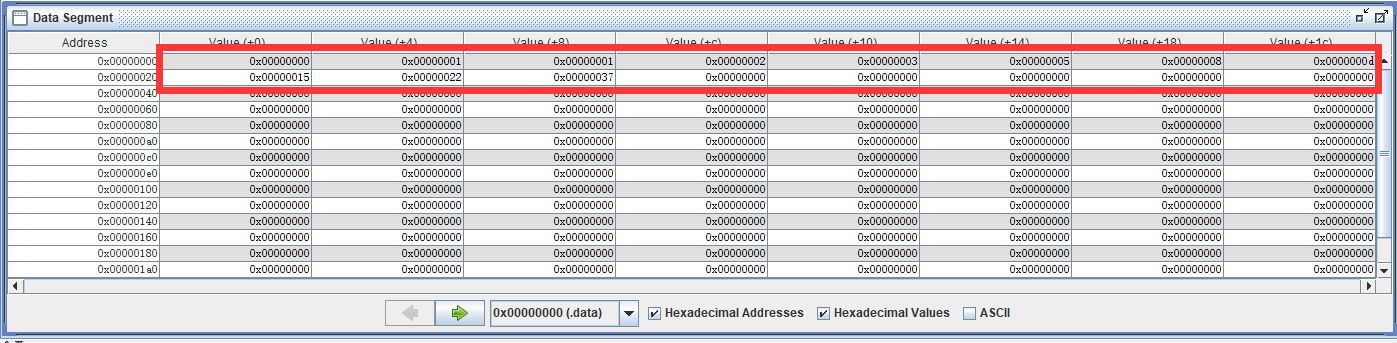
“File” -> “Dump Memory”

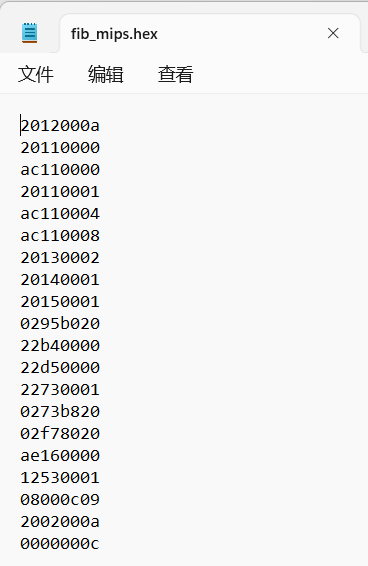
保存为十六进制文本格式

保存到sum\_mips.hex文件中



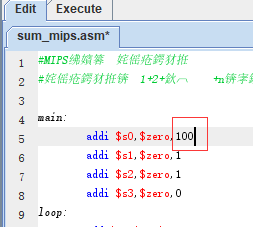
1. 计算费波那契数列的MIPS汇编语言程序

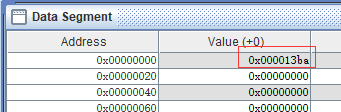




1. 要求

请同学们修改“sum\_mips.asm”程序，计算1+2+3+……+100=5050=13bah，然后在MARS 4.5汇编仿真器中运行，并将该程序的机器码保存到sum100\_mips.hex文件中。

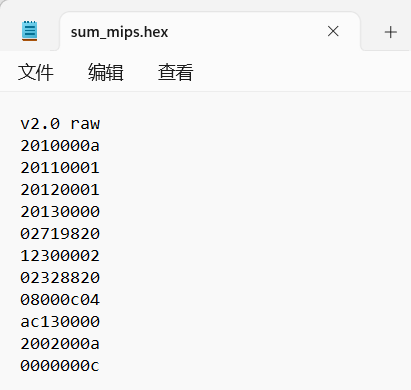






1. 在Logisim上运行MIPS程序
2. Logisim实现的单周期MIPS处理器能够运行24条指令（核心指令8条+基础指令16条）
3. Logisim实现的单周期MIPS处理器
4. 在Logisim实现的单周期MIPS处理器上运行求累加和程序

第一步：修改前面导出的机器码文件sum\_mips.hex，增加：v2.0 raw

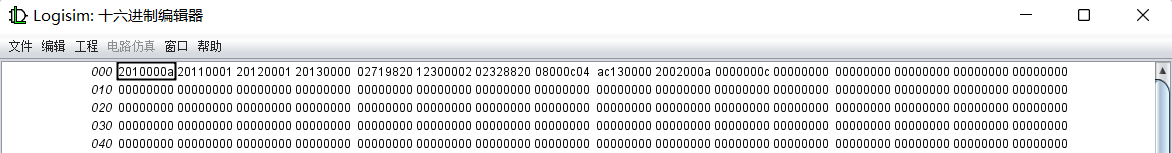


第二步：打开Logisim，在Logisim中增加寄存器文件库“cs3410.jar”（位于\mips目录中）

第三步：在Logisim中打开设计文件“单周期MIPS处理器.circ”（位于\mips目录中）



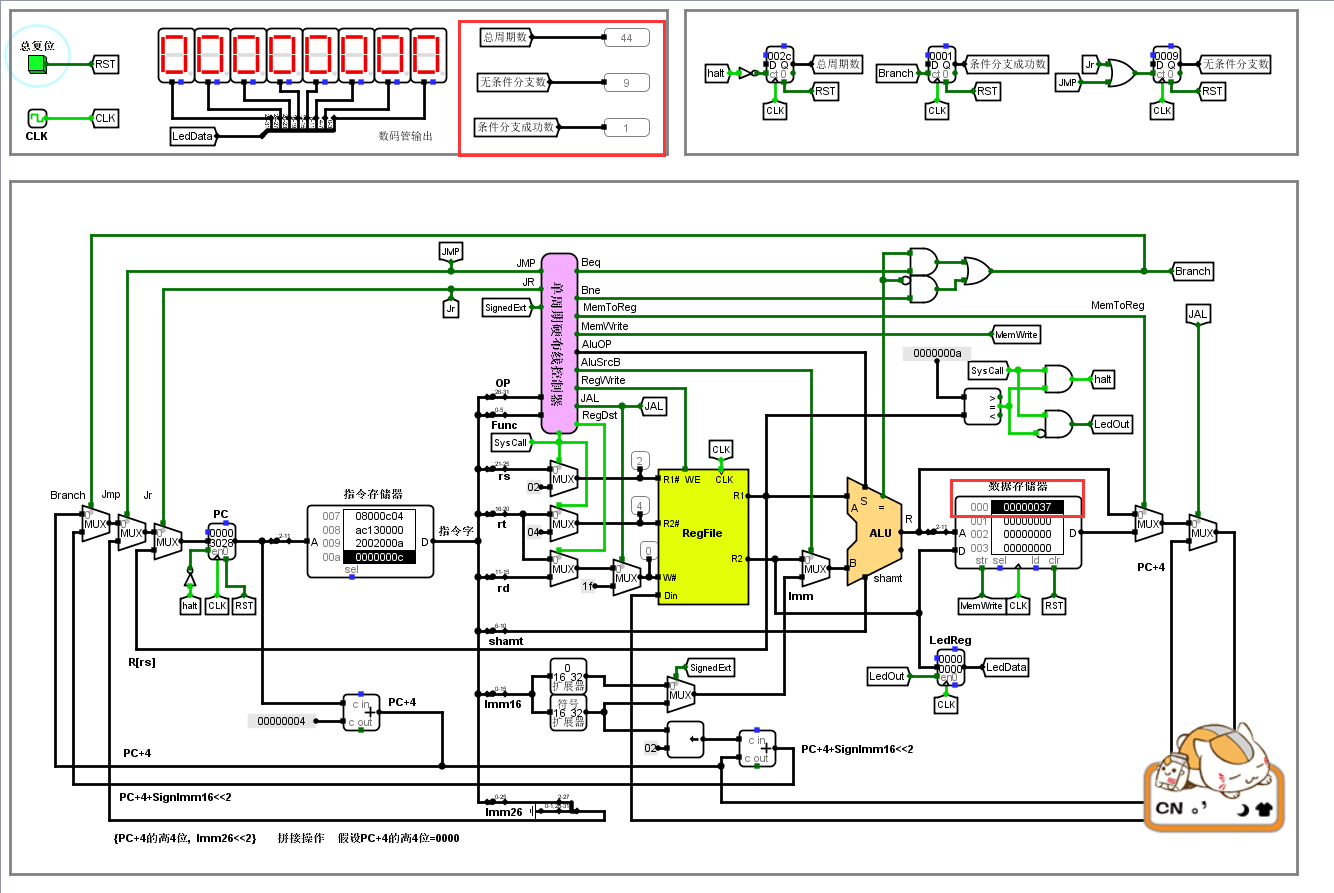
第四步：在单周期MIPS处理器数据通路的指令存储器中装入机器码文件sum\_mips.hex



第五步：设置时钟频率=1KHz，按Ctrl+K启动时钟，程序开始执行

第六步：按总复位，程序重新执行；指令存储器中的内容停止滚动后，表示程序执行结束；再按Ctrl+K停止时钟

第七步：此时观察数据存储器第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



1. 在Logisim实现的单周期MIPS处理器上运行计算费波那契数列程序

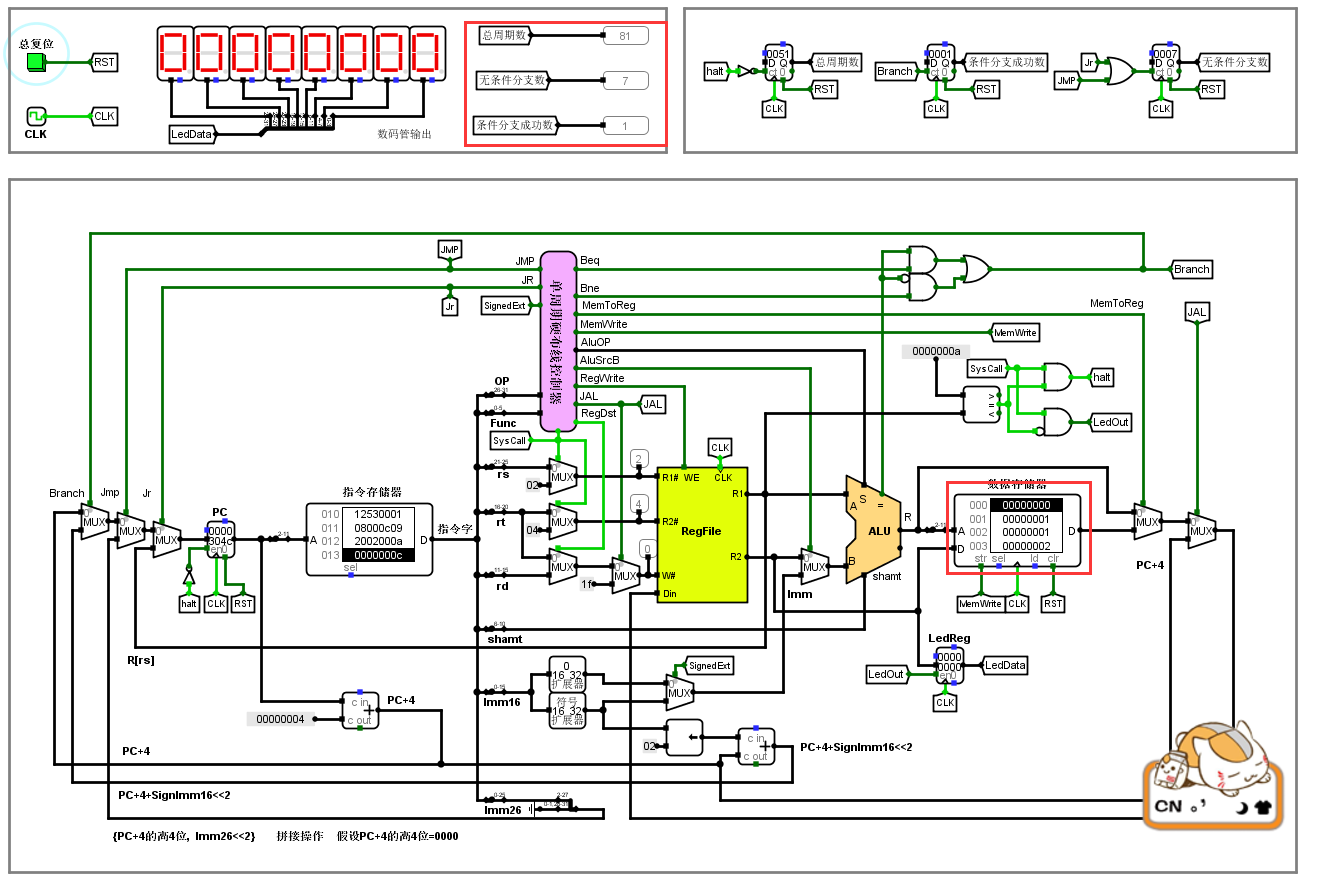
第一步：修改前面导出的机器码文件fib\_mips.hex，增加：v2.0 raw

