1. MIPS 模拟器原理

1.1. 寄存器

MIPS 包含 32 个通用寄存器，没有强制性的指定寄存器的使用规则，但是在实际使用中，这些寄存器的用法都遵循着一系列的约定，同时也引入了一系列的寄存器的约定名，在调用寄存器时，应使用这些约定名，而不直接引用寄存器的编号。

编号	名称	用法
0	$zero	始终为 0，不可变
1	$at	用作汇编器的暂时变量
2-3	$v0-$v1	子函数调用返回结果
4-7	$a0-$a3	子函数调用的参数
8-15	$t0-$t7	暂时变量，子函数使用时不需保存或恢复
24-25	$t8-$t9	
16-23	$s0-$s7	子函数寄存器变量。在返回之前，子函数必须保存和恢复使用过的变量。
26-27	$k0-$k1	通常被中断或异常处理程序使用，保存一些系统参数
28	$gp	全局指针。系统维护这个指针方便存取 static 和 extern 变量。
29	$sp	堆栈指针。
30	$fp	框架指针。
31	$ra	子函数的返回地址。

另外，MIPS 还包含了两个与乘法运算器相关的两个寄存器大小的用来存放结果的地方：hi 和 lo，它们不是通用寄存器，除了用在乘除法之外，不能有其他用途。MIPS 里定义了一些指令可以向 hi 和 lo 中存入任何值。如：mfhi、mflo、mthi、mtlo。这些指令也在本模拟器中进行了实现。

1.2. MIPS 指令与汇编

1.2.1 指令格式

所有 MIPS 指令均为 32 位的指令，分为 3 中格式，R 指令、I 指令和 J 指令。

(1)R 型指令

一条 32 位的 MIPS R 型指令按下表 bit 数划分为 6 个字段：

6	5	5	5	5	6

各段的含义如下：

opcode	rs	rt	rd	shamt	funct

操作码	操作数 1	操作数 2	目标寄存器	偏移量	函数码

(2)I 型指令

I 型指令用于有立即数的指令。

一条 32 位的 MIPS I 型指令按下表 bit 数划分为 4 个字段：

6	5	5	16

各段含义如下：

Opcode 操作码	Rs 操作数	Rt 目标寄存器	Address 地址相对偏移量

(3)J 型指令

J 指令为跳转的指令

一条 32 位的 MIPS J 型指令按下表 bit 数分为 2 个字段：

6	26

各段含义如下：

opcode	target address

1.2.2 寻址方式

(1)寄存器寻址

例如 MIPS 算术运算指令的操作数必须从 32 个 32 位寄存器中选取。

如：

add $t0, $s1, $s2

sub $s0, $t0, $t1

(2)立即数寻址

以常数为操作数，无需访问储存器即可使用常数，因为常数操作出现的非常频繁，所以在算术指令中加入常数字段，比从储存器中读取常数快的多。

如：

addi $sp, $sp, 4

(3)基址或偏移寻址

操作数在储存器中，且储存器地址是某寄存器与指令中某常量的和。

如：

lw $t0, 8($s0)

