## Opgave 1:

Formål og rolle af OS:

* Kunne udnytte pc’en fuldt ud, oversætter systemkald til maskinkode.
* Binder BIOS og Device drivers sammen, til et fælles interface, som betyder at et program kan køre på flere systemer.
* Give mulighed for portability.

CPU, main memory(RAM), secondary memory:

* CPU’en udfører beregninger.
* RAM holder den data som CPU’en har brug for, dette gøres fordi HDD’er er meget langsomme.
* Secondary Memory, er permanent hukommelse, her lægges filer der skal gemmes i længere tid end PC’en er tændt.

Interrupt og interrupt-vector:

* Interrupt stopper en process, da der er en anden process der skal bruge cpu tiden.
* Interupt-vector peger på den service routine som CPU’en skal køre i stedet.

System Call:

* Applikationers vej til hardwaren(Gennem OS, til BIOS til Hardware og herefter retur.)

BIOS:

* Basic Input/Output System.
* BIOS håndtere alt hardware i PC’en
* Den giver et fælles syn på hardware af samme type for operativsystemet.

CCP:

* Oversætter kommandoer til systemkald.
* Eksekvere programmer.
* CLI (CMD, Terminal).

Figur 1.1:

* Musens Controller sender signal til CPU’en.
* CPU’en kalder et interrupt.
* Interruptet bliver håndteret af device drivers(BIOS)
* Eventet bliver sendt til applikationen.
* Applikationen håndtere eventet og sender et kald til Device drivers.
* Device driver oversætter kaldet og sender det til video controlleren
* Video controlleren afspejler kaldet på skærm.

Figur A.1:

* Oversigt over computeren
* Data bliver sendt fra CPU og main memory til hinanden og andre enheder via BUSSEN.

## Øvelse 2:

File, Directory(Mapper)

* File directory indeholder metadata om filerne på harddisken.
* File lægger i file directory area.

Boot Area:

* Et reserveret område af HDD’en eller SSD’en hvor OS’ets eksekverbare fil er lokaliseret.

Sikkerhed af Filer udført af OS:

* Hver fil har et usernumber der korrespondere med den bruger som har rettigheder til at tilgå filens indhold.

Eksempler på sikkerhed i filer i windows:

* Brugerrettigheder.
* Skrivebeskyttelse.
* Windows Defender.

## Opgave 4:

Forskel på *program* og *proces:*

* Et program kan bestå af flere processer.
* En proces er en sekvens CPU’en kører.

Figur 3.5 (side 60 i [OS]):

* Det er en memory stack når man eksekvere et program.
* Lavst i hukommelsen ligger der interupt vectoren herefter kommer program headeren, der indeholder metadata om programmet, så kommer Main program executeable, det er der programmet starter med at eksekvere kommandoer derefter kommer overlay som kan være endte 1,2 eller 3 alt efter hvilken del af koden der eksekveres overlayet udskiftes undervejs. Fixed data er den del af programmet der aldrig ændres f.eks. string literals. Program stack ligger lige under BDOS og indeholder dynamiske variabler, og vokser sig mod fixed data. BDOS eller basic disk operation system som i bund og grund er dit OS.

*Shell*. Hvor mange *shells* kender du? Hvor mange har du anvendt?

* CMD(Windows)
* Bash(Unix Shell)
* Cisco IOS
* Terminal(MacOS)

*Heap* og *stack* ifm. (primær) memory:

* Heap bruges til fixed data. En anden udgave er Binary search tree(BST)
* Stack som bliver brugt til at holde de data der ændre sig under runtime.(FILO)

*Overlay*, formål og overordnet beskrivelse af operativsystemets rolle ifm. overlays:

* Hvis du ikke har plads i hukommelsen til alt din kode bruges der overlays til at håndtere det resterende kode.
* Overlays udskiftes undervejs hvis der er brug for det under runtime.
* OS’et står selv for at administrere overlays.
* Overlays indeholder blocks af kode der bliver anvendt på forskellige tidspunkter i programmet

Figur A.4 (side 497 i [OS]), herunder: *data* *cache*, *instruction cache*, *L1*, *L2* og *L3 cache:*

* Det er et opbevarings hierarki, øverst har man det hurtigste(Primary storage(Registers, Cache memory og RAM)), i midten ligger secondary storage(HDD/SSD og flash memory(BIOS)). Nederst har vi Tertiary storage som er alle flytbare drev f.eks ektern HDD, usb osv.
* L1 ligger i CPU’en.
* L2 og L3 ligger udenfor CPU’en
* De højere caches(L2 og L3) er hurtigere end de lave cahes(L1), men er også dyre og indeholder mindre data.

Forskel på *volatile* og *non-volatile* memory:

* Volatile forsvinder når du slukker pc’en(RAM)
* Non-volatile bliver liggene på pc’en(HDD’er , SSD’er osv.)

## Opgave 5

*Instruktions* *set*:

* I bund og grund maskinsprog.
* Det er implementeret på selve CPU’en.
* Kommandoer: Load, Add, Store, Compare, Jump If, JUMP, Out og IN.

*ALU*:

* Den laver beregninger og sammenligninger.
* Der er 2 ALU’er en integer ALU(Heltal) og en Floating point ALU(Kommatal)

*Instruction* *registers*:

* CPU’ens instruktioner indlæses en ad gangen i instruktion registers.
* Hver instruktion har sin egen unikke bitmønster.

*Program* *Counter* (instruction counter):

* Den indeholder adressen til næste instruktion.
* Den starter ved første program instruktion.
* Indeholder ofte også længden på den nuværende instruktion.(Længden er hvor meget hukommelse instruktionen fylder)

*Data* *registers* / *accumulator*:

* Holder dedikeret operander.
* I bund og grund den data CPU’en bruger på nuværende tidspunkt.

*Address* *registers*:

* De indeholder hukommelses adresse på operander og instruktion i main memory(Ram, register eller cache).

*Clock*, *time* *quantum* (aka time slice), *timer* *interrupt:*

* Clock administrere timing i CPU’en.
* Når clock register rammer 0, kaldes det er timer interrupt. og CPU tiden gives videre til en ny process.
* Time quantum er den mængde tid som OS tildeler program.

*System* *clock*:

* System Clock administrere timing i CPU’en.

Hvor mange *bits* er der i en *byte*? I et *word*?

* 4 bits i en nibble, 2 nibbles i en byte så der er 8 bits.
* Der alt fra en til 8 bytes i et word.

Hvad er forskellen på en *16*-, *32*- og 64-bits arkitektur for en CPU?

* En 16 bits CPU vil have en standard enhedsstørrelse på 16 bit.
* Forskellen er altså størrelsen på et word.
* 16 bits cpu’er har 2 bytes i et word.
* 32 bits cpu’er har 4 bytes i et word.
* 64 bits cpu’er har 8 bytes i et word.