Методические указания к лабораторным работам № 3-8 по дисциплине “Системное Программирование”

"Цикл ЛР: Программирование на языке Ассемблер "

Москва, МГТУ - 2019 год

СОДЕРЖАНИЕ

1. 1. Цели и задачи ЛР по Ассемблеру и общие принципы их выполнения

Комплекс лабораторных работ (3-9 ЛР) по языку Ассемблер выполняется студентами для освоения языка программирования, получения навыков разработки и отладки программ на нем, изучения и использования компонентов системы программирования Ассемблер (компилятора, редактора связей, отладчика) и получения навыков оформления документации по программным разработкам, реализуемым на языке Ассемблера.

Все лабораторные работы по курсу СП взаимосвязаны друг с другом, поэтому их необходимо выполнять последовательно, начиная с 3-й ЛР СП (первые две ЛР курса посвящены другим темам). Это позволяет значительно упростить задачу выполнения всего цикла лабораторных работ. При этом в новую работу могут быть успешно включены проработки и отлаженные фрагменты из предыдущей работы (процедуры и циклы). Кроме того, самостоятельное выполнение цикла лабораторных работ позволит студентам успешно справиться с заданием на курсовую работу, которая выполняется также в данном семестре (4-й семестр). В курсовой работе студенты на языке Ассемблера разрабатывают резидентную программу.

2. 2. Общие принципы выполнения ЛР по СП и оборудование

При выполнении лабораторных работ студенты могут их сделать с учетом основных требований и с учетом дополнительных требований. Выполнение дополнительных требований фиксируется в журнале приема ЛР и отражается в дифференцированной оценке по дисциплине на зачете и защите КР.

Студенты выполняют работы с использованием систем программирования на языке Ассемблера, предоставленных преподавателем (все программные системы можно скачать с кафедрального сайта или переписать с сайта преподавателя – **www.sergebolshakov.ru**). Можно использовать разные версии систем программирования (СП): TASM (v 2,3,4,5), MASM и QC25. Использование TASM, на мой взгляд, сейчас является предпочтительным. Кроме того можно использовать интерактивную оболочку QC25 (она есть на сайте, а ее использование описано в методическом пособии для ЛР по курсу – см. на сайте[7]). Она удобна, так как это интегрированная оболочка, включает одновременно следующие компоненты: текстовый редактор, компилятор, справочную подсистему, редактор связей и встроенный отладчик. Это в свою очередь обеспечивает более оперативную и эффективную работу студента по разработке программ и освоению дисциплины. Наиболее прозрачной и мобильной для применения все же будет система TASM3. Поэтому, для выполнения ЛР по ассемблеру на сайте представлен отдельный архив (TASM3.ZIP), в который включены все необходимые компоненты для программирования в среде TASM3 (см. на сайте) и русификации командной строки или DosBox:

1. Компилятор с языка Ассемблер - **tasm.exe**.
2. Компоновщик для Ассемблера - **tlink.exe**.
3. Отладчик для Ассемблера - **td.exe**.
4. Текстовый редактор для Ассемблера - **Asm_ed.exe** (с ним можно перекодировать текст).
5. Русификатор клавиатуры и экрана- **RKM.COM**.
6. Утилита поиска в текстах для системного программиста - **GREP.com**.
7. Программа перекодировки DOS-WINDOWS - **trans.exe** (перекодировщик текстов).
8. Справки для основных программ - ***.HLP** (Они получены после перенаправления >>).
9. Пример простейшей программы (**FirstD.ASM/First.ASM**) для DOS и WINDOWS - **,** рассмотренной в данных методических указаниях.
10. Другие примеры программ на Ассемблере для продвинутых студентов - *.arj.

Особенности технологии работы в системах программирования для языка Ассемблера подробно изложены в общих методических указаниях[7] и другой литературе. Много информации и справочных данных Вы найдете в книгах, размещенных на сайте. И в отдельных методических указаниях по выполнению лабораторных работ. Для успешного выполнения ЛР3-8 СП по программированию на языке Ассемблер целесообразно: познакомиться, изучить и освоить следующие разделы общего методического пособия: 1, 2, 3, 7, 10, 15, 17 [7].

Ежегодно требования и варианты ЛР уточняются и видоизменяются. Поэтому в заголовке названия каждой ЛР проставляется год (например, год **2016**). Если этот год соответствует текущему году, то данные требования к выполнению этой ЛР актуальны, и их можно применить для работы. В ближайшее время доработаю задания по всем ЛР. Изменять требования в ЛР приходиться по вине тех студентов, которые привыкли списывать чужие результаты. Будьте **внимательны и используйте актуальные требования МУ.**

3. 3.Основы работы на Ассемблере

Как было отмечено выше, для построения программ на языке Ассемблера, нужно иметь подходящую систему программирования (СП). Эта система включает минимальный набор программ, позволяющих достичь цели. Ниже, мы очень кратко, покажем, как можно воспользоваться СП TASM3 (все компоненты уже включены в архив, рассмотренный выше и размещенный на сайте). В равной степени можно использовать и другие СП для Ассемблера, но здесь мы предложим воспользоваться соответствующей технической документацией на конкретную программную систему.

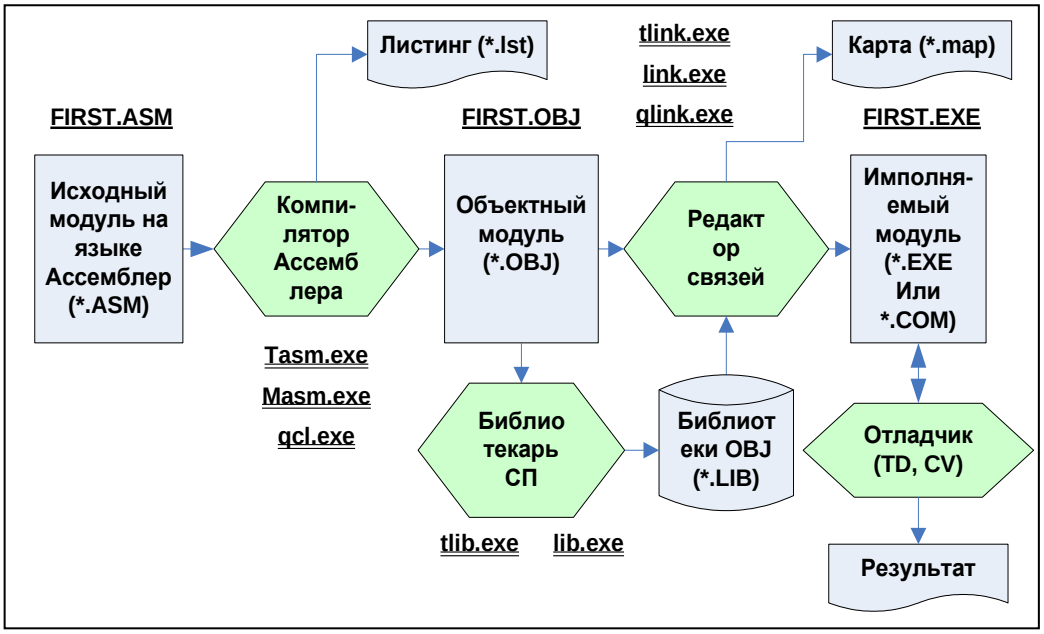
3.1. 3.1 Среда разработки программ

Для работы в системе программирования ЯА (Языка Ассемблер) необходимо сначала уточнить в какой среде и на каком компьютере предстоит работать. В принципе для правильной

работы подойдет операционная система (ОС) MS DOS или любая другая, позволяющая эту ОС эмулировать. Для режима эмуляции нужно уточнить характеристики процессора (32 или 64 бит). В первом случае меньше проблем, так как программы ДОС с успехом могут работать в окне командной строки (**CMD.EXE** или COMMAND.COM). Эти эмуляторы обеспечивают практически полную совместимость с режимами функционирования MS DOS. Исключение, пожалуй, составляет полноэкранный режим работы программ. Если такие режимы невозможны, то другой возможный способ – использование универсального эмулятора DOS (система DOSBox) разных версий. Недостатком использования **DOSBox** является выборочная поддержка команд операционной системы. Третий, и самый универсальный способ воспроизведения MS DOS – это развертывание эмулятора виртуальных машин (например, **Oracle VM VirtualBox**), который позволяет установить любую ОС и ее использовать. В нашем случае мы для простоты будем рассчитывать на первый и второй случай: окно командной строки и эмулятор **DOSBox**. Для примеров работы в командной строке в нашем случае используется ОС Windows XP 3-й редакции.

3.2. 3.2 Технология обработки программ

Для обработки программ в современных системах программирования и, в частности на языке Ассемблера, характерна схема, представленная на рисунке расположенном ниже.



Данная технология формирования программ характерна практически для всех систем программирования, используемых в настоящее время. В тех случаях, когда мы работаем в интегрированной оболочке СП (QC25, FASM и др.), мы можем и не увидеть в явном виде промежуточных компонент, участвующих в такой обработке, однако они всегда присутствуют. В других случаях отдельные фазы обработки программ выполняются отдельно (TASM , MFAS). Эти

компоненты отмечены на рисунке двойным подчеркиванием. Рассмотрим разновидности модулей и программ, представленных на рисунке выше.

3.3. 3.3 Системы программирования для Ассемблера

Для наших лабораторных работ желательно использовать одну из трех систем программирования (они все есть на сайте дисциплины):

- **Turbo Assembler** (**TASM**) разных версий. Он включает: **tasm.exe, tlink.exe, tlib.exe, td.exe** и много других вспомогательных программ.
- **Microsoft Assembler** (**MASM**) разных версий. Он включает: **masm.exe, link.exe, lib.exe, cv.exe** и много других вспомогательных программ.
- **Quick C and Quick Assembler** (**QC25**) - Он включает: **qcl.exe, qlink.exe, lib.exe** и много других вспомогательных программ. QC является интегрированной оболочкой, позволяющей выполнять все операции создания программ без переключения в командную строку. В QC встроен также текстовый редактор для подготовки исходных текстов программ.

Другие разнообразные системы программирования на языке Ассемблера (FASM, NASM, RADASM, SASM, EMU8086 и др.) имеют отдельные недостатки для методики преподавания и в данном курсе их использование не рекомендуется. Система программирования **TASM** (Turbo Assembler) – разных версий (3-5) в нашем курсе, в настоящий момент, является предпочтительной. Компоненты разных СП для Ассемблера вы найдете на сайте дисциплины в разделе “Лабораторные работы”.

3.4. 3.4 Модули представления программ

Существующая технология создания исполнимых программ включает в себя следующие разновидности модулей:

- Исходные модули (ИМ), написанные на языке программирования (например, языке Ассемблер). Исходные модули, обычно, представляют собой длинную строку символов (или совокупность строк) и имеют следующее расширение файлов: ***.ASM, *.INC.** Исходные модули формируются программистом или подключаются из библиотек (заголовочные файлы – ***.INC**). Исходные модули кодируются в форматах: ASCII (ДОС) или ANSI (Windows)/
- Объектные модули (ОМ) формируются компиляторами после успешной компиляции (без ошибок) исходных модулей. Объектные модули имеют расширение *.OBJ и имеют стандартизованную структуру. Изменять содержание объектных модулей вручную не рекомендуется.
- Исполнимые модули (ИСМ) формируются редактором связей (компоновщиком), который объединяет множество объектных модулей в единую программу. Исполнимые модули могут непосредственно выполняться на компьютере. Они имеют формат: *.EXE или *.COM.
- Библиотеки объектных модулей объединяют множества объектных моделей, связанных по смыслу и функциям. Библиотеки бывают стандартными и

пользовательскими. Для построения библиотек используется специальная компонента СП – библиотекарь (LIB.EXE или TLIB.EXE).

