ЛЕНИНГРАДСКИЙ ОРДЕНА ЛЕНИНА
И ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. А. А. ЖДАНОВА

МОСКОВСКИЙ ОРДЕНА ЛЕНИНА
И ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. М. В. ЛОМОНОСОВА

Вычислительный центр ЛГУ и Вычислительный центр МГУ

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ВОПРОСЫ КИБЕРНЕТИКИ

Выпуск 8

ИЗДАТЕЛЬСТВО ЛЕНИНГРАДСКОГО УНИВЕРСИТЕТА 1971

Н. П. БРУСЕНЦОВ, Е. А. ЖОГОЛЕВ

СТРУКТУРА И АЛГОРИТМ ФУНКЦИОНИРОВАНИЯ МАЛОЙ ВЫЧИСЛИТЕЛЬНОЙ МАШИНЫ

1. ВВЕДЕНИЕ

Настоящее сообщение подводит итог очередного этапа ведущейся в Вычислительном центре МГУ работы по созданию эффективных структур малых автоматических цифровых вычислительных машин.

Результаты первого этапа (1956—1960 гг.) этой работы были воплощены в машине «Сетунь» [1, 2], внедренной в серийное производство в 1962 г. Разработка программного обеспечения для этой машины [3, 4, 5] позволила уточнить требования, предъявляемые к машинам этого класса, и выявить пути усовершенствования их структуры. Некоторые теоретические результаты в этом направлении были опубликованы в работе [6].

В нашей последующей работе был сохранен в качестве основы и получил дальнейшее развитие успешно реализованный в машине «Сетунь» принцип экономии оборудования, осуществляемого инженерным путем (hardware), за счет передачи ряда выполняемых обычно этим путем операций программному оборудованию (software). Использование этого принципа позволяет не только существенно удешевить машину, но и сделать ее более универсальной в том смысле, что расширение области эффективного применения такой машины в нужном направлении достигается лишь соответствующей модификацией ее программного оборудования без изменения аппаратной части. При этом система команд машины и другие ее возможности, реализованные аппаратно и доступные программным путем, должны создавать наиболее благоприятные условия для построения соответствующего программного оборудования.

В настоящем сообщении дано законченное алгоритмическое описание разработанного на данном принципе варианта структуры и системы команд малой машины, получившей условное название «Сетунь-70». Это описание является документом, в котором, с одной стороны, точно сформулировано задание на разработку логики машины, а с другой стороны, содержится вся информация, необходимая для разработки программного оборудования этой машины.

Описываемые структура машины и ее система команд многократно обсуждались на совместном семинаре лаборатории систем автоматического программирования и математического обеспечения и проблемной лаборатории ЭВМ ВЦ МГУ. Авторы выражают благодарность всем участникам этого семинара и в особенности Х. Рамилю Альваресу, замечания которого способствовали устранению ряда ошибок и улучшению текста данного описания машины.

2. ОСНОВНЫЕ ОСОБЕННОСТИ МАШИНЫ

Рассматриваемые ниже особенности машины «Сетунь-70» обусловлены стремлением создать основу для разработки программной надстройки над аппаратной частью машины такой, которая допускала бы простую ее перестройку и возможность эффективной реализации, сводя при этом к минимуму количество необходимого аппаратного оборудования. В связи с этим предполагалось, что внутренний язык машины не предназначен для широкой массы пользователей, поскольку система программного обеспечения должна включать возможность составления программ на языках более высокого уровня. На внутреннем языке машины должна быть создана лишь определенная часть программной надстройки.

Характерными особенностями машины «Сетунь-70» являются следующие.

2.1. Послоговая адресация памяти. Базовой единицей информации в машине «Сетунь-70» является слог, состоящий из 6 троичных цифр (как и в машине «Сетунь» [1, 2], используется троичный код, обладающий по сравнению с двоичным кодом рядом существенных преимуществ [7]). Слогу соответствует непосредственно адресуемая ячейка оперативной и постоянной памяти машины (памяти первого уровня, см. ниже).

Слогом представляется всякая команда машинного языка, а также номер страниц памяти, адрес слога в пределах страницы, и т. п. Слог позволяет закодировать любой знак алфавита, состоящего из $3^6=729$ знаков, что с избытком покрывает действующие ныне и перспективные стандартные коды. В язык машины включены специальные операции над слогами.

Арифметическое устройство машины рассчитано на действия с числами, представленными тремя слогами (18 троичных разрядов), а результат операции умножения представляется шестью слогами, но вместе с тем число, посылаемое в это устройство из памяти, может состоять и из одного слога или из двух слогов — оно автоматически дополняется до трехсложного числа добавлением нулей в младшие разряды.

Послоговая адресация памяти позволяет более экономно использовать память (за счет послогового кодирования информации) одновременно с ускорением процессов обработки данных. В то же время это не приводит к сколь-нибудь заметному усложнению машины.

