```
add R1 R2 R3
00000_00_001_010_011
// 2 unused bits
sub R1 R2 R3
00001_00_001_010_011
// 2 unused bits
mov R1 $Imm (mov R1 $12) //// 12 is a decimal value
00010_0_001_0001100
// 1 unused bit
mov R1 R2
00011_00000_001_010
// 5 unused bits
Id R1 var_name2
                                  //// while creation of binary code var_name2 will be
replaced by 7 bit binary address
00100 0 001 0001100
// 1 unused bits
st R2 var name1
                                  //// while creation of binary code var_name1 will be
replaced by 7 bit binary address
00101 0 010 0001111
// 1 unused bits
mul R1 R2 R3
00110_00_001_010_011
// 2 unused bits
div R3 R4
00111_00000_011_100
// 5 unused bits
rs R1 $Imm (rs reg1 $12) // 12 is in decimal
01000 0 001 0001100
// 1 unused bit
Is R1 $Imm
             (Is reg1 $13) // 13 is in decimal
01001_0_001_0001101
// 1 unused bit
```

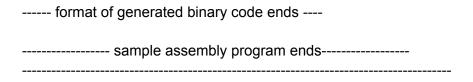
Note: ALL NUMBERS ARE UNSIGNED.

xor R1 R2 R3 01010**_00**_001_010_011 // 2 unused bits or R1 R2 R3 01011_**00**_001_010_011 // 2 unused bits and R1 R2 R3 01100 00 001 010 011 // 2 unused bits not R1 R2 01101_00000_001_010 // 5 unused bits cmp R1 R2 01110 00000 001 010 // 5 unused bits jmp label1 //// while creation of binary code label1 will be replaced by 7 bit binary address 01111**_0000**_0001101 // 4 unused bits ilt label2 /// while creation of binary code label2 will be replaced by 7 bit binary address 11100 **0000** 0001000 // 4 unused bits jgt label3 /// while creation of binary code label3 will be replaced by 7 bit binary address 11101**_0000**_0001100 // 4 unused bits je label4 /// while creation of binary code label4 will be replaced by 7 bit binary address 11111 0000 0001110 // 4 unused bits hlt 11010_00000000000

// 11 unused bits

```
addf R1 R2 R3
10000_00_001_010_011
// 2 unused bits
subf R1 R2 R3
10001_00_001_010_011
// 2 unused bits
```

```
----- one sample assembly program-----
var var1
var var2
var var3
ld R1 var1
ld R2 var2
st R3 var3
jmp hlt_label
add R1 R2 R3
hlt label: hlt
----- memory content explanation---- //// remember we are following von-numen architecture so
only
                                                //// single memory both for instruction and
data
addr instruction code
0000000: 00100_0_001_0000110
0000001: 00100 0 010 0000111
0000010: 00101_0_011_0001000
0000011: 01111 0000 0000101
0000100: 00000 00 001 010 011
0000101: 11010_00000000000
---- locations allocated to variables in the order of their declaration (var1,var2,var3)
0000110:
0000111:
0001000:
---- memory content explanation ends ----
---- format of generated binary code ----
0010000010000110
0010000100000111
0010100110001000
0111100000000101
000000001010011
1101000000000000
```



```
------ corected program from the shared project pdf ------
----- assembly program----
var X
mov R1 $10
mov R2 $100
mul R3 R2 R1
st R3 X
hlt
------ assembly program ends -----
------ explanation of binary from the assembler--
0000000: 00010_0_001_0001010
0000001: 00010_0_0101_1100100
```

0000010: 00110 _00 _011_010_001
0000011: 00101_ 0 _011_0000101
0000100: 11010 _0000000000
explanation of binary from the assembler ends
binary genrated form assembler
0001000010001010
0001000101100100
0011000011010001
0010100110000101
110100000000000
binary genrated form assembler ends
corected program from the shared project pdf ends