Архитектура, CISC и RISC.

Предыстория

Мы уже говорили ранее (в главе ”От арифмометров к электромеханическим компьютерам”) о том, как экстенсивное развитие в середине XX века привело к тупику. Компьютеры всё усложнялись, развитие технологии полупроводников приводило к увеличению тактовой частоты и в конце концов тактовую частоту стало невозможно увеличивать т.к. из-за ограниченности скорости света возникли ограничения на физические размеры компьютеров. Подвох в том, что при частоте в 300 мГц за один такт свет успеет пролететь только 1 метр. Следовательно, физически удалённые устройства процессора могут оказаться в разных тактах. Вентили перестанут работать синхронно. Речь о больших и сложных системах, так называемых мэйнфреймах (mainframe).

Интересно, что выход из технологического тупика разработчики архитектур мэйнфреймов искали в векторных конвейерных вычислителях. Векторные вычисления производятся с целой линейкой (вектором) регистров, если, например, требуется умножить вектор из 4 элементов на два, параллельно производятся сразу четыре умножения. Это позволяло пусть и экстенсивным путём, на специализированных задачах, но поднять пиковую производительность.

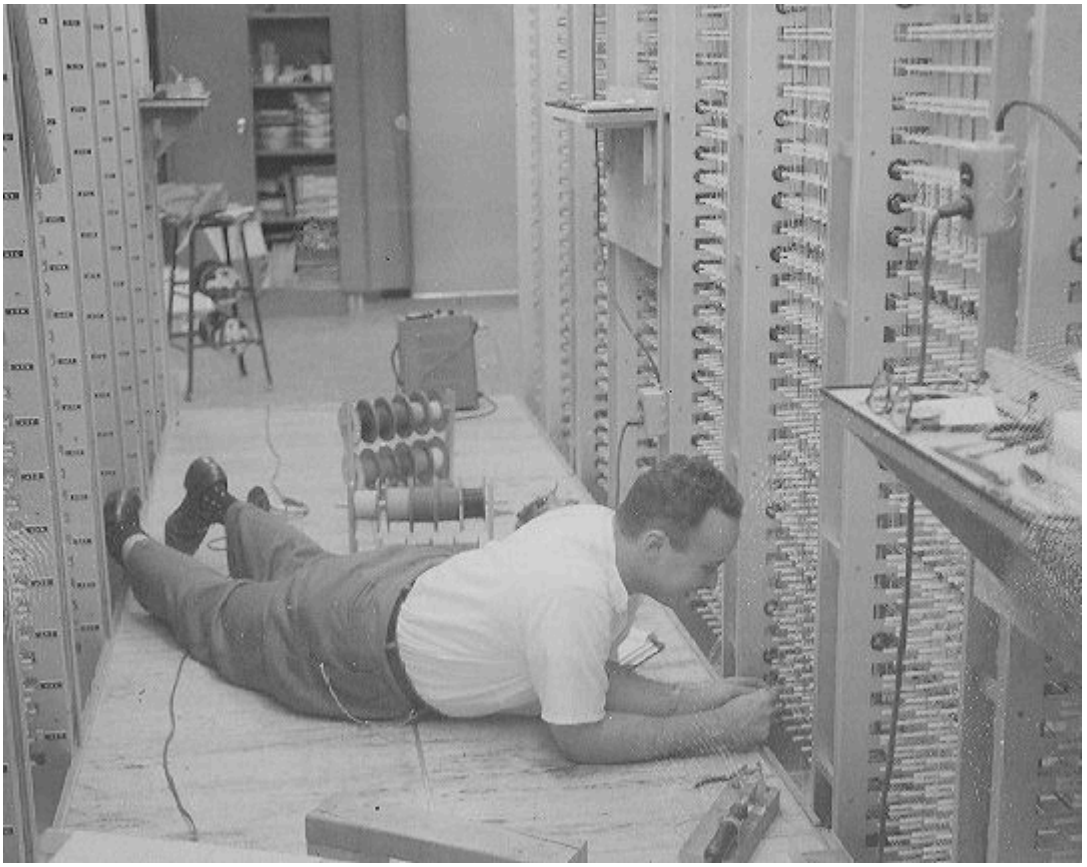
На другом конце спектра производительности тоже происходили интересные события. В 60-е годы появилась целая плеяда мини-компьютеров. Навскидку:

- HP-2100 появился в продаже в 1966, выпущены десятки тысяч экземпляров, продукты этой серии сделали HP крупнейшим производителем миникомпьютеров и выпускались вплоть до 1990 г. Доступные языки программирования - диалекты FORTRAN-а, ALGOL, BASIC. И ассемблер, конечно, куда же без него.
- DDP-516 - также 1966. Его вариант Honeywell H316 известен как “кухонный компьютер”, в его память был зашит список кулинарных рецептов, а в корпус встроена разделочная доска. Между прочим, именно H316 использовался в качестве основного компьютера для мониторинга температуры реактора АЭС Bradwell вплоть до 2000 года.
  Доступные языки - BASIC, FORTRAN, FORTH.
- МИР - 1965, для инженерных расчетов, основной язык - АЛМИР-65
- IBM-7094/7074 - 1962/1963 развитие IBM-7090 (первый транзисторный мэйнфрейм). Языки программирования - FORTRAN, COBOL
- Наири 1964
- Сетунь 1962
- Минск 1963
- PDP-1/4/.../10 1960...1969 Языки программирования FORTRAN, FOCAL, DIBOL, BASIC

- PDP-11 (1970) стоит особо, он 16-разрядный и в нём появился аппаратный стек

Кроме миниатюризации шло и удешевление техники, если IBM-7090 стоил миллионы (тех ещё) долларов, то “кухонный компьютер” H316 уже около 10 000. В результате взрывным образом выросла доступность компьютеров, если число мэйнфреймов исчислялось сотнями, то мини-компьютеры продавались десятками тысяч.

Появилась физическая возможность небольшим коллективам разрабатывать и выпускать мини-компьютеры, что в свою очередь вызвало острую потребность в переносимом программном обеспечении. Ведь операционные системы в то время писались на ассемблере, языки высокого уровня не были приспособлены для системных задач. Впрочем, это уже был прогресс по сравнению с недавним прошлым, вот так, например, выглядела отладка на R1 с памятью на CRT(1960)



Фиг.1 Отладка на R1 (отсюда)

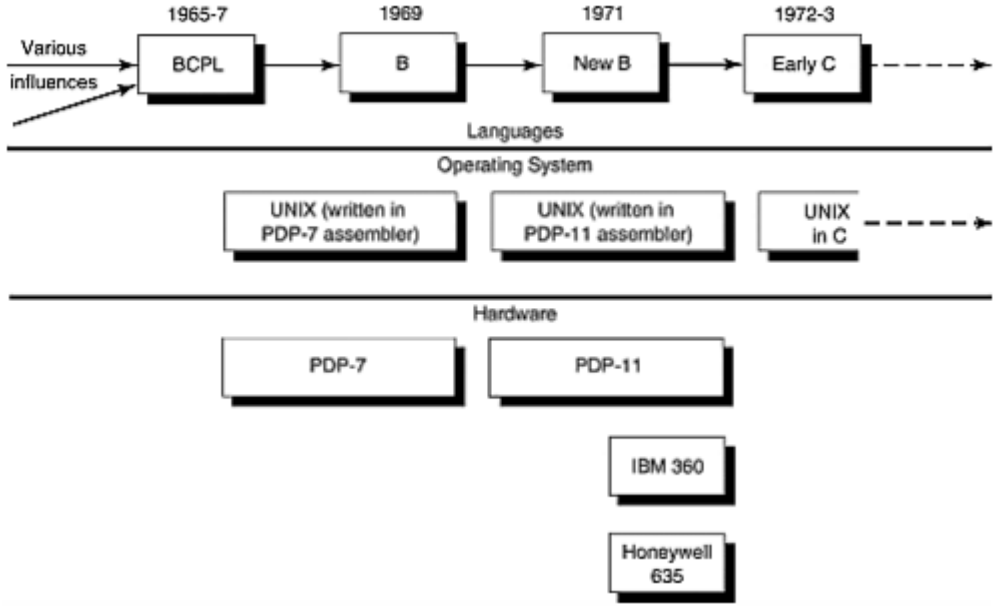
В то же время на более мощных компьютерах операционные системы достигли определённой зрелости. Стоит упомянуть совместный проект Bell Labs, GE и MIT под название Multics. Multics была задумана как операционная система, реализующая целую кучу революционных идей

- вытесняющая многозадачность
- официальное разделение памяти процессов на внутреннюю (оперативную) и внешнюю (файлы)
- централизованная файловая система
- сегментно - страничная виртуальная память

- динамическое связывание - разделяемые библиотеки
- многопроцессорность
- встроенная система безопасности
- конфигурация ОС на-лету
- в основном написана на языке высокого уровня (PL/1)
- …

В общем, всё то, чего мы сейчас интуитивно ожидаем от операционной системы, впервые в совокупности появилось именно в Multics.

