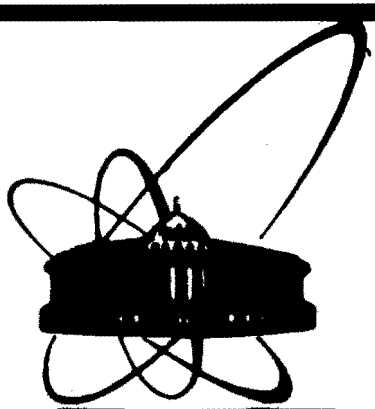

сообщения
объединенного
института
ядерных
исследований
дубна

Р11-86-827

З.Гонс

ASM86•• - КРОСС-АССЕМБЛЕР ДЛЯ МИКРОПРОЦЕССОРА КР1810ВМ86 (INTEL 8086/8088)

1986

В последние годы большое распространение в ЛЯП ОИЯИ приобрели интеллектуальные контроллеры крейта КАМАК[1], основанные на базе восьмиразрядного микропроцессора КР580ИК80. Модернизация экспериментального оборудования и возрастающие требования к скорости накопления данных, в том числе и к необходимости их предварительной обработки, логически ведут к разработке систем на базе более мощных шестнадцатиразрядных процессоров (наиболее часто КР1810ВМ86), эффективность которых по крайней мере на порядок выше. Помимо разработки новой микропроцессорной системы необходимо обеспечить пользователя возможностью составлять свое собственное программное обеспечение. Описываемый кросс-ассемблер ASM86** позволяет создавать программы для микропроцессорных систем, основанных на микропроцессоре КР1810ВМ86 (INTEL 8086/8088). Так как ассемблер ASM86** написан на языке BASIC, он может использоваться почти на любой ЭВМ, обладающей памятью не менее 40К.

1. ОБЩИЕ СВЕДЕНИЯ

Программа ASM86** является кросс-ассемблером для микропроцессора КР1810ВМ86, причем для образования кодов инструкций требуются два или три прохода текста программы. Необходимость третьего прохода вызвана тем, что мнемоническое обозначение команд микропроцессора не всегда позволяет определить число байтов инструкции, до тех пор, пока полностью не идентифицированы ее операнды. Отсюда также следует для пользователя необходимость последовательно определять тип каждого использованного или косвенно адресованного операнда, находящегося в памяти. Программа ASM86** соблюдает мнемоническое обозначение инструкций, введенное разработчиком микропроцессора — фирмой INTEL[2,3]; имеются лишь два исключения (см. приложение 1).

INTEL		ASM86**	
PUSH	сегментный регистр	PUSHS	сегментный регистр
POP	—"—	POPS	—"—

Кроме того, программа ASM86** не позволяет транслировать инструкции, выполняемые арифметическим копроцессором INTEL 8087.

Ассемблер ASM86** формирует оптимизированные объектные коды. Это означает, что во время транслирования определенной инструк-

ции последовательно выбирается вариант, обладающий наименьшим числом байтов и требующий наименьшее время для исполнения инструкции микропроцессором. Полученные в результате ассемблирования объектные коды не требуют дальнейшей обработки программами редактирования связей или загрузки. Это влечет за собой значительное упрощение работы с ассемблером. Составной частью ассемблера является диалоговый построчный редактор текста, позволяющий вводить и модифицировать текст программы.

2. МЕТКИ, ДАННЫЕ, ТИПЫ ОПЕРАНДОВ

Ассемблер работает с двумя типами объектов: метками и данными.

Метками считаются выражения, стоящие либо в начале текстовой строки программы, либо определяемые псевдокомандами EQUN, EQUF (см. ниже). Они используются исключительно в инструкциях типа JMP и CALL с непосредственными операндами. Символы, стоящие в начале текстовой строки или определяемые при помощи псевдоинструкции EQUN, относятся к переходам внутри логического элемента, в то время как символ, определяемый псевдоинструкцией EQUF, относится к межсегментным переходам.

Данные являются либо константами, либо переменными. Константы представляются как в форме десятичных или шестнадцатеричных (с идентификатором H в конце числа) чисел, так и в виде мнемонических имен, определяемых псевдоинструкциями EQUB (длиной в один байт), или EQUW (длиной в два байта). Переменные определяются псевдоинструкциями DB, DW, DD, DB%, DW%, DD%, которые дают непосредственно длину переменной: один байт, два байта (слово) или четыре байта (двойное слово) соответственно.

