1.Какво е компютър?

Компютърът е устройство, което може да бъде инструктирано да извършва автоматично произволни последователности от аритметични или логически операции. Способността на компютрите да следват обобщен набор от операции, наречени програми, им позволява да изпълняват изключително широк спектър от задачи.

Има аналогови и дигитални компютри.

Аналогови компютри-Операционни усилватели, безкрайни стойности.

Дигитални компютри-краен набор от възможни стойности.

Аналого-дигитални(колбички)-вакуумни туби, биполярен преходен транзистор(BJT).

2.Какво е компютърна архитектура?

В компютърното инженерство компютърната архитектура е набор от правила и методи, които описват функционалността, организацията и изпълнението на компютърните системи. Архитектурата на една система се отнася до нейната структура по отношение на отделно определени компоненти на тази система и техните взаимовръзки.

Някои дефиниции на архитектурата я определят като описваща възможностите и модела на програмиране на компютъра, но не и конкретна реализация. В други дефиниции компютърната архитектура включва архитектурен дизайн на набор от инструкции, дизайн на микроархитектура, логическо проектиране и реализация.

Първата документирана компютърна архитектура е в кореспонденцията между Charles Babbage и Ada Lovelace, описваща аналитичната машина.

Charles Babbage (26 декември 1791 г. – 18 октомври 1871 г.) е английски математик. Математик, философ, изобретател и машинен инженер, Babbage създава концепцията за цифров програмируем компютър.

Някои го смятат за "баща на компютъра".

Механичният компютър е компютър, изграден от механични компоненти като лостове и зъбни колела, а не от електронни компоненти. Най-често срещаните примери са добавяне на машини и механични броячи, които използват завъртането на зъбни колела за увеличаване на изходните дисплеи. По-сложни примери могат да извършват умножение и деление - Friden използва движеща се глава, която спира на всяка колона - и дори диференциален анализ. Един модел, счетоводната машина Ascota 170, продадена през 60-те години на миналия век, изчислява квадратни корени.

Аналитичният двигател е предложен механичен компютър с общо предназначение, проектиран от английския математик и компютърен пионер Charles Babbage. За първи път е описан през 1837 г. като наследник на диференциалния двигател на Babbage, който е дизайн за по-опростен механичен калкулатор.

Augusta Ada King, графиня на Lovelace (10 декември 1815 – 27 ноември 1852) е английски математик и писател, известна главно с работата си върху предложения механичен компютър с общо предназначение на Charles Babbage, аналитичната машина. Тя беше първата, която разпозна, че машината има приложения извън чистото изчисление, и публикува първия алгоритъм, предназначен да бъде изпълнен от такава машина. В резултат на това тя често се смята за първия компютърен програмист.

През 1945г. John von Neumann(унгарско-американски математик, физик, компютърен учен и инженер) създава първият проект за доклад на EDVAC, който описва организация на логически елементи.

Първият проект на доклад за EDVAC (обикновено съкращаван до First Draft) е непълен документ от 101 страници, написан от John von Neumann и разпространен на 30 юни 1945 г. от Herman Goldstine, служител по сигурността на класифицирания проект ENIAC. Той съдържа първото публикувано описание на логическия дизайн на компютър, използващ концепцията за запаметена програма, която противоречиво стана известна като архитектурата на фон Нойман.

CPU(Control Processor Unit) се състои от регистри,CU(Control Unit) и от ALU(Arithmetic Logic Unit).

Alan Turing е предложил по-подробен електронен калкулатор за автоматичната изчислителна машина през 1945г. и цитира статията на John von Neumann.

Alan Turing(23 юни 1912 – 7 юни 1954) е английски математик, компютърен учен, логик, криптоаналитик, философ и теоретичен биолог.

Автоматичният изчислителен двигател (ACE-Automatic Computing Engine) е британски ранен електронен сериен компютър със запаметена програма, проектиран от Alan Turing.

Проектът беше управляван от John R. Womersley, началник на отдела по математика на Националната физическа лаборатория (NPL). Използването на думата Двигател е в почит към Charles Babbage и неговата различна машина и аналитична машина. Техническият дизайн на Turing „Предложен електронен калкулатор“ е продукт на неговата теоретична работа през 1936 г. „За изчислимите числа“ и неговия военен опит в Bletchey Park, където компютрите Colossus са успели да разбият немските военни кодове. В своя документ от 1936 г. Turing описва идеята си като „универсална изчислителна машина“, но сега е известна като универсална машина на Turing.

За език за програмиране или изчислителна машина се казва, че е Turing Complete, ако може да се използва за симулиране на машина на Turing.

Машината на Turing е хипотетична машина, замислена от математика Alan Turing през 1936 г. Въпреки своята простота, машината може да симулира ВСЕКИ компютърен алгоритъм, без значение колко сложен е той!

Компютри, които имат същата (или много подобна) ISA (архитектура на набора от инструкции-Instruction Set Architecture):

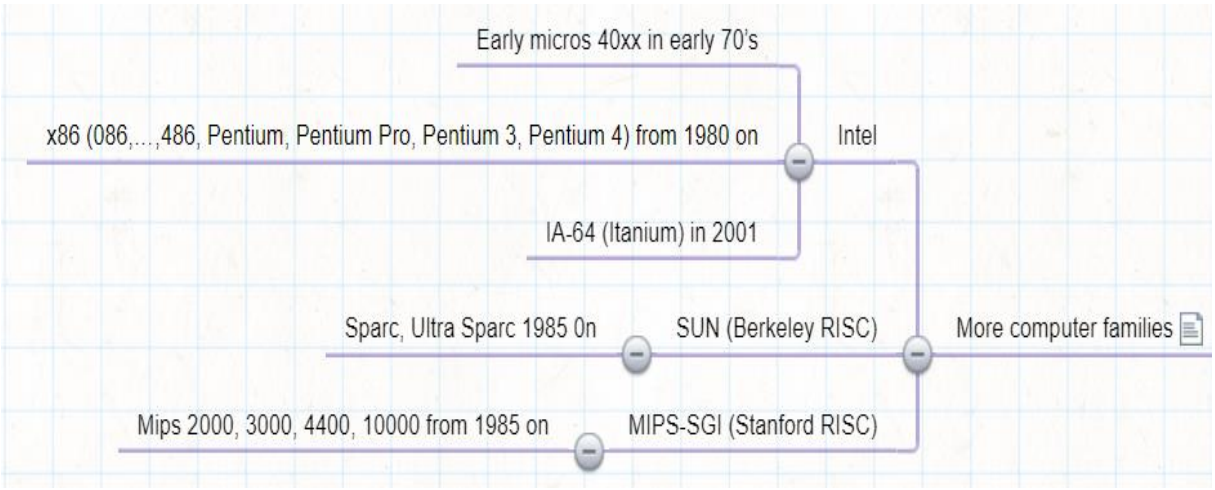
IBM-704,709,70xx и др. От 1955 до 1965 г.

360,370,43xx,33xx от 1965 г. до момента,

Power PC;

DEC-PDP-11,VAX от 1970 до 1985 г.,

Alpha (сега Compaq, сега HP) през 1990-те.



Какво е MIPS?

MIPS (Microprocessor without Interlocked Pipelined Stages) е семейство от компютърни архитектури с набор от инструкции (RISC) с намален набор от инструкции (ISA), разработени от MIPS Computer Systems, сега MIPS Technologies, базирана в Съединените щати.

Има множество версии на MIPS: включително MIPS I, II, III, IV и V; както и пет издания на MIPS32/64 (съответно за 32- и 64-битови реализации). Ранните MIPS архитектури бяха 32-битови; 64-битовите версии бяха разработени по-късно. От април 2017 г. текущата версия на MIPS е MIPS32/64, издание 6. MIPS32/64 се различава основно от MIPS I–V чрез дефиниране на System Control Coprocessor в режим на ядрото в допълнение към архитектурата на потребителския режим.

Каква е истинската разлика между x86, ARM и MIPS архитектурите?

Наборът от инструкции(The Instruction Set) е единствената последователна разлика.

Какво е операционната система?

Трудно е да се определи какво представлява една операционна система, освен да се каже, че това е

софтуер, който работи в режим на ядрото - и дори това не винаги е вярно. Част от

проблемът е, че операционните системи изпълняват две по същество несвързани функции:

предоставяне на приложни програмисти (и приложни програми, естествено) чист

абстрактен набор от ресурси вместо разхвърляните хардуерни и управлението им

хардуерни ресурси. В зависимост от това кой говори, може да чуете най-вече

за една или друга функция. Нека сега да разгледаме и двете.

Операционната система като разширена машина

Архитектурата (набор от инструкции, организация на паметта, I/O и структура на шината) на повечето компютри на ниво машинен език е примитивна и неудобна за

програма, особено за вход/изход.

Операционните системи съдържат много драйвери за управление на I/O устройства. Но дори това ниво е твърде ниско за повечето приложения. Поради тази причина всички

операционните системи предоставят още един слой абстракция за използване на дискове: файлове. Използвайки тази абстракция, програмите могат да създават, пишат и четат файлове, без да се налага да

справете се с разхвърляните детайли за това как всъщност работи хардуерът. Тази абстракция е ключът към управлението на цялата тази сложност. Трябва да се отбележи, че истинските клиенти на операционната система са приложните програми (чрез приложните програмисти, разбира се). Те са тези

които се занимават директно с операционната система и нейните абстракции. За разлика от това, край

потребителите се занимават с абстракциите, предоставени от потребителския интерфейс, или обвивка на командния ред, или графичен интерфейс.

Операционната система като мениджър на ресурси

Концепцията за операционна система като основно предоставяне на абстракции на приложните програми е изглед отгоре надолу. Алтернативен възглед отдолу нагоре го поддържа

операционната система е там, за да управлява всички части на сложна система. В изглед отдолу нагоре работата

на операционната система е да осигури подредено и контролирано разпределение на

процесори, памети и I/O устройства сред различните програми, които ги искат. Съвременните операционни системи позволяват множество програми да бъдат в паметта и да се изпълняват

в същото време. Когато компютър (или мрежа) има повече от един потребител, необходимостта от управление

а защитата на паметта, входно/изходните устройства и други ресурси е още повече от

В противен случай потребителите биха могли да се намесват един в друг. Накратко,

този възглед за операционната система твърди, че нейната основна задача е да следи

кои програми използват кой ресурс, за да предоставят заявки за ресурси, за сметка

за използване и за посредничество на конфликтни заявки от различни програми и потребители. Управлението на ресурсите включва мултиплексиране (споделяне) на ресурси по два различни начина: във времето и в пространството. Когато ресурсът е мултиплексиран по време, различно

програмите или потребителите се редуват да го използват. Първият от тях може да използва ресурса,

после още един и т.н. Например само с един процесор и множество програми

които искат да работят на него, операционната система първо разпределя процесора на една програма,

след това, след като е работила достатъчно дълго, друга програма може да използва процесора, след това друга и накрая отново първата. Определяне на това как ресурсът е време

мултиплексиране — кой следва и за колко време — е задачата на операционната система. Друг пример за времево мултиплексиране е споделянето на принтера. Когато множество

заданията за печат са поставени на опашка за печат на един принтер, трябва да се вземе решение

за това коя следва да бъде отпечатана.

Другият вид мултиплексиране е космическото мултиплексиране. Вместо клиентите

като се редуват, всеки получава част от ресурса. Например, основната памет обикновено е разделена между няколко изпълнявани програми, така че всяка от тях може да бъде резидентна

по едно и също време (например, за да се редувате с помощта на процесора). Ако приемем, че има

е достатъчно памет за съхраняване на множество програми, по-ефективно е да държите няколко

програми в паметта наведнъж, вместо да даде на една от тях цялата, особено ако тя

се нуждае само от малка част от общия брой. Разбира се, това повдига въпроси за справедливостта,

защита и т.н., а решаването им зависи от операционната система. Друг

ресурс, който е мултиплексиран в пространството, е дискът. В много системи един диск може

задържат файлове от много потребители едновременно. Разпределяне на дисково пространство и запазване

проследяването на това кой кои дискови блокове използва е типична задача на операционната система.

Разликата между микроконтролер и микропроцесор

1. Ключова разлика и в двете е наличието на външна периферия, където микроконтролерите имат вградени RAM, ROM, EEPROM, докато ние трябва да използваме външни схеми в случай на микропроцесори.