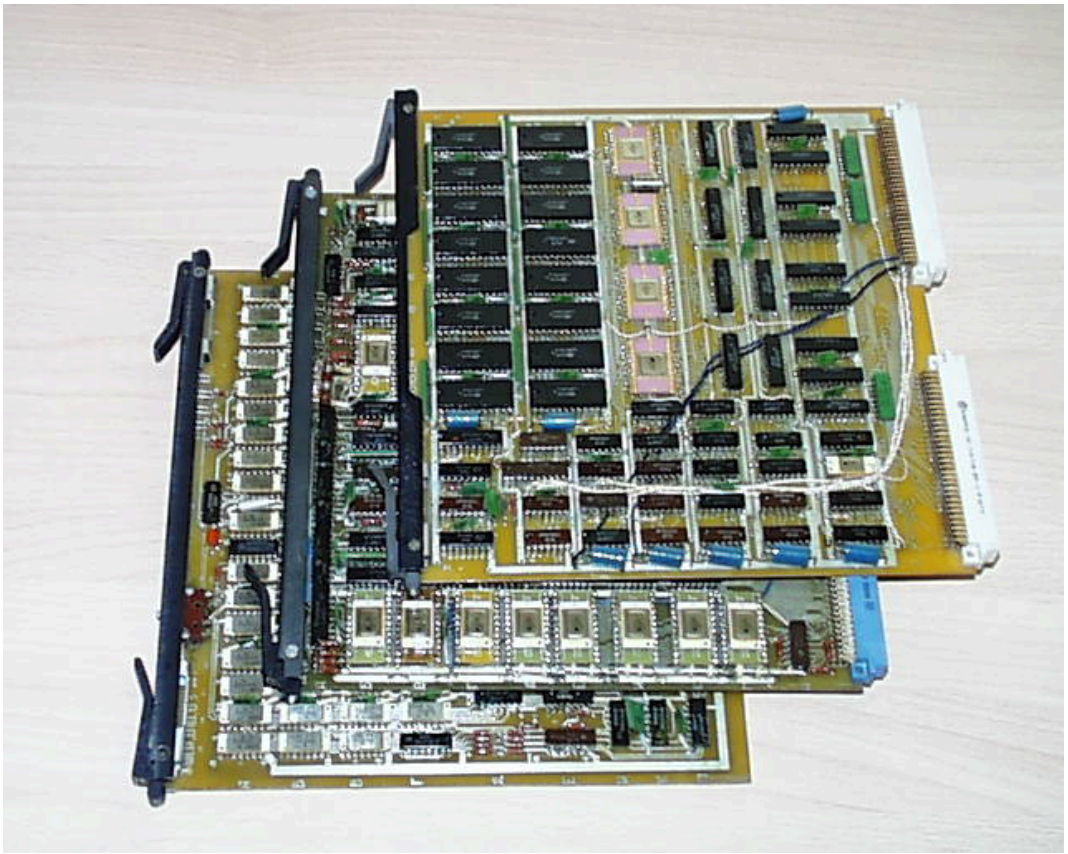
К сожалению, данный проект, как и многие до и после него, был погребён под собственной сложностью. Произошло это в 1969 году и на остатках Multics, благодаря поначалу в основном Кену Томпсон и Деннису Ритчи из Bell Labs, появились ОС UNIX и язык программирования C.



Фиг.2 Происхождение C (отсюда)

Невозможно удержаться еще от того, чтобы привести еще одну фотографию.

