Machine Architecture
Introduction & LMC model

Dr. Magnus Bordewich

Computer Systems Schedule

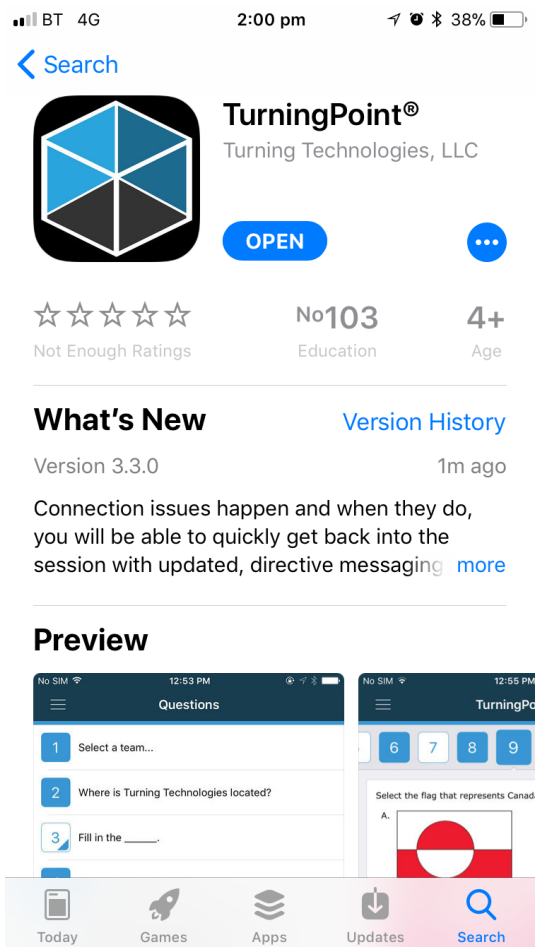
This term:
Digital Electronics & Machine architecture

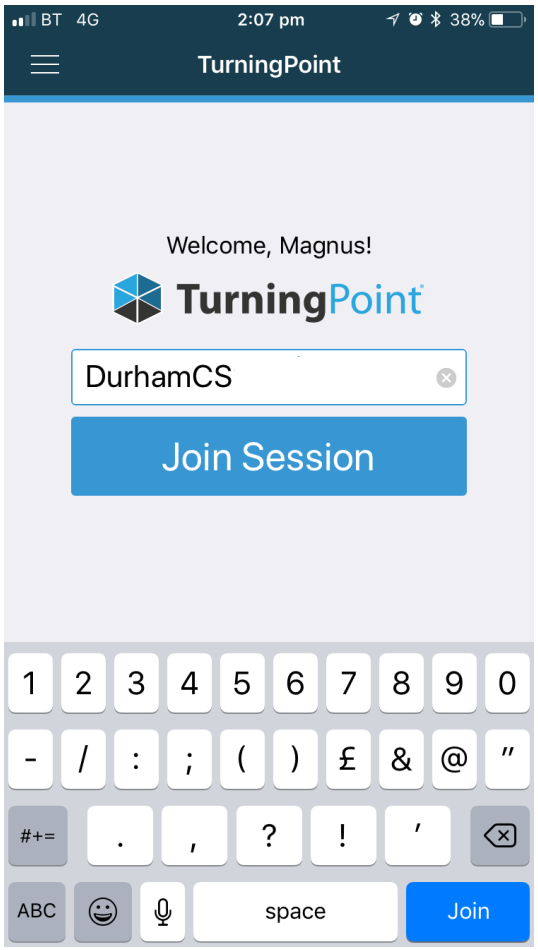
Next term: **Operating Systems** and **Databases**

Details of lecture schedules, practical classes and assignments are on **DUO.**

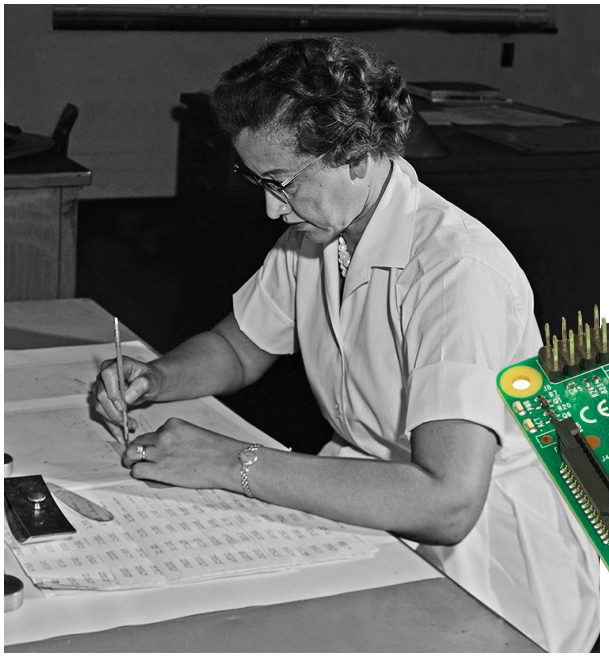
Summative assignment (one that ‘counts’)
LMC Programming: deadline 9th Nov 2pm

TurningPoint

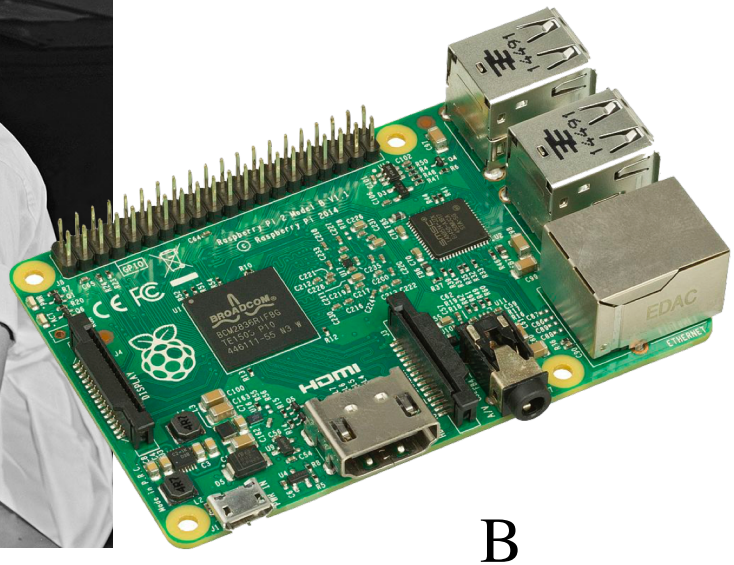




Which of these is a computer?



A

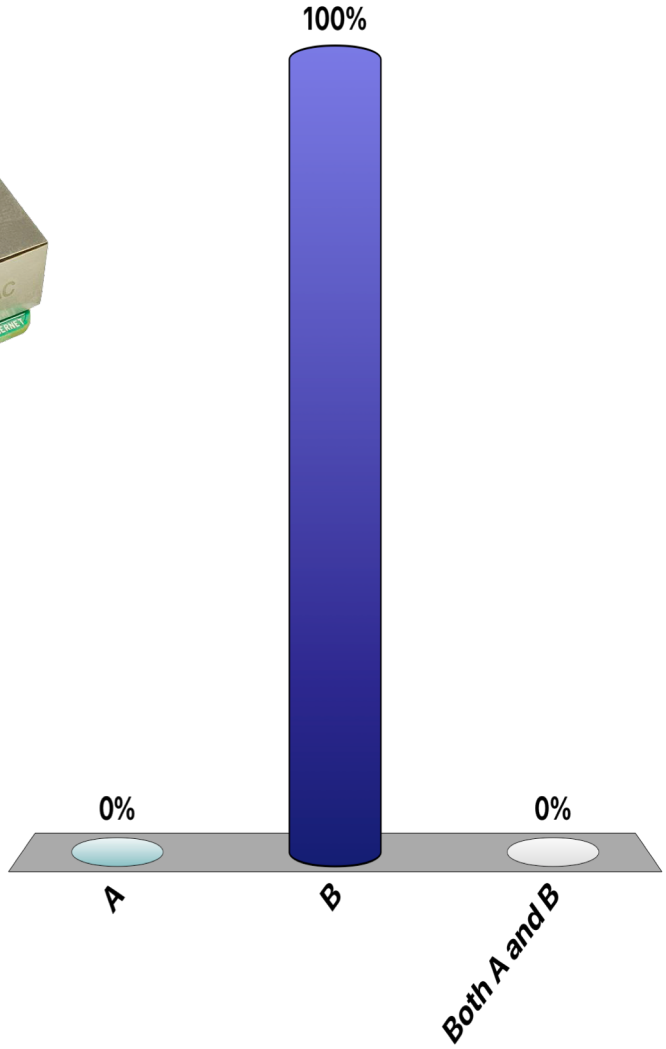


B

A. A

B. B

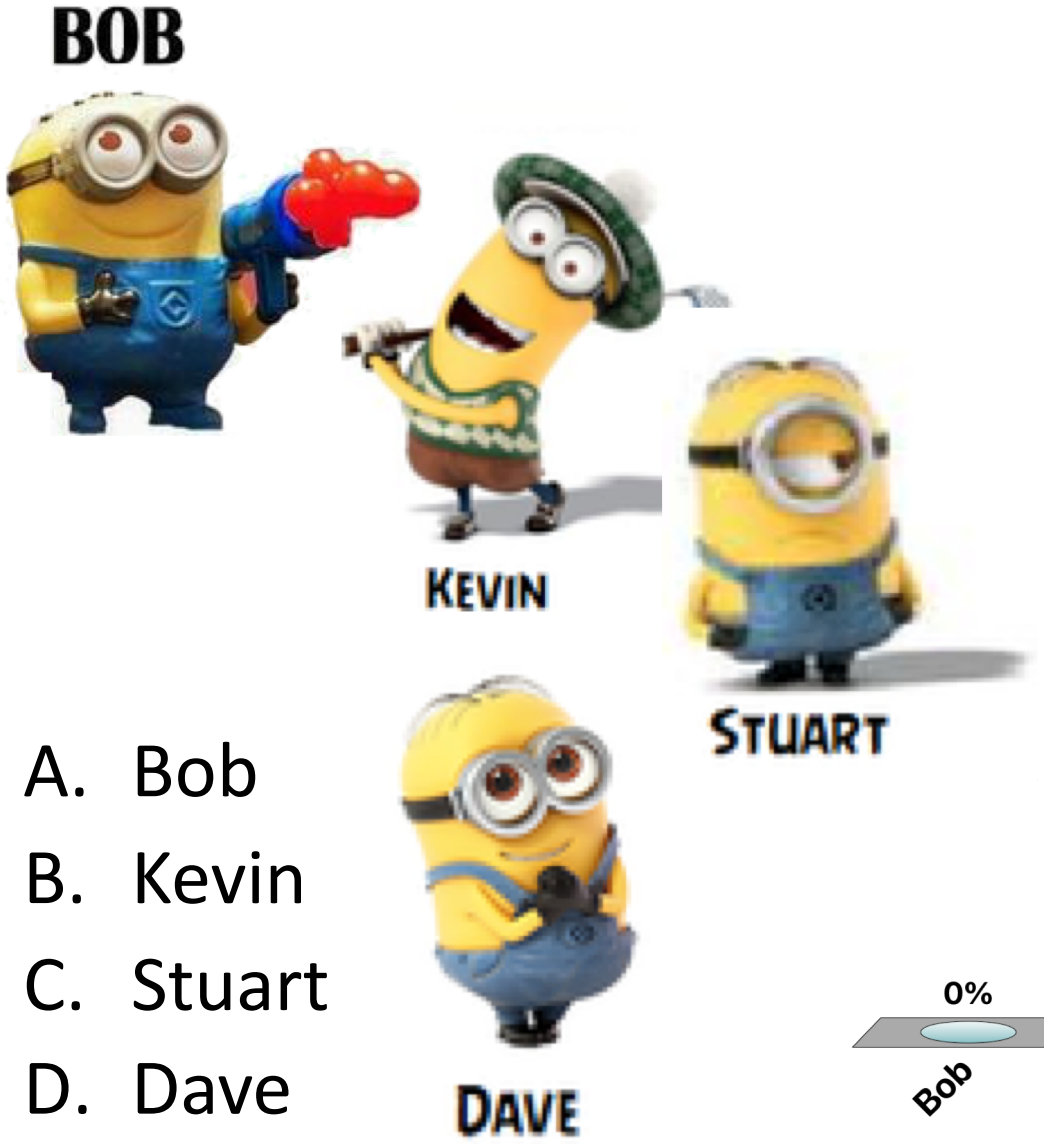
C. Both A and B



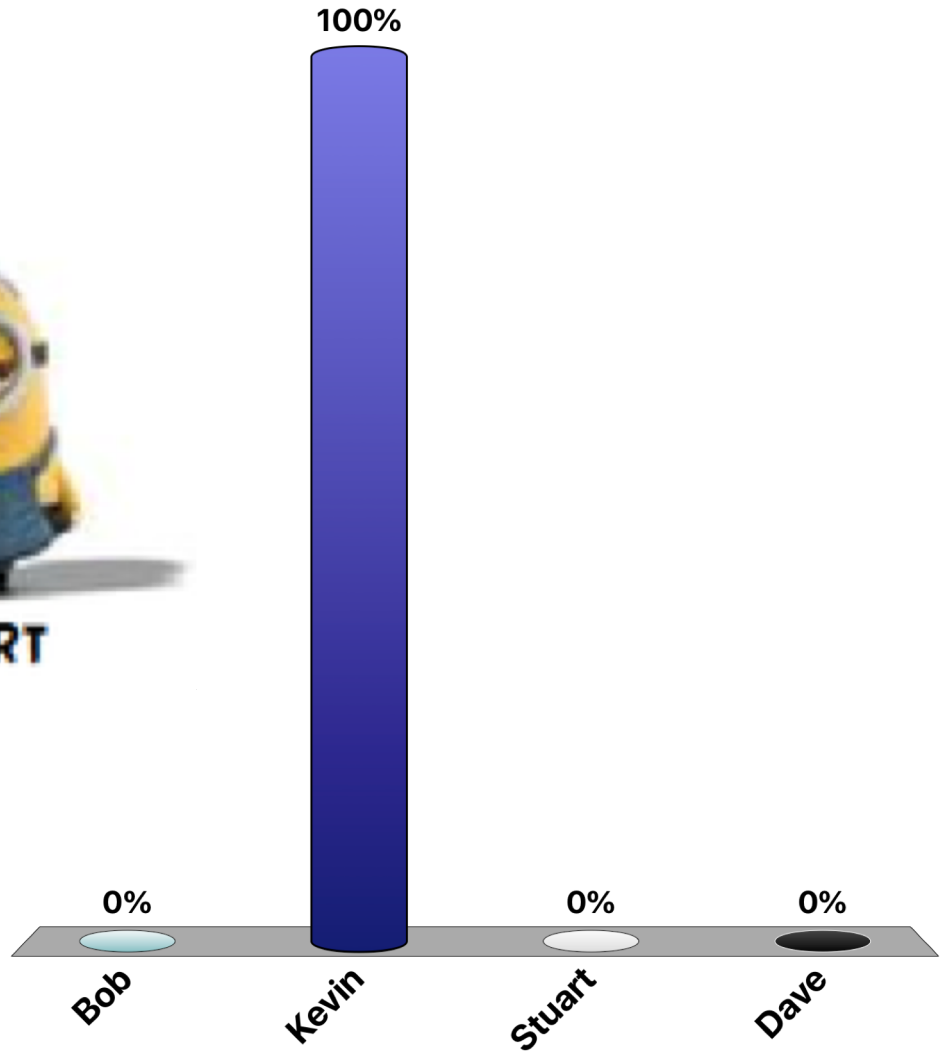
Fastest Responders

Seconds	Participant	Seconds	Participant

Which is your favourite minion?



A. Bob
B. Kevin
C. Stuart
