

Note : ALL NUMBERS ARE UNSIGNED.

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

ld R1 var_name2
replaced by 7 bit binary address
00100_0_001_0001100
// 1 unused bits
/// while creation of binary code var_name2 will be

st R2 var_name1
replaced by 7 bit binary address
00101_0_010_0001111 ✓
// 1 unused bits
/// while creation of binary code var_name1 will be

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

ls R1 \$Imm (ls reg1 \$13) // 13 is in decimal
01001_0_001_0001101 ✓
// 1 unused bit

```
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
```

```
jlt 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_000000000000
// 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
```

```
//// our floating point representation have no sign bit , 3 exponent bits, 5 mantissa bits ////
//// exponent_in_float = exponent_in_binary + bias
////(1.5)10 = (1.1)2 = 1.1 X 20
////1.1 X 20 in binary = (0 + 3) in exponent 10000 in mantissa
////representation = 111_10000
////----- bias formula for floating point = bias_in_binary = 2(k-1) -1 where k: no of bits in
exponent
```

```
movf reg1 $1.5          //// 1.5 is in decimal representation
10010_001_111_10000
// 0 unused bit
```

```
addr_label: add R1 R2 R3  //// here when the addr_label will be used then it will be replaced
                        //// by the 7-bit address (where this add
instruction is stored)
addr_label2: hlt
```

----- 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_000000000000
```

---- 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
0000000001010011
1101000000000000
```

----- format of generated binary code ends ----

----- sample assembly program ends-----

----- 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_010_1100100

0000010: 00110_00_011_010_001

0000011: 00101_0_011_0000101

0000100: 11010_000000000000

----- explanation of binary from the assembler ends--

----- binary generated from assembler-----

0001000010001010

0001000101100100

0011000011010001

0010100110000101

1101000000000000

----- binary generated from assembler ends---

----- corected program from the shared project pdf ends -----
