Продолжение. Начало в № 9'2003.

Стековые процессоры, или Новое — это хорошо забытое новое

Иосиф Каршенбойм

iosifk@narod.ru

Применение стековых процессоров

Очень часто в телеконференциях задается один и тот же вопрос: «Надо бы что-то сделать на микропроцессоре, какой посоветуете взять?» Причем половина отвечающих предлагает PIC от Microchip, а другая половина советует применить AVR от Atmel. И очень редко бывает, что кто-то задает вопрос: «А для решения какой именно задачи вы хотите применить микропроцессор и какие критерии выбора для вас будут определяющими?»

Есть много критериев для выбора пути решения проекта, и трудно предложить какое-либо однозначное решение. Все зависит от того, что и как нужно сделать, а главное, какую именно задачу будет выполнять данное устройство. Причем, если речь идет не о разовой разработке-поделке, а о серийном изделии, то разработчику хочется заложить в свое изделие наиболее новые технические решения, что позволит устройству быть конкурентоспособным как можно дольше. Таким образом, разработчики встроенных систем сталкиваются с такими проблемами, с которыми неоднократно сталкивались разработчики вычислительных систем: они нуждаются в микропроцессоре или микроконтроллере, который показывает хорошую производительность на тех задачах, для решения которых он будет оптимален, имеет высокую плотность кодов команд и при этом потребляет малую мощность. Архитектуры компьютера со сложным набором команд достигают высокой эффективности по использованию памяти, и они просты для программирования, но им иногда не хватает производительности. Чипы RISC просты и мощны, но, часто они слишком «громоздки» для маленьких встроенных приложений. Что же кроме RISC можно применить для повышения эффективности встроенной системы? Возможно, что стековые процессоры — ключ к решению проблемы.

Микропроцессоры в ближайшем будущем

Как-то автор провел исследование у себя дома и выяснил, что он использует по крайней мере 9 микропроцессоров: в аэрогриле, СВЧ-печи, стиральной машине, телевизоре, видеомагнитофоне, телефоне, радиоле, и еще пару штук в ПК.

Это исследование отражает «вчерашнее» предложение на потребительском секторе рынка. А «завтра» в этом секторе рынка ожидается устойчивый рост, так как из сектора ПК в потребительский сектор перемещаются такие приложения, которые требуют большой производительности (например трехмерные игры). В случае использования ПК только для игр они оказываются слишком дорогими и слишком «трудными» в использовании, чтобы получить широкое применение.

Кроме игровых приставок появились и другие домашние приложения, например, интернет-серфинг и цифровое телевидение. Идеальный вариант состоит в том, чтобы иметь отдельный специальный блок для каждого применения:

- игровой пульт, чтобы запускать трехмерные игры;
- блок для доступа в Интернет;
- блок для цифрового телевидения;
- блок, который выдает запрос на цифровой видеомагнитофон;
- DVD-проигрыватель.

Все эти блоки имеют одну общую черту: они требуют большую мощность центрального процессора для того, чтобы декодировать потоки MPEG или отрисовывать трехмерную графику. Все эти функциональные возможности могут быть объединены в одном блоке, и для всех этих задач такой блок может и не быть собственно ПК, потому что ни одно из этих приложений не требует доступа к жесткому диску, но, в то же время, игры, цифровой видеомагнитофон и DVD-плейер должны быть способны читать (и писать) DVD-диск.

Есть также и другой аспект, почему ПК не используется вместо этих блоков: вы не можете иметь дома только один такой блок. Представьте себе ситуацию, когда дети хотят включить видеоигры, а папа хочет посмотреть спортивные состязания по TV, которые мама не выносит...

Таким образом, к блоку, а соответственно и к процессору, могут быть предъявлены следующие требования:

1) низкая цена;
2) высокая эффективность обработки данных, особенно цифровая обработка сигналов и трехмерная графика;
3) блок связи с Интернетом должен выполнять программы на Java;
4) малые объемы памяти (иначе блоки не будут достаточно дешевыми).

