4

Computers go to War

In September 2009 Gordon Brown, the British Prime Minister, issued a statement on Downing Street's official website. The statement read:[1]

> "*...Thousands of people have come together to demand justice for Alan Turing and recognition of the appalling way he was treated. While Turing was dealt with under the law of the time and we can't put the clock back, his treatment was of course utterly unfair and I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him... Alan deserves recognition for his contribution to humankind... It is thanks to men and women who were totally committed to fighting fascism, people like Alan Turing, that the horrors of the Holocaust and of total war are part of Europe's history and not Europe's present... Without his outstanding contribution, the history of world war two could well have*

[1] The full text of the apology is in the Appendix

been very different… The debt of gratitude he is owed makes it all the more horrifying, therefore, that he was treated so inhumanely. In 1952, he was convicted of gross indecency – in effect, tried for being gay… His sentence – and he was faced with the miserable choice of this or prison – was chemical castration by a series of injections of female hormones… So on behalf of the British government, and all those who live freely thanks to Alan's work I am very proud to say: we're sorry, you deserved so much better."

What had caused this most unusual government apology?

The Turing Machine

Alan Mathison Turing was born in 1912 in Maida Vale London, the son of a British civil servant in India. Like many children of the Empire he was sent at the age of thirteen as a boarder to a public school, called Sherborne,[2] in rural Dorset. My elder brother went to Sherborne so I know the school well.

Although he excelled at math, he rarely got good marks as he spent most of his time on advanced study of his own rather than attending to the elementary tasks for which he was being graded. Sherborne believed that the classics were the proper subjects for young gentlemen to study. His work was also always very poorly presented, as if his mind was on other matters and right to the end of his

[2] Note that in England *public* schools are exclusive fee paying *private* schools.

life his notebooks were always a mess. In addition to mathematics Turing was a keen athlete and particularly enjoyed long distance running.

Alan Turing

After Sherborne Turing went up to King's College Cambridge, where he studied under some brilliant mathematicians. Although shy and awkward, Turing could hold his own intellectually amongst them, he graduated with first class honors and in 1934, at just 22, was elected a fellow of King's College. The fellowship came with an annual stipend of £300, enough to enable him to follow an academic career.

He started to work on the *Entscheidungsproblem*, or in English, the *decision problem*. Mathematics relies on formal proofs; a complex mathematical statement or formula can be proved to be correct by proving that the formulae upon which it depends are correct, and proving that statements upon which they depend are correct and so on. I'm sure you've seen movies and TV shows were professors cover blackboards with complex formula and finally ecstatically exclaim "*QED!*"

QED is an abbreviation from the Latin "*quod erat demonstrandum*" that means, "*what was to be demonstrated*." The phrase is used by mathematicians and logicians to show that a proof has been completed.

In the 1930s mathematics and the closely related discipline of logic were in a spot of trouble. A brilliant young German, called Gödel, had proven that our mathematical and logical systems were either *complete* (one should be able to prove or disprove every statement) or they were *consistent* (one could not prove a statement that is false or disprove a statement that is true). But, importantly they could not be both complete and consistent.

"*Hold on a minute,*" you're saying, "*I don't understand, and why is this important anyway?*" Well, actually, you do understand, which I can demonstrate through a simple example called the "*liars paradox*".

All statements on this page are false.

The Liars Paradox

This is a popular brainteaser with college kids and stoners, invented in the fourth century BC by a Cretan philosopher, called Epimenides. Let's consider the statement. If the statement is true, then all statements are false, then the statement can't be true, it must be false, in which case it's true, but then it has to be false and so on and so on. You can think about this until it makes your brain hurt, but you can never reach a conclusion.

This paradox was famously used in an episode of *Star Trek* called "*I, Mudd*" in which Harold Mudd destroys an android by forcing it to think about a version of the paradox. As the android goes through the logical consequences of the paradox it's voice gets faster and faster until smoke comes out of its head and finally it explodes. So you see that a logical statement can have a powerful effect if it can't be proved. You might argue that we should just disallow such paradoxes and indeed that is largely what we do. We create logical or mathematical models that are either complete, where every statement can be proved or disproved, or are consistent where true statements can't be disproved and false statements can't be proved.

The mathematical and philosophical establishment, were horrified by Gödel's discovery. It seemed as if the very foundations of their universe had been torn away, could nothing be relied upon? As the mathematician Andre Weil said:

"*God exists since mathematics is consistent, and the Devil exists since we cannot prove it.*"

However, young mathematicians, like Turing, saw the new orthodoxy as fertile territory for research. The *Entscheidungsproblem* relates to completeness and consistency, in that to test for either you have to be able to reach a decision. Put simply, the decision problem is, "*is it possible to reach a decision that a statement is true or false.*" So we might say: "*is 10 exactly divisible by 2?*" The answer, of course is "*yes.*" "*Is 10 exactly divisible by 3?*" and the answer is "*no*."

For simple statements the decision problem is easy, you can do it in your head, but for increasingly complex statements it will become harder and we might need to use a formal way of coming to a decision. These formal methods are called *algorithms* - you don't need to be scared of algorithms here's a simple one.

Euclid's Algorithm calculates the greatest common factor (GCF) of two numbers. Consider the following: what is the GCF of 252 and 105? The greatest common factor is the largest whole number that will exactly divide into both numbers and Euclid found a simple, step-by-step, repetitive procedure, an algorithm, for solving it.

First we divide the larger number by the smaller:

252 ÷ 105 = 2 with a remainder of 42

Now we divide the smaller of the two initial numbers by the remainder.

105 ÷ 42 = 2 with a remainder of 21

Now we divide the smaller of the two previous numbers by 21.

42 ÷ 21 = 2 with a remainder of 0

QED. 21 is the greatest common factor of 252 and 105.

We can write this algorithm down more formally as:

```
a/b gives a remainder of r
b/r gives a remainder of s
r/s gives a remainder of t
…
w/x gives a remainder of y
x/y gives no remainder
```

y is the GCF of a and b. If the first step produces no remainder, then b (the lesser of the two numbers) is the GCF.

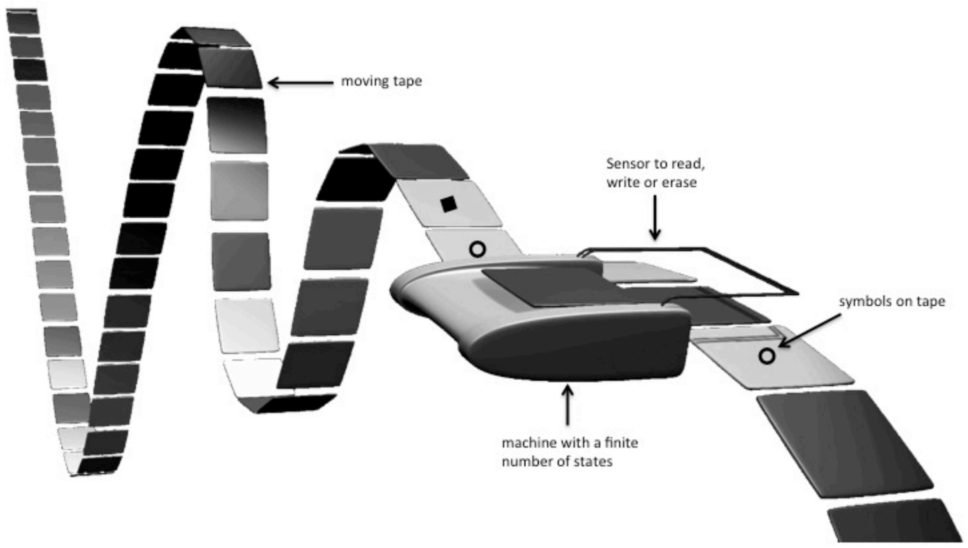
Or even more succinctly in computer code as:

```
function gcf(a, b)
    while b ≠ 0
        t := b
        b := a mod b
        a := t
    return a
```

Using this algorithm we can easily find the greatest common factor between any two numbers. Turing saw that by applying simple algorithms he might be able to satisfy the decision problem. The problem was how to make the algorithms simple enough, yet universal enough, to work on any decision problem, or indeed on all decision problems.

