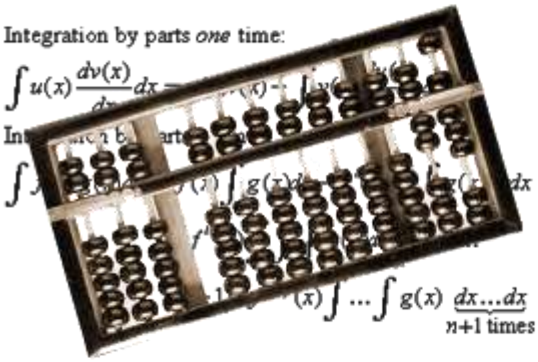
0.1-History of Computers

What do *calculate*, *calcium* and *abacuses* all have in common? The verb *calculate* comes from the Latin *calculāt-* participial stem of *calculāre* to count, reckon, and from *calculus* a stone. *Calculus*, comes from the Latin diminutive *calx* stone, pebble



In the Oxford English Dictionary,1933
"**Calculator** – One who calculates; a reckoner".
"**Computer** – One who computes; a calculator, reckoner; specifically a person employed to make calculations in an observatory, in surveying, etc."

0.2- Theoretical mathematical computers (URM, TM)

1. The unlimited register machine (URM)

Our mathematical idealization of a computer is called an unlimited register machine (URM). The URM has a infinite number of registers labeled R_1, R_2 , R_3, … each of which at any moment of time contains a natural number: we denote the number contained R_n by r_n . This can be represented as follows:

R1	R2	R3	R4	R5	
r1	r2	r3	r4	r5	…

Instructions correspond to very simple operations used in performing calculations with numbers. A finite list of instructions constitutes a *program*. The 4 instructions kinds are:

Instruction	Form	Meaning
Zero	Z(n)	$0 \rightarrow R_n$ or $r_n := 0$
Successor	S(n)	$r_n + 1 \rightarrow R_n$ or $r_n := r_n + 1$
Transfer	T(m,n)	$r_m \rightarrow R_n$ or $r_m := r_n$
Jump	J(m,n,q)	If $r_m = r_n$ jump to q^{th} instruction

Example: Consider the following program:

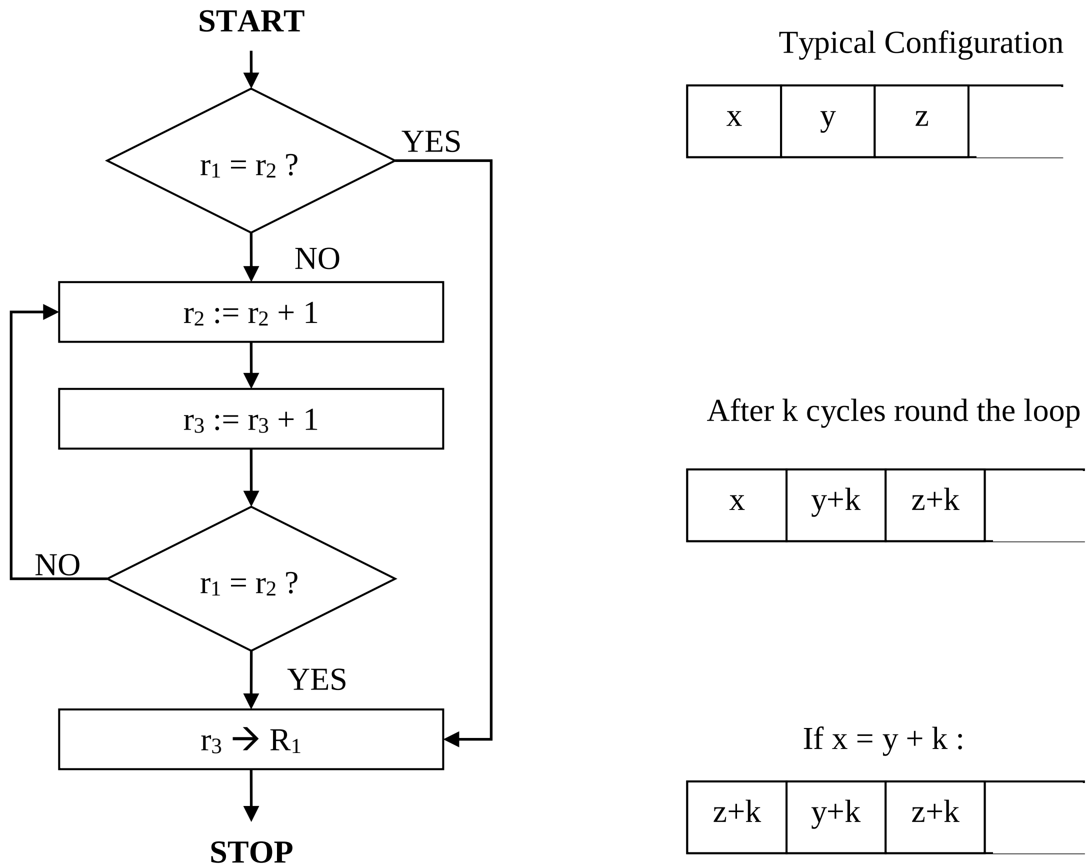
I_1: J(1, 2, 6)
I_2: S(2)
I_3: S(3)
I_4: J(1, 2, 6)
I_5: J(1, 1, 2)
I_6: T(3, 1)

