TK1100

FORELESNING 0x01 BINHEX

1) Binær koding

a) Hvilke ulike måter er det vist i forelesning at man kan representere karakterene i ASCII tabellen?

Svar: Desimaltall (base 10), Hexadesimal (base 16), Oktaltall (base 8) og Html-enkoding.

b) Hvilke hovedtyper data, eller informasjon, behandler en datamaskin?

- Numerisk
- Karakterbasert(alfanumerisk)
- Visuell
- Audio
- Instruksjoner
-

c) Fyll inn tabellen under.

2^0 =	1
2^1 =	2
2^2 =	4
2^3 =	8
2^4 =	16
2^5 =	32
2^6 =	64
2^7 =	128
2^8 =	256
2^9 =	512
2^{10} =	1024

d) Hvor mange bitmønstre (forskjellige kominasjoner av 0 og 1) kan du lage med 8 bits?

- 8, fordi det er åtte bits
- 16, fordi det er åtte bits som hver kan ha verdien 0 eller 1, som gir 8*2=16 kombinasjoner
- 64, fordi 8*8 = 64
- **256, fordi hver bit kan være 1 eller 0, da er det med 8 bit mulig å lage 2*2*2*2*2*2*2*2 = 28 = 256 kombinasjoner**
- 1024, fordi det er en Ki

e) Hva har det binære tallsystemet som base?

Svar: Base 2

f) Hvorfor bruker vi presisjon? Nevn fire vanlige presisjoner.

Datamaskiner har begrensninger på hvor mange bit som kan lagres i f.eks. minnet. Når vi gjør utregninger i binære tall må vi derfor ha et konsept av hvor mange bit vi må oppgi. Dette kan også føre til at når vi gjør utregninger vil enkelte bit havne utenfor kapasiteten i minnet. Disse såkalte «overflow»-bitene kan man ignorere.

Noen vanlige presisjoner er:
Byte – 8 bit
Word – 16 bit
Dword – 32 bit
Qword – 64 bit

g) Fyll inn tallene som skal erstatte x i utrekningene under. Skriv det ned på papir.

0001 = $0 + 0 + 0 + 2^0$ = $0 + 0 + 0 + 1$ = **1**

0010 = $0 + 0 + 2^1 + 0$ = $0 + 0 + 2 + 0$ = **2**

0011 = $0 + 0 + 2^1 + 2^0$ = $0 + 0 + 2 + 1$ = **3**

0100 = $0 + 2^2 + 0 + 0$ = $0 + 4 + 0 + 0$ = **4**

0101 = $0 + 2^2 + 0 + 2^0$ = $0 + 4 + 0 + 1$ = **5**

0110 = $0 + 2^2 + 2^1 + 0$ = $0 + 4 + 2 + 0$ = **6**

0111 = $0 + 2^2 + 2^1 + 2^0$ = $0 + 4 + 2 + 1$ = **7**

1000 = $2^3 + 0 + 0 + 0$ = $8 + 0 + 0 + 0$ = **8**

1001 = $2^3 + 0 + 0 + 2^0$ = $8 + 0 + 0 + 1$ = **9**

1010 = $2^3 + 0 + 2^1 + 0$ = $8 + 0 + 2 + 0$ = **10**

h) Fyll inn tabellen under, skriv det ned på papir.

$4^0 = 1$
$4^1 = 4$
$4^2 = 16$
$4^3 = 64$
$4^4 = 256$
$4^5 = 1\,024$
$4^6 = 4\,096$
$4^7 = 16\,384$
$4^8 = 65\,536$
$4^9 = 262\,144$
$4^{10} = 1\,048\,576$

2) Konvertering

a) Konverter disse tallene fra binærtall (base 2) til desimaltall (base 10)

- 1111 1111 = **255**
- 0000 0000 = **0**
- 1001 1001 = **153**
- 1100 0011 = **195**
- 1010 1010 = **170**
- 0100 0101 = **69**
- 0010 1101 = **45**
- 1011 0010 = **178**

b) Konverter disse tallene fra desimaltall (base 10) til binærtall (base 2)