2. Тъй като всички периферни устройства на микроконтролера са на един чип, той е компактен, докато микропроцесорът е обемист.

История на Операционните системи

1.Първо поколение(1945-55): Вакуумни туби

Професор Джон Атанасов и неговият аспирант Клифърд Бери построиха

което сега се счита за първия функциониращ цифров компютър в Щатския университет на Айова. Използва 300 вакуумни тръби. Приблизително по същото време Конрад Цузе в Берлин

построи компютъра Z3 от електромеханични релета. През 1944 г. Колосът е бил

построен и програмиран от група учени (включително Алън Тюринг) в Блечли

Парк, Англия, Mark I е построен от Хауърд Айкън в Харвард и ENIAC

е построен от Уилям Маучли и неговия аспирант Дж. Преспер Екерт в

Университет на Пенсилвания. Някои бяха двоични, някои използваха вакуумни тръби, някои

бяха програмируеми, но всички бяха много примитивни и отне секунди, за да се изпълнят равномерно

най-простото изчисление.

В тези ранни дни една група хора (обикновено инженери) проектира,

изгражда, програмира, управлява и поддържа всяка машина. Цялото програмиране беше

направено на абсолютен машинен език или още по-лошо, чрез свързване на електрически вериги чрез свързване на хиляди кабели към таблата за управление на машината

основни функции. Езиците за програмиране бяха непознати (дори асемблер

беше неизвестно). Операционните системи бяха нечувани. На практика всички проблеми бяха прости

прости математически и числени изчисления, като смилане

таблици на синуси, косинуси и логаритми или изчисляване на артилерийски траектории.

До началото на 50-те години рутината се подобри донякъде с въвеждането

на перфокарти. Вече беше възможно да се пишат програми на карти и да се четат

вместо да използвате табла; иначе процедурата беше същата.

2.Второ поколение(1955-65): Транзистори и бач системи

Въвеждането на транзистора в средата на 50-те промени радикално картината. Тези машини, сега наречени мейнфрейми, бяха заключени в големи, специално

климатизирани компютърни зали, с екип от професионални оператори, които да ги управляват. За изпълнение на работа (т.е. програма или набор от програми),

програмистът първо написва програмата на хартия (на FORTRAN или асемблера), след което я перфорира на карти. Като се има предвид високата цена на оборудването, не е изненадващо, че хората бързо

търси начини за намаляване на загубеното време. Общоприетото решение беше

пакетна система. Идеята зад нея беше да се събере тава, пълна с работни места във входа

стая и след това ги прочете на магнитна лента с помощта на малък (относително) евтин компютър, като IBM 1401, който беше доста добър в четенето на карти,

копиращи ленти и печат, но изобщо не са добри в числените изчисления. Други, много по-скъпи машини, като IBM 7094, бяха използвани за

реални изчисления. След това операторът зареди специална програма (прародител на днешния

операционна система), който прочете първото задание от лентата и го стартира. Резултатът беше

записани на втора лента, вместо да бъдат отпечатани. След приключване на всяка работа,

операционната система автоматично прочете следващата задача от лентата и започна да я изпълнява. Започна с а

$JOB карта, указваща максималното време за изпълнение в минути, номера на сметката, която да бъде

заредени и името на програмиста. След това дойде $FORTRAN карта, която казва на

операционна система за зареждане на компилатора на FORTRAN от системната лента. Тя беше пряко последвана от програмата, която трябваше да бъде компилирана, и след това $LOAD карта, насочваща

операционната система, за да зареди току-що компилираната обектна програма.Напред

дойде картата $RUN, която казва на операционната система да стартира програмата с данните

следвайки го. Накрая картата $END отбеляза края на заданието. Тези примитивни

контролните карти бяха предшествениците на съвременните черупки и интерпретаторите от командния ред.

Големите компютри от второ поколение се използват най-вече за научни и инженерни изчисления, като например решаване на частични диференциални уравнения, които често се срещат във физиката и инженерството. Те до голяма степен са програмирани във FORTRAN и

асемблер. Типичните операционни системи бяха FMS (Fortran Monitor

System) и IBSYS, операционната система на IBM за 7094.

3.Трето поколение(1965-80): ICs и Multiprogramming

До началото на 60-те години повечето производители на компютри имаха две отделни, несъвместими продуктови линии. От една страна, имаше ориентирани към думи, широкомащабни научни компютри, като 7094, които бяха използвани за числени изчисления на индустриална сила в науката и инженерството. От друга страна, имаше ориентирани към символи, търговски компютри, като 1401, които бяха широко

използвани за сортиране и печат на ленти от банки и застрахователни компании. Разработването и поддържането на две напълно различни продуктови линии беше скъпо предложение за производителите. Освен това много нови потребители на компютри първоначално се нуждаеха от малка машина, но по-късно я надраснаха и искаха по-голяма

машина, която ще изпълнява всичките им стари програми, но по-бързо.

IBM се опита да реши и двата проблема с един удар, като представи System/360. 360 беше серия от софтуерно съвместими машини, вариращи от модели с размер 1401 до много по-големи, по-мощни от мощните

7094.Тъй като всички имаха

същата архитектура и набор от инструкции, програмите, написани за една машина, биха могли

работи на всички останали - поне на теория. От 360г

е проектиран да обработва както научни (т.е. числени), така и търговски изчисления,

едно семейство машини можеше да задоволи нуждите на всички клиенти. IBM 360 беше първата голяма компютърна линия, използваща (дребномащабни) интегрални схеми (интегрирани схеми), като по този начин осигури голямо предимство цена/производителност спрямо второто поколение машини, които са изградени от отделни транзистори. Най-голямата сила на идеята за „единично семейство“ беше едновременно нейната най-голяма слабост. Първоначалното намерение беше целият софтуер, включително и операционната

система, OS/360, трябваше да работи на всички модели.

Нямаше начин IBM (или някой друг по този въпрос) да може да напише част от софтуера, която да отговори на всички тези противоречиви изисквания. Резултатът беше огромна и изключително сложна операционна система, вероятно две до три

порядък по-голям от FMS. Всяка нова версия коригира някои грешки и въвежда нови

такива, така че броят на грешките вероятно е останал постоянен с течение на времето.

Един от дизайнерите на OS/360, Фред Брукс, впоследствие написа остроумен и

проницателна книга (Brooks, 1995), описваща опита му с OS/360. Те също така популяризираха няколко ключови

техники, отсъстващи във второ поколение операционни системи. Вероятно най-важното от тях беше мултипрограмирането. На 7094, когато текущата работа

на пауза, за да изчака завършване на лента или друга I/O операция, процесорът просто се включи

неактивен, докато I/O приключи. Решението, което се разви, беше да се раздели паметта на няколко части, с

различна работа във всеки дял. Докато една работа се чакаше

I/O за завършване, друга работа може да е използването на процесора. Ако можеха да се заемат достатъчно работни места

в основната памет наведнъж, процесорът може да бъде зает почти 100% от времето. Друга основна функция, присъстваща в операционните системи от трето поколение, беше

възможност за четене на задания от карти на диска веднага щом бъдат пренесени на

компютърна зала. Тази техника се нарича буфериране (от Simultaneous Peripheral Operation On Line) и

също се използва за изход. Въпреки че операционните системи от трето поколение бяха много подходящи за големи научни

изчисления и масивна търговска обработка на данни, те все още бяха основно

партидни системи. Това желание за бързо време за реакция проправи пътя за споделяне на време, вариант

на мултипрограмиране, при което всеки потребител има онлайн терминал

първата система за споделяне на време с общо предназначение, CTSS (Compatible Time Sharing System), е разработена в M.I.T. на специално модифициран 7094 (Corbato´ et al., 1962). След успеха на системата CTSS, M.I.T., Bell Labs и General Electric

(по това време голям производител на компютри) решава да се заеме с разработването на „компютърна помощна програма“, тоест машина, която ще поддържа стотици потребители с едновременна споделяне на време. Техният модел беше електрическата система — когато

имате нужда от електричество, просто забивате щепсел в стената и в рамките на разумното, както

много мощност, от която се нуждаете, ще бъде там. Проектантите на тази система, известна като

MULTICS (MULTIPlexed Information and Computing Service), предвиден

една огромна машина, осигуряваща изчислителна мощност за всички в района на Бостън. MULTICS имаше смесен успех. Той е проектиран да поддържа стотици потребители

на машина само малко по-мощна от компютър, базиран на Intel 386. Това не е толкова лудо, колкото звучи, тъй като в тези

дни хората знаеха как да пишат малки, ефективни програми, умение, което впоследствие беше напълно загубено. Накратко, MULTICS въведе много основополагащи идеи в

компютърна литература, но превръщайки я в сериозен продукт и голяма реклама

успехът беше много по-труден, отколкото някой очакваше. Въпреки това,

M.I.T. продължи и в крайна сметка накара MULTICS да заработи.

В крайна сметка беше продаден като

търговски продукт от компанията (Honeywell), която купи компютърния бизнес на GE и беше инсталирана от около 80 големи компании и университети по целия свят. До края на 20-ти век концепцията за компютърна помощна програма се провали

но може да се върне под формата на облачни изчисления. Въпреки липсата на търговски успех, MULTICS имаше огромно влияние върху

последващи операционни системи (особено UNIX и неговите производни, FreeBSD,

Linux, iOS и Android). Друго голямо развитие през третото поколение беше феноменалното

растеж на миникомпютрите, като се започне с DEC PDP-1 през 1961 г. PDP-1 имаше

само 4K 18-битови думи, но при $120 000 на машина (по-малко от 5% от цената на

7094), продава се като горещи торти. Един от компютърните учени в Bell Labs, който е работил върху MULTICS

проект, Кен Томпсън, впоследствие намери малък миникомпютър PDP-7, който не

един използваше и се зае да напише съкратена версия за един потребител на MULTICS.

Тази работа по-късно се развива в операционната система UNIX, която става популярна

в академичния свят, с правителствени агенции и с много компании. Засега е достатъчно да кажем, че защото източникът

кодът беше широко достъпен, различни организации разработиха свои собствени (несъвместими) версии, което доведе до хаос. Разработени две основни версии, System V, от

AT&T и BSD (Berkeley Software Distribution) от Калифорнийския университет в Бъркли.

Те имаха и малки варианти. За да стане възможно да се пише

програми, които могат да работят на всяка UNIX система, IEEE разработи стандарт за

UNIX, наречен POSIX, който повечето версии на UNIX сега поддържат. POSIX дефинира а

минимален интерфейс за системно повикване, който трябва да поддържат съвместимите UNIX системи. В

Всъщност някои други операционни системи вече също поддържат POSIX интерфейса. Като настрана, заслужава да се спомене, че през 1987 г. авторът пусна малък

клонинг на UNIX, наречен MINIX, за образователни цели. Функционално MINIX е

много подобен на UNIX, включително поддръжка на POSIX. Желанието за безплатна производствена (за разлика от образователната) версия на MINIX

накара финландския студент Линус Торвалдс да напише Linux.

4.Четвърто поколение(1980-В наши времена):Лични компютри

През 1999 г. Apple

прие ядрото, получено от микроядрото Mach на университета Карнеги Мелън

който първоначално е разработен да замени ядрото на BSD UNIX. Така, Mac

OS X е UNIX-базирана операционна система, макар и с много отличителен интерфейс.

Когато Microsoft реши да създаде наследник на MS-DOS, това беше силно

повлиян от успеха на Macintosh. Той създаде базирано на GUI системно обаждане ed Windows, което първоначално работеше върху MS-DOS. Въпреки това, започвайки през 1995 г.

Беше пусната свободно стояща версия, Windows 95, която включваше много операционни

системни функции в него, като използва основната система MS-DOS само за зареждане и

стартиране на стари MS-DOS програми. През 1998 г. беше пусната леко модифицирана версия на тази система, наречена Windows 98. Независимо от това, както Windows 95, така и Windows 98 все още съдържаха голямо количество 16-битов асемблер на Intel.

Друга операционна система на Microsoft, Windows NT (където NT означава

Нова технология), който беше съвместим с Windows 95 на определено ниво, но вътрешно пренаписване от нулата. Беше пълна 32-битова система. Водещият автор на Windows NT беше Дейвид Кътлър, който също беше един от дизайнерите на

VAX VMS операционна система, така че някои идеи от VMS присъстват в NT. Това също не се получи, така че Microsoft излезе с още една версия на Windows 98, наречена Windows Me (Millennium Edition). През 2001 г. а

беше пусната леко обновена версия на Windows 2000, наречена Windows XP.