Фиг.3 Кен Томпсон (сидит) и Деннис Ритчи мучают PDP-11 в Нью-Джерси (отсюда)



Фиг.3.1 экспериментальный процессор Кронос П2.6, так и не появившийся в однокристальном виде (отсюда)

Тем временем шел и невероятно быстрый прогресс в элементной базе. Первыми доступными **однокристальными** микропроцессорами стали 4-разрядные Intel

4004 (1971) и TMS1000 (1972) от TI. Первым 8-разрядным микропроцессором стал Intel 8008 (1972), а 16-разрядными, по видимому, HP BPC (1975) и TMS9900 от TI (1976). К первым 32-разрядным микропроцессорам можно отнести MC68000 от Motorola (1979), который, впрочем, имел 16-разрядную шину данных и АЛУ. И, наконец, первым 64-разрядным микропроцессором стал MIPS R4000 (1991). Intel i860 появился в 1989 г, но имел 32-разрядное АЛУ и 64-разрядный вычислитель с плавающей точкой (который, впрочем, умел работать с 64-разрядными целыми числами). i860 не был всё же процессором общего назначения, поэтому первенство по праву забирает именно R4000.

Итого, к середине-концу 80-х была доступна целая линейка честных 32-разрядных однокристальных микропроцессоров - Motorola MC68020 (1982), DEC-J11 (1983), Intel i386 (1985), NS32032 (1985), MIPS R2000 (1986), Fairchild Clipper (1986), SPARC V7 (1987), MIPS R3000 (1988), Motorola 88K (1988), AMD29K (1988), Intel i960 (1989).

Далее подтянулись MIPS R4000 64 (1991), HP PA-RISC 64 (1992), SPARC-V8 32(1992), PowerPC 32/64 (1993), DEC Alpha AXP 64 (1993), SPARC-V9 64 (1995)...

Что можно сказать об оригинальных советских архитектурах того периода?
Из микропроцессоров можно вспомнить проект КРОНОС (1984...1990), созданный в рамках проекта МАРС. Проект дошел до стадии работающего прототипа, но, увы, не был реализован в виде однокристального микропроцессора. Имел стековую архитектуру, ориентированную на язык программирования Modula-2 и производительность соизмеримую с NS32032. В рамках комплексного проекта МАРС под эту архитектуру была написана операционная система Excelsior и набор прикладных программ, в частности, САПР[1], который в дальнейшем был переложен на x86 и развивается до сих пор под названием bCAD.

Упомянем еще развитие ветки такой архитектуры, как Эльбрус. Первая версия Эльбрус-1 выпущена в 1979 г., вторая Эльбрус-2 в 1984 г. Разработка третьей версии разделилась на две ветки. Первая - векторно-конвейерный Эльбрус-3.1(1990 МКП Модульный Конвейерный Процессор), который прошел госиспытания, но не был запущен в серию. Вторая ветка - Эльбрус-3(М), который стал одним из родоначальников архитектур с широким командным словом (VLIW - Very Long Instruction Word, будем рассматривать отдельно), развитием этой ветки стала линейка современных процессоров Эльбрус от МЦСТ.

Войны CISC vs. RISC

К началу восьмидесятых однокристальные микропроцессоры освоились в 16-разрядной нише и наметился переход к 32 разрядам. Это уже была область компьютеров среднего класса мощности со своими лидерами, традициями, инструментами, опытом, … но также тупиками и накопленными ошибками.

В то же самое время шло движение и сверху вниз, компьютеры среднего класса стремительно дешевели, так, Interdata представила в 1973 г. первый 32-разрядный

[1] Система Автоматического ПРоектирования

мини-компьютер (IBM 360 совместимый) дешевле 10 000 долларов. В какой-то момент эти два процесса должны были столкнуться.

Академик И.П.Павлов наблюдал столкновение волн возбуждения и торможения в коре головного мозга, приводящее к лабораторным неврозам и назвал это “сшибкой”. Здесь эффект был отчасти похожим.

- Часть разработчиков видела возможность сделать всё тоже самое, но компактнее, дешевле и мощнее. Например, DEC выпустила однокристальные процессоры T11(PDP-11) / J11(VAX). Или Texas Instruments с TMS9900 - однокристальным процессором, совместимым с популярной серией миникомпьютеров TI-990. Упомянем еще NS32K как развитие VAX. Глядя из будущего, легко понять, почему этот путь оказался тупиковым.

- Другая часть проектов росла снизу вверх и имела успешные эволюционирующие продукты. К таким отнесём x86 и MC68K.

  У Intel была простая и успешная серия 8086 наряду со сложной и провальной 432 серией. Может поэтому, архитектура развивалась достаточно эволюционно. 8086 превратился в дешевый 8088. Следующий шаг - 16-разрядные 80186 и более простой и дешевый 80188. 80286 16 разрядный процессор с 24-разрядной адресной шиной. 32-разрядный 80386 процессор с MMU тоже был совместим с изначальным 8086. При этом нельзя сказать, что Intel боялась экспериментировать - у них был и многопроцессорный iPCS с топологией гиперкуб и i860 с элементами VLIW и i960 с регистровыми окнами. Но об этом позже.

