DET BINÆRE TALSYSTEM

I vort sædvanlige, decimale talsystem er grundtallet 10, hvilket betyder, at f.eks. heltallet 3567 skal forstås som

$$3 \cdot 10^3 + 5 \cdot 10^2 + 6 \cdot 10^1 + 7 \cdot 10^0 ,$$

og brøken 0.3975 skal forstås som

$$0 \cdot 10^0 + 3 \cdot 10^{-1} + 9 \cdot 10^{-2} + 7 \cdot 10^{-3} + 5 \cdot 10^{-4} ,$$

således at et tal (heltal, brøk eller blandet) i virkeligheden skal læses som en produktsum af de angivne cifre og passende potenser af 10, hvis eksponenters størrelse afhænger af det tilsvarende ciffers placering i afstand fra kommaet.

En talværdi kan naturligvis udtrykkes i et talsystem med vilkårligt grundtal - speciel 2, d.v.s. i det binære talsystem, hvor de enkelte cifre kaldes binære cifre og cifrene efter kommaet specielt binaler.

Det decimale tal 47.625 skrives således i binær form

$$101111.101$$

som skal forstås

$$1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

hvilket netop er lig

$$32 + 0 + 8 + 4 + 2 + 1 + 0.5 + 0 + 0.125 = 47.625$$

Transformation af decimalbrøk til binærbrøk - og omvendt

Af speciel interesse er binærbrøken, idet cifrene i et ord i lageret eller et register normalt tillægges en talværdi, der numerisk er $\leq$ 1.

Omformningen fra decimalbrøk til binærbrøk sker ved successiv multiplikation med 2:

$$a = b_1 \cdot 2^{-1} + b_2 \cdot 2^{-2} + \ldots\ldots = 0.b_1 b_2 b_3 \ldots . b_v \leq 1.$$

Ved multiplikation af denne ligning med 2 fås

$$2a = b_1 + b_2 \cdot 2^{-1} + \ldots . = b_1 . b_2 b_3 \ldots . b_v$$

Heraf aflæses, at b_1 = heldelen af 2a, som er enten 0 eller 1.
Sættes $2a - b_1 = a_1 = 0.b_2b_3 \ldots\ldots b_v$
og gentages processen, findes b_2 o.s.fr.

0 . 5625 . 2
1 . 1250 . 2
0 . 2500 . 2
0 . 5000 . 2
1 . 0000 . 2
0 . 0000 d.v.s. 0.5625 = 0.1001

Omdannelse fra binærbrøk til decimalbrøk kan ske på den tilsvarende måde, nemlig ved at multiplicere med basis i 10-talsystemet blot noteret i 2-talsystemet, altså 1010. Måske foregår omformninger denne vej dog lettere ved en anden metode, som vi derfor vil vise ved et eksempel; til gængæld er den ikke særlig egnet ved overgangen fra decimal- til binærbrøk.

$0.1001 = (((1 \cdot \frac{1}{2} + 0) \cdot \frac{1}{2} + 0) \cdot \frac{1}{2} + 1) \cdot \frac{1}{2}$, ren omskrivning

$= ((0.5 \cdot \frac{1}{2} + 0) \cdot \frac{1}{2} + 1) \cdot \frac{1}{2}$, inderste parentes udregnes,

$= (0.25 \cdot \frac{1}{2} + 1) \cdot \frac{1}{2}$, næste parentes udregnes

$= (0.125 + 1) \cdot \frac{1}{2}$

$= 1.125 \cdot \frac{1}{2}$

$= 0.5625$

Ved omsætningerne, der er vist her, er de givne brøker opfattet som eksakte, uafkortede tal, og det er derfor fyldestgørende at skrive

$0.5625 \longrightarrow 0.1001$
$0.1001 \longrightarrow 0.5625$.

Tænker man sig derimod, at de givne tal er almindelige, afrundede værdier, bør man skrive:

$0.5625 \longrightarrow 0.1001000000000 \pm 2^{-14}$
$0.1001 \longrightarrow 0.56 \pm 0.03$

Bemærk iøvrigt, at en vilkålig endelig binalbrøk altid omsættes tilen endelig decimalbrøk, mens det omvendte ikke gælder.

Udover det binære (og decimale) system får vi også brug for det sedicimale system, d.v.s. 16-talsystemet (undertiden kaldet det hexadecimale). Dette fremkommer ved, at 4 bimaler slås sammen til eet ciffer. De ciffersymboler, man kommer til at mangle, er kaldt 0 1 2 3 4 og 5.

