1 a)

Et digitalt signal er et signal som kun kan inneholde et endelig antall ulike signalverdier. Det vil si at signalverdien for eksempel enten er 0,1,3,4 eller 5 osv. Et signal som bare opererer ut ifra to signal verdier kalles et binærsignal. Det er ofte 0 og 1.

Et analogt signal er signal som har kontinuerlige/vedvarende verdier. Det vil i praksis si at signalet kan være hvilken som helst verdi mellom en gitt maks-verdi og en min-verdi.

Et eksempel på en sensor som gir et digitalt signal kan være en bryter som enten er av eller på, og gir ut digitale verdier basert deretter. Et eksempel på en sensor som gir et analogt signal kan være temperaturmåler. Nesten alle fysiske systemer har for øvrig en analog oppførsel.

b)

Datamaskiner opererer etter digitale signaler, den bruker binære verdier for å representere tall, informasjon og symboler. Fordi dette er en digital maskin som kun opererer med et gitt antall symboler må analoge signaler digitaliserer for datamaskinen skal kunne benytte signalet.

Dette gjøres ved at man velger et punkt i rekken av kontinuerlige verdier og bestemmer verdien til signalet i dette punktet. (Kalles å kvantisere signalet i tid og verdi) Samlebetegnelsen for denne metoden er Sampling eller punktprøving. Den samplene verdien kan for øvrig kun være en av et gitt antall ulike verdier. Verdien må også være et heltall.

c)

Klokkesignalet er en periodisk firkantsvingning, det vil si at det er en puls som er periodisk, og signalet gjentas etter en gitt T- verdi (T). Dette kalles periodetiden. Dette klokkesignalet brukes i all hovedsak av datamaskiner for å kunne tidsstyre operasjoner. Det bestemmer da hastigheten en datamaskiner opererer med. Frekvensen til klokkesignalet (klokkefrekvensen) bestemmes ved $F = \frac{1}{T}$

I datamaskiner (PC-er) oppgis ofte klokkefrekvensen. PC-en som dette dokumentet skrives på har for eksempel en CPU med klokkefrekvens på 1.6 Ghz. Klokkefrekvensen benyttes for å synkronisere alle operasjoner i datamaskinen, slik at de kan skje til riktig tidspunkt i forhold til hverandre. Det er viktig å påpeke at klokkesignalet ikke inneholder informasjon, men bestemmer heller hvor hurtig en bit, som inneholder faktisk informasjon, kan slås av og på.

Audrino mikroprosessoren har en klokkefrekvens på 16 Mhz

$$F = \frac{1}{T} \ , \ T = \frac{1}{F}$$

$$T = \frac{1}{16\,000\,000}$$

$$T = 6.25 * 10^{-8}\ S$$

$$T = \ \ 62.5 * 10^{-9}\ S$$

$$T = 62.5\ \ ns$$

Vi ser av utregningene at pulstiden til Arduino- mikroprosessoren sin klokkefrekvens er på 62.5 nanosekunder.

d)

Et tidsdiagram en graf som består av digitale signaler (på bølgeform). Grafen vise hvordan de ulike signalene forandrer seg over tid, men også hvordan de forandrer seg i forhold til hverandre. Gjennom å studere et tidsdiagram kan man fastslå de ulike stadiene til hvert signal (0/1, High/low) til enhver tid.

e)

Definisjonen av data er grupper av bit som angir ulik type av informasjon. I en PC, og for øvrig også andre datasystemer må denne dataen overføres mellom enheter. Det kan være mellom enheter innad i datasystemet med også fra datasystemet til eksterne enheter. Dataen kan enten overføres serielt eller parallelt.

I en seriell overføring blir et og et bit overført over en linje. Det vil si at et bit blir overført i løpet av et tidsintervall. Etter dette blir neste bit overført, slik fortsetter det til alle bits i gruppen er overført. (all informasjonen er overført). Overføring av data gjennom USB (til eller fra en ekstern enhet) er en typisk seriell overføring.

I en parallell overføring har hver bit sin egen linje, noe som betyr at alle bits i gruppen blir overført samtidig. Hele «gruppen» av informasjon blir overført over et tidsintervall. Overføring av data mellom komponenter innad i et datasystem er en typisk parallell overføring. En fordel med denne metoden sammenliknet med en seriell overføring er at man kan overføre mye mer informasjon over et kortere tidsintervall.

2 a)

Et posisjonsbasert system går ut på at plasseringen av sifferet angir «vekten» til tallet. Man kan tenke seg at det avgjør hvor mye tallet «er verdt» i forhold til de andre plasseringene. Plassen til tallet angir hvilken verdi tallet har.

