ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
1973 год Выпуск 57

УДК 681.142.2

ЯЗЫК ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ

Н.К.Гришаева, В.Г.Кербель, Ю.И.Колосова, В.И.Константинов, В.Д.Корнеев, Т.А.Левагина, Н.Н.Миренков, С.Б.Фишерман

Описывается входной язык для записи р-алгоритмов, являющийся частью средств программирования однородных вычислительных систем (ОВС-язык). ОВС-язык позволяет создавать параллельные программы на основе р-алгоритмов, представляющих собой совокупность идентичных р-ветвей, полученных распараллеливанием по циклам [1], или совокупность неидентичных р-ветвей, полученных, в общем случае, распараллеливанием по крупным программным блокам [2]. В обоих случаях объем параллельной программы отличается от объема последовательной программы (алгоритм которой положен в основу р-алгоритма), как правило, совокупностью операторов обменных взаимодействий. Число этих операторов зависит от разделения информации между ветвями параллельного процесса [3] и обычно составляет небольшую долю от общего объема программы.

Основа ОВС-языка - средства для задания обменных взаимодействий р-ветвей: системные операторы и системные стандартные функции. Вместе с другими операторами языка (в основном это операторы языка БЕЙСИК [4]) они образуют набор, позволяющий записывать как последовательные, так и параллельные вычислительные процессы.

§ 1. Начальные сведения о языке

1.1. Рассмотрим процесс решения алгебраического уравнения второй степени $Bx^2+Cx+D=0$. Если $C^2-4BD \geq 0$, то существуют действительные решения, вычисляемые по формуле

$$x_{1,2}=\left(-C \pm \sqrt{C^2-4BD}\right)\cdot(2B)^{-1}.$$

Последовательная программа решения таких уравнений имеет на рассматриваемом языке следующий вид:

```
I0  DIM B,C,D,X1,X2,E,F
20  INPUT 1,A(B,C,D)
30  LET E = C↑2-4*B*D*)
40  IF E 'LT' 0 THEN 110**)
50  LET E = 'SQR'(E)
60  LET F = 2*B
70  LET X1 = (-C+E)/F
80  LET X2 = (-C-E)/F
90  OUTPUT 1,A(X1,X2)
I00 STOP
II0 OUTPUT 1,A("NOT REAL")
I20 END
```

Отметим несколько особенностей, видных уже на этом простом примере. Программа есть последовательность операторов (см. табл. 3, с. 43), каждый из которых начинается с заголовка, записанного заглавными латинскими буквами**). Перед заголовком стоят номера строк (операторов), определяющие порядок выполнения операторов. Это позволяет вводить программу с любым порядком следования операторов. Перед реализацией транслятор расстанавливает операторы согласно их номерам и редактирует программу.

Для удобства записи и чтения пробелы в программе можно расставлять произвольно, за исключением сообщений, выделенных кавычками в операторе OUTPUT . Например, оператор 60 можно записать как 60 LET F = 2 * B .

Рассмотрим теперь встретившиеся в программе операторы.

Оператор описания DIM резервирует место в оперативной памяти для переменных программы.

*) В реализации вместо отсутствующего на клавиатуре знака используется буква Э (см. п. I.2).

**) Ограниченность клавиатуры телетайпа для "Минск-22" несколько утяжеляет исходные программы (так, необходимо писать 'LT' вместо < , использовать только заглавные буквы).

Оператор ввода INPUT N,A задает автономный (A) ввод*) данных с одного из внешних устройств (N = I соответствует телетайпу, N = 2 - перфоленте, N = 3-магнитной ленте). В скобках перечисляются идентификаторы вводимых переменных.

В случае ввода с телетайпа последовательность взаимодействий следующая. Машина печатает B = . Пользователь продолжает начатую строку, печатая нужное ему значение в следующем виде:

знак числа	точка	мантисса	E	знак порядка	порядок	X -признак конца

где мантисса содержит от I до 7 десятичных цифр, порядок - две цифры.

Например:
```
B = +.I00 8567   + 07 X
C = -.296I       + 02 X
D = +.74I0000    - 0I X
```

Оператор присваивания LET присваивает переменной (или списку переменных) значение выражения, стоящего справа от знака равенства. Строка LET X,Y = 2*C эквивалентна строкам:

```
LET X = 2*C
LET Y = 2*C.
```

Оператор условного перехода IF-THEN проверяет выполнение условия, заданного формулой, стоящей между IF и THEN . 'LT' - означает функцию отношения "меньше" (см. п. I.2.). При невыполнении условия передает управление следующему за ним оператору.

Оператор вывода OUTPUT N,A задает автономный вывод цифровой и текстовой информации на внешние устройства (N = 4 соответствует быстропечатающему механизму (ТНМ), 5 - алфавитно-цифровому печатающему устройству (АЦПУ).

Оператор STOP - оператор программного останова.

Оператор конца программы END стоит в строке с наибольшим номером.

Обратим внимание на выбор номеров строк. Номера можно выбирать произвольно, лишь бы операторы выполнялись в том порядке, в каком требуется. Можно пронумеровать их: 1,2,...,I2, но это не

*) Обычный ввод в последовательной программе, или ввод, независимый от других р-ветвей, в параллельной программе.

рекомендуется. Лучше нумеровать I0,20,...,I20, чтобы иметь воз - можность вставить позже дополнительные операторы.

З а м е ч а н и е : номер любого оператора должен быть отличным от нуля.

I.2. Ф о р м у л ы . На ОВС-языке можно задавать вычисления всевозможных формул, использующих обычный набор арифметических и логических операций: +, -, /, x, ↑ , AND, OR и т.д. (примеры приведены в табл. I.

Т а б л и ц а I

Знак операции	Символ языка	Пример	Значение
+	+	A + B	A сложить с B
-	-	A - B	из A вычесть B
x.	x	A x B	A умножить на B
/	/	A / B	A разделить на B
↑	Э	X32	X возвести в квадрат
NOT	'NOT'	'NOT'X	Результат = 0{X≠0},иначе = I
AND	'AND'	A'AND'B	Результат = I{A≠0,B≠0} ,иначе=0
OR	'OR'	A'OR'B	Результат = 0{A=0,B= 0},иначе=I
>	'GT'	X ≡ A'GT'B	x = I, если A больше B
<	'LT'	X ≡ A'LT'B	x = I, если A меньше B
≤	'LE'	X ≡ A'LE'B	x = I,если A меньше или равно B
≥	'GE'	X ≡ A'GE'B	x = I,если A больше или равно B
=	=	X ≡ A = B	x = I, если A = B
≠	'NE'	X ≡ A'NE'B	x = I, если A не равно B

Кроме арифметических и логических операций, в формулах используются стандартные функции (табл. 2).

Т а б л и ц а 2

Стандартные функции	Символы языка	Стандартные функции	Символы языка
sin x	'SIN'(X)	|x|	'ABS'(X)
cos x	'COS'(X)	lg x	'LOG'(X)
tg x	'TNG'(X)	$\sqrt{x}$	'SQR'(X)
Arc tg x	'ATN'(X)	[x]	'INT'(X)
e^x	'EXP'(X)	sign x	'SGN'(X)

При написании формул нужно учитывать то, что выражения в скобках вычисляются самостоятельно, например, (A↑B)↑C ≠ A↑(B↑C) . Для достижения большей наглядности записи формул рекомендуется пользоваться дополнительными скобками.

Ч и с л а , встречающиеся в формулах, могут быть положительными или отрицательными и содержать не более семи значащих десятичных цифр:

3456; 2. 84I; . 03567I8; I..

Для обозначения десятичного порядка употребляется буква E.

Например, число I285 можно записать как I.285E3 или +I2850E - I, число I000 0000 - как IE7, но не E7.

Ч и с л о в ы е п е р е м е н н ы е, входящие в формулы, представляются буквой или буквой с последующей единственной цифрой: A7,B,X, N9 и т.д.

Каждая переменная должна быть описана, причем описание должно предшествовать использованию переменной.

Таким образом, формула есть последовательность, составленная из знаков операций, скобок, чисел, переменных, стандартных функций (и функций пользователя, см. I.6). Например,

(A + BX'LOG'(X))↑N - 'SIN' (3,14),

(-0.56E3XA - 2XBX 'COS'(V↑2))/C.

I.3. Ц и к л ы . Для компактной записи вычислений, содержащих повторяющиеся участки, используются циклы.

Они могут задаваться двумя способами: с помощью уже рассмотренного оператора IF-THEN или с помощью специальных операторов

цикла FOR и NEXT . Например, пусть требуется вычислить значение функции e^x для x = 1,2,3,...,10 и отпечатать на АЦПУ таблицу этих значений.

Первый способ:

```
10 'DIM X,Y
20  LET X = 1
30  LET Y = 'EXP'(X)
40  OUTPUT 5,A(Y)
50  LET X = X+1
60  IF X'LE' 10 THEN 30
70  END
```

Второй способ:

```
10  DIM X,Y
20  FOR X = 1 TO 10      } Заголовок цикла
30  LET Y = 'EXP'(X)     }
40  OUTPUT 5,A(Y)        } Тело цикла
50  NEXT X               } Конец цикла
60  END
```

Допускаются более сложные формы оператора заголовка цикла FOR :

а) с указанием шага цикла, например:

```
FOR    X = I      TO 10      STEP 2
       нижняя     верхняя        шаг
       граница    граница
```

(если шаг цикла не указан, то он считается равным единице);

б) с использованием в качестве верхней границы, нижней границы и шага формул любой сложности.

Перед каждым выполнением тела цикла проверяется условие: не больше ли (не меньше ли для отрицательного шага) верхней границы значение параметра цикла.

Циклы могут быть вложенными, но не должны иметь частичных пересечений. Это иллюстрирует следующая схема:

Можно

```
FOR x
  FOR y
  NEXT y
NEXT x
```

Можно

```
FOR x
  FOR y
    FOR z
    NEXT z
    FOR w
    NEXT w
  NEXT y
    FOR z
    NEXT z
NEXT x
```

Нельзя

```
FOR x
  FOR y
NEXT x
NEXT y
```

I.4. М а с с и в ы. В дополнение к простым переменным языка для обозначения элементов массивов существуют переменные с индексом. Они образуются из единственной буквы, являющейся именем массива, и следующих за ней индексов в круглых скобках.

Переменные с индексом описываются с помощью оператора DIM. Пример вычисления скалярного произведения векторов при этом будет иметь вид:

```
10  DIM A(25), B(25), C, I
20  INPUT 1,A(A,B)
30  LET C=0
40  FOR I = 1 TO 25
50  LET C = C+A(I)*B(I)
60  NEXT I
70  END
```

Массивы A и B в данном примере одномерные, но в языке допускаются массивы любой размерности, например:

D(25,13,41), G(N,M,L,K).

При описании массивов используются простые переменные и целые без знака, которые задают допустимые верхние границы изменения индексов; нижняя граница - фиксирована, считается равной единице. При использовании простых переменных в качестве верхней границы берется 'INT'(N+0,5).

Переменные с индексом употребляются наряду с простыми переменными, числами и т.д. В качестве текущего значения индекса может быть использована формула любой сложности, не содержащая переменной с индексом,например: A(('SIN'(X)+5),12,3XL) .При этом значения индексов должны находиться в пределах заданных границ.

I.5. П о д п р о г р а м м ы. Если некоторая часть программы используется более одного раза или в разных местах исходной программы, то ее удобно задавать как подпрограмму. Передача управления на подпрограмму осуществляется оператором GOSUB , в котором стоит номер строки, являющейся началом подпрограммы. Подпрограмма начинается оператором SUB , заканчивается оператором RETURN . Последний осуществляет возврат на первый оператор за GOSUB .

Вход в подпрограмму возможен только через оператор SUB, посредством передачи управления ему от GOSUB ,выход - только через оператор RETURN .

GOSUB может также использоваться в теле подпрограммы для вызова другой подпрограммы (вложение подпрограммы). Рекурсивные вложения не допускаются:

Можно

```
 40  GOSUB 100
     . . .
 60  GOSUB 160
     . . .
 90  STOP
100  SUB
     . . .
130  GOSUB 160
     . . .
150  RETURN
160  SUB
     . . .
180  RETURN
190  END
```

Нельзя

```
 90  SUB
     . . .
110  SUB
     . . .
130  RETURN
     . . .
150  RETURN
160  END
```

Нельзя

```
 10  SUB
     . . .
 40  GOSUB 100
     . . .
 60  RETURN
     . . .
100  SUB
     . . .
140  GOSUB 10
     . . .
160  RETURN
170  END
```


I.6. Ф у н к ц и и п о л ь з о в а т е л я. Часто встречающиеся выражения, которые можно записать в одной строке,удобно описывать не подпрограммами, а функциями пользователя, задаваемыми с помощью оператора DEF . Имя такой функции должно состоять из трех букв, первые две из которых есть FN , например, FNA,FNB. Оператор DEF может появиться в любом месте программы. В правой части равенства в нем допускается любая формула, включающая в себя различные комбинации других функций, в том числе определенных с помощью DEF . Рекурсивное использование функций пользователя не допускается:

можно:

```
DEF FNA (X) = 'COS'('EXP'(X↑2 - 1))
LET Z = (2+C) ↑3
LET Y = 'SIN'(Z) + FNA(Z+1) ↑ 2
```

нельзя:

```
DEF FNB(X) = X↑2 + FNB(A+B+C)
```

I.7. О п е р а т о р ы GOTO, CALL, REM.

Оператор безусловной передачи управления GOTO изменяет естественный порядок выполнения операторов; например, GOTO 100.

Оператор вызова библиотечных подпрограмм CALL вызывает библиотечные подпрограммы, записанные в кодах машины. Номер подпрограммы - положительное число,указывается сразу после заголовка оператора; параметры подпрограммы - в скобках через запятую, например: CALL II(2,A), где A - идентификатор простой переменной или массива.

Оператор комментария REM позволяет писать после себя любую информацию о программе, идентификаторах и т.д. Используется при выдаче листинга (распечатки программы), например:

```
 10  REM PROGRAMMA N5, IWANOW
 20  DIM A,B,C,D
     . . .
120  END
```

§ 2. Операторы системных взаимодействий

2.I. Рассмотрим решение системы линейных уравнений X = BX+G методом последовательных приближений. Ветвь параллельной программы на ОВС-языке при распределении горизонтальными полосами для B,X и G [3] имеет следующий вид:

```
 I0  DIM N,D, Z, E
 20  INPUT 1, PD(N,E)
 30  LETD = 'DEL'(N)
 40  CALL 100 (N,Z)
 50  DIM F,I, J, X(N), Y(D), G(D), B(D,N), L(D)
 60  INPUT 2,A(B,G)
 70  EXCHANGE TC (G,X)
 80  GOSUB 245
 90  FOR I = 1 TO D
I00  LET F = 0
II0  FOR J = 1 TO N
I20  LET F = F+B(I,J) X X (J)
I30  NEXT J
I40  LET L(I) = G(I)+F
I50  NEXT 1
I55  EXCHANGE TC(L,X)
I60  FOR J =1 TO D
I70  IF 'ABC'(X(J+Z-1)-Y(J)) 'GE' E THEN 210
I80  NEXT J
I90  LET F = -1
200  GOTO 220
2I0  LET F = +1
220  GCP F, 230
225  GOTO 80
230  GOSUB 245
240  STOP
245  SUB
250  FOR I = 1 TO D
260  LET Y(T) = X(I+Z-1)
270  NEXT I
280  EXCHANGE TC(Y,X)
285  RETURN
290  END
```


Опишем новые операторы (табл.3) и функции, а также модификации некоторых ранее рассмотренных операторов.

Таблица 3

Операторы ОВС-языка

Название оператора	Форма задания оператора
LET	LET ⟨список переменных⟩ = ⟨формула⟩
DIM	DIM ⟨список описаний⟩
DEF	DEFFN ⟨буква⟩(⟨ простая переменная ⟩) = ⟨формула⟩
REM	REM ⟨строка символов⟩
GOTO	GOTO ⟨номер строки⟩
IF-THEN	IF ⟨формула⟩ THEN ⟨ номер строки ⟩
CALL	CALL ⟨номер подпрограммы⟩ (⟨список параметров⟩)
FOR	FOR ⟨ простая переменная⟩ =⟨ формула⟩ TO ⟨формула⟩STEP⟨формула ⟩ *)
NEXT	NEXT ⟨простая переменная ⟩
GOSUB	GOSUB ⟨номер строки ⟩
RETURN	RETURN
END	END
STOP	STOP
INPUT	INPUT ⟨номер устройства⟩ , ⟨ тип ввода⟩ (⟨спи - сок ввода ⟩)
OUTPUT	OUTPUT ⟨номер устройства ⟩ , ⟨ тип вывода⟩ (⟨спи- сок вывода ⟩)
EXCHANGE	EXCHANGE ⟨тип обмена ⟩ (⟨список взаимодействующих массивов⟩)
GCP	GCP ⟨простая переменная⟩ , ⟨номер строки⟩
GUP	GUB ⟨номер строки ⟩
SUB	SUB

2.2. Оператор INPUT (оператор ввода) может использоваться не только для автономного ввода в каждой ветви,но

*) В случае, когда шаг цикла = I, STEP может быть опущен .

и для ввода через выделенную ветвь*) с последующей рассылкой вводимых данных. При этом имеются следующие возможности [3]:

- ввод с полным дублированием (PD) информации, например, после выполнения оператора 20 значения N и E окажутся во всех p-ветвях;

- ввод с распределением информации полосами (RP) ,например, после выполнения оператора INPUT 2, RP, M(B) массив B окажется разрезанным по координате M;

- ввод с распределением информации полосами, но с дублированием смежных строк (RPD): INPUT 2, RPD, M(B) ;

- ввод с распределением информации циклическими полосами (CP): INPUT 2, CP, M, K(B) , где K - ширина полосы;

- ввод с распределением информации одновременно по двум координатам - распределение RS: INPUT 2, RS, M1, M2(B).

2.3. Функция "DEL". Системная стандартная функция 'DEL' (N) вычисляет величину, равную $\left[\frac{N}{L}\right]+1$ для ветвей с относительными номерами, не превышающими остаток от деления N на L, и $\left[\frac{N}{L}\right]$ - для остальных ветвей, N - простая переменная, L - число ветвей, $\left[\frac{N}{L}\right]$ - целая часть числа $\frac{N}{L}$.

Данную функцию удобно применять, когда верхняя граница координаты, по которой распределяется массив, не кратна числу ветвей.

2.4. Оператор CALL вызывает библиотечную подпрограмму, в нашем примере - с номером 100, которая для каждой ветви k вычисляет $Z_k = 1 + \sum_{i=1}^{k-1} U_i$, где $U_i = 'DEL'(N)$ вычислена в i-ой ветви, Z_k есть номер первой компоненты части вектора X, вычисляемой в k-ой ветви:

$$\underbrace{X_{Z_1} \cdots X_{Z_1+U_1-1}}_{\text{I-я ветвь}} \quad \underbrace{X_{Z_2} \cdots X_{Z_2+U_2-1}}_{\text{2-я ветвь}} \cdots \underbrace{X_{Z_L+U_L-1}}_{L\text{-я ветвь}}$$

2.5. Оператор EXCHANGE (оператор обмена) позволяет обмениваться информацией между ветвями по одной из следующих схем.

*) Выделенная ветвь, определяемая в паспорте задачи, задает номер машины, с внешнего устройства которой будет осуществляться ввод.

Трансляционная схема обмена (TO) - передача данных из выделенной ветви во все остальные, например:

EXCHANGE TO,X(Y,Z),

где X-номер выделенной (передающей) ветви, Y, Z - идентификаторы соответственно передаваемого и принимающего массивов.

Трансляционно-циклическая схема обмена (TC) - последовательное выполнение трансляционной схемы всеми ветвями, например:

EXCHANGE TC(Y,Z).

После выполнения такого обмена в каждой ветви сформируется массив $\mathcal{Z}$ следующего вида:

Y из ветви I
Y из ветви 2
. . .
Y из ветви L

Коллекторная схема обмена (KO) - передача данных в выделенную p-ветвь из всех остальных, например:

EXCHANGE KO, X(Y,Z)

где X - номер выделенной(принимающей) ветви, Y и Z - взаимодействующие массивы.

Обмен сдвигом (парный обмен - PO) - передача данных из каждой ветви в соседнюю, например:

EXCHANGE PO, L(Y,Z)

или

EXCHANGE PO, R(Y,Z),

где L означает сдвиг данных влево (по убыванию номеров ветвей), R - вправо (по возрастанию номеров ветвей). При этом одна из крайних ветвей (с максимальным или минимальным номерами) только передает информацию, а другая - только принимает.

Дифференцированный обмен (DO) - передача данных из выделенной ветви в одну или несколько выделенных ветвей, например:

EXCHANGE DO, X, X1, X2, X3 (Y,Z),

где X - номер передающей ветви, X1, X2 и X3 - номера принимающих ветвей.

2.6. Оператор GCP (оператор обобщенного условного перехода) вызывает изменение естественного порядка выполнения операторов во всех ветвях при выполнении обобщенного условия, например: GCP F, 230

передает управление оператору 230, если F - отрицательная вели - чина во всех ветвях, и следующему оператору, если F - положитель- на хотя бы в одной ветви.

2.7. О п е р а т о р GUP (оператор обобщенного безусловно- го перехода) вызывает изменение естественного выполнения опера - торов во всех ветвях при появлении его хотя бы в одной из них, например:

GUP 100.

В отличие от GCP, для выполнения которого необходим выход на него всех ветвей, GUP осуществляется при выходе на него од- ной ветви.

2.8. О п е р а т о р OUTPUT (оператор вывода) может ис - пользоваться как для автономного вывода в каждой ветви, так и для вывода через выделенную ветвь с предварительной сборкой данных, например:

OUTPUT 5, RP, X(Y,Z)

вызовет сборку распределенных полосами по X-ой координате масси- вов Y и Z и печать их на АЦПУ машины, реализующей выделенную ветвь. Число типов вывода оператора OUTPUT совпадает с числом типов ввода оператора INPUT .

2.9. С и с т е м н ы е с т а н д а р т н ы е ф у н к - ц и и. В ОВС-языке, кроме уже рассмотренной функции 'DEL', имеют- ся еще пять системных стандартных функций.

'NUMI' - позволяет использовать в программе значения от - носительных номеров ветвей, например, оператор:

IF 'NUMI' = X THEN 100

передаст управление оператору 100 в той ветви, относительный но- мер которой равен X.

'NUM2' - позволяет использовать в программе значения об - щего числа ветвей*), например:

IF 'NUMI' = 'NUM2' THEN 200

передаст управление оператору 200 в ветви с максимальным номером.

'NUMI' и 'NUM2' - функции без аргумента.

*) Число ветвей задается в паспорте задачи.

SUM(X) - вычисляет сумму всех элементов массива X, распре- деленного по ветвям.

MIN (X) и MAX(X) - определяют значение минимального (макси- мального) элемента массива X, распределенного по ветвям, напри - мер:

IF MAX(X) 'GT' E THEN 300

передаст управление оператору 300, если значение максимального элемента массива X, распределенного по ветвям, больше E.

§ 3. Краткое описание синтаксиса и семантики ОВС-языка

3.1. ОВС - п р о г р а м м а.

Семантика. Необходимая и достаточная информация о вычисли - тельном процессе, реализуемом на ОВС.

Синтаксис.

⟨ ОВС - программа ⟩ ::= ⟨ОВС - оператор⟩ | ⟨ ОВС - програм - ма⟩ ⟨ ОВС - оператор ⟩

3.2. ОВС - о п е р а т о р.

Семантика. Информация о единице действия ОВС - языка и по - рядке единиц действия.

Синтаксис.

⟨ ОВС - оператор ⟩ ::= ⟨ номер строки ⟩ ⟨основной оператор⟩ возврат каретки

⟨ номер строки ⟩ ::= ⟨ целое ⟩

⟨ основной оператор ⟩ ::= ⟨ оператор **LET**⟩ | ⟨ оператор **DIM** ⟩ | ⟨ оператор **DEF**⟩ | ⟨ оператор **REM**⟩ | ⟨ оператор **GOTO** ⟩ | ⟨ оператор **IF-THEN** ⟩ | ⟨ оператор **CALL** ⟩ | ⟨ оператор **FOR** ⟩ | ⟨ оператор **NEXT**⟩ | ⟨оператор **GOSUB** ⟩ | ⟨оператор **RETURN**⟩ | ⟨ оператор **END** ⟩ | ⟨опера- тор **STOP** ⟩ | ⟨ оператор **INPUT**⟩ | ⟨ оператор **OUTPUT**⟩ | ⟨ оператор **EXCHANGE** ⟩ | ⟨ оператор **GCP**⟩ | ⟨ оператор **GUP**⟩ | ⟨ оператор **SUB** ⟩ .

3.2.1. О п е р а т о р п р и с в а и в а н и я **LET**.

Семантика. Выполняет определенные вычисления и результат присваивает списку переменных.

Синтаксис.

⟨ оператор **LET** ⟩ ::= ⟨ заголовок **LET** ⟩ = ⟨формула ⟩

⟨ заголовок **LET** ⟩ ::= **LET** ⟨переменная⟩ | ⟨ заголовок **LET** ⟩ ⟨ переменная ⟩

⟨ формула⟩ ::= ⟨конъюнкция ⟩ | ⟨ формула⟩ **OR** ⟨конъюнкция ⟩

⟨ конъюнкция ⟩ ::= ⟨первичное⟩ | ⟨ конъюнкция ⟩ **AND** ⟨ первичное ⟩

⟨ первичное ⟩ ::= ⟨ арифметическое выражение ⟩ | ⟨ первичное ⟩ ⟨ знак отношения ⟩ ⟨ арифметическое выражение ⟩

⟨ арифметическое выражение ⟩ ::= ⟨ терм ⟩ | ⟨ арифметическое выражение ⟩ + ⟨ терм ⟩ | ⟨ арифметическое выражение ⟩ - ⟨ терм ⟩

⟨ терм ⟩ ::= ⟨ множитель ⟩ | ⟨ терм ⟩ * ⟨ множитель ⟩ | ⟨ терм ⟩ / ⟨ множитель ⟩

⟨ множитель ⟩ ::= ⟨ первичное выражение ⟩ | ⟨ знак ⟩ ⟨ первичное выражение ⟩ | **NOT** ⟨ первичное выражение ⟩

⟨ первичное выражение ⟩ ::= ⟨ операнд ⟩ | ⟨ первичное выражение ⟩ ↑ ⟨ операнд ⟩

⟨ знак отношения ⟩ ::= > | < | ≤ | ≥ | = | ≠

⟨ операнд ⟩ ::= ⟨ переменная ⟩ | ⟨ число без знака ⟩ | ⟨ стандартная функция ⟩ | ⟨ функция ⟩ | ⟨ операндная формула ⟩

⟨ переменная ⟩ ::= ⟨ простая переменная ⟩ | ⟨ переменная с индексами ⟩

⟨ простая переменная ⟩ ::= ⟨ буква ⟩ | ⟨ буква ⟩ ⟨ цифра ⟩

⟨ переменная с индексами ⟩ ::= ⟨ идентификатор массива ⟩ (⟨ список индексов ⟩)

⟨ список индексов ⟩ ::= ⟨ индекс ⟩ | ⟨ список индексов ⟩ , ⟨ индекс ⟩

⟨ индекс ⟩ ::= ⟨ формула ⟩

⟨ идентификатор массива ⟩ ::= ⟨ 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**

⟨ цифра ⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨ знак ⟩ ::= + | -

⟨ число без знака ⟩ ::= ⟨ десятичная часть ⟩ | ⟨ десятичная часть ⟩ ⟨ порядок ⟩

⟨ десятичная часть ⟩ ::= ⟨ целое ⟩ | ⟨ целое ⟩ . | ⟨ целое ⟩ . ⟨ целое ⟩ | . ⟨ целое ⟩

⟨ целое ⟩ ::= ⟨ цифра ⟩ | ⟨ целое ⟩ ⟨ цифра ⟩

⟨ порядок ⟩ ::= **E** ⟨ целое ⟩ | **E** ⟨ знак ⟩ ⟨ целое ⟩

⟨ стандартная функция ⟩ ::= ⟨ название стандартной функции ⟩ | ⟨ название стандартной функции ⟩ ⟨ аргумент ⟩

⟨ название стандартной функции ⟩ ::= **'SIN'** | **'COS'** | **'TNG'** | **'ATN'** | **'EXP'** | **'ABS'** | **'LOG'** | **'SQR'** | **'INT'** | **'SGN'** | **'MAX'** | **'MIN'** | **'SUM'** | **'DEL'** | **'NUM1'** | **'NUM2'**

⟨ аргумент ⟩ ::= (⟨ фактический параметр ⟩)

⟨ фактический параметр ⟩ ::= ⟨ формула ⟩

⟨ функция ⟩ ::= **FN** ⟨ буква ⟩ ⟨ аргумент ⟩

⟨ операндная формула ⟩ ::= (⟨ формула ⟩)

3.2.2. Оператор описания DIM.

Семантика. Описывает идентификаторы переменных, резервирует место в оперативной памяти.

Синтаксис.

⟨ оператор **DIM** ⟩ ::= **DIM** ⟨ список описаний ⟩

⟨ список описаний ⟩ ::= ⟨ описание ⟩ | ⟨ список описаний ⟩ , ⟨ описание ⟩

⟨ описание ⟩ ::= ⟨ идентификатор массива ⟩ (⟨ список верхних границ ⟩) | ⟨ идентификатор простой переменной ⟩

⟨ список верхних границ ⟩ ::= ⟨ верхняя граница ⟩ | ⟨ список верхних границ ⟩ , ⟨ верхняя граница ⟩

⟨ верхняя граница ⟩ ::= ⟨ целое ⟩ | ⟨ простая переменная ⟩

3.2.3 Оператор описания функции пользователя DEF .

Семантика. Описывает функцию, которая может использоваться в программе неоднократно. Название функции состоит из трех букв, первые из которых **FN**. Максимальное число функций, задаваемых в программе, - 26 : **FNA**, **FNB**, ..., **FNY**, **FNZ**.

Синтаксис.

⟨ оператор **DEF** ⟩ ::= **DEFFN** ⟨ буква ⟩ (⟨ простая переменная ⟩) = ⟨ формула ⟩

3.2.4. Оператор комментарий REM .

Семантика. Служит для вставки в программу пояснительного текста

Синтаксис.

⟨ оператор **REM** ⟩ ::= **REM** ⟨ строка символа ⟩

⟨ строка символов ⟩ ::= ⟨ символ ⟩ | ⟨ строка символов ⟩ ⟨ символ ⟩

⟨ символ ⟩ ::= любой символ телетайпа, за исключением "возврата каретки", "перевода строки" и "?"

3.2.5. Оператор безусловного перехода GOTO.

Семантика. Изменяет естественный порядок выполнения программы. При переходе на пассивные операторы (**REM** , **DEF**) управление

передается ближайшему по порядку исполнительному оператору. Переходы в тело цикла и в тело подпрограммы запрещены.

Синтаксис.

⟨ оператор GOTO ⟩ ::= GOTO ⟨ номер строки ⟩

3.2.6. Оператор условного перехода IF-THEN.

Семантика. Изменяет естественный порядок выполнения операторов при выполнении некоторого условия. Имеет ограничения предыдущего оператора.

Синтаксис.

⟨ оператор IF-THEN ⟩::= IF ⟨формула⟩ THEN ⟨ номер строки ⟩

3.2.7. Оператор начала цикла FOR.

Семантика. Служит для организации циклов в программе.

Синтаксис.

⟨ оператор FOR ⟩::= FOR ⟨ простая переменная ⟩ = ⟨формула ⟩ TO ⟨формула⟩STEP ⟨ формула ⟩ | FOR ⟨простая переменная ⟩ = ⟨формула ⟩ TO ⟨ формула⟩

3.2.8. Оператор конца цикла NEXT.

Семантика. Определяет конец цикла.

Синтаксис.

⟨ оператор NEXT⟩ ::= NEXT ⟨ простая переменная ⟩

3.2.9. Оператор перехода на подпрограмму GOSUB.

Семантика. Служит для перехода на подпрограмму, являющуюся частью данной программы. Вход в подпрограмму - только через оператор SUB.

Синтаксис.

⟨оператор GOSUB⟩ ::= GOSUB ⟨ номер строки⟩

3.2.10. Оператор выхода из подпрограммы RETURN.

Семантика. Определяет конец подпрограммы. Выход из подпрограммы - только через оператор RETURN .

Синтаксис.

⟨ оператор RETURN⟩ ::= RETURN

3.2.11. Оператор начала подпрограммы SUB.

Семантика. Определяет начало подпрограммы.

Синтаксис. ⟨оператор SUB ⟩ ::= SUB

3.2.12. Оператор конца программы END.

Семантика. Определяет конец программы.

Синтаксис.

⟨ оператор END⟩ ::= END

3.2.13. Оператор останова STOP.

Семантика. Эквивалентен оператору GOTO $x_1\ x_2\ x_3\ x_4$, где $x_1\ x_2\ x_3\ x_4$ номер строки оператора END . Удобен в программах , имеющих более одной конечной точки.

Синтаксис.

⟨ оператор STOP ⟩ ::= STOP

3.2.14. Оператор обобщенного безусловного перехода GUP.

Семантика. Вызывает изменения естественного порядка выполнения операторов во всех параллельных ветвях.

Синтаксис.

⟨ оператор GUP ⟩ ::= GUP ⟨ номер строки⟩

3.2.15. Оператор обобщенного условного перехода GCP.

Семантика. Вызывает изменения естественного порядка выполнения операторов во всех параллельных ветвях при выполнении обобщенного условия (если значения простых переменных из оператора GCP во всех ветвях отрицательны).

Синтаксис.

⟨ оператор GCP⟩ ::= GCP ⟨простая переменная⟩ , ⟨номер строки⟩

3.2.16. Оператор вызова библиотечных подпрограмм CALL.

Семантика. Служит для включения в СВС - программу библиотечных подпрограмм, записанных в кодах системы.

Синтаксис.

⟨ оператор CALL⟩ ::= CALL ⟨номер подпрограммы⟩ (⟨список параметров ⟩)

⟨список параметров⟩ ::= ⟨элемент списка параметров⟩ | ⟨ список параметров⟩ , ⟨ элемент списка параметров ⟩

⟨элемент списка параметров⟩ ::=⟨ число без знака⟩ | ⟨ простая переменная ⟩ | ⟨ идентификатор массива ⟩

⟨ номер подпрограммы⟩ ::= ⟨целое ⟩

3.2.I7. О п е р а т о р о б м е н а **EXCHANGE**.

Семантика. Реализует обмены массивами между ветвями ОВС-программы.

Синтаксис.

⟨ оператор **EXCHANGE** ⟩ ::= **EXCHANGE** ⟨тип обмена⟩ (⟨список взаимодействующих массивов ⟩)

⟨ тип обмена ⟩ ::= ТО, ⟨ передающая ветвь ⟩| ТС| КО, ⟨ принимающая ветвь⟩| **PO,R|PO,L| DO** , ⟨ передающая ветвь ⟩, ⟨ выделенные ветви⟩

⟨ передающая ветвь ⟩ ::=⟨ простая переменная ⟩

⟨ принимающая ветвь ⟩ ::=⟨ простая переменная ⟩

⟨ выделенные ветви ⟩ ::= ⟨простая переменная ⟩| ⟨ выделенные ветви ⟩, ⟨ простая переменная ⟩

⟨ список взаимодействующих массивов⟩ ::= ⟨идентификатор взаимодействующего массива ⟩, ⟨ идентификатор взаимодействующего массива ⟩

⟨ идентификатор взаимодействующего массива ⟩ ::=⟨ идентификатор массива⟩| ⟨ идентификатор массива ⟩ (⟨список значений индексов ⟩) ⟨простая переменная ⟩

⟨ список значений индексов ⟩::=⟨ индекс⟩| ⟨ список значений индексов ⟩,⟨ индекс ⟩

⟨ индекс⟩ ::= ⟨целое ⟩| ⟨ простая переменная ⟩|*

3.2.I8. О п е р а т о р в в о д а **INPUT**.

Семантика. Вводит данные в систему либо через выделенную ветвь Р-программы, либо через все ветви параллельно.

Синтаксис.

⟨ оператор **INPUT** ⟩::= **INPUT** ⟨номер устройства ⟩, ⟨ тип ввода/вывода ⟩ (⟨ список ввода ⟩)

⟨ номер устройства ⟩ ::= ⟨целое ⟩

⟨ тип ввода/вывода ⟩ ::= **A|PD|RP,** ⟨ простая переменная ⟩|**RPD** , ⟨ простая переменная⟩ | CP, ⟨ простая переменная⟩ , ⟨ простая переменная ⟩| **RS** , ⟨ простая переменная ⟩, ⟨ простая переменная ⟩

⟨ список ввода⟩ ::= ⟨простая переменная⟩ | ⟨ идентификатор массива ⟩| ⟨ список ввода⟩ , ⟨идентификатор массива⟩| ⟨список ввода⟩ , ⟨простая переменная⟩

3.2.I9. О п е р а т о р в ы в о д а **OUTPUT**.

Семантика. Выводит данные либо через выделенную ветвь, либо через все ветви параллельно.

Синтаксис.

⟨ оператор **OUTPUT** ⟩ ::= **OUTPUT** ⟨ номер устройства⟩ , ⟨ тип ввода/вывода⟩ (⟨список вывода ⟩)

⟨ список вывода ⟩ ::=⟨ элемент списка вывода ⟩| ⟨ список вывода⟩ , ⟨ элемент списка вывода ⟩

⟨ элемент списка вывода ⟩::= ⟨ идентификатор массива ⟩|⟨ простая переменная ⟩| ⟨ спецификация ⟩

⟨ спецификация ⟩ ::= '' ⟨ текст⟩ ''

⟨ текст⟩::= последовательность символов телетайпа, за исключением "возврата каретки", "перевода строки", "кавычек".

Примечание:

⟨ номер строки⟩ - это непустое целое, не превышающее 7999. Пробелы между цифрами допускаются.

3.3. С т а н д а р т н ы е с и с т е м н ы е ф у н к ц и и.

3.3.I. Ф у н к ц и я **MAX** определяет значение максимального элемента массива, распределенного по р-ветвям. Аргументом этой функции является идентификатор распределенного массива.

3.3.2. Ф у н к ц и я **MIN**, подобно предыдущей, определяет значение минимального элемента массива.

3.3.3. Ф у н к ц и я **SUM** определяет сумму всех значений массива, распределенного по ветвям. Аргумент функции - идентификатор распределенного массива.

3.3.4. Ф у н к ц и я **DEL** вычисляет величину, равную $\left[\frac{N}{L}\right]+1$ для ветвей с относительными номерами не больше остатка от деления N на L, и $\left[\frac{N}{L}\right]$ для остальных ветвей, N - значение распределяемой величины, L - число ветвей, $\left[\frac{N}{L}\right]$ - целая часть числа $\frac{N}{L}$. Аргумент функции (распределяемая величина) - простая переменная.

3.3.5. Ф у н к ц и я **NUM1** вычисляет относительный номер ветви. Функция без аргумента.

3.3.6. Ф у н к ц и я **NUM2** вычисляет общее число ветвей. Функция без аргумента.

Транслятор с подмножества языка параллельных алгоритмов реализован для системы "Минск-222" [5,6]. Общий объем трансля -

тора – 6600 ячеек (вместе с библиотекой транслятора и сервисными программами), скорость трансляции 700–800 команд готовой программы в минуту. Транслятор допускает ввод и исправление исходных программ в режиме диалога.

Для ОВС МИНИМАКС [7] разрабатываются системные трансляторы с АССЕМБЛЕРа, ФОРТРАНа и АЛГОЛа. Предполагается реализовать системные операторы и функции как модули, к которым посредством операционной системы можно обращаться из программ, полученных с помощью любого из трансляторов (с АССЕМБЛЕРа, ФОРТРАНа и АЛГОЛа).

Литература

1. КОСАРЕВ Ю.Г. Распараллеливание по циклам. –"Вычислительные системы", Новосибирск, 1967, вып. 24, с. 3–15.

2. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука", 1966.

3. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на однородных вычислительных системах. Настоящий сборник, с.3–32.

4. **A Pocket Guide to Hewlett-Packard Computers. Palo Alto, California, 1969.**

5. ЕВРЕИНОВ Э.В., ЛОПАТО Г.П. Универсальная вычислительная система "Минск-222".–"Вычислительные системы", Новосибирск, 1966, вып. 23, с. 13–20.

6. ГРИШАЕВА Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И., КОНСТАНТИНОВ В.И., КОРНЕЕВ В.Д., ЛЕВАГИНА Т.А., МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Транслятор с языка параллельных алгоритмов. Настоящий сборник, с.55–73.

7. НИНОКУРОВ В.Г., ДИМИТРИЕВ Ю.К., ЕВРЕИНОВ Э.В., КОСТЕЛЯНСКИЙ В.М., ЛЕХНОВА Г.М., МИРЕНКОВ Н.Н., РЕЗАНОВ В.В., ХОРОШЕВСКИЙ В.Г. Однородная вычислительная система из мини-машин.–"Вычислительные системы", Новосибирск, 1972, вып. 51, с. 127–145.

Поступила в ред.-изд.отд.
13 декабря 1972 года