Computer systems

College of Saint Benedict & Saint John's University

great insights of computer science[1]

Bacon, Leibniz, Boole, Turing, Shannon, & Morse

There are only **two nouns** that a computer has to deal with in order to represent “anything”: 0, 1.

- “anything”: there are some things computers cannot do — like determine if a program will ever finish.

[1] The great insights of computer science / CC BY-SA 3.0

great insights of computer science, cont'd

Turing

There are only **five verbs** that a computer has to perform in order to do "anything":

1. move left one location;
2. move right one location;
3. read symbol at current location;
4. print 0 at current location;
5. print 1 at current location.

great insights of computer science, cont'd

Boehm and Jacopini

There are only **three grammar rules** needed to combine these verbs (into more complex ones) that are needed in order for a computer to do "anything":

1. *sequence*: first do this, then do that;
2. *selection*: IF such-and-such is the case, THEN do this, ELSE do that;
3. *repetition*: WHILE such-and-such is the case DO this.

a simple language

1. two nouns
2. five verbs
3. three grammar rules

`<`	move left one location
`>`	move right one location
`0`	print `0` at current location
`1`	print `1` at current location
`[`	if current location is `0`, then go to instruction after matching `]`
`]`	go to matching `[` instruction

```
1>1>0>1>0<<<<[0>]1
```


- *sequence*: start at left-most instruction and progress a single instruction to the right
- *selection* and *repetition:* `[...]` provide both — repetition is just fancy selection

a simple language, cont'd

`<`	move left one location
`>`	move right one location
`0`	print `0` at current location
`1`	print `1` at current location
`[`	if current location is `0`, then go to instruction after matching `]`
`]`	go to matching `[` instruction
`^`	add one to the 4-bit number ending at the current location

`^` replaces the sequence `>0<<<<[0>]1`

```
1>1>0>1^
```

- let's add another instruction to increment a number by one
- we have introduced some *abstraction*

abstraction

abstraction

A mechanism and practice to reduce and factor out details so that one can focus on a few concepts at a time.

Abstraction allows program designers to separate categories and concepts related to computing problems from specific instances of implementation.[2]

[2] Abstraction / CC BY-SA 3.0

types of abstraction

data abstraction

The separation of a data type's logical properties from its concrete implementation.

In fact, a data type is a data abstraction.

```
boolean found := false
```

control abstraction

The separation of the behavior of a set of actions from its concrete implementation.

One of the main purposes of programming languages.

```
a := (2 + 3) / 4
```


- Data abstraction — take 8 bits, how do you know how to interpret the 8 bits? You will need to know if it is an: integer, a floating-point value, or something else...all three are abstractions of the bits
- Here is an example — How is an 32-bit number actually stored in our machine? $0x0000010F = 271$

- In the example code, `:=`, `+`, and `/` are all examples of control abstraction
- What vale does `a` hold?
  - You will need to now what data abstraction we are using for the numbers `2`, `3`, and `4`, so that you know their logical properties, like how addition and division of two of them works
- Other examples include decisions, iterations, functions

- Where do classes in Java fit?

abstraction levels

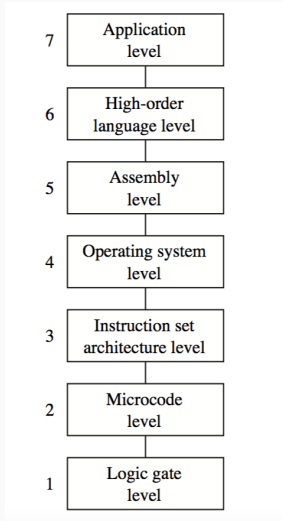
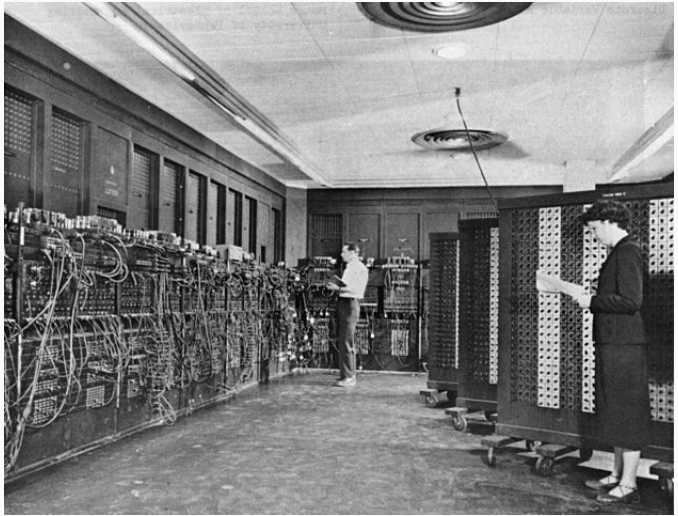


Figure P.1

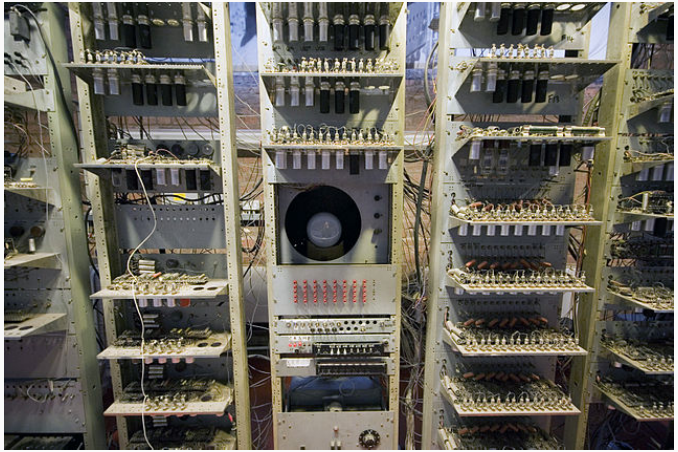
history of abstraction



U.S. Army Photo / Public Domain

- Circa 1946 — World War II
- LG1 & ISA3
- Six women: Kay McNulty, Marlyn Wescoff, Ruth Lichterman, Betty Jean Jennings, and Fran Bilas, programmed the first large-scale general purpose machine, ENIAC, to compute ballistics trajectories. Programming was done by reorganizing wiring of plugboards.

history of abstraction, cont'd



SSEM Manchester museum close up / CC BY 3.0

- Circa 1948
- LG1 & ISA3
- Stored program in memory. Programs were entered in binary form by stepping through each word of memory in turn, and using a set of 32 switches known as the input device to set the value of each bit of each word to either 0 or 1.

history of abstraction, cont'd

Commodore Grace M. Hopper, USN / Public Domain

- Circa 1951
- ASM5 & HOL6
- assembly and high-order languages were developed around the same time
- Assembly programs are machine-specific, 1-to-1 mapping between assembly instructions and machine instructions
- some early programming languages still in use today:
  - FORTRAN (1954)
  - LISP (1958)
  - COBOL (1959)
  - ALGOL (1958)

history of abstraction, cont'd

- Circa 1956
- OS4
- operating system is a piece of software that runs other pieces of software
- allow multi-tasking and multi-user environments
- gives software an interface to hardware — it manages hardware resources