In 1936, whilst studying for a PhD at Princeton in the US, Turing published his breakthrough paper: "*On Computable Numbers, with an application to the Entscheidungsproblem*" in the *Proceedings of the London Mathematical Society*. Turing had an idea; he imagined a hypothetical factory filled with floor upon floor, and rows upon rows of hundreds upon hundreds of computers, each using an algorithm to solve a particular decision problem. Such a factory could in theory solve the decision problem for all problems. He actually at first envisages the computers as women meticulously following algorithms to solve problems like the common factor problem above. Remember, at this time as we've already seen, computers were people who computed.

However, this is where he makes his imaginative leap, realizing that people would be too slow and error prone to realistically tackle the decision problem he imagines a machine that could compute. Turing doesn't imagine a giant mechanical calculating engine like Babbage, nor even an electronic machine, but a purely hypothetical machine now called a "*Turing Machine*."


moving tape
Sensor to read, write or erase
symbols on tape
machine with a finite number of states

A Turing Machine

A Turing Machine is an imaginary machine that can carry out all sorts of computations on numbers or any type of symbols. Turing saw that if you could write down a set of rules describing a computation (an algorithm) then his machine could accurately carry it out. Thus, a Turing Machine is at the center of the modern theory of computation even though it was invented before the creation of the first computer. A Turing Machine consists of:

• an Input/Output tape,

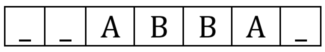
• the Turing Machine itself, and

• a set of rules.

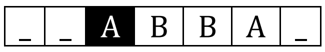
The tape is a roll of paper that is infinitely long and can be moved forwards and backwards. The tape is divided into cells. The cells can contain symbols.

Above the tape sits a device that can read from, write to, or erase a cell on the tape. The machine can move the tape one cell to the left or right. The machine has an internal state that can be changed. The machine can use the tape as a memory by writing into a cell; since the tape is infinite the machine's memory is infinite. The set of rules, or algorithm is what determines the machine's move at any particular point based on its internal state.

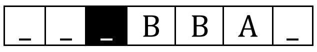
Let's see how a Turing Machine works for a simple example. We want to decide if a string of characters is a palindrome; does the word read the same in either direction. "ABBA" is a simple palindrome; "*Able was I ere I saw Elba*," is a more complex and famous palindrome.



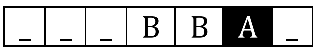
Consider the simple tape above with the word "ABBA" on it; just as with Euclid's Algorithm earlier we can break the search for a palindrome down into a series of simple, repetitive, steps. For our word to have a chance of being a palindrome the first and last letters *must* be the same, if they are not then it cannot be a palindrome. So our Turing Machine can step through the tape starting from the left until it detects the first character "A."



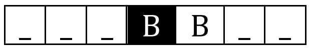
It erases the "A" and then applies a rule looking for an "A" at the end of the word. It advances cell by cell through the tape until it detects the last character.



It detects the last character by finding the first blank cell and then moving the tape one cell back to the left.



If the character in the last cell is an "A" our word may be a palindrome. The machine then erases the "A" and moves back to the left until it finds the new first character.



The machine then repeats the steps above, but now on the shorter word “BB”. If that is a palindrome, which it is, then our whole word is a palindrome. This algorithm will work for any length palindrome.

_	_	A	B	L	E	_	W	A	S	_	I	_	E	R	E	_	I	_	S	A	W	_	E	L	B	A	_	_	_

The only changes we need to make is first, we need to detect two blank cells to find the end of the phrase instead of just one blank to find the end of a word. Second, we need to have rules for each letter of the alphabet. However, the principle is the same and the algorithm will work as before, comparing the first and last letters, erasing them and then moving on to compare the new first and last letters, until it is determined that the phrase is a palindrome or not.

_	_	A	B	L	E	_	W	A	S	_	I	_	E	R	E	_	I	_	S	A	W	_	E	L	B	A	_	_	_

_	_	_	B	L	E	_	W	A	S	_	I	_	E	R	E	_	I	_	S	A	W	_	E	L	B	_	_	_	_

_	_	_	_	L	E	_	W	A	S	_	I	_	E	R	E	_	I	_	S	A	W	_	E	L	_	_	_	_	_

A Turing Machine couldn’t use our wordy description of the algorithm, so we need to translate our algorithm into code for an actual Turing Machine. This code will determine if any word comprised of As & Bs (like ABBA) is a palindrome and print “YES” or “NO” as its answer on the tape.[3]

In our ABBA example above we used just As and Bs, but we might as easily use 1s and 0s. If we do, then we are using binary, which is of course the language

[3] You can see a Turing Machine simulator decide if words are palindromes at http://ironphoenix.org/tril/tm/

that all modern computers use. Turing Machines can be created to solve any problem that can be represented by symbols.

Turing had another great insight. He realized that he didn't have to have a separate Turing Machine for each problem; one to detect palindromes another to solve Euclid's Algorithm and so on. You could create a single *universal* Turing Machine that by combining the algorithm and the data on the paper tape could solve *any* problem!

This was Turing's great breakthrough. He had shown that a very simple computational machine could theoretically solve any problem using very simple instructions. *"We may now construct a machine to do the work of this computer,"* he writes towards the end of his paper, and. *"It is possible to invent a single machine which can be used to compute any computable sequence."* Such a machine would be a *Universal Turing Machine*.

Consider for a moment, Turing had envisaged a single machine that could do any task that could be programmed into it. This may not seem like a very big idea to you because you've grown up with computers. Your computer is a single machine that can do: word processing, spreadsheets, send email, surf the web, play games, play music and videos, edit photos, and a host of other things. But, in the 1930s, as we saw in the previous chapter, people used different machines for different tasks. Turing's *universal machine* with its stored program concept was a revolutionary idea – one machine that could do anything! This is why Turing is known as the "*father of computing*."

There is however one limitation. Our Turing Machine with paper tape runs very slowly, but even if it ran at the speed of light, some problems might never be completed. It may be impossible to come to a decision for all problems. Now there's no point in waiting around for eternity for an answer that may never come. It would be useful to determine in advance if a problem is likely to complete. This is called the "*Halting Problem*," it has not been solved yet. Turing showed that a general algorithm to solve the halting problem for all possible program-input pairs couldn't exist. The halting problem is *undecidable* over Turing Machines, so he had proven that the *Entscheidungsproblem* cannot be solved. Gödel was correct; all mathematical statements are not decidable.

