Activity 9: Addressable Memory
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 #5

b) Model 2, Question #11

c) Model 3, Question #20

Activity 9: Addressable Memory

We have looked a simple programmable computer. Now we add hardware and instructions for storing the code and the data in the same memory module.

Content Learning Objectives

After completing this activity, students should be able to:

- Do 16-bit math with 8-bit memory.

Process Skill Goals

During the activity, students should make progress toward:

- No additional process skills.

Sources

Model 1 Load Instruction

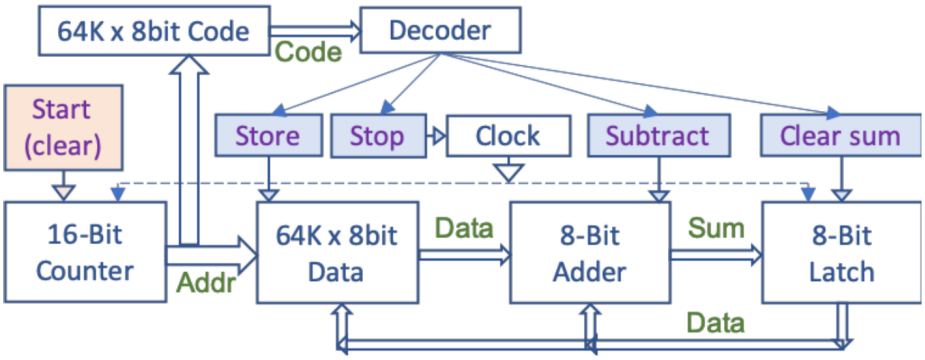

Figure 1: Programmable Computer

Instruction	Hexadecimal Code	Binary Code
Add (data to latch)	0x00	0000 0000
Clear (latch)	0x01	0000 0001
Subtract (data from latch)	0x02	0000 0010
Stop (the clock)	0x04	0000 0100
Store (latch to data)	0x08	0000 1000

Figure 1 (copied from the previous lesson) shows a simple programmable computer with the following characteristics:

- A counter with 16-bit output that will increment on each clock tick (it can be reset to zero with the Start control line),
- Separate addressable memory for code and data that each output an 8-bit value based on the same 16-bit address (the value of the data memory can be set to the current data latch value by the Store control line),
- An 8-bit adder that calculates the sum (or difference based on the Subtract control line) of the latch and the value from memory,
- A latch that captures the result of the adder on each clock tick (or is cleared based on the Clear sum control line), and
- A decoder that takes each 8-bit instruction and sets the control lines appropriately.

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

Questions (15 min) Start time:

1. In Figure 1, the data path into the latch comes from what component?

2. In Figure 1, what two instructions are required to get the value of a single memory location into the latch? (Hint: what would you add to a value to get the same value?)

Figure 2 adds a new control line and a new component to our computer.

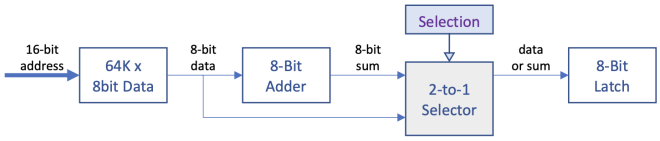

Figure 2: Load Instruction

3. What are the new elements?

A 2-to-1 Selector will pass on one of two inputs based on a control line. In this configuration, if the control line is 0, then the value from the adder will be passed to the latch. If the control line is 1, then the value from memory will be passed to the latch.

4. How does this circuit simplify loading a value from memory into the latch?

5. Suggest a name and a code for this instruction. (Hint: see the name of Figure 2 and the pattern of hexadecimal codes used for previous instructions.)

