**1: Forklar Cloud Computing. Herunder IaaS, PaaS og SaaS. Hvad er det. Hvad er det godt for. Alternativer.**

Man kan bruge cloud som man har lyst til og har behov til.

Hardware er ikke tilgængelig under clounding.

**SaaS:** software as a service

* aplikation som kører i cloud, man har login, internet…. Primært mod bruger af aplikation. Netflix, google docs, eller de hjemmesider vi udvilker færdig, som kunden bruger.

**Paas:** Platform as a service:

* Det er primært for udvikler, som gerne vil hoste deres program.
* Man behøves ikke at vide noget om clouding, man skal bare vide hvad servicen hedder
* Man skal kun bekymre sig om sin applikation

**IaaS:** Infrastruktur as a Service

* Mere frihed
* Den går ud på, at kunden får adgang til en virtualiseret infrastruktur med servere, netværksswitche, storageudstyr og hvad man ellers måtte ønske.

Fordele:

* Man behøver ikke at købe sin egen server, man kan bare leje en når man vil og koble den fra når man vil.
* Man betaler kun for det man bruger
* Jeg behøver ikke at tænke på hukommelsen
* Jeg behøver ikke at tænke på vedligeholdelsen og sikkerhed

Fælles for alle 3 lag

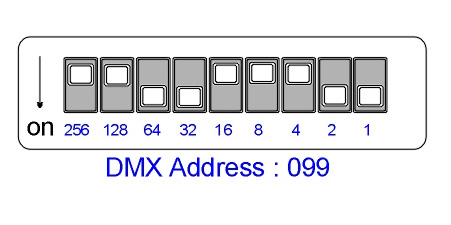
* Ingen hardware

Alternativer

* Have server derhjemme

**2: Forklar hvorfor computer bruger det binære talsystem og ikke det decimale talsystem**

* Grunden til at man ikke bruger decimale talsystem er fordi computeren indeholder nogle switches. Hvis man skulle bruge det decimale talsystem, så skal antallet af decimaltal være tændt i form af switches. F.eks. Hvis decimaltallet er 30, så skal der være 30 switches tændt i computeren. Hvilket kan blive utrolig meget, da computeren er begrænset for switches.
* Derfor anvender man binære tal, da det minimerer antallet af swicthes(bytes). F.eks. Så kan man med 8 bytes får op til 256 tal, hvilket gør det meget mere fleksibelt end decimaltal. Hvis man nu har 16 og 32 bytes, så har man lige pludseligt rigtig mange tal muligheder.



**3: Forklar hvordan en computer fungerer med udgangspunkt i "Fetch, Decode & Execute"**

Von Neumann layout:

* Control unit = Er ansvarlig for at decode instruktioner og kontroller hvordan data kører rundt i computer systmet
* ALU = Står for udregninger og logiske beslutninger
* BUS = Har ansvaret for transporterer data rundt i computeren (Det er de små ledninger vi ser på motherboard)
* PC = Program counter holder memory address for den næste instruks der skal feches fra RAM
* MAR = Memory address register
* MDR = Memory data register
* CIR = Current Instruction register
* Accumulator = Den gemmer data midlertidigt mens programmet kører.

F.eks. En adresse bliver gemt i PC og bliver kopiret til MAR og så bliver data i den adresse sat ind i MDR, hvor den bliver placeret i CIR, efter inkrementer PC med 1. Nu bliver data decoded og eksekveret. Hvis instruktion siger at hent adresse 10, så bruger control unit MAR og henter data til MDR, hvor den bliver sat ind i accumulator.

Fortæl hvad von neumann arkitekturen består af:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Hvordan den fetch, decode & execute

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

**4:Hvad er et interrupt?**

**Hvordan samarbejder HW og operativsystemets SW ved en interrupt.**

**Hvilke to typer interrupts kan der komme. Hvad sker der med processerne ved interrupt.**

Hvis skærmen f.eks. Er på pause skærm og man så bevæger musen, så kommer man så tilbage til normal skærm. Dette kaldes en interrupt, fordi hardware spørger om lov til at afbryde CPU'en som var i gang med at kører pauseskærmen. Når CPU'en bliver afbrudt, så overgiver den kontrollen til en interrupt handler som er en komponent i operativsystemet.

