

计算机组成原理实验

THCO-MIPS 模拟器 Version 2.3 使用说明文档

计 63 李思锐 leethree9@gmail.com

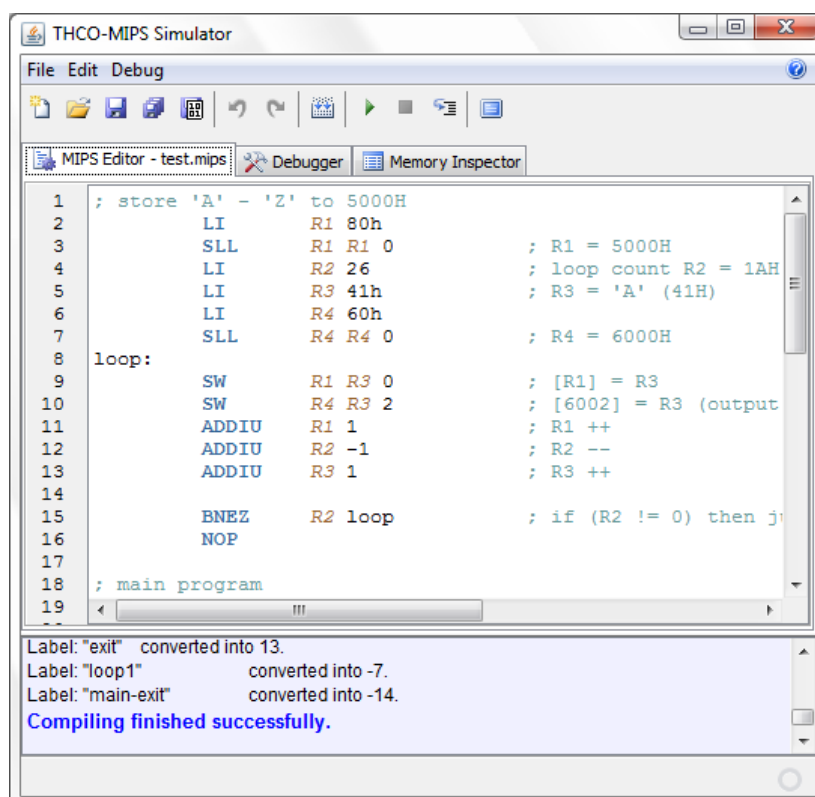
概述




THCO-MIPS Simulator 是一个基于 Java Swing Application Framework 的小程序，可用于对 THCO-MIPS 指令系统的汇编程序进行编辑、模拟和调试。设计初衷是让它比原来的命令行界面的模拟器更加便于使用。

运行环境


- Java JRE 6

主要功能



程序主界面上包括菜单栏、工具栏、三个功能标签、输出栏和状态栏，三个标签分别为编辑器 、调试器  和内存查看器 .

编辑器 Editor


编辑器标签  是一个标准的代码编辑器，左侧是行号显示，内置语法高亮显示。

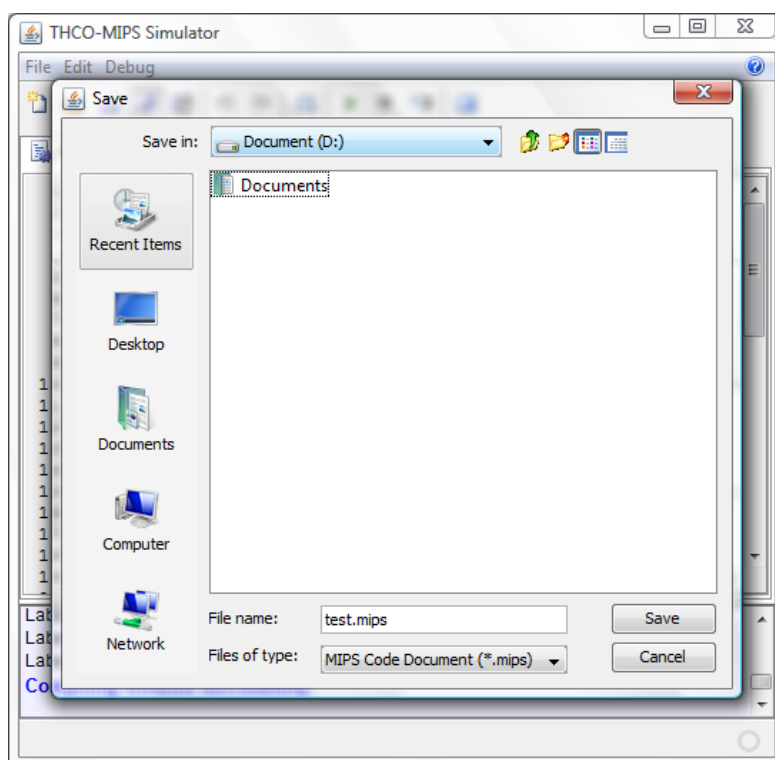
通过工具栏或菜单可以新建、打开、保存、另存为当前文件。通过 **Ctrl+Z**、**Ctrl+Y** 快捷键，或工具栏按钮可以进行撤销（undo）或回复（redo）。

