3 Ответы на контрольные вопросы

Вопрос 1 Что такое сегментный (базовый) адрес? В микропроцессоре есть несколько 16-битных регистров, которые называются сегментными и обозначаются как CS, DS, SS и ES. В них хранятся 16-битные значения, которые называются базовым адресом сегмента. Микропроцессор объединяет 16-битный исполнительный адрес и 16- битный базовый адрес следующим образом: он расширяет содержимое сегментного регистра (базовый адрес) четырьмя нулями в младших разрядах, делая его 20-битным (полный адрес сегмента), и прибавляет смещение (исполнительный адрес). В результате получается 20-битный адрес, который является физическим или абсолютным адресом ячейки памяти. Физический адрес нулевого смещения есть начало сегмента в виртуальной памяти.

Вопрос 2 Сделайте листинг для первой программы (файл с расширением lst), выпишите из него размеры сегментов. Из таблицы трассировки к этой программе выпишите базовые адреса сегментов (значение DS при этом нам нужно взять после инициализации адресом сегмента данных). В каком порядке расположились сегменты программы в памяти? Расширяя базовый адрес сегмента до физического адреса, прибавляя размер этого сегмента и округляя до кратного 16 значения, мы можем получить физический адрес следующего за ним сегмента. Сделайте это для первых 2-х сегментов. (Если данные не совпали, значит, неверно заполнена таблица трассировки.) Code size 0011 (PARA), data size 0014 (PARA), stak size 0100 (PARA). Выпишем регистры после инициализации регистра DS: • DS = 48BD • SS = 48AD • CS = 48BF В памяти программа разделена на сегменты, которые расположены в порядке увеличения размера: SS, DS, CS. Физический адрес определяется как базовый адрес, умноженный на 16h, плюс размер сегмента. То есть расширенный базовый адрес сегмента 𝑆𝑆 = 48𝐴𝐷 ⋅ 10 + 0100(𝑠𝑡𝑎𝑘𝑠𝑖𝑧𝑒) = 48𝐴𝐷0 + 0100 = 48𝐵𝐷0(кратно 16) = 48𝐵𝐷 = 𝐷𝑆. 8 Аналогично для DS = 48BD ⋅ 10 + 0014 (data size) = 48BD0 + 0014 = 48BE4 (не кратно 16, округляем до ближайшего кратного 16 вверх) = 48BF = CS.

Вопрос 3 Почему перед началом выполнения первой программы содержимое регистра DS в точности на 10h меньше содержимого регистра SS? (Сравниваются данные из первой строки таблицы трассировки) Потому что сегменты расположены последовательно и выровнены по 0x10, сегмент стека весит 0x100 байт, а в сегментных регистрах адреса памяти хранятся без разрядов, отвечающих за смещение.

Вопрос 4 Из таблицы трассировки к первой программе выпишите машинные коды команд mov AX,data и mov AH,09h. Сколько места в памяти в байтах они занимают? Почему у них разный размер? mov AX, data - B8BD48 0003 - 0000 = 3 байта сдвиг - размер машинного кода. mov AH, 09h - B409 0005 - 0000 = 2 байта сдвиг - размер машинного кода. Размер команд разный в связи с разным типом данных, помещаемых в регистры.

Вопрос 5 Из таблицы трассировки ко второй программе выпишите базовые адреса сегментов (значение DS при этом нам нужно взять после инициализации). При использовании модели small сегмент кода располагается в памяти первым. Убедитесь в этом. (Если это не так, значит, вы неверно заполнили таблицу трассировки.) • DS = 48AF • SS = 48B1 • CS = 48AD Сегменты в модели small расположились в следующем порядке: CS, DS, SS. Убедимся в этом: 𝐶𝑆 = 48𝐴𝐷 ⋅ 10ℎ + 0011(code size) = 48𝐴𝐷0 + 0011 = 48𝐴𝐸1 (не кратно 16, округляем вверх) = 48𝐴𝐹0 (кратно 16, значит можем условно отбросить ноль в конце) = 48𝐴𝐹 = 𝐷𝑆. 𝐷𝑆 = 48𝐴𝐹 ⋅ 10ℎ + 0012 = 48𝐴𝐹0 + 0012 = 48𝐵02 (не кратно 16, округляем вверх) = 48𝐵1 = 𝑆𝑆, в чём и следовало убедиться. 9

Вопрос 6 Сравните содержимое регистра SP в таблицах трассировки для программах 2 и 3. Объясните, почему получены эти значения. В программах модели TINY используется только сегмент стека, в связи с этимм он инициализируется максимальным возможным значением, поскольку он будет сдвигаться при помещении в него данных. В программе SMALL используются ещё сегменты данных и кода.

