Operativ Systemer 1

Lavet af: Vivek Misra

Hardware Operativ Systems

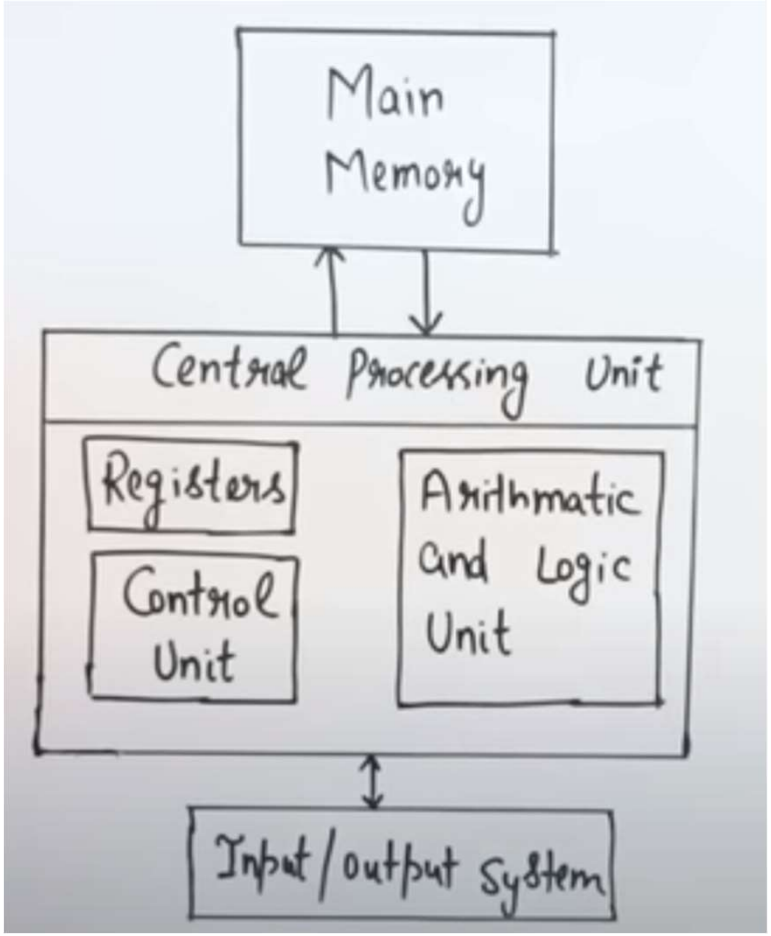
Nu introducerer vi opbygning af Hardware Operativ Systemer i en Computer

Von Neumann Arkitektur

- Von Neumanns Arkitektur er også kaldt for ”Store Arkitektur”.
- Det er der, hvor vi indsætter data inde i Main Memory.
- Vi indsætter også et program, som er en sæt af instruktioner i Main Memory.
- Alle arkitektur som bruges i dagens verden, kører på Von Neumanns Arkitektur.
  - Von Neumanns Arkitektur hjælper os med at konvertere vores input data om til output data
- CPU er også kendt som Central Processing Unit.
  - Arithmetic Logic Unit: Det er her, hvor regning eller operationerne på dataet foregår.
    - Eksempel på Operationer er Addition, Subrationer, Multiplikation, Divison og andre logiske operationer.
  - Registers: Det er den hurtigste memory som er tilgængelig, og gemmer midlertidig data.
    - Størrelsen af registre er meget lille som er eksempelvis 8 bits, og 16 bits.
    - Registers hjælper med, at holde byrden på CPU’en nede ved fx midlertidig gemme resultatet og give det videre.
  - Control Unit: Det er her, hvor vi scheduler tidssignaler. Det betyder eksempelvis, hvilken instruktion eksekverer vi først osv.
    - Control Signaler er der, hvor vi styrer læse/skrive operationer på de midlertidige data i Registeren.

Von Neumann Arkitektur

- Input/Output System: Det er alle de sfæriske enheder som befinder rundt omkring os i form af Hardware. Eksempelvis tastatur, mus, skærm og kamera.
- Fordi vi har alle forskellige komponenter, så kommer spørgsmålet om hvordan de forbindes sammen?
  - Dette er muligt gennem en BUS, med forskellige typer af Busser.
  - Busser er implementeret ved hjælp af multipleksere.
- SUMMARY: Det er en hukommelsesprogram som sørger for at gemme dataer og udføre programmer på det.



Type af Registers

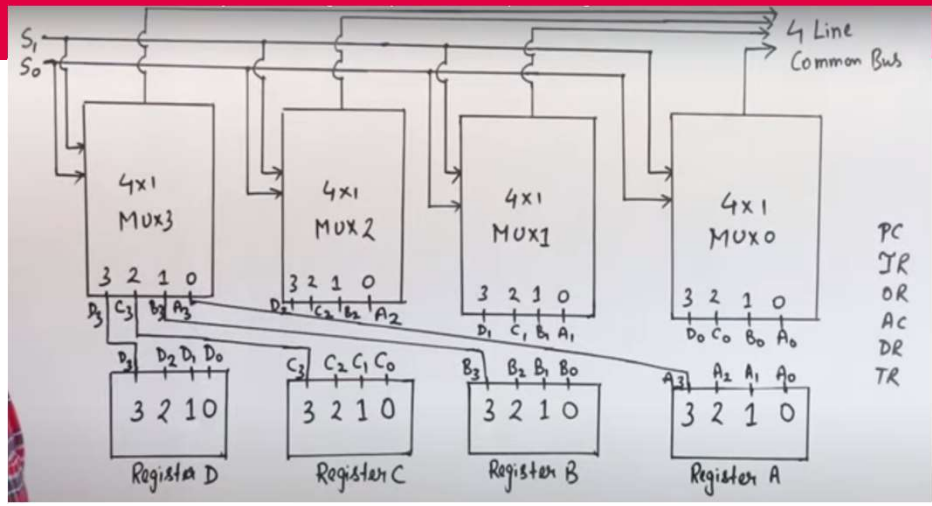
- Register implementeres gennem Busser.
- Man kan i en Memory putte antal af ord, hvor man i et ord kan sætte forskellige antal bits inde i.
  - Et ord er en ”Memory Representational Unit”.
- Address Register: Det er der, hvor man finder data inde i en Main Memory.
  - Addressens arbejde er, at finde data som ligger i en specifik adresse.
  - En adresse bit fylder 12 bit, grundet længden på registeren svarende til 2^12.
- Data Register: Det er der, hvor man gemmer dataet.
  - Data registerens størrelse er 16 bit.
- Accumulator: Det bruges til at gemme mellemdata fra Arithmetic Logic Unit.
  - EKSEMPEL: Hvis der er noget data som kommer efter ALU’en, så kan den midlertidig gemmes inden den kommer ud ved Output.
- Program Counter: Den bruges til at gemme addressen for den næste instruktion ved Program Counteren.

