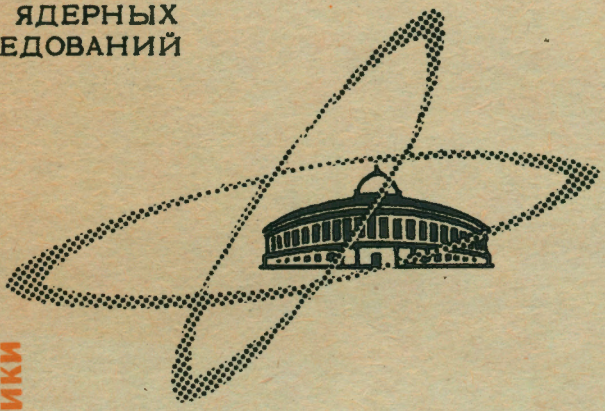
ОБЪЕДИНЕННЫЙ
ИНСТИТУТ
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ

Дубна



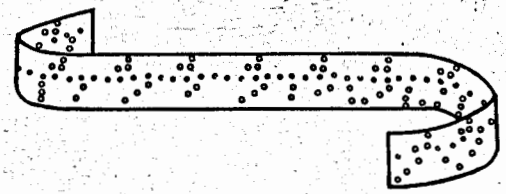
Р11 - 3993

В.А.Загинайко

ИНВАРИАНТНОЕ ПРОГРАММИРОВАНИЕ НА МАШИНЫ М-20, МИНСК-22, БЭСМ-6

ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И АВТОМАТИЗАЦИИ

1968



Р11 - 3993

Объединенный институт
ядерных исследований

ЛВТА

В.А.Загинайко

ИНВАРИАНТНОЕ ПРОГРАММИРОВАНИЕ НА МАШИНЫ М-20, МИНСК-22, БЭСМ-6

Язык инвариантного программирования был разработан автором в связи с необходимостью писать программы на основе одних и тех же алгоритмов на машины с разными системами команд.

Это относится, прежде всего, к логическим программам математического обслуживания машин, трансляторным подпрограммам и т.п.

Благодаря некоторому упрощению языка по сравнению с аналогичными машинно-ориентированными языками (см. /I/), удалось сравнительно простыми программными средствами обеспечить транслирование программ, написанных в инвариантном языке с одного и того же носителя информации (перфокарт) на машины с разными системами команд.

Опишем вкратце процесс трансляции программы с инвариантного языка в команду конкретной машины. Имеется транслятор, работающий на М-20, который каждый оператор языка преобразует в последовательность команд автокода один к одному конкретной машины. Схема трансляции задается таблицей соответствий, представляющей собой описание команд или операторов инвариантного языка в терминах автокода с командами конкретной машины. Таблица соответствий пробивается тем же способом, что и программа на входном (инвариантном) языке, и вводится предварительно транслятором в машину, после чего вводится программа на входном языке и начинается трансляция. Таблица соответствий доступна для пользующегося машиной, что позволяет ему менять в определенных пределах входной инвариантный язык, а также систему команд конкретной машины.

Переход с одной машины на другую осуществляется простой сменой таблиц соответствий.

Ассемблер на М-20 может переводить в истинные адреса автокодный текст с командами не только М-20, а также с командами "Минск-22" и БЭСМ-6. Это устраняет необходимость написания

ассемблеров на "Минск-22" и БЭСМ-6.

Поэтому транслирование программ с входного языка проводится в три этапа.

На первом этапе программа транслируется транслятором с помощью таблицы соответствий в последовательность команд конкретной машины, но пробитых в виде автокодного текста ассемблера М-20.

На втором этапе ассемблер М-20 выдает программу в истинных адресах конкретной машины, но пробитую в виде, пригодном для ввода в память только машины М-20 (или БЭСМ-4). Чтобы ввести эту программу в память другой машины, необходимо перекодировать программу в вид, пригодный для ввода в память этой машины.

На третьем этапе трансляции это делается с помощью набора перекодировочных программ, реализованных на машине "Минск-22". Если программа будет считаться на машине "Минск-22" или "Минск-2", то можно выдать ее на перфоленту или перфокарты, а для БЭСМ-6 выдаются карты в кодировке перфоратора УПП. Переход на машину с какой-либо другой системой команд, например, на СДС-1604, достигается написанием новой таблицы соответствия и перекодировочной программы. Имеется возможность также выдавать транслированные программы в виде автокода один к одному конкретной машины.

В настоящее время реализована выдача программ с инвариантного языка на автокод SIBESM Е.А.Жоголева на машину БЭСМ-6. Ниже будет описан более подробно входной инвариантный язык и таблица соответствий для транслятора.

I. Входной инвариантный язык и таблица соответствий транслятора.

Входной инвариантный язык оперирует с величинами, которые в ячейках могут быть в трех различных представлениях: вещественном, целом представлении, а также в виде составных величин или шкал.

Можно представить себе некоторую абстрактную машину, система

команд которой совпадает с операторами инвариантного языка. Эта машина имеет один индекс-регистр, в который может быть заслано число в целом представлении. Имеется априорное ограничение на длину ячейки абстрактной машины: 36 разрядов (такую длину имеет ячейка "Минск-22").

Расположение целой или вещественной информации внутри ячейки не определено, и может варьироваться в зависимости от конкретной машины.

Программа в инвариантном языке представляет собой последовательность операторов. Операторы могут быть помеченными или непомеченными. Метка отделяется от оператора двоеточием. В конце каждого оператора должен стоять признак конца (точка с запятой). Оператор (непомеченный) представляет собой последовательность идентификаторов, разделителей (знаки + − × / ( ) = > ↑) и служебных символов. Служебный символ представляет собой идентификатор, окаймленный с обеих сторон точками, и выполняет в операторе функции разделителя. Количество и расположение идентификаторов, разделителей и служебных символов внутри оператора в настоящем варианте транслятора строго определено.

Таблица соответствий представляет собой последовательность простых соответствий, каждое из которых имеет вид:

левая часть ≺ правая часть ≺

Металингвистический символ ≺ не должен употребляться внутри левой и правой частей.

Левая часть представляет собой команду инвариантного языка, у которой в качестве идентификаторов используются символы $x№$ где $№$ – восьмеричный номер, идущий в порядке возрастания слева направо от единицы с шагом единица (в работающем варианте $№$ не должен превосходить восьми).

Правая часть представляет собой последовательность любых символов, кроме ≺ , причем за x -ом обязательно должно следовать натуральное 8 -число.

Рассмотрим теперь принцип работы транслятора с инвариантного языка на примере.

Пусть имеется таблица соответствий, описывающая команду условного перехода по совпадению в терминах команд автокода М-20:

.if. x1 = x2 . go to. x3 ;<15, x1, x2; 36,, x3;<.

В этом случае, если в программе входного языка встретится оператор

label : .if. alfa = beta . go to. label 1 ;

то он транслируется так:

label : 15 , alfa , beta ; 36 , , label 1.

Трансляция происходит следующим образом. Сначала оператор подготавливается для поиска. При этом фактически параметры и метка выбрасываются в специальный буфер и заменяются формальными параметрами x №:

.if. x1 = x2 . go to. x3.

При этом содержимое буфера будет иметь следующий вид:

bufer : label
bufer+1: alfa
bufer+2: beta
bufer+3 : label1.

Затем с помощью таблицы соответствий находится правая часть, у которой формальные параметры x № заменяются содержимым соответствующих ячеек буфера.

2. Краткое описание операторов инвариантного языка

x1 > x2 ;	Пересылка содержимого $x1$ в ячейку $x2$.
x1 = x2 ;	Содержимое ячейки $x2$ пересылается в $x1$.
.go to. x1 ;	Безусловный переход в ячейку с меткой $x1$.
.if. x1 = x2 .go to. x3;	Этот оператор описан выше.
.if. x1 = x2 . else. x3 ;	При совпадении содержимого ячеек $x1$ и $x2$ управление передается на следующий оператор, в противном случае (при несовпадении) - в ячейку $x2$.

x1 > .ind. ;	содержимое ячейки $x1$ в целом виде засылается в индекс-регистр;
x1 > x2 + .ind. ;	$x1$ засылается в $x2+i$, где i - содержимое индекс-регистра;
x1 + .ind. > x2;	$x1+i$ засылается в $x2$, где i - содержимое индекс-регистра;
. call. x1 , x2 ;	команда обращения к блоку, причем каждый блок, к которому производится такое обращение, должен начинаться с команды x1 : . subr., и заканчиваться командой x2 ; . exit.
x1 + x2 > x3 ;	Целое сложение: содержимое ячейки $x1$ в целом представлении складывается с содержимым ячейки $x2$ в целом представлении и результат засылается в ячейку $x3$ также в целом представлении;
x1 - x2 > x3 ;	Целое вычитание;
x1 = x2 + x3 ;	арифметическое сложение; содержимое ячеек $x2$ и $x3$ должно быть в вещественном представлении, результат в ячейке $x1$ также вещественный;
x1 = x2 - x3 ;	арифметическое вычитание;
x1 = x2 × x3 ;	арифметическое умножение;
x1 = x2 / x3 ;	арифметическое деление;
x1 = x2 ↑ x3 ;	возведение в степень (вещественное);
x1 = x2 .and. x3;	логическое (поразрядное) умножение ячеек;
x1 = x2 . ∅r. x3 ;	логическое сложение.
. int. x1 > .real. x2 ;	Целое содержимое ячейки $x1$ пересылается в ячейку $x2$ в вещественном представлении.
. real. x1 > . int. x2 ;	Вещественное $x1$ преобразуется в целое $x2$.

.form. x1, x2 >x3 Содержимое ячейки $x3$ сдвигается влево на $x2$ разрядов и в освободившиеся $x2$ разрядов (крайних справа) засылается содержимое ячейки $x2$ в целом представлении, $x1$ должно быть восьмеричным натуральным числом, $x3$-шкала (составная величина).

.desh. x1, x2 >x3; Содержимое крайних правых $x1$ разрядов ячейки $x2$ пересылается в ячейку $x3$ в целом представлении, после чего производится сдвиг составной величины $x2$ вправо на $x1$ разрядов, $x1$ – восьмеричное натуральное число.

.lform. x1,x2,x3>x4; Содержимое ячейки $x4$ сдвигается вправо на $x2$ разрядов, затем в $x2$ левых разрядов ячейки $x4$, начиная с разряда $x1$ засылается содержимое ячейки $x3$ в целом представлении. Нумерация разрядов ячейки $x4$ идет справа налево, начиная с единицы, $x1$ и $x2$ – натуральные восьмеричные числа.

.l desh. x1,x2,x3>x4; Левые $x2$ разрядов, начиная с разряда $x1$ ячейки $x3$, пересылаются в ячейку $x4$, после чего содержимое ячейки $x3$ сдвигается на $x2$ разрядов влево. $x1$ и $x2$–натуральные восьмеричные числа. Нумерация разрядов ячейки $x3$ идет, начиная с единицы, справа налево.

.lform. x1,x2,x3,x4 >x5; Эта команда производит накопление текста по $x2$ разрядов в ячейку $x5$, начиная с разряда $x1$, причем при последовательном выполнении этой команды текст будет располагаться слева направо в отличие от предыдущей команды *lform*. $x4$ – ячейка, из которой (в целом виде) выбирается код символа текста, $x3$– рабочая ячейка. Перед первым обращением к команде накопления текста необходимо очистить ячейки $x3$ и $x5$.

.call. (x1), (x2); Команда обращения к блоку, адрес начала которого содержится (в целом представлении) в ячейке $x1$, адрес конца – в ячейке $x2$ (в целом представлении).

.clean. x1,x2 >x3; Засев массива из $x1$ ячеек, начиная с ячейки $x3$ содержимым ячейки $x2$; $x1$ – натуральное восьмеричное число.

.ecikl. x1 ,x2 ; Повторить выполнение команд, начиная с команды $x2$ до данной команды $x1$ раз. $x1$ – натуральное восьмеричное число; ячейка $x2$ обязательно должна быть описана как начало цикла, а именно – $x2$: .bcikl. ;

В следующих восьми командах условного перехода $x1$ и $x2$ в целом представлении, символы .g. ,.ge. ,.l. ,.le означают соответственно больше, больше или равно, меньше, меньше или равно. Сами команды имеют вид:

```
.if. x1 .g. x2 . go to. x3 ;
.if. x1 .ge. x2 . go to. x3 ;
.if. x1 .l. x2 . go to. x3 ;
.if. x1 .le. x2 . go to. x3 ;
.if. x1 . g. x2 . else. x3 ;
.if. x1. ge. x2. else. x3 ;
.if. x1 . l. x2. else. x3 ;
.if. x1 . le. x2. else. x3 ;
```

В следующих восьми командах условного перехода $x1$ и $x2$ имеют вещественное представление. Смысл символов между $x1$ и $x2$ за вычетом буквы r такой же, как и в предыдущих восьми командах. Команды имеют следующий вид:

.if. x1 . gr. x . go to. x3 ;
.if. x1 . ger. x2. go to. x3 ;
.if. x1. lr. x2. go to. x3 ;
.if. x1. ler. x2. go to. x3 ;
.if. x1 . gr. x2. else. x3 ;
.if. x1 . ger. x2. else. x3 ;
.if. x1. lr. x2 . else. x3 ;
.if. x1 . ler. x2 . else. x3 ;

label: x1 ; Этот оператор указывает, что в ячейке label содержится восьмеричное натуральное число $x1$ в целом представлении.

.if. x1=x2, x3.goto. x4;
.if. x1=x2, x3 .else. x4;

Это условные переходы по совпадению вещественных величин в ячейках $x1$ и $x2$, с точностью до вещественного числа в ячейке $x3$. Условие равенства считается выполненным, если модуль разности $x1 - x2$ не превосходит модуля $x3$.

.stop. Оператор конца задачи (на "Минск-22" и М-20 - команда останова, на БЭСМ-6 это может быть запрещенная команда либо экстракод 074).

x1: bss , x2 ;
x1 : equ , x2;
x1 : bss , x1 ± x2 ;
x1 : equ , x1 ± x2 ;

Эти операторы имеют тот же смысл, что и в ассемблере М-20, а именно: x1: bss, x2 означает описание массива из $x2$ ячеек, начиная с ячейки $x1$. x1: equ, x2; означает, что в транслированной программе $x1$ будет иметь тот же адрес, что и ячейка $x2$.

,x1, x2, x3 ;
,x1, x2, x3, x4;

шкалы по 36 разрядов, причем величины между запятыми занимают по I2 десятичных разрядов.

$x4$ - идентификатор - комментарий.

ЗАМЕЧАНИЕ:

Таблицы соответствий на машины М-20, Минск-22 и БЭСМ-6 выдают автокодные тексты в соответствии с требованиями инструкций [2] , [3] , [4] .

Настоящая система программирования опробована на машинах Минск-22, М-20 (БЭСМ-4) и БЭСМ-6. Имеются программы выдачи текстов и таблиц на широкую печать, транслированные с одних и тех же перфокарт на машины БЭСМ-4 и БЭСМ-6.

В заключение автор выражает благодарность А.Корнейчуку за помощь в корректуре и издании настоящей работы.

Л И Т Е Р А Т У Р А

I. Камынин С.С., Любимский Э.З. Алгоритмический машинно-ориентированный язык алмо. Серия: алгоритмы и алгоритмические языки. Вып. I.

2. Загинайко В.А., Силин И.Н. Инструкция по использованию программы "Ассемблер".

3. Загинайко В.А. Программа ввода карт с перфоратора М-20 в машину "Минск-22".

4. Загинайко В.А. Перекодировочные программы на машине "Минск-22".

5. Загинайко В.А. Инструкция к микротранслятору.

Рукопись поступила в издательский отдел
I8 июля I968 года.