Те же требования предъявляются и к электронной начинке современного автомобиля.

Число микроконтроллеров, используемых в быту, увеличивается. А это значит, что так же, как и в про-

мышленности, наступает пора объединения бытовых приборов в сети. Добавление функций любому микроконтроллеру увеличивает объем кодов программ и усложняет отладку. Именно поэтому для работы и отладки такие микроконтроллеры потребуют программной поддержки в виде языка высокого уровня, да еще не просто языка высокого уровня, а языка, ориентированного на сетевые приложения. Аналогичный процесс происходит и в промышленности. Если внимательно приглядеться, то можно увидеть, что увеличивается потребность в устройствах, напоминающим по функциям ПК, но имеющим значительно меньшую цену. Это PDA, торговые терминалы и системы, работающие через Интернет, а также «пластиковые деньги». А это, в свою очередь, приведет к увеличению потребности в микропроцессорах, оптимизированных для выполнения команд языка Java.

Си и Форт

Все началось с того, что с распространением вычислительных машин начались интенсивные поиски путей разработки языков высокого уровня для того, чтобы сделать труд программиста более производительным.

Немного о Си

Хотя Forth и C были задуманы почти в одно и то же время, их происхождение значительно отличалось. Си, с его братом-близнецом UNIX, был рожден в одном из оплотов «академии информатики» — Bell Labs, и его автор, Деннис Ричи, в последующих годах был поднят почти к состоянию святости. В течение долгого времени качественные Си-компиляторы было доступны только в среде UNIX, в то время как UNIX был слишком дорог для коммерческих пользователей. Компания AT фактически отдала Си университетам. Таким образом, многие поколения студентов были воспитаны на UNIX и Си, и язык Си сделался для них «истинной верой».

О рождении Форта

В отличие от Си, Форт — это язык, который, выражаясь фигурально, развивался от «уровня травы». Он пробивал себе дорогу в мир, опираясь только на свои собственные достоинства. Язык Форт был изобретен Чарльзом Муром, внештатным программистом, который писал программы для обработки данных на фабрике по производству ковров в Stanford Linear Accelerator Center. Мур был знаком со многими языками программирования, включая ассемблер, ФОРТРАН, КОБОЛ, PL/1, APL, но чувствовал, что все они не вполне отвечали его потребностям. Он экспериментировал с различными трансляторами, и в начале 1960-х он пришел к идее использовать стек, чтобы передавать параметры и результаты между интерпретируемыми командами.

Мур считает, что его первый интерпретатор, в котором уже можно было узнать Форт, был написан на Фортране для IBM 1130, во время его работы на фабрике по производству ковров (Mohasco Industries) в 1968 году. Несколько позже Мур поменял место работы — он поступил Национальную астрономическую радио-обсерваторию (NRAO) и перенес Форт на Honeywell 316. Язык довольно быстро развился. NRAO не оценил значение Форта, и после некоторого предварительного исследования его патентоспособности передал все права на язык Муру. В 1973 Мур, Элизабет Ратэр (второй Форт-программист), и еще несколько человек сформировали FORTH Inc., чтобы продавать Форт как коммерческое изделие, первоначально названное miniFORTH, а позднее — polyFORTH. Вскоре Форт был установлен на ряд процессоров, из которых наиболее важным был PDP-11 от фирмы DEC. Процессор PDP-11, известный у нас как «Электроника-60», оставался основной машиной FORTH Inc. для пользователя на долгое время.

В течение следующих пяти лет понятие «язык Форт» и «Форт», как продукт, поставляемый фирмой FORTH Inc. были тождественны. Не было других коммерческих продавцов Форт-систем, и фактически само название — Forth было торговой маркой.

