Assignment 3

Ross O. Shoger

July 23, 2014

1 Convert IEEE.754 Single Precision to Decimal

Convert the value of 01000001001011000000000000000000 to decimal.

1.1 Determining Sign

The first bit is the sign, which is1. Our result is positive.

1.2 Unbiased Exponent

The second part of the sequence is the exponent stored as an 8-bit unsigned integer with a bias of 127.

10000010

So our exponent must be such that:

$$10000010 = x + 1111111 \tag{1}$$

or $x = 10000010 - 1111111$ or $x = 130 - 127$ which is

$$x = 00000011 \tag{2}$$

So our exponent is 3.

1.3 Fraction

The fraction of the binary representation is the remaining 23-bits

01011000000000000000000

Placing a 1 in front of the significant bits and disregarding the trailing '0's

101011

1.4 Denormalize

If we combine the exponent, signficand and sign, we end up with a binary number.

$$+1.01011 * 2^3$$

If we denormalize the binary number we get:

Binary Value	Biased Exponent	Sign, Exponent, Fraction
1010.11	130	0 10000010 01011000000000000000000

$$+1010.11 = (1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (0 * 2^0) + (1 * 2^{-1}) + (1 * 2^{-2})$$

And from that we can find out final decimal value:

$$1010 = 10$$

$$.11 = 1/2 + 1/4$$

$$10 + 1/2 + 1/4 = 10 + .5 + .25 = 10.75$$

So we end up with **10.75**

2 Convert a Decimal to IEEE Double Precision

Convert -75.025 to IEEE.754 double precision.

2.1 Determine Sign

We can determine the first bit to be a '1' as the decimal is negative.
So far we have:

1 00000000000 0000000000000000000000000000000000000000000000000000

11-bits for the exponent, and 52 bits for the fractional component.

2.2 Find the Binary

$$75 = 1001011$$
$$.025 = 0000011001$$

Which is:

$$1001011.000001101$$

and, Which is:

$$1001011.000001101 = 1.001011000001101 * 2^6$$

We now know our exponent is 6. We must add our bias (1023 to get our 11-bits:

$$6 + 1023 = 1028$$

$$00000000110 + 01111111111 = 10000000101$$

Now have:

1 10000000101 0000000000000000000000000000000000000000000000000000

Then we can normalize our binary number and replace the left most bits with our result:

$$1.001011000001101 \text{ normalized becomes } 001011000001101$$

and our answer is:

1 10000000101 0010110000011010000000000000000000000000000000000000

3 IEEE Floating Point Representation

Single Precision	32-bits: 1-bit for the sign, 8-bits for the exponent, and 23-bits for the fractional part of the significand. Approximate nomalized range 2^{-126} to 2^{127}. Also called a *short real*.
Double Precision	64-bits: 1-bit for the sign, 11-bits for the exponent, and 52-bits for the fractional part of the significand. Approximate normalized rang: 2^{-1022} to 2^{1023}. Also called a *long real*.

4 Passing Parameters

4.1 C Style Call

In order to make a C style call, the values or addresses to be used must be pushed to the stack. The function being called must adjust the base pointer to access the appropriate memory, and then restore ebp for a proper return.

```
.code
addnumbers proc
        push ebp
        mov ebp, esp
        sub eax, eax
        mov eax, dword ptr [ebp+8]
        add dword ptr [ebp+12]
        add dword ptr [ebp+16]
        ret
add_numbers endp

main proc
        push 1
        push 2
        push 3
        call add_numbers
        add esp, 12
main endp
end main
```

In the above function 3 values are pushed to the stack. 'addnumbers' is called and control is transferred. In the subroutine eax is given the first value, and the rest are added. The return value is passed in eax to the caller. The stack is adjusted by the caller.

4.2 Passing by Registers

Passing by registers is generally a bad idea for several reasons. Passing by registers can make your source difficult to follow an maintain as its not always clear without stepping through the program how a particular register might be use in a process, unless of course it has been documented. Furthermore, by relying on registers to hold and pass data, you make the program logic overly complex, as well as introduce errors, as register states must now be constantly maintained via the stack each time a process, or function, is to be called.

5 16-bit 2s' Compliment

The representation and method to arrive at -378 in 16-bit 2s' compliment is:

$$((2^{16} - 1) - 378) + 1) = 1111111010000110$$