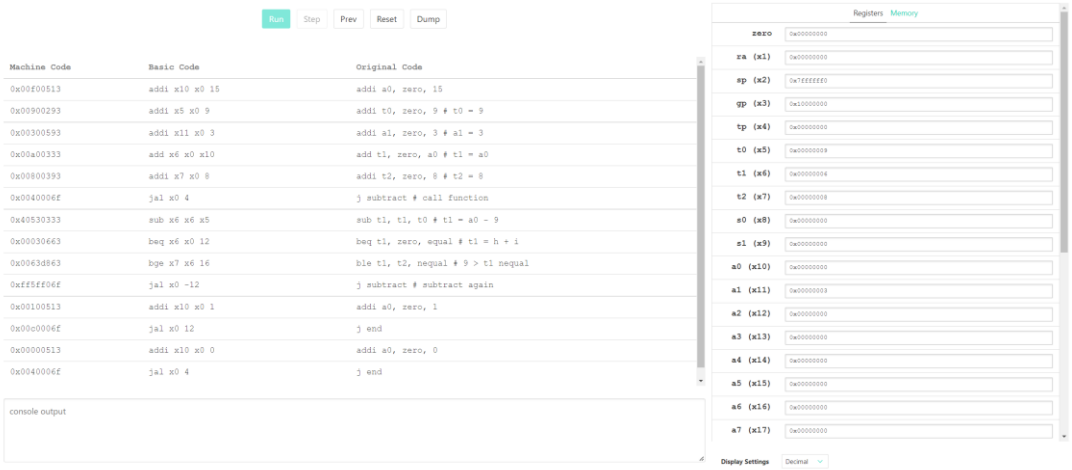
I spent 2h for this lab.

div9(15):

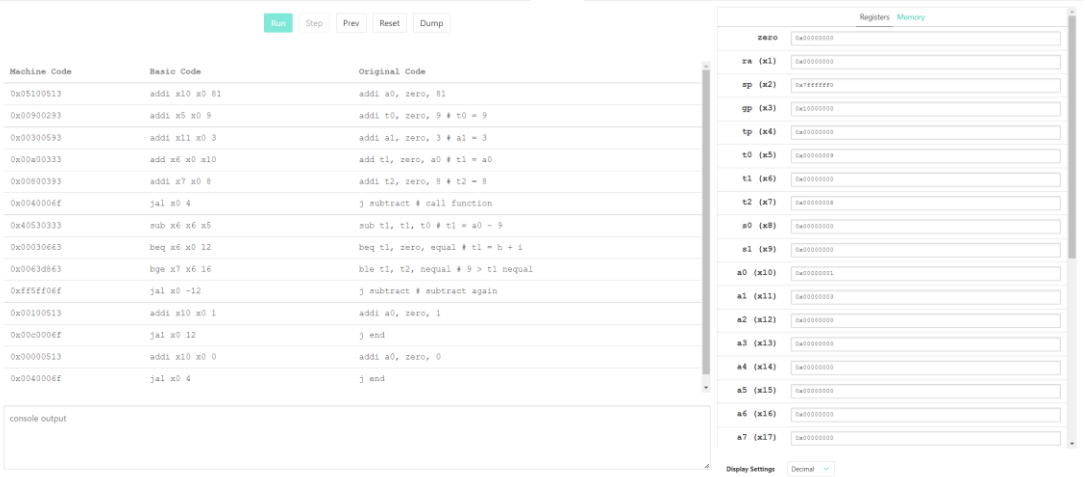
code:

```
main:
addi a0, zero, 15
addi t0, zero, 9  # t0 = 9
addi a1, zero, 3  # a1 = 3
add t1, zero, a0  # t1 = a0
addi t2, zero, 8  # t2 = 8
j subtract # call function

subtract:
sub  t1, t1, t0   # t1 = a0 - 9
beq  t1, zero, equal   # t1 = h + i
ble  t1, t2, nequal # 9 > t1 nequal
j subtract # subtract again

equal:
addi a0, zero, 1
j end

nequal:
addi a0, zero, 0
j end

end:
#Finished
```

div(15):



div(81):