Тази версия имаше много по-дълъг срок на действие (6 години), като по същество заменя всички предишни версии на Windows. Все пак раждането на версиите продължаваше без прекъсване. След Windows 2000,

Microsoft раздели семейството на Windows на клиентска и сървърна линия. Клиентът

линията беше базирана на XP и неговите наследници, докато сървърната линия включваше Windows

Server 2003 и Windows 2008. Трети ред, за вградения свят, се появи

малко по-късно. Всички тези версии на Windows се разделиха на своите вариации във формата

на сервизни пакети. Беше достатъчно, за да накараме някои администратори (и автори на учебници по операционни системи) да омразим.

След това през януари 2007 г. Microsoft най-накрая пусна наследника на Windows

XP, наречен Vista. Той дойде с нов графичен интерфейс, подобрена сигурност и

много нови или подобрени потребителски програми. Microsoft се надяваше, че ще замени Windows

XP напълно, но никога не го направи. Вместо това получи много критики и лоша преса,

най-вече поради високите системни изисквания, рестриктивните условия за лицензиране и поддръжката за управление на цифровите права, техники, които затрудняват потребителите да

защитен от копиране материал. С пристигането на Windows 7, нова и много по-малко гладна за ресурси версия

на операционната система, много хора решиха да пропуснат изцяло Vista. Windows 7

не въведе твърде много нови функции, но беше сравнително малък и доста стабилен. За по-малко от три седмици Windows 7 спечели повече пазарен дял от

Vista след седем месеца. През 2012 г. Microsoft пусна своя наследник, Windows 8, an

операционна система с напълно нов външен вид и усещане, насочена към сензорни екрани. Другият основен претендент в света на персоналните компютри е UNIX (и неговият

различни производни). UNIX е най-силен на мрежови и корпоративни сървъри, но е така

също често присъства на настолни компютри, преносими компютри, таблети и смартфони. На компютри, базирани на x86, Linux се превръща в популярна алтернатива на Windows за студенти и все повече корпоративни потребители. Терминът x86 се отнася за всички съвременни процесори, базирани на семейство архитектури с набор от инструкции, които започнаха с

8086 през 70-те години. Има много такива процесори, произведени от компании като AMD и Intel, и под капака те често се различават значително:

процесорите могат да бъдат 32 бита или 64 бита с малко или много ядра и тръбопроводи, които могат

бъде дълбоко или плитко и т.н. Въпреки това за програмиста всички те изглеждат доста

подобни и всички те все още могат да изпълняват 8086 код, който е написан преди 35 години. Където

разликата е важна, вместо това ще се позоваваме на изрични модели — и ще ги използваме

x86-32 и x86-64 за обозначаване на 32-битови и 64-битови варианти. FreeBSD също е популярна производна на UNIX, произхождаща от проекта BSD

в Бъркли. Всички съвременни компютри Macintosh работят с модифицирана версия на FreeBSD

(OS X). UNIX също е стандартен за работни станции, захранвани с висока производителност

RISC чипове. Неговите производни са широко използвани на мобилни устройства, като тези, работещи с iOS 7 или Android. Много потребители на UNIX, особено опитни програмисти, предпочитат интерфейс, базиран на команди, пред GUI, така че почти всички UNIX системи поддържат прозоречна система

наречена X Window System (известна още като X11), произведена в M.I.T. Тази система управлява основното управление на прозорците, позволявайки на потребителите да създават, изтриват, преместват,

и преоразмерете прозорците с помощта на мишка. Често пълен GUI, като Gnome или

KDE е достъпен за работа върху X11, което дава на UNIX вид и усещане

като Macintosh или Microsoft Windows, за онези потребители на UNIX, които искат такъв

нещо.

Интересно развитие, което започна да се случва в средата на 80-те години, е

растежа на мрежите от персонални компютри, работещи с мрежови операционни системи и разпределени операционни системи (Tanenbaum and Van Steen, 2007).

мрежова операционна система, потребителите са наясно със съществуването на множество компютри и могат да влизат в отдалечени машини и да копират файлове от една машина на друга. Всяка машина работи със собствена локална операционна система и има собствен локален потребител

(или потребители).

Мрежовите операционни системи не се различават фундаментално от еднопроцесорните операционни системи. Очевидно се нуждаят от контролер на мрежов интерфейс и

някакъв софтуер на ниско ниво, който да го управлява, както и програми за постигане на дистанционно влизане

и отдалечен достъп до файлове, но тези допълнения не променят основната структура на

операционната система. Разпределената операционна система, за разлика от това, е тази, която изглежда на своите потребители като а

традиционната еднопроцесорна система, въпреки че всъщност е съставена от множество

процесори. Потребителите не трябва да са наясно къде се изпълняват техните програми или

къде се намират техните файлове; всичко това трябва да се обработва автоматично и ефективно от операционната система.

5.Пето поколение(1990-Наше време):Мобилни компютри

Първият истински мобилен телефон се появи в

1946 г. и тежи около 40 кг. Можете да го вземете, където и да отидете, стига

имал си кола, в която да го носиш.

Първият истински ръчен телефон се появи през 70-те години на миналия век и с приблизително един килограм грама беше положително леко перо. Той беше известен като „тухлата“. Днес проникването на мобилни телефони е близо до

90% от световното население. Можем да осъществяваме разговори не само с нашите преносими телефони

и ръчни часовници, но скоро с очила и други предмети за носене. Освен това,

телефонната част вече не е толкова интересна. Получаваме имейл, сърфиране в мрежата, текстови съобщения

нашите приятели, играят игри, се движат из натоварения трафик — и дори не мислят

два пъти за това.

Докато идеята за комбиниране на телефония и компютри в устройство, подобно на телефон

съществува от 70-те години на миналия век, първият истински смартфон се появи едва преди

средата на 1990-те, когато Nokia пусна N9000, който буквално комбинира две,

предимно отделни устройства: телефон и PDA (Персонален цифров асистент). През 1997 г.

Ericsson въведе термина смартфон за своя GS88 „Penelope“.

Сега, когато смартфоните станаха повсеместни, конкуренцията между

различни операционни системи е ожесточена и резултатът е дори по-малко ясен, отколкото в

PC свят. Към момента на писане Android на Google е доминиращата операционна система с iOS на Apple за секунда, но това не винаги е било така и всичко може

да бъде отново различен само след няколко години. Ако нещо е ясно в света на смарт телефоните, то е, че не е лесно да останеш крал на планината за дълго. В края на краищата повечето смартфони през първото десетилетие след създаването им са работели със Symbian OS. Това беше операционната система по избор за популярни марки като

Samsung, Sony Ericsson, Motorola и особено Nokia. Въпреки това, други операционни системи като Blackberry OS на RIM (въведена за смартфони през 2002 г.) и

iOS на Apple (пуснат за първия iPhone през 2007 г.) започна да навлиза в Symbian

пазарен дял. Мнозина очакваха, че RIM ще доминира на бизнес пазара, докато

iOS ще бъде кралят на потребителските устройства. Пазарният дял на Symbian спадна. През 2011 г. Nokia се отказа от Symbian и обяви, че ще се съсредоточи върху Windows Phone като основна платформа. Android, Linux-базирана операционна система, пусната от Google в

2008 г., за да изпревари всички свои съперници.

За производителите на телефони Android имаше предимството, че беше с отворен код

и се предлага при разрешителен лиценз. В резултат на това те биха могли да се занимават с него и

да го адаптират към собствения си хардуер с лекота. Освен това има огромна общност от разработчици, които пишат приложения, предимно на познатия език за програмиране Java.

Преглед на компютърния хардуер

1.Процесори

„Мозъкът“ на компютъра е процесорът. Извлича инструкции от паметта

и ги изпълнява. Основният цикъл на всеки процесор е да извлече първата инструкция

от паметта, декодирайте го, за да определите неговия тип и операнди, изпълнете го и след това

извличане, декодиране и изпълнение на следващите инструкции.

Всеки процесор има специфичен набор от инструкции, които може да изпълнява. Така x86

процесор не може да изпълнява ARM програми и ARM процесор не може да изпълнява

x86 програми.

- X86, AMD64 x64, CISC, RISC, MIPS
- Фотолитография
- Si Wafle
- CPU, FPGA, GPU, ASIC

Защото достъпът до паметта за получаване на инструкция или дума за данни отнема

много по-дълго от изпълнението на инструкция, всички процесори съдържат някои регистри вътре

за съхраняване на ключови променливи и временни резултати. По този начин наборът от инструкции обикновено съдържа инструкции за зареждане на дума от паметта в регистър и съхраняване на дума

от регистър в паметта. Други инструкции комбинират два операнда от регистри, памет или и двете в резултат, като например добавяне на две думи и съхраняване на резултата

в регистър или в паметта.

В допълнение към общите регистри, използвани за съхраняване на променливи и временни резултати, повечето компютри имат няколко специални регистри, които са видими за програмиста. Един от тях е програмният брояч, който съдържа адреса на паметта на следващата инструкция, която трябва да бъде извлечена. След като тази инструкция бъде извлечена,

броячът на програмата се актуализира, за да сочи към неговия наследник.

Друг регистър е указателят на стека, който сочи към върха на текущия

стек в паметта. Стекът съдържа един кадър за всяка процедура, която е била

влезе, но все още не е излязъл. Стековата рамка на процедурата съдържа тези входни параметри,

локални променливи и временни променливи, които не се съхраняват в регистри.

Още един регистър е PSW (Program Status Word). Този регистър съдържа кодовите битове на условието, които се задават от инструкции за сравнение, CPU

приоритет, режим (потребител или ядро) и различни други контролни битове. Потребителски програми

може нормално да чете целия PSW, но обикновено може да запише само някои от неговите полета.

PSW играе важна роля в системните повиквания и I/O.

Операционната система трябва да е напълно наясно с всички регистри. Когато времево мултиплексира процесора, операционната система често спира изпълняваната програма

(отново) стартирайте друг. Всеки път, когато спира работеща програма, операционната система

трябва да запишете всички регистри, за да могат да бъдат възстановени, когато програмата стартира по-късно. За да подобрят производителността, дизайнерите на процесора отдавна са изоставили простото

модел на извличане, декодиране и изпълнение на една инструкция в даден момент. Много модерни

CPU’s имат възможности за изпълнение на повече от една инструкция едновременно. За

например, процесорът може да има отделни единици за извличане, декодиране и изпълнение, така че while

изпълнява инструкция n, може също да бъде декодираща инструкция n + 1 и извличане на инструкция n + 2. Такава организация се нарича конвейер.

Суперскаларен процесор е дори по-усъвършенстван от конвейера.

В този дизайн присъстват множество изпълнителни единици, например един

за целочислена аритметика, една за аритметика с плаваща запетая и една за булеви операции. Две или повече инструкции се извличат наведнъж, декодират се и се изхвърлят в а

задържане на буфер, докато могат да бъдат изпълнени. Веднага след като единица за изпълнение стане

наличен, той търси в буфера за задържане, за да види дали има инструкция, която може да обработва, и ако е така, премахва инструкцията от буфера и я изпълнява.

Повечето процесори, с изключение на много простите, използвани във вградените системи, имат два

режими, режим на ядрото и потребителски режим.

Когато работи в режим на ядрото, процесорът може да изпълни всяка инструкция в своя набор от инструкции и да използва всяка характеристика на хардуера. На работния плот

и сървърни машини, операционната система обикновено работи в режим на ядрото, което го дава

достъп до пълния хардуер. На повечето вградени системи работи малка част

в режим на ядрото, като останалата част от операционната система работи в потребителски режим.

Потребителските програми винаги работят в потребителски режим, който позволява само изпълнение на подмножество от инструкции и достъп до подмножество от функции. Като цяло, всички

инструкции, включващи I/O и защита на паметта, са забранени в потребителски режим.

Задаването на бит за режим PSW за влизане в режим на ядрото също е забранено, разбира се.

