Computersystemer 2

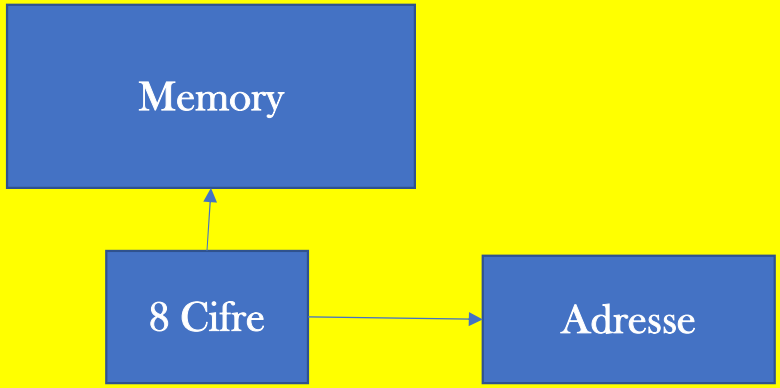
Lavet af Vivek ☺

Back to the drill.....

- Nu gennemgår vi Kapitel 2.
  - Her kommer vi til at gennemgå opbygning af Computeren.
  - Her kommer vi også til at snakke om de forskellige ting som eksempelvis.
    - Hvordan de forskellige USB-stik er forbundet til Computeren.
    - Forholdet mellem CPU og Main Memory.

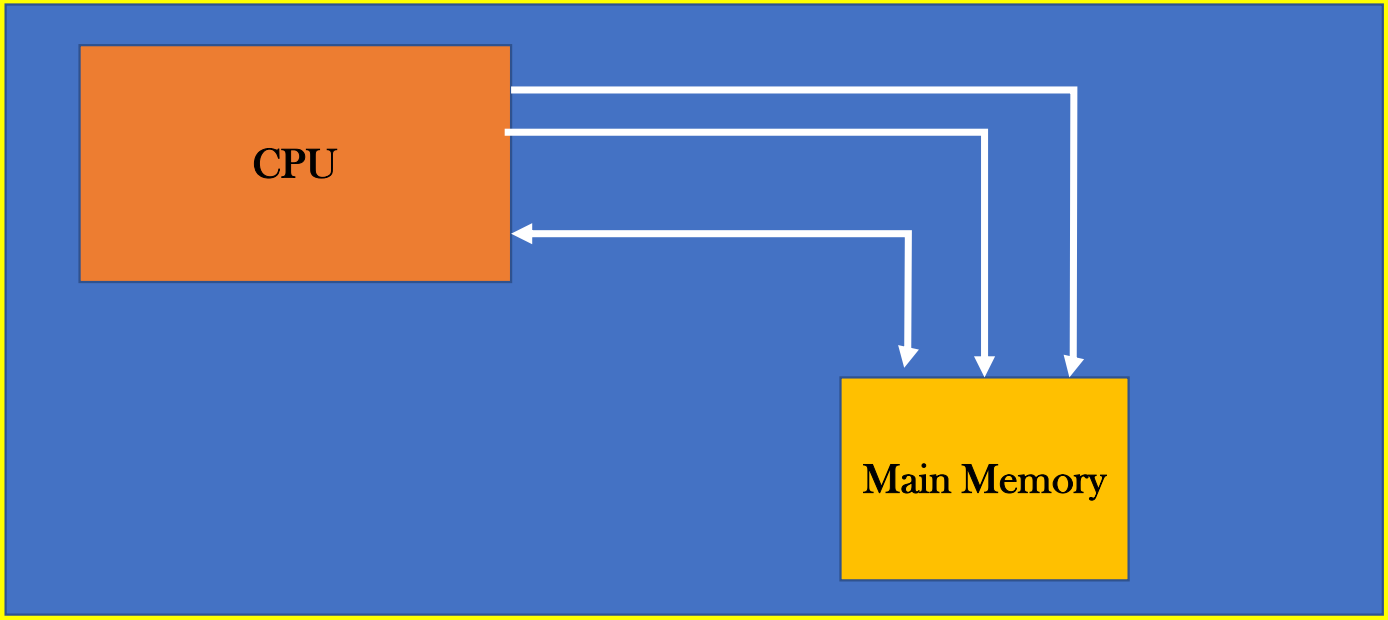
Opbygning af Bytes og Bites

- En bit og byte udgør lageret i en computer. På hver byte en memory kan der ligge 8 cifre, men vi kan ikke se hvad det betyder...?
- Derfor skal vi have en aftaler/adresse hvor vi siger at der skal ligges nogle 0'er og 1'er som skal tolkes som karakterer.



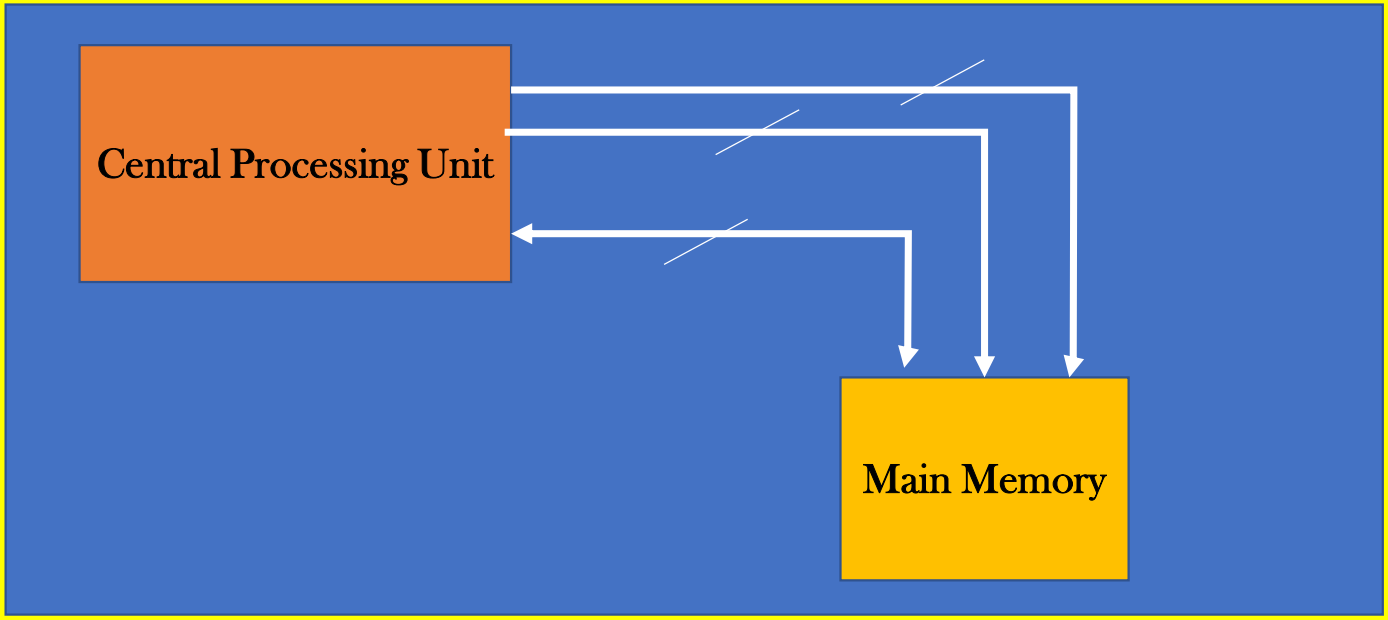
Opbygning af Computeren

- Hvis man kigger på opbygning af computeren, så kan vi se at vi har en main memory. CPU'en i denne sammenhæng er opbygget af XOR, AND or OR. Man kan også have flip flops som kan have main resultater.



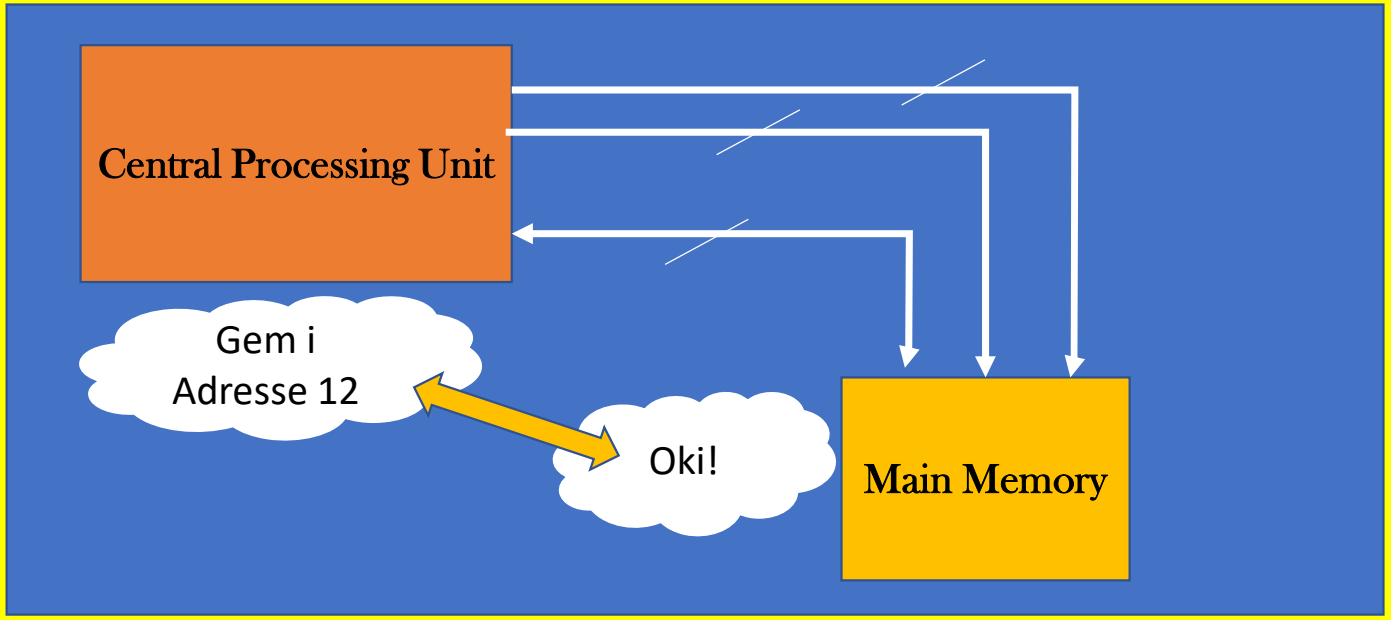
Slash og Busser

- Vi kan lave en computer, hvor vi kan forbinde memory og CPU'en som hedder BUS. **Slash / angiver antal ledninger**, fordi hver ledning kan indeholde to forskellige spændinger. CPU'en er hjernen bag det hele, så den fortæller hvad der skal ske.



Main Memorys Arbejde

- Memory skal svare på CPU'en, hvis CPU'en siger at der skal gemmes en memory på adresse 12.
- Vi skal bruge en memory til midlertidig opbevaring af data. Det er noget meget tæt elektronik, som tilgås hurtigt. Hvis man skal have fat i noget meget hurtigt.



Facts ☺

- Vi har forskellige typer af computer. I starten byggede man computere med radorører og transister. Nu har man Intel og lignende hvor man printede ting i forskellige størrelse med god strømforsyninger.

- Man sagde et tidspunkt i 71, hvor man lavede SPU'en i et chip. Nu har man en CPU i en chip.

