

Tölvutækni og forritun Heimadæmi 5

brj46

September 2024

1

Segjum að 64-bitu orðið $0x0123456789ABCDEF$ sé geymt í minnishólfi númer $0x100$.

a)

Í lágenda bætarið þá eru minnstu bætin geymd í minnishólfinu með lægstu tölunni svo talan $0x0123456789ABCDEF$ verður geymd eins og í eftirfarandi töflu:

Minnishólf	Gildi
$0x100$	$0xEF$
$0x101$	$0xCD$
$0x102$	$0xAB$
$0x103$	$0x89$
$0x104$	$0x67$
$0x105$	$0x45$
$0x106$	$0x23$
$0x107$	$0x01$

Gerum ráð fyrir lágenda bætarið og svörum eftirfarandi spurningum:

i) Hvert er gildið á bætinu í hólfi númer $0x100$

Svar : $0xEF$

ii) Hvað er gildið á 16-bitu orðinu í hólfi númer $0x100$?

Svar : 16-bitu orð eða 2 bæti í hólfi $0x100$ væri þá $0xCDEF$

iii) Hvað er gildið á 16-bitu orðinu í hólfi númer $0x106$?

Svar : 16-bitu orðið eða 2 bæti í hólfi $0x106$ væri þá $0x0123$

iv) Hvað er gildið á 32-bitu orðinu í hólfi númer $0x104$?

Svar : 32-bitu orðið eða 4 bæti í hólfi $0x104$ væri þá $0x01234567$

b)

Í háenda bætarið þá eru stærstu bætin geymd fremst í minnishólfinu svo talan $0x0123456789ABCDEF$ væri geymd eins og í eftirfarandi töflu:

Minnishólf	Gildi
0x100	0x01
0x101	0x23
0x102	0x45
0x103	0x67
0x104	0x89
0x105	0xAB
0x106	0xCD
0x107	0xEF

Gerum ráð fyrir háenda bætarið og svörum eftirfarandi spurningum:

i) Hvert er gildið á bætinu í hólfi númer 0x100

Svar : 0x01

ii) Hvað er gildið á 16-bitu orðinu í hólfi númer 0x100??

Svar : 16-bitu orðið í hólfi númer 0x100 væri 0x0123

iii) Hvað er gildið á 16-bitu orðinu í hólfi númer 0x106?

Svar : 16 bitu orðið í hólfi númer 0x106 væri 0xCDEF

iv) Hvað er gildið á 32-bitu orðinu í hólfi númer 0x104?