- 21 = **0001 0101**
- 9 = **0000 1001**
- 16 = **0001 0000**
- 196 = **1100 0100**
- 232 = **1110 1000**
- 72 = **0100 1000**
- 32 = **0010 0000**
- 256 = **1 0000 0000**

3) Addisjon binært

Adder disse tallene, bruk en byte (8 bit) presisjon. Bruk penn og papir.

a) 0000 0000 + 0010 0010 **= 0010 0010**

b) 1001 1001 + 0110 0110 **= 1111 1111**

c) 0001 1010 + 0000 0101 **= 0001 1111**

d) 1111 1111 + 0001 0000 **= 0000 1111**

e) 1111 1010 + 0101 0110 = **0101 0000**

f) 1010 1010 + 0001 0101 = **1011 1111**

g) 0110 0110 + 1100 1100 = **0011 0010**

h) 1011 1100 + 0001 0011 = **1100 1111**

i) 1100 1001 + 1111 1001 = **1100 0010**

j) 1010 1000 + 1110 1010 = **1001 0010**

Adder disse tallene. Oppgi svaret på en word (16 bit) presisjon. Bruk penn og papir.

a) 0000 0000 0000 0000 + 0101 1110 1100 1000 = **0101 1110 1100 1000**

b) 1001 1001 1001 1001 + 0110 0110 0110 0110 = **1111 1111 1111 1111**

c) 0001 0111 0110 1001 + 0010 1010 1001 = **0001 1010 0001 0010**

d) 1111 0101 0001 + 0100 0110 1101 0011 = **0101 0110 0010 0100**

e) 1010 0110 1101 1010 + 0101 1010 0110 1111 = **0000 0001 0100 1001**

4) Enerkomplement og toerkomplement

a) Hva er enerkomplement og hva er toerkomplement?

Enerkomplement er en måte å representere negative tall binært der man «flipper» alle bitene i tallet. Den negative representasjonen av et tall vil altså være den inverse versjonen av det «vanlige binærtallet».
Toerkomplement er en måte å blant annet representere negative tall binært. Den første biten i binærtallet på toerkomplement form er såkalt «signed» og angir om tallet er positivt(0) eller negativt(1). For å gjøre om et vanlig binærtall til toerkomplement gjør man det først om til enerkomplement, deretter legger man til 1.

```
Eksempel: Vi vil finne hvordan man representerer -7 binært
1. Finn det binære tallet til den positive versjonen av
   tallet, altså 7: 0110
2. Gjør det om til enerkomplement (flipper bitene): 1001
3. Legger til 1: 1001 + 0001 = 1011

Svar: -7 er 1011 på toerkomplement binærtall
```

b) Bruk enerkomplement på disse(5) tallene:

- 0000 = **1111**
- 0101 = **1010**
- 1100 1110 = **0011 0001**
- 0011 0001 1111 0000 = **1100 1110 0000 1111**
- 1010 0011 = **0101 1100**

c) Hvorfor trenger man presisjon i toerkomplement?

For at den signede biten skal bli riktig, altså det som angir om noe er et positivt eller negativt tall, må vi vite presisjonen. Om vi ikke hadde hatt presisjon og beholdt evt. overflow ville tall som egentlig er positive tall kunne blitt negative og vice versa.

d) I 8 bit presisjon, hva er det største og det minste tallet man kan representere binært uten toerkompliment?

Det største tallet man kan representere er 255 eller 1111 1111.
Det minste tallet man kan representere er 0 eller 0000 0000.

e) I 8 bit presisjon, hva er det største og det minste tallet man kan representere med toerkomplement?

Det største tallet man kan representere er 127 eller 0111 1111.
Det minste tallet man kan representere er -128 eller 1000 0000.

Vi kan ikke representere tallet 128 med 8 bits presisjon i toerkomplement fordi den første biten angir om tallet er positivt (0) eller negativt (1). For å representere tallet 128 trenger vi minst 9 bits presisjon (noe vi i dette tilfellet ikke har). Binærtallet 1111 1111 representerer i toerkomplement tallet -1.

f) Hvilken bit er det som gjør at et tall er negativt i toerkomplement?

