NUST CHIP DESIGN CENTRE

Digital Design Verification

EXERCISE # 01
RISCV Assembly Tasks

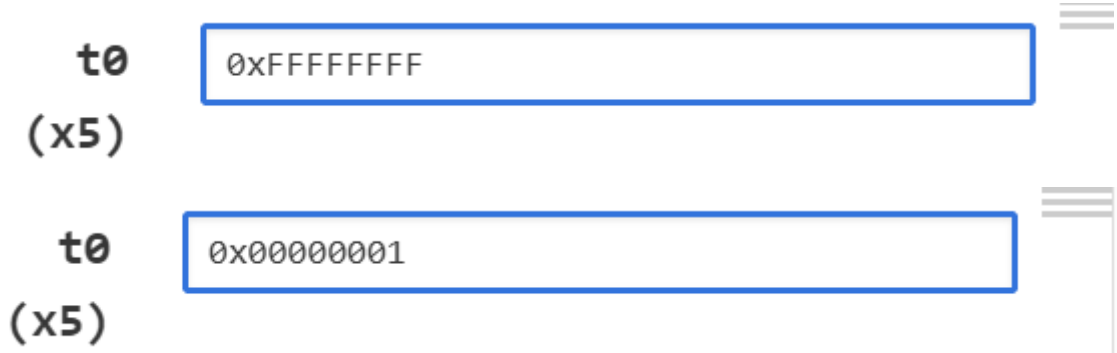
Name	*Khalil Rehman*
Instructor	*Dr Imran*
Date	*15th July 2025*

1. Tasks:

Task # 1

```
#Task 1
addi x5, x0, -1    # x6 = -1 (which is 0xFFFFFFFF in 32-bit two's complement)
sub x5, x6, x5     # x5 = -1 - x5 (flips all bits)
```

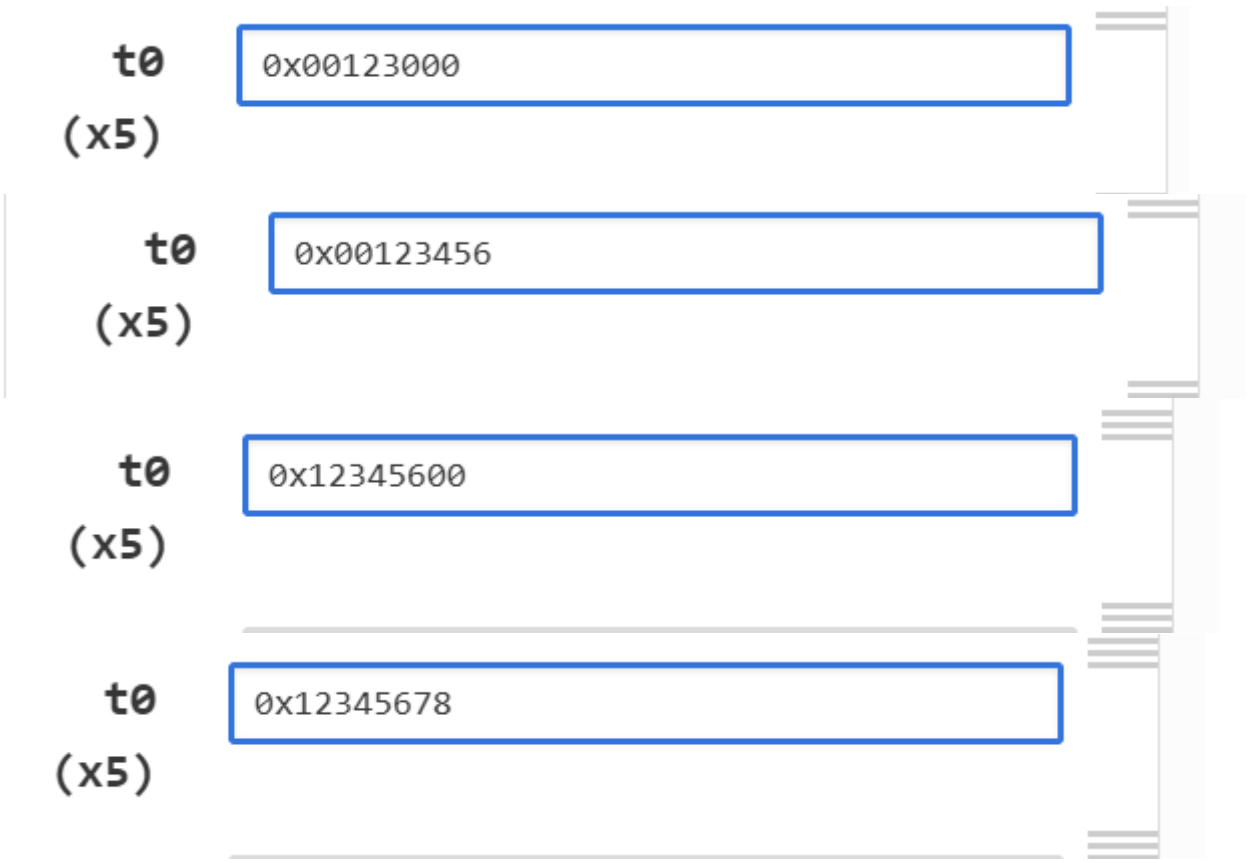
Execution:



