УДК 004

О ЦЕЛЕСООБРАЗНОСТИ ПРАКТИЧЕСКОГО ПРИМЕНЕНИЯ ЯЗЫКА АССЕМБЛЕРА В СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Таратынов А.С.

студент,

ФГБОУ ВО «Калужский государственный университет

им. К.Э. Циолковского»

Калуга, Россия

Белаш В.Ю.

к.пед.н., доцент,

ФГБОУ ВО «Калужский государственный университет

им. К.Э. Циолковского»

Калуга, Россия

Аннотация: Язык ассемблера является одним из самых известных низкоуровневых языков программирования. Этот язык позволяет разработчику напрямую «общаться» с аппаратным обеспечением электронной вычислительной техники и максимально оптимизировать работу программы. В статье рассмотрены история возникновения языка ассмемблер, его виды, общий синтаксис и основные функции. Также выделены сложности при обучении работе с данным языком, его достоинства и недостатки и целесообразность практического приложения языка ассемблера в современных информационных системах (ИС).

Ключевые слова: язык ассемблера, низкоуровневые языки программирования, работа с языком ассемблера, программирование.

ON THE EXPEDIENCY OF PRACTICAL APPLICATION OF THE ASSEMBLY LANGUAGE IN MODERN INFORMATION SYSTEMS

Taratynov A.S.
student,
Kaluga State University named after K. E. Tsiolkovsky
Kaluga, Russia
Belash V.Yu.
Ph.D., Associate Professor,
Kaluga State University named after K. E. Tsiolkovsky
Kaluga, Russia

Abstract: Assembly language is one of the most famous low-level programming languages. This language allows the developer to directly "communicate" with the hardware of electronic computing equipment and optimize the operation of the program as much as possible. The article discusses the history of the origin of the assembler language, its types, general syntax and basic functions. The difficulties in learning to work with this language, its advantages and disadvantages and the usefulness of the practical application of the assembly language in modern information systems are also highlighted.
Keywords: assembly language, low-level programming languages, working with assembly language, programming.

В настоящее время существует множество языков программирования, способных выполнить любой спектр задач – от создания простого калькулятора до полноценных операционных систем с различными уровнями взаимодействия и свободы действий. Вместе с тем, в значительной части информационных систем до сих пор используются языки ассемблера, которые, с одной стороны, позволяют целиком и полностью контролировать работу процессора и других частей аппаратного обеспечения конкретной ЭВМ, с другой стороны – обладают некоторыми сложностями и нюансами в обучении и работе с ними.

Для того, чтобы оценить целесообразность подобного практического приложения языков ассемблера в современных информационных системах, сначала нужно рассмотреть его историю, понять, для чего и как он использовался, а также с какими сложностями при обучении и работе с данными языками может столкнуться программист.

Создание первых ассемблеров приписывается Кэтлин Бут для компьютеров ADC2 (Advanced Relay Calculator) в 1947 году [4] и Дэвидом Уилером для EDSAC (Electronic Delay Storage Automatic Calculator) в 1948 году, однако тогда конкретного термина «ассемблер» не существовало. Так, например, в загрузочной программе EDSAC находился ассемблер, который назывался «начальными командами». Также EDSAC использовал написанные Уилером однобуквенные мнемоники (например, для команды Add использовался понятный для компьютера код для буквы A). Таким образом, Уилер создал язык, который мог использоваться любителями для написания программ под EDSAC [2].

Признание же языки ассемблера получили на рубеже 70-80-ых гг. XX в., когда стали использоваться как основные языки для большинства популярных в то время домашних компьютеров, например систем из 8-битного семейства Atari, Apple II, MSX, ZX Spectrum и Commodore 64.

В настоящее же время языки ассемблера используются для прямой манипуляции с аппаратным обеспечением, а также для получения доступа к особым инструкциям процессора и исправления критических проблем с производительностью на аппаратном уровне.

В мире не существует такого ассемблера, который подошел бы к любому аппаратному обеспечению, ведь для каждой архитектуры процессора существует свой язык ассемблера, который она и только она понимает. Кроме того, любая программа-ассемблер может использовать свой, отличающийся от других синтаксис. Существуют также различные диалекты ассемблера, созданные в разные временные промежутки и обладающие разными наборами команд под

разные процессоры. Поэтому целесообразно рассмотреть общий синтаксис для любой программы на ассемблере.

Программа на ассемблере является совокупностью сегментов, которые всегда имеют определенное назначение. Это может быть сегмент кода, данных или стека. Каждый сегмент, в свою очередь, состоит из предложения (совокупности строк). Предложения могут быть следующих типов:

- команды;
- макрокоманды;
- директивы;
- комментарии.

Рассмотрим подробнее каждый из них.

Команды – это аналоги машинных команд, которые в процессе трансляции «переводятся» на язык, который понимает процессор.

Макрокоманды – это предложения текста программы, которые в процессе трансляции заменяются другими предложениями.

Директивы – это указания транслятору ассемблера на выполнение каких-либо действий.

Комментарии – это любые символы, в том числе символы русского алфавита, которые игнорируются транслятором [1].

В ассемблере не существует присущих высокоуровневым языкам программирования отступов и операторных скобок. Код записывается в несколько колонок, которые обычно включают в себя:

- адрес инструкции (необязательная колонка);
- метки – указатели на места программы, на которые можно производить переход, вызов процедуры и т. д., которые в процессе сборки преобразуются в адреса;
- мнемокод;
- операнды – регистры, константы, адреса ячеек памяти и т. д.;

- комментарии.

Так, любая команда языка ассемблера обычно выглядит следующим образом:
[метка [:]] КОП [Операнд] [,Операнд] [;Комментарий] (где КОП – мнемокод, краткое представление команды).

Директива, в свою очередь, представляется следующим образом:
[Идентификатор] КПОП [Операнд] [,Операнд] … [;Комментарий] (где идентификатор – имя директивы, а КПОП – её мнемокод).

На уровне машинных кодов программы не имеют никакой структуры и из одного места в программе может осуществиться переход в другое независимо от того, где началось выполнение программы. Программа просто продолжит свое выполнение с того места, куда был произведен переход.

Используя все эти инструменты и знания, можно создать программу на любом диалекте ассемблера и под любую архитектуру процессора, которая позволит производить любые манипуляции с аппаратным обеспечением компьютера напрямую.

К основным сложностям при обучении и работе с ассемблерами относится тот факт, что, как уже говорилось, для каждой архитектуры и режима работы процессора существует свой язык ассемблера, который они понимают. Так, например, в руководстве по разработке программ под архитектуру Intel IA-32 содержится более 1000 страниц, посвященных описанию всех кодов инструкций для семейства процессоров Intel Pentium [3]. Отсюда исходят и высокие требования к программисту: он должен досконально изучить характеристики и требования конкретной машины, прежде чем начать разработку программы.

Также к сложностям при работе относится и то, что собственно программирование и отладка кода на ассемблере требует больше усилий, т. к. смысл того или иного значения и действия, которые над этим значением допустимо

проводить, контролируются самим программистом. Программист всегда должен обладать внимательностью и хорошей памятью, ведь на любом языке ассемблера требуется постоянно пользоваться стеком (список элементов, построенный по принципу LIFO – “Last In, First Out”) и ограниченным набором команд общего назначения.

Отсюда можно сделать вывод, что перед началом работы с любым из языков ассемблера программист должен быть готов столкнуться с трудностями, преодолевать которые ему придется самому, используя свои навыки и предоставленную разработчиком конкретного аппаратного обеспечения документацию.

К преимуществам использования языков ассемблера относятся:

- возможность полного контроля работы конкретной части аппаратного обеспечения (процессора или памяти);
- возможность полного контроля над кодом;
- возможность максимальной оптимизации работы (по размеру и скорости) программы на конкретном аппаратном обеспечении;
- возможность их использования для вставки в программы, написанные на высокоуровневых языках, которые не могут обеспечить выполнение той задачи, которая под силу языку ассемблера;
- программы, написанные на языке ассемблера, используют минимальное количество памяти.

К недостаткам использования языков ассемблера относятся:

- низкая продуктивность, ведь любую программу на языке ассемблера можно более ёмко и легко создать на каком-либо из высокоуровневых языков программирования;
- отсутствие какой-либо структуризации кода, которое влечет за собой низкую его читаемость;
- полное отсутствие кроссплатформенности;

- сложность для использования при совместной работе.

Можно сделать вывод, что использование языков ассемблера при программировании обладает рядом достоинств и недостатков, и лишь самим программистом, исходя из них, определяется необходимость его использования в конкретных случаях.

В настоящее время языки ассемблера применяются в следующих случаях:

- разработка программ для микроконтроллеров и адаптеров различных периферийных устройств;
- разработка собственного компилятора, виртуальной машины, драйвера и т. п.;
- быстрая разработка определенных частей операционной системы (загрузчик, BIOS, файловые системы);
- отладка, исследование и разработка различного рода программ;
- разработка вредоносного ПО.

Несмотря на всё вышеперечисленное, использование языков ассемблера в современных ИС крайне ограничено. Так, например, при создании загрузчиков UEFI, используемых вместо BIOS в современных ИС, востребован язык С, в котором необходимость написания кода на «чистом» языке ассемблера сведена к минимуму. Помимо этого, в каждом из ныне используемых языков программирования уже существуют встроенные, более простые и понятные для ознакомления и работы инструменты для отладки, а создание программ в них не является столь трудоемкой, долгой и кропотливой задачей. Однако, как уже было замечено, языки ассемблера все еще можно и нужно применять, когда конкретная часть программы на языке более высокого уровня требует этого.

Исходя из вышесказанного можно сделать вывод о том, что практическое приложение языка ассемблера в информационных системах является лишь частично целесообразным.

Библиографический список

1. Assembler. Учебник для вузов. 2-е изд. / В. И. Юров – СПб.: Питер, 2003. – 637 с.
2. IEEE Computer Society: David Wheeler. [Электронный ресурс]. – Режим доступа: URL: https://www.computer.org/profiles/david-wheeler
3. Richard Blum. Professional Assembly Language. / Richard Blum – Indianapolis: Wiley Publishing, 2005 – 577 p.
4. Wayback Machine: General Considerations In The Design Of An All Purpose Electronic Digital Computer. [Электронный ресурс]. – Режим доступа: URL: https://web.archive.org/web/20200324161441/http://mt-archive.info/Booth-1947.pdf

Оригинальность 86%