МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ

«МОСКОВСКИЙ ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ»



Институт Интеллектуальных Кибернетических Систем

Кафедра «Компьютерные системы и технологии»

Отчёт о лабораторной работе №4   
«Программная реализация конечного автомата»  
по курсу «Микропроцессорные устройства и системы»

Студент группы Б20-503 Коломенский В. Г. / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

Москва 2023

**Оглавление**

1. Введение1

2. Реализация программы2

3. Тестирование программы6

4. Список литературы и ссылки8

1. Введение

По заданию лабораторной работы необходимо разработать автомат, который распознает заданную последовательность одинарных переключений линий порта р0.

Условие представлено на рисунке 1.

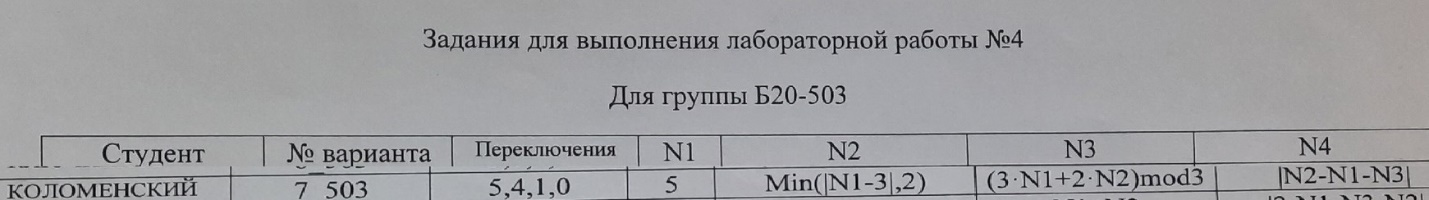


Рисунок 1. – Условие варианта 7\_503

N1, N2, N3, N4 – определяют количество допустимых ошибок при вводе очередного переключения. При определении допустимого количества ошибок начиная со 2-го переключения учитывается количество ошибок, допущенных при вводе предшествующих переключений.

При нарушении условий по допустимым ошибкам происходит блокировка работы системы, а в порт р1 выводится код «ааh».

Если распознавание переключения выполнено корректно с допустимым количеством ошибок ввода на каждом шаге, то в порт р1 выводится код 55h, а в порт р2 выводится hex номер варианта задания.

2.Реализация программы

Реализация программы представлена в приложении 1 ниже.

Приложение 1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186 | ; P0 - input  ; P1 - state  ; P2 - variant if succes  ; P3 - number of misses at this time (to del)  ; P4 - maximum allowable number of misses at this time (to del)  ; R0 - previous key (P0) set  ; R1 - number of misses in the 1st state  ; R2 - number of misses in the 2nd state  ; R3 - number of misses in the 3rd state  ; R4 - number of misses in the 4th state  ; R5 - maximum allowable number of misses in current state (P1)  ; R6 - shows changed bit  ; R7 - register for calc (tmp register)  P4 EQU 0C0h ; TODEL  mov P0, #0ffh  mov P1, #01h  mov P2, #00h  mov R0, P0  mov R1, #00h  mov R2, #00h  mov R3, #00h  mov R4, #00h  mov R5, #05h  mov P3, R1 ; TODEL  mov P4, R5 ; TODEL  wait\_input:  mov A, P0  xrl A, R0  jnz process\_input  jmp wait\_input  process\_input:  mov R0, P0  mov R6, A  mov A, P1  cjne A, #01h, pi\_1  jmp state\_1  pi\_1:  cjne A, #02h, pi\_2  jmp state\_2  pi\_2:  cjne A, #03h, pi\_3  jmp state\_3  pi\_3:  cjne A, #04h, pi\_4  jmp state\_4  pi\_4:  jmp failed  state\_1:  mov A, R6  jb ACC.5, s1\_1  mov A, R5  clr C  subb A, R1  jnz s1\_0  jmp failed  s1\_0:  inc R1  mov P3, R1 ; TODEL  jmp wait\_input  s1\_1:  mov P1, #02h  mov A, R1  clr C  subb A, #03h  jc s1\_2  mov A, R1  clr C  subb A, #03h  mov R7, A  jmp s1\_3  s1\_2:  mov A, #03h  clr C  subb A, R1  mov R7, A  s1\_3:  mov A, R7  clr C  subb A, #02h  jc s1\_4  mov R5, #02h  jmp s1\_5  s1\_4:  mov A, R7  mov R5, A  s1\_5:  mov P3, R2 ; TODEL  mov P4, R5 ; TODEL  jmp wait\_input  state\_2:  mov A, R6  jb ACC.4, s2\_1  mov A, R5  clr C  subb A, R2  jz failed  inc R2  mov P3, R2 ; TODEL  jmp wait\_input  s2\_1:  mov P1, #03h  mov A, R1  mov B, #03h  mul AB  mov R7, A  mov A, R2  mov B, #02h  mul AB  add A, R7  mov B, #03h  div AB  mov R5, B  mov P3, R3 ; TODEL  mov P4, R5 ; TODEL  jmp wait\_input  state\_3:  mov A, R6  jb ACC.1, s3\_1  mov A, R5  clr C  subb A, R3  jz failed  inc R3  mov P3, R3 ; TODEL  jmp wait\_input  s3\_1:  mov P1, #04h  mov A, R1  add A, R3  mov R7, A  clr C  subb A, R2  jc s3\_2  mov A, R7  clr C  subb A, R2  mov R5, A  jmp s3\_3  s3\_2:  mov A, R2  clr C  subb A, R7  mov R5, A  s3\_3:  mov P3, R4 ; TODEL  mov P4, R5 ; TODEL  jmp wait\_input  state\_4:  mov A, R6  jb ACC.0, passed  mov A, R5  clr C  subb A, R4  jz failed  inc R4  mov P3, R4 ; TODEL  jmp wait\_input  failed:  mov P1, #0AAh  jmp $  passed:  mov P1, #55h  mov P2, #07h  jmp $  end |

