6. Tietokoneet ja tietojenkäsittely

Johdanto

Tietokoneet ovat keskeinen osa informaatioteknologiaa ja jokaisen alalla työskentelevän täytyy osata vähintään ohjelmoinnin perusteet. Koska ohjelmointi on välttämätön taito kaikilla tekniikan aloilla, niin tässä oletuksena on, että jokainen opiskelija perehtyy ohjelmointiin syvemmin muilla kursseilla. Tietokoneiden rakenteen osalta tilanne on hieman toinen. Sen osalta tämän materiaalin pyrkimys on luoda yleiskuva ja opettaa muutamia perusasioita, joiden tuntemus on hyödyllistä muillakin aloilla. Tavoitteena on luoda perusymmärrys siitä, miten tietokoneohjelma muunnetaan fyysisen tason sähköisiksi ilmiöiksi.

Tämän osion keskeisimmät aiheet ovat:

1. Mooren laki ja sen vaikutukset
2. Tietokoneen rakenneosat ja toimintaperiaate
3. Informaation esitystavat mukaan lukien negatiiviset luvut
4. Boolen algebra ja loogiset piirit
5. Ohjelmointikielten tasot ja käyttöalueet

Laskennallisena mallina käsitellään Hamming-koodauksen periaatetta.

Tietotekniikan sanoista ja historiasta

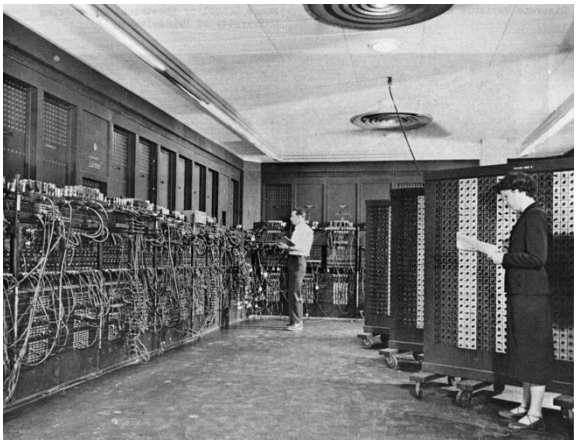
Kun sanan *tietojenkäsittely* laittaa hakukoneeseen,[228] yksitoista ensimmäistä tulosta viittaa tavalla tai toisella ammattikorkeakouluihin, kahdestoista on Wiktionaryn sivu koskien termiä tietojenkäsittely.[229] Sen sijaan sana *tietotekniikka* antaa erilaisen listan: Wikipedian tietotekniikkaa käsittelevä artikkeli on ensimmäisenä ja Aalto-yliopiston tietotekniikan koulutusohjelma on kolmantenatoista. Tietojenkäsittely on siis jotain käytännöllistä, kun taas tietotekniikka on akateemisempi ja laaja-alaisempi termi.

Suurta epäselvyyttä oli aikanaan nykyisin tietokoneeksi (computer) kutsuttujen laitteiden nimeämisessä. Vielä 1950-luvulla nimeämiskäytäntö vaihteli suuresti. Suomen Akatemian kielilautakunnan kokouksessa joulukuussa 1959 esitettiin, että käyttöön otetaan sana tietokone, eikä esimerkiksi tiedin, jota myös oli ehdotettu.[230] Ensimmäinen Teknillisen korkeakoulun diplomityö, jossa käytettiin sanaa tietokone, on vuodelta 1961. Samaan aikaan

[228] Tässä tapauksessa hakukone on www.bing.com, mutta tulos on olennaisesti sama muillakin hakukoneilla.
[229] http://en.wiktionary.org/wiki/tietojenk%C3%A4sittely. J. Korpelan sanasto: http://jkorpela.fi/dt.html8
[230] J. Suominen, *Sähköaivo sinuiksi, tietokone tutuksi. Tietotekniikan kulttuurihistoriaa*, Jyväskylä 2000.

Reikäkortti-lehti muutti nimensä Tietokone-lehdeksi ja Reikäkorttiyhdistys Tietokoneyhdistykseksi. Tietokone vakiintui terminä 1960-luvun aikana. Suuren Tietokirjan kolmannen painoksen viides osa vuodelta 1966 sisältää kolmen sivun artikkelin hakusanalla ”Matematiikkakone - laskeva robotti.” Artikkelissa ei esiinny sanaa tietokone vaan termit sähköaivot ja numerokone synonyymeinä matematiikkakoneelle. Ohjelmoinnista ei ole sanaakaan. Monet ohjelmointikielet, kuten Fortran ja Cobol, oli kuitenkin kehitetty jo lähes kymmenen vuotta aiemmin. Eikä silloin ollut Wikipediaa, josta olisi voinut tarkistaa mitä ohjelmointi tarkoittaa. Tietotekniikka otettiin käyttöön terminä vasta 1980-luvun aikana.

Siirrytään vielä vähän ajassa taaksepäin ja luodaan lyhyt katsaus tietotekniikan kehitykseen perustuen pääosin Walter Isaacsonin näkemykseen.[231] Ensimmäisen tietokoneen arvonimi on kiistanalainen kysymys, johon voidaan vastata täsmällisesti vain määrittelemällä tietokone riittävän tarkasti. Voidaan vaatia, että tietokoneen on oltava digitaalinen, toiminnaltaan täysin sähköinen, ohjelmoitava, yleiskäyttöinen ja toimiva. Näillä kriteereillä ensimmäisen tietokoneen arvon saa ENIAC (Electronic Numerical Integrator And Computer), kuvassa 6.1. ENIAC:n kehityksen rahoitti USA:n armeija alkuperäisenä tavoitteena laskea ballistisia liikeratoja. Laskutoimitukset olivat erittäin hankalia suorittaa mekaanisilla laskimilla ja ihmistyövoimalla.



Kuva 6.1. ENIAC.[232]

Parhaimmillaan ballistisia laskutoimituksia teki yli 170 henkeä, pääosin naisia, mutta tarvittavia taulukoita ei pystytty silti laskemaan riittävän nopeasti. Tällä ongelmalla oli jopa sotilaallista merkitystä. ENIAC valmistui kuitenkin vasta toisen maailmansodan loputtua, joten se ei ehtinyt vaikuttaa sodan kulkuun. Valmistuessaan marraskuussa 1945 ENIAC oli yli sata kertaan nopeampi kuin mikään aikaisempi laskukone. Lisäksi se oli suunniteltu siten, että sillä voitiin tehdä periaatteessa minkälaisia laskutoimituksia tahansa. Ensimmäiseksi sillä tehtiin vetypommin toimintaan liittyviä laskelmia.

Mekaanisia laskukoneita oli jo kehitetty erityistarkoituksia varten, esimerkkinä Yhdysvalloissa vuoden 1890 väestölaskennan tarpeisiin kehitetty reikäkorttikone. Käytännössä pitää siirtyä 1930-luvun lopulle ennen kuin mitään merkittävää tapahtui laskentakoneiden

[231] Walter Isaacson: The Innovators, How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution, Simon & Schuster, 2014. Tietotekniikan kehitysvaiheista on kuvaus myös sivuilla www.computerhistory.org/timeline/
[232] Kuva: http://en.wikipedia.org/wiki/ENIAC#mediaviewer/File:Eniac.jpg.

alalla. Alan pioneereihin kuului saksalainen Konrad Zuse, joka rakensi mekaanisen laskentakoneen vuonna 1938. Laitteen toiminta ei kuitenkaan ollut kovin luotettavaa, eivätkä sähkömekaaniset versiot laitteesta koskaan valmistuneet.[233] AT&T:n tutkimuslaboratoriossa (Bell Labs[234]) saatiin vuonna 1939 valmiiksi sähkömekaaninen laskin. Myös John Atanasoff (USA, Iowan osavaltio) rakensi tietokoneen prototyypin vuosina 1939-42. Vaikka hän käyttikin laskentaan tyhjiöputkia, muisti perustui mekaanisesti pyörivään rumpuun, joka olennaisesti hidasti laitteen toimintaa.[235] Atanasoff ei saanut laitettaan koskaan toimintakuntoon eikä se ollut vapaasti ohjelmoitavissa. IBM rakensi Howard Aikenin johdolla 1940-luvun alussa laskentakoneen, nimeltään Mark I. Siinä käytettiin mekaanisia kytkimiä, joita ohjattiin sähkömoottoreilla, joten edellä esitettyjä tietokoneen kriteerejä sekään ei täyttänyt.[236] Iso-Britanniassa rakennettiin toisen maailmansodan aikana digitaalinen laskentakone, nimeltään Colossus, joka pystyi vain yhden ennalta määritellyn ongelman ratkaisuun, eli Saksan salakirjoitusten murtamiseen.[237]

Miksi kaikki tämä tapahtui juuri 1930-luvun lopulla ja 1940-luvulla? Mekaanisia ja jopa sähkömekaanisia koneita olisi voitu rakentaa jo aikaisemmin. Ehkä riittävän suurta tarvetta ei aikaisemmin ollut, jotta toimivan laskentakoneen kehittämiseen vaadittu monipuolinen osaaminen olisi saatu koottua yhteen paikkaan. Vasta armeijan laskentatarpeet toivat riittävän kimmokkeen tietokoneiden järjestelmälliseen kehittämiseen. Tämä yhdistettynä muutamaan nerokkaaseen ajattelijaan, joista erityisesti on mainittava Alan Turing ja John von Neumann, johti muutamassa vuodessa kehitysharppaukseen. Lisäksi tarvittiin lukuisia insinöörejä ja teknikkoja toteuttamaan hienot ajatukset käytännössä. ENIAC painoi 27 tonnia ja siinä oli 17 468 tyhjiöputkea ja 5 miljoonaa juotosta, joten yksinäinen nero ei olisi sitä koskaan pystynyt kokoamaan.

Tarvittiin siis pakottava tarve, syvällinen ymmärrys tehtävän luonteesta, riittävä tekniikan taso ja paikka johon kaikki kertynyt taito ja tietämys saatiin koottua yhteen. Nämä kaikki vaatimukset täyttyivät vasta USA:ssa toisen maailmansodan aikana. Iso-Britannia oli myös lähellä läpimurtoa, mutta tehtävän luonteesta johtuen brittien Colossus-projekti oli niin salainen, ettei sen kehitys jatkunut yleiskäyttöisen tietokoneen suuntaan.

Miksi Bellin puhelimelle myönnettiin patentti, mutta kenellekään ei myönnetty patenttia koskien tietokoneen toimintaperiaatetta ja rakennetta? Monimutkaista todellisuutta hieman oikoen voidaan sanoa, että John von Neumann halusi tietokoneen toimintaperiaatteen

[233] Zuse rakensi uusia versioita laskentakoneestaan toisen maailmansodan aikana, mutta Saksan armeija ei missään vaiheessa kiinnostunut projektista. http://waste.informatik.hu-berlin.de/Diplom/ww2/zuse_e.html
[234] Nokia Bell labs on nykyisin osa Nokia-yhtiötä.
[235] Tyhjiöputket olivat vielä tällöin kalliita.
[236] Aiken oli hyvin perillä Charles Babbagen laskentakoneesta.
[237] Salakirjoitusten onnistuneella purkamisella oli ilmeisesti merkittävä vaikutus sodan kulkuun.

vapaaseen käyttöön ja sai tämän aikaan julkaisemalla laajan artikkelin ennen kuin kukaan ehti jättää asiaa koskevia patenttihakemuksia.[238] Tietokoneen patenteista seurasikin melkoisia kiistoja, kun useat henkilöt, erityisesti John Atanasoff sekä ENIAC:n keskeisimmät kehittäjät Presper Eckert ja John Mauchly, katsoivat, että heidän olisi pitänyt voida patentoida tekemänsä keksinnöt. ENIAC:iin liittyvistä patenteista käytiin kiistaa vuoteen 1973 asti, jolloin silloiset kiistan osapuolet, Sperry ja Honeywell, sopivat asian keskenään.[239]

Mikään olennainen osa tietotekniikkaa ei siis alun perin syntynyt yliopistoissa. Ennen 1930-luvun loppua ei juuri ollut tutkimuslaitoksia, joissa olisi yhdistynyt teoreettinen osaaminen, käytännöllinen insinööritaito ja tutkimuksen tavoitteellinen johtaminen. Varhaisimpia esimerkkejä perusteellisesti valmistellusta ja rahoitetusta tuotekehitysprojektista on transistorin kehittäminen. Projekti aloitettiin jo vuonna 1936 ja sen tarkoituksena oli kehittää luotettavampi, halvempi ja kooltaan pienempi vaihtoehto tyhjiöputkelle, johon siis ENIAC ja muutama muu ensimmäisistä tietokoneista perustui. Transistorin tapauksessa toinen maailmansota viivytti kehitystyötä, sillä avainhenkilöt olivat sodan aikana muissa tehtävissä. Transistorin kehitti lopulta ryhmä, jonka muodostivat John Bardeen, Walter Brattain ja William Shockley. Heissä yhdistyi teoreettinen ymmärrys ja tekninen taito sekä Bardeenin ja Brattainin tapauksessa myös kyky kommunikoida tehokkaasti muiden tutkijoiden kanssa.[240] Lisäksi ehdottomana edellytyksenä projektin onnistumiselle oli korkeatasoinen ja monipuolinen tutkimusympäristö, jonka AT&T:n tutkimusorganisaatio Bell Labs tarjosi.[241]

Bell Labs on hyvä esimerkki ainakin kolmeen ilmiöön. 1) Yksityisellä monopolilla (vaikka säädellyllä) on kyky luoda sekä aineellista että henkistä pääomaa, jota voidaan käyttää myös tarkoituksiin, jotka eivät palvele lyhyen ajan liiketoiminnallisia tarpeita.[242] 2) Monopolit eivät yleensä ole hyviä tuotteistamaan luomiaan keksintöjä. Bell Labsissa on keksitty tai kehitetty mm. transistori, laser ja avaruuden taustasäteily.[243] Tutkimustulosten kaupallinen hyödyntäminen on jäänyt usein muille yhtiöille. 3) Se syntyykö johonkin monopolin

[238] Von Neumann, John, First Draft of a Report on the EDVAC, Univ. Of Pennsylvania, June 30, 1945. Projektin hallinnoimisesta vastannut Herman Goldstine levitti julkaisun paperikopiota useille eri tahoille. Paperi johti useisiin tietokoneiden kehitysprojekteihin, joita ei mitä ilmeisimmin muuten olisi yhtä nopeasti syntynyt.

