1. Разработка и выполнение программы

В этой главе мы рассмотрим, как можно записать, оттранслировать и выполнить программу на языке ассемблера.

Ассемблер — это специальная программа, осуществляющая перевод (трансляцию) программы на языке ассемблера в программу в машинных кодах. Поэтому любой ассемблер можно назвать и транслятором.

Программа записывается на диск с помощью текстового редактора, затем транслируется ассемблером в коды, после этого загружается в память компьютера и выполняется.

Если этих средств нет, то иногда можно оттранслировать программу на языке ассемблера вручную и записать в память машины полученные коды с помощью специальных операторов языка MSX BASIC.

Команда на языке ассемблера — это, как правило, мнемоническая запись одной команды микропроцессора (машинной команды).

Кроме команд в тексте программы на языке ассемблера могут встретиться и директивы ассемблеру. Директивы обычно не транслируются, а являются указанием ассемблеру выполнить определенное действие.

Для микропроцессора Z80 применимы ассемблеры DUAD, M80, GEN80, Роботрон-1715 и другие. Кроме различных качественных характеристик они имеют различающиеся наборы директив.

Оттранслированная машинная программа обычно может быть представлена в одном из двух форматов — `.OBJ` или `.COM`. Программы типа `.OBJ` загружаются и выполняются в среде MSX BASIC и обычно транслируются с адреса 8000h и выше, для запуска используется команда:

```
BLOAD"PROG.OBJ",r
```

.

Программы типа `.COM` работают как задачи операционной системы MSX-DOS и размещаются с адреса 100h.

Разные трансляторы могут давать машинный код для микропроцессоров разных типов, например для Z80 или для Intel 8080.

1.1. Редактирование текста программы

Итак, мы хотим написать программу. Для этого надо загрузить текстовый редактор (например: TOR, MIM, SCED, TED или другой). Это можно сделать в MSX-DOS, вставив диск с редактором в дисковод и набрав в ответ на приглашение DOS имя редактора:

```
A>ted
```

После загрузки редактора можно начинать набирать программу.

Если мы хотим получить файл типа `.OBJ`, то можно воспользоваться ассемблером DUAD. Пример программы для этого транслятора приводится ниже.

Укажем адрес, с которого нужно оттранслировать нашу программу и с которого она будет загружена в память компьютера. Адрес может быть в интервале от 8000h до DE76h для машины учителя и 8000h - F37Fh для машины ученика. Это можно сделать директивой ассемблера ORG. Например,

```
ORG  9000h
```

При написании программ поле экрана обычно распределяют следующим образом:

- первые восемь позиций отводится под метки;
- вторые восемь под команду;
- следующие восемь под операнды;
- все остальное место под комментарии.

Директиву ORG нужно разместить во втором столбце, а адрес — в третьем. При наборе удобно пользоваться клавишей табуляции TAB.

В программу в любом месте можно вставлять комментарии. Комментарий должен начинаться с символа ';' (точка с запятой). После комментария не может стоять команда (т.е. вся строка после ';' считается комментарием). Комментарии не транслируются ассемблером ни в какой код, а служат для пояснения смысла программы другим программистам или его напоминания самому автору.

Вставим в нашу программу комментарий и напишем несколько команд и директив:

```
        ORG     9000h
; установили начальный адрес компиляции
        LD      b,40h   ; загрузили в регистр b 40h
beg:    DEC     b       ; b <= b-1
        JR      NZ,beg  ; если не 0 - перейти на метку beg
        RET             ; иначе возврат
; конец программы
        END
```

В конце программы ставится директива END — указание ассемблеру, что дальше транслировать не надо или нечего.

Если разрабатывается не главная программа, а подпрограмма, например, вызываемая из MSX BASIC, то в конце подпрограммы необходимо поставить команду возврата в место вызова — RET.

После того как программа написана, её необходимо записать на диск. Желательно присвоить ей расширение `.ASM` или `.MAC`, указывающее, что это исходный текст программы на языке ассемблера или макроассемблера, например `example.ASM` или `keys.MAC`.

Затем программу можно оттранслировать ассемблером.

1.2. Ассемблирование программы

В примерах мы будем использовать ассемблер системы DUAD фирмы ASCII или макроассемблер M80 (Вы можете использовать любой другой). Название ассемблера можно увидеть в первых строках листинга программы.

1.2.1. Ассемблирование в системе DUAD

Как уже говорилось, если мы хотим получить файл типа `.OBJ`, можно воспользоваться ассемблером DUAD.

Для вызова системы DUAD необходимо перейти в режим MSX BASIC и набрать команду:

```
run"DUAD.BAS"
```

Затем Вы должны выбрать в «меню» Assembler, нажав клавишу `1`. При правильном запуске ассемблера Вам будут заданы вопросы. В ответ на них Вы должны ввести имя файла или нажать `Ввод⏎`, если вам соответствующее действие не нужно. Системой задаются следующие вопросы:

```
Source   : [ файл - исходный текст ]
List     : [ файл для записи листинга ]
Object   : [ файл для записи оттранслированной программы ]
Label A  : [ файл для меток и адресов по алфавиту]
Label L  : [ файл для меток и адресов по возрастанию адресов ]
Cross ref: [ файл перекрестных ссылок ]
```

На первый вопрос ответ обязателен, на другие — нет. Например, можно ответить так:

```
Source   : example.asm
List     : example.lst
Object   : example.obj
Label A  :
Label L  :
```

```
Cross ref:
```

После этого ассемблер начнет трансляцию и через некоторое время выдаст сообщение «No errors in text» (Нет ошибок в тексте). Это значит, что наша программа написана без ошибок и успешно оттранслирована. Если будут другие сообщения, то транслирование не удалось (причиной может быть ошибка в программе, защита на диске или еще что-то).

Для выхода из системы DUAD на вопрос «Source:» введите

```
*BASIC
```

Если в программе были ошибки, можно просмотреть с помощью текстового редактора файл с расширением «.LST», например,

```
A>TOR example.lst
```

Листинг нашей программы, если все было сделано правильно, будет иметь вид:

```
            Z80-Assembler   Page:    1
                    ORG     9000h
   ; установили начальный адрес компиляции
9000 0640           LD      B,40h   ; загрузили в регистр B 40h
9002 05      BEG:   DEC     B       ; B <= B-1
9003 20FD           JR      NZ,BEG  ; если не 0 - перейти на BEG
9005 C9             RET             ; иначе возврат
   ; конец программы
                    END
No errors in text
```

Колонка слева — шестнадцатеричные адреса машинных команд (9000, 9002, 9003, 9005). Рядом с адресом — машинный код команды. Таким образом наша программа в машинных кодах выглядит так:

```
06400520FDC9
```

1.2.2. Ассемблирование посредством M80

С помощью ассемблера M80 можно получить программы типа `.COM`. Текст программы, которую мы набирали выше, будет иметь несколько отличающийся вид:

```
       .Z80
       LD      B,40h   ; load 40h  in B
BEG:   DEC     B       ; B <= B-1
       JR      NZ,BEG  ; if not 0 - go to BEG
       RET             ; else return
; end of program
       END
```

Первая директива этой программы сообщает ассемблеру, что команды записаны в соответствии с мнемоникой Z80.

После записи текста программы в файл с расширением `.ASM`, не выходя из MSX-DOS в MSX BASIC, наберите команду вида:

```
A>M80 =example.asm/L
```

Листинг программы `example.ASM` будет записан в файл `example.PRN`. Результат трансляции будет записан в файл `example.REL` (который затем должен быть обработан редактором связей и преобразован в файл типа `.COM`).

Более полная форма вызова ассемблера M80 выглядит так:

```
A>M80 example,example=example.asm/L
```

Здесь первое имя — имя результирующего файла, второе — для листинга, третье — текст на ассемблере.

