CS Bridge Moduel 2: Positional Number Systems

Digital Data

- Data in the computer's memory is represented using units that can each be in one of 2-states (Ø or 1)

- Memory is made out of Bits, that are grouped into Bytes
  - Bits are physically made with electricity flow or Magnetic Field
    - Abstractlly Bits are either 0 or 1

Data is represented Digitally using Binary Numbers
- Kinds of Data:
  - Numbers: Represented in "Binary"
  - Text: Numbers are mapped to Characters (ASCII)
  - Images: Numbers are used to display color in a pixel (RGB)
  - Video: Sequence of Images
  - Audio: Sampled Voltage Levels

All Data should be transformed into numbers

Counting in Base 10 (Decimal Number System)

- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- We the group 10 - 1's in to a single 10
- Then we place the 1 in tens place and continue adding 1
- 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
- 2 groups of ~~t~~ten = 20

- 10 tens would represent 1-hundred

Counting in Base 5

- 0, 1, 2, 3, 4
- Grouping 5 ones into a single 5 object
- 10, 11, 12, 13, 14
- Another group into a 5 object
- Up till 44
- ~~At~~ 44 we go into 1-100 5 object
- 100, 101, 102, 103, 104

Counting in Base 8

Digits: 0, 1, 2, 3, 4, 5, 6, 7
- One-Octal ten
- 10, 11, 12, 13, 14, 15, 16, 17
- Two-Octal ten
- After 77 we would have - Ten-Octal ten or 100 octals. digit

Counting in Base 2

- Only Digits are 0 and 1
- 0, 1
- One-Binary ten
- 10, 11
- Two-Binary ten or One-Binary Hundred
- 100, 101
- 110, 111
- Two Binary Hundred or One-Binary Thousand
- 1000, 1001
- 1010, 1011
- 1110, 1111

- Numbers grow much faster

Counting in Base 16 (Hexadecimal)

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- We cannot use two digit numbers so instead use letters
- 16 - Hexadecimal Ones = 10 - Hexadecimal ten
- 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F
- What is after "FF"? 100

Equivalent Representations

Example: 13

$(13)_{10}$ = 13 in Decimal System

$(15)_8$ = 13 in Octal System

$(23)_5$ = 13 in Base-5 System

$(1101)_2$ = 13 in Binary System

$(D)_{16}$ = 13 in Hexadecimal System

Base Conversions

(i) N in base b → N in decimal
(ii) N in decimal → N in base b
- These two translations let you convert to any number system

; Base b → decimal

- Example

$(375)_{10} = 5\cdot 10^0 + 7\cdot 10^1 + 3\cdot 10^2$

hundreds tens ones > Each digit has its own weight

$10^2\ 10^1\ 10^0$

$(\underbrace{1}_{8^2}\underbrace{2}_{8^1}\underbrace{5}_{8^0})_8 = 5\cdot 8^0 + 2\cdot 8^1 + 1\cdot 8^2 = 5+16+64 = (85)_{10}$

- So $(125)_8$ amounts to $(85)_{10}$ objects

$(\underbrace{1}_{2^3}\underbrace{0}_{2^2}\underbrace{1}_{2^1}\underbrace{1}_{2^0})_2 = 1\cdot 2^0 + 1\cdot 2^1 + 0\cdot 2^2 + 1\cdot 2^3 = 1+2+0+8 = (11)_{10}$

$(\underbrace{3}_{16^2}\underbrace{b}_{16^1}\underbrace{2}_{16^0})_{16} = 2\cdot 16^0 + 11\cdot 16^1 + 3\cdot 16^2 = 2+176+768 = (946)_{10}$

General Formula:

$(a_n \dots a_2 a_1 a_0) = a_0\cdot b^0 + a_1\cdot b^1 + a_2\cdot b^2 \dots + a_n\cdot b^n$

- Where ever the position of the number is, you multiply that number (a_n) by the power of the weight of the base (b^n)

Knowledge Check

$(123)_8 = 3\cdot 8^0 + 2\cdot 8^1 + 1\cdot 8^2 = 3+16+64 = (83)_{10}$

$8^2\ 8^1\ 8^0$

(ii) Decimal $\rightarrow$ Base b (demonstrated on b=2)

Example $(75)_{10} = (1001011)_2$

$$\ldots \underline{0}\ \underline{0}\ \underline{0}\ \underline{0}\ \underline{1}\ \underline{0}\ \underline{0}\ \underline{1}\ \underline{0}\ \underline{1}\ \underline{1}$$