2.2. Польская инверсная запись в качестве внутреннего языка машины. Работы по программному обеспечению машины «Сетунь» показали, что трансляция с

алгоритмических языков существенно облегчается, если выходным языком трансляторов служит польская инверсная запись, которая на машине «Сетунь» интерпретировалась [4, 5]. Следующим шагом на этом пути явился отказ от традиционной структуры машинных команд и принятие в качестве внутреннего языка машины польской инверсной записи. Такой язык удачно сочетается с послоговой адресацией памяти: программа в польской инверсной записи представляется последовательностью адресных и операционных слогов, причем каждый слог программы представляется одним (адресуемым) машинным слогом.

Так как польская инверсная запись содержит только существенные адреса и операции, то в сочетании с использованием магазинной памяти (см. ниже) при этом получается существенная экономия памяти, хранящей программы, и в значительной мере исключаются пересылки машинных слов, связанные с рабочими ячейками.

2.3. Использование магазинной организации памяти при работе процессора. Принятие в качестве машинного языка польской инверсной записи обусловило включение в структуру машины особо организованной памяти, тесно связанной с работой процессора машины и называемой *магазином*. Имеется в виду память, работающая по принципу: первым читается последнее из записанных, но еще не прочитанных слов [4, 5]. Это означает, что слова читаются из магазина в порядке, обратном их поступлению в магазин.

Пересылкой операндов из обычной памяти машины в магазин управляют адресные слоги программы. Операционные слоги управляют преобразованием хранящихся в магазине значений и пересылкой результатов из магазина в обычную память машины. При этом одноместная операция преобразует последнее из записанных в магазин значений, т. е. при этом потребляется последнее из записанных в магазин значений и записывается в магазин новое значение, являющееся результатом этой операции (если только он имеется). Двухместная операция потребляет два из последних записанных в магазин значений и записывает в магазин новое значение, являющееся результатом этой операции.

Ячейка магазина, в которую возможна запись или из которой возможно чтение в данный момент, называется *окном магазина*.

При таком использовании магазина сильно сокращается потребность в рабочих ячейках.

2.4. Двухуровневая система операций. Все операции, которые могут обозначаться отдельным слогом, образуют два уровня.

Операции первого уровня реализуются аппаратно. Они в свою очередь разбиты на две группы: в первую группу входят 27 операций, которые могут выполняться в любом режиме работы машины (см. ниже), а во вторую группу входят 27 операций, которые могут выполняться только в привилегированном режиме, точнее говоря, их выполнение запрещено в так называемом *режиме пользователя*.

Операции второго уровня связываются с определенными подпрограммами, входящими в состав программного оборудования машины. Выполнение такой операции, обеспечиваемое аппаратно, исчерпывается осуществлением перехода к выполнению связанной с этой операцией подпрограммы. Таким образом, смысл каждой такой операции определяется содержанием соответствующей подпрограммы. Это позволяет легко вводить (программным путем) необходимые операции в процессе разработки программного оборудования машины, а также перестраивать программное оборудование на эффективное решение определенного класса задач. В силу того, что каждая такая операция обозначается одним машинным слогом и наряду с любыми другими операциями может быть использована в польской инверсной записи, язык машины легко расширяется за счет включения любых достаточно содержательных операций. Такие операции мы называем *макрооперациями*. В машине имеется 27 таких операций.

2.5. Двухуровневая страничная организация памяти. Память машины «Сетунь-70» разбита на части по 81 слогу в каждой, называемые *страницами*, и состоит из двух уровней.

Страница памяти первого уровня характерна тем, что возможен доступ к любому ее слогу при выполнении какого-либо слога программы, если только номер этой страницы указан в одном из пяти специальных регистров машины. В одном из этих регистров указывается номер страницы с выполняемой программой, в другом — номер страницы, отведенной под магазин, в остальных трех — номера страниц, в которых размещаются операнды выполняемой части программы. Имеются специальные (привилегированные) операции, позволяющие изменять содержимое этих регистров, открывая доступ к одним страницам памяти первого уровня и закрывая доступ к другим.

Имеется всего 27 страниц памяти первого уровня, которые разделяются на две группы. Первая группа состоит из 9 страниц и образует оперативную память машины. К каждой странице этой памяти возможен доступ как для чтения, так и для записи слогов. Вторая группа состоит из 18 страниц и образует постоянную память машины. К каждой странице этой памяти возложен доступ только для чтения слогов.

Наличие быстродействующей постоянной памяти небольшой емкости позволяет обеспечить легкость включения программного оборудования в процессе функционирования, а также соз-

дать постоянное ядро программного оборудования. В то же время это не приводит к существенному увеличению аппаратного оборудования машины в силу сравнительно небольшой емкости этого вида памяти.

Страница памяти второго уровня характерна тем, что она способна принимать на хранение копию содержимого любой страницы первого уровня (только целиком) и позволяет снимать копию ее содержимого на любую страницу оперативной памяти. Предполагается наличие в машине памяти второго уровня достаточно большой емкости, возможно, нескольких видов (до трех), создающей определенный резерв для программного оборудования машины. По крайней мере один из видов вторичной памяти предполагается достаточно быстродействующим (время чтения одной ее страницы должно быть сравнимо со временем выполнения 81 слога программы).

2.6. Система прерываний. При возникновении в процессе выполнения программы некоторых особых ситуаций, а также при поступлении внешних сигналов процессор приостанавливает (прерывает) выполнение основной программы и осуществляет переход к выполнению специальной программы (сопоставленной этой ситуации или этому внешнему сигналу), входящей в состав программного оборудования. При этом осуществляется включение привилегированного режима работы машины, в котором уже не может осуществляться прерывание программ. В этом режиме могут выполняться любые операции машины, в том числе и операция, включающая рабочий режим машины (режим пользователя), в котором разрешены любые прерывания.

Целью системы прерываний машины «Сетунь-70» является не организация мультипрограммной работы, а повышение производительности машины в пределах выполнения одной программы и создание пользователю определенных удобств при работе на машине. Система прерываний позволяет аппаратно обнаружить такие ситуации, как исчерпание очередной страницы программы, и автоматически включить в работу стандартную служебную программу, обеспечивающую вызов в оперативную память новой страницы программы. Прерывания по сигналам от периферийных устройств позволяют организовать некоторое совмещение работы этих устройств с работой процессора. Прерывания, возникающие при нажатии кнопки на пульте управления, позволяют предоставить пользователю некоторые возможности для вмешательства в процесс выполнения программы.

3. ЯЗЫК ОПИСАНИЯ ФУНКЦИОНИРОВАНИЯ МАШИНЫ

В качестве языка описания функционирования машины «Сетунь-70» выбран несколько расширенный АЛГОЛ-60 [8]. Изменения этого языка заключаются в следующем:

3.1. Введен новый тип значения — **ternary**. Значением этого типа может быть одно из трех целых чисел: — 1, 0 или 1. В связи с этим определение метапеременной <тип> заменяется следующим:

<тип> : : = **real** | **integer** | **Boolean** | **ternary**.

В частности, вводятся и массивы, состоящие из компонент типа **ternary**.

3.2. Расширено понятие переменной. В качестве индексов переменной с индексами допускаются граничные пары в том случае, если массив, с которым связана эта переменная, состоит из компонент типа **ternary**. В связи с этим определение метапеременной <индексное выражение> заменяется следующим:

<индексное выражение> : : = <арифметическое выражение> |
<граничная пара>

Конкретное значение переменной с индексами

$$a\,[m_1 : s_1,\ \ldots,\ m_n : s_n],$$

связанной с массивом типа **ternary**, определяет значение типа **integer** по следующей формуле:

$$\sum_{i_1=0}^{s_1-m_1} \ldots \sum_{i_n=0}^{s_n-m_n} a\,[s_1 - i_1,\ \ldots,\ s_n - i_n] \times 3 \uparrow \left(\sum_{j=1}^{n} l_j \times i_j \right),$$

где $l_n = 1$,

$$l_{n-k} = l_{n-k+1} \times (s_{n-k+1} - m_{n-k+1})$$

для $k = 1,\ \ldots,\ n-1$.

Конкретные значения других видов переменных с индексами, связанных с массивами типа **ternary**, определяются по тем же формулам, если принять во внимание, что индексное выражение, состоящее из одного арифметического выражения A, эквивалентно граничной паре $A : A$.

Переменная с индексами определена только в том случае, если индексное выражение (включая и случай граничной пары) не выходит за пределы соответствующей граничной пары, указанной при описании массива, с которым связана эта переменная.

В арифметических выражениях везде, где допустимо использование значений типа **integer**, могут использоваться и рассмотренные здесь переменные, связанные с массивами типа **ternary**. Считается, что перед выполнением соответствующих операций автоматически включается функция преобразования значений к виду **integer** в соответствии с приведенными в этом пункте формулами.

3.3. Расширено понятие оператора присваивания для того случая, когда в левой части указана переменная с индексами, связанная с массивом типа **ternary**. В этом случае результат выполнения оператора вида

$$a\,[m_1 : s_1,\ \ldots,\ m_n : s_n] := A$$

можно описать следующей последовательностью действий.

3.3.1. Вычисляется значение арифметического выражения A (с учетом расширения, введенного в п. 3.2) и приводится к типу **integer**.

3.3.2. Формируется значение одномерного массива $x\,[0 : L]$ типа **ternary** таким образом, чтобы оно определяло (по формулам п. 3.2) целочисленное значение, определяемое выражением A (см. п. 3.3.1).

3.3.3. Всем компонентам переменной $a\,[m_1 : s_1,\ \ldots,\ m_n : s_n]$ присваивается значение 0 (типа **ternary**).

3.3.4. Выполняются операторы присваивания

$$a\,[s_1 - i_1,\ \ldots,\ s_n - i_n] := x\,[L - (l_i \times i_i + \ \ldots\ + l_n \times i_n)]$$

для всех $i_j \leqslant m_j - s_j$, $j = 1, 2, \ldots, n$, и при условии, что

$$l_1 \times i_1 + \ \ldots\ + l_n \times i_n \leqslant L.$$

3.4. Допускается сокращенная запись переменных с индексами. Самое правое индексное выражение, выраженное граничной парой, в переменной с индексами может быть опущено, если эта граничная пара совпадает с соответствующей граничной парой описания массива, с которым связана эта переменная. Это правило применимо и к сокращенным записям переменных с индексами, т. е. понимается рекурсивно. Например, пусть дано описание массива

ternary array $m\,[-13 : 13,\ -40 : 40,\ 1 : 6]$;

тогда переменная $m\,[i, j]$ является сокращенной записью переменной $m\,[i, j, 1 : 6]$, а переменная $m\,[i]$ является сокращенной записью переменной $m[i, -40 : 40, 1 : 6]$.