За да получи услуги от операционната система, потребителска програма трябва да направи системно извикване, което се залавя в ядрото и извиква операционната система. КАПАНЪТ

инструкцията превключва от потребителски режим в режим на ядрото и стартира операционната система. Когато работата приключи, контролът се връща на потребителската програма на адрес

инструкцията след системното извикване. Струва си да се отбележи, че компютрите имат капани, различни от инструкцията за изпълнение на системно повикване. Повечето от другите капани са причинени от хардуера за предупреждение

на изключителна ситуация като опит за делене на 0 или с плаваща запетая

подлив. Във всички случаи операционната система получава контрол и трябва да реши какво да прави

направи. Понякога програмата трябва да бъде прекратена с грешка. Друг път на

грешката може да бъде игнорирана (недопълнено число може да бъде настроено на 0). И накрая, когато

програмата е обявила предварително, че иска да се справи с определени видове условия, контролът може да бъде прехвърлен обратно на програмата, за да й позволи да се справи с проблема.

Каква е разликата между Капана и Прекъсването?

Капанът е изключение в потребителски процес. Причинява се от делене с нула или невалиден достъп до паметта. Това също е обичайният начин за извикване на рутина на ядрото (системно извикване), защото те се изпълняват с по-висок приоритет от потребителския код. Обработката е синхронна (така че потребителският код е спрян и продължава след това). В известен смисъл те са "активни" - през повечето време кодът очаква капанът да се случи и разчита на този факт.

Прекъсването е нещо, генерирано от хардуера (устройства като твърд диск, графична карта, I/O портове и т.н.). Те са асинхронни (т.е. не се случват на предвидими места в потребителския код) или "пасивни", тъй като манипулаторът на прекъсвания трябва да изчака да се случат в крайна сметка.

Можете също така да видите trap като вид вътрешно прекъсване на CPU, тъй като манипулаторът за манипулатор на trap изглежда като манипулатор на прекъсване (регистрите и указателите на стека са запазени, има превключвател на контекста, изпълнението може да се възобнови в някои случаи, когато е спряло) .

Капаните и прекъсванията са тясно свързани. Капаните са вид изключение, а изключенията са подобни на прекъсванията.

Intel x86 дефинира две припокриващи се категории, векторни събития (прекъсвания срещу изключения) и класове на изключения (неизправности срещу прихващания срещу прекратяване).

Всички цитати в тази публикация са от версията от април 2016 г. на ръководството за разработчици на софтуер на Intel. За (окончателна и сложна) x86 перспектива препоръчвам да прочетете главата на SDM за обработка на прекъсвания и изключения.

Векторни събития

Векторни събития (прекъсвания и изключения) карат процесора да скочи в манипулатор на прекъсване след запазване на голяма част от състоянието на процесора (достатъчно, за да може изпълнението да продължи от тази точка по-късно).

Изключенията и прекъсванията имат идентификатор, наречен вектор, който определя към кой манипулатор на прекъсване прескача процесорът. Манипулаторите на прекъсвания са описани в таблицата с дескриптори на прекъсване.

Прекъсване

Прекъсванията възникват в произволни моменти по време на изпълнението на програма, в отговор на сигнали от хардуера. Системният хардуер използва прекъсвания за обработка на външни за процесора събития, като например заявки за обслужване на периферни устройства. Софтуерът може също да генерира прекъсвания чрез изпълнение на инструкцията INT n.

Изключения

Изключения възникват, когато процесорът открие състояние на грешка по време на изпълнение на инструкция, като деление на нула. Процесорът открива различни състояния на грешки, включително нарушения на защитата, грешки в страницата и вътрешни грешки в машината.

Класификации на изключения

Изключенията се класифицират като грешки, капанчета или прекъсвания в зависимост от начина, по който са докладвани и дали инструкцията, причинила изключението, може да бъде рестартирана без загуба на непрекъснатост на програмата или задачата.

Резюме: уловките увеличават показалеца на инструкциите, грешките не и прекратяванията „експлодират“.

Капан

Trap е изключение, което се докладва веднага след изпълнението на инструкцията за прихващане. Капаните позволяват изпълнението на програма или задача да бъде продължено без загуба на непрекъснатост на програмата. Адресът за връщане на манипулатора на trap сочи към инструкцията, която трябва да се изпълни след инструкцията за прихващане.

Грешка

Грешка е изключение, което обикновено може да бъде коригирано и което, след като бъде коригирано, позволява програмата да бъде рестартирана без загуба на непрекъснатост. Когато се докладва грешка, процесорът възстановява състоянието на машината до състоянието преди началото на изпълнение на инструкцията за грешка. Адресът за връщане (запазено съдържание на регистрите CS и EIP) за манипулатора на грешки сочи към инструкцията за грешка, а не към инструкцията, следваща инструкцията за грешка.

Пример: Грешка в страницата често е възстановима. Част от адресното пространство на приложението може да е била разменена на диск от RAM. Приложението ще задейства грешка в страницата, когато се опита да получи достъп до паметта, която е била разменена. Ядрото може да изтегли тази памет от диск към ram и ръчния контрол обратно към приложението. Приложението ще продължи там, където е спряло (при грешката инструкция, която е осъществявала достъп до разменена памет), но този път достъпът до паметта трябва да успее без грешка.

Манипулатор на грешки с незаконни инструкции, който емулира инструкции с плаваща запетая или други липсващи инструкции, ще трябва ръчно да увеличи адреса за връщане, за да получи необходимото поведение, подобно на капан, след като види дали инструкцията за грешка е тази, която може да обработи. x86 #UD е "неизправност", а не "капан". (Обработчикът ще се нуждае от указател към инструкцията за грешка, за да разбере коя е инструкцията.)

Прекратяване

Прекратяването е изключение, което не винаги съобщава точното местоположение на инструкцията, причиняваща изключението, и не позволява рестартиране на програмата

или задачата, която е причинила изключението. Прекъсванията се използват за докладване на сериозни грешки, като хардуерни грешки и непоследователни или незаконни стойности в системните таблици.

Крайни калъфи

Извиканите от софтуера прекъсвания (задействани от инструкцията INT) се държат като капан. Инструкцията завършва, преди процесорът да запази състоянието си и да премине към манипулатора на прекъсване.

Многонишкови и многоядрени чипове

Законът на Мур гласи, че броят на транзисторите на чипа се удвоява на всеки 18

месеци. Този „закон“ не е някакъв закон на физиката, като запазването на импулса, а е наблюдение на съоснователя на Intel Гордън Мур за това колко бързо инженерите на процесите в полупроводниковите компании са в състояние да свият транзисторите си.

Изобилието от транзистори води до проблем: какво да правим с всички

тях? Видяхме един подход по-горе: суперскаларни архитектури с множество функционални единици. Но с увеличаване на броя на транзисторите е възможно дори повече. едно

очевидно нещо, което трябва да направите, е да поставите по-големи кеш памети на чипа на процесора. Това определено се случва, но в крайна сметка точката на намаляваща възвръщаемост ще бъде достигната.

Очевидната следваща стъпка е да се копират не само функционалните единици, но и

част от логиката на управлението. Intel Pentium 4 представи това свойство, наречено

multithreading или hyperthreading (името на Intel за него), към x86 процесора и

няколко други CPU чипа също го имат – включително SPARC, Power5, Intel

Xeon и семейството на Intel Core. В първо приближение, това, което прави, е да позволява

CPU, за да задържи състоянието на две различни нишки и след това да превключва напред-назад на а

наносекундна времева скала. (Нишка е вид олекотен процес, който от своя страна,

е работеща програма)

Например, ако един

от процесите трябва да прочете дума от паметта (което отнема много часовник

цикли), многонишков процесор може просто да превключи към друга нишка. Многонишковост

не предлага истински паралелизъм. Изпълнява се само един процес в даден момент, но

времето за превключване на нишките се намалява до порядъка на наносекунда.

Многонишковостта има последици за операционната система, тъй като всяка нишка

се появява на операционната система като отделен процесор. Помислете за система с две

реални процесори, всеки с две нишки.

Освен многонишковостта, много процесорни чипове вече имат четири, осем или повече пълни процесора или ядра.

Някои процесори, като Intel Xeon Phi и Tilera TilePro, вече разполагат с повече от 60 ядра на един чип. Използване на такъв многоядрен

чип определено ще изисква многопроцесорна операционна система.

Между другото, по отношение на чистите числа, нищо не може да превъзхожда съвременния GPU (Graph ics Processing Unit). Графичният процесор е процесор с, буквално, хиляди малки ядра.

Те са много добри за много малки изчисления, извършени паралелно, като рендиране

полигони в графичните приложения. Не са толкова добри в серийните задачи. Те са

също трудно се програмира. Въпреки че графичните процесори могат да бъдат полезни за операционни системи (например криптиране или обработка на мрежов трафик), не е вероятно голяма част от операционната

самата система ще работи на графичните процесори.

2.Памет

Вторият основен компонент във всеки компютър е паметта. В идеалния случай паметта трябва да е изключително бърза (по-бърза от изпълнението на инструкция, така че процесорът да е

не се поддържа от паметта), изобилно голям и евтин. Никоя настояща технология не удовлетворява всички тези цели, така че се използва различен подход. Системата с памет е изградена като йерархия от слоеве.

Най-горните слоеве

имат по-висока скорост, по-малък капацитет и по-висока цена на бит от по-ниските,

често с фактори от милиард или повече.

Най-горният слой се състои от вътрешни регистри на процесора. Те са направени от

от същия материал като процесора и по този начин са също толкова бързи, колкото и процесора. следователно,

няма забавяне при достъпа до тях.

Обикновено 32 × 32 бита на 32-битов процесор и 64 × 64 бита на 64-битов процесор. По-малко от

1 KB и в двата случая. Програмите трябва да управляват регистрите (т.е. да решават какво да пазят

в тях) самите, в софтуера.

Следва кеш паметта, която се контролира предимно от хардуера.

Основната памет е разделена на кеш редове, обикновено 64 байта, с адреси 0

до 63 в кеш ред 0, 64 до 127 в кеш ред 1 и т.н. Най-използваната

кеш линиите се съхраняват във високоскоростен кеш, разположен вътре или много близо до процесора.

Когато програмата трябва да прочете дума от паметта, хардуерът на кеша проверява, за да види

ако необходимият ред е в кеша. Ако е, наречено кеш хит, заявката е удовлетворена

от кеша и не се изпраща заявка за памет по шината към основната памет.

Попаденията в кеша обикновено отнемат около два такта. Пропуските в кеша трябва да отидат

памет, със значително намаляване на времето. Кеш паметта е ограничена по размер поради нейната

висока цена. Някои машини имат две или дори три нива на кеш, всяко едно по-бавно

и по-голям от този преди него.

Кеширането играе важна роля в много области на компютърните науки, не само в кеширането

линии RAM. Винаги, когато ресурсът може да бъде разделен на части, някои от които са

използван много по-интензивно от други, кеширането често се използва за подобряване на производителността. Операционните системи го използват през цялото време. Например повечето операционни системи

съхранявайте (частици от) силно използвани файлове в основната памет, за да избегнете необходимостта да ги извличате

от диска многократно.

Кешовете са толкова добра идея, че съвременните процесори имат два от тях. Първият

ниво или L1 кеш винаги е вътре в процесора и обикновено подава декодирани инструкции

в двигателя за изпълнение на процесора.

При многоядрените чипове дизайнерите трябва да решат къде да поставят кешовете. В

Фиг. 1-8(a), един L2 кеш се споделя от всички ядра. Този подход се използва в

Многоядрени чипове на Intel. Обратно, на фиг. 1-8(b), всяко ядро има свой собствен L2 кеш.

Този подход се използва от AMD. Всяка стратегия има своите плюсове и минуси. Например,

споделеният L2 кеш на Intel изисква по-сложен кеш контролер, но

Начинът на AMD прави поддържането на последователност на L2 кеша по-трудно.

Основната памет е следващата в йерархията на фиг. 1-9. Това е работният кон

на системата памет. Основната памет обикновено се нарича RAM (произволен достъп

памет). Старите хора понякога го наричат основна памет, защото компютрите в

50-те и 60-те години на миналия век са използвали малки намагнетизиращи се феритни ядра за основна памет. Те имат

изчезна от десетилетия, но името си остава.

В допълнение към основната памет, много компютри имат малко количество енергонезависима памет с произволен достъп. За разлика от RAM, енергонезависимата памет не губи

съдържанието му, когато захранването е изключено. ROM (памет само за четене) е програмиран фабрично и не може да бъде променен след това. Той е бърз и евтин. На някои компютри зареждащият зареждане, използван за стартиране на компютъра, се съдържа в ROM. Освен това някои I/O карти идват с ROM за управление на ниско ниво на управление на устройства.