Model 2 Addressable Memory

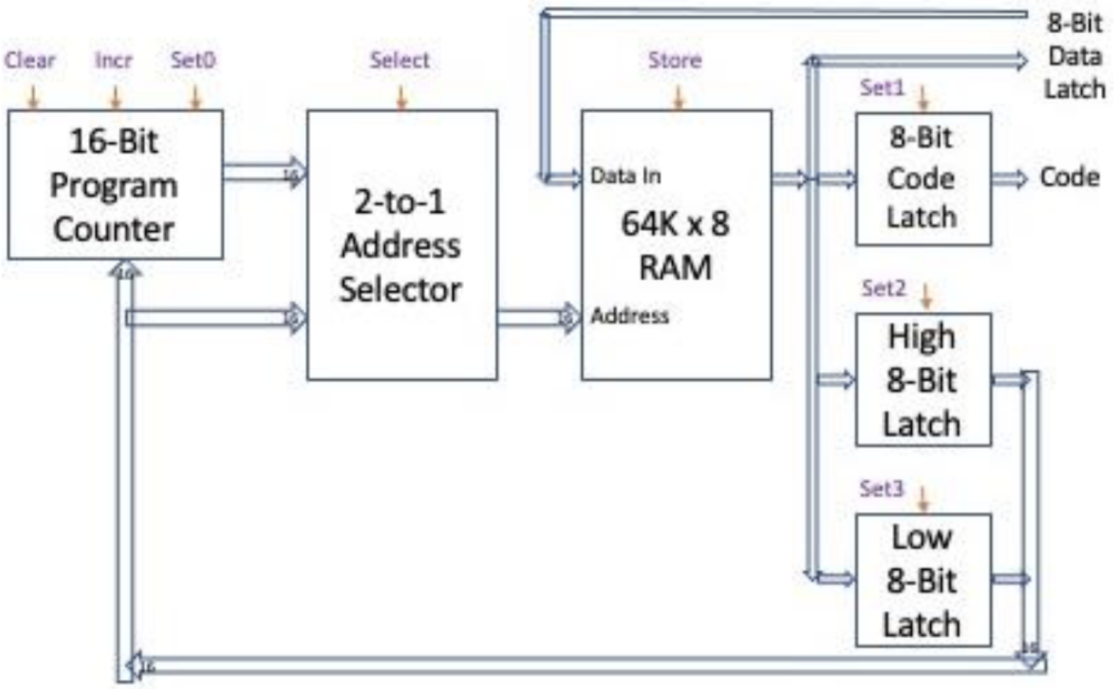

Figure 3: Addressable Memory

Figure 3 shows the final major enhancement to the computer we are building.

- To the 16-Bit Program Counter (PC) we have added a Set control line that allows us to set the next instruction address to a 16-bit value obtained from the bottom two 8-bit (address) latches.

- The memory address for read/write is now specified by either the PC or the bottom two (address) latches based on a Select control line sent to the 2-1 Address Selector.

- Instead of having separate memory for code and data, we now share the memory (reducing cost and increasing flexibility).

- Instead of having a sequence of 8-bit instruction codes, we now have instructions made of an 8-bit opcode (operation code) followed by a 16-bit address (operand location). These values are read into the three latches shown in three consecutive clock ticks.

- Not shown are the instruction Decoder (to the right of the arrow labeled "Code"), the 8-Bit Adder and the 8-Bit Data Latch (to the right of the arrows labeled "8-Bit Data"), and the control lines coming out from the Decoder (the endpoints are shown with the affected components).

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

Questions (15 min) **Start time:**

6. What was the previous instruction size? What is the new instruction size? By what factor has the overall size of the instructions increased?

7. Assuming that we can still read only one byte per clock cycle and that the clock speed is the same, what is the impact on speed by the instruction size changes?

8. In Figure 1, what happened to the counter on each clock tick?

9. In Figure 1, what can be done to change the value of the counter to something other than the next integer? What value(s) (if any) can be assigned to the program counter?

10. What new control line is added to the program counter in Figure 3? What does it do?

11. What is the impact of allowing a "Set" instruction ("Jump") for the Program Counter?

12. In Figure 1, what determines which address in memory will be provided to the adder?

13. What new component is added in Figure 3 between the counter and the memory?

14. In Figure 3, what determines which address in memory will be provided as output?

15. What is the impact of allowing a load/add/store instruction to specify an address?

16. A variation of the Jump command is a Conditional Jump where the change to the PC happens only on if some specified condition exists. What conditions would be interesting?

Model 3 Multi-Byte Numbers

Refer to Model 3 above as your team develops consensus answers to the questions below.

Questions (20 min) **Start time:**

17. Consider the expression with two unsigned integers: ($1100\ 0001_2 + 0100\ 0011_2$).

- What is the binary sum?
- How many bits are required for the answer?

18. Consider the 8-Bit Adder shown in Figure 4.

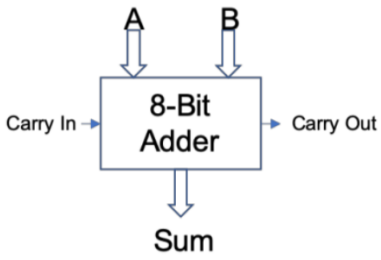

Figure 4: 8-Bit Adder

- What are the data path widths (number of bits) for A, B, and Sum?
- What are the data path widths for Carry In and Carry Out?
- How is this 8-Bit Adder different from the one shown in Figure 1?

- If you were to use this 8-Bit Adder in the circuit shown in Figure 1, what value would you need for Carry In? What circuit component (from an earlier lesson) would you attach to the Carry In line to do proper math?

- What component would you add to the Carry Out line to ignore the overflow bit?

- If the carry lines are ignored for simple 8-bit addition, why do they exist?

19. Consider the expression with two unsigned hexadecimal integers: $(2345_{16} + 3456_{16})$ 

- How many bits are required for each number?

- How many bits are required to represent the sum?

- Describe how you would use an 8-Bit Adder to perform this operation.

20. Consider the expression with two unsigned hexadecimal integers: $(2385_{16} + 3496_{16})$.

- How would you use the Carry In line to add the low-order 8 bits $(85_{16} + 96_{16})$?

- How would you use the Carry In line to add the high-order 8 bits?

- Suggest a name for this new kind of Add command.

21. What does the circuit in Figure 1 use to hold intermediate 8-bit sums?

22. What has been added in Figure 5 compared to Figure 4?

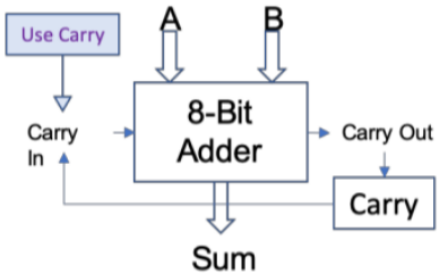

Figure 5: Add with Carry

23. If we used 0x20 as the code for the Add with Carry instruction, how would it differ from the Add instruction code?

24. If we attached bit 5 of the instruction decoder to the "Use Carry" control line, what logic gate(s) would provide the desired behavior?