  Аналогично поступила Motorola с удачной серией 68K, которая развивалась эволюционно - наращивали частоту, сделали полноценную 32-разрядную шину данных, АЛУ, … Причем, до этого существовала относительно удачная 8-разрядная серия 6800, которую они бросили и не побоялись сделать всё с нуля.

- С третьей группой интереснее всего. Здесь отметились люди из академической среды, равно как и разработчики/исследователи из индустрии, которые увидели редкий шанс, имея за плечами и знания и практический опыт разработки процессоров, сделать новую[ые] архитектуру[ы] с нуля, на новом техническом уровне, по возможности избавиться от груза накопленных ошибок. Назовем такие имена как Эндрю Таненбаум, Дэвид Паттерсон, Джон Кок, Джон Хеннесси.

Необходимо учитывать также следующие обстоятельства.
Во-первых. На тот момент уже существовала свободно распространяемая (~ по цене магнитной ленты) переносимая операционная система (UNIX) с переносимым системным языком C. Причем, они еще не успели обрасти мышцами и накопить жирка, так что добавление новой архитектуры или учет её изменений не были слишком дорогостоящими. В начале UNIX развивался преимущественно энтузиастами и в 1985 году уже существовал в версии V8/BSD4.2 и был портирован с PDP-11 на VAX и

Interdata 8/32 (IBM 360),
а также существовали коммерческие
386/ix от Interactive Systems Corp,
Xenix от Microsoft (x86, PDP 11, Z80, MC68K),
Idris от Whitesmiths (PDP 11, VAX, MC68K, IBM 370, Atari).

Во-вторых, произошла тихая революция в компиляторах. Условной точкой отсчета можно считать 1981 год, когда математиком Грегори Хайтином (Gregory Chaitin) был предложен способ распределения регистров.

Речь идет о той стадии компиляции (роспись регистров, register allocation), между преобразованием во внутреннее представление в виде трёхадресного кода и стадией кодогенерации. Подробнее об этом в главе “стековые машины против регистровых”. Трёхадресный код - фактически код для процессора с бесконечным количеством регистров. Но в конечной архитектуре количество регистров ограничено и требуется решить, какие значениях можно оставить в регистрах, а какие следует переместить в память (грубо).

До того момента эта стадия компиляции выполнялась эмпирически, с помощью набора разнообразных рецептов/приёмов, которые следовало применять в тех или иных ситуациях, качество такого кода зачастую оказывалось сомнительным. Грегори Хайтин пришел к выводу, что задача распределения регистров сводится к задаче раскраски графа. Про эту задачу известно, что она NP-полная, т.е. стоимость ее решения экспоненциально зависит от числа вершин (исходных регистров, которые надо распределить). К счастью, в данном случае это не приговор, поскольку была также предложена приемлемая эвристика. В результате удалось создать компилятор (для языка PL/1), причем качество генерируемого им кода было сопоставимо с написанным вручную на ассемблере.

Грегори Хайтин участвовал в проекте IBM 801 который не получил широкого распространения, но опыт и знания, полученные в нём были использованы в дальнейшем в архитектуре POWER. Проект начат в 1975 г. как инициативная исследовательская разработка. Была собрана и проанализирована статистика по огромному количеству реальных программ (частоты использования инструкций, тайминги, работа с памятью ...) и сформулированы требования к перспективной архитектуре.

Надо сказать, что в то время идея перспективной архитектуры просто висела в воздухе и занимались ею не только в IBM, стоит упомянуть Berkeley RISC и Stanford MIPS проекты. Само слово RISC означает Reduced Instruction Set Computer в противопоставление с CISC. Консенсус, сложившийся на тот момент был таков (рассмотрим основные идеи):