Псевдоинструкции DB, DW, DD определяют данные, размещенные непосредственно в блоке машинного кода программы, в то время как псевдоинструкции DB%, DW%, DD% определяют данные, которые могут находиться в любом месте оперативной памяти микропроцессорной системы. В качестве операндов псевдоинструкций DB, DW и DD выступают начальные значения переменных, а их адрес определяется ассемблером во время обработки соответствующей текстовой строки. В случае псевдоинструкций DB%, DW% и DD % операнды всегда рассматриваются в качестве шестнадцатиразрядного адресного смещения в определенном логическом сегменте. Данные могут выступать в качестве операндов любой инструкции за исключением непосредственных операндов в инструкциях перехода типа JMP и CALL.

Предположим, что операнд инструкции находится в памяти и способ адресации не позволяет определить его тип (байт, слово, двойное слово). В таком случае требуется дополнительная информация. Для этого служат идентификаторы %BYTE, %WORD, %DWORD, определяющие тип операнда: байт, слово, двойное слово соответственно. Изменение сегментного регистра, используемого в случае косвенной адресации по умолчанию, достигается при помощи инструкции (префикса) CS:, DS:, SS: или ES: (детальное описание см. в [3]).

3. ФОРМАТ ИНСТРУКЦИЙ И ПСЕВДОИНСТРУКЦИЙ

Инструкции и псевдоинструкции вводятся построчно. Каждая строка имеет четыре секции. В первую секцию помещается метка длиной до 6 знаков. Метка должна начинаться буквой. Во вторую секцию вводится мнемоническое обозначение инструкции или псевдоинструкции. Третья и четвертая секции зарезервированы для операндов. Метки, инструкция и первый операнд разделяются пробелами. Между операндами должна быть запятая. Символические обозначения должны соблюдать такие же ограничения, которые были наложены на метку. Кроме того, в качестве символов запрещено пользоваться зарезервированными символами, которые обозначают, например, регистровые операнды. Нумерация строк производится автоматически. Для иллюстрации служат следующие примеры инструкций и псевдоинструкций:

```
Метка     Мнемоника     Операнд 1          Операнд 2
          ORG           2000H
NRCALL    EQUN          2F70H
FAJMP     EQUF          1000H       ,      0FE00H
OFFSET    DB%           0F000H
          MOV           CX          ,      00FEH
METKA1    ES:
          OR            (BX)(DI)BETA ,     AX
BETA      DW            1
          DW            2
          DW            3
          CALL          (BX)%WORD
LOOPX     LOOPNE        LOOPX
          JMP           FAJMP
          OUT           DX          ,      AL
          TEST          CH          ,      (BP)%BYTE
          END
```

Список мнемонических обозначений инструкций микропроцессора КР1810ВМ86 и зарезервированных символов приводится в приложении 1.

4. ПСЕВДОИНСТРУКЦИИ

Псевдоинструкции управляют работой ассемблера и позволяют присваивать определенные атрибуты используемым символам.

Псевдоинструкция ORG устанавливает абсолютный адрес начала программы внутри логического сегмента. Псевдоинструкция END должна следовать после последней текстовой строки программы, подлежащей трансляции. Псевдоинструкции EQUF, EQUN присваивают символу-метке адрес длиной 32 или 16 бит соответственно. В качестве первого операнда псевдоинструкции EQUF должно стоять шестнадцатиразрядное смещение в логическом сегменте, в то время как вторым операндом должно быть желаемое значение сегментного регистра CS. Кроме того, псевдоинструкции EQUN и EQUF приписывают символу атрибут непосредственного операнда для инструкций перехода типа JMP и CALL. Псевдоинструкции EQUB, EQUW определяют константы типа байт и слово соответственно. Их назначение отличается от назначения символов, определенных предыдущим образом, так как они обладают атрибутом данных — констант. Данными также являются объекты, определенные псевдоинструкциями типа DB, DW, DD и DB%, DW%, DD%. Кроме атрибута длины (байт, слово, двойное слово) они обладают также атрибутом переменных. Символы %BYTE, %WORD и %DWORD можно считать псевдоинструкциями, так как они служат лишь для определения типа операнда (байт, слово или двойное слово) в случае косвенной адресации без адресного смещения.

Список псевдоинструкций приводится в приложении 2.

