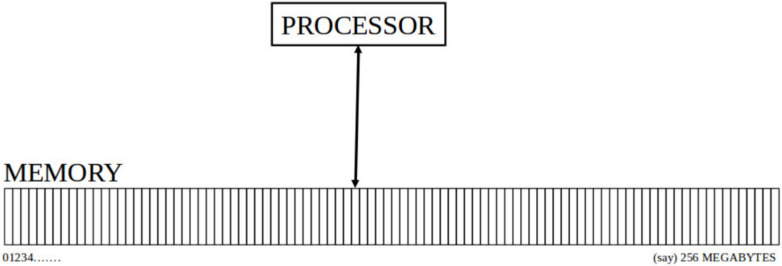
CS1100 – Introduction to Programming

Lecture 3

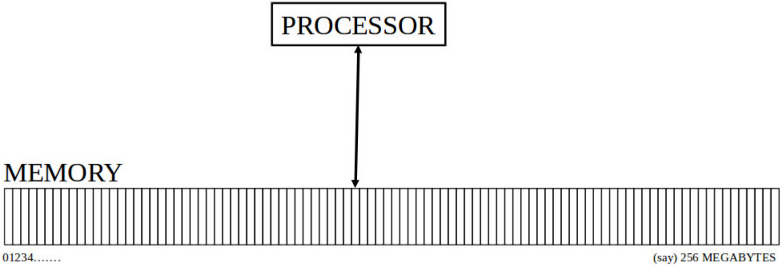
Instructor: Shweta Agrawal (shweta.a@cse.iitm.ac.in)

The Computing Machine



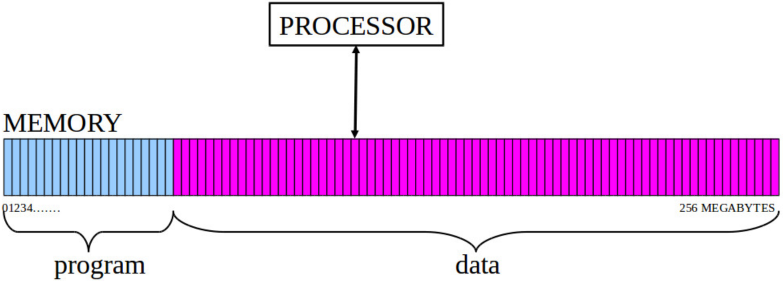
- The computer is made up of a processor and a memory.

The Computing Machine



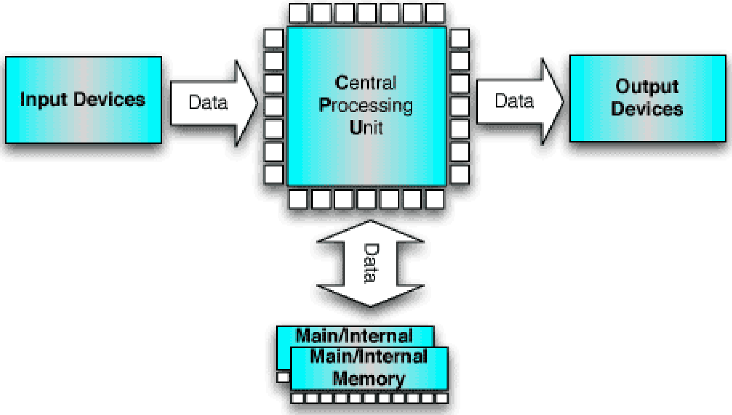
- The computer is made up of a processor and a memory.
- The memory can be thought of as a series of locations to store information.

The Computing Machine



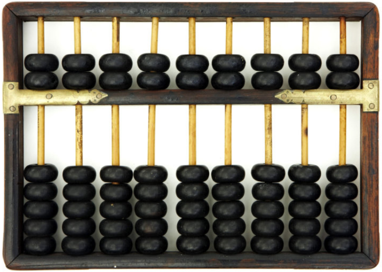
- A program is a sequence of instructions assembled for some given task.
- Most instructions operate on data.
- Some instructions control the flow of the operations.

The Computing Machine : von Neuman Architecture



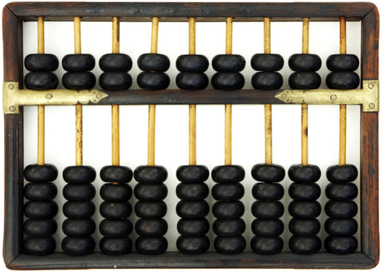
A brief look into the history...