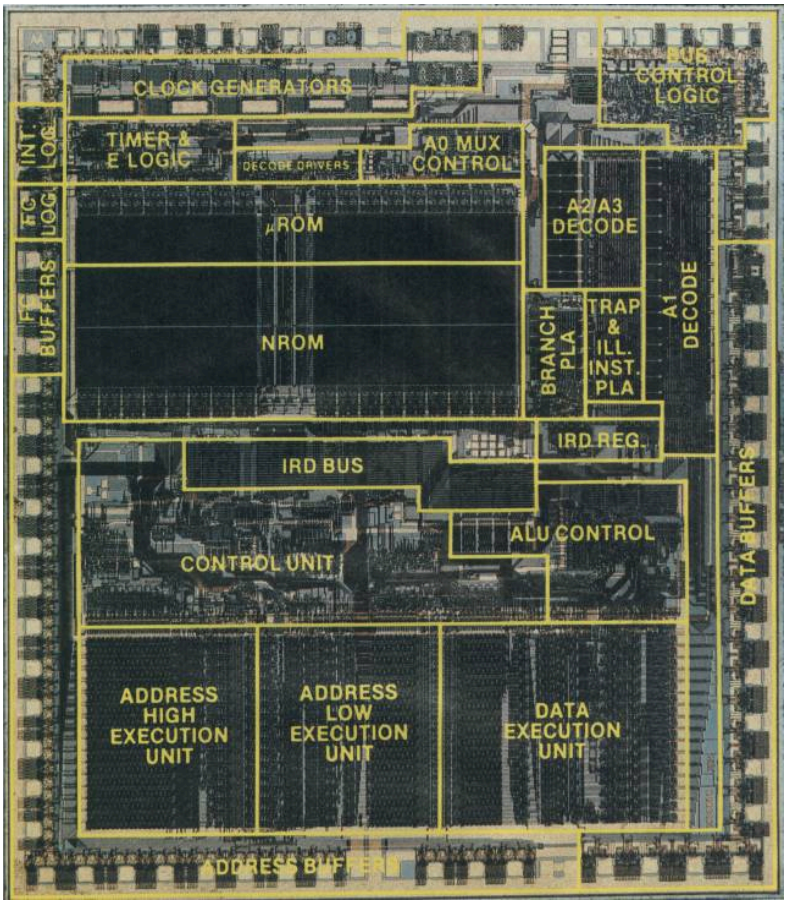
- **Микрокод - зло**. При появлении первых компьютеров никто особо не задумывался об архитектурных изысках, будущем развитии и поддержке существующих программ. Тем не менее, очень быстро выяснили, что переписывать заново весь накопленный код (языков программирования на тот момент не было) не очень удобно. Так появился т.н. микрокод - внешнее

представление компьютера, отделённое от его физического устройства. Кроме того, ~~Компьютеры с традиционной, т.н. CISC архитектурой традиционно имели набор инструкций, отличный от внутреннего представления этого кода. Причин несколько.~~ разработчики стремились к как можно более компактному коду, ведь память была не просто дорогой, её было совсем немного и хотелось поместить в неё как можно больше кода. ~~Кроме того,~~ Если разработчики при анализе реально работающего кода (написанного преимущественно вручную на ассемблере) видели постоянно повторяющиеся паттерны инструкций, они стремились в следующей итерации архитектуры внести такие паттерны в архитектуру в виде новых инструкций [*]. Позднее это стало называться CISC (Complex Instruction Set Computer, компьютер со сложной системой команд).

В результате возник такой механизм как микрокод, который преобразовывал внешнее представление во внутреннее, пригодное для исполнения.

Эндрю Танненбаум пишет, что опкоды CISC инструкций напоминают ему коды Хаффмана, применяющиеся при сжатии данных. Пожалуй, это не удивительно. Микрокод можно/стоит воспринимать как аппаратный компрессор для минимизации входного потока кода.

Объем логики, требовавшейся для работы микрокода, был не так уж и мал. Например, в MC68000 он занимал до 20% от площади микросхемы, т.е. его обслуживало ~ ~15 000 из 68 000 транзисторов (да, процессоры серии 68000 состояли из чуть более чем 68000 транзисторов).



Фиг.4 Микрофотография MC68000, к микрокоду относятся области NROM & uROM (отсюда)

Идея, которую заложили в новый класс архитектур (RISC) заключалась в том, чтобы выровнять внутреннее и внешнее представление кода, избавиться от микрокода, а высвободившиеся транзисторы отдать под что-нибудь полезное, например, сделать побольше регистров.

Да, исполняемый код от этого распухнет, но к тому времени стоимость памяти уже не была столь критична, а объем адресуемого 32-разрядными процессорами пространства был достаточен для размещения кода любого объёма.
Глядя на это из современности, отказ от микрокода видится затеей сомнительной и конъюнктурной. Да, микрокод 68000 занимал 20%, но в транзисторах это величина примерно постоянная, а общий размер процессора нет. Для 68020 с его 190 000 транзисторов это уже 8%, а для 68040 - ~1%.