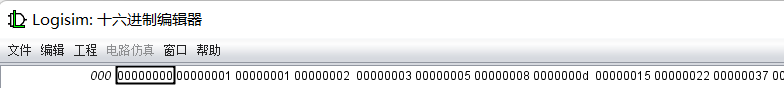
第二步：在单周期MIPS处理器数据通路的指令存储器中装入机器码文件fib\_mips.hex

第三步：设置时钟频率=1KHz，按Ctrl+K启动时钟，程序开始执行

第四步：按总复位，程序重新执行；指令存储器中的内容停止滚动后，表示程序执行结束；再按Ctrl+K停止时钟

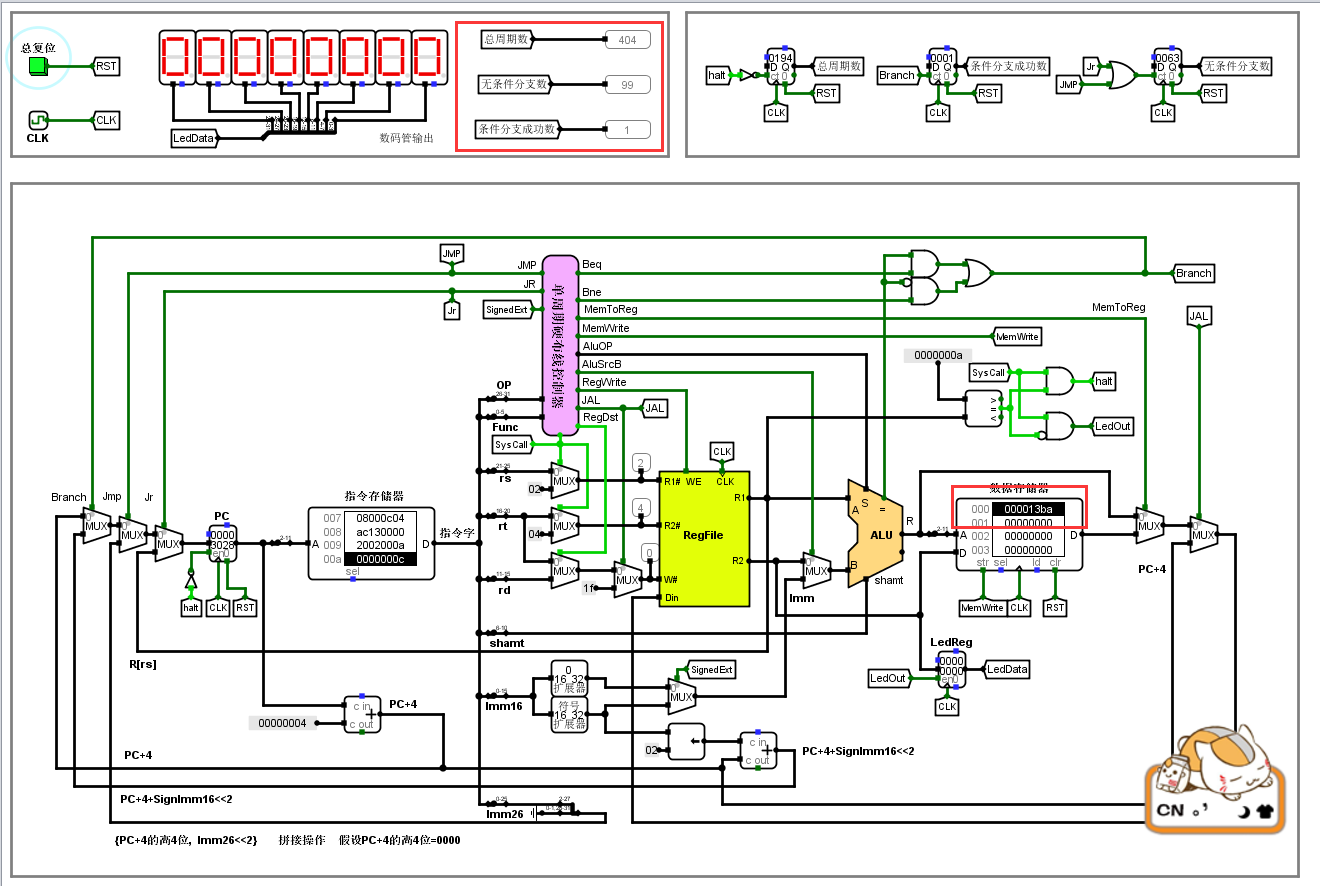
第五步：此时观察数据存储器中的内容（0号单元开始的11个单元），以及程序运行的总周期数、无条件分支数、条件分支成功数





1. 要求

请同学们在Logisim实现的单周期MIPS处理器上，运行计算1+2+3+……+100=5050=13bah的机器语言程序，观看数据存储器0号单元的值是不是13bah？程序运行的总周期数、无条件分支数、条件分支成功数分别是多少？

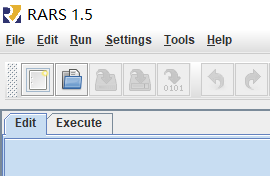


总周期数：404

无条件分支数：99

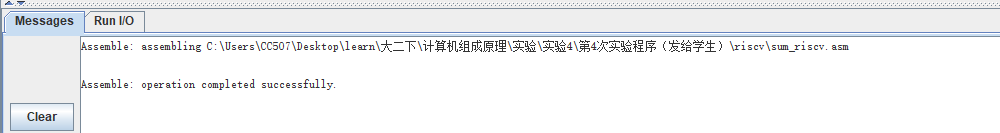
条件分支数：1

1. RISC-V汇编语言程序的运行
2. 运行RISC-V汇编仿真器

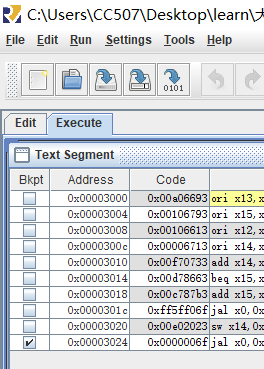


1. RISC-V求累加和程序

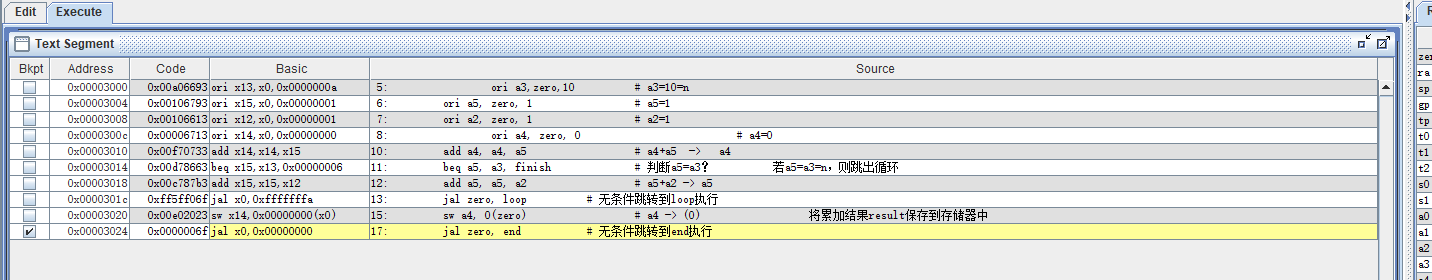
在RARS中汇编“sum\_riscv.asm”



