

# Computersystemer 3

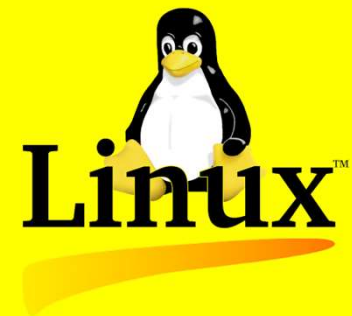
Lavet af: Vivek Misra

# Operating System

- En operating system er en Software som Kontrollere det samlede operation af en computer!
- Operating System er Eksempelvis Windows, Mac, Oracle.
  - Linux er også et eksempel på Operating System, som er blevet udviklet af en række "Entusiastiske Programmører".



ORACLE



# Historien af Operating Systems

- Nuværende Software Programmer er store og komplekse pakker som har været udviklet gennem tiden.



# Batch Processing

- I Process systemer, findes der jobs siddende i Mass Storage.
  - Disse jobs venter i en Job-Kø, når man eksempelvis er ved printeren.
    - PRØV, AT KIGGE PÅ WINDOWS INDSITLLINGER PÅ WINDOWS OG FIND PRINTER-KØ OG LIGN.
- Jo-Kø = Queue kører efter (FIFO-Princippet). FIFO=First Input og First Out.
  - FIFO siger, at Objekter bliver fjernet på samme måde som de blev de ankom på.
- **FACTS FRA BOGEN**

Tidligere Process Systemer gik på, at hvert job mødte med en række instruktioner som forklarede de steps der var brugt til at forberede masknen på det specifikke job.

Man brugte Job Control Language til at kunne forberede Maskinen ved at Enkodere Instruktionerne ved at anvende JCL

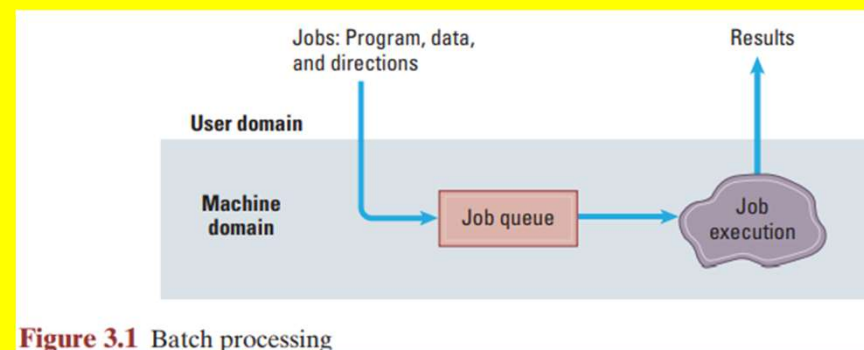
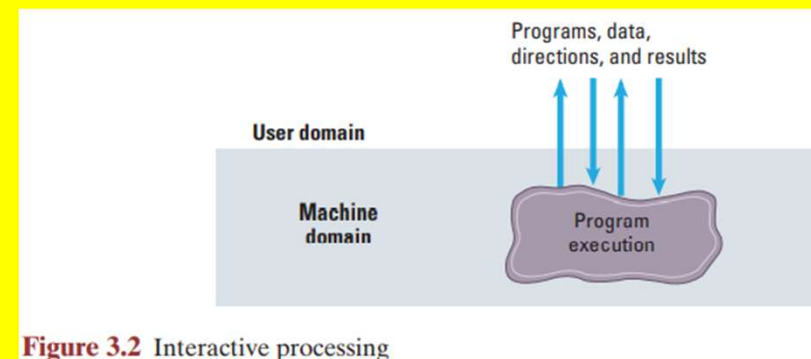


Figure 3.1 Batch processing

# Interaktive- og Real-Time Processering

- **Interaktive Processering** er en feature.
  - Den består af en Termin som består af en lille bitte Type Writer (Elektronisk).
  - Type Writer kan tage imod input og derved læse computerens Respons som var printet på Papir.
- **Real-Time Processering** er en proces på Real Tid.
  - Det er en processering som betyder, at aktioner/handlinger er udført i Real/Nuværende Tid.
  - Det betyder, at PC skal udføre arbejde med Deadline fra det rigtige Verden!