10-talsystem	2-talsystem	16-talsystem
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	0
11	1011	1
12	1100	2
13	1101	3
14	1110	4
15	1111	5
16	10000	10

Repræsentation af tal i GIER

De 40 binære positioner i en helcelle eller i et register er, som vi ved, nummereret fra 0 til 39. Placerer vi kommaet mellem pos. 0 og 1, og betegnes indholdet i pos. j med p_j, vil et vilkårligt indhold i de 40 positioner umiddelbart have værdien

$$p = \sum_{j=0}^{39} p_j \cdot 2^{-j}$$

Dette giver tal i intervallet $0 \leq p \leq 2 - 2^{-39}$. Vi går imidlertid frem på en lidt anden måde, således at vi, uden anden information end den de 40.pos. kan give, også får negative tal med, mod tilgængæld at indskrænke værdiområdet numerisk, nemlig til $-1 \leq p \leq 1 - 2^{-39}$. Tal i dette område kaldes for maskintal. Tænker vi på et maskintal, p, skrevet som binærbrøk med sædvanligt fortegn, + eller -, repræsenteres det på følgende måde i maskinen:

i celler eller registre med 40 positioner:

når $p \geq 0$ ved p's binære cifre

når $p < 0$ ved cifrene i $2^1 + p$ (= 10 + p)

Eks: $p = +\ 0.1100\ \ldots\ldots\ 0$ (= + 0.75)

vil forstås som

$0.1100\ \ldots\ldots\ 0$

$p = -\ 0.0110\ \ldots\ldots\ 0$ (= - 0.375)

$= \quad 1.1010\ \ldots\ldots\ 0 - 2^1$ (= 1.625 - 2)

vil stå som

$1.1010\ldots\ldots\ 0.$

Vi ser, at der for maskintal gælder, at cifret foran kommaet altid er 0, når $p \geq 0$ og altid 1, når $-1 \leq p < 0$. Dette ciffer kaldes derfor fortegncifret og vil sammen med de øvrige cifre i registret entydigt bestemme p's størrelse og fortegn. Når vi altså har et indhold i en celle: $P_0.P_1P_2 \ldots\ P_{39}$ er værdien:

når $P_0 = 0$ simpelthen $\sum_{0}^{39} P_j \cdot 2^{-j}$,

når $P_0 = 1$ derimod $\sum_{0}^{39} P_j \cdot 2^{-j} - 2^1$,

hvilket samlet kan skrives

$$\sum_{j=0}^{39} P_j \cdot 2^{-j} - 2P_0 .$$

Ønsker man at skifte fortegn på et tal, som står i GIER, gøres det simpelthen ved at udskifte alle 1-taller med nul og alle nuller med et 1-tal og derefter addere 1 i sidste position - eller anderledes sagt: at udskifte alle cifre med deres komplementære, begyndende fra venstre og til, men ikke med, den position der indeholder sidste betydende ciffer (1-tal). Dette kaldes at danne tallets komplement i det pågældende register.

Eks. Idet vi nøjes med 10 positioner i registret, er:

$p = 0.101101100$

$-p \sim 1.010010011 + 0.000000001$

$= 1.010010100.$

Alt i det foregående svarer nøje til forholdene i almindelige bord-regnemaskiner, hvor negative tal i et register (med fuld menteoverføring) repræsenteres ved 10-talskomplementet regnet i forhold til 1-tal i en tænkt position, umiddelbart tilvenstre for registrets yderste position. Iøvrigt svarer det også til fremstillingsmåden ved negative logaritmer.

Vi gør tilsidst opmærksom på, at vor fremstilling af tal, når der er brug for det, selvfølgelig kan udvides til registre med flere positioner. Er der f.eks. to positioner foran kommaet, ialt 41 positioner, repræsenteres et maskintal, p, ved

$$\text{når } p \geq 0 \quad : \quad p\text{'s binære cifre,}$$

$$\text{når } p < 0 \quad : \quad \text{cifrene i } 2^2 + p \ (= 100+p).$$

I eksemplerne ovenfor bliver det henholdsvis

00.1100 0

og 11.1010 ..... 0.

Addition

I totalsystemet gælder følgende regneregler for addition:

0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 10.

Som eksempel vises addition af to maskintal, idet der dog kun regnes med 9 pladser efter kommaet:

00	0	1	2	3	4	5	6	7	8	9
0	0.	1	1	0	1	0	1	1	0	0
+0	0.	1	0	1	1	0	1	0	1	0
0	1.	1	0	0	0	1	0	1	1	0

Dette eksempel giver os lejlighed til at påpege visse forhold ved maskinens talrepræsentation udover det, der er indeholdt i additionsreglerne. Er begge addender som vist i eksemplet større end en halv, fås ved additionen et resultat, der er større end 1, og som altså ikke er et maskintal. Hvis resultatet står i R-registret, vil det ikke desto mindre på grund af R_{00} opfattes rigtigt, når læser det umiddelbart uden at tage hensyn til den foran omtalte fortegnskonvention. Ved forskydning af R's indhold 1 plads til højre fremkommer den halverede sum i R_0 - 39.

