

Lecture #2 – Number Representation

2007-01-19

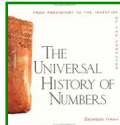
There is one handout today at the front and back of the room!



Lecturer SOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

Great book ⇒
The Universal History of Numbers
 by Georges Ifrah



Putting it all in perspective...

“If the automobile had followed the same development cycle as the computer,

– Robert X. Cringely



Decimal Numbers: Base 10

Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Example:

3271 =
 $(3 \times 10^3) + (2 \times 10^2) + (7 \times 10^1) + (1 \times 10^0)$



Numbers: positional notation

• Number Base B ⇒ B symbols per digit:

- Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Base 2 (Binary): 0, 1

• Number representation:

- $d_{31}d_{30} \dots d_1d_0$ is a 32 digit number
- value = $d_{31} \times B^{31} + d_{30} \times B^{30} + \dots + d_1 \times B^1 + d_0 \times B^0$

• Binary: 0,1 (In binary digits called “bits”)

• $0b11010 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
 $= 16 + 8 + 2$
 #s often written = 26



0b... • Here 5 digit binary # turns into a 2 digit decimal #

• Can we find a base that converts to binary easily?



Hexadecimal Numbers: Base 16

- Hexadecimal:
 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
 - Normal digits + 6 more from the alphabet
 - In C, written as 0x... (e.g., 0xFAB5)
- Conversion: Binary ⇔ Hex
 - 1 hex digit represents 16 decimal values
 - 4 binary digits represent 16 decimal values
 - ⇒ 1 hex digit replaces 4 binary digits
- One hex digit is a “nibble”. Two is a “byte”
- Example:
 - 1010 1100 0011 (binary) = 0x_____?



Decimal vs. Hexadecimal vs. Binary

Examples:	00 0 0000
	01 1 0001
1010 1100 0011 (binary)	02 2 0010
= 0xAC3	03 3 0011
	04 4 0100
10111 (binary)	05 5 0101
= 0001 0111 (binary)	06 6 0110
= 0x17	07 7 0111
	08 8 1000
0x3F9	09 9 1001
= 11 1111 1001 (binary)	10 A 1010
	11 B 1011
	12 C 1100
	13 D 1101
	14 E 1110
	15 F 1111

How do we convert between hex and Decimal?

MEMORIZE!



Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta

physics.nist.gov/cuu/Units/binary.html

- Common use prefixes (all SI, except K [= k in SI])

Name	Abbr	Factor	SI size
Kilo	K	$2^{10} = 1,024$	$10^3 = 1,000$
Mega	M	$2^{20} = 1,048,576$	$10^6 = 1,000,000$
Giga	G	$2^{30} = 1,073,741,824$	$10^9 = 1,000,000,000$
Tera	T	$2^{40} = 1,099,511,627,776$	$10^{12} = 1,000,000,000,000$
Peta	P	$2^{50} = 1,125,899,906,842,624$	$10^{15} = 1,000,000,000,000,000$
Exa	E	$2^{60} = 1,152,921,504,606,846,976$	$10^{18} = 1,000,000,000,000,000,000$
Zetta	Z	$2^{70} = 1,180,591,620,717,411,303,424$	$10^{21} = 1,000,000,000,000,000,000,000$
Yotta	Y	$2^{80} = 1,208,925,819,614,629,174,706,176$	$10^{24} = 1,000,000,000,000,000,000,000,000$

- Confusing! Common usage of “kilobyte” means 1024 bytes, but the “correct” SI value is 1000 bytes
- Hard Disk manufacturers & Telecommunications are the only computing groups that use SI factors, so what is advertised as a 30 GB drive will actually only hold about 28×2^{30} bytes, and a 1 Mbit/s connection transfers 10^6 bps.



