Lab No. 04: Logical Instruction, Use of lo and hi Registers through, Multiplication and Division & Character Manipulation

Objectives:

- Recognize and apply logical MIPS assembly language instructions.
- Acquire the ability to multiply and divide using the lo and hi registers.
- Use shift instructions to manipulate bits.
- Recognize and use masking techniques to change or isolate particular parts.
- To identify and flip particular bits in a register, use exclusive-or.

Task 1: Division Multiplication

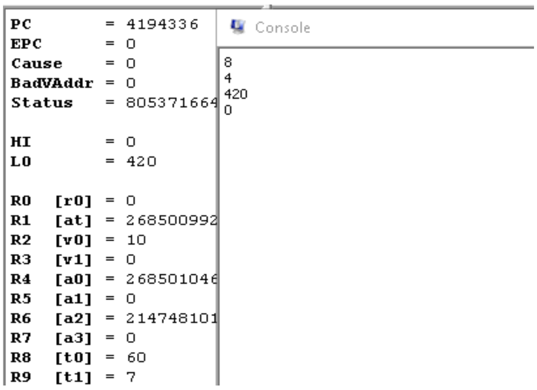
1. Create a directory for this lab's files and open SPIM with hex display in the register pane.
2. Write code to perform integer division using div and use mflo and mfhi to retrieve the quotient and remainder.
3. Modify the code to also calculate and display the product using the `mult` instruction.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# Task 1

.data
quotient: .asciiz "The quotient is: "
remainder: .asciiz "The remainder is: "
product: .asciiz "The product is: "
newline: .asciiz "\n"
.text
main:
    addi $t0, $0, 60  # $t0 = 60
    addi $t1, $0, 7   # $t1 = 7

    div $t0, $t1
    mflo $a0          # Quotient in $a0
    mfhi $a1          # Remainder in $a1

    li $v0, 1
    syscall           # Print quotient

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 1
    move $a0, $a1
    syscall           # Print remainder

    li $v0, 4
    la $a0, newline
    syscall

    mult $t0, $t1
    mflo $a0          # Lower 32 bits of product
    mfhi $a1          # Upper 32 bits of product

    li $v0, 1
    syscall           # Print lower product bits
```

```
  li $v0, 4
  la $a0, newline
  syscall

  li $v0, 1
  move $a0, $a1
  syscall           # Print upper product bits

  li $v0, 4
  la $a0, newline
  syscall

  jr $ra
```

Output:



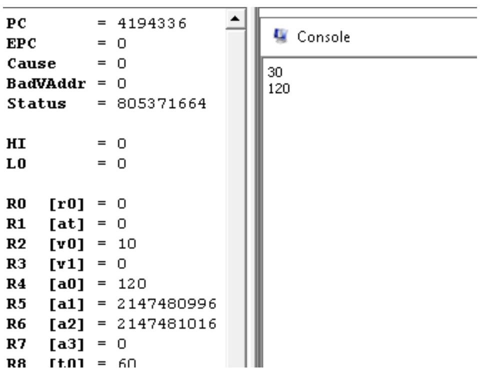
Task 2: Bitwise Shifting

1. Write code to shift a value to the left and right.
2. Experiment with different shift amounts and analyze the binary and hex representation.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# Task 2

.data
newline: .asciiz "\n"

.text
main:
    addi $t0, $0, 60      # $t0 = 60

    srl $a0, $t0, 1       # Shift right by 1
    li $v0, 1
    syscall               # Print result

    li $v0, 4
    la $a0, newline
    syscall

    sll $a0, $t0, 1       # Shift left by 1
    li $v0, 1
    syscall               # Print result

    jr $ra
```

Output:



Observations:

- Right shifts correspond to division by powers of 2.
- Left shifts correspond to multiplication by powers of 2.

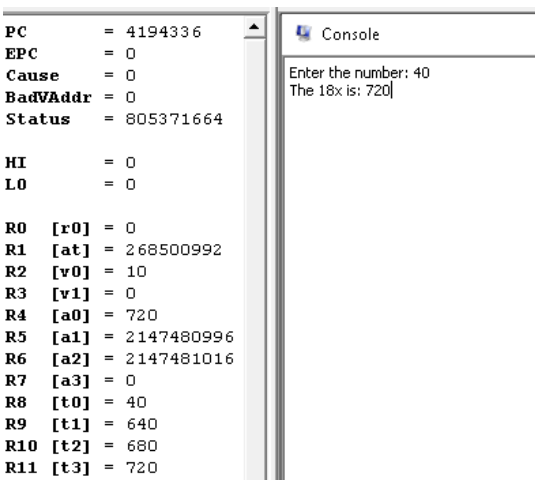
Task 3: Integer Multiplication by Shifting

1. Calculate 18 * x using shift instead of the mult instruction.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# Task 3

