**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Пермский национальный исследовательский политехнический университет»**

Кафедра ИТАС

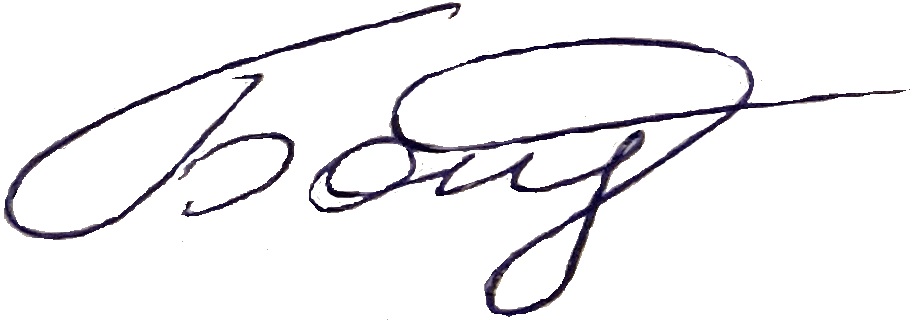
**К У Р С О В АЯ РАБОТА**

**по дисциплине «ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ»**

**Тема: «Структурно-алгоритмическое проектирование ЭВМ»**

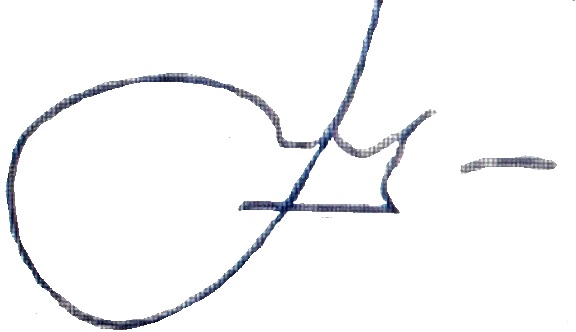
**Выполнил**

студент группы РИС-19-1б



\_\_\_\_\_\_\_\_\_\_\_\_\_ Миннахметов Э.Ю.

**Руководитель**



\_\_\_\_\_\_\_\_\_\_\_\_ Погудин А.Л.

**Дата сдачи** 18.01.2022 г.

**Дата защиты** 20.01.2022 г.

**Оценка** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Пермь, 2020**

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет»

Кафедра ИТАС

«УТВЕРЖДАЮ» Заведующий кафедрой ИТАС

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Р.А. Файзрахманов

«18» сентября 2021 год

**ЗАДАНИЕ**

**на выполнение курсовой работы**

Фамилия И.О. Миннахметов Эльдар Юлдашевич

Факультет Электротехнический Группа РИС-19-1б

Начало выполнения работы: 21 сентября 2021 года

Контрольные сроки просмотра работы: \_18.01.2022\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Защита работы: \_20.01.2022\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Наименование темы: «Структурно-алгоритмическое проектирование ЭВМ».

2. Исходные данные к работе (проекта):

Объект исследования – Устройство управления

Предмет исследования – Алгоритм работы и структура устройства

Цель работы (проекта) – Разработать контроллер ассоциативной памяти, хранящей 64\*8- разрядных двоичных кодов с выходом по «равно признаку» и «не равно признаку». Признаком поиска может быть 1,2,4,8 двоичных разрядов. Результатом поиска, выводимым на ШД, является весь байт.

3. Содержание:

3.1 Исследование предметной области курсовой работы

3.2 Анализ исходных данных задания на курсовую работу

3.3 Спецификация устройства на уровне «черного ящика»

3.4 Представление устройства в виде операционной и управляющей частей

3.5 Разработка структуры устройства

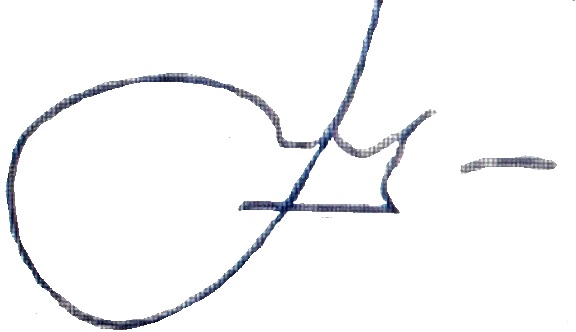
3.6 Составление алгоритма работы устройства.