5. РЕДАКТОР ТЕКСТА

Внутренней частью ASM86** является диалоговый редактор текста. В режим редактора можно входить командой EDIT. Для образования текстового файла служат команды CREATE и ADD. Команда CREATE стирает содержимое текстового буфера, в то время как ADD продолжает добавлять строки текста к уже существующим. Команда DELETE m_1, m_2 позволяет устранить участок текста программы между строками m_1 и m_2. Команда REPLACE m_1, m_2 дает возможность заменить текстовые строки программы $m_1 \div m_2$ новым текстом. При необходимости включения участка текста перед строкой можно воспользоваться командой INSERT m. Вывод на экран дисплея всего текста или его части, ограниченной строками $m_1 \div m_2$ проводится при помощи команд LIST или LIST m_1, m_2 соответственно. Ввод текста программы прекращается командой *, в то время как выход из режима редактора текста производится командой BYE.

Список команд редактора текста приводится в приложении 3.

6. РАБОТА С ПРОГРАММОЙ ASM86**

Настоящий раздел посвящен главным образом обучению пользователя работе с ассемблером ASM86**, поэтому рекомендуется одновременно с чтением проводить все операции непосредственно на ЭВМ.

Предположим, что надо решить следующую задачу: через порт с адресом E0H подключен датчик, который информирует о необходимости включения или выключения устройства, соединенного с микроЭВМ через порт с адресом FFF0H. Предположим, что число "1"на выходе E0H означает — "включи устройство в порту FFF0H", в то время как число "0" сигнализирует обратное действие. Включение или выключение устройства обеспечивается записью в порт FFF0H чисел 52H или 10H соответственно.

Старт программы ASM86** проводится либо командой GOTO 1000 либо GOTO 1. Первая команда обеспечивает так называемый "горячий" старт, соблюдающий статус программы в момент ее пуска, в то время как вторая команда реализует "холодный" старт — инициализацию всех массивов и переменных, при которой потеряна вся ранее занесенная информация. Поскольку следует начинать отладку программы с ввода ее текста в ЭВМ, набираем GOTO 1 и выполняющий код CR (только при нажатии клавиши CR ЭВМ будет интерпретировать любую команду, о которой в дальнейшем пойдет речь). На экране появляется курсор # , напоминающий, что находимся в режиме главного меню ASM86** и можно выбирать следующую команду:

CREATE — очистка текстового буфера,
EDIT — переход в режим редактора текста;
LOADT — ввод текста программы с внешнего запоминающего устройства;
LOADC — ввод машинного кода с внешнего запоминающего устройства;
SAVET — сброс текстовой информации на внешнее запоминающее устройство;
SAVEC — сброс машинного кода на внешнее запоминающее устройство;
ASSEMBLE — ассемблирование;
PRINTON — листинг во время ассемблирования идет на экран дисплея и на АЦПУ;
PRINTOFF — листинг идет только на экран дисплея;
RETURN — возврат в BASIC.

Так как команда PRINTOFF действует по умолчанию, набираем

```
CREATE
EDIT
```

и курсор > информирует о том, что мы перешли в режим редактора текста. Для ввода текста служит команда ADD. Действие этой команды приводит к тому, что в начале текущей строки появляется номер текстовой строки (в нашем случае 1) и метка — показывает, где на

экране появится знак, соответствующий следующей нажатой клавише. Так как первой инструкцией программы должна быть псевдоинструкция ORG, которая не требует метки, нажатием клавиши SPACE переместим указатель _ в начало следующей секции — в секцию мнемоники и введем первую псевдоинструкцию и, после пробела, адрес, определяющий абсолютное положение программы в логическом сегменте. Далее вводим другие псевдоинструкции, определяющие использованные символы и продолжаем вводить мнемонические обозначения инструкций программы вместе с их операндами. Текст программы закончим псевдоинструкцией END, символом * вернемся в режим редактора текста и при помощи команды BYE, наконец, в режим главного меню. На экране дисплея высвечивается при этом следующая картинка.

```
CREATE
EDIT
ADD
 1                ORG       1000H
 2      PORT1     EQUB      0E0H
 3      PORT2     EQUW      0FFF0H
 4      OFF       EQUB      10H
 5      TEST      MOV       DX, PORT1
 6                IN        AL, DX
 7                CMP       AL, 1
 8                JZ        GO-ON
 9      GO-OFF    MOC       AL, OFF
10      OUT       MOV       DX, PORT2
11                OUT       DX, AL
12                JMP       TEST
13      GO-ON     MOV       AL, ON
14                JMP       OUT
15                END
16      *
BYE
#
```

Так как мы хотим продемонстрировать по возможности все варианты работы программы ASM86**, в тексте тестовой программы мы специально допустили некоторые ошибки:

1. Символу PORT1 (строка 2) приписан тип байта, хотя константа, обозначенная этим символом, загружается в дальнейшем в шестнадцатиразрядный регистр DX (строка 5).

