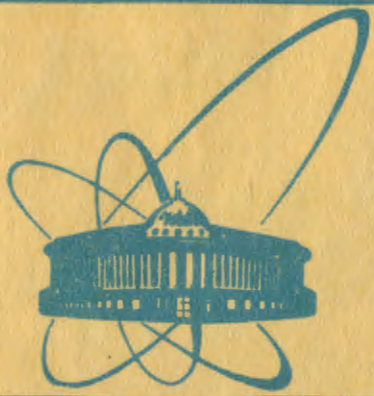

сообщения
объединенного
института
ядерных
исследований
дубна

Р11-80-805

Ли Рен Хи, А.А.Хошенко

МОДЕРНИЗАЦИЯ ТРАНСЛЯТОРА CDL НА БЭСМ-6

1980

Введение

Эксплуатация первой версии транслятора для языка CDL-1 /I, 2,3,5/ для БЭСМ-6 в 1978 году/6/ показала ряд недостатков, которые ограничивали область его применения:

I. Невозможность создания очень больших программ:

а) прежде всего из-за особенностей генерируемой структуры объектного текста на языке МАДЛЕН, представляющей всю программу на языке CDL-1 как одну подпрограмму на МАДЛЕНе,

б) и из-за отсутствия возможности строить оверлейные схемы программ.

2. Отсутствие сервисных средств в трансляторе:

а) управления трассировкой объектной программы;

б) управления выдачей листинга входного текста, распечаткой диагностики, выдачей таблицы идентификаторов, объектного кода.

3. Неоптимальность кодирования скобочных конструкций языка CDL-1.
4. Невозможность использовать в теле макросов глобальные имена.
5. Невозможность использовать библиотечные процедуры непосредственно в теле правил: передача параметров и вызов библиотечных процедур мониторной системы возлагались на пользователя при записи макросов пользователя, что приводило в итоге к неэффективной генерации объектного кода.
6. Невозможность использовать правила с пустым телом.
7. Непригодность существовавшей схемы выхода из правил и макросов для достаточно сложных конструкций: она неудобна при их чтении и неоптимальна для генерации объектного кода.
8. Неудобство существовавшей внутренней кодировки символов в языке CDL-1 для БЭСМ-6, ограничивающей прямое использование библиотечных процедур обработки текстов.
9. Жесткая фиксация логического номера файла и начального номера зоны для правила PUSYM в подпрограммах сопровождения транслятора.

Синтаксические "вкрапления"

I. В плане устранения перечисленных недостатков прежде всего было задействовано понятие раздела, которое имеет следующую форму записи в программе:

'OVERLAY'' < префикс раздела > '
< имя раздела >

где

I) символами " < ", " > " выделены синтаксические единицы языка;

2) имя раздела - обычный идентификатор языка CDL-1;

3) префикс раздела - короткий идентификатор (содержащий не более I+3 знаков), начиная с которого строятся внутренние идентификаторы глобальных имен в объектном тексте.

Например:

'OVERLAY''A' ВТОРОЙ ␣ ПРОХОД

Тогда объектное имя какого-либо правила будет Ai , где i - внутренний глобальный номер правила в текущей программе. Объявление программы в виде раздела должно предшествовать телу программы пользователя и может следовать за текстом, который не содержит глобальных имен (например, за списком стандартных макросов).

Программа пользователя, не содержащая описания (I), объявляется головной программой; это значит, что в объектном теле этой программы будет проводиться генерация кодов для начальной связи с мониторной системой. На рис. I дается схема построения объектной программы для головного раздела.

В объектной программе в имени раздела значащими являются первые шесть значащих символов. Разделы внешние по отношению к данному разделу описываются декларацией:

'external'' overlay' < имя раздела >

Внешние разделы вызываются в объектной программе через LOADGO - подпрограмму.

2. Библиотечные процедуры, которые используются в правилах входного текста (в аффиксной форме - единой форме параметризации вызываемых алгоритмов в CDL), должны быть описаны следующим образом:

'external' 'function' < имя функции > либо
< external' 'subroutine'> < имя подпрограммы >

```
CDL    : 4 ,NAME   ,xxx,INIT
SS     :   ,EQU    ,36276-1B
LIST   :   ,LC     ,SS+1B
GO     :   ,EQU    ,LIST
         3 ,VTM    ,LIST      }
           ,CALL   ,CDLSYS    }  запуск главного
           ,CALL   ,BEGIN     }  раздела
           ,END    ,
G1     :   ,NAME   ,xxx
        14 ,VTM    ,= 6HOVERA  }
           ,ITS    ,14         }
           ,CALL   ,LOADGO     }  вызов нерезидент-
                               }  ного раздела
           ,END    ,
G23    :   ,NAME   ,xxx.READ
           .
           ,END    ,
G396   :   ,NAME   ,xxx.SENTENCE
           .
           ,END    ,
G111   :   ,NAME   ,xxx.INITIALIZEFORREADING

           ,END    ,
BEGIN  : 4 ,NAME   ,xxx  ВХОД В ГЛАВНЫЙ РАЗДЕЛ
           ,CALL   ,G111.INITIALIZEFORREADING
G43    :   ,LC     ,1
        14 ,VTM    ,G43.ZLOGLOB
           ,ITA    ,14
         2 ,ATX    ,-1-1
           ,CALL   ,G23.READ
G45    :   ,LC     ,1
           ,CALL   ,    READ
           ,CALL   ,G396 .SENTENCE
           ,CALL   ,CLOSE
           ,END    ,
```

Рис.I. Структура главного раздела.

Передача параметров этого тела процедурам, как и разделам, соответствует требованиям мониторной системы ОС "Дубна" /4/.

Дополнительно к этому теперь можно придать предикату, флагу, пойнтеру и списку свойство external' , что не допускалось в первой версии транслятора.

3. Глобальные объекты входного текста программы (действие, предикат, пойнтер, флаг, списки) могут быть заявлены как экспортируемые или импортируемые в этой программе, если будут использованы описания типа:

```
'export' 'list' abc (a:b)

'export' 'flag' flag1,flag2          и т.д.
```

Для объектов с этими свойствами не будут в объектную программу генерироваться внутренние идентификаторы – за ними сохраняются имена, которые используются в тексте программы. Импортируемые правила можно теперь описывать на языке CDL . Для того, чтобы они оформлялись в форме неголовной программы, достаточно описать их как оверлей.

Например:

```
          :
      'import'  'action'  действие.   } Описание в
          :                             вызывающей программе
      'overlay' 'b'  действие.        }
          :                           }
вход действие   + x - y% . . .    .   } Описание в
          :                           } вызывающей программе
       abc% . . . .                   }
      'result'   действие.            }
```

4. I) Декларации условной трансляции (прагматы), к которым в первую очередь относятся 'TRACEON' и 'TRACEOF', производят включение и выключение режима генерации объектного кода программы или его участка (какого-либо правила или группы правил), при запуске которого на счет будет проводиться следующий вид трассировки (см. рис.2):
при каждом входе в правило печатаются имя правила и текущие значения его фактических параметров, при выходе из него – имя правила, признак выхода ("ложь" либо "истина"), текущие значения фактических и локальных параметров.

Внутри правил использование этих деклараций (в том числе всех прагматов) запрещено.

. При входе в правило:

```
имя правила:      I     2     3     4     5
                  6     7
```

При выходе из правила:

```
имя правила:   FF I     2     3     4     5
                  6     7

печать локальных  FF 8     9     I0    II    I2
параметров           I3    I4    I5
```

Рис.2. Формат выдачи информации о трассировке.

1) I,2,3,... - номера параметров,
2) признак FF предназначен для сигнализации о способе выхода из правила, а именно:
   FF="++" - обозначает выход из действия,
   FF="-+" - обозначает выход из предиката с результатом "истина",
   FF="--" - обозначает выход из предиката с результатом "ложь".

2) Прагмат 'PAGE' переводит выдачу последующего входного текста и диагностики на следующую страницу листинга.

3) Прагматы 'WARNING ON' и 'WARNING OFF' управляют распечаткой диагностики типа "WARNING".

4) Прагматы 'ASSEMBER ON' и 'ASSEMBLER OFF'управляют выдачей объектного кода.

5) Прагматы 'LIST ON' 'LIST OFF' включают режим выдачи или невыдачи листинга исходного текста программы.

6) 'DICTIONARY OFF' - отключают таблицу ссылок.

7) 'CHECK ON' и 'CHECK OFF' - управляют проверкой переполнения магазина при входе в правило.

5. Введен дополнительный тип комментариев,которые можно использовать также и внутри правил в любом месте, только не внутри членов правил.

Пример:

```
      $  комментарии  $
rule+x:
        equal+x+1,  $  проверка типа    $.
nxt:  clear+x;
      equal+x+2,  $  тип real  $
            :
```

6. Правила с пустым телом можно, например, записать так:

```
действие + x-y %.
предикат I + x-y %+.
предикат 2 + x-y %-.
```

Знак "+" использован для обозначения выхода из правила с признаком "истина", знак "–" – для выхода с признаком "ложь".

Эти знаки можно использовать также в любом месте правил как отдельные члены альтернативы (в том числе и в теле макросов). Знак "–" можно использовать только в предикатах либо в макрофлагах.

7. Введен управляющий символ 'NEXT' , использование которого в качестве члена текущей альтернативы эквивалентно безусловной передаче управления на следующей альтернативе правила. Это его свойство можно, например, использовать при написании и отладке незавершенных правил. Второе его свойство состоит в том, что его использование позволяет сокращать текст объектного правила и ускорять его работу на ЭВМ, а также упрощает чтение сложных правил.

Рассмотрим в качестве примера следующий текст правила:

```
GOOD BOUNDS + LWB + UPB%
        R+SUB,
         (CONSTANT+LWB,
          (R+COLON,
           (CONSTANT+UPB,
            (R+BUS;OMEGA);OMEGA);OMEGA);OMEGA).
```

До введения символа 'NEXT' фрагмент

```
;OMEGA);OMEGA);OMEGA);OMEGA)
```

транслировался в более чем в два десятка автокодных команд.

После введения символа 'NEXT' этот же фрагмент транслируется в несколько декларативных команд автокода МАДЛЕН.

Далее, если скобочные структуры внутри правил оканчиваются передачей управления, то использование символа 'NEXT' позволяет

сократить в объектном коде повторную генерацию команд передачи управления.

Пример участка правила, которое использует 'NEXT'

```
LOAD 1ST FUN PARAM+AFFIX,
NEW NPARS+HEAD+REG,
PUT MACRO1+CALL+HEAD,STORE FUNCT PARAM+AFFIX);
EQUAL+INPT+PLUS,
(IS CONS+AFFIX,%STR;'NEXT');'NEXT');%ERR.
```

результат объектного кода этого участка:

```
L15G251   : , BSS  ,0          .LABDECL
L14G251   : , BSS  ,0          .LABDECL
L1G251    : , BSS  ,0          .LABDECL
            , UJ   ,LERRG251   .JUMP
```

8. Видоизменен синтаксис тела макросов:

- можно использовать глобальные имена внутри тела макросов;
- можно использовать знаки "–" и "+", семантический смысл которых описан в пункте 6; это дает пользователю не только более компактную форму записи тела макроса, но также и более оптимальное кодирование объектной программы при обработке выходов из предикатов и макрофлагов.

Организационные "мероприятия"

I. Кодировка знаков ISO , стандартная в ОС "Дубна", стала стандартной и в CDL .

2. В подпрограммы сопровождения транслятора введены два дополнительных действия:

resymt+char, prsymt+char , с помощью которых можно проводить обмен с терминалом в системе МУЛЬТИТАЙП[4]. При запуске задачи с терминала командой TEP можно проводить прием и выдачу с терминала.

Если задача запущена не с терминала, то выдача через правило prsymt будет проводиться на печать, а ввод – правилом с карт resymt .

3. Введена сокращенная форма выхода из правила за счет применения I4-го регистра, который используется при проверке результата предиката (истина или ложь).

Заключение

Необходимость проведенной модернизации мы связываем с проводящимися в настоящее время работами на языке CDL-1 ,требованиям которых Костер, естественно, не мог удовлетворить в своей первоначальной разработке 1971 года[1]. Реализация же транслятора с языка CDL-2 [6] на БЭСМ-6 требует значительных трудозатрат и явилась бы причиной приостановки текущих работ.

Авторы выражают благодарность А.Корнейчуку за ряд ценных советов по оформлению данной работы.

Литература

1. C.H.A. Koster, A Compiler Compiler, MR 127/71 Nov., Stiching Matematish Centrum, Amsterdam.

2. C.H.A. Koster, Using the CDL Compiler Compiler, Compiler Construction-an Advanced Course T.U. BERLIN.

3. J.C. Jackel. Bootstrap Eines CDL Compiler Auf DIE CDL-6500, T.U. BERLIN ,1975.

4. Мазный Г.Л. Программирование на БЭСМ-6 в системе "Дубна", М., "Наука", 1978.

5. Макаренкова А.Д., Назаров Ю.А., Хошенко А.А. CDL , инструкция для пользователей на БЭСМ-6 и CDC-6500 . ОИЯИ, Б1-11-12214, Дубна, 1978.

6. Макаренкова А.Д., Назаров Ю.А., Хошенко А.А. Внедрение компилятора компиляторов CDL на ЭВМ БЭСМ-6. ОИЯИ, Р11-12340, Дубна, 1978;
"Программирование" № 3, М., "Наука", 1980.

Рукопись поступила в издательский отдел
12 декабря 1980 года.