**TK1100**

**FORELESNING 0x01 BINHEX**

Noen kjappe tips til oppgavene:

* Skriv med dine egne ord, oppsøk gjerne andre kilder enn forelesningen som kan hjelpe forståelsen din.
* Hold det til maks. 5 – 6 setninger på spørsmålene der det står «Forklar kort».

Andre spørsmål kan det være fint å utdype grundigere.

* Enkelte av oppgavene kan gå litt utenom det som er presentert i forelesningene, bruk gjerne veilederene til å peke deg i riktig retning!

# Binær koding

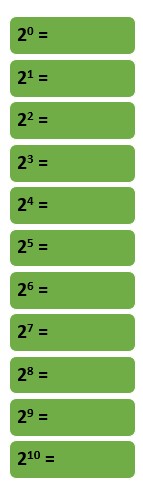
1. Hvilke ulike måter er det vist i forelesning at man kan representere karakterene i ASCII tabellen?

* Decimal, Hexadecimal, Octo, Character

1. Hvilke hovedtyper data, eller informasjon, behandler en datamaskin?

* Binære tall.

1. Fyll inn tabellen under.



* 1
* 2
* 4
* 8
* 16
* 32
* 64
* 128
* 256
* 512
* 1024
* 2048
* 4096

Oppgavene fortsetter på neste side.

1. 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

1. Hva har det binære tallsystemet som base?

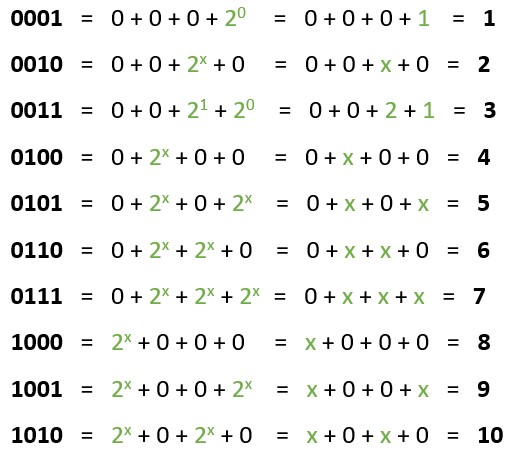
* Det binære tallsystemet har 2 som base.

1. Hvorfor bruker vi presisjon? Nevn fire vanlige presisjoner.

* Vi bruker presisjon for at verdien skal være lik når vi bruker ulike operasjoner (addisjon, subtraksjon, osv.).
* De vanlige presisjonene:
  + 8-bit
  + 16-bit
  + 32-bit
  + 64-bit

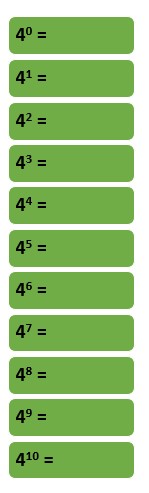
1. Fyll inn tallene som skal erstatte x i utrekningene under. Skriv det ned på papir.

* \_\_\_\_
* x = 1
* \_\_\_\_
* x = 2
* x1 = 2, x2 = 0
* x1 = 2, x2 = 1
* x1 = 2, x2 = 1, x3 = 0
* x = 3
* x1 = 3, x2 = 0
* x1 = 3, x2 = 1



Oppgavene fortsetter på neste side.

1. Fyll inn tabellen under, skriv det ned på papir.



* 1
* 4
* 16
* 64
* 256
* 1024
* 4096
* 16384
* 65536
* 262144
* 1048576
* 4194304
* 4096

# Konvertering

1. Konverter disse tallene fra binærtall (base 2) til desimaltall (base 10) • 1111 1111 **=**1\*27 + 1\*26 + 1\*25 + 1\*24 + 1\*23 + 1\*22 + 1\*21 + 1\*20 = 255
   * 0000 0000 **= 0**
   * 1001 1001 = 1\*27 + 0\*26 + 0\*25 + 1\*24 + 1\*23 + 0\*22 + 0\*21 + 1\*20 = 153
   * 1100 0011 **=** 195
   * 1010 1010 **=** 170
   * 0100 0101 **=** 69
   * 0010 1101 **=** 45
   * 1011 0010 **=** 178

Oppgavene fortsetter på neste side.

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

* + 21 **= 16 + 4 + 1 = 0001 0101**

* + 9 **= 8 + 1 = 0000 1001**

* + 16 **= 0001 0000**

* + 196 **= 128 + 64 + 4 = 1100 0100**

* + 232 **= 128 + 64 + 32 + 8 = 1110 1000**

* + 72 **= 64 + 8 = 0100 1000**

* + 32 **= 0010 0000**

* + 256 **= 0001 0000 0000**

# Addisjon binært

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

1. 0000 0000 +

0010 0010 **=**

0010 0010

1. 1001 1001 +

0110 0110 **=**

**1111 1111**

1. 0001 1010 +

0000 0101 **=**

**0001 1111**

1. 1111 1111 +

0001 0000 **=**

**0000 1111**

1. 1111 1010 +

0101 0110 =

**0101 0000**

1. 1010 1010 +

0001 0101 =

**1011 1111**

1. 0110 0110 +

1100 1100 =

**0011 0010**

Oppgavene fortsetter på neste side.

1. 1011 1100 +

0001 0011 =

**1100 1111**

1. 1100 1001 +

1111 1001 =

**1100 0010**

1. 1010 1000 +

1110 1010 =

**1001 0010**

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

1. 0000 0000 0000 0000 +

0101 1110 1100 1000 =

**0101 1110 1100 1000**

1. 1001 1001 1001 1001 +

0110 0110 0110 0110 =

**1111 1111 1111 1111**

1. 0001 0111 0110 1001 +

------ 0010 1010 1001 =

**0001 1000 0001 0010**

1. ------ 1111 0101 0001 +

0100 0110 1101 0011 =

**0101 0110 0010 0100**

1. 1010 0110 1101 1010 +

0101 1010 0110 1111 =

**0000 0001 0100 1001**

# Enerkomplement og toerkomplement

1. Hva er enerkomplement og hva er toerkomplement?

* Enerkomplement er når man flipper alle bittene. Eksempelvis var det her brukt når man først skulle representere negative tall binært, men problemet var at 0 var representert av både -0 og +0 som ble ganske tungt å regne ut.
* Toerkomplement var løsningen på dette. Denne metoden bærer med at man flipper alle bitene (fra 0 til 1 og fra 1 til 0) og plusser på 1.  
  Dette var en god løsning som fikset -0 og +0 problemet, og fungerer bra. Eneste denne metoden tilbrakte med seg var ‘the silly number’ som er det høyeste negative tallet, som ikke har en positiv motpart.

1. 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**

1. Hvorfor trenger man presisjon i toerkomplement?

* Man trenger en definert presisjon i toerkomplement slik at den ene biten man legger på kan overloade riktig og falle ut når man utfører metoden.

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

* 255, 256 med 0

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

* 127 og -128

1. Hvilken bit er det som gjør at et tall er negativt i toerkomplement?

* Første biten.

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

* Sant.

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

* Når man bruker toerkomplement blir det mulig å representere et «større» negativt tall enn det positive. 127 vs -128.

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

* 1110 1111

+ 0010 0000

= 0000 1111

# 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

Oppgavene fortsetter på neste side.

1. 1111 1111 **= 0000 0001**
2. 0000 1000 = **1111 1000**
3. 1000 0000 = **0011 1111**
4. 1110 0000 = 0010 0000
5. 0010 0000 = **1110 0000**
6. 0110 0010 = **1001 1110**
7. 1100 1000 = **0110 0010**
8. 1111 0111 = **0000 1001**

Oppgavene fortsetter på neste side.

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?     |  | | --- | | 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. |   (-128) + 58 = 70  70 – 64 = 6  6 – 4 = 2  2 – 2 = 0    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 |

1. 10 = **0000 1010**
2. -8 = **1111 1000**
3. -77 = 1011 0011

Oppgavene fortsetter på neste side.

1. 69 = 0100 0101
2. -42 = **1101 0110**
3. -153 = **Går ikke.**
4. -12 = **1111 0100**

# 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   1. Plusse deretter det nye tallet med det opprinnelige positive tallet.      1. 1 1111 1   0001 0111  + 1111 1100 0001 0011   1. Svar: 0001 0011 2. Siden vi får en overflow og MSB er 0 blir svaret positivt. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. 16 – 8 = ?      1. 24 – 9 = ?      1. 127 – 128 = ?      1. 98 – 100 = ?      1. 87 – 19 = ?      1. 78 – 12 = ? |  | 1. (10 – 12) – 2 = ?      1. 127 – 119 – 7 – 1 = ?      1. 50 – 10 = ?      1. 77 – 99 = ? |  | 1. 12 – 20 = ?      1. 15 – 24 = ?      1. (10 – 5) – 9 = ?      1. (20 – 15) – 2 = ? |

1. 82 – 69 = ?

1. 42 – 42 = ?

# Heksadesimal

1. Hvorfor bruker vi heksadesimale tall?

1. Hva er prefix-en for heksadesimale tall?

1. Hvor stor er en nibble?

1. Skriv alle tall sifrene i det heksadesimale tallsystemet

1. Hvor mange heksadesimale siffer trenger man for å representere 16 bit?

Oppgavene fortsetter på neste side.

# Konvertering heksadesimal

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

1. 1000 **= ?**
2. 0010 1101 **= ?**
3. 0101 1100 **= ?**
4. 1110 1110 **= ?**

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

1. 0x2 **= ?**
2. 0xFF **= ?**
3. 0xD3 **= ?**
4. 0x5F **= ?**

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

1. 0x3 **= *?***
2. 0xAB **= *?***
3. 0xFF **= *?***
4. 0x01 **= ?**

Oppgavene fortsetter på neste side.

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

1. 7 **= ?**
2. 130 **= *?***
3. 54 **= *?***
4. 23 457 **= ?**

# 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 = ?

1. 0x0F + 0xF0 = **?**
2. 0xA5 + 0xC4 = **?**
3. 0x1D + 0xF4 = **?**
4. 0xA3 + 0x37 = ?
5. 0xDCBA + 0x1234 = ?
6. 0x2121 + 0x0F0F = ?
7. 0xAC5F + 0x0001 = ?
8. 0xFFFF + 0x0 = ?
9. 0xBB8 + 0xB2D2 = ?

Oppgavene fortsetter på neste side.

# Subtraksjon heksadesimal

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

1. 0x14 – 0x03 **= ?**
2. 0xA6 – 0xA1 = ?
3. 0xFF – 0xBC = ?
4. 0xBB – 0xB5 = ?
5. 0x7C – 0x33 = ?
6. 0xF09D – 0xF00A = ?
7. 0xEDB – 0x8DA = ?
8. 0xFFFF – 0xABCD = ?
9. 0x600F – 0x0A01 = ?
10. 0xFFFF – 0xFFFF = ?

# ASCII

1. Bruk ascii-tabellen og finn disse tegnene:
   * G
   * %
   * =
   * }
   * h
   * q

Hva er heksadesimale tallet til hvert tegn?

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

1. Hva er forskjellen på 0x41 og 0x61?

1. Hva har binærtallet 0000 1000 som ASCII kode, binært?