МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1979 год Выпуск 78

УДК 681.322-192

ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ ОРГАНИЗАЦИИ ОТКАЗОУСТОЙЧИВЫХ ВЫЧИСЛЕНИЙ В СИСТЕМЕ МИНИМАКС

В.Д.Корнеев

В в е д е н и е

Во многих применениях ЭВМ программа считается надежной, если она не только корректно выполняет (в большинстве случаев) определенную целевую задачу, но и "соответствующим" образом реагирует на ошибки как в самой себе, так и в аппаратурном и программном обеспечении. Для организации отказоустойчивых вычислений, т.е. вычислений, которые могут быть возобновлены автоматически либо достаточно быстро с помощью оператора, обычно используют внесение аппаратурной или программной избыточности.

Особую актуальность организация таких вычислений приобретает в многопроцессорных системах. Из работ, посвященных аппаратурной поддержке таких вычислений, отметим [1-4]. В них рассматриваются различные методы обнаружения неисправностей: контрольные коды [1-2], дублирование и сравнение [2], детекторы расхождения с мажоритарными схемами голосования [3], специальные схемы контроля источников питания, блоков памяти и т.д. [2], логические схемы самопроверки [4]. Наиболее полно аппаратно-управляемое восстановление нашло воплощение в экспериментальной ЭВМ IPL-STAR [1]. В ее состав включен специальный процессорный модуль контроля и ремонта, предназначенный для управления восстановлением и самовосстановлением. Программное обеспечение используется только для копирования содержимого памяти и возобновления нормальной работы после самовосстановления.

Из работ, посвященных программной поддержке отказоустойчивых вычислений, можно отметить [5-8]. Программные методы для

обнаружения неисправностей используют либо параллельное исполнение нескольких программ, либо введение в программу дополнительных функций. В первом случае работают несколько одинаковых программ, пользующихся отдельными процессорами и/или отдельными запоминающими устройствами; сравнение осуществляется с помощью обмена результатами или контрольными суммами [5,6]. Во втором случае дополнительные функции включают в себя: контрольное суммирование, средства задания повторных вычислений требуемых сегментов и сравнение результатов повторных вычислений [5] и т.д. Среди практически реализованных систем, отвечающих повышенным требованиям по надежности, обеспечиваемой в основном программными средствами, можно отметить [7,8]. В этих системах готовое программное обору - дование почти не имеет средств обнаружения неисправностей и полностью отсутствует аппаратура для выполнения восстановления. Особенностью этих систем является требование сохранения работоспособности программного обеспечения восстановления при наличии неисправностей, поскольку иначе нельзя начать восстановление. В работе [5] рассмотрены вопросы специальной предварительной подго - товки параллельных (р-) программ и выбора оптимальных временных шагов, через которые необходимо делать контроль вычислений.

Данная работа посвящена программной организации устойчивых вычислений в многопроцессорной системе МИНИМАКС [9]. Основное внимание уделяется I) общим схемам организации устойчивых вычислений; 2) методам и средствам динамического выявления ошибок, связанных с использованием операторов, задающих взаимодействия в разных элементарных машинах системы; 3) языковым средствам, необхо - димым для возобновления вычислений при обнаружении аварийных ситуаций.

§I. Общие схемы организации устойчивых вычислений

Отказоустойчивость в системах рассматривается по отношению к физическим и субъективным неисправностям. Физическая неисправность - это непредусмотренное изменение значений одной или нескольких логических переменных в системе. Субъективные неисправности связаны с недостатками программного обеспечения системы, которые остались невыявленными до момента возникновения отказа, с ошибками в специальных пользовательских программах, с неисправным взаимодействием человека с машиной через пульт управления и др. [II].

В системе МИНИМАКС отказоустойчивость имеет место как для некоторых субъективных неисправностей (неверное задание системного взаимодействия), так и для физических неисправностей (сбоев аппаратуры, выхода из строя отдельных ЭМ).

Обнаружение неисправностей разделяется на два типа: оперативное, осуществляемое одновременно с работой системы, и автономное, осуществляемое при временном перерыве работы.

Программные методы оперативного обнаружения основаны на параллельном исполнении нескольких программ и/или сводятся к введению в программу дополнительных функций [5]. При параллельном исполнении несколько одинаковых программ, соответствующих одной ветви параллельной программы, пользуются отдельными процессорами и разной памятью; сравнение результатов осуществляется ими через определенные промежутки времени путем обмена данными. При выполнении каждой ветви р-программы на одной ЭМ для контроля вычислений вводятся специальные подпрограммы, обеспечивающие: сравнение содержимого заданных областей памяти, повторное выполнение определенных сегментов программы, обмен информацией между ветвями, перестройку р-программы на меньшее число ветвей [5], обнаружение семантических ошибок в программах и т.д.

Программные методы автономного обнаружения неисправностей предполагают использование стандартных тестовых и диагностических программ элементарной машины системы.

Для выполнения дополнительных функций в программе должны быть выбраны точки обращения к соответствующим подпрограммам. Эти точки делятся на два типа: контроля и возврата.

Точка контроля связывается с обменом информации между элементарными машинами и анализом определенных областей памяти.

Точка возврата связывается с запоминанием I) информации, необходимой для перезапуска программы, и 2) места, с которого этот перезапуск нужно осуществить.

Ниже приведены общие схемы применения точек контроля и возврата при создании параллельных программ системы МИНИМАКС. Эти схемы ориентированы на программные методы организации отказоустойчивости.

I. Выполнение р-программы с повторением счета. На рис.I представлена схема выполнения р-программы из L ветвей. Точки $B_1^1, B_2^1, \ldots, B_L^1, B_1^2, B_2^2, \ldots, B_L^2, B_1^3, B_2^3, \ldots, B_L^3$ являются точками возврата; $K_1^1, K_2^1, \ldots, K_L^1, K_1^2, K_2^2, \ldots,$

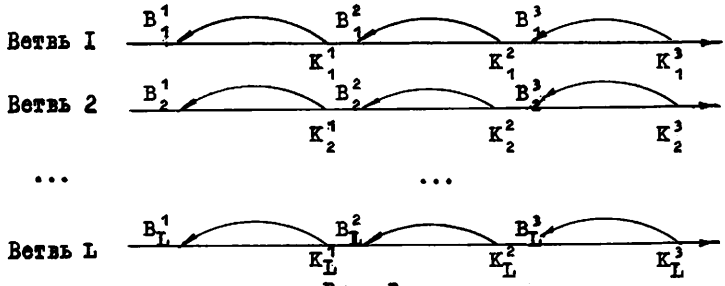


Рис. I

$K_L^2, K_1^3, K_2^3, \ldots, K_L^3$ – точками контроля. Метод состоит в повторном выполнении сегментов, заключенных между точкой возврата и точкой контроля, и сравнением результатов обоих выполнений.

Несовпадение результатов хотя бы в одной ветви ведет к возобновлению вычислений с начала сегмента (точки возврата) во всех ветвях.

2. Выполнение р-программы на нескольких подсистемах. На рис.2 представлена

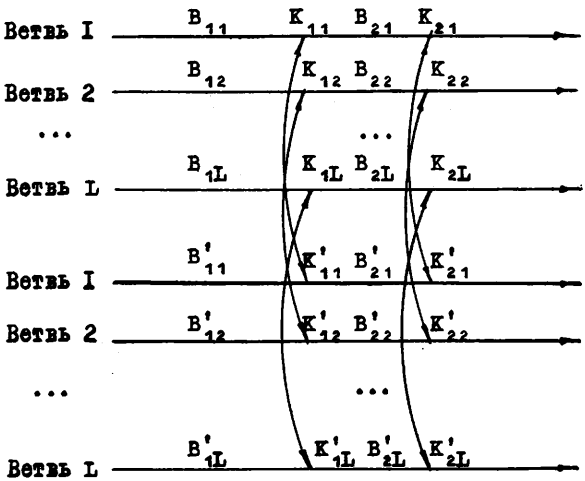


Рис. 2

схема выполнения р-программы из L-ветвей с дублированием счета на двух подсистемах. Точки $B_{11},B_{12},\ldots,B_{1L},B_{21},B_{22},\ldots,B_{2L}$, B'_{11}, $B'_{12},\ldots,B'_{1L},B'_{21},B'_{22},\ldots,B'_{2L}$ являются точками возврата; K_{11},K_{12}, $\ldots,K_{1L},K_{21},K_{22},\ldots,K_{2L},K'_{11},K'_{12},\ldots,K'_{1L},K'_{21},K'_{22},\ldots,K'_{2L}$ – точками контроля.

Метод состоит в сравнении результатов счета одинаковых ветвей р-программы из разных подсистем. Сравнение осуществляется в точках контроля. Несовпадение результатов хотя бы в одной ветви ведет к возобновлению вычислений с начала сегмента во всех ветвях обеих подсистем.

3. Выполнение р-программы с простой расстановкой точек возврата и контроля. На рис.3 представлена простая схема выполнения р-программы из L-ветвей, в которой $B^1_1,B^1_2,\ldots,B^1_L,B^2_1,B^2_2,\ldots,B^2_L$ являются точками возврата; $K^1_1,K^1_2,\ldots,K^1_L,K^2_1,K^2_2,\ldots,K^2_L$ – точками контроля. При данном методе осуществляется анализ ошибок, обнаруживаемых только аппаратурой. Разбор ситуаций производится в точках контроля. В случае сбоя аппаратуры в какой-либо ЭМ все ветви р-программы возобновляют вычисления с точек возврата. При отсутствии сбоев вычисления продолжаются без дополнительных действий.

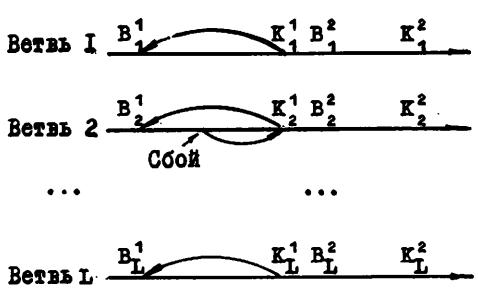


Рис. 3

4. Выполнение р-программы при сбоях на операторах групповых взаимодействий. Системные взаимодействия являются важной составной частью процесса вычислений, поэтому необходим контроль правильности непосредственно этих взаимодействий.

На рис.4 представлена схема выполнения р-программы из L-ветвей. Точки $B_1,B_2,\ldots,B_L$ являются точками возврата; $K_1,K_2,\ldots,K_L$ – точками контроля. Точки контроля находятся внутри оператора группового взаимодействия. В них осуществляется анализ ситуации при

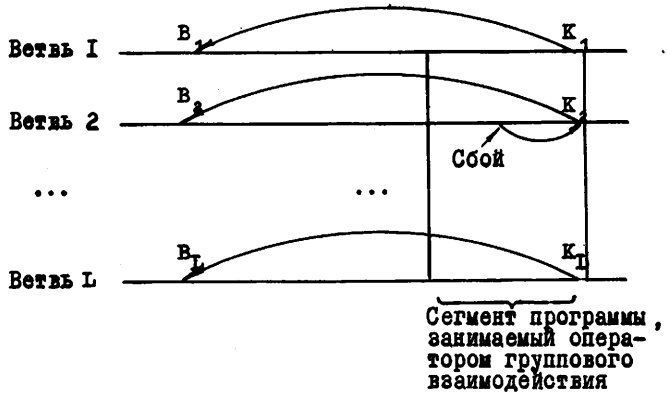


Рис. 4

системном взаимодействии. В случае сбоя во время системного взаимодействия все ветви возобновляют вычисления с точек возврата.

5. Выполнение р-программы при сбоях на операторах, индивидуальных взаимодействий. На рис.5 представлена схема взаимодействий старшего диспетчера с ветвью i р-программы; B_1, B_2 - точки возврата; K_1, K_1', K_2, K_2' - точки контроля, на которых осуществляется анализ ситуации при системном взаимодействии. При сбойной ситуации во время взаимодействия процессы старшего диспетчера переходят на точку возврата B_1, процессы ветви i на B_2.

На рис.6 представлена схема взаимодействий старшего диспетчера, ветви i и ветви j р-программы. Процессы старшего диспетчера, как и в предыдущей схеме, при сбойной ситуации переходят на точку возврата B_1. Процессы ветви i при сбое во время взаимодействия со старшим диспетчером либо во время взаимодействия с ветвью j переходят на точку возврата B_2. Процессы ветви j переходят на диспетчер ЭМ для восстановления и продолжения прерванного процесса.

