CHAPITRE : 02 MODES D'ADRESSAGE

MP 6809 MOTOROLA

1.Définition :

Des modes d'adressage puissants alliées à un jeu d'instruction très complet donnent au microprocesseur 6809 d'importantes possibilités logicielles.

En effet le MPU 6809 possède 59 instructions de base mais en conjonction avec les neuf modes d'adressage disponibles. Où parvient à 1 464 possibilités.

Ce chapitre présente tous les modes d'adressage du 6809 avec leurs avantages leurs limites. Une importance toute particulière a été donnée au mode d'adressage indexé. Très puissant sur ce processeur.

2.Modes d'adressage du 6809

2.1 Adressage inhérent

L'adressage est utilisé par les instructions qui agissent sur les registres internes du MPU et non pas sur la mémoire. Il existe deux types de mode d'adressage inhérent sur le 6809 :

1. **Adressage inhérent simple :** Le code opération contient toute l'information nécessaire à l'exécution de l'instruction. Ces instructions codées sur un octet sont : **ABX, ASLA, ASLB, ASRA, ASRB, CLRA, CIRB, COMA, COMB, DAA, DECA, DECB, INCA, INCB, LSLA, LSLB, LSRA, LSRB, NEGA, NBGB, ROLA, ROLB, RORA, RORB, RTI, RTS, SEX, SWI, SYNC, TSTA, TSTB**. L'instruction de **SW1** et **SW2** nécessite chacune 2 octets.

Exemple :

ABX ⟶ Addition l'accumulateur **B** à l'index **X** **[Code (3A)]**

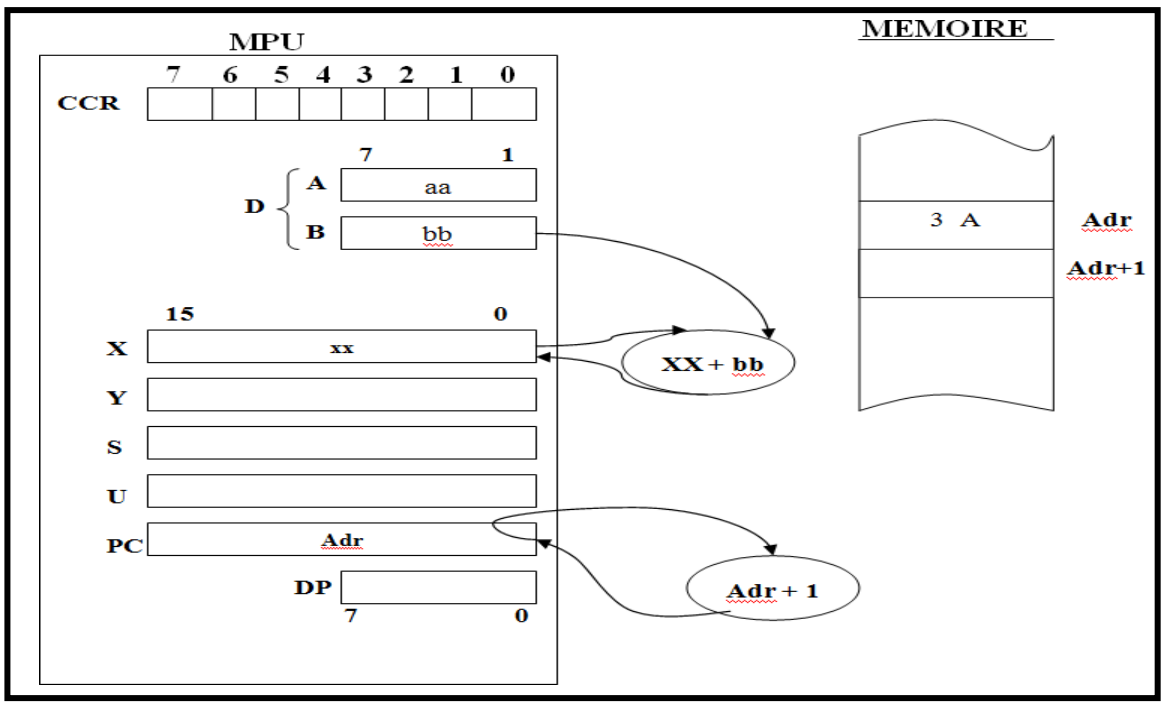


FIG 01** Addition l'accumulateur **B** à l'index **X

2. **Adressage inhérent paramétré :** L'instruction compte un octet supplémentaire permettant de préciser les opérandes intervenant dans l'instruction. La présence de cet octet supplémentaire est indispensable pour les instructions de type :
   - échange et transfert de registres.
   - Instruction d'accès aux piles.
   - Attente d'interruption.

Echange et transfert de registre : le **premier octet** détermine le code opératoire (**OP Code**) pur, le second les registre **source** et **destination**.

Exemple :

TFR U, S ⟶ transfert de **U** dans **S**

Code (**1F 34**)

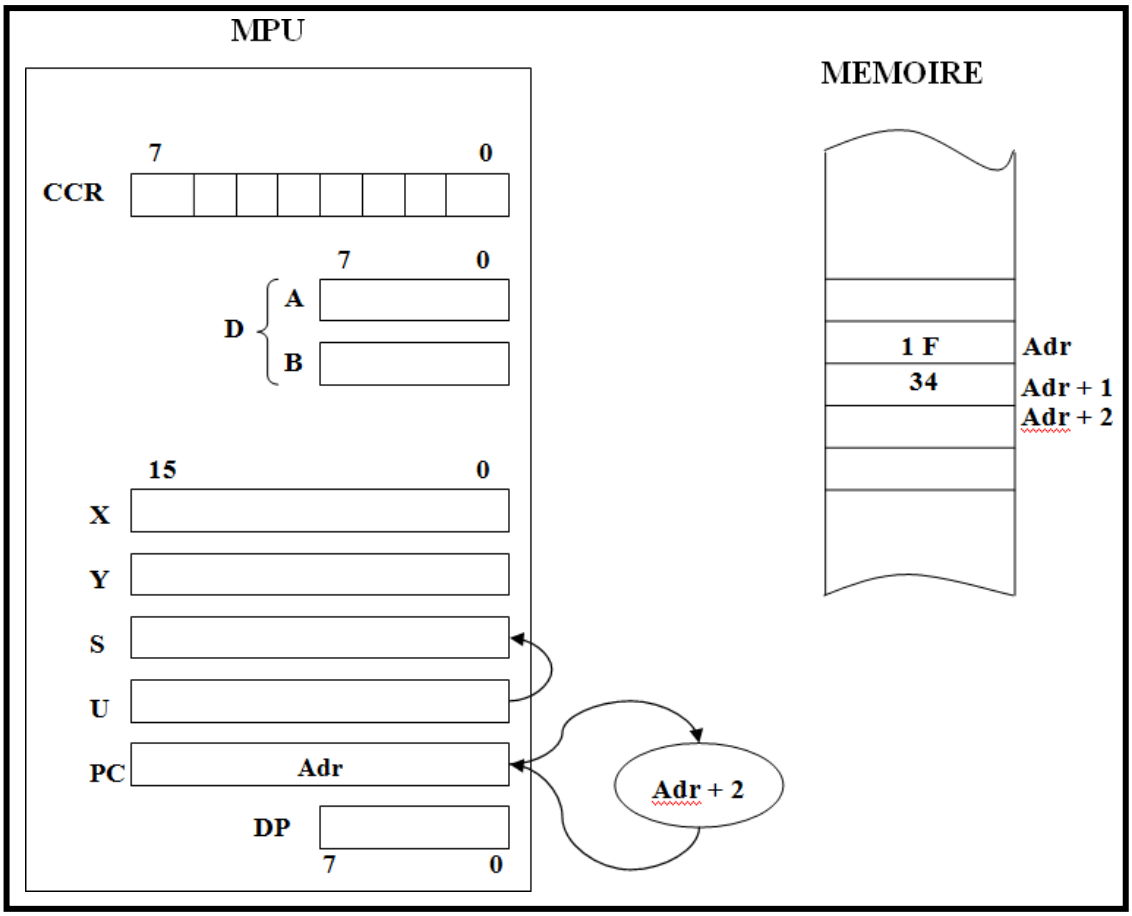


FIG 02 : *Transfert de* ***U*** *dans* ***S***

Instruction d'accès aux piles : le **premier octet** détermine le code opératoire (**OP Code**) pur, le **post – octet** (le 2eme octet) les **registre concernés** par l'accès à la pile.

Exemple :

PSHS A, B, X, ⟶ Sauvegarde dans la pile de **A, B, X**

Code (**34 16**)

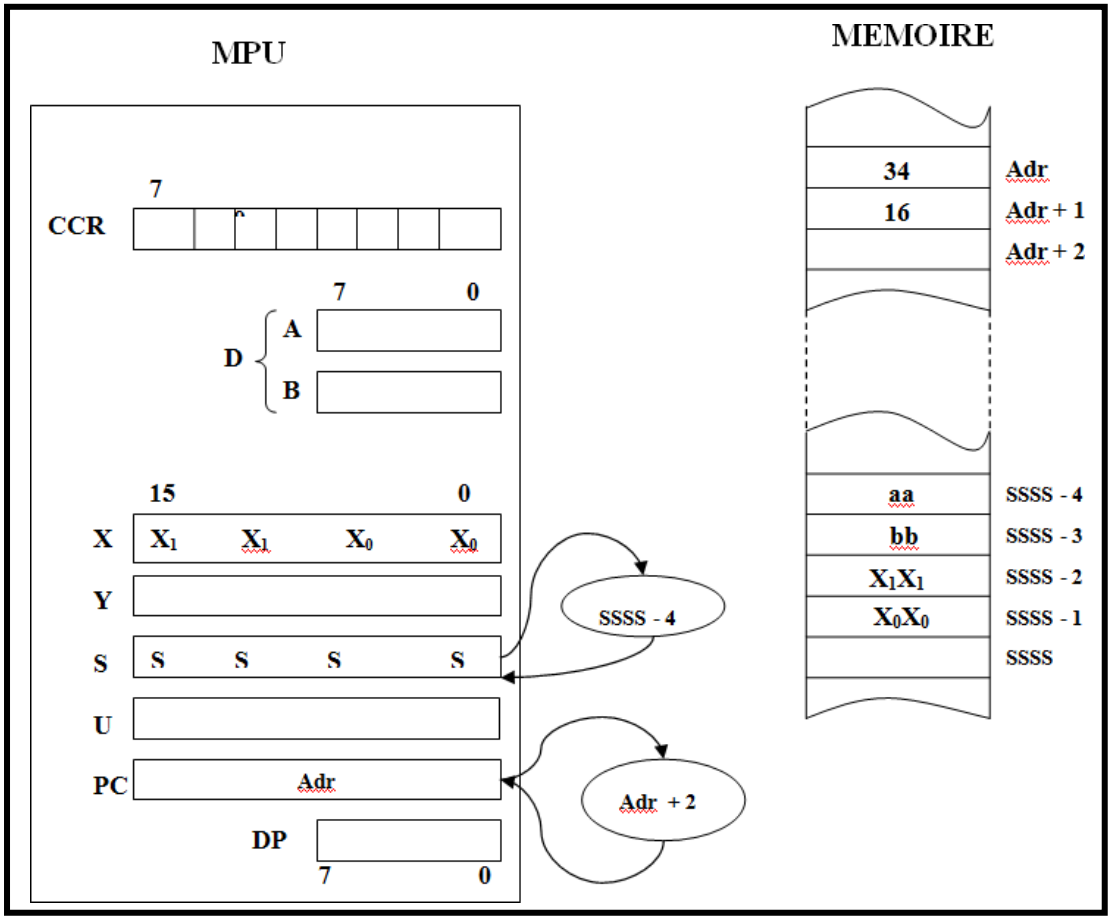


FIG 03 : *Sauvegarde dans la pile de* ***A, B, X***

Attende d'interruption : le **premier octet** est associé à l'instruction **CWAY, le second** sert à **masquer ou à valider** les interruptions.

Exemple :

CWAY # $ FF (attente d'interruption) *NMI* *IRQ* Et *FIRQ* sont masquées

Code (**3C** FF)

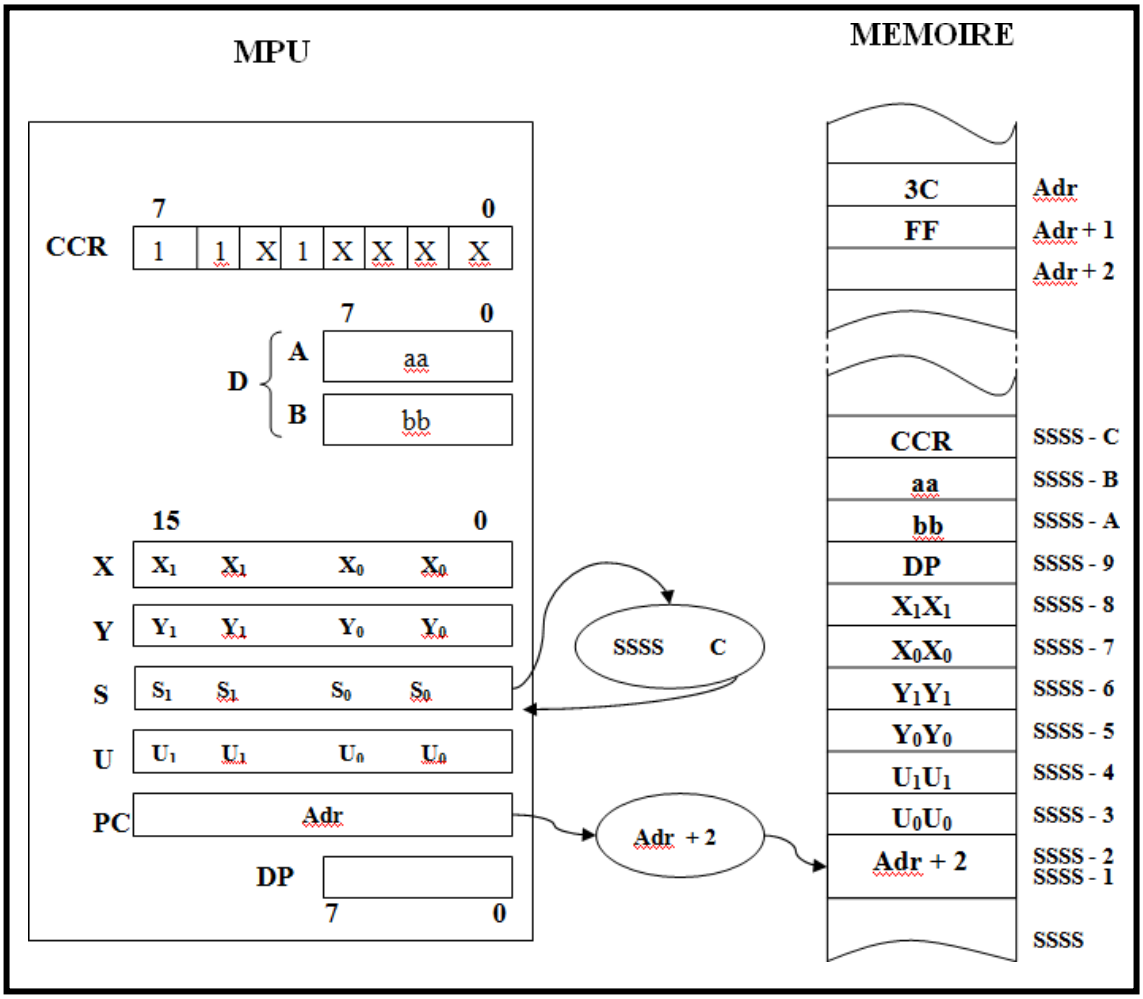


FIG 04 : *Exécution d'une attente d'interruption*

2.2 Mode d'adressage immédiat :

Dans ce mode d'adressage, le **code opératoire 8 bits** est suivi d'une valeur qui est l'opérande de l'instruction

La longueur de l'opérande (**1 ou 2 octet**) est fonction de la nature du support d'information concerne (indiqué par le code opératoire --registres 8 ou 16 bits)

Ce type d'adressage permet de charger les registres internes du microprocesseur avec la valeur de l'opérande.

Le symbole « # » signifie immédiat dans la syntaxe assembleur. Il existe **trois types** d'instructions dans ce mode d'adressage.

3. **Instruction sur deux octets :** Le premier octet contient le code opératoire, le second la constance 8 bits **.**Ce type d'instruction est réservé pour travailler sur les registres 8 bits du microprocesseur.

Exemple :

LDA # $43 (chargé la valeur **$ 43** dans l'accumulateur A)

(**$:** Hexadécimale.)

(**# :** Immédiat.)

Code (**86** 43)

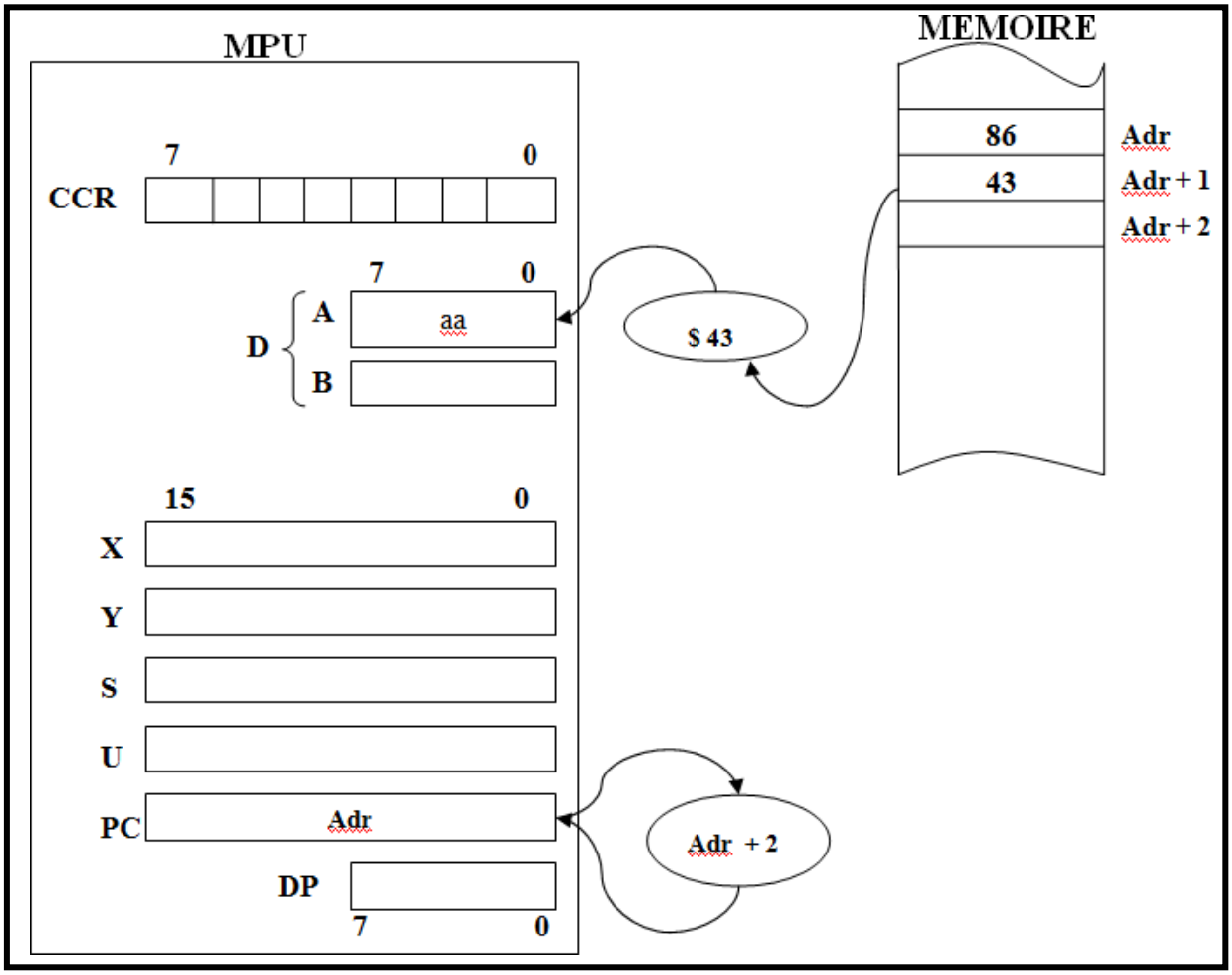


FIG 05 : *Chargement de la valeur* ***$ 43*** *dans l'accumulateur A*

4. **Instruction sur trois octets :** Le premier octet contient le code opération, le second et le troisième contiennent la constante 16bits. Ce type d'instruction est réservé pour travailler sur les registres de 16 bits du µP

Exemple :

ADDD # $ 1981 (addition du contenu de l'accumulateur **D** et de **$ 1981** le résultat dans **D**)

Code (**C3** 19 81

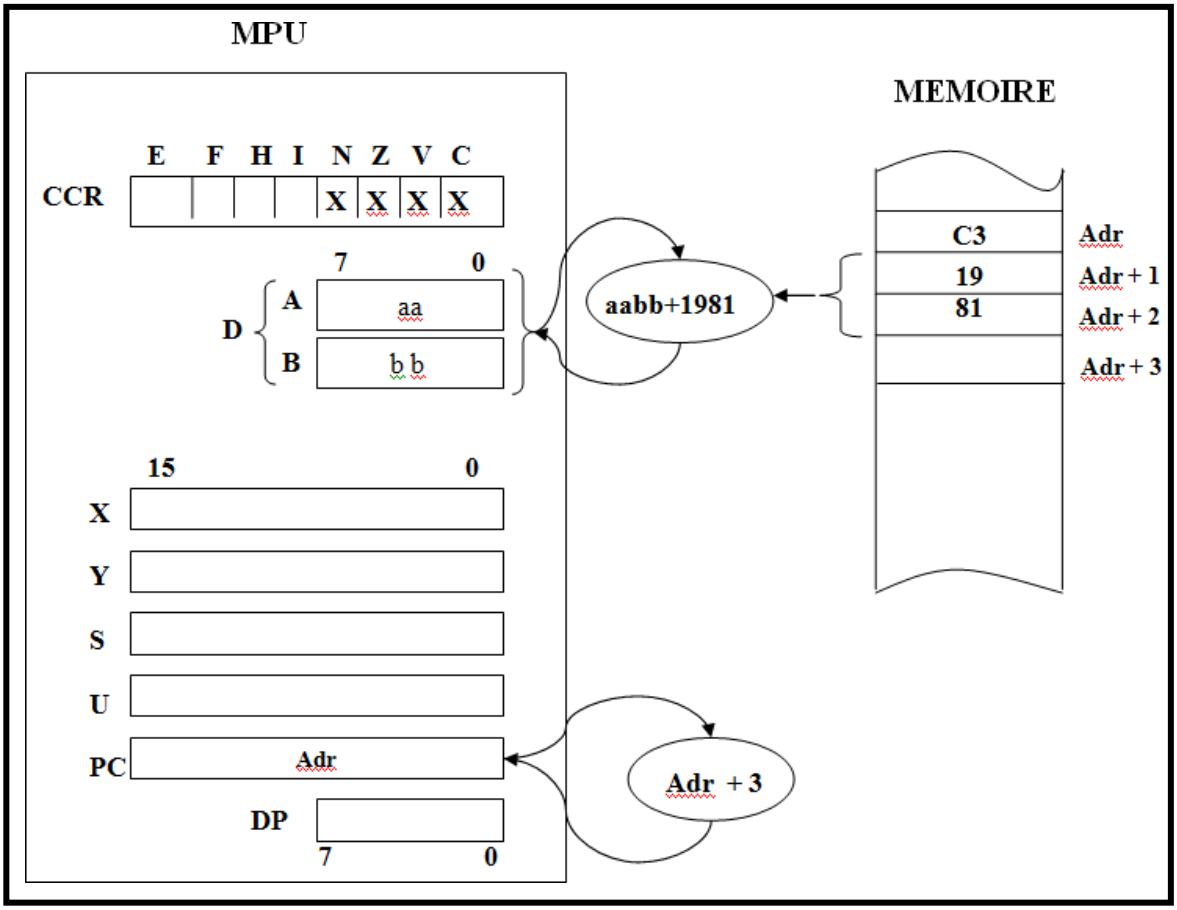


FIG 06 : *Exécution de l'addition du contenu de l'accumulateur* ***D*** *et de la valeur* ***$ 1981***