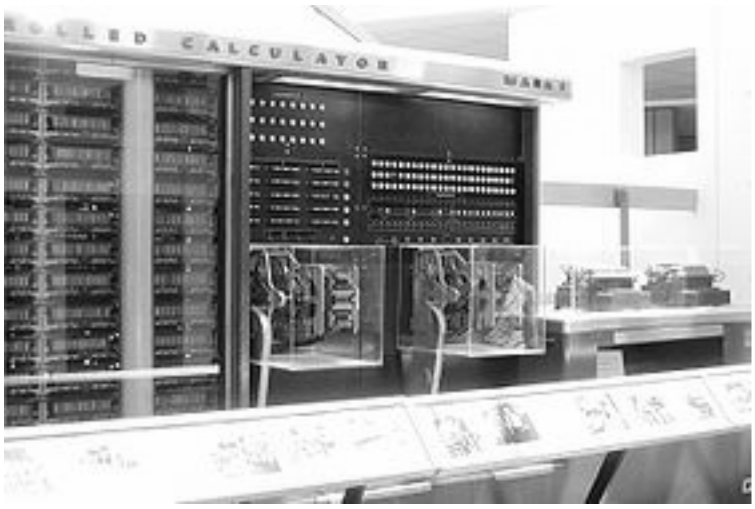
However, fortunately for us we don't need to worry about arcane mathematical theories since many practical and useful problems can be computed in a reasonable time. Thus, at the core of modern computers, are lots of simple tiny machines, performing very simple computations based upon strict concise instructions. In fact it is now almost impossible for us to think of a Turing Machine without thinking of a computer and its hardware and software.

If you search online you can find several examples of simulations of Turing Machines there is even a YouTube clip of a Turing Machine made from Lego that actually works.

Total war

By the start of World War II the navies of the world were dominated by huge battleships with guns so large they could fire shells the size of VW Beetles 20 miles or more. To aim these guns the gunners were dependent on firing tables published in gunnery manuals. The preparation of these tables was laborious as the complex equations were solved by human computers. 100 years after Babbage the US and British navies faced the same old problem of generating tables fast, but without errors. This problem was replicated for all the land-based artillery used by the huge armies fighting all around the world.

During the war the US military hired every person, usually female math majors, who had the skills to create these firing tables. Don't forget new tables were required for every new type of artillery and shell. Because there actually weren't that many female math graduates the US was not able to produce enough tables and pieces of artillery were sometimes shipped to the battlefield without firing tables. Without an accurate firing table an artillery piece was virtually useless since it couldn't be aimed accurately. The US military were desperate to remedy this situation and they turned to their best university, Harvard, and their biggest tabulation company, IBM, who between them by 1944 developed the Harvard Mark 1.

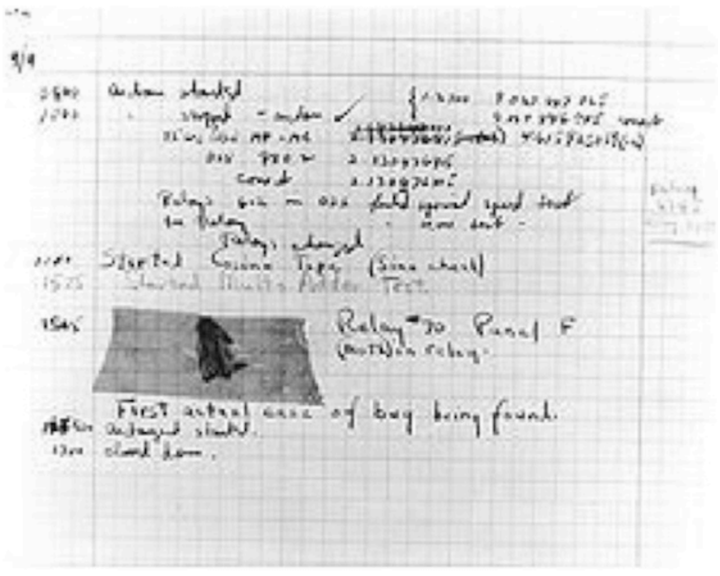


A portion of the right side of the Harvard Mark 1

Designed by Howard H. Aiken, the Mark 1 was programmable, like Babbage's Analytical Engine, but it was not an electronic computer. Instead it was an electromechanical hybrid of switches, relays, rotating drive shafts, gears and clutches. The machine was massive, like the Analytical Engine, weighing five tons and standing eight feet tall and 50 feet long. A rotating shaft ran its entire length powered by a five horsepower motor. You can get a sense of its scale by noticing the two typewriters to the right of the picture. The noise the machine made was apparently quite deafening.

It's difficult for us now to understand just how limited this huge electromechanical machine was. Despite its huge size and quarter of a million components the Mark 1 could only store 72 numbers. Your smartphone today can store billions of numbers. The Mark 1 could add, subtract, multiply and divide numbers of over 20 digits in a matter of seconds, but today, your computer can perform calculations in a billionth of a second. Speed was just not possible from a machine with mechanical components - it was faster and more reliable than people though.

In a strange parallel to the Analytical Engine and Ada Lovelace a woman was also closely involved with the Mark 1. Grace Hopper was one of those rare female math graduates with a PhD from Yale. She was *volunteered* into the Navy in 1943 and quickly ended up on the team developing the Mark 1. Hopper was one of the first programmers of the Mark 1 and is credited with finding the first computer bug. A moth had flown into the Mark 1 and got caught in a relay causing an error – literally a bug! The remains of the bug can still be seen in her team's logbook in the Smithsonian museum. Before performing any computations operators of the Mark 1 would routinely *debug* the machine.



9/9
0800 Antan started
1000 " stopped - antan ✓ [illegible]
[illegible]
Relays [illegible] failed special speed test
in relay [illegible]
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
[illegible] Antangent started.
1700 closed down.

The first computer bug

Hopper later, when working on the UNIVAC computer, invented the first high-level computing language called "*Flow-matic*" that subsequently evolved into

COBOL, a language still in use to this day. A high-level computer language looks a bit like English and is easier for people to understand, read and create than the binary that a computer actually understands. A program, called a *compiler,* is needed to translate the high-level language into machine-readable binary. So she also invented the first compiler. After the war Hopper remained in the Navy Reserve and retired as a Rear Admiral when she was 79.

Grace Hopper (January 1984)

Incidentally, Hopper is also credited with coining the phrase: "*It's easier to ask forgiveness than it is to get permission.*" I imagine this is very true in the Navy and it certainly is in university where I work.

Chasing an enigma

Back across the Atlantic the British were also conscripting mathematicians into the war effort. If you know anything at all about Alan Turing it is probably that during World War II he cracked the German Enigma code and saved hundreds of thousands of lives by bringing the war to a speedier end. Unfortunately like so many popular legends this is not strictly true.

The Enigma code was produced by German electromechanical rotor machines that could be used to encrypt and decrypt secret messages. First invented at the end of World War I they were used by commercial companies and then the German military. The basic Enigma machine was refined and made more complicated during WWII.

An Enigma Machine at Bletchley Park

The basic idea of a cipher is to take a piece of ordinary text and scramble it so only the intended recipient can descramble it. The simplest cipher is a *substitution* cipher. For example, take a word and substitute each letter by the letter two positions after it in the alphabet: "*BREAD*" becomes "*DTGCF*". These simple ciphers were used in the Middle Ages for secret messages, but are very easy to crack. To make a cipher more complicated you need to vary the method by which you make the substitution for each letter, but in a way that the recipient can still decipher. This is called *polyalphabetic substitution*.