在最后一条指令上设置断点

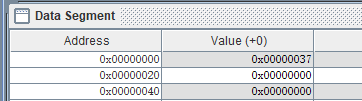


运行结束后，程序将停在断点处

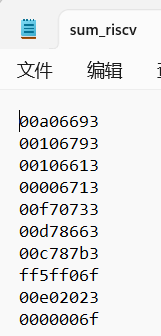


此时可以看到程序运行的结果在0号存储单元中

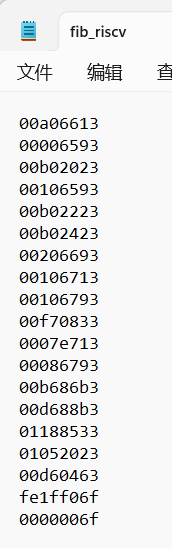
结果 = 37h = 55 = 1+2+3+…+10



将汇编后的机器码，导出到一个文本文件中：



1. RISC-V计算费波那契数列的程序

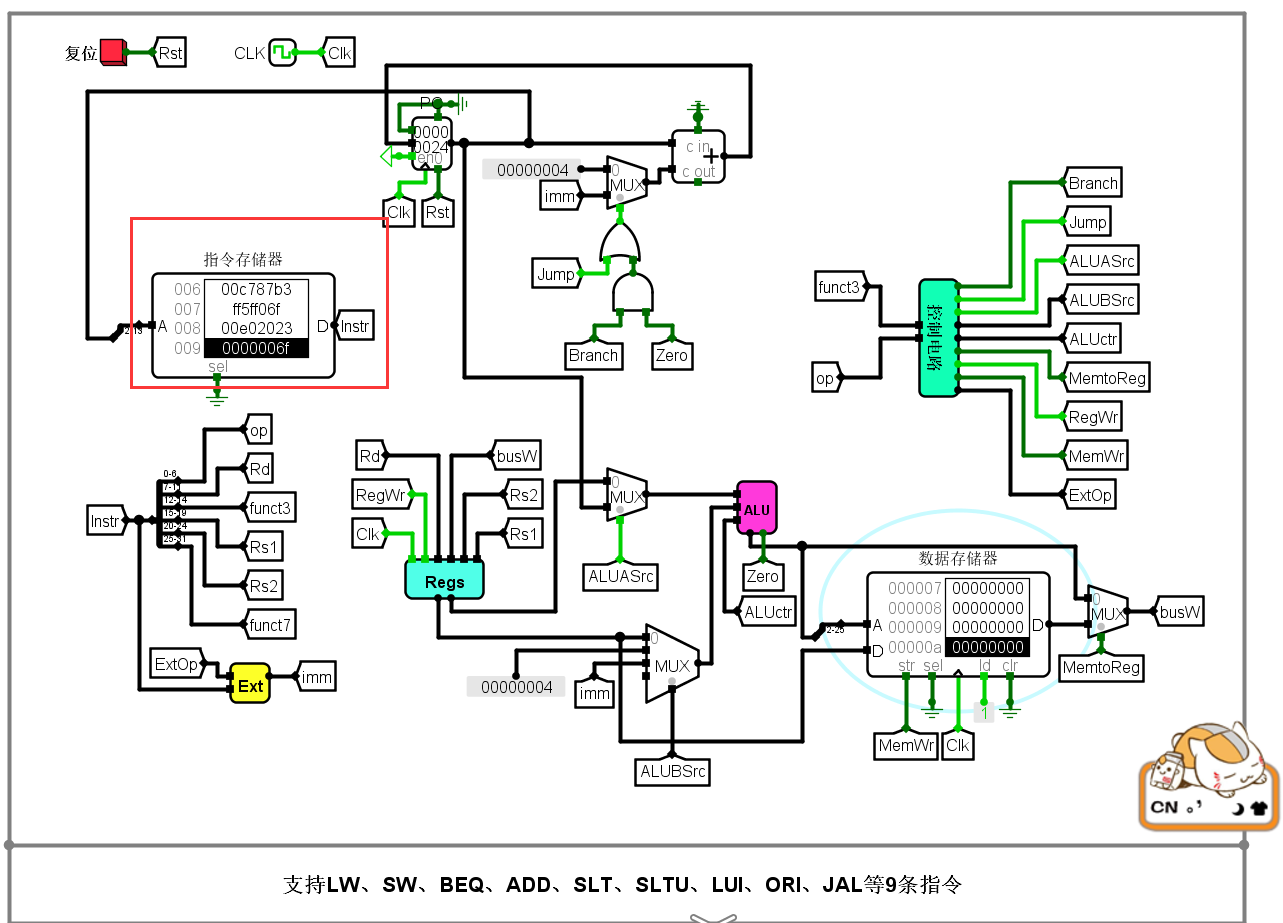


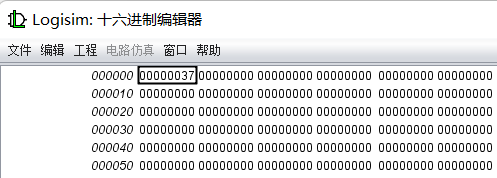
1. 要求

请同学们修改sum\_riscv.asm程序，计算1+2+3+……+100=5050=13bah，然后在RARS 1.5汇编仿真器中运行，并将该程序的机器码保存到sum100\_riscv.hex文件中。

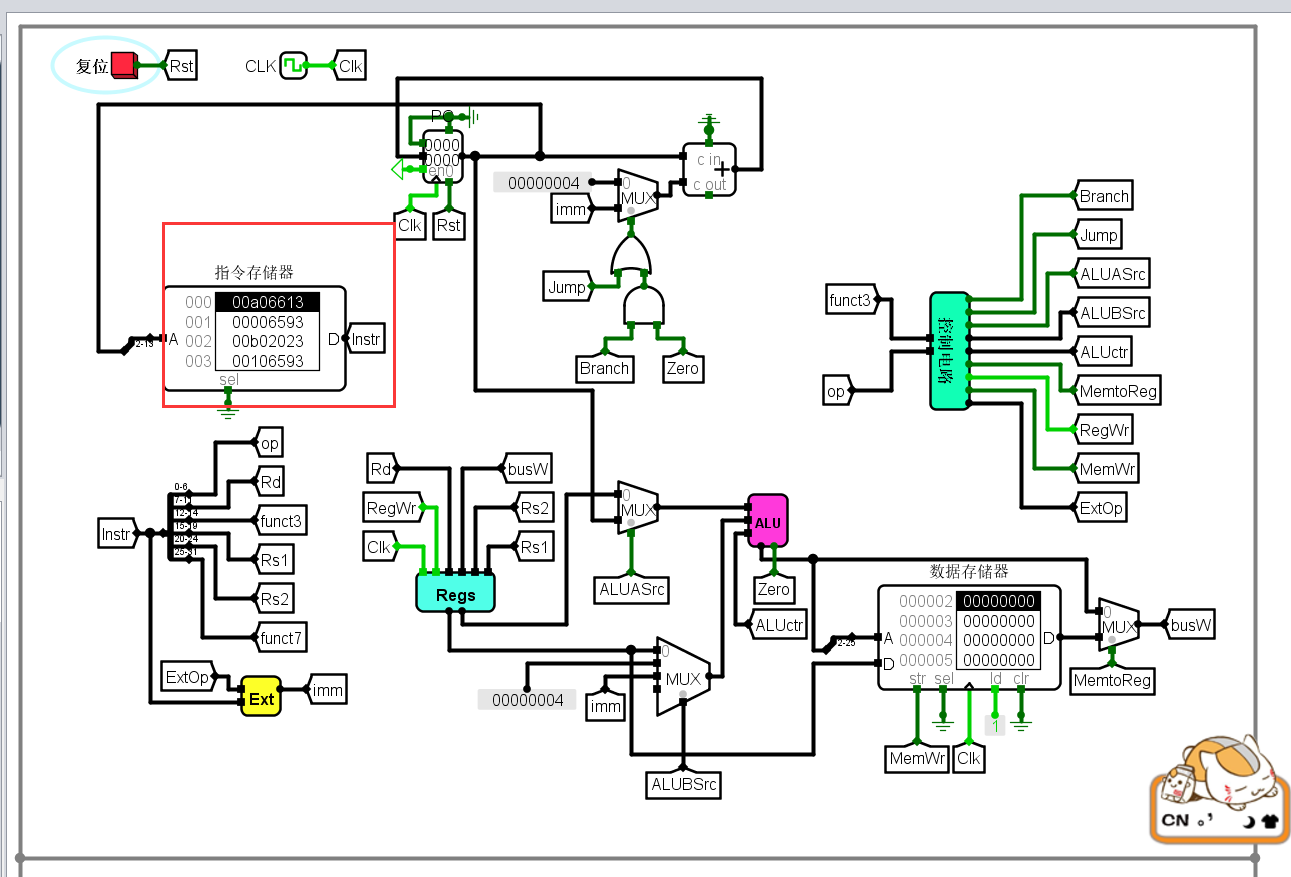


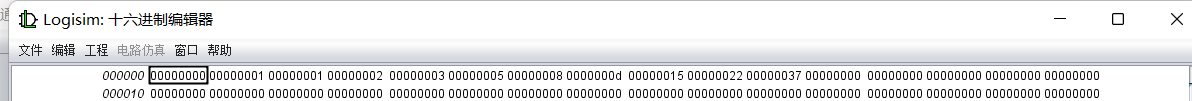
1. 在Logisim上运行RISC-V程序
2. Logisim实现的单周期RISC-V处理器能够运行以下9条指令
3. Logisim实现的单周期RISC-V处理器
4. 在Logisim实现的单周期RISC-V处理器上运行求累加和的程序





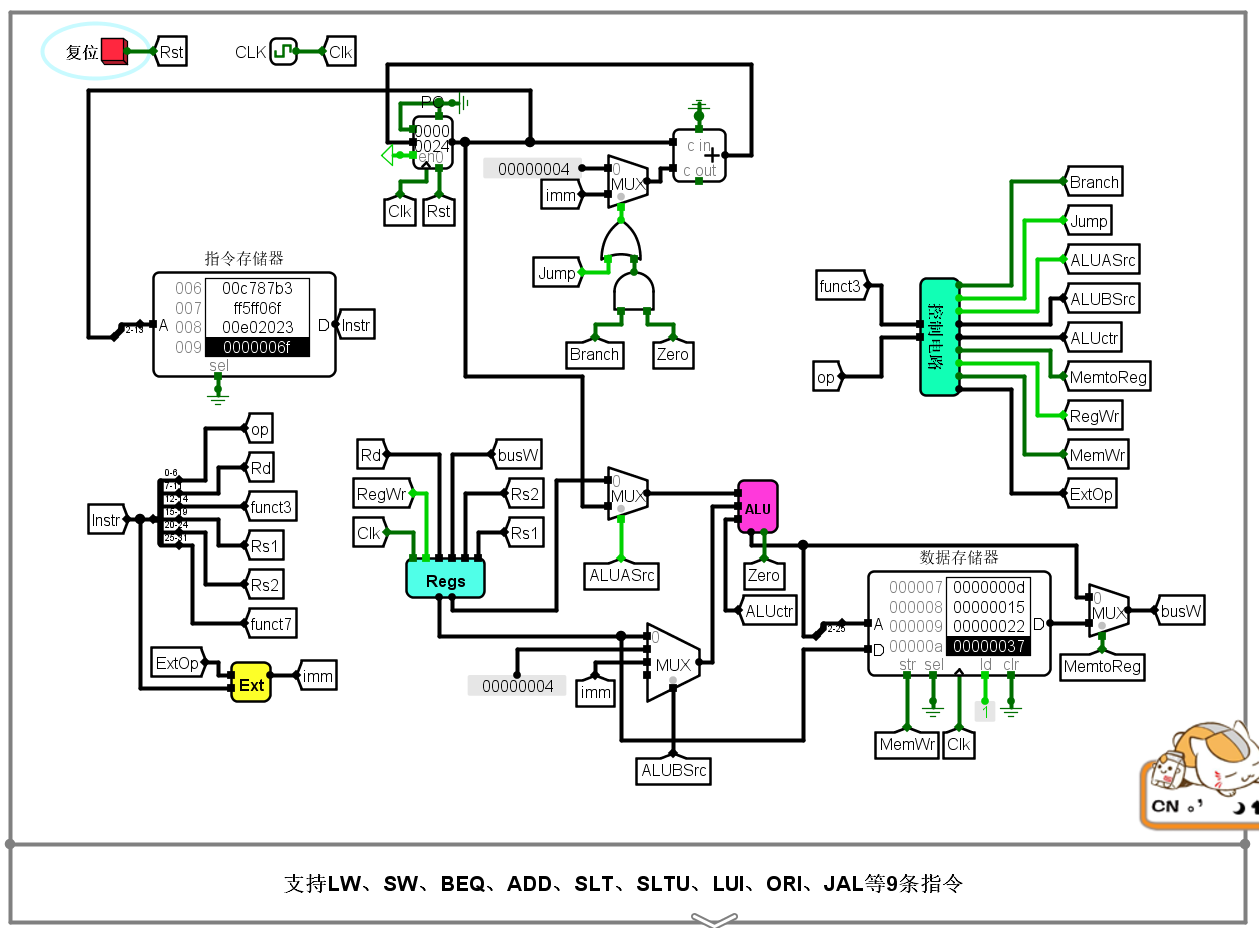
1. 在Logisim实现的单周期RISC-V处理器上运行计算费波那契数列的程序

