ГУАП

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ассистент |  |  |  | А.Н. Долидзе |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| ПРОЕКТИРОВАНИЕ КОНЕЧНОГО АВТОМАТА ДЛЯ УСТРОЙСТВА ПРОСТЕЙШЕГО 8-БИТНОГО ПРОЦЕССОРА |
| по курсу: ОРГАНИЗАЦИЯ ЭВМ И ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4941 |  |  |  | Н.С. Горбунов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель работы:** разработать конечный автомат для устройства 8-битного процессора.

**Задание на лабораторную работу**

**Вариант 7:**

Вид архитектуры: Гарвардская (команды и данные в разных ЗУ)

Тип данных: беззнаковый

Операции в АЛУ:

Деление с остатком

lsr

not

Пересылка: mov

Переход по условию, в соответствии с операцией сравнения: <

Формат команды: КОП АО1 АО2

Способы адресации:

1. Косвенная адресация по смещению
2. Адресация с автоувеличением (простая косвенная с автоувеличением)

**Системы команд**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Код | Команда | Свойство | Кол-во оп. | Функция АЛУ | Сигналы УУ | Флаги |
| 0000 | NOT | Логическое отрицание | 1 | Логическое отрицание | CC – 4 бита код команды на АЛУ  DR – 1 выдать данные в ШД данных  RO – команда АЛУ записать операнд в аккумулятор  Ready – команда от АЛУ о выполнении операции |  |
| 0001 | LSR | Сдвиг вправо | 2 | Сдвиг вправо | CC – 4 бита код команды на АЛУ  DR – 1 выдать данные в ШД данных  RO – команда АЛУ записать операнд в аккумулятор  Ready – команда от АЛУ о выполнении операции | C |
| 0010 | DIV | Деление с остатком | 2 | Деление с остатком | CC – 4 бита код команды на АЛУ  DR –выдать данные в ШД данных  Dw – записать в память данных из ШД  RO – команда АЛУ записать операнд в аккумулятор  Rw – записать остаток в РОН  RN- номер регистра  Ready – команда от АЛУ о выполнении операции |  |
| 0011 | MOVB | Пересылка байта | 2 |  | RR – прочитать из РОН  RN- номер регистра  DR – выдать данные в ШД данных  DA- адрес в памяти данных |  |
| 0111 | MOVW | Пересылка слова | 2 |  | RR – прочитать из РОН  RN- номер регистра  DR – выдать данные в ШД данных  DA- адрес в памяти данных |  |
| 1111 | MOVD | Пересылка двойного слова | 2 |  | RR – прочитать из РОН  RN- номер регистра  DR – выдать данные в ШД данных  DA- адрес в памяти данных |  |
| 0100 | BLLSSU | Переход если меньше | 1 |  | RR – прочитать из РОН  RN- номер регистра  Увеличить PC на значение из регистра | С |