3.5. 3.5 Компоненты СП и стадии обработки программ

Система программирования (СП) это большой комплекс программ, включающий в себя следующие основные программы:

- Компилятор (TASM или MASM) системы программирования, который проверяет правильность написания программ (исходных модулей) и формирует объектные модули. Примеры - **tasm.exe** и **masm.exe.**
- Редактор связей(TLINK иди LINK) (или компоновщик) необходим для объединения множества объектных модулей в единую программу (исполнимый модуль). Редактор связей настраивает связи между отдельными модулями, которые могут быть двух видов: связи по управлению (вызов функций и процедур) и связи по данным (использование данных из одного модуля в другом). Примеры: **tlink.exe** и **link.exe.**
- Библиотекарь (TLIB или LIB) – программа СП (иногда называют такие программы утилита), которая позволяет создавать библиотеки объектных модулей (ОМ). Библиотеки объектных модулей подключают в программные проекты и, тем самым, обеспечивают подключение нужных объектных модулей в исполнимый модуль.
- Менеджер проектов (MAKE), позволяющий создавать проекты, выполнять сборку из многомодульных программ и проводить их отладку. Чаще всего его работа невидима, эта компонента настраивается на специальный файл проекта, который оформляется на специальном языке.
- Отладчик СП (TD или CV), который обеспечивает возможности эффективной проверки программ и исправления ошибок в программах.
- Другие сервисные утилиты предназначены для упрощения процесса программирования и обслуживания создаваемых проектов: текстовые редакторы, справочные системы, примеры использования разных технологий и т.д.

3.6. 3.8 Фазы подготовки и создания программ на Ассемблере

Таким образом, для построения исполнимой программы необходимо пройти следующие основные фазы:

- Подготовка алгоритма, кодирование и создание исходных модулей проекта (***.ASM**).
- Компиляция и синтаксическая отладка исходных модулей, формирование объектных модулей проекта (***.OBJ**).
- Редактирование связей для всех объектных модулей проекта, включая и модули из объектных библиотек, и формирование исполнимого модуля проекта (***.EXE** или ***.COM**).
- Отладка исполнимого модуля (***.EXE** или ***.COM**) с помощью отладчика, если эти модули специально подготовлены для отладки на этапах компиляции и редактирования связей.

– Если программа не дает нужный (планируемый/прогнозируемый) результат, то следует “многократное” повторение всех предыдущих этапов (начиная, с любого предшествующего), пока правильный результат не будет получен.

Рассмотрим эти фазы на простейшем примере программы для языка Ассемблер в система программирования TASM.

3.7. 3.7 Пример простой программы

В этом разделе мы рассмотрим простейшую (возможно, первую для Вас) программу на языке Ассемблер и все необходимые действия в режиме командной строки, которые нужно выполнить для получения исполнимой программы.

Эта программа: настраивает сегменты, выводит на экран один символ (буква “**A**”) и корректно завершает работу. Программу будем компилировать в режиме командной строки с использованием системы программирования **TASM** (Turbo Assembles – СП можно найти на сайте). Нужно выбрать среду для работы: **CDM.EXE** или **DOSBox.** Эту среду нужно корректно запустить. Для получения готовой программы нужно выполнить следующие действия:

– Подготовить исходный текст программы, для этого необходимо использовать любой текстовый редактор, но ввод символов необходимо выполнять в нужном формате ANSI (Windows) или ASCII (DOS). Рекомендую использовать программу **ASM_ED.EXE** – есть на сайте. Можно использовать также текстовые редакторы любого файл менеджера, и даже NOTEPAD (Только в однобайтовой кодировке ,а не в **UNICOD**!).
– Выполнить компиляцию программы с помощью компилятора **TASM.EXE** и получить листинг программы.
– Выполнить редактирование связей с помощью редактора связей **TLINK.EXE**.
– Выполнить программу в режиме командной строки для проверки (**CDM.EXE** или **DOSBox**).
– Проверить также выполнение программы с помощью отладчика **TD.EXE**.

3.8. 3.8 Подготовка исходного текста программы

Введем текст программы, расположенный ниже, и сохраним файл с названием “**FIRST.ASM**” в каталоге, где расположена система программирования TASM (развернут наш архив – TASM3.ZIP). Для редактирования текста можно использовать любой текстовый редактор (для ЛР СП рекомендуется ASM_ED.EXE – есть на сайте и в архиве TASM3.ZIP). Данный файл называется также исходным текстом программы, исходным модулем или ее “распечаткой”. Ввод текста можно вводить в любом текстовом редакторе, даже в notepad, но имейте в виду, что ввод для Ассемблера должен осуществляться однобайтовыми символами и должен включать служебные символы. Поэтому ввод текста в MS WORD и других мощных текстовых процессорах недопустим. Обратите внимание, что в исходном тексте программы форматирование текста (пробелы перед строками) выполняет сам программист. Наша программа на языке Ассемблер выглядит так:

```
MYCODE SEGMENT 'CODE'
    ASSUME CS:MYCODE,DS:MYCODE
```

```
LET  DB 'A'
START:
; Загрузка сегментного регистра данных DS
     PUSH CS
     POP  DS
; Вывод одного символа на экран
     MOV AH, 02
     MOV DL, LET
     INT 21H
; Ожидание завершения программы
      MOV AH, 01H
      INT 021H
; Выход из программы
     MOV AL, 0
     MOV AH, 4CH
     INT 21H
MYCODE ENDS
END START
```

В нашем примере СП TASM располагается по адресу (в каталоге): C:\BORLANDC\TASM. В этот каталог запомним нашу программу - "**FIRST.ASM**"

3.9. 3.9 Компиляция программы

Далее нужно перейти в режим командной строки (CMD или DOSBox) и запустить программу на компиляцию:

```
C:\BORLANDC\TASM>tasm.exe /l /zi /c first.asm↵
Turbo Assembler  Version 3.1  Copyright (c) 1988, 1992 Borland International

Assembling file:   first.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  414k
```

Примечание. Символ "↵" не будет появляться на экране. Более подробно параметры запуска компилятора TASM.EXE мы рассмотрим ниже.

Результатом правильной работы компилятора будет формирование в текущем каталоге объектного модуля с именем файла "**FIRST.OBJ**". У нас это каталог - **C:\BORLANDC\TASM**.

Проверить наличие объектного модуля можно с помощью команды **DIR**, не выходя из режима командной строки. Однако удобнее использовать для этих целей файл менеджер (например, **VC.COM**). Например, проверим наличие:

```
C:\BORLANDC\TASM>dir first.obj↵
 Том в устройстве C имеет метку SYSTEM
 Серийный номер тома: D08B-21A4

 Содержимое папки C:\BORLANDC\TASM
```

```
24.02.2009  13:23               354 FIRST.OBJ
```

Если компиляция завершена с ошибками, то объектный модуль не формируется и нужно вернуться к этапу синтаксической отладки исходного модуля (см. ниже). После успешной компиляции можно приступить к следующему этапу редактированию связей и созданию исполнимого модуля.

3.10. 3.10 Редактирование связей в программы

Операция редактирования связей важна для многомодульных программ. Но даже для программ состоящих из одного модуля эту фазу обработки опустить нельзя. Нужно также учитывать, что на этой стадии формируется исполнимый модуль программы в виде ***.EXE** или ***.COM** файлов. Для запуска редактирования связей нужно запустить утилиту TLINK следующим образом:

```
C:\BORLANDC\TASM>tlink.exe /v /l /m first.obj↵
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International
Warning: No stack
```

Примечание. Символ "↵" не будет появляться на экране. Более подробно параметры запуска редактора связей TLINK.EXE мы рассмотрим ниже. Предупреждение "**Warning: No stack** " выдается по причине отсутствие явного описания сегмента стека в программе на Ассемблере.

При правильном завершении работы редактора связей исполнимый модуль типа ***.EXE** будет построен, а его формирование можно также проверить командой **DIR**. Например, см. ниже:

```
C:\BORLANDC\TASM>dir first.exe↵
 Том в устройстве C имеет метку SYSTEM
 Серийный номер тома: D08B-21A4

 Содержимое папки C:\BORLANDC\TASM

24.02.2015  14:20               990 FIRST.EXE
```

Если модуль (**FIRST.EXE**) с новой датой (24.02.2015) и временем (14:20) существует, то фаза обработки редактором связей прошла без ошибок и исполнимый модуль готов к запуску.

3.11. 3.11 Ошибки компиляции, редактирования и листинг программы.

Если при компиляции обнаружена ошибка, объектный модуль не формируется и на консоль (окно командной строки) будет выведено сообщение об ошибке. Пусть при компиляции обнаружены ошибки, например, Вы ошиблись в названии команды Ассемблера (я сознательно заменил в программе команду PUSH на команду **PASH** – такой нет в перечне команд), то в поток командной строки будет выведено сообщение с номером строки текста, в которой была обнаружена ошибка и ее причина на английском языке (см. ниже – выделено жирным):

```
C:\BORLANDC\TASM>tasm /l /zi first.asm↵
Turbo Assembler  Version 3.1  Copyright (c) 1988, 1992 Borland International

Assembling file:   first.asm
```

```
**Error** first.asm(6) Illegal instruction
Error messages:    1
Warning messages:  None
Passes:            1
Remaining memory:  414k
```

Текст сообщения об ошибке - “**Illegal instruction**” означает “**недопустимая команда**”. Если при запуске компилятора задан параметр “**/z**”, то будет распечатана та строка, в которой была обнаружена ошибка (см. ниже).

```
C:\BORLANDC\TASM>tasm /z /zi /l first.asm↵
Turbo Assembler  Version 3.1  Copyright (c) 1988, 1992 Borland International

Assembling file:   first.asm
     PASH CS
**Error** first.asm(6) Illegal instruction
Error messages:    1
Warning messages:  None
Passes:            1
Remaining memory:  414k
```

Ошибку нужно исправить в текстовом редакторе и повторить компиляцию. Если ошибок больше нет, то можно приступать с фазе редактирования вязей.

Параметр запуска компилятора “**/l**”, заставляет ассемблер сформировать специальный документ – листинг программы (“распечатку” обработанного текста программы в файл). Если не указано другого имени, листинг формируется с названием файла исходного текста программы и расширением - ***.lst** (см. общий рисунок выше). Листинг форматируется компилятором Ассемблера и содержит много полезной информации для программиста (пример листинга смотри ниже). Листинг, полученный в нашем случае (**FIRST.LST**), будет иметь вид:

```
Turbo Assembler    Version 3.1          24/02/09 12:46:17       Page 1
first.asm

1     0000                    MYCODE SEGMENT 'CODE'
2                              ASSUME      CS:MYCODE, DS:MYCODE
3     0000  41                LET  DB 'A'
4     0001                    START:
5                  ; Загрузка   сегментного регистра данных DS
6     0001  0E                 PUSH CS
7     0002  1F                 POP  DS
8                              ; Вывод одного символа на экран
9     0003  B4 02         MOV AH, 02
10    0005  2E: 8A 16   0000r          MOV DL, LET
11    000A  CD 21         INT 21H
12                               ; Выход из   программы
13    000C  B0 00          MOV AL, 0
14    000E  B4 4C         MOV AH, 4CH
15    0010  CD 21         INT 21H
```

```
16    0012                    MYCODE ENDS
17                            END START

Turbo Assembler    Version 3.1          24/02/09 12:46:17       Page 2
      Symbol Table

      Symbol Name               Type       Value

      ??DATE                          Text       "24/02/09"
      ??FILENAME                Text       "first       "
      ??TIME                          Text       "12:46:17"
      ??VERSION                 Number 030A
      @CPU                      Text       0101H
      @CURSEG                         Text       MYCODE
      @FILENAME                 Text       FIRST
      @WORDSIZE                 Text       2
      LET                         Byte   MYCODE:0000           #3  10
      START             Near    MYCODE:0001               #4  17
      Groups & Segments         Bit Size Align  Combine Class

      MYCODE                          16  0012 Para     none        CODE
```

Если обнаружены синтаксические ошибки, то они отображаются в листинге программы сразу после оператора с ошибкой. Пример:

```
         6                          PASH CS
**Error** first.asm(6) Illegal instruction
```

Если при редактировании возникают ошибки, то их содержание и место появляется на дисплее.

```
C:\BORLANDC\TASM>tlink /v /l /m first↵
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International
Error: Undefined symbol VAR in module FIRST.ASM
Warning: No stack
```

В этом случае команда обращается к внешней переменной (**VAR** из другого модуля), но при редактировании связей ссылка на нее не была найдена.

При редактировании связей мы можем получить специальный файл (***.map**- см. рисунок выше) содержащий информацию о редактировании и возможные ошибки. Такой файл создается при задании параметра при запуске компоновщика “**/m**”. такой файл называется картой редактирования (map) и играет большую роль при создании многомодульных программ и их отладки. Ниже приведен пример такого файла с включенной ошибкой.

```
      Start  Stop   Length Name               Class
       00000H 00016H 00017H MYCODE             CODE
        Address         Publics by Name
        Address         Publics by Value

      Line numbers for first.obj(FIRST.ASM) segment MYCODE
```

```
     7 0000:0001     8 0000:0002    10 0000:0003    11 0000:0005
    12 0000:000A    14 0000:000C    15 0000:0011    16 0000:0013
    17 0000:0015
Error: Undefined symbol VAR in module FIRST.ASM
Program entry point at 0000:0001
Warning: No stack
```

При правильном завершении редактирования связей ошибки должны отсутствовать:

```
Start  Stop   Length Name               Class
 00000H 00016H 00017H MYCODE             CODE
  Address         Publics by Name
  Address         Publics by Value

Line numbers for first.obj(FIRST.ASM) segment MYCODE
     7 0000:0001     8 0000:0002    10 0000:0003    11 0000:0005
    12 0000:000A    14 0000:000C    15 0000:0011    16 0000:0013
    17 0000:0015
Program entry point at 0000:0001
Warning: No stack
```

3.12. 3.12 Выполнение программы

Если этапы компиляции и редактирования выполнены без ошибок, то можно выполнить запуск программы на выполнение. Для этого нужно ее запустить в окне командной строки:

```
C:\BORLANDC\TASM>first.exe↵
A
```

Если программа выводит информацию на экран и работает в пакетном режиме (т.е. выполняется и сама завершается, как в нашем примере), то можно перенаправить вывод на дисплей в текстовый файл и затем его просмотреть или распечатать. В нашем примере это может быть сделано так:

```
C:\BORLANDC\TASM>first >> first.txt↵
```

Сформированный файл “**first.txt**” сейчас содержит одну строку, в начальной позиции которой будет напечатана буква “**A**”.

3.13. 3.13 Отладка программы, использование отладчика

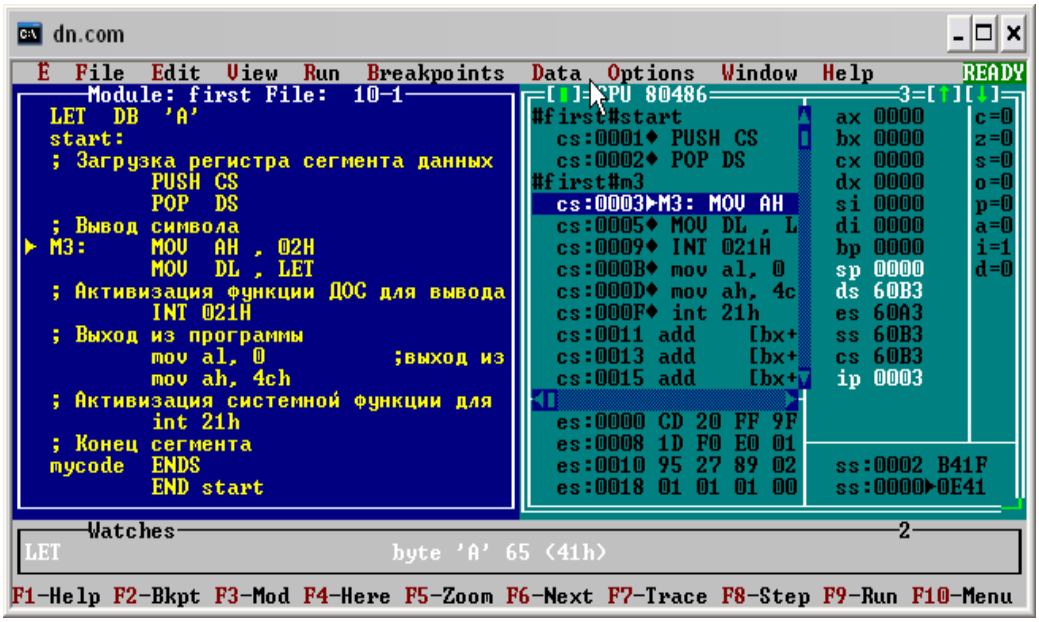
Для отладки программ в системе программирования **TASM** используется утилита TD.EXE. Для ее полнофункциональной работы на этапах компиляции и редактирования связей мы установили специальные режимы (см. выше):

- TASM.EXE – режим **“/zi”** – включение символьной отладочной информации в объектный модуль.
- TLINK.EXE – режим **“/v”** - включение символьной отладочной информации в исполнимый модуль.

В этом случае мы можем запустить TD и иметь в отдельном окне исходный текст нашей программы. Запуск TD выполняется так:

```
C:\BORLANDC\TASM>TD.EXE first.exe↵
```

После запуска в окне командной строки появиться окно отладчика. Настройка экрана отладчика может быть различной. В нашем примере представлено 2 окна: окно исходного текста (треугольник "►" – задает текущий исполняемый оператор) и окно информации о состоянии CPU (Control Program Unit – содержимое микропроцессора). Нажимая клавишу F7, мы можем выполнить по отдельным командам (по шагам) нашу программу. Результат работы программы мы можем увидеть, нажав комбинацию **Alt+F5.** Более подробно об отладке и отладчике мы будем говорить ниже. Для завершения отладчика нужно нажать "**Alt + x**". Подсказка снизу, определяет возможные действия, в том числе при нажатии управляющих клавиш (Ctrl и Alt, кстати, подсказка будет изменяться). В пунктах главного меню отладчика есть пояснения для их назначения.



Примечание. При демонстрации работы ЛР окно исходного текста должно быть актуальным и русифицированным. Без выполнения этих требований отработка работы **не будет зачтена**.

4. 4. Содержание ЛР и общие требования (ЛР 3-8)

Лабораторные работы выполняются в дисплейном зале под руководством преподавателей. Для успешного выполнения каждой лабораторной работы и получения зачета ее отработки студенты обязаны:

1. Предварительно подготовиться к выполнению данной работы: прочитать методические указания к ЛР, задание и освоить необходимые разделы общего методического пособия.
2. Выполнить задание на ЛР, продемонстрировав преподавателю правильный результат работы программы в отладчике. При выполнении задания преподаватели обязаны консультировать студентов по любым вопросам, которые связанны с данной лабораторной работой.

3. Оформить и защитить отчет по ЛР: подготовить ответы на контрольные вопросы и защитить работу. Допускается оформление отчета и ее защита во время самостоятельной подготовки. Защита должна быть выполнена не позже следующего занятия в часы лабораторных занятий по расписанию группы (не позже двух недель по срокам).

5. 5.Общий порядок работы.

Студенты разрабатывают работоспособную программу на языке Ассемблер по заданию ЛР, выполняют следующие действия (порядок выполнения работы):

- Знакомятся и осмысливают задание на ЛР.
- Разрабатывают алгоритм реализации задачи (блок-схема программы можно оформить в MS VISIO, MS WORD или на листе бумаги).
- Выполняют написание текста программы на языке Ассемблер и вводят его в отдельном текстовом редакторе (можно использовать текстовый редактор **ASM_ED.EXE** – есть на сайте) или в интегрированной оболочке (например, в QC).
- Выполняют отладку программы в отладчике (**TD.EXE**), демонстрируют преподавателю умение работать в отладчике, выполняя различные действия (выполнение по шагам, просмотр данных и т.д.).
- Формируют исполнимый модуль программы заданного типа (**COM** или **EXE** см. задание).
- Демонстрируют преподавателю работоспособную программу.
- По требованию преподавателя, если нужно, вносят изменения в программу и демонстрируют знание действий необходимых для создания исполнимого модуля (это предварительная сдача ЛР).
- Оформляют отчет по данной лабораторной работе в соответствии с требованиями приведенными ниже и на основе шаблона отчетов по ЛР.
- На основе отчета по ЛР (распечатанного) выполняют защиту ЛР у преподавателя (ответы на контрольные вопросы), после чего в журнале отмечается: срок сдачи ЛР, срок защиты ЛР, оценка за защиту данной ЛР и выполнение дополнительных требований к ЛР. На защите задаются вопросы, перечисленные в разделе “**Контрольные вопросы по каждой ЛР и общие вопросы**”, а также вопросы по листингу программы (отметьте себе, не по тексту программы, по листингу).

Работа считается выполненной полностью и в срок, если студент полностью сдал и защитил отчет ЛР в срок. Если студент сделал работу с дополнительными требованиями, то это обязательно отмечается в журнале ЛР и учитывается в оценке при подведении итогов семестра по данной дисциплине и на экзамене. Если студент выполнил все ЛР с дополнительными требованиями и получил отметки не ниже “хорошо”, то на зачете он освобождается от решения задачи (задачи на зачете заключаются в написании процедуры на языке Ассемблер или

командного файла) и может претендовать на получение автоматической оценки по курсовой работе - **ОТЛИЧНО**, при своевременной ее сдаче.

Если преподаватель обнаруживает (поверьте, сделать очень просто), что программа и отчет по ней сделаны **несамостоятельно** (проще - списаны), то он отмечает данный факт в журнале, а на зачете в этом случае задаются дополнительные вопросы по лабораторным работам, методическому пособию и материалам лекций. Кроме того, студент в этом случае не вправе рассчитывать на оценку по курсовой работе выше чем – **удовлетворительно**.

Для выполнения цикла лабораторных работ по курсу полезно познакомиться с указанными выше разделами методических указаний к ЛР[7], подготовленных преподавателем (отдельный документ – есть на сайте – **оранжевая кнопка**).

6. 6. Общие требования и замечания к ЛР по языку Ассемблер

Данные требования относятся к 3, 4, 5, 6, 7, 8 и 9 ЛР по дисциплине системное программирование, выполняемые в виде относительно простых и взаимосвязанных программ на языке Ассемблер (следующая программа может быть разработана на основе предыдущей с соответствующими добавлениями или исключениями). Важной частью работы является оформление отчета по ЛР. Нужно учитывать, что время необходимое на грамотное оформление отчета может быть соизмеримо со временем разработки и отладки программы. При разработке отчета полезно использовать опыт, полученный в предыдущем семестре. Требования к оформлению отчета и проведения работы приведены ниже в данном документе. Общие требования к оформлению ЛР заключаются в следующем:

- (**Новое в 2019!!!**) В листинге программы на ассемблере в первой строке вставляется **комментарий**, в котором обязательно указывается:
  1. ФИО студента и номер варианта по журналу
  2. Индекс группы
  3. Номер ЛР по ассемблеру.

Например:

; Большаков С.А вар№3, ИУ5-00 , ЛР №9

- (**Новое в 2019!!!**) В отчет по любой ЛР вставляется скриншот с первой страницей программы при выполнении в отладчике (TD.EXE), в котором виден комментарий, описанный выше.
- Для лучшего усвоения материала нужно внимательно прочитать теоретический раздел данного документа и **выполнить по шагам действия**, связанные с обработкой простой программы (печать символа – см. выше).
- Блок-схемы программ оформляются в MS WORD или MS VISIO (предпочтительнее), для чего используется специальный набор объектов и связей. Можно делать свои.

- Если студенты используют процедуры, блок-схемы которых уже были оформлены в ЛР с предыдущими номерами (а это возможно и даже необходимо), то допускается не дублировать оформление этих блок-схем в новых отчетах.
- Даты сдаваемых программ (*.com или ***.exe**) и распечаток программ должны соответствовать датам сдачи ЛР в семестре.
- При защите отчета по ЛР студент должен иметь в наличии на внешнем носителе (дискета, карта памяти, CD и т.д.): исходный текст программы, работающую программу в загрузочном виде, отчет в электронном формате MS WORD. Отчет по ЛР для защиты также должен быть распечатан.
- На титульном листе отчета должно быть отмечено, что работа выполнена с дополнительными требованиями или без них.
- В отчете должен содержаться листинг программы, формируемый компилятором Ассемблера (**!!!!!**), а не исходный текст программы. В противном случае отчет считается оформленным неправильно и защита не проводиться.
- Студент должен свободно ориентироваться в отчете и листинге своей программы и быстро находить место выполнения действий и команд, указанных преподавателем.
- Не следует комментировать каждую строку программы, такой вариант комментирования практически считается подсказкой и может привести к заданию лишних вопросов. Традиционно комментарии отмечают выполнение действий, связанных с группами команд (блокам), соответствующих в блок-схеме программы. Блок-схема тоже не должна быть чрезмерно детализирована (нет необходимости в покомандной детализации). Блок-схема должна четко соответствовать алгоритму и логике программы.
- Студент должен владеть навыками и знаниями по использованию служебных программ: отладчика, файл менеджера, и продемонстрировать их при сдаче и защите ЛР.
- Все работы выполняются индивидуально, каждый студент имеет свой отдельный результат по варианту и защищает работу индивидуально.