**Figure 3.2** Interactive processing

# Historien bag de opfundne Features

Tilbage i 1970 var Computere meget dyre. Maskiner skulle betjene mere end en bruger. Mange oplevede Forhindringer. Eksekvering skete KUN på et program/arbejde ad gangen!

For at kunne løse problemet designede man et/flere operativ system som havde mulighed for at betjene kunder/Brugere på samme tid.

Man ønskede at have features som timesharing, multiprogramming og multitasking.

# Time-Sharing og Multiprogramming

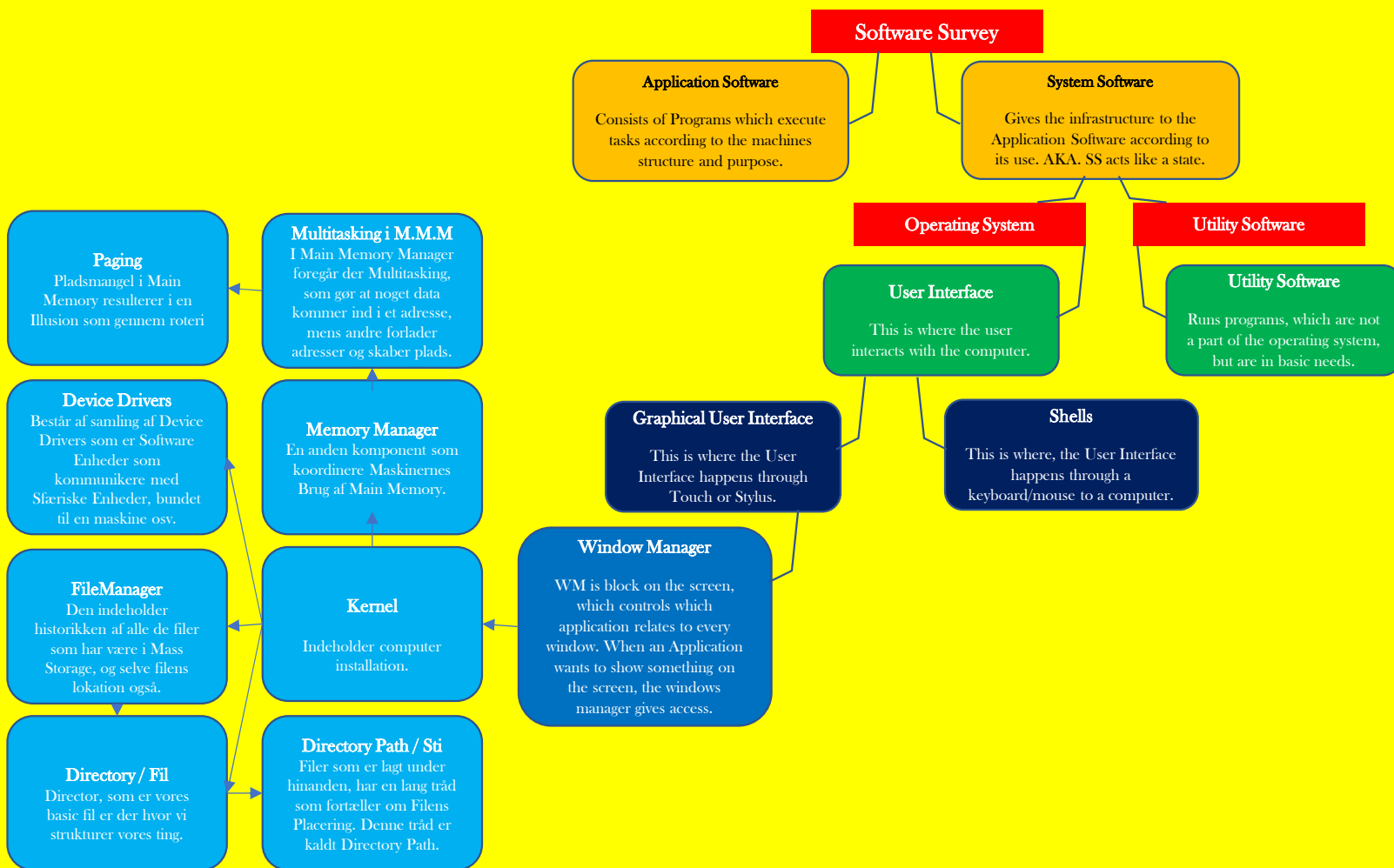
- Det er her, hvor Kunder betjenes på samme tid ved Maskinen!
  - Man implementerede også begrebet "Multiprogramming".
- Multiprogramming
  - Det er der, hvor tiden er delt inde i intervaller og er så Eksekveret ved hvert Job.
  - Hvert Job er begrænset til en Interval ad gangen.
  - Ved slutningen af Arbejdet blev det sat til side og et andet program var tilladt til at Eksekvere ved et Interval.

