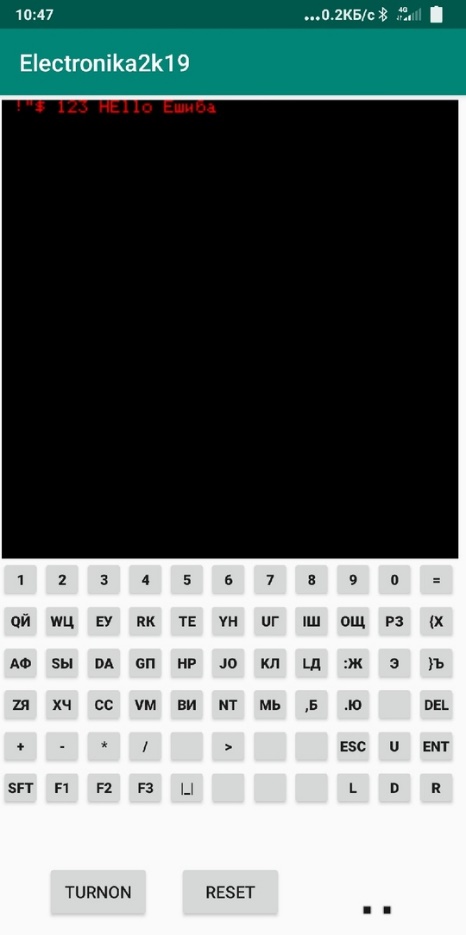
## Выбор платформы и языка разработки.

Цель проекта, создать кроссплатформенный эмулятор виртуальной машины, который сможет запуститься на максимально большом количестве устройств, от компьютера до мобильного телефона. Телефон, доступный мне, работает на платформе Android, которая запускает своё программное обеспечение внутри java virtual machine, поэтому изначально язык java был выбран для разработки эмулятора. Был разработан компилятор языка ассемблер в байт код, который в свою очередь запускался ядром виртуальной машины. Также был написан интерфейс для компьютера на библиотеке swing, универсальный для mac os, windows linux и интерфейс для платформы Android на Kotlin (Изображение 2). Но создав всё это, я понял, что не получил новых знаний и умений, что было целью данной курсовой работы. Поэтому начались поиски чего-то нового, языка или платформы, к которым я не прикасался во время основной работы или учёбы. Выбор пал на web разработку, а точнее интерфейс на html+css и ядро на js, что стало новым опытом для меня.



Изображение 2 : “Интерфейс на Kotlin под Android”

Направление проекта стало создание одной веб страницы, которая запускает весь код из единой js вставки, без развёртывания веб сервера и прочего, вроде nodejs и Apache.

Система должна запускаться по клику на \*.html файле и включать в себе всё необходимое.

## Структура готового продукта.

Физически проект состоит из основного electroniks2k19.html который запускает ядро lectroniks2k19.js, а также, стороннюю css библиотеку bootstrap для внешнего вида страницы.

Хотя по изначальной задумке сервера не планировалось, пришлось развернуться простейший http сервер на Python 3.8 HTTPServer. Это простейшая утилита, которая позволила расшить файловую систему проекта по http протоколу и по определённому порту в моей локальной сети, для отладки продукта сразу на нескольких устройствах.

Код сервера выглядит следующим образом:

*from http.server import HTTPServer, CGIHTTPRequestHandler*

*server\_address = ("192.168.68.110", 8000)*

*httpd = HTTPServer(server\_address, CGIHTTPRequestHandler)*

*httpd.serve\_forever()*

В код напрямую записывается ip адрес и протокол, и программа готова к работе.

## Язык описания алгоритмов работы виртуальной машины.

По своей сути, язык описания моей виртуальной машины, это модифицированный под нужные задачи ассемблер.

В качестве источников данных выступают регистры, прямая адресация в память и простое число, а также декларируемые статические переменные.