Det binære tallsystemet har to lovlige verdier, nemlig 0 og 1. For å for eksempel angi et binært tall benytter man posisjonssystemet for å bestemme verdier. Grunntallet er antall lovlige verdier, nemlig 2. Vekten til tallet blir doblet fra en posisjon til neste, eksempelvis 2^2 og 2^3 som er 4 og 8.

0101 1001

$$1 * 2^0 + 0 * 2^1 + 0 * 2^2 + 1 * 2^3 + \ 1 * 2^4 + 0 * 2^5 + 1 * 2^6 + \ 0 * 2^7$$

$$1 + 0 + 0 + 8 + 16 + \ 0 + 64 + \ 0 = 89$$

$$\underline{0101\ 1001_2 = 89_{10}}$$

b)

$1010\ 011_2$

$$1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 0 * 2^3 + 1 * 2^4 + 0 * 2^5 + 1 * 2^6$$

$$1 + 2 + 0 + 0 + 16 + 0 + 64 = 83$$

$$\underline{1010\ 011_2 = 83_{10}}$$

c)

43_{10}

Utfører heltalsdivisjon med to, og resten etter divisjonen utgjør de forskjellige bit-ene.

$$\frac{43}{2} = 21 + 1, \qquad rest = 1$$

$$\frac{21}{2} = 10 \ + 1, \qquad rest = 1$$

$$\frac{10}{2} = 5 + 0, \qquad rest = 0$$

$\frac{5}{2} = 2 + 1, \qquad rest = 1$

$\frac{2}{2} = 1 + 0, \qquad rest = 0$

$\frac{1}{2} = 0 + 1, \qquad rest = 1$

Leser av restene etter divisjonene fra bunn og opp

$\underline{43_{10} = 101011_2}$

d)

En datamaskin bruker det binære tallsystemet, altså 0 og 1 til å holde på informasjon. Binære tall fungerer godt til å holde på positive heltall, og kan lett omgjøres. Toer kompliment er dermed en av metodene en datamaskin bruker for å holde på negative heltall. Toer komplimentet bygger på den første metoden for å holde på negative heltall, nemlig enerkompimentet. Denne metoden går ut på å snu hver bit i et binært tall, og si at dette er den negative versjonen av heltallet. Toer-komplimentet er dette, samt å legge til 1.

Av forrige oppgave vet vi at $43_{10} = 101001_2$

Bruker metoden for toer-kompliment:

$-43_{10} = 010110_2 \ \ (ener - kompliment)$

$0101110 + 1 = 0101111 \ (legger \ til \ en)$

$\underline{\underline{-43_{10} = 0101111_2}}$

e)

MSB og LSB i et binært tall forteller noe om hvilket bit som har størst eller minst verdi. MSB står for «Most significant bit» og LSB for «least significant bit

I det binære tallet: **1010010,** vil tallet helt til venstre være MSB, fordi posisjonssystemet tilsier at dette tallet har størst verdi. Tallet helt til høyre er LSB, fordi posisjonssystemet tilsier at dette tallet har minst verdi.

f)

10 + 18

$10_2 = 0000\ 1010$

$18_2 = 0001\ 0010$

$$+\frac{0000\ 1010}{0001\ 0010}$$

$= 0001\ 1100$

g)

1.

Datamaskinen løser **29-15** ved å omforme den negative delen til sitt 2-er kompliment. Løses deretter slik **29 + (-15)**

15_{10} = $0000\ 1111_2$

29_{10} = $0001\ 1101_2$

Bruker toer-kompliment til å finne -15

1-er kompliment: 1111 0000

Legger så til en for å finne to-er kompliment

-15= 1111 0001

Kan deretter legge sammen tallene

$0001\ 1101 +$

$1111\ 0001$

$= 0000\ 1110$

$0000\ 1110_2 = 14_{10}$

Det binære tallet tilsvarer 14, noe vi ser stemmer med det opprinnelige regnestykket.

2.

-17 – 18 regnes ut på samme måte som ovenfor, men forskjellen er her at vi må bruke toer-komplimentet på begge tallene, slik at vi kan regne **(-17) + (-18)**

17_{10} = $0001\ 0001_2$

18_{10} = $0001\ 0010_2$

-17= 1110 1110+ 1 = 1110 1111

-18= 1110 1101 + 1 = 111 0 1110

$1110\ 1111 +$

$1110\ 1110$

$\underline{= 1101\ 1101}$

h)

Overflow i heltallsberegninger er et problem som kan oppstå når man sumerer enten to positive eller to negative tall. Hvis fortegnet etter en slik regneoperasjon ikke er det fortegnet som er forventet indikerer det at vi har fått en overflow feil. Man benytter for få bit til å holde tallet, og det blir for stort. Ofte blir slike feil varslet om, men ikke en sikkerhet. I mange programmeringsspråk må man benytte datatyper som har nok plass i minne til å holde på heltallet man skal benytte, for å unngå slike feil.

55 + 85

$\mathbf{55_{10} = 0011\ 0111_2}$

$\mathbf{85_{10} = 0101\ 0101_2}$

$\mathbf{0011\ 0111} +$

$\mathbf{0101\ 0101}$

$\mathbf{= 11000\ 1100}$

$\mathbf{11000\ 1100_2 = 396}$

Vi ser at vi får et tall som ikke stemmer overens med addisjonen vi skulle utføre, og det har oppstått en overflow-feil.

i)

$\mathbf{1100\ 1101_2}$

Deler opp tallet i grupper på fire bits

Første gruppe: $1100:\ 0 * 2^0 +\ 0 * 2^1 + 1 * 2^3\ + 1 * 2^3 = 12 = C$

Andre gruppe: $1101: 1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 = 13 = D$

$1100\ 1101_2 = CD_{16}$

j)

$\mathbf{4FA1_{16}}$

$= 1 * 16^0\ + A * 16^1 + F * 16^2 + 4 * 16^3 = 16 + 10 * 16 + 15 * 256 + 4 * 4096 = 20400_{10}$

$\underline{4FA1_{16} = 20400_2}$

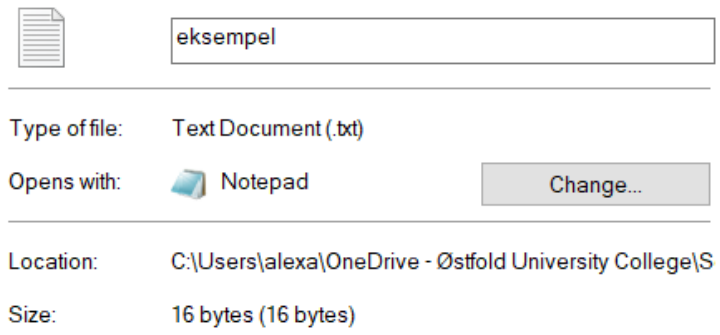
k)

ASCll- kode er viktig fordi det utgjør en standard som alle datamaskiner kan følge. Når en datamaskin skal lagre symboler som bokstaver, tall og spesialtegn er det nødvendig med en instruksjon som kan følge, hvis ikke kan ulike datamaskiner ende opp med å lagre de på en rekke forskjellige måter, og problemet blir da når man skal utveksle/tolke tekst mellom datamaskiner. Hvis disse maskinene lagrer symboler, bokstaver etc. på forskjellige måter vil det bli umulig å kunne tolke tekst som blir sendt mellom dem. Slike standarder er derfor viktig slik at informasjon kan tolkes likt på tvers av ulike datasystemer og språk.

Datateknikk 2020

Skal man benytte ASCll-kode for å lagre dette ordet på en fil vil hvert symbol lagres med en binær-kode som inneholder åtte bit, eller en byte. Bokstaven D lagres for eksempel med hjelp av det binære tallet «1000 0100». Det finnes også en desimalrepresentasjon av de ulike tegnene på samme måte som den binære representasjonen. Denne standarden brukte åtte bit-verdier for å lagre tegnene noe som gav mulighet til å holde 256 tegn i standarden. Denne standarden er ellers utdatert og i dag inneholder standarden nesten alle tegn som er brukt hvilket som helst språk vi har.

Vi ser at ordet Datateknikk 2020 inneholder 16 tegn (med mellomrommet), noe som tilsvarer 16 byte. En fil med dette ordet vil dermed ha en filstørrelse på 16 byte.



I)

For å kunne gjøre beregninger med desimaltall på en effektiv måte forholder datamaskinene seg til en valgt standard. Alle desimaltall kan oppgis på eksponentform, der vi har en desimalverdi og en tier-potens. 23.45 kan for eksempel skrives som $0.2345 * 10^2$.

I en datamaskin lagres desimaltall på en liknende måte, en modell på dette ser slik ut:

$s * Mantisse * 2^{eksponent}$. Her representerer s fortegnet til tallet, Mantissen representerer den desimale delen. Eksponenten representerer følgelig eksponenten. Når datamaskinen gjør om desimaldelen til binær form hender det at dette tallet blir større enn det standarden kan holde og en del av det binære tallet må kuttes vekk. Den desimale delen blir da i praksis avrundet, og gir bare et gitt antall desimaler vi kan stole på.

Standardene som blir benyttet er:

Single Precision	32 bit
Double Precision	64 bit
Extended Precision	80 bit
Quad Precision	128 bit

De forskjellige standardene gir en nøyaktighet av desimaler. En av grunnene til at flere formater benyttes kan være at man noen ganger ikke trenger høy nøyaktighet av desimaler og kan bruke et format som tar mindre minne og omvendt hvis man trenger mer nøyaktighet.

Bibliography

Floyd, T. L. (2015). *Digital Fundamentals.* Edinburgh: Pearson.

Forelesningsnotater