

Филиал федерального государственного бюджетного образовательного
учреждения высшего образования
«Саратовский государственный технический университет имени
Гагарина Ю.А.» в г. Петровске

УТВЕРЖДАЮ
Директор филиала СГТУ
имени Гагарина Ю.А. в г.Петровске
Е.А.Бесшапошникова
«30» июня 2025 г.

специальности
15.02.10 «Мехатроника и робототехника (по отраслям)»

Методические указания рассмотрены
на заседании предметной (цикловой)
комиссии общепрофессиональных дисциплин и
профессиональных модулей
«16» июня 2025 года, протокол № 13

Председатель ПЦК Табарова /Ю.А. Табарова/

Петровск 2025

Пояснительная записка

Методические указания по выполнению практических работ подготовлены на основе рабочей программы учебной дисциплины «Основы вычислительной техники», разработанной на основе ФГОС СПО по специальности 15.02.10 «Мехатроника и робототехника (по отраслям)» и соответствующих общих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам.

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях.

Целью освоения учебной дисциплины «Основы вычислительной техники» является:

- теоретическая и практическая подготовка студентов в области информационных технологий в такой степени, чтобы они могли выбирать необходимые технические, алгоритмические, программные и технологические решения, уметь объяснить принципы их функционирования и правильно их использовать;
- формирование у студентов знаний по дисциплине, достаточных для самостоятельного освоения вычислительных систем с новыми архитектурами.

При выполнении практических работ студент должен **уметь**:

- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- определять этапы решения задачи;
- выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы;
- владеть актуальными методами работы в профессиональной и смежных сферах;
- оценивать результат и последствия своих действий (самостоятельно или с помощью наставника);
- определять задачи для поиска информации;
- определять необходимые источники информации;
- планировать процесс поиска; структурировать получаемую информацию
- выделять наиболее значимое в перечне информации;
- оценивать практическую значимость результатов поиска;

- оформлять результаты поиска, применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- использовать различные цифровые средства для решения профессиональных задач;
- применять современную научную профессиональную терминологию;
- определять и выстраивать траектории профессионального развития и самообразования.

При выполнении практических работ студент должен **знать**:

- актуальный профессиональный и социальный контекст, в котором приходится работать и жить;
- основные источники информации и ресурсы для решения задач и проблем в профессиональном и/или социальном контексте;
- алгоритмы выполнения работ в профессиональной и смежных областях;
- методы работы в профессиональной и смежных сферах;
- структуру плана для решения задач;
- номенклатура информационных источников, применяемых в профессиональной деятельности;
- приемы структурирования информации;
- формат оформления результатов поиска информации, современные средства и устройства информатизации;
- порядок их применения и программное обеспечение в профессиональной деятельности в том числе с использованием цифровых средств;
- современная научная и профессиональная терминология;
- возможные траектории профессионального развития и самообразования.

Содержание практических занятий определено рабочей программой и тематическим планированием, соответствует теоретическому материалу изучаемых разделов учебной дисциплины.

Объём практических занятий по дисциплине определяется учебным планом по данной специальности.

Продолжительность практического занятия - 2 академических часа. Перед проведением практического занятия преподавателем организуется инструктаж, а по ее окончании – обсуждение итогов.

Комплект методических указаний по выполнению практических работ дисциплины «Основы вычислительной техники» содержит 44 практических занятий.

**Перечень практических работ
по дисциплине «Основы вычислительной техники»**

ПРАКТИЧЕСКАЯ РАБОТА № 1.

Тема: «Алгебра логики».

ПРАКТИЧЕСКАЯ РАБОТА № 3-4.

Тема: «Алгебра логики».

ПРАКТИЧЕСКАЯ РАБОТА № 5.

Тема: «Минимизация логических схем».

ПРАКТИЧЕСКАЯ РАБОТА № 6.

Тема: «Минимизация логических схем».

ПРАКТИЧЕСКАЯ РАБОТА № 7.

Тема: «Минимизация логических схем».

ПРАКТИЧЕСКАЯ РАБОТА № 8-9.

Тема: «Минимизация логических схем».

ПРАКТИЧЕСКАЯ РАБОТА № 10-13.

Тема: «RS-триггер».

ПРАКТИЧЕСКАЯ РАБОТА № 14-16.

Тема: «D-триггер».

ПРАКТИЧЕСКАЯ РАБОТА № 17-19.

Тема: «JK-триггер».

ПРАКТИЧЕСКАЯ РАБОТА № 20-27.

Тема: T-триггер.

ПРАКТИЧЕСКАЯ РАБОТА № 28-33.

Тема: T-триггер.

ПРАКТИЧЕСКАЯ РАБОТА № 34.

Тема: «Шифраторы».

ПРАКТИЧЕСКАЯ РАБОТА № 35.

Тема: «Шифраторы».

ПРАКТИЧЕСКАЯ РАБОТА № 36.

Тема: «Дешифраторы».

ПРАКТИЧЕСКАЯ РАБОТА № 37.

Тема: «Дешифраторы».

ПРАКТИЧЕСКАЯ РАБОТА № 38.

Тема: Полусумматоры.

ПРАКТИЧЕСКАЯ РАБОТА № 39.

Тема: Полусумматоры.

ПРАКТИЧЕСКАЯ РАБОТА № 40.

Тема: Сумматоры.

ПРАКТИЧЕСКАЯ РАБОТА № 41.

Тема: Сумматоры.

ПРАКТИЧЕСКАЯ РАБОТА № 42.

Тема: «Преобразование и передача данных».

ПРАКТИЧЕСКАЯ РАБОТА № 43.

Тема: «Преобразование и передача данных».
ПРАКТИЧЕСКАЯ РАБОТА № 44.
Тема: «Преобразование и передача данных».

ИНСТРУКЦИИ ДЛЯ ОБУЧАЮЩИХСЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Прежде чем приступить к выполнению заданий, внимательно прочитайте данные рекомендации. Практические работы включают в себя задания следующих видов:

Правила выполнения практических работ за компьютером.

1. Студент должен выполнить практическую работу самостоятельно (или в группе, если это предусмотрено заданием).
2. Если студент не выполнил практическую работу или часть работы за отведенное время, то он может выполнить работу или оставшуюся часть во внеурочное время, согласованное с преподавателем.
3. Каждый студент после окончания урока, должен представить преподавателю выполненную работу в электронном виде с анализом полученных результатов и выводом по работе.
4. Дифференцированную оценку по практической работе студент получает, с учетом срока выполнения работы, если:
 - работа выполнена правильно и в полном объеме;
 - сделан анализ проделанной работы и вывод по результатам работы;
 - студент может пояснить выполнение любого этапа работы.

Зачет по практическим работам студент получает при условии выполнения всех предусмотренной программой работ, после сдачи отчетов по работам при удовлетворительных оценках за опросы и контрольные вопросы во время практических занятий.

Инструкция по технике безопасности при выполнении практических работ студентами.

Запрещается:

1. Трогать разъёмы соединительных кабелей.
2. Прислоняться к экрану и тыльной стороне монитора.
3. Включать и выключать ЭВМ без разрешения преподавателя.
4. Прислоняться к проводам и устройствам заземления.
5. При обнаружении запаха гари немедленно остановить работу, выключить клавиатуру и сообщить преподавателю.

Перед началом работы:

1. Убедиться в отсутствии видимых повреждений рабочего места.
2. Запрещается работать во влажной одежде (и вообще в верхней одежде) и с влажными руками.
3. На рабочем месте размещать тетрадь и учебные пособия так, чтобы они не мешали работе.

Во время работы:

1. Работать 60-80 см на расстоянии от ЭВМ.
2. Строго выполняйте вышеуказанные правила.
3. Следить за исправностью аппаратуры.
4. Немедленно прекратить работу при появлении постороннего звука и сообщить преподавателю.
5. Работать на клавиатуре чистыми руками, правильно нажимать на клавиши.
6. Никогда не пытаться самим устранить неисправность при работе с аппаратурой.
7. Не вставать со своих мест, когда входит посетитель.

По окончании работы:

1. Отключить ЭВМ, навести порядок на рабочем месте.
2. Сдать рабочее место преподавателю.

Выполнение практических работ.

Студент должен:

- выполнять требования по охране труда;
- соблюдать инструкцию по правилам и мерам безопасности в учебном кабинете.
- строго выполнять весь объем работы, указанный в задании;
- соблюдать требования эксплуатации компьютерной техники (правила включения и выключения).
- предоставить отчет о проделанной работе по окончании выполненной работы, который должен содержать: название работы, цель работы, задание и его решение, вывод о проделанной работе.

Отчет о проделанной работе может быть выполнен на компьютере или в тетрадях для практических работ.

Требования к отчету по практическим работам, выполненным на компьютере.

Текст отчета по практической работе должен быть набран на компьютере шрифтом TimesNewRoman размером 14 пт. (при оформлении текста используется текстовый редактор MS Word). Шрифт, используемый в иллюстративном материале (таблицы и рисунки), рекомендуется уменьшить до 12 пт. Межстрочный интервал в основном тексте - полуторный. В иллюстративном материале межстрочный интервал рекомендуется сделать

одинарным. Поля страницы должны быть: левое поле - 30 мм; правое поле – 1,5 мм; верхнее и нижнее поле - 20 мм.

Каждый абзац должен начинаться с красной строки. Отступ абзаца – 1,25 мм от левой границы текста.

Требования к отчету по практическим работам, выполненным в тетради.

1. В тетради для выполнения отчета по практическим занятиям пишется: «Практическое занятие №...»
2. Под надписью «Практическое занятие №...» укажите тему.
3. Ниже напишите: «Цель занятия».
4. Под надписью «Цель занятия» в центре укажите: «Вариант №...». Поставьте номер своего варианта.
5. Оформите порядок выполнения практической части занятия, опираясь задание.
6. Напишите вывод по занятию.

Студент должен выполнить практическую работу самостоятельно (или в группе, если это предусмотрено заданием). Практическая работа выполняется согласно заданию и методическим рекомендациям. После выполнения практической работы обучающийся самостоятельно себя контролирует путем ответов на вопросы. Результат работы представляется преподавателю в виде файла (файлов) в личном каталоге, защищается обучающимися.

По ходу выполнения работы при возникновении вопросов обучающийся может получить консультацию у преподавателя или самостоятельно воспользоваться лекционным материалом, рекомендуемой литературой.

ПРАКТИЧЕСКАЯ РАБОТА № 1-2

Тема: «Алгебра логики»

Цель: научиться выполнять перевод чисел в разные системы счисления.

Оборудование: справочные пособия.

Справочный материал

1. Основные понятия систем счисления

Система счисления — это система записи чисел с помощью определенного набора цифр. Количество цифр, необходимых для записи числа в системе, называют основанием системы счисления. Основание системы записывается в справа числа в нижнем индексе: $AF178_{16}$.

Различают два типа систем счисления: 5_{10} ; 1110110_2 ;

— позиционные, когда значение каждой цифры числа определяется ее позицией в записи числа;

– непозиционные, когда значение цифры в числе не зависит от ее места в записи числа. Примером непозиционной системы счисления является римская: числа IX, IV, XV и т.д.

Примером позиционной системы счисления является десятичная система, используемая повседневно.

Десятичная система счисления – в настоящее время наиболее известная и используемая. Десятичная система использует десять цифр — 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9, а также символы “+” и “–” для обозначения знака числа и запятую или точку для разделения целой и дробной частей числа.

В вычислительных машинах используется двоичная система счисления, её основание — число 2. Для записи чисел в этой системе используют только две цифры — 0 и 1.

Таблица 1. Соответствие чисел, записанных в различных системах счисления

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
1	001	1	1
2	010	2	2
3	011	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

2. **Правила перевода чисел из одной системы счисления в другую**

Перевод чисел из одной системы счисления в другую составляет важную часть машинной арифметики. Рассмотрим основные правила перевода.

1. Для перевода двоичного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 2, и вычислить по правилам десятичной арифметики.

При переводе удобно пользоваться таблицей степеней двойки:

Таблица 2. Степени числа 2

n	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

2. Для перевода восьмеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 8, и вычислить по правилам десятичной арифметики.

При переводе удобно пользоваться таблицей степеней восьмерки:

Таблица 3. Степени числа 8

n	0	1	2	3	4	5	6
8^n	1	8	64	512	4096	32768	262144

Пример. Число 750138 перевести в десятичную систему счисления.

$$75013_8 = 7 \cdot 8^4 + 5 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 3 \cdot 8^0 = 31243_{10}$$

3. Для перевода шестнадцатеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 16, и вычислить по правилам десятичной арифметики.

При переводе удобно пользоваться таблицей степеней числа 16:

Таблица 4. Степени числа 16

n	0	1	2	3	4	5	6
16^n	1	16	256	4096	65536	1048576	16777216

4. Для перевода десятичного числа в двоичную систему его необходимо последовательно делить на 2 до тех пор, пока не останется остаток, меньший или равный 1. Число в двоичной системе записывается как последовательность последнего результата деления и остатков от деления в обратном порядке.

Пример. Число 2210 перевести в двоичную систему счисления.

$$2210 = 101102$$

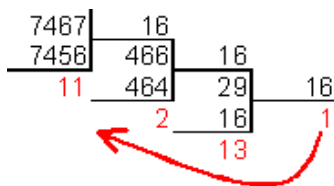
Для перевода десятичного числа в восьмеричную систему его необходимо последовательно делить на 8 до тех пор, пока не останется остаток, меньший или равный 7. Число в восьмеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

Пример. Число 57110 перевести в восьмеричную систему счисления.

$$571_{10} = 1073_8$$

1. Для перевода десятичного числа в шестнадцатеричную систему его необходимо последовательно делить на 16 до тех пор, пока не останется

остаток, меньший или равный 15. Число в шестнадцатеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.



Пример. Число 7467₁₀ перевести в шестнадцатеричную систему счисления.

$$7467_{10} = 1D2B_{16}$$

2. Чтобы перевести число из двоичной системы в восьмеричную, его нужно разбить на триады (тройки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую триаду нулями, и каждую триаду заменить соответствующей восьмеричной цифрой (табл. 3).

Пример. Число 100101₂ перевести в восьмеричную систему счисления.

$$001\ 001\ 011_2 = 113_8$$

3. Чтобы перевести число из двоичной системы в шестнадцатеричную, его нужно разбить на тетрады (четверки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую тетраду нулями, и каждую тетраду заменить соответствующей шестнадцатеричной цифрой (табл. 3).

Пример. Число 101110001₂ перевести в шестнадцатеричную систему счисления.

$$0010\ 1110\ 0011_2 = 2E3_{16}$$

4. Для перевода восьмеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной триадой.

Пример. Число 531₈ перевести в двоичную систему счисления.

$$531_8 = 101011001_2$$

5. Для перевода шестнадцатеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной тетрадой.

Пример. Число EE8₁₆ перевести в двоичную систему счисления.

$$EE8_{16} = 111011101000_2$$

6. При переходе из восьмеричной системы счисления в шестнадцатеричную и обратно, необходим промежуточный перевод чисел в двоичную систему.

Пример 1. Число FEA₁₆ перевести в восьмеричную систему счисления.

$$FEA_{16} = 111111101010_2$$

$$111\ 111\ 101\ 010_2 = 7752_8$$

Пример 2. Число 6653₈ перевести в шестнадцатеричную систему счисления.

$$6653_8 = 110110101011_2$$

$$1101\ 1010\ 1011_2 = DAB_{16}$$

3. *Арифметические действия над целыми числами в 2-ой системе*

счисления:

- 3 Операция сложения выполняется с использованием таблицы двоичного сложения в одном разряде:

Пример.

$$\begin{array}{r} \text{а) } \square 1001_2 \\ 1010_2 \\ 10011_2 \end{array} \quad \begin{array}{r} \text{б) } \square 1101_2 \\ 1011_2 \\ 11000_2 \end{array} \quad \begin{array}{r} \text{в) } \square 11111_2 \\ 1_2 \\ 100000_2 \end{array}$$

- 4 Операция вычитания выполняется с использованием таблицы вычитания, в которой 1 обозначается заем в старшем разряде.

Пример.

$$\begin{array}{r} \text{а) } -101110011_2 \\ 100011011_2 \\ 001011000_2 \end{array} \quad \begin{array}{r} \text{б) } 110101101_2 \\ 101011111_2 \\ 001001110_2 \end{array}$$

- 5 Операция умножения выполняется по обычной схеме, применяемой в десятичной с/с с последовательным умножением множимого на очередную цифру множителя.

Пример.

$$\begin{array}{r} \text{а) } \square 11001_2 \\ 1101_2 \overline{) 11001} \\ 11001 \\ 11001 \\ 11001 \\ 101000101_2 \end{array} \quad \begin{array}{r} \text{б) } \square 101_2 \\ 11_2 \overline{) 101} \\ 101 \\ 1111_2 \end{array}$$

- 6 Операция деления выполняется по алгоритму, подобному алгоритму выполнения операции деления в 10-ой с/с.

Сложение и вычитание в восьмеричной системе счисления.

При выполнении сложения и вычитания в 8-ой с/с необходимо соблюдать следующие правила:

- 1) в записи результатов сложения и вычитания могут быть использованы только цифры восьмеричного алфавита;
- 2) десяток восьмеричной системы счисления равен 8, т.е. переполнение разряда наступает, когда результат сложения больше или равен 8. В этом случае для записи результата надо вычесть 8, записать остаток, а к старшему разряду прибавить единицу переполнения;
- 3) если при вычитании приходится занимать единицу в старшем разряде, эта единица переносится в младший разряд в виде восьми единиц.

Пример

$$\begin{array}{r} + 770_8 \\ 236_8 \\ \hline 1226_8 \end{array} \quad \begin{array}{r} + 750_8 \\ 236_8 \\ \hline 512_8 \end{array}$$

Сложение и вычитание в шестнадцатеричной системе счисления.

При выполнении этих действий в 16–ой с/с необходимо соблюдать следующие правила:

- 1) при записи результатов сложения и вычитания надо использовать цифры шестнадцатеричного алфавита: цифры, обозначающие числа от 10 до 15 записываются латинскими буквами, поэтому, если результат является числом из этого промежутка, его надо записывать соответствующей латинской буквой;
- 2) десяток шестнадцатеричной системы счисления равен 16, т.е. переполнение разряда поступает, если результат сложения больше или равен 16, и в этом случае для записи результата надо вычесть 16, записать остаток, а к старшему разряду прибавить единицу переполнения;
- 3) если приходится занимать единицу в старшем разряде, эта единица переносится в младший разряд в виде шестнадцати единиц.

Пример

$$\begin{array}{r} + B09_{16} \\ TFA_{16} \\ 1A03_{16} \end{array} \quad \begin{array}{r} + B09_{16} \\ 7FA_{16} \\ 30F_{16} \end{array}$$

Содержание работы

Задание

1. Выполнить перевод чисел

- а) из 10–ой с/с в 2–ую систему счисления: 165; 541; 600; 720; 43,15; 234,99.
- б) из 2–ой в 10–ую систему счисления: 110101₂; 11011101₂; 110001011₂; 1001001,111₂ в) из 2–ой с/с в 8–ую, 16–ую с/с:
100101110₂; 100000111₂; 111001011₂; 1011001011₂; 110011001011₂; 10101,10101₂; 111,011₂
- г) из 10–ой с/с в 8–ую, 16–ую с/с: 69; 73; 113; 203; 351; 641; 478,99; 555,555
- д) из 8–ой с/с в 10–ую с/с: 35₈; 65₈; 215₈; 327₈; 532₈; 751₈; 45,454₈
- е) из 16–ой с/с в 10–ую с/с: D8₁₆; 1AE₁₆; E57₁₆; 8E5₁₆; FAD₁₆; AFF,6A7₁₆

2. Выпишите целые десятичные числа, принадлежащие следующим числовым промежуткам:

[10101₂; 110000₂]; [14₈; 20₈]; [18₁₆; 30₁₆]

ПРАКТИЧЕСКАЯ РАБОТА № 3-4

Тема: «Алгебра логики»

Цель: научиться выполнять перевод чисел в разные системы счисления.

Оборудование: справочные пособия, микрокалькуляторы.