[239] Patentit ovat kauppatavara, joka voi vaihtaa moneen kertaan omistajaa patentin voimassaoloaikana. Alkuperäiset keksijät saavat yleensä kertakorvauksen keksinnöstään ja vain joskus harvoin osuuden patentin laskennallisesta tuotosta.

[240] Shockley sen sijaan oli hankala persoona. Tutkimusryhmä hajosi välittömästi transistorin keksimisen jälkeen.

[241] Huomatkaa kuvan 2.6 puheluiden reaalihintojen kehitys USA:ssa vuosien 1920 ja 1940 välillä. Puhelinpalvelut olivat tuolloin mitä ilmeisimmin erittäin kannattavaa liiketoimintaa.

[242] Säädellyn monopolin liiketaloudellinen tulos riippuu ensisijaisesti säätelyn tiukkuudesta.

[243] Bell Labsissa tehdyn tutkimuksen perusteella on myönnetty kahdeksan kertaa Nobelin palkinto, joita on ollut jakamassa yhteensä neljätoista tutkijaa.

kaltaiseen yritykseen luovaa toimintaa, riippuu siitä, sattuuko sopivaan asemaan nousemaan henkilö, jolla on riittävä näkemys, kyky ja innostus luoda ja ylläpitää riittävän suuri ja monipuolinen ryhmä. Joskus nousee, useimmiten ei.

Mooren laki

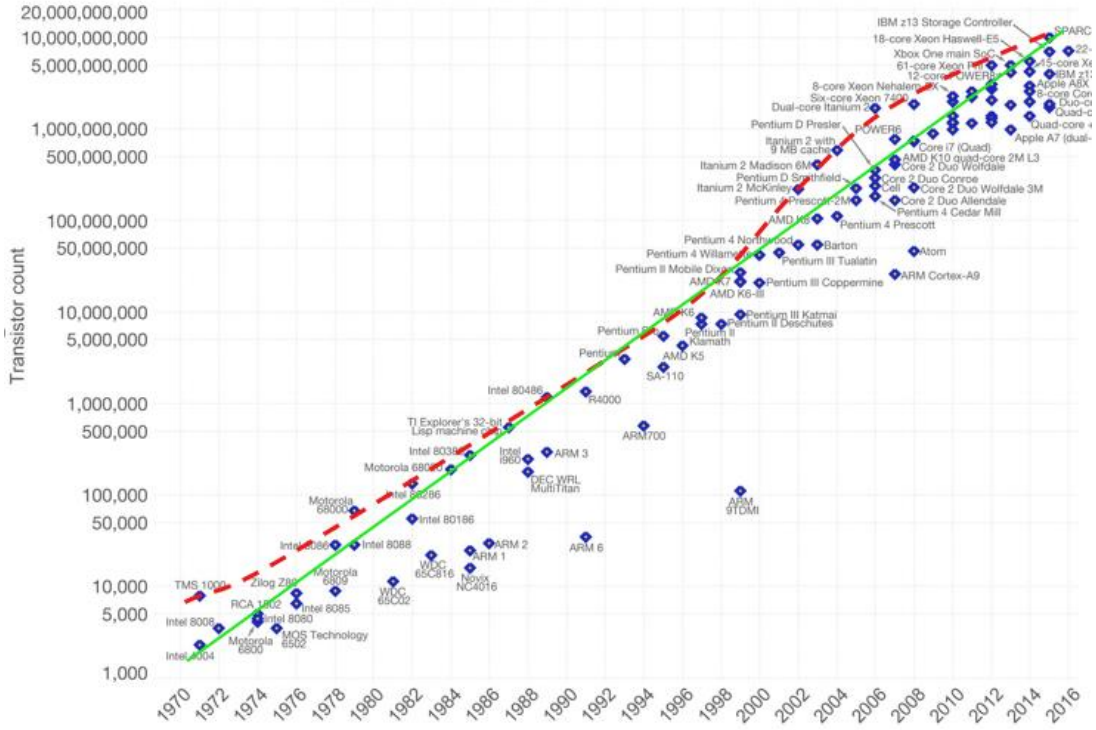
Transistori ja puolijohdeteknologia ylipäätään osoittautui erittäin keskeiseksi tietotekniikan kehittymisen kannalta. Tämän havaitsi yksi puolijohteiden kehittämisen avainhenkilöistä, Gordon Moore. Hän julkaisi keväällä 1965 artikkelin, jossa hän esitti ennusteen transistorien määrän kasvusta mikropiirillä.[244] Myöhemmin ennusteesta alettiin käyttää nimitystä Mooren laki (Moore's law). Mooren laissa (siinä muodossa kuin se on alkuperäisessä paperissa esitetty) kyseessä ei ole suurin mahdollinen integraatioaste vaan se määrä transistoreja, jolla kustannus transistoria kohden on alhaisin. Aina on siis kysymys myös siitä, mikä on taloudellisesti järkevää, ei pelkästään siitä, mikä on teknisesti mahdollista.

Helpointa ennustaminen on silloin kun kyseessä on jokin mitattavissa oleva suure, kuten siirtokapasiteetti tai tietokoneiden laskentanopeus. Alkuperäisessä artikkelissaan Moore esitti, että yhdelle piirille integroitavien transistorien määrä kasvaa 10-kertaiseksi vuoteen 1970 mennessä, eli siis viiden vuoden aikana. Tämä vastaa transistorien määrän kaksinkertaistumista 18 kuukauden välein. Myöhemmin kaksinkertaistumisen ajaksi on vakiintunut 2 vuotta, kuten kuvasta 6.2 voidaan havaita (kaksinkertaistuminen kahdessa vuodessa tarkoittaa tuhatkertaistumista kahdessakymmenessä vuodessa). Mooren laki on toiminut erittäin hyvin viisikymmentä vuotta ja joidenkin arvioiden mukaan se toimii ainakin vuosikymmenen loppuun, jolloin pienimpien transistorien koko on viiden nanometrin luokkaa.[245] Kvanttimekaaniset ilmiöt saattavat estää tätä pienempien transistorien toiminnan.

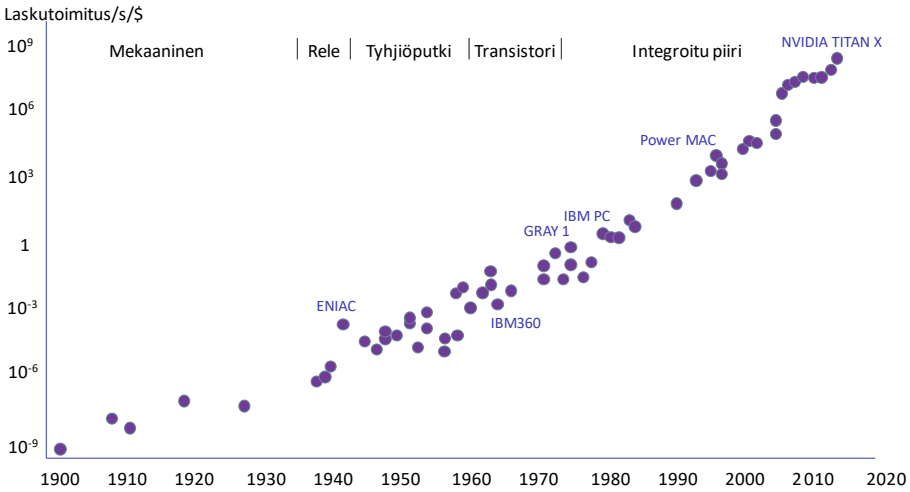
Tarkastelu voidaan viedä vielä paria astetta pidemmälle. Transistorien määrä sinänsä ei ole niin olennaista kuin se, kuinka monta laskutoimitusta voidaan suorittaa aikayksikössä. Lisäksi yleensä rajoittavana tekijänä ei ole se, kuinka paljon jotain saadaan tungettua yhteen prosessoriin vaan hinta.[246] Kuvassa 6.3 on esitetty miten laskentakapasiteetin määrä 1000 US dollaria kohti on kehittynyt 1900-luvun alusta. Vuotuinen kasvuprosentti näyttäisi siis olevan kasvussa. 1900-luvun alkupuolella samalla hinnalla sai keskimäärin 20 prosenttia enemmän laskutoimituksia sekunnissa kuin edellisenä vuonna. Vuoden 1970 jälkeen vastaava luku on ollut noin 50 prosenttia, viimeisen kymmenen vuoden aikana jopa 100 prosenttia. Kun nyt kapasiteetin 100 miljardia laskutoimitusta sekunnissa saa muutamalla dollarilla, niin vastaava kapasiteetti olisi maksanut 60-luvun alussa ehkä 1000 miljardia

[244] Gordon E. Moore, "Cramming More Components onto Integrated Circuits," Electronics, April 19, 1965.
[245] Tim Bradshaw, Intel chief raises doubts over Moore's Law, Financial Times, July 16, 2015. Artikkelissa esitetyn arvion mukaan kehitys olisi nyt hidastumassa siitä mitä Mooren laki ennustaa, mutta ei kuitenkaan pysähtymässä.
[246] Älypuhelimissa myös tilavuudella on toki olennainen merkitys.

dollaria. Kaikesta tästä on seurauksena ollut se, että älypuhelimessa oli jo vuonna 2012 enemmän tietojenkäsittelykapasiteettia kuin tehokkaimmassa supertietokoneessa (Cray X-MP) vuonna 1982.[247]



Kuva 6.2. Transistorien määrän kehitys mikroprosessoreissa, vihreä viiva = Mooren lain mukainen kasvu, punainen katkoviiva kuvaa tiheimmin pakattujen piirien kehittymistä.[248]



Kuva 6.3. Laskentakapasiteetin hinnan kehittyminen.[249]

[247] Lähteet: http://www.walkingrandomly.com/?p=3079 ja http://www.computerhistory.org/timeline/computers/
[248] http://en.wikipedia.org/wiki/Transistor_Count
[249] https://en.wikipedia.org/wiki/Moore's_law#/media/File:Moore%27s_Law_over_120_Years.png

Tietokoneen toiminta

Informaation esitystapa

Tietokone käsittelee pelkkää dataa ilman informaatiota tai tulkintaa sen sisällöstä. Sen sijaan mikään nykyinen tietokone eivät varsinaisesti ”ymmärrä” datan sisältöä, eli sitä mitä se on tekemässä tai mitä tulos tarkoittaa.[250] Käytännössä data esitetään aina bitteinä, eli siis sarjana nollia ja ykkösiä, eli binäärilukuina (binary number).[251] Tietty bittikuvio voi yhtä hyvin merkitä numeroa, kirjainta kuin äänen voimakkuutta tai pikselin väriä. Erityisen tärkeää nykyisissä tietokoneissa on, että data voi merkitä myös tietokoneohjelman käskyä.

Binäärinumerot esitetään yleensä kahdeksan bitin ryhmissä eli tavuissa. Yhdellä tavulla voidaan siten esittää kokonaisluvut 0:sta 255:een tai -128:sta +127:ään. Suurempien numeroiden esittämiseen käytetään useita tavuja, yleensä kaksi, neljä tai kahdeksan. Toisaalta ihminen on tottunut käsittelemään lukuja kymmenjärjestelmässä. Tästä johtuen binäärilukuja esitetään usein ns. BCD (Binary Coded Decimal) muodossa, jossa 10-järjestelmässä esitetyn luvun jokainen kymmenjärjestelmän numero[252] esitetään digitaalisessa muodossa neljällä bitillä. Lisäksi käytetään ns. heksadesimaalista esitystapaa jossa kokonaisluvuilla 0-15 on oma merkki, kuten taulukossa 6.1 on esitetty, jolloin kyseessä on lukujärjestelmä, jossa kantalukuna on 16. Etuna kymmenjärjestelmään verrattuna on helppo muunnettavuus binäärijärjestelmään ja binäärijärjestelmään verrattuna tiiviimpi esitystapa.

Muunnokset eri kantalukujärjestelmien välillä, joskin käsin vähän työläitä, ovat sinänsä suoraviivaisia. Lisäksi on pystyttävä esittämään negatiivisia lukuja, kuten -7. Koska binäärisessä luvussa esiintyy siis vain nollia ja ykkösiä mutta ei etumerkkiä (-), negatiivisuuskin täytyy esittää bittien avulla. Yksi bitti voitaisiin varata etumerkin esittämiseen, mutta tämä ei ole laskutoimitusten toteuttamisen kannalta paras ratkaisu. Yleensä laskutoimitusten sisäisenä esitysmuotona käytetään ns. kahden komplementtia, jossa vastaluku saadaan kääntämällä luvun kaikki bitit (`0` → `1`, `1` → `0`) ja lisäämällä tulokseen yksi. Esimerkiksi kolmen bitin järjestelmässä **2** = `010` joten **-2** = `101 + 1` = `110`. Kahdeksan bitin järjestelmässä vastaavasti **-2** = `11111101 + 1` = `11111110`. Kahdeksalla bitillä voidaan siten esittää kokonaisluvut välillä **-128** (`10000000`) ja **+127** (`01111111`). Tällä järjestelmällä vähennyslasku toimii täsmälleen samoin kuin yhteenlasku (ylimääräinen ykkönen tuloksen edessä jätetään pois). Kokeile!