# Multiprogramming Teknikker

- **Multiprogramming = Multiuser Systemer = Multitasking.**
  - You know the drill! = At lave flere arbejde på samme tid.
- **Computer Operator**
  - Også kendt som den tidligere tidliggende system administrator som kiggede efter hvert detalje.
  - Nu er det ændret og her kom brugeren og computer tættere på hinanden.
- **Load Balacing**
  - Er der, hvor man dynamisk alokkere arbejde/job ved adskillige Processor.
  - Derefter resulterer det i en effektivt brug af Processor.
- **Scaling**
  - Er der, hvor man bryder arbejdet/task til små numre af arbejde/subtask.
  - Disse "subtasks" er passende med de Processor som er tilgængelige for det antal arbejde.
- **Embedded Devices og Embedded Systems**
  - Device som har systemer, der dedikeret til noget specifikt arbejde har et særligt operativ system.
  - Disse særlige operativ systemer er Embedded i sig og er kaldt for Embedded Systems.



# Software Survey



# Koordinering af Maskine Aktiviteter

- **Hvad er Process generelt?**
  - Det er den aktivitet som bliver Eksekveret under Kontrol af Operating System.
  - Man kan sige, at proces er den tid som tager for at kunne eksekvere et program/regnestykke.
- **Hvad er Process State?**
  - Process state er en snapshot af den situation som Eksekverering foregår i på den nøjagtige tidspunkt.
- **Hvad er hensigten med Scheduler og Dispatcher?**
  - De har arbejdet i at kunne holde styr på og koordinere Eksekveringer af Processer.
- **Scheduler aka. Gulvagten.**
  - Holder styr på historik af processer som er igangværende i PC-Systemet og derimod introducerer nye processer og fjernede færdige processer.
  - For, at kunne holde styr på igangværende processer benytter Scheduler sig af Process Tabel.
  - Hvert gang et Eksekvering af et program er anmodet danner Scheduler nyt entré i Process Tabel for Programmet.

# Koordinering af Maskine Aktiviteter

- **Hvad er Process generelt?**
  - Det er den aktivitet som bliver Eksekveret under Kontrol af Operating System.
  - Man kan sige, at proces er den tid som tager for at kunne eksekvere et program/regnestykke.
- **Hvad er Process State?**
  - Process state er en snapshot af den situation som Eksekverering foregår i på den nøjagtige tidspunkt.
- **Hvad er hensigten med Scheduler og Dispatcher?**
  - De har arbejdet i at kunne holde styr på og koordinere Eksekveringer af Processer.
- **Scheduler aka. Gulvagten.**
  - Holder styr på historik af processer som er igangværende i PC-Systemet og derimod introducerer nye processer og fjernede færdige processer.
  - For, at kunne holde styr på igangværende processer benytter Scheduler sig af Process Tabel.
  - Hvert gang et Eksekvering af et program er anmodet danner Scheduler nyt entré i Process Tabel for Programmet.
- **Dispatcher**
  - Er en komponent og som kigger på effektiviteten af tid i Eksekvering ved en Scheduled Process.
  - EKS: I tidsdeling/multitasking system, hvor arbejdet er udført af "Multiprogramming" som sker ved at man inddeler tiden i Korte Segementer som er kaldet Time Slice.
    - Disse er målt i Milisekunder / Mikrosekunder.