Type af Registers

- Instruktions Register: Denne Register bruges til at udføre instruktioner inde ved Main Memory.
  - Man har 16 bits på Instruktions Register, hvor vi bruger instruktioner som STORE, LOAD osv.
  - Vi bruger de ting som vi kender COS som er Opcode, Operand osv.
- Temporary Register: Det er de data som gemmes midlertidig.
  - Temporary Register har 16 bits.
- Input Register: Det henter data fra vores Input Devices såsom tastatur og mus.
  - Dataet hentes fra vores Input Devices og indsættes ved Input Registeren.
  - Input Register er på 8 bits.
- Output Register: Det henter data fra vores Arithmetic Logic Unit, som gives videre til Output Devices.
  - Den har det samme, og det er 8 bits.

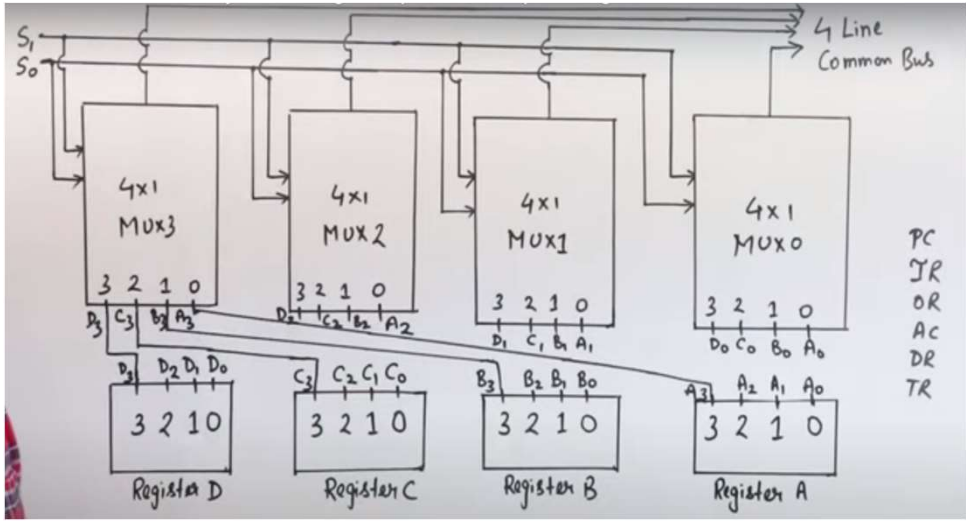
Kommunikation gennem Busser

- Vi kommer i denne afsnit til at snakke om, hvordan Registre kommer til at snakke med hinanden gennem en Bus.
- Vi har snakket om forskellige Registre som eksempelvis Program Counter, Inputregister, Outputregister osv.
- Vi kan se på billedet til højre, at vi har 4 Registre som normaltvis plejer at være 16 bit, men vi har taget udgangspunkt i 4 bit registre.
- På venstre hjørne øverst, kan det tydeligt ses at vi har Selektlinjer som er S0 og S1, hvor man ved hjælp af disse udvalgte linjer giver registrene adgang til at få output og derved indlæse dataene og dele dem med hinanden.



Kommunikation gennem Busser

- Vi kan se på billedet, at der er dannet forbindelse mellem Register 1, hvor A0 er forbundet med Register 0, A1 er forbundet med Register 1 osv.
- Det betyder, at når vi eksempelvis sender data fra Register A til Register B, så kan det være at vi sender noget fra Register 3 og over til Register B under Registeren 3.
- Når vi sender data'er afsted, plejer vi at loade vores data i en Bus.
- Man plejer også, at blive spurgt efter hvor mange Multiplekser der er krævet.