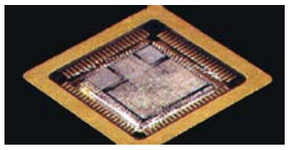
Хочется напомнить один эпизод из истории развития языка Форт. В 1978 году маленькая группа энтузиастов сформировала Forth Interest Group (FIG). Была собрана группа программистов, которая за девять месяцев разработала простую и переносимую на различные процессоры версию языка Форт. Это версия языка, называемая FIG-Forth, и была распространена во всем мире в течение нескольких месяцев. Затем Форт-системы были установлены на Intel 8080, DEC PDP-11, TI 9900, 6502, PACE и Motorola 6800. Впоследствии Форт-системы были установлены на Intel 8086, Nova, RCA 1802, Motorola 68000 и 6809, VAX и на многих других процессорах. Остальную часть описания исторического развития языка Форт приходится опустить, поскольку рамки журнальной статьи не позволяют привести всю историю целиком. Просто хочется подчеркнуть, что FIG действовала по своей собственной инициативе, не ждала указания правительства и не искала богатого спонсора. Когда автор данной статьи предложил в телеконференции отечественным разработчикам микропроцессорных устройств высказаться по поводу создания микропроцессора, наиболее удовлетворяющего их потребностям, то чего только не пришлось прочитать в ответ. Видимо, нет еще у нас в стране курортов, подобных калифорнийским, где можно без помех собраться и обсудить назревшие проблемы.

Далее необходимо отметить некоторые захватывающие события, которые произошли «аппаратном фронте». Было разработано несколько центральных процессоров с системами команд языка Форт, включая Novix NC 4016 Чарльза Мура, MF1600 Алана Винфилда и WISC CPU/32 Фила Купмана (Writable Instruction Set Computer — компьютер с перезаписываемой системой команд). Проект чипа Novix был в конечном счете продан фирме Harris Semiconductor, где он был переспроектирован в RTX 2000 (Фотографии микропроцессоров RTX 2000, Дофин 1610, 1620 и 1630 любезно предоставлены Алексеем Чепыженко, alex@radiopage.by).

RTX 2000 (Harris Semiconductor, 1987-89)

RTX 2000 — Real Time eXpress (Экспресс реального времени) 2-микронный кристалл 16-разрядного микроконтроллера был задуман как развитие NC 4016. На кристалле появились два стека по 256 вложений, умножитель, канал ввода-вывода. RTX 2000 выпускался серийно и активно применялся во встраиваемых системах для управления и обработки сигналов.

Форт в СССР и России



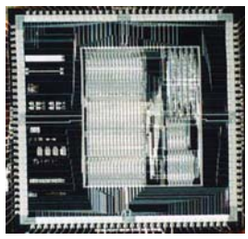
В СССР велись работы по созданию языка программирования для стековых машин. Были разработаны и сами машины типа «Эльбрус». Но если переместиться «ближе к микропроцессорам», то можно начать со следующего исторического периода.

Начало перестройки запомнилось первыми разноцветными автобусами с броскими рекламными надписями. Первые кооперативы, продающие продукты отечественного, заметьте, производства — джинсы-варенки и вафельные трубочки. Кто теперь помнит хотя бы их названия, и кто из них уцелел? Но вот один из первых отечественных кооперативов, кооператив «Форт-Инфо», организованный в 1988 г. и преобразованный в 1991 году в инновационно-техническую фирму «Технофорт» (www.technoforth.ru) — производитель программного обеспечения и новейших разработок в области вычислительной техники и изделий электронной техники — жив и до сих пор и успешно продолжает свою деятельность.

