Глава 1

Организация ЭВМ на уровне ассемблера

1.1 Лекция 1

Замечание. *Максим Викторович Баклановский.*

Контакты:**baklanovsky**: skype, @mail.ru, @gmail.com
Варианты организации работы:

1. Загрузочная флешка.
2. Эмулятор(w32).
3. VM (Virtual Box).

Замечание. *Для эмулятора нужно попросить USBDOS.zip (по skype).*

Оффтоп. *В будущем нужно будет сделать доклад по настоящим спецификациям. Intel. Гук - 9 ссылок. По легальным ресурсам. Длинная арифметика. Ларри Уолл - создатель Perl.*

1.1.1 Работа с эмулятором

Разархивируем в C. NC далее. NC.bat его на рабочий стол(там пути). MAX каталог. Источники информации Shift + F1 = Norton Guide(Help по asm). Options DB = Подгружает базы. Справочник программиста. (Лукач, Гук).

Ralf Brown - сбор информации, перерабатывает. Файл с ошибками процессоров. Каталог Ints с прерываниями (F4) - просмотреть. Cntl + B = функциональные кнопки. MultiEdit - редактор. (Alt + F4 = альтернативный редактор, F2 тут вызывате меню, Alt + F1 = список окон). F2 - свою меню в Norton Commander.

LU => IO.DOC читать тут.

Util - утилиты. pepores.com резидент - программа на постоянное присутствие. "Взять вектора"и получают переодически .

TSR - terminate and state resident. Можно посмотреть, где сидит Norton. Когда что-то из Нортона вызвали, то вся память у него, то он себя отрезает, ограничивая свой блок. Сюда загружается именно COMMAND.COM. Далее - свободная память, куда будут помещены наши программы.

NG => Лукач => undocumented.

Основные задачи - обработать прерывание с клавиатуры.

"pepores"показал нам память(dump). Можно узнать, где лежит наш резидент. Нужно найти в память своей программы.

Turbo ASM, macroASM. MASM. Ассемблеры, где TASM от Borland, MASM от Microsoft.

DOS => Work => Shift + F4 = новый файл. Пробуем написать код.

model tiny ; чтоб не было сегментов.

.code

ДЗ. *org 100h ; Смещение в 256 байт, где хранятся параметры, а адресация сегмента кода, начинается с ; 100h*

here: jmp start

m1 db 'Hello, ASM!', 13, 10,'доллар'

start:

mov dx, offsset m1

mov ah, 9; когда вызываем прерывание, то передается управление бибилотчечной функции по адресу 21h

int 21h ; способ оформления библиотечной функции. MS-DOS call, который смотрит параметр из ah

ret ; - почему процессорная(ассемблерная) команда ret завершает исполнение программы. ; по сути ret возвращает управление, т.е. восстанавливает значение регистра команд их стека.

end here

далее Shift + F1, print string - ему соответствуте int 21h прерывание + 9 в ah, а то, адрес того, что мы выводим должен помещаться в dx.

Вспомнить Рюриковича - сегментная адресация. F10 - выход их NG.

Сохранили и создаем объектный файл с именем "1"

tams /m 1

Замечание. */? - узнать ключи. m - кол-во проходов. Написать программу, на которой ASM сделает столько проходов, сколько мы хотим.*

Проходим линковщиком

tlink /x/t 1

.COM файл, который в отличие от .EXE файла имеет ограничения на размер(64 КБ). Как-то связано с 100h.

Сделаем резидент программу 2.com - на постоянное присутствие в памяти. Примером резидет-программы является pepores. menu => mem (тут можно его увидеть).

model tiny

.code

org 100h

m1:jmp start

m2:db 'Hello, I am TSR',13,10,'доллар'; shift F1 ASCII char

start:

mov dx, offset m2; смещение считаем с нуля.

mov ah, 9 ;

int 21h ; передает управление на библиотеку

mov dx, 0 ; пишем и выходим. потом 12h

mov ax, 3100h ;

int 21h ;

end m1

ДЗ. *Найти реpores't это место. Можно искать по тем символам-словам, которые есть в программе.*

1.2 Лекция 2

1.2.1 Прерывание.Таймер

Все началось с того, что определить версию процессора Intel не такая уж и простая задача. Но с появление команда **cpuid** проблема решилась. Но в процессорах до Pentium этой функции не было, поэтому приходилось что-то придумывать.