5. **Instruction sur quatre octets :** Dans ce type d'instruction, le **code opératoire** utilise **deux octets** mémoire, la **constante également**. Le premier octet est nécessaire pour les instructions **CMPD, CMPS, CMPU, CMPY, LDS, LDY, STY, STS**

Example:

LDY # $ 1623 (charger l'index **Y** avec la valeur hexadécimale **$1623**)

Code (**10 8C** 16 23

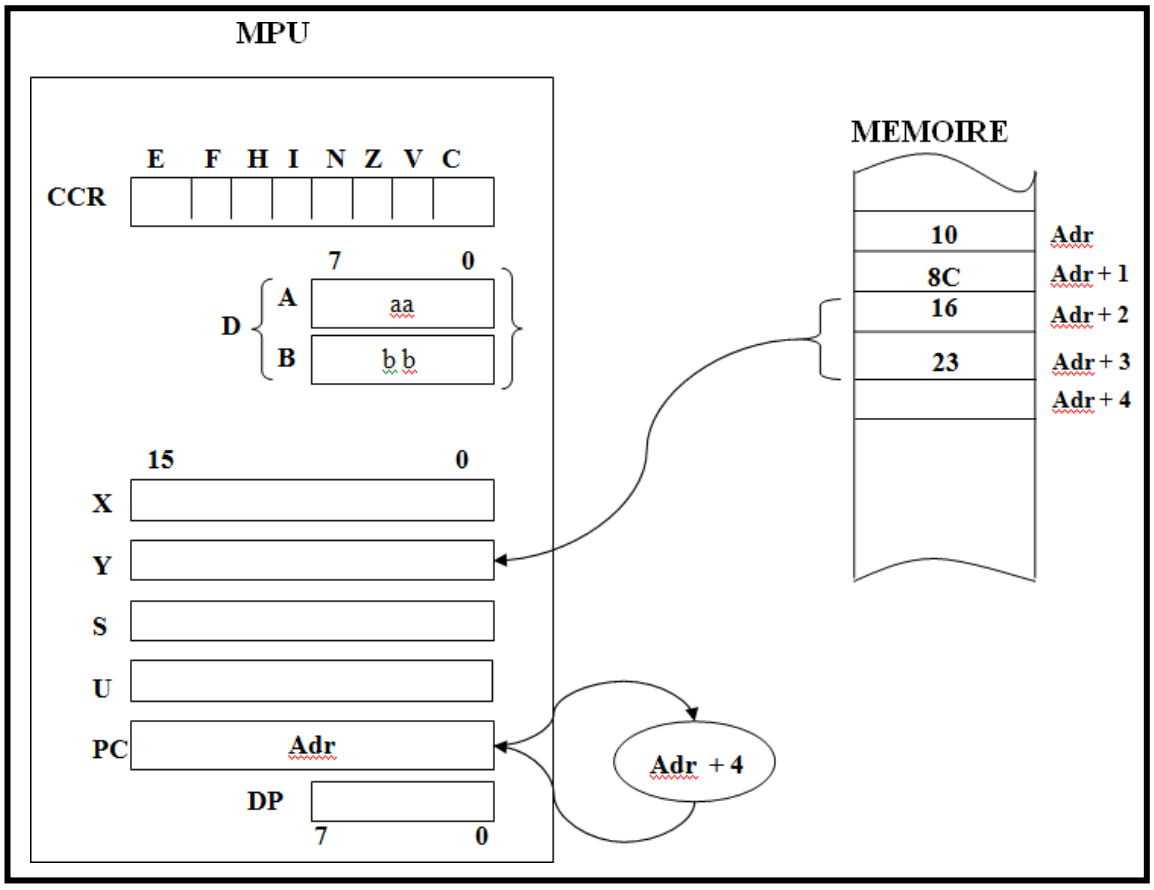


FIG 07 : Chargement l'index **Y** avec la valeur hexadécimale **$1623**

REMARQUE : l'instruction immédiat est valable pour tous les registres internes du MPU sauf le registre **DPR**. Toutefois, il est possible en deux temps de charger **DRP**

Exemple :

LDA # $ 03 (Chargement de l'acc **A** avec **$ 03**)

EXGA ,DP (Echange des contenus de **A** et **DP**)

2.3 Mode d'adressage direct

Ce mode présente l'avantage de ne nécessiter que 2 octets pour avoir accès a des données situées sur l'ensemble de l'espace mémoire du µP.

Le premier octet définit le code opération, le second représente les 8 bits de poids faible de l'adresse effectif (réelle) dont les 8 bits de poids fort se trouvent dans le registre de page du µP (**DPR**).

Il suffit donc d'initialiser le registre de page **DPR** pour pouvoir travailler en adressage directe sur les 256 octets de la page choisie ; au –delà, il faut de nouveau accéder au **DRP.**

A la mise sous tension, le registre de page est mis a zéro. On aura donc accès aux 256 octets de la page 0. Le symbole « < » est une directive assembleur qui force l'adressage directe.

Il existe deux types d'instructions dans ce mode d'adressage

Instruction sur deux octets : Le premier octet définit le code opératoire. Le second les poids faibles de l'adresse affective.

Exemple :

LDA $37 ou **LDA < $37** (Chargement de l'acc **A** avec le contenu de **$D437**) (**DRP=$D4**)

Code (**96** 37)

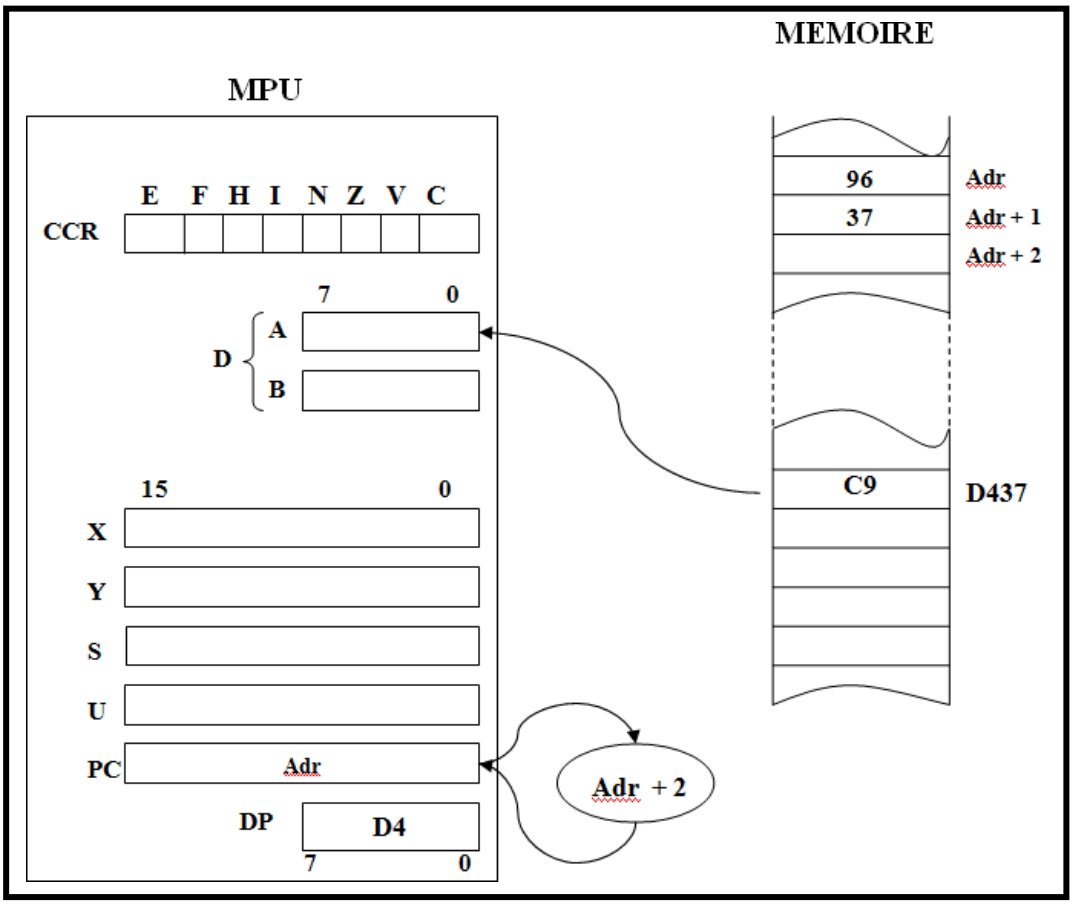


FIG 08 : *Chargement de l'acc* ***A*** *avec le contenu de* ***$D437***

Instruction sur trois octets :

Le code opératoire est définit par les deux premiers octets. Le troisième octet précise les poids faibles de l'adressage effectif. Le premier octet est seulement nécessaire pour les instructions qui opèrent sur le pointeur de pile **S**, le registre d'index **Y** et sur les instructions de comparaison **CMP** opérant sur 16 bits.