Фирма «Технофорт» является высокотехнологической компанией, работающей в области программирования и производства микропроцессоров, микропроцессорных устройств и систем. По сегодняшней терминологии «Технофорт» можно также назвать фаблесс-фирмой, выпускающей полупроводниковые микросхемы. Один из первых проектов фирмы в этой области — разработка Форт-микропроцессора. В результате трехлетней работы в 1991 г. был создан самый быстрый в СССР микропроцессор — 16-разрядный Форт-процессор Дофин-1610. Предназначенный для систем реального времени, он превосходил производимые в то время в СССР микропроцессоры по быстродействию в 50 раз и выпускался опытными партиями в минском НПО «Интеграл». Основными заказчиками фирмы до 1995 года были предприятия Санкт-Петербурга, в первую очередь ВПК, стремившиеся применить новые отечественные устройства на базе Дофин-1610 в своих разработках и системах автоматизации. Была спроектирована новая, усовершенствованная архитектура процессора — Дофин-1620 (совместно с минской фирмой НКТ, организованной в 1992 г. ИТФ «Технофорт» совместно с минскими коллегами). «Технофорт» выполнил ряд НИРовских

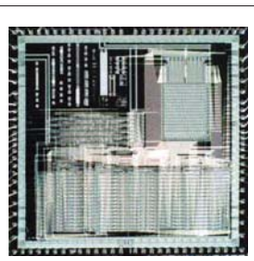
работ, на основании которых было подготовлено решение о постановке Форт-процессора на серийное производство в Зеленограде. К сожалению, экономический кризис и фактический упадок ВПК привели к свертыванию этих работ.

Дофин-1610 (1990)



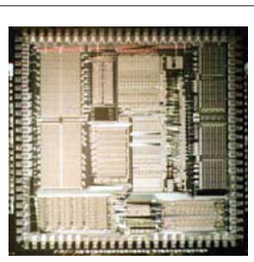
Советский вариант NC 4016. Разработчики повторили известный микропроцессор фирмы Novix, пользуясь при этом лишь спецификацией на архитектуру и систему команд. В результате получился совместимый кристалл, показавший хорошие параметры быстродействия для своего времени. Характерной особенностью Дофин-1610 стало использование дополнительной шины для подключения памяти стеков параметров и возвратов.

Дофин-1620 (октябрь 1994)



Во втором процессоре были добавлены внутренние стеки и урезаны шины для подключения внешних стеков. Кроме того, в него введена развитая система прерываний и аппаратный умножитель 16×16. В результате получился процессор, очень близкий по архитектуре с RTX 2000. Кристалл выполнен по КМОП-технологии 2,2 микрона и заключен в 100-выводной корпус PQFP.

Дофин-1630 (май 1997)



Совершенно оригинальная разработка. Главной целью была поддержка DSP и языка Си. Для этого была использована мультистековая архитектура. К фортовским стекам добавились второй стек данных и стек адресов. Введены дополнительные методы адресации через стек адресов и стек возвратов. К сожалению, поддержка Форта (по сравнению с Дофин-1620) пострадала.

Необходимо отметить, что во все, даже наиболее сложные периоды своей деятельности, коллектив ИТФ «Технофорт» продолжал и продолжает проводить научно-исследовательские работы. Наиболее крупная из них — разработка нового процессора. Программное моделирование было завершено еще в 1997 г. Появление технологии ПЛИС позволило без больших затрат получить конечный продукт — семейство высокопроизводительных однокристальных микроконтроллеров на ядре Форт-процессора TF16 собственной разработки. Отметим, что ядро TF16 и комплект периферии к нему, так же разработанный фирмой «Технофорт», позволили получить такое конкурентоспособное изделие, как кассовый регистратор. О конструктивных особенностях Форт-процессора TF16 будет рассказано в следующих разделах данной статьи.

Форт — язык, среда разработки, операционная система и философия программирования

Как сложить 2 и 3 или Сначала было «слово».

Каждый элемент языка Форт, который может использоваться в интерактивном режиме или в пределах программы, называют словом. Исходный текст для специфического слова называют его определением. Некоторые слова являются примитивами, они закодированы на родном машинном языке центрального процессора, остальные слова — вторичные определения или определения высокого уровня, закодированные непосредственно в Форте. Формально соответствующее определение выглядит следующим образом:

```
: имя тело ;
```

- «имя» как раз и есть новое придуманное слово;
- «тело» представляет собой перечень через пробелы уже имеющихся в Форт-системе слов, совокупность их функций образует те действия, которые будут выполнены при исполнении данного слова;
- наличие слова «;» (точка с запятой) обязательно, оно завершает определение.

