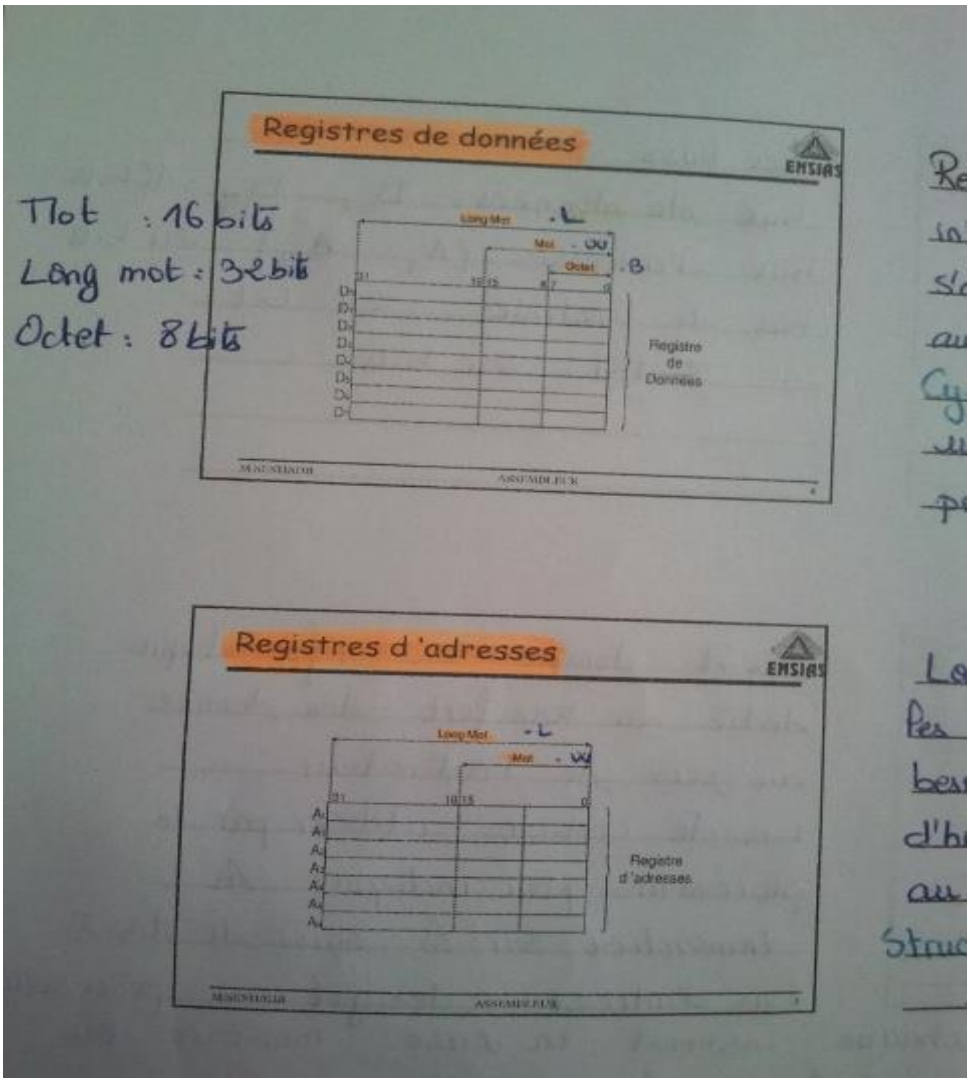
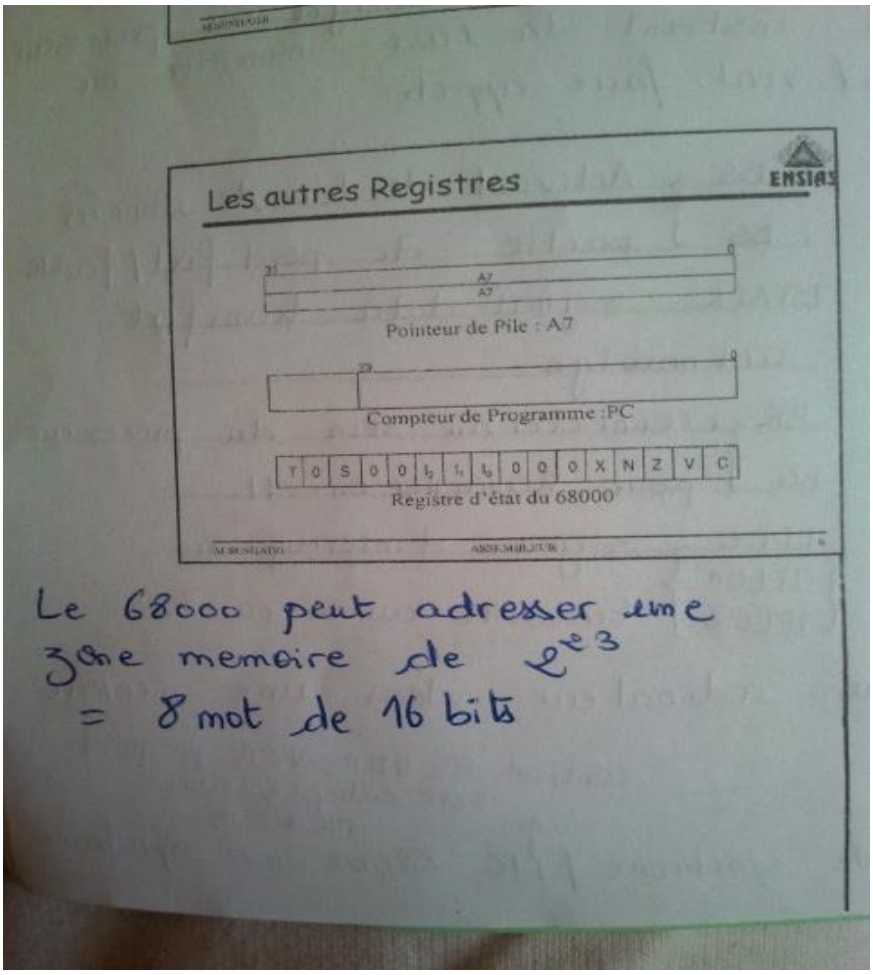
# Assembleur :

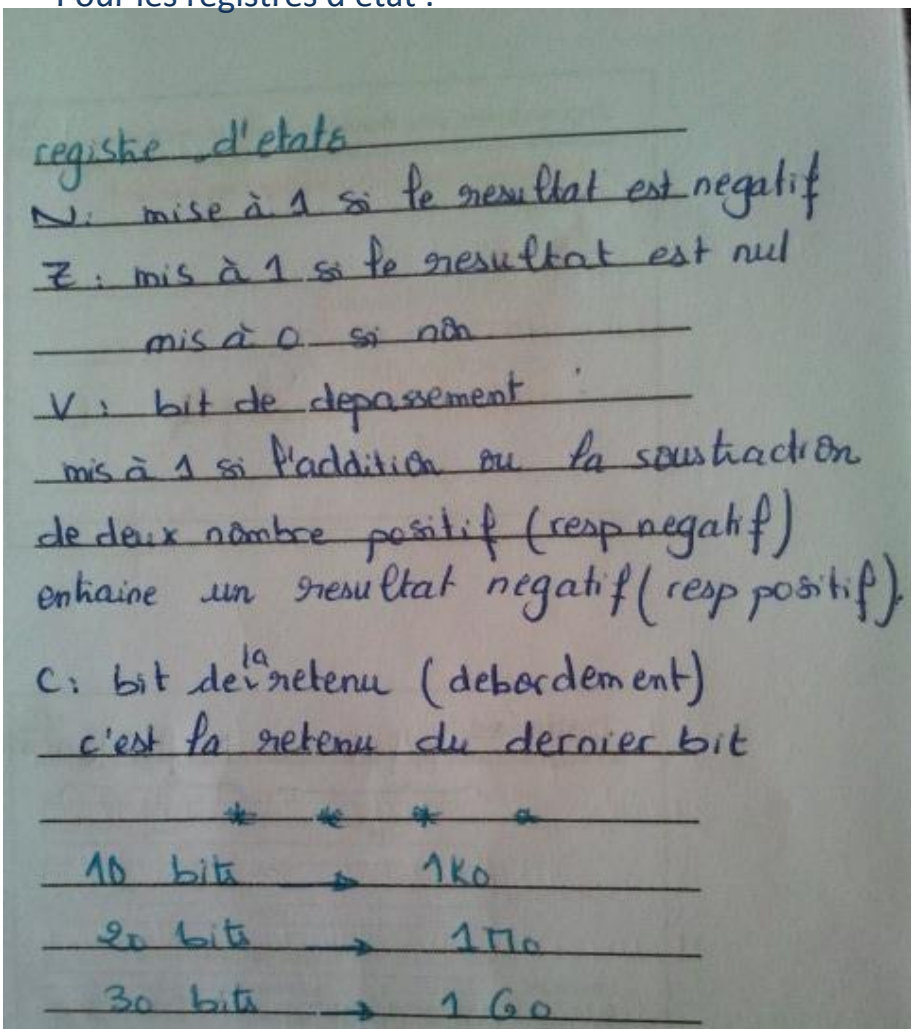
Tout d'abord, pour un registre de donnée on peut stocker des données sur (.B ,.W , .L ) par contre pour les registre d'adresse on peut seulement je dis seulement stocker (.w et .L) cad PAS DE .B .



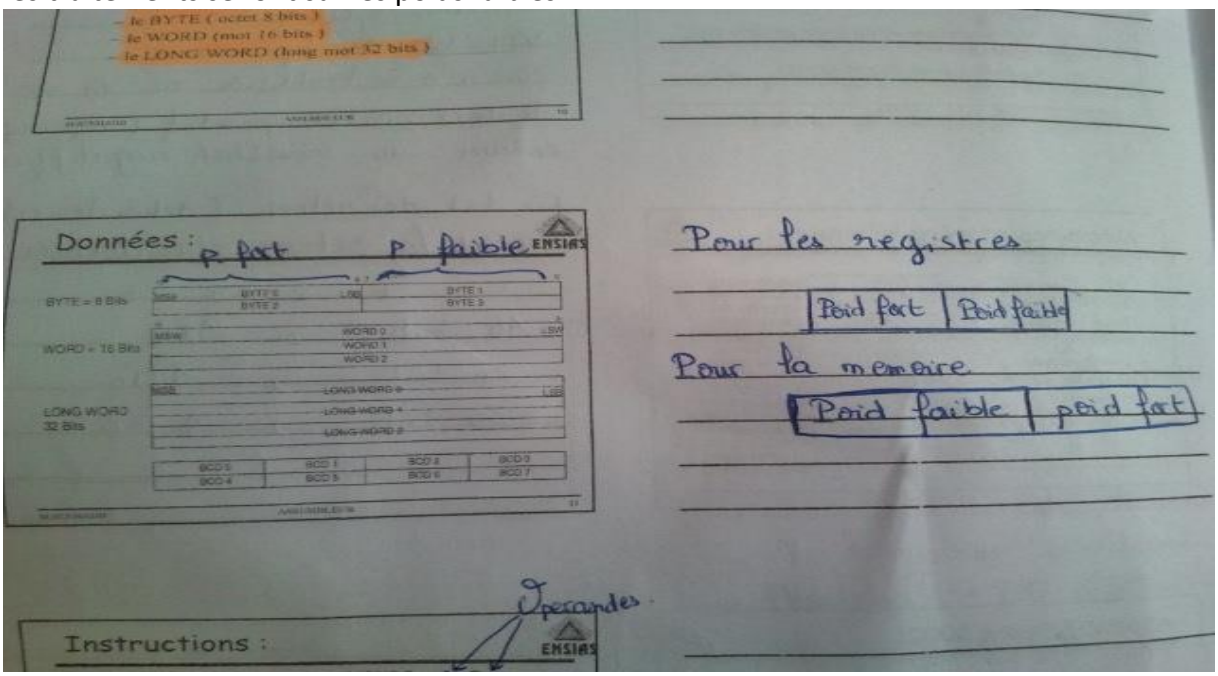

Pour les autres registres :




Pour les registres d'etat :



On verra des exemples après !
Puis après tu dois savoir que pour un registre le poids fort est la moitié gauche , et le poids faible c'est la moitié droite , par contre pour la mémoire c'est l'inverse . je dis ça parce que les traitements se font sur les poids faibles.

