NUST CHIP DESIGN CENTRE

Digital Design Verification

EXERCISE # 04
RISCV Assembly Tasks

Name	*Khalil Rehman*
Instructor	*Sir Umer*
Date	*22th July 2025*

1. Tasks:

Task # 1

```
.data

.text
.globl main

main:
    li a0, -6       # first number
    li a1, 4        # second number
    jal ra, multiply

    # Exit properly
    li a7, 10
    ecall

multiply:
    add t0, a0, x0      # t0 = a
    add t1, a1, x0      # t1 = b
    addi a2, x0, 0      # result = 0
    addi t2, x0, 0      # negative flag = 0

    # Check if a is negative
    slt t3, t0, x0      # t3 = 1 if t0 < 0
    beq t3, x0, check_b
    sub t0, x0, t0      # a = -a
    xori t2, t2, 1      # toggle negative

check_b:
    # Check if b is negative
    slt t3, t1, x0      # t3 = 1 if t1 < 0
    beq t3, x0, loop_mul
    sub t1, x0, t1      # b = -b
    xori t2, t2, 1      # toggle negative

loop_mul:
    beq t1, x0, done_mul  # if b == 0, exit loop
    add a2, a2, t0      # result += a
    addi t1, t1, -1     # b--
    j loop_mul

done_mul:
    beq t2, x0, end_mul
    sub a2, x0, a2      # result = -result

end_mul:
    jr ra               # return to caller
```

Execution:

```
-24
```

a0 (x10)	-6
a1 (x11)	4
t0 (x5)	-6
t1 (x6)	4
ra (x1)	12
a2 (x12)	-24

Task # 2

```
.data
msg_div: .asciiz "Division result done\n"

.text
.globl main

main:
    li a0, -17      # dividend
    li a1, 5        # divisor

    jal ra, divide   # call divide(a0, a1)

    # a2 = quotient
    # a3 = remainder
    li a7, 4         # syscall: print string
    la a0, msg_div
    ecall

    # Exit program
    li a7, 10
    ecall

divide:
    add t0, a0, x0      # dividend
    add t1, a1, x0      # divisor
    addi a2, x0, 0      # quotient = 0
    addi a3, x0, 0      # remainder = 0
    addi t2, x0, 0      # negative flag = 0

    # if (a0 < 0) → make positive
    slt t3, t0, x0
    beq t3, x0, check_div_b
    sub t0, x0, t0
    xori t2, t2, 1

check_div_b:
    slt t3, t1, x0
    beq t3, x0, div_loop
    sub t1, x0, t1
    xori t2, t2, 1

div_loop:
    slt t3, t0, t1      # if (t0 < t1) break
```

```
42 div_loop:
43     slt t3, t0, t1       # if (t0 < t1) break
44     bne t3, x0, div_done
45     sub t0, t0, t1       # dividend -= divisor
46     addi a2, a2, 1       # quotient++
47     j div_loop
48
49 div_done:
50     add a3, t0, x0       # remainder = leftover dividend
51     beq t2, x0, div_end
52     sub a2, x0, a2       # apply sign to quotient
53
54 div_end:
55     jr ra
```

Execution: