Exercice 1 : Soit les deux codes assembleur suivants (code A et code B) :

Code A

	B .S1	PE1
||	SUB .S2	
	ADD .L1	PE2
||	ADD .L2	
||	MPY .M1	
	MPY .M1	PE3
||	LDW .D1	
||	LDB .D2	

Code B

	B .S1	PE1
	SUB .S2	PE2
||	ADD .L1	
	B .S2	PE3
||	MPY .M1	
||	MPY .M1	
	LDW .D1	PE4
	LDB .D2	PE5

1. Déterminer le nombre de PE dans chacun de ces codes ? **Code A (3 PE) et Code B (5 PE)**

2. Remplissez les tableaux décrivant l'évolution du pipeline de chacun des codes ci-dessus (restreindre l'affichage au cycle 7) :

Code A :

Decode		Execute (5)					
DP	DC	E1	E2	E3	E4	E5	E6
B							
SUB							
ADD							
ADD							
MPY							
MPY							
LDW							
LDB							

Decode		Execute (6)					
DP	DC	E1	E2	E3	E4	E5	E6
	B						
	SUB						
ADD							
ADD							
MPY							
MPY							
LDW							
LDB							

Decode		Execute (7)					
DP	DC	E1	E2	E3	E4	E5	E6
		B					
		SUB					
	ADD						
	ADD						
	MPY						
MPY							
LDW							
LDB							

Dec	Execute (8)						OK
DC	E1	E2	E3	E4	E5	E6	
		B	+	+	+	+	
							SUB
	ADD						
	ADD						
	MPY	+					
MPY							
LDW							
LDB							

Dec	Execute (9)						OK
DC	E1	E2	E3	E4	E5	E6	
			B	+	+	+	
							SUB
							ADD
							ADD
		MPY					
	MPY	+					
	LDW	+	+	+	+		
	LDB	+	+	+	+		

Dec	Execute (10)						OK
DC	E1	E2	E3	E4	E5	E6	
				B	+	+	
							SUB
							ADD
							ADD
		MPY					
		LDW	+	+	+		
		LDB	+	+	+		

Dec	Execute (11)						OK
DC	E1	E2	E3	E4	E5	E6	
					B	+	
							SUB
							ADD
							ADD
							MPY
			LDW	+	+		
			LDB	+	+		

Dec	Execute (12)						OK
DC	E1	E2	E3	E4	E5	E6	
						B	
							SUB
							ADD
							ADD
							MPY
					LDW	+	
					LDB	+	

Dec	Execute (13)						OK
DC	E1	E2	E3	E4	E5	E6	
							B
							SUB
							ADD
							ADD
							MPY
							MPY
					LDW		
					LDB		

Dec	Execute (14)						OK
DC	E1	E2	E3	E4	E5	E6	
							B
							SUB
							ADD
							ADD
							MPY
							MPY
							LDW
							LDB

Code B :

Decode		Execute (5)					
DP	DC	E1	E2	E3	E4	E5	E6
B							
SUB							
ADD							
B							
MPY							
MPY							
LDW							
LDB							

Decode		Execute (6)					
DP	DC	E1	E2	E3	E4	E5	E6
	B						
SUB							
ADD							
B							
MPY							
MPY							
LDW							
LDB							

Decode		Execute (7)					
DP	DC	E1	E2	E3	E4	E5	E6
		B	+	+	+	+	+
	SUB						
	ADD						
B							
MPY							
MPY							
LDW							
LDB							

Decode		Execute (8)					
DP	DC	E1	E2	E3	E4	E5	E6
			B	+	+	+	+
		SUB					
		ADD					
	B						
	MPY						
	MPY						
LDW							
LDB							

Dec	Execute (9)						OK
DC	E1	E2	E3	E4	E5	E6	
			B	+	+	+	
							SUB
							ADD
	B	+	+	+	+	+	
	MPY	+					
	MPY	+					
LDW							
LDB							

Dec	Execute (10)						OK
DC	E1	E2	E3	E4	E5	E6	
				B	+	+	
							SUB
							ADD
		B	+	+	+	+	
		MPY					
		MPY					
	LDW	+	+	+	+		
LDB							

Dec	Execute (11)						OK
DC	E1	E2	E3	E4	E5	E6	
					B	+	
							SUB
							ADD
			B	+	+	+	
							MPY
							MPY
		LDW	+	+	+		
	LDB	+	+	+	+		

Dec	Execute (12)						OK
DC	E1	E2	E3	E4	E5	E6	
						B	
							SUB
							ADD
				B	+	+	
							MPY
							MPY
			LDW	+	+		
	LDB	+	+	+			

Dec	Execute (13)						OK
DC	E1	E2	E3	E4	E5	E6	
							B
							SUB
							ADD
					B		
							MPY
							MPY
				LDW	+		
			LDB	+	+		

Dec	Execute (14)						OK
DC	E1	E2	E3	E4	E5	E6	
							B
							SUB
							ADD
							B
							MPY
							MPY
					LDW		
				LDB	+		

Dec	Execute (15)						OK
DC	E1	E2	E3	E4	E5	E6	
							B
							SUB
							ADD
							B
							MPY
							MPY
							LDW
					LDB		

Dec	Execute (16)						OK
DC	E1	E2	E3	E4	E5	E6	
							B
							SUB
							ADD
							B
							MPY
							MPY
							LDW
							LDB

3. En déduire le nombre de cycles nécessité pour l'exécution de chaque code ?

Code A=14 cycles; Code B = 16 cycles.

Exercice 2 :

1. Donner les résultats après exécution de ces instructions :

MPY : Multiplication 16 bits x 16 bits signée = Résultat sur 32 bits (0x87C2 (valeur négative) x 0001)

MPY 0XFE1A87C2, 0X00020001, A1 ; A1= **0XFFFF87C2** (extension du bit de signe)

MPY2 : 2 Multiplications 16 bits x 16 bits signée = Résultat sur 64 bits (A3:A2)

MPY2	0X0001	0002,	0X0002	0002,	A3:	A2
	b1	b0	a1	a0	b1*a1	b0*a0
	0001	0002	0002	0002	0x00000002	0x00000004

A3 = 0x00000002 A2=0x00000004

MPY4 : 4 Multiplications 8 bits x 8 bits signée = Résultat sur 64 bits (B1:B0)

MPY4	0x24	25	26	27,	0x01	00	01	01	B1		B0	
	b3	b2	b1	b0	a3	a2	a1	a0	b3*a3	b2*a2	b1*a1	b0*a0
									0x0024	0x0000	0x0026	0x0027

B1 =0x00240000 B0=0x00260027

ADD2 : 2 additions 16 bits + 16 bits signée = Résultat sur 32 bits (registre A0)

ADD2	0XEFFE	EFFE,	0X2101	1012,	A0	
	b1	b0	a1	a0	b1+a1	b0+a0
					0x10FF	0x0010

A0=0x10FF0010

ADD4 : 4 additions 8 bits + 8 bits signée = Résultat sur 32 bits (B1:B0)

ADD4	0xEF	FE	EF	FE,	0x11	11	11	11,	B1			
	b3	b2	b1	b0	a3	a2	a1	a0	b3+a3	b2+a2	b1+a1	b0+a0
									0x00	0x0F	0x00	0x0F

A0=0x000F000F

MVKH : Chargement des 16 bits du **poids fort** d'une constante dans les 16 bits du **poids fort** du registre A1

MVKH	0x00010005	A1	
		0x0001	0000

A1=0x00010000

MVKLH : Chargement des 16 bits du **poids faible** d'une constante dans les 16 bits du **poids fort** du registre A1