Når resultatet af en operation ikke ligger i intervallet $+1 \leq x < 1$, siger man, at der er overløb.

Vi kan opstille følgende regler:

1: Hvis R_{00} og R_0 er forskellige er der overløb.

2: R_{00} viser det rigtige fortegn. (Da addition af to positive maskintal altid giver et resultat mindre end 2, vil R_{00} aldrig blive"forkert" ved een additionsoperation).

Det overlades til læseren selv at overbevise sig om, at disse regler gælder ved en vilkårlig kombination af positive og negative tal.

Hvis resultatet af ovenstående addition ikke dannes i R men i en celle, f.eks. ved AC-ordren, lagres kun positionerne 0 til 39. Tallet vil ved senere regninger opfattes som et negativt tal, hvis additionen har bevirket overløb. Samtidig med operationens udførelse kan vi dog registrere overløb og det rigtige fortegn for resultatet ved hjælp af indikatoroperationerne.

Vi modificerer derfor regel 2:

I R-registret er R_{00} fortegnsvisende, i cellen er bit 0 fortegnsvisende.

Subtraktion

I GIER foretages subtraktion ved at subtrahenden byttes ud med sit komplement, hvorefter processen foregår som en addition.

Multiplikation

I totalsystemet kan multiplikation foretages analogt til regning i titalssystemet. Som eksempel vises opstilling for 13 x 5 = 65.

$$\begin{array}{r} 1101 \times 101 \\ 1101 \\ \underline{1101} \\ 1000001 \end{array}$$

1000001 er netop 65 skrevet binært.

Ved multiplikation er det dog nødvendigt at tage særlig hensyn til maskinens fortegnskonvention. Vi skal derfor omtale beregningsgangen lidt nærmere. Multiplikation udføres med multiplikator b i M og multiplikand a i en celle.

Der er to former:

Lang multiplikation, hvor produktet plus indholdet af R multipliceret med 2^{-39} sættes uafkortet i R,M og hvor multiplikator går tabt.

Kort multiplikation, hvor det til 39 binaler afrundede produkt plus indholdet af R sættes i R, og multiplikator er bevaret i M.

Vi betragter den lange multiplikation nøjere idet vi erindrer om, at et maskintal kan skrives som: $\sum_{j=1}^{39} b_j \cdot 2^{-j} - b_0$.

$$ab = \sum_{j=1}^{39} ab_j \cdot 2^{-j} - ab_0 = (\ldots(((ab_{39})\tfrac{1}{2} + ab_{38})\tfrac{1}{2} + ab_{37})\tfrac{1}{2} + \ldots + ab_1)\tfrac{1}{2} - ab_0.$$

Multiplikationen foregår derfor på den måde, at $a.M_{39}$ adderes til R. *) Derefter forskydes R,M (incl M_0) een plads til højre. (Herved dannes $(c(R) + ab_{39})\frac{1}{2}$ uanset overløb ved addition). Denne proces udføres 39 gange, hvorefter $-ab_0$ adderes til R, og M alene forskydes een plads til højre. Det overlades til læseren selv at overbevise sig om, at denne kalkyle netop giver det ønskede resultat.

Ved kort multiplikation "gemmes" indholdet af R, hvori der derefter sættes 0,100...0 ($=\frac{1}{2}$) af hensyn til afrunding. Multiplikationen udføres som før, idet dog R_{39} går tabt ved hver forskydning, og der i M foretages cyklisk skift. Til sidst adderes det oprindelige indhold af R.

Division

Også division udføres på sædvanlig måde i totalssystemet, f.eks. 47:5=9, rest 2.

```
101 | 101111 | 1001 = 9
    101
     01
      11
      111
      101
       10 = 2
```

Division udføres med dividenden a i R (kort division) eller i R,M (lang division) og med divisor b i en celle. Kvotienten fås i R og resten fås i M.

Maskinens divisionskalkyle er beslægtet med ovenstående opstilling, idet den på lignende måde undersøger, om man kan trække divisor fra "resten og næste ciffer" indtil alle cifre er trukket ned. (Papirberegningens gradvise forskydning mod højre er naturligvis i maskinen erstattet af venstreskift i et register af dobbelt længde). Når subtraktionen " kan udføres ", er næste ciffer i kvotienten 1, ellers 0. Tilsidst står resten tilbage på divisors plads (med enhed svarende til sidste ciffer), hvorefter ombytning af registre anbringer kvotienten og resten på deres plads.

Hvis kvotienten er negativ(i så fald sættes R_{00} lig 1) skal resten af kvotienten indeholde 2-komplementet til kvotienten. Dette opnås ved at udbytte dividenden a med 2b+a inden den egentlige division udføres.

Hvis kvotienten ikke ligger i intervallet $-1 \leq q < 1$ fremkommer der altid overløb.

Det må dog yderligere bemærkes, at maskinen undersøger, om man kan trække divisor fra"resten og næste ciffer"ved at forlange, at den nye rest har samme fortegn som divisor. Hvis divisor er negativ, kan man derfor ikke få resten nul!

*) Faktisk foregår alle regninger i et hjælperegister H og ikke i R. Første trin i beregningen er derfor at H sættes lig R. Dette har imidlerid ingen betydning for princippet i regningerne, og der er derfor set bort fra dette forhold i denne fremstilling.

Specielle forhold ved GIER-aritmetikken.

Det bemærkes, at (-1) (-1) = 1 med overløb. Hvis produktet læses til en celle uden videre, vil der derfor i cellen komme tallet -1.

På grund af den særlige restdefinition vil division med negativ divisor aldrig gå op. Hvis divisionen faktisk går op, får man en rest lig divisor gange 2^{-39}, og kvotienten bliver 2^{-39} for lille. Man har f.eks. (a>0):

0/-a = 11.111... Rest = -a (2^{-39})
a/-a = 10.111... Rest = -a (2^{-39})
-a/-a = 00.111... Rest = -a (2^{-39})

Da 2 ligger uden for det værdiinterval, der kan være i R har man endvidere:

2a/ a = 01.111... Rest = a (2^{-39})

Det ses, at ved addition af 2^{-39} fås i alle tilfælde netop det naturlige resultat. Divisionerne 0/a, a/a og 2a/-a giver alle det rigtige resultat. (Når dette også gælder 2a/-a, hvor divisor er negativ, skyldes det, at der divideres op i et nul i henhold til bemærkningen om 2-komplementet.)

Fast og flydende komma.

Vi har omtalt den talværdi indholdet i en celle eller et register tillægges, en værdi i intervallet $-1 \leq t < 1$. (I R $-2 \leq t < 2$). Da det imidlertid er klart, at man ikke kan være begrænset til at behandle opgaver, hvor tallene af sig selv ligger mellem -1 og +1, skal vi nu se, hvordan man sætter sig ud herover. Der vil blive antydet ialt tre veje at gå:

1. Anvendelse af skalafaktorer.
Det simpleste er tilfælde, hvor man forlods kan forsyne udgangsstørrelserne, konstanter som variable, med sådanne faktorer, skalafaktorer, at udgangsværdierne såvel som mellemregninger og slutresultater alle bliver maskintal. Som eksempel vil vi lægge 2 og 3 sammen. Det ses, at hvis begge addender ganges med 10^{-1}, får vi udregningen 0.2 + 0.3 = 0.5, altså en regning med maskintal, og facit bliver det rigtige facit ganget med 10^{-1}. (NB. Det er selvfølgelig ikke på forhånd givet, at facit kan gøres til maskintal med samme skalafaktor som addenderne. Vælger koderen at anvende skalafaktorer, må han ved et overslag sikre sig, at resultatet (og mellemregninger!) er maskintal.) Hvis problemet ikke er 2+3 men 2 3 fås på samme måde 0.2 0.3 = 0.06, hvilket facit naturligvis ikke har samme skalafaktor som faktorerne. Ved multiplikation og division (0.3:0.2 = 1.5) med tal med skalafaktor er det derfor nødvendigt, at tage ekstra forholdsregler.

2. Heltal.
Et specielt tilfælde af skalafaktor er 2^{-39}. Herved opfattes celleindholdet som heltal med enhed i pos. 39. Fordelen herved er, at man netop indfører faktoren 2^{39} ved at opfatte M som et selvstændigt tal efter regninger i R,M (lang multiplikation). På grund af M 's særlige stilling, er det dog en forudsætning, at facit er positivt, ligesom heltalsdivision med negativ dividend ikke kan udføres uden videre. Eksempler på regning med heltal gives i eksempelkapitlet i lærebogen.

3. Flydende tal.
Ofte kan regningerne imidlertid ikke tilrettelægges så at anvendelse af skalafaktor er hensigtsmæssig. Man gør da det, at man forsyner hvert tal med sin skalafaktor, idet ethvert tal kan skrives på formen

$$x = p\,2^q, \text{ hvor } -2 \leq p < -1 \text{ eller } p = 0 \text{ eller } 1 \leq p < 2.$$

Hvert tal er altså repræsenteret i maskinen ved to tal (der kan lagres i hver sin del af samme celle). Det er indlysende, at multiplikation og division let udføres med tal på denne form, medens addition er mere omstændeligt. I mange maskiner kræver denne regnemåde stor plads til koden og lang regnetid. I GIER er flydende regning derimod indbygget, således at de vigtigste operationer, blot de er F-mærkede, opfatter celleindholdet som de to tal, p og q, og regner overensstemmende hermed.