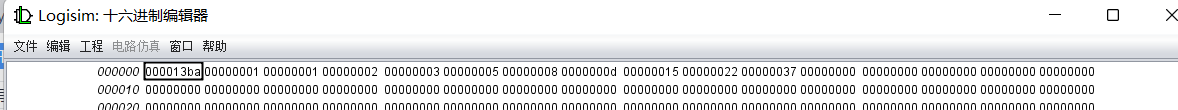




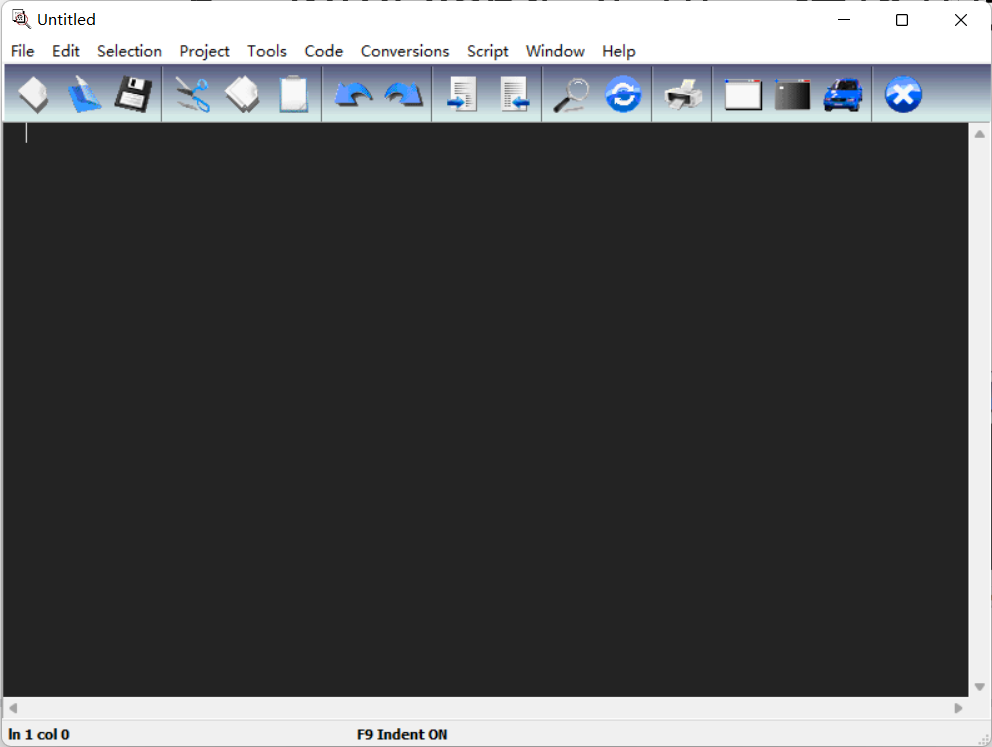
1. 要求

请同学们在Logisim实现的单周期RISC-V处理器上，运行计算1+2+3+……+100=5050=13bah的程序，观看数据存储器0号单元的值是不是13bah？是



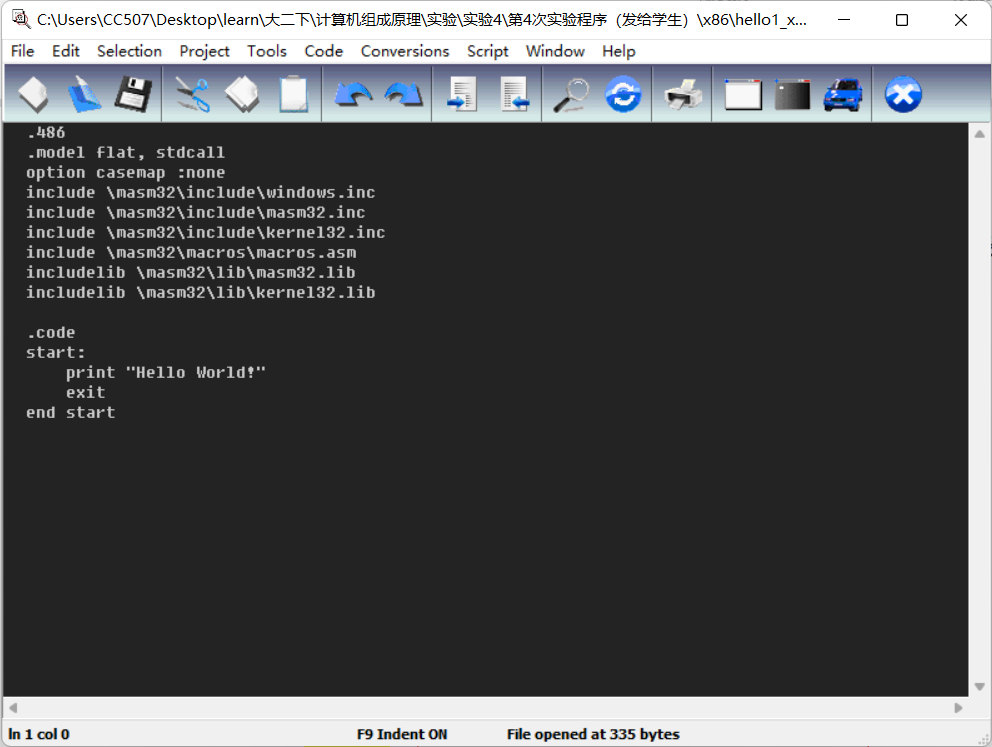


1. Intel x86汇编语言程序的运行
2. Intel x86汇编工具



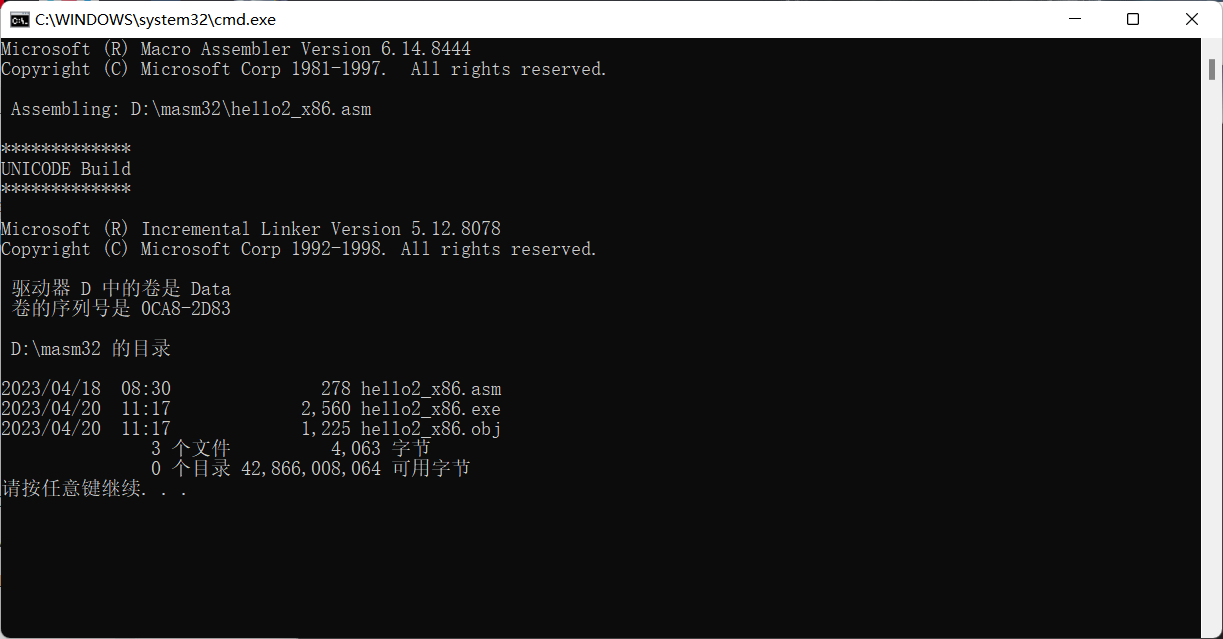
1. 运行Intel x86汇编语言程序

第一步：在masm32汇编工具中打开源程序“hello1\_x86.asm”



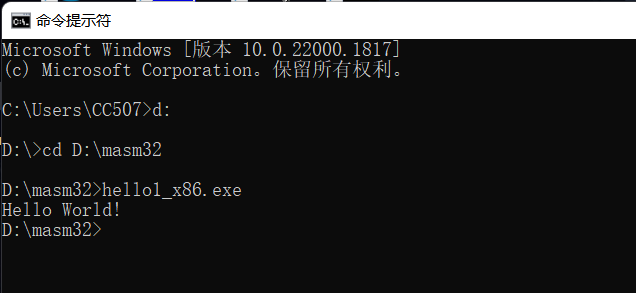
第二步：汇编

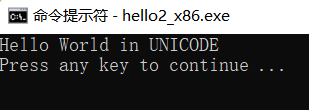
点击菜单：Project -> Console Assemble & Link

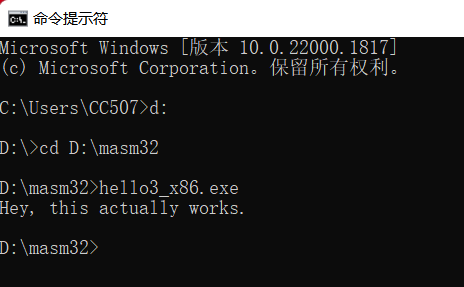


得到：hello1\_x86.obj、hello1\_x86.exe

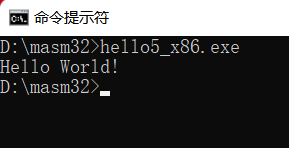
第三步：运行“hello1\_x86.exe”