Чтобы донести распухший (на десятки процентов) RISC код до процессора, его нужно пропустить через интерфейс памяти, а это достаточно дорогой ресурс, именно пропускная способность памяти зачастую является узким местом.

И не забудем про разработчиков процессоров, которые при вынесенной наружу внутренней архитектуре потеряли определённую гибкость.

- **Нужно большее регистров**. С этим не поспоришь. Регистры являются сверхоперативной памятью - значения в них уже готовы для загрузки в исполнительные устройства процессора. Иногда их называют кэшем уровня L0.

  В CISC архитектурах число регистров общего назначения невелико
  - 68K - 8
  - x86 - 8
  - PDP-11 - 8
  - IBM 360 - 16

  и увеличить их число невозможно, это будет уже другая архитектура. Одно уже только использование большего числа регистров в новой, написанной с нуля архитектуре способно увеличить её производительность в разы (по сравнению с CISC) при прочих равных.

  Этим и вызваны работы Грегори Хайтина по автоматическому распределению регистров. В IBM-801 поначалу было 16 штук 24-разрядных регистров, но примерно в 1980 архитектуру перепроектировали под 32 32-разрядных регистра. Распределять руками 32 регистра не слишком удобно, обычный компилятор PL\1 так же не мог эффективно справляться с ними в силу того, что был оптимизирован под 16 регистров IBM-360/370.

  Сама по себе концепция кэш-памяти в те годы была не в новинку и впервые в серийной машине появилась советской БЭСМ-6 (1967). Кэш был совсем небольшой (на 16 48-битных слов: 4 - для данных, 4 - для команд, 8 — буфер записи). До того был как минимум экспериментальный компьютер М-100 (1958г.), где роль кэша для памяти на ферритовых сердечниках выполняла также память на ферритовых сердечниках, только очень маленьких и сверхбыстрых. . Чуть позже появилась IBM 360/85 (1968), где размер кэша уже был 16 или 32 Кб в зависимости от модели. В производительных моделях мини компьютеров она также присутствовала, но в первых однокристальных микропроцессорах её не было. Когда кэш память появилась в CISC процессорах, она в какой-то мере компенсировала малое количество регистров. Попробуем оценить эту компенсацию.

  Сравнительные тесты экспериментального RISC II и 68000 показали, что первый в 1.4...4.2 раза быстрее. При этом 68000 имеет 16 регистров общего назначения, а RISC II - 22 регистра в окне вызова процедуры (про архитектуры с регистровыми окнами поговорим позже) плюс 10 глобальных (всего 138 регистров). Известно, что 68000 не имеет кэш памяти, она в линейке появилась в процессоре 68020, это были 256 байт кэша инструкций. Кэш данных добавили в 68030, тоже 256 байт. Сравним их относительную производительность.

  Известно, что 68000 выполнял в среднем 0.175 инструкций за такт. Для 68030 эта величина - 0.36 инструкций за такт. Следовательно, если привести RISC II к 68030, ускорение первого составит 0.68...2.04 раза. Что ж, результаты становятся не столь однозначными. Оказывается, на части задач CISC даже быстрее новомодного RISC.

Интересно, что VAX 11/780, который выполняет 0.2 инструкции за такт, но содержит 8K кэш, в 0.85...2.65 раза медленнее RISC II. Даже если архитектура устаревшая, кэш может придать ей прыти.

- **Конвейер - благо**. Каждый знает что такое конвейер. В компьютерных архитектурах идея распространена с 60-х годов и к началу 80-х использование по крайней мере двухстадийного (пока одна инструкция исполняется, другая декодируется) конвейера было общепринятой практикой. Во всяком случае в 68000 был именно такой двухстадийный конвейер.

Фиг.5 типичный RISC конвейер (отсюда)
IF - instruction fetch
ID - instruction decode
EX - execute
MEM - memory access
WB - write back, сохранение результата

Обращает на себя внимание то, что обращение к памяти стоит после исполнения инструкции. На самом деле это взаимоисключающие стадии, идёт либо исполнение, либо обращение к памяти. Вообще, конвейер весьма эффективен, когда все стадии длятся одинаковое время. И на первых, очень простых RISC-процессорах, где не было тяжелых инструкций, так и было. Но, скажем, чтение из памяти может занимать много тактов, так же как умножение или, тем более, деление. Пока длится продолжительная стадия, весь конвейер не сдвинется. Но всё равно это намного эффективнее, чем выполнять инструкции последовательно.