3.5. Некоторые тела процедур не выразимы в данном языке, такие тела считаются кодами и их описание дается в комментарии, который должен следовать непосредственно за заголовком этой процедуры.

3.6. Вводится понятие параллельных процедур. Обращение к такой процедуре запускает процесс ее выполнения с одновременным переходом к выполнению следующего за этим обращением оператора. Однако в языке не дается никаких средств для явного указания параллельности процедур. Все такие указания должны делаться в комментариях.

3.7. Вместо знака арифметического умножения ($\times$) используется звездочка ($*$), а вместо знака логического умножения ($\wedge$) используется знак &.

4. АЛГОРИТМ ФУНКЦИОНИРОВАНИЯ МАШИНЫ

Ниже приводится алгоритмическое описание машины «Сетунь-70». В самом внешнем блоке описаны все виды памяти машины (включая и все ее регистры), а также все работающие параллельно процедуры, используемые в этом описании. Работа процессора представлена процедурой SETUN 70. Остальные параллельные процедуры представляют периферийные устройства машины и другие датчики внешних сигналов.

В этом блоке имеется всего лишь один оператор — обращение к процедуре SETUN 70, запускающее машину в работу. Фактические параметры процедуры SETUN 70, заданные при этом обращении, указывают те реальные ресурсы машины, которые в описании процедуры SETUN 70 представлены в более удобной и компактной форме с помощью формальных параметров.

