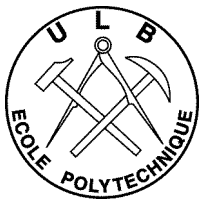

ELEC –H-305 Circuits Logiques et Numériques

Séance 1:

Systèmes de numération

1. Convertir dans les autres bases utiles les nombres suivants :

$(82)_{10}$	$(B65F)_{16}$
$(122)_{10}$	$(0.625)_{10}$
$(1001110001)_2$	$(10110001101011.111100000110)_2$
$(762)_8$	$(127.4)_8$
$(214)_8$	$(673.12)_8$
$(F6D)_{16}$	$(3A6.C)_{16}$

2. Effectuer l'addition suivante dans toutes les bases utiles. Vérifier les résultats en les convertissant en base 10 :

   $(3633)_{10} + (254)_{10}$

3. Représenter $(0.345)_{10}$ en base 2 et en base 8.

4. Effectuer les opérations suivantes :

$(10110)_2 - (10010)_2$	$(1011)_2 * (1001)_2$	quotient et reste :
$(10110)_2 - (10011)_2$		$(1100)_2 : (011)_2$
$(5475)_8 - (3764)_8$	$(762)_8 * (45)_8$	$(110101)_2 : (111)_2$
$(540045)_8 - (325654)_8$	$(543)_8 * (27)_8$	$(533)_8 : (26)_8$
$(E46)_{16} - (59F)_{16}$	$(1CF)_{16} * (B6)_{16}$	$(2ECD)_{16} : (12)_{16}$
$(4321)_{16} - (2ECD)_{16}$	$(2ECD)_{16} * (4321)_{16}$	

5. Représentation des nombres négatifs

   a)Représenter $(-14)_{10}$ sur 8 bits en base 2 dans les 3 modes de représentation.

   b)Si on utilise 4 bits, quels sont, dans les 3 modes de représentation, les plus petites et les plus grandes valeurs représentables ? Comment se représente la valeur 0 ?

   c)Effectuer les additions suivantes (sur 8 bits) dans les 3 modes de représentation :

52 + 84	84 - 52
52 - 84	-84 - 52

6. Représentation en virgule flottante (IEEE standard)

   a) Représenter (10100.10011)2 en virgule flottante.

   b) Convertir en binaire les nombres suivants représentés en virgule flottante :

   0 10000010 10000010 ... 000 1 01111000 01000110 ... 000

ELEC –H-305 Circuits Logiques et Numériques

1.

$(82)_{10}=(1010010)_2=(122)_8=(52)_{16}$
$(122)_{10}=(1111010)_2=(172)_8=(7A)_{16}$
$(1001110001)_2=(1161)_8=(625)_{10}=(271)_{16}$
$(762)_8=(111110010)_2=(1F2)_{16}=(498)_{10}$
$(214)_8=(10001100)_2=(8c)_{16}=(140)_{10}$
$(F6D)_{16}=(7555)_8=(3949)_{10}=(111101101101)_2$
$(B65F)_{16}=(133137)_8=(46687)_{10}=(1011011001011111)_2$
$(0.625)_{10}=(0.101)_2=(0.5)_8=(0.A)_{16}$
$(10110001101011.111100000110)_2=(26153.7406)_8=(2C6B.F06)_{16}=(11371.93896484375)_{10}$
$(127.4)_8=(1010111.1)_2=(57.8)_{16}=(87.5)_{10}$
$(673.12)_8=(110111011.001010)_2=(1BB.28)_{16}=(443.15625)_{10}$
$(3A6.C)_{16}=(001110100110.1100)_2=(1646.6)_8=(934.75)_{10}$

2.

$(3633)_{10}+(254)_{10}=(111100101111)_2=(7457)_8=(F2F)_{16}$

3.

$(0.010110\ldots..)_2$
$(0.260507\ldots..)_8$

4.

$(10110)_2$ - $(10010)_2$	$=(00100)_2$	$(1011)_2$ * $(1001)_2$	$=(1100011)_2$	$(1100)_2$: $(011)_2$	$=(100)_2$; $(100)_2$
$(10110)_2$ - $(10011)_2$	$=(00011)_2$	$(762)_8$ * $(45)_8$	$=(43772)_8$	$(110101)_2$:$(111)_2$	$=(111)_2$; $(100)_2$
$(5475)_8$ - $(3764)_8$	$=(1511)_8$	$(543)_8$ * $(27)_8$	$=(17745)_8$	$(533)_8$: $(26)_8$	$=(17)_8$; $(21)_8$
$(540045)_8$ - $(325654)_8$	$=(212171)_8$	$(1CF)_{16}$ * $(B6)_{16}$	$=(1492A)_{16}$	$(2ECD)_{16}$: $(12)_{16}$	$=(299)_{16}$; $(B)_{16}$
$(E46)_{16}$ - $(59F)_{16}$	$=(8A7)_{16}$	$(2ECD)_{16}$* $(4321)_{16}$	$=(C45AF6D)_{16}$		
$(4321)_{16}$ - $(2ECD)_{16}$	$=(1454)_{16}$				

5.

a) -14 -> SVA: 1 0001110 ->C1: 11110001 ->C2: 11110010
b) SVA: +7:-7 C1: +7:-7 C2: -8:7
c) SVA: 0000,1000 C1: 0000, 1111 C2:0000
Règles:

54+82
SVA - Addition normale. Overflow puisque 7 bits.
C1 - Les signes de deux nombres sont les mêmes. Résultat n'est pas de même signe, alors overflow.
C2 - Idem.
52-84
SVA - L'opération devient 84-52, on change le signe de résultat.
C1 - Additionner.
C2 - Additionner.
84-52
SVA - Il faut soustraire!
C1 - Additionner. Report généré, donc il faut corriger (ajouter un 1).
C2 - Additionner. Ignorer le report.
-84-52
SVA - Même signes. Additionner. Overflow puisque 7 bits.
C1 - Additionner. Signe de résultat n'est pas le même: underflow.
C2 - Signe de résultat n'est pas le même: underflow.

	SVA	C1	C2
52 (00110100)	0 0110100	00110100	00110100
84 (01010100)	0 1010100	01010100	01010100
	0 (1)0001000 (8+overflow=136)	10001000 (8+overflow=136)	10001000 (8+overflow=136)
52	0 1010100 (84)	00110100	00110100
-84	1 0110100 (52)	10101011	10101100
	1 0100000 (32) signe!	11011111 (-32)	11100000 (-32)
84	0 1010100	01010100	01010100
-52	1 0110100	11001011	11001100
	0 0100000 (32) soustraction!	(1) 00011111 (corr)	(1) 00100000 (32)
		00100000 (32)	
-84	1 1010100	10101011	10101100
-52	1 0110100	11001011	11001100
	1 (1)0001000(-8+overflow=-136)	(1) 01110110	(1) 01111000

6.

a)signe: 0
exposant: 10000011
partie fr.: 01001001100…0 (23 bits)
b)signe : +
exposant : 130 = 127+3
→ 1100.0001

c)signe : -
exposant : 120 = 127-7
→ -0.00000010100011