LUT
University

LUT University

CT30A3370 - KÄYTTÖJÄRJESTELMÄT JA SYSTEEMIOHJELMOINTI 6 OP

Jussi Kasurinen (etu.suku@lut.fi)

JOHDANTO, KÄYTTÖJÄRJESTELMÄ JA TIETOKONEARKKITEHTUURI

CT30A3370 - Käyttöjärjestelmät ja systeemiohjelmointi

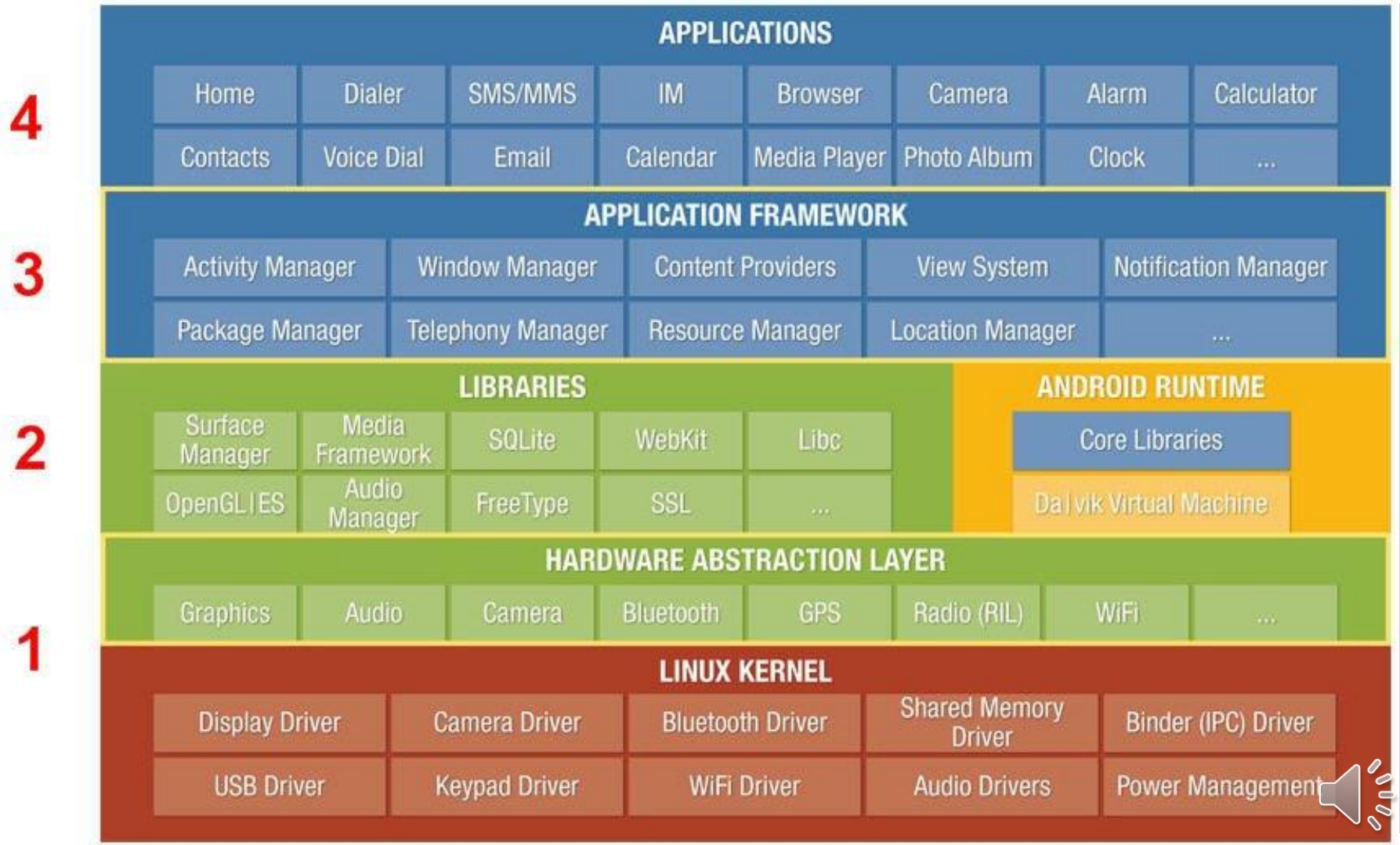
TIETOTEKNIIKASTA

- Entisaikaan tietokoneet ei olleet niin ubiikkeja kuin nykyään.
  - Ubiikki = yleinen, jatkuvasti vastaan tuleva, esim. autot moottoritiellä
- Tietokoneiden kanssa työskentelevät ihmiset tunsivat hyvin tietokoneen toimintaperiaatteet.

