**Preliminary work**

**EE 447: Lab #2**

Parallel Input/Output and Keyboard Interface

**Berkay İPEK**

**2304814 – Sec.2**

**Question 1)**A picture containing text

Description automatically generated**Question 2)**Table

Description automatically generated with medium confidenceA picture containing chart

Description automatically generated

**Question 3)**

1. **How can you detect whether any key is pressed?**

The answer is between line 25-35 in the main function (programming\_directives.s file). If there is a push in any keys. The output data will be changed accordingly. I connected L1-4 to PB0-3, R1-4 to PB 4-7. I set PB0-3 as outputs and PB4-7 as inputs. Also, I gave high voltages to these outputs one by one in a loop. Therefore, when there is pushed button, the input pins (PB4-7) should change. If there is no change in PB4-7, i.e. R2 is 0x00, that means there is no pushed button. If there is a pushed in a button, there will be a connection between L1-4 and R1-4. Therefore, PB4-7 values will be changed, and input data will be no longer zero. (It can be 0x10, 0x20, 0x40 or 0x80) I implemented this algorithm by comparing input data with possible data values(0x10, 0x20, 0x40, 0x80). If there is no matching , then it will go into the start point in a loop without calling OutChar function. If there is a matching, then it will go into the line COL1, COL2, COL3, or COL4.

Diagram

Description automatically generated with medium confidence

1. **How can you detect whether any key is released?**

The answer is in line 38-40, 54-56, 70-72, 86-88 in the main function (programming\_directives.s file). If there is a button pressed it will jump into COL1, COL2, COL3, COL4. After this jump, input will be read again, and it will be compared with previous read data. If they are equal, it will go into COL1 again. If they are not equal, that means key is released.

1. **Assuming that you have detected that a key is pressed. Explain your algorithm to determine which one is pressed.**

I gave high voltages into L1, L2, L3, L4 respectively and check R1, R2, R3, R4 status. If there is an R pin with high voltages, that means there is a pressed key. To determine which one is pressed, I store data for which L pin and R pin are high. To explain algorithm better,

At time t=0, L1 is high. Checks any R is also high.

At time t=t, L2 is high. Checks any R is also high.

At time t=2t, L3 is high. Checks any R is also high.

At time t=3t, L4 is high. Checks any R is also high.

By checking the register that stores input pin data, we can determine key in which column is pressed. For example, if K4 is pressed, then input register will be 1000.0000(0x80) when L1 is high. If K6 is pressed, then input register will be 0010.0000(0x20) when L2 is high. This input register will determine the column for the pressed key. That’s why it will compare input register with

Input registers:

0000.0000 (0x00) -> there is no pushed button

0001.0000 (0x10) -> there is a pushed button in column 1

0010.0000 (0x20) -> there is a pushed button in column 2

0100.0000 (0x40) -> there is a pushed button in column 3

1000.0000 (0x80) -> there is a pushed button in column 4

1. **Discuss what can happen due to bouncing. How can you avoid bouncing effects?**

With bouncing effect, the result will not be acceptable. When we pushed a button, and if we do not put any delay, the result will be undetermined. It will do it for maybe lots of time since its value will be changed ‘1’ and ‘0’ very often. However, by putting little bit delay, bouncing effect can be omitted. After the first read in the code, it will go into DELAY100 subroutine. After that, it will read again and check if there is a change input data. If there is not any change it will do the rest of code. If there is, it will go into the first read part again. It can be seen in lines 17-23 in main function (programming\_directives.s file).

1. **Now, develop your overall end-to-end algorithm that outputs ID of the pressed key to the terminal window and draw its flow chart.**

**Diagram, engineering drawing

Description automatically generated**

1. **Implemented the developed algorithm in part-e by using assembly language**

**Table

Description automatically generated**

**A picture containing text

Description automatically generated**

**Table

Description automatically generated with medium confidence**

After running these files, I pushed keys from KEY1 to KEY16. The results are shown below.

**Graphical user interface, text, application, email

Description automatically generated**