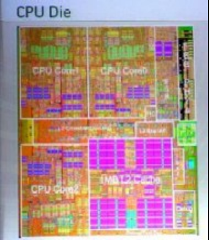
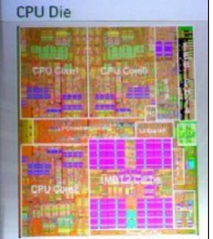


COMP 273

Digital Logic (Part 2)

Information Representation in Today's Computers

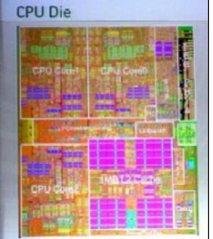
Prof. Joseph Vybihal



COMP 273
Introduction to Computer Systems

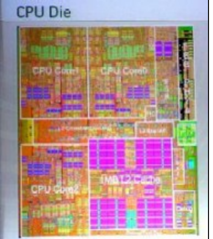
Announcements

- Ass#1 out this week



Readings

- Our textbook
    - Chapter 2.4 Sign & unsigned numbers
- Wikipedia
    - Binary number
    - Octal number
    - Hexadecimal number



COMP 273
Introduction to Computer Systems

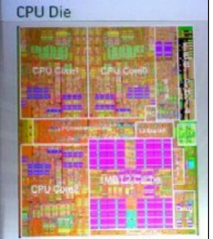
At Home

- What does this ASCII message say:
  - 01010111011001010110110001100011011011110
    110110101100101

- Compute the following binary equation:
  10110 + 11001 – 00001

- Start reading the Soul Of A New Machine

- Think of data representation as we did in class.



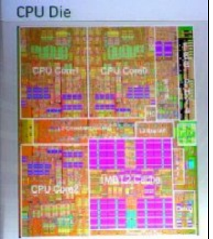
binary

Tabular encodings | binary numbers

ASCII | Machine Language | integers | reals

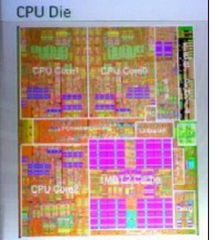
Tabular encodings are an ad-hock **mapping** of a string of bits to some meaning.
ASCII => symbol displayed on screen
ML => circuitry that has an effect
Need ROMS to convert to meaning.

Binary numbers are **true numbers** in the mathematical sense, supporting all normal operators like + - * / etc.
Do not need ROMS for meaning.



Computers Use

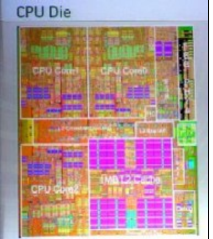
- Binary
  - Flags, numbers, strings and encodings
- Hexadecimal
  - A more convenient human readable form
- Octal
  - Historical
  - Current
    - Unix permissions
    - C, Perl special character escape codes



Part 1

Number Representations

Understanding number systems is important for this course since computers operate in number systems not commonly used by humans



Numerical Binary Representation

Counting in binary:

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10

Good to memorize

Another way is to add

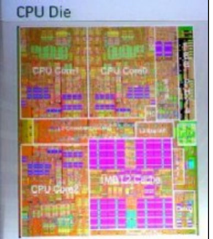
Numerical Binary Representation

System	Base	Digits
Decimal	10	0,1,2,3,4,5,6,7,8,9
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F (A=10, B=11, C=12, D=13, E=14, F=15)

All number systems use Positional Notation:

$$152_{10} = 1x10^2 + 5x10^1 + 2x10^0 = \sum_{i=0}^{n} a_i * base^i$$

COMP 273
Introduction to Computer Systems



COMP 273
Introduction to Computer Systems

Base Conversions can use the Positional Notation as a rule

- Decimal to Binary

$123_{10} = N_2 = 1111011_2$

123 /2	=	61 R 1
61/2	=	30 R 1
30/2	=	15 R 0
15/2	=	7 R 1
7/2	=	3 R 1
3/2	=	1 R 1
1/2	=	0 R 1

Read

Base Conversions can use the Positional Notation as a rule

- Decimal to Hex $53241_{10} = N_{16} = CFF9_{16}$

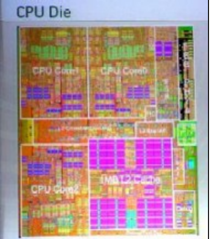
$$53241/16 = 3327 \text{ R } 9$$
$$3327/16 = 207 \text{ R } F$$
$$207/16 = 12 \text{ R } F$$
$$12/16 = 0 \text{ R } C$$