# Koordinering af Maskine Aktiviteter

- **Dispatcher**

- Det som sker efterfølgende er, at man skifter CPU'ens opmærksomhed blandt processerne som hvert process tilbyder eller kun tillod at blive Eksekveret i Time-Slices

- **Proces Switch**

- Husk, at skifte fra en proces til det andet en kaldt for Proces Switch / Kontekst Switch.

- **Interrupt**

- Når en Dispatcher sætter tid på vha. Time Slice Proces, så har den også en "Interrupt" som er en slags vækkeur.
- Vækkeuret er den som giver signal ved slutning af Time Slice.
- CPU reagerer på Interrupter på den nøjagtige samme måde som man ville gøre hvis man blev vækket pludseligt op.
- CPU færdiggøre Maskine Cyklus, når den modtager en Interrupt og gemmer dets cyklus i et position i Current Proces.
- Derefter begynder den, at Eksekvere Programmet som er kaldt for Interrupt Handler.
- I Interrupt Handler er det gemt i Prædetermineret lokation i Main Memory.

- **Den primære Hensigt med Interrupt Handler**

- Det er primært Eksempel på en del af Dispatcher, som forklarer hvad der sker under en/hvordan Dispatcher skal reagerer til en Interrupt Signal.
- Husk, at Effekten af Interrupt Signal er at stoppe nuværende process og derved overføre kontrollen tilbage til Dispatcher.
- Dette betyder, at Dispatcher udvælger processen som står først på proces tabellen og derved starter processen igen!
- Det er vigtigt, at kunne danne den samme stadie som var før Interrupt, fordi på den måde kan man starte Eksekveringen igen!

# Flag aka. Grænsevagt

- **Hvad er det?**
  - Hvis to computere fik adgang til en Printer, ville de ikke få noget ud af det. Derfor anvendes en flag, for at kontrollere adgangen til Printereren.
- **Tydelig Flag (0)**
  - Indikation om, at Printereren er klar!
- **Set Flag (1)**
  - Betyder, at printeren er allokeret.
- **Hvordan foregår processen?**
  - Når flaget er klar, så er anmodningen om printer adgang accepteret og processen startes. Hvis Flaget er set anmoder operating System en "request proces" som er "wait".
  - Problemet med denne metode er, at det kræver mange instruktioner. Her er Interrupter ikke nogen løsning, fordi det resulterer i en stak af anmodning til benyttelsen af Printereren.
- **Disable Interrupt og Enable Interrupt**
  - Disable Interrupt = Blokere flere fremtidige Interrupters
  - Enable Interrupt = Gør det muligt for CPU at kunne reagere til Interrupt signaler.

**FACTS:** Hvis en flag aktivitet starter med en Disable Interrupt og afslutter med en Enable Interrupt, kan ingen andre aktivitet afbryde rutinen.

# Test-Set Instruktion

Det er metode som er tilgængeligt i Machine Languages. Instruksen direkte CPU til at modtage værdier af en "flag" og derved sætte flage med inden en Maskine Instruktion. På den måde er der en fordel og det er, at man kan ikke splitte test-set instruktionen, da det allerede er et samlet.

# Semaforer

Opfattes som lys signaler i Flag implementering og styrer adgangen til sektionen af arbejde. **Sektionen som kan egentligt opfattes som sporet af jernbane af toget.** **Process kan opfattes som vores tog, hvor det er tilladt for toget at være på sporet en ad gangen.** Her kan det ses, at sekvensen af

Instruktioner kan kun være eksekveret en proces ad gangen.