TIETOTEKNIIKASTA

- Modernit tietokonelaitteet ovat vuosien saatossa kuitenkin tulleet niin monimutkaisiksi, että niiden toiminta ei ole loppukäyttäjille selvillä
  - Tietotekniikan perusajatukset ovat nykyään piilossa monen abstraktiokerroksen alla.
  - Käyttäjäkunnat eivät myöskään enää ole ”ammattilaisia”
  - Meillä on monta kerrosta erilaisia rajapintoja jotka piilottavat sovelluskohtaisia ohjelmatoteutuksia alleen.
- Monimutkaisuudesta, ja monitasoisuudesta, on seurannut tämän tieteenalan **erikoistuminen**:
  - Yksi erikoisala kattaa vain pienen osan tietotekniikkaa
    - Tietokonetekniikka, tietojenkäsittelytiede, tietoliikenne, ohjelmistotuotanto…

VRT. ANDROID STACK (GITHUB DEVACADEMY)



TIETOTEKNIIKASTA

- Miksi tämä on tärkeää?
  - Nykyään on havaittavissa, kuinka moni tite-opiskelija ei oikein oikeasti tunne tietokoneen toimintaa
  - Ei nähdä metsää puilta!
  - Tyypillisesti opiskellaan ohjelmointia useamman kurssin verran, eri kieliä ja paradigmoja opetellen, sekä tietojenkäsittelyä (tietojenkäsittelyn perusteet, tietorakenteet ja algoritmit ym.) ilman, että missään kohtaa pohditaan, miten ne asiat sopii kokonaiskuvaan.
  - Voidaanko puhua tietotekniikasta ollenkaan, jos ei ymmärretä, miten tietotekniikka oikeasti toimii?

TIETOTEKNIIKASTA

- Tällä kurssilla pääpaino on käyttöjärjestelmän toiminnassa
  - Tarkastellaan ensin vähän tietokonearkkitehtuurin ja ohjelmistoarkkitehtuurin erityispiirteitä.
- Erityinen ohjelmisto siinä mielessä, että se sitoo laitteiston ja ohjelmiston toisiinsa!
  - Tästä johtuen sivutaan sekä ohjelmistoon että laitteistoon liittyviä asioita
  - Kokonaisuutena tietokonejärjestelmä, jossa käyttis on yksi tärkeimmistä komponenteista ellei tärkein, sisältää aika monta liikkuvaa osaa joiden toimintaa tarkastellaan myös.

KORKEAN TASON OHJELMOINTI

- Mietitään vaikkapa yksinkertaista, pientä **Hello world!** -ohjelmaa esimerkkinä.
- Riippuen ohjelmointikielestä saattaa näyttää erilaiselta
  - Esim. oliopohjaiset kielet vs. proseduraaliset / funktionaaliset

MIKÄ ON *TIETOKONEOHJELMA*

- Ohjelma = käskysarja, joka kertoo, miten jokin tehtävä suoritetaan
  - Voi olla matemaattinen tehtävä, esim. funktion (tai yhtälöryhmän) ratkaisu..
  - Tai symbolista laskentaa, esim. tekstistä etsiminen ja korvaaminen..
  - tai, kummallista kyllä, (toisen) ohjelman kääntäminen
- Eri ohjelmointikielet näyttävät omanlaisiltaan (=erilaisilta), mutta niissä yleensä on pieni joukko samankaltaisia käskyjä:
  - Syöttö: luetaan dataa syöttölaitteelta, esim. näppäimistöltä tai tiedostosta
  - Tulostus: Syötetään dataa tulostuslaitteelle, esim. näytölle tai tiedostoon.
  - Matemaattiset operaatiot: Suoritetaan yksinkertaista aritmetiikkaa
  - Ehdollinen suoritus: Verrataan totuusarvoja ja muutetaan ohjelman suoritusta niiden mukaan
  - Toisto: Toistetaan jotain/joitain käskyjä, joko samanlaisina tai “pienin muutoksin” joka kierroksella

MIKÄ ON *TIETOKONEOHJELMA*

- **Hello world -ohjelma on harhaanjohtavan yksinkertainen**
- Oletteko tulleet koskaan miettineeksi, miten kirjoittamamme tietokoneohjelma oikeastaan ajetaan?
- Ohjelma on vain tekstiä - eli kirjaimia tekstitiedostossa. Kirjaimet sinänsä ovat elottomia, eli ne eivät tee yhtään mitään.

OHJELMOINTIKIELEN KÄÄNTÄMINEN

- Teksti pitää ensin parsia eli ymmärtää sen semantiikka
  - Semantiikka: Mitä tekstin merkitys on? Mitä ohjelmoija haluaa tietokoneen tekevän?
- Sen jälkeen ohjelmakoodi pitää kääntää tietokoneen ymmärtämään muotoon - konekieleksi.
  - Tämä prosessi on tietysti nimeltään kääntäminen
- Toki konekielikin on abstraktio
  - Sovittu joukko binäärikoodeja, jotka ohjaavat prosessorin toimintaa.

KONEKIELI JA LAITTEISTOARKKITEHTUURI

- Jotta konekielisen ohjelman voi ajaa, jonkin prosessoriarkkitehtuurin on oikeasti toteutettava tuo sovittu käskykanta.
- Prosessoriarkkitehtuuri itsessään on vain spesifikaatio (paperi, dokumentti) - Jonkin piirisarjan on oikeasti toteutettava se prosessoriarkkitehtuuri.
  - Eli miten rekisterit, muistiyksiköt, väylät ym. rakentuvat ja toimivat.

