**floating-point formula:**

V = (-1)S x M x 2E

**single precision:**

0 00000000 000...--> 23

s exponent (e) mantissa (f)

**double precision:**

0 00000000000 000...--> 52

s exponent (e) mantissa (f)

**normalized:**

- exponent = not all 0s or all 1s

- e = is unsigned exponent value

- E = e - Bias

- Bias = 2k-1-1

- f = the mantissa

- M = f + (implied 1)

**denormalized:**

- exponent = all 0s

- E = 1 - Bias

- M = f

**special:**

- exponent is all 1s

- if fraction is all 0s:

+ value is +∞ when s = 0

+ value is -∞ when s = 1

- if the fraction is non-zero:

+ value is NaN

**decimal to float (add bias):**

1. Turn decimal into binary:

24 / 2 = 12 remainder 0 <- LSB

12 / 2 = 6 remainder 0

6 / 2 = 3 remainder 0

3 / 2 = 1 remainder 1

1 / 2 = 0 remainder 1 <- MSB

2. Move decimal so one 1 left of it

3. Unbiased exponent = num shifts

+ positive if left shifts

- negative if right shifts

4. Determine the bias: 2k-1-1

5. Add bias to unbiased exponent

6. Turn biased exponent into binary

7. Put it all together

8. Sign bit!

**float to decimal (subtract bias):**

1. Separate the fields:

0 00000000 00000000000000000000000

s exponent mantissa

2. Put biased exponent in decimal

3. Subtract bias from exponent

E = e - Bias

13 = 140 - 127

4. Put in scientific (implicit 1):

1.1000000111001 x 213

mantissa

5. Move the decimal N times

right if N is positive

left if N is negative

6. Convert to decimal

7. Sign!

**push %ebp:**

%esp -= 4

movl %ebp, %esp

**pop %ebp:**

movl (%esp), %ebp

%esp += 4

**call:**

%eip + 4

j \*adrs

**leave:**

movl %ebp, %esp

popl %ebp

**ret:**

popl (rtrn adrs)

j \*rtrn adrs

**caller-save registers:**

%eax, %ecx, %edx

**callee-save registers:**

%ebx, %edi, %esi

**array access by address:**

L = size of array type

xA = address of A

element i is at xA+L\*i

**array access in assembly:**

suppose:

int E[]

%edx = &E

%ecx = i

then:

movl (%edx,%ecx,4),%eax

results in:

%eax = E[i]

**pointer arithmetic examples:**

suppose:

%edx = address of array E

%ecx = value of i

%eax = result

then E[2] is:

movl 8(%edx),%eax

and E+i-1 is:

leal -4(%edx,%ecx,4),%eax

and \*(&E[i]+i) is:

leal (%edx,%ecx,4),%eax

movl (%eax,%ecx,4),%eax

or:

movl (%eax,%ecx,8),%eax

**divide by power of two:**

- to round, you must bias by 2k - 1

int mask =

((~(x>>((sizeof(int)<<3)-1)))&1)+1;

int bias = mask & ((1<<k) - 1);

return (x + bias) >> k;

**registers:**

31 15 8 7 0

%eax %ax %ah %al

%ebx %bx %bh %bl

%ecx %cx %ch %cl

%edx %dx %dh %dl

%esi %si %sil

%edi %di %dil

%ebp %bp %bpl

%esp %sp %spl