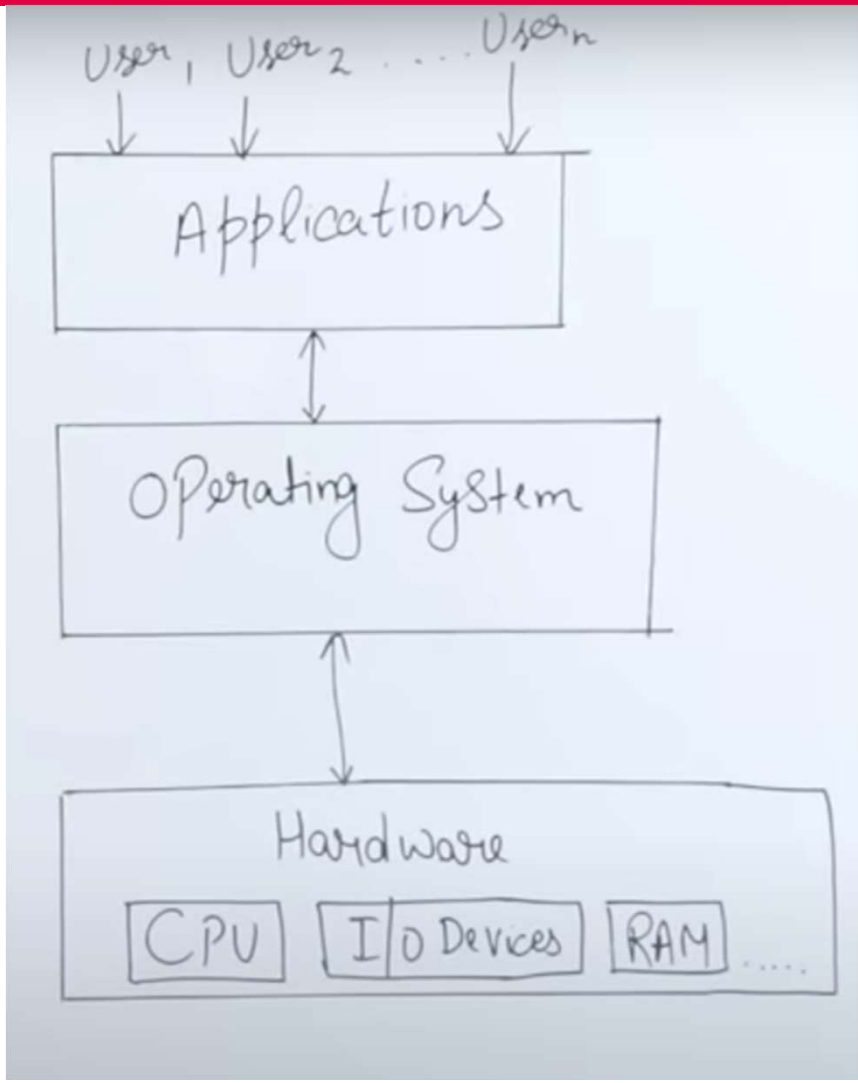


Software Operativ Systems

Nu introducerer vi opbygning af Software Operativ Systemer i en Computer

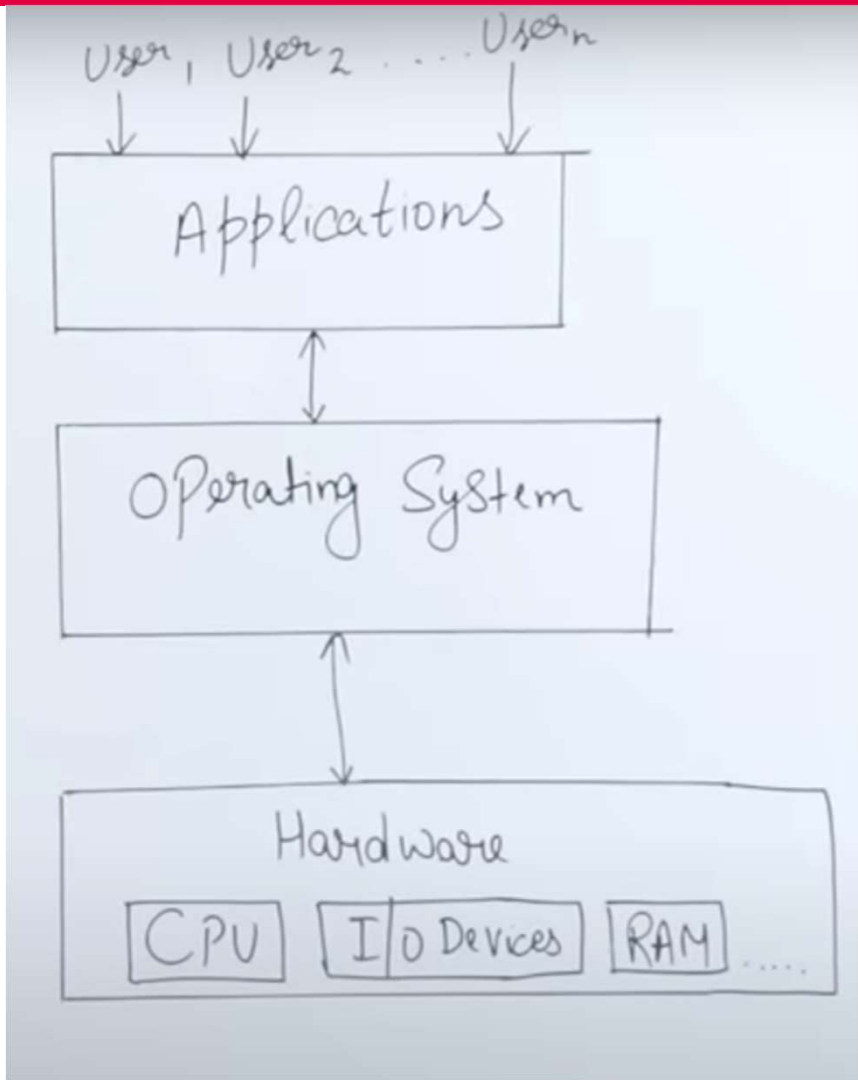
Software Operativ Systemer

- Vi kender allerede om Operativ Systemer på forhånd.
- Operative Systemer er et system, som kører på Computerens Software.
- Man kan sige, at det fungerer som et slags interface mellem brugeren og hardwaren.
- Vi har et billede til højre, som er inddelt i 3 dele.
  - Applikationsprogrammer
  - Operativ Systemer
  - Hardware



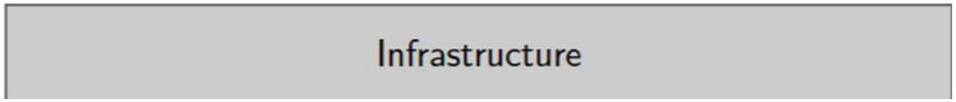
Software Operativ Systemer

- Applikationsprogrammer er eksempelvis som Microsoft Office 365 der kører på Interface.
- Operating Systemer kan være eksempelvis Windows, Apple og Linux
  - OBS: Vi brugere OS til at kunne interagerer med systemer bedre.
  - Hvis vi ikke havde OS, så skulle vi bruge kommandoer til at finde frem til filer, derfor fokuserer vi på Linux.
- Hardware er den fysiske genstand, nemlig computeren selv.
  - Eksempel på Hardware, kan være I/O-Devices, nemlig tastaturen, musen, skærmen, monitoren. Nemlig alt den fysiske genstand som er en del af computeren, og som kan arbejdes med.
  - Vi plejer også, at forbinde Hardware med Infrastruktur i sammenhæng med Servers.