- Man mener også at mikrokontroller er et computer i et chip.

Inde i CPU’en og ALU’en

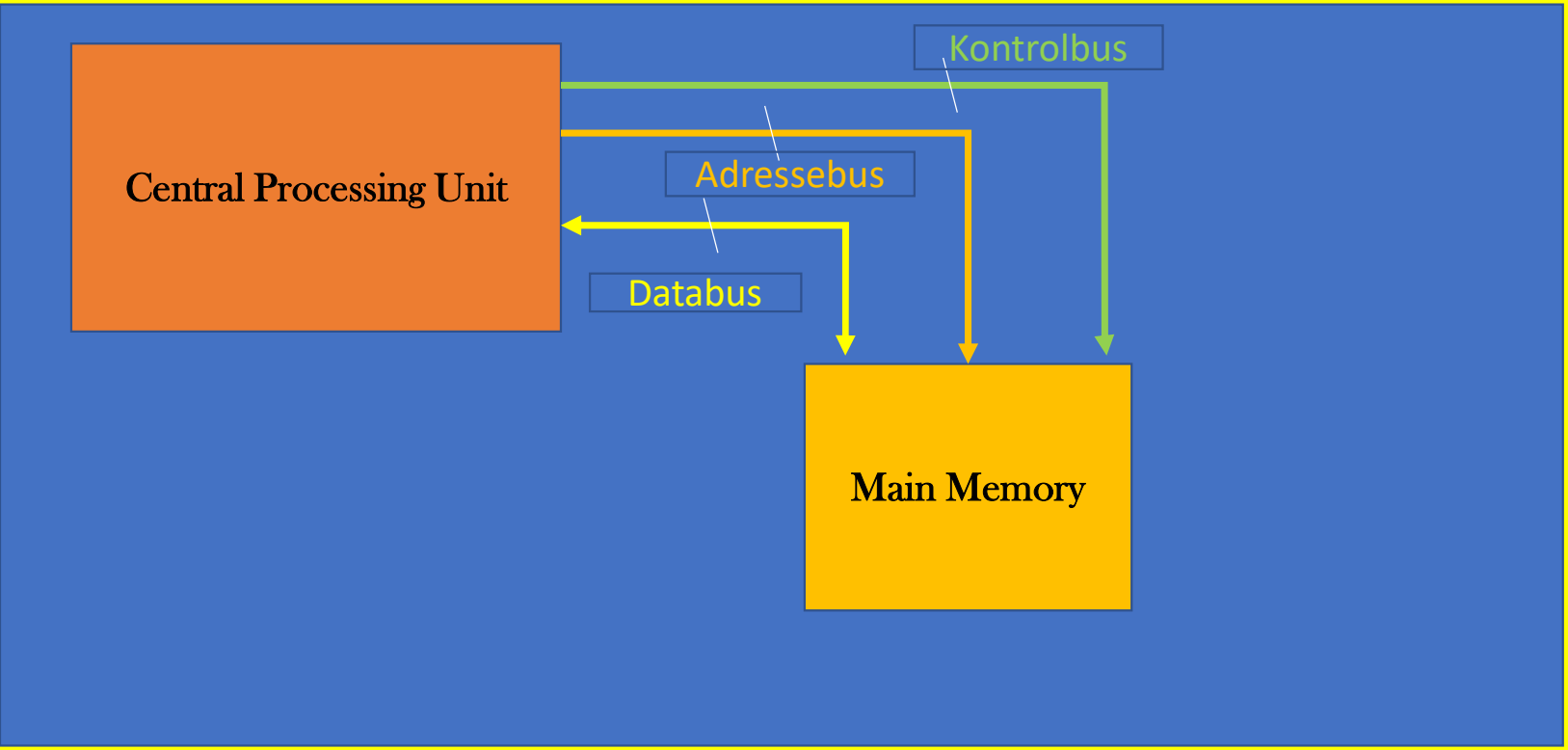
- Der er 3 tinge inde i CPU'en og det er virkeligheden også noget memory. Nogen gange er vi benævnt med bogstaver med ABCD og nogengange med 0123 - afhængig af CPU’en.
- Man har også ALU, som er regnedelen af computeren og udfører addition, subtraktion og multiplikation. Den tager input fra to registrere 1/0 og så laver den sin regne operation, hvor den giver sit input 2 eller 3. register.
- Den tredje ting er kaldt for CU, som holder styr på hvad der sker og hvilken sekvens det sker i.

