Computer Arithmetic (IV)

- Non-Restoring Division
- Floating Point Arithmetic
- Codes

Restoring division: when we subtract the shifted divisor from the dividend, if the result is negative we restore the dividend to its previous (positive) value.

Non–restoring division: we do not restore the dividend if it becomes negative.

If the result of subtracting the shifted divisor from the dividend is negative: the corresponding bit in the quotient is then 0;
If the result of the subtraction is positive the relevant quotient bit is set to 1.

The difference with non-restoring division is that if a subtraction step produces a negative dividend, the next time, we add the shifted divisor. If the result of the addition is positive the relevant quotient bit is set to 1.

Divide 14 = 1110 by 3 = 11 using non-restoring division

n = 3

n	Dividend	Shifted divisor	Operation	Result	Quotient bit
3	14	$2^3 \times 3$	-	-10	0
2	-10	$2^2 \times 3$	+	+2	1
1	2	$2^1 \times 3$	-	- 4	0
0	-4	$2^0 \times 3$	+	-1	0
				2	

If final result is negative, one final addition is necessary to restore the remainder. (+3 in this case)

Non-restoring Division

$456 \div 23$ by non-restoring division

Dividend	Shifted divisor	Op	Result	Quotient digit
456	$2^8 \times 23$ 5888	-	-5432	0
-5432	$2^7 \times 23$ 2944	+	-2488	0
-2488	$2^6 \times 23$ 1472	+	-1016	0
-1016	$2^5 \times 23$ 736	+	-280	0
-280	$2^4 \times 23$ 368	+	88	1
88	$2^3 \times 23$ 184	-	-96	0
-96	$2^2 \times 23$ 92	+	-4	0
-4	$2^1 \times 23$ 46	+	42	1
42	$2^0 \times 23$ 23	-	19	1

Number Representation

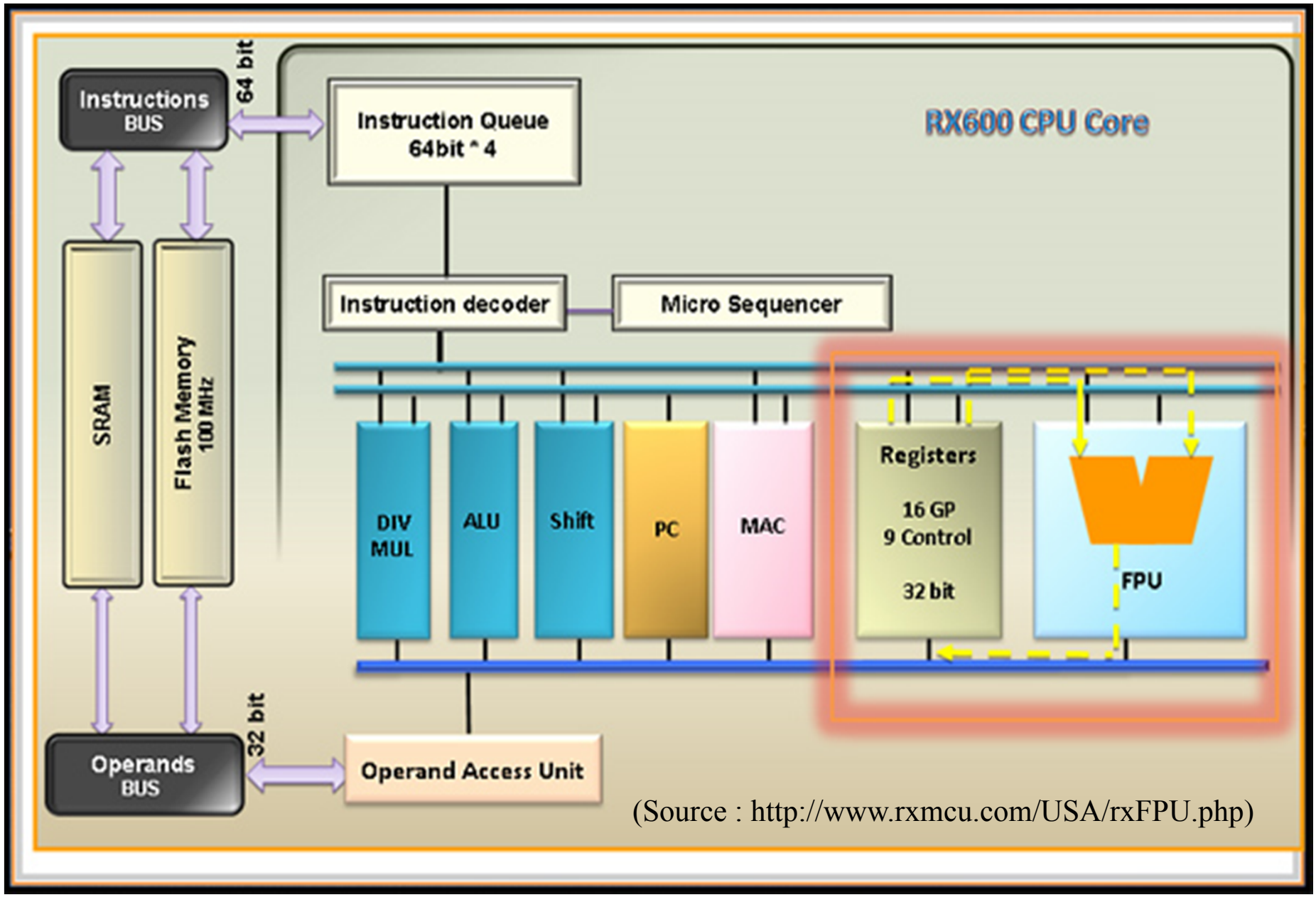
We have seen how to represent positive integers in binary, decimal, hexadecimal and negative integers using the 2s complement form.

We have performed integer binary addition by adding individual bits and dealing appropriately with the carries produced.

We have performed integer multiplication and division based on the long-hand techniques used in decimal.

Integer values are generally machine dependent (64 bits). What happens when a much larger dynamic range is required ?

e.g. mass of an electron $9.10938291 \times 10^{-31}$ kilograms



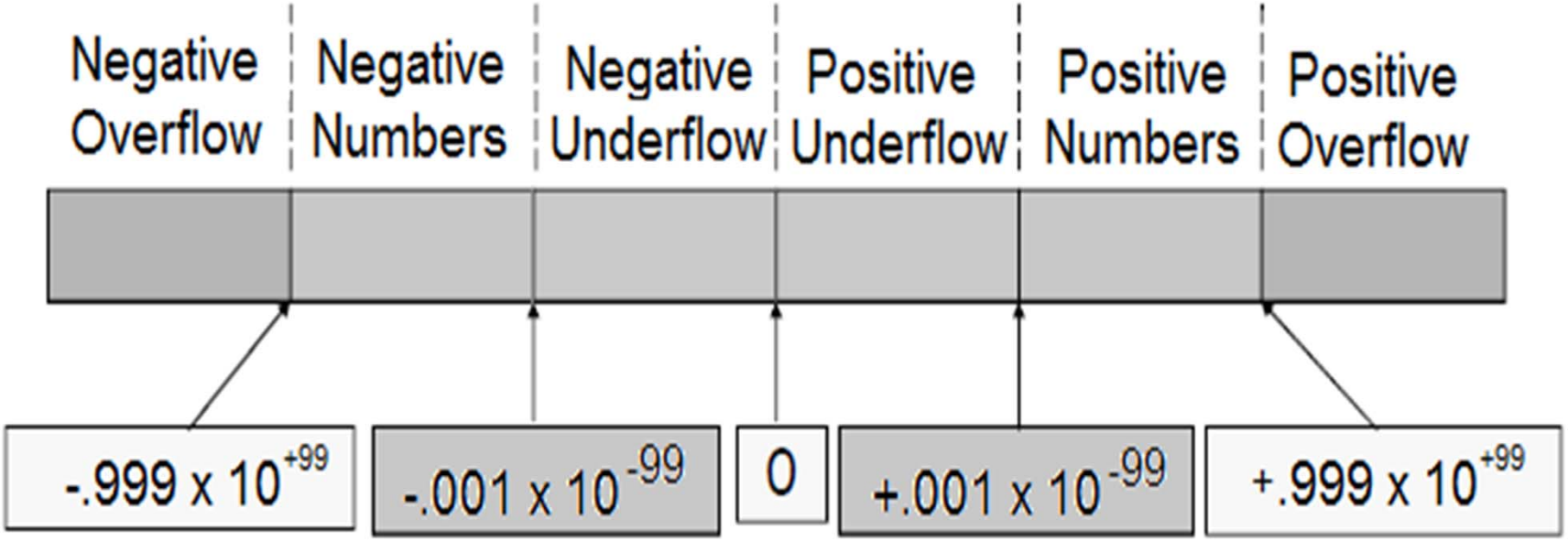
(Source : http://www.rxmcu.com/USA/rxFPU.php)

