**Федеральное государственное образовательное бюджетное учреждение  
высшего образования  
«Финансовый университет при Правительстве РФ»**

**КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ**

**ОТЧЁТ**

**По лабораторной работе №1**

Студенты: Солнцева Ильи Андреевича

Дисциплина/Профессиональный модуль: Машинно-ориентированное программирование по защите информации

Группы: 3ОИБАС-818

Преподаватель

Сибирев И. В. /\_\_\_\_\_\_\_\_\_/

Оценка за работу: \_\_\_\_\_\_\_

Москва

2020

**Цель работы:** приобрести опыт работы с GitHub, а также развитие логических навыков в программировании.

**План проведения занятия:**

1. Ознакомиться с программным обеспечением VirtualBox.

2. Создать виртуальную машину исходя из предоставленной информации о минимальные аппаратные требования, предлагаемой к установке и изучению операционной системы (ОС).

3. Установить ОС на виртуальный компьютер. Разобрать процесс установки ОС на этапы.

4. Познакомиться с основными группами программ входящих в состав ОС.

**Оборудование:**

*Аппаратная часть:*персональный компьютер.

*Программная часть:*программаGitExtensions-Portable, Windows Seven, текстовый процессор Microsoft Word.

**Краткие теоретические сведения:**

 Под архитектурой ЭВМ (или архитектурой системы команд), чаще всего принято определять те средства процессора, которые видны и доступны программисту. Системой команд процессора называют полный перечень команд, которые способен выполнять данный процессор. Данные, доступные программисту, могут храниться в памяти (внешней по отношению к процессору) и регистрах (внутренней памяти процессора). При обращении к оперативной памяти в языках высокого уровня программист оперирует понятием переменной, а процессор – адресом размещения этой переменной в оперативной памяти. При обращении к регистру процессора программист определяет его по обозначению (имени), заданном разработчиком процессора, а процессор – по номеру (или адресу). Разработчики процессоров могут не разделять регистры по их назначению, имена таких регистров чаще всего отличаются номерами, например: R1, R2, R3 и т.д. Такие регистры называют регистрами общего назначения. Исторически сложилось, что регистры процессоров фирмы Intel имеют различия по назначению и, соответственно, имеют разные обозначения. Когда регистры имели 8 двоичных разрядов (процессор был восьмиразрядный), регистры обозначались одной буквой. Например, у процессора Intel 8080 были следующие обозначения регистров: A – аккумулятор (accumulator), предназначенный для хранения данных при выполнении арифметических и логических операций; B – базовый регистр (base), используемый для задания смещения адреса по базе; C – счетчик (count), для организации циклов; D – регистр для хранения данных (data). В соответствии с разрядностью шины данных (16 разрядов) у процессора Intel 8086 есть свой набор регистров. Чтобы показать, что разрядность регистров стала больше, к имени регистра была добавлена буква «X» (eXtended – расширенный). Получаются, соответственно, имена: AX, BX, CX, DX. У процессора Intel 8086 можно выбрать не весь регистр, а только младшую (Low) или старшую (High) часть регистра. Соответствующими будут и имена: AH, AL, BH, BL. У старшего поколения процессоров, с 32-х разрядной шиной, кроме всего сказанного для процессора Intel 8086, наименование 32-х разрядных регистров стало еще длиннее: EAX, EBX, ECX и т. д. Лабораторными заданиями предусмотрено изучение архитектуры процессора персонального компьютера, точнее говоря, в большей степени будет изучаться система команд процессора персонального компьютера во встроенном ассемблере Free Pascal, при этом за основу взяты команды для процессора Intel 8086.

