Tugas Mandiri - 2

Pengantar Sistem Digital Semester Ganjil 2022/2023

Revisi 1.1

Petunjuk pengerjaan:

- Kerjakan dengan tulisan tangan atau diketik.
- Tuliskan Nama, Kelas, dan NPM pada setiap lembar jawaban.
- Tuliskan penjelasan dari cara mendapatkan jawaban tersebut.
- Apabila ditulis tangan, hasil pekerjaan di scan / foto dan dimasukan ke dalam satu file berformat .pdf.
- Format nama file (tanpa tanda kurung): [KodeAsdos]_TM2_[Nama]_[NPM].pdf.
- Tugas mandiri dikumpulkan Jumat, 16 September 2022 pukul 17.00 pada slot yang sudah disediakan di SCELE.
- Jika mengumpulkan telat sebelum pukul 23:59 pada hari yang sama, akan dikenakan penalti sebesar 50 poin. Terlebih dari waktu tersebut, tugas mandiri tidak akan dinilai

1. [24] Lengkapilah tabel berikut!

Desimal	BCD	Excess-3	2421	Excess-5
15	0000 0000	0000 0000	0000 0000	0000 0000
	0001 0101	0100 1000	0001 1011	0110 1010
290	0000 0010	0000 0101	0000 0010	0000 0111
	1001 0000	1100 0011	1111 0000	1110 0101
1050	0001 0000	0100 0011	0001 0000	0110 0101
	0101 0000	1000 0011	1011 0000	1010 0101
2142	0010 0001	0101 0100	0010 0001	0111 0110
	0100 0010	0111 0101	0100 0010	1001 0111



- 2. [16] Ubah bilangan desimal berikut menjadi heksadesimal *floating-point* IEEE-754 32-bit! **Jawab dengan cara pengerjaan!**
 - a. 1927 = **0x44F0E000**

$$(1927)_{10} = 1024 + 512 + 256 + 128 + 4 + 2 + 1$$

$$= 2^{10} + 2^{9} + 2^{8} + 2^{7} + 2^{2} + 2^{1} + 2^{0}$$

$$= (11110000111)_{2}$$

$$= 1.1110000111 \times 2^{10}$$

Sign Bit = 0

Exponential Bit = 127 + 10 = 137 = 1000 1001

Fraction Bit = 111 0000 1110 000

Heksadesimal: 0x44F0E000

b. 20.5 = **0x41A40000**

$$(20.5)_{10} = 16 + 4 + 0.5$$
$$= 2^{4} + 2^{2} + 2^{-1}$$
$$= (10100.1)_{2}$$
$$= 1.01001 \times 2^{4}$$

Sign Bit = 0

Exponential Bit = 127 + 4 = 131 = 1000 0011

Fraction Bit = 010 0100 0000 0000 0000 0000

IEEE-754 32 Bit : 0100 0001 1010 0100 0000 0000 0000

Heksade simal : 0x41A40000

c. -525 = 0xC4034000

$$(-525)_{10} = -(512 + 8 + 4 + 1)$$

$$= -(2^{9} + 2^{3} + 2^{2} + 2^{6})$$

$$= -(1000001101)_{2}$$

$$= -(1.000001101)_{2} \times 2^{9}$$

Sign Bit

Exponential Bit = 127 + 9 = 136 = 1000 1000

Fraction Bit = 000 0011 0100 0000 0000 0000

Heksade simal : 0x C40 34000

d. -2004.25 = 0xC4FA8800

$$(-2004.25)_{2} = -(1024 + 512 + 256 + 126 + 64 + 16 + 4 + 0.25)$$

$$= -(2^{10} + 2^{3} + 2^{8} + 2^{7} + 2^{6} + 2^{4} + 2^{-2})$$

$$= -(11111010100001)$$

$$= -(1.111101010001), \times 2^{10}$$

Sign Bit

Exponential Bit = 127 + 10 = 137 = 1000 1001

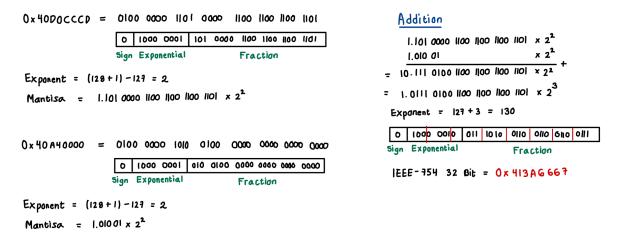
Fraction Bit = 111 1010 1000 1000 0000 6000

IEEE-754 32 Bit : 1100 0100 1111 1010 1000 1000 0000

Heksade simal : 0x C4 F A9800

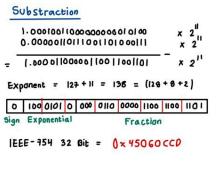
3. [18] Lakukanlah operasi aritmatika pada bilangan floating-point IEEE-754 32-bit berikut beserta hasilnya dalam heksadesimal IEEE-754 32-bit! Jawab dengan cara pengerjaan!

0x40D0CCCD + 0x40A40000 = 0x413A6667



b. 0x45098014 - 0x425CD1EC = 0x45060CCD

```
0 \times 45098014 = 0100 0101 0000 1001 1000 0000 0001 0100
                                                                       Substraction
                 0 1000 1010 000 1001 1000 0000 0001 0100
                                                                         0.001001100000000010100
                                                                        0.0000110111001101000111
                 Sign Exponential
Exponent = (128 + 8 + 2) - 127 = 11
 Mantisa = 1.00010011000000000101 x 211
                                                                     Sign Exponential
0x425CDIEC = 0100 0010 0101 1100 1101 0001 1110 1100
                 0 1000 0160 101 100 1101 0001 110 1100
                 Sign Exponential
                                        Fraction
Exponent = (128+4)-127=5
Mantisa = 1.1011100110100011110 |1 x 25
          = 0.00000110111001101000111 × 2
```



- 4. [22] Saat kita melakukan operasi aritmatika antara dua bilangan *floating-point* IEEE-754, salah satu titik desimal *mantissa* harus digeser jika eksponen kedua *mantissa* tidak sama.
 - a. Apakah arah pergeseran titik desimal (ke kanan atau kiri) jika mantissa yang di-shift memiliki eksponen yang lebih kecil?
 Titik desimal akan digeser atau di-shift ke arah kiri.
 - b. Mengapa titik decimal harus digeser dalam kasus ini? Tujuan digesernya titik desimal yaitu agar kedua mantissa memiliki nilai eksponen yang sama dan mempertahankan nilai leading bit. Lalu, kenapa harus digeser ke arah kiri?

Karena *mantissa* terbatas hanya menyimpan 24 bit (dengan 23 bit *fraction*) saja, maka lebih baik kita **menghilangkan bit paling kanan** (LSB) dibandingkan menghilangkan leading bit. Ketika digeser ke arah kiri pun maka nilai eksponen juga akan membesar hal ini menyebabkan nilai eksponen yang lebih kecil akan membesar sampai sama dengan nilai eksponen yang satunya.

Contoh:

Jika N1 dan N2 dilakukan operasi aritmatika maka perlu disetarakan dulu eksponen nya dengan menggeser titik desimal N1 sebanyak 20 kali ke kiri. Sehingga menjadi

Terlihat bahwa nilai LSB pada N1 menghilang ketika dilakukan penyetaraan eksponen.