Ключевое различие между Фортом и другими структурными языками — то, что компилятор Форта является расширяемым. Форт позволяет программисту объявлять новые типы данных и определять новые ключевые слова компилятора, а затем использовать их. Но Форт — не просто набор ключевых слов и правил синтаксиса, это — описание виртуальной машины. Самый важный аспект виртуальной машины — то, что она имеет два стека: стек параметров (или стек данных) и стек возвратов. Почти все слова Форта используют стек параметров и для аргументов, и для результатов; стек возвратов используется для потока управления в пределах программы и (иногда) для временной памяти рабочих значений. Именно из-за такой архитектуры на основе стека Форт является языком естественно рекурсивных алгоритмов. В программировании на «классическом Форте» глобальные переменные не желательны и редко используются.

Чтобы определить виртуальную машину Форта и поведение интерпретатора (компилятора) по существующим стандартам Форта, определяют менее чем 200 названий и действий. Коммерческие системы разработки программ на Форте намного сложнее, они имеют не менее 500 резидентных слов. Фактически, 500–600 элементов языка — почти то же самое, что и общее число операторов, ключевых слов и функций библиотеки Microsoft C или Borland C++.

Есть два аспекта Форта, которые часто критикуются — его загадочные названия и его постфиксный синтаксис. В те времена, когда были разработаны определения фундаментальных команд Форта, телетайпы на 10 символов в секунду были так же обычны, как и миникомпьютерные системы с 8 или 16 кбайт оперативной памяти. Следовательно, короткие названия были очень желательны для того, чтобы сохранить память и сэкономить нажатия клавиш, поэтому в языке применяются символы типа @ для выборки из памяти или! для записи в память, и т. д.

Инфиксная нотация, например, в Си выглядит так:

```
2 + 3
```

В системах постфиксным синтаксисом аргументы предшествуют оператору и, например, чтобы сложить числа 2 и 3, в Форте нужно писать так:

```
2 3 +
```

Почему Форту «нужно» именно так, будет показано далее, но для предварительного ознакомления можно посмотреть диаграмму работы стековой машины, в верхней части рис. 6 на зеленом поле (см. начало статьи в «КиТ» № 9'2003). Для того чтобы обработать две переменные — Var-1 (в нашем случае это число — 2) и Var-2 (в нашем случае это число — 3) выполняются следующие действия: сначала одна, затем вторая переменные заносятся на вершину стека, после чего над ними выполняется требуемая операция, результат которой заносится в определенное место памяти. То есть машина производит именно эти действия и именно в этом порядке. Поэтому такое написание позволило сократить затраты на интерпретаторы и компиляторы для Форта.

Например, текст

```
: S2 DUP * SWAP DUP * + ;
```

определяет слово «S2», вычисляющее сумму квадратов двух чисел из стека

```
a b --> a*a+b*b
```

Скомпилированные слова сразу же могут использоваться и в вычислениях, и в определении других слов. Например, сумму четырех квадратов можно определить так:

```
: S4 S2 -ROT S2 + ;
a b c d --> a*a+b*b+c*c+d*d
```

Можно отменить уже определенное слово («забыть» его), но при этом забываются также и все слова, определенные позже него. Для этого используется слово «FORGET». Например, действие

```
FORGET S2
```

«забудет» S2 и все определенные позже слова.

Итак, на возражения о постфиксном синтаксисе можно сказать следующее: легко читаемые и удобные в сопровождении программы могут быть написаны на любом языке, так же как могут быть написаны и неудобные в сопровождении программы. Тайна получения качественного программного продукта — это хороший проект, дисциплина и документация, а не использование специфического языка.

Форт — это среда разработки и операционная система

Форт-система в основном написана на самом языке Форт. Она занимает от 8 до 16 кбайт в зависимости от предоставляемых возможностей (таких, как встроенный ассемблер, экранный редактор, взаимодействие с файловой системой).