(4)PC 相对寻址

由于几乎所有的条件分支指令都是跳转到附近的地址，因此使用 PC 寄存器+分支地址进行寻址。

如：

bne $s0, $s1, EXIT

(5)伪直接寻址

跳转地址=PC 高 4 位|指令中 26 位|00

如：

j 10000

此指令会跳转到 PC|10000。

本模拟器对这几种寻址方式都进行了模拟实现。

1.2.3 指令系统

本系统实现了几乎所有的 MIPS 指令，可以根据相应的功能将其分为三种指令：

1、数据传输指令

如 sw、sh、sb、lw、lh、lb、lhu、lbu、lui 等对数据进行读写的指令。

2、算术、逻辑指令

add、addu、sub、subu、and、or、xor、nor、slt、sltu

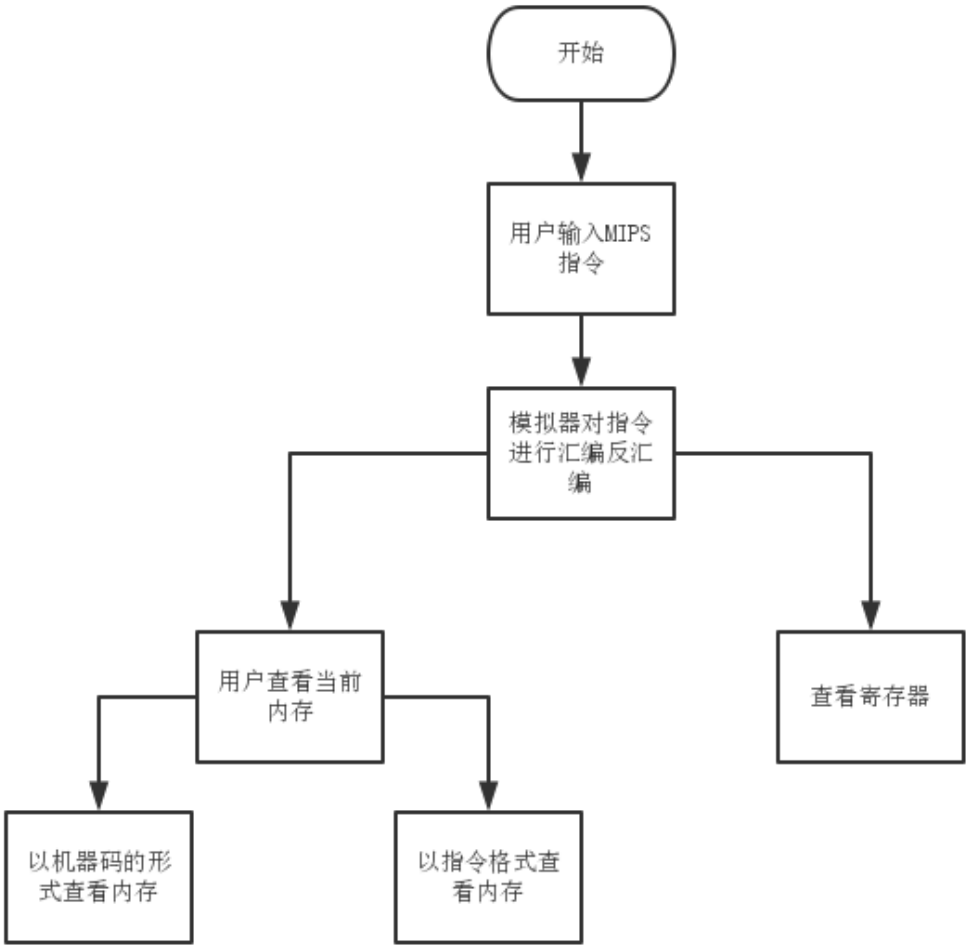
addi、addiu、slti、sltiu、andi、ori、xori

sll、srl、sra、sllv、srlv、srav

3、控制类指令

beq、bne、j、jal、jr

2.软件流程图



3.用户手册

3.1. R 命令

用户可以使用 R 命令以查看当前寄存器的值，包括 MIPS 的 32 个通用寄存器，以及 hi、lo 寄存器和当前 PC 的值。

3.2. D 命令

用户可以使用 D 命令以数据方式查看内存，即模拟器会对用户输入的 MIPS

指令进行汇编操作，将其转换为机器码存放在内存当中，D 命令可以显示当前内存中存放的 2 进制的机器码。

3.3. U 命令

用户使用 U 命令时，模拟器会将内存中的机器码反汇编成 MIPS 指令的格式，并输出到屏幕上。

3.4. A 命令

用户使用 A 命令后，可向模拟器中输入一段 MIPS 指令，输入完成后模拟器会自动对这些指令进行汇编操作，并存放在内存中。

3.5. T 命令

用户使用 T 命令，可以从当前的 PC 处开始单步执行 MIPS 指令，每使用一次 T 命令都会执行一条，并且程序会输出该命令对哪些寄存器进行了改变。

3.6. Q 命令

用户使用 Q 命令可以退出 MIPS 模拟器。

4.使用实例

一下为 MIPS 模拟器在本机上运行的截图，运行系统为 Mac OS。
初始寄存器的值均为 0

```
mac-2:mips chen$ ./mips
请输入命令:
R: 查看寄存器
D: 数据方式查看内存
U: 指令方式查看内存
A: 写汇编指令到内存
T: 单步执行内存中的指令
Q: 退出
R

PC : 0
$zero:0    $at:0    $v0:0    $v1:0
$a0:0    $a1:0    $a2:0    $a3:0
$t0:0    $t1:0    $t2:0    $t3:0
$t4:0    $t5:0    $t6:0    $t7:0
$s0:0    $s1:0    $s2:0    $s3:0
$s4:0    $s5:0    $s6:0    $s7:0
$t8:0    $t9:0    $k0:0    $k1:0
$gp:0    $sp:0    $fp:0    $ra:0
hi : 0    lo : 0
```

输入一段简单的 MIPS 指令：

```
A
请输入指令，最后一行请输入end
addi $t0,$t0,3
l1: addi $t1,$t1,1
bne $t0,$t1,l1
jr $ra
end
```

以数据方式查看内存：

```
D
0 : 00100001000010000000000000000011
4 : 00100001001010010000000000000001
8 : 00010101001010000000000000000100
12 : 00000011111000000000000000001000
```

以指令方式查看内存：

```
U
0 : addi $t0,$t0,3
4 : addi $t1,$t1,1
8 : bne $t1,$t0,l1
12 : jr $ra
```

单步执行指令，可以看到程序执行了 3 次 lable 处的指令：

```
T
addi $t0,$t0,3
T
addi $t1,$t1,1
T
bne $t1,$t0,l1
T
addi $t1,$t1,1
T
bne $t1,$t0,l1
T
addi $t1,$t1,1
T
bne $t1,$t0,l1
T
jr $ra
```

指令执行完后再次查看寄存器的值：

```
R

PC : 16
$zero:0    $at:0    $v0:0    $v1:0
$a0:0    $a1:0    $a2:0    $a3:0
$t0:3    $t1:3    $t2:0    $t3:0
$t4:0    $t5:0    $t6:0    $t7:0
$s0:0    $s1:0    $s2:0    $s3:0
$s4:0    $s5:0    $s6:0    $s7:0
$t8:0    $t9:0    $k0:0    $k1:0
$gp:0    $sp:0    $fp:0    $ra:0
hi : 0    lo : 0
```

可以看到寄存器$t0 和$t1 的值已经变为 3.