3 Busser / 3 Muskater ☺

- Hvis man kigger hvordan CPU'en snakker med memory. Det er forbindelse med en ledning, hvor man kan dele det op i dele busser.
- Man har en data bus, adresse bus og kontrol bussen. De har pile som går fra CPU, som er styret af CPU.
- Hvorimod pile som er gået fra main memory og over til CPU.
  - KIG NÆSTE SLIDE

- Det er memory's indhold at tage imod funktionsarbejde. Kontrolbussen har to funktioner, hvor det er enten læse eller skrive og hvornår det kan gøre det.

3 Busser / 3 Muskater ☺



Adressebussen er den som lokkere værdierne ved Main Memorys adresse.

Kontrolbussen er den som kontrollere flowet.

Databussen er den som kontrollere hvilke data går igennem.

3 Bussernes Egenskaber ☺

- Den ene siger, at det er read og write. Og den anden siger, at nu er vi klar.
- 1 bit til angivelse af READ eller WRITE.
- 1 BIT TIL AT ANGIVE TIMING I DATAOVERFØRSLEN.

- Der skal være 2 ledninger i kontrolbussen, men i addressen er det afhænget af hvor meget plads i memory man vil gerne have fat i. Det er her, hvor vole har 8 bit. Det fortæller om, hvor meget memory man kan sætte det på.
- Databussen fortæller, hvor meget man kan overføre i 1 byte, 2 byte osv.

John von Neuman ☺


John von Neumann
(1903-1957)

• John von Neuman er ham som fandt på, at hvis man har en computer som er istand til at udføre forskellige instruktioner. Så i stedet for at lave computeren, så kan man lægge programmet ud i main memory og se om det er noget data eller main program.
• Kan vi ikke lægge nogle 0'er ELLER 1'er ude ved memory, så laves nye kombinationer af 0'er og 1'er. Det er også hele idéen med at se på computeren nu.

• Hvis en program, kan blive kodet som bit så kan kontrol unit se og fortælle hvad den skal gøre ved at fortælle CPU'en specifikt i starten. Maskinkoderne er 0'erne og 1'erne, hvor man i sample kunne se hvad loops og move var.

Main Memory's Indhold

- Når man kigger i memory main, så kan man ikke se hvad der er. Her kan man se, at hexodecimale 0,28 udefra binært tal. Hvordan ved Cpu'en, at den skal bruge en register datafunktion?
- CPU'en åbner den værdi som står i counter, og counteren er den værdi som står i adressen – hvilket også er kaldt for unit. Det program som bliver peget på, bliver til instruktionen.

- Kigger man på VOLE-Programmet i IntelliJ, så kan det ses at man kan lægge et nyt program og justere værdien og derved køre det.

CPU og dens opfattelse

- Når cpu'en ser vores program, er den som en maskineinstruktion.
- Maskinsprog er en beskrivelse af de mulige instruktioner, som cpu'en vil genkende som noget meningsløst. CPU'en vil reagerer på adskillige måder.

- Men spørgsmålet kommer nu, om man kan tage 2 approaches til at lave maskine sprog?

Approaches til Maskinesprog

- Her kan det ses, at man laver nogle simple instruktioner, hvor man har noget bite.

- Bite skal bruges, når CPU'en skal udføre noget. Noget af de instruktioner som man udfører data, skal have formålet i at flytte noget data. Disse hedder load, store data osv.

