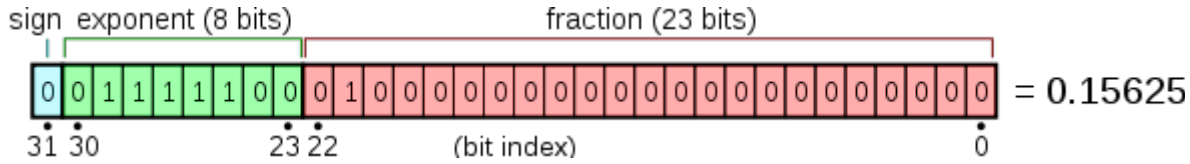
Extra arbete C-programmering
Lasse Karagiannis
2016-12-18

1 Extra arbete i C. Flyttalsrepresentation

IEEE 754 Single precision floating point format har studerats, genom att artikel har lästs på Wikipedia, samt körning av C-program som tillhandahållits av Jens Björnhager lärare på en EC-utbildning i Inbyggda System.

2 IEEE 754 Single precision floating point

Kodningen av ett decimal tal görs med 32 bitar där bit 31 är teckenbiten, bitarna 22 till 0 är en muliplicerande faktor den s.k. mantissan, och exponenten utgörs av bitarna 30 till 23, Se Figur 1



Exponenten skall ses som ett unsigned 8 bitars tal med s.k. *bias*, dvs. det ska subtraheras med 127 för att få en korrekt mening.
I exemplet från Figur så ska alltså exponenten 0111 1100 = 124, uppfattas som 124-127 = -3.

Mantissan (*fraction*) ska ses som 1.b22 b21....b0, om inte exponenten lagrats med enbart nollor.

Flyttalets värde fås genom följande matematiska uttryck

$$(-1)^{b_{31}} * 1.b_{22}b_{21}...b_0 * 2^{b_{30}b_{29}...b_{23}-127}$$

I exemplet ovan så fås alltså talet genom att multiplicera följande tre faktorer:

$(-1)^0 = 1$
$1.b_{22}\ b_{21}...b_0 = 1.25$
$2^{(124-127)} = 2^{-3} = 1/8 = 0.125$

Således fås 1*1.25*0.125 = 0.15625

3 Verifiering med C-program

Prgrom från tidigare utbildning har används för att verifiera IEEE-754 standarden. Författaren är Jens Björnhager från MotionControl AB.

Försök gjordes att köra detta C-program i Eclipse, men kunde inte begripa hur man skulle konfigurera sökvägen til GNU-kompilatorn. Detta skulle förmodligen tagit en hel dag i anspråk, tid som rapportskrivaren inte har tillgång till, så programmet körs företrädesvis i CodeBlocks, en lättviktsmiljö, som man kan hämta hem från:
Använd codeblocks-16.01mingw-setup.exe , så fås nedladdning och installation av mingw kompilatorn, annars får man försöka konfigurerar sökvägar och kompileringsflaggor själv. Något som man förvisso borde behärska, men inte ändå gör.

Programmet väntar på användarens input av ett decimaltal, skrivet med decimalpunkt, och skriver ut bitfältsrepresentationen på skärmen, genom att typkasta det float-pekaren pekar på till en int. Scanf vill ha adressen till input variabel av typ float vilket i C-syntaxen åstadkommes såsom &f.

Denna adress är av typen ”float-pekare”, som måste typkastas till en ”int-pekare” vilket åstadkommes med (int *)&f;
Därefter ska int-pekaren derefereras och tilldelas variablen i, så därför blir hela uttrycket för tilldelningen:
i = *(int *) &f;

När man har bitfältet som en int, så kan man använda bitvisa operationer.
Mask är en 32-bitars unsigned int med värdet b1000..0, vars MSB skiftas åt höger md uttrycket mask >> = 1, vilket är en kompaktare form av uttrycket mask = mask >> 1.
Hade denna inte varit unsigned så hade ettor skiftats in under högersskiften.

Därefter körs en select-sats där bitvis AND görs mellan int och mask. Om resultat resulterar i TRUE, så skrivs '1' annars '0'. Resten är textformatering för att urskilja teckenbiten, mantissan och exponenten.

.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    float f;
```

```
    int n,i;
    unsigned int mask;
    scanf("%f",&f);
    i = *(int *)&f;
    n = 0;
    for (mask = 0x80000000;mask != 0;mask>>= 1){
        if((n==1) || (n==9))
            putchar(' ');
        putchar((i & mask)? '1':'0');
        n++;
    }
    printf("\n");
    return 0;
}
```