Denne sekvens af instruktioner er kaldt for en Critical Region.

# Mutual Exclusion

Den kritiske region som blev nævnt tidligere er Mutual Exclusion. For at kunne komme i en kritisk region, skal processen finde en tydeligt semafor og så skal semaforer sættes inden den kommer i den kritiske region, skal den "clear" semaforen.

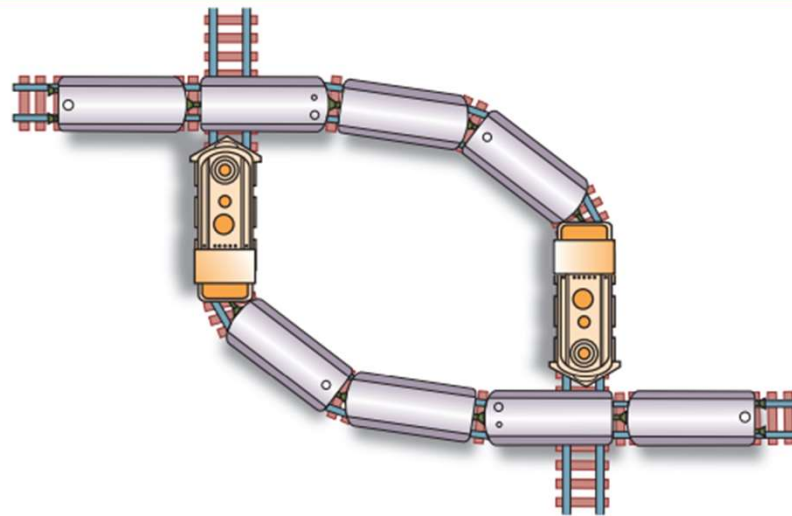
Hvis semaforen er fundet i en set-state, skal processen som kommer i det kritiske region vente indtil semaforer er "cleared".



# DeadLock

Det er her, hvor to processer er blokeret til at gå videre, fordi en ressource er alokkeret til den anden.

EKS: Hvis en proces er ved at printe, men vil gerne til DVD men hvor der i DVD som står en anden processer som vil printer.



**Figure 3.7** A deadlock resulting from competition for nonshareable railroad intersections

# Deadlock

På side 183 i bogen, er der opstillet 3 punkter som forklarer årsagerne til hvorfor Deadlock opstår. Hvis nogen af de årsager bliver angrebet, så kan Deadlock fjernes.

**KILL:** En bruger kan fjerne nogle processer væk som er med til at fjerne Deadlock og derved skabe plads i Proces Tabel.

Når man angriber de 2 første årsager af Deadlock, er det kaldt for Deadlock Avoidance Schemes.

# Spooling

Spooling, er der hvor man gemmer noget i en Driver i stedet for at direkte sende det til Enheden.

Gennem forestilling/illusion kan man i stedet sige at man er ved Enheden og derved holde data for output for senere brug, men i godt tidsperiode/tidsfordriv er kaldt Spooling.

For at give en Konkret Eksempe på Spooling, kan det eksempelvis ses at file Manager kan give adgang til flere processer, men med begrænset magt.

De fleste filer kan eksempelvis læse filen, men en proces kan skrive i det. Vi begrænser magten, fordi vi ønsker at undgå konflikten og derved slåskamp mellem processer.

# Login og Administrator

- **Login:** Et procedure til at få initial kontakt med PC'ens operating system.
- **Administration:** Også kendt som Super User og det er vedkommende som etablere konto.
- **Auditing Software:**
  - Er det som holder styr på historik af Aktivitet på Computersystem. Nogen hacker kan finde på at ændre register purpose som gør at man kan pludseligt opkræve ekstra plads i Memory Cellerne.
  - For at kunne rede sig selv imod disse aktioner er det nødvendigt at kende til privilegie levels. I tilfældet har vi to kategorier.

## Non-Privilege:

- Det er her, hvor CPU kan eksekvere alle instrukser i Machine Language.

