T.D. 8 – Corrigé Codage machine des instructions

Exercice 1

Pour chacune des instructions suivantes, vous indiquerez :

- le mode d'adressage de chaque opérande ;
- le code machine 68000 de l'instruction.
- 1. MOVE.L D0, D6

Mode d'adressage

Source	Dn	Direct par registre de donnée
Destination	Dn	Direct par registre de donnée

Mot de 16 bits de l'instruction MOVE

	0	СТ	717		D:	ESTI	VATIC	N			SOU	RCE				
		21	ZE	RE	GIST	ER		MODE			MODE			REGISTER		
0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	
МО	VE	•	L			D	6			D0						

Code machine complet en représentation hexadécimale : 2C00

2. MOVE.B (A1),\$123456

Mode d'adressage

Source	(An)	Indirect par registre d'adresse
Destination	(xxx).L	Absolu long

Mot de 16 bits de l'instruction MOVE

	0	QT	ZE		D.	ESTII	OITA	N		SOURCE						
	0	21	تا ت	RE	GIST	ER		MODE			MODE			REGISTER		
0	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1	
МО	VE		В			(xx)	٤).L			(A1)						

Information à ajouter pour la destination : (xxx).L = \$00123456 Un adressage absolu long représente une adresse sur 32 bits non signés.

Code machine complet en représentation hexadécimale : 13D1 0012 3456

T.D. 8 – Corrigé

3. ADD.W \$2A(A5),D7

Mode d'adressage

Source	d16(An)	Indirect par registre d'adresse avec déplacement
Destination	Dn	Direct par registre de donnée

Mot de 16 bits de l'instruction ADD

1	1	0	1	DE	'CTCT	r D		DMOD	C C	EFFECTIVE ADDRESS						
_	_			NE.	REGISTER			OPMODE			MODE		REGISTER			
1	1	0	1	1	1 1 1		0	0	1	1 0 1		1	1 0 1			
	AI	DD			D7			.W <ea>,Dn</ea>			d16 (A5)					

Information à ajouter pour la source : **d16 = \$002A** d16 représente un déplacement sur 16 bits signés.

Code machine complet en représentation hexadécimale : DE6D 002A

4. ADDI.W #515, -\$10(A5, D5.L)

Mode d'adressage

Source	# <data></data>	Immédiat
Destination	d8(An, Xn)	Indirect par registre d'adresse avec déplacement et index

Mot de 16 bits de l'instruction ADDI

	0	0	0	0	1	1	0	SIZE		EFFECTIVE ADDRESS						
			U	U			U	21	SIZE		MODE			REGISTER		
0	0	0	0	0	1	1	0	0	1	1	1	0	1	0	1	
	ADDI # <data></data>							.W d8 (A5, Xn				5,Xn)				

Information à ajouter pour la source : #<data> = #515 = #\$0203

La taille de la donnée du mode d'adressage immédiat correspond à la taille de l'instruction. L'instruction possède ici l'extension . W. La taille de la donnée est donc de 16 bits.

Il y a deux informations à ajouter pour la destination : la valeur de **d8** et la valeur de **xn**. Ces deux informations doivent être contenues dans ce qui s'appelle le **mot d'extension**. La structure du mot d'extension est présente dans le manuel de référence du 68000.

T.D. 8 – Corrigé 2/7

David Bouchet – Architecture des ordinateurs – Info-Spé 2011/2012

Mot d'extension du 68000

D/A	RE	GIST	ER	W/L	0	0	0		DISPLACEMENT INTEGER					ER	
0	1	0	1	1	0	0	0	1 1 1 1 0 0 0 0					0		
	D	5		.L					d8 = -\$10 (\$F0 en complément à						

Les 5 bits de poids fort du mot d'extension servent à identifier le registre Xn et les 8 bits de poids faible contiennent la valeur de d8. d8 est un déplacement codé sur 8 bits signés. Lorsque sa valeur est négative, il faut déterminer son complément à 2 afin d'obtenir les bits à positionner dans le champ du mot d'extension.

La représentation hexadécimale du mot d'extension est : 58F0

Lorsque deux informations sont à ajouter (une pour la source et une pour la destination), il faut commencer par ajouter celle de la source puis enfin celle de la destination.

