Муниципальное бюджетное общеобразовательное учреждение «Плодовая среднеобразовательная школа» г.Ульяновска

Индивидуальный итоговый проект

**Тема: ”Создание собственной архитектуры ЭВМ”**

Предмет: Информатика

Выполнил Фазылзянов Руслан ученик 11 класса.

Руководитель: Павлушина Мария Александровна.

Ульяновск, Сентябрь 2020г.

**Оглавление**

1. Свойства проекта
2. Введение
3. Теоретическая часть
   1. Логические вентили
   2. Архитектуры
   3. Структура ЭВМ
   4. Структура Процессора
      1. АЛУ (Арифметико-логическое устройство)
      2. УУ (Устройство управления)
      3. Регистры
      4. Дополнительные устройства
      5. Работа с командами
4. Практическая часть
   1. Базовые знания по построению логических схем.
   2. Триггеры, Регистры и прочие функциональные элементы
   3. Сумматоры
   4. БПЛО (Блок побитовых логических операций)
   5. БАО (Блок арифметических операций)
   6. АЛУ (Арифметико-логическое устройство)
   7. УУ (Устройство управления)
      1. Составление списка команд
      2. УДК (Устройство декодирования команд)
      3. УАК (Устройство адресации команд)
      4. УТ (Устройство тактирования)
   8. Готовая схема и проверка работоспособности
   9. Исправление ошибок
5. Итог

**Свойства проекта**

**Актуальность проекта:**

* Малое количество простых индивидуальных архитектур, как процессоров, так и ЭВМ.
* Малое количество свободной для понимания информации в Интернете.

**Цель проекта:**

* Собрать собственную архитектуру процессора.
* Рассказать о логических схемах в формате книги.

**Объект исследования:**

* Построение логических схем (как наука)
* Архитектура ЭВМ

**Предмет исследования:**

* Архитектура и устройство простейших ЭВМ и их компонентов.

**Теоретическая значимость:**

* Люди смогут узнать и разобраться в устройстве многих компонентов ЭВМ, и далее применить знания на практике (при соответствующем желании).

**Практическая значимость:**

* Практическая часть даст возможность досконального изучения и выявления некоторых нюансов при создании ЭВМ.

**Задачи:**

* Найти информацию об архитектурах.
* Найти похожие проекты и рассказать о них.
* Рассказать факты, касающиеся данной темы, для большего понимания.
* Собрать информацию о структуре компонентов и их разновидностях в ЭВМ.
* Предложить собственные идеи о структуре компонентов и их разновидностях в ЭВМ.
* Составить набор выполняемых команд собственной архитектуры.
* Составить логическую схему собственной архитектуры. (продукт проекта 1)
* Составить программу на собственной архитектуре «Калькулятор» (продукт проекта 2)
* Создать книгу по данному проекту (продукт проекта 3)

**Введение**

Здравствуйте, это мой индивидуальный проект учащегося 10 класса. Тема, которую я затрагиваю очень редка, и мало кому интересна. Наука изучающая Логические схемы забывается среди населения из-за перехода в промышленность. Крупные компании монополизируют рынок микросхем, а малые производства, не имея многих знаний в данной сфере, практически не способны развиваться. Самые основные устройства и их схемы в интернете, конечно, доступны, но разнообразия почти что нет. Это я имел в виду лишь современную информацию, а литература советского времени довольно богата знаниями. Что говорит об остановке развития науки среди доступного населению пространства. Основные исследования проводят крупные компании, они патентуют почти всё, ведь в ином случае произойдёт потеря прибыли. На момент 2020г. компании ведут яростную борьбу за потребителя, так будет всегда, поэтому в доходах они особо заинтересованы. По данным причинам энтузиасты, желающие сделать что-то ~~и зачем-то~~, не могут проникнуть в данную сферу. А ведь таких людей не мало, и я сам знаю 6 человек (правда они меня не знают). Так что я вовлечённый в данную научную сферу решил написать данный проект.

Проект рассказывает о создании собственной простейшей архитектуры ЭВМ. Современные ЭВМ как правило имеют процессор, то есть я не буду рассказывать о создании просто процессора, я решил замахнуться и спроектировать ЭВМ целиком. Как и вам полезно, можете прочитать об интересующем вас элементе, так и мне есть чем заняться.

**Теоретическая часть**

Перед началом собственно проектирования нужно понимать, что мы собрались проектировать.

ЭВМ – Электронно-Вычислительная Машина, выполняет вычисления используя электроэнергию. Вы ведь помните, что такое ламповый компьютер, первейший класс ЭВМ? Тогда и названия “компьютер” было распространенно только среди учёных. Они выполняли различные логические и математические задачи. Потребляли много энергии и были размером в 1-4 комнат. А потом с изобретения Транзистора началась новая эра.

Транзистор был отличной заменой Вакуумным электронным лампам. Потребления энергии меньше, размеры намного меньше, способ эксплуатации легче и надёжность выше. Потом люди додумались делать несколько транзисторов в одном корпусе, придумали литографию и начали выпускать первые микросхемы, а там и до процессоров не далеко было.

И да! Процессор как термин появился позже ЭВМ. Но у многих ЭВМ структура позволяла наличие Процессора, как главного вычислительного блока.

Что нам надо знать для проектирования собственной ЭВМ? На самом деле, очень много. Начиная с логических вентилей, заканчивая устройством адресации доп. устройств.

Приступим…

**Логические вентили**

Логические вентили (Логические элементы ЛЭ) - базовый элемент цифровой схемы, выполняющий элементарную логическую операцию, преобразуя таким образом множество входных логических сигналов в выходной логический сигнал.

А проще говоря это, элемент выполняющий какую-то простейшую логику основываясь на входящих сигналах. Они могут иметь как 1 вход, так и 8, или более. И могут состоять из Электронных ламп, транзисторов или чего-нибудь ещё что будет сказываться на их характеристиках. Но всегда имеют 1 выход. Для двоичной логики, Логических вентилей существует ровно 7. *(таб. 1)* Если количество выходных сигналов больше одного, то Логическим элементом это назвать нельзя. Такой элемент уже будет являться Логическим модулем, состоящим из Логических элементов. *(рис. 1, 2)*

Логику работы этих Логических элементов можно записать в таблицу, которая называется Таблица истинности. *(таб. 3)* Такие таблицы могут быть бесконечны! Смысл в том, что они дают понять логику работы и возможность взглянуть на схему, с другой стороны.

Также стоит запомнить и научиться составлять Логические формулы. *(таб. 2)* Чем-то они похожи на физические или математические формулы, но здесь используются определённые символы (которые вам тоже стоит понимать). С помощью этих формул можно посмотреть на логику работы с математической стороны, что может помочь с оптимизацией схемы.

Если вы уже создали свою Логическую схему и хотите её реализовать в жизнь, начните с оптимизации Логической схемы. При оптимизации необходимо учитывать радиоэлектронную схему Логических вентилей, их структуру. Например, Элемент И сложнее чем элемент И-НЕ, разница всего на 1-3 Транзистора вам принесёт много проблем с реализацией вашей схемы. Конечно, это зависит от технологии Логического вентиля (ТТЛ, МОП, КМОП, и т.д.). Но оптимизация никому пока не мешала, а в индустрии микроэлектроники это, очень важная задача от которой, можно сказать, зависит всё.

*Таблица 1* **Логические элементы, их функции и обозначения**

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Обозначение** | | **Описание** |
| **Международное** | **IEC 60617-12 : 1997** |
| NOT  (НЕ) |  |  | Инвертирует входящий сигнал. |
| AND  (И) |  |  | Выдаёт сигнал только при наличии сигнала на всех своих входах. |
| OR  (ИЛИ) |  |  | Выдаёт сигнал только при наличии сигнала на любом из своих входов. |
| XOR  (Исключающее ИЛИ) |  |  | Выдаёт сигнал только при наличии сигнала на одном любом из своих входов. |
| NAND  (И-НЕ) |  |  | Выдаёт сигнал только при отсутствии сигнала на хотя бы одном из всех своих входов. |
| NOR  (ИЛИ-НЕ) |  |  | Выдаёт сигнал только при отсутствии сигнала на всех своих входах. |
| XNOR  (Исключающее ИЛИ-НЕ) |  |  | Не выдаёт сигнал только при наличии сигнала на одном любом из своих входов. |

|  |  |
| --- | --- |
| *Таблица 3* Таблица истинности | |
| Название | Таблица истинности |
| NOT  (НЕ) | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | | A |  | **B** | **Out** | | 0 |  | - | 1 | | 1 |  | - | 0 | | - |  | - | - | | - |  | - | - | |  |  |  |  | |
| AND  (И) | |  |  |  | | --- | --- | --- | |  |  |  | | A | **B** | **Out** | | 0 | 0 | 0 | | 1 | 0 | 0 | | 0 | 1 | 0 | | 1 | 1 | 1 | |  |  |  | |
| OR  (ИЛИ) | |  |  |  | | --- | --- | --- | |  |  |  | | A | **B** | **Out** | | 0 | 0 | 0 | | 1 | 0 | 1 | | 0 | 1 | 1 | | 1 | 1 | 1 | |  |  |  | |
| XOR  (Исключающее ИЛИ) | |  |  |  | | --- | --- | --- | |  |  |  | | A | **B** | **Out** | | 0 | 0 | 0 | | 1 | 0 | 1 | | 0 | 1 | 1 | | 1 | 1 | 0 | |  |  |  | |
| NAND  (И-НЕ) | |  |  |  | | --- | --- | --- | |  |  |  | | A | **B** | **Out** | | 0 | 0 | 1 | | 1 | 0 | 1 | | 0 | 1 | 1 | | 1 | 1 | 0 | |  |  |  | |
| NOR  (ИЛИ-НЕ) | |  |  |  | | --- | --- | --- | |  |  |  | | A | **B** | **Out** | | 0 | 0 | 1 | | 1 | 0 | 0 | | 0 | 1 | 0 | | 1 | 1 | 0 | |  |  |  | |
| XNOR  (Исключающее ИЛИ-НЕ) | |  |  |  | | --- | --- | --- | |  |  |  | | A | **B** | **Out** | | 0 | 0 | 1 | | 1 | 0 | 0 | | 0 | 1 | 0 | | 1 | 1 | 1 | |  |  |  | |

|  |  |  |
| --- | --- | --- |
| *Таблица 2* **Логические операций** | | |
| Название | Обозначение | Описание |
| **Инверсия** |  | Инвертирует значение.  Соответствует вентилю НЕ. |
| **Конъюнкция** | , | Умножает значения.  Соответствует вентилю И. |
| **Дизъюнкция** | , | Складывает значения.  Соответствует вентилю ИЛИ. |
| **Эквиваленция** | , | Сравнивает значения.  Соответствует вентилю Исключающее ИЛИ-НЕ. |
| **Импликация** |  | Если A то возвращается B, иначе 1.  Соответствует вентилю И. |

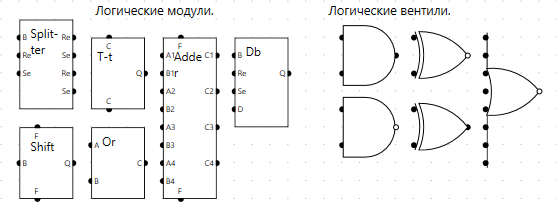


Рисунок 1 **Пример Логических модулей и Логических вентилей**

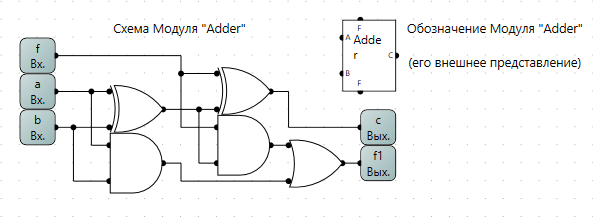


Рисунок 2 **Логический модуль и его логическая схема**

**Архитектуры**

Архитектура, это довольно общее понятие, собирающее в себе много определённых терминов. Она описывает как структуру с функциями, так и характеристики. Легче будет усвоить информацию разбирая отдельные классы Архитектур, которые соответствуют нашей теме.

Начнём с Архитектуры ЭВМ. Коих имеется много, но основными являются всего два. Это Архитектура Фон-Неймана и Архитектура Гарварда. *(таб. 4)*

Обе этих архитектур совершенно различны и имеют разные задачи. Например, Архитектура Фон-Неймана позволяет написать и запустить программу на уровне машинного кода (Ассемблер), что мы и встречаем в современных компьютерах, а Архитектура Гарварда используется в устройствах где всегда будет выполнятся одна и та же программа, как в микроконтроллерах. Так же стоит отметить, что Архитектура Фон-Неймана уступает по производительности Архитектуре Гарварда, из-за ограничений на запись и чтение данных. Прирост составляет 20% при использовании одноканальной памяти.

*Таблица 4* **Схема Архитектур Фон-Неймана и Гарварда**

|  |  |
| --- | --- |
| Фон Нейман | Гарвард |
| |  | | --- | | Общая  память | |  | | Процессор | |  | | Прочие  устройства | | |  |  |  | | --- | --- | --- | | Память  данных |  | Память  команд | |  |  |  | | Процессор | | | |  | | | | Прочие  устройства | | | |

Далее идёт Архитектура Процессора. Это совокупность главных принципов его конструирования, общая схема расположения деталей и схема взаимодействия программного обеспечения с чипом. И данных Архитектур огромное множество, что были созданы различные классификации. Такие как: RISC, TTA, OISC, CISC, итд. А уже, например, Zen3, M68k, x86 и IceLake это непосредственно Архитектуры, которые мы детально рассмотреть увы не сможем.

Основываясь на таблице Классов Архитектур *(таб. 5)*, классифицируются они по командам выполняемым процессором, или же по их Ассемблеру. Но различные модули процессора и их строение, которое нам сильно понадобятся, относятся непосредственно к Архитектурам Процессора, и классификации не принадлежат (из-за разных реализаций разными производителями).

(Например, никто не знает логическую схему модуля предсказания ветвлений, который выполняет заранее более ожидаемые, после предыдущей, команды)

Конечно в Интернете есть схемы мелких модулей. И находятся они в оцифрованных патентах СССР. Как например, устройство двоичного деления, в схеме которого я не разобрался. И спроектировал своё устройство деления. Но об этом поговорим позже.

|  |  |  |
| --- | --- | --- |
| Класс | Описание | Архитектуры |
| RISC | **Reduced Instruction Set Computer**  С Сокращенным Набором Команд Компьютера. Особенностью данной группы является увеличение производительности за счёт сокращения набора команд, редко используемых разработчиками ПО. Является золотой серединой. | ARM, MIPS, OpenRISC, RISC-V и др. |
| TTA | **Transport Triggered Architecture**  Транспортно-вызываемая архитектура. Архитектура на основе всего одной инструкции перемещения из одного адреса памяти в другую. Очень производительная архитектура, которая расплачивается сложностью программирования. | MOVE Project и др. |
| OISC | **One Instruction Set Computer**  Архитектура с единственной инструкцией. Такие архитектуры часто имеют вид: Сделать действие и в зависимости от результата сделать прыжок или продолжить исполнение. Зачастую ее реализация достаточно простая, производительность низкая. | BitBitJump, ByteByteJump, SUBLEQ и др. |
| CISC | **Complex Instruction Set Computer**  Компьютер Со Сложным Набором Команд. Ее особенность в увеличенном количестве действий за инструкцию. Таким образом можно было теоретически увеличить производительность программ за счет увеличения сложности компилятора. Огромное множество архитектур этой группы почти не имеют схожести между собой. | x86, x86-64, M68K и др. |

*Таблица 5*. **Классы Архитектур процессоров**

**Структура ЭВМ**

Как и всё в этом мире, ЭВМ тоже состоит из чего-то. И это компоненты в виде различных модулей или устройств.

Для выполнения списка команд ЭВМ имеет Память, как скоростную, так и постоянную. Отличие в том, что энергонезависимую память сделать быстрой очень проблематично. Современные SSD накопители так же не удовлетворяют нашим скоростным потребностям. И тут без скоростной энергозависимой Оперативной памяти не обойтись. Она состоит из тех же компонентов что и Процессор и способна работать чуть медленнее самого Процессора. Поэтому, в любом универсальном вычислительном устройстве есть как Оперативная память, так и Постоянная. ОЗУ и ПЗУ соответственно.

Далее любому устройству, что рассчитан на возможность использования Интерфейсов, требуются Устройства Ввода и/или Устройства Вывода информации. Допустим, калькулятор не поймёт, что считать если вы не передали ему что считать. А вы в свою очередь не поймёте, что получилось в итоге если калькулятор толком не может никак вывести информацию. Даже 1 кнопка или просто антенна из оголённого провода являются Устройствами Ввода, как и обычный электромотор, нить накала или обычный светодиод являются Устройствами Вывода. Но в нормальных условиях Клавиатура – Устройства Ввода, а дисплей – Устройство Вывода.

Наверное, самый главный элемент любого ЭВМ, это Процессор, но он не обязателен. Например, в ЭВМ 1960 года не было Процессора. Как отдельное устройство он отсутствовал, и зачастую выглядел скорее, как множество отдельных блоков. Элементы процессора, которые, казалось бы, есть только в нём же, вплетены в саму Архитектуру ЭВМ. Следует помнить, что у ЭВМ без процессора, есть все основные компоненты самих процессоров. Такие как: АЛУ, УУ и регистры.

Так же в ЭВМ присутствуют: устройство тактирования, Устройство декодирования команд, Блок питания и т.д..

**Структура Процессора**

Процессор основное вычислительное устройство всех современных ЭВМ. От вашего холодильника до Квантовых компьютеров. Данное устройство имеет блочную структуру своих Функциональных блоков. Некоторые из них являются необязательными, но наличие каждого из блоков сильно отражается на Архитектуре Процессора.

Самыми основными и базовыми блоками процессора являются: Управляющее Устройство, Арифметико-Логическое Устройство и Регистры.

Второстепенными блоками являются: Ядро Процессора, Стек, Устройство Математических Операций, Устройство Блокировки Доступа или Устройство Уровней Доступа, Устройства Предсказания Ветвлений, Кеш, Устройство Шифрования, Устройство Аппаратного Декодирования, Графический Модуль и т.д..

Различных компонентов Процессора очень много. И вы, наверное, обратили своё внимание на Ядро. Вроде в каждом процессоре есть хотя бы одно ядро, но вспомните первые Процессоры, где Ядер как отдельного блока не было. Ядра лишь включили в себя некоторые блоки процессора, так чтобы они работали независимо от таких же блоков. Позже, Ядра стали достаточно самостоятельными элементами и теперь уже напоминают первые процессоры.

Стоит сказать и о способах объединения Функциональных блоков между собой. Проводники передают электрический потенциал от одних блоков процессора к другим. Так же существует разновидность проводников, Шина. Не является устройством и состоит только из группы проводников, передающих одинаковую информацию всем подключенным к ней модулям.

Вроде всё просто, структура процессора, казалось бы, легка. Так оно и есть. В процессорах главной проблемой являются не сама структура, а именно её Функциональные блоки. Причина этому, постоянные доработки этих блоков для достижения максимальной эффективности. Далее мы будем рассматривать основные функциональные блоки процессора

**АЛУ (Арифметико-логическое устройство)**

Это устройство является одним из основных устройств процессора. Функциональность данного блока довольно обширна, и над его созданием и его модернизацией люди до сих пор ломают головы.

Данное устройство выполняет арифметические и логические операции над двоичными числами. Такие как: сложение, вычитание, умножение, деление, изменение знака, сравнение, булевы операции, и т.д..

АЛУ бывает двух видов: цельный и блочный. Цельный АЛУ является единой схемой, что является очень экономично. К сожалению, такие АЛУ очень сложны в разработке. Блочные АЛУ разбиты на блоки (Устройство сравнения, Устройство Арифметики, Устройство логических операций), где каждый блок выполняют свою функцию. Блочные АЛУ чрезмерно легки в разработке и могут похвастаться возможностью выполнения сразу нескольких операций последовательно.

Здесь пока. Ещё до создания первых ЭВМ люди открыли пользу Дополнительного кода (инвертированного числа). С помощью него можно сложить два числа так, чтобы получилась разница этих чисел, то есть для вычитания. Работает это следующим образом: вычитаемое инвертируется и складывается с уменьшаемым, после чего ответ обратно инвертируется. Как побочный эффект мы получаем активный флаг переноса, от которого можно избавится опять же инверсией.

--Физика  
--Общество  
--Английский  
--Алгебра  
->Литература

|  |  |  |
| --- | --- | --- |
| Записать в регистр число | 00`00`00`$$ число | 00`00`00`00 00000101  Записать в регистр А число 5 |
|  |  | 00`00`00`01 00000110  Записать в регистр Б число 6 |
|  |  | 00`00`00`10 00000111  Записать в регистр В число 7 |
|  |  | 00`00`00`11 00001000  Записать в регистр Г число 8 |
| Записать в регистр число из ячейки памяти данных по регистру адреса | 00`01`$$`$$ | 00`01`00`00  Записать в регистр А число из ячейки данных по адресу из регистра А |
|  |  | 00`01`00`01  Записать в регистр Б число из ячейки данных по адресу из регистра А |
|  |  | 00`01`00`10  Записать в регистр В число из ячейки данных по адресу из регистра А |
|  |  | 00`01`00`11  Записать в регистр Г число из ячейки данных по адресу из регистра А |
|  |  | 00`01`01`00  Записать в регистр А число из ячейки данных по адресу из регистра Б |
|  |  | 00`01`01`01  Записать в регистр Б число из ячейки данных по адресу из регистра Б |
|  |  | 00`01`01`10  Записать в регистр В число из ячейки данных по адресу из регистра Б |
|  |  | 00`01`01`11  Записать в регистр Г число из ячейки данных по адресу из регистра Б |
|  |  | 00`01`10`00  Записать в регистр А число из ячейки данных по адресу из регистра В |
|  |  | 00`01`10`01  Записать в регистр Б число из ячейки данных по адресу из регистра В |
|  |  | 00`01`10`10  Записать в регистр В число из ячейки данных по адресу из регистра В |
|  |  | 00`01`10`11  Записать в регистр Г число из ячейки данных по адресу из регистра В |
|  |  | 00`01`11`00  Записать в регистр А число из ячейки данных по адресу из регистра Г |
|  |  | 00`01`11`01  Записать в регистр Б число из ячейки данных по адресу из регистра Г |
|  |  | 00`01`11`10  Записать в регистр В число из ячейки данных по адресу из регистра Г |
|  |  | 00`01`11`11  Записать в регистр Г число из ячейки данных по адресу из регистра Г |
| Записать в ячейку памяти данных число из регистра по адресу из регистра | 00`10`$$`$$ | 00`10`00`00  Записать в ячейки данных по адресу из регистра регистр А число регистра А |
|  |  | 00`10`00`01  Записать в ячейки данных по адресу из регистра регистр Б число регистра А |
|  |  | 00`10`00`10  Записать в ячейки данных по адресу из регистра регистр В число регистра А |
|  |  | 00`10`00`11  Записать в ячейки данных по адресу из регистра регистр Г число регистра А |
|  |  | 00`10`01`00  Записать в ячейки данных по адресу из регистра регистр А число регистра Б |
|  |  | 00`10`01`01  Записать в ячейки данных по адресу из регистра регистр Б число регистра Б |
|  |  | 00`10`01`10  Записать в ячейки данных по адресу из регистра регистр В число регистра Б |
|  |  | 00`10`01`11  Записать в ячейки данных по адресу из регистра регистр Г число регистра Б |
|  |  | 00`10`10`00  Записать в ячейки данных по адресу из регистра регистр А число регистра В |
|  |  | 00`10`10`01  Записать в ячейки данных по адресу из регистра регистр Б число регистра В |
|  |  | 00`10`10`10  Записать в ячейки данных по адресу из регистра регистр В число регистра В |
|  |  | 00`10`10`11  Записать в ячейки данных по адресу из регистра регистр Г число регистра В |
|  |  | 00`10`11`00  Записать в ячейки данных по адресу из регистра регистр А число регистра Г |
|  |  | 00`10`11`01  Записать в ячейки данных по адресу из регистра регистр Б число регистра Г |
|  |  | 00`10`11`10  Записать в ячейки данных по адресу из регистра регистр В число регистра Г |
|  |  | 00`10`11`11  Записать в ячейки данных по адресу из регистра регистр Г число регистра Г |
| Записать в регистр числа из регистра | 00`11`$$`$$ | 00`11`00`00  Записать в регистр А число из регистра А |
|  |  | 00`11`00`01  Записать в регистр Б число из регистра А |
|  |  | 00`11`00`10  Записать в регистр В число из регистра А |
|  |  | 00`11`00`11  Записать в регистр Г число из регистра А |
|  |  | 00`11`01`00  Записать в регистр А число из регистра Б |
|  |  | 00`11`01`01  Записать в регистр Б число из регистра Б |
|  |  | 00`11`01`10  Записать в регистр В число из регистра Б |
|  |  | 00`11`01`11  Записать в регистр Г число из регистра Б |
|  |  | 00`11`10`00  Записать в регистр А число из регистра В |
|  |  | 00`11`10`01  Записать в регистр Б число из регистра В |
|  |  | 00`11`10`10  Записать в регистр В число из регистра В |
|  |  | 00`11`10`11  Записать в регистр Г число из регистра В |
|  |  | 00`11`11`00  Записать в регистр А число из регистра Г |
|  |  | 00`11`11`01  Записать в регистр Б число из регистра Г |
|  |  | 00`11`11`10  Записать в регистр В число из регистра Г |
|  |  | 00`11`11`11  Записать в регистр Г число из регистра Г |
| Произвести запись в регистр Ак | 01`00`00`$$ | 01`00`00`00  Записать в Ак значение из регистра А |
|  |  | 01`00`00`01  Записать в Ак значение из регистра Б |
|  |  | 01`00`00`10  Записать в Ак значение из регистра В |
|  |  | 01`00`00`11  Записать в Ак значение из регистра Г |
| Произвести запись из регистра Ак | 01`01`00`$$ | 01`00`00`00  Записать в регистр А значение из Ак |
|  |  | 01`00`00`01  Записать в регистр Б значение из Ак |
|  |  | 01`00`00`10  Записать в регистр В значение из Ак |
|  |  | 01`00`00`11  Записать в регистр Г значение из Ак |
| Произвести математическую операцию между регистрами А и Ак | 01`10`00`00 | 01`10`00`00  Сложение А и Ак |
|  |  | 01`10`00`01  Вычитание из А числа Ак |
|  |  | 01`10`00`10  Вычитание из Ак числа А |
|  |  | 01`10`01`01  Инкремент Ак |
|  |  | 01`10`01`10  Декремент Ак |
| Произвести логическую операцию между регистрами А и Ак | 01`11`00`00 | 01`11`00`00  Инвертировать Ак |
|  |  | 01`11`00`01  Побитовое И между числами А и Ак |
|  |  | 01`11`00`10  Побитовое ИЛИ между числами А и Ак |
|  |  | 01`11`00`11  Побитовая Эквиваленция между числами А и Ак |
|  |  | 01`11`01`00  Алгебраический сдвиг Ак на 1 разряд вправо |
|  |  | 01`11`01`01  Алгебраический сдвиг Ак на 1 разряд влево |
|  |  | 01`11`01`10  Логический сдвиг Ак на 1 разряд вправо |
|  |  | 01`11`01`11  Логический сдвиг Ак на 1 разряд вправо |
| Условный оператор между регистрами (проверка на Больше) | 10`00`$$`$$ | 10`00`00`00  Если регистр А больше регистра А |
|  |  | 10`00`00`01  Если регистр А больше регистра Б |
|  |  | 10`00`00`10  Если регистр А больше регистра В |
|  |  | 10`00`01`11  Если регистр А больше регистра Г |
|  |  | 10`00`01`00  Если регистр Б больше регистра А |
|  |  | 10`00`01`01  Если регистр Б больше регистра Б |
|  |  | 10`00`01`10  Если регистр Б больше регистра В |
|  |  | 10`00`01`11  Если регистр Б больше регистра Г |
|  |  | 10`00`10`00  Если регистр В больше регистра А |
|  |  | 10`00`10`01  Если регистр В больше регистра Б |
|  |  | 10`00`10`10  Если регистр В больше регистра В |
|  |  | 10`00`10`11  Если регистр В больше регистра Г |
|  |  | 10`00`11`00  Если регистр Г больше регистра А |
|  |  | 10`00`11`01  Если регистр Г больше регистра Б |
|  |  | 10`00`11`10  Если регистр Г больше регистра В |
|  |  | 10`00`11`11  Если регистр Г больше регистра Г |
| Условный оператор между регистрами (проверка на Меньше) | 10`01`$$`$$ | 10`01`00`00  Если регистр А меньше регистра А |
|  |  | 10`01`00`01  Если регистр А меньше регистра Б |
|  |  | 10`01`00`10  Если регистр А меньше регистра В |
|  |  | 10`01`01`11  Если регистр А меньше регистра Г |
|  |  | 10`01`01`00  Если регистр Б меньше регистра А |
|  |  | 10`01`01`01  Если регистр Б меньше регистра Б |
|  |  | 10`01`01`10  Если регистр Б меньше регистра В |
|  |  | 10`01`01`11  Если регистр Б меньше регистра Г |
|  |  | 10`01`10`00  Если регистр В меньше регистра А |
|  |  | 10`01`10`01  Если регистр В меньше регистра Б |
|  |  | 10`01`10`10  Если регистр В меньше регистра В |
|  |  | 10`01`10`11  Если регистр В меньше регистра Г |
|  |  | 10`01`11`00  Если регистр Г меньше регистра А |
|  |  | 10`01`11`01  Если регистр Г меньше регистра Б |
|  |  | 10`01`11`10  Если регистр Г меньше регистра В |
|  |  | 10`01`11`11  Если регистр Г меньше регистра Г |
| Условный оператор между регистрами (проверка на Равность) | 10`01`$$`$$ | 10`10`00`00  Если регистр А равен регистру А |
|  |  | 10`10`00`01  Если регистр А равен регистру Б |
|  |  | 10`10`00`10  Если регистр А равен регистру В |
|  |  | 10`10`01`11  Если регистр А равен регистру Г |
|  |  | 10`10`01`00  Если регистр Б равен регистру А |
|  |  | 10`10`01`01  Если регистр Б равен регистру Б |
|  |  | 10`10`01`10  Если регистр Б равен регистру В |
|  |  | 10`10`01`11  Если регистр Б равен регистру Г |
|  |  | 10`10`10`00  Если регистр В равен регистру А |
|  |  | 10`10`10`01  Если регистр В равен регистру Б |
|  |  | 10`10`10`10  Если регистр В равен регистру В |
|  |  | 10`10`10`11  Если регистр В равен регистру Г |
|  |  | 10`10`11`00  Если регистр Г равен регистру А |
|  |  | 10`10`11`01  Если регистр Г равен регистру Б |
|  |  | 10`10`11`10  Если регистр Г равен регистру В |
|  |  | 10`10`11`11  Если регистр Г равен регистру Г |
| Установка каретки команд на значение из регистра | 11`00`00`$$ | 11`00`00`00  Установка каретки команд на число из регистра А |
|  |  | 11`00`00`01  Установка каретки команд на число из регистра Б |
|  |  | 11`00`00`10  Установка каретки команд на число из регистра В |
|  |  | 11`00`00`11  Установка каретки команд на число из регистра Г |
| Установка каретки команд на значение | 11`00`01`00 | 11`00`01`00 00011100  Установка каретки команд на ячейку 28 |
| Установка каретки команд на значение из регистра | 11`00`00`$$ | 11`00`00`00  Установка каретки команд на число из регистра А |
|  |  | 11`00`00`01  Установка каретки команд на число из регистра Б |
|  |  | 11`00`00`10  Установка каретки команд на число из регистра В |
|  |  | 11`00`00`11  Установка каретки команд на число из регистра Г |
| Установка каретки команд на значение из память данных по адресу из регистра | 11`00`01`$$ | 11`00`01`00  Установка каретки команд на число из памяти данных по адресу из регистра А |
|  |  | 11`00`01`01  Установка каретки команд на число из памяти данных по адресу из регистра Б |
|  |  | 11`00`01`10  Установка каретки команд на число из памяти данных по адресу из регистра В |
|  |  | 11`00`01`11  Установка каретки команд на число из памяти данных по адресу из регистра Г |

|  |  |  |
| --- | --- | --- |
| А→Ак | = |  |
| Б→Ак | = |  |
| Ак→А | = |  |
| Ак→Б | = |  |
| Ак+А→Ак |  |  |
| Ак-А→Ак |  |  |
| А-Ак→Ак |  |  |
| Ак+1→Ак |  |  |
| Ак-1→Ак |  |  |
| →Ак |  |  |
| →Ак |  |  |
| →Ак |  |  |
| →Ак |  |  |
| Ак>>→Ак |  |  |
| Ак<<→Ак |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |