Program 1:

```
000011000  // mov 1
000100110  // ldr r0 r2
000001000  // mov 0
000110110  // ldr r0 r3 # load input into r2 and r3
000001000  // mov 0
110100000  // bne r2 0
111100000  // bne r3 0
000111001  // mot r1 1   # move 0 into r1
000101000  // mov 2
000010111  // str r0 r1
000111000  // mov 3
000010111  // str r0 r1  # done flag
111111111  // DONE
000011000  // mov 1          ZERO TRAP JUMP (0)
000001010  // lso r0 0
000001010  // lso r0 0
000001010  // lso r0 0
000001010  // lso r0 0
000001010  // lso r0 0
000001010  // lso r0 0
000001010  // lso r0 0
110100001  // bne r2 1
000001000  // mov 0
111100001  // bne r3 1
000011000  // mov 1
000111001  // mot r1 1
011111000  // mov 15
000001010  // lso r0 0
001000100  // add r1 r0
000101010  // lso r1 0
000101010  // lso r1 0
000101010  // lso r1 0
000111000  // mov 3
000010111  // str r0 r1  # stores the most significant bits
000001000  // mov 0
000111001  // mot r1 1
000101000  // mov 2
000010111  // str r0 r1  # stores the least significant bits done flag
111111111  // DONE
000011000  // mov 1 MAX NEG TRAP JUMP (1)
000111001  // mot r1 1  # place a 1 in r1 to compare with r0 later
001001001  // mot r2 0  # move r2 into r0, right shift 7 times
000011010  // lso r0 1
000011010  // lso r0 1
000011010  // lso r0 1
000011010  // lso r0 1
```

```
000011010  // lso r0 1
000011010  // lso r0 1
000011010  // lso r0 1
011111001  // mot r7 1  # r7 holds sign val for later
010110010  // lsl r2 r3
001101010  // lso r3 0
101100010  // bne r1 2
010000101  // neg r2 r0
011000101  // neg r3 r0
000001000  // mov 0
000000101  // neg r0 r0
111100011  // bne r3 3
011000101  // neg r3 r0  # this case is for lower byte full
000011000  // mov 1
010000100  // add r2 r0
000001000  // mov 0
000111001  // mot r1 1
101000010  // beq r1 2  # skipping the normal case
000011000  // mov 1    NOMAL CASE JUMP (3)
011000100  // add r3 r0
011111000  // mov 15   SIGN CHECK JUMP (2)
000001010  // lso r0 0  we have 30 as out exponent
011011001  // mot r6 1
000011000  // mov 1
000111001  // mot r1 1  hold a 1 in R1 for easy comparison
000011000  // mov 1    # loop start (4)
011001001  // mot r6 0
000011101  // sub r0 r1 first loop puts us at 29
011011001  // mot r6 1
000101010  // lso r1 0
000101010  // lso r1 0
000101010  // lso r1 0
000101010  // lso r1 0
000101010  // lso r1 0
000101010  // lso r1 0
000101010  // lso r1 0
001001011  // cfb r2
010110010  // lsl r2 r3
001101010  // lso r3 0
000111010  // lso r1 1
000111010  // lso r1 1
000111010  // lso r1 1
000111010  // lso r1 1
000111010  // lso r1 1
000111010  // lso r1 1
000111010  // lso r1 1
101100100  // bne r1 4 # loop end
```

```
011101001  // mot r7 0
000111001  // mot r1 1
000011000  // mov 1
101100101  // bne r1 5
011001001  // mot r6 0
000111001  // mot r1 1
000000101  // neg r0 r0
100000110  // beq r0 6
011001001  // mot r6 0 END OF IF JUMP (5)
000111001  // mot r1 1
000001000  // mov 0
000101010  // lso r1 0 END OF ELSE JUMP (6)
000101010  // lso r1 0
000101010  // lso r1 0
000010010  // lsl r0 r1
000101010  // lso r1 0
000010010  // lsl r0 r1
000101010  // lso r1 0
000010010  // lsl r0 r1
000101010  // lso r1 0
000010010  // lsl r0 r1
000101010  // lso r1 0
000010010  // lsl r0 r1
000101010  // lso r1 0
000010010  // lsl r0 r1
000101010  // lso r1 0
000010010  // lsl r0 r1
000101010  // lso r1 0
001001110  // mbs r2 1
010110010  // lsl r2 r3
001101010  // lso r3 0
010110010  // lsl r2 r3
001101010  // lso r3 0
000111001  // mot r1 1
000111000  // mov 3
000010111  // str r0 r1
000101000  // mov 2
000100111  // str r0 r2
111111111  // DONE
```

Program 2:

```
Mov 4          # Load LSW address
Ldr R0, R2     # Load LSW into R2
Mov 5          # Load MSW address