2. Строка 9 содержит неправильное мнемоническое обозначение инструкции пересылки данных MOC на место MOV.

3. Символ ON не определен (строка 13).

Теперь набираем команду ASSEMBLE. Во время первого прохода ассемблер, строя таблицу символов, замечает, что символ ON не определен и соответственно выдает сообщение: ON — undefined symbol. Так как таблица символов является неполной, ассемблер возвращается

в главное меню. Исправление ошибки 3 проводится следующей последовательностью команд:

```
EDIT
INSERT           4
4  ON            EQUB      52H
5  *
BYE
```

После исправления ошибки 3 набираем снова команду ASSEMBLE. Теперь ассемблер правильно построит таблицу символов, однако во время второго прохода обнаружит до сих пор не устраненные ошибки 1 и 2 и даст следующие объявления:

```
line = 5         illegal data type mixing
line = 9         illegal operands or mnemonic
```

Исправление сделаем следующим образом:

```
EDIT
REPLACE          2,2
2  PORT1         EQUW      0E0H
3  *
REPLACE          10,10
10 GO-OFF        MOV       AL, OFF
11 *
BYE
```

Если будем теперь транслировать отредактированный текст, получим на экране дисплея следующий листинг:

```
 1                ORG       1000H
 2  PORT1         EQUW      0E0H
    1000
 3  PORT2         EQUW      0FFF0H
    1000
 4  ON            EQUB      52H
    1000
 5  OFF           EQUB      10H
    1000
 6  TEST          MOV       DX, PORT1
    1000    BA  E0  00
 7                IN        AL, DX
    1003    EC
 8                CMP       AL, 1
    1004    3C  01
 9                JZ        GO-ON
    1006    74  09
10  GO-OFF        MOV       AL, OFF
    1008    B0  10
11  OUT           MOV       DX, PORT2
    100A    BA  F0  FF
12                OUT       DX, AL
```

```
     100D        EE
13               JMP          TEST
     100E        E9  EF  FF
14   GO-ON       MOV          AL, ON
     1011        B0  52
15               JMP          OUT
     1013        E9  F4  FF
16               END
OK
#
```

Видно, что после каждой строки текста следует абсолютный адрес инструкции в логическом сегменте, а также машинный код, соответствующий мнемоническому обозначению инструкции и набору операндов. Весь ассемблированный модуль программы хранится по байтам в знаковом массиве m$ и его длина указана в переменной "counter" или "counter + число байтов последней инструкции" в зависимости от числа проходов, сделанных ассемблером (два или три соответственно)*.

В конце можно сбросить желаемую информацию на внешнее запоминающее устройство (SAVET, SAVEC).

7. ЗАКЛЮЧЕНИЕ

Описанный ассемблер ASM86** позволяет отлаживать программы, написанные на полном ассемблере микропроцессора INTEL 8086/88, минуя весьма сложный и довольно необычный язык ассемблера фирмы INTEL[3]. Он служит промежуточным звеном для пользователей, идущих от программирования микропроцессора 8080A или других восьмиразрядных микропроцессоров к программированию микропроцессоров шестнадцатиразрядных. Написание программы ASM86** именно на языке BASIC позволяет применять ее на весьма большом круге ЭВМ, в том числе на микроЭВМ. Однако при использовании только интерпретатора языка BASIC программа будет работать во много раз медленнее, чем соответствующий ее вариант, написанный на ассемблере.

В заключение автор выражает благодарность П.Чижеку, Н.А.Бонч-Осмоловской, В.Т.Сидорову, П.Чалоуну за помощь на отдельных этапах работы.

* В случае, если ассемблеру нужен третий проход, выдается два листинга подряд, из них только второй является полным.

Приложение 1

Мнемоническое обозначение инструкций микропроцессора INTEL 8086/88. Обозначение регистровых операндов

AAA	AAD	AAM	AAS	CBW	CLC
CLD	CLI	CMC	CMPSB	CMPSW	CWD
DAA	DAS	HLT	INTO	IRET	LAHF
LOCK	LODSB	LODSW	MOVSB	MOVSW	NOP
POPF	PUSHF	RET	INTR	SAHF	SCASB
SCASW	STC	STD	STI	STOSB	STOSW
WAIT	XLAT	REPNZ	REPN	ES:	CS:
SS:	DS:	JA	JAE	JB	JBE
JCXZ	JE	JG	JGE	JL	JLE
JNA	JNAE	JNB	JNBE	JNE	JNG
JNGE	JNL	JNLE	JNO	JNP	JNS
JNZ	JO	JP	JPE	JPO	JS
JZ	LOOP	LOOPE	LOOPNE	LOOPNZ	LOOPZ
INT	RET	DIV	IDIV	IMUL	MUL
NEG	NOT	DEC	INC	POP	POPS
PUSH	PUSHS	LEA	LDS	LES	SHL
SAL	SHR	SAR	ROL	ROR	RCL
RCR	ESC	IN	OUT	XCHG	ADC
ADD	AND	CMP	OR	SBB	SUR
TEST	XOR	JMP	CALL	MOV	

AX	CX	DX	BX	SP	BP
SI	DI	AL	CL	DL	BL
AH	CH	DH	BH	ES	CS
SS	DS				

Приложение 2

Псевдоинструкции ассемблера ASM86**

ORG	EQUF	EQUN	EQUB	EQUW
DB	DW	DD	%BYTE	%WORD
%DWORD	END	DB%	DW%	DD%

Приложение 3

Меню ассемблера ASM86**

CREATE	EDIT	LOADT	LOADC
SAVET	SAVEC	ASSEMBLE	PRINTON
PRINTOFF	RETURN		
LIST	DELETE	REPLACE	INSERT
BYE	ADD		

ЛИТЕРАТУРА

1. Сидоров В.Т., Синаев А.Н., Чурин И.Н. ОИЯИ, Р10-1281, Дубна, 1979.
2. MCS-86 User's Manual. Intel Corporation,1978.
3. MCS-86 Macro Assembly Language Reference Manual. Intel Corporation, 1979.

Рукопись поступила в издательский отдел
22 декабря 1986 года.

ТЕМАТИЧЕСКИЕ КАТЕГОРИИ ПУБЛИКАЦИЙ ОБЪЕДИНЕННОГО ИНСТИТУТА ЯДЕРНЫХ ИССЛЕДОВАНИЙ

Индекс	Тематика
1.	Экспериментальная физика высоких энергий
2.	Теоретическая физика высоких энергий
3.	Экспериментальная нейтронная физика
4.	Теоретическая физика низких энергий
5.	Математика
6.	Ядерная спектроскопия и радиохимия
7.	Физика тяжелых ионов
8.	Криогеника
9.	Ускорители
10.	Автоматизация обработки экспериментальных данных
11.	Вычислительная математика и техника
12.	Химия
13.	Техника физического эксперимента
14.	Исследования твердых тел и жидкостей ядерными методами
15.	Экспериментальная физика ядерных реакций при низких энергиях
16.	Дозиметрия и физика защиты
17.	Теория конденсированного состояния
18.	Использование результатов и методов фундаментальных физических исследований в смежных областях науки и техники
19.	Биофизика

Гонс З.

P11-86-827

ASM86 — кросс-ассемблер для микропроцессора КР1810ВМ86 (INTEL 8086/8088)

Разработан кросс-ассемблер для микропроцессора КР1810ВМ86, составной частью которого является редактор текста. Для генерирования объектных кодов не требуются программы типа редактора связей или загрузки. Поскольку ассемблер написан на языке BASIC, им можно пользоваться на ЭВМ разных типов.

Работа выполнена в Лаборатории ядерных проблем ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1986

Перевод О.С.Виноградовой

Hons Z.

P11-86-827

ASM86 — Cross Assembler for KR1810VM86 (INTEL 8086/8088) Microprocessors

A cross assembler with dialogue editor supporting the full INTEL 8086/8088 microprocessor instruction set is described. To generate definitive machine codes neither loader nor linkage programs are required. The BASIC programming language makes possible to use the assembler on various types of computers.

The investigation has been performed at the Laboratory of Nuclear Problems, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1986