---

[250] Tilanne saattaa vielä joskus muuttua: https://twitter.com/newscientist/status/629619029695680512.

[251] Nykyisen binäärijärjestelmän alkuna pidetään Gottfried Leibnizin vuonna 1703 julkaisemaa artikkelia: Explanation of the Binary Arithmetic, which uses only the characters 1 and 0, with some remarks on its usefulness, and on the light it throws on the ancient Chinese figures of Fu Xi. https://en.wikipedia.org/wiki/Binary_number

[252] Suomen kielellä voidaan ajatella, että luku (esimerkiksi 437) koostuu numeroista (4, 3 ja 7). Käytännössä numerosanaa käytetään laajemmassa merkityksessä. Englannin kieli on selkeämpi: 437 on number ja 4, 3 ja 7 ovat kukin digit.

Taulukko 6.1. Lukujen esitysmuotoja (negatiiviset luvut 8 bitin järjestelmässä)

Desimaali-luku	Binääri-luku	Heksadesi-maali	BCD-koodi
0	0000	0	0000
1	0001	1	0001
2	0010	2	0010
4	0100	4	0100
8	1000	8	1000
10	1010	A	0001 0000
15	1111	F	0001 0101
16	10000	10	0001 0110
17	10001	11	0001 0111
45	101101	2D	0100 0101
-1	11111111		
-2	11111110		
-45	11010011		
-128	10000000		

Käytännön elämässä olemme useimmiten kiinnostuneita jostain muusta kuin numeroista. Vaikka tämäkin teksti on tietokoneesi muistissa bitteinä, tuskin kukaan pystyy lukemaan tekstiä ruudulta pelkkinä bitteinä. Näytölläsi näkyy siis kirjaimia. Koska muistiin tallennus ja tiedon siirto tapahtuu bitteinä, on välttämätöntä, että bittien ja kirjainten välillä on yksiselitteinen suhde. Ratkaisuksi tähän tarpeeseen kehitettiin 1960-luvulla ASCII (American Standard Code for Information Interchange) –merkistö. Kyseessä on 7-bittinen merkistö, joka sisältää amerikanenglannissa tarvittavat kirjaimet (A-Z, a-z), numerot (0-9), yleisesti käytetyt välimerkit ja joitakin erikoismerkkejä kuten # ja $. Lisäksi merkistö sisältää 33 ohjauskoodia kuten rivinsiirto ja tekstin loppu.[253] Monissa kielissä käytetään kuitenkin muita kirjainmerkkejä, joita ASCII-merkistö ei sisällä, kuten ä ja ö. Tästä on seurannut erilaisia paikallisesti käytettyjä ASCII-muunnelmia. Suomalainen versio korvasi merkit [\]^ merkeillä ÄÖÅÜ ja merkit {|}~ merkeillä äöåü. Tästä seurasi myös ongelmia, koska monissa yhteyksissä, esimerkiksi ohjelmoinnissa, tarvittiin myös hakasulkuja. Yhteensopivuusongelmia ilmenee edelleen.

Miksi tyydyttiin 7 bittiin, kun 8 bittiä olisi ollut monessa suhteessa järkevämpi valinta. Olisi ollut mahdollista myös määritellä vaihtokoodi (shift), jolla olisi voinut laajentaa merkistöä tehokkaasti.[254] Suoraviivaista seitsemän bitin (eli 128 merkin) ratkaisua pidettiin kuitenkin tiedonsiirron kannalta tehokkaampana ja vaihtokoodin luotettavuutta ongelmallisena. Yleensä 7-bittisen ASCII-koodin esittämiseen käytetään yhtä kahdeksan

[253] Lisäksi on muistettava, että kun asiasta kirjoitetaan, niin bitit 0 ja 1 esitetään ASCII-merkkeinä, jotka vievät muistista tilaa vähintään 7 bittiä. Samoin on miinusmerkin (-) laita.

[254] Siis samaan tapaan kuin näppäimistössä, jossa voidaan vaihtaa pienistä isoihin kirjaimiin ja takaisin yhdellä näppäimellä, jolloin ei tarvita erillisiä näppäimiä pienille ja isoille kirjaimille.

bitin tavua, jolloin ylimääräistä bittiä voidaan käyttää pariteettibittinä. Laajemmissa merkistöissä ensimmäiset 128 merkkiä ovat edelleen ASCII-koodin mukaisia. Merkistöstandardeista tärkein on Unicode, joka määrittelee koodiarvon yli sadalle tuhannelle kirjoitusmerkille.[255] Sen yleisin esitystapa on UTF-8.

Tietokoneen perusrakenne

Nyt pidämme tietokonetta (älypuhelin yhtenä esimerkkinä) itsestäänselvyytenä myös henkilökohtaisessa käytössä, mukana kulkevana ja jatkuvasti toimintavalmiina. Toisin asiat olivat 1940-luvulla. Ensimmäiset tietokoneet olivat jättimäisiä rakennelmia, joiden käyttämiseen tarvittiin joukko eri tehtäviin erikoistuneita henkilöitä (kuten kuvasta 6.1 nähdään). Siihen nähden on yllättävää, että Vannevar Bush kuvasi tietokoneiden tulevaisuutta vuonna 1945:[256]

> ”Memex on laite, mihin henkilö tallentaa kaikki kirjansa, äänitteet ja viestintänsä, ja joka on mekanisoitu siten, että sitä voidaan hyödyntää äärimmäisen nopeasti ja joustavasti. Se on laajennettu intiimi muistin jatke.”

Memex ei saanut suosiota terminä, mutta ajatus on oiva kuvaus siitä mitä tietokoneesta on tullut vuosikymmeniä myöhemmin. Tietokoneet olivat aikanaan liian kalliita ja kömpelöitä henkilökohtaiseen käyttöön. Tämä tilanne jatkui pitkälle 1970-luvulle. DEC:n pääjohtaja Ken Olson ei vielä vuonna 1974 nähnyt mitään syytä miksi kukaan haluaisi oman tietokoneen.[257] Mikropiirien kehittyminen muutti sitten tilanteen nopeasti, mutta henkilökohtaisten tietokoneiden kehittäminen jäi aluksi pienempien yrittäjien pelikentäksi.

Miten sitten nykyaikainen tietokone varsinaisesti toimii?[258] John von Neumann jaotteli vuonna 1945 tietokoneen rakenteen kuuteen osaan: aritmeettinen laskenta, ohjausosa, muisti, syöttö, tulostus ja ulkoinen muisti. Eräs merkittävä oivallus oli se, että data ja ohjelmat voivat käyttää samaa muistia eli jakaa saman osoiteavaruuden. Vaihtoehtoinen tapa on käyttää eri osoiteavaruuksia ja eri väyliä toisaalta datalle ja toisaalta ohjelmistokäskyille. Näistä kahdesta toimintaperiaatteesta käytetään nimityksiä von Neumannin arkkitehtuuri (von Neumann architecture) ja Harvard-arkkitehtuuri (Harvard architecture). Koska von Neumannin jaettu väylä voi muodostua suorituskykyä rajoittavaksi tekijäksi, esimerkiksi suorittimen välimuistissa käytetään usein Harvard-arkkitehtuuria vaikka tietokoneessa muilta osin sovellettaisiin von Neumannin arkkitehtuuria.

---

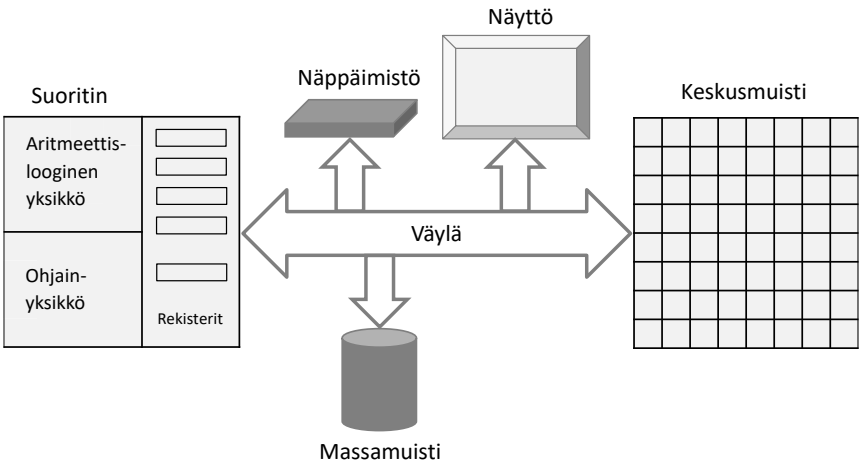
[255] Unicoden merkistöä kehittää ja hallinnoi Unicode Consortium (http://unicode.org/) joka aloitti toimintansa vuoden 1991 alussa. Käytännössä ISO/IEC 10646 määrittelee samat merkistöt. https://fi.wikipedia.org/wiki/Unicode

[256] V. Bush, As We May Think, *Atlantic*, June 1945 (myös kirjassa W. Isaacson, *The Innovators*, s. 263).

[257] DEC oli siihen aikaan, IBM:n jälkeen, suurimpia tietokonevalmistajia maailmassa.

[258] Yleisesitys tietokoneen toiminnasta: N. Nisan & S. Schocken: *The Elements of Computing Systems, Building A Modern Computer from First Principles*, The MIT Press, 2008.

Yleiskäyttöinen tietokone koostuu siten seuraavista pääkomponenteista (kuvassa 6.4): aritmeettis-looginen yksikkö (arithmetic logic unit, ALU), ohjainyksikkö (control unit), muisti (memory) ja siirräntälaitteet (I/O devices).[259]



Kuva 6.4. Tietokoneen keskeisimmät rakenneosat.

Eri komponentit on yhdistetty väylällä, joka tyypillisesti koostuu rinnakkaisista johdoista. Lisäksi voidaan erottaa rekisterit, joihin väylä on liitetty. Aritmeettis-looginen yksikkö, ohjainyksikkö ja rekisterit (register) muodostavat yhdessä suorittimen (prosessori, processor). Rekisterit ovat suorittimen muisteja, jotka tehostavat olennaisesti laskentaa verrattuna siihen, että kaikki data pitäisi hakea keskusmuistista. Aikaisemmin käytettiin paljon myös termiä keskusyksikkö (Central Processor Unit, CPU), joka käytännössä on lähes sama asia kuin suoritin.[260] Se mistä toiminnoista suoritin vastaa, on muuttunut ajan myötä integrointiasteen kasvaessa. Nykyisin yhdelle mikropiirille voidaan integroida monta laskentaa tekevää ydintä (core), välimuistia (cache) ja erilaisia ohjaintoimintoja, jotka kaikki olisi aikaisemmin toteutettu erillisillä piireillä. Kuvassa 6.5 on esimerkki pienikokoisesta tietokoneesta ilman oheislaitteita.

Tietokoneen osat koostuvat elektronisista komponenteista eli mikropiireistä (integrated circuit). Yhdelle mikropiirille voidaan integroida suuri määrä, jopa miljardeja, aktiivisia (transistoreita ja diodeja) ja passiivisia (vastuksia ja kondensaattoreita) komponentteja. Suoritinosassa piirit on järjestetty siten, että ne muodostavat loogisia kokonaisuuksia, jossa piirit ohjaavat toisten piirien toimintaa Boolen algebran (Boolean algebra) periaatteiden mukaisesti.

[259] Sanastokeskuksen termipankin (http://www.tsk.fi/tepa/fi/) mukainen termi on syöttö-tulostus.
[260] Keskusyksikön voidaan myös ajatella sisältävän suorittimen (tai suorittimien) lisäksi väylän.

Kuva 6.5. Esimerkki tietokoneesta käytännössä (Raspberry Pi 2).

Ohjainyksikkö

Ohjainyksikkö hallitsee tietokoneen eri osia, lukee ja tulkitsee ohjelman käskyjä ja muuntaa ne muotoon, jolla voidaan ohjata muita tietokoneen osia. Ohjainyksikkö voi jopa muuttaa käskyjen suoritusjärjestystä suorituskyvyn lisäämiseksi. Tärkeä osa ohjausyksikön toimintaa on ohjelmalaskuri (program counter), joka pitää kirjaa siitä mistä muistipaikasta seuraava käsky pitää lukea. Ohjainyksikön keskeisimmät tehtävät ovat (tehtävien suoritusjärjestys saattaa vaihdella ja niitä voidaan suorittaa rinnakkain):

- Seuraavan käskyn lukeminen ohjelmalaskurin ilmoittamasta paikasta.
- Numeerisen koodin muuntaminen joukoksi käskyjä järjestelmän muille osille.
- Ohjelmalaskurin arvon kasvattaminen siten, että se osoittaa seuraava käskyä.
- Tarvittavan datan lukeminen muistista tai mahdollisesti oheislaitteesta.
- Tarvittavan datan lähettäminen aritmeettis-loogiseen yksikköön tai rekisteriin.
- Tarvittaessa muiden yksiköiden ohjaaminen suorittamaan vaaditut operaatiot.
- Aritmeettis-loogisesta yksiköstä saadun tuloksen kirjoittaminen muistiin, rekisteriin tai mahdollisesti tulostuslaitteeseen.
- Palaaminen takaisin ensimmäiseen kohtaan.