There is a distinction between *codes* and *ciphers*, though the words are commonly used interchangeably. A *code* is technically where words have different prearranged meanings. I might say: "*Meet me in the park and bring an apple*." But if you knew in advance that *park* was code for *bank* and *apple* meant *gun* then you'd know to meet me at the bank with a gun. A *cipher* is where the original letters in words have been scrambled.

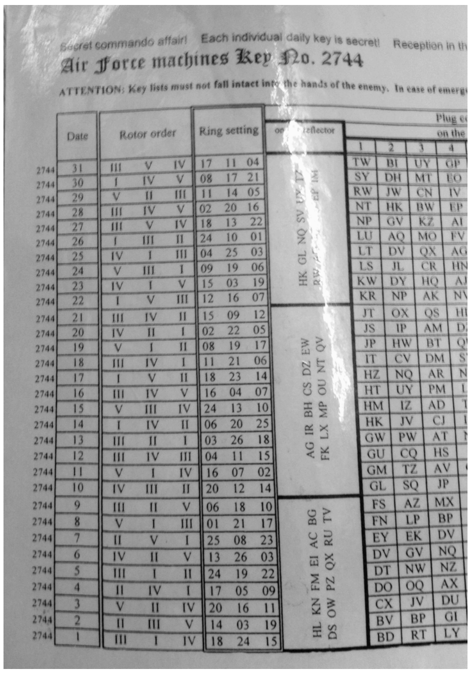
What the Enigma machine does is to automate polyalphabetic substitution to make a cipher with, at its most complex, approximately 10^{114} possible permutations. Far too many for anyone to crack, or so the Germans thought.

If you look at the photo of the Enigma machine you'll see at the front are a number of electrical wires with plugs and sockets. These wires can be plugged into different sockets to change the machine's basic configuration by swapping pairs of letters. Above the wires is a keyboard, and above the keyboard, in the center of the machine a panel of lights that are illuminated letters. Above the lights is a set of small rotors; these can be inserted in different orders and to different settings to further alter the machine's configuration.

An Enigma Machine showing the rotors

To send an encrypted message, the operator chooses which rotors to use for the day, sets the individual rotor's settings and inserts each into the machine in a particular order. He sets the Enigma's plug wirings and the rotor wheels' settings to a predefined initial combination known to him and to the receiver. Then he types the text of the message on the Enigma's keyboard. For each typed letter, a different letter lights up on the panel above the keyboard. The operator's assistant writes down each illuminated letter. When he has finished typing the original message, he will have a seemingly random sequence of letters - the Enigma encrypted message. A radio operator can then transmit the encrypted message by radio using standard Morse code.

The receiving radio operator must write down the received encrypted message, set his Enigma machine to the same pre-defined settings (both electrical and mechanical), and then type the message on the machine's keyboard. As he types his assistant can read the original deciphered text message from the letters illuminated in the panel above the keyboard. The daily settings for the Enigma machine were contained in monthly settings books held by radio operators who were under strict instructions to never let these books fall into enemy hands.



Secret commando affair! Each individual daily key is secret! Reception in th

Air Force machines Key No. 2744

ATTENTION: Key lists must not fall intact into the hands of the enemy. In case of emerg

	Date	Rotor order			Ring setting			on [illegible] reflector	Plug c on the			
									1	2	3	4
2744	31	III	V	IV	17	11	04	HK GL NQ SV UY TZ RW [illegible] EP IM	TW	BI	UY	GP
2744	30	I	IV	V	08	17	21		SY	DH	MT	EO
2744	29	V	II	III	11	14	05		RW	JW	CN	IV
2744	28	III	IV	V	02	20	16		NT	HK	BW	EP
2744	27	III	V	IV	18	13	22		NP	GV	KZ	AI
2744	26	I	III	II	24	10	01		LU	AQ	MO	FV
2744	25	IV	I	III	04	25	03		LT	DV	QX	AG
2744	24	V	III	I	09	19	06		LS	JL	CR	HN
2744	23	IV	I	V	15	03	19		KW	DY	HQ	AJ
2744	22	I	V	III	12	16	07		KR	NP	AK	NV
2744	21	III	IV	II	15	09	12	AG IR BH CS DZ EW FK LX MP OU NT QV	JT	OX	QS	HI
2744	20	IV	II	I	02	22	05		JS	IP	AM	D
2744	19	V	I	II	08	19	17		JP	HW	BT	Q
2744	18	III	IV	I	11	21	06		IT	CV	DM	S
2744	17	I	V	II	18	23	14		HZ	NQ	AR	N
2744	16	III	IV	V	16	04	07		HT	UY	PM	L
2744	15	V	III	IV	24	13	10		HM	IZ	AD	T
2744	14	I	IV	II	06	20	25		HK	JV	CJ	I
2744	13	III	II	I	03	26	18		GW	PW	AT	N
2744	12	III	IV	III	04	11	15		GU	CQ	HS	
2744	11	V	I	IV	16	07	02		GM	TZ	AV	
2744	10	IV	III	II	20	12	14		GL	SQ	JP	
2744	9	III	II	V	06	18	10	HL KN FM EI AC BG DS OW PZ QX RU TV	FS	AZ	MX	
2744	8	V	I	III	01	21	17		FN	LP	BP	
2744	7	II	V	I	25	08	23		EY	EK	DV	
2744	6	IV	II	V	13	26	03		DV	GV	NQ	
2744	5	III	I	II	24	19	22		DT	NW	NZ	
2744	4	II	IV	I	17	05	09		DO	OQ	AX	
2744	3	V	II	IV	20	16	11		CX	JV	DU	
2744	2	II	III	V	14	03	19		BV	BP	GI	
2744	1	III	I	IV	18	24	15		BD	RT	LY	

The daily settings from an Enigma settings book

By the end of 1932, the Polish *Cipher Bureau* had developed a laborious method for cracking the Enigma's ciphers. The Poles had good reason to fear the Nazis and before the outbreak of WWII they shared their decryption techniques and equipment with French and British military intelligence. The British understood the Polish techniques for breaking the Enigma cipher and had a working copy of an electromechanical machine designed by the Poles to semi-automate the process called the *Bomba*. However, the Germans had made improvements to the Enigma machine that made it more complicated than the pre-war version. In particular the Naval Enigma machine used additional rotors, which made it much more complicated to crack. The German Navy used Enigma to send messages to their U-Boat wolf packs in the Atlantic that were decimating the British convoys. It was vital that Enigma was cracked if Britain was not to be starved into submission.

At the outbreak of war Turing went to work at the top-secret *Government Code and Cipher School* housed at Bletchley Park,[4] which I visited recently. If you are ever in the vicinity I can highly recommend the trip, it was fascinating. Many of the photographs in this chapter I took there. During the war this establishment was filled with the best and brightest mathematicians and cryptographers as well as people who excelled at crosswords and all sorts of puzzles. Many people think that Bletchley Park went by the code name "*Station X.*" But it was explained to me there that if you think about it, calling a top-secret base "*Station X*" is a bit daft, because it rather calls attention to it. In fact Bletchley Park was a radio receiving station and it was the 10th in a series of similar stations, hence it's designation *Station X* using the Roman numeral.

