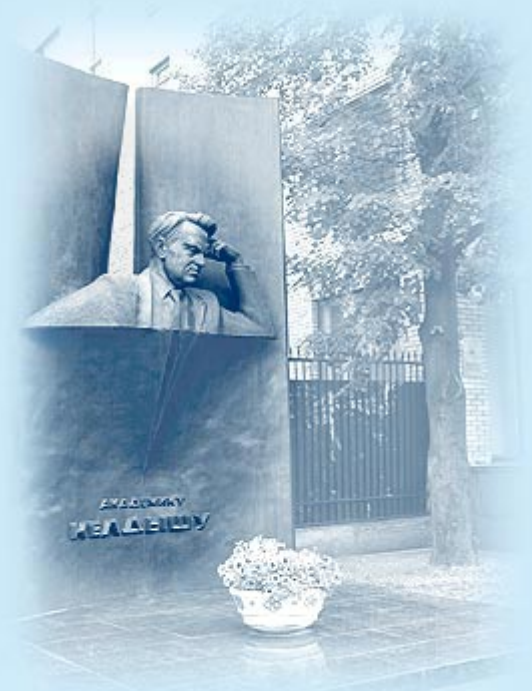




ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 27 за 1983 г.



ISSN 2071-2898 (Print)  
ISSN 2071-2901 (Online)

С. А. Романенко

## Применение смешанных вычислений к ассемблерам и загрузчикам

Статья доступна по лицензии  
[Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/)



**Рекомендуемая форма библиографической ссылки:** Романенко С. А. Применение смешанных вычислений к ассемблерам и загрузчикам // Препринты ИПМ им. М.В.Келдыша. 1983. № 27. 19 с.

<https://library.keldysh.ru/preprint.asp?id=1983-27>



Ордена Ленина  
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ  
имени М.В. Келдыша.  
Академии Наук СССР

С.А. Романенко .

ПРИМЕНЕНИЕ СМЕШАННЫХ ВЫЧИСЛЕНИЙ  
К АССЕМБЛЕРАМ И ЗАГРУЗЧИКАМ

Препринт № 27 за 1983г.

Москва

ОРДЕНА ЛЕНИНА  
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ  
ИМЕНИ М. В. КЕЛДЫША  
АКАДЕМИИ НАУК СССР

С. А. Романенко

Применение смешанных вычислений  
к ассемблерам и загрузчикам

Москва  
1983

УДК 681.3.06

В работе описан опыт применения смешанных вычислений при разработке ассемблера и связывающего загрузчика. Загрузочный модуль рассматривается как остаточная программа, полученная в результате смешанного вычисления ассемблера над исходным модулем с замороженными значениями параметров. Загрузка - как выполнение загрузочного модуля при заданных значениях параметров, результатом которого является абсолютная программа.

**КЛЮЧЕВЫЕ СЛОВА И ФРАЗЫ:** смешанные вычисления, ассемблер, загрузчик.

This paper describes an application of mixed computation to the development of an assembler and linking loader. Load module is considered to be a residual program, produced by mixed computation of an assembler over a source module with the parameter values being frozen. The parameter values given, the load module can be executed to generate an absolute program.

**KEY WORDS AND PHRASES:** mixed computation, assembler, loader.

## СОДЕРЖАНИЕ

Введение.....	4
1. Обозначения.....	4
2. Связь между специализацией и загрузкой программ.....	5
3. Практические выводы.....	6
4. Язык загрузки.....	7
5. Ассемблер.....	10
Литература.....	12
Приложение. Пример работы ассемблера.....	13

## ВВЕДЕНИЕ.

Как указывал А.П.Ершов [3], одной из возможных областей применения смешанных вычислений является 'широкий круг манипуляций, включающих ассемблирование... и в целом обеспечивающий сборку программы из частей, имеющих самостоятельное содержание'.

В данной работе описан опыт использования смешанных вычислений при разработке ассемблера и связывающего загрузчика для системы программного обеспечения (СПО) 'Экономик' для ЭВМ СМ-1800 [4].