- Binary to Decimal $1011_2 = N_{10} = 11_{10}$

$$= 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 11_{10}$$

- Hex to Decimal $1AB_{16} = N_{10} = 427_{10}$

$$= 1x16^2 + Ax16^1 + Bx16^0$$
$$= 1x16^2 + 10x16^1 + 11x16^0 = 427_{10}$$

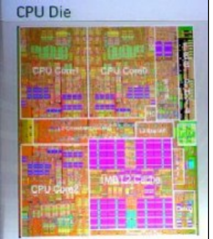


Binary to Hex Conversion

Notice that 1 nibble ≡ 1 hex digit (it works in both directions)

$0000_2 = 0_{16}$
$0001 = 1$
$0010 = 2$
$0011 = 3$
$0100 = 4$
:
:
$1111 = F$

What does this equal to?
$001001001111_2 = ?$
$F310_{16} = ?$



Binary and Octal Conversion

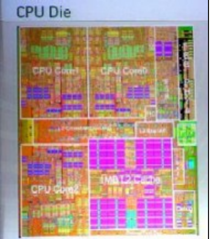
Notice that every 3 bits $\equiv$ 1 octal digit (it works in both directions)

$000_2 = 0_8$
$001 = 1$
$010 = 2$
$011 = 3$
$100 = 4$
$101 = 5$
$110 = 6$
$111 = 7$

What does this equal to?

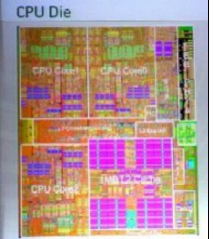
$100111010101_2 = ?$

$123_8 = ?$



Why so many representations?

- Humans are used to decimal
  - Convert binary to decimal… slow
  - Do fast binary conversions…
    - Bytes are 8 bits = 2 Hex digits
    - Since Binary to Hex conversion requires simple circuitry and since Hex is more readable to humans, the computer often auto converts binary to hex for error dumps
- Therefore Dec, Bin & Hex conversions
  - Octal was a cool fad…

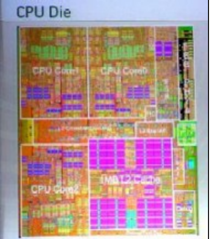


Why Octal?

- No reason these days other than educational
- Historically the PDP11, a famous powerful mini computer from the 80s was based on Octal
- Teaching of octal still occurs because of that machine