## Privilege:

- Listen af de begrænsede instrukser er "aktivere/accepteret" her. (VIP).

# Privilegeret Instruktions og "Forsøget"

- Eksempler på accepteret og privilegeret instrukser kan være således, at man kan ændre indholdet af Memory Limit Registers og Instruksen der ændrer nuværende privilegie mode af CPU.
- Eksempel på den såkaldte "Forsøg"

En måde/forsøg at eksekvere et privilegeret instruktion, er når CPU er i non-privilegeret mode, og skaber en interrupt.

Interruptmkonverterer CPU'ens mode og overføre kontrollen af Interrupt handler med inden det opererende system.

Det understreges, at CPU kan ændre sit privilegie mode, gennem instruktionen (PRIVILEGE MODE).

Hvis nogen forsøg bliver lavet på at ændre privilegie mode, bliver operating system gjort opmærksom på det.

På den vil Operating System være i stand til at kontrollere integritet af Computersystemet!

# Andre Facts fra Undervisning

Nu kommer der en masse slides, hvor der er forklaret nogle ting som er fra forelæsning og relaterer sig også til Bogen (men) i form af noter og stikord.

# Binære Addition

- Vi starter med Binære Addition. Vi har cifre, hvor det ene var 0 eller 1.
- Det gode er, at man kan anvende den samme additionsmetode som man har lært i Gymnasiet.
- Man kan se på slide 3, at hvis man lægger den binærværdi 52 sammen med 46, så kan det ses at vi ender med resultatet 98.
- Man har et simpelt kredsløb, som kan lægge 2 bit sammen. Den kan generere hvad bit den bliver til resultatet.