Если не нужен ни машинный код, ни листинг, пишут:

```
A>M80 ,=example.asm
```

Если нужен только листинг, то так:

```
A>M80 ,example=example.asm/L
```

Если исходный файл имеет расширение MAC, то в командах это расширение можно опускать.

При трансляции могут быть использованы следующие ключи:

Ключ	Описание
/O	печатать восьмеричные адреса и числа в восьмеричном виде
/P	создание стека для ассемблера
/R	имя объектного файла как у исходного
/X	установка STFCOND
/L	вывод листинга в файл типа PRN с именем как у исходного
/C	генерировать файл перекрестных ссылок с именем как у исходного и расширением CRF
/Z	используется мнемоника Z80
/I	используется мнемоника Intel 8080
/H	печатать шестнадцатеричные адреса и числа в шестнадцатеричном виде
/M	инициализировать память для директивы DS нулями

Для создания файла типа COM вызывается редактор связей L80.

1.3. Редактирование связей и сборка программы

Редактирование связей и сборка программы выполняются после трансляции ассемблером M80. Они осуществляются при помощи редактора связей L80. На этом этапе объединяются воедино все разрозненные части программы, записанные в различных REL–файлах или библиотеках. Могут также быть подключены библиотеки стандартных или дополнительных функций языка C или других языков. Результатом редактирования является неперемещаемый объектный код программы, записываемый в файл с расширением .COM.

Задание на редактирование программы, написанной на языке ассемблера, как правило, выглядит следующим образом:

```
A>L80 имя,имя/n/e
```

Здесь имя — это имя REL–файла, оттранслированного ассемблером M80.

Задание на редактирование программы, в которой управляющим модулем является функция main, написанная на языке C, обычно производится следующим образом:

```
A>L80 ck,Осн-Имя,Имя-Файла1[/s],Имя-Файла2[/s],...,clib/s,crun/s,cend,Имя-COM-файла/n/e:xmain
```

Здесь Осн-Имя — это имя REL–файла, содержащего код функции main. REL–файлы ck, clib, crun и cend входят в комплект ASCII C и их подключение обязательно для пользователя, избегающего вникать в тонкости структуры компилятора языка C.

Пользователь должен подключить все REL–файлы, содержащие модули, необходимые для разрешения внешних ссылок.

Параметры редактора связей L80:

Ключ	Описание
/s	подключить не всю библиотеку, а только необходимые модули
/p	установить начальный адрес размещения программы в памяти
/d	установить адрес размещения сегмента данных
/u	выдать список неразрешенных ссылок
/m	выдать адреса глобальных имен
/n[:имя]	записать COM–файл на диск[,установить точку входа в программу]
/g[:имя]	выполнить программу [с указанной точки входа]
/e[:имя]	выйти в DOS [,установить точку входа в программу]

Редактор связей создаёт программу, загружающуюся и стартующую с адреса 100H.

Чтобы создать программу, загружающуюся с адреса, отличного от 100H, необходимо не только использовать параметры /p, но и указать имя точки входа в параметрах /n или /e.

1.4. Выполнение программы

В случае успешной трансляции мы можем выполнить нашу программу. Для запуска программы типа .OBJ надо выйти в MSX BASIC и загрузить программу командой:

```
BLOAD"example.obj",R
```

Буква R обозначает «выполнить». Машина тут же должна выдать Ok. Так как наша программа уже загружена, её можно выполнить снова. Для этого надо определить её как функцию и передать ей управление.

```
DEFUSR = &H9000: I = USR(0)
```

Машина снова выдаст Ok.

Как Вы могли понять, программа написанная нами выше — не что иное, как обыкновенная задержка во времени. Но надо сказать, что эту задержку при выполнении Вы не заметите. Это объясняется большой скоростью выполнения программ на языке ассемблера. Поэтому для написания задержек обычно используют пару регистров.