Part 2

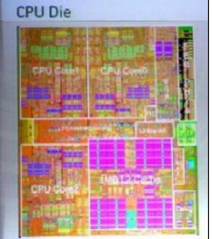
Binary Encodings



COMP 273
Introduction to Computer Systems

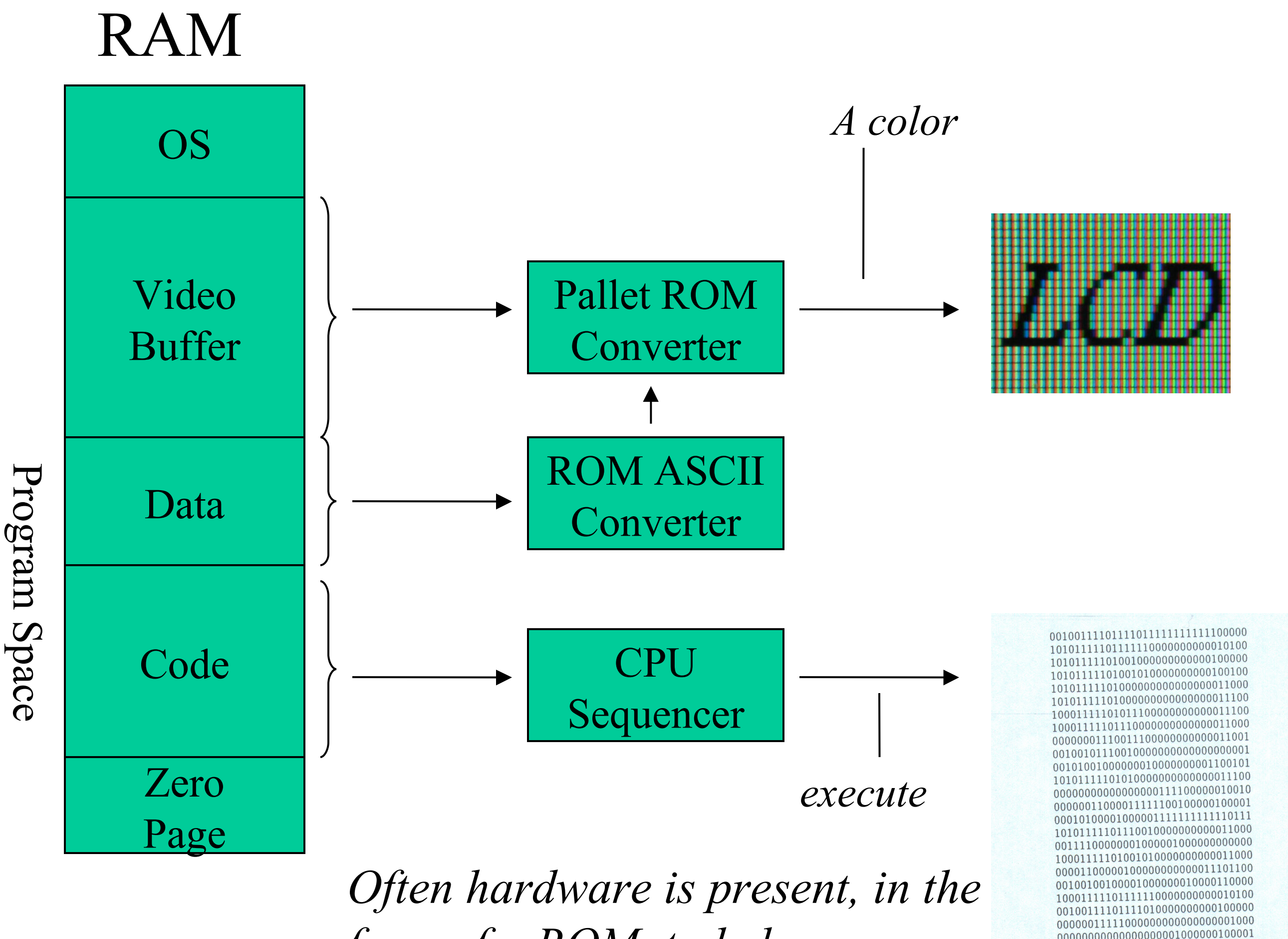
ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

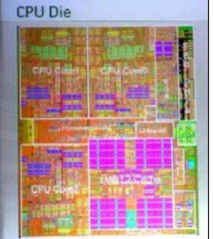


COMP 273
Introduction to Computer Systems

Memory to Device



Often hardware is present, in the form of a ROM, to help convert one representation into another.