EEPROM (Electrically Erasable PROM) и флаш паметта също са енергонезависими, но за разлика от ROM могат да бъдат изтрити и презаписани.

Флаш паметта често се използва и като носител за съхранение в преносими електронни устройства. Той служи като филм в цифровите фотоапарати и като диск в преносимата музика

играчи, да назовем само две употреби. Флаш паметта е средна по скорост между тях

RAM и диск. Освен това, за разлика от дисковата памет, ако се изтрие твърде много пъти, тя се износва

навън.

Още един вид памет е CMOS, която е непостоянна. Много компютри използват

CMOS памет за съхраняване на текущия час и дата. CMOS паметта и

часовниковата верига, която увеличава времето в нея, се захранват от малка батерия, така че

времето се актуализира правилно, дори когато компютърът е изключен. CMOS паметта може също да съхранява конфигурационните параметри, като например от кой диск да се стартира.

CMOS се използва, защото консумира толкова малко енергия, че оригиналната фабрично инсталирана

батерията често издържа няколко години. Въпреки това, когато започне да се проваля, компютърът

може да изглежда, че има болест на Алцхаймер, забравяйки неща, за които е известен

години, като например от кой твърд диск да стартирате.

DDR - Double Data Rate

При изчисленията компютърна шина, работеща с двойна скорост на данни (DDR), прехвърля данни както за нарастващия, така и за падащия ръб на тактовия сигнал.[1] Това е известно още като двойно изпомпване, двойно изпомпване и двоен преход. Терминът режим на превключване се използва в контекста на флаш паметта NAND.

Връзка на честотната лента и честотата

Описването на честотната лента на шина с двойно изпомпване може да бъде объркващо. Всеки ръб на часовника се нарича бийт, с два удара (един възходящ и един по-нисък) на цикъл. Технически, херцът е единица цикли в секунда, но много хора се отнасят към броя на трансферите в секунда. Внимателната употреба обикновено говори за "500 MHz, двойна скорост на данни" или "1000 MT/s", но много от тях се отнасят небрежно към "1000 MHz шина", въпреки че нито един сигнал не цикли по-бързо от 500 MHz.

DDR SDRAM популяризира техниката за позоваване на честотната лента на шината в мегабайти в секунда, произведението на скоростта на трансфер и ширината на шината в байтове. DDR SDRAM, работещ с тактова честота 100 MHz, се нарича DDR-200 (след скоростта на пренос на данни от 200 MT/s), а 64-битов (8-байтов) DIMM, работещ с тази скорост на данни, се нарича PC-1600, след неговия 1600 MB/s пикова (теоретична) честотна лента. По същия начин скоростта на предаване 1,6 GT/s DDR3-1600 се нарича PC3-12800.

Някои примери за популярни обозначения на DDR модули:

Names	Memory clock	I/O bus clock	Transfer rate	Theoretical bandwidth
DDR-200, PC-1600	100 MHz	100 MHz	200 MT/s	1.6 GB/s
DDR-400, PC-3200	200 MHz	200 MHz	400 MT/s	3.2 GB/s
DDR2-800, PC2-6400	200 MHz	400 MHz	800 MT/s	6.4 GB/s
DDR3-1600, PC3-12800	200 MHz	800 MHz	1600 MT/s	12.8 GB/s
DDR4-2400, PC4-19200	300 MHz	1200 MHz	2400 MT/s	19.2 GB/s
DDR4-3200, PC4-25600	400 MHz	1600 MHz	3200 MT/s	25.6 GB/s
DDR5-4800, PC5-38400	300 MHz	2400 MHz	4800 MT/s	38.4 GB/s
DDR5-6400, PC5-51200	400 MHz	3200 MHz	6400 MT/s	51.2 GB/s

DDR SDRAM използва сигнализиране с двойна скорост на данни само по линиите за данни. Адресните и контролните сигнали все още се изпращат към DRAM веднъж на такт (за да бъдем точни, на нарастващия фронт на часовника), а параметрите на времето, като CAS латентност, се определят в тактови цикли. Някои по-рядко срещани DRAM интерфейси, по-специално LPDDR2, GDDR5 и XDR DRAM, изпращат команди и адреси, използвайки двойна скорост на данни. DDR5 използва две 7-битови шини за команди/адреси с двойна скорост на данни към всеки DIMM, където регистриран чип на драйвера на часовника се преобразува в 14-битова SDR шина към всеки чип памет.

DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM и DDR4 SDRAM (които всички използват DDR сигнализация)

1.DDR SDRAM

Синхронна динамична памет с произволен достъп с двойна скорост на данни (DDR SDRAM) е клас интегрални схеми с памет, използвани в компютрите, синхронна динамична памет с произволен достъп с двойна скорост (DDR). DDR SDRAM, наричана също със задна дата DDR1 SDRAM, е заменена от DDR2 SDRAM, DDR3 SDRAM, DDR4 SDRAM и DDR5 SDRAM. Нито един от неговите наследници не е съвместим напред или назад с DDR1 SDRAM, което означава, че модулите памет DDR2, DDR3, DDR4 и DDR5 няма да работят в дънни платки, оборудвани с DDR1, и обратно.

2.DDR2 SDRAM

Синхронна динамична памет с произволен достъп с двойна скорост на данни 2 (DDR2 SDRAM) е интерфейс за синхронна динамична памет с произволен достъп (SDRAM) с двойна скорост на данни (DDR). Той замени оригиналната спецификация на DDR SDRAM, а самият той беше заменен от DDR3 SDRAM (лансиран през 2007 г.). DDR2 DIMM модулите не са нито съвместими напред с DDR3, нито обратно съвместими с DDR.

В допълнение към двойното изпомпване на шината за данни, както в DDR SDRAM (прехвърляне на данни за нарастващите и падащи ръбове на тактовия сигнал на шината), DDR2 позволява по-висока скорост на шината и изисква по-ниска мощност, като работи на вътрешния часовник на половината от скоростта на шината за данни . Двата фактора се комбинират, за да произведат общо четири трансфера на данни на вътрешен такт.

3.DDR3 SDRAM

Синхронна динамична памет с произволен достъп с двойна скорост на данни 3 (DDR3 SDRAM) е вид синхронна динамична памет с произволен достъп (SDRAM) с интерфейс с висока честотна лента („двойна скорост на данни“) и се използва от 2007 г. по-висикоскоростният наследник на DDR и DDR2 и предшественик на DDR4 чиповете с синхронна динамична памет с произволен достъп (SDRAM). DDR3 SDRAM не е съвместим нито напред, нито назад с който и да е по-ранен тип памет с произволен достъп (RAM) поради различни сигнални напрежения, тайминги и други фактори.

DDR3 е спецификация на интерфейса на DRAM. Действителните DRAM масиви, които съхраняват данните, са подобни на по-ранните типове, с подобна производителност. Основното предимство на DDR3 SDRAM пред неговия непосредствен предшественик, DDR2 SDRAM, е способността му да прехвърля данни с два пъти по-висока скорост (осем пъти по-висока от скоростта на вътрешните си памети), което позволява по-висока честотна лента или пикова скорост на данни.

Стандартът DDR3 позволява капацитет на DRAM чип до 8 гигабита (Gbit) и до четири ранга от 64 бита всеки за общ максимум 16 гигабайта (GB) на DDR3 DIMM. Поради хардуерно ограничение, което не е фиксирано до Ivy Bridge-E през 2013 г., повечето по-стари процесори на Intel поддържат само до 4-Gbit чипове за 8 GB DIMM (чипсетите Core 2 DDR3 на Intel поддържат само до 2 Gbit). Всички AMD процесори поддържат правилно пълната спецификация за 16 GB DDR3 DIMM.

Общ преглед

В сравнение с DDR2 паметта, DDR3 паметта използва по-малко енергия. Освен това някои производители предлагат използването на транзистори с двойна врата за намаляване на изтичането на ток.

Според JEDEC, 1,575 волта трябва да се счита за абсолютен максимум, когато стабилността на паметта е основното съображение, като например в сървъри или други критични за мисия устройства. В допълнение, JEDEC заявява, че модулите с памет трябва да издържат до 1,80 волта, преди да получат трайна повреда, въпреки че не се изисква да функционират правилно на това ниво.

Друго предимство е неговият буфер за предварително извличане, който е с дълбочина 8 пакета. За разлика от тях, буферът за предварително извличане на DDR2 е с 4 пакета, а буферът за предварително извличане на DDR е с 2 пакета. Това предимство е позволяваща технология в скоростта на трансфер на DDR3.

DDR3 модулите могат да прехвърлят данни със скорост от 800–2133 MT/s, използвайки както нарастващи, така и падащи фронтове на 400–1066 MHz I/O часовник. Това е два пъти по-висока скорост на предаване на данни на DDR2 (400–1066 MT/s при 200–533 MHz I/O часовник) и четири пъти по-висока от скоростта на DDR (200–400 MT/s при използване на 100–200 MHz I/O часовник) . Високопроизводителната графика беше първоначалният драйвер за такива изисквания за честотна лента, където се изисква пренос на данни с висока честотна лента между фреймбуферите.

Тъй като херцът е мярка за цикли в секунда и няма цикли на сигнал по-често от всеки друг трансфер, описването на скоростта на предаване в единици MHz е технически неправилно, макар и много често. Също така е подвеждащо, защото различните тайминги на паметта са дадени в единици тактови цикли, които са наполовина по-ниски от скоростта на трансфер на данни.

DDR3 използва същия стандарт за електрическа сигнализация като DDR и DDR2, терминална логика от серия Stub, макар и при различни моменти и напрежения. По-конкретно, DDR3 използва SSTL_15.

През февруари 2005 г. Samsung демонстрира първия прототип на паметта DDR3, с капацитет от 512 Mb и пропускателна способност от 1,066 Gbps. Продукти под формата на дънни платки се появиха на пазара през юни 2007 г., базирани на чипсет P35 "Bearlake" на Intel с DIMM модули с честотна лента до DDR3-1600 (PC3-12800). Intel Core i7, пуснат през ноември 2008 г., се свързва директно към паметта, а не чрез чипсет.

Процесорите Core i7, i5 и i3 първоначално поддържаха само DDR3. Процесорите AM3 Phenom II X4 на AMD, пуснати през февруари 2009 г., бяха първите, които поддържаха DDR3 (като същевременно поддържаха DDR2 за обратна съвместимост).

Двулинейни модули памет

DDR3 модулите памет с двойна линия (DIMM) имат 240 извода и са електрически несъвместими с DDR2. Ключов прорез, разположен по различен начин в DDR2 и DDR3 DIMM модули, предотвратява случайната им размяна. Те не само са с различни ключове, но DDR2 има заоблени прорези отстрани, а DDR3 модулите имат квадратни прорези отстрани. DDR3 SO-DIMM имат 204 извода.

За микроархитектурата Skylake Intel също така е проектирал SO-DIMM пакет, наречен UniDIMM, който може да използва DDR3 или DDR4 чипове. След това интегрираният контролер на паметта на процесора може да работи и с двете. Целта на UniDIMM е да се справят с прехода от DDR3 към DDR4, където цената и наличността може да направят желателно смяната на типа RAM. UniDIMM имат същите размери и брой изводи като обикновените DDR4 SO-DIMM, но прорезът е поставен по различен начин, за да се избегне случайно използване в несъвместим DDR4 SO-DIMM гнездо.

Латенции

Закъсненията на DDR3 са числено по-високи, тъй като тактовите цикли на I/O шината, чрез които се измерват, са по-кратки; действителният интервал от време е подобен на латентностите на DDR2, около 10 ns. Има известно подобрение, тъй като DDR3 обикновено използва по-нови производствени процеси, но това не е пряко причинено от промяната към DDR3.

CAS латентност (ns) = 1000 × CL (цикли) ÷ тактова честота (MHz) = 2000 × CL (цикли) ÷ скорост на предаване (MT/s)

Докато типичните латентности за устройство JEDEC DDR2-800 бяха 5-5-5-15 (12,5 ns), някои стандартни латентности за устройства JEDEC DDR3 включват 7-7-7-20 за DDR3-1066 (13,125 ns) и 8- 8-8-24 за DDR3-1333 (12 ns).