[4] http://www.bletchleypark.org/

Whilst I lived in England I had the privilege of meeting Donald Michie, a colleague of Turing's, who worked at Bletchley Park with him. Donald told me that Turing liked to play chess, but surprisingly for a genius, though a very enthusiastic player, he wasn't very good at the game. He also told me that Turing was such a keen long distance runner he sometimes used to run forty miles into London, and back, for meetings. Around this time it is recorded that Turing ran a marathon in 2 hours, 46 minutes, and 3 seconds. His time was only 11 minutes slower than the winner of the marathon in the 1948 Olympic Games. Clearly Turing was a competitive long-distance runner.

In collaboration with the mathematician Gordon Welchman Turing designed a new version of the Polish Bomba, called the "*Bombe*," to handle the new more complicated version of Enigma.

The Bombe didn't decipher the cipher, but it could search the billions of possible permutations of the Enigma settings and rule out those that were incorrect. This allowed human code-breakers to concentrate on the more probable settings. Enigma had one particular weakness, it was impossible for an Enigma machine to code a letter as itself - an "*A*" would never be encrypted as an "*A*." This sounds obvious, even sensible but it gave code-breakers a vital way into the code. This in combination with other factors such as operator mistakes, occasional captured hardware and settings books allowed the cryptographers to be so successful.

A common mistake that was exacerbated by military protocol was to send messages containing the same phrases. This was actually very common as military jargon is standardized and repetitive. One phrase that was often sent was "*no special events*" or "*keine besonderen Ereignisse.*" German units would send this message daily if all was quiet in their sector and there was nothing to report and as any soldier will tell you war can often be quite dull, so this message was sent a lot. One German unit every night for a while sent the same message, "*Beacons lit as ordered*" or "*Feuer brannten wie befohlen*."

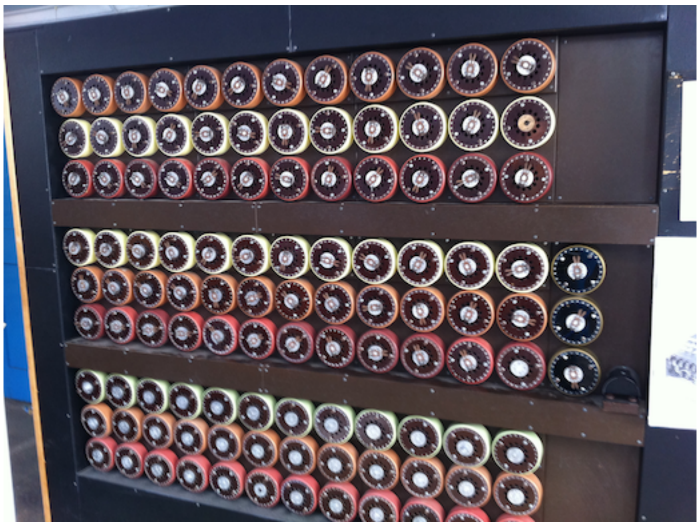
Knowing that a message could contain a known phrase gave the code-breakers an advantage. Another common error was to transmit standardized messages at standard times, such as weather reports to German submarines. If the Allied meteorologists predicted the weather the same as their German counterparts then they might already know the basic content of the message. So a message might read: "*Forecast to 12:00 hours wind south westerly ten knots sea state slight visibility good. HH.*" These standard pieces of expected text were called *cribs*.

If they suspected a crib might be in the transmission they could more easily decipher it, and thus deduce the Enigma machine's settings for the day. Here's how their process worked. They could take the intercepted encrypted German radio message, suspected of containing the crib and run it through various settings of their Enigma machine and see if decrypted plain German text emerged. If German didn't come out they could change the Enigma settings and repeat the process. Of course there were hundreds of millions of possible Enigma settings and somebody would need to check the output for German each time.

Here's were the flaw in Enigma's process came to help them. They didn't need to check if the possibly decrypted text was German because if any of the letters in the original intercepted message and the possible decrypted message matched (i.e., were identical) then those Enigma settings could not be correct, since Enigma never encodes a letter as itself. Checking to see if letters in the encrypted message and possible decrypted message matched was easier than checking for German language.

This would of course still be a very laborious process if done by hand and also it would be prone to error, but Turing knew the process could be mechanized. A machine could be built that could automate the entire process. In fact, before the war the Poles had built just such a machine, which they called a Bombe. It could simulate several Enigma machine settings simultaneously and evaluate nearly 9,000 different settings an hour. However, since the start of the war the Germans had made the Enigma machine more complicated by the use of the plug board and additional rotors.

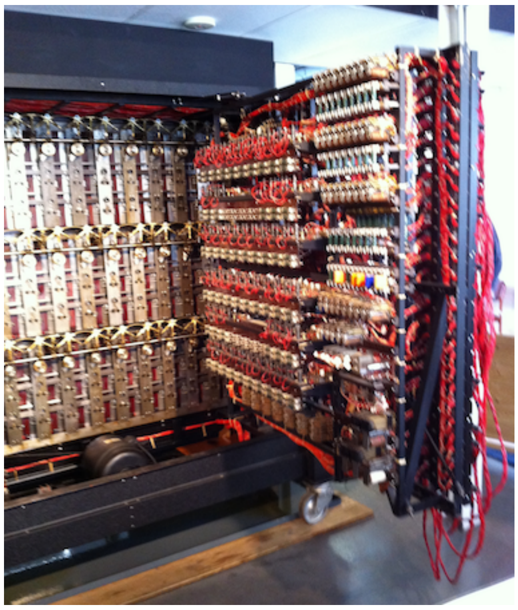
Turing's *Bombe* could simulate 30 Enigma machine settings at a time and would rapidly crank through setting after setting until it found a setting where no letters matched in the crib and the encrypted message. An operator (usually a female naval ensign called a WREN) would read the Enigma settings off the Bombe, set up her Enigma simulator to the same setting, type in the encrypted message and if German came out they had the day's Enigma settings. If not the Bombe would be restarted.



The front of the Bombe at Bletchley Park

The photo of the front of the Bombe shows 12 sets of three rotors in three panels. Each vertical set of three rotors represents an individual Enigma machine. The middle panel has an extra set of three rotors on the right. When a Bombe run stops the operator can read the Enigma settings off these rotors.

The front looks complex enough, but step around the back and there is over ten miles of wiring and numerous electromechanical relays. If you visit Bletchley Park you can see a working replica of the Bombe and be taken step by step through the entire process, from the encoding of a message using a real Enigma machine to obtaining the Enigma settings using the Bombe and decoding the message.



The back of the Bombe at Bletchley Park

Bletchley Park turned military intelligence into an industrial process. 12,000 people worked there in three eight-hour shifts. At any time there might be 4,000 people on duty. 200 Bombes were built by the *British Tabulating Machine Company,* which manufactured and sold Hollerith Tabulators under license in the UK. Teams of mechanics serviced the Bombes, for they were unreliable, and further teams of mostly female code-breakers ran them. Once one team had the day's Enigma settings further teams would set about decoding all the day's intercepted German radio traffic.

By the end of the war the Allies were reading all of the German Enigma transmissions, but military intelligence had to be careful how it acted on the information it gained from the German's encrypted messages lest the Germans realized that their communications were no longer secure. This must have caused a terrible moral dilemma for those in authority, as no doubt many lives were sacrificed to ensure that the Allies could continue to read the German's messages. It has even been alleged that Winston Churchill allowed the city of Coventry to be destroyed by the Luftwaffe rather than risk letting the Germans suspect the Allies had cracked Enigma. However, some historians and some who worked at Bletchley refute this allegation. Nonetheless it does serve to illustrate the terrible moral dilemma Churchill would have faced.

