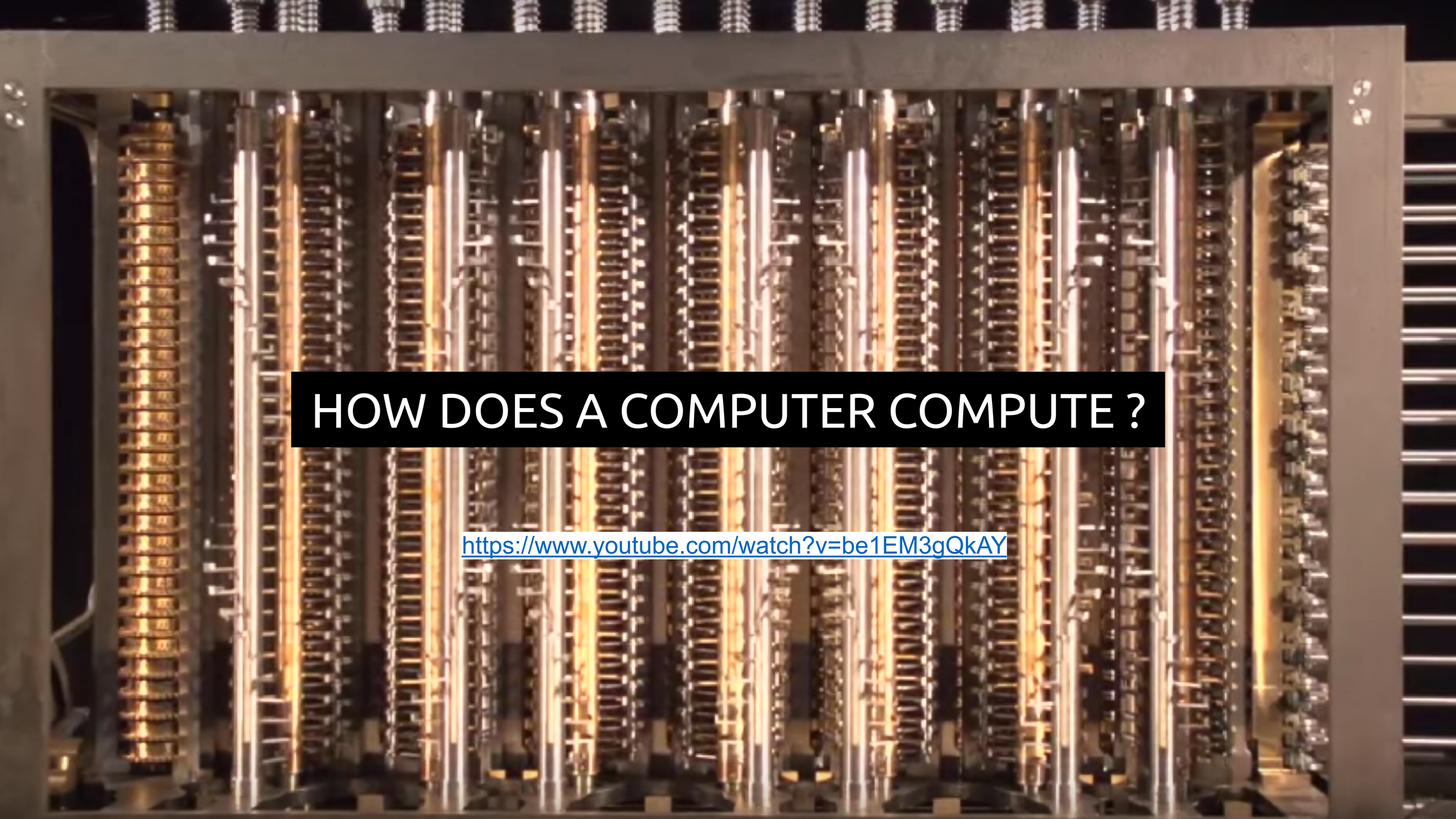
BUILDING CLOCKS

Basic programming / Over Time

Lior Ben-Gai
August 2019



HOW DOES A COMPUTER COMPUTE ?

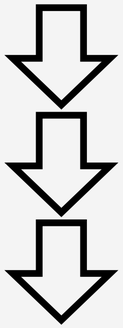
https://www.youtube.com/watch?v=be1EM3gQkAY

HOW DOES A COMPUTER COMPUTE ?

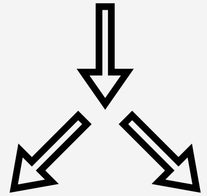
https://www.youtube.com/watch?v=SPaI5BJxs5M

THE BUILDING BLOCKS OF PROGRAMMING:

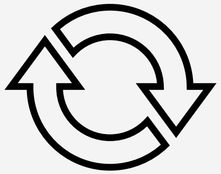
Memory



Decision



Action



Repetition

THE BUILDING BLOCKS OF PROGRAMMING:

Variables

```
var x = 0;
```

Memory

Conditions

```
if(x > 10){
  x = 0;
}
```

Decision

Functions

```
function mult(y){
  x = x * y;
}
```

Action

Loops

```
for(var i = 0; i < 100; i++){
  x = x + 1;
}
```

Repetition

1. VARIABLES

```
var x = 100;
var y = 0.234;
var isDrawing = true;
var firstName = "Lior";
var lastName = "Ben-Gai";
```

```
console.log(x); // 100
x = x * 10;
console.log(x); // 1000
x = x * y;
console.log(x); // 234
```

```
var numbers = [1, 2, 3, 4, 5];
var words = ["hello" , "World!"];

var msg = words[0] + " " + firstName + " " + numbers[3];
console.log(msg); // "hello Lior 4"
```

A program remembers what it is doing by declaring and updating variables

A variable is a slot in memory that has a name.

Variables can hold: numbers, letters, words, functions and more compound data structures.

2. CONDITIONS

```
if(x > width){
  x = 0;
}

if(step >= maxStep){
  stop();
}else{
  step++;
}
```

```
if(2 + 2 == 5 || 2 * 2 == 4){
  console.log("yes!");
}

if( (true && false) || true ){
  console.log("yes?");
}
```

```
(a > b) // is a larger than b?
(a < b) // is a smaller than b?

(a <= b) // smaller OR equal?
(a >= b) // larger OR equal?
(a == b) // is a equal to b?
(a != b) // is a NOT equal to b?

(a || b) // is a true OR b true?
(a && b) // is a true AND b true?
```

Programs make decisions by using conditional statements

A conditional is a block of code

it will be executed based on a true/false statement

Conditionals may be nested - creating complicated decisions (AKA: algorithms).

3. FUNCTIONS

```
// no parameters, no return
function myFunc(){
  point(10, 20);
}

myFunc();
```

```
// two parameters, no return
function myFunc(x, y){
  point(x, y);
}

myFunc(20, 10);
```

```
// no parameters, return
function myFunc(){
  return 10;
}

point( 20, myFunc() );
```

```
// parameters and return
function myFunc(x, y){
  point(x, y);
  return x / y;
}

var ratio = myFunc(20, 30);
```

Functions are blocks of operations that have meaning.

A function may expect a parameter or return a value, or neither, or both.

functions often call other functions and sometimes themselves.

4. LOOPS

```
for(var i = 0; i < 100; i++){
  line(i*4, 0, i*4, height);
}
```

for(“define something” ; “as long as that thing is ___ “ ; “change the thing”)
{ “keep doing this” }

```
var x = 0;
while(x < 100){
  line(x*4, 0, x*4, height);
  x++;
}
```

while(“some condition is true”)
{ “keep doing this” }

Loops are a way to do multiple things at the same time.

Loops help us to a simple operation many times, instead of a complicated thing once.

CLASS/HOME WORK

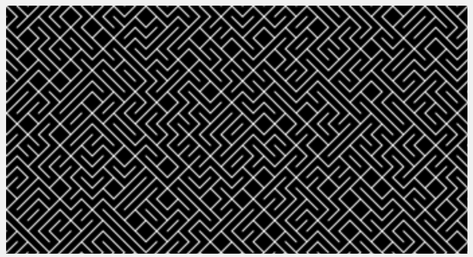
"A computer is just a very complicated and very fast CLOCK."

Please design and program a clock app.

Your clock should not tell time in the traditional way - you can't use digits or dials.

Instead, try to re-imagine how time could be communicated in a new way.

Clock


Building Blocks