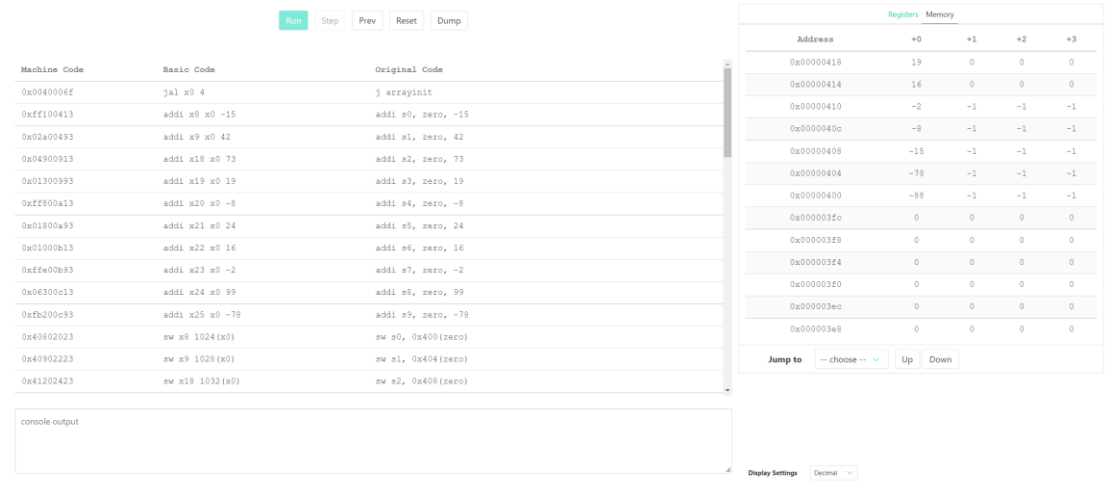


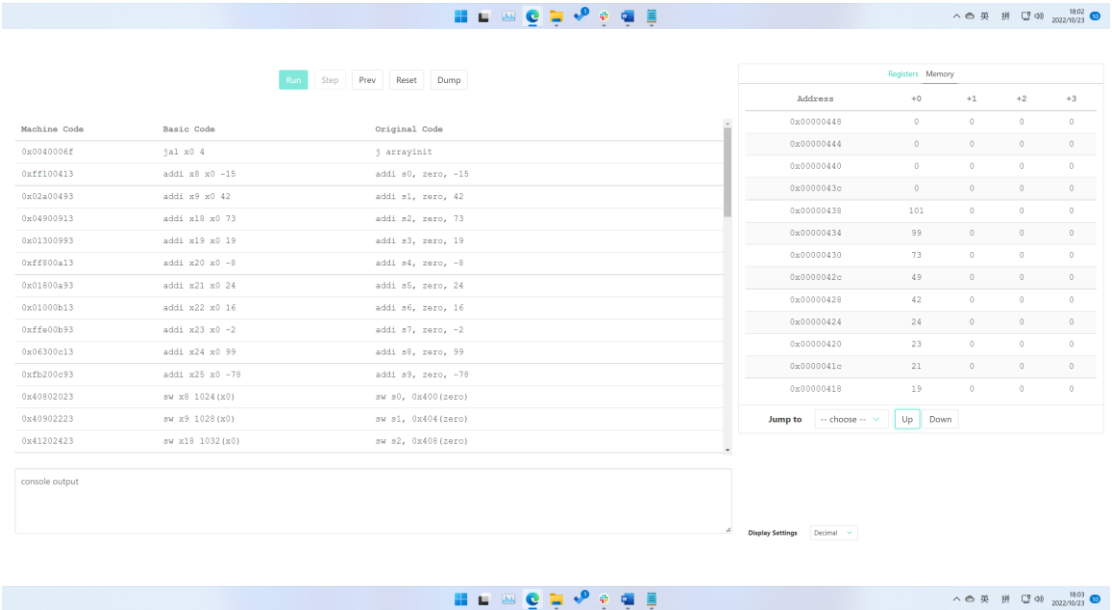
bubblesort:

code

```
main:
j arrayinit

arrayinit: #initialize array
addi s0, zero, -15
addi s1, zero, 42
addi s2, zero, 73
addi s3, zero, 19
addi s4, zero, -8
addi s5, zero, 24
addi s6, zero, 16
addi s7, zero, -2
addi s8, zero, 99
addi s9, zero, -78

sw s0, 0x400(zero)
sw s1, 0x404(zero)
sw s2, 0x408(zero)
sw s3, 0x40c(zero)
sw s4, 0x410(zero)
sw s5, 0x414(zero)
sw s6, 0x418(zero)
sw s7, 0x41c(zero)
sw s8, 0x420(zero)
sw s9, 0x424(zero)

addi s5, zero, 21
addi s6, zero, 23
addi s7, zero, -88
addi s8, zero, 49
addi s9, zero, 101

sw s5, 0x428(zero)
sw s6, 0x42c(zero)
sw s7, 0x430(zero)
sw s8, 0x434(zero)
sw s9, 0x438(zero)
```

memory:





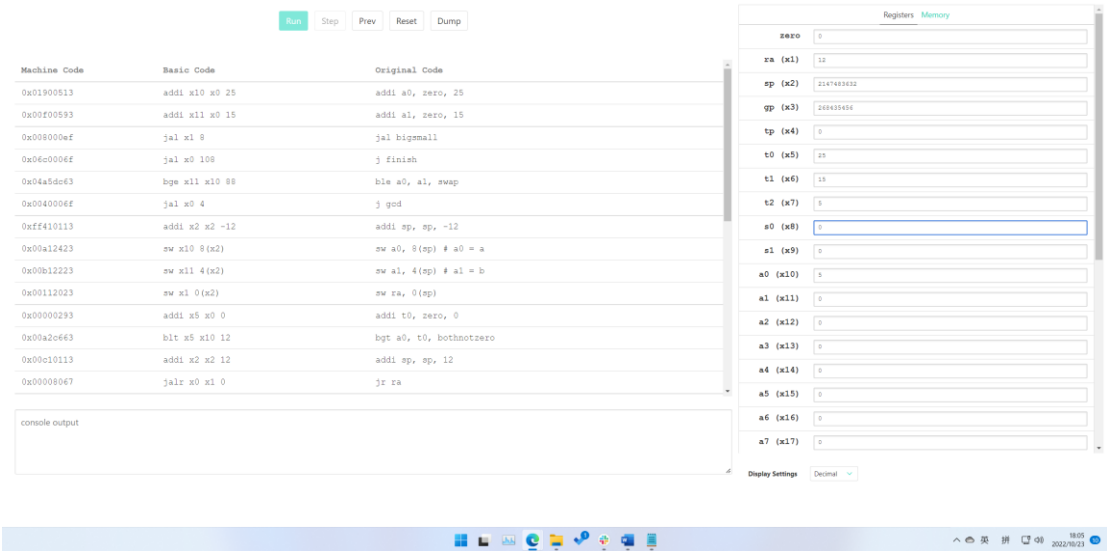
gcd:

code

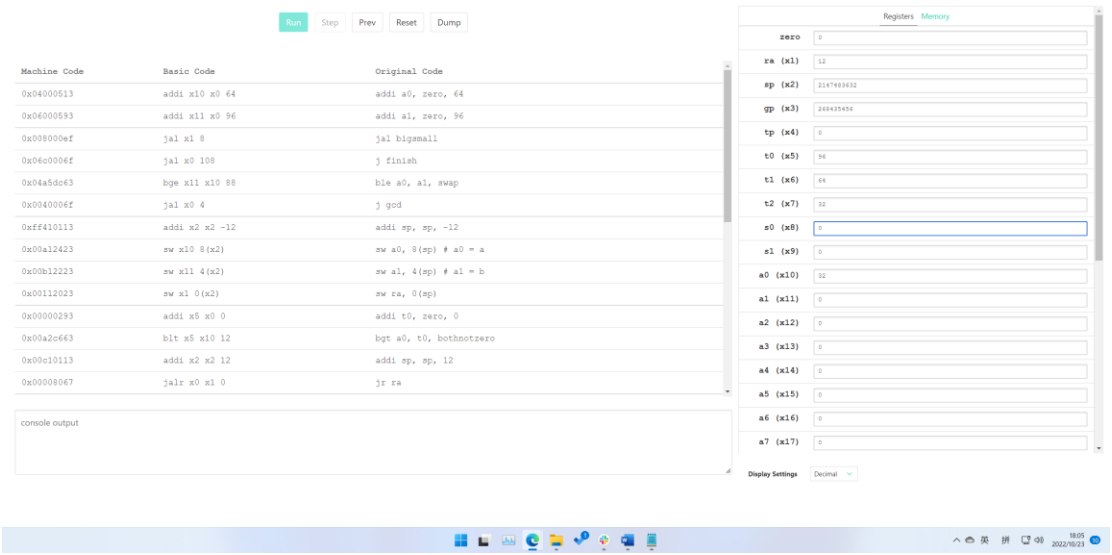
```
addi a0, zero, 25
addi a1, zero, 15

main:
jal bigsmall
j finish

bigsmall:
ble a0, a1, swap
j gcd

gcd:
addi sp, sp, -12
sw a0, 8(sp) # a0 = a
sw a1, 4(sp) # a1 = b
sw ra, 0(sp)
addi t0, zero, 0
bgt a0, t0, bothnotzero
addi sp, sp, 12
jr ra

bothnotzero:
bgt a1, t0, else # a0 or a1 is 0
addi sp, sp, 12
jr ra


else:
add t2, zero, a1
rem a1, a0, a1 # b = a mod b
add a0, zero, t2 # a = b
jal bigsmall
lw t0, 8(sp)
lw t1, 4(sp)
lw ra, 0(sp)
addi sp, sp, 12
jr ra
```

```
ble a0, a1, swap
j gcd

gcd:
addi sp, sp, -12
sw a0, 8(sp) # a0 = a
sw a1, 4(sp) # a1 = b
sw ra, 0(sp)
addi t0, zero, 0
bgt a0, t0, bothnotzero
addi sp, sp, 12
jr ra

bothnotzero:
bgt a1, t0, else # a0 or a1 is 0
addi sp, sp, 12
jr ra


else:
add t2, zero, a1
rem a1, a0, a1 # b = a mod b
add a0, zero, t2 # a = b
jal bigsmall
lw t0, 8(sp)
lw t1, 4(sp)
lw ra, 0(sp)
addi sp, sp, 12
jr ra

swap: # swap front and behind element
add t0, zero, a0
add a0, zero a1
add a1, zero t0
j gcd

finish:

```

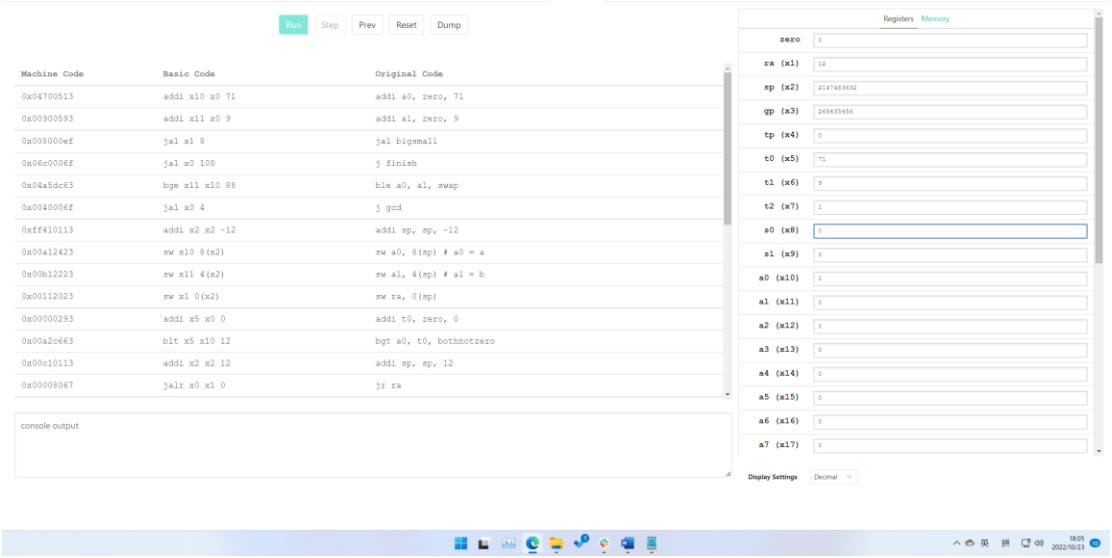
gcd(25, 15)



gcd(64,96) 96)

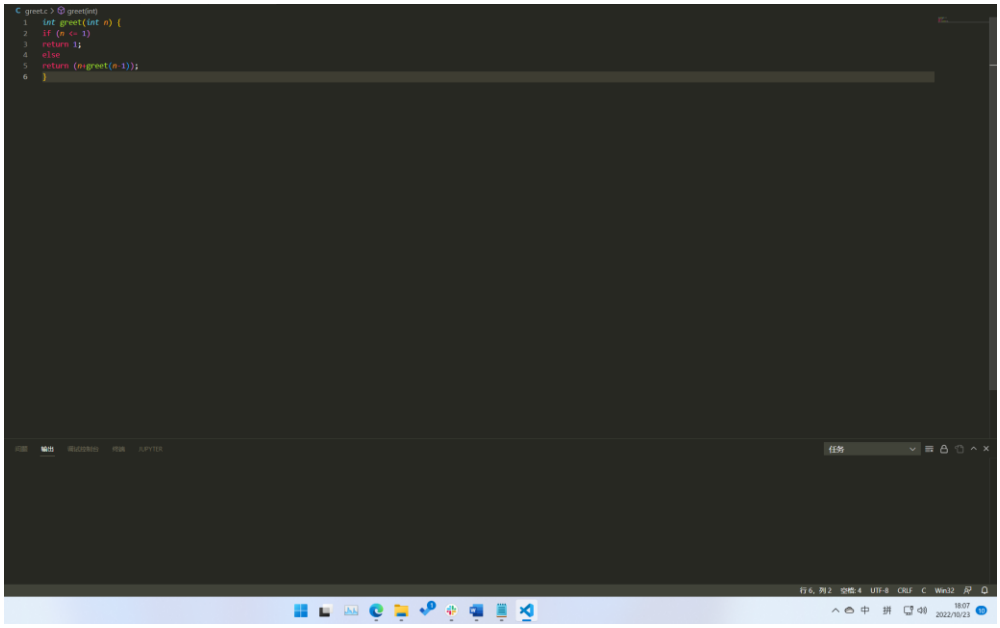


gcd(71, 9)



greet

C code

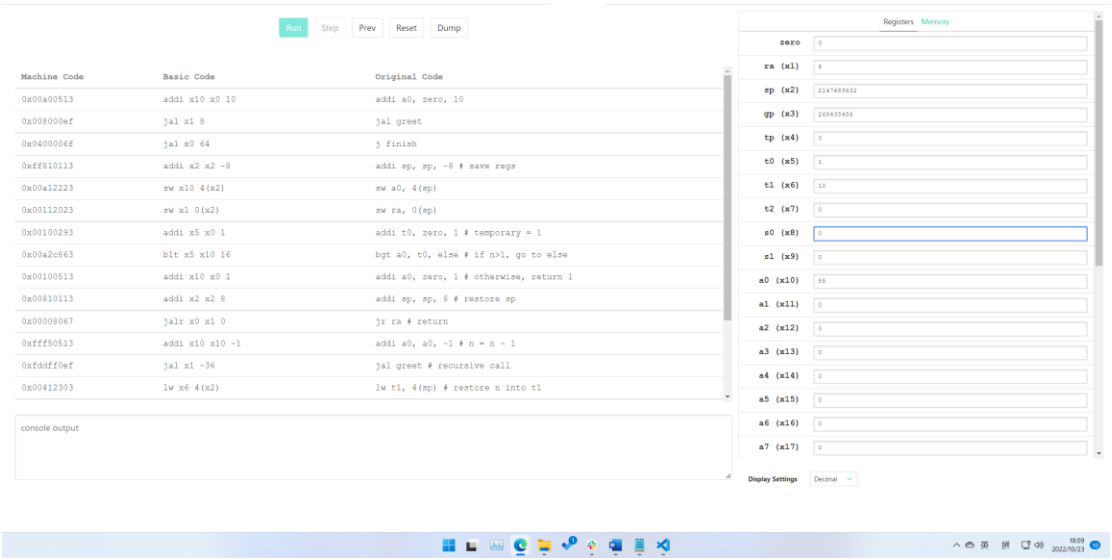


Assembly

```
addi a0, zero, 7

main:
jal greet
j finish

greet:
addi sp, sp, -8   # save regs
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, 1  # otherwise, return 1
addi sp, sp, 8    # restore sp
jr ra             # return

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

finish:
```

greet(7)

greet(10)


Run
Step
Prev
Reset
Dump
Machine Code
Basic Code
Original Code
0x00a00513 addi x10 x0 10 addi a0, zero, 10
0x008000ef jal x1 8 jal greet
0x0400006f jal x0 64 j finish
0xff810113 addi x2 x2 -8 addi sp, sp, -8 # save regs
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
0x00100513 addi x10 x0 1 addi a0, zero, 1 # otherwise, return 1
0x00810113 addi x2 x2 8 addi sp, sp, 8 # restore sp
0x00008067 jalr x0 x1 0 jr ra # return
0xfff50513 addi x10 x10 -1 addi a0, a0, -1 # n = n - 1
0xfddff0ef jal x1 -36 jal greet # recursive call
0x00412303 lw x6 4(x2) lw t1, 4(sp) # restore n into t1
console output
Registers
Memory
zero
ra (x1)
sp (x2)
gp (x3)
tp (x4)
t0 (x5)
t1 (x6)
t2 (x7)
s0 (x8)
s1 (x9)
a0 (x10)
a1 (x11)
a2 (x12)
a3 (x13)
a4 (x14)
a5 (x15)
a6 (x16)
a7 (x17)
Display Settings
Decimal