Лабораторные работы №№ 1 – 6 по курсу
"Системное программное обеспечение"
(Разработка программ с помощью ассемблера TASM)

Цикл лабораторных работ содержит шесть занятий, проводимых последовательно по следующим темам:

Занятие 1. Интерфейс с пользователем
Занятие 2. Отладка программы, отладчик кодов
Занятие 3. Определение и обработка данных
Занятие 4. Обработка строк
Занятие 5. COM-файлы
Занятие 6. Процедуры и макросы

Каждый студент выполняет свое индивидуальное задание на текущую работу (в рамках общей темы занятия). Списки заданий выдаются преподавателем, ведущим занятие.

Примечание: Порядковый номер фамилии студента в журнале группы является номером его индивидуального задания. Если порядковый номер фамилии превышает длину списка заданий, то номер задания выбирается "по кругу", начиная с первого.

Имея индивидуальное задание, каждый студент создает программу (новую для каждого занятия) с заданным или произвольно выбранным сценарием ее работы. Обязательное условие выполнения работ #1-#6 - использование TASM, TLINK и упрощенных директив сегментации (использование стиля MASM на этих занятиях не допускается)

Критерием оценки предъявляемой к защите программы служит глубина (!) и разносторонность (!) исследования темы задания.

Примечание: Если в исходном тексте программе воспроизведен только хрестоматийный синтаксис некоторой конструкции (например, взят из справочника), то такая работа не может рассчитывать на высокую оценку. Ссылка на то, что "программа работает" в данном случае несостоятельна, поскольку успешно выполняется и "пустая" программа.

В предъявляемой программе, например, должны содержаться ответы на следующие вопросы (разумеется, следующий перечень условный, поскольку здесь не учитывается конкретное задание):

- ВАРИАНТЫ синтаксиса и форматов объявлений и определений
- ТОЧКИ (области) исходного текста, в которых допустимо или недопустимо использовать данную конструкцию
- ОГРАНИЧЕНИЯ на использование конструкции (как можно использовать ее и как - нельзя)
- Внутреннее ПРЕДСТАВЛЕНИЕ конструкции в памяти или механизм выполнения
- Побочные действия, производимые при применении исследуемой конструкции
- ВЛИЯНИЕ структуры программы или других объектов на работу исследуемой конструкции
- ОТЛИЧИЯ и особенности, которыми обладает данная конструкция по сравнению с другими, выполняющими эквивалентные действия. Например, вывести символ на экран мож-

но десятком разных способов. Каковы сравнительные характеристики того способа, который исследуется в работе?

- ВОЗМОЖНОСТИ, которые приобретает программист, используя данную конструкцию, и прочее.

Специальных требований к объему исходного текста не предъявляется, - это учебная программа и ее содержимое должно быть таким, чтобы в максимальной степени отражать ответы на перечисленные выше вопросы. Общий принцип создания программы - ее лаконичность. Рекомендуемый объем - не более 2 печатных страниц.

Следует избегать включения фрагментов, не поддерживающих исследуемые конструкции и не относящихся непосредственно к теме задания.

Рекомендуется использовать внешние отладчики только в том случае, когда внутренние средства языка не позволяют компактно закодировать вывод результатов работы исследуемой конструкции. Во всех прочих случаях следует стремиться к тому, чтобы сама программа отображала результаты программного эксперимента.

Отчетом по лабораторной работе служит исходный текст программы, продемонстрированный на экране монитора. "Бумажный" отчет не требуется. Исходный текст обязательно должен быть корректно оформлен и документирован:

- Заголовок (Студент, группа, дата, тема задания и номер варианта, версия компилятора и линкера, действия по запуску и завершению работы программы, параметры командной строки при вызове).
- Комментарий функциональных блоков и подпрограмм.
- Комментарий применяемых структур данных и используемых системных вызовов.
- Построчный комментарий, связанный с темой исследований.
- Функциональные блоки разделяются пустыми строками.
- Используется выравнивание столбцов меток, команд и операндов при помощи табуляции.

Тексты без комментариев и надлежащего форматирования не рассматриваются.

Результат выполнения каждой лабораторной работы оценивается преподавателем по пятибалльной шкале в процессе ее защиты. Оценка в общем зачете по курсу будет основываться на оценках всех лабораторных работ.

Перечисленные требования к среде разработки программ, а также к содержанию и оформлению работ являются обязательными.

ЗАНЯТИЕ 1

ТЕМА: Программирование интерфейса с пользователем в TASM (ввод с клавиатуры и вывод на экран).

ЗАДАНИЕ: Составить и отладить индивидуальную программу, реализующую взаимодействие с пользователем и содержащую вызов одной из указанных ниже функций DOS/ BIOS.

Варианты:

1. DOS Int 21h, func. 01h (ввод символа с эхо).
2. DOS Int 21h, func. 02h (вывод символа).
3. DOS Int 21h, func. 06h (ввод/вывод символа).
4. DOS Int 21h, func. 07h (ввод символа без эхо).
5. DOS Int 21h, func. 0Ah (буферизованный ввод строки).
6. DOS Int 21h, func. 08h (ввод символа без эхо).
7. DOS Int 21h, func. 0Bh (проверить состояние ввода).
8. BIOS Int 10h, func.2 и func 3. (позиция курсора).
9. BIOS Int 10h, func.1 (размер курсора).
10. BIOS Int 10h, func.6 или 7 (скроллинг экрана).
11. BIOS Int 10h, func.8 и 9 (чтение/запись символа).
12. BIOS Int 10h, func. 0Ah и 0Eh (запись символа).
13.* Прямой вывод символов в видеопамять.
14.* DOS Int 21h, func.40h (вывод строки символов на экран. Указание: Дескриптор устройства-Дисплей равен 1, т.е. BX=1).
15. BIOS Int 10h, func. 13h (вывод строки на экран. Проверить подфункции AL=1 и какую-либо еще).
16. Функции BIOS для посимвольного вывода строки на экран.
17. Вывод символов на экран (прерывания, которые отображают управляющие символы, и прерывания, которые не отображают их).

ЗАНЯТИЕ 2

ТЕМА: Отладка прикладной программы в отладчике кодов

Общее задание:

Дизассемблировать и трассировать индивидуальную EXE-программу. Программа должна содержать сегменты данных, кода и стека. Выяснить начальную установку регистров МП и размещение программы в памяти. Получить дамп памяти (области данных) и дизассемблированный код программы. После выполнения этих действий применить к программе названную в индивидуальном задании команду Отладчика в различных ее вариантах.

Разрешается использовать любой отладчик кодов: DEBUG, AFD, Turbo Debugger. Использование последнего предпочтительно, т.к. он позволяет, проделав определенные манипуляции, проводить отладку с просмотром ИСХОДНОГО ТЕКСТА программы и видеть все используемые в исходном тексте символические имена.

Общий вопрос при защите: РЕГИСТРОВАЯ МОДЕЛЬ ПРОЦЕССОРА, назначение регистров, их инициализация при загрузке программы.

Варианты индивидуальных заданий: исследовать возможности отладчика по выполнению следующих действий:

0*. (Вариант вместо любого из вариантов.) Написать на ЯВУ (С или Паскаль) короткую программу, содержащую вызов примитивной процедуры с локальными переменными и параметрами. Например, процедура может просто складывать два передаваемых ей значения. Изучить в отладчике функционирование кода, генерируемого компилятором ЯВУ, на уровне машинных команд.

1. Ввод литеральных значений в область памяти
2. Заполнение области памяти шаблоном
3. Перемещение (копирование) блоков памяти
4. Поиск в блоке памяти
5. Установка "точек останова" в программе
6. Загрузка дискового файла в отладчик
7.* Запись созданного в отладчике программного файла на диск
8. Загрузка блока в память из дискового файла
9. Запись блока памяти в дисковый файл
10. Сравнение областей памяти
11.* Создание программы в отладчике и запись на диск (в формате COM)
12.* Создание программы в отладчике в машинных кодах и запись на диск (в формате COM)

ЗАНЯТИЕ 3

ТЕМА: Определение и обработка данных

ЗАДАНИЕ: Составить и отладить индивидуальную программу, реализующую обработку числовых данных в соответствии с индивидуальным заданием. Предусмотреть возможность удобного изменения размеров массивов в исходном коде, максимально используя средства языка ассемблера (символьные имена, константы, метки, арифметические выражения). Использовать циклические конструкции. **Применить различные типы данных и способы адресации, объявление данных в различных сегментах, директивы EQU и =, выражения, содержащие символические имена.**

Численные результаты вывести на экран программными средствами, разработав для этого специальную подпрограмму перевода чисел в строковое представление. Допускается вывод в шестнадцатеричном коде. Отрицательные числа должны выводиться со знаком "минус".