From Abacus to Apple



- Counting frame.
- One of the earliest form of calculator.
- Still used by kids to do fast simple arithmetic.

From Abacus to Apple



- Counting frame.
- One of the earliest form of calculator.
- Still used by kids to do fast simple arithmetic.

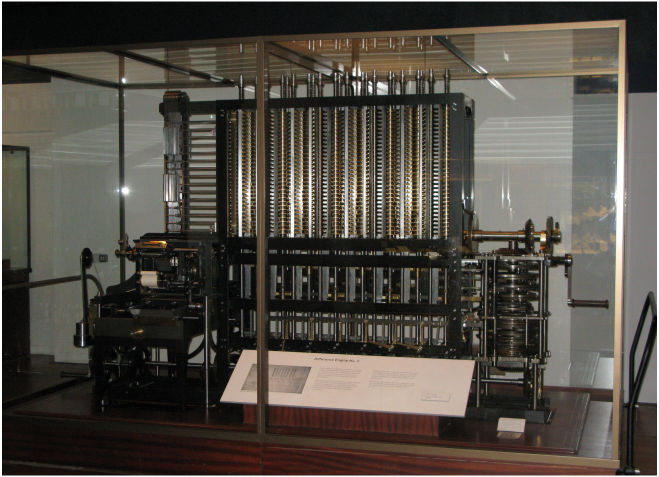
- Followed by mechanical calculators by B. Pascal (1642), G. W. Leibniz (1671).
  - Used cogs / interlocking gears.
  - Performed $+, -, *, / \sqrt{}$.
  - Leibniz is credited of creating the binary system.

Jaquard looms (1804)

Charles Babbage (1791–1871)

- Regarded as the "Father of Computer".
- Conceived of a machine that has all the parts of a modern computer, input, a memory, a processor, and an output (1850).

Difference Engine (1850)



Difference engine built from Babbage's design
(London Science Museum).

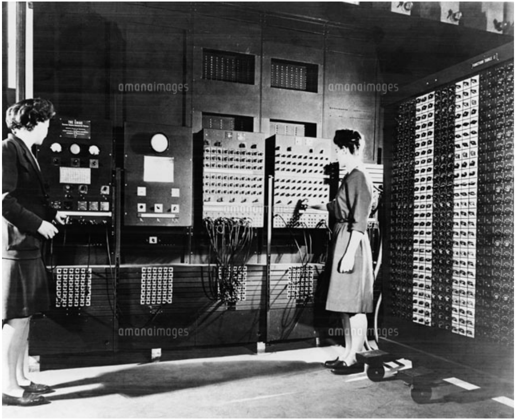
Ada Lovlace (1815–1852)

- “Wrote” the description of the mechanical computer of Babbage.
- Regarded as the first programmer ever.
- The programming language ADA is named after her.

Alan Turing (1912 – 1954)

- Father of Theoretical Computer Science (TCS) and Artificial Intelligence (AI).
- Turing machine – a model for a general purpose computer.
- Turing test – how intelligent is a machine?

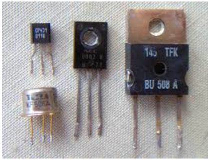
First Electronic Computer : ENIAC 1946



Electronic Numerical Integrator And Calculator.

- 50,000 vacuum tubes, diodes, relays, resistors, capacitors.
- 5 million hand-soldered joints.
- Weighed 27 tons.
- Covered $167m^2$ area.
- Consumed 150 kW of power.