Code machine complet en représentation hexadécimale : 0675 0203 58F0

5. CMPA.W (A7) + A0

Mode d'adressage

Source	(An)+	Indirect par registre d'adresse avec postincrémentation
Destination	An	Direct par registre d'adresse

Mot de 16 bits de l'instruction CMPA

1	0	1	1								CTIVE	E ADDRESS				
			Δ.	KE	REGISTER			OFMODE			MODE		REGISTER			
1	0	1	1	0	0 0 0		0	1	1	0	0 1 1		1 1 1		1	
	СМ	PA.			A 0			. W			(A7)+					

Code machine complet en représentation hexadécimale : **BODF**

6. SUBI.L #\$112233,\$456

Mode d'adressage

Source	# <data></data>	Immédiat
Destination	(xxx).L	Absolu long

T.D. 8 – Corrigé 3/7

Mot de 16 bits de l'instruction SUBI

	0	0	0	0	1	0	0	СТ	SIZE		EFFECTIVE ADDRESS							
	U	U	U	U		U	U	21	ΔĿ		MODE		RE	GIST	ER			
0	0	0	0	0	1	0	0	1	0	1	1	1	0	0	1			
	SUBI # <data></data>								L			(xxx)	k).L					

Information à ajouter pour la source : #<data> = #\$00112233

La taille de la donnée du mode d'adressage immédiat correspond à la taille de l'instruction. L'instruction possède ici l'extension . L. La taille de la donnée est donc de 32 bits.

Information à ajouter pour la destination : (xxx).L = \$00000456 Un adressage absolu long représente une adresse sur 32 bits non signés.

Code machine complet en représentation hexadécimale :

04B9 0011 2233 0000 0456

Exercice 2

Soit le programme ci-dessous :

```
PRINT
        EQU
                $7CA480
                $2000
        ORG
DEBUT
                MESSAGE, A0
        LEA
BOUCLE MOVE.W $FFFF85FC.L,D1
        MOVE.W #$65F3,D0
        JSR
                PRINT
        RTS
MESSAGE DC.B
                "Archi",13,10,0
        DS.W
FIN
```

1. Différenciez les directives d'assemblage des instructions.

Ce programme possède les quatre directives d'assemblage suivantes : EQU, ORG, DC.B et DS.W. Une directive n'est pas une instruction : elle n'appartient pas au jeu d'instructions du microprocesseur et ne sera pas convertie en langage machine. Elle sert simplement, comme son nom l'indique, à donner une directive à l'assembleur.

2. Assemblez-le et donnez la valeur hexadécimale de toutes les étiquettes.

Afin de déterminer la valeur de chacune des étiquettes, nous allons examiner en détail ce programme.

T.D. 8 – Corrigé 4/7

Commençons par la première ligne :

```
PRINT EQU $7CA480
```

Il y apparaît la directive EQU qui permet d'assigner une expression à une étiquette. L'étiquette PRINT prend donc la valeur \$7CA480.

```
ORG $2000
```

La directive ORG permet de sélectionner l'adresse à partir de laquelle seront assemblées les instructions qui suivent.

	ORG	\$2000		
DEBUT	LEA	MESSAGE, A0		

L'instruction LEA sera donc assemblée à l'adresse \$2000.

De plus, une étiquette placée en début de ligne prend la valeur de l'adresse de l'instruction qui suit. L'étiquette DEBUT prendra donc la valeur \$2000.

	ORG	\$2000
DEBUT	LEA	MESSAGE, AO
BOUCLE	MOVE.W	\$FFFF85FC.L,D1

Afin de déterminer la valeur de l'étiquette BOUCLE, il faut trouver à quelle adresse est située l'instruction MOVE. Or, cette instruction sera placée en mémoire à la suite de l'instruction LEA. Il faut donc déterminer sur combien d'octets est codée l'instruction LEA. Pour cela, il faut la coder en langage machine.

Mot de 16 bits de l'instruction LEA

	1	0	0	חם	'C T C M	STER		1 1		EFFECTIVE ADDRESS					
		U		KL	GISI	LK	Δ.	Δ.	1		MODE		RE	GIST	ER
0	1	0	0	0	0	0	1	1	1	1	1	1	0	0	1
	Ll	ΞA			A0		LEA			(xxx).L = MESSAGE					

L'adresse effective source est ici l'étiquette MESSAGE. Cette étiquette représente une adresse située légèrement plus loin dans le code source. Il s'agit donc d'un adressage absolu long. Le problème est qu'à ce stade de l'assemblage, la valeur de MESSAGE n'est pas connue. Par contre, sa taille est connue : un adressage absolu long représente une adresse codée sur 32 bits. Il faut donc réserver un espace dans la mémoire d'une taille de 32 bits.

Code machine de l'instruction : 41F9 ???? ????

T.D. 8 – Corrigé 5/7

David Bouchet – Architecture des ordinateurs – Info-Spé 2011/2012

Afin de faciliter la notation, le contenu de la mémoire sera maintenant représenté à gauche du code source :

```
ORG $2000

002000 41F9 ??????? DEBUT LEA MESSAGE,A0

BOUCLE MOVE.W $FFFF85FC.L,D1
```

Avec cette nouvelle représentation, on constate que l'instruction LEA est située à l'adresse \$2000 et que son code machine débute par \$41F9 pour se terminer par une valeur inconnue sur 32 bits.

On peut maintenant en déduire l'adresse de l'instruction MOVE qui suit puisque l'on connaît la taille de l'instruction LEA:

			ORG	\$2000
002000	41F9 ????????	DEBUT	LEA	MESSAGE, A0
002006		BOUCLE	MOVE.W	\$FFFF85FC.L,D1

Assemblons maintenant, de la même manière, le reste du programme afin de déterminer la valeur de toutes les étiquettes.

				.
7CA480		PRINT	EQU	\$7CA480
002000			ORG	\$2000
002000	41F9 00002018	DEBUT	LEA	MESSAGE, AO
002006	3239 FFFF85FC	BOUCLE	MOVE.W	\$FFFF85FC.L,D1
00200C	303C 65F3		MOVE.W	#\$65F3,D0
002010	4EB9 007CA480		JSR	PRINT
002016	4E75		RTS	
002018	41726368690D0A00	MESSAGE	DC.B	"Archi",13,10,0
002020	0000 0000 0000		DS.W	3
002026		FIN		

Une fois que la valeur de l'étiquette MESSAGE est connue (\$2018), on peut terminer le codage machine de l'instruction LEA située à l'adresse \$2000.

La directive DC.B sert à écrire une suite d'octets en mémoire. Les octets peuvent s'exprimer sous la forme d'une chaîne de caractères s'ils sont contenus entre deux doubles *quotes*.

La directive DS.W 3 permet de réserver un espace mémoire de 3 mots de 16 bits. Cet espace mémoire est généralement rempli par des zéros.

T.D. 8 – Corrigé

Exercice 3

Sachant que seules les instructions JMP, RTS et ADDI sont utilisées, donnez le programme assembleur correspondant au contenu de la mémoire ci-dessous :

007000 06 40 4E 75 007004 4E 75 4E F9 007008 00 00 40 00

Il suffit de trouver à quelle instruction appartient le premier mot de 16 bits (\$0640). On s'aperçoit rapidement qu'il s'agit de l'instruction ADDI. Il faut ensuite déterminer les modes d'adressage utilisés afin d'identifier les informations de l'opérande source et de l'opérande destination.

Mot de 16 bits de l'instruction ADDI #<data>,<ea>

	0	0	0	0	1	1	0	SIZE -		EFFECTIVE ADDRESS							
	U	U	U	U			U	51	스타		MODE		RE	GIST	ER		
0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0		
	ADDI # <data></data>							W			D	0					

La taille de la donnée immédiate est sur 16 bits. On obtient donc :

007000 0640 4E75 ADDI

Le mot de 16 bits suivant est \$4E75. Il s'agit d'un RTS qui ne nécessite aucun opérande.

007000	0640 4E75	ADDI.W #\$4E75,D0
007004	4E75	RTS

Le mot de 16 bits suivant est \$4EF9. On l'identifie facilement à l'instruction JMP.

Mot de 16 bits de l'instruction JMP <ea>

	1	0	0	1	1	1	0	1	1		EFFE	CTIVE	E ADD	RESS	
		0					0				MODE		RE	GIST	ER
0	1	0	0	1	1	1	0	1	1	1	1	1	0	0	1
	JMP											(xxx)	k).L		

Le mode d'adressage est le mode absolu long. Ce mode correspond à une adresse codée sur 32 bits.

007000	0640 4E75	ADDI.W	#\$4E75,D0
007004	4E75	RTS	
007006	4EF9 00004000	JMP	\$4000

T.D. 8 – Corrigé 7/7