

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Встраиваемые микропроцессорные системы»

Инструкция
по выполнению лабораторной работы
«Программирование на языке Ассемблер. Составление и отладка программ с командами логических операций»

Минск
2017

Лабораторная работа № 9

Тема работы: «Программирование на языке Ассемблер. Составление и отладка программ с командами логических операций»

1. Цель работы:

Формирование практических навыков по составлению и отладке программ с командами логических операций и сдвига.

2. Задание

Изучить команды логических операций и сдвига. Написать на языке ассемблера, выполнить и исследовать с помощью отладчика программу для работы с логическими командами и командами сдвига.

3. Оснащение работы

Техническое задание, ПК, эмулятор DOSBox.

4. Основные теоретические сведения

4.1. Описание работы

Для использования программы-отладчика Adfpro нужно скопировать в каталог с этой программой скопировать ассемблированную и скомпонованную программу в виде исполняемого модуля. Команда запуска отладчика имеет вид Adfpro имя_программы.

Работа с отладчиком Adfpro несложна и ведется с помощью меню и команд. При работе на экран можно выводить подсказку. Следует разобраться с отладчиком в рамках этого задания самостоятельно. Для выхода из отладчика используется команда quit.

4.2. Тексты программ-образцов

Примеры допустимых команд с непосредственными операндами – команд сдвигов и логических.

ПРОГРАММА-ОБРАЗЕЦ

```
TITLE      EXIMM      (EXE) Пример непосредственных операндов

DATASG     SEGMENT    PARA 'Data'
FLD1       DB         ?
FLD2       DW         ?
DATASG     ENDS

CODESG     SEGMENT    PARA 'Code'
BEGIN      PROC       FAR
            ASSUME     CS:CODESG,DS:DATASG

;          Ротация и сдвиг (только на 1 бит):
;          -----
            RCL        BL,1          ; ротация влево с переносом
            RCR        AH,1          ; ротация вправо с переносом
```

```

        ROL     FID2,1           ; ротация влево
        ROR     AL,1            ; ротация вправо
        SAL     CX,1            ; сдвиг влево
        SAR     BX,1            ; арифметический сдвиг вправо
        SHR     FLD1,1          ; сдвиг вправо

;           Логические операции:
;           -----
        AND     AL,00101100B     ;AND (регистр)
        OR      BH,2AH          ;OR (регистр)
        TEST    BL,7AH          ;TEST (регистр)
        OR      FLD1,23H        ;XOR (память)
BEGIN    ENDP
CODESG   ENDS
        END

```

4.3. Пример практического использования логических команд.

Программа: изменение строчных букв на прописные.

Существуют различные причины для преобразований между строчными и прописными буквами. Например, некая программа должна позволить пользователям вводить команды как прописными, так и строчными буквами (например, YES или yes) и преобразовать их в прописные для проверки. Прописные буквы от A до Z имеют шестнадцатеричные коды от 41 до 5A, а строчные – от 61 до 7A. Единственная разница в том, что пятый бит равен 0 для заглавных букв и 1 для строчных:

Биты: 76543210	Биты: 76543210
Буква A: 01000001	Буква a: 01100001
Буква Z: 01011010	Буква z: 01111010

Приведенная программа-образец преобразует данные в поле TITLX из строчных букв в прописные, начиная с адреса TITLX+1. Программа инициализирует регистр BX адресом TITLX+1 и использует его для пересылки символов в регистр AH, начиная с TITLX+1. Если полученное значение лежит в пределах от шест. 61 и до 7A, то команда AND устанавливает бит 5 в 0:

```
AND AH,11011111B
```

Все символы, отличные от строчных букв (от a до z), не изменяются. Измененные символы засылаются обратно в область TITLX, значение в регистре BX увеличивается для очередного символа и происходит переход на следующий цикл.

ИЗМЕНЕНИЕ СТРОЧНЫХ БУКВ НА ПРОПИСНЫЕ

```

TITLE    CASE      (COM) Перекодировка в заглавные буквы
CODESG   SEGMENT   PARA 'CODE'
        ASSUME     CS:CODESG,DS:CODESG,SS:CODESG
        ORG        100H
BEGIN:   JMP       MAIN
; -----
TITLX    DB         'Change to uppercase letters'

```

```

; -----
MAIN    PROC    NEAR
        LEA      BX,TITLEX+1    ; адрес первого символа
        MOV      CX,31          ; число символов
B20:
        MOV      AH,[BX]        ; символ из TITLEX
        CMP      AH,61H         ; прописная буква?
        JB       B30            ;
        CMP      AH,7AH         ;
        JA       B30            ;
        AND      AH,11011111B   ; да - преобразовать
        MOV      [BX],AH        ; записать в TITLEX
B30:
        INC      BX             ; следующий символ
        LOOP     B20            ; повторить цикл 31 раз
        RET
MAIN    ENDP
CODESG  ENDS
        END      BEGIN

```

Используемый таким образом регистр BX действует как индексный регистр для адресации в памяти. Для этих же целей можно использовать регистры SI и DI.