D. Dave

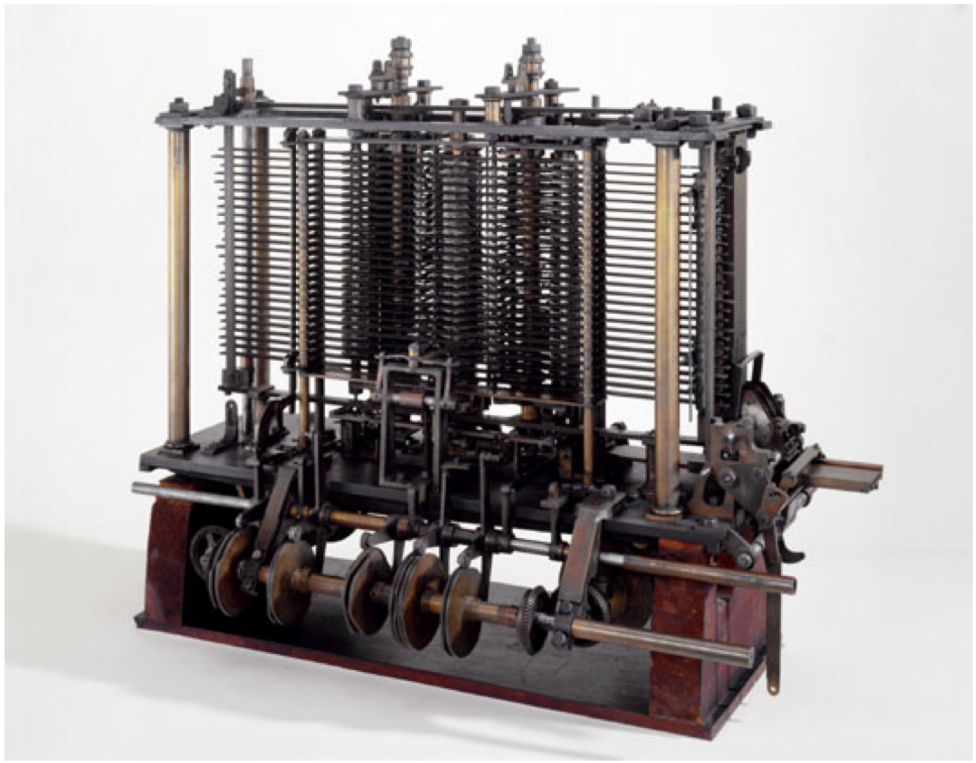


What is a computer?

Pre-1950s:

Human computers in the NACA High Speed Flight Station "Computer Room"
Dryden Flight Research Center Facilities

Babbage's analytical engine 1871



And Ada Lovelace –
the world's first computer programmer?

