МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО
СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РСФСР

КУЙБЫШЕВСКИЙ ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
АВИАЦИОННЫЙ ИНСТИТУТ имени академика С. П. КОРОЛЕВА

# КРОССОВЫЕ СРЕДСТВА ПРОГРАММИРОВАНИЯ ДЛЯ МИКРОЭВМ „ЭЛЕКТРОНИКА-60“

КУЙБЫШЕВ 1983

Министерство высшего и среднего специального
образования Р С Ф С Р

Куйбышевский ордена Трудового Красного Знамени
авиационный институт имени академика С.П.Королева

# КРОССОВЫЕ СРЕДСТВА ПРОГРАММИРОВАНИЯ ДЛЯ МИКРО-ЭВМ "ЭЛЕКТРОНИКА-60"

Утверждено редакционным советом института
в качестве методических указаний
к лабораторным работам по курсу
"Программирование для УВМ"

Куйбышев 1984

УДК 681.3

Предлагаемые методические указания связаны с использованием кросс-системы программирования для микро-ЭВМ "Электроника-60". Использование этой системы позволяет выполнять программы для микро-ЭВМ в среде ДОС ЕС в пакетном режиме, что по многим причинам оказывается удобным при организации обучения программированию для микро-ЭВМ.

Кросс-система программирования разработана в Воронежском государственном университете и внедрена в учебный процесс для специальностей 0646, 0647 КуАИ.

Составители : М.А.Кораблин, А.А.Сидоров, М.А.Шамашов

Рецензенты : В.С.Семенов, Б.К.Брюханов

Эффективность использования микро-ЭВМ и систем, построенных на их основе в значительной степени зависит от эффективности программного обеспечения, а последнее, в свою очередь, определяется технологией его изготовления. Разработка и отладка программ непосредственно на резидентной микро-ЭВМ зачастую затруднена из-за ее ограниченных ресурсов (небольшая оперативная память, отсутствие устройств отображения и регистрации информации). Учитывая стоимость дополнительного оборудования, необходимого для разработки программ непосредственно на самой микро-ЭВМ, и трудоемкость этого процесса, часто оказывается дешевле и проще создавать программное обеспечение микро-ЭВМ на больших универсальных ЭВМ, называемых в этом случае инструментальными.

Для разработки программ микро-ЭВМ с помощью универсальных ЭВМ на последних создаются специальные программные системы, получившие название кросс-систем, включающие набор кросс-ассемблеров, кросс-компиляторов, компановщиков, эмуляторов и достаточно удобные средства отладки.

Целью данной лабораторной работы является изучение программирования для микро-ЭВМ "Электроника-60" и выполнения составленных программ в инструментальной среде кросс-системы.

## I. Назначение и состав кросс-системы

Описываемая кросс-система программирования представляет собой комплекс программ на ЕС ЭВМ, обеспечивающий удобные и эффективные средства разработки и отладки программ для ЭВМ "Электроника-60". Она предоставляет пользователю большой спектр возможностей, которые реализуются набором программ, включенных в ее состав. К ним относятся кросс-ассемблер; встроенный в ассемблер макро-генератор; компановщик; загрузчик; эмулятор; встроенные в эмулятор средства пакетной и интерактивной отладки; программы, обеспечивающие переносимость программного обеспечения из кросс-системы на ЭВМ "Электроника-60"

и обратно на уровне объектных, загрузочных и исходных модулей (конверторы).

Процесс выполнения программ в рамках кросс-системы включает следующую последовательность технологических операций. Первоначально исходный текст программы обрабатывается кросс-ассемблером, в результате чего формируется листинг программы, объектный код и фиксируются синтаксические ошибки. При отсутствии последних, в соответствии с директивами .CSECT и .ASECT /I/ компоновщиком формируется абсолютный модуль, поступающий на вход эмулятора. Эмулятор, это программа имитирующая выполнение команд процессора ЭВМ "Электроника-60" в ЕС ЭВМ. В том случае, если задан режим трассировки, эмулятор формирует отладочный файл. В тех случаях, когда программа уже отлажена, она при помощи конвертора выводится на перфоленту, которая может быть затем введена и выполнена в "Электронике-60".

