Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

С. К. Черноножкин

СРЕДСТВА ПРОФИЛИРОВАНИЯ ПРОГРАММ В ПРОЕКТЕ СОКРАТ

Препринт
48

Новосибирск 1998

Описываются возможности и назначение системы профилирования ассемблерных и Модула-2 программ, созданной в рамках проекта СОКРАТ. Приводятся необходимые определения и указываются основные подходы при реализации подобных систем. Выделены характерные черты данного подхода. Приводятся примеры работы системы.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

S. K. Chernonozhkin

PROGRAM PROFILING FACILITIES IN THE SOKRAT PROJECT

Preprint
48

Novosibirsk 1998

The system of profiling assembler and Modula-2 programs which has been implemented in the frameworks of the SOKRAT project, is described. Necessary definitions are given and possible approaches to implementation of these systems are analyzed. Main characteristics of the approach used in the project are considered. Examples of the work of the system are given.

1. ВВЕДЕНИЕ

Появившаяся в 1971 г. статья Д.Кнута "Эмпирическое исследование программ, написанных на ФОРТРАНе" [1] вызвала большой интерес как у разработчиков языков программирования и компиляторов, так и программистов. В ней было показано, что язык программирования (во всяком случае ФОРТРАН) используется не так, как это предполагалось. Статья положила начало новой серии экспериментальных исследований программ, их статических и динамических характеристик.

В данной работе будут рассматриваться только инструменты, позволяющие узнать динамику выполнения исследуемой программы. Совокупность таких средств для исследования динамических характеристик программ получила название *профилировщик*. Основной задачей данных инструментов является построение так называемого профиля программы.

В зависимости от рода выполняемых исследований профили могут содержать различные наборы динамических характеристик программ. Обычно в профиль входит информация о частоте и времени выполнения операторов, о вероятности выполнения логических условий, о предельных значениях некоторых переменных, о количестве итераций в циклах и т. д. Для сбора информации о работе программы в течение длительного периода имеются средства для сохранения или обобщения полученных результатов.

Данная работа описывает инструменты получения и визуализации профиля программ, написанных на языке Ассемблер ЕС ЭВМ и Модула-2 в кросс-системе программирования. Для создания профиля используется диалогово-пакетный кросс-отладчик, который моделирует работу встроенной ЭВМ с архитектурой типа ЕС или VAX. Полученный набор средств для измерений динамических характеристик программ (сокращенно ИДП) — составная часть подсистемы контроля качества программного продукта, которая, в свою очередь, является частью системы СОКРАТ [2].

2. ОБЗОР ПРИМЕНЕНИЯ ПРОФИЛЕЙ И МЕТОДОВ ИХ ПОСТРОЕНИЯ

Профили программ имеют самые разнообразные применения. Первые свои плоды профилирование принесло в экспериментальных исследованиях программ [3]. Основные цели данных исследований состояли в

развитии средств программирования и аппаратуры вычислительных систем, а достигались путем выяснения реального использования средств языка. В каждом отдельном случае производилась выборка программ, написанных на языке высокого уровня, и с помощью инструментальных средств собиралась и анализировалась информация о ходе их исполнения. Результаты произведенного анализа позволяли увеличивать как скорость трансляции на этапе лексического [4] и синтаксического [5] анализа, так и скорость выполнения получаемых программ [1], оптимизируя компиляцию тех конструкций, которые применяются чаще всего.

Результаты профилирования можно использовать на этапе улучшения качества своей программы, а именно: подвергать улучшению те участки программы, на которых она затрачивает больше всего времени исполнения. Эта же информация может быть использована для построения оптимизирующего компилятора, работу которого можно ускорить, если он будет оптимизировать лишь отдельные конструкции языка, а не весь текст [6].

Вместе с этим профиль программы используется также для получения динамической смеси операторов программы [7], которая, в свою очередь, позволяет определить подходящую вычислительную установку для программ, порождающих эту смесь и обрабатываемых на ней в дальнейшем.