При разборе данного алгоритма может быть полезной содержательная интерпретация некоторых объектов этого алгоритма, приведенная в разделе 5.

```
begin ternary array f[−1:1, −3280:3280, −40:40, 1:6],
                    m[−13:13, −40:40, 1:6], q[−1:1, 1:8],
                    g[−1:1, 1:7], u[−1:1, 1:4], h[−1:1, 1:3],
                    c[1:32], R, Y[1:18], p[1:10], v[1:9],
                    e, k[1:6], w[1:4], hf[1:3], a[−1:1],
                    start[1:1];
  procedure INOUT(i);
    comment Процедура INOUT, будучи активизированной
    в результате обращения к ней, определяет процесс, вы-
    полняемый параллельно с процедурой SETUN 70, в кото-
    рой сразу же начинают выполняться следующие за этим
    обращением операторы. Повторное обращение к процедуре
    INOUT определяет новый процесс, не прекращая выпол-
    нения старого, определяемого той же процедурой INOUT.
    В процедуре SETUN 70 обращение к процедуре INOUT
    производится трижды с разными значениями параметра
    i = −1, 0, 1;
    begin
      procedure SWON(j);
        comment Процедура SWON(j) включает устройство
            ввода-вывода группы j с номером u[j, 3:4] и
            выключает все другие устройства ввода-вывода
            этой группы;
      procedure SWOFF(j);
        comment Процедура SWOFF(j) выключает все уст-
            ройства ввода-вывода группы j;
      procedure PASS(j);
        comment Процедура PASS(j) передает значение g[j]
          устройству ввода-вывода с номером u[j, 3:4];
```

```
      procedure TAKE(j);
        comment Процедура TAKE(j) присваивает массиву
          g[j] значение, поступившее от устройства ввода-
          вывода с номером u[j, 3:4];
LO: if a[1] = 1 then
      begin
        if u[i, 3:4] = 0 then SWOFF(i)
        else
        begin SWON(i);
          if u[i, 1] = 1 then
          begin PASS(i); v[i + 3] := 1 end
          else if u[i, 1] = -1 then
            begin TAKE(i); if g [i] = 0 then go to LO;
              v[i + 3] := 1
            end ELSE IF
        end ELSE;
        a[i] := 0
      end THEN;
      go to LO
    end INOUT;
  procedure WATCH;
    comment Процедура WATCH формирует заявки на пре-
        рывание v[1] := 1 через определенные промежутки
        времени. Как и процедура INOUT, работает парал-
        лельно с процедурой SETUN 70;
  procedure SUIT(i);
    comment Процедура SUIT, будучи активизированной в
        в результате обращения к ней, определяет процесс,
        выполняемый параллельно с процедурой SETUN 70.
        Сразу же после активизации процедуры SUIT выпол-
        няется оператор, следующий за обращением к этой
        процедуре. Процедура SUIT(i) осуществляет поиск
        страницы с номером q[i] в памяти второго уровня
        вида f[i] и завершает свою работу заявкой на пре-
        рывание v[i+6] := 1, указывающей на то, что поиск
        закончен и страницу можно использовать. Повторное
        обращение к процедуре SUIT определяет новый про-
        цесс, прекращая старый, определяемый этой процеду-
        рой с тем же значением параметра i, но не прекра-
        щая другие, ранее активизированные процессы,
        определяемые этой процедурой с другими значения-
        ми параметров;
  procedure STOP;
    comment Процедура STOP осуществляет выполнение
        оператора start := 0 в некоторый заранее не извест-
        ный момент времени. Как и процедура INOUT, ра-
        ботает параллельно с процедурой SETUN 70;
  procedure ATTENT;
```



```
    comment Процедура ATTENT формирует заявки на пре-
        рывание v[8] := 1 при нажатии специальной кнопки
        на пульте управления машины. Как и процедура
        INOUT, работ ет параллельно с процедурой SETUN 70;
  procedure SETUN 70 (T, S, t, k1, k2, k3, ka, ko, c1, ch, ca,
        ph, pa);
    comment Формальные параметры используются для вве-
        дения сокращенных обозначений наиболее употре-
        бительных частей описанных выше массивов;
    begin ternary array x[1:36], z[1:18];
      integer i;
      switch operation := MACRO, BASIC, SPEC;
      procedure INCRC;
        comment Процедура INCRC осуществляет последо-
            вательное увеличение адреса очередного выпол-
            няемого слога (значения ca);
        begin if c1 ≠ 1 & ca = 40 then v[9] := 1;
          ca := ca + 1
        end INCRC;
      procedure REFSYL;
        comment Процедура REFSYL реализует слог-ссылку;
        begin
          if c1 ≠ 1 & ka + k1 + 1 > 40 then
          begin w := -29; go to IR end;
          T := 0; m[ph, 3*pa - 1:3*pa + k1] := m[h[k2],
              ka:ka + k1 + 1];
          INCRC
        end REFSYL;
START: if start = 0 then
    begin
      start := 1; a[-1] := a[0] := a[1] := 0;
      c1 := 1; ch := 12; ca := -40;
      ph := 0; pa := 0; Y := 0;
      INOUT(-1); INOUT(0); INOUT(1);
      WATCH; ATTENT; STOP;
CYCLE: begin
        if start = 0 then go to FINISH;
        if c1 ≠ 1 then
        for i := 1 step 1 until 9 do
          if v[i] ≠ 0 then
          begin w := i - 41; v[i] := 0; go to IR end;
        k := m[ch, ca];
        if k1 = 0 & k2 = 0 then go to operation [k3 + 2]
        else
        begin
          if c1 ≠ 1 & pa = 13 then
          begin w := -30; go to IR end;
          pa := pa + 1; REFSYL
```

```
        end ELSE;
        go to CYCLE
      end CYCLE;
MACRØ:  begin
        if c1 = 1 then go to FINISH;
        if c1=0 then begin w: = — 28; go to IR end;
        if pa = 13 then begin w: = — 30; go to IR end;
        pa: = pa + 1;
        c[21:32]: = c[9:20];    c[9:20]: = 0;    c[17:20]: =
          = c[5:8];
        c[11:14]: = c[1:4]; c1: = 0; ch: = ca: = — 13;
        T: = 0; t: = m[ch, ka];
        go to CYCLE
      end MACRO
BASIC: begin switch basop: = B1, B2, B3, B4, B5, B6, B7, B8,
                             B9, B10, B11, B12, B13, B14, B15, B16,
                             B17, B18, B19, B20, B21, B22, B23, B24,
                             B25, B26, B27;
        if c1 ≠ 1 & pa = — 13 then
        begin w: = — 31; go to IR end;
        go to basop [ko + 14];
 B1:  x[1:18]: = S; x[19:36]: = Y; x: = x * 3 ↑ t;
        S: = x[1:18]; Y: = x[19:36]; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „LST“;
 B2: if abs(S) (3 ↑ 17)/2 then
        begin ca: = t; pa: = pa — 1 end
        else begin pa: = pa — 1; INCRC end;
        go to CYCLE; comment Name: „COT“;
 B3: x[1:18]: = T; x[19:36]: = Y;
        if x[1] ≠ 0 then begin i: = 1; x: = x/3 end
        else
        begin
          for i: = 0 step —1 until —34 do
            if x[2] ≠ 0 then go to EX else x: = x * 3;
          i: = 0
        end ELSE;
   EX:  e: = e + 1; T: = x[1:18]; Y: = x[19:36]; INCRC;
          go to CYCLE; comment Name: „XNN“;
 B4:  e: = e — 1; INCRC;
        go to CYCLE; comment Name: „E — 1“;
 B5:  e: = 0; INCRC;
        go to CYCLE; comment Name: „E = 0“;
 B6:  e: = e + 1; INCRC;
        go to CYCLE; comment Name: „E + 1“;
 B7:  T: = T — e * 3 ↑ 12; INCRC;
        go to CYCLE; comment Name: „T — E“;
 B8:  e: = t; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „E = T“;
```



```
 B9:  T: = T + e * 3 ↑ 12; INCRC;
        go to CYCLE; comment Name: „T + E“;
 B10: if S < 0 then begin ca: = t; pa: = pa — 1 end
        else begin pa: = pa — 1; INCRC end;
        go to CYCLE; comment Name: „CLT“;
 B11: if S = 0 then begin ca: = t; pa: = pa — 1 end
        else begin pa: = pa — 1; INCRC end;
        go to CYCLE; comment Name: „CET“;
 B12: if S ⩾ 0 then begin ca: = t; pa: = pa — 1 end
        else begin pa: = pa — 1; INCRC end;
        go to CYCLE; comment Name: „CGT“;
 B13: T: = 0; t: = ca; INCRC;
        go to CYCLE; comment Name: „T = C“;
 B14: R: = T; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „R = T“;
 B15: ca: = t; pa: = pa — 1;
        go to CYCLE; comment Name: „C = T“;
 B16: k: = t; REFSYL;
        go to CYCLE; comment Name: „T = W“;
 B17: Y: = 0; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „YFT“;
 B18: k: = t;
        if c1 ≠ 1 & ka + k1 + 1 > 40 then
        begin w: = — 29; go to IR end;
        if abs(h[k2]) ⩽ 4 then
        m[h[k2], ka: ka + k1 + 1]: =
                        m[ph, 3 * (pa — 1) — 1: 3 (pa — 1) + k1];
        pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „W = S“;
 B19: x[1:18]: = T; z: = S;
        for i: = 1 step 1 until 18 do
          z[i]: = z[i] * x[i];
        S: = z; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „SMT“;
 B20: Y: = T; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „Y = T“;
 B21: x[1:18]: = T; z: = S;
        for i: = 1 step 1 until 18 do
          begin
            if z[i] = 0 then z[i]: = x[i]
            else if z[i] ≠ x[i] & x[i] ≠ 0 then z[i]: = 0
          end FØR;
        S: = z; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „SAT“;
 B22: S: = S — T; pa: = pa — 1; INCRC;
        go to CYCLE; comment Name: „S — T“;
 B23: T: = — T; INCRC;
        go to CYCLE; comment Name: „TDN“;
```

```
B24: S := S + T; pa := pa − 1; INCRC;
     go to CYCLE; comment Name: „S + T“;
B25: x[1:18] := S; x[19:36] := 0; x := x + T * R * 3 ↑ 2;
     S := x[1:18]; Y := x[19:36]; pa := pa − 1; INCRC;
     go to CYCLE; comment Name: „LBT“;
B26: R := S; x := T * R * 3 ↑ 2; S := x[1:18]; Y := x[19:36];
     pa := pa − 1; INCRC;
     go to CYCLE; comment Name: „L * T“;
B27: x[1:18] := S; x[19:36] := Y; x := x + T * R * 3 ↑ 2;
     S := x[1:18]; Y := x[19:36]; pa := pa − 1; INCRC;
     go to CYCLE; comment Name: „LHT“;
  end BASIC;
SPEC: begin switch specop := S1, S2, S3, S4, S5, S6, S7, S8,
                             S9, S10, S11, S12, S13, S14, S15, S16,
                             S17, S18, S19, S20, S21, S22, S23, S24,
                             S25, S26, S27;
  procedure TRANSIN(l);
    comment Процедура TRANSIN(l) передает в T значение
      g[l]. Если g[l, 7] = 1, то в T передается g[l, 1:6],
      если g[l, 7] = 0, то в T передается − g[l, 1:6];
    begin T := 0;
      if g[l, 7] = 1 then t := g[l, 1:6]
      else t := − g[l, 1:6];
      a[l] := 1
    end TRANSIN;
  procedure TRANSOUT(l);
    comment Процедура TRANSOUT(l) передает в g[l]
      значение t, преобразуя его из троичного кода в не-
      который двоичный код;
    begin x[1:6] = t; g[l, 7] := 1
      for i := 1 step 1 until 6 do
        begin
          if x[i] ≠ 0 then g[l, i] := 1
          else g[l, i] := 0;
          if x[i] = − 1 then g[l, 7] := 0
          end;
        a[l] := 1;
    end TRANSOUT;
  procedure COPY(l);
    comment Процедура COPY(l) присваивает странице
      оперативной памяти, номер которой определяет t, зна-
      чение страницы с номером q[l] памяти второго уров-
      ня вида f[l];
    begin z := T; hf := z[4:6];
      if abs(hf) ⩽ 4 then
      for i := 0 step 1 until 80 do
        m[hf, 40 − i] := f[l, q[l], 40 − i]
    end COPY;
```



```
  procedure LOAD(l);
    comment Процедура LOAD(l) присваивает странице с
      номером q[l] памяти второго уровня вида f[l] значе-
      ние страницы памяти первого уровня, номер которой
      определяет t:
    begin z := T; hf := z[4:6];
      for i := 0 step 1 until 80 do
        f[l, q[l], 40 − i] := m[hf, 40 − i]
    end LOAD;
  if c1 = −1 then
  begin w := −28: go to IR end;
  if c1 = 0 & pa = − 13 then
  begin w := − 31; go to IR end;
  go to specop [ko + 14];
S1: TRANSIN(−1); INCRC;
    go to CYCLE; comment Name: „COPYG1“;
S2: TRANSIN(0); INCRC;
    go to CYCLE; comment Name: „COPYG2“;
S3: TRANSIN(1); INCRC;
    go to CYCLE; comment Name: „COPYG3“;
S4: COPY(−1); pa := pa − 1; INCRC;
    go to CYCLE; comment Name: „COPYF1“;
S5: COPY(0); pa := pa − 1; INCRC;
    go to CYCLE; comment Name: „COPYF2“;
S6: COPY(1); pa := pa − 1; INCRC;
    go to CYCLE; comment Name: „COPYF3“;
S7: z := T; q[−1] := z[5:12]; SUIT(−1); pa := pa − 1;
    INCRC;
    go to CYCLE; comment Name: „LOADQ1“;
S8: z := T; q[0] := z[5:12]; SUIT(0); pa := pa − 1; INCRC;
    go to CYCLE; comment Name: „LOADQ2“;
S9: z := T; q[1] := z[5:12]; SUIT(1); pa := pa − 1; INCRC;
    go to CYCLE; comment Name: „LOADQ3“;
S10: z := 0; z[5:6] := ph; z[9:11] := pa; T := z; INCRC;
    go to CYCLE; comment Name: „COPYP“;
S11: z[1:5] : p[1:5]; p[1:5] := p[6:10]; p[6:10] := z[1:5];
    INCRC;
    go to CYCLE; comment Name: „EXCHP“;
S12: z := T; ph := z[5:6]; pa := z[9:11]; INCRC;
    go to CYCLE; comment Name: „LOADP“;
S13: z := 0; z[1:12] := c[9:20]; c[9:20] := c[21:32];
    c[21:32] := z[1:12]; T := z; INCRC;
    go to CYCLE; comment Name: „COPYMC“;
S14: z[1:12] := c[9:20]; c[9:20] := c[21:32];
    c[21:32] := z[1:12]; c[1:4] := z[3:6];
    c[5:8] := z[9:12];
    go to CYCLE; comment Name: „RETNMC“;
```

```
S15: z := T; c[21:32] := c[9:20]; c[9:20] := z[1:12];
     pa := pa − 1; INCRC;
     go to CYCLE; comment Name: „LOADMC“;
S16: z := T; h[−1] := z[4:6]; pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADH1“;
S17: z := T; h[0] := z[4:6]; pa := pa − 1; INCRC;
     go to CYCLE; comment Name: „LOADH2“;
S18: z := T; h[1] := z[4:6]; pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADH3“;
S19: u[−1, 1:4] := t; a[−1] := 1; pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADU1“;
S20: u[0, 1:4] := t; a[0] := 1; pa := − 1; INCRC;
     go to CYCLE; comment Name: „LOADU2“;
S21: u[1, 1:4] := t; a[1] := 1; pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADU3“;
S22: LOAD(−1); pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADF1“;
S23: LOAD(0); pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADF2“;
S24: LOAD(1); pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADF3“;
S25: TRANSOUT(−1); pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADG1“;
S26: TRANSOUT(0); pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADG2“;
S27: TRANSOUT(1); pa := pa−1; INCRC;
     go to CYCLE; comment Name: „LOADG3“;
  end SPEC;
  IR: begin
    c[21:32] := c[9:20]; c[9:20] := 0; c[17:20] := c[5:8];
    c[11:14] := c[1:4]; c1 := 1; ch := 13; ca := −13;
    x[1:5] := p[6:10]; p[6:10] := p[1:5]; p[1:5] := x[1:5];
    pa := pa + 1; T := 0; t := m[ch, w];
    go to CYCLE;
  end IR;
FINISH: start := 0;
  end START
  end SETUN 70;
SETUN 70 (m[p[1:2], 3*p[3:5] − 1:3*p[3:5] + 1],
          m[p[1:2], 3*(p[3:5]−1) −1:3*(p[3:5]−1)+1],
          m[p[1:2], 3*p[3:5]−1],
          k[1], k[2], k[3], k[3:6], k[4:6],
          c[1], c[2:4], c[5:8],
          p[1:2], p[3:5]);
end Алгоритмического описания функционирования машины
    SETUN 70;
```


5. СОДЕРЖАТЕЛЬНАЯ ИНТЕРПРЕТАЦИЯ ОБЪЕКТОВ АЛГОРИТМА

Ниже дается соответствие между массивами, использованными при описании алгоритма функционирования машины „Сетунь-70“, и реальными запоминающими устройствами машины, специфицируются заявки на прерывания и поясняется характер выполнения операций разных групп в различных режимах работы машины.

5.1. Спецификация массивов.

Обозначение	Интерпретация
$f[-1:1, -3280:3280, -40:40, 1:6]$	Память второго уровня
$f[i, j, -40:40, 1:6]$	Одна страница памяти второго уровня
$m[-13:13, -40:40, 1:6]$	Память первого уровня
$m[i, -40:40, 1:6]$	Одна страница памяти первого уровня
$m[i, j, 1:6]$	Один слог памяти первого уровня
$m[-4:4, -40:40, 1:6]$	Оперативная память
$m[-13:-5, -40:40, 1:6]$ $m[5:13, -40:40, 1:6]$	Постоянная память
$q[i, 1:8]$	Указатель (регистр) страницы памяти второго уровня (одного вида), участвующей в обмене
$g[i, 1:7]$	Буферный регистр для ввода и вывода информации для i-й группы устройств ввода-вывода
$u[i, 1:4]$	Указатель режима работы устройств ввода-вывода i-й группы
$h[i, 1:3]$	Указатель открытых страниц операндов программы
$c[1:32]$	Указатель режимов и выполняемого слога программы
$c1 \sim c[1:1]$	Указатель включенного режима работы машины
$ch[1:3] \sim c[2:4]$	Указатель выполняемой страницы программы
$ca[1:4] \sim c[5:8]$	Указатель выполняемого слога программы
R[1:18]	Регистр множителя
Y[1:18]	Регистр младших разрядов

Обозначение	Интерпретация
$p[1:10]$	Указатель магазина
$ph[1:2] \sim p[1:2]$	Указатель страницы оперативной памяти, отведенной под магазин
$pa[1:3] \sim p[3:5]$	Указатель окна магазина
$v[1:9]$	Регистр заявок на прерывание
$e[1:6]$	Малый сумматор
$k[1:6]$	Регистр выполняемого слога
$k1 \sim k[1:1]$	Указатель длины слова-операнда
$k2 \sim k[2:2]$	Указатель значения индекса указателя открытой страницы операндов
$ka[1:4] \sim k[3:6]$	Адрес операнда на странице
$ko[1:3] \sim k[4:6]$	Указатель выполняемой операции
$w[1:4]$	Указатель причины прерывания
$hf[1:3]$	Указатель обменной страницы памяти первого уровня
$a[i]$	Триггер, синхронизирующий работу устройств ввода-вывода i-й группы
$start[1:1]$	Триггер состояния машины
$T \sim m[p[1:2], 3*p[3:5]-1:3*p[3:5]+1]$	Окно магазина
$t \sim m[p[1:2], 3*p[3:5]-1]$	Старший слог окна магазина
$S \sim m[p[1:2], 3*(p[3:5]-1)-1:3*(p[3:5]-1)+1]$	Ячейка магазина, расположенная непосредственно за его окном

5.2. Спецификация заявок на прерывание

$v[i]$	w	Источник или причина заявки
$v[1]$	-40	Часы
$v[2]$	-39	Сигнал от (−1-й, 0-й, 1-й) группы устройств ввода-вывода
$v[3]$	-38	
$v[4]$	-37	
$v[5]$	-36	Сигнал от памяти второго уровня (вида −1, вида 0, вида 1)
$v[6]$	-35	
$v[7]$	-34	

$v[i]$	w	Источник или причина заявки
$v[8]$	-33	Сигнал от кнопки пульта управления
$v[9]$	-32	Исчерпание страницы программы (переполнение ca)
—	-31	Исчерпание страницы магазина
—	-30	Переполнение страницы магазина
—	-29	Захват операндом следующей страницы (переполнение ka)
—	-28	Запрещенный слог

5.3. Условия выполнения операций разных типов

Тип операции	Режим работы машины		
	пользователя ($c1=-1$)	макроопераций ($c1=0$)	прерываний ($c1=1$)
Макрооперация ($k3=-1$)	Выполняется с переходом в режим макроопераций	Прерывание ($w=-28$)	Останов машины
Основная ($k3=0$)	Выполняется нормально	Выполняется нормально	Выполняется нормально
Служебная ($k3=1$)	Прерывание ($w=-28$)	Выполняется нормально	Выполняется нормально

Примечание. В режиме прерываний прерывания заблокированы.

ЛИТЕРАТУРА

1. Н. П. Брусенцов, Е. А. Жоголев, В. В. Веригин, С. П. Маслов, А. М. Тишулина. Вестник МГУ, 1962, № 4, стр. 3—12.
2. Н. П. Брусенцов, С. П. Маслов, В. П. Розин, А. М. Тишулина. Малая цифровая вычислительная машина «Сетунь». Изд. МГУ, 1965.
3. Е. А. Жоголев. Ж. вычисл. матем. и матем. физ., 1961, № 3, стр. 499—512.
4. Е. А. Жоголев. Ж. вычисл. матем. и матем. физ., 1965, № 1, стр. 67—76.
5. Е. А. Жоголев, Н. Б. Лебедева. В сб.: Вычислительные методы и программирование, вып. 9. Изд. МГУ, 1967, стр. 66—85.
6. Е. А. Жоголев. В сб.: Вычислительные методы и программирование, вып. 5. Изд. МГУ, 1966, стр. 344—361.
7. Н. П. Брусенцов. Научный отчет ВЦ МГУ, № 24-ВТ (378). Изд. МГУ, 1969.
8. Алгоритмический язык АЛГОЛ-60. Пересмотренное сообщение. Пер. с англ. под ред. А. П. Ершова, С. С. Лаврова и М. Р. Шура-Бура. Изд. «Мир», 1965.

Статья поступила в редакцию 2 февраля 1970 г.