Interrupt handler gemmer først det data CPU er i gang med at bruge, så den senere kan genoptage processen igen.

Timer interrupt

I/O interrupt

**5:Forklar Operating System og Processer**

**Hvorfor operativsystem. Hvad er en process. Forskel til program. Process life cycle.**

Operating System er noget der forbinder hardware og software sammen. OS er et program der kontrollerer hardware, applikationer, hukommelse og proces på en computer.

Hovedformål:

* Styrer hardware, da det kun er OS der direkte, kommunikerer med hardware
* Styrer applikationerne
* Laver en user interface

Process vs Program

* Et program er en file som indeholder kode
* Process er når et program bliver eksekveret.

Process states and life cycle:

Et billede, der indeholder enhed

Automatisk genereret beskrivelse

Scheduler

* Den kigger ind i ready og ser hvilken process den nu skal vælge
* OS Scheduler benytter interupts til at skifte mellem processer.

Proces Control Block

* Det er når et program bliver interrupt, så bliver programmet gemt i det stadie den befinder sig i.
* OS har en tabel med alle PCB.
* Hver proces har et PCB (Process Control Block)

**6: Forklar Threads**

**Relatere til processer.**

Thread er en måde at gør flere ting på en gang. Det er dele af en process. F.eks. Bliver det brugt i spil, hvor der sker mange ting på en gang.

Data og koden er altid ens i threads. Der bliver oprettet flere threads, i stedet for at man opretter flere processer.

Hvis man skal oprette en proces til hver thread, så skal det gemme sin egen data hver gang, hvilket er spild af resourcer. Det vil også tage længere til fordi man skal skifte mellem processerne.

Derfor er en proces med flere thread meget bedre.

**8: Hvad er paging?**

**Hvorfor laver OS memory management. Hvad er virtuelle og fysiske adresser. Hvad er pages, frames.**

Paging er en memory management process/funktion, som gør at computeren gemmer og henter data fra harddisken til RAM'en når den skal bruge den.

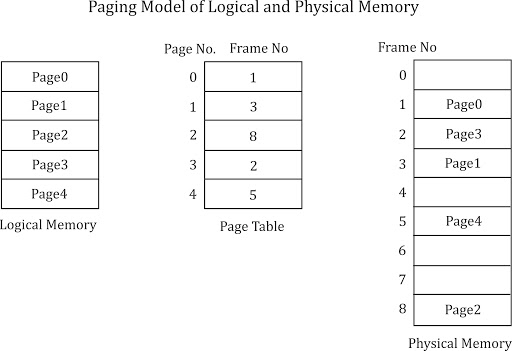
Memory management er virkelig effektivt, da det får programmer til at kører hurtigere. Normal består en computer af RAM og harddisk, som begge er hukommelser. RAM bliver brugt som hoved hukommelse i det at programmer bliver udført vedhjælp af den, da den er meget hurtigere til at finde alle adresser frem.

Men typisk vil alle data være gemt i harddisken som er en meget stor hukommelse og når et program skal køres kopier den data til RAM'en, da harddisken henter data enkelvis og tager en del længere tid. RAM'en er derimod meget hurtigere til at hente og eksekverer data.

Når noget data ikke bliver brugt i den fysiske adresse i RAM'en så bliver det imidlertidigt ført til den virtuelle adresse i harddisken og gemt som pages som er lige store. Dette gøres ved hjælp af CPU'en.

I RAM'en er hukommelsen også delt op i frames med lige store størrelser.

