Toerskomplement

Skrevet av:
Are Willumsen

Innledning

I forbindelse med lab 4 foregår det et toerskomplent-operasjon ved det mest signifikante bitet (MSB). Det er ikke umiddelbart intuitivt hvorfor dette gjøres, og dette notat skal prøve å gi en liten forklaring på hvorfor dette gjøres. Notatet vil samtidig vise en ny måte å se på tall på toerskomplement form.

Negative tall

Først en liten oppfrisking i negative tall. Når de er på binær form beskriver det fremste bit'et fortegnet. Det vil si hvis det fremste bit'et er 1 så er tallet negativt og hvis det fremste bit'et er 0 så er tallet positivt ($0 = +, 1 = -$). På labben har vi for det meste 16 bits busser. Tall som går på disse bussene er derfor negativt om det 16. bitet er 1. Vi deler vanligvis opp bit'ene i fire og fire. Der hver "4-bit" kan representeres med et hexadesimalt siffer (0-F). Slik at ved 16 bit så har vi 4 slike "4-bit", og på labben representerer vi disse med 4 hexadesimale siffer. F.eks. så skrive 0011 1010 0000 1111 på hexadesmal form som:

$$\begin{array}{cccccc} & 0011 & 1010 & 0000 & 1111 \\ = & 3 & A & 0 & F \end{array}$$

Når er da et 16 bits tall representert med 4 hexadesimale siffer negativt?. Hvis tallet er negativt så er som sagt det fremste bitet 1. Dvs det fremste bitet i det fremste hexadesimale sifferet er 1. Dette sier oss igjen at den minste verdien som det fremste hexadesimale sifferet kan ha er 1000, og dette er:

$$\begin{array}{cccccc} 1 & 0 & 0 & 0 \\ 1 \bullet 2^3 & +0 \bullet 2^2 & +0 \bullet 2^1 & +0 \bullet 2^0 & = 8 \end{array}$$

Dermed kan vi slå fast at dersom det fremste sifferet er 8 eller større så er tallet negativt.

Det er to måter å representere negative tall på disse kalles på engelsk "sign-magnitude" og "two's complement". I den følgende avsnittet vil de disse to bli kjapt gjennomgått.

Sign-magnitude

Sign-magnitude er den mest intuitive, der det fremste bit'et forteller fortegnet mens de andre bit'ene forteller verdien. Slik at for et 4-bits tall så har vi:

$$\begin{array}{cccccc} 1 & 0 & 1 & 0 \\ - & (+0 \bullet 2^2 & +1 \bullet 2^1 & 0 \bullet 2^0) & = -2 \end{array}$$

Mens det for åtte bit vil være

$$\begin{array}{cccccccccc} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ - & (+0 \bullet 2^6 & +1 \bullet 2^5 & +0 \bullet 2^4 & +1 \bullet 2^3 & +0 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0) & = -58 \end{array}$$

Vi legger her merke til at de tallene med samme tallverdi og motsatt fortegn har samtlige bit like bortsett fra det fremste:

$$\begin{array}{cccccc} 0 & 0 & 1 & 0 \\ + & (+0 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0) & = 2 \end{array}$$

$$\begin{array}{cccccccc} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ + & (+0 \bullet 2^6 & +1 \bullet 2^5 & 0 \bullet 2^4 & +1 \bullet 2^3 & +0 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0) \end{array} = 58$$

Man kan også legge merke til at de forskjellige bit'ene har ulik verdi avhengig av om det fremste bitet er 1 eller 0. Dette vises tydelig om man regner ut parantesen:

$$\begin{array}{cccc} 1 & 0 & 1 & 0 \\ & -0 \bullet 2^2 & -1 \bullet 2^1 & -0 \bullet 2^0 \end{array} = -2$$

$$\begin{array}{cccc} 0 & 0 & 1 & 0 \\ & +0 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0 \end{array} = 2$$

Her ser vi at fortegnsbitet forteller fortegnet på samtlige bit. Dvs dersom MSB er 1 så vil samtlige bit ha negativ verdi.

Two's complement

Toers komplements form av tall er den måten negative tall vanligvis representeres i digitale systemer. Det er det flere grunner til, og notatet vil ikke gå inn på dette. For tall på toerskomplement form er regelen noe annerledes. Det som har blitt forelest og det som står i boka er at man finner det negative tallet ved å ta toers-komplementet til det tilsvarende postivie tallet. Toers komplement av et tall er å invertere samtlige bit for deretter å legge til en (se også lab 2). Det vil si dersom vi har tallet 2 og skal finne ut hva -2 er for noe så gjør vi følgende

Vi finner ut hvordan 2 representeres på binær form:

$$\begin{array}{cccc} 0 & 0 & 1 & 0 \\ & +0 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0 \end{array} = 2$$

også inverterer vi samtlige bit (også det fremste bit'et (MSB), dvs fortegnsbitet)

$$1 \quad 1 \quad 0 \quad 1$$

også legger vi til 1

$$\begin{array}{cccc} 1 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \end{array}$$

Dermed kan vi fastslå at -2 er 1110 når vi representerer negative tall på toerskomplement form. For å finne ut hva et toers komplement negativt tall er, tas bare toers komplementet av det negative tallet

$$\begin{array}{rcccc} \text{vi tar det opprinnelig tallet} & 1 & 1 & 1 & 0 \\ \text{inverterer og får:} & 0 & 0 & 0 & 1 \\ \text{legger til 1 og får:} & 0 & 0 & 1 & 0 \end{array}$$

Som vi ser gir dette 0010 = 2 som vi hadde forventet å få. Vi tar et eksempel til, denne gangen med 8 bit:

$$\begin{array}{rcccccccc} \text{vi tar det opprinnelig tallet: } 58 = & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ \text{inverterer og får:} & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \text{legger til 1 og får: } -58 = & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}$$

Ny måte å se på toerskomplement-tall

Siden tallene i labben er representert på toerskomplement form er det denne som er mest

nyttig å se mer på. I dette avsnittet vil det bli presentert en ny måte å se tall i toerskomplement form på. I toers komplement kan man tolke det første bitet som et negativt tall, og resten som positive tall. Hææ, sier du kanskje nå, men jeg skal isteden prøve å illustrere dette med et eksempel. Gitt tallet 2, som vi vet skrives:

$$0 \quad 0 \quad 1 \quad 0$$

For å regne om dette til desimaltall bruker vi vanligvis følgende algoritme:

$$\begin{matrix} 0 & 0 & 1 & 0 \\ 0 \bullet 2^3 & +0 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0 \end{matrix} = 2$$

Det nye er fra nå av at vi skal tolke det fremste bit'et som negativt, slik at omregning isteden blir

$$\begin{matrix} 0 & 0 & 1 & 0 \\ 0 \bullet (-2^3) & +0 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0 \end{matrix} = 2$$

Vi tar en titt på tallet -2 to for å kontrollere at dette faktisk stemmer

$$\begin{matrix} 1 & 1 & 1 & 0 \\ 1 \bullet (-2^3) & +1 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0 \end{matrix} = -8 + 4 + 2 = -2$$

Hvis du kontrollerer andre tall så vil du se at det også stemmer for disse. Legg samtidig merke til at bit'ene har samme fortegn og verdi uavhengig om MSB er 0 eller 1 i motsetning til ved sign-magnitude. Der skiftet alle bit'ene fortegn (og dermed verdi) dersom MSB skiftet verdi.

Sign extension (fortegnsutvidelse)

Det er forelest om sign extension av toers komplement tall. Dette går i korthet på at man kan utvide ethvert binært tall med et siffer front som er lik det sifferet som er MSB fra før. Dette er vanskelig å forklare med ord så vi tar et eksempel isteden:

1010	1 er MSB og vi utvider derfor med en 1 som ny MSB
= 11010	1 er MSB og vi utvider derfor med en 1 som ny MSB
= 111010	1 er MSB

Dette er forsåvidt ikke så uvant for oss. Det som er litt uvant er at dette også gjelder for enere. For vi er fra før vant til at (dette gjelder også for desimaltall)

$$\begin{matrix} & 0101 \\ = & 00101 \\ = & 000101 \end{matrix}$$

Som jo også egentlig bare er sign extension. Vi tar en ny titt på tallet -2 for å verifisere at dette faktisk stemmer, og vi bruker den nye måten å tolke toers-komplement tall:

$$\begin{matrix} & & 1 & 1 & 1 & 0 & & = -2 \\ \text{kan utvides til:} & 1 & 1 & 1 & 1 & 0 & & \\ = & 1 \bullet (-2^4) & +1 \bullet 2^3 & +1 \bullet 2^2 & +1 \bullet 2^1 & +0 \bullet 2^0 & = -16 + 8 + 4 + 2 = -2 & \end{matrix}$$

Hvis vi titter litt nøyere på det som står over ser vi at, da vi gjorde en sign-extension så la vi til $1 \bullet (-2^4)$ på grunn av den eneren vi la til i front. Men i og med at det kom et bit foran bit 3 så skiftet bit 3 fortegn i og med at dette bittet ikke lenger var MSB. Det vil si at bit 3 gikk fra å ha verdien $1 \bullet (-2^3)$ til det å ha verdien $1 \bullet 2^3$. Dermed har verdien av bit 3 blitt endret med

$$1 \bullet 2^3 - 1 \bullet (-2^3) = 2 \bullet 2^3 = 2^4$$

Vi kan dermed si at vi har lagt til 2^4 til det opprinnelige tallet. Samtidig så har vi trukket fra -2^4 ved å sette inn en 1 foran som ny MSB. Siden vi har trukket fra og lagt til den samme

verdien så vil det nye tallet ha samme verdi som det gamle.

Mer generelt kan vi si at dersom vi har et tall på n+1 bit:

$$x_n x_{n-1} \ldots . x_1 x_0$$

der hver x_i representerer et bit (0 eller 1). Verdien av dette binære tallet er da:

$$x_n \bullet (-2^n) + x_{n-1} \bullet 2^{n-1} + \ldots + x_1 \bullet 2^1 + x_0 \bullet 2^0$$

Hvis vi fortegnsutvider med et bit $x_{n+1} = x_n$ så vil vi få tallet og verdien:

$$x_{n+1} x_n x_{n-1} \ldots . x_1 x_0$$

$$x_{n+1} \bullet (-2^{n+1}) + x_n \bullet 2^n + x_{n-1} \bullet 2^{n-1} + \ldots + x_1 \bullet 2^1 + x_0 \bullet 2^0$$

Dersom vi trekker dette tallet fra det opprinnelige ser vi at:

$$\begin{array}{rccccccc} & x_{n+1} \bullet (-2^{n+1}) & x_n \bullet 2^n & +x_{n-1} \bullet 2^{n-1} & +\ldots+ & +x_1 \bullet 2^1 & +x_0 \bullet 2^0 \\ - & [& +x_n \bullet (-2^n) & +x_{n-1} \bullet 2^{n-1} & +\ldots+ & +x_1 \bullet 2^1 & +x_0 \bullet 2^0] \\ \hline = & x_{n+1} \bullet (-2^{n+1}) & +2 \bullet x_n \bullet 2^n & +0 \bullet 2^{n-1} & +\ldots+ & 0 \bullet 2^1 & 0 \bullet 2^0 \end{array}$$

Vi fjerner 0'ene og får

$$\begin{aligned} &= x_{n+1} \bullet (-2^{n+1}) + 2 \bullet x_n \bullet 2^n \\ &= x_{n+1} \bullet (-2^{n+1}) + x_n \bullet 2^{n+1} \\ &= (x_n - x_{n+1}) \bullet 2^{n+1} \\ &= 0 \end{aligned}$$

siden $x_{n+1} = x_n$. Dermed er det bevist at en fortegnsutvidelse ikke endrer verdien på tallet.

Det samme gjelder selvfølgelig når vi fjerner de fremste bit'ene, i og med at dette bare er det motsatte av det å legge til et bit foran. Kravet er selvfølgelig at vi bare kan fjerne det fremste bitet dersom bitet bak (det nye MSB) er lik det bit'et vi fjerner. Hvis vi gjentar denne prosessen kan vi legge til/fjerne flere i front på tallet. Det eneste kravet er at disse bit'ene er like. Dvs.:

$$\begin{aligned} &000101 \\ = &0101 \end{aligned}$$

og for tallet -2 har vi da.

$$\begin{aligned} &1110 \\ = &10 \\ = &1111111111111110 \end{aligned}$$

Oppsummering av negative tall

Etter å ha lest delkapitelet over så har vi kommet fram til og vist at:

- Det er to måter å representer negative binære tall. Digitale systemer bruker i hovedsak toers komplement
- Å skifte fortegn på et (toers komplement) tall er det samme som å ta toers komplementet av dette tallet.
- For toers komplement så er det fremste bit'et negativt med en tallverdi tilsvarende posisjonen. Dvs står det på plass 10 så har det verdien -2^{10} Alle andre bit er negative.
- For toers komplement så har bit'ene samme verdi uavhengig av verdien på MSB. I sign-magnitude skifter disse verdi (fortegn)
- Ved toers komplement så kan vi fortegns utvide/forminske ethvert tall ved å legge til/trekke et bit i front slik at det nye MSB er lik det gamle.

Multiplikasjon

Når man multipliserer to tall så tar man som regel hvert siffer hvor seg av multiplikatoren og

ganger sammen med multiplikanden. Skal her vise hvordan dette gjøres. For å legge det litt mer opp mot slik det gjøres på labben, så velger jeg her å summere etter hvert ledd. Vi multipliserer de to tallene 102 og 15 slik:

$$
\begin{array}{rrrrrrrrrr}
1 & 0 & 2 & & & \bullet & 1 & 5 & = & 1530 \\
\hline
 & & & & & & 0 & 0 & & \\
 & & & & + & & 3 & 0 & & \\
\hline
 & & & & = & & 3 & 0 & & \\
 & & & & + & 0 & 0 & 0 & & \\
\hline
 & & & & = & & 3 & 0 & & \\
 & & & + & 1 & 5 & 0 & 0 & & \\
\hline
 & & & = & 1 & 5 & 3 & 0 & &
\end{array}
$$

Kort sagt kan vi si at vi regner ut først 2 • 15 også legger vi til 10 • 0 • 15 og til slutt så legger vi til 100 • 1 • 15. Skriver vi dette litt mer kompakt og matematisk får vi:

$$102 \bullet 15 = (((2 \bullet 15) + 10 \bullet 0 \bullet 15) + 100 \bullet 1 \bullet 15)$$

Dette er samme måten som vi regner ut svaret i lab 4, bortsett fra at her er tallene binære. Slik at f.eks. 5•1 (0101•0011) regnes ut slik

$$
\begin{array}{rrrrrrrrrrr}
0 & 1 & 0 & 1 & & \bullet & 0 & 0 & 0 & 1 & = 0101 = 5 \\
\hline
 & & & & & & & 0 & 0 & & \\
 & & & & & + & & 0 & 1 & & \\
\hline
 & & & & & = & 0 & 0 & 1 & & \\
 & & & & & + & 0 & 0 & 0 & & \\
\hline
 & & & & & = & 0 & 0 & 1 & & \\
 & & & & + & 0 & 1 & 0 & 0 & & \\
\hline
 & & & & = & 0 & 1 & 0 & 1 & & \\
 & & & & + & 0 & 0 & 0 & 0 & & \\
\hline
 & & & & = & 0 & 1 & 0 & 1 & &
\end{array}
$$

I dette tilfellet ser vi at det endelige svaret er regnet ut som

$$0101 \bullet 0001 = (0 \bullet 1000 \bullet 0001 + (1 \bullet 100 \bullet 0001 + (0 \bullet 10 \bullet 0001 + (1 \bullet 0001 + 0))))$$

eller om vi skriver det hele med desimaltall så får vi

$$5 \bullet 1 = (0 \bullet -2^3 \bullet 0001 + (1 \bullet 2^2 \bullet 0001 + (0 \bullet 2^1 \bullet 0001 + (1 \bullet 2^0 \bullet 0001 + 0))))$$

Hvis vi skriver om mulitplikatoren (X)(det vil si det første tallet: 5), et binært tall bestående av n+1 siffer som

$$X = x_n x_{n-1} \ldots . x_1 x_0$$

der x'ene er hvert siffer (bit) og har verdien 0 eller 1. Da er verdien av X gitt som

$$X = x_n \bullet (-2^n) + x_{n-1} \bullet 2^{n-1} + \ldots + x_1 \bullet 2^1 + x_0 \bullet 2^0$$

Ja som vi husker er det fremste bitet x_n er negativt. I tillegg kaller vi multiplikanden for Y (det vil si det høyre tallet i multiplikasjonsstykket: 1). Så regner vi ut verdien av multiplikasjonen ved å regne ut:

$$X \bullet Y = (x_n \bullet (-2^n) \bullet Y + (x_{n-1} \bullet 2^{n-1} \bullet Y + ... + (x_1 \bullet 2^1 \bullet Y + (x_0 \bullet 2^0 \bullet Y))...))$$

Når vi i multiplikatoren så regner ut alle disse svarene oppnår vi toerpotensene ved å skifte til venstre. Fordi: Å skifte til venstre k ganger er det samme som å mulitiplisere tallet med 2^k. På lik linje med at at skift til k hakk til venstre i desimaltall er å multiplisere med 10^k. ($150000 = 15 \bullet 10^4$) I multiplikatoren realiserer vi dette ved å skifte multiplikanden (Y) til venstre en gang for hvert bit i multiplikatoren. Vi definere en hjelpevariabel S_{15} definert som:

$$S_{14} = (x_{14} \bullet 2^{14} \bullet Y + ... + (x_1 \bullet 2^1 \bullet Y + (x_0 \bullet 2^0 \bullet Y))...)$$

Vi legger her merke til at dette er svaret vi vil ha når vi har regnet ut de bakerste 15 bit'ene. Det vil si det siste mellomsvaret. Det eneste som står igjen er å regne ut fremste bitet multiplisert med Y og til dette.:

$$\begin{aligned} X \bullet Y &= (x_{15} \bullet (-2^{15}) \bullet Y + (x_{n-1} \bullet 2^{n-1} \bullet Y + ... + (x_1 \bullet 2^1 \bullet Y + (x_0 \bullet 2^0 \bullet Y))...)) \\ &= x_{15} \bullet (-2^{15}) \bullet Y + S_{14} \\ &= -1 \bullet 2^{15} \bullet x_{15} \bullet Y + S_{14} \end{aligned}$$

Hvordan regner vi så ut $-1 \bullet 2^{15} \bullet x_{15} \bullet Y$?

Vell 2^{15} har vi allerede oppnådd ved å skifte Y til venstre 15 ganger (en gang for hvert bit vi allerede har tatt med i multiplikasjonen). x_{15} er 1 eller 0. Dvs som før hvis den er 0 så skal vi ikke legge til noe og hvis det er 1 så skal vi legge til $-1 \bullet (2^{15} \bullet Y)$ til S_{14}. Vi har i forrige avsnitt vist at toer koplementet av et tall er det samme som å skifte fortegn. Og det å skifte fortegn er det samme som å multiplisere med -1. Dermed oppnår vi $-1 \bullet (2^{15} \bullet Y)$ ved å ta toers komplement av det som ligger i multiplikand blokken (nemlig $2^{15} \bullet Y$, den opprinnelige multiplikanden, Y, venstreskiftet 15 ganger)

Konklusjoner

- Vi tar toerskomplement på MSB fordi dette bit'et er negativt.
- Svaret blir riktig uansett fortegn på mulitplikanden siden toerskomplementet uansett bare skifter fortegn.
- Man er nødt til å representere tallene på toerskomplement form for at dette skal kunne gjøres på denne måten.
- Siden bit'ene i toerskomplement form har lik verdi uansett fortegn, så trenger vi ikke ta hensyn til fortegnet når vi tar for oss de 15 LSB i multiplikatoren. Dvs hvert steg i multiplikatoren blir likt, uavhengig av fortegnet og vi trenger ikke å ta hensyn til dette underveis i regningen.