CHAP03 LE JEU D'INSTRUCTIONS DU 6809

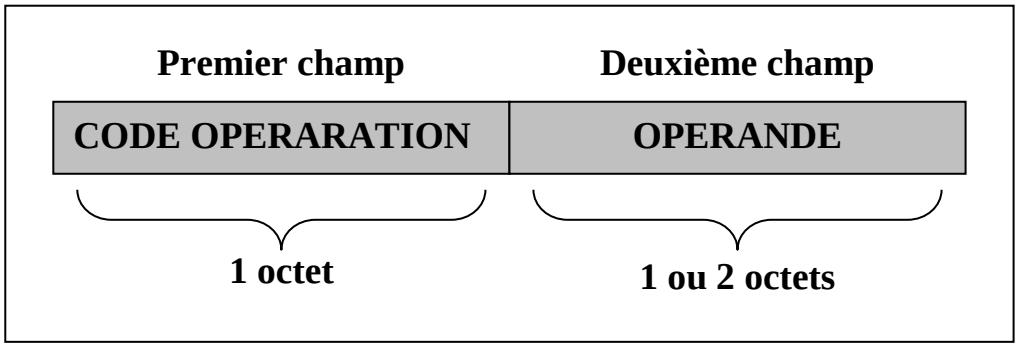
I. INTRODUCTION :

Bien que le nombre de mnémoniques soit limité, le jeu d'instruction du 6809 est très performant. Cette troisième partie présente un classement alphabétique de toutes les instructions, chaque instruction est examinée en détail. Le rôle de l'instruction, le travail sur les indicateurs d'état, les modes d'adressage utilisable, les différentes formes du mnémonique sont mis alors en évidence

II. FORMAT D'UNE INSTRUCTION

L'écriture des instructions d'un programme doit respecter un certain format.

Le code machine d'une instruction se divise en 2 parties ou champs qui contiennent chacun une information spécifique.



CODE MACHINE

- **Le premier champ :** Il spécifie ce qui est par un code opération (**op-code**)
- **Le second champ :** Identifie sur quoi porte l'opération, c'est l'opérande (valeur)

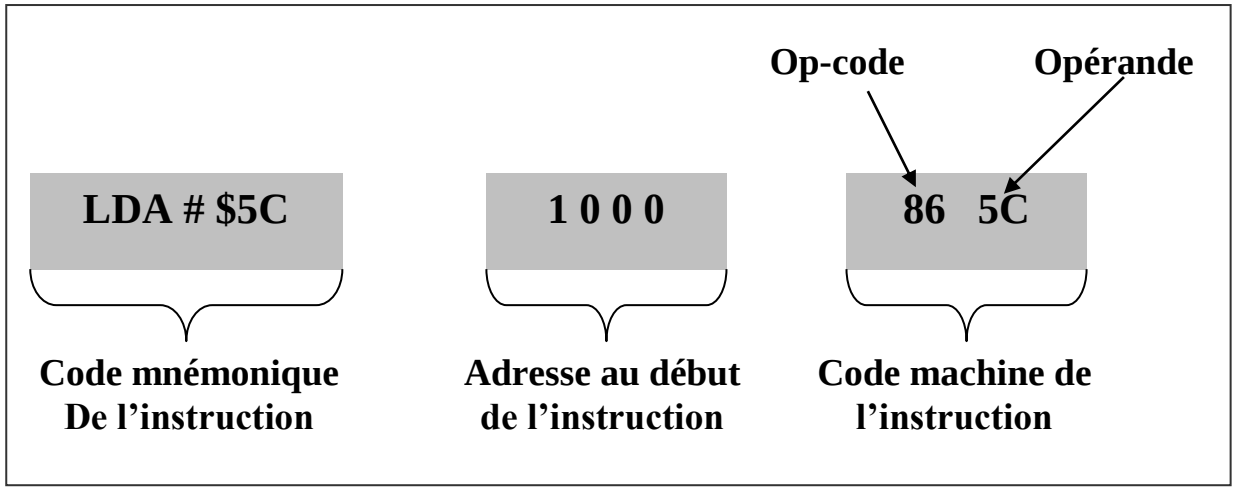
Exemple :

LDA # $5C Elle signifie : Charger l'accumulateur **A** par la valeur hexadécimal **5C**

: Signifie immédiat dans la syntaxe assembleur

$: Signifie hexadécimal

On schématise cette instruction par 3 parties :



III. CLASSIFICATION DES INSTRUCTIONS :

On retrouve dans le jeu d'instruction du 6809 différents groupes d'instructions :

- Instructions de traitement des données ;
- Instructions de transfert des données ;
- Instructions de tests et branchements ;
- Instruction opérant sur les pointeurs ;
- Traitement des interruptions ;

III.1. Instructions de traitement des données :

Les instructions de traitement des données se répartissent en quatre catégories :

- Les instructions arithmétiques ;
- Les instructions de rotation et décalage ;
- Les instructions logiques ;
- Les instructions d'incrémentation-décrémentation, mise a 0, complémentation.

A. Les instructions arithmétiques

Instruction	Fonction
ADD	Addition du contenu mémoire à un accumulateur
ADC	Addition du contenu mémoire à un accumulateur avec retenue
ABX	Addition l'accumulateur B à X (non signé)
DAA	Ajustement décimal de l'accumulateur A
MUL	Multiplication de A par B (non signée)
SUB	Soustraction du contenu mémoire à l'accumulateur
SBC	Soustraction du contenu mémoire à l'accumulateur avec retenue
SEX	Extension de signe de l'accumulateur B à l'accumulateur A

B. Les instructions de rotation et décalage :

Instruction	Fonction
ASR	Décalage arithmétique à droite
LSL ou ASL	Décalage logique ou arithmétique à gauche
LSR	Décalage logique à droite
ROL	Rotation à gauche
ROR	Rotation à droite

C. Les instructions logiques:

Instruction	Fonction
AND	« ET LOGIQUE » entre mémoire et registre interne
EOR	« OU EXCLUSIF » entre mémoire et registre interne
OR	« OU LOGIQUE » entre mémoire et registre interne

D. Les instructions d'incrémentation/décrémentation, mise à zéro, Complémentation

Instruction	Fonction
CLR	Remise à zéro du contenu mémoire ou de l'accumulateur
DEC	Décrémentation du contenu mémoire ou de l'accumulateur
INC	Incrémentation du contenu mémoire ou l'accumulateur
NOP	Pas d'opération. Incrémentation du compteur programme
COM	Complément à un du contenu mémoire ou l'accumulateur
NEG	Complément à deux du contenu mémoire ou de l'accumulateur

III.2. Les instructions de Transferts entre mémoire et registre internes :

Ces instructions se répartissent en trios catégories :

- Instructions opérant sur les registres internes et la mémoire.
- Instructions de transfert opérant sur les pointeurs.
- Instructions opérant seulement sur les registres internes du µP

A. Les instructions opérant sur les registres internes et la mémoire :

Les registres concerné sont **A, B, D, X, Y, S, U**

Les registres PC et DP ne sont pas accessibles directement.

Instruction	Fonction
LD	Chargement des registres internes du MPU
ST	Mise en mémoire des registres internes du MPU

B. Les instructions de transfert opérant sur les pointeurs:

Instruction	Fonction
PSH	Empilement de(s) registre(s) sur la pile
PUL	Dépilement de(s) registre(s)

C. Les instructions opérant sur les registres internes du µP :

Instruction	Fonction
EXG	Echange du contenu de deux registres
TFR	Transfert de registre à registre

III.3. Les instructions de tests et branchements :

Ces instructions peuvent se répartir en trois catégories distinctes :

- Les instructions de test et de comparaisons ;
- Les instructions de test et branchement ;
- Les instructions de saut et branchement ;

1. Les instructions de test et de comparaisons :

Instruction	Fonction
BIT	Test de bits entre accumulateur et contenu mémoire
CMP	Comparaison du contenu mémoire (1 ou 2 octets) avec un registre interne (8 ou 16 bits) du microprocesseur3.
TST	Test du contenu mémoire ou d'un accumulateur

2. Instructions de test et de branchement :

Ces instructions permettent de réaliser des branchements conditionnels. Les tests s'effectuent sur 4 indicateurs du CCR (**N, Z, V, C**)

Ces **4 bits** sont testés seuls ou en combinaisons, on obtient de ce fait 14 instructions de test. Les branchements sont réalisés a partir de l'adresse de l'instruction en cours, ils sont codés sur **8 bits** (déplacement compris entre **-128** et **+127**) ou sur **16 bits** (déplacement compris entre **-32768** et **+32767**).

La lettre **« L »** précédent le mnémonique précise qu'il s'agit d'un déplacement long (16 bit)

Instruction	Fonction
(L)BCC ou (L)BHS	Branchement si pas de retenue
(L)BCS ou (L)BLO	Branchement si retenue
(L) BEO	Branchement si égal à zéro
(L) BNE	Branchement si différent de zéro
(L) BGE	Branchement si supérieur ou égal à zéro (signé)
(L) BLT	Branchement si inférieur (signé)
(L) BGT	Branchement si supérieur (signé)
(L) BLE	Branchement si inférieur ou égal (signé)
(L) BHI	Branchement si supérieur (non signé)
(L) BLS	Branchement si inférieur ou égal (non signé)
(L) BMI	Branchement si négatif
(L) BPL	Branchement si positif
(L) BVC	Branchement si pas de débordement
(L) BVS	Branchement si débordement

3. Les instructions de saut et branchement :

Le tableau suivant regroupe les instructions entraînant des ruptures de séquence, sans conditions préalables. Il faut toutefois différencier les instructions de branchement qui sont suivies d'un déplacement long ou court, des instructions de saut qui sont, elles, suivies de l'adresse effective.

Instruction	Fonction
(L) BRA	Branchement inconditionnel
(L) BRN	Non branchement (non opération)
(L) BSR	Branchement à un sous-programme
JMP	Saut inconditionnel à une adresse effective
JSR	Saut à un sous-programme
RTS	Retour de sous-programme

III.4. Instructions opérant sur les pointeurs :

Instruction	Fonction
LEA	Chargement d'un registre pointeur avec une adresse' effective

III.5. Traitement des interruptions:

Il existe sur le 6809 **3 interruptions** matérielles et **3 interruptions log**icielles :

Instruction	Fonction
CWAY	Validation puis attente d'une interruption
SYNC	Synchronisation du logiciel avec une ligne d'interruption
RTI	Retour de sous-programme d'interruption
SWI/SWI2/SWI3	Interruptions logicielles

IV. LE JEU D'INSTRUCTIONS DU 6809 :

Cette partie est consacrée à une étude détaillée des instructions du 6809.

1. AND ET logique entre mémoire et registre interne

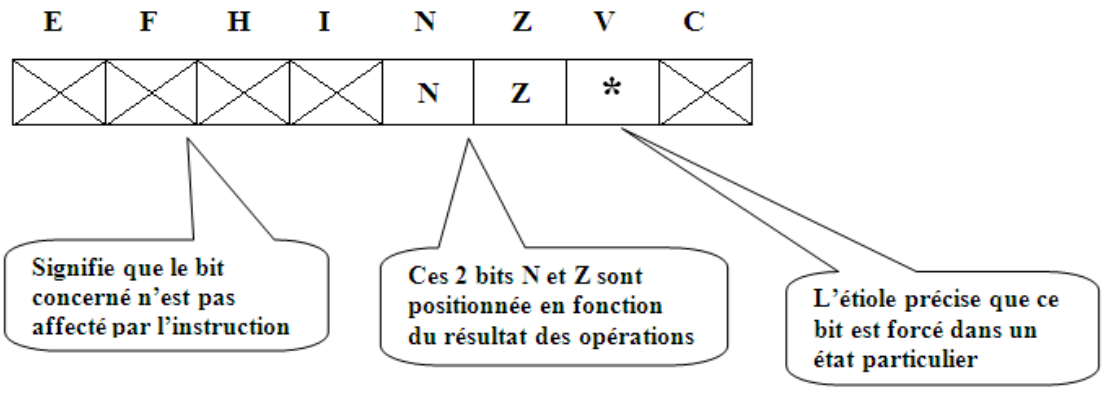
Nom de l'instruction

Fonction de l'instruction

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ANDA	84	2	2	94	4	2	B4	5	3	A4	4+	**2+**
ANDB	C4	2	2	D4	4	2	F4	5	3	E4	4+	**2+**
ANDCC	1C	3	2									

C_O : Code opération (hexadécimal) de l'instruction en fonction de l'adressage choisi
N_C : Nombre de cycles de l'instruction en fonction de l'adressage choisi
N_O : Nombre d'octets de l'instruction en fonction de l'adressage choisi

Etat du registre



Exemple :

- **ANDA adr (8bits)** (Codé 94 adr) Adressage directe

  (ET logique du contenu de l'accumulateur A et le contenu de la position mémoire d'adresse **DPadr.** Le résultat est transféré dans l'accumulateur A)

- **ANDA adr(16bits)** (Codé B4 adr) Adressage Etendu

  (ET logique du contenu de l'accumulateur A et le contenu de la position mémoire d'adresse **adr.** Le résultat est transféré dans l'accumulateur A)

2. ABX Addition de l'accumulateur B au registre X (Non signée)

	INHERENT		
	C_O	N_C	N_O
ABX	3A	3	1

X ⟵ (X) + (B) : ajouté au registre d'indexe **X**, le contenu non signé de l'accumulateur **B**, le résultat est dans **X**

Registre d'état :

E	F	H	I	N	Z	V	C

Exemple :

X=$403C
B=$15
⟶ **ABX** ⟶
X=$403C
B=$15

3. ADC Addition du contenu mémoire a un accumulateur avec retenue

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ADCA	89	2	2	99	4	2	B9	5	3	A9	4+	2+
ADCB	C9	2	2	D9	4	2	F9	5	3	E9	4+	2+

R ⟵ (R) + Donnée + (C) : ajouté a l'accumulateur choisi, le contenu de l'adresse mémoire ou la valeur suivant l'instruction, puis le contenu de l'indicateur de retenue **C**. Le résultat est dans l'accumulateur.

Registre d'état :

E	F	H	I	N	Z	V	C
		H		N	Z	V	C

Exemple :

B=$30

C=1

⟶ ADCB # $11 ⟶ B=$42 (code C9 11)

H	N	Z	V	C
0	0	0	0	0

4. ADD Addition du contenu mémoire à un accumulateur

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ADDA	8B	2	2	9B	4	2	BN	5	3	A9	4+	2+
ADDB	CB	2	2	DB	4	2	FB	5	3	E9	4+	2+
ADDD	C3	4	3	D3	6	2	F3	7	4	E3	6+	2+

R ⟵ (R) + (M) : ajouté le contenu de l'adresse mémoire ou la valeur suivant l'instruction à l'accumulateur chois. Le résultat est dans l'accumulateur.
D ⟵ (A : B) + (M : M+1) : Idem, mais sur 16bits

Registre d'état :

	E	F	H	I	N	Z	V	C
ADDA/ADDB			H		N	Z	V	C
ADDD					N	Z	V	C

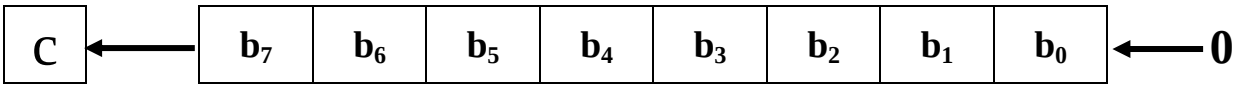
Exemple :

A=$30
B=$C5 } **D=$30 C5 ⟶ ADDD # $1200 ⟶ D=$42C5 (code C3 1200)**

N	Z	V	C
0	0	0	0

5. ASL Décalage arithmétique vers la gauche

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ASL				08	6	2	78	7	3	68	6+	2+
ASLA	48	2	1									
ASLB	58	2	1									



b_7 ⟶ C
.
0 ⟶ b_0 } Décale le contenu de l'accumulateur ou l'adresse mémoire de un bit vers la gauche
Le bit 7 va dans l'indicateur de retenue, le bit 0 est mis a0

Registre d'état :

E	F	H	I	N	Z	V	C
				N	Z	*	C

*** ⟶ V=N⊕C** Apres décalage

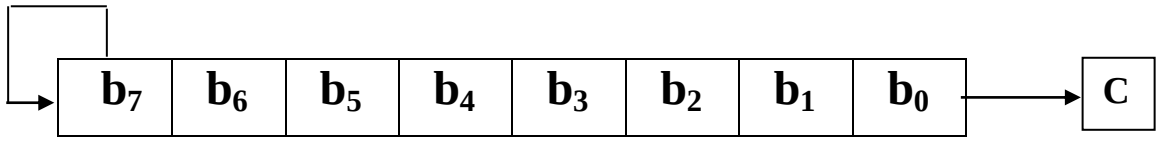
Exemple

A=$83 ⟶ ASLA ⟶ A=$06
(Code **48)**

N	Z	V	C
0	0	1	1

6. ASR Décalage arithmétique à droite

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ASR				07	6	2	77	7	3	67	6+	2+
ASRA	47	2	1									
ASRB	57	2	1									