Программы на языке Форт реентерабельны, допускают рекурсию. Форт является одним из самых быстрых и эффективных языков программирования и широко используется в системах реального времени и специализированных приложениях. Такие программы обычно пишутся на «высокоуровневом» Форте. Однако можно значительно ускорить их выполнение, переписав интенсивно используемые слова в машинных кодах. Программист может написать программу в машинных командах на встроенном в Форт-систему ассемблере и в дальнейшем использовать ее как обычную подпрограмму. Вследствие этого, Форт можно применять для создания программ непосредственного управления аппаратурой. Новое определение создается по форме

```
CODE <имя-слова> <ассемблерная программа> END-CODE
```

Ассемблерная программа представляет собой запись операторов машинного кода в обратной польской записи. Определенное таким образом слово вызывается и выполняется подобно Форт-слову. Оно может работать со значениями из стека, что позволяет передавать аргументы так же, как в Форт-словах. Основное отличие состоит в том, что слово «CODE» устанавливает контекст словаря ассемблерных мнемоник «ASSEMBLER». В этом же словаре имеются ассемблерные версии структур управления (условных операторов и циклов). Преимуществом Форт-ассемблера является его расширяемость и «встроенность в Форт». Внутри ассемблерного определения можно воспользоваться определением через «:» как макрокомандой; можно обратиться к переменной.

Форт, как говорилось выше, является интерпретирующей, диалоговой, объединенной средой. В типичных системах редактор, ассемблер и различные утилиты отладки написаны непосредственно на Форте и всегда сохраняются резидентно. Исходный текст для этих инструментальных средств обычно сохраняется интерактивно, так, чтобы программист мог настроить его по своему личному вкусу. Хотя Форт может быть выполнен многими способами, «классическое» выполнение представляет собой объединенный транслятор и возрастающий компилятор.

Во встроенных приложениях Форт, конечно, продолжает использоваться в его «родном» воплощении, чтобы уменьшить аппаратные требования до минимума. Специальный тип компилятора Форт, названный целевым компилятором, выполняется на абсолютной системе развития и создает автономное выполнимое приложение (которое может быть помещено в ROM). Это приложение может быть загружено в целевую систему любыми удобными средствами, например из ROM, EPROM, либо по последовательному каналу связи.

Философия Форта

Можно сказать, что основная идея, реализованная в Форте — это малые требования к ресурсам, простота и скорость. Программисты, работающие на Форте, традиционно презирают роскошные интерфейсы пользователя и сложную обработку ошибок. Этот стиль работы имеет свои определенные выгоды — так, например, вся интерактивная система Форт, включая редактор, ассемблер и дополнительные средства многозадачности может поместиться в программируемое ПЗУ объемом всего 8 кбайт. Но это также имеет и свои недостатки, поскольку проверка ошибок во время выполнения в Форт-системе фактически отсутствует. Частые полные системные сбои во время разработки программы на Форте считаются само собой разумеющимися, поскольку они ожидаются. И еще одна особенность философии Форта — это простой доступ к аппаратным средствам.

При разработке проекта на языке Форт обычно производится «восходящее» программирование. В начале разработки проекта определяется большинство аппаратно-зависимых слов, затем определяются слова более высокого уровня, оперирующие аппаратно-зависимыми словами, и, затем, определяются слова верхнего уровня, которыми описывается задача пользователя. Таким образом, если слова всех нижних уровней определены, а они, как правило, редко требуют любого существенного изменения, то проектирование определений верхнего уровня позволяет легко изменять проект до тех пор, пока не будут получены определенные функциональные возможности и требуемый интерфейс пользователя.