Особый интерес представляет применение профилирования на этапе тестирования программ [8]. Получаемые динамические характеристики помогают выделить интенсивно используемые компоненты программ, которые целесообразно подвергать наиболее тщательной проверке. Вместе с этим находятся компоненты, с малой или нулевой частотой проверки исполнения, подлежащие дополнительному тестированию или исследованию на достижимость. Анализ статистики условных переходов и итераций циклов позволяет обнаружить некоторые типы ошибок.

Выделяют три основных способа получения профилей. Первый — использование частотных счетчиков [1,9]. Основной его идеей является введение в исследуемую программу, представленную на исходном языке программирования, средств на том же языке, обеспечивающих подсчет и запоминание числа выполнений операторов. В качестве таких средств могут выступать специальные операторы, увеличивающие при каждом выполнении заданной конструкции счетчик, соответствующий этой конструкции, или операторы обращения к специальной изме-

рительной подпрограмме, обеспечивающей подсчет и вывод на печать или в файл полученной информации. Этот наиболее простой метод хорошо интерпретируется в терминах исходной программы. Недостатком его является выделение дополнительной памяти под счетчики, а кроме того, введенные измерительные средства могут значительно увеличить объем исследуемой объектной программы и стать причиной дополнительных ошибок, так как такая разметка программы в общем случае не формальный процесс, и поэтому его нельзя автоматизировать, а человека, как основной источник ошибок, нежелательно в него включать.

Следующий метод основан на прерывании выполнения программы через одинаковые промежутки времени с запоминанием адреса прерывания и, возможно, связанной с этим адресом семантической информацией [1] во внешней памяти. Эти средства пригодны для программ, написанных на любом языке. Существенный недостаток метода — замедление выполнения программы в связи с частыми обменами с внешней памятью и трудностью трактовки полученных результатов.

И последний метод — интерпретация объектной программы с подсчетом используемых команд [1,10]. Обеспечивается сбор столь же полной информации, как и при применении частотных счетчиков, ее трактовка в терминах исходной программы тоже затруднена, так как получаемые результаты относятся к адресам команд объектной программы. С некоторыми изменениями данный метод используется в нашей работе. Средства интерпретатора и среды отладки позволяют как получать по исходному тексту программы доступ (адрес) к части объектной программы, так и определять по адресам исследуемой объектной программы соответствующие им объекты исходного текста,что обеспечивает легкую трактовку результатов.

3. ПОСТАНОВКА ЗАДАЧИ И ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Цель данной работы — разработка и реализация набора инструментов для измерения динамических характеристик программ и их визуализации.

Профиль программы — это набор определенных характеристик, полученный в процессе рабочего функционирования программы. Этот набор не постоянен и может меняться от задачи к задаче в зависимости от того, какие характеристики необходимо получить о ходе выполнения программы.

В данной работе профиль содержит информацию о динамике вы-

полнения строк, операторов, процедур и характере изменения значений переменных программы. Средства для построения профиля называются *набором инструментов* для измерения динамических характеристик программ, или *профилировщиком*.

Здесь под динамикой выполнения операторов понимается набор характеристик, получаемый при исполнении программы. Одна из таких характеристик — частота выполнения оператора при прогоне программы, получившая название *счетчик частоты*. Из этой характеристики можно получить другую, не менее важную — отношение счетчика частоты одного оператора к общему числу выполненных к определенному моменту операторов, — позволяющую легко выделить наиболее активные компоненты программы.

Следующая характеристика определяет длительность выполнения оператора в процессе исполнения программы. В нашей работе она измеряется в тактах работы процессора; может применяться для обнаружения тех операторов программы, которые затрачивают на свое исполнение наибольшее или наименьшее время по отношению ко всей программе или к отдельным операторам.

Говоря о характере изменения значений переменных, мы имеем в виду отрезок, в котором лежат все принимаемые ею значения в ходе выполнения программы. Сюда входит также еще одна характеристика, позволяющая узнать реальное использование переменной — число обращений к ней и число ее изменений.

Проиллюстрируем эти характеристики на примере.

Пусть

- CS (Count Statement) — число выполнений оператора,
- TS (Time Statement) — время выполнения оператора,
- BV (Bound Variable) — граничные значения переменной,
- CV (Count Variable) — число обращений к переменной.

Ниже приведен пример ассемблерной программы и результаты профилировщика для данного фрагмента.

```
IND  DC   F'-1'                          BV(IND) = [-1,-1]
     ...
     L    R1,5        CS = 1    TS = 2
     LA   R2,LP       CS = 1    TS = 2
     L    R3,7        CS = 1    TS = 2
     ...
LP   ST   R1,IND      CS = 5    TS =15   BV(IND) = [-1, 5]
```

```
     BCTR R1,R2       CS = 5    TS =10
     L    R4,IND      CS = 1    TS =2
     S    R4,IND      CS = 1    TS =2
     BZ   IF1         CS = 1    TS =3
     ST   R3,IND      CS = 0    TS =0
IF1  ...
                                          BV(IND) = [-1, 5]
                                          CV(IND) = 7
```

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ ПРОФИЛИРОВЩИКА

Основу входных данных вместе с текстом исследуемой программы составляет набор отслеживаемых объектов. В зависимости от целей профилировки описанный в данной работе алгоритм позволяет управлять степенью детализации динамической информации о программе.

Характеристики, связанные с объектами программы, можно детализировать по числу и типу отслеживаемых объектов. При измерении количественной характеристики исполнения выбираемый набор может быть всей программой, некоторыми ее процедурами или же состоять из отдельных строк операторов.

Временны́е характеристики также можно получать для всей программы в целом или для выбранных фрагментов — блоков. Время для таких фрагментов равняется сумме времен исполнения каждого входящего в него оператора. Блоки операторов не могут вкладываться друг в друга, пересекаться, начинаться в одном модуле программы и кончаться в другом. Блок может содержать одну строку программы. Блоки полезны для получения информации о выполнении фрагментов программы. Для ассемблерных программ они могут быть эквивалентами операторов языков высокого уровня. В частности, из-за отсутствия явного оператора цикла (его вводят с помощью операторов перехода и сравнения) можно легко находить временны́е характеристики его работы, помещая в блок все его операторы. Тогда профиль программы будет содержать время выполнения цикла.

Следует отметить, что в данной работе эти две характеристики — нахождение времен выполнений блоков и отдельных строк операторов — неразрывно связаны между собой и получаются одновременно по мере выполнения программы.

Специфика данного профилировщика еще состоит в том, что он поз-

воляет получать информацию об использовании переменных программы. Так же, как и для операторов, может производиться выбор отслеживаемых переменных, и если переменные сложные, то только тех их простых полей, которые необходимы.

В отличие от других подобных систем в данной работе было введено и реализовано понятие *частичного* профиля, получаемого с помощью средств, предоставляемых рабочей средой профиля (РСП) — диалогово-пакетным кросс-отладчиком [11,12]. Обычно профилировщики строят профиль программы для всего времени ее исполнения, так как элементы профилирования задаются до исполнения и не могут быть изменены до завершения исполнения и нет возможности посмотреть результаты во время исполнения программы. Профиль создается после исполнения путем обработки собранных данных. Эта система позволяет в отличие от других, где необходимые характеристики получаются при завершении выполнения программы, расширить возможности профилирования, вводя *частичный* профиль, который характеризует работу программы лишь на некотором ее временно́м участке. Другими словами, пользователь задает точки останова, используя средства отладчика или РСП, и в ходе исполнения программы после остановки на одной из ранее поставленных точек останова задает набор отслеживаемых характеристик для профилирования и, возможно, новые точки останова. При дальнейшем исполнении, в момент остановки программы, существует возможность посмотреть желаемые характеристики и сохранить их в файле, а если необходимо, изменить набор характеристик, и т. д. на протяжении всего сеанса исполнения программы. Это очень полезный объект при отладке программ, когда необходимо узнать динамические характеристики программы на некотором участке ее исполнения.

Результаты работы профилировщика, являясь выходными данными, составляют профиль программы. Их можно просмотреть используя средства визуализации профилировщика или исследуя файл, в который записываются результаты. Данный файл можно использовать для накопления и хранения результатов профилирования на нескольких тестах, в этом случае из файла не только выбираются объекты программы для профилирования, но и суммируются получаемые результаты для всех тестов, что является прекрасной возможностью для исследования поведения программы на наборе тестов и позволяет каждый раз экономить на задании набора исследуемых характеристик.

5. СРЕДСТВА ДЛЯ ПОСТРОЕНИЯ ПРОФИЛЯ

Рассматриваемый в данной работе профилировщик собирает информацию о динамике исполнения ассемблерных программ и Модула-2-программ для машин с архитектурой типа ЕС ЭВМ и VAX. Речь идет о двух различных кросс-профилировщиках с похожими функциями и целями и реализованными в среде соответственно двух кросс-отладчиков (КРОТ [11] и DVX [12]) с похожими функциональными возможностями. Первый из них — прототип второго, который вбирает в себя все достоинства и развивает основные идеи первого.

Отладчик предоставляет самые необходимые средства типичного отладчика — возможность введения в программу различных точек останова, средства для трассировки выполнения программ, возможность доступа к переменным, проверку и, возможно, модификацию их значений в этих точках и т. д. Профилировщик, встроенный в эту среду, является дополнительным средством отладчика при исследовании динамических характеристик программы и определении ее качества. Средства профилировщика могут находиться в одном из двух состояний — активированном или деактивированном. Активированное состояние профилировщика заставляет отладчик запускать подпрограммы профилировщика по мере надобности. После деактивации средства профилировщика становятся недоступными.

Оба отладчика — диалогово-пакетные кросс-отладчики. Обеспечивается возможность работы как в диалоге, так и в пакете. Диалоговые возможности отладчиков весьма богаты и ничем не уступают всем известным отладчикам. Для пакетной отладки разработан специальный язык отладки. Естественно, для целей профилирования обе эти возможности расширены: в диалог введены дополнительные меню и кнопки, в пакете в язык отладки — набор команд профилирования, а также разработаны инструменты, реализующие добавленные функции и команды.

6. ОСНОВНЫЕ ФУНКЦИИ И ИХ РЕАЛИЗАЦИЯ

Главное назначение профилировщика при исполнении программы состоит в подсчете числа исполнений определенных, заданных пользователем объектов программы, а также числа обращений и/или изменений ее переменных (тоже не всех, а заданных), а для исполняемых объектов — времени их исполнения. Объектами, для которых производится подсчет их числа исполнений, являются строки, операторы, процедуры

и блоки — подряд расположенные строки программы. Для ассемблерных программ строка совпадает с оператором и отсутствует понятие “процедура”. Время исполнения подсчитывается для блоков и процедур, а в случае ассемблерных программ — и для каждого выделенного оператора.

Таким образом, для реализации перечисленных функций необходимо:

— обеспечить выделение пользователем следующих объектов программы: строк, операторов, блоков, процедур, переменных и в случае сложных переменных их простых составляющих;

— обеспечить во время исполнения программы порождение определенного события при достижении выделенных объектов;

— написать подпрограммы обработки порождаемых событий.

В свою очередь, среда отладчика уже содержит механизмы порождения событий, необходимо только добавить типы вновь создаваемых событий и реализовать методы их обработки. Создаваемые методы с идеологической точки зрения очень просты — надо для каждого исследуемого объекта завести счетчик или счетчики и при необходимости увеличивать их значения. Основная задача — обеспечение эффективного доступа к исследуемым объектам.

Диалоговый режим предоставляет максимальные возможности для задания исследуемых объектов. Для этого разработана специальная система меню, позволяющая задать и пометить объекты в программе (строки, операторы, блоки, процедуры, переменные и их составные части); все исследуемые объекты специальным образом выделяются на экране (так, все строки профилирования при показе текста модуля помечаются специальным символом в некоторой колонке экрана примерно так же, как и точки останова в отладчике, а при просмотре переменных те из них, которые исследуются профилировщиком, выделяются цветом). При исполнении программы в диалоге можно в некоторой точке останова отладчика посмотреть все собранные на текущий момент значения характеристик для любого из исследуемых объектов. Возможно сохранение полученного профиля и, например, изменение исследуемых объектов для построения нового профиля и т. д.

В пакетном режиме возможности для задания исследуемых объектов скромнее, но существуют прекрасные условия для сочетания диалоговых и пакетных возможностей, а именно: в диалоге задаются нужные объекты и сохраняется пустой профиль для них — получается так назы-

ваемый “режимный” файл для профилировщика, содержащий все необходимые объекты для исследования. В дальнейшем “режимный” файл используется в пакетном режиме для получения полного профиля программы на наборе тестов.

Естественная разница в уровне языков Модула-2 и ассемблера привела к разным функциональными возможностям профилировщиков для них. Для ассемблерных программ отсутствует возможность работы с таким объектом, как процедура, но подсчитывается, сколько всего команд исполнено, а также время выполнения каждой команды (время — в тактах процессора). Для Модула-2-программ введена возможность работы с процедурами: подсчитывается не только число вызовов, но и время их счета, причем при нескольких вызовах — суммарное; среднее время одного вызова; минимальное и максимальное время вызова, а также в процентах собственное время исполнения программы без учета времен исполнения вызываемых из нее процедур. Для работы с объектами-процедурами нет нужды в расстановке точек для порождения событий. Для доступа к данным объектам используется возможность порождения событий “вызов процедуры” и “возврат из процедуры” при исполнении программы, что, как показывает опыт, позволяет получить эффективно работающие инструменты. Также для Модула-2-программ нельзя задать и такого объекта, как оператор, если это не строка, и подсчитать число выполненных операторов, а только лишь число выполненных помеченных строк.

Результаты работы представляются как в виде текстового файла, так и в виде диаграммы, на которой более наглядно, чем в текстовом виде, видны все аномалии.

В приложении приведены примеры работы профилировщика как для ассемблерных, так и Модула-2-программ.

7. ЗАКЛЮЧЕНИЕ

Таким образом, реализованный набор инструментов измерения динамических характеристик программ предназначен для:

— построения профиля программы с управлением по его детальности (точки в программе, для которых вычисляется число их исполнений, могут задаваться в диалоге произвольно: ставиться на все строки программы или на все входы в процедуры);

— вычисления времени исполнения как всей программы, так и любых ее фрагментов;

— подсчета количества итераций определенных фрагментов программы;

— подсчета числа обращений и/или изменений определенных глобальных переменных, и в случае их изменения при работе программы определяется минимальное и максимальное значения (фактически определяется диапазон изменения).

Описанные системы построения профиля ассемблерных и Модула-2-программ реализованы на IBM PC в среде Top Speed Modula-2 v.3.2. и XDS, являются частью диалоговых кросс-отладчиков KPOT и DVX, позволяют в процессе отладки и тестирования программы получить информацию о динамических характеристиках исследуемой программы. Данная информация, в свою очередь, дает возможность принять нужные решения либо о неизбежных ее доработках, либо необходимости более тщательного тестирования ее определенных частей.

СПИСОК ЛИТЕРАТУРЫ

1. **Knuth D. E.** An empirical study of FORTRAN programs // Stanford artifical intelligence project MEMO AIM-137. — 1971. — P. 42. — (Rep./ Computer science department; NSC-186).
2. **Поттосин И. В.** Система СОКРАТ: Окружение программирования для встроенных систем. — Новосибирск, 1992. — 20с. — (Препр. /СО РАН, ИСИ; № 11).
3. **Коган Б. И.** Экспериментальное исследование программ. — М.: Наука, 1988.
4. **Lurie D. , Vandoni C.** Statistics for FORTRAN identifiers and scatter storage techniques // Software Practice and Experience. — 1973. — Vol.3, №2. — P. 171—177.
5. **Alexander W. G., Wortman D. B.** Statistic and dynamic characteristics of XPL programs // Computer. — 1975. — Vol.8, №11. — P. 41—45.
6. **Grune D.** Some statistics on ALGOL 68 programs // SIGPLAN Notices. — 1975. — Vol.14, №7. — P. 38—46.
7. **Феррари Д.** Оценка производительности вычислительных систем. — М.: Мир, 1981.
8. **Липаев В. В.** Тестирование программ. — М.: Радио и Связь, 1986.
9. **Fitch J.** Profiling a large programm // Software Practice and Experience. — 1977. — Vol. 7, № 3, — P. 511—518.
10. **Shimasaki M, Fukaya S., Ikeda K, Kiyono T.** An analysis of Pascal programs in compiler writting // Software Practice and Experience. — 1980. — Vol. 10, №2. — P. 149—157.
11. **Захаров Л. А.** Организация средств тестирования и отладки в кросс-системе программирования // Среда программирования: методы и инструменты. — Новосибирск, 1992. — С. 68—79.
12. **xTech** Development System. Native XDS v 2.21 for IBM Operating System/2. User's Guide. — xTech Ltd., 1997.

Приложение

Пример 1. Пусть ассемблерная программа состоит из двух модулей и их имена — MOD1 и MOD2. Рассмотрим фрагмент модуля MOD1:

```
      VAR1 DC   H'2'                строка 10
|П    LAB1 L    R1,10               строка 11
|П         LA   R2,LO               строка 12
|П    LOP  ST   R1,VAR1             строка 13
|П         BCTR R1,R2               строка 14
      LAB2 ...                      строка 15
```

Строки 11–14 и переменная VAR1 — "отмеченные"профилировщиком. В модуле MOD2 "отмеченных"объектов нет. После того как данный фрагмент отработал, т. е. счетчик команд соответствует команде с меткой LAB2, профиль, сохраняемый во внешней памяти, будет иметь следущий вид:

— Файл с профилем —

Динамический профиль.

1. Информация о выполнении операторов.

Модуль	Метка	Строка	Количество исполнений	Время
MOD1	LAB1	11	1	3
		12	1	3
	LOP	13	10	20
		14	10	30
MOD2				

Всего исполнилось 23 оператора

2.Информация об использовании переменных.

Имя	Адрес	Обращений	min/max	
VAR1	131402	11	min:	00000000 0000001
			max:	00000000 0001010

— Конец файла —

Если "отмеченная"переменная имеет кратность (аналог массива) больше чем 1, то в файл записывается информация о всех "отмеченных"индексах.

Причем все они будут помечаться в выходном файле одинаковыми именами, но с разными адресами.

Пример 2. Ниже приведен файл с профилем программы DRY, написанной на Модуле-2 и состоящей из семи модулей, причем профиль строился для строк только одного модуля MYSTR.

— Файл с профилем —

Динамический профиль

1.Информация о выполнении операторов

Модуль	Процедура (метка)	Строка, участок	Количество исполнений	Время
START				
CONVERT				
DRY				
MYINOUT				
MYSTOR				
MYSTR	Length	7, 1	10000	40000
	Length	8, 2	0	0
	Length	8, 1	10000	20000
	Length	9, 2	300000	16030000
	Length	9, 1	10000	380000
	Length	10, 1	10000	2550000
	StrCompare	18, 1	5000	40000
	StrCompare	19, 1	5000	1230000
	StrCompare	20, 1	5000	1230000
	StrCompare	21, 1	5000	115000
	StrCompare	22, 1	0	0
	StrCompare	23, 1	5000	115000
	StrCompare	24, 1	0	0
	StrCompare	26, 1	5000	75000
	StrCompare	28, 1	100000	7300000
	StrCompare	29, 1	0	0
	StrCompare	31, 1	100000	2300000
	StrCompare	32, 1	5000	30000
	StrCompare	34, 1	95000	2185000
	StrCompare	35, 1	0	0
	StrCompare	38, 1	95000	1425000
	StrCompare	42, 1	5000	64092179
		45, 1	1	17
		46, 1	1	39981
SUPPORT				
TIME				

Всего исполнилось 770002 строки

Диаграмма профиля

модуль MYSTR

Length	7, 1	#|
Length	8, 2	
Length	8, 1	#|
Length	9, 2	################################
		##########################|[1]
Length	9, 1	#|
Length	10, 1	#|
StrCompare	18, 1	|
StrCompare	19, 1	|
StrCompare	20, 1	|
StrCompare	21, 1	|
StrCompare	22, 1	
StrCompare	23, 1	|
StrCompare	24, 1	
StrCompare	26, 1	|
StrCompare	28, 1	##################|
StrCompare	29, 1	
StrCompare	31, 1	##################|
StrCompare	32, 1	|
StrCompare	34, 1	##################|
StrCompare	35, 1	
StrCompare	38, 1	##################|
StrCompare	42, 1	|
	45, 1	!
	46, 1	!

Масштаб: один символ экрана "#"соответствует 5358 разам исполнения;
один символ экрана "!"менее чем 2679 разам исполнения;
символ экрана "|"добавляется при необходимости.

— Конец файла —

Некоторые пояснения к содержимому профиля: графа таблицы “Процедура” содержит имя той процедуры, которой принадлежит строка профилируемого модуля и номер которой указан в следующей колонке таблицы. “Меток” для программ на Модуле-2 не бывает, но если в программе есть ассемблерный модуль, то метки обязательно присутствуют, тогда в данной графе будет указана “метка” данного оператора. Если имя процедуры или метки длинное, то возможно его усечение при печати. Графа таблицы “Строка, участок” содержит номера строки исходного текста программы и участка, на которые она может разбиваться оптимизирующим кодогенератором, если он переставляет некоторые

[1]При печати данная строка была представлена двумя, поскольку ее длина превышала формат полосы.

фрагменты — участки кода. Последние в дальнейшем исполняются не в указанном в программе порядке, и поэтому на каждый такой участок ставится собственная точка останова и для каждого участка собирается вся необходимая информация. Диаграмма показывает графическое представление профиля для частоты исполнения строк. При этом, естественно, по вертикали расположены номера строк, а по горизонтали — сколько раз исполнялась данная строка, причем в таком масштабе, чтобы не выйти за экран монитора. Число исполнений обозначается тремя символами: !, |+, #. Определяется масштаб и символ (третий), соответствующий значению масштаба и являющийся основным при изображении диаграммы. Однако всегда есть строки с числом исполнений, не кратным и меньшим масштаба, для обозначения которых используются первый (число исполнений меньше половины масштаба) и второй (число исполнений равно половине масштаба) символы соответственно. Значение масштаба обычно свое для каждого модуля.

Если активирован режим подсчета времени процедур, то в конец данного файла допишутся таблицы 1 и 2 (или весь файл будет состоять только из них).

Если во втором столбце табл. 1 нет имени, то это тело модуля — нулевая процедура. Суммарное время исполнения — время исполнения от момента начала программирования до формирования отчета — обозначено в табл. 1 t; полное время исполнения данной процедуры — t; суммарное время исполнения процедуры без учета времени исполнения вызываемых из нее функций и процедур — t'; среднее время исполнения одного вызова процедуры — t; минимальное время исполнения одного вызова процедуры — t; максимальное время исполнения одного вызова процедуры — t.

Видно, что при построении процедурного профиля, дополнительно к сказанному ранее, подсчитывается максимальная глубина процедурного стека, которая также является важной характеристикой исследуемой программы.

Таблица 1

Процедурный профиль

Модуль	Процедура	Число вызовов	Время					$\frac{t'}{t}$, %
			t	$t'_{\text{сум}}$	$t_{\text{ср}}$	$t_{\text{мин}}$	$t_{\text{макс}}$	
START	X2C_BEGIN	1	252	252	252	252	252	0.0003
START	X2C_EXIT	1	252	252	252	252	252	0.0003
CONVERT		1	269	269	269	269	269	0.0003
DRY	Func1	15000	4350000	4350000	290	290	290	4.3838
DRY	Func2	5000	43295000	5510000	8659	8659	8659	5.5528
DRY	time	4	17504	1936	4376	4376	4376	0.0020
DRY	Proc3	5000	4715000	2915000	943	943	943	2.9377
DRY	Proc7	15000	5400000	5400000	360	360	360	5.4420
DRY	Proc6	5000	4265000	2880000	853	853	853	2.9024
DRY	Func3	5000	1385000	1385000	277	277	277	1.3958
DRY	Proc1	5000	17550000	6770000	3510	3510	3510	6.8226
DRY	Proc2	5000	2275000	2275000	455	455	455	2.2927
DRY	Proc4	5000	1270000	1270000	254	254	254	1.2799
DRY	Proc5	5000	1280000	1280000	256	256	256	1.2900
DRY	Proc8	5000	3155000	3155000	631	631	631	3.1795
DRY	Func4	15000	7440000	7440000	496	426	566	7.4979
DRY	Proc0	1	99223203	18208967	99223203	99223203	99223203	18.3506
DRY		1	99228186	1902	99228186	99228186	99228186	0.0019
MYINOUT	WriteLn	5	20935	6305	4187	4031	4316	0.0064
MYINOUT	WriteString	3	10137	10137	3379	2966	3674	0.0102
MYINOUT		1	1267	731	1267	1267	1267	0.0007
MYSTOR	ALLOCATE	2	660	660	330	330	330	0.0007
MYSTOR		1	271	271	271	271	271	0.0003
MYSTR	Length	10000	19010000	19010000	1901	1901	1901	19.1578
MYSTR	StrCompare	5000	36335000	17325000	7267	7267	7267	17.4597
MYSTR		1	269	269	269	269	269	0.0003
SUPPORT	Action	9	2286	2286	254	254	254	0.0023
SUPPORT	ClearBuffer	9	21762	21762	2418	2418	2418	0.0219
SUPPORT	GetLocBuffer	9	2286	2286	254	254	254	0.0023
SUPPORT		2	538	538	269	267	271	0.0005
TIME	GetTime	4	15568	3864	3892	3892	3892	0.0039
TIME		1	770	499	770	770	770	0.0005

Всего процедур исполнилось 31
Суммарное время исполнения (t) 99228433
Максимальная глубина вызовов 5

Таблица 2

Диаграмма собственного времени процедур (в процентах ко всему)

START	X2C_BEGIN	!
START	X2C_EXIT	!
CONVERT		!
DRY	Func1	####
DRY	Func2	######
DRY	time	!
DRY	Proc3	###
DRY	Proc7	#####
DRY	Proc6	###
DRY	Func3	#
DRY	Proc1	#######
DRY	Proc2	##
DRY	Proc4	#
DRY	Proc5	#
DRY	Proc8	###
DRY	Func4	#######
DRY	Proc0	##################
DRY		!
MYINOUT	WriteLn	!
MYINOUT	WriteString	!
MYINOUT		!
MYSTOR	ALLOCATE	!
MYSTOR		!
MYSTR	Length	##################
MYSTR	StrCompare	################
MYSTR		!
SUPPORT	Action	!
SUPPORT	ClearBuffer	!
SUPPORT	GetLocBuffer	!
SUPPORT		!
TIME	GetTime	!
TIME		!

Масштаб: один символ экрана "#"соответствует одному проценту (с округлением); один символ экрана "!"соответствует значению меньшему одного процента.

С. К. Черноножкин

СРЕДСТВА ПРОФИЛИРОВАНИЯ ПРОГРАММ В ПРОЕКТЕ СОКРАТ

Препринт
48

Рукопись поступила в редакцию 30.01.98
Рецензент И. В. Поттосин
Редактор Л. А. Карева

Подписано в печать 20.03.98
Формат бумаги 60×84 1/16 Объем 1,1 уч.-изд.л., 1,2 п.л.
Тираж 100 экз.

НФ ООО ИПО "Эмари" РИЦ, 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6