

## 一、实验题目

用RISC-V汇编语言实现二位rotate（这里用到的模拟器为<http://venus.cs61c.org/>和spike）

## 二、实验设计思路

这里要实现使a0中的数左移n位（要被移位的数为第6行的值-25，改变n可修改L7的2）后将结果储存到ra中的过程。首先将要用到的寄存器中的值归零，之后将需要的值赋值给相应的寄存器，将a1的值作为counter,在循环的过程中递减，每次循环的过程将寄存器R0中的数字左移一位。

循环的过程：对要操作的数进行判断，判断其最高位是否为1，如果是0，则直接将其值 $\times 2$ 即可得到左移后的数，如果是1，则需要将 $\times 2$ 后的值+1后(进入add1分支)在将其储存到r0中,最后将r0中的值赋值给ra。

```
.align    2
.globl    main
.type     main, @function
main:
    andi a0,a0,0
    andi a1,a1,0 #serve as a counter
    addi a2,a0,0
    andi a3,a3,0#serve as a flag
    andi ra,ra,0#store the result
    addi a0,a0,-25#the number to be rotated L6
    addi a1,a1,2#the number of the process of rotating L7
    addi a3,a3,0
    blt a3, a1,loop
loop: addi a2,a0,0
    add a0,a0,a0
    blt a2,a3,add1
    addi a2,a0,0
    addi a1,a1,-1
    blt a3,a1,loop
    addi ra,a0,0
    ret
add1: addi a0,a0,1
    addi a2,a0,0
    addi a1,a1,-1
    blt a3,a1,loop
    addi ra,a0,0
    ret
```

## 三、测试用例

【测试用例1】

将45对应的二进制数左移2, 3, 4位，测试得到的结果依次为：

ra (x1)

180

ra (x1)	360
ra (x1)	720

在spike模拟器上的结果是：（注意这里a0的值并非初始值，而是已经在运行过程中被修改过，所以这里a0的值并不是45，而是与ra的值相同。

45 2:

```

bbl loader
z 00000000 ra 000000b4 sp 7fbeada0 gp 00011dd0
tp 00000000 t0 000100a8 t1 0000000f t2 00000000
s0 00000000 s1 00000000 a0 000000b4 a1 00000000
a2 000000b4 a3 00000000 a4 00000001 a5 00000000
a6 00000000 a7 00000000 s2 00000000 s3 00000000
s4 00000000 s5 00000000 s6 00000000 s7 00000000
s8 00000000 s9 00000000 sA 00000000 sB 00000000
t3 00000000 t4 00000000 t5 00000000 t6 00000000
pc 000000b4 va 000000b4 insn ffffffff sr 80046020
User fetch segfault @ 0x000000b4

```

45 3:

```

bbl loader
z 00000000 ra 00000168 sp 7fbeada0 gp 00011dd0
tp 00000000 t0 000100a8 t1 0000000f t2 00000000
s0 00000000 s1 00000000 a0 00000168 a1 00000000
a2 00000168 a3 00000000 a4 00000001 a5 00000000
a6 00000000 a7 00000000 s2 00000000 s3 00000000
s4 00000000 s5 00000000 s6 00000000 s7 00000000
s8 00000000 s9 00000000 sA 00000000 sB 00000000
t3 00000000 t4 00000000 t5 00000000 t6 00000000
pc 00000168 va 00000168 insn ffffffff sr 80046020
User fetch segfault @ 0x00000168

```

45 4:

```

bbl loader
z 00000000 ra 000002d0 sp 7fbeada0 gp 00011dd0
tp 00000000 t0 000100a8 t1 0000000f t2 00000000
s0 00000000 s1 00000000 a0 000002d0 a1 00000000
a2 000002d0 a3 00000000 a4 00000001 a5 00000000
a6 00000000 a7 00000000 s2 00000000 s3 00000000
s4 00000000 s5 00000000 s6 00000000 s7 00000000
s8 00000000 s9 00000000 sA 00000000 sB 00000000
t3 00000000 t4 00000000 t5 00000000 t6 00000000
pc 000002d0 va 000002d0 insn ffffffff sr 80046020
User fetch segfault @ 0x000002d0

```

### 【测试用例2】

将-25对应的二进制数左移2,3,4位，观察结果

-25对应的二进制补码为1111 1111 1110 0111

-97对应的二进制补码为1111 1111 1001 1111

-193对应的二进制补码为1111 1111 0011 1111

-385对应的二进制补码为1111 1110 0111 1111

ra (x1)	<input type="text" value="-97"/>
ra (x1)	<input type="text" value="-193"/>
ra (x1)	<input type="text" value="-385"/>

-23 2:

```
bbl loader
z 00000000 ra ffffffff9f sp 7fbeada0 gp 00011dd0
tp 00000000 t0 000100a8 t1 0000000f t2 00000000
s0 00000000 s1 00000000 a0 ffffffff9f a1 00000000
a2 ffffffff9f a3 00000000 a4 00000001 a5 00000000
a6 00000000 a7 00000000 s2 00000000 s3 00000000
s4 00000000 s5 00000000 s6 00000000 s7 00000000
s8 00000000 s9 00000000 sA 00000000 sB 00000000
t3 00000000 t4 00000000 t5 00000000 t6 00000000
pc ffffffff9e va ffffffff9e insn ffffffff sr 80046020
User fetch segfault @ 0xffffffff9e
willing@willing-virtual-machine:~$ vim rotate.s
```

-25 3:

```
willing@willing-virtual-machine:~$ vim rotate.s
bbl loader
z 00000000 ra ffffffff3f sp 7fbeada0 gp 00011dd0
tp 00000000 t0 000100a8 t1 0000000f t2 00000000
s0 00000000 s1 00000000 a0 ffffffff3f a1 00000000
a2 ffffffff3f a3 00000000 a4 00000001 a5 00000000
a6 00000000 a7 00000000 s2 00000000 s3 00000000
s4 00000000 s5 00000000 s6 00000000 s7 00000000
s8 00000000 s9 00000000 sA 00000000 sB 00000000
t3 00000000 t4 00000000 t5 00000000 t6 00000000
pc ffffffff3e va ffffffff3e insn ffffffff sr 80046020
User fetch segfault @ 0xffffffff3e
```

-25 4:

```
z 00000000 ra fffffffe7f sp 7fbeada0 gp 00011dd0
tp 00000000 t0 000100a8 t1 0000000f t2 00000000
s0 00000000 s1 00000000 a0 fffffffe7f a1 00000000
a2 fffffffe7f a3 00000000 a4 00000001 a5 00000000
a6 00000000 a7 00000000 s2 00000000 s3 00000000
s4 00000000 s5 00000000 s6 00000000 s7 00000000
s8 00000000 s9 00000000 sA 00000000 sB 00000000
t3 00000000 t4 00000000 t5 00000000 t6 00000000
pc fffffffe7e va fffffffe7e insn ffffffff sr 80046020
User fetch segfault @ 0xffffffff
wulian@wulian-virtual-machine:~$
```

测试结果与理论要求完全符合。