$b_0 \longrightarrow C$
$b_7 \longrightarrow b_7$
} Décale le contenu de l'accumulateur ou l'adresse mémoire de un bit vers la droite. Le bit 0 va dans l'indicateur de retenue, le bit 7 reste identique

Registre d'état :

E	F	H	I	N	Z	V	C
				N	Z		C

Exemple

A=$83 ⟶ ASRA ⟶ A=$C1
(Code **47)**

N	Z	C
1	0	1

7. CLR Mise à zéro

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
CLR				0F	6	2	7F	7	3	6F	6+	2+
CLRA	4F	2	1									
CLRB	5F	2	1									

00 ⟶ **R** ou **M :** Les accumulateurs **A, B** ou la mémoire sont positionnés à 0

❖ **Registre d'état :**

E	F	H	I	N	Z	V	C
				*0	*1	*0	*0

❖ **Exemple 1**

A=$F4 ⟶ CLRA ⟶ A=$00 ⟶ (Code 4F)

N	Z	V	C
*0	*1	*0	*0

PC = PC+1

❖ **Exemple 2**

M ⟶1000 :34 ⟶ CLR$1000 ⟶1000 : 00 (Code 7F 1000)

N	Z	V	C
*0	*1	*0	*0

PC = PC+3

8. CMP Comparaison du contenu mémoire avec un accumulateur

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
CMPA	81	2	2	91	4	2	B1	5	3	A1	4+	2+
CMPB	C1	1	1	D1	4	2	F1	5	3	E1	4+	2+
CMPD	10.83	5	4	10.93	7	3	10.B3	8	4	10.A3	7+	3+
CMPS	11.8C	5	4	11.9C	7	3	11.BC	8	4	11.AC	7+	3+
CMPU	11.83	5	4	11.93	7	3	11.B3	8	4	11.A3	7+	3+
CMPX	8C	4	3	9C	7	3	BC	7	3	AC	6+	2+
CMPY	10.8C	5	4	10.9C	7	3	10.BC	8	4	10.AC	7+	3+

(R) - (M) : Compare le contenu de mémoire (1 ou 2 octets) avec le registre spécifié (8 ou 16 bits). Les bits du registre d'état sont modifiés en fonction du résultat.

Registre d'état :

H				N	Z	V	C
				N	Z	V	C

(N ⊕ V) = 1 **(R) < (M) signé**
C = 1 ⟶ (R) < (M) non signé
Z = 1 ⟶ (R) = M

Exemple 1 :

A=$2Ø ⟶ CMPA # $ 42

N	Z	V	C
1	0	0	1

$ 42 en complément à 2 ⟶ 0010 0000 → 1011 1110

0 1101 1110

Exemple 2 :

U=$3000 ⟶CMPU # $ 1000

N	Z	V	C
0	0	0	0

$1000 en complément à 2 ⟶ 0011 0000 0000 0000 → 1111 0000 0000 0000

1 0010 0000 0000 0000

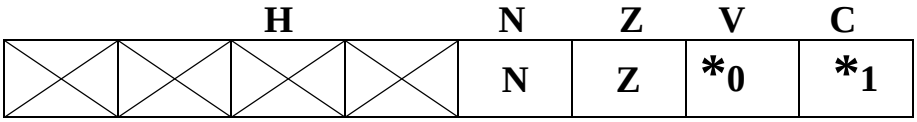
Dans ces deux exemples. Le bit C représente une retenue de soustraction, il est positionné à l'inverse du report binaire du résultat.

9. COM – Complémentation

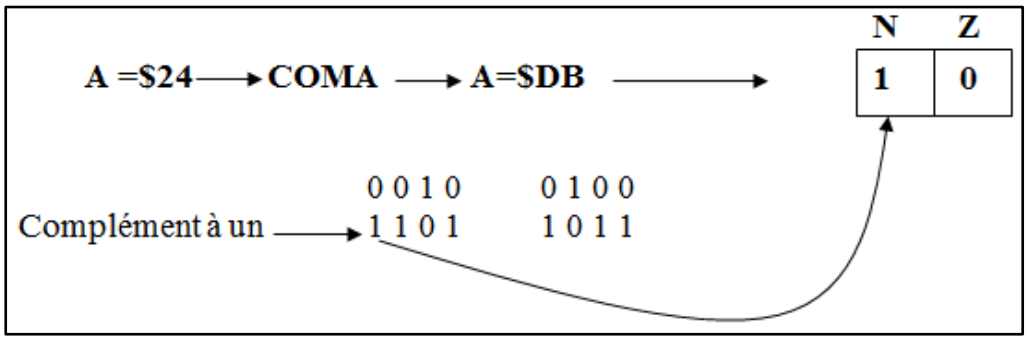
	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
COM				03	6	2	73	7	3	63	6+	2+
COMA	43	2	1									
COMB	53	2	1									

$(\overline{R}) \longrightarrow$ (R) **ou** $(\overline{M}) \longrightarrow$ M **:** Remplacer le contenu d'un octet mémoire ou d'un accumulateur (A ou B) par son complément à un.