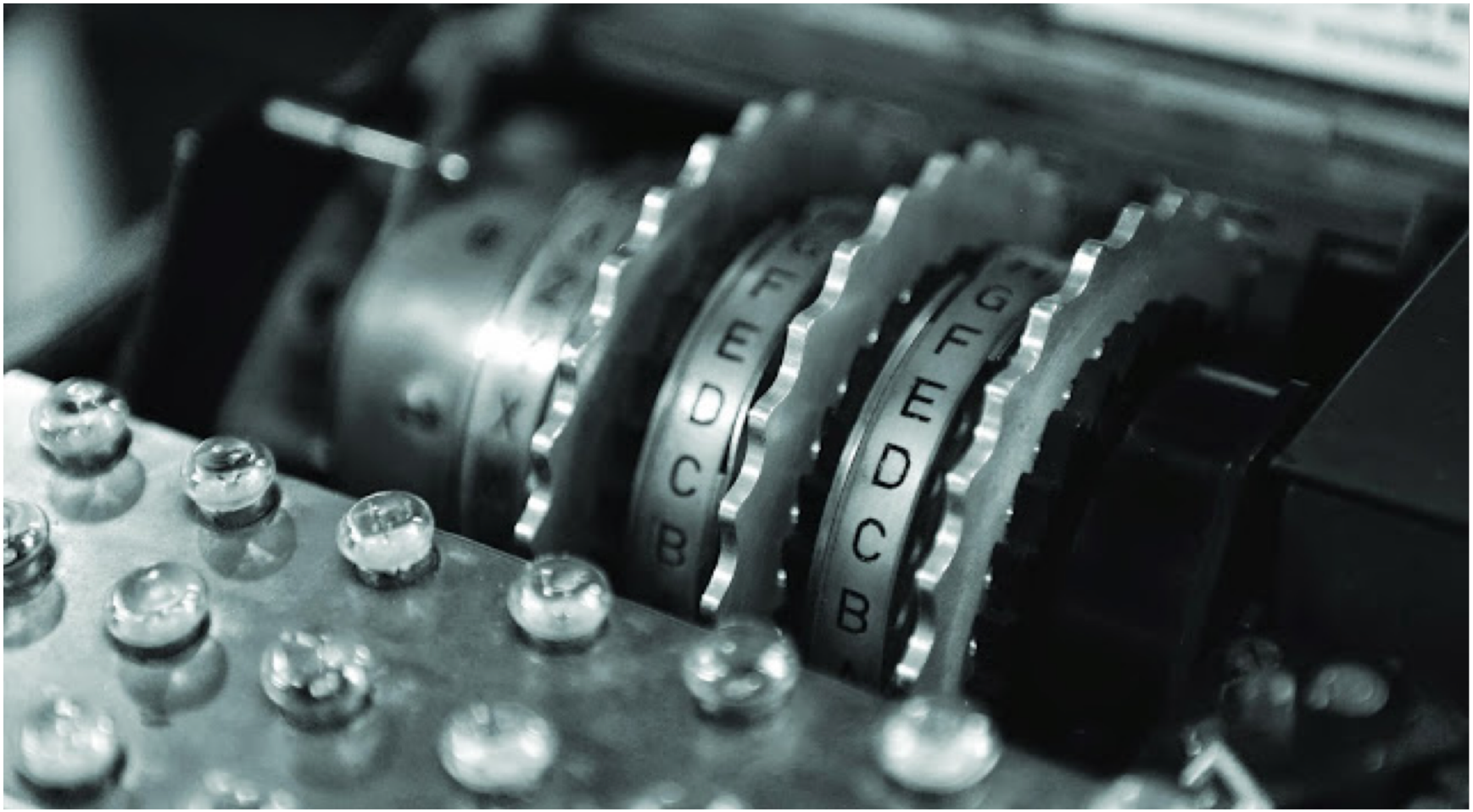
Curta Calculator

Peak of mechanical calculation from 1950-60s

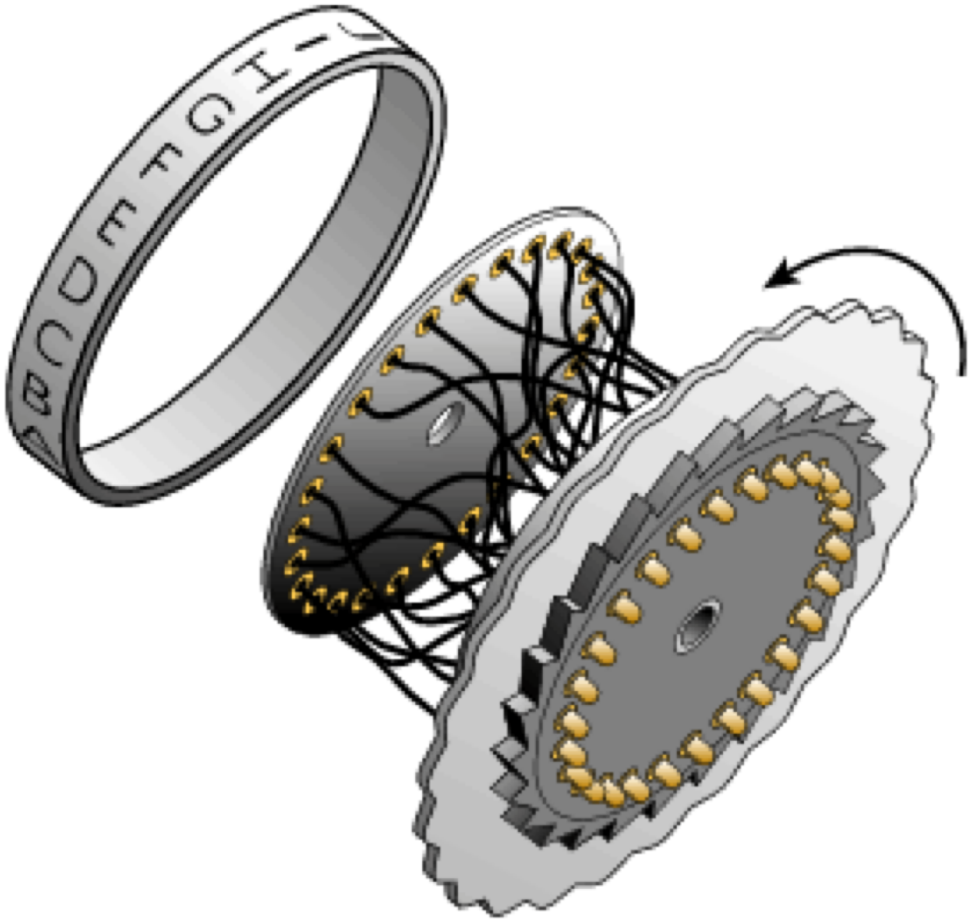
Enigma


Q W E R T Z U I O
A S D F G H J K
P Y X C V B M L
Q W E R T Z U I O
A D F G H J K
X C V B N M L

Rotor wheels



Rotor wheel



Enigma

A **polyalphabetic substitution cipher.**

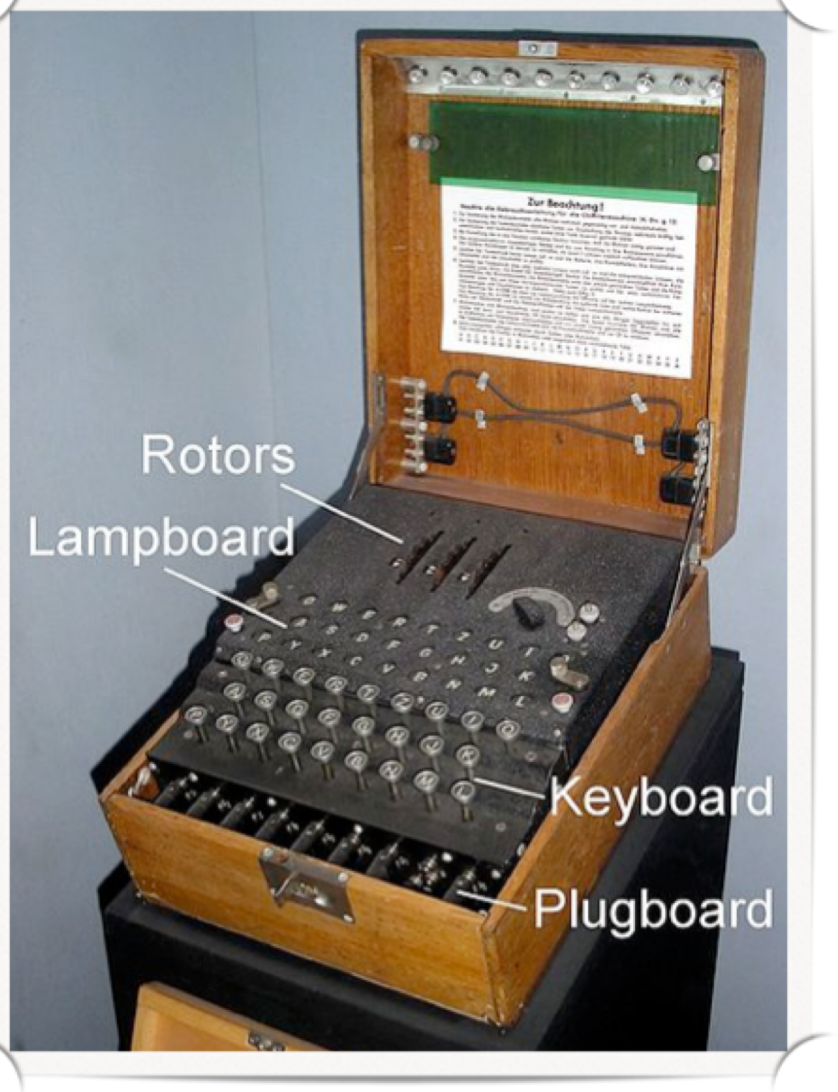
Based on electromechanical **rotors** that changed the substitution cipher after each letter.

Substitution cipher wouldn't repeat for 16900 characters.