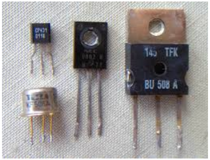
1946 – 1976



Transistors

Integrated Circuits

1946 – 1976

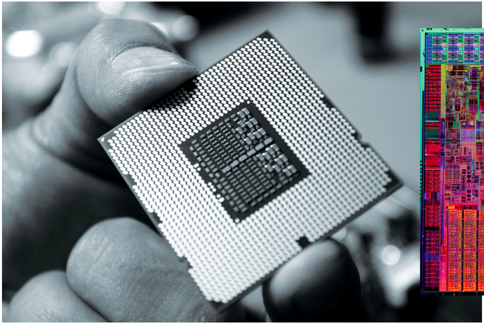


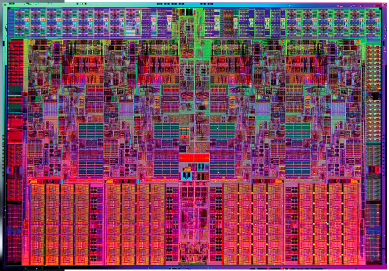
Transistors

Integrated Circuits

Apple Macintosh

Today's World : Core i7 Processor

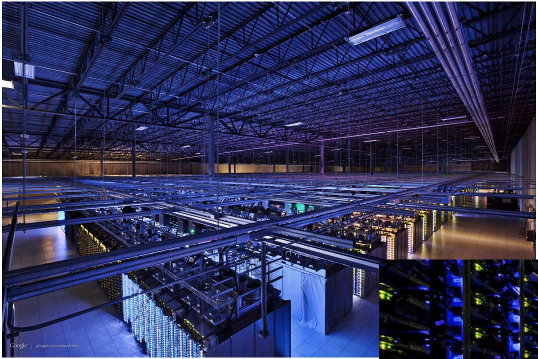


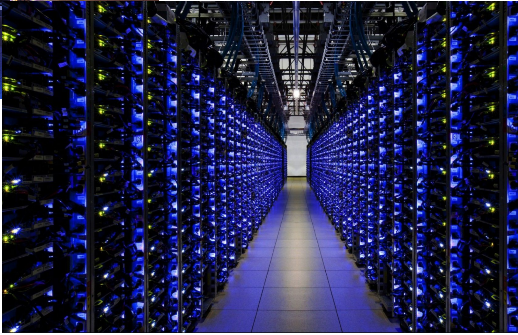


2008-15: Intel Core i7 Processor
Clock speed: > 2.5 GHz
No. of Transistors: $0.731 - 1.3B$
Doubles every two years (Moore's law!)
Technology: $45 - 22nm$ CMOS Area: $263 - 181mm^2$.
Nowadays: Multicore (as clock speed increased) with cooling units!

