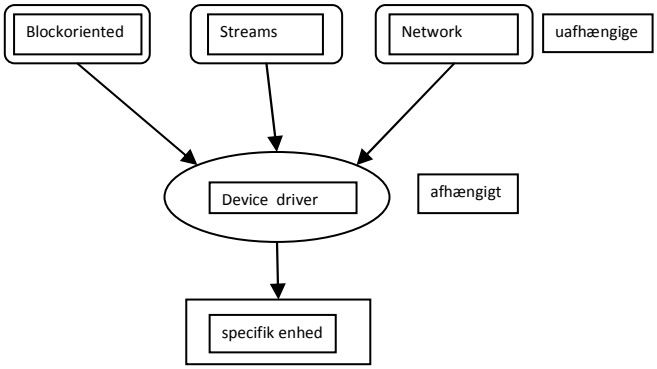
9. Input/Output

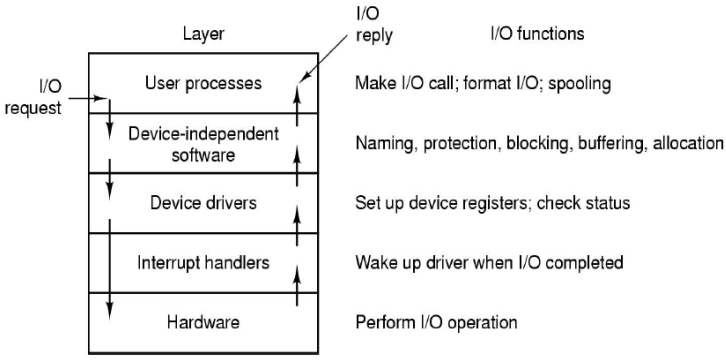
I/O Systemet



Vi har tre vigtige opgaver I/O systemet skal tage sig af:

1. **Abstraherer implementation** af de enkelte enheder væk. Device driveren kender enheden, omsætter til en fælles sprog.
2. **Effektiv udnyttelse** af I/O enheder.
3. **Understøtte deling** af I/O enheder. Nogle kan deles,

   andre kan ikke (f.eks. ingen deling af printer under printjob – stream problem).

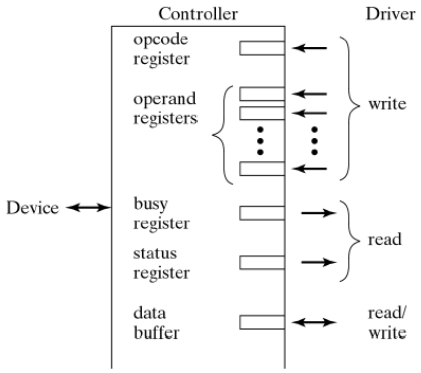


I/O systemet kan deles op i device dependant og independant software, hhv controllere og drivere.

Device drivers: kommunikerer med device controlleren via hardware registrere.

I/O med Polling

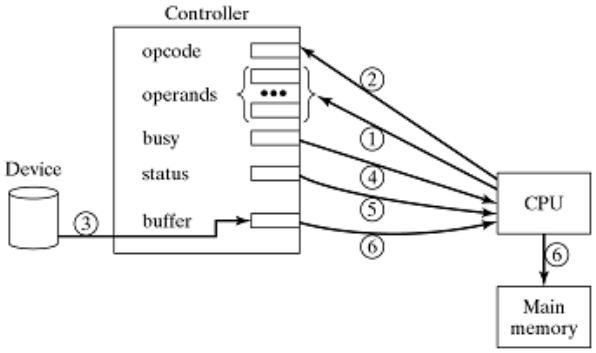
Programmed I/O med Polling: CPU holder hele tiden øje med device, ved at checke busy flag. Spild CPU kræfter. Gør det svært med multiprogramming.



I/O med Interrupts

Programmed I/O med interrupts: Suspenderer proces indtil device er klar, sender interrupt. Så bliver tråden aktiv igen. Ingen busy wait. Prob: Håndtering af interrupt tager tid, CPU afbrydes for hver byte der kommunikeres.

DMA (direct memory access): CPU er for langsom til at flytte data fra device til main memory. CPU siger til controller at der skal overføres data. DMA står for den egentlige overførsel. Problem: cycle stealing (DMA og CPU vil ha MM).



Memory-mapping

Vi skal tilgå hardware registre i enhederne.

1. Memory mapped (udvid main memory med adresser til I/O device registre). Fordele: ingen specielle maskininstruktioner, nem adgang til device registre fra højniveausprog
2. Eksplicit: angiv hvilket device man vi kigge på. Kræver specielle maskininstruktioner til I/O.

Specielle I/O instruktioner

```
IO_STORE CPU_REG,
         DEV_NO,
         DEV_REG

IO_GET DEV_NO,
       DEV_REG,
       CPU_REG
```

Memory-mapped I/O

```
MOV CPU_REG, ADDR
MOV ADDR, CPU_REG
```