СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
ДУБНА

10 - 9768

Г.Е.Бабаян, Г.А.Ососков

ОПИСАНИЕ СПЕЦИАЛИЗИРОВАННОГО ЯЗЫКА **COPLAN**, ПРЕДНАЗНАЧЕННОГО ДЛЯ АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ НА УПРАВЛЯЮЩИХ МИНИ-ЭВМ

1976

10 - 9768

Г.Е.Бабаян, Г.А.Ососков

ОПИСАНИЕ СПЕЦИАЛИЗИРОВАННОГО ЯЗЫКА **COPLAN** , ПРЕДНАЗНАЧЕННОГО ДЛЯ АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ НА УПРАВЛЯЮЩИХ МИНИ-ЭВМ

I. ВВЕДЕНИЕ

Разработка управляющих программ для ЭВМ, работающих на линии с экспериментальной аппаратурой или измерительными автоматизированными устройствами, требует больших затрат труда на программирование в автокоде. Использование алгоритмических языков высокого уровня, таких, как ФОРТРАН или АЛГОЛ, оказывается невозможным, так как в отличие от автокода они не содержат средств работы с аппаратурой (обработка прерываний, специальные команды управления отдельными устройствами и т.д.). Кроме того, только автокод дает возможность максимальной оптимизации программ по занимаемой памяти и быстроте работы.

Помимо этой трудности можно отметить еще две проблемы, характерные для большеобъемных задач накопления данных и управления в реальном времени с использованием минимашин:

- ограниченность памяти мини-ЭВМ;
- нехватка необходимого количества машинного времени для отладки программ.

Все эти проблемы взаимосвязаны, поскольку именно большой объем таких программ, быстро заполняющих всю наличную память управляющей мини-ЭВМ, делает особенно мучительными любые работы по внесению изменений и требует поэтому дополнительных затрат времени на отладку, в то время как по самому своему назначению ЭВМ должна быть постоянно загружена своей непосредственной работой. Тем не менее, именно размеры работ по написанию *on-line* программ, выполняемых обычно не программистами, а самими разработчиками системы, т.е. инженерами или физиками, особенно нуждающимися в преимуществах языков высокого уровня,-

все это настоятельно требует автоматизации этих работ, т.е. введения средств для более краткой и понятной записи алгоритмов, освобождения от механического труда по их записи в автокоде, обеспечения в подготовке к вводу и отладке программы.

Было бы очень желательно ввести эти преимущества языков высокого уровня, сохранив необходимые средства для *on-line* управления, и создать новый промежуточный по уровню язык с фортраноподобными операторами и специальными инструкциями для управления устройствами и обработки прерываний. Эти машинно-зависимые операторы языка должны быть разумно сконструированы на базе основных обслуживающих программ-модулей, которые являются общими в разных программах. Конечно, операторы нового языка не должны быть эквивалентными целым подпрограммам, иначе язык стал бы слишком зависим от конкретной системы. С другой стороны, эти операторы должны выполнять функции каких-то одинаковых, общих для многих подпрограмм групп инструкций, чтобы упростить программирование и сократить размеры программ.

Машинно-зависимые операторы, естественно, должны быть дополнены другими необходимыми элементами любого языка: константами и переменными величинами, группой машинно-независимых операторов, с помощью которых образуются выражения (арифметические, логические, переходов, проверки условий для осуществления разветвления в программах и т.д.). Необходим оператор *FNCT* для организации замены формальных параметров подпрограмм на их фактические значения.

Для сохранения совместимости с имеющимися уже программами этот новый язык должен автоматически включать в себя все символы и операторы соответствующего автокода.

На основании такого рода соображений несколько лет тому назад одним из авторов был предложен язык *COPLAN* – специализированный язык для целей управления / I /.

За прошедшие годы появилось уже целое семейство подобных языков, некоторый обзор которых можно найти в содержательной работе Р.Рассела / 2 /, где проводится также детальное обсуждение преимуществ таких языков, промежуточных между автокодами и языками высоких уровней.