Biten lengst til venstre, også kjent som den mest signifikante biten (MSB).

g) Sant / Usant: Man kan ikke se av seg selv om et binærtall er på toerkomplement form, det er noe man må få oppgitt.

Svar: Sant.

h) Hva er «the silly number»? Også kjent som «tulletallet».

Med 8 bits presisjon på toerkomplement er «the silly number» -128 da den positive versjonen av tallet ikke kan representeres med den presisjonen som er oppgitt. I et eksempel med 16 bits presisjon på toerkomplement vil -1024 være «the silly number».

i) Forklar og vis med eksempler hva overflow er, i 8 bits presisjon.

Si at vi skal regne `1111 0011 + 1111 1100 = ?` og oppgi svaret i 8-bit presisjon, i utregningen under ser vi at svaret kommer bli 9 bit men på grund av at svaret skal oppgis i 8-bit presisjon så blir den niende biten «overflow» og vil med det ikke være med i svaret.

Mente som blir «Overflow»

```
  1  111
    1111 0011
+   1111 1100
    1110 1111
```

5) Konvertering toerkomplement

Konverter binærtallene på toerkomplement med 8 bits presisjon under til desimaltall. Bruk penn og papir.

Når man skal konvertere et binærtall til desimal, og det er oppgitt at det er brukt toerkomplement på det, vil man først se til den mest signifikante biten, og avgjøre om den er positiv eller negativ. Om den er 0 er tallet positivt, og om den er 1 er tallet negativt. (Her må man også passe på at man vet presisjonen)

Eksempel:

Si at vi har tallet 1010 0101 med 8 bits presisjon på toerkomplement. Da må vi se på tallet 8 plasser fra høyre, og avgjøre om tallet er negativt eller positivt. Vi ser at 'Sign'-biten er et 1-tall, som vil si at tallet er negativt. Den første biten tilsvarer tallet –128.

Alle de andre bitene kan man anse som positive, og dermed kan vi addere dem med –128 slik:

`1010 0101 = (-128) + 32 + 4 + 1 = -91`

Om den mest signifikante biten hadde vært **0** ville vi fått et helt annet resultat:

`0010 0101 = (0) + 32 + 4 + 1 = 37`

Og om vi skulle kjørt regnestykket **uten** toerkompliment er resultatet helt annerledes:

`1010 0101 = (128) + 32 + 4 + 1 = 165`

a) 1111 1111 **= -1**

b) 0000 1000 = **8**

c) 1000 0000 = **-128**

d) 1110 0000 = **-32**

e) 0010 0000 = **32**

f) 0110 0010 = **98**

g) 1100 1000 = **-56**

h) 1111 0111 = **-9**

Konverter desimaltallene under til binærtall på toerkomplement med 8 bits presisjon. Bruk penn og papir.

Når man skal konvertere et desimaltall til binært, og det er oppgitt at det er brukt toerkomplement på det, vil man først se om det er et positivt eller negativt tall, det vil avgjøre om den mest signifikante biten er 0 eller 1. Om den er 0 er tallet positivt, og om den er 1 er tallet negativt. (Her må man også passe på at man vet presisjonen).

Om tallet er negativt vet vi altså att MSB er 1, det betyr at vi nå må finne ut av hvilke fler at bitene som skal være 1 før å frem rett svar, «Hva må vi plusse -128 med for å få rett svar?».

Eksempel:

Hva er -58 i binært med 8-bit presisjon og toerkomplement?

(-128) + 58 = 70

70 – 64 = 6

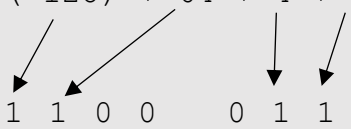
6 – 4 = 2

2 – 2 = 0

For å finne ut av hvilke tall som skal være 1 må vi følge det binære tallsystemet og se hvilket tall som er det nærmeste tallet under det opprinnelige tallet, eller samme tall som det opprinnelige tallet.

Nå har vi funnet ut av hvilke bit som skal være positive (1 og ikke 0), disse er 64, 4 og 2.

(-128) + 64 + 4 + 2 = -58



1 1 0 0 0 1 1 0

a) 10 = **0000 1010**

b) -8 = **1111 1000**

c) -77 = **1011 0011**

d) 69 = **0100 0101**

e) -42 = **1101 0110**

f) -153 = **Ingen løsning, tallet -153 er utenfor presisjonen vi har fått oppgitt.**

g) -12 = **1111 0100**

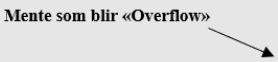
6) Subtraksjon og toerkomplement

Subtraher tallene under ved hjelp av toerkomplement med 8 bits presisjon. Oppgi svaret i binærtall (base 2) og vis utregning. Bruk penn og papir.

Eksempel:

Si at vi skal regne 23 – 4, det som er viktig å huske er at dette er de samme som 23 + (- 4).

1. Konverter til binært → 0001 0111 + (- 0000 0100) = ?
2. Bruk enerkomplement (flippe bitene) på det negative tallet.
3. (- 0000 0100) → 1111 1011
4. Bruk deretter toerkomplement (+ 1) på det flippede tallet.
5.
```
 1111 1011
+         1
 1111 1100
```
6. Plusse deretter det nye tallet med det opprinnelige positive tallet.



7.
```
    1  1111  1
     0001 0111
+    1111 1100
-------------
     0001 0011
```
8. Svar: 0001 0011
9. Siden vi får en overflow og MSB er 0 blir svaret positivt.

a) 16 – 8 = ?
0001 0000
+1111 1000
0000 1000

b) 24 – 9 = ?
0001 1000
+1111 0111
0000 1111

c) 127 – 128 = ?
0111 1111
+1000 0000
1111 1111

d) 98 – 100 = ?
0110 0010
+1001 1100
1111 1110

e) 87 – 19 = ?
0101 0111
+1110 1101
0100 0100

f) 78 – 12 = ?
0100 1110
+1111 0100
0100 0010

g) 82 – 69 = ?
0101 0010
+1011 1011
0000 1101

h) 42 – 42 = ?
0010 1010
+1101 0110
0000 0000

m) (10 – 12) – 2 = ?
0000 1010
+1111 0100
1111 1110

1111 1110
1111 1110
0000 0000

n) 127 – 119 – 7 – 1 = ?
0111 1111
+1000 1001
0000 1000

0000 1000
+1111 1001
0000 0001

0000 0001
+1111 1111
0000 0000

o) 50 – 10 = ?
0011 0010
+1111 0110
0010 1000

p) 77 – 99 = ?
0100 1101
+1001 1101
1110 1010

i) 12 – 20 = ?
0000 1100
+1110 1100
1111 1000

j) 15 – 24 = ?
0000 1111
+1110 1000
1111 0111

k) (10 – 5) – 9 = ?
0000 1010
+1111 1011
0000 0101

0000 0101
+1111 0111
1111 1100

l) (20 – 15) – 2 = ?
0001 0100
+1111 0001
0000 0101

0000 0101
+1111 1110
0000 0011

a) Hvorfor bruker vi heksadesimale tall?

   Med det binære tallsystemet blir det ofte mange bit å holde styr på, spesielt jo høyere tall man skal representere. Med det blir det også vanskeligere for oss mennesker å lese tallene. Vi bruker det heksadesimale tallsystemet fordi det kan forenkle 4 bit til kun ett siffer/karakter. Høye tall i det heksadesimale tallsystemet vil også være kortere enn i vårt «vanlige» desimale tallsystem.

b) Hva er prefix-en for heksadesimale tall?
   Svar: 0x

c) Hvor stor er en nibble?
   Svar: 4 bit

d) Skriv alle tall sifrene i det heksadesimale tallsystemet
   Svar: 0123456789ABCDEF

e) Hvor mange heksadesimale siffer trenger man for å representere 16 bit?

