БЕЛОРУСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Курсовая работа

по дисциплине: “Архитектура вычислительных систем”

по теме “Обзор векторных вычислительных систем”

Выполнил: студент гр. 853505

Говин В.П.

Проверил: руководитель работы

Леченко А.В.

Минск 2020

Оглавление

Введение

Улучшение производительности вычислительных систем подразумевает под собой в первую очередь высокоскоростное исполнение программ. Это обосновано как требованиями пользователей, которые в свою очередь заинтересованы в наиболее быстром получении результатов исполнения программы, так и тем, что скорость исполнения определяет общее количество вычислительной работы, которую позволяет выполнять устройство за определённый период времени.

В некоторых областях увеличение скорости вычислений играет огромную роль, потому что время решения задач на стандартных ЭВМ зачастую слишком велико с точки зрения практического использования результатов. Конечно, стоимость в данном аспекте также играет огромную роль, но в итоге важнейшим фактором остаётся обеспечение возможности вычисления результатов за минимальный промежуток времени при минимальных (насколько это возможно достичь) затратах на эти вычисления..

Факторами, определяющие высокую стоимость суперЭВМ:

- Сложность оборудования и немалые затраты на разработку и конструирование при относительно небольшой серийности.
- Аппаратура, для создания которой требуются новые технологии, способные обеспечить предельные для нынешнего уровня развития техники показатели, имеет высокую стоимость. Стоимость напрямую зависит от увеличения числа логических элементов, количества выделяемой теплоты в единице объема и других подобных факторов.
- Программное обеспечение, позволяющее реализовать весь потенциал быстродействующей системы так же требуют не малых затрат на разработку.

Постановка задачи: проанализировать информацию о различных типах векторных вычислительных систем приведённую в различных источниках и сделать выводы о потенциале их применения.

Описание и классификация векторных процессоров

Векторный процессор - это процессор, который в качестве операндов для некоторых команд позволяет использовать векторы (массивы данных).

Векторный процессор может быть реализован в двух вариантах:

- Блок-расширение для универсальной ВС.
- Векторный процессор реализуется как самостоятельная ВС.

Существует два подхода к обработке в векторной архитектуре - векторно-параллельный или векторно-конвейерный.

В векторно-параллельном процессоре несколько функциональный блоков (ФБ) одновременно проводят операции над элементами векторов с плавающей запятой, каждый из которых отвечает за одну пару элементов.

В векторно-конвейерном процессоре (рис. 1.) элементы векторов обрабатываются одним конвейерным ФБ. Операции с числами с ПЗ достаточно сложны в вычислениях, но их можно разделить на несколько шагов. Каждый шаг обработки может быть реализован с помощью отдельной ступени конвейерного ФБ. Следующая пара операндов подается на вход конвейера как только освобождается его первая ступень. Одновременные операции возможно выполнять при помощи нескольких конвейерных ФБ, но в таком случае обработка будет реализовывать как векторно-параллельный, так и векторно-конвейерный подходы.

