**Student Activity Guide: von Neumann versus Dual Core Names \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Unit 3 Lesson 4 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Directions**: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Record your answers in the space provided.
* This activities requires a team of 5 students. Select one team member for each of these roles for the activities in Part 1 (the fifth person in your team will be assigned in part 2):
  + CPU – performs all commands.
  + Memory – holds values.
  + Runner – delivers information back and forth between the CPU and memory and can only hold one thing at a time.
  + Compiler – Delivers instructions to the CPU.
* Commands:
  + Load – Memory copies the value requested onto a card and the runner delivers it to the CPU.
  + Add – the CPU performs addition on two values loaded from memory.
  + Subtract – the CPU performs subtraction on two values loaded from memory.
  + Store – the CPU copies an answer onto a card and the runner delivers it to memory to be stored in the variable listed.

**Part 1: von Neumann Architecture**

The compiler should deliver the instructions one at a time and in the specified order to the CPU. Start with these two values in Memory:

x = 57 y = 19

Run the commands:

|  |
| --- |
| Load x  Add 30  Store x  Load y  Subtract 11  Store y |

1. What value is stored in x after this program is executed? \_\_\_\_\_\_\_\_
2. What value is stored in y after this program is executed? \_\_\_\_\_\_\_\_

Then, run a second program using the original values of x = 57 y = 19

|  |
| --- |
| Load x  Add 10  Store x  Load y  Subtract 1  Store y |

1. What value is stored in x after this program is executed? \_\_\_\_\_\_\_\_
2. What value is stored in y after this program is executed? \_\_\_\_\_\_\_\_

**Part 2: Multi Core Architecture**

In part two, you will add a second CPU. Remember that there is still **only one bus** so **only one runner at a time can deliver messages** from memory to the CPUs, so **the two programs should take turns** going back and forth.

Start with the values:

x = 11 y = 20

|  |  |
| --- | --- |
| Program 1 – CPU 1 | Program 2 – CPU 2 |
| Load x  Add 5  Store x  Load y  Subtract 6  Store y | Load x  Add 12  Store x  Load y  Subtract 5  Store y |

1. Trace program 1 with the starting x and y values. What is the final value of x? \_\_\_\_\_\_\_\_; y? \_\_\_\_\_\_\_\_
2. Trace program 2 with the starting x and y values. What is the final value of x? \_\_\_\_\_\_\_\_; y? \_\_\_\_\_\_\_\_
3. Now run the two programs concurrently acting out the steps of the program like you did in Part 1. Remember to make sure the runner only holds one command at a time.
4. What is the final value of x? \_\_\_\_\_\_\_\_; y? \_\_\_\_\_\_\_\_
5. Is this a correct result? How do you know?
6. Why did you get this result?
7. How could you improve the process?
8. What would change if we added 2 more CPUs?
9. What are some advantages of using multiple CPUs?