Teckenkodning

- **7-bitars ASCII:** Har plats för 128 olika tecken. Innehåller bara jenkarnas bokstäver och favvotecken.
- **Latin-1 (eller ISO-8859-1)** innehåller 8-bitar och har således plats för 256 tecken. Innehåller västerländska språk (dock ej Ryska el Hebreiska).
- **UTF-X** innehåller en **BOM** som indikerar vilken vilken av UTF-8,16,32 som använts och ifall det är big endian eller little endian. BE:00 46, LE: 46 00.

BOM	**Teckenkodning**
FE FF	UTF-16BE
FF FE	UTF-16LE
EF BB FF	UTF-8

  I UTF-32XX lägger man antingen till 00 00 framför eller bakom :)

- *Unicode* är en förteckning över hur olika tecken motsvaras av en teckenkod. UTF-X är standerden med vilket man bestämmer hur tecken ska lagras i bytes. (ETTOOR OCH NOLLOOOR).

Moores lag säger att antalet transistorer på ett chip fördubblas ungefär vartannat år. Vi börjar närma oss den teoretiska gränsen för hur små kretsar kan bli. T.ex. kan inte en detalj på ett chip vara mindre än en atom. Just nu (2009-10-09) ligger de minsta detaljerna på 80 atomers bredd.

Binär och Hexadecimal matte

- **Tvåkomplementsform:** är ett sätt att representera negativa tal i det binära talsystemet. För att skriva talet -1 på tvåkomplementsform går vi från **00000001** till **11111111**. Genom att invertera alla siffror (0 blir 1 och tvärtom) sen lägger vi till 1 på slutet.
- **Mellan binär och hexadecimal form:** Talet 11010100 delas upp i grupper om fyra à: 1101 0100 (i det decimala talsystemet: 13 och 4) 13 i det hexadecimala talssystemet är D och 4 är 4.

  Således är $11010100_{bi} = D4_{hex} = 212_{dec}$
- **Från decimal till binär form:** Gå från närmsta 2^x som får plats i talet, dra av 2^x och testa 2^{x-n}. Skriv en sanningstabell för ifall de får plats eller ej som tabellen nedan visar:

Decimalt	2^x	**Får plats**
78	128 (2^7)	0
78	64 (2^6)	1
14	32 (2^5)	0
14	16 (2^4)	0
14	8 (2^3)	1
6	4 (2^2)	1
2	2 (2^1)	1
0	1 (2^0)	0
Binär form:		(0)1001110

Den binära formen blir alltså sanningstabellen uppifrån och ner.

Boolesk algebra

- **De Morgans lag:** För att negera ett Bollesk uttryck byter man ut OCH mot ELLER (eller vice v) samt negerar varje enskild term". Tänk att $\wedge$ ser ut som ett a...därav betyder det AND. I praktiken fungerar De Morgans lag enligt följande:

  $\neg(x \vee y) = \neg x \wedge \neg y$
  $\neg(x \wedge y) = \neg x \vee \neg y$

  Detta gäller oavsett hur många tal som står innanför parenteserna:

  $\neg(a \vee b... \vee x) = \neg a \wedge \neg b... \wedge \neg x$
  $\neg(a \wedge b... \wedge x) = \neg a \vee \neg b... \vee \neg x$

- **Från sanningstabell till Booleskt uttryck:** Vi kan skriva en Boolesk funktion från en sanningstabell. För varje värde där funktionen är sann skriver vi ett uttryck och sammaskriver dem med hjälp av ett $\vee$-tecken.

x	y	z	f	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\neg x \wedge y \wedge \neg z$
0	1	1	0	
1	0	0	1	$x \wedge \neg y \wedge \neg z$
1	0	1	0	
1	1	0	1	$x \wedge y \wedge \neg z$
1	1	1	0	

  Vi får då genom ovanstående regel uttrycket:

  $f = (\neg x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge y \wedge \neg z)$

  Detta uttryck kan i sin tur skrivas om enligt reglerna som kommer stå i formelsamlingen på provet. Första delen av förenklingen är att bryta ut det som är gemensamt för alla parenteser. D.v.s. $\neg z$. Eftersom detta är så enkelt tänker jag inte gå igenom det.