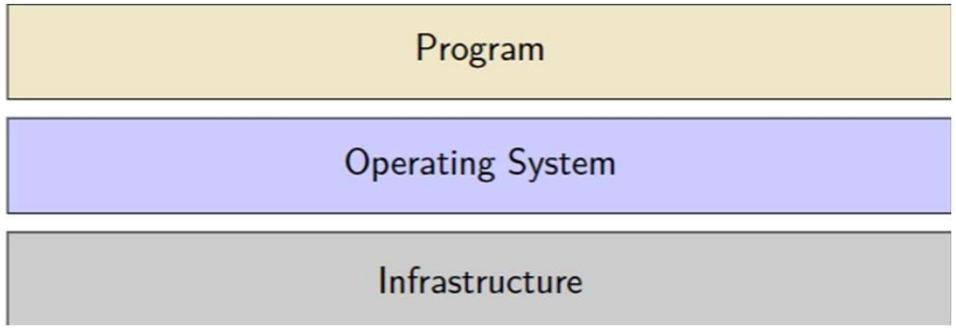
Infrastruktur

- Ved Hardware nævnte vi om Input og Output Devices.
  - Det hjælper os med, at skabe interaktion mellem brugeren og systemet i PC'en.
- Vi plejer typisk at referere til Hardwaren.
- Vi plejer også, at snakke om systemer og hvilken server en hardware kører på?
- MEN TYPISK ER INFRASTRUKTUREN FORBUNDET MED HARDWAREN!



Server

- Server er der, hvor maskinen kører og udleverer et eller andet form a materialer!
- Når man snakker om en server, så er der mange ting som kører på en server.
- Eksempelvis, så har man Operativ Systemer som primært er kørt i Windows, IOS og Linux!
- Det er forskelligt fra præferencer, men vi kommer til at fokusere på Linux!



Virtual Machine Manager VMM

- En Virtual Machine Manager er også kendt som en Virtual Machine.
- De fleste computer, der opererer på Linux kører på Headless-Linux!
- **Virtual Machine har til ansvaret for at kunne køre gæst-operativ systemer ved siden af den oprindelige operativ system.**
- **EKS: På en Acer Windows Computer, kan man installere MacIOS.**
  - **MEN HVORDAN ER DET MULIGT, AT INSTALLERE IOS?**
- **Den måde, at det er muligt, at installere IOS på er gennem Virtualbox.**
  - **DET ER EN LUKKET SYSTEM SOM IKKE INDEBLANDER SIG MED DEN ORIGINALE SYSTEM.**
  - **Vi plejer, at anvende begreberne Hypervisors og Closed Box på Næste Side!**

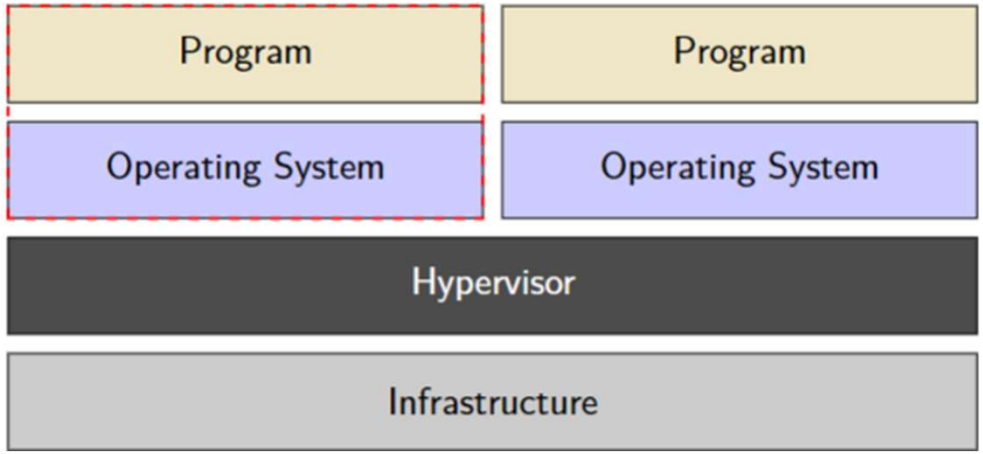
Image 1: Dette billede viser, at vi har en Acer Nitro Computer, som oprindeligt kører på Windows - har fået Ubuntu downloadet som et sekundært styresystem gennem en Virtual Box.

Closed Box & Hypervisors

- **Hypervisors er dem som skaber det grafiske overblik og den virtuelle ram for den virtuelle boks.**
  - Den placerer efterfølgende rammen af den virtuelle infrastruktur, som er kaldt virtuel maskine.
  - En Virtual Machine er en kombination af et program og et operativsystem.
  - Man kan sige, at en Hypervisor er en Virtual Box, hvor man kan installere et operativsystem og arbejde på det.
  - Der findes mange typer af Hypervisors, men vi har fokus på Type 1 og Type 2 Hypervisors. Disse fremvises på næste side!

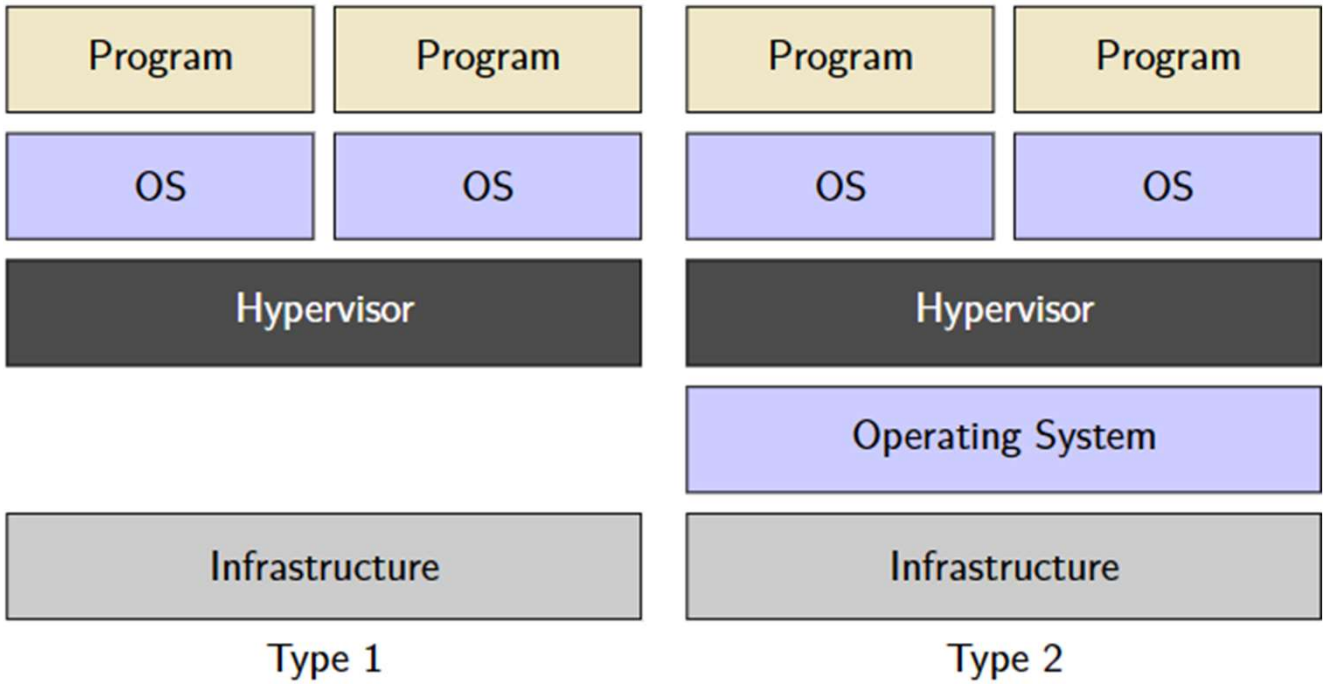
Shared Libraries

- **Shared Libraries er det som ligger mellem programmet og det operative systemer.**
  - **Man er her i stand til at kunne dele alt med andre mennesker i en gæst-OS.**



Eksempel på Hypervisors

- **Vi kan se, at vi har Eksempel på Type 1 og Type 2 Hypervisors.**



Fordele

- Vi har en Operativsystemsprogram, hvor man i containeren har en Hypervisor.
  - Inde i Containeren findes der en Hypervisor.
  - Denne Hypervisor er kaldt for Container Runtime, og er virtuelle maskiner, hvor alt virtuelle maskineri er skåret ned.
- EKS: Hvis man ønsker at have kernen, så skal computeren bruge Hypervisor, så den kører direkte fra brugerens computer og er istand til at vise grafikken på en lukket måde.

Ulemper

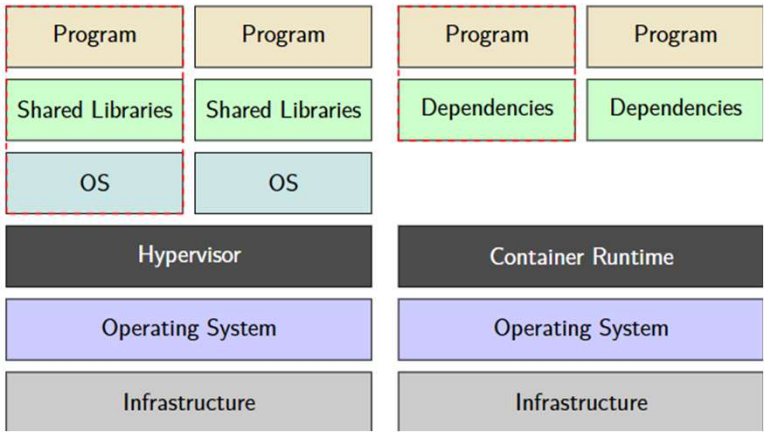
- Virtuelle Maskiner forhindrer deling af resourcer.
- Man har flere forskellige GB'er som fylder på ens computer.
- Når man har flere GB'er som fylder på computeren, så resulterer det at computeren bliver langsommere.
  - EKS: Vi kan se, at Docker Desktop og Virtual Oracle Box fylder meget på Computeren.

Docker

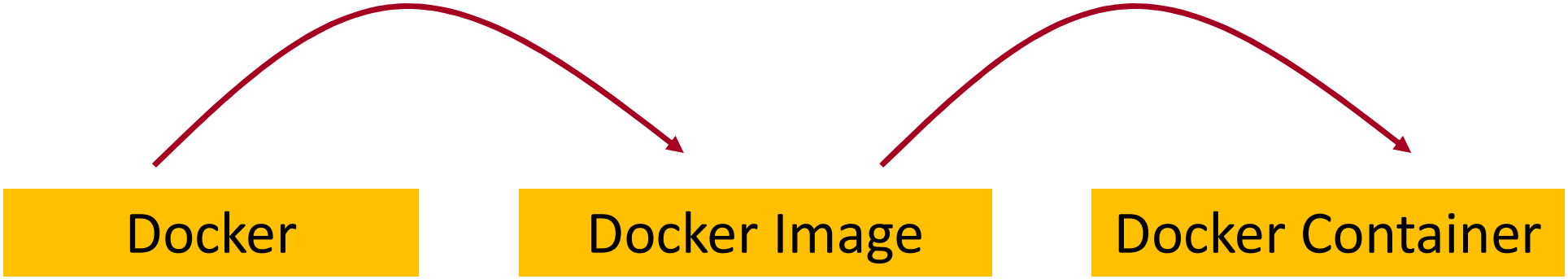
I denne afsnit kommer vi til at introducere til Docker!

