ЛЕКЦИИ
ДЛЯ МОЛОДЫХ
УЧЕНЫХ

Н.Ю.ШИРИКОВА

Начинающим работать на ЭВМ CDC-6500

ДУБНА

ОБЪЕДИНЕННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИССЛЕДОВАНИЙ

Р11 - 11739

Н.Ю.Ширикова

НАЧИНАЮЩИМ РАБОТАТЬ НА ЭВМ CDC-6500

Дубна 1978

Ширикова Н.Ю.

Р11 - 11739

Начинающим работать на ЭВМ CDC-6500

Лекции предназначены для начинающих работать на ЭВМ CDC-6500 и знакомят с основными характеристиками машины и ее операционной системы.

В первой лекции дается краткое описание установки CDC-6500 и ее оборудования в ОИЯИ, объясняется работа операционной системы.

Во второй лекции приводятся основные управляющие карты системы, при помощи которых происходит общение пользователя с системой.

В третьей лекции описывается дополнительная возможность операционной системы для хранения информации в виде перманентных файлов.

Четвертая лекция посвящена использованию редактирующей программы для создания и корректирования библиотеки программ.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1978

Данный курс из четырех лекций был прочитан в феврале-марте 1977 года для сотрудников института, незнакомых с машиной CDC-6500 и ее операционной системой. Автор не ставил своей целью дать полное описание машины и операционной системы, а познакомил слушателей только с основными понятиями. Для более детального изучения операционной системы рекомендуем обращаться к следующим руководствам:

1. NOS/BE 1 reference manual (№ 60493800),
2. NOS/BE 1 user's quide (№ 60494000)
3. UPDATE reference manual (№ 60449900),
4. FORTRAN extended version 4 reference manual (№ 60497800)

Эти руководства составлены как справочники и рассчитаны на то, что пользователи в общих чертах знакомы с работой машины и операционной системы. Для начального изучения они довольно трудоемки.

Автор выражает признательность слушателям лекций за внимание; В.П.Ширикову, Л.А.Кулюкиной, Т.И.Забой, И.И.Шелонцеву, прочитавшим работу в рукописи и сделавшим критические замечания; Б.В.Феоктистову за настойчивую рекомендацию издать лекции; группе Н.А.Буздавиной за помощь в подготовке лекций к изданию.

СОДЕРЖАНИЕ

Основные характеристики CDC-6500

(1 ЛЕКЦИЯ)

Общая характеристика CDC-6500

Вычислительные машины серии 6000 являются мощными вычислительными комплексами общего назначения, сделанными на транзисторах. Они имеют один или два центральных процессора (СР), от семи до десяти периферийных вычислительных машин (РР) и от девяти до двадцати каналов, к которым присоединены внешние устройства. Каждая периферийная вычислительная машина имеет отдельную память и может выполнять программы независимо от других периферийных вычислительных машин и от центрального процессора.

Большая центральная память является общей и для периферийных машин и для центральных процессоров. На рис.I представлена блок-схема установки ОИЯИ CDC-6500 и ее оборудования в ОИЯИ. Центральная память (СМ) имеет I3I072 60-разрядных слов (400000_8 слов). Комплекс включает в себя 2 центральных процессора, I0 периферийных машин и 9 каналов, к которым присоединены:

3 печатающих устройства,
I выходной перфоратор,
2 читающих устройства,
2 семидорожечных магнитофона типа 667,
6 девятидорожечных магнитофонов типа 669,
6 дисководов для съемных дисков типа 844 (II млн. слов на каждом диске),
6 дисководов для съемных дисков типа 84I (3,5 млн. слов на каждом диске),
2 индивидуальных телетайпа типа TTY-32,
8 индивидуальных терминалов-дисплеев типа 40I2T,
2 индивидуальных терминала-дисплея типа 40I4T,
2 станции ввода-вывода (в ЛВЭ и ЛНФ).

Периферийные машины производят управление системой, ввод-вывод информации; например, из канала, подсоединенного к читающему устройству, информация принимается периферийной машиной в собственную память, а затем из этой памяти передается либо в центральную память, либо через другой канал на диск. Центральные процессоры производят действия над числами, хранящимися в центральной памяти.

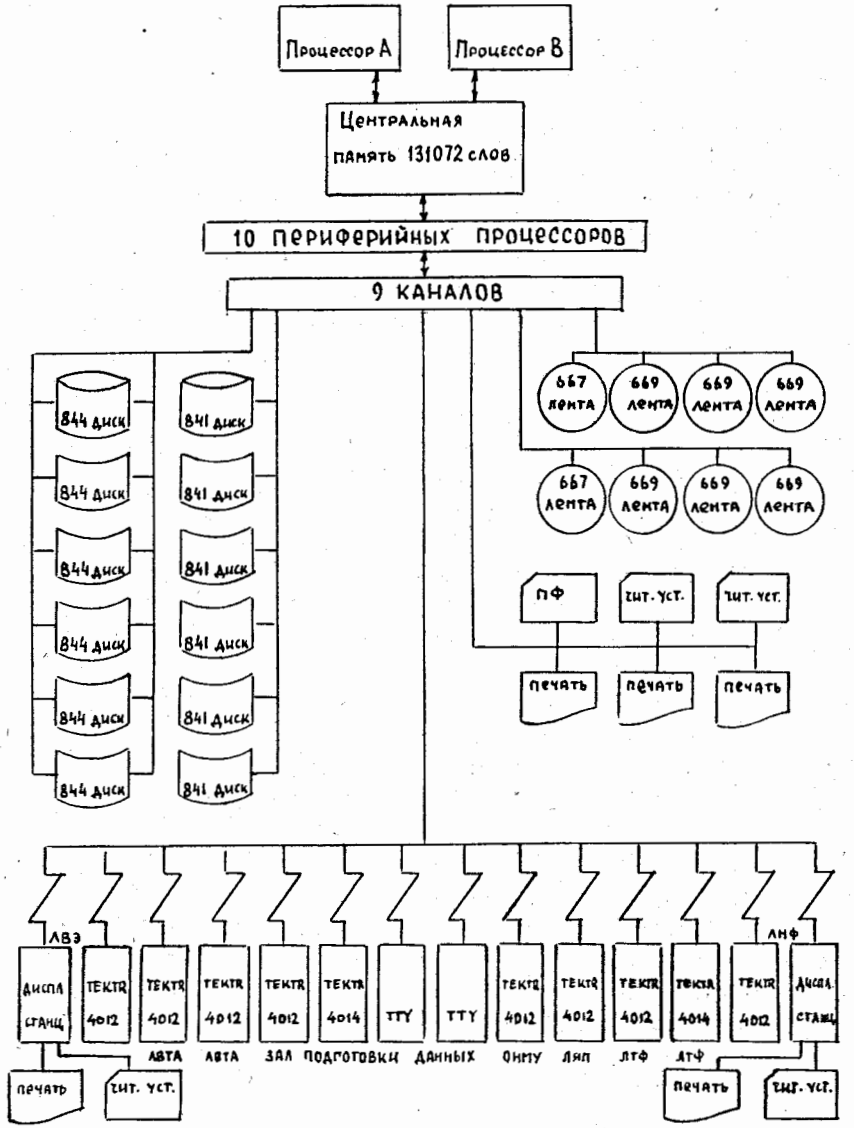
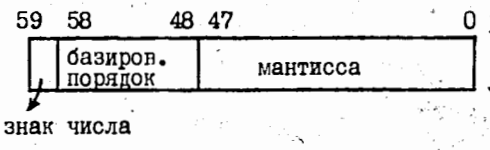


Рис.I. Установка ОИЯИ CDC-6500 и ее оборудование.

Представление чисел в памяти

Числа в памяти имеют различное представление в зависимости от типа числа (вещественное, целое, буквенно-цифровое, с двойной точностью).

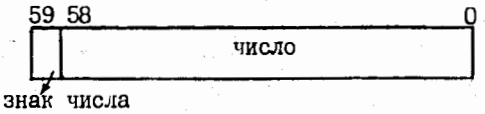
Вещественное число занимает одно слово (ячейку памяти) и имеет следующее представление:



Например,

I720 4000 0000 0000 0000 соответствует I.0
6057 3777 7777 7777 7777 соответствует -I.0
I723 5000 0000 0000 0000 соответствует I0.0
6054 2777 7777 7777 7777 соответствует -I0.0

Целое число занимает также одно полное слово:



Например,

0000 0000 0000 00000 0001 соответствует I
7777 7777 7777 7777 7776 соответствует -I

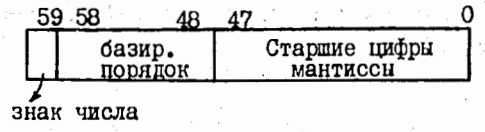
Однако в арифметических действиях умножения, деления используются только 48 разрядов.

Буквенно-цифровая информация занимает для каждого символа 6 разрядов и в слове может поместиться I0 символов. На CDC для представления буквенно-цифровой информации в памяти используется дисплейный код.

Например,

0I02 0304 0506 07I0 III2 соответствует ABCDEFGHIK

Вещественное число с двойной точностью представляется двумя словами памяти:



Среди вещественных чисел есть два особых "числа":

3777 XXXX XXXX XXXX XXXX соответствует + бесконечность
(+ infinite)

4000 XXXX XXXX XXXX XXXX соответствует - бесконечность

и

I777 XXXX XXXX XXXX XXXX соответствует + неопределенность

6000 XXXX XXXX XXXX XXXX соответствует - неопределенность,

где XX....X любые восьмеричные числа.

Бесконечный результат получается, например, при делении отличного от нуля числа на 0, т.е. $\infty = \frac{\text{число}}{0}$, неопределенность получается при делении 0 на 0.

Многие подпрограммы выдают неопределенный результат вычислений при неверно заданных значениях аргументов. Так, например, при попытке вычислить корень квадратный из отрицательного числа подпрограмма SQRT в качестве результата выдает неопределенное число. Однако всякие попытки произвести вычисления с этими двумя числами (неопределенным и бесконечным) приводят к прекращению счета задачи с выдачей диагностики:

ERROR MODE =m. ADDRESS = адрес,

где m - число от 0 до 7 и указан адрес команды, при выполнении которой были обнаружены запрещенные числа.

MODE=00 означает, что центральный процессор не распознает содержимое слова как набор понимаемых им команд.

MODE=02 означает, что была попытка произвести арифметическое действие с бесконечностью.

MODE=04 означает, что была попытка произвести арифметическое действие с неопределенным числом.

Число m , большее 4, надо рассматривать как сумму I,2,4, например, MODE = 06 соответствует тому, что прекращение счета произошло по двум причинам:

MODE=02 и MODE=04 .

Регистры и команды центрального процессора

Каждый центральный процессор имеет следующие регистры:
P регистр (I8 разрядов) для хранения адреса выполняемой команды,
8 адресных I8-разрядных регистров: A0, AI,....A7,
8 индексных I8-разрядных регистров: B0,BI,..., B7,
8 операндовых 60-разрядных регистров: X0,XI,...,X7.

На адресных регистрах A_i хранятся адреса слов, из которых выбирается содержимое или в которые заносится новое содержимое из операндовых регистров X_i .

Если на регистр A_i ($1 \le i \le 5$) заносится адрес слова, то содержимое из памяти автоматически заносится соответственно на регистр X_i ($1 \le i \le 5$).

Если на регистр A_i ($i = 7,6$) заносится адрес слова, то содержимое соответствующего регистра X_i ($i = 7,6$) автоматически записывается по указанному адресу в память.

Команды для CDC-6500 бывают двух видов: короткие (I5 разрядов) и длинные (30 разрядов). Короткие команды - команды действий с регистрами:

6 разрядов	3 разряда	3 разряда	3 разряда
код операции	номер регистра результата	номер регистра I-го операнда	номер регистра 2-го операнда

Например, команда 40ijk производит умножение содержимого регистра Xj на содержимое регистра Xk и засылает результат умножения в Xi , т.е. Xi=Xj*Xk , где i,j,k любые числа от 0 до 7. Команда 44ijk производит деление содержимого регистра Xj на содержимое регистра Xk и засылает результат деления в Xi , т.е. Xi=Xj/Xk.

Только команды с кодами 50-57 дают доступ к памяти. Например, выбор содержимого из ячейки I2000 на XI можно выполнить по команде 5II00I2000. Эта команда относится к классу длинных команд, которые имеют вид

6 разр.	3 разр.	3 разр.	I8 разрядов
код операции	номер регистра результата	номер индексн. регистра	операнд или адрес операнда

или

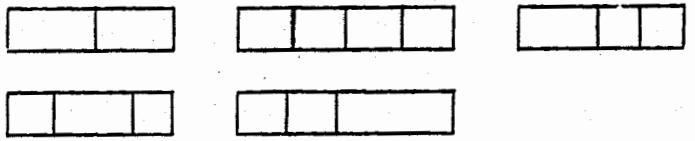
9 разрядов	3 разр.	I8 разрядов
код операции	номер регистра I-го операнда	операнд или адрес операнда

Так, например, обращение к подпрограммам, находящимся в памяти, происходит по команде 0I00 <начало прогр.> . Транслятор FTN добавляет в команду обращения восьмеричный номер строки фортранного текста, где было обращение, и указывает, из какой программы было обращение.