## 1. ОБОЗНАЧЕНИЯ.

Пусть  $S$  - двуместная функция, которая для каждой двуместной функции  $F$  и для заданного  $X$  - конкретного значения первого аргумента функции  $F$ , порождает функцию  $S(F, X)$ , удовлетворяющую соотношению

$$S(F, X)(Y) = F(X, Y)$$

для любых  $X$  и  $Y$  из области определения  $F$ . Такую функцию  $S$  мы называем 'специализатором', а функцию  $S(F, X)$  - 'специализацией'  $F$  по  $X$ .

Здесь и далее предполагается, что все функции представлены некоторыми программами. Тривиальной специализацией  $F$  по  $X$  является программа, которая берет аргумент  $Y$  и вызывает  $F$ , подставив в качестве второго аргумента  $Y$ , а в качестве первого - всегда одно и то же  $X$ .

Практический интерес представляет то, что можно построить специализаторы, порождающие нетривиальные специализации, которые являются значительно более эффективными, чем специализации тривиальные [2, 3, 7, 8].

Исходя из этой возможности в [7, 8] была предложена следующая схема построения компилятора компиляторов, выполняющего преобразование интерпретаторов языков программирования в компиляторы этих языков.

Пусть  $I$  - интерпретатор, который для программы  $P$  и исходных

данных  $D$  вырабатывает  $L(P, D)$  – результат применения  $P$  к  $D$ . Тогда  
 $L(P, D) = S(L, P)(D) = S(S, L)(P)(D) = S(S, S)(L)(P)(D)$ .

При этом

$S(L, P)$  – скомпилированная программа,

$S(S, L)$  – скомпилированный компилятор для  $L$ ,

$S(S, S)$  – скомпилированный компилятор компиляторов.

## 2. СВЯЗЬ МЕЖДУ СПЕЦИАЛИЗАЦИЕЙ И ЗАГРУЗКОЙ ПРОГРАММ.

Пусть имеется абсолютный ассемблер  $A$ , который по исходной программе  $P$  строит абсолютную программу  $C$  в машинном коде:  $A(P) = C$ .

Предположим, что  $P$  содержит параметры. Тогда перед запуском абсолютного ассемблера необходимо подставить в  $P$  их конкретные значения. Меняя эти значения, мы получим различные абсолютные программы.

Для дальнейших построений удобно рассматривать абсолютный ассемблер как функцию от исходной программы  $P$  и параметров  $E$ :  $A(P, E) = C$ .

Параметры  $E$  традиционно называют внешними ссылками. (Начальный адрес расположения абсолютной программы в памяти можно тоже считать внешней ссылкой.)

Рассмотрим  $S(A, P)$ . Имеем

$$A(P, E) = S(A, P)(E) = C.$$

Таким образом, остаточная программа  $S(A, P)$  – это функция, которая по значениям параметров  $E$  порождает абсолютную программу  $C$ . Поскольку интерпретация загрузочного модуля загрузчиком при заданных значениях параметров точно так же приводит к порождению абсолютной программы, мы приходим к выводу:  $S(A, P)$  – не что иное, как модуль загрузки.

Далее,  $S(S, A)$  – это ассемблер параметризованных ('перемещаемых') программ, который получается 'автоматически' из абсолютного ассемблера, а компилятор компиляторов  $S(S, S)$  может использоваться как преобразователь абсолютных ассемблеров в ассемблеры параметризованных программ.

Как мы видели, загрузка – это вычисление выражения  $S(A, P)(E)$ , осуществимое только если известны значения всех

параметров  $E$ . Однако, пользуясь специализатором  $S$ , мы можем осуществить 'частичную загрузку' модуля  $S(A, P)$ .

А именно, пусть вектор параметров разбит на две части:  $E = (E1, E2)$ . Тогда

$$\begin{aligned} S(A, P)(E1, E2) &= S(S(A, P), E1)(E2) = \\ &= S(S, S(A, P))(E1)(E2) \end{aligned}$$

где  $S(S(A, P), E1)$  – загрузочный модуль, который зависит только от  $E2$ . Частичная загрузка – одна из основных операций, выполняемых редакторами связей.

### 3. ПРАКТИЧЕСКИЕ ВЫВОДЫ.

К сожалению, только что описанную схему получения параметрического ассемблера еще нельзя применить механически, т.к. работы по созданию автоматических специализаторов еще не завершены. Однако, используя методы, описанные в [2, 3, 7, 8], мы можем строить специализации  $S(S, A)$  'вручную'.

Предложенная схема дает удобную методологическую основу для конструирования ассемблера, языка загрузки и загрузчика, и позволяет на многие частные вопросы взглянуть с новой, а иногда и неожиданной точки зрения.

Например, ясно, что загрузочный модуль  $S(A, P)$  – это остаточная программа, состоящая из 'остатков' абсолютного ассемблера  $A$ , и записанная на том же языке, что и  $A$ . Следовательно, язык загрузки – это подмножество того языка, на котором написан абсолютный ассемблер  $A$ , а загрузчик – это та машина (реализованная аппаратно или программно), которая способна исполнять абсолютный ассемблер. Поэтому язык загрузки, вообще говоря, не имеет ничего общего с языком абсолютных программ, а загрузочные модули могут быть совершенно не похожи на абсолютные программы.

Между тем, широко распространено мнение, что загрузочный модуль – это 'почти' абсолютная программа, но только снабженная комментариями – указаниями для загрузчика. Поскольку в абсолютном тексте нет ни одного лишнего бита – эти комментарии приходится выносить в отдельные таблицы (иногда поражающие своей вычурностью). В результате усложняется генерация загрузочного модуля, усложняются загрузчики и часто возникают жесткие ограни-



чения на вид выражений, обрабатываемых загрузчиком.

Конечно, в те времена, когда приходилось писать программы на языке машины или непосредственно на языке загрузки [5] такая точка зрения была вполне оправдана: по крайней мере программисту не приходилось изучать два совершенно разных языка. Однако сейчас, программирование на языке загрузки, в основном, — удел ассемблеров и компиляторов, поэтому изменились и требования, которые следует предъявлять к языку загрузки.

#### 4. ЯЗЫК ЗАГРУЗКИ.

Как уже отмечалось, язык загрузки — это подмножество языка, на котором написан абсолютный ассемблер. В частности, этим языком может оказаться язык машины. Однако, такой выбор языка загрузки нерационален, ибо язык машины, будучи универсальным языком, не отражает специфику абсолютного ассемблера, из-за чего загрузочные модули получаются непомерно длинными. Поэтому, если мы хотим получить выразительный и компактный язык загрузки, нам следует разработать специализированный язык для написания абсолютного ассемблера.

Существуют различные средства, позволяющие строить специализированные языки на базе универсальных. Для наших целей оказывается достаточным даже самое простое из них — подпрограммы. А именно, можно написать некоторый базовый набор подпрограмм нижнего уровня, а затем, рассматривая вызовы базовых подпрограмм как элементарные операции, запрограммировать 'верхние этажи' абсолютного ассемблера. При этом, загрузочный модуль будет состоять из вызовов базовых подпрограмм, а сами базовые подпрограммы окажутся составной частью загрузчика. Таким образом, и загрузочный модуль, и загрузчик будут представлять собой 'остатки' абсолютного ассемблера.

Пользуясь описанной методикой при разработке ассемблера и связывающего загрузчика для СЮ 'Экономика' [4], автор пришел к следующему языку загрузки.

Программа на языке загрузки состоит из предписаний. Первый байт каждого предписания содержит номер предписания, последующие байты могут содержать кон-

станты - операнды предписания.

Программа на языке загрузки вычисляет арифметические выражения. Каждый раз, когда очередное выражение вычислено, его значение заносится в байт или слово по адресу, содержащемуся в переменной 'текущий адрес' (при этом этот адрес увеличивается на 1 или 2), либо заносится в переменную 'текущий адрес'.

Арифметические выражения представлены в польской инверсной записи и вычисляются с помощью стека. (Как и почти во всех загрузчиках и редакторах связей, способных вычислять сложные выражения. См., например, загрузчик IAL [1] и редактор связей LINK [6]. )

Загрузочный модуль может иметь до 256 параметров ('внешних ссылок'): &0, &1, ..., которым присвоены последовательные номера. Перед началом загрузки параметры получают двухбайтовые значения, которые заносятся в одномерный массив. По принятому соглашению значением параметра &0 становится адрес, с которого должна начинаться загруженная абсолютная программа.

