В главе про электромеханические вычисления мы видели, что одной из причин развития была необходимость производить масштабные вычисления для составления таблиц функций. Через сто лет мало что изменилось в том, что людям всё еще требовались расчеты. В частности, военному флоту (в данном случае США) были необходимы таблицы стрельб. Конечно, была масса других задач, но мало кто мог соперничать с флотом в размере бюджета.

Таблица стрельб - это набор параметров прицельных приспособлений для стрельбы в зависимости от дальности, азимута (силы Кориолиса), высоты над уровнем моря, температуры, веса снаряда, скоростей и направлений движения - собственной и цели, … Определить всё это экспериментальным путём - слишком долго и слишком дорого, проще рассчитать на основе математической модели.

На тот момент для расчетов существовали т.н. табуляторы, но это устройства скорее для обработки потоковых данных, их сложно применять там, где вычисления зависят от логических условий. К тому же, табуляторы долго и сложно перенастраиваются на новую задачу. Поэтому неудивительно, что чиновники флота обратили внимание на зарождающиеся проекты универсальных программируемых вычислителей. Таких нашлось два - Mark-I и ENIAC.

Mark-I (1944)

Фиг.1 Harvard Mark-I (отсюда)

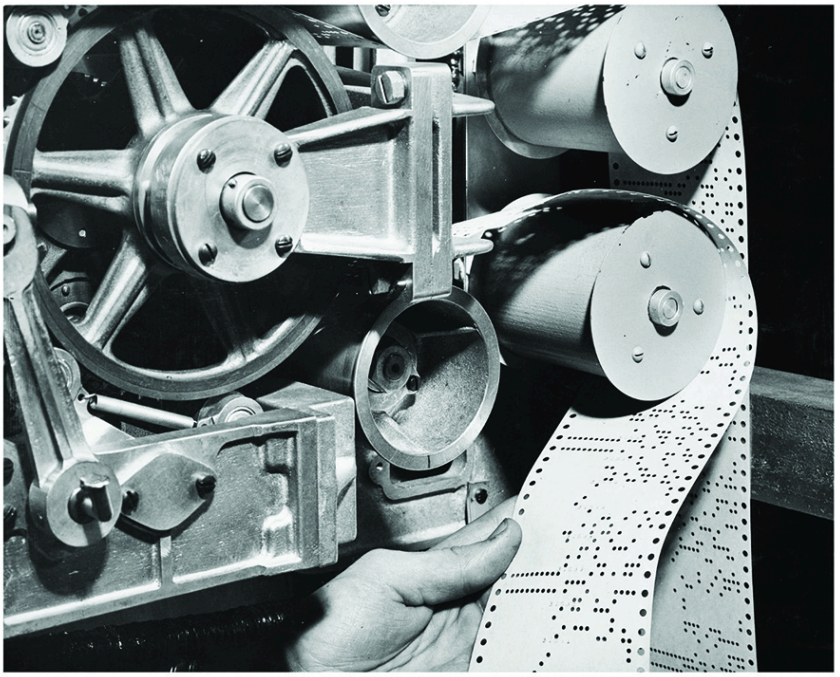
Проект развивался в Гарвардском университете, возглавлял его Говард Эйкен (Howard Aiken). Эйкен был знаком с работами Бэббиджа над разностной и аналитической машинами, поэтому при работе над диссертацией в Гарвардском университете, где он решал дифференциальные уравнения (и при этом приходилось выполнять массу расчетов), у него возникла идея реализовать идею Бэббиджа на современном техническом уровне. С этой идеей Эйкен обратился в IBM (на тот момент

производитель табуляторов), где он, пусть и не с первой попытки, но нашел понимание. С этого момента проект стал называться IBM ASCC (Automatic Sequence Control Calculator). Разработка велась с 1937 по 1944 гг., когда, наконец, проект был сдан флоту.

Конструктивно Mark-I был построен на элементной базе от выпускаемых IBM табуляторов - механические сумматоры в духе Паскалины, память данных на электромеханических реле - 72 регистра по 23 десятичных числа плюс знак. Программы размещались на перфоленте. В наборе команд не было условных операций (ветвления), для организации циклов перфоленту склеивали в кольцо.

Одна из первых программ для Mark-I была написана Джоном фон-Нейманом (John von Neumann) в рамках работы над т.н. Манхэттенским Проектом, компьютер проработал до 1959 г., после чего был разобран. В процессе развитие проекта появились Mark-II (1947 г.), Mark-III (1949 г.) и Mark-VI (1952 г.), все за авторством Говарда Эйкена. Mark-II - улучшенный Mark-I, Mark-III - часть компонентов заменена на электронные, Mark-IV - полностью собран на электронных компонентах, память на магнитных сердечниках и магнитном барабане.