Для описания команд были использованы следующие сокращения:

RefIn от reference (анг. ссылка) и input (анг. ввод) – точка назначения, операнд или ячейка памяти, куда в результате выполнения команды записываются данные. (при этом нужно учитывать размеры цели назначения при записи).

RefOut от reference (анг. ссылка) и output (анг. вывод) – точка из которой берётся значение, для дальнейших действий.

Всего присутствует 10 команд:

MOV – Копирует значение из истопника данных в пункт назначения, мнемоника: MOV RefIn, RefOut

Пример:

*MOV AL, 25; //AL становится = 25*

ADD – Прибавляет RefOut к RefIn и записывает в RefIn. Действие команды вклиняет на флаги F0,F1,F2.

Пример:

*MOV AL, 2; //AL становится = 2*

*ADD AL, 25; //AL становится = 27*

ADC - Вычитает RefOut из RefIn и записывает в RefIn. Действие команды вклиняет на флаги F0,F1,F2.

Пример:

*MOV AL, 10; //AL становится =10*

*ADC AL, 2; //AL становится = 8*

MUL – Умножает RefOut на RefIn и записывает в RefIn. Действие команды вклиняет на флаги F0,F1,F2.2

Пример:

*MOV AL, 2; //AL становится = 3*

*MUL AL, 5; //AL становится = 15*

DIV – Делит RefOut на RefIn и записывает разность в регистр DL и остаток DH. Действие команды вклиняет на флаги F0,F1,F2.

Пример:

MOV AL, 9; *//AL становится = 9*

DIV AL, 5 // DL *становится* = 1; DH *становится* = 4;

INC – Увеличивает RefIn на 1. Действие команды вклиняет на флаги F0,F1,F2.

Пример:

MOV AL, 1; *//AL становится = 1*

INC AL; *//AL становится = 2*

DEC – Уменьшает RefIn на 1. Действие команды вклиняет на флаги F0,F1,F2.

Пример:

MOV AL, 1; *//AL становится = 1*

DEC AL; *//AL становится = 0*

СP – Сравнивает RefIn и RefOut. Действие команды вклиняет на флаги F3,F4,F5.

F3 – флаг включен, если результат операции “>”

F4 - флаг включен, если результат операции “<”

F5 - флаг включен, если результат операции “=”

Пример:

*MOV AL, 20; // AL становится = 20*

*MOV AH, AL; // AH становится = 20*

*CP AL, 10; // F4 = true, остальные false*

*CP AL, AH; // F5 = true, остальные false*

CALL – Вызов команды записанной по её имени в операнде, текущее состояние регбистов CP, SC записывается в стек.

Пример:

CALL DRAW; // Прыгаем в функцию DRAW состояние регбистов CP, SC записывается в стек.

CALLF - Вызов команды записанной по её имени в операнде, флаг,чей номер записан во второй операнд, включен. текущее состояние регбистов CP, SC записывается в стек.

Пример:

*MOV AL, 20; // AL становится = 20*

*CP AL, AH; // F5 = false*

*CALLF DRAW, 5; // 5 флаг выключен, поэтому не прыгаем*

*MOV AH, AL; // AH становится = 20*

*CP AL, AH; // F5 = true*

*CALLF DRAW, 5; // 5 флаг включен, прыгаем в функцию DRAW состояние регбистов CP, SC записывается в стек*

POP – Взять значение из стека, и записать его в RefIn

Пример:

*MOV AL, 25; // AL становится = 25*

*PUSH AL; // Помещаем в стек 25*

*POP AH; // Берём 25 из стека и помещаем его в AH*

PUSH – Отправить значение RefOut в стек

Пример:

*MOV AL, 25; // AL становится = 25*

*PUSH AL; // Помещаем в стек 25*

### Запись констант.

Константы записываются перед первой функцией программы, записывается сначала команда DECLARE затем имя константы, зачем значение переменной. Можно записать массив и строку;

DECLARE A = 55; //Создаёт константу A со значением 25

DECLARE VAL\_ARR = 45121; //Создаёт константу VAL\_ARR со значением 45121

Его можно записать в 16 битный операнд (AX,BX,CX,DX) а также можно получить одно из чисел,

