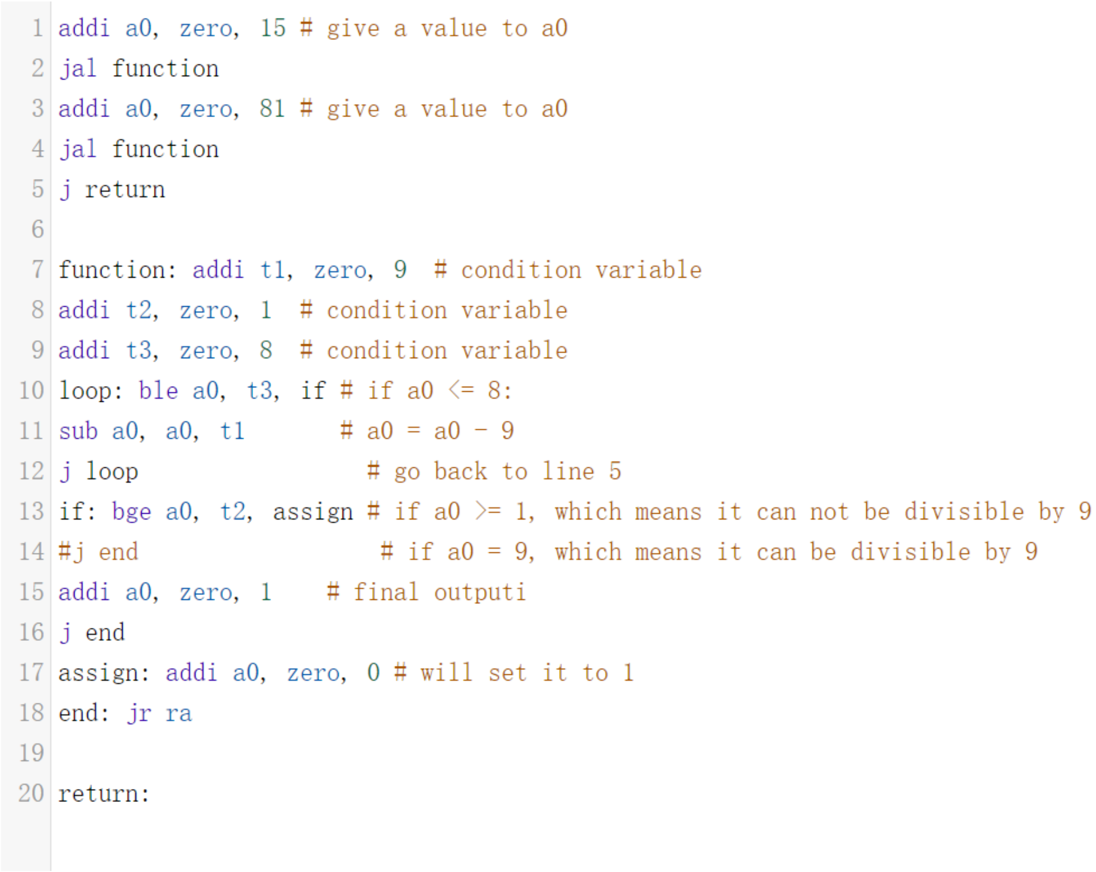
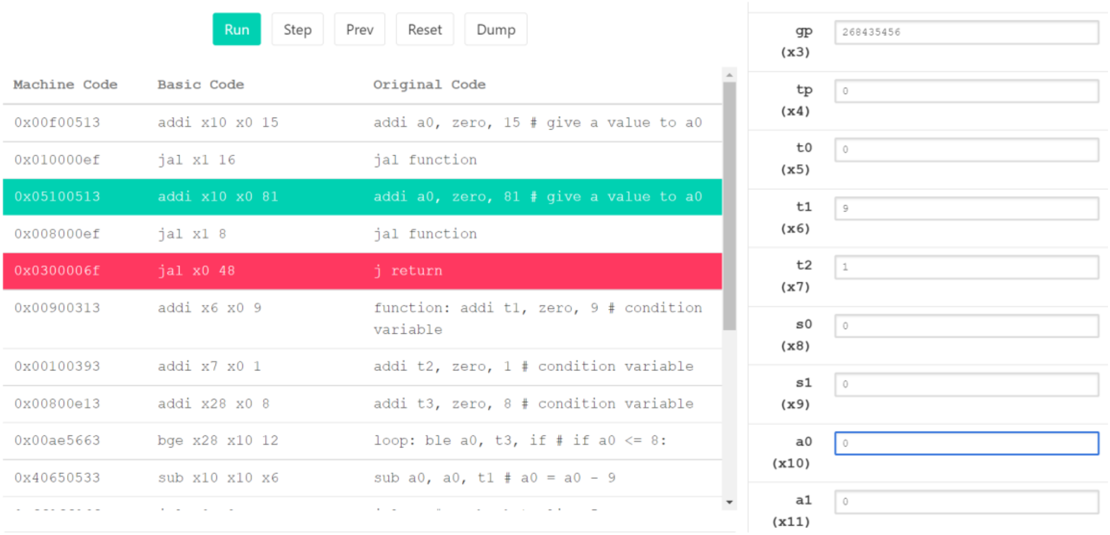
Lab 6

div9Function.s：



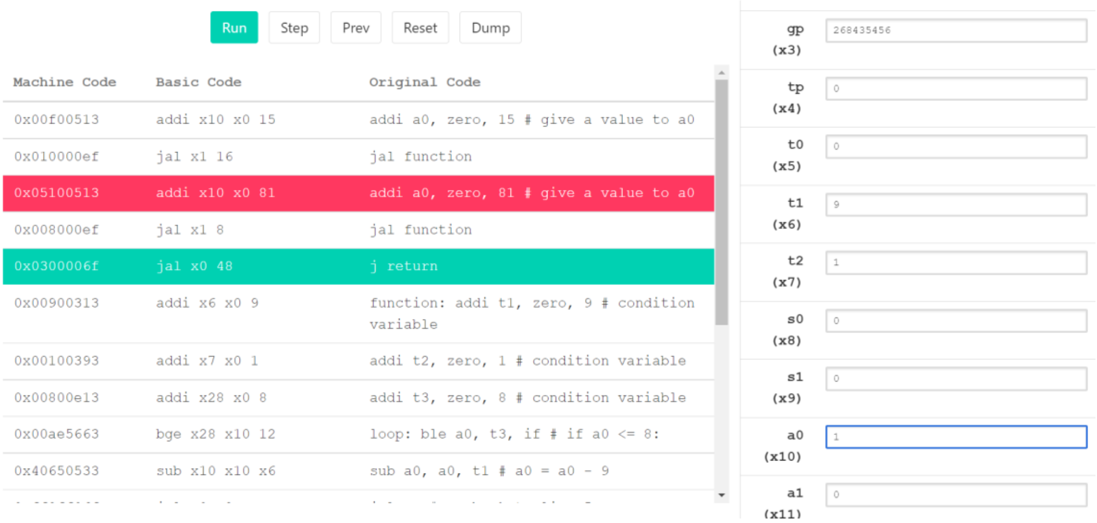
```
1  addi a0, zero, 15 # give a value to a0
2  jal function
3  addi a0, zero, 81 # give a value to a0
4  jal function
5  j return
6
7  function: addi t1, zero, 9  # condition variable
8  addi t2, zero, 1  # condition variable
9  addi t3, zero, 8  # condition variable
10 loop: ble a0, t3, if # if a0 <= 8:
11 sub a0, a0, t1       # a0 = a0 - 9
12 j loop                 # go back to line 5
13 if: bge a0, t2, assign # if a0 >= 1, which means it can not be divisible by 9
14 #j end                   # if a0 = 9, which means it can be divisible by 9
15 addi a0, zero, 1    # final outputi
16 j end
17 assign: addi a0, zero, 0 # will set it to 1
18 end: jr ra
19
20 return:
```



Machine Code	Basic Code	Original Code
0x00f00513	addi x10 x0 15	addi a0, zero, 15 # give a value to a0
0x010000ef	jal x1 16	jal function
0x05100513	addi x10 x0 81	addi a0, zero, 81 # give a value to a0
0x008000ef	jal x1 8	jal function
0x0300006f	jal x0 48	j return
0x00900313	addi x6 x0 9	function: addi t1, zero, 9 # condition variable
0x00100393	addi x7 x0 1	addi t2, zero, 1 # condition variable
0x00800e13	addi x28 x0 8	addi t3, zero, 8 # condition variable
0x00ae5663	bge x28 x10 12	loop: ble a0, t3, if # if a0 <= 8:
0x40650533	sub x10 x10 x6	sub a0, a0, t1 # a0 = a0 - 9

gp (x3)	268435456
tp (x4)	0
t0 (x5)	0
t1 (x6)	9
t2 (x7)	1
s0 (x8)	0
s1 (x9)	0
a0 (x10)	0
a1 (x11)	0

When input is 15 which can not be divided by 9 returns a0 of 0



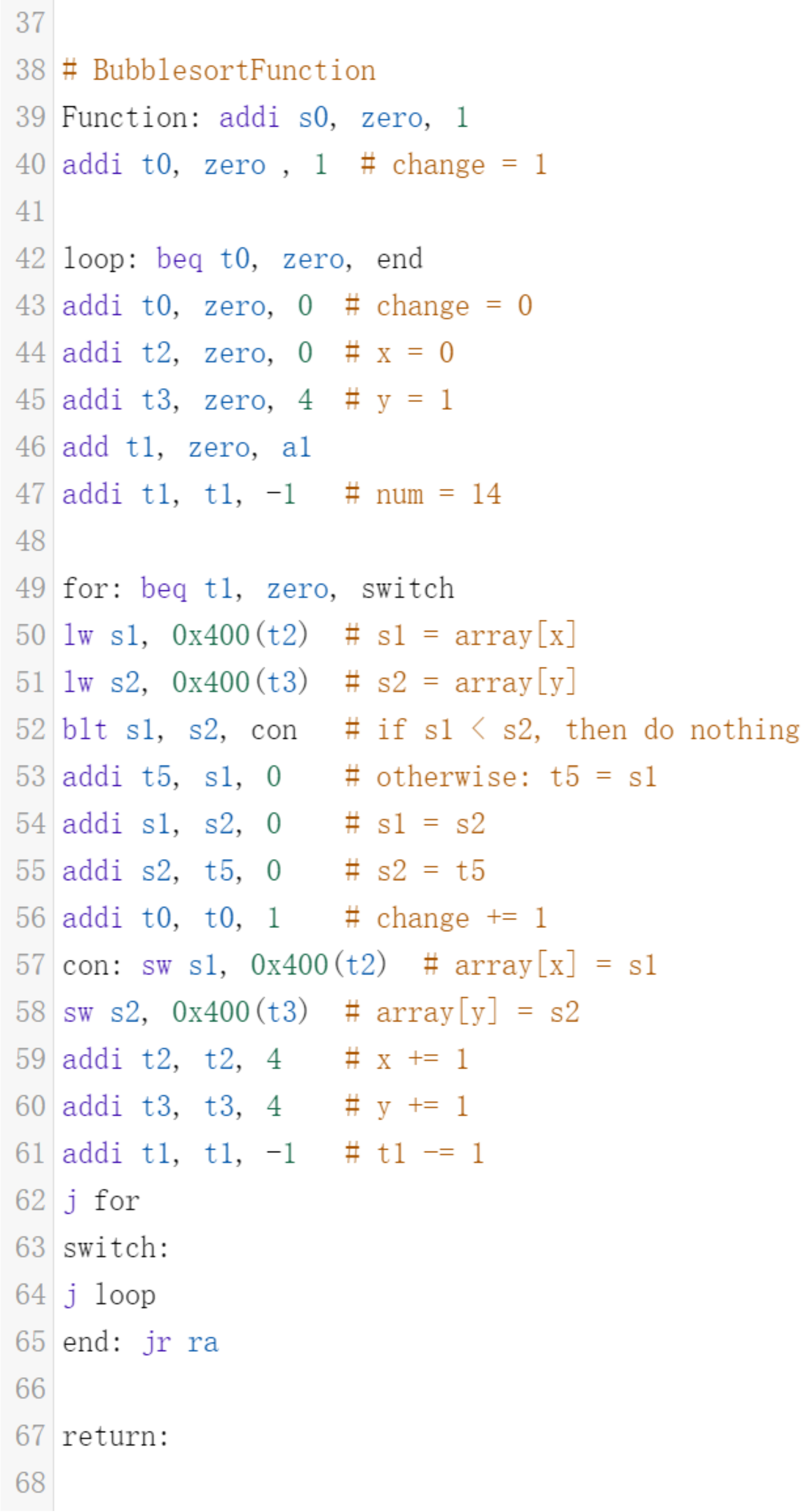
Machine Code	Basic Code	Original Code
0x00f00513	addi x10 x0 15	addi a0, zero, 15 # give a value to a0
0x010000ef	jal x1 16	jal function
0x05100513	addi x10 x0 81	addi a0, zero, 81 # give a value to a0
0x008000ef	jal x1 8	jal function
0x0300006f	jal x0 48	j return
0x00900313	addi x6 x0 9	function: addi t1, zero, 9 # condition variable
0x00100393	addi x7 x0 1	addi t2, zero, 1 # condition variable
0x00800e13	addi x28 x0 8	addi t3, zero, 8 # condition variable
0x00ae5663	bge x28 x10 12	loop: ble a0, t3, if # if a0 <= 8:
0x40650533	sub x10 x10 x6	sub a0, a0, t1 # a0 = a0 - 9

gp (x3)	268435456
tp (x4)	0
t0 (x5)	0
t1 (x6)	9
t2 (x7)	1
s0 (x8)	0
s1 (x9)	0
a0 (x10)	1
a1 (x11)	0

When input is 81 which can be divided by 9 returns a0 of 1

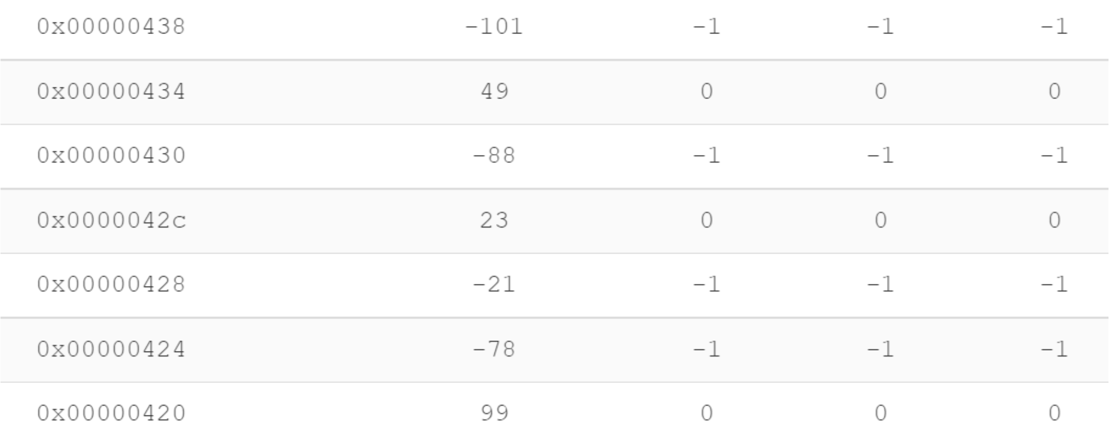
bubblesortFunction.s:

```
addi s0, zero, -15
sw s0, 0x400(zero) # memory[400] = -15
addi s0, zero, 42
sw s0, 0x404(zero) # memory[404] = 42
addi s0, zero, 73
sw s0, 0x408(zero) # memory[408] = 73
addi s0, zero, 19
sw s0, 0x40c(zero) # memory[40c] = 19
addi s0, zero, -8
sw s0, 0x410(zero) # memory[410] = -8
addi s0, zero, 24
sw s0, 0x414(zero) # memory[414] = 24
addi s0, zero, 16
sw s0, 0x418(zero) # memory[418] = 16
addi s0, zero, -2
sw s0, 0x41c(zero) # memory[41c] = -2
addi s0, zero, 99
sw s0, 0x420(zero) # memory[420] = 99
addi s0, zero, -78
sw s0, 0x424(zero) # memory[424] = -78
addi s0, zero, -21
sw s0, 0x428(zero) # memory[428] = -21
addi s0, zero, 23
sw s0, 0x42c(zero) # memory[42c] = 23
addi s0, zero, -88
sw s0, 0x430(zero) # memory[430] = -88
addi s0, zero, 49
sw s0, 0x434(zero) # memory[434] = 49
addi s0, zero, -101
sw s0, 0x438(zero) # memory[438] = -101

# main function part
addi a0, zero, 0x400
addi a1, zero, 15
jal Function
j return
```

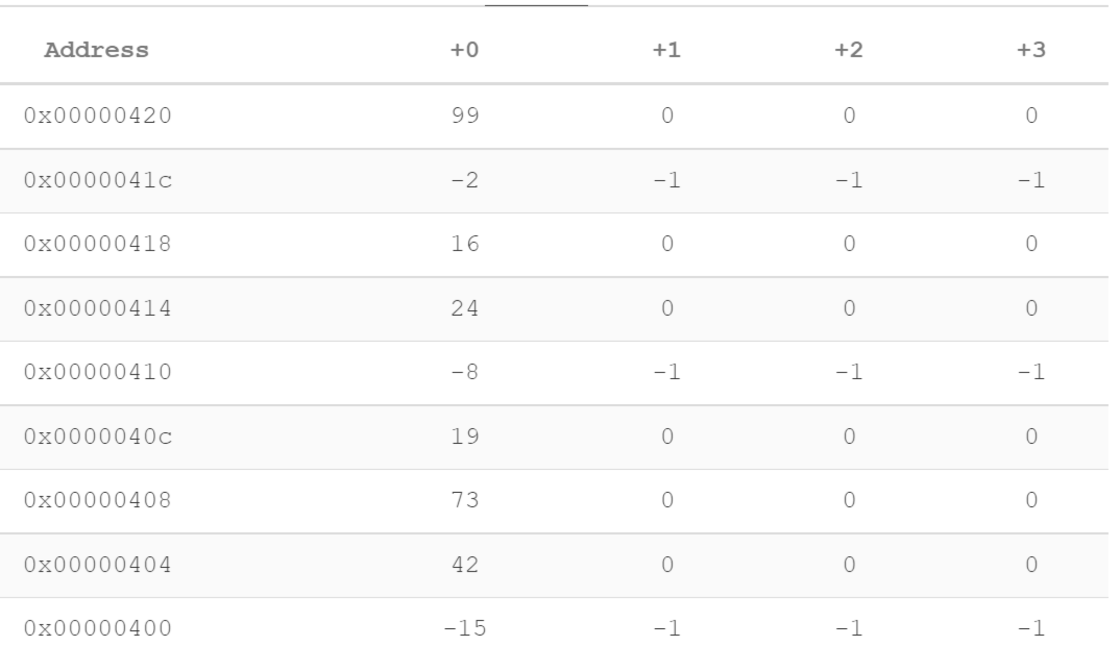


```

# BubblesortFunction
Function: addi s0, zero, 1
addi t0, zero , 1  # change = 1

loop: beq t0, zero, end
addi t0, zero, 0  # change = 0
addi t2, zero, 0  # x = 0
addi t3, zero, 4  # y = 1
add t1, zero, a1
addi t1, t1, -1   # num = 14

for: beq t1, zero, switch
lw s1, 0x400(t2)  # s1 = array[x]
lw s2, 0x400(t3)  # s2 = array[y]
blt s1, s2, con   # if s1 < s2, then do nothing
addi t5, s1, 0    # otherwise: t5 = s1
addi s1, s2, 0    # s1 = s2
addi s2, t5, 0    # s2 = t5
addi t0, t0, 1    # change += 1
con: sw s1, 0x400(t2)  # array[x] = s1
sw s2, 0x400(t3)  # array[y] = s2
addi t2, t2, 4    # x += 1
addi t3, t3, 4    # y += 1
addi t1, t1, -1   # t1 -= 1
j for
switch:
j loop
end: jr ra

return:

```

Before:

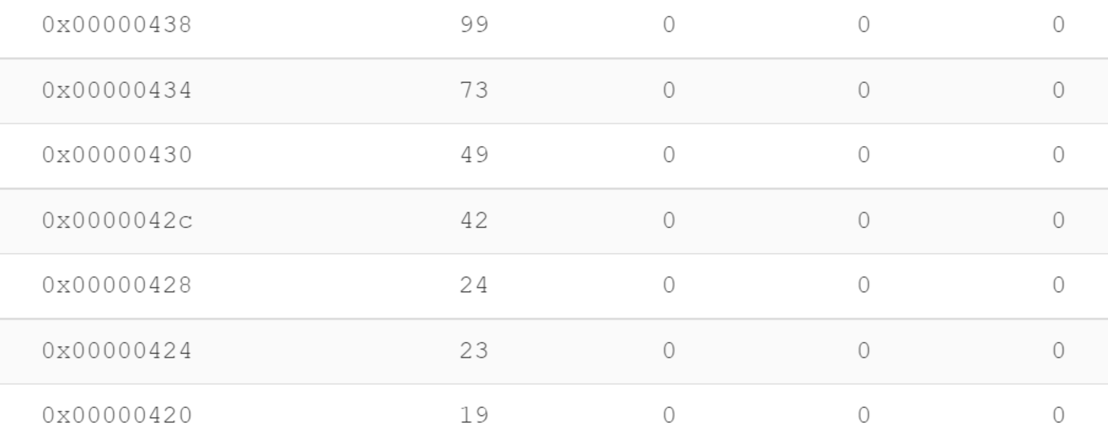


0x00000438	-101	-1	-1	-1
0x00000434	49	0	0	0
0x00000430	-88	-1	-1	-1
0x0000042c	23	0	0	0
0x00000428	-21	-1	-1	-1
0x00000424	-78	-1	-1	-1
0x00000420	99	0	0	0

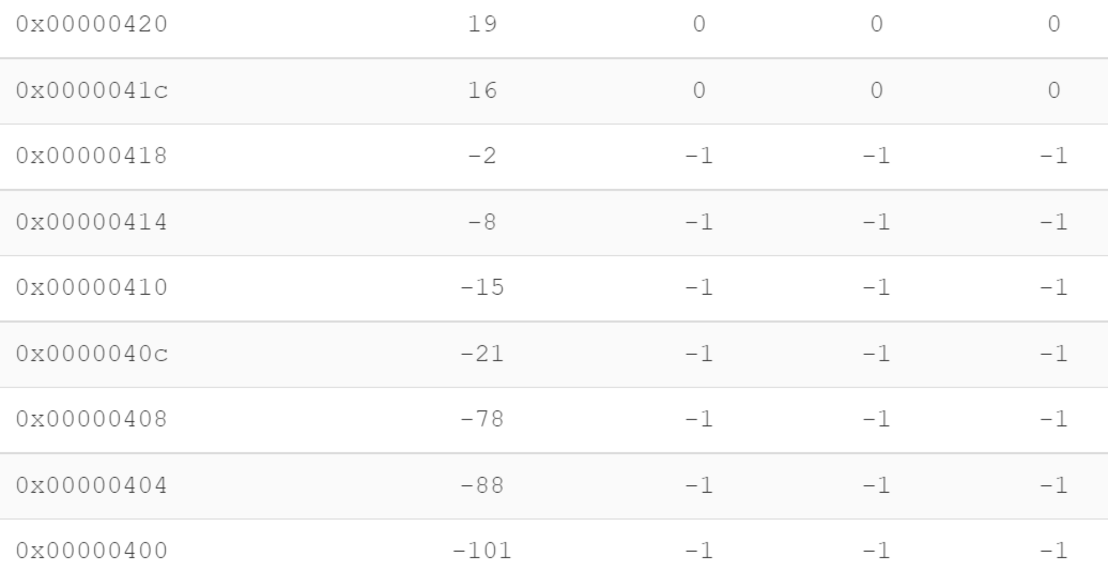


Address	+0	+1	+2	+3
0x00000420	99	0	0	0
0x0000041c	-2	-1	-1	-1
0x00000418	16	0	0	0
0x00000414	24	0	0	0
0x00000410	-8	-1	-1	-1
0x0000040c	19	0	0	0
0x00000408	73	0	0	0
0x00000404	42	0	0	0
0x00000400	-15	-1	-1	-1

After:

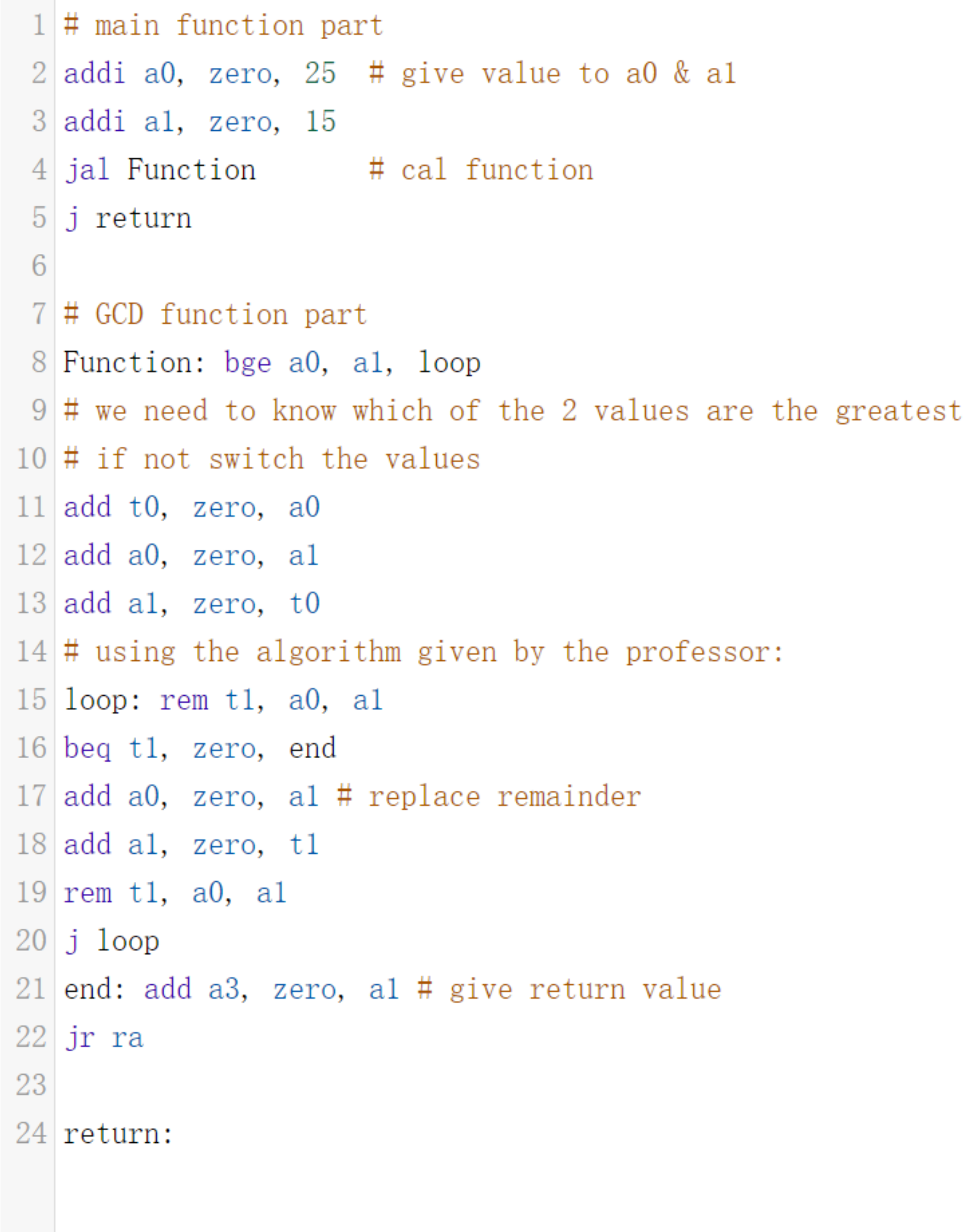


0x00000438	99	0	0	0
0x00000434	73	0	0	0
0x00000430	49	0	0	0
0x0000042c	42	0	0	0
0x00000428	24	0	0	0
0x00000424	23	0	0	0
0x00000420	19	0	0	0



0x00000420	19	0	0	0
0x0000041c	16	0	0	0
0x00000418	-2	-1	-1	-1
0x00000414	-8	-1	-1	-1
0x00000410	-15	-1	-1	-1
0x0000040c	-21	-1	-1	-1
0x00000408	-78	-1	-1	-1
0x00000404	-88	-1	-1	-1
0x00000400	-101	-1	-1	-1

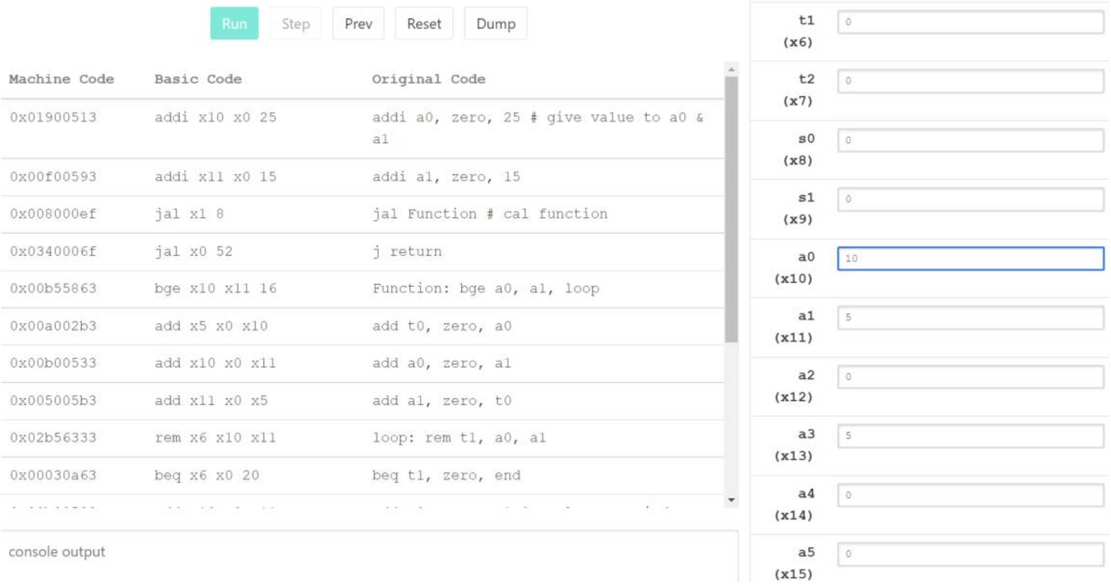
gcd.s:



```
# main function part
addi a0, zero, 25  # give value to a0 & a1
addi a1, zero, 15
jal Function       # cal function
j return

# GCD function part
Function: bge a0, a1, loop
# we need to know which of the 2 values are the greatest
# if not switch the values
add t0, zero, a0
add a0, zero, a1
add a1, zero, t0
# using the algorithm given by the professor:
loop: rem t1, a0, a1
beq t1, zero, end
add a0, zero, a1 # replace remainder
add a1, zero, t1
rem t1, a0, a1
j loop
end: add a3, zero, a1 # give return value
jr ra

return:
```

Note that the return value is stored in a3 register

gcd(25, 15);

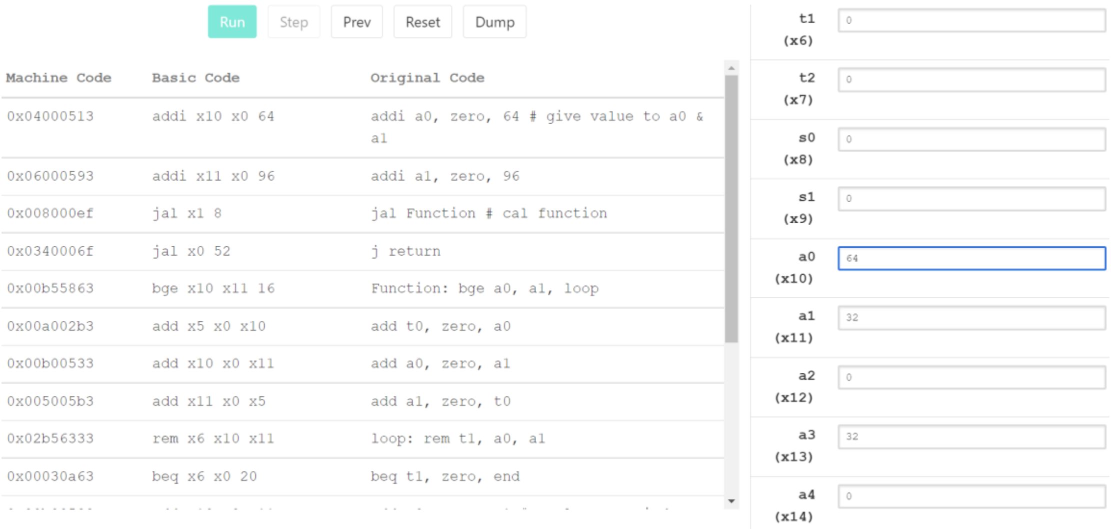


Machine Code	Basic Code	Original Code
0x01900513	addi x10 x0 25	addi a0, zero, 25 # give value to a0 & a1
0x00f00593	addi x11 x0 15	addi a1, zero, 15
0x008000ef	jal x1 8	jal Function # cal function
0x0340006f	jal x0 52	j return
0x00b55863	bge x10 x11 16	Function: bge a0, a1, loop
0x00a002b3	add x5 x0 x10	add t0, zero, a0
0x00b00533	add x10 x0 x11	add a0, zero, a1
0x005005b3	add x11 x0 x5	add a1, zero, t0
0x02b56333	rem x6 x10 x11	loop: rem t1, a0, a1
0x00030a63	beq x6 x0 20	beq t1, zero, end

console output

t1 (x6)	0
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	10
a1 (x11)	5
a2 (x12)	0
a3 (x13)	5
a4 (x14)	0
a5 (x15)	0

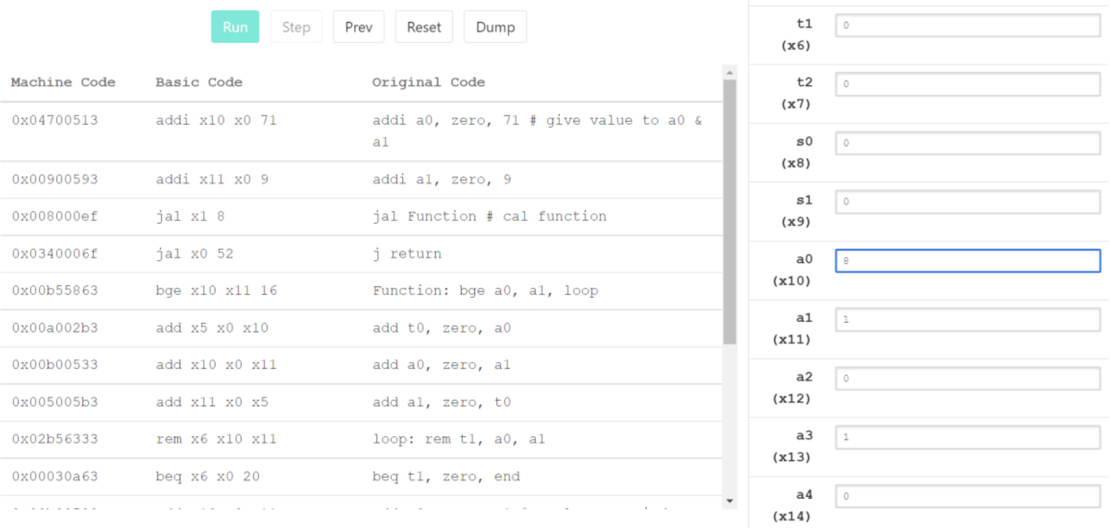
gcd(64, 96);



Machine Code	Basic Code	Original Code
0x04000513	addi x10 x0 64	addi a0, zero, 64 # give value to a0 & a1
0x06000593	addi x11 x0 96	addi a1, zero, 96
0x008000ef	jal x1 8	jal Function # cal function
0x0340006f	jal x0 52	j return
0x00b55863	bge x10 x11 16	Function: bge a0, a1, loop
0x00a002b3	add x5 x0 x10	add t0, zero, a0
0x00b00533	add x10 x0 x11	add a0, zero, a1
0x005005b3	add x11 x0 x5	add a1, zero, t0
0x02b56333	rem x6 x10 x11	loop: rem t1, a0, a1
0x00030a63	beq x6 x0 20	beq t1, zero, end

t1 (x6)	0
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	64
a1 (x11)	32
a2 (x12)	0
a3 (x13)	32
a4 (x14)	0

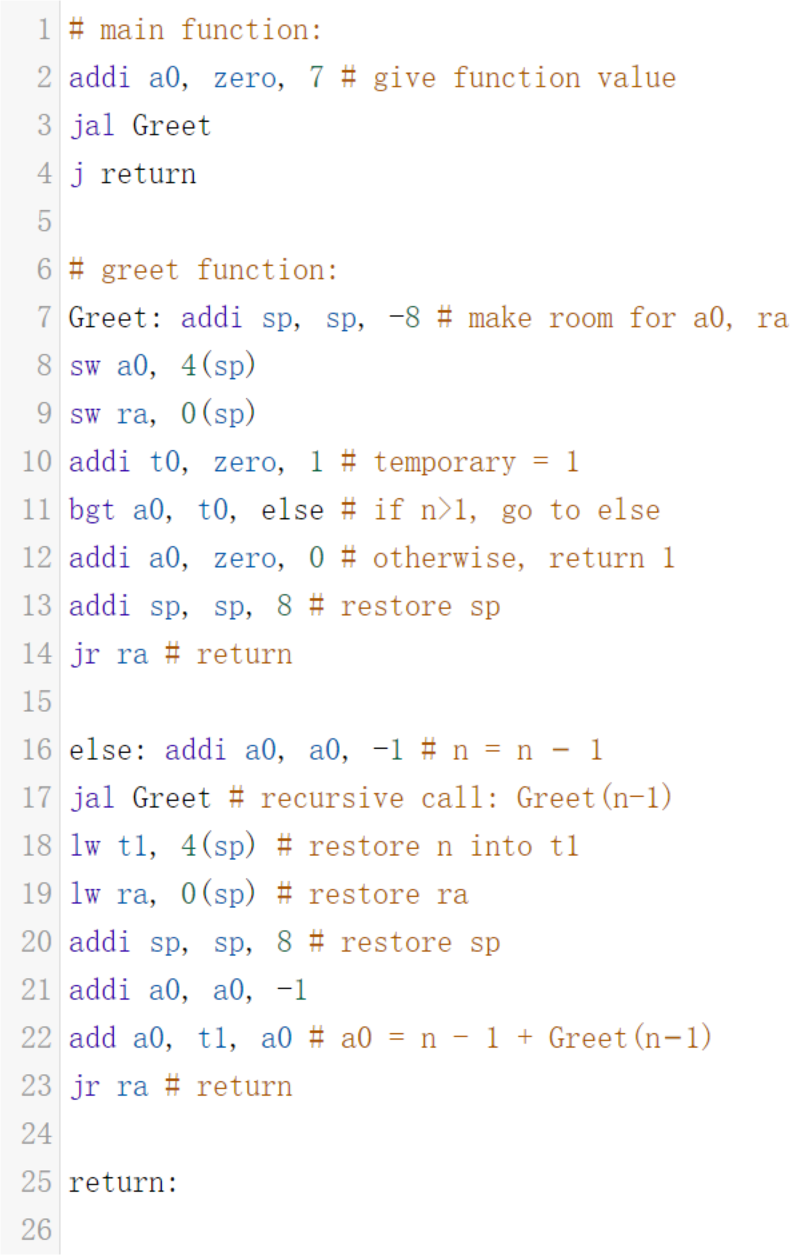
gcd(71, 9);



Machine Code	Basic Code	Original Code
0x04700513	addi x10 x0 71	addi a0, zero, 71 # give value to a0 & a1
0x00900593	addi x11 x0 9	addi a1, zero, 9
0x008000ef	jal x1 8	jal Function # cal function
0x0340006f	jal x0 52	j return
0x00b55863	bge x10 x11 16	Function: bge a0, a1, loop
0x00a002b3	add x5 x0 x10	add t0, zero, a0
0x00b00533	add x10 x0 x11	add a0, zero, a1
0x005005b3	add x11 x0 x5	add a1, zero, t0
0x02b56333	rem x6 x10 x11	loop: rem t1, a0, a1
0x00030a63	beq x6 x0 20	beq t1, zero, end

t1 (x6)	0
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	8
a1 (x11)	1
a2 (x12)	0
a3 (x13)	1
a4 (x14)	0

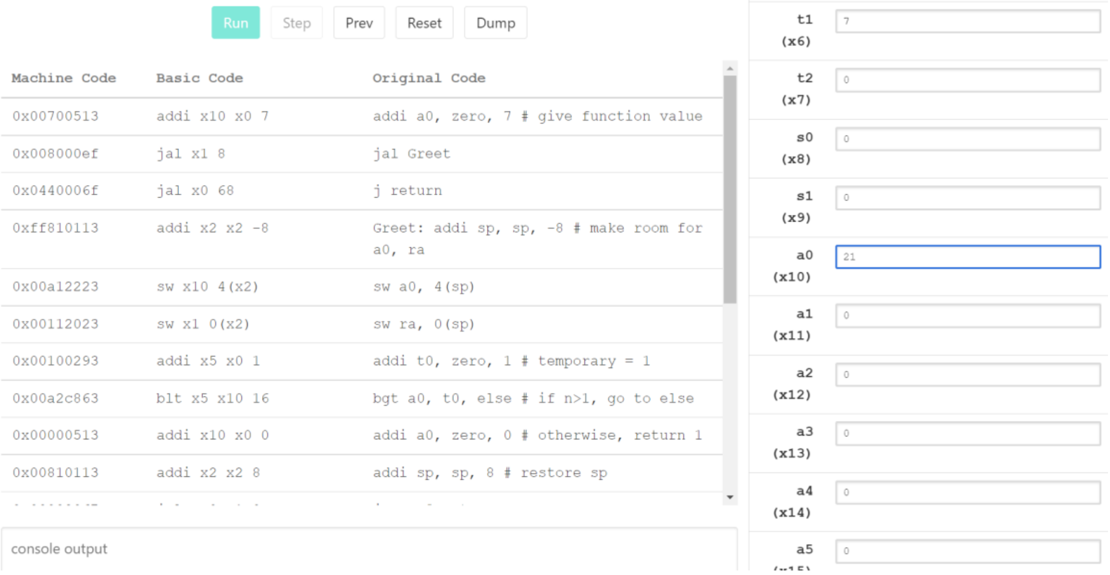
greet.s:



```
# main function:
addi a0, zero, 7 # give function value
jal Greet
j return

# greet function:
Greet: addi sp, sp, -8 # make room for a0, ra
sw a0, 4(sp)
sw ra, 0(sp)
addi t0, zero, 1 # temporary = 1
bgt a0, t0, else # if n>1, go to else
addi a0, zero, 0 # otherwise, return 1
addi sp, sp, 8 # restore sp
jr ra # return

else: addi a0, a0, -1 # n = n - 1
jal Greet # recursive call: Greet(n-1)
lw t1, 4(sp) # restore n into t1
lw ra, 0(sp) # restore ra
addi sp, sp, 8 # restore sp
addi a0, a0, -1
add a0, t1, a0 # a0 = n - 1 + Greet(n-1)
jr ra # return

return:

```

\

greet(7);

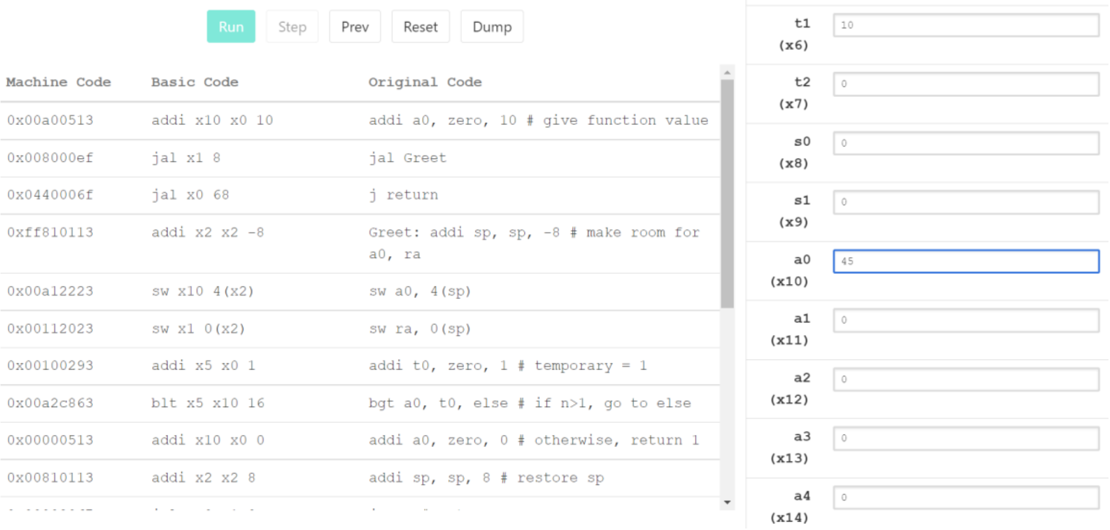


Machine Code	Basic Code	Original Code
0x00700513	addi x10 x0 7	addi a0, zero, 7 # give function value
0x008000ef	jal x1 8	jal Greet
0x0440006f	jal x0 68	j return
0xff810113	addi x2 x2 -8	Greet: addi sp, sp, -8 # make room for a0, ra
0x00a12223	sw x10 4(x2)	sw a0, 4(sp)
0x00112023	sw x1 0(x2)	sw ra, 0(sp)
0x00100293	addi x5 x0 1	addi t0, zero, 1 # temporary = 1
0x00a2c863	blt x5 x10 16	bgt a0, t0, else # if n>1, go to else
0x00000513	addi x10 x0 0	addi a0, zero, 0 # otherwise, return 1
0x00810113	addi x2 x2 8	addi sp, sp, 8 # restore sp

console output

t1 (x6)	7
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	21
a1 (x11)	0
a2 (x12)	0
a3 (x13)	0
a4 (x14)	0
a5	0

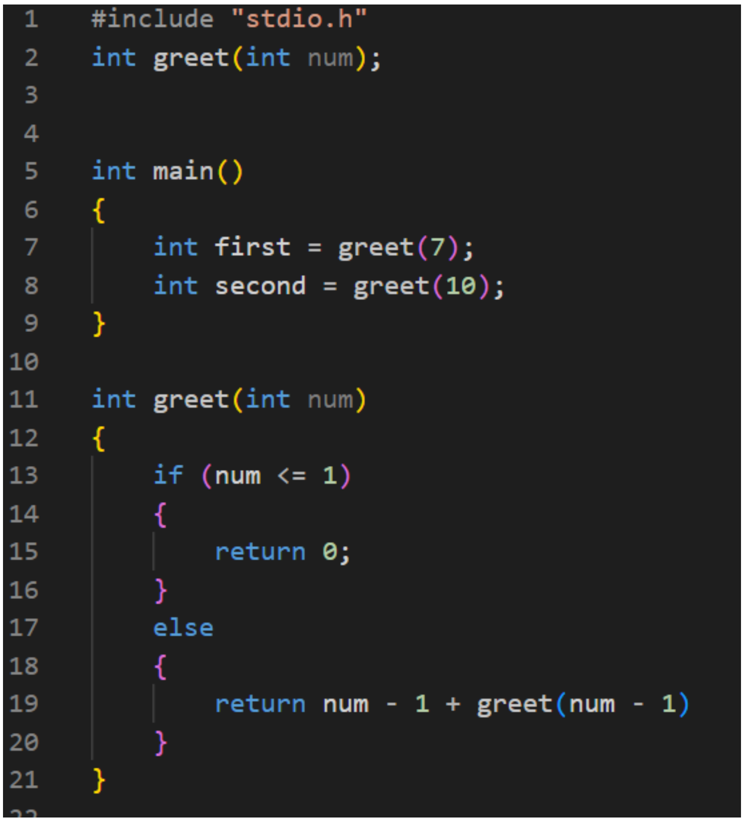
greet(10);



Machine Code	Basic Code	Original Code
0x00a00513	addi x10 x0 10	addi a0, zero, 10 # give function value
0x008000ef	jal x1 8	jal Greet
0x0440006f	jal x0 68	j return
0xff810113	addi x2 x2 -8	Greet: addi sp, sp, -8 # make room for a0, ra
0x00a12223	sw x10 4(x2)	sw a0, 4(sp)
0x00112023	sw x1 0(x2)	sw ra, 0(sp)
0x00100293	addi x5 x0 1	addi t0, zero, 1 # temporary = 1
0x00a2c863	blt x5 x10 16	bgt a0, t0, else # if n>1, go to else
0x00000513	addi x10 x0 0	addi a0, zero, 0 # otherwise, return 1
0x00810113	addi x2 x2 8	addi sp, sp, 8 # restore sp

t1 (x6)	10
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	45
a1 (x11)	0
a2 (x12)	0
a3 (x13)	0
a4 (x14)	0

greet.c:



```c
#include "stdio.h"
int greet(int num);


int main()
{
    int first = greet(7);
    int second = greet(10);
}

int greet(int num)
{
    if (num <= 1)
    {
        return 0;
    }
    else
    {
        return num - 1 + greet(num - 1)
    }
}
```

Time used: 8hours

No bugs or suggestions at this point