Примечание. В заголовках названий ЛР указывается год их актуальности. Для текущего семестра год должен соответствовать текущему году (например, **2016**).

Примечание. Изучите также раздел в конце данного документа: глава № 11. - “Общие требования к ЛР по СП”.

7. 7. Лабораторная работа № 3. (Вывод трех символов)- 2019!

7.1. 7.1 Задание на ЛР №3 СП

Разработать и отладить программу на языке Ассемблер для вывода на экран дисплея трех первых заглавных русских букв (А, Б, В), на трех отдельных строках дисплея подряд (отдельно программируется перевод строки и возврат каретки!). Работа выполняется в режиме командной строки (допускается использование для отладки и демонстрации любого файлового менеджера). В процессе работы студент учиться вводить исходный текст программы, выполнять ее компиляцию, редактирование связей и компоновать исполнимый модуль программы. При выполнении работы необходимо обязательно использовать и освоить отладчик Ассемблера (TD или QC). Для правильного вывода русских букв необходимо запустить русификатор (RKM или другой) перед запуском программы..

После завершения вывода букв на экран организовать ожидание ввода любого символа с клавиатуры (нажатие клавиши).

Оформить отчет по ЛР. Для оформления отчета студент должен знать или найти способ для вывода результата работы программы в текстовый файл. Лучше использовать копирование текста из окна командной строки (нежелательно снимать графическую картинку с экрана).

Методическое пояснение 1: Программе должны быть задействованы подфункции**: 1h, 2h, 4Ch** (в регистре АН) прерывания **021h**. Смотрите справочники и разделы методического пособия по ЛР.

Методическое пояснение 2: Простой запуск на компиляцию, редактирование связей, отладку программы рассмотрен в разделе № 2 методического пособия [7]. Там есть примеры запуска программ и поученные в программе результаты.

7.2. 7.2 Обязательные требования к ЛР

Необходимо использовать процедуры при разработке программы. Предусмотреть минимально **три** процедуры: для ввода символа (1-я процедура - **GETCH** название процедуры ввода символа желательно взять такое название), для вывода одного символа (2-я процедура - **PUTCH**) и для перевода строки с возвратом каретки (3-я процедура - **CLRF**) на дисплее (оформление процедур - PROC - ENDP, вызов процедур - CALL). Детальное оформление процедур описано в разделе № 10 методического пособия[7]. После вывода букв программа переходит в состояние ожидания нажатия любой клавиши. Выход из программы осуществить посредством системного прерывания **21H - 04CH** с указанием кода возврата 0. Ввод символа используется для организации ожидания завершения программы и просмотра результата.

Примечание: Вывод русских букв в правильном начертании должен быть выполнен: при работе программы, в листинге программы (в отчете) и в распечатке результатов ее работы. Нужно иметь виду, что при работе программы (эмуляторы ДОС и командной строки) и ее распечатке в документе отчета (MS WORD) используется разная кодировка (ANSI и ASCII). Для перекодировки можно использовать программу: Программа перекодировки DOS-WINDOWS - trans.exe (есть на сайте) или редактор текста с аналогичными возможностями. Все вопросы, связанные с кодами и использованием русификаторов изложены в разделе 23 методического пособия.

Методическое пояснение 3: Во всех ЛР не разрешается использовать системные макрокоманды и псевдооператоры Ассемблера для оформления сегментов (команды типа – .model, .data, .code и др.). Оформление сегментов выполняется директивами ассемблера: SEGMENT, ENDS и ASSUME (смотрите простой пример в пособии [7]). Это нужно для того, чтобы вы освоили приемы правильного оформления программы. Макрокоманды мы будем изучать и использовать в других ЛР (ЛР №9).

Методическое пояснение 4: На первом этапе работы, для усвоения правил оформления и обработки программ на Ассемблере, можно создать и отладить самую простую программу без процедур для вывода одного символа (раздел №2 пособия). Затем можно преобразовать эту программу в свой вариант программы с процедурами.

Методическое пояснение 5: Оформление и использование процедур в программе рассмотрено в разделе № 10 методического пособия.

Методическое пояснение 6: Более детальное описание технологии подготовки программы рассмотрено в разделе № 3 общего методического пособия и МУ данного цикла ЛР по ассемблеру.

Методическое пояснение 7: Перевод строки и возврат каретки осуществляется выводом на дисплей специальных кодов, которые нужно найти в электронном справочнике и в разделе № 23 пособия по ЛР (процедура **CLRF,** выводим на экран коды 10,13). При оформлении этой процедуры желательно из нее дважды вызвать уже разработанную и отлаженную процедуру вывода одного символа (**PUTCH**).

Методическое пояснение 8: Оформление блок-схемы программы этой ЛР рассмотрено в разделе № 21 общего методического пособия.

7.3. 7.3 Дополнительные требования к ЛР № 3

В программе организовать очистку экрана до начала вывода символов, а также после завершения работы программы. Очистка экрана должна выполняться отдельной дополнительной

процедурой на языке Ассемблер (название ее - **CLRSCR**). Очистка экрана должна быть выполнена без организации циклов вывода символов с помощью соответствующего прерывания (найденного вами в справочнике). При выполнении дополнительных требований в текст программы добавляется специальный комментарий, подтверждающий их выполнение. На титульном листе отчета нужно отметить факт выполнения ЛР с дополнительными требованиями

7.4. 7.4 Дополнительные требования к ЛР № 3 для сильных студентов

Организовать циклическое выполнение основной части программы при нажатии любой клавиши (отличной звездочки “*”) для повторного вывода 3-х символов. Использовать СП QC25 (есть на сайте) для отладки программы ЛР №3, освоив все основные режимы (ввода текста, компиляции и компоновки, пошаговой отладки, точек останова, просмотра регистров и процедур!). В программе дополнительно вывести столбиком (построчное и побуквенное) фамилию, имя и отчество студента -ФИО (для организации вывода этого использовать команду LOOP и собственные процедуры).

Примечание. На титульном листе **отметить наличие дополнительных требований** так: (**1**)Цикл по '*', (**2**)освоение QC25, (**3**)очистка экрана в процедуре - **CLRSCR**.

7.5. 7.5 Контролируемые требования по 3-й ЛР

- Русификация комментариев (в отладчике) и вывода символов (на экран КС)
- Блок-схема программы
- Наличие скриншота и комментария в отчете ЛР №3
- Занесение значения сегментного регистра данных
- Очистка экрана перед работой программы (INT)
- Оформление и вызов процедур программы (CALL,RET,PROC,ENDP)
- Вывод одного символа (процедура - **PUTCH**).
- Перевод строки после вывода буквы (процедура **CLRF**)
- Вывод подсказки перед запросом завершения (Прерывание 09h-21h)
- Запрос ввода клавиши без эха (процедура **GETCH**)
- Корректное завершение работы программы. (Прерывание 4Ch-21h)
- Наличие грамотно-оформленного отчета по ЛР №3 (на основе шаблона).
- Распечатка результатов работы программы без скриншотов и без ручного набора вывода!!!
- Умение поменять в стандартной простой программе (TASM3.ZIP – **firstd.ASM** выводимый на экран символ на другой, провести повторную компиляцию и показать изменения в отладчике.)
- Наличие дополнительных требований (цикл вывода, процедура **CLRSCR, ФИО**).
- Ответы на контрольные вопросы ЛР №3

7.6. 7.6 Контрольные вопросы по 3-й ЛР

1. "Для чего нужна данная команда?". Для строки листинга программы указанной преподавателем?
2. Какие основные функции выполняет отладчик при программировании на Ассемблере?
3. Какие основные режимы выполнения программы в режиме отладки Вы знаете?
4. Зачем нужен компилятор ассемблера? Его основные функции.
5. Как можно получить информацию о режимах работы компилятора (параметры компилятора) в командной строке? Какие параметры компилятора установлены по - умолчанию?
6. Зачем нужен редактор связей (компоновщик) в СП? Его основные функции.
7. Как можно получить информацию о режимах работы редактора связей в командной строке? Какие параметры компоновщика установлены по - умолчанию?
8. Что нужно сделать для создания программы в формате .COM - исполнимого файла?
9. Как задаются параметры процедуры на Ассемблере и как вызываются процедуры?
10. Что такое стек? Какие команды работы со стеком Вы знаете?
11. Поясните машинное представление команды Ассемблера, указанной преподавателем по листингу (в левой колонке листинга).
12. Какой отладчик вы применяли в работе? Какие режимы отладки Вы знаете?
13. Как можно получить информацию о режимах работы отладчика в командной строке?
14. Вопрос по меню QC 2.5 и опциям компилятора TASM (в зависимости от используемой в ЛР системы программирования).
15. Какое прерывание используется для вывода одного символа на экран?
16. Какое прерывание используется для ввода одного символа с клавиатуры?
17. Какое прерывание используется для корректного завершения программы?
18. Какой командой выполняется вызов процедуры?
19. Что происходит с регистрами IP, SS и стеком при коротком (NEAR) вызове процедуры? Показать в отладчике.
20. Что происходит с регистрами IP, SS и стеком при выполнении команды RET для возврата из процедуры. Показать в отладчике.
21. Какие возможности и операции есть у отладчика при выполнении программ?
22. Как в Ассемблере оформляются и вызываются процедуры?
23. Как в процедуру можно передать параметры и возвратить результат из нее?
24. Какие коды используются для перевода строки и возврата каретки на дисплее?

7.7. 7.7 Требования к оформлению отчета по ЛР №4

Отчет по ЛР № 3 должен содержать (см. шаблон в конце МУ):

- Титульный лист (Смотри образец ниже в конце этого документа).
- Кратко – Цель и задание на ЛР.
- Перечень собственных ошибок.

- Блок-схема алгоритма программы.
- Распечатка листинга программы в формате Ассемблера (.LST).
- Распечатка результатов работы программы (не набор в WORD, а реальная распечатка).

Более детальные требования к оформлению отчетов для всех лабораторных работ по Ассемблеру рассмотрены ниже.

8. 8. Лабораторная работа № 4. (Циклы и перевод символов) – 2019!

8.1. 8.1 Задание на ЛР №4

Разработать и отладить циклическую программу на языке Ассемблер для вывода на экран **20** последовательных прописных букв русского алфавита (начиная с символа “А” или другого символа, введенного с клавиатуры – см. коды ASCII – пособие [7] раздел 23.). Символы должны быть представлены в символьном(печатном) и шестнадцатеричном представлении (через черточку) в виде столбчатой таблицы (см. ниже). Каждая буква выводится в виде ее символьного представления и его 2-х разрядного шестнадцатеричного числа на одной строке. Например (СИМВОЛ – Шестнадцатеричный код):

А – 80h.
Б – 81h.
В – 82h.
Г – 83h.
…

8.2. 8.2 Обязательные требования к ЛР №4

В программе должна быть выполнена автоматическая шестнадцатеричная перекодировка, на основе преобразования машинного представления кода символа.

Шестнадцатеричная перекодировка (перевод одного представления в другое) должна выполняться командой **XLAT** (изучить в пособии [7] раздел 16.1) по специальной таблице перекодировки вида: 0123456789ABCDEF (применение этой команды рассмотрено в разделе № 16 методического пособия по ЛР [7]). Переведенные представления русских букв выводятся на экран дисплея последовательно. В каждой строке выводиться только одна буква с переводом (например, "**`A – 80h`**" – пример для кодировки ДОС - ASCII). Для организации цикла использовать команду **LOOP** (см. в пособии [7] раздел 16.7). Разработать блок-схему программы (см. в пособии [7] раздел 21!). Использовать MS VISIO для блок-схемы или другой доступный графический редактор.

После завершения вывода таблицы нужно организовать ожидание ввода нового символа с клавиатуры для вывода новой таблицы (процедура - **GETCH**). Если вводиться заранее предопределенный символ (например, символ “*****”), то программа должна завершаться с сообщением о своем завершении. В противном случае циклически выводиться новая таблица для нового введенного символа. В программе разработать и использовать **четыре** отдельные процедуры:

- для ввода символа(без эха) (1 - **GETCH**),
- вывода одного символа (2 - **PUTCH**),

- для перевода буквы в двух символьное шестнадцатеричное представление (3-я процедура **HEX**) и перевода строки и возврата “каретки” экрана дисплея (4 - **CLRF**) и
- для очистки экрана (процедура - **CLSSCR**).

Выход из программы выполнить посредством прерывания 21H - 04CH после нажатия любой клавиши, с заданием кода завершения – 5 См Справочники (ЛР № 3) - ERRORLEVEL.