.data
prompt1: .asciiz "Enter the number: "
prompt2: .asciiz "The 18x is: "
.text
main:

    li $v0, 4
    la $a0, prompt1
    syscall

    li $v0, 5             # Read integer x
    syscall
    move $t0, $v0         # Store x in $t0

    li $v0, 4
    la $a0, prompt2
    syscall

    sll $t1, $t0, 4       # x * 16
    add $t2, $t1, $t0     # x * 16 + x
    add $t3, $t2, $t0     # (x * 16 + x) + x = 18x



    li $v0, 1
    move $a0, $t3
    syscall               # Print 18x

    jr $ra
```

Output:



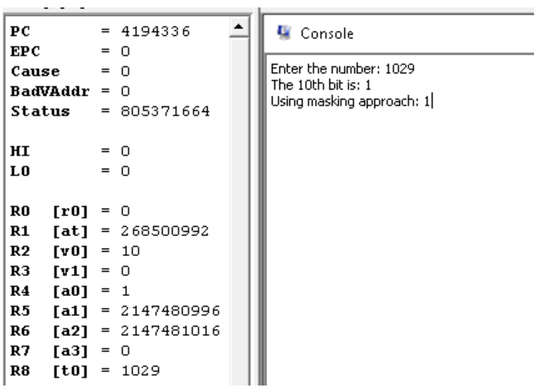
Task 4: Isolating a Specific Bit

1. Isolate bit #10 of an integer by using shifts and masking techniques.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# Task 4

.data
prompt1: .asciiz "Enter the number: "
prompt2: .asciiz "The 10th bit is: "
prompt3: .asciiz "Using masking approach: "
newline: .asciiz "\n"

.text
main:

    li $v0, 4
    la $a0, prompt1
    syscall

    li $v0, 5            # Read integer
    syscall
    move $t0, $v0        # Store input in $t0

    li $v0, 4
    la $a0, prompt2
    syscall

    sll $a0, $t0, 21     # Shift left by 21
    srl $a0, $a0, 31     # Shift right by 31
    li $v0, 1
    syscall              # Output bit #10

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, prompt3
    syscall

    # Masking approach
    andi $a0, $t0, 1024  # Mask bit #10
    srl $a0, $a0, 10
    li $v0, 1
    syscall              # Output bit #10 with mask

    jr $ra
```

Output:



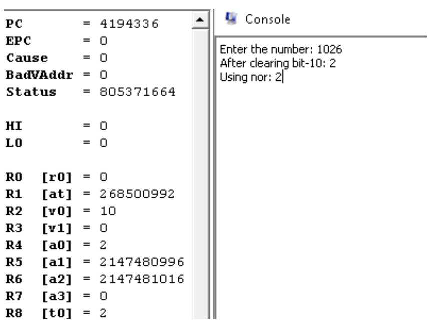
Task 5: Clearing a Specific Bit

1. Clear bit #10 in an integer using a mask and verify using `andi` and `ori`.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

.data
prompt1: .asciiz "Enter the number: "
prompt2: .asciiz "After clearing bit-10: "
prompt3: .asciiz "Using nor: "
newline: .asciiz "\n"

.text
main:

    li $v0, 4
    la $a0, prompt1
    syscall

    li $v0, 5
    syscall
    move $t0, $v0

    li $v0, 4
    la $a0, prompt2
    syscall

    # Create mask for clearing bit #10
    lui $t5, 0xffff
    ori $t5, $t5, 0xfbff
    and $t0, $t0, $t5

    li $v0, 1
    move $a0, $t0
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, prompt3
    syscall

    # Alternative mask generation using `nor`
    li $t6, 1024
    nor $t6, $zero, $t6  # ~1024
    and $t0, $t0, $t6

    li $v0, 1
    move $a0, $t0
    syscall

    jr $ra
```

Output:



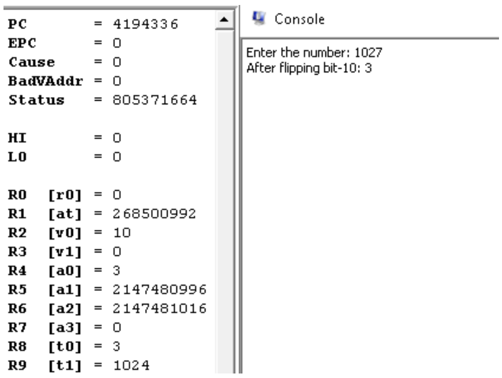
Task 6: Flipping a Specific Bit (Lab4_6.s)

1. Flip bit #10 using the xor instruction with an appropriate mask.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# Task 6

.data
prompt1: .asciiz "Enter the number: "
prompt2: .asciiz "After flipping bit-10: "

.text
main:

    li $v0, 4
    la $a0, prompt1
    syscall

    li $v0, 5              # Read integer
    syscall
    move $t0, $v0          # Store input in $t0

    li $t1, 1024           # Mask for bit #10
    xor $t0, $t0, $t1      # Flip bit #10

    li $v0, 4
    la $a0, prompt2
    syscall

    li $v0, 1
    move $a0, $t0
    syscall                # Output result

    jr $ra
```

Output:



Conclusion:

In this lab, we learnt how to do arithmetic and logical operations using MIPS assembly instructions, specifically using the `lo` and `hi` registers for division and multiplication. We also looked at shifting and masking, which are effective ways to divide, multiply, and manipulate bits.