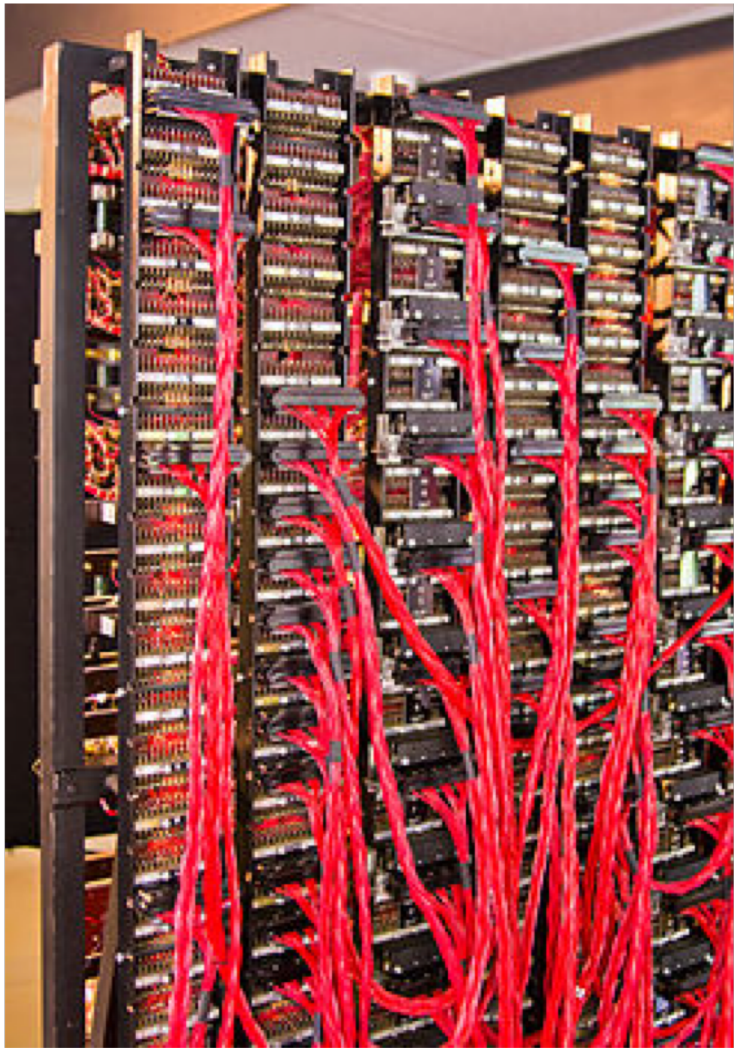
Probably would have been unbreakable if **better procedures** had been followed during use.

Recommended reading:

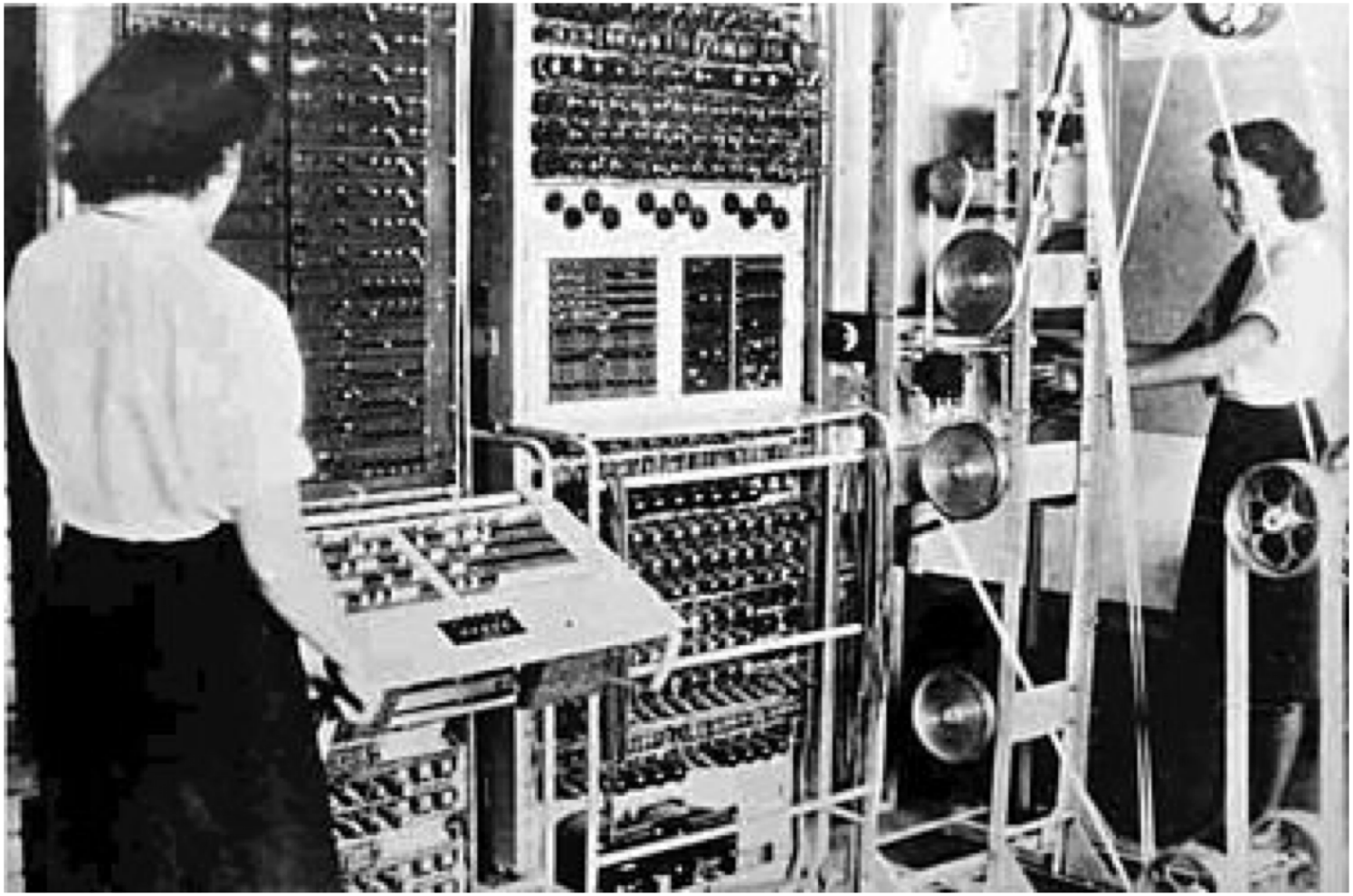
- **Wikipedia** and other internet resources.
- **The Code Book** (Simon Singh).



The BOMBE



Colossus

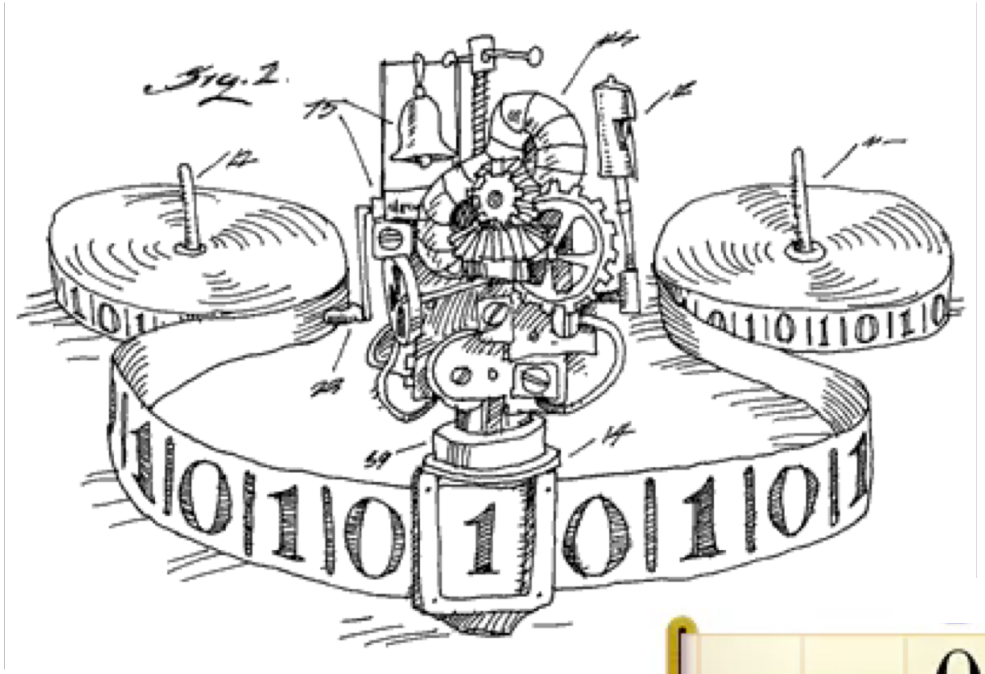


What is a computer?

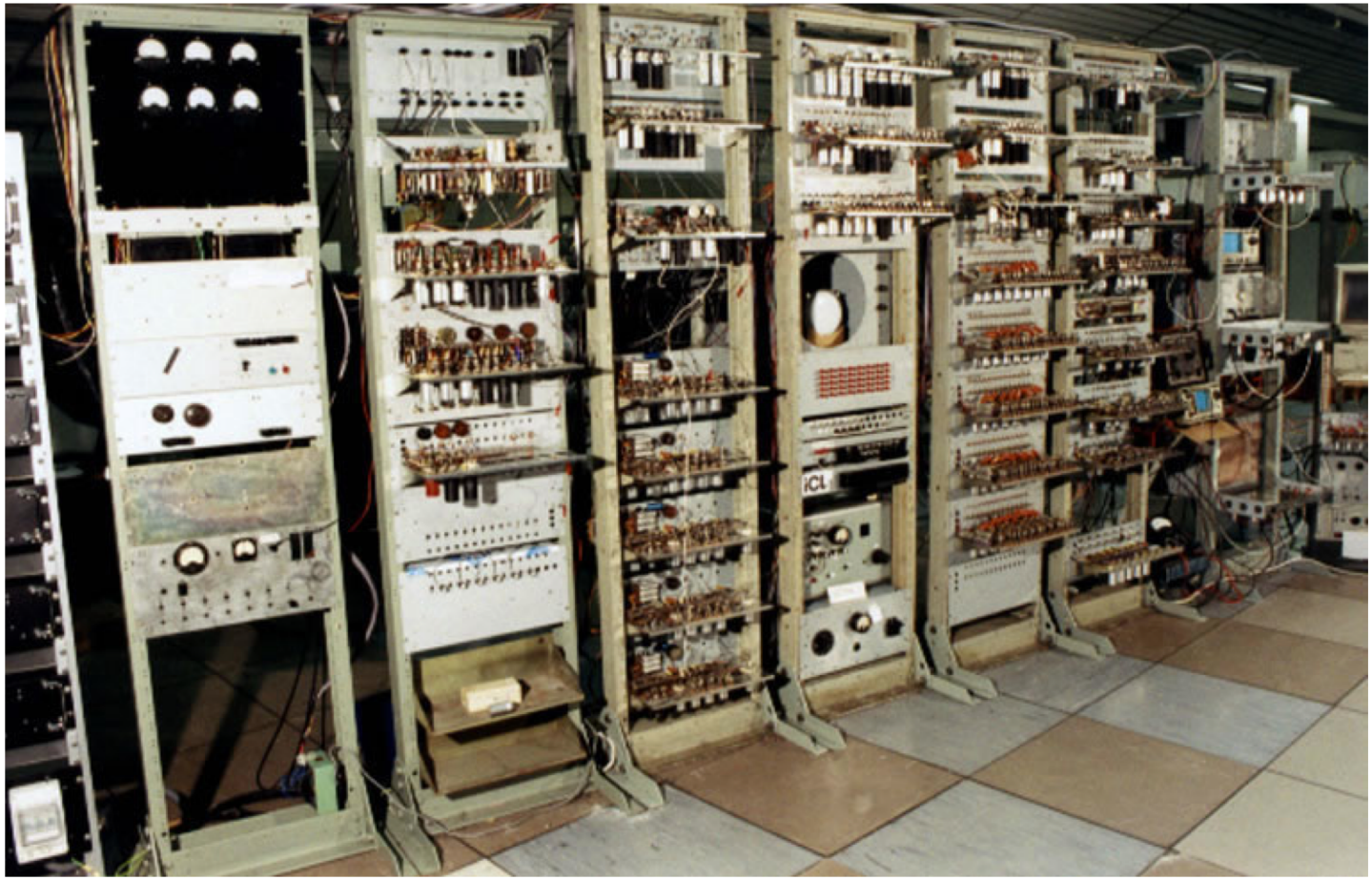
"The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer."

Alan Turing

A Turing Machine



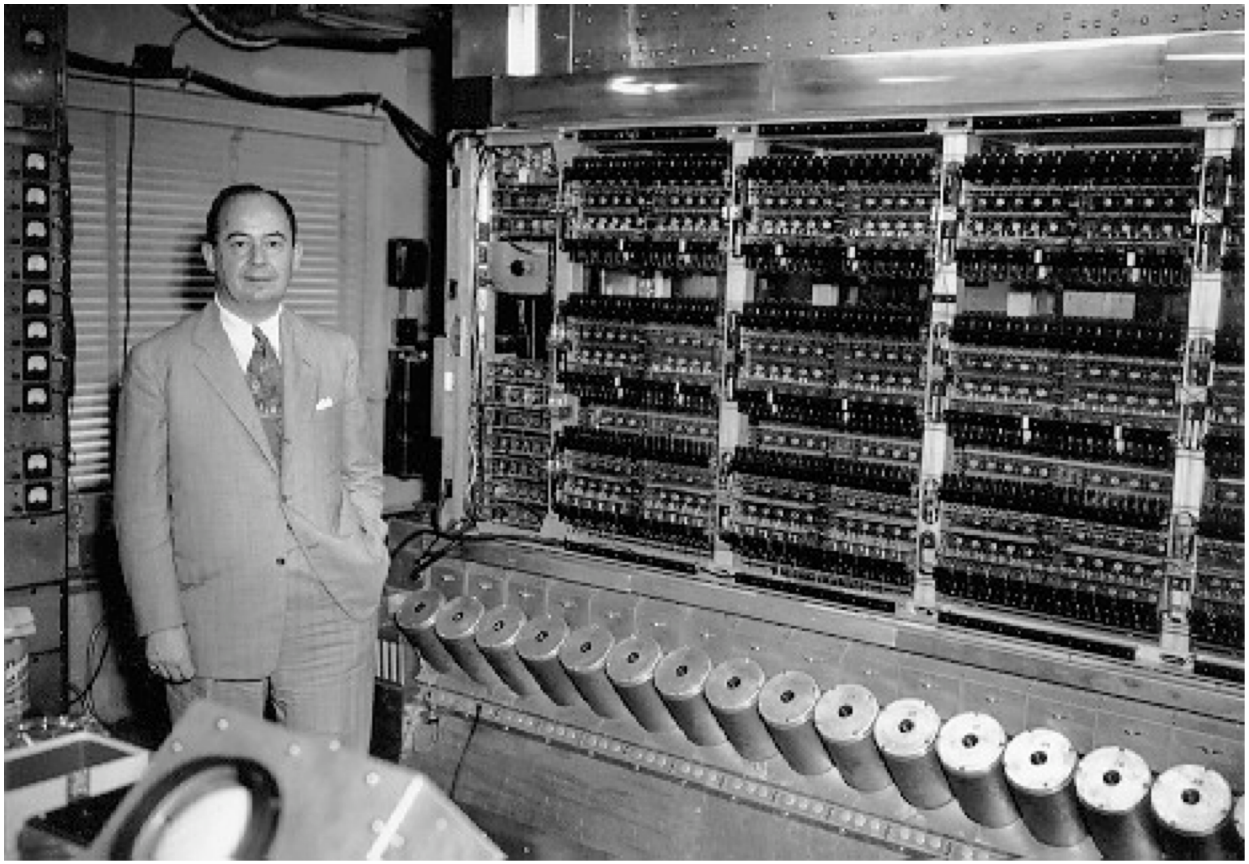
Manchester Baby - 1948



Winner of the race to build the first **Turing-complete** computer

IAS machine - 1952

von Neumann



Perhaps the first more practical computer. Design basis is still the fundamental architecture of all computers.

von Neumann architecture

An architecture of computing devices developed by the EDVAC design team for the U.S. Army 1945-51.

Normally attributed to John von Neumann due to the (leaked) 1945 publication: **First Draft of a Report on the EDVAC.**

The main elements of computers are still virtually unchanged.

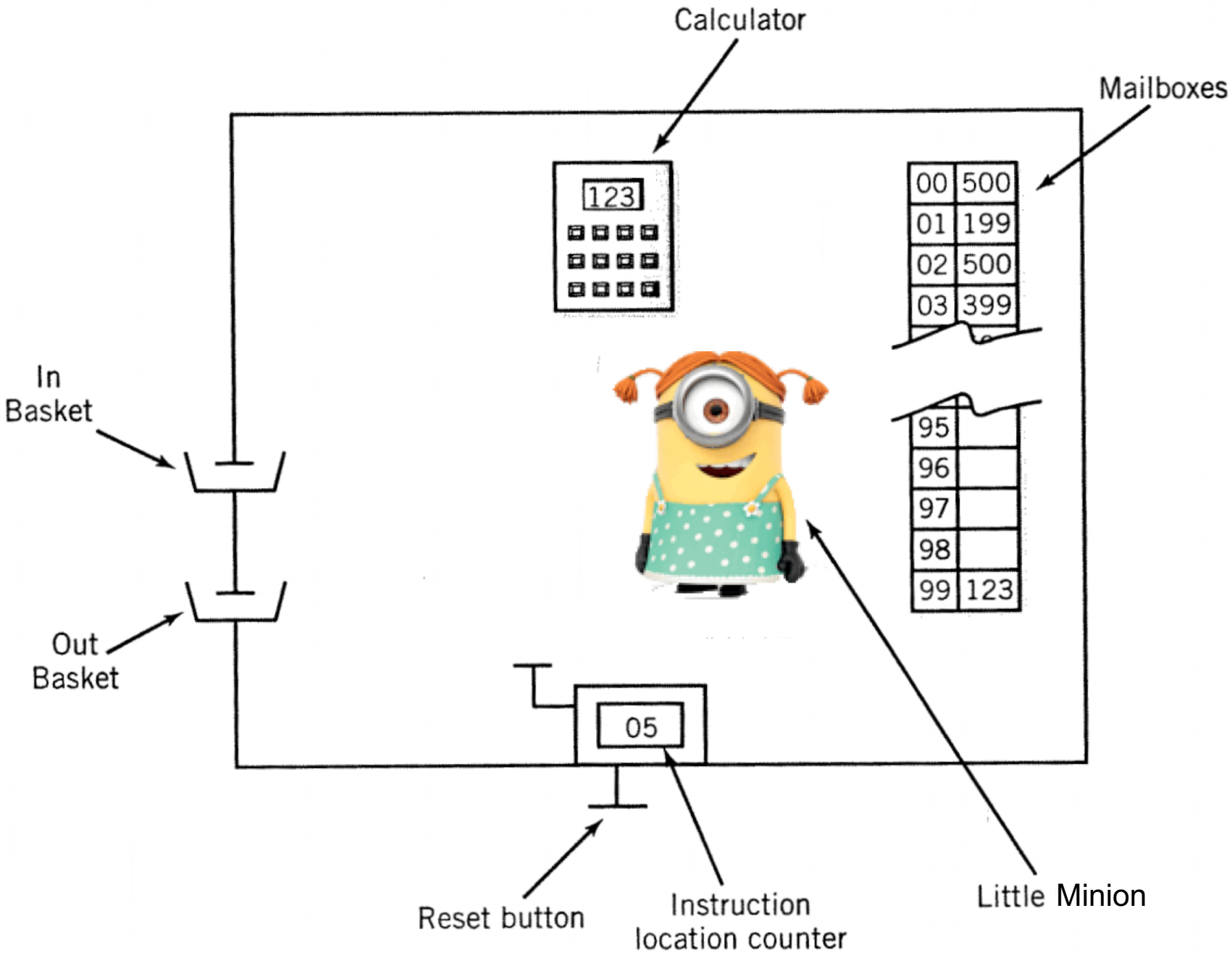
Key concepts:

- Memory holds both programs and data – “Stored Program Concept”.
- Memory is addressed linearly.
- Instructions executed sequentially (unless a branch or reset occurs).

LMC model of computing

Sounds simplistic, but gives insight into the important functions.

Little
Minion
Computer


Calculator
Mailboxes
123
00 500
01 199
02 500
03 399
In Basket
Out Basket
95
96
97
98
99 123
05
Reset button
Instruction location counter
Little Minion

Little Minion Computer

Essential features:

- **Mailboxes** that have a 2-digit address, so 100 mailboxes is the limit
  Each mailbox contains a slip of paper with 3 digits on it.
- **Calculator** that can only display 3 digits, and has 0-9 and +,- buttons, and a flag for negative results.
- **2-digit counter** with two buttons on – the first button increment the counter by 1, the other (external) resets it to 00.
- **Input and output trays** –
  - the user can put slips of paper with 3 digits on in the input tray, to be read when the little minion next looks at his in tray.
  - Likewise the little minion can write 3-digit notes and put them in the out tray, to be read by the user.

Little Minion Computing

The little minion:

- Starts by looking at the counter for a mailbox number X.
- The minion increments the counter by 1.
- The minion goes to mailbox X and reads what is written on the slip of paper in the mailbox – 3 digits.
- The minion takes the appropriate action depending on those digits
- The minion then starts again…

Instruction set

The little minion will read a 3-digit message in the mailbox the minion is sent to.

e.g. 5 8 4

The first digit is the instruction: 5

The second and third digits are another mailbox number: 84

The instruction part is also known as an “operation code” or “op code”

Let’s define some op codes and give the little minion a task.

LMC instruction set

LOAD – op code 5

go to the mailbox address specified, read the 3-digit number at that address, then go to the calculator and enter that number in.

Note: the mailboxes will still contain the same 3-digit messages as before, but the calculator will have whatever was there replaced.

LMC instruction set

STORE – op code 3

go to the calculator and note the 3-digit number displayed, then go to mailbox address specified and enter that number on a new slip of paper.

Note: the calculator will still contain the same 3-digits as before, but the mailbox will have whatever was there before discarded.

LMC instruction set

ADD – op code 1

go to the mailbox address specified, read the 3-digit number at that address, then go to the calculator and add the number to the number already on the calculator.

Note: the mailboxes will still contain the same 3-digit messages as before, but the calculator will have changed.

LMC instruction set

SUBTRACT – op code 2

go to the mailbox address specified, read the 3-digit number at that address, then go to the calculator and subtract the number from the number already on the calculator.

Note: the mailboxes will still contain the same 3-digit messages as before, but the calculator will have changed. If the calculator ends with a negative value, there is some flag to indicate this.

LMC instruction set

INPUT/OUTPUT – op code 9

INPUT or READ op code 9 address 01

go to the IN tray, read the 3-digit number there, then go to the calculator and enter that number in.

Note: The slip of paper in the IN tray with the 3-digits is removed. If there are several slips in the IN tray, the little minion takes the first one put there only, the other remain for future visits.

OUTPUT or PRINT op code 9 address 02

go to the calculator, read the 3-digit number there, then go to the OUT tray and leave a slip of paper with that number on it.

LMC instruction set

BREAK – op code 0

The little minion has a rest.

LMC instruction set

Data movement:

LOAD – op code 5

STORE – op code 3

Arithmetic:

ADD – op code 1

SUBTRACT – op code 2

Input/Output:

INPUT – op code 901

OUTPUT – op code 902

Machine control:

BREAK – op code 0

LMC example

Can we get the Little Minion to add two user-inputted numbers for us, and tell us the answer?

Of course.

Mailbox	code	Description
00	901	INPUT 1st #
01	399	STORE DATA
02	901	INPUT 2nd #
03	199	ADD 1st
04	902	OUTPUT
05	000	STOP
99		DATA

Extended Instruction Set

To make the programs the little minion can execute more flexible, we need **branching** and **looping**.

BRANCH – op code 6

set the counter to the 2-digits specified in the address, and start fetch of instruction.

Note: we assume the little minion can adjust the counter. The next instruction the minion reads is the one in the mailbox specified.

Extended Instruction Set

BRANCH on ZERO – op code 7

go to the calculator and read the 3-digit number. If it is zero, set the counter to the 2-digits specified in the address, and start fetch of instruction, otherwise continue with the next instruction as normal.

Note: we assume the little minion can adjust the counter. The next instruction the minion reads is the one in the mailbox specified.

Extended Instruction Set

BRANCH on POSITIVE – op code 8

go to the calculator and read the 3-digit number. If it is positive (including zero), set the counter to the 2-digits specified in the address, and start fetch of instruction, otherwise continue with the next instruction as normal.

Note: This is why the calculator must have a flag for negative values.

Branching example

Output the smaller of two numbers:

Mailbox	code	Description
00	901	INPUT 1st #
01	311	STORE DATA
02	901	INPUT 2nd #
03	312	STORE DATA
04	211	SUBTRACT 1st
05	808	BR on +ve
06	512	LOAD 2nd
07	609	BR to 09
08	511	LOAD 1st
09	902	OUTPUT
10	000	STOP

A Loop

```
value = INPUT
do while value >=0
    print value
    value --
next
print 999
end
```

Mailbox	code	Description
00	901	INPUT value
01	902	OUTPUT
02	299	SUBTRACT 1
03	705	BRZ
04	601	BRANCH
05	598	LOAD
06	902	OUTPUT
07	000	STOP
98	999	DATA - 999
99	001	DATA - 1

Instruction set & mnemonics

Mnemonic	op code	Description
LDA	5xx	Load
STO	3xx	Store
ADD	1xx	Add
SUB	2xx	Subtract
IN	901	Input
OUT	902	Output
HLT	000	Halt or Stop
BR	6xx	Branch
BRZ	7xx	Branch on zero
BRP	8xx	Branch on positive
DAT		Data storage location

LMC – fetch, execute cycle

The steps taken by the little minion are called the **instruction cycle**.

There are two essential phases of the instruction cycle:

- **fetch** – in which the little minion finds the instruction to execute
- **execute** – in which the minion performs the work specified in the instruction.

fetch:

- The minion goes to the counter and reads the address there
- The minion goes to the mailbox at that address and reads the 3-digit number in the mailbox

LMC – fetch, execute cycle

The steps taken by the little minion are called the **instruction cycle**.

There are two essential phases of the instruction cycle:

fetch – in which the little minion finds the instruction to execute

execute – in which the minion performs the work specified in the instruction.

execute: e.g. STORE

The minion remembers the target address

The minion goes to the calculator and reads (remembers) the 3-digits

The minion goes to the mailbox at the remembered address

The minion writes the 3-digits on a slip at that address

The minion goes to the counter and increments it.

The minion needs to **remember** an address and a 3-digit value – MEM address, data

von Neumann architecture

An architecture of computing devices developed by the EDVAC design team for the U.S. Army 1945-51.

Normally attributed to John von Neumann due to the (leaked) 1945 publication: **First Draft of a Report on the EDVAC.**

The main elements of computers are still virtually unchanged.

Key concepts:

- Memory holds both programs and data – “Stored Program Concept”.
- Memory is addressed linearly – there is a number for every mailbox.
- Memory is addressed by location, the contents is irrelevant.
- Instructions executed sequentially, unless a branch or reset occurs.

Harvard Architecture

Separate memory for instructions and data

- Quicker to execute as can access instruction and data at the same time
- Simpler to follow / analyze code
- Avoids the potential for some malware / bugs due to self-modifying code

Most modern processors have a CPU cache which partitions instruction and data, giving a ‘modified Harvard architecture’ giving the best of both worlds.

Design a LMC

Input two numbers and output all integer values between these two numbers.

Pseudo-Code of Solution:

Input first number as (assumed) smaller
Input second number as (assumed) larger
IF larger < smaller
--- exchange larger and smaller values
END_IF
Add 1 to smaller
WHILE smaller < larger
--- Output smaller
--- Add 1 to smaller
END_WHILE
Stop

Examples

Division:

Take two inputs, a and b, and compute a divided by b (rounded down).

Greatest common divisor:

Euclid's algorithm. The largest common divisor of two numbers a, b is unchanged if the smaller is subtracted from the larger: a, (b-a).