Оформить отчет по ЛР. Для оформления отчета студент должен знать или найти способ для вывода результата работы программы в текстовый файл (без скриншртов!!!). Лучше использовать копирование текста из окна командной строки (нежелательно снимать графическую картинку с экрана).

Методическое пояснение 1: Процедуры **GETCH**, **PUTCH** и **CLRF** могут быть использованы из 3-й ЛР. Процедура **HEX** может/должна выполнять перевод и одновременно выводить на экран сразу две шестнадцатеричные цифры для конкретного кода символа (буквы), который передается в качестве параметра (буферизация кодов не обязательна). Оформление и использование процедур на ассемблере рассмотрено в разделе №23 общего методического пособия по ЛР СП [7].

Методическое пояснение 2: Применение команды XLAT показано в разделе № 16.1 методического пособия по СП [7].

Методическое пояснение 3: Цикл вывода символов организовать с помощью команды цикла - **LOOP**. Применение команды LOOP рассмотрено в разделе № 16. методического пособия.

8.3. Дополнительные требования к ЛР

Организовать очистку экрана до начала работы основной программы лабораторной работы, после вывода текущей таблицы и после завершения работы программы. Очистка экрана выполняется библиотечной функцией ОС (видео сервис - 10H). Организовать вывод букв в виде таблицы с рамкой из одинарных линий (Для этого нужно использовать символы псевдографики, например, такие как: “┌”, “─” , “┐”, “├”, “┤” и др. – смотрите в справочниках и в пособии см. раздел № 23 пособия раздел о кодах). Например:

```
┌─────────┐
│ A = 80H │
├─────────┤
│ Б = 81H │
└─────────┘
```

…

Таблицу для вывода символов можно построить и по-другому: с двойной рамкой, звездочками и т.д. Ввод символов псевдографики может быть выполнен в режиме alt – ввода (в

командной строке с цифровой клавиатурой !!! – изучить и опробовать режим alt – ввода: в ДОС УДЕРЖИВАЕТСЯ КЛАВИША ЛЕВОГО **ALT** и на NUMPAD НАБИРАЕТСЯ НУЖНЫЙ КОД. Для " ┌" – код – 218 или DAh). См. также коды ASCII в пособии [7] раздел 23.

Методическое пояснение 4: Нужно познакомиться с понятием кодов и их разновидностями Вы можете в разделе № 23 методического пособия. Уметь различать кодировки ANSI и ASCII.

Методическое пояснение 5: При выводе символов (см. дополнительные требования для сильных студентов) нужно контролировать вывод символов управления дисплеем и других управляющих кодов на экран, при необходимости их блокировать (Например, код 07 – сигнал через динамик). Эти символы не нужно выводить на экран, а нужно их распознавать и вместо них выводить знак "?".

Методическое пояснение 6: Для правильного оформления блок-схем программ нужно познакомиться с разделом № 21 методического пособия.

8.4. 8.4 Дополнительные требования к ЛР № 4 для сильных студентов

Обеспечить контроль ввода значения начального символа (печатные символы находятся в диапазоне 20 – 255, недопустимые в диапазоне 0 – 20), начиная с которого выводится на экран 20 символов. Ввод недопустимого символа проверить в режиме ALT –ввода (см. ниже). При вводе символа вне диапазона выдать сообщение: "НЕДОПУСТИМЫЙ СИМВОЛ!". Оформить специальную процедуру вывода таблицы с **двойными** рамками (см. символы:"║" – код 186, "╦"и т.д.) и символами.

8.5. 8.5 Контролируемые требования по 4-й ЛР

- Русификация комментариев (в отладчике) и вывода символов (на экран КС)
- Блок-схема программы, оформленная по ГОСТ.
- Наличие скриншота и комментария в отчете ЛР №4
- Занесение значения сегментного регистра данных для сегмента данных.
- Оформление сегментов данных и стека (ASSUME, SEGMENT, ENDS).
- Очистка экрана перед работой программы (INT 10h, процедура CLSSCR)
- Оформление и вызов процедур программы (CALL,RET,PROC,ENDP)
- Вывод подсказки перед запросом завершения (Прерывание 09h-21h)
- Запрос ввода клавиши и контроль кода для таблицы.
- Циклы их структура и команда LOOP.
- Процедуры ввода и вывода (PUTCH, CLRF GETCH).
- Распечатка результатов работы программы без скриншотов и без ручного набора вывода!!!
- Перевод в шестнадцатеричное представлении (процедура HEX – команда XLAT)

- Корректное завершение работы программы с сообщением. (Прерывание 4Ch-21h) с кодом 5.
- Умение поменять в стандартной простой программе (TASM3.ZIP – **firstd.ASM** выводимый на экран символ на другой, провести повторную компиляцию и показать изменения в отладчике.)
- Наличие грамотно-оформленного отчета по ЛР №4 .
- Наличие заявленных дополнительных требований и их проверка (цикл, рамки, контроль ввода и т.д.).
- Распечатка результатов работы программы.
- Ответы на контрольные вопросы ЛР №4

8.6. 8.6 Контрольные вопросы по 4-й ЛР

1. "Для чего нужна данная команда?". Для строки листинга указанной преподавателем?
2. Какие разновидности команды XLAT Вы знаете, и чем они отличаются?
3. Как выполняется перекодировка с помощью XLAT? Покажите в программе и поясните.
4. Какова максимальная длинна таблицы перекодировки в команде XLAT?
5. Что такое ASCII коды?
6. Что такое ANSY коды?
7. Что такое UNICOD код?
8. Что такое Scan коды?
9. Что такое **extended key** коды?
10. Как организовать ожидание ввода символа в Ассемблере?
11. Можно ли в процедуру Ассемблера передать параметры при ее вызове и как?
12. Что такое стек и для чего он нужен?
13. В каких командах Ассемблера явно используется стек?
14. В каких командах Ассемблера неявно используется стек?
15. Как задается число повторений в команде LOOP?
16. Как проверяется конец окончания цикла в команде LOOP?
17. Можно ли прервать выполнение цикла до его завершения по счетчику?
18. Какие есть ограничения использования команды LOOP? Что делать при их возникновении?
19. Поясните по листингу работу команды LOOP?
20. Поясните по листингу шестнадцатеричное представление указанной команды?
21. Какое прерывание используется для вывода одного символа на экран?
22. Какое прерывание используется для ввода одного символа с клавиатуры?
23. Какое прерывание используется для корректного завершения программы?
24. Зачем нужен компилятор ассемблера?
25. Как организовать в программе двойной цикл (вложенный) на языке Ассемблер?

26. Как можно получить информацию о режимах работы компилятора в командной строке?
27. Зачем нужен редактор связей в СП?
28. Как можно получить информацию о режимах работы редактора связей в командной строке?
29. Как выполнить Alt – ввод? Ввод символов псевдографики.
30. Что нужно сделать для вывода и ввода русских букв в окне командной строки Дос.

8.7. 8.7 Требования к оформлению отчета по ЛР №4

Отчет по ЛР № 4 должен содержать (см. шаблон в архиве с МУ по ассемблеру):

- Титульный лист (Смотри образец ниже в конце этого документа).
- Кратко – Цель и задание на ЛР.
- Привести перечень собственных ошибок.
- Блок-схема алгоритма программы.
- Распечатка листинга программы в формате Ассемблера (.LST).
- Распечатка результатов работы программы.

Более детальные требования к оформлению отчетов для всех лабораторных работ по Ассемблеру рассмотрены ниже.

9. 9. Лабораторная работа № 5. (Ввод/вывод в адреса и числа) – 2019!

9.1. 9.1 Задание на ЛР №5

Разработать и отладить программу на языке Ассемблер для ввода и буферизации строки символов с клавиатуры (последовательности символов) и затем последовательного их вывода на экран в шестнадцатеричном представлении (через пробел). В данной программе для корректной работы необходимо предусмотреть запоминание строки символов в байтовом массиве. Программа и блок-схема должны содержать вложенные циклы (двойные циклы). Программу оформить в виде исполнимого *.COM файла.

9.2. 9.2 Обязательные требования к ЛР №5

Признак завершения ввода отдельной строки с клавиатуры – это символ "**$**" (он вводиться с клавиатуры для завершения ввода строки). Между введенной строкой символов и их шестнадцатеричным представлением должен располагаться знак равенства (“=”). Максимальное число вводимых символов не должно превышать 20-ти. В данной программе цикл ввода (с клавиатуры) организуется с помощью команд условного (JE, JNE) перехода и команды безусловного перехода (JMP). После завершения ввода строки выполняется ее автоматический вывод. Организовать цикл ввода строк до ввода специального символа ('*'). Пример результата работы одного цикла программы показан ниже:

```
АБВ$ = 80 81 82
```

Требования к процедурам и их именованию совпадают с требованием предыдущих ЛР. Программа должна работать в циклическом режиме ввода строк (для внешнего цикла используется команда **LOOP**): после ввода одной строки запрашивается следующая (максимальное число вводимых строк для одного запуска программы равно **10**). Завершение цикла ввода строк может быть выполнено при вводе символа звездочка (“*****”), который должен быть введен в первой позиции строки. Вводимые символы строки записываются в символьный массив (буфер символов), максимальное число введенных символов равно 20-ти. Цикл ввода строки организуется командами условного и безусловного перехода. При вводе нужно подсчитать число введенных символов, включая символ доллара (“**$**”). Для вывода организуется цикл с помощью команды цикла (**LOOP**). В программе использовать процедуры предыдущих лабораторных данного цикла (ввода символа, печати, перевода строки и др.).

Для ввода/вывода строки и ее шестнадцатеричного представления разрабатываются дополнительная процедура HEX (см. ЛР №4). Организовать очистку экрана до начала работы программы, а также после ее завершения (С помощью специальной процедуры - CLRSCR).

В данной программе необходимо отдельно объявить отдельно сегмент данных (**DTSEG**) и сегмент стека (**STSEG**). Проверить загрузку сегментного регистра данных (**DS**) с помощью команды пересылки (**MOV)** , но через промежуточный регистр (**AX**).

Оформить отчет по ЛР. Для оформления отчета студент должен знать или найти способ для вывода результата работы программы в текстовый файл. Лучше использовать копирование текста из окна командной строки (нежелательно снимать графическую картинку с экрана).

Методическое пояснение 1: Процедура **HEX** для перевода символа может быть использована из 4-й ЛР.

Методическое пояснение 2: Для очистки экрана использовать отдельную процедуру – CLRSCR, а в ней нужно использовать прерывание **BIOS 010h**, для очистки экрана.

Методическое пояснение 3: В программе должно быть построено три цикла: цикл ввода символов, цикл вывода их шестнадцатеричного представления и общий цикл ввода строк. При организации вложенных циклов необходимо сохранять регистр CX.

Методическое пояснение 4: Проверка завершения внешнего цикла может быть выполнена командой CMP (раздел № 15.4 пособия) и командами условного перехода, например JE <метка>.

Методическое пояснение 5: Обратите внимание В текущем семестре снимается требование оформления этой программы в виде *.COM файла!

9.3. 9.3 Дополнительные требования к ЛР

Использовать для буферизации строки стек, а не массив. Предусмотреть дополнительно, кроме строк, ввод и перевод чисел, которые начинаются с цифры (0-9) и могут быть заданы в десятичном или шестнадцатеричном формате (шестнадцатеричные должны завершаться символом "h/H", десятичные числа без символов). Числа должны быть также переведены в машинное представление. По завершению программы на отдельной строке должно выдаться сообщение об ее успешном завершении программы и данные студента: **ФИО, группа и номер варианта студента**. Проверить оформление, компиляцию и компоновку в виде исполнимой программы формата ***.COM**.

Методическое пояснение 5: Считать, что размер вводимых числовых данных ограничивается размером, ограниченным двойным словом (DD – 4 байта в машинном виде).

9.4. 9.4 Дополнительные требования к ЛР для сильных студентов

Подавить ввод символа конца строки (“$”) при вводе строки (ввод без эха символов).

```
АБВ = 80 81 82
```

Создать отдельную универсальную процедуру для ввода и перевода в машинный формат числовых данных (разных типов: десятичных, восьмеричных и шестнадцатеричных) с клавиатуры. В процедуре предусмотреть контроль размерности вводимых чисел и выдачу диагностических сообщений при ошибках ввода. Процедура должна в стеке принимать параметры настройки и возвращать результат работы.