Содержание работы

Задание

1. Выполнить операции:

а) сложение в двоичной системе счисления

$+ 10010011_2$	$+ 1011101_2$	$+ 10110011_2$	$+ 10111001,1_2$
1011011_2	11101101_2	1010101_2	$10001101,1_2$

вычитание в 2-ой системе счисления

б) $- 100001000_2$	$- 110101110_2$	$- 11101110_2$	$- 10111001,1_2$
10110011_2	10111111_2	1011011_2	$10001101,1_2$

в) умножение в 2-ой системе счисления

$\square 100001_2$	$\square 100101_2$	$\square 111101_2$	$\square 11001,01_2$
111111_2	111011_2	111101_2	$11,01_2$

г) деление в 2-ой системе счисления

1) $111010001001_2 / 111101_2$

2) $100011011100_2 / 110110_2$

3) $10000001111_2 / 111111_2$

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: «Минимизация логических схем»

Цель: изучить выполнение арифметических операций $A+B$ в различных системах счисления, изучить принципы выполнения арифметических операций в различных системах счисления.

Оборудование: справочные пособия, микрокалькуляторы.

Справочный материал

Сложение двоичных чисел

Способ сложения столбиком в общем-то такой же как и для десятичного числа. То есть, сложение выполняется поразрядно, начиная с младшей цифры. Если при сложении двух цифр получается СУММА больше девяти, то записывается цифра=СУММА- 10, а ЦЕЛАЯ ЧАСТЬ (СУММА /10), добавляется в старшему разряду. (Сложите пару чисел столбиком вспомните как это делается.) Так и с двоичным числом. Складываем поразрядно, начиная с младшей цифры. Если

получается больше 1, то записывается 1 и 1 добавляется к старшему разряду (говорят "на ум пошло").

Выполним пример: $10011 + 10001$.

	1	0	0	1	1
	1	0	0	0	1
1	0	0	1	0	0

Первый разряд: $1+1=2$. Записываем 0 и 1 на ум пошло.

Второй разряд: $1+0+1$ (запомненная единица) $=2$. Записываем 0 и 1 на ум пошло.

Третий разряд: $0+0+1$ (запомненная единица) $=1$. Записываем 1.

Четвертый разряд: $0+0=0$. Записываем 0.

Пятый разряд: $1+1=2$. Записываем 0 и добавляем к шестым разрядом 1.

Переведём все три числа в десятичную систему и проверим правильность сложения. $10011 = 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 16 + 2 + 1 = 19$

$10001 = 1*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 16 + 1 = 17$

$100100 = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 0*2^0 = 32 + 4 = 36$

$17 + 19 = 36$ верное равенство

Вычитание двоичных чисел

Вычитать числа, будем также столбиком и общее правило тоже, что и для десятичных чисел, вычитание выполняется поразрядно и если в разряде не хватает единицы, то она занимается в старшем. Решим следующий пример:

	1	1	0	1
-		1	1	0
		1	1	1

Первый разряд. $1 - 0 = 1$. Записываем 1.

Второй разряд $0 - 1$. Не хватает единицы. Занимаем её в старшем разряде. Единица из старшего разряда переходит в младший, как две единицы (потому что старший разряд представляется двойкой большей степени) $2 - 1 = 1$. Записываем 1.

Третий разряд. Единицу этого разряда мы занимали, поэтому сейчас в разряде 0 и есть необходимость занять единицу старшего разряда. $2 - 1 = 1$. Записываем 1.

Проверим результат в десятичной системе: $1101 - 110 = 13 - 6 = 7$ (111) Верное равенство.

Умножение в двоичной системе счисления

Для начала рассмотрим следующий любопытный факт. Для того, чтобы умножить двоичное число на 2 (десятичная двойка это 10 в двоичной системе) достаточно к умножаемому числу слева приписать один ноль.

Пример. $10101 * 10 = 101010$

Проверка.

$$10101 = 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 16 + 4 + 1 = 21$$

$$101010 = 1*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 32 + 8 + 2 = 42$$

$$21 * 2 = 42$$

Если мы вспомним, что любое двоичное число разлагается по степеням двойки, то становится ясно, что умножение в двоичной системе счисления сводится к умножению на 10 (то есть на десятичную 2), а стало быть, умножение это ряд последовательных сдвигов. Общее правило таково: как и для десятичных чисел, умножение двоичных выполняется поразрядно. И для каждого разряда второго множителя к первому множителю добавляется один ноль справа. Пример (пока не столбиком):

$1011 * 101$ Это умножение можно свести к сумме трёх поразрядных умножений:

$$1011 * 1 + 1011 * 0 + 1011 * 100 = 1011 + 101100 = 110111$$

В столбик это же самое можно записать так:

		1	0	1	1
	*		1	0	1
		1	0	1	1
	0	0	0	0	
1	0	1	1		
1	1	0	1	1	1

*Примечание: Кстати таблица умножения в двоичной системе состоит только из одного пункта $1 * 1 = 1$*

Проверка:

$$101 = 5 \text{ (десятичное)}$$

$$1011 = 11 \text{ (десятичное)}$$

$$110111 = 55 \text{ (десятичное)}$$

$$5 * 11 = 55 \text{ верное равенство}$$

Деление в двоичной системе счисления

Мы уже рассмотрели три действия и думаю уже понятно, что в общем-то действия над двоичными числами мало отличаются от действий над десятичными числами. Разница появляется только в том, что цифр две а не десять, но это только упрощает арифметические операции. Так же обстоит дело и с делением, но для лучшего понимания алгоритм деления разберём более подробно. Пусть нам необходимо разделить два десятичных числа, например 234 разделить на 7. Как мы это делаем.

2	3	4	7	

Мы выделяем справа (от старшего разряда) такое количество цифр, чтобы получившееся число было как можно меньше и в то же время больше делителя. 2 - меньше делителя, следовательно, необходимое нам число 23. Затем делим полученное число на делитель с остатком. Получаем следующий результат:

	2	3	4	7	
-	2	1		3	
		2	4		

Описанную операцию повторяем до тех пор, пока полученный остаток не окажется меньше делителя. Когда это случится, число полученное под чертой, это частное, а последний остаток - это остаток операции. Так вот операция деления двоичного числа выполняется точно также. Попробуем

Пример: 10010111 / 101

1	0	0	1	0	1	1	1	1	0	1

Ищем число, от старшего разряда которое первое было бы больше чем делитель. Это четырехразрядное число 1001. Оно выделено жирным шрифтом. Теперь необходимо подобрать делитель выделенному числу. И здесь мы опять выигрываем в сравнении в десятичной системой. Дело в том, что подбираемый делитель это обязательно цифра, а цифры у нас только две. Так как 1001 явно больше 101, то с делителем всё понятно это 1.

	1	0	0	1	0	1	1	1	1	0	1
-		1	0	1					1		
		1	0	0							

Итак, остаток от выполненной операции 100. Это меньше чем 101, поэтому чтобы выполнить второй шаг деления, необходимо добавить к 100 следующую цифру, это цифра 0. Теперь имеем следующее число:

Полученное число 11 меньше 101, поэтому записываем в частное цифру 0 и опускаем вниз следующую цифру. Получается так:

	1	0	0	1	0	0	1	1		1	0	1		
-		1	0	1						1	1	1	0	
		1	0	0	0									
-			1	0	1									
				1	1	0								
			-	1	0	1								
							1	1	1					

Полученное число больше 101, поэтому в частное записываем цифру 1 и опять выполняем действия. Получается такая картина:

	1	0	0	1	0	0	1	1		1	0	1		
-		1	0	1						1	1	1	0	1
		1	0	0	0									
-			1	0	1									
				1	1	0								
			-	1	0	1								
							1	1	1					
					-	1	0	1						
							1	0						

Полученный остаток 10 меньше 101, но у нас закончились цифры в делимом, поэтому 10 это окончательный остаток, а 1110 это искомое частное.

Проверим в десятичных числах $10010011 = 147$

$101 = 5$

$10 = 2 \quad 11101 = 29$

	1	4	7	5	
-	1	0		2	9
		4	7		
	-	4	5		
			2		

Содержание работы

1. Выполнить сложение двоичных чисел:

- а) $11001 + 101 =$ _____
- б) $11001 + 11001 =$ _____
- в) $1001 + 111 =$ _____
- г) $10011 + 101 =$ _____
- д) $11011 + 1111 =$ _____
- е) $11111 + 10011 =$ _____

2. Выполнить вычитание двоичных чисел:

- а) $11001 - 1001 =$ _____ б) $1011 - 110 =$ _____
- в) $10001 - 101 =$ _____ г) $10101 - 11 =$ _____
- д) $101001 - 1111 =$ _____ е) $111111 - 101010 =$ _____

3. Выполнить умножение двоичных чисел:

- а) $1101 * 1110 =$ _____ б) $1010 * 110 =$ _____
- в) $1011 * 11 =$ _____ г) $101011 * 1101 =$ _____
- д) $10010 * 1001 =$ _____

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: «Минимизация логических схем»

Цель: научиться выполнять перевод чисел в разные системы счисления.

Оборудование: справочные пособия, микрокалькуляторы.

Содержание работы

Задание

1) сложение 8-ых чисел

$$\begin{array}{ccccccccc} + 715_8 & + 524_8 & + 712_8 & + 321_8 & + 5731_8 & + 6351_8 \\ 73_8 & 57_8 & 763_8 & 765_8 & 1376_8 & 737_8 \end{array}$$

2) вычитание 8-ых чисел

$$\begin{array}{ccccccccc} - 137_8 & - 436_8 & - 705_8 & - 538_8 & - 7213_8 \\ 72_8 & 137_8 & 76_8 & 57_8 & 537_8 \end{array}$$

3) сложение 16-ых чисел

$$\begin{array}{ccccccccc} + A13_{16} & + F0B_{16} & + 2EA_{16} & + ABC_{16} & + A2B_{16} \\ 16F_{16} & 1DA_{16} & FCE_{16} & C7C_{16} & 7F2_{16} \end{array}$$

4) вычитание 16-ых чисел

$$\begin{array}{ccccccccc} - A17_{16} & - DFA_{16} & - FO5_{16} & - DE5_{16} & - D3C1_{16} \\ 1FC_{16} & 1AE_{16} & AD_{16} & AF_{16} & D1F_{16} \end{array}$$

5) Вычислите выражения (ответ записать в восьмеричной С/С):

$$(1111101_2 + AF_{16}) / 36_8;$$

$$125_8 + 11101_2 \square A2_{16} / 1417_8$$

ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема: «Минимизация логических схем»

Цель: научиться переводить основные логические элементы.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Логическое высказывание – это любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Математический аппарат алгебры логики очень удобен для описания того, как функционируют аппаратные средства компьютера, поскольку основной системой счисления в компьютере является двоичная, в которой используются цифры 1 и 0, а значений логических переменных тоже два: 1 и 0.

Логический элемент компьютера — это часть электронной логической схемы, которая реализует элементарную логическую функцию.

Логическими элементами компьютеров являются электронные схемы И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и др. (называемые также вентилями), а также триггер. С помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Работу логических элементов описывают с помощью таблиц истинности.

Схемы, реализующие логические функции, называются логическими элементами. Основные логические элементы имеют, как правило, один выход (Y) и несколько входов, число которых равно числу аргументов ($X_1; X_2; X_3 \dots$

X_N). На электрических схемах логические элементы обозначаются в виде прямоугольников с выводами для входных (слева) и выходных (справа) переменных. Внутри прямоугольника изображается символ, указывающий функциональное назначение элемента.

Схема И реализует конъюнкцию двух или более логических значений. Условное обозначение на структурных схемах схемы И с двумя входами представлено на рисунке 1.

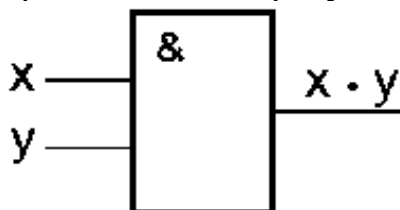


Рисунок 1 - Структурная схема схемы И

Таблица 1 - Таблица истинности схемы И

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Единица на выходе схемы И будет тогда и только тогда, когда на всех входах будут единицы. Когда хотя бы на одном входе будет ноль, на выходе также будет ноль.

Связь между выходом z этой схемы и входами x и y описывается соотношением: $z = x \cdot y$ (читается как "x и y"). Операция конъюнкции на структурных схемах обозначается знаком "&" (читается как "амперсэнд"), являющимся сокращенной записью английского слова *and*.

Схема ИЛИ реализует дизъюнкцию двух или более логических значений. Когда хотя бы на одном входе схемы ИЛИ будет единица, на её выходе также будет единица.

Условное обозначение на структурных схемах схемы ИЛИ с двумя входами представлено на рисунке 2. Знак "1" на схеме — от устаревшего обозначения дизъюнкции как " ≥ 1 " (т.е. значение дизъюнкции равно единице, если сумма значений операндов больше или равна 1). Связь между выходом z этой схемы и входами x и y описывается соотношением: $z = x \vee y$ (читается как "x или y").

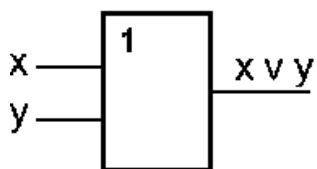


Рисунок 2 – Структурная схема схемы ИЛИ

Таблица 2 – Таблица истинности схемы ИЛИ

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Схема НЕ (инвертор) реализует операцию отрицания. Связь между входом x этой схемы и выходом z можно записать соотношением $z = \bar{x}$, где \bar{x} читается как "не x " или "инверсия x ".

Если на входе схемы 0, то на выходе 1. Когда на входе 1, на выходе 0. Условное обозначение на структурных схемах инвертора представлена на рисунке 3.

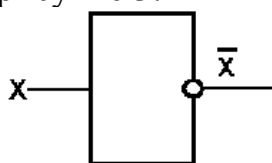


Рисунок 3 - Структурная схема инвертора

Таблица 3 - Таблица истинности схемы НЕ

x	\bar{x}
0	1
1	0

Схема И—НЕ состоит из элемента И и инвертора и осуществляет отрицание результата схемы И. Связь между выходом z и входами x и y схемы записывают следующим образом: $z = \overline{x \cdot y}$, читается как "инверсия x и y ". Условное обозначение на структурных схемах схемы И—НЕ с двумя входами представлено на рисунке 4.

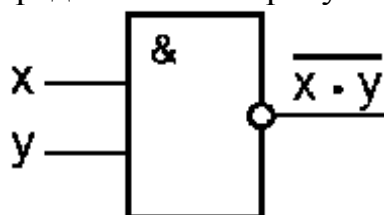


Рисунок 4 - Структурная схема схемы И—НЕ

Таблица 4 - Таблица истинности схемы И—НЕ

x	y	$x * y$
0	0	1
0	1	1
1	0	1
1	1	0

Схема ИЛИ—НЕ состоит из элемента ИЛИ и инвертора и осуществляет отрицание результата схемы ИЛИ. Связь между выходом z и входами x и y схемы записывают следующим образом: $x \vee y$, читается как "инверсия x или y ". Условное обозначение на структурных схемах схемы ИЛИ—НЕ с двумя входами представлено на рисунок 5.

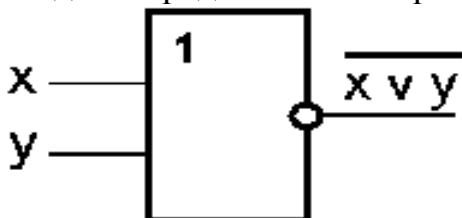


Рисунок 5 - Структурная схема схемы ИЛИ—НЕ

Таблица 5 - Таблица истинности схемы ИЛИ—НЕ

x	y	$x \vee y$
0	0	1
0	1	0
1	0	0
1	1	0

2. Пример выполнения заданий

Проверить тождественность логических функций X и Y

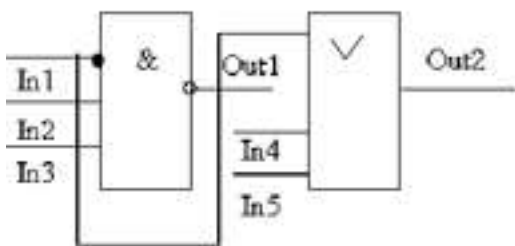
$$X = \neg(f_1 * f_2 + \neg f_1 * f_3) * (f_1 * \neg f_2 + f_2 * f_3) * (f_1 * f_3 + \neg f_2) \quad Y = (f_1 + f_2 * f_3) * (\neg f_1 + \neg f_2 * f_3) * (\neg f_1 * \neg f_2 + f_1 * f_3)$$

Таблица 6 – Значения функции x							
f1	f2	f3	$\wedge(f1*f2+\wedge f1*f3)$	$(f1*\wedge f2+f2*f3)$	$(f1*f3+\wedge f2)$	x	
0	0	0	$(0*0+1*0) 1$	$(0*1+0*0) 0$	$(0*0+1) 1$	0	
0	0	1	$(0*0+1*1) 0$	$(0*1+0*1) 0$	$(0*1+1) 1$	0	
0	1	0	$(0*1+1*0) 1$	$(0*0+1*0) 0$	$(0*0+0) 0$	0	
1	0	0	$(1*0+0*0) 1$	$(1*1+0*0) 1$	$(1*0+1) 1$	1	
0	1	1	$(0*1+1*1) 0$	$(0*0+1*1) 1$	$(0*1+0) 1$	0	
1	0	1	$(1*0+0*1) 1$	$(1*1+0*1) 1$	$(1*1+1) 1$	1	
1	1	0	$(1*1+0*0) 0$	$(1*0+1*0) 0$	$(1*0+0) 0$	0	
1	1	1	$(1*1+0*1) 0$	$(1*0+1*1) 1$	$(1*1+0) 1$	0	

Таблица 7 – Значения функции y

f1	f2	f3	$(f1+f2*f3)$	$(\wedge f1+\wedge f2*f3)$	$(\wedge f1*\wedge f2+f1*f3)$	y
0	0	0	$(0+0*0) 0$	$(1+1*0) 1$	$(1*1+0*0) 1$	0
0	0	1	$(0+0*1) 0$	$(1+1*1) 1$	$(1*1+0*1) 1$	0
0	1	0	$(0+1*0) 0$	$(1+0*0) 1$	$(1*0+0*0) 0$	0
1	0	0	$(1+0*0) 1$	$(0+1*0) 0$	$(0*1+1*0) 0$	0
0	1	1	$(0+1*1) 1$	$(1+0*1) 1$	$(1*0+0*1) 0$	0
1	0	1	$(1+0*1) 1$	$(0+1*1) 1$	$(0*1+1*1) 1$	1
1	1	0	$(1+1*0) 1$	$(0+0*0) 0$	$(0*0+1*1) 0$	0
1	1	1	$(1+1*1) 1$	$(0+0*1) 0$	$(0*0+1*2) 0$	0

Вывод: при одинаковых значениях на входе функций f_1, f_2, f_3 , значения функций x и y различны, соответственно x и y не тождественны.
Составить логическое выражение по схеме и таблицы истинности для выходных функций.



$$\text{Out 1} = \text{In1} * \text{In2} * \text{In3}$$

Таблица 8 – Значения функции Out1

In1	In2	In3	$\overline{\text{In1}}$	$\overline{\text{In1}} * \text{In2} * \text{In3}$	$\overline{\overline{\text{In}} * \text{In2} * \text{In3}}$
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	0	0	1
1	1	1	0	0	1

$$\text{Out2} = \text{In1} + \text{In4} + \text{In5}$$

Таблица 9 – Значения функции Out2

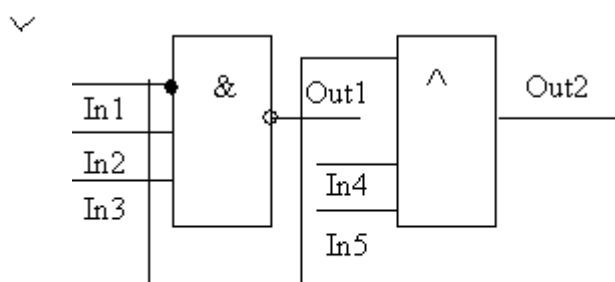
In1	In4	In5	$\overline{\text{In1}}$	$\overline{\text{In1}} + \text{In4} + \text{In5}$
0	0	0	1	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0

1	1	0	0	0
1	1	1	0	1

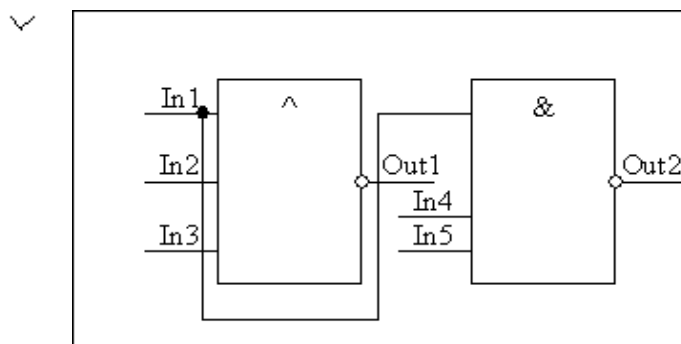
Содержание работы

Задание 1. Составить логическое выражение по схеме и таблицы истинности для выходных функций.