MVKLH	0x00010005	A1	
		0x0005	0000

A1=0x00050000

MVK : Chargement des 16 bits du **poids faible** d'une constante dans les 16 bits du **poids faible** du registre A1

MVK	0x00010005	A1	
		0x0000	0005

A1=0x00000005

2. **Soit A4=2.25 et B4=8.5**

MPYSP A4, B4, A6 ; A4=……..……; B4=……; A6=……

Float (IEEE754 Single precision 32-bit) : $(-1)^s * (1.m) * 2^{e-127}$

2.25 = 10.01 = 1.001 * 2^1 = **$(-1)^0$ * (1.001) * $2^{1=e-127}$**

e-127 = 1 ➔ e = 128 ➔ **exposant = 10000000**

1.m = 1.001 ➔ **m = 001**

A= $1.001*2^{128-127}$

S	Exposant								Mantisse																			
0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4				0					1			0			0			0			0			0				

A = 0x 40100000

8.5 = 1000.1 = 1.0001 * 2^3

e-127 = 3 ➔ e = 130 ➔ **exposant = 10000010**

1.m = 1.0001 ➔ **m = 0001**

B= $1.001*2^{130-127}$

S	Exposant								Mantisse																			
0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4				1					0			8			0			0			0			0				

B = 0x 41080000

A*B = 1.001 * 2^1 * 1.0001 * 2^3 = 1.0011001 * 2^4

e-127 = 4 ➔ e = 131 ➔ **exposant = 10000011**

1.m = 1.0011001 ➔ **m = 0011001**

A*B = 1.0011001 * 2^4 = 10011.001 = **19.125**

2.25 * 8.5 = 19.125

```
          1 0 0 0 1
            1 0 0 1
          ---------
          1 0 0 0 1
        0 0 0 0 0
      0 0 0 0 0
    1 0 0 0 1
  ---------------
  1 0 0 1 1 0 0 1
```

S	Exposant								Mantisse																			
0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4				1					9			9			0			0			0			0				

A * B = 0x 41990000

3. **Soit le code suivant:**

```
        MVK   .S1   0x1000, A2     ; Chargement constante 16bits LSB➔ A2=0x0000 1000
        MVK   .S1   0x28, A1       ; Chargement constante 16bits LSB➔ A1=0x0000 0028
[!A2]   SUB   .D1   A1, 1, A1 ; Exécution si A2=0 ➔ or A2 !=0 ➔ Instruction non exécutée➔A1=0x0000 0028
||[A2]  ADD   .S1   A5, A3, A3 ; Exécution si A2!=0 ➔ A3=0x0000 0035
```

Sachant que A5= 0x10 et A3 = 0x25, quelle est la valeur de A1 et A3 après l'exécution de ce code?

A5	A3	A2	A1
0x00000010	0x00000035	0x00001000	0x00000028

Exercice 3 :

1. Sachant que et que dans la mémoire à partir de l'adresse 0x00001234 on a les données suivantes :

   Quel est la valeur de A0 et A1 après l'exécution de ces instructions : (**NB**: Les 4 cas sont indépendants)

Charger dans le registre A1 l'élément sur 16bits=2 octets (LDH) pointé par le registre A0 **puis saut** du pointeur de deux éléments (2 octets x 2 = 4 octets)

a) LDH *A0++ [2], A1;

A0= 0x00001238 et A1= 0x00000676

Adresse	Valeur	Signification
A0= 0x00001234➔	0x06	1er élément
	0x76	
	0x54	2ème élément
	0x12	
A0= 0x00001238➔	0x11	3ème élément
	0xAF	
	0x72	4ème élément
	0x52	

Se déplacer à l'élément suivant de 32 bits=4octets (LDW) **puis charger** dans le registre A1 l'élément pointé par le registre A0.

b) LDW *++A0, A1;

A0= 0x00001238 et A1= 0x11AF7252

Adresse	Valeur	Signification
A0= 0x00001234➔	0x06	1er élément
	0x76	
	0x54	
	0x12	
A0= 0x00001238➔	0x11	2ème élément
	0xAF	
	0x72	
	0x52	

Charger dans le registre A1 l'élément sur 32bits (4 octets) pointé par le registre A0 **puis pointer** sur l'élément suivant (+ 4 octets)

c) LDW *A0++, A1;

A0= 0x00001238 et A1= 0x06765412

Adresse	Valeur	Signification
A0= 0x00001234➔	0x06	1er élément
	0x76	
	0x54	
	0x12	
A0= 0x00001238➔	0x11	2ème élément
	0xAF	
	0x72	
	0x52	

Se déplacer à l'élément suivant de 32 bits=4octets (LDW) et charger dans le registre A1 l'élément pointé par le registre A0 puis revenir à l'adresse de départ

d) LDW *+A0, A1;

A0= 0x00001234 et A1= 0x11AF7252

Adresse	Valeur	Signification
A0= 0x00001234➔	0x06	1er élément
	0x76	
	0x54	
	0x12	
	0x11	2ème élément
	0xAF	
	0x72	
	0x52	

Charger dans le registre A1 l'élément sur 8 bits (1 octet =1 Byte) pointé par le registre A0 **puis saut** du pointeur de 4 éléments (+ 4 octets)

e) LDB *A0++[4], A1;

A0= 0x00001238 et A1= 0x00000006

Adresse	Valeur	Signification
A0= 0x00001234➔	0x06	1er élément
	0x76	
	0x54	
	0x12	
	0x11	2ème élément
	0xAF	
	0x72	
A0= 0x0000123B➔	0x52	

Charger dans le registre A1 l'élément sur 64bits (8 octets) pointé par le registre A0 **puis pointer** sur l'élément suivant (+ 8 octets)

f) LDDW *A0++, A3 : A2;

A0= 0x0000123B;
A3= 0x06765412; A2= 0x11AF7252

Adresse	Valeur	Signification
A0= 0x00001234➔	0x06	1er élément
	0x76	
	0x54	
	0x12	
	0x11	2ème élément
	0xAF	
	0x72	
	0x52	
A0= 0x0000123C➔	0xFA	

Exercice 4 :

1. Soit le code assembleur suivant, insérer le minimum d'instructions NOPs pour avoir un résultat correct :

Source	1-Minimum de NOPs	2-Rem NOP par Instruction	3-Instructions Parallèles
start: LDH .D1 *A0, A1 MPY .M1 A1, A2, A3 ADD .L1 A1, A2, A4 LDB .D1 *A5, A6 LDW .D1 *A7, A8 MPY .M1 A6, A9, A10 ADD .L1 A9, A10, A11 B .S1 **start**	**start:** LDH .D1 *A0, **A1** **NOP 4** MPY .M1 **A1**, A2, A3 ADD .L1 A1, A2, A4 LDB .D1 *A5, **A6** **NOP 3** LDW .D1 *A7, A8 MPY .M1 **A6**, A9, **A10** **NOP** ADD .L1 A9, **A10**, A11 B .S1 **start** **NOP 5**	**Start:** LDH .D1 *A0, **A1** LDB .D1 *A5, **A6** LDW .D1 *A7, A8 B .S1 **start** **NOP** MPY .M1 **A1**, A2, A3 MPY .M1 **A6**, A9, **A10** ADD .L1 **A1**, A2, A4 ADD .L1 A9, **A10**, A11	**Start:** LDH .D1 *A0, **A1** || LDB .D2 *B5, **B6** LDW .D1 *A7, A8 B .S1 **start** **NOP 2** MPY .M1 **A1**, A2, A3 || MPY .M2 **B6**, B9, **B10** || ADD .L1 **A1**, A2, A4 **NOP** ADD .L1**x** A9, **B10**, A11
Nombre de cycles	**21**	**9**	**8**