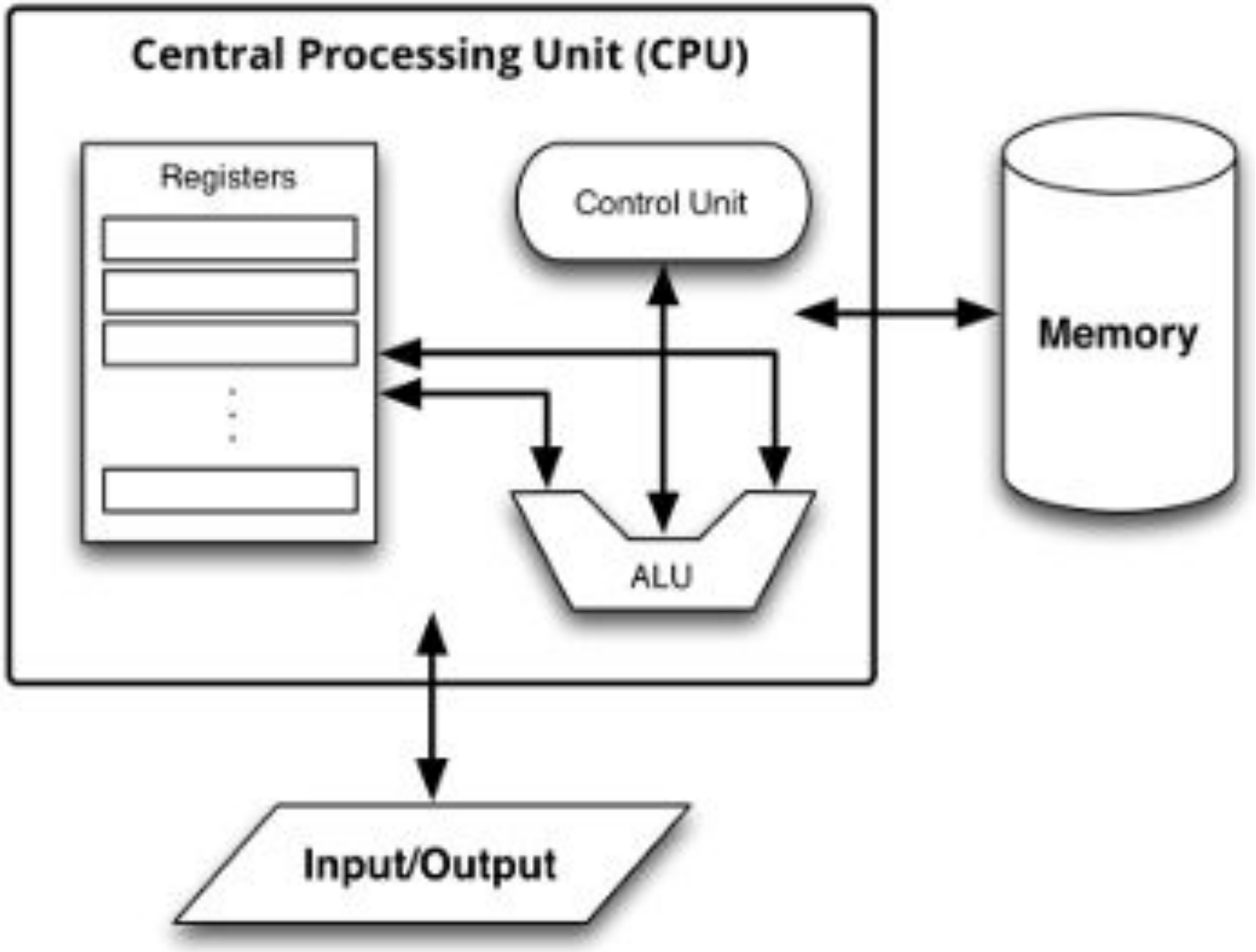
FIT9137
Introduction to Computer Architecture and Networks

