Лабораторная работа N 1

СТРУКТУРА ПРОГРАММ НА ЯЗЫКЕ АССЕМБЛЕРА

Цель работы. Целью лабораторной работы является получение знаний об основных правилах записи и структуре программ на языке ассемблера для ПЭВМ типа IBM PC, а также приобретение практических навыков по составлению простейших программ на языке ассемблера, их выполнению на ЭВМ и отладке с использованием интерактивных отладчиков.

Основные теоретические сведения

Программы на языке ассемблера представляют собой последовательность операторов, описывающих выполняемые операции.

Операторы в свою очередь подразделяются на команды и директивы (или псевдооператоры, псевдокоманды).

Каждая команда порождает одну или несколько команд в машинных кодах. Директивы не порождают машинных команд, а лишь информируют транслятор с языка ассемблера об определенных действиях.

Формат команды языка ассемблера в общем случае содержит до 4 полей:

[метка:] команда [поле операндов] [;комментарий]

Обязательным является только поле команды. По крайней мере один пробел между полями обязателен. Максимальная длина строки - 132 символа.

Метка используется для указания адреса команды и служит для присвоения символического имени первому байту области памяти, в котором будет записан машинный эквивалент команды. Метки могут содержать:

- буквы от A до Z, причем ассемблер не делает разницы между строчными и прописными буквами;
- цифры от 0 до 9;
- спец.символы: '?', '.' (только первый символ), '@'(коммерческое эт), '$', '_'.

Первым символом в метке должна быть буква или спец.символ. Максимальная длина метки - 31 символ.

Поле команды содержит имя команды, например, MOV - команда пересылки, ADD - команда сложения и другие.

Операнды определяют:

- начальные значения данных;
- элементы, над которыми выполняется действие.

Например, команда

MOV AX, 4

заносит в регистр AX значение 4.

В поле операнда, если оно присутствует, может быть один или два операнда. Между собой операнды отделяются запятой. В командах с двумя операндами первый представляет собой приемник, а второй - источник. Источник никогда не изменяется, приемник изменяется почти всегда.

Комментарий начинается с символа ';' в любом месте программы. Все символы справа от символа ';' до конца строки воспринимаются ассемблером как комментарий.

Директивы могут иметь до 4-х полей:

[метка] директива [операнд] [;комментарий]

Наиболее употребительные директивы можно разделить на две группы: директивы данных и директивы управления листингом. Директивы данных в свою очередь делятся на пять групп:

- директивы определения идентификаторов;
- директивы определения данных;
- директивы внешних ссылок;
- директивы определения сегмента/процедуры;
- директивы управления трансляцией.

Существует две директивы определения идентификаторов: EQU и '='. EQU присвайвает имя выражению постоянно, а '=' - временно и его можно переопределить.

Существуют следующие директивы определения данных:

- DB - определить байт;
- DD - определить двойное слово;
- DW - определить слово;
- DT - определить 10 байт.

Обычно DB и DW резервируют память под переменные, а DD и DT - под адреса. В любом случае можно указать начальное значение или знак '?' только для резервирования памяти. Операция DUP позволяет повторять одно и тоже значение несколько раз. Например, директива

```
PEREM DB 4 DUP(?)
```

резервирует четыре байта в памяти под переменную PEREM.

Если перечислять значения переменной через запятую, то тем самым определяется таблица данных. Например, директива

```
TABL DW 5,7,9
```

выделяет область памяти с именем TABL длиной три слова и заносит в первое слово значение 5, во второе значение 7, в третье слово - значение 9.

Директивы определения сегмента делят исходную программу на сегменты. В программе на языке ассемблера возможно 4 вида сегментов:

- сегмент данных;
- сегмент стека;
- сегмент команд;
- дополнительный сегмент.

По крайней мере один сегмент - сегмент команд - должен присутствовать в каждой программе.

