5/6/2021

**CO LAB-08**

**NAME: Amina Qadeer**

**SYNDICATE: CE-42-A**

**TEACHER: Rimsha Tariq**

**ROLL # 359607**

**Lab Tasks:**

**Question 1**

Perform sll on the following bits of data, Shift the bits 2 times (shift amount =2) and at each shift

Store the result in a different register, Analyse the values of registers changed after each shift

1- 0000 1010 1111 1010 0010 0100 1100 1110

2- 1111 1000 1111 1010 0010 0111 1100 1110

3- 1111 1100 1110 0000 1111 1001 0000 1111

#Question: Perform sll on the following bits of data, Shift the bits 2 times

#1-0000 1010 1111 1010 0010 0100 11001110

#2-1111 1000 1111 1010 0010 0111 11001110

#3-1111 1100 1110 0000 1111 1001 00001111

.data

.text

Main:

li $s0, 0xAFA24CE #converting first to hexa decimal

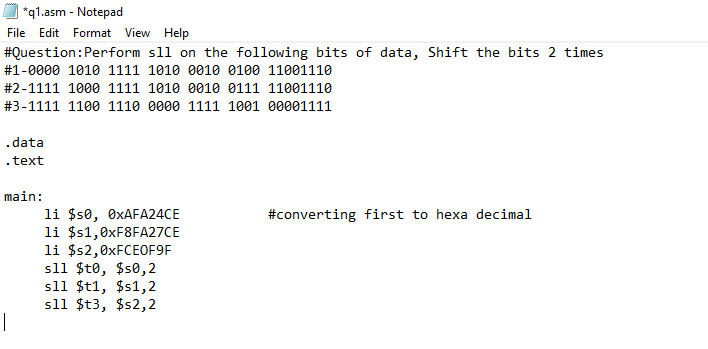
li $s1,0xF8FA27CE

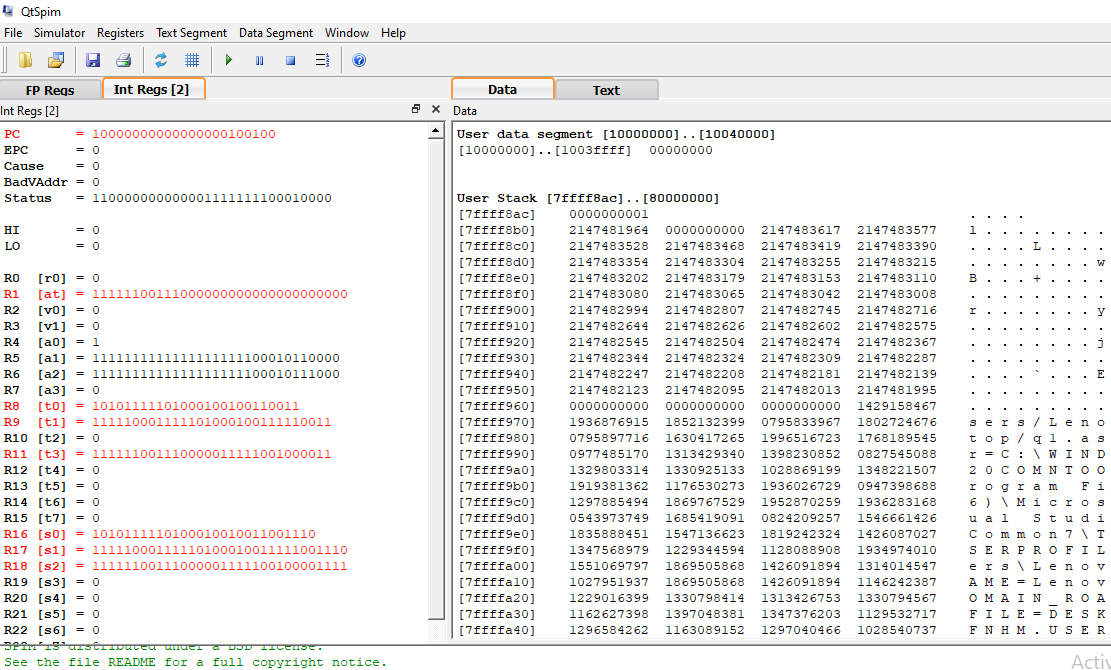
li $s2,0xFCEOF90F

sll $t0, $s0,2

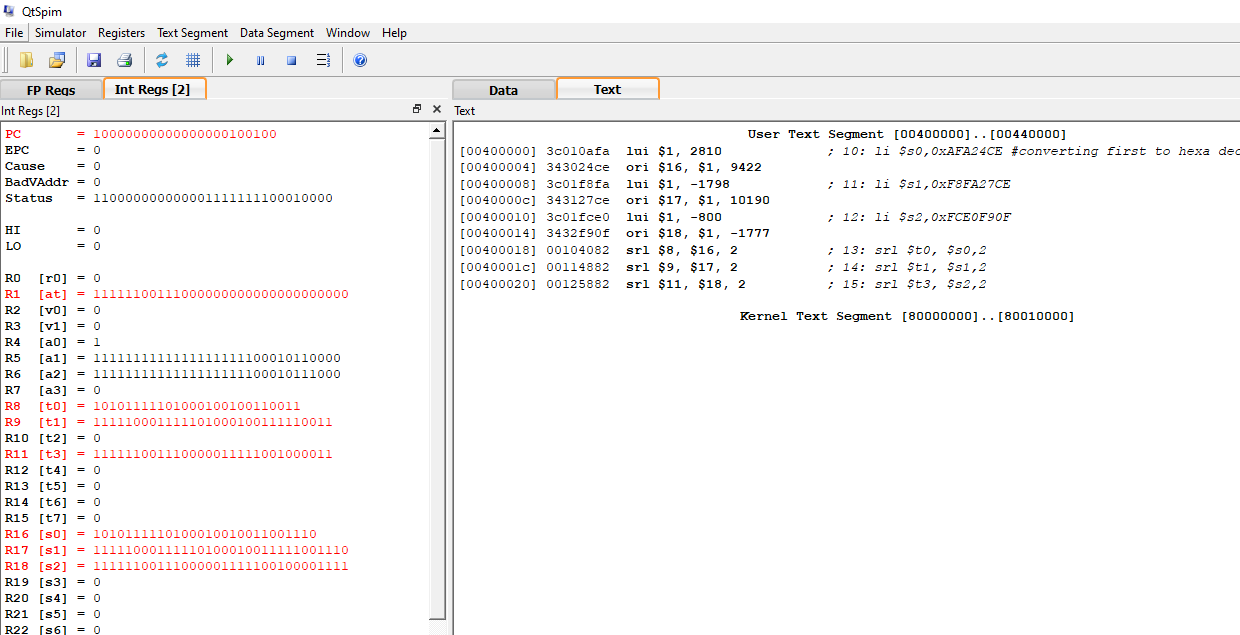
sll $t1, $s1,2

sll $t3, $s2,2





**OUTPUT:**



**Question 2**

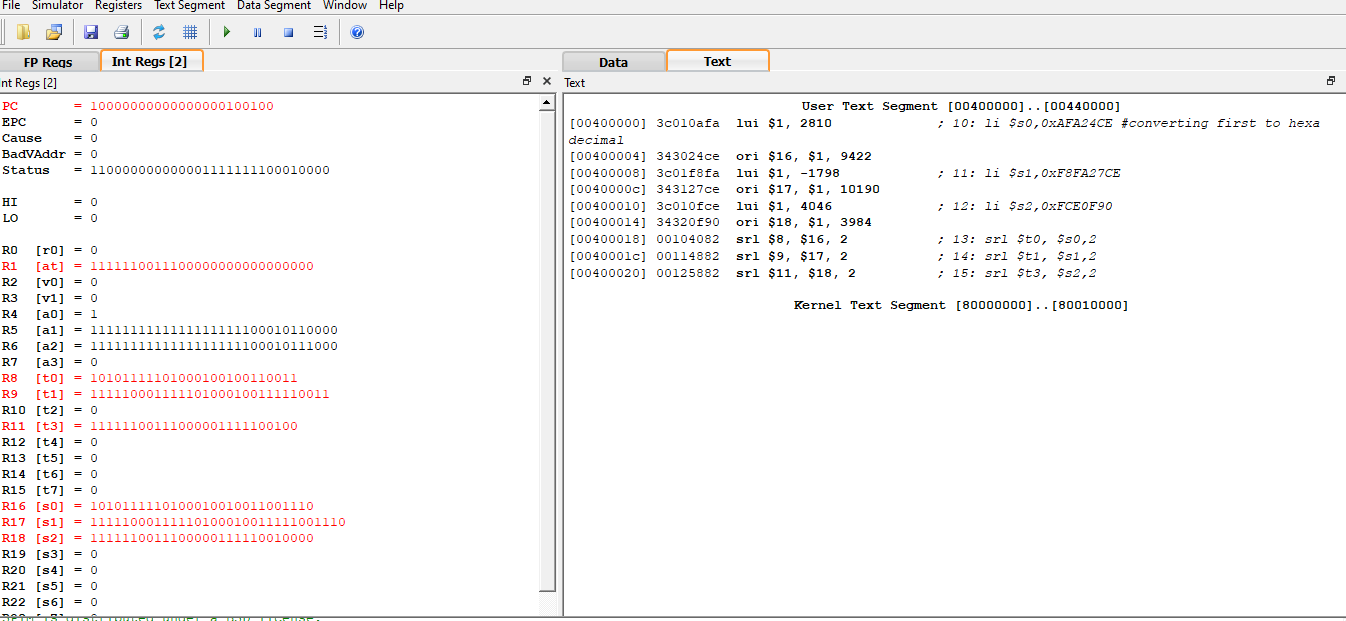
Perform srl on the following bits of data, Shift the bits 2 times (shift amount =2) and at each

