Tölvutækni og forritun Heimadæmi 5

Ragnar Björn Ingvarsson, rbi3

21. september 2024

1

a) i. **EF**

ii. **CDEF**

iii. **0123**

iv. **01234567**

b) i. **01**

ii. **0123**

iii. **CDEF**

iv. **89ABCDEF**

2

a) Við sjáum að við keyrslu fæst þetta

```
Ítra með fleytitölum frá 0.00 til 1.00, upphækkun 0.10
val = 0.000000000000000;  sum = 0.000000000000000
val = 0.100000001490116;  sum = 0.100000001490116
val = 0.200000002980232;  sum = 0.300000011920929
val = 0.300000011920929;  sum = 0.600000023841858
val = 0.400000005960464;  sum = 1.000000000000000
val = 0.500000000000000;  sum = 1.500000000000000
val = 0.600000023841858;  sum = 2.099999904632568
val = 0.700000047683716;  sum = 2.799999952316284
val = 0.800000071525574;  sum = 3.599999904632568
val = 0.900000095367432;  sum = 4.500000000000000
val = 1.000000119209290;  sum = 5.500000000000000
```

En forritið átti að stoppa í 1.0.

Vandamálið hér er að í for lykkjunni er athugað hvort núverandi gildi sé **jafnt** og 1.0 en með fleytitölum fáum við ekki svoleiðis nákvæmni. Við sjáum af myndinni að þegar val er rúmlega 1.0 er það í raun gildið 1.000000119209290 sem c telur ekki vera jafnt og 1.0. Fleytitalan er svona ónákvæm vegna þess að hún hefur byggt upp skekkju með því að reyna að sýna 0.1, 0.2 o.s.frv. sem geta ekki verið sýndar fullkomlega með fleytitölum. Þess vegna er skilyrði for lykkjunnar ennþá satt og við höldum endalaust áfram.

b) Með því að breyta í double fæst

```
Ítra með fleytitölum frá 0.00 til 1.00, upphækkun 0.10
val = 0.000000000000000;  sum = 0.000000000000000
val = 0.100000000000000;  sum = 0.100000000000000
val = 0.200000000000000;  sum = 0.300000000000000
val = 0.300000000000000;  sum = 0.600000000000000
val = 0.400000000000000;  sum = 1.000000000000000
val = 0.500000000000000;  sum = 1.500000000000000
val = 0.600000000000000;  sum = 2.100000000000000
val = 0.700000000000000;  sum = 2.800000000000000
val = 0.800000000000000;  sum = 3.600000000000000
val = 0.900000000000000;  sum = 4.500000000000000
val = 1.000000000000000;  sum = 5.500000000000000
val = 1.100000000000000;  sum = 6.600000000000000
```

Við sjáum að með því að breyta í double höfum við vissulega meiri nákvæmni en forritið heldur ennþá endalaust áfram þar sem þó meiri nákvæmni sé góð, er hún ekki fullkomin, svo skekkjan er ennþá til, hún er bara minni. Þess vegna helst skilyrði for lykkjunnar satt og hún heldur áfram.

3

a) 1.01100

b) Með því að rúnna að næstu tölu viljum við fá að síðasti bitinn sé 0 en ekki 1, semsagt slétt tala. Þess vegna fáum við 1.10, semsagt 1.5.

Við höfðum tvo möguleika á rúnnuninni þar sem parturinn sem við vildum rúnna er mitt á milli tveggja gilda, 1.01 og 1.10, þ.e. 1.25 eða 1.50. Þar sem við viljum rúnna að sléttum LSB er valin 1.50 þar sem 1.01 endar á ás svo við veljum frekar 1.10.

4

Við erum semsagt með 10 bita með einn formerkisbita fremst en vitum ekki hvernig skiptingin er á veldishluta og brothluta.

a) Við byrjum á að breyta 3.5625 í binary og fáum 11.1001. Síðan, þar sem IEEE vill hafa tölur á formi $(-1)^{formerkisbiti} \cdot 2^{(e_m e_{m-1} ... e_0)_2 - 2^{m-1}-1} \cdot 1.b_n b_{n-1} ... b_0$ þá shiftum við tölunni um einn bitastað og setjum hana fram sem $2 \cdot 1.11001$. Þá erum við komin með töluna á rétt form og við sjáum að brothlutinn er að lágmarki fimm bitar, 11001.

b) Þar sem við erum með 5 bita í brothlutann og 1 bita í formerkið, þá erum við með 4 bita fyrir veldishlutann. Þá fáum við að form tölunnar er $(-1)^{formerkisbiti} \cdot 2^{(e_m e_{m-1} ... e_0)_2 - 2^{4-1}-1} \cdot (1.b_n b_{n-1} ... b_0)_2$. Þar sem við fáum $2 \cdot 1.11001_2$ í a lið fæst:

$$(-1)^0 \cdot 2^{1000_2 - 7} \cdot 1.11001_2$$

$$= 2^{8-7} \cdot 1.11001_2$$

$$= 2 \cdot 1.11001_2$$

Svo talan er skilgreind svo, 0 í formerkisbita, 1000 í veldishluta og 11001 í brothluta, eða 0100011001 alls.

c) Það er hægt að tákna staðlaðar tölur á þessu formi, þar sem veldishlutinn getur verið samblanda af núllum og ásum.

Hæsta staðlaða talan er þá 0111111101 sem gefur

$$(-1)^0 \cdot 2^{11111110_2 - 127} \cdot 1.1_2$$
$$= 2^{254-127} \cdot 1.5$$
$$= 2^{127} \cdot 1.5$$

d) Hér er ekki hægt að tákna staðlaðar tölur þar sem aðeins er einn veldisbiti. Hann er annaðhvort 0 eða 1, sem í báðum tilfellum er allur veldishlutinn sama talan, sem gerir töluna ekki staðlaða.

5

a) Notum til dæmis IEEE töluna 0000010000000001_2. Við breytum henni í decimal með

$$(-1)^0 \cdot 2^{00001_2 - 2^{5-1} - 1} \cdot 1.0000000001_2$$
$$= 2^{1-15} \cdot (1 + 2^{-10})$$
$$= 2^{-14} \cdot (1 + 2^{-10})$$
$$= 0.00006109476$$

Ef við reynum svo að breyta henni í bfloat, þá sjáum við að við getum haft eins veldishluta, þá 00010000, en ekki eins brothluta þar sem við erum aðeins með 7 bita miðað við 10 í IEEE. Þá er næsta talan sem við getum fengið 0000100000000000 eða

$$(-1)^0 \cdot 2^{01110001_2 - 2^{8-1} - 1} \cdot 1.0000000_2$$
$$= 2^{113-127} \cdot 1$$
$$= 2^{-14}$$
$$= 0.00006103516$$

Sem er nálægt réttu tölunni, en getur aldrei verið nógu nákvæm vegna þess að bfloat er bara með minni nákvæmni.

b) Hér viljum við þá nota styrkleika bfloat sem er stórar tölur, svo við tökum töluna 0111111100000000.

$$(-1)^0 \cdot 2^{11111110_2 - 2^{8-1} - 1} \cdot 1.0000000_2$$
$$= 2^{254-127} \cdot 1$$
$$= 2^{127}$$

Hér hefur IEEE ekki rað í töluna þar sem hæsta talan sem fæst úr því er 0111101111111111.

$$
\begin{aligned}
&(-1)^0 \cdot 2^{11110_2 - 2^{5-1} - 1} \cdot 1.1111111111_2 \\
&= 2^{30-15} \cdot 1.9990234375 \\
&= 2^{15} \cdot 1.9990234375 \\
&= 65504
\end{aligned}
$$

Sem er langt frá 2^{127}.