❖ **Registre d'état :**



❖ **Exemple 1 :**



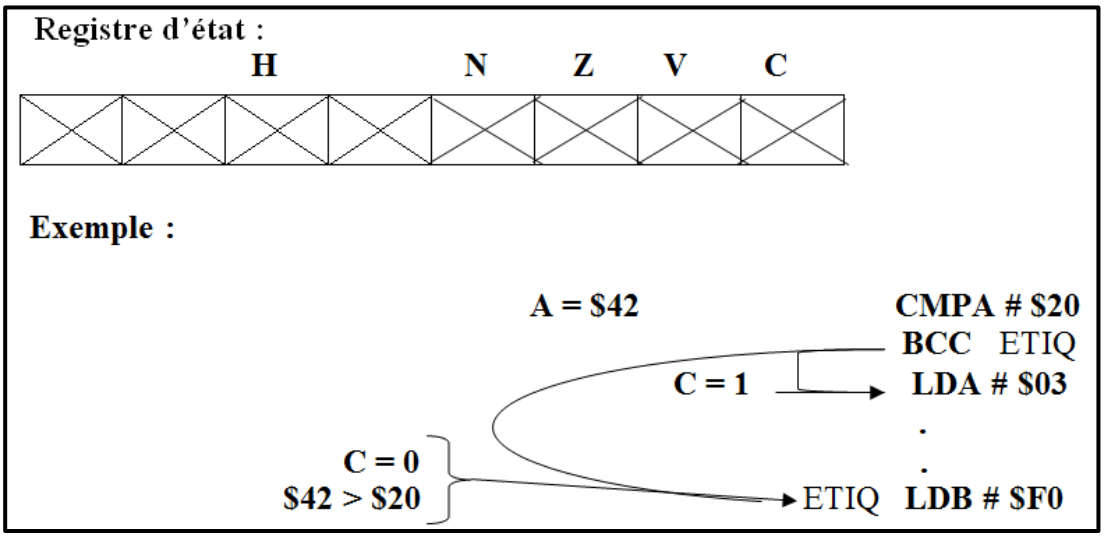
Remarque : Sur des valeurs non signées, on pourra seulement utiliser **BEQ** et **BNE.** Sur des valeurs en complément à deux, tous les branchements signés sont possibles.

10. <u>BCC</u> – <u>Branchement si pas de retenue</u> :

	RELATIF (LONG)		
	C_O	N_C	N_O
BCC	24	3	2
LBCC	10 24	5(6)	4

« **L** » Précèdent le mnémonique précise qu'il s'agit d'un déplacement long (16 bits)

Si C = 0 => PC = PC + déplacement : effectue un branchement si l'opération précédente n'entraîne pas retenue. Déplacement codé sur 8 bits ou 16 bits en complément a 2



11. <u>BCS</u> – <u>Branchement si retenue</u> :

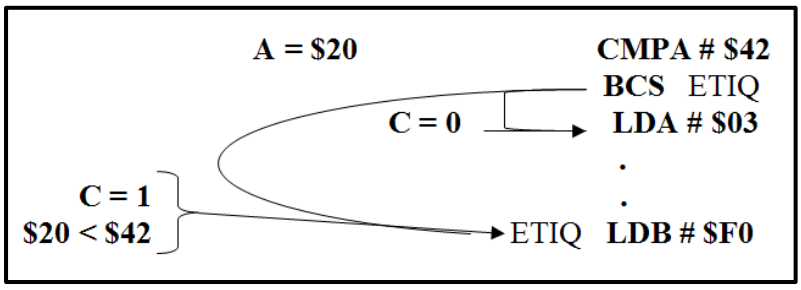
	RELATIF (LONG)		
	C_O	N_C	N_O
BCS	25	2	2
LBCS	10 25	5(6)	4

Si C = 1 => PC = PC + déplacement: effectue un branchement si l'opération précédente entraîne une retenue. Déplacement codé sur 8 bits ou 16 bits en complément à2

Registre d'état :

H		N	Z	V	C

Exemple :



12. <u>BEQ</u> – <u>Branchement si égale a zéro :</u>

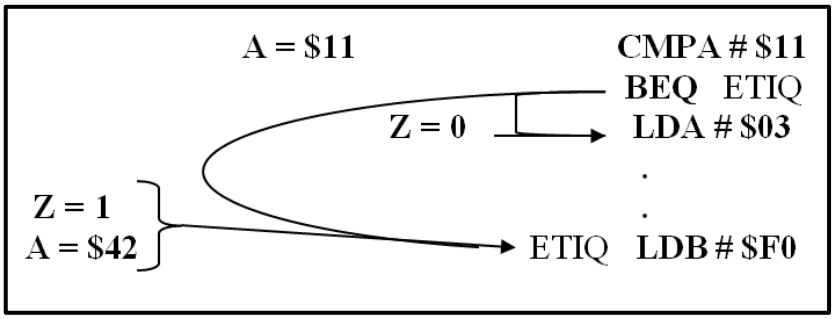
	RELATIF (LONG)		
	C_O	N_C	N_O
BEQ	27	3	2
LBEQ	10 27	5(6)	4

Si Z = 1 => PC = PC + déplacement: effectue un branchement si l'opération précédente entraîne un résultat nul **(Z = 1).** On se branche à l'adresse courante + le déplacement (8 ou 16 bits).

Registre d'état :

H		N	Z	V	C

Exemple :

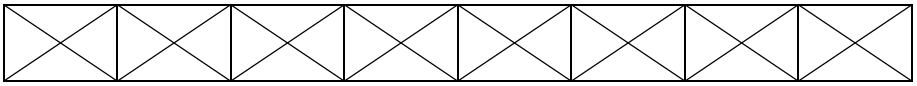


13. <u>BGE</u> – <u>Branchement si supérieur ou égal (signé)</u> :

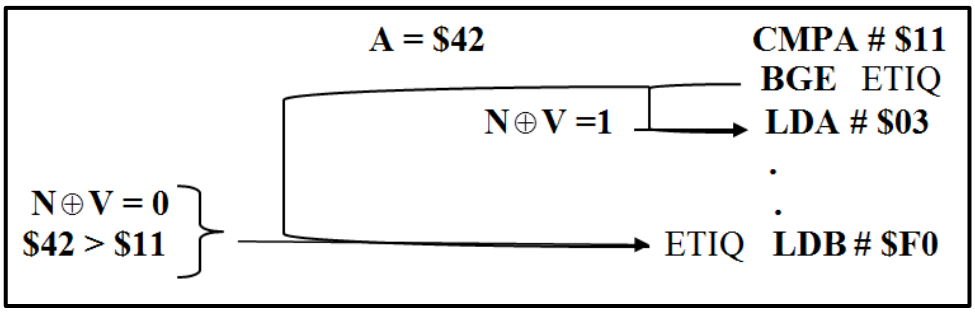
	RELATIF (LONG)		
	C_O	N_C	N_O
BGE	2C	3	2
LBGE	10 2C	5(6)	4

Si (N⊕V) = 0 => PC = PC + déplacement: effectue un branchement si l'opération précédente (en complément à 2) donne un résultat valide et positif

Registre d'état :



Exemple :



14. <u>BGT</u> – <u>Branchement si supérieur (signé)</u> :

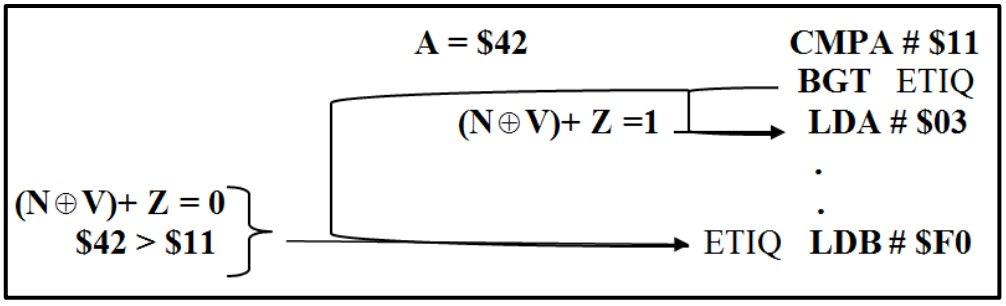
	RELATIF (LONG)		
	C_O	N_C	N_O
BGT	**2E**	**3**	**2**
LBGT	**10 2E**	**5(6)**	**4**

Si Z + (N⊕V) = 0 => PC = PC + déplacement: effectue un branchement si l'opération précédente (en complément à 2) donne un résultat valide et strictement positif

Registre d'état :

H		N	Z	V	C

Exemple :



15. <u>BHI</u> – <u>Branchement si supérieur (non signé)</u> :

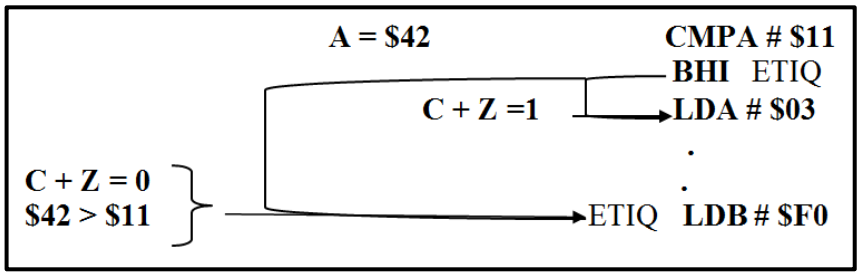
	RELATIF (LONG)		
	C_O	N_C	N_O
BHI	22	3	2
LBHI	10 22	5(6)	4

Si C + Z = 0 => PC = PC + déplacement: effectue un branchement si l'opération précédente entraîne ni retenue ni résultat nul. Instruction inutile après **(INC/DEC, LD/ST, TST/CLR/COM)**

Registre d'état :

H		N	Z	V	C

Exemple :

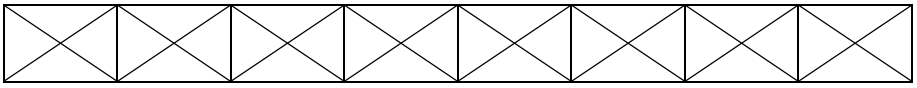


16. <u>BHS</u> – <u>Branchement si supérieur ou égal (non signé)</u> :

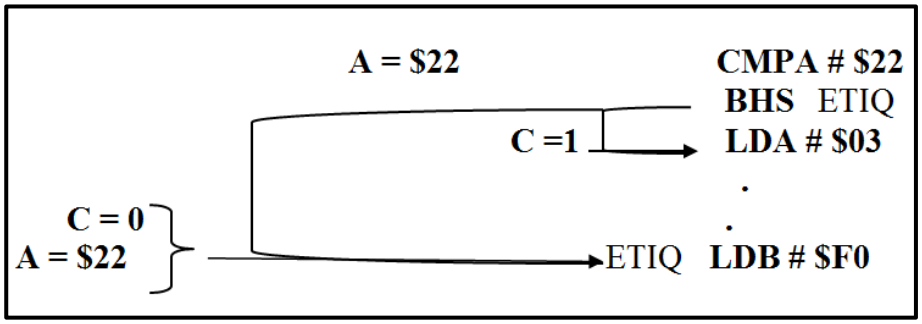
	RELATIF (LONG)		
	C_O	N_C	N_O
BHS	24	3	2
LBHS	10 24	5(6)	4

Si C = 0 => PC = PC + déplacement: Cette instruction est identique à **BCC**, c'est une forme de mnémonique. Si l'indicateur de retenue est nul, on se branche à l'adresse courante + le déplacement.

Registre d'état :



Exemple :



17. BIT : Test de bit

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
BITA	85	2	2	95	4	2	B5	5	3	A5	4+	2+
BITB	C5	2	2	D5	4	2	F5	5	3	E5	4+	2+

(R). (M) : effectue un ET logique entre le contenu d'un accumulateur (**A** ou **B)** et le contenu de l'opérande mémoire seul le registre **CCR** est modifié

Registre d'état :

E	F	H	I	N	Z	V	C
				N	Z	*	

*** V = 0**

Exemple : l'instruction **BIT** précède généralement les branchements conditionnels

A = $ C5 ⟶ BITA # $91 ⟶ { **A = $ C**, **N = 1**, **Z = 0** }

18. BLE – Branchement si inférieur ou égal (signé) :

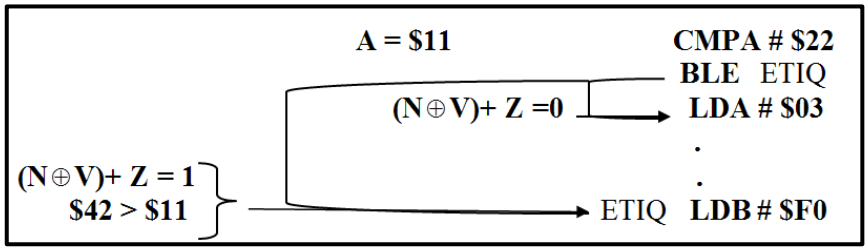
	RELATIF (LONG)		
	C_O	N_C	N_O
BLE	2F	3	2
LBLE	10 2F	5(6)	4

Si Z + (N⊕V) = 1 => PC = PC + déplacement: effectue un branchement si l'opération précédente (en complément à 2) donne un résultat valide et négatif ou nul.

Registre d'état :

H	N	Z	V	C

Exemple :



19. BLO – Branchement si inférieur (non signé) :

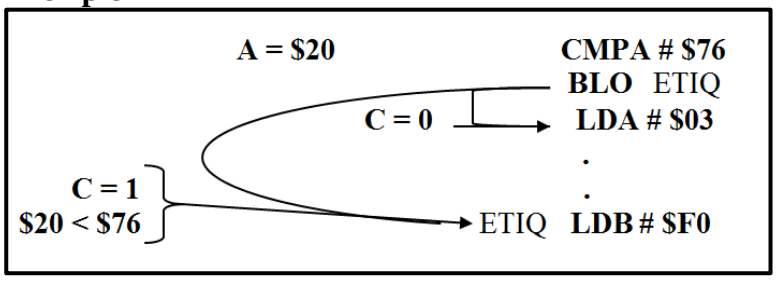
	RELATIF (LONG)		
	C_O	N_C	N_O
BLO	**25**	**3**	**2**
LBLO	**10 25**	**5(6)**	**4**

Si C = 1 => PC = PC + déplacement: Cette instruction est identique à **BCS**, c'est une seconde forme de mnémonique. Si l'indicateur de retenue est à 1, on se branche à l'adresse courante + le déplacement.

Registre d'état :

H		N	Z	V	C

Exemple



20. BLS – Branchement si inférieur ou égal (non signé) :

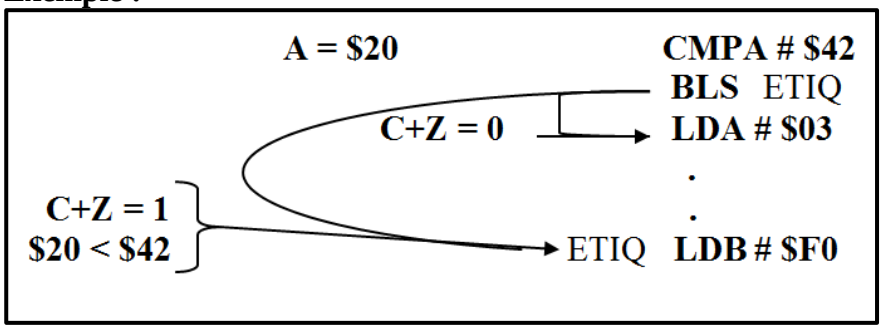
	RELATIF (LONG)		
	C_O	N_C	N_O
BLS	23	3	2
LBLS	10 23	5(6)	4

Si C+Z = 1 => PC=PC + déplacement: effectue un branchement si l'opération précédente entraîne une retenue ou un résultat nul. Cette instruction est inutile après **INC/DEC, LD/ST, TST/CLR/COM**

Registre d'état :

H		N	Z	V	C

Exemple :



21. <u>BLT</u> – <u>Branchement si inférieur (signé)</u> :

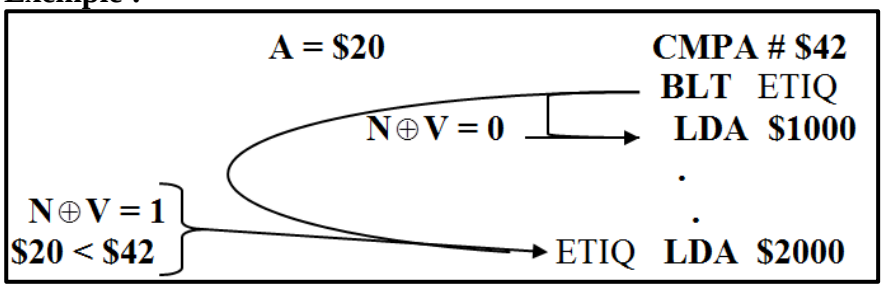
	RELATIF (LONG)		
	C_O	N_C	N_O
BLT	2D	3	2
LBLT	10 2D	5(6)	4

Si (N⊕V) = 1 => PC = PC + déplacement: effectue un branchement si l'opération précédente (en complément à 2) donne un résultat valide et négatif.

Registre d'état :

H		N	Z	V	C

Exemple :



22. <u>BRN</u> – <u>Non Branchement</u> :

	RELATIF (LONG)		
	C_O	N_C	N_O
BRN	**21**	**3**	**2**
LBRN	**10 21**	**5(6)**	**4**

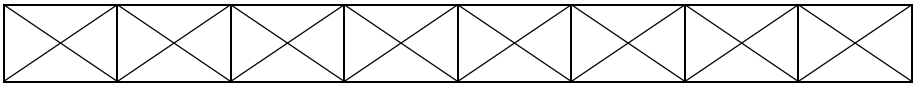
L'instruction ne fait pas de test. Cette instruction n'entraîne aucun branchement, c'est en fait une instruction de non opération. Elle existe par opposition à l'instruction **BRA.**

23. <u>BMI</u> – <u>Branchement si négatif</u> :

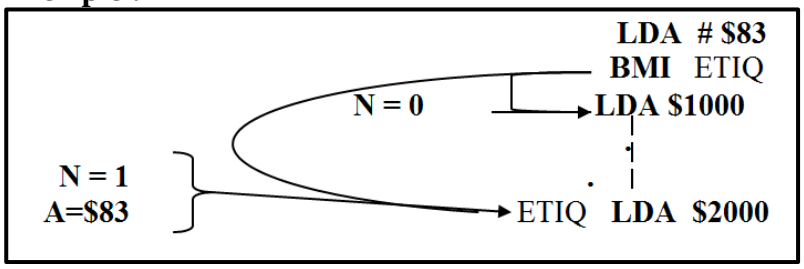
	RELATIF (LONG)		
	C_O	N_C	N_O
BMI	**2B**	**3**	**2**
LBMI	**10 2B**	**5(6)**	**4**

Si N = 1 => PC=PC + déplacement: effectue un branchement si le signe du résultat en complément à deux est négatif. On ne tient pas compte de la validité du résultat en complément (branchement quelque soit V)

Registre d'état :



Exemple :



24. <u>BNE</u> – <u>Branchement si différent de Zéro</u> :

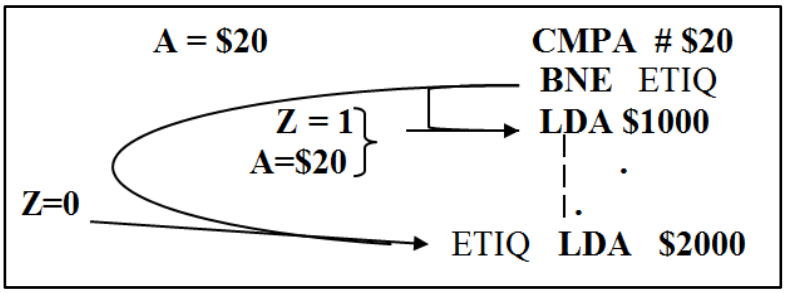
	RELATIF (LONG)		
	C_O	N_C	N_O
BNE	26	3	2
LBNE	10 26	5(6)	4

Si Z = 0 => PC=PC + déplacement: effectue un branchement si le résultat de l'opération précédente est différent de 0.

Registre d'état :

H		N	Z	V	C

Exemple :



25. BPL – Branchement si positif :

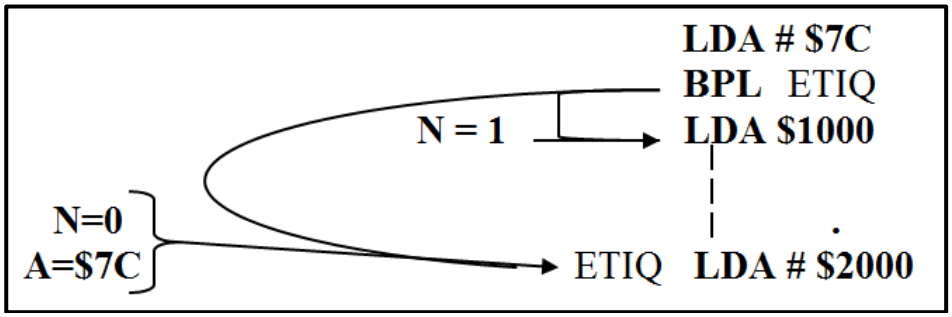
	RELATIF (LONG)		
	C_O	N_C	N_O
BPL	2A	3	2
LBPL	10 2A	5(6)	4

Si N = 0 => PC=PC + déplacement: effectue un branchement si l'opération précédente (en complément a deux) donne un résultat positif 'on ne tient pas compte de V).

Registre d'état :

H		N	Z	V	C

Exemple :



26. BRA – Branchement inconditionnelle :

	RELATIF (LONG)		
	C_O	N_C	N_O
BRA	20	3	2
LBRA	16	5	3

PC=PC+ déplacement : effectue un branchement inconditionnelle, il n y a pas de test.

Registre d'état :

H		N	Z	V	C

Exemple : LDA# $28
BRA

Le programme sur la dernière instruction, en permanence sans condition. Le déplacement est de 2 octets, c'est-à-dire FE (en complément à deux)

27. BSR – Branchement a un sous-programme :

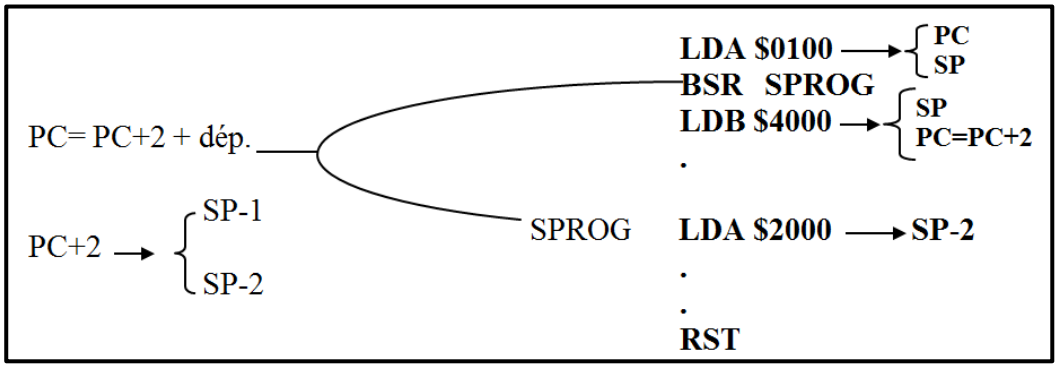
	RELATIF (LONG)		
	C_O	N_C	N_O
BSR	BD	7	2
LBSR	17	9	3

PC_L ⟶ SP-1
PC_H ⟶ SP-1
} **PC