KONEKIELI JA LAITTEISTOARKKITEHTUURI

- Jokainen piirisarjan elementti rakennetaan alkeissiruista.
- Alkeissirut koostuvat loogisista porteista.
- Loogiset portit koostuvat erilaisista kytkinlaitteista (transistorit)
- Transistorit ovat… Tässä kohtaa listausta tietotekniikka loppuu ja sähkötekniikka alkaa.
  - Tämä aihe jatkuu elektroniikan perusteiden ja digitaalielektroniikan puolella (ihan mielenkiintoisia kursseja, btw)

ABSTRAKTIOT / ABSTRAHOINTI

- Miten monimutkaisia tietokonejärjestelmiä rakennetaan tyhjästä?
  - Miten ensimmäinen tietokone on rakennettu?
    - Nykyäänhän käytämme tietokoneita tietokoneiden suunnitteluun, esim. ohjelmointikieli rakennetaan ohjelmoimalla, piirisarjat suunnitellaan hdl:llä…

ABSTRAKTIOT / ABSTRAHOINTI

- Loppupelissä koko laite (laitteisto ja ohjelmisto) koostuu vain alkeispalasista, jotka tässä tapauksessa ovat loogisia portteja
  - Miten näin kompleksista sätöstä voi mitenkään hallita?
  - Mieti esim. isoa ohjelmaa, jossa ei ole yhtään funktiota / aliohjelmaa, jossain kohtaa (missä, 10 riviä, 50 riviä, 100 riviä?) homma alkaa käydä hallitsemattomaksi.
    - Abstraktioon / abstrahointiin törmää ohjelmoidessa äkkiä: Jäsennetään koodia “luettavammaksi” aliohjelmien (funktioiden, metodien) avulla

ABSTRAKTIO

- Ihmisen erottaa koneesta kyky abstraktoida asioita!
  - Abstraktio = “Mitä tekee, ei miten”
  - Rakennetaan alemmalla tasolla jokin palikka joka tekee jotain. Sen jälkeen sitä voidaan käyttää tekemään “jotain” loputtomiin ilman, että meidän tarvitsee murehtia palikan lopullisesta toteutuksesta
    - Toki vain niin kauan, kuin palikka toimii.
  - Esim. kun ohjelmointikielessä sanomme *print “nakki”* luotamme siihen, että tulostuskäsky tosiaan tulostaa ruudulle jotain, koska näin on ohjelmoinnin perusteissa opetettu. Meillä ei ole mitään käsitystä, miten komento sen tekee, eikä tarvitsekaan olla.
- +Mooren laki→ tehokkaampi kone, kompleksisempia rakennelmia

ABSTRAKTIO

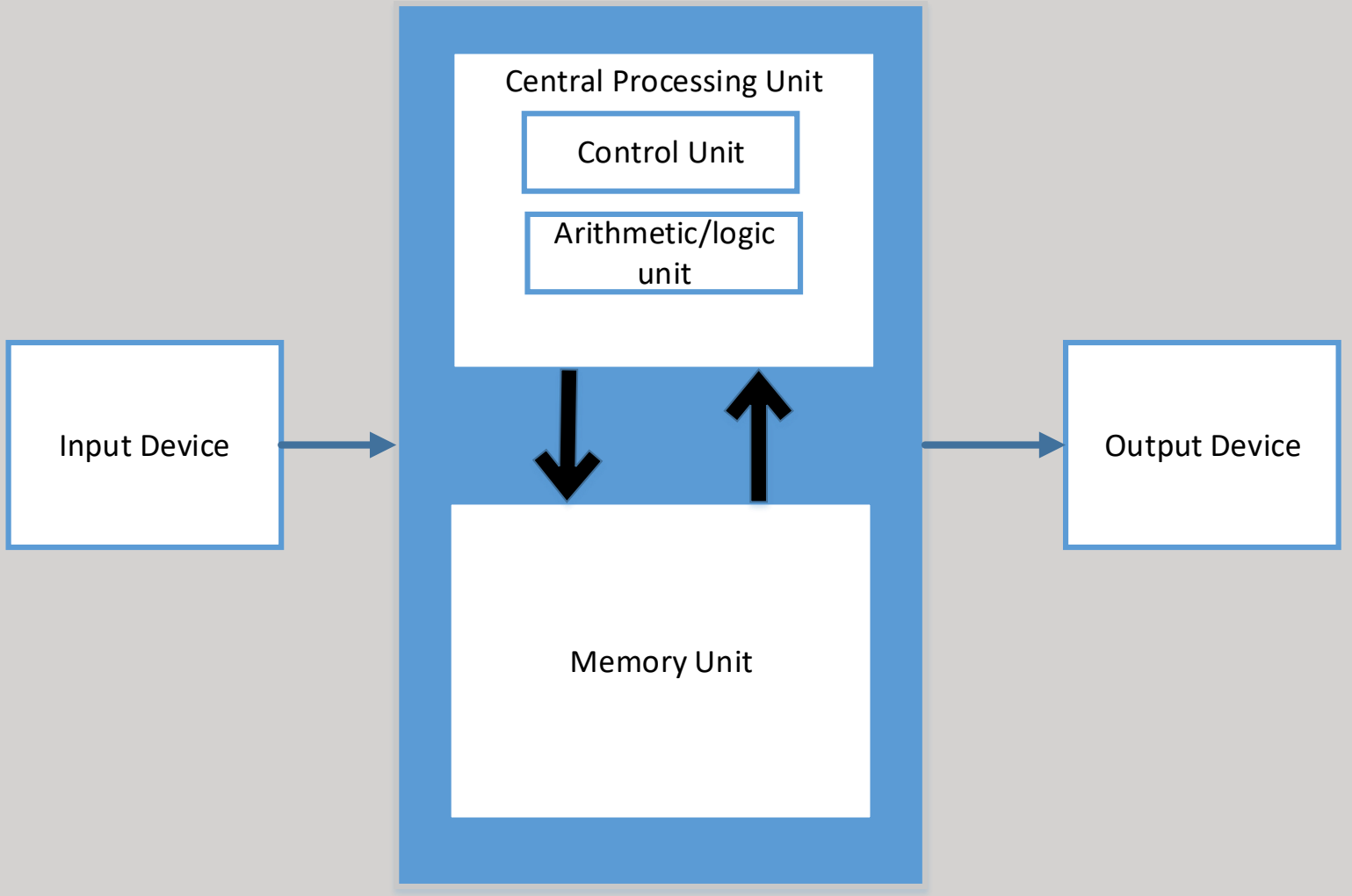
- Abstraktiolla on joku rajapinta, minkä kautta sen toimintaan pääsee käsiksi, ilman että sen sisäistä toiminnallisuutta tarvitsee ymmärtää.
  - Jokainen softa- tai hardisohjelmoija käyttää rajapintoja, rakentaa niitä tai pyytää jotakuta muuta rakentamaan ne.
- Tietokonearkkitehtuuri, eli iso kokoelma eri tasoisia abstraktioita tai rajapintoja voidaan kuvata joko
  - Ylhäältä alas, jolloin nähdään, miten korkean tason abstraktiot suppenevat tai uudelleenkirjoitetaan yksinkertaisempien komentojen avulla.
  - Alhaalta ylös, jolloin nähdään, miten yksinkertaisista komponenteista voidaan rakentaa monimutkaisia laitteita.

HETKI HISTORIAA

- Alan Turingin tietokoneen perusajatus 1936
- Conrad Zusen Plankalkül – siirrettävän ohjelmointikielen konsepti – vuonna 1943
- Von Neumann tietokonearkkitehtuuri vuonna 1945.
- Fortran, 1. siirrettävä ohjelmointikieli, 1957.

  → Tietokone ei ole enää pääosin yhteen käyttöön suunniteltu, joten sen suorittamaa tehtävää pitäisi pystyä vaihtamaan, ohjaamaan ja valvomaan.

VON NEUMANN -ARKKITEHTUURI



TIETOKONEARKKITEHTUURI (KERTAUSTA TIETOJEKNKÄSITTELYN PERUSTEISTA)

- Boolen algebra
  - And, or, not, not and
  - Kaikki tietojenkäsittely perustuu yhteen operaatioon, joka on nand
  - Kaikki muut Loogiset funktiot voidaan rakentaa nandin avulla
- Boolen aritmetiikka
  - == miten loogisilla operaatioilla suoritetaan laskutoimituksia.
  - 2-kantainen lukujärjestelmä
    - Looginen operaatio antaa true/false, 2-kantaisessa järjestelmässä tämä riittää, kun true = 1 ja false = 0
- Sekventiaalilogiikka
  - Boolen algebra ja aritmetiikka on kombinatorista
  - Perustuu siis pelkästään eri Boolen funktioiden yhdistämiseen
  - Jos halutaan rakentaa muistilaitteita, tarvitaan myös **peräkkäisiä** operaatioita
    - Funktio, jonka arvo on yhdellä ajanhetkellä jotain ja toisella jotain muuta...

TIETOKONEARKKITEHTUURI

- Laitteisto
  - Loogisia piirejä rakentelemalla voidaan toteuttaa laskureita ja muistipiirejä (rekisterit ja käyttömuisti)
  - esim. n-bittinen rekisteri == n-kpl vierekkäisiä kiikkuja
- Arkkitehtuuri
  - Piirisarjaa ohjataan ohjaussignaaleilla
  - Konekieli = ohjaussignaalien muodostama kokonaisuus
  - assembly on symbolinen konekieli, ts. käännetään “luonnollista kieltä muistuttavaa” ohjelmointikieltä ohjaussignaaleiksi (eli konekieleksi)

TIETOKONEARKKITEHTUURI

- Käyttöjärjestelmät
  - Käyttöjärjestelmä jakaa varus- ja sovellusohjelmille muistia ja suoritinaikaa
  - Huolehtii myös kaikesta syöttö- ja tulostuslaitteiden ohjauksesta, (eli ohjelmoijan ei tarvitse huolehtia, miten IO-operaatiot hoituvat)
  - Järjestelmäkirjastot toteuttavat yhteisiä, matalan tason laskutoimituksia, mitä prosessori ei pysty /osaa tehdä (voi olla esim. ei-primitiiviset matikkaoperaatiot, neliöjuuret tms)
- Ohjelmointikielet
- Kääntäjät ja tulkit
- Tietorakenteet ja algoritmit
  - Aritmeettiset alg.
    - ns. “toistoalgoritmit”, summa, tulo jne.
  - Geometriset alg.
    - Geometriaan perustuvat laskut, esim. ympyröiden tai viivojen piirto…