Task # 2

```
#Task 2
addi x5,x0, 0x123
slli x5,x5, 12 #12 bits shift because immediate stores only 12 bits
addi x5,x5, 0x456
slli x5,x5, 8
addi x5,x5, 0x78
```

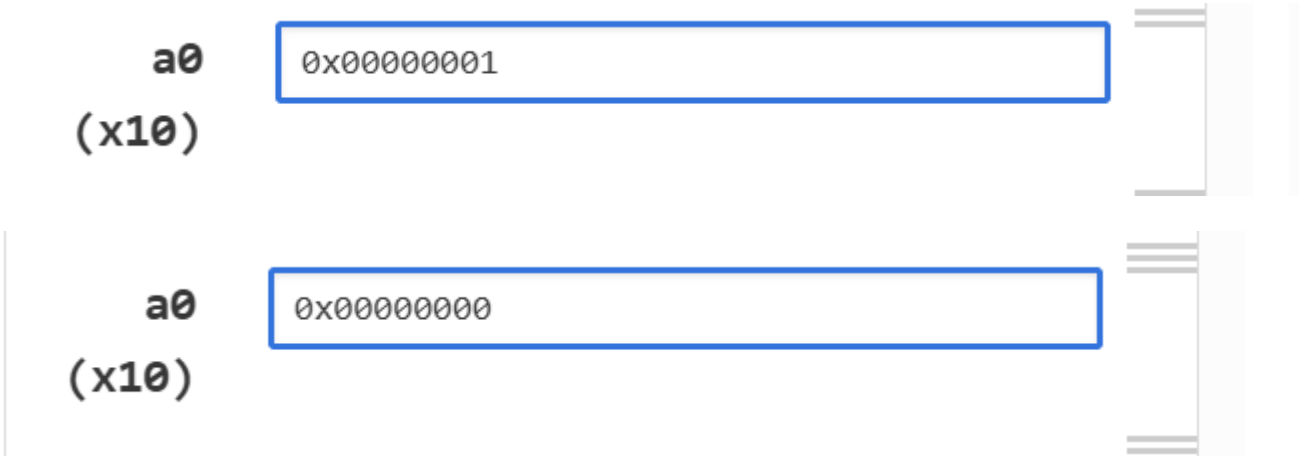
Execution:



Task #3

```
13 #Task 3
14 srli x10, x5, 1    # Shift right by 1
15 slli x10, x10, 1    # Shift left by 1
16 xor x10, x5, x10    # Compare values with shifted
17
18 # andi x10, x5, 1 #easy way
```

