CMPSC 100

Computational Expression

Java

Represents an object which is screen output

Output has a characteristic (it can be printed in lines)

```
System.out.println("Hello, class!");
```

Complete statement ("method" call)

```java
/** Implements a Java "Hello, World!" program.
 *
 * @author Douglas Luman
 */
public class HelloWorld {

  /** Entry point.
   *
   * @param args The command line arguments
   */
  public static void main(String[] args) {
    // The following prints a single line to the screen
    System.out.println("Hello, World!");
  }
}
```

Classes

Methods

Statements

```java
/** Implements a Java "Hello, World!" program.
 *
 * @author Douglas Luman
 */
public class HelloWorld {

  /** Entry point.
   *
   * @param args The command line arguments
   */
  public static void main(String[] args) {
    // The following prints a single line to the screen
    System.out.println("Hello, World!");
  }
}
```

```
3   /** Implements a Java "Hello, World!" program.
4    *
5    * @author Douglas Luman
6    */
7   public class HelloWorld {
8
9     /** Entry point.
10     *
11     * @param args The command line arguments
12     */
13    public static void main(String[] args) {
14      // The following prints a single line to the screen
15      System.out.println("Hello, World!");
16    }
17  }
```

Atom with “Indent Guide” turned on

Markdown

- A “convention” (agreed format)
- Emphasizes document structure using a “hierarchy”
  - Headings
  - Paragraphs
  - Lists
  - ...and more
- Interpreted by web browsers to display clear documents
  - Raw Markdown isn’t necessarily “pretty”
- We will use “GitHub Flavored Markdown” in this class (see pocket guides on your table)

In this practical session, we focus on writing our initial Java program, the "Hello, World!". We continue to pr

```
* [Slack](https://cmpsc-100-
* [GitHub](https://www.githu
* git
* Markdown
* [Atom](https://atom.io)
* [Docker](https://www.docke
* GatorGrader
* gradle

## Table of contents

* [Evaluation](#evaluation)
* [Accepting the assignment]
* [The "Hello, World!"](#the
* [GatorGrader](#gatorgrader

## General guidelines for pr

* **Experiment!** We design
```

- Slack
- GitHub
- git
- Markdown
- Atom
- Docker
- GatorGrader
- gradle

Table of contents

General guidelines for practical sessions

- **Experiment!** We design practical sessions to create a space for you to *try things*. Given the expertise of our classroom TLs and my interest in fixing stuff, I am sure that even if something breaks, we can fix it.
- **Complete *something*.** Grading for practical assignments hinges on *completion*. As long as you provide a good faith effort to finish a task, your grade should reflect your effort.
- **Practice skills.** If you work in the discipline of computer science, many of the skills you revisit or establish here are

```
se
stuff, I am sure that even if something breaks, we can fix it.
* **Complete _something_.** Grading for practical assignments hinges on _completion_. As long as you provide a
should reflect your effort.
```

