THE UNIVERSITY OF TEXAS AT DALLAS

CS4341 Digital Logic & Computer Design

Lecture Notes 2

Omar Hamdy
Assistant Professor
Department of Computer Science

Review: Number Representation

Remember the two examples:

(Hello)$_{\text{English}}$ = (Salut)$_{\text{French}}$ = (สวัสดี)$_{\text{Thai}}$

They all mean the same thing but in different “language system”.

Different language systems may or may not share the same alphabet “symbols”

(Gift)$_{\text{English}}$ =  (Gift)$_{\text{German}}$ =  (Gift)$_{\text{Norwegian}}$ = 

Sometimes same words have different meanings in the different “language systems”

Review: Positional Number System

Binary System:

r = 2 or base 2

Digit set: {0, 1}

Octal System:

r = 8 or base 8

Digit set: {0, 1, 2, 3, 4, 5, 6, 7}

Hexadecimal System:

r = 16 or base 16

Digit set: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Review: Numerical Values

An integer N of length n and base r is represented as:

$(N)_{\boldsymbol{r}} = (b_{n-1}, b_{n-2}, \ldots, b_1, b_0)_{\boldsymbol{r}}$

And has a numerical value of:

$b_{n-1}\boldsymbol{r}^{n-1} + b_{n-2}\boldsymbol{r}^{n-2} + \ldots + b_1\boldsymbol{r}^1 + b_0\boldsymbol{r}^0$

Can be representation using the summation formula:

$$\sum_{i=0}^{n-1} b_i r^i$$

Conversion Between Bases

- From any base ***r*** to Decimal: use the numerical value polynomial

- From Decimal to base ***t***: use the repeated division method

- To convert $(N)_{10}$ to $(M)_t$, we repeatedly divide N by ***t***. The remainder after each step is one digit of the required base ***t***, starting from the right most digit.

Conversion Example

Convert $(37)_{10}$ to Binary

Division	Quotient	Remainder
37 / 2	18	1
18 / 2	9	0
9 / 2	4	1
4 / 2	2	0
2 / 2	1	0
1 / 2	0	1

Least significant bit

Most significant bit

Stop when 0

$(37)_{10} = (100101)_2$

Conversion Example

Convert $(422)_{10}$ to Hexadecimal

$(422)_{10} = (1A6)_{16}$

Division	Quotient	Remainder
422 / 16	26	6
26 / 16	1	A
1 / 16	0	1

Least significant bit

Most significant bit

Stop when 0

Binary, Octal & Hexadecimal Conversions

- Observation: Octal (8) is a power of 2 = 2^3. Hence, each octal digit can be represented in exactly 3 binary bits.
  - To convert from binary to octal, divide the word into groups of 3 starting from the LSB and convert each into an octal digit.
  - To convert from octal to binary, simply reverse the process
- Observation: Hexadecimal (16) is a power of 2 = 2^4. Hence, each hexadecimal digit can be represented in exactly 4 binary bits.
  - To convert from binary to hexadecimal, divide the word into groups of 4 starting from the LSB and convert each into a hexadecimal digit.
  - To convert from hexadecimal to binary, simply reverse the process

Conversion Example

Convert $(1101000101)_2$ to Octal

$1|101|000|101 = 1\ 5\ 0\ 5 = (1505)_8$

Convert $(12A7F)_{16}$ to Binary

$1\ 2\ A\ 7\ F = 0001\ 0010\ 1010\ 0111\ 1111 =$

$(10010101001111111)_2$

Famous Conversion Table

Decimal	Binary	Octal	Hex
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Important Property

The largest possible value of a number of length n-digits in base system r is:

$$r^n - 1$$

Representing Fractions

- A Number N_r can have a fraction part:

$$N_r = \underbrace{b_{n-1}b_{n-2} \ldots b_1 b_0}_{\text{Integer Part}} \,.\, \underbrace{b_{-1}\, b_{-2} \ldots b_{-m+1}\, b_{-m}}_{\text{Fraction Part}}$$

Radix Point

- The numerical value is calculated as:
  - $b_{n-1} \times r^{n-1} + \ldots + b_1 \times r + b_0 +$ (Integer part)
  - $b_{-1} \times r^{-1} + b_{-2} \times r^{-2} \ldots + b_{-m} \times r^{-m}$ (Fraction part)

Representing Fractions

$$N_r = \sum_{i=0}^{i=n-1} b_i \times r^i + \sum_{j=-1}^{j=-m} b_j \times r^j$$

Examples

$(2409.87)_{10} = 2\times10^3 + 4\times10^2 + 9 + 8\times10^{-1} + 7\times10^{-2}$

