“It All Depends Upon How You Look At It” Lab

### Introduction

During this lab you will learn:

* How the central processing unit (CPU) knows whether the contents of a particular word in memory is an instruction or data, and, if the latter, the type of the data.
* How to interpret character and floating point data.
* About some of the consequences of a CPU’s “knowledge” of the contents of memory.

### Starters

Type your answers into the text box following each question. You may work with someone else on this lab, and submit one copy of this, including your answers, on Canvas.

*What are the names of everyone who worked on this lab with you, including yourself?*

|  |
| --- |
|  |

### Instructions vs. Data

Referring to the buffer overflow exploit, we discussed how it's possible to “trick” a CPU into executing data as code. Explain why this isn't possible with the Hack CPU.

|  |
| --- |
|  |

### Floating-Point Data

* + 1. Referring to the Wikipedia article on half-precision floating point format ( <https://en.wikipedia.org/wiki/Half-precision_floating-point_format>), perform the following conversions.

Convert -0.25 to half-precision floating-point format. Hint: -0.25 decimal is -0.01 binary. Show your work, leading to your answer, in the text box below. Express your answer in hexadecimal.

|  |
| --- |
|  |

**0X5140** is a half-precision floating-point format number expressed in hexadecimal. Convert it to its decimal value. Show your work, leading to your answer, in the text box below.

|  |
| --- |
|  |

### Representing Characters

Referring to <http://www.neurophys.wisc.edu/comp/docs/ascii/> , convert the characters of

**Hello world!**

to ASCII, expressing each character as two hexadecimal digits. The first two character conversions are completed for you.

|  |
| --- |
| 48 65 |

Why is Unicode “superior” to ASCII? Give a specific example.

|  |
| --- |
|  |

Decode the following Unicode code point sequence.

**U+0043 U+0055 U+0020 U+004C U+0054 U+0052 U+0020 U+1F44B U+1F600**

|  |
| --- |
|  |

### Representing Strings

Refer to <https://en.wikipedia.org/wiki/Null-terminated_string>.Consider a 32-bit memory (i.e., four bytes per word) containing this data:

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **D** |
| **E** | **F** | **G** | **H** |
| **\0** | **\0** | **\0** | **\0** |

where **\0** represents the null character.

What string does this represent on a little-endian machine?

|  |
| --- |
|  |

What string does this represent on a big-endian machine?

|  |
| --- |
|  |