Svar : 32-bitu orðið í hólfi númer 0x104 væri 0x89ABCDEF

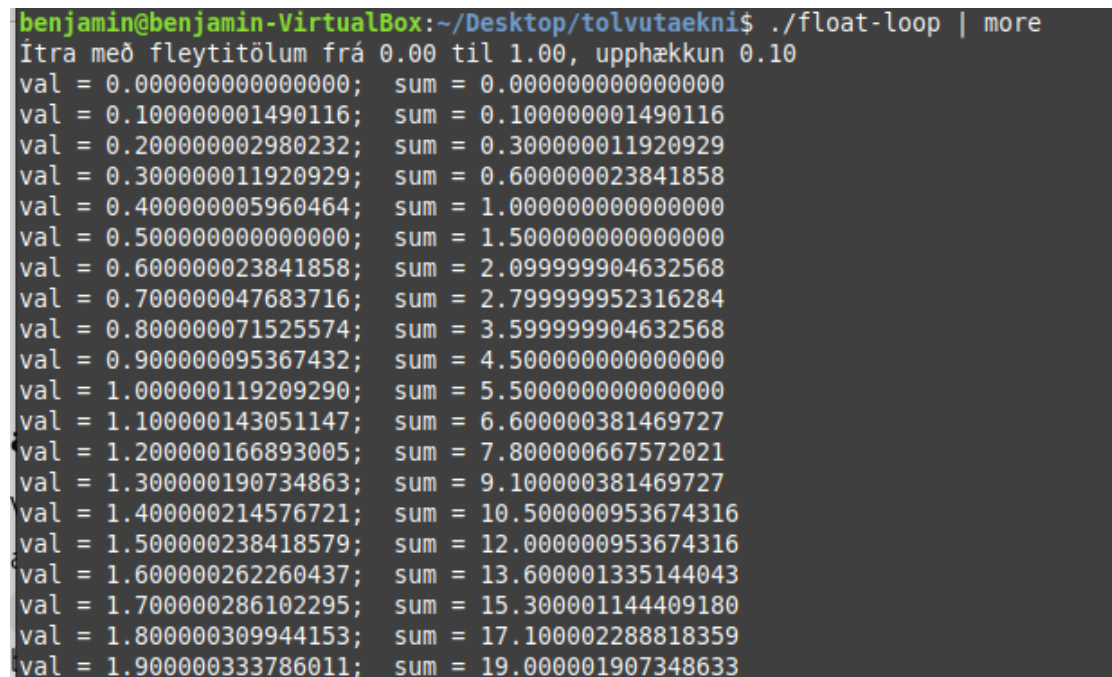
2

Við fáum gefið forritið float-loop.c

```
#include <stdio.h>
int main ()
{
    float val = 0.0;
    float end = 1.0;
    float inc = 0.1;
    float sum = 0.0;
    printf ("Ítra með fleytitölum frá %.2f til %.2f, upphækkun %.2f\n", val, end, inc);
    for (val=0.0; val!=end; val+=inc) {
        sum += val;
        printf ("val = %.15f; sum = %.15f\n", val, sum);
    }
    printf ("Lykkju lokid með val = %.2f; sum = %.2f\n", val, sum);
}
```

forritið á að ítra frá 0.0 upp í 1.0 með 0.1 í hverri ítrun lykkjunnar. Þetta virkar ekki eins og hefði búist við og við að keyra forritið kemur upp villa.

sjá skjáskot:



```
benjamin@benjamin-VirtualBox:~/Desktop/tolvutaekni$ ./float-loop | more
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 = 0.3000000011920929
val = 0.3000000011920929; sum = 0.6000000023841858
val = 0.400000005960464; sum = 1.000000000000000
val = 0.500000000000000; sum = 1.500000000000000
val = 0.6000000023841858; sum = 2.099999904632568
val = 0.7000000047683716; sum = 2.799999952316284
val = 0.8000000071525574; sum = 3.599999904632568
val = 0.9000000095367432; sum = 4.500000000000000
val = 1.000000119209290; sum = 5.500000000000000
val = 1.100000143051147; sum = 6.600000381469727
val = 1.200000166893005; sum = 7.800000667572021
val = 1.300000190734863; sum = 9.100000381469727
val = 1.400000214576721; sum = 10.500000953674316
val = 1.500000238418579; sum = 12.000000953674316
val = 1.600000262260437; sum = 13.600001335144043
val = 1.700000286102295; sum = 15.300001144409180
val = 1.800000309944153; sum = 17.100002288818359
val = 1.900000333786011; sum = 19.000001907348633
```

Út frá þessum upplýsingum skulum við svara dæmi 2 a.

a)

Forritið hegðar sér ekki eins og við myndum búast við. Útskýrum í nokkrum orðum hvert vandamálið er. Útþrentunin í lykkjunni hjálpar að sjá vandamálið

Við sjáum að vandamálið er tvennskonar, Lykkjan stöðvar ekki eftir 1.0 og lykkjan virðist ekki hækka nákvæmlega um 0.1 við hverja ítrun.

Þetta er vegna þess að verið er að nota fleytitölur **Float** sem nota bara 32 bita í minninu meðan ef verið væri að nota **Double** þá væri verið að nota 64 bita í minninu svo Double er nákvæmari.

b)

Breytið öllum gagnatögnum frá float yfir í double. Breytist eitthvað við það? Útskýrið.

Við að skipta út öllum gagnatögum úr float yfir í double fáum við mun betri niðurstöður

Sjá skjáskot:

```
benjamin@benjamin-VirtualBox:~/Desktop/tolvutaekni$ ./float-loop | more
Ítra með fleytitölum frá 0.00 til 1.00, upphækkun 0.10
val = 0.0000000000000000; sum = 0.0000000000000000
val = 0.1000000000000000; sum = 0.1000000000000000
val = 0.2000000000000000; sum = 0.3000000000000000
val = 0.3000000000000000; sum = 0.6000000000000000
val = 0.4000000000000000; sum = 1.0000000000000000
val = 0.5000000000000000; sum = 1.5000000000000000
val = 0.6000000000000000; sum = 2.1000000000000000
val = 0.7000000000000000; sum = 2.8000000000000000
val = 0.8000000000000000; sum = 3.6000000000000000
val = 0.9000000000000000; sum = 4.5000000000000000
val = 1.0000000000000000; sum = 5.5000000000000000
val = 1.1000000000000000; sum = 6.6000000000000000
val = 1.2000000000000000; sum = 7.8000000000000000
val = 1.3000000000000000; sum = 9.1000000000000000
val = 1.4000000000000000; sum = 10.5000000000000000
val = 1.5000000000000000; sum = 12.0000000000000000
val = 1.6000000000000000; sum = 13.6000000000000000
val = 1.7000000000000000; sum = 15.3000000000000001
val = 1.8000000000000000; sum = 17.1000000000000001
val = 1.9000000000000001; sum = 19.0000000000000004
```

Þetta er vegna þess að nú er hægt að geyma mun fleirri aukastafi í minninu og verður því niðurstaðan nákvæmari. Við sjáum hinsvegar að hún er ekki alveg fullkominn enda er ekki hægt að ná fullkomnri nákvæmni.

3

Gefið er almenna brotið $\frac{11}{8} = 1.375$

a)

sýnum brotið sem tvíundartölu með 5 bita fyrir aftan kommu.

Táknum heiltöluna 1 í 1.375 sem 1

næsti finnum við út úr brotahlutanum:

$$0.375 \times 2 = 0.75 \rightarrow \text{heiltöluhlutinn verður að 0}$$

$$0.75 \times 2 = 1.5 \rightarrow \text{heiltöluhlutinn verður að 1}$$

$$0.5 \times 2 = 1 \rightarrow \text{heiltöluhlutinn verður að 1}$$

Svo brotahlutinn 0.375 verður að 0.011₂.

Þar sem við eigum að tákna töluna með 5 bita fyrir aftan kommu þá bætum við tveim núllum að aftan:

$$1.01100_2$$

b)

Rúnnum nú þessa tölu niður í 3 bita þannig að það séu bara 2 bitar fyrir aftan kommu.

1. finnum rúnnunarbitann sem er þá þriðji bitinn fyrir aftan kommu eða 1.01*1*00
2. Rúnnum að næsta jöfnu. Þegar rúnnunarbitinn er 1 eins og í þessu tilviki þá skoðum við hvort næsta rúnnaða tala sé jöfn eða oddatala
Ef næsta rúnnaða tala er jöfn þá rúnnum við niður og ef næsta rúnnaða tala er oddatala þá rúnnum við upp.
3. rúnnum 1.01100₂ þar sem GRS er 110 þá hækkum við og fáum 1.10₂
4. Niðurstaða:

Tvíndarform: 1.10₂

Tugaform: 1.5

4

Við höfum 10-bitu fleytitölur sem fylgja IEEE staðlinum. Við vitum ekki skiptingu þeirra í veldishluta (exp) og brothluta (frac), en það er einn formerkisbiti fremst í þeim.

a)

Hver er lágmarks bitafjöldi í brothluta fleytitölunnar til að hægt sé að tákna töluna $\frac{9}{16} = 3.5625$ nákvæmlega á þessu formi?

Umbreytum tölunni 3.5625 yfir í tvíundarform:

Heiltöluhluturinn er 3 eða í tvítöluformi: 11_2

Brotahlutinn er 0.5625 gerum þá eftirfarandi reikniaðgerðir

1. $0.5625 \times 2 = 1.125$ fáum 1
2. $0.125 \times 2 = 0.25$ fáum 0
3. $0.25 \times 2 = 0.5$ fáum 0
4. $0.5 \times 2 = 1.0$ fáum 1

Svo við táknum 3.5625 sem 11.1001_2

táknum á IEEE formi sem fylgir formúlunni

$$v = (-1)^s \times M \times 2^E$$

þar sem s er formerkisbitinn, xxx er brothlutinn og E er veldisgildið

talán sem við fengum verður þá $1.11001_2 \times 2^1$

Svo 5 bitar er lágmarks bitafjöldi í brothluta fleytitölunnar.

b)

Frá lið a höfum við töluna $1.11001_2 \times 2^1$

Svo brotahlutinn er : 11001 og veldishlutinn er 1

Röðum þessu nú í 10-bitu IEEE fleytitölusniði

Formerkisbitinn (s) er 0 þar sem talan er jákvæð.

Veldishlutinn er táknaður næst með þrem biased bitum og hliðrunin fyrir 3 bita er

$2^{3-1} - 1 = 3$ og verður því

$$E_{biased} = 1 + 3 = 4 = 100_2$$

við vitum að **brothlutinn** er 11001.

Niðurstaða:

0 100 11001

c)

Segjum að í þessum 10-bita fleytitölum hefur verið ákveðið að nota einn bita fyrir formerki, einn bita fyrir brothluta og restina af bitunum fyrir veldishlutann er þá hægt að tákna staðlaðar tölur á þessu formi, ef svo er, hver er þá stærsta staðlaða talan sem hægt er að tákna?

Við erum með 10 bita fleytitölu:

- 1 biti fyrir formerki
- 1 biti fyrir brothluta
- 8 bitar fyrir veldishlutann

Notum jöfnuna fyrir staðlaða IEEE

$$v = (-1)^s \times M \times 2^{E_{exp}-bias}$$

Brothlutinn er aðeins 1 biti annaðhvort 0 eða 1.

Veldishlutinn er biased þar sem við erum að vinna með staðlað gildi. við höfum 8 bita fyrir veldishlutiann sem er þá reiknað svona:

$$bias = 2^{8-1} - 1 = 127$$

Við erum þá með jöfnuna

$$v = 1 \times 1.1 \times 2^{127} = 1.5 \times 2^{127}$$

Niðurstaða: Stærsta staðlaða talan sem hægt er að tákna með þessari fleytitölu er þá

$$v = 1.5 \times 1.7 \times 10^{38} = 2.55 \times 10^{38}$$

d)

Nú erum við með 10 bita fleytitölu þar sem við notum einn bita í formerki, einn bita í veldishluta og restina í brothlutann. Hver er stærsta staðlaða talan sem við getum táknað?

við táknum þá formerkishlutann þá annaðhvort með 0 fyrir jákvæðartölur og 1 fyrir neikvæðar tölur.

Fyrir Veldishlutann þá notum við 1 fyrir staðlaðar tölur og 0 fyrir óstaðlaðar tölur.

Brothlutinn er þá á bilinu 00000000 til 11111111

Svo brothlutinn væri 1.11111111_2 eða $1 + 1 - \frac{1}{256} = 1.99609375$

Við margöldum brothlutann síðan með 2^1 og fáum út 3.9921875

Niðurstaða: Stærsta Staðlaða talan sem hægt er að tákna með 10-bita fleytitölu þar sem 1 biti fer í formerki, 1 biti í veldishluta og restin í brothluta er:

$$3.9921875$$

5

IEEE staðalinn skilgreinir líka fleytitölur með hálfri nákvæmni, þ.e. 16-bita. Í þeim er einn bitur fyrir formerki, 5 bitar fyrir veldishluta (exp) og 10 bitar fyrir brothluta (frac). Google hefur skilgreint aðra skiptingu á 16-bita fleytitölu, sem kallast bfloat (eða brain floating point). Í þeim er einn formerkisbitur, 8 bitar í veldishluta og aðeins 7 bitar í brothlutanum. Helsti kosturinn við þær er að 16-bita bfloat getur táknað jafnstórar tölur og venjuleg 32-bita fleytitala (u.þ.b. 10 ± 38), en auðvitað er nákvæmnin (precision) talsvert minni.

a)

Sýnum eina tölu sem er hægt að tákna nákvæmlega á 16 bita IEEE forminu, en er ekki hægt að tákna nákvæmlega með bfloat.

Tökum töluna 1.0000000011_2 , þessi tala krefst 10 bita í brothlutanum til að vera nákvæm og við getum nákvæmlega táknað hana með IEEE 16-bita fleytitölunni en ekki með bfloat.

Reiknum: Éf við táknum töluna í bfloat þá þurfum við að rúnna niður brothlutann og væri þá brothlutinn 1.00000000_2 sem er bara 1.0 í tugarformi en ekki 1.00000012 eins og talan á að vera.

b)

finnum nú tölu sem er hægt að nákvæmlega lýsa með bfloat en ekki með IEEE 16-bita fleytitölu:

við vitum að bfloat hefur fleiri bita fyrir veldishlutann, svo sú aðferð getur táknað mun stærri tölur.

finnum stærstu tölurnar sem hægt er að tákna:

bfloat16 hefur 8 í veldishluta sem getur tekið gildi E frá -126 til 127

16-bita IEEE hefur 5 bita í veldishluta svo E gildið getur verið á bilinu -14 til 15

bfloat getur táknað töluna 2^{40} nákvæmlega, þar sem veldishlutinn er nógu stór til að geyma það. (gæti farið allt upp í 2^{127}) Þessi tala væri táknum með brothlutany 1.000000_2 og veldishlutanum 40

táknað:

0 10100111 0000000

16-bita IEEE getur ekki táknað 2^{40} , þar sem stærsta veldið sem hægt væri að tákna er 2^{15} , þannig þessi tala myndi fara út fyrir svið þess.