4 1. АРХИТЕКТУРА ПРОЦЕССОРА Наименьшей единицей данных, с которой работает процессор, является бит (bit). Значением бита может быть либо ноль, либо единица. Группа из восьми битов называется байтом (byte) и представляет собой наименьшую адресуемую единицу – ячейку. Биты в байте нумеруют справа налево цифрами 0...7. Двухбайтовое поле образует шестнадцатиразрядное машинное слово (Word), биты в котором нумеруются от 0 до 15 справа налево. Байт с меньшим адресом считается младшим. Аналогично представляются 32-х разрядные слова. В процессоре принята двоичная система представления данных. Числовые данные кодируются в соответствии с двоичной арифметикой. Отрицательные числа представляются в дополнительном коде. Для удобства представления данных используется шестнадцатеричная система счисления. Принято двоичные числа сопровождать латинской буквой B или b, например, 101B, а шестнадцатеричные – буквой H или h на конце. Если число начинается с буквы, то обязательной является постановка нуля впереди, например, 0BA8H. Регистры В интегрированной среде Free Pascal можно просмотреть содержимое пятнадцати 32-разрядных регистров процессора, которые используются для управления исполнением команд, адресации и выполнения арифметических операций. Регистр, содержащий одно слово, адресуется по имени. Регистры сегмента CS, DS, SS, ES, FS и GS Регистр CS содержит начальный адрес сегмента кода. Регистр DS содержит начальный адрес сегмента данных. Регистр SS содержит начальный адрес регистра стека. Регистры ES, FS и GS дополнительные сегментные регистры. Регистры общего назначения EAX, EBX, ECX и EDX Регистры общего назначения являются основными рабочими регистрами ассемблерных программ. Их отличает то, что к ним можно адресоваться одним 32-х разрядным словом, 16-ти разрядным словом или однобайтовым кодом. Например, у регистра EAX можно использовать все 32 разряда, тогда будет использоваться все его имя EAX. Если используются два младших байта (16 разрядов), тогда его имя – AX. Из шестнадцатиразрядного регистра AX также можно выделить две части: младший байт AL и старший байт AH. Аналогично могут по частям рассматриваться и другие указанные регистры. Регистр EAX – аккумулятор, используется во всех операциях ввода/вывода, в операциях со строками и в арифметических операциях. Регистр EBX – базовый регистр. В процессоре Intel 8086 регистр BX был единственным из регистров общего назначения, используемым в косвенной адресации. 5 Регистр EDX – регистр данных. Используется в некоторых операциях ввода/вывода, в операциях умножения и деления больших чисел совместно с регистром EAX. Любой из регистров общего назначения может быть использован для суммирования или вычитания. Регистры указателя ESP и EBP Регистры указателя используются для обращения к данным в сегменте стека. Могут использоваться все 32 разряда или только младшие 16 разрядов, тогда имя регистра используется без буквы «E». Регистр ESP – указатель стека (stack pointer). Используется для определения адреса вершины стека. Регистр EBP – указатель базы (base pointer). Обеспечивает ссылки на параметры (данные и адреса, передаваемые через стек). Индексные регистры ESI и EDI Индексные регистры используются для адресации, а также для выполнения операций сложения и вычитания. В них могут быть использованы все 32 разряда или только младшие 16 разрядов, тогда имя регистра используется без буквы «E». Регистр ESI – индекс источника (source index). Используется в некоторых операциях со строками или символами. Регистр EDI – индекс приемника (destination index). Используется в тех же операциях, что и регистр ESI. Регистр указателя команд EIP Регистр IEP используется для выборки очередной команды программы с целью ее исполнения. Во Free Pascal программисту не доступен. Регистр флагов Flags Регистр Flags содержит девять активных битов (из 16), которые отражают состояние процессора и результаты выполнения машинных команд. Биты: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Флаг O D I T S Z A P C Флаги O (переполнения) – равен 1, если возникает арифметическое переполнение, например, при сложении числа 01111111B (127 десятичное) с числом 00000001B получится число 10000000B (-128 десятичное), то есть семь разрядов, используемых для представления абсолютной величины числа, переполнились, и был задействован знаковый разряд. D (направления) – устанавливается в 1 для автоматического декремента в командах обработки строк и в 0 – для инкремента. 6 I (разрешения прерывания) – прерывания разрешены, если I=1. Если I=0, то распознаются лишь немаскированные прерывания. T (трассировки) – если T=1, то процессор переходит состояние прерывания INT 3 после выполнения каждой команды. SF(знака) – S=1, когда старший бит результата равен 1. Иными словами, S=0 для положительных чисел, и S=1 для отрицательных чисел. Z (нулевого результата) – Z=1, если результат равен нулю. A (дополнительный флаг переноса) – этот флаг устанавливается в 1 во время выполнения команд десятичного сложения и вычитания при необходимости выполнения переноса или заема между полубайтами. P (четности) – этот флаг устанавливается в 1, если результат имеет четное число единиц. C (переноса) – этот флаг устанавливается в 1, если имеет место перенос или заем из старшего бита результата; он полезен для произведения операций над числами длиной в несколько слов, которые сопряжены с переносами и заемами из слова в слово. Сегменты Данный параграф в полной мере справедлив только для процессоров с шиной адреса менее 32 разрядов. У 32-х разрядного процессора нет необходимости в формировании полного 32-х разрядного адреса ячейки памяти с помощью каких-либо дополнительных регистров. Сегментом называется область памяти, которая начинается на границе параграфа, то есть в любой точке, адрес которой кратен 16 (восемь младших битов равны нулю). Существуют три основных типа сегментов:

− сегмент кода (CS) – содержит машинные команды;

− сегмент данных (DS) – содержит данные;

− сегмент стека (SS) – содержит адреса возврата в точку вызова подпрограмм, локальные переменные и параметры значения.

Каждый из упомянутых регистров содержит адрес начала сегмента (базовый адрес). Чтобы выполнить обращение к данным по любому адресу, процессор выполняет суммирование адреса, записанного в регистре сегмента DS, со смещением. При этом содержимое регистра DS сдвигается на четыре двоичных разряда влево, чтобы результирующий адрес занимал 20 позиций (для процессора 8086), что и позволяет адресовать 1 Мбайт памяти (220 = 1048586). Например, если в регистре DS было шестнадцатеричное число 045Fh, после сдвига влево на 4 двоичных разряда оно примет вид 045F0h. К полученному числу прибавляется смещение (например, 0032h), и получается исполнительный (или эффективный) адрес равный 04622h. 7 Режимы адресации Режимы адресации приведены в соответствии с возможностями встроенного ассемблера Free Pascal. В колонке «Режим адресации» приведено наименование режима. В колонке «Формат адреса», что используется в качестве операнда. В колонке «Стандартный сегментный регистр» – в каком сегменте по умолчанию располагаются данные. Режим адресации Формат адреса Стандартный сегментный регистр Регистровая регистр (указывается имя регистра) Нет Непосредственная данные (указывается число) Нет Прямая переменная (указывается имя переменной) DS [EBX] DS [EBP] SS [EDI] DS Косвенная регистровая [ESI] DS [EBX + смещение] DS [EBP + смещение] SS [EDI + смещение] DS Косвенная регистровая со смещением [ESI + смещение] DS [BX + DI + смещение] DS [BX + SI + смещение] SS Косвенная регистровая по базе со смещением и с индексированием [BP + DI + смещение] SS исходный адрес DS:SI Строковые команды место назначения ES:DI Стеки Стек – это структура данных типа LIFO (Last Input First Output, «последний пришел – первый ушел»). Наиболее важное использование стека связано с процедурами. Стек обычно рассчитан на косвенную адресацию через регистр ESP – указатель стека. При включении элементов в стек производится автоматический декремент указателя стека, а при извлечении – инкремент, то есть стек всегда «растет» в сторону меньших адресов памяти. Адрес последнего включенного в стек элемента называется вершиной стека (TOS – Top of Stack). Физический адрес стека формируется из ESP и SS или EBP и SS, причем ESP служит неявным указателем стека для всех операций включения и извлечения, а SS – сегментным регистром стека. Содержимое SS называется базой стека. Первоначальное содержимое ESP считается наибольшим смещением, которого может достигать стек. Регистр EBP предназначен, главным образом, для произвольных обращений к стеку.