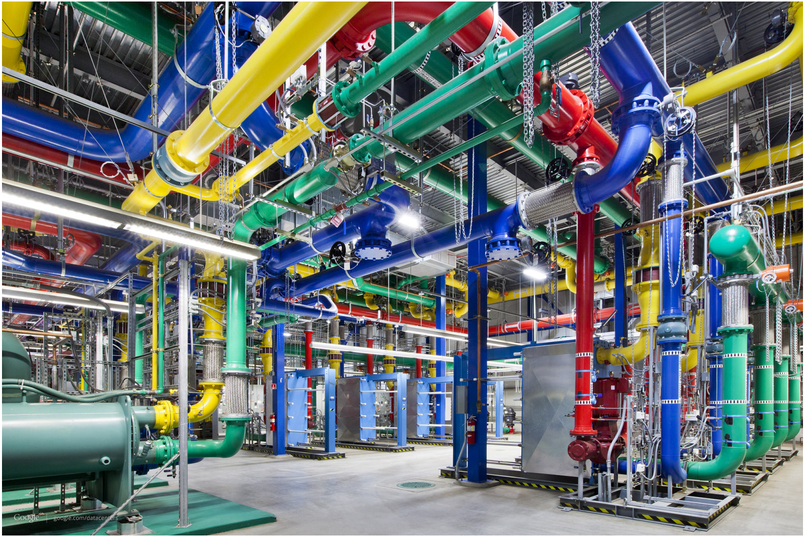
Modern computing devices

Data Centers: Processing/Storing Huge volume of data

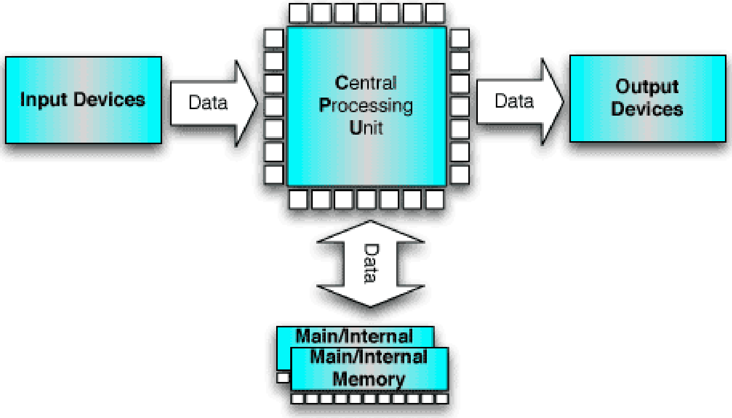




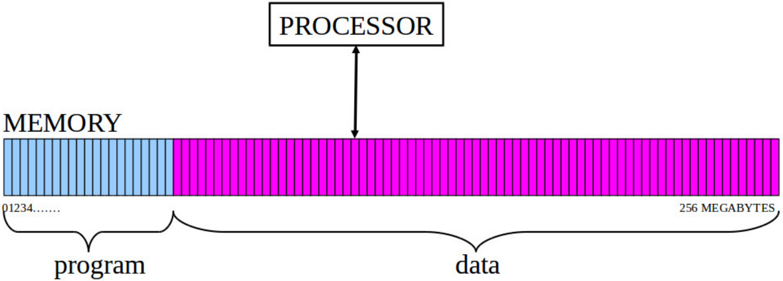
Even Cooling them is a big deal ...



The Computing Machine : von Neuman Architecture



The Computing Machine



- A program is a sequence of instructions assembled for some given task.
- Most instructions operate on data.
- Some instructions control the flow of the operations.

How does the computer represent data?

- To store : Numbers, text, graphics and images, video, audio, program instructions.

How does the computer represent data?

- To store : Numbers, text, graphics and images, video, audio, program instructions.
- In some way, all information is digitized - broken down into pieces and represented as numbers.

How does the computer represent data?

- To store : Numbers, text, graphics and images, video, audio, program instructions.
- In some way, all information is digitized - broken down into pieces and represented as numbers.
- Example : Representing Text Digitally.
  - Every character is stored as a number, including spaces, digits, and punctuation.
  - Corresponding upper and lower case letters are separate characters.

The ASCII table

American Standard Code for Information Interchange (ASCII).

The ASCII table

American Standard Code for Information Interchange (ASCII).

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

But, how does number get stored?

Number Systems.

- Decimal (base 10 - uses 10 symbols $\{0 \ldots 9\}$. Eg : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 . . ..

But, how does number get stored?

Number Systems.

- Decimal (base 10 - uses 10 symbols $\{0 \ldots 9\}$. Eg : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 ....
- Unary (base 1 - uses 1 symbol)
  Eg : 1, 11, 111, 1111, ...

But, how does number get stored?

Number Systems.

- Decimal (base 10 - uses 10 symbols $\{0 \ldots 9\}$. Eg : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 . . ..
- Unary (base 1 - uses 1 symbol)
  Eg : 1, 11, 111, 1111, ...
- Binary (base 2) – uses 2 symbols $\{0,1\}$)
  Eg : 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010 . . .

But, how does number get stored?

Number Systems.

- Decimal (base 10 - uses 10 symbols $\{0 \dots 9\}$. Eg : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 ....
- Unary (base 1 - uses 1 symbol)
  Eg : 1, 11, 111, 1111, ...
- Binary (base 2) – uses 2 symbols $\{0,1\}$)
  Eg : 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010 ...
- Octal (base 8 – uses 8 symbols $\{0 \dots 7\}$)
  Eg : 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13 ...

But, how does number get stored?

Number Systems.

- Decimal (base 10 - uses 10 symbols $\{0 \ldots 9\}$. Eg : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 . . . .
- Unary (base 1 - uses 1 symbol)
  Eg : 1, 11, 111, 1111, ...
- Binary (base 2) – uses 2 symbols {0,1})
  Eg : 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010 . . .
- Octal (base 8 – uses 8 symbols $\{0 \ldots 7\}$)
  Eg : 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13 . . .
- Hexadecimal(base 16 – uses A-F for 10-15)
  Eg : 0, 1, ..., 9, A, B, C, D, E, F, 10, 11, ... 19, 1A, 1B, ...

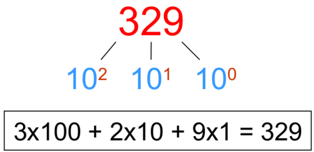
But, how does number get stored?

Number Systems.