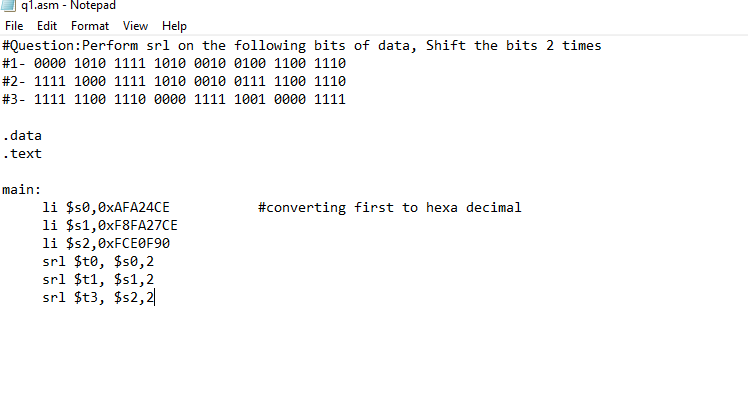
Shift store the result in a different register, Analyse the values of registers changed after each shift

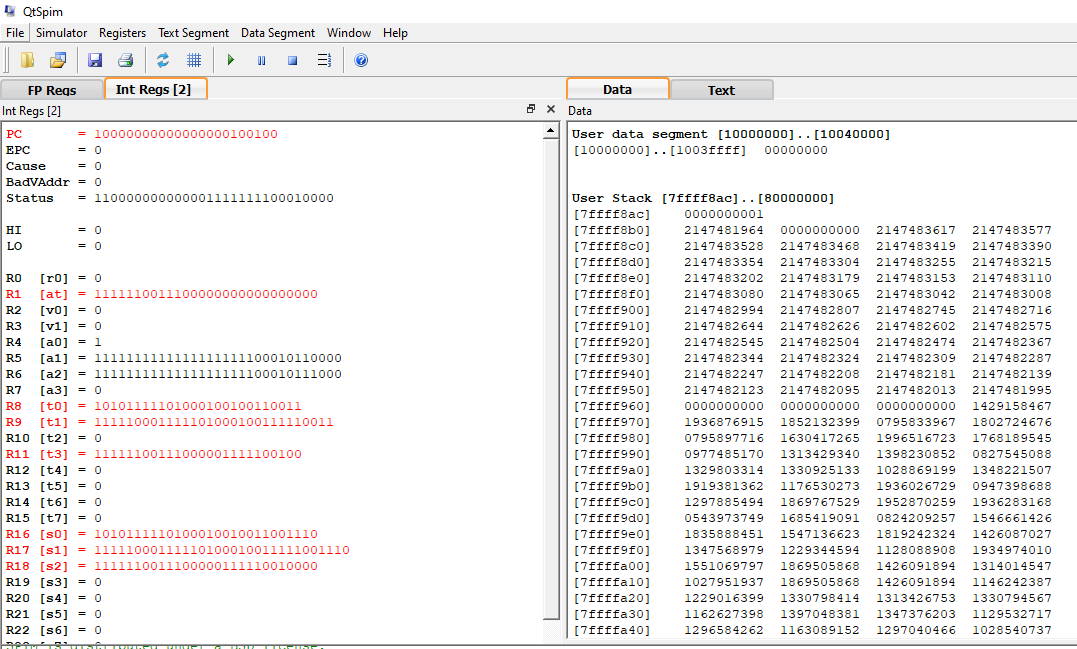
1- 0000 1010 1111 1010 0010 0100 1100 1110