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	=	=	=	=	=	=	=	=	=	=
	512	256	128	64	32	16	8	4	2	1

$\longrightarrow$

Exposes Bits from left to right

- Need to figure out for each position, should it be a $\emptyset$ or 1 ?
- All digits greater than 75 must be $\emptyset$
- All Bases between 2^6 to 2^0 must equal up to 75
- Geometrical Progression: $1+2+4+8+\ldots 2^k = 2^{k+1}-1$ (Memorize) *

↳ One less than the next power of that sum

example: $1+2+4+8+16+32 = 63 = 2^{5+1}-1 = 64-1 = 63$

- So $75-64=11$; rest of digits must add up to 11.
- $8+2+1 = 11$ so $2^3+2^1+2^0$

(iii) Binary $\longleftrightarrow$ Hexadecimal

Example $(3b9)_{16} = (\underset{3}{\underline{0011}}\,\underset{b}{\underline{1011}}\,\underset{9}{\underline{1001}})_2$

Hex digit	0	1	2	3	4	5	6	7	8	9	a	b
4 bit Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011

c	d	e	f
1100	1101	1110	1111

- Work Digit by Digit and write its 4 bit binary extension
- Allows us an easier way to Convert between Binary and Hexadecimal

Example 2: $(\overset{\text{add }0}{\underset{6}{\underline{0110}}}\,\underset{d}{\underline{1101}}\,\underset{3}{\underline{0011}})_2 = (6d3)_{16}$

- If the numbers don't add up to groups of 4 bit Binary add $\emptyset$'s to leftmost

(iii) Binary $\leftrightarrow$ Hexadecimal (Cont.)

Example 3 : $(0110110100 11)_2$ =

$$= \underbrace{1\cdot 2^0 + 1\cdot 2^1 + 0\cdot 2^2 + 0\cdot 2^3}_{} + \underbrace{1\cdot 2^4 + 0\cdot 2^5 + 1\cdot 2^6 + 1\cdot 2^7}_{} + \underbrace{0\cdot 2^8 + 1\cdot 2^9 + 1\cdot 2^{10} + 0\cdot 2^{11}}_{}$$

$$= (\ \text{"}\ )\,2^0 \qquad\qquad (\ \text{"}\ )\cdot 2^4 \text{ (Factor)} \qquad\qquad (\ \text{"}\ )\cdot 2^8 \text{ (Factor)}$$

$$= (3)\cdot 2^0 \text{ equiv.} + (13)\cdot 2^4 \text{ equiv.} + (6)\cdot 2^8 \text{ equiv.}$$

$$= (3)\cdot 16^0 + (13)\cdot 16^1 + (6)\cdot 16^2$$

$$= (6d3)_{16}$$

$16^2 \; 16^1 \; 16^0$

Knowledge Check : Convert $(3DF)_{16}$ to Binary

$3 = 0011 + D = 1101 + F = 1111 = (001111011111)_2$

Addition

$$\begin{array}{r} {}^{1\,1}\\ 325_{10} \\ +\ 692_{10} \\ \hline 1017 \end{array} \qquad \begin{array}{r} {}^{1\,1}\\ 365_8 \\ +\ 243_8 \\ \hline 630 \end{array}$$

- In Octal Number System Carry the 8!

$$\begin{array}{r} {}^{1\,1}\\ 10011100_2 \\ +\ 11011001_2 \\ \hline 101110101_2 \end{array}$$

- In Binary Number System Carry the 2!

Subtraction

$$\begin{array}{r} {}^{3}\cancel{4}{}^{1}27_{10} \\ -\ 192_{10} \\ \hline 235_{10} \end{array} \qquad \begin{array}{r} {}^{4}\cancel{5}{}^{1}36_{8} \\ -\ 351_{8} \\ \hline 165_{8} \end{array}$$

- When you carry from a higher digit, count by the base $(13)_8 - (5)_8 = (6)_8$

Knowledge Check: Add 1010 + 1111 and convert to Decimal

$$\underset{2^3}{1}\,\underset{2^2}{0}\,\underset{2^1}{1}\,\underset{2^0}{0} = 1 \cdot 2^1 + 1 \cdot 2^3 = 2 + 8 = 10$$

$$1111 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 = 1 + 2 + 4 + 8 = 2^{3+1} - 1 = 2^4 - 1 = 16 - 1 = 15$$

$$10 + 15 = 25$$

Signed Numbers

Example $(26)_{10} = (11010)_2$
- To express this as a negative $(-26)_{10} = (-11010)_2$
- But computers don't have a negative sign represented through bits