Koska ohjelmalaskuri on kuin mikä tahansa muistipaikka, sen arvoa voidaan muuttaa ALU:n laskelmien mukaisesti. Tällä tavoin voidaan tehdä ohjelmallisesti hyppyjä tai silmukoita. Edellä luetellut operaatiot ovat siinä määrin monimutkaisia, että ne muodostavat itsessään lyhyen tietokoneohjelman. Usein tätä tehtävää hoitaa yksi tai useampi erillinen ohjausyksikkö, joissa ajetaan mikrokoodia (mikro-ohjelma, microcode).

Aritmeettis-looginen yksikkö

Aritmeettis-loogisessa yksikössä suoritetaan nimensä mukaisesti kahdentyyppisiä toimintoja: aritmeettisia ja loogisia. Aritmeettiset operaatiot voivat rajoittua jopa yhteen- ja vähennyslaskuun, tai operaatiot voivat kattaa huomattavasti monimutkaisempia toimintoja

kuten kerto- ja jakolaskuja, trigonometrisiä funktioita, logaritmeja ja neliöjuuria. Laskutoimitukset voidaan joko tehdä ainoastaan kokonaisluvuille tai numerot voidaan esittää liukulukuina. Yksinkertaisten operaatioiden avulla voidaan toteuttaa mielivaltaisen monimutkaisia laskuja jakamalla tehtävä riittävän pieniin palasiin.

Loogiset operaatiot sisältävät Boolen logiikan toiminnot JA (∧, AND), TAI (∨, OR) ja EI (¬, NOT). Lisäksi voidaan määritellä mm. ”poissulkeva tai” (⊕, XOR)[261]. Taulukossa 6.2 on esitetty totuusarvot kommutatiivisille (commutative) operaatioille (operation, op), eli siis operaatioille joille pätee aina: a op b = b op a. Näistä voidaan koostaa mielivaltaisen monimutkaisia ehdollisia lausekkeita. Jopa pelkällä NOR-operaatiolla voidaan toteuttaa kaikki tarvittavat lausekkeet.

Taulukko 6.2. Totuusarvotaulukko kommutatiivisille binäärisille operaatioille.

		Epätosi	JA		TAI				Tosi		
		FALSE	AND	XOR	OR	NOR	XNOR	NAND	TRUE		
a	b		∧	⊕	∨	↓		| tai ↑		¬a	¬b
0	0	0	0	0	0	1	1	1	1	1	1
0	1	0	0	1	1	0	0	1	1	1	0
1	0	0	0	1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	1	0	1	0	0

Boolen algebran peruslait ovat:

- $a \vee \neg a = 1$ $a \wedge \neg a = 0$
- $a \vee b = b \vee a$ $a \wedge b = b \wedge a$
- $a \vee (a \wedge b) = a$ $a \wedge (a \vee b) = a$
- $a \vee (b \vee c) = (a \vee b) \vee c$ $a \wedge (b \wedge c) = (a \wedge b) \wedge c$
- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

Näiden lausekkeiden avulla voidaan lausekkeita muuntaa muodosta toiseen ilman, että lausekkeen totuusarvo muuttuu. Lisäksi DeMorganin lait (De Morgan's laws) ovat usein käyttökelpoisia. Kolmelle muuttujalle (a, b ja c) esitettynä lait ovat:

$$\neg(a \vee b \vee c) = \neg a \wedge \neg b \wedge \neg c$$

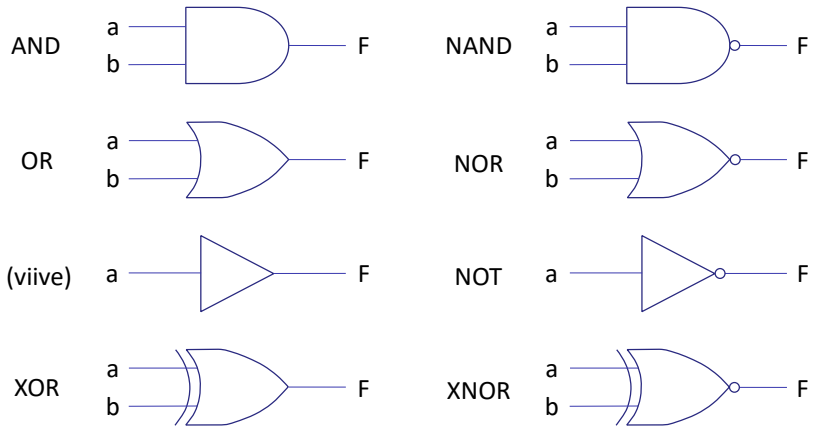
$$\neg(a \wedge b \wedge c) = \neg a \vee \neg b \vee \neg c$$

Sama periaate voidaan yleistää rajattomalle määrälle muuttujia.

[261] *Poissulkeva tai* voidaan esittää monessa muodossa, esimerkiksi: $a \oplus b = (a \wedge \neg b) \vee (\neg a \wedge b)$

Loogisten operaatioiden tekninen toteutus

Tietokone perustuu siis loogisten operaatioiden tehokkaaseen eli nopeaan ja luotettavaan suorittamiseen. Miten kaikki miljardit operaatiot sitten tapahtuvat fyysisellä tasolla? Tietokoneet perustuvat puolijohdeteknologiaan, jolla voidaan toteuttaa loogisia portteja (logic gate). Loogiset portit vastaavat kuvan 6.6 mukaisesti Boolen algebran operaatioita.



Kuva 6.6. Boolen algebran operaatioita vastaavien loogisten porttien esitystavat. Portti F = a aiheuttaa vain viiveen ilman mitään varsinaista operaatiota.

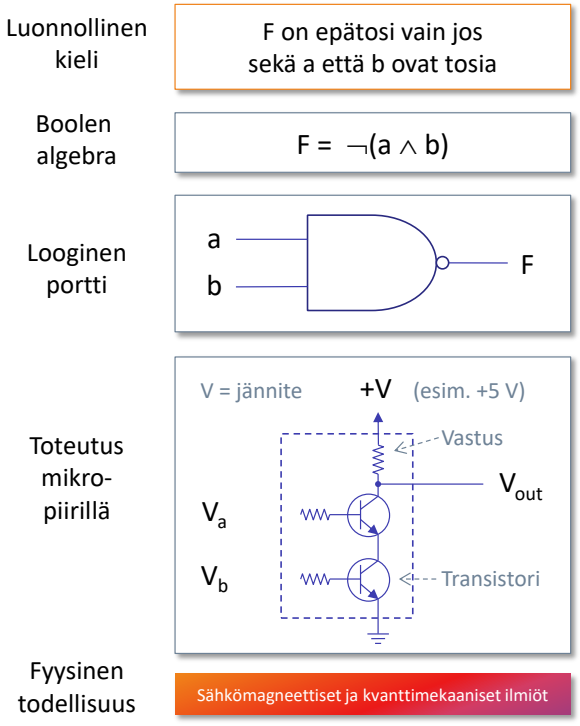
Looginen portti toimii jännitevahvistimena, jonka vahvistus on erittäin suuri. Sisääntulo ottaa hyvin pienen sähkövirran ja tuottaa ulostuloon jännitteen. Virta ei siten kulje sisääntulosta ulostuloon vaan ulostulon jännite ja virta tuotetaan erillisen sisääntulon kautta. Portin ulostulo voidaan yhdistää seuraavan portin sisääntuloksi, jolloin voidaan toteuttaa hyvin monimutkaisia loogisia piirejä ilman, että suunnittelijan tarvitsee miettiä loogisen portin sisäistä toteutusta. Käytännön toteutuksissa on omat rajoituksensa esimerkiksi sen suhteen, kuinka monta sisääntuloa yhdellä loogisella portilla voi olla (fan-in) ja kuinka moneen porttiin yksi ulostulo voidaan kytkeä (fan-out). Esimerkiksi kuvassa 6.8 (muutama sivu eteenpäin) vasemmanpuoleisessa kuvassa yhdessä portissa (oikealla ylhäällä) on neljä sisääntuloa. Toisaalta sisääntulo a on yhdistetty viiteen porttiin, joten sitä syöttävän porttipiirin fan-out täytyy olla vähintään 5. Yleensä sekä sisääntulojen että ulostulojen määrän lisääminen kasvattaa viivettä, joten niitä ei kannata kasvattaa kovin suuriksi ilman pakottavaa tarvetta. Vaikka viiveet ovat erittäin pieniä (ihmisen näkökulmasta katsottuna), äärimmäisen suurilla nopeuksilla tai kellotaajuuksilla viive voi muodostua suorituskykyä rajoittavaksi tekijäksi.

Huomaa, että se mikä näyttäytyy abstraktilla tasolla loogisina operaationa, joka käsittelee muuttujia (jotka voivat saada arvon 0 tai 1), on mikropiirien tasolla jännite. Yleensä arvoa 0 vastaa alempi jännite (käytännössä maa) ja arvoa 1 vastaa joku maahan verrattuna positiivinen jännite. Jännite pyritään pitämään mahdollisimman alhaisena, jotta piirin tehonkulutus ja lämmöntuotto pysyisivät siedettävinä. Lisäksi pienempi jännite mahdollistaa

korkeamman kellotaajuuden käytön koska jännite ehtii nousta tarvittavaan arvoon nopeammin (jännitteen muutokset tapahtuvat aina rajallisessa ajassa). Nykyisissä tehokkaissa prosessoreissa jännite on tyypillisesti hieman yli 1 V.

Integrointiaste (komponenttien määrä piirillä) riippuu olennaisesti käytetystä viivanleveydestä. Nykyisillä tiheimmin pakatuilla integroiduilla piireillä viivanleveys on 14 nm.[262] Pienemmillä viivanleveyksillä kvanttimekaaniset ilmiöt alkavat olennaisesti vaikuttaa elektronien käyttäytymiseen, mikä tekee piirien suunnitellusta erittäin vaikeaa. Vastaavasti, kellotaajuuksia on hyvin vaikea nostaa juurikaan yli 5 GHz:n, koska tehokulutus nousee suunnilleen suorassa suhteessa kellotaajuuteen.[263] Viime vuosina kehitys ei olekaan enää tapahtunut kellotaajuuden kasvattamisella vaan rinnakkaisuuden lisäämisellä.

Periaatteessa tietokoneen ja sen prosessorin toiminta perustuu siis sähkömagneettisiin ja kvanttimekaanisiin ilmiöihin ja kaikki muu sen yläpuolella on vain abstraktia kuvausta, transistorista alkaen, kuten kuvassa 6.7 on hahmoteltu. Tyypillisen tietotekniikan insinöörin kannalta atomifysiikan ilmiöt ovat kuitenkin toissijaisia, kunhan lopputulos sopivalla abstraktiotasolla on juuri se mihin pyritään.



Kuva 6.7. NAND-operaatio eri abstraktio- tai esitystasoilla. Huomaa erityisesti, että mikropiireissä sisään- ja ulostulot ovat jännitteitä, ei totuusarvoja.

[262] Tämä 14 nm on noin viidestuhannesosa tyypillisen hiuksen paksuudesta. Piin atomeja 14 nm leveyksiselle väylälle mahtuu rinnakkain noin 100 kappaletta.
[263] Vertailun vuoksi mainittakoon, että IBM PC:n kellotaajuus vuonna 1981 oli 4,77 MHz eli tuhannesosa nykyisistä.

Ohjelmoinnin kannalta olennaista on, että integroidulla piirillä toteutetut loogiset laskutoimitukset ovat äärimmäisen luotettavia. Esimerkiksi jos NAND-portin sisääntuloissa a = 0 ja b = 1, voimme olla äärimmäisen varmoja, että ulostulon arvo on 1. Toisaalta tämä varmuus vaatii, että toimintaympäristö on piirille sopiva (ei esimerkiksi liian kuuma) ja kellotaajuus on sopiva (jos kellotaajuus on liian suuri, niin ulostulo ei välttämättä ehdi vaihtua oikeaksi). Yhteenvetona voidaan todeta, että integroiduilla piireillä on neljä merkittävää etua: laskennan nopeus, pieni fyysinen koko, pieni tehonkulutus verrattuna laskentatehoon ja lähes virheetön toiminta.

Esimerkki 6.1. Yhteenlaskun toteuttaminen loogisilla porteilla

Suunnittele loogisten porttien avulla piiri, joka toteuttaa binäärilukujen yhteenlaskun.

Ratkaisu

Otetaan esimerkiksi desimaaliluvut a = 55 ja b = 114 ja mietitään mitä niiden yhteenlaskussa tapahtuu. Lasku binäärimuodossa on tällainen:

```
z   11101100
a   00110111
b   01110010
    --------
S   10101001
C   01110110
```

Yhteenlasku tapahtuu siten täysin samoin kuin 10-järjestelmässä. Aloitetaan oikealta ja havaitaan, että ensin pitää laskea yhteen 0 ja 1 ja tulokseksi saadaan 1 (tulosrivillä S, sanasta sum). Seuraavassa sarakkeessa lasketaan yhteen 1 ja 1, jolloin tulokseksi saadaan binääriluku 10. Tulosriville tulee siis 0 ja seuraavaan yhteenlaskuun jää 1, joka merkitään riville C, sanasta carry, Sillä tarkoitetaan siirtoa seuraavaan sarakkeeseen, joka on merkitty lyhenteellä z (ylin rivi). Kolmannessa sarakkeessa yhteenlaskettavia onkin siten kolme, eli 1, 1 ja 0. Lopputulos on jälleen C = 1 ja S = 0. Näin jatketaan, kunnes laskutoimitus on saatu suoritettua loppuun. Havaitaan, että koko laskutoimitus koostuu aina rakenteesta, joissa kolme bittiä lasketaan yhteen siten, että tulos voidaan esittää kahdella bitillä (S ja C). Yhteenlaskua varten tarvitsee suunnitella looginen piiri, joka toteuttaa taulukon 6.3 mukaiset totuusarvot.

