Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Теория автоматов**

(название дисциплины)

**Лабораторная (практическая) работа №2**

**Микропрограммный автомат**

(название (тема) работы)

**Вариант №8**

Выполнил

Студент группы ИВТАСбд-21

Зюзин Г.А.

(Фамилия И.О.)

Проверил(а):

ст. преподаватель кафедры «ВТ»\_\_\_\_

(должность)

\_\_\_\_\_\_\_\_\_Лылова А.В.\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)

Ульяновск

2023

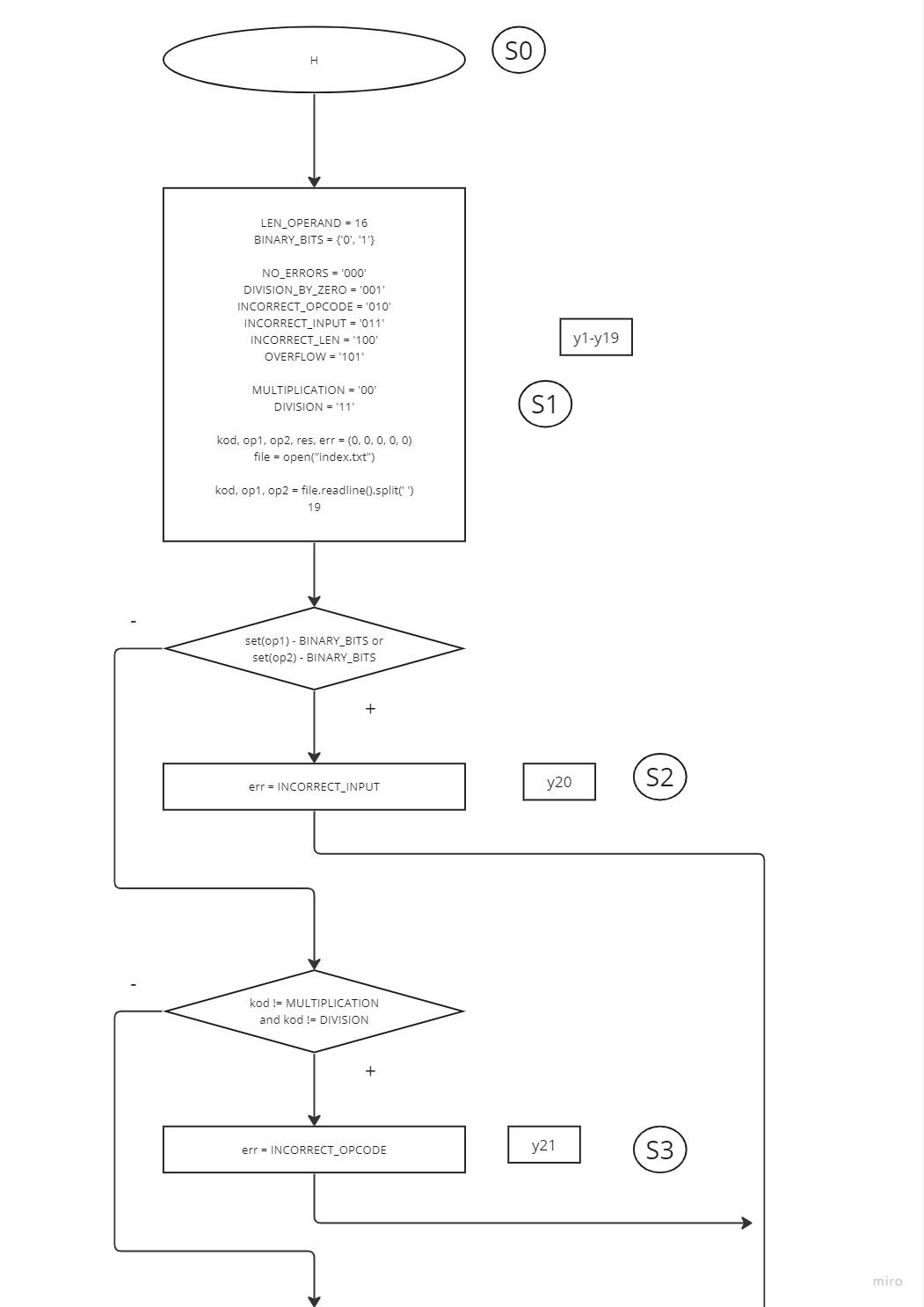
# **Постановка задачи**

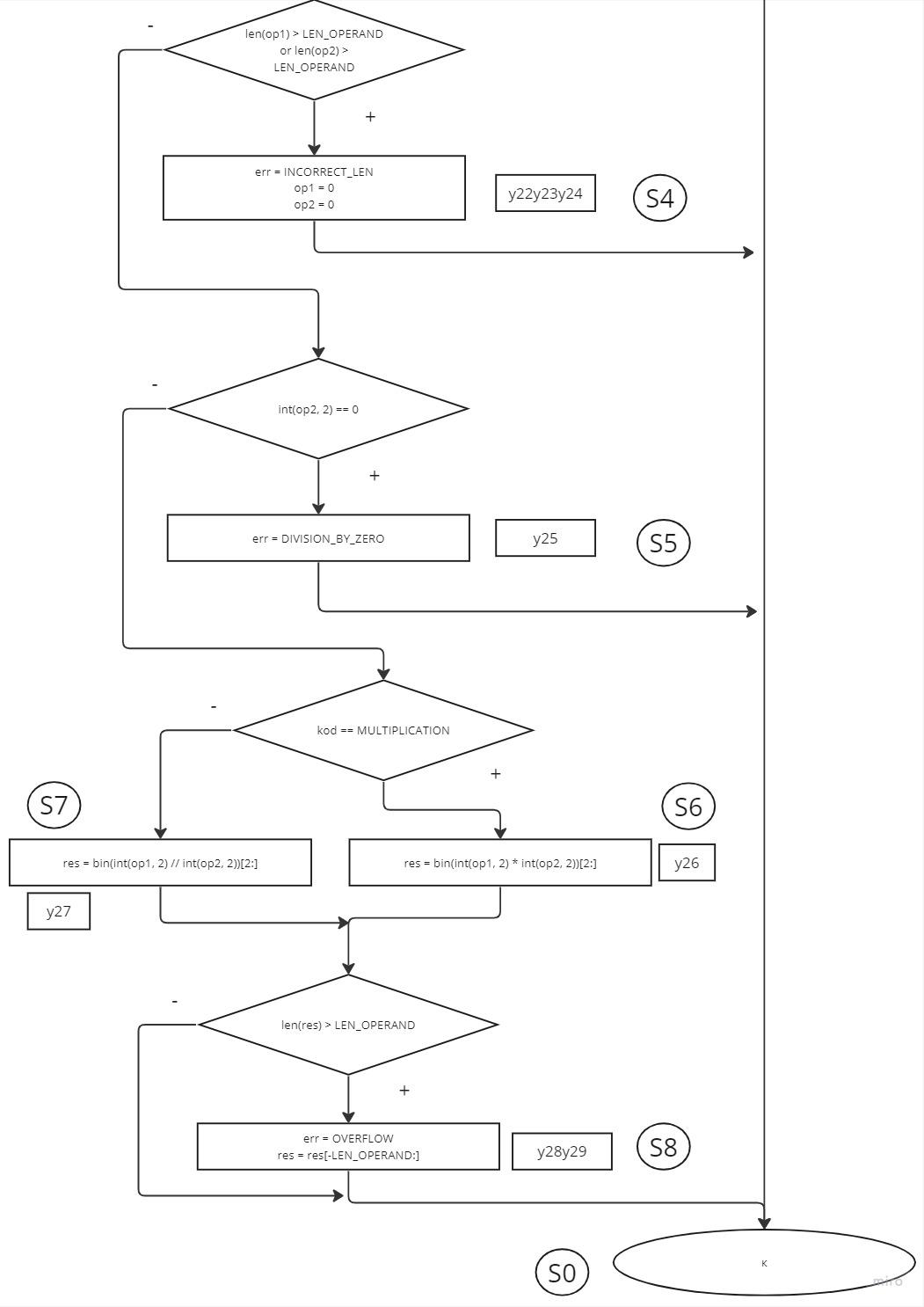
Выполнить программную реализацию автомата Мура, выполняющего микропрограмму операций умножения (код операции - 00) и деления (код операции - 11) с 16 разрядными операндами. Программа должна на каждом шаге выводить значения параметров и текущее состояние.