3. Тестирование программы

Разработанная программа тестировалась на примерах, представленных на рисунке 2. В этих примерах проверяются все «интересные» случаи: блокировка автомата на каждом из возможных состояний, ввод правильной комбинации и т.п. Также проверяется корректность вычисления N2-N4. К примеру, для операции «модуль числа» проверяется 3 случая: значение под модулем больше нуля, равно нулю, меньше нуля. Подобные проверки делаются и для операции взятия минимального значения.

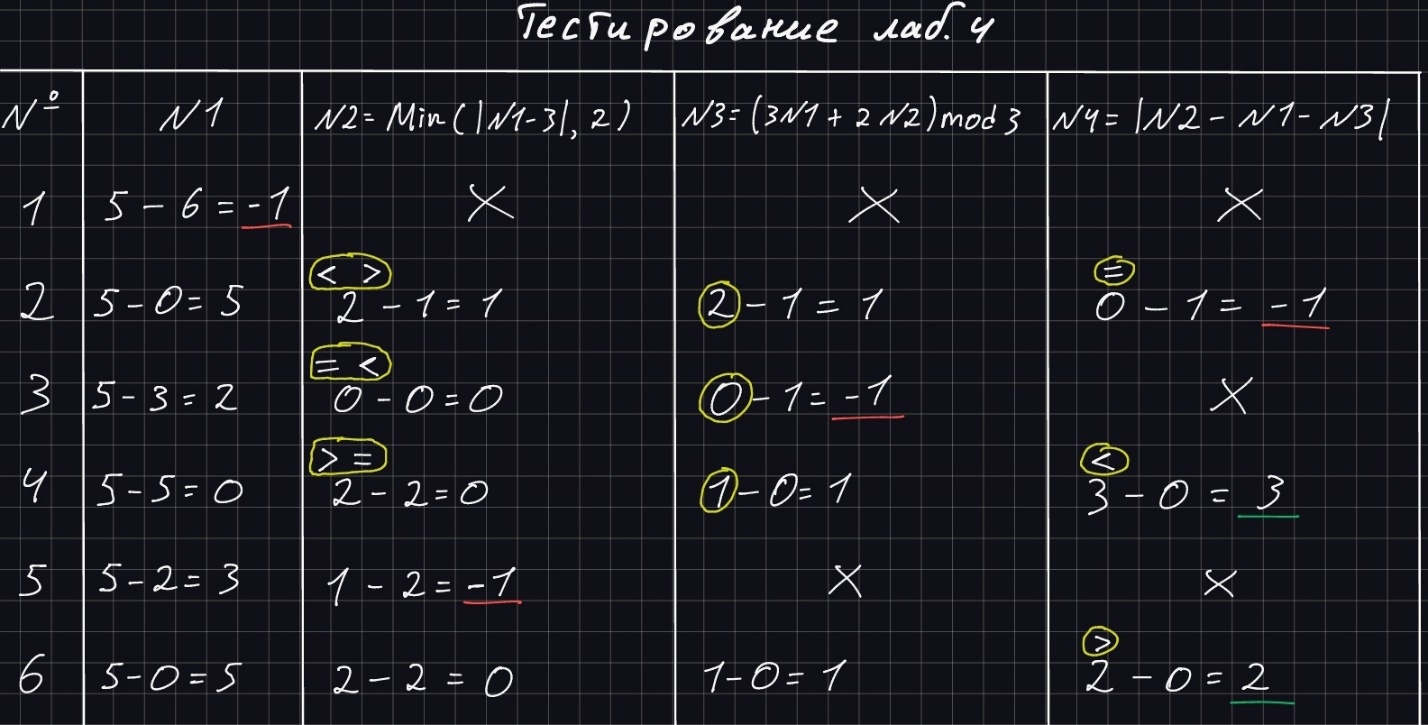


Рисунок 2. – Тестирование программы

Ввиду невозможности адекватно представить результаты тестирования в виде изображений будет показаны только результат ввода правильной последовательности переключений и результат ввода неправильной последовательности с превышением допустимого количества ошибок (см. рисунок 3.1 и 3.2).

