План:

1. Определение

2. Архитектура

3. Структура

3.1. Регистры предикатов

3.2. Кэш-память данных и инструкций 1ого и 2ого уровней

3.3. Устройство предварительной подкачки данных

3.4. 3 стека регистров

4. Возможности

4.1. Режим безопасных вычислений

4.2. Использование кода под Intel

5. Характеристики

6. Примеры

7. Области эффективного применения

8. Подведение итогов

9. Выводы

Итак, микропроцессоры «Эльбрус». Сегодня о них говорят очень много так, что каждый из нас хотя бы раз слышал это название. Давайте попробуем разобраться с тем, что же это такое. … для начала дадим определение, потом посмотрим на его архитектуру, после посмотрим на структуру. В ней будет сделан акцент на регистре предикатов, разделении регистров на стеки, кэш-памяти и устройства предварительной подкачки данных. Далее рассмотрим некоторые возможности. Потом посмотри на характеристики, приведем примеры микропроцессоров, рассмотрим области применения. И в конце перечислим достоинства и недостатки и на основе их сделаем выводы.

Начнем с определения.

Эльбрус - Процессорная архитектура и одноимённое семейство универсальных VLIW-микропроцессоров.

Нам встретилось незнакомое слово VLIW. А значит, самое время переходить к архитектуре процессора.

VLIW – название архитектуры процессора, как CISC или RISC. Расшифровывается VLIW – Very long Instruction Word, то есть очень длинное командное слово. Идея VLIW базируется на том, что задача эффективного планирования параллельного выполнения нескольких команд возлагается на «разумный» компилятор. Такой компилятор вначале исследует исходную программу с целью обнаружить все команды, которые могут быть выполнены одновременно, причем так, чтобы это не приводило к возникновению конфликтов. В процессе анализа компилятор может даже частично имитировать выполнение рассматриваемой программы. На следующем этапе компилятор пытается объединить такие команды в пакеты, каждый из которых рассматривается как одна сверхдлинная команда.

Другими словами, если у вас есть программа, в которой есть следующие действия:

- покормить кота

- погладить кота

- напечатать доклад

- ответить на письмо в аутлуке

То компилятор, анализируя ваш код, сделает вывод, что такие действия, как напечатать доклад и ответить на письмо можно параллельно. Также как, кормя кота, вы можете его погладить. И он объединит их в 2 разных пакета, создав 2 сверхдлинных инструкции.

Если у вас одна команда дробиться на 2коротких микрооперации, то вы можете занять в 2 раза больше ресурсов для выполнения этой команды. Если правильно свой код организуете.

С архитектурой разобрались, теперь переходим к структуре

И сразу замечаем, что появляется какие-то новые регистры. Регистры предикатов. Это регистры, в которых записывается либо 0, либо 1. Это нужно для условия.

У вас есть какое-то условие, псевдокод которого приведен на слайде. Эльбрус загружает обе части условия в память. (и код при верном условии и код при неверном условии). И как только выполнилось условие «Меня слушают» он записывает в регистр предикатов 1. И когда надо определиться с тем, какой код сейчас выполнять, он не думает, а считывает значение из регистра предиката значение и выполняет нужный участок кода.

То есть у вас нет простоя процессора при выполнении переходов. Плюс процессор может готовиться к прыжку параллельно с исполнением других команд. Это возможно благодаря тому, что компилятор знает, что происходит во всей проге.

В структуре есть несколько уровней кэш-памяти. Для чего нужна кэш-память для ускорения выполнения операция. Чтобы не обращаться к оперативной памяти и не делать простои в работе процессора. Более подробно нам об этом, скорее всего, расскажут позже и на другом предмете, поэтому и я углубляться в эту тему пока не буду.

Дальше мы видим устройство для предварительной подкачки данных. Оно также необходимо для уменьшения простоя процессора при обращении к оперативной памяти.

Если нам нужно на экран вывести содержимое нескольких файлов. То мы каждый раз будем обращаться к памяти, получать файл выводить содержимое и ждать, пока не получим следующий файл. Решение: заранее подкачивать.

И в эльбрусе эту функцию выполняет отдельный модуль, но при условии, что вы корректно написали программу и компилируете ее с – fprefetch (опция, включающая предподкачку).

Еще в процессорах Эльбрус все регистры распределены на 3 стека

PS – стек, PSC – стек, US – стек.

Стек процедур, стек связующей информации, стек пользователя. И программист имеет доступ только к US стеку, к остальным он никогда не имеет доступа. За то какие данные хранятся и как они хранятся в этих 2х регистрах отвечает компилятор.

Для упрощения представим, что программист – человек, который только смотрит видео. Пусть, программа, позволяющая запускать вам видео, VLS хранится в регистре 1. В регистре 5 хранятся данные этой программы. То есть видео, которое вы хотите запустить.

При запуске фильма, процессор сначала обращается к ячейке 1, запуская программу, а потом к ячейке 5, запуская в этой самой программе конкретный фильм. И адреса программы и адреса данных этой программы хранятся в PS стеке и к ним никто не имеет доступа.

Что касается PCS стека, то тут хранится информация о том, как вы пользовались данными в этой программе. В нашем случае: скорость воспроизведения, наличие субтитров, какой язык и прочие настройки.

Раз уж мы начали говорить о безопасности, давайте поговорим о возможностях, которые предоставляет микропроцессор, а точнее компилятор.

Первое – режим безопасных вычислений.

Эльбрус умеет компилировать приложения в защищенном режиме. И в таком режиме прога не может вылезти за предел тех данных, с которыми ей явно разрешили работать. Если вы сказали программе работать с массивам размером 256 бит, то при попытке подать ей массив 257 бит, будет выдана ошибка и работа прекратиться.

Плюс здесь в том, что ваш софт будет сложнее взломать. Не получится подать один вид данных под видом других.

Но в это же время стоит понимать, что во многих пользовательских программах есть ошибки. И данный режим будет использоваться в очень узкой сфере, где ошибок оприори не может быть, никаких.

Второе – оптимизация под интел.

Если у вас есть оптимизация кода, разработанная под Интел, то она сработает и на Эльбрусе. То есть вы пишите код на ассемблере для одного процессора, а он работает на другом процессоре. Вам может показаться, что Эльбрус – продукт Интел, только под другим именем, поэтому и работает. Но нет, все дело в компиляторе LCC. А дело в том, что компилятор Эльбруса LCC может находить для низкоуровнего кода для процессора Интел находить аналогичные команды для процессора Эльбрус и вставлять их при компиляции.

Характеристики

Примеры

- Эльбрус 8С

- Эльбрус 8СВ

- Эльбрус 16С

Области эффективного использования:

|  |  |
| --- | --- |
| Расширенный температурный диапазон, возможность локализации производства | Государственный заказ, промышленные компьютеры, автомобильная электроника |
| Повышенная защищенность от вирусных атак | Платежные терминалы, сетевые экраны, взломоустойчивые серверы |
| Высокая производительность на криптографических алгоритмах | Модули шифрования, защищенные тонкие клиенты, прочие системы безопасности |
| Высокая производительность на вычислениях с действительными числами (float, double) | Робототехника, авионика, промышленные контроллеры, системы обработки изображений, суперкомпьютеры |
| Работа под управлением бинарного компилятора в режиме совместимости с архитектурой х86 | Интернет-терминалы, маломощные рабочие станции, малогабаритные настольные и встраиваемые компьютеры |
| Защищенный режим | Особо ответственные системы, отладочные стенды |

Достоинства:

1. Это действительно разработка, а не простое копирование западного аналога

2. Конкурентно способен

То есть производительность и скорость может быть такой же или даже лучше, чем у процессоров Интел и AMD

3. Возможность установки различных ОС

Недостатки:

1. Производство располагается на Тайване.

2. Требуется высокая квалификация программистов.

Для того, чтобы ваша программа работала максимально эффективно, она должна быть хорошо написана.

3. Не все ОС будут работать с требуемой скоростью