哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2022 春季

课程名称：计算机组成原理（实验）

实验名称：从 C 语言到机器码

实验性质：综合设计型

实验学时：2 地点：

学生班级：12 班

学生学号：200111205

学生姓名：李聪

作业成绩：

实验与创新实践教育中心制

2022 年 3 月

1、实验结果截图

（需贴出执行文件运行的结果截图）

```
comp2008@comp2008:~/riscv/riscv-tools$ riscv64-unknown-elf-gcc -E square.c -o square.i
comp2008@comp2008:~/riscv/riscv-tools$ riscv64-unknown-elf-gcc -S square.i -o square.s
comp2008@comp2008:~/riscv/riscv-tools$ riscv64-unknown-elf-gcc -c square.s -o square.o
comp2008@comp2008:~/riscv/riscv-tools$ riscv64-unknown-elf-gcc square.c -o square
comp2008@comp2008:~/riscv/riscv-tools$ spike pk square
bbl loader
res = 25
comp2008@comp2008:~/riscv/riscv-tools$
```

2、汇编代码注释（只需写主程序和子程序即可）

示例：

```
addi    sp,sp,-16
```

将堆栈指针寄存器 sp 与立即数(-16)相加，再存入堆栈指针寄存器 sp，即 sp = sp + (-16)

```
main:
```

```
    addi sp,sp,-64
```

将堆栈指针寄存器 sp 与立即数(-64)相加，再存入堆栈指针寄存器 sp,即 sp=sp+(-64)

```
    sd   ra,56(sp)
```

将寄存器 sp 的 8 个字节存入内存地址 R[sp]+56

```
    sd   s0,48(sp)
```

将寄存器 s0 的 8 个字节存入内存地址 R[sp]+48

```
    addi s0,sp,64
```

将堆栈指针寄存器 sp 存的值与立即数 64 相加，再存入寄存器 s0

```
    li    a5,5
```

将立即数 5 保存到寄存器 a5 中

```
sw   a5,-28(s0)
```

将寄存器 a5 的低 4 个字节存入内存地址 R[s0]-28

```
lui   a5,%hi(.LC0)
```

将.LC0 的高 20 位地址保存到寄存器 a5 的高 20 位，并清除寄存器 a5 的低 12 位。

```
addi a5,a5,%lo(.LC0)
```

将寄存器 a5 的值加上.LC0 的低 12 位地址后，保存到寄存器 a5

```
ld    a2,0(a5)
```

从地址为 R[a5]的内存中读取 8 个字节，存入寄存器 a2 中

```
ld    a3,8(a5)
```

从地址为 R[a5]+8 的内存中读取 8 个字节，存入寄存器 a3 中

```
ld    a4,16(a5)
```

从地址为 R[a5]+16 的内存中读取 8 个字节，存入寄存器 a4 中

```
ld    a5,24(a5)
```

从地址为 R[a5]+24 的内存中读取 8 个字节，存入寄存器 a5 中

```
sd    a2,-64(s0)
```

将寄存器 a2 所存的 8 个字节写到地址为 R[s0]-64 的内存中

```
sd    a3,-56(s0)
```

将寄存器 a3 所存的 8 个字节写到地址为 R[s0]-56 的内存中

```
sd    a4,-48(s0)
```

将寄存器 a4 所存的八个字节写到地址为 R[s0]-48 的内存中

```
    sd    a5,-40(s0)
```

将寄存器 a5 所存的 8 个字节写到地址为 R[s0]-40 的内存中

```
    sw    zero,-20(s0)
```

将寄存器 zero 的低位 4 个字节写到地址为 R[s0]-20 的内存中

```
    lw    a5,-28(s0)
```

从地址为 R[s0]-28 的内存中读取 4 个字节，进行有符号扩展后存入寄存器 a5 中

```
    slliw       a5,a5,8
```

把寄存器 a5 存的值左移 8 位，空出的位置填入 0，结果截为 32 为，进行有符号扩展后写入寄存器 a5

```
    sw    a5,-28(s0)
```