- Decimal (base 10 - uses 10 symbols $\{0 \ldots 9\}$. Eg : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 ....
- Unary (base 1 - uses 1 symbol)
  Eg : 1, 11, 111, 1111, ...
- Binary (base 2) – uses 2 symbols {0,1})
  Eg : 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010 ...
- Octal (base 8 – uses 8 symbols $\{0 \ldots 7\}$)
  Eg : 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13 ...
- Hexadecimal(base 16 – uses A-F for 10-15)
  Eg : 0, 1, ..., 9, A, B, C, D, E, F, 10, 11, ... 19, 1A, 1B, ...

Quick Primer on Number System : Base n

Take every "digit" and multiply by increasing powers of n and add.



Quick Primer on Number System : Base n

Take every "digit" and multiply by increasing powers of n and add.

$$329$$

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

$$\boxed{3 \text{x} 100 + 2 \text{x} 10 + 9 \text{x} 1 = 329}$$

most significant $\quad 101 \quad$ *least significant*

$$2^2 \quad 2^1 \quad 2^0$$

$$\boxed{1 \text{x} 4 + 0 \text{x} 2 + 1 \text{x} 1 = 5}$$

Converting from Decimal to Binary

Conver the decimal number 39 to binary (base 2).

2	39	
2	19	+ Remainder 1
2	9	+ Remainder 1
2	4	+ Remainder 1
2	2	+ Remainder 0
2	1	+ Remainder 0
	0	+ Remainder 1

$$
\begin{aligned}
39 &= 2*19 + 1 \\
&= 2*(2*9 + 1) + 1 \\
&= 2^2*9 + 2^1*1 + 1 \\
&= 2^2*(2*4+1) + 2^1*1 + 1 \\
&= 2^3*4+2^2*1+ 2^1*1 + 1 \\
&= 2^3*(2*2+0)+2^2*1+ 2^1*1 + 1 \\
&= 2^4*2+2^3*0+ 2^2*1+ 2^1*1 + 1 \\
&= 2^4*(2*1+0) + \ldots \\
&= 2^5*1+2^4*0+2^3*0+2^2*1+ 2^1*1+ 1
\end{aligned}
$$

$$(100111)_2 = (1 \times 2^0) + (1 \times 2^1) + (1 \times 2^2) + (0 \times 2^3) + (0 \times 2^4) + (1 \times 2^5)$$

$$= (39)_{10}$$

Which Number System? Binary !

- Devices that store and process information are cheaper and more reliable if they have to represent only two states.

Which Number System? Binary !

- Devices that store and process information are cheaper and more reliable if they have to represent only two states.
- A single bit can represent two possible states, like a light bulb that is either on (1) or off (0). Hence representable by even voltage levels in wires.

Which Number System? Binary !

- Devices that store and process information are cheaper and more reliable if they have to represent only two states.
- A single bit can represent two possible states, like a light bulb that is either on (1) or off (0). Hence representable by even voltage levels in wires.
- The other number systems can be "encoded in" binary

Which Number System? Binary !

- Devices that store and process information are cheaper and more reliable if they have to represent only two states.
- A single bit can represent two possible states, like a light bulb that is either on (1) or off (0). Hence representable by even voltage levels in wires.
- The other number systems can be "encoded in" binary

Octal - 2 7 3 2 5 6 0 6

Binary - 010111011010101110000110

Hexadecimal - 5 13 10 11 8 6

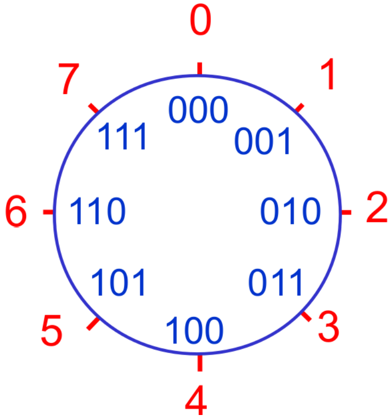
5 D A B 8 6

Representing values in Binary

If we have m bits, we can represent 2^m unique different values.

Representing values in Binary

If we have m bits, we can represent 2^m unique different values.
A useful circle :



Representing negative numbers

Sign Magnitude notation

- Use one bit for sign, others for magnitude of the number.

Representing negative numbers

Sign Magnitude notation

- Use one bit for sign, others for magnitude of the number.

			Sign Magn.
0	0	0	0
0	0	1	$+1$
0	1	0	$+2$
0	1	1	$+3$
1	0	0	0
1	0	1	-1
1	1	0	-2
1	1	1	-3

Representing negative numbers

Sign Magnitude notation

- Use one bit for sign, others for magnitude of the number.

