Teori og oppgaver om 2-komplement

1) Binær addisjon

Vi legger sammen binære tall på en tilsvarende måte som desimale tall (dvs. tall i 10-talssystemet). Vi må imidlertid huske at 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 og 1 + 1 = 10, dvs. 0 og 1 i mente. Videre er 1 + 1 + 1 = 11 og det gir 1 og 1 i mente.

Eksempel 1:

```
  10101110111
+     1001101
-------------
```

Vi starter bakerst. Vi får 1 + 1 = 10. Det gir 0 under streken og 1 i mente. Deretter får vi 1 + 1 + 0 = 10. Det gir igjen 0 under streken og 1 i mente. Deretter blir det 1 + 1 + 1 = 11. Det gir 1 under streken og 1 i mente. Osv. Svaret blir slik:

```
  10101110111
+     1001101
-------------
= 10111000100
-------------
```

Eksempel 2:

```
   11001100110
+  10100110101
--------------
= 101110011011
--------------
```

2) Binær subtraksjon

Her kan vi bruke en tilsvarende teknikk som for desimale tall, dvs. vi "låner" verdier.

Eksempel 1:

```
  10101110111
-     1001101
-------------
```

Vi starter bakerst. Der får vi 0 under streken siden 1 – 1 = 0. Deretter 1 under streken siden 1 – 0 = 1, så 0 under streken siden 1 – 1 = 0. Men så får vi 0 – 1. Dermed må vi "låne" 1-eren ved siden av. Men når den flyttes en posisjon mot høyre blir den til en 2-er. Det gir 2 – 1 = 1 under streken. Osv. Resultatet blir:

```
  10101110111
-     1001101
-------------
= 10100101010
-------------
```

Eksempel 2:

```
   11001100000
-  10100111111
-------------
=    100100001
-------------
```

3) Fast bitformat – fortegnsbit og 2-komplement

Vi kan på samme måte som for desimale tall, bruke et fortegn for å markere at et tall er negativt. Det desimale tallet 21 skrives som 10101 på binærform og dermed kunne –21 skrives som –10101. Men dette er ikke slik det normalt gjøres på en datamaskin. Der brukes det som kalles 2-komplement med fortegnsbit.

2-komplement kan kun brukes når heltall skrives og lagres med et fast antall binære siffer. Hvert siffer er 0 eller 1 og dette kalles biter. Ikke-negative binære tall skrives vanligvis uten ledende 0-er. For eksempel skrives tallet 21 normalt som 10101 på binærform. Men med et fast antall biter, for eksempel 8 biter, blir tallet normalt oppgitt med ledende 0-er. For 21 blir det 00010101.

Når vi bruker et fast antall biter kalles den første biten en fortegnsbit. Hvis fortegnsbiten er 1 er tallet negativt og hvis den er 0 er tallet ikke-negativt. Det fører til at det største mulige positive heltallet i et fast 8 biters format er det binære tallet 01111111 og det er tallet 127. Det binære tallet 00000000 representerer tallet 0 og det er verken positivt eller negativt. Tallet 1 skrives som 00000001, osv. Spesielt betyr dette at alle de positive tallene fra og med 1 til og med 127 vil kunne skrives i et fast 8 biters format.

Hvis fortegnsbiten er 1 så er tallet negativt. La oss som eksempel bruke 8 biter som det faste antallet biter. Da vil 10110100 være et negativt tall. Men hvilket tall? Flg. regel viser hvordan vi kan finne ut det:

1. Gitt et negativt binært tall n med 8 binære siffer. Dermed er første bit lik 1.
2. Finn komplementet k til n. (Det betyr å la alle 1-ere bli 0-er og alle 0-er bli 1-ere)
3. Finn $m = k + 1$. Bruk vanlig binær addisjon.
4. Tallet n er det samme som $-m$.

Eksempel 1:

1. Gitt $n = 10110100_2$
2. Komplementet til n er $k = 01001011_2$
3. $m = k + 1 = 01001011_2 + 1 = 01001100_2 = 76$
4. $n = 10110100_2$ er det samme som -76

Eksempel 2:

1. Gitt $n = 11111111_2$
2. Komplementet til n er $k = 00000000_2$
3. $m = k + 1 = 00000000_2 + 1 = 00000001_2 = 1$
4. $n = 11111111_2$ er det samme som –1

Eksempel 3:

1. Gitt $n = 10000000_2$
2. Komplementet til n er $k = 01111111_2$
3. $m = k + 1 = 01111111_2 + 1 = 10000000_2 = 128$
4. $n = 10000000_2$ er det samme som –128

Vi kan også gå den omvendte veien. Dvs. vi starter med et ikke-negativt tall og ønsker å finne de binære sifrene til det samme tallet, men med motsatt fortegn. Da er reglen slik:

1. Gitt et ikke-negativt binært tall n med 8 siffer, dvs. første bit er 0.
2. Finn komplementet k til n.
3. Finn $m = k + 1$. Bruk vanlig binær addisjon.
4. Tallet m er det samme som $-n$.

Eksempel 4:

1. Gitt $n = 108_{10} = 01101100_2$
2. Komplementet til n er $k = 10010011_2$
3. $m = k + 1 = 10010011_2 + 1 = 10010100_2$
4. 10010100_2 er det samme som –108

Oppgaver

Oppgave 1.

I denne oppgaven ser vi bare på positive binær tall. Da brukes ikke fortegnsbit og ingen tall har ledende 0-er. Gitt de binære tallene a = 1011 1000, b = 110 0111 og c = 1000 0000.

a) Bruk binær addisjon slik som beskrevet over til å finne $a + b$, $a + c$ og $b + c$

b) Bruk binær subtraksjon slik som beskrevet over til å finne $a - b$, $a - c$, $a - 1$, $b - 1$ *og* $c - 1$

c) Windows 7 inneholder en kalkulator (Alle programmer | Tilbehør). Sett den til å være en programmeringskalkulator (gå inn i menyen Vis/View). Der kan du regne med binær siffer. Løs også oppgavene i a) og b) ved hjelp av kalkulatoren. Da bør du helst få samme svar.

Oppgave 2.

I denne oppgaven skal vi bruke et fast antall biter på 8, la første bit være fortegnsbit og bruke 2-komplement til negative tall.

a) Hvordan skrives flg. heltall (i 10-tallssystemet) på binærform med hensyn på dette bitformatet? i) 0, ii) 10, iii) 100, iv) –1, v) –10, vi) –100 ?

b) Hvilke heltall (i 10-tallssystemet) representerer flg. bitsekvenser? i) 01010101 ii) 10101010 iii) 11111111 iv) 10000000 v) 01111111

c) Legg sammen (binæraddisjon) flg. par av bitsekvenser: i) 01001101 og 00110011 ii) 10101010 og 01010101.

Oppgave 3.

I denne oppgaven skal vi bruke et fast antall biter på 32, la første bit være fortegnsbit og bruke 2-komplement til negative tall. Dette svarer til svarer til datatypen *int* i Java.

a) Hvordan skrives flg. heltall (i 10-tallssystemet) på binærform med hensyn på dette bitformatet? i) –1, ii) –2, iii) –3

b) Hvilke heltall (i 10-tallssystemet) representerer flg. bitsekvenser?

i) 10000000000000000000000000000001 ii) 11111111111111111111111111111000

Oppgave 4.

Klassen *Integer* i Java inneholder metoden *toBinaryString*. Den konverterer heltall til en bitsekvens i form av en *String*. Den kan for eksempel brukes slik:

```
String s = Integer.toBinaryString(1234);
String t = Integer.toBinaryString(-5678);

System.out.println(s);
System.out.println(t);

Utskrift:

10011010010
11111111111111111110100111010010
```

Metoden fjerner ledende 0-er hvis heltallet er positivt.

Bruk teknikken over til å løse Oppgave 3a).

Oppgave 5.

Flg. programkode gjør slik som i regnstykket i Eksempel 4 i Avsnitt 3:

```
int n = 108;
int k = ~n;   // komplementet til n
int m = k + 1;
System.out.println(m);
```

a) Lag et Java-program der dette inngår. Hva bli utskriften?

b) La $n = -108$. Hva blir utskriften nå?

c) La $n = -2147483648$. Hva blir utskriften nå? Hva er det som skjer?

Løsningsforslag til oppgavene

Oppgave 1.

a) 1 0001 1111, 1 0011 1000, 1110 0111
b) 101 0001, 11 1000, 1011 0111, 110 0110, 111 1111

Oppgave 2.

a) i) 0 = 00000000 ii) 10 = 00001010 iii) 100 = 01100100
iv) –1 = 11111111 v) –10 = 11110110 vi) –100 = 10011100

b) i) 85 ii) –86 iii) –1 iv) –128 v) 127

c) i) –128 ii) –1

Oppgave 3.

a) i) 11111111111111111111111111111111 ii) 11111111111111111111111111111110
iii) 11111111111111111111111111111101

b) i) –2147483647 ii) –8