4.4. Типовое обрамление программ

О типовом обрамлении программ уже говорилось в теме "03 Двоичные арифметические вычисления – линейные программы". Но здесь используется другое обрамление – простейшее – ему соответствует файл EXE_MINI.ASM. Вот содержимое этого файла с некоторыми пояснениями.

```

TITLE PROGNAME (EXE)
;-----
STACKSG SEGMENT PARA STACK 'Stack'
        DW 32 DUP (?)
STACKSG ENDS
;-----
DATASG SEGMENT PARA 'Data'
;      ... |
;      ... |      Здесь будут нужные определения данных
;      ... |
        EXIT DW (?) ; выход по любой клавише
DATASG ENDS
;-----
CODESG SEGMENT PARA 'Code'
BEGIN PROC FAR
        ASSUME CS:CODESG,DS:DATASG,SS:STACKSG
        PUSH DS
        XOR   AX,AX          ; запись в стек
        PUSH AX              ; нулевого адреса
        MOV   AX,DATASG      ; засылка адреса
        MOV   DS,AX          ; DATASG в регистр DS
;-----
;      ... |
;      ... |      Здесь будет нужный программный код
;      ... |

```

```
RET                                ; завершение программы
BEGIN ENDP
CODESG ENDS
END BEGIN
```

Обрамление содержит:

- определение сегмента стека, оформленное директивами STACKSG SEGMENT и STACKSG ENDS; сам сегмент состоит из 32-х слов памяти, что задается директивой DW 32 DUP (?).
- определение сегмента данных, оформленное директивами DATASG SEGMENT и DATASG ENDS;
- определение сегмента кода, оформленное директивами CODESG SEGMENT и CODESG ENDS.

4.5. Подготовка программы к выполнению.

Первая программа-образец. Она имеет тип EXE. Текст этой программы нужно ввести в обрамление. Для ассемблирования программы нужна команда:

```
TASM.EXE /la /z /zi PROG.ASM
```

где PROG.ASM – условное имя исходной программы.

Для компоновки программы нужна команда:

```
TLINK.EXE PROG.OBJ, PROG-EXE
```

где PROG.OBJ – имя OBJ-модуля ассемблированной программы,

PROG-EXE – имя исполнимой EXE-программы (без расширения).

Вторая программа-образец. Программа для изменения строчных букв на прописные имеет тип COM. Такие программы организованы иначе. Их ассемблирование не имеет особенностей, но после ассемблирования программу надлежит преобразовать в COM-формат. Приведенный программный текст полностью готов для ассемблирования, содержит все необходимое и не требует включения в обрамления.

Для создания программы типа COM необходимо при вызове компоновщика добавочно ввести ключ /T:

```
TLINK.EXE /t PROG.OBJ, PROG-EXE
```

Если не сделать этого, будет получена программа типа EXE, но она окажется неработоспособной.

5. Порядок выполнения работы

1. Подробно разобраться в приведенных текстах программ-образцов.
2. Скопировать из указанного каталога текст обрамления программы LOG_FORM.ASM, изменив ему имя по усмотрению, в свой рабочий каталог.
3. Взяв за основу текст первой программы-образца, ввести его в обрамление со следующими изменениями:
 - заменить непосредственные операнды на операнды, вводимые как переменные в сегменте данных;
 - внести в текст программы команду, необходимую для занесения первого из этих операндов в регистр BL – для работы с командами сдвига;
 - использовать второй из операндов в команде AND.

4. Ассемблировать программу, скомпоновать ее и получить EXE-модуль.
5. Скопировать этот модуль в каталог, где находится adfpro.
6. Запустить отладчик для отладки и выполнения этой программы и разобраться с ее работой. Результаты выполнения программы можно видеть на экране отладчика.
7. Используя текст второй программы-образца, составить файл NAME.ASM, где вместо NAME использовать другое имя – по своему выбору.
8. Ассемблировать программу, скомпоновать ее и получить COM-модуль.
9. Скопировать этот модуль в каталог, где находится adfpro.
10. Запустить отладчик для отладки и выполнения этой программы и разобраться с ее работой. Результаты выполнения программы можно видеть на экране отладчика.

6. Форма отчета о работе

Лабораторная работа № _____

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Индивидуальное задание на работу

Указание имен исходного и исполняемого файлов

Результат выполнения работы: _____

Отчет представляется в виде текстового файла. К отчету должны прилагаться файл исходного кода Программы и рабочий исполняемый файл.

7. Контрольные вопросы и задания

1. Назовите формат команды «AND», ее операнды?
2. Назовите формат команды «OR», ее операнды?
3. Назовите формат команды «TEST», ее операнды?
4. Назовите формат команды «XOR», ее операнды?
5. Предположим, что регистр BL содержит 11100011 и поле по имени BOONO содержит 01111001. Определите воздействие на регистр BL для следующих команд:
 - а) XOR BL,BOONO;
 - б) AND BL,BOONO;
 - в) OR BL,BOONO;
 - г) XOR BL,11111111B;
 - д) AND BL,00000000B.

8. Рекомендуемая литература

Финогенов, К. Г. Основы языка Ассемблера [Текст] / К. Г. Финогенов. – М.: Радио и связь, 2000.

Финогенов, К. Г. Использование языка Ассемблера [Текст]: учеб. пособие для вузов / К.Г. Финогенов. – М.: Горячая линия Телеком, 2004.

Юров, В. И. Assembler [Текст]: учеб. пособие для вузов / В. И. Юров. 2-е изд. – СПб.: Питер, 2007.