Варианты индивидуальных заданий:

1. Задан массив двухбайтовых целых чисел со знаком. Рассчитать среднее значение элементов массива.

2. Задан массив двухбайтовых целых чисел без знака. Рассчитать среднее значение элементов массива.

3. Задан массив однобайтовых целых чисел со знаком. Рассчитать среднее значение элементов массива.

4. Задан массив однобайтовых целых чисел без знака. Рассчитать среднее значение элементов массива.

5. Задан двумерный массив двухбайтовых целых чисел со знаком. Определить минимальное значение элементов массива и его индексы.

6. Задан двумерный массив двухбайтовых целых чисел без знака. Определить максимальное значение элементов массива и его индексы.

7. Задан двумерный массив однобайтовых целых чисел со знаком. Определить минимальное значение элементов массива и его индексы.

8. Задан двумерный массив однобайтовых целых чисел без знака. Определить максимальное и значение элементов массива и его индексы.

9. Задан массив четырехбайтовых чисел, содержащий нулевые элементы. Определить максимальную длину последовательности ненулевых элементов массива.

10. Задан массив четырехбайтовых чисел, содержащий нулевые элементы. Определить минимальную длину последовательности ненулевых элементов массива.

11. Задан массив дальних указателей. Подсчитать количество элементов массива, указывающих на элементы, расположенные по четным адресам.

12. Задан массив дальних указателей. Подсчитать количество элементов массива, указывающих на элементы, расположенные в исполняющемся сегменте кода программы.

13. Задан массив указателей на однобайтовые числа. Подсчитать количество указателей на ненулевые значения.

14. Задан массив указателей на двухбайтовые числа со знаком. Заменить все указатели на числа меньше 10 на нулевые указатели.

15. Задан массив указателей на однобайтовые числа со знаком. Заменить все указатели на числа меньше -15 на нулевые указатели.

16. Задан массив однобайтовых чисел со знаком. Заменить все числа в массиве, кроме первого, на разность между текущим числом и предыдущим.

17. Задан массив двухбайтовых чисел со знаком. Заменить все числа в массиве, кроме последнего, на среднее значение текущего числа и последующего.

18. Задан массив двухбайтовых чисел без знака. Определить минимальную разность между соседними элементами массива.

19. Задан массив однобайтовых чисел без знака. Определить максимальную разность между соседними элементами массива.

20. Определить расстояние (количество элементов массива) между минимальным и максимальным элементами массива двухбайтовых чисел со знаком.

ЗАНЯТИЕ 4

ТЕМА: Обработка строк

ЗАДАНИЕ: Составить и отладить индивидуальную программу, реализующую обработку строк в соответствии с индивидуальным заданием. Применить различные типы данных,

объявление их в различных сегментах, различные виды меток (переменные, константы, собственно метки) и выражения, содержащие символические имена. При написании программы максимально использовать возможности системы команд процессора по обработке строковых данных - строковые команды, префиксы условного и безусловного повторения. Продемонстрировать результаты в отладчике.

При формулировке задания использованы термины:

Z-string - строка, завершающаяся нулем. Признак конца строки - символ с кодом 0. Длина строки нигде специально не хранится. После нуля в буфере, выделенном под строку, может располагаться "мусор", который игнорируется при обработке.

Pascal-string - строка в стиле языка Pascal. Массив байтов, нулевой байт содержит длину строки, остальные байты - символы строки. Маркер конца строки не предусмотрен. В конце буфера, выделенного под строку, может располагаться "мусор", который игнорируется при обработке.