Floating Point Numbers

- numbers in which dynamic range is controlled separately to precision.
- In an integer dynamic range and precision are controlled together. A 32 bit integer has a dynamic range of 2^{32} and a precision of 1 part in 2^{32}.
- A floating point number is made from 2 parts:
  - The mantissa, ***m*** - sets the precision (how exact the number is)
  - The exponent, ***e*** - sets the dynamic range (how big or small the number is)
- The value of a floating point number is $\boldsymbol{m \times 2^e}$

Dynamic Range

- Consider a decimal number formed with a signed 3-digit mantissa and a signed 2-digit exponent
- Dynamic Range is $\pm .001 \times 10^{-99}$ to $\pm .999 \times 10^{+99}$



If there are ***m*** bits in the mantissa then the precision should be **1** part in $\mathbf{2^{m-1}}$.

If there are ***e*** bits in the exponent then the dynamic range is approximately $\mathbf{2^{-(2^{e-1})}}$ **to** $\mathbf{2^{2^{e-1}-1}}$

assuming that m and e are signed

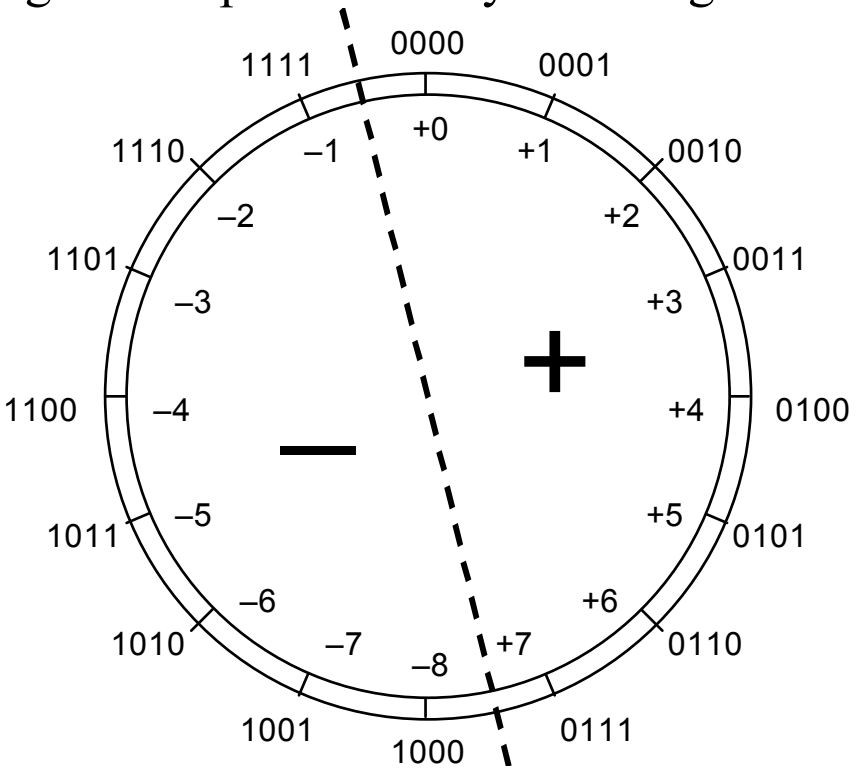
Typical values of *m* and *e* are:

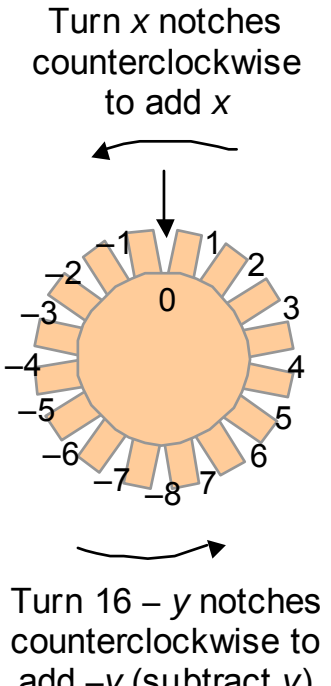
Precision	***m***	***e***	**resolution**
single	24	8	1 in 2^{24}
double	48	16	1 in 2^{48}
extended	64	16	1 in 2^{64}

Two's-Complement Representation

With k bits, numbers in the range $[-2^{k-1}, 2^{k-1} - 1]$ represented.

Negation is performed by inverting all bits and adding 1.



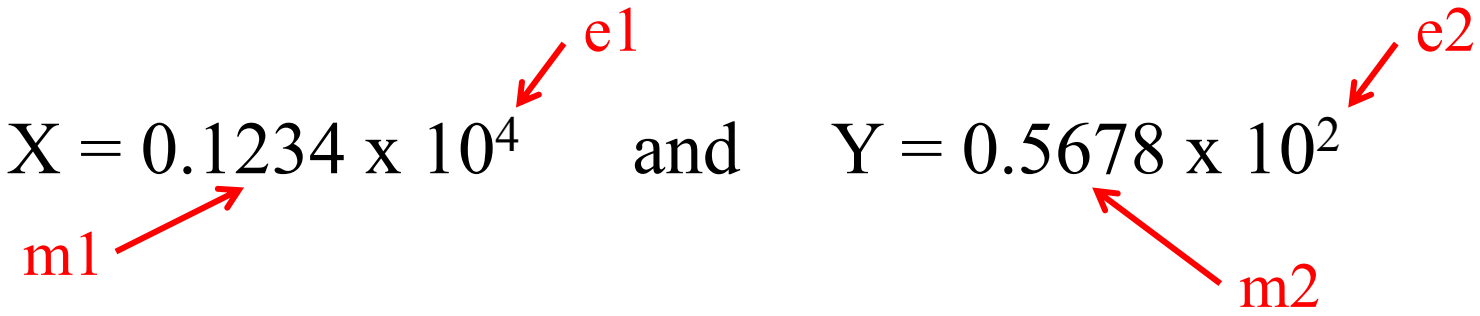


Schematic representation of 4-bit 2's-complement code for integers in [–8, +7].

Normalisation

- It is mandatory that numbers fit into an integer number of bytes, e.g. 4
- for single precision, $m = 24$ and $e = 8$. This would only give a resolution of 1 in 2^{23} when the sign bit is taken into account (floating point numbers use sign-magnitude format)
- You can normalise a floating point number e.g.
- $0.00110 \times 2^{n} = 0.01100 \times 2^{n-1} = 0.11000 \times 2^{n-2} = 1.10000 \times 2^{n-3}$
- Before storage, floating point numbers are normalised so that the mantissa is between 0.5 and 1 e.g. $m = 0.1abcdef...$
- This being the case, the 0.1 is not stored, only *abcdef*...
- Therefore, $m = 24$ gives 1 part in 2^{24}

Consider adding the following decimal numbers:



We are actually trying to add X = 1234 and Y = 56.78

$$X + Y = 1290.78 = 0.129078 \times 10^4$$

Align to the number with the largest exponent,
in this case X as e1 > e2

Increasing Y's exponent from 10^2 to 10^4 increases its value by 100 times, so its mantissa must be decreased by 100 times.

$$Y = 0.5678 \times 10^2 = 0.005678 \times 10^4$$

This is equivalent to shifting the Y mantissa right by 2 places.

