

Correction

TD N°1 DSP

Section : 2^{eme}Année GII

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

	C	ode A	
	В	.S1	DE1
	SUB	.S2	PE1
	ADD	.L1	
	ADD	.L2	PE2
	MPY	.M1	
	MPY	.M1	
II	LDW	.D1	PE3
	LDB	.D2	

	Co	ode B	
	В	.S1	PE1
	SUB	PE2	
II	ADD	.L1	PEZ
	В	.S2	
- II	MPY	.M1	PE3
- II	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:

D	ecode			Execu	ıte (5)			D	ecod	е		Execu	ıte (6	E4 E5 E6 DP DC E1 E B SUB ADD ADD MPY MPY LDW LDB								ıte (7)		C)ec			Execu	te (8)			ОК
D	P DO	· F1	Lra	L ra l	E4	E5	E6	D	P D	C E1	E2	E3	E4	E5	E6		DP	DC	E1	E2	E3	E4	E5	E6	I	DC	E1	E2	E3	E4	E5	E6	
В	PDC	, ET	EZ	ES	E4	E3	E0		В										В									В	+	+	+	+	
SU	D	+	\vdash						SU	В									SUB														SUB
AD	_	+	1					AD	D									ADD									ADD						
AD		+	1					AD	D									ADD									ADD						
MF		+	+					MF	_									MPY									MPY	+					
MF		+	1					MF	γ							h	MPY								M	ΙPΥ							
LD'		1	1					LD																		w							
LD	_		\vdash					LD	В							L	.DB								LE)B							
Dec	_	-	Execu	ite (9))	!	ОК	Dec		E	xecu	te (10)		ОК	OK Dec Execute (11) OK							Ħ	Dec			Execu	te (12)			ОК		
DC	E1	E2	E3	E4	E5	E6		DC	E1	E2	E3	E4		F.C		Ļ		-4	F2	гэ	E4	E5	E6	-	4 6	DC	E1	E2	E3	E4	E5	E6	1
ьс	ET	EZ	В	+	+	+		DC	ET	EZ	ES	В В				۳	٠, ١	:1	EZ	ES	E4	В	+		1 F	DC	ET	EZ	E3	E4	ED	В	1
 -	<u> </u>		В	+	+		SUB					В	+	-	CLIB	┢	+		-			В	Т.	SUB	1 F							В	SUB
_							ADD									╟	+							ADD	1 F								ADD
_							ADD									╟	+							ADD	1 F								ADD
_		MPY					AUU									╟	+							MPY	1 F								MPY
-	MPY	+			1					MPY					IVIF	╟	+	+	-+				1	MPY									MPY
-	LDW	+	+	+	+					LDW	+	+	+	<u> </u>		一	+		_	DW	+	+			1					LDW	+		1
	LDB	+	+	+	+					LDB	+	+					T		_		+	+			11					LDB	+		
Dec	_		xecu	te (13		·	ОК	Dec				te (14		-	ОК	-		_							+								1
DC	E1	E2	E3	E4	E5	E6		DC	E1	E2	E3	E4	ES	F6																			
- DC		LZ	LJ	L-4		LU	В	ьс		LZ	LJ		LJ	LU	R																		
							SUB								SUB																		
						_	ADD								ADD										1								
							ADD								ADD										1								
					1		MPY						1	1	MPY	1									1								
					1		MPY						1	1	MPY	1									1								
					LDW		1						1	1	LDW	1									1								
					LDB										LDB	ĺ									1								

Code B:

, 00	ie E) :											_											-								
D	ecode	:		Exec	ute (5)			l D	ecode	:		Exec	ıte (6)				ecod	е		Exec	ute (7))			ecode	·		Execu	ute (8)		
DI	P DO	^ F1	. E2	E3	E4	E5	5 F	6	DI	D D	E1	E2	E3	E4	E5	E6	D	P D	C E1	L E2	E3	E4	E5	E6	D	P DO	C E1		E3	E4	E5	E6
В	+-	+	+	1-5		+=	- -	-		В									В	+	+	+	+	+	l 📙			В	+	+	+	+
SUI	3					1	1		SUE	_	_						<u> </u> _	SU	_						l	_	SU	_	₩			
AD									AD	D		-					l <u>L</u>	AD	D						l 📙		AD	D				
В									B MP	.,	+	-					B M	D)/		-	1				l	B MP		+	\vdash	-+		_
MP	Υ								MP		+-	1	\vdash				M		_					_	l	MP	_	+	├	\rightarrow		
MP	Υ								LD\		+	-					_	W		-					LD'	_	Y	+	H	-+		
LD۱									LDE		+	1		-			LD	_	+					_	LD	_	-	+-	\vdash	-+		
LDE	3								LDL								122								100	_			Щ			
De	С		Exe	cute (9)			ОК	Dec		-	Execu	te (10))		ОК	Dec	:		Execu	te (11	L)		ОК	Dec			Execu	te (12	<u>'</u>)		ОК
DC	E1	E2	E3	E4	ı I	E5 I	E6		DC	E1	E2	E3	E4	E5	E6		DC	E1	E2	E3	E4	E5	E6		DC	E1	E2	E3	E4	E5	E6	
			В	+	*	+	+						В	+	+							В	+								В	
							_	SUB								SUB								SUB				<u> </u>	<u> </u>			SUB
				_	4	_	_	ADD								ADD								ADD				<u> </u>	Щ.	<u> </u>		ADD
	В	+	+	+		-	+				В	+	+	+	+					В	+	+	+		l			├	В	+	+	
-	MP	_	-	-		_					MPY								<u> </u>					MPY	 			├	├	—	<u> </u>	MPY
LDV	MP۱	+	-	-	+	+				LDW	MPY +	+	+	+				1	LDW	1 +	+	+		MPY				LDW	+	+		MPY
LDV	V		+	-	+	+			LDB	LDW	+	+	+	+				LDB	+	+	+	+			 		LDB		+	+	-	
Dec	1		Execu	ıte (1	3)			ОК	Dec	<u> </u>		Xecu	te (14)	1	ОК	Dec	_		Execu			1	ОК	Dec			Execu		4	<u> </u>	ОК
									DC								DC						T ==		DC							
DC	E1	E2	E3	E4	+	5 E	E6	В	DC	E1	E2	E3	E4	E5	E6	В	DC	E1	E2	E3	E4	E5	E6	В	DC	E1	E2	E3	E4	E5	E6	В
				-	╁	+	-	SUB						-	-	SUB		1		1	-	1		SUB	-			 	 	+		SUB
					+	+		ADD								ADD					1			ADD	l 			 	 	+		ADD
					В	+	_	100								В		1		1	1	1		В	l 			 	 	+		В
				1	Ť	t	1	ИРҮ								MPY		1		1	1	1	1	MPY				 	 	†		MPY
					T	T	_	MPY						t	t	MPY			l –					MPY						†	t	MPY
				LDW	/ +									LDW							1		1	LDW						1		LDW
			LDB	+	+								LDB	+								LDB										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	В	81	В	80
	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	b1 b0		0X2101	1012,	A	0
	bl	b0	al	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 13								
	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=0x0000005

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) * 2e-127

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

$$e-127 = 1 \rightarrow e = 128 \rightarrow exposant = 10000000$$

 $1.m = 1.001 \rightarrow m = 001$

A= 1.001*2¹²⁸⁻¹²⁷

S				Ex	кро	sar	nt													Ma	nti	sse										
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	0	0	0
		4				()			1	L			()			C)			C)			C)			C)	

A = 0x 40100000

8.5 =
$$1000.1 = 1.0001 * 2^3$$

$$e-127 = 3 \rightarrow e = 130 \rightarrow exposant = 10000010$$

$$1.m = 1.0001 \rightarrow m = 0001$$

B= 1.001*2¹³⁰⁻¹²⁷

	S	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	0	0	0
ĺ		4	1			1	L			()			8	3			C)			C)			()			C)	

B = 0x 41080000

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

$$e-127 = 4 \Rightarrow e = 131 \Rightarrow exposant = 10000011$$

$$1.m = 1.0011001 \Rightarrow m = 0011001$$

$$A*B = 1.0011001 * 2^{4} = 10011.001 = 19.125$$

$$2.25 * 8.5 = 19.125$$

$$1 0 0 0 1$$

$$0 0 0 0 0$$

$$1 0 0 0 1$$

	S	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	0	0	0
I			4			1	L			ç)			ç)			C)			C)			C)			C)	

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		
0x0000010	0x00000035	0x00001000	0x00000028		

Exercice 3:

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

Ouel est la valeur de A0 et A1 a	près l'exécution de ces instructions :	(NR: Les 4 cas sont indépendants)
Quel est la valeur de Ao et Ar a	pres reaccunon de ces misu denons.	(11 b . Les 4 cas sont independants)

	Adresse	Valeur	Signification
Charger dans le registre A1 l'élément sur 16bits=2	$A0 = 0 \times 00001234$	0x06	1 ^{er} élément
octets (LDH) pointé par le registre A0 puis saut du		0x76	1" element
pointeur de deux éléments (2 octets x 2 = 4 octets)		0x54	2 ^{ème} élément
pointeur de deux elements (2 detets x 2 - 1 detets)		0x12	2 element
a) I DII *A0 [2] A1.	$A0 = 0 \times 00001238$	0x11	3 ^{ème} élément
a) LDH *A0++ [2], A1;		0xAF	3 element
A0= 0x00001238 et A1= 0x00000676		0x72	4 ^{ème} élément
		0x52	4 element

	Adresse	Valeur	Signification
Se déplacer à l'élément suivant de 32 bits=4octets	$A0 = 0 \times 00001234$	0x06	
(LDW) puis charger dans le registre A1 l'élément		0x76	1 ^{er} élément
pointé par le registre A0.		0x54	
pointe par le registre no.		0x12	
L) I DW \$1140 A1.	$A0 = 0 \times 00001238$	0x11	
b) LDW *++A0, A1;		0xAF	2 ^{ème} élément
A0= 0x00001238 et A1= 0x11AF7252		0x72	2 element
		0x52	

	Adresse	Valeur	Signification
Charger dans le registre A1 l'élément sur 32bits (4	A0 = 0x00001234	0x06	
octets) pointé par le registre A0 puis pointer sur		0x76	1 ^{er} élément
l'élément suivant (+ 4 octets)		0x54	
referrence survaire (* 1 octob)		0x12	
a) I DW/ *AO++ A1.	$A0 = 0 \times 00001238$	0x11	
c) LDW *A0++, A1;		0xAF	2 ^{ème} élément
A0= 0x00001238 et A1= 0x06765412		0x72	2 element
		0x52	

	Adresse	Valeur	Signification	
Se déplacer à l'élément suivant de 32 bits=4octets	$A0 = 0 \times 00001234$	0x06		
(LDW) et charger dans le registre A1 l'élément		0x76	1 ^{er} élément	
pointé par le registre A0 puis revenir à l'adresse de		0x54		
départ		0x12		
depart		0x11		
I) I DYV to A O A 4		0xAF	2 ^{ème} élément	
d) LDW *+A0, A1;		0x72	2 element	
A0= 0x00001234 et A1= 0x11AF7252		0x52		

	Adresse	Valeur	Signification
Charger dans le registre A1 l'élément sur 8 bits	$A0 = 0 \times 00001234$	0x06	
(1 octet =1 Byte) pointé par le registre A0 puis		0x76	1 ^{er} élément
saut du pointeur de 4 éléments (+ 4 octets)		0x54	
saut au pointeur de l'elements (* l'octets)		0x12	
e) LDB *A0++[4], A1;		0x11	
		0xAF	2 ^{ème} élément
A0= 0x00001238 et A1= 0x00000006		0x72	Z eiement
	$A0 = 0 \times 0000123B$	0x52	

	Adresse	Valeur	Signification
Charger dans le registre A1 l'élément sur 64bits (8	$A0 = 0 \times 00001234$	0x06	
octets) pointé par le registre A0 puis pointer sur		0x76	1 ^{er} élément
l'élément suivant (+ 8 octets)		0x54	
retement survaint (* o octob)		0x12	
6 IDDW * 40 + 42 . 42.		0x11	
f) LDDW *A0++, A3 : A2;		0xAF	2 ^{ème} élément
A0= 0x0000123B;		0x72	2 Cicilicit
A3= 0x06765412; A2= 0x11AF7252		0x52	
·	$A0 = 0 \times 0000123C \rightarrow$	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, <mark>A1</mark>	start: LDH .D1 *A0, A1	Start: LDH .D1 *A0, A1	Start: LDH .D1 *A0, A1
	MPY .M1 <mark>A1</mark> , A2, A3	NOP 4	LDB .D1 *A5, A6	LDB .D2 *B5, <mark>B6</mark>
	ADD .L1 A1, A2, A4	MPY .M1 A1 , A2, A3	LDW .D1 *A7, A8	LDW .D1 *A7, A8
	LDB .D1 *A5, <mark>A6</mark>	ADD .L1 A1, A2, A4	B .S1 start	B .S1 start
	LDW .D1 *A7, A8	LDB .D1 *A5, A6	NOP	NOP 2
	MPY .M1 <mark>A6</mark> , A9, <mark>A10</mark>	NOP 3	MPY .M1 A1, A2, A3	MPY .M1 A1, A2, A3
	ADD .L1 A9, <mark>A10</mark> , A11	LDW .D1 *A7, A8	MPY .M1 A6 , A9, A10	MPY .M2 <mark>B6</mark> , B9, <mark>B10</mark>
	B .S1 start	MPY .M1 A6 , A9, A10	ADD .L1 A1 , A2, A4	ADD .L1 A1 , A2, A4
		NOP	ADD .L1 A9, A10 , A11	NOP
		ADD .L1 A9, A10 , A11		ADD .L1 <mark>x</mark> A9, <mark>B10</mark> , A11
		B.S1 start		
		NOP 5		
N	ombre de cycles	21	9	8