Все компьютеры линейки сохраняли изначальную особенность - разные интерфейсы доступа к памяти программ и данных. Собственно поэтому все более поздние архитектуры с этой особенностью объединяют в общее семейство - процессоры с гарвардской архитектурой.



Фиг.2 Считыватель перфолент Mark-I (отсюда)

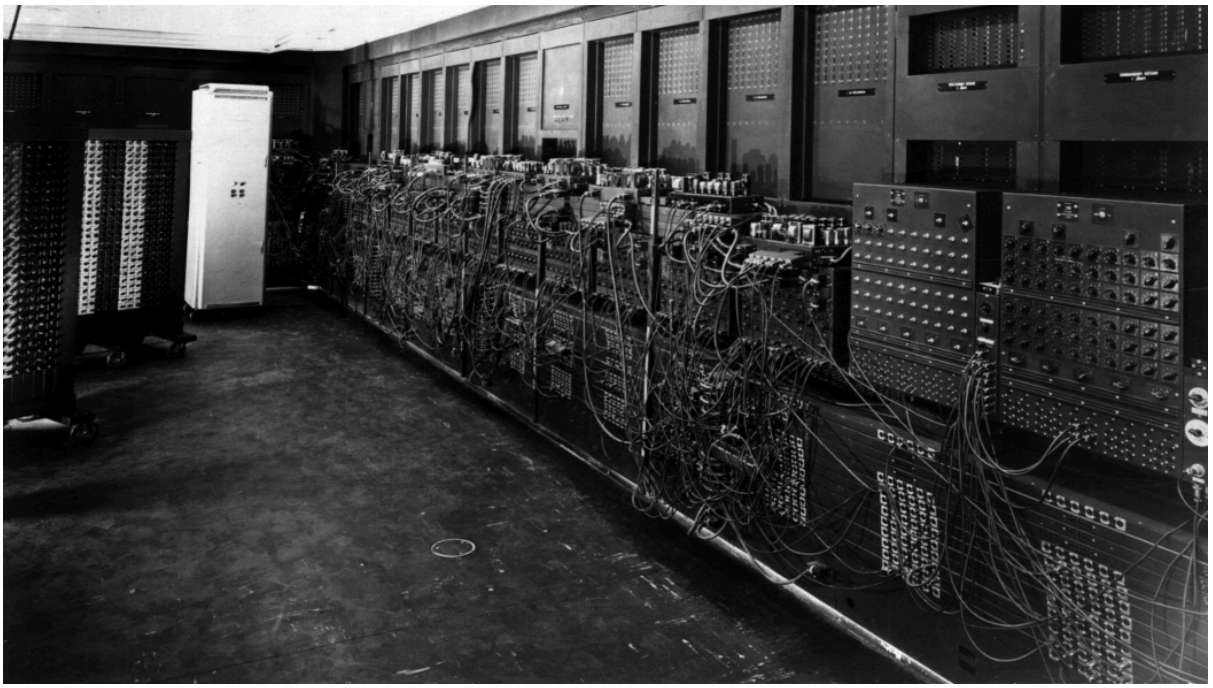
ENIAC (Electronic Numerical Integrator and Computer, 1945)

У истоков проекта стояли два человека - Джон Мокли (John Mauchly) и Джон Эккерт (J. Presper Eckert), первый работал преподавателем в Пенсильванском университете,

второй был его студентом. Кстати, память на ртутных линиях задержки - это тоже во многом заслуга Эккерта.

Мокли работал с одной из немногих “разностных машин”, применявшейся в том числе для вычисления таблиц стрельб и продвигал идею высокоскоростных универсальных вычислений. В конце концов его услышали и в течении 1943…1945 гг. был построен компьютер ENIAC. В 1944 г. в качестве научного консультанта к проекту присоединился Джон фон-Нейман, который также как и в случае с Mark-I делал расчеты в рамках Манхэттенского проекта.

В отличие он Mark-I, ENIAC был построен на электронных компонентах - ламповых триодах, с памятью на триггерах. Всего компьютер располагал памятью на 20 слов (по 20 десятичных разрядов). Память программ задавалась вручную с помощью матриц коммутации (Mark-1) или массивами тумблеров (ENIAC), на это уходило немало времени. Комплекс включал в себя более 17 000 электронных ламп и был не слишком надёжен, средняя наработка на отказ была около 2 дней.



