

Digital Systems and Binary Numbers

EECE 256
Dr. Sidney Fels
Steven Oldridge

Topics

- Number systems
 - Decimal, Binary, Octal, Hex and more
- Complements
 - arithmetic with complements
- Signed binary numbers and codes
 - 2 weeks on, 1 week off
- Binary storage

8/23/10 (c) S. Fels, since 2010 2

Number systems

- Everything in the computer must be represented with a number
 - analog values converted to number
 - all arithmetic done with numbers
- What kind of numbers? What are numbers?
 - let's look at our friend: decimal numbers first

8/23/10 (c) S. Fels, since 2010 3

Decimal numbers

- 5984 =
- 1.298 =
- -6.45 =
- 3.14159.... =

8/23/10

(c) S. Fels, since 2010

4

Decimal numbers – base 10

- $5984 = 5 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 4 \times 10^0$
- $1.298 = 1 \times 10^0 + 2 \times 10^{-1} + 9 \times 10^{-2} + 8 \times 10^{-3}$
- $-6.45 = (\text{negative?}) 6 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
- $3.14159\dots = ??$

8/23/10

(c) S. Fels, since 2010

5

Binary numbers – base 2

- $(1011)_2 =$
- $(1.1011)_2 =$
- $(-100.1)_2 =$
- $(102.1)_2 =$

8/23/10

(c) S. Fels, since 2010

6

Binary numbers – base 2

- $(1011)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = (11)_{10}$
- $(1.1011)_2 = 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} + 1x2^{-4}$
 $= 1+(1/2)+(0/4)+(1/8)+(1/16) = (1.6875)_{10}$
- $(-100.1)_2 = (\text{negative?}) 1x2^2 + 0x2^1 + 0x2^0 + 0x2^{-1}$
 $= (-4.5)_{10}$
- $(102.1)_2 = ??$ – doesn't exist; can only be 0 or 1

8/23/10 Numbers are called bits and take on value of either 0 or 1 7

What about other bases?

- base 8 – called octal
 – $(725)_8 = 7x8^2 + 2*8^1 + 5*8^0 = (448)_{10} + (16)_{10} + (5)_{10}$
 $= (469)_{10}$
- base 16 – called hexadecimal
 – what to do about running out of symbols?
 - use: A = 10, B=11, C=12, D=13, E=14, F=15
 – $(FAD1)_{16} = (15)_{10}x16^3 + (10)_{10}x16^2 + (13)_{10}x16^1 + (1)_{10}x16^0$
 $= (61,440)_{10} + (2,560)_{10} + (208)_{10} + (1)_{10} = (64209)_{10}$

8/23/10 (c) S. Fels, since 2010 8

What about other bases?

- base 8 – called octal
 – $(725)_8 = 7x8^2 + 2*8^1 + 5*8^0 = (448)_{10} + (16)_{10} + (5)_{10}$
 $= (469)_{10}$
- base 16 – called hexadecimal
 – what to do about running out of symbols?
 - use: A = 10, B=11, C=12, D=13, E=14, F=15
 – $(FAD1)_{16} = (15)_{10}x16^3 + (10)_{10}x16^2 + (13)_{10}x16^1 + (1)_{10}x16^0$
 $= (61,440)_{10} + (2,560)_{10} + (208)_{10} + (1)_{10} = (64209)_{10}$

8/23/10 (c) S. Fels, since 2010 9

Converting from one base to another

- divide by new base for integer (left of point)
- multiply by base for fractions (right of point)
- $(41)_{10} = (?)_2$
- $(0.6875)_{10} = (?)_2$
- $(255)_{10} = (?)_8$
- $(357)_8 = (?)_2$

8/23/10

(c) S. Fels, since 2010

10

And a few simple tricks...

- base 2 -> base 8
 - group in 3 bits
- base 2 -> base 16
 - group in 4 bits
- and vice versa
- $011101111 = 011\ 101\ 111 = (357)_8$
 $= 1110\ 1111 = (EF)_{16}$

8/23/10

(c) S. Fels, since 2010

11

A few more...

- Multiply by 10 in decimal
 - move decimal place to right one digit, fill with 0
- Divide by 10 in decimal
 - move decimal place to left one digit, discard fraction if only using integers
- Same for all the other bases:
 - base 2:
 - multiply by 2, shift to right (fill with 0)
 - divide by 2, shift to left (discard fraction if not needed)

8/23/10

(c) S. Fels, since 2010

12

Remember how to add and subtract?