Week 2: Computer Architecture - the Von Neumann Model and the MARIE Architecture

Safi Uddin

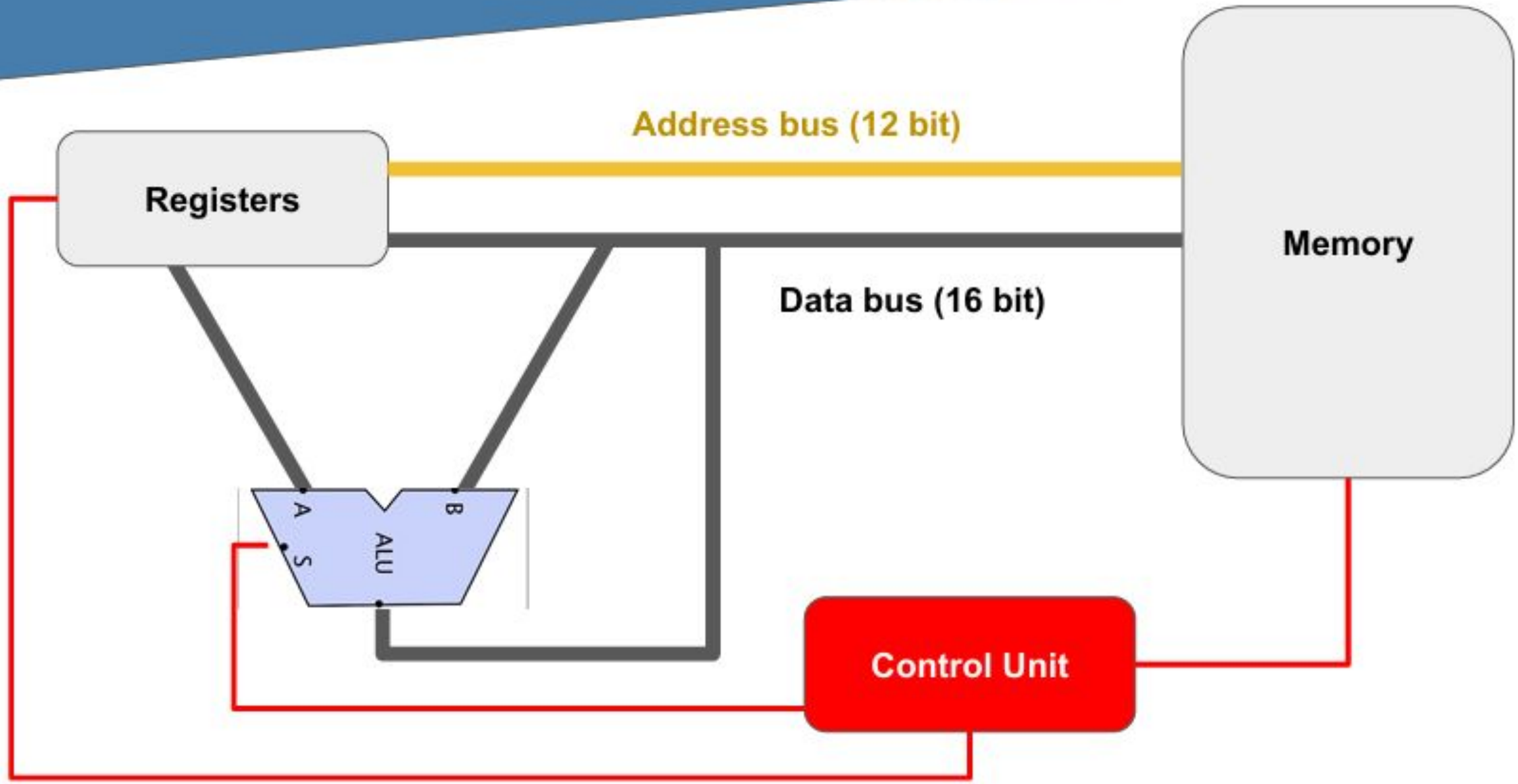
www.shutterstock.com • 548025055

The Von Neumann Model

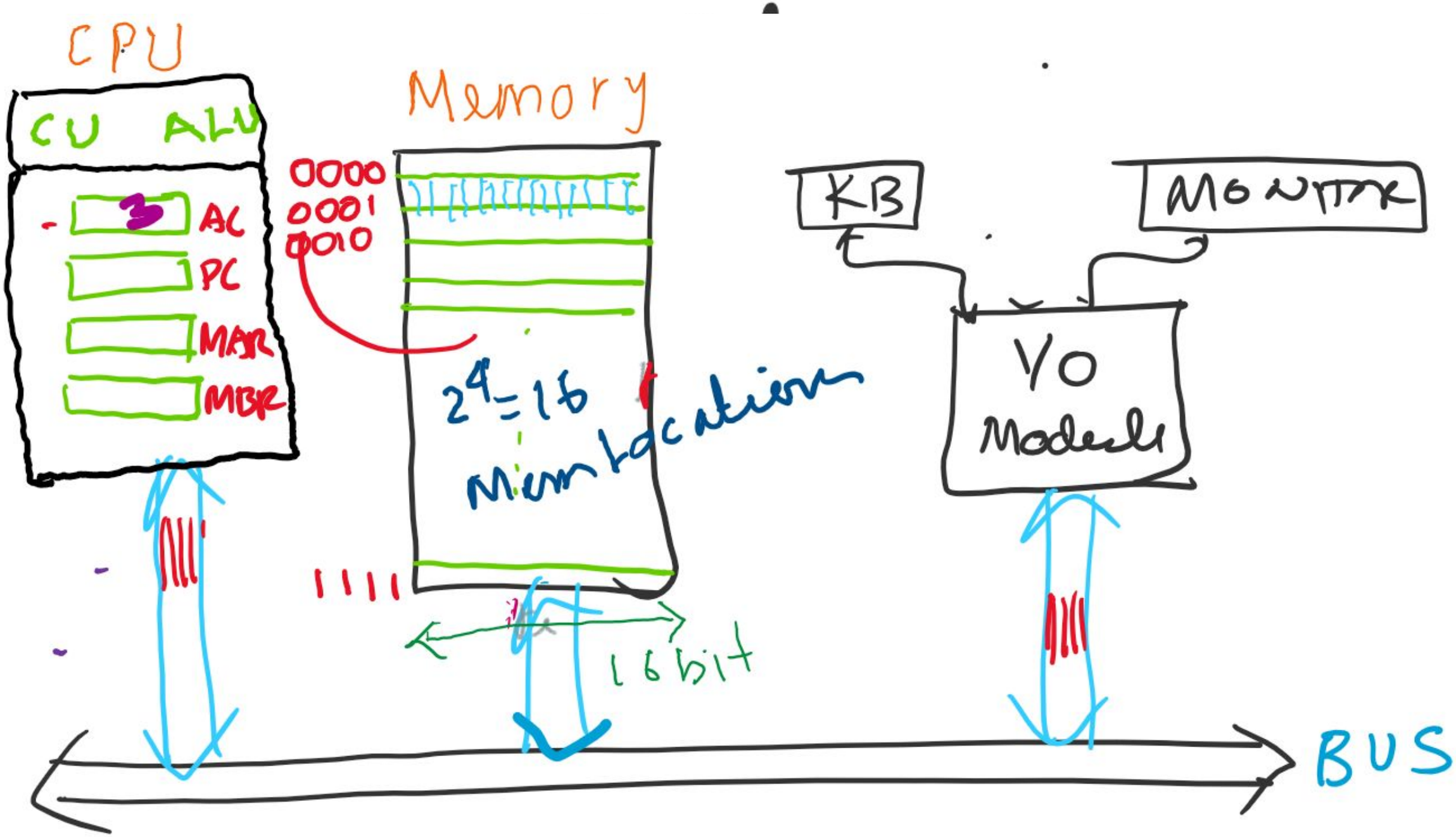


The MARIE Architecture

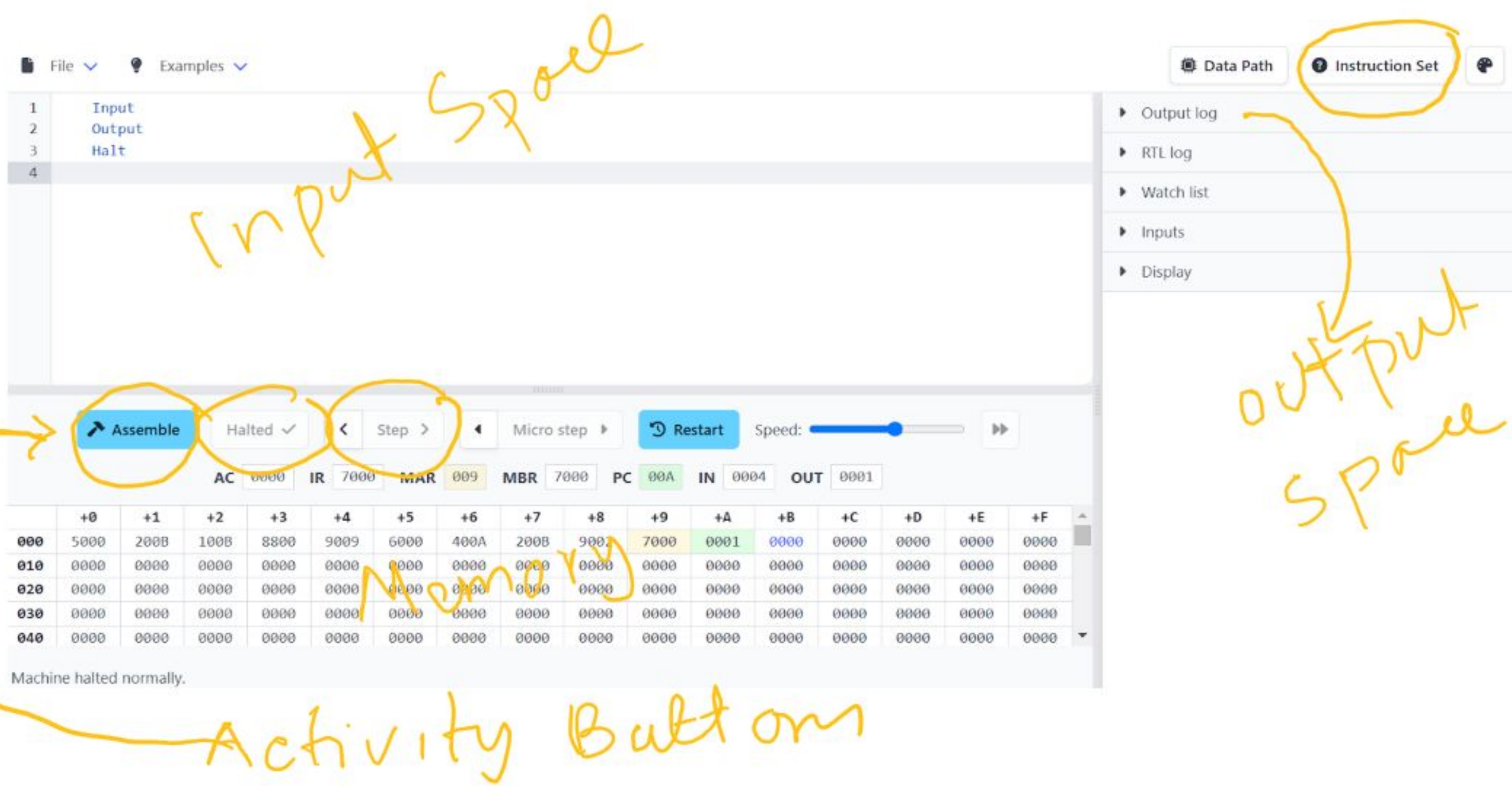
MARIE Architecture



The MARIE Architecture



The MARIE Simulator



Activity A:

Von Neumann to MARIE Architecture

The MARIE Simulator

Using the MARIE simulator explore different components (CPU, Memory and Address/Data Bus) and answer the following questions:

1. Name the components of the CPU. You can use the button “Data Path” to switch between viewing modes.

