COMPUTER ORGANIZATION (CDA – 3103)
SPRING 2005
LAB # 1: NUMBER SYSTEMS

1 NUMBER SYSTEMS / BASES 1

1.1 BINARY 1
1.2 OCTAL 1
1.3 DECIMAL 1
1.4 HEXADECIMAL 1

2 CONVERSION BETWEEN BASES 2

2.1 BINARY TO DECIMAL 2
2.2 DECIMAL TO BINARY 2

3 REPRESENTATION OF A NUMBER IN COMPUTER 3

3.1 UNSIGNED NUMBERS 3
3.2 SIGN NUMBERS 3
3.2.1 Sign Magnitude Form 3
3.2.2 1's Complement Form 4
3.2.3 2's Complement Form 4

1 NUMBER SYSTEMS / BASES

1.1 BINARY

- Also called Base 2
- Number system used by the computer.
- Only two digits, **0** and **1**.
- Example : $(1011)_2$

1.2 OCTAL

- Also called Base 8
- A compact form of Base 2
- Total 8 digits, **0 – 7**.
- Not frequently used.
- Example : $(762)_8$

1.3 DECIMAL

- Also called Base 10
- Number system used in normal life (understandable to humans).
- Total 10 digits, **0** – **9**.
- Example : $(3498)_{10}$

1.4 HEXADECIMAL

- Also called Base 16
- A more compact form of Base 2
- Total 16 digits, **0 – 9, A – F**.
- Example : $(73AD)_{16}$

Decimal Number	Binary Number	Octal Number	Hexadecimal Number
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10
17	10001	21	11

2 CONVERSION BETWEEN BASES

2.1 BINARY TO DECIMAL

$$[\, b_n\, b_{n-1}\, b_{n-2} \ldots b_0 \,.\, b_{-1}\, b_{-2}\, b_{-3} \ldots b_{-m} \,]_2$$

weights $\longrightarrow$

$$2^n\; 2^{n-1}\; 2^{n-2} \ldots 2^0 \,.\, 2^{-1}\; 2^{-2}\; 2^{-3} \ldots 2^{-m}$$

$$= \quad [\, (b_n * 2^n) + (b_{n-1} * 2^{n-1}) + (b_{n-2} * 2^{n-2}) + \ldots + (b_0 * 2^0) + (b_{-1} * 2^{-1}) + (b_{-2} * 2^{-2}) + (b_{-3} * 2^{-3}) + \ldots + (b_{-m} * 2^{-m}) \,]_{10}$$

Example

1. Convert $(1011)_2$ into decimal number.
   $1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 \quad = \quad 8 + 0 + 2 + 1 \quad = \quad (11)_{10}$

2. Convert $(11100)_2$ into decimal number.
   $1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 0*2^0 \quad = \quad 32 + 16 + 8 + 0 + 0 + 0 \quad = \quad (56)_{10}$

3. Convert $(11.101)_2$ into decimal number.
   $1*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} \quad = \quad 2 + 1 + 0.5 + 0.125 \quad = \quad (3.625)_{10}$

2.2 DECIMAL TO BINARY

Example

1. Convert $(18)_{10}$ into binary number.

2	18
2	9 – 0
2	4 – 1
2	2 – 0
2	1 – 0
	0 – 1

   $(18)_{10} = (10010)_2$

2. Convert $(33.25)_{10}$ into binary number.
   Convert 33 and 0.25 separately

2	33
2	16 – 1
2	8 – 0
2	4 – 0
2	2 – 0
2	1 – 0
	0 – 1

2 * 0.25	0.5 + 0
2 * 0.5	0.0 + 1

$(33.25)_{10} = (33)_{10} + (0.25)_{10}$

$= (100001)_2 + (01)_2$

$= (100001.01)_2$

3 REPRESENTATION OF INTEGERS IN COMPUTER

In computer we represent a number using some memory block whose size can be a multiple of 8 bits (bit = binary digits). Using this memory block, unsigned number and signed numbers of different kinds can be represented.

3.1 UNSIGNED NUMBERS

For unsigned numbers, the whole memory block of size N-bits is used for storing the magnitude of the number. The range of numbers that can be represented is 0 -> $(2^N – 1)$

Example

34 = $(100010)_2$

8-Bit representation

0	0	**1**	0	0	0	**1**	0

16-Bit representation

0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0

32-Bit representation

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0

3.2 SIGN NUMBERS

3.2.1 Sign Magnitude Form

If it is an N-bit number, then lower N-1 bits represent the magnitude and the N-th bit represents the sign of that number. The range of numbers that can be represented is $(- 2^{(N-1)} + 1)$ -> $(2^{(N-1)} – 1)$. Zero has two representation; one -0 and +0

Example

16-Bit representation of $(34)_{10}$

0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0

16-Bit representation of $(-34)_{10}$

1	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0

3.2.2 1's Complement Form

Here we represent a positive number and negative number differently.

A positive number is saved in the memory without performing any operation. While for a negative number, its magnitude is first negated (or complemented) and then saved in the memory.

Again +0 and -0 have two different representation; one having all 0's and second having all 1's.

The range of numbers that can be represented is $(-2^{(N-1)} + 1) \rightarrow (2^{(N-1)} - 1)$

Example

16-Bit 1's complement of of $(34)_{10}$

0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

16-Bit 1's complement of of $(-34)_{10}$

$34 = (100010)_2 = (0000000000100010)_2 \overset{\text{complement}}{=} (1111111111011101)_2$

1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1

3.2.3 2's Complement Form

Here we represent a positive number and negative number differently.

A positive number is saved in the memory without performing any operation. While for a negative number, its magnitude is first negated (or complemented), one is added and then saved in the memory.

Now we have only one representation of 0 which is all 0's.

The range of numbers that can be represented is $(-2^{(N-1)}) \rightarrow (2^{(N-1)} - 1)$

Example

16-Bit 1's complement of of $(34)_{10}$

0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

16-Bit 1's complement of of $(-34)_{10}$

$34 = (100010)_2 = (0000000000100010)_2 \overset{\text{complement}}{=} (1111111111011101)_2$

$\overset{\text{add 1}}{=} (1111111111011110)_2$

1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0

4 REPRESENTATION OF FLOATING POINTS IN COMPUTER

IEEE Single Precision Floating Point Number

Represented by 32 bit memory chunk.

31	30 – 23	22 – 0
Sign Bit	Biased Exponent	Significant / Mantissa

4.1 CONVERSION OF A NUMBER INTO IEEE 32-BIT FLOATING POINT REPRESENTATION

Example: Convert $(33.25)_{10}$ in IEEE 32-bit floating point number.

$$(33.25)_{10} = (100001.01)_2$$
$$= (1.0000101 * 2^5)_2$$

- ⇨ Sign = 0
- ⇨ Exponent = $00000101 + (127)_{10}$ = 00000101 + 01111111 = 10000100
- ⇨ Significant = 00001010000000000000000

	Magnitude								Significant																							
0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0