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.

[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
```

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^
```

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
```

abstraction levels

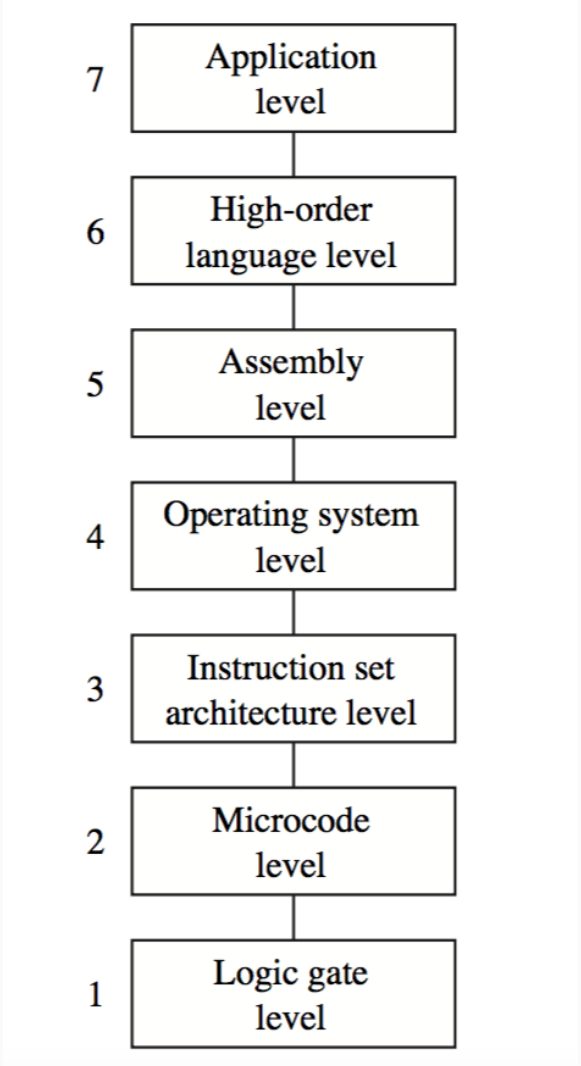
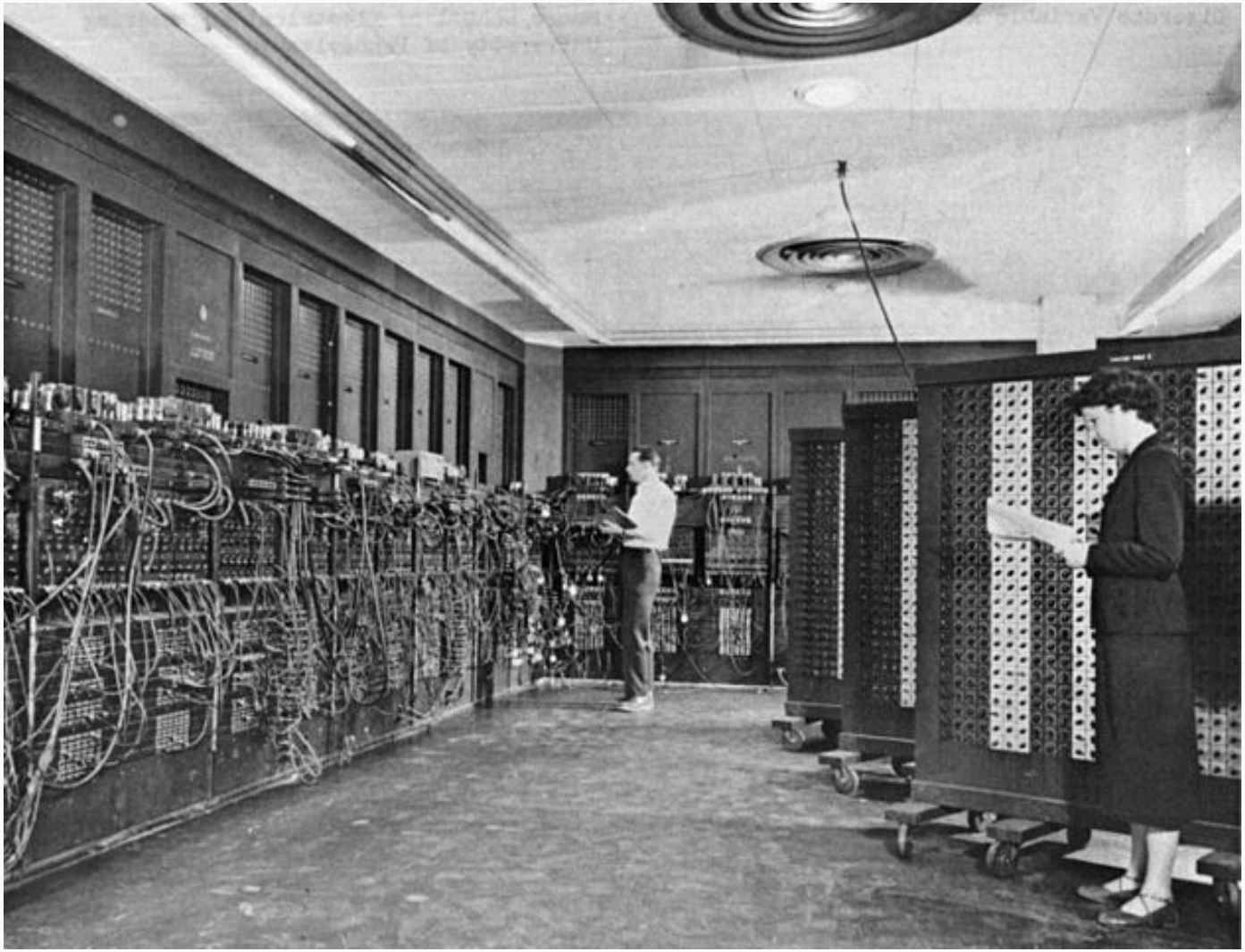


Figure P.1

history of abstraction



U.S. Army Photo / Public Domain

history of abstraction, cont'd

SSEM Manchester museum close up / CC BY 3.0

history of abstraction, cont'd

Commodore Grace M. Hopper, USN / Public Domain

history of abstraction, cont’d