Пример команды обращения к подпрограмме, полученной после трансляции в FTN ,

```
0I000 47 I60 0I43 037302
начало
вызываемой
программы
                    в этой ячейке
                    название вызывающей
                    программы в диспл.
                    коде
99 строка
текста
```

Таким образом, в слове памяти может находиться от 2 до 4 команд:



Например, в двух ячейках памяти расположены коды:

5II000I0005I20002000
407I25I7000300046000

Они осуществляют умножение двух чисел из ячеек I000 и 2000 и засылку результата в ячейку 3000. Так как выдача при принудительном прекращении счета состоит из печати содержимого всех регистров, ста начальных слов задачи и ± I00 слов от того адреса, где произошло прекращение счета, в виде чисел и команд так, как они представлены в машине, то, пользуясь только указанными выше сведениями о машине, нетрудно определить причину прекращения счета. Глядя на выданные коды с диагностикой MODE=00 , часто без особого труда можно, например, увидеть, что на месте программы оказались

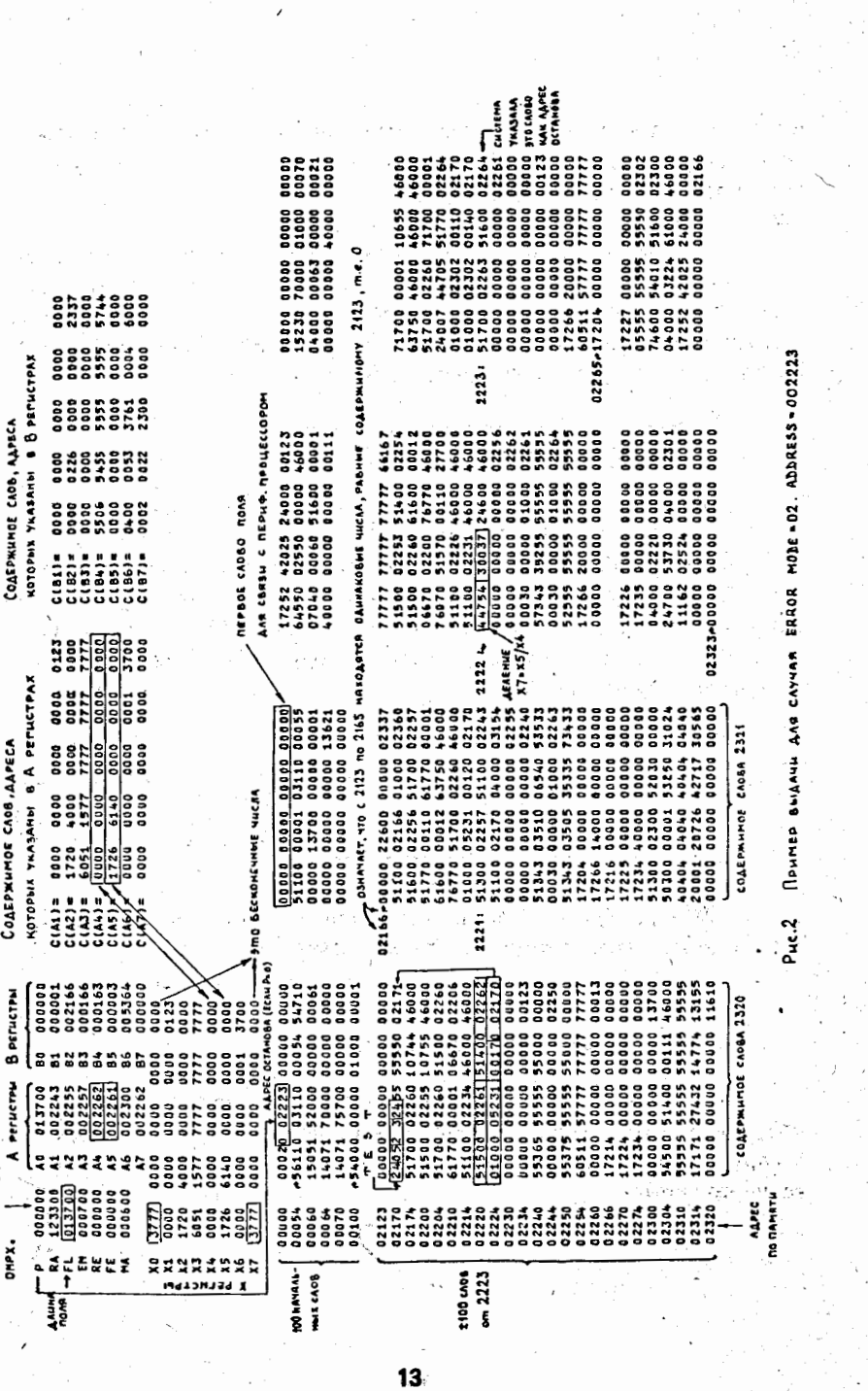
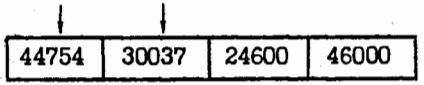


Рис.2 Пример выдачи для случая ERROR MODE=02. ADDRESS=002223

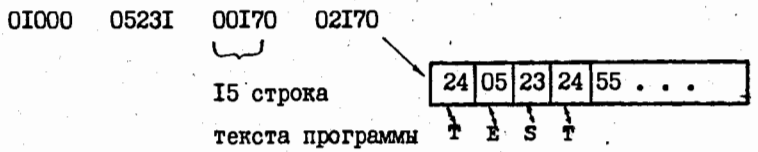
числа, т.е. из-за ошибки в программе сама программа была испорчена. На рис.2 производится выдача содержимого всех регистров при принудительном прекращении счета задачи

ERROR MODE = 02. ADDRESS 002223 .

Из приведенного примера видно, что при счете задачи было получено бесконечное число на регистре X7, с которым потом было произведено действие, т.к. бесконечное число появилось на регистре X0. Эти действия были произведены командами из ячейки 2222 ранее, чем указано в адресе 2223 прекращения счета)



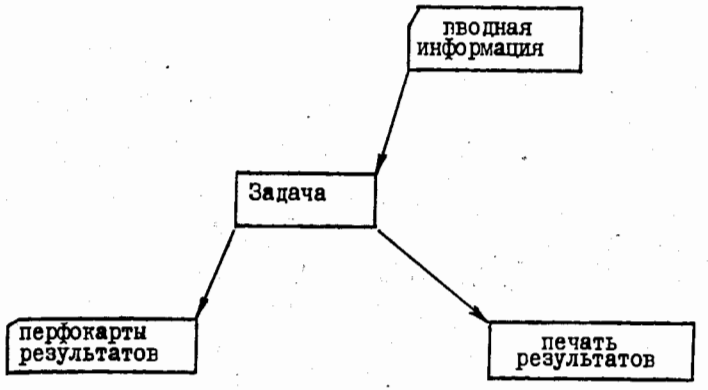
Видно, что на X7 посылался результат деления X5 на X4. Выборка на эти регистры была произведена в ячейке 2220 и осуществлялась из слов 2261 и 2262 соответственно. Видно, что в слове 2262 находится 0, т.е. прекращение счета произошло при попытке произвести действия с бесконечным числом, которое было получено от деления на 0. Близко от этих команд находится обращение к какой-то подпрограмме в ячейке 2224:



Таким образом, приблизительно прекращение счета произошло до I5 строки фортранного текста программы TEST .

Прохождение задачи в машине

Прежде чем пояснить работу операционной системы на машине CDC, рассмотрим пример простой задачи, которая читает информацию с карт, производит вычисления и выдает результаты на печать и на перфокарты:



Так как обычно ввод информации в задаче предусматривается частями, т.е. заказывается ввод нескольких перфокарт, затем счет, снова ввод и т.д., причем, по времени могут быть большие промежутки между вводами, то нерационально приписывать устройство ввода одной задаче на все время счета. Для увеличения скорости счета более разумно операционной системе всю вводную информацию ввести заранее как одно целое и записать на диск, чтобы потом при обращении к оператору чтения информации с карт в задаче информация бралась с диска, как с более быстро работающего устройства. Аналогично и информация, предназначенная для выдачи на печать и перфокарты, при счете тоже может писаться на диск с тем, чтобы после конца счета всю накопленную информацию для печати и перфорации выдать целиком на печатающем устройстве и перфораторе. Прохождение задачи показано на рис. 3.

На рис. 3 изображено для простоты три диска, но в действительности это может быть один и тот же диск. Таким образом, даже для такой обыкновенной задачи на диске будет храниться три вида информации: вводная информация, информация для печати, информация для перфорации. Причем два последних набора информации после окончания счета надо выдать на разные устройства. В этом случае операционной системе надо различать виды информации и знать, на какие устройства их потом выдавать. Поэтому операционная система CDC рассматривает любую информацию как часть или целое определенного вида информации и использует для этого понятие файла. Таким образом, файл - это информация одного вида (например, информация

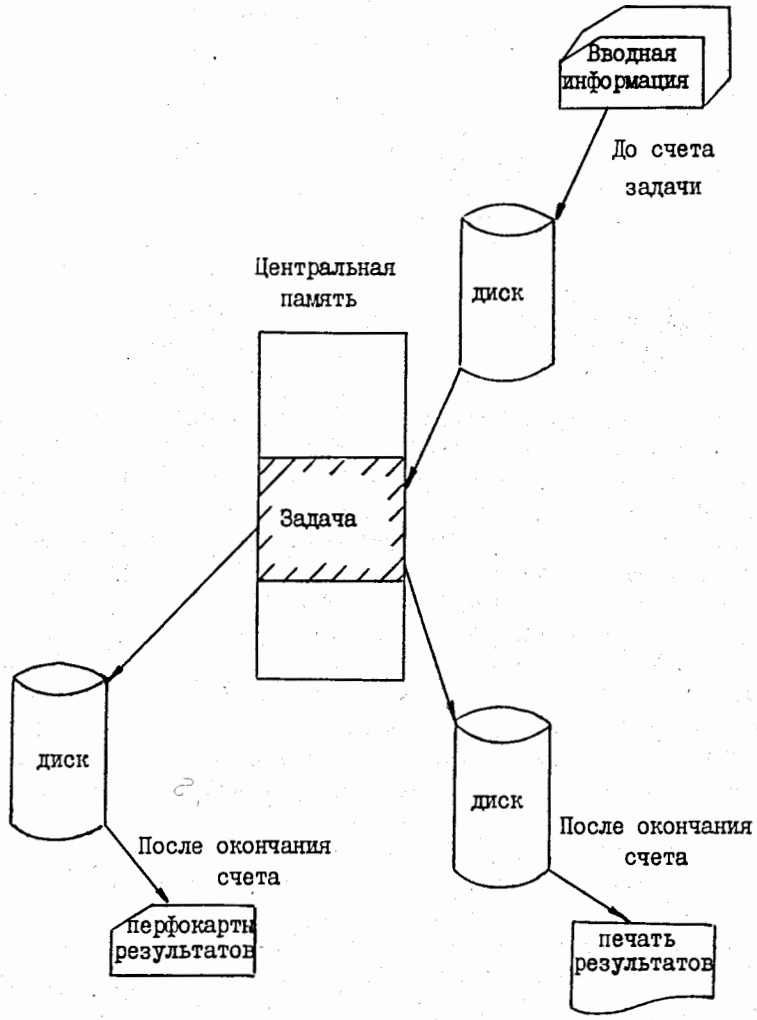


Рис.3. Выполнение счета на машине.

печати), которую можно рассматривать как единое целое или как объединение нескольких частей (рекордов) и которая приписывается какому-то устройству. Для различия файлов им присваиваются имена, состоящие из I – 7 символов, начинающиеся с буквы. Система предлагает использовать в задачах следующие имена файлов, которые она сама закрепляет за различными устройствами:

INPUT – файл, информация которого вводится с читающего устройства,

OUTPUT – файл, информация которого выводится на печатающее устройство,

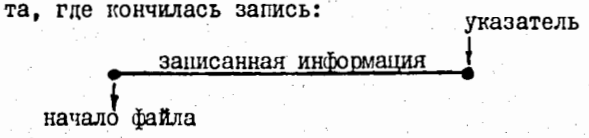
PUNCH – файл, информация которого выводится на перфоратор в буквенно-цифровом виде,

PUNCHB – файл, информация которого выводится на перфоратор в двоичном виде,

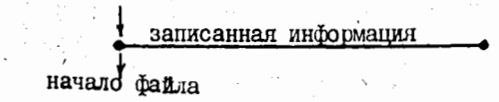
LGO – файл, информация которого записывается на диск и представляет собой программу, полученную после трансляции с алгоритмического языка.

Пользователь не может использовать имя ZZZZZ для своих файлов, это имя резервировано для системы.

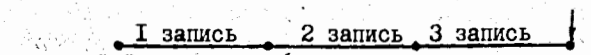
В приведенном на рис.3 примере задача использовала информацию из трех файлов: INPUT , OUTPUT и PUNCH . Каждая задача обязана сообщать системе, с какими файлами она будет работать. Действия, которые система может совершать с файлами, напоминают действия с лентой бытового магнитофона. Если мы записали на файл, то указатель положения на файле будет находиться после того места, где кончилась запись:



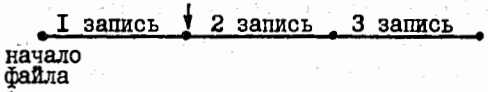
Если после записи попросить систему вернуться к началу файла, то система просто передвинет указатель в начало файла



Если на файле было несколько записей (рекордов),



то можно вернуться в файле, например, на 2 записи назад:



При действиях с файлами пользователь должен учитывать, что находится на файле и как расположен указатель положения. Однако и выделение диска как промежуточного устройства для увеличения скорости прохождения задачи может оказаться недостаточным. Например, если задача печатает через некоторые промежутки времени по одному числу, то неразумно обращаться к диску для записи каждого отдельного числа. Операционная система заводит для каждого файла в поле, отведенном в центральной памяти для задачи, определенное место (буфер), где она производит накопление информации, и только после заполнения буфера производится перепись на диск или ленту. Вместе с буфером отводится место для таблиц FIT , FET , которые ведут, в частности, учет,-сколько было накоплено чисел в буфере. Свои запросы на работу с внешним устройством задача производит, помещая в первую ячейку поля задачи специального вида коды, начинающиеся с трех буквенно-цифровых символов. В случае, если в первом слове поля появляется непонятный системе код, то выдается диагностика

PP CALL ERROR,

счет задачи прекращается и выдается содержимое всех регистров и 100 начальных слов поля.

Таким образом, когда задача находится в центральной памяти, то ей выделяется часть памяти (FL), в которой будет размещаться программа, результаты, буфера файлов, таблицы FIT , FET для каждого файла и некоторое число слов, находящихся в начале поля, для овязи с системой. Система сообщает пользователю, какой длины кусок памяти (поле задачи) она использовала для счета его задачи. В ОИЯИ установлена максимально допустимая длина поля задачи из 140000_8 слов ($\approx 50000_{10}$ слов). При попытке записи или чтения результатов в слова, расположенные вне поля задачи, происходит прекращение счета с выдачей диагностики:

MODE = 01.ADDRESS = адрес

Неправильная же работа программы, которая приводит к порче содержимого слов памяти внутри поля задачи, может и не быть обнаружена системой, однако может привести,например, к потере инфор-

мации какого-либо файла, если были испорчены его буфер или таблицы FIT, FET.

В общих чертах прохождение задач в машине выглядит следующим образом. Читающие устройства вводят колоды карт каждой задачи, и система записывает введенную информацию на диск, образуя очередь из введенных задач. По определенным правилам каждой задаче приписывается приоритет . Чем меньше задача просит от системы (память, время и т.д.), тем выше приоритет. Задачи с большими приоритетами переводятся в очередь счета. В центральной памяти выделяются поля для задач с наибольшим приоритетом. Если задача попала в центральную память и есть свободный процессор (A или B), то ей отдается центральный процессор на какой-то промежуток времени, при этом происходит снова пересчет приоритета. Если в указанный промежуток времени задаче потребуется выполнить что-то с помощью периферийного процессора, например, запись на диск, то центральный процессор у задачи отбирается,и она переводится в очередь заданий на выполнение в периферийном процессоре. В этом случае задача снова получит центральный процессор после выполнения действий в периферийном процессоре и при наличии высокого приоритета. Если же за выделенный промежуток времени работы в центральном процессоре таких обращений к периферии не было, то по истечении времени центральный процессор тоже отбирается для счета другой задачи с высшим приоритетом,и задача может быть возвращена в очередь счета. После окончания счета задачи пользователю сообщается,какое количество времени центральных процессоров A и B, периферийных процессоров было истрачено.

Управляющие карты системы NOS/BE 1

(2 ЛЕКЦИЯ)

Операционная система NOS/BE 1 обеспечивает мультипрограммную работу машины CDC-6500; заведение и сохранение перманентных файлов (т.е. файлов, которые после конца счета сохраняются в машине); протоколирует все действия операторов, требований, получаемых от задач, все возникающие неполадки в работе внешних устройств. В состав операционной системы, поставленной для ОИЯИ, входят:

трансляторы FTN v 4 (с языка Фортран),
ALGOL (с языка Алгол),
DDL ,
COMPASS -автокод,
SIMULA (с языка Симула),
FORM,
SORT-MERGE,
BASIC,
SYMPL,
QU,
COBOL v 4 (с языка Кобол, версия 4),
COBOL v 5 (с языка Кобол, версия 5),
PASCAL (с языка ПАСКАЛЬ),
LISP (с языка ЛИСП),

служебные программы, дающие пользователю дополнительные возможности сервиса,

редакторы текста
UPDATE,
EDIT (из системы INTERCOM , обслуживающей терминалы),

библиотеки программ общего назначения.

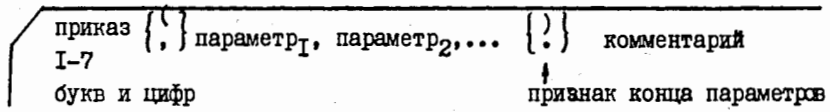
Система расположена на 4 дисках типа 844 и 3 дисках типа 841 (сюда включено и место на дисках, отводимое для хранения перманентных файлов), в центральной памяти машины для работы системы отведено от 40000_8 до 50000_8 слов.

Общение пользователя с операционной системой происходит либо при помощи управляющих карт, либо при помощи управляющих приказов, посылаемых с терминала. В лекции разбираются управляющие карты.

Каждая управляющая карта – это некоторый приказ операционной системе на выполнение одного или нескольких действий, которые задаются с помощью параметров на управляющей карте.

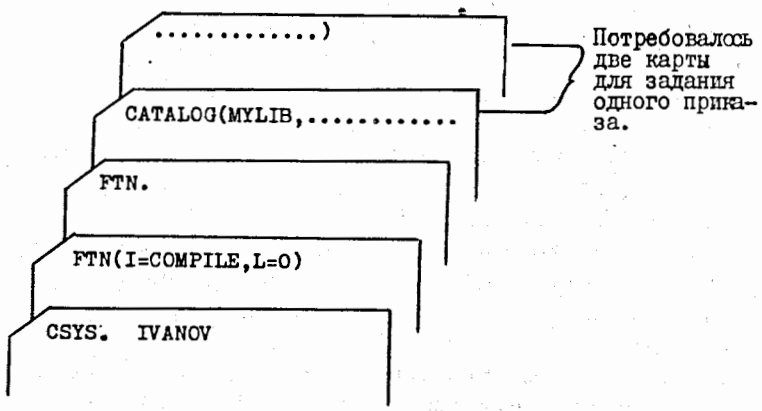
Чтобы исключить аварийные ситуации для системы, как, например, забывчивость пользователя(указать все требуемые параметры), операционная система для каждой управляющей карты имеет стандартный набор параметров, которые будут использованы в случае, если пользователь не указал какой-либо параметр. Стандартно подставляемые параметры дают еще одно преимущество пользователю: обращение к системе, не имеющее параметров, более короткое, требует меньше затрат при пробивке карты и, следовательно, уменьшает количество возможных ошибок.

Все приказы системе, как правило, располагаются на одной карте, пробиваются с первой колонки и имеют вид:



В фигурных скобках { } указаны допустимые символы, которые можно использовать в качестве разделителей. Для большинства управляющих карт только некоторые параметры задаются в определенном порядке, порядок большинства параметров несущественен. Если все требуемые параметры не уместились на одной карте, то следующая карта будет являться продолжением (начинается продолжение с I колонки).

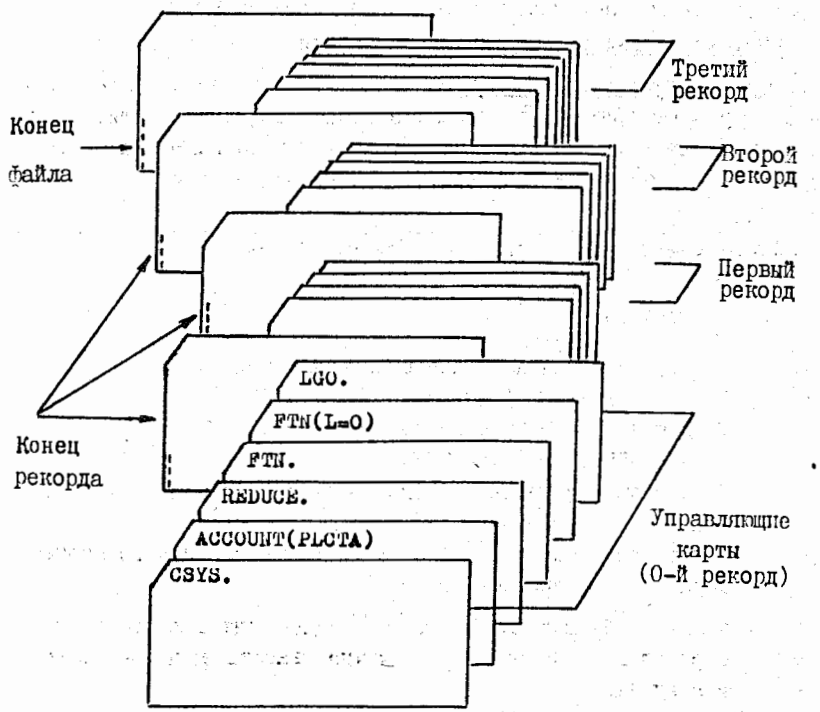
Примеры управляющих карт:



Если пользователь составляет колоду карт для счета, то его приказы системе должны быть подложены самыми первыми в колоде в том порядке, как они должны выполняться, и отделяться от остальных карт картой с пробивками 7/8/9 в первой колонке, называемой картой "Конец рекорда" (EOR).

Дальше в колоде могут встретиться карты 7/8/9, которые будут исполнять роль делителей входной информации на части (например, для отделения карт с текстом программы от карт с данными для счета). Вся колода должна оканчиваться также специальной картой с пробивками 6/7/8/9 в первой колонке, называемой картой конца файла (EOF). Так составленная колода будет представлять файл INPUT для задачи.

Пример подготовленной колоды.



Приказы образуют 0-й рекорд, остальные рекорды (если такие будут в колоде) будут пронумерованы системой по порядку 1,2, и т.д. Если при вводе карт какая-либо карта колоды не будет понята системой, то будет выдана диагностика с указанием - из какого рекорда карта и какой ее порядковый номер в этом рекорде.

Приблизительно все управляющие карты можно разделить на пять групп:

обращение к трансляторам, редакторам;
указания программе-загрузчику;
действия с файлами;
заказ внешних устройств (лент, частных дисков);
действия с перманентными файлами.

Если при исполнении система не понимает управляющую карту, то печатает

CONTROL CARD ERRORS

и распечатывает саму карту. Счет задачи прекращается.

Первые управляющие карты любой задачи

Самая первая управляющая карта в любой задаче сообщает системе шифр и фамилию пользователя, заказы времени и памяти, какого типа магнитофоны требуются задаче. Шифр и фамилию указывать обязательно. Остальные указываются при необходимости.

```
xxxxx, { Tt  } , CMFL,DYm,MTk,NTk.фамилия
       { TLm }
шифр
(5 символов)
```

Шифр - 5 символов, начинающихся с определенной буквы:
C (для ЛВТА), P (для ЛЯП), N (для ЛНФ),
T (для ЛТФ), H (для ЛВЭ), R (для ЛЯР),
S (для ОРБ), M (для ОНМУ).
Шифр можно получить у представителя, который имеется в каждой лаборатории.

Tt - заказ требуемых для счета t секунд центрального процессора (в восьмеричном виде)

TLm - заказ требуемых для счета m секунд центрального процессора (в десятичном виде). Можно указать заказ в часах и минутах.
Например, TL2H30M - заказ на 2 часа 30 минут,
TL15M5 - заказ на 15 минут и 5 секунд.

Буква H ставится после количества часов, а буква M - после числа минут.
Если заказ времени отсутствует, то система выделяет 400_8 с.

CMFL - заказ поля в центральной памяти из FL_8 слов. Например, CM50000. Можно и не указывать, сколько слов центральной памяти потребуется. Сама система выделит требуемое количество, но не более чем 140000_8 слов.

DYm - Y - параметр зависимости счета данной задачи от счета других задач пользователя,
m - значение счетчика зависимости.

MTk - задаче требуется k семидорожечных магнитофонов.
NTk - задаче требуется k девятидорожечных магнитофонов.

Примеры первой карты задачи:

```
CINT. IVANOVA
```

Задаче будет выделено 400_8 секунд для счета. Магнитофоны задаче не нужны.

```
CINT,TL15M,MT1.IVANOVA
```

Задаче требуется 15 мин для счета, один семидорожечный магнитофон.

Вторая управляющая карта задачи:

```
ACCOUNT(P)
```

P - признак лаборатории, также сообщаемый представителем лаборатории.

Третья управляющая карта задачи, не использующей транслятор с Алгола, -

```
REDUCE.
```

Указание системе использовать экономно память, выделяемую пользователю.

Остальные управляющие карты подкладываются в зависимости от желаний пользователя.

Действия с файлами

```
REWIND,имя файла 1,имя файла 2,...,имя файла n.
```

Возвращает в начальное положение указатель записи или чтения из перечисленных файлов.

```
RETURN,имя файла 1,имя файла 2,...,имя файла n.
```

Пользователь сообщает системе, что указанные файлы ему больше не нужны. Устройство, к которому был приписан файл, забирается у задачи. Если файлы перманентные, то они сохраняются.

```
COPY (имя файла 1,имя файла 2)
```

Копирует содержимое первого указанного файла 1, начиная с места, где находится указатель положения, на указанный файл 2.

Примеры.

```
REWIND,TAPE1, TAPE2.
```

```
RETURN,TAPE1,OLDPL.
```

Файлы TAPE1, OLDPL в данный момент больше не нужны задаче. В дальнейшем пользователь может снова использовать такие имена для других файлов.

```
COPY(A,DISK)
```

Копирует информацию с файла A на файл с именем DISK .

Загрузка программ

Указание на загрузку осуществляется следующими управляющими картами:

```
LOAD(имя файла)
```

Загрузить все программы, находящиеся на указанном файле, который предварительно возвращается в начало. Однако, если имя файла - INPUT , то загрузка начинается с того места, где находился указатель положения на файле.

```
имя файла.
```

Указанный файл вернуть в начало, если это не файл INPUT , загрузить все имеющиеся на нем программы в память и начать счет по загруженной программе.

Если пользователь производил загрузку с разных файлов, то переход на счет можно осуществить управляющей картой

```
EXECUTE.
```

Примеры.

```
LGO.
LOAD(INPUT)
```

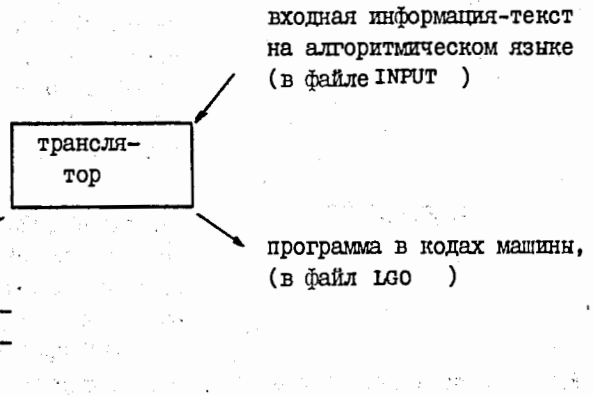
Эти две управляющие карты осуществляют загрузку программ с файла INPUT , затем с файла LGO и уход на счет.

Описанный выше пример загрузки можно осуществить и другим набором управляющих карт.

```
EXECUTE.
LOAD(LGO)
LOAD(INPUT)
```

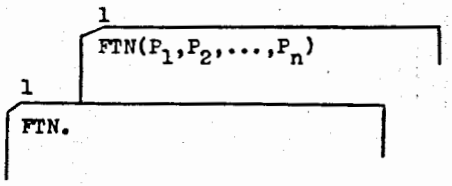
Обращение к транслятору FTN.

Рассмотрим работу транслятора с какого-нибудь алгоритмического языка:



Таким образом, для работы транслятора в упрощенном варианте требуется три файла: файл с входной информацией, файл с полученной после трансляции программой и файл с входным текстом и диагностикой работы транслятора. Обычно, если пользователь не указывает, какие это файлы, то система использует файлы: INPUT , LGO , OUTPUT .

Наиболее часто используемым транслятором в нашем институте является транслятор FTN с алгоритмического языка Фортран. Приведем параметры, задаваемые этому транслятору, которые являются самыми необходимыми для начинающего использовать транслятор FTN (всего FTN имеет более 30 параметров). Для каждого параметра в скобках ( ) будет указано стандартное значение параметра, подставляемое транслятором, если параметр не был задан. Управляющая карта вызова FTN может иметь следующий вид:



P_i – либо параметр,
либо параметр = значение.

Параметры разделяются запятыми и могут быть заданы в любом порядке. Нераспознаваемые параметры игнорируются.

ПАРАМЕТРЫ

B (сокращение от binary) — Получение транслированной программы (стандартно: B=LGO)

- **B** — Транслированную программу поместить в файл LGO .
- **B=имя файла** — Транслированную программу поместить в указанный файл.
- **B=0** — Не получать транслированную программу. В этом случае нельзя задавать параметр GO . Заметим, что указатель в файле с транслированной программой после работы транслятора не возвращается в начало файла.

GO (идти) — Автоматическое выполнение (стандартно: GO=0) (загрузка программы в память и счет).

- **GO** — Файл с транслированной программой загружается в память и начинается счет. Запись в файл происходила с того места, где находился указатель записи в файл.
- **GO=0** — Получить транслированную программу, но на счет не уходить.

I (сокращение от INPUT) — Входной текст берется с файла (стандартно I=INPUT)

- **I=имя файла** — Входной текст берется с указанного файла.
- **I** — Входной текст берется с файла COMPILE .

L (сокращение от Listable output (выводимое на печать)) — Выводимая транслятором информация (стандартно: L=OUTPUT)

L =имя файла — Транслятор выдает входной текст программы и диагностику своей работы на указанный файл.

L — Транслятор выдает входной текст и диагностику своей работы на файл OUTPUT .

L=0 — Транслятор выдаст на файл OUTPUT только диагностику о фатальных ошибках, которые были обнаружены во входном тексте программы.

OPT (сокращение от optimisation) — Оптимизация транслированной программы (стандартно: OPT=1).

OPT=0 — Транслированную программу не оптимизировать.

OPT=1 — Получить транслированную программу с уровнем оптимизации, равным 1.

OPT=2 — Получить транслированную программу с уровнем оптимизации, равным 2. Рекомендуется использовать OPT=2 только для уже отлаженной программы.

PD (сокращение от print density) — Плотность печати (стандартно PD=6)

PD=6 — Плотность 6 строк на дюйм.

PD=8 — Плотность 8 строк на дюйм.

PD — Подразумевает PD=8.

PL (сокращение от print limit) — Предел печати (стандартно: PL=5000)

PL=n — n – максимальное число выдаваемых на печать строк при счете программы: n ⩽ 9 999 999 999.

R (сокращение от reference (ссылка)) — Таблицы ссылок (стандартно: R=1)

R=0 — Не выдавать таблицы ссылок.

R=1 — Выдать таблицы ссылок в упрощенном варианте.

R=2 — Выдать таблицы ссылок в упрощенном варианте и ссылки по номерам строк входного текста.

R=3 — Выдать таблицы ссылок в полном виде.

R — Подразумевает R=2 .

TS (сокращение от time-sharing) — Работа транслятора в режиме разделения времени (Стандартно: TS=0)

Быстрый режим трансляции. Рекомендуется использовать только при отладке программы, т.к. в этом случае транслированная программа получается более длинной и требует большего количества счетного времени.

Q (сокращение от Quick) — Быстрый режим трансляции для проверки правильности программы. (Стандартно: Q=0).

Q=0 — Обычный режим трансляции.

Q — Быстрый режим трансляции без получения программы в кодах машины. Рекомендуется использовать этот режим при отладке программы.

Примеры обращений к транслятору FTN

```
1
FTN.
```

Вызывается транслятор FTN со стандартным набором значений для 33 параметров (для сокращения указываются только упомянутые выше параметры):

```
FTN (I=INPUT,B=LGO, GO=0,L=OUTPUT,OPT=1, R=1, Q=0)
```

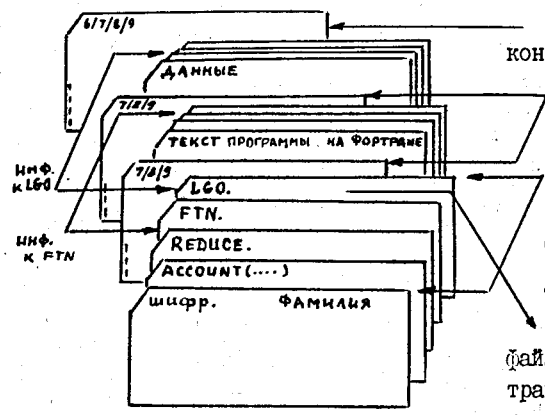
- Входная информация о программе берется с карт
- Транслированная программа пишется на файл LGO
- На счет программа не запускается
- Входная информация и диагностика будут выданы на печать
- Трансляция будет выполнена с режимом оптимизации полученной программы, равным 1
- Нормальный режим трансляции
- Будут выданы таблицы ссылок упрощенного вида на файл печати

```
1
FTN(B=PUNCHB)
```

Вызывается транслятор для записи полученной в кодах программы на файл PUNCHB, т.е. после счета задачи программа будет выдана в двоичном виде на карты. Остальные параметры – как в приведенном выше примере. Этот режим используется после окончания отладки с тем, чтобы считать, используя транслированную программу.

Примеры составления колоды карт для счета задач, написанных на Фортране с использованием транслятора FTN .

Пример I. Транслирование и выполнение программы, написанной на Фортране.



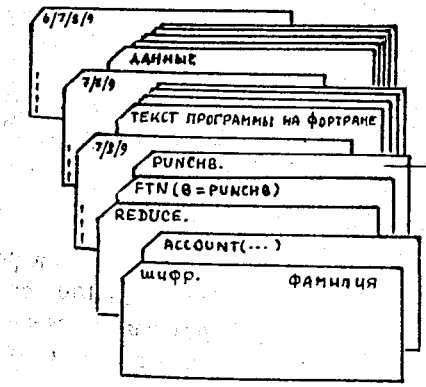
конец информации

разделители информации

эти управляющие карты образуют 0 рекорд
Текст образует I рекорд
Данные образуют 2 рекорд

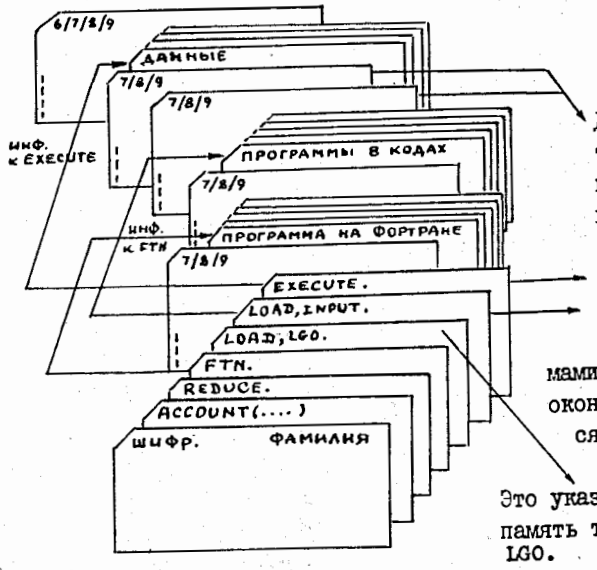
Это указание системе: файл LGO , на который транслятор FTN поместил программу, вернуть в начало, прочитать с него программу, загрузить ее в память и запустить на счет.

Пример 2. Транслирование, выдача транслированной программы на карты в двоичном виде и счет.



Это указание системе взять программу на счет с файла PUNCHB , т.к. FTN поместил транслированную программу на PUNCHB.

Пример 3. Трансляция части программы, написанной на Фортране, загрузка этой части с LGO , загрузка остальной части с карт, представляющих программу в кодах, из файла INPUT и счет.



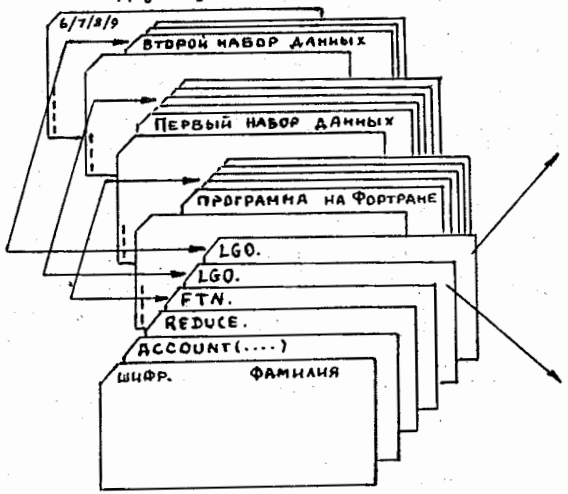
Две карты 7/8/9 требуются после карт с программой в кодах.

Уйти на счет

Это указание ввести карты с программами в кодах. Ввод будет окончен, когда встретятся две подряд идущие карты 7/8/9

Это указание загрузить в память то, что есть в файле LGO.

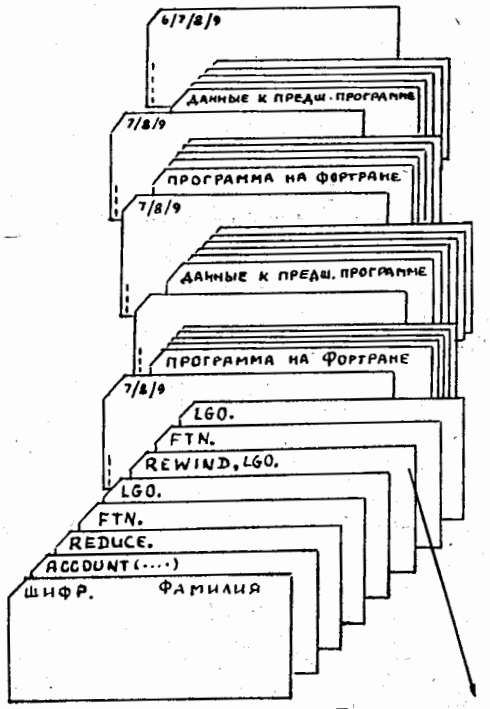
Пример 4. Трансляция программы, написанной на Фортране, и счет с двумя разными наборами данных.



По второй управляющей карте LGO снова загрузится программа и произойдет счет по этой программе.

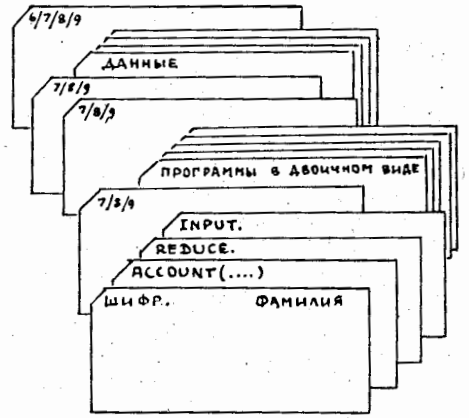
По первой управляющей карте LGO загрузится программа и произойдет счет.

Пример 5. Трансляция и выполнение двух программ, написанных на Фортране, каждая из которых имеет свой набор данных.

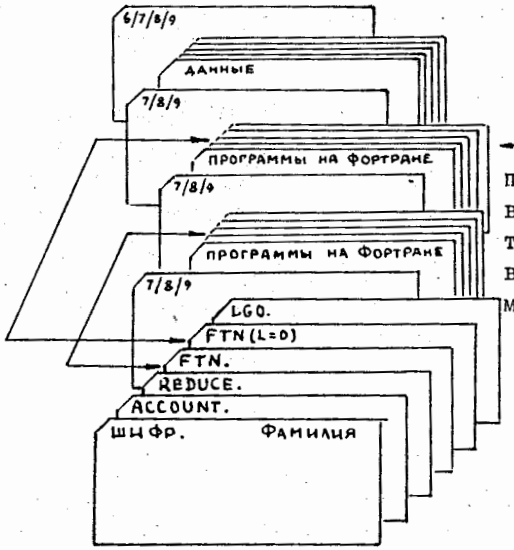


Такое указание системе вернуть файл LGO в начало необходимо по следующим причинам. После трансляции первой программы ее текст в кодах попадает на файл LGO . Когда LGO загружается, то загрузчик возвращает его в начало, загружает и оставляет указатель положения после конца первой программы. Если бы карты REWIND не было, то следующая трансляция поместила бы следующую программу дальше, после первой программы. Затем встретилась бы вторая операция загрузки, и загрузчик вернул бы LGO в начало, тем самым загрузил бы снова первую программу.

Пример 6. Счет по программе, выданной ранее в двоичном виде.



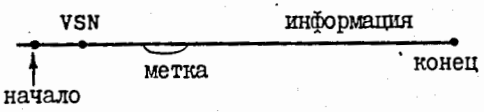
Пример 7. Трансляция с выдачей печати только для одной части программы.



←Текст этих программ не будет выдан на печать, т.к. транслятор вызывался в режиме L=0.

Заказ ленты для одного файла

Для записи или чтения файла на магнитной ленте надо сообщить системе информацию о том, какого типа магнитофон требуется, с какой плотностью записи пользователь предпочитает работать, где и как была сделана запись или как ее выполнить. Дело в том, что пока нет единого формата записи на ленту. Система NOS/BE 1 различает три вида лент: приготовленные самой системой (SI) ленты; ленты типа S (сокращение от strange-чужой),приготовленные другими системами;ленты типа L (сокращение от long-длинный), для которых длина записи определяется самой записью. Системные ленты (SI) и некоторые S - ленты могут иметь метку и номер ленты (VSN), которые записываются в начале ленты:



Поэтому система делит свои ленты на меченые и немеченые. Метка ленты - это дополнительная информация (например, дата, фамилия пользователя), которая обеспечивает пользователю дополнительную предосторожность при опознавании и использовании ленты. Номер ленты - VSN -информация, которая сообщается системой операторам для заблаговременной постановки на магнитофонах нужных лент для счета; тем самым уменьшается вероятность путаницы лент при счете. Заказ ленты можно осуществить двумя управляющими картами: REQUEST и LABEL . Приводятся самые необходимые параметры.

REQUEST(имя файла, {MT / NT}, {LO / HI / HY / PE / HD}, {S / L}, {E / N}, {RING / NO RING}, VSN=uuuuuu)

MT - обозначение семидорожечного магнитофона, на котором допустимы плотности:

LO - 200 бит на дюйм (только для чтения),

HI - 556 бит на дюйм,

HY - 800 бит на дюйм,

NT – обозначение девятидорожечного магнитофона, на котором допустимы плотности:
PE – 1600 бит на дюйм,
HD – 800 бит на дюйм.
При работе с девятидорожечным магнитофоном плотности указываются только при записи.

S – указывается для чтения ленты, приготовленной на другой машине другой операционной системой,

L – указывается для чтения ленты, приготовленной на другой машине, причем длина зоны записи определяется количеством записанных в ней чисел.
Для лент, приготовляемых самой системой NOS/BE , параметр (S или L) не указывается.

E – лента меченая ,

N – просьба записать на ленту "пустую" метку, т.е. метку без информации,

RING – на ленту вставлено специальное кольцо (возможна запись на такую ленту),

NORING – лента без кольца, т.е. может использоваться только в режиме чтения,

VSN – номер, присвоенный ленте, от 1 до 6 буквенно-цифровых символов. Этот номер можно узнать заранее у сотрудника ЛВТА, ответственного за ленты.

Управляющая карта LABEL выполняет функции REQUEST , но имеет еще дополнительные возможности:
писать метку,
читать и проверять метку.

Формат карты LABEL :

LABEL, имя файла , {W / R} , {RING / NORING} , D=d, F={S / L} , L=z, VSN=uuuuuu.

W – будет режим чтения и контролл метки,
R – режим записи новой метки,
d – плотность, {LO / HI / HY} для МТ ленты, {PE / HD} для NT ленты

F – тип записи на ленте: S или L . Указывается только для лент с других машин,
L – метка, которая записывается на ленту или с которой будет сравнена метка ленты: от 1 до 17 буквенно-цифровых символов.

Остальные параметры аналогичны параметрам на карте REQUEST.

Если пользователь считает одну и ту же задачу, но с лентами разных номеров, то лучше параметр VSN вынести на отдельную управляющую карту:

VSN(имя файла = $uuuuuu_1$=$uuuuuu_2$=...=$uuuuuu_n$)

$uuuuuu_1$,..,$uuuuuu_n$ – номера лент, с которых берется или пишется информация для указанного файла.

Эта карта ставится перед картой REQUEST или LABEL .

Так как CDC-6500 в ОИЯИ имеет 8 магнитофонов, а задач с лентами много, то более разумно в задаче после того, как прочитана информация с ленты, отказаться от магнитофона, если он данной задаче больше не нужен. Это выполняется с помощью управляющей карты RETURN . Однако, если задаче магнитофон нужен, а требуется только сменить ленту, то управляющая карта

UNLOAD,имя файла

разгрузит ленту, сообщит об этом системе, но устройство останется за задачей.

Пример I. Использование 2 семидорожечных магнитофонов для задачи, состоящей из двух различных программ, первая из которых читает информацию с двух лент с номерами II50 и II5I, а вторая читает информацию с ленты № II52.

```
шифр,MT2. IVANOV
ACCOUNT(...)
REDUCE.
FTN.
VSN(TAPE1=1150,TAPE2=1151,TAPE3=1152) ← Поставить ленты II50, II5I, II52
LABEL(TAPE1,R,D=HY,L=MYTAPE1,NORING) ← Меченая лента с меткой MYTAPE1 без кольца.
LABEL(TAPE2,R,D=HY,L=MYTAPE2,NORING) ← Меченая лента с меткой MYTAPE2 без кольца.

LGO.
RETURN(TAPE1)  ← возвращает магнитофон системе
UNLOAD(TAPE2)  ← разгружает ленту, но магнитофон будет снова использован
REWIND,LGO.
FTN.
LABEL(TAPE3,R,D=HY,L=MYTAPE3,NORING) ← Меченая лента с меткой MYTAPE3 без кольца

LGO.
7/8/9
    программа I
7/8/9
    данные к программе I
7/8/9
    программа 2
7/8/9
    данные к программе 2
6/7/8/9
```


Пример 2. Трансляция программы, написанной на Фортране, счет с чтением данных с немеченой ленты I300, записанной на семидорожечном магнитофоне с плотностью 800 бит на дюйм.

```
шифр, MT1. IVANOV
ACCOUNT(...)
REDUCE.
FTN.
VSN(TAPE1=1300)
REQUEST(TAPE1,MT,HY,NORING)
LGO.
7/8/9
    программа
7/8/9
    данные
6/7/8/9
```

Эта задача будет держать магнитофон до конца счета. Если информации на ленте немного, то лучше предварительно скопировать содержимое ленты на диск и пустить счет с чтением с диска:

```
шифр, MT1.IVANOV
ACCOUNT(...)
REDUCE.
FTN.
VSN(A=1300)
REQUEST(A,MT,HY,NORING)
COPYBF(A,TAPE1)  ← копирует с ленты на дисковый файл
RETURN,A.        ← отдает магнитофон
REWIND,TAPE1.
LGO.
7/8/9
    программа
7/8/9
    данные
6/7/8/9
```

Указания загрузчику программ

Системная программа-загрузчик производит загрузку программ с файла в память, осуществляет загрузку системных подпрограмм, вызываемых из программы, составляет таблицу загрузки (в полном или частичном виде) и предоставляет много других сервисных возможностей.

Указания загрузчику, которые надо задать непосредственно перед выполнением загрузки:

LDSET(режим$_I$, режим$_2$,...)

режим$_i$ имеет вид:

- режим$_i$
- режим$_i$ = параметр
- режим$_i$ = параметр$_I$/параметр$_2$/...

Возможные режимы:

PRESET=p		Заслать во все слова выделяемого поля число указанного вида P ,
где p -	NONE	Не делать засылку
	ZERO	Заслать нули
	ONES	Заслать числа вида 7777 7777 7777 7777 7777
	INDEF	Заслать числа вида 1777 0000 0000 0000 0000
	INF	Заслать числа вида 3777 0000 0000 0000 0000
MAP=p/имя файла или /имя файла или p		- Режим выдачи таблицы загрузки,

p - либо один символ, либо несколько символов, каждый из которых обеспечивает определенный вид выдачи.

- N - таблицу загрузки не выдавать.
- S - выдать статистику загрузки.
- B - выдать загрузку блоков.
- E - выдать входы каждой подпрограммы (ENTRY POINTS).
- X - выдать таблицу взаимных ссылок подпрограмм.

В качестве нескольких символов можно использовать любые наборы из S,B,E,X.

имя файла - имя файла, на который записывается таблица загрузки. Если имя не указывается, то подразумевается запись на файл OUTPUT..

Пример. Трансляция программы, написанной на Фортране, предварительная засылка чисел вида "бесконечность" перед счетом, выдача на печать таблицы загрузки, состоящей из входов подпрограммы и таблицы взаимных ссылок,и счет.

```
шифр.
ACCOUNT(...)
REDUCE.
FTN.
LDSET(PRESET=INF,MAP=EX)
LGO.
7/8/9
    программа на Фортране
7/8/9
    данные
6/7/8/9
```

Перманентные файлы

(3 ЛЕКЦИЯ)

Файлы, возникающие в любой задаче, по решению самого пользователя пишутся на какое-то внешнее устройство: магнитную ленту, частный диск пользователя, системный диск. На системный диск тем самым пишутся файлы от многих задач.

После счета каждой задачи операционная система анализирует файлы задачи и в зависимости от типа файла предпринимает определенные действия. Если файл находился на магнитной ленте, то магнитная лента сматывается, и оператору дается указание снять ленту. Если файл находился на системном диске, но представлял собой файл, информация которого должна быть отпечатана на печатающем устройстве, то информация печатается, а затем файл стирается. Если же пользователь использовал системный диск или частный диск вместо ленты, то после конца счета операционная система должна решать, а что делать с информацией файлов, записанных на этих дисках. Стирание всех таких файлов создавало бы некоторое неудобство для пользователя. Может же случиться, что информация какого-то файла понадобится пользователю снова через день, например. Тем самым система должна предусматривать возможность сохранения на некоторое время информации некоторых файлов, записанных на системный или частный диск. В операционной системе NOS/BE такая возможность есть. Файлы, информация которых сохраняется на системном диске или частном, называются перманентными. Пользователь в своей задаче сам решает, какие файлы нужно сохранить после конца счета, для этого он делает специальные указания системе.

Учитывая ограниченную емкость системных дисков, каждое подразделение может завести такое количество перманентных файлов, для которых суммарный размер занимаемого на дисках места не превышает установленного подразделению лимита. Поэтому каждый пользователь заведение перманентного файла должен согласовать с представителем своей лаборатории.

Каждому создаваемому в задаче перманентному файлу пользователь должен дать два имени : имя перманентного файла (от I - 40 символов из букв и цифр) и имя владельца этого перманентного файла (от I-9 символов из букв и цифр).

В ОИЯИ принята следующая договоренность: в имени перманентного файла обязательно должна быть фамилия пользователя и его рабочий телефон, например, PETROV62321 , а в качестве владельца указывается подразделение, в котором работает пользователь. Следует использовать следующие названия подразделений:
LCTA(ЛВТА),LNR(ЛЯР),LVE(ЛВЭ), ORB(ОРБ),LTPH(ЛТФ),
ONMU(ОНМУ),LNF(ЛНФ).

Можно хранить под одним именем перманентного файла до пяти разных файлов, которые будут называться циклами этого перманентного файла. Для различия циклов им нужно присвоить номера (в диапазоне от I до 999).

При создании перманентного файла можно предусмотреть, что в дальнейшем можно будет делать с этим файлом. Система допускает четыре возможных действия с перманентным файлом:

RD(READ) читать - Разрешается читать содержимое файла, копировать содержимое файла в другой файл;

MD(MODIFY) модифицировать - разрешается перезаписать содержимое файла или уничтожить часть файла;

EX(EXTEND) расширять - разрешается уничтожить часть файла или увеличить файл;

CN(CONTROL) - разрешается уничтожать файл, создавать новый цикл в добавление к уже существующим циклам.

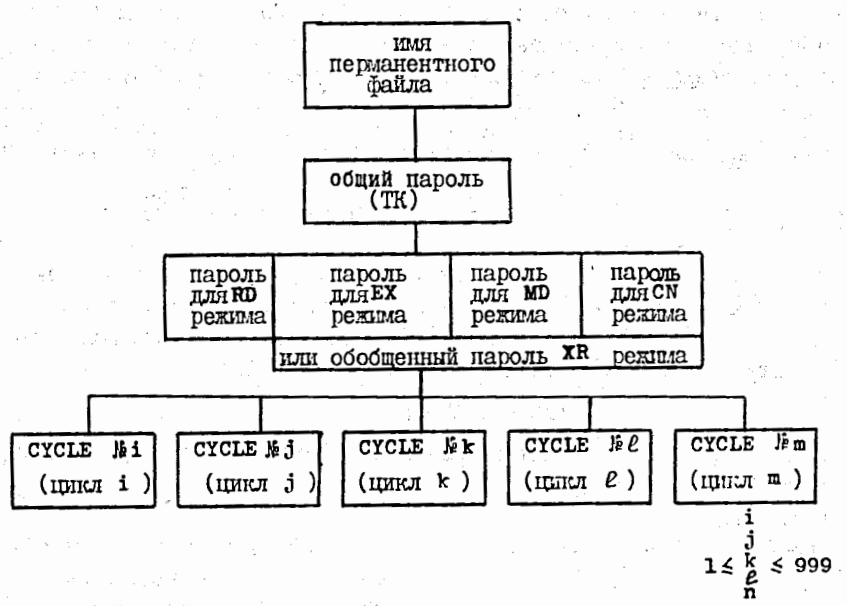
Для каждого действия можно ввести свой пароль (набор от I до 9 символов, состоящий из букв и цифр). Система при создании перманентного файла запомнит эти пароли. При дальнейшем требовании пользователя предоставить ему информацию перманентного файла, он должен сообщить системе пароли файла. Система сравнит указанные пароли с запомненными ею паролями и разрешит пользователю только такие действия с перманентными файлами, какие были гарантированы указанными пользователем паролями.

Для удобства пользователя вместо указателей трех действий (MD, EX, CN), которые частично дублируют друг друга, можно воспользоваться объединяющим указателем действий - XR, который автоматически гарантирует выполнение всех трех действий.

Если же пользователь при создании перманентного файла не завел никаких паролей, то право на все четыре действия с перманентным файлом предоставляется автоматически при любом обращении к перманентному файлу. Для начинающего впервые работать с перманентными файлами рекомендуется не указывать пароли, чтобы не путаться и не затруднять себе работу.

Система допускает еще дополнительный общий пароль (ТК), который как бы главенствует над остальными четырьмя паролями: этот пароль должен сообщаться при любом обращении к перманентному файлу, в то время как остальные пароли требуются в зависимости от того, что пользователь собирается делать с информацией перманентного файла.

Таким образом, перманентный файл выглядит следующим образом:



Действия с перманентными файлами можно разделить на пять стадий:

I) Начальное заведение перманентного файла.

2) Доступ к заведенному перманентному файлу, чтение информации или ее изменение.

3) Добавление нового файла (как цикла) к уже существующему перманентному файлу.

4) Изменение названия перманентного файла и отмена, замена или введение паролей.

5) Уничтожение перманентного файла.

Начальное заведение перманентного файла

Заведение перманентного файла осуществляется двумя управляющими картами. Сначала надо сообщить системе, что файл с указанным именем пользователя просит поместить на диск, отведенный для хранения перманентных файлов:

```
REQUEST,имя файла,*PF.
```

Эта карта ставится до создания файла. После того, как вся информация в файл записана, т.е. файл готов, надо дать указание системе сделать файл перманентным, употребив управляющую карту

```
CATALOG,имя файла,имя перм.ф.,ID=имя подразд,CY=№,CN=контр.,EX=расшир.,MD=модиф.,
RD=пароль чтения,TK=пароль общий.
```

Если пользователь указал номер цикла (№), то будет заведен цикл с этим номером. Если же пользователь не указал номер, то будет заведен цикл с номером I. При начальном заведении перманентного файла система выдает в протокол задачи:

INITIAL CATALOG.

В приведенных ниже примерах указывается только часть управляющих карт.

Пример I.

```
REQUEST,LGO,*PF.
FTN.
CATALOG(LGO,PERMANFILE,ID=LCTA)
```

файл LGO запоминается системой как перманентный файл PERMANFILE, принадлежащий ЛВТА. Пароли не указывались, т.е. в дальнейшем все четыре действия с файлом разрешены.

Пример 2.

```
REQUEST,LF,*PF.
      :
CATALOG(LF,PF,ID=LTPH,RD=X,CN=Y,MD=A,TK=C)
```

Заводится перманентный файл с именем PF , принадлежащий ЛТФ, с паролями чтения -X , модификации A , контроля Y и общим паролем C .

Доступ к уже созданному перманентному файлу

```
ATTACH,имя файла,имя перм. файла,ID= имя подразд,CY=n,PW=пар_I,пар_2,...., MR=1.
```

имя файла – название, с которым этот файл будет дальше использован пользователем в задаче.

CY=n – указание выбрать из файла цикл с номером n . Если пользователь не указывает номер цикла, то из всех существующих циклов будет выбран цикл, у которого самый большой номер.

PW= пароль$_I$,пароль$_2$,... – указываются те пароли, которые необходимы для выполнения требуемых действий с перманентным файлом.

MR=1 – этот параметр обеспечивает режим чтения с этого файла другим задачам. Рекомендуется использовать при обращении к перманентному файлу, на котором записана библиотека программ общего пользования.

Пример 3. Доступ к файлу, созданному в примере I:

```
ATTACH(LGO,PERMANFILE,ID=LCTA)
```

Пример 4. Доступ к файлу, созданному в примере 2:

```
ATTACH(A,PF,ID =LTPH,PW=X,Y,A,C)
```

В этом случае с файлом можно производить все четыре действия.
При обращении

```
ATTACH(A,PF,ID = LTPH, PW=X,C)
```

гарантируется только чтение информации с файла.

Пример 5. Доступ к черновской библиотеке, хранящейся в виде текстов фортрановских подпрограмм

```
ATTACH(OLDPL,LIBCDC,ID=CERN,MR=1)
```

Добавление нового цикла

Добавление нового цикла к уже существующему перманентному файлу осуществляется также управляющими картами REQUEST и CATALOG . Отличие от начального заведения состоит в указании пароля контроля и общего пароля на карте CATALOG:

```
CATALOG,имя файла,имя перм.ф.,ID=подразд.,PW=пароль контроля,общий пароль,CY=n.
```

т.е. при добавлении цикла надо указать только пароль контроля и общий пароль, если они были заданы при начальном создании перманентного файла. Если номер цикла не указан, то создается цикл с номером на единицу больше, чем существующий наибольший номер. Если такой номер цикла n уже существует, то номер присваивается, как и в случае, когда номер не указывается. После добавления система печатает в протоколе задачи

```
NEWCYCLE CATALOG
```

Примеры.

Начальное заведение

```
REQUEST,LGO,*PF.
      :
CATALOG(LGO,PERMANFILE,ID=LCTA)
```

```
REQUEST,LF,*PF.
      :
CATALOG(LF,PF,ID=LTPH,RD=X,
        CN=Y,MD=A,TK=C)
```

Добавление

```
REQUEST,A,*PF.
      :
CATALOG(A,PERMANFILE,ID=LCTA)
```

заведет цикл с номером 2.

```
REQUEST,A,*PF.
      :
CATALOG(A,PF,ID=LTPH,P=Y,C)
```

заведет цикл с номером 2.

Изменение названия перманентного файла и замена паролей

Прежде чем переименовать перманентный файл или ввести новые пароли, надо заказать доступ к файлу, указав все заведенные ранее пароли.

Изменение осуществляется управляющей картой:

```
RENAME(имя файла,новое имя перм.файла,ID=новое имя подразд.,{CN=P1,EX=P2,MD=P3},RD=P4,TK=P5)
                                                            {XR=P1}
```

$P_1,\ldots,P_5$ – новые пароли. Если после знака равенства новый пароль не задан, то это означает отмену ранее заведенного пароля.

Пример. Перманентный файл PFILE был заведен с ID=ABC с паролями чтения X , расширения Y и модификации Z .
Следующие карты

```
ATTACH(A,PFILE,ID=ABC,PW=X,Y,Z)
RENAME(A,PFILE2,RD=,CN=W)
```

дают новое имя перманентному файлу PFILE2 , ID не меняет, уничтожает пароль чтения и добавляет пароль контроля W .
А следующие карты

```
ATTACH(A,PFILE,ID=ABC,PW=X,Y,Z)
RENAME(A, , RD=, EX=, MD= )
```

оставят имя перманентного файла и ID без изменения и уничтожат все пароли.

Уничтожение цикла перманентного файла

Для уничтожения цикла n применяют либо две управляющие карты

```
ATTACH(имя файла,имя перм. файла,ID=подразд.,PW=пароль контроля,общий пароль,CY=n)
```

```
PURGE(имя файла)
```

либо одну управляющую карту

```
PURGE( имя файла, имя перм. файла, ID =подразд., PW =пароль контроля, общий пароль , CY=n )
```

После выполнения этих управляющих карт уничтожается только служебная информация, необходимая системе для хранения этого цикла; информация же цикла, который в задаче пользователя фигурирует как локальный файл, остается на время счета задачи и может быть использована.

Если номер цикла не задается, то уничтожается цикл с наибольшим номером.

Если же после уничтожения цикла в перманентном файле нет больше циклов, то уничтожается вся информация о перманентном файле.

Выдача информации об использовании перманентных файлов

Операционная система, заведя перманентный файл, протоколирует все действия, производимые с этим файлом: когда был создан файл, сколько места занимает он на диске, сколько у него циклов, сколько было обращений к каждому циклу. Получить эту информацию можно при помощи программы AUDIT . Вызов осуществляется управляющей картой

```
AUDIT(LF =имя файла, ID=подразд., PF= имя перм. файла, AI=P )
```

имя файла – на какой файл выдать требуемую информацию. Если этот параметр не указан, то подразумевается LF=OUTPUT.

Пример.

```
AUDIT,ID=LJAP,AI=P.
```

Выдает информацию о всех перманентных файлах, принадлежащих ЛШ.

```
AUDIT,ID=LCTA,PF=имя перм. файла, AI=P.
```

Выдает информацию о всех циклах указанного перманентного файла, принадлежащего ЛВТА.

В этом случае AUDIT печатает следующую информацию о цикле перманентного файла: имя подразделения, имя перманентного файла, номер цикла, на каком системном диске он находится, сколько места на диске занимает (количество секторов, сектор содержит 64 слова), дату создания, до какого срока он будет использован, дату последнего доступа к циклу, дату последнего изменения содержимого цикла.

Здесь были приведены простые примеры использования AUDIT. В действительности, программа AUDIT имеет много других возможностей.

Примеры использования перманентных файлов

Сохранение транслированной программы на диске в виде перманентного файла.

```
шифр.IVANOV
ACCOUNT(...)
REDUCE.
REQUEST,LGO,*PF.
FTN.
CATALOG(LGO,IVANOV62107,ID=LTPH)
 7/8/9
     программа
6/7/8/9
```

Счет по этой программе:

```
шифр. IVANOV
ACCOUNT(...)
REDUCE.
ATTACH(LGO,IVANOV62107,ID=LTPH)
LGO.
 7/8/9
    данные к программе
6/7/8/9
```

Редактирующая программа **UPDATE**

(4 ЛЕКЦИЯ)

В операционной системе NOS/BE 1 есть две редактирующие программы: EDIT , которая работает с терминала, и UPDATE . Слово UPDATE означает "модернизировать, приводить в соответствие с требованием сегодняшнего дня". Работа с программой UPDATE напоминает работу по изданию книги: сначала надо набрать книгу, а потом ее корректировать.

Авторы UPDATE предложили использовать вместо понятия "книги" более емкое понятие - библиотека: а вместо страницы - часть (deck). Таким образом, программа UPDATE предназначена для редактирования библиотеки, состоящей из строк текста, который для удобства разбит на части.

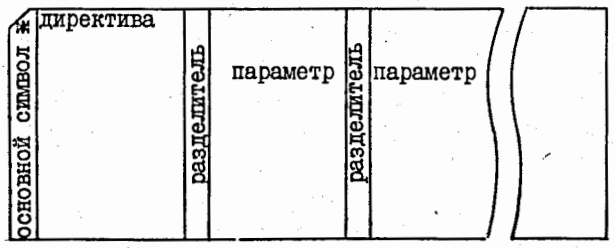
Вся работа UPDATE состоит из двух этапов:

1. Заведение библиотеки.
2. Коррекция библиотеки.

Информация для работы UPDATE задается через параметры p_i при вызове самой UPDATE на управляющей карте:

UPDATE(p_1,...,p_n)

и через специальные карты - директивы:



которые либо вставляются в редактируемый текст, либо находятся среди карт во входной информации к UPDATE .

Основной символ - это некоторый символ (стандартно *), по которому UPDATE отличает текстовую строку от директивы.

Заведение библиотеки

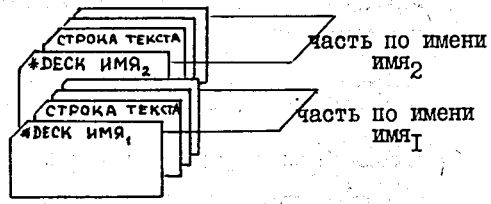
Рассмотрим подробно первый этап – заведение библиотеки. Текст надо пробить на картах, каждая строка текста – это одна карта, информация обычно берется с I по 72 колонку. Этот набор карт надо разделить на части и перед каждой частью положить карту-директиву:

```
*DECK  имя части
       (name)
```

Имя части – от I до 9 символов из набора следующих символов A÷ Z, 0÷9, +-*/( ) $ = . Например, `*DECK A` присваивает имя A всей части. `*DECK 1` дает имя, аналогичное нумерации страниц в книге, т.е. эта часть соответствует странице I.

Все карты, принадлежащие одной части, UPDATE пронумерует и присвоит каждой карте идентификатор, состоящий из названия части и порядкового номера карты в этой части. Тем самым каждая карта приобретает такое имя, по которому нахождение карты в библиотеке определяется однозначно.

Таким образом, для заведения библиотеки надо подготовить следующую информацию:



Если же окажется, что несколько строк текста встречаются в тексте много раз, то нет смысла вставлять один и тот же набор строк много раз. Для этого случая UPDATE предлагает выделить такие повторяющиеся строки в особый вид – общую часть, положить перед ней карту-директиву

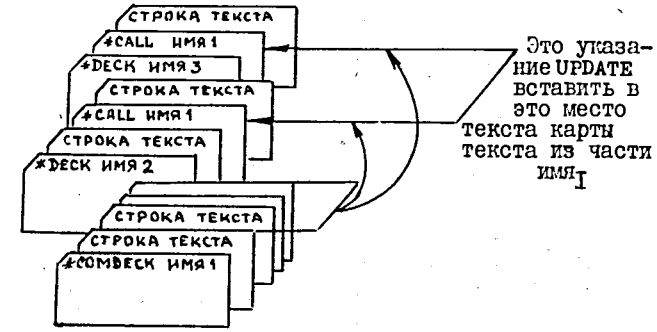
```
*COMDECK  имя общей части
```

а в те места текста, где должны находиться эти строки, вставить директиву

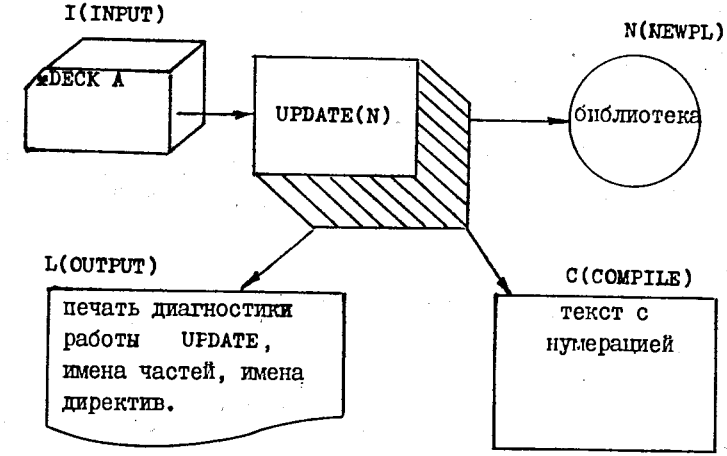
```
*CALL  имя общей части
```

которая и дает указание UPDATE сделать вставку соответствующих строк текста.

В этом случае для работы с UPDATE надо подготовить информацию в таком виде:



Начальный этап заведения библиотеки выглядит следующим образом:



т.е. подготовленную информацию UPDATE возьмет с карт (с файла INPUT), диагностику работы выдаст на печать (файл OUTPUT), библиотеку запишет в файл NEWPL, чистый (без директив к UPDATE) текст с нумерацией карт выдает на файл COMPILE, готовый к использованию каким-либо транслятором, если этот текст был текстом программы. Следует помнить, что представление библиотеки в машине будет разным в зависимости от устройства, на котором будет находиться файл NEWPL.

Пример заведения библиотеки

Пусть у нас есть программа, пробитая на картах:

```
      PROGRAM TEST
      COMMON F
      DIMENSION F(10)
      A=B
      B=C*D
      CALL E
      STOP
      END
      SUBROUTINE E
      COMMON F
      DIMENSION F(10)
      C=0
      D=F
      RETURN
      END
```

Поделим ее на три части. Заметим, что карты

```
      COMMON F
      DIMENSION F(10)
```

встречаются два раза (в программе и подпрограмме). Выделим их в общую часть C, саму программу выделим в часть A, а подпрограмму - в часть B. Заведем библиотеку в виде перманентного файла MYLIBRARY с ID=LCTA. Это можно сделать, пропустив следующую задачу (каждая строка - это одна карта).

```
шифр.
ACCOUNT(...)
REDUCE.
REQUEST,NEWPL,*PF.
UPDATE(N)
CATALOG(NEWPL,MYLIBRARY,ID=LCTA)
COPYSBF (COMPILE,OUTPUT)
7/8/9
*COMDECK C
      COMMON F
      DIMENSION F(10)
```



```
*DECK A
      PROGRAM TEST
*CALL C
      A=B
      B=C*D
      CALL E
      STOP
      END
*DECK B
      SUBROUTINE E
*CALL C
      C=0
      D=F
      RETURN
      END
7/8/9
6/7/8/9
```

В рассмотренном примере UPDATE пронумерует карты, начиная с I в каждой части и добавляя имя части, т.е.

```
*COMDECK C          получит номер      C.1
      COMMON F      ——— " ———          C.2
      DIMENSION F(10)                  C.3
*DECK A                                A.1
      PROGRAM TEST                     A.2
*CALL C                                A.3
        и т.д.
```

Эту нумерацию и видно на печати содержимого файла COMPILE.

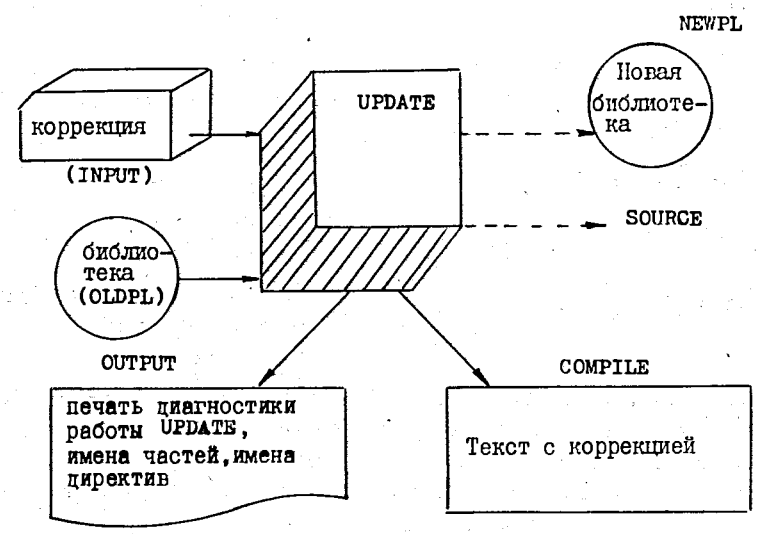
На файл COMPILE будет записан текст с нумерацией, начинающейся с 73 колонки:

```
6 колонка                73 к.
PROGRAM TEST             A.2
COMMON F                 C.2
DIMENSION F(10)          C.3
A=B                      A.4
B=C*D                    A.5
CALL E                   A.6
STOP                     A.7
END                      A.8
SUBROUTINE E             B.2
COMMON F                 C.2
DIMENSION F(10)          C.3
C=0                      B.4
D=F                      B.5
RETURN                   B.6
END                      B.7
```

Таким образом,на файл COMPILE программа UPDATE помещает текст "в чистом виде", и пользователь в дальнейшем может использовать его, например, как входную информацию какого-то транслятора (выбор названия файла COMPILE (транслировать) подчеркивает, что именно пишется в этот файл).

Коррекция библиотеки

После того как библиотека заведена,пользователь может опять же с помощью UPDATE выбирать текст, вводить коррекцию текста, создавать новую библиотеку на базе старой библиотеки и коррекций. В этом случае говорят, что UPDATE работает в режиме коррекций:



Коррекция должна начинаться с карты, где указывается имя, даваемое всей коррекции:

```
*IDENT имя коррекции
      от 1 до 9 символов
```

В качестве символов можно использовать буквы A÷Z , цифры 0÷9, знаки + - * / ( ) $ = .

Например, *IDENT COR1

или *IDENT 1/10/77.

Дальше подкладываются исправления. Исправления, производимые в тексте, аналогичны корректуре книги: зачеркивание строки или нескольких строк текста, замена одной строки или нескольких строк новыми строками, добавление новых строк и т.д. Так же как при корректуре книги, надо указать местонахождение корректируемой строки – имя части, номер, т.е. в какой части находится строка и ее порядковый номер в этой части.

```
*INSERT  C
```

Вставка новых строк после карты с

Карты, находящиеся в коррекции после этой карты и до следующей управляющей карты, будут вставлены после указанной карты С.

*BEFORE C	Вставка новых строк перед картой C

Аналогично предыдущему случаю все карты, находящиеся в коррекции после этой карты и до следующей управляющей карты, будут вставлены в текст перед указанной картой C .

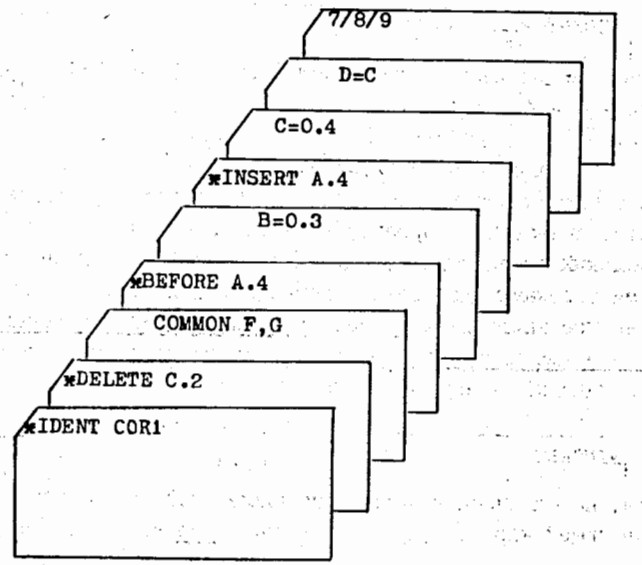
*DELETE C	Зачеркнуть карту C

Если после этой управляющей карты будут находиться карты с текстом, то они будут вставлены вместо вычеркнутой строки C .

*DELETE C_a, C_b	Вычеркнуть карты, начиная с карты C_a и кончая картой C_b.

Если после этой карты будут также находиться карты с текстом, то они будут вставлены вместо вычеркнутых строк.

Рассмотрим пример коррекции для приведенного ранее примера создания библиотеки. Необходимо перед оператором A=B вставить B=0.3 , а после A=B вставить два оператора C=0.4 и D=C , исправить оператор COMMON F на COMMON F,G . Назовем нашу коррекцию COR1 . Тогда вся коррекция будет представлена следующими картами:



Даже на таком простом примере видно преимущество выделения общей части C , т.к. мы корректируем только одну карту C.2(COMMON F), заменяя ее на COMMON F,G . UPDATE сама вставит откорректированный общий текст в нужные места.

Есть две возможности внесения исправлений:
временно (только на данный момент работы UPDATE)
и постоянно (для запоминания на будущее).

Для второй возможности работы необходимо указать режим N в параметрах к UPDATE - заведение новой библиотеки, в которую войдет старая библиотека,и введенные исправления. Следует помнить при этом и о режимах выборки частей из библиотеки, которые записываются в файл COMPILE . Если указать параметр F к UPDATE, то будут выбраны все части, скорректированы и записаны на COMPILE. Если указать параметр Q к UPDATE , то будут выбираться только те части, имена которых указываются на карте-директиве *COMPILE, находящейся среди карт коррекции.

Если не указывать параметр F или Q , то из библиотеки будут выбираться только те части, исправления которым указываются среди карт коррекции, и части, указанные в карте-директиве *COMPILE.

*COMPILE имя части$_1$,имя части$_2$,...,имя части$_n$	Выбрать указанные части из библиотеки.

или

*COMPILE имя части$_1$-имя части$_n$	Выбрать части из библиотеки, начиная с части I и кончая частью n .

Для рассматриваемого нами примера коррекции директивы *COMPILE не потребуется, т.к. библиотека состоит из частей A,B,C и каждую из них мы корректируем, т.е. достаточно вызвать UPDATE в нормальном режиме работы выборки частей в файл COMPILE.

Рассмотрим два варианта внесения исправлений:

```
шифр.
ACCOUNT(...)
REDUCE.
ATTACH(OLDPL,MYLIBARY,ID=LCTA)
UPDATE(L=124)
FTN(I=COMPILE)
7/8/9
*IDENT COR1
*DELETE C.2
      COMMON F,G
*BEFORE A.4
      B=0.3
*INSERT A.4
      C=0.4
      D=C
7/8/9
6/7/8/9
```

```
шифр.
ACCOUNT(...)
REDUCE.
REQUEST,NEWPL,*PF.
ATTACH(OLDPL,MYLIBRARY,ID=LCTA)
UPDATE(N,L=124)
CATALOG(NEWPL,MYLIBRARY2,ID=LCTA)
FTN(I=COMPILE)
7/8/9
*IDENT COR1
*DELETE C.2
      COMMON F,G
*BEFORE A.4
      B=0.3
*INSERT A.4
      C=0.4
      D=C
7/8/9
6/7/8/9
```

В обоих вариантах на файл COMPILE будет записано

```
      PROGRAM TEST          A.2
      COMMON F,G            COR1.1
      DIMENSION F(10)       C.3
      B=0.3                 COR1.2
      A=B                   A.4
      C=0.4                 COR1.3
      D=C                   COR1.4
      B=C*D                 A.6
      CALL E                A.7
      END                   A.8
      SUBROUTINE E          B.2
      COMMON F,G            COR1.1
      DIMENSION F(10)       C.3
      C=0                   B.4
      D=F                   B.5
      RETURN                B.6
      END                   B.7
```


Однако после работы первого варианта библиотека осталась неизменной. И если мы еще раз будем пропускать задачу с данной библиотекой, то исправления надо подкладывать снова. После работы второго варианта в созданной библиотеке будем иметь части A,B,C и корректуру COR1 , т.е. при повторном вызове библиотеки коррекция будет вставляться автоматически.

Если требуется добавить несколько новых частей к библиотеке, то используется карта-директива:

```
*ADDFILE  имя файла, с
```

или

```
*ADDFILE  имя файла, имя части
```

или

```
*ADDFILE  имя файла
```

которая добавляет к библиотеке новые части, записанные в указанном файле (не разрешается добавлять одновременно и коррекции), после указанной карты с (I карта), или после указанной части (2 карта), или в конец старой библиотеки (3 карта).

Добавим к рассмотренной ранее библиотеке две части D,E , пробитые на картах. Это можно сделать следующим образом:

```
      шифр.
      ACCOUNT(...)
      REDUCE.
      ATTACH(OLDPL,...)
      REQUEST,NEWPL,*PF.
      UPDATE(N)
      CATALOG(NEWPL,name1,ID=LAB)
        :
      7/8/9
      *ADDFILE INPUT
      *DECK D
        :
      *DECK E
инф     :
      7/8/9
        :
      6/7/8/9
```

Многие пользуются библиотекой программ ЦЕРНа, которая создана при помощи UPDATE и записана как перманентный файл LIBCDC с ID=CERN.

Чтобы включить несколько частей из этой библиотеки в свою фортрановскую программу, требуется вызвать UPDATE и указать ей, какие части выбрать, с помощью директивы *COMPILE.

Пример использования двух библиотек.

```
        шифр.
        ACCOUNT(...)
        REDUCE.
        ATTACH(OPL,LIBCDC,ID=CERN)
        UPDATE(Q,P=OPL,L=0)
        FTN(I,L=0)
        ATTACH(OLDPL,MYLIB,ID=LCTA)
        UPDATE(F,L=124)
        FTN(I)
        LDSET(PRESET=INF)
        LGO.
        7/8/9
        *COMPILE C335,D520,D122
        7/8/9
        7/8/9
            данные к счету
        7/8/9
        6/7/8/9
```

В этом примере первый раз UPDATE вызывается в режиме Q для выборки из черновской библиотеки, названной OPL, частей C335, D520 и D122. Затем эти части транслируются с помощью транслятора FTN.

Второй раз UPDATE вызывается в режиме F для выборки всей библиотеки, которая хранилась в виде перманентного файла MYLIB с ID=LCTA. Дополнительной коррекции, задаваемой на картах, для этой библиотеки не делалось, т.к. информация к UPDATE состояла из пустого рекорда файла INPUT (т.е. две подряд идущие карты 7/8/9).

В рассмотренных выше примерах обращения к UPDATE мы задавали различные режимы ее работы с помощью параметров N,P,F,Q. В действительности UPDATE имеет до 26 параметров.

Параметры UPDATE

Обращение к UPDATE имеет вид

UPDATE(p_1,...,p_n)

p_i либо режим работы,
либо режим работы = имя файла,
либо режим работы = 0.

Параметры могут задаваться в любом порядке. Если значение параметра не задано, то UPDATE использует стандартное значение.

Укажем наиболее часто используемые параметры:

N	Создать новую библиотеку на файле NEWPL . Если нужно создать новую библиотеку, то этот параметр задавать обязательно.
N=имя файла	Создать новую библиотеку на указанном файле (здесь пользователь сам задает имя новой библиотеки).
P	Использовать старую библиотеку с файла OLDPL, этот параметр можно не задавать, т.к. он является стандартно подставляемым (подразумеваемым).
P=имя файла	Использовать старую библиотеку с указанного файла.
I	Входную информацию брать с файла INPUT, т.е. с карт. Этот параметр можно не задавать, так как он является стандартно подставляемым.
I=имя файла	Входную информацию брать с указанного файла.
C	Текст в виде, пригодном для транслирования, записать в файл COMPILE . Этот параметр можно не задавать, т.к. он является стандартно подставляемым.
C=имя файла	Текст в виде, пригодном для транслирования, записать в указанный файл.
S	Отредактированную библиотеку записать в файл SOURCE в виде входной информации, которая может быть задана в дальнейшем UPDATE для начального заведения библиотеки. Если такой файл нужен, то этот параметр задать обязательно.

S=имя файла — Отредактированную библиотеку записать в указанный файл в виде входной информации, которая в дальнейшем может быть задана UPDATE для начального заведения библиотеки.

Режимы работы

F — Выбрать все части из библиотеки и записать в файл, приготовляемый к трансляции.

Q — Быстрый режим работы UPDATE . Выбрать части из библиотеки, имена которых указаны в карте *COMPILE...

не задан — Нормальный режим работы UPDATE . Выбрать части из библиотеки, имена которых указаны в карте *COMPILE , и части, которые корректировались.

Режим печати

L=0 — Не выдавать на печать никакой информации о работе UPDATE.

L=режимы печати — Режимы печати – это любой набор из букв и цифр, которые соответствуют различным видам печати.

A – напечатать список названий частей и коррекций.

I – напечатать информацию о неправильных директивах с диагностикой, что неправильно.

2 – напечатать список директив, которые UPDATE выполнила.

4 – напечатать карты, которые были во входной информации.

Если не задавать режимы печати, то UPDATE задает L=A12 , если это был режим заведения библиотеки, и L=A1234 , если это был режим коррекции.

Обычно достаточно при работе в режиме коррекции задать L =I24.

Пример обращения к UPDATE

```
UPDATE.
```

Будет выполнено следующее обращение к UPDATE:

```
UPDATE(C=COMPILE,L=A1234,P=OLDPL,
       I=INPUT) ,
```

т.е. берется библиотека с файла OLDPL; исправления этой библиотеки берутся с карт; текст, готовый к трансляции, записывается в файл COMPILE; на печать выдается список всех частей библиотеки, ошибки, если они будут обнаружены, список всех зачеркнутых карт,все введенные карты.

Отметим, что программа UPDATE при своей работе занимает в памяти не более 40000_8 слов и устанавливает указатель записи или чтения файлов OLDPL,COMPILE,NEWPL, SOURCE (или им соответствующих) в начальное положение до начала и после конца своей работы.

На примерах были рассмотрены самые необходимые режимы работы UPDATE. Возможности, предоставляемые программой UPDATE , самые различные. Был рассмотрен пример коррекции, которая записывалась в библиотеку. Карты, составляющие коррекцию, UPDATE тоже нумерует, используя для этого имя коррекции и порядковые номера карт в самой коррекции. Тем самым наряду с коррекцией частей мы можем аналогично корректировать и саму коррекцию. UPDATE предоставляет возможность вычеркивания отдельных строк текста, целиком частей и коррекций, т.е. имеется возможность вернуться на любой предыдущий вариант редактирования. Причем UPDATE допускает два типа вычеркивания: вычеркивание совсем (равносильно уничтожению) и вычеркивание на момент работы коррекции (равносильно зачеркиванию текста карандашом, которое потом легко стереть резинкой). Все эти возможности следует использовать, поработав с UPDATE сначала в самых простых режимах.

Рукопись поступила в издательский отдел
10 июля 1978 года