For at holde styr på hvilke frame tilhører hvilket page, så laver computeren en page table



**9:Concurrency**

**Samtidlige processer. Delte resurser. Fordele og Problemer. Løsninger.**

Concurrent proces går ud på at flere processer deler ressourcer. F.eks. Hvis vi har et toilet, så kan man kun bruge den en adgangen og ikke flere på en gang, derfor er der sat en lås, som indikerer at toilettet er optaget. I computerverdenen så deler flere processer den samme hukommelse, hvilket kan være problematisk, da de måske kan få en opgave f.eks. At de skal ændre variablen x til noget andet.

Derfor er det vigtigt at ressourcer ikke bliver anvendt af flere processer på en gang.

Derfor skal man låse de processer som deler ressourcer, så en anden process ikke kan manipulere med det samme data.

Fordel:

* Ressourcer bliver behandlet uden fejl

Ulæmpe:

* Det tager længere tid, fordi man skal vente til at processen bliver færdig, før man kan køre den anden process.

Race condition

**10:Transactions**

**Hvad er en transaktion. Hvad er ACID for transaktioner. Locking. Commit og Abort.**

ACID

* Atomicity (all or nothing is performed)
* Consistency (data never shows unfinished results, ingen mellem resultater, dvs. enten er det gennemført eller ikke)
* Isolation (transactions overlapper ikke hinanden)
* Durability (data skal overleve fejl, man kan stole på data på disken og ikke programmet)

Hvad er er en transactions?

* Er en antal læsninger og skrivning som gennemføres
* En logisk sammenhæng mellem data
* E.eks. Hvis man skal overføre 5 kr fra a og b, det kraver en transaction

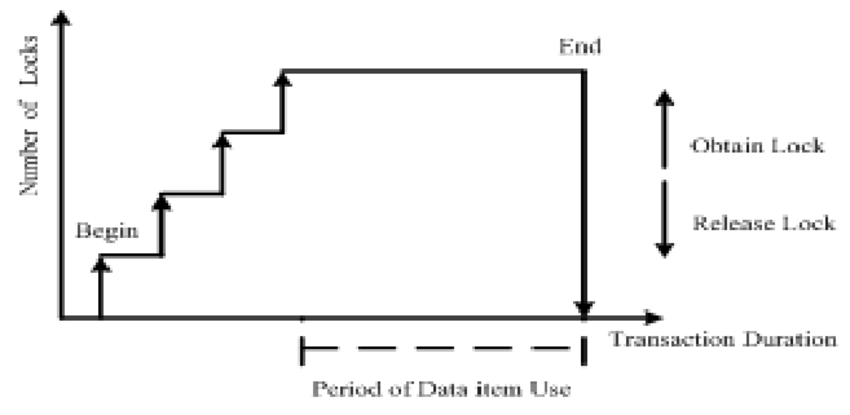
Hvis den under halvvejs stopper, så er data ikke consistens og det gør at transaction er atomicity

Hvis ikke hele transaction bliver gennemført, så skal alle ændringer ikke gemmes, men skal tilbage som det var. Transaction skal kun gennemføres hvis alle læsninger og skrivninger bliver gennemført.

Derfor hedder der atomicity, alt eller ingenting

Two phase lock

* The transactions låser alle før udførelsen
* The full computing and writing is done
* Låsen bliver open igen



**11:Distributed Systems**

**Hvad er Distribuerede Systemer.**

Er en samling af separate og uafhængige software /hardwarekomponenter er netværkede og arbejder sammen ved at koordinere og kommunikere gennem meddelelsesoverførsel eller begivenheder for at opfylde det ene slutmål.

Man kan sige at det er flere computer der arbejder sammen for at løse et problem eller skabe et resultat

Distributed systmer har hver deres egen hukommelse

Distributed systmer kan flere computer tilslutte eller forlade systemet.

Et godt eksempel distributed system er cloud computing

Coordinering.

Fordele:

* Reliability
* Dele resources
* Hvis en del af system går ned, så kører det andet stadig (fx cloud computing)
* Øger performance

og ulæmper.

* At finde fejl, da systemet er kæmpe stort