Както при по-ранните поколения памет, по-бързата DDR3 памет стана достъпна след пускането на първоначалните версии. Паметта DDR3-2000 с латентност 9-9-9-28 (9 ns) беше налична във времето, за да съвпадне с изданието на Intel Core i7 в края на 2008 г., докато по-късните разработки направиха DDR3-2400 широко достъпен (с CL 9–12 цикъла = 7,5–10 ns) и налична скорост до DDR3-3200 (с CL 13 цикъла = 8,125 ns).

Консумация на енергия

Консумацията на енергия на отделните SDRAM чипове (или, в допълнение, DIMM) варира в зависимост от много фактори, включително скорост, тип използване, напрежение и т.н. Power Advisor на Dell изчислява, че 4 GB ECC DDR1333 RDIMM модули използват около 4 W всеки. За разлика от тях, по-модерната част, ориентирана към настолни компютри, 8 GB, DDR3/1600 DIMM, е оценена на 2,58 W, въпреки че е значително по-бърза.

Модули

List of standard DDR3 SDRAM modules

Name			Chip		Bus			Timings	
Standard	Type	Module	Clock rate (MHz)	Cycle time (ns)[22]	Clock rate (MHz)	Transfer rate (MT/s)	Bandwidth (MB/s)	CL-T_{RCD}-T_{RP}	CAS latency (ns)
DDR3-800	D	PC3-6400	100	10	400	800	6400	5-5-5	12.5
	E							6-6-6	15
DDR3-1066	E	PC3-8500	133⅓	7.5	533⅓	1066⅔	8533⅓	6-6-6	11.25
	F							7-7-7	13.125
	G							8-8-8	15
DDR3-1333	F*	PC3-10600	166⅔	6	666⅔	1333⅓	10666⅔	7-7-7	10.5
	G							8-8-8	12
	H							9-9-9	13.5
	J*							10-10-10	15
DDR3-1600	G*	PC3-12800	200	5	800	1600	12800	8-8-8	10
	H							9-9-9	11.25
	J							10-10-10	12.5
	K							11-11-11	13.75
DDR3-1866	DDR3-1866J*	PC3-14900	233⅓	4.286	933⅓	1866⅔	14933⅓	10-10-10	10.56
	DDR3-1866K							11-11-11	11.786
	DDR3-1866L							12-12-12	12.857
	DDR3-1866M*							13-13-13	13.929
DDR3-2133	DDR3-2133K*	PC3-17000	266⅔	3.75	1066⅔	2133⅓	17066⅔	11-11-11	10.313
	DDR3-2133L							12-12-12	11.25
	DDR3-2133M							13-13-13	12.188
	DDR3-2133N*							14-14-14	13.125

* по избор

DDR3-xxx обозначава скоростта на предаване на данни и описва DDR чиповете, докато PC3-xxxx означава теоретична честотна лента (с последните две цифри съкратени) и се използва за описание на сглобени DIMM. Пропускателната способност се изчислява, като се вземат трансфери в секунда и се умножава по осем. Това е така, защото модулите памет DDR3 прехвърлят данни по шина с ширина 64 бита данни и тъй като един байт се състои от 8 бита, това се равнява на 8 байта данни на трансфер.

С две прехвърляния на цикъл на четворен тактов сигнал, 64-битов DDR3 модул може да постигне скорост на трансфер до 64 пъти по-висока от тактовата скорост на паметта. Тъй като данните се прехвърлят 64 бита наведнъж на модул памет, DDR3 SDRAM дава скорост на трансфер (тактова честота на паметта) × 4 (за умножител на тактова честота на шината) × 2 (за скорост на данни) × 64 (брой прехвърлени битове) / 8 (брой битове в байт). Така с тактова честота на паметта от 100 MHz, DDR3 SDRAM дава максимална скорост на трансфер от 6400 MB/s.

Скоростта на данни (в MT/s) е два пъти по-висока от тактовата честота на I/O шината (в MHz) поради двойната скорост на данни на DDR паметта. Както беше обяснено по-горе, честотната лента в MB/s е скоростта на данни, умножена по осем.

CL – CAS Цикли на закъснение, между изпращане на адрес на колона към паметта и началото на данните в отговор

tRCD – Циклите на часовника между активиране на ред и четене/записване

tRP – Цикли на часовника между предварително зареждане на ред и активиране

Дробните честоти обикновено се закръглят надолу, но закръгляването до 667 е често срещано, тъй като точното число е 666⅔ и закръгляването до най-близкото цяло число. Някои производители също закръглят до определена точност или вместо това закръглят нагоре. Например паметта PC3-10666 може да бъде посочена като PC3-10600 или PC3-10700.

Забележка: Всички изброени по-горе елементи са посочени от JEDEC като JESD79-3F. Всички скорости на RAM данни между или над тези изброени спецификации не са стандартизирани от JEDEC – често те са просто оптимизации на производителя, използващи чипове с по-висок толеранс или пренапрежение. От тези нестандартни

спецификации, най-високата отчетена достигната скорост е еквивалентна на DDR3-2544 към май 2010 г.

Алтернативно именуване: DDR3 модулите често са неправилно етикетирани с префикс PC (вместо PC3), по маркетингови причини, последван от скоростта на данни. Съгласно тази конвенция PC3-10600 е посочен като PC1333.

4.DDR4 SDRAM

Синхронна динамична памет с произволен достъп с двойна скорост на данни 4 (DDR4 SDRAM) е вид синхронна динамична памет с произволен достъп с интерфейс с висока честотна лента („двойна скорост на данни“).

Пуснат на пазара през 2014 г., той е вариант на динамична памет с произволен достъп (DRAM), някои от които се използват от началото на 70-те години на миналия век, и по-високоскоростен наследник на технологиите DDR2 и DDR3.

DDR4 не е съвместим с по-ранен тип памет с произволен достъп (RAM) поради различно напрежение на сигнализиране и физически интерфейс, освен други фактори.

DDR4 SDRAM беше пусната на публичния пазар през Q2 2014 г., като се фокусира върху ECC паметта, докато модулите без ECC DDR4 станаха налични през Q3 2014, придружавайки пускането на процесори Haswell-E, които изискват DDR4 памет.

Дискове

Следващият в йерархията е магнитният диск (твърд диск). Дисковото съхранение е две поръчки

по-евтино от RAM за бит и често с два порядъка по-голямо

също така. Единственият проблем е, че времето за произволен достъп до данните за него е близо

с три порядъка по-бавно.

Дискът се състои от една или повече метални пластини, които се въртят на 5400, 7200, 10 800

RPM или повече.

- SCSI – Small Computer System Interface
- LBA – Linear Block Address

- Track, cylinder, cluster, sector – 512 bytes

Механично рамо се завърта над чиниите от ъгъла, подобно

до рамото на пикапа на стар фонограф с 33 оборота в минута за възпроизвеждане на винилови плочи.

Информацията се записва на диска в серия от концентрични кръгове. Във всеки дан

позиция на ръката, всяка от главите може да разчете пръстеновидна област, наречена писта. Заедно с нея всички писти за дадено положение на ръката образуват цилиндър.

Всяка песен е разделена на определен брой сектори, обикновено 512 байта на сектор. При съвременните дискове външните цилиндри съдържат повече сектори от вътрешните.

Преместването на рамото от един цилиндър към следващия отнема около 1 мс. Преместването му в а

произволният цилиндър обикновено отнема от 5 до 10 msec, в зависимост от устройството. Веднъж

рамото е на правилния път, задвижването трябва да изчака необходимия сектор да се завърти под

главата, допълнително забавяне от 5 msec до 10 msec, в зависимост от оборотите на устройството.

След като секторът е под главата, четенето или записването се извършва със скорост от 50 MB/sec

на дискове от нисък клас до 160 MB/sec на по-бързи.

Понякога ще чуете хората да говорят за дискове, които всъщност изобщо не са дискове,

като SSD дискове (Solid State Disks). SSD дисковете нямат движещи се части, не съдържат

плочи под формата на дискове и съхраняват данни във (Flash) памет. Единствените начини за влизане

което приличат на дискове е, че съхраняват и много данни, които не се губят

когато захранването е изключено.

Много компютри поддържат схема, известна като виртуална памет, което ще направим

обсъждат надълго и нашироко в гл. 3. Тази схема дава възможност за стартиране на програми

по-големи от физическата памет, като ги поставите на диска и използвате основната памет

като вид кеш за най-тежко изпълнените части. Тази схема изисква повторно съпоставяне на адресите на паметта в движение, за да преобразува адреса, генериран от програмата, във физическия адрес в RAM, където се намира думата. Това картографиране е

извършва се от част от процесора, наречена MMU (Memory Management Unit).

Наличието на кеширане и MMU може да има голямо влияние върху производителността. В многопрограмна система, при преминаване от една програма към

друг, понякога наричан превключвател на контекста, може да се наложи да изтриете всички модифицирани блокове от кеша и да промените регистрите за преобразуване в MMU. Двамата от

това са скъпи операции и програмистите усилено се опитват да ги избягват. Ние ще

вижте някои от последиците от техните тактики по-късно.

SSD – Solid State Device

Nand Flash Data Storage Overview – SLC, MLC and TLC

SLC – Single Level Cell

MLC – Multi Level Cell

TLC – Tri Level Cell

Head Actuator

- Stepper Motor(Стъпков мотор)
- Voice coil(Гласова намотка)

NVMe

NVM Express (NVMe) или спецификация на интерфейса на хост контролера на енергонезависимата памет (NVMHCIS) е отворена спецификация на интерфейса на логическото устройство за достъп до енергонезависимата памет на компютъра, обикновено свързана чрез PCI Express (PCIe) шина. Акронимът NVM означава енергонезависима памет, която често е флаш памет NAND, която се предлага в няколко физически форм-фактора, включително твърдотелни устройства (SSD), PCI Express (PCIe) карти за добавяне и карти M.2, наследник на mSATA карти. NVM Express, като интерфейс на логическо устройство, е проектиран да се възползва от ниската латентност и вътрешния паралелизъм на твърдите устройства за съхранение.

По своя дизайн NVM Express позволява на хардуера и софтуера на хоста да използват напълно възможните нива на паралелизъм в съвременните SSD дискове. В резултат на това NVM Express намалява разходите за I/O и носи различни подобрения в производителността в сравнение с предишни интерфейси на логически устройства,

включително множество дълги опашки за команди и намалена латентност. Предишните интерфейсни протоколи като AHCI бяха разработени за използване с много по-бавни твърди дискове (HDD), където съществува много дълго забавяне (в сравнение с операциите на процесора) между заявка и трансфер на данни, където скоростите на данните са много по-ниски от скоростите на RAM и където ротацията на диска и времето за търсене пораждат допълнителни изисквания за оптимизация.

Звуковата намотка замени стъпковите двигатели.

4.I/O Устройства (SATA)

Процесорът и паметта не са единствените ресурси на операционната система

трябва да управлява. I/O устройствата също взаимодействат силно с операционната система. Тъй като ние

1-6, I/O устройствата обикновено се състоят от две части: контролер и самото устройство. Контролерът е чип или набор от чипове, които физически контролират устройството. Той приема команди от операционната система, например, за четене на данни

от устройството и ги изпълнява.

Другата част е самото устройство. Устройствата имат доста прости интерфейси, както защото не могат да направят много, така и за да ги направят стандартни. Последното е

необходими, така че всеки SAT A дисков контролер да може да обработва всеки SAT A диск, например.

SATA означава Serial ATA, а AT A от своя страна означава AT Attachment. В случай

любопитно ви е какво означава AT, това беше второто поколение „Personal“ на IBM

Компютърни усъвършенствани технологии“, изградени около тогавашните изключително мощни 6-MHz

80286 процесор, който компанията представи през 1984 г.

AHCI – Advanced Hot Controller Interface

5.Buses

Тази система има много шини (напр. кеш, памет, PCIe, PCI, USB, SATA и

DMI), всеки с различна скорост на предаване и функция. Операционната система трябва

бъдете наясно с всички тях за конфигуриране и управление. Основният автобус е

PCIe (Peripheral Component Interconnect Express) шина.

PCIe шината е изобретена от Intel като наследник на по-старата PCI шина, която

на свой ред беше заместител на оригиналната ISA (Industry Standard Architecture)

автобус. Способен да прехвърля десетки гигабита в секунда, PCIe е много по-бърз от

неговите предшественици. Той също е много различен по природа. До създаването си през 2004 г. повечето

автобусите бяха успоредни и споделени. Архитектурата на споделена шина означава, че множество устройства използват едни и същи проводници за прехвърляне на данни. По този начин, когато няколко устройства имат данни за

изпратите, имате нужда от арбитър, който да определи кой може да използва автобуса. За разлика от това, PCIe

използва специални връзки от точка до точка.