Introduktion til Docker

- **Docker har monopol ved tankegangen for Containers i dag.**
- **Container Runtime er beregnet til den sidste del af Docker.**
- **Man starter altid med at oprette en docker fil, hvor en der angives en tekstlig kommando om hvordan en container skal oprettes og hvordan den skal fungere.**
- **Når Docker-filen er færdig, så kan der bygges en docker image som er teknisk set en opskrift til at danne en Container.**
- **Í en Container har man Miljøvariabler, som er en liste over variabler, hvad enhver variabel foretrækker.**
  - **EKS: Vore præference er, at kunne have information om brugeren og andre konfigurationer. Derfor kan det være, at vi får mange liste over disse ting.**
- **På næste side gengiver vi en Eksempel over de 3 Begreber i Docker.**



Docker Container Runtime



Jeg er en fil!

Jeg opstiller en opskrift!

Jeg er den endelige produkt efter at have dannet opskriften i filen.

Docker Registry er ligesom en Docker Github eller en Google Drev, hvor man kan dele filer gennem pull og push requests og lignende.

Typer af Operativ Systemer

Vi kommer til at introducere forskellige typer af Operativ Systemer.

Batch

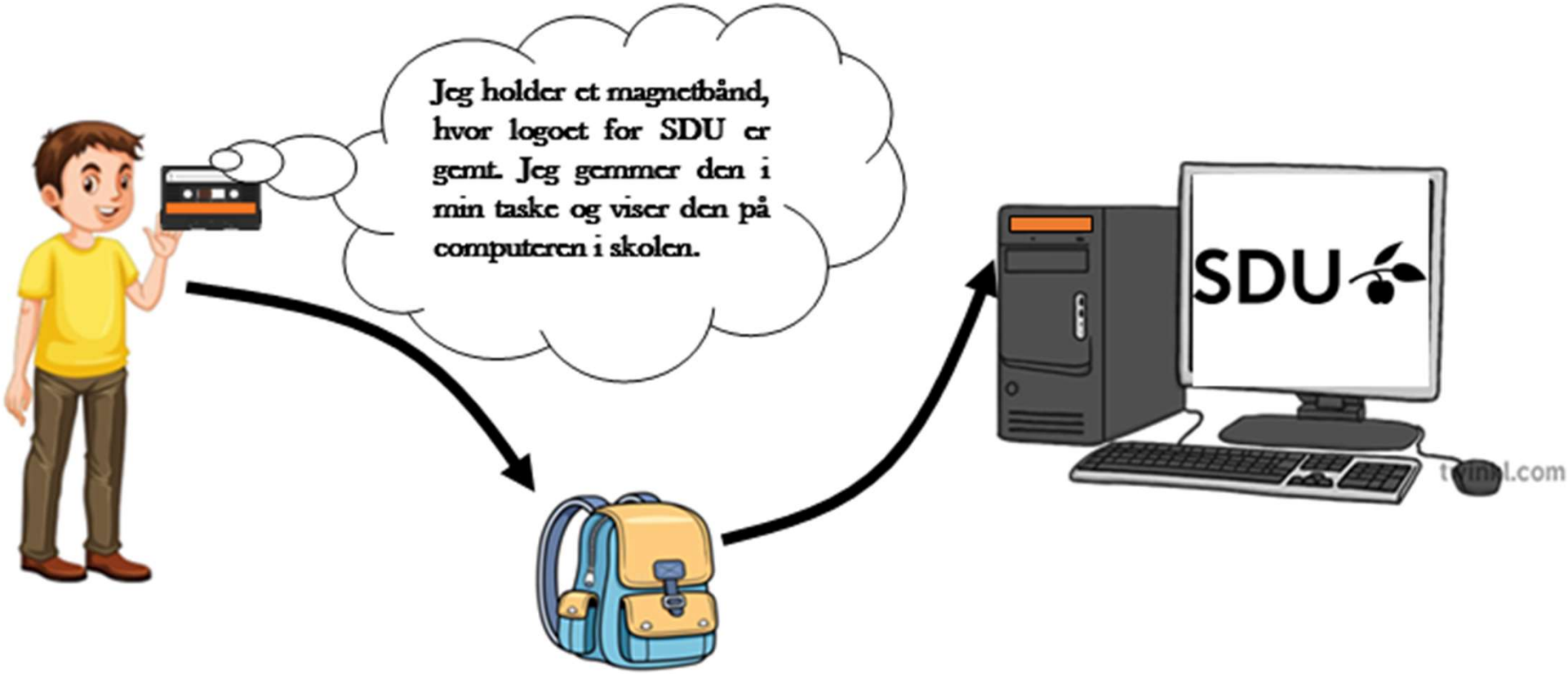
- Batch-operativsystem er, hvor vi laver en batch af lignende slags job og giver den til computeren, så den kan udføre den. I 1960'erne, da denne systemkunst blev brugt dengang, de store virksomheder som NASA eller ISRO, der plejede at arbejde med tungt beregnings- og forskningsarbejde på det tidspunkt. De var nødt til at bruge en eller anden beregningsmetode for at udføre deres arbejde. I dette tilfælde på det tidspunkt blev ting som hulkort, papirkort, papirtape og magnetbånd brugt af disse virksomheder, hvilket betød, at hvis nogen ville aflaste deres arbejde, så ville de højst sandsynligt gøre det på disse ting.