More generally, shifting the mantissa right (e1 – e2) places.

Thus,

$$m_1 \times 2^{e1} + m_2 \times 2^{e2} = (m_1 + (m_2 \; shr \; e_1 - e_2)) \times 2^{e1} \text{ if } e_1 > e_2$$

Operations

Addition

$m_1 x\ 2^{e1} + m_2 x\ 2^{e2} = (m_1 + (m_2\ shr\ e_1 - e_2)) x\ 2^{e1}$ if $e_1 > e_2$

$m_1 x\ 2^{e1} + m_2 x\ 2^{e2} = (m_2 + (m_1\ shr\ e_2 - e_1)) x\ 2^{e2}$ if $e_2 > e_1$

$m_1 x\ 2^{e1} + m_2 x\ 2^{e2} = (m_1 + m_2) x\ 2^{e1}$ if $e_1 = e_2$

Multiplication

$m_1 x\ 2^{e1}$ x $m_2 x\ 2^{e2} = m_1$ x $m_2 x\ 2^{e1+e2}$

Size of result

- for addition, the mantissa can lie between 0.5 and 2
- for multiplication the mantissa can lie between 0.25 and 1
- Hence, results will need to be post-normalised to ensure that the mantissa lies between 0.5 and 1

Precision

- When an operation is performed, the mantissa must be truncated back to the defined size. Therefore it must be rounded.
- Usually, floating point operations are performed ‘internally’ to a greater precision (3 bits extra, for example) and these ‘guard bits’ are used to decide whether to round the least significant bit of the mantissa to 1 or 0.

Serious consequences can result from problems with number precision and conversion.

Research the following:

- Pentium Bug
- Patriot missile software problem
- Ariane 5 rocket, Flight 501

ASCII Character Code

- ASCII is a 7-bit code, commonly stored in 8-bit bytes.
- "A" is at 41_H. To convert upper case letters to lower case letters, add 20_H. Thus "a" is at $41_H + 20_H = 61_H$.
- The character "5" at position 35_H is different than the number 5. To convert character-numbers into number-numbers, subtract 30_H:

  $35_H - 30_H = 5$

- Extended ASCII uses 8 bits to give a few more characters

00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	´	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

NUL	Null	FF	Form feed	CAN	Cancel
SOH	Start of heading	CR	Carriage return	EM	End of medium
STX	Start of text	SO	Shift out	SUB	Substitute
ETX	End of text	SI	Shift in	ESC	Escape
EOT	End of transmission	DLE	Data link escape	FS	File separator
ENQ	Enquiry	DC1	Device control 1	GS	Group separator
ACK	Acknowledge	DC2	Device control 2	RS	Record separator
BEL	Bell	DC3	Device control 3	US	Unit separator
BS	Backspace	DC4	Device control 4	SP	Space
HT	Horizontal tab	NAK	Negative acknowledge	DEL	Delete
LF	Line feed	SYN	Synchronous idle		
VT	Vertical tab	ETB	End of transmission block		

Unicode

- Characters available for most of the world's writing systems. Each character has a 32 bit code point.
- Backward compatible with ASCII.
- Implemented using character encodings such as UTF-32, UTF-16, UTF-8.
- UTF-8 keeps the ASCII subset as 8 bits and uses 16-32 bits for other characters.

0000	NUL	0020	SP	0040	@	0060	`	0080	Ctrl	00A0	NBS	00C0	À	00E0	à
0001	SOH	0021	!	0041	A	0061	a	0081	Ctrl	00A1	¡	00C1	Á	00E1	á
0002	STX	0022	"	0042	B	0062	b	0082	Ctrl	00A2	¢	00C2	Â	00E2	â
0003	ETX	0023	#	0043	C	0063	c	0083	Ctrl	00A3	£	00C3	Ã	00E3	ã
0004	EOT	0024	$	0044	D	0064	d	0084	Ctrl	00A4	¤	00C4	Ä	00E4	ä
0005	ENQ	0025	%	0045	E	0065	e	0085	Ctrl	00A5	¥	00C5	Å	00E5	å
0006	ACK	0026	&	0046	F	0066	f	0086	Ctrl	00A6	¦	00C6	Æ	00E6	æ
0007	BEL	0027	'	0047	G	0067	g	0087	Ctrl	00A7	§	00C7	Ç	00E7	ç
0008	BS	0028	(	0048	H	0068	h	0088	Ctrl	00A8	¨	00C8	È	00E8	è
0009	HT	0029	)	0049	I	0069	i	0089	Ctrl	00A9	©	00C9	É	00E9	é
000A	LF	002A	*	004A	J	006A	j	008A	Ctrl	00AA	ª	00CA	Ê	00EA	ê
000B	VT	002B	+	004B	K	006B	k	008B	Ctrl	00AB	«	00CB	Ë	00EB	ë
000C	FF	002C	´	004C	L	006C	l	008C	Ctrl	00AC	¬	00CC	Ì	00EC	ì
000D	CR	002D	-	004D	M	006D	m	008D	Ctrl	00AD	–	00CD	Í	00ED	í
000E	SO	002E	.	004E	N	006E	n	008E	Ctrl	00AE	®	00CE	Î	00EE	î
000F	SI	002F	/	004F	O	006F	o	008F	Ctrl	00AF	¯	00CF	Ï	00EF	ï
0010	DLE	0030	0	0050	P	0070	p	0090	Ctrl	00B0	°	00D0	Ð	00F0	¶
0011	DC1	0031	1	0051	Q	0071	q	0091	Ctrl	00B1	±	00D1	Ñ	00F1	ñ
0012	DC2	0032	2	0052	R	0072	r	0092	Ctrl	00B2	²	00D2	Ò	00F2	ò
0013	DC3	0033	3	0053	S	0073	s	0093	Ctrl	00B3	³	00D3	Ó	00F3	ó
0014	DC4	0034	4	0054	T	0074	t	0094	Ctrl	00B4	´	00D4	Ô	00F4	ô
0015	NAK	0035	5	0055	U	0075	u	0095	Ctrl	00B5	µ	00D5	Õ	00F5	õ
0016	SYN	0036	6	0056	V	0076	v	0096	Ctrl	00B6	¶	00D6	Ö	00F6	ö
0017	ETB	0037	7	0057	W	0077	w	0097	Ctrl	00B7	·	00D7	×	00F7	÷
0018	CAN	0038	8	0058	X	0078	x	0098	Ctrl	00B8	¸	00D8	Ø	00F8	ø
0019	EM	0039	9	0059	Y	0079	y	0099	Ctrl	00B9	¹	00D9	Ù	00F9	ù
001A	SUB	003A	:	005A	Z	007A	z	009A	Ctrl	00BA	º	00DA	Ú	00FA	ú
001B	ESC	003B	;	005B	[	007B	{	009B	Ctrl	00BB	»	00DB	Û	00FB	û
001C	FS	003C	<	005C	\	007C	|	009C	Ctrl	00BC	1/4	00DC	Ü	00FC	ü
001D	GS	003D	=	005D	]	007D	}	009D	Ctrl	00BD	1/2	00DD	Ý	00FD	Þ
001E	RS	003E	>	005E	^	007E	~	009E	Ctrl	00BE	3/4	00DE	ý	00FE	þ
001F	US	003F	?	005F	_	007F	DEL	009F	Ctrl	00BF	¿	00DF	ß	00FF	ÿ

NUL	Null	SOH	Start of heading	CAN	Cancel	SP	Space
STX	Start of text	EOT	End of transmission	EM	End of medium	DEL	Delete
ETX	End of text	DC1	Device control 1	SUB	Substitute	Ctrl	Control
ENQ	Enquiry	DC2	Device control 2	ESC	Escape	FF	Form feed
ACK	Acknowledge	DC3	Device control 3	FS	File separator	CR	Carriage return
BEL	Bell	DC4	Device control 4	GS	Group separator	SO	Shift out
BS	Backspace	NAK	Negative acknowledge	RS	Record separator	SI	Shift in
HT	Horizontal tab	NBS	Non-breaking space	US	Unit separator	DLE	Data link escape
LF	Line feed	ETB	End of transmission block	SYN	Synchronous idle	VT	Vertical tab

Computer Architectures
Introduction

- Historical Introduction
- Moore's Law
- General Computer Classifications

A Historical Introduction …..

