Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ульяновский государственный технический университет»
Кафедра «Вычислительная техника»

Теория автоматов

Лабораторная работа №2
«Микропрограммный автомат»

Выполнил
Студент группы ИВТбд-21
Ведин В. А.
Проверил(а):
ст. преподаватель кафедры «ВТ»
Лылова А.В.

Ульяновск
2024

ЗАДАНИЕ

Требуется реализовать автомат типа Мили для выполнения микропрограммной операции умножения или деления. Данные подаются через файл, в котором содержатся двоичные числа: код операции («00» - умножение, «11» - деление) и сами операнды, состоящие из 16 битов. Помимо этого, реализация автомата требует обрабатывать некоторые ошибки: длинна операндов, нужный ли код операции, переполнения бита, деление на ноль и др.

ТЕКСТОВОЕ ОПИСАНИЕ РЕАЛИЗОВАННОГО АВТОМАТА РАЗБОРА

На вход программе подаётся файл «operands.txt», содержащий код операции и два 16-ти битных операнда. По начала производится проверка на то являются ли все символы в файле нулями и единицами, т.е. бинарными, затем производится проверка на то является ли операция известной нам, затем производится проверка на длину операндов, после чего производится проверка на деление на 0, если операция – деление, и только после этого начинаются действия. При операции деления, производится сама операция, после добавляются незначащие нули. При операции умножения производится сама операция, после проверяется не переполнился ли наше 16-ти битное число, а после нужно проверить длину нашего числа, и добавить незначащие нули, если это требуется.

ТАБЛИЦЫ ОБОЗНАЧЕНИЙ

Таблица 1. Обозначение входных сигналов.

Входной сигнал	Обозначение
x1	if not all(char in bin_chars for char in temp_str) or temp_str = “”:
x2	if operation != “11” and operation != “00”:
x3	if len(data[1]) != 16 or len(data[2]) != 16:
x4	if operand2 == 0 and operation == "11":
x5	if operation == "11":
x6	if len(res) < 16:
x7	if operation == "00":
x8	if len(res) > 16:
x9	if len(res) < 16:

Таблица 2. Обозначение выходных фукнций.

Выходная функция	Обозначение
y1	file = open("operands.txt", "r")
y2	data = file.readline().split(" ")
y3	bin_chars = ["0", "1"]
y4	temp_str = "".join(data)
y5	error_code = "001"
y6	exit()
y7	operation = data[0]
y8	error_code = “010”

Выходная функция	Обозначение
y9	error_code = "011"
y10	operand1 = int(data[1], 2)
y11	operand2 = int(data[2], 2)
y12	error_code = "100"
y13	res = bin(operand1 // operand2)[2:]
y14	res = "0" * (16 - len(res)) + res
y15	res = bin(operand1 * operand2)[2:]
y16	error_code = "101"