8 2. СИСТЕМА КОМАНД ПРОЦЕССОРА Из всего набора команд процессора в лабораторных заданиях предусмотрено рассмотрение следующего набора команд: ПЕРЕСЫЛКА ДАННЫХ

MOV

PUSH

POP

XCHG

PUSHF

POPF

XLAT

LEA

LDS

LES

LAHF

SAHF

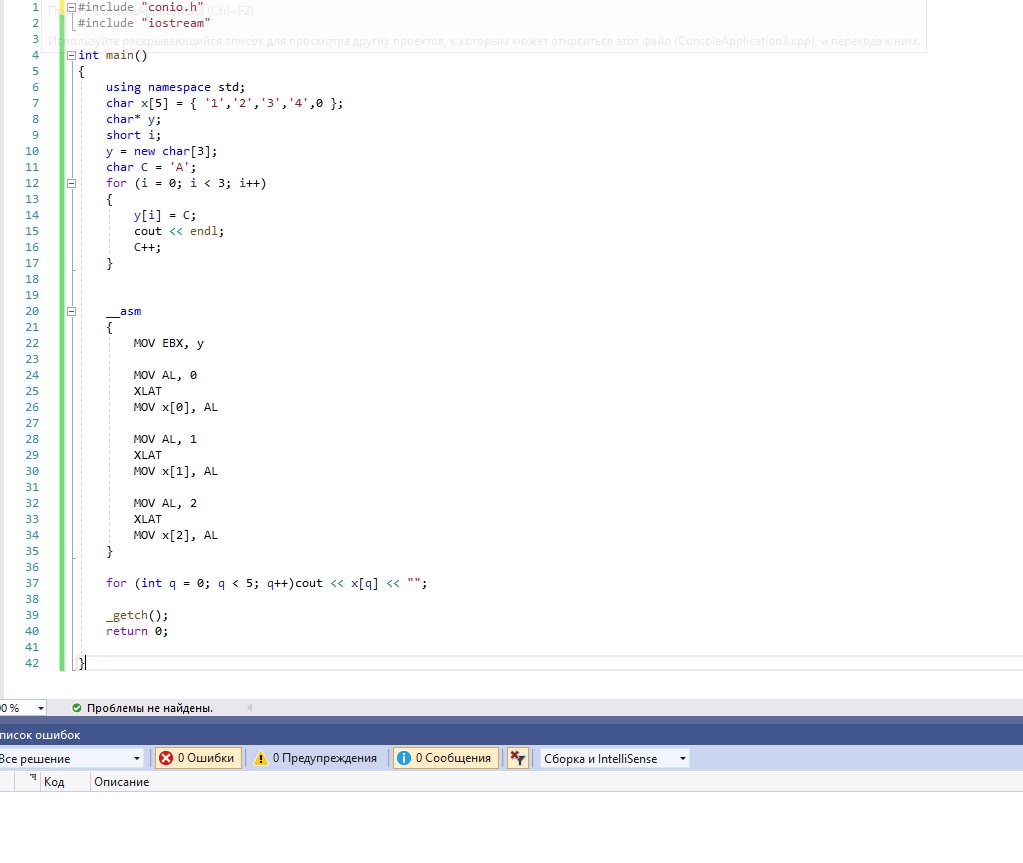
**Ход работы:**

Выполнить два задания:

1. Обменять значения в переменных языка Паскаль x: integer и y: ^integer.
2. Определите, сколько байт требуется на запись в оперативной памяти команды LEA EBX, M, и какие числа записаны в этих байтах.

Выполнение заданий со скринами кода программы:

1. Код написан в программе Visual Studio



1. Код написан в программе Visual Studio

