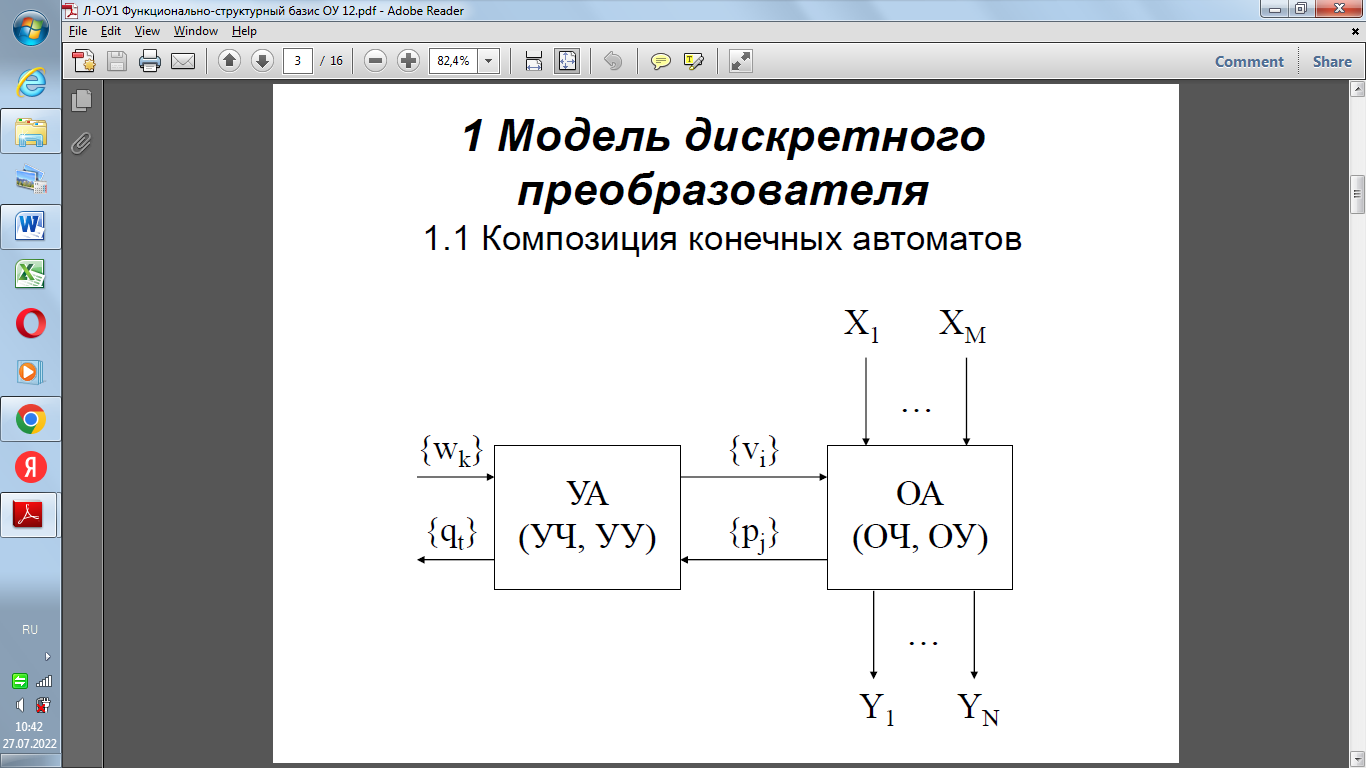
**Операционное устройство**

Предназначено для выполнения операций над операндами в соответствии с кодом выполняемой команды.



УА – формирует код выполняемой команды

ОА – выполняет действие, соответствующее коду

ОА выполняет действие при помощи микроопераций (МО)

Распространенные МО:

- хранение, пересылка

- счет, суммирование

- сдвиги (арифметические и логические)

- логика (конъюнкция, сумма по модулю 2, инверсия и т.д. )

Реализация микроопераций (МО счета) :

- КС инкремента

- накапливающий счетчик

- АЛУ

- сдвиговый регистр

- ПЗУ

Формирование логических условий(ЛУ)

S – sign – знак

Z – zero – ноль

C – перенос

OVR – переполнение

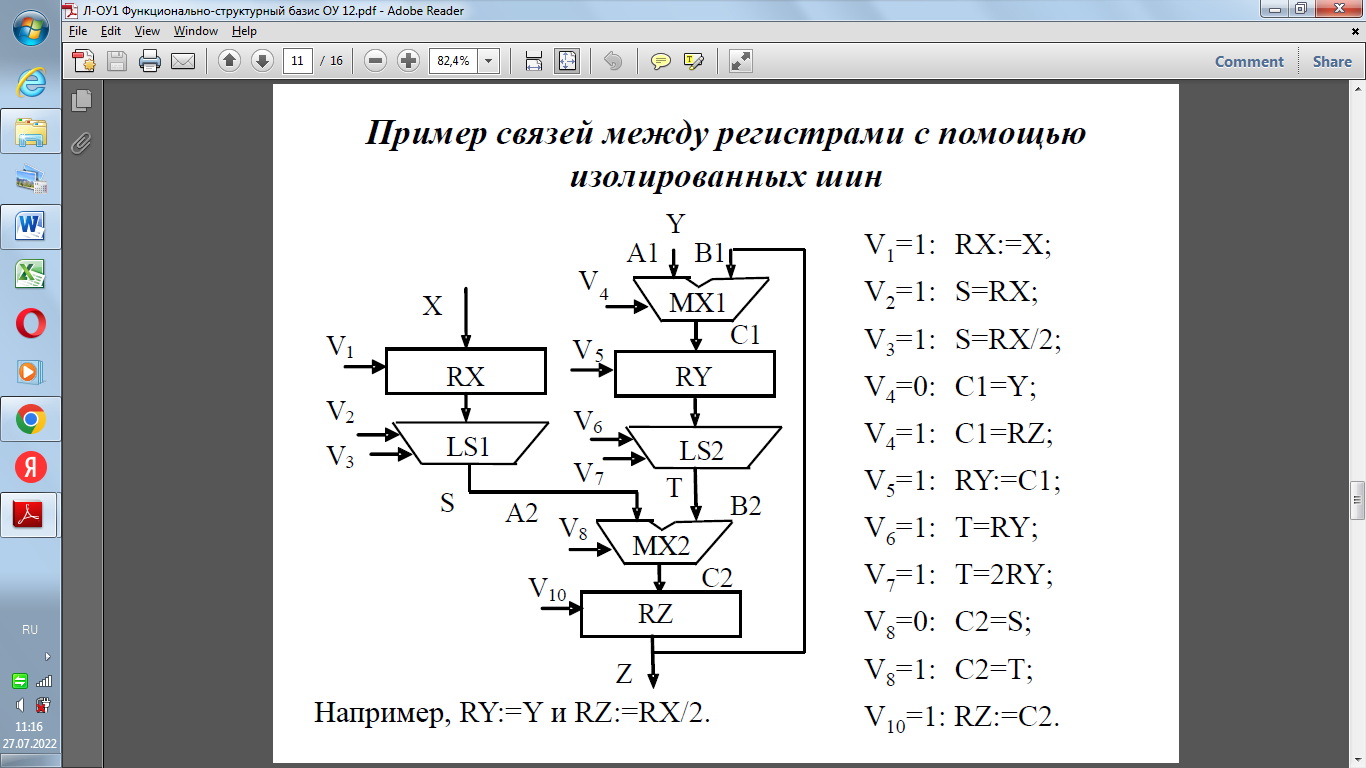
Данные, приходящие в ОУ, хранятся в регистрах. Обработка информации предполагает передачу её из регистра в регистр. Это можно сделать несколькими способами:

1. создание индивидуальной связи (изолированные шины)

То есть мы соединяем регистр-источник с регистром-приемником

- если таких связей мало или связь один к одному, то можно соединять напрямую

- если таких связей много или данные из регистра-источника уходят не всегда в один регистр, то добавляется мультиплексор, который координирует эти передвижения



1ый способ плох в том случае, когда у нас много связей между регистрами. Нам придется устанавливать много связей и это усложнит читаемость

2. магистрали

Способы создания:

- монтажное или

- объединение регистров с отключаемыми выходами

- регистры с обычными выходами, но выдача данных управляется логическими элементами

Базовые структуры ОУ

1. ОУ с закрепленными МО

- строятся на изолированных шинах

- строятся из многофункциональных регистров

- число регистров = числу данных для хранения инфы

2. ОУ с общими МО

- блок регистров

- логический преобразователь – АЛУ+ схемы

- связываются регистры магистралями

Пример ОУ с общими МО

Структура и микроинструкции

ОУ состоит из следующих блоков:

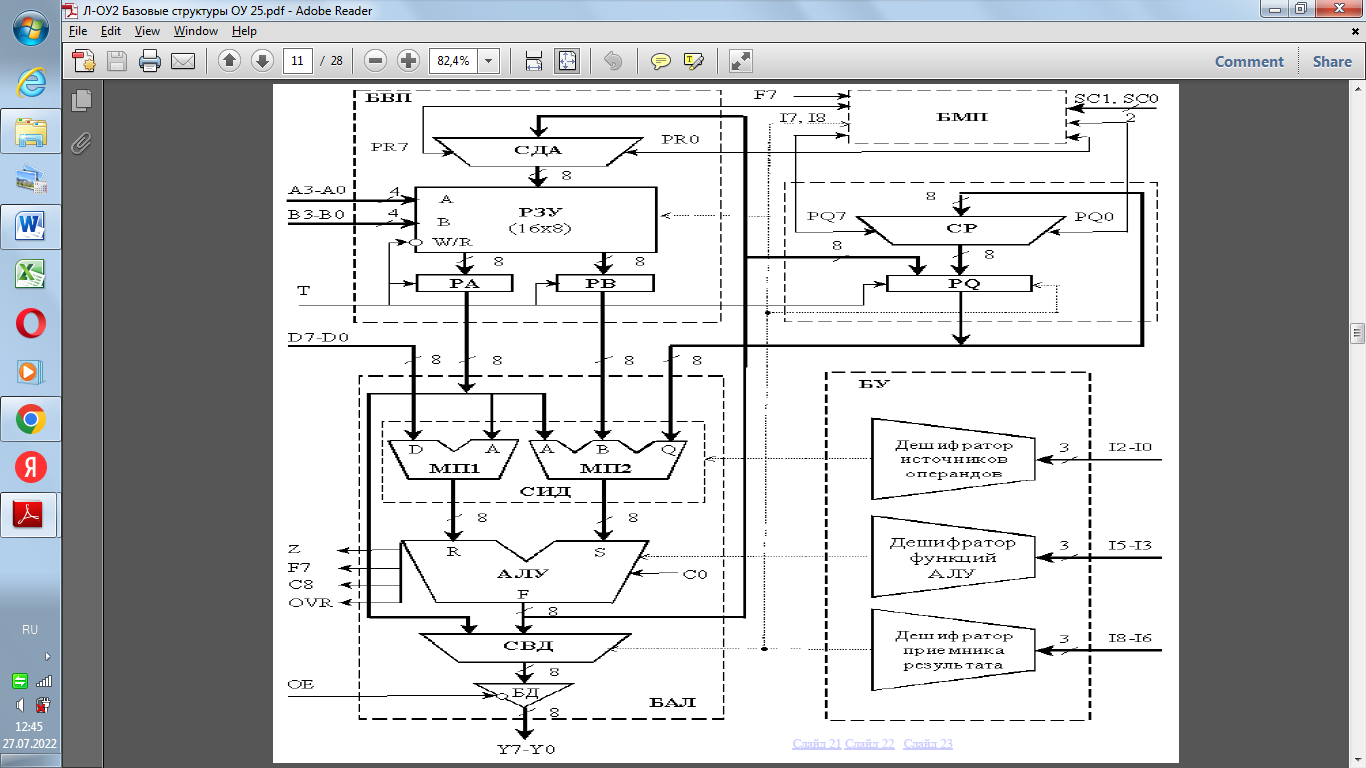
- арифметико-логический

- внутренней памяти

- регистра

- мультиплексора

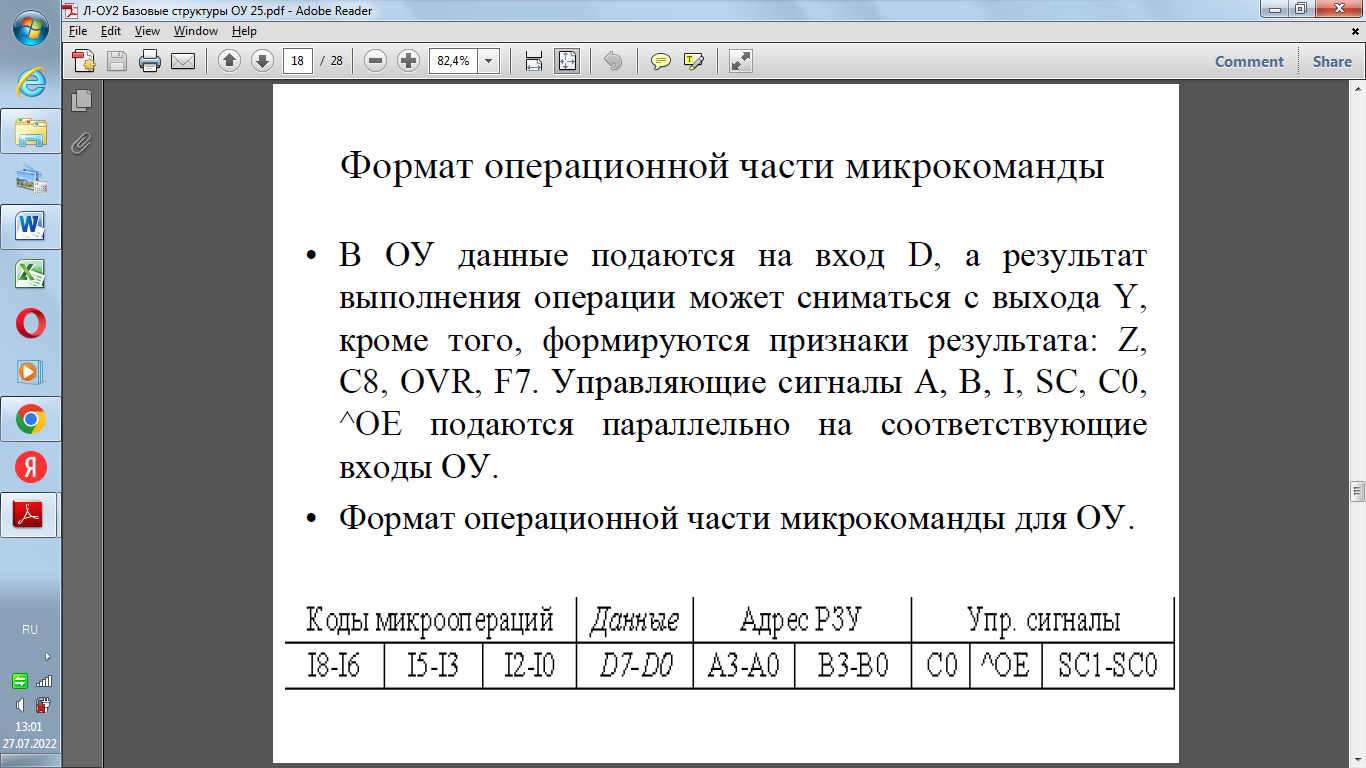
- управления



Для хранения данных выделяется 16 регистров(0-15), разрядностью 8. Доступ к адресам через

A3-A0 для Р0-Р7, и B3-B0 для Р8-Р15

Формат для МО



I2-I0 – откуда взять данные

I5-I3- что сделать

I8-I6 – в какой регистр положить результат

SC1-SC0 – команды для логического сдвига

С0 –

**Запоминающие устройства**

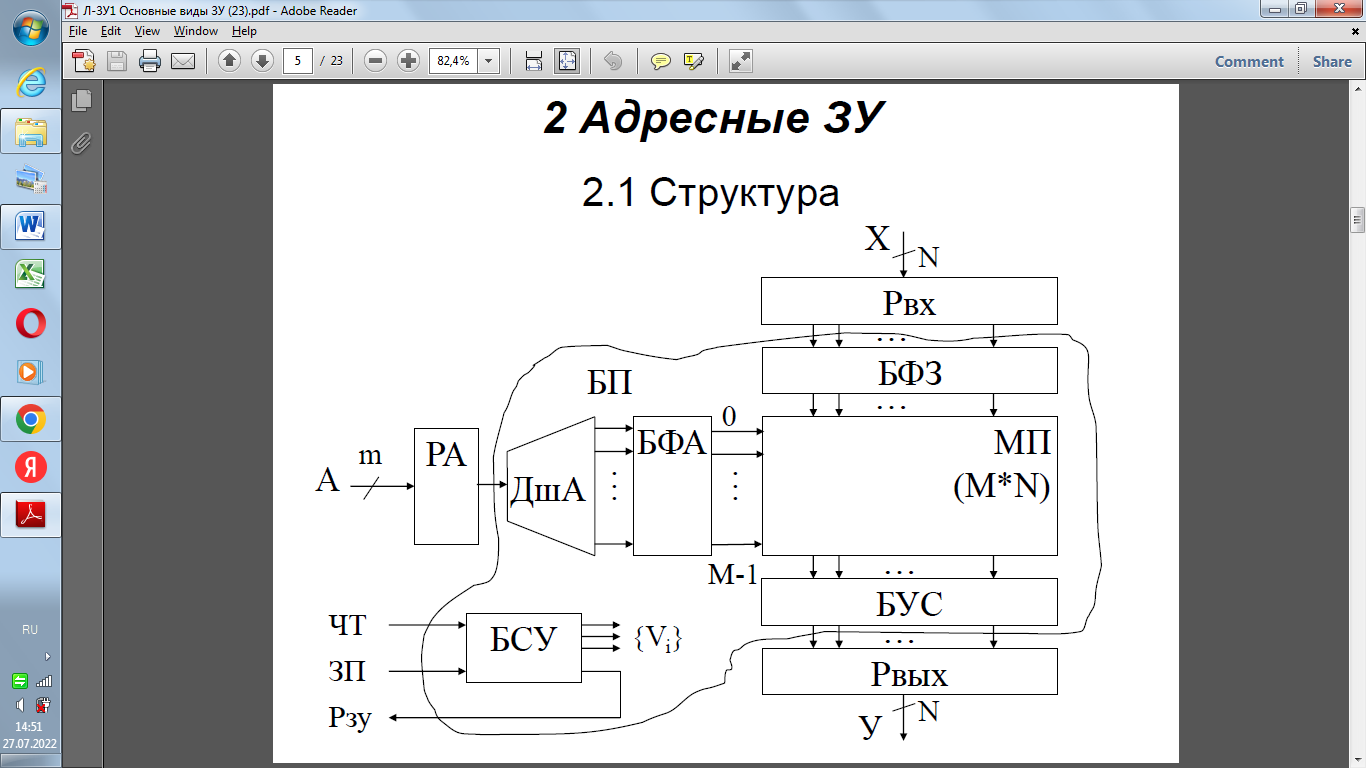
Классификация по принципу обращения:

- адресные (поиск по номеру ячейки)

- стек или очередь

- ассоциативная (поиск по содержанию)

Адресные ЗУ



Адрес поступает в регистр адреса, потом он преобразуется в физический адрес при дешифратора и записывается в блок формирования адреса.

(это чем-то похоже на принцип виртуальной памяти в приложении)

Входные данные хранятся во входном регистре, потом они поступают в блок формирования записи.

В момент выборки информации идет сравнения кода адреса с кодом полученной ячейки, и выдается ячейка при положительном результате. Всё это поступает в блок усилителя считывания и выдается в регистре выхода.

Стек или очередь ЗУ

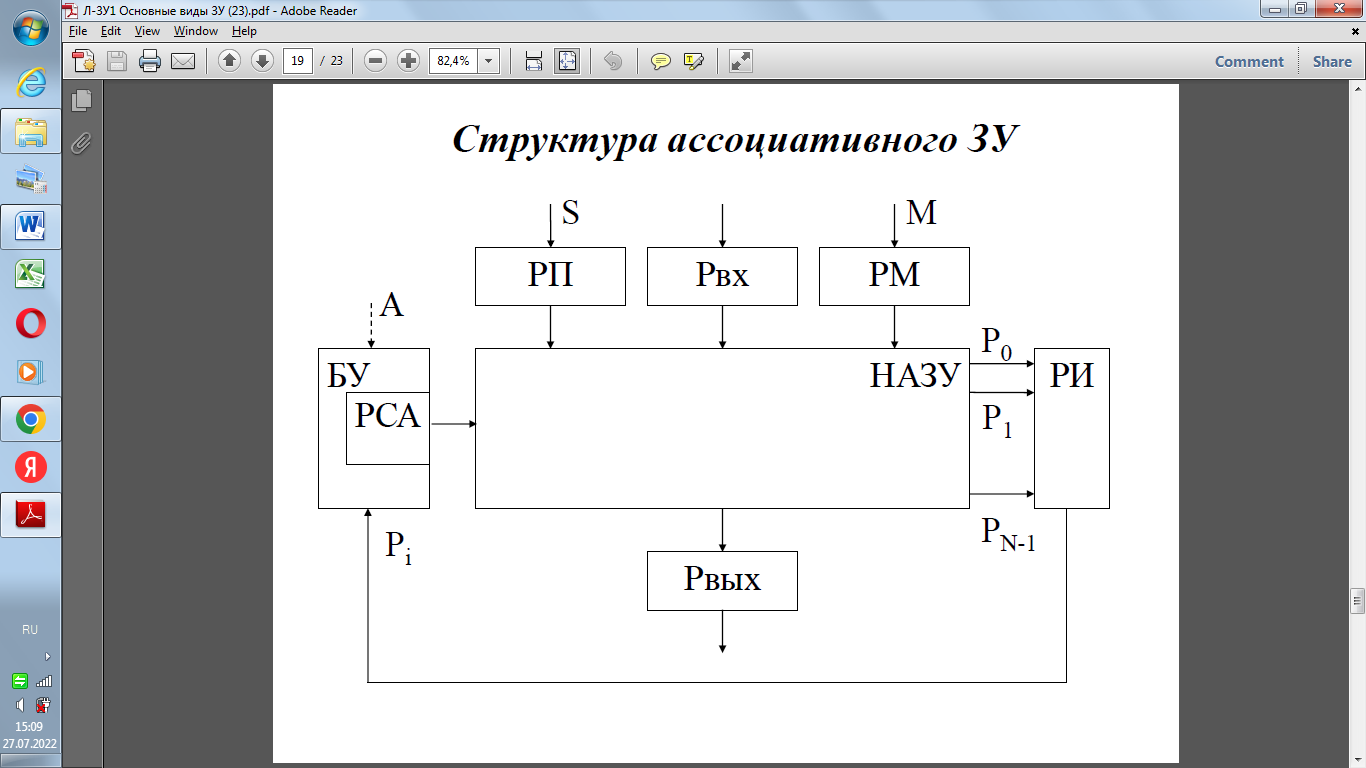
Работают так же, как и аналогичные структуры данных в программировании

Применение(стек):

- точки возврата при подмикропрогамм

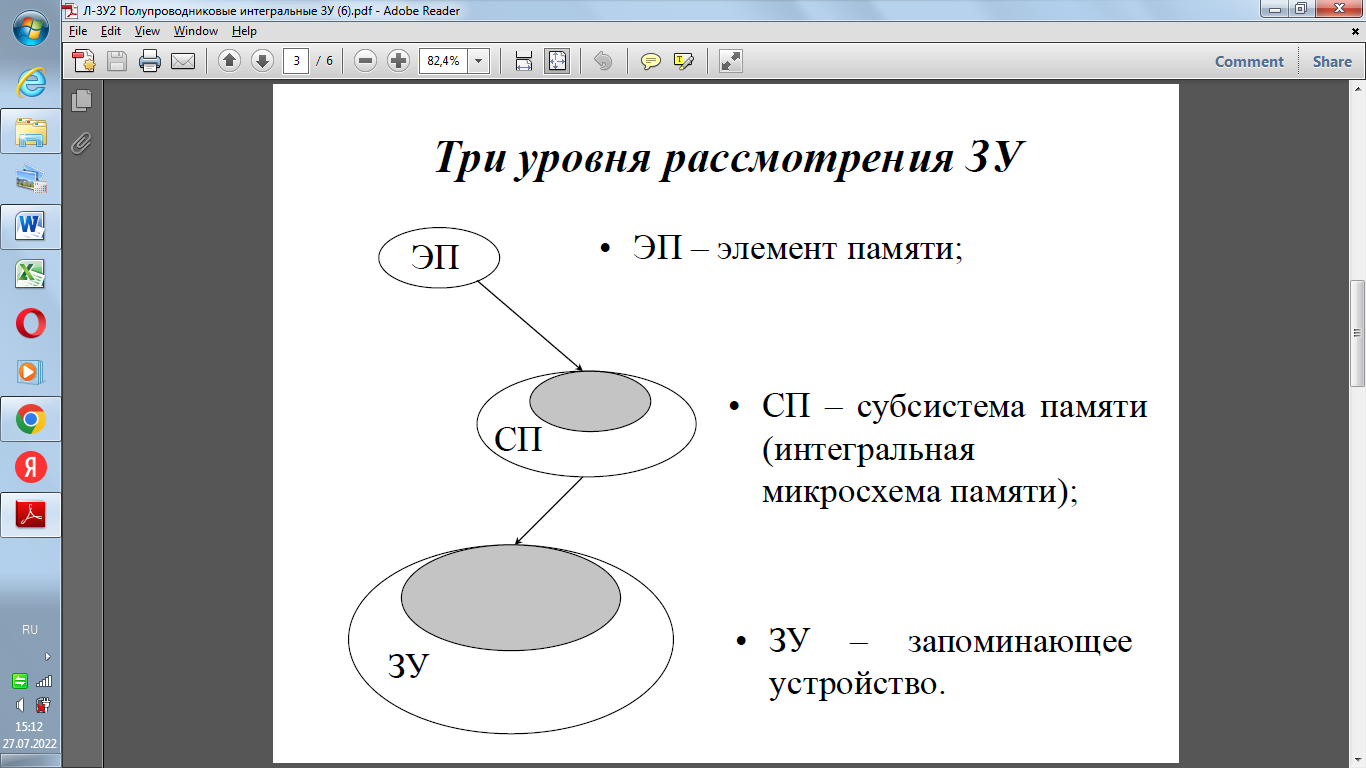
- сохранение состояния внутренних регистров процессора в точках прерывания

Ассоциативные ЗУ



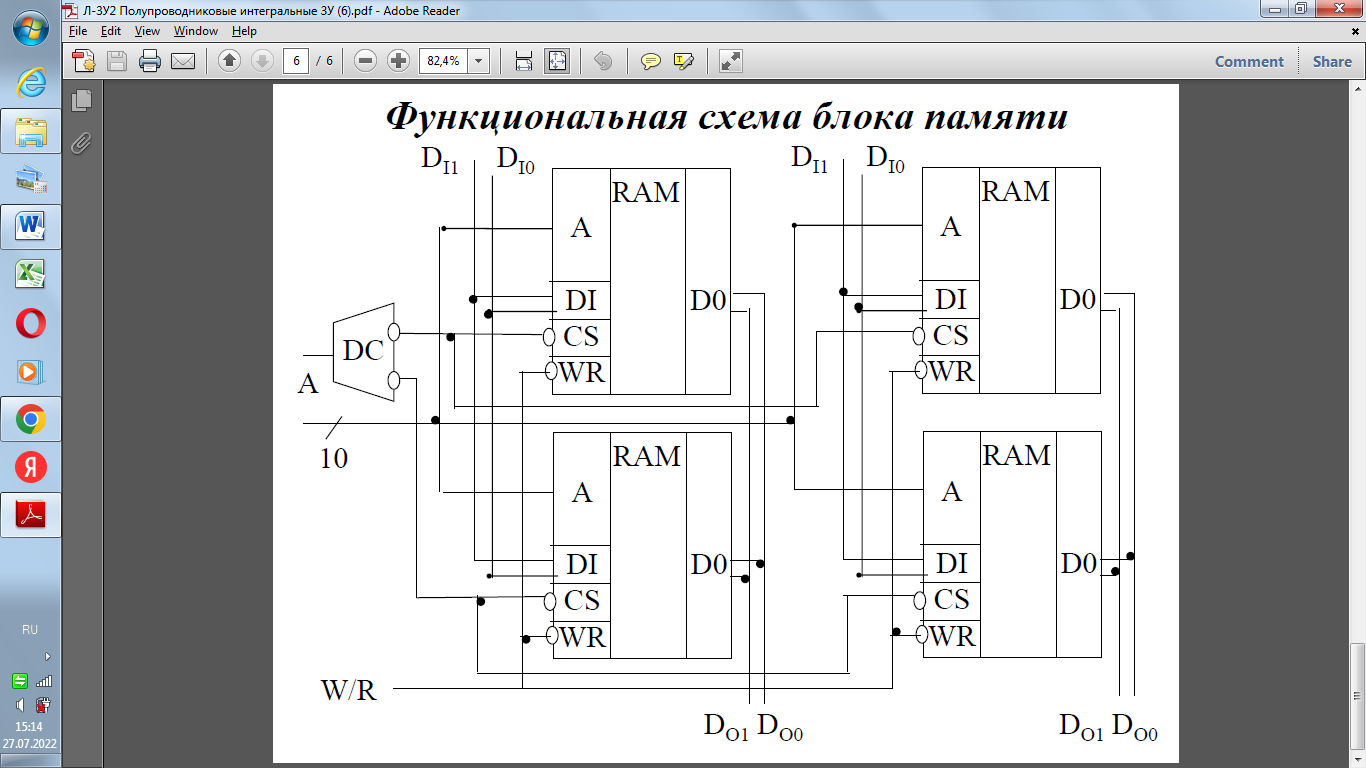
АЗУ применяются для преобразования виртуальных адресов в физические

3 уровня рассмотрения ЗУ



Интегральные микросхемы памяти

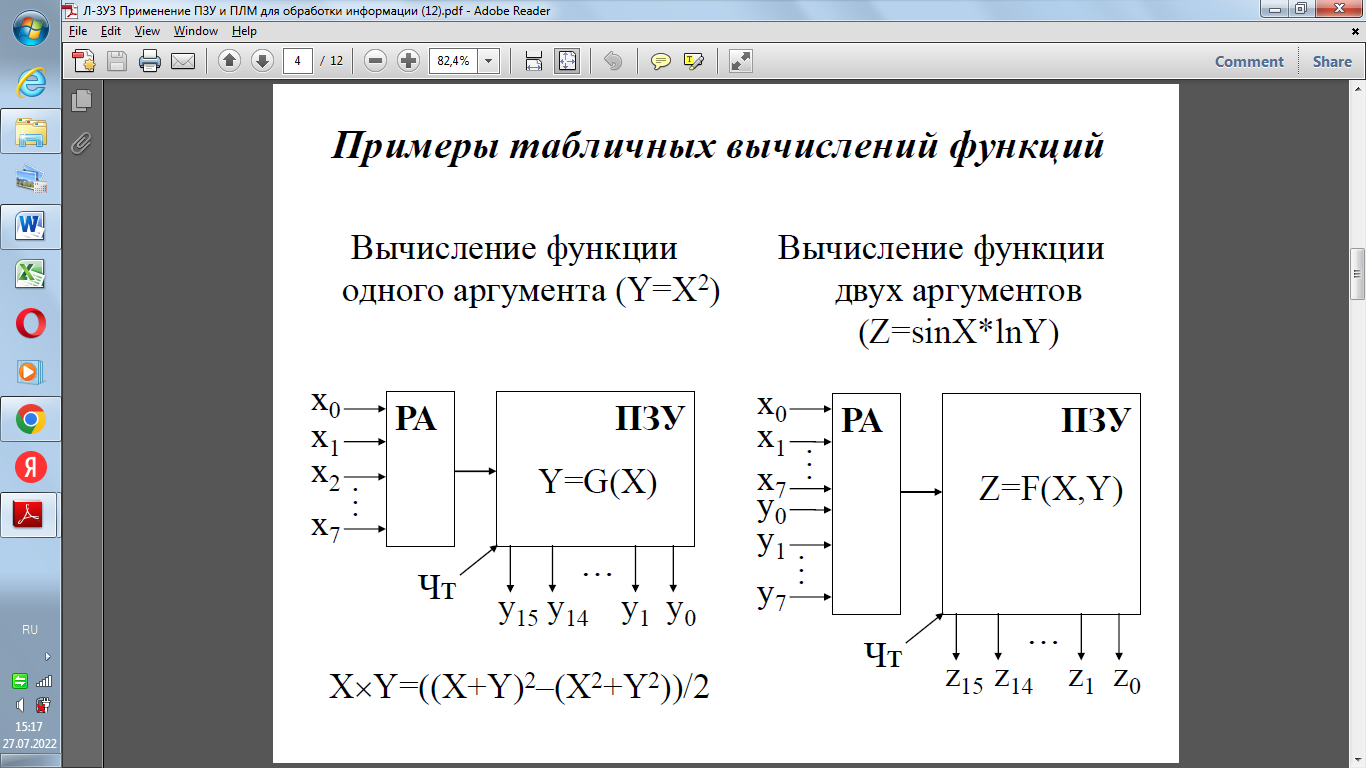
RAM



Применение ПЗУ и ПЛМ

ПЗУ – табличная реализация функций

1. арифметические операции и элементарные функции



Тут смысл в том, что мы сразу вычисляем все значения, которые нам могут потребоваться при вычислении. Заносим их в регистры ЗУ, а потом просто обращаемся к уже вычисленным значениям по адресу

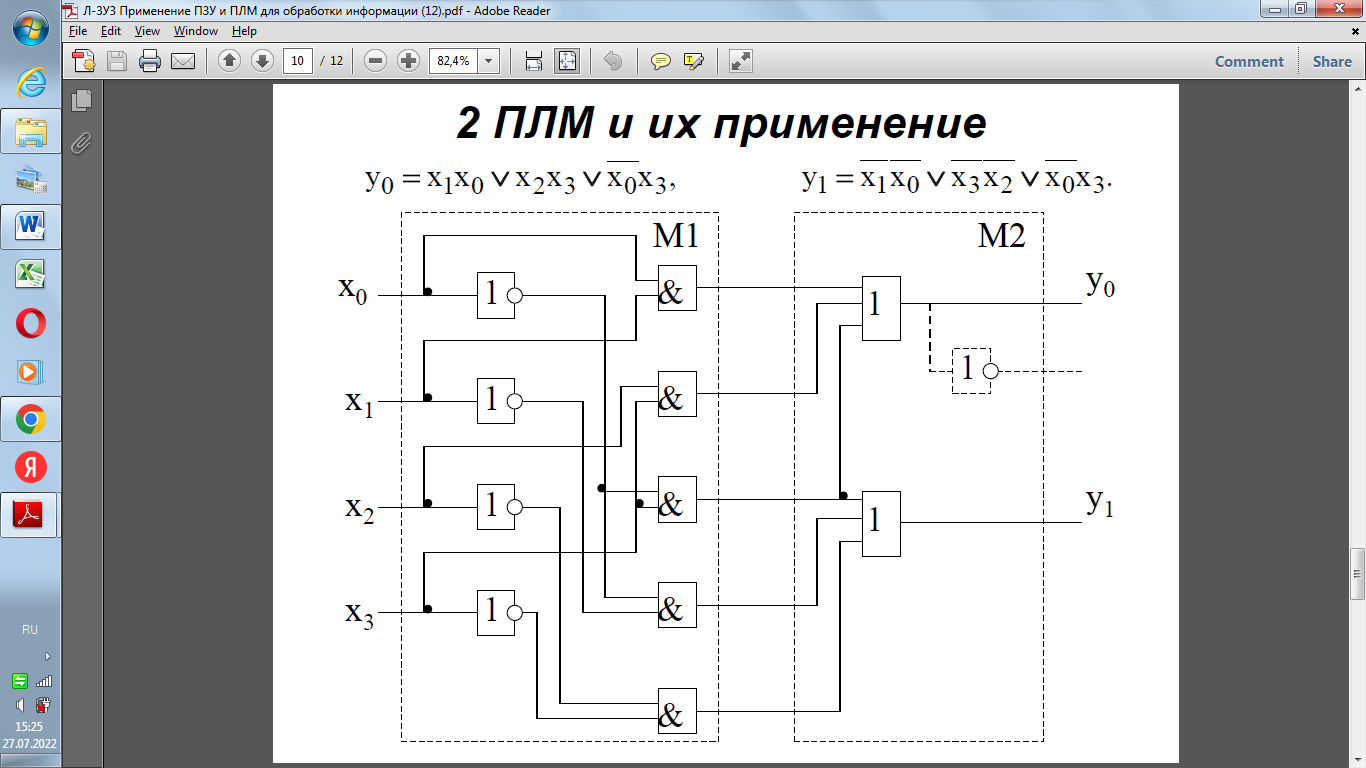
2. вычисление логических функций

Принцип такой же, как и у арифметических вычислений, за исключением того, что в адрес придется кодировать (то есть говорить, что пусть 000- x xor y и т.д.)

3. знакогенераторы

Хз как они работают

ПЛМ



М1 – матрица конъюкторов

М2 – матрица дизъюнкторв

Они соединены шинами

Применение:

- автоматы с жесткой логикой (реализация функций возбуждения, переходов, выходов)

- вычисление булевых функций

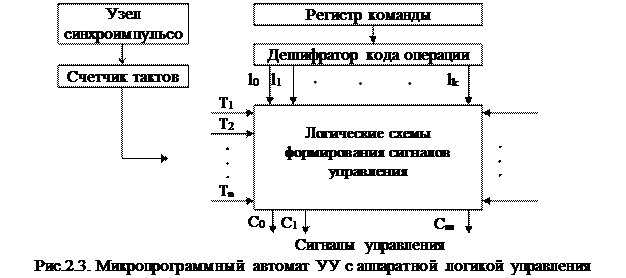
- преобразователь начального адреса (код операции в начальный адрес соответствующей микропрограммы)

**Управляющее устройство**

Классификация:

- с жесткой логикой

Выходные сигналы формируются за счет соединения между собой логических схем

https://helpiks.org/helpiksorg/baza8/58408224663.files/image008.gif https://helpiks.org/helpiksorg/baza8/58408224663.files/image007.gif

Информацией для УУ: содержимое регистра команд, флаги, тактовые импульсы и сигналы, поступающие с шины управления

Код операции, содержащийся в регистре команд (РК), используется для определения того, какие сигналы управления (СУ) и в какой последовательности должны формироваться.

Чтобы упростить логику управления код операции преобразуется в сигнал на j-ом проводе при помощи дешифратора.

Машинный цикл выполнения любой команды состоит из нескольких тактов, поэтому к узлу синхроимпульсов прикручен счетчик тактов. К выходам счетчика подсоединен дешифратор тактов.

Кроме того, на формирование СУ оказывают влияние флаги, поступаемые в УУ.

Применение: архитектура RISC

Дурацкий пример:

Нам дают 3 маленьких задания: дописать конспект, попить чаю, погладить кота.

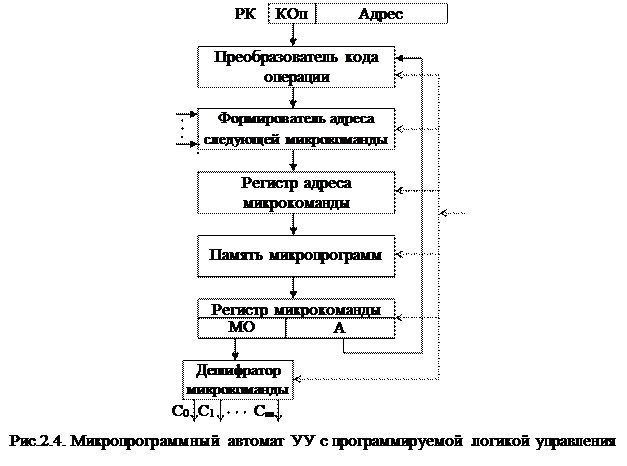
В случае устройства с жесткой логикой для нас все 3 маленькие задания – 3 разные микрооперации, код которых поступает в регистр команд последователь (то есть сначала первая, после выполнения – вторая и по ее завершении третья)

Итак, нам поступает код операции (набор 0 и 1), мы отправляем его на дешифратор кода, чтобы расшифровать, что нам нужно сделать. После этого мы формируем сигналы для операционной части, чтобы выполнить действия (Например, мы формируем сигнал для нажатия на клавиши, в ОУ идет нажатие этих клавиш; в качестве флага мы используем «точку», как конец предложения и конец микрооперации; как только мы поставили точку в УУ уходит флаг конец операции). Как только флаг устанавливается в конец операции, в регистр команд поступает новый код операции и происходит ее выполнение.

- с программируемой логикой

В основе лежит идея, что для инициирования любой микрооперации достаточно сформировать соответствующий СУ на соответствующей линии управлении

Для указания микрооперации, выполняемых в данном такте можно сформировать слово, в котором каждый бит соответствует управляющей линии. Слово – микрокоманда

https://helpiks.org/helpiksorg/baza8/58408224663.files/image010.gif https://helpiks.org/helpiksorg/baza8/58408224663.files/image011.gif 

Запуск выполнения операции осуществляется путем передачи кода операции из РК в преобразователь кода операции, где он преобразуется в начальный (первый) адрес микропрограммы Ан. Этот адрес поступает в РАМ через ФАСМ. Выбранная по адресу Ан из ПМП заносится в РМК. Каждая микрокоманда состоит из микрооперации, которая поступает на дешифратор микрокоманды, откуда выдаются сигналы управления. Адресная же часть микрокоманды поступает в ФАСМ, где формируется адрес следующей мк Амк.

Информация о том, какие сигналы должны быть сформированы в процессе выполнения текущей микрокоманды, закодированы в микрооперационной части (МО) микрокоманды.

3 типа кодирования:

- горизонтальное

под каждый сигнал управления выделен один разряд

Плюсы:

- не нужно раскодировать сигналы

- выходы регистров МО можно напрямую подключать к соответствующим управляемым точкам

- можно формировать любые сочетания СУ = максимальный параллелизм

Минусы:

- большие затраты на хранение МО частей

- при большом числе микроопераций в каждой отдельной МК выполняется 1 или несколько из них, остальные нули

- вертикальное

Каждой микрооперации присваивается свой код. Например, порядковый номер в полном списке возможных операций. Этот код и заносится в МО.

Плюсы:

- маленькие затраты на хранение МО частей

Минусы:

- нужен дешифатор

- увеличение времени выполнения за счет дешифратора

- смешанное

Микрокоманда разделяется на поля, число которых определяется числом максимальным числом выполняемых одновременно команд

МО распределяются по полям так, чтобы в одном поле находились МО, которые не выполняются одновременно никогда

Производится кодирование МО , если при выполнении микрокоманды не выполняется ни одна из МО, то вводится пустая МО

Плюсы:

- уменьшение части МО

- если дешифратор передать в ОА, то будет минимизация

Минусы:

- увеличение времени выполнения за счет дешифратора

- сложность модификации

Адресация микрокоманд

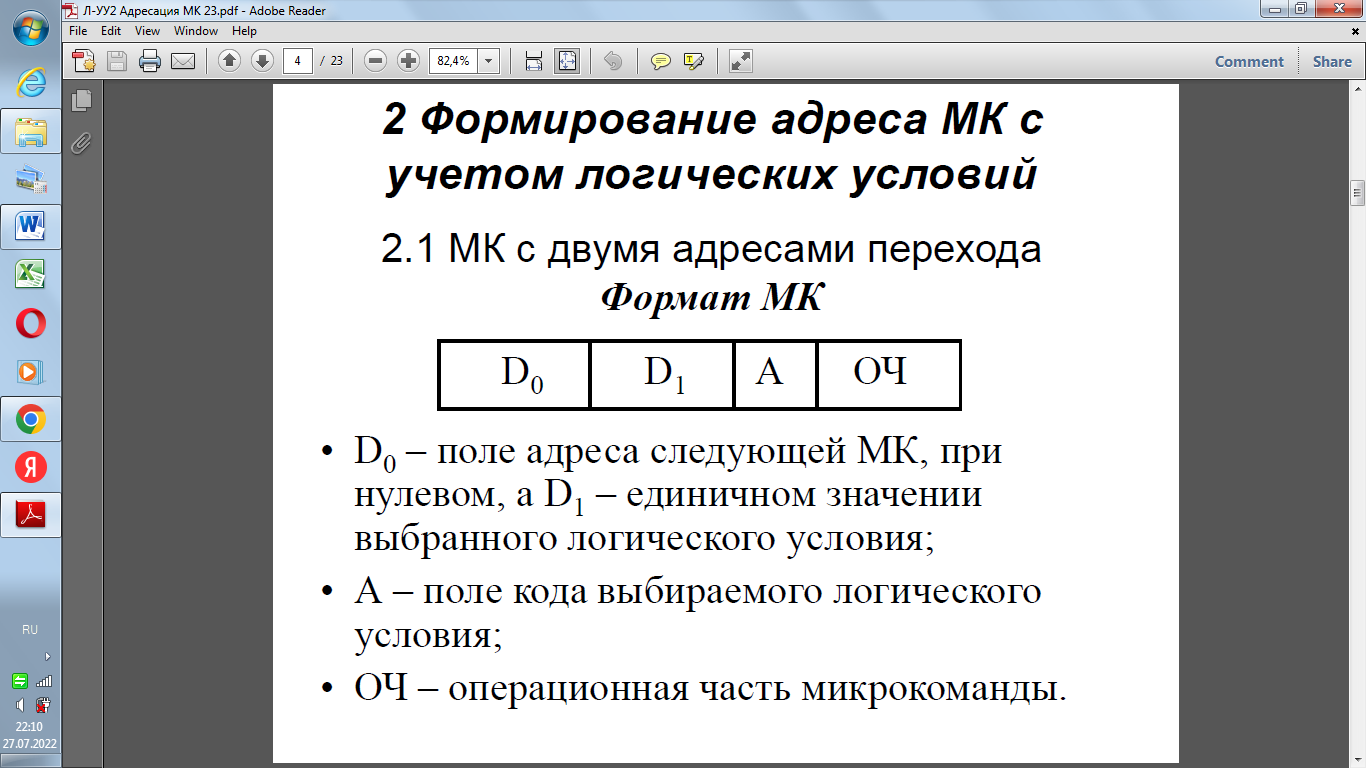
Естественная:

Адрес следующей МК образуется путем добавления 1 к адресу текущей МК

Принудительная:

В каждой МК, включая операционные, указывается адрес следующей за ней МК

МК с 2мя адресами перехода



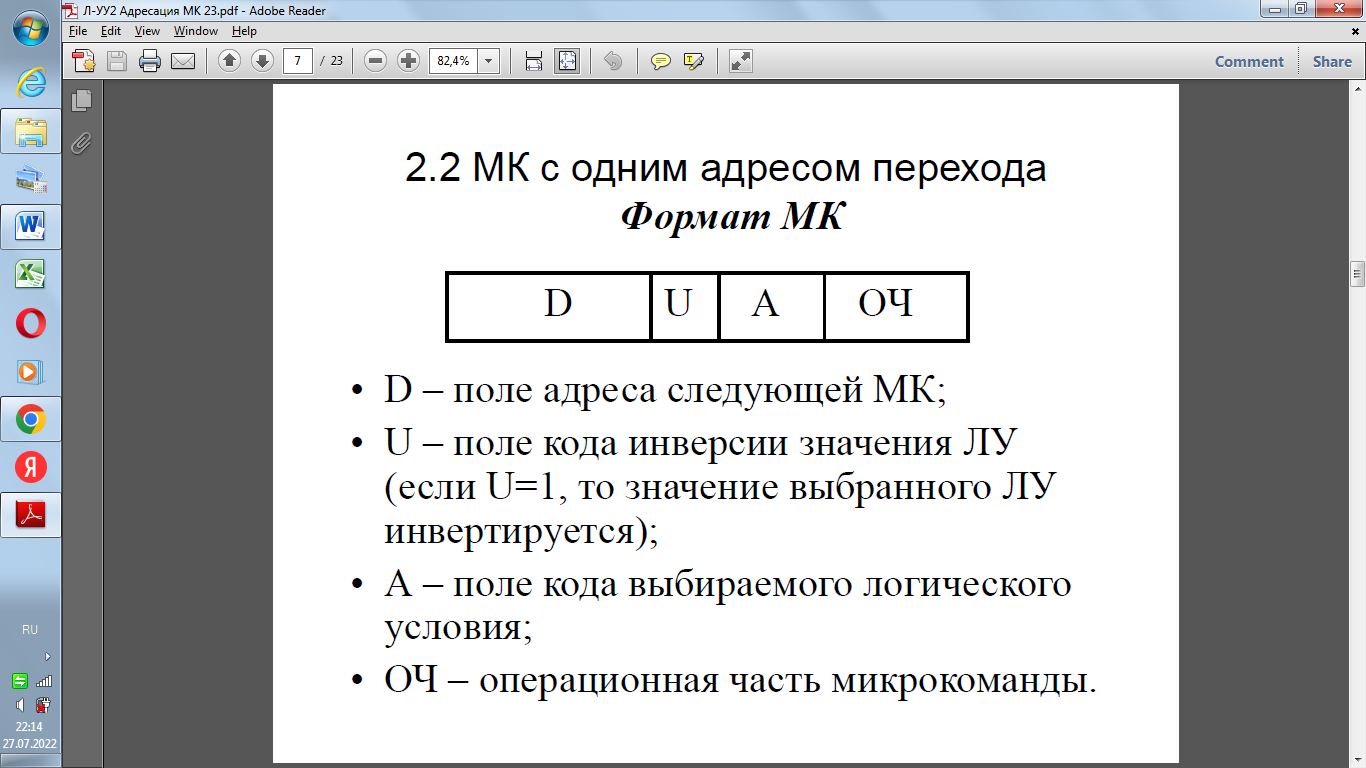
D0 – поле адреса следующей МК (при 0 логического условия)

D1 – (при 1 логического условия)

A – поле кода выбираемого логического условия

ОЧ – операционная часть

МК с 1м адресом перехода



**Вычислительные устройства**