На этом заканчивается один цикл моделирования системы и управление снова передается на блок *2*.

Блоки *5, 8, 11, 16* и *21* выполняют вспомогательные функции по управлению процессом моделирования. В начале моделирования $l=0$. После вычисления момента наступления события 1 (блок *4*) значение l увеличивается на единицу (блок *5*). Вслед за расчетом момента наступления события 2 (блок *10*) также происходит увеличение l на единицу (блок *11*). После начала моделирования величина l может быть равной только единице или двум. Если $l=1$, это означает, что в модели выработано одно из двух событий. Если $l=2$, то в модели выработаны оба события. Только тогда управление передается через блок *7* на блок *14*. Затем происходит реализация одного из событий в блоках *15* или *20*. В блоке *15* реализуется событие 2, в блоке *20* — событие 1. В результате реализации любого из событий значение l уменьшается на единицу (блоки *16, 21*).

Когда осуществляется событие 2, очередной порожний контейнер из запаса должен быть поставлен под загрузку. Однако прежде необходимо проверить, есть ли в запасе порожние контейнеры; это выполняет блок *17*. Если контейнеры имеются, производится уменьшение запаса на единицу (блок *18*). В противном случае моделируется процесс ожидания прибытия партии порожних контейнеров (блок *19*).

Блоки *13, 25* управляют процессом окончания моделирования. Когда значение τ достигает заданного времени моделирования T, управление из блока *12* передается в блок *13*, в котором величине k присваивается значение, равное единице. Процесс моделирования продолжается до тех пор, пока значение t не достигнет заданного времени моделирования. Только после этого управление из блока *6* через блок *25* передается в блок *26*. Если заданного времени моделирования первой достигает величина t, то процесс моделирования продолжается, пока и значение τ не станет равным T. Блок *27* выводит результаты моделирования на печать.

Предложенная имитационная модель позволяет провести эксперимент с целью определить наилучшие условия функционирования системы.

Поступила в редколлегию 30.11.83.

УДК 681.142

А. С. ДЕРЕВЯНКО

О НЕКОТОРЫХ СПОСОБАХ СУПЕРВИЗОРНОГО ПРОГРАММИРОВАНИЯ В ОС ЕС

Эксплуатация современных вычислительных систем третьего поколения немыслима без использования их развитого прог-

раммного обеспечения и операционных систем (ОС). ОС ЕС, применяемая ныне почти повсеместно на моделях ЕС ЭВМ среднего и большого быстродействия, является одной из наиболее совершенных современных ОС и предоставляет пользователю широкие программные возможности. Однако, как отмечалось и разработчиками этой системы, ориентация на наиболее универсальное применение привела к тому, что для решения некоторых специальных задач системные средства оказываются малоэффективными или вообще непригодными [1]. В то же время доступ программиста к аппаратным средствам ЭВМ ограничен теми средствами, которые предоставляет ему ОС. На аппаратном уровне это ограничение достигается за счет наличия привилегированных команд и механизма защиты памяти. Возможность применения привилегированных команд и ключ доступа к памяти задаются соответствующими полями в Слове Состояния Программы (ССП). Но сами команды изменения ССП относятся к числу привилегированных, недоступных для программы пользователя [2]. Выполнение программы с доступом ко всей оперативной памяти ЭВМ и с правом выдачи привилегированных команд будем называть супервизорным (*SVC*) режимом выполнения, а написание таких программ — *SVC*-программированием. Необходимость в *SVC*-программировании возникает при осуществлении нештатных режимов ввода-вывода, при разработке систем коллективного доступа, систем реального времени и в других задачах.

Супервизор ОС ЕС состоит из набора программ, имеющих номера от 0 до 255. Программы супервизора (*SVC*-программы) вызываются командой обращения к супервизору *SVC*, которая порождает *SVC*-прерывание [2]. По спецификациям системы последние шесть *SVC*-номеров отводятся для программ пользователя, которые могут быть включены в библиотеку супервизора и выполняются в *SVC*-режиме [3]. Возможность добавления программ пользователя в библиотеку супервизора и номера этих программ должны быть указаны на этапе генерации ОС. Таким образом, объем и количество написанных пользователем *SVC*-программ ограничены размером памяти в библиотеке супервизора и числом резервных *SVC*-номеров, отведенных при генерации. Как правило, в распределении памяти для библиотеки супервизора во время ее создания не предусматривается возможность внесения в библиотеку значительных добавлений, а ресурс свободных *SVC*-номеров быстро исчерпывается, поскольку целый ряд программных средств коллективного доступа и организации работы системы должен записывать свои *SVC*-программы в библиотеку супервизора.

Нам пришлось использовать *SVC*-программирование при разработке программного обеспечения нештатных внешних устройств ЕС ЭВМ, так как программирование ввода-вывода на физическом уровне невозможно без употребления привилегированных команд ввода-вывода и доступа к младшим адресам оперативной памяти. Указанные выше ограничения не позволили записать в библиотеку супервизора все программные модули, работающие в *SVC*-режиме, и заставили искать пути сообщения *SVC*-режима любой программной секции, выступающей по отношению к ОС как программа пользователя. Программа, примененная нами для этих целей, заносилась в библиотеку супервизора, как это описано выше, однако обладала рядом преимуществ. Указанная программа обеспечивала выполнение любой программной секции пользователя в *SVC*-режиме. Адрес программной секции пользователя сообщался *SVC*-программе через один из общих регистров. Объем данной *SVC*-программы минимальный, так как она состоит лишь из двух команд: передачи управления с возвратом (вместе с управлением программной секции пользователя передается и *SVC*-режим) и команды *SVC* 3, обеспечивающей возврат из *SVC*-прерывания.

Преимуществами этой программы перед *SVC*-программами, полностью реализующими все действия, которые необходимо выполнить в *SVC*-режиме, являются: минимальный объем, возможность использования одной и той же *SVC*-программы в различных задачах, требующих *SVC*-программирования. Кроме того, поскольку все действия в *SVC*-режиме реализуются в программных секциях пользователя, они могут быть легко изменены без вмешательства в системные библиотеки.

Применение описанной *SVC*-программы в значительной степени уменьшает неудобства для пользователя, связанные с ограниченностью возможностей *SVC*-программирования, но не ликвидирует их полностью. Поэтому для дальнейших разработок нами применен иной метод получения *SVC*-режима пользовательской программой. Для его реализации использовался принятый в ОС ЕС порядок обработки прерываний [4]. При прерывании текущее ССП запоминается в Блоке Запросов. Сущность метода состоит в том, что программа для получения *SVC*-режима прерывает сама себя, находит в Очереди Блоков Запросов свой Блок Запросов и в хранящемся в этом блоке ССП устанавливает признаки *SVC*-режима.

Для выполнения самопрерывания программы используются особенности работы ОС с библиотечными наборами данных [5]. Расширение системной макрокоманды поиска заданного раздела библиотеки *(FIND)* содержит команду *SVC*-обращения к системной *SVC*-программе. Последняя обращается, в свою очередь, к системной подпрограмме подвода блока *(POINT)*. Адрес подпрограммы *POINT* после выполнения мак-

рокоманды *OPEN* — «открыть набор данных» — помещается системой в Блок Управления Данными, который доступен для программиста. Изменив этот адрес, программист может обеспечить выполнение своей программной секции вместо подпрограммы *POINT*. Эта программная секция находит Блок Запросов, соответствующий программе, и изменяет в хранящемся в нем ССП соответствующие поля. При возврате из прерывания ССП, хранившееся в Блоке Запросов, становится текущим и программа получает *SVC*-режим.

Поиск Блока Запросов ведется по цепочке ссылок: от Таблицы Векторов Связи, адрес которой находится в оперативной памяти в поле с адресом 16, — к Блоку Управления Задачей, от него — к Очереди Блоков Запросов. Следует иметь в виду, что если подпрограмма установки *SVC*-режима вызывается макрокомандой *LINK*, то Блок Запросов, соответствующий программе, является не последним, а предпоследним в очереди.

Программная реализация второго метода хотя и невелика по объему, но достаточно сложна, так как предполагает знание структуры системных блоков и таблиц. Кроме того, перед выдачей макрокоманды *FIND* необходимо построить управляющие блоки для ввода-вывода и, пользуясь средствами метода доступа *EXCP*, которые описаны в работе [3], выполнить начальную установку на устройстве внешней памяти прямого доступа. Это предъявляет повышенные требования к разработчику, однако подразумевается, что квалификация программиста, разрабатывающего *SVC*-программы, достаточно высока.

Таким образом, применение программ, разработанных по описанной методике, позволяет обеспечить возможность *SVC*-программирования при минимальной модификации ОС. Если же в данной конфигурации ОС нельзя осуществить *SVC*-программирование, с помощью предложенного способа можно обходить это ограничение.

Список литературы: 1 *Брукс Ф. П.* Как разрабатываются и создаются программные комплексы.— М.: Наука, 1979.— 151 с. 2. *Вычислительная* система ИБМ/360. Принципы работы /Под ред. В. С. Штаркмана.— М.: Сов. радио, 1969.— 440 с. 3. *Пеледов Г. В., Райков Л. Д.* Введение в ОС ЕС ЭВМ.— М.: Статистика, 1977.— 120 с. 4. *Супервизор* ОС ЕС ЭВМ / В. В. Наумов, Г. В. Пеледов, Ю. А. Тимофеев и др.— М.: Статистика, 1977.— 87 с. 5. *Операционная* система ИБМ/360. Супервизор и управление данными / Под ред. А. И. Илюшина.— М.: Сов. радио, 1973.— 312 с.

Поступила в редколлегию 22.11.83.

УДК 519.8

А. В. ГОРЕЛЫЙ, канд. техн. наук,
В. Я. ЗАРУБА, канд. техн. наук, *С. В. СУХОРУКОВ*

ВОПРОСЫ АВТОМАТИЗАЦИИ ПЕРСПЕКТИВНОГО ПЛАНИРОВАНИЯ В ОТРАСЛИ ГРАЖДАНСКОЙ АВИАЦИИ

Автоматизированная подсистема перспективных плановых расчетов в отрасли гражданской авиации (АСПР ГА-П) пред-