Для реализации описанных схем в системе МИНИМАКС имеются специальные языковые средства: операторы системных взаимодействий и специальные операторы для задания точек возврата (см. [I4] и разд. 2, 3).

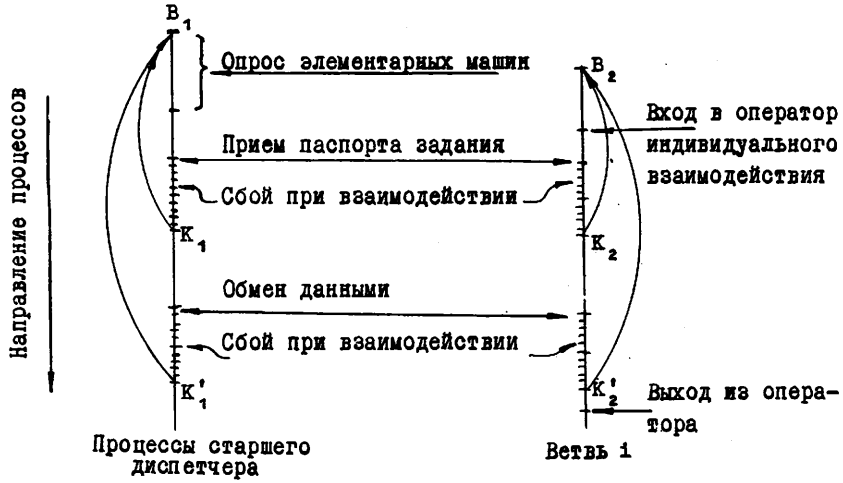


Рис. 5. Схема взаимодействий старшего диспетчера с ветвью i р-программы.

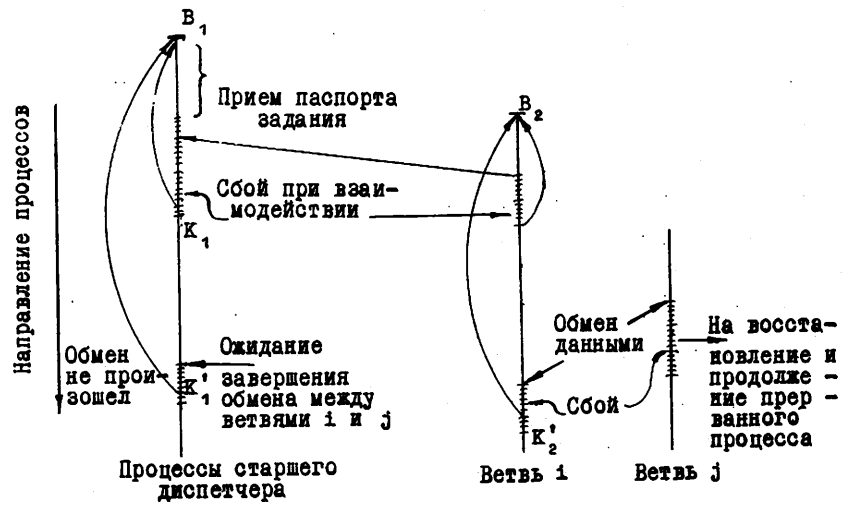


Рис.6. Схема взаимодействий старшего диспетчера, ветви i и ветви j р-программы.

§2. Обнаружение семантических ошибок в операторах системных взаимодействий

Средствами параллельного программирования в системе МИНИМАКС являются алгоритмические языки высокого уровня (АЛГОЛ,ФОРТРАН...), расширенные специальными системными операторами, реализующими индивидуальные и групповые схемы взаимодействий между ветвями параллельных процессов [I2,I3].

Первые реализуются с помощью старшего диспетчера и диспетчеров ЭМ [I4] и имеют централизованное управление аппаратом событий, каналами межмашинных связей. Вторые реализуются с помощью диспетчеров ЭМ и имеют децентрализованное управление каналами межмашинных связей; в основе "аппарата" событий лежат моменты выхода р-ветвей на разные копии одного и того же системного оператора.

Неправильное использование операторов системных взаимодействий может приводить к отказам всего многопроцессорного комплекса. Поэтому в операционной системе МИНИМАКС предусмотрены специальные средства контроля ошибок программирования.

