COMPSCI 210 S1T 2005 Tutorial Six -------Data Representation cont.
Aim for the tutorial:

In this tutorial, we will study more detail about Data Representation. After tutorial five we already know about how to transformation different base number and how to calculate unsigned and signed number, now we will go though overflow and underflow in arithmetic, Integer data type, excess-k , Floating-point representation, the VAX formats and the IEEE formats. Also we will do some exercise together in the tutorial.

1. Overflow and underflow in computing:

Overflow and Underflow in addition:

- Adding two numbers with different signs can never produce an overflow or underflow.
- Adding two positive numbers produces an overflow if the sign of the result is negative.
- Adding two negative numbers produces an underflow if the sign of the result is positive.
- Note that in one case there is a carry out and in the other there is not

```
 (+7 )   0111          (-7)   1001
 (+6)   0110          (-4)     1100
(+13)   1101         (-11)     0101
```

Overflow and Underflow in Subtraction:

- Subtracting two numbers with the same signs can never produce an overflow or underflow.
- Subtracting a negative number from a positive number produces an overflow if the sign of the result is negative.
- Subtracting a positive number from a negative number produces an underflow if the sign of the result is positive.

```
 (+4)      0100 0100         -4      1100 1100
-(-5)     -1011 0101      -(+5)     -0101 1011
  +9            1001         -9           0111
```

Carry from MSB?	Carry into MSB?	overflow
no	no	no
no	yes	yes
yes	no	yes
yes	yes	no

2. Integer data type and floating-point representation:

Integer data type value table:

Size	bits	Signed	unsigned
byte	8	-128 to +127	0 to 255
word	16	-32768 to +32767	0 to 65535
longword	32	-2**31 to +(2**31-1)	0 to 2**32-1
Quadword	64	-2**63 to +(2**63-1)	0 to 2**64-1
Octaword	128	-2**127 to +(2**127-1)	0 to 2**128-1

Binary floating-point representation：

we can display the floating-point number X to:

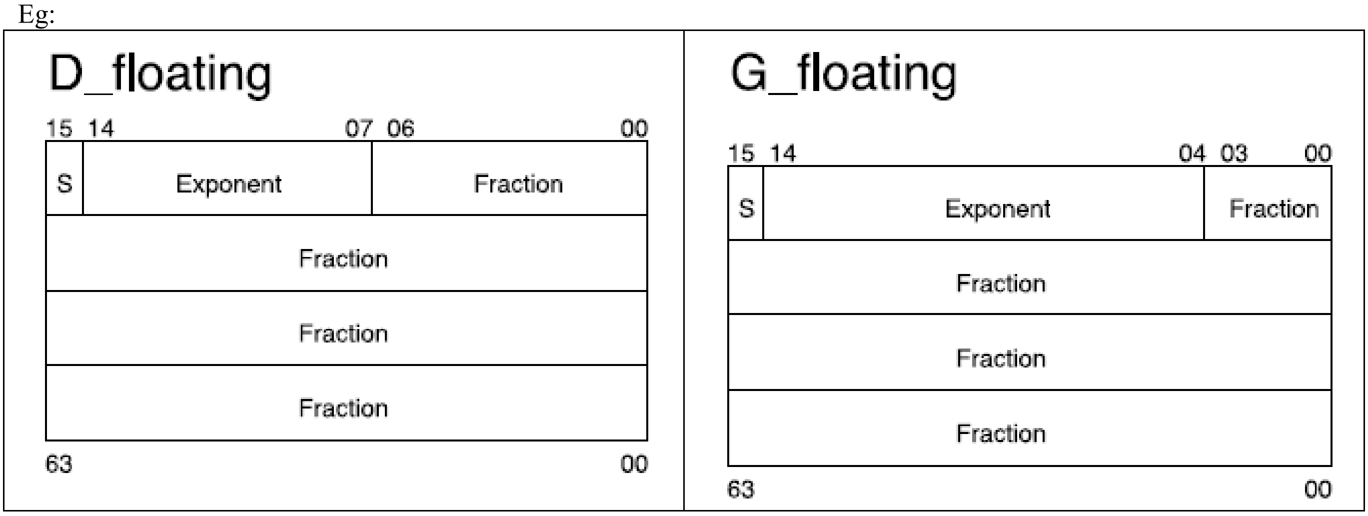
X= (-1)**S x *fraction * 2 ** {exponent –k}

example:
1000.10000000_2 =8 + 1/512

3. The VAX and IEEE formats:

The AVX formats:
F (Floating, single precision, 32bits, 2words, 4 bytes)
D (Double precision, 64bits,4words)
G (Grand 64 bits)
H (Huge 128 bits 8 words, quadruple precision)

Eg:



For example :
We have a G_Floating number: 0000000028004070_{16}
Step1: the sing bit is 0,so the number is positive number.
Step2: Exponent $407_{16} = 1031_{10}$, so the exponent is 1031-1024= 7
Step3: Fraction the remind part is :0280000000000_{16}
 In binary is:0000 0010 1000 0000……0000
 Normalised form:
 So Add a 1 on the left with the binary point:
 Then we get : .1 0000 0010 1000 …… 0000

Final Conversion: multiply by the exponent 2 **7 with .1000000101
 Then we get 1000000.101 → 32+0.5+0.125 =32.625

The IEEE formats:

1. Single precision(32bits)

2. double precision(64bits)
3. quadruple precision(128bits)

eg:

```
  1      8               23              width in bits
 +-+--------+-----------------------+
 |S|   Exp    |   Fraction                |
 +-+--------+-----------------------+
 31 30     23 22                    0    bit index (0 on right)
    bias +127
```

An example:
Let's encode the decimal number −118.625 using the IEEE 754 system.
We need to get the sign, the exponent and the fraction.
Because it is a negative number, the sign is "1". Let's find the others.
First, we write the number (without the sign) using binary notation. Look at binary numeral system to see how to do it. The result is 1110110.101
Now, let's move the radix point left, leaving only a 1 at its left: $1110110.101=1.110110101 \cdot 2^6$
The fraction is the part at the right of the radix point, filled with 0 on the right until we get all 23 bits. That is 11011010100000000000000.
The exponent is 6, but we need to convert it to binary and bias it (so the most negative exponent is 0, and all exponents are non-negative binary numbers). For the 32-bit IEEE 754 format, the bias is 127 and so 6 + 127 = 133. In binary, this is written as 10000101.
Putting them all together:

```
  1      8               23              width in bits
 +-+--------+-----------------------+
 |S|   Exp    |   Fraction                |
 |1|10000101|11011010100000000000000|
 +-+--------+-----------------------+
 31 30     23 22                    0    bit index (0 on right)
    bias +127
```

Difference between VAX and IEEE formatting:

The main difference between VAX and IEEE formatting is the convention of fraction part. VAX is 0.1M, however IEEE is 1.M.
Eg:
For same $1101\ 1010\ 1000\ 0000\ 0000\ 000_2$
In VAX: 0.11101 1010 1000 0000 0000 000
In IEEE: 1.1101 1010 1000 0000 0000 000

4. Exercise:

Question 1:
What is the 32 bits 2's complement representation for -78?
78→ 1001110→ $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 1110_2$

So -78→$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011\ 0010_2$

Question 2:
What is result for 17 Add 19 in binary? And check is overflow or not in 5 bits(unsigned).

17_{10} = 10001_2 19_{10} = 10011_2

```
                        1  11  <--- Carry bits
  (Showing sign bits)   010001
                      + 010011
                      --------
                        100100
```

That will be overflowing just use 5 bits binary, but not overflow in 6 bit binary.

Question 3:

What is result for -17 Add -19 in binary? And check is overflow or not in 6 bits.

-17_{10} = 101111_2 -19_{10} = 101101_2

```
                       1 1111  <--- Carry bits
  (Showing sign bits)   101111
                      + 101101
                      --------
     Discard extra bit → 1011100
```

FINAL ANSWER: 011100_2 = $+28_{10}$

But if we use 8 bits binary to represent the result that will be 11011100(-36).

Question 4:

What is decimal number for D_Floating number 0000000080004402_{16}?

(Also do at home with same question for IEEE double precision)

Step1: the sing bit is 1, so the number is negative number.

Step2: Exponent $10001000_2 = 136_{10}$, so the exponent is 136-128= 8

Step3: Fraction the remind part is :028000000000000_{16}

In binary is:0000 0010 1000 0000……0000

Normalized form:

So Add a 1 on the left with the binary point:

Then we get : .1 0000 0010 1000 …… 0000

Final Conversion: multiply by the exponent 2 **8 with .1 0000 00101

Then we get 10000001.01 → 2**8+1+0.25=257.25

Final answer is – 257.25

Question 5:

What is decimal number for G_Floating number 0000000038004080_{16}?

(also do at home with same question for IEEE double precision)

Step1: the sing bit is 1,so the number is positive number.

Step2: Exponent $408_{16} = 1032_{10}$, so the exponent is 1032-1024= 8

Step3: Fraction the remind part is :038000000000000_{16}

In binary is:0000 0011 1000 0000……0000

Normalized form:

So Add a 1 on the left with the binary point:

Then we get : .1 0000 0011 1000 …… 0000

Final Conversion: multiply by the exponent 2 **8 with .1000000111

Then we get 10000001.11 → 128+1+0.5+0.25= 129.75

Final answer is 129.75

Question 6:
What is decimal number for IEEE754 Single precision number:
1 10000110 1100 0011 1000 0000 0000 000_2?
do the same exercise for F_floating?
Sign first bit is 1, so the number is negative;
10000110 = 134, 134-127= 7: so Exponent is 7.
Fraction the remind part is: 1100 0011 1000 0000 0000 000
1.1100 0011 1 * 2**7 = 1110000.111 = 64+32+16+0.5+0.25+0.125=112.875
Final answer is: - 112.875

Question 7:
What is binary number for IEEE754 Single precision number -123.25?
(also do at home with same question for IEEE double precision)
Cover 123.25 to binary number: 1111011.01 = 1.11101101*2**6
So Fraction is 1110 1101 0000 0000 0000 000
Exponent is 127+6 = 134 = 10000101

So the answer is: 1 10000101 1110 1101 0000 0000 0000 000_2

5. Reference:

http://scholar.hw.ac.uk/site/computing/topic1.asp?outline=no
http://www-ee.eng.hawaii.edu/Courses/EE150/Book/chap1/subsection2.1.2.1.html

the IEEE formats:
http://www.psc.edu/general/software/packages/ieee/ieee.html
http://en.wikipedia.org/wiki/IEEE_floating-point_standard
http://home.earthlink.net/~mrob/pub/math/floatformats.html