Описываемый загрузчик является связывающим, поэтому он решает две задачи:

1. Связывание (т.е. определение значений параметров модулей).
2. Загрузка (т.е. выполнение загрузочных модулей при известных значениях параметров, в результате которого порождается абсолютная программа).

Поскольку в исходных модулях параметры обозначаются символическими именами и эти же имена используются для связывания модулей, к каждому загрузочному модулю прилагается таблица, которая каждому из параметров &1, &2, ... ставит в соответствие некоторое символическое имя. Символические имена нужны только для связывания модулей и никак не используются при загрузке.

Язык загрузки включает следующие предписания:

0. END; Конец модуля.
1. UNDEF; Неопределенное значение операнда, т.е. ошибка.
2. ABS(C); Занести в стек целое число C.
3. EXT(E,C); Занести в стек сумму значения параметра &E и целого C.

4. ADD; Сложение.
5. SUB; Вычитание.
6. MLT; Умножение.
7. DIV; Деление.
8. W; Взять из стека и занести в текущее слово.  
Текущий адрес увеличить на 2.
9. B; Взять из стека, отбросить старший байт, и занести в текущий байт. Текущий адрес увеличить на 1.
10. ORG; Взять из стека и занести в переменную 'текущий адрес'.
11. REL(C); Эквивалентно EXT(0,C); .
12. XREF(E); Эквивалентно EXT(E,0); .

Кроме того, имеется 128 предписаний с номерами 128, 129,...

255:

128. MO(BO);  
 129. MI(BO, BI);  
 .....  
 255. MI27(BO, BI, ..., BI27),

Предписание МК с номером 128+K имеет K+1 однобайтовых операндов и означает, что нужно занести в память K+1 байтов B1, B2, ..., BK начиная с текущего адреса, а затем - увеличить текущий адрес на K+1.

Рассмотрим следующий пример программы на языке загрузки.

Пусть загрузочный модуль имеет четыре параметра &0, &1, &2, &3 и должен в байт по адресу &0+7 занести значение выражения &2-&3, а в слово по адресу &1 - значение выражения &2/(&3+5). Тогда этот загрузочный модуль можно следующим образом записать на языке загрузки:

```
REL(7); ORG;
XREF(2); XREF(3); SUB; B;
XREF(1); ORG;
XREF(2); EXT(3,5); DIV; W;
END;
```

## 5. АССЕМБЛЕР.

Задача ассемблера – перевести исходный модуль в загрузочный модуль.

Некоторые символы, входящие в исходный модуль, можно объявить параметрами модуля с помощью предложений EXT, имеющих следующий вид:

EXT            символ1, символ2, ..., символN

Каждому из перечисленных символов ставится в соответствие параметр модуля. При этом, первый символ, указанный в первом из предложений EXT, получает значение &1, следующий – значение &2 и т.д.

Значения, приобретаемые символами в процессе ассемблирования, равно как и сами символы, заносятся в таблицу символов. Таблица символов устроена так, что каждый символ может иметь либо неопределенное значение, либо абсолютное значение C, либо относительное значение &E+C, где C – целое число, а &E – параметр. Заметим, что перемещаемые значения являются частным случаем относительных, ибо при E = 0 имеем &E+C = &0+C. В дальнейшем для выражений вида &0+C будет использоваться также сокращенная запись :C.

Выражения исходной программы обрабатываются следующим образом. Каждый операнд выражения заменяется либо на UNDEF; , либо на ABS(C); , либо на EXT(E,C); . Знаки операций +, -, \*, / заменяются на ADD;, SUB;, MLT;, DIV; соответственно. Затем выражение переводится в польскую инверсную запись, причем по ходу дела выполняются следующие сокращения:

EXT(E,C1); ABS(C2); ADD;    → EXT(E,C1+C2);

ABS(C1); EXT(E,C2); ADD;    → EXT(E,C1+C2);

EXT(E,C1); ABS(C2); SUB;    → EXT(E,C1-C2);

EXT(E,C1); EXT(E,C2); SUB;    → ABS(C1-C2);

ABS(C1); ABS(C2); ADD;    → ABS(C1+C2);

ABS(C1); ABS(C2); SUB;    → ABS(C1-C2);

ABS(C1); ABS(C2); MLT;    → ABS(C1\*C2);

ABS(C1); ABS(C2); DIV;    → ABS(C1/C2);

Если обрабатываемое выражение удовлетворяет ограничениям, принятым в большинстве ассемблеров, описанная процедура приводит к полному вычислению выражения, в том смысле, что выражение упрощается до абсолютного значения ABS(C); или относительного значения EXT(E,C); . В противном случае, выполнение части операций задерживается до периода загрузки.

Примеры обработки различных выражений ассемблером приведены в Приложении. Для их понимания необходимо знать следующее. Псевдооперация EQU вычисляет выражение и присваивает его значение символу, стоящему в поле метки. При этом требуется, чтобы выражение упростилось до абсолютного или относительного значения. Псевдооперация DW помещает значение выражения в очередное слово, а DB - в очередной байт. При этом вычисление выражения может быть полностью возложено на загрузчик.

## ЛИТЕРАТУРА

1. Баррон Д. Ассемблеры и загрузчики. М. 'Мир', 1974, стр. 58-60.
2. Ершов А.П. О сущности трансляции. Программирование, 1977, N 5, стр. 21-39.
3. Ершов А.П. Смешанные вычисления: потенциальные применения и проблемы исследования. В сб. 'Теория и практика программного обеспечения ЭВМ. Труды советско-французского симпозиума, сентябрь 1978. Часть I.' Новосибирск 1981, стр. 5-40.
4. Система обработки экономической информации на малых ЭВМ Глава 3. М. 'Книга', 1981, стр. 144-151.
5. Шура-Бура М.Р. Интерпретирующая система ИС-2 для М-20. ВЦ АН СССР, 1965.
6. Fraser C.W., Hanson D.R. A Machine-Independent Linker. Software - Practice and Experience, Vol. 12, 351-366(1982).
7. Turchin V.F. A Supercompiler System Based on the Language Refal. SIGPLAN Notices 14(2), February 1979, pp.46-54.
8. Turchin V.F. The Language Refal, the Theory of Compilation and Metasystem Analysis. Technical Report No. 018, January 1980, Computer Science Department, New York University.

ПРИЛОЖЕНИЕ. Пример работы ассемблера.

\*\*\* АССЕМБЛЕР \*\*\* 29.11.82 13:19:58

ИСХОДНАЯ ПРОГРАММА: ПРИМЕР

ЗАГРУЗОЧНЫЙ МОДУЛЬ: -

ЛИСТИНГ? ПОЛНЫЙ

<<< ДИСК01/БИБ:ПРИМЕР (Т)

>>> ДИСК01/БИБ:ПРИМЕР (Т)

ДЛИНА МОДУЛЯ = 0070

КОЛИЧЕСТВО ВНЕШНИХ МОДУЛЕЙ = 0000

КОЛИЧЕСТВО ВХОДНЫХ ТОЧЕК = 0000

КОЛИЧЕСТВО ВНЕШНИХ ССЫЛОК = 0003

<<< ДИСК01/БИБ:ПРИМЕР (Т)

0001 ;

0002 ; ПРИМЕР РАБОТЫ АССЕМБЛЕРА

0003 ;

:0000 00 0004 R0: DB 0

:0001 00 0005 R1: DB 0

:0002 00 0006 R2: DB 0

:0003 00 0007 R3: DB 0

0008 ;

:0004 0009 DB 0,1,2,3,4,5,6,7,8,9

:0004 00 01 02 03 04 05 06 07 08 09

0010 ;

0011 C10 EQU 10

0012 C20 EQU 20

0013 C30 EQU 30

0014 ;

0015 EXT E1, E2, E3

0016 EIP10 EQU E1+C10

0017 EIP20 EQU C20+E1

```

                                0018 ;
:000E                                0019          DW  0, 1, 2, 3, 4, 5, 6, 7, 8
:000E 0000 0001 0002 0003 0004 0005 0006 0007 0008
                                0020 ;
:0020                                0021          DW  EI, E2, E3, EIP10, EIP20
:0020 &01 &02 &03 &01+000A &01+0014
:002A                                0022          DW  EI+E2, EI-E2, EI*E2, EI/E2
:002A &01,&02,+ &01,&02,- &01,&02,* &01,&02,/
:0032                                0023          DW  EI+E2+E3
:0032 &01,&02, +,&03, +
:0034                                0024          DW  EI*E2+E3
:0034 &01,&02, *,&03, +
:0036                                0025          DW  EI+E2*E3
:0036 &01,&02,&03, *, +
:0038                                0026          DW  EI/E2/E3
:0038 &01,&02, /,&03, /
:003A                                0027          DW  (EI+E2)*E3
:003A &01,&02, +,&03, *
:003C                                0028          DW  EI*(E2+E3)
:003C &01,&02,&03, +, *
:003E                                0029          DW  (EI+E2)*(EI-E2)
:003E &01,&02, +,&01,&02, -, *
:0040 &01+000A 0030          DW  EI+I0
:0042 000A 0031          DW  EIP10-EI
:0044 0000,&01, - 0032          DW  -EI
:0046 0033          DW  (-EI)*E2
:0046 0000,&01, -, &02, *
:0048 0034          DW  -EI*E2
:0048 0000,&01,&02, *, -
                                0035 ;
:004A                                0036          DW  R0, R1, R2, R3
:004A &00 :0001 :0002 :0003
:0052 0001 0002 0037          DW  R3-R2, R3-R1
:0056 :0004 0038          DW  R3-R1+R2
:0058 0039          DW  R3+R1+R2
:0058 :0003, :0001, +, :0002, +

```



```

:005A :001F      0040      DW      (R3-R0)*10+R1
:005C :0067      0041      DW      R3+100
:005E :FF9F      0042      DW      R3-100
                0043 ;
:0060 0006      0044      DW      0+1+2+3
:0062 0002      0045      DW      5-3
:0064 000F      0046      DW      5*3
:0066          0047      DW      5/3,6/7,7/3,8/3,9/3
:0066 0001 0002 0002 0002 0003
                0048 ;

```

>>> ДИСК01/БИБ:ПРИМЕР (Т)

ЧИСЛО ОШИБОК = 0000

ТАБЛИЦА МЕТОК

000A C10	0014 C20	001E C30	&01	E1
&01+000A E1P10	&01+0014 E1P20	&02	E2	&03
:0000 R0	:0001 R1	:0002 R2		:0003 R3

--- КОНЕЦ РАБОТЫ ---

С.А.Романенко. "Применение смешанных вычислений к ассемблерам  
и загрузчикам."

Редактор А.Б.Ходулев.                      Корректор С.А. Романенко.

---

Подписано к печати 16.02.83г. № Т- 03489. Заказ №76.

Формат бумаги 60X90 1/16. Тираж 155 экз.

Объем 0,7 уч.-изд.л.      Цена 5 коп.

055 (02)2



---

Отпечатано на роталпринтах в Институте прикладной математики АН СССР  
Москва, Мясуская пл. 4.

Все авторские права на настоящее издание принадлежат Институту прикладной математики им. М.В. Келдыша АН СССР.

Ссылки на издание рекомендуется делать по следующей форме:  
и.о., фамилия, название, препринт Ин. прикл. матем. им. М.В. Келдыша  
АН СССР, год, №.

Распространение: препринты института продаются в магазинах Академкниги г. Москвы, а также распространяются через Библиотеку АН СССР в порядке обмена.

Адрес: СССР, 125047, Москва-47, Миусская пл. 4, Институт прикладной математики им. М.В. Келдыша АН СССР, ОНТИ.

Publication and distribution rights for this preprint are reserved by the Keldysh Institute of Applied Mathematics, the USSR Academy of Sciences.

The references should be typed by the following form: initials, name, title, preprint, Inst.Appl.Mathem., the USSR Academy of Sciences, year, N(number).

Distribution. The preprints of the Keldysh Institute of Applied Mathematics, the USSR Academy of Sciences are sold in the bookstores "Academkniga", Moscow and are distributed by the USSR Academy of Sciences Library as an exchange.

Adress: USSR, I25047, Moscow A-47, Miusskaya Sq.4, the Keldysh Institute of Applied Mathematics, Ac.of Sc., the USSR, Information Bureau.

**Цена 5 коп.**