EE309 Assignment 1, Sheel Shah, 19D070052

Overall flow of the program:

Every 25 ms T0 overflows , resulting in an interrupt. The handler for this interrupt sets T0 – associated variables to reproduce another interrupt after 25 ms and it then jumps to the FSM subroutine. In this subroutine, the current state is evaluated and based on this state a test is chosen to be performed. Now based on the result of this test, an action is done and then finally the state is updated.

If a valid key press is detected and confirmed, the code corresponding to the key is stored in a circular buffer, of which the starting address and index are maintained in separate variables.

Intricacies:

In the FSM routine, all significant variables (acc, psw, dph, dpl) are pushed to stack and then after the execution of the function, popped out in reverse order in order to not disturb the flow of the other (main) program that was running before the interrupt occurred.

The following table (derived in class using the state diagram) stores the set of actions/tests based on the state and result of test (stored in F0 of PSW).

```
Test_Tab:    DB 0, 1, 1, 1
Yes_Actions: DB 1, 2, 0, 0
No_Actions:  DB 0, 0, 0, 0
Yes_Next:    DB 1, 2, 2, 2
No_Next:     DB 0, 0, 3, 0
```

```
Test_Jmp:
AJMP AnyKey
AJMP TheKey
```

```
Action_Jmp:
AJMP DoNothing
AJMP FindKey
AJMP ReportKey
```

- Test_Tab indicates the number of test to be performed given the state. (if j is the ith number in Test_Tab, then jth test is performed when the current state is i)
- If the test resulted in a “YES”, then the Yes_Actions row is used to choose the next action. If the result was a “NO”, the the No_Actions one is used (in a similar fashion as in Test_Tabs).

- Similarly, based on the result of the test, the next state is decided via the Yes_Next and No_Next rows.
- The routine for each test/action is jumped to by storing AJMP calls to these routines at a specific destination, and now jumping to this destination + i*2 sends the PC to the ith routine. (i*2 as each ajmp instruction is 2 bytes long)

The following functions were implemented by me to complete the assignment:

- AnyKey: Check if any key is pressed
- TheKey: Check if the key stored previously (at memory address given by “KeyCode”) is still pressed
- DoNothing: Do nothing, just RET
- FindKey: Store in the address given by “KeyCode”, the value associated to the key pressed.
- ReportKey: Add this key to the circular buffer and shift the position of the buffer’s index appropriately