БЛОК-СХЕМА

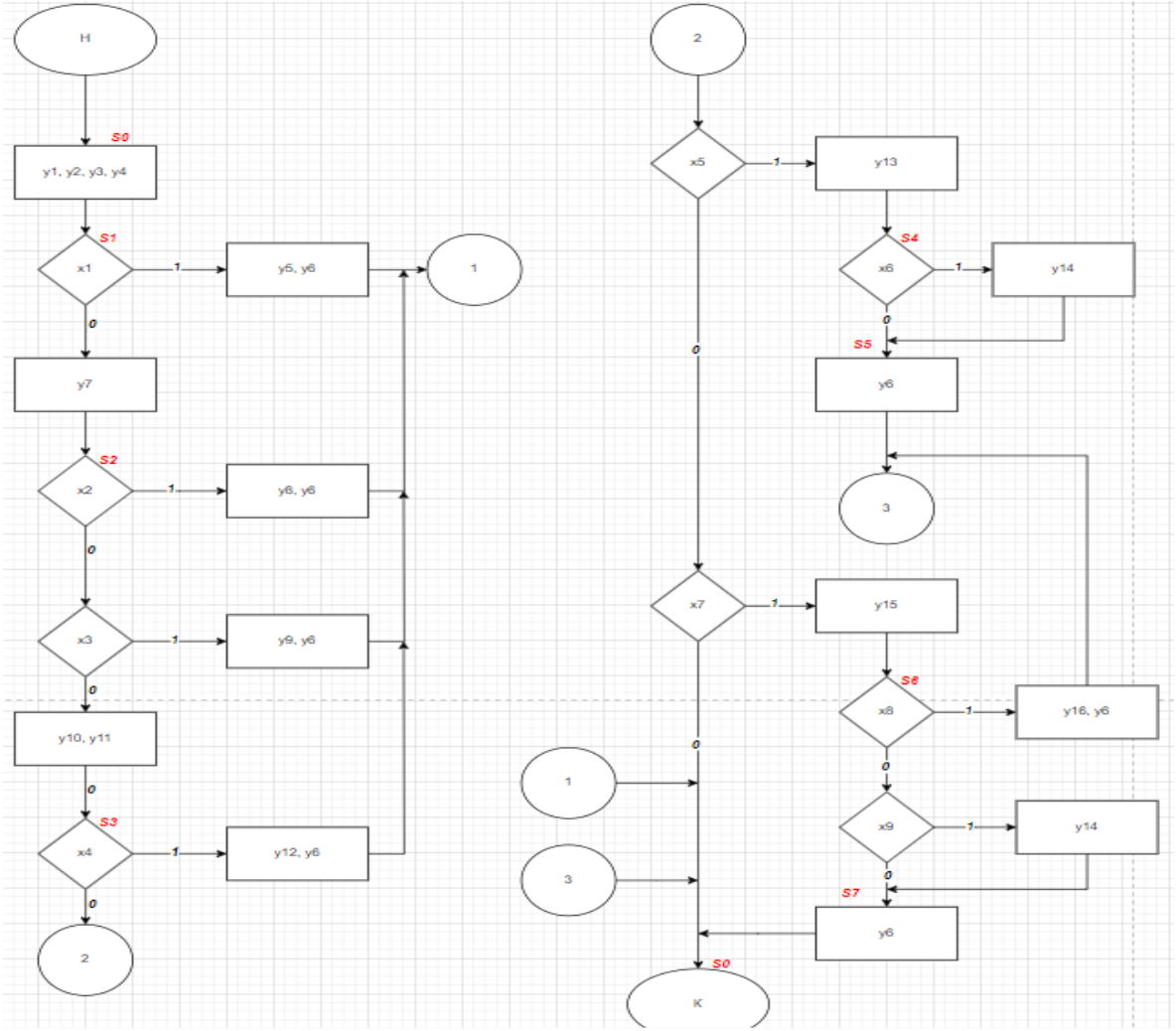

Рис 1. Блок-схема автомата типа Мили для выполнения микропрограммы.