Это мы рассмотрим несколько ниже, а сейчас попробуем написать эту же программу, но «оттранслировав» самостоятельно и записав через MSX BASIC (можно предварительно загрузить в текстовый редактор листинг нашей программы и посмотреть, как она оттранслирована):

```
10 DATA 06,40        :REM LD   B,40    9000
20 DATA 05           :REM DEC  B       9002
30 DATA 20,FD        :REM JR   NZ,9002 9003
40 DATA C9           :REM RET          9005
100 REM загружаем коды в память
110 DATA Z
120 N=&H9000
130 READ A$:IF A$<>"Z" THEN POKE N+I,VAL("&H"+A$):I=I+1:GOTO 130
140 DEFUSR=N:I=USR(0)
150 END
```

Если Вы не уверены, что оттранслировали правильно, то загрузите в текстовый редактор листинг (расширение LST) и сверьте коды.

Запустив эту программу, Вы получите тот же результат, что и при запуске файла OBJ, однако постоянного обращения к диску уже не требуется.

Как уже говорилось выше, программу типа .REL нужно обработать редактором связей, чтобы получить соответствующую программу типа .COM.

Для запуска программы с расширением .COM наберите в режиме MSX-DOS её имя без расширения:

```
A>example
```

Итак, если у Вас все получилось, поздравляем Вас с выполнением Вашей первой программы на языке ассемблера Z80 !!!

1.5. Организация связей с программами на языке MSX BASIC

При разработке подпрограмм, написанных в кодах, которые должны вызываться из программ на языке MSX BASIC, часто возникает проблема передачи параметров в подпрограмму и получения результата из подпрограммы.

Для осуществления этого возможны два основных способа — использование общей памяти и собственно передача/получение параметров. Может использоваться и комбинация этих способов.

1.5.1. Общая память

В этом случае выделяется одна или несколько ячеек памяти с заранее известными адресами, и программы обмениваются данными через эти ячейки. На языке MSX BASIC для этого используются оператор POKE и функция PEEK.

Пример программы на языке MSX BASIC.

```
10 CLEAR 200,&H9000      : REM установка границ
20 DEFUSR = &H9000       : REM адрес подпрограммы
30 BLOAD "example.obj"   : REM загрузка с диска
40 X = 124               : REM число для передачи
50 POKE &HA000,X         : REM запись аргумента
60 I = USR(0)            : REM вызов подпрограммы
70 Y = PEEK(&HA001)      : REM берем результат
80 PRINT Y
90 END
```

Листинг вызываемой программы на языке ассемблера:

```
'comm. memory '   Z80-Assembler   Page:    1
                        TITLE   'comm. memory '
                        ORG     9000h
9000 3A00A0             LD      A,(0A000h) ; берем аргумент
9003 3C                 INC     A          ; увеличить A
9004 3C                 INC     A          ; еще раз
9005 3201A0             LD      (0A001h),A ; записываем результат
9008 C9                 RET                ; возврат
                        END
'comm. memory '   Z80-Assembler   Page:    2
No errors in text
```

В результате работы MSX BASIC–программы с этой подпрограммой должно получиться число 126.

Достоинством этого способа передачи данных является возможность передачи и получения из подпрограммы больших массивов информации.

1.5.2. Передача и получение параметров

При вызове подпрограммы, определенной как USR, интерпретатор языка MSX BASIC записывает в регистр HL адрес арифметической переменной, в DE — ссылку на адрес строкового выражения, а в аккумулятор A и ячейку &hF663 — тип переменной. При этом для значений целого типа адрес в HL нужно увеличить на 2, а первые три байта строкового указателя в DE хранят длину и реальный адрес строки.

Иногда удобнее воспользоваться готовыми подпрограммами передачи и преобразования данных.

Для передачи однобайтного аргумента из программы на языке MSX BASIC подпрограмме в кодах через регистр A используется подпрограмма ПЗУ по адресу &h521F.