This wolf radius bone from around 25,000 B.C. shows 55 cuts in groups of five.

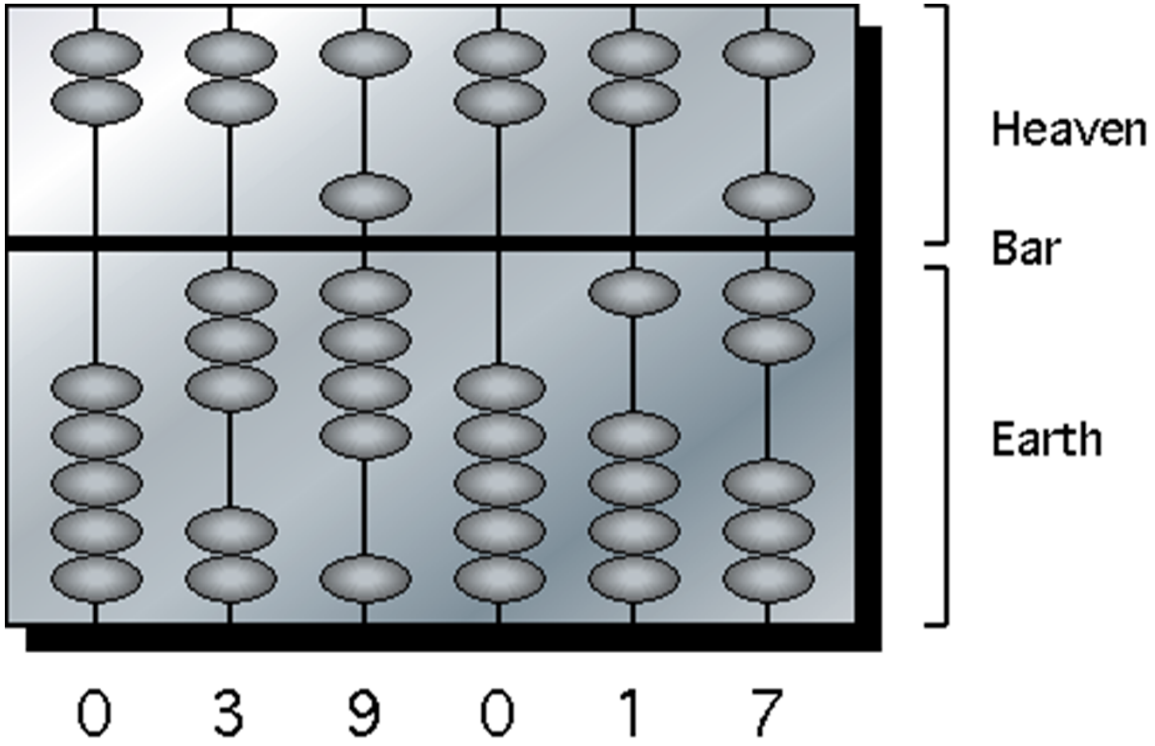
(Source: Illustrated London News, October 2, 1937.)