Поступил вопрос: "Как посмотреть, где находится программа во время выполнения?". Как попсмотреть это место, не делая её программой резидентом? Из всех предложеложенных вариантов больше всего подошел вариант с breakpoint'ами. Но посчитали, что вставив в код прерывание сделаем лучше.

xor ax, ax;

int 16h ; прерывание bios по обслуживанию клавиатуры

Вообще прерывание бывают(это не полный список, как я понял):

1. Программные
2. Внутренние
3. Аппаратные

Заметим, что аппаратные прерывания зависят от таймера. Таймер зависит от системного времени. Тест был в том, что мы написали hlt (тем самым остановив процессор), но так как аппаратный таймер производит отсчет 18.2 раза в секунду, то он практически сразу же "завел"процессор.

Вообще есть Real Time Clock который обеспечивает переключение процессов в многозадачных системах, причем там таймер делает отсчет 1024 раза в секунду. Поэтому квант процессорновго времени приблизительно равени 1 милисекунде.

1.2.2 Управление.Отладчик

Передать управление, есть не что иное, как перейти к исполнению другой команды. Помним, что за это отвечает регистр IP = Instruction Pointer. В чем идея? Когда одна подпрограмма вызывает другую, то адрес возврата помещается в stack, причем когда вызываемая программа завершает работу, то она ret'ом загружает из stack'a в IP адрес вызывающей подпрограммы.

Отметим, что int param; - где param по сути означает семещение, которое мы отсупаем, чтобы попасть на нужную библиотечную функцию. По этому адресу как раз и передается управление.

Заметим, что когда вызываем tsr, то там вложенно все вызывается. nc => command.com => tsr. Но tsr знает, что после того, как он отработает, он вернет управление и закончит работу, поэтому в памяти он себя не прописывает. Command.com на самом деле тоже возвращает управление (т.е. меняет регистр IP), но он все ещё отображается, поэтому это искажет наше предствление.

Замечание. *Передать управлениие - делать переход на область памяти*

ДЗ. *int x => CD | X (такое представление). Намек на то, что алфафит OPCODE'ов переполнен. Почему для breakpoint'а int 3 выделили отдельный байтовый опкод CC?*

Замечание. *Если мы видим, что первый байт, в 2-х байтной связке для OPCODE равен OF, то это нам говорит о том, что мы должны смотреть в другой таблице команд.*

ОС всегда завершает программы, поэтому мы можем увидеть в начале нашей программы символы **CD 20**, что говорить нам о том, что вызывалось прерывание int 20h (прервать программу = terminate programme).

Ещё раз вспоминаем, что ret восстанавливает в IP адрес из stack.

Посмотрели в памяти, что отступ, который мы прописывал *orh 100h*, использовался для размещения параметров. Заметим, что это смещение учитывалось, когда мы писали offset m1.

Интересная вещь, то то, что мы объявляли строку, как набор 1-байтных элементов, поэтому и писали "db"(data byte). Чтобы взять не 1 символ, а 2, например, нужно будет указать "word ptr m1 где word подразумевает 2 байта.

1.3 Лекция 3

Организационные вопросы: дополнительная пара для проверки домашенго задания.

ДЗ. *Задача: написать свой примитивный отладчик. 2 способа: реализовать троассировку и breakpoint. Загрузить файл - будет проблемно. Выполняем функции операционной системы.*

MS-DOS: Тим Патерсон. DR-DOS: Гарри Киллалл (король 8-битных операционок). Ему IBM предлагали контракт, но он отказался.

Как открыть файл? MS-DOS переняла наследие от других ОС. Фредерик Брукс - получил Тьюринговскую премию. Shift + F1 - искать функции для работы с файлами.