CS61C L02 Number Representation (10)

Garcia, Spring 2007 © UCB

kibi, mebi, gibi, tebi, pebi, exbi, zebi, yobi

en.wikipedia.org/wiki/Binary_prefix

- New IEC Standard Prefixes [only to exbi officially]

Name	Abbr	Factor
kibi	Ki	$2^{10} = 1,024$
mebi	Mi	$2^{20} = 1,048,576$
gibi	Gi	$2^{30} = 1,073,741,824$
tebi	Ti	$2^{40} = 1,099,511,627,776$
pebi	Pi	$2^{50} = 1,125,899,906,842,624$
exbi	Ei	$2^{60} = 1,152,921,504,606,846,976$
zebi	Zi	$2^{70} = 1,180,591,620,717,411,303,424$
yobi	Yi	$2^{80} = 1,208,925,819,614,629,174,706,176$

As of this writing, this proposal has yet to gain widespread use...

- International Electrotechnical Commission (IEC) in 1999 introduced these to specify binary quantities.
 - Names come from shortened versions of the original SI prefixes (same pronunciation) and *bi* is short for “binary”, but pronounced “bee” :-)
 - Now SI prefixes only have their base-10 meaning and never have a base-2 meaning.



CS61C L02 Number Representation (11)

Garcia, Spring 2007 © UCB

The way to remember #s

- What is 2^{34} ? How many bits addresses (i.e., what's $\text{ceil} \log_2 = \lg$ of) 2.5 TiB?

- Answer! 2^{XY} means...

X=0 \Rightarrow ---	Y=0 \Rightarrow 1
X=1 \Rightarrow kibi $\sim 10^3$	Y=1 \Rightarrow 2
X=2 \Rightarrow mebi $\sim 10^6$	Y=2 \Rightarrow 4
X=3 \Rightarrow gibi $\sim 10^9$	Y=3 \Rightarrow 8
X=4 \Rightarrow tebi $\sim 10^{12}$	Y=4 \Rightarrow 16
X=5 \Rightarrow pebi $\sim 10^{15}$	Y=5 \Rightarrow 32
X=6 \Rightarrow exbi $\sim 10^{18}$	Y=6 \Rightarrow 64
X=7 \Rightarrow zebi $\sim 10^{21}$	Y=7 \Rightarrow 128
X=8 \Rightarrow yobi $\sim 10^{24}$	Y=8 \Rightarrow 256
	Y=9 \Rightarrow 512



MEMORIZE!



CS61C L02 Number Representation (12)

Garcia, Spring 2007 © UCB

What to do with representations of numbers?

- Just what we do with numbers!

- Add them $1 \ 1$
- Subtract them $1 \ 0 \ 1 \ 0$
- Multiply them $+ \ 0 \ 1 \ 1 \ 1$
- Divide them
- Compare them

- Example: $10 + 7 = 17$

- ...so simple to add in binary that we can build circuits to do it!
- subtraction just as you would in decimal
- Comparison: How do you tell if $X > Y$?



CS61C L02 Number Representation (13)

Garcia, Spring 2007 © UCB

Which base do we use?

- **Decimal:** great for humans, especially when doing arithmetic

- **Hex:** if human looking at long strings of binary numbers, its much easier to convert to hex and look 4 bits/symbol

- Terrible for arithmetic on paper

- **Binary:** what computers use; you will learn how computers do +, -, *, /

- To a computer, numbers always binary
- Regardless of how number is written:
- $32_{\text{ten}} = 32_{10} = 0x20 = 100000_2 = 0b100000$
- Use subscripts “ten”, “hex”, “two” in book, slides when might be confusing




CS61C L02 Number Representation (14)

Garcia, Spring 2007 © UCB

BIG IDEA: Bits can represent anything!!

- Characters?

- 26 letters \Rightarrow 5 bits ($2^5 = 32$)
- upper/lower case + punctuation \Rightarrow 7 bits (in 8) (“ASCII”)
- standard code to cover all the world’s languages \Rightarrow 8,16,32 bits (“Unicode”)  www.unicode.com

- Logical values?

- $0 \Rightarrow$ False, $1 \Rightarrow$ True

- colors ? Ex: Red (00) Green (01) Blue (11)

- locations / addresses? commands?

- **MEMORIZE:** N bits \Leftrightarrow at most 2^N things



CS61C L02 Number Representation (15)

Garcia, Spring 2007 © UCB

How to Represent Negative Numbers?

