# Kontrol spørgsmål

#### C1. Binary numbers

How are integers encoded?

Heltal kodes som binære tal. Det kan enten være som signed integers (kan være negative) eller som unsigned integers (kun positive). Ved signed er det den højeste bit, der koder tegnet og bestemmer, om tallet er positivt eller negativt (“1” = “-“).

Eksempel på heltal, der skal kodes om kan være tallet 37 = 2(5)+2(2)+2(0) = 100101.

#### C2. Architecture

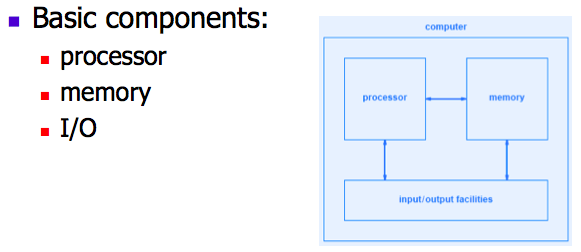
What is the Von Neumann architecture and what is its bottleneck?

Hans desgin går på fire dele**: ALU (A**rithmatic**L**ogic**U**nit)**, Control Unit, Memory and Input/output**.

**Von Neumanns** ide bygger kun på **én memory** (og bus) for **både data lagring og program lagring**.

Dette giver problemer da man **ikke kan hente instruktioner og data på samme tid**. En **løsning** for dette, er at tilføje en hurtigt **cache hukommelse**.

Grundlæggende koncept - programstyret. (John Von Neumann - matematiker). Modellen passer til de fleste moderne processorer. Grundlæggende komponenter: processor, hukommelse, I / O.



#### C3. Microcode

What are microcodes and macrocodes and why do we have both?

**Microcode** er det **interne sprog I CPUen**. **Macrocode** er det **synlige maskinkode** (Programmers reference), som kan ses udad til. I CPUen er der en **microcontroller** som **afkoder** og **udføre** **macro** **instruktionen**.

#### C4. Instruction sets

Compare the CISC and RISC instruction sets.

***CISC – Complex Instruction Set Computing***:.

Denne har mange måder at adressere på og mange operationer. Instruktionerne skal kunne fortolkes.

Komplekse instruktioner kan blive nedbrudt til mindre dele og blive udført som en sekvens af mindre instruktioner.

***RISC – Reduced Instruction Set Computing:***

Dette sæt er færre instruktioner, som ikke behøver at blive fortolket. Dette giver en hurtigere udførelses tid pr. instruktion, men kræver flere instruktioner i forhold til CISC.

I dag er mange CPUer hybride på den måde at komplekse instruktioner er fortolket som i CISC og simple instruktioner er udført direkte.

Fordel ved RISC er at hardware kan optimeres meget pga. det begrænsede instruktionssæt, men kan så ved afvikling koste ekstra ved at de samme instruktioner skal udføres mange gange for at få det samme resultat, som ved en CISC.

#### C5. Assembly languages

Which types of instructions are available? Give an example of common programming pattern (pseudo-assembly code).

Assembly language er et ***lav niveau sprog*** der er meget tæt på maskinkode. Alternativt findes f.eks. høj niveau programmerings sprog som C# eller Java.

Hver erklæring producerer ***præcis*** én maskine instruktion.

Assembly sprog er hardware afhængig og OS orienteret. Hvert assembly sproget er tilknyttet en processor type.

***Eksempler er:***

ADD, SUB, MUL og DIV

***Eksempel på pseudo kode:***

***If*** student's grade is greater than or equal to 2

Print "passed"

***else***

Print "failed"

C6. Memory

What is data alignment and why is it important?

Hvis data ikke tilpasses (***misaligned***), så skal CPU’en arbejde ekstra for at få adgang til dataen, dette sker typisk ved at den skal loade to dele data, fjerne de bytes der ikke har interesse og forbinde dem.

Denne process sløver performancen ned og spilder CPU’en kraft bare for at få det rigtige information fra hukommelsen.

Derfor er det vigtigt som programmør at gemme informationen ordentlig, dette kan også betyde at vi bliver nødt til at indsætte tomme felter hvis en data fx kun fylder 3 byte for at undgå misalignment.

#### C7. Size units

Why is it important to use powers of 2?

Pga. det **binære talsystem**, **0 og 1**. Da der kun er **to muligheder** med binære tal, opløfter man det **antal af tal** pladser **ønsket *n* med to**. På denne måde kan man udregne hvor mange muligheder man har ved et vist antal *n*. Fx fire pladser regnes således: **24 = 16**. Altså er der **16 tal kombinationer** ved brug af **4 pladser**.

Et spørgsmål fra censor kunne være: kunne man så ikke anvende f.ske. 10-tals-systemer som vi mennesker bedre kan forstå og så få 10 gange så meget performance?

Svaret: nej for en elektrisk component som en **transistor** kan **kun** **have** **2 værdier**, **strøm/ikke-strøm**, repræsenteret ved 0 og 1, derfor er det binære system begrænsningen.

#### C8. Technology

Compare SRAM and DRAM technologies.

RAM er en fysisk lagerenhed hvor computeren lagrer de data den arbejder med her og nu.

***DRAM – Dynamic***

Lagrer hver bit i sin egen capacitor (transistor) og denne lagring skal have strøm for at kunne opretholde informationen.

Bruges i bærbare, stationære, konsoller og lignende…

***SRAM – Static***

Er statisk ram og behøver derfor ikke en jævn tilgang af strøm for at opretholder information. Hver bit er repræsenteret i 4 transistorer, der er koblet i par.

Bruges i form af f.eks. hukommelseskort til kamera, usb-lagerenheder og lignende.

#### C9. Memory

Explain the term memory hierarchy.

**Computerens hukommelse** er inddelt i **5 levels,** etsåkaldt **hieraki.**

* **CPU-registers** (cpu full speed).
* **Cache** (32kb to few megabytes).
* **Main memory** (from 16mb to 10 gigabytes).
* **Magnetic disk**.
* **Tapes and optical disks**.

Dette hieraki er den traditionelle måde at gemme data på. Den er baseret på tre vigtige faktorer:

1. Jo længere vi kommer ned i hierarkiet, **des længere tid tager det for CPUen at tilgå dataen.**
2. Man får **mere hukommelses plads**, jo længere ned man kommer i hierarkiet.
3. Man får **mere plads for pengene**, jo længere ned man kommer i hierarkiet.