Предложенный Р.Расселом язык *PL/I* с успехом использовался при реализации проекта "Омега" в ЦЕРНе. Этот язык, в трансляторе которого использованы все возможности автокода ЭВМ РДР-II, богаче *COPLAN'а*, но у последнего есть одно важное свойство – простота. Это делает фортранный транслятор с языка *COPLAN* легко модифицируемым, максимально независимым от конкретных особенностей управляющей ЭВМ и прибора, а также позволяет использовать богатую практику ЦЕРНа и ОИЯИ передачи работ по трансляции и отладке програмы для мини-ЭВМ на большие машины / 3 /.

Тем же требованием простоты объясняются основные ограничения языка *COPLAN* (цифры в скобках относятся к номерам соответствующих разделов настоящей работы).

I. В языке *COPLAN* допустимы только целые константы, т.к.:

а) для целей управления величины с плавающей запятой, как правило, не нужны;

б) отечественные машины обычно не имеют аппаратной расширенной арифметики с плавающей запятой.

2. Имеют место ограничения на размеры констант (3.3), так как язык *COPLAN* ориентирован на малоразрядные ЭВМ (I2-I6 бит).

3. Имеют место ограничения на количество букв и цифр в идентификаторах, литералах, а также в индексах (4.I), обусловленные разрядностью машинных слов ЭВМ, на которых реализованы фортранные трансляторы с языка *COPLAN*, зависящие от того, сколько символов можно поместить в одном машинном слове этих ЭВМ.

4. Формат карт не произвольный, а фиксированный (7). Это ограничение наложено для простоты трансляции с языка *COPLAN*.

Несмотря на то, что *COPLAN* специально предназначен для написания программ управления, в последующем изложении соответствующие стандартные функции, реализующие машинно-зависимые операторы языка, предназначенные для целей управления, описаны довольно бегло и только для случая, когда управление осуществляется через командно-статусные регистры. Дело в том, что именно в силу зависимости стандартных функций от конкретных систем управления, разнообразие последних делает необозримым разнообразие видов таких функций управления. *COPLAN* предоставляет пользователю готовый, достаточно гибкий аппарат, с помощью которого тот легко сможет сам оформить имеющиеся подпрограммы, модули, управляющие работой отдельных устройств автомата или, например, блоков *CAMAC* / 4 /, в виде соответствующих стандартных функций языка.

Как показала практика, машинно-зависимые операторы управления осуществляются в большинстве случаев достаточно длинными последовательностями инструкций ЭВМ. Это и определило выбор аппарата стандартных функций для включения в программу машинно-зависимых операторов, а не макроопределений, как в случае машинно-независимых арифметических и логических операторов.

Ниже будет дано формальное описание языка COPLAN и автокода в бэкусовской нормальной форме (БНФ), описание формата карт. Подробное описание фортранного транслятора с языка для ЭВМ типа Электроника-100 на машине БЭСМ-6 дано в работе /5/.

Авторы считают своим приятным долгом поблагодарить А.А.Корнейчука за полезные замечания по тексту данной работы.

2. Основные символы, идентификаторы, числа.

<основной символ>::=<буква>|<цифра>|<ограничитель>

2.1. Буква

<буква>::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|

Буквы используются для образования идентификаторов в названии операций и других понятий языка.

2.2. Цифры

<цифра>::=<восьмеричная цифра>|8|9|

<восьмеричная цифра>::= 0|1|2|3|4|5|6|7|

Цифры используются для образования констант, идентификаторов и других понятий языка.

2.3. Ограничители

<ограничитель>::=<знак операции>|<скобка>|<разделитель>

<знак операции>::= +|-|*|/|GT|EQ|LT|

<скобка>::= (|)

<разделитель>::= ,

Ограничители имеют фиксированный смысл, который в большинстве случаев очевиден, а в остальных будет пояснятьсяся.

2.4. Идентификаторы

Синтаксис

<идентификатор> ::= <буква> | <идентификатор> <буква> |

<идентификатор> <цифра>

С е м а н т и к а. Идентификаторы служат для обозначения переменных, меток и массивов. Один и тот же идентификатор нельзя использовать для обозначения различных переменных или меток, за исключением того случая, когда эти переменные или метки, согласно описаниям программы, имеют несовместимые области действия.

Общее количество букв и цифр идентификатора, обозначающего массив (5.I), должно быть не более 3, а для остальных случаев - не более 6*).

Примеры

COUNT
SUM
XP33

Не допустимы *5CNQ* первый символ не буква
**INT* - " - - " - - " -
PARAMETR слишком длинный

2.5. Числа

Синтаксис

<число> ::= <последовательность цифр>

<последовательность цифр> ::= <цифра> | <последовательность цифр> <цифра>

*) Все примеры и ограничения в данной работе приведены для случая, когда применяется ЭВМ типа Электроника-I00, а машиной, на которой написан фортранный транслятор, является БЭСМ-6

<восьмеричное число> ::= <последовательность восьмеричных цифр>

<последовательность восьмеричных цифр> ::= <восьмеричная цифра> | <последовательность восьмеричных цифр>

<восьмеричная цифра>

Примеры

0
19
143
Не допустимы: 777777 слишком длинное
998 число 9 не восьмеричное

Семантика. В языке *COPLAN* используются только изобразимые числа и изобразимые восьмеричные числа, то есть числа могут быть числовыми константами.

3. Выражения

Синтаксис

<выражение> ::= <арифметическое выражение>

Семантика. В ходе выполнения рабочей программы может происходить вычисление значения некоторого выражения. Значение выражения есть некоторый набор двоичных знаков. Считается, что оно расположено в некотором регистре машины, имеющем ту же структуру и нумерацию разрядов, что и ячейки в оперативной памяти, и занимает целиком этот регистр.

Здесь уместно отметить, что в языке *COPLAN* есть аналог логических выражений, реализованных с помощью функций *OR* и *AND*

(3.4), значениями которых являются соответственно логическое произведение и логическая сумма.

3.I. Переменные

Синтаксис

<переменная>::=<простая переменная>|<переменная с индексом>

<простая переменная>::=<идентификатор>

<переменная с индексом>::=<идентификатор> (<индексное выражение>)

<индексное выражение>::=<число>|<арифметическое выражение>

Примеры

A12
SASHA
M(10)
A(I+2-1)

Не допустимы:

5B	первый символ не буква
MAXIMUM	длинная переменная
K(.1)	индексное выражение не является арифметическим выражением.
A(3,4)	

Семантика. Переменная – это наименование, данное некоторой одной величине. Эта величина может быть использована в выражениях для образования других величин и может быть изменена впоследствии посредством операторов присваивания.

Переменные с индексом обозначают слова, которые являются элементами одномерных массивов слов. Значение индексного выражения указывает на положение элемента в массиве.

3.2. Метка

Синтаксис

<метка>::= <идентификатор метки>

<идентификатор метки>::=<идентификатор>

Примеры

ALFHA
LOOP

Не допустимы: 537 числа не могут быть метками
* не является идентификатором
APSGEN1 длинный идентификатор.

Семантика. Оператор на языке COPLAN может быть помеченным (иметь метку). Метки используются для ссылок на операторы (например, для указания того, какой оператор должен выполняться следующим).

3.3. Константы

Синтаксис

<константа>::=<числовая константа>|<программная константа>

<числовая константа>::=<число>

<программная константа>::=<литерал>

<литерал>::= =<число>| =<идентификатор>

Примеры

199
77B
=11
=AX

Не допустимы: 20000 много цифра в константе
7118 8 - не восьмеричное число
=APPAIK идентификатор интеграла длинный.

Семантика. Если за последовательностью цифр в константе следует буква B, то константа задаёт число в восьмеричной системе счисления, иначе-в десятичной.Максимальное значение константы зависит от количества разрядов в машинном слове ЭВМ, для которой генерируется программа в машинном коде.

Литерал принимает числовое значение, равное адресу соответствующей переменной или константы. Например, значение литерала =AX равно адресу, по которому в памяти находится переменная AX; значение литерала =1 - адрес константы "1".

Общее количество букв и цифр идентификатора в литерале не должен превышать 5.

3.4. Функции

Синтаксис

<функция>::=<идентификатор функции>(<список фактических параметров>)

<идентификатор функции>::=<идентификатор>

<список фактических параметров>::=<фактический параметр>|<список фактических параметров>,<фактический параметр>

<фактический параметр>::=<выражение>

Примеры

```
SIN(A-B)
T(X,Y)
RSH4(AND(WORD1,360))
ARG(=LIT)
```

Семантика. Функция определяет одно числовое значение, которое является результатом применения заданной совокупности правил, определяемой описанием функции, к фиксированной совокупности фактических параметров.