Варианты индивидуальных заданий:

1. Удалить все пробелы из строки, Z-string
2. Удалить все ведущие пробелы (в начале строки), Z-string
3. Удалить все концевые пробелы, Z-string
4. Удалить все пробелы из строки, Pascal-string
5. Удалить все ведущие пробелы (в начале строки), Pascal-string
6. Удалить все концевые пробелы, Pascal-string
7. Написать программу выделения из исходной строки подстроки символов заданной длины с указанного номера позиции, Z-string
8. Написать программу, определяющую номер позиции, с которой начинается первое слева вхождение заданной подстроки в исходную строку, Z-string
9. Написать программу формирования строки из исходной путем заданного числа повторений исходной строки, Z-string
10. Написать программу выделения из исходной строки подстроки символов заданной длины с указанного номера позиции, Pascal-string
11. Написать программу, определяющую номер позиции, с которой начинается первое слева вхождение заданной подстроки в исходную строку, Pascal-string
12. Написать программу формирования строки из исходной путем заданного числа повторений исходной строки, Pascal-string
13. Написать программу, которая бы инвертировала (зеркально переворачивала) исходную строку, Z-string
14. Написать программу, заменяющую все десятичные цифры в исходной строке на заданный символ, Z-string
15. Написать программу, удаляющую из исходной строки многократные вхождения заданного символа, заменяя их однократными, Z-string
16. Написать программу, которая бы инвертировала (зеркально переворачивала) исходную строку, Pascal-string
17. Написать программу, заменяющую все десятичные цифры в исходной строке на заданный символ, Pascal-string

18. Написать программу, удаляющую из исходной строки многократные вхождения заданного символа, заменяя их однократными, Pascal-string

ЗАНЯТИЕ 5

ТЕМА: COM - файлы

ЗАДАНИЕ: Разработать индивидуальную программу для .COM-файла. Допускается взять в качестве прототипа свою программу Лаб. #1 (Ввод-Вывод в TASM). В программе иметь секции Данных и явно работать со Стеком. Использовать Отладчик для выявления начальной установки регистров и определения типов адресов.

Варианты индивидуальных заданий:

1. Создать "собственный" Стек в области кодов программы.
2. Использование директив определения данных (DB, DW, DD,..) для записи инструкций (кодов) программы.
3. Примеры использования директивы ORG .
4. Данные, размещенные в конце текста программы.
5. Варианты Завершения исполнения .COM-программы.
6. Отличие .COM-программы от аналогичной .EXE-программы (в .EXE имеется только один сегмент - .CODE).
7. Использование директивы LABEL в области кодов .COM-программы.
8. Использование директивы LABEL в области данных .COM-программы.
9. Различные виды меток в .COM-программе.
10. Анализ .LST-файла для .COM-программы
11. Данные, размещенные в начале текста .COM-программы.
12. Использование директивы OFFSET и инструкции LEA в .COM-программе.
13. Использование инструкций LEA и LES в .COM-программе.

ЗАНЯТИЕ 6

ТЕМА: Процедуры в TASM.

ЗАДАНИЕ: Составить и отладить программу, содержащую процедуры с передаваемыми и возвращаемыми параметрами, оформленные с помощью директивы PROC. Процедуры должны многократно вызываться. В качестве прототипа взять работу 3 или 4, оформив все действия в виде нескольких процедур. Программа должна выводить результаты работы на экран. **Обязательно продемонстрировать передачу параметров через стек.** Дополнительно исследовать особенности применения процедур в соответствии с вариантом индивидуального задания. **Обязательно оформить какие-либо действия как макросы с параметрами.**

Варианты индивидуальных заданий:

1. Локальные метки в процедурах и ограничение их области действия (NoLocals и обычной меткой).
2. Вложенные процедуры.
3. Передача параметров через стек (передать строку и прочие параметры, варианты синтаксиса).
4. Локальные данные в процедурах.
5. Директива "PROC": а) как часть заголовка; б) как тип метки.
6. Вызов процедуры с именем регистра (например, Call CX).
7. Варианты форматов объявления процедур.
8. Вызов процедуры, использующий базовый регистр (например, Call WORD PTR [BX]).
9. Использование инструкций возврата (Ret, RetF, RetN, Ret число)
10. Передача параметров через регистры (передать строку и прочие параметры).
11. Передача параметров через фиксированную область памяти (например, передать строку и прочие параметры).
12. Содержимое стека до/после вызова процедуры (Near/Far-вызовы). Использовать отладчик кодов.
13. Принудительная установка NEAR/FAR типа вызова процедур.
14. Авторекурсия (вызов процедурой "самой себя").
15. Несколько точек входа в процедуру (вызов процедуры "с середины").
16. Взаиморекурсия ("перекрестный" вызов одной процедуры из другой).
17. Процедура, обьявленная с директивой PASCAL. (И директивы ARG и USES).