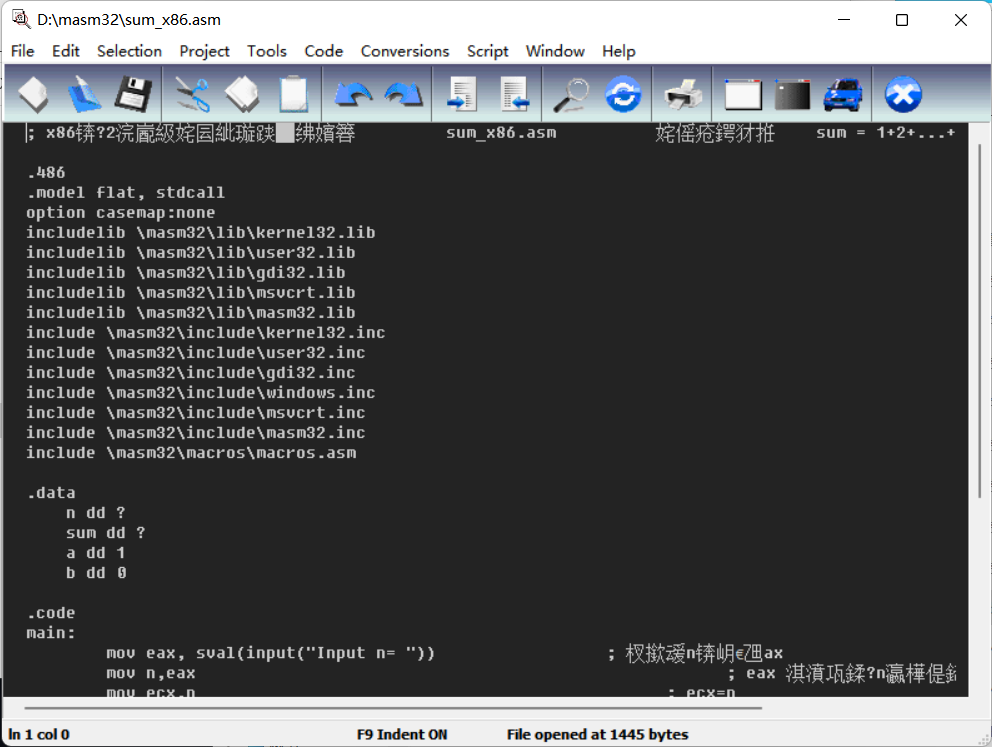




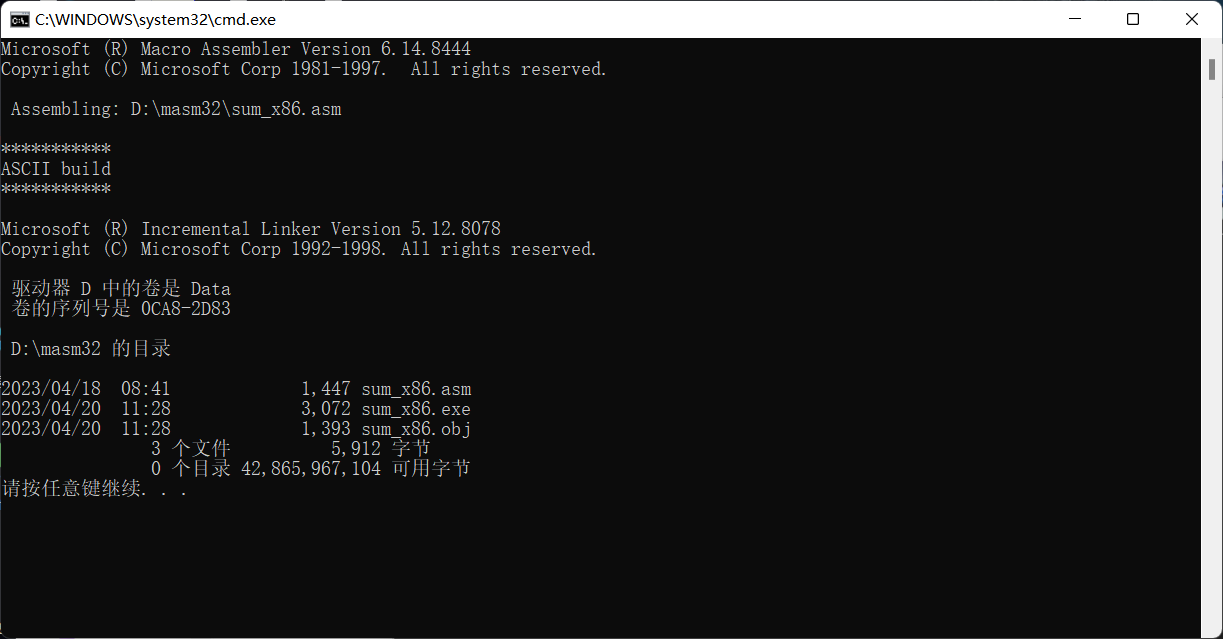


1. 运行求累加和的Intel x86汇编语言程序

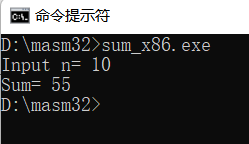
第一步：在masm32汇编工具中打开源程序“sum\_x86.asm”



第二步：汇编

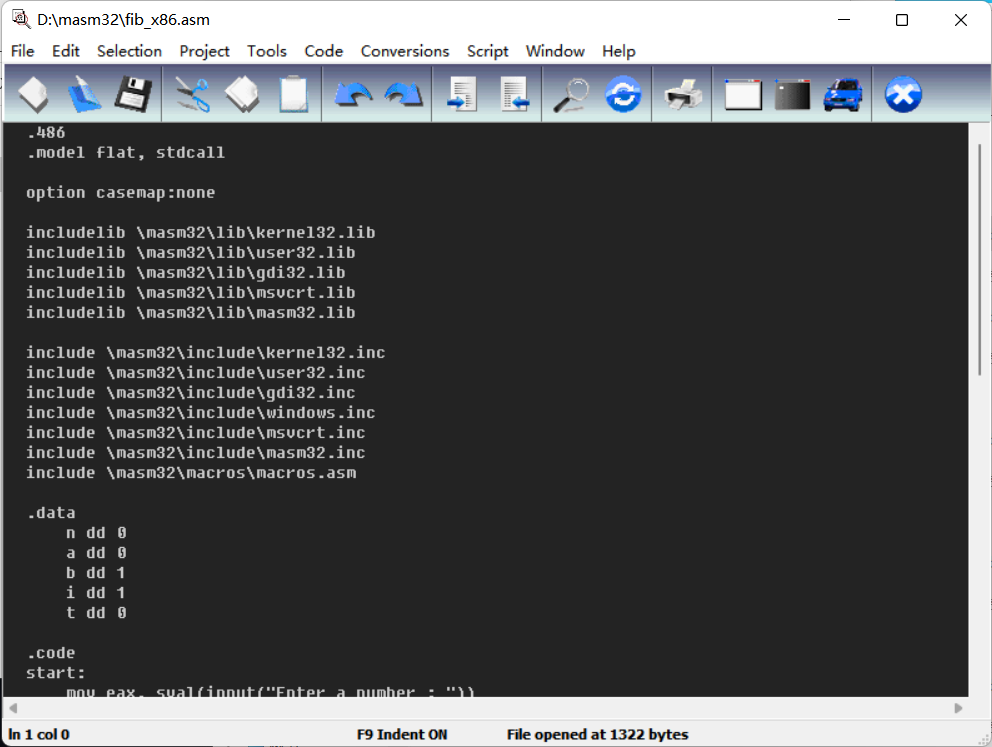


第三步：运行“sum\_x86.exe”

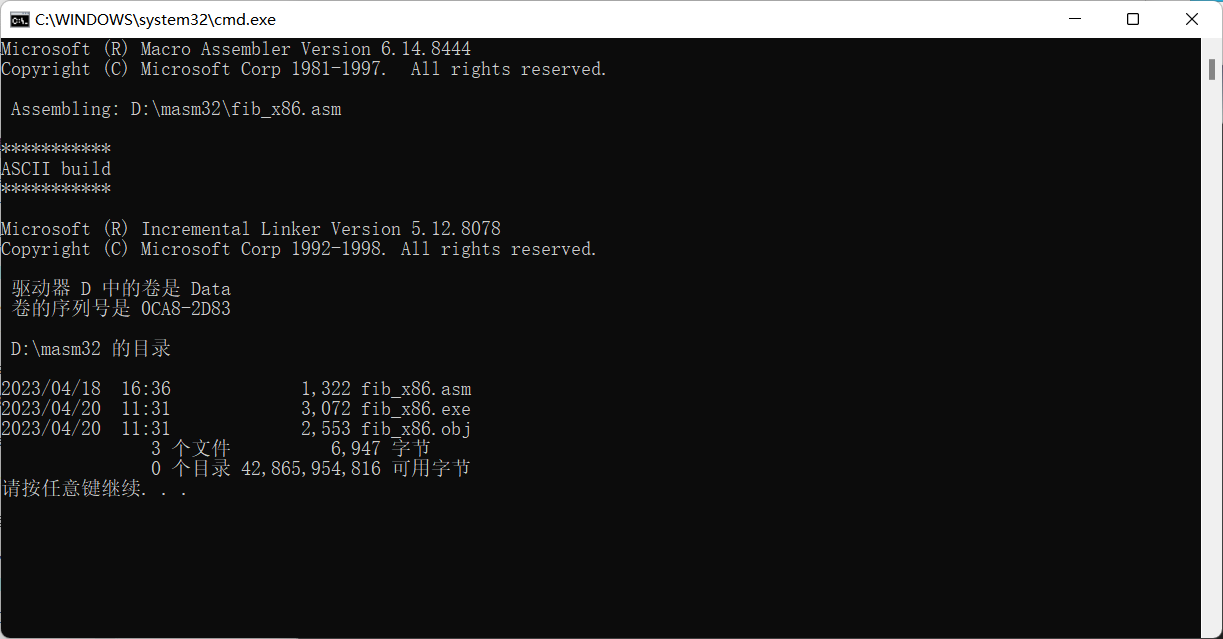


1. 运行计算费波那契数列的Intel x86汇编语言程序

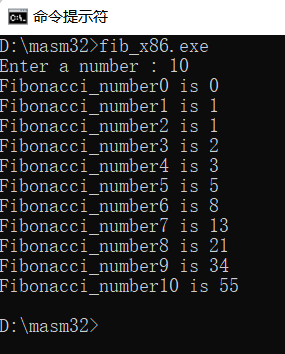
第一步：在masm32汇编工具中打开源程序“fib\_x86.asm”



第二步：汇编

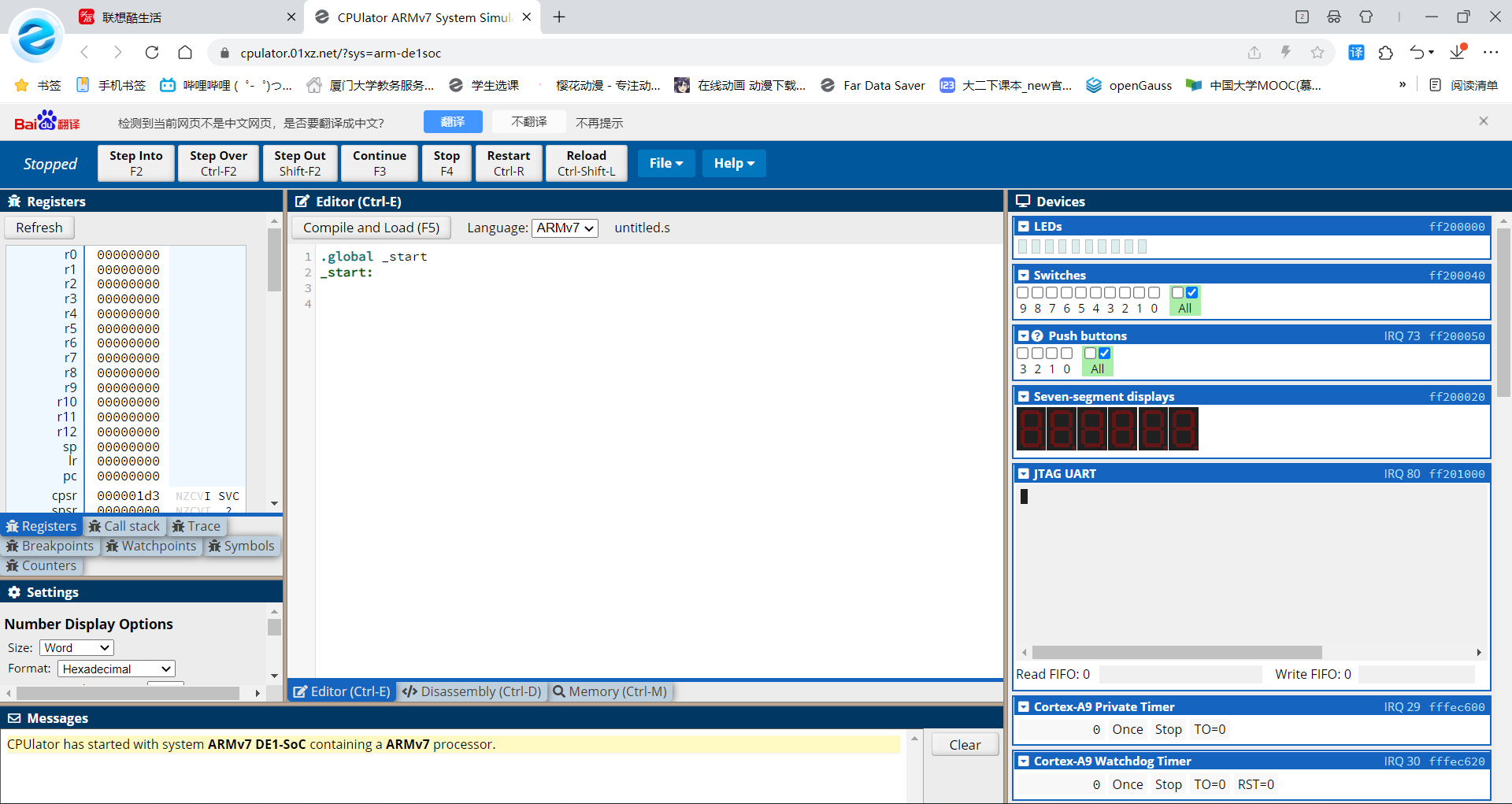


第三步：运行“fib\_x86.exe”



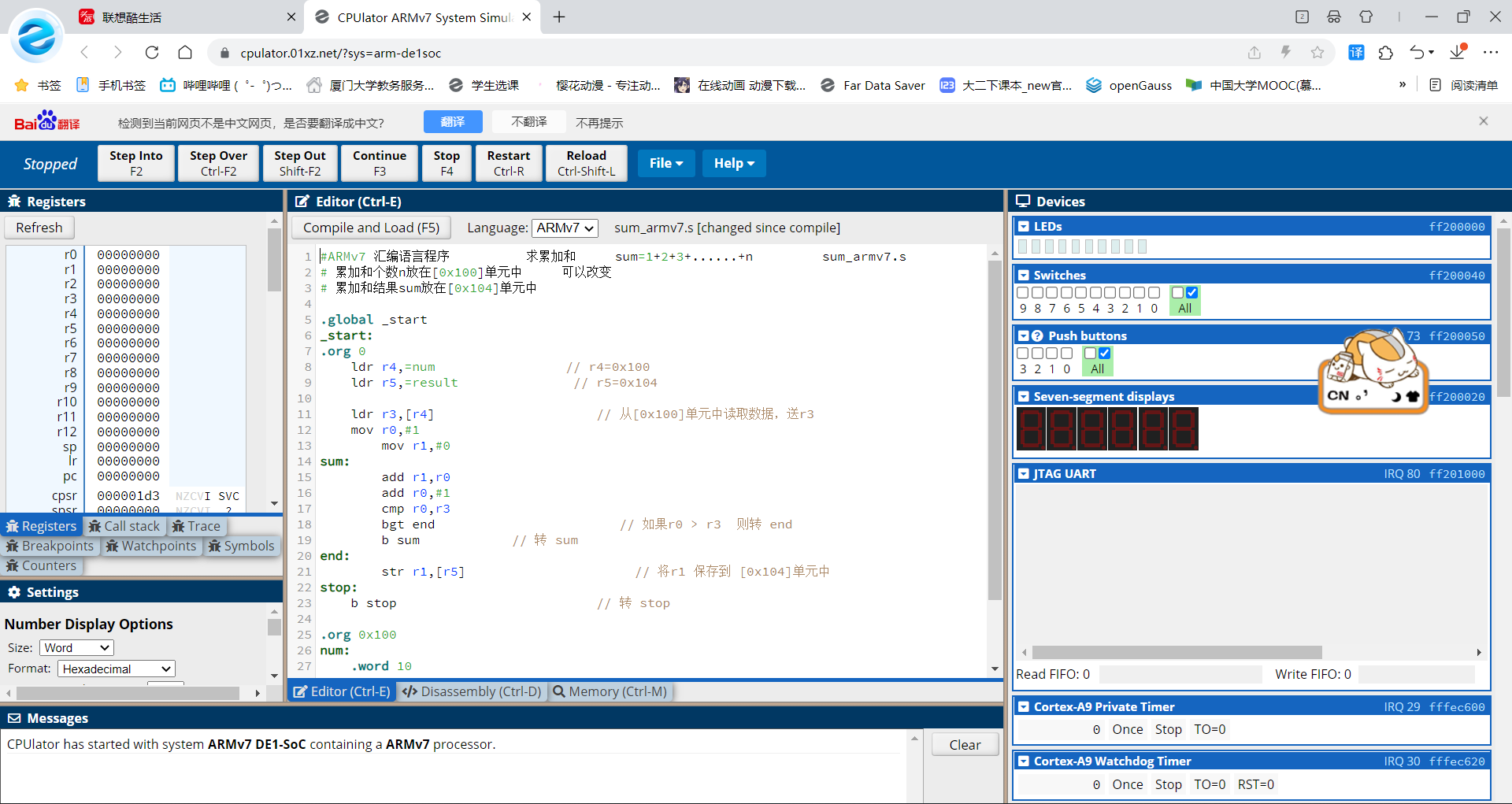
6. ARMv7汇编语言程序的运行

（1）ARMv7汇编工具

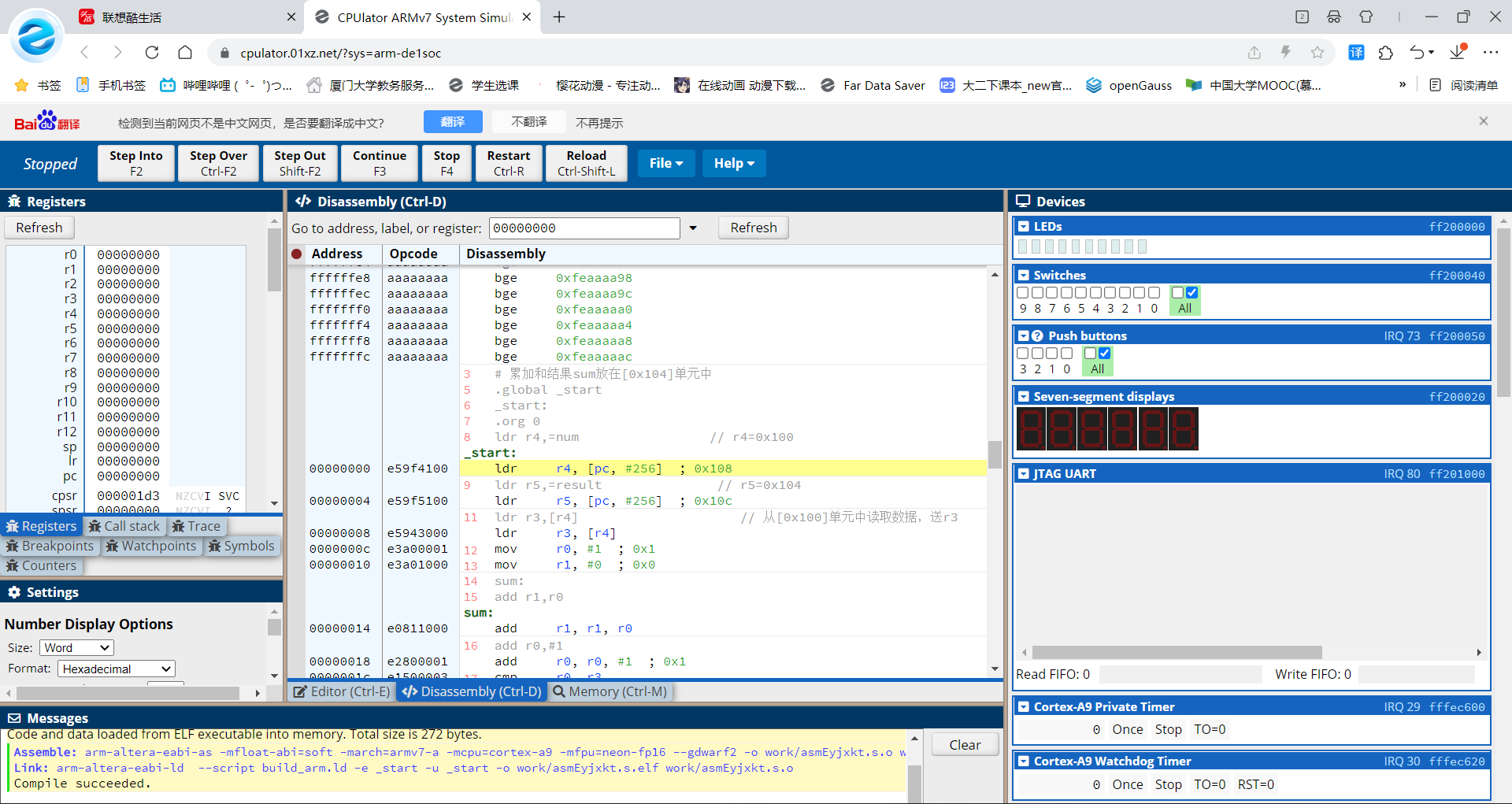


1. 在ARMv7汇编工具中运行求累加和程序

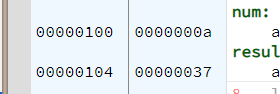
第一步：打开源程序（sum\_armv7.s）



第二步：汇编

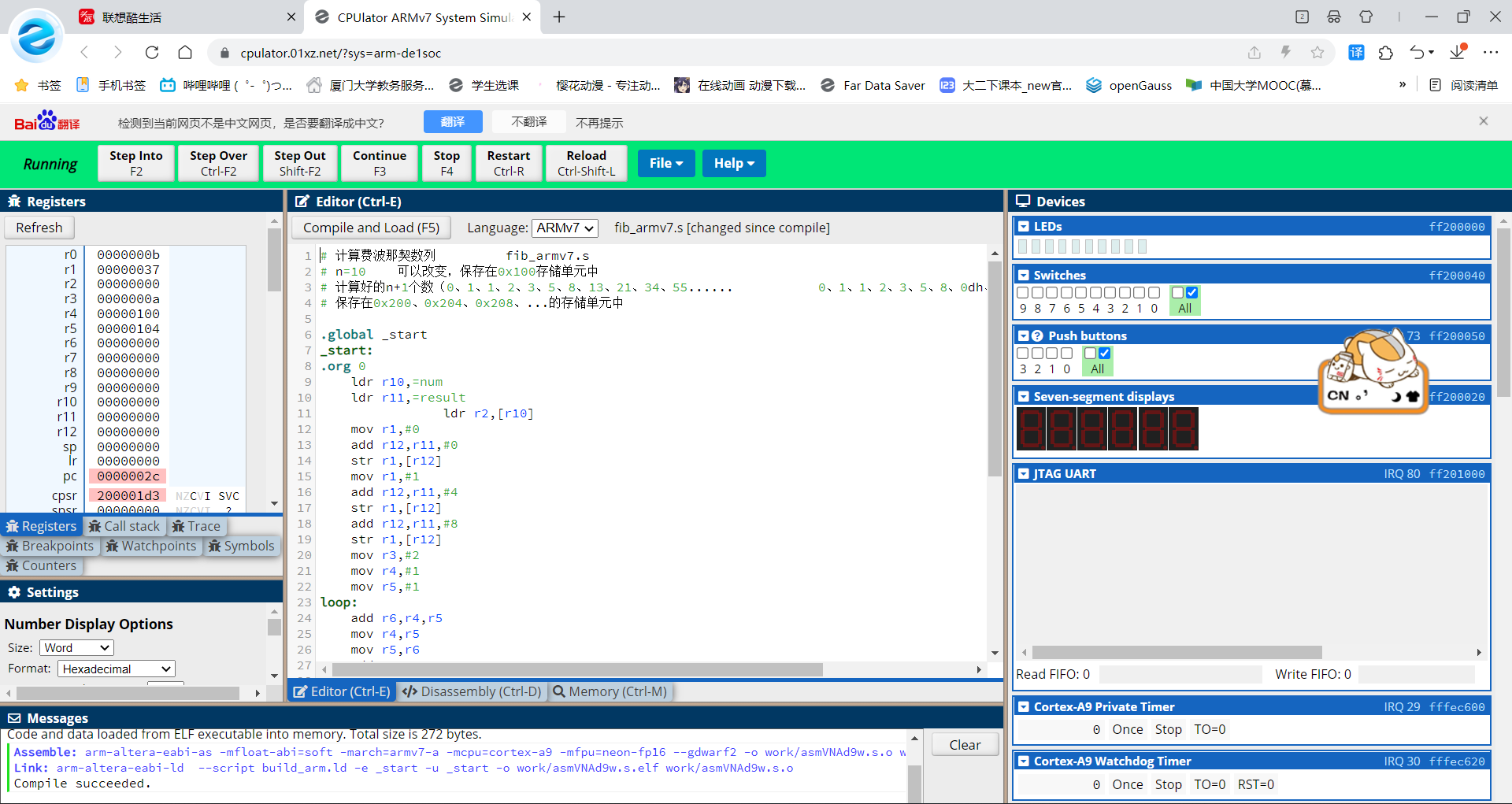


第三步：连续执行程序

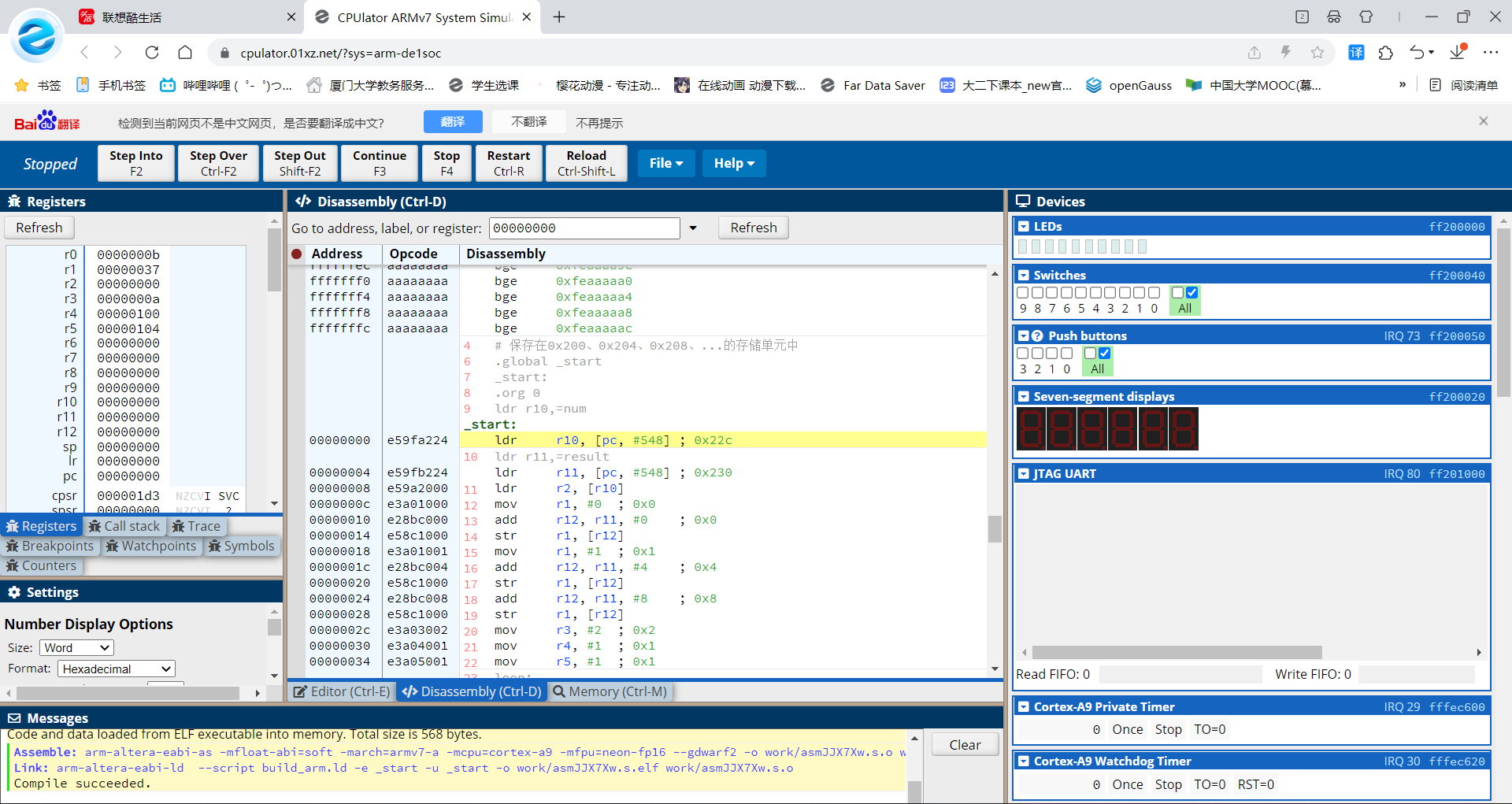


1. 在ARMv7汇编工具中运行计算费波那契数列程序

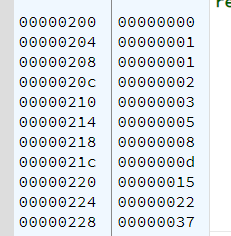
第一步：打开源程序（fib\_armv7.s）



第二步：汇编



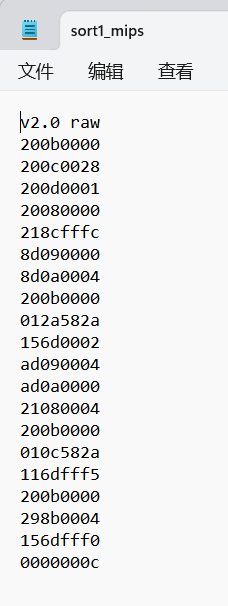
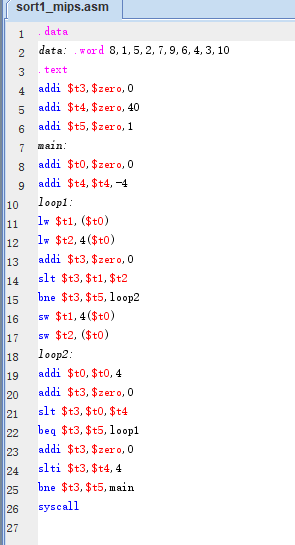
第三步：连续执行程序



* 1. **课后完成部分（设计实验的内容）**

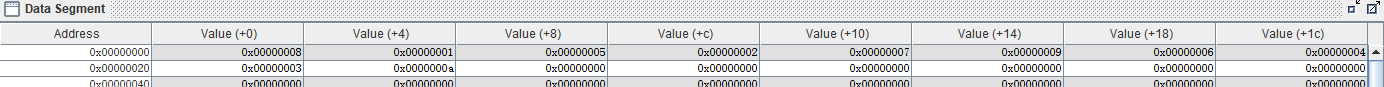
1. **MIPS汇编语言程序设计（排序程序）（设计实验）——文件名：sort1\_mips.asm（降序排序）、sort2\_mips.asm（升序排序）。**

**实现思路：采用冒泡排序，$t0用于储存当前数据地址，通过lw指令每次取出($t0)存入$t1，4($t0)存入$t2，其后利用slt指令将两者进行比较，结果存入$t3，$t5始终为常数1与$t3比较，根据结果决定是否用sw指令交换两数据，$t4作为计数器，用于判断循环进行次数，初始化为40，每进行一次循环-4（$t0每进行第一次一次循环会累加到36，进行第二次循环只需比较前9个数，地址只需累加到32，因此用$t4作为一次循环是否结束的判断）。**

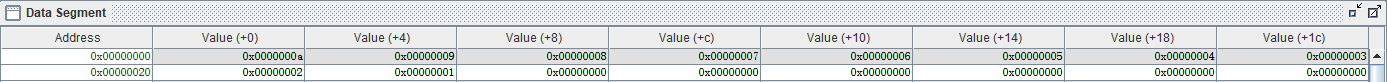


**运行效果：**

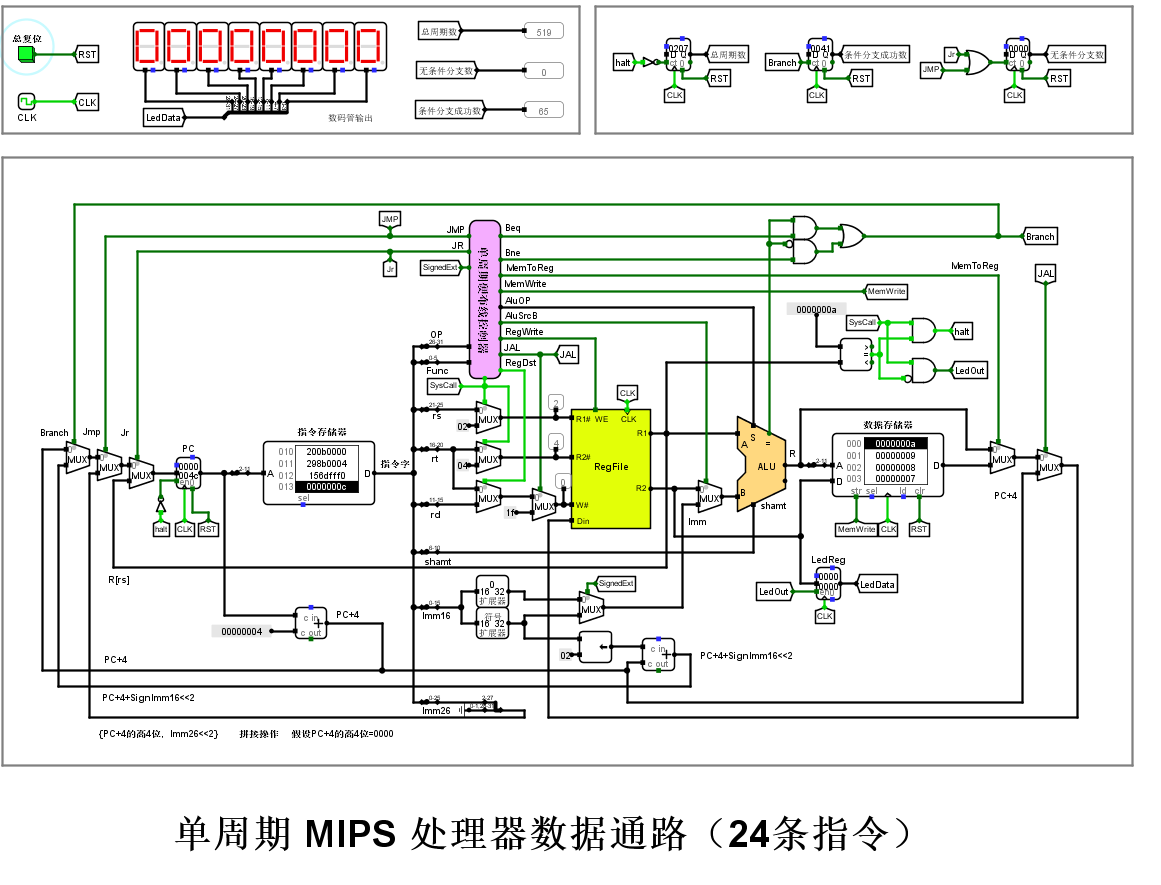
**初始：**



**结束：**



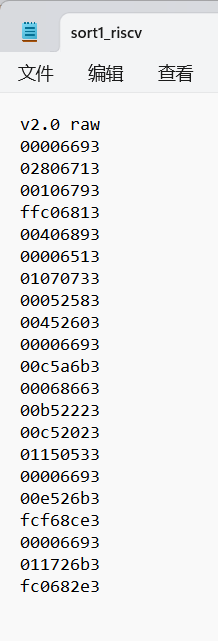
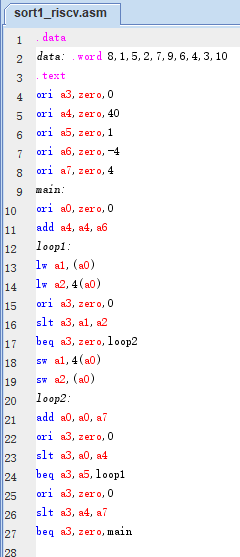
**Logisim：**



**能够根据数据储存器的数据对前10个进行排序**

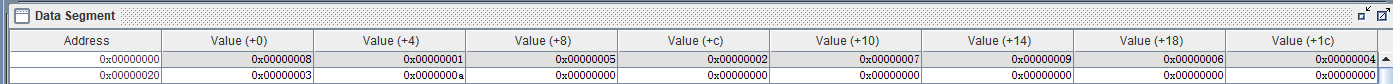
**（升序类似不再赘述）**

1. **RISC-V汇编语言程序设计（排序程序）（设计实验） ——文件名：sort1\_riscv.asm（降序排序）、sort2\_riscv.asm（升序排序）。**

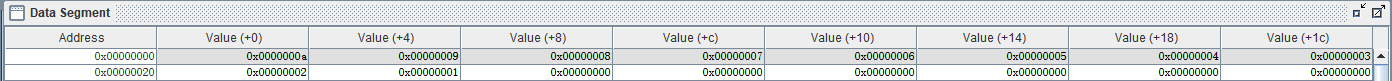


**运行效果：**

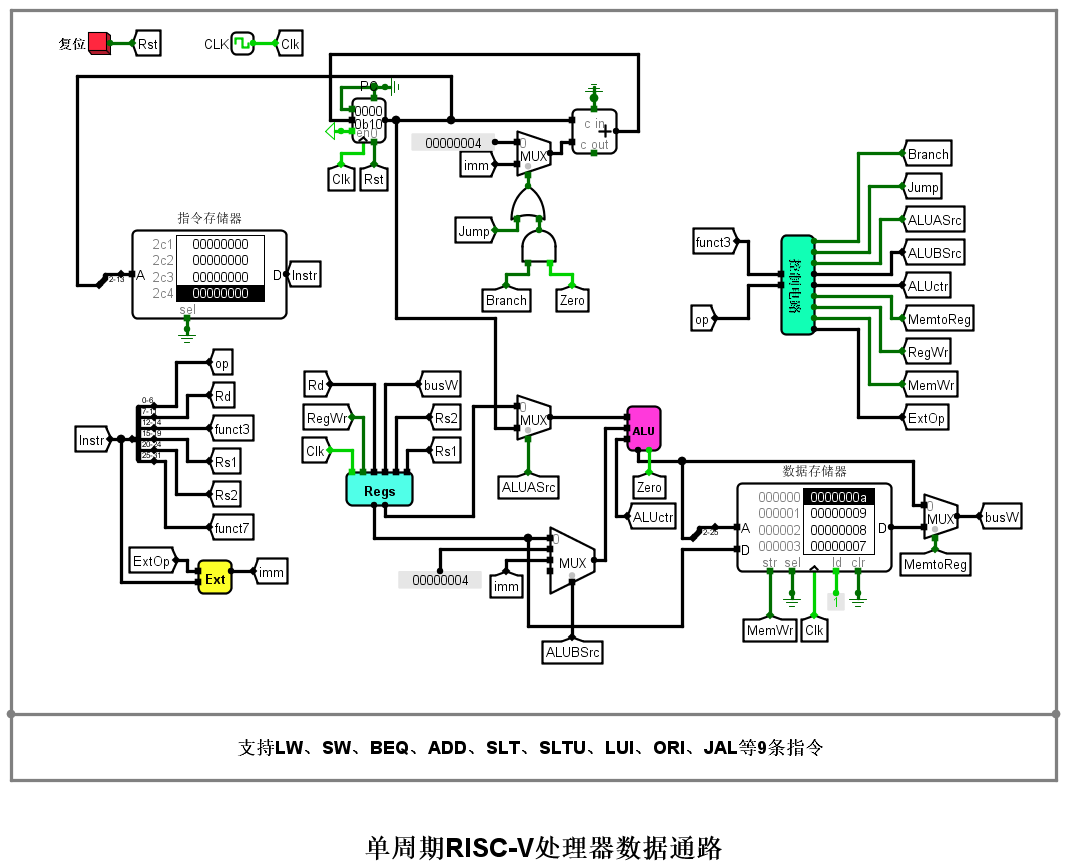
**初始：**



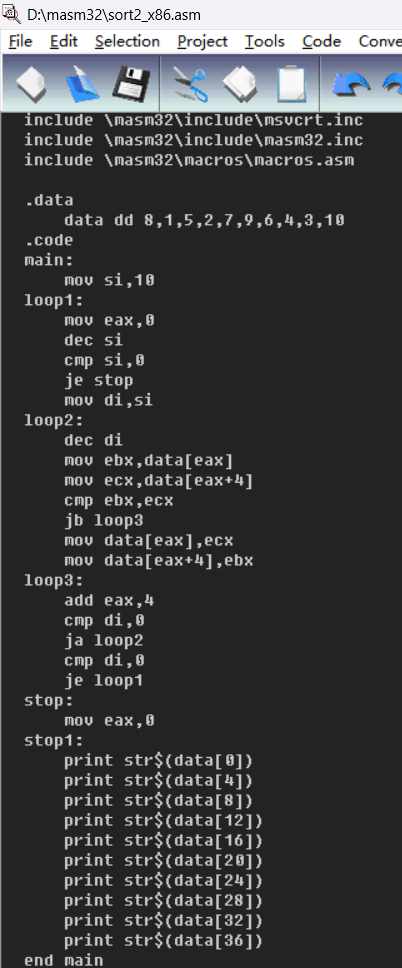
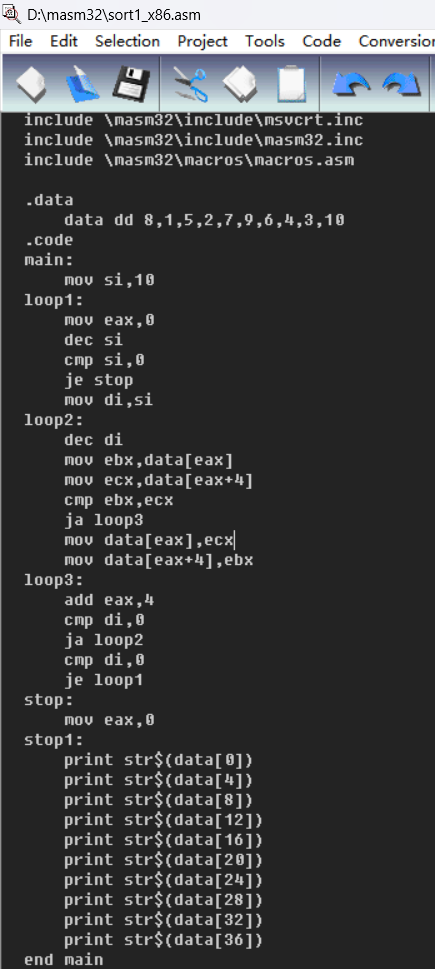
**结束：**

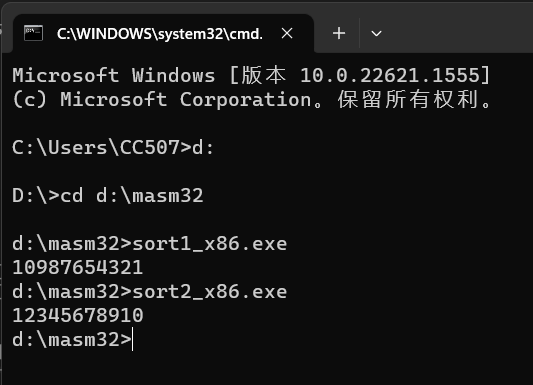


**Logisim：**



1. **Intel x86汇编语言程序设计（排序程序）（设计实验）——文件名：sort1\_x86.asm（降序排序）、sort2\_x86.asm（升序排序）。**





1. **ARMv7汇编语言程序设计（排序程序）（挑战性实验）——文件名：sort1\_armv7.s（降序排序）、sort2\_armv7.s（升序排序）。**

**请将上述设计实验（挑战性实验）的运行结果，以屏幕拷贝形式保存，黏贴在实验报告中，并给出相关的文字说明。**