Activity 8: Automation
Recorder's Report

Manager: Reader:

Recorder: Driver:

Date: Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with ) below.

a) Model 1, Question #13

b) Model 2, Question #18

Activity 8: Automation

We have looked at basic logic gates and combined them into a circuit that can do 8-bit addition. Now we look at a programmable computer.

Content Learning Objectives

After completing this activity, students should be able to:

- given a simple instruction set, read, and write a program using machine code.

Process Skill Goals

During the activity, students should make progress toward:

- No additional process skills.

Sources

Model 1 Infinite Addition

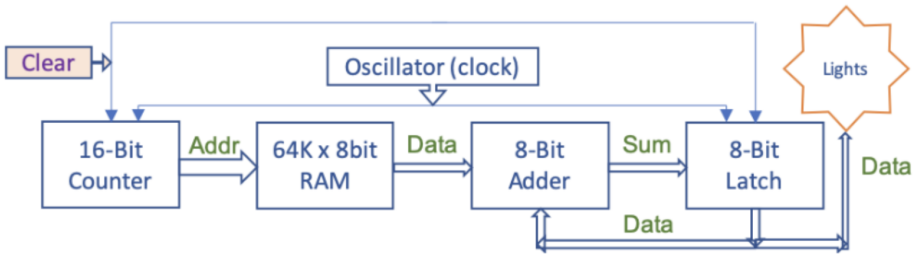

Figure 1: Simple Addition

With digital logic circuits we can build various components:

- An oscillator with output that switches rapidly between 0 and 1;
- A counter with 16-bit output that can be reset to all zeros and will increment on each "clock tick" (the oscillator has a change from 0 to 1);
- Addressable memory that outputs an 8-bit value based on a 16-bit address (a separate control panel with various switches is used to manipulate memory and is not shown);
- An adder that takes as input two 8-bit values and gives as output an 8-bit sum (carry in and out is ignored for now);
- A latch that captures an input value at a clock tick (and can be reset); and,
- Lights to show the current latch value.

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (25 min) Start time:

1. What is the size of the data path out of RAM?

2. What are the sizes of the data paths into and out of the Adder and Latch?

3. What is the size (number of bits) of the address path into RAM?

4. The range of values for a decimal digit is [0, 9]. What is the range of values for a hexadecimal digit?

5. How many bits are required to represent the full range of a single hexadecimal digit? (Hint: not 16!)

6. How many hexadecimal digits can be represented on the address path into RAM?

7. What happens to the Counter when you close the "Clear" switch?

8. What happens to the Latch when you close the "Clear" switch?

9. What happens to the Counter on a 0-to-1 clock tick?

10. What happens to the Latch on a 1-to-0 clock tick?

11. Describe the value on the data line coming out of RAM when the value on the address line into RAM is all zero bits (you might not be able to give an exact value).

12. Describe the value of the Counter

- after a clear,
- after one clock tick,
- and after N clock ticks.

13. Describe the value of the Latch

- after a clear,
- after one clock tick (hint: probably not 0 or 1),
- And after N clock ticks (hint: probably not N).

14. What is the maximum value of the Counter in hexadecimal (the convention for addresses)? (Hint: see question 6.)

15. If the Counter is at the maximum value, what is the value after a clock tick?

16. Does circuit ever stop? If, so, how? If not, why not?

Model 2 A Programmable Computer

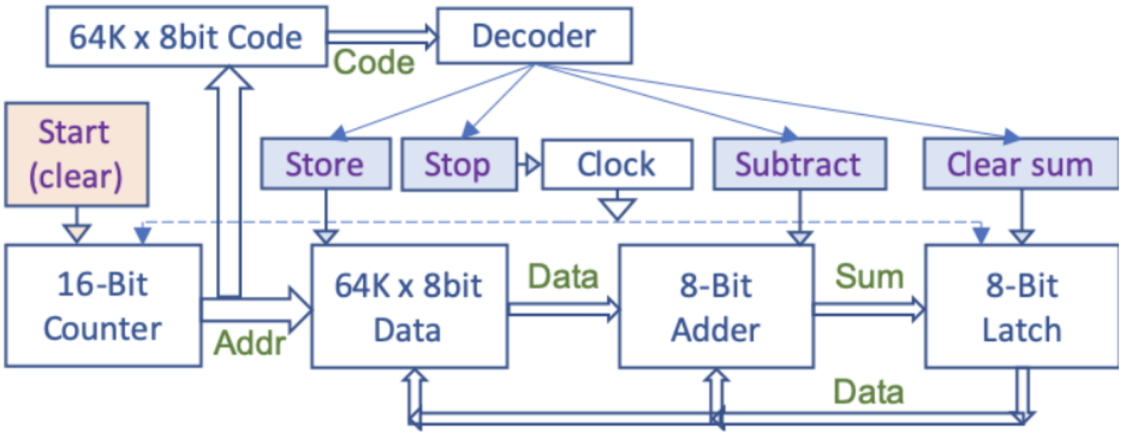

Figure 2: Programmable Computer

Early computers were hard-wired to perform certain tasks, and this worked okay for well-defined tasks such as calculating logarithms or trigonometry functions. But building a new circuit for each new task is cumbersome and computer engineers looked for a more flexible approach.
The circuit shown above demonstrates some enhancements. It has:

- a new memory module for code (having code in a separate module is less common now),
- a decoder to interpret the instructions and set the appropriate control lines,
- a control line to specify whether to read or write (store) memory,
- a control line to stop the clock,
- a control line to subtract instead of add, and
- a separate control line to clear the latch (independent of the memory counter).

Refer to Model 2 above as your team develops consensus answers to the questions below.

Questions (25 min) Start time:

17. The following table shows the five instructions supported by our computer. Convert the instruction codes from hexadecimal to binary and then identify what each bit means.

Instruction	Hexadecimal Code	Binary Code
Add (data to latch)	0x00	0000 0000
Clear (latch)	0x01	
Subtract (data from latch)	0x02	
Stop (the clock)	0x04	
Store (latch to data)	0x08	

Bit	3	2	1	0
Meaning				Clear

Note that bit 0 is the least significant (right-most) bit.

18. Describe the behavior of the decoder. How does it translate instruction codes to controls? (Hint: compare the binary code bits with the control lines coming from the decoder.)

19. What is the relationship between a code address and a data address?

20. If the instruction at memory address 0x100 in the memory module labeled Code is a Store, at what address in Data will it place a value?

21. Using the appropriate control panels, a programmer loaded five bytes of code and two bytes of data into memory. The remaining portions of both memory are undefined. Complete the "Meaning" column with the name of the instruction then "execute" the code showing what value is in the appropriate memory location and the latch after each instruction. Unspecified values should be indicated with a question mark (?).

Address	Code	Meaning	Data	Latch
0x00	0x01			
0x01	0x00		5	
0x02	0x02		3	
0x04	0x08			
0x08	0x04			

22. Write a program to calculate and store (11 + 6 – 9) and then calculate and store (40 + 17). You need not use all memory.

Address	Code	Meaning	Data	Latch
0x00				
0x01				
0x02				
0x03				
0x04				
0x05				
0x06				
0x07				
0x08				
0x09				
0x0A				
0x0B				