ОБЪЕДИНЕННЫЙ
ИНСТИТУТ
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ

Дубна

2292

В.В. Захаров

НЕПОСРЕДСТВЕННЫЙ ВВОД-ВЫВОД ДАННЫХ В ВЫЧИСЛИТЕЛЬНУЮ МАШИНУ БЭСМ-3М

ЛАБОРАТОРИЯ ВЫСОКИХ ЭНЕРГИЙ

1965

В.В. Захаров x/

НЕПОСРЕДСТВЕННЫЙ ВВОД-ВЫВОД ДАННЫХ В ВЫЧИСЛИТЕЛЬНУЮ МАШИНУ БЭСМ-3М

x) On leave from CERN.

§ 1. Общие соображения

Хотя первоначально предполагалось использовать БЭСМ-3М в качестве вычислительной машины для обработки данных, получаемых в пузырьковых камерах, и поступающих с полуавтоматических измерительных устройств с ручным управлением (IEP), сейчас становится очевидным, что любая система, осуществляющая непосредственную связь с экспериментом, должна предусматривать наличие и других внешних устройств. Таким образом, любое усовершенствование системы внутренней логики вычислительной машины или какие-нибудь новые контрольные программы входа (выхода), предназначенные для непосредственной связи IEP – машина должны быть достаточно общими для того, чтобы произвести обработку данных, например, экспериментов "on-line" с использованием беспленочных искровых камер или годоскопов из счетчиков. Тогда было бы возможно ввести данные в БЭСМ-3М из любого непосредственно связанного с ней устройства, включая как измерительные устройства с бегущим лучом в форме растра, так и системы, управляемые с помощью программ (FSD). Прежде чем перейти к обсуждению определенной системы непосредственной связи, необходимо рассмотреть некоторые характеристики вычислительной машины.

Вычислительная машина БЭСМ-3М – трехадресная, 45-разрядная. Имеет магнитные барабаны и ферритовую память, рассчитанную на 4096 слов. Минимальное время, необходимое для проведения любой операции, составляет 24 мксек. Имеется один индексный регистр. В этой машине не предусматривается общих входных-выходных команд, явного входного-выходного регистра и нет устройств для прерывания. Передача всех данных в периферические системы осуществляется посредством явных[x/] команд к определенным внешним устройствам. Таким образом, ясно, что в вычислительной машине БЭСМ-3М не существует основных предпосылок для непосредственной связи, так как последняя предполагает:

[x/] Здесь и в дальнейшем явными командами называются команды системы основных операций машины.

а) непосредственный доступ (а именно, после выполнения в машине текущей команды или набора команд) за время, обусловленное внешним устройством; т.е. режим прерывания;

б) обратную связь данных в режиме управления программой; например, в виде напечатанных на внешнем устройстве машине команд операторам IEP или в виде сигналов контроля системы FSD.

Конечно, может быть так, что или вычислительная машина неспособна принять данные, потому что предыдущие данные еще не были ассимилированы, или же внешние устройства неспособны обработать поступающие данные. Любая входная-выходная система прерывания должна ясно предвидеть эту возможность – т.е. внешнее устройство должно знать о положении "занято" или "свободно" вычислительной машины и наоборот.

И, наконец, так как имеется очень мало неиспользованных кодов команд и так как нет явных команд (или наборов команд), которые производят необходимые входные-выходные операции, то любые вновь определенные команды должны быть достаточно общими для того, чтобы оказаться полезным при общем программировании.

Вначале рассмотрим проблему входа отдельно от проблемы выхода. Затем мы рассмотрим, до какой степени могут быть объединены решения этих проблем.

§ 2. Ввод данных в режиме прерывания

2.1. Требования

Общие требования, предъявляемые к программе прерывания состоят в том, что

а) программа исполняемая в данный момент должна узнать как можно скорее, что поступило требование прерывания и прерывание должно быть обеспечено как можно скорее, так как в это время уже имеются подготовленные для ввода слова-данные или слова-состояния.

б) первоначальная программа должна затем возобновить свое функционирование, как будто бы не происходило никакого прерывания.

В принципе эти требования могут быть удовлетворены либо в режиме внутреннего управления программой, либо в режиме внешнего управления, т.е. или внутренняя управляющая программа сама по себе решает проверить условие внешнего прерывания, или внешний сигнал, управляя последовательностью внутренних операций, прерывает ее. Первый путь может быть более экономным с точки зрения требований к созданию дополнительной электронной запаянной схемы и логики машины (" hardware "), но более сложным в отношении программирования (" software"), т.к. программа, подготавливая прерывание, должна организовать накопление в центральной памяти всех необходимых регистров и промежуточных результатов.

Вторая возможная система внешнего управления более экономна в отношении программирования, но при этом необходимо произвести усовершенствование схем внутреннего управления вычислительной машины.

После того как произошло прерывание, обе системы вводят данные с одной и той же скоростью. Отметим, что вторая система имеет гораздо меньшее время доступа для внешнего устройства и является более экономной для накопления данных в центральной памяти (последнее замечание особенно важно для машин с ферритовой памяться объемом всего ~4000 слов).

В случае БЭСМ-3М возникает дополнительный фактор в пользу второй системы внешнего управления, т.к. не имеется команды или группы команд, которые бы в общем случае сохраняли адрес команды, исполняемой в данный момент.

Если прерывание происходит в режиме внутреннего управления программой по команде адрес которой известен самой программе, то система прерывания может функционировать. Хотя и существуют команды, которые накапливают содержание индексового регистра и одноразрядного регистра ω , очевидно, что прерывание в режиме внутреннего управления программой для машины БЭСМ-3М было бы очень неэффективным решением.

Единственный разумный выход состоит в создании системы внешнего прерывания с таким усовершенствованием внутренних схем вычислительной машины, при котором содержимое адресного регистра, выдающего текущую команду, сохраняется автоматически (предпочтительно вместе с содержимым индексового регистра и W -регистра). В конце последовательности прерывания содержимое регистров восстанавливается, и возобновляется работа основной программы.

Следующий раздел описывает метод, посредством которого простая, но общая схема ввода данных в режиме внешнего управления прерыванием может быть претворена в жизнь для БЭСМ-3М.

2.2. Предлагаемая схема

Машина БЭСМ-3М имеет следующие 3 регистра:

С-Рег. Накапливает адрес команды, используемой в настоящее время (12-разрядный);

1-Рег. Индекс-регистр, или регистр модификации адреса (12-разрядный);

ω-Рег. Одноразрядный регистр управления;

Требуются дополнительные регистры:

S -Рег. Регистр хранения для С, 1 и ω -регистров (25-разрядный).

Р -Рег. Одноразрядный регистр прерывания; Р = 0 нормально, Р = 1 для условия прерывания.

W –Рег. 45–разрядный регистр ввода одного слова.

Самый младший разряд W –регистра, обозначаемый через W_{45} ; используется для обозначения присутствия или отсутствия вводимого слова;

W –Рег. обычно свободен, когда нет ввода, т.е. $W_{45} = 0$. Таким образом, только 44 разряда могут быть использованы для ввода данных.

Требуется также четыре дополнительные команды. Они определяются следующим образом:

1. Введение одного слова в W –Рег., если $W_{45} = 0$.

2. Условная передача на W_{45} . Если $W_{45} = 1$, то W –Рег. слово с адресом A_3 и следующая команда принимаются последовательно; в противном случае следующая команда от слова с адресом A_2 ; W –Рег. всегда свободен после этой команды.

3. Обмен содержимого C,1 и ω –регистров с содержимым S –регистра и возобновление операции.

4. "Условная передача на P": если P = 1, прием следующей команды от слова с адресом A_2; в противном случае следующая команда принимается последовательно.

ПРИМЕЧАНИЕ: Из этих пунктов ясно, что первые три команды могут быть использованы обычным образом для ввода данных из W –регистра при отсутствии любого сигнала прерывания.

Теперь рассмотрим схему прерывания, логика которой изображена на рисунке 1. В быстрой памяти вычислительной машины имеется три программы: основная программа, которая может или не может обрабатывать вводимую информацию в режиме прерывания, программа контроля прерыванием (ICP), которая всегда находится в ферритовой памяти (первая команда всегда располагается в определенном месте памяти), и подпрограмма прерывания (ISP), которая может выполнить любую функцию, необходимую при вводе входных данных. Вообще ISP будет обеспечивать буфер для ввода данных, проверять, не заполнен ли буфер, и наконец будет проводить всю проверку кодов, арифметическую проверку и отсортировку данных, которые обычно требуются при работе в режиме прерывания.

Предположим, что прерывание имеет место (P = 1) в некоторое время в течение исполнения n –ой команды основной программы. Посредством усовершенствованной схемы в вычислительной машине может иметь место такая последовательность операций:

i) Текущая команда будет полностью выполнена. Не может иметь места никакое прерывание сразу же после какого-нибудь умножения или деления, потому что могли бы потребоваться вычисления с двойной точностью и наиболее значительная часть результатов, все еще находящихся в арифметической ячейке, была бы потеряна;

ii) Содержание C,1 и ω –регистров будет передано параллельно 25–разрядному S –регистру;

iii) C –Регистр будет загружен адресом первой команды ICP, и управление передано этой команде. При этом БЭСМ–3М войдет в режим прерывания. Первая команда ICP проверяет, действительно ли $W_{45} = 0$, и если только это условие выполняется, то слово может быть введено в W –регистр. Это достигается передачей входного импульса во внешнее устройство, если только $W_{45} = 0$. Следующая команда проверяет, прибыло ли вводимое слово из внешнего устройства на W –регистр. Это осуществляется посредством команды "условной передачи на W_{45} ". Если по истечении определенного времени или после совершения некоторого числа входных циклов не прибыло ни одного слова, то ICP передает управление на особую "спасательную" (" rescue ") последовательность команд. Однако в обычном случае слово должно на W –Рег поступать в течение одного основного цикла машины (24 μc) и содержимое W –регистра должно быть передано в ячейку ферритовой памяти с адресом A_3 текущей команды программы ICP .

Далее совершается переход к программе ISP , которая будет содержать последовательность команд, зависящую от типа внешних устройств, присоединенных "on-line" и от рода обработки данных, который при этом требуется.

В общем случае ISP будет обеспечивать буферовку ввода данных в режиме прерывания, и если имеется более чем один внешний источник непосредственной связи с экспериментом, она будет сортировать данные в соответствии с их кодированием или последовательностью.

Обычный возврат от ISP к ICP является командой „условной передачи управления на P" , которая проверяет наличие или отсутствие условия прерывания. Если все же имеется требование прерывания от внешнего устройства (P = 1), то центральная программа возвращается для ввода следующего слова в W –регистр. Если не происходит дальнейшего прерывания (P = 0), то восстанавливается первоначальное содержание C, 1 и ω –регистров от S –регистра и возобновляется исполнение основной программы. Если же ICP или ISP обнаруживают ошибку во входиых данных или если имеет место любое другое ненормальное положение (например, заполнен буфер входа), то управление возвращается к „спасательной" последовательности команд, которая посылает информацию в соответствующее внешнее устройство и задерживает вход любых других данных до тех пор, пока не возобновляется нормальная работа.

§ 3. Вывод данных к непосредственно связанной аппаратуре

3.1. Общие требования

Как было уже показано, БЭСМ–3М не имеет явного регистра вывода или команды для вывода одного слова.

Вывод обычно совершается посредством буфера с 512 словами на одном из магнитных барабанов. Так как существующая система является громоздкой и неэффективной для передачи одного слова и в любом случае может быть использована по основной программе во время прерывания, целесообразно иметь дополнительный регистр вывода для передачи одного слова. Такой регистр обеспечивал бы обратную связь с аппаратурой, непосредственно связанной во время исполнения программ ICP или ISP.

Естественно, дополнительный W -регистр, предложенный для непосредственного ввода данных, мог бы и дальше использоваться для вывода, но очень удобно иметь два независимых регистра (один только для ввода, а другой только для вывода) так, чтобы обратная связь могла бы быть независимой. И, действительно, независимая обратная связь имеет существенное значение в случае уже указанной процедуры выполнения команд.

3.2. Предлагаемая схема вывода

Работа предложенной системы, логика которой приводится на рисунке 2, указывает на необходимость иметь один дополнительный регистр в качестве выводного буфера для одного слова. Этот регистр называется 0-регистром и имеет 45 разрядов. Младшим разрядом, обозначаемым 0_{45}, является управляющий разряд, который означает условие регистра вывода; условно $0_{45} = 1$ означает "выводной регистр занят". Когда бы ни происходила какая-нибудь передача из вычислительной машины в 0 -регистр, соответствующий сигнал синхронизации посылается во внешнее устройство. Когда внешнее устройство считывает 0 -регистр, оно также очищает его автоматически, оставляя $0_{45} = 0$.

Любая обычная передача от вычислительной машины к 0 -регистру любого слова, будь то данные или функция, должна иметь $0_{45} = 1$.

Требуются две дополнительные команды. Они определяются следующим образом:

1. "Условная передача на 0_{45}", если $0_{45} = 0$, посылает содержимое слова с адресом A_1 к 0 -регистру, устанавливает $0_{45} = 1$ и посылает сигнал синхронизации во внешнее устройство; следующая команда передается от A_2. Если $0_{45} = 1$, следующая команда передается последовательно.

2. "Безусловная передача к 0 -регистру" посылает содержимое слова с адресом A_1 к 0 -регистру и посылает сигнал синхронизации во внешнее устройство только в том случае, если $0_{45} = 1$. Следующая команда передается от слова с адресом A_2.

ПРИМЕЧАНИЕ: Эта команда обычно используется только для очищения 0-регистра.

Рассмотрим теперь типичную последовательность вывода. В такой последовательности вообще будет находиться более чем одно устройство, связанное параллельно с

0-регистром. Логика в программе управления выходом (OCP) устанавливает связь между вычислительной машиной (а именно, 0-регистром) и соответствующим устройством и затем, если связь установлена правильно, соответствующее слово данных является выходным. Такая операция достигается в режиме управления программой посредством последовательного вывода двух слов: слова-функции и слова-данных.

Предположим, что n -ная команда основной программы требует вывода одного слова данных. Соответствующая команда в основной программе будет устанавливать слова в ферритовой памяти содержащие

а) адрес слова-данных,

б) слово-функцию, т.е. номер (признак) соответствующего выходного устройства.

в) адрес команды возврата к основной программе, а затем адрес передачи управления команде OCP.

Первая команда OCP будет условной передачей к 0-регистру. Обычно 0-регистр будет свободен, $0_{45} = 0$ и слово-функция, содержащее закодированный логический номер требуемого внешнего устройства, будет послано в 0-регистр. Одновременно сигнал синхронизации посылается из вычислительной машины по соответствующей линии.

Внешнее устройство после получения сигнала синхронизации будет узнавать слово как слово-функцию посредством модели разряда. Оно будет производить раскодирование слова и устанавливать необходимую связь с устройством, физическое число которого соответствует логическому числу в адресной части слова-функции. Если только эта связь установлена правильно, то свободный сигнал будет посылаться обратно в 0-регистр, устанавливая таким образом $0_{45} = 0$.

Следующая команда OCP условно передается снова к 0-регистру. Если был произведен правильный выбор необходимого внешнего устройства, $0_{45} = 0$ и соответствующее слово-данные остается в 0-регистре до тех пор, пока внешнее устройство, которое было уже выбрано, не получит переданный сигнал синхронизации и не считает содержимое 0-регистра, оставляя его свободным. Не дожидаясь проверки правильности передачи слова-данных, OCP возвращает управление основной программе, которая продолжает работу.

Несомненно, может случиться, что либо данное внешнее устройство не отыскивается к тому времени, когда передается слово-данные, или требуется произвести передачу слова-данных до того, как было передано предыдущее. К тому же может оказаться, что какое-нибудь данное устройство не работает или что внешнее устройство блокирует любую передачу от 0-регистра. В любом случае $0_{45} = 1$, что узнает OCP.

Для того, чтобы принять соответствующие меры в только что указанных случаях,

необходима дополнительная выходная подпрограмма (O S P). O S P знает., сколько внешних узлов подключено к 0-регистру и помнит в качестве параметров программы их временные характеристики (например, максимальную частоту повторения).

Назначение OSP состоит в запоминании положения внешнего устройства и в способности узнавать, что кроется под условием $0_{45} = 1$ в любой данный момент. Кроме того, она может отличить полное несрабатывание передачи от 0-регистра и частичное несрабатывание в передаче данных или в выборе данного внешнего устройства. В последнем случае OSP будет запоминать, что соответствующие устройства не исправны, и будет уведомлять об этом оператора машины (или подавать какой- нибудь другой соответствующий сигнал). Эта программа будет освобождать 0-регистр так, чтобы не происходило блокировки передачи к другим устройствам. Если ввиду полного несрабатывания все передачи от 0-регистра не проходят, OSP будет уведомлять об этом оператора машины, который примет необходимые меры.

Обычно функция OSP состоит в том, чтобы информировать основную программу, что выход слова-данных на соответствующее устройство разрешен или не разрешен. Неважно, является ли это временным ограничением (ввиду того, что занято соответствующее устройство) или же постоянным (из-за несрабатывания устройства).

§ 3.3. Оптимизация передач выходных данных

Как было показано в § 3.2, для исполняемой программы на вычислительной машине важно знать временные характеристики внешнего устройства, к которому осуществляется выход. Это необходимо для того, чтобы информация выдавалась как можно скорее после того, как любое данное устройство оказывается в состоянии принимать дальнейшую информацию. Обычно это осуществляется программой наблюдения, которая знает время и которая проверяет выходную программу через определенные интервалы с целью определения, требуется ли какой-нибудь выход информации. К сожалению, вычислительная машина БЭСМ – 3М не имеет внутренних числовых часов и основная скорость машины слишком мала для того, чтобы произвести удобное прерывание существующих подпрограмм посредством программ наблюдения. Однако эту трудность можно преодолеть использованием системы прерывания. В тех случаях, когда выход на данное внешнее устройство должен быть осуществлен с минимальной задержкой, устройство может после получения слова-функции, послать в вычислительную машину сигнал, что оно готово к приему данных. Посылка сигнала осуществляет использованием внешней системы прерывания, описанной в § 2. Однако при таком решении было бы необходимо отличать обычное прерывание от прерывания, указывающего, что тот или иной выход свободен; это может быть достигнуто подходящим кодированием входного слова.

Другое решение состоит в автоматическом прерывании исполнения текущей программы, когда освобождается выходной регистр. Очевидно, это потребует дополнительной команды, и внутренняя логика машины должна бы отличать такое прерывание от внешнего путем передачи управления в адрес другой команды для двух случаев.

§ 4. Некоторые выводы

Следует сделать несколько дополнительных замечаний о только что описанной входной-выходной системе прерывания. Ясно, что, если написана тщательно разработанная программа управления, которая всегда знает состояние внешнего устройства и знает, имеется или нет требование прерывания, то она может объединить две отдельных входных-выходных системы в одну и использовать тот же самый регистр как для входа, так и для выхода. Однако этого можно достигнуть только посредством сложной системы приоритетов. Поэтому предлагается, чтобы две системы для входа и выхода, действительно, оставались независимыми и использовали свой собственные соответствующие регистры W и O. Такой выбор имел бы огромное преимущество, состоящее в том, что истинная программа разделения времени могла бы использоваться для одновременного ввода или вывода данных, когда исполняется основная программа.

И, наконец, следует серьезно рассмотреть возможность, которая состоит в том, что последовательность прерывания "жесткой части" аппаратуры (" hardware ") должна представлять собой выбор программы. Другими словами, последовательность операций, в которых сохраняются C, 1 и ω -регистры, и особый адрес записываются в C-регистр посредством особой команды.

Например, может быть следующая команда:

Сохраняется содержимое C,-1- и ω -регистров в S -регистре, загружается C-регистр из A_1, 1-регистр из A_2 и возобновляется исполнение.

Такая команда наряду с другой командой, предложенной в § 2.2, которая восстанавливает C,-1 – и ω -регистры из S -регистра и возобновляет исполнение, обеспечила бы эффективный путь присоединения к подпрограммам с двигающимися (" relocatable ") адресами. Это, в свою очередь, значительно упростило бы запись общей трансляционной (" assembly ") и загрузочной (" loader ") программ.

Однако огромное дополнительное преимущество получилось бы здесь, если бы вместо отдельного S -регистра могло бы быть использовано определенное расположение ферритовой памяти, скажем A_1.

Единственная дополнительная команда, которая объединяла бы функции команды 3 в § 2.2 и вышеуказанной команды, состояла бы в следующем:

"Принять слово с адресом A_1 и обменять A_1 на содержимое C-регистра, A_2 – на содержимое 1-регистра и ω – на самый старший разряд A_3. Затем возобновить выполнение".

Выполнение вышеопределенной программы в режиме прерывания происходило бы автоматически, и A_1 представляло бы собой особый адрес.

В заключение выражаю благодарность В. Морозу, Ю. Каржавину, В. Федорину, В. Шигаеву за полезные обсуждения, а также сотруднику издательского отдела Л. Смирновой за неоценимую помощь в составлении русского текста этой статьи.

Рукопись поступила в издательский отдел
20 июля 1965 г.

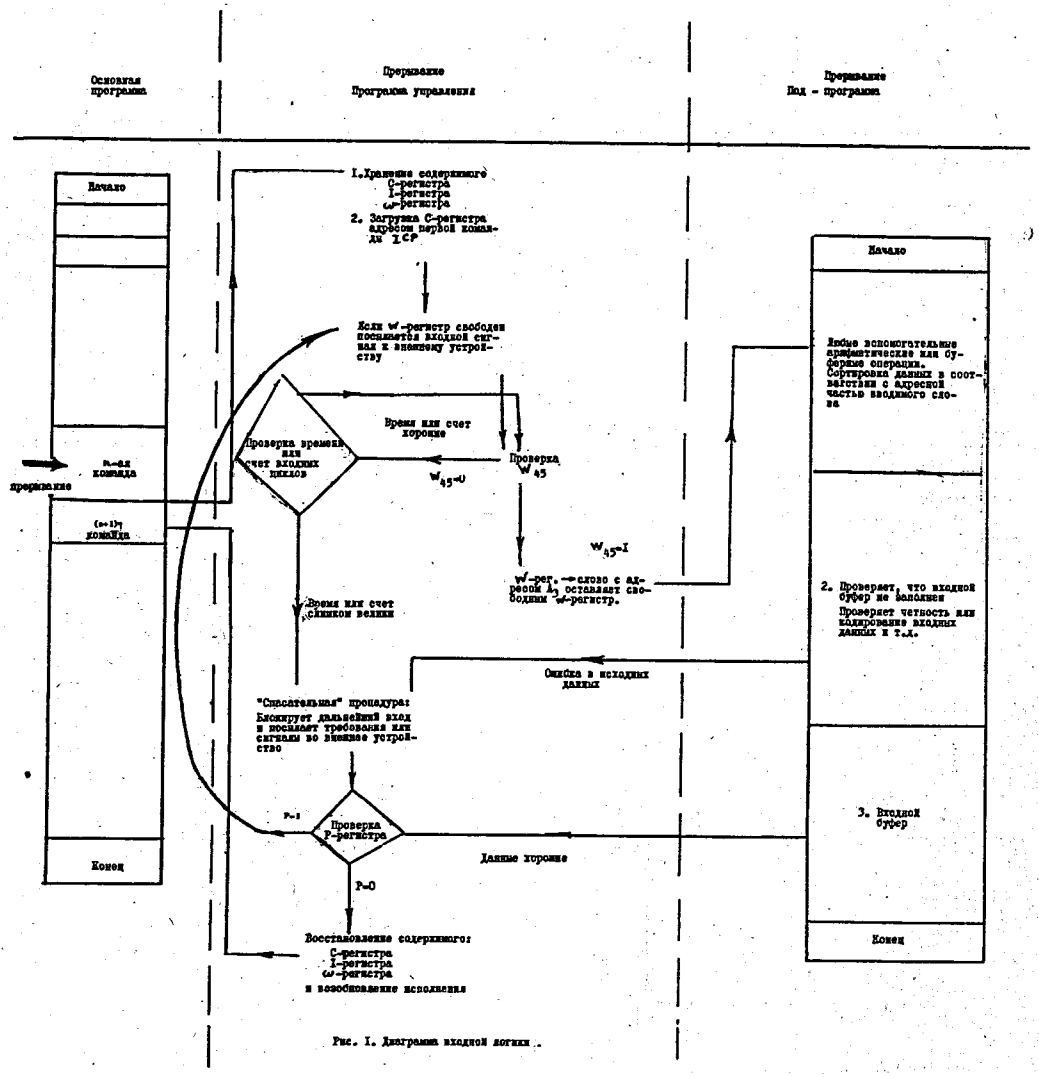

Рис. I. Диаграмма входной логики.

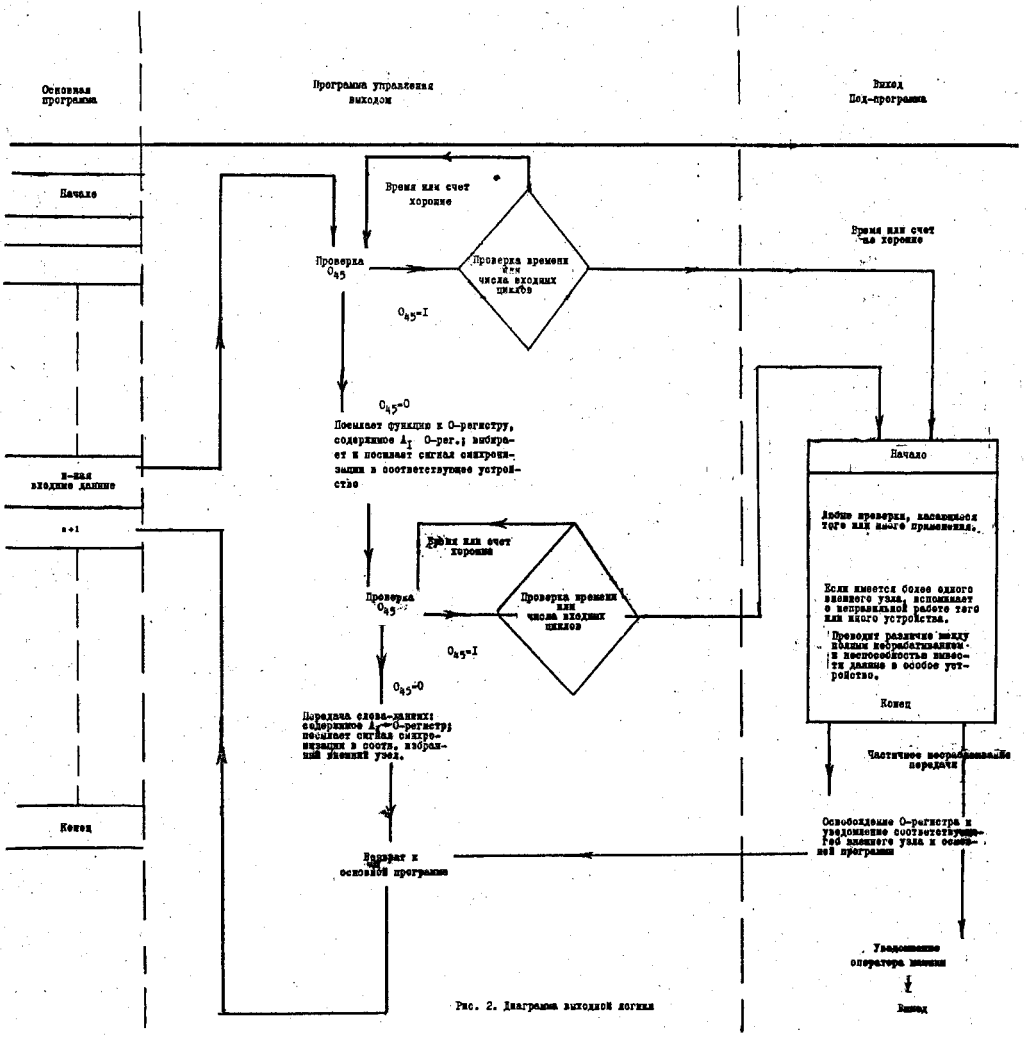

Рис. 2. Диаграмма выходной логики