Для обнаружения семантических ошибок в операторах индивидуальных взаимодействий устанавливается "правильность" слов в паспортах заданий для операционной системы. Например, для оператора IW (I,A1,A2,W,A,N,B,P) (ИНДИВИДУАЛЬНАЯ ЗАПИСЬ) в его паспорте задания выявляется наличие в множестве семафоров подмножества с именем I . Для остальных параметров должны выполняться соотношения: $0 \le W \le L$,где W - номер ветви р-программы, с которой требуется взаимодействие, L -число машин в системе; $0 \le \hat{A} \le D_i$, $0 \le \hat{B} \le D_j$,где $\hat{A}$ и $\hat{B}$ - прямые адреса массивов A и B соответственно в передающей и принимающей ЭМ, D_i и D_j - размеры памяти в этих ЭМ; $0 \le N \le \min[(D_i - A), (D_j - B)]$, где N - размер передаваемого (принимаемого) массива. Кроме того, номер ветви р-программы, с которой требуется взаимодействие, не должен совпадать с номером ветви, от которой получен паспорт задания. В операторе INP (D,I,P) (ЗАДАТЬ ПАСПОРТ СОБЫТИЙ), параметр D , задающий размер семафора в битах, должен удовлетворять неравенству: $0 \le D \le 15$. Правильность устанавливается специальным модулем старшего диспетчера [I4].

Старший диспетчер последовательно проверяет наличие запросов от ветвей р-программы. При наличии запроса от какой-либо ветви он принимает задания от этой ветви и проверяет условие синхронизации,

указанное в задании. При выполнении этого условия старший диспетчер реализует требуемое взаимодействие, при невыполнении задания - ставится в очередь. Может возникнуть ситуация,когда запросы от всех ветвей выполняемой р-программы будут стоять в очереди заданий и не будет ни одной ветви, которая могла бы изменить значение множества семафоров.

Такая ситуация называется глобальным тупиком. В реализованной версии операционной системы контролируется число ветвей р-программы, ожидающих нужных им событий, выявляется ситуации глобального тупика и выдается диагностическое сообщение о нем.

Для выявления семантических ошибок в операторах групповых взаимодействий также устанавливается "правильность" соответствующих параметров [I2], которые проверяются перед выполнением каждого оператора специальной подпрограммой, входящей в тело этого же оператора. Например, для операторов ВЕХС (N2,N1,N,A,B) (ТРАНСЛЯЦИОННЫЙ ОБМЕН) и DEXC (N2,N1,N,A,B,Z1,...,ZK) (ДИФФЕРЕНЦИРОВАННЫЙ ОБМЕН) проверяются:

- несовпадение имен операторов,выполняющих одно и то же групповое взаимодействие в разных машинах системы (параметр N2 во всех ветвях должен иметь одно и то же числовое значение);
- превышение номера заданной ветви р-программы числа всех возможных ветвей ($0 < N1 \le L$; $0 < Z1 \le L$, $0 < Z2 \le L$, ...,$0 < ZK \le L$) ;
- несовпадение заданных номеров ветвей и числа обмениваемых кодов, указанных в операторах разных процессов, вышедших на взаимодействие.

Кроме того, в этих операторах предусмотрены средства, непосредственно направленные на контроль определенных сбоев аппаратуры. Первый тип этих средств связан с трехкратной попыткой считывания слова состояния модуля межмашинной связи, если при считывании указанного слова происходит сбой аппаратуры. Второй тип связан с трехкратной попыткой выполнения команд ПРИЕМ и ПЕРЕДАЧА [I2],если при выполнении этих команд имеют место аварийные ситуации. В рассмотренных случаях вычисления могут быть продолжены, если неисправность аппаратуры не является постоянной поломкой.

Описанные средства обнаружения семантических ошибок в операторах системных взаимодействий одновременно могут служить и для выявления определенных сбоев аппаратуры.

§3. Языковые средства для задания отказоустойчивых вычислений

Согласно общим схемам организации отказоустойчивых вычислений, описанных в §I, необходимо задавать совокупность точек контроля и возврата. Для задания точек контроля в системе МИНИМАКС имеются операторы системных взаимодействий [I2,I3], частичный контроль выполняется ими непосредственно при реализации самих операторов (см. §2), а более полный контроль может быть запрограммирован с помощью этих операторов.

Для задания точек возврата в системе имеются специальные языковые средства, которые описываются в данном параграфе.

Оператор RPT1(R) . предназначен для задания пользователем точки возврата, на которую передается управление в случае сбойной ситуации во время взаимодействия между ветвями р-программы. При R = 1 операционная система запоминает адрес следующего за RPT1 оператора в определенной ячейке, куда и передается управление в случае указанной сбойной ситуации. Изменить заданный адрес может очередной RPT1 , а сбросить этот адрес - RPT1(R) при R = 0.

С помощью RPT1(R) удобно задавать автоматический перезапуск выполнения оператора системных взаимодействий или автоматическую передачу управления на а) пользовательскую подпрограмму анализа ситуации и продолжение вычислений в условиях несостоявшегося взаимодействия; б) специальный модуль операционной системы, выдающий диагностическое сообщение об аварийной ситуации. При выполнении оператора запоминается только адрес возврата. Ниже приведен при-

```
                    ...
                    X:= 1
                    GOTO F1
              F     X:= 0
              F1    CALL AVZZ(X,R)
                    ...
                    P:= 0
                    A:= 0
                    CALL RPT1(1)
        Точка
        возврата  → A:= A+1
                    IF A≥ 2 GOTO F
Оператор системного
взаимодействия    → CALL IW(I,A1,A2,W,A,N,B,P)
                    GOTO M
```

```
                CALL RPT1(0)
                ...
M               ...
```

мер использования оператора RPT1(R) . В программе задается переход на метку F , если дважды при попытке выполнить оператор IW обнаруживались сбойные ситуации. Метка F может быть как меткой пользовательской программы, так и меткой оператора вызова специальной подпрограммы AVZZ . Оператор CALL RPT1(0) указывает границу действия оператора CALL RPT1(1).

Оператор RPT2 (1,M,Z1,...,ZN) предназначен для задания пользователем точки возврата, на которую операционная система передает управление в случае сбойной ситуации во время выполнения сегмента, начинающегося с RPT2 и завершающегося точкой контроля. При выполнении оператора внешние переменные Z1,...,ZN записываются в массив M. Переменные называются "внешними" по отношению к заданной точке возврата, если они не восстанавливаются при повторном выполнении программы от точки возврата, иначе они называются внутренними.

После очередной точки возврата, задаваемой подобным оператором, записывается оператор RPT2 (0,M,Z1,...,ZN) , который внешним переменным Z1,...,ZN приписывает новые значения из массива M . Адрес возврата оператор не задает. Последовательности переменных в операторах RPT2 (1,M,Z1,...,ZN) и RPT2 (0,M,Z1,..., ZN) должны совпадать.

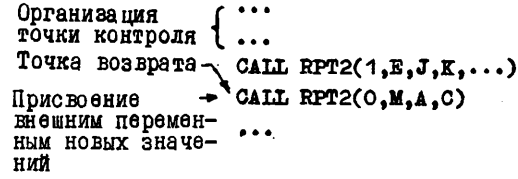
Оператор может иметь и один параметр: RPT2(1) . В этом случае задается только адрес возврата. Ниже приведен пример использования операторов RPT2 .

Фрагмент исходной программы

```
...
D:= 10
A:= A+B
C:= C*B-D
...
```

Фрагмент программы, подготовленной к отказоустойчивым вычислениям

```
                ...
                COM M(2),E(10)
                ...
                CALL RPT2(1,M,A,C)
Точка      →    D:= 10
возврата        M(1):= M(1)+B
                M(2):= M(2)*B-D
                ...
```



Если внутри заданного сегмента есть внешние переменные с индексом либо простые внешние переменные, не "поместившиеся" в операторе, то пользователь должен сам организовать запоминание таких переменных и программа будет иметь следующий вид:

```
                    ...
                    COM M(90), E(20)
                    ...
                    CALL RPT2(1)
Точка возврата  →   M(1):= A(1)
                    FOR I=1, 90 DO 10
                    M(I):= M(I)+B(I)
          10        NEXT
                    ...
                    ...
Точка возврата      CALL RPT2(1,E,J,K ... )
                    FOR I=1, 90 DO 5
Присвоение          A(I):= M(I)
внешним пере-
менным новых
значений      5     NEXT
                    ...
```

Оператор RPT3(R) задает точку возврата для возобновления вычислений при сбойной ситуации, не обнаруживаемой операционной системой и точками контроля. При R = 1 операционная система в специальном поле запоминает адрес следующего за RPT3 оператора, при R = 0 сбрасывается этот адрес. При неконтролируемой аварийной ситуации возобновление вычислений осуществляется вручную путем запуска процессора с адреса, хранящегося в упомянутом специальном поле.

Оператор SSS (сброс системных семафоров) параметров не имеет. Приводит определенные рабочие ячейки (системные семафоры) операционной системы в исходное состояние и запоминает адрес, с ко-

торого начинается его выполнение. Должен быть первым системным оператором в каждой ветви. В случае неконтролируемой аварийной ситуации вычисления можно возобновить вручную путем запуска процессора с адреса, задаваемого оператором SSS , т.е. с начала программы.

В заключение отметим, что описанные программные средства следует рассматривать как первый этап в программных методах организации отказоустойчивых вычислений. Однако и эти средства позволяют в ряде применений использовать качественно новые возможности параллельной обработки информации.

Литература

I. AVIZIENIS A. The STAR Computer: An investigation of the theory and practice of fault-tolerant computer design.- IEEE Trans. Computers,1971,v.C-20,N 11,p.1312-1321.

2. DOWNING R.W.,NOWAK J.S.,TUOMENOKSA L.S. No 1ESS maintenance plan.-Bull.Syst,Techn.J.,1964,v.43,N 5,pt 1,p.1961-2019.

3. ANDERSON J.E., MACRI F.J. Multiple redundancy applications in a computer.-In: Proc.1967 Ann.Symp. Reliability, Washington , 1967,Jan.,p.553-562.

4. CARTER W.C. Theory and use of chec ing circuits.- In: Computer Systems Reliability. Maidenhead, England: Infotech Informa - tion Ltd.,1974,p.413-454.

5. КОРНЕЕВ В.Д., МИРЕНКОВ Н.Н. Организация высоконадежного счета в однородных вычислительных системах.-Управляющие системы и машины, 1976, № 4, с. 46-54.

6. УЭНСЛИ Дж.Х., ЛЭМПОРТ Л. и др. SIFT : Проектирование и анализ отказоустойчивой вычислительной системы для управления полетом летательного аппарата. -Труды института инж. по электротехнике и радиоэлектронике, 1978, т. 66, № 10, с.3-11.

7. КАТСУКИ Д., ЭЙСАМ Э.С. и др. Pluribus - отказоустойчивый операционный мультипроцессор.-Там же, 1978, т.66, №10, с.12-18.

8. ИХАРА Х., ФУКУОКА К. и др. Отказоустойчивая вычислительная система с тремя симметричными вычислительными машинами. -Там же, 1978, т.66, №10, с.19-26.

9. ДИМИТРИЕВ Ю.К., ХОРОШЕВСКИЙ В.Г., МИРЕНКОВ Н.Н. и др. Однородная вычислительная система из мини-машин. -В кн.: Вычислительные системы, вып. 51, Новосибирск, 1974, с.143-153.

10. КОРНЕЕВ В.Д. Операционная система МИНИМАКС. -Настоящий сборник, с.3-30.

11. АВИЖЕНИС А. Отказоустойчивость - свойство,обеспечивающее постоянную работоспособность цифровых систем. - Труды института инж. по электронике и радиоэлектронике, 1978, т.66, №10,с.5-25.

12. КОЛОСОВА Ю.И. Языки для описания параллельных процессоров. Операторы групповых взаимодействий. Отчет ИМ СО АН СССР,1977.

I3. КОРНЕЕВ В.Д., КЕРБЕЛЬ В.Г., КРЫЛОВ Э.Г. Языки для описания параллельных процессоров. Операторы индивидуальных взаимодействий. Отчет ИМ СО АН СССР, I977.

I4. КОРНЕЕВ В.Д. Супервизор реального времени однородной вычислительной системы МИНИМАКС. Отчет ИМ СО АН СССР, I977.

Поступила в ред.-изд.отд.
I7 сентября I979 года