A top-secret intelligence group called *Ultra* was established to act upon the intercepted Enigma messages. To cover their success they persuaded German intelligence that their secrets were being leaked by double agents within the German establishment. The Germans spent a lot of time and effort during the war hunting down spies that didn't exist!

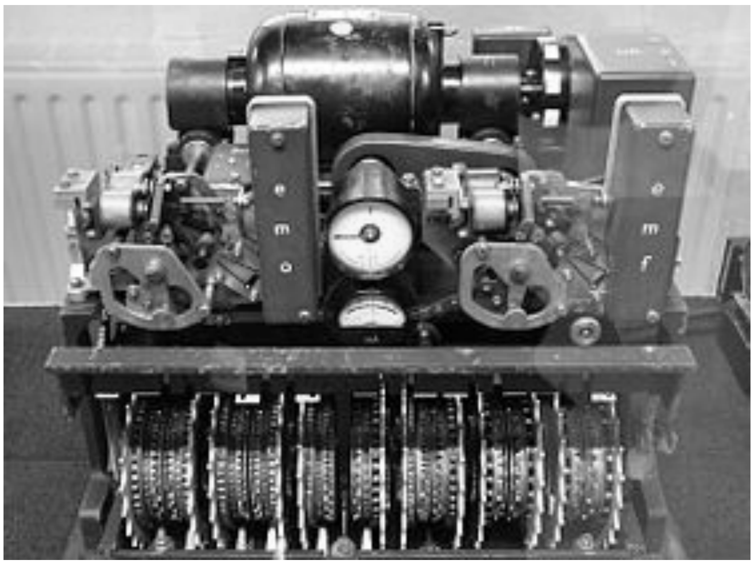
Turing's Bombe was not a universal machine, it was a very specific machine completely tailored to the Enigma problem. It was treating each possible Enigma setting as a decision problem and was using a simple algorithm to solve it. After the war all the Bombes were destroyed, because there was no need for them. Bletchley now has two replicas, one created for the movie *Enigma*, which does not work and a fully working replica built by enthusiasts.

Hitler's secret writer

In 1941 the British starting intercepting radio signals that were quite different to the Morse code used to send Enigma messages. They called it "*new music*;" and it sounded like the rapid squeal that a fax machine makes, this was a radio signal generated from a telex machine. It was also encrypted, but by a machine the Allies had never seen. This new cipher machine was used on teleprinters by German High Command to communicate with Army Group Commanders. It was initially only used on telephone lines and was considered completely secure. Later as the German forces became stretched out in the field it was also used on wireless transmissions. At Bletchley Park it was codenamed "*Tunny*."

The Germans called the machine their "*Geheimnis Schriftsteller*" or "*secret writer*" and unlike Enigma it was state-of-the-art. Using Enigma was a slow and labor intensive process; typically one person keyed in the original message, another wrote down the cipher text and a third transmitted it via Morse code. The whole process was reversed at the receiving end – only short messages could be sent using Enigma. The new secret writer, or *Lorenz machine* as it was actually called automated the whole process; a person typed in the message, the Lorenz machine encrypted it and transmitted it via telex and at the receiving end a Lorenz machine automatically decrypted it and printed out the original message. Long messages could be sent with complete security, which was why German High Command used it.

The code-breakers' problem was that, unlike Enigma, they had no way into the code since they'd never seen a Lorenz machine. Fortunately for the Allies on August 30 1941 two German Lorenz operators made a terrible mistake. A 4,500 character long message was transmitted from Athens to Vienna but the receiver in Vienna said the message wasn't properly transmitted. The Athens' operator retransmitted the message using exactly the same Lorenz settings. To make it worse instead of keying exactly the same message the Athens operator used abbreviations in the retransmission so the second message was shorter - only 4,000 characters long. The Allies now had two long similar messages sent using the same Lorenz settings. It was the stroke of luck the code-breakers needed. From this mistake a young mathematician, called Bill Tutte, was able after days of work to find a 41 character repeat in the cipher. Over the next few months Tutte, working with others at Bletchley Park, was able to reverse engineer the Lorenz machine and infer its complete logical structure. Tutte's achievement is still considered a remarkable intellectual feat.



A Lorenz Machine at Bletchley Park

After the war the Allies captured some *Lorenz* cipher machines, which were like big Enigma machines with 12 rotors – they were more than twice as complicated as Enigma and they worked exactly as Tutte had inferred. From the logical structure of the Lorenz machine Turing devised a laborious technique, named "*Turingery,*" to obtain the Lorenz settings from a cipher text. Max Newman, who ran a group at Bletchley called the *Newmanry*, designed an electromechanical machine nicknamed "*Heath Robinson,*" after the cartoonist who drew fabulously intricate contraptions, to automate this process. Unfortunately, the Heath Robinson was slow and unreliable and took up to 8 weeks to find the correct Lorenz settings from an intercepted cipher.

The Tunny project required more computing power than Heath Robinson provided; relays just weren't fast enough to sift through all the permutations of the Lorenz machine. This resulted in Bletchley commissioning the *Post Office Research Station* to develop the world's first programmable, digital, electronic computer. It's another common falsehood that Alan Turing designed and even built this machine. This is not true, though his ideas certainly influenced its design.

Colossus

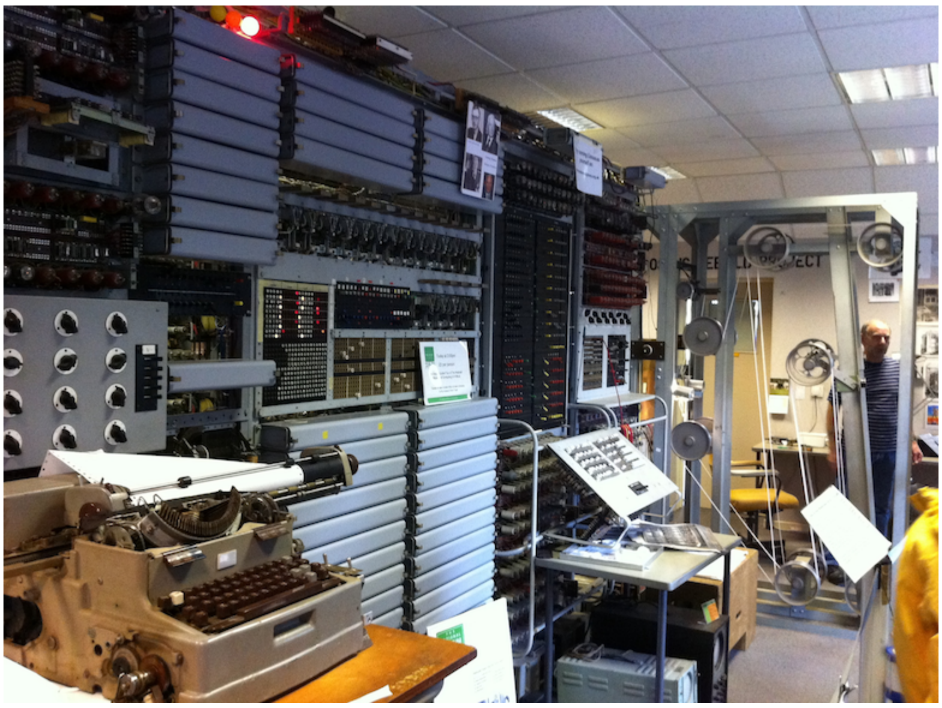
Let's kill another falsehood right away. Sorry guys, the Americans did not build the first electronic computer. ENIAC (*Electronic Numerical Integrator And Computer*) was first used in February 1946 at the University of Pennsylvania. Colossus Mark 1, was working in December 1943. Why then do so many histories state the Americans built the first computer? The reason is simple; Colossus was classified top-secret during the war and remained so until the 1970s! Everyone working with Colossus was bound by the *Official Secrets Act* and would be tried for treason if they spoke or wrote about it. ENIAC was built after the war and was publicly announced and thus entered all the history books as the first computer. Regrettably, because of the secrecy surrounding Colossus its creators did not receive the accolades, awards and fame that they deserved during their lives.