i) Perform computation under this program with the initial configuration 8,4,2,0, …
ii) Analyze the function of the program and Draw the flow diagram for the program

	R1	R2	R3	R4	R5		Next Instruction
Initial configuration	8	4	2	0	0	…	I_1
	8	4	2	0	0	…	I_2 (since $r_1 \neq r_2$)
	8	5	2	0	0	…	I_3
	8	5	3	0	0	…	I_4
	8	5	3	0	0	…	I_5 (since $r_1 \neq r_2$)
	8	5	3	0	0	…	I_2 (since $r_1 = r_1$)
	8	6	3	0	0	…	I_3
	8	6	4	0	0	…	I_4
	8	6	4	0	0	…	I_5 (since $r_1 \neq r_2$)
	8	6	4	0	0	…	I_2 (since $r_1 = r_1$)
	8	7	4	0	0	…	I_3
	8	7	5	0	0	…	I_4
	8	7	5	0	0	…	I_5 (since $r_1 \neq r_2$)
	8	7	5	0	0	…	I_2 (since $r_1 = r_1$)
	8	8	5	0	0	…	I_3
	8	8	6	0	0	…	I_4
	8	8	6	0	0	…	I_6 (since $r_1 = r_2$)
Final configuration	**6**	8	6	0	0	…	I_7 : STOP

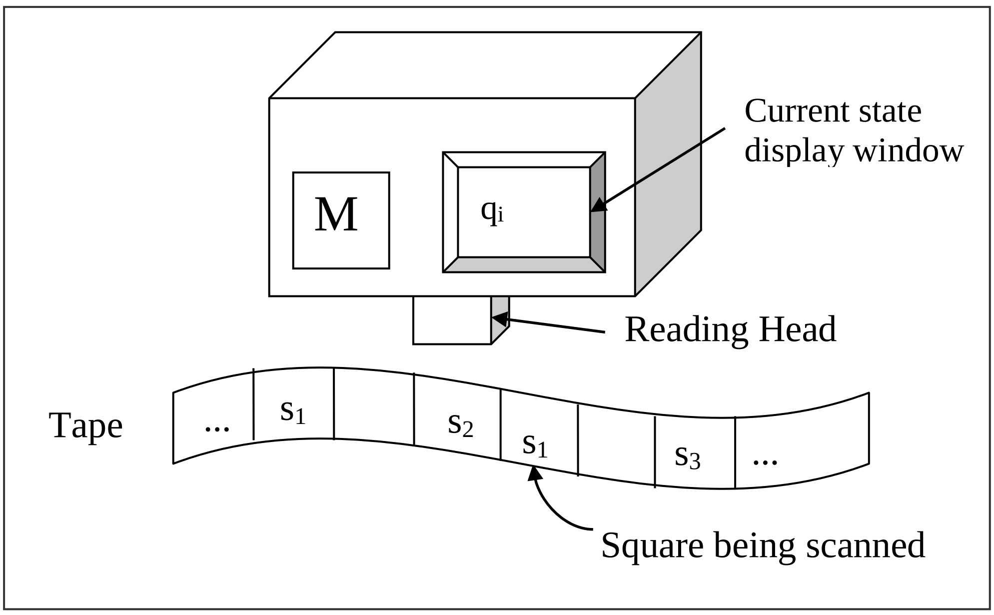
Assume the initial configuration is : x,y,z. After k iterations, the value of first register is z+k.