Taulukko 6.3. Kolmen bitin (a, b ja z) yhteenlasku totuusarvotaulukkona.

a	b	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S saa arvon 1, jos joko yhdellä kolmesta muuttujasta tai kaikilla kolmella muuttujalla on arvo 1, muissa tapauksissa S =0, eli:

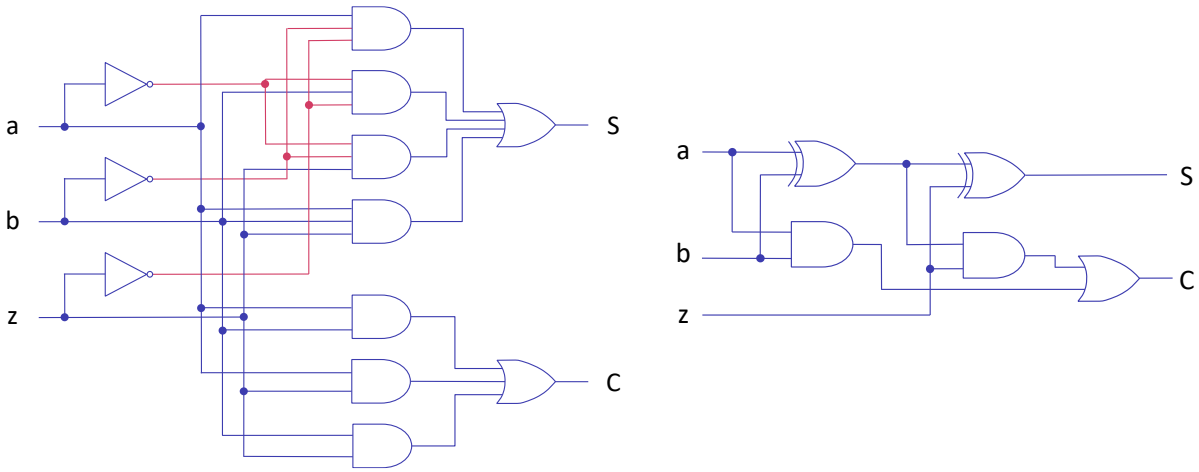
$$S = (a \wedge \neg b \wedge \neg z) \vee (\neg a \wedge b \wedge \neg z) \vee (\neg a \wedge b \wedge \neg z) \vee (a \wedge b \wedge z)$$

Vastaavasti C saa arvon 1, jos vähintään kahdella muuttujalla on arvo 1, eli:

$$C = (a \wedge b) \vee (a \wedge z) \vee (b \wedge z)$$

Tapausta a = b = z = 1 ei tarvitse käsitellä erikseen, koska muut tekijät kattavat tämän tapauksen.

”Suoraviivainen” toteutus perustuen edellä esitettyihin kaavoihin loogisilla porteilla JA, TAI, ja EI on esitetty kuvassa 6.8. Kuvan toteutus on kaukana optimaalisesta sen suhteen, kuinka monta loogista porttia tarvitaan. Täsmälleen sama totuusarvotaulukko voidaan toteuttaa viidellä loogisella portilla: kaksi XOR-, kaksi JA- ja yksi TAI -portti, joissa kussakin on vain kaksi sisääntuloa. Koko laskutoimitus, esimerkiksi 8 bitin yhteenlasku, saadaan toteutettua siten, että tietyn sarakkeen ulostulo C yhdistetään seuraavan sarakkeen sisäänmenoon z (ensimmäisessä sarakkeessa z on aina 0).



Kuva 6.8. Kolmen bitin yhteenlasku, vasemmalla ”suoraviivainen” toteutus, oikealla täsmälleen samaan lopputulokseen johtava yksinkertaisempi toteutus.

Muisti

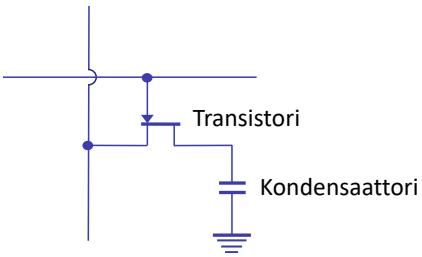
Muisti on ollut laskenta- ja tietokoneiden ongelma alusta saakka. Haasteena on ollut sekä muistista haun nopeus että muistin koko. Erityisesti siinä vaiheessa, kun laskenta suoritettiin tyhjiöputkilla, muistista muodostui yleensä suorituskykyä eniten rajoittava tekijä. Laskentakoneiden alkuvaiheessa oli monenlaisia muistivirityksiä, joista mielenkiintoisimpia oli mm. ENIAC:n seuranneessa EDVAC:ssa[264] käytetty akustinen viivelinja. Muisti perustui akustiseen signaaliin, joka kulki elohopealla täytetyssä putkessa. Vaikka tällaiset muistit olivat aikaisempia parempia, niiden avulla ei voitu tehdä kovin suuria muisteja. Viivelinja aiheutti myös huomattavan viiveen, varsinkin jos muisti oli suuri.

[264] EDVAC (Electronic Discrete Variable Automatic Computer) valmistui 1949 ja se sisälsi mm. 6000 tyhjiöputkea. Yhteenlaskuun käytetty keskimääräinen aika oli noin 0,8 ms. https://en.wikipedia.org/wiki/EDVAC

1950-luvun puolivälistä alkaen tietokoneissa siirryttiin ferriittirengasmuisteihin (magnetic core memory) kuvassa 6.9. Se perustuu pieniin keraamisiin renkaisiin, joihin tieto tallennetaan magneettikentän polariteetin avulla siten, että sisällön säilyttämiseen ei tarvita sähköenergiaa. Ferriittirengasmuistien hakuajoissa päästiin 1970-luvulla jo 600 nanosekunnin luokkaan.[265] Tämän jälkeen siirryttiin integroituihin puolijohdemuisteihin. Kuvassa 6.5 esitetyn pienen tietokoneen keskusmuisti on 1 gigatavun kokoinen eli kaksisataakertainen verrattuna kuvan 6.9 muistiin.[266] Muistikapasiteetti painoyksikköä kohti on kaksinkertaistunut keskimäärin kahden vuoden välein 1950-luvun lopulta nykypäivään.

Kuva 6.9. IBM:n viiden megatavun muistiyksikkö vuodelta 1956, paino yli 1000 kiloa.[267]

Muistia on periaatteessa kahta tyyppiä: luku- ja kirjoitusmuisti (Random Access Memory, RAM) ja lukumuisti (Read Only Memory, ROM). Tietokoneen keskusmuisti on aina RAM-tyyppistä. RAM voi olla joko dynaaminen (dynamic RAM, DRAM) tai staattinen (static RAM, SRAM). Dynaamista muistia täytyy virkistää jatkuvasti, sillä se perustuu kondensaattorien käyttöön, joiden varaus osittain purkautuu niitä luettaessa. Kuvassa 6.10 on esitetty DRAM:n yhden bitin säilyttämiseen tarvittava muistielementti, joka siis koostuu yhdestä transistorista ja yhdestä kondensaattorista (capacitor).

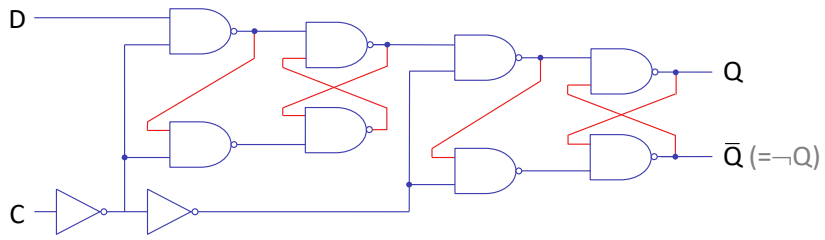


Kuva 6.10. Dynaamisen RAM:n yhden bitin muistiyksikkö.

ROM-muistia käytetään tyypillisesti tietokoneen alustukseen käynnistettäessä tietokonetta. Käynnistystä varten ROM-muistissa on BIOS (Basic Input/Output System) ohjelma, jonka avulla massamuistista ladataan tietokoneen käyttöjärjestelmä keskusmuistiin. Lisäksi suoritin voi käyttää välimuistia (cache), joka on hitaampi kuin rekisteri mutta nopeampi kuin keskusmuisti. Tietokone voi siirtää dataa eri muistiyksiköiden välillä ilman, että sovellusten ohjelmoijien tarvitsee puuttua asiaan. Tämän asian hoitaa muistinhallintayksikkö (Memory Management Unit, MMU).

[265] http://fi.wikipedia.org/wiki/Ferriittirengasmuisti
[266] Koska kyseessä on keskusmuisti, 1 GB luultavasti tarkoittaa 2^{30} = 1 073 741 824 bittiä kuten 3. luvussa mainittiin.
[267] https://nextshark.com/ibm-5mb-hard-drive/

Myös suorittimen sisällä tarvitaan muistia. Pieniä muisteja voidaan toteuttaa loogisten porttien avulla. Kuvassa 6.11 on esitetty yhden bitin muisti, jonka toteuttamiseen on käytetty 8 NAND-porttia ja 2 NOT-porttia. Olennaisin ero esimerkiksi kuvassa 6.8 esitettyihin loogisiin piireihin on takaisinkytkennät, joiden avulla piiri muistaa aikaisemman tilansa. Kuvan rakennetta kutsutaan kiikuksi (flip-flop). Kiikun tehtävänä on siis säilyttää sille asetettu looginen tila, eli se muistaa yhden bitin. Säilytettävän bitin arvo riippuu kiikun tyypistä, tulojen tilasta ja lähtöjen edellisestä tilasta. Kuvassa 6.11 kellosisääntulon (C) muutos nollasta ykköseksi laukaisee prosessin, jonka lopputuloksena ulostulo (Q) saa datasisääntulon (D) arvon ja pitää sen, kunnes kellosisääntulo muuttuu seuraavan kerran nollasta ykköseen. Suhteellisen monimutkainen rakenne tarvitaan sen takaamiseksi, että piiri toimii kaikissa tilanteissa täysin ennustettavalla tavalla. Piirin yksityiskohtaista toimintaa on vaikeahkoa hahmottaa, eikä sitä tämän kurssin puitteissa käsitellä.



Kuva 6.11. Yhden bitin muisti toteutettuna NAND- ja NOT-porteilla.[268]

Pääsyntä (Input/output)

Pääsynnällä tarkoitetaan niitä keinoja, joilla tietokone vaihtaa informaatiota ulkomaailman kanssa. Laitteita, joita käytetään tähän tarkoitukseen, kutsutaan oheislaitteiksi (peripherals). Oheislaitteita on periaatteessa neljää tyyppiä: 1) syöttölaitteet (input devices), 2) tulostuslaitteet (output devices), 3) muistilaitteet (storage devices) ja 4) yhdistetyt syöttö- ja tulostuslaitteet (input/output). Henkilökohtaisessa tietokoneessa tyypillisiä oheislaitteita ovat näyttö, näppäimistö, hiiri, kaiuttimet, mikrofoni ja tulostin. Lisäksi muut tallennuslaitteet paitsi keskusmuisti luokitellaan oheislaitteeksi, samoin verkkoyhteydet.

Esimerkkejä tiedonsiirtoon käytettyjen liitäntöjen nopeuksista:[269]

- USB 1.1: 12 Mbit/s: (v. 1996)
- USB 2.0: 480 Mbit/s (2000)
- USB 3.0: 5 Gbit/s (2010)
- USB 3.1: 10 Gbit/s

[268] Kuva https://fi.wikipedia.org/wiki/Kiikku_%28digitaalitekniikka%29, kiikkujen toimintaa on selostettu tarkemmin esimerkiksi Wikipedian englanninkielisillä sivulla https://en.wikipedia.org/wiki/Flip-flop_%28electronics%29 ja alan oppikirjoissa, esim. M. Mano & C. Kime: Logic and Computer Design Fundamentals, s. 191-201.
[269] http://en.wikipedia.org/wiki/List_of_device_bit_rates

- HDMI 1.0: 4.95 Gbit/s [270]
- HDMI 2.0: 18 Gbit/s

Käytännössä saavutettavat nopeudet voivat olla huomattavasti alhaisempia. Toisaalta uusimpien standardien nopeudet ovat suuria: kokonaisen elokuvan voi ladata muutamassa sekunnissa. Kehitys on menossa siihen suuntaan, että samaa liitäntää käytetään kaikkeen tiedonsiirtoon sekä lisäksi tehonsyöttöön. Tähän pyritään USB Type-C -liitännällä.[271]

Tietokoneen sisäisten väylien nopeus on parhaimmillaan useita satoja gigabittejä sekunnissa. Sen sijaan kiintolevyiltä voidaan lukea dataa huomattavasti hitaammin ja lisäksi saavutettava nopeus riippuu siitä, miten data sijaitsee levyllä. Useat oheislaitteet ovat monimutkaisia laitteita ja ne voidaan hyvin luokitella tietokoneiksi. Esimerkiksi aikanaan tulostimissa saattoi olla niin tehokas prosessori, että sillä saattoi ratkaista tietyntyyppisiä laskennallisia tehtäviä nopeammin kuin tavallisella tietokoneella. Nykyisinkin näytönohjaimissa on enemmän tietojenkäsittelykapasiteettia kuin useimmissa yleiskäyttöön suunnitelluissa suorittimissa. Tosin näytönohjain suorittaa tietojenkäsittelyä rinnakkain kymmenissä suorittimissa, jotka suorittavat saman operaation samanaikaisesti suurelle datamäärälle, joten näytönohjainta ei voi käyttää samalla tavoin kuin täysin yleiskäyttöistä suoritinta.