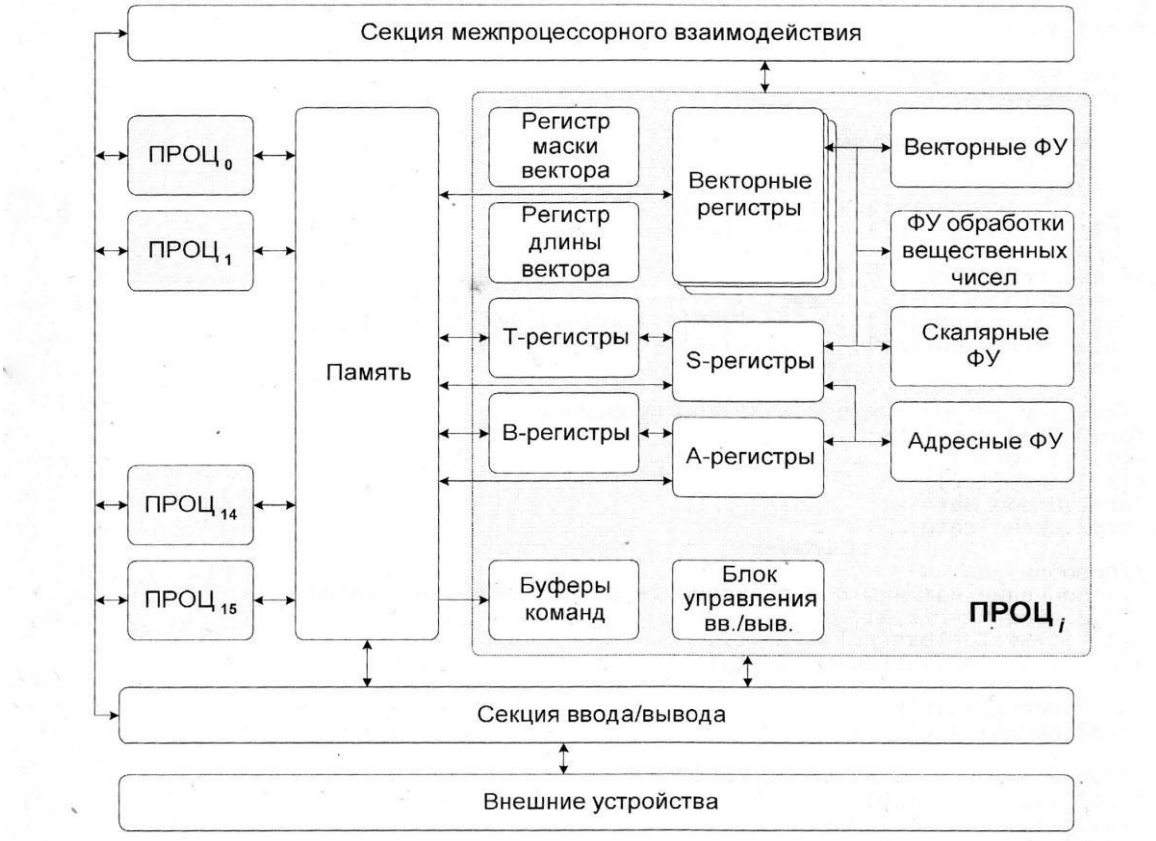

рис. 1. Схема векторно-конвейерного процессора

Применение векторных вычислительных систем на практике

Для моделирования реальных объектов характерно оперирование большими массивами чисел с плавающей запятой, массивы представляются матрицами и векторами, а алгоритмы их обработки описываются в терминах матричных операций. Основные операции над матрицами сводятся к однотипным действиям над парами элементов исходных матриц. Эти операции можно производить параллельно. В вычислительных системах, выполняющих скалярные операции, обработка матриц выполняется поэлементно и последовательно. Если размер входных данных достаточно велик, то их последовательная обработка займет большое количество времени. Это говорит о том, что универсальные ВС неэффективны для такого класса задач. При обработке массива данных требуются вычислительные устройства, позволяющие с помощью одной команды выполнять операцию сразу над всеми элементами массива - средства векторной обработки (рис2.).

Вектор (в векторной обработке) - одномерный массив данных (обычно в форме с плавающей запятой), которые находятся в памяти ВС. Длина вектора - количество элементов массива.

Многомерные же массивы представляются в виде набора одномерных массивов-векторов. Действия над многомерными массивами учитывают специфику их размещения. Способ размещения многомерного массива влияет на шаг изменения адреса элемента, выбираемого из памяти. Так, если матрица расположена в памяти построчно, адреса соседних элементов строки различаются на единицу, а для элементов столбца шаг равен четырем. При размещении матрицы по столбцам единице будет равен шаг по столбцу, а четырем - шаг по строке. В векторной концепции для обозначения шага, с которым элементы вектора извлекаются из памяти, применяют термин stride (шаг по индексу).

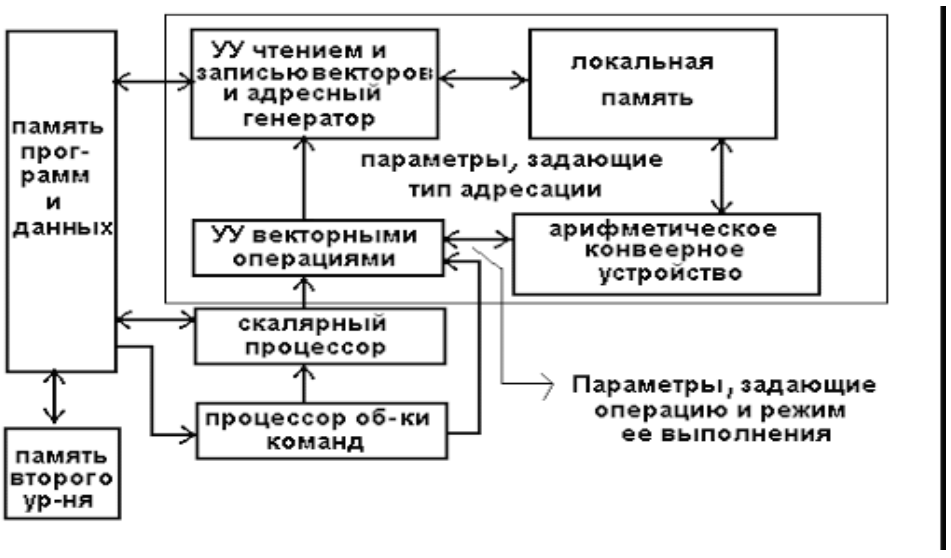

рис. 2. Принцип работы векторно-конвейерного процессора

Структура векторной вычислительной системы

В реальности векторные вычисления являются только частью всего вычислительного процесса. Скалярные операции так же являются неотъемлемой частью вычислений, и, возможно, имеют даже больший вес. Именно поэтому в векторных вычислительных системах, помимо векторного процессора, также присутствует и скалярный процессор.

Для SIMD-системы присуще выполнение единой программы, в которой присутствуют как скалярные, так и векторные операции. Программа, и все используемые ей данные хранятся в памяти ВС. Процессор обработки команд последовательно выбирает из памяти команды и направляет их в скалярный или векторный процессоры соответственно. Для повышения скорости обработки векторов все функциональные блоки векторных процессоров строятся по схеме конвейера так, чтобы каждая ступень конвейера выполняла операцию за один такт (В разных ФБ число ступеней различно). В некоторых векторных вычислительных системах, например Cray C90, этот подход несколько усовершенствован - конвейеры во всех функциональных блоках продублированы.

Некоторых векторные процессоры типа «регистр-регистр» реализуют технологию vector linking (зацепление векторов). В таких процессорах выходной векторный регистр одной операции над векторами используется в качестве входного регистра для следующей операции. Такая комбинация из последовательности умножения и суммирования характерна для операции свертки и встречается во многих векторных и матричных вычислениях. Суть vector linking заключается в том, что выполнение векторной команды начинается сразу, как только вычисляются компоненты участвующих в ней векторных операндов, не дожидаясь завершения вычисления полного вектора.

С середины 90-х годов прошлого века векторные ВС стали уступать другим более технологичным видам систем в производительности, но тем не менее разработка корпорации NEC - ВС SX-9 (рис. 3.) - является векторно-конвейерной ВС, пиковая производительность которой с 16 ядрами составляет 26,2 триллионов операций с плавающей запятой в секунду (TFLOPS).

рис. 3. ВС SX-9

Обзор системы типа SIMD

SIMD (Single Instruction stream Multiple Data stream (рис. 4.)) – одиночный поток команд и множественный поток данных. Системы типа SIMD оперируют большим количеством процессоров (их количество колеблется от 1024 до 16384), которые могут выполнять одну инструкцию используя различные данные в жесткой конфигурации. Единственная инструкция параллельно выполняется над многими элементами данных. В качестве примера машин, работающих на системе типа SIMD можно привести системы CPP DAP, Gamma II и Quadrics Apemille. Вторым подклассом SIMD-систем являются векторные компьютеры. Векторные компьютеры обрабатывают массивы входных данных почти так же, как и скалярные машины обрабатывают элементы таких массивов. Такого результата удалось достичь благодаря специальным векторным центральным процессорам.

При обработке данных при помощи векторных процессоров результаты можно получить на один, два или три такта частотогенератора (основной временной параметр системы). При работе в векторном режиме такие процессоры оперируют данными практически параллельно, что делает их в несколько раз более быстрыми, чем при работе в скалярном режиме. Примером такой системы является компьютер Hitachi S3600.

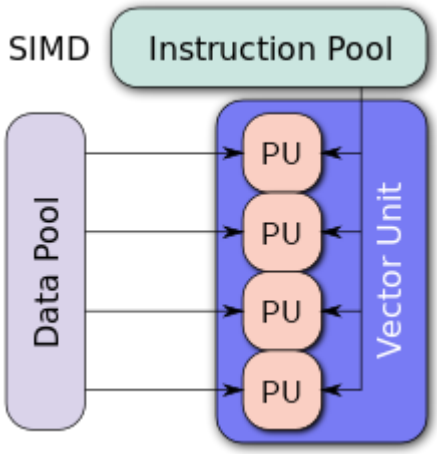

рис 4. Single Instruction stream Multiple Data stream

Преимущества SIMD

SIMD очень хорошо применим в программах, где одно и то же значение добавляется к (или вычитается из) большому массиву данных. В качестве примера можно привести изменение яркости гизображения. Каждый пиксель изображения состоит из трех значений: яркости красного (R), зеленого (G) и синего (B) составляющих. Для понижения или повышения яркости, значения R, G и B считываются из памяти, значение добавляется (или вычитается из него) их, а полученные значения записываются обратно в память.