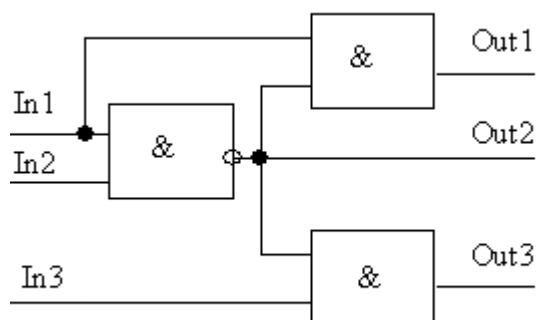
Вариант 1



Вариант 2

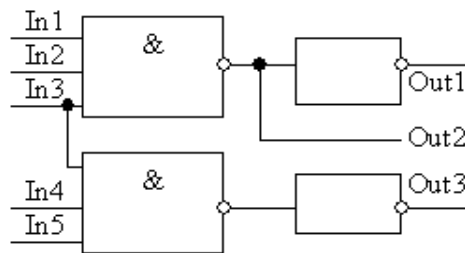


Вариант 3

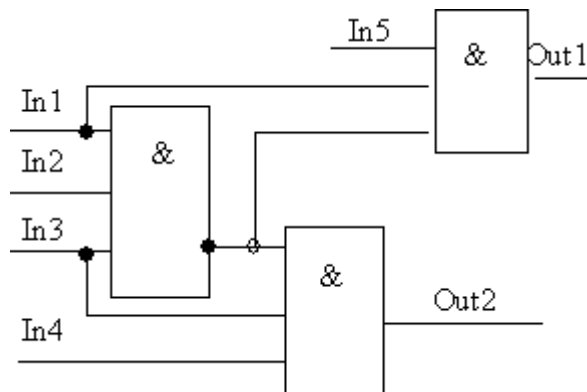


Вариант 4

✓ ✓

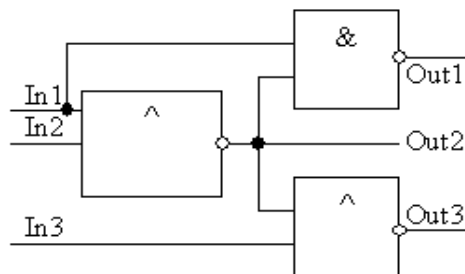


Вариант 5



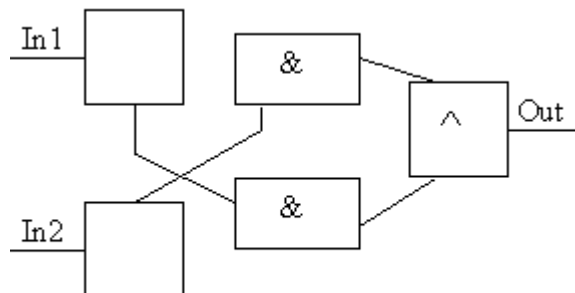
Вариант 6

✓ ✓

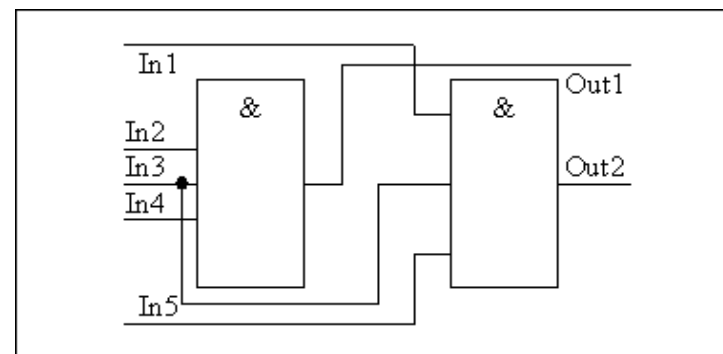


Вариант 7

✓ ✓



Вариант 8



Задание 2. Проверить тождественность логических функций X и Y.

1. $X = (f1 + f2 * f3) * (f1 * f3 + \wedge f2) * (\wedge f1 * \wedge f2 + f1 * f3)$
 $Y = (\wedge f1 + \wedge f2 * f3) * (f1 * f2 + \wedge f1 * f3) * (f1 * \wedge f2 + f2 * f3)$
2. $X = (f1 + f2 * f3) * \wedge (\wedge f1 * \wedge f2 + f * f3) * (\wedge f1 + \wedge f2 * f3)$
 $Y = \wedge (f1 * f3 + \wedge f2) * (f1 * \wedge f2 + f2 * f3) * (f1 * f2 + \wedge f1 * f3)$
3. $X = (f1 + f2 * f3) * (\wedge f1 * \wedge f2 + f1 * f3) * \wedge (\wedge f1 + \wedge f2 * f3)$
 $Y = (\wedge f1 * f2 + \wedge f2 * f3) * \wedge (f1 * f3 + f2) * \wedge (f1 * \wedge f2 + f2 * f3)$
4. $X = (f1 + f2 * f3) * (\wedge f1 * \wedge f2 + f1 * f3) * \wedge (f1 * \wedge f2 + f2 * f3)$
 $Y = (\wedge f1 + \wedge f2 * f3) * (f1 * f3 + \wedge f2) * \wedge (f1 * f2 + \wedge f1 * f3)$
5. $X = \wedge (\wedge f1 + \wedge f2 * f3) * \wedge (f1 * \wedge f2 + f2 * f3) * (\wedge f1 * \wedge f2 + f1 * f3)$
 $Y = \wedge (f1 * f3 + f2) * \wedge (f1 + f2 * f3) * (f1 * f2 + \wedge f1 * f3)$
6. $X = \wedge (f1 + f2 * f3) * \wedge (f1 * f3 + \wedge f2) * (f1 * \wedge f2 + f2 * f3)$
 $Y = (f1 * f2 + \wedge f1 * f3) * (\wedge f1 + \wedge f2 * f3) * (\wedge f1 * \wedge f2 + f1 * f3)$
7. $X = (f1 + f2 * f3) * (f1 * f2 + \wedge f1 * f3) * (f1 * \wedge f2 + f2 * f3)$
 $Y = \wedge (\wedge f1 * f2 + \wedge f2 * f3) * (\wedge f1 * \wedge f2 + f1 * f3) * (f1 * f3 + \wedge f2)$
8. $X = (\wedge f1 * f2 + f2 * f3) * \wedge (f1 + f2 * f3) * (\wedge f1 + \wedge f2 * f3)$
 $Y = \wedge (f1 * f3 + \wedge f2) * (f1 * \wedge f2 + f2 * f3) * \wedge (\wedge f1 * \wedge f2 + f1 * f3)$
9. $X = \wedge (f1 * f2 + \wedge f1 * f3) * (f1 * \wedge f2 + f2 * f3) * (f1 * f3 + \wedge f2)$
 $Y = (f1 + f2 * f3) * (\wedge f1 + \wedge f2 * f3) * (\wedge f1 * \wedge f2 + f1 * f3)$

ПРАКТИЧЕСКАЯ РАБОТА № 8-9

Тема: «Минимизация логических схем»

Цель: научиться представлять логические функции.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Основные понятия и определения.

Определение. Формула называется тождественно-истинной (тавтологией), если для любых наборов переменных она принимает значение И.

[illegible]

По таблице составляем дизъюнктивную нормальную форму (ДНФ). ДНФ в булевой логике — нормальная форма, в которой булева формула имеет вид дизъюнкции нескольких конъюнктов.

Алгоритм получения СДНФ по таблице истинности:

- 1) Отметить те строки, в последнем столбце которых стоят 1;
- 2) Выписать для каждой отмеченной строки конъюнкцию всех переменных следующим образом: если значение некоторой переменной в данной строке =1, то в конъюнкцию включают саму эту переменную, если =0, то ее отрицание;
- 3) Все полученные конъюнкции связать в дизъюнкцию:

Выбираем в таблице строки, в которых булева функция принимает значение 1. В данном случае – это 2-ая, 3-ая, 4-ая, 6-ая и 7-ая строки.

Для каждой строки составляем конъюнкцию: если значение переменной равно 0, то берем ее отрицание, а если 1, то берем саму переменную. Затем составляем дизъюнкцию полученных конъюнкций:

$$f(x, y, z) = (x \wedge \bar{y} \wedge z) \vee (x \wedge y \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge \bar{y} \wedge z) \vee (x \wedge \bar{y} \wedge z)$$

Выбираем в таблице строки, в которых булева функция принимает значение 0. В данном случае – это 1-ая, 5-ая, и 8-ая строки:

$$f(x, y, z) = (\bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge y \wedge z)$$

ДНФ называется минимальной, если она содержит наименьшее число букв среди всех ДНФ ей равносильных.

Метод Квайна основывается на применении двух основных соотношений.

Соотношение склеивания :

$$(a \wedge b) \vee (\bar{a} \wedge b) = b ; (a \vee b) \wedge (\bar{a} \vee b) = b$$

Соотношение поглощения:

$$a \wedge (a \vee b) = a \quad a \vee (a \wedge b) = a$$

Используя соотношение склеивания получаем:

$$(\bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) = \bar{y} \wedge \bar{z} ,$$

$$(x \wedge y \wedge \bar{z}) \vee (x \wedge y \wedge z) = x \wedge y . \text{ Отсюда,}$$

$$(\bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge y \wedge \bar{z}) \vee (x \wedge y \wedge z) = (\bar{y} \wedge \bar{z}) \vee (x \wedge y) - \text{сокращенная}$$

ДНФ

Порядок выполнения работы

Задание. Построить таблицу истинности, найти СДНФ, найти минимальную ДНФ для высказывания:

Вариант 1

$$1. (\bar{z} \vee y) \rightarrow (\bar{z} \oplus \bar{x})$$

$$2. \left((\overline{A \wedge B}) \Rightarrow A \right) \Rightarrow A \vee B$$

$$3. (\bar{z} \vee y) \wedge (\bar{z} \oplus \bar{x})$$

Вариант 2

1. $\left((\overline{A \wedge B}) \Rightarrow A \right) \Leftrightarrow (A \vee B)$
2. $x \mid (y \rightarrow z) \oplus (x \mid y) \rightarrow (x \mid z)$
3. $(\bar{z} \Rightarrow y) \Leftrightarrow (\bar{z} \vee \bar{x})$

Вариант 3

1. $(x \mid y) \rightarrow (x \mid z)$
2. $(\overline{A \wedge B}) \Leftrightarrow (\bar{B} \oplus \bar{A}) \Leftrightarrow (A \vee B) \oplus (A \oplus \bar{B})$
3. $(\bar{z} \oplus y) \Rightarrow (\bar{z} \mid (y \vee \bar{x}))$

Вариант 4

1. $(\overline{A \Rightarrow B}) \Leftrightarrow (\bar{B} \wedge \bar{A})$
2. $(x \wedge y) \oplus (x \wedge z) \Leftrightarrow x \wedge (y \oplus z)$
3. $(\bar{z} \oplus x) \vee (\bar{z} \mid (y \vee \bar{x}))$

Вариант 5

1. $((x \downarrow y) \rightarrow z) \oplus y$
2. $(x \mid y) \rightarrow (x \mid z) \oplus (\bar{z} \vee y) \rightarrow (\bar{z} \oplus \bar{x})$
3. $(\bar{z} \vee y) \rightarrow (\bar{z} \mid (y \vee \bar{x}))$

Вариант 6

1. $(x \vee \bar{y}) \rightarrow (\bar{z} \oplus \bar{x})$
2. $(\overline{A \Rightarrow B}) \Leftrightarrow (\bar{B} \wedge \bar{A}) \oplus ((A \Rightarrow B) \wedge \bar{B}) \Rightarrow A$
- 3.

Вариант 7

- 1.
2. $(\overline{A \Rightarrow B}) \vee (\bar{B} \wedge \bar{A}) \Rightarrow ((A \Rightarrow B) \wedge \bar{B}) \oplus A$
3. $(\bar{z} \vee x) \Leftrightarrow (\bar{z} \mid (y \vee \bar{x}))$

Вариант 8

1. $((A \vee B) \wedge B) \Rightarrow A$
2. $x \mid (y \Rightarrow z) \Leftrightarrow (x \mid y) \vee (x \mid z)$
3. $(\bar{z} \Leftrightarrow y) \Leftrightarrow (\bar{z} \mid (y \oplus \bar{x}))$

Вариант 9

1. $(x \mid \bar{y}) \oplus (z \rightarrow \bar{x})$

2. $(\overline{A \Rightarrow B}) \vee (\overline{B} \wedge \overline{A}) \Leftrightarrow ((A \Rightarrow B) \oplus \overline{B}) \vee A$
3. $((A \vee B) \oplus \overline{B}) \Rightarrow A$

Вариант 10

1. $\left(\overline{A \vee B \wedge A} \right) \Leftrightarrow \overline{A}$
2. $(x \wedge y) \vee (x \wedge z) \Rightarrow$
3. $(x \vee \overline{y}) \rightarrow (\overline{z} \Leftrightarrow \overline{x})$

Вариант 11

1. $(\overline{z \rightarrow x}) \Leftrightarrow (\overline{y|x})$
2. $(\overline{A \vee B}) \vee (\overline{B} \wedge \overline{A}) \Leftrightarrow ((A \vee B) \oplus \overline{B}) \Rightarrow A$
3. $(\overline{z} \oplus y) \vee (\overline{z}|(y \vee \overline{x}))$

Вариант 12

- 1.
2. $x|(y \oplus z) \oplus (x|y) \vee (x|z)$
3. $(\overline{A \vee B}) \Leftrightarrow (\overline{B} \wedge \overline{A})$

Вариант 13

1. $(x \wedge y) \oplus (x \wedge z)$
2. $(\overline{A \Rightarrow B}) \wedge (\overline{B} \Leftrightarrow \overline{A}) \Leftrightarrow ((A \Rightarrow B) \wedge \overline{B}) \oplus A$
3. $(\overline{z} \Leftrightarrow y) \vee (\overline{z}|(z \vee \overline{x}))$

Вариант 14

1. $(x|\overline{y}) \oplus (\overline{z} \rightarrow x)$
2. $(\overline{A \oplus B}) \Leftrightarrow (\overline{B} \oplus \overline{A}) \Leftrightarrow A \Rightarrow ((A \vee B) \wedge \overline{B})$
3. $(\overline{z} \Rightarrow y) \oplus (\overline{z}|(y \vee \overline{x}))$

Вариант 15

1. $\left(\overline{(A \wedge B) \Rightarrow A} \right) \Leftrightarrow (A \downarrow B)$
2. $x|(y \wedge z) \Rightarrow (x|y) \oplus (x|z)$
3. $(\overline{z} \vee y) \rightarrow (\overline{z} \oplus \overline{x})$

ПРАКТИЧЕСКАЯ РАБОТА № 10-13

Тема: «RS-триггер»

Цель: изучить понятие триггер и способы его реализации.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

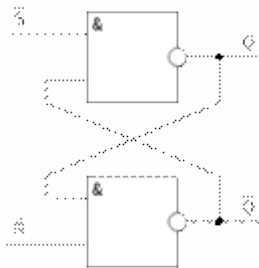
Входы триггера

Входы триггеров обычно обозначают следующим образом:

- S (от англ. *Set*, установить) — вход в RS-триггере;
- R (от англ. *Reset*, сброс) — вход в RS-триггере;
- J (от англ. *Jump*^[4], прыжок) — вход в JK-триггере;
- K (от англ. *Kill*, убить) — вход в JK-триггере;
- T (от англ. *Toggles*, переключить) — счётный вход в T-триггере;
- C (от англ. *Clock*, время) вход синхронизирующего сигнала. При тактировании по фронту он часто обозначается стрелкой: стрелка внутрь — тактирование по переднему фронту, наружу — по заднему.
- D (от англ. *Delay*, задержка) — вход в D-триггере;
- E или EN (от англ. *Enable*, разрешить) — дополнительный асинхронный управляющий вход для разрешения приёма информации (иногда используют букву V).

Входы **J, K, T, D** всегда синхронные, т.е. тактируются по синхронизирующему сигналу на входе **C**. Разумеется, в каждом конкретном триггере имеются лишь некоторые из перечисленных входных линий. Входы **S** и **R** зачастую присутствуют не только в RS триггерах, но и в других типах триггеров, где предназначены, в основном, для асинхронного сброса устройства в 0 или установки в 1.

RS-триггер, или SR-триггер



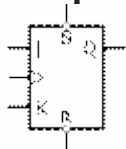
Одна из наглядных схем реализации асинхронного RS-триггера на базе двух элементов **2И-НЕ(NAND2)**

RS-триггер, или SR-триггер — триггер, который сохраняет своё предыдущее состояние при нулевых входах и меняет своё выходное состояние при подаче на один из его входов единицы. При подаче единицы на вход **S** (от англ. *Set* - установить) выходное состояние становится равным логической единице. А при подаче единицы на вход **R** (от англ. *Reset* - сбросить) выходное состояние становится равным логическому нулю. Если RS-триггер синхронный, то состояние его входов учитывается только в момент тактирования, например по

переднему фронту импульса. Состояние, при котором на оба входа **R** и **S** одновременно поданы логические единицы, является запрещённым. Так, например, схема RS-триггера, изображённая на рисунке, при подаче на оба инверсных входа логического нуля перейдёт в состояние, когда на обоих выходах будут единицы, что не соответствует логике выхода триггера, поскольку инверсный выход \bar{Q} будет равен неинверсному Q , т.е.

RS-триггер используется для создания сигнала с положительным и отрицательным фронтами, отдельно управляемыми посредством стробов, разнесённых во времени.

JK-триггер

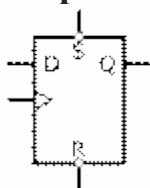


Символ JK-триггера с дополнительными асинхронными входами **S** и **R**, аналогично представлению в среде разработки

JK-триггер работает также как RS-триггер, с одним лишь исключением: при подаче логической единицы на оба входа **J** и **K** состояние выхода триггера изменяется на противоположное. Вход **J** (от англ. *Jump* - прыжок) аналогичен входу **S** у RS-триггера. Вход **K** (от англ. *Kill* - убить) аналогичен входу **R** у RS-триггера. При подаче единицы на вход **J** и нуля на вход **K** выходное состояние триггера становится равным логической единице. А при подаче единицы на вход **K** и нуля на вход **J** выходное состояние триггера становится равным логическому нулю. JK-триггер в отличие от RS-триггера не имеет запрещённых состояний на основных входах, однако это никак не помогает при нарушении правил разработки логических схем. На практике применяются только синхронные JK-триггеры, то есть состояния основных входов **J** и **K** учитываются только в момент тактирования, например по положительному фронту импульса на входе синхронизации.

На базе JK-триггера возможно построить D-триггер или T-триггер. Как можно видеть в таблице истинности JK-триггера, он переходит в инверсное состояние каждый раз при одновременной подаче на входы **J** и **K** логической 1. Это свойство позволяет создать на базе JK-триггера T-триггер, объединив входы **J** и **K**^[5].

D-триггер



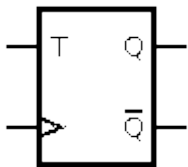
Символ D-триггера с дополнительными асинхронными входами **S** и **R**

D-триггер (**D** от англ. *delay* - задержка) - запоминает состояние входа и выдаёт его на выход. D-триггеры имеют, как минимум, два входа: информационный **D** и синхронизации **C**. Сохранение информации в D-

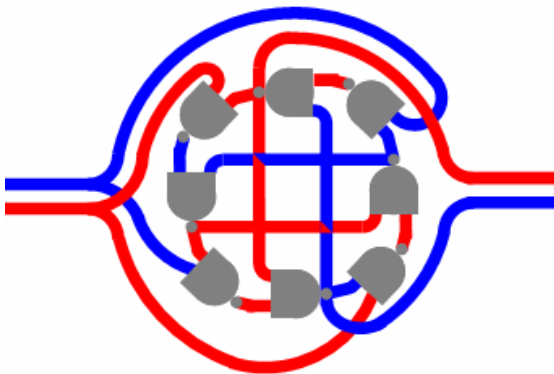
триггерах происходит в момент прихода активного фронта на вход С. Так как информация на выходе остаётся неизменной до прихода очередного импульса синхронизации, D-триггер называют также триггером с запоминанием информации или триггером-защёлкой. Рассуждая чисто теоретически, D-триггер можно образовать из любых RS- или JK-триггеров, если на их входы одновременно подавать взаимно инверсные сигналы.

D-триггер в основном используется для реализации защёлки. Так, например, для снятия 32 бит информации с параллельной шины, берут 32 D-триггера и объединяют их входы синхронизации для управления записью информации в защёлку, а 32 D входа подсоединяют к шине.

Т-триггер



Изображение Т-триггера на схемах.



Работа схемы Т-триггера (при $T=1$) на базе восьми **2И-НЕ** логических вентилей. Слева — входы, справа — выходы. Синий цвет соответствует 0, красный — 1. **Т-триггер** по каждому такту изменяет своё логическое состояние на противоположное при единице на входе **T**, и не изменяет выходное состояние при нуле на входе **T**. Т-триггер часто называют счётным триггером. Т-триггер может строиться как на JK, так и на D-триггерах. Как можно видеть в таблице истинности JK-триггера, он переходит в инверсное состояние каждый раз при одновременной подаче на входы **J** и **K** логической 1. Это свойство позволяет создать на базе JK-триггера Т-триггер, объединяя входы **J** и **K**. Наличие в D-триггере динамического С входа позволяет получить на его основе Т-триггер. При этом вход D соединяется с инверсным выходом, а на вход С подаются счётные импульсы. В результате триггер при каждом счётном импульсе запоминает значение \bar{Q} , то есть будет переключаться в противоположное состояние.

Т-триггер — триггер, который меняет своё предыдущее состояние на противоположное под воздействием каждого входного сигнала равного логической «1».

Характеристическое уравнение, определяющее логику функционирования асинхронного Т-триггера:

$$Q^{n+1} = T^n \overline{Q^n} \vee \overline{T^n} Q^n$$

Экспериментальное исследование работы асинхронного Т-триггера.

Для исследования работы Т-триггера используется модуль «Триггеры» лабораторного стенда. Если текущее состояние триггера Q_n не совпадает с требуемым состоянием согласно таблице истинности, то для переключения триггера необходимо подать единичный импульс или сигнал логической «1» на вход триггера.

Перед началом работы с триггером его вход необходимо подключить с помощью соединительных проводников к одному из выходов логических сигналов модуля «Задание логических уровней и логические элементы». Исследование работы Т-триггера производится путем установки триггера в необходимое начальное состояние Q_n и подачи на его вход сигналов логического «0» или логической «1» в соответствии с таблицей истинности.

Таблица истинности асинхронного Т-триггера.

№ набора Т	Q_n	Q_{n+1}
0	0	0
1	0	1
2	1	0
3	1	1

Содержание работы

Задание. В рабочей тетради изобразить УГО исследуемого триггера и указать назначение входов/выходов. Полученные значения сигналов на выходах триггера занести в таблицу истинности в рабочей тетради.

ПРАКТИЧЕСКАЯ РАБОТА № 14-16

Тема: «D-триггер»

Цель: Изучение назначения и функций регистра. Знакомство с принципом работы регистров.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Регистры – устройства для временного хранения и преобразования информации в виде многоразрядных двоичных чисел. Регистры наряду со счетчиками и

запоминающими устройствами являются наиболее распространенными устройствами цифровой техники. При сравнительной простоте регистры обладают большими функциональными возможностями.

Они используются в качестве управляющих и запоминающих устройств, генераторов и преобразователей кодов, счетчиков, делителей частоты, узлов временной задержки.

Элементами структуры регистров являются триггеры D- или JK- типа с динамическим или статическим управлением. Одиночный триггер может запоминать (регистировать) один разряд (бит) двоичной информации. Такой триггер можно считать одноразрядным регистром. Занесение информации в регистр называют операцией ввода или записи. Выдача информации к внешним устройствам характеризует операцию вывода или считывания.

Запись информации в регистр не требует его предварительного обнуления.

Все регистры в зависимости от функциональных свойств подразделяются на две

категории – накопительные (регистры памяти, хранения) и сдвигающие. В свою очередь, сдвигающие регистры делятся по способу ввода и вывода информации на параллельные и последовательно-параллельные, и комбинированные, по направлению передачи (сдвига) информации – на однонаправленные и реверсивные.

На рисунке 1 показана схема простейшего четырехразрядного регистра на D – триггерах, в котором информация заносится последовательно, начиная с младшего разряда.

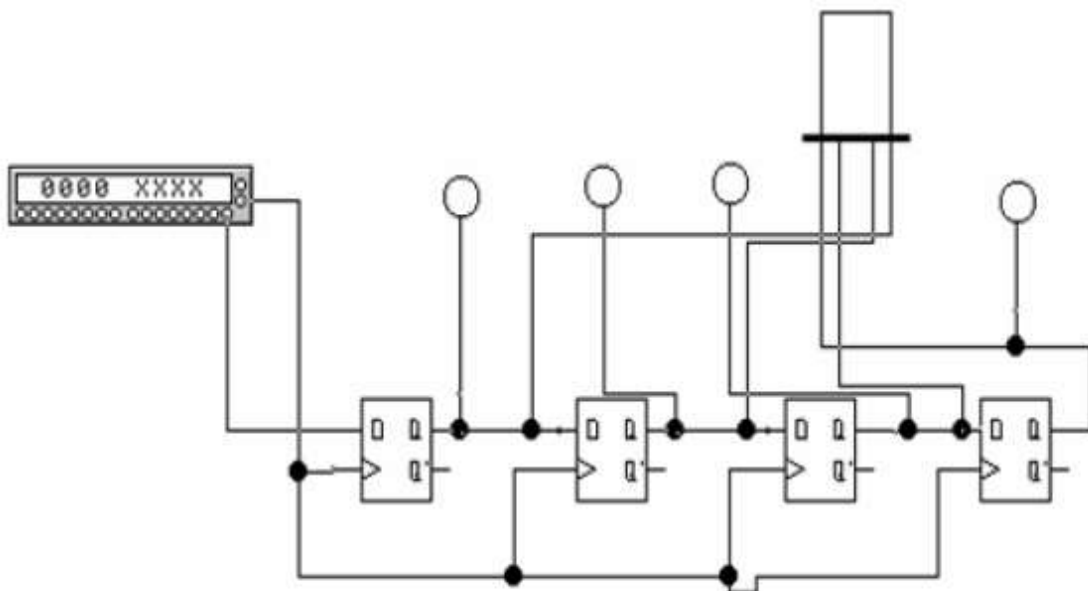


Рисунок 1

Содержание работы

Задание: Ответить на вопросы и выполнить задание

1. Что такое регистр, какие функции он может выполнять?
2. Назовите типы регистров и их возможные применения.

3.Смодулируйте приведенную выше схему и проанализируйте работу регистра.

ПРАКТИЧЕСКАЯ РАБОТА № 17-19

Тема: «ЖК-триггер»

Цель: Изучение назначения и функции устройства счетчик. Знакомство с принципом работы устройства счетчик.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Счетчиком называют устройство, сигналы на выходе которого отображают число импульсов, поступивших на счетный вход. Триггер может служить примером простейшего счетчика. Такой счетчик считает до двух. Счетчик, образованный цепочкой из m триггеров, может подсчитать в двоичном коде 2^m импульсов. Каждый из триггеров такой цепочки называют разрядом счетчика. Число m определяет количество разрядов двоичного числа, которое может быть записано в счетчик. Число $K_{сч}=2^m$ называют коэффициентом (модулем) счета. Информация снимается с прямых и (или) инверсных выходов всех триггеров. В паузах между входными импульсами триггеры сохраняют свои состояния, т. е. счетчик запоминает число входных импульсов.

Нулевое состояние всех триггеров принимается за нулевое состояние счетчика в целом. Остальные состояния складываются по числу поступивших входных импульсов.

Когда число входных импульсов $N_{вх} > K_{сч}$ происходит переполнение, после чего счетчик возвращается в нулевое состояние и цикл повторяется. Коэффициент счета, таким образом, характеризует число входных импульсов, необходимое для одного цикла и возвращения в исходное состояние.

Счетчики различаются числом и типами триггеров, способами связей между ними, кодом, организацией счета и другими показателями. Цифровые счетчики классифицируются по следующим параметрам:

- Коэффициент счета – двоичные; двоично-десятичные или с другим основанием счета; с произвольным постоянным и переменным (программируемым) коэффициентом счета;
- Направление счета – суммирующие, вычитающие и реверсивные;
- Способ организации внутренних связей – с последовательным, параллельным или комбинированным переносом, кольцевые.

Классификационные признаки независимы и могут встречаться в различных сочетаниях: например, суммирующие счетчики бывают как с последовательным, так и с параллельным переносом, могут иметь двоичный, десятичный и иной коэффициент счета.

Схема четырехразрядного двоичного счетчика с последовательным переносом на D – триггерах приведена на рисунке 1.

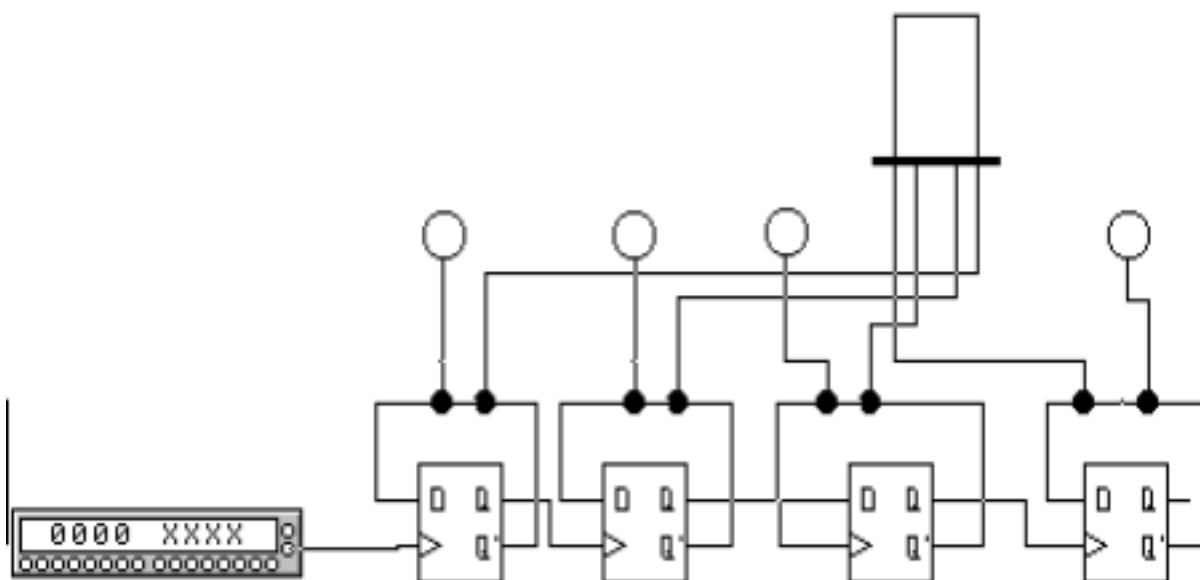


Рисунок 1

На вход счетчика подаются импульсы с выхода синхросигналов генератора слова, которые генерируются при каждом нажатии клавиши STEP. Каждый триггер счетчика осуществляет деление на 2, сигнал переноса передается последовательно от одного разряда к другому. Состояние разрядов счетчиков в двоичном коде индицируются логическим пробником (индикатором), а в десятичном – семисегментным индикатором.

Содержание работы

Задание: Ответить на вопросы и выполнить задание

1. Что такое счетчик, какие функции он может выполнять?
2. Назовите типы счетчиков и их возможные применения.
3. Смоделируйте приведенную выше схему и проанализируйте работу счетчика.

ПРАКТИЧЕСКАЯ РАБОТА №20-27

Тема: «Т-триггер»

Цель: разобраться в арифметико-логических устройствах.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Подготовка к выполнению практической работы

1. Изучить описание практической работы.
2. Ответить на все контрольные вопросы.
3. Произвести синтез параллельного АЛУ по варианту.

4. Разработать функциональную схему параллельного АЛУ.
5. Разработать алгоритмы выполнения операций.
6. Выполнить операции над двоичными данными (кодами), согласно списка указанного в таблице вариантов (таблица 3).

Порядок выполнения работы

1. Произвести исследование i -го разряда комбинационной части АЛУ.
2. Произвести исследование АЛУ.

Отчет должен содержать

1. Функциональную схему АЛУ со списком операций и таблицей значений управляющих сигналов.
2. Таблицу установки признаков.

Варианты индивидуальных заданий

Таблица

N/N n/n	Число Разрядов АЛУ	Список команд	Коды операндов	
			A	B
0	3	$A+B, A-1, A-B, A \oplus B, \bar{A}$	011	011
1	4	$A-1, A \vee B, A/B, A+B+1, B$	0110	0111
2	2	$A-B, B-1, B+1, \bar{A} \cdot B, A$	01	11
3	4	$A-B+1, A \oplus B, A \downarrow B, \bar{A}$	0111	0011
4	2	$B-1, A+B, A \cdot \bar{B}, A$	10	01
5	3	$B+1, A-B, A \oplus B, A$	011	101
6	4	$A+B, A-B, B-1, A/B, B$	1110	0010
7	3	$A-1, A+B+1, A \vee B, A$	011	011
8	4	$A-B, A+B+1, B+1, \bar{A} \cdot B, B$	1111	0101
9	2	$B-1, A+B+1, A \cdot \bar{B}, B$	11	10
10	3	$A-1, A+B-1, A \cdot B, \bar{A}, \bar{B}$	111	010
11	2	$B-1, A+B+1, A/B, A$	01	01
12	4	$B+1, A-B+1, A \oplus B, \bar{A}$	0101	0111
13	2	$A-B, A+B+1, B+1, A \downarrow B, B$	11	10
14	3	$B+1, A-B, A+B, B$	011	011
15	4	$A+B, A-B, B+1, A \cdot B, \bar{A}$	1010	0101
16	3	$A+1, A-B-1, A \oplus B, \bar{A} \cdot B, A$	011	110
17	4	$B-1, A+B+1, A/B, B$	0011	0101
18	2	$A-B, A+B+1, A \vee B, A-B, B$	11	01
19	4	$A-1, A+B+1, A \downarrow B, A \oplus B, A$	0111	0011
20	3		101	011

		$A+1, A+B-1, \bar{A} \vee B, A$		
--	--	---------------------------------	--	--

Контрольные вопросы

1. Назовите этапы синтеза АЛУ?
2. Назначение АЛУ.
3. Для чего служат флаговые регистры?
4. Как производится запись признаков во флаговый регистр?
5. Назначение буферных регистров А и А' , В и В' в схеме АЛУ?
6. Для чего используется схема MS при синтезе АЛУ?
7. Как производится использование сумматора при выполнении арифметической операции сложения?
8. Как используется сумматор при выполнении арифметической операции вычитания?
9. Как используется сумматор при выполнении арифметической операции декремент?
10. Как используется сумматор при выполнении арифметической операции инкремент?

ПРАКТИЧЕСКАЯ РАБОТА №28-33

Тема: «Т-триггер»

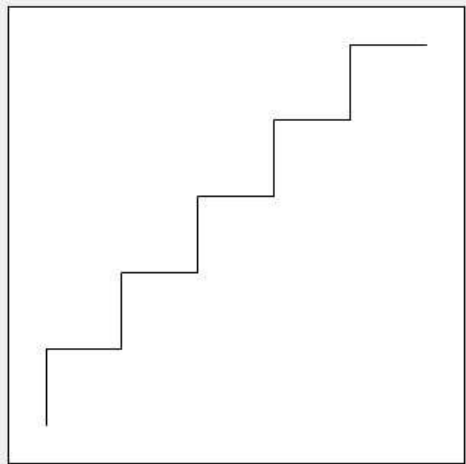
Цель: закрепление навыков программного управления исполнителем алгоритмов.

Оборудование: ПК, исполнитель «Стрелочка».

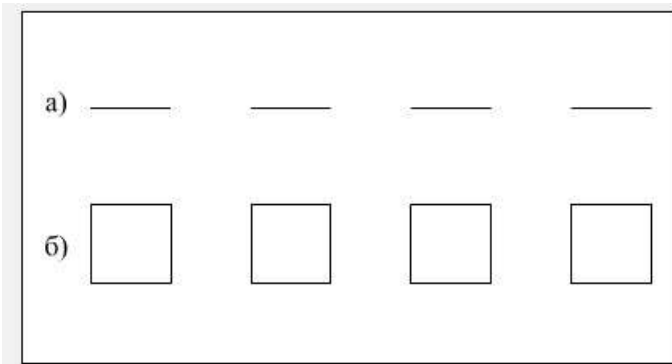
Порядок выполнения работы

Ход работы

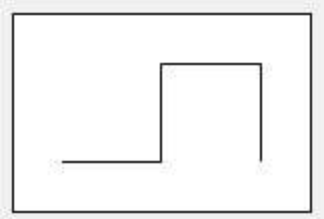
1. С помощью исполнителя алгоритмов стрелочка выполните следующий рисунок.



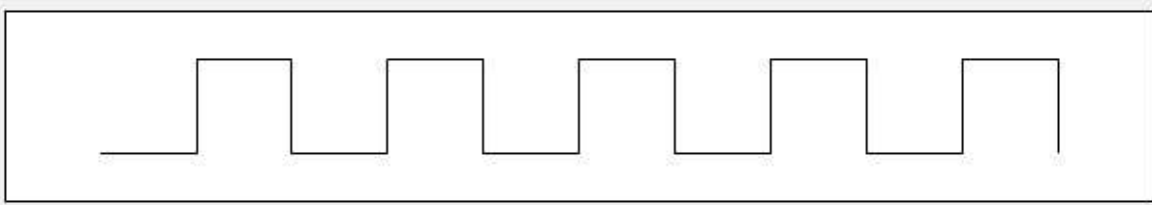
2. Написать программы для рисования следующих рисунков на всю длину листа с использованием вспомогательных алгоритмов (подпрограмм).



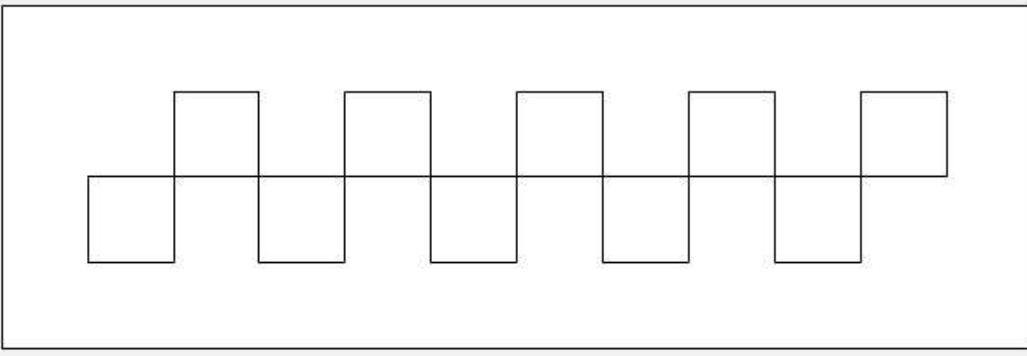
3. Написать подпрограмму для изображения следующей фигуры



4. Используя подпрограмму из предыдущего задания, отобразить следующий элемент.



5. Составить программу для рисования следующей фигуры



ПРАКТИЧЕСКАЯ РАБОТА № 34

Тема: «Шифраторы»

Цель: изучить принцип организации и функционирования современных микропроцессоров.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

1. Микропроцессор

Микропроцессор выполняют всю обработку информации в компьютере. Микропроцессор дешифрирует и выполняет команды программы, организует обращения к оперативной памяти, в нужных случаях инициирует работу периферийных устройств, воспринимает и обрабатывает запросы, поступающие из устройств машины и из внешней среды (“запросы прерывания”).

Для определений временных соотношений между различными этапами операции используется понятие *машинного такта*. Машинный такт определяет интервал времени, в течение которого выполняется одна или одновременно несколько микроопераций. Границы тактов задаются синхросигналами, вырабатываемыми специальной схемой — генератором синхросигналов. Также данная характеристика микропроцессора носит название *тактовая частота*, которая определяет, сколько микроопераций процессор выполнит за одну секунду.

2. Общая классификация микропроцессоров

В настоящее время насчитывается большое количество разнообразных процессоров. Приведем их общую классификацию.

По числу больших интегральных схем (БИС) в составе микропроцессора различают:

- однокристалльные микропроцессоры;
- многокристалльные микропроцессоры;
- многокристалльные секционные микропроцессоры.

Однокристалльные микропроцессоры получают при реализации всех аппаратных средств процессора в виде одной БИС или СБИС (сверхбольшой интегральной схемы). При усложнении микропроцессора приходится разбивать его на отдельные блоки. В этом случае каждый блок реализуется на отдельном кристалле, в результате чего процессор становится многокристалльным.

Многокристалльные секционные микропроцессоры получают в том случае, когда отдельные блоки процессора приходится логически разбивать дополнительно на секции. Секционность микропроцессоров дает возможность наращивать разрядность обрабатываемых данных или усложнять устройство управления микропроцессора.

По назначению различают:

- универсальные микропроцессоры;
- специализированные микропроцессоры.

Универсальные микропроцессоры могут быть применены для решения широкого круга разнообразных задач. Специализация МП, т.е. его проблемная ориентация на ускоренное выполнение определенных функций позволяет резко увеличить эффективную производительность при решении только определенных задач.

По виду обрабатываемых входных сигналов различают:

- цифровые микропроцессоры;
- аналоговые микропроцессоры.

Цифровые микропроцессоры работают с информацией представленной в виде числовых значений (дискретная форма). Аналоговые микропроцессоры работают с информацией, которая представлена в аналоговой форме, т.е. в виде непрерывного ряда значений.

По характеру временной организации работы микропроцессоры делятся на:

- синхронные микропроцессоры;
- асинхронные микропроцессоры.

Синхронные микропроцессоры – это микропроцессоры, в которых начало и конец выполнения операций задаются специальным устройством управления. Т.е. если в микропроцессоре присутствует устройство управления, то он относится к синхронным.

Асинхронные микропроцессоры позволяют начало выполнения каждой следующей операции определить по сигналу фактического окончания выполнения предыдущей операции.

По количеству выполняемых программ различают:

- однопрограммные микропроцессоры;
- многопрограммные микропроцессоры.

В однопрограммных микропроцессорах выполняется только одна программа. Переход к выполнению другой программы происходит после завершения текущей программы. В много- или мультипрограммных микропроцессорах одновременно выполняется несколько программ.

Содержание работы

Практическое задание

1. Используя свой персональный компьютер (или его макет) определите модель используемого микропроцессора в вашем персональном компьютере. Запишите **ответ в отчет**.
2. Определите фирму-производителя микропроцессора. Запишите **ответ в отчет**.
3. Определите тактовую частоту микропроцессора. Запишите ответ в отчет.
4. Определите установочный разъем микропроцессора (можете использовать Интернет для поиска информации). Запишите **ответ в отчет**.
5. Самостоятельно проведите классификацию имеющегося микропроцессора.
Запишите ответ в отчет.

ПРАКТИЧЕСКАЯ РАБОТА № 35

Тема: «Шифраторы»

Цель: изучить структуру современных микропроцессоров.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

1. Архитектуры микропроцессоров

Существует две основные архитектуры современных процессоров – это архитектуры CISC и RISC. **CISC** (CISC - Complete Instruction Set Computer) – это процессоры с полным набором команд, **RISC** (RISC - Reduced Instruction Set Computer) – это процессоры с сокращенным набором команд. Разберемся, чем одна архитектура отличается от другой.

Набор команд CISC был разработан для удобства программистов, которые вынуждены были писать программы для компьютеров на языке Ассемблер. Для ускорения процесса разработки программ в систему команд CISC были введены удобные команды, которые как бы представляли собой подпрограммы. В итоге, команды CISC-процессора имеют разную длину и время выполнения. К тому же CISC-процессор отличается невысокой производительностью, т.к. для выполнения некоторых команд требуется несколько машинных тактов.

В общем случае для **CISC-процессоров** характерно следующее:

- небольшое число регистров общего назначения;
- большое количество машинных команд, которые выполняются за много тактов;
- большое количество методов адресации;
- большое количество форматов команд различной разрядности;
- наличие команд обработки типа регистр-память.

К процессорам класса CISC относятся широко распространенные в персональных компьютерах процессоры фирм Intel, AMD, Cyrix.

В процессорах с набором команд **RISC** все команды имеют одинаковую длину и формат, а также простую адресацию памяти. Каждая команда выполняет только простые действия за один такт.

В общем случае для **RISC-процессоров** характерно следующее:

- отделение команд обработки данных от команд работы с памятью;
- выполнение любой команды занимает небольшое количество машинных тактов (предпочтительно один машинный такт);
- логика выполнения команд с целью повышения производительности ориентируется на аппаратную, а не на микропрограммную реализацию;
- используются команды фиксированной длины и фиксированного формата;
- наличие большого числа регистров, что позволяет большему объему данных храниться в регистрах на процессорном кристалле большее время. Это значительно увеличивает быстродействие процессора.

Проще говоря, сущность архитектуры RISC состоит в том, что в процессоре выполняются простые команды за один такт. При этом любую сложную команду можно разбить на несколько простых. Выполнение простых команд происходит быстрее, чем сложных, причем выполнение простых команд может происходить параллельно. Поэтому быстродействие RISC процессоров в общем случае выше, чем у CISC.

Считается, что в будущем процессоры с архитектурой RISC заменят менее перспективные процессоры с архитектурой CISC.

Существует еще одно понятие архитектуры процессоров, которые мы также рассмотрим. Наверняка вы часто встречались с термином «**x86**» (мы его несколько раз упомянули выше), или «Intel-совместимый процессор». Что за этим скрывается на самом деле? Современный x86-процессор – это процессор, способный исполнять машинный код архитектуры IA32 (архитектура 32-битных процессоров Intel). Этот код исполнял процессор Intel 80386 (известный как «386-й»). В настоящее время всё программное обеспечение для ПК разрабатывается именно для x86-процессоров. Оно выполняется на любом x86-процессоре, независимо от того, кто его произвел.

Кроме того, у архитектуры IA32 существуют дополнительные наборы команд от разработчика, компании Intel: MMX, SSE, SSE2 и SSE3. Также существуют неофициальные расширенные наборы команд: EMMX, 3DNow! и Extended 3DNow! – их разработала компания AMD.

Все перечисленные дополнительные наборы команд служат для увеличения быстродействия при выполнении некоторых операций. Одна команда из дополнительного набора, как правило, выполняет действие, для которого понадобилась бы небольшая программа, состоящая из команд основного набора.

2. Понятия irq и dma

Прерывание IRQ (Interrupt ReQuest - запрос прерывания) – это сигнал, по которому процессор узнает о совершении некоторого события, на которое необходимо “обратить” внимание. Пусть, к примеру, микропроцессор выполняет некоторую программу, и пусть в это время в каком-то внешнем устройстве произошло событие, на которое нужно обратить внимание, (например, на клавиатуре нажата клавиша). Естественно, ждать пока закончится выполнение текущей программы нельзя, она может работать еще долго и за это время может быть нажато много других клавиш, так что информация о первой из нажатых клавиш будет потеряна. Надо сразу, оперативно прореагировать на это событие.

Получив сигнал прерывания, микропроцессор прерывает выполнение текущей последовательности команд, а вместо нее начинает выполнять другую последовательность, соответствующую данному прерыванию.

Все прерывания делятся на три группы:

- аппаратные прерывания;
- логические прерывания;
- программные прерывания.

Аппаратные прерывания связаны с запросами от внутренних или периферийных устройств. Логические возникают при работе самого микропроцессора. Программные инициируются выполняемой программой.

Для IBM PC AT на базе процессоров Pentium предусмотрено было **16 линий IRQ**, часть которых заняты внутренними устройствами, а остальные используются внешними или не используются. В настоящее время число прерываний составляет несколько десятков.

Таким образом, число периферийных устройств, подключаемых к персональному компьютеру с использованием прерываний IRQ, не может превышать пяти.

DMA (Direct Memory Access) – это режим прямого доступа к памяти, когда периферийное устройство связано с оперативной памятью компьютера непосредственно, минуя микропроцессор. Этот режим наиболее эффективен, когда требуется высокая скорость обмена при передаче большого количества информации.

На IBM PC AT есть **8** независимых каналов DMA. Каналы DMA распределены следующим образом:

- 0 - микропроцессор;
- 1 - не используется;
- 2 - контроллер флоппи-диска;

- 3 - не используется;
- 4 - не используется;
- 5 - не используется;
- 6 - не используется;
- 7 - не используется.

Таким образом, к ПК можно подключить 5 различных устройств, которые используют режим DMA. При этом следует помнить, что не все устройства, требующие применения прерываний IRQ, используют DMA.

Содержание работы

Практическое задание

1. Загрузите ПК. Вызовите программу **Сведения о системе (Пуск – Программы – Стандартные – Служебные** или файл **MSINFO32.EXE**).
2. Используя программу **Сведения о системе**, выпишите в отчет общее число прерываний IRQ вашего компьютера.
3. **Выпишите в отчет** основные устройства, которые используют прерывания IRQ вашего компьютера.
4. **Укажите в отчете**, сколько свободных прерываний есть в вашем компьютере.
5. Выпишите в отчет все занятые каналы DMA вашего компьютера.
6. Выпишите в отчет все свободные каналы DMA вашего компьютера.
7. Определите, какой процессор в настоящее время является наиболее оптимальным при выборе компьютера для дома, для выполнения графических работ, для офисной работы. Обоснуйте и докажите свой ответ.

ПРАКТИЧЕСКАЯ РАБОТА № 36

Тема: «Дешифраторы»

Цель: изучить структуру и группы команд, выполняемые процессором.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Программа помещается в оперативную память вместе с данными. Каждая команда хранится в ячейке или группе ячеек и имеет свой *адрес*.

Команда – это элементарная инструкция машине, выполняемая ею автоматически без каких либо дополнительных указаний и пояснений.

Все команды имеют одинаковую *структуру*. Машинная команда состоит из двух частей: операционной и адресной.

Код ОПерации (КОП)	Адреса
--------------------	--------

1. Код операции (операционная часть) определяет, какую команду нужно выполнить

2. Операнд (адресная часть) – над чем выполняется операция

Операционная часть команды (КОП) – это группа разрядов в команде, предназначенная для представления кода операции машине.

Адресная часть – это группа разрядов в команде, в которых записывают коды адресов операндов.

Операнд может быть простым и составным. Он может содержать в себе адресную часть, которая определяет, где хранятся данные и куда поместить результат операции.

В зависимости от количества используемых адресов, различают одноадресные, двух-, трех-, четырехадресные и безадресные команды.

- *Одноадресные команды.* Здесь указывается, где находится одно из чисел, второе должно быть помещено в АЛУ. Для этого существуют специальные команды пересылки данных.
- *Двухадресные команды.* Оба операнда в памяти, их адреса указаны. Результат заносится по одному из адресов.
- *Трехадресные команды.* Два адреса являются операндами, а третий служит для помещения туда результата.
- *Четырехадресные команды.* Два адреса – операнды, третий – результат, четвертый – адрес следующей команды.
- *Безадресные команды.* используются для выполнения служебных операций (очистить экран, заблокировать клавиатуру и так далее).

Команды выполняются друг за другом, а также могут быть выполнены ветвления и циклы.

Группы команд микропроцессора

Семейство микропроцессоров фирмы Intel от 8086 до Pentium имеет базовую систему команд, в состав которых входят:

Команды пересылки данных:

внутри процессора (Mov, Push, Pop и др.)

ввода-вывода (In, Out)

Арифметические команды:

основные (сложение, вычитание, умножение, деление)

дополнительные (INC, DEC)

Логические команды:

сдвиг, дизъюнкция, конъюнкция, и/или и др.

Обработка строковых данных:

пересылка, сканирование, сравнение, слияние.

Передачи управления:

безусловный переход, условный переход, прерывания, переход с возвратом.

Управления:

нет операции, внешняя синхронизация и так далее.

Каждая команда имеет много модификаций, чаще всего определяемых режимом адресации операндов.

Типы операндов команд

- Регистровые операнды указываются именами используемых регистров процессора.
- Непосредственные являются всегда числовыми величинами и могут быть в различных системах счисления. Числа различаются по последней букве, сопровождающей число:

b – двоичное число;

q – восьмеричное число;

d – десятичное число;

h – шестнадцатеричное число.

- Операнды в памяти могут указываться с помощью адресов ячеек, символическими именами, константами.

Различные комбинации этих элементов в команде называются *способами адресации*. Например, команда mov (переслать числа) может иметь следующие способы адресации:

mov	r, r
mov	r, m
mov	m, r
mov	imed, r
mov	imed, m
mov	Sr, m
mov	Sr, r
mov	m, Sr
mov	r, Sr

Где r – РОН (регистр общего назначения); m – адрес в памяти;
imed – число; Sr – с

Содержание работы

Задание:

1. Изучить материал теоретической части;
2. Записать микропрограмму суммирования двух чисел;
3. Сделать вывод о проделанной работе.

ПРАКТИЧЕСКАЯ РАБОТА № 37

Тема: «Дешифраторы»

Цель: изучить структуру и группы команд, выполняемые процессором.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Содержание работы

Задание:

Составить программу для условия: переписать 10000 байтов начиная с адреса А в другое место памяти начиная с адреса В. Оба эти имени относятся к сегменту данных, на начало которого указывает регистр A_s .

ПРАКТИЧЕСКАЯ РАБОТА №38

Тема: «Полусумматоры»

Цель: знакомство с компонентной структурой современного персонального компьютера.

Оборудование: ПК с операционной системой Windows.

Порядок выполнения работы

Задание:

Определение ключевых параметров аппаратного обеспечения рабочего ПК.

Ход работы

Определение типа и характеристик центрального процессора и объёма оперативной памяти.

На рабочем столе найдите иконку **Мой компьютер**. Через контекстное меню вызовите команду **Свойства** и откройте (если она не открыта) вкладку **Общие**.

Мой компьютер ☐ Свойства (в контекстном меню) ☐ Общие
MyComputer ☐ Properties ☐ General

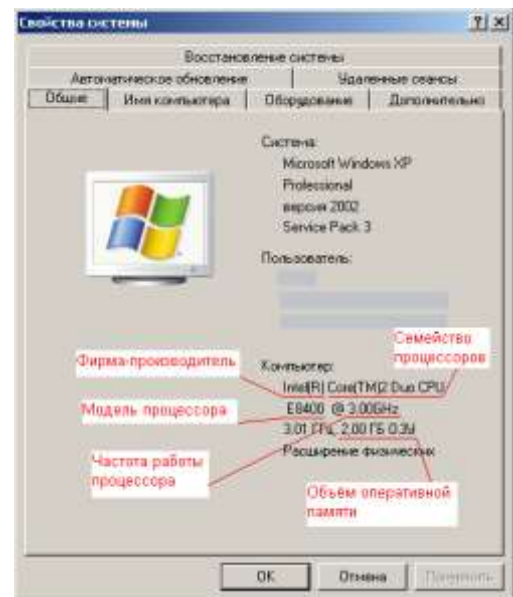
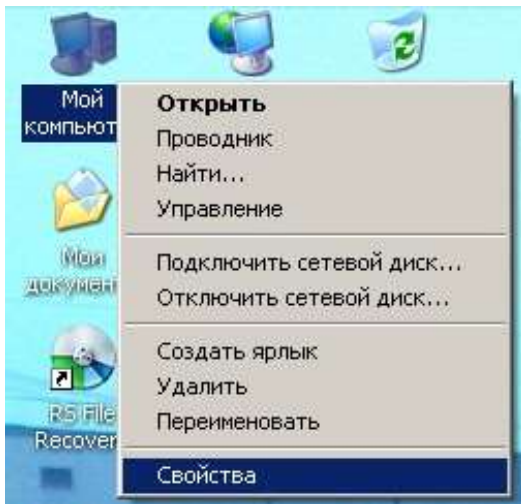
В открывшемся окне найдите информацию о процессоре и оперативной памяти. Для приведённого на рисунках ниже примера:

Фирма производитель процессора: Intel Семейство процессоров: Core2 DUO

Модель процессора: E8400

Частота работы процессора: 3 ГГц Объём оперативной памяти: 2 ГБ

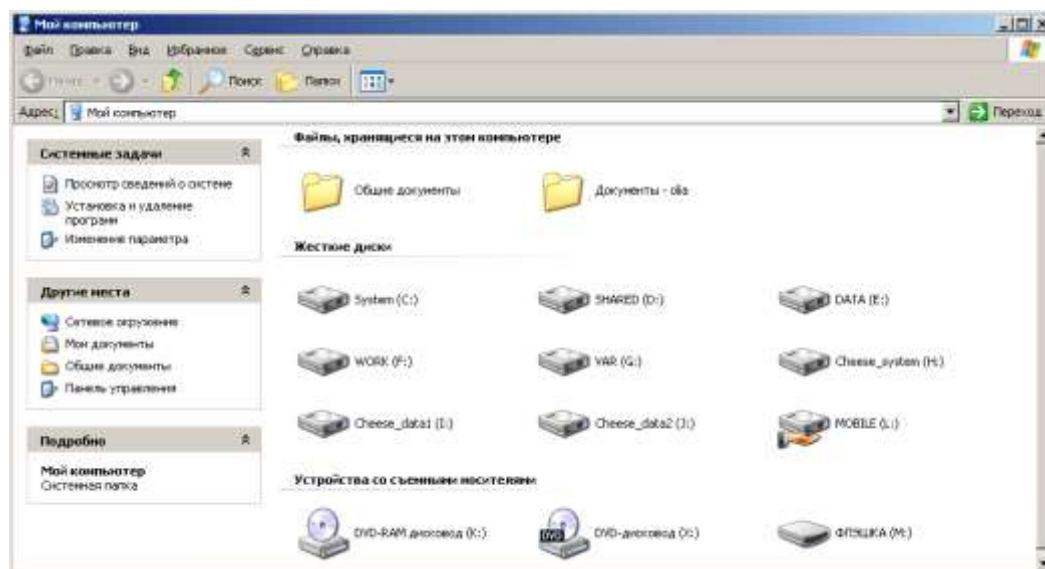
Приведите в отчёте данные о процессоре и оперативной памяти для Вашего рабочего компьютера в лаборатории и для Вашего домашнего компьютера. Сравните полученные характеристики. Какой из компьютеров имеет потенциально бóльшую производительность?



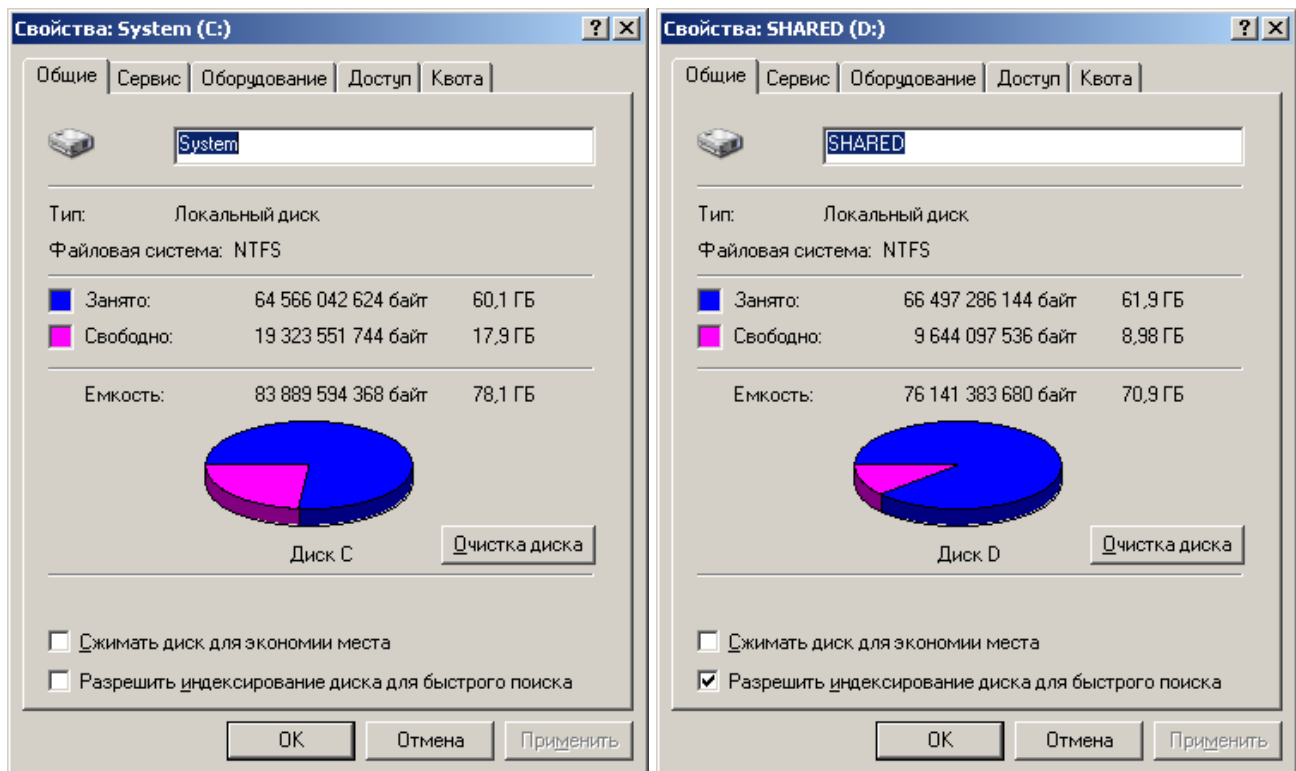
Определение объема памяти на жестких накопителях

На рабочем столе найдите и дважды щелкните на иконку **Мой компьютер**. В появившемся окне будут показаны иконки для всех внешних накопителей, подключённых в настоящий момент к системе.

My computer (на рабочем столе) ☐ контекстное меню жесткого диска ☐ Properties



Вызовите окно **Свойства** через пункт меню **Свойства** в контекстном меню одного из дисков. В появившемся окне найдите информацию об общем объеме диска, о занятом и свободном месте. Для дисководов без дисков объем равен 0.

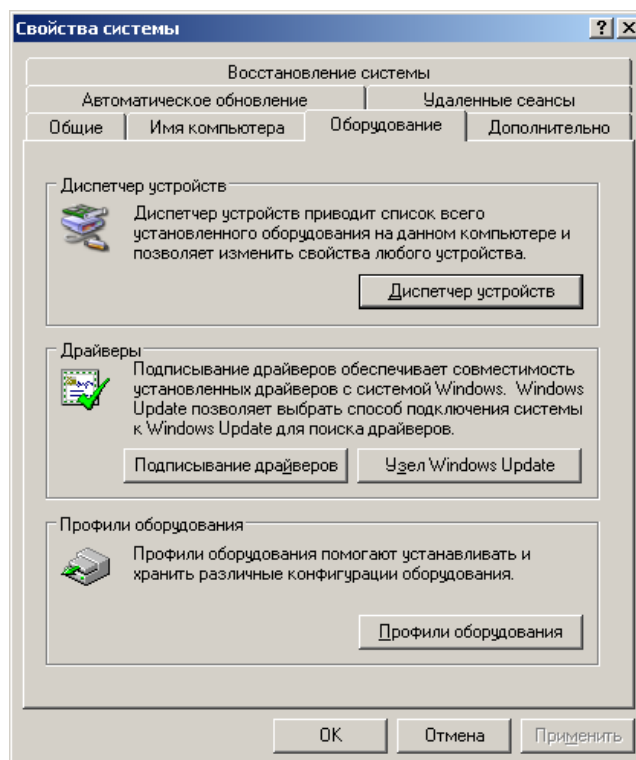


Определение количества физических накопителей, подключённых к компьютеру. Определение модели видеокарты.

Фактическое количество физических накопителей, подключённых к компьютеру, может быть меньше показанного в папке **Мой компьютер**, поскольку один физический накопитель может быть разбит на несколько разделов, отображающихся независимо друг от друга.

На рабочем столе найдите иконку **Мой компьютер**. Через контекстное меню вызовите команду **Свойства**, откройте вкладку **Оборудование** и нажмите кнопку **Диспетчер устройств**.

Мой компьютер □ Свойства □ Оборудование □ Диспетчер устройств
MyComputer □ Properties □ Hardware □ Device manager



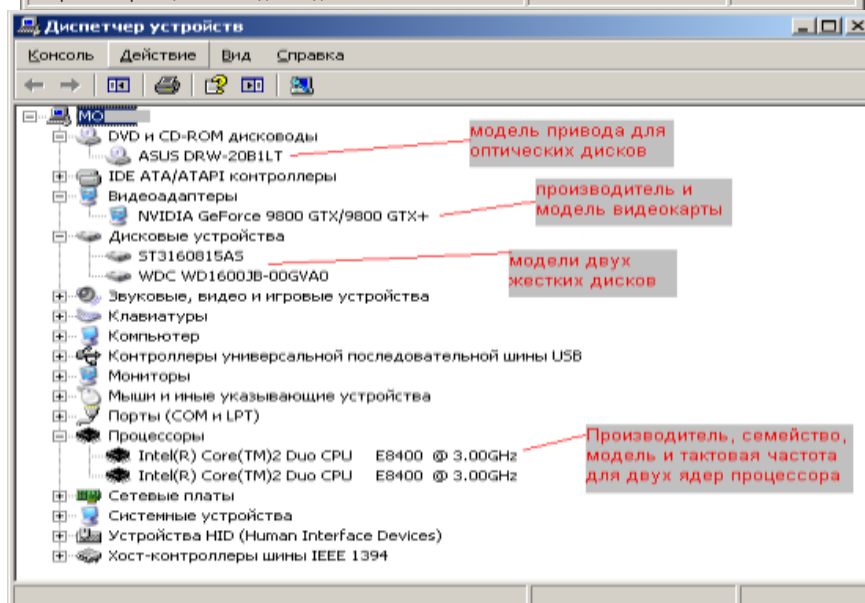
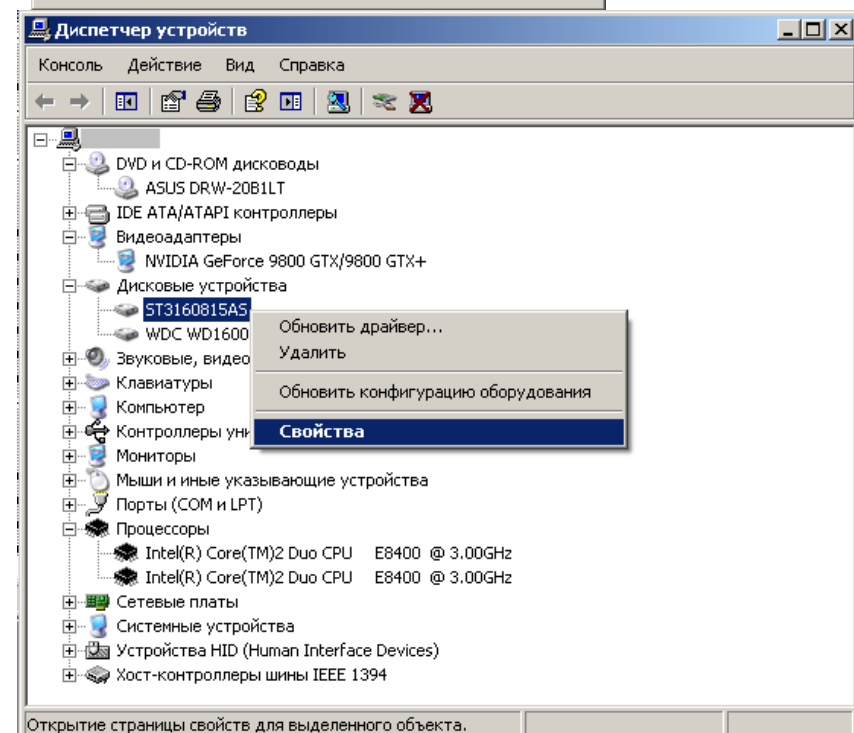
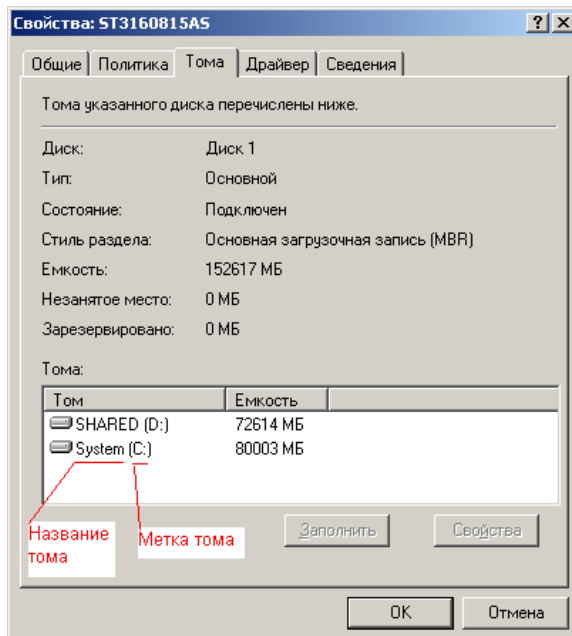
В появившемся окне найдите раскрывающееся меню **Дисковые устройства** (ищите иконку с жёстким диском). Раскройте меню, нажав на плюс. В раскрывшемся меню будут показаны все физически подключённые к компьютеру жесткие диски. Первые две буквы в названии винчестера кодируют название производителя. Остальные символы в зависимости от производителя каким-то образом кодируют в том числе и объём диска. Например, диск ST3160815AS, как и диск WD1600JB-00GVA0, имеет объём 160 ГБ.

Производители винчестеров (жёстких дисков): WDC – Western Digital, ST – Seagate, Samsung, Hitachi.

Вызовите окно **Свойства** через контекстное меню для одного из дисков и откройте вкладку **Тома**. Нажмите кнопку **Заполнить**.

Контекстное меню дискового устройства □ Свойства □ Тома □ Заполнить
Контекстное меню дискового устройства □ Properties □ Volumes □ Populate

В нижней части окна появится информация о виртуальных разделах - томах на физическом диске. Сопоставьте все физические жесткие диски всем виртуальным жестким дискам в папке **Мой компьютер**.



Приведите в отчёте полный список внешних накопителей для Вашего рабочего компьютера в лаборатории и для Вашего домашнего компьютера. Для каждого накопителя укажите принадлежность к физическому жесткому диску, общую ёмкость и процент свободного места. Результаты приведите в таблице по примеру.

Как Вы думаете, достаточно ли имеющегося свободного места на дисках для полноценной работы?

К компьютеру из примера на рисунке подключены следующие внешние накопители:

Физический накопитель	Название тома	Метка тома	Общая ёмкость, ГБ	Процент свободного места, %
ST3160815AS	System	C	78.1	$100 * 17.9 / 78.1 = 23$
ST3160815AS	SHARED	D	70.9	13
	...			

ПРАКТИЧЕСКАЯ РАБОТА №39

Тема: «Полусумматоры»

Цель: знакомство с компонентной структурой современного персонального компьютера.

Оборудование: ПК с операционной системой Windows.

Порядок выполнения работы

Задание:

Подбор аппаратной конфигурации ПК по индивидуальному заданию.

Ход работы

Для подбора компонентов Вы можете воспользоваться сервисом Конфигуратор системного блока на сайте www.ulmart.ru (<http://www.ulmart.ru/configurator.php#configer>) или на сайте key.ru <http://key.ru/shop/devices/>.

Выберите конфигурацию по заданию. Проверьте согласованность параметров выбранных компонентов в строках, помеченных цветом. Обоснуйте выбор каждого компонента в поле для примечания в строке **Выбранная модель**. Приведите значения дополнительных характеристик для выбранных компонентов в остальных строках.

Дополните конфигурацию периферийным оборудованием по желанию - наушники, микрофон, принтер, сканер и пр. Подсчитайте суммарную стоимость вы-

бранной комплектации.

Варианты

0. Intel Core i7, OEM, видео от NVIDIA
1. Intel Core i7, BOX, встроенное видео
2. Intel Core i5, OEM, видео от AMD
3. Intel Core i5, BOX, встроенное видео
4. AMD A10, BOX, видео от NVIDIA
5. AMD A8, BOX, видео от AMD
6. AMD A10, OEM, встроенное видео
7. AMD A8, OEM, видео от NVIDIA

Пример. Сборка для 0 варианта. Жирным цветом отмечены фиксированные заданием параметры. Цветом отмечены параметры, которые требуют согласования между компонентами.

Компонент	Характеристика	Значение	Примечание
Процессор	Производитель	Intel	По заданию
	Модель	Intel Core i7	
	Версия поставки	OEM - без кулера	
	Выбранная модель	Intel Core i7-3820 3.6/10Mb LGA2011	Средний ценовой диапазон
	Ссылка	http://key.ru/shop/devices/processors/intel_core_i7-3820_3_6_10mb_gla2011_box	
	Тип разъёма (Socket)	LGA2011	
	Частота собственная	3.6 ГГц	
	Число ядер	4	
	Кэш память (наличие и размер)	10 МБ	
	Мощность	130 Вт	
	Стоимость	10 190 р	
Кулер процессора	Тип разъёма (Socket)	LGA2011	Необходим, поскольку процессор поставляется без вентилятора
	Выбранная модель	CPU cooler Cooler Master Hyper 412 Slim	Самый тихий
	Ссылка	http://key.ru/shop/devices/computer_cooling/kulery_dlya_p	

Материнская плата		rocessorov/cpu_cooler_cooler_master_hyper_412_slim	
	Уровень шума	8 дБ	
	Стоимость	1 899	
	Тип разъёма (Socket)	LGA2011	
Материнская плата	Выбранная модель	MB Gigabyte GA-X79-UD3	самая недорогая модель без встроенного видео с достаточным запасом под оперативную память
	Ссылка	http://key.ru/shop/devices/motherboards/mb_gigabyte_ga-x79-ud3/	
	Встроенная видеокарта	нет	
	Интерфейс видеокарты	2 шт. PCI Express 3.0 x16	
	Количество слотов памяти	2	
	Тип модулей памяти	DDR3	
	Частота системной шины	2400/2133/1866/1600/1333/1066 МГц	
	Форм-фактор	ATX	
	Разъёмы	Сетевая LAN-розетка RJ-45 2 порта USB 3.0/2.0 1 порт PS/2 для подключения клавиатуры и мыши 8 портов USB 2.0/1.1 6 аудио разъемов 1 x SPDIF out (коаксиальный) Оптический выход SPDIF-интерфейса	пришлось искать дополнительные характеристики от производителя не совпали с характеристиками на сайте магазина http://www.gigabyte.ru/products/page/mb/ga-x79-ud3rev_10/specs/
	Стоимость	6890	
	Производитель	NVIDIA	По заданию
Видеокарта	Интерфейс	до 2 PCI Express 3.0 x16	
	Выбранная модель	2048M Asus GeForce GTX660 DDR5	Средний ценовой диапазон

	Ссылка	http://key.ru/shop/devices/videokarty/2048m_asus_geforce_gtx660_ddr5_2xdvi_hdmi_dp_pci-e/	
	Чипсет	GeForce GTX 660	
	Частота	1072 МГц	
	Объём памяти	2048 Мб	
	Видеовыходы	2xDVI HDMI DP	
	Мощность	150 Вт	информация с сайта производителя
	Стоимость	8090	
	Длина видеокарты	10 дюймов = 255 мм	с сайта производителя
Оперативная память	Тип модулей памяти	DDR3	
	Количество слотов памяти	2	
	Частота системной шины	2400/2133/1866/1600/1333/1066 МГц	
	Выбранная модель	Модуль памяти DDR3 16Gb 2133MHz Kingston XMP Predator CL11 Kit of 2	Максимальный объём с учетом ограничения на количество слотов памяти
	Ссылка	http://key.ru/shop/devices/memory/ddr3_16gb_2133mhz_kingston_xmp_predator_non-ecc_cl11_kit_of_2/	
	Стоимость	5090	
Привод CD/DVD	нет		Нет необходимости
Жесткий диск HDD	Интерфейс	SATA	Современный стандарт дефакто, должен совпадать с интерфейсом на материнской плате чем быстрее - тем лучше
	Выбранная модель	WD4001FAEX	Самый дешевый из самых больших
	Ссылка	http://key.ru/shop/devices/ustroystva_hraneniya_i_chteniya_dannyh/hdd_ssd/vinchester_4tb_wd_caviar_black_wd4001faex/	
	Объём	4 ТБ	

	Стоимость	9290	
Корпус	Минимальная требуемая мощность	300 Вт	сумма мощности процессора и видеокарты
	Форм фактор	ATX	или совместимый
	Длина видео-карты	10 дюймов = 255 мм	
	Выбранная модель	Codegen Q3339-A2 Black ATX	По внешнему виду
	Наличие блока питания	да	
	Ссылка	http://key.ru/shop/devices/korpusa/korpus_codegen_q3339-a2_black_atx/	
	Стоимость	1490	

ПРАКТИЧЕСКАЯ РАБОТА №40

Тема: «Сумматоры»

Цель: изучить архитектуру и принципы построения микроконтроллера AVR ATMEGA128.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Основные характеристики микроконтроллера AVR ATMEGA128.

AVR-архитектура объединяет высокопроизводительный RISC-процессор с отдельным доступом к памяти программ и данных, 32 регистра общего назначения, каждый из которых может работать как регистр-аккумулятор, и развитую систему команд с фиксированной (16-бит) длиной. Конвейерная архитектура с одновременным исполнением текущей и выборкой следующей команды позволяет выполнять большинство команд за один машинный цикл, что обеспечивает производительность до 1 MIPS на каждый МГц тактовой частоты.

Ниже приводятся основные характеристики микроконтроллера AVR ATMEGA128:

производство по КМОП-технологии с низким энергопотреблением;

тактовая частота может изменяться в широких пределах от 0 до 16 МГц (полностью статическая архитектура);

ядро микроконтроллера основано на RISC архитектуре с двухступенчатым конвейером, обеспечивающим выполнение одной команды за один машинный цикл;

гарвардская архитектура с отдельной памятью программ и данных;
регистровый файл содержит 32 регистра общего назначения;
все регистры общего назначения непосредственно подключены к АЛУ;
совмещенная архитектура ввода/вывода (регистры общего назначения и порты ввода/вывода находятся в адресном пространстве ОЗУ данных);
наличие программного стека;
наличие в составе АЛУ аппаратного умножителя;
19 источников внутренних прерываний, 8 источников внешних прерываний;
Объем FLASH-памяти программ: 128 кБт;
Объем статической оперативной памяти (ОЗУ) : 4 кБт
Объем памяти данных на основе электрически-стираемого ПЗУ (EEPROM): 4 кБт;
Интерфейсы программирования: SPI и JTAG;
Напряжение питания: 4.5–5.5 В;
Периферийные устройства:
8-разрядные параллельные порты ввода/вывода;
8 и 16 разрядные таймеры-счетчики;
широотно-импульсные модуляторы;
аналоговые компараторы,
10-разрядный 8-канальный АЦП,
Встроенный универсальный асинхронный передатчик (USART).
Высокая производительность, наличие развитой подсистемы ввода/вывода и широкого спектра встроенных периферийных устройств позволяют отнести микроконтроллеры AVR ATMEGA128 к классу наиболее функциональных микроконтроллеров для встроенных систем управления, применяемых в бытовой и офисной технике, мобильных телефонах, контроллерах периферийного оборудования (принтеры, сканеры, приводы CD-ROM), портативных медицинских приборах, интеллектуальных датчиках (охранных, пожарных) и др.
ATmega128 – маломощный 8-разр. КМОП микроконтроллер, основанный на расширенной AVR RISC-архитектуре. За счет выполнения большинства инструкций за один машинный цикл ATmega128 достигает производительности 1 млн. операций в секунду/МГц, что позволяет проектировщикам систем оптимизировать соотношение энергопотребления и быстродействия.

Функциональная схема

Ядро AVR сочетает богатый набор инструкций с 32 универсальными рабочими регистрами. Все 32 регистра непосредственно подключены к арифметико-логическому устройству (АЛУ), который позволяет указать два различных регистра в одной инструкции и выполнить ее за один цикл. Данная архитектура обладает большей эффективностью кода за счет достижения производительности в 10 раз выше по сравнению с обычными CISC-микроконтроллерами.

ATmega128 содержит следующие элементы: 128 кбайт внутрисистемно программируемой флэш-памяти с поддержкой чтения во время записи, 4 кбайт

ЭСППЗУ, 4 кбайт статического ОЗУ, 53 линии универсального ввода-вывода, 32 универсальных рабочих регистра, счетчик реального времени (RTC), четыре гибких таймера-счетчика с режимами сравнения и ШИМ, 2 УСАПП, двухпроводной последовательный интерфейс ориентированный на передачу байт, 8-канальный 10-разр. АЦП с опциональным дифференциальным входом с программируемым коэффициентом усиления, программируемый сторожевой таймер с внутренним генератором, последовательный порт SPI, испытательный интерфейс JTAG совместимый со стандартом IEEE 1149.1, который также используется для доступа к встроенной системе отладке и для программирования, а также шесть программно выбираемых режимов уменьшения мощности.

Рисунок – Функциональная схема

Режим холостого хода (Idle) останавливает ЦПУ, но при этом поддерживая работу статического ОЗУ, таймеров-счетчиков, SPI-порта и системы прерываний. Режим выключения (Powerdown) позволяет сохранить содержимое регистров, при остановленном генераторе и выключении встроенных функций до следующего прерывания или аппаратного сброса. В экономичном режиме (Power-save) асинхронный таймер продолжает работу, позволяя пользователю сохранить функцию счета времени в то время, когда остальная часть контроллера находится в состоянии сна. Режим снижения шумов АЦП (ADC Noise Reduction) останавливает ЦПУ и все модули ввода-вывода, кроме асинхронного таймера и АЦП для минимизации импульсных шумов в процессе преобразования АЦП. В дежурном режиме (Standby) кварцевый/резонаторный генератор продолжают работу, а остальная часть микроконтроллера находится в режиме сна. Данный режим характеризуется малой потребляемой мощностью, но при этом позволяет достичь самого быстрого возврата в

рабочий режим. В расширенном дежурном режиме (Extended Standby) основной генератор и асинхронный таймер продолжают работать.

Микроконтроллер производится по технологии высокоплотной энергонезависимой памяти компании Atmel. Встроенная внутрисистемно программируемая флэш-память позволяет перепрограммировать память программ непосредственно внутри системы через последовательный интерфейс SPI с помощью простого программатора или с помощью автономной программы в загрузочном секторе. Загрузочная программа может использовать любой интерфейс для загрузки прикладной программы во флэш-память. Программа в загрузочном секторе продолжает работу в процессе обновления прикладной секции флэш-памяти, тем самым поддерживая двухоперационность: чтение во время записи. За счет сочетания 8-разр. RISC ЦПУ с внутрисистемно самопрограммируемой флэш-памятью в одной микросхеме ATmega128 является мощным микроконтроллером, позволяющим достичь высокой степени гибкости и эффективной стоимости при проектировании большинства приложений встроенного управления.

ATmega128 поддерживается полным набором программных и аппаратных средств для проектирования, в т.ч.: Си-компиляторы, макроассемблеры, программные отладчики/симуляторы, внутрисистемные эмуляторы и оценочные наборы.

Программная модель микроконтроллера avr mega128. Механизм работы с регистрами, памятью и портами ввода/вывода.

В микроконтроллере AVR ATMEGA128 реализована гарвардская архитектура, в соответствие с которой адресные пространства памяти программ и данных физически разделены (доступ к этим областям памяти осуществляется по отдельным шинам). Такая организация позволяет ядру процессора одновременно работать с памятью программ и данных, что повышает быстродействие. Карта распределения памяти в микроконтроллере AVR ATMEGA128 приведена на рисунке 1.1. Память программ представляет собой электрически стираемое перепрограммируемое постоянное запоминающее устройство ППЗУ объемом 128 кБт, выполненное по технологии FLASH – памяти, и предназначена для хранения команд, управляющих функционированием микроконтроллера, а также для хранения констант, не меняющих своих значений в ходе выполнения программы. Так, как длина команды составляет 16 бит, то память программ имеет 16-разрядную организацию. Для адресации памяти программ используется 16-разрядный регистр – программный счетчик PC (Program Counter). Программа выполняется последовательно. Для управления ходом выполнения программы существуют команды перехода, изменяющие соответствующим образом значение PC.

Память данных организована по принципу совмещенной архитектуры ввода/вывода и разделена на 3 части: регистровая память, память портов (регистров) ввода/вывода и статическое ОЗУ (SRAM), расположенные в едином адресном пространстве.

Рисунок – Распределение памяти в микроконтроллере AVR ATMEGA128

Регистровая память (см. рисунок 1.2) включает 32 8-разрядных регистра общего назначения (R0 - R31), объединенных в регистровый файл. Каждый из регистров общего назначения непосредственно связан с АЛУ. АЛУ поддерживает арифметические и логические операции с регистрами, между регистром и константой или непосредственно с регистром. При исполнении арифметической или логической команды за один такт из регистрового файла выбираются два операнда, выполняется действие, и результат возвращается в регистровый файл. Регистровый файл отображается на младшие 32 адреса 0000h-001Fh памяти данных и к его регистрам можно обращаться как к ячейкам памяти. Шесть 8 - разрядных регистров (R26 - R31) могут использоваться как три 16-разрядных регистра-указателя для косвенной адресации (см. рисунок).

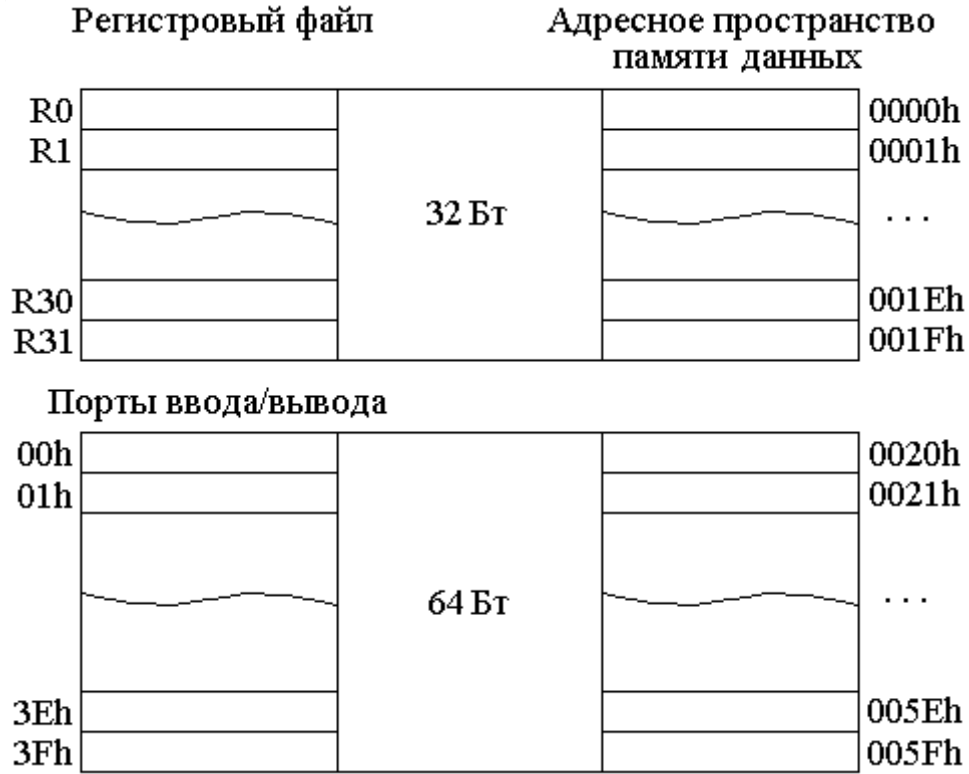


Рисунок – Иллюстрация отображения регистров общего назначения и портов ввода/вывода на адресное пространство памяти данных

X		Y		Z	
27	26	29	28	31	30

Рисунок – 16-разрядные регистры X, Y, Z, использующиеся для косвенной адресации памяти.

Пространство ввода/вывода состоит из 64 адресов портов 0000h-003Fh, предназначенных для взаимодействия с внутренними и внешними устройствами по отношению к микроконтроллеру. Порты ввода/вывода отображаются на область памяти данных с адресами 0020h-005Fh и допускают возможность обращения к ним как к ячейкам памяти. При доступе к порту ввода/вывода как к ячейке памяти к адресу порта необходимо добавить 20h.

Адрес порта ввода/вывода в пространстве памяти часто указывается в скобках после адреса в пространстве портов ввода/вывода. Ввиду того, что основной функцией микроконтроллера является управление внешними устройствами, в таблице 1.1. приводятся названия и адреса (в пространстве портов ввода/вывода) основных интерфейсных портов с указанием режима работы и функций отдельных регистров. Для реализации операций ввода вывода для каждого порта выделяются три регистра PIN*(для ввода данных), PORT* (для вывода данных) и DDR*(для настройки порта).

По адресам памяти 0060h-00FFh расположены 160 дополнительных регистров ввода/вывода.

Непосредственно память данных представляет собой статическое ОЗУ (SRAM) объемом 4 кБт, занимающее диапазон адресов 0100h-10FFh.

Таблица – Порты ввода/вывода микроконтроллера AVR MEGA128 для подключения внешних устройств

Название порта ввода/вывода	Идентификаторы отдельных регистров	Адрес	Режим / функция
PORTA	PINA	19h	IN
	DDRA	1Ah	OUT / DIRECTION
	PORTA	1Bh	OUT
PORTB	PINB	16h	IN
	DDRB	17h	OUT / DIRECTION
	PORTB	18h	OUT
PORTC	PINC	13h	IN
	DDRC	14h	OUT / DIRECTION
	PORTC	15h	OUT
PORTD	PIND	10h	IN
	DDRD	11h	OUT / DIRECTION
	PORTD	12h	OUT
PORTE	PINE	01h	IN
	DDRE	02h	OUT / DIRECTION
	PORTE	03h	OUT
PORTF	PINF	00h	IN
	DDRF	61h	OUT / DIRECTION

	PORTF	62h	OUT
PORTG	PING	63h	IN
	DDRG	64h	OUT / DIRECTION
	PORTG	65h	OUT

Регистр состояния SREG расположен в области ввода/вывода по адресу 3Fh (5Fh) и содержит информацию о текущем состоянии микроконтроллера. Расположение флаговых битов регистра состояния приведено на рисунке.

№ бита	7	6	5	4	3	2	1	0
3Fh (5Fh)	I	T	H	S	V	N	Z	C

Рисунок – Регистр состояния SREG.

Назначение отдельных битов регистра состояния приведено ниже:

- **C** – флаг переноса, устанавливается в 1 при наличии переноса в арифметических операциях;
- **Z** – флаг нуля, устанавливается в 1, если результат операции равен 0;
- **N** – флаг отрицательного результата, устанавливается в 1 при получении отрицательного результата;
- **V** – флаг переполнения, фиксирует выход результата за пределы допустимого диапазона значений;
- **S** – флаг знака, $S = N \text{ xor } V$;
- **H** – флаг дополнительного переноса (из младшей тетрады байта в старшую);
- **T** – флаг для временного хранения бита из регистров общего назначения;
- **I** – управляющий флаг разрешения прерываний, разрешает (1) или запрещает (0) процессору реагировать на аппаратные прерывания.

Система команд микроконтроллера avr mega128.

- Базовый набор команд языка ASSEMBLER для микроконтроллеров AVR содержит 120 инструкций, которые можно разделить на 4 группы: команды пересылки данных; арифметические и логические команды; инструкции для работы с битами; команды управления ходом исполнения программы.
- **Команды пересылки данных.** Группа команд пересылки данных включает в себя инструкции по загрузке значений констант, пересылки данных типа регистр – регистр, регистр – память, регистр – порт ввода/вывода. Команды данной группы являются двух-операндными, причем первым операндом является приемник данных, а вторым – источник данных.
- Команда загрузки констант **ldi R, K** применяется для записи непосредственного значения K в регистр – приемник R. В качестве регистра – приемника могут использоваться регистры общего назначения

R16 – R31. Если константа представлена в двоичной или шестнадцатеричной системах счисления, то перед значением константы К необходимо указать спецификатор системы счисления 0b – для двоичной, 0x – для шестнадцатеричной соответственно. Примеры:

- **ldi R16, 125** загрузка в R16 десятичного числа 125;
- **ldi R20, 0xFF** загрузка в R20 шестнадцатеричной константы FFh;
- **ldi R23, 0b11011001** загрузка в R23 двоичной константы 11011001;
- Команда пересылки данных между регистрами **mov Rd, Rs** используется для пересылки значения из регистра-источника Rs в регистр-приемник Rd. Операнды в команде являются исключительно регистрами общего назначения R0 – R31.
- Примеры:
- **mov R16, R0** загрузка в R16 значения из регистра R0;
- **mov R17, R20** загрузка в R17 значения из регистра R20;
- В командах пересылки данных между регистром и ячейкой памяти используется механизм косвенной адресации, при котором адрес ячейки памяти заносится в один из 16-разрядных регистров X,Y,Z (см. рисунок 1.3). Форматы команд:
- **ld R₈, (R₁₆)** – загрузка данных из ячейки памяти, адрес которой находится в 16-разрядном регистре R₁₆, в регистр общего назначения R₈
- **st (R₁₆), R₈** – загрузка данных из регистра общего назначения R₈ в ячейку памяти, адрес которой находится в 16-разрядном регистре R₁₆,
- **ldd R₈, (R₁₆+Q)** – загрузка данных в регистр общего назначения R₈ из ячейки памяти, адрес которой находится как сумма значения, находящегося в 16-разрядном регистре R₁₆, и смещения Q,.
- **std (R₁₆+Q), R₈** – загрузка данных из регистра общего назначения R₈ в ячейку памяти, адрес которой находится как сумма значения, находящегося в 16-разрядном регистре R₁₆, и смещения Q,.
- **ld R₈, (R₁₆)** – загрузка в регистр общего назначения R₈ данных из ячейки памяти, адрес которой находится в 16-разрядном регистре R₁₆;
- Примеры:
- **ld R2, X** загрузка в R2 значения из памяти по адресу, указанному в X;
- **st Y, R5** загрузка значения из регистра R5 в память по адресу, указанному в Y.
- **ldd R5, Z+1** загрузка в R5 байта из памяти по адресу Z+1;
- **std Y+4, R7** загрузка байта из регистра R7 в память по адресу Y+4.
- Для обращения к портам ввода/вывода в микропроцессоре предусмотрены специальные команды **in** и **out**:
- **in R, P** ввод данных из порта с адресом P в регистр общего назначения R;
- **out P, R** вывод данных из регистра общего назначения R в порт с адресом P;
- Примеры:
- **in R10, 0x15** ввод данных из порта с адресом 15h в регистр общего назначения R10;

- **out 0x2F, R8** вывод данных из регистра общего назначения R8 в порт с адресом 2Fh;
- **Арифметические и логические команды.** Для работы с целыми двоичными числами целочисленное АЛУ микроконтроллера AVR MEGA128 поддерживает более десятка арифметических и логических команд.
- Основными арифметическими командами являются инструкции сложения, вычитания и умножения. Операндами в командах данной группы могут быть только регистры общего назначения. Результат операции (кроме умножения) записывается по адресу первого операнда.
- Основные команды для выполнения операций сложения, вычитания и умножения (для чисел без знака):
- **add Rd, Rs** команда сложения (addition), действие: $Rd = Rd + Rs$;
- **adc Rd, Rs** команда сложения (addition) с учетом переноса, действие: $Rd = Rd + Rs + C$;
- **sub Rd, Rs** команда вычитания (subtraction), действие: $Rd = Rd - Rs$.
- **mul Rd, Rs** команда умножения (multipl.), действие: $R1, R0 = Rd * Rs$;
- Команды изменяют флаги переноса **C**, переполнения **V**, знака **N**, **S**, и нуля **Z**. При выполнении операции умножения **n**-значных чисел местонахождение результата разрядностью $2n$ фиксировано и не указывается в команде: при умножении двух байтов результат размером в слово заносится в регистровую пару (R1, R0), в R0 – младшее слово, в R1 – старшее слово.
- Примеры:
- **add R10, R15** $R10 = R10 + R15$;
- **sub R2, R7** $R2 = R2 - R7$;
- **mul R5, R16** $R1, R0 = R5 * R16$.
- Команды положительного и отрицательного приращения (инкремента и декремента):
- **inc R** инкремент (increment), действие $R = R + 1$;
- **dec R** декремент (decrement), действие $R = R - 1$;
- В качестве операнда в этих командах допускается использовать только регистр общего назначения.
- Примеры:
- **inc R20** действие $R20 = R20 + 1$;
- **dec R16** действие $R16 = R16 - 1$;
- Основными логическими командами микроконтроллера AVR MEGA128 являются:
- **or Rd, Rs** логическое “или”; действие: $Rd = Rd \text{ or } Rs$;
- **and Rd, Rs** логическое “и”; действие: $Rd = Rd \text{ and } Rs$;
- **eor Rd, Rs** “исключающее или”; действие: $Rd = Rd \text{ eor } Rs$.
- Данные команды выполняют операции поразрядного логического “или”, логического “и”, “исключающего или” (см. таблицу 1.2) над операндами,

находящимися в регистрах общего назначения, причем результат записывается по адресу первого операнда. Примеры:

- **or R7, R11** действие: $R7 = R7 \text{ or } R11$;
- **and R10, R11** действие: $R10 = R10 \text{ and } R11$;
- **eor R25, R30** действие: $R25 = R25 \text{ eor } R30$.

Таблица – Таблицы истинности логических операций or, and, eor

or (или)			and (и)			eor (исключ. или)		
Вход		Выход	Вход		Выход	Вход		Выход
A	B	Q	A	B	Q	A	B	Q
0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	1	1
1	0	1	1	0	0	1	0	1
1	1	1	1	1	1	1	1	0

- Указанные команды используются для выполнения операций поразрядного маскирования: **or** – для установки единиц в заданных разрядах, **and** – для установки нулей, **eor** – для выяснения совпадений значений битов первого операнда с маской. Команды изменяют флаги нуля, **Z**, знака **N** и переполнения **V**. Примеры поразрядных логических операций, иллюстрирующие применение механизма маскирования битов, приводятся в таблице.

Таблица – Примеры поразрядных логических операций

Пример поразрядного маскирования or		Пример поразрядного маскирования and		Пример поразрядного маскирования eor	
Rd	xxxxxxxx	Rd	xxxxxxxx	Rd	10100110
Rs	00010010	Rs	10110101	Rs	00010010
Rd=Rd or Rs	xxx1xx1x	Rd=Rd and Rs	x0xx0x0x	Rd=Rd eor Rs	10110100

- Команда поразрядного инвертирования:
- **com R** логическое отрицание; действие: $R = 0b11111111 - R$,
- выполняет изменение значений двоичных разрядов операнда (регистр общего назначения) на противоположные.
- Пример:
- **com R3** действие: $R3 = 0b11111111 - R3$.
- Полный перечень арифметических и логических команд микроконтроллера AVR MEGA128 приводится в Приложении 2.
- **Команды для работы с битами.** Дополняют совокупность логических операций команды сброса, установки и проверки значений отдельных битов.
- Команды сброса **cbi P, n** и установки **sbi P, n** битов предназначены для присваивания значений 0 и 1 отдельным битам портов ввода/вывода

соответственно. Первым операндом в этих командах является адрес порта ввода/вывода, вторым – номер бита (от 0 до 7).

- Примеры:
- **cbi 0x17, 5** действие: $0x17_5 = 0$;
- **sbi 0x40, 1** действие: $0x40_1 = 1$.
- Команда логического сдвига **lsl R** осуществляет сдвиг влево на одну позицию всех битов операнда, а в младший разряд добавляется нуль. Старший бит операнда поступает в флаг переноса **C**. В качестве операнда могут использоваться только регистры общего назначения. Команда **lsr R** выполняет сдвиг вправо на одну позицию всех битов операнда, а в старший разряд добавляется нуль. Младший бит операнда поступает в флаг переноса **C**. Механизм работы и синтаксис аналогичен команде **lsl**. Примеры использования команд логического сдвига:

lsl R17	выполнить логический сдвиг влево всех разрядов в R17;
lsr R9	выполнить логический сдвиг вправо всех разрядов в R9.

- Поменять местами младшую и старшую тетрады байта, загруженного в регистр общего назначения, можно с помощью команды **swap R**. Следующий фрагмент иллюстрирует действие команды **swap**:
- **ldi R19, 0b01001101** Загрузить константу 0b01001101 в регистр R19;
- **swap R19** В результате исполнения команды **swap** в регистре R19 будет сохранено значение 0b11010100.
- Дополняют перечень команд для работы с битами инструкции для сброса/установки значений флаговых разрядов в регистре статуса **SREG**, описание которых приводится в Приложении 2.
- **Команды сравнения, условного и безусловного перехода.** Команда сравнения **cp Rd, Rs** – осуществляет действие **Rd–Rs** и устанавливает флаги нуля **Z**, отрицательного результата **N**, переполнения **V**, переноса **C** и дополнительного переноса **H**. Результат не сохраняется по адресу первого операнда, а только формируются флаги. Операндами могут быть только регистры общего назначения.
- Команды условного перехода вызываются сразу после команд сравнения (или других инструкций, вызывающих изменения битов регистра состояния **SREG**) и на основе анализа флагов осуществляют переход по указанному адресу (метке) в памяти команд.
- Наиболее распространенными среди команд этой группы являются:
- **breq M** переход на M, если равно;
- **brne M** переход на M, если неравно;
- **brlo M** переход на M, если меньше;
- **brsh M** переход на M, если больше или равно.
- Пример совместного использования команд сравнения и условного перехода:
- **cp R1, R5** сравнить значения в регистрах R1 и R5;

- **breq lbl1** выполнить переход на метку lbl1, если значения в регистрах R1 и R5 равны ($R1-R5=0$).
- Команда **rjmp M** осуществляют безусловный переход по указанному 8-разрядному адресу (метке, label) в памяти команд. Пример:
- **rjmp lbl2** безусловный переход на метку lbl2.
- Команда **jmp M** осуществляют безусловный переход по указанному 16-разрядному адресу (метке, label) в памяти команд. Пример:
- **rjmp lbl3** безусловный переход на метку lbl3.
- Полный перечень команд сравнения и перехода приводится в таблицах

Порядок выполнения работы

Занятие 1.

Проработайте теоретические материалы и конспекты лекций, ознакомьтесь со структурой и принципами функционирования микроконтроллера AVR ATMEGA128.

Задание 1. Ответьте на контрольные вопросы с №1 - №36 (согласно заданному варианту).

Вариант выбирается по последней цифре номера строки в журнале группы.

Набор контрольных вопросов для отчета выбирается согласно варианту из таблицы.

№ Вар.	0	1	2	3	4	5	6	7	8	9
Вопрос 1	1	2	3	4	5	6	7	8	9	10
Вопрос 2	10	9	8	7	6	4	5	2	3	1
Вопрос 4	11	12	13	14	15	16	17	18	19	20
Вопрос 5	19	20	11	12	13	14	15	16	17	18
Вопрос 6	21	22	23	24	25	26	27	28	29	30
Вопрос 7	31	32	33	34	35	36	21	22	23	24
Вопрос 8	25	26	27	28	29	30	31	32	33	34

ПРАКТИЧЕСКАЯ РАБОТА №41

Тема: «Сумматоры»

Цель: изучить архитектуру и принципы построения микроконтроллера AVR ATMEGA128.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Порядок выполнения работы

Занятие 2.

Перед началом выполнения практической части лабораторной работы проводится экспресс–контроль знаний по принципам функционирования микроконтроллера AVR ATMEGA 128. При подготовке к занятию 2 необходимо составить предварительный вариант листинга программы, в соответствие с индивидуальным заданием (см. таблицу).

Задание 2. Разработать в среде программирования Code Vision AVR программу на языке ASSEMBLER для микроконтроллера AVR ATMEGA 128, выполняющую сложение двух однобайтных чисел.

Порядок выполнения задания:

1. Включить лабораторный макет (установить выключатель электропитания в положение I, и убедиться в свечении индикатора электропитания красным цветом).
2. Запустить компилятор Code Vision AVR.
3. Создать пустой проект.
4. Создать файл ресурса для кода программы и подключить его к проекту.
5. Ввести код исходного модуля программы управления светодиодами в соответствие с вариантом задания, указанным в таблице.
6. Выполнить компиляцию (нажав клавишу **F9**) исходного модуля программы и устранить ошибки, полученные на данном этапе.
7. Настроить параметры программатора.
8. Создать загрузочный модуль программы (нажав комбинацию клавиш **Shift+F9**) и выполнить программирование микроконтроллера.
9. Проверить работоспособность загруженной в микроконтроллер программы и показать результаты работы преподавателю.
10. В случае некорректной работы разработанной программы, выполнить аппаратный сброс микроконтроллера, провести отладку исходного модуля программы и заново проверить функционирование программы, повторив выполнение пункта 9.

Пример выполнения задания

Разработать программу, выполняющую сложение двух однобайтных чисел. Эта программа будет реализовать функцию:

$$f(a,b) = a + b$$

Допустим, что первое слагаемое будет находиться в памяти с адресом указанным в регистре **X**, а второе слагаемое будет располагаться в регистре **R16**. Результат вычислений должен находиться в памяти с адресом указанным в регистре **Y**. Для правильного выполнения программы необходимо выполнить начальную инициализацию слагаемых.

Так как, первое слагаемое находится в памяти с адресом указанным в регистре **X**, выполним инициализацию его. Допустим значение первого слагаемого будет число 125 (0x7Dh) Для этого в программе необходимо внести такой код:

ldi R20, 125; промежуточное внесение числа в регистр R20.

st X, R20; перенесение значения из регистра R20 в область памяти, где располагается первое слагаемое (адрес храниться в двухбайтовом регистре **X**).

Для инициализации второго слагаемого, которое будет находиться в регистре **R16**, используем число 60 (0x3C)

ldi R16, 60

Теперь оба слагаемые имеют значения (125 и 60). Для выполнения сложения двух чисел использует команду **add**. Специфика этой команды в том, что команда может оперировать только с регистрами общего назначения.

Для выполнения сложения необходимо перенести первое слагаемое из памяти, например в регистр **R17**. Операция, а также операция сложения показана в следующем листинге.

ld R17, X

add R16, R17

Результат выполнения команды сложения **add** находится в регистре **R16**. Переноса результата в память, согласно заданию, реализуется такой командой:

st Y, R16

Полный листинг программы сложения двух чисел, показан ниже:

ldi R20, 125.

st X, R20

ldi R16, 60

ld R17, X

add R16, R17

st Y, R16

Задание 3. Разработать в среде программирования Code Vision AVR программу на языке ASSEMBLER для микроконтроллера AVR ATMEGA 128, выполняющую сложение двух двухбайтных чисел. Варианты выполнения задания находятся в таблице.

Таблица – Таблица значений слагаемых

	0	1	2	3	4	5	6	7	8	9
Первое слагаемое	0CFE	565A	AD32	FE63	3CB6	3271	5CA6	78DF	FC17	1978

Второе слагаемое	2A12	AACD	77FF	5657	5287	9FE3	36DE	2317	6E8C	0912
---------------------	------	------	------	------	------	------	------	------	------	------

В таблице значения представлены в шестнадцатеричной системе. Номер варианта выбирается, как предпоследняя цифра номера зачетной книжки.

Таблица – Таблица выбора размещения операндов

	0	1	2	3	4	5	6	7	8	9
Первое слагаемое	1	1	1	2	2	2	2	1	1	2
Второе слагаемое	1	2	2	2	2	1	2	2	2	1
Результат	1	1	2	2	1	1	1	1	2	1



Рисунок – Блок-схема алгоритм сложения двух двухбайтных чисел

В таблице значения соответствуют расположению операндов:

1 – регистры;

2 – память.

Номер варианта выбирается, как последняя цифра номера зачетной книжки.

3. Содержание отчета

Отчет должен содержать:

- Титульный лист;
- Название;
- Цель работы;
- Ответы на контрольные вопросы (согласно заданному варианту).
- Алгоритм работы программы согласно индивидуальному заданию
- Листинг программы согласно индивидуальному заданию

Контрольные вопросы

1. Какая архитектура лежит в основе данных микропроцессоров фирмы AVR?
2. Какая тактовая частота может устанавливаться для микропроцессоров?
3. Сколько каналов АЦП насчитывает микропроцессор?
4. Сколько разрядов в одном канале АЦП?
5. Количество регистров общего назначения?
6. Что такое АЛУ?
7. Объем статической памяти?
8. Объем Flash – памяти?
9. Объем электрически стираемой памяти?
10. Количество источников внутреннего прерывания?
11. Количество источников внешнего прерывания?
12. Какие интерфейсы программирования существуют у данного типа микропроцессоров?
13. Какие периферийные устройства входят в состав микроконтроллера?
14. В чем заключается гарвардская структура микроконтроллера?
15. Что такое программный счетчик?
16. Какие режимы уменьшения мощности существуют для микропроцессора?
17. Какие возможности предоставляет режим холостого хода (Idle)?
18. Какие возможности предоставляет режим выключения (Powerdown)?
19. Какие возможности предоставляет экономичный режим (Power-save)?
20. Какие возможности предоставляет дежурный режим (Standby)?
21. Какие возможности предоставляет режим снижения шумов АЦП (ADC Noise Reduction)?
22. Какие возможности предоставляет расширенный дежурный режим (Extended Standby)?
23. Какой интерфейс позволяет программировать встроенную Flash-память?

24. Назначение регистра SREG
25. Назначение флага C регистра SREG
26. Назначение флага Z регистра SREG
27. Назначение флага N регистра SREG
28. Назначение флага V регистра SREG
29. Назначение флага S регистра SREG
30. Назначение флага H регистра SREG
31. Назначение флага T регистра SREG
32. Назначение флага I регистра SREG
33. Что такое порт ввода вывода?
34. Назначение регистров PIN*.
35. Назначение регистров DDR*
36. Назначение регистров PORT*
37. Что такое система команд?
38. Какие виды команд существуют?
39. Какие команды относятся к командам пересылки данных и их назначение?
40. Какие команды относятся к арифметическим командам и их назначение?
41. Какие команды относятся к логическим командам пересылки данных и их назначение?
42. Какие команды относятся к инструкциям для работы с битами и их назначение?
43. Какие команды относятся к командам управления ходом исполнения программы и их назначение?

ПРАКТИЧЕСКАЯ РАБОТА № 42

Тема: «Преобразование и передача данных»

Цель: познакомиться с основными понятиями языка Турбо Паскаль, правилами записи арифметических выражений. формировать навыки представления арифметических выражений на Паскале.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Справочный материал

Ознакомиться с теоретическим материалом «Основные сведения о языке Турбо Паскаль»:

Для того чтобы правильно записывать арифметические выражения, нужно соблюдать следующие правила:

1. Все символы пишутся в строчку на одном уровне. Проставляются все знаки операций (нельзя пропускать знак умножения).
2. Не допускаются два следующих подряд знака операций (нельзя $A+-B$; можно $A+(-B)$).

3. Операции с более высоким приоритетом выполняются раньше операций с меньшим приоритетом. Порядок убывания приоритетов:

- вычисление функции;
- унарная операция смены знака (-);
- *, /, div, mod;
- +, - .

4. Несколько записанных подряд операций одинакового приоритета выполняются последовательно слева направо.

5. Часть выражения, заключенная в скобки, вычисляется в первую очередь.

(Например, $(A+B) * (C-D)$ — умножение производится после сложения и вычитания.)

Не следует записывать выражений, не имеющих математического смысла. Например, деление на нуль, логарифм отрицательного числа и т.п.

В Паскале нет операции или стандартной функции возведения числа в произвольную степень. Для вычисления x^y рекомендуется поступать следующим образом:

- если y — целое значение, то степень вычисляется через умножение; например, x^3 — $x \times x \times x$.
- если y — вещественное значение, то используется следующая математическая формула:
$$x^y = e^{y \ln(x)}.$$

На Паскале это будет выглядеть так: $\text{Exp}(y * \text{Ln}(x))$.

Содержание работы

Используя правила записи арифметических выражений на Паскале, выполнить следующие задания:

1) Для следующих формул записать соответствующие арифметические выражения на Паскале:

в) $\frac{a+b}{c} + \frac{c}{ab}$;

г) $\frac{x+y}{a_1} \cdot \frac{a_2}{x-y}$;

д) $10^4 \alpha - 3 \frac{1}{5} \beta$;

е) $\left(1 + \frac{x}{2!} + \frac{y}{3!}\right) / \left(1 + \frac{2}{3+xy}\right)$.

2) Записать математические формулы, соответствующие следующим выражениям на Паскале:

а) $(p+q)/(r+s) - p * q / (r * s)$;

б) $1E3 + beta / (x - gamma * delta)$;

в) $a/b * (c+d) - (a-b)/b/c + 1E-8$.

3) Для следующих формул записать соответствующие арифметические выражения на Паскале:

$$\begin{array}{llll} \text{a)} (1+x)^2; & \text{б)} \sqrt{1+x^2}; & \text{в)} \cos^2 x^2; & \text{г)} \log_2 \frac{x}{5}; \\ \text{д)} \arcsin x; & \text{е)} \frac{e^x + e^{-x}}{2}; & \text{ж)} x^{\sqrt{2}}; & \text{з)} \sqrt[3]{1+x}; \\ \text{и)} \sqrt{x^2+8^2}; & \text{к)} \frac{xyz - 3,3|x + \sqrt[4]{y}|}{10^7 + \ln 4!}; & \text{л)} \frac{\beta + \sin^2 \pi^4}{\cos 2 + |\operatorname{ctg} \gamma|} \end{array}$$

4) Вычислить значения выражений:

- а) $\text{round}(6.9)$; в) $20 \operatorname{div} 6$; д) $20 \bmod 6$;
 б) $\text{round}(6.)$; г) $2 \operatorname{div} 5$; ж) $2 \bmod 5$;
 и) $3*7 \operatorname{div} 2 \bmod 7/3 - \text{trunc}(\sin(1))$.

ПРАКТИЧЕСКАЯ РАБОТА № 43

Тема: «Преобразование и передача данных»

Цель: разобрать структуру программ в общем виде; сформировать навыки объявления переменных, использование инструкции присваивания, ввода, вывода.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Содержание работы

Задание. Инструкция присваивания

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкция присваивания используется для изменения значений переменных, в том числе и для вычислений по формулам;
- тип выражения, находящегося в правой части инструкции присваивания, должен соответствовать типу переменной, имя которой стоит слева от символа инструкции присваивания (при нарушении соответствия типа переменной и выражения компилятор выводит сообщение об ошибке **Type miss match** — несоответствие типов).

Пример. Запишите инструкцию, которая увеличивает на единицу значение переменной n .

Решение: $n := n + 1$;

2.1. Запишите инструкцию, которая присваивает переменной summa нулевое значение.

2.2. Запишите инструкцию вычисления среднего арифметического переменных x_1 и x_2 .

2.3. Запишите в виде инструкции присваивания формулу вычисления значения функции

$$y = -2,7x^3 + 0,23x^2 - 1,4.$$

2.4. Запишите в виде инструкции присваивания формулу пересчета веса из фунтов в килограммы (один фунт равен 409,5 г).

2.5. Запишите в виде инструкции присваивания формулу пересчета

расстояния из километров в версты (одна верста равна 1066,8 м).

2.6. Запишите в виде инструкции присваивания формулу вычисления площади треугольника:

$s = 1/2 \cdot ah$, где a — длина основания треугольника, h — его высота.

ПРАКТИЧЕСКАЯ РАБОТА № 44

Тема: «Преобразование и передача данных»

Цель: учиться составлять простейшие программы.

Оборудование: справочный материал, персональный компьютер с выходом в Интернет.

Содержание работы

Задание. Вывод

Приступая к решению задач этого раздела, следует вспомнить, что:

- инструкции `write` и `writeln` предназначены для вывода на экран монитора сообщений и значений переменных;
- одна инструкция `write` (`writeln`) может вывести на экран значения нескольких переменных и (или) несколько сообщений;
- инструкция `writeln` без параметров переводит курсор в начало следующей строки экрана.

Пример. Написать инструкции вывода значений переменных a , b и c (тип вещественный, 2 знака после запятой). Значение каждой переменной должно быть выведено на отдельной строке.

Решение:

```
writeln (a:6:2);  
writeln (b:6:2);  
writeln (c:6:2);
```

Написать программу, которая выводит на экран имя и фамилию.

Написать программу, которая выводит на экран четверостишие:

Унылая пора! Очей очарованье!
Приятна мне твоя прощальная краса —
Люблю я пышное природы увяданье,
В багрец и золото одетые леса.

А. С. Пушкин

3.3. Написать инструкцию вывода значения переменной a (тип *real*) с тремя цифрами в дробной части.

Задание. Ввод

Приступая к решению задач этого раздела, следует вспомнить, что:

- для ввода с клавиатуры во время работы программы исходных данных (значений переменных) предназначена инструкция `readln`;

- используя одну инструкцию `readln`, можно ввести значения нескольких переменных;
- тип данных, вводимых во время работы программы, должен соответствовать типу переменной, указанной в инструкции `readln`;
- в случае несоответствия типа введенных данных типу переменной, значение которой вводится с клавиатуры, программа завершает работу и на экран выводится сообщение `Error 106: Invalid numeric format` (если программа запущена из среды разработки, т. е. из Turbo Pascal) или `Run time error 106` (если программа запущена из операционной системы).

Пример. Написать инструкцию, которая обеспечивает ввод значений переменных `u` и `r`. Предполагается, что во время работы программы пользователь будет набирать числа в одной строке.

Решение: `readln (u,r);`

- 4.1. Написать инструкции, которые обеспечивают ввод значений переменных `u` и `r`. Предполагается, что во время работы программы пользователь будет после набора каждого числа нажимать клавишу `<Enter>`.
- 4.2. Объявите необходимые переменные и напишите фрагмент программы вычисления объема цилиндра, обеспечивающий ввод исходных данных.
- 4.3. Объявите необходимые переменные и напишите инструкции ввода исходных данных для программы вычисления стоимости покупки нескольких тетрадей и карандашей. Предполагается, что во время работы программы пользователь будет вводить данные о каждой составляющей покупки в отдельной строке: сначала цену, затем количество.

Информационное обеспечение обучения

Печатные и электронные издания

Основные учебные издания:

1. Дерягин А.В. Основы автоматики и вычислительной техники: учебное пособие для СПО / А.В. Дерягин, Ф.М. Сабирова. – Санкт-Петербург: Лань, 2024. – 108 с.: ил. – Текст: непосредственный.
2. Синаторов С.В. Пакеты прикладных программ: учебное пособие / Синаторов С.В. — Москва: КноРус, 2021. — 195 с. — ISBN 978-5-406-08111-2. — URL: <https://book.ru/book/939069>

Электронно-библиотечная система:

3. ЭБС «Znanium»
4. ЭБС «PROFобразование»
5. ЭБС «Book.ru»