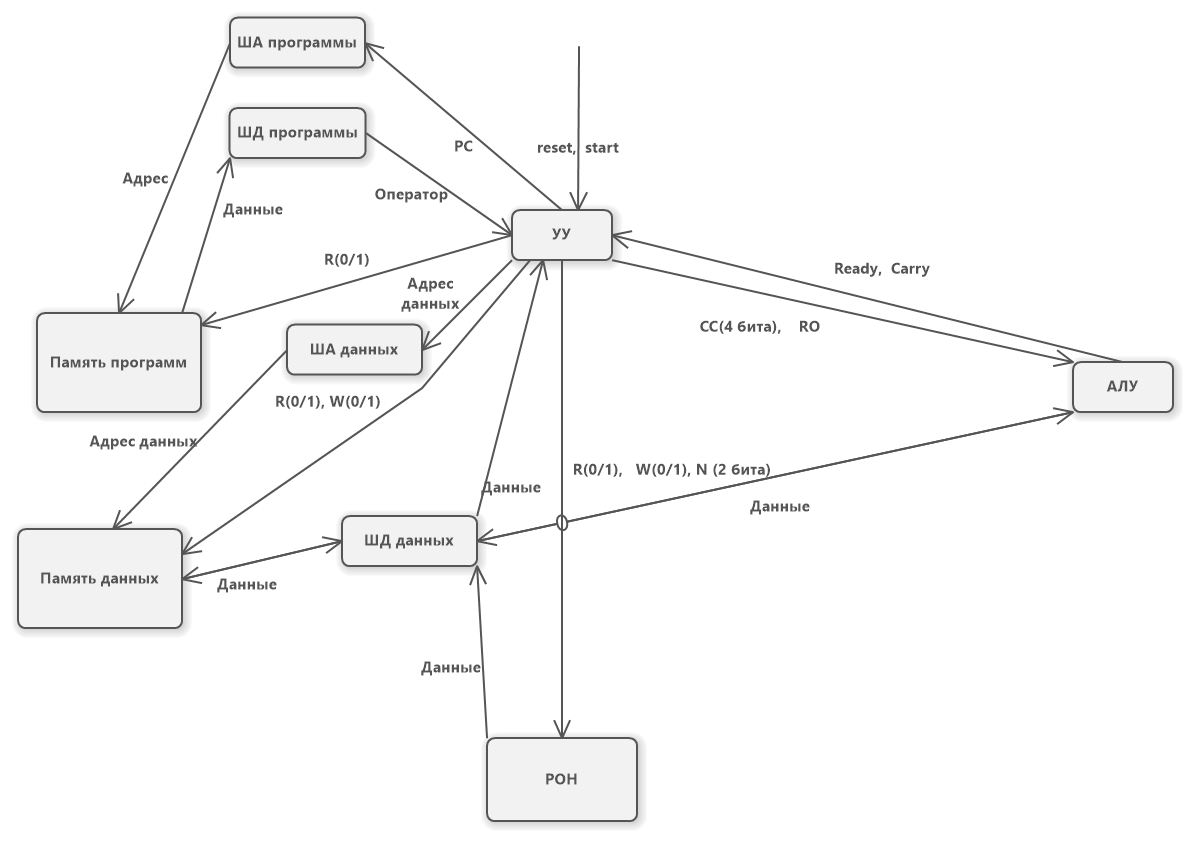


Рисунок 1. Детализированная схема архитектуры

**Примеры кодировки режимов адресации**

|  |  |  |
| --- | --- | --- |
| Мнемокод | Машинный код | Комментарий |
| MOVB (R0)+,(R0)+ | 00000011  10000000  10000000 | Начальное значение R0 – 00000000  Команда переместит байт по адресу 0h в адрес 1h |
| MOVD (R0)+,(R0)+ | 00001111  10000000  10000000 | Начальное значение R0 – 00000000  Команда переместит 4 байта по адресу 0h-3h в адрес 4h-7h |
| MOVB (R0)+, 5 (R0) | 00000011  10000000  00000101 | Начальное значение R0 – 00000000  Команда переместит байт по адресу 0h в адрес 6h |

КОП, бит адресации, номер регистра, смещение.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| КОП | | | | | | | |  | Адресация | | |  |  |  |  |  |  | Адресация | | |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
|  |  |  |  |  |  |  |  |  |  | Регистр | | |  |  |  |  |  |  | Регистр | |  | Смещение | | | |

**Таблица входов/выходов**

**АЛУ**

|  |  |
| --- | --- |
| Входы | Выходы |
| СС – код команды на выполнения от УУ | Ready – сигнал выполнения команды |
| RO – сигнал на запись операнда из ШД во внутренний аккумулятор | Carry – флаг результата операции |

**УУ**

|  |  |
| --- | --- |
| Входы | Выходы |
| Reset – сброс PC | PR– чтение из памяти программ |
| Start – запуск программы | PC – программный счетчик в ША программы |
| PO – оператор из памяти программы | DR\DW – чтение\запись в память данных |
| D – данные из ШД памяти данных | DA- адрес в память данных |
| Ready – сигнал выполнения команды от АЛУ | СС – код команды на выполнения для АЛУ |
| Carry – флаг результата операции от АЛУ | RO – сигнал на запись операнда из ШД во внутренний аккумулятор АЛУ |
| RR\RW – чтение\запись в РОН |  |
| RN – номер регистра |  |