- We can approach this problem in a few ways:
  - Sign and Magnitude
  - The left most digit provides the sign (Positive = 0 & Negative = 1)
  - To Represent the magnitude we use the Binary Digit Value $(26)_{10} = (11010)_2$
  - Now combine the sign & magnitude $(\underbrace{1}_{\substack{(-) \\ \text{Sign}}}\ \underbrace{11010}_{\substack{(26) \\ \text{Magnitude}}})_2$

  - The sign itself Has NO WEIGHT

  - Two's Complement - Usually way computers represent negatives

Two's Complement

- In a "k-bit" two's complement representation of a number:

1. A positive integer is represented in its $(k-1)$-bit unsigned binary representation, padded with a "0" to its left

Example: if there are "5" 2-bit digits to represent a positive digit $\underbrace{0}_{k}\underbrace{1011}_{k-1}$

2. The sum of a number and its additive inverse is 2^k

Example: $(26)_{10} = (00011010)_{8 \text{ bit } 2\text{'s complement}}$

- First get the binary representation $= (11010)_2$

0	0	1	1	0	1	0
2^6	2^5	2^4	2^3	2^2	2^1	2^0

$26-16=10-8=2-2=0$

- That representation is 7-bits, the 8th bit would denote that this is positive
- If asking 8 bit rep.; $8=k$ so $k-1=7$
- $(26)_{10} = (\underbrace{0}_{k}\underbrace{0011010}_{k-1})_{8 \text{ bit } 2\text{'s complement}}$

Example: $(-26)_{10} = (11100110)_{8 \text{bit } 2\text{'s complement}}$

- Adding a binary number to its additive inverse should give 2^k

```
  1 1 1 1 1 1 1
  0 0 0 1 1 0 1 0   -> Denotes (26)10 to Binary
  1 1 1 0 0 1 1 0   -> Denotes 8bit 2's complement to (-26)10
  ---------------
  0 0 0 0 0 0 0 0   -> Represents 2^k which is 2^8 (k=8)
```

Two's Complement

- In a "k-bit" two's complement representation of a number:
  1. A positive integer is represented in its $(k-1)$-bit unsigned binary representation, padded with a "0" to its left

Example: if there are "5" 2-bit digits to represent a positive digit $\underbrace{0}_{k}\underbrace{1011}_{k-1}$

  2. The sum of a number and its additive inverse is 2^k

Example: $(26)_{10} = (00011010)_{8\text{ bit } 2\text{'s complement}}$

- First get the binary representation $= (11010)_2$

0	0	1	1	0	1	0
2^6	2^5	2^4	2^3	2^2	2^1	2^0

$26-16=10-8=2-2=0$

- That representation is 7-bits, the 8th bit would denote that this is positive
- If asking 8 bit rep.; $8=k$ so $k-1=7$
- $(26)_{10} = (\underbrace{0}_{k}\underbrace{0011010}_{k-1})_{8\text{ bit } 2\text{'s complement}}$

Example: $(-26)_{10} = (11100110)_{8\text{bit } 2\text{'s complement}}$

- Adding a binary number to its additive inverse should give 2^k

$$\begin{array}{r} 00011010 \\ +\ 11100110 \\ \hline 100000000 \end{array}$$

→ Denotes $(26)_{10}$ to Binary
→ Denotes 8bit 2's complement to $(-26)_{10}$
→ Represents 2^k which is 2^8 $(k=8)$

Two's Complement (Cont.)

Example : $(\underbrace{0}_{k}\underbrace{0101101}_{k-1})_{8\text{bit 2's complement}} = (+45)_{10}$

$$\begin{array}{ccccccc} \underline{0} & \underline{1} & \underline{0} & \underline{1} & \underline{1} & \underline{0} & \underline{1} \\ 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 \end{array} = 1\cdot 2^0 + 1\cdot 2^2 + 1\cdot 2^3 + 1\cdot 2^5 =$$

$$1 + 4 + 8 + 32 = (45)_{10}$$

- We know "k" is the sign of the number (k-th Digit = 8th Digit)
- k-1 is the number itself (k-1 = 8-1 = 7 Digits)

Example $(\underbrace{1}_{k}\underbrace{1101010}_{k-1})_{8\text{ bit 2's complement}} = (-22)_{10}$

