Curtin University

Computational Thinking

ISYS2001, School of Marketing and Management

I acknowledge the traditional custodians of the land on which I work and live, and recognise their continuing connection to land, water and community. I pay respect to elders past, present and emerging.

Today

- Welcome
- Computers
- Programming
- Data

Welcome!

- Assume no programming experiences
- Series of short lectures
- Weekly lab sessions
- Discuss the unit outline in first lab
- If confused ask questions
- Practice, practice, practice!

90%

of students have never taken a programming course

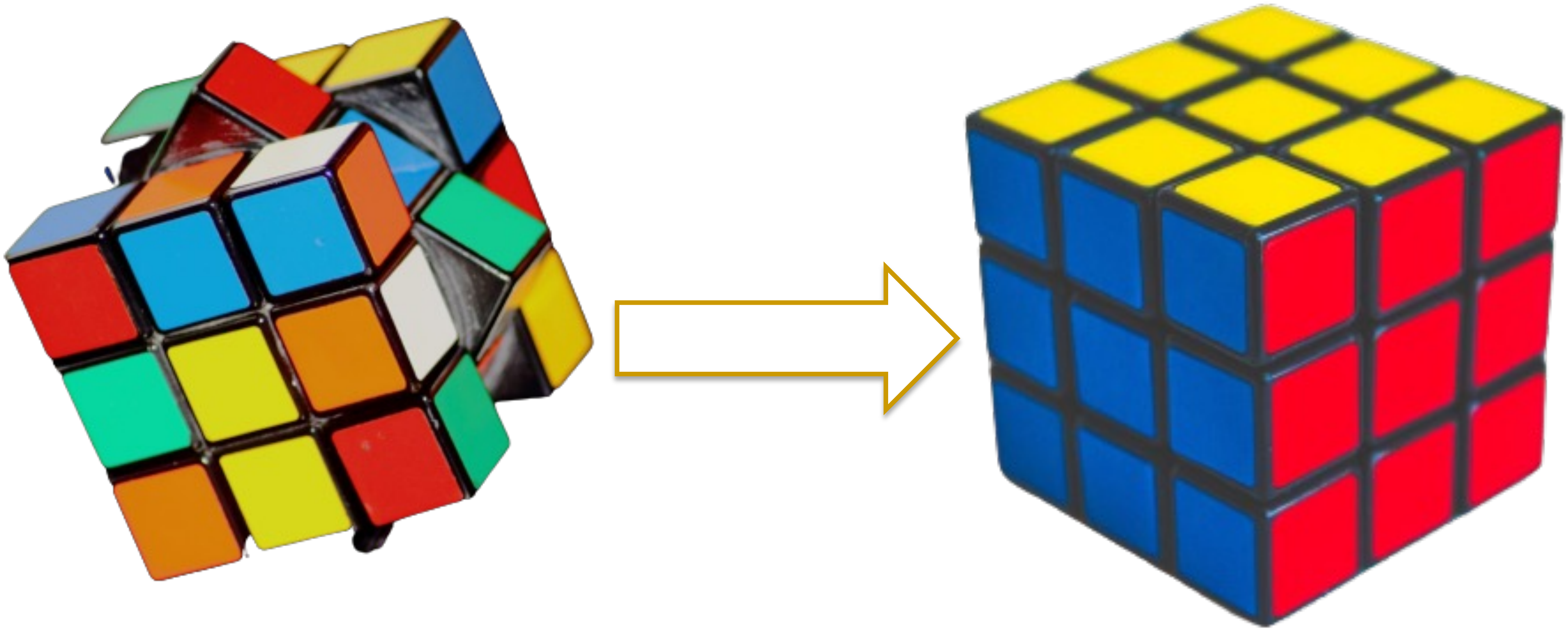
Laptop or Desktop

Modern Browser

Google/Github

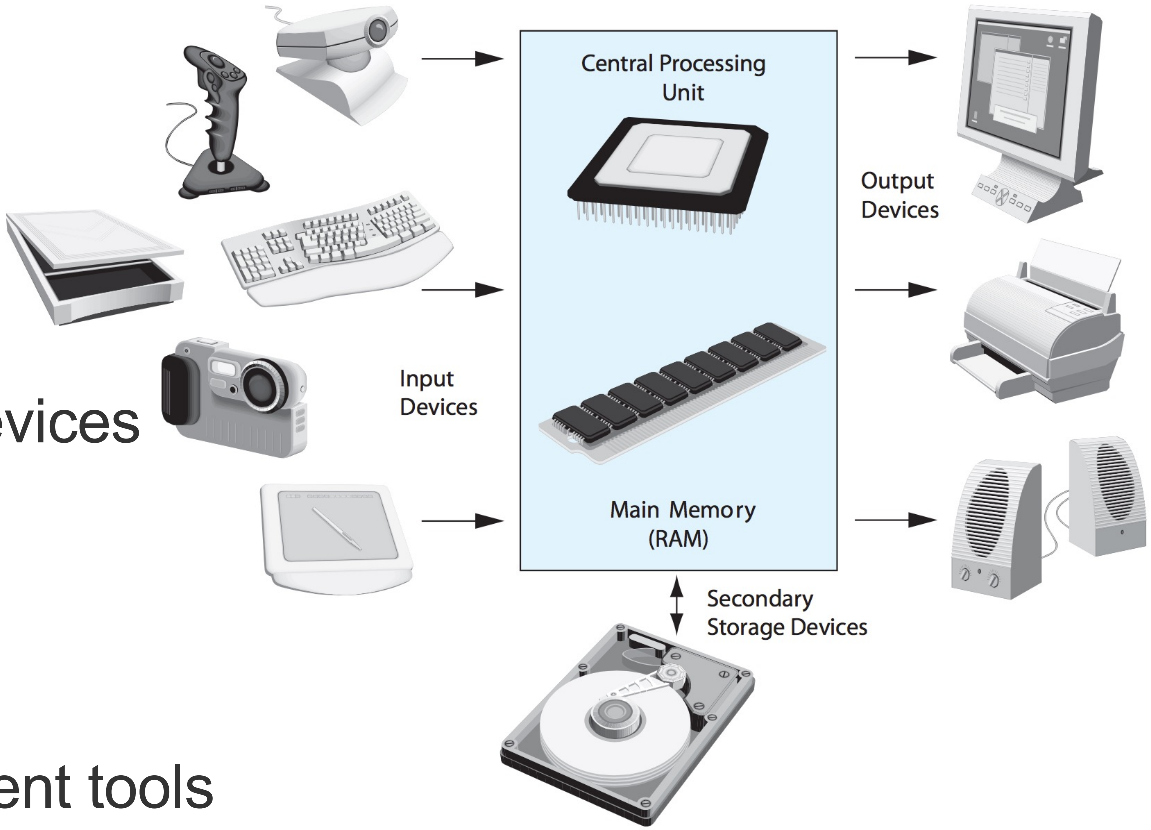
Problems with Computer!

- One thing at a time
- Basic Steps (fast)
- Limited Space
- Limited Language

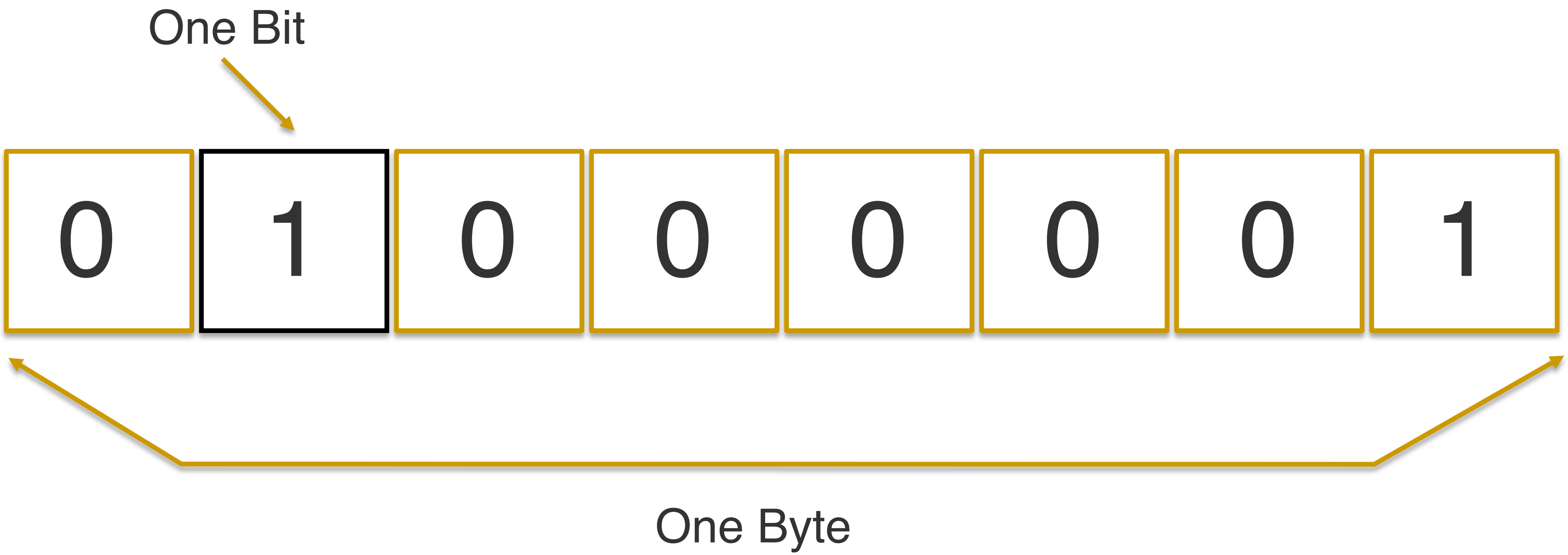


Hardware and Software

- Hardware
  - CPU
  - Main Memory
  - Secondary Storage
  - Input and Output devices
- Software
  - Operating systems
  - Utility programs
  - Software development tools
  - Applications



How Computers Store Data



One Bit

0	1	0	0	0	0	0	1

One Byte

A → 65 → 10000001

How a Program Works

- CPU uses the “fetch-decode-execute” cycle for processing
  - Fetch: Reads instructions from memory
  - Decode: Decodes the instructions that were just read to determine how to perform operations
  - Execute: Actually performs the operations

Programming Languages

Machine Language

```
10101010
11001100
11100011
10001000
```

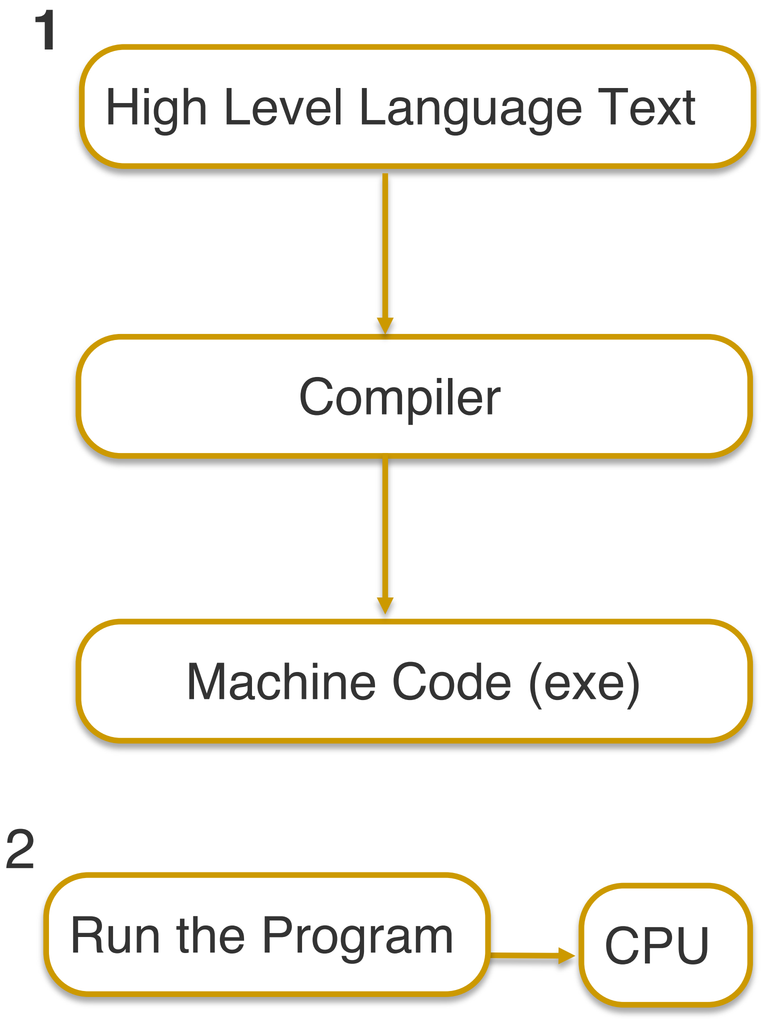
Assembly

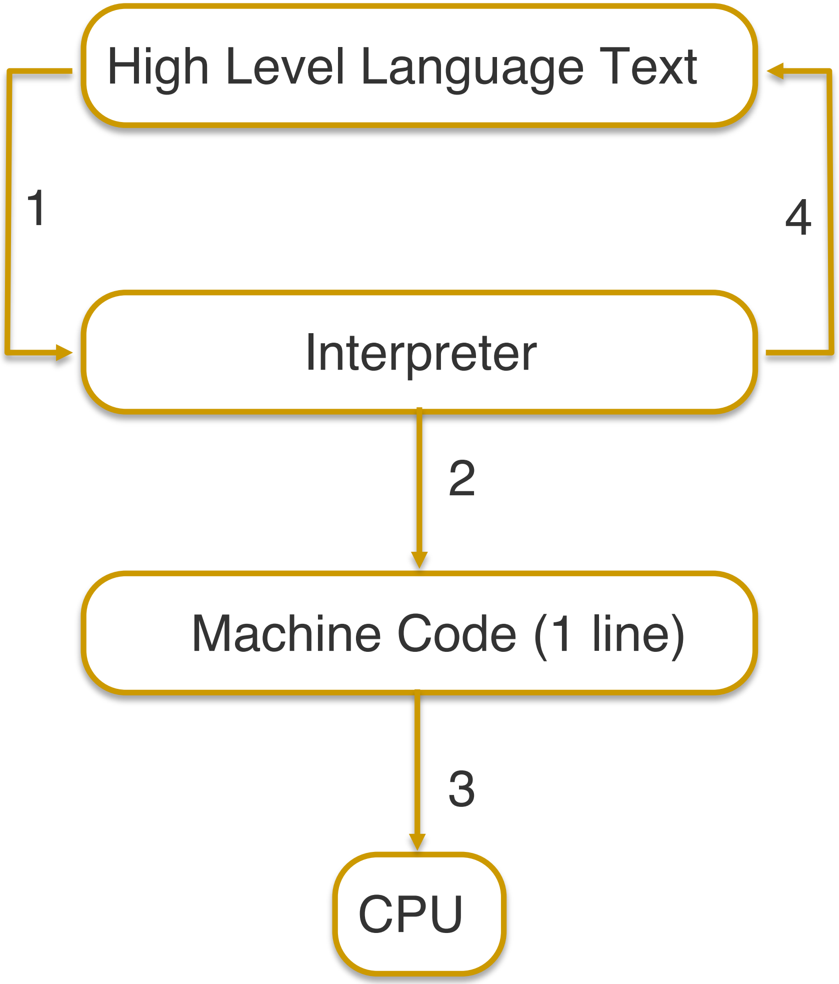
```
MOV    R1, 0x1000
MOV    R2, 0x1004
ADD    R1,R2
STA    0x1008
```

High Level Language

```
C = A + B
```

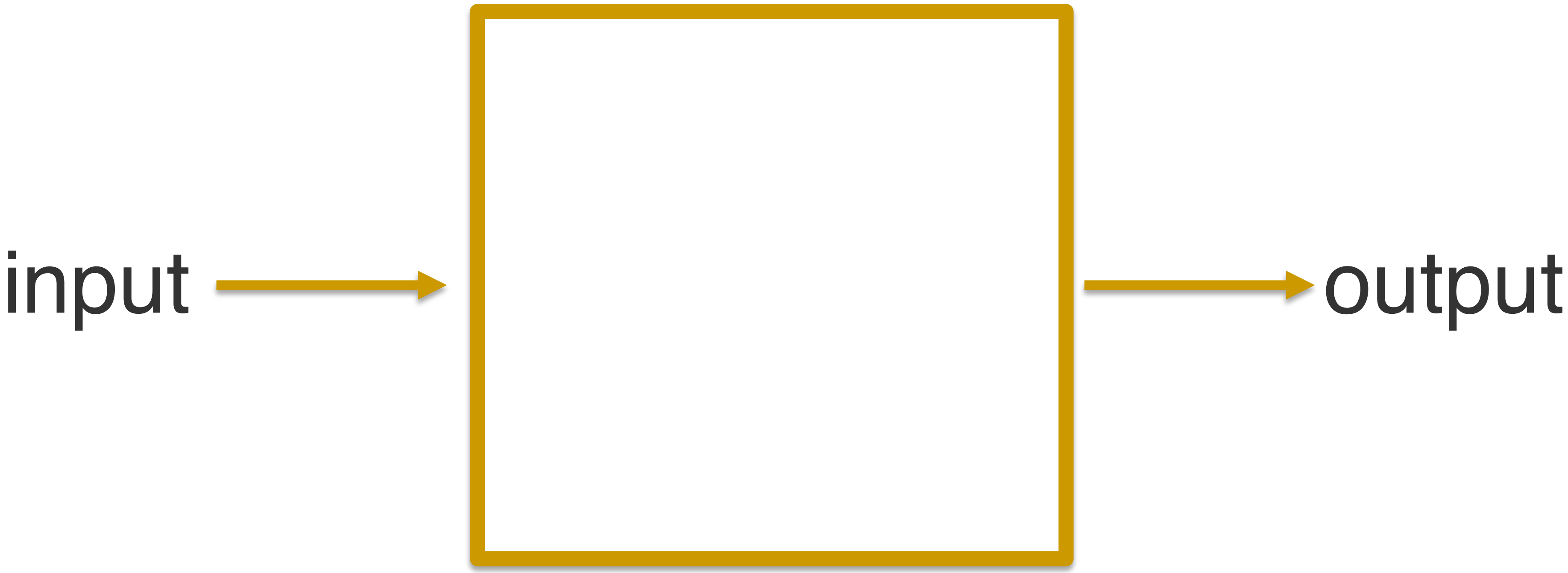
Compilers vs Interpreters

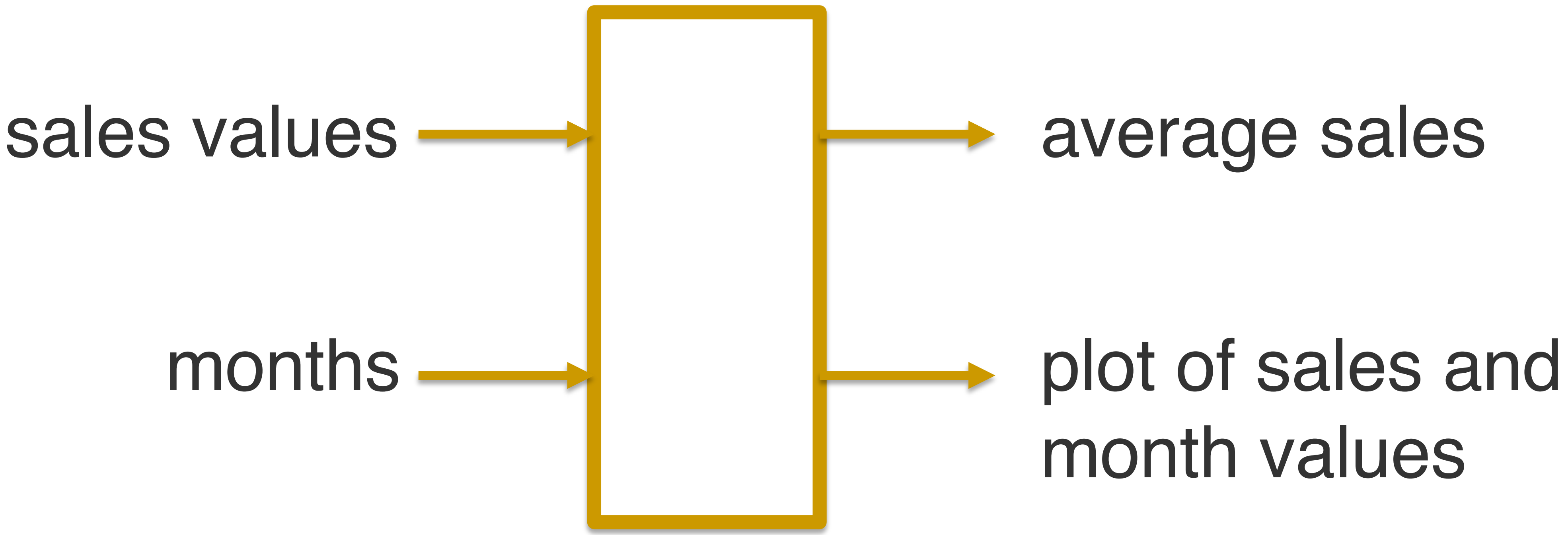




Computers

- Hardware
- Software
- Digital
- Fetch-Decode-Execute
- Compiler vs Interpreter


input
output



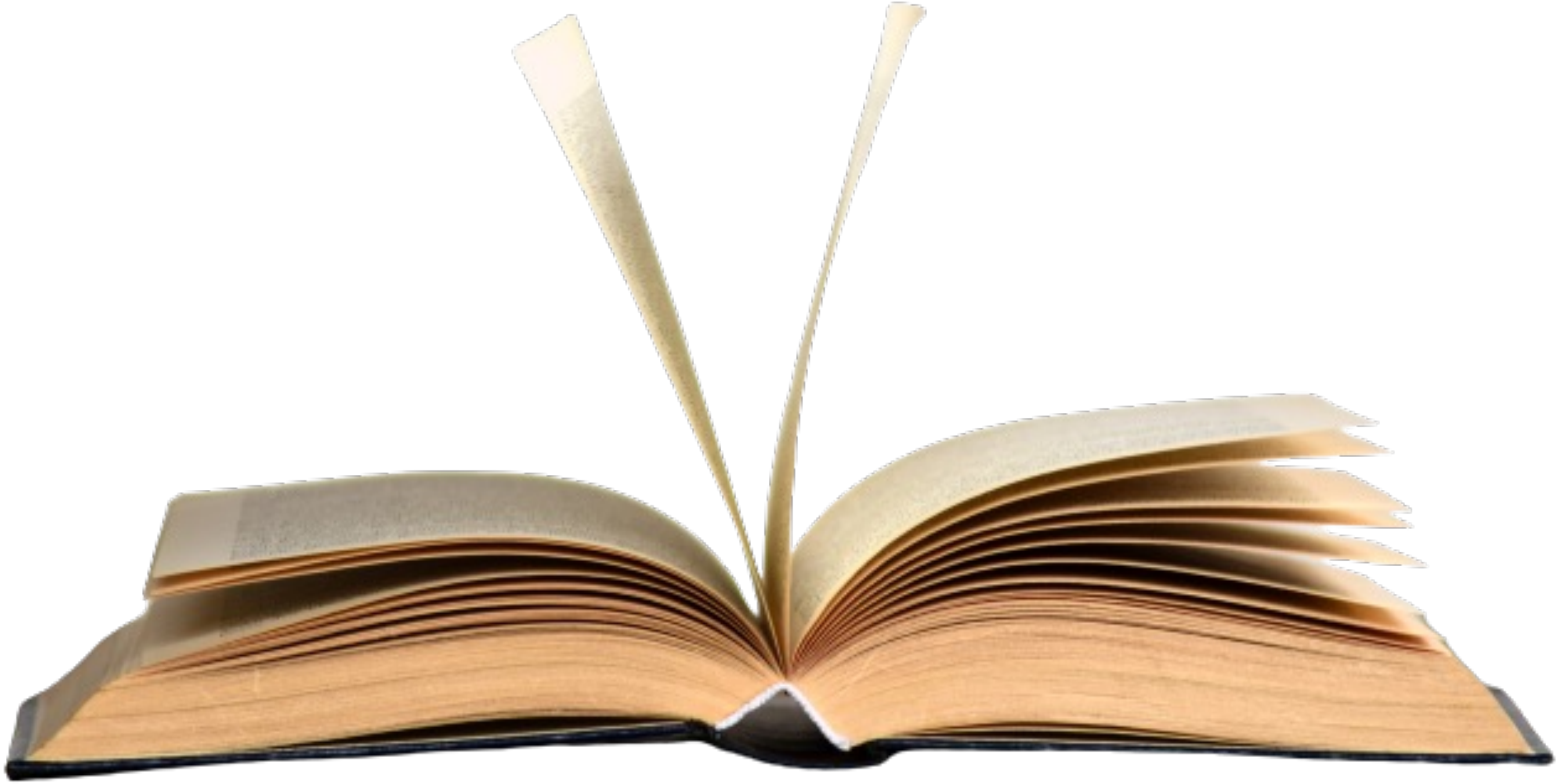
sales values

months

average sales

plot of sales and month values

algorithms



Is this algorithm, turning the pages, step by step, looking for the definition of "Programming" correct?

Instead of looking for the definition one page at a time, lets do two, four, six, eight, pages at a time. It it is faster. Is this algorithm correct?

Pseudocode

The following example is inspired from Harvard's CS50x OpenCourseWare, see https://cs50.harvard.edu/x/2021/weeks/0/

Pseudocode

```
Pick up dictionary
Open to middle of dictionary
Look at page
If word is on page
    Read definition
Else if word is earlier in book
    Open to middle of left half of book
    Go back to line 3
Else if word is later in book
    Open to middle of right half of book
    Go back to line 3
Else
    Quit
```

Functions

```
Pick up dictionary
Open to middle of dictionary
Look at page
If word is on page
    Read definition
Else if word is earlier in book
    Open to middle of left half of book
    Go back to line 3
Else if word is later in book
    Open to middle of right half of book
    Go back to line 3
Else
    Quit
```

Selection

```
Pick up dictionary
Open to middle of dictionary
Look at page
If word is on page
    read definition
Else if word is earlier in book
    Open to middle of left half of book
    Go back to line 3
Else if word is later in book
    Open to middle of right half of book
    Go back to line 3
Else
    Quit
```

Boolean Expressions

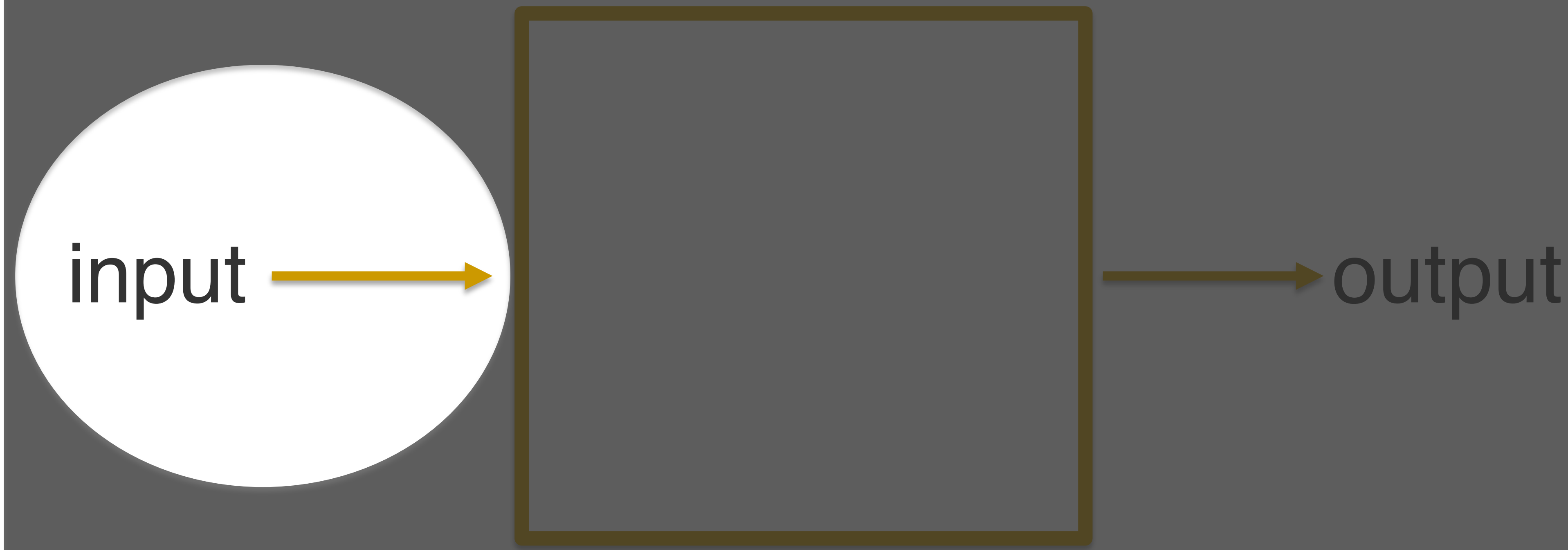
```
Pick up dictionary
Open to middle of dictionary
Look at page
If word is on page
    read definition
Else if word is earlier in book
    Open to middle of left half of book
    Go back to line 3
Else if word is later in book
    Open to middle of right half of book
    Go back to line 3
Else
    Quit
```

Loops

```
Pick up dictionary
Open to middle of dictionary
Look at page
If word is on page
    read definition
Else if word is earlier in book
    Open to middle of left half of book
    Go back to line 3
Else if word is later in book
    Open to middle of right half of book
    Go back to line 3
Else
    Quit
```

Programming

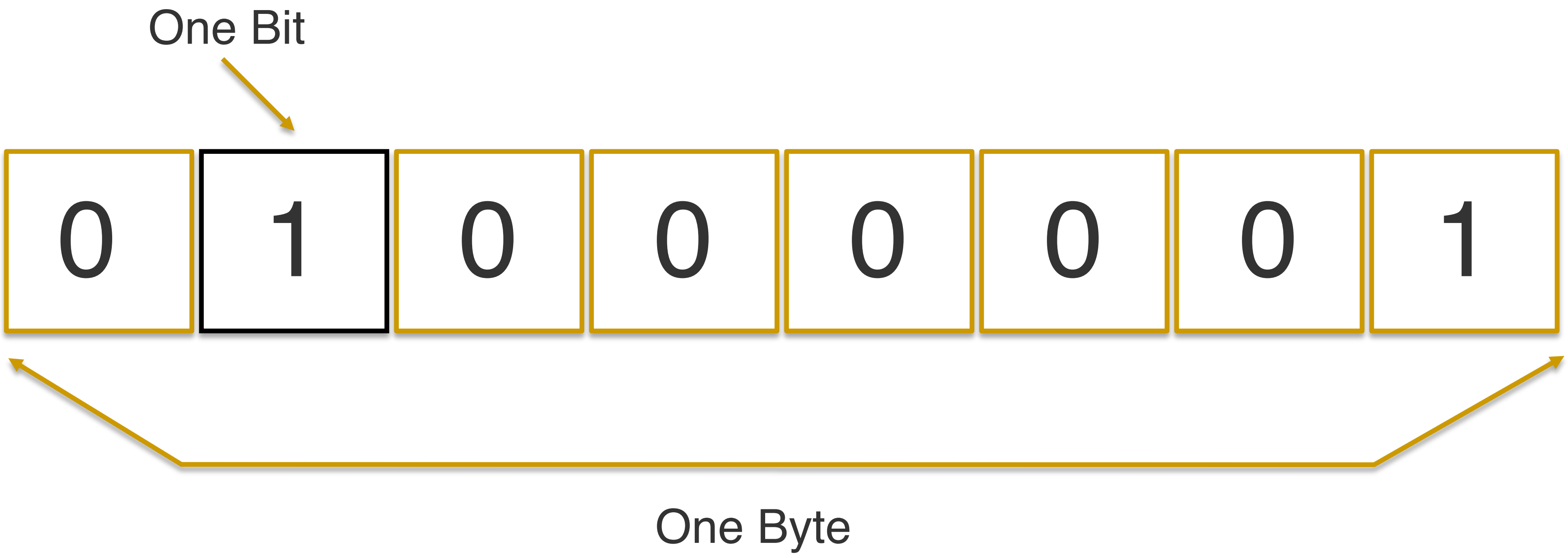
- Input/Output
- Algorithms
- Pseudocode
- Functions
- Conditions
- Boolean expressions
- Loops



input

output

Recall - How Computers Store Data



One Bit

0 1 0 0 0 0 0 1

One Byte

A → 65 → 10000001

Decimal (base-10) Digits

0 1 2 3 4 5 6 7 8 9

Decimal (base-10)

1 2 3

Decimal (base-10)

100	10	1
1	2	3

Decimal (base-10)

100	10	1
#	#	#

Decimal (base-10)

10^2	10^1	10^0
#	#	#

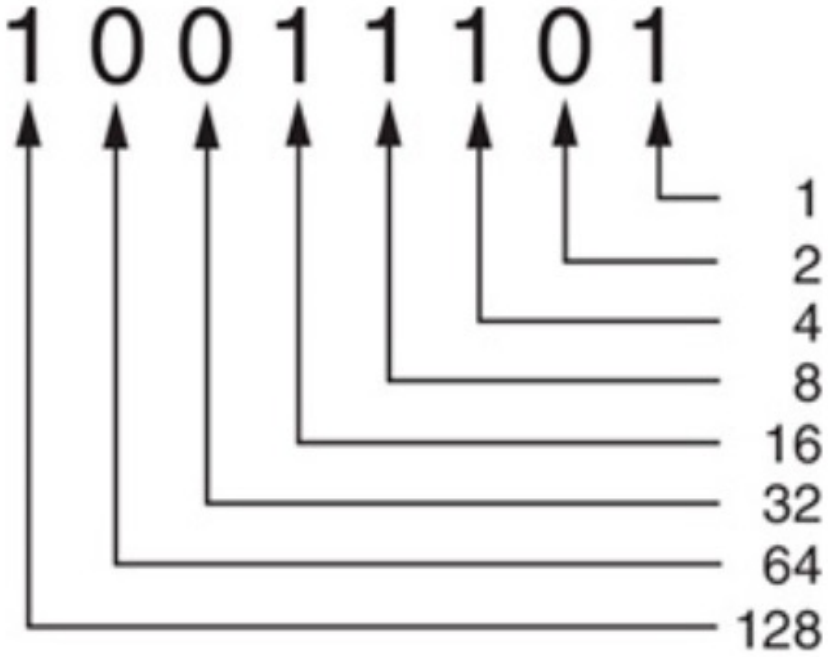
Binary (base-2) Digits

0 1

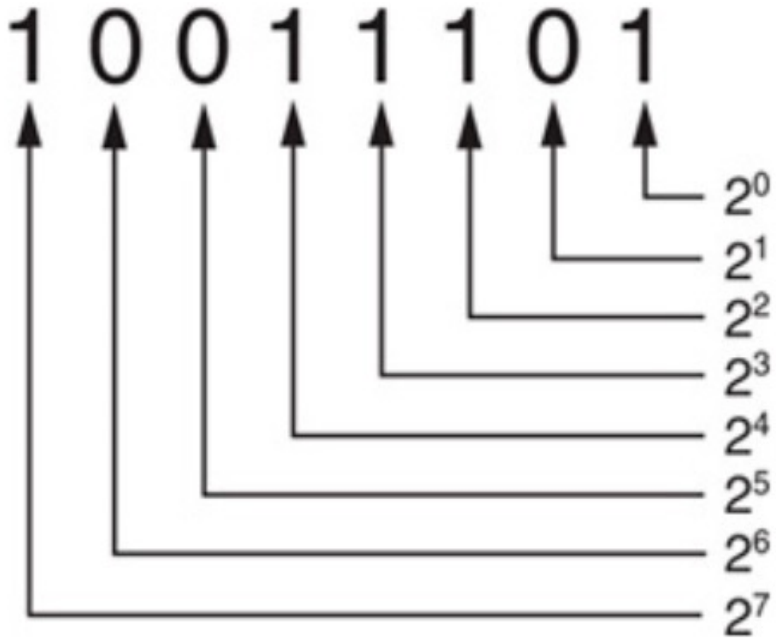
Binary (base-2)

2^2	2^1	2^0
#	#	#

Values of binary digits



Values of binary digits



... A B C D E F G H...

... 65 66 67 68 69 70 71 72...

Text

72 73 33

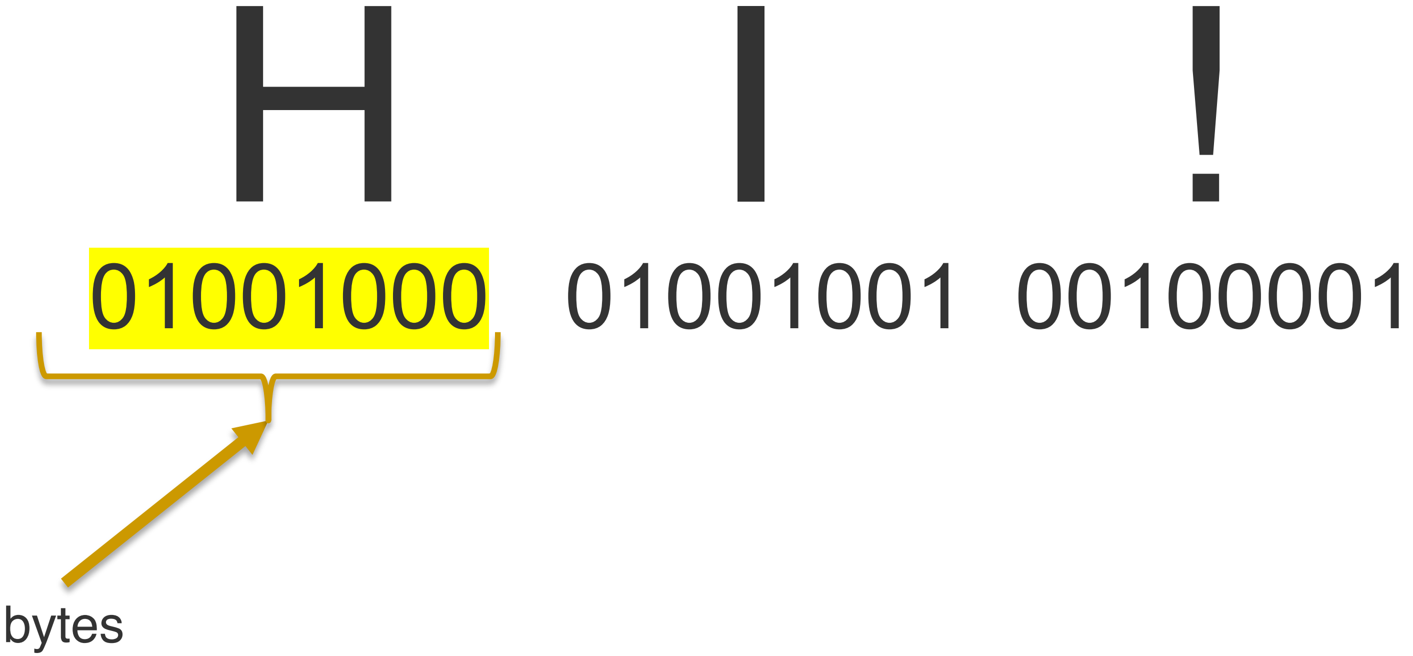
H	I	
72	73	33

```
The decimal set:

  0 nul    1 soh    2 stx    3 etx    4 eot    5 enq    6 ack    7 bel
  8 bs     9 ht    10 nl    11 vt    12 np    13 cr    14 so    15 si
 16 dle   17 dc1   18 dc2   19 dc3   20 dc4   21 nak   22 syn   23 etb
 24 can   25 em    26 sub   27 esc   28 fs    29 gs    30 rs    31 us
 32 sp    33  !    34  "    35  #    36  $    37  %    38  &    39  '
 40  (    41  )    42  *    43  +    44  ,    45  -    46  .    47  /
 48  0    49  1    50  2    51  3    52  4    53  5    54  6    55  7
 56  8    57  9    58  :    59  ;    60  <    61  =    62  >    63  ?
 64  @    65  A    66  B    67  C    68  D    69  E    70  F    71  G
 72  H    73  I    74  J    75  K    76  L    77  M    78  N    79  O
 80  P    81  Q    82  R    83  S    84  T    85  U    86  V    87  W
 88  X    89  Y    90  Z    91  [    92  \    93  ]    94  ^    95  _
 96  `    97  a    98  b    99  c   100  d   101  e   102  f   103  g
104  h   105  i   106  j   107  k   108  l   109  m   110  n   111  o
112  p   113  q   114  r   115  s   116  t   117  u   118  v   119  w
120  x   121  y   122  z   123  {   124  |   125  }   126  ~   127 del
```

H	I	!
72	73	33

H	I	!
01001000	01001001	00100001



H I !

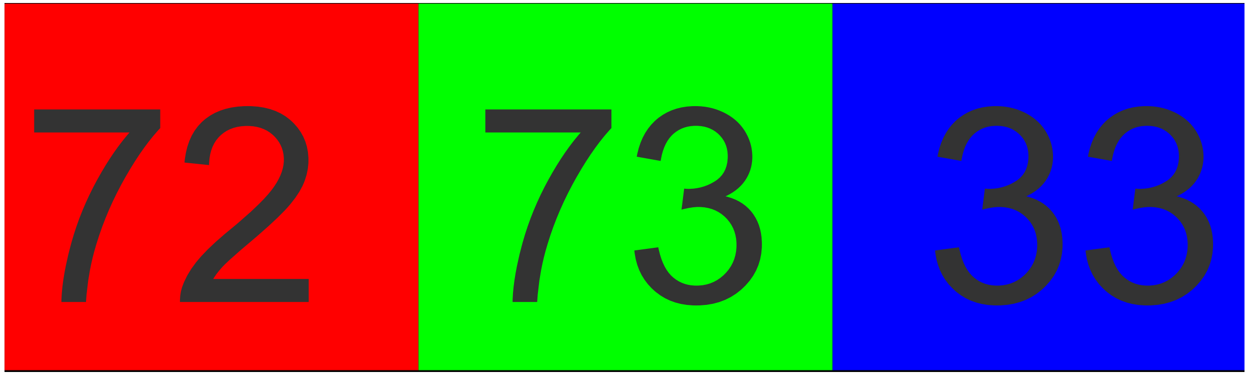
01001000 01001001 00100001

bytes

Images

72 73 33

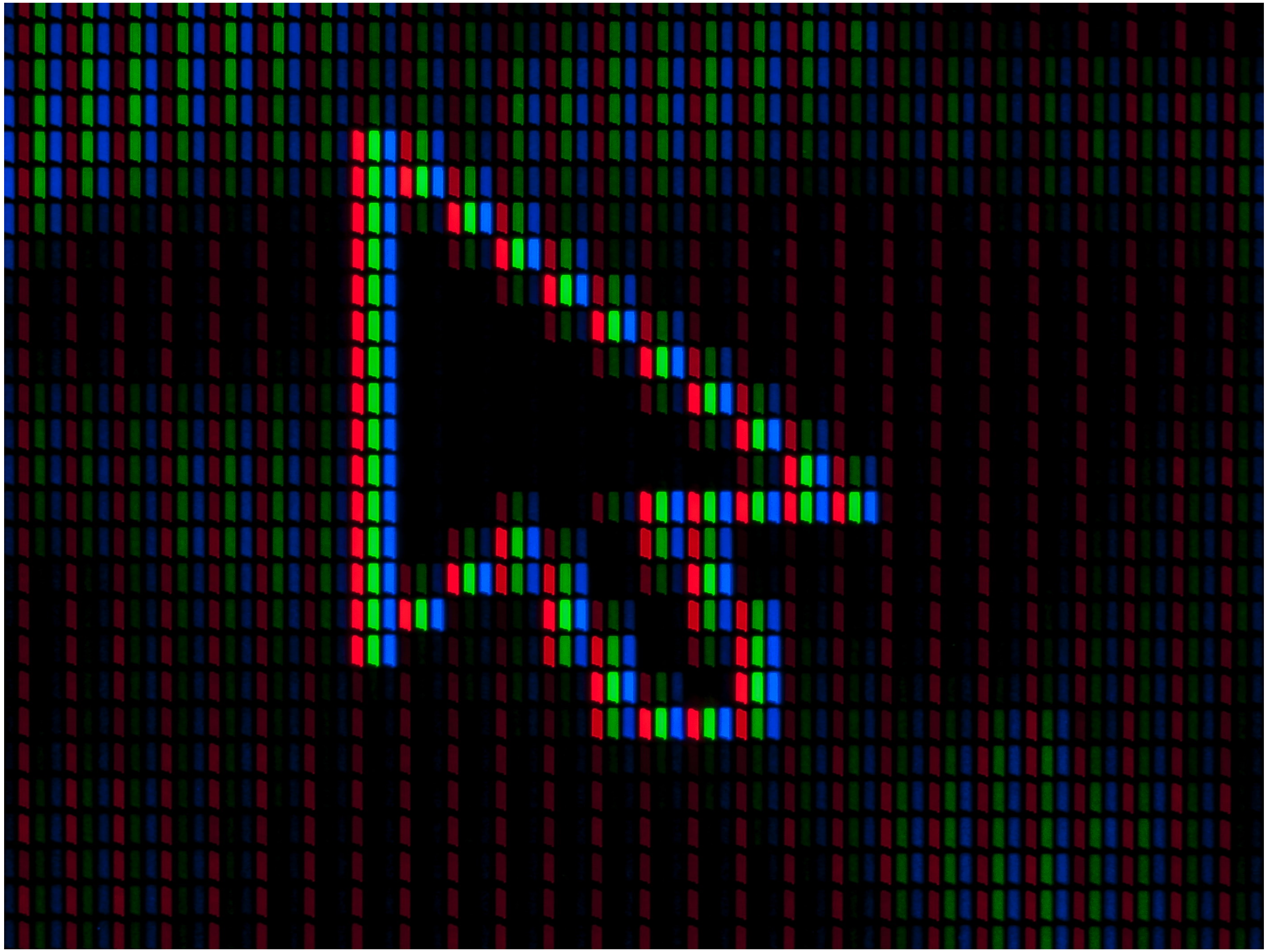
Images



72 73 33

Images

Images



Inputs

- Binary
- Agree on formats
  - Text
  - Images
  - Audio
  - Video