**РОН**

|  |  |
| --- | --- |
| Входы | Выходы |
| RR\RW – чтение\запись в РОН | D – данные в ШД памяти данных |
| RN – номер регистра |  |

**Память программ**

|  |  |
| --- | --- |
| Входы | Выходы |
| PR– чтение из памяти программ | P – данные из памяти |
| A – адрес из ША |  |

**Память данных**

|  |  |
| --- | --- |
| Входы | Выходы |
| DR\DW – чтение\запись в память данных | D – данные из памяти |
| A – адрес из ША |  |
| D – данные из памяти |  |

**Граф конечного автомата**



Рисунок 2. Обработка команд с двумя операндами и обращением к АЛУ



Рисунок 3. Обработка однооперандных команд с обращением к АЛУ



Рисунок 4. Обработка пересылок



Рисунок 5. Обработка перехода по сравнению



Рисунок 6. Обработка операнда



Рисунок 7. Выполнение пересылки

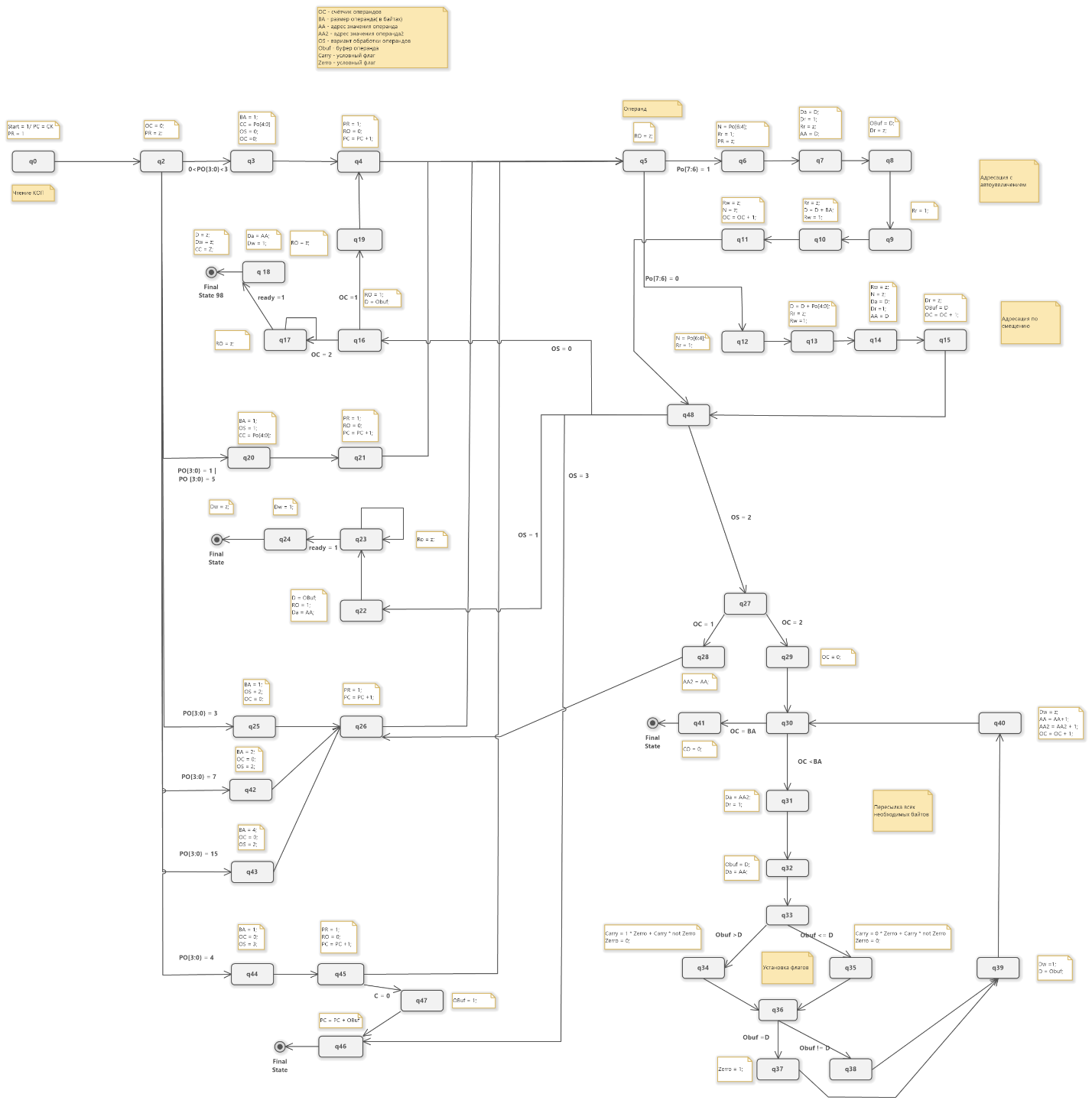


Рисунок 8. Общий вид конечного автомата

**Реализация конечного автомата Verilog**

Листинг 1.

module EVM\_5LW(clk, start,reset, ready, program\_data, data\_bus\_in, data\_bus\_out, program\_address, data\_address, program\_r, reg\_r, reg\_w, data\_r, data\_w, alu\_r, reg\_N, alu\_cc );

input clk;

input start, reset, ready;

input [7:0] program\_data;

input [7:0] data\_bus\_in;

output reg [7:0] data\_bus\_out;

output reg [7:0] program\_address;

output reg [7:0] data\_address;

output reg program\_r, reg\_r, reg\_w, data\_r, data\_w, alu\_r;

output reg [1:0] reg\_N;

output reg [3:0] alu\_cc;

reg [7:0] state;

reg [7:0] pc;

reg [1:0] operand\_counter;

