

Приложение 1. MSX BASIC и машинный язык

Покажите, как операторы языка ассемблера можно чередовать с операторами данного языка.

—Д.Уолш. Руководство по созданию документации для математического обеспечения

1.1. Связь программы на MSX BASIC с подпрограммами в машинных кодах



Перед выполнением оператора **MSX BASIC** выполняется проверка его правильности, и в случае обнаружения ошибок на экран выводятся сообщения. Выполнимый оператор сначала преобразуется в последовательность команд, которые может «понять» процессор. Эти подпрограммы заранее помещены в ROM на этапе изготовления компьютера.

Интерпретация операторов **MSX BASIC** занимает определенное время; в некоторых случаях, однако, это время должно быть сокращено (например, для игровых программ). Это можно сделать двумя способами:

1. основная программа пишется на **MSX BASIC**, а в ситуациях, когда интерпретатор **MSX BASIC** не обеспечивает нужного быстродействия, используются подпрограммы, написанные в машинных кодах;
2. вся программа пишется на машинном языке.

Отметим, что существует обширная литература по этому кругу проблем, и читатель, овладевший языком **MSX BASIC** и желающий научиться хорошо программировать на машинном языке, может обратиться к этой литературе [[45](#)], [[66](#)], [[67](#)] .

Для обращения к подпрограммам в машинных кодах из программ на языке **MSX BASIC** используются оператор **DEFUSR** и функция **USR**.

Оператор **DEFUSR** определяет начальный адрес машинной подпрограммы.

Синтаксис оператора:

```
DEFUSR[n] = адрес ,
```

где:

- **DEFUSR**(«**DE**Fine **US**ing Routine«-»определить используемую подпрограмму») — служебные слова;
- **n** — целое число, принадлежащее [0,9] и указывающее номер подпрограммы (по умолчанию значение **n** равно 0);
- **адрес** — арифметическое выражение, целая часть значения которого определяет начальный адрес подпрограммы.

Таким образом, в программе на языке **MSX BASIC** может использоваться до десяти подпрограмм в машинных кодах одновременно. Они различаются по номеру (от 0 до 9), который записывается вслед за служебным словом **DEFUSR**.

Например, оператор **DEFUSR1=&HA000** указывает, что начальным адресом программы с номером 1 является **&HA000** .

Функция

```
USR[n] (α) ,
```

где:

- **USR** («**US**er«-»пользователь») — служебное слово;

- n — десятичная цифра (0,1,2,3,4,5,6,7,8,9), причем по умолчанию $n=0$;
- α — арифметическое или строковое выражение (которое, кстати, может отсутствовать), позволяет *выполнить* подпрограмму, написанную на машинном языке.

Пример 1.

Пусть, например, одна машинная подпрограмма начинается с адреса &HD000, а другая - с адреса &HD100 :

```
DEFUSR0=&HD000:DEFUSR1=&HD100
```

Теперь выполнить эти подпрограммы можно при помощи операторов:

```
A=USR0(A):A=USR1(A)
```

Значение выражения α передается подпрограмме, написанной в машинных кодах. При вызове подпрограммы число байтов, занимаемое этим значением, заносится в регистр A микропроцессора Z80 и в ячейку памяти с адресом &HF663 .

Содержимое регистра A	Тип значения выражения α
2	Целочисленный
4	Вещественный с одинарной точностью
8	Вещественный с двойной точностью
3	Строковый

Если аргументом функции USR является *нуль* , то в подпрограмму в машинных кодах ничего не передается!

Отметим, что значение, возвращаемое функцией USR, представляет собой значение ее аргумента.

Если α — арифметическое выражение, то адрес его значения хранится в регистре HL .

Если же α — строковое выражение, то ссылка на адрес, по которому хранится значение α , содержится в регистре DE.

Ниже мы расскажем Вам и о других способах передачи аргументов машинным подпрограммам.

Примеры

Некоторые люди учатся читая, другие - делая.
Преуспевающие делают и то, и другое.

—П.Браун

Примеры, немногочисленные, лаконичные, приведенные к месту, сообщают рассуждению блеск, глубину и убедительность, но оно становится невразумительным, если примеров и подробностей чересчур много...

—Л.К.де Вовенарг. Размышления и максимы

Пример 1. Загрузка регистров A и BC константами.