# **Реализация**

Изначально была написана программа, выполняющая необходимые операции с операндами и обрабатывающая возможные ошибки(в случае возникновения ошибки ее код записывается в отдельную переменную).

Работа данной программы была проверена на разных входных данных, программа работает корректно. Далее, на ее основе была построена блок схема:



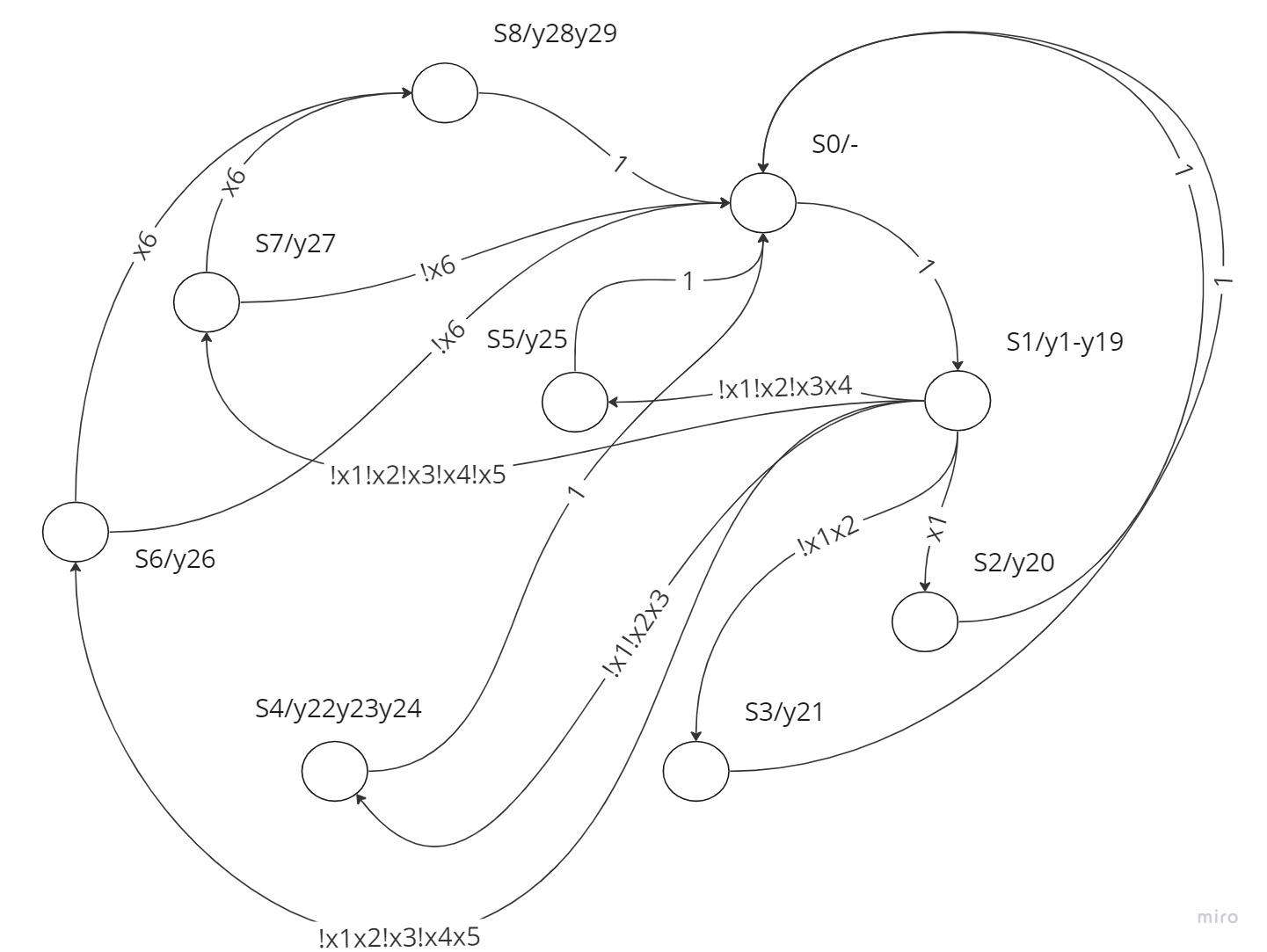


# **Кодирование входных и выходных сигналов**

| x1 | set(op1) - BINARY\_BITS or set(op2) - BINARY\_BITS |
| --- | --- |
| x2 | kod != MULTIPLICATION and kod != DIVISION |
| x3 | len(op1) > LEN\_OPERAND or len(op2) > LEN\_OPERAND |
| x4 | int(op2, 2) == 0 |
| x5 | kod == MULTIPLICATION |
| x6 | len(res) > LEN\_OPERAND |

| y1 | LEN\_OPERAND = 16 | y16 | file = open("index.txt") |
| --- | --- | --- | --- |
| y2 | BINARY\_BITS = {'0', '1'} | y17 | kod, op1, op2 = file.readline().split(' ') |
| y3 | NO\_ERRORS = '000' | y18 |
| y4 | DIVISION\_BY\_ZERO = '001' | y19 |
| y5 | INCORRECT\_OPCODE = '010' | y20 | err = INCORRECT\_INPUT |
| y6 | INCORRECT\_INPUT = '011' | y21 | err = INCORRECT\_OPCODE |
| y7 | INCORRECT\_LEN = '100' | y22 | err = INCORRECT\_LEN |
| y8 | OVERFLOW = '101' | y23 | op1 = 0 |
| y9 | MULTIPLICATION = '00' | y24 | op2 = 0 |
| y10 | DIVISION = '11' | y25 | err = DIVISION\_BY\_ZERO |
| y11 | kod, op1, op2, res, err  =  (0, 0, 0, 0, 0) | y26 | res = bin(int(op1, 2) \*  int(op2, 2))[2:] |
| y12 | y27 | s = bin(int(op1, 2) //  int(op2, 2))[2:] |
| y13 | y28 | err = OVERFLOW |
| y14 | y29 | res = res[-LEN\_OPERAND:] |
| y15 |  |  |

**Граф с указанными переходами и выходами.**



**Тестирование программы.**

После добавления вывода состояний в программу, она соответствует всем требования задания. Рассмотрим примеры ее работы:

Тест №1:

Входные данные (из файла):