reg [1:0] byte\_amount;

reg [7:0] operand\_address;

reg [7:0] operand\_address2;

reg [1:0] operand\_state;

reg [7:0] operand\_buf;

//reg [7:0] data\_bus\_reg;

reg carry\_flag;

reg zerro\_flag;

//assign program\_address = pc;

//assign data\_bus = (data\_w) || (reg\_w)? data\_bus\_reg: 8'bzzzzzzzz; // работа с inout

//assign data\_bus = (data\_w)? data\_bus\_reg: 8'bzzzzzzzz; // работа с inout

always @(posedge clk)

begin

if (reset)

begin

program\_address = 8'd0;// обнуление счётчика

state = 8'd0;// обнуление состояния автомата

end

case (state)

//активация

8'd0: begin

if (start == 1'b1)

begin

program\_r <= 1'b1;

state = 8'd2;

end

end

// 8'd1: begin

// program\_r <= 1'bz;

// pc = pc + 1;

// state <= 8'd2;

// end

// выбор обработки по операции

8'd2: begin

operand\_counter = 2'd0;

program\_r <= 1'bz;

if (program\_data[3:0]> 4'b0000 && program\_data[3:0]< 4'b0011)

begin

state = 8'd3;

end

else begin

if (program\_data[3:0]== 4'b0001 || program\_data[3:0] == 4'b0101)

begin

state = 8'd20;

end

else begin

if (program\_data[3:0] == 4'b0011)

begin

state = 8'd25;

end

else begin

if (program\_data[3:0] == 4'b0111)

begin

state = 8'd42;

end

else begin

if (program\_data[3:0] == 4'b1111)

begin

state = 8'd43;

end

else begin

if (program\_data[3:0] == 4'b0100)

begin

state = 8'd44;

end

end

end

end

end

end

end

// команды с двумя операндами

8'd3: begin

byte\_amount = 1;

alu\_cc = program\_data[3:0];