- Man skal have noget instruktioner, hvor man skal have noget at gøre med data. **Men man skal have datamanipulationen, hvor man undrer sig over hvad man skal gøre med data'et.** Man skal også have nogle kontrolinstruktioner, hvor man giver nogle nye værdier og udfører ting.

- **CPU'en er foreviget levende. Den skal bare blive ved med at gøre noget, ellers har den ikke noget eksistens. Eksempelvis hvis man starter word og slukker den bagefter, men Windows kører videre.**

Adding/Dividend Values Stored in Memory

❖Adding Values Stored in Memory

1.Få en af de værdier til at være tilføjet fra memory og placere det i et register.
2.Få et andet værdi til at være tilføjet fra memory og placere det i et andet register.

❖Dividend Values Stored in Memory

- Samme procedure i starten, hvor man henter det ene og det andet.

- Vole har 16 registrer og de er navngivet fra 0-F. Her kan man bruge det til at lægge midlertidige dataer ind.
- Man har også en instruktionsregister, så hvis den ændrer noget i main memory så ændrer den også det i instruktionsregister.

Vole Program og Adresser (VIGTIG)

❖Ses nærmere på Vole er de adskillige numre (0)'er kaldt for adresser.

❖Men spørgsmålet er bare, hvor mange bit er der i en adresse?

- Der er 8 bit i en adresse i en vole, hvorimod hvis man snakker om byte er det 256.
- Man har 16 bit til rådighed, når man laver en instruktion.
- Man har delt 16 bit op i to dele, den ene som er operationskode. Hvad skal cpu'en gøre? Og det resterende er operand, hvor den fortæller hvad den skal gøre tilbage.

- Vole har 16 registrer, hvor den har en program counter som har 8 bit og som skal indhente instruktioner.
- OBS: Det er 8 bit databusser.

Vole Program og Adresser (VIGTIG)

- Hvis signalet hedder Read/, så er det ikke read. Hvorimod hvis den har en streg over sig, så er det Write. Her i tilfældet kan det ses, at det bliver i modsat par. Så ikke read, giver en write.

- I instruktioner, kan den rotate som betyder at den kan kan skifte plads, sammen med at den kan lave load, store, move, add og or osv.

Program Eksekvering

- Kontrolleret af to særlige formål registrer:
- Instruktionsregister
    - Holder nuværende instruktion
- Program tæller
    - Holder addresse af næsten instruktion
-
- Maskinecyklus: Gentager 3 trin:
- Fetch:
- Decode:
- Execute:

Arkimetriske Logiske Instruktioner

- Man kunne også se, at man har logiske operationer som er AND, OR og XOR.
- Dette er brugt til mask som operand.
- Rotation og Skift Operation
- Cirkulært skift, logisk skift, arkimetriske skift.
- Det er ikke dem som vi plejer, at bruge i normal programmering.
- CPU'en har en anden måde at se AND på, og det er ved at lave to AND sammen som den plejer at gøre.

Arkimetriske Logiske Instruktioner

- Det man kan bruge til er, at man eksempelvis bruger det i masking, hvor man har en række bit og hvor man ønsker at tænde en lampe. Så hvis man skriver en på en plads, så er det 8 bit af gangen. I stedet for det kan man anvende en OR, så man kan sørge for at tænde bit 2 og anvende en maske.
- Hvis man havde 8 kontakter og ville undersøge om bit 2 var tændt eller slukket. Så kunne man AND, og hvis resultatet bliver 0 så betyder det at det ikke var slukket. Og hvis resultatet er anderledes, kan man udvælge den kontakte man vil gerne undersøge om.

- Rotating the bit patterns 0x65
- Den rykker bit pladserne ud til højre, så den sidste tal står først i den nye række.

Computersystemer 2

Lavet af Vivek ☺

Håber I kunne få noget teoretisk ud af det ☺