9.5. 9.5 Контролируемые требования по 5-й ЛР

- Русификация комментариев (в отладчике) и вывода символов (на экран КС)
- Блок-схема программы, оформленная по ГОСТ.
- Наличие скриншота и комментария в отчете ЛР №5
- Использование индексной адресации в массиве.
- Оформление программы трех сегментов (программы , данных и стека).
- Занесение значения сегментного регистра данных
- Циклы их структура и команда LOOP.
- Оформление и вызов процедур программы (CALL,RET,PROC,ENDP, XLAT)
- Процедуры ввода, вывода и очистки (PUTCH, CLRF GETCH, CLSSCR).
- Описание и использование байтового массива или стека для буферизации символов.
- Использование вложенных циклов с командой LOOP (повторный ввод строк).
- Выделение в программе *. COM области для PSP. Доступ к области PSP из *.EXE программ.
- Перевод в шестнадцатеричный формат (процедура HEX).
- Распечатка результатов работы программы без скриншотов и без ручного набора вывода!!!
- При разработке с дополнительными требованиями их контроль (, стек, COM, циклы, и т.д.)

9.6. 9.6 Контрольные вопросы по 5-й ЛР

1. "Для чего нужна данная команда?". Для строки листинга указанной преподавателем?
2. Какие разновидности команды XLAT Вы знаете, и чем они отличаются?
3. Как организовать ожидание ввода символа в Ассемблере?
4. Какие системы счисления Вы знаете?
5. В чем отличие символьного и шестнадцатеричного представления в ОП?
6. Можно ли в процедуру Ассемблера передать параметры при ее вызове и как?
7. Что такое стек и для чего он нужен?
8. В каких командах Ассемблера неявно используется стек?
9. Поясните по листингу работу команды CMP?
10. Поясните по листингу работу команды условного перехода - JE/JNE?
11. Как работает команда LOOP?
12. Какие регистры неявно используются в команде LOOP?
13. Поясните по листингу работу команды LOOP?
14. Поясните по листингу шестнадцатеричное представление указанной команды (в левой части листинга Ассемблера программы)?
15. Как организовать вложенный цикл?
16. Как по листингу определить размер программы в оперативной памяти (ОП)?

17. Как по листингу определить длину фрагмента программы, указанного преподавателем?
18. Расшифруйте по листингу формат команды, указанной преподавателем?
19. Чем отличаются *.COM и *.EXE исполнимые файлы, построенные Ассемблером?
20. Как можно в программе ассемблера организовать ветвление?
21. Поясните команды условного и безусловного перехода в программе.
22. Какие команды сравнения Вы знаете?
23. Какие команды условного перехода Вы знаете?
24. Как выполнить ввод символа с клавиатуры без отображения его в окне командной строки?

9.7. 9.7 Требования к оформлению отчета по ЛР №5

Отчет по ЛР № 5 должен содержать (см. в конце данных МУ):

- Титульный лист (Смотри образец ниже в конце этого документа).
- Кратко – Цель и задание на ЛР.
- Привести перечень собственных ошибок.
- Блок-схема алгоритма программы.
- Распечатка листинга программы в формате Ассемблера (.LST).
- Распечатка результатов работы программы.

Более детальные требования к оформлению отчетов для всех лабораторных работ по Ассемблеру рассмотрены ниже.

10. 10 Лабораторная работа № 6. (Ввод и распечатка параметров к.с.) – 2019!

10.1. 10.1 Задание на ЛР №6

Разработать и отладить программу на языке Ассемблер для ввода, анализа (расшифровки, фактически грамматического разбора) и распечатки параметра командной строки, которые задаются при запуске программы (параметры размещаются в области PSP со смещением 081h, ПРОБЕЛ в DOSBox!). Нужно также описать в БНФ синтаксис запуска вашей программы с параметрами в командной строке. Для этого надо изучить раздел 4 методических указаний к ЛР. Программа должна быть скомпонована в виде *.EXE - исполнимого файла. После запуска нужно проверить правильность первого параметра и наличие второго, после этого выдать соответствующие диагностические сообщения. Изучить структуру PSP и способы получения в программе адреса этого блока. Распечатать заданные параметры.

10.2. 10.2 Обязательные требования к ЛР

Предусмотреть ввод и анализ двух позиционных параметров командной строки (параметры читаются из области PSP), адрес PSP получается в программе автоматически. Параметры имеют строгую позицию в командной строке. Первый параметр задает фамилию студента (студентки). Нужно проверить правильность первого параметра – сообщение "Первый параметр верен", параметр распечатать, а наличие второго параметра – сообщение "Второй параметр есть/отсутствует". Написать и оформить в БНФ инструкцию для работы данной программы. Записать параметр в буфер программы командой **MOVSB**. Для проверки параметра использовать команду цепочек **CMPSB**.

Пример возможного вывода результата работы ЛР № 6:

```
Первый параметр верен= Иванов
Второй параметр отсутствует!
```

Или

```
Первый параметр неправильный
Второй параметр есть!
```

Методическое пояснение 1: После запуска программы *.EXE список параметров (текст вводимой командной строки сохраняется в PSP программы). Доступ к PSP может быть выполнен с помощью прерывания 21h – 51h или из сегментного регистра ES после первоначального запуска программы. Поле списка параметров начинается в PSP со смещение **081h** (См. справочник). В области PSP со смещением **80H** содержится число символов введенных параметров (один байт). **Примечание**. При создании *.COM – это д.т. программы PSP располагается непосредственно в начале программы (ORG 100h – область, в которую загрузчик записывает блок PSP).

Назначение и формат параметров должен быть следующий:

1. Первый параметр задает фамилию студента в именительном падеже.
2. Второй параметр произвольный – не менее 3-х символов.

Все параметры перед завершением программы распечатываются.

Данная программа компонуется и выполняется в виде *.EXE модуля.

Значения параметров программы (справка о параметрах) **дополнительные требования**:

Примеры запуска программы для **дополнительных** требованиях:

Пример **распечатки** введенных параметров обязательного варианта:

```
Введенные значения параметров программы:
1-й параметр – Большаков
2-й параметр – присутствует/отсутствует
```

Описать в БНФ синтаксис запуска в командной строке для разработанной программы. Для этого нужно изучить раздел 4 методических указаний к ЛР [7].

Оформить отчет по ЛР. Для оформления отчета студент должен знать или найти способ для вывода результата работы программы в текстовый файл. Лучше использовать копирование текста из окна командной строки (нежелательно снимать графическую картинку с экрана).

Методическое пояснение 2: С описанием и использованием метаязыка БНФ можно познакомиться в разделе № 4 методического пособия по ЛР.

Методическое пояснение 3: Между параметрами может быть произвольное число пробелов (**дополнительные требования**). В этом случае предусмотреть специальную процедуру для сброса лишних пробелов (название - **PROBCLR**).

Методическое пояснение 4: Для доступа к **PSP** нужно выполнить процедуру доступа с помощью прерывания 021H –функция 051H (на BX – адрес PSP). или запомнить адрес в регистре **ES** сразу после запуска программы, в нем тоже задается адрес PSP для *EXE файлов.

Методическое пояснение 5: Для доступа к PSP можно также использовать значение регистра **ES**, формируемое сразу при запуске программы, в нем тоже задается адрес PSP для *EXE файлов. Желательно в отладчике удостоверится, что оно совпадает с адресом PSP, полученным с помощью прерывания 021H –функция 051H.

Методическое пояснение 6: Для очистки экрана использовать отдельную процедуру – CLRSCR и использовать для специальной функции прерывания BIOS 010h.

10.3. 10.3 Дополнительные требования к ЛР

Использовать вариант минимум с тремя параметрами. Добавить ключевой параметр с проверкой (/c=[Y/N]) для очистки экрана до начала работы программы и выдачи справки (/h=[Y/N]). Предусмотреть дополнительный параметр, который задает ключ доступа к своей

программе (см. выше). При несовпадении ключа доступа программа завершается с диагностическим сообщением. Придумать позиционные и ключевые параметры.

Методическое пояснение 7: Разбор а проверку новых параметров желательно для упрощения отладки выделить в специальные процедуры для разбора новых параметров.

10.4. 10.4 Дополнительные требования к ЛР для сильных студентов

Учесть все дополнительные требования (см. выше раздел 10.3). Придумать и использовать новые дополнительные параметры (минимум два) для управления вашей программой. Все параметры программы могут занимать произвольное(!) место в перечне параметров командной строки (они все не позиционные, а ключевые). Например, один из параметров может задавать признак выдачи дополнительной информации о студенте и группе или инструкцию для ее запуска.

10.5. 10.5 Контролируемые требования по 6-й ЛР

- Русификация комментариев (в отладчике) и вывода символов (на экран КС)
- Блок-схема программы, оформленная по ГОСТ.
- Наличие скриншота и комментария в отчете ЛР №6
- Оформление в программе трех явных сегментов (программы – CODESGM, Стека – StackSGM, Данных - DTSGM).
- Получение адреса PSP двумя способами (функция 21H – 51H, и при запуске)
- Использование команды **MOVSB** для копирования первого параметра в буфер.
- Использование команды **CMPSB** для проверки первого параметра (сравнение с шаблоном).
- Чтение и проверка числа параметров КС.
- Занесение значения сегментного регистра данных (два способа).
- Оформление и вызов процедур программы (CALL,RET,PROC,ENDP, XLAT)
- Процедуры ввода, вывода и очистки (PUTCH, CLRF GETCH, CLSSCR).
- Описание и использование байтового массива для буферизации параметров.
- Использование вложенных циклов с командой LOOP.
- Выдача трех диагностических сообщений (проверка параметра, наличия второго параметра, завершения программы).
- Инструкция для работы программы в БНФ.
- Распечатка результатов работы программы без скриншотов в отчете.
- Наличие оформленного отчета по ЛР №6.

10.6. 10.5 Контролируемые требования по 7-й ЛР

- Все стандартные требования предыдущих работ по СП (см. выше)

- Алгоритм преобразования символьного числа в машинное (шестнадцатеричное) представление
- Преобразование в десятичное число двоичного числа и его вывод.
- Использование и знание схемы Горнера для взаимного перевода чисел.

10.7. 10.6 Контрольные вопросы по 6-й ЛР

1. "Для чего нужна данная команда?". Для строки листинга указанной преподавателем?
2. Что такое блок PSP? Как можно получить его адрес в программе?
3. Для каких целей PSP может использоваться в программах?
4. Какая информация хранится в PSP и как с ней работать?
5. Как можно узнать число байт введенных параметров в командной строке?
6. Как можно определить число введенных параметров?
7. Где можно получить информацию о структуре PSP?
8. Как выполнить доступ к PSP в *.COM файле?
9. Как выполнить доступ к PSP в *.EXE файле?
10. Поясните по листингу работу команд CALL и RET с учетом изменения и использования стека?
11. Поясните по листингу шестнадцатеричное представление указанной команды?
12. Как можно использовать в программе счетчик команд ($)?
13. Расшифруйте по листингу формат команды, указанной преподавателем (тип, формат и значение исполнительного адреса)?
14. Чем отличаются *.COM и *.EXE исполнимые файлы, построенные Ассемблером?
15. Какие классы прерываний Вы знаете?
16. Каким символом разделяются отдельные параметры при запуске программ?
17. Какое максимальное число параметров можно задать в командной строке (вычислите)?
18. Как Вы определили число параметров в своей программе?
19. Что такое БНФ и как ее используют программисты?
20. Какие данные из PSP Вы можете использовать в своих программах?
21. Какие виды прерываний Вы знаете?
22. В чем главное отличие программных и аппаратных прерываний?
23. Как сделать прерывания недоступными?

10.8. 10.7 Требования к оформлению отчета по ЛР №6

Отчет по ЛР № 6 должен содержать (см. ниже шаблон отчета):

- Титульный лист (Смотри образец ниже в конце этого документа).
- Кратко – Цель и задание на ЛР.
- Привести перечень собственных ошибок.

- Блок-схема алгоритма программы.
- Инструкция в БНФ для запуска разработанной программы.
- Распечатка листинга программы в формате Ассемблера (.LST).
- Распечатка результатов работы программы.

Более детальные требования к оформлению отчетов для всех лабораторных работ по Ассемблеру рассмотрены ниже.

11. 11.Лабораторная работа № 7. (Ввод, вывод и перевод адреса) – 2019!