operand\_state = 0;

operand\_counter = 0;

state = 8'd4;

end

// взять операнд

8'd4: begin

program\_r <= 1;

alu\_r <= 0;

program\_address = program\_address + 1'd1;

state = 8'd5;

end

// Начало: Обработка операнда

8'd5: begin

if (program\_data[7] == 1'b1)

begin

state = 8'd6;

end

else begin

state = 8'd12;

end

end

// Начало: Адресация с автоувеличением

8'd6: begin

reg\_N = program\_data[6:5];

program\_r <= 1'bz;

reg\_r <= 1;

state = 8'd7;

end

8'd7: begin

data\_address = data\_bus\_in;

operand\_address <= data\_bus\_in;

data\_r <= 1'b1;

//

reg\_r <= 1'bz;

state = 8'd8;

end

8'd8: begin

data\_r <= 1'bz;

operand\_buf = data\_bus\_in;

state = 8'd9;

end

8'd9: begin

reg\_r <= 1'b1;

state = 8'd10;

end

8'd10: begin

reg\_r <= 1'bz;

reg\_w <= 1'b1;

data\_bus\_out = data\_bus\_in + byte\_amount;

state = 8'd11;

end

8'd11: begin

reg\_w <= 1'bz;

reg\_N = 2'bzz;

operand\_counter = operand\_counter + 1;

state = 8'd48;

end

// Конец: Адресация с автоувеличением

// Начало: Адресация по смещению

8'd12: begin

reg\_N = program\_data[6:5];

reg\_r <= 1;

state = 8'd13;

end

8'd13: begin

operand\_address = data\_bus\_in + program\_data[4:0];

data\_bus\_out = operand\_address;

program\_r <= 1'b0;

reg\_r <= 1'bz;

reg\_w <= 1'b1;

state = 8'd14;

end

8'd14: begin

reg\_w <= 1'bz;

reg\_N = 2'bzz;

data\_address = operand\_address;

data\_r <= 1'b1;

state = 8'd15;

end

8'd15: begin

data\_r <= 1'bz;

operand\_buf = data\_bus\_in;

operand\_counter = operand\_counter + 1;

state = 8'd48;

end

// Конец: Адресация по смещению

8'd16: begin

alu\_r = 1'b1;

data\_bus\_out = operand\_buf;

if (operand\_counter == 2)

begin

state = 8'd17;

end

else begin

state = 8'd19;

end

end

8'd17: begin

alu\_r = 1'bz;

if (ready == 1)

begin

state = 8'd18;

end

end

8'd18: begin

data\_address = operand\_address;

data\_w = 1'b1;

state = 8'd98;

end

8'd98: begin // конечное состояние

data\_w = 1'bz;

alu\_cc = 0;

end

8'd19: begin

alu\_r = 1'bz;

state = 8'd4;

end

// Конец: Обработка операнда

// Начало: Обработка однооперндных команд с АЛУ

8'd20: begin

byte\_amount = 1;

operand\_state = 1;

operand\_counter = 0;

alu\_cc = program\_data[3:0];

state = 8'd21;

end

8'd21: begin

program\_r = 1'b1;

alu\_r = 1'b0;

program\_address = program\_address+1;

state = 8'd5;

end

8'd22: begin

data\_bus\_out = operand\_buf;

alu\_r = 1'b1;

data\_address = operand\_address;

state = 8'd23;

end

8'd23: begin

alu\_r = 1'b0;

if (ready == 1)

begin

state = 8'd24;

end

end

8'd24: begin

data\_w = 1'b1;

state = 8'd97;

end

8'd97: begin // конечное состояние

data\_w =1'bz;

end

// Конец: Обработка однооперндных команд с АЛУ

// Начало: Обработка пересылок

8'd25: begin

byte\_amount = 1;

operand\_state = 2;

operand\_counter = 0;

state = 8'd26;

carry\_flag = 0;

zerro\_flag = 0;

end

8'd26: begin

program\_r = 1'b1;

program\_address = program\_address + 1;