将寄存器 a5 的低位 4 个字节写到地址为 R[a5]-28 的内存中

```
    li    a5,7
```

将立即数 7 保存到寄存器 a5 中

```
    sw    a5,-24(s0)
```

将寄存器 a5 的地位 4 个字节到地址为 R[s0]-24 的内存中

```
    j     .L2
```

把 PC 设置为标签.L2 表示的地址，即跳转到标签.L2 所表示的指令

```
.L4:
```

```
    lw    a5,-24(s0)
```

从地址为 R[s0]-24 的内存中读取 4 个字节，进行有符号扩展后存入寄存器 a5 中

slli a5,a5,2

把寄存器 a5 存的值左移 2 位，空出的位置填入 0，存入寄存器 a5

addi a4,s0,-16

将立即数-16 进行符号位扩展后与寄存器 s0 存的值相加，存入寄存器 a4

add a5,a4,a5

将寄存器 a4 的值与寄存器 a5 的值相加后，存入寄存器 s5

lw a5,-48(a5)

从地址为 R[a5]-48 的内存中读取 4 个字节，进行有符号扩展后存入寄存器 a5 中

mv a4,a5

将寄存器 a5 的值复制到寄存器 a4 中

li a5,1

将立即数 1 存到寄存器 a5 中

bne a4,a5,.L3

如果寄存器 a4,a5 的值不相等，把 PC 设置为标签.L3 表示的地址，即跳转到标签.L3 所表示的指令

lw a4,-20(s0)

从地址为 R[s0]-20 的内存中读取 4 个字节，进行有符号扩展后存入寄存器 a4 中

lw a5,-28(s0)

从地址为 R[s0]-28 的内存中读取 4 个字节，进行有符号扩展后存入寄存器 a5 中

addw a5,a4,a5

将寄存器 a4,a5 的值相加，将结果截断为 32 位，把符号位扩展后的结果存到寄存器 a5

中

```
sw   a5,-20(s0)
```

将寄存器 a5 低 4 个字节写到地址为 R[s0]-20 的内存中

```
.L3:
```

```
lw   a5,-20(s0)
```

从地址为 R[s0]-20 的内存中读取 4 个字节，进行有符号扩展后存入寄存器 a5 中

```
sraiw    a5,a5,1
```

把寄存器 a5 的低 32 位右移 1 位，空位用 R[a5][31]填充，进行有符号扩展后写入寄存器 a5 中

```
sw   a5,-20(s0)
```

将寄存器 a5 的低 4 个字节写到地址为 R[s0]-20 的内存中

```
lw   a5,-24(s0)
```

从地址为 R[s0]-24 的内存中读取 4 个字节，进行符号位扩展后存入寄存器 a5 中

```
addiw    a5,a5,-1
```

将立即数-1 进行符号位扩展后加到寄存器 a5 所存的数中，将结果截断位 32 位，把符号位扩展的结果写入寄存器 a5

```
sw   a5,-24(s0)
```

将寄存器 a5 的低 4 个字节写到地址为 R[s0]-24 的内存中

```
.L2:
```

```
lw   a5,-24(s0)
```

从地址为 R[s0]-24 的内存中读取 4 个字节，进行有符号扩展后存到寄存器 a5 中

```
sext.w    a5,a5
```

读取寄存器 a5 的低 32 位，进行有符号扩展，结果写入寄存器 a5

```
bge  a5,zero,.L4
```

如果寄存器 a5 的值大于或等于寄存器 zero 的值，把 PC 设置为标签.L4 表示的地址，即跳转到标签.L4 所表示的指令

```
lw   a5,-20(s0)
```