[10101-01.bas](#)

 [10101-01.bas](#)

```
10 CLEAR 200,&HD000:DEFUSR=&HD000
20 DATA 3E,C3      : 'LD  A,C3h
30 DATA 01,A1,56   : 'LD  BC,56A1h
40 DATA 32,10,D0   : 'LD  (D010h),A
50 DATA ED,43,11,D0 : 'LD  (D011h),BC
```

```

60 DATA C9,"RET"      : 'RET
80 READ A$
90 IF A$="RET" THEN 120
100 POKE &HD000+T,VAL("&h"+A$)
110 T=T+1:GOTO 80
120 A=USR(A)
130 PRINT "A=";HEX$(PEEK(&HD010))
140 PRINT "BC=";HEX$(PEEK(&HD012));HEX$(PEEK(&HD011))
run
A=C3
BC=56A1
Ok

```

Пример 2. Пересылка содержимого регистра C в регистр A.

[10101-02.bas](#)

 [10101-02.bas](#)

```

10 CLEAR 200,&HD000:DEFUSR=&HD000
20 DATA 0E,C3          : 'LD  C,C3h
30 DATA 79             : 'LD  A,C
40 DATA 32,10,D0       : 'LD  (D010h),A
50 DATA C9,"RET"       : 'RET
70 READ A$:IF A$="RET" THEN 110
90 POKE &HD000+T,VAL("&h"+A$)
100 T=T+1:GOTO 70
110 A=USR(A)
120 PRINT "C=A= ";HEX$(PEEK(&HD010))
run
C=A= C3
Ok

```

Пример 3. Загрузка содержимого регистра A в память по адресу, указанному в регистре BC.

[10101-03.bas](#)

 [10101-03.bas](#)

```

10 CLEAR 200,&HD000:DEFUSR=&HD000
20 DATA 3E,C3          : 'LD  A,C3h
30 DATA 01,10,D0       : 'LD  BC,D010h
40 DATA 02             : 'LD  (BC),A
50 DATA C9,"RET"       : 'RET
70 READ A$:IF A$="RET" THEN 110
90 POKE &HD000+T,VAL("&h"+A$):T=T+1:GOTO 70
110 A=USR(A)
120 PRINT "A=";HEX$(PEEK(&HD010))
run
A=C3
Ok

```

Пример 4. Проиллюстрируем работу команды EX DE, HL.

[10101-04.bas](#)

 [10101-04.bas](#)

```

10 CLEAR 200,&HD000:DEFUSR=&HD000
20 DATA 11,C3,98       : 'LD  DE,98C3h
30 DATA 21,A1,56       : 'LD  HL,56A1h
40 DATA ED,53,30,D0    : 'LD  (D030h),DE
50 DATA ED,63,32,D0    : 'LD  (D032h),HL
60 DATA EB             : 'EX  DE,HL
70 DATA ED,53,34,D0    : 'LD  (D034h),DE
80 DATA ED,63,36,D0    : 'LD  (D036h),HL
90 DATA EB             : 'EX  DE,HL
100 DATA ED,53,38,D0   : 'LD  (D038h),DE
110 DATA ED,63,3A,D0   : 'LD  (D03Ah),HL

```

```

120 DATA C9,"RET"      : 'RET
140 READ A$
150 IF A$="RET" THEN 180
160 POKE &HD000+T,VAL("&h"+A$)
170 T=T+1:GOTO 140
180 A=USR(A)
190 PRINT "До команды EX DE,HL:"
200 PRINT " DE=";HEX$(PEEK(&HD031));HEX$(PEEK(&HD030))
210 PRINT " HL=";HEX$(PEEK(&HD033));HEX$(PEEK(&HD032))
220 PRINT "После команды EX DE,HL:"
230 PRINT " DE=";HEX$(PEEK(&HD035));HEX$(PEEK(&HD034))
240 PRINT " HL=";HEX$(PEEK(&HD037));HEX$(PEEK(&HD036))
250 PRINT "После еще одной команды EX DE,HL:"
260 PRINT " DE=";HEX$(PEEK(&HD039));HEX$(PEEK(&HD038))
270 PRINT " HL=";HEX$(PEEK(&HD03B));HEX$(PEEK(&HD03A))
run
До команды EX DE,HL:
  DE=98C3
  HL=56A1
После команды EX DE,HL:
  DE=56A1
  HL=98C3
После еще одной команды EX DE,HL:
  DE=98C3
  HL=56A1
Ok

```

Пример 5. Сложение целых чисел, принадлежащих отрезку [0,255].

[10101-05.bas](#)

 [10101-05.bas](#)

```

10 CLEAR 100,&HD000      'Очистка памяти
20 POKE &HD000,9          'Запись в память первого операнда
30 POKE &HD001,9          'Запись в память второго операнда
40 AD=&HD003               'Адрес начала подпрограммы
50 READ T$:IF T$="z" THEN 90
70 POKE AD,VAL("&h"+T$):AD=AD+1:GOTO 50
90 DEFUSR=&HD003:H=USR(0)
105 PRINT PEEK(&HD002)    'Печать результата
110 DATA 3A,00,D0        : 'LD  A,(D000) ; (D000h) → A
120 DATA 47              : 'LD  B,A      ; (A)   → B
130 DATA 3A,01,D0        : 'LD  A,(D001) ; (D001h) → A
140 DATA 80              : 'ADD  A,B      ; (A)+(B) → A
150 DATA 32,02,D0        : 'LD  (D002),A ; (A)   → (D002h)
160 DATA C9,"z"          : 'RET          ; Возврат из подпрограммы

```

Пример 6. Умножение содержимого двух ячеек памяти.

[10101-06.bas](#)

 [10101-06.bas](#)

```

10 CLEAR 200,&HA000
20 INPUT A,B:POKE &HA100,A:POKE &HA101,B 'Ввод начальных данных
30 FOR I=0 TO 15:READ R$:POKE &HA000+I,VAL("&h"+R$):NEXT
40 DEFUSR=&HA000:Z=USR(0)
50 PRINT PEEK(&HA102)
60 DATA 3A,00,A1        : ' LD   A,(A100h)
70 DATA 47              : ' LD   B,A
80 DATA 05              : ' DEC  B
90 DATA 3A,01,A1        : ' LD   A,(A101h)
100 DATA 57             : ' LD   D,A
110 DATA 82             : ' ADD  A,D ←
120 DATA 10,FD          : ' DJNZ $-1 ←
130 DATA 32,02,A1       : ' LD   (A102h),A

```

```
150 DATA C9          : ' RET
```

Пример 7. Нахождение суммы целых чисел от 1 до 10.

[10101-07.bas](#)

 [10101-07.bas](#)

```
10 CLEAR 100,&HD000:DEFUSR=&HD000:AD=&HD000
40 READ A$:IF A$="Z"THEN 80
60 POKE AD,VAL("&h"+A$):AD=AD+1:GOTO 40
80 A=USR(0):PRINT"Сумма: "PEEK(&HD00C):END
110 DATA 06,0A      : ' LD    B,0Ah
120 DATA 3E,00      : ' LD    A,00h
130 DATA 80         : ' ADD   A,B      ; ←
140 DATA 10,FD      : ' DJNZ  D004h    ; ←
150 DATA 32,0C,D0   : ' LD    (D00C),A
160 DATA C9,"Z"     : ' RET
```

Пример 8. Нахождение большего из двух чисел.

[10101-08.bas](#)

 [10101-08.bas](#)

```
10 PRINT"Найду большее из двух чисел (0÷255)"
20 INPUT"Первое число ";N1:INPUT"Второе число ";N2
40 POKE &HC000,N1:POKE &HC001,N2
50 CLEAR 200,&HD000:DEFUSR=&HD000:I=&HD000
70 READ A$:IF A$="z" THEN 110
90 POKE I,VAL("&h"+A$):I=I+1:GOTO 70
110 A=USR(0)
120 PRINT"Большее число: ";PEEK(&HC003):END
130 DATA 3A,00,C0    : ' LD A,(C000h)
140 DATA 47          : ' LD B,A
150 DATA 3A,01,C0    : ' LD A,(C001h)
160 DATA B8          : ' CP B
170 DATA F2,0C,D0    : ' JP P,D00Ch
180 DATA 78          : ' LD A,B
190 DATA 32,03,C0    : ' LD (C003h),A
200 DATA C9,"z"      : ' RET
```

Напомним, что подпрограмма в машинных кодах вызывается следующим образом:

```
Переменная = USR[n](аргумент)
```

где

- n — целое число, принадлежащее отрезку [0,9], которое указывает номер подпрограммы в машинных кодах.

Значение *аргумента* передается подпрограмме в машинных кодах. При вызове подпрограммы число байтов, занимаемое этим значением, однозначно связанное с его типом, заносится в регистр A микропроцессора Z80 и в ячейку памяти F663h. Возможны следующие значения:

- 2 — целочисленные,
- 4 — вещественные с одинарной точностью,
- 8 — вещественные с двойной точностью,
- 3 — строковые.

Адрес числовой *переменной* хранится в регистре HL; адрес строковой *переменной* находится в регистре DE.

Пример 9. Нахождение большего из двух чисел.

[10101-09.bas](#)

 [10101-09.bas](#)

```
10 CLEAR 200,&HF000
```

```

20 DATA ED,63,02,F3      : 'LD  (F302h),HL
30 DATA 3A,F8,F7         : 'LD  A,(F7F8h)
40 DATA 4F               : 'LD  C,A
50 DATA 81               : 'ADD  A,C
60 DATA 32,00,C0         : 'LD  (C000h),A
70 DATA C9,"RET"         : 'RET
80 DEFUSR=&HF000
90 READ Z$:POKE &HF000+T,VAL("&H"+Z$)
100 T=T+1:IF Z$<>"RET" THEN 90
110 A=USR(6) 'Обратите внимание на то, что аргумент имеет целый тип!
120 PRINT" В регистре HL находится число: &h";HEX$(PEEK(&HF303));RIGHT
    $("00"+HEX$(PEEK(&HF302)),2)
125 PRINT" Это есть адрес аргумента функции USR"
130 PRINT" Проверяем этот факт:... и вправду ";LEFT$(HEX$(PEEK(&HF7F7)),1)
140 PRINT" Результат работы подпрограммы в машинных кодах: ";PEEK(&HC000)
guy
В регистре HL находится число: &hF7F6
Это есть адрес аргумента функции USR
Проверяем этот факт:... и вправду 6
Результат работы подпрограммы в машинных кодах: 12
Ok

```

Пример 10. Нахождение большего из двух чисел.

[10101-10.bas](#)

 [10101-10.bas](#)

```

10 CLEAR 200,&HF000
20 DATA 32,00,F3          : 'LD  (F300h),A
30 DATA 3A,63,F6          : 'LD  A,(F663h)
40 DATA 32,01,F3          : 'LD  (F301h),A
50 DATA ED,53,02,F3       : 'LD  (F302h),DE
60 DATA C9,"RET"          : 'RET
70 DEFUSR=&HF000
80 READ Z$:POKE &HF000+T,VAL("&h"+Z$)
90 T=T+1:IF Z$<>"RET" GOTO 80
100 A$="MSX":A$=USR(A$) 'Обратите внимание на то, что тип аргумента - строковый !
110 PRINT"В регистре A: ";PEEK(&HF300)
120 PRINT"В ячейке F663h: ";PEEK(&HF301)
130 PRINT"Адрес переменной: ";HEX$(PEEK(&HF303));RIGHT$("00"+HEX$(PEEK(&HF302)),2)
guy
В регистре A: 3
В ячейке F663h: 3
Адрес переменной:8183
Ok
?hex$(peek(&h8185));hex$(peek(&h8184))
80E7
Ok
?chr$(peek(&h80E7))
M
Ok

```

Итак, информация о типе переменной A\$ хранится в ячейке с адресом &HF663, а ссылка на адрес переменной A\$ (&H8183) хранится в регистре DE.

Если вы не тот, кто наверху, значит,
вы тот, кто внизу.

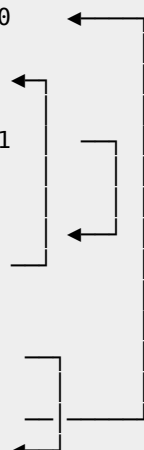
—С.Поттер

Пример 11. Сортировка целочисленного массива , содержащего 5 элементов.

[10101-11.bas](#)

10101-11.bas

```
10 CLEAR 200,&HA000:DEFUSR=&HA000:AD=&HA000
20 READ A$:IF A$="z" THEN 50
30 POKE AD,VAL("&h"+A$)
40 AD=AD+1:GOTO 20
50 FOR I=0 TO 4:INPUT A:POKE &HB000+I,A:NEXT
60 A=USR(0)
70 FOR I=0 TO 4:PRINT PEEK(&HC000+I):NEXT
80 END
90 DATA 0E,00      : 'LD    C,00
100 DATA 11,00,C0  : 'LD    DE,C000
110 DATA 21,00,B0  : 'LD    HL,B000
120 DATA 06,05     : 'LD    B,05
130 DATA 7E        : 'LD    A,(HL)
140 DATA B9        : 'CP     C
150 DATA C2,11,A0  : 'JP     NZ,A011
160 DATA 12        : 'LD     (DE),A
170 DATA 13        : 'INC    DE
180 DATA 23        : 'INC    HL
190 DATA 10,F6     : 'DJNZ   A00A
200 DATA 79        : 'LD     A,C
210 DATA FE,FF     : 'CP     FF
220 DATA CA,1E,A0  : 'JP     Z,A01E
230 DATA 0C        : 'INC    C
240 DATA C3,05,A0  : 'JP     A005
250 DATA C9,"z"    : 'RET
```

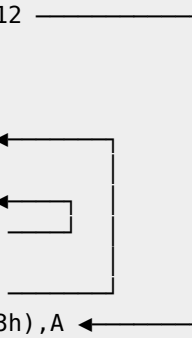


Пример 12. Вычисление факториала «маленьких» целых чисел

10101-12.bas

10101-12.bas

```
10 CLS:KEYOFF:PRINT"Найду факториал заданного Вами целого числа"
20 INPUT"Введите целое число, меньше 6";F
30 POKE &HC000,F
40 IF F=0 OR F=1 THEN POKE &HC003,1:GOTO 90
50 CLEAR 200,&HD000:DEFUSR=&HD000:I=&HD000
60 READ A$:IF A$="z" THEN 80
70 POKE I,VAL("&h"+A$):I=I+1:GOTO 60
80 A=USR(0)
90 PRINT "Факториал "PEEK(&HC000)"=" PEEK(&HC003):END
100 DATA 3A,00,C0      : 'LD    A,(C000h)
110 DATA FE,02        : 'CP     2h
120 DATA 28,0A        : 'JR     Z,$+12
130 DATA 47            : 'LD     B,A
140 DATA 05            : 'DEC     B
150 DATA 05            : 'DEC     B
160 DATA 48            : 'LD     C,B
170 DATA 57            : 'LD     D,A
180 DATA 82            : 'ADD     A,D
190 DATA 10,FD         : 'DJNZ   $-1h
200 DATA 41            : 'LD     B,C
210 DATA 10,F8         : 'DJNZ   $-6h
220 DATA 32,03,C0     : 'LD     (C003h),A
230 DATA C9,"z"       : 'RET
```



Пример 13. Умножение шестнадцатиразрядных целых чисел без знака.

10101-13.bas

10101-13.bas

```
20 CLEAR 200,&HF000
40 INPUT"Первое целое число";N1%
50 INPUT"Второе целое число";N2%
```

```

60 POKE &HF000,PEEK(VARPTR(N1%)):POKE &HF001,PEEK(VARPTR(N1%)+1)
70 POKE &HF002,PEEK(VARPTR(N2%)):POKE &HF003,PEEK(VARPTR(N2%)+1)
80 DEFUSR=&HF006
90 DATA ED,6B,00,F0 : ' ARG1: LD HL,(F000)
100 DATA ED,5B,02,F0 : ' ARG2: LD DE,(F002)
110 DATA 06,10 : ' LD B,10
120 DATA 4A : ' LD C,D ; Пересылка множителя
130 DATA 7B : ' LD A,E
140 DATA EB : ' EX DE,HL ; Пересылка множимого
150 DATA 21,00,00 : ' LD HL,0000 ; Установка частичного результата
                        в исходное состояние
160 DATA CB,39 : ' ML00P:SRL C ; Сдвиг множителя вправо
170 DATA 1F : ' RRA ; Наименее значащий разряд к переносу
180 DATA 30,01 : ' JR NC,NOADD-S; Если нет переноса, то
                        пропуск суммирования
190 DATA 19 : ' ADD HL,DE ; Иначе суммирование мно-
                        жимого в частичный результат
200 DATA EB : ' NOADD:EX DE,HL ; Сдвиг множимого влево
210 DATA 29 : ' ADD HL,HL ; при умножении на 2
220 DATA EB : ' EX DE,HL
230 DATA 10,F5 : ' DJNZ ML00P-S ; Повторение до последнего
                        разряда
240 DATA ED,63,04,F0 : ' REZ: LD (F004),HL
250 DATA C9 : ' RET
260 FOR T=0 TO 31:READ Z$:POKE &HF006+T,VAL("&h"+Z$):NEXT
270 X=USR(X) 'Выполним умножение чисел N1% и N2%
280 MULT=256*PEEK(&HF005)+PEEK(&HF004)
290 IF MULT>32767 THEN MULT=MULT-65536!
300 PRINT"Ваш результат: ";MULT:END

```


http://sysadminmosaic.ru/msx/basic_dialogue_programming_language/101

2023-06-11 23:58

