SW2: Control flow, Bit masking and Disassembly

Summer 2021

Thiha Myint

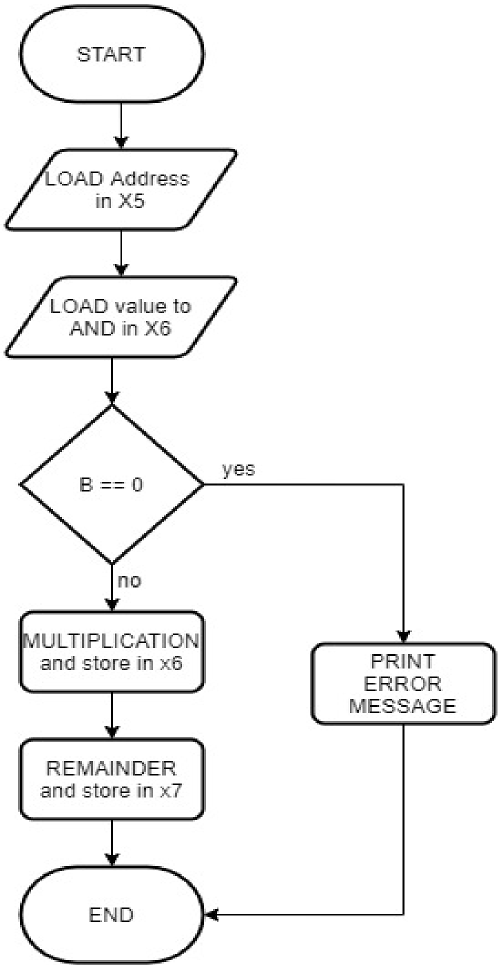
Description

This programing has two parts. First one is as follow: Put a value A in register x4. Put a value B in register x5. Calculate A/B and store the quotient in register x6, and the remainder in register x7. Repeat. Be careful about holding the divide by zero case.

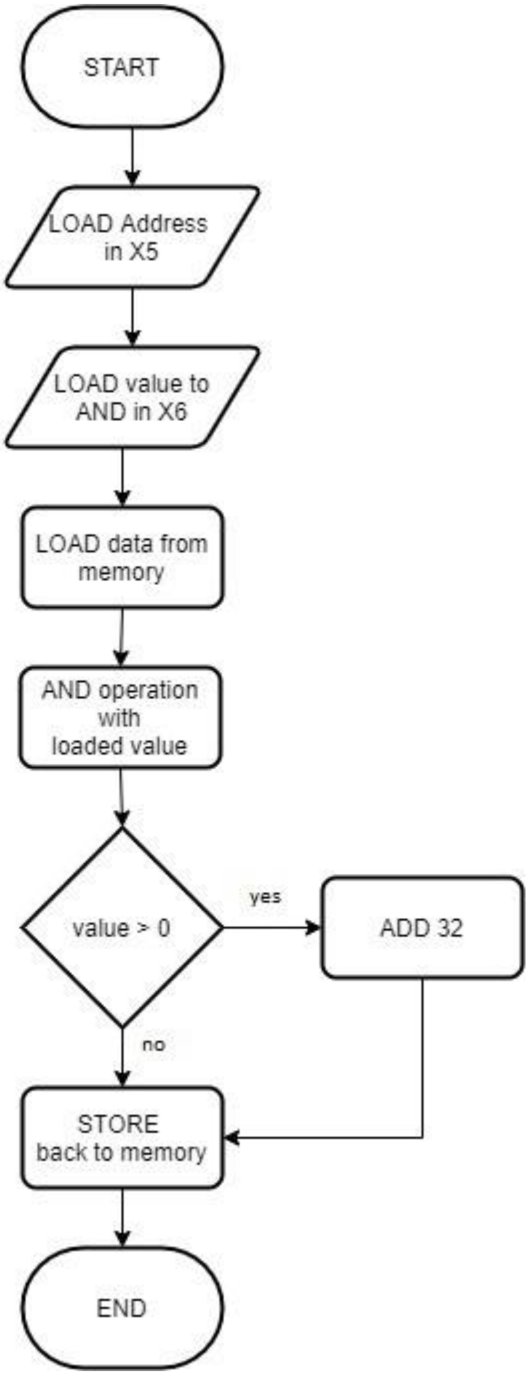
The second part is as follow: Read a value from memory at location 0xFFFF, then AND the value with 0xFF. If the result is > 0, add 32 to the value, otherwise keep the value the same. Then store the value back into the memory at 0xFFFF.

Flowchart

1)



2)


START
LOAD Address in X5
LOAD value to AND in X6
LOAD data from memory
AND operation with loaded value
value > 0
yes
ADD 32
no
STORE back to memory
END

RISC-V Program code

1)

```
.data
A: .word 0x64             # Let A be 100 decimal
B: .word 0x05             # Let B be 5

.text

Start:
li x2,0                   # Load 0 to x2
la x1,A                   # Load address of A to x1
lw x4,0(x1)               # Load value at A to x4
lw x5,4(x1)               # Load value at B to x5

beq x5,x0,invalid         # If divisor is 0 then division is invalid so
branch to invalid label
Division:
bge x4,x5,next            # If x4 is greater than or equal to x5 branch to
next
add x7,x4,x0              # If x4 is less than x5, then it will be remainder,
so move it to x7 by addion with x0 which is 0
add x6,x2,x0              # x2 will contain quotient so move to x6
j end                     # End the division of current  values
next:
addi x2,x2,1              # Add 1 to x2
sub x4,x4,x5              # Subtract x5 from x4, and store result in x4
j Division                # Jump to division
invalid:
j Start                   # Jump to start till divisor is non zero

end:
```

2)

```
li x5, 0xFFFF #Address
        li x6, 0xFF   #Value to AND

        lw x7, 0(x5)  #LOADING DATA FROM Address

        and x7,x7,x6 #Performing AND operation
        bgtz x7, add_value #checking if value is > 0
store:
        sw x7, 0(x5) #storing back to memory
        j end
add_value:
        addi x7, x7, 32 #adding 32 in memory
        j store
end:
```