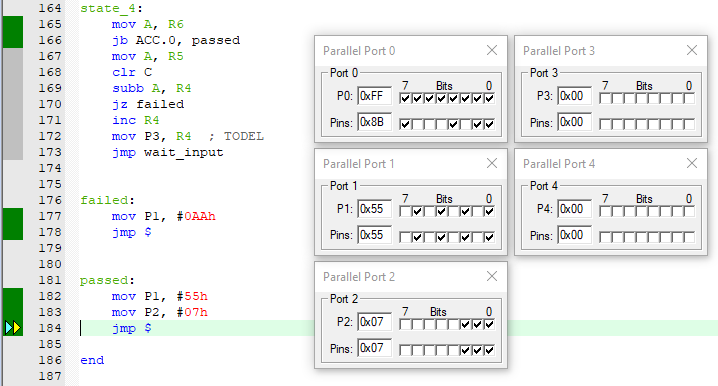


Рисунок 3.1 – Успех

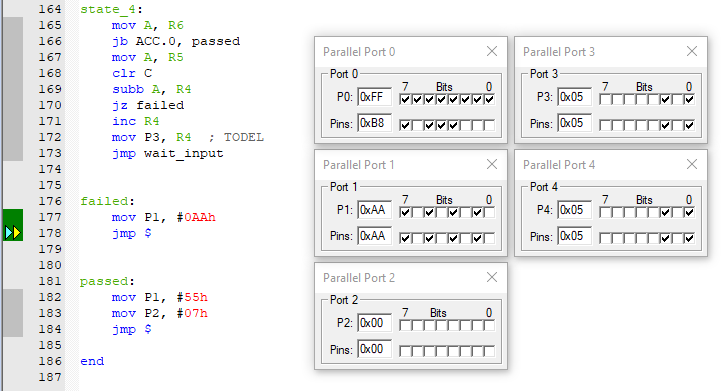


Рисунок 3.2 – Блокировка автомата

4. Список литературы и ссылки

1. Стрелец А. И., Иванников В. С., Ёхин М.Н. Методические указания для выполнения лабораторной работы “Битовый процессор” по курсу “Микропроцессорные устройства и системы” с использованием виртуального стенда. Москва 2018.
2. Е. В. Моисейкин. Микроконтроллеры семейства MCS-51 Теория и практика. Учебно-методическое пособие. Екатеринбург Издательство Уральского университета 2017.
3. Исходный код программы [Электронный ресурс] // <https://github.com/Hypex146/MDnS-Lab-4>.