Chinese Abacus

500 B.C. – count in decimal or hex !

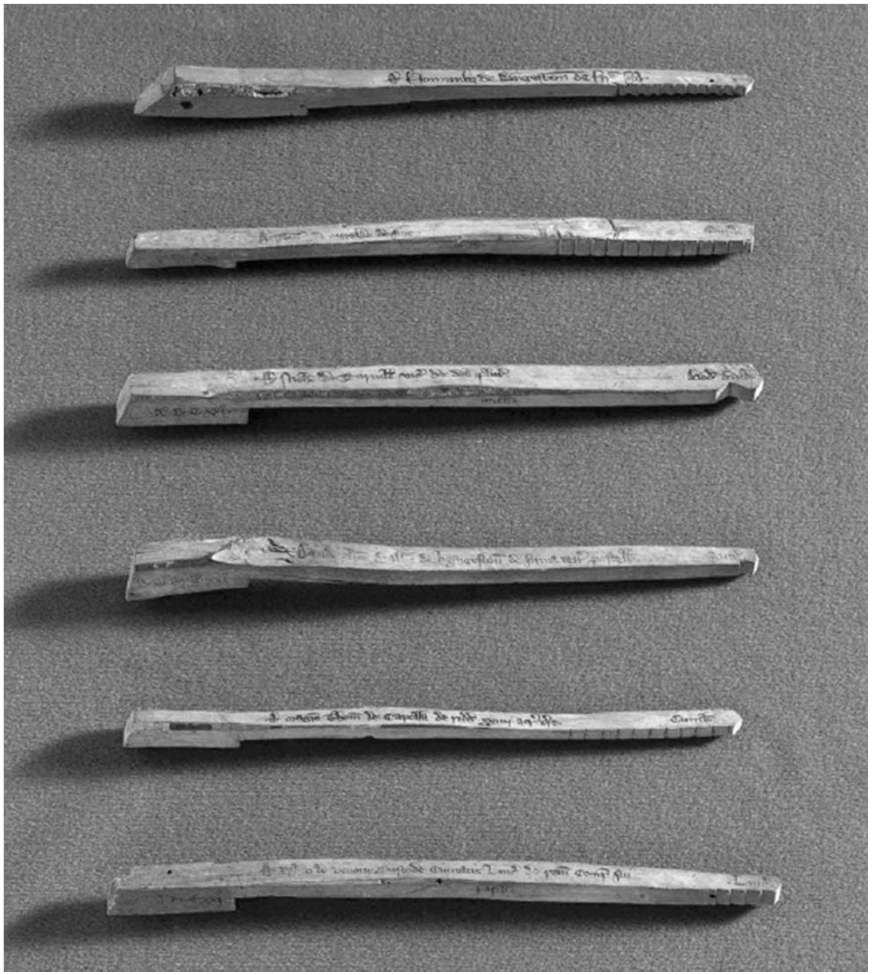
heaven bead is worth 5

earth bead is worth 1

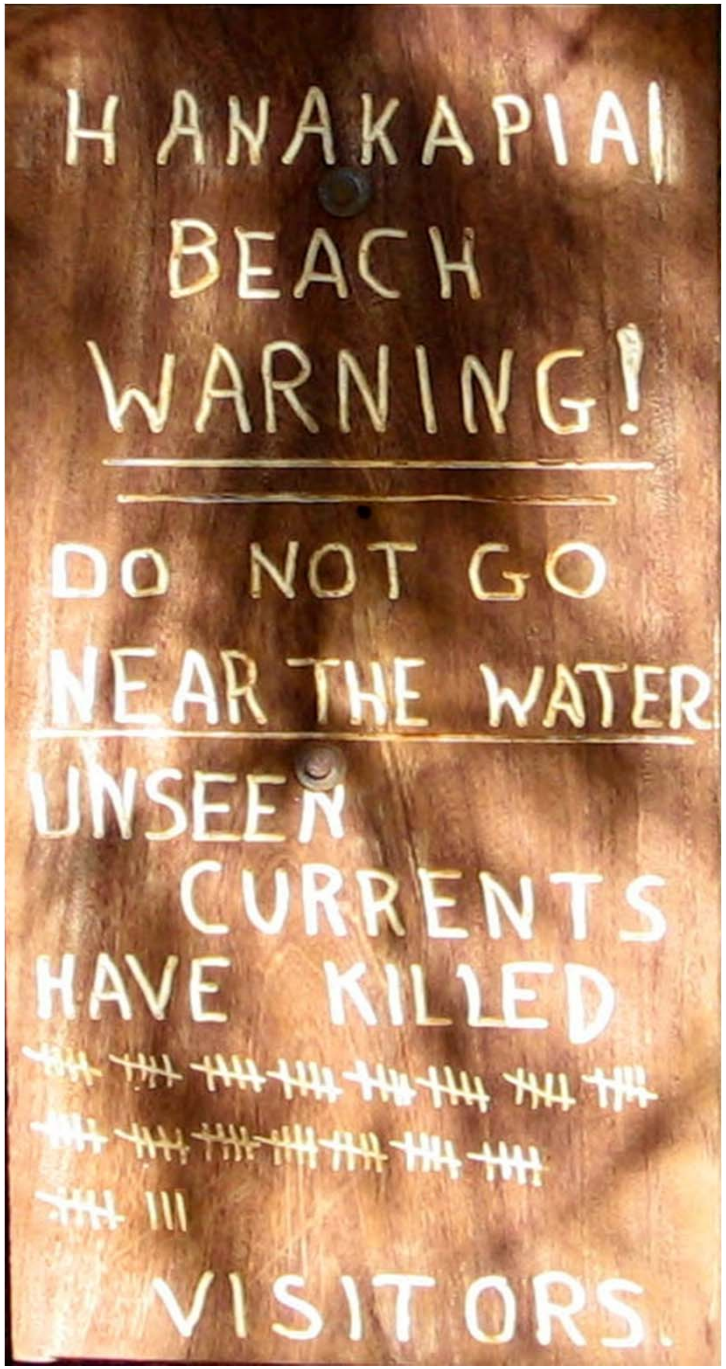


Tally Sticks

Wooden tally sticks from Westminster, England, 1250–1275 A.D.



(© SSPL/The ImageWorks.)



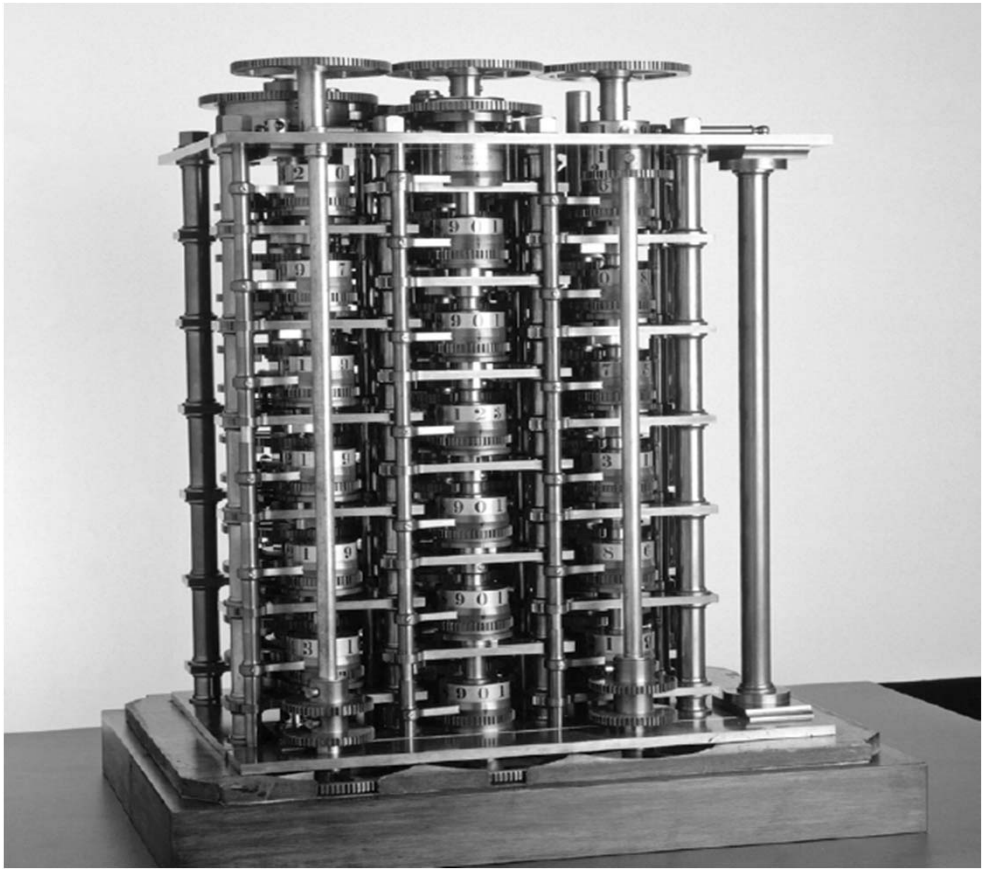
Music Box – 1500s

A mechanical device.

GNU Free Documentation License.

Babbage's Difference Engine

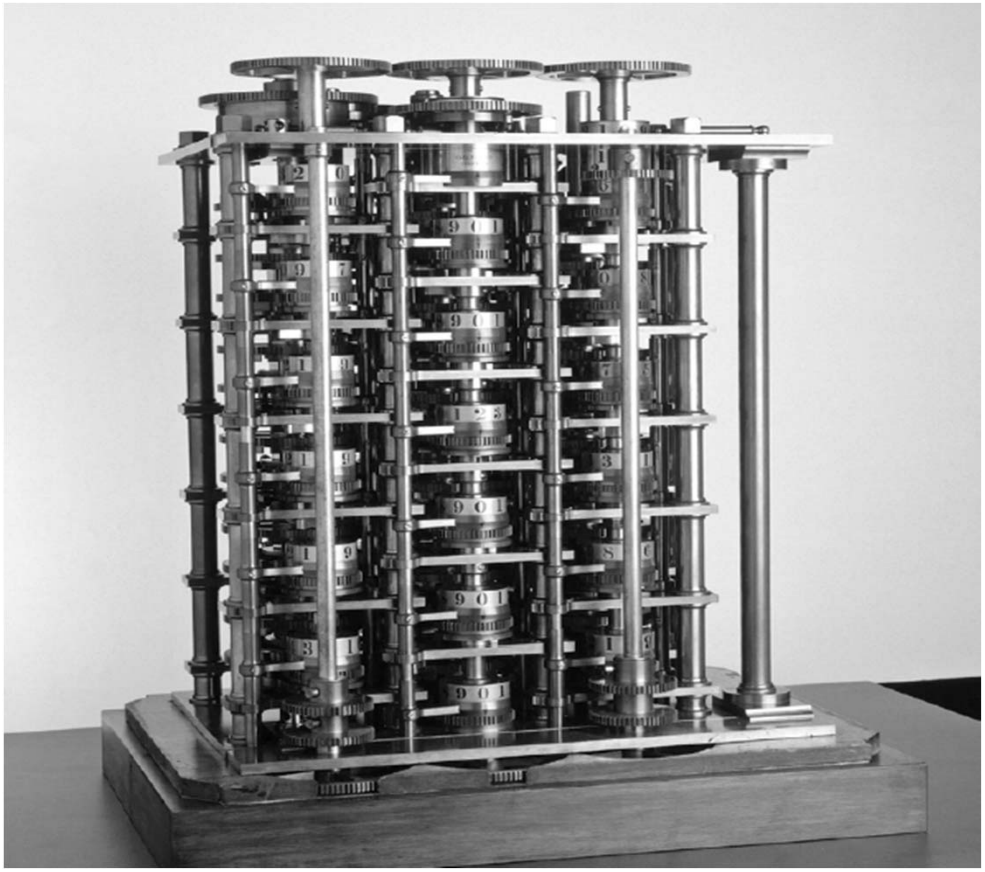
a mechanical calculator



(© SSPL/The ImageWorks.)