Архитектура на паралелна шина като

използвано в традиционния PCI означава, че изпращате всяка дума от данни по множество проводници.

Например, в обикновените PCI шини едно 32-битово число се изпраща по 32 паралелни

проводници. За разлика от това, PCIe използва архитектура на серийната шина и изпраща всички битове в съобщение чрез една връзка, известна като лента, подобно на мрежа

пакет.

Това е много по-просто, защото не е нужно да гарантирате, че всички 32 бита

пристигане на дестинацията точно по същото време. Все още се използва паралелизъм, т.к

можете да имате няколко ленти успоредно. Например, можем да използваме 32 ленти за пренасяне

32 съобщения паралелно. Тъй като скоростта на периферни устройства като мрежови карти и

графичните адаптери се увеличават бързо, стандартът PCIe се надгражда на всеки 3-5 години.

Например, 16 ленти на PCIe 2.0 предлагат 64 гигабита в секунда. Надстройка до PCIe

3.0 ще ви даде два пъти по-голяма скорост, а PCIe 4.0 ще удвои тази скорост отново.

В тази конфигурация процесорът говори с паметта през бърза DDR3 шина, с външно графично устройство през PCIe и с всички други устройства чрез хъб през DMI

(Direct Media Interface) шина.

USB (Universal Serial Bus) е изобретен за свързване на всички бавни I/O устройства, като клавиатура и мишка, към компютъра.

Обаче обаждането на модерно USB 3.0 устройство, бръмчащо при 5 Gbps „бавно“ може да не е естествено за

поколението, което израства с 8-Mbps ISA като основна шина в първите компютри на IBM.

USB използва малък конектор с четири до единадесет проводника (в зависимост от версията),

някои от които доставят електрическо захранване на USB устройствата или се свързват към земята.

USB е централизирана шина, в която коренно устройство анкетира всички I/O устройства на всеки 1

msec, за да видите дали имат някакъв трафик. USB 1.0 може да се справи с общо натоварване от 12

Mbps, USB 2.0 увеличи скоростта до 480 Mbps, а USB 3.0 най-високите стойности не по-малко от

5 Gbps. Всяко USB устройство може да бъде свързано към компютър и то ще функционира незабавно, без да се налага рестартиране, нещо, което се изисква предварително USB устройства, много

до ужас на поколение разочаровани потребители.

SCSI (Small Computer System Interface) шината е високопроизводителна шина

предназначени за бързи дискове, скенери и други устройства, които се нуждаят от значителна ширина на лентата. В днешно време ги намираме предимно в сървъри и работни станции. Те могат да бягат

до 640 MB/сек.

Преди plug and play всяка I/O карта имаше фиксирано ниво на заявка за прекъсване и фиксирани адреси за своите I/O регистри. Например, клавиатурата беше с прекъсване 1 и използваше I/O адреси от 0x60 до 0x64, контролерът на флопи диска беше с прекъсване 6 и използваше I/O

адреси от 0x3F0 до 0x3F7 и принтерът е прекъснат 7 и използва I/O адреси

0x378 до 0x37A и т.н.

Дотук добре. Проблемът се появи, когато потребителят купи звукова карта и а

модемна карта и двете случайно използваха, да речем, прекъсване 4. Те биха се сблъскали и

не биха работили заедно. Решението беше да се включат DIP превключватели или джъмпери

всяка I/O карта и инструктирайте потребителя да го настрои да избере ниво на прекъсване

и адреси на I/O устройства, които не са в конфликт с други в системата на потребителя.

Това, което прави plug and play, е системата автоматично да събира информация

относно I/O устройствата, централно задайте нива на прекъсване и I/O адреси и след това

кажете на всяка карта какви са нейните номера. Тази работа е тясно свързана със зареждането на

компютър, така че нека да разгледаме това. Не е съвсем тривиално.

6.Booting the Computer

Всеки компютър съдържа родителска дъска

(наричано преди дънна платка, преди политическата коректност да удари компютърната индустрия). На родителската платка има програма, наречена системен BIOS (Basic Input Output put System). BIOS съдържа I/O софтуер от ниско ниво, включително процедури за

четете клавиатурата, пишете на екрана и правете дисков I/O, наред с други неща. Днес той се съхранява във флаш RAM, която е енергонезависима, но която може да се актуализира от

операционната система, когато се открият грешки в BIOS.

Когато компютърът се стартира, BIOS се стартира. Първо проверява как

много RAM е инсталирано и дали клавиатурата и другите основни устройства са в застой и реагират правилно. Започва със сканиране на PCIe и PCI шините

за откриване на всички устройства, свързани към тях. Ако наличните устройства са различни от

когато системата е заредена последно, новите устройства са конфигурирани.

След това BIOS определя устройството за стартиране, като изпробва списък с устройства, съхранени в

CMOS паметта. Потребителят може да промени този списък, като въведе конфигурация на BIOS

програма веднага след стартиране. Обикновено се прави опит за зареждане от CD-ROM

(или понякога USB) устройство, ако има такова. Ако това не успее, системата се зарежда от

твърд диск. Първият сектор от устройството за зареждане се чете в паметта и се изпълнява.

Този сектор съдържа програма, която обикновено разглежда таблицата на дяловете в

края на сектора за зареждане, за да определите кой дял е активен. След това вторично зареждане

loader се чете от този дял. Този зареждане чете в операционната система

от активния дял и го стартира.

След това операционната система отправя запитване към BIOS, за да получи информация за конфигурацията. За всяко устройство той проверява дали има драйвер на устройството. Ако не, пита

потребителят да постави CD-ROM, съдържащ драйвера (доставен от производителя на устройството) или да го изтегли от интернет. След като има всички драйвери на устройства,

операционната система ги зарежда в ядрото. След това инициализира своите таблици, създава

каквито и фонови процеси да са необходими, и стартира програма за влизане или GUI.

Какво е UEFI и с какво се различава от BIOS

UEFI замени стандартния BIOS в персоналните компютри. Ако даден PC е с BIOS, замяната му с UEFI е невъзможна, понеже е необходим хардуер, поддържащ UEFI. От друга страна, повечето версии на UEFI поддържат емулацията на BIOS, за да е възможно инсталирането и работа със стари операционни системи, очакващи наличието на BIOS вместо UEFI. Може да се каже, че е реализирана обратната съвместимост.

Новият стандарт премахва всички ограничения на BIOS. UEFI работи с дискове над 2,2 TB – теоретичният предел за UEFI е 9,4 зетабайта, Това е около три пъти повече от всички данни в днешния интернет. UEFI поддържа подобни информационни обеми заради използването на GPT дялове вместо MBR. И още, процесът на зареждане е стандартизиран и могат да се стартират различни изпълними програми вместо кода, разположен в MBR.

UEFI може да работи в 32-битов и 64-битов режим. Адресното пространство е много по-голямо от това в BIOS и съответно, зареждането е по-бързо. Това означава още, че програмните екрани за настройките могат да бъдат много по-функционални от тези в BIOS и в тях е възможно да има сложна графика и поддръжка на мишка. Но не е задължително. Много компютри използват UEFI с текстов режим, който изглежда като старите екрани на BIOS.

В UEFI са вградени редица други функции. Поддържа се безопасното стартиране Secure Boot чрез което е възможно да се провери, дали зареждането на ОС не е променено от някоя вредоносна програма. Поддържа се мрежова работа, която дава възможност за отдалечена настройка и отдалечено премахване на грешки.

UEFI не е просто някаква замяна на BIOS. UEFI е нещо като малка операционна система, работеща на ниво фърмуер, която може много повече неща от BIOS. Тя може да бъде записана на флаш памет на дънната платка или да се зарежда от хард или флаш диска на компютъра. Има решения за зареждане на UEFI по мрежата.

Различните компютри имат UEFI с различни свойства и различен интерфейс. Всичко зависи от производителя на компютъра, но основните възможности са еднакви за всички.

MBR – Master Boot Record

Видове операционни системи :

1. **Основни операционни системи**
2. **Сървърни операционни системи**
3. **Мултипроцесорни операционни системи**
4. **Операционни системи за личен компютър**
5. **Ръчни операционни системи**
6. **Вградени операционни системи**
7. **Операционни системи на сензорни възли**
8. **Операционни системи в реално време**
9. **Операционни системи за смарт карти**

Процеси

Ключова концепция във всички операционни системи е процесът. Процесът е основно а

програма в изпълнение. С всеки процес е свързано неговото адресно пространство, списък с

места в паметта от 0 до някакъв максимум, които процесът може да чете и записва.

Адресното пространство съдържа изпълнимата програма, данните на програмата и нейните

стека.

Адресни пространства

Всеки компютър има някаква основна памет, която използва за поддържане на изпълнявани програми. В много проста операционна система само една програма е в паметта. За да стартирате втора програма, първата трябва да бъде премахната, а втората

поставени в паметта.

Файлове

Input/Output

Всички компютри имат физически устройства за получаване на вход и производство на изход.

В крайна сметка каква полза би бил компютърът, ако потребителите не могат да му казват какво да прави

и не можа да получи резултатите, след като е извършил исканата работа? Много видове вход

и съществуват изходни устройства, включително клавиатури, монитори, принтери и т.н. то е

до операционната система за управление на тези устройства.

Следователно всяка операционна система има I/O подсистема за управление на нейната

I/O устройства. Част от I/O софтуера е независим от устройството, тоест се отнася за

много или всички I/O устройства еднакво добре. Други части от него, като драйвери на устройства, са

специфични за определени I/O устройства. В гл. 5 ще разгледаме I/O софтуера.

Защита

Черупката(The Shell)

Онтогенезата рекапитулира филогенезата

Системни повиквания

1.Системни повиквания за управление на процесите

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Directory- and file-system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Figure 1-18. Some of the major POSIX system calls. The return code *s* is −1 if an error has occurred. The return codes are as follows: *pid* is a process id, *fd* is a file descriptor, *n* is a byte count, *position* is an offset within the file, and *seconds* is the elapsed time. The parameters are explained in the text.

Структура на операционната система :

1. **Монолитни системи**
2. **Многослойни системи**
3. **Микроядра**

4. **Модел клиент-сървър**
5. **Виртуални машини**
6. **Екзоядра**

Процеси и нишки

Резюме

За да скрият ефектите от прекъсванията, операционните системи предоставят концептуален модел

състояща се от последователни процеси, протичащи паралелно. Могат да се създават процеси и

прекратява се динамично. Всеки процес има свое собствено адресно пространство.

Процесът е просто екземпляр на изпълняваща се програма, включваща текущите стойности на програмния брояч, регистри и променливи.

За някои приложения е полезно да имате множество нишки за контрол в рамките на а

единичен процес. Тези нишки се планират независимо и всяка от тях има своите

собствен стек, но всички нишки в един процес споделят общо адресно пространство. Конци

може да се реализира в потребителско пространство или в ядрото.

Процесите могат да комуникират помежду си, като използват примитиви за междупроцесна комуникация, например семафори, монитори или съобщения.

(IPC - Междупроцесна комуникация)

Междупроцесните комуникационни примитиви могат да се използват за решаване на проблеми като

продуцент-потребител, философи за хранене и читател-писател. Дори и с тези

примитиви, трябва да се внимава, за да се избегнат грешки и застой.

Управление на паметта

Видяхме, че най-простите системи изобщо не се разменят или страници.

Следващата стъпка е размяната. Когато се използва размяна, системата може да се справи

повече процеси, отколкото има място в паметта. Процеси, за които няма

стаята се разменят на диска. Свободното място в паметта и на диска може да се запази

проследяване с растерна карта или списък с дупки.

Съвременните компютри често имат някаква форма на виртуална памет. В най-простото

форма, адресното пространство на всеки процес е разделено на блокове с еднакъв размер, наречени

страници, които могат да бъдат поставени във всяка налична рамка на страницата в паметта. Има

много алгоритми за подмяна на страници; два от по-добрите алгоритми са стареене и

WSClock.

За да накарате системите за пейджинг да работят добре, изборът на алгоритъм не е достатъчен;

внимание към такива въпроси като определяне на работния набор, политика за разпределяне на паметта,

и се изисква размер на страницата.

Сегментирането помага при работа със структури от данни, които могат да променят размера по време на изпълнение и опростява свързването и споделянето.

