Л. Б. БАБАЯН

ОБ ОДНОЙ ОТЛАДОЧНОЙ ПРОГРАММЕ

Программа, описываемая в настоящей работе, предназначена для облегчения процесса отладки программ. В силу того, что существующие программы отладки носят в основном интерпретирующий характер, их применение для отладки программ требует сравнительно большого машинного времени. Для сокращения этого времени в описываемой отладочной программе (ОП) интерпретация команд используется лишь частично.

Работа состоит из трех параграфов. В первом параграфе даются некоторые определения, необходимые для изложения дальнейшего материала, и определяется класс задач, к которым применима данная отладочная программа. Во втором параграфе описываются входная информация, задаваемая программистом, и выходная информация, выдаваемая ОП. В третьем параграфе описаны основные особенности самой программы.

§ 1. Определения

Приводимые в настоящем параграфе определения преследуют цель дать содержательное представление используемых в дальнейшем понятий и определить класс задач, к которым применима настоящая отладочная программа.

Пусть дана программа

$$\begin{pmatrix} \alpha_1, \alpha_2, \cdots, \alpha_n \\ A_1, A_2, \cdots, A_n \end{pmatrix}$$

где $\alpha_1, \alpha_2, \cdots, \alpha_n$ — адреса машины, расположенные в порядке возрастания, а каждый код A_i есть содержимое адреса α_i $(i = 1, 2, \cdots, n)$.

Определение 1. Реализацией программы назовем последовательность команд, которая получается, если мы будем выписывать последовательно все команды, выполняемые по ходу работы программы, начиная с начальной команды

$$Q = \begin{pmatrix} \beta_1, \beta_2, \cdots, \beta_n \\ B_1, B_2, \cdots, B_n \end{pmatrix}.$$

Заметим, что каждая команда выписывается в том виде, в котором она выполняется.

Определение 2. Переходными командами назовем команды условного и безусловного перехода, условного перехода с возвратом и групповых операций. Команды остальных видов назовем непереходными.

Определение 3. Реализация называется корректной, если для любой пары (β_i, B_i), где B_i команда безусловного перехода с возвратом $B_i = (пу_B - \theta\rho)$ (θ—адрес команды, к которой переходим, а ρ—адрес команды, от которой возвращаемся), в реализации имеется пара (β_k, B_k), такая, что $k > i$, $\beta_k = \rho$ и $B_k = (пу - \beta_i + 1 -)$, причем для всех $i < j < k$, $\beta_j \neq \rho$. (Пару (β_k, B_k) назовем дополнительной парой к паре (β_i, B_i)).

Пусть в реализации

$$S = \begin{pmatrix} \beta_1, \cdots, \beta_i, \cdots, \beta_k, \beta_{k+1}, \cdots, \beta_n \\ B_1, \cdots, B_i, \cdots, B_k, B_{k+1}, \cdots, B_n \end{pmatrix}$$

B_i—команда безусловного перехода с возвратом и (β_k, B_k)—пара, дополнительная к (β_i, B_i).

Определение 4. Элементарным сжатием корректной реализации S относительно пары (β_i, B_i) назовем последовательность

$$\begin{pmatrix} \beta_1, \cdots, \beta_i, \beta_{k+1}, \cdots, \beta_n \\ B_1, \cdots, \omega, B_{k+1}, \cdots, B_n \end{pmatrix}$$

где ω — непереходная команда (очевидно, что $\beta_{k+1} = \beta_i + 1$).

Легко доказывается утверждение: каждое элементарное сжатие корректной реализации S относительно первой пары (β_i, B_i), где B_i—команда безусловного перехода с возвратом, будет корректным.

Определение 5. Полным сжатием корректной реализации S назовем последовательность, полученную из S последовательными элементарными сжатиями относительно всех пар (β_i, B_i), где B_i—команда безусловного перехода с возвратом, строго в порядке их расположения в S.

Пусть задана программа

$$P = \begin{pmatrix} \alpha_1, \cdots, \alpha_r, \cdots, \alpha_{r+s}, \cdots, \alpha_n \\ A_1, \cdots, A_r, \cdots, A_{r+s}, \cdots, A_n \end{pmatrix}$$

имеющая корректную реализацию Q, полное сжатие которой

$$Q_1 = \begin{pmatrix} \beta_1, \beta_2, \cdots, \beta_m \\ B_1, B_2, \cdots, B_m \end{pmatrix}$$

Определение 6. Участок программы P

$$P_0 = \begin{pmatrix} \alpha_r, \cdots, \alpha_{r+s} \\ A_r, \cdots, A_{r+s} \end{pmatrix}$$

назовем простым циклом при ее реализации, если

а) $\alpha_{i+1} = \alpha_i + 1$, $r \leqslant i \leqslant r+s-1$;

б) команда A_{r+s} есть команда условного перехода или групповой операции;

в) ни один из A_i, $i = r, r+1, \cdots, r+s-1$ не является переходной командой;

г) в Q_1 (при данной реализации P_0) имеется участок

$$\begin{pmatrix} \beta_{r_1}, \beta_{r_1+1}, \cdots, \beta_{r_1+s}; \beta_{r_1+s+1}, \cdots, \beta_{r_1+2s+1} \\ B_{r_1}, B_{r_1+1}, \cdots, B_{r_1+s}, B_{r_1+s+1}, \cdots, B_{r_1+2s+1} \end{pmatrix}$$

такой, что

$$\beta_{r_1+t} = \begin{cases} \alpha_{r+t}, & \text{если } t \leqslant s, \\ \alpha_{r+t-s-1}, & \text{если } s < t \leqslant 2s+1. \end{cases}$$

Команду α_{r+s} назовем командой условного перехода данного простого цикла. Адреса α_r и α_{r+s} назовем границами простого цикла или соответственно его началом и концом. Числом повторений цикла (при данной реализации) называется число последовательных вхождений в Q_1 последовательности с адресами $(\alpha_r, \alpha_{r+1}, \cdots, \alpha_{r+s})$.

Пусть заданы последовательности пар

$$P_1 = \begin{pmatrix} \gamma_1, \gamma_2, \cdots, \gamma_r, \cdots, \gamma_{r+k}, \cdots, \gamma_n \\ C_1, C_2, \cdots, C_r, \cdots, C_{r+k}, \cdots, C_n \end{pmatrix}$$

и

$$P_2 = \begin{pmatrix} \bar{\gamma}_{r_1}, \cdots, \bar{\gamma}_{r_1+k} \\ C_{r_1}, \cdots, C_{r_1+k} \end{pmatrix}.$$

Определение 7. Мы скажем, что P_1 содержит P_2 ($P_1 \supset P_2$), или P_2 содержится в P_1, если существует в P_1 такое r

$$1 \leqslant r;\ r+k < n, \text{ что } \gamma_{r+i} = \bar{\gamma}_{r_1+i},\ C_{r+i} C_{r_1+i}\ 0 \leqslant i \leqslant k.$$

Определение 8. Участок программы назовем охватывающим циклом, если:

а) он содержит хоть один простой (охватывающий) цикл;

б) заменяя команды условного перехода всех простых (охватывающих) циклов, содержащихся в рассматриваемом участке, на непереходные команды, участок превращаем в простой цикл. Циклы (простые или охватывающие), содержащиеся в некотором охватывающем их цикле, назовем вложенными по отношению к охватывающему циклу.

В дальнейшем под циклом будем понимать один из определенных выше видов цикла. Под границами цикла будем понимать границы цикла соответствующего вида (границы охватывающего или вложенного цикла определяются аналогично определению границ простого цикла).

Определение 9. Размеченной программой будем называть программу, в которой некоторые адреса непереходных команд снабжены отметками*.

* Содержательный смысл отметок дан в § 2.

Пусть

$$P=\begin{pmatrix} \alpha_1,\cdots, \alpha_r,\cdots, \alpha_{r+s},\cdots, \alpha_n \\ A_1,\cdots, A_r,\cdots, A_{r+s},\cdots, A_n \end{pmatrix}$$

— размеченная программа с корректной реализацией.

Определение 10. Участок программы P

$$L=\begin{pmatrix} \alpha_r,\cdots, \alpha_{r+s} \\ A_r,\cdots, A_{r+s} \end{pmatrix}$$

назовем линейным при ее реализации, если

а) он удовлетворяет условиям а) и в) определения 6;

б) отметками могут быть снабжены только адреса команды A_{r+s};

в) команда A_{r+s} является либо переходной командой, либо командой, адреса которой снабжены отметками, либо командой „останов“;

г) при данной реализации L в полном сжатии реализации P имеется участок

$$\begin{pmatrix} \beta_t,\cdots, \beta_{t+u} \\ B_t,\cdots, B_{t+u} \end{pmatrix}$$

такой, что $\beta_{t+i}=\alpha_{r+i}$ $(i=0, 1, 2,\cdots, u;\ u=s-r)$.

Пусть программа P содержит линейный участок

$$L=\begin{pmatrix} \alpha_r,\cdots, \alpha_{r+s} \\ A_r,\cdots, A_{r+s} \end{pmatrix}$$

и

$$Q=\begin{pmatrix} \beta_1,\cdots, \beta_{t-1}, \beta_t, \beta_{t+1},\cdots, \beta_{t+s},\cdots, \beta_n \\ B_1,\cdots, B_{t-1}, B_t, B_{t+1},\cdots, B_{t+s},\cdots, B_n \end{pmatrix}$$

— полное сжатие корректной реализации P, причем

$$\beta_{t+i}=\alpha_r+i;\ i=0, 1,\cdots, s.$$

Определение 11. L назовем максимальным линейным участком, если последовательность пар

$$\begin{pmatrix} \beta_{t-1}, \alpha_r, \alpha_{r+1},\cdots, \alpha_{r+s} \\ B_{t-1}, A_r, A_{r+1},\cdots, A_{r+s} \end{pmatrix}$$

не является линейной.

В силу того, что в дальнейшем мы будем иметь дело только с максимальными линейными участками, для простоты изложения слово „максимальный“ будем опускать. Условимся обозначать через k_0 и k_1, соответственно номера первой и последней команд рассматриваемого линейного участка.

§ 2. Входная и выходная информация

В качестве входной информации дается начало и конец отлаживаемого участка. Даются также адреса тех команд, результаты выполнения которых нужно выводить (которые снабжены отметками) и для каждого заданного адреса—признак, определяющий вид вывода

этой информации. Для каждого заданного адреса задается число s, определяющее количество необходимых выводов результата при первых s-выполнениях данной команды.

Эти сведения размещаются в памяти следующим образом:

0	0	0	0	0
0	s_3	r_3	N_3	n_3
0	s_2	r_2	N_2	n_2
0	s_1	r_1	N_1	n_1

n_i — адреса команд, которые снабжены отметками. Числа n_i должны удовлетворять следующему условию: $n_i < n_{i+1}$. Нуль означает, что входные данные, задаваемые программистом, окончены. N_i показывает — нужно ли до выполнения команды выдавать содержимые первого и второго адреса. r_i показывает вид вывода (восьмиричный или десятичный). Остальная часть разрядной сетки делится на две части. В первой части записывается нуль, а во второй части записывается количество (s_i) необходимых выводов результата выполнения команды n_i. Отметим, что при работе ОП входная информация сохраняется.

Информация, выдаваемая программой отладки, дает достаточное представление о работе отлаживаемой программы.

Это информация двух видов:

1. Информация о выполнении команд, которые снабжены отметками:

а) адрес команды, содержимое индексного регистра и указание об адресах, подлежащих модификации;

б) содержимое третьего адреса команды, а по желанию программиста и содержимое первого или второго адреса той же команды.

Эта информация выдается в следующем виде:

		A	
		B	
		C	
t 00	000	n	k

где A, B, C — соответственно содержимые первого, второго и третьего адреса; t показывает, какие адреса подлежат модификации, n — содержимое индексного регистра, k — номер команды, снабженной отметкой.

Чтобы отличать эту информацию от другого вида информации, она заключается между двумя определенными метками.

2. Информация о порядке выполнения отлаживаемой программы:

а) для каждого цикла — границы цикла и число его повторений;

б) для каждой команды перехода, не являющейся командой условного перехода цикла—информацию, определяющую данный переход;

в) содержимое индексного регистра (последняя выдается после каждой выдачи информации, указанной в пунктах а) и б)).

Информация вида

—	a	q_1	b
00B	—	m_1	—
—	c	—	d
00B	—	m_2	—

означает, что участок программы, начинающийся с команды a и кончающийся командой b, выполнился q_1 раз. После выполнения указанного цикла содержимое индексного регистра — число m_1. Дальше выполнялись подряд команды с $b+1$ до c и управление было передано на команду d. При последней передаче содержимое индексного регистра равно m_2.

§ 3. Алгоритм реализации отладки программы

Предварительно опишем подпрограммы, используемые в программе отладки.

1. *Подпрограмма засылки команд.* Засылает текущую команду отлаживаемой программы с адресом l в ячейку отладочной программы с адресом l_1.

2. *Подпрограмма выполнения линейного участка.* Формирует команду безусловного перехода на команду k_0 с возвратом по k_1. Выполняется сформированная команда (фактически выполняется соответствующий линейный участок) и восстанавливается содержимое команды с адресом k_1. Заметим, что перед выполнением настоящей подпрограммы всегда выполняется подпрограмма засылки команд, которая запоминает команду с адресом k_1.

3. *Подпрограмма запоминания предыдущего результата и организации условного перехода.* Эта подпрограмма состоит из трех частей. Первая часть меняет код условного перехода с возвратом на код условного перехода и обеспечивает формирование возврата. Вторая часть засылает команду с адресом k_1—1 в ячейку l_3, если $k_1 - k_0 \geqslant 1$, в противном случае выводит нас из подпрограммы. Третья часть организует запоминание содержимого модифицированного третьего адреса команды l_3 с учетом признака результата.

В дальнейшем будет использоваться либо подпрограмма в целом, либо вторая и третья части, либо только третья часть.

4. *Подпрограмма вывода информации о выполненной команде с отмеченными адресами.* Эта подпрограмма служит для выдачи результата выполнения команд, адреса которых указаны во входной ин-

формации. Подпрограмма сравнивает адрес рассматриваемой команды l с адресами команд, результаты выполнения которых нужно выдавать, начиная с наименьшего. Если l не равно ни одному из них, то работа подпрограммы окончена. Заметим, что сравнение выполняется только для адресов команд n_i, для которых $n_i \leqslant l$. С целью сокращения времени работы отладочной программы в линейных участках (для которых $k_0 < k_1$) сравнение выполняется только с наименьшим адресом команд n_i, для которого $n_i > k_0$. Пусть l совпадает с адресом n_0, являющимся адресом команды, результат выполнения которой нужно выдавать (количество требуемых выводов отлично от нуля). Следовательно, l — конец линейного участка. Тогда обращаемся к подпрограммам засылки текущей команды и выполнения линейного участка. Далее, подпрограмма выделяет содержимое первого и второго модифицированных адресов команды l, восстанавливает при необходимости результат предыдущего действия и выполняет команду l. После этого вычитает единицу из количества требуемых выводов результата выполнения команды l заданного в исходной информации. Обращением к третьей части подпрограммы 3 запоминается результат выполнения команды l с учетом признаков. Она выводит при необходимости содержимое первого и второго адресов и содержимое третьего адреса в требуемом виде (в восьмеричной или десятичной системе счисления). Подпрограмма выводит также адрес команды — l, указание об адресах, подлежащих модификации, и содержимое индексного регистра.

5. *Подпрограмма, организующая выполнение начальной команды линейного участка.* Если первая команда линейного участка использует результат предыдущего действия, то в силу того, что этот результат записан при работе подпрограммы 4, данная подпрограмма организует выполнение этой команды с использованием результата.

6. *Подпрограмма фиксации взаимно простых циклов.* Данная подпрограмма фиксирует взаимно простые циклы в рабочем массиве переменной длины (ячейки этого массива обозначим через T_{r-1}, $T_{r-2}, \cdots, T_{r-n}$), предшествующем непосредственно входной информации. После выполнения i-го взаимно простого цикла подпрограмма записывает выводимую информацию о данном цикле в ячейку T_{r-i}, а в ячейку T_{r-i-1} засылает нуль (нуль является признаком окончания рабочего массива). Если выполнен цикл, охватывающий простые циклы, информация о которых записана в ячейках T_{r-i_1}, $T_{r-i_1-1}, \cdots, T_{r-i_1-s}$, то в ячейку T_{r-i_1} записывает выводимую информацию об охватывающем цикле, а в ячейку T_{r-i_1-1} — нуль. Если выполнена команда условного перехода, не являющаяся переходом цикла, то в ячейку T_{r-1} засылается нуль.

7. *Подпрограмма определения циклов.* К данной подпрограмме обращаемся, если рассматриваемая команда является командой условного перехода к команде t_1, причем $t_1 < l$. По информации о взаимно простых циклах, имевшейся в ячейках T_{r-i} ($i = 1, 2, \cdots, n$), дан-

ная подпрограмма определяет, являются ли рассматриваемые циклы взаимно простыми, охватывающими, вложенными, или ни одним из них.

Пусть в данный момент работы ОП в ячейках T_{r-1}, $T_{r-2}, \cdots, T_{r-n}$ записана информация о взаимно простых циклах, а в ячейке T_{r-n-1} записан нуль. Если t_1 больше конца цикла, информация о котором записана в ячейке T_{r-n}, то имеем взаимно простой цикл. Если t_1 меньше начала цикла, информация о котором записана в ячейке T_{r-i}, и больше конца цикла, информация о котором записана в ячейке T_{r-i-1}, то имеем охватывающий цикл, содержащий циклы, информация о которых записана в ячейках T_{r-i}, T_{r-i-1}, $\cdots$, T_{r-n}. Если t_1 больше начала цикла, информация о котором записана в ячейке T_{r-n}, и l меньше конца того же цикла, то имеем цикл, вложенный в цикл с информацией в ячейке T_{r-n}. Теперь опишем содержание самой программы.

Для того чтобы при работе программы отладки сохранить входную информацию, преобразуем ее к виду:

0	0	0	0	0
s_3	s_3	r_3	N_3	n_3
s_2	s_2	r_2	N_2	n_2
s_1	s_1	r_1	N_1	n_1

В дальнейшем программа вывода информации о выполненной команде с отмеченными адресами будет рассматривать и преобразовывать при необходимости информацию, написанную в первой части.

По определению линейного участка, очевидно, что всякую программу можно разбить на такие участки. Такое разбиение осуществляется параллельно с работой программы отладки. Опишем отладку одного линейного участка. Адрес последней команды линейного участка определяется: 1) с помощью подпрограммы вывода информации о выполненной команде, если она является командой, результат выполнения которой нужно выдавать; 2) самой программой отладки, если она является переходной командой.

Программа обнаруживает ее анализом кода операции. Подпрограмма выполнения линейного участка осуществляет работу рассматриваемого линейного участка, кроме последней команды. Последняя команда выполняется в подпрограмме вывода информации о выполненной команде, если она—команда с отмеченными адресами, и в самой программе, если она—переходная команда (к команде t_1). Заметим, что во втором случае ОП меняет адрес t_1 с тем, чтобы переход осуществлялся к определенной команде отладочной программы (t_2). Последнее делается для того, чтобы ОП не отключалась от отлаживаемой программы и имела информацию о данном переходе.

Если переходная команда является командой условного перехода, то при необходимости подпрограмма запоминания предыдущего результата восстанавливает результат предыдущего действия, после чего выполняется данная команда. В случае, когда t_1 больше адреса

команды условного перехода, после выдачи информации о данном переходе в ячейку T_{r-1} засылается нуль, и t_1 рассматривается как начало следующего линейного участка. Заметим, что аналогичные действия выполняются и в том случае, если последняя команда рассматриваемого линейного участка является командой безусловного перехода.

Пусть t_1 меньше адреса команды перехода. Если имеет место переход к t_1 (t_2), то подпрограмма определения циклов определяет, является ли полученный участок циклом, и если это так, то вид полученного цикла. Если не имеем цикла, то выдается информация о данном переходе. Если данный участок является циклом и если при i-м вложении данного цикла не выдавался результат выполнения ни одной команды этого участка, то, начиная с $(i+1)$-го выполнения цикла, интерпретируется только ее последняя команда. После окончания выполнения цикла подпрограмма определения циклов определяет тип данного цикла, а подпрограмма фиксации взаимно простых циклов запоминает информацию о нем в соответствующем месте. После этого выдается информация о выполнении данного цикла. Если имеем охватывающий цикл, то при его вторичном выполнении для каждого цикла, непосредственно вложенного в него, интерпретируется только конец этого вложенного цикла.

Заметим, что до определения вида данного цикла число t_1 сравнивается с определенным числом A — адресом команды, перед выполнением которой последний раз в ячейке T_{r-1} записан нуль. При $t_1 < A$ в ячейку засылается нуль, в противном случае ОП продолжает свою работу.

Программа отладки при каждой возможности переключается на работу отлаживаемой программы.

Пусть некоторый цикл должен быть выполнен n раз и содержит две команды, результаты выполнения которых нужно выдавать соответственно r и s раз ($r, s < n$; в большинстве случаев $r, s \ll n$).

Обозначим

$$\max(r, s) = t,$$

тогда после t-го повторения остальные $(n-t)$-повторений данного цикла выполняются в самой отлаживаемой программе без вмешательства программы отладки.

Отметим, что особенно последнее дает возможность существенно сократить время отладки.

Լ. Բ. ԲԱԲԱՅԱՆ

ՄԻ ՀԶԿՄԱՆ ԾՐԱԳՐԻ ՄԱՍԻՆ

Ամփոփում

Այս աշխատանքում նկարագրվում է մի հզկման ծրագիր, որը ի տարբերություն գոյություն ունեցող նման ծրագրերից, կիսաինտերպրետացիոն բնույթ ունի։ Այս ծրագրին պետք է տալ մուտքի մինիմալ տվյալներ, իսկ հզկվող ծրագրի մասին ստացվում է բավականին լրիվ և կոմպակտ տեղեկություն։