3.5. Стандартные функции

К стандартным функциям относятся все машинно-зависимые операторы языка *COPLAN*, необходимые для управления устройствами и обработки прерываний. В силу зависимости стандартных функций от конкретных систем управления, разнообразие последних делает необозримым различие видов таких функций.

Ниже приводится список некоторых из них, которые могут быть использованы в случае метода *on-line* управления посредством командно-статусных регистров измерительных устройств.

Рассмотрение структуры подпрограмм, реализующих аналогичные стандартные функции для различных подустройств, показывает, что этим подпрограммам можно придать вид, когда они будут отличаться только кодами специальных дополнительных команд, введенных для операций с командно-статусными регистрами. Такое единообразие вида стандартных функций позволяет использовать следующий прием: каждому подустройству присваивается номер $\mathcal{D}$, который используется как параметр в данной функции, определяющий, к какому устройству она относится. По этому параметру ЭВМ выбирает код требуемой спецкоманды и подставляет его в соответствующее место подпрограммы, реализующей данную функцию.

Подобная же, но более общая идея содержится в предложениях по языку для управления электронными устройствами, выполненных

в стандарте *САМАС* / 4 /, где код управляющей функции определяет конкретное подустройство (крейт, блок, субадрес) и саму управляющую функцию.

Для того, чтобы управлять подустройством D с помощью командно-статусного регистра, необходимы стандартные функции, позволяющие считать статусные данные об устройстве, проанализировать их и отдать соответствующую команду через командный регистр (часть командного регистра). Пусть это будет каретка в приборе, измеряющем декартовы координаты изображения, проектируемого с фильма. Для ее передвижений, которые дадут возможность провести измерения, нам потребуются следующие функции:

READV(D) – считывает содержимое регистра скорости подустройства D в младшие разряды сумматора.

READC(D) – считывает содержимое счетчика положений подустройства в младшие разряды сумматора.

SENSE(D,J) – равна содержимому бита J командно-статусного регистра.

SETS(D,A) – вводит в командно-статусный регистр содержимое ячейки A.

SETV(D,A) – вводит в регистр скорости содержимое ячейки A.

SET0(D,J) и *SET1(D,J)* – вводят в бит J командно-статусного регистра соответственно 0 и I.

SEEK(D,A,LOC) – устанавливает подустройство D положение, при котором содержимое счетчика положения D будет равно содержимому ячейки A. После управление будет передано ячейке *LOC*.

SEEKS(A,B,LOC) – X и Y каретки устанавливает в положение с координатами $X=A$, $Y=B$, затем передает управление ячейке *LOC*.

Пример: Полагая, что *X* и *Y* каретки имеют номера I и 2, соответственно, следующая последовательность команд установит эти каретки в положение с координатами (5517, I237) и проверит, что она была установлена точно:

```
          SETV(1,100)
          SETV(2,100)
     A =  5517
          SEEKS(A,1237,NEXT)
          DELAY(1000)
NEXT TEST1 = ABS(A-READC(1))
     TEST2 = ABS(1237-READC(2))
          IF(ABS(10-TEST1-TEST2),GE,ERROR1)
          . . . . . .
          . . . . . —
```

Здесь была использована функция *DELAY(TIME)*, которая осуществляет ждущий цикл системы (до появления прерываний или до конца интервала времени *TIME* мсек). Поскольку функция *DELAY* организуется посредством простого зацикливания, возможно ее использовать также в случае отсутствия часов. С другой стороны, это означает, что эта функция должна быть скорректирована для каждой данной машины.

Этот список должен быть дополнен функциями для обработки прерываний.

Приведем также примеры функций ввода-вывода, предназначенных для непосредственной связи между оператором и вычислительной машиной с помощью телетайпа или алфавитно-цифрового дисплея.

KEYOCT(A) – вводит одно восьмеричное число (до 6 цифр). Если число состоит менее чем из 6 цифр, то оно должно следовать за пробелами.

KEYONE – вводит отдельный символ от клавиатуры и помещает в сумматор.

PRINT(TEXT) – печатает в коде ВСД сообщение, помещенное в памяти, начиная с ячейки TEXT. Первые шесть битов ячейки TEXT должны определять восьмеричное число N, указывающее длину сообщения.