У SIMD процессора есть два преимущества в реализации этого процесса:

- Данные на вход подаются блоками, поэтому значения могут быть полностью загружены сразу. Вместо серии инструкций получения значения одного пикселя, у процессора SIMD исполнится одна команда (получение N пикселей" (где N число, которое изменяется во время разработки)). Это занимает намного меньше времени, чем получение каждого пикселя в отдельности, так как происходит при традиционной конструкции процессора
- Команда оперирует всеми загруженными данными за одну операцию. Если система SIMD работает путем загрузки до восьми значений одновременно, операция добавления происходит над всеми восемью значениями одновременно, в отличие от суперскалярного процессора.

Недостатки SIMD

- Векторизации поддаются не все алгоритмы. Управление потоком тяжелых задач, нельзя легко ускорить при помощи SIMD. В теоретически возможно векторизовать сравнения для того, чтобы по максимуму оптимизировать кэш (и в всё равно по итогу это потребует больше промежуточных вычислений).
- Большое энергопотребление и занимаемая площадь кристалла за счёт больших регистров.
- Реализация алгоритма с инструкциями SIMD требует написания вручную (большинство компиляторов не умеют генерировать SIMD инструкции из программы на языке C).
- Программирование с помощью определенных наборов инструкций SIMD может включать в себя многочисленные проблемы низкого уровня.
    - SIMD может иметь ограничения на выравнивание данных.
    - Сбор данных в регистры SIMD и распределение их на правильные места - сложный процесс, что впоследствии приводит к потере эффективности.
    - Не все процессоры имеют инструкции SIMD, поэтому приходится дописывать не-векторизованные реализации (или модифицировать существующие векторизованные реализации).
    - В раннем наборе инструкций MMX общий файл регистров стека был с плавающей точкой, что привело к неэффективности при смешивании плавающей точкой и MMX кода. Тем не менее, SSE2 исправляет это.

Хронология SIMD

Примеры суперкомпьтеров, реализующих SIMD (не включая векторные процессоры):

- ILLIAC IV, c. 1974
- ICL Distributed Array Processor (DAP), c. 1974
- Burroughs Scientific Processor, c. 1976
- Geometric-Arithmetic Parallel Processor, from Martin Marietta, starting in 1981, continued at Lockheed Martin, then at Teranex and Silicon Optix
- Massively Parallel Processor (MPP), from NASA/Goddard Space Flight Center, c. 1983-1991
- Connection Machine, models 1 and 2 (CM-1 and CM-2), from Thinking Machines Corporation, c. 1985
- MasPar MP-1 and MP-2, c. 1987-1996
- Zephyr DC computer from Wavetracer, c. 1991
- Xplor, from Pyxsys, Inc., c. 2001.

Аппаратное обеспечение

В начале 90-х годов мелкосерийная реализация (64 и 128 бит) SIMD стала популярной на процессорах общего назначения.

Инструкции SIMD можно найти, в той или иной степени, на большинстве процессоров, в том числе AltiVec фирмы IBM и SPE для PowerPC, HP PA-RISC Multimedia Acceleration eXtensions (MAX), MMX и iwMMXt, SSE, SSE2, SSE3 SSSE3 и SSE4.x компании Intel и так далее.

Современные графические процессоры (GPU) часто широко реализуют SIMD, допускающие ветви, нагрузки, и хранение 128 или 256 бит за один раз.

Инструкции от Intel AVX SIMD в настоящее время обрабатывает 256 бит данных одновременно. Larrabee прототип микроархитектуры Intel включает в себя более двух 512-битный SIMD регистров на каждом из ядер (VPU: Wide Vector Processing Units), и это 512-бит SIMD возможно будет присутствовать во многих интегрированных в ядро архитектурах (Intel MIC).

Программное обеспечение

В обработке 3D-графики довольно часто используется SIMD (см. Обзор GPU). У некоторых систем присутствуют инструкции, переставляющие собой повторную упаковку элементов внутри векторов (особенно широко используется при обработке данных, сжатии и криптографии).

Внедрение SIMD в программное обеспечение ПК было сначала медленным, из-за ряда проблем. Одна из этих проблем заключается в том, что ранние наборы инструкций SIMD приводили к потере в производительности системы из-за повторного использования существующих регистров с плавающей точкой. Другие системы, такие как MMX и 3DNow!, предложили поддержку для типов данных, которые не пользовались большой популярностью и требовали много ресурсов переключения контекста инструкции, чтобы переключаться между регистрами FPU и MMX. Составителям также часто не хватало поддержки, что требовало от программистов знания ассемблера.

У SIMD на x86 был медленный старт. Введение 3DNow! от AMD и Intel SSE было беспорядочным, но сегодня система, успокоились (после того, как AMD приняла SSE) и новые составители должны привести к SIMD с большей поддержкой программного обеспечения. Intel и AMD теперь оба обеспечивают оптимизированные математические библиотеки, которые используют инструкции SIMD. Стали появляться альтернативы с открытым исходным кодом, как libSIMD, SIMDx86 и SLEEF.

Apple Computer имел несколько больший успех, несмотря на то, что они вышли на рынок SIMD позже, чем остальные. AltiVec предложил богатую систему и может быть запрограммирована с помощью все более сложных компиляторов от Motorola, IBM и GNU, поэтому прибегать к ассемблеру приходится всё реже. Кроме того, многие из систем, которые выиграют от SIMD поставлялись самой Apple, например, ITunes и QuickTime. Тем не менее, в 2006 году компьютеры Apple перешли на процессоры Intel x86. API-интерфейсы Apple, и средства разработки (Xcode) были изменены для поддержки SSE2 и SSE3, а также AltiVec. Apple, был доминирующим покупателем чипов PowerPC от IBM и Freescale Semiconductor и несмотря на то, что они отказались от платформы, дальнейшее развитие AltiVec продолжается. На WWDC '15, Apple объявила о поддержке SIMD векторов для версии 2.0 своего нового языка программирования Swift.

SIMD в регистре, или Swar - приемы, используемые для выполнения SIMD в регистрах общего назначения. Такие приёмы используются для параллелизации алгоритмов на процессорах, которые не поддерживают SIMD напрямую.

Компания Microsoft добавила SIMD в платформу .NET в RyuJIT. Использование библиотек, реализующих SIMD в .NET доступны в пакете NuGet Microsoft.Bcl.Simd.

.

Обзор GPU

Графический процессор (graphics processing unit(рис.5.)) - устройство в персональном компьютере, игровой приставке, мобильном устройстве, отвечающее за графический рендеринг. После 2000-го года GPU получили широкое применение во многих других устройствах: планшетные ПК, встраиваемые системы, портативные консоли, телевизоры.

Современные GPU довольно эффективно справляются с обработкой и реализуют компьютерную графику, благодаря тому, что работают на схожей с системой SIMD конвейерной архитектур. С задачей обработки графической информации они справляются в разы эффективнее, чем типичный CPU.

В современных видеокартах (рис.6.) графические процессоры используются для значительного ускорения обработки трёхмерной графики. Может применяться как в дискретной видеокарте, так и в интегрированных решениях (встроенных в северный мост либо в гибридный процессор).