Exemple :

LDY $ 17 ou **LDY < $17** (Chargement du registre **Y** avec le contenu de **$ 2317/18**) (**DRP=$23**)

Code (**10** 9E 17)

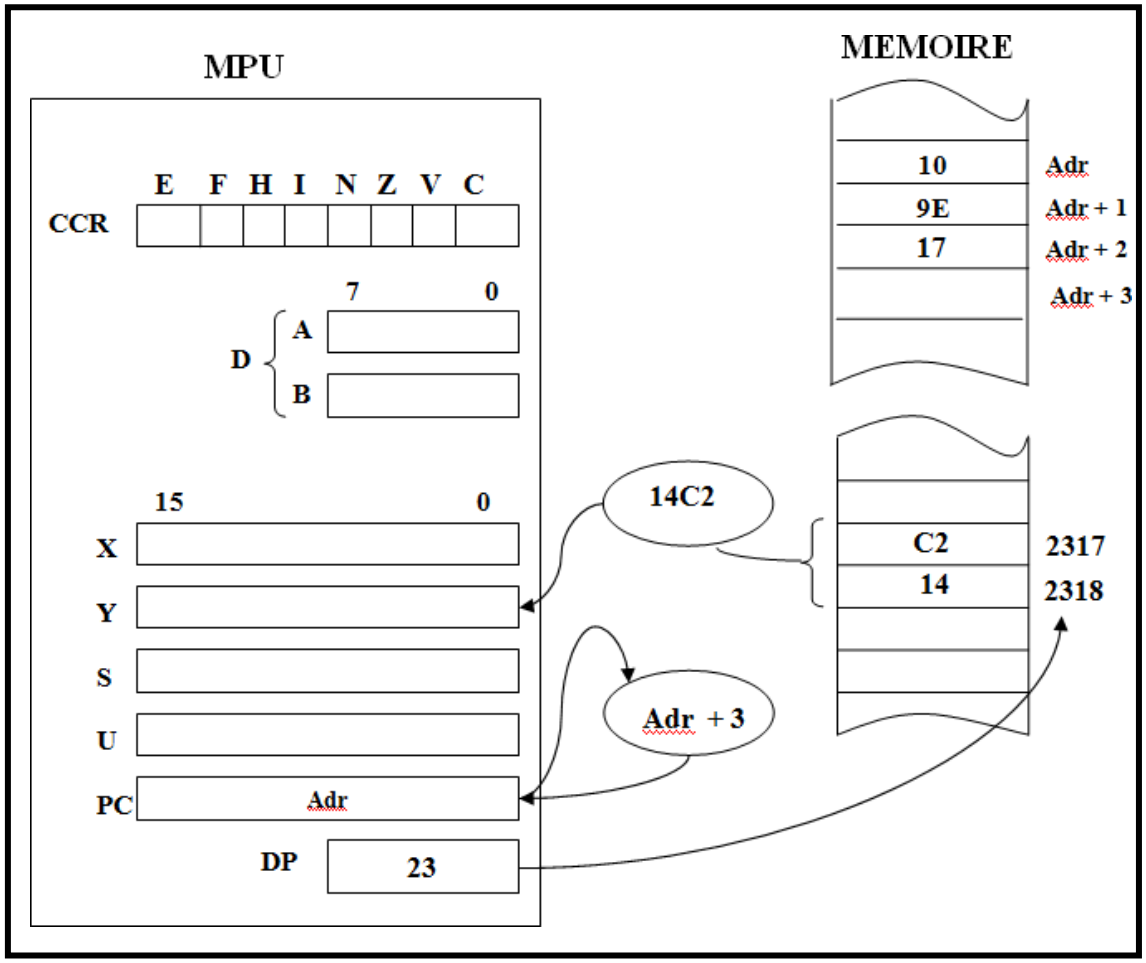


FIG 09 :** Chargement du registre **Y** avec le contenu de **$ 2317/18

2.4 Adressage étendu

L'adressage étendu ou absolu permet de retrouver des données en mémoire. Le code opératoire (8bits) est suivi de 2 octets constituant l'adresse effective de l'opérande proprement dit.

Il existe deux types d'instructions dans ce mode d'adressage.

Le symbole « > » est une directive assembleur qui force l'adressage étendu.

Instruction sur trois octets : Le premier octet (code opératoire « op-code ») est suivi de l'adressage16 bits spécifiant l'emplacement de l'opérande (8 ou16bits) .

Exemple :

LDA $21CO ou **LDA >$21C0** (chargement de l'accumulateur **A** avec le contenu de l'adresse $21C0) (Code **B6** 21C0)

Dans ce cas l'opérande est l'accumulateur **A** (8 bits).

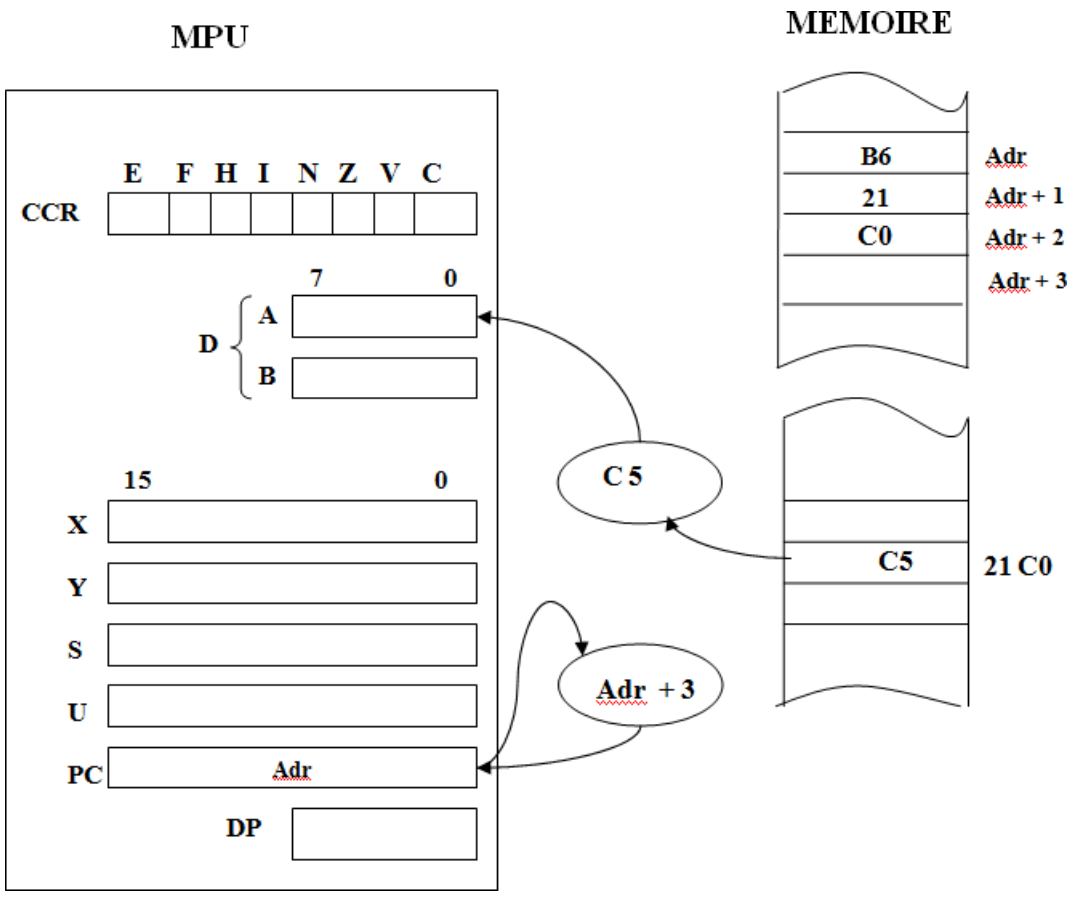


FIG 10 : *chargement de l'accumulateur* ***A*** *avec le contenu de l'adresse $21C0*

Instruction sur quatre octets : Le **code opératoire** est défini par les deux premiers octets.

Les **3° et 4e octets** constituent l'adressage effectif. **Le premier octet** est seulement nécessaire pour les instructions qui opèrent sur les **pointeurs S et Y** et sur les instructions de comparaison **CMPU. CMPD.**

Exemple :

LDY $ 347B ou **LDY > $ 347B** (Chargement du registre **Y** avec le contenu de **$ 347B**).

(Code **10BE** 347B)

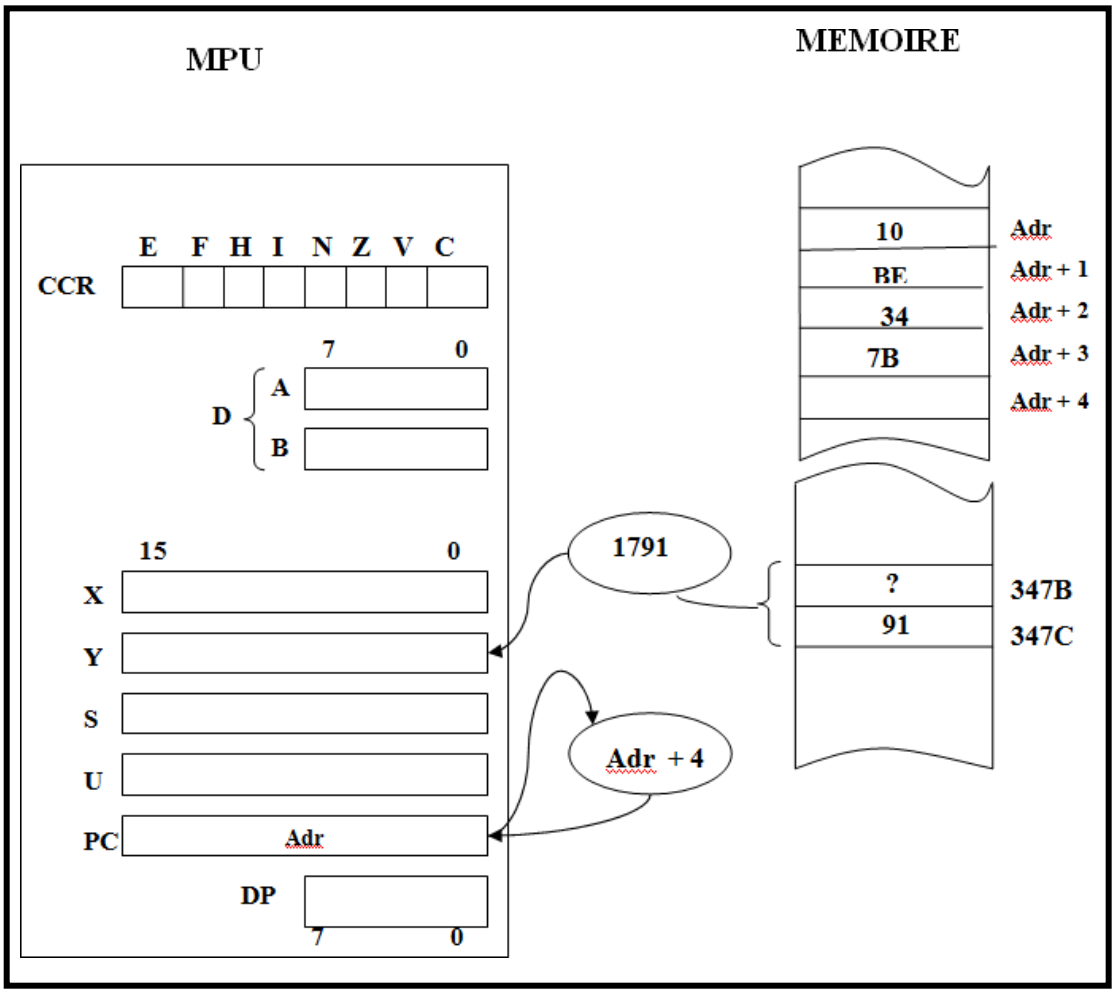


FIG 11 :** Chargement du registre **Y** avec le contenu de **$ 347B

2.5 Adressage étendu indirect

Ce mode d'adressage est identique au mode d'adressage étendu mais il possède en plus une indirection. En effet, on accède à une adresse effective en « transitant »par une adresse intermédiaire. L'adressage étendu indirect est codé avec un code opératoire identique à celui de l'adressage indexé suivi d'un post-octet (cas particulier de cet adressage indexé).

La notion assembleur « **[ ]** » force l'adressage étendu indirect.

Il existe deux types d'instruction dans ce mode d'adressage.

Instruction sur quatre octets : Les deux premiers octets déterminent le code opératoire ; codeopératoire de l'adressage étendu simple suivi d'un post-octet d déterminant l'indirection.

Les 3° et 4° octets représentent l'adresse de transit.

Exemple :

LDA [$ 2000] (Chargement de l'accumulateur **A** avec le contenu de l'adresse se trouve en **$ 2000 $ 2001**).

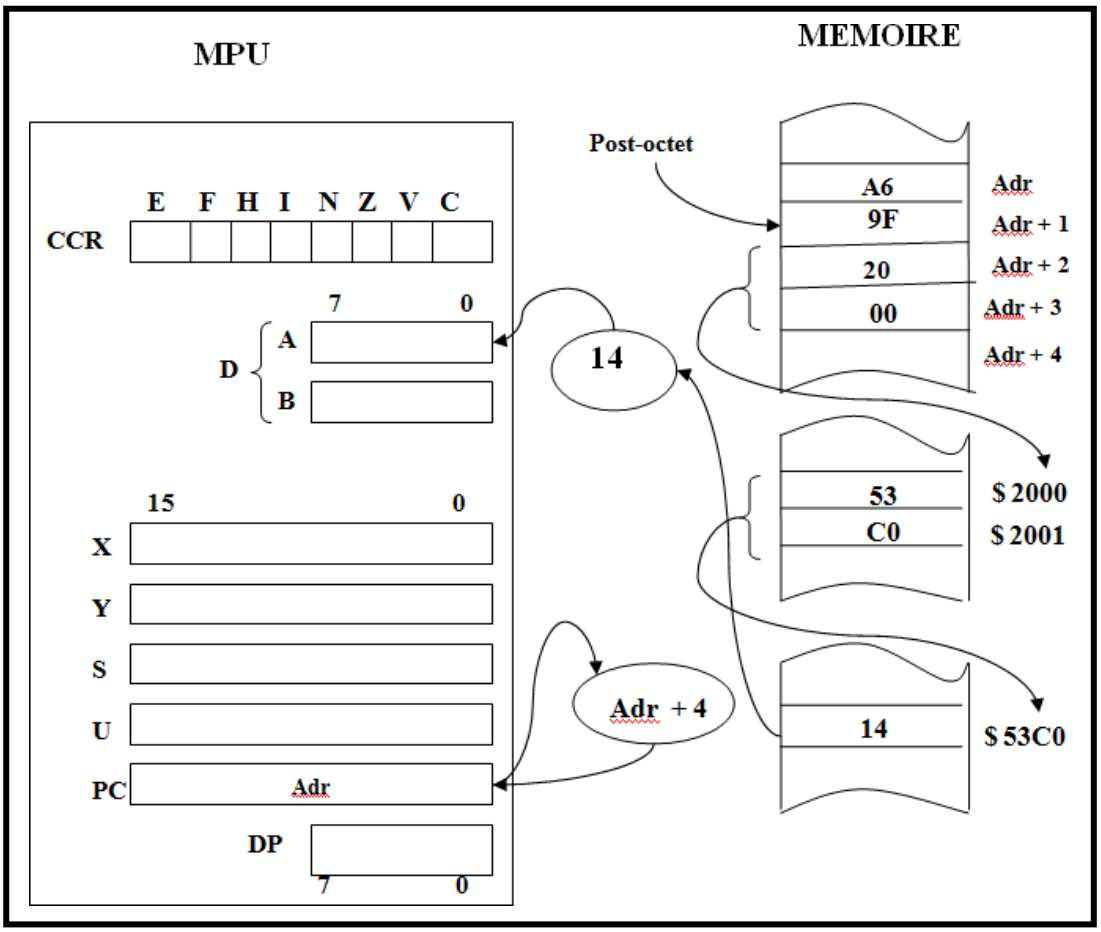


FIG 12 : *Chargement de l'accumulateur* ***A*** *avec le contenu de l'adresse se trouve en* ***$ 2000 $ 2001***

Instruction sur cinq octets : Pour certaines instructions, il est nécessaire d'ajouter un pré-octet.

Celaest nécessaire pour les instructions opérant sur les pointeurs **S** et **Y** et pour les instructions de comparaison **CMPU** et **CMPD.**

Exemple :

LDY [$2000] : Chargement du registre **Y** avec le contenu dont l'adresse se trouve dans l'adresse $2000, $2001

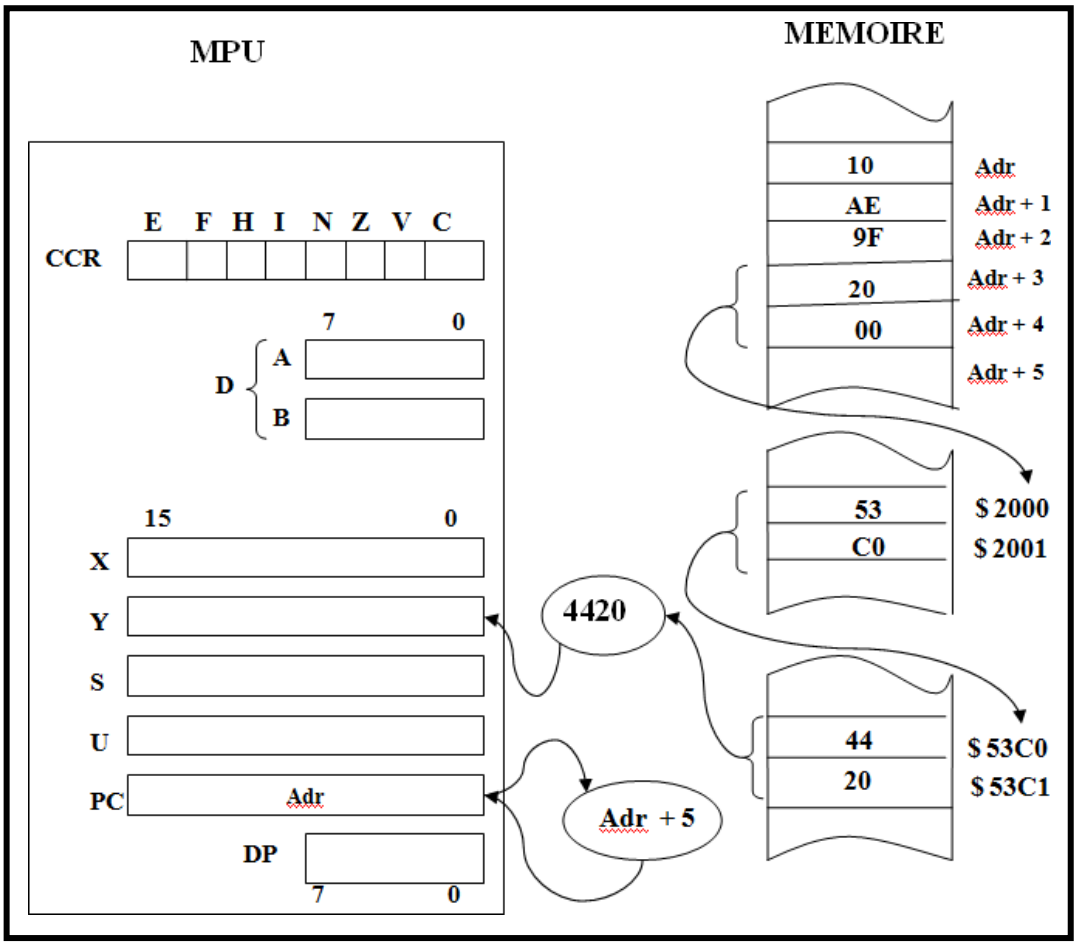


FIG 13 : *Chargement du registre* ***Y*** *avec le contenu dont l'adresse se trouve dans l'adresse $2000, $2001*

2.6 Adressage relatif

2.6.1 Adressage relatif court

Ce mode d'adressage est réservé pour les instructions de branchement. L'instruction de branchement opérant en adressage relatif court fait appel à un format de 2 Octets seulement. Le **premier octet** détermine le **OP Code** qui spécifie le **type de branchement** en même temps que le **test correspondant**. Le **second octet** est la **valeur du déplacement** qui peut être positif ou négatif. Afin de bien saisir le fonctionnement de ce mode d'adressage, le mieux est de donner un exemple.

Exemple 01:

```
ETIQ  EQU  $2000
  .
  .
  .
CMPA # $42
BCS ETIQ
LDA #  $03
  .
  .
  .
  .
ETIQ   LDB #  $F0
```

⟹ Effectue un branchement à l'adresse **ETIQ** si l'opération précédente entraîne une retenue (**C=1**)

En mode d'adressage relatif court, le déplacement étant code sur **un octet**. On pourra se déplacer de **$(-2^{n-1})=(128)_{10} = ($80)$** octets en arrière et de **$+ (2^{n-1}-1) = (127)_{10}= ($7F)$** octets en avant par rapport à la valeur du compteur ordinal (PC) à la fin du traitement de l'instruction de branchement **(+$2)**

Exemple 02 :

```
ARRET EQU $3000
BPL  ARRET
```

L'étiquette **ARRET** correspond à l'adresse **$3000.** L'instruction **BPL ARRET** doit se trouver impérativement entre **$2F7F** [$2F7F = $3000-($2 + $7F)] **et $307E** [$307E = $3000-($2 - $80)**]**
Dans ce cas, **BPL** fait un test sur le bit **N** du **CCR** le branchement à l'adresse **$3000** A lieu si **N=0** (résultat de l'opération précédente positif)

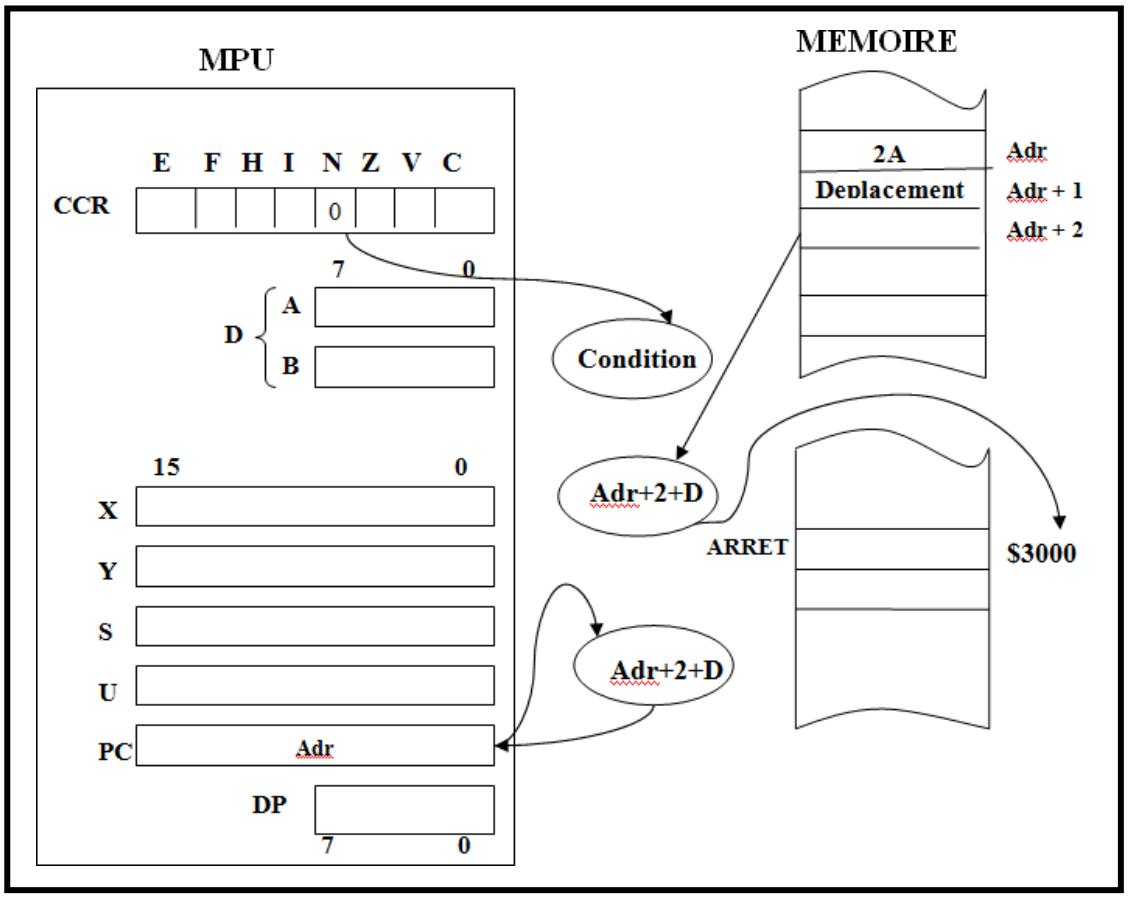


FIG 14 : *Exécution de l'instruction BPL ARRET (Branchement à l'étiquette ARRET)*

Remarque : Le déplacement est calculé en fonction de la valeur de adresse **Adr** par rapport à l'étiquette **ARRET** ($3000)

D = Adr saut – (Adr+2)

Exemple 03 :

```
        ADDA $53F8
        BEQ  FIN
             .
             .
             .
FIN     SWI
```

On effectue l'addition du contenu de la case mémoire d'adresse $53F8 avec l'accumulateur. L'instruction BEQ veut dire « branchement si égalité ». Ici il ya donc branchement vers FIN si le contenu de l'accumulateur après addition est égale à zéro. Le bit Z du CCR est alors égal à 1. Si non, le programme continue et exécute les instructions présentes après le BEQ.

2.6.2 Adressage relatif long

Ce mode d'adressage est identique au précédent, il est toujours réservé aux branchements. Les instructions sont codées sur **quatre octets**, les **deux premiers déterminent le code opération**, les **3e et 4e octets** donnent **la valeur signée du déplacement**.

Dans ce mode d'adressage le déplacement est codé sur **16 bits** d. On pourra donc se déplacer de **+ (2^{n-1}-1) =** + 32 767 ($ 7 FFF) octets en avant et de **(-2^{n-1})=** – 32 768 ($8000) octets en arrière, par rapport à la valeur du compteur ordinal (PC) à la fin du traitement de l'instruction de branchement. **(+$4)**

Au niveau du code machine, ce qui différencie l'adressage relatif long de l'adressage relatif court, c'est la présence d'un pré octet dont la valeur est constante **($ 10)**.le OP Code proprement dit est le même dans les deux cas.

Exemple :

ARRÊT EOU $ 6ØØØ

LBPL ARRÊT

L'étiquette ARRÊT correspond à 6000 en hexadécimal. Dans ce cas il n'y a pas de **restriction** (limitation) quant à la position de l'instruction **LBPL** sur l'espace mémoire du microprocesseur. Si le résultat de l'opération précédent l'instruction de branchement est positive, le compteur ordinal se positionnera à l'adressage $ 6000.

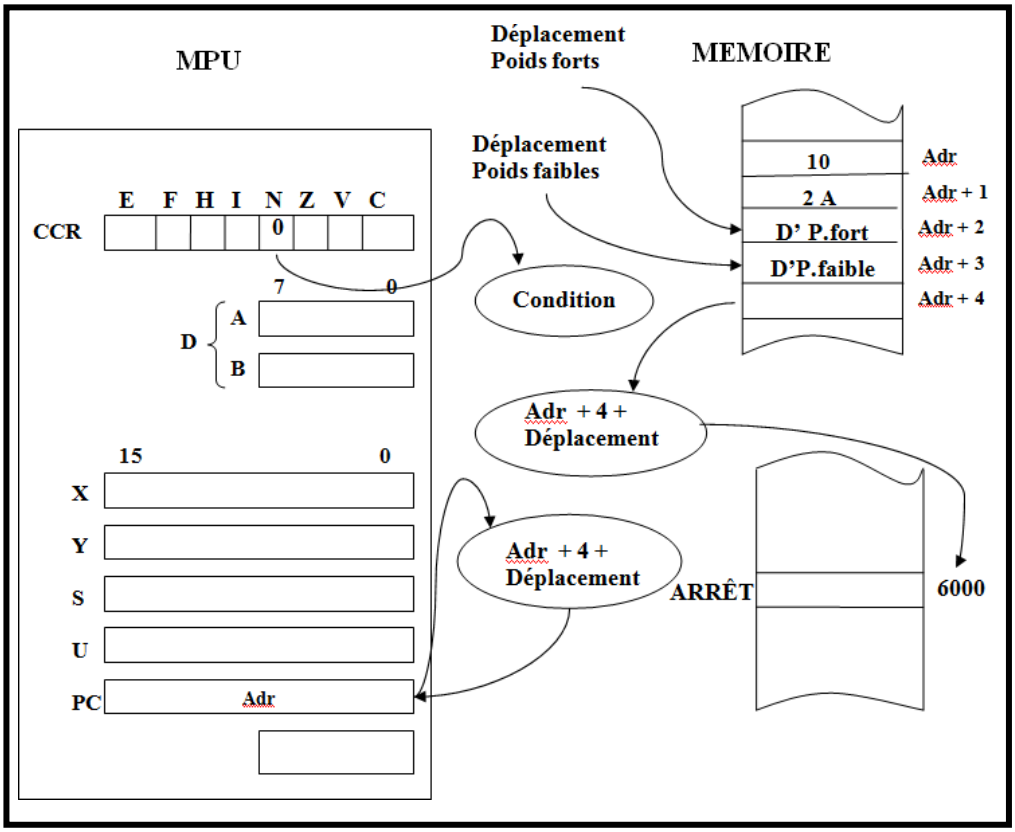


FIG 15 : *Exemple d'un Mode d'adressage relatif long*

2.7 Adressage indexé

Le principe de l'adressage indexé est que l'instruction spécifie une base (index) plus un déplacement par rapport à cette base. On peut donc écrire.

Adresse effective = base + déplacement

La puissance d'un mode d'adressage indexé est déterminée par le choix des **bases** dont on dispose et par toutes les possibilités de **déplacement** que l'on a.

Dans le cas du 6809, **la base** peut être soit un des deux **registre d'index** (ce qui est normal), mais aussi un des deux pointeurs de pile (**U ou S**) ou, ce qui est très intéressant, le **compteur programme** lui-même.

Pour ce qui est du **déplacement** il y'a de multiples possibilités, celui-ci peut être **nul**, codé sur **cinq**, **huit** ou **seize bits**, ou **variable** dans le cas de l'utilisation d'un accumulateur **A, B ou D.**

Enfin, l'adressage indexé offre des possibilités **d'auto- incrémentation** ou **décrémentation** de **1** ou de **2.**

Toutes ces options sont sélectionnées par le **post-octet** qui suit le **code opératoire**.

Le tableau 1 ci-après montre les formats utilisé par le post-octet.

Bit du registre post octet								**Mode d'adressage indexé**
7	6	5	4	3	2	1	0	AE = Base (R) + Déplacement
0	R		DEPLACEMENT					AE = R ± 4 bits
1	R		0	0	0	0	0	AE = R +
1	R		1	0	0	0	1	AE = R $^{++}$
1	R		0	0	0	1	0	AE = - R
1	R		1	0	0	1	1	AE = - - R
1	R		1	0	1	0	0	AE = R ± 0
1	R		1	0	1	0	1	AE = R ± Acc B
1	R		1	0	1	1	0	AE = R ± Acc A
1	R		1	1	0	0	0	AE = R ± 8 bits
1	R		1	1	0	0	1	AE = R ± 16 bits
1	R		1	1	0	1	1	AE = R ± D (Acc A + Acc B)
1			1	1	1	0	0	AE = PC ± 7 bits
1			1	1	1	0	1	AE = R ± 15 bits
1	R		1	1	1	1	1	AE = Adresse

REMARQUE : Les bits 5 et 6 du post-octet permettent de définir la base.

Base R	b6	b5
Index X	0	0
Index Y	0	1
Pointeur U	1	0
Pointeur S	1	1
Compteur programme		

Indifférent, la sélection de la base PC se fait à l'aide des bits 2 et 3 (1, 1).

Le bit 7 définit le rôle du bit 4

b7 = 0 ⟶ b4 = bit de signe

b7 = 1 ⟶ b4 = choix du mode direct (b4 = o) ou indirect (b4 = 1)

Si b4 =1, le mode peut travailler en direct ou en indirect.

Les bits 0 à 3 définissent le champ du mode d'adressage.

Afin de mieux cerner toutes les possibilités du mode d'adressage indexé, nous allons voir maintenant toutes les combinaisons possibles.

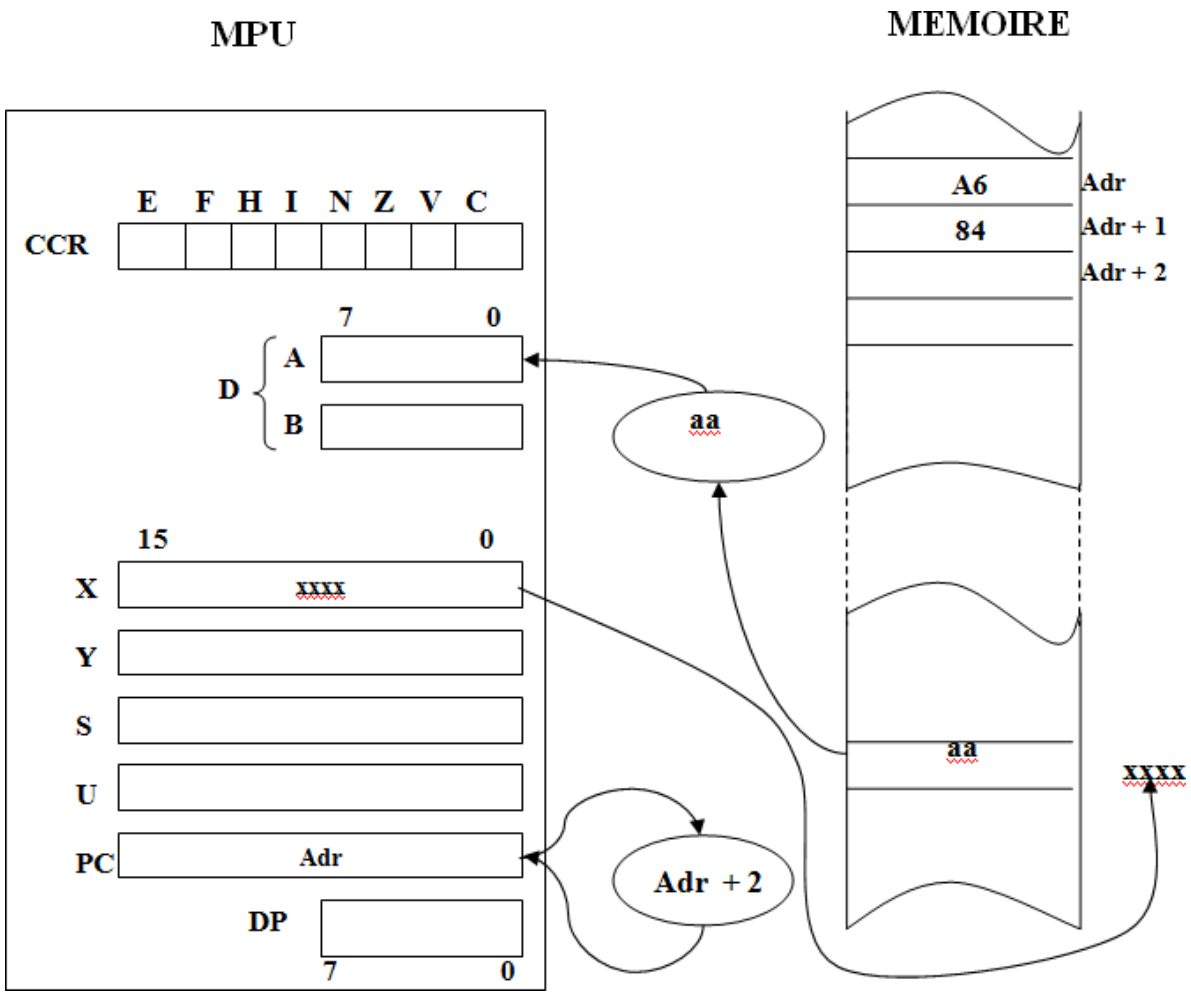
2.7.1Adressage indexé avec déplacement nul

Dans ce mode, le registre pointeur sélectionné contient l'adresse effective des données devant être utilisées par l'instruction. Ce mode est le mode indexé le plus rapide. Il existe deux types d'utilisation :

Instruction sur deux octets : le code opératoire est suivi du post-octet précisant les options choisies pourl'instruction en cours (tableau 1).

Exemple :

LDA, X chargement de A avec la valeur dont l'adresse est le contenu de l'index X



Dans ce cas le post-octet est égal à $ 84

b7 = 1 ⟶ b4 = 0 signifie que nous sommes en adressage indexé direct.

b5.b6 = 0.0 ⟶ on va travailler avec l'index X.

b3.b2.b1.b0 = 0.1.0.0 ⟶ le déplacement est nul.

Instruction sur trois octets : pour certains registres du microprocesseur il est nécessaire d'ajouter un pré-octet, aux deux octets opératoires proprement dits.

Cela mis à part, ce mode est identique.

Exemple :

LDY, X : chargement de l'index Y avec la valeur dont l'adressage est le contenu de **X**

2.7.1 Adressage indexé avec auto incrémentation/décrémentation

Dans ce type d'adressage le contenue du registre d'index est décrémenté avant de pointer l'adresse effective ou incrémenté après l'avoir fait.

Le registre sélectionné contient l'adresse effective des données utilisées par l'instruction.

Les symboles **« + », « ++ », « -», « - - »** placés avant ou après la base définissent le type Auto-incrémentation /décrémentation choisi.

Exemple : Si X contient $1000

LDA, X+ : Charge A avec le contenu de l'adresse $1000, mais X contiendra $1001 après l'instruction terminée

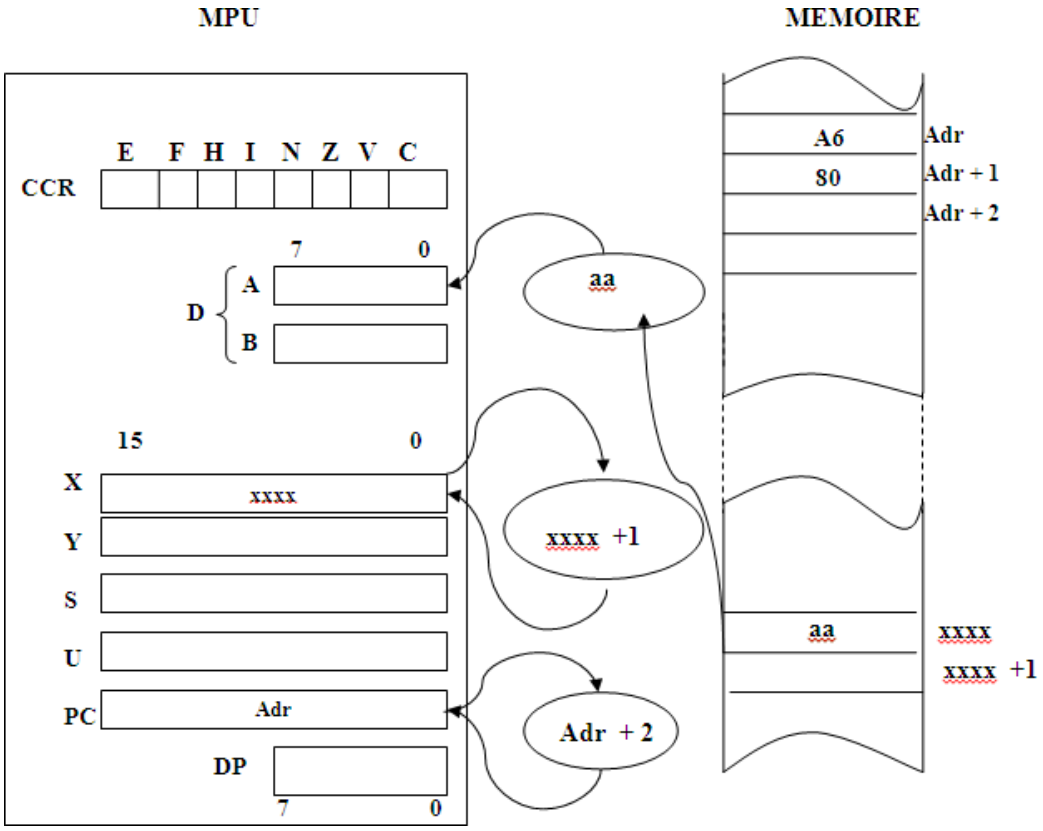
LDB,-X : Commence par décrémenter X. donc X= $0FFF ensuite charge B avec le contenue de $0FFF

LDD,X++ : Charge D avec les contenus de $1000 et $1001 ensuite incrémente deux fois X après . donc X=$1002

2.7.1.1 Instruction sur deux octets :

Exemple :

LDA, X^+ : Chargement de **A** avec la valeur dont l'adresse est le contenu de **X**, poste- incrémentation par un de **X (X=X+1)**



Dans ce cas le post-octet est égal à $ 80 (Tableau 1) :

b7 = 1 b4 = 0 adressage indexé direct.

b6.b5 = 0.0 utilisateur de l'index X.

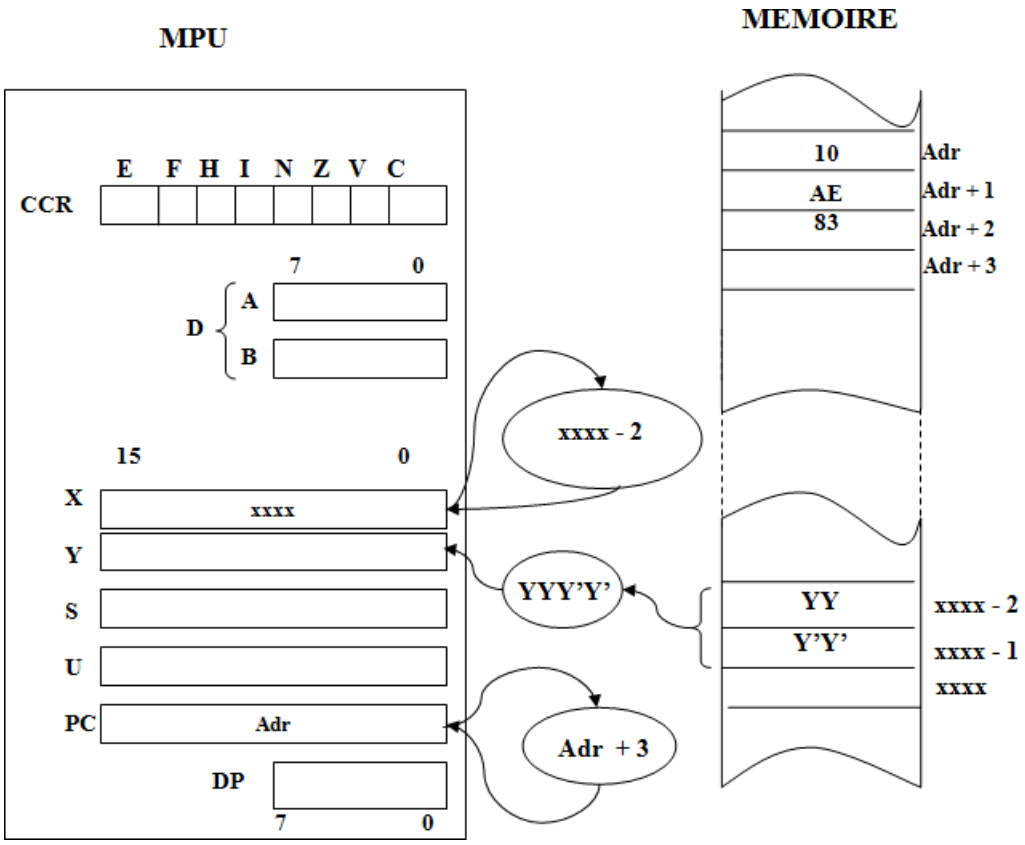
b3.b2.b1.b0 = 0.0.0.0 mode post-incrémentation.

Le mode auto- incrémentation /décrémentation par un est utilisé pour gérer des tables de données

Instruction sur trois octets :

Exemple :

LDY, - -X : Chargement de **Y** avec la valeur dont l'adresse de base est le contenu de **X**- 2 (décrémenté X de 2 (X=X-2) ensuite charger Y par l'adresse contenue dans X est qui est X-2)



Le post-octet prend la valeur $83 :

- b7 = 1 b4 = 0 adressage indexé direct.
- b6.b5 = 0.0 utilisateur de l'index X.
- b3.b2.b1.b0 = 0.0.1.1 double pré-décrémentation.

Le mode auto- incrémentation /décrémentation par deux est utilisé pour gérer des tables d'adresse.

L'aspect pré décrément, post-incrément permet de créer des piles logicielles avec **X** et **Y**, gérées de la même façon que les piles **S** et **U**

2.7.2 Adressage indexé avec déplacement constant

Dans ce mode d'adressage, l'adresse effective de l'opérande est la somme du déplacement (en complément a deux) et du contenu du registre constituant la base. Le registre de base n'est pas modifié. Il existe 3 formes d'adressage indexé a déplacement constant, suivant la valeur de cette constante

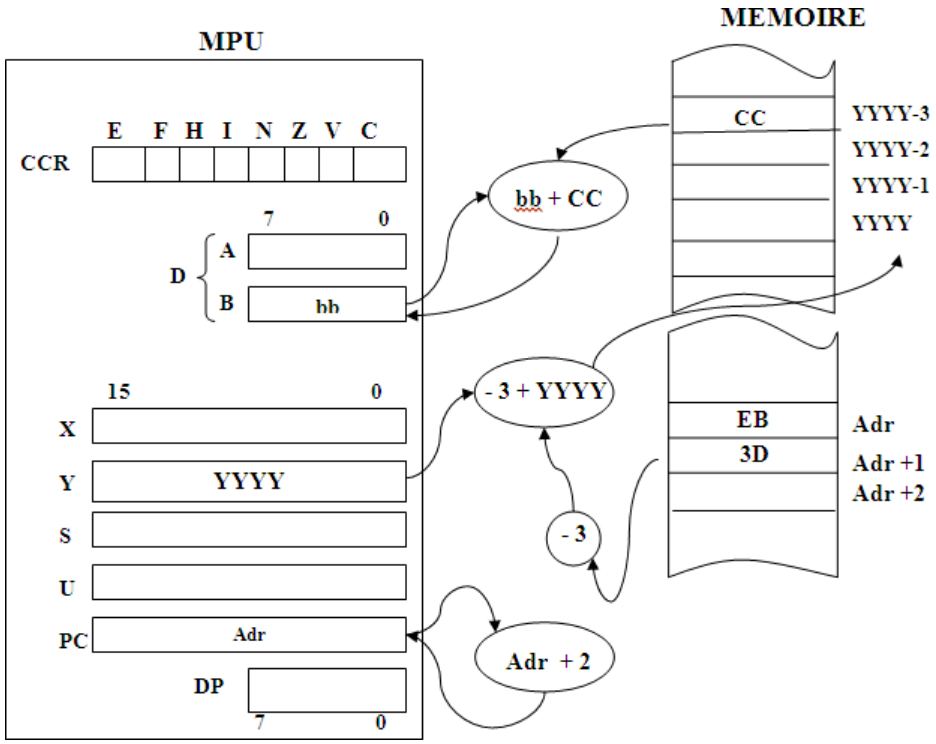
Déplacement sur +/- 4bits :

Ce déplacement codé sur cinq bits (en complément a deux) présente l'avantage d'être contenu dans le post-octet d'indexation. Ce qui permet un gain de place mémoire et une exécution plus rapide de cette instruction.

Les bits **0** à **4** du **post-octet** déterminent donc la valeur du déplacement qui pourra être de -16octets arrière (10000) et de +15 octets en avant (01111). Dans ce cas, le bits 7 constamment a zéro initialise le bit 4 comme bit de signe.

Exemple :

ADDB – 3, Y **:** Addition du contenu d'adresse mémoire **Y-3** au contenu de l'accumulateur **B**, le résultat est dans **B**



Le post-octet prend la valeur $3D (tableau 1) :

b7 = 0 , le bit 4 est le bit de signe du déplacement.
b6.b5 = 0.1 le pointeur est l'index **Y**
b4.b3.b2.b1.b0 = 1.1.1.0.1 le déplacement est -3 en complément à 2

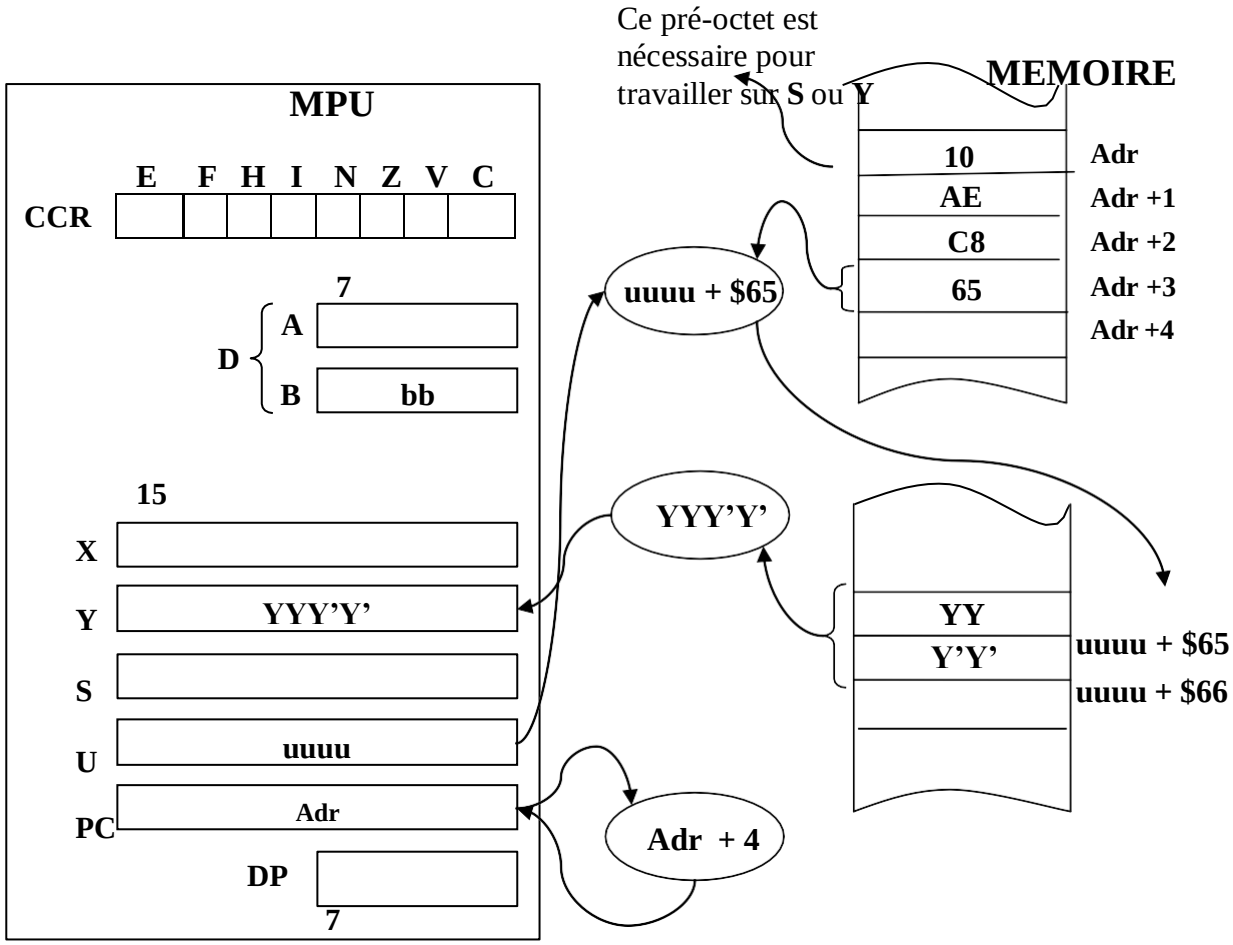
Déplacement sur +/- 7bits :

Ce déplacement codé sur 8bits (en complément a2) est contenu dans un seul octet, placé à la suite du code opératoire proprement dit et du post-octet.

Les déplacements possibles sont donc compris entre -128 et +127 octets.

Exemple :

LDY $65, U : chargement du pointeur **Y** avec le contenu mémoire dont l'adresse de base est le contenu de **U +$65.**



Le post-octet prend la valeur $C8.

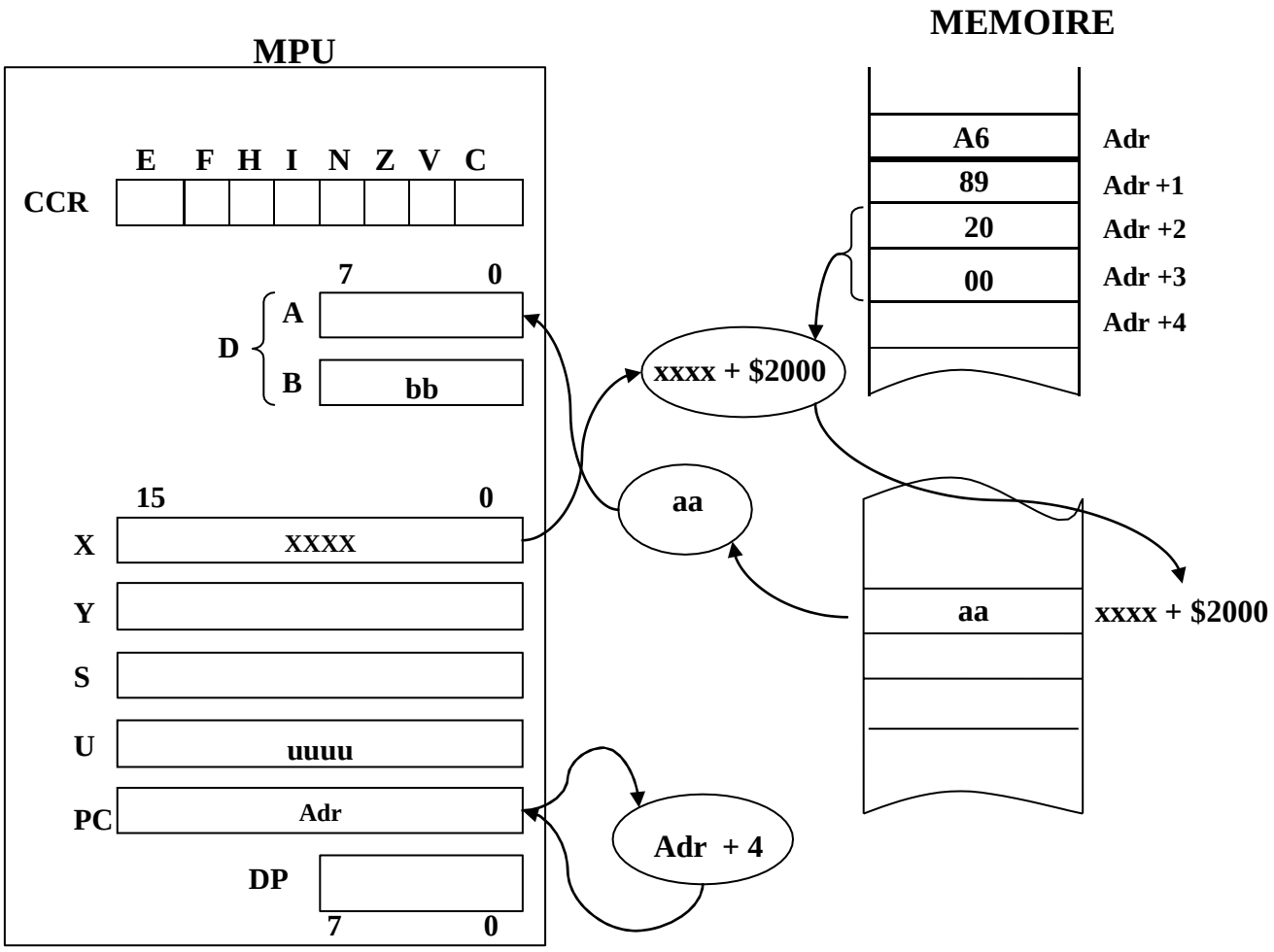
- b7 = 1 , bit 4 = 0 : adressage indexé en directe.
- b6.b5 = 1.0 : le pointeur et **U**
- b3.b2.b1.b0 = 1.0.0.0 : le déplacement est codé sur 8 bits en complément à 2

Déplacement sur +/- 15 bits :

Ce déplacement codé sur 16 bits (en complément a2) est contenu dans 2 octets placés a la suite de ceux de l'instruction (op-code + post-octet). Les déplacements possibles sont donc compris entre – 32768 et + 32767 octets.

Exemple :

LDA $2000, X **:** chargement de l'accumulateur **A**, avec le contenu mémoire d'adresse **X + $2000**



Valeur du post-octet : $89.

- b7 = 1, b4 = 0 : indexé direct.
- b6.b7 = 0.0 : index **X**
- b3.b2.b1.b0 = 1.0.0.1 : déplacement +/- 15 bits

2.7.3 Adressage indexé avec déplacement accumulateur

Dans ce cas le contenu des accumulateurs A, B ou D sert comme déplacement. L'adresse de la donnée considérée est doc obtenue en ajoutant le contenu de l'un de ces accumulateurs avec le contenu du registre d'index spécifié.

Comme dans les modes à déplacement constant, le poste-octet contient un code de 2 bits permettant de sélectionner le registre d'index considéré.

Si l'accumulateur est A ou B, le champ adressable par variation du contenu de ces registres sera de 256 octets. Par contre si l'accumulateur est D on pourra accéder à la totalité de l'espace adressable du 6809

Exemple :

LDA A, X

Additionne X et A pour obtenir l'adresse effective ou l'on doit trouver le nouveau contenu de A