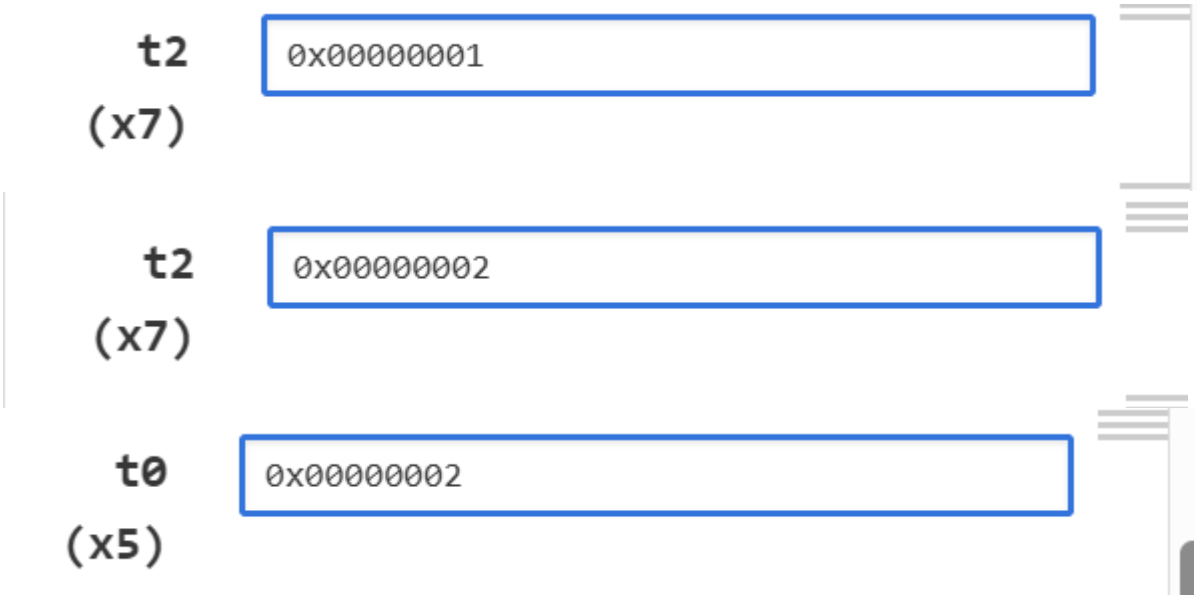
Execution:



Task # 4

```
21 #Task 4
22 addi x7, x0, 1     # x7 = 1
23 sll x7, x7, x10    # x7 = 1 << bit_position
24 xor x5, x5, x7     # Toggle the bit
```

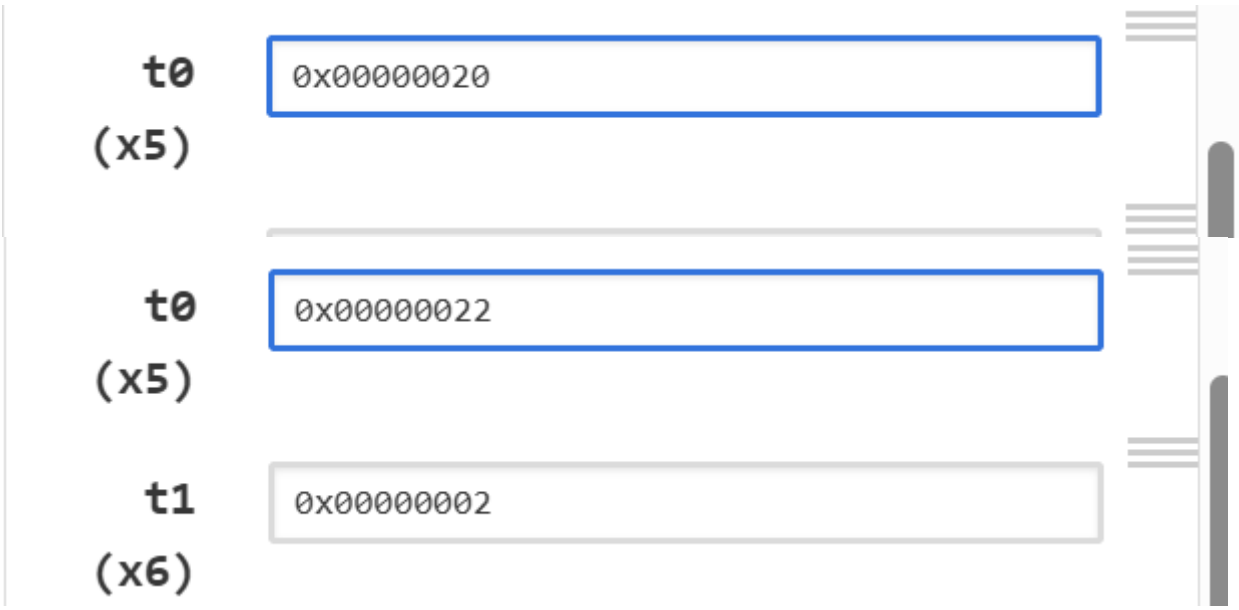
Execution:

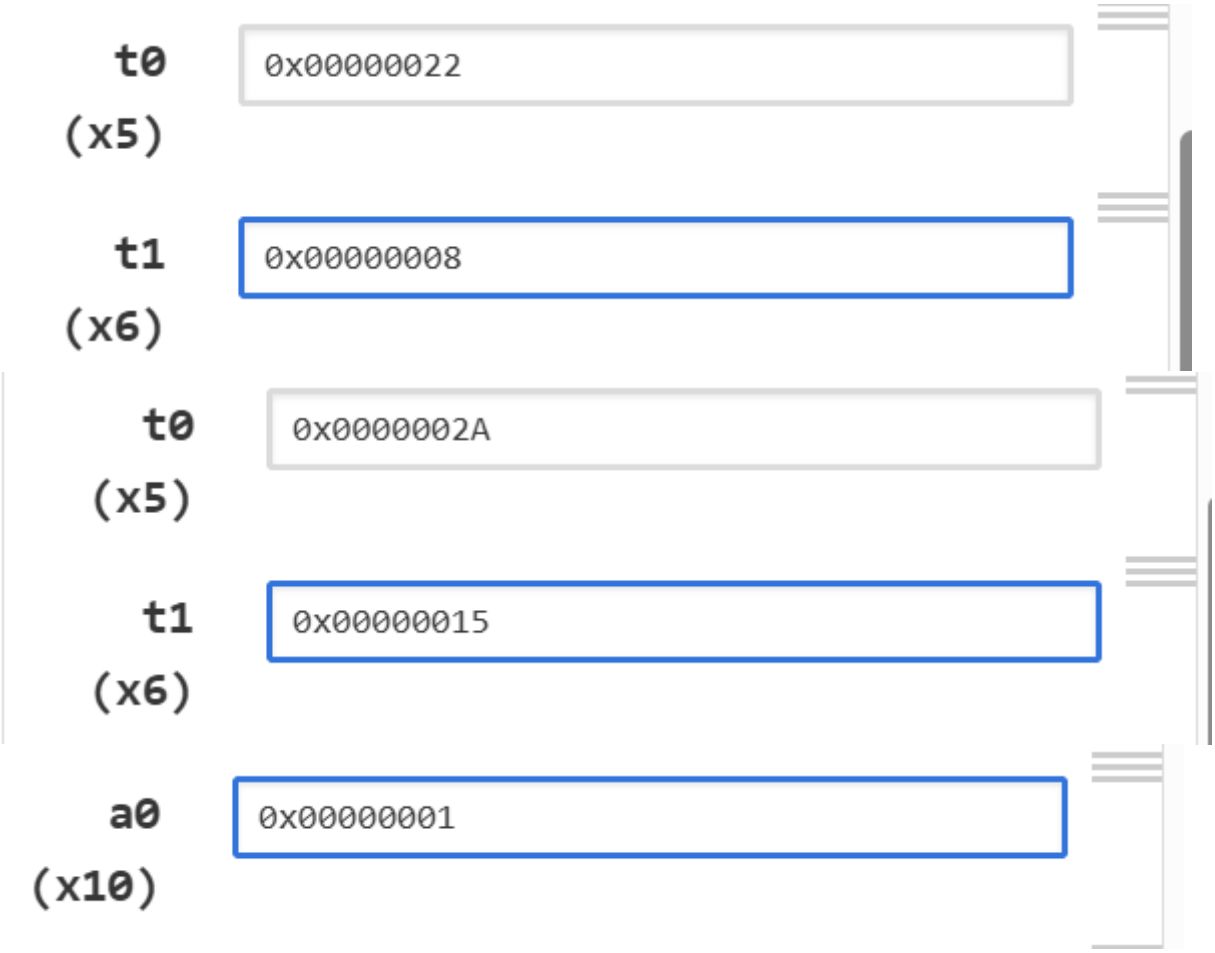


Task # 5

```
27 #Task 5
28 # Parity check for x5, result in x10 (0=even, 1=odd)
29 addi x5, x4, 32
30 srli x6, x5, 16     # Shift right 16 bits
31 xor x5, x5, x6      # XOR upper and lower
32 srli x6, x5, 8      # Shift right 8 bits
33 xor x5, x5, x6      # XOR again
34 srli x6, x5, 4      # Shift right 4 bits
35 xor x5, x5, x6      # XOR again
36 srli x6, x5, 2      # Shift right 2 bits
37 xor x5, x5, x6      # XOR again
38 srli x6, x5, 1      # Shift right 1 bit
39 xor x5, x5, x6      # Final XOR
40 andi x10, x5, 1     # LSB (parity bit)
```

Execuion:





Task # 6

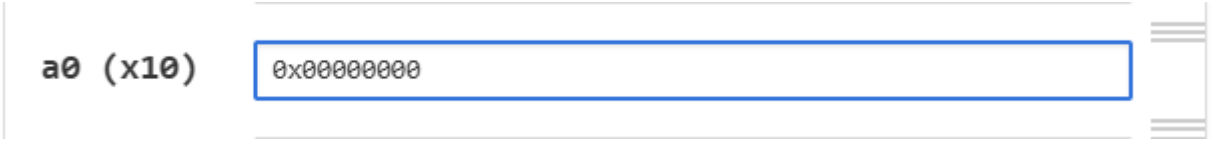
```
#Task 6
# x5 contains the 32-bit value
# x6 contains byte position (0-3)
andi x6, x6, 0x3
# Calculate shift amount (bytes to bits)
slli x7, x6, 3      # x7 = x6 * 8
# Extract the byte
srl x10, x5, x7     # Shift desired byte to LSB position
andi x10, x10, 0xFF # Zero-extend
```

Execution:

Task # 7

```
54 #Task 7
55 andi x6, x6, 0x3     # x6 = x6 % 4
56
57 # Calculate shift amount (byte position * 8)
58 slli x7, x6, 3       # x7 = x6 * 8
59
60 # Extract the byte and move to LSB position
61 srl x10, x5, x7      # Shift desired byte to bit position 0