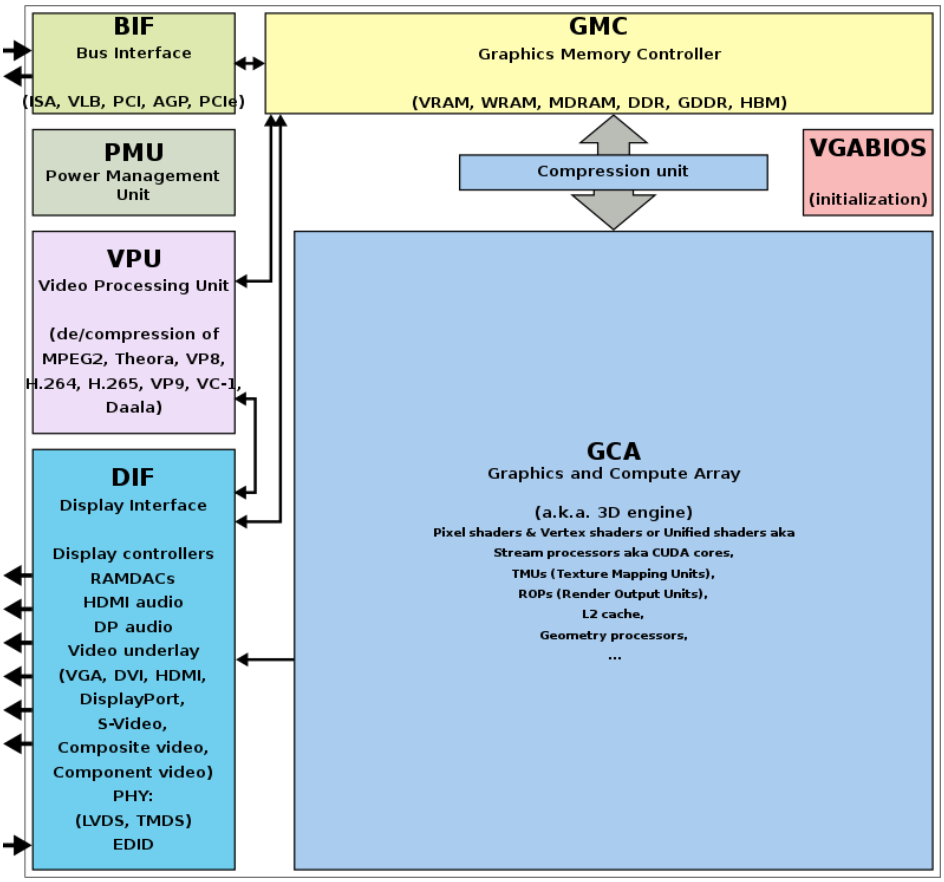

рис 5. Устройство графического процессора

Отличительными особенностями GPU от CPU являются:

- архитектура, направленная на увеличение производительности при расчёте текстур и сложнейших графических объектов и моделей;
- ограниченный набор команд.

Высокая вычислительная производительность GPU достигается за счёт особенностей архитектуры. Современные центральные процессоры имеют небольшое количество ядер, по сравнению с графическими процессорами, которые изначально проектируются как многопоточная структура с большим количеством ядер.

Разные архитектуры подразумевают разные принципы работы. Если архитектура CPU подразумевает последовательность обработки потока информации, то GPU с самого раннего этапа разработки предназначался для обработки компьютерной графики, поэтому рассчитан на выполнение массивных параллельных вычислений.

Каждая из этих архитектур разрабатывалась для определённых задач и все обе они имеют свои достоинства. CPU лучше выполняет последовательные задачи, в то время, как GPU имеет явное преимущество при обработке большого объёма данных. Существует только одно условие — в задаче должен присутствовать параллелизм.

Современные графические процессоры (в составе видеокарты) могут полноценно применяться для общих вычислений (GPGPU). Примерами таковых могут служить чипы 5700XT (от AMD) или GTX 1660 Super (от nVidia).

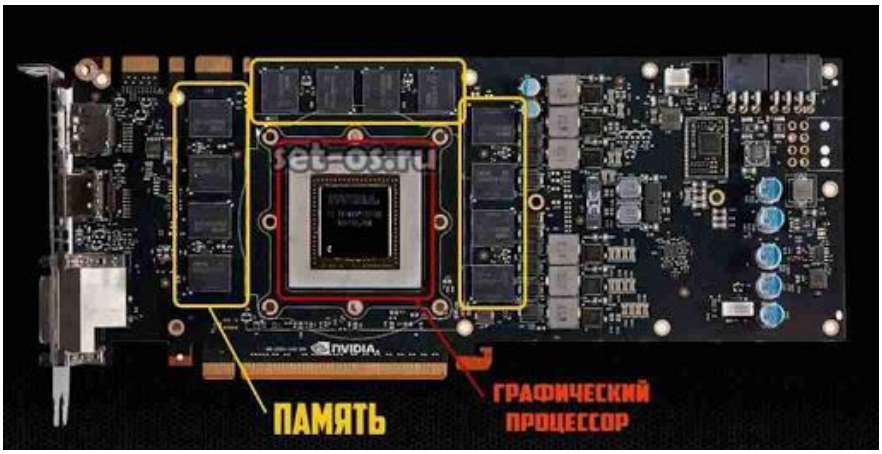

рис 6. Графический процессор в составе видеоадаптера.