00 0000000001000000 0000000000001000

Выходные данные:

S0:

kod = 00

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 000

S1:

kod = 00

op1 = 0000000001000000

op2 = 0000000000001000

res = 0000000000000000

err = 000

S6:

kod = 00

op1 = 0000000001000000

op2 = 0000000000001000

res = 0000001000000000

err = 000

S0:

kod = 00

op1 = 0000000001000000

op2 = 0000000000001000

res = 0000001000000000

err = 000

Тест №2:

Входные данные (из файла):

11 0000000001001001 0000000000001000

Выходные данные:

S0:

kod = 00

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 000

S1:

kod = 11

op1 = 0000000001001001

op2 = 0000000000001000

res = 0000000000000000

err = 000

S7:

kod = 11

op1 = 0000000001001001

op2 = 0000000000001000

res = 0000000000001001

err = 000

S0:

kod = 11

op1 = 0000000001001001

op2 = 0000000000001000

res = 0000000000001001

err = 000

Тест №3

Входные данные (из файла):

10 0000000001001001 0000000000001000

Выходные данные:

S0:

kod = 00

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 000

S1:

kod = 10

op1 = 0000000001001001

op2 = 0000000000001000

res = 0000000000000000

err = 000

S3:

kod = 10

op1 = 0000000001001001

op2 = 0000000000001000

res = 0000000000000000

err = 010

S0:

kod = 10

op1 = 0000000001001001

op2 = 0000000000001000

res = 0000000000000000

err = 010

Тест №4

Входные данные (из файла):

11 0000000001001001 0000000000001

Выходные данные:

S0:

kod = 00

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 000

S1:

kod = 11

op1 = 0000000001001001

op2 = 0000000000000001

res = 0000000000000000

err = 000

S4:

kod = 11

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 100

S0:

kod = 11

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 100

Тест №5

Входные данные (из файла):