ГРАФ АВТОМАТА

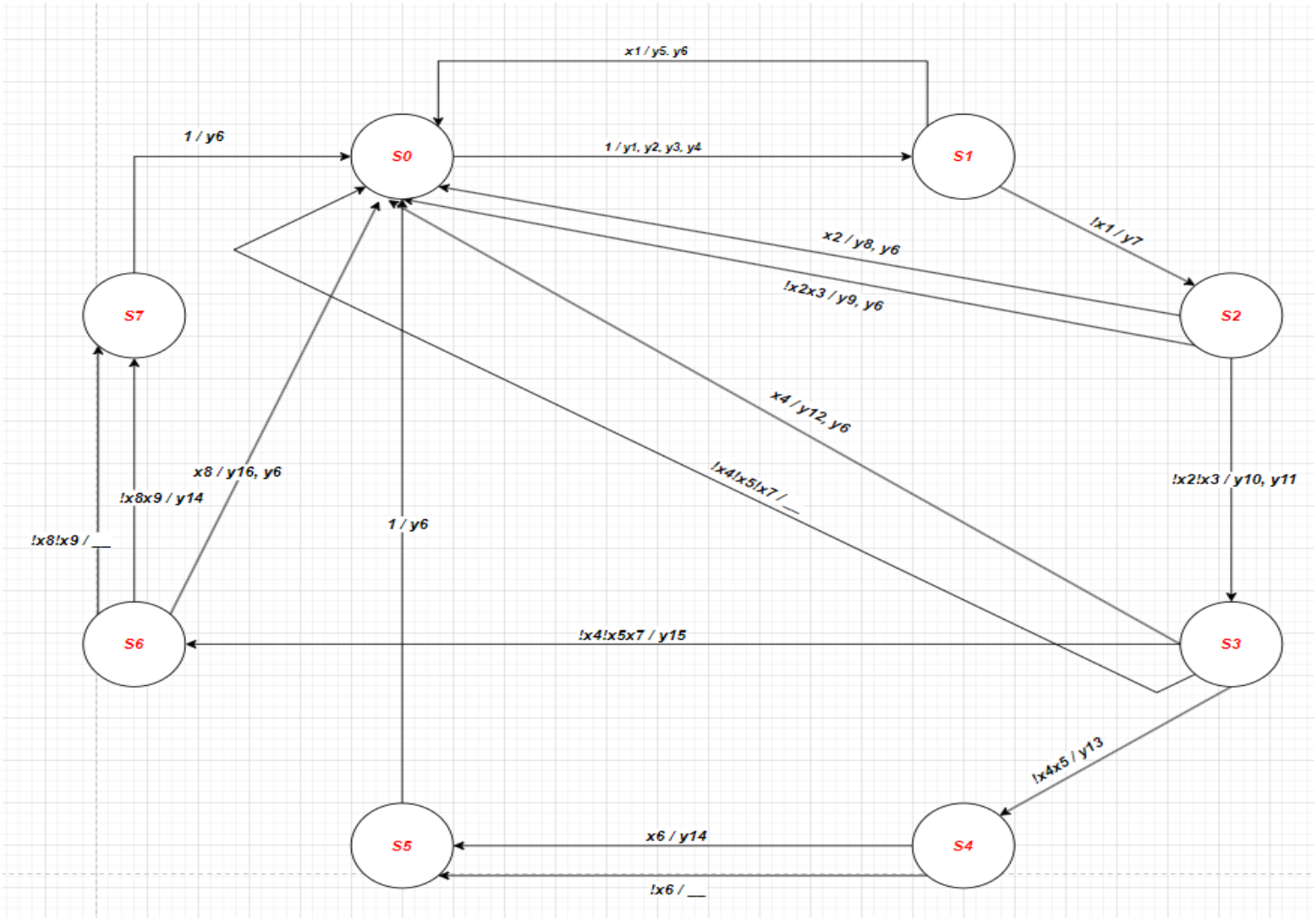

Рис 2. Граф автомата Мили для выполнения микропрограммы.

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Отсутствие в файле каких-либо ошибок с операциями деления и умножения:

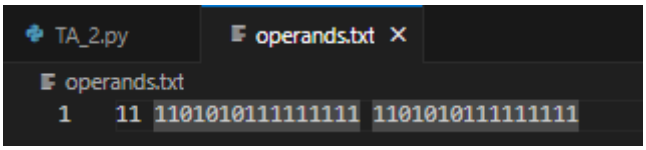

Рис 3.1.1 Исходный файл без ошибок с операцией деления

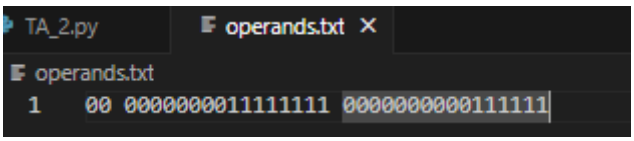

Рис 3.2.1 Исходный файл без ошибок с операцией умножения

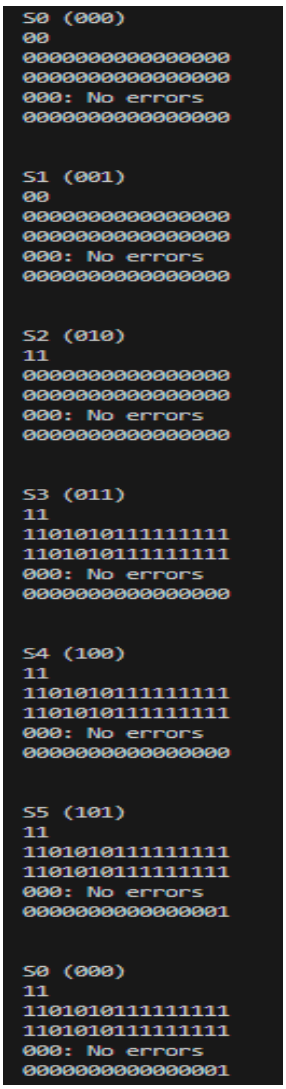

Рис 3.1.2 Выходные данные с операцией деления

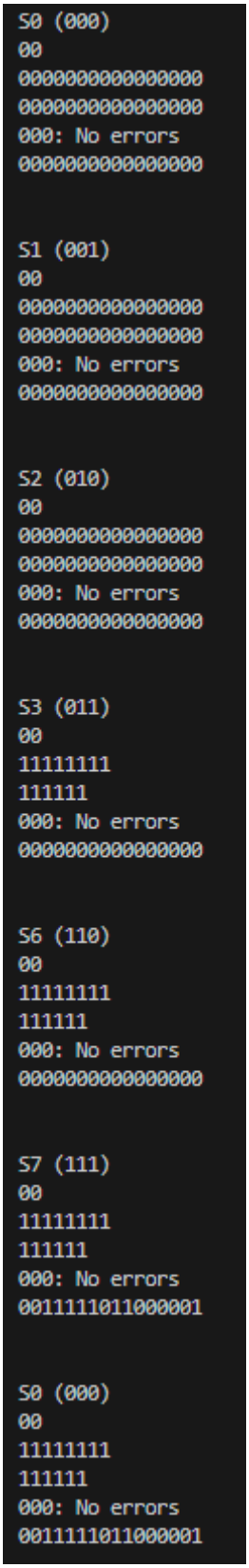

Рис 3.2.2 Выходные данные с операцией умножения

Наличие в файле различных ошибок:

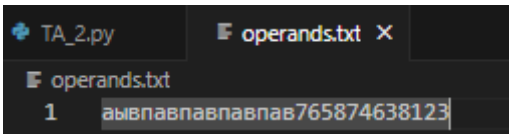

Рис 4.1.1 Файл, содержащий не только бинарные символы.

