2024\13-codegen-riscv\riscv-in-class-2023\add.asm

```
li t0, 20       # li: load immediate
li t1, 22
add t2, t0, t1
```

2024\13-codegen-riscv\riscv-in-class-2023\addi.asm

```
li t0, 100
addi t0, t0, 20 # addi: add immediate
addi t0, t0, -20
```

2024\13-codegen-riscv\riscv-in-class-2023\sub-add.asm

```
# f = (g + h) - (i + j)
# t6 = (t0 + t1) - (t3 + t4)

li t0, 0
li t1, 10
add t2, t0, t1

li t3, 30
li t4, 40
add t5, t3, t4

sub t6, t2, t5
```

2024\13-codegen-riscv\riscv-in-class-2023\ecall.asm

```
# f = (g + h) - (i + j)
# t6 = (t0 + t1) - (t3 + t4)

li t0, 0
li t1, 10
add t2, t0, t1

li t3, 30
li t4, 40
add t5, t3, t4

sub t6, t2, t5

li  a7, 1
mv a0, t6    # add a0, t6, zero
ecall
```

2024\13-codegen-riscv\riscv-in-class-2023\data.asm

```
# f = (g + h) - (i + j)
# t6 = (t0 + t1) - (t3 + t4)

.data
g: .word 0
h: .word 10
i: .word 30
j: .word 40

result: .word 0

msg: .string "The result is :"  # .ascii

.text
la t0, g                # la: load address
lw t0, 0(t0)    # lw: load word

la t1, h
lw t1, 0(t1)
add t2, t0, t1

la t3, i
lw t3, 0(t3)
la t4, j
lw t4, 0(t4)
add t5, t3, t4

sub t6, t2, t5

la t0, result
sw t6, 0(t0)    # sw: store word

li  a7, 4
la a0, msg
ecall

li  a7, 1
mv a0, t6    # add a0, t6, zero
ecall
```

2024\13-codegen-riscv\riscv-in-class-2023\branch-max.asm

```
# c = max(a, b)

.data
a: .word 100
b: .word 200
c: .word 0

.text
lw t0, a
lw t1, b

bge t0, t1, greater_equal
mv t2, t1
j end   # j: jump

greater_equal:
mv t2, t0

end:
sw t2, c, t3
```

2024\13-codegen-riscv\riscv-in-class-2023\array.asm

```
.data
numbers: .word -30, 30, -20, 20, -10, 10, 0

.text
la t0, numbers
lw t1, 12(t0)
addi t1, t1, 80
sw t1, 12(t0)
```

2024\13-codegen-riscv\riscv-in-class-2023\array-for.asm

```
.data
numbers: .word -30, 30, -20, 20, -10, 10, 0
size: .word 7

positive_sum: .word 0
negative_sum: .word 0

.text
la t0, numbers          # t0: the address of the array
lw t1, size                     # t1: size = 7

mv t2, zero                     # counter, initially 0

li t3, 0                                # t3: sum of positive numbers <- 0
li t4, 0                                # t4: sum of negative numbers <- 0

loop:
        bge t2, t1, end_loop
        # numbers[t2]
        # mul t5, t2, 4
        slli t5, t2, 2          # slli: shift left logical immediate
        add t5, t0, t5
        lw t5, 0(t5)

        addi t2, t2, 1

        bltz     t5, negative                           # bltz: branch if less than zero
        add t3, t3, t5
        j loop

negative:
        add t4, t4, t5
        j loop

end_loop:
        sw, t3, positive_sum, t5
        sw, t4, negative_sum, t5
```

2024\13-codegen-riscv\riscv-in-class-2023\proc-max.asm

```
# proc-max.asm

.data
max_result: .word 0

.text
.global main

max:
# a0 (argument 0), a1
blt a0, a1, smaller
j end_max

smaller:
mv a0, a1

end_max:

ret
# jr ra # jr: jump register
# jalr zero 0(ra)       # jalr: jump and link register
# jal: jump and link

############### main ###############
.data
a: .word 100
b: .word 200

.text
main:
lw a0, a
lw a1, b

call max
# jal max
# jal ra, max           # jal: jump and link            ra: return address register
# TODO
sw a0, max_result, t0
```

2024\13-codegen-riscv\riscv-in-class-2023\proc-fact.asm

```
.text
.global main

factorial:
beqz a0, base_case

addi sp, sp, -8
sw a0, 4(sp)
sw ra, 0(sp)

# n > 0: n * factorial(n - 1)
addi a0, a0, -1         # a0: n - 1
call factorial          # a0: factorial(n - 1)
mv t0, a0                       # t0: factorial(n - 1)

lw a0, 4(sp)                    # a0: n
lw ra, 0(sp)
addi sp, sp, 8

mul a0, a0, t0          # a0: n * factorial(n - 1)
j end

base_case:
        li a0, 1

end:
        ret


######### main ########
.data
n: .word 10

.text
main:
lw a0, n
call factorial
```