- For at give en klar forståelse af dette, kan vi lave et scenarie, hvor vi som ingeniører på det tidspunkt gemte nogle data inde i vores cd'er og dvd'er, og så havde vi vores stationære computere såsom Macintosh på SDU og tog derefter til universitetet for at udføre eller afspille dataene i dem. Vi kan sige, at universitetet eller de computere, der er på fakultetet, fungerer som en operatør, der har til opgave at fortælle brugeren resultaterne af inputtet.

- For at give et mere detaljeret indblik i operatøren, kan vi se, at operatøren konverterer filen til batches, som er ligesom inputtet. Batchene vil blive givet videre til CPU'en, som på det tidspunkt vil fungere som input til CPU'en og returnere som en Output. Bemærk, at CPU'en under denne proces er i inaktiv tilstand, hvilket gør, at denne proces kræver mere tid.

Batch

- Vi fremviser et billede af Batch!

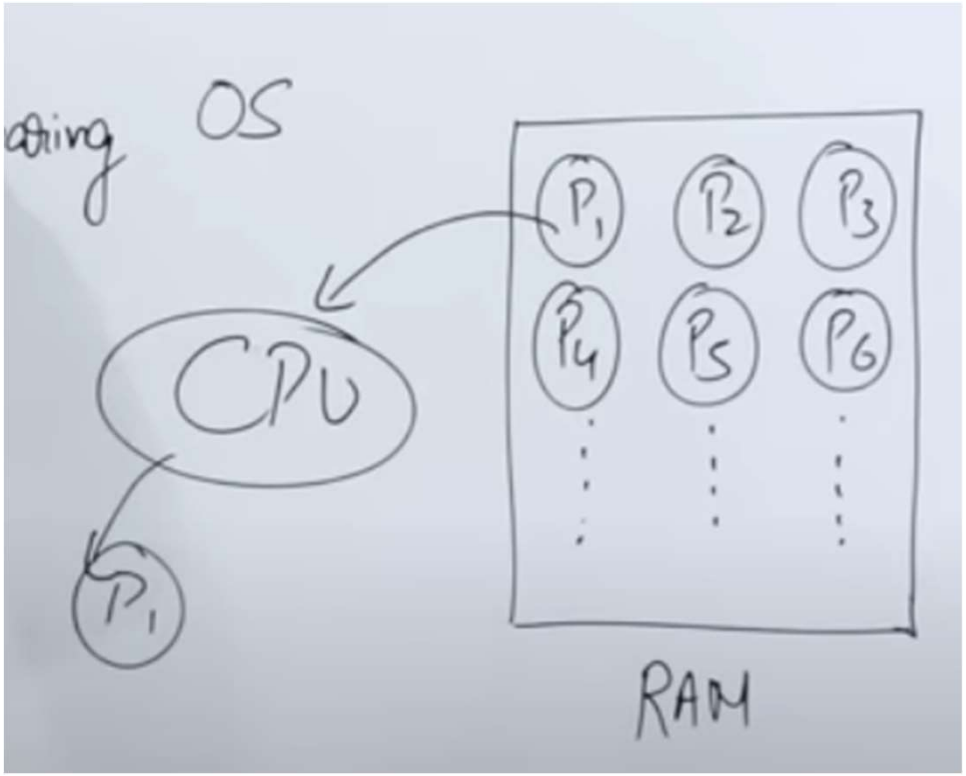


Multiprogramming

- I denne del skal vi diskutere om Multiprogrammering og Multitasking, og hvad forskellen er mellem dem.

- Multiprogram: Dette er et gammelt koncept, som betyder at bringe flere processer ind i en RAM. Ser vi på billedet til højre, kan vi se, at vi har flere processer inde i en RAM. Det betyder, at hvis vi tager en proces uden for RAM'en og giver den til CPU'en, så vil CPU'en udføre denne proces direkte, indtil den proces ikke gør noget.

- Hvis en proces tilfældigvis har en kommando, som øjeblikkeligt stopper udførelsen af den pågældende proces, så vil CPU'en fortsætte til den næste proces i en ikke-forebyggende tilstand. Dette betyder, at CPU'en aldrig vil mangle arbejde med hensyn til at udføre Process. Dette vil resultere i, at CPU er aktiv!



Multiprogramming

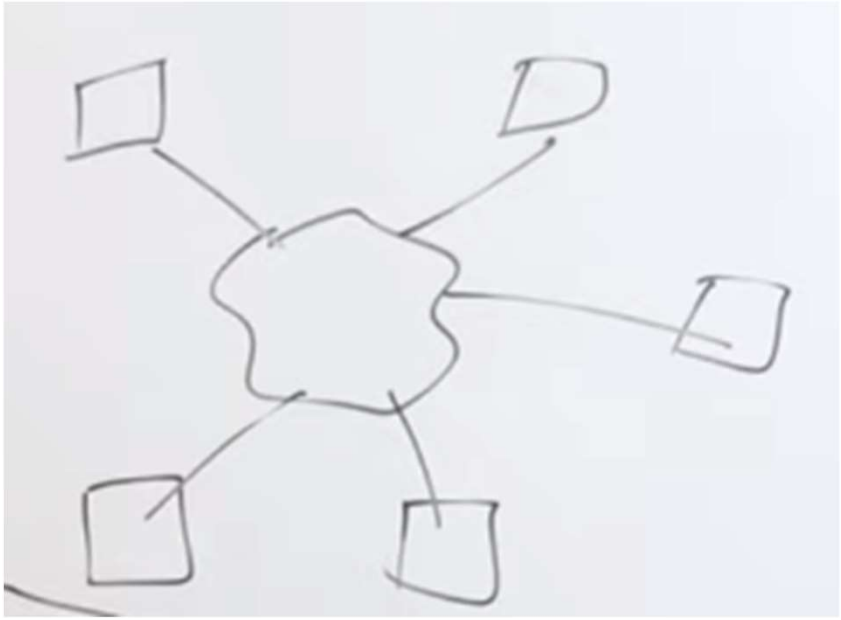
- I det foregående afsnit talte vi om konceptet non-Preemptive, hvor CPU'en er aktiv. Men i Multitasking er CPU'en forebyggende.
- I dette tilfælde kan vi se, at Multitasking er, hvor timeshare er inkluderet. Hvad der i bund og grund sker i timesharing er, at mens CPU'en udfører den forskellige proces, er en specifik tidsplanlægning allokering for eksekveringstiden.
- Det betyder, at CPU'en måske løser den første eller anden del af den første proces og derefter går videre til den næste. Hvad dette resultat er, at udførelsen af processen kører hurtigere, mens i Multiprogrammering var processerne en smule langsommere. Vi vil snart dække Round-Robin-konceptet.