Фиг.3 ENIAC (отсюда)

В сущности, ENIAC так же как и Mark-I имел отдельный интерфейс к памяти программ и его архитектуру смело можно назвать гарвардской.

EDVAC (Electronic Discrete Variable Automatic Computer, 1949)

Программирование (ручная коммутация) ENIAC занимала много времени, но это считалось приемлемым, поскольку компьютер разрабатывался преимущественно для однотипных вычислений. Тем не менее, жизнь не остановилась и команда разработчиков продолжала двигаться вперёд.

В начале 1944 г. Эккерт выдвинул идею памяти на ртутных трубках задержки и предложил хранить вместе и сами программы и их данные. Модифицировать ENIAC для использования новой памяти оказалось невозможно, началась разработка нового проекта компьютера с условным названием EDVAC. В итоге он содержал два блока по 64 ртутных трубки с памятью по 8 слов в каждой. Идея активно обсуждалась в том числе с участием фон-Неймана. В октябре 1944 г. фон-Нейман опубликовал статью (First Draft of a Report on the EDVAC), в которой он обобщил идеи Эккерта и придал им наукообразие (правда в авторах статьи только он один и никаких ссылок на Эккерта нет). Можно сказать, именно с этой статьи и пошел термин **фон-Неймановская архитектура** - общая память для программ и данных.

В 1946 г. пути разработчиков разошлись, Мокли и Эккерт решили заниматься разработкой коммерческих компьютеров, фон-Нейман и некоторые другие перешли в Институт Перспективных Исследований (Institute for Advanced Study) в Принстоне (не имеющего накакого отношения к Принстонскому Университету), где начали строить свой компьютер для научной работы. Поэтому фон-Неймановскую архитектуру иногда называют Принстонской.

Отвечая на напрашивающийся вопрос - да, возможно, фон-Неймановскую архитектуру стоило бы назвать Эккертовской, впрочем сейчас, спустя 80 лет, это совершенно неважно. Говарду Эйкену (Mark-I) приписывают фразу: “Don't worry about people stealing your ideas. If your ideas are any good, you'll have to ram them down people's throats” (Не стоит беспокоиться, что кто-то украдёт вашу идею, даже если она хороша, вам придётся буквально запихивать её в их глотки). Что-ж, в данном случае это сработало по другому.

Гарвардская vs фон-Неймановская

И Mark-I и ENIAC имели разные механизмы доступа к памяти программ и данных не из-за архитектурных изысков, а лишь по причине примитивности элементной базы. Вот Mark-IV уже имел два интерфейса памяти - магнитный барабан на 4 000 слов с доступом за 2…20 мс (под инструкции) и память на магнитных сердечниках емкостью в 230 слов с доступом за 1 мс (данные).

Это совершенно обычная ситуация в то время, учитывая, с какой скоростью появлялись и уходили в забытье новые виды физической памяти. С такой же скорость появлялись и исчезали разные компьютерные архитектуры, которые зачастую были представлены единственным экземпляром. Разные виды памяти имели разные интерфейсы, иногда с произвольным доступом, иногда с последовательным, это тоже приходилось учитывать в системе команд. Привычного нам деления памяти на оперативную и дисковую не было, файловая система в обычном для нас виде впервые появилась в ОС Multics (1964..1970 гг.), до этого зачастую разные виды памяти причудливо перемешивались, образуя подобие виртуальной памяти, управляемой самой прикладной программой.

Само деление архитектур на гарвардскую и фон-Неймановскую появилось в 1970-е годы, когда оперативная и дисковая память окончательно размежевались и

повсеместно распространилась технология виртуальной памяти. Речь уже не шла об интерфейсах к физически разным видам памяти, а о разных массивах физически одинаково устроенной памяти для данных и кода. А сами названия: Гарвардская и фон-Неймановская (Принстонская) - скорее, маркетинговый ход, нежели принципиальное расхождение. Время было отнюдь не “травоядное”, разработчики мини-компьютеров начали поджимать производителей mainframe-ов, коими были так называемая группа “IBM и семь гномов”, в ход шли не только не только технические новинки, но и маркетинговые уловки.

Разберем достоинства и недостатки обоих методов на примере относительно устоявшегося современного состояния.

1) **Производительность**. Независимые интерфейсы доступа к памяти могут работать не мешая друг другу. Это с одной стороны. С другой, интерфейс доступа к самой распространённой DRAM памяти не так уж и тривиален, плюс к нему полагается и шина доступа к данным. Если речь о внешней, не внутри-кристальной памяти, шина памяти должна быть обеспечена нужным количеством выводов с микросхемы. А это немало - DDR4, в частности имеет 288 выводов (шина адреса, шина данных, …), удвоить их количество не так то и просто.