The publicity surrounding ENIAC became so entrenched that for the rest of the twentieth century countless history books, magazines, newspapers, documentaries and even websites continued to spread this falsehood. The Americans did **not** build the first electronic computer - the British did during WWII.

Colossus was the link between Turing's pre-war work on the universal Turing Machine and his post-war work on digital computing. ENIAC was developed in complete ignorance of Colossus, because it was a closely guarded secret, and the British considered ENIAC to be just a big "*number cruncher*" with less sophistication than Colossus.

An RCA Triode valve or vacuum tube

An electrical engineer called Tommy Flowers who worked for the Post Office designing telephone exchanges saw Newman's Heath Robinson machine at Bletchley and he knew he could build a faster version that used electronic valves instead of relays. Colossus Mark 1 had 1,500 valves, sometimes called vacuum tubes, whilst Mark 2 had 2,400 and was four times faster than the Mark 1. Electronic valves operate much quicker than the elctromechanical relays in machines, like the Harvard Mark 1 and the Bombe. Relays have to flip a small mechanical switch, whereas valves just move a beam of electrons. The use of state-of-the-art valves was crucial in developing fast digital computers.



The Colossus Mark 2 Computer at Bletchley Park

Colossus optically read a paper tape, which you can see to the right of the photo running around some wheels, and then applied a programmable logical function to every character, counting how often this function returned "*true*". Although machines with many valves were known to have high failure rates, Tommy Flowers knew that valve failures occurred when switching on a machine, in much the same way that light bulbs sometimes pop when you turn them on. Colossus, once turned on, was never powered down unless it malfunctioned. By the end of the war 10 Colossus computers were installed at Bletchley Park. Each computer generated so much heat that the female operators used to bring their laundry into its room to dry and an American visitor, one freezing winter's day, was heard to say that it was, "*the only goddam place in this country that's warm!*"

Colossus was the first electronic digital machine that could be programmed, albeit in very limited ways by modern standards. It was not, though, a Universal Turing Machine, and it was not then realized that what is now known as *Turing completeness* was important. All of the other pioneering computing machines were also not Turing complete. The notion of a computer as a general-purpose universal machine, that is, as more than a calculator devoted to solving hard, but specific problems, would not be seen as important for many years.

By the end of the war the Allies were reading all Enigma transmissions and most of German High Command's Lorenz traffic. Whilst at Bletchley I was told that if in the last days of the war if a German officer had wanted to know the location and status of any particular German unit he'd get the answer quicker by telephoning Bletchley than by asking his own High Command. Historians all seem to agree that the contribution of Turing and all the other code-breakers at Bletchley Park shortened the war by perhaps as much as two years and saved millions of lives. Some historians go even further and suggest that without Turing the Allies would not have won the war at all!

Calculating space

The Germans in addition to constantly improving their cryptography machine were also developing computers. It seems though that the Nazis didn't fully understand the importance of computing to the war effort. The first German computer was built by a brilliant lone amateur. Konrad Zuse was a civil engineer who also designed poster advertisements. He worked for Ford in Germany and then in 1935 for a plane factory. This work required him to perform countless complex computations by hand. Like Babbage before he started to think about mechanizing the computations. Working alone in his parent's home, completely isolated from mathematicians and university research, he completed the *Z1* in 1938. The Z1 was mechanical, had 30,000 metal components and used 35mm camera film as punch tape for program instructions. The Z1 never worked because Zuse couldn't engineer the parts to a sufficient precision. The parallels with Babbage continue when in 1989 *Siemens*, the German technology company, sponsored an expensive rebuild of the Z1 even though its original plans had been destroyed during the war.

Zuse was enlisted into the army in 1938 but was eventually given some resources to build the Z2 and Z3, in which he now used relays. The German aviation industry in particular was keen to support his work as they relied on numerous complex calculations to design their increasingly sophisticated planes. All his prototypes were destroyed by Allied air raids and after Germany's defeat there were no resources to continue development.

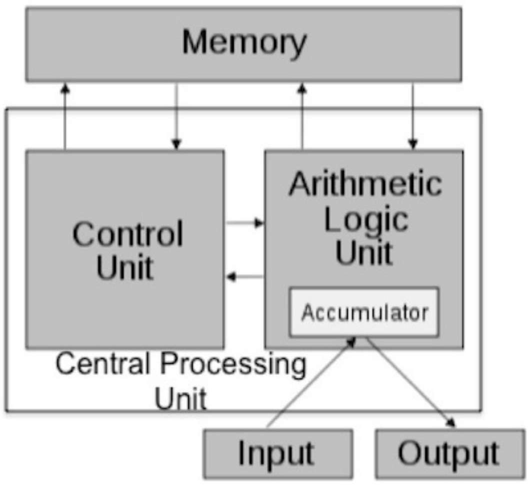
The replica Z1

When Germany recovered economically Zuse was eventually able to start his own computer company, which was bought by Siemens in 1967. Zuse built over 250 computers but is more famous now for a theoretical idea he published in a book called the *Rechnender Raum*. Translated as *Calculating Space,* Zuse proposed that the universe was actually a construct running on a network of computers. This idea is still influencing thinkers today in philosophy, science and entertainment. Recently, the 2010 movie *Tron: Legacy*, about a digital world inside a computer, pays homage to Konrad Zuse - Michael Sheen plays a nightclub owner called *Zuse*. The real Konrad Zuse died in December 1995.

The Imitation Game

Turing, because of his background as a mathematician, is often considered to be a theoretician and to have had little direct influence on the actual design of computers. This could not be further from the truth. Turing was a skilled electrical engineer and after the war he got a job at the *National Physical Laboratory* where he worked on the design of the ACE (*Automatic Computing Engine*), which was to be one of the first stored-program computers. Turing developed the complete circuit design for ACE and worked hard to make the hardware as simple as possible to improve reliability. He also developed the first software for the ACE and was a keen programmer.

ACE, like all modern computers, stored its program and the data upon which the program operated in the same memory; this is called a *Von Neumann architecture*. Machines like the Harvard Mark 1 had their programs hardwired into them so no program memory was needed. Colossus was programmable, but loading a new program was a laborious task by today's standards. As Turing commented in a report on ACE, "*There will positively be no internal alteration [of the computer] to be made even if we wish suddenly to switch from calculating the energy levels of the neon atom to the enumeration of groups of order 720. It may appear somewhat puzzling that this can be done. How can one expect a machine to do all this multitudinous variety of things? The answer is that we should consider the machine to be doing something quite simple, namely carrying out orders given to it in a standard form which it is able to understand.*"



The Von Neumann architecture

Turing eventually became tired of the secrecy of working at government labs and went back to Cambridge for a sabbatical year and then moved to Manchester University where he became Deputy Director of its new *Computing Laboratory*. Here he developed software for the new Manchester Mark 1 computer, an early stored-program computer. Again his work was of a very practical nature including writing: system software to make the machine usable, interface software for a paper tape reader, a random number generator, and writing what is probably the world's first programming manual - *Programmers' Handbook for the Manchester Electronic Computer Mark II*[5]*.* In this he lays out his "*Programming Principles: i) Make a Plan. ii) Break the problem down. iii) Do the programming of new subroutines. iv) Programme the main routine.*" This will be still instantly recognizable to any programmers reading this book.