Babbage's Difference Engine

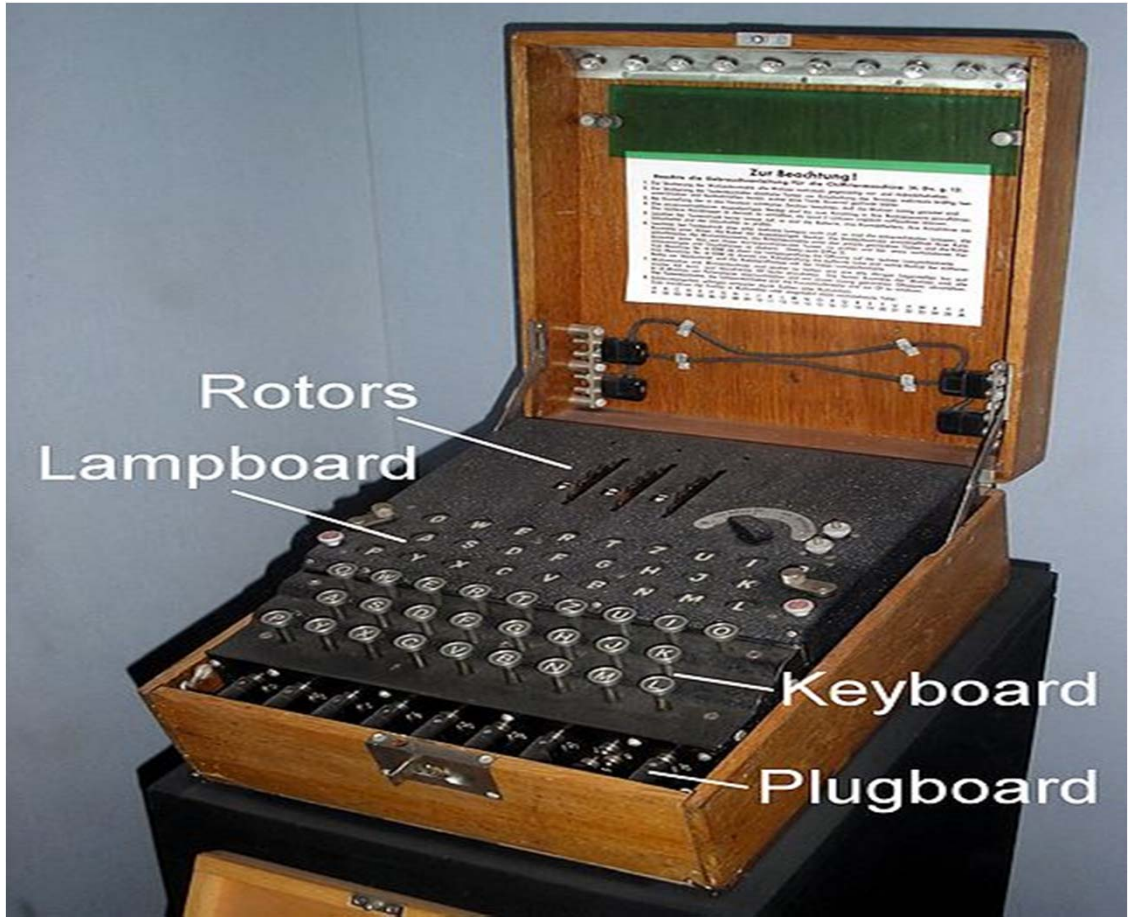
a mechanical calculator



(© SSPL/The ImageWorks.)

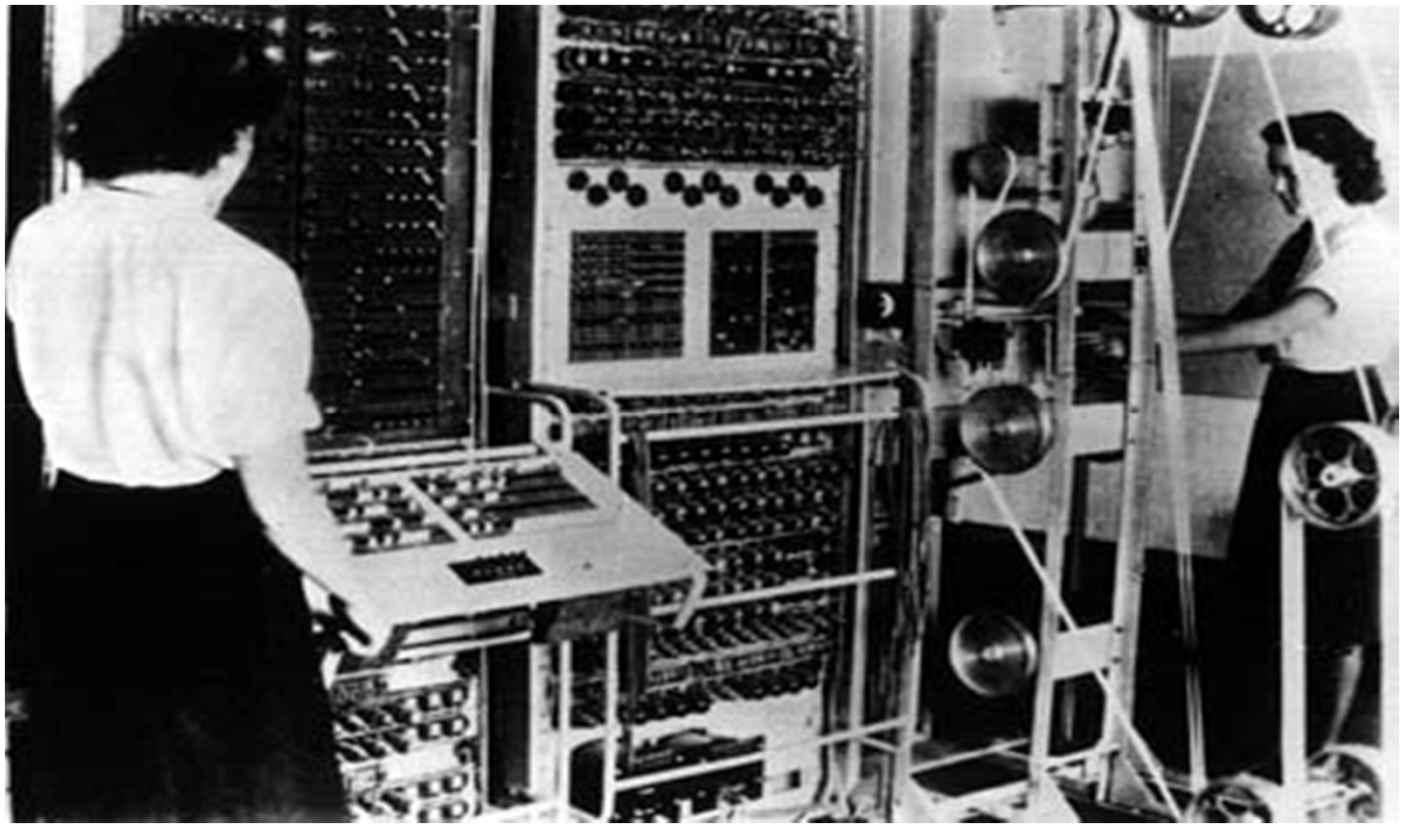
Enigma machine

electro-mechanical rotor cipher machine



Colossus

electronic computing device (valves)



