COMP2611 COMPUTER ORGANIZATION

DATA REPRESENTATION REVIEW NOTES

1. Unsigned integer

- Just use positional notation
- Range: $[000...0_2, 111...11_2] = [0, 2^k - 1]$
- Get HEX: Still positional notation. Or, simply combine every 4 bits in binary.
- Example:

  $(1001\ 0011)_2 = 2^7 + 2^4 + 2^1 + 2^0 = 147_{10} = \mathrm{0x93}$

2. Signed integer (2's complement)

- Use left most bit to represent +/- ve, 0 for positive, 1 for negative
- For positive: the value is the same as positional notation.
- For negative: Reverse all bits, then add 1.
- The smallest representable value is $100...00_2 = -2^{k-1}$, the largest is $011...11_2 = 2^{k-1} - 1$
- Get HEX: Combine every 4 bits in binary.(*Which means that hexadecimal numbers still obey 2's complement, while decimal numbers don't. Find details in Q4*)
- Examples:

(Q1)Represent $+6$?

(Ans)Simply apply positional notation: 0110_2

(Q2)Represent -6?

(Ans)We know $6 = 0110_2$, then we reverse all bits,
get 1001, then add 1, get 1010.

(Q3)A number is represented as 1001 in 2's complement, what's the original value?

(Ans)As left most bit is 1, the number is **negative**. Then still perform the "Reverse-Add" method, $1001 \to 0110 \to 0111 = 7$,
so we know the value is -7.

(Q4)Given the following **signed integer values**, write their corresponding 16-bit 2's complement representations and decimal/hexadecimal values.

(1) $\mathrm{C7A3}_{(16)}$

(Ans) Since the number is **signed**, the hexadecimal value still obeys 2's complement rules. So to get 2's complement representations, we just need to expand each hex bit into 4 binary bits. That is:

$\mathrm{C} = 1100, 7 = 0111, \mathrm{A} = 1010, 3 = 0011,$

Thus the 2's complement representation is $1100\ 0111\ 1010\ 0011$

To get decimal number, we apply "Reverse-Add" method. Notice left most bit is 1, so the number is **negative**. By "Reverse and Add", we get $0011\ 1000\ 0101\ 1101$, and apply positional notation, this equals to 14429. So the decimal value is -14429.

(2)237_{10}

(Ans) Since this a +ve number, the 2's complement is the same as positional notation. So 2's complement is $0000\ 0000\ 1110\ 1101$, hexadecimal number is $\mathrm{0x00ED}$
.

3. More about integer

- Overflow: The number is too large to be represented. $(number > 2^{k-1} - 1)$
- Underflow: The number is too small to be represented. $(number < -2^{k-1})$
- Examples:

(Q1) Is it possible to get an overflow error in 2's complement when adding numbers of opposite signs?

(Ans) Impossible. The representable range of 2's complement is $[-2^{k-1}, 2^{k-1} - 1]$, you can see that adding numbers of opposite signs will always falls in this range.

- Sign extension: when we perform extension on **signed int**.
- Zero extension: when we perform (1)bitwise logical operations, or (2)unsigned int
- Examples:

(Q2) Given the following **signed integer values**, write their corresponding 32-bit 2's complement representations and decimal/hexadecimal values.

(1) $\mathrm{F6C5}_{16}$.

(Ans) This is **signed integer**, so to get 2's complement, we just expand each hex bit into 4 binary bits, that is $1111\ 0110\ 1100\ 0101$. To expand the value into 32-bit, we perform **Sign extension**, that is, extend the left most bit. Thus we get $1111\ 1111\ 1111\ 1111\ 1111\ 0110\ 1100\ 0101$.Notice that this extension doesn't change the value. If we perform zero extension, the value will be changed.

To get decimal value, first perform "Reverse-Add" method, get $0000\ 1001\ 0011\ 1011$, the corresponding positional notation is 2363. Don't forget the sign bit is 1, so the value is -2363.

(2) -2021_{10}

(Ans) First write positional notation, and then perform "Reverse-Add" method, we get 2's complement
$1111\ 1111\ 1111\ 1111\ 1111\ 1000\ 0001\ 1011$. Notice here we still use sign extension to keep the original value unchanged.

To get hex value, just combine 4 binary bits into 1 hex bit, $\mathrm{0xFFFF81B}$.