[5] A facsimile of the manual can be seen on the The Turing Archive for the History of Computing http://www.alanturing.net/turing_archive/archive/m/m01/m01.php

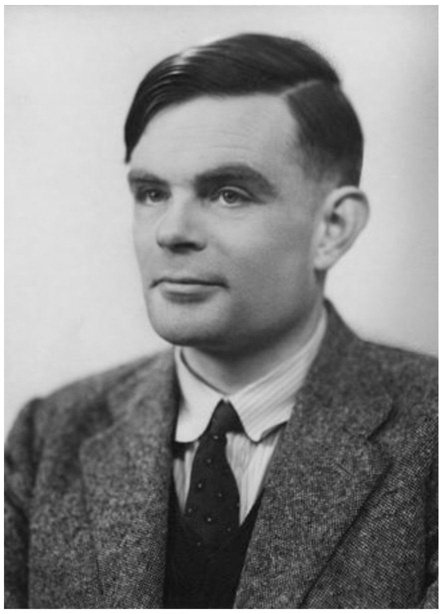
In 1950 he published another theoretical paper that was to have profound and long reaching impact. This would firmly cement Turing as the greatest computer scientist who has ever lived.

"*Computing machinery and intelligence*" was published in *Mind* in October 1950, and as it's title suggests Turing was dealing with the subject of machine intelligence, or artificial intelligence as we now call it. In the paper Turing asks what it would take for a machine to be considered "*intelligent*" and he proposed a test for machine intelligence he called *the imitation game*. This test is now known as the *Turing Test*.

In a Turing Test a person sitting at a computer communicates via the keyboard (like instant messaging) with two entities. One is another person and the other is a computer. The person is allowed to ask the two entities any questions they want. If after a set period of time the person cannot tell the machine from the human, then the machine has passed the Turing Test.

If you think about this test it is really demonstrating that the machine *appears* to be intelligent. It is not claiming that the machine actually *is* intelligent. You could say that the computer is imitating intelligence. Indeed this is exactly what Turing thought and he wrote: "*Are there imaginable digital computers which would do well in the imitation game?"* Turing believed that this question could be answered positively although nobody could build a computer to even attempt the test in his lifetime.

Not content with inventing the disciplines of computer science and artificial intelligence Turing also turned his remarkable mind to a biological problem – *morphogenesis*. In a paper called *The Chemical Basis of Morphogenis* he described the *Turing hypothesis* of pattern formation, which explains how spots may occur on a leopard's skin for instance. Turing is now considered a founder of bioinformatics.



Alan Turing, 1951

To die perchance to sleep…

On the June 8 1954 Turing was found dead by his cleaner. He had taken a bite from an apple poisoned with cyanide and the subsequent inquest found he had committed suicide.

During my brief biography of Alan Turing I have omitted a significant detail about him, which though not relevant to his professional career, his remarkable discoveries and contributions, is of great significance to his private life. Turing was homosexual and had been since his school days. We now don't care about people's sexual orientation, but back in the 1950s homosexuality was still illegal in Britain and could result in imprisonment.

In 1952 Turing reported a burglary at his house to the local police. He was able to name a prime suspect, Arnold Murray, a young man he had met a few days before outside a cinema in Manchester. During the police investigation Turing naively told the police that he and Murray had a sexual relationship. Turing and Murray were charged, tried and found guilty of gross indecency.

Turing was given a choice of sentence - imprisonment or chemical castration by estrogen injections. He chose the latter. He also had his security clearance revoked, and though his passport wasn't confiscated, he was denied entry to the US because he now had a criminal conviction.

Nobody really knows why Turing seems to have volunteered to the police details of his relationship with Murray. They were probably suspicious of the relationship between an eminent scientist and a young working class man anyway, but crucially they had no proof. Did Turing believe his position would protect him? Did he want to be caught? We just don't know.

We also don't know why he decided to kill himself, though it is believed that the unnatural hormonal changes to his body may have led to a deep state of depression; he was seeing a psychiatrist before his death. We do know why he chose to eat a poisoned apple. Disney's animated masterpiece *Snow White & the Seven Dwarfs* was Turing's favorite movie and it is believed that he was mimicking the poisoned apple from the story. There is a difference though, Snow White falls into a deep sleep, she doesn't die after eating from the witch's apple. A kiss from a prince awakens her. Is Turing still waiting for his prince?

There is a popular myth that the Apple logo, an apple with a bite out of it, honors Alan Turing and his poisoned apple. There might also even be a play on words between *bite* and *byte,* a computer term. Unfortunately, though I love this myth, Apple historians say there is no truth to it. Stephen Fry the British TV presenter, comedian and friend of Steve Jobs, recalls Jobs saying when asked about the origins of the logo, "*It isn't true, but God we wish it were!*" Whenever I see the Apple logo I remember Turing though, for without his discoveries Apple's products would not exist.

Apple's logo

Turing's legacy

Unfortunately, because of the secrecy surrounding Turing's work during the war, and because of the double taboos surrounding his death - homosexuality and suicide, Turing remained virtually unknown for years. Since 1966 the *Association* *of Computing Machinery* has named its highest annual award the *Turing Award*; it's considered the computing world's equivalent to a *Nobel Prize*. However, outside of the computer science community almost nobody had heard of Turing. It wasn't until 1983 when the mathematician and author Andrew Hodges published a book called *Alan Turing: The Enigma* that a wider public learnt of Turing. The book was quickly followed by a very successful West End and Broadway play called, *Breaking the Code*. The BBC later broadcast a version of the play on television.

The true history of Turing's great contributions both to Britain's war effort and to the development of the computer has since led to a succession of different honors being conferred on him: a blue plaque was unveiled at his birthplace in London, a main road in Manchester was renamed the *Alan Turing Way* and a bridge the *Alan Turing Bridge*, a statue of him was unveiled at Bletchley Park and another in Manchester with a plaque that reads "*Alan Mathison Turing 1912-1954 Father of Computer Science, Mathematician, Logician, Wartime Codebreaker, Victim of Prejudice.*"

Alan Turing's statue at Bletchley Park

In 1999 *Time Magazine* named him as one of the "*100 Most Important People of the 20th Century*" stating: "*The fact remains that everyone who taps at a keyboard, opening a spreadsheet or a word-processing program, is working on an incarnation of a Turing machine.*"

In 2009 John Graham-Cumming, a computer programmer and author, started a petition asking the British government to apologize posthumously for persecuting Turing for being homosexual. The petition quickly received over 30,000 signatures and the Prime Minister issued a rare official apology.

Turing's name now sits comfortably amongst the great thinkers of our civilization: Socrates, Aristotle, Copernicus, Galileo, Newton, Darwin, Einstein -

none of them directly saved as many lives though or could run a competitive marathon.

The subsequent chapters will show how Turing's universal machine has changed our world.