2. There are a number of registers in MARIE. How many are there? What is the number of bits each of them can hold?

3. In MARIE memory, we can store instructions and data of fixed lengths. What is that length? How many bits long instructions or data can we store in MARIE memory?

4. Explore the memory display in the simulator and find out how they are arranged for us to view the entire memory content. Any comments? Why is it arranged in 16 blocks (a word) in a row? How many memory locations are all together? Is there any relationship between this number and the length of the address bus, i.e.number bits in the address?

5. Any CPU needs a certain number of steps (micro-step) to execute any instruction. What is the maximum number of micro-steps MARIE can perform? Check the Control Unit step count for this..

PollEv Question 1

Q: Can you find out the processor (CPU) used in your laptop?

Hint: Windows PC - use Alt+Ctrl+Del to open Task Manager. Apple Mac use Applications -> Utilities

https://PollEv.com/safiuddin051

Activity B:

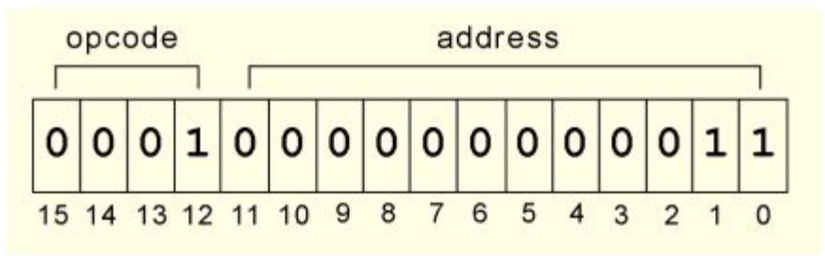
Assembly to Machine Language in MARIE

MARIE Programming

- No CPU can execute a high level program like this:

```
import sys
name = sys.argv[1]
print 'Hello, ' + name + '!'
```

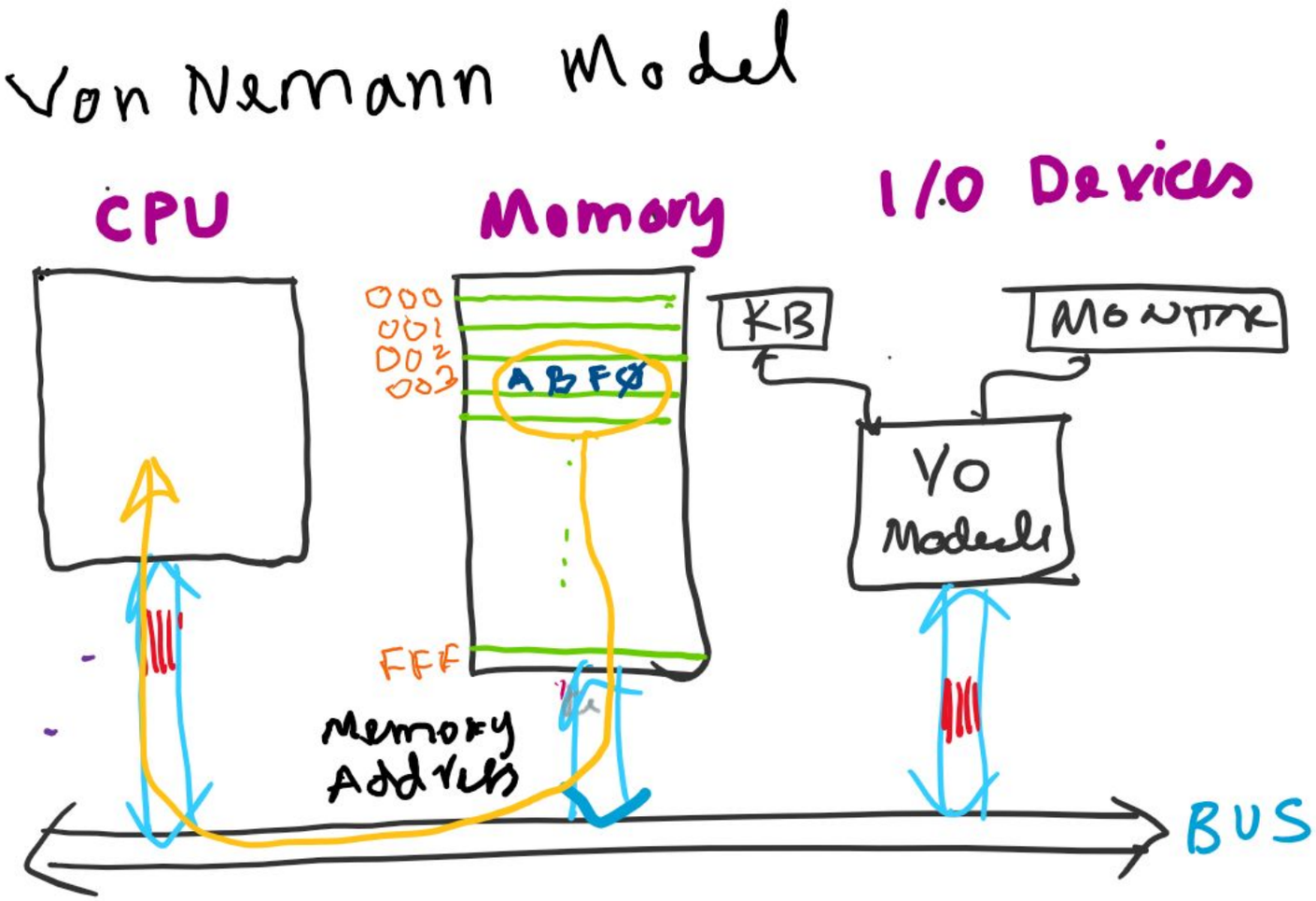
- A CPU (e.g. MARIE) can execute code shown below:



MARIE Programming

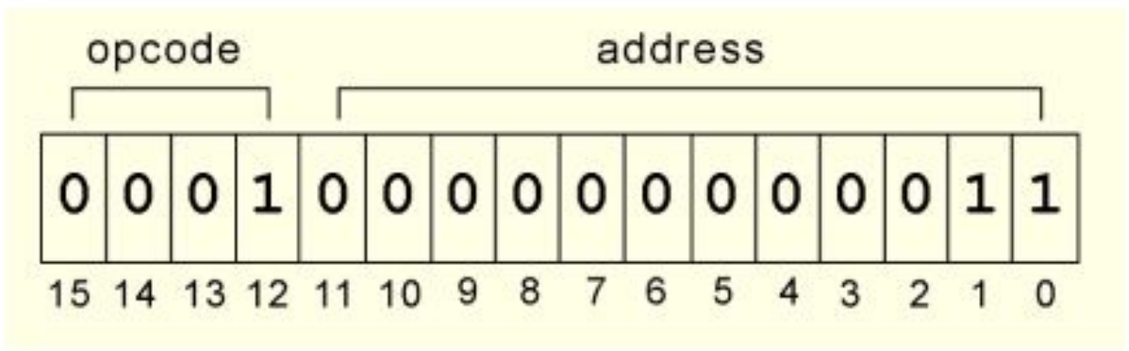
- Not writing programs using “0 and 1” for a CPU to execute.
- We use Assembly Language to write programs.
  - LOAD 003

    Get the memory content from location 003 and store it in a register in CPU.



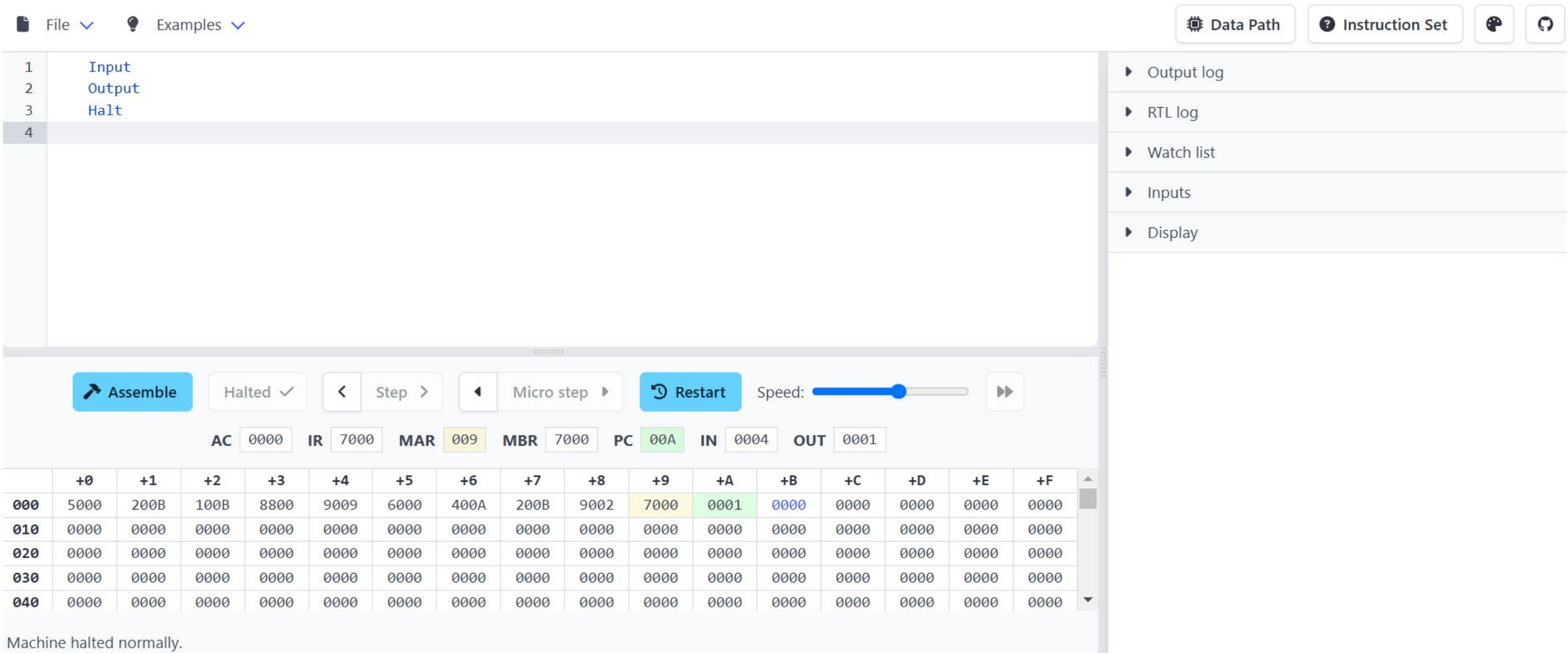
MARIE Programming

- Not writing programs using “0 and 1” for a CPU to execute.
- We use Assembly Language to write programs.
  - LOAD 003
    Get the memory content from location 003 and store it in a register in CPU.
- An Assembler will convert it to a machine code as below:



MARIE Programming

- How to write a program to execute in MARIE?
- What are the commands available?
- How to write an instruction?



MARIE Programming

The MARIE assembler translates the assembly language program into machine code.

1. Can you find the Assembler (button) in the MARIE simulator? Also, locate the buttons to run or step through your assembly language program.

2. Is there any command to take data from KB or send data to display? Can we do mathematical operations in MARIE? How about multiplication and division?

3. When you write an assembly language program, how do you inform the CPU that you are at the end of your task? Do you think it is very important to inform the CPU?

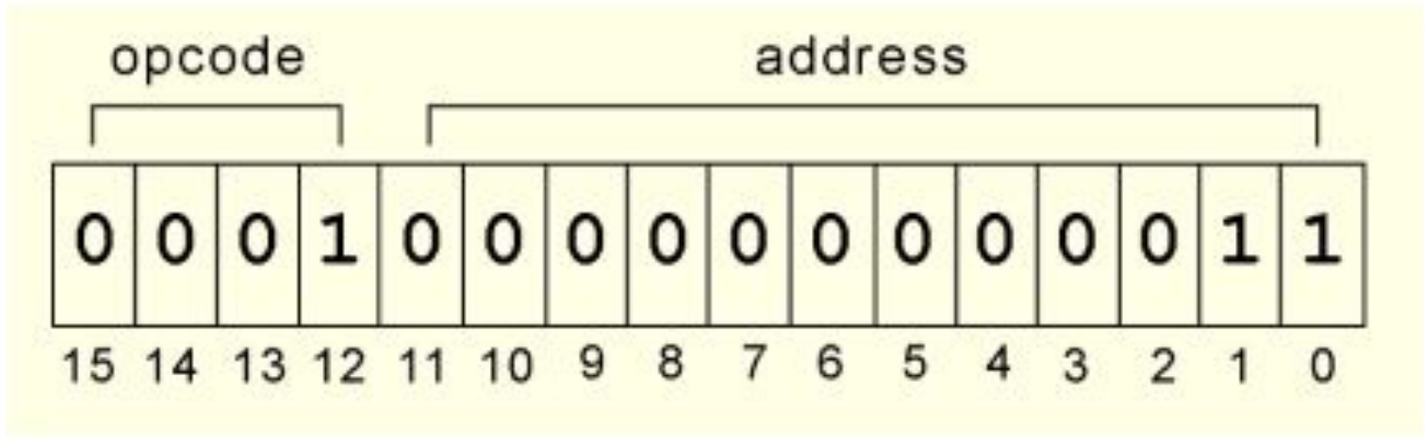
4. Write a small program to input a number from KB and display it on the display window.

Activity C:

A Simple MARIE Instruction

MARIE Instructions

- A MARIE Instruction: LOAD 003



- Find the opcode. What is it?
  - -> MARIE Command in machine code.
- Find the address?
  - it is one of the memory locations of MARIE
- Can you find the MARIE Instruction in machine code ?

MARIE Instructions

Using the MARIE Instruction Set found in the simulator:

1. How many instructions are there in the MARIE ISA? Can you name them? Do the names make any indication of their actions? You can explore a few simple instructions and their functional details.

2. Using the MARIE instruction set format shown in fig: 5, find out the opcodes for at least 6 MARIE instructions.

3. From the instructions format above, we can see that in an instruction, the address field is 12-bits long. What does it imply in computer design? Any relationship to the total memory addresses a MARIE computer can have?

4. Assume, you have a computer with opcode using 8 bits and the address field is 24 bits long. How many instructions are there in this computer's ISA? What are the total memory addresses this computer can have or can handle?

PollEv Question 1

You have a computer with opcode using 8 bits and the address field is 24 bits long.

How many instructions are there in this computer's ISA?

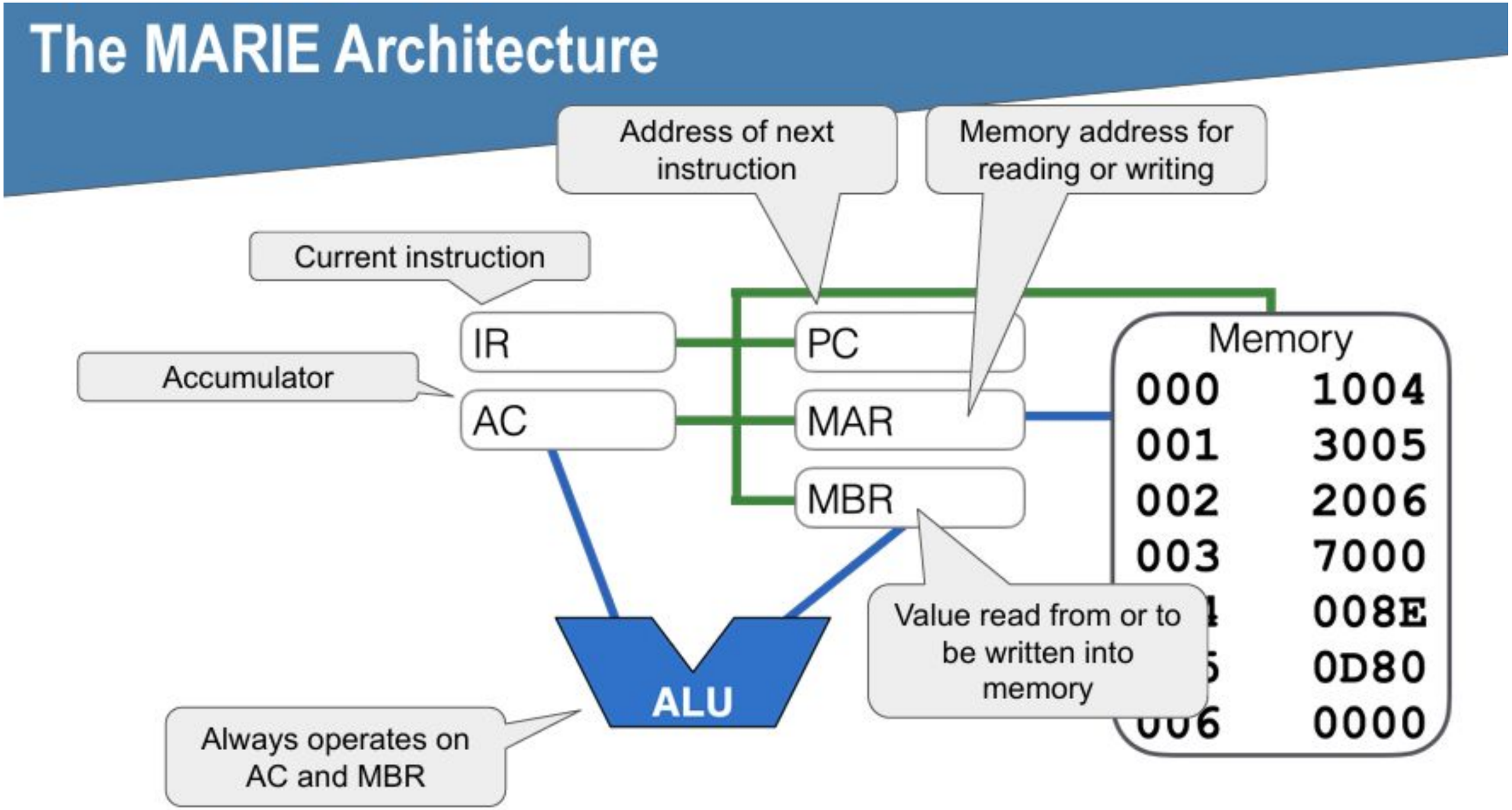
What are the total memory addresses this computer can have or can handle?

