Архитектура вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ПРАКТИЧЕСКОЙ РАБОТЕ НА ТЕМУ:

«Архитектура ВС уровня системы (набора) команд»

Работу выполнил:

студент группы БПИ207
Пендищук Владислав

Москва
2021 г.

ОГЛАВЛЕНИЕ

ОПИСАНИЕ ЗАДАНИЯ

Задача состояла в разработке программного продукта с использованием языка ассемблера NASM (Netwide Assembler) и применением библиотеки языка C для архитектуры x86-64.

Требования к функционалу

В соответствии с полученным вариантом задания (258) функционал, требуемый к реализации в программе, содержал в себе следующие пункты:

1. Реализация обобщённого артефакта – пассажирского транспорта, и его параметров:
    a. Скорость – целое число;
    b. Расстояние между пунктами отправления и назначения – действительное число;
2. Реализация базовых альтернатив и уникальных параметров, задающих их отличительные признаки:
    a. Самолёт, отличительные признаки:
        i. Дальность полёта – целое число;
        ii. Грузоподъёмность – целое число.
    b. Поезд, отличительные признаки:
        i. Количество вагонов – целое число.
    c. Корабль, отличительные признаки:
        i. Водоизмещение – целое число;
        ii. Вид судна – перечислимый тип (лайнер, буксир, танкер).
3. Реализация общей для всех альтернатив функции – вычисления идеального времени прохождения пути (действительное число).
4. Реализация контейнера для объектов типа обобщённого артефакта с массивом фиксированной длины в своей основе.
5. Реализация функции удаления из контейнера тех элементов, для которых значение, полученное с использованием функции, общей для всех альтернатив, меньше чем среднее арифметическое для всех элементов контейнера, полученное с использованием этой же функции.

Требования к запуску и вводу\выводу

К процессу запуска программы и ввода\вывода при работе с ней были представлены следующие требования:

1. Запуск программы осуществляется из командной строки, в которой указываются: имя запускаемой программы; имя файла с исходными данными; имя файла с выходными данными.

2. Для каждого программного объекта, загружаемого в контейнер, исходный файл с тестовым набором должен содержать: признак альтернативы, а также список параметров, необходимых этой альтернативе. Этот список должен быть представлен в формате, удобном для обработки компьютером.
3. При больших данных во входном файле должны быть указаны только параметры для генератора случайных наборов данных, который и заполняет контейнер.
4. В выходной файл необходимо вывести введенные в контейнер данные. Помимо этого, необходимо вывести информацию об общем количестве объектов, содержащихся в контейнере. После этого в тот же файл необходимо вывести новые данные в соответствии с результатами, полученными в ходе работы программы. Информация для вывода должна быть представлена в форме, удобной для восприятия пользователем.

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ПО

Исходный код программы содержится в 9 файлах. 8 из них являются модулями реализации (единицами компиляции с описанием процедур), один файл содержит в себе макроопределения, включаемые в модули реализации. Общий размер исходных текстов следующий:

	С форматированием и комментариями	*Без форматирования и комментариев*
Количество строк	1324	918

Размер исполняемого файла, полученного после компиляции кода на ОС Linux, равен 27072 байт (26,4 килобайт).

Результаты тестов с использованием тестовых файлов в соответствующих директориях следующие:

Тестовый файл	Тестовый кейс	Время работы программы
correct/test1.txt	5 корректных элементов	0,00014 секунды
correct/test2.txt	19 корректных элементов	0.0002 секунды
correct/test3.txt	13 корректных элементов	0.00019 секунды
error/test1.txt	Некорректное число аргументов в строке	0.00013 секунды
error/test2.txt	Некорректный идентификатор альтернативы	0.00015 секунды
error/test3.txt	Удалённая строка параметров	0.00016 секунды

Результаты тестов с использованием случайной генерации объектов контейнера следующие:

Количество элементов	*Время работы программы*
20 элементов	0.00018 секунды
1000 элементов	0.00574 секунды
10000 элементов	0.0985 секунды

СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА ПО

Ранее в ходе курса программа с идентичным функционалом была разработана на высокоуровневых универсальных языках программирования C, C++ и Python для статически и динамически типизированных архитектур ВС. Данная же реализация задачи была выполнена для бестиповой архитектуры ВС уровня набора команд процессоров Intel, чем обусловлено большое количество различий в характере реализации:

1. В отличие от разработанных ранее программ, в данной реализации нет типов данных. Вместо них данные содержатся в специально выделенных именованных областях памяти заданного размера, а навигация по ним осуществляется при помощи адресации и регистров процессора. В связи с этим процедуры (подпрограммы) в данной реализации не привязаны к каким-либо типам и способны принимать любые данные.
2. Все операции (за исключением случаев вызова функций библиотеки языка C) выполняются при помощи инструкций процессора Intel архитектуры x86-64, что в некоторых случаях повышает производительность ввиду прямого обращения к регистрам и памяти. В предыдущих же реализациях использовались исключительно высокоуровневые средства используемых языков программирования.
3. Разработанная программа, в отличие от предыдущих, наиболее сильно зависима от платформы ввиду специфики ассемблера NASM, его языка и используемой системы команд, присущей определённой архитектуре процессоров.

Сравнение характеристик текущей реализации с предыдущей также выявило ряд различий:

1. Объём исходных текстов в значительной мере увеличился ввиду использования низкоуровневого набора команд и системы доступа к памяти, подразумевающей ручное помещение данных в регистры и память.
2. Размер исполняемого файла уменьшился практически в два раза по сравнению с наименьшим ранее полученным размером (для реализации на языке C – 43,9 килобайт). В первую очередь это связано с бестиповым характером рассматриваемой архитектуры ВС и используемым языком программирования.
3. Время работы программы незначительно уменьшилось в большинстве случаев и незначительно увеличилось для генерации содержимого контейнера по сравнению с реализацией на языке C – самой производительной из ранее разработанных реализаций задачи. Это связано с тем, что в программе широко используются функции из

стандартной библиотеки данного языка и с тем, что компиляция кода на языке C сопровождается оптимизацией со стороны компилятора (например, компилятором gcc).

Тестовый файл	*Тестовый кейс*	*Время работы программы*
correct/test1.txt	5 корректных элементов	0.00021 секунды
correct/test2.txt	19 корректных элементов	0.00024 секунды
correct/test3.txt	13 корректных элементов	0.00022 секунды
error/test1.txt	Некорректное число аргументов в строке	0.00013 секунды
error/test2.txt	Некорректный идентификатор альтернативы	0.00019 секунды
error/test3.txt	Удалённая строка параметров	0.00016 секунды

Количество элементов	*Время работы программы*
20 элементов	0.00023 секунды
1000 элементов	0.00298 секунды
10000 элементов	0.13457 секунды

Рис. 1: время выполнения программы для реализации на языке C (процедурный подход)