			Sign Magn.
0	0	0	0
0	0	1	+1
0	1	0	+2
0	1	1	+3
1	0	0	0
1	0	1	-1
1	1	0	-2
1	1	1	-3

- using n bits: $-(2^{n-1}-1)\ldots(2^{n-1}-1)$.
- zero has two representations.

Representing negative numbers

Ones complement notation

- for a negative number n, represent the number by the bit complement of its binary rep. using k bits.

Representing negative numbers

Ones complement notation

- for a negative number n, represent the number by the bit complement of its binary rep. using k bits.

			Sign Magn.	Ones comp.
0	0	0	0	0
0	0	1	$+1$	$+1$
0	1	0	$+2$	$+2$
0	1	1	$+3$	$+3$
1	0	0	0	-3
1	0	1	-1	-2
1	1	0	-2	-1
1	1	1	-3	0

Representing negative numbers

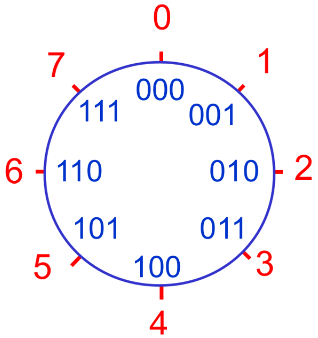
Ones complement notation

- for a negative number n, represent the number by the bit complement of its binary rep. using k bits.

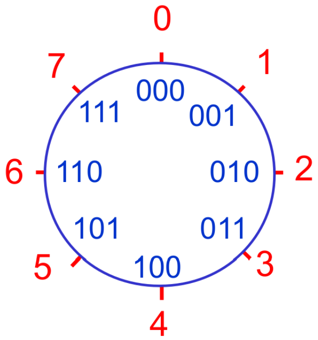
			Sign Magn.	Ones comp.
0	0	0	0	0
0	0	1	+1	+1
0	1	0	+2	+2
0	1	1	+3	+3
1	0	0	0	-3
1	0	1	-1	-2
1	1	0	-2	-1
1	1	1	-3	0

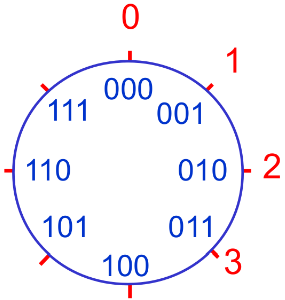
- using n bits: $-(2^{n-1} - 1) \ldots (2^{n-1} - 1)$.
- zero has two representations.
- not very widely used representation.

Representing negative numbers - A neat trick

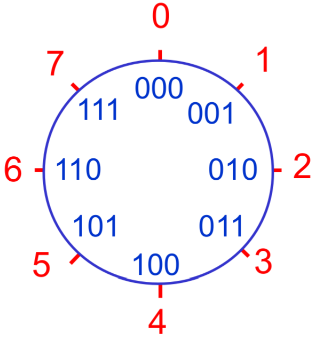


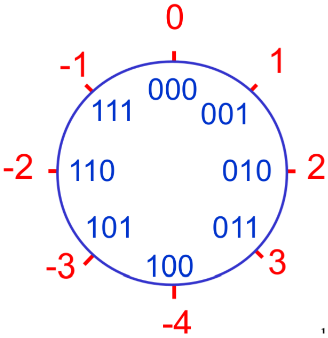
Representing negative numbers - A neat trick





Representing negative numbers - A neat trick





Representing negative numbers - A neat trick

Twos complement notation

- for a positive number n, represent the number by its binary rep. using k bits.
- for a negative number $-n$, represent the number as $2^k - n$.

Representing negative numbers - A neat trick

Twos complement notation

- for a positive number n, represent the number by its binary rep. using k bits.
- for a negative number $-n$, represent the number as $2^k - n$.

			Sign Magn.	Ones comp.	Twos comp.
0	0	0	0	0	0
0	0	1	$+1$	$+1$	$+1$
0	1	0	$+2$	$+2$	$+2$
0	1	1	$+3$	$+3$	$+3$
1	0	0	0	-3	-4
1	0	1	-1	-2	-3
1	1	0	-2	-1	-2
1	1	1	-3	0	-1

Representing negative numbers - A neat trick

Twos complement notation

- for a positive number n, represent the number by its binary rep. using k bits.
- for a negative number $-n$, represent the number as $2^k - n$.