Real Time Operating System

- Når vi taler om Real Time System, så refererer vi normalt til tiden, som er aktuel. Det betyder, at hvis vi udfører en proces i nuværende, kaldes det et realtidsoperativsystem. Herunder har vi to små koncepter, som er Hard og Soft.
- Definitionen af Hard Real Time System er, hvor forsinkelser IKKE er undtaget, hvorimod definitionen af Soft Real Time System er, hvor små forsinkelser accepteres på grund af nogle eksterne faktorer.

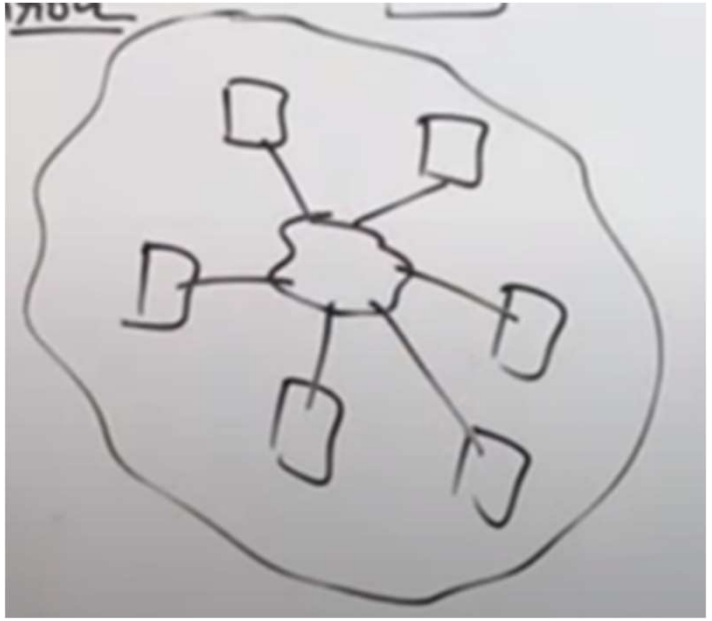
Distributed Systems

- Når vi taler om Real Time System, så refererer vi normalt til tiden, som er aktuel. Det betyder, at hvis vi udfører en proces i nuværende, kaldes det et realtidsoperativsystem. Herunder har vi to små koncepter, som er Hard og Soft.
- Definitionen af Hard Real Time System er, hvor forsinkelser IKKE er undtaget, hvorimod definitionen af Soft Real Time System er, hvor små forsinkelser accepteres på grund af nogle eksterne faktorer.



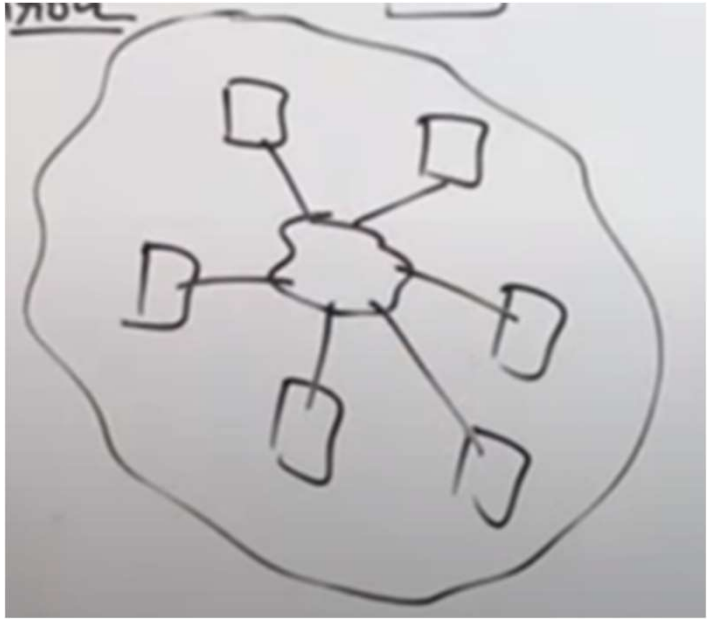
Clustered Systems

- Clustered System er det samme som Distributed System, den særlige ting, der gør det distinkt, er, at det er klynget inde i et lokalt netværk.
- Det betyder, at alle de maskiner, der var lokalt spredt, er en del af en gruppe, hvor de deler data inde med hinanden. Dette er meget fordelagtigt med hensyn til skalerbarhed, men også i tilfælde af standard som et distribueret system.



Embedded Systems

- Dette system er baseret på en fast funktionalitet, hvor der ikke kan foretages ændringer inde i operativsystemet, når først en funktionalitet er defineret i begyndelsen. Det betyder, at hvis vi har en mikrobølgeovn og vi gerne vil varme maden, og så beslutter vi os for at fryse maden mere. Mikrobølgesystemet vil ikke tillade brugeren/klienten at gøre det og vil i stedet bede klienten om at gå til et køleskab.



SLUT 1

Lavet af: Vivek Misra