Внешний графический процессор (eGPU)

Внешний графический процессор(eGPU) — графический процессор, который размещается вне корпуса ПК. Внешние графические процессоры иногда используются совместно с портативными компьютерами. Ноутбуки часто имеют большой объём оперативной памяти (RAM) и производительный CPU, но часто им не хватает мощного графического процессора, вместо которого используется менее мощный, но более энергоэффективный встроенный графический чип. Дискретные GPU обычно маломощны для обработки большого объёма данных или для других графически интенсивных задач, таких как редактирование видео.

Внешние GPU не пользовались большой официальной поддержкой поставщиков. Однако это не остановило энтузиастов от внедрения настроек eGPU.

рис 6. eGPU от фирмы Lenovo.

Обзор STAR-100

Один из видов конвейерной системы STAR-100 (STAR - STring ARray computer) разработан фирмой CDC с 1965 по 1973 г. Система была анонсирована ещё в 1970 г. Первый ее аналог был произведён в августе 1973 г.

Система STAR-100 создавалась с учетом языка программирования APL (A Programming Language). APL - диалоговый язык, которому характерны развитые средства работы с векторами, матрицами, массивами и набором базовых операций и компактностью записи.

ВС STAR-100 состояла из двух подсистем: одна из которых отвечала за обработку данных, а другая - за функции операционной системы. Ядром первой подсистемы был процессор, который состоял из конвейеров. В типовых конфигурациях STAR-100 процессоры состояли из трех конвейеров: K_1 K_2, K_3 (рис.7.). Конвейеры имели свои задачи: первые два из них (K_1, K_2) были предназначены для выполнения векторных операций, а последний, третий конвейер (K_3) предназначался для выполнения скалярных операций, т.е. K_1 и K_2 - конвейеры (Floating-Point Pair Pipelines), каждый из которых служил для выполнения операций с плавающей запятой над парами векторов данных, K_3 - конвейер (string data pipeline), предназначавшийся для обработки обычных операндов, не организованных в векторы. Основной объем вычислительного процесса выпадал на конвейеры K_1 и K_2, поэтому, быстродействие системы зависело от них.

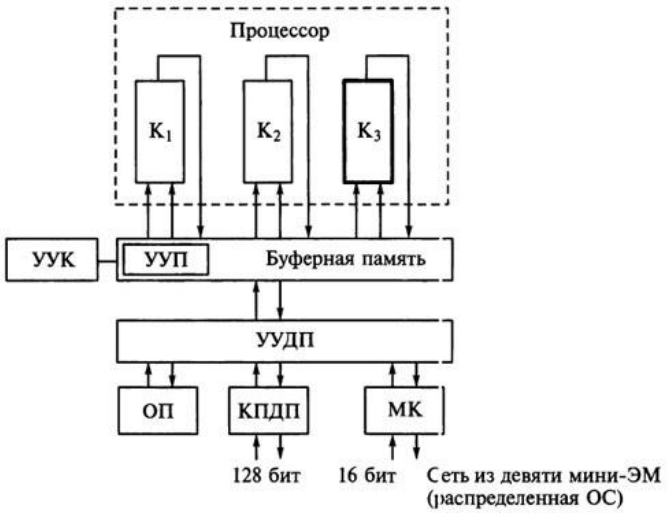

Рис. 4.3. Функциональная структура системы STAR-100:

K_1, K_2, K_3 — конвейеры; УУК — устройство управления командами; УУП — устройство управления потоками; УУДП — устройство управления доступом к памяти; ОП — оперативная память; КПДП — канал прямого доступа в память; МК — мультиплексный канал

Рис.7. архитектура ВС STAR-100.

Конвейеры STAR-100 имели программируемую структуру (конфигурация могла модифицироваться), в следствии чего, имелась возможность (на одном и том же множестве элементарных блоков обработки) выполнять различные арифметические операции. Но, перед началом следующей операции конвейер подлежал перенастройке (программирование на выполнение очередной операции).

В конвейерах K_1 и K_2 благодаря введению булевского вектора была обеспечена избирательная обработка компонентов векторов-операндов. Единица в *i*-м разряде булевского вектора означала, что операция над *i*-ми компонентами соответствующей пары векторов не производиться.

В каждом конвейере была заложена возможность реализации операции сложения, а в двух из них - K_1 и K_2 - операций умножения и деления. Состав элементарных блоков обработки информации конвейеров был выбран с учетом распределения вероятностей использования микроопераций различных типов.

