**Score: \_\_\_\_\_**

**MA1 – A Microprogrammed Machine**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

**Introduction:**

Today’s class introduced the Machine Abstractions topic including several levels of abstraction at which we will view the machine itself (Physical Machine, Microprogram Machine, Machine Language Machine). The Knob & Switch computer, a visualization of a simple computer that allows you to experience these levels in a hands-on way was also described. Today’s activities start by having you work with the Knob & Switch computer at the Physical Machine and Microprogram Machine levels of abstraction. They then give you the chance to take a deeper look at the history of a few of the important early physical machines that we’ve heard of before (Colossus and ENIAC).

**The Knob & Switch Datapath:**

A datapath is the part of the computer’s central processing unit that performs computations. The *Register Bank* contains registers that provide a small amount of temporary storage for the data on which the computations will be performed. The *Arithmetic and Logic Unit (ALU)* contains the circuitry that performs the computations (e.g. addition and subtraction). The data moves from the Register Bank to the ALU and back to the Register Bank on *Data Busses*. A Data Bus is essentially a collection of wires, one for each bit, that move the data from place to place.

The Knob & Switch (K&S) computer provides a simulation of the *datapath* of small manually programmable machine. It has been designed to give you some insight into the way that modern computers perform computation. The calculations performed by the K&S Datapath are configured by setting knobs in the datapath to choose the registers containing the data to be used, the operation to be performed and the register into which the result will be stored.

You can open the K&S datapath simulation using the link below and use it to answer the following basic questions.

* <https://dickinson-comp256.github.io/Knob-And-Switch-Computer/datapath.html>

🔑 1. Consider the configuration of the K&S datapath shown below.

![A screenshot of a cell phone

Description automatically generated]()

Describe in a sentence or two what computation the K&S would perform in this configuration. Be sure to indicate what registers are used as inputs, what their values are, what operation is performed, the value of the output and in which register it will be stored.

🔑 2. Put the values 4321 and -1234 into registers. Then configure the K&S datapath to add those registers together and store the result in a different register. Give a screen shot of the K&S datapath after it has executed this operation.

🔑 3. Change the configuration of the K&S datapath from question #2 so that the result of the ALU operation is stored into the same register as one of the inputs. What happens to the value in that register?

**ALU Flags:**

You may have noticed that there are several check boxes in the ALU (Zero, Negative, Unsigned Overflow and Signed Overflow). These are *flags* that indicate properties of the result produced by the ALU when it performs an operation. For example, the Zero flag will be set (checked) if the result of the ALU operation is zero. The Negative flag is set if the ALU result is negative. The Unsigned Overflow flag is set if the operation produces overflow in 16-bit unsigned binary. The Signed Overflow flag is set if the operation produces overflow in 16-bit Two’s Complement.

4. For this question, configure the A Bus, B Bus, ALU and C Bus knobs of K&S datapath as shown below.

Graphical user interface, diagram

Description automatically generated

a. Using the above configuration, give values for R0 and R1 that would cause the zero flag to be set. Be sure to test your answer by executing the datapath. It is okay if other flags are also set as long as the zero flag is set.

b. Give values for R0 and R1 that would cause the negative flag to be set. Be sure to test your answer by executing the datapath. It is okay if other flags are also set as long as the negative flag is set.

🏆 c. Give values for R0 and R1 that would cause the Signed Overflow flag to be set but will not cause the Unsigned Overflow flag to be set. Hint: Thinking in two’s complement binary is the best way to solve this problem.

Show your values in both base 10 and in binary and explain in a few sentences why they set the Signed Overflow flag and do not set the Unsigned Overflow flag. Be sure to test your answer by executing the datapath.

**Bitwise Operations:**

You will have noticed that the K&S ALU can perform four different operations (+, |, & and -). The + and - perform addition and subtraction using the binary representations as we have learned about in class.

In addition to addition and subtraction computer ALUs also perform a range of *bitwise operations*. In a bitwise operation, the bits in the corresponding locations of the inputs are operated on in pairs. For example, consider the 4-bit examples of the *bitwise OR (|)* and *bitwise AND (&)* operations shown below:

Bitwise OR (|) Bitwise AND (&)

1011 A 1011 A

| 0010 B & 0010 B

1011 A|B 0010 A&B

Notice that each bit of the result is obtained by applying the operation (OR or AND) to the corresponding bits of A and B. For example, the yellow highlighted bit of the result in the bitwise OR example is a 1 because the two bits above it are 1 (A) and 0 (B) and 1 OR 0 is 1. Similarly, the yellow highlighted bit of the result in the bitwise AND example is a 0 because the two bits above it are 1 (A) and 0 (B) and the 1 AND 0 is 0. The other 3 bits of each result are computed in the same way, using the bits directly above them.

5. This question gives you a little practice with these bitwise operators.

a. Give the result of the following 8-bit bitwise operations (you can use the K&S in Base 2 mode to check your answer, just enter leading 0’s to get to 16 bits):

1010 1101 0101 0110

& 1100 1001 | 1100 0101

🏆 b. Try the bitwise & or a bitwise | operation when A and B are the same number (e.g. A=257, B=257). Try these operations on a few more values when A and B are the same number. Briefly explain why these operations produce the result that they do.

**Knob & Switch with Main Memory:**

The K&S with Main Memory adds a main memory (i.e. a RAM) to the machine and also adds some switches to the datapath. The main memory provides a larger storage area that can hold more data, and as we will eventually see, program instructions. The new switches have been added to make it possible to move data between the main memory and the registers.

You can open the K&S with Main Memory using the link below.

* <https://dickinson-comp256.github.io/Knob-And-Switch-Computer/dpandmem.html>

🔑 6. Enter a non-zero value into main memory location 510. Then configure the K&S datapath so that the value you entered will be transferred into register R2. In doing so, be sure that the ALU is disconnected from the C bus so that its output does not *collide* with the value coming in from memory.

Execute the datapath to be sure your configuration works. Include a screen shot of the main memory and data path after execution as your answer for this question.

🏆 7. Enter a non-zero value into each of the registers (R0-R3). Then configure the datapath so that the value that is in R3 is transferred to main memory address 710. The execution should store the value from R3 into memory but it may not change the value in any other register. The execution should also work for any value that is placed into R3. Hint: Question #5b provides a way to move the value in register through the ALU without changing it!

Execute the datapath to be sure your configuration works. Include a screen shot of the main memory and data path after execution as your answer for this question.

**Memory Addresses:**

🔑 8. Every location in the main memory has a *memory address*, given by the number beside the cell. The K&S will show us these addresses in base 10. But we know that they, like everything else in the computer, must be expressed in binary. Because all of the addresses are non-negative, we use unsigned binary values for these addresses.

a. What is the address of the first (top most) memory location? Give your answer in both 5-bit unsigned binary and base 10.

b. What is the address of the last (bottom most) memory location? Give your answer in both 5-bit unsigned binary and base 10.

c. The K&S main memory had 32 memory locations. Given this, briefly explain why the K&S uses 5 bits for its memory addresses.

d. If the designer of the K&S had wanted to have 64 different memory locations (instead of the 32 it has now), how many bits would have been needed for the memory addresses?

e. If the K&S had been designed to use 8 bits for its memory addresses, how many memory locations could there be?

**K&S Microinstructions:**

The Microprogrammable K&S adds the ability to program the machine using microinstructions. Each *microinstruction* is a pattern of 1’s and 0’s that represent the configuration of the machine’s knobs and switches. Stated another way, a micro-instruction is an abstraction for a particular configuration of the machine. The information that the abstraction lets you focus on is what the individual bits of the microinstruction cause to happen. The abstraction lets you forget that how that those things happen. For example, you need to know that setting the A Bus value to 10 will send R2 onto the A Bus. But once you know that, you no longer need to be aware of the fact that the 10 is actually positioning the A Bus knob to point to R2. These exercises introduce you to using the microinstruction abstraction for the K&S.

You can open the Microprogrammable K&S using the link below.

* <https://dickinson-comp256.github.io/Knob-And-Switch-Computer/micromachine.html>

Each microinstruction for the K&S has six fields (e.g. A Addr, B Add, ALU op, etc) as shown below:

A screenshot of a cell phone

Description automatically generated

Each field will contain a pattern of bits (1’s and 0’s) that specify the configuration of the corresponding element of the K&S data path. More specifically:

* Each Knob (A Addr, B Addr, ALU Op, C Addr) has 4 positions. So, two bits will be used to represent its position as Up (00), Right (01), Down (10) and Left (11). For example, the B Addr field for the knob shown below would be encoded as 10.

A picture containing drawing

Description automatically generated

* The Switch Pos field uses 4-bits to indicate the position of each of the four switches in the data path as Open (0) or Closed (1). The switches are represented in counterclockwise order starting at the top and going counterclockwise. For example, the switch positions below would be encoded into a micro instruction as 1010.