11 0000000001001001 0000000000001

Выходные данные:

S0:

kod = 00

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 000

S1:

kod = 11

op1 = 0000000001001001

op2 = 0000000000001234

res = 0000000000000000

err = 000

S2:

kod = 11

op1 = 0000000001001001

op2 = 0000000000001234

res = 0000000000000000

err = 011

S0:

kod = 11

op1 = 0000000001001001

op2 = 0000000000001234

res = 0000000000000000

err = 011

Тест №6

Входные данные (из файла):

11 0000000001001001 0000000000000

Выходные данные:

S0:

kod = 00

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 000

S1:

kod = 11

op1 = 0000000001001001

op2 = 0000000000000000

res = 0000000000000000

err = 000

S5:

kod = 11

op1 = 0000000001001001

op2 = 0000000000000000

res = 0000000000000000

err = 001

S0:

kod = 11

op1 = 0000000001001001

op2 = 0000000000000000

res = 0000000000000000

err = 001

Тест №7

Входные данные (из файла):

00 1100000001001001 1000000010100000

Выходные данные:

S0:

kod = 00

op1 = 0000000000000000

op2 = 0000000000000000

res = 0000000000000000

err = 000

S1:

kod = 00

op1 = 1100000001001001

op2 = 1000000010100000

res = 0000000000000000

err = 000

S8:

kod = 00

op1 = 1100000001001001

op2 = 1000000010100000

res = 1010110110100000

err = 101

S0:

kod = 00

op1 = 1100000001001001

op2 = 1000000010100000

res = 1010110110100000

err = 101

**Листинг кода.**

def print\_state(state, kod, op1, op2, res, err):

print(f"\n{state}:\nkod = {str(kod).zfill(2)}\nop1 = {str(op1).zfill(LEN\_OPERAND)}\nop2 = {str(op2).zfill(LEN\_OPERAND)}\n"

f"res = {str(res).zfill(LEN\_OPERAND)}\nerr = {str(err).zfill(3)}")

LEN\_OPERAND = 16

BINARY\_BITS = {'0', '1'}

NO\_ERRORS = '000' # Нет ошибок

DIVISION\_BY\_ZERO = '001' # Деление на ноль

INCORRECT\_OPCODE = '010' # Неверный код операции

INCORRECT\_INPUT = '011' # Неверная форма ввода

INCORRECT\_LEN = '100' # Неверная длина ввода

OVERFLOW = '101' # Переволнение разрядной сетки

MULTIPLICATION = '00'

DIVISION = '11'

if \_\_name\_\_ == "\_\_main\_\_":

kod, op1, op2, res, err = (0, 0, 0, 0, 0)

print\_state("S0", kod, op1, op2, res, err)

file = open("input.txt")

kod, op1, op2 = file.readline().split(' ')

print\_state("S1", kod, op1, op2, res, err)

if set(op1) - BINARY\_BITS or set(op2) - BINARY\_BITS:

err = INCORRECT\_INPUT

print\_state("S2", kod, op1, op2, res, err)

elif kod != MULTIPLICATION and kod != DIVISION:

err = INCORRECT\_OPCODE

print\_state("S3", kod, op1, op2, res, err)

elif len(op1) != LEN\_OPERAND or len(op2) != LEN\_OPERAND:

err = INCORRECT\_LEN

op1 = 0

op2 = 0

print\_state("S4", kod, op1, op2, res, err)

elif int(op2, 2) == 0:

err = DIVISION\_BY\_ZERO

print\_state("S5", kod, op1, op2, res, err)

else:

if kod == MULTIPLICATION:

res = bin(int(op1, 2) \* int(op2, 2))[2:]

if len(res) > LEN\_OPERAND:

err = OVERFLOW

res = res[-LEN\_OPERAND:]

print\_state("S8", kod, op1, op2, res, err)

else:

print\_state("S6", kod, op1, op2, res, err)

else:

res = bin(int(op1, 2) // int(op2, 2))[2:]

print\_state("S7", kod, op1, op2, res, err)

print\_state("S0", kod, op1, op2, res, err)

# Листинг кода