   To siffer som representerer 8 bit hver. Husk at 8 bit kan representere tall fra 0 – 15 som er tallene man kan representere på én plass i det heksadesimale tallsystemet.

8) Konvertering heksadesimal

Konverter tallene under fra det binære tallsystemet (base 2) til det heksadesimale tallsystemet (base 16).

a) 1000 = **0x8**

b) 0010 1101 = **0x2D**

c) 0101 1100 = **0x5C**

d) 1110 1110 = **0xEE**

Konverter tallene under fra det heksadesimale tallsystemet (base 16) til det binære tallsystemet (base 2).

a) 0x2 = **0010**

b) 0xFF = **1111 1111**

c) 0xD3 = **1101 0011**

d) 0x5F = **0101 1111**

Konverter tallene under fra det heksadesimale tallsystemet (base 16) til det desimale tallsystemet (base 10).

a) 0x3 = ***3***

b) 0xAB = ***171***

c) 0xFF = ***255***

d) 0x01 = **1**

Konverter tallene under fra det desimale tallsystemet (base 10) til det heksadesimale tallsystemet (base 16).

a) 7 = **0x7**

b) 130 = ***0x82***

c) 54 = ***0x36***

d) 23 457 = **0x5BA1**

9) Addisjon heksadesimal

Legg sammen tallene, og oppgi svaret på base 16 (heksadesimalt), bruk penn og papir. (Flere konverteringer kan være nødvendig underveis)

a) 0x00 + 0xFF = **0xFF**

b) 0x0F + 0xF0 = **0xFF**

c) 0xA5 + 0xC4 = **0x169**

d) 0x1D + 0xF4 = **0x111**

e) 0xA3 + 0x37 = **0xDA**

f) 0xDCBA + 0x1234 = **0xEEEE**

g) 0x2121 + 0x0F0F = **0x3030**

h) 0xAC5F + 0x0001 = **0xAC60**

i) 0xFFFF + 0x0 = **0xFFFF**

j) 0xBB8 + 0xB2D2 = **0xBE8A**

10) Subtraksjon heksadesimal

Subtraher tallene, og oppgi svaret på base 16 (heksadesimalt), bruk penn og papir. (Flere konverteringer kan være nødvendig underveis)

a) 0x14 – 0x03 **= 0x17**

b) 0xA6 – 0xA1 = **0x05**

c) 0xFF – 0xBC = **0x43**

d) 0xBB – 0xB5 = **0x06**

e) 0x7C – 0x33 = **0x49**

f) 0xF09D – 0xF00A = **0x0093**

g) 0xEDB – 0x8DA = **0x601**

h) 0xFFFF – 0xABCD = **0x5432**

i) 0x600F – 0x0A01 = **0x560E**

j) 0xFFFF – 0xFFFF = **0x0000**

11) ASCII

a) Bruk ASCII-tabellen og finn disse tegnene:
   - G – **0x47**
   - % **- 0x25**
   - = **- 0x3D**
   - } – **0x7D**
   - h – **0x68**
   - q – **0x71**

   Hva er heksadesimale tallet til hvert tegn?

b) Skriv ditt eget navn i heksadesimal fra ASCII tabellen på både store og små bokstaver.

   Eksempel:
   ole = 0x6F 0x6C 0x65
   OLE = 0x4F 0x4C 0x45

c) Hva er forskjellen på 0x41 og 0x61?

   0x41 = A og 0x61 = a. Dette er altså store og lille «A», hvis man ser på disse talene i binært ser vi at det er kun 1 bits forskjell mellom de.

   0x41 = 01**0**0 0001
   0x61 = 01**1**0 0001

d) Hva har binærtallet 0000 1000 som ASCII kode, binært?

   0000 1000 = 8, da må vi finne tallet «8» i ASCII tabellen hvilket er 0x38. 0x38 = 0011 1000, det betyr at svaret er **0011 1000**.