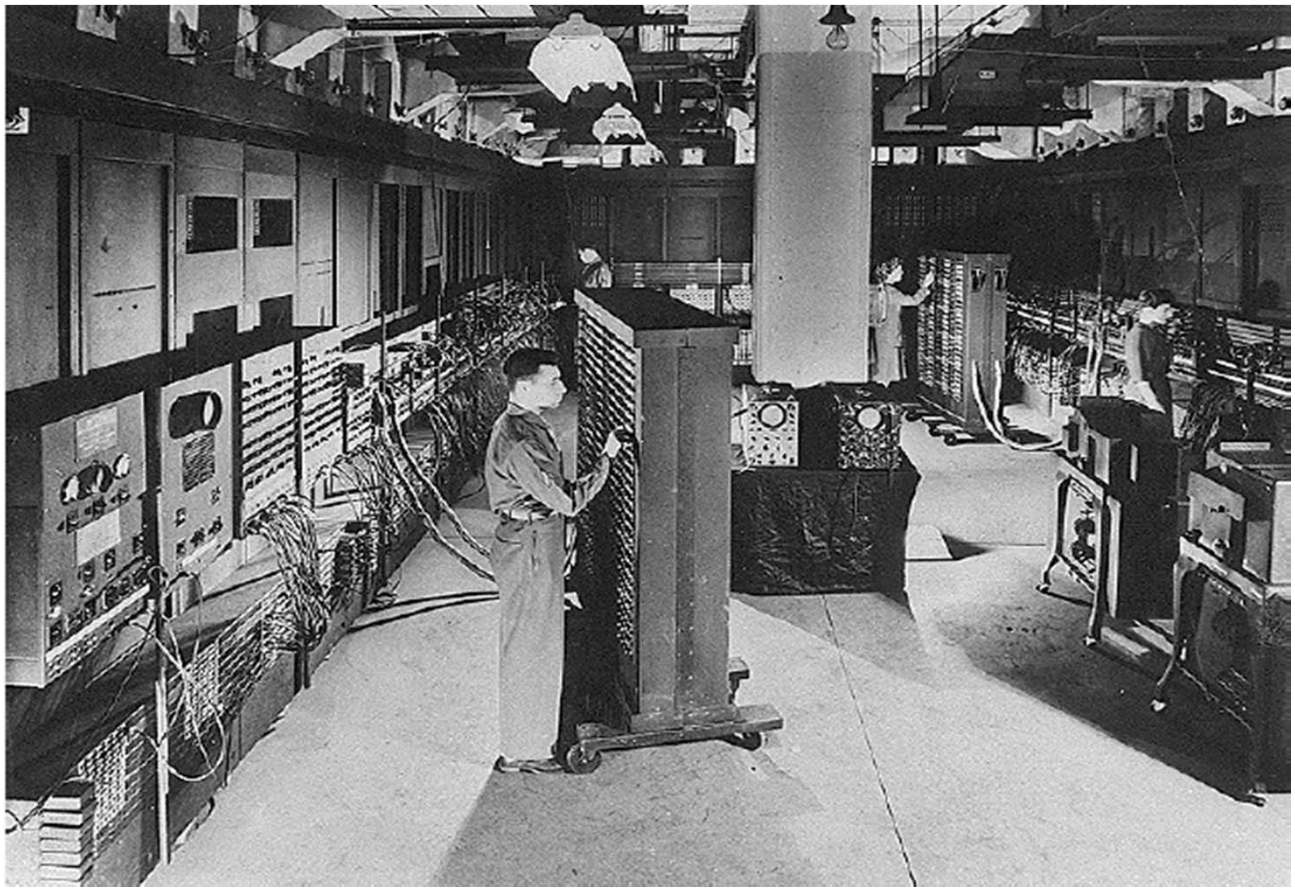
(Source: http://www.turing.org.uk/turing/scrapbook/electronic.html.)

Thermionic Valve used as a switch in computing.

GNU Free Documentation License.

ENIAC – Electronic Numerical Integrator and Calculator

general purpose electronic computer

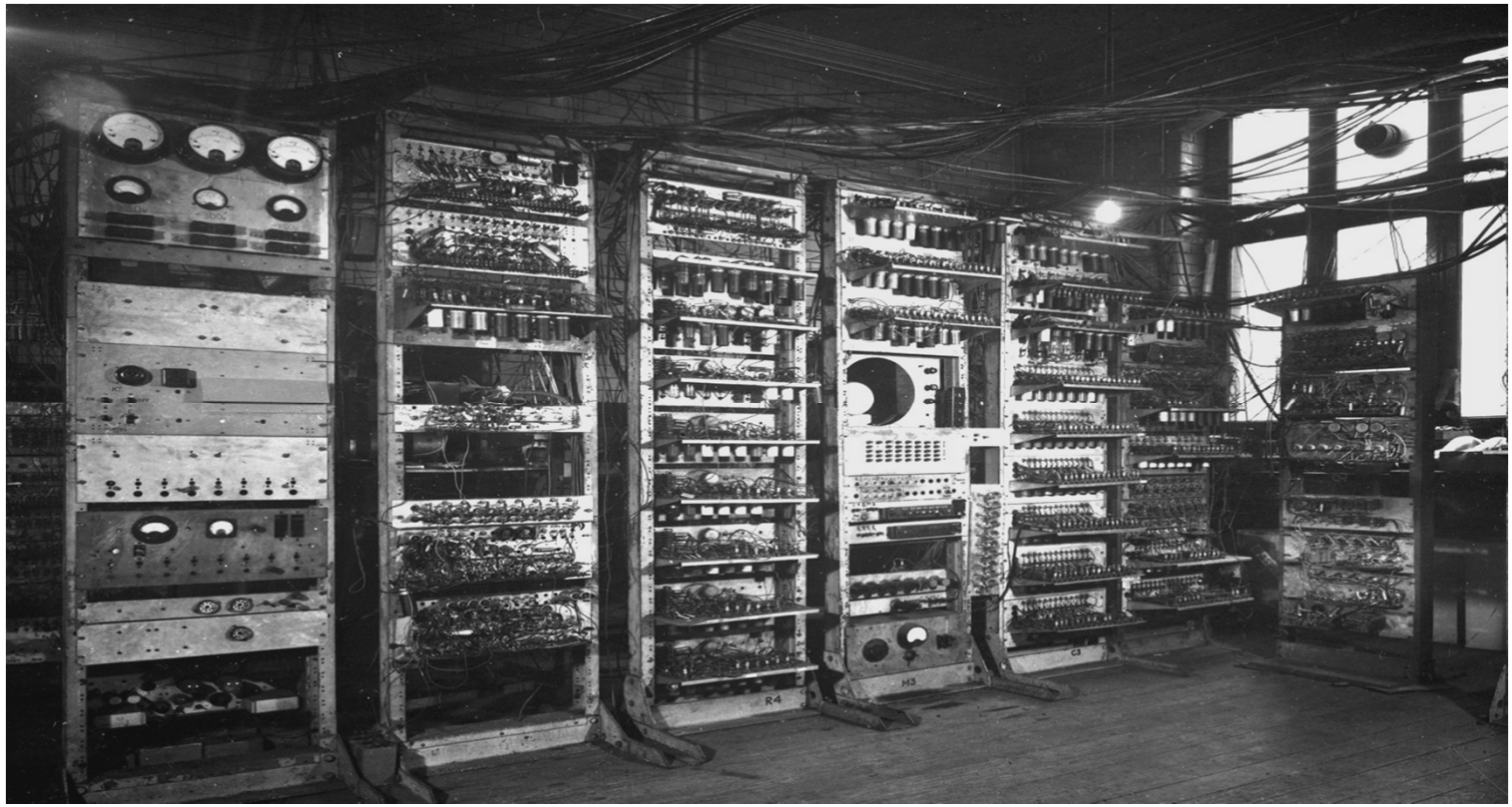


(Time & Life Pictures/Getty Images.)