state = 8'd5;

end

8'd27: begin

if (operand\_counter == 1)

begin

state = 8'd28;

end

else begin

state = 8'd29;

end

end

8'd28: begin

operand\_address2 = operand\_address;

state = 8'd26;

end

8'd29: begin

operand\_counter = 0;

state = 8'd30;

end

8'd30: begin

if (operand\_counter == byte\_amount)

begin

state = 8'd41;

end

else begin

state = 8'd31;

end

end

8'd31: begin

data\_address = operand\_address2;

data\_r = 1'b1;

state = 8'd32;

end

8'd32: begin

operand\_buf = data\_bus\_in;

data\_address = operand\_address;

state = 8'd33;

end

8'd33: begin

if (operand\_buf > data\_bus\_in)

begin

state = 8'd34;

end

else begin

state = 8'd35;

end

end

8'd34: begin

carry\_flag = 1 & zerro\_flag | carry\_flag & !zerro\_flag;

zerro\_flag = 0;

state = 8'd36;

end

8'd35: begin

carry\_flag = 0 & zerro\_flag | carry\_flag & !zerro\_flag;

zerro\_flag = 0;

state = 8'd36;

end

8'd36: begin

if (operand\_buf == data\_bus\_in)

begin

state = 8'd37;

end

else begin

state = 8'd39;

end

end

8'd37: begin

zerro\_flag = 1;

state = 8'd39;

end

//8'd38: begin

//

// end

8'd39: begin

data\_w = 1'b1;

data\_r = 1'bz;

data\_bus\_out = operand\_buf;

state = 8'd40;

end

8'd40: begin

data\_w = 1'bz;

operand\_address = operand\_address +1;

operand\_address2 = operand\_address2 + 1;

operand\_counter = operand\_counter +1;

state = 8'd30;

end

8'd41: begin //Конечное состояние

operand\_counter = 0;

end

// Конец: Обработка пересылок

8'd42: begin

byte\_amount = 2;

operand\_state = 2;

operand\_counter = 0;

state = 8'd26;

carry\_flag = 0;

zerro\_flag = 0;

end

8'd43: begin

byte\_amount = 4;

operand\_state = 2;

operand\_counter = 0;

state = 8'd26;

carry\_flag = 0;

zerro\_flag = 0;

end

8'd44: begin

byte\_amount = 1;

operand\_state = 3;

operand\_counter = 0;

state = 8'd45;

end

8'd45: begin

if (carry\_flag == 0) begin

state = 8'd47;

end

else begin

program\_r = 1'b1;

alu\_r = 1'b0;

program\_address = program\_address + 1;

state = 8'd5;

end

end

8'd46: begin //Конечное состояние

program\_address = program\_address + operand\_buf;

end

8'd47: begin

operand\_buf = 1;

state = 8'd47;

end

8'd48: begin

if (operand\_state == 0) begin

state = 8'd16;

end

else begin

if (operand\_state == 1) begin

state = 8'd22;

end

else begin

if (operand\_state == 2) begin

state = 8'd27;

end

else begin

if (operand\_state == 1) begin

state = 8'd46;

end

end

end

end

end

endcase

end

endmodule

**Скриншоты симуляции**

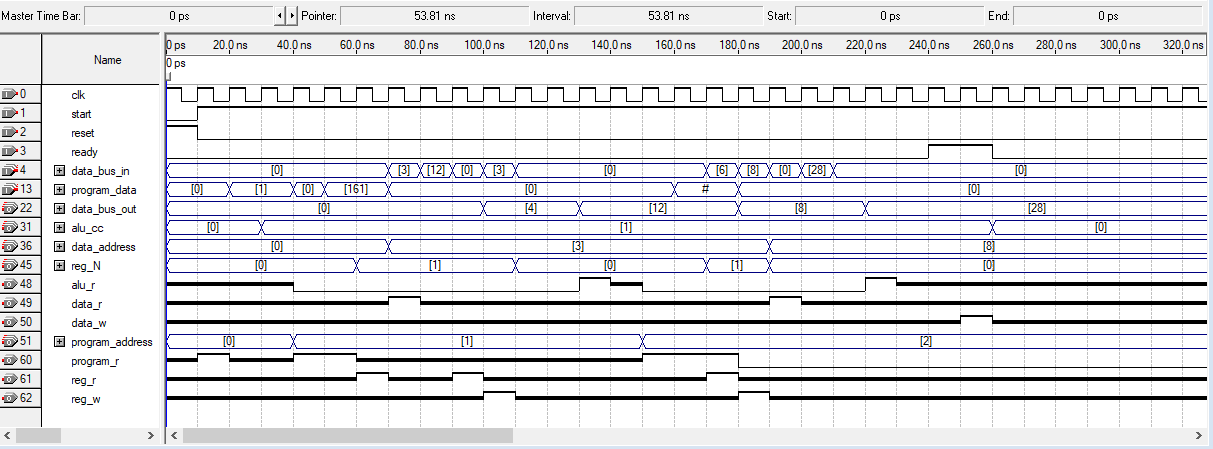


Рисунок 9. Симуляция команды LSR

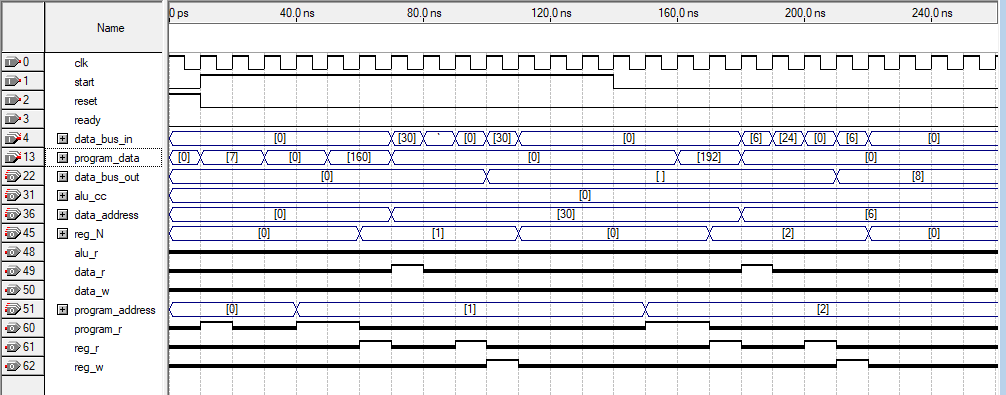


Рисунок 10. Симуляция команды MOVW часть 1

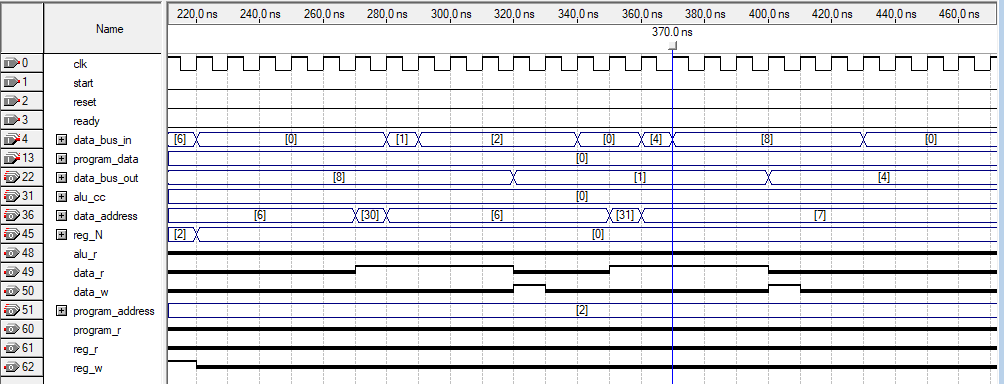


Рисунок 11. Симуляция команды MOVW часть 2

**Вывод:** разработал конечный автомат для устройства 8-битного процессора.