11.1. 11.1 Задание на ЛР №7

Разработать и отладить программу на языке Ассемблер для ввода с клавиатуры четырехразрядного шестнадцатеричного числа – символами! (короткого адреса NEAR) в машинное шестнадцатеричном представлении (доступные шестнадцатеричные цифры – 0123456789ABCDEF). Введенное значение переводиться в машинное представление в виде отдельного слова (2 байта – DW – тип переменной). Полученное значение выводится затем на экран также в шестнадцатеричном представлении, но заново переведенное из машинного формата. Кроме того, выполняется перевод по схеме Горнера (см. в Википедии) в десятичное представление и на экран выводится в десятичном формате (нужно выполнить программный перевод из одной системы счисления в другую).

11.2. 11.2 Обязательные требования к ЛР

Между введенным символьным значением адреса и выводимым шестнадцатеричным представлением должен располагаться знак равенства ("="), а между – формируемыми представлениями пробел (шестнадцатеричным и десятичным).

Например (сначала машинное - **00FEh ,а затем десятичное - 254**):

```
Введите число( длинный адрес: HHHH:HHHH)>00FE=00FEh 254
>...
>*
Завершение ввода чисел!
```

Программа должна работать в циклическом режиме, то есть после ввода одного числа, запрашивается ввод нового. Завершение цикла ввода чисел выполняется по знаку “*****” в первой позиции строки ввода. Для ввода и перевода должны быть использованы базовые процедуры (см. ЛР выше). При вводе необходимо проверять вводимые шестнадцатеричные символы (0-9 и A -F)/ Нужно организовать очистку экрана до начала работы программы, и после ее завершения. По завершению программы выдается сообщение об ее успешном окончании и данные студента: ФИО, группа и номер варианта. Для запроса вводимого числа предварительно должна выдаваться подсказка в виде:

```
”Введите число( длинный адрес: HHHH:HHHH)>”:
```

Оформить отчет по ЛР. Для оформления отчета студент должен знать или найти способ для вывода результата работы программы в текстовый файл. Лучше использовать копирование текста из окна командной строки (нежелательно снимать графическую картинку с экрана). Программа может быть выполнена в виде *.EXE исполнимого модуля.

Методическое пояснение 1: Вывод информации нужно выполнить с помощью функции вывода строки 09h – 021h (предварительно нужно записать введенные и выводимые данные в буферные массивы). Не забудьте в конце строки выполнить перевод строки и возврат каретки с помощью закодированных в конце строки символов 0Ah и 0Dh. Строка в каждом из массивов должна при этом завершаться символом – "$".

11.3. 11.3 Дополнительные требования к ЛР

Предусмотреть ввод длинного адреса (FAR – <сегмент>: <смещение>), в виде двух шестнадцатеричных чисел разделенных знаком ":" (Например – 0099:01AFh).

11.4. 11.4 Дополнительные требования к ЛР для сильных студентов

Учесть все дополнительные требования предыдущих пунктов. Перевести введенные данные длинного адреса в десятичную и восьмеричную систему счисления, и вывести на экран дисплея в этой же строке.

11.5. 11.5 Контролируемые требования по 7-й ЛР

- Все стандартные требования предыдущих работ по СП (см. выше, процедуры, сегменты, сообщения и т.д.)
- Наличие скриншота и комментария в отчете ЛР №7
- Алгоритмы буферизации и преобразования символьного числа в машинное шестнадцатеричное представление.
- Перевод и печать числа в десятичном виде.
- Использование и знание схемы Горнера для взаимного перевода чисел (см. в Википедия).
- Проверка выполнения дополнительных требований к ЛР.

11.6. 11.6 Контрольные вопросы по 7-й ЛР

1. "Для чего нужна данная команда?". Для строки листинга указанной преподавателем?
2. Поясните назначение регистров: CS, DS, SS и ES.
3. Что такое короткий адрес (NEAR)?
4. Что такое длинный адрес (FAR)?
5. Как и когда заполняются сегментные регистры (CS, DS, SS и ES) в *.EXE .файле
6. Как в отладчике посмотреть содержимое стека?
7. Как получить адрес и местоположение PSP?
8. Какие регистры МП управления вы знаете? Их назначение?
9. Как съема Горнера используется при переводе чисел из одной системы счисления в другую?
10. Для чего нужны сегментные регистры?

11. Как определить в программе на языке Ассемблера адрес выполняемой команды?
12. Для чего в программе и как заноситься регистр DS?
13. Как в отладчике выполнить анализ выполняемой команды? Что для этого нужно сделать?
14. Как запустить компилятор без формирования отладочной информации?
15. Как запустить TLINK (или LINK) без формирования отладочной информации?
16. Для чего нужна утилита make.exe? Как ее использовать?
17. Для чего нужна утилита **grep.com**? Как ее использовать?
18. Для чего нужна утилита **tlib.exe** (**lib.exe**)? Как ее использовать?
19. Как при редактировании связей подключить объектные модули (*.obj) из библиотеки (*.lib)?

11.7. 11.7 Требования к оформлению отчета по ЛР №7

Отчет по ЛР № 7 должен содержать (см. ниже данные МУ):

- Титульный лист (Смотри образец ниже в конце этого документа).
- Кратко – Цель и задание на ЛР.
- Привести перечень собственных ошибок.
- Блок-схема алгоритма программы.
- Распечатка листинга программы в формате Ассемблера (.LST).
- Распечатка результатов работы программы.

Более детальные требования к оформлению отчетов для всех лабораторных работ по Ассемблеру рассмотрены ниже.

12. 12.Лабораторная работа № 8. (Вывод дампа оперативной памяти) – 2019!

12.1. 12.1 Задание на ЛР №8

Разработать и отладить программу на языке Ассемблер для вывода на экран дампа оперативной памяти в шестнадцатеричном виде (распечатки содержимого ОП) по адресу, задаваемому с клавиатуры в шестнадцатеричном виде (адрес FAR - пара чисел, например - 0000:12A0). При вводе задается смещение и сегментный адрес. Вывод информации на экран может производиться также и символьной форме, как при просмотре оперативной памяти в отладчиках и файл менеджерах. После вывода дампа одной страницы запрашивается новый адрес. Завершение работы программы выполняется при вводе звездочки (“*”).

12.2.12.2 Обязательные требования к ЛР № 8

Должно быть выведено 16 строк дампа, в каждой строке выводиться по 24 байта (48 шестнадцатеричных цифры). Вывести на экран значение регистров **AX** и **CS**. Каждая выводимая строка дампа должна начинаться с соответствующего адреса ее расположения (адрес отделяется от информации с помощью “: ” пробелом). Предусмотреть использование всех процедур, разработанных ранее, при разработке программы (см. ЛР выше). Организовать очистку экрана до вывода, после вывода нового дампа и после нажатия заданной клавиши выхода. По завершению программы выдается сообщение об ее успешном окончании. Адрес для вывода дампа может быть задан в программе или введен с клавиатуры (см. ЛР№7).

Формат вывода дампа памяти должен быть таким:

В результате на экран выводиться дамп памяти в шестнадцатеричном формате:

```
Введите адрес( длинный адрес: HHHH:HHHH)>1000:0200
Адрес дампа: 1000:0200
Диапазон: 0200 – 037F
AX=00FE  CS=00FE
1000:0200: 0A 0B DD 0A 0B DD 0A 0B DD 0A 0B DD 0B DD 0A 0B DD 0A 0B DD 0B DD 0A 0B
1000:0218: 00 0C 01 ...
1000:0230: 00 0C 01 ...
     ...
     ...
0368: 10FF 0F ...
```

Оформить отчет по ЛР. Для оформления отчета студент должен знать или найти способ для вывода результата работы программы в текстовый файл. Лучше использовать копирование текста из окна командной строки (нежелательно снимать графическую картинку с экрана).

Методическое пояснение 1: Для ввода адреса вывода дампа можно использовать процедуру или фрагмент программы из 7-й ЛР.

Методическое пояснение 2: Для вывода сообщений можно использовать процедуру или фрагмент программы из 7-й ЛР.

Методическое пояснение 3: Вывод дампа выполняется в двойном цикле (команда LOOP). Для организации двойного цикла нужно запоминать значение регистра CX при входе во внутренний цикл и восстанавливать его после его завершения (в стеке или в отдельной переменной).

12.3. 12.3 Дополнительные требования к ЛР

Программа должна работать в циклическом режиме и обеспечивать просмотр следующей страницы дампа при нажатии на клавишу "**Enter**"(листание дампа). Ввод длинного адреса дампа (FAR) с клавиатуры обязателен. Завершение цикла вывода дампов выполняется по знаку "*". Выводить дамп в отдельном окне, ограниченном рамкой из псевдосимволов текстового режима в фиксированном месте экрана. Значение выводимых регистров микропроцессора выводятся для начального момента запуска программы.

12.4. 12.4 Дополнительные требования к ЛР для сильных студентов

Выполнить все дополнительные требования. Предусмотреть прокрутку дампа вверх и вниз (стрелками ↑,↓) или листание по страницам (по клавишам PGUP и PGDN) в отдельном окошке с рамкой. Обеспечить режим обязательного одновременного просмотра дампа памяти в символьном и шестнадцатеричном виде (как в файловых менеджерах, например FAR). Оптимизировать построение программы с грамотным использованием старых и новых процедур. Для этого можно изменить число байт в шестнадцатеричном виде на экране (Например, до 16 байт). Изучить раздел методического пособия по резидентным программам. Перед дампом на экран в виде отдельных строк выводятся: **адрес** дампа, значения регистров данных, сегментов и указателей: **AX, BX, CX, DX, CS, DS, SS, ES, SP, SI, DI, BP**, а также диапазон адресов выводимой памяти (значения регистров и адресов должны быть разделены пробелом и указаны названия регистров).

12.5. 12.5 Контролируемые требования по 8-й ЛР

- Все стандартные требования предыдущих работ по СП (см. выше)
- Алгоритм преобразования символьного числа в машинное шестнадцатеричное представление.
- Наличие скриншота и комментария в отчете ЛР №8
- Использование и знание схемы Горнера.
- Переключение адреса на границе сегмента (д.т.), при циклическом вводе адреса и выводе данных, приводящего к проходу такой границы сегмента.

- Циклическое задание нового адреса.
- Умение показать в отладчике (TD.EXE) соответствие значений выводимых в виде дампа и реальных значений ОП.
- Вывод значений общих и сегментных регистров.
- Отчет по ЛР с результатами работы.
- Выполнение дополнительных требований.

12.6. 12.6 Контрольные вопросы по 8-й ЛР

1. "Для чего нужна данная команда?". Для строки листинга указанной преподавателем?
2. Какую функцию выполняет команда LEA?
3. Что такое ближний (NEAR) и дальний (FAR) адрес?
4. Как задается ближний (NEAR) и дальний (FAR) адрес?
5. Какие способы вы можете назвать для очистки экрана (минимум 2)?
6. Как организовать в программе Ассемблера вложенный цикл (с помощью 2-х команд LOOP)?
7. Как при выводе дампа памяти проверить выход за границу сегмента?
8. Поясните назначение регистров IP и EIP? В чем их отличие?

12.7. 12.7 Вопросы с учетом дополнительных требований по 8-й ЛР

9. В чем отличие между 16-ти и 32-ти разрядными приложениями?
10. Какие компиляторы нужно использовать для 16-ти и 32-ти разрядных приложений?
11. Что необходимо сделать для подключения собственного обработчика прерывания?
12. Что необходимо сделать для вызова старого обработчика прерывания, если Вы загрузили новый?
13. Какими способами можно проверить наличие в памяти собственного обработчика прерывания?
14. В чем специфика приложений под WINDOWS разработанных на языке Ассемблер?

12.8. 12.8 Требования к оформлению отчета по ЛР №8

Отчет по ЛР № 8 должен содержать:

- Титульный лист (Смотри образец ниже в конце этого документа).
- Кратко – Цель и задание на ЛР.
- Привести перечень собственных ошибок.
- Блок-схема алгоритма программы.
- Распечатка **листинга** программы в формате Ассемблера (.LST).
- Распечатка результатов работы программы.

Более детальные требования к оформлению отчетов для всех лабораторных работ по Ассемблеру рассмотрены ниже. Смотрите требования к отчетам, расположенные ниже в разделе 12 данного документа.