2) **Безопасность**. Когда данные и код расположены в одном адресном пространстве (вполне возможно что и вперемешку), есть риск испортить исполняемый код, небрежно обращаясь с данными. Или же фрагменты кода могут быть злонамеренно модифицированы с использованием особенностей архитектуры.

   Так, например, в архитектуре x86 предусмотрен один сегмент стека, в котором вперемешку идут и локальные данные процедур и адреса возврата из них. Если испортить в стеке адрес возврата, то возврат из функции произойдет в нештатное место, заблаговременно подготовленное злоумышленником. В случае гарвардской архитектуры, возврат внутрь сегмента данных невозможен.

   Злоумышленник может попытаться создать злонамеренный код в адресном пространстве кода, но это не совсем обычное поведение программы и ОС легко контролировать и пресекать такие попытки.

   Могут ли фон-Неймановские архитектуры противопоставить что-то этому? Определённо, да. Память имеет страничную организацию, каждой странице соответствует описатель (дескриптор) в т.н. таблице страниц (page table, см гл. 3.3. Виртуальная память). В этом дескрипторе в неиспользуемых разрядах адреса (в x86-64(AMD64) задействованы, например, 48 разрядов из 64) можно разместить атрибуты страницы: только для чтения, разрешено исполнение, … Таким образом, при попытке возврата в неправильное место, при преобразовании виртуального адреса возврата в физический, аппаратно будет обнаружено, что сегмент неисполняемый, что повлечет ошибку и остановку программы. Если же злоумышленник попытается создать страницу с соответствующими правами, это легко (в теории) будет обнаружено и пресечено

ОС.

В архитектурах с тегированной памятью это решается еще кардинальнее, например, в Эльбрусах от МЦСТ, каждое 64-разрядное слово имеет признак что в нём - данные или код. А чтобы поменять этот атрибут, нужны права суперпользователя.

3) **Гибкость**. Разные интерфейсы памяти могут иметь разную ширину. Например, в архитектуре SHARC (Super Harvard ARChitecture) от Analog Devices, внутри-кристальная память может быть поделена на три части:
- для кода, слово размером в 48 разрядов
- для данных, слово размером в 32 разряда
- для чисел с плавающей точкой размером в 40 разрядов (может быть опущена).
Внешняя память может быть сконфигурирована только под один (любой) размер слова.

   Это т.н. сигнальный процессор, оптимизированный под решение задач обработки сигналов. Разработчик имеет возможность максимально гибко конфигурировать процессор под конкретную задачу. Но для задач/процессоров общего назначение такая работа неприменима т.к. оптимальное разбиение памяти для разных задач разное, а переконфигурация налету невозможна.

Что же в результате? Гарвардские архитектуры имеют сомнительные преимущества и существенный недостаток - неуниверсальность. В результате на данный момент они частично занимают узкую нишу сигнальных процессоров и микроконтроллеров. Из существующих систем можно назвать PIC от Microchip Technology, AVR от Atmel Corp, SHARC, TigerSHARC и Blackfin от Analog Devices, серия TMS320 от Texas Instruments..

Модифицированная Гарвардская архитектура

“Чистые” Гарвардские машины имеют раздельные адресные пространства для кода и данных. Но, как мы видели ранее, раздельные адресные пространства влекут за собой раздельные шины доступа, два комплекта выводов из микросхемы …

А что если архитектура будет разделять работу с кодом и данными, но хранить их в одном адресном пространстве. Фон-Неймановская снаружи, Гарвардская внутри, если можно так сказать. Это называется “расширенная Гарвардская архитектура”.

Почти все современные архитектуры можно формально отнести к модифицированным Гарвардским, раздельный кэш для данных и кода (по 256 байт), по-видимому, впервые появился в микропроцессоре 68030 от Motorola (1987 г.). Сейчас это обычная практика, кэш первого уровня делают раздельным (обычно их называют L1I и L1D), более высокие уровни общими. Имеются также раздельные TLB (Translation Lookaside Buffer) как минимум для кэша первого уровня, часто и для второго.

Сделано это не из теоретических соображений, а, прежде всего, связано с оптимизацией процессора. Функции кэша данных и кода немного разные. Кэш кода неизменен, в него можно только загрузить и выгрузить данные. Для кэша кода не нужно поддерживать межядерную/межпроцессорную когерентность, т.е. его организация существенно проще. Было бы странно этим не воспользоваться.