TYPNUM(=A,N) – печатает содержимое сумматора как восьмеричное число.

3.6. Подпрограммы

Синтаксис

< подпрограмма > : : = < идентификатор подпрограммы > |

< идентификатор подпрограммы > (< список фактических параметров >)

Примеры

SEARCH(X1, X2, X3)

WRITEB

Семантика. Имеется два типа подпрограмм в языке *COPLAN*. Подпрограммы, которые не используют фактические параметры, а только переменные, помещенные заранее в общие блоки, и подпрограммы, которые используют фактические параметры. В первом случае подпрограммы вызываются с помощью оператора CALL (см. 4.4), а во втором NAME(p_1,...,p_n) где *NAME* – имя подпрограммы, а $p_1 \dots p_n$ – фактические параметры ($n \leq 6$).

Подпрограммы определяют массив числовых значений, которые являются результатом применения заданной совокупности правил, определяемой описанием подпрограммы.

3.7 Арифметическое выражение

Синтаксис

⟨операция типа сложения⟩::= +|-

⟨операция типа умножения⟩::= *|/

⟨первичное выражение⟩::= ⟨константа⟩|⟨переменная⟩|

⟨функция⟩

⟨терм⟩::= ⟨первичное выражение⟩|⟨терм⟩⟨операция типа умножения⟩⟨первичное выражение⟩

⟨арифметическое выражение⟩::= ⟨терм⟩|⟨операция типа сложения⟩⟨терм⟩|⟨арифметическое выражение⟩

⟨операция типа сложения⟩⟨терм⟩

Примеры:

137	A(I)
-75	A(I)*5-DELTA
XORD	SIN(X)*SIN(X)+COS(X)*COS(X)
=25	(A-F(X))*3-(C+B*K(1*2-1)

Семантика. Арифметическое выражение определяет последовательность операций, которые должны быть выполнены слева направо, с учетом следующих правил:

а) по отношению к синтаксису выдерживается следующий порядок старшинства:

первый	*	/
второй	+	-

б) выражение между левой скобкой и соответствующей правой скобкой вычисляется самостоятельно, и его значение используется в дальнейших вычислениях. Желаемый порядок выполнения операций всегда может быть достигнут соответствующей расстановкой скобок.

4. Операторы

Операторы описывают действия, которые должны быть выполнены для решения задачи. Обычно они выполняются в той последовательности, в которой они написаны. Однако эта последовательность выполнения может быть изменена оператором перехода или условным оператором. Для того, чтобы иметь возможность фактически указывать порядок следования операторов в процессе работы, операторы должны снабжаться метками.

Синтаксис

<оператор>::= <непомеченный оператор>|<помеченный оператор>

<помеченный оператор>::=<метка><непомеченный оператор>

<непомеченный оператор>::=<описание>|<команда автокода>|<оператор присваивания>|<оператор безусловного перехода>|<оператор условного перехода>|<оператор возврата>|<оператор вызова подпрограммы>|<оператор замены формальных параметров на фактические параметры>|<оператор начала>|<оператор конца>.

4.I. Оператор присваивания

Синтаксис

<оператор присваивания>::=<левая часть оператора присваивания>=<правая часть оператора присваивания>

<левая часть оператора присваивания> ::= <переменная>

<правая часть оператора присваивания> ::= <выражение>

Примеры

```
ADK(I)  =  SUM
Y       =  FIRST(X,Y)
K1      =  A+B*C
```

Не допустимы:

```
IAS(IL) =  5        индексное выражение в левой части не однобуквенная переменная
KR(A*2-I) = X       
M+N     =  Y1       левая часть не переменная
B RIQ   =  /B       переменная в правой части начинается ни с буквы.
```

Семантика. Операторы присваивания служат для присваивания значения выражения переменной. Если в левой части оператора присваивания имеется индексное выражение, оно должно быть обязательно однобуквенной простой переменной.

4.2. Оператор безусловного перехода

Синтаксис

<оператор безусловного перехода> ::= GOTO <метка>

Примеры

```
GOTO  LOOP
GOTO  SASHA
```

Семантика. Операторы перехода прерывают естественную последовательность выполнения операторов, определяемый порядком их написания. Они определяют следующий выполняемый оператор по метке.

