Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Ярославский государственный университет им. П.Г. Демидова
Кафедра компьютерной безопасности и математических методов
обработки информации

Связь разноязыковых модулей

Лабораторный практикум на ЭВМ

Методические указания

Рекомендовано
Научно-методическим советом университета
для студентов специальности Прикладная математика
и информатика

Ярославль 2006

ŋqdqɏeɎ bɆ dədɆbɎɜə d ɕɔɑbɔɏ Ɉqddɏqɏè

УДК 004.5
ББК В 18
С 24

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2006 года

Рецензент
кафедра компьютерной безопасности и математических методов обработки информации Ярославского государственного университета им. П.Г. Демидова

Составители:
О.В. Власова. Н.Б. Чаплыгина

Связь разноязыковых модулей : лабораторный практикум на ЭВМ : метод. укзания / сост. О.В. Власова, Н.Б. Чаплыгина; Яросл. гос. ун-т. – Ярославль : ЯрГУ, 2006. – 40 с.

С 24

Цель лабораторных работ по практикуму на ЭВМ – изучение архитектуры связи программы и подпрограммы, получение навыков компоновки программ из модулей, написанных на разных языках программирования: языке высокого уровня C++ или Pascal и машинно-ориентированном языке Ассемблере.

Методические указания предназначены для студентов 1-го курса математического факультета, обучающихся по специальности 010200 Прикладная математика и информатика (дисциплина «Практикум на ЭВМ», блок ОПД), очной формы обучения. Методические указания будут полезны и студентам других специальностей, интересующимся вопросами взаимосвязи разноязыковых программ.

Ил. 7

УДК 004.5
ББК В 18

Введение

Модульный подход к программированию является одним из видов структурного программирования, который наряду с объектно-ориентированным программированием широко применяется при проектировании и составлении программ. Основа модульной технологии состоит в разбиении задачи на более мелкие составляющие части, называемые модулями. Эти модули являются независимыми функционально и связываются между собой посредством передачи данных. Их можно создавать на разных языках программирования, связывая в единый выполняемый модуль на этапе компоновки. При выборе языка программирования высокого уровня существует возможность некоторые модули программировать на языке Ассемблера в целях повышения эффективности программы, уменьшения ее рабочего времени. Так, например, если в данной части программы используются многократно выполняемые циклические фрагменты или обращение к некоторым аппаратным средствам, которое затрудняется использованием языка высокого уровня, то данный модуль можно реализовать на машинно-ориентированном языке Ассемблере.

Существуют различные способы включения ассемблерного кода в программы. Большинство современных компиляторов с языков высокого уровня имеют специальные операторы, позволяющие прямо в тексте исходной программы делать ассемблерные вставки. Такое средство, называемое встроенным Ассемблером, позволяет использовать инструкции Ассемблера наравне с командами языка высокого уровня. Это дает возможность сочетать преимущества программирования на языке высокого уровня с эффективностью локальных фрагментов программ. В этом состоит одно из решений проблемы связи программ, написанных на разных языках. Его недостаток в том, что программирование вставки на Ассемблере в большой степени зависит от синтаксиса языка высокого уровня, от контекста программы.

Другое решение состоит в использовании внешних процедур или функций, написанных на разных языках. Можно написать на

Ассемблере отдельную функцию и вызывать ее из программы на языке высокого уровня, например C++ или Паскаль. Этот подход удобен тем, что функции создаются и отлаживаются независимо друг от друга, связываясь лишь на этапе компоновки. В этом случае функции имеют более универсальный характер и могут быть использованы другими программами, кроме того, могут быть легко изменены или заменены другими в случае необходимости. За раздельную компиляцию приходится платить затратами на разработку кода: программист, работающий с Ассемблером, должен вникать во все детали организации интерфейса между кодом C++ и кодом Ассемблера. В то время как при использовании встроенного Ассемблера Borland C++ сам выполняет спецификацию сегментов, передачу параметров, ссылки на переменные C++, и т.д. Отдельно компилируемые функции Ассемблера и C++ должны все это выполнять самостоятельно.

В представляемых методических указаниях рассмотрим обе возможности связывания частей программ, написанных на разных языках: языках высокого уровня (C++, Паскаль) и машинно-ориентированном языке Ассемблере. Кроме того, приведем пример непосредственного использования переменных из одного сегмента данных одновременно вызываемой и вызывающей разноязыковыми функциями, т.е. пример организации глобальных (внешних) имен в случае создания модулей на разных языках.

1. Вызов программой на языке C++ функции на Ассемблере

В интерфейсе Ассемблера и C++ есть два основных аспекта. Во-первых, различные части кода C++ и Ассемблера должны правильно компоноваться, а функции и переменные в каждой части кода должны быть доступны (если это необходимо) в других частях кода.

Во-вторых, код Ассемблера должен правильно работать с вызовами функций, соответствующими соглашениям языка C++, что включает в себя доступ к передаваемым параметрам, возврат значений различного типа, передачу управления в вызываемую программу и возврат управления из нее в вызывающую программу,

соблюдение правил сохранения информации в регистрах, корректную работу со стеком.

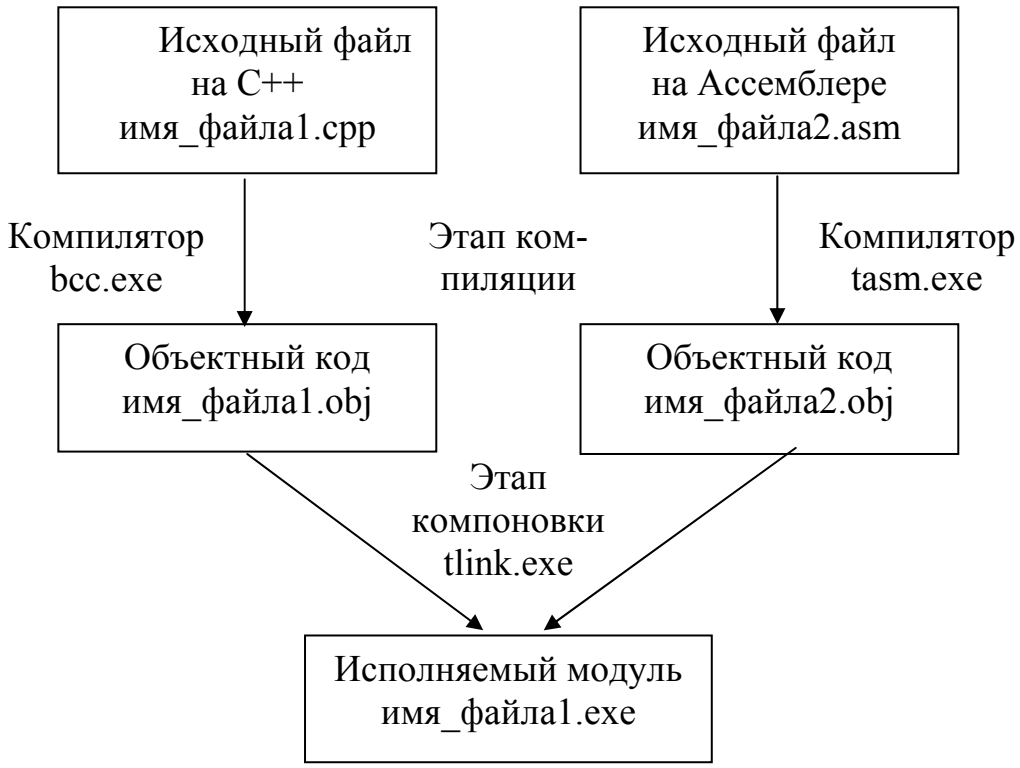

Рис. 1. Цикл компиляции, ассемблирования и компоновки программы, созданной на языках C++ и Ассемблере

Си и ассемблер используют одни и те же регистры процессора. Если в ассемблерной подпрограмме используется какой-либо регистр, то его содержимое необходимо сохранить в стеке и восстановить перед завершением функции. Если нарушить какое-либо соглашение о связи подпрограммы с вызывающей программой, то программа в момент выхода из подпрограммы может просто «вылететь» в неизвестном направлении.

Разберем связь модулей на конкретном примере.

Пример 1. Постановка задачи

1. Программа на языке C++ читает с клавиатуры двумерный целочисленный массив (матрицу) и передает его ассемблерной функции, с тем чтобы получить от нее линейный массив, составленный из максимальных по столбцам положительных элементов