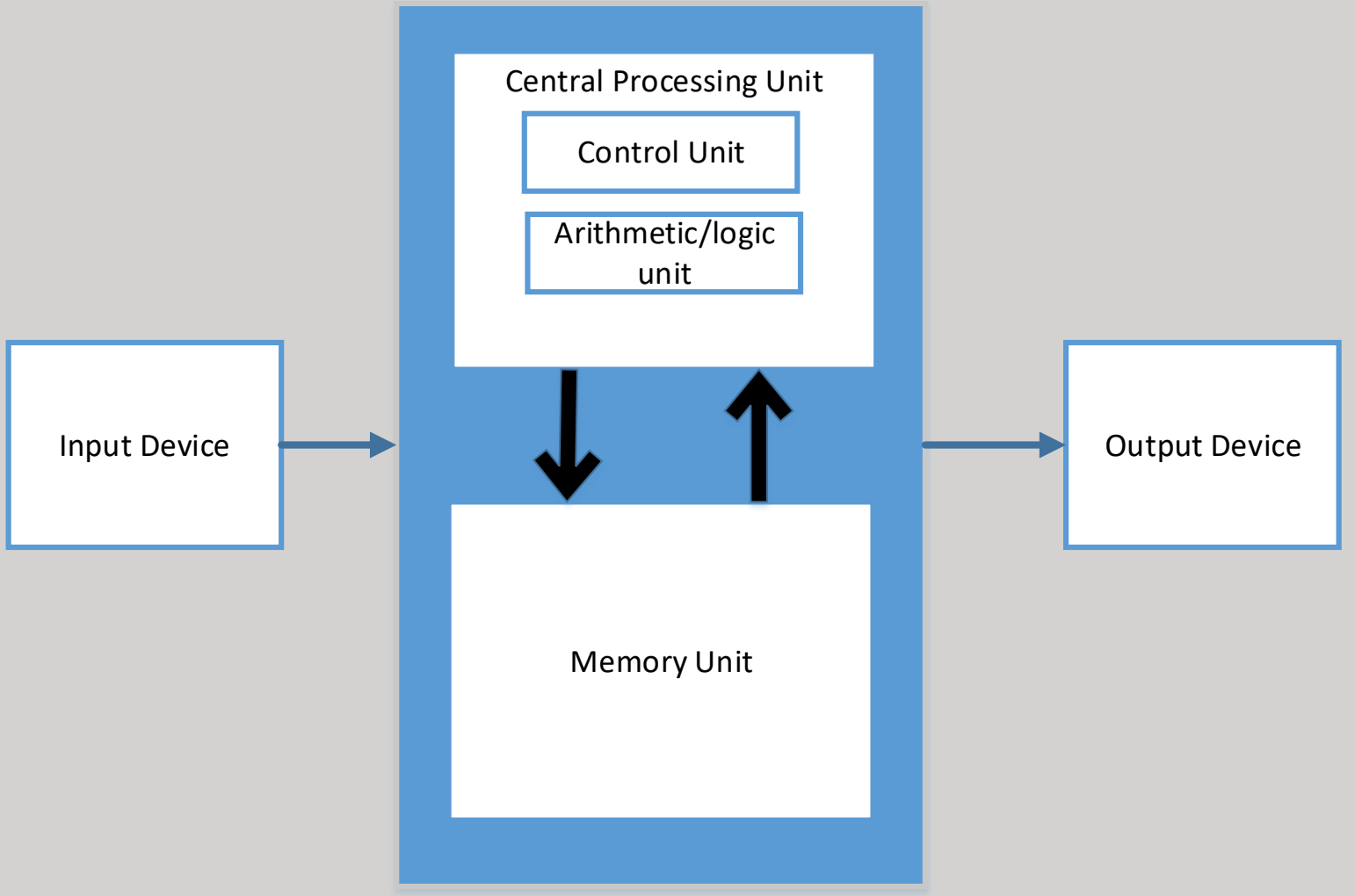
TIETOKONEARKKITEHTUURIN ABSTRAKTIOTASOSTA

- Korkein abstraktiotaso on ohjelmointi: Ohjelmoija suunnittelee ohjelmistoja ja kirjoittaa ohjelmia ja moduuleja, joista ne koostuvat.
  - Ohjelmointikielissäkin on abstraktioeroja mutta tämän kurssin sisältöä ajatellen kaikki on ”korkean tason kieliä”. Myös C.
- Työssään ohjelmoija luotta kahden oleellisen työkalun voimaan:
  - Käytössä oleva ohjelmointikieli
  - Käytössä olevat kirjastot, jotka toteuttavat monipuolisia palveluita ohjelmointikielen tueksi.