4.3. Оператор условного перехода

Синтаксис.

‹оператор условного перехода›::= IF (‹ выражение ›, ‹ знак операции отношения ›, ‹ метка›)

‹знак операции отношения›::= GT|LT|EQ|

Примеры

```
IF(A,LT,L001)
IF (K*3-4,GT,MET)
IF (IOP,EQ,KL)
```

Семантика. Операторы условного перехода приводят к пропуску или выполнению некоторых операторов в зависимости от текущего значения выражения и условия. Если выражение больше, меньше или равно нулю, то соответственно при условии выполнения данного условия либо совершается переход к оператору с меткой, указанной в операторе, либо выполняется следующий оператор в последовательности.

Оператор условного перехода на перфокартах пробивается в поле выражения (см. 7).

4.4. Оператор вызова подпрограммы

Синтаксис

‹оператор вызова подпрограммы›::= CALL ‹идентификатор подпрограммы ›

Примеры

```
CALL SUM
CALL SQRT
```

Семантика. Этот оператор служит для вызова подпрограмм, не использующих фактические параметры. Как только встречается этот оператор, управление передается к данной подпрограмме с запоминанием точки возврата.

4.5. Оператор возврата

Синтаксис

<оператор возврата>::=EXIT<идентификатор функции>|

EXIT<идентификатор подпрограммы>

Примеры

```
EXIT  MAX
EXIT  TRIG
```

Семантика. Этот оператор используется в описании функции и подпрограмм для возврата управления вызывающей программе.

4.6. Оператор замены формальных параметров на фактические.

Синтаксис

<оператор замены формальных параметров на фактические>::=

FNCT(<список формальных параметров>)

<список формальных параметров>::=<формальный параметр>|

<список формальных параметров>,<формальный параметр>

<формальный параметр>::=<идентификатор>

Примеры

```
FNAME     FNCT(A,X)
GAME      FNCT(X,Y,Z)
```

Семантика. Этот оператор всегда употребляется с меткой, являющейся именем функции или подпрограммы и применяется для организации замены формальных параметров на их фактические значения. На перфоркартах оператор замены формальных параметров на фактические пробивается в поле выражения (см. 7).

4.7. Оператор начала

Синтаксис

<оператор начала>::= CORG <восьмеричное число>

Пример

CORG 1777

Семантика. Оператор начала дает указание загрузчику, с какой ячейки памяти помещать данную программу.

4.8. Оператор конца

Синтаксис

<оператор конца>::= END | FIN

Семантика. Оператор END относится к концу данного сегмента программы. Оператор FIN является признаком конца для ассемблера.

5. Описание.

<описание>::= <описание массива> | <описание общих блоков> | <описание эквивалентных имен> | <описание констант> | <описание внутренних точек> | <описание внешних точек> | <описание подпрограммы> | <описание функции>

5.1. Описание массива

Синтаксис

<описание массива>::= DIMEN(<список массивов>)

<список массивов>::= <массив> | <список массивов>,<массив>

<массив>::= <идентификатор массива> (<размерность>)

<идентификатор массива>::= <идентификатор>

<размерность>::= <число>

Примеры

```
DIMEN   A(10), B(33), ALT(40)
DIMEN   KPP(132)
```

Семантика. Это невыполняемый оператор, который в программе должен стоять раньше всех выполняемых операторов. Он резервирует области памяти. Число, обозначающее размерность,- десятичное.

5.2. Описание общих блоков

Синтаксис

<описание общих блоков>::= COM(<список массивов и переменных>)

<список массивов и переменных>::= <список массивов> | <список переменных> | <список массивов и переменных>,<список переменных> | <список массивов и переменных>,<список массивов>

<список переменных>::= <переменная> | <список переменных>,<переменная>

<переменная>::=<идентификатор>

Примеры

```
COM   A,B,C
COM   A(10),BUB(10),CENT(12)
COM   ARG(11),DIG,L,B(20),DIMS
```

Семантика. Это невыполняемый оператор, который в программе должен стоять раньше всех выполняемых операторов. Он резервирует общие области памяти для программы и его подпрограмм и подпрограмм-функций. Правило пользования COM блоками такое же, как для непомеченных общих блоков в языке ФОРТРАН /6 /.Число, обозначающее размерность массивов в общих блоках,- десятичное.