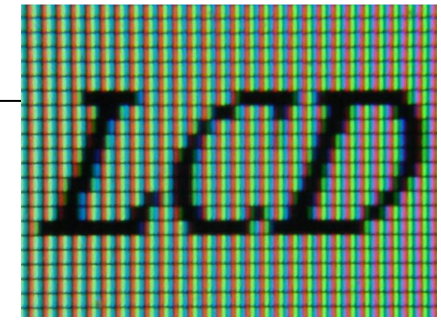
COMP 273
Introduction to Computer Systems

Tabular Mappings: Screen

1001110

Register with binary value

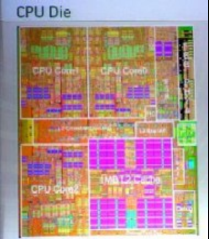
ROM
(ASCII Table)



Monitor

Circuitry to draw the letter (i.e. each wire lights up a pixel)

(x,y,color)



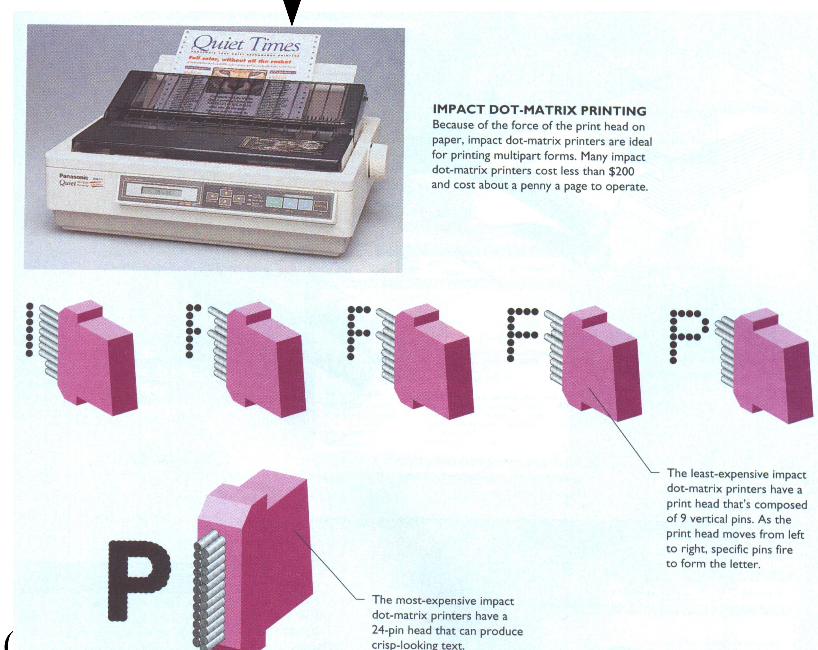
COMP 273
Introduction to Computer Systems