Существует две директивы определения сегмента: SEGMENT и ENDS, а также директива ASSUME, которая устанавливает для ассемблера соответствие между конкретными сегментами и сегментными регистрами. Например, директива

```
DATASG  SEGMENT PARA 'DATA'
```

определяет начало сегмента с именем DATASG, а директива

```
DATASG  ENDS
```

конец этого сегмента. Директива

```
ASSUME  CS:CODESG, DS:DATASG
```

указывает ассемблеру, что сегмент CODESG будет адресоваться с помощью сегментного регистра CS, а сегмент DATASG - с помощью сегментного регистра DS.

Директивы PROC и ENDP определяют процедуру. Процедура может иметь атрибут дистанции NEAR или FAR. Процедура с атрибутом NEAR может быть вызвана только из того сегмента команд, в

котором она определена, а процедура с атрибутом FAR может быть вызвана и из другого сегмента команд. Например директива

```
FUN PROC NEAR
```

определяет начало процедуры FUN с атрибутом дистанции NEAR, а директива

```
FUN ENDP
```

конец этой процедуры. Программа на ассемблере имеет атрибут FAR.

Директивы внешних ссылок PUBLIC и EXTRN делают возможным использование переменных и процедур, определенных в разных файлах.

Директива управления трансляцией END отмечает конец исходного модуля, поэтому она должна присутствовать в каждом модуле.

Директивы управления листингом PAGE и TITLE могут быть использованы для определения формата вывода распечаток программ и выдачи заголовков.

Таким образом, типовая структура программы на языке ассемблера для ПЭВМ типа IBM PC выглядит следующим образом:

```
        TITLE заголовок программы
        PAGE 60,132
;-------------------------------------------------
; Определение сегмента стека
STACKSG  SEGMENT PARA STACK 'STACK'
        DB    64 DUP(?) ; область стека, не менее 32 слов
STACKSG  ENDS
;-------------------------------------------------
; Определение сегмента данных
DATASG   SEGMENT PARA 'DATA'
; здесь поместить, если необходимо, директивы определения
; данных
        ...
DATASG   ENDS
;-------------------------------------------------
; Определение сегмента команд (основная программа)
CODESG   SEGMENT PARA 'CODE'
ASSUME   CS:CODESG, DS:DATASG, SS:STACKSG
```

```
ENTRY   PROC FAR
; Инициализировать программу
        PUSH  DS          ; сохранить в стеке адрес возврата
        SUB   AX,AX       ; обнулить регистр AX
        PUSH  AX          ; занести в стек нулевое
                          ; смещение для адреса возврата
; Инициализировать адрес сегмента данных
        MOV   AX,DATASG   ; занести адрес сегмента
        MOV   DS,AX       ; данных в регистр DS
; Программа
        ...
        RET               ; вернутся в DOS
ENTRY   ENDP
CODESG  ENDS
        END   ENTRY
```

В данном фрагменте показана группа команд PUSH DS ... MOV DS,AX, которая является стандартной для программ на ассемблере и обеспечивает возврат управления в DOS после выполнения программы. Команда RET обеспечивает выход из программы и передачу управления DOS.

Пример программы на языке ассемблера приведен ниже. В программе определены 3 сегмента:

STACKSG - сегмент стека. Оператор DB отводит под стек 64 байта (32 слова), не присваивая им начальных значений. Стек будет содержать адрес возврата в DOS.

DATASG - сегмент данных. В этом сегменте с помощью директивы определения данных DW определяется переменная PP, которой присваивается начальное значение - 0123h.

CODESG - сегмент команд. В этом сегменте содержатся выполняемые команды программы.

Директивы SEGMENT - ENDS определяют начало и конец сегмента.

Директива ASSUME назначает регистр CS для адресации сегмента команд, регистр DS - сегмента данных, регистр SS - сегмента стека.