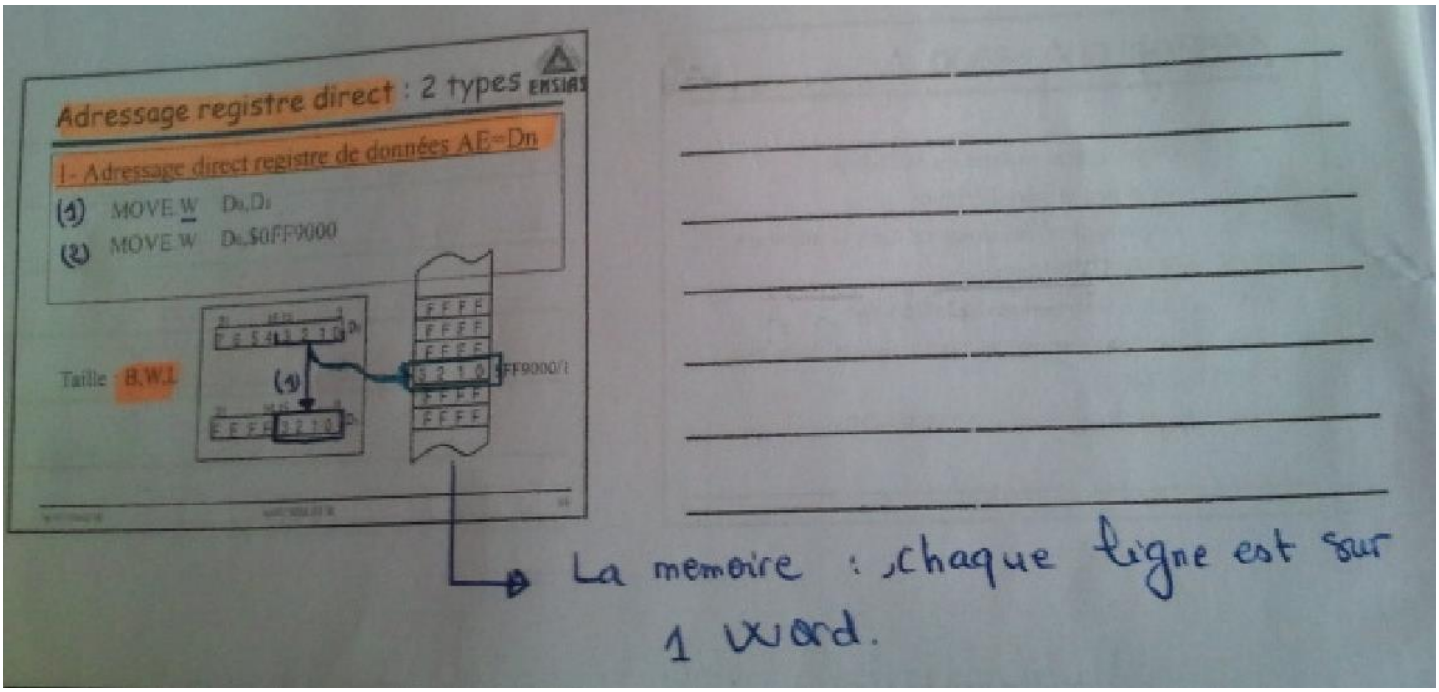



Tu dois savoir aussi autre chose :
Le code opération de toutes les instructions 68000 est fixé sur 1 word ,cad quand tu vois Par exemple move.X XX,XX avant meme que tu vois les modes d'adressage et tt tu ajoute 1word si jms taygolik sur combien est codé l'instruction,puis apres tu ajoute les words dyal l'extention oula dyal la taille .L ou .W. on verra ca après
Autre chose une instruction est codée au maximum sur 5 word .
**********
**********

## Mode d'adressage

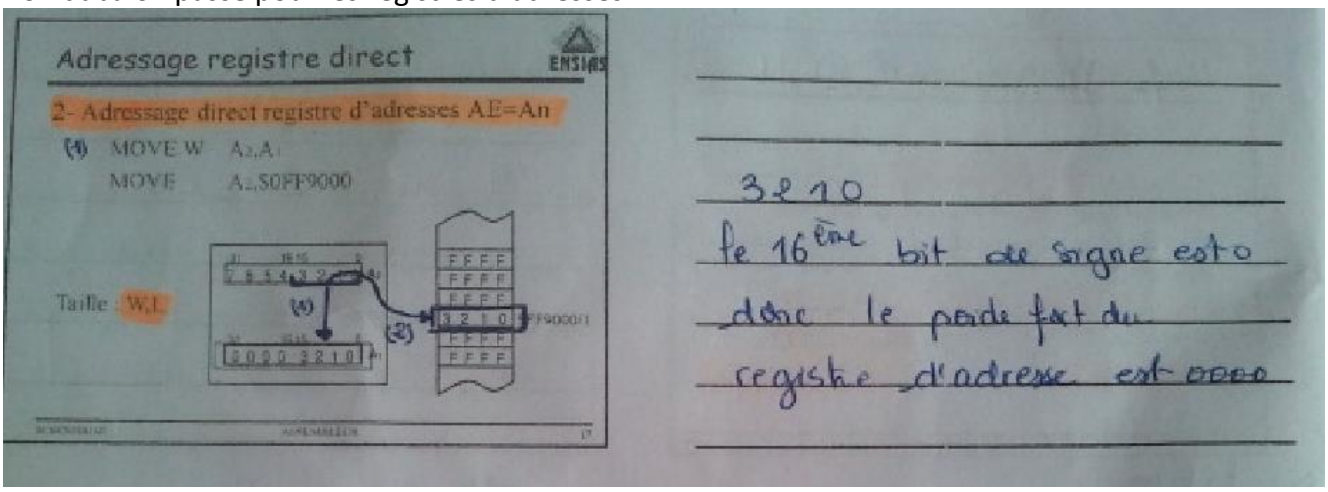
Comme notation :
$ : donnée est en HEXADECIMAL
# : donnée est une constante décimale

### Adressage registre direct :




Si c pas claire tu trouveras sur ton poly page 6 .
Bon là j'ai un move.W D0,D1 ,comme on a dit précédemment on fait les traitements sur les poids faibles , ici le poids faible c'est la partie droite du registre et je transfère la valeur du poids faible de D0 qui est 3210 dans le poids faible de D1 , et je touche pas ce qui précède cad les FFFF qui se trouvaient dans D1 o debut .
Pour MOVE.W D0, $0FF9000 je tranfère le poids faible de D0 ,(.w) cad 16 bit cad 3210 , dans une case mémoire ayant l'adresse $0FF9000.
Si jamais on avait un move .B par exemple je prends le poids faible du registre qui est 10 et je le fait dans le poids faible de la mémoire qui est l'extrémité gauche de la case mémoire $0FF9000. Hope it's clear.

Bon daba on passe pour les registres d'adresses .

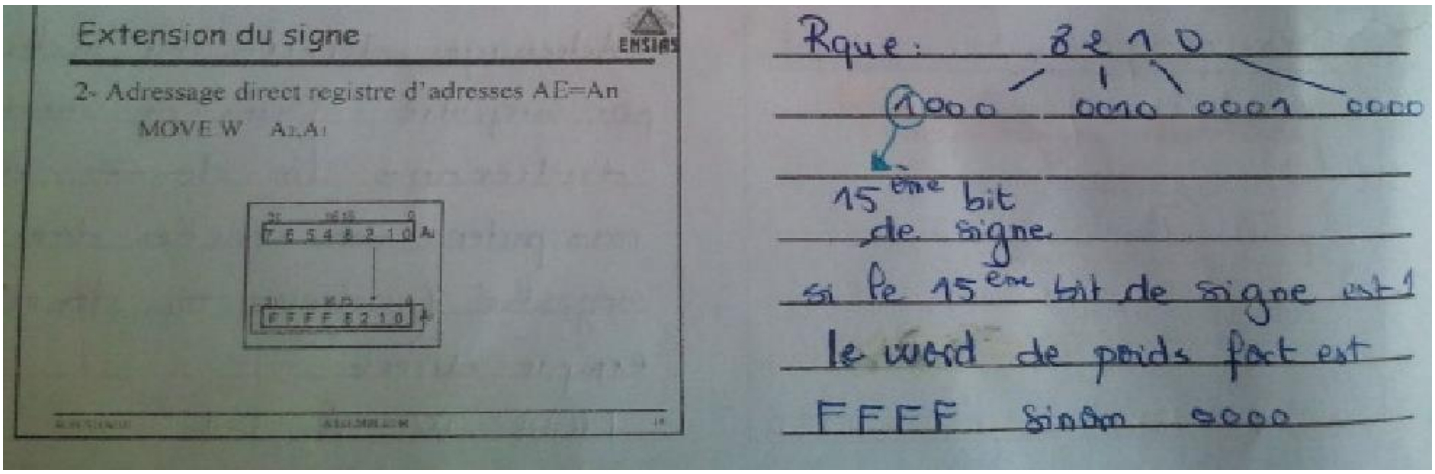



Bon pour move.w A2,A1 je prends le poids faible du registre A2 et je le met dans A1, la question qui se pose : avec quoi je v remplir les 16 bits du poids fort du registre A1 , et bah là la valeur qui se trouve dans A2 c'est 3210 , le 16$^{ème}$ bit de signe je l'obtient en convertissant

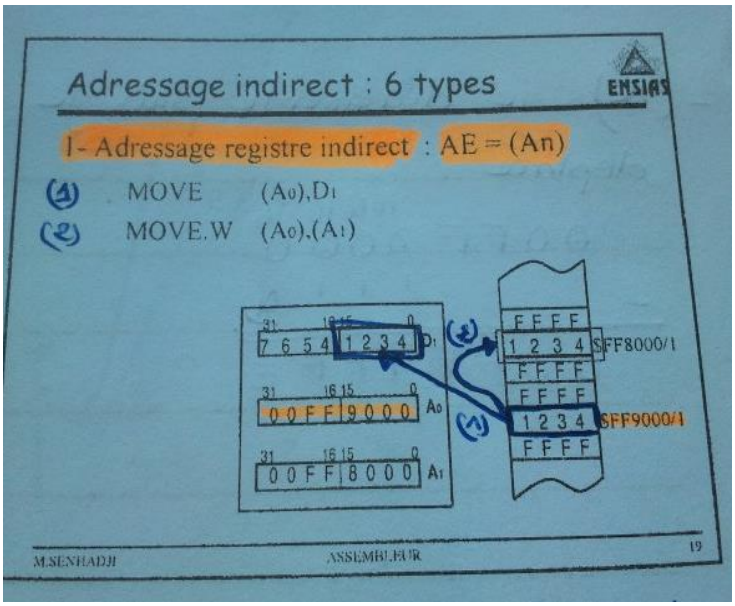
3 en binaire c 0011 , le dernier bit a gauche c 0 donc je rempli le poids fort de A1 par des 0 , si jamais c'était des 1 je le rempli par des FFFF.

Pour move  A2,$0FF9000 (rapp on a jamais de move .B pour les adresses)

Bon exemple dyal quand on rempli avec des FFFF la partie droite du registre d'adresse ,




Adressage indirecte



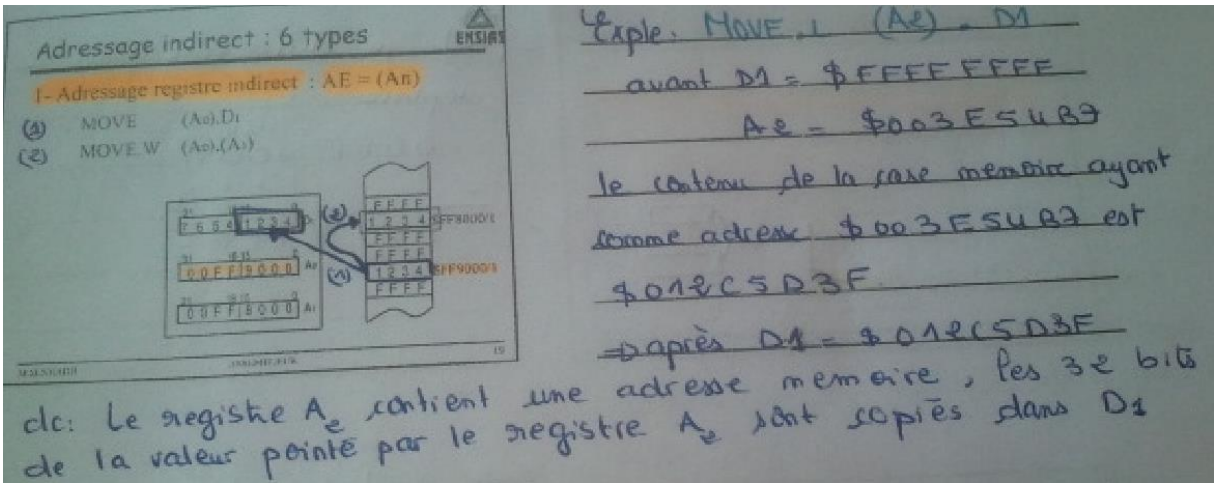

Les () indique : indirect , il ne peut se faire que sur les registres d'adresses .

MOVE.W (A0), D1

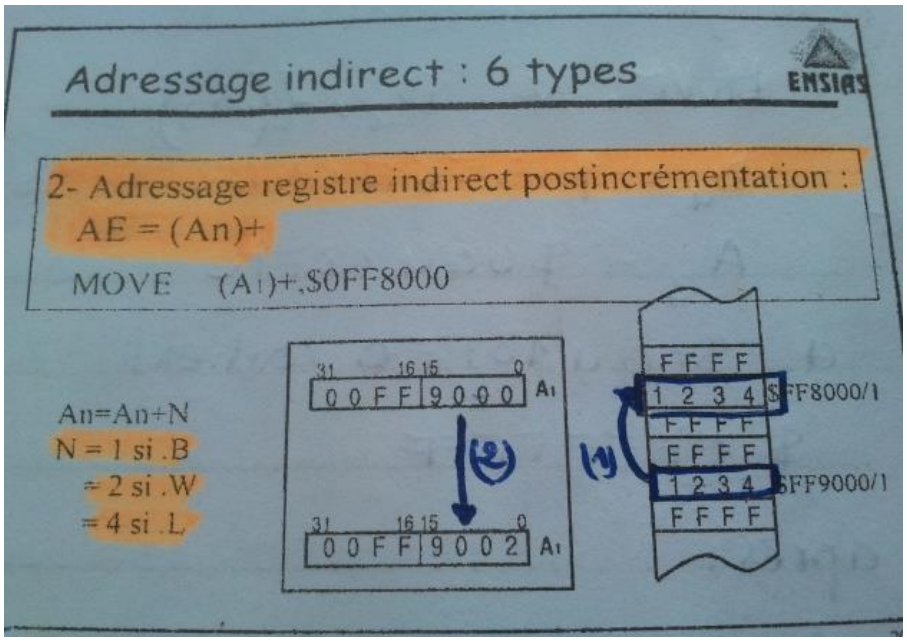
Ici A0 contient 00ff9000 , ainsi (A0) tu prends le contenu de la case mémoire 00ff9000 cad 1234 pour ce cas et tu la déplace dans le poids faible de D1 , on ne touchant ps le contenu des poids fort du registre D1 .

Si par exemple on avait un .B je vais prendre seulement le poids faible de la mémoire de la case mémoire ff9000 cad 12 et je le met dans D1 en remplaçant 34 par 12 dans d1 .

Voici un autre exemple .




Adressage indirect avec post incrémentation.



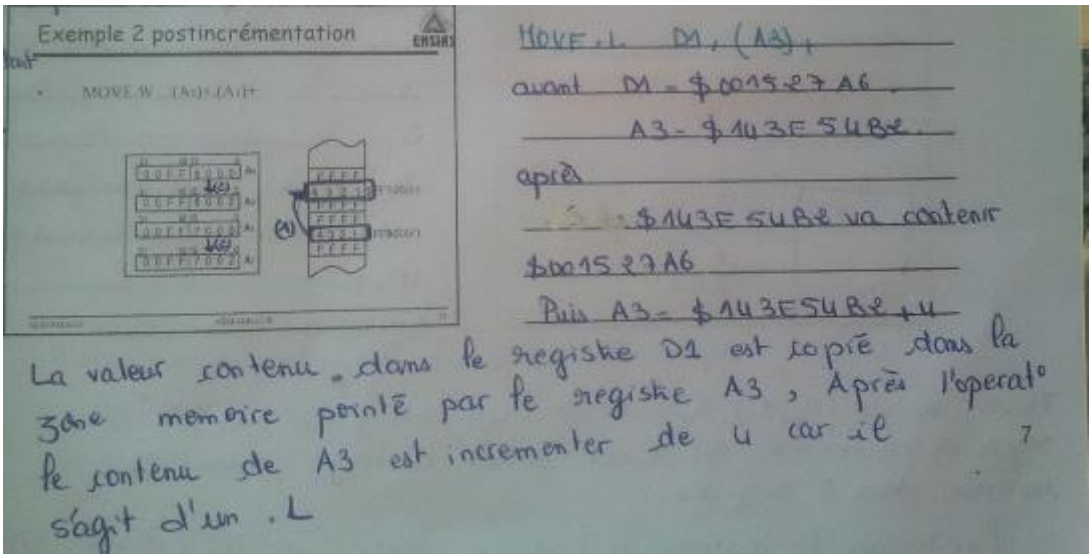

Move (A1)+ ,$0FF8000

A1 contient 00ff900 , donc tu déplace le contenu de la case 00ff9000 dans 0ff8000, la deuxième etape c'est que tu dois incrémenter le contenu de A1 , cela dep si c .B tu incrémente de 1 , si c'est .W tu incrémente de 2 , si c .L tu incrémente de 4 ,

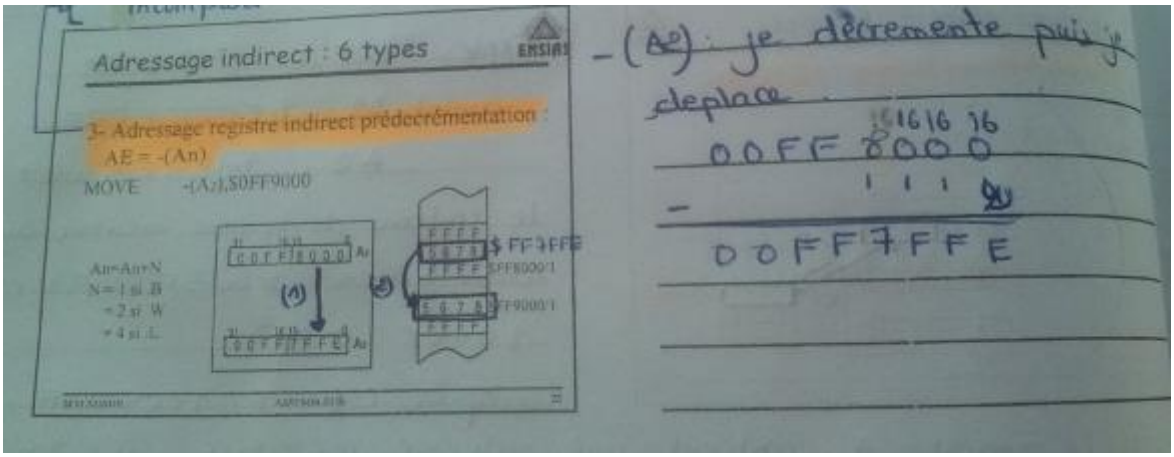
Bon ici move.W donc tu incrémente 00ff9000 de 2 donc A1 se modifie est devient 00FF9002.

Bon là un tit astuce pr aider ta mémoire ,quand tu vois (A1)+ cad tu fait le traitement de l'instruction par exemple move puis tu incrémente , je te le dis car après tu trouveras pré décrémentation -(A1) , donc tu décrémente puis tu fais le traitement .

Autre exemple :

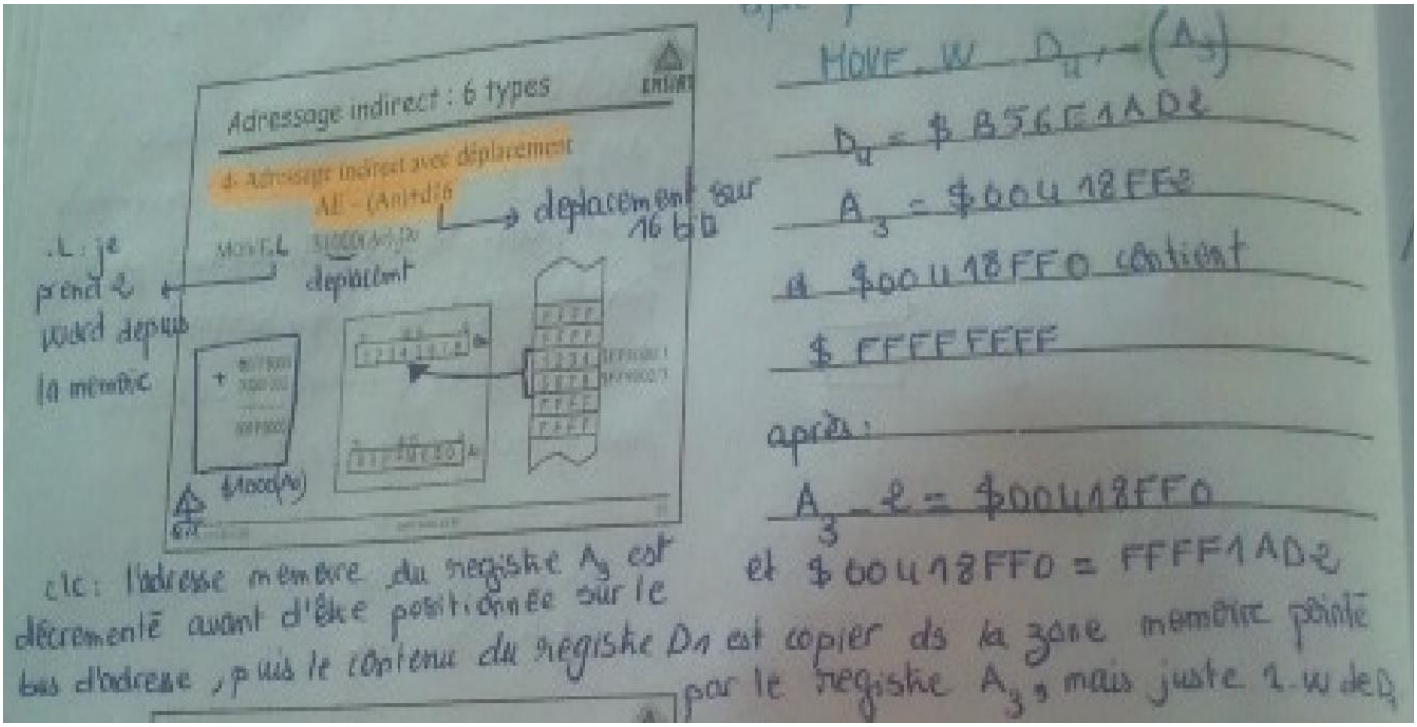



Adressage indirect avec prédecrémentation .




Bon là tu décremente le contenu de A2 de 2 puisque c'est un .w ! puis tu fait le traitement de l'instruction move en utilisant la nouvelle valeur de A2 . bon la soustraction se fait en hexadécimal , c écris la dessus .

Adressage indirecte avec déplacement.




Page 8 slide 23 ,c pas claire ici .

MOVE.L $1000(A0),D0

Le (A0) comme on a déjà vu avant c la case mémoire du contenu de A0,

Le $1000 c le déplacement ,

Donc je fais le contenu de A0 plus 1000 :

00FF8000 + 00001000(c'est un long) = 00FF9000

Bon comme on a déjà vu qu' une ligne de la mémoire est sur un word or ici c'est un .L , donc je prends deux ligne commençant par par le consutenu de l'adresse 00ff9000

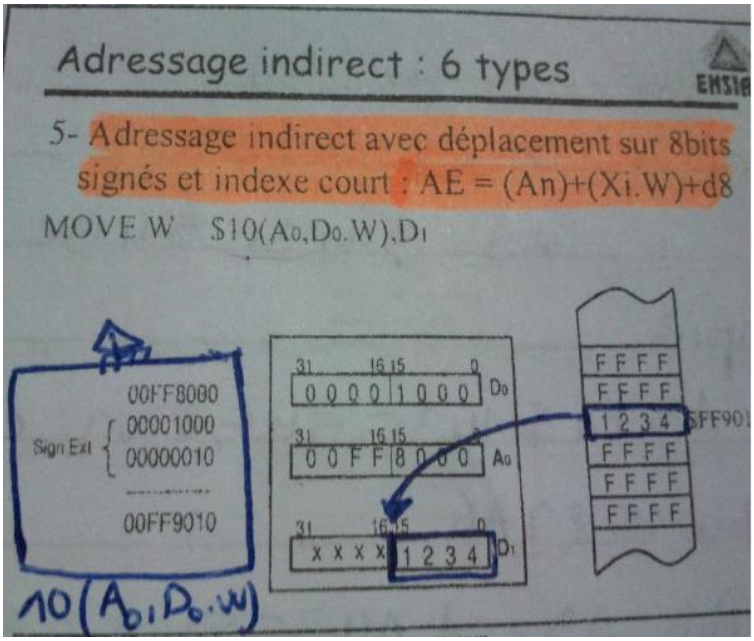
Donc je prends 12345678 et je le met dans D0.

Exemple :

MOVE .B $12F4(A0),D2

Copie la valeur pointe par A0+ $12F4 dans le registre de donnée D1,mais seulement le .B

Adressage indirect avec déplacement sur 8 bits et indexé court




Une chose quand tu vois $10 ca veut dire que le 10 est en hexadimal et des fois il ecrit

Move.w 10(A0,D0.W),D1

Ici indexé court car on a D0.W on ne voit pas le move.W .

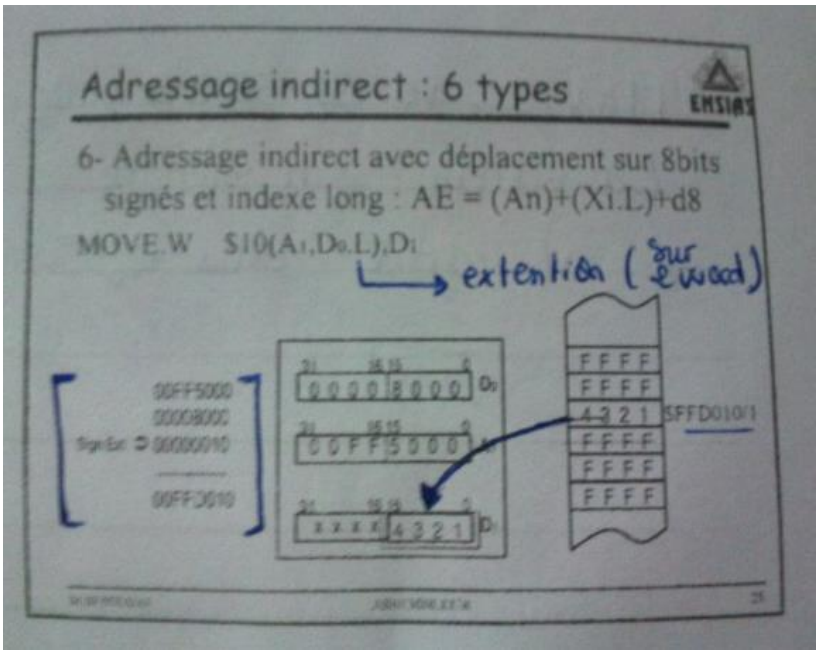
La le 10 est une constante décimale, mais en assembleur on travaille avec l'hexa donc on va faire le meme traitement qui va suivre sauf qu'a la place de 00000010 on fait 000000A , (l'hexa de 10 est A)

Move.w $10(A0,D0.W),D1

Cette instruction permet de déplacer le contenue de l'adresse (00ff8000+00001000+00000010 =00ff9010) qui est 1234 dans le poids faible de D1.

Adressage indirect avec déplacement et indexe long .

Une fois qu'on voit indexe long donc il va s'agir d'un (registre de donnée).L



C’est donc le même principe de celui qui précède . la seule diff c’est l’extension

- ➔ On parlera maintenant d’un peu d’extension  mais on va en revenir après
  L’extension est exigé quand il s’agit de mode d’adressage suivant :
  -constante (cad mode immediat )
  -adressage absolu
  -déplacement
  -mode d’adressage indexe.

Ici pour le mode indéxe quand on a l’extension, on l’utilise dans le codage ou pr savoir sur combien de word est codée une instruction , bon on verra ca apres, mais la diff entre c’est deux type de mode d’adressage indexe (court et long) c’est pour le court l’extention est sur un word , pr le long sur  deux .

- ➔ Une remarque piège il le fait dans les exams  j’essayerai de faire des exemple après ,

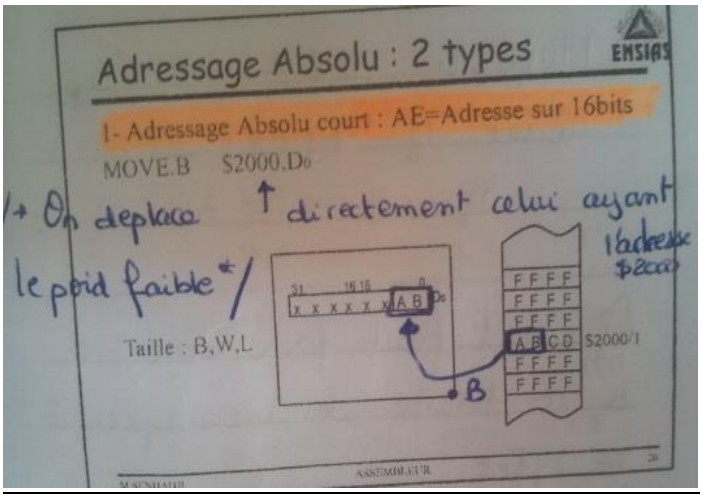
Mais il faut noté que l’adresse effective obetenu apres (deplacement+ An +Dn) doit être paire si on travaille avec .W et doit être un multiple de 4 si on travaille avec .L

En effet pour la mémoire c comme suit

| 0 | 1 |
| --- | --- |
| 2 | 3 |
| 4 | 5 |
| 6 | 7 |
| 8 | 9 |
| 10 | 11 |

Donc si jamais on commence avec l’adresse FF8001 par exemple c une erreur pour un .W mais pas pr un .B , le .W doit commencer par les pair pour les remplissage de la mémoire ,cad ff8000 , FF8002 .
..