语法高亮 Code highlight

语法高亮有三种形式，一是 THCO-MIPS 指令集中的指令，如 **ADDIU**，用深蓝色粗体显示。而 THCO-MIPS 中的通用寄存器，如 **R3**，用深红色斜体显示。另外代码注释（在每行结尾以 “;”、“%” 或 “#” 开头）用墨绿色显示。这样使得代码清晰，整洁而且方便检查错误。


编译和运行 Compile and run

输入 **F7** 或点击工具栏中的编译按钮 ，模拟器将开始编译编辑器中的代码。需要注意的是，在每次编译的时候模拟器都将保存当前文件，如果没有保存过将弹出保存窗口：

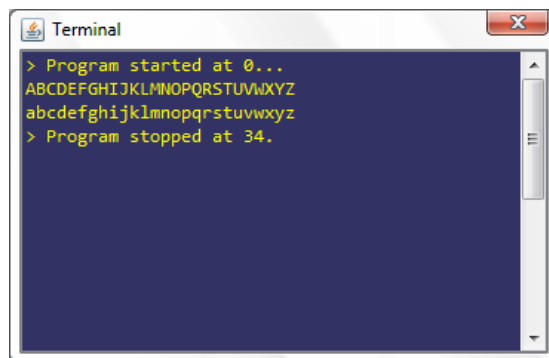


此时选择取消程序仍将正确编译，不过建议在编译以前保存文件以免丢失。

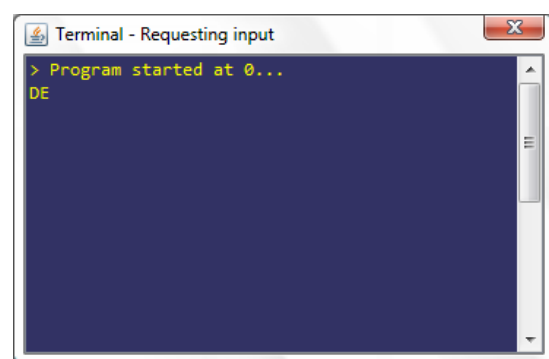
编译结果将显示在输出栏中，如果编译成功，运行、调试等功能才能够使用。

输入 **F5** 或点击工具栏中的运行按钮 ，模拟器将运行上一次编译的程序。**注意，运行时不会自动保存当前编辑器中的修改并重新编译。**

运行时将弹出终端窗口（Terminal），如下图所示：



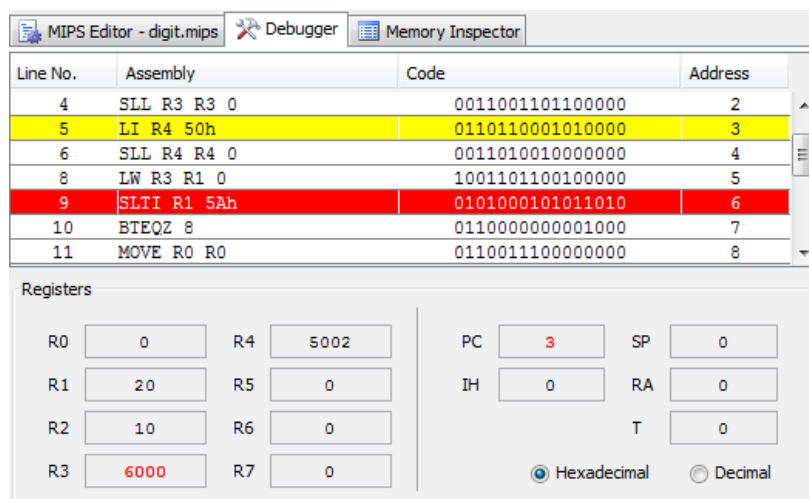
终端窗口中将显示程序输出。如果程序请求输入终端将提示“Requesting input”：