- in decimal:
 - 316 + 59 =
 - 316 + 905 =
 - 316 - 59 =
 - 316 - 905 =
- require either carry or borrow
 - not so good with computer
 - and we need a negative representation
 - and what if we run out of digits?
- How do these work in binary?
 - adding can be made efficient
 - subtraction is not at all efficient if done this way
 - let's make subtraction the same as addition
 - complements do this

8/23/10

(c) S. Fels, since 2010

13

Subtracting with complements

- 10's complement; n digits
 - 10^n - number
 - 10's complement of 316 is:
 - $1000 - 316 = 684$
 - note: when number is 0, 10's complement is 0
- 9's complement; n digits
 - $(10^n - 1)$ - number
 - 9's complement of 316 is:
 - $999 - 316 = 683$

8/23/10

(c) S. Fels, since 2010

14

Now lets subtract with these

- $M - N$ (M is minuend, N is subtrahend)
- with 10's complement
 - add M to 10's complement of N
 - if $M \geq N$, sum produces carry which you can ignore and you are done!
 - if $M < N$, sum does NOT produce carry so result is negative and is in 10's complement form

8/23/10

(c) S. Fels, since 2010

15

Example with 10s complement

- **316-59 =**
 M = 316, N = 59; 10's complement of 059 is 941
 316+941 = 1257 -> **257** (ignore the carry)
- **316 - 905 =**
 M = 316, N = 905; 10's comp is 095
 316 + 095 = **411** (no carry -> negative, in 10's complement)
 411 converted back is **-589**
- Wasn't that easier?

8/23/10
(c) S. Fels, since 2010
16

Subtracting with 9's complement

- same as for 10s complement
 - but you need to add one if there's a carry
 - remember, 9s complement is 10s complement minus 1
 - this is called end-around carry

8/23/10
(c) S. Fels, since 2010
17

What's this have to do with binary numbers?

- Can do the same thing with binary numbers!
- 2s complement ($2^n - \text{number}$)
 1001 -> 10000 - 1001 -> **0111**
 100100 -> 1000000 - 100100 -> **011100**
- 1s complement ($2^n - \text{number} - 1$)
 1001 -> 1111 - 1001 -> **0110**
 100100 -> 111111 - 100100 -> **011011**
 (just invert the bits)

8/23/10
(c) S. Fels, since 2010
18

Subtract two binary numbers with 2's complement

- $100111100 - 000111011 = (316-59 \text{ again})$
 $000111011 \rightarrow 111000101$
 $100111100 + 111000101 = 1\ 100000001$
 $= 257 \text{ (ignore carry)}$
- $0100111100 - 1110001001 = (316-905 \text{ again})$
 $1110001001 \rightarrow 0001110111$
 $0100111100 + 0001110111 = 0110110011$
 (no carry)
 $0110110011 \rightarrow -1001001101 = -589$

8/23/10 (c) S. Fels, since 2010 19


Can do the same with 1s complement

- But, don't forget that the result is one less when there is an end-carry
- So, just add the end-carry to the result
 – called end-around carry

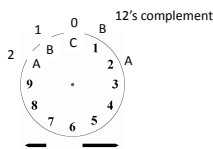
8/23/10 (c) S. Fels, since 2010 20

How does this work?

- Consider an odometer



- Or even a clock



8/23/10 (c) S. Fels, since 2010 21

What else can we do with bits?

- Bits can be used for codes:
 - n bits = ? codes

Table 1.1
Powers of Two

<i>n</i>	2^n	<i>n</i>	2^n	<i>n</i>	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

8/23/10 (c) S. Fels, since 2010 figure: © Pearson Ed., 2009 22

What else can we do with bits?

- things we want to encode:
 - negative sign
 - letters and characters
 - decimal numbers
 - redundancy
 - fractions
 - many other types of information

8/23/10 (c) S. Fels, since 2010 23

Let's deal with the -ve sign

- How could we do this in decimal?
 - special digit with 0 for positive, 1 for negative?
 - special digit with 0 for positive, 9 for negative?
 - how does this work with 10s complement?
 - how does this work with 9s complement?
- How to apply to binary?