<u>Adressage absolu</u>




Pour l'adressage absolu c'est simple.
Move.B $2000, D0, On déplace directement la valeur stocker dans la case mémoire $2000 dans D0 mais seulement le .B

Pour absolu court : l'adresse à manipuler est codé sur 16bits max (exmple $2000)

Pour absolu long : l'adresse à manipuler est codé sur 32 bit max exemple ($FF8000)

Exemple :

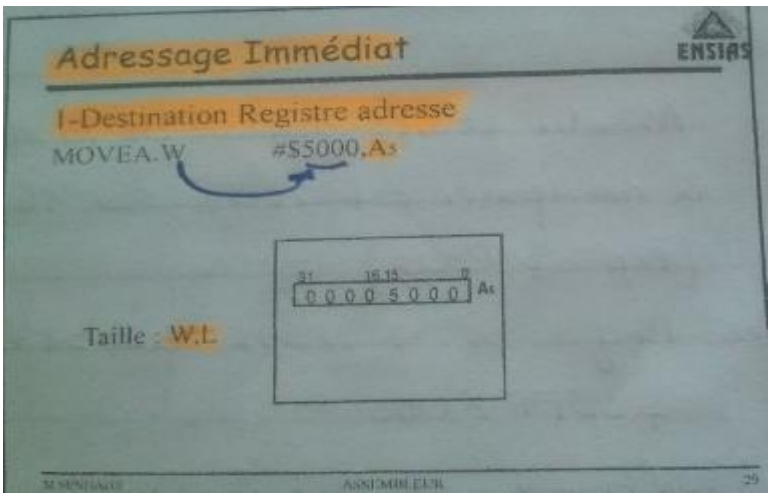
MOVE.L $14FE, D0

Avant D0=$FFFF FFFF ET $14FE = $13DFE01E

Après on aura D0 =$34DFE01E

## Adressage immédiat

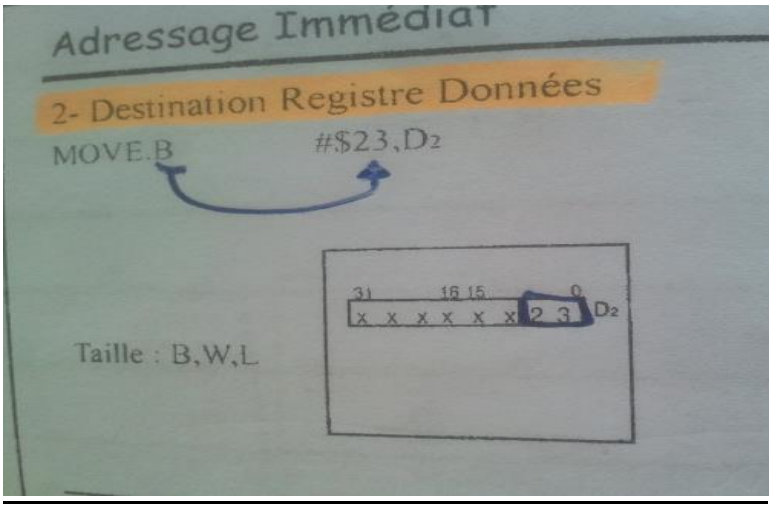
### 1- Destination registre d'adresse.



On transmet directement la constante 5000 dans le poids faible de A5 , (le $5000 => il est en hexa )

On remarque aussi que pour le registre d'adresse on a pas de .B .

2- Destination registre de donnée.



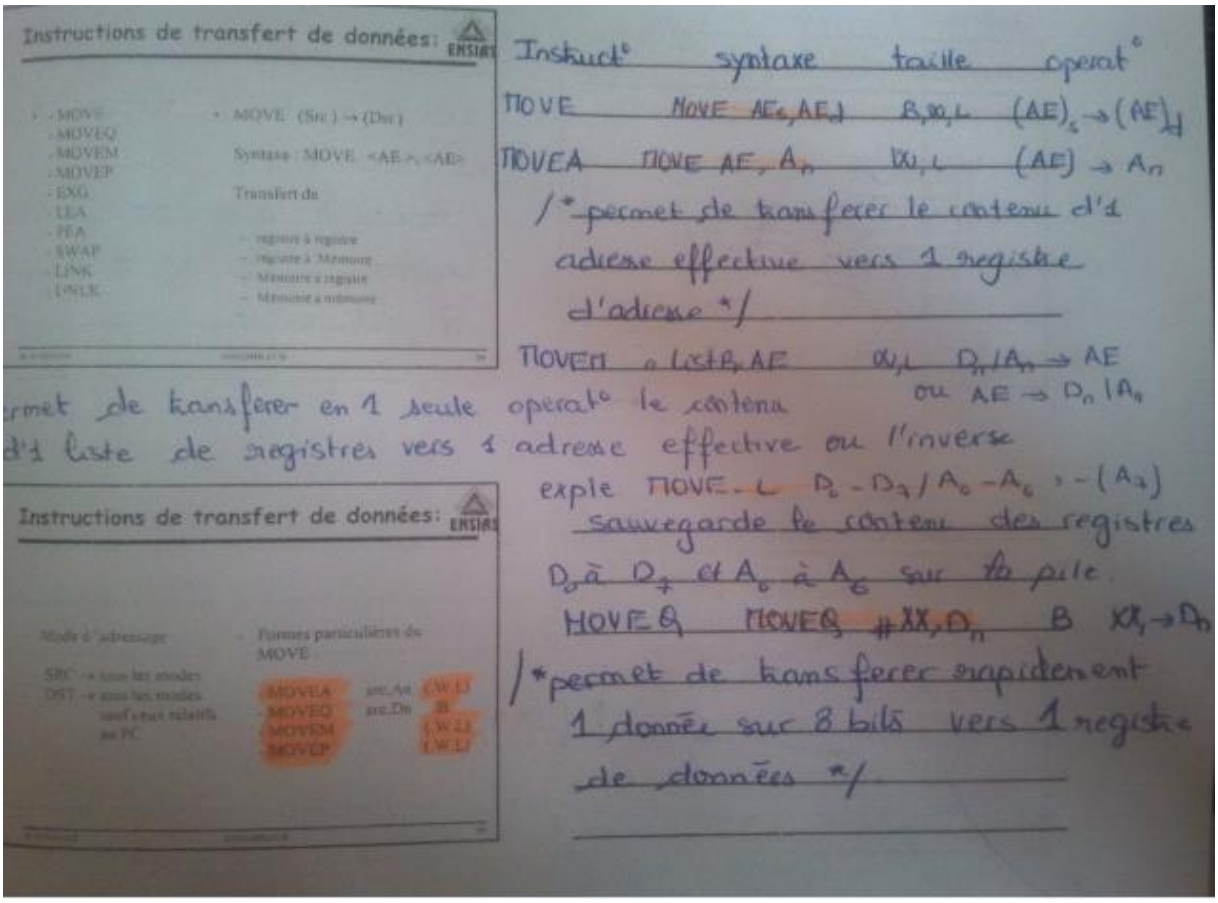

Même principe juste ici.

Il faut faire attention à la taille du mot spécifié en effet

MOVE.B #$ 356D, D0

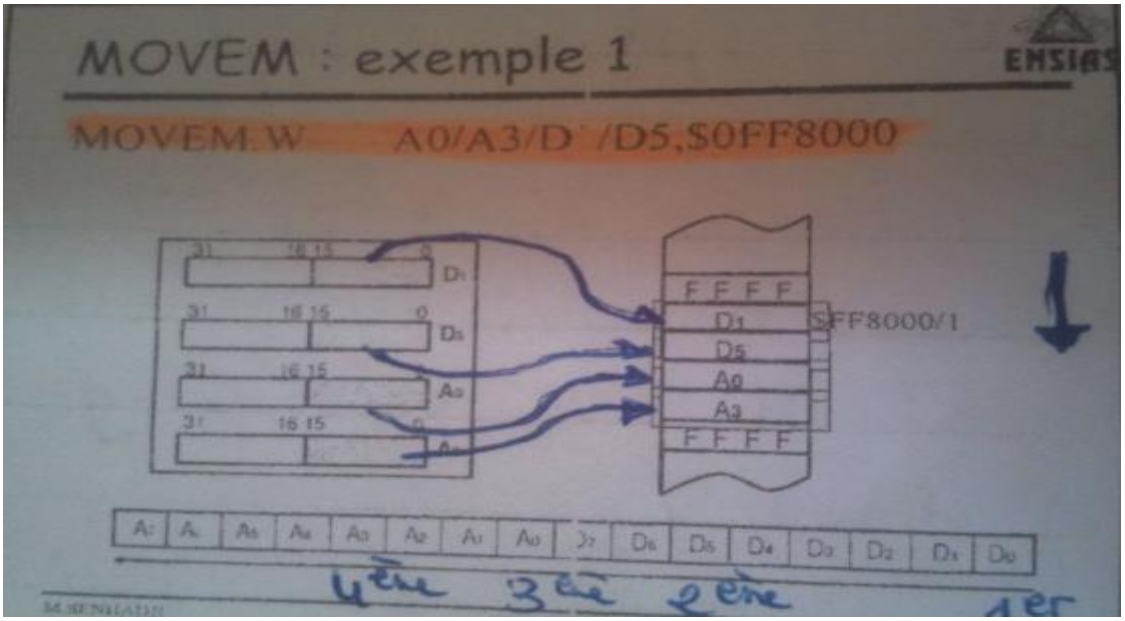
Génère une erreur car la valeur 356D est sur 1Word.

**********
**********




Quand j'ecris A0|A3|D5 je parle de A0 ,A3 ET D5
Si j'ecris A0-A4 , je parle de A0 A1 A2 A3 A4
Les adressage immédiat rapide sont codés sur 1 byte (.B)

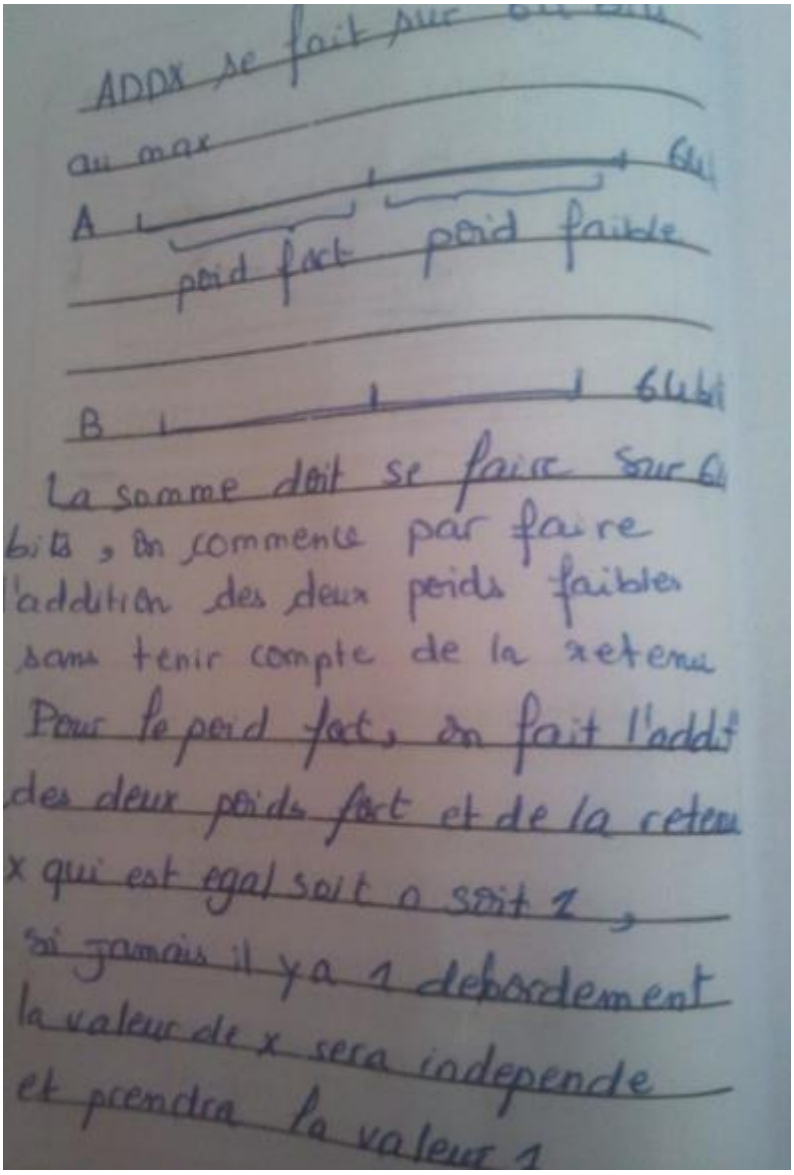
L'instruction MOVEM :




Je te donne ici un exemple au cas ou , bah la ya la barre li en dessous qui commence par D0 jusqua A7 , on commence la lecture de notre gauche ,
C'est le meme principe que move sauf que c'est pour une série , et le remplisage ici se fait de haut en bas, commencant par l'adresse FF8000 ,

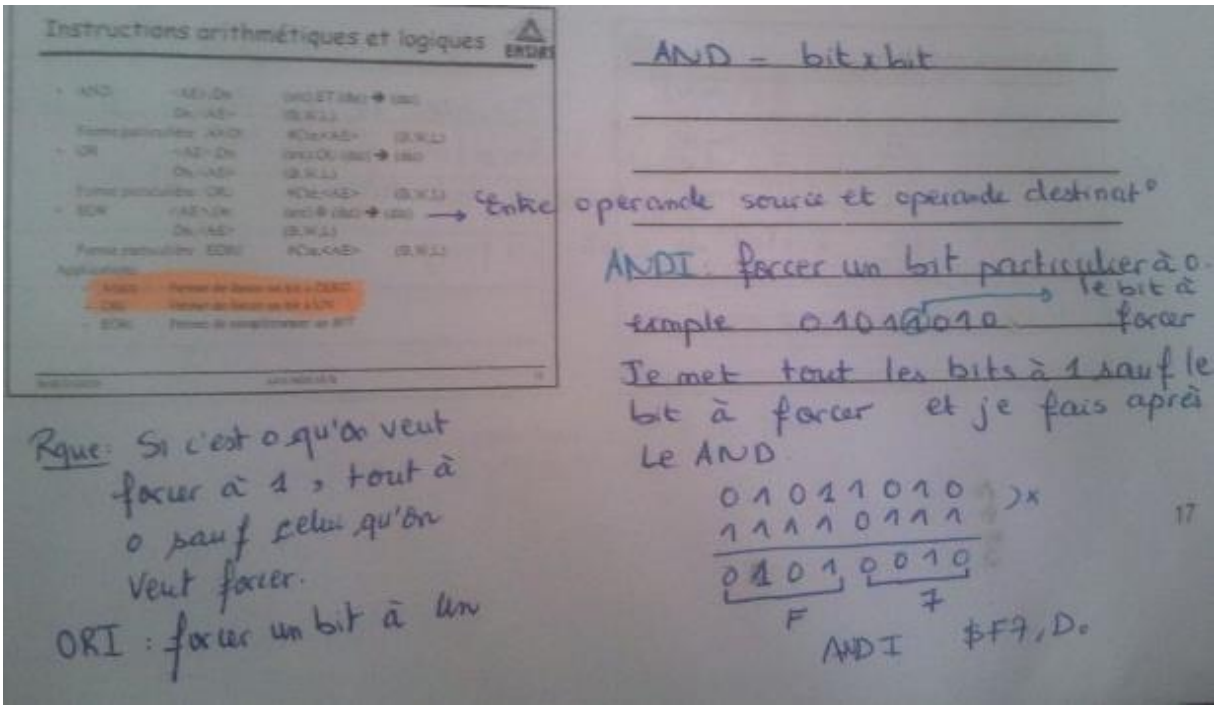
Si c postincrémentation comme c le cas pr le slide qui suit sur ton polycopié , eh bah et commence du bas vers le haut .
Pour l'exemple 3 du cours , (sp) + TU c l'inverse tu rempli les registres  a partir de la pile mémoire

L'instruction ADDX : ☹ je c que c long mais poly autorisé tu px t'en servir



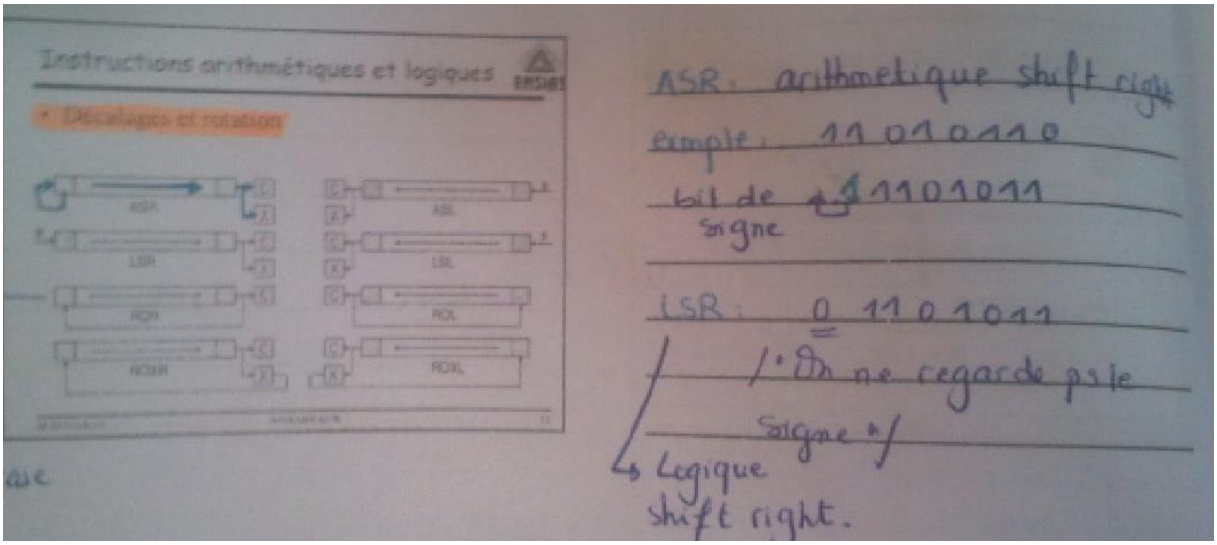

NEG ⇔ complément à 2 ⇔ complément à 1 puis on lui ajoute 1
Si jamais il demande le résultat de l'instruction Neg , tu met le nombre en binaire sur 16 BITS pour avoir le bit de signe aussi qui sera ici 0 si on parle d'un nombre + , puis on fait le Complèment à 1 cad le  0 devient 1 et vis versa , puis tu ajoute 1 . tu obient donc le complèment à 2.

L'instruction ANDI :



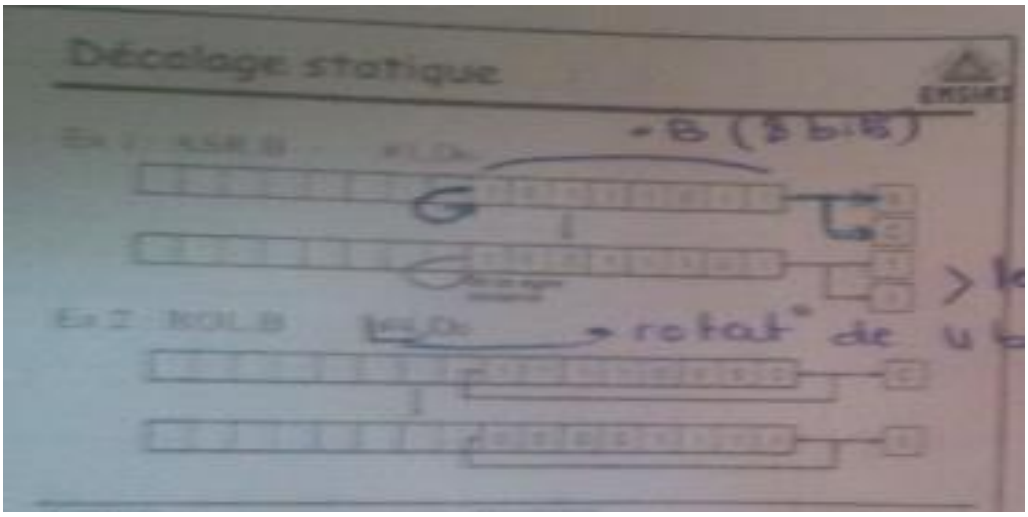

# Décalage et rotation :

Voir slide 52 sur le poly si c pas claire la dessous :




Si la suite était par exemple 11010110
Pour ASR en reprends le premier 1 ca divient 111010110 et je fais sortir le dernier 0 vers C et x .
Pour ASL je commence par ecrir 0 vers la gauche je translate le tout et je met le 1$^{er}$ a gauche à c et x

Pour OP.X #cte,Dn (slide 53)
L'op peut etre d'une des instruction vu la dessus
Cte : cte de déclage cad combien de fois je v décaler




C pas trop claire mais bon :
ASR.B #1,D0
Decalage du registre D0 d'une position à droite
ROL.B #4,D0
Rotation de 4 BITS (4 fois )

# Sur combien est codée une instruction ?

**Il ya toujours 1word pr le code opération et puis le reste des word c'est des extension et aux déplacement**

ADDI.L #3,D2
Le code opération est sur un word
L'extension ici c'est l'adressage immédiat dans la partie source (#3) donc on aura une extension ,l'extention dans le codage immédiat depend sil sagir de .L ou .W ou .B ,
Si c .b je fait l'extentiion sur 1Word en terminant le word par des 0
Si c .w L'extension est sur un word
Si c .L l'extension est sur 2 word
Sans ce cas : c'est .L donc sur 2 Word .
En total il ya un word du code opération , et 2 pour l'extension donc ADDI.L #3,D2 est codé sur 3 word

ADDQ #3,D2
Le code opération sur 1 word
Puisque ADDQ est un adressage imédiat rapide c sur 1 byte , (.B) l'extesion est donc sur 1word
Ainsi
ADDQ #3,D2 est codé sur 2 word
Remarque : ici c'est #3 donc c en décimal (car sans $)

MOVE.L #$02,D0
1 word pr le code opération
2 word pr l'extension de l'adressage immadiat
Donc codé sur 3 word

MOVE.L D0,(A1,D2)
1 word pr le code opération

Ici adressage indéxé court , puisque c'est D2.W donc extension sur  1word (remarque on a pas donnée aucune importance aux move.L ou .W) tout se qui compte c'est l'indexe
Donc codé sur 2word

MOVE.L  $20(A2 ,D5) ,$ff9000

1 word pr le code opération
L'opérande source : adressage indexé D5.W => 1word
L'opérande destination :  absolu long ➔ sur 2 word
Donc le tout sur 4 word

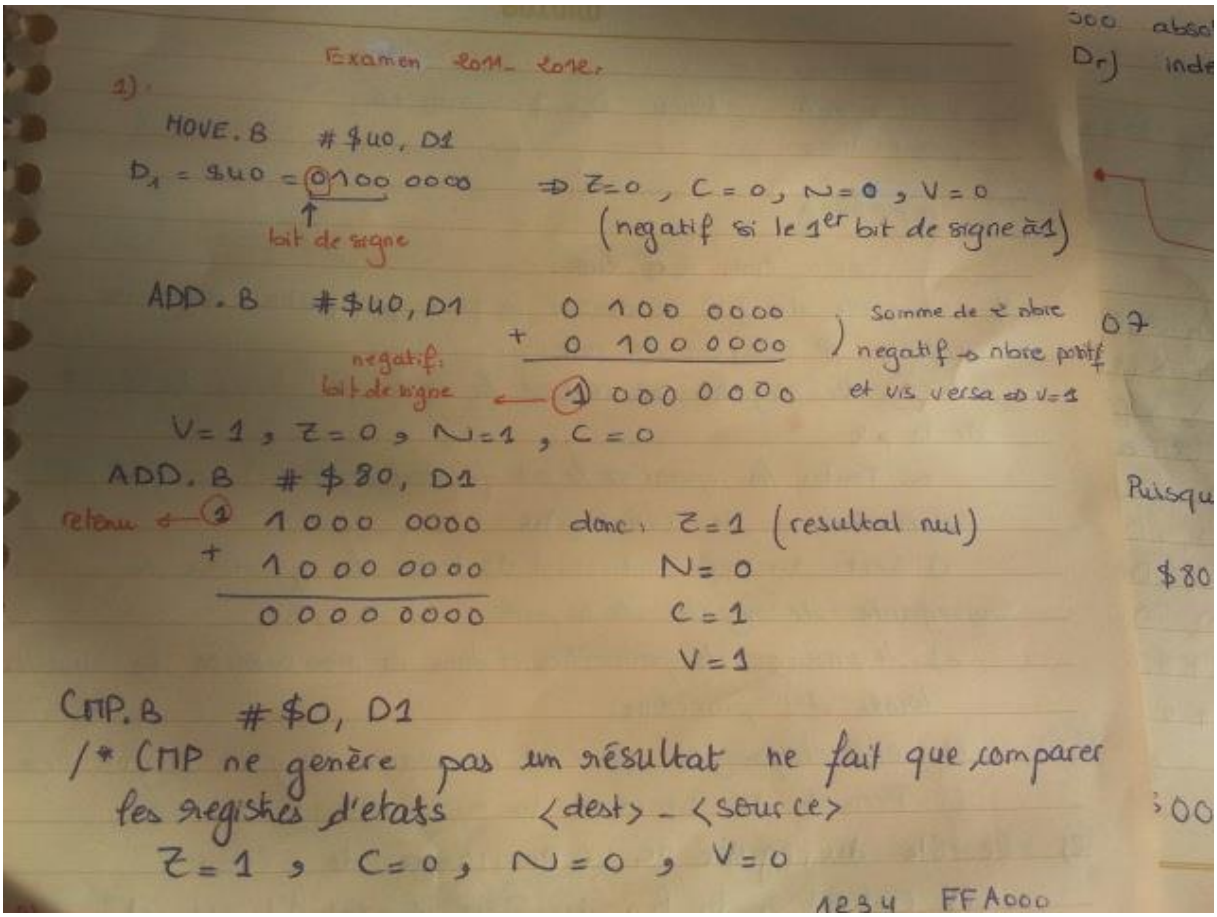
MOVE.L  $20(A2 ,D5 .L) ,$2000

1 word pr le code opération
L'opérande source : adressage indexé D5.L=> 2word
L'opérande destination :  absolu court  ➔ sur 1 word
Donc le tout sur 4 word

# Les valeurs de C, V, N, Z

Si le nombre du résultat est négatif ➔ N=1
Si positif➔ N=0
Resultat nul => Z= 1 si non mis à 0
Si par exemple le traitement de deux nombre positif donne un nombre negatif ➔ V=1
Si par exemple le traitement de deux nombre négatif donne un nombre positif➔ V=1
(donc changement de signe) => V= 1 sinon V =0
Si il ya un une retenue ➔ C= 1 si  non C=0

Exemple :

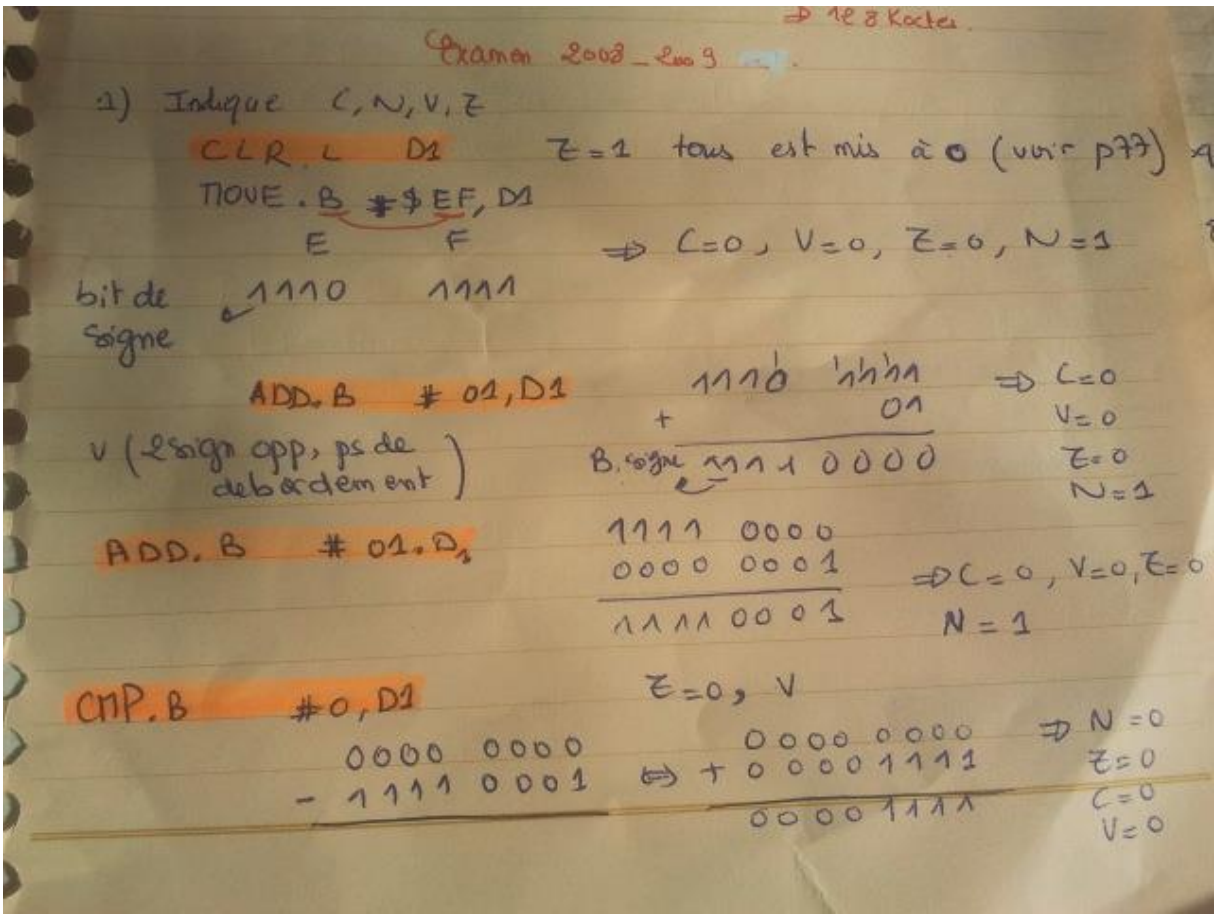



Explication :
MOVE.B #$40 ,D1
Pour faire les traitement il faut tout dabord mettre 40 en binaire ➔ 0100 0000
le 1er bit est 0 donc positif => N=0
V=0 pas de changement de signe
C=0 pas de retenu
Z=0 resultat non nul

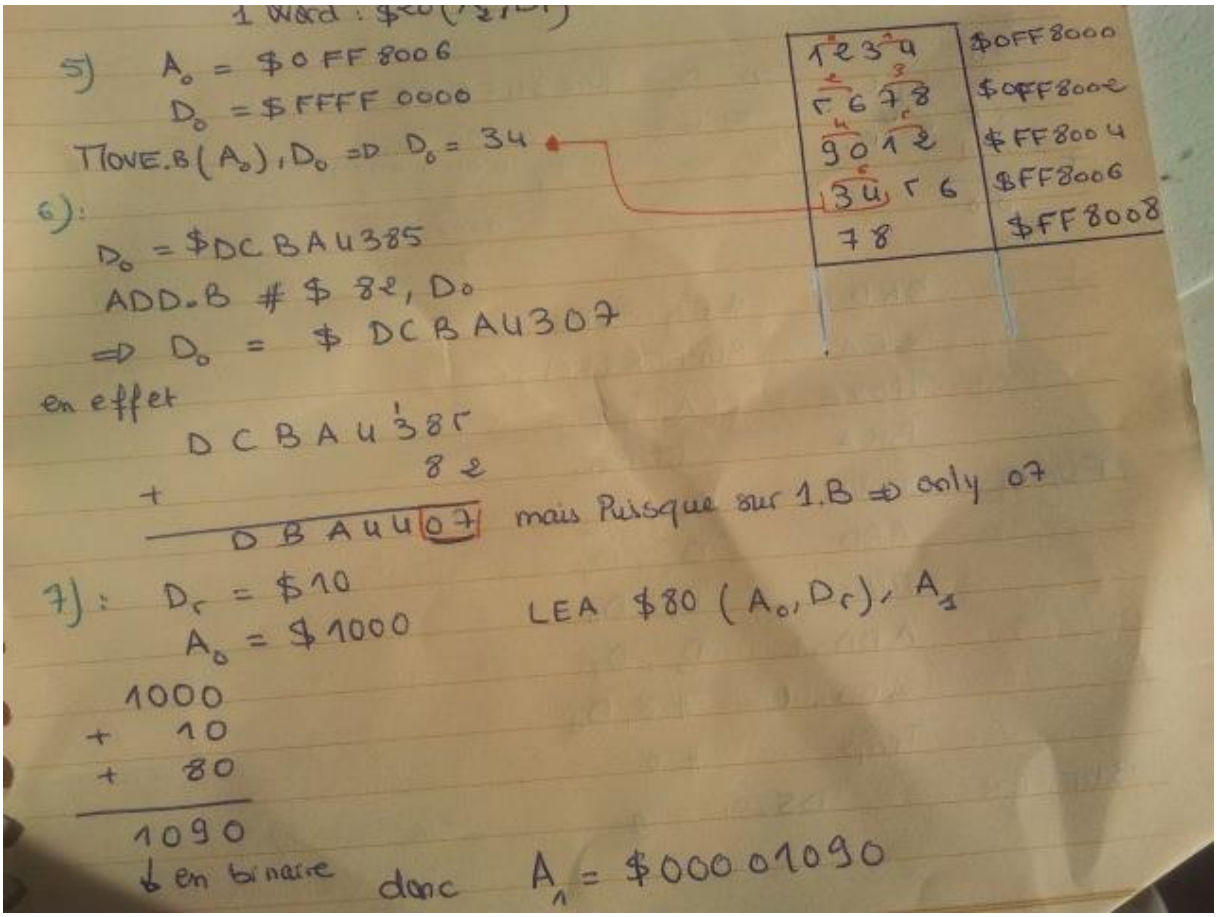
On fait de sorte que les traitement sont consécutif ,la premiere instruction a stocker 40 en hexa dans D1

ADD.B #$40,D1
(voir laddition sur limage)
La somme est negatif (bit de signe mis à 1)➔ N= 1
Somme de 2 nombre positif => negatif ➔ V=1
Pas de retenu ➔ C= 0
Resultat nn nul ➔ Z= 0
Tu peux faire les autres pour mieux comprendre et voir dautre cas

Voici d'autre :




# Résultats d'instructions




# Explications :

## QUESTION 5 °

Le prof avait dit que dans la mémoire il yavait la suite 1234567890124567.. commencent par l'adresse FF8000 bah on commence par remplir la mémoire et pour chaque word on augmente de 2 , bon on a déjà vu la présentation de la mémoire ,et que pr les word il faut des pairs . donc chaque ligne c ff8000 ,ff8002 ,ff8004 et ainsi de suite

Move.B (A0),D0 ➔ le contenu de l'adresse stocker dans A0 est transmet A d0 MAIS SEULEMENT en .B

Avant D0 =$FFFF0000

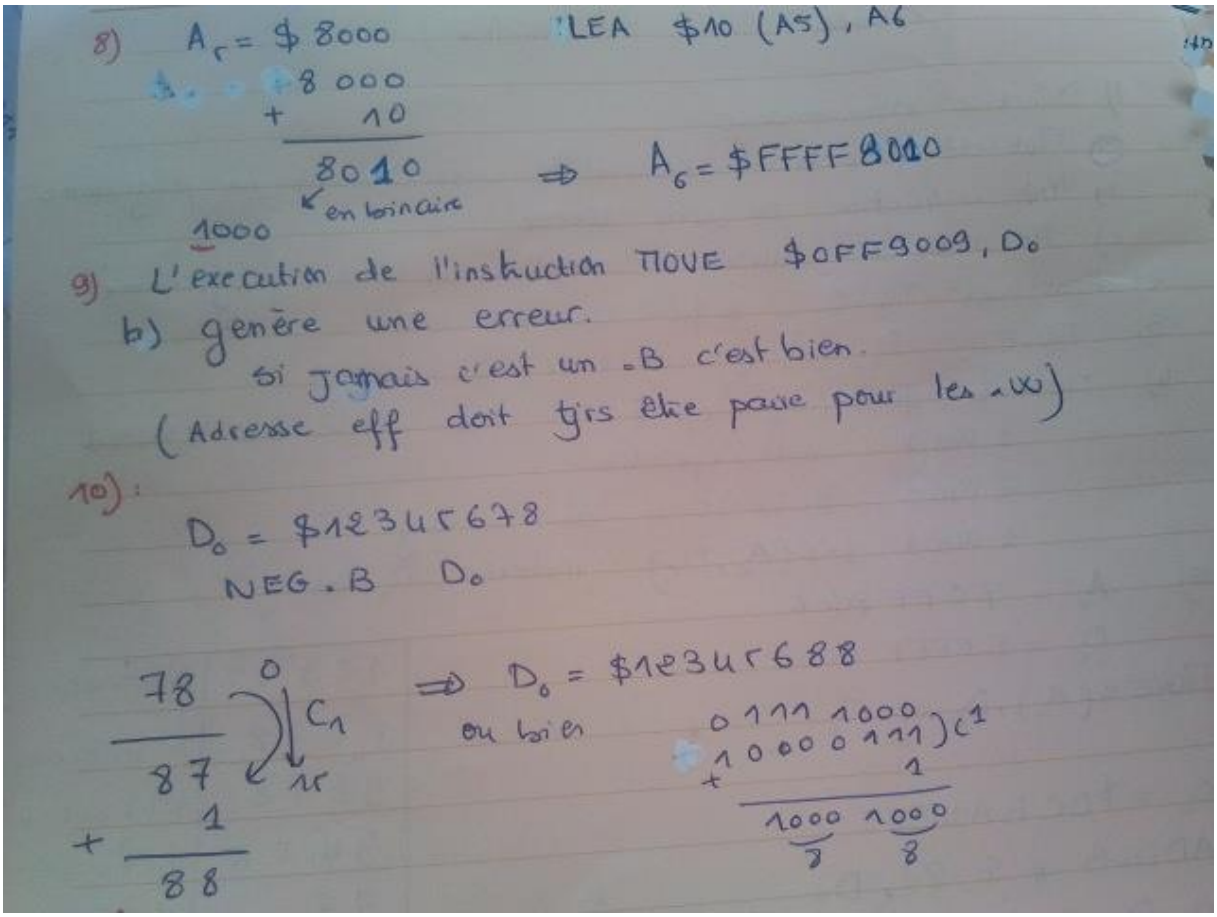
Et le contenu de la case mémoire FF8006 est 3456 mais nous on prends juste le .B donc only 34 .

D0 devient donc ➔ D0 =$FFFF0034

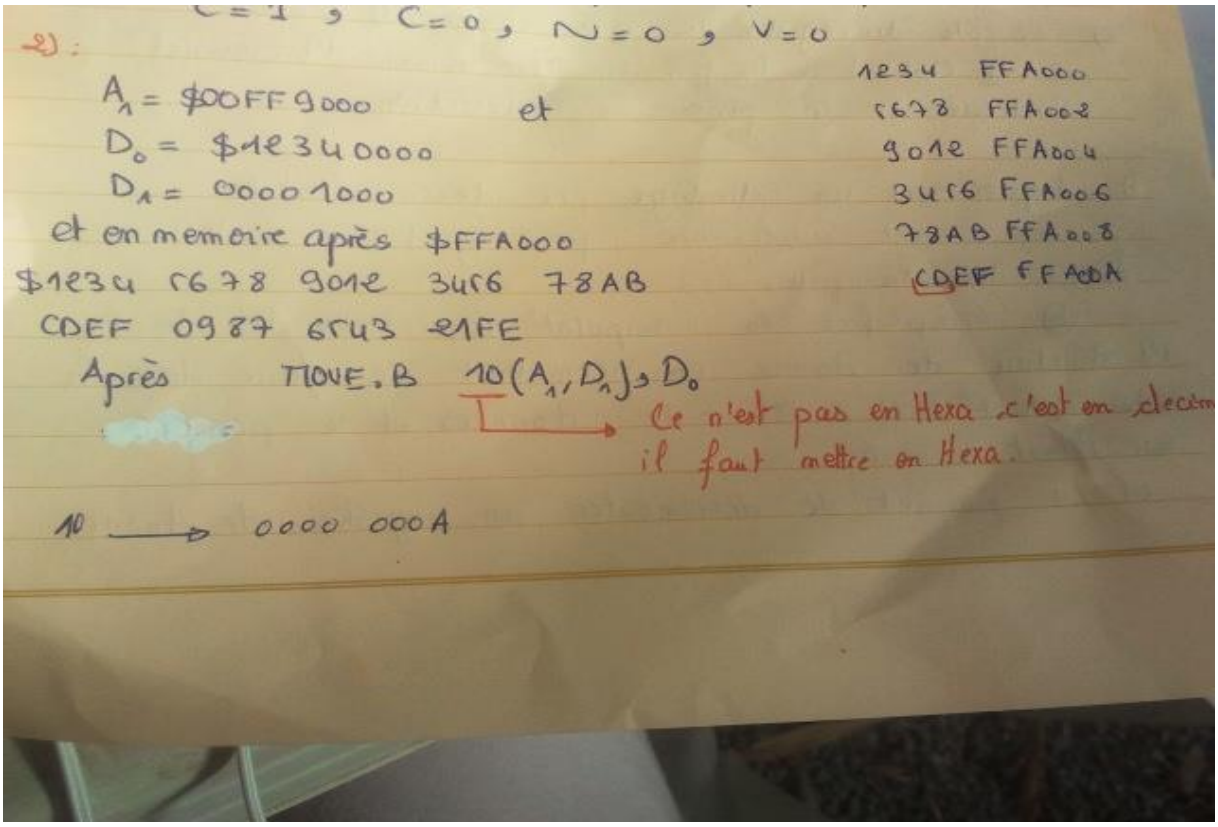
Question 6 ) c claire je pense , je fais laddition normalement mais je prends juste le .B .

Question7 ) : bon LEA c'est exactement comme move sauf c pr les registres d'adresse

Donc je fais l'addition qui est claire sur l'image, on obtient 1090 comme 1 en binaire est 0001 alors on complète le rgistre A1 par des 0 .




8) parail juste ici 8 en binaire commence par 1 donc on complète par des F
9) : move $0FF9009,D0 genere erreur comme on a dit pr word ladresse doit commencer par pair
10- neg.B D0
Le byte ici c seulement 78 ,
On fait le complement à 2 = complément à 1 +1
Soit tu px le faire en binaire
Mais là j'ai fais un complément en hexa.




2) On a ici dans la mémoire 123455678901234567ABCDEF …. Commencant par 0FFA000.
MOVE.B 10(A1,D1),D0
Remarque que c 10 non pas $10 donc on doit convertir en hexa donc le 10 devient A en hexa
On fait donc le calcul 10+A1+D1 et donc on obtient