Manchester University Mark 1 / Baby

First stored program computer.

Prototype for Ferranti Mark 1, worlds first commercially available general purpose computer.

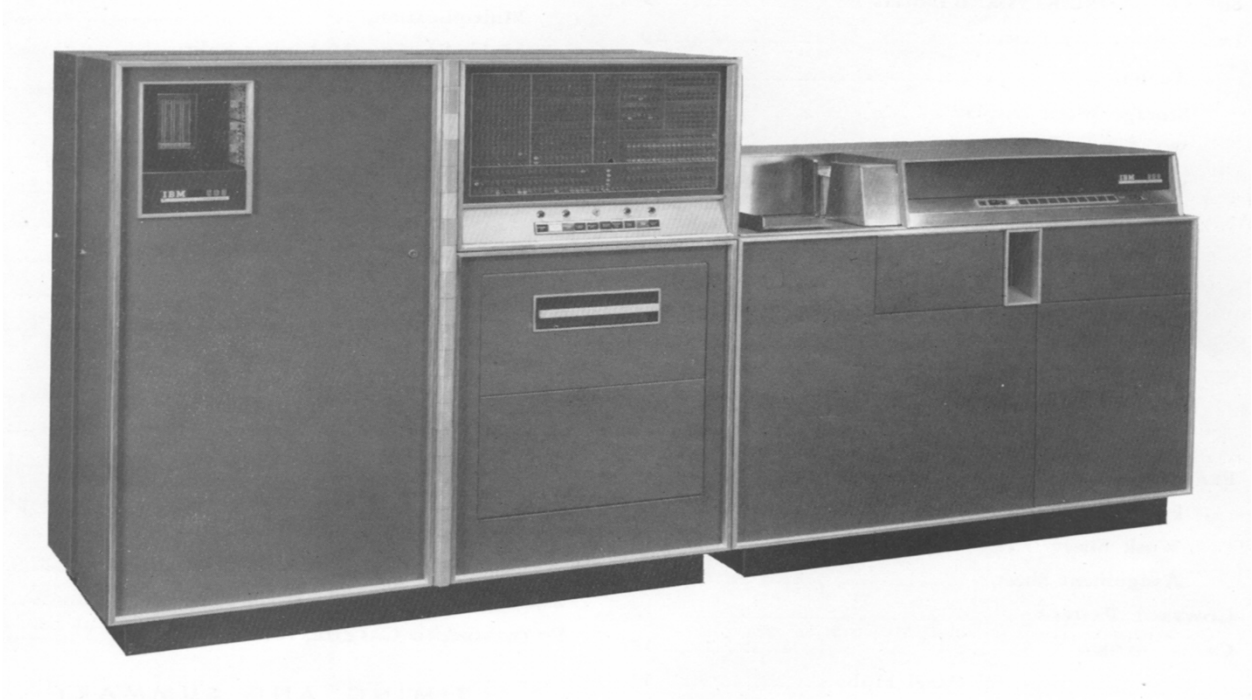


(Source: The University of Manchester, www.computer50.org/mark1/ip-mm1.mark1.html)

IBM 608

first commercially available transistor based calculator, 1955

3000 individual transistors, each no bigger than a paper-clip



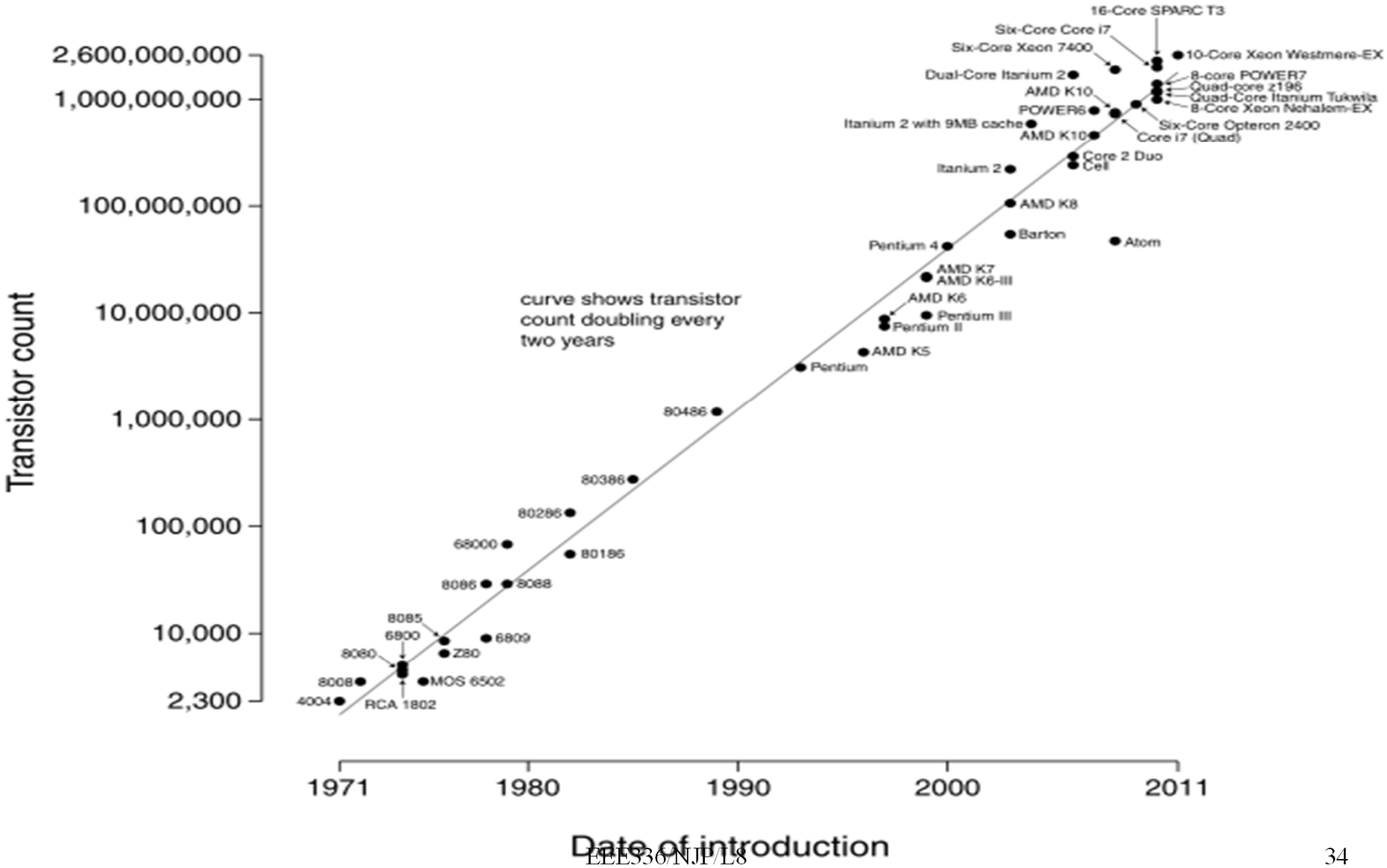
(Source: IBM training manual)

Today : millions of transistors integrated onto a single silicon chip.

Intel Xeon, 2.6 billion transistors.

Moore's Law : The number of transistors on a chip roughly doubles every two years.

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Computer Applications

Desktop Computer:	Used by an individual, low cost, runs third-party software.
Server:	Large workloads, multiple users, accessed via a network, greater emphasis on reliability
Super Computer:	High end scientific calculations, multiple processors e.g. weather forecasting
Embedded Computer:	Computer embedded in another device, generally run one predetermined application e.g. cars, phones, set-top boxes, washing machine