Упрощенные инструкции RISC на первый взгляд более подходят для конвейерной работы. Исполнительные устройства в RISC работают только с регистрами, если нужны данные из памяти, их предварительно требуется загрузить. Т.е. более сложные CISC инструкции после микро-кодирования распадаются на несколько внутренних (микро)инструкций, тогда как стадии конвейера касаются их всех, что менее эффективно.

В действительности, особой разницы нет, грубо, в CISC к конвейеру добавляется еще одна стадия - микро-кодирования. На производительности это явным образом не сказывается.

Для примера, в 68020 (1982 г.) появился трёхстадийный конвейер, в 68040 (1990 г) стадий стало 6 (и, кстати, кэш по 4 килобайта для данных и кода) с соответствующим приростом производительности - 1.1 инструкции на такт.

Для сравнения, MIPS R3000 (1988 г.) имел 5-стадийный конвейер, не содержал внутри-процессорного кэша и имел производительность близкую к 1 инструкции за такт.

- **Проще инструкции - проще процессор - больше частота**.
  Сравним одноклассников
  CISC
  - 68030 - появился в 1987 г., 273 000 транзисторов, 3-стадийный конвейер, производительность 0.36 инструкций на такт, частоты 16...50 МГц, кэш внутренний 256 байт + 256 байт, техпроцесс .8 мкм
  - 386DX - появился в 1985 г., 275 000 транзисторов, производительность 0.13 инструкций на такт, 16...40 Мгц, кэш внешний, техпроцесс 1.5 мкм

  RISC
  - SPARC MB86900 (V7) - появился в 1986 г., 110 000 транзисторов, 5-стадийный конвейер, производительность 0.51 инструкций за такт, частота 14.3...33 МГц, кэш внешний, техпроцесс 1.2 мкм
  - R3000 - появился в 1988 г., 110 000 транзисторов, 5-стадийный конвейер, производительность 0.7...0.86 инструкций за такт, частоты 20...33 МГц, кэш внешний, техпроцесс 1.3 мкм

  Производительность этих микропроцессоров сопоставима, особенно учитывая, что CISC инструкции “тяжелее” - выполняют больше работы. Насчет простоты - да, это правда, при сравнимой производительности RISC содержат на кристалле в 2.5 раза меньше логических элементов.

Итого. С технологической точки зрения, сколь-нибудь существенным элементом новизны в RISC процессорах было лишь увеличение числа регистров общего назначения. Это увеличение было возможно благодаря прогрессу в производстве СБИС (Сверх Больших Интегральных Схем) а также тому, что архитектуры писались с чистого листа, без оглядки на обратную совместимость и т.п..

При разработке RISC архитектур ориентировались в первую очередь на существующие CISC процессоры и по сравнению с ними RISC выглядел весьма перспективно. Но к моменту выпуска в производство оказалось, что некоторые CISC производители смогли мобилизоваться и подтянуть производительность своих процессоров. В первую очередь это касалось молодых производителей микропроцессоров, которым не нужно было поддерживать пользователей их стремительно устаревающей техники.

Так или иначе, началась технологическая гонка, которую мы знаем как “CISC/RISC войны”. Основное технологическое преимущество RISC - регистры было нивелировано внутри-кристальным кэшем, обе стороны наращивали его объем, тактовую частоту и сложность конвейеров, интегрировали сопроцессоры с плавающей точкой … В какой-то момент микропроцессоры научились переименовывать регистры, переставлять микро-инструкции… и тут на сцену вышли суперскалярные микропроцессоры (о них отдельный рассказ), которым стало (по большому счету) всё равно, какого типа - RISC или CISC у них набор внешних инструкций. С этого момента

понятие RISC - не более, чем название раскрученного брэнда, которое используется преимущественно по инерции (либо маркетологами).

Главными проигравшими в этой борьбе стали те технологические компании, что не смогли или не успели включиться в соревнование и постепенно потеряли свою аудиторию. Главными выигравшими - те, кто смог выдержать 10-летнюю гонку. И, конечно, пользователи, которые с осторожным оптимизмом оплатили всё это великолепие.