OF(15) устаревшая функция. Используем функции с 3Dh (60-го) номера. Пользуемся этими функциями. Важное слово handle(ручка к ящику). Чтобы открыть файл нужно "описать ящик". Есть основные handle: stdin, stdout, stderr. Для нас тут handle с 5-м номером. Обращаем внимание на зависимость от CF (для возврата информации об ошибки). JNC (jump no carry) удобная команда. Перкидываем handle в BX (посмотрели в close). Нужен буть READ. Поиск длины файла: lseek(66-я функция) с конца до 00. Эрик Рэймонд. После открытия нужно делать seek 3-го типа. Так узнаем длину файла. Можно длину указать (FF). Из диска - в память загрузились и больше не читаем. Нужно обработаться все завершения работы. Ограничения: отлаживаемая программа .COM и завершается ret'м. Сам отладчик .COM.

Понадобится стэковый фокус. Прерывания - 2е дело(внутреннее дело).

Посадить резидента.

Прыжок на 107, но 107 нет. в 108

90 - универсальная затиралка. Когда мы не ставим ключ /m => получили "nop".

E9 2 байта (величина прыжка).EB 1 байт.

Ассемблер, стоя на строчке - не знает, где будет старт. То он кладет максимум, 3 байта. Если заменит 3 на 2, то адреса поедут, поэтому оставляет 3 байта и вставляет "nop". С ключем /m мы уже будем знать, что обойдемся 2-мя байтами, т.е. на следующий проход сделаем все хорошо, без nop.

ДЗ. *Задача - сделать столько проходов, сколько мы задаем.*

Дизъасемблирует все подрядр. Процессор не понимает разницу между данными и инструкциями. Идет по логике линейоного ассемблера.

Замечание. *Говард Эйкен. Алан Тьюринг. 2 разных типа архитектуры: Гарвардская, Принстонская.*

Попали нетуда, но IP верно указывает. Cntr + G и прямо указываем то, что нужно.

0 в стеке положил DOS, чтобы ret'm сделать CD 20. Что лежит в стэке, чтобы мы вернулись. call делает call и кладет следующий за собой адрес.

В начале сегмента лежит наша программа.

ДЗ. *Нужно стэком убить свой код. Чтобы стэк дошел до наших инструкций и затер код.*

Golf - решить задачу, уменьшив размер программы.

ret - один, но возвращать он будет нас в разные места.

Как смотреть результат работы ассемблера без дебагера? Получаем листинг программы (tasm /m/l 5.asm).*shl ax, 4* => разваливалась в 4 команды *shr ax,1*. Это некруто. Написали .486 - разрешили использовать команды 486 процессора. Теперь, посмотрел листинг, видим, что у нас все заменилось на одну 3-х байтовую команду.

1.3.1 Пишем прерывания

Использование программных прерываний. Пишем свой код. Пишем свою функцию для вывода строки.

256 векторов. Младшие 8 - (0-7) процессорные исключения. Пример 0-е исключени - это деление на 0. Процессор не может выполнить команду. Следующие 8 штук (8-F) аппаратные int'ы первого контроллера, 8 таймер, 9 клавиатура. (10h - 1Fh) BIOS'а прерывания, 13 - работа с диском, 16 клавиатура. (20h - 2Fh) забрал себе DOS. Итого: 48 номеров. Посмотреть в Brown'е что осталось для пользователей. Это от (F0-FF) зарезервировано изначально.

iret выталкивает 3 слова. Заполняет IP, CS, RFLAGS. 25, 35 поставить вектор, снять вектор. ds всегда обещан как cs.

Делаем диспетчеры и вызываем разные функции. Shift + F3 = убрать выделение. Написали разные функции, вызываемые в зависимости от нашего условия.

Гарри Киллдал. Создавал массив точек входа. Можно все тесты убрать. Диспетчеризация в начале. В самих функциях ничего не пишем.

mov dx, (offset start + 15)/16;

Ошибка будет. Relative quantity illegal. Тут требуется какая-то типизация. Смещение не рассматривает как смещение. Нужно вычесть то, и добавить 100h.

Разделяли на клиентскую и резидентскую часть. Заполнилась колонка: взятые вектора.

Посмотрели разраотку сервисного разработичка прерываний.

ДЗ. *Думать про задачу. Взять вектор трассировка - первый int, breakpoint 3 - й int*

Нужно грамотн проследить, куда возвращаться. Флагие трассировки. Сначала перехватить int 1. Нужно отследить завершение программы.

загрузить программу в память. Сохранить регистры. Установить флаги. Печатать все IP. Когда ret, должны выйти. Breakpoint на IP.