TIETOKONEARKKITEHTUURIN ABSTRAKTIOTASOSTA

- Ennen kuin ohjelman voi edes ajaa, se pitää kääntää, tai ajonaikaisesti tulkata
  - Syntaksin analysointi ja kohdekoodin generointi
  - Lähdekoodi pitää analysoida ja ryhmitellä kielioppirakenteisiin - ryhmittelystä syntyy **jäsennyspuu**
  - Jäsennyspuuta aletaan käydä järjestelmällisesti läpi, ja muodostetaan sen avulla **välikielinen** ohjelma
    - Esim. Javassa ja C#:ssa välikieli kuvaa käskypinoa virtuaalikoneessa.
    - Virtuaalikone toteuttaa käskykannan “oikeassa” tietokoneessa.
    - Jos / Kun virtuaalikoneen välikielestä tulkkaama ohjelma on assembly-koodia (=symbolista konekieltä) se pitää vielä tulkata binäärikoodiksi (=varsinainen konekieli, binääriset ohjauskoodit). Tämän tekee assembler-tulkki.

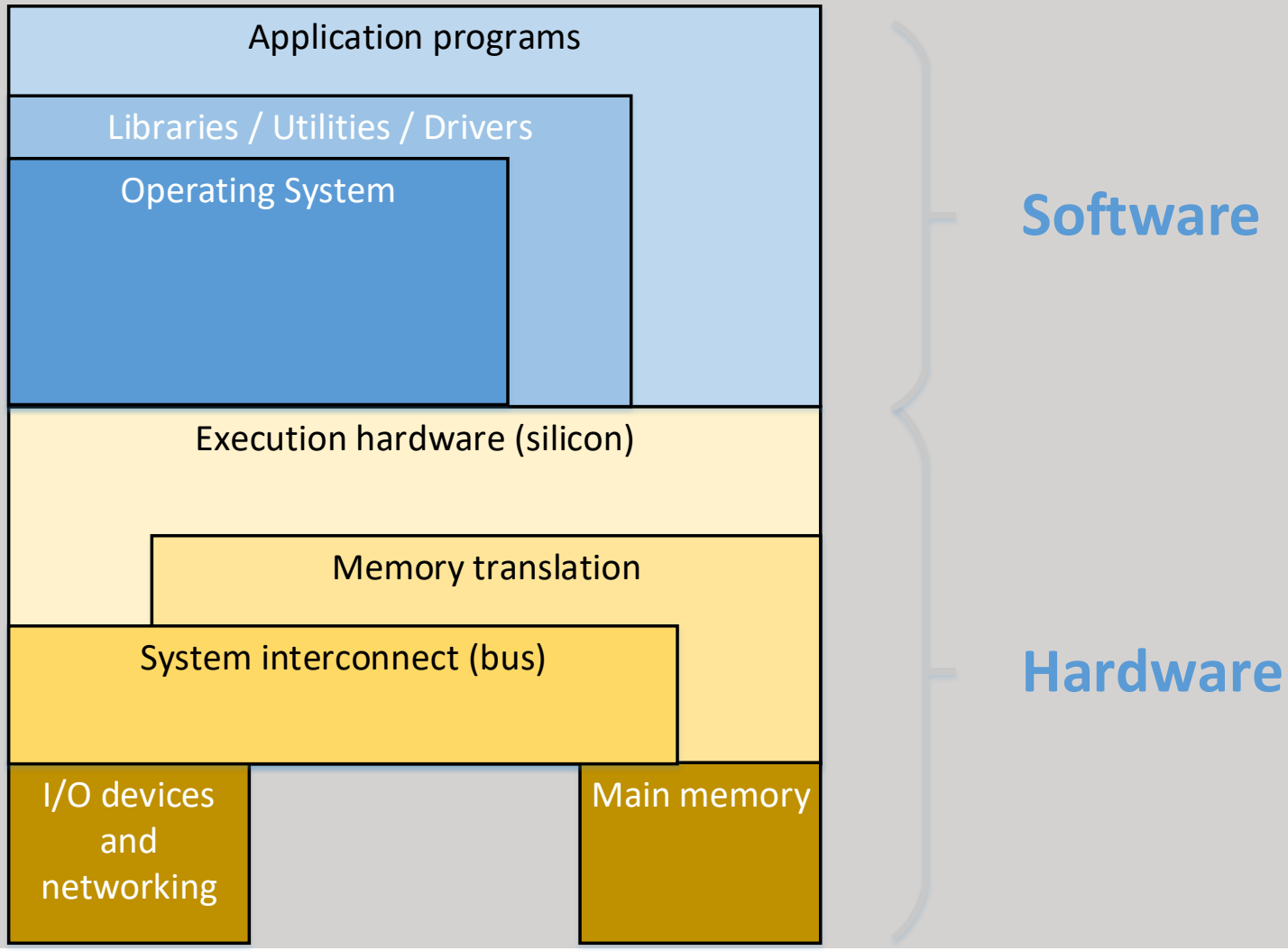
RE: VON NEUMANN -ARKKITEHTUURI



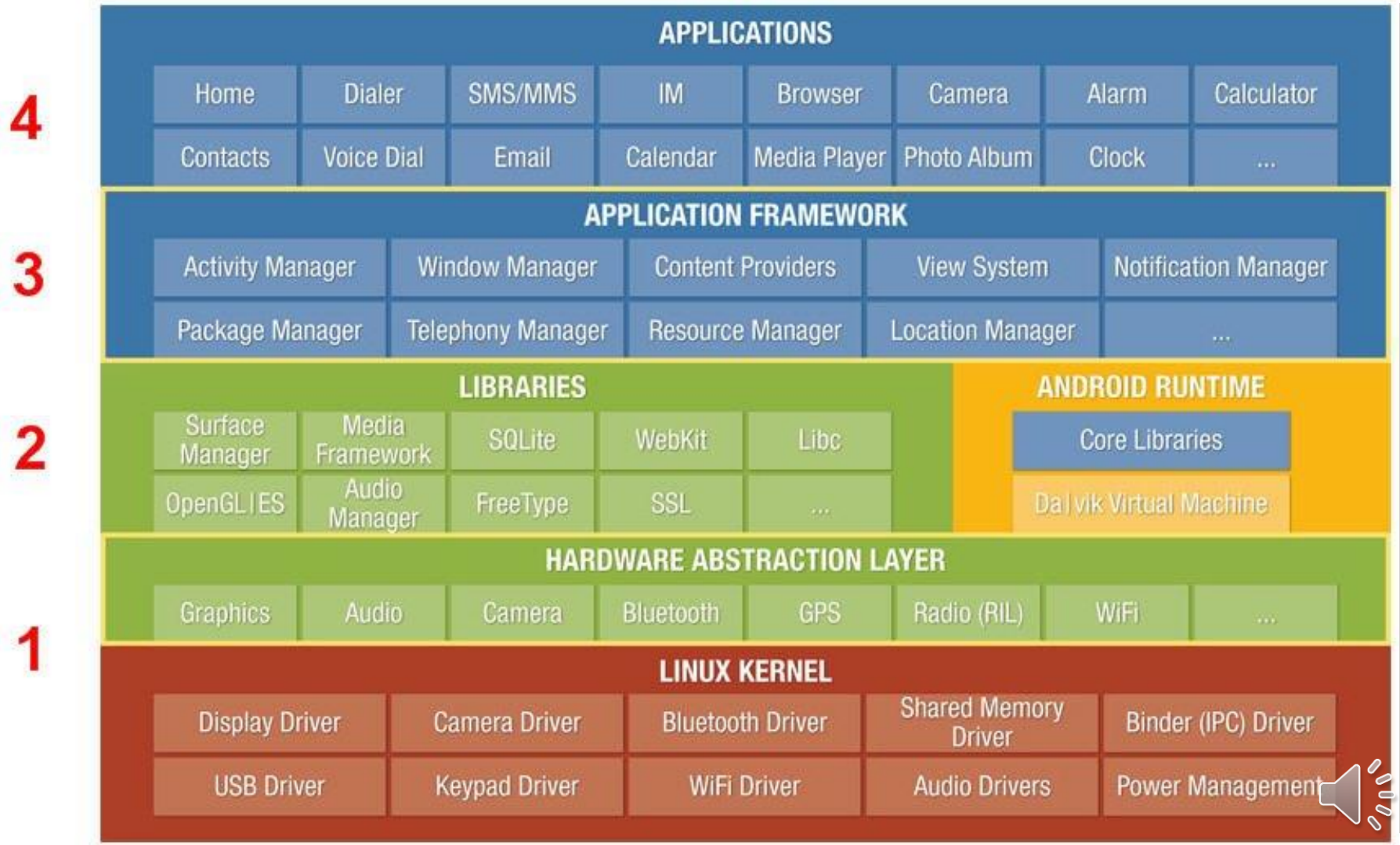
VON NEUMANIN KONE

- Perustuu (keskus)suorittimeen, joka keskustelee muistilaitteen sekä IO (syöttö ja tulostus) -laitteiden kanssa
- Arkkitehtuurin takana on ohjelmoitavan tietokoneen periaate
  - Keskusmuistissa on käsiteltävän datan lisäksi myös ohjelmalliset käskyt tiedonkäsittelyä ohjaamaan.
- Von Neumanin kone on käytännön arkkitehtuurimalli, ja käytännössä pohjapiirros, jokaiselle modernille tietokoneelle, kännykälle ja tabletille!