Ldr R0, R3     # Load MSW into R3
Cfb R3
Mot R4 1 #move sign bit to R4
Mov 124
```

```
And R0, R3
Mot R1 1
Mov 15
Sub R1 R0 #(R1 = Exp - Bias)
Mot R1 0 #Move adjust exp to R0
Mot R5 1 #Move adjust exp to R5
Mot R5 0
Mot R1 1
Mov 0
Beq R1
Mov 31
Beq R1 special
Mov 3
And R3 R0 #R3 holds MSW of mantissa (masked)
Mov 4
Or R0, R3 #add leading 1
Mot R5 0
Mot R1 1
Mov 10
Sub R1 R0          # Calculate Adjusted_exp - 10, result in R1
Mov 0
Beq R1, align done     # If adjusted exponent == 10, no need to align
Cmp R1 R0
Mov 0
Beq R0 leftshift
Bne R0 rightshift
Lsl R3 R2 #start LeftShift(39):
Mov 1
Sub R1 R0
Mov 0
Bne R1 leftshift
Beq R1 align done
Lsr R3 R2 #start RightShift(45):
Mov 1
Add R1 R0
Mov 0
Bne R1 rightshift
Beq R1 align done
Mot R4 0#start align_done(51):
Mot R1 1
Mov 1
Beq R1 negative
Bne R1 result
Neg R2 R2 #start Negative(56):
Neg R3 R3
```

```
Mov 1
Add R2 R0
Mov 0
Beq R2 carry1
Bne R2 result
Mov 1 #start Carry1(63):
Add R3 R0
Mov R1 1
Mov 0
Bne R1 result
Mov 0 #start Special_case(67):
Mot R1 1
Mov 6
Str R0 R1
Mov 7
Str R0 R1
Bne R1 11111 #DONE
Mov 6 #start result(74):
Str R0 R2
Mov 7
Str R0 R3
Bne R1 11111 #DONE
```

Program 3:

```
Mov 9           # Load XMSW address
Ldr R0, R2      # Load XMSW into R6
Mot R2 0
Mot R6 1
Mov 11          # Load YMSW address
Ldr R0, R3      # Load YMSW into R7
```

```
Mot R3 0
Mot R7 1
mov 15
lso r0 0
mot r1 1
mov 1
add r1 r0
lso r1 0
lso r1 0
mot r1 0
And R2 R0 #mask Xexp and put in R2
Lso R2 1
Lso R2 1
And R3 R0 #mask Yexp and put in R3
Lso R3 1
Lso R3 1
Mov 15
Sub R2 R0 #subtract 15(bias) from Xexp and store in R2
Sub R3 R0 #subtract 15(bias) from Yexp and store in R3
Cmp R2 R3 #if X>Y R0=0, X<Y R0 = 1, X=Y R0=2
Mot R1 1
Mov 0
Beq R1 ShiftY # if Xexp > Yexp
Mov 1
Beq R1 ShiftX if Xexp <= Yexp
Mot R2 0 #start ShiftY:
Mot R1 1 #move Xexp to R1
Sub R1 R3 #R1 = Xexp-Yexp (shift amount)
Mot R2 0
Mot R5 1 #move exponent to R5
Mot R7 0
Mot R2 1
Mov 3
And R0 R2 #mask mantissa for Y Msw
Mot R7 1 #Move YMSW mantissa to R7
Mov 9
Ldr R0 R3 #load XMSW into R3
Mov 3
And R0 R3 #mask mantissa for XMsw
Mot R6 1 #Move XMSW mantissa to R6
Mov 10
Ldr R0 R3 #move Ylsw into R3
Mov 8
Ldr R0 R4 #move Xlsw into R4
Mov 0
```

```
Bne R1 loop #shift Y right until exp aligned
Mot R3 0 #start ShiftX:
Mot R1 1 #move Yexp to R1
Sub R1 R2 #R1 = Yexp-Xexp (shift amount)
Mot R3 0
Mot R5 1 #move exponent to R5
Mot R6 0
Mot R2 1
Mov 3
And R0 R2 #mask mantissa for X Msw
Mot R6 1 #Move XMSW mantissa to R6
Mov 11
Ldr R0 R3 #load YMSW into R3
Mov 3
And R0 R3 #mask mantissa for Y Msw
Mot R7 1 #Move YMSW mantissa to R7
Mov 8
Ldr R0 R3 #move Xlsw into R3
Mov 10
Ldr R0 R4 #move Ylsw into R4
Mov 0
Bne R1 loop #shift Y right until exp aligned
Lsr R2, R3 #start loop:
Mov 1
Sub R1 R0
Mov 0
Bne R1 loop
Beq R1 addLsw
Mot R4 0 #start addLSW:
Add R0 R3 #R0=Xlsw + Ylsw
Mot R4 1
Cmp R0 R3 #if 69 < x
Mot R1 1
Mov 1
Beq R1 carry2
Bne R1 nocarry
Mov 1 #start carry2:
Mot R1 1 #Carry is in R1
Beq R1 addmsw
Mov 0 #start nocarry:
Mot R1 1
Beq R1 addmsw
Mot R6 0 #start addmsw:
Mot R2 1
Mot R7 0
Mot R3 1
Add R1 R2
Add R3 R1 #R3 = Xmsw + Ymsw + carry
Mov 7
Cmp R3 R0
Mot R1 1
Mov 0
Beq R1 overflow #if 69ing MSW > 7
Bne R1 combine
Mot R4 0 #start Overflow:
Mot R2 1
Lsr R3 R2
Mot R5 0
Mot R1 1
Mov 1
Add R0 R1
Mot R5 1
Mov 0
Mot R1 1
Mov 1
Bne R1 Combine
Mov 3 #start combine:
And R3 R0 #remove leading 1 from mantissa
Mov 15
Mot R1 1
Mot R5 0
Add R1 R0 #add bias back
Mot R5 1
Mov 9
Ldr R0, R1      # Load XMSW into R1
Cfb R1
Mot R1 1
Lso R1 0
Lso R1 0
Lso R1 0
Lso R1 0
Lso R1 0
Lso R1 0
Lso R1 0
Mot R5 0
Lso R0 0
Lso R0 0
Or R1 R0 #109 sign bit and exponent bits
Or R3 R1 #109 MSW of mantissa
Mov 13
```

```
Str R0 R3
Mov 12
Str R0 R2
Bne R1 11111 #DONE FLAG
```