Activity

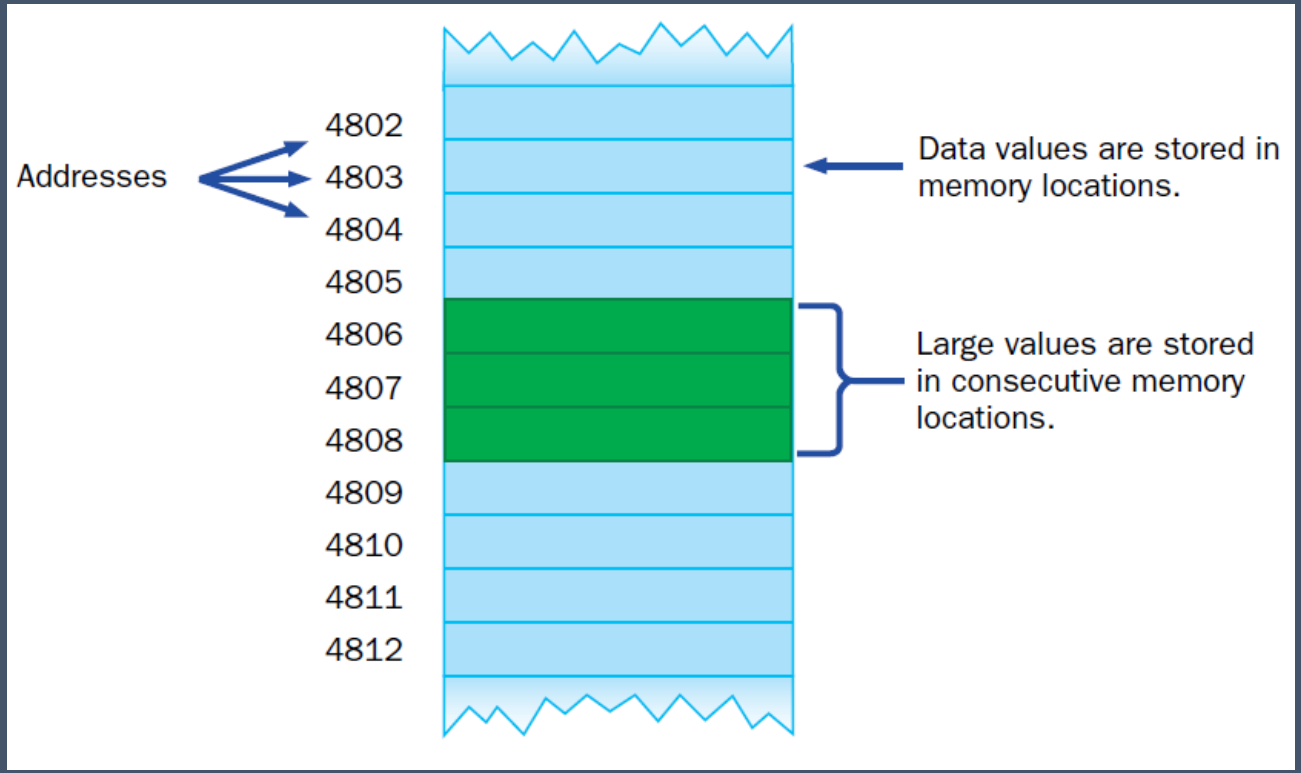
- In your terminal, `cd` to your class activities folder
  - Should be located in your `~/Desktop/CMPSC100` folder
  - Type `ls` to see your copy and `cd` into it
- Go to the `#class-activities` channel in our course Slack.
  - Copy and paste the command from the channel into your terminal.
  - Perform the following command:

    ```
    git pull download master
    ```

Activity

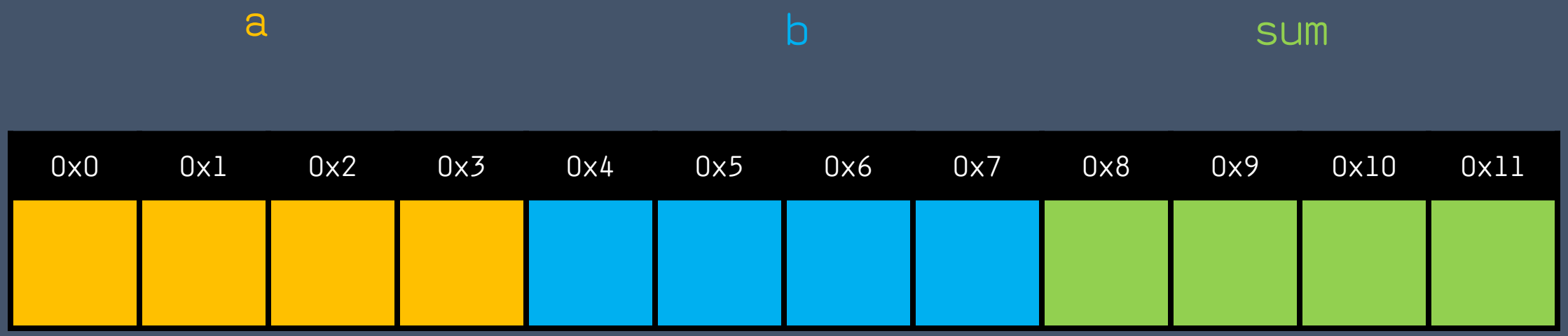
- Your Markdown the writing/activity.md file should contain at least one of each of the following:
  - 5 headings of “descending hierarchy”
  - 1 paragraph
    - At least one word or phrase in **bold** type
    - At least one word or phrase in *italic* type
  - 1 list
  - 1 “fenced” code block using Java formatting
    - Use a line of Java that you already know (print, println)
  - 1 image
    - URL provided in Slack channel
  - 1 link
    - URL provided in Slack channel

Assignments and variables



Assignments and variables

- Java programs use variables to store information in secondary ("working") memory
- These stored values use "identifiers" for easy reference



Assignments and variables

type

assignment operator

```
int cardA = 2;
```

identifier

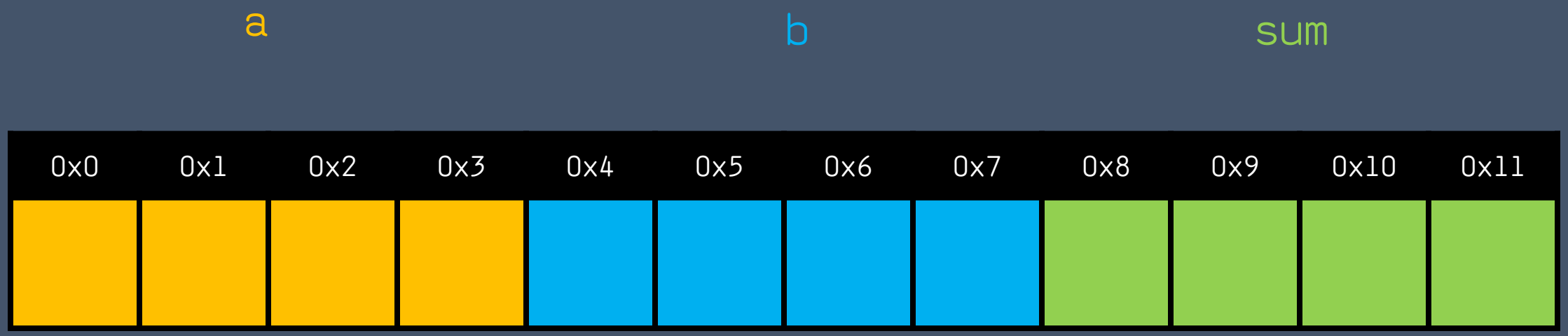
expression (value)

Parts of an assignment statement

Assignments and variables

```
int a = -56;
int b = 91;
int sum = a + b; // 35
```

"signed" integers



Assignments and variables

Assignment operator

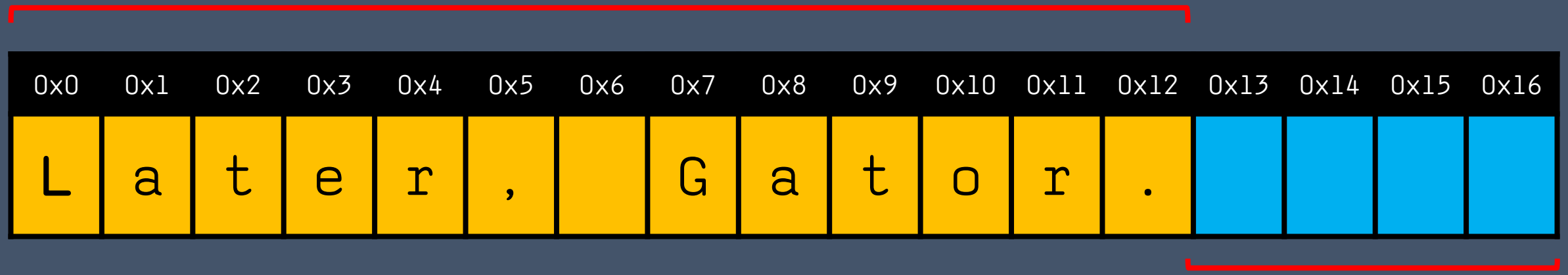
Locations in secondary “working” memory

```
String goodbye = “Later, Gator.”;
```

Data type:	String
Value:	“Later, Gator.”
Size:	12 bytes

Assignments and variables

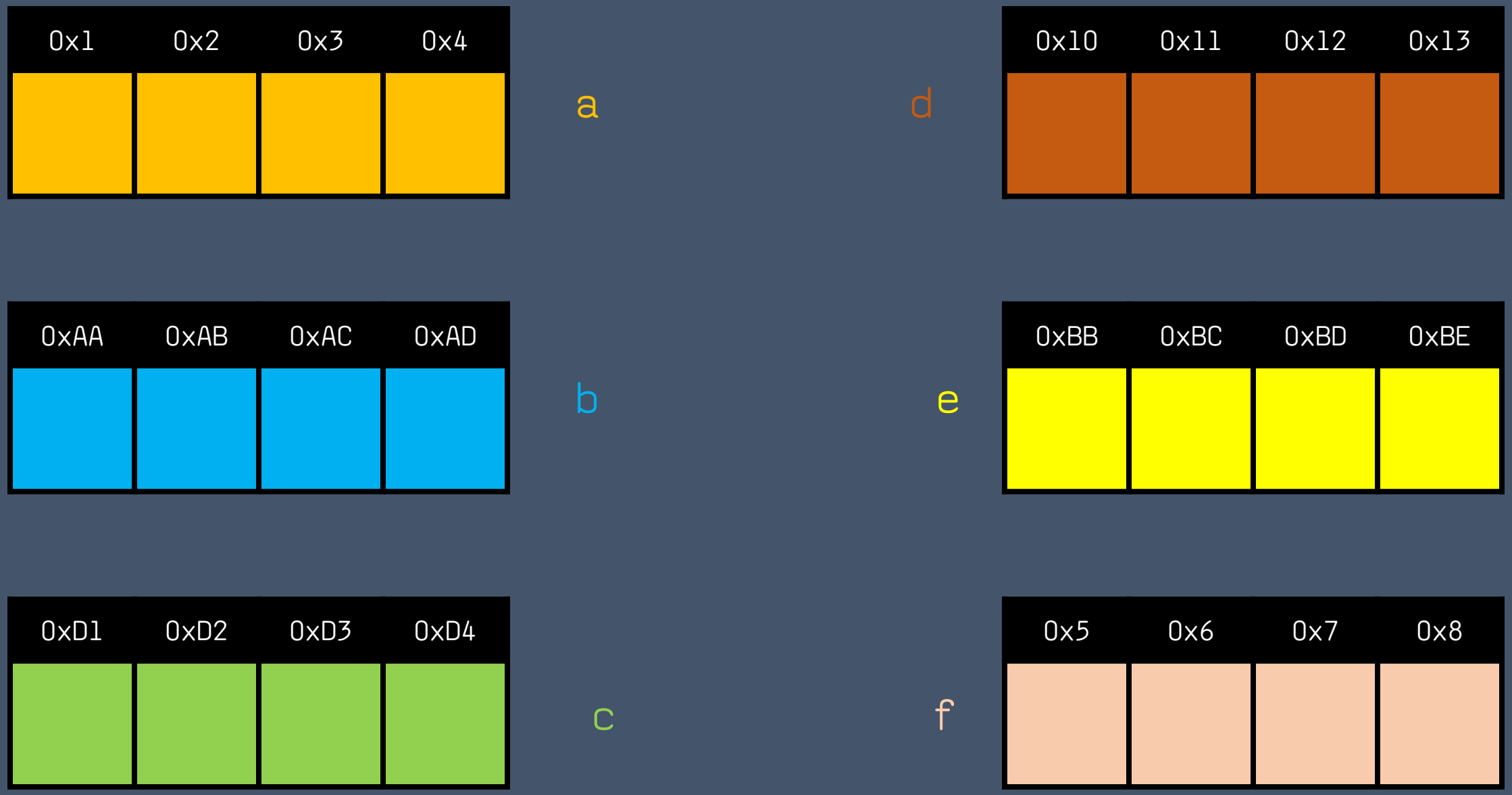
Data type: String
Value: “Later, Gator.”
Size: 12 bytes

