ВОПРОСЫ ТЕОРИИ И ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1977 год Выпуск 70

УДК 681.142.4

ЯДРО ОПЕРАЦИОННОЙ СИСТЕМЫ СУММА

С.Г.Седухин, М.П.Желтов, И.Н.Кашун

Под однородной вычислительной системой (ОВС) [1] понимается группа одинаковых ЭВМ, которые автоматически связываются между собой с помощью каналов межмашинных связей с целью повышения:

а) производительности вычислений;

б) эффективности использования оборудования;

в) надежности и живучести системы.

ЭВМ и ее канал межмашинной связи образуют элементарную машину (ЭМ) ОВС. Каждая ЭМ соединена непосредственно не со всеми ЭМ ОВС, а только с частью ЭМ, называемых соседними.

В настоящей работе рассматривается функционирование ОВС СУММА [2], построенной на базе мини-ЭВМ "Электроника-100И". Мини-ЭВМ расширяется до ЭМ ОВС подключением специализированного внешнего устройства, называемого в дальнейшем системным устройством, которое выполняет роль канала межмашинной связи [3]. Каналы межмашинной связи, являясь распределенным ресурсом системы, требуют определенной дисциплины управления, реализуемой в операционной системе.

Под операционной системой понимается некоторый комплекс, состоящий из людей, оборудования и программ, который направлен на наилучшее использование ресурсов системы [4]. В общем случае операционную систему можно представить как систему асинхронных взаимодействующих процессов. Ядром операционной системы называется минимально необходимый программно-аппаратный комплекс, реализующий асинхронные процессы и осуществляющий динамическое управление созданием, уничтожением и взаимодействием асинхронных процессов [5].

При создании ядра операционной системы ОВС СУММА учитывались требования живучести, наращиваемости и простоты (однородности)

всей системы в целом. Эти требования привели к децентрализации управления ОВС и распределению ядра операционной системы по всем ЭМ. Полное ядро операционной системы СУММА представляет собой набор одинаковых супервизоров элементарных машин.

I. Система параллельных процессов

П р о ц е с с о м называется последовательная реализация некоторого алгоритма*) в реальном времени. Процесс реализуется некоторым ресурсом системы (оператором, процессором, каналом и т.п.), существует наряду с другими процессами и при реализации взаимодействует с ними. В з а и м о д е й с т в и е есть обмен данными и управляющей информацией между процессами.

В системе процесс представлен информацией, которая должна быть загружена на определенный ресурс для того, чтобы продолжить выполнение команд процедуры процесса. Эта информация представляет собой в е к т о р с о с т о я н и я процесса. Физически вектор состояния процесса определяется в каждый момент времени состоянием регистров ЭМ.

В системе различают асинхронные процессы, реализуемые аппаратурой, программами и операторами. Программный процесс является последовательным исполнением запрограммированных на машинном языке действий аппаратуры процессора ЭМ. С каждым внешним устройством (диски, ленты, терминалы и т.д.) связан соответствующий программный процесс, так называемый драйвер внешнего устройства, осуществляющий взаимодействие соответствующего внешнего устройства с ЭВМ. Взаимодействие операторов с системой осуществляется через терминальные устройства.

По отношению к каждому супервизору различают внутренние и внешние процессы. Для в н у т р е н н е г о процесса в супервизоре определен вектор состояния данного процесса, для в н е ш н е г о такой вектор отсутствует.

Внутренний процесс может находиться либо в потенциальном, либо в реальном состояниях. П о т е н ц и а л ь н о е состояние характеризуется тем, что процесс не включен в данный момент в операционную обстановку, т.е. для супервизора не определен вектор состояния данного процесса. В момент включения потенциального процесса (определение вектора состояния) в операционную обстановку осуществляется переход процесса из потенциального состояния в р е а л ь н о е. Следовательно, для перевода процесса в реальное состояние должен существовать пусковой процесс.

*) Алгоритм понимается в смысле [9].

В свою очередь, множество реальных процессов может находиться в действующем и ждущем состояниях. Считается, что реальный процесс находится в д е й с т в у ю щ е м состоянии, если ему логически разрешено выполняться. Отметим также, что в системе параллельных асинхронных процессов логический смысл имеет не скорость выполнения отдельного процесса, а моменты достижения процессом существенных для системы процессов фаз - с о б ы т и й. Реальный процесс может ожидать предельного события в системе, переходя при этом в ж д у щ е е состояние.

Множество действующих процессов находится по отношению к супервизору в текущем состоянии или в прерванном состоянии. Процесс находится в т е к у щ е м состоянии, если он в данный момент выполняется на определенном ресурсе системы. Текущий процесс может быть прерван системой для освобождения ресурса, на котором этот процесс выполняется, и переведен в п р е р в а н н о е состояние. Иерархия состояний процесса показана на рис.I.

Рис. I

Супервизор ЭМ осуществляет динамические переходы управляемых процессов из одного состояния в другое. Для перевода программных процессов из потенциального состояния в реальное пусковой процесс использует процедуру супервизора *START*. В свою очередь, программный процесс, находящийся в реальном состоянии, может перейти в потенциальное, используя процедуру супервизора *HALT*. При этом супервизор удаляет вектор состояния данного процесса из области своего действия. Таким образом, наличие в супервизоре процедур *START* и *HALT* позволяет создавать и уничтожать программные процессы.

Для синхронизации программных процессов используются процедуры супервизора *POST* и *WAIT*. В этом случае каждому событию сопоставляется таблица управления событием (рис.2).

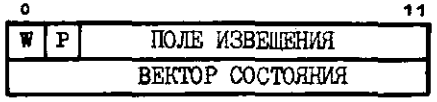

Рис. 2

Программный процесс может ожидать определенного события в системе, указывая соответствующую таблицу управления событием при использовании процедурн *WAIT*. При этом вектор состояния данного процесса переводится супервизором в о ч е р е д ь ж д у щ и х процессов. В этом случае говорят, что программный процесс находится в ждущем состоянии.

Действующий программный процесс, завершив существенное для системы процессов событие, переводит процесс, ждущий этого события, в действующее состояние с помощью процедуры супервизора *POST*, которая использует соответствующую таблицу управления событием. Таблица управления событием обеспечивает связь между процессом, ждущим определенного события, и процессом, извещающим о завершении данного события. При начальном определении таблицы управления событием и н д и к а т о р о ж и д а н и я (*W*) и и н д и к а т о р и з в е щ е н и я (*P*) должны быть сброшены.

Когда реализуется процедура *WAIT* индикатор ожидания соответствующей таблицы управления событием "взводится в единицу". Если индикатор извещения в данной таблице управления событием взведен (это говорит о том, что событие было совершенно ранее), то процесс, выдавший *WAIT*, продолжается дальше, иначе в таблице управления событием запоминается вектор состояния данного процесса и текущий процесс переводится в состояние ожидания необходимого события.

Когда реализуется процедура *POST*, индикатор извещения указанной таблицы управления событием "взводится в единицу", а в поле извещения заносится заданный к о д и з в е щ е н и я. Этот код дает информацию о том, как было завершено событие.

Поскольку событие может произойти как до выполнения соответствующей процедуры *WAIT*, так и после нее, то в процедуре проверяется состояние индикатора ожидания. Если индикатор установлен в единицу, то процедура переводит вектор состояния ждущего процесса в очередь прерванных, тем самым активизируя его. В любом случае извещающий процесс продолжается дальше.

Программный процесс находится в т е к у щ е м состоянии, если его вектор состояния содержится на регистрах процессора ЭМ. Все остальные действующие процессы образуют очередь на реализацию к

соответствующему процессору ЭМ, которая называется о ч е р е д ь ю п р е р в а н н ы х процессов. Очередь хранит векторы состояний программных процессов, находящихся в прерванном состоянии, и имеет структуру стека. В частности, текущий программный процесс может быть прерван аппаратным процессом через систему прерываний ЭМ и переведен в состояние прерванный.

Прерванный программный процесс активизируется (или переводится в текущее состояние) супервизором на основе приоритетного выбора соответствующего вектора состояния из очереди прерванных.

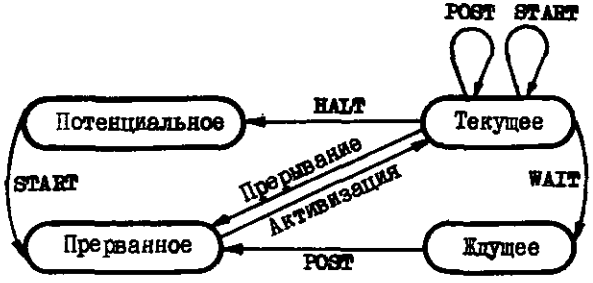

Рис. 3

Потенциальное состояние отображено в супервизоре максимально допустимой глубиной мультипрограммирования. Диаграмма переходов состояния программного процесса показана на рис.3.

Информационное взаимодействие программных процессов осуществляется определенными в супервизоре операциями *READ* (чтение) и *WRITE* (запись), с помощью которых процессы обмениваются необходимой информацией.

Так как взаимодействующие процессы могут реализоваться различными ЭМ системы, к функциям супервизора ЭМ добавляется организация взаимодействий процессов через канал межмашинных связей. Ориентируясь на однородность управления, такие операции, как *READ*, *WRITE*, *POST* и *START*, распространяются на взаимодействие программных процессов различных супервизоров ядра операционной системы.

При организации связи взаимодействующих процессов через канал межмашинных связей, в свою очередь, возникает задача выбора

режима управления каналом межмашинных связей.При выборе дисциплины управления каналом межмашинных связей анализировались два полярных режима управления: децентрализованное и централизованное [6]. Анализ этих дисциплин управления показал, что невозможно отдать предпочтение чисто централизованному или чисто децентрализованному управлениям каналом межмашинных связей. В ядре операционной системы СУММА используется смешанный режим управления каналом межмашинных связей, который рассматривается при описании структуры супервизора ЭМ.

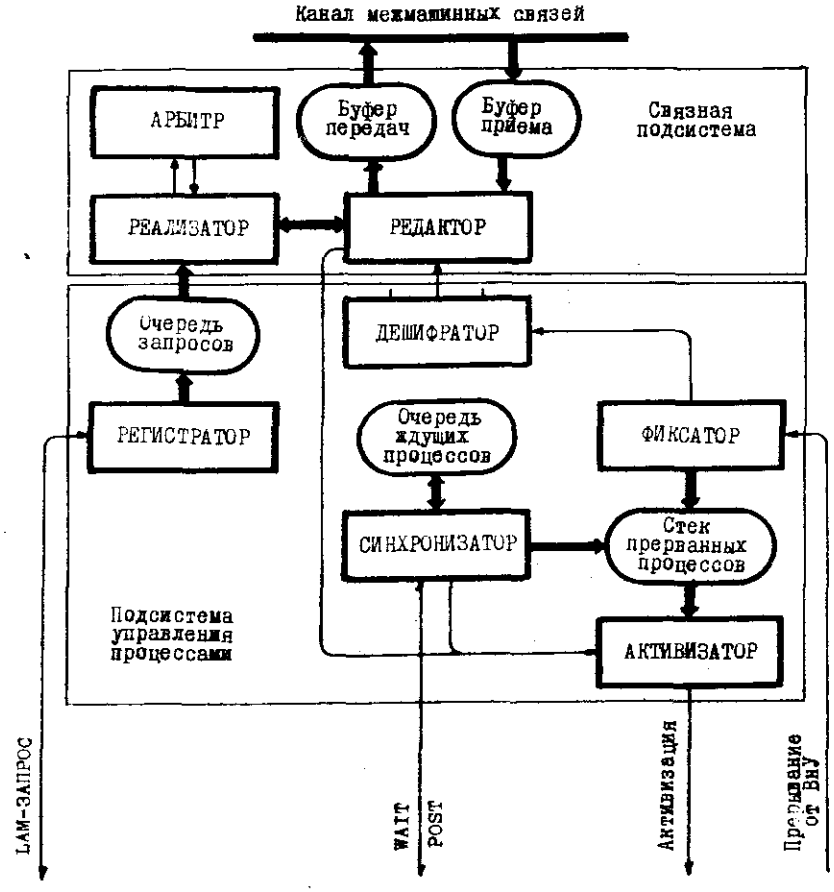

Рис. 4

2. Структура супервизора ЭМ

Функциональная структура супервизора ЭМ представлена на рис.4. Супервизор является резидентом в каждой ЭМ и функционирует в двух режимах - обычном и связном. При этом супервизор осуществляет следующие функции:

I. Обычный режим:

- анализ и обработку прерываний от внешних устройств;
- режим мультипрограммирования;
- синхронизацию внутренних взаимодействующих процессов;
- накопление запросов от программных процессов на взаимодействие через канал межмашинных связей.

II. Связной режим:

- предотвращение наложений при обменах через канал межмашинных связей;
- реализацию передачи сообщений между процессами;
- контроль межмашинных обменов.

Два режима функционирования разбивают супервизор на две подсистемы - управления процессами и связную.

При необходимости взаимодействия с другими процессами, программный процесс инициирует событие,генерируя запрос на взаимодействие к супервизору. Процесс,сгенерировавший запрос на взаимодействие,называется в ы з ы в а ю щ и м, а вовлекаемые в данное взаимодействие смежные процессы - в ы з ы в а е м ы м и. Параметры запроса на взаимодействие определяют конкретное сообщение, характер которого, в свою очередь,определяет ф у н к ц и ю взаимодействия. Функция взаимодействия в ядре определяется над данными, состоянием и управлением вызываемых процессов.

При взаимодействии процессов через канал межмашинных связей сообщения, определяемые запросом, преобразуются связной подсистемой супервизора в определенный формат (п а к е т). Каждый пакет сообщения передается супервизором через канал связи в режиме "запрос-ответ". При взаимодействии программные асинхронные процессы используют процедуры синхронизации *POST* и *WAIT*, которые реализуются в супервизоре блоком СИНХРОНИЗАТОР. Взаимодействие программных и аппаратных процессов осуществляется через систему прерываний ЭМ. При наступлении прерывания от внешнего устройства вектор состояния прерванного процесса фиксируется блоком ФИКСАТОР в очереди прерванных процессов, имеющей структуру стека. После фиксации в стеке вектора состояния супервизорным блоком ДЕШИФРАТОР

производится выбор внешнего устройства, требующего обслуживания. Этот же блок активизирует конкретный процесс обслуживания внешнего устройства. В частности, если ДЕШИФРАТОР выбрал для обслуживания прерывание от системного устройства, то активизируется блок РЕДАКТОР связной подсистемы, который будет обслуживать входной пакет поступившего сообщения.

После обслуживания прерывания блоком АКТИВИЗАТОР производится активизация ранее прерванного программного процесса восстановлением из стека вектора состояния.

Запросы на взаимодействие процессов через канал межмашинной связи фиксируются блоком РЕГИСТРАТОР в очереди запросов. Эта очередь будет обслужена связной подсистемой супервизора после получения ЭМ статуса управляющей. Осуществлением межмашинного обмена и регистрацией ошибок при взаимодействии процессов через канал связи занимается блок РЕАЛИЗАТОР. Для выбранного из очереди запроса формируется (обращением к блоку РЕДАКТОР) выходной пакет определенного формата (рис. 5).

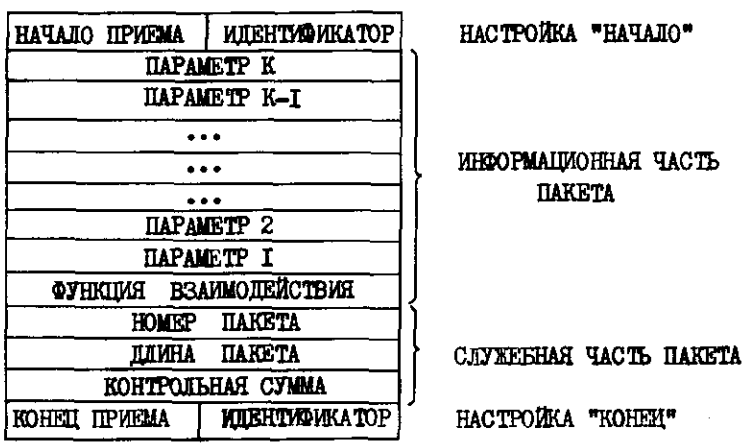

Рис. 5

Передача пакета через канал межмашинной связи организуется программно в симплексном режиме. Основной принцип симплексного режима передачи данных (рис.6) базируется на двух идеях: первая заключается в нумерации пакетов сообщения, так что связная подсистема управляемой ЭМ может просигнализировать супервизору управ -

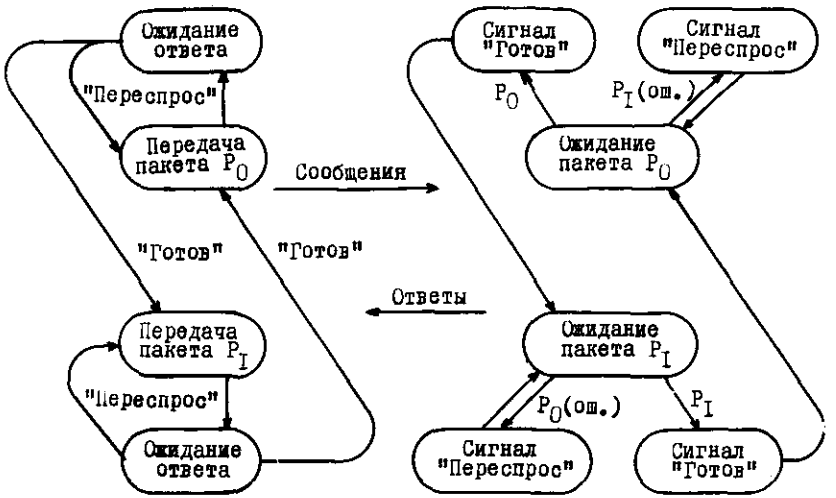

Рис. 6

ляющей, был ли данный безошибочный пакет принят ранее; а вторая - в подтверждении супервизором управляемой ЭМ каждого принятого пакета. Связная подсистема супервизора управляемой ЭМ посылает ответный сигнальный пакет "Готов" (подтверждение), когда входной пакет принят правильно, и сигнальный пакет "Переспрос" (отрицательное подтверждение), когда входной пакет принят с ошибками. Проверкой принимаемых пакетов на ошибки занимается РЕДАКТОР связной подсистемы.

Заложенный в ядре операционной системы симплексный режим передачи данных обеспечивает высокую надежность передачи сообщений при условии обнаружения ошибок. Необнаруженные ошибки могут привести к некоторым затруднениям. Например, если сигнальный пакет "Переспрос" искажается в сигнальный пакет "Готов" и это остается необнаруженным, то пакет передаваемого сообщения будет потерян. Если предполагать полную потерю пакета в канале межмашинной связи, то система может зациклиться (работа системы приостановится, если управляющая ЭМ ожидает ответа на предыдущий пакет, между тем как ответный пакет потерян). Данная форма зацикливания снимается введением таймаута, длительность которого зависит от вида межмашин-

ного обмена. Таймаут взводится супервизором управляющей ЭМ,и после его окончания производится соответствующее извещение процесса ожидания ответа (в супервизоре процесс ожидания реализуется блоком РЕАЛИЗАТОР).

Ядро операционной системы СУММА использует смешанный режим управления каналом межмашинной связи,при котором канал связи рассматривается как общий ресурс системы. Для управления доступом процесса к каналу связи супервизор ЭМ создает очередь запросов на данный ресурс, но в каждый момент времени ядро операционной системы разрешает использование канала только одному супервизору ядра, который обслуживает свою очередь запросов.

Каждая ЭМ системы имеет р а б о ч и й и д е н т и ф и к а т о р, являющийся ее адресом при межмашинных обменах. ЭМ, управляющая в данное время каналом связи, помечена,кроме рабочего, и д е н т и ф и к а т о р о м у п р а в л е н и я и называется у п р а в л я ю щ е й. ЭМ, взаимодействующие с управляющей, называются у п р а в л я е м ы м и.

При функционировании системы любая ЭМ может быть в один момент времени управляющей, а в другой - управляемой. Процедура, реализующая арбитраж и динамическую передачу статуса управляющей конкурирующим ЭМ, носит название процедуры реализации предложений, разрыва связей [7] или конкурса подрядчиков [8].Данная процедура построена таким образом, что управляющей может стать только одна ЭМ, даже если одновременно несколько ЭМ предложили себя в качестве управляющей. При арбитраже ядро операционной системы исходит из относительных приоритетов ЭМ. Приоритет ЭМ в данном ядре представляет собою текущий размер очереди запросов в данной ЭМ. Как только некоторой ЭМ присваивается статус управляющей, супервизор данной ЭМ переходит в связной режим и начинает обслуживание очереди запросов на межмашинный обмен. После обслуживания запросов ЭМ отказывается от функций управляющей, реализуя процедуру арбитража. Если после арбитража выясняется, что запросов на межмашинный обмен нет (очереди запросов во всем ядре пусты), управляющая ЭМ ожидает появления запросов в ядре, который выводит данную ЭМ на новый цикл арбитража. В супервизоре смену управляющей ЭМ осуществляет блок связной подсистемы АРБИТР.

3. Язык супервизора

Входным языком ядра операционной системы ОВС СУММА является язык *MACRO-8C*, представляющий собой модификацию языка *MACRO-8*. Модификация языка *MACRO-8* достигнута за счет определения некоторых макрокоманд (макросов) и идентификаторов, позволяющих создавать программы с учетом ядра операционной системы. Ассемблер *MACRO-8C* используется для трансляции исходной программы,написанной на языке *MACRO-8C*, в программу, готовую к реализации на ОВС СУММА.

Общая форма запроса на взаимодействие процессов через канал межмашинных связей на языке *MACRO-8C* выглядит следующим образом:
LAM ERROR, FUNC, PARAM, ABON, DECB.

Операнд *ERROR* определяет метку, по которой супервизор передает управление в случае некорректности задания запроса. Причину некорректности супервизор задает как входной параметр процедуры *ERROR*.

Операнд *FUNC* указывает функцию взаимодействия,которая определяется идентификаторами *FREAD* (читать), *FWRITE* (писать), *FPOST* (известить) и *FSTART* (пустить).

Операнд *PARAM* определяет имя списка исходных параметров заданной функции, структура которого конкретизируется ниже.

Операнд *ABON* определяет имя списка рабочих идентификаторов ЭМ, вовлекаемых во взаимодействие данным запросом:

$$ABON, N;\ JD1;\ JD2;\ \ldots;\ JDN,$$

где N – мощность списка, а JD_j – рабочий идентификатор j-й ЭМ-абонента.

Операнд *DECB* определяет имя таблицы управления событием (взаимодействием): *DECB, 0; 0*. Необходимо отметить, что запрос обслуживается супервизором в последовательности, заданной списком *ABON*.

Для функции чтения (*FREAD*) исходные параметры задаются списком *PARAM*, имеющим следующую структуру:

$$PARAM, ADRIN;\ DL1;\ ADROT;\ DL2,$$

где *ADRIN* – имя приемного поля данной ЭМ; *DL1* – длина приемного поля; *ADROT* – имя поля данных, читаемого в ЭМ-абонентах; *DL2* – длина читаемого поля данных. Для функции чтения должно выполняться условие $DL1 \geq DL2 \times N$, где N – мощность списка *ABON*.

Для функции записи (*FWRITE*) исходные параметры задаются списком *PARAM*, аналогично функции чтения, с учетом того, что

ADRIN – имя поля приема данных в ЭМ-абонентах, рабочие идентификаторы которых определены списком *ABON* исходного запроса; *ADROT* – имя передаваемого поля данных; *DL1*, *DL2* – длины приемного и передаваемого полей. Для функции записи должно выполняться условие $DL1 \geq DL2$.

В случае запроса на извещение процессов (*FPOST*) исходные параметры извещения задаются списком

PARAM, CODE; DECBP,

где *CODE* – задаваемый код извещения; *DECBP* – имя таблицы управления событием, определенной на множестве вовлекаемых во взаимодействие через канал межмашинной связи процессов.

Для функции пуска (*FSTART*) в списке *PARAM* определяется вектор состояния пускаемого процесса:

PARAM, AC; LINC; PC,

где *AC* – значение накопительного регистра; *LINC* – значение дополнительного разряда накопительного регистра; *PC* – значение счетчика команд запускаемого процесса.

Запросы процессов на внутреннюю синхронизацию описываются в языке с помощью макросов *WAIT DECB* и *POST DECB, ADRCD*, где *DECB* – имя таблицы управления событием; *ADRCD* – идентификатор кода извещения.

4. Пример программирования на *MACRO-8C*

В качестве примера рассмотрена программа, которая в процессе выполнения порождает некоторое количество процессов в системе. В ходе выполнения (тело программы не приведено) каждый процесс определяет некоторую булевскую переменную *OMEGA*. Конъюнкция значений переменных *OMEGA* от всех порожденных процессов, распределенных по ЭМ ОВС, определяет дальнейший ход параллельного процесса: либо останов каждого процесса (в случае истинного значения конъюнкции), либо новый цикл вычислений (в случае ложного значения). Структурно процесс вычислений представлен на рис.7.

Данная операция, определяющая дальнейший ход параллельного процесса в зависимости от обобщенного условия всех процессов, называется операцией обобщенного условного перехода (ОУП).

Следует заметить, что над булевскими переменными *OMEGA* может быть произведена любая логическая функция, что достигается простым перепрограммированием соответствующего участка программы. Более того, переменная *OMEGA* может быть не только булевской, но

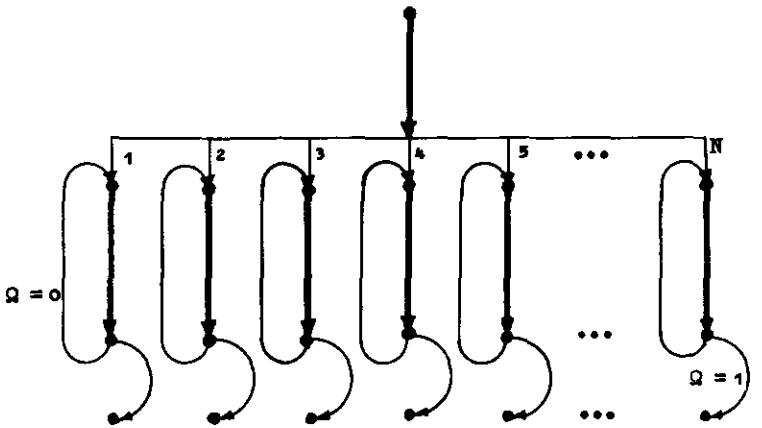

Рис. 7

и вещественной, над которой можно определить любую функцию. Например, можно представить ОВС как параллельный процессор с длиной сумматора, равной $I2 \times N$ разрядов, где N определяется количеством ЭМ в требуемой подсистеме. При такой организации можно производить конвейерную обработку данных. Ниже приводится листинг программы, использующей обобщенный условный переход.

```
/ФРАГМЕНТ ПРОГРАММЫ, ИСПОЛЬЗУЮЩЕЙ
/ОБОБЩЕННЫЙ УСЛОВНЫЙ ПЕРЕХОД
                /MACRO-8C
1               LAM ERROR1,FWRITE,PARAM1,ABON,DECB1
2               LAM ERROR2,FSTART,PARAM2,ABON,DECB2
                WAIT DECB1;WAIT DECB2
                HLT
        BEGIN,  *
                *
                *
3               DCA OMEGA
4               IAC
                DCA SUM
5               LAM ERROR3,FREAD,PARAM3,ABON,DECB1
                WAIT DECB1
```

```
6                JMS FAND
                 RAR
7                SNL
8                JMP INWAIT
9                LAM ERROR4,FREAD,PARAM4,ABON,DECB1
                 WAIT DECB1
10               LAM ERROR5,FWRITE,PARAM5,ABON,DECB1
11               LAM ERROR6,FWRITE,PARAM6,ABON,DECB2
12               LAM ERROR7,FPOST,PARAM7,ABON,DECB3
                 WAIT DECB1;WAIT DECB2;WAIT DECB3
13   INWAIT,     WAIT DECB1
14               JMS FAND
                 RAR
15               SNL
16               JMP BEGIN
17               HLT
     FAND,       0
                 CLA CLL
                 TAD I ARR
                 RAR
                 SNL CLA
                 JMP I FAND
                 ISZ ARR
                 ISZ IR
                 JMP FAND+1
                 TAD (ARRAV)
                 DCA ARR
                 TAD (-N)
                 DCA IR
                 IAC
                 JMP I FAND
     PARAM1,     BEGIN;DL;BEGIN;DL
     PARAM2,     0;0;BEGIN
     PARAM3,     ARRAV;N;SVN;1
     PARAM4,     ARRAV;N;OMEGA;1
     PARAM5,     ARRAV;N;ARRAV;N
     PARAM6,     SVN;1;NUL;1
     PARAM7,     0;DECB1
     ABON,       N;ID1;ID2;...;IDN
```



```
     ARRAV,      0;0;0;...;0
     DECB1,      0;0
     DECB2,      0;0
     DECB3,      0;0
     ARR,        ARRAV
     IR,         -N
     OMEGA,      0
     CV,         0
     SVN,        0
     NUL,        0
     N=
     DL=
     ERROR1=
     ERROR2=
     ERROR3=
     ERROR4=
     ERROR5=
     ERROR6=
     ERROR7=
```

Приведенная программа работает следующим образом:

I. Записывается программный сегмент, начиная с метки *BEGIN* длиной *DL* во все, определенные в списке *ABON* элементарные машины ОВС.

2. Производится запуск тиражированных программ по метке *BEGIN*.

3. В результате счета программы определяется значение булевской переменной *OMEGA*.

4. Производится отметка входа процесса в операцию обобщенного условного перехода через переменную *SYN*.

5. Осуществляется чтение всех отметок *SYN* от всех процессов в массив *ARRAY*.

6. Выполняется функция $\&_{i=1}^{N} SYN(i)$.

7. Проверяется вхождение всех процессов в операцию обобщенного условного перехода.

8. Если не все процессы вошли в операцию ОУП, то происходит переход в ожидание события, когда все процессы войдут в ОУП, далее – пункт I3.

9. Если все процессы вошли в операцию ОУП, то осуществляется чтение всех признаков *OMEGA* в массив *ARRAY*.

I0. Производится запись массива *ARRAY*, содержащего все признаки *OMEGA*, по всем процессам.

II. Осуществляется "обнуление" синхронизирующих признаков *SYN* во всех процессах.

I2. Производится извещение процессов о наступлении операции ОУП.

I3. Ожидание входа всех процессов в операцию ОУП.

I4: Вычисление функции $\Omega = \&_{i=1}^{N} OMEGA(i)$.

I5. Проверка значений функции Ω.

I6. Значение $\Omega = 0$ (ложь): переход в начало по метке *BEGIN*.

I7. Значение $\Omega = 1$ (истина): останов.

Ядро операционной системы СУММА приспособлено для реализации процессов достаточно общего вида и имеет средства для логического объединения и синхронизации отдельных процессов. Это обеспечивает решение на однородной вычислительной системе сложных задач, представимых в виде совокупности алгоритмов (процессов). Выходным языком ядра операционной системы является макроассемблер *MACRO-8C*. Объем оперативной памяти, занимаемый супервизором элементарной машины, составляет $2K$ слов. На реализацию описанного ядра операционной системы СУММА было затрачено I5 человеко/месяцев.

В заключение авторы благодарят Воробьева В.А. за содействие при спецификации проекта.

Литература

I. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. "Наука", Новосибирск, I966.

2. АФАНАСЬЕВ В.П., ГОЛОДОК В.Л., ГОРЯЧКИН В.И., ЕРЕМИН А.П., ЖЕЛТОВ М.П., ИЛЬИН М.Н., СЕДУХИН С.Г., ТОМИЛОВ Ю.Ф., ХОРОШЕВСКИЙ В.Г., ШУМ Л.С. Вычислительная система СУММА. -В кн.: Вычислительные системы. Вып. 60. Новосибирск, I974.

3. АФАНАСЬЕВ В.П., ИЛЬИН М.Н., СЕДУХИН С.Г., ШУМ Л.С. Системное устройство однородной вычислительной системы СУММА. -В кн.:Однородные вычислительные системы и среды. Материалы IУ Всесоюзной конференции. Ч.I. Киев, "Наукова думка", I975, стр. 47-48.

4. УОРВИК М. Введение в операционные системы. Основные понятия. -В кн.: Супервизоры и операционные системы. М., "Мир", I972.

5. HANSEN B. The Nucleus of a Multiprogramming System. - "Communs ACM",1970,v.13,Apr.,p.237-241,250.

6. ВОРОБЬЕВ В.А., СЕДУХИН С.Г., КАШУН И.Н. Исследование децентрализованных дисциплин взаимодействия между элементарными машинами однородной вычислительной системы. -В кн.: Однородные вычислительные системы и среды. Материалы IУ Всесоюзной конференции, Ч.I. Киев, "Наукова думка", I975, с. 36-38.

7. МИЛЗ Д.М. Программное обеспечение систем передачи данных.-"Труды института инженеров по электротехнике и радиоэлектронике", № II: I973, с. 84-94.

8. РАЙЛИ. Вычислительные сети и повышение эффективности вычислительных работ. -"Электроника", I974, № 9, с. 34-47.

9. КНУТ Д. Искусство программирования для ЭВМ. Т.I. Основные алгоритмы. М., "Мир", I976, с. 25-35.

Поступила в ред.-изд.отд.
28 ноября I976 года