Файлови системи

Когато се гледа отвън, файловата система е колекция от файлове и директории, плюс операции върху тях. Файловете могат да се четат и записват, директориите могат да бъдат

създадени и унищожени, а файловете могат да се преместват от директория в директория. Повечето

съвременните файлови системи поддържат йерархична система от директории, в която директории

може да има поддиректории и те могат да имат поддиректории безкрайно.

Когато се погледне отвътре, файловата система изглежда съвсем различно. Файловата система

дизайнерите трябва да се интересуват от това как се разпределя паметта и как системата

следи кой блок отива с кой файл. Възможностите включват съседни

файлове, свързани списъци, таблици за разпределение на файлове и i-възли. Различните системи имат различни структури на директории. Атрибутите могат да отидат в директориите или някъде другаде

(например i-възел). Дисковото пространство може да се управлява с помощта на безплатни списъци или растерни изображения. Надеждността на файловата система се подобрява чрез правене на инкрементални дъмпове и чрез наличие на програма

които могат да поправят болни файлови системи. Производителността на файловата система е важна и може да бъде

подобрени по няколко начина, включително кеширане, четене напред и внимателно поставяне на

блокове от файл близо един до друг. Файловите системи с лог структура също подобряват производителността, като извършват запис в големи единици.

Примерите за файлови системи включват ISO 9660, -DOS и UNIX. Тези се различават по

много начини, включително как следят кои блокове вървят с кой файл, структура на директорията и управление на свободното дисково пространство.

Input/Output

I/O може да се осъществи в един от

три начина. Първо, има програмиран I/O, при който главният процесор въвежда или извежда всеки байт или дума и седи в стегнат цикъл, чакайки, докато може да получи или изпрати

следващ. Второ, има I/O управляван от прекъсване, при който процесорът стартира I/O

прехвърляне за знак или дума и тръгва да направи нещо друго до прекъсване

пристига сигнал за завършване на I/O. Трето, има DMA, в който отделно

чип управлява пълното прехвърляне на блок от данни само при прекъсване

когато целият блок е прехвърлен.

I/O може да бъде структуриран на четири нива: процедурите за обслужване на прекъсване, устройството

драйвери, независимия от устройството I/O софтуер и I/O библиотеки и спулери, които

изпълнявайте в потребителско пространство. Драйверите на устройства обработват подробностите за стартирането на устройствата и

осигуряване на еднакви интерфейси към останалата част от операционната система. Независимият от устройството I/O софтуер прави неща като буфериране и отчитане на грешки.

Дисковете се предлагат в различни видове, включително магнитни дискове, RAID, флаш

устройства и оптични дискове. При въртящи се дискове алгоритмите за планиране на дисково рамо могат

често се използва за подобряване на производителността на диска, но наличието на виртуални геометрии

усложнява нещата. Чрез сдвояване на два диска, стабилен носител за съхранение с определени

могат да се конструират полезни свойства.

Мъртви хватки

Мъртвата хватка е потенциален проблем във всяка операционна система. Това се случва, когато всички

членовете на набор от процеси са блокирани в очакване на събитие, което само друго

членовете на едно и също множество могат да причинят. Тази ситуация кара всички процеси да чакат завинаги. Обикновено събитието, което процесите чакат, е освобождаването на

някакъв ресурс, притежаван от друг член на набора. Друга ситуация, в която

Възможна е безизходица, когато набор от комуникационни процеси чакат а

съобщение и комуникационният канал е празен и няма изчакващи изчаквания.

Мъртва хватка на ресурсите може да бъде избегнат, като се следи кои състояния са безопасни

и които са опасни. Безопасно състояние е това, в което съществува последователност от

събития, които гарантират, че всички процеси могат да завършат. Несигурна държава няма такова

гаранция. Алгоритъмът на банкера избягва задънена улица, като не дава заявка, ако

тази заявка ще постави системата в опасно състояние.

Застой на ресурсите може да бъде структурно предотвратен чрез изграждане на системата

по такъв начин, че никога да не може да се случи по замисъл. Например чрез разрешаване на процес

да държи само един ресурс във всеки момент, за което се изисква условието за кръгово изчакване

безизходица е прекъсната. Застой на ресурсите може също да бъде предотвратен чрез номериране на всички

ресурси и изработващи процеси ги изискват в строго нарастващ ред.

Застой на ресурсите не е единственият вид задънена улица. Застой в комуникацията

също е потенциален проблем в някои системи, въпреки че често може да бъде разрешен от

задаване на подходящи изчаквания.

Livelock е подобен на застой, тъй като може да спре целия напредък, но е така

технически различен, тъй като включва процеси, които всъщност не са блокирани. Звездолюбието може да бъде избегнато чрез политика за разпределение „първи дошъл, първи обслужен“.

Множествени процесорни системи

Компютърните системи могат да бъдат направени по-бързи и по-надеждни чрез използване на множество

процесори. Четири организации за многопроцесорни системи са мултипроцесори, мултикомпютри, виртуални машини и разпределени системи. Всеки от тях има свои собствени връзки и проблеми.

Мултипроцесорът се състои от два или повече процесора, които споделят обща RAM памет.

Често самите тези процесори се състоят от множество ядра. Ядрата и процесорите могат

да бъдат свързани помежду си чрез шина, превключвател или многостепенна комутационна мрежа.

Възможни са различни конфигурации на операционната система, включително предоставяне на всеки процесор

собствена операционна система, като има една главна операционна система, а останалите са

подчинени устройства или със симетричен мултипроцесор, в който има едно копие на операционната система, която всеки процесор може да изпълнява. В последния случай са необходими ключалки за осигуряване

синхронизация. Когато заключването не е налично, процесорът може да върти или да прави контекст

превключвател. Възможни са различни алгоритми за планиране, включително споделяне на време, пространство

споделяне и планиране на банди.

Мултикомпютрите също имат два или повече процесора, но всеки от тези процесори има своите

собствена лична памет. Те не споделят обща RAM памет, така че цялата комуникация

използва предаване на съобщения. В някои случаи мрежовата интерфейсна платка има своя собствена

CPU, в който случай комуникацията между главния CPU и CPU на интерфейса трябва да бъде внимателно организирана, за да се избегнат условия на състезание.
Потребителско ниво

комуникацията на мултикомпютри често използва отдалечени извиквания на процедури, но може да се използва и разпределена споделена памет. Балансирането на натоварването на процесите е проблем тук,

и различните алгоритми, използвани за него, включват инициирани от изпращача алгоритми, инициирани от получател алгоритми и алгоритми за наддаване.

Разпределените системи са слабо свързани системи, всеки от които възел е а

пълен компютър с пълен комплект периферни устройства и собствена операционна система. Често тези системи са разпространени в голяма географска област. Мидълуерът често се поставя върху операционната система, за да осигури единен слой за приложенията

да взаимодействат с. Различните видове включват базиран на документи, базиран на файлове, базиран на обекти и базиран на координация междинен софтуер. Някои примери са Светът

Wide Web, CORBA и Linda.

Случай за учене 1 : UNIX, Linux and Android

Linux започна живота си като клонинг на UNIX с отворен код, пълно производство и сега е така

използвани на машини, вариращи от смартфони и преносими компютри до

суперкомпютри. Съществуват три основни интерфейса към него: обвивката, библиотеката C и

системните обаждания. В допълнение, графичен потребителски интерфейс често се използва за опростяване на взаимодействието на потребителя със системата.

Ключовите концепции в Linux включват процеса, модела на паметта, I/O и

файловата система. Процесите могат да отделят подпроцеси, което води до дърво от процеси.

Управлението на процесите в Linux е различно в сравнение с други UNIX системи в това

Linux разглежда всеки изпълнителен обект – еднонишков процес или всяка нишка в многонишков процес или ядрото – като различима задача. Процес или а

една задача като цяло, след това се представя чрез два ключови компонента, структурата на задачата и допълнителната информация, описваща адресното пространство на потребителя. Бившият

винаги е в паметта, но последните данни могат да бъдат качени в и извън паметта. Създаването на процес се извършва чрез дублиране на структурата на задачата на процеса и след това задаване на

информация за изображението на паметта, за да сочи към изображението на паметта на родителя. Реални копия

от страниците с изображения в паметта се създават само ако споделянето не е разрешено и е необходима модификация на паметта. Този механизъм се нарича копиране при запис. Планирането е

направено с помощта на претеглен алгоритъм за справедлива опашка, който използва червено-черно дърво за

управление на опашката от задачи.

Моделът на паметта се състои от три сегмента на процес: текст, данни и

стека. Управлението на паметта се извършва чрез пейджинг. Карта в паметта следи

състоянието на всяка страница, а демонът на страницата използва модифициран алгоритъм на часовника с двойна стрелка, за да поддържа достатъчно безплатни страници.

Достъпът до I/O устройства се осъществява чрез специални файлове, всеки от които има основен номер на устройство и второстепенен номер на устройство. I/O на блоково устройство използва основната памет за кеширане

дискови блокове и намаляване на броя на достъпите до диска. Вход/изход на символи може да се извърши

необработен режим или потоците от знаци могат да бъдат модифицирани чрез дисциплини на редове. Работа в мрежа

устройствата се третират малко по-различно, чрез свързване на целия мрежов протокол

модули за обработка на потока от мрежови пакети към и от потребителския процес.

Файловата система е йерархична с файлове и директории. Всички дискове са монтирани

в едно дърво на директории, започващо от уникален корен. Отделни файлове могат да бъдат свързани

в директория от другаде във файловата система. За да използвате файл, той трябва да е първи

отворен, което дава файлов дескриптор за използване при четене и запис на файла. Вътрешно файловата система използва три основни таблици: таблицата с файлови дескриптори, таблицата

таблица с описание на отворения файл и таблицата с i-възел. Таблицата i-node е най-важната от тях, съдържаща цялата административна информация за файл и

местоположението на неговите блокове. Директориите и устройствата също са представени като файлове, заедно

с други специални файлове.

Защитата се основава на контролиране на достъпа за четене, запис и изпълнение за

собственик, група и други. За директории битът за изпълнение означава разрешение за търсене.

Android е платформа, която позволява на приложения да работят на мобилни устройства. Тя се основава

на ядрото на Linux, но се състои от голяма част от софтуер върху Linux, плюс

малък брой промени в ядрото на Linux. Повечето от Android са написани на Java.

Приложенията също са написани на Java, след това се превеждат в Java байткод и след това в Dalvik

байт код. Приложенията за Android комуникират чрез форма на защитено съобщение, предавайки транзакции за повикване. Специален модел на ядрото на Linux, наречен Binder, обработва IPC.

Пакетите за Android са самостоятелни и имат манифест, описващ какво се съдържа

Пакетът. Пакетите съдържат дейности, получатели, доставчици на съдържание и намерения.

Моделът за сигурност на Android е различен от модела на Linux и внимателно съхранява всяко приложение, тъй като всички приложения се считат за ненадеждни.

Дизайн на операционната система

Проектирането на операционна система започва с определяне какво трябва да прави. В

интерфейсът трябва да бъде прост, пълен и ефективен. Той трябва да има ясна парадигма на потребителския интерфейс, парадигма на изпълнение и парадигма за данни.

Системата трябва да бъде добре структурирана, като се използва една от няколкото известни техники,

като наслояване или клиент-сървър. Вътрешните компоненти трябва да са ортогонални на

един друг и ясно отделят политиката от механизма. Значителна мисъл

трябва да бъдат дадени на въпроси като статична спрямо динамична структура на данни, именуване, време за свързване и ред на внедряване на модули.

Производителността е важна, но оптимизацията трябва да се избира внимателно, така че

да не се разваля структурата на системата. Пространствено-времеви компромиси, кеширане, намеци, експлоатиране

местността и оптимизирането на общия случай често си струва да се направи.

Написването на система с няколко души е различно от създаването на голяма система с 300 души. В последния случай, екипна структура и управление на проекти

играят решаваща роля за успеха или провала на проекта.

И накрая, операционните системи се променят, за да се адаптират към новите тенденции и да се срещнат с нови

предизвикателства. Те включват системи, базирани на хипервизор, многоядрени системи, 64-битови адресни пространства, ръчни безжични компютри и вградени системи. Няма

съмнявам се, че следващите години ще бъдат вълнуващи времена за дизайнерите на операционни системи.

Logical gates, Latches, Flip-flops(Логически порти, ключалки, джапанки)