$(1101.1001)_2 =$

$2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-4} = 13.5625$

$(703.64)_8 =$

$7\times8^2 + 3 + 6\times8^{-1} + 4\times8^{-2} = 451.8125$

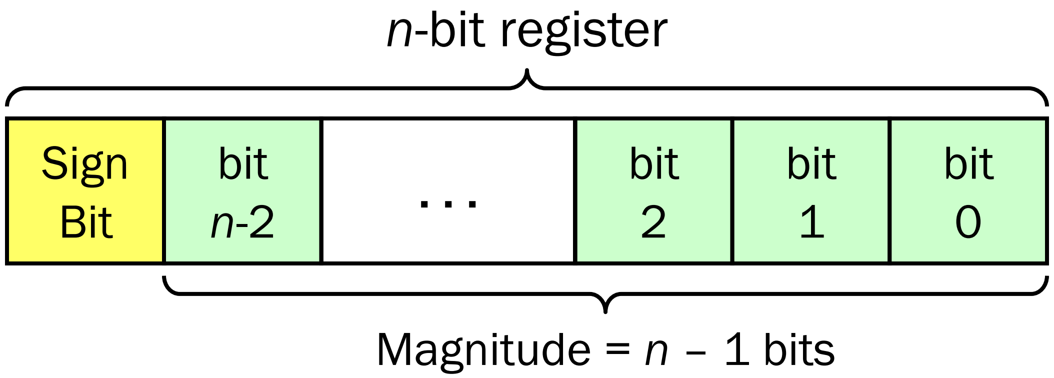
$(A1F.8)_{16} =$

$10\times16^2 + 16 + 15 + 8\times16^{-1} = 2591.5$

Signed Numbers

- *To carry out the various arithmetic operations, we need numbers as positive or negative.*
- *Three notations are commonly used:*
  - Sign-magnitude
  - (r-1)'s Complement
  - r's complement
- *In Binary, 2's complement is widely used.*

Sign-Magnitude Representation



- *Independent representation of sign and magnitude.*
- *MSB is the sign bit: 0 means positive and 1 means negative*
- *Using n-bits, then largest represented magnitude is:*

$$2^{n-1} - 1$$

Sign-Magnitude Examples

Sign-magnitude representation of +27 using 8-bit register

0	0	0	1	1	0	1	1

Sign-magnitude representation of -27 using 8-bit register

1	0	0	1	1	0	1	1

Properties of Sign-Magnitude

- *Two representations of the zero: +0 and -0*
- *Symmetric range of represented positive and negative values*
  - For n-bit register, range is from $-(2^{n-1} – 1)$ to $+(2^{n-1} – 1)$
  - For example, using 8-bit register, range is -127 to +127
- *Hard to implement addition and subtraction*
  - Sign and magnitude parts must be processed separately
  - Sign bit should be examined to determine addition or subtraction
  - Increases the cost of the logic circuit

1's Complement Representation

- *Given a binary number N, the 1's complement of N is obtained by flipping each bit in N (0 becomes 1, and 1 becomes 0).*
- *Example: 1's complement of $(01101001)_2 = (10010110)_2$*
- *If N consists of n-bits then, 1's complement of N =* $\boldsymbol{(2^n - 1) - N}$

2's Complement Representation

- *Used in most computers today*
- *Definition: given a binary number N, the 2's complement of N =1's complement + 1*
- *Example: 1's complement of* $(01101001)_2 = (10010110)_2$
- *2's complement of* $(01101001)_2 = (10010110 + 1)_2 = (10010111)_2$
- *If N consists of n-bits then, 2's complement of N* = $\mathbf{2^n - N}$

2’s Complement Example

starting value	00100100_2 = +36
step1: reverse the bits (1's complement)	11011011_2
step 2: add 1 to the value from step 1	+ 1_2
sum = 2's complement representation	11011100_2 = -36

2's Complement Representation

- *Another way to obtain the 2's complement: Starting at the least significant 1:*
- Leave all the 0s to its right unchanged
- Complement all the bits to its left

```
Binary Value
= 00100 1 00
2's Complement
= 11011 1 00
```

Ranges for Unsigned and Signed Numbers

- For n-bit unsigned integers: Range is 0 to $(2^n - 1)$
- For n-bit signed integers: Range is -2^{n-1} to $(2^{n-1} - 1)$:
  - Positive range: 0 to $(2^{n-1} - 1)$
  - Negative range: -2^{n-1} to -1

Note: in 2's Complement, there is only one zero: 2's complement of 0 = 0

Unsigned and Signed Values

Positive numbers:

- Signed value = Unsigned value

Negative numbers:

- To obtain the signed value, we assign a negative weight to MSB and add to the rest of the bit values.

$$N = b_{n-1} \times (-2^{n-1}) + b_{n-2} \times 2^{n-2} + \ldots + b_0 \times 2^0$$

Example

Find the numerical value of signed number $(10110100)_2$

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

- Value = -128 + 32 + 16 + 4 = -76
- If we store this number in 16-bit memory instead, would the numerical value change?

Example

Find the signed and unsigned ranges in 4-bit binary representation

- Unsigned: 0000_2 to 1111_2 = 0_{10} - 15_{10}
- Signed (using 2’s Complement):
  - Positive Range: 0111_2 ($+7_{10}$) to 0000_2 (0_{10})
  - Negative Range: 1000_2 (-8_{10}) to 1111_2(-1_{10})

Ranges for Unsigned and Signed Numbers

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	+1
00000010	2	+2
. . .	. . .	. . .
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
. . .	. . .	. . .
11111110	254	-2
11111111	255	-1

To Do List

- Review lecture notes
- Read chapter 1 until 1.5