X!!!

$$\boxed{\begin{array}{l} \begin{array}{ccccccc} \underline{1} & \underline{1} & \underline{0} & \underline{1} & \underline{0} & \underline{1} & \underline{0} \\ 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array} = 1\cdot 2^1 + 1\cdot 2^3 + 1\cdot 2^5 + 1\cdot 2^6 = \\ \qquad 2 + 8 + 32 + 64 = (106)_{10} \end{array}}$$

- K-th digit is 1 ; this Method is Invalid!!

$$\begin{array}{rl} \ 11101010 & \leftarrow (-x) \\ +\ 00010110 & \leftarrow (+x) \\ \hline 100000000 & \leftarrow 2^k = 2^8 \end{array}$$

We use (+x) to find the decimal/num. via conversion

$$\begin{array}{ccccc} \underline{1} & \underline{0} & \underline{1} & \underline{1} & \underline{0} \\ 16 & 8 & 4 & 2 & 1 \end{array} = 2 + 4 + 16 = (22)_{10}$$

-Remember $k \neq 0$ so number is negative

$(-22)_{10}$

Knowledge Check : $(\underbrace{1}_{k}\underbrace{1001101}_{k-1})_{8\text{ bit }2\text{'s comp.}} = (?)_{10}$

- $k \neq 0$ so number is negative

$$
\begin{array}{r l}
 & 11001101 \leftarrow (-x) \\
+ & 00110011 \leftarrow (+x) \\
\hline
 & 100000000 \leftarrow 2^k = 2^8
\end{array}
$$

$$
\begin{array}{cccccc}
\underline{1} & \underline{1} & \underline{0} & \underline{0} & \underline{1} & \underline{1} \\
32 & 16 & 8 & 4 & 2 & 1
\end{array} \leftarrow (+x)
$$

$= 1+2+16+32 = (51)_{10}$

$k \neq 0$ so $(11001101)_{8\text{ bit }2\text{comp}} = (-51)_{10}$

Knowledge Check: $(-48)_{10} = (?)_{8\text{ bit }2\text{'s comp}}$

Convert 48 to binary $(48 < 64 = 2^6)$

$$
\begin{array}{cccccc}
\underline{1} & \underline{1} & \underline{0} & \underline{0} & \underline{0} & \underline{0} \\
32 & 16 & 8 & 4 & 2 & 1
\end{array}
$$

$\rightarrow$ 6 digits so

$$
\begin{array}{r l}
 & 00110000 = (+48)_{8\text{ bit }2\text{comp}} \\
+ & 11010000 \rightarrow (-48)_{8\text{ bit }2\text{comp}} \\
\hline
 & 100000000 \rightarrow 2^k = 2^8 \quad (k=8)
\end{array}
$$

$48-32 = 16 - 16 = 0$

$(-48)_{10} = (11010000)_{8\text{ bit }2\text{comp}}$

Knowledge Check : $(\underbrace{1}\underbrace{101})_{4\text{ bit }2\text{'s comp}} = (?)_{10}$

$k=4$, kth digit $=1$, $k-1=3$

kth digit $=1 \neq 0$ so negative

$$
\begin{array}{r l}
 & 1101 \ (-x) \\
+ & 0011 \ (+x) \\
\hline
 & 10000 \ 2^4
\end{array}
$$

$$
\begin{array}{cccc}
0 & 0 & \underline{1} & \underline{1} \\
 & & 2 & 1
\end{array} = 2+1 = 3
$$

$(-3)_{10}$

Knowledge Check: Add 00101111 + 01001000

$$
\begin{array}{r}
00101111 \\
+\ 01001000 \\
\hline
01110111
\end{array}
$$

Knowledge Check: Convert $(\underbrace{1010}_{A}\underbrace{1111}_{F})_2 = (?)_{16}$

0000	0001	0010	0011	0100	0101	0110	0111
0	1	2	3	4	5	6	7

1000	1001	1010	1100	1110	1101	1011	1111
8	9	A	B	C	D	E	F

Knowledge Check: Represent in Binary $(345)_{10} = (?)_2$

$(345 < 512 < 2^9)$

1	0	1	0	1	1	0	0	1
256	128	64	32	16	8	4	2	1

$(345)_{10} = (101011001)_2$

$$
\begin{array}{r}
345 \\
-\ 256 \\
\hline
089 \\
-\ 64 \\
\hline
25 \\
-\ 16 \\
\hline
09
\end{array}
$$