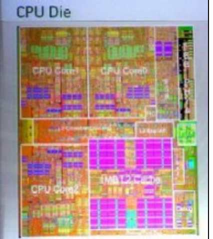
Tabular Mappings: Printer

1001110

Register with binary ASCII value

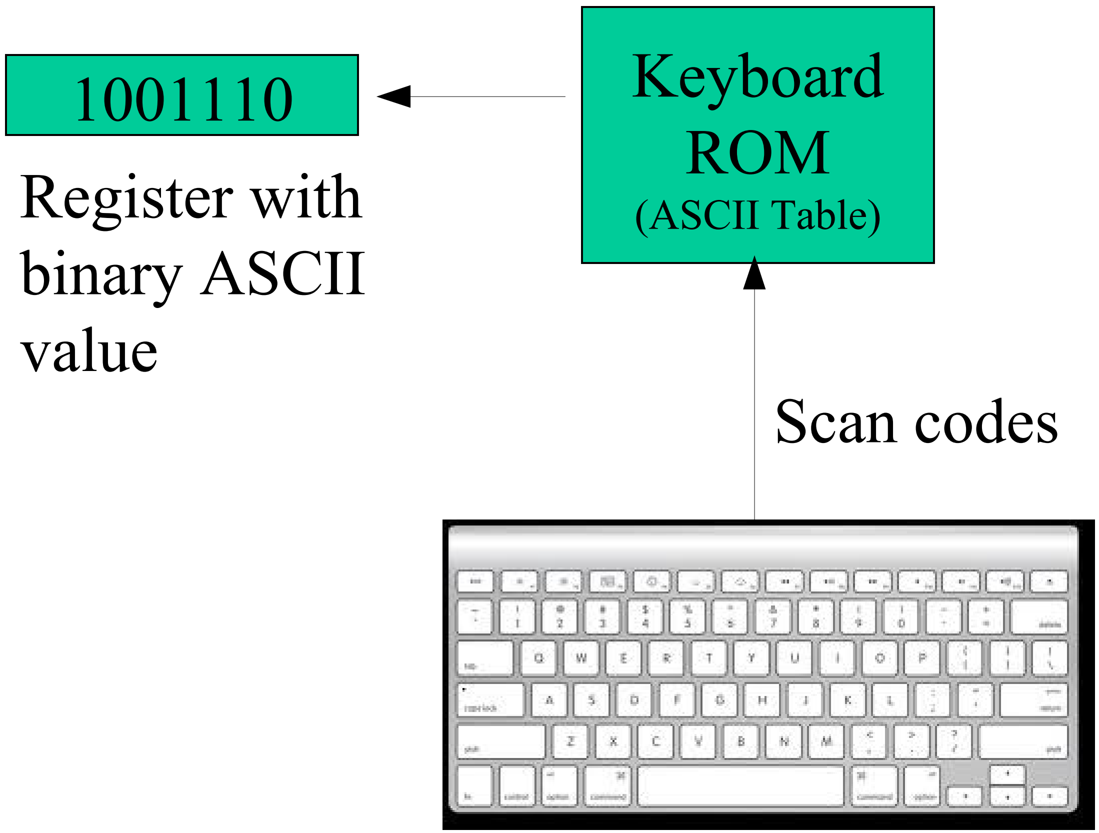
Printer ROM
(ASCII Table)

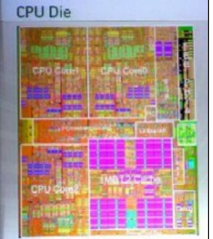




COMP 273
Introduction to Computer Systems

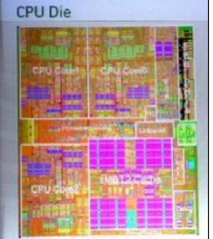
Tabular Mappings: Keyboard





Part 3

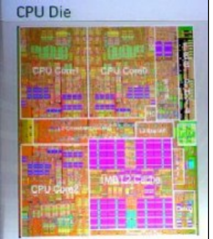
Data Representations & Mathematics



COMP 273
Introduction to Computer Systems

Data Types

- Data has 3 representations...
  - A logical description
    - How it truly looks (integer, real, char),
    - How it truly behaves:
      - Behaviour is defined by operations/operators
      - Operations are algorithms
  - A physical construction
    - Based on the logical definition
  - A circuit that implements the algorithms

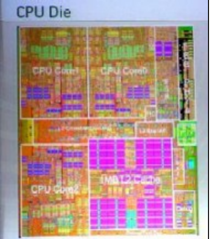


Basic Principles of Data

Imagining our own data

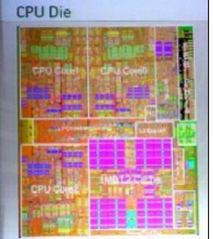
- First we need to imagine our data type:
  - Legal operators
  - A philosophical abstraction of what is being recorded (e.g. characters do not really exist)

- Second we need to determine how we want to represent the information in binary
  - Size in bits



Question

- How could we implement a string data type?



An example with issues

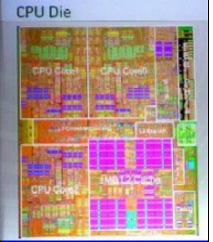
If bit size is 4 and we want to store integers:

0000
0001
0010
0011
0100
0101
0110
0111
:

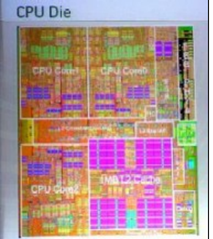
What are the next values?

<u>Issues</u>:

- Is there a limit?
  - What should happen at that limit?
- Overflow looks like?
- Negative looks like?