Output is in first register. If x = y + k. then the output is z + x - y



2. Turing machine

A Turing machine M is a finite device, which performs operations on a paper tape ([Cutland-1980], [Benenti-2004]). This tape is infinite in both directions, and is divided into same-sized squares. At any given time each square of the tape is either blank (B) or contains a single symbol from a fixed finite list of symbols s_1, s_2 …, s_n that form the alphabet of M. M has a reading head which at any given time scans or reads a single square of the tape. M is capable of three kinds of simple operations: Replacing the symbol in the square being scanned ay another symbol from the alphabet of M, Moving the reading head one square to right, or Moving the reading head one square to left.



Turing Machine

The action that M takes at any instant depends on the current state of M and the symbol currently being scanned. This dependence is described in M's specification which consists of a finite set Q of quadruples, each of which takes one of the following forms:

$q_i\ s_j\ s_k\ q_l$

$q_i\ s_j\ R\ q_l$ $\qquad \begin{pmatrix} 1 \le i, l \le m \\ 0 \le j, k \le n \end{pmatrix}$

$q_i\ s_j\ L\ q_l$

Figure 4: A Turing machine specification

A quadruple $q_i\ s_j\ \alpha\ q_l$ in Q specifies the action to be taken when the state is q_i and scanning the symbol s_j , as follows:

1- Operate on the tape thus:

a) if $\alpha = s_k$ erase s_j, and write s_k n the square being scanned;

b) if α = R move the reading head one square to the right:

c) if α = L move the reading dean one square to the left:

2- Change into state q_l.

Example: Performing Addition on a Turing-Machine

The Turing machine given by the following specification Turing-computes the function x + y:

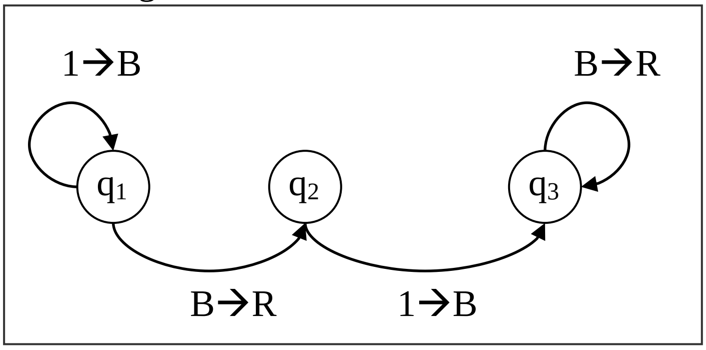
q_1 1 B q_1

q_1 B R q_2

q_2 1 B q_3

q_2 B R q_2

The tape representation of (x, y) contains x+y+2 occurrence of 1 symbol. The machine is designed just to erase two of these occurrences from left. Here is a state machine illustrating addition on a Turing machine.



State machine for addition on a Turing machine

Note that arcs are labelled with a couple of symbols separated by an arrow in the form: $\alpha \rightarrow \beta$ which is read as: If α then β.

References

http://metaphortuitous.blogspot.com.eg/2012/04/calculated-answer-shared-etymology-of.html

Cutland, **Nigel** J (1980), **Computability**: An Introduction to Recursive Function Theory