Novix

Приступая к рассмотрению принципов работы микропроцессора Novix, я не буду писать о том, что Novix «и теперь живее всех живых». Но посмотрим на его старшего брата MCS-51 — он живет и здравствует именно потому, что оказался первым и, к тому же, идеи, заложенные в него, не устарели и до сих пор. То же можно сказать и о Novix. Он был первым, под его архитектуру написано достаточно много ПО, он был повторен во многих последующих разработках. Novix NC4016, первоначально называвшийся NC4000, является 16-разрядным стековым микропроцессором, разработанным для выполнения примитивов языка программирования Форт. Он предназначен для выполнения программ управления объектами в режиме реального времени и для высокоскоростного выполнения языка Форт для универсального программирования. Поскольку уровень технологии производства микросхем того времени не позволял разместить все требуемые ресурсы для архитектуры микропроцессора на одном кристалле, то часть ресурсов пришлось разместить на дополнительных микросхемах. У более поздних клонов Novix этот недостаток был устранен, и сейчас Novix-подобные ядра представляют собой «монолитную» структуру, пригодную для размещения даже в FPGA.

Блок-схема и описание архитектуры Novix

К сожалению, г-н Купман не дал разрешение на использование блок-схемы микропроцессора NC4016 в нашей статье. Поэтому читатели, желающие увидеть эту блок-схему, смогут найти ее в книге «Stack Computers» [1].

Целью проекта при реализации микропроцессора было немедленное выполнение большинства примитивов языка Форт в единственном машинном цикле без внутренней микропрограммы. NC4016 был разработан и реализован по 3-микронной технологии вентильных матриц HCMOS, и в нем было использовано менее 4000 вентилей. Тактовая частота — до 8 МГц. Эта технология не позволяла размещать на кристалле блоки памяти стека. Поэтому чип NC4016 выпускался в 121 корпусе pin-grid, и минимальная система для NC4016 состояла из трех 16-разрядных блоков памяти: один для программ и данных, один для стека данных и один для стека возврата. Для связи с внешними микросхемами памяти и периферии используются шины, которые могут быть сгруппированы в пять различных функциональных групп:

- адресная шина и данные из оперативной памяти;
- адресная шина и данные, расположенные в стеке данных;
- адресная шина и данные, расположенные в стеке возвратов;
- порты ввода-вывода;
- синхронизация и управление.

Внутренняя структура NC4016 разработана для выполнения команды процессора в одном тактовом цикле. Все примитивные операции, кроме выборки данных из памяти, записи в память и выборки длинного литерала, выполняются в единственном тактовом цикле. За один цикл также выполняются команды IF, ELSE и LOOP. Это требует, чтобы на кристалле находилось несколько больше линий связи, чем должно быть в канонической стековой машине, но такая архитектура обеспечивает намного лучшую производительность. Для достижения высокой эффек-

тивности по передаче данных NC4016 может использовать четыре отдельные 16-разрядные шины для передачи данных на каждом тактовом цикле (память программы, стек данных, стек возврата и шины ввода-вывода). Микропроцессор может производить одновременный доступ к стеку возврата, стеку данных, оперативной памяти и портам ввода-вывода, параллельно с операцией арифметико-логического устройства и сдвигающего устройства.

NC4016 позволяет объединять несколько неконфликтующих последовательных операций в одну команду. Более подробно о таком режиме работы стековой машины будет рассказано далее на примере работы микропроцессора TF16 в разделе описания команд.

Другое новшество NC4016 — механизм, позволяющий обратиться к первым 32 адресам памяти программ как к глобальным «пользовательским» переменным. Этот механизм позволяет избежать проблем, связанных с работой языков высокого уровня, позволяя сохранять ключевую информацию для задачи в быстродоступной переменной, например указатель на вспомогательный стек в оперативной памяти. Это также обеспечивает достаточную производительность при использовании компиляторов языка высокого уровня, которые, возможно, были первоначально разработаны не для стековых машин, а для машин с регистрами, позволяя использовать 32 переменные с быстрым доступом для того, чтобы смоделировать набор регистров.