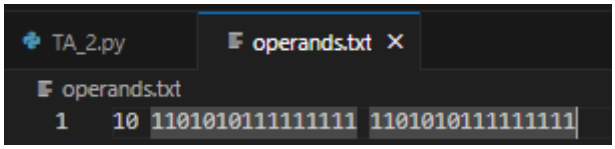

Рис 4.2.1 Файл, содержащий неизвестную операцию

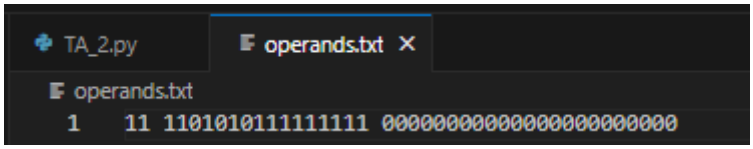

Рис 4.3.1 Файл, содержащий излишнюю длину операнда

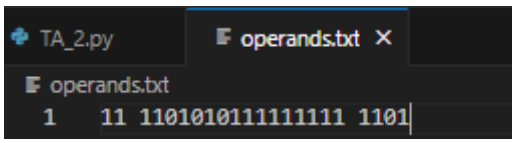

Рис 4.4.1 Файл, содержащий недостающую длину операнда

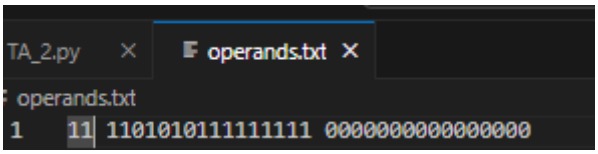

Рис 4.5.1 Файл, содержащий операцию деления и второй операнд – 0

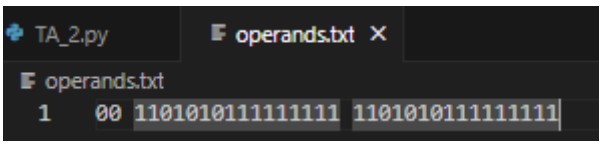

Рис 4.6.1 Файл, содержащий операцию умножения и операнды, произведение которых даст переполнение бита