Tietokoneen nopeuden havainnollistaminen

Oletetaan, että tietokoneen kellotaajuus on 3,15 GHz eli se pystyy tekemään yksinkertaisen toimenpiteen 3 150 000 000 kertaa sekunnissa. Miten tämän äärimmäisen nopeuden voisi ymmärtää käyttäen apuna arkipäivän ilmiöitä? Hahmotamme ulkomaailman sekunnin aikaskaalalla. Vaikka voimme havaita lyhyempiäkin tapahtumia, mikä tahansa havainto vaatii aina noin 400 ms ajan ennen kuin voimme tulla siitä tietoiseksi. Eli noin sekunti on lyhyin aika mikä riittää tietoiseen reagointiin ulkoisen ärsykkeeseen. Aivojen ”kellotaajuus” on toki korkeampi, mutta tässä vertailukohtana on siis tietoiset teot.

Miten kauan siis kestää tietoisessa maailmassa siihen, minkä tietokoneen maailmassa kestää yhden sekunnin? 3 150 000 000 sekuntia on 100 vuotta. Eli jos suorittimessa olisi 5 miljardia transistoria ja yksi henkilö pystyisi tekemään saman kuin transistori mutta 1 sekunnin aikana, niin tarvittaisiin suunnilleen kaikki maailman aikuiset 100 vuoden ajan tekemään sama mitä yksi suoritin tekee sekunnissa. Huh. Ei ihme, että nykyiset tietokoneet tekevät ihmeellisiä asioita. Tällä mallilla voi myös hahmottaa erilaisten muistien ja oheislaitteiden nopeuseroja kuten taulukossa 6.4 on esitetty.

[270] HDMI (High-Definition Multimedia Interface) on liitäntästandardi kuvan ja monikanavaäänen siirtämiseen.
[271] https://en.wikipedia.org/wiki/USB-C

Taulukko 6.4. Tietokoneen toimintojen aikaskaala muunnettuna tietoiselle aikaskaalalle.

	Kestoaika tietokoneen toimintanopeudella (kellotaajuus 3,15 GHz)	Kestoaika ihmisen tietoisella toiminta-nopeudella (1 Hz)
Yksi kellojakso	0,32 ns	1 s
Datan haku erittäin nopeasta välimuistista	10 ns	32 s
Datan haku RAM-muistista	1 µs	53 min
Datan haku kiintolevyltä	1 ms	36 päivää
Aika näppäimistön painallusten välillä	0,2 s	20 vuotta
Ihmisen tarvitsema aika näytön lukemiseen	30 s	3000 vuotta
Käyttöjärjestelmän päivitys	2 tuntia	720 000 vuotta
Yhden tavun siirto 1 Gbit/s linkillä	8 ns	24 s
Yhden megatavun siirto 1 Gbit/s linkillä	8 ms	10 kk
Yhden teratavun siirto 1 Gbit/s linkillä	133 min	800 000 vuotta

Moniajo ja rinnakkaislaskenta

Useimmissa nykyisissä järjestelmissä on välttämätöntä ajaa lukuisia ohjelmistoja siten, että ulospäin näyttää siltä kuin niitä ajettaisiin samanaikaisesti. Käytännössä tämä tapahtuu moniajona, jossa tietokone vaihtaa hyvin nopeasti ajettavaa ohjelmaa. Perinteisesti moniajo (multitasking) mikrotietokoneissa on tarkoittanut yhden suorittimen käyttöä. Ensimmäinen moniajotapa oli yhteistyömoniajo, missä ohjelmat vapaaehtoisesti jakoivat suoritinaikaa toisille ohjelmille.[272] Ongelmana tällaisessa järjestelyssä on, että yksi huonosti käyttäytyvä ohjelma voi varastaa koko koneen laskentatehon. Nykyaikaisissa tietokoneissa käyttöjärjestelmän osa, vuorontaja (scheduler) hoitaa vuorojen jaon. Koska tietokoneet toimivat nykyisin monta kertaluokkaa nopeammin kuin ihminen voi havaita, inhimillisestä näkökulmasta katsottuna ohjelmat näyttävät pyörivän rinnakkain. Toiminta on siis periaatteessa samankaltaista kuin aikajakoinen kanavointi, jota käsiteltiin luvussa 3.

Voisi kuvitella, että jatkuva ohjelmien vaihtaminen heikentäisi tietokoneen suorituskykyä. Käytännössä monet ohjelmat kuitenkin käyttäisivät suurimman osan ajasta (jos suoritin olisi siis pelkästään yhden ohjelman käytössä) odottaen jonkun oheislaitteen toimintoja. Erityisen hidas on ihminen näppäimistön tai hiiren käyttäjänä (kuten taulukko 6.4 havainnollistaa). Jotkut intensiiviset ohjelmat, kuten virustorjuntaohjelmisto tarkistaessaan kiintolevyn sisältöä, voivat vaikuttaa havaittavasti muiden ohjelmien suorittamiseen.

Sama tehtävä voidaan myös jakaa useamman suorittimen tehtäväksi. Rinnakkaislaskenta (parallel processing) tarkoittaa yhden laskentatehtävän ratkaisemista samanaikaisesti useita suorittimia tai suoritinytimiä (multi-core) käyttämällä. Rinnakkaislaskennan avulla suurikin laskentatehtävä voidaan ratkaista nopeasti, jos se pystytään jakamaan

[272] http://fi.wikipedia.org/wiki/Moniajo

pienempiin tehtäviin siten, että tehtävät voidaan suorittaa pääosin toisistaan riippumatta. Rinnakkaislaskennan haasteena on tehtävien jakaminen ja laskentaprosessien välinen tiedonsiirto ja keskinäinen ajoittaminen. Saavutettava nopeusetu ei siten ole läheskään suorassa suhteessa suorittimien lukumäärään. Rinnakkaislaskenta oli vuosikymmenien ajan vain tehokkaimmissa tietokoneissa käytetty ominaisuus. Rinnakkaislaskennan käyttö on nyt levinnyt pieniinkin laitteisiin kuten älypuhelimiin, jossa suoritinpiirillä voi olla kahdeksan rinnakkaista laskentaa suorittavaa ydintä. Vastaavasti supertietokoneissa rinnakkaisia suorittimia voi olla tuhansia tai jopa miljoonia.

Virheiden havaitseminen ja korjaus

Kuten kanavakoodauksen yhteydessä jo lyhyesti käsiteltiin, datan oikeellisuutta voidaan tarkistaa esimerkiksi tarkistussumman avulla. Käytännössä tarkistussumman laskentaa ei tehdä tyytymällä laskemaan vain ykkösbittien lukumäärää, vaan erilaisin polynomimenetelmin, joissa biteille annetaan painokertoimia niiden sijaintipaikan mukaan.

Piirien ja tallennustiheyden kasvaessa riski bittivirheisiin kasvaa puolijohdemuisteissa mm. kosmisten hiukkasten aiheuttamien virheiden takia. Palvelinkoneissa keskusmuisti on yleensä virheenkorjaavaa ja modernit levyjärjestelmät käyttävät tarkistussummia havaitsemaan bittivirheitä. Bittivirhe saattaa syntyä siis jo tietokoneen sisällä, joten tiedon suojaaminen vain siirron aikana ei välttämättä riitä.

Suunnittelemalla tarkistussummat sopivasti voidaan virheitä myös korjata. Yksinkertaisin yhden virheen korjaava menetelmä on Richard Hammingin mukaan nimetty Hamming-koodi (Hamming code). Hamming-koodi perustuu pariteettibitteihin, joita lasketaan järjestelmällisesti eri kohdista kehystä. Esimerkiksi [7/4] Hamming-koodissa on neljä databittiä ja kolme pariteettibittiä siten, että ensimmäinen bitti on pariteettibitti biteille 1, 3, 5 ja 7, toinen bitti on pariteettibitti biteille 2, 3, 6 ja 7 ja neljäs bitti on pariteettibitti biteille 4, 5, 6 ja 7 kuten taulukossa 6.4 on havainnollistettu. Samaa periaatetta voidaan jatkaa periaatteessa niin pitkälle kuin halutaan. Kehyksen koko on siten aina 2^r -1 bittiä, jossa r on kahta suurempi kokonaisluku. Pariteettibittejä on r kappaletta ja loput 2^r - r - 1 ovat databittejä.

Hamming-koodilla voidaan *joko* korjata 1 bittivirhe *tai* havaita korkeintaan kaksi bittivirhettä. Jos yksittäisiä bittivirheitä pyritään korjaamaan, kahden bitin virhe aiheuttaa virheellisen korjauksen. Lisäämällä yksi pariteettibitti voidaan samanaikaisesti korjata yhden bitin virheet ja havaita kahden bitin virheet.

Jos oletetaan, että ensimmäinen bitti on vähiten merkitsevä, niin esimerkiksi kahdeksan bitin tavu `10011000` vastaa kymmenjärjestelmässä lukua $2^0+0+0+2^3+2^4+0+0+0$ = 1+0+0+8+16+0+0+0=25. Toisaalta on syytä huomata, että Hamming-koodaus ei itsessään

tee mitään oletusta bittien tulkinnasta, vaan tulkinta on ylemmän eli sovellustason kysymys. Bittien järjestyksen suhteen on kuitenkin syytä olla tarkkana![273]

Taulukko 6.5. Hamming-koodauksen periaate. Databitit: d1, d2, … , pariteettibitit : p1, p2, …. Esimerkiksi pariteettibitti p1 kattaa kaikki järjestysnumeroltaan parittomat bitit (3, 5, 7, 9, …).

Bitti	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7	d8	d9	d10	d11	p5	d12	d13
p1	x		x		x		x		x		x		x		x		x	
p2		x	x			x	x			x	x			x	x		x	x
p3				x	x	x	x					x	x	x	x			
p4								x	x	x	x	x	x	x	x			
p5																x	x	x

Esimerkki 6.2. Hamming-koodaus

Vastaanotat Hamming-koodatun kehyksen 0110111. Onko kehyksessä virheellinen bitti tai bittejä? Jos oletetaan, että kehyksessä on yksi virheellinen bitti, niin mitä ovat lähetetyn kehyksen databitit? (Punaisella merkityt bitit ovat siis pariteettibittejä.)

Ratkaisu

Tässä siis vastaanotetut bitit ovat: p1 = 0, p2 = 1, d1 = 1, p3 = 0, d2 = 1, d3 = 1 ja d4 = 1. Siten:

p1 + d1 + d2 + d4 = 0 + 1 + 1 + 1 = 3, joten tulos on pariton eli ilmoittaa virheestä.

p2 + d1 + d3 + d4 = 1 + 1 + 1 + 1 = 4, joten tulos on parillinen ja siten virheetön.

p3 + d2 + d3 + d4 = 0 + 1 + 1 + 1 = 3, joten tulos on pariton eli ilmoittaa virheestä.

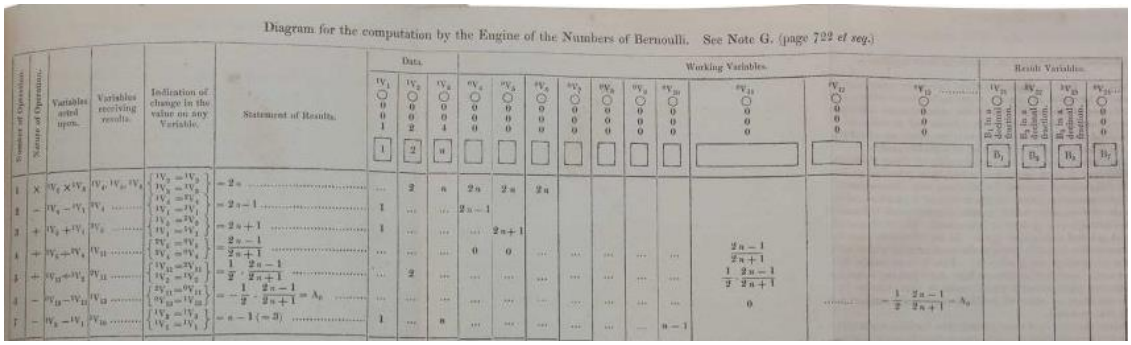
Koska oletettiin, että virheitä on korkeintaan yksi, voidaan keskimmäisen rivin perusteella päätellä, että bitit p2, d1, d3 ja d4 ovat oikein. Tästä seuraa käyttäen lisäksi ensimmäisen rivin tietoa, että joko p1 tai d2 on virheellinen (mutta ei molemmat). Vastaavasti kolmannen rivin mukaan joko p3 tai d2 on virheellinen. Tästä seuraa, että d2 eli toinen databitti on virheellinen, joten sen pitäisi olla 0 eikä 1. Jos siis lähetys sisälsi vain yhden virheen, alkuperäinen kehys oli 0110011 eli lähetetyt databitit olivat 1011 (kymmenjärjestelmässä 1 + 0 + 4 + 8 = 13, kun oletetaan, että vähiten merkitsevä bitti on ensimmäisenä).

[273] Katso esimerkiksi https://fi.wikipedia.org/wiki/Tavuj%C3%A4rjestys

Algoritmit, ohjelmointi ja ohjelmistot

Ohjelmointikielien kehitys

Walter Isaacson aloittaa kirjassaan digitaalisen vallankumouksen käsittelyn, hieman yllättäen, naishahmosta nimeltä Ada Lovelace, joka eli Englannissa 1800-luvun alkupuolella.[274] Ada Lovelacea pidetään ensimmäisenä, joka kehitti ja esitti ajatuksen ohjelmoitavasta laskentakoneesta ja siinä käytettävistä algoritmeista. Tämä tapahtui vuonna 1843! Kuvassa 6.12 on esitetty Lovelacen kehittämä algoritmi Bernoullin lukujen laskemiseksi.



Kuva 6.12. Osa ensimmäisestä julkaistusta algoritmista tietokoneelle: Bernoullin lukujen laskeminen, Ada Lovelace, 1843.[275]

Mutta ei Adakaan olisi voinut ajatuksiaan kehittää ilman Charles Babbagen mekaanista laskentakonetta, jonka tarkoituksena oli alun perin logaritmitaulukoiden laskeminen. Babbage sai tähän tarkoitukseen rahoitusta valtiolta, mutta ei valitettavasti saanut koskaan laitteitaan täysin valmiiksi. Ongelmaksi muodostui se, että hänellä oli vain yksi tekninen avustaja. Kunnolla toimivan koneen rakentaminen olisi vaatinut suuremman ryhmän asialle omistautuneita riittävän kyvykkäitä eri alojen edustajia. Ohjelmoinnin kehitys pysähtyikin lähes täysin sadaksi vuodeksi. Seuraava merkittävä askel oli vasta tämän luvun alussa mainittu ENIAC.

Vaikka lähes kaikki laitteistokehittäjät olivat miehiä, ensimmäiset ohjelmoinnin kehittäjät olivat naisia! Miksi näin? ENIAC:n kuten muidenkin alkuvaiheen laskentalaitteiden ajatuksena oli korvata ihmistyövoimalla ja mekaanisilla laskimilla suoritetut laskentatehtävät, joita useimmiten tekivät naiset.[276] ENIAC:n tapauksessa ”ohjelmointi” oli hyvin työlästä ja aikaa vievää, sillä mitään nykyisen kaltaisia ohjelmointikieliä ei ollut. ENIAC:n ohjelmointiin käytettiin 3000 kytkintä ja kymmeniä johtoja, joilla mekaanisesti ohjattiin koneen

[274] Ada Lovelace oli runoilija lordi Byronin tytär jonka elämä kesti, isänsä tapaan, vain 36 vaiheikasta vuotta. Ohjelmointikieli Ada nimettiin vuonna 1979 Ada Lovelacen mukaan.

[275] "Diagram for the computation of Bernoulli numbers" by Ada Lovelace, Licensed under Public Domain via Wikimedia Commons, http://www.sophiararebooks.com/pictures/3544a.jpg.

[276] Laskentaa suorittavista henkilöistä käytettiin nimeä ”computer” jo 1600-luvulla ja vielä 1940-luvulla

sähköistä toimintaa. Työ vaati äärimmäistä kärsivällisyyttä ja huolellisuutta. Ohjelmointitehtävä annettiinkin aikaisemmin itse laskentaa tehneille naisille, jotka hoitivat vaativan ohjelmoinnin erittäin ansiokkaasti.

Nopeasti tuli ilmeiseksi, että tietokoneiden käyttö eli siis ohjelmointi piti saada kätevämmäksi. Esimerkiksi A-2 kielellä, kun jokin muuttuja haluttiin korottaa kolmanteen potenssiin, koodi oli:

```
APN034 012038
```

Tässä muuttujan arvo oli muistipaikassa 34 ja laskun tulos sijoitettiin muistipaikkaan 38. Varmaankin koodia saattoi oppia lukemaan ja kirjoittamaan, mutta intuitiivista se ei ollut. Ohjelmointi oli kuitenkin helpompaa kuin kytkinten ja johtojen asettelu. Olennaista edistystä oli se, kun ohjelmointikielistä saatiin kehitettyä luonnollista (yleensä englannin) kieltä muistuttavia. Näistä ensimmäinen merkittävää suosiota saanut kieli oli Fortran, joka julkaistiin vuonna 1957. Fortran on edelleen laajassa käytössä, joskin siihen on tehty merkittäviä laajennuksia vuosien varrella. Fortranilla potenssiin korotus ilmaistiin näin:

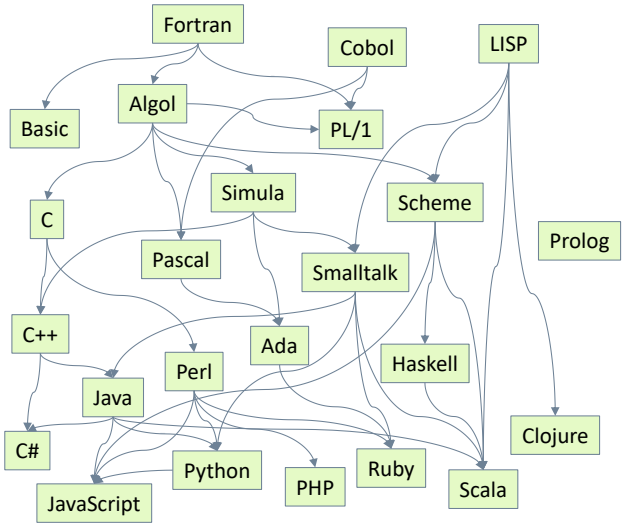
```
Y = T**3
```

Muita Fortranin uusia ominaisuuksia olivat mahdollisuus lisätä kommentteja koodin joukkoon sekä sisään- ja ulostuloa koskevat muotoilukomennot.[277] Vaikka Fortran-koodin lukeminen oli huomattavasti helpompaa kuin konekielen lukeminen, niin alkuvaiheen optimismi koodin ymmärrettävyydestä:

> “After an hour course in FORTRAN notation, the average programmer can fully understand the steps of a procedure stated in FORTRAN language without any additional comments.”[278]

oli ehkä realistista muutaman ohjelma-askeleen osalta. Pidempi ohjelma, jossa käytetään ehdollisia silmukoita ja muita monimutkaisia ominaisuuksia, on paljon vaikeampi ymmärtää. Ohjelmointikieliä on kehitetty sadoittain erilaisiin tarpeisiin. Kuvassa 6.13 on esitetty muutama suhteellisen yleisesti käytetty ohjelmointikieli ja karkea arvio eri kielten välisistä kehityssuhteista.

[277] Katso esimerkiksi: Knuth, D. E., & Pardo, L. T., The early development of programming languages, kirjassa N. Metropolis (ed.), *A history of computing in the twentieth century*, Elsevier, s. 197-273.
[278] Preliminary report, Specifications, for the IBM Mathematical FORmula TRANslating System, New York: IBM Cort., Nov. 1954.



Kuva 6.13. Eräitä ohjelmointikieliä ja niiden välisiä suhteita.

Mitä näistä sadoista kielistä sitten informaatioteknologian alalla toimivan asiantuntijan olisi syytä osata itse koodata, jos tähtäimessä ei ole ura ohjelmoijana? Kysymyshän ei ole lainkaan siitä, tarvitsevatko kaikki ohjelmointitaitoa ja -kokemusta vai ei, vaan siitä miten paljon ja miten syvällistä. Yksi vastaus voisi olla: C ja Python, jotka antavat suhteellisen kattavan kuvan ja kokemuksen hieman eri tason ohjelmointikielistä. Lisäksi kokemusta olisi hyvä hankkia koodaamalla jotain alhaisen tason konekieltä, koska silloin ohjelman ja varsinaisen tekniikan välinen rajapinta tulee konkreettisesti tutuksi. Toisessa suunnassa on hyödyllistä tutustua ns. ohjelmistokehyksiin (software framework). Ohjelmistokehys on ohjelmistotuote, joka muodostaa rungon, jonka päälle voi rakentaa tietokoneohjelmia.

Ohjelmointikielet voidaan siten jakaa taulukon 6.6 mukaisesti eri tasoihin. Rajat tasojen välillä ovat hämäriä. Esimerkiksi Perl on alun perin suunniteltu lähinnä tekstinkäsittelyyn, mutta siitä on käytännössä tullut yleiskäyttöinen (korkean tason) ohjelmointikieli. Erityisen epämääräinen on luokittelu korkean tason ja symbolisen konekielen välimaastossa. Suhteellisen selkeää on, että konekielen ja korkeimman tason kielten välillä on välimuotoja, mutta mitkä kielet kuuluvat niihin ja mitä nimeä tasosta käytetään, vaihtelee suuresti. Monissa lähteissä symbolinen konekieli luokitellaan alhaisen tason kieleksi, kun taas joissakin lähteissä C-ohjelmointikieltä kutsutaan alhaisen tason kieleksi ja sijoitetaan konekielten yläpuolelle. Joskus, mukaan lukien taulukko 6.6, käytetään nimitystä keskitason kieli, johon C-kieli ja sen johdannaiset sijoitetaan, konekielten ja (varsinaisten) korkean tason kielten väliin.[279] Joskus C-kieli lasketaan korkean tason kieleksi.

[279] Esimerkiksi: http://fresh2refresh.com/cprogramming/c-language-history/

Taulukko 6.6. Ohjelmointikielen tasot (eräs luokittelu).[280]

Taso	Esimerkkejä
Ohjelmistokehykset (software framework)	Ajax framework
Täsmäkielet (domain-specific languages)	Mathematica, SQL
Korkean tason kielet (high level languages)	Python, Perl, Ruby
Keskitason kielet (middle-level languages)	C, C++
Symbolinen konekieli (assembly language)	Assembler
Konekieli (machine language, machine code)	

Seuraavassa keskeisimpien tasojen määritelmät:[281]

- Täsmäkieli on ohjelmointikieli, joka ei ole yleiskäyttöinen vaan sopii käytettäväksi nimenomaisen aihealueen yhteydessä.
- Korkean tason ohjelmointikieli on kieli, jonka abstraktiotaso on korkea ja joka ei riipu ohjelmaa suorittavan tietokoneen erityispiirteistä.
- Konekieli on ohjelmointikieli, joka sopii tietynlaisten prosessorien suoritettavaksi, mutta jota ihminen ei yleensä itse lue tai kirjoita.

Merkittävimmät erot alhaisempien ja korkeampien tason kielten välillä liittyvät muistin hallintaan ja koodin luonnollisuuteen kielenä. Korkean tason kielissä muistin hallinta on automaattista ja turvallista, kun taas alhaisemman tason kielissä koneen muistia voidaan hallita suoraan, jolloin sen käyttöön liittyy huomattavia riskejä. Konekieltä käytetään erityisesti silloin, kun koodin tehokkuus on tärkeää. Esimerkkinä tästä lähes neljän vuosikymmenen takaa on Nokian valinta koodata ensimmäisen digitaalisen puhelinkeskuksen käyttöjärjestelmä assemblerilla sen sijaan, että olisi käytetty korkeamman tason ohjelmointikieltä.[282] Tämä ratkaisu (joka ei ollut lainkaan itsestään selvä) oli merkittävä, koska sillä tavoin saatiin enemmän suorituskykyä irti samoista suorittimista kuin korkeamman tason ohjelmointikieliä käyttäneet kilpailijat. Mitä korkeammalle tasolle kielissä mennään, sitä enemmän koodi muistuttaa luonnollista kieltä.

Edellä kuvatut asiat voidaan tiivistää alhaalta ylöspäin seuraavasti:

1. Tietokone rakentuu transistorien ja muiden peruskomponenttien avulla toteutetuista loogisista piireistä. Loogiset piirit suorittavat operaatioita biteillä, joiden avulla kuvataan kaikki mahdollinen, sekä data että suoritettavat toimenpiteet eli käskyt. Kun tietokone toimii virheettömästi, niin samoilla syötteillä ja samalla

[280] Mitään yksiselitteistä totuutta tasoista ei ole, varsinkaan välitasoilla, katso esimerkiksi http://www.cse.hut.fi/fi/opinnot/CSE-A1121/2015/yleista/sanasto.html#term-ohjelma, http://www.codecommit.com/blog/java/defining-high-mid-and-low-level-languages

[281] Ohjelmoinnin peruskurssi Y2, Sanasto, http://www.cse.hut.fi/fi/opinnot/CSE-A1121/2015/yleista/sanasto.html

[282] M. Sandelin, J. Partanen (2015), *Nokian jalokivi Tarina suomalaisesta DX 200 puhelinkeskuksesta*, s. 89. Päätös tehtiin silloisessa Telefenno-nimisessä yrityksessä, joka syntyi kun Nokia ja valtion omistaman Televan puhelinliiketoiminta yhdistettiin vuonna 1977. Televa myytiin Nokialle ja samalla Telefenno fuusioitiin Telenokiaan kesäkuussa 1981.

ohjelmalla lopputulos on aina täsmälleen sama. Jos toimintaan halutaan satunnaisuutta, se voidaan tehdä vain syöttämällä järjestelmään satunnaistettua dataa.

2. Koska mikropiirit käsittelevät bittejä, ohjelma täytyy syöttää prosessorille bitteinä. Konekielinen käsky voi esimerkiksi määrätä, että rekisterin 1 ja rekisterin 2 sisältämät luvut lasketaan yhteen ja sijoitetaan rekisteriin 5. Operaatio (tässä yhteenlasku) määritetään tietyn paikan määrätyllä bittikuviolla ja vastaavasti käytettävät rekisterit määritellään omilla paikoillaan koodissa.