* The RW Addr field is 5 bits and gives the unsigned binary value of the memory address of the location that will be involved in the instruction (i.e. the location where the RW button is selected). For example, if memory address 1810 is being used then the RW Addr field will be 100102.

🔑 9. Consider the micro instruction shown below:

A screenshot of a cell phone

Description automatically generated

Using the screen shot of the K&S shown below, draw in the knob and switch positions and indicate the RW memory location corresponding to the above micro instruction.

Graphical user interface, application

Description automatically generated

🔑 10. Consider the K&S machine configuration shown below:

A screenshot of a cell phone

Description automatically generated

Fill in the table below with the bits of the micro-instruction that will produce the machine configuration shown above.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **A Addr** | **B Addr** | **ALU Op** | **Switch Pos** | **C Addr** | **RW Addr** |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**K&S Microprograms:**

When you look at the microprogram memory in the K&S you can see there are rows for up to five microinstructions. The machine will execute these micro-instructions one after the other from top to bottom. Thus, you can small write programs for the K&S to perform multiple step calculations using microinstructions. These programs written in microinstructions are called *microprograms*.

🏆 11. Write a microprogram for the K&S that will perform the following computation:

* Add the value in main memory location 5 to the value in main memory location 20 and store the result in main memory location 16. Hint: You’ll need to use multiple microinstructions that move the values to be added into registers, compute the result and then move the result back to main memory.

Be sure to enter values into the K&S memory at locations 5 and 20 and then run your program to test that it works. Your solution should be general. It should not rely on having any specific values in the registers R0-R3 when the program starts. It should also work for any values that are entered into main memory locations 5 and 20.

Place a screen shot of the microprogram memory containing your microprogram in the box below as your answer to this question.

**Connecting the Dots:**

🏆 12. Explain in a few sentences how the circuits that you built in Lab #3 are related to ALU in the Knob and Switch computer.

🔑 13. The ALU in the K&S machine you have been working with can perform 4 different operations (+, |, &, -). The ALU in a real machine can perform many more operations including multiplication, division, operations on floating point numbers and even sound and graphics.

While these more complex ALU will contain more complex circuits, as an abstraction they still behave the same way. A pattern of bits is used to select which operation they perform (e.g. like the pattern 11 causes the K&S to do a subtraction). Fill in the table below with the number of bits that would be needed to specify the operation for an ALU that can perform the given number of different operations.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Number of Different ALU Operations | Number of Bits Needed in ALU Op |  |
|  | 4 | **2** |  |
|  | 8 |  |  |
|  | 16 |  |  |
|  | 32 |  |  |
|  | 128 |  |  |
|  |  |  |  |

**Let’s Add a Little History for Perspective:**

Many of the world’s most important inventions are an almost inevitable product of the their moment in time. For each of these inventions there were many different people in many different places all working at solving similar problems using similar ideas. Thus, it can often be hard to say who was the inventor of X (e.g. steam engine, automobile, lightbulb, etc.) or which was the first X. Computers are no different.

14. Using your favorite search engine identify at least four candidates for *the world’s first electronic computer*. For each machine identify the year it was created, the primary inventors and the location where it was constructed.

15. Watch the video: *1946 ENIAC Computer History Remastered* from the Computer History Archives Project and answer the questions below:

* <https://www.youtube.com/watch?v=bGk9W65vXNA> (9:37)

a. What does ENIAC stand for?

b. How was the ENIAC programmed?

c. What was the original problem that ENIAC was designed to solve?Top of Form

d. What were a few of the later computers designed by ENIACs engineers?

e. What were a few of the commercial computing companies that were created by those that created the ENIAC?

Though, not exactly the same, working with the first version of the K&S that you used will have given you a little bit of a feel for how machines like the ENIAC were programmed. In the ENIAC, there would have been many simple computational units similar to the K&S datapath. Each one would have been configured to perform some operation. These units would have then been connected together to perform a sequence of operations (i.e. a program).

16. **Optional**: The ENIAC is one of the United States’ contenders for the world’s first electronic computer. May other countries also have contenders. You can watch the video *Bottom of FormColossus: The World's First Electronic Computer* from The Open University to learn about a contender from the United Kingdom:

* <https://www.youtube.com/watch?v=EdxBO9jfU8k> (5:32)

Use this video to answer the questions below.

a. What was the problem that the Colossus was designed to solve?

b. How was the Colossus programmed?

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.