3.7 Разработка микропрограммы работы устройства

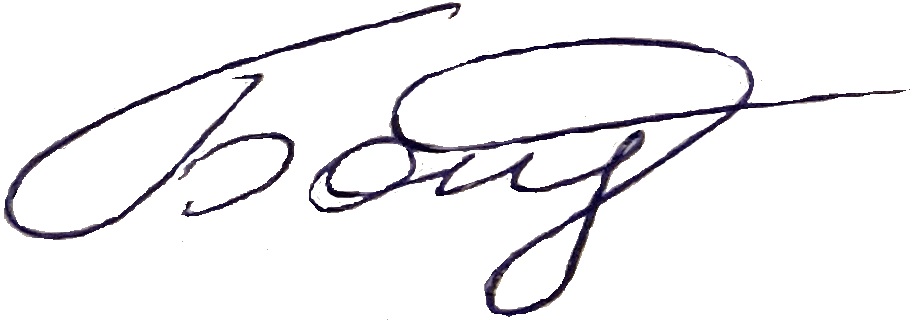
3.8 Составление полной спецификации устройства

3.9 Составление фрагмента функциональной схемы устройства

3.10 Контрольный пример



Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Л. Погудин

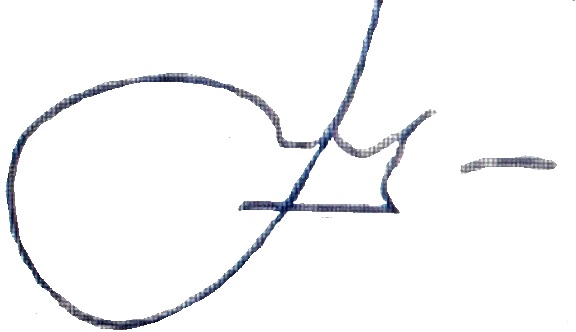


Задание получил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Э.Ю. Миннахметов

КАЛЕНДАРНЫЙ ГРАФИК ВЫПОЛНЕНИЯ

КУРСОВОЙ РАБОТЫ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  пп | Этапы работы | Объём  этапа, % | Сроки выполнения | | Примеча-ние |
| Начало | Конец |
| 1. | Исследование предметной области | 10 | 01.10.20 | 15.10.20. | выполнено |
| 2. | Устройство управления | 5 | 15.10.20 | 31.10.20 | выполнено |
| 3. | Сложение | 5 | 31.10.20 | 15.11.20 | выполнено |
| 4. | Вычитание | 5 | 15.11.20 | 30.11.20 | выполнено |
| 5. | Сравнение | 5 | 30.11.20 | 15.12.20 | выполнено |
| 6. | Логическое «И» | 5 | 15.12.20 | 30.12.20 | выполнено |
| 7. | Адресация | 5 | 01.10.20 | 15.10.20. | выполнено |
| 8. | Разработка устройства. | 5 | 15.10.20 | 31.10.20 | выполнено |
| 9. | Анализ исходных данных задания на курсовую работу | 5 | 31.10.20 | 15.11.20 | выполнено |
| 10. | Спецификация устройства на уровне «черного ящика» | 5 | 15.11.20 | 30.11.20 | выполнено |
| 11. | Представление черного ящика в виде операционной и управляющей частей | 5 | 30.11.20 | 15.12.20 | выполнено |
| 12. | Разработка структуры операционной части устройства | 5 | 01.10.20 | 15.10.20. | выполнено |
| 13. | Составление схемы алгоритма работы устройства и его микропрограммы | 5 | 15.10.20 | 31.10.20 | выполнено |
| 14. | Разработка схемы алгоритма работы | 5 | 31.10.20 | 15.11.20 | выполнено |
| 15. | Составление полной спецификации устройства | 5 | 15.11.20 | 30.11.20 | выполнено |
| 16. | Разработка фрагмента функциональной схемы вычитателя | 5 | 15.10.20 | 31.10.20 | выполнено |
| 17. | Контрольный пример | 5 | 31.10.20 | 15.11.20 | выполнено |
| 18. | Временная диаграмма работы УУ | 5 | 15.11.20 | 30.12.20 | выполнено |
| 19. | Оформление курсовой работы | 5 | 30.11.20 | 30.12.20 | выполнено |



Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**(А.Л. Погудин)

«18» сентября 2020 года

**Реферат**

Отчет 25 с., 12 рис., 1 табл., 3 источника.

Арифметико-логическое устройство (АЛУ), управляющее устройство (УУ), суммирование, вычитание, сравнение, поразрядное логическое «и», косвенная адресация памяти.

Цель работы – разработка алгоритма работы и структуры работы устройства для выполнения четырех команд.