- So far, **unsigned numbers**
- Obvious solution: define leftmost bit to be sign!
 - $0 \Rightarrow +, 1 \Rightarrow -$
 - Rest of bits can be numerical value of number
- Representation called **sign and magnitude**
- MIPS uses 32-bit integers. $+1_{ten}$ would be:

0000 0000 0000 0000 0000 0000 0000 0001
- And -1_{ten} in sign and magnitude would be:

1000 0000 0000 0000 0000 0000 0000 0001



CS61C L02 Number Representation (16)

Garcia, Spring 2007 © UC

Shortcomings of sign and magnitude?

- Arithmetic circuit complicated
 - Special steps depending whether signs are the same or not
- Also, **two zeros**
 - $0x00000000 = +0_{ten}$
 - $0x80000000 = -0_{ten}$
 - What would two 0s mean for programming?
- Therefore sign and magnitude abandoned

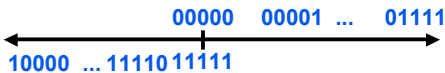


CS61C L02 Number Representation (17)

Garcia, Spring 2007 © UC

Another try: complement the bits

- Example: $7_{10} = 00111_2$ $-7_{10} = 11000_2$
- Called **One's Complement**
- Note: positive numbers have leading 0s, negative numbers have leading 1s.



- What is -00000 ? Answer: 11111
- How many positive numbers in N bits?



How many negative numbers?

CS61C L02 Number Representation (19)

Garcia, Spring 2007 © UC

Shortcomings of One's complement?

- Arithmetic still a somewhat complicated.
- Still two zeros
 - $0x00000000 = +0_{ten}$
 - $0xFFFFFFFF = -0_{ten}$
- Although used for awhile on some computer products, one's complement was eventually abandoned because another solution was better.



CS61C L02 Number Representation (20)

Garcia, Spring 2007 © UC

Standard Negative Number Representation

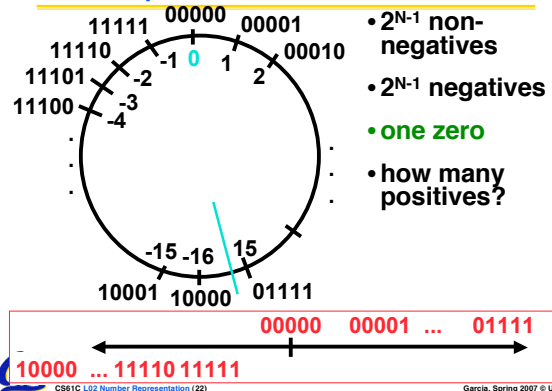
- What is result for unsigned numbers if tried to subtract large number from a small one?
 - Would try to borrow from string of leading 0s, so result would have a string of leading 1s
 - $3 - 4 \Rightarrow 00...0011 - 00...0100 = 11...1111$
 - With no obvious better alternative, pick representation that made the hardware simple
 - As with sign and magnitude, leading 0s \Rightarrow positive, leading 1s \Rightarrow negative
 - $000000...xxx$ is ≥ 0 , $111111...xxx$ is < 0
 - except $1...1111$ is -1 , not -0 (as in sign & mag.)
- This representation is **Two's Complement**



CS61C L02 Number Representation (21)

Garcia, Spring 2007 © UC

2's Complement Number "line": N = 5



CS61C L02 Number Representation (22)

Garcia, Spring 2007 © UC

Two's Complement for N=32

0000 ... 0000 0000 0000	$_{two} =$	0	$_{ten}$
0000 ... 0000 0000 0001	$_{two} =$	1	$_{ten}$
0000 ... 0000 0000 0010	$_{two} =$	2	$_{ten}$
...			
0111 ... 1111 1111 1101	$_{two} =$	2,147,483,645	$_{ten}$
0111 ... 1111 1111 1110	$_{two} =$	2,147,483,646	$_{ten}$
0111 ... 1111 1111 1111	$_{two} =$	2,147,483,647	$_{ten}$
1000 ... 0000 0000 0000	$_{two} =$	-2,147,483,648	$_{ten}$
1000 ... 0000 0000 0001	$_{two} =$	-2,147,483,647	$_{ten}$
1000 ... 0000 0000 0010	$_{two} =$	-2,147,483,646	$_{ten}$
...			
1111 ... 1111 1111 1101	$_{two} =$	-3	$_{ten}$
1111 ... 1111 1111 1110	$_{two} =$	-2	$_{ten}$
1111 ... 1111 1111 1111	$_{two} =$	-1	$_{ten}$