5.3. Описание эквивалентных имён

Синтаксис

<описание эквивалентных имён>::= EQU <переменная>

Примеры

```
A    EQU  B
XI2  EQU  YI2
```

Семантика. Это невыполняемый оператор, который в программе должен стоять раньше всех выполняемых операторов. Он двум переменным (одна из которых метка) отводит одну и ту же ячейку памяти.

5.4. Описание константы

Синтаксис

<описание константы>::= CON<числовая константа>

Примеры

```
A   CON  137
B   CON  -16B
```

Семантика. Это невыполняемый оператор, который в программе должен стоять раньше всех выполняемых операторов. Он резервирует ячейку с данным именем (метка) и помещает туда значение константы.

5.5. Описание входов

Синтаксис

<описание входов>::= ENTRY<список переменных>

<список переменных>::= <переменная>| <список переменных>, <переменная>

Примеры

```
ENTRY   NAG
ENTRY   BUB,XIT,DIK
```

Семантика. Это невыполняемый оператор, который в программе должен стоять раньше всех выполняемых операторов. Если необходимо иметь несколько входов в программу для того, чтобы использовать её отдельные части, используется этот оператор.

5.6. Описание внешних имён.

Синтаксис

<описание внешних имён>::=EXTERN<список переменных>

<список переменных>::= <переменная>|<список переменных>,<переменная>

Примеры

```
EXTERN  NAG
EXTERN  BOB,XIT,DIK
```

Семантика. Это невыполняемый оператор, который в программе должен стоять раньше всех выполняемых операторов. Если в данной программе есть ссылка на вход в другую программу, то этот вход оператором EXTERN описывается как внешнее имя в данной программе.

5.7. Описание функции.

Синтаксис

<описание функции>::= FUNCT<идентификатор функции><тело функции>

<тело функции>::= <список операторов>

<список операторов>::= <оператор>|<список операторов><оператор>

Примеры:

1.

```
*ABS        FUNCT       ABS
                        FNCT(A)
                        IF(A,GE,POS)
            A       =   -A
            CLA
            TAD         A
            EXIT        ABS
POS         CLA
            TAD         A
            EXIT        ABS
            END
```

2.

```
            FUNCT       F(X,Y)
            EXTERN      DIB
            COM         C,D
            DIMEN       A(5)
K           CON         0
F                       FNCT(X,Y)
LOOP        K       =   K+1
            A(K)    =   DIB(C,D)
                        IF(K-5,EQ,LOOP1)
            GOTO        LOOP
LOOP1       F           (A(1)*A(4)-A(2))/A(3)+A(5)
            EXIT        F
            END
```

Семантика. Описание функции служит для задания алгоритма вычисления значения функции. Все невыполняемые операторы должны стоять в теле описании функции раньше всех выполняемых операторов. Первым выполняемым оператором в описании функции должен обязательно быть оператор замены формальных параметров на фактические.

5.8. Описание подпрограммы

Синтаксис

<описание подпрограммы>::=SUBR <идентификатор подпрограммы> <тело подпрограммы>

<тело подпрограммы> ::= <список операторов>

<список операторов> ::= <оператор> | <список операторов> <оператор>

Примеры :

```
1.          SUBR     SQSUM
            COM      BIP(10), GRIF, DOB
            EXTERN   CALIBR
            ENTRY    LOOP
     SUBR   Ø
     A      CON      137B
            GRIF  =  CALIBR (DOB)
                     SEEKS (GRIF,1029,LOOP)
            K     =  DOB*DOB/15
            BIP(4) = K-GRIF
            EXIT     SQSUM
            END
```

Это подпрограмма вызывается в другой программе так: CALL SQSUM

```
2.          SUBR     ZERO
            DIM      S(10)
     ZERO            FNCT(A,N)
            K     =  =A-2*N
     LOOP   K     =  K+1
            S(K)  =  0
                     IF(10-K,GE,LOOP)
            EXIT     ZERO
            END
```

Это подпрограмма вызывается так: ZERO (BUFF, 200)

Семантика. Описание подпрограммы служит для того, чтобы показать, что некоторая группа операторов будет выполняться как нечто целое, и дать этой группе имя. Все невыполняемые операторы должны стоять в теле описания подпрограммы раньше всех выполняемых операторов. Заметим, что в зависимости от типа под-

программы (3.6) первым выполнимым оператором в теле описания подпрограммы должен быть либо

SUBR Ø

(когда подпрограмма не использует фактические параметры), либо оператор замены формальных параметров на фактические.

6. Включение автокодных операторов и программ.

В исходной программе, написанной на языке COPLAN, допускаются как отдельные команды автокода данной машины, так и отдельные сегменты (подпрограммы, подпрограммы-функции) на языке автокода.

Правила написания программы на автокоде, допустимого на языке COPLAN, изложены ниже в виде перечня понятий / 7 /. Для краткости и однозначности изложения перечень понятий дан в виде последовательности формул. В этих формулах названия понятий заключены в скобки. Определяемое понятие находится слева от знака, определяющее - справа.

‹автокодная программа›::= ‹последовательность автокодных команд›

‹последовательность автокодных команд›::=‹автокодная команда›|‹последовательность автокодных команд›‹автокодная команда›

‹автокодная команда›::=‹комментарий›|‹команда›|‹псевдокоманда›*

---

*) Допустимо применение только основных псевдокоманд используемых автокодов.

<комментарий>::=/<текст>

<команда>::=<метка> <код операции> <адрес>

<адрес>::=<простой адрес>

<простой адрес>::=<истинный адрес> | <относительный адрес> | <символический адрес>

<истинный адрес>::= <восьмеричные цифры>

<относительный адрес>::= *<признак операции над адресами> <восьмеричные цифры>

<символический адрес>::=<буква> | <символический адрес><буква> | <символический адрес><цифра>

<признак операции над адресами>::= <признак сложения адресов> | <признак вычитания адресов>

<признак сложения адресов>::= +

<признак вычитания адресов>::= -

<восьмеричная цифра>::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

<код операции>::= <символ операции> | <восьмеричное число>*

7. Формат карт

Текст программы, написанной на языке *COPLAN*, пробивается на перфокартах. Для пробивки карт можно использовать перфоратор *УПЛ*, а также перфораторы, пробивающие один символ в колонке и печатающие на верхней кромке карты содержание карты (ICL, ARITMA).

Каждый оператор *COPLAN*'а всегда занимает только одну перфокарту. Для пробивки оператора используется фиксированный формат, содержащий пять полей перфокарты.

---

*Все комбинированные команды ассемблера определяются их действительными значениями - восьмеричными числами. Это ограничение исходит из фиксированного формата карт.

Поле, содержащее колонки с I по 7, используется для метки (если на данной карте перфорируется команда автокода, то метка команды автокода).

Колонки с I0-I5 - для левой части оператора присваивания (если команда автокода, то код операции).

Знак присваивания в операторе присваивания должен находиться в любой из колонок от I6 до I9 включительно.

Колонки с 20 по 72 отводятся для пробивки выражений или адресов команд автокода.

Колонки с 73 по 80-для комментариев.

При пробивке на перфокартах в любом из указанных полей для удобства между символами можно оставлять пробелы, но их количество не должно превышать четырех. В противном случае остальная часть перфокарты транслятором не воспринимается. Например: *NIKOLA* и *N␣␣I␣␣␣KOLA* транслятором будут восприниматься как индентичные, а при *N␣␣␣␣␣I␣KOLA* транслятор воспримет только символ *N* и *IKOLA*

Если в первой колонке стоит знак / , то данная карта понимается как комментарий.

Литература

I. G.Ososkov. CERN/D.Ph. 11/PROG 70-6, 1970.

2. R.Russel. CERN/DD/72/15.

3. Мини-ЭВМ (под редакцией Опперля). М., Мир, I975.

4. CAMAC bulletin No. 5, November, 1972.

5. Г.Е.Бабаян, Г.А.Ососков. Сообщение ОИЯИ II-9766, Дубна, 1976.
6. С.С.Лавров. Универсальный язык программирования. М., Наука, 1967.
7. К.М.Железнова, А.А.Корнейчук и др. Сообщение ОИЯИ 10-7904, Дубна, 1974.

Рукопись поступила в издательский отдел
11 мая 1976 года.