此时在终端内的输入将被正在运行的程序接受。

调试器 Debugger

调试器标签用于对所编写程序的调试，界面如图所示：





上方的表格是代码查看器，下方是寄存器查看器。


代码查看器中的代码由汇编和二进制编码两种方式表示，此外还有该语句的行号和地址。行号是指这行代码在编辑器中的行号，地址是指代码编译后所存储的内存地址。

寄存器查看器可以切换十六进制和十进制两种显示模式，其中刚变化的寄存器的值将用红色显示。

单步调试 Step

输入 **F11** 或者点击单步调试按钮 ，程序将执行当前 **PC** 寄存器所指向的指令。同时，在调试器代码查看器中可以看到黄色高亮的行是当前 **PC** 所指向的指令（即下一条要执行的指令）。


如果在单步运行中需要终端输入，将不允许继续单步执行，除非向终端提供输入或者终止程序 。

单步运行以后程序将暂停，此时如果点击运行  按钮程序将继续运行到断点或结束。


断点调试 Breakpoint

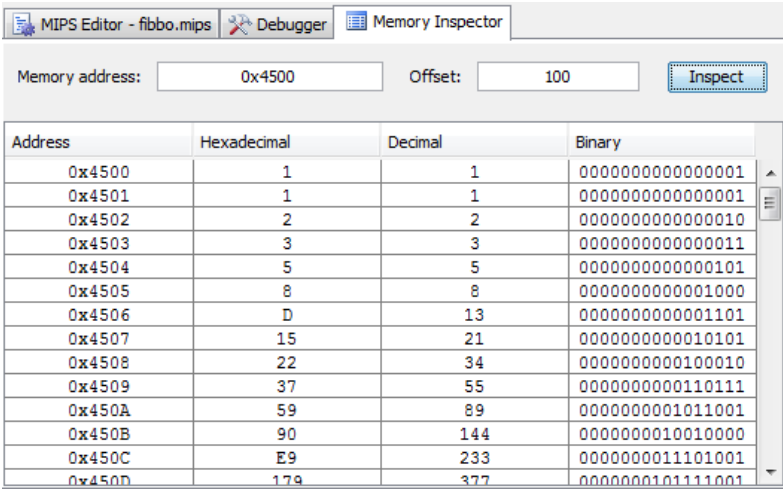
双击代码查看器中的一行，该行将变成红色，意味着在此行设置了断点。再次双击该行断点会被取消。设置和取消断点没有对应的按钮或快捷键。

设置断点后，当程序连续执行到断点位置时（即将执行的 **PC** 指向的指令）将暂停。此时断点所在行的指令尚未执行。

另外需要注意的是，连续运行  过程中代码查看器和寄存器查看器将不会刷新，直到程序遇到断点或者终止。所以，要想查看程序连续运行过程中的寄存器变化情况必须单步执行或设置断点。

内存查看器 Memory Inspector

内存查看器标签  用于查看当前模拟器中虚拟的内存空间的状况，当运行的汇编程序将数据存到指定的内存地址后，将可以通过内存查看器直接查看这些数据，如图所示：



Address	Hexadecimal	Decimal	Binary
0x4500	1	1	0000000000000001
0x4501	1	1	0000000000000001
0x4502	2	2	0000000000000010
0x4503	3	3	0000000000000011
0x4504	5	5	0000000000000101
0x4505	8	8	0000000000001000
0x4506	D	13	0000000000001101
0x4507	15	21	0000000000010101
0x4508	22	34	0000000000100010
0x4509	37	55	0000000001101111
0x450A	59	89	000000001011001
0x450B	90	144	000000010010000
0x450C	E9	233	000000011101001
0x450D	179	377	000000101111001

在上方输入要查看的地址和要查看内存空间的偏移量，点击查看（Inspect）按钮即可。

地址和偏移量接受十进制，十六进制，八进制输入，十六进制数以“0x”或“0X”开头表示，八进制数以“0”开头表示。显示的内存区间为[address, address + offset)，分别以十六进制、十进制、二进制显示。

编译说明

基本规则：

模拟器的编译过程忽略代码大小写，不再另行说明。

模拟器将按语句顺序从上到下编译，允许空行，但是一行只能有一条指令。

接受的注释：

注释必须在一行的结尾，以该行第一个“;”、“%”或“#”开头，后面部分都将被视为注释，可以为任何英文字符。

接受的指令（参考 THCO-MIPS 指令系统）：

```
sll srl sra sllv srlv srav mtsp move addu subu mfpc slt  
sltu cmp neg and or xor not mfih mtih sw-rs sw-sp sw lw-sp  
lw addiu3 addsp3 addsp addiu li slti sltui cmpi int b beqz  
bnez bteqz btnez jr jrra jalr nop
```

应该特别注意 **sw-rs**、**sw-sp**、**lw-sp** 三条指令中的连字符（hyphen）不是下划线。

接受的通用寄存器：

```
r0 r1 r2 r3 r4 r5 r6 r7
```

接受的立即数格式：

1. 十进制整数，如 0、1、-1、32。
2. 十六进制整数（以下两种表示不能同时使用）：
 - a. 以 h 结尾，如 ffh、-10h。
 - b. 以 0x 开头，如 0xff、-0x10。
3. 八进制整数以 0 开头，如 010、-070。

这里需要特别注意，在不需要使用八进制表示时数字不要以 0 开头（0 本身除外）。

接受的立即数范围：

接受的立即数范围由所在指令的编码格式确定，如 **ADDIU** 指令编码中立即数的位数为 8 位，则 **ADDIU** 指令第二个立即数参数范围为[255, -127]，超出此范围将不能通过

编译。这里需要注意，第一位为 1 的正立即数在需要符号扩展的语句中将会被理解为负数，并且编译器不会给出任何额外提示，例如以下三条语句是等价的：

```
ADDIU    R0 FFh
ADDIU    R0 255
ADDIU    R0 -1
```

接受的跳转标号：

跳转标号必须在一行的开头，以“:”作为标识。一行中冒号前的所有内容都将被视为标号（注释中的冒号除外）。

标号只允许含有字母、数字、连字符或下划线，且不能为空。标号前后允许留有空白，但是标号本身不允许存在空白字符。

标号所代表的将是标号后最近的一条语句，无论这条语句和标号在同一行还是在以后。在跳转到该标号后，这条语句会被执行。

允许将标号写在程序的最后一行，这个标号将代表程序退出的位置。

允许重复定义同一个标号，但第一次以后的定义都将被忽略。

另外，如果要跳转的标号离当前行太远可能会出现跳转偏移量超过跳转语句所支持的立即数的情况，虽然实际使用中很难遇到。

编译结果的存储

编译结果将会按照顺序从地址 0 开始连续存储到内存中，最后添加一个字长的 0 作为程序结束的标志。

运行说明

内存和寄存器的默认值

所有内存单元和寄存器在模拟器刚启动时都为 0。

内存和寄存器的清零

在整个模拟器过程中将不会自动进行内存和寄存器的清零，编写的汇编程序需要考虑内存单元为任意值的可能性。

内存访问

按照 TEC-2008 实验平台的要求，模拟器内存中 0x0000 至 0x27FF 是只能读取的，0x2800 至 0x3FFF 是无效地址，0x4000 至 0x6003 是可读可写的有效地址。非法访问其他地址将会产生模拟器运行时错误。

软中断指令 INT

模拟器暂不支持中断。在编译过程中 **INT** 指令会被编译成正确的编码，但是模拟运行中 **INT** 指令将不会产生任何作用。

延时槽

延时槽是流水线处理器的特有结构，模拟器没有实现延时槽。在程序跳转时延时槽内（跳转指令的下一条）的指令不会被执行，但为了鼓励良好的编程习惯，建议仍然在每条跳转指令后加一条空指令，如 **NOP** 或 **MOVE R0 R0**。

输入输出 IO

模拟器运行时，向终端输出字符的 IO 地址为 0x6002，从终端请求字符输入的 IO 地址为 0x6000。模拟器不支持串口通信等其他 IO 功能。

快捷键

Ctrl + N	新建	Ctrl + Z	撤销
Ctrl + O	打开	Ctrl + Y	回复
Ctrl + S	保存	F7	编译
Ctrl + Shift + S	另存为	F5	运行
Ctrl + Q	退出	Shift + F5	终止
Ctrl + X	剪切	F11	单步
Ctrl + C	复制	F1	关于
Ctrl + V	粘贴		

Acknowledgement

由于时间和水平所限，程序的疏漏和错误之处在所难免，希望使用者批评指正。

感谢以下同学在程序编写过程中给予我帮助（不分先后）：

朱晨光， 石晟， 林添， 赵威