4. Floating point

- IEEE 754 standard defines a binary representation for floating point values using 3 fields:
    - The *sign* is the left most bit, 0 for positive while 1 for negative.
    - The *exponent* is in **biased notation**, which means the final value is the positional notation value of exponent **plus** bias, where bias is 127 for single-precision number, 1023 for double-precision number.
    - The *significand* or *mantissa* store the numbers **AFTER decimal point** in scientific notation.(Since under scientific notation, every number can be written as $1.abcd...\cdot 10^x$, so we can ignore the 1 before decimal point)
- How many bits for each part?

S	Exponent (biased)	Significand
1	8 (single) / 11 (double)	23 (single) / 52 (double)
0/1	Exponent + Bias = Exponent + $2^{k-1}-1$	after an implicit 1 (hidden bit)

- Denormalized and Special cases

Exponent Significand	0 (denormalized)	$[1, 2^k-2]$ (normalized)	2^k-1 (special cases)
0	0	$(-1)^S \times 1.F \times 2^{E-Bias}$	$(-1)^S \times \infty$
$\neq$ **0**	$(-1)^S \times 0.F \times 2^{-126}$	$(-1)^S \times 1.F \times 2^{E-Bias}$	NaN

- Examples

(Q1) Write the **IEEE754 single-precision** representation of the following decimal numbers. If cannot be represented exactly, find the **nearest** approximation of the number.

(1) -2843.625

$2843 = \text{0xB1B} = 1011\ 0001\ 1011_2$
$-2843.625 = 1011\ 0001\ 1011.101_2 = 1.0110\ 0011\ 0111\ 01_2 \times 2^{11}$
$S = 1$, $\text{Significand} = 0110\ 0011\ 0111\ 0100\ 0000\ 000$
$\text{Biased Exponent} = 11 + 127 = 138 = 1000\ 1010_2$

Therefore, $-2843.625 = 1 \quad 1000\ 1010 \quad 0110\ 0011\ 0111\ 0100\ 0000\ 000$

(2)0.085

Bit	f	f × 2	Bit
	0.085	0.17	0
	0.17	0.34	0
	0.34	0.68	0
Implicit	0.68	1.36	1
22	0.36	0.72	0
21	0.72	1.44	1
20	0.44	0.88	0
...	...	...	...
1	0.72	1.44	1
0	0.44	0.88	**1**

(As $|0.88 - 1| < |0.88 - 0|$, we choose **1** as the last bit.)

$0.085 = 1.0101\ 1100\ 0010\ 1000\ 1111\ 011_2 \times 2^{-4}$
$S = 0$, $\text{Significand} = 0101\ 1100\ 0010\ 1000\ 1111\ 011$

$$\text{Biased Exponent} = -4 + 127 = 123 = 0111\ 1011_2$$

Therefore, $0.085 = 0 \quad 0111\ 1011 \quad 0101\ 1100\ 0010\ 1000\ 1111\ 011$

(Q2) Convert these IEEE 754 floating numbers into decimal numbers.

(1)0 1001 0011 11101100101110110110000

(Ans)

$S = 0, \text{Exponent} = 1001\ 0011_2 = 147$

$$\begin{aligned}
&(-1)^S \times (1 + \text{Significand}) \times 2^{\text{Exponent}-\text{Bias}} \\
=&(-1)^0 \times 1.1110\ 1100\ 1011\ 1011\ 0110\ 000_2 \times 2^{147-127} \\
=&1\ 1110\ 1100\ 1011\ 1011\ 0110.000_2 \\
=&0\text{x}1\text{ECBB6} \\
=&2018230
\end{aligned}$$

(2)1 0000 0000 11100000000000000000000

(Ans)

Notice this is denormalized situation.

$$\begin{aligned}
\text{Significand} &= .1110\ 0000...000_2 \\
&= .111_2 \\
&= .875
\end{aligned}$$

$$\begin{aligned}
&(-1)^S \times \text{Significand} \times 2^{-126} \\
=&(-1)^1 \times 0.875 \times 2^{-126} \\
=&-0.875 \times 2^{-126}
\end{aligned}$$

5. Character

- **Unsigned byte** (8 bits) under the ASCII standard.
- I don't want to say more XD.