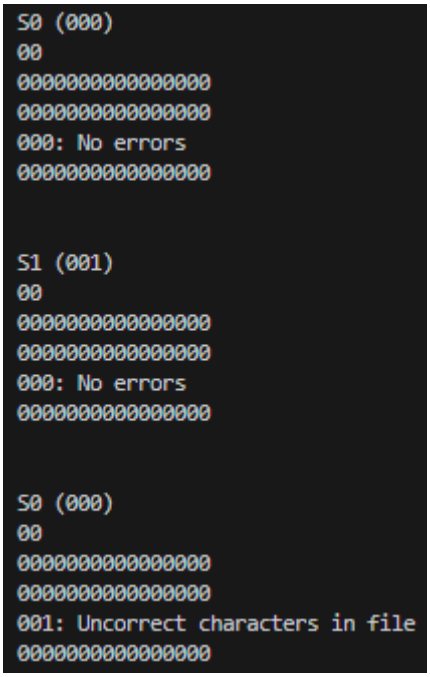

Рис 4.1.2 Выходные данные с ошибкой не только бинарных данных в файле

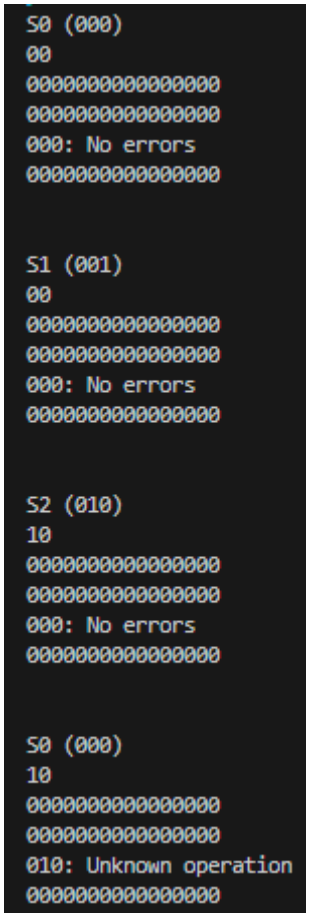

Рис 4.2.2 Выходные данные с ошибкой неизвестной операции

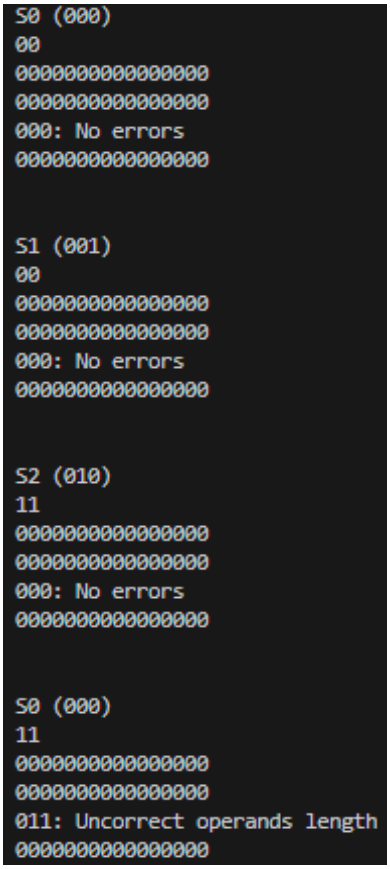

Рис 4.3.2 Выходные данные с ошибкой излишней длинны операнда

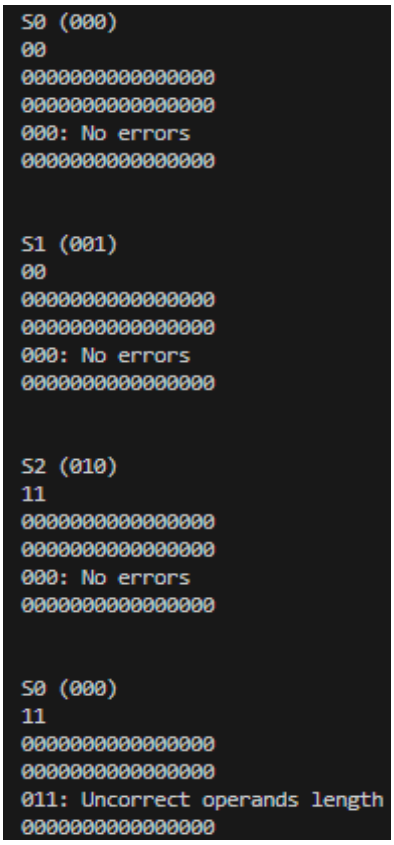

Рис 4.4.2 Выходные данные с ошибкой недостаточной длинны операнда

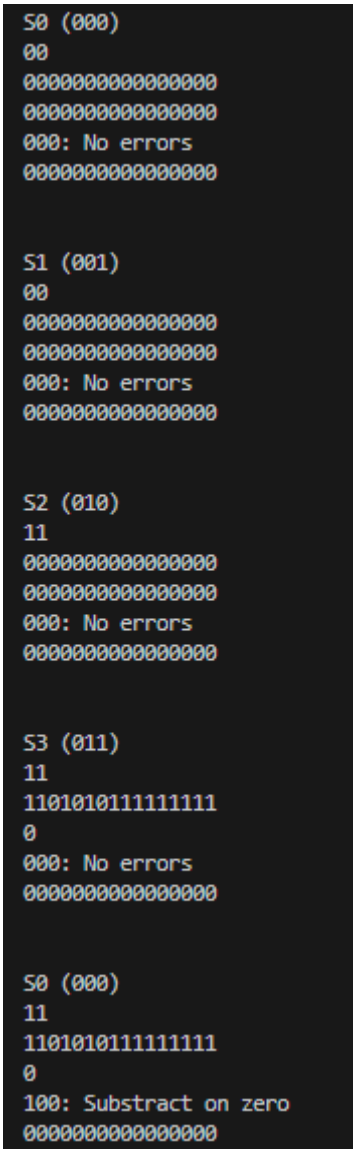

Рис 4.5.2 Выходные данные с ошибкой деления на ноль

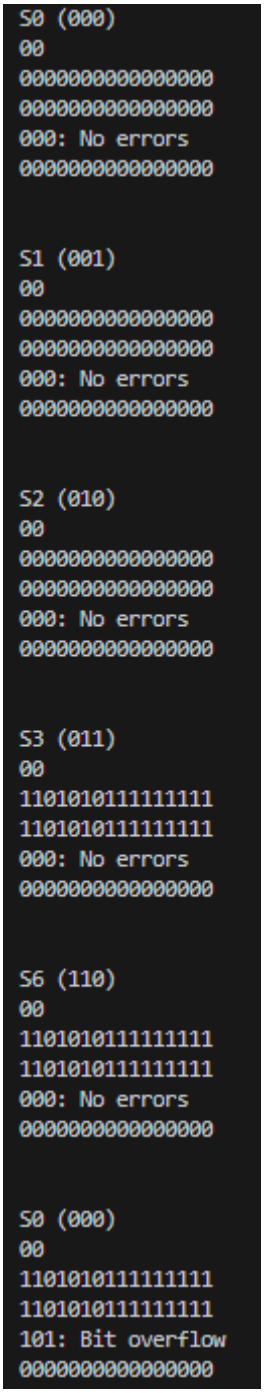

Рис 4.6.2 Выходные данные с ошибкой переполнения бита при умножении

ИТОГИ

По итогу данной работы был спроектирован и реализован автомат типа Мили для реализации микроопераций типа умножения и деления с проверкой различных ошибок. Также были построены блок-схема и граф-схемы алгоритма.

ПРИЛОЖЕНИЕ

```
def printInfo(state, operation, operand1, operand2, error_code, result):
    print(state)
    print(operation)
    print(operand1)
    print(operand2)
    print(error_code)
    print(result)
    print("\n")


printInfo("S0 (000)", "00", "0"*16, "0"*16, "000" + ": No errors", "0"*16)
file = open("operands.txt", "r")
data = file.readline().split(" ")
bin_chars = ["0", "1"]
temp_str = "".join(data)
printInfo("S1 (001)", "00", "0"*16, "0"*16, "000" + ": No errors", "0"*16)
if not all(char in bin_chars for char in temp_str) or temp_str == "":
    error_code = "001"
    printInfo("S0 (000)", "00", "0"*16, "0"*16, error_code + ": Uncorrect characters
in file", "0"*16)
    exit()
operation = data[0]
printInfo("S2 (010)", operation, "0"*16, "0"*16, "000" + ": No errors", "0"*16)
if operation != "11" and operation != "00":
    error_code = "010"