8/23/10 (c) S. Fels, since 2010 24

Let's deal with the -ve sign

Table 1.3
Signed Binary Numbers

Decimal	Signed Magnitude
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
-0	1000
-1	1001
-2	1010
-3	1011
-4	1100
-5	1101
-6	1110
-7	1111
-8	---

first bit
0 = pos
1 = neg

8/23/10 (c) S. Fels, since 2010 figure: © Pearson Ed., 2009 25

Let's deal with the -ve sign

Table 1.3
Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	---	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	---	---

first bit
0 = pos
1 = neg

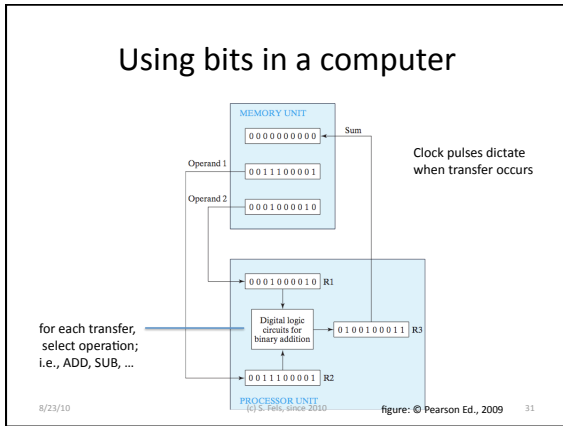
8/23/10 (c) S. Fels, since 2010 figure: © Pearson Ed., 2009 26

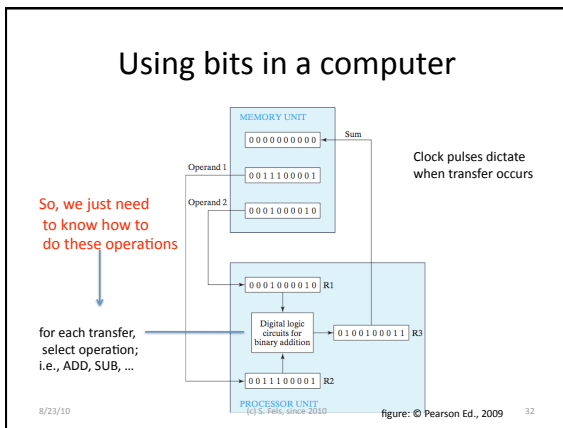
Coding Decimal Numbers

Table 1.4
Binary-Coded Decimal (BCD)

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

8/23/10 (c) S. Fels, since 2010 figure: © Pearson Ed., 2009 27





- ### Binary Logic and Operators
- Binary variables that take on 2 levels
 - Yes/No; True/False
 - these are implemented as voltages
 - 1 and 0
 - The most basic operations
 - AND (represented with a dot (.))
 - $x \bullet y$ is TRUE if and only if x is TRUE and y is TRUE
 - OR (represented with a plus (+))
 - $x + y$ is TRUE if and only if x is TRUE or y is TRUE or both are TRUE
 - NOT (represented with a prime (') or bar (—))
 - x' is TRUE if and only if x is FALSE
- 8/23/10 33

Truth Tables

- A Truth Table enumerates all combinations of inputs and the output of logical operation
 - very helpful for simple logic designs

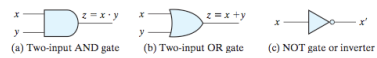
Table 1.8
Truth Tables of Logical Operations

AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

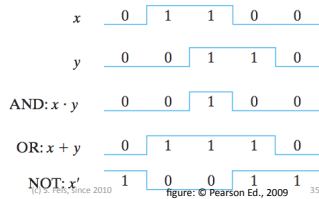
8/23/10 (c) S. Fels, since 2010 figure: © Pearson Ed., 2009 24

Logic Gates

- Can be represented as gates too



Timing signals
- don't forget that gates take time to propagate results



8/23/10 (c) S. Fels, since 2010 figure: © Pearson Ed., 2009 35

Summary

- positive numbers are represented by sequence of symbols
 - binary has two (0 and 1)
 - decimal has 10 (0-9)
 - octal has 8 (0-7)
 - hex has 16 (0-F)
- Subtraction easier to implement using complements
- Binary digits (bits) can be used to represent items:
 - +ve and -ve numbers, fractions, decimals, letters and so on
- Binary logic
 - AND, OR, NOT provide basic operations on binary variables
 - represent (and implement) as gates
- Foundation of all computing today

8/23/10 (c) S. Fels, since 2010 36