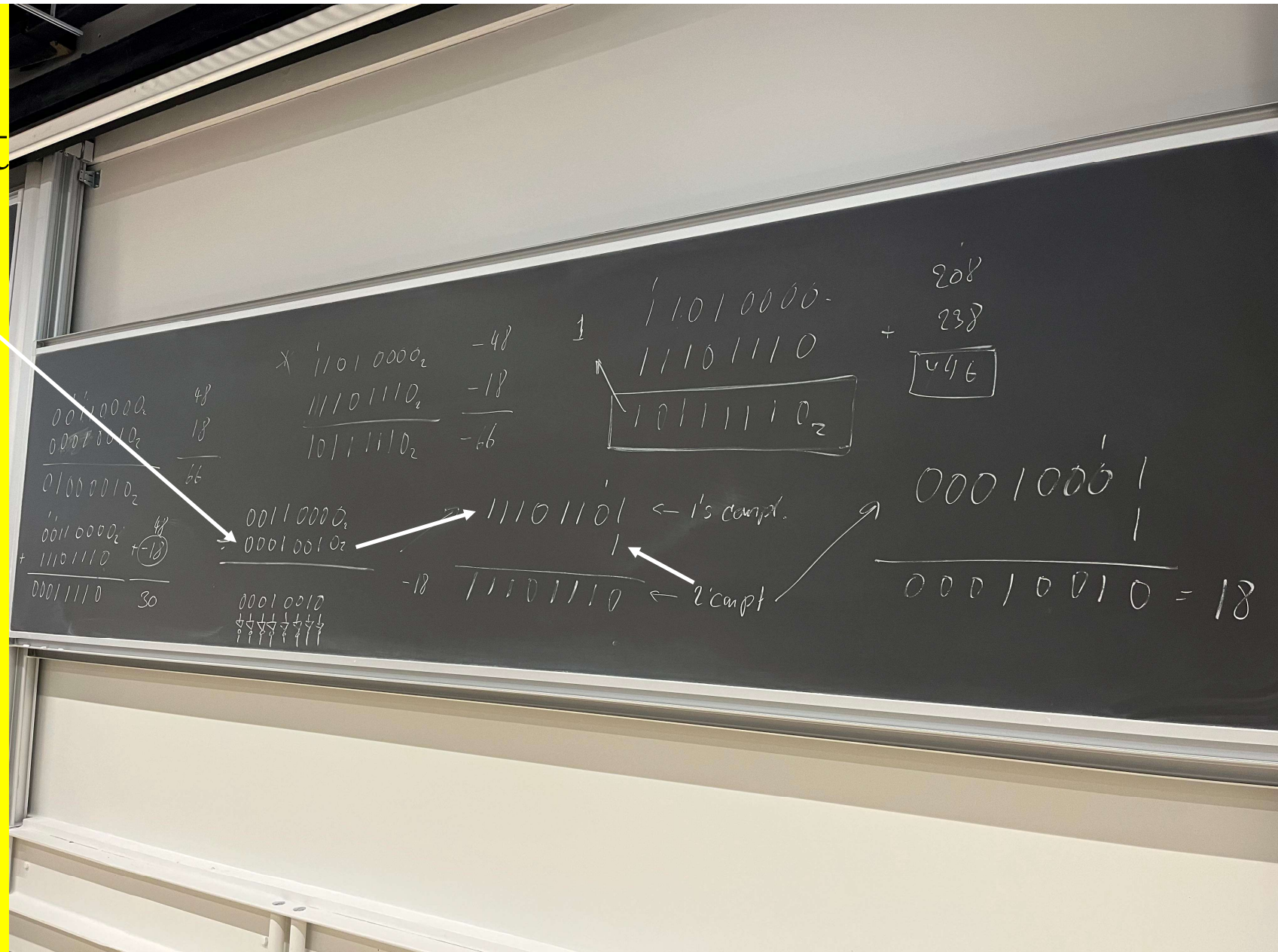
# Two's Complement

- Kredsløb på slide 5 ligger på vores computer, hvor man kan lægge to tal sammen.
- Det gode ved, at anvende 2's komplement er at man kan anvende det samme kredsløb til at lægge sammen. Man kan bare lægge det sammen, så er det  $00110000$  med  $00010010 = 01000010 = 1000010$  66
- Du har skrevet to's komplement udregning i din blå hæfte.
- Udtrykket 2's complement, det betyder at mindst betydende cifre er vægtet med to cifrede tal!!!
- Man har en kommunikation mellem main memory og CPU. I denne sammenhæng har man en adresse bus, databus og kontrolbus.
- Main Memory og CPU må ikke snakke i munden på hinanden, ved tilstedeværelse af kontrolbus.



# Complement

- 1's complement, hvor tallet vendes om.
- 2's complement hvor tallet lægges sammen med 1.



# Kommunikation mellem CPU og Main Memory

- Kigger man på slide 9, så kan det ses at main memory indeholder alle dataerne og adresserne. Hvorimod hvis man kigger over mod CPU'en, så kan det ses at selve CPU'en har den arbejde i at udføre handling og give ordre.
- Kontrol-bussen, har den arbejde i at afgive og modtage data hos CPU'en fra main memory.
  - EKS: I Kontrolbussen bliver der spurgt, hvem er klar og hvem er ikke klar?
- Adressen får at vide fra CPU'en, at den skal sætte et adresse op i main memory, hvor den efterfølgende går videre og siger til databussen, at den skal give noget data tilbage til CPU fra Memory.

# Hvad kan Main Memory Indeholde?

- Data i forskellige repræsentationer.
  - Maskininstruktioner (Program).
  - Hver af positioner i main memory, kan indeholde en byte.
  - Main memory er ligeglad med, hvad der ligger i selve byte, men oftest er det bare 1 og 0-taller som ligger i disse.
- 
- Den får bare, at vide hvad adresse den skal ligge på og hvad den skal sende til.
  - RAM = Random Access Memory.
- 
- De enkelte CPU, siger at den starter med program counter med specifik værdi fra eksempelvis adresse 0, hvor den starter med sin fetch execute cyklus.