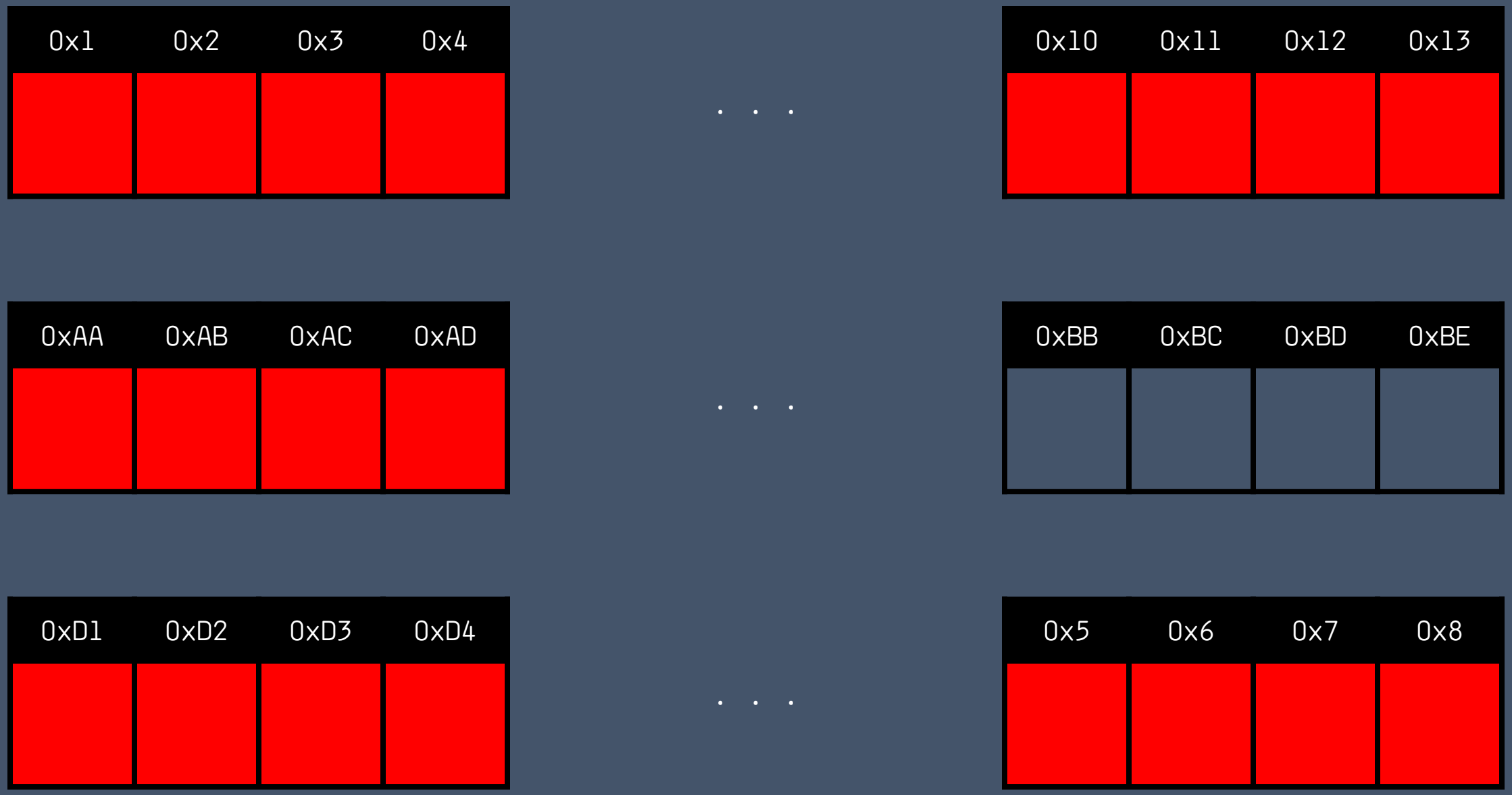


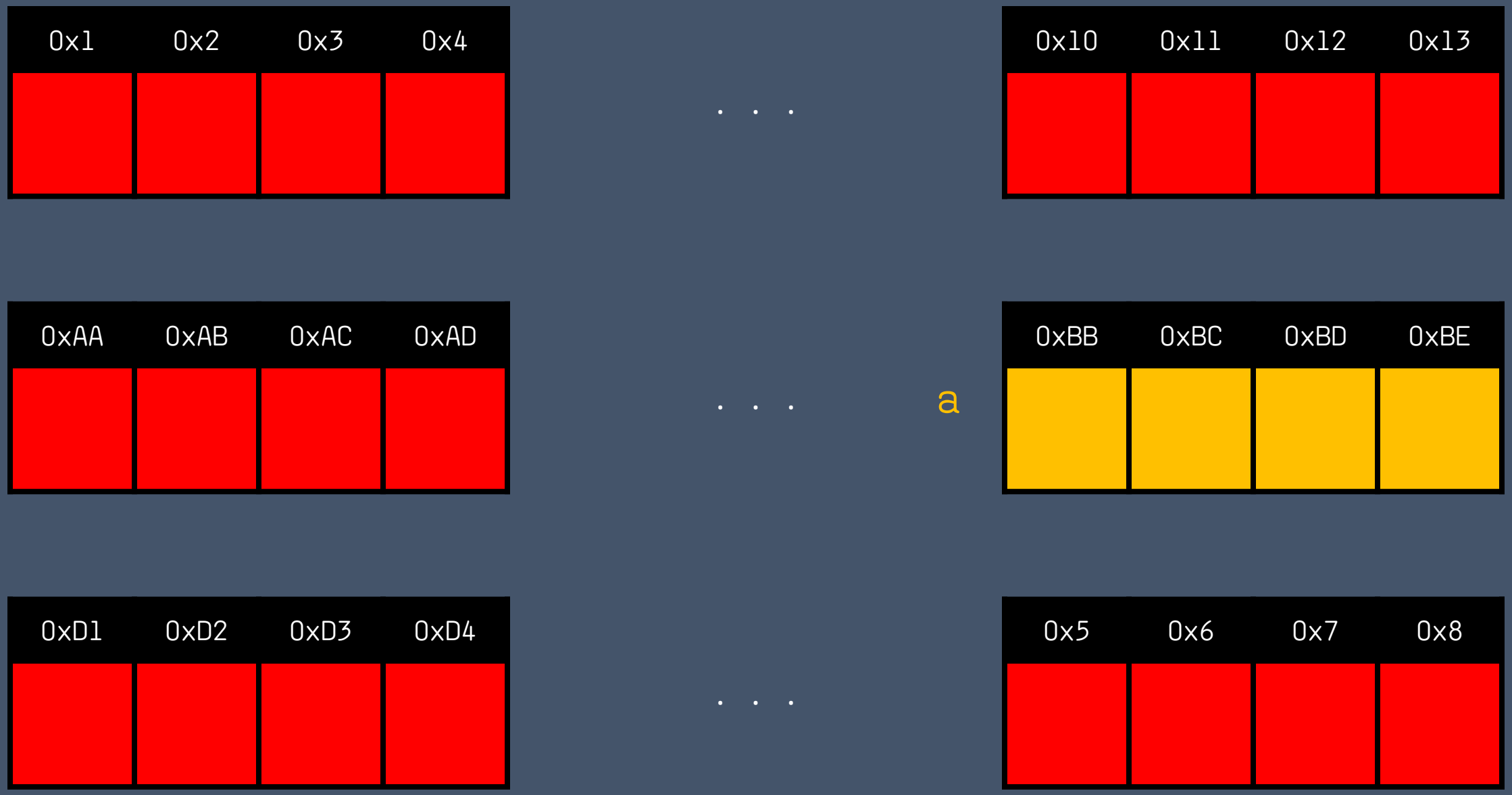
Data type: integer
Value: 48
Size: 4 bytes


a
0x1
0x2
0x3
0x4
. . .
b
0xAE
0xAF
0xB0
0xB1


b
0x1
0x2
0x3
0x4
. . .
a
0xAE
0xAF
0xB0
0xB1

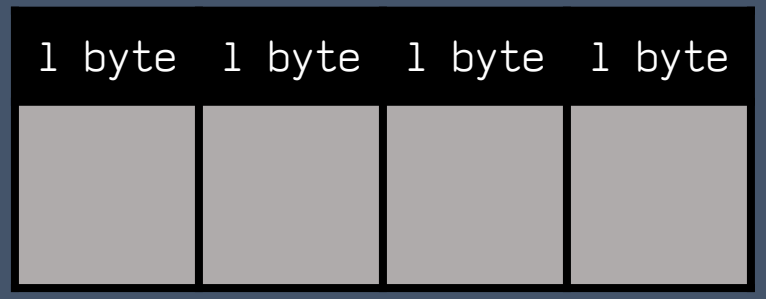

0x1
0x2
0x3
0x4
a
0xAA
0xAB
0xAC
0xAD
b
0xD1
0xD2
0xD3
0xD4
c
d
0x10
0x11
0x12
0x13
e
0xBB
0xBC
0xBD
0xBE
f
0x5
0x6
0x7
0x8


0x1
0x2
0x3
0x4
0x10
0x11
0x12
0x13
0xAA
0xAB
0xAC
0xAD
0xBB
0xBC
0xBD
0xBE
0xD1
0xD2
0xD3
0xD4
0x5
0x6
0x7
0x8

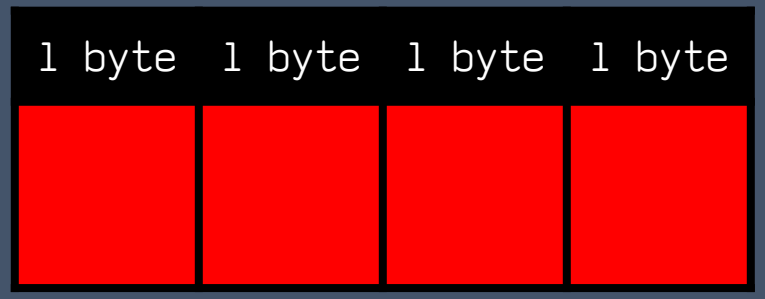

0x1
0x2
0x3
0x4
. . .
0x10
0x11
0x12
0x13
0xAA
0xAB
0xAC
0xAD
. . .
a
0xBB
0xBC
0xBD
0xBE
0xD1
0xD2
0xD3
0xD4
. . .
0x5
0x6
0x7
0x8

Write down your number, label it “working total”
Multiply working total by 2 and write this down
Add 2 to working total and write this down
Multiply working total by 5
Subtract the number I give you from working total

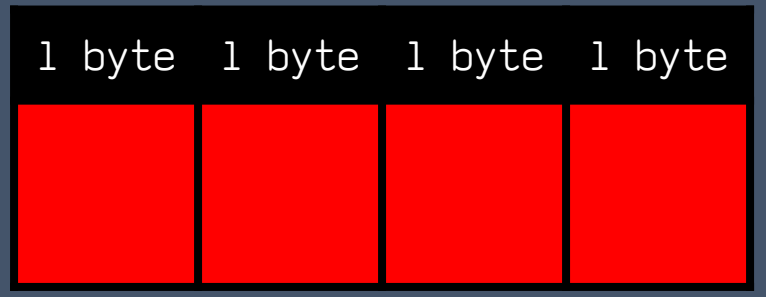
```
int cardA = 8;
```




```
int cardB = 7;
```




```
int workingTotal;
```




```
workingTotal = cardB
workingTotal = workingTotal * 2;
workingTotal = workingTotal + 2;
workingTotal = workingTotal * 5;
workingTotal = workingTotal - (10 - cardA);
```