Binary Mathematics



COMP 273

Introduction to Computer Systems

Binary Addition

```
  1  <----  The carry
 1011
+0010
--------
 1101₂
```

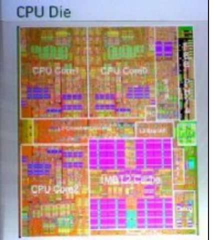
Addition in binary functions identically with decimal

$$\begin{array}{r} 257 \\ +\ 102 \\ \hline 359_{10} \end{array}$$

Carry past defined type size:

- A physical property constraint
- Called an *Arithmetic Overflow*, what about *signed overflow*?

```
      1   1
  1  [1|1|1|1]
  +  [1|0|1|0]
  ------------
?    [1|0|0|1]  <---- This is only remembered
```



COMP 273
Introduction to Computer Systems

Binary Subtraction

- To make computers easier to build…
    - X – Y = X + (-Y) = X + (2’s complement of Y)
- Two’s Complement Notation:
    - Conversion process…
        - Take Y
        - Flip Y’s bits
        - Then add 1
    - Now your value is in Two’s Complement
- Add

An Example

```
  00111
- 00101
----------
    ?
```

→ Convert to 2's comp.

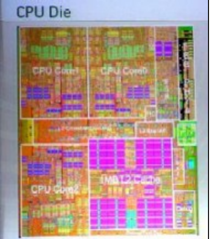
Take value: 00101
Flip bits: 11010
Add 1: 11011

Now finish the problem:

```
  00111
+11011
----------
  00010₂
```

Overflow (Expected) (Ignored)

00010_2 ← $7 - 5 = 2$



Base 10 Two's Complement Method

$-N = \text{Base}^{\text{size}} - N$

E.G.: $-12_{10} = 10^2 - 12 = 100 - 12 = 88_{10}$

$-N = 2^{\text{size}} - N$

$-27_{10} = 2^8 - 11011_2 =$

Carry in

```
  100000000
-  00011011
--------------
   11100101 = flip bits + 1
```

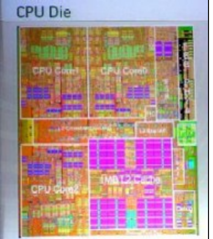
Want 15 - 12

Get 15 + 88

Gives 103

Drop carry= 3

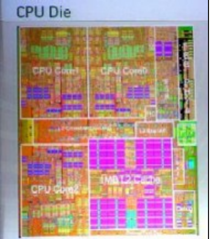
Properties:
- Unique 0
- MSB is sign bit
- -2^{n-1} to $+2^{n-1}$
- $-(-Y) = +Y$



Signed Binary Numbers using 2's Complement

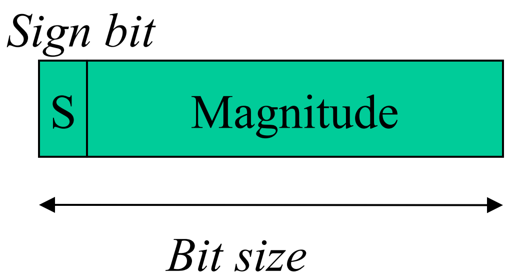
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two}$	=	0_{ten}
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two}$	=	1_{ten}
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two}$	=	2_{ten}
...		...
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two}$	=	$2,147,483,645_{ten}$
$0111\ 1111\ 1111\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

sign



Signed Number Representation

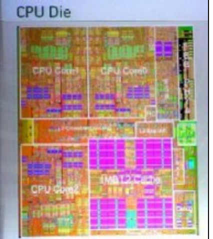
Sign & Magnitude Method



In other words:
8-bit +7 = 00000111
8-bit –7 = 10000111
The MSB is S

Notice problems with this?
00000000 = +0
10000000 = -0

This is not corrected for in hardware sometimes (simple calculators)



Basic Signed vs. 2's Complement

- Benefits…
  - Basic Signed
    - Easy to read
    - No pre-conversion needed
  - 2's Complement
    - Only one zero value
    - Auto subtracts when adding two numbers
      - Assuming we ignore the overflow...