3. Jos halutaan koodata suoraan konekielisenä koodina, binääriset koodit voidaan esittää helpommin ymmärrettävällä symbolisella konekielellä. Symbolisessa konekielessä käskyt, esimerkiksi lukujen vertailu tai yhteenlasku, esitetään lyhyellä kirjainkoodilla, samoin rekistereille on omat ymmärrettävät nimensä. Konekieltä käytetään yleensä sellaisten ohjelmapätkien koodauksen, jotka pyritään toteuttamaan mahdollisimman tehokkaasti.

4. Yleensä ohjelmointi tapahtuu korkeamman tason ohjelmointikielellä, joka muunnetaan konekieliseen muotoon joko tulkin (interpreter) tai kääntäjän (compiler) avulla. Tulkki käsittelee yhtä käskyä kerrallaan, kun taas kääntäjä käsittelee kokonaista ohjelmaa. Korkean tason ohjelmointikieli käsittelee erilaisia objekteja, kuten numeroita, sanoja tai kuvia. Ohjelma on siis täsmällinen kuvaus siitä, miten suorittimen tulee käsitellä sille annettuja objekteja.

Ohjelmointi työurana tai uran osana

Kuten edellä todettiin, jokaisen informaatioteknologian alalla toimivan täytyy hallita ohjelmoinnin perusteet. Entä jos tavoitteena ei ole varsinaisesti ohjelmointi eikä edes tekniikan kehittäminen, vaan pikemminkin yritystoiminta ja vaurastuminen sen avulla? Monet viime vuosikymmenien menestyneimmistä yrityksistä ovat syntyneet ohjelmistoalalle, kuten Microsoft, Apple, Google ja Facebook. Onko näiden yritysten menestyksen salaisuus sitten ylivertainen ohjelmointitaito? Useimmat avainhenkilöt ovat olleet ohjelmoijia. Toisaalta he ovat keskittyneet yrityksen kehittämiseen kaiken muun kustannuksella. Yliopiston näkökulmasta näitä henkilöitä voidaan pitää pudokkaina, mutta J. Kemppisen sanoin: [283]

> Bill Gatesista ja Steven Jobsista ei tullut professoreita Berkeleyhin. Mitä putoamista sellainen on? Molemmat omaksuivat suhteessa yleisöön 1800-luvun lopun toimintaperiaatteet eli kurkunleikkauskapitalismin ja monopolien rakentamisen.

[283] Kemppinen, J., Sotamiehellä on tikapuuhermosto, http://kemppinen.blogspot.fi/, 1.1.2015.

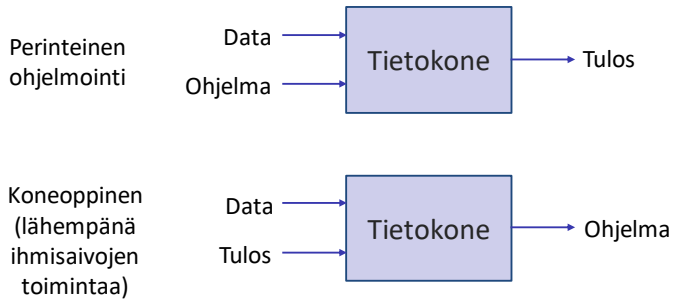
Tämä on provosoivasti kirjoitettu, mutta kun Gates, Jobs ja Zuckerberg ovat usein käytettyjä esimerkkejä siitä, ettei yliopisto-opintoja tarvitse suorittaa loppuun tullakseen rikkaaksi, niin samalla on hyvä miettiä, mitkä ominaisuudet tekivät heistä lopulta superrikkaita. Tuskin ylivertainen ohjelmointitaito, vaan pikemminkin kokonaisnäkemys siitä mihin maailma oli menossa sekä kyky hyödyntää kohdalle osuneet tilaisuudet ja kova työ.

Douglas Comerin mukaan sovellusten ohjelmoija, joka ymmärtää tietoliikenneverkon toiminnan, pystyy kirjoittamaan sovelluksia, jotka ovat luotettavampia ja tehokkaampia kuin ilman näkemystä tehdyt sovellukset.[284] Järjestelmän kehittäjien liiallinen erikoistuminen ja toiminnan kapea-alaisuus on usein vaarallinen yhdistelmä lopputuloksen kannalta, varsinkin jos kyseessä ovat monimutkaiset kokonaisuudet, kuten tietoverkot.

Vaihtoehtoinen malli: koneoppiminen ja hahmontunnistus

Menemättä syvemmälle monimutkaiseen aihepiiriin on syytä muistuttaa, että tässä oppimateriaalissa pääosin tarkasteltu perinteiden tietojenkäsittely pohjautuu rakenteeseen, joka on hyvin toimiva, mutta ei ainoa mahdollinen. Kuvassa 6.14 on esitetty karkealla tasolla perinteisen ohjelmoinnin ja koneoppimisen (machine learning) välinen ero (voidaan myös käyttää termiä tekoäly, artificial intelligence)[285]. Perinteisessä ohjelmoinnissa monimutkaisuus on koodissa, jonka ihmiset kirjoittavat. Koneoppimisessa algoritmit ovat suhteellisen yksinkertaisia ja monimutkaisuus on datassa. Koneoppimisessa on olennaista, että tietokone päättelee automaattisesti datan rakenteen itse datan perusteella ilman, että ihmisen tarvitsee määritellä rakennetta.

Toisin sanoen perinteisessä ohjelmoinnissa data ja ohjelma määrittelevät tuloksen, kun taas koneoppimisessa data ja tulos yhdessä johtavat ”ohjelmaan.” Data voi esimerkiksi olla suuri joukko kuvia, joista kone etsii automaattisesti säännönmukaisia hahmoja. Tulos voi olla henkilö, jolloin koneoppimisen avulla tunnistetaan henkilö kasvokuvien avulla.



Kuva 6.14. Perinteisen ohjelmoinnin ja koneoppimisen välinen periaatteellinen ero.

[284] Comer, D. E. (2008). *Computer Networks and Internets*. Prentice Hall Press, sivu 3.
[285] Erään vitsin mukaan koneoppiminen tehdään Python-koodilla kun taas tekoäly tehdään PowerPointilla.

Koneoppimisen periaate muistuttaa aivojen toimintaperiaatetta. Aivoissa tietojenkäsittelyn perusmoduuli toteuttaa yksinkertaisen hahmontunnistuksen algoritmin. Moduuli sisältää suunnilleen 100 hermosolua sekä moduulien väliset hermoyhteydet ja hermoyhteydet muihin osiin aivoja.[286] Ihmisen aivojen tietojenkäsittelyn vahvuudet ovat:

1. Tasojen määrässä: yhden moduulin tuloksia käytetään ”ylemmän” tason moduulin sisääntulossa. Ihmisellä moduulit ovat järjestyneet pääosin kuuteen tasoon.
2. Takaisinkytkennöissä: ylemmän abstraktiotason moduulien tuloksia käytetään alemman tason moduulien sisääntuloissa.
3. Äärimmäisessä rinnakkaisuudessa joka mahdollistaa miljoonien rinnakkaisten hahmontunnistusketjujen läpikäymisen.

Hyvin korkean tason hahmontunnistus, esimerkiksi ironian havaitseminen puheesta, voi vaatia kymmenien hahmontunnistusvaiheiden läpikäymisen—ei siis ole ihme, että edes kaikki ihmiset eivät ymmärrä ironiaa.

Miten tämä sitten liittyy informaatioteknologiaan? Ihmisen aivojen toiminnan ymmärryksen pohjalle on jo nyt rakennettu merkittävää teknologiaa, esimerkiksi puheen- ja kasvojentunnistuksen aloilla. Samoja periaatteita voidaan soveltaa myös verkkoihin, esimerkiksi ruuhka- tai vikatilanteen nopeaan havaitsemiseen tai jopa niiden ennakointiin.[287] Jos noudatetaan aivojen toimintaperiaatetta, koneoppiminen voisi perustua yhteen yleiskäyttöiseen hahmontunnistuksen yksikköön, jota ei tarvitse sovittaa kunkin ongelman erityisvaatimuksiin. Optimaalista kuitenkin lienee jossain määrin sopeuttaa hahmontunnistuksen moduulia siihen, minkälaisia hahmoja on tarkoitus tunnistaa. Puhetta voidaan tunnistaa samoilla algoritmeilla kuin kuvia, mutta lopputulos ei ole yhtä tehokas kuin kuvantunnistusta varten kehitetyt algoritmit. Samaa sääntö pätenee koneoppimiseenkin.

Mahdollisia kehityskulkuja

On hämmästyttävää miten vähäisen tiedonkäsittelykapasiteetin turvin lennettiin ensin avaruuteen maan kiertoradalle ja sitten kuuhun ja takaisin. Dramaattinen muutos tietojenkäsittelyssä on vaikuttanut lähes kaikkiin elämän osa-alueisiin, ei vain varsinaiseen informaatioteknologiaan. Miten tästä eteenpäin? Useimmat ennusteet menevät täysin pieleen, silti joillakin on ollut taito nähdä hämmästyttävän tarkkaan tulevaa kehitystä. Yksi parhaista esimerkeistä on Isaac Asimovin vuonna 1964 tekemä ennuste siitä, miltä maailma

[286] Aivojen toiminnan periaatteista helposti luettavia kirjoja ovat mm.: Kurzweil, R. (2012): *How to create a mind: The secret of human thought revealed*, Penguin, ja Hawkins, J. & Blakeslee, S. (2007): *On Intelligence*, Macmillan.
[287] Katso esimerkiksi https://www.ietf.org/proceedings/92/slides/slides-92-sdnrg-0.pdf

näyttää vuonna 2014.[288] Vaikka kaikki hänen ennusteensa eivät ole toteutuneet, ennusteet koskien esimerkiksi litteitä näyttöjä ja viestintää ovat oikeaan osuneita:

> "Communications will become sight-sound and you will see as well as hear the person you telephone. ... The screen can be used not only to see the people you call but also for studying documents and photographs and reading passages from books."

Yhtä hyvää arviota vuodelle 2069 en voi luvata, mutta yksinkertainen laskelma voi antaa suuntaa tulevalle kehitykselle. Laskelma on sinänsä varsin karkea, mutta olennaista on suuruusluokat ja tietyn rajan ylittyminen. Miten nykyisten prosessorien tietojenkäsittelykyky ja hinta suhtautuvat ihmisaivojen kapasiteettiin ja sen hintaan? Aivoissa on noin 100 miljardia hermosolua ja hermosolu voi olla yhteydessä muihin hermosoluihin keskimäärin noin 10000 synapsin kautta. Vaikka hermosolujen ”kellotaajuus” on ehkä vain luokkaa 1000 Hz, niin rinnakkaisuus tekee aivoista hyvin tehokkaan. Eräs arvio aivojen tietojenkäsittelykyvystä on päätynyt lukuun 100 miljoonaa MIPS:iä.[289] Tämä ei ole pieni luku, mutta ei kovin suurikaan luku verrattuna nykyisten tietokoneiden laskentatehoon.

Tämä on sangen dramaattinen tulos. Ovatko tietokoneet jo nyt älykkäämpiä kuin ihmiset? Asia on paljon monimutkaisempi, kun otetaan huomioon se, miten aivot hyödyntävät tietojenkäsittelykykyään eli millainen ”ohjelmisto” aivoissa on. On asioita joita tietokoneet ei edelleenkään osaa tehdä yhtä hyvin kuin ihmiset. Vapaamuotoisen ja –sisältöisen puheen kääntäminen kieleltä toiselle on edelleen vaikea tehtävä mille tahansa koneelle tai ohjelmistolle. Toisaalta monilla alueilla, kuten kasvojen tunnistuksessa ja luonnollisen puheen tuottamisessa, edistys on ollut huimaa viimeisten vuosien aikana.

Tietojenkäsittelyn hinta on painunut niin alhaiseksi, että kaikki ne tietojenkäsittelyn tehtävät, joihin pystytään kehittämään suhteellisen tehokkaat menetelmät ja algoritmit, siirtyvät väistämättä koneiden tehtäväksi. Suhteellinen tarkoittaa tässä tehokkuutta suhteessa aivojen käyttämiin menetelmiin. Evoluutio on kehittänyt aivoille elämän kannalta keskeisiin tehtäviin hyvin tehokkaat menetelmät, mutta tästä eteenpäin automaattisten laitteiden käyttämien menetelmien ei tarvitse edes olla yhtä tehokkaita kuin ihmisaivojen menetelmien. Suurten numeroiden kertolasku ja shakki eivät kuitenkaan ole ihmisen elämän kannalta kriittisiä taitoja, sen sijaan kasvojen ja puheen tunnistus ovat. Lopulta siinä vaiheessa, kun tietokoneet pystyvät itse kehittämään omat tietojenkäsittelyn menetelmänsä, seuraukset ihmiselämälle voivat olla järisyttäviä. Tässä suhteessa Googlen AlphaZero saattaa ennakoida merkittävää käännettä koneoppimisen saralla.[290]

---

[288] Katso esimerkiksi http://www.huffingtonpost.com/2014/01/02/isaac-asimov-2014_n_4530785.html

[289] H. Moravec (1998), "When will computer hardware match the human brain." *Journal of evolution and technology*, MIPS = Million Instructions Per Second.

[290] Lukekaa: http://continuations.com/post/168426788670/alphazero-chess-computers-will-think-differently.