Директива PROC определяет имя ENTRY как точку входа в

основную программу из DOS (начало программы). Следующая группа команд PUSH DS ... MOV DS,AX является стандартной, команда RET обеспечивает выход из программы и передачу управления DOS. Директивы ENDP, ENDS, END заканчивают программу.

```
TITLE   FIRST
STACKSG SEGMENT PARA STACK 'STACK'
        DB    64 DUP(?)
STACKSG ENDS
DATASG  SEGMENT PARA 'DATA'
PP      DW    0123h
DATASG  ENDS
CODESG  SEGMENT PARA 'CODE'
ASSUME  CS:CODESG, DS:DATASG, SS:STACKSG
ENTRY   PROC  FAR
        PUSH  DS
        SUB   AX,AX
        PUSH  AX
        MOV   AX,DATASG
        MOV   DS,AX
        MOV   AX,PP       ; записать 0123 в AX
        ADD   AX,0025h    ; прибавить 25 к AX
        MOV   BX,AX       ; переслать AX в BX
        ADD   BX,AX       ; прибавить AX к BX
        MOV   CX,BX       ; переслать BX в CX
        SUB   CX,AX       ; вычесть AX из CX
        SUB   AX,AX       ; очистить AX
        RET
ENTRY   ENDP
CODESG  ENDS
        END   ENTRY
```

Выполнение программ на ассемблере

1. Построение исполнимого файла

Для создания программ на ассемблере необходимо выполнить следующие действия:

1) С использованием какого-либо текстового редактора ввести текст программы в ЭВМ и запомнить его в файле с расширением .ASM (например, PROG.ASM).

2) Выполнить трансляцию программы. Для этого необходимо набрать команду

```
TASM PROG, PROG, PROG / Z
```

В результате будет создан объектный файл под именем имя_программы.OBJ, файл листинга имя_программы.LST, файл перекрестных ссылок имя_программы.CRF.

3) Построить исполняемый .EXE файл. Для этого вызвать компоновщик TLINK командой

```
TLINK имя_программы.OBJ
```

В результате на диске будет построен исполняемый файл имя_программы.EXE и листинг распределения памяти имя_программы.MAP.

Просмотр листинга программы можно осуществить стандартными средствами персонального компьютера в файле имя_программы.LST.

Кроме листинга программы можно получить:

- листинг таблицы перекрестных ссылок;
- листинг распределения памяти.

В таблице перекрестных ссылок указывается номер строки, в которой определен каждый идентификатор, и номера тех строк, в которых есть ссылки на него.

Листинг распределения памяти содержит сводные сведения о сегментах программы и находится в файле имя_программы.MAP. Для каждого сегмента указывается адрес начала и конца, длина сегмента в байтах, его имя и категория.

2. Выполнение программы

Выполняемый .EXE файл можно вызвать как из DOS, так и выполнить под управлением отладчика TD (Turbo Debbuger).

Вызов программы из DOS осуществляется стандартным образом.

Последовательность действий при отладке программ под управлением интерактивного отладчика TD приведена в приложении.

Задание на лабораторную работу

1. Изучить структуру программы на ассемблере и основные требования к ней.

2. Разработать простую программу обмена данными между двумя одномерными таблицами, причем данные в таблице-"приемнике" должны размещаться в порядке, обратном таблице-"источнику". Для этого:

1) С использованием директивы DB определить в сегменте данных две таблицы с именами TABL1 и TABL2 по 4 байта каждая для таблицы-"источника" и таблицы-"приемника" соответственно. При этом таблица TABL1 должна иметь начальные значения.

2) Обнулить таблицу TABL2 командами:

```
MOV TABL2,0     ; обнулить первый байт
MOV TABL2+1,0   ; обнулить второй байт
и т.д.
```

3) Скопировать таблицу TABL1 в TABL2 последовательностью команд:

```
MOV AL,TABL1    ; переслать в регистр AL первый
                ; байт таблицы TABL1
MOV TABL2+3,AL  ; переслать в четвертый байт
                ; таблицы TABL2 содержимое
                ; регистра AL
и т.д.
```

3. С помощью текстового редактора ввести программу в ЭВМ и записать ее на диск. Оттранслировать программу и построить исполнимый файл. Распечатать листинг программы и изучить его структуру.

4. Выполнить программу под управлением отладчика TD. Провести пошаговую трассировку программы, распечатывая на каждом шаге содержимое сегмента данных. Изучить изменение указателя стека SP, командного указателя IP и сегмента данных от шага к шагу выполнения программы.

5. Завершить выполнение программы под управлением отладчика TD.