MODERNI TIETOKONE



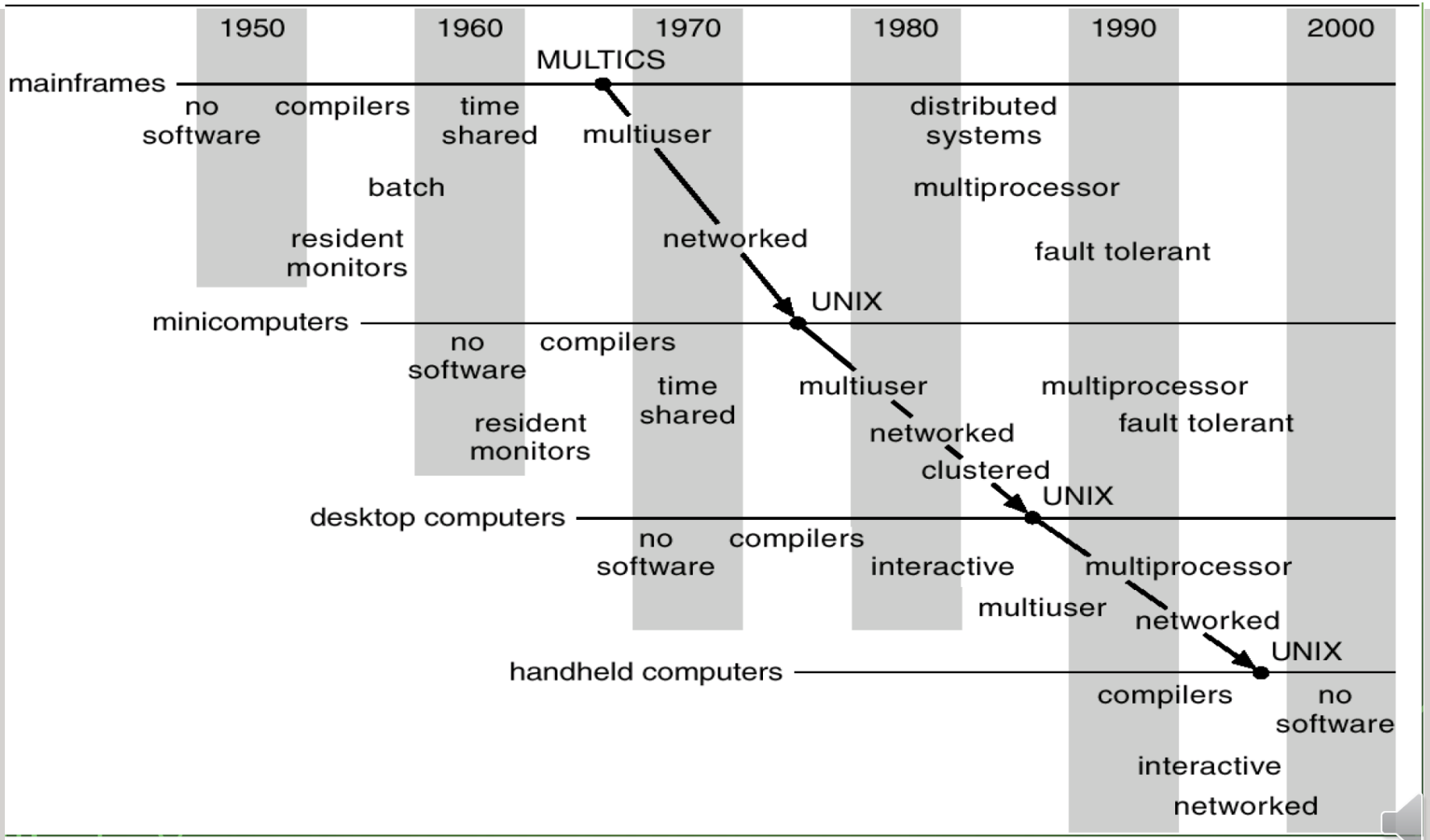
VRT. ANDROID (GITHUB DEVACADEMY)



KÄYTTÖJÄRJESTELMÄ YLEISESTI

- Käyttöjärjestelmä käytännössä hoitaa tiedonvaihtoa ohjelmisto- ja ajuritason kanssa (ja avulla) raudan suuntaan.
- Käyttöjärjestelmä mahdollistaa samanaikaisen ohjelmistojen ajamisen, resurssien hyödyntämisen sekä toiminnan valvonnan.
  - Prosessit, säikeet ja niiden valvonta
  - Levynhallinta
  - Suoritinajan jakaminen

VRT: NYKYISET KÄYTTÖJÄRJESTELMÄT, KEHITYS



- Kuva: Silberschatz, Galvin ja Gagne

KÄYTTÖJÄRJESTELMÄN 3 PÄÄOMINAISUUTTA

- Käyttöjärjestelmä on siis monimutkainen, mutta samalla keskeinen osa koko tietokonetta.
- Käytännössä käyttöjärjestelmälle voidaan asettaa kolme pääominaisuutta:
  - Virtualisointi (Virtualization)
  - Samanaikaisuus (Concurrency)
  - Pysyvyys (Persistence)
- Näitä ominaisuuksia käsitellään myöhemmin lisää sopivissa asiayhteksissä.

MITÄ TÄSTÄ LUENNOSTA PITÄÄ MUISTAA?

- Käyttöjärjestelmän määritelmä
- Tietokoneen rakenne, Von Neumann-arkkitehtuuri
- Käyttöjärjestelmällä on 3 päätehtävää: 1) virtualisaatio 2) samanaikaisuus ja 3) pysyvyys.
- Nämä kolme kun saa toimimaan niin käytännössä loppu on viilailua ja viimeistelyä.

HARJOITUKSET

- Ekat viikkotehtävät viikolla 2; ennen tätä ei erillisiä harjoitusryhmiä.
- Hakekaa kurssikirja verkosta, tutustukaa sen sisältöön yleisellä tasolla.
- Tutustukaa virtualisointityökaluihin, luokaa Linux-virtuaalikone.
- Käykää lukemassa kurssiin liittyvät tiedotteet ja ohjelma Moodle-sivuilta.

LUT
University