Для передачи двухбайтного аргумента из MSX BASIC используется подпрограмма с начальным адресом &h2F8A. Она записывает аргумент в регистр HL.

Для передачи двухбайтного результата из подпрограммы в кодах в MSX BASIC используется подпрограмма &h2F99. Она возвращает значение, записанное в HL.

Рассмотрим примеры.

Передача однобайтного аргумента и получение результата.

Программа на MSX BASIC:

```
10 CLEAR 200,&H9000      : REM установка границ
20 DEFUSR = &H9000       : REM адрес подпрограммы
30 BLOAD "example.obj"   : REM загрузка с диска
40 X = USR(45)           : REM вызов подпрограммы
50 PRINT X               : REM возврат значения в  X
60 A = 73
70 X = USR(A)            : REM вызов подпрограммы
80 PRINT X
90 END
```

Нельзя в качестве аргумента использовать значение не в диапазоне 0..255. Например, значение 260 вызовет ошибку. Листинг программы `example.asm`:

```
'pass one-byte'   Z80-Assembler   Page:    1
                        TITLE   'pass one-byte'
                        ORG     9000h
       ; --- записать аргумент в A
9000 CD1F52             CALL    521Fh
       ; --- обработка аргумента
9003 3C                 INC     A       ; увеличить A
9004 3C                 INC     A       ; еще раз
       ; --- возврат результата через HL
9005 2600               LD      H,0
9007 6F                 LD      L,A
9008 C3992F             JP      2F99h   ; возвращаем результат
                        END
```

После выполнения программы будут напечатаны числа 47 и 75.

Передача двухбайтного аргумента и получение результата.

Программа на MSX BASIC:

```
10 CLEAR 200,&H9000      : REM установка границ
20 DEFUSR = &H9000       : REM адрес подпрограммы
30 BLOAD "example.obj"   : REM загрузка с диска
40 X = USR(1045)         : REM вызов подпрограммы
50 PRINT X
60 A = 23678
70 X = USR(A)            : REM вызов подпрограммы
80 PRINT X
90 END
```

Листинг программы `example.asm`:

```
'pass two-byte'   Z80-Assembler   Page:    1
                        TITLE   'pass two-byte'
                        ORG     9000h
       ; --- записать аргумент в  HL
```

```
9000 CD8A2F             CALL    2F8Ah   ; перед. 2-х байт. пар.
        ; --- обработать аргумент
9003 23                 INC     HL      ; увеличить  HL
9004 23                 INC     HL      ; еще раз
        ; --- возврат результата через  HL
9005 C3992F             JP      2F99h   ; возвращаем результат
                        END
'pass two-byte'   Z80-Assembler   Page:    2
```

После выполнения программы будут напечатаны числа 1047 и 23680.

Как Вы заметили, программы на языке MSX BASIC отличаются фактически только допустимым диапазоном передаваемых функции USR значений. Отличен же способ их обработки в подпрограмме: вызов &h2F8A или &h521F.

1.6. Организация связей с программами на языке C

Основные вопросы, которые нужно иметь в виду при организации связей программ, написанных на языке C с программами на языке ассемблера, — это порядок передачи параметров, правила написания имен и редактирование связей.

1.6.1. Передача параметров

Передача параметров для функций языка C с фиксированным числом параметров подчинена следующему соглашению о связях: Первый параметр передается:

- через регистр A, если он занимает один байт;
- через пару HL, если занимает два байта.

Второй и третий параметры передаются соответственно через регистры E и C или через DE и BC. Остальные параметры записываются в стек, по 2 байта на параметр независимо от типа. В случае функции с переменным числом параметров количество параметров передается через HL, все параметры записываются в стек по два байта независимо от типа. Результат любой функции помещается в A (char) или в HL (int и другие типы). Параметры из стека убирает вызывающая функция. При выходе из вызываемой функции значение SP должно быть равным значению при входе.

1.6.2. Символические имена

Транслятор языка MSX-C распознает первые 16 символов имён. Внешние имена распознаются по первым 6 символам. При флаге `-l cg` в полученном тексте на ассемблере будет более 6 символов. Отметим, что ASCII C заменяет при компиляции символ «_» на «@», а к каждому символическому имени, являющемуся внешней ссылкой или глобальной переменной и состоящему менее чем из 6 символов, приписывает справа тот же символ «@». Пользователь, пишущий программы на языке ассемблера, должен это учитывать, чтобы правильно обращаться к C-функциям, и чтобы к его процедурам можно было обращаться из C-программ. например, пусть функция на языке C имеет заголовок

```
char a_fun (arg1,arg2)
int arg1,arg2;
```

Чтобы обратиться к ней из программы на языке ассемблера, следует написать:

```
LD   hl,...
LD   de,...
CALL a@fun@
LD   (...),a
...
```

Очевидно, что к программам на языке ассемблера, имена которых содержат символ «_» или не имеют на конце «@» при длине менее 6 байт, доступ из C-программ невозможен. Все глобальные переменные при трансляции C-программы в ассемблерный текст получают описатель PUBLIC, а внешние ссылки — описатель EXTRN. Такое же соглашение работает в ассемблере M80.

1.6.3. Трансляция и сборка разноязыковых модулей

Возможно создание программ, в которых часть модулей написана на языке ассемблера, а часть — на языке С. Чаще всего основу составляют программы на языке С, в том числе и главный модуль, а «тонкие» вещи делаются на языке ассемблера.

Допустим, мы разработали следующие подпрограммы на языке ассемблера:

```
      MSX.M-80 1.00 01-Apr-85    PAGE    1
                                .Z80
                                PUBLIC KillBf@
                                PUBLIC InKey@
                                PUBLIC Wait@
; === чистка буфера клавиатуры
0000'   F7           KillBf@: RST  30h
0001'   00                    DB   0
0002'   0156                  DW   156h
0004'   C9                    RET
; === ввод кода нажатой клавиши
; === выход: [a]=0, если ничего не нажато,
; ===     иначе - [a] - код нажатой клавиши
0005'   AF           InKey@:  XOR  A      ; чистим A
0006'   F7                    RST  30h    ; нажато что-нибудь ?
0007'   00                    DB   0
0008'   009C                  DW   9Ch
000A'   C8                    RET  Z      ; если нет - выход
000B'   F7                    RST  30h    ; иначе берем код
000C'   00                    DB   0
000D'   009F                  DW   9Fh
000F'   C9                    RET
; === ожидание нажатия клавиши
; === вход [a] - код символа, который нужно ждать
0010'   CD 0000'     Wait@:   CALL KillBf@
0013'   47                    LD   B,A
0014'   CD 0005'             CALL InKey@
0017'   B8                    CP   B
0018'   20 FA                JR   NZ,$-4
001A'   C9                    RET
                              END
```

Если эти подпрограммы записать в файл `keys.MAC`, то получить из него файл типа `REL` можно командой:

```
 A>m80 =Keys.mac/L
```

Теперь приведем исходный текст программы на языке С, вызывающей эти подпрограммы на языке ассемблера.

```
  #include <stdio.h>
  main()
{
  VOID KillBf();  /* описания подпрограмм */
  VOID Wait();
  char InKey();
  KillBf();       /* чистим буфер и ждем нажатия любой клавиши */
  while(InKey() == '\0');
  printf("%s\r\n\n"," .... ");
  Wait(' ');      /* ждем нажатия пробела */
  printf("%s\r\n"," ::::");
}
```

Для трансляции и редактирования файлов `exam.C` и `keys.REL` можно выполнить следующие команды:

```
cf -me exam
cg exam
```

```
m80 =exam.asm
l80 ck,exam,keys,clib/s,crun/s,cend,exam/n/e:xmain
exam
```

---

http://sysadminmosaic.ru/msx/assembler_programming_guide-fakhrutdinov_bocharov/01

2023-02-04 22:21