- One zero; 1st bit called **sign bit**
- 1 "extra" negative: no positive 2,147,483,648 $_{ten}$



CS61C L02 Number Representation (23)

Garcia, Spring 2007 © UC

Two's Complement Formula

- Can represent positive and negative numbers in terms of the bit value times a power of 2:

$$d_{31} \times (-2^{31}) + d_{30} \times 2^{30} + \dots + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0$$

- Example: 1101 $_{two}$

$$\begin{aligned} &= 1x(-2^3) + 1x2^2 + 0x2^1 + 1x2^0 \\ &= -2^3 + 2^2 + 0 + 2^0 \\ &= -8 + 4 + 0 + 1 \\ &= -8 + 5 \\ &= -3_{ten} \end{aligned}$$



CS61C L02 Number Representation (24)

Garcia, Spring 2007 © UC

Two's Complement shortcut: Negation

*Check out www.cs.berkeley.edu/~dsw/twos-complement.html

- Change every 0 to 1 and 1 to 0 (invert or complement), then add 1 to the result
- Proof*: Sum of number and its (one's) complement must be 111...111 $_{two}$
However, 111...111 $_{two} = -1_{ten}$
Let $x' \Rightarrow$ one's complement representation of x
Then $x + x' = -1 \Rightarrow x + x' + 1 = 0 \Rightarrow -x = x' + 1$

- Example: -3 to +3 to -3

x:	1111 1111 1111 1111 1111 1111 1111 1101	$_{two}$
x':	0000 0000 0000 0000 0000 0000 0000 0010	$_{two}$
+1:	0000 0000 0000 0000 0000 0000 0000 0011	$_{two}$
():	1111 1111 1111 1111 1111 1111 1111 1100	$_{two}$
+1:	1111 1111 1111 1111 1111 1111 1111 1101	$_{two}$



You should be able to do this in your head...

17 © UC

Two's comp. shortcut: Sign extension

- Convert 2's complement number rep. using n bits to more than n bits
- Simply replicate the most significant bit (sign bit) of smaller to fill new bits
 - 2's comp. positive number has infinite 0s
 - 2's comp. negative number has infinite 1s
 - Binary representation hides leading bits; sign extension restores some of them
 - 16-bit -4 $_{ten}$ to 32-bit:

1111 1111 1111 1100 $_{two}$

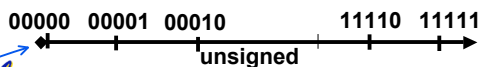


1111 1111 1111 1111 1111 1111 1111 1100 $_{two}$

Garcia, Spring 2007 © UC

What if too big?

- Binary bit patterns above are simply **representatives** of numbers. Strictly speaking they are called "numerals".
- Numbers really have an ∞ number of digits
 - with almost all being same (00...0 or 11...1) except for a few of the rightmost digits
 - Just don't normally show leading digits
- If result of add (or -, *, /) cannot be represented by these rightmost HW bits, **overflow** is said to have occurred.

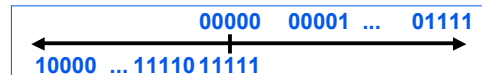


CS61C L02 Number Representation (27)

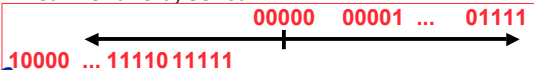
Garcia, Spring 2007 © UC

Number summary...

- We represent "things" in computers as particular bit patterns: $N \text{ bits} \Rightarrow 2^N$
- Decimal for human calculations, binary for computers, hex to write binary more easily
- 1's complement - mostly abandoned



- 2's complement universal in computing: cannot avoid, so learn



Overflow: numbers ∞ ; computers finite, errors!

CS61C L02 Number Representation (29)

Garcia, Spring 2007 © UC