Блок арифметико-логического устройства (АЛУ) содержит буфер с 2 элементами для главных элементов стека данных (T для верхнего элемента стека данных и N для второго от верха элемента стека данных). Он также содержит специальный регистр MD для поддержки умножения и деления, а также регистр SR для быстрого вычисления целочисленных квадратных корней. АЛУ может выполнить операции на регистре T и любом из регистров N, MD или SR.

NC4016 связывается с оперативной памятью через 16 линий адреса, 16 линий данных, используя сигнал разрешения записи WED. Таким образом, непосредственно адресуемая память составляет 64 Кслова или 128 кбайт. Пространство памяти может быть расширено, если 5 строк X-порта ввода-вывода (порт расширения) используются как дополнительные линии адреса, чтобы управлять оперативной памятью. Учитывая использование порта расширения адреса (X-порт) странично-адресуемая память может быть расширена до 2 Мслов или 4 Мбайт. Адресация непосредственно одного байта не поддерживается.

Механизмы работы стековой машины требуют не менее двух стеков, один — для того, чтобы хранить адреса возвратов из подпрограмм, другой — чтобы хранить параметры для обмена данными между подпрограммами.

В NC4016 каждый стек использует 16 строк данных, 8 строк адреса и сигнал разрешения записи. Так как шины адреса имеют разрядность 8 бит, то глубина стеков ограничена 256 словами во внешней памяти стека. Стек данных — память, расположенная вне кристалла. Указатель вершины стека данных расположен на чипе и обеспечивает адрес стека для памяти, находящейся вне кристалла. Отдельная 16-разрядная шина данных стека позволяет читать или писать в стек данных параллельно с другими операциями. Два верхних элемента стека данных буферизированы регистрами T и N в АЛУ.

Стек возвратов представляет собой отдельную память, которая очень похожа на стек данных, за исключением того, что только верхний элемент стека возвратов буферизирован на кристалле в индексном регистре. Так как Форт сохраняет значение данных счетчиков циклов так же, как адреса возврата подпрограммы в стеке возвратов, то индексный регистр может быть декрементирован для того, чтобы эффективно осуществлять циклы счета в обратном направлении.

На кристалле микропроцессора нет аппаратной защиты стека от переполнения или опустошения. Так как большинство микросхем памяти имеет емкость более 256 байт, то можно использовать линии X-порта для того, чтобы переключать блоки стеков странично. Можно сделать переключатель страниц стека в виде отдельного регистра, находящегося вне кристалла, и он может управляться как порт ввода-вывода. Это очень полезно при поддержке многозадачной системы, в которой каждая задача имеет свои собственные данные и стеки возврата. Это также является аппаратной защитой данных разных задач — данные не будут записаны поверх стека другой задачи. Переключение задач в этой среде чрезвычайно быстро, так как эксплуатационные режимы каждой задачи полностью сохраняются в их индивидуальном месте стека и это так же до минимума уменьшает перекачку данных при переключении контекста.

NC4016 имеет два порта ввода-вывода: B-порт на 16 бит и X-порт на 5 бит. Эти два порта полностью программируются пользователем через 4 внутренних регистра для каждого порта: регистр направления, определяющий направление передачи данных каждым битом, регистр маски, регистр перевода бита в третье состояние и регистр данных считывания данных со входов или записи данных на выходы. Оба порта могут делать операции ввода-вывода в единственном машинном цикле.

Система команд Novix

Описание системы команд микропроцессора NC4016 представляет собой довольно объемный материал и здесь приводиться не будет. Желающие ознакомиться с этими материалами могут обратиться к книге Ф. Купмана [1]. Пример системы команд стекового процессора будет приведен далее при описании процессора TF16.

NC4016 был предназначен для применения во встроенных устройствах управления. Он имел очень высокую эффективность при разумно маленьких аппаратных затратах. NC4016 был первоначально разработан как прототип, поэтому он имеет некоторые недочеты, которые были исключены в последующих проектах стековых машин.

Окончание следует.

Литература

1. Купман Ф. Stack Computers. http://www.ece.cmu.edu/~koopman/stack.html.