6 Назначение и принципы работы соглашений ABI для архитектуры IA-32

архитектура IA-32 (Intel Architecture 32-bit) - система архитектур процессора, разрабатываемая компанией Intel. использует 32-разрядные вычисления. архитектуру часто называют i386 (по названию первого выпущенного на ней процессора) и x86-32 (по применяемому набору команд).

ABI - Application Binary Interface (двоичный интерфейс приложений) - набор соглашений для доступа приложения к операционной системе и другим низкоуровневым сервисам, спроектированный для переносимости исполняемого кода между машинами, имеющими совместимые ABI. в отличие от API, который регламентирует совместимость на уровне исходного текста, ABI можно рассматривать как набор правил, позволяющих компоновщику объединять откомпилированные модули компонента без перекомпиляции всего исходного текста, в то же время определяя двоичный интерфейс

ABI регламентирует:

- использование регистров процессора
- состав и формат системных вызовов и вызовов одного модуля другим
- формат передачи аргументов и возвращаемого значения при вызове функции
- спецификацию типов данных (размеры, формат, последовательность байт)

- форматы исполняемых файлов
- соглашения о вызовах
- формат и номера системных вызовов

ABI описывает функциональность, предоставляемую ядром ОС и архитектурой набора команд (без привилегированных команд). если интерфейс программирования приложений разных платформ совпадает, исходный текст для этих платформ можно компилировать без изменений. если для разных платформ совпадают и API, и ABI, исполняемые файлы можно переносить на эти платформы без изменений. Если API или ABI платформ различаются, исходный текст требует изменений и повторной компиляции. API не обеспечивает совместимости среды выполнения программы — это задача двоичного интерфейса.

регистр - небольшой объем быстрой памяти, размещенный на процессоре. предназначен для хранения промежуточных результатов вычислений и информацией для работы процессора. поскольку регистры размещены на процессоре, время доступа к данным в них очень низкое по сравнению с оперативной или внешней памятью.

регистры делятся на пользовательские и системные регистры.

стек - часть динамической памяти, которая используется при вызове функций для хранения аргументов и переменных. регистр SP указывает на текущую вершину стека, а BP на базу стека, которая используется для разделения стека на логические части, относящиеся к одной функции - фреймы (кадры). в x86 архитектурах вершина стека имеет самый маленький адрес.

как программа будет работать с процессором, определяется соглашениями о вызовах.
соглашение о вызовах - это схема, в соответствии с которой вызов функции высокоуровневого языка программирования реализуется в исполняемом коде.
соглашение о вызовах решает:

- как передаются аргументы и возвращаются значения? в регистрах, через стек, и так и так?
- как распределяется работа между вызывающей и вызванной стороной?

для архитектуры IA-32 соглашения определены документом SYSTEM V APPLICATION BINARY INTERFACE Intel386 Architecture Processor Supplement Fourth Edition.

эти соглашения используются как в языке ассемблера, так и в языке C (для него эти соглашения обозначаются аббревиатурой cdecl), что позволяет в одном исполняемом модуле использовать функции, написанные на обоих языках и, тем самым, повышать эффективность и производительность программ. большинство современных реализаций других языков также поддерживают правила соглашения ABI о
вызовах.

Основные элементы функции:

- Вызывающая функция (caller) — функция, выполняющая команду call.
- Вызываемая функция (callee) — функция, которой передает управление команда call.
- Параметры функции – единицы входных или выходных данных, значения которых должна получать, обрабатывать и возвращать функция. Вызывающая

функция должна записать их значения в аппаратный стек.

- Имя функции – символьное имя, которое при ассемблировании получает значение адреса первой команды функции (точка входа в функцию).
- call — команда передачи управления в точку входа в функцию.
- Адрес возврата – адрес команды, следующей за командой call, которая при выполнении вычисляет его и записывает в аппаратный стек.
- ret — команда, которая должна быть выполнена последней в вызываемой функции. Она читает из стека адрес возврата и передает управление в вызывающую функцию на этот адрес.
- По соглашениям после завершения работы функции удалить ее параметры из стека должна вызывающая программа
- Код пролога — две стандартные команды, которые должны быть выполнены первыми при входе в функцию.
- Код эпилога — две стандартные команды, которые должны быть выполнены в функции непосредственно перед выполнением каждой командой ret.
- Коды пролога и эпилога используют регистры %ebp и %esp, обеспечивая вместе с командами call и ret произвольную глубину вложенности вызовов функций.

Порядок записи параметров функции в стек:

Пусть запись функции с n параметрами на языке С имеет вид: $f(P_1, P_2 \ldots P_n)$. По соглашениям о вызовах, значения параметров нужно передавать стек в порядке **ОБРАТНОМ** представленному в этой записи в следующей последовательности:

```
pushl Pn
pushl Pn-1
...
pushl P1
```