https://PollEv.com/safiuddin051

Activity D:

The Control Unit in MARIE Architecture

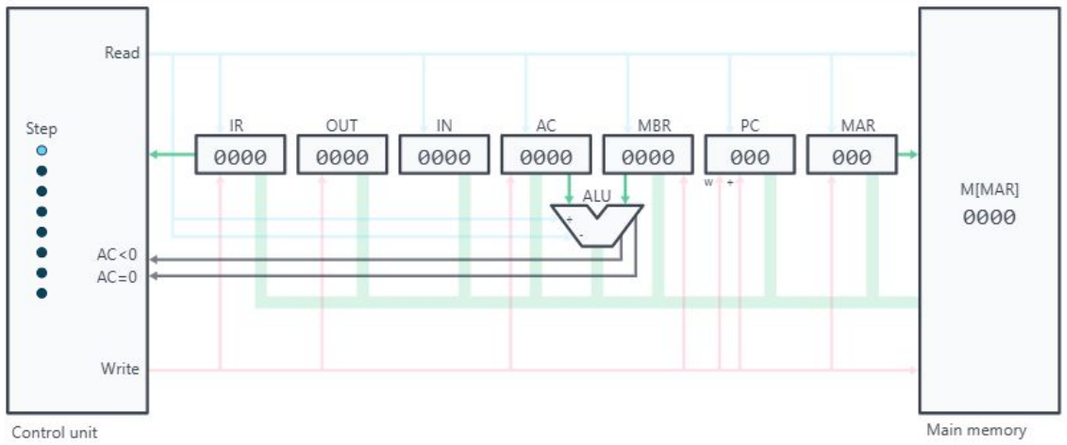
The Control Unit & RTL



The Control Unit & RTL

File Examples Data Path Instruction Set

1



Output log

RTL log

(log empty)

Watch list

Inputs

Display

Assemble Run Step Micro step Restart Speed:

AC 0000 IR 0000 MAR 000 MBR 0000 PC 000 IN 0000 OUT 0000

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

The Control Unit & RTL

Answer to the following questions.

1. Use the MARIE simulator to find the RTL log window and investigate the micro-operations using the "micro-steps " button. You may use a simple command like "Input".
2. Can you identify the RTL steps for (i) Fetch the (next) instruction from memory, (ii) Decode the instruction, (iii) Execute the instruction from your RTL log?
3. Investigate RTL log for the following program:

```
//—----------------------
Load      myData
Subt          one
Store         myData
Halt
myData, DEC 5
One,      DEC 1
//—----------------------
```

The Control Unit & RTL

Answer to the following questions.

1. Switch your MARIE simulator view to “Data Path” and observe the data movement for the “Input” command. Check the CU “step” lights changing from black to blue for every RTL movement. Each step counts one CPU clock pulse. Can you count the number of clock pulses (steps) needed for the “Input” command?

2. Check the number of steps/pulses for other commands as well. Are they the same? Do all the commands/instructions require the same amount of clock pulses?

End of the Workshop Tasks

Have a nice week ahead.

See You all Next Week