    printInfo("S0 (000)", operation, "0"*16, "0"*16, error_code + ": Unknown
operation", "0"*16)
    exit()
if len(data[1]) != 16 or len(data[2]) != 16:
    error_code = "011"
    printInfo("S0 (000)", operation, "0"*16, "0"*16, error_code + ": Uncorrect
operands length", "0"*16)
    exit()
operand1 = int(data[1], 2)
operand2 = int(data[2], 2)
printInfo("S3 (011)", operation, bin(operand1)[2:], bin(operand2)[2:], "000" ": No
errors", "0"*16)
if operand2 == 0 and operation == "11":
    error_code = "100"
    printInfo("S0 (000)", operation, bin(operand1)[2:], bin(operand2)[2:],
error_code + ": Substract on zero", "0"*16)
    exit()
if operation == "11":
    res = bin(operand1 // operand2)[2:]
```

```
    printInfo("S4 (100)", operation, bin(operand1)[2:], bin(operand2)[2:], "000" +
": No errors", "0"*16)
    if len(res) < 16:
        res = "0" * (16 - len(res) ) + res
    printInfo("S5 (101)", operation, bin(operand1)[2:], bin(operand2)[2:], "000" +
": No errors", res)
    printInfo("S0 (000)", operation, bin(operand1)[2:], bin(operand2)[2:], "000" +
": No errors", res)
    exit()
if operation == "00":
    res = bin(operand1 * operand2)[2:]
    printInfo("S6 (110)", operation, bin(operand1)[2:], bin(operand2)[2:], "000" +
": No errors", "0"*16)
    if len(res) > 16:
        error_code = "101"
        printInfo("S0  (000)",  operation,  bin(operand1)[2:],  bin(operand2)[2:],
error_code + ": Bit overflow", "0"*16)
        exit()
    if len(res) < 16:
        res = "0" * (16 - len(res)) + res
    printInfo("S7 (111)", operation, bin(operand1)[2:], bin(operand2)[2:], "000" +
": No errors", res)
    printInfo("S0 (000)", operation, bin(operand1)[2:], bin(operand2)[2:], "000" +
": No errors", res)
    exit()
```