2- 1111 1000 1111 1010 0010 0111 1100 1110

3- 1111 1100 1110 0000 1111 1001 0000 1111

**OUTPUT:** 

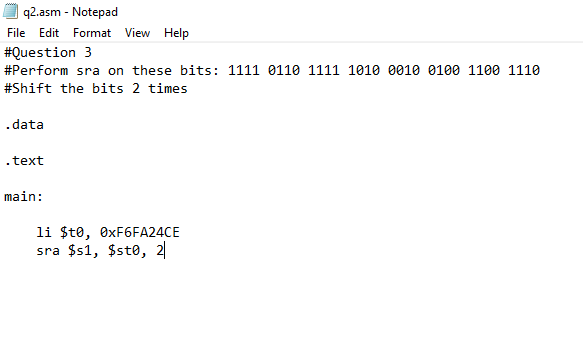


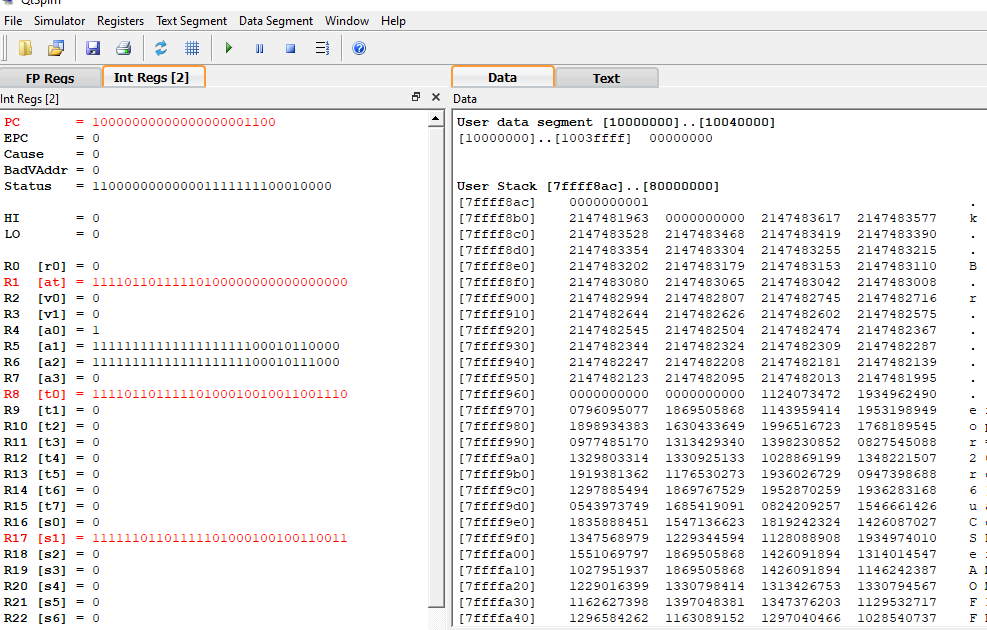


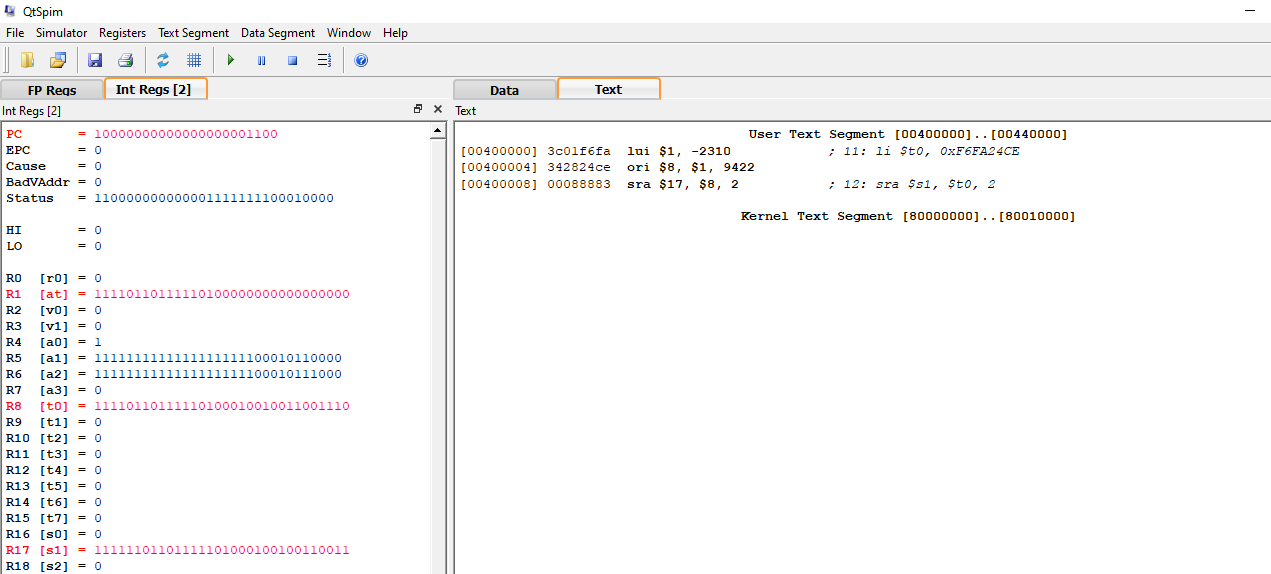
**Question 3**

Perform sra on these bits: 1111 0110 1111 1010 0010 0100 1100 1110

Shift the bits 2 times







**Question 4**

Assuming the following data declarations:

Num1: .word 101110001

Num2: .word 100111101

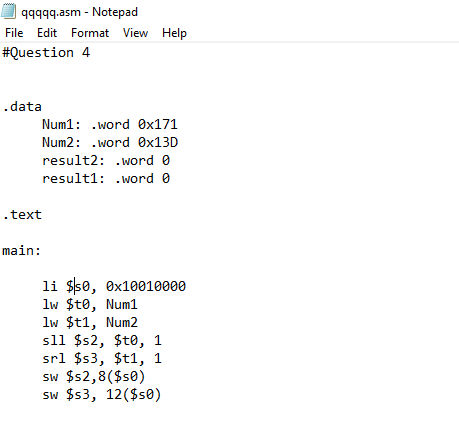
result1: .word 0

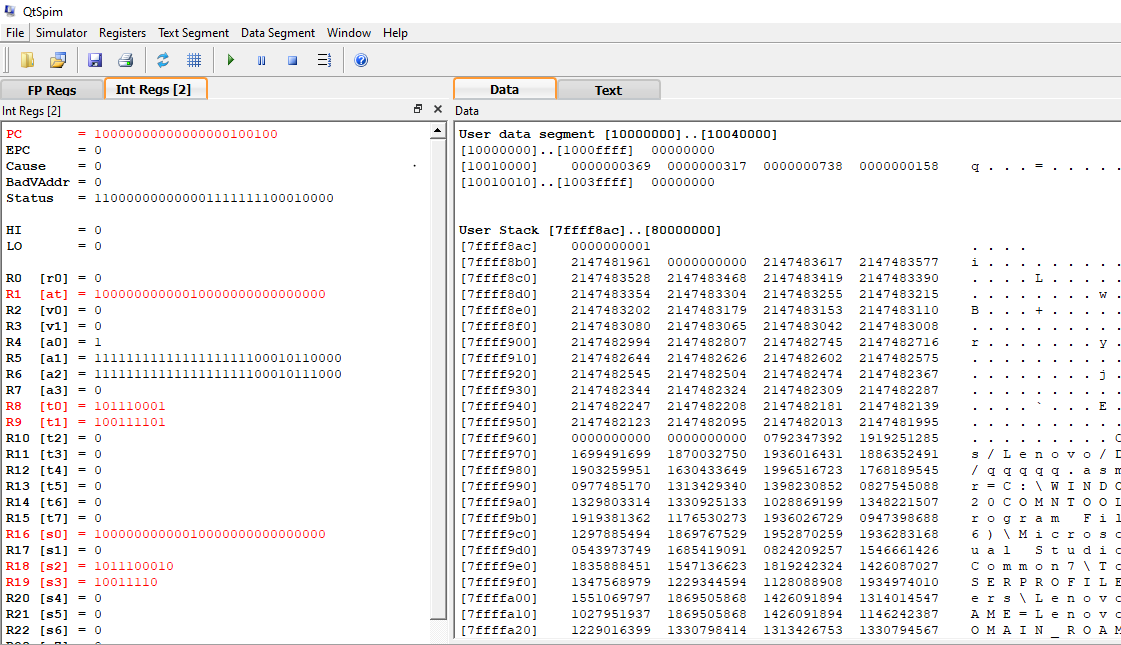
result2: .word 0

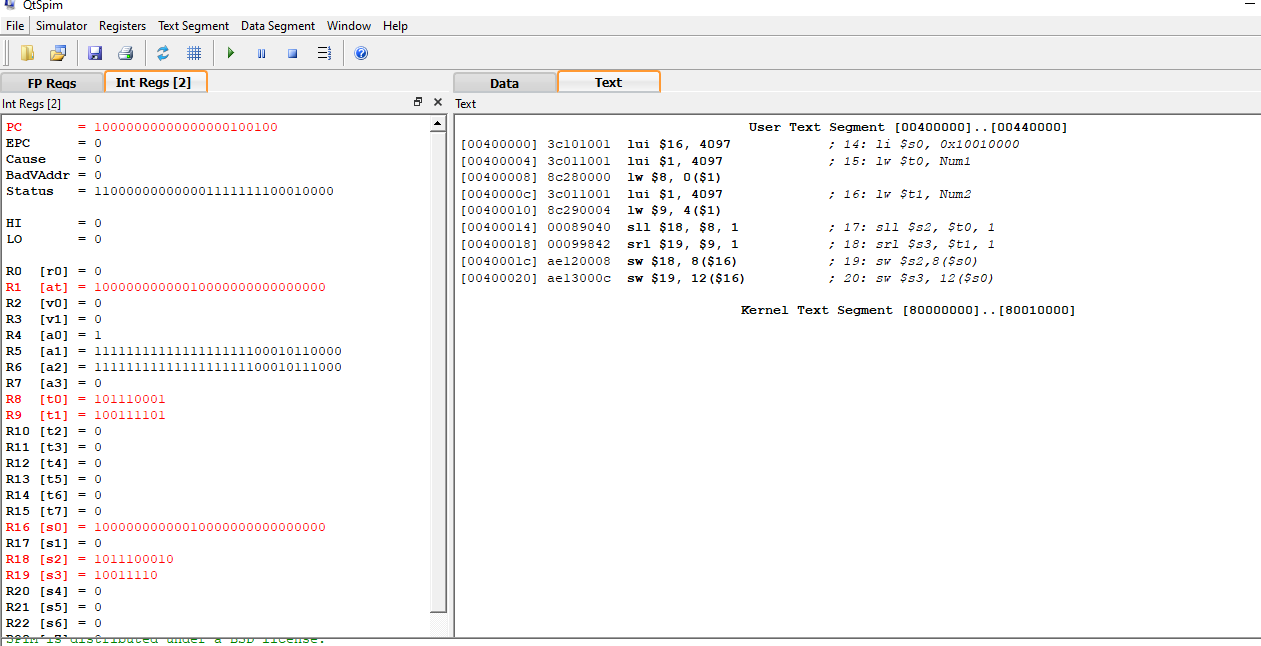
Perform the basic operations of:

result1 = wnum1, shift left 1 bit

result2 = wnum2, shift right 1 bit







**Lab Tasks:**

**What is difference between arithmetic left shift and logical left shift?**

A **shift left logical** of one position moves each bit to the **left** by one. The low-order bit (the right-most bit) is replaced by a zero bit and the high-order bit (the **left**-most bit) is discarded. **Shifting** by two positions is the same as performing a one-position **shift** two times.

**Arithmetic left shifts** are equivalent to multiplication by a (positive, integral) power of the radix (e.g., a multiplication by a power of 2 for binary numbers). Logical **left shifts** are also equivalent, except multiplication and **arithmetic shifts** may trigger **arithmetic** overflow whereas logical **shifts do** not.

**Some differences**:

1. Arithmetic shift preserve sign bit, whereas Logical shift cannot preserve sign bit.
2. Arithmetic shift perform multiplication and division operation, whereas Logical shift perform only multiplication operation.
3. Arithmetic shift is used for signed interpretation, whereas Logical shift is used for unsigned interpretation.
4. With a logical shift, the bits of a word are shifted left or right. On one end, the bit shifted out is lost. On the other end, a 0 is shifted in.
5. The arithmetic shift operation treats the data as a signed integer and does not shift the sign bit. On a right arithmetic shift, the sign bit is replicated into the bit position to its right. On a left arithmetic shift, a logical left shift is performed on all bits but the sign bit, which is retained.