Вопрос 7 Какие операторы называют директивами ассемблера? Приведите примеры директив. Директивы передают ассемблеру метаданные, необходимые для создания объектного и исполняемого файлов или же листинга. Директива ASSUME передаёт ассемблеру информацию о соответствии между адресами сегментных регистраов и программными сегментами. Директива имеет следующий формат: assume [[, ]] assume nothing где  - это  : либо  :NOTHING Например, директива assume es:a, ds:b, cs:c сообщает ассемблеру, что для сегментирования адресов из сегмента А выбирается регистр ES, для адресов из сегмента В – регистр DS, а для адресов из сегмента С – регистр CS.

Вопрос 8 Зачем в последнем предложении end указывают метку, помечающую первую команду программы? Программа на языке ассемблера состоит из программных модулей, содержащихся в различных файлах. Каждый модуль, в свою очередь, состоит из инструкций процессора или директив ассемблера и заканчивается директивой END. Метка, стоящая после кода псевдооперации END, определяет адрес, с которого должно начаться выполнение программы и называется точкой входа в программу. Каждый модуль также разбивается на отдельные части директивами сегментации, определяющими начало и конец сегмента. Каждый сегмент начинается директивой начала сегмента – SEGMENT и заканчивается директивой конца сегмента – ENDS . В начале директив ставится имя сегмента. 10 Таким образом, метка, указанная в END определяет начальный адрес, с которого процессор должен начать выполнение команды.

Вопрос 9 Как числа размером в слово хранятся в памяти и как они заносятся в 2-ух байтовые регистры? В зависимости от архитектуры процессора, применяется прямой или обратный порядок байт. Почти во всех современных процессорах байты с меньшим адресом считаются младшими, такой порядок называется Little Endian. То есть 2 байта ложатся в 2 байта и сначала байт с младшими битами числа, затем байт со старшими. В случае с Big Endian (прямым порядком байт) ситуация обратная, сначала идут старшие разряды: так, как мы привыкли записывать числа на бумаге.

Вопрос 10 Как инициализируются в программе выводимые на экран текстовые строки? Выводимые на экран текстовые строки инициализируются в секции .data с помощью db, строка должна оканчиваться знаком $. Прежде чем делать int 21h нужно в DX положить адрес начала строки mov dx, offset имя\_метки\_с\_которой\_начинается\_строка

Вопрос 11 Что нужно сделать, чтобы обратиться к DOS для вывода строки на экран? Как DOS определит, где строка закончилась? Вывод на экран строки текста и выход из программы осуществляются путем вызова стандартных процедур DOS, называемых прерываниями. Прерывания под кодом 21h (33 – в десятичной системе счисления) называются функциями DOS, у них нет названий, а только идентификаторы. Номер прерывания и его параметры передаются в регистрах процессора, при этом номер должен находиться в регистре AH. Так, например, прерывание INT 21h, с помощью которого на экран выводится строка символов, управляется двумя параметрами: в регистре AH должно быть число 9, а в регистре DX – адрес строки символов, оканчивающейся знаком ‘$’: это спецсимвол языка ассемблера, которым обозначается нулевой байт (так называемый NUL-символ). Адрес строки Hello загружается в регистр DX с помощью оператора OFFSET (смещение): offset имя 11 Выход из программы осуществляется через функцию DOS с номером 4Ch. Эта функция предполагает, что в регистре AL находится код завершения программы, который она передаст DOS. Если программа завершилась успешно, код завершения должен быть равен 0, поэтому в примере загружаем регистры AH и AL с помощь одной команды MOV ax,4C00h , после чего вызываем прерывание 21h.

Вопрос 12 Программы, которые должны исполняться как .EXE и .COM, имеют существенные различия по: • размеру • сегментной структуре • механизму инициализации EXE-программы содержат несколько программных сегментов, включая сегмент кода, данных и стека. EXE-файл загружается, начиная с адреса PSP:0100h. В процессе загрузки считывается информация EXE-заголовка в начале файла, при помощи которого загрузчик выполняет настройку ссылок на сегменты в загруженном модуле, чтобы учесть тот факт, что программа была загружена в произвольно выбранный сегмент. После настройки ссылок управление передается загрузочному модулю к адресу CS:IP, извлеченному из заголовка EXE. COM-программы содержат единственный сегмент (или, во всяком случае, не содержат явных ссылок на другие сегменты). Образ COMфайла считывается с диска и помещается в память, начиная с PSP:0100h , в связи с этим COM -программа должна содержать в начале сегмента кода директиву позволяющую осуществить такую загрузку (ORG 100h). Они быстрее загружаются, ибо не требуется перемещения сегментов, и занимают меньше места на диске, поскольку EXE-заголовок и сегмент стека отсутствуют в загрузочном модуле.