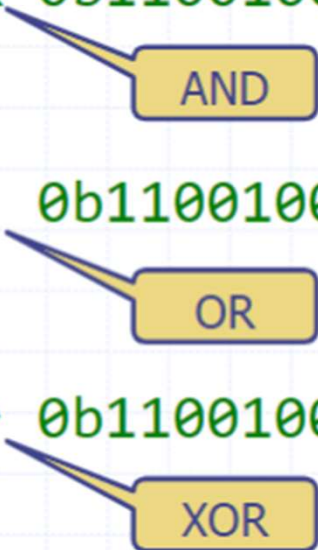
# Hvad kan Main Memory Indeholde?

- Spørg lige main memory? (gjort)
- Den som gemmer alle dataerne og adresserne.
- I virkeligheden kan vi summere ind og se, at den består af forskellige slags elektronik, altså hukommelses energier. Hvilken del var det i? De to mest betydende bit, er 14 og 15 går til dekoder. Denne dekode spørger om der er 00.
- A15 og A14 er 1, så er der chip select til datahukommelsen.
- Jo mere memory, der kommer desto mere bit kommer der i.
- VOLE er en CPU, hvor den kører i simulatoren. I Vole CPU'en er der 8 bit data. Vole-cpu'en har ingen ram.

# Python

- Man kan putte noget input inde i Python, og så kunne vi regne det på computeren. Hvor den så finder ud af hypotenusen og arealet af trekanten.

```
print(bin(0b10011010 & 0b11001001))  
# Prints '0b10001000'  
  
print(bin(0b10011010 | 0b11001001))  
# Prints '0b11011011'  
  
print(bin(0b10011010 ^ 0b11001001))  
# Prints '0b1010011'
```



The diagram illustrates three logical operations: AND, OR, and XOR. Each operation is shown in a code snippet, followed by a callout box with the operation name. The AND operation is shown with the code `print(bin(0b10011010 & 0b11001001))` and the output `# Prints '0b10001000'`. The OR operation is shown with the code `print(bin(0b10011010 | 0b11001001))` and the output `# Prints '0b11011011'`. The XOR operation is shown with the code `print(bin(0b10011010 ^ 0b11001001))` and the output `# Prints '0b1010011'`.

# Kontroller

- Det er ting, som vi sætter på vores computere. Den snakker med CPU'en på en bestemt måde. Computeren snakker gennem en port.
- Vi kan se, at vi har en Bus hvor der lyder kommunikation. Vi har eksempelvis en mus, hvor man får input fra musen. Det kan eksempelvis være, at dvd som kommunikere gennem bus.
- Der er eksempler på forskellige enheder, hvor man sætter ting på:
- Vi har datakommunikation. Det er meget vigtigt for en apparat at kommunikere, fordi det er den måde den kan udveksle noget data.
- Vi har også R232 som er historiske set brugt.
- Vi har eksempler på USB.
- USB, står allerførst for Universal Serial Bus.

# Kontroller

- Nu udvider vi vores computerbegreb, hvor vi sætter computerne ved siden af main memory.
- I/O siger om det er den ene eller det andet.
- Eksempelvis, hvis man tager en stykke ud af main memory, så kan man importere og lignende.
- cpu'en siger til main memory, og giv mig den næste instruktion. Eksempelvis hent noget data ved port nummer 4. Så vil cpu'en sætte pport 4 på adresse bussen og så kan den få det i et register i cpu'en. Fordi det er noget som den ønsker, at have og som ønsker at have senere hen.



# Kontroller

- DMA, det er ny aktiv spiller som kan bruge busserne og den kan også sende dataerne ud af bussen. Hvis CPU'en ønsker at have fat i noget, skal I smide det oppe i memory. Men problemer er, at man kan få problemer mellem indhentning af data mellem cpu og dma.
- Derfor laver man en handshaking, hvor der er koordinering af dataoverførsel mellem to eller flere enheder (software eller hardware).
- Man har protokoller, når man snakker mellem to forskellige mennesker. Der er også en protokol i internettet, hvordan computeren forsøger at snakke med browseren og laver sekvenserne og lignende. Dette sker gennem aftale.



# Computersystemer 3

Lavet af: Vivek Misra