Каждый конвейер K_i (i = 1, 2, 3) мог включать в себя приблизительно 30 блоков обработки информации. Работа всех блоков осуществлялась параллельно, но каждый из них оперировал с вполне определенными элементами векторов данных либо со своими операндами.
Любой конвейер воспринимал 64-разрядный код либо как один 64-разрядный операнд, либо как два 32-разрядных операнда. Время выполнения операции над парой операндов в любом из блоков конвейеров не превышало 40 нс. Следовательно, данные могли поступать в конвейеры K_1 и K_2 со скоростью 103 млн опер./с.

В систему STAR-100 входил набор из 230 команд, 65 из которых предназначались для работы с векторами данных и 130 команд - для работы со скалярами. Средства управления подсистемой переработки данных были представлены композицией из устройства управления командами (УУК), устройства управления потоками (УУП) и устройства управления доступом к памяти (УУДП).

Первое (УУК) имело буфер опережающего просмотра команд (емкостью в четыре 512-разрядных суперслова) механизмом работы, организованном на стековой системе. Второе устройство (УУП) использовалось для управления потоками операндов и команд между УУДП, конвейерами и УУК.

В оперативной памяти хранились программы и данные. Она была реализована на магнитных сердечниках и имела емкость 512-1024 К 64-разрядных слов, т.е. до 8 Мбайт. Память могла включать в себя до 32 модулей и относилась к классу памятей с перемежающимися адресами. Время цикла памяти было равно 1,28 мкс, однако допускались одновременные обращения к составляющим модулям. Имелись четыре виртуальных канала обращения к памяти, которые реализовывались устройством управления доступом к памяти. Два канала использовались для чтения операндов (для каждого из конвейеров K_1 и K_2, работавших параллельно, из памяти

выбиралось по два 64-разрядных операнда); один - для записи результатов (64-разрядный результат от каждого из конвейеров K_1 и K_2); один - для обмена информацией с устройствами ввода-вывода (либо с одним быстродействующим устройством с полосой пропускания 128 бит, либо с восемью медленными устройствами в режиме разделения времени).

Буферная память была реализована вследствие того, что быстродействие оперативной памяти было существенно ниже быстродействия процессора. Буферная память представляла собой совокупность регистров с временем цикла 40 нс. Назначение канала прямого доступа в память (КПДП) и мультиплексного канала (МК) следует из их названий и структуры связей между устройствами STAR-100.

Функции операционной системы STAR-100 реализовывались специальной вычислительной сетью из девяти мини-машин. Система программирования STAR-100 включала компиляторы с языков APL-STAR, COBOL и FORTRAN.

Первый образец системы STAR-100 был установлен в Ливерморской радиационной лаборатории им. Лоуренса (Lawrence Livermore Laboratory). Были осуществлены поставки системы в правительственные организации и в армию США. Система STAR-100 использовалась для управления запуском антиракет в системе противоракетной обороны США; она широко применялась при решении сложных проблем науки, техники и экономики.

Вычислительная система STAR-100 поддерживала изменение числа конвейеров, числа и состава внешних устройств, емкости оперативной и внешней памяти. Создавались и усеченные варианты STAR-IB, а также система SUPERSTAR (или CDC 8600), которая в 1,5-4 раза превосходила по производительности STAR-100 и была более компактной (реализована на более совершенных интегральных схемах).

Выводы

Развитие в сфере векторных вычислений происходило на протяжении многих лет. За эти годы было создано и опробовано множество аналогов векторных ВС. Все они имели свои достоинства и недостатки, но с течением времени развитие технологий позволяло всё более эффективно производить обработку больших объемов данных.

На данный момент векторные вычисления играют важную роль в современных вычислительных системах, так как с ростом производительности растут и объёмы данных, которые необходимо обрабатывать. Данная сфера разработок по прежнему является крайне перспективной благодаря тому, что подавляющее большинство современной техники так, или иначе, работает, основываясь на параллельных вычислениях.

Литература

[1] Kunzman, D. M.; Kale, L. V. (2011), Programming Heterogeneous Systems, 2011

[2] Цилькер Б.Я. Организация ЭВМ и систем : Учебник для вузов / Б.Я. Цилькер, С.А. Орлов. - 2-е изд. - СПб.: Питер, 2011.

[3] Супер ЭВМ. Аппаратная и программная организация / Под. ред. С. Фернбаха: Пер. с англ.- М.: Радио и связь, 1991.

[4] nsc.ru [интернет-ресурс].

[5] bmstu.wiki [интернет-ресурс].

[6] wikipedia.org [интернет-ресурс].

[7] habr.com [интернет-ресурс].