MOV AL, $VAL\_ARR.[0] - AL становится =4;

DECLARE STR = “Hello Eshiba!” // Создаёт строковую переменную

Можно получить размер строки через MOV AL, $STR.SIZE - AL становится =4;

### Запись ссылок.

Числа пишутся следующим образом *MOV AL, 55; //Число 55*

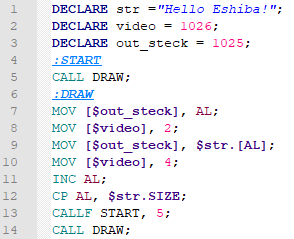
Нельзя писать в число (первым операндом).

Прямая ссылка на память *MOV AL, [1025]* - взять число из ячейки памяти и записать в операнд

Прямая ссылка на память *MOV [1025], [AL]* – взять адрес из операнда AL и записать в ячейку памяти 1025.

### 2.2. Первая программа - Hello world.

По канонам программирования первой программой сталла hello world (рисунок)



Изображение – Hello World

Первым делом идёт объявление констант:

Цель программы вывод строки поэтому объявляем строку str содержащую “hello Eshiba!”.

Затем указываем адреса video команд и выходного стека.

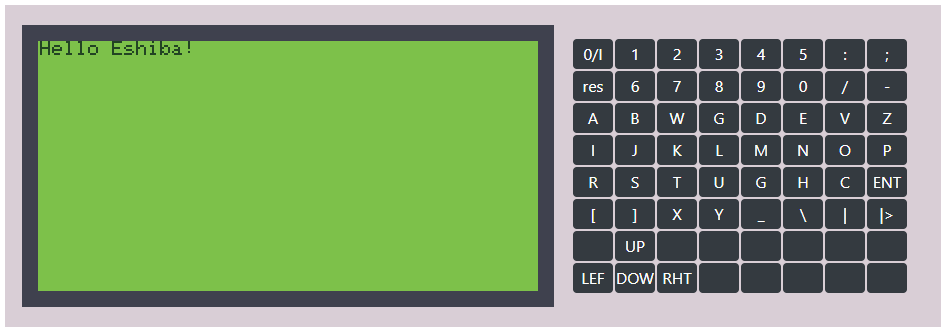
Программа стартует с функции START и сразу же прыгает в функцию DRAW.

Draw должна записать каждый символ на дисплей, выводя каждый символ друг за другом.

Для этого нужно получить номер сивола и поместить его во внешний стек (MOV [$out\_steck], AL), а затем вызвать 2 функцию видео контроллера, что бы записать положение сивола по оси X (MOV [$video], 2), затем получаем сивол мз переменной str и отправляем его в видео контроллер с помощью стека (MOV [$out\_steck], $str.[AL]) и команды 4 (MOV [$video], 4), проверяем вывелена ли вся строка, и если нет запускаем DRAW заново.

CP AL, $str.SIZE; CALLF START, 5;

Результат выполнения программы строка на дисплее (Изображение)



Изображение – вывод на дисплей программы Hello World

## Виртуальная машина

В основе данной виртуальной машины лежит симулятор простейшего, 8 битного процессора, который имеет 16 битную адресную шину, и 8 битную шину данных. Присутствуют 3 системных регистра PC – Program counter (указывает на адрес текущей функции), SC – steck counter (счётчик стека, указывает на верхний адрес стека), SC – stroke counter (счётчик команд, указывает на адрес текущей команды).

Также существует специализированный регистр, который в настоящем процессоре занимается регенерацией оперативной памяти, и используется программистами для реализации тандемных функций. В нашем случае, на регистре генерируется рандомное значение, а сам он состоит из двух частей (2 8битных регистра) RL,RH которые в свою очередь объединяются в 16 битный регистр RX.

Пользовательских регистров 8 штук AL, AH, BL, BH, CL, CH, DH, DL которые объединяются в AX