## 2. КРОСС-АССЕМБЛЕР

Кросс-ассемблер воспринимает исходную программу с любых носителей информации или из библиотеки исходных модулей. (В лабораторной работе исходная программа будет подготавливаться на перфокартах). Результатом его работы является листинг программы с сообщениями об ошибках, получение загрузочного модуля и вызов эмулятора, осуществляющего имитацию выполнения и отладку программы.

В качестве входного языка кросс-ассемблера используется перемещающий ассемблер перфоленточной операционной системы микро-ЭВМ "Электроника-60" с некоторыми расширениями :

- в идентификаторах разрешено использование русских букв,
- символы ВК (возврат каретки), ПС (перевод строки) при подготовке данных в коде ДКОИ (на перфокартах) не используются,
- для управления режимами ассемблирования в языке введена директива РЕЖИМ . Таких режимов довольно много. Остановимся только на двух из них.

Для передачи управления эмулятору после кросс-трансляции, с целью имитации выполнения программы и ее отладки необходимо в качестве первой карты исходного модуля задать :

```
.РЕЖИМ  SYM, ВЫП
```

Если же программа отлажена и необходимо получить загрузочный модуль на перфоленте с целью его использования на микро-ЭВМ необходимо указать :

РЕЖИМ ПЕРФ

Для написания программы на языке ассемблера "Электроника-60" необходимо ознакомиться с литературой /1,2/. Состав команд микро-ЭВМ приведен в Приложении I. Там же приведены диагностические сообщения о наиболее распространенных синтаксических ошибках.

## 3. ЭМУЛЯТОР И СРЕДСТВА ОТЛАДКИ

Эмулятор предназначен для имитации выполнения машинных программ ЭВМ "Электроника-60" на ЕС ЭВМ. Результаты такой имитации адекватны результатам, которые были бы получены на резидентной микро-ЭВМ. Кроме имитации всех машинных команд микро-ЭВМ эмулятор обеспечивает моделирование работы ряда ее внешних устройств.

В эмулятор встроены средства пакетной и интерактивной отладки позволяющие совмещать имитацию выполнения программы с ее отладкой. Отладка осуществляется с помощью специальных директив отладки или операторов, которые выполняются в процессе имитации программы.

В лабораторной работе отладку небольших по объему программ предполагается проводить с помощью полной трассировки. Для задания полной трассировки необходимо использовать оператор отладки :

```
<< TR
```

Полная трассировка - классический метод отладки, позволяющий анализировать работу программы после выполнения каждой команды микро-ЭВМ.

После выполнения функции TR на АЦПУ будет выводиться информация, которая для различных типов команд "Электроника-60" будет выглядеть по разному :

I) Безадресные команды :

```
TP: АКОМ = XXXXXX   КОМ = МНЕМ   КОД = XXXXX   ССП = IIIII
```

2) Команды ветвлений :

```
TP: АКОМ = XXXXXX   КОМ = МНЕМ   КОД = XXXXX
                    НСК = XXXXX   ССП = IIIII
```

3) Одноадресные команды :

```
TP: АКОМ = XXXXXX   КОМ = МНЕМ   КОД = XXXXX
АОП1 = XXXXXX   ОП1 = XXXXXX   РЕЗ = XXXXX   ССП = IIIII
```

4) Двухадресные команды :

ТР: АКОМ= XXXXXX КОМ=МНЕМ КОД=XXXXXX АОП1= XXXXXX
ОП1 = XXXX АОП2=XXXXXX РЕЗ= XXXXXX ССП=IIIIII

5) Команды переходов :

ТР: АКОМ=XXXXXX КОМ=МНЕМ КОД=XXXXXX УС=XXXXXX
АСК = XXXXXX ССП = IIIIII

6) Команды прерываний :

ТР: АКОМ = XXXXXX КОМ=МНЕМ КОД=XXXXXX АВЕКТ=XXXXX
УС = XXXXXX НСК = XXXXXX ССП=IIIIII

Здесь ТР – признак трассировочной печати, XXXXXX – восьмиричное число, IIIIII – двоичное число, МНЕМ – набор символов. При выдаче трассировки в перечисленных сообщениях XXXXXX , МНЕМ , IIIIII являются конкретными значениями следующих параметров :

АКОМ – адрес команды;
КОМ – мнемоника команды;
КОД – код команды;
ССП – слово состояние процессора;
НСК – новое значение счетчика команд;
РЕЗ – результат выполнения команды;
АОП1 – адрес первого операнда;
АОП2 – адрес второго операнда;
ОП1 – первый операнд;
ОП2 – второй операнд;
УС – значение указателя стека;

ABEKT – адрес вектора прерывания.

## 4. ОФОРМЛЕНИЕ ПРОГРАММ ДЛЯ ВЫПОЛНЕНИЯ В КРОСС-СИСТЕМЕ

Рассмотрим оформление пакета заданий на трансляцию, эмуляцию и отладку программ для микро-ЭВМ "Электроника-60" в ДОС ЕС ЭВМ на примере программы получения суммы SUM ряда чисел X . Все числа ряда длиной в слово, число членов ряда содержится в области памяти с именем N . Предполагается, что сумма членов ряда не превышает по абсолютной величине 32767. На рисунке I приведен пример оформления задания.

```
// JOB    ПЕТРОВ   ГР.651 ПУЕМ-Л1
// EXEC   ASS60
   .РЕЖИМ SYM,ВЫП
   .CSECT
    R0=%0
    R1=%1
    R3=%3
N:    .WORD  10.
X:    .WORD  6,4,-7,5,33,9.,53,1,-1,13
SUM:  .BLKW 1
; ++++    НАЧАЛО  ПРОГРАММЫ   ++++
BEG:  CLR    R0          ; ОЧИСТКА R0
      MOV    N,R1        ; 10=>R1
      MOV    #X,R3       ; В R3 АДРЕС X
M:    ADD    (R3)+,R0    ; НАКОПЛЕНИЕ СУММЫ
      SOB    R1,M        ; ЦИКЛ
      MOV    R0,SUM
      HALT               ; ОСТАНОВКА Ц.П.
      .END   BEG
/*
 <<TR
/*
/*
/&
```

Выполнение данного задания приведет к трансляции приведенной программы и ее трассировке.

Инструкция оператору, записанная на колоде, должна содержать следующую информацию :

1. Установить диск "учебный процесс.Резидент" на X'190'.
2. Установить диск "учебный процесс. Рабочий том" на X'191'.
3. При завершении программы с сообщением HALT ввести символ E и нажать ВК.

## 5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Процесс выполнения лабораторной работы можно разделить на 4 этапа :

1. Ознакомительный. На этом этапе студент знакомится с теоретическими основами кросс-систем программирования.
2. Расчетный. В соответствии с полученным заданием разрабатывается алгоритм, составляется блок-схема и программа на кросс-ассемблере.
3. Лабораторный. Осуществляется перфорация и отладка созданной программы с использованием эмулятора микро-ЭВМ "Электроника-60" в среде ДОС ЕС.
4. Оформление отчета. Отчет начинается с названия работы и содержит следующие разделы.

- Задание;
- блок-схему алгоритма;
- распечатку отлаженной программы;
- трассировку результатов выполнения.

## 6. ВАРИАНТЫ ЗАДАНИЙ

1. Составить программу определения максимального элемента и его порядкого номера в линейном целочисленном массиве A(10). Результат записать в ячейки MAX и NUM соответственно.
2. Составить программу для подсчета числа нулевых и единичных битов в массиве из пяти последовательных ячеек памяти. Результат записать в ячейки N и E .
3. Составить программу, изменяющую значение i -го бита в слове W в соответствии со значением R0 : если R0=1 , то бит ус-

танавливается в I; если $R\emptyset = \emptyset$, то бит устанавливается в $\emptyset$; при $R\emptyset \neq \emptyset$ и $R\emptyset \neq 1$ бит не изменяется. Значение $c$ задается в регистре $R1$. Результат записывается в ячейку $W$.

4. Составить программу нахождения среднего значения элементов целочисленного массива $B(16)$. При возникновении переполнения в процессе вычислений в ячейку $ERR$ записывается I. Результат записывается в ячейку $RE\,z$

5. Составить программу для преобразования массива целых чисел в символьную строку в коде $ASCII$

6. Составить программу для поиска и подсчета в символьной строке $STR$ числа символов $'A'$. Символьная строка задана в коде $ASCII$. Признак конца строки – символ "точка".

7. Составить программу сортировки массива целых $A(1\emptyset)$ чисел в порядке возрастания их значений.

8. Составить программу для подсчета числа элементов массива целых чисел $X(1\emptyset)$ для которых выполняется условие $a < x_i < b$

9. Составить программу для подсчета разности положительных и суммы отрицательных чисел в целочисленном массиве.

I0. Составить программу для определения среднего и дисперсии выборки из I0 действительных чисел.

II. Составить подпрограмму для вычисления функции

$$e^x = 1 + x + \frac{x^2}{2}! + \frac{x^3}{3}! + \ldots$$

I2. Составить подпрограмму для вычисления функции

$$SIN(x) = x - \frac{x^3}{3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)} + \ldots$$

I3. Составить программу для вычисления значения полинома в точке по схеме Горнера

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 =$$
$$= (\ldots (a_n x + a_{n-1}) \cdot x + \ldots + a_1) x + a_0$$

где $a_i$ и $x$ – целые числа.

I4. Составить программу вычисляющую сумму абсолютных величин чисел, записанных в массиве $A(1\emptyset)$

I5. Составить программу поиска числа равного $x$, в упорядоченном массиве $A(1\emptyset)$ по методу двоичного поиска.

I6. Составить программу преобразования строки из 5 цифр в массив чисел $X(5)$

I7. Составить подпрограмму умножения целых чисел путем много-

кратного сложения. При возникновении переполнения в ячейку записывается I; при отсутствии переполнения – 0.

I8. Составить программу сложения двух целочисленных линейных массивов $A(10)$ и $B(10)$. Результат записать в массив $C(10)$

I9. Составить программу сжатия строки символов $S$ путем удаления из нее пробелов. Признак конца строки – символ "точка".

20. Составить программу определения максимального и минимального элементов в массиве $A(1Ø)$, каждый элемент которого занимает один байт памяти.

2I. Составить программу построения гистограммы по выборке $X(1Ø)$ целых чисел.

Приложение I

СОСТАВ КОМАНД МИКРО-ЭВМ "ЭЛЕКТРОНИКА-60"

| Мнемокод | Команда | Коды условий Z | N | C | V | Описание операции |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | | | | 4 |
| ADD | Сложение | ? | ? | ? | ? | (П) := (П) + (Ц) |
| ADC | Прибавление переноса | ? | ? | ? | ? | (П) := (П) + (С) |
| ASR(B) | Арифм. сдвиг вправо | ? | ? | ? | ? | (С)\|\|(П) := (П)/2 |
| ASL(B) | Арифм. сдвиг влево | ? | ? | ? | ? | (С)\|\|(П) := 2 * (П) |
| * ASH | Многократный арифм. сдвиг | ? | ? | ? | ? | (Р) := (Р)/2^(П<Ф)<br>(Р) := (Р)*2^(П>Ф) |
| * ASHC | Многократный арифм. сдвиг двойного слова | ? | ? | ? | ? | |
| BIT(B) | Побитовая проверка | ? | ? | - | 0 | (Ц) ∧ (П) |
| BIS(B) | Побитовая установка | ? | ? | - | 0 | (П) := (Ц) ∨ (П) |
| BIC(B) | Побитовый сброс | ? | ? | - | 0 | (П) := $(\bar{Ц})$ ∧ П |
| BR | Безусловный переход | - | - | - | - | (СК) := (СК) + 2·хх |
| BNE | Переход, если не нуль | - | - | - | - | (СК) := (СК) + 2·хх, если Z=Ф |
| BEQ | Переход, если нуль | - | - | - | - | (СК) := (СК) + 2·хх, если Z=1 |
| BPL | Переход, если плюс | - | - | - | - | (СК) := (СК) + 2·хх, если N=0 |
| BMI | Переход, если минус | - | - | - | - | (СК) := (СК) + 2·хх, если N=1 |
| BGE | Переход, если больше или | - | - | - | - | (СК) := (СК) + 2·хх, если N∨=0 |

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| BLT | Переход, если меньше нуля | — — — — | (CK): = (CK) + 2 XX, если N⊻V=0 |
| BGT | Переход, если больше нуля | — — — — | (CK): = (CK) + 2·XX, если Z∨(N⊻V)=0 |
| BLE | Переход, если меньше или равно нулю | — — — — | (CK): = (CK) + 2·XX, если Z∨(N⊻V)=1 |
| BHI | Переход, если больше | — — — — | (CK): = (CK) + 2·XX, если |
| BLOS | Переход, если меньше или равно | — — — — | (CK): = (CK) + 2·XX, если C∨Z=1 |
| BHIS | Переход, если больше или равно | — — — — | идентично BCC |
| BLO | Переход, если меньше | — — — — | идентично BCS |
| BVC | Переход, если нет переполнения | — — — — | (CK): = (CK) + 2·XX, если V=0 |
| BVS | Переход, если переполнение | — — — — | (CK): = (CK) + 2·XX, если V=1 |
| BCC | Переход, если нет переноса | — — — — | (CK): = (CK) + 2·XX, если C=0 |
| BCS | Переход, если есть перенос | — — — — | (CK): = (CK) + 2·XX, если C=1 |
| BPT | Командное прерывание для отладки | ? ? ? ? | ↓(YC): = (CCП); ↓(YC): = CK; (CK): = (A14); CK: = (A16) |
| CLR(B) | Очистка | 1 0 0 0 | (П): = 0 |
| COM(B) | Инвертирование | ? ? 1 0 | (П): = ($\overline{П}$) |
| CLN | Очистка N | — 0 — — | N: = 0 |
| CLZ | Очистка Z | 0 — — — | Z: = 0 |
| CLV | Очистка V | — — — 0 | V: = 0 |
| CLC | Очистка C | — — 0 — | C: = 0 |
| CCC | Очистка всех разрядов | 0 0 0 0 | N: = Z: = V: = C = 0 |

| 1 | 2 | 3 | | | | 4 |
|---|---|---|---|---|---|---|
| CMP(B) | Сравнение | ? | ? | ? | ? | (Ш) - (П) |
| DEC | Вычитание единицы | ? | ? | - | ? | (П):=(П)-1 |
| DIV | Деление | ? | ? | ! | ? | (R,R+1):=(R,R+1)/(4) |
| EMT | Системное прерывание | ? | ? | ? | ? | ↓(УС):=(ССП); ↓(УС)=(СК) (СК):=(А30); (ССП):=(А32) |
| *FADD | Сложение с плавающей точкой | ? | ? | ф | ф | [(R)+4,(R)+6]:=[(R)+4,(R)+6] +[(R),(R)+2] |
| *FSUB | Вычитание с плав.точкой | ? | ? | ф | ф | [(R)+4,(R)+6]:=[(R)+4,(R)+6] -[(R),(R)+2] |
| *FMUL | Умножение с плав.точкой | ? | ? | ф | ф | [(R)+4,(R)+6]:=[(R)+4(R)+6] x[(R),(R)+2] |
| *FDIV | Деление с плав.точкой | ? | ? | ф | ф | [(R)+4,(R)+6]:[(R)+4,(R)-6] /[(R),(R)+2] |
| HALT | Останов | – | – | – | – | |
| INC | Прибавление единицы | ? | ? | - | ? | (П):=(П)+1 |
| IOT | Прерывание по вводу-выводу | ? | ? | ? | ? | ↓(УС):=(ССП); ↓(УС):=(СК); (СК):=(А2ф); (ССП):=(А22) |
| JMP | Безусловный переход | – | – | – | – | (СК):=(П) |
| JSR | Обращение к подпрограмме | – | – | – | – | ↓(УС):=R; (R):=(СК); (СК):=(П) |
| MFPS | Чтение ССП | ? | ? | - | ф | (П):=(ССП) |
| MTPS | Запись ССП | ? | ? | ? | ? | (ССП):=(Ш) |
| MOV(B) | Пересылка | ? | ? | - | ф | (П):=(Ш) |
| MARK | Восстановление УС | – | – | – | – | (УС):=(СК)+2·XX; (СК):=(R5); (R5):=(УС)↑ |
| *MUL | Умножение | ? | ? | ? | ф | (R, R+1):=(R)x(Ш) |
| NEG(B) | Изменение знака | ? | ? | ф | ? | (П):=-(П) |

| 1 | 2 | 3 | | | | 4 |
|---|---|---|---|---|---|---|
| NOP | Нет операции | – | – | – | – | |
| ROR(B) | Циклический сдвиг вправо | ? | ? | ? | ? | (П,С):=(С,П) |
| ROL(B) | Циклический сдвиг влево | ? | ? | ? | ? | (С,П):=(П,С) |
| RTS | Возврат из подпрограммы | – | – | – | – | (СК):=(R); (R):=(УС)↑ |
| RTI | Возврат из прерывания | ? | ? | ? | ? | (СК):=(УС)↑; (ССП):=(УС)↑ |
| RTT | Возврат из прерывания | ? | ? | ? | ? | (СК):=(УС)↑; (ССП):=(УС)↑ |
| RESET | Сброс внешних устройств | – | – | – | – | |
| SBC | Вычитание переноса | ? | ? | ? | ? | (П):=(П)-(С) |
| SXT | Расширение знака | ? | – | – | 0 | (П):=0, если N=0<br>(П):=-1, если N=1 |
| SWB | Перестановка байтов | ? | ? | ∅ | ∅ | (байт 1, Байт 0):=<br>(байт 0, байт 1) |
| SUB | Вычитание | ? | ? | ? | ? | (П):=(П)-(ЦУ) |
| SOB | Вычитание единицы и переход | – | – | – | – | (R):=(R)-1, если<br>(R)≠0, (СК):=(СК)+2·XY |
| SEZ | Установка Z | 1 | – | – | – | (Z):=1 |
| SEN | Установка N | – | 1 | – | – | (N):=1 |
| SEC | Установка C | – | – | 1 | – | (C):=1 |
| SEV | Установка V | – | – | – | 1 | (V):=1 |
| SCC | Установка Z,N,C,V | 1 | 1 | 1 | 1 | (Z):=(N):=(C):=(V)=1 |
| TST(B) | Проверка признаков | ? | ? | 0 | 0 | ∅-(П) |
| TRAP | Командное прерывание | ? | ? | ? | ? | ↑(УС):=(ССП); ↑(УС):=(ССП)<br>(СК):=(А34); (ССП):=(А36 |
| WAIT | Ожидание | – | – | – | – | |

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| XOR | Исключающее "ИЛИ" | ? ? - 0 | (Р) ∀ (П) |

Примечание. В таблице применены следующие условные обозначения :

1. В колонке 1 буква *В* указывает на то, что данная команда может выполняться над отдельными байтами (например, команда *MOV* - пересылка ячеек, *MOVB* - пересылка байтов). Символом '*' помечены команды расширенной арифметики. Эти команды в данной версии эмулятора не реализованы.

2. В колонке 3 признаки результатов операции обозначены
   - если признак определяется по результатам выполнения операции;
   - признак не изменяется;
   - признак устанавливается в 0 или 1.

3. При описании операций в колонке 4 :

   (Ц) - содержимое ячейки - источника,

   (П) - содержимое ячейки - приемника,

   (Р) - содержимое регистра общего назначения,

   УС - указатель стека,

   Ск - счетчик команд,

   ССП - слово состояния процессора,

   Х - смещение,

   $\wedge$ - логическое И ,

   $\vee$ - логическое ИЛИ,

   $\forall$ - исключающее ИЛИ,

   $\neg$ - отрицание,

   $\downarrow$ - занесение в стек,

   $\uparrow$ - извлечение из стека.

## Приложение 2

### СООБЩЕНИЯ ОБ ОШИБКАХ

В процессе трансляции кросс-ассемблер выдает листинг сообщения об ошибках вслед за оператором, содержащим ошибки. Если ошибку можно привязать к тексту, то в этой же строке печатается символ '¤', указывающий позицию оператора, где обнаружена ошибка. При этом ошибку необходимо искать в окрестностях символа ¤ .

Сообщения, выдаваемые в процессе трансляции и эмуляции программ, приведены соответственно в таблицах П2.1 и П2.2.

Таблица П2.1

Сообщения кросс-ассемблера

| Код ошибки | Сообщение | Примечание |
|---|---|---|
| 1 | 2 | 3 |
| 007 | Повторное определение метки | Метки игнорируются |
| 011 | Значение не помещается в слове | Усекается слева до 16 разрядов |
| 012 | Неверное окончание оператора | Наиболее вероятная причина: отсутствие перед комментарием |
| 020 | Неверная запись директивы | Отсутствует или неверны параметры директивы |
| 025 | Возможны другие ошибки в операторе | Слишком много ошибок в операторе |
| 027 | Отсутствует закрывающий символ | В директивах .ASCII и .RAD50 отсутствует конечный символ |
| 031 | Пропущен операнд, вставлен 0 | В выражении между знаками операции отсутствует операнд |
| 032 | Пропущена операция, вставлен + | В выражении между операндами отсутствует знак операции |
| 035 | Неверен синтаксис оператора | Пропущен операнд(ы) |
| 038 | выражение содержит неопределенные имена | Имя из выражения не определено в программе |
| 046 | Неверно задан режим ра- | Ошибка в директиве РЕЖИМ |

боты ассемблера

Таблица П2.2

Сообщения эмулятора

| Код | Сообщение | Примечание |
|---|---|---|
| EEM 001 | Прерывание по резервной команде | Попытка выполнить команду, не используемую в "Электронике-60" |
| EEM 002 | Прерывание по ошибке обращения к каналу | Попытка обратиться к несуществующему слову памяти или неверная адресация |
| EAT030 | Недействительный признак отладочного оператора | Эти сообщения формируются в случае неверного задания операторов отладки |
| EAT Ø18 | Неверен оператор | |
| EAT Ø15 | Нет конца оператора | |

Литература

I. Основы программирования на Ассемблере для СМ ЭВМ. /Г.В.Вигдорчик, А.Ю.Воробьев, В.Д.Праченко. -М.: Финансы и статистика, 1983. -256 с.

2. Б.Соучек. Микропроцессоры и микро-ЭВМ. -М.: Сов.радио, 1979. -520 с.

Составители: Михаил Александрович К о р а б л и н,
Андрей Анатольевич С и д о р о в,
Михаил Анатольевич Ш а м а ш о в

КРОССОВЫЕ СРЕДСТВА ПРОГРАММИРОВАНИЯ
ДЛЯ МИКРО-ЭВМ "ЭЛЕКТРОНИКА-60"

Подписано в печать 20.01.84г. Формат 60х84 1/16.
Бумага оберточная белая. Оперативная печать.
Усл.п.л. 0,93. Уч.-изд.л. 0,9. Т. 500 экз.
Заказ № 98 Бесплатно.

Куйбышевский ордена Трудового Красного Знамени
авиационный институт имени академика С.П.Королева,
г. Куйбышев,ул. Молодогвардейская, 151.

Офсетный участок КуАИ, г. Куйбышев,ул.Ульяновская,18.