从地址为 R[s0]-20 的内存中读取 4 个字节，进行有符号扩展后存入寄存器 a5 中

```
mv  a1,a5
```

将寄存器 a1 的值复制到寄存器 a5 中

```
lui   a5,%hi(.LC1)
```

将.LC1 的高 20 位地址保存的寄存器 a5，并清除寄存器 a5 的低 12 位

```
addi a0,a5,%lo(.LC1)
```

将寄存器 a5 的值加上.LC1 的低 12 位地址，保存到寄存器 a0 中

```
call  printf
```

调用 printf 函数，打印输出

```
li    a5,0
```

将立即数 0 保存到寄存器 a5 中

```
mv  a0,a5
```

将寄存器 a5 的值复制到寄存器 a0 中

```
ld    ra,56(sp)
```

从地址为 R[sp]+56 的内存中读取 8 个字节，保存到寄存器 ra 中

```
ld    s0,48(sp)
```

从地址为 R[sp]+48 的内存中读取 8 个字节，保存到寄存器 s0 中

```
addi sp,sp,64
```

将寄存器 sp 的值加上立即数 64 后保存到寄存器 sp 中

```
jr    ra
```

将 PC 设置为寄存器 ra 保存的值

3、机器码注释（只需写主程序和子程序即可）

示例：

1141：二进制为 0001 0001 0100 0001

fun3：000，imm：110000，rd/rs1：00010，op：01

c.addi 指令：sp = sp + (-16)

0000000000000000 <main>:

0: 7139 addi sp,sp,-64

7139：二进制为 0111 0001 0011 1001

fun3:011，imm:1111000000，rd/rs1:00010，op:01

c.addi16sp 指令：sp=sp+(-64)

2: fc06 sd ra,56(sp)

fc06：二进制为 1111 1100 0000 0110

fun3:111，imm:00111000，rs2:00001，op:10

c.fswsp 指令：M[R[sp]+56] = R[ra][63:0]

4: f822 sd s0,48(sp)

f822：二进制为 1111 1000 0010 0010

fun3:111，imm:00110000，rs2:01000，op:10

c.fswsp 指令：等价于 M[R[sp]+48] = R[s0][63:0]

6: 0080 addi s0,sp,64

0080：二进制为 0000 0000 1000 0000

fun3:000，imm:0001000000，rd’:000，op:00

c.addi4spn 指令：s0=sp+64

8: 4795 li a5,5

4795：二进制为 0100 0111 1001 0101

fun3:010，imm:000101，rd:01111，op:01

c.li 指令：a5=5

a: fef42223 sw a5,-28(s0)

fef42223：二进制为 1111 1110 1111 0100 0010 0010 0010 0011

imm:111111100100，rs2:0111，rs1:01000，fun3:010，opcode:0100011

sw 指令:M[R[s0]-28] = R[a5][31:0]

e: 000007b7 lui a5,0x0

000007b7：二进制为 0000 0000 0000 0000 0000 0111 1011 0111

imm:0000 0000 0000 0000 0000 0000 0000 0000，rd:01111，opcode:0110111

lui 指令：R[a5]=0

12: 00078793 mv a5,a5

00078793：二进制为 0000 0000 0000 0111 1000 0111 1001 0011

fun7:000000000000，rs1:01111，fun3:000，rd:01111，opcode:0010011

addi 指令：a5=a5+0

16: 6390 ld a2,0(a5)

6390：二进制为 0110 0011 1001 0000

fun3:011，uimm:00000000，(rs1':111，rd':100)，rs1:1111，rd:1100，op:00

c.ld 指令：R[a2] = M[R[a5]+0][63:0]

18: 6794 ld a3,8(a5)

6794：二进制为 0110 0111 1001 0100

fun3:011，uimm:00001000，(rs1':111,rd'101)，rs1:1111，rd:1101，op:00

c.ld 指令：R[a3] = M[R[a5]+8][63:0]

1a: 6b98 ld a4,16(a5)

6b98：二进制为 0110 1011 1001 1000

fun3:011，uimm:00010000，(rs1':111,rd':110)，rs1:1111，rd:1110，op:00

c.ld 指令：R[a4] = M[R[a5]+16][63:0]

1c: 6f9c ld a5,24(a5)

6f9c：二进制为 0110 1111 1001 1100

fun3:011，uimm:00011000，(rs':111,rd':111)，rs1:1111，rd:1111，op:00

c.ld 指令：R[a5] = M[R[a5]+24][63:0]

1e: fcc43023 sd a2,-64(s0)

fcc43023：二进制为 1111 1100 1100 0100 0011 0000 0010 0011

imm:111111000000，rs2:01100，rs1:01000，fumct3:011，opcode:0100011

sd 指令：M[R[s0]+(-64)] = R[a2][63:0]

22: fcd43423 sd a3,-56(s0)

fcd43423：二进制为 1111 1100 1101 0100 0011 0100 0010 0011

imm:111111001000，rs2:01101，rs1:01000，funct3:011，opcode:0100011

sd 指令：M[R[s0]+(-56)] = R[a3][63:0]

26: fce43823 sd a4,-48(s0)

fce43823：二进制为 1111 1100 1110 0100 0011 1000 0010 0011

imm:111111010000，rs2:01110，rs1:01000，funct3:011，opcode:0100011

sd 指令：M[R[s0]+(-48)] = R[a4][63:0]

2a: fcf43c23 sd a5,-40(s0)

fcf43c23：二进制为 1111 1100 1111 0100 0011 1100 0010 0011

imm:111111011000，rs2:01111，rs1:01000，funct3:011，opcode:0100011

sd 指令：M[R[s0]-40] = R[a5][63:0]

2e: fe042623 sw zero,-20(s0)

fe042623：二进制为 1111 1110 0000 0100 0010 0110 0010 0011

imm:111111101100，rs2:00000，rs1:01000，funct3:110，opcode:0100011

sw 指令：M[R[s0]+(-20)] = R[zero][31:0]

32: fe442783 lw a5,-28(s0)

fe442783：二进制为 1111 1110 0100 0100 0010 0111 1000 0011

imm:111111100100，rs1:01000，funct3:010，rd:01111，opcode:0000011

lw 指令：R[a5] = M[R[s0]+(-28)]

36: 0087979b slliw a5,a5,0x8

0087979b：二进制为 0000 0000 1000 0111 1001 0111 1001 1011

imm:000000001000，rs1:01111，funct3:001，rs2:01111，opcode:0011011

slliw 指令：R[a5] = sext((R[a5] << 8)[31:0])

3a: fef42223 sw a5,-28(s0)

fef42223：二进制为 1111 1110 1111 0100 0010 0010 0010 0011

imm:111111100100，rs2:01111，rs1:01000，funct3:010，opcode:0100011

sw 指令：M[R[s0]+(-28)] = R[a5][31:0]

3e: 479d li a5,7

479d：二进制为 0100 0111 1001 1101

fun3:010，imm:000111，rd:01111，op:01

c.ld 指令：R[a5] = 7

40: fef42423 sw a5,-24(s0)

fef42423：二进制为 1111 1110 1111 0100 0010 0100 0010 0011

imm:111111101000，rs2:01111，rs1:01000，funct3:010，opcoed:0100011

sw 指令：M[R[s0]+(-24)] = R[a5][31:0]

44: a83d j 82 <.L2>

a83d：二进制为 1010 1000 0011 1101

fun3:101，imm:000000111110

c.j 指令：PC = 0x0000000000000082

0000000000000046 <.L4>:

46: fe842783 lw a5,-24(s0)

fe842783：二进制为 1111 1110 1000 0100 0010 0111 1000 0011

imm:111111101000，rs1:01000，funct3:010，rd:01111，opcode:0000011

lw 指令：R[a5] = M[R[s0]+(-24)][31:0]

4a: 078a slli a5,a5,0x2

078a：二进制为 0000 0111 1000 1010

fun3:000，uimm:000010，op:10

c.slli 指令：R[a5] = R[a5] << 2

4c: ff040713 addi a4,s0,-16

ff040713：二进制为 1111 1111 0000 0100 0000 0111 0001 0011

imm:111111110000，rs1:01000，funct3:000，rd:01110，opcode:0010011

addi 指令：R[a4] = R[s0] + (-16)

50: 97ba add a5,a5,a4

97ba：二进制为 1001 0111 1011 1010

fun3:100，rd:01111，rs2:01110，op:10

c.add 指令：R[a5] = R[a5] + R[a4]

52: fd07a783 lw a5,-48(a5) # ffffffffffffffd0 <.L2+0xffffffffffffff4e>

fd07a783：二进制为 1111 1101 0000 0111 1010 0111 1000 0011

imm:111111010000，rs1:01111，funct3:010，rs2:01111，opcode:0000011

lw 指令：R[a5] = M[R[a5]+(-48)][31:0]

56: 873e mv a4,a5

873e：二进制为 1000 0111 0011 1110

fun3:100，rd:01110，rs2:01111，op:10

c.mv 指令：R[a4] = R[a5]

58: 4785 li a5,1

4785：二进制为 0100 0111 1000 0101

fun3:010，imm:000001，rd:01111，op:01

c.li 指令：R[a5] = 1

5a: 00f71963 bne a4,a5,6c <.L3>

00f71963：二进制为 0000 0000 1111 0111 0001 1001 0110 0011

imm:0000000010010，rs2:01111，rs1:01110，funct3:001，opcode:1100011

bne 指令：if(R[a4] != R[a5]) PC = 0x000000000000006c

5e: fec42703 lw a4,-20(s0)

fec42703：二进制为 1111 1110 1100 0100 0010 0111 0000 0011

imm:111111101100，rs1:01000，funct3:010，rd:01110，opcode:0000011

lw 指令：R[a4] = M[R[s0]+(-20)][31:0]

62: fe442783 lw a5,-28(s0)

fe442783：二进制为 1111 1110 0100 0100 0010 0111 1000 0011

imm:111111100100，rs1:01000，funct3:010，rs2：01111，opcode:0000011

lw 指令：R[a5] = M[R[s0]+(-28)][31:0]

66: 9fb9 addw a5,a5,a4

9fb9：二进制为 1001 1111 1011 1001

fun3:100，rd:11111，rs2:01110，op:10

c.add 指令：R[a5] = R[a5] + R[a4]

68: fef42623 sw a5,-20(s0)

fef42623：二进制为 1111 1110 1111 0100 0010 0110 0010 0011

imm:111111101100，rs2:0111，rs1:01000，funct3:010，opcode:0100011

sw 指令：M[R[s0]+(-20)] = R[a5][31:0]

000000000000006c <.L3>:

6c: fec42783 lw a5,-20(s0)

fec42783：二进制为 1111 1110 1100 0100 0010 0111 1000 0011

imm:111111101100，rs1:01000，funct3:010，rd:01111，opcode:0000011

lw 指令：R[a5] = M[R[s0]+(-20)][31:0]

70: 4017d79b sraiw a5,a5,0x1

4017d79b：二进制为 0100 0000 0001 0111 1101 0111 1001 1011

imm:010000000001，rs1:01111，funct3:101，rd:01111，opcode:0011011

sraiw 指令：R[a5] = sext(R[a5][31:0]>>1)

74: fef42623 sw a5,-20(s0)

fef42623：二进制为 1111 1110 1111 0100 0010 0110 0010 0011

imm:111111101100，rs2:01111，rs1:01000，funct3:010，opcode:0100011

sw 指令：M[R[s0]+(-20)] = R[a5][31:0]

78: fe842783 lw a5,-24(s0)

fe842783：二进制为 1111 1110 1000 0100 0010 0111 1000 0011

imm:111111101000，rs1:01000，funct3:010，rd:01111，opcode:0000011

lw 指令：R[a5] = M[R[s0]+(-24)][31:0]

7c: 37fd addiw a5,a5,-1

37fd：二进制为 0011 0111 1111 1101

fun3:001，imm:1111111，rd:01111，op:01

c.addiw 指令：R[a5] = sext((R[a5]+(-1))[31:0])

7e: fef42423 sw a5,-24(s0)

fef42423：二进制为 1111 1110 1111 0100 0010 0100 0010 0011

imm:111111101000，rs2:01111，rs1:01000，funct3:010，opcode:0100011

sw 指令：M[R[s0]+(-24)] = R[a5][31:0]

0000000000000082 <.L2>:

82: fe842783 lw a5,-24(s0)

fe842783：二进制为 1111 1110 1000 0100 0010 0111 1000 0011

imm:111111101000，rs1:01000，funct3:010，rd:01111，opcode:0000011

lw 指令：R[a5] = M[R[s0]+(-20)][31:0]

86: 2781 sext.w a5,a5

2781：二进制为 0010 0111 1000 0001

fun3:001，imm:000000，rd:01111，op:01

c.addiw 指令：R[a5] = sext((R[a5]+0)[31:0])

88: fa07dfe3 bgez a5,46 <.L4>

fa07dfe3：二进制为 1111 1010 0000 0111 1101 1111 1110 0011

imm:111111011111，rs2:00000，rs1:01111，funct3:101，opcode:1100011

bge 指令：if(R[a5] >= 0) PC = 0x0000000000000046

8c: fec42783 lw a5,-20(s0)

fec42783：二进制为 1111 1110 1100 0100 0010 0111 1000 0011

imm:111111101100，rs1:01000，funct3:010，rd:01111，opcode:0000011

lw 指令：R[a5] = M[R[s0]+(-20)][31:0]

90: 85be mv a1,a5

85be：二进制为 1000 0101 1011 1110

fun3:100，rd:01011，rs2:01111，op:10

c.mv 指令：R[a1] = R[a5]

92: 000007b7 lui a5,0x0

000007b7：二进制为 0000 0000 0000 0000 0000 0111 1011 0111

imm:00000000000000000000，rd:01111，opcode:0110111

lui 指令：R[a5] = sext(00000000000000000000<<12)

96: 00078513 mv a0,a5

00078513：二进制为 0000 0000 0000 0111 1000 0101 0001 0011

imm:000000000000，rs1:01111，funct3:000，rd:01010，opcode:0010011

addi 指令：R[a0] = R[a5] + 0

9a: 00000097 auipc ra,0x0

00000097，二进制为 0000 0000 0000 0000 0000 0000 1001 0111

imm:00000000000000000000，rd:00001，opcode:0010111

auipc 指令：R[ra] = PC+sext(00000000000000000000<<12)

9e: 000080e7 jalr ra # 9a <.L2+0x18>

000080e7，二进制为 0000 0000 0000 0000 1000 0000 1110 0111

imm:000000000000，rs1:00001，funct3:000，rd:00001，opcode:1100111

jalr 指令：PC=R[ra]

a2: 4781 li a5,0

4781：二进制为 0100 0111 1000 0001

fun3:010，imm:000000，rd:01111，op:01

c.li 指令：R[a5] = 0

a4: 853e mv a0,a5

853e：二进制为 1000 0101 0011 1110

fun3:100，rd1:01010，rs2:01111，op:10

R[a0] = R[a5]

a6: 70e2 ld ra,56(sp)

70e2：二进制为 0111 0000 1110 0010

fun3:011，uimm:000111000，rd:00001，op:10

c.ldsp 指令：R[ra] = M[R[sp]+56][63:0]

a8: 7442 ld s0,48(sp)

7442：二进制为 0111 0100 0100 0010

fun3:011，uimm:000110000，rd:01000，op:10

c.ldsp 指令：R[s0] = M[R[sp]+48][63:0]

aa: 6121 addi sp,sp,64

6121：二进制为 0110 0001 0010 0001

fun3:011，imm:0001000000，op:01

c.addi16sp 指令：R[sp] = R[sp]+64

ac: 8082 ret

8082 二进制为：1000 0000 1000 0010

fun3:100，rs1:00001，op:10

c.jr 指令：PC = R[ra]