13. 13. Общие требования к ЛР по языку Ассемблера

Лабораторные работы выполняются студентами **индивидуально**. В тексте каждой программы в виде комментария должны быть отражены: **ФИО, группа, номер варианта в группе и номер ЛР по общей нумерации в курсе СП.** Результатом работы должна быть отлаженная программа (в виде *.COM файла или - *.EXE исполнимого модуля) и правильно оформленный отчет по каждой ЛР. В отдельных случаев указывается тип модуля (*.COM или *.EXE). Нужно следовать, в том числе следующим требованиям (кроме того, прочтите внимательно требования для каждой Лабораторной работы):

1) Программа должна быть скомпонована в виде .COM или *.EXE.
2) Программа должна быть снабжена комментариями для пояснения ее работы, интерфейса процедур и логики работы (только на русском языке!). Комментарии должны быть правильно видны в отладчике и в отчете ЛР. Комментарии не должны (!) сопровождать каждую строку текста, а должны выделять блоки программы (как на блок-схеме), пояснять назначение процедур и их параметров.
3) Дополнительные требования к ЛР не являются обязательными, но учитываются в дифференцированном зачете по КР, рейтингах и при проведении зачета по курсу (об этом делается отметка в журнале ЛР группы и студента).
4) Для каждой программы оформляется Блок-схема. Она должна учитывать все связи, переходы и процедуры программы, она составляется на достаточно детальном уровне, но не до уровня каждой команды. Для оформления блок-схем используются обозначения, рассмотренные в разделе № 21 методического пособия [7]. Блок-схемы необходимо оформлять в MS VISIO.
5) Каждая программа должна иметь заголовок в листинге (Псевдооператор/Директива Ассемблера **TITLE**), в котором указывается: **группа, номер лабораторной работы, группа и ФИО студента.**
6) При защите ЛР студент должен объяснить назначение и выполняемые действия для любого фрагмента **собственной программы** по листингу программы, указанному преподавателем, и любой команды. Студент в процессе выполнения ЛР должен научиться отвечать на все контрольные вопросы каждой ЛР, используя полученные знания, литературу, документацию и экспериментальную проверку фактов при отладке.
7) При выполнении лабораторных работ студенты должны использовать отладчик (либо TD, либо интегрированный отладчик в QC или др. инструменты). **Работу всех программ нужно уметь продемонстрировать в отладчике (код, данные и стек) с правильно русифицированным исходным текстом.** Необходимо знать все основные операции для отладки и все данные, которые можно просмотреть в процессе отладки и поля изменяемые при пошаговом выполнении команд (регистры, флаги, память и т.д.). Использование различных отладчиков поясняется в разделах 2, 3 и 4 методического пособия[7]. Во время

демонстрации программы студент должен показать умение работать с отладчиком Ассемблера, при этом в отладчике должен высвечиваться исходный текст программы Ассемблера, а не результаты дизассемблирования отладчиком. Для этого необходимо указать специальные параметры при запуске компилятора и компоновщика [7].

8) Студент должен хорошо разбираться в листинге и блок-схеме программы, а также в своем отчете по ЛР и результатам работы программы.
9) Студент должен **уметь** находить ошибки с помощью отладчика для Ассемблера.
10) Студент должен уметь работать **в файл-менеджерах.**
11)Студент обязан разработать и отладить программу **самостоятельно**!

14. 14.Требования к оформлению отчетов для ЛР по Ассемблеру

Для оформления ЛР СП по Ассемблеру целесообразно использовать шаблон отчета, который расположен на сайте в архиве с данным документом (МУ). По ЛР защищается по предоставлению работающей программы (на дискете или CD) и отчета (на бумажном носителе). Структура отчета дана в конце данного документа. Имейте в виду, что из-за сложного документа заголовки и оглавление нужно переоформить, если будете копировать (можно взять шаблон предыдущего семестра, там такие вносить исправления не нужно). Требования к отчету:

1) Отчет выполняется в текстовом редакторе MS WORD.
2) Титульный лист обязательно должен присутствовать. (Смотри образец ниже в конце этого документа).
3) Кратко – Задание и цель выполнения ЛР.
4) Перечень собственных ошибок зафиксированных при отладке. В отчете должны быть выделены не менее 3-х собственных ошибок. Ошибки фиксируются в таблице во время отладки программы ЛР.
5) Блок-схема алгоритма программы (в рукописном или машинном оформлении – MS VISIO). Блок-схемы процедур, повторяющихся в ЛР со старшими номерами ЛР можно не приводить.
6) Распечатка листинга программы в формате Ассемблера (.LST). Только в этом формате, простая распечатка исходного текста не допускается.
7) Распечатка результатов работы программы. Можно посредством перенаправления потока (>>) в файл, можно с помощью распечатки экрана или использованием копировщика экрана (ScrCapture) и др., но не вручную в WORD! Наиболее предпочтительным является вариант копирования текста результатов из окна командной строки (в окне командной строки: системное меню-> Изменить->Пометить->Enter, Для вставки в документ: Shift+Ins).
8) Для распечатки отчетов может быть использована двухсторонняя печать или печать двух страниц на одном листе (для экономии бумаги).

Методическое пояснение 1: Для подготовки отчетов, для уменьшения затрат времени, можно грамотно использовать шаблоны отчетов предыдущего семестра, не забыв изменить все данные: дисциплина, название ЛР и т.п. Эти шаблоны уже содержат оглавление, заголовки и т.д. Можно создать свой шаблон.

Для оформления отчетов в MS WORD нужно учитывать следующие требования к оформлению отчетов:

- Текстовый редактор - **MS WORD** (не Open Office!),
- Шрифт - **Times New Roman**.
- Кегль шрифта - **12**,
- Интервал между строками - **одинарный**,
- Способы форматирования текста – **по ширине**,
- Размеры страниц – **А4 (верх - 2 , низ - 2 , слева - 3 , справа - 2)**,
- Содержание колонтитулов (**вариант, группа, ФИО студента, № ЛР)**,
- Способы рисования иллюстраций документа (Предпочтительнее **MS Visio** можно **MS WORD**),
- Нумерация страниц – **центр – верх**.

15. 15. Общие контрольные вопросы к лабораторным работам по Ассемблеру

1. Необходимо хорошо ориентироваться в листинге программы (распечатки компилятора).
2. **Нужно** быть готовым ответить на вопрос: "Для чего нужна данная команда?" для любой строки листинга.
3. Какие основные функции выполняет отладчик (TD)?
4. Какие основные режимы выполнения программы в режиме отладки Вы знаете?
5. Зачем нужен компилятор и его настройки?
6. Зачем нужен редактор связей и его настройки?
7. Что необходимо сделать для создания программы в формате .COM – исполнимого файла?
8. Чем отличаются форматы *.com и *.exe?
9. Как задаются параметры процедуры на Ассемблере и как вызываются процедуры?
10. Что такое стек, и какие команды работы со стеком Вы знаете?

Эти вопросы могут быть заданы при защите любой ЛР из данного цикла!!!

16. 16. Литература по ЛР СП

1. К.Г.Финогенов Основы языка Ассемблера.— М.: Радио и связь, 2001.— 288 с.
2. П.И.Рудаков, К.Г.Финогенов “Язык ассемблера: Уроки программирования” – М.: ДИАЛОГ-МИФИ, 2001 г., 640с.

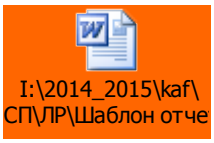
3. К.Г. Финогенов “Самоучитель по системным функциям MSDOS”-М.,РиС,Энтроп, 1995 г. 382с.
4. Скэнлон Л. “Персональные ЭВМ IBM PC. Программирование на языке ассемблера.” - М.,РиС,1991 г.
5. Р.Джордейн “Справочник программиста персональных компьютеров типа IBM PC”- М.,ФиС, 1991г.
6. Список литературы по дисциплине СП, представленный на сайте (www.sergebolshakov.ru).
7. “ Методическое пособие для выполнения лабораторных работ по дисциплине СП ”, представленное на сайте (www.sergebolshakov.ru)..

17. 17. Сроки представления и защиты заданий по курсу:

1 ЛР - 2 неделя семестра (справочники)
2 ЛР - 2 неделя семестра (командные файлы)
3 ЛР - 4 неделя семестра (Ассемблер)
4 ЛР - 4 неделя семестра (Ассемблер)
РЕЙТИНГ ПО ЛР И ЛЕКЦИЯМ - 5-6 неделя
5 ЛР - 6 неделя семестра (Ассемблер)
6 ЛР - 8 неделя семестра (Ассемблер)
7 ЛР - 10 неделя семестра (Ассемблер)
8 ЛР - 12 неделя семестра (Ассемблер)
9 ЛР - 13 неделя семестра (Ассемблер)
РЕЙТИНГ ПО ЛР И ЛЕКЦИЯМ - 11-13 неделя
СДАЧА ДЗ – 4 неделя
ЗАЧЕТ ПО КУРСУ - (зачетная неделя)
Сдача курсовой работы 14-неделя

При обнаружении на носителях информации студента и в программах курсовых и лабораторных работ вирусов любой породы, оценка студенту снижается на *один* балл !!!

18. 18. Шаблон отчета по ЛР № 3..8



Шаблон отчета в виде документа с полями форматирования

Смотрите ниже структуру шаблона отчета без автоматических полей!!!:

Защищено:
Большаков С.А./Аксенова М.В.

"__"_____201X г.

Демонстрация:
Большаков С.А. Аксенова М.В.

"__"_____201X г.

Лабораторная работа №XX по курсу Системное программирование

"**********?Название?**********"

(есть дополнительные требования или требования для продвинутых студентов)

53
(количество листов)

ИСПОЛНИТЕЛЬ:
студент группы _____
ИУ5-4X
Большаков С.А.
"__"_____201X г.

Москва - 201X

СОДЕРЖАНИЕ

1. Цель выполнения лабораторной работы № X

(Приводится цель выполнения работы из методических указаний по ЛР).

2. Порядок и условия проведения работы № X

(Описывается последовательность шагов при выполнении ЛР. Эту информацию Вы сможете найти в методических указаниях к каждой конкретной ЛР).

3. Описание ошибок при отладке программы ЛР № X

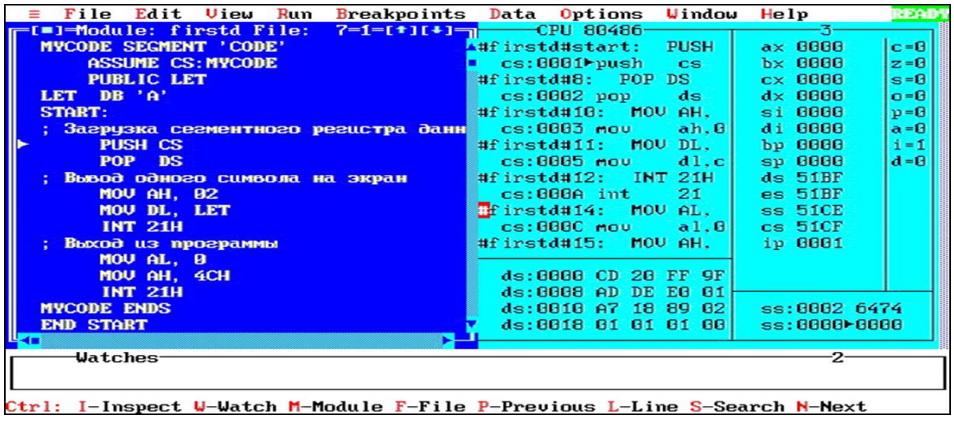
(Приводится перечень ошибок, которые проявились при отладке программы студентом и конкретизируются способы их устранения. Лучше в виде следующей таблицы с колонками:

№ п/п	Проявление ошибки	Причина ошибки	Способ устранения
	Синтаксис – ошибка написания команды: MOW вместо MOV	Ошибка ввода текста	Редактирование ntrcnf
	…	…	…

).

4. Скриншот программы ЛР № XX в TD.exe

(Вставляется скриншот собственной программы ЛР № XX, в котором видна первая строка с комментарием: ФИО, Группа, Вариант, ЛР №)



4. Блок-схема программы

(Блок схема оформляется в MS WORD в виде рисунка или MS VISIO в виде вставленного рисунка. Блок схема строиться для основной программы проекта)

5. Текст программы на языке Ассемблера

(Включается листинг программы выдаваемы компилятором Ассемблера, а не просто распечатка текста программы)

6. Результаты работы программы

(Вставляются результаты работы программы в виде текста или снятых с экрана скриншотов. Можно перекомпилировать программу в MS DOS версии BC, сменив кодировку, и получить результаты с помощью перенаправления потока в файл “>”)

7. Выводы по ЛР № X

(Формулируются выводы, которые были сформулированы при выполнении работы.)