62 andi x10, x10, 0xFF  # Isolate the byte (bits 7-0)
63
64 # Sign extend the byte
65 slli x10, x10, 24    # Move byte to MSB position
66 srai x10, x10, 24    # Arithmetic shift right to sign extend
67
68
```

Execution:



Task # 8

```
# Task 8: Signed comparison (x5 > x6) without slt
# Input: x5, x6
# Output: x10 = 1 if x5 > x6 (signed), else 0

# Get signs (arithmetic shift right 31 bits)
srai x7, x5, 31      # x7 = 0xFFFFFFFF if x5 negative, else 0
srai x8, x6, 31      # x8 = 0xFFFFFFFF if x6 negative, else 0

# Check if signs differ (XOR)
xor x9, x7, x8       # x9 = 0xFFFFFFFF if signs differ, else 0

# Case 1: Signs differ (x9 != 0)
# x5 > x6 if x5 is positive (x7 == 0)
xori x10, x7, 1      # x10 = 1 if x5 positive, else 0
and x10, x10, x9     # Only keep if signs differ

# Case 2: Signs same (x9 == 0)
# Compute x5 - x6 and check if positive
sub x11, x5, x6      # x11 = x5 - x6
# Get sign of result (x11)
srai x11, x11, 31    # x11 = 0xFFFFFFFF if x11 negative, else 0
xori x11, x11, 1     # x11 = 1 if x11 >= 0, else 0
# Only use this result when signs same
not x9, x9           # x9 = 0 when signs differ
and x11, x11, x9     # Mask

# Combine results
or x10, x10, x11     # Final result in x10
```

Execution: