ОБЪЕДИНЕННЫЙ
ИНСТИТУТ
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
ДУБНА

11 - 10817

М.Ю.Попов, Е.Д.Федюнькин

МОДЕРНИЗАЦИЯ АВТОКОДА **MADLEN**

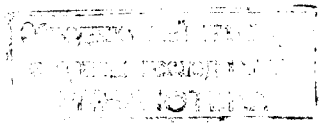
1977

11 - 10817

М.Ю.Попов, Е.Д.Федюнькин

МОДЕРНИЗАЦИЯ АВТОКОДА **MADLEN**

Направлено на IV семинар специалистов СССР и ГДР "Проблемы повышения эффективности ЭВМ БЭСМ-6", Бад-Штур, ГДР, 1977.



Попов М.Ю., Федюнькин Е.Д. 11 - 10817

Модернизация автокода MADLEN

Обсуждаются принципы организации современных автокодов. Описаны возможности новой версии автокода MADLEN. Введена условная трансляция, создан генератор системных текстов. Реализованы новые, более эффективные алгорифмы в основном модуле транслятора. Расширены синтаксические возможности языка.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1977

1. MADLEN и принципы организации современных автокодов

Созданный А.И.Волковым [1,2] транслятор с автокода MADLEN до сих пор является самым быстрым автокодным транслятором для машины БЭСМ-6. В момент своего создания MADLEN, по-видимому, был лучшим из автокодов для этой машины. Однако с тех пор транслятор и структура языка автокода практически не подвергались развитию (некоторые локальные модификации описаны в [3]). Сегодня MADLENуже не удовлетворяет требованиям, предъявляемым к современным автокодам. Ниже перечислены основные аргументы, которые учитывали авторы настоящей работы, создавая новую версию автокода.

1. Отсутствие аппарата условной трансляции является препятствием при создании текстов операционных систем, рассчитанных на варьируемую конфигурацию вычислительного комплекса. Раньше вычислительные комплексы, созданные на базе определенной машины, имели стандартную (заводскую) конфигурацию. Теперь эта эпоха необратимо уходит в легендарное прошлое кибернетики. Возникновение сетевой идеологии незаметно привело к своеобразной революции: вычислительный комплекс перестал быть компактным и статичным; его конфигурация динамически изменяется, следуя запросам пользователя. Короче говоря, сегодня мы имеем дело с развивающимся вычислительным комплексом. С точки зрения эффективности системы человек – машина новый подход определенно имеет прогрессивный характер, а

ностальгические переживания об ушедшей эпохе лишены серьезных оснований.

Практика системного программирования в фирме Control Data Corporation доказала неоспоримые преимущества (в том числе и коммерческие) операционных систем, допускающих широкую автоварьируемость не только по параметрам, связанным с конфигурацией вычислительного комплекса. В частности, поскольку такие системы легко модифицируются, в том числе и глобально, неизмеримо облегчается дальнейшее развитие этих систем.

Итак, сегодня любая операционная система должна допускать изменчивость своей структуры в широких пределах. Автокод, как основной язык системного программирования, должен содержать в себе соответствующие средства.

2. Наблюдается тенденция к созданию узко специализированных языков высокого уровня. Совершенно ясно, что эта тенденция станет одним из главных направлений развития вычислительных систем. Предполагается, что трансляторы с языков такого рода преобразуют входную информацию в текстовый файл, имеющий вид автокодной программы. Дальнейшая трансляция производится автокодом. Такой подход заметно экономит мышление и время при создании трансляторов. Автокод должен при этом допускать возможность высокой степени свертки своей входной информации, т.е. должен иметь развитый аппарат макросов.

3. Автокод по своим техническим возможностям должен стремиться к языкам высокого уровня, освобождая программиста от рутинной работы. Возрастающая сложность операционных систем требует использования более совершенных инструментов. Автокод - главный инструмент системного программиста. Практика показывает, что уровень операционной системы в значительной степени определяется уровнем языка, на котором она написана.

4. Заметную роль в современной идеологии автокодов играет концепция системного текста, на которой мы хотим остановиться подробнее. Системный текст - это

перемещаемая программа, которая должна находиться во временной библиотеке перед вызовом автокода. Системный текст содержит в себе описание идентификаторов и макросов. Имя системного текста является одним из параметров, сообщаемых автокоду в момент его вызова. Автокод, загрузившись, прежде всего настраивает свою таблицу идентификаторов и таблицу кодов операций, используя информацию, полученную из системного текста. Эта информация остается в действии до появления следующей карты вызова автокода. В результате в каждой транслируемой подпрограмме соответствующие идентификаторы (системные символы) и макросы (системные макросы) оказываются заранее определенными. Пользователь может иметь в своей личной библиотеке несколько системных текстов, рассчитанных на разные группы подпрограмм. Практика использования системных текстов позволяет достичь значительной свертки информации и уменьшить количество ошибок. При программировании операционных систем именно системный текст содержит в себе информацию о конфигурации вычислительного комплекса и самой операционной системы.

2. Особенности структуры транслятора MADLEN-2

Транслятор с автокода MADLEN-2 вызывается управляющей картой:

*ASSEMBLER(P1,P2,...,PN) или *ASSEMBLER,P1,P2,...,PN.

Параметры P1-PN имеют вид:

ключевое слово = аргумент .

Следующая информация может быть сообщена транслятору в момент вызова: математический номер файла, с которого читается входная информация (INPUT =); формат листинга и математический номер файла, на который он будет записан (LIST =); имя системного текста (STEXT =); имя внешнего текста и математический номер файла чтения (XTEXT =); входная информация для услов-

ной трансляции по спецификации M (MODEL =); сервисные процедуры и математический номер файла для записи отредактированного текста (EDIT =).

В частности, при EDIT=D включается DEBUG-мода трансляции. При этом локальные идентификаторы заносятся в таблицу описаний стандартного массива для дальнейшего использования отладчиком.

Транслятор является оверлеем, состоящим из двух абсолютных модулей: инициатора (программа- самоубийца) и основного модуля. Когда во входном файле встречается карта *ASSEMBLER, в оперативную память загружается инициатор. Он обрабатывает параметры, загружает основной модуль, загружает системный текст (если нужно), настраивает основной модуль и этот настроенный модуль списывает на барабан. В дальнейшем, до появления новой карты *ASSEMBLER, основной модуль транслятора будет загружаться в оперативную память не из статической библиотеки, будет загружаться его настроенный двойник, спасенный на барабане. Окончив свою работу, инициатор передает управление основному модулю. Место, занимаемое в оперативной памяти инициатором,утилизируется.

Инициатор предполагается в дальнейшем использовать в качестве нулевого оверлея для трансляторов с языков высокого уровня, выходящих на автокод.

Основной модуль является классическим двухпроходным транслятором (со сканированием входного текста на каждом из проходов). Получил дальнейшее развитие удачно использованный Волковым принцип динамического переключателя символов /2/. Переключатель теперь имеет три иерархических уровня и "висит" на двух индексных регистрах. Три уровня сортировки символов позволяют, в основном, избежать использования медленной команды WTC при сканировании входного текста.

В текстовых константах теперь могут быть использованы все символы соответствующего кода: существовавшие раньше ограничения сняты. Русские символы, имеющие одинаковое написание с латинскими, преобразуются к своему латинскому эквиваленту.

Поиск кода операции (КОП), который раньше осуществлялся прямым перебором таблицы, теперь реализован в виде ассоциативного HASH-процесса. Базисный адрес определяется мультипликативным методом. Сама HASH-константа вычислена нами на ЭВМ таким образом, чтобы она давала оптимальное распределение для первоначального фиксированного набора КОПов и обладала достаточно равномерным распределением для произвольного набора. Удалось достичь 80%-ного заполнения базисной таблицы. Длина ассоциативной цепочки только в двух случаях достигает 3. Вход в базисную таблицу состоит из двух слов:

имя КОПа в коде TEXT	
,UJ,SIN	,ZOO,OMON

Здесь SIN - адрес ветви синонимии, OMON - адрес ветви омонимии.

Расширения базисной таблицы могут быть разбросаны в произвольных местах оперативной памяти, каждый вход имеет такую же двухсловную структуру. Ассоциативный принцип использован также в таблице идентификаторов. Этот принцип позволяет достигнуть значительной экономии памяти за счет плотности упаковки: для объектов разной природы возникает единое поле расширений.

Ассоциативная структура таблицы КОПов позволяет легко вводить новые КОПы (в процессе трансляции) и уничтожать существующие. Новый КОП определяется псевдокомандами:

ИМЯ2: ,OPSYN, ИМЯ1 или ИМЯ: A,OPSYN,bbb.

Здесь bbb - восьмеричное число, определяющее машинную операцию, A - восьмеричное число, представляющее собой совокупность атрибутов, ассоциированных с данным КОПом (каждый бит соответствует атрибуту, например: 1 - выход к началу ячейки после трансляции данной команды).

Таблица описаний начинает заполняться уже на первом проходе. В результате удалось избежать очень неприятного ограничения, существовавшего в прежней версии: в адресной части команды теперь может быть использована ссылка на идентификатор, определенный позже.

3. Новые типы констант и идентификаторов

К адресным константам и константам типа INT, REAL LOG может быть добавлен суффикс вида S±M , где ±M--целое десятичное число, которое показывает, на какую степень двойки нужно умножить константу (знак "+" может быть опущен). В случае LOG производится циклический сдвиг на соответствующее число бит влево (+) или вправо (-) в 48-битовой ячейке. Примеры: 11BS4, 100ES-2, 10S+20, 1.1S5.

Константа типа REAL может теперь определять восьмеричное число с плавающей запятой (мантисса снабжается суффиксом B). Константа типа REAL может быть снабжена суффиксом вида P±M. Здесь ±M - целое десятичное число, показывающее, на сколько бит влево от начала ячейки должна быть сдвинута позиция десятичной (восьмеричной) точки. Иначе говоря, таким способом можно генерировать ненормализованные числа. При производстве константы типа REAL все вычисления производятся в 80-битовом регистре (т.е. с двойной точностью), затем используются старшие 40 бит. Только при таком методе гарантируется верность полученного результата во всех случаях. Примеры: 337.51B, 1P2, P0, 777.77BE-1S-2P2 3.14P3E-1S+4, 10BS5E-1 .

В адресных выражениях могут быть использованы константы вида: nHsss, nGsss, nTsss . Здесь n - число символов, а sss - символы, соответственно, из кодов ISO, GOST или TEXT. Производится число, равное двоичному коду соответствующего набора символов, нормализованных вправо. Лидирующие неиспользованные байты заполняются кодом 0B. Примеры: 1HA=101B, 1T0= 20B, 1G2=2, 1HAS3=1010B.

В команде LOG (если она не в группе DATA) в адресном поле может быть записан идентификатор. Тогда генерируется пара команд:

```
:   ,Z00,
    ,Z00,IDENT    .
```


В командах ISO , GOST, TEXT можно опустить ссылку на число символов перед H . В этом случае первый символ после H интерпретируется как разделитель. Пример:

```
,ISO,HWSYSTEM TEXTSW    .
```

Сгенерированная константа будет содержать 12 символов: SYSTEM TEXTS .

Если некоторый идентификатор имеется в системном тексте и определен в теле самой подпрограммы, используется его значение, заданное в подпрограмме.

В автокоде MADLEN-2 появились идентификаторы нового типа: переменные идентификаторы. Они должны быть определены впервые одной из псевдокоманд:

```
A : ,REQU,PA

B : ,MAX  ,AA1,AA2,...

C : ,MIN  ,AA1,AA2,... .
```

В дальнейшем их значение может быть переопределено с помощью этих же псевдокоманд или псевдокоманд EQU, BLOCK . Здесь PA - полный адрес, AA1 , AA2 - абсолютные адреса. Идентификатору B присваивается значение, равное MAX(AA1,AA2,...); идентификатору C, соответственно, - значение, равное минимуму.

Появились новые возможности у псевдокоманды BLOCK:

```
A : W,BLOCK,B(2),C(3),D
```

эквивалентно:

```
B : ,WEQ,A
C : ,WEQ,A+2
D : ,WEQ,A+5
```

```
A : ,EQ,BLOCK,B(2),C(IDENT),D
```

эквивалентно:

```
B : ,EQU,A
C : ,EQU,A+2
D : ,EQU,A+2+IDENT .
```

В обоих примерах A может быть идентификатором, целым десятичным числом, целым восьмеричным числом (с суффиксом B). IDENT- абсолютный идентификатор.

EX,BLOCK,ID1,ID2,ID3(15),(PC),ID4,ID5(10)

эквивалентно:

ID1 : ,SUBP,
ID2 : ,SUBP,
ID3 : ,LC ,15
ID4 : ,SUBP,
ID5 : ,PC ,10 .

4. Производство системного текста

Системный текст производится с помощью обычной автокодной подпрограммы, в которой имя задается оператором:

ИМЯ : , STEXT, .

В подпрограмме могут быть только описания и псевдокоманды OPSYN (текстовые макросы будут реализованы в следующей версии автокода). Все идентификаторы должны быть абсолютными, переменные идентификаторы в системный текст не заносятся. Признак глобального базирования (если он есть) фиксируется в системном тексте. Системный текст помещается во временную библиотеку.

5. Условная трансляция

С помощью аппарата условной трансляции автокодная программа может управлять работой транслятора примерно так же, как программа в машинном коде управляет работой машины. Создавая аппарат условной трансляции, мы следовали в основном синтаксическим принципам, развитым фирмой CDC для автокода COMPASS /4/.

Будем называть оператором IF псевдоинструкцию, которая в результате сравнения некоторых объектов вырабатывает признак, разрешающий или запрещающий трансляцию некоторого числа инструкций, следующих после оператора IF . Общий вид оператора IF :

NAME : IF ,N

Здесь через IF обозначено выражение, охватывающее поле модификатора, поле КОП и часть адресного поля; N - целое число или абсолютный идентификатор, показывающие, сколько инструкций транслировать (или не транслировать). Будем называть оператор IF именованным, если его поле метки непусто. Будем называть оператор IF счетным, если N - не опущено и не равно нулю. Область действия счетного оператора IF определяется числом N , область действия несчетного именованного оператора IF ограничена снизу одинаково именованным или неименованным оператором:

NAME: ,ENDIF, .

Область действия несчетного неименованного оператора IF ограничена снизу любым оператором ENDIF. Оператор END ограничивает снизу область действия любого оператора условной трансляции. Область действия операторов SKIP и ELSE определяется так же, как для оператора IF . Оператор

NAME : ,SKIP,N

безусловно запрещает трансляцию инструкций в своей области действия, он полностью аналогичен оператору IF, запрещающему трансляцию. Оператор

NAME : ,ELSE,N

открывает область условной трансляции, опрокидывая текущую моду трансляции, т.е. после оператора ELSE разрешается трансляция, если до оператора ELSE она запрещалась, и наоборот. Неименованный ELSE срабатывает всегда. Именованный ELSE срабатывает, если он

находится в области действия одинаково именованного или неименованного оператора IF , SKIP или ELSE .

Оператор ENDIF , находящийся в области действия любого счетного оператора условной трансляции, будет включен в счет числа инструкций и больше не окажет никакого действия.

Обозначим через op любую пару символов EQ, NE, GT , GE, LT, LE, имеющих обычный смысл фортранных операторов сравнения.

Реализовано три типа условной трансляции.

1 . Условная трансляция по сравнению полных адресов:

NAME : ,IFop,ADDRESS1,ADDRESS2,N .

При сравнении идентификаторам, определяемым в момент загрузки, приписывается нулевое значение. Операторы IFEQ и IFNE сравнивают полные адреса не только по величине, но и по атрибутам. Пример:

,IFGT,IDENT1+3,IDENT2,5 .

Читается: если IDENT1+3 строго больше чем IDENT2, транслируй 5 инструкций, следующих за настоящей.

2 .Условная трансляция по спецификациям:

NAME : SN,IFSop,STRING,N

NAME: SN,IFIS,STRING,N

NAME: SN,IFOS,STRING,N .

Здесь SN - имя спецификации, STRING - максимум шесть символов кода ISO , с которыми будет производиться сравнение спецификации (за исключением точки, запятой и круглых скобок). Если в STRING присутствует символ *, в соответствующих позициях в обоих объектах сравнения символ заменяется кодом 0B. Оба объекта сравнения нормализуются влево и сравниваются как целые восьмеричные числа. Приведенные выше операторы IF читаются соответственно:

если спецификация op STRING, , то ...

если все символы из STRING содержатся в спецификации, то...

если все символы из STRING не содержатся в спецификации, то...

Спецификация - это максимум шесть символов кода ISO. Существует пять спецификаций: M - вводится с карты вызовов транслятора (параметр MODEL =); V - три символа, версия автокодного транслятора; DD - три символа, версия диспетчера; DT - шесть символов, дата, YYMMDD (YY - две последние цифры года, MM - месяц, DD - день); TM - шесть символов, астрономическое время, HHMMSS (HH - часы, MM - минуты, SS - секунды).

3 . Условная трансляция по атрибутам:

NAME : ,IF,±ATR,IDENT,N .

Здесь IDENT - идентификатор, ATR - одно из ключевых слов DEF , SYS , SUB , LC , PC, ABS и т.д. Оператор при этом интерпретируется следующим образом (соответственно, знак "-" - отрицание атрибута): если IDENT определен к моменту сканирования оператора IF, если IDENT является системным символом, является относительным адресом подпрограммы, общего массива, страничного массива, является абсолютным идентификатором и т.д., то транслируй.

Примеры:

V,IFSGT,003,2

,IF ,-DEF,IDENT,1 .

Операторы читаются соответственно: если версии транслятора строго больше 3, транслируй две инструкции; если идентификатор IDENT все еще не определен, транслируй одну инструкцию.

К аппарату условной трансляции необходимо также отнести инструкцию ERROR:

op,ERROR,N .

Читается: если текущий относительный адрес подпрограммы op N , то подпрограмма объявляется ошибочной, её стандартный массив уничтожается, выход на счет блокируется. Если поле модификатора псевдоинструкции ERROR пусто, подпрограмма безусловно объявляется ошибочной. Если в поле метки псевдоинструкции ERROR находится слово LOCAL, то выход на счет не блокируется. Инструкция ERROR может быть использована в условной трансляции, чтобы при необходимости исключить из пакета целиком некоторые подпрограммы.

Приведем пример эффективного использования аппарата условной трансляции:

```
FLAG : ,REQU ,0
       M,IFSNE,MIN,4
       M,IFSEQ,MAX,2
FLAG : ,REQU ,1
       ,SKIP ,1
       ,ERROR,
MASK : ,EQU  ,=7777777777777777
       ,XTA  ,STORE1
       ,A-X  ,STORE2
       ,AAX  ,=:002
      15,ATX  ,
       ,ARX  ,MASK
      15,AEX  ,
      15,ATX  ,
       ,AEX  ,MASK
       ,IFEQ ,FLAG,1,4
       ,AAX  ,STORE1
       ,STX  ,STORE
       ,AAX  ,STORE2
       ,ELSE ,3
       ,AAX  ,STORE2
       ,STX  ,STORE
       ,AAX  ,STORE1
       ,AOX  ,STORE
```

Если с помощью параметра MODEL вводится "MAX", транслируется вычисление максимума двух чисел, содержащих в ячейках STORE1 и STORE2 (результат - на сумматоре); если с помощью параметра MODEL вводится "MIN ", транслируется вычисление минимума этих чисел; если вводится другая комбинация символов, подпрограмма объявляется ошибочной.

Будем говорить, что один оператор IF вложен в другой, если первый находится в области действия второго. Аппарат условной трансляции имеет бесстековую организацию и поэтому допускает любую глубину вложения операторов IF .

6. Средства управления печатью листинга

На каждой странице листинга, в первой паре строк, печатается заголовок, номер страницы и основные параметры транслятора. Псевдоинструкция TITLE вводит новый заголовок:

, TITLE , ЗАГОЛОВОК - МАКСИМУМ 30 СИМВОЛОВ.

Если в поле метки этой псевдоинструкции находится слово EJECT следующая строка будет печататься на новой странице. Если в подпрограмме отсутствует псевдоинструкция TITLE , имя подпрограммы используется в качестве заголовка.

Печать можно вывести на новую страницу, используя псевдоинструкцию EJECT, (комментарий, максимум 60 символов, печатается в третьей строке новой страницы с первой позиции):

, EJECT , КОММЕНТАРИЙ.

Псевдоинструкция

M, SPACE , N, КОММЕНТАРИЙ

печатает M пустых строк, после чего проверяет, осталось ли на данной странице еще N+1 строк (или N строк, если комментарий пуст). Если да, то печатается комментарий (пустой комментарий не печатается); если нет, то SPACE срабатывает, как EJECT (это происходит и в том случае, когда число оставшихся строк не превышает M).

Указанные выше псевдоинструкции управления листингом в нормальном режиме сами на листинге не печатаются.

Псевдокоманда LOC изменяет печать относительных адресов подпрограммы, приравнивая (для печати) текущий адрес к N:

NAME : ,LOC,N .

Если NAME непусто, NAME объявляется меткой. Если псевдоинструкция LOC помечена, происходит выход к началу новой ячейки. Возвращение к нормальному режиму осуществляется псевдоинструкцией LOC со звездочкой в адресном поле.

Формат листинга задается на управляющей карте вызова транслятора параметром LIST =. Кроме того, существует возможность динамического управления форматом листинга с использованием псевдоинструкции

,LIST,AAA-BBB .

Здесь AAA - форматные спецификации, которые вводятся, BBB - форматные спецификации, которые отменяются, LIST имеет стековую организацию (глубина стека 15). Возвращение к предшествующему формату осуществляется инструкцией LIST со звездочкой в адресном поле. Пример: в нормальном состоянии инструкции, которые не транслируются в результате работы аппарата условной трансляции, не появляются на листинге; инструкция

,LIST,F

вводит в действие спецификацию печати нетранслируемых инструкций.

В конце листинга печатается время процессора, коммерческое и астрономическое время, использованное для трансляции данной подпрограммы.

7. Предыдущие версии автокода

Транслятор "понимает" программы, написанные на автокоде MADLEN (версия 1) и автокоде SIBESM-6.

Транслятор MADLEN-2 находится в опытной эксплуатации.

Во время работы над транслятором MADLEN-2 авторы обсуждали возникавшие проблемы с В.Ю.Веретеновым, А.И.Волковым, М.И.Гуревичем, Г.Л.Мазным, А.П.Сапожниковым, И.Н.Силиным. Всех этих лиц мы искренне благодарим.

Литература

1. Волков А.И. ОИЯИ, Б4-11-4654, Дубна, 1969.
2. Волков А.И. ОИЯИ, 11-5427, Дубна, 1970.
3. Волков А.И. ОИЯИ, 11-5426, Дубна, 1970.
4. COMPASS version 3 reference manual. Pab.60492600. Control Data Corporation. Sunnyvale, California, 1976.

Рукопись поступила в издательский отдел
6 июля 1977 года.