При разработке устройства использовались концепции «черного ящика», т.е. первоначальное определение общих функций устройства и системы входных и выходных сигналов. В основе дальнейшей работы с «черным ящиком» использовался принцип декомпозиции, т.е. последовательное разложение функций на подфункции до получения описания функций на элементарном уровне.

В результате работы была составлен алгоритм работы и структура устройства.

Приведен контрольный пример в числовой форме.

**Содержание**

Перечень используемых условных обозначений, сокращений и терминов 6

Введение 7

1. Исследование предметной области 8

1.1 Устройство управления 8

1.2 Сложение 8

1.3 Вычитание 9

1.4 Сравнение 9

1.5 Логическое «И» 9

1.6 Адресация 9

1. Разработка устройства 11

2.1 Анализ исходных данных задания на курсовую работу 11

2.2Спецификация устройства на уровне «черного ящика» 11

2.3 Представление черного ящика в виде операционной и управляющей частей 11

2.4 Разработка структуры операционной части устройства 12

2.5 Составление схемы алгоритма работы устройства и его микропрограммы 14

2.6 Разработка схемы алгоритма работы 15

2.7 Составление полной спецификации устройства 19

2.8 Разработка фрагмента функциональной схемы вычитателя 19

2.9 Контрольный пример 21

2.10 Временная диаграмма работы УУ 23

Заключение 24

Список литературы 25

**Перечень используемых условных обозначений, сокращений и терминов**

|  |  |
| --- | --- |
| ЧЯ | Черный ящик |
| УЧ | Управляющая часть устройства |
| ОЧ | Операционная часть устройства |
| АЛУ | Арифметико-логическое устройство |
| МО | Микрооперация |
| МПР | Микропрограмма |

**Введение**

Цель работы (проекта) – Разработать УУ выполнением четырёх команд (сложения, вычитания, сравнения и поразрядного логического «и» (конъюнкции)). Формат команды КОП А1, А2; разрядность операндов и результата - 8 бит, разрядность А1 и А2 = 3. Использовать следующие способы размещения операндов: регистр - косвенная регистровая адресация памяти, косвенная регистровая адресация памяти - регистр. Результат операции сохранять по адресу А1.Объектом исследования является устройство управления. Предметом исследования – алгоритм работы и структура устройства.

В работе представлена спецификация устройства на уровне «черного ящика», разработана схема алгоритма работы устройства и его микропрограммы, составлена полная спецификация устройства. Построена временная диаграмма работы устройства управления. Приведен листинг разработанной программы на языке программирования С++ и результаты расчета контрольного примера.

**1 Исследование предметной области курсовой работы**

**1.1 Устройство управления**

Управляющее устройство (УУ) - устройство управления, часть вычислительной машины (ВМ), координирующая работу всех её устройств, предписывая им те или иные действия в соответствии с заданной программой. Управляющее устройство вырабатывает управляющие сигналы, обеспечивающие требуемую последовательность выполнения операций, контролирует работу машины в различных режимах, обеспечивает взаимодействие человека-оператора с ВМ.

Для выполнения своих функций УУ должно иметь входы, позволяющие определить состояние управляемой системы, и выходы, через которые реализуется управление поведением системы.

Входная информация:

Тактовые импульсы – с каждым тактовым импульсом УУ инициирует выполнение одной или нескольких микроопераций.

Код операции – код операции текущей команды поступает из регистра команды и используется, чтобы определить, какие микрооперации должны выполняться в течение машинного цикла.

Сигналы из системной шины– часть сигналов с системной шины, обеспечивающая передачу в управляющее устройства запросов прерывания, подтверждений и т.д.

В свою очередь УУ, а точнее микропрограммный автомат, формирует следующую выходную информацию:

Внутренние сигналы управления– эти сигналы воздействуют на внутренние схемы центрального процессора и относятся к одному из двух типов: тем, которые вызывают перемещение данных из регистра в регистр, и тем, что инициируют определенные функции операционного устройства ВМ.

Сигналы в системную шину– также относится к одному из двух типов: управляющие сигналы в память и управляющие сигналы в модули ввода/вывода.

После извлечения команды из памяти, она загружается в регистр команд. Дешифратор команд, входящий в устройство управления, преобразует код команды в управляющие сигналы:

* внутренние, необходимые для считывания/записи данных в регистры и управления АЛУ;
* внешние, подаваемые на шину управления.

Арифметическо - логическое устройство (АЛУ)— блок процессора, который служит для выполнения арифметических и логических преобразований над словами, называемыми в этом случае операндами.

АЛУ в зависимости от выполнения функций можно разделить на две части:

2)операционное устройство (АЛУ), в котором реализуется заданная последовательность микрокоманд (команд).

В АЛУ выполняются требуемые операции:

* сложение
* вычитание
* сравнение
* логическое «и»

**1.2 Сложение**

Результат размещается по адресу первого операнда.

Для сложения первого операнда с отрицательным числом на Ассемблере используется инвертор.

**1.3 Вычитание**

Результат размещается по адресу первого операнда.

Приемник не может быть адресован непосредственно, оба операнда одновременно не могут быть в памяти.

**1.4 Сравнение**

Сравнивается 1 и 2 операнд (А и В).

Если первый операнд меньше второго, то выводится 2 (записывается в реестр 1 операнда), если больше, то выводится 1 (записывается в реестр 1 операнда).

if (a2 < b) a3 = b; else a3 = a2

**1.5 Логическое «и»**

Если выполняется условие (первый операнд меньше 0), то выводится правда, если нет, то ложь.

boolr; // создаем переменную bool типа

r = (a> 0) && (b>0);

cout<< "r = " <<r<<endl; // вывод результата

**1.6 Адресация**

Операция косвенной адресации также является **унарной**, обозначается знаком \* и ее операндом может быть только **адресное выражение**, т.е. выражение, значением которого является адрес памяти. Операция косвенной адресации записывается в виде:

\*<адресное выражение>

В результате выполнения операции косвенной адресации происходит обращение к памяти по адресу, определяемому значением адресного выражения, и из этого адреса извлекается данное либо по этому адресу посылается данное в зависимости от контекста использования операции.

Извлечь данное из регистра с помощью косвенной адресации.

Например, если мы рассмотрим фрагмент программы:

**int a,b,\*ba;**

**a=5; ba=&a;**

**a=\*ba;**

то в результате его выполнения переменная **a** примет значение **5**. В последнем операторе a=\*ba; по адресу, записанному в указателе ba, извлекается значение переменной a, т,е. числo 5, и присваивается переменной a. Значение извлекается не прямым обращением к переменной путем использования ее имени – a, а косвенно, путем использования ее адреса, имеющегося в указателе ba.

Выражение \*ba записано справа от знака присваивания и является правоопределенным (*r-value, rightvalue*) вариантом этого выражения. Как в указатель ba попал адрес переменной a – понятно, в результате выполнения предыдущего оператора ba=&a;.

Таким образом, в этом контексте операция косвенной адресации трактуется как ***"обратиться к данному по адресу, извлечь данное».***

Для записи данного в регистр с помощью косвенной адресации:

**int a,b,\*ba;**

**a=5; ba=&a;**

**\*ba=a1;**

операция косвенной адресации использована слева от операции присваивания.

Это случай так называемого левоопределенного выражения (*l-value, left value*) вида:**\*<адресное\_выражение>**. Теперь левосторонняя операция косвенной адресации с учетом операции присваивания трактуется как***"записать данное по адресу"***, и в результате выполнения этого фрагмента программы переменная a примет значение 5, как и в предыдущем случае.

Разница заключается в том, что мы присвоили ей значение путем косвенной адресации через переменную–указатель, содержащую ее адрес. Таким образом, мы установили, что операция косвенной адресации может быть использована как для извлечения значения объекта через его адрес, так и для присвоения значения через адрес объекта.

**2 Разработка устройства**

**2.1 Анализ исходных данных задания на курсовую работу**

Согласно заданию, устройство должно быть предназначено для выполнения следующих операций:

* сложение;
* вычитание;
* сравнение;
* логическое «И».

Разрядность операндов и результата должна быть – 8 бит.

Исходя из этого видно, что входы(А,В) и выход (результат операции С) должны иметь 8 разрядов.

**2.2 Спецификация устройства на уровне «черного ящика»**

Упрощенно разрабатываемое устройство можно представить как последовательное объединение 4 команд изображенных на рис.1-4.

8

8

A

С

Сумматор

8

B

Рисунок П1. –Система выводов сумматора

8

8

A

С

8

B

Вычитатель

Рисунок П2. –Система выводов вычитателя

8

8

A

С

Устройство сравнения

8

B

Рисунок П3. –Система выводов устройства сравнения

8

8

A

С

Устройство логическое «и»

8

B

Рисунок П4. –Система выводов устройства логическое «и»

**2.3 Представление «черного ящика» в виде операционной и управляющей частей**

Пусть операнды размещаются в регистрах А и В, как показано на рисунке 5. Выделим четыре блока для каждой команды. После каждого выполнения команды значение операнда А изменяется, а значение операнда В остается неизменным. Результат выполнения четырех команд (изменения операнда А) записывается в регистр С.

Сигналы

Управления

к

8

A

Операнды

к

k

Сумматор

8

В

8

ГТИ

m

ОЧ

УЧ

l

С

Осведомительные сигналы

Операнды

к

8

В

8

Вычитатель

С

8

8

8

Операнды

к

Устройство сравнения

В

С

8

8

8

Операнды

к

В

С

Устройство логическое «и»

8

Рисунок П5.—Разбиение схемы устройств на ОЧ и УЧ.

**2.4 Разработка структуры операционной части устройства**

Пусть операнды размещаются в регистре А (уменьшаемое) и в регистре В (вычитаемое). Алгоритм приведен на рисунке 6.

Алгоритм не учитывает микрооперации (в пользовательском интерфейсе можно добавить дополнительные условия: «выполнить операцию суммирования?», «выполнить операцию вычитания?», «выполнить операцию сравнения?», «выполнить операцию логического и?»).

Регистр А

Регистр В

А

В

Блок обработки

сумматор

С (результат А1)

Регистр С

Регистр А

Регистр В

С

В

Блок обработки

вычитатель

С (результат А2)

Регистр С

Регистр А

Регистр В

С

В

Блок обработки

Устройство сравнения

С (результат А3)

Регистр С

Регистр А

Регистр В

С

В

Блок обработки

Устройство логическое «и»

True or false

Регистр С

Рисунок П6.—Упрощенная структура ОЧ четырёх команд.

**2.5 Составление схемы алгоритма работы устройства и его микропрограммы**

Схема алгоритма выполнения четырёх команд (сложения, вычитания, сравнения и поразрядного логического «и» (конъюнкции)) приведена на рисунке П7.

Отметим операторные блоки символами Yi, а логические блоки символами Xi.

Операторных блоков получилось 17, логических – 6.

Таким образом, для операционной части (ОЧ) устройства потребуется 17 сигналов управления из управляющей части устройства, а для управляющей части (УЧ) устройства – 6 осведомительных сигналов из операционной части

**2.6 Разработка схемы алгоритма работы**

Схема алгоритма на уровне микроопераций изображена на рисунке П8.

А, В

В<0

равны?

нет

да

А+В

A+B

C запись результата (A1) в регистр А

да

В<0

равны?

нет

А-В

А-В

C запись результата (A2) в регистр А

да

А<B

равны?

нет

В

А

C запись результата (A3) в регистр А

да

А>0 and B>0

равны?

нет

true

false

C запись результата (true or false) в регистр C

Рисунок П7–-Схема алгоритма работы четырёх команд

**Y1**

А, В

**Y2**

Запись значения A в регистр А

**Y3**

Чтение значенияAиз регистра А

**Y4**

В<0

равны?

нет

да

**Y5**

**Y6**

А+В

A+B

**Y7**

C, запись результата (A1) в регистр А

**Y8**

Чтение значения A из регистра А

**Y9**

да

В<0

равны?

нет

**Y11**

**Y10**

А-В

А-В

**Y12**

C, запись результата (A2) в регистр А

**Y13**

Чтение значенияAиз регистра А

**Y14**

да

А<B

равны?

нет

**Y15**

**Y16**

В

А

**Y17**

C, запись результата (A3) в регистр А

**Х1**

Чтение значенияAиз регистра А

**Х2**

да

А>0 and B>0

равны?

нет

**Х3**

**Х4**

true

false

**Х5**

C запись результата (true or false) в регистр C

**Х6**

Рисунок П8–-Схема алгоритма работы четырёх команд на уровне микроопераций

**2.7 Составление полной спецификации устройства**

Опишем в таблице все линии и сигналы, полученные в процессе разработки вычитателя. В данном примере курсовой работы , конечно, приведены не все строки этой таблицы).

Таблица Сигналы вычитателя

|  |  |  |
| --- | --- | --- |
| Имя сигнала/шины и разрядность | Тип  (In/Out) | Назначение сигнала |
| Y8 | I для ОЧ | Сигнал на чтение значения A из регистра А |
| Y9 | I для ОЧ | Сигнал о выполнении или не выполнении условия В<0 |
| ……….. | … | …….. |
| Y10 | I для ОЧ | Сигнал управления из УЧ на выполнение вычитания |
| Y11 | I для ОЧ | Сигнал управления из УЧ на выполнение вычитания |
| ………. | … | …….. |
| Y12 | I для ОЧ | Сигнал о записи результата (A2) в регистр А |
| ………… | … | …….. |

Структура вычитателя (фрагмента программы) на микропрограммном уровне управления приведена на рисунке П9.

Сигналы

Управления

Y[1:16]

Операнды

А[ 0:8]

ГТИ

УЧ

ОЧ

У[1:17]

ОЧ

вычитателя

В[0:8]]]

l

А-В[0:8]

Рисунок П9. - Структура вычитателя на микропрограммном уровне управления

**2.8 Разработка фрагмента функциональной схемы вычитателя**

Фрагмент схемы УЧ дан на рисунке П10. Схема составлена с блока Y8 по блок Y12 (см. рисунок П5).

Y8

С1

Счетчик

тактов

Дешифра-тор

тактов

С2

Y9

из ГТИ

**……….**

Y10

Y11

Y12

**…………………**

С5

Рисунок П10. - Фрагмент схемы УЧ

**2.9 Контрольный пример**

Исходные данные:

A = 1

B = -3

Выполнение программы для исходных данных показано на рисунке П11.

Листинг программы

#include <iostream>

#include <math.h>

using namespace std;

void main()

{

cout.setf(ios::boolalpha);

bool r; // создаем переменную bool типа

int a, b, a1, a2,a3, b1,\*ba;

cout<< "vvedite a:" <<endl;

cin>> a;// ввода

cout<< "vvedite b:" <<endl;

cin>> b;// ввод b

ba = &a;

a = \*ba; //по адресу, записанному в указателе ba, извлекается значение переменной a

a1 = a + b; b1 = b; // сумма

\*ba = a1; // записать значение по адресу a1

a1 = \*ba; //по адресу, записанному в указателе ba, извлекается значение переменной a1

if (b1 < 0) a2 = a1 - b1; else a2 = a1 - b1; // разность

\*ba = a2; // записать значение по адресу a2

a2 = \*ba;//по адресу, записанному в указателе ba, извлекается значение переменной a2

if (a2 < b) a3 = b; else a3 = a2; // сравнение

\*ba = a3;// записать значение по адресу a3

a3 = \*ba;//по адресу, записанному в указателе ba, извлекается значение переменной a3

r = (a3 > 0) && (b >0); // логическое и

// вывод результатов

cout<< "" <<endl;

cout<< "Parametri:" <<endl;

cout<< "a:" <<endl;

cout<< a <<endl;

cout<< "b:" <<endl;

cout<< b <<endl;

cout<< "" <<endl;

cout<< " Rezult:" <<endl;

cout<< " a+b" <<endl;

cout<< a1 <<endl;

cout<< " -b" <<endl;

cout<< a2 <<endl;

cout<< " if a<b rezult b, else rezult a" <<endl;

cout<< a3 <<endl;

cout<< " if (a>0) and (b>0) rezult true, else rezult false" <<endl;

cout<< "r = " << r <<endl; // вывод результата

}

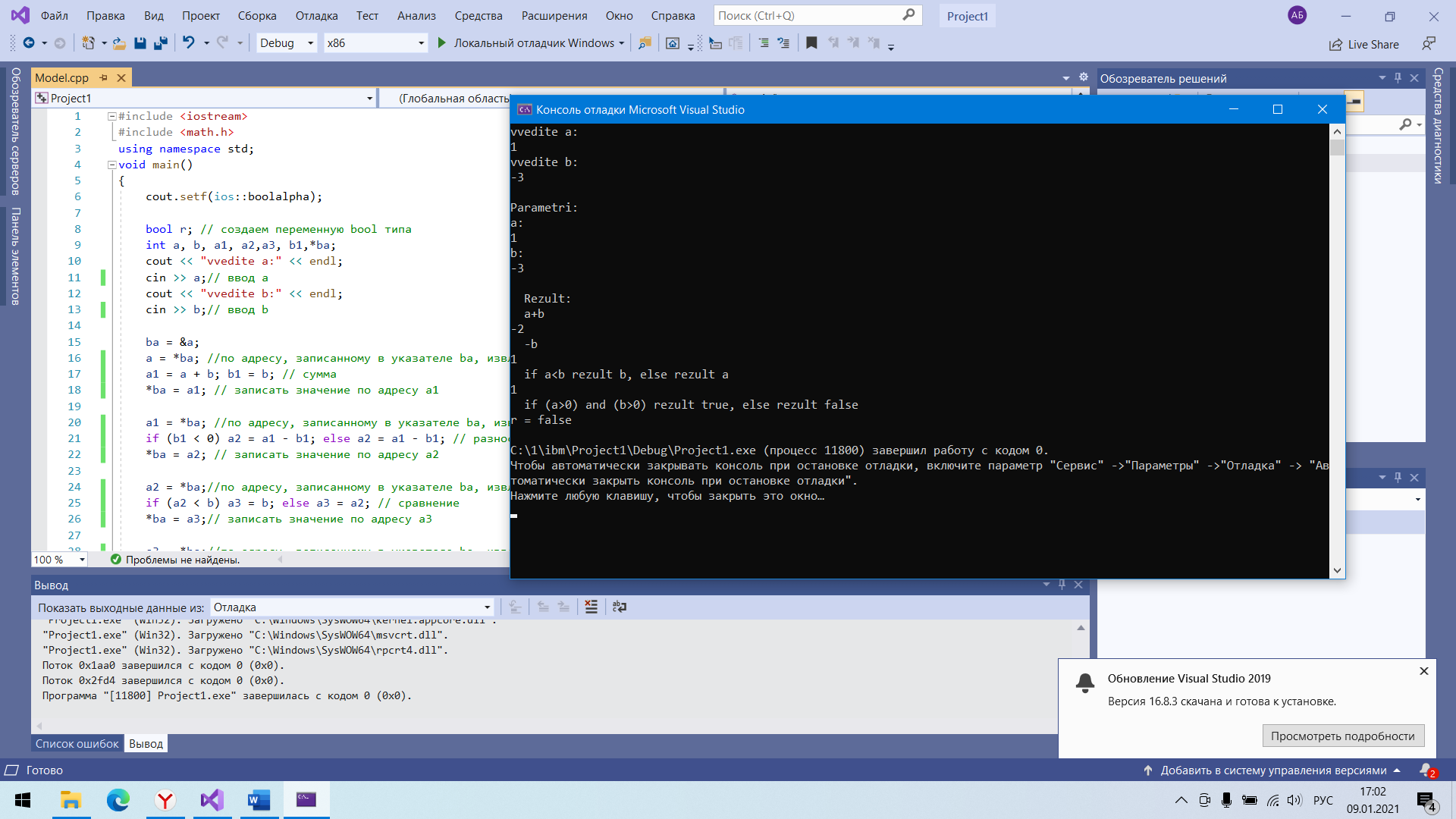


Рисунок П11. –Выполнение программы

**2.10 Временная диаграмма работы УУ**

На рисунке П12 приведен фрагмент временной диаграммы работы УУ (Схема составлена с блока Y8 по Y12).

ТАКТ

ГТИ**t**

С1

С2

С3

Y8

Y9

Т1=Y10

Т2=Y11

Y12

Рисунок П12.- Фрагмент временной диаграммы работы вычитателя

Закрашенные интервалы времени соответствуют логическим 1, горизонтальные штриховые линии указывают интервалы времени, в которых значения X не имеют смысла, т.к. в эти интервалы сигналы X не проверяются в УЧ. Вертикальные штриховые линии разделяют временные такты.

**Заключение**

Задача курсовой работы – разработка алгоритма работы и структуры работы устройства для выполнения четырех команд

Поставленная задача выполнена. В ходе курсовой работы была изучена специальная литература, разработана структура ОЧ, алгоритм их работы, спецификация сигналов, фрагмент функциональной схемы УЧ, контрольный числовой пример и временная диаграмма работы устройства.

**Список используемой литературы**

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня: Учеб. пособие. – СПб.:Питер, 2007. – 461 с.
2. Павловская Т.А., Щупак Ю.А. С/С++. Программирование на языке высокого уровня. Структурное программирование: Практикум. – СПб.:Питер, 2003. – 240 с.
3. Жмакин А. П. Архитектура ЭВМ: 2-е изд., перераб. и доп.: учеб. пособие. — СПб.: БХВ-Петербург, 2010. — 352 с.

**Приложение А**

Листинг файла main.c

#include "assoc.h"  
#include <locale.h>  
  
int main() {  
 setlocale(**LC\_ALL**, "Russian");  
 run();  
 return 0;  
}

Листинг файла assoc.h

#pragma once  
  
void run();  
  
void create(void \*\*mem);  
void out(void \*\*mem);  
void read(void \*\*mem);  
void write(void \*\*mem);  
void rem(void \*\*mem);  
void drop(void \*\*mem);

Листинг файла assoc.c

#include "assoc.h"  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
typedef unsigned char byte;  
typedef void (\*task)(void\*\*);  
  
struct bin {  
 char s[9];  
};  
  
const byte BYTE\_COUNT = 64;  
  
const char \*DIALOG =  
 "Меню:\n"  
 "\t1. Создать устройство ассоциативной памяти.\n"  
 "\t2. Вывести все записи.\n"  
 "\t3. Прочитать из памяти.\n"  
 "\t4. Записать в память.\n"  
 "\t5. Удалить из памяти.\n"  
 "\t6. Удалить устройство ассоциативной памяти.\n"  
 "\t(\*). Выход.\n"  
 "Введите номер действия: ";  
  
const char \*INCORRECT = "Некорректный индекс!";  
  
const task TASKS[] = {  
 &create,  
 &out,  
 &read,  
 &write,  
 &rem,  
 &drop  
};  
  
int read\_index(byte \*result);  
byte read\_value();  
struct bin byte\_to\_bin(byte b);  
byte bin\_to\_byte(struct bin b);  
  
void run() {  
 int number;  
 void \*mem = **NULL**;  
 while(1) {  
 printf("%s", DIALOG);  
 scanf("%d", &number);  
 if(1 <= number && number <= 6) {  
 if(number > 1 && mem == **NULL**) {  
 printf("Устройство ассоциативной памяти не было создано!\n\n");  
 continue;  
 } else {  
 TASKS[number - 1](&mem);  
 printf("\n");  
 }  
 } else {  
 break;  
 }  
 }  
}  
  
void create(void \*\*mem) {  
 if(\*mem) {  
 free(\*mem);  
 }  
 \*mem = malloc(BYTE\_COUNT);  
 for(byte i = 0; i < BYTE\_COUNT; ++i) {  
 ((byte\*)\*mem)[i] = 0;  
 }  
 printf("Устройство ассоциативной памяти создано.\n");  
}  
  
void out(void \*\*mem) {  
 byte count = 0;  
 printf("Содержимое памяти:\n");  
 for(byte i = 0; i < BYTE\_COUNT; ++i) {  
 byte value = ((byte\*)\*mem)[i];  
 if(value) {  
 printf("\t%s -> %c\n", byte\_to\_bin(i).s, value);  
 ++count;  
 }  
 }  
 if(count == 0) {  
 printf("\t<пусто>\n");  
 }  
 printf("Всего записей: %d\n", count);  
}  
  
void read(void \*\*mem) {  
 byte index;  
 if (read\_index(&index)) {  
 byte value = ((byte \*) \*mem)[index];  
 printf("%s -> %c\n", byte\_to\_bin(index).s, value);  
 } else {  
 printf("%s\n", INCORRECT);  
 }  
}  
  
void write(void \*\*mem) {  
 byte index;  
 if (read\_index(&index)) {  
 byte value = read\_value();  
 ((byte\*)\*mem)[index] = value;  
 printf("%s -> %c\n", byte\_to\_bin(index).s, value);  
 } else {  
 printf("%s\n", INCORRECT);  
 }  
}  
  
void rem(void \*\*mem) {  
 byte index;  
 if (read\_index(&index)) {  
 if(((byte\*)\*mem)[index]) {  
 ((byte\*)\*mem)[index] = 0;  
 printf("Байт по индексу %s обнулён.\n", byte\_to\_bin(index).s);  
 } else {  
 printf("Байт по индексу %s уже был обнулён.\n", byte\_to\_bin(index).s);  
 }  
 } else {  
 printf("%s\n", INCORRECT);  
 }  
}  
  
void drop(void \*\*mem) {  
 free(\*mem);  
 \*mem = **NULL**;  
 printf("Устройство ассоциативной памяти удалено.\n");  
}  
  
int read\_index(byte \*result) {  
 struct bin b;  
 printf("Введите индекс: ");  
 scanf("%s", b.s);  
 \*result = bin\_to\_byte(b);  
 getchar();  
 return \*result < 64;  
}  
  
byte read\_value() {  
 byte value;  
 printf("Введите значение: ");  
 scanf("\n%c", &value);  
 return value;  
}  
  
struct bin byte\_to\_bin(byte b) {  
 struct bin r;  
 r.s[8] = '\0';  
 for(int i = 0; i < 8; ++i) {  
 r.s[i] = (b & 128) ? '1' : '0';  
 b <<= 1;  
 }  
 return r;  
}  
  
byte bin\_to\_byte(struct bin b) {  
 byte r = 0;  
 for(byte i = 0, n = strlen(b.s); i < n; ++i) {  
 r <<= 1;  
 r |= (b.s[i] != '0' ? 1 : 0);  
 }  
 return r;  
}