			Sign Magn.	Ones comp.	Twos comp.
0	0	0	0	0	0
0	0	1	+1	+1	+1
0	1	0	+2	+2	+2
0	1	1	+3	+3	+3
1	0	0	0	-3	-4
1	0	1	-1	-2	-3
1	1	0	-2	-1	-2
1	1	1	-3	0	-1

- using n bits: $-(2^{n-1}) \ldots (2^{n-1} - 1)$.
- widely used representation.

Representing negative numbers

Arithmetic with these representations

			Sign Magn.	Ones comp.	Twos comp.
0	0	0	0	0	0
0	0	1	+1	+1	+1
0	1	0	+2	+2	+2
0	1	1	+3	+3	+3
1	0	0	0	-3	-4
1	0	1	-1	-2	-3
1	1	0	-2	-1	-2
1	1	1	-3	0	-1

Representing negative numbers

Arithmetic with these representations

			Sign Magn.	Ones comp.	Twos comp.
0	0	0	0	0	0
0	0	1	+1	+1	+1
0	1	0	+2	+2	+2
0	1	1	+3	+3	+3
1	0	0	0	-3	-4
1	0	1	-1	-2	-3
1	1	0	-2	-1	-2
1	1	1	-3	0	-1

- $2 + (-3)$

Representing negative numbers

Arithmetic with these representations

			Sign Magn.	Ones comp.	Twos comp.
0	0	0	0	0	0
0	0	1	+1	+1	+1
0	1	0	+2	+2	+2
0	1	1	+3	+3	+3
1	0	0	0	-3	-4
1	0	1	-1	-2	-3
1	1	0	-2	-1	-2
1	1	1	-3	0	-1

- $2 + (-3)$
- $3 + (-2)$

More examples : The case of 4 bits

$$
\begin{array}{lr}
\begin{array}{r}
1\;\;1\;\;\;\;\;\; \\
0110 \\
+1101 \\ \hline
\underbrace{1\,0011}_{} = +3
\end{array}
&
\begin{array}{r}
\text{corresp.} \\
\text{dec. oper.} \\
+6 \\
+\quad -3 \\ \hline
+3
\end{array}
\end{array}
$$

correct result

Example (a)

$$
\begin{array}{lr}
\begin{array}{r}
0100 \\
+1001 \\ \hline
\underbrace{1101}_{} = -3
\end{array}
&
\begin{array}{r}
\text{corresp.} \\
\text{dec. oper.} \\
+4 \\
+\quad -7 \\ \hline
-3
\end{array}
\end{array}
$$

correct result

Example (b)

More examples : The case of 4 bits

$$
\begin{array}{rcr}
 & & \text{corresp.} \\
 & & \text{dec. oper.} \\
0011 & & +3 \\
+0100 & & + \quad +4 \\
\hline
\underbrace{0111}_{\text{correct result}} = \; +7 & & +7
\end{array}
$$

Example (c)

$$
\begin{array}{rcr}
1\;1\;1 & & \text{corresp.} \\
 & & \text{dec. oper.} \\
1110 & & -2 \\
+1010 & & + \quad -6 \\
\hline
1\underbrace{1000}_{\text{correct result}} = \; -8 & & -8
\end{array}
$$

Example (d)

More examples : The case of 4 bits

$$\begin{array}{rl} 1 & \\ 1101 & \\ +1010 & \\ \hline \underbrace{10111}_{\text{incorrect result}} & = +7 \end{array} \qquad \begin{array}{r} \text{corresp.} \\ \text{dec. oper.} \\ -3 \\ + \quad -6 \\ \hline -9 \end{array}$$

Example (e)

$$\begin{array}{rl} 1 & \\ 0101 & \\ +0110 & \\ \hline \underbrace{1011}_{\text{incorrect result}} & = -5 \end{array} \qquad \begin{array}{r} \text{corresp.} \\ \text{dec. oper.} \\ +5 \\ + \quad +6 \\ \hline +11 \end{array}$$

Example (f)

Overflow Detection Rule : If two numbers with the same sign (both positive or both negative) are added, then overflow occurs if and only if the binary representation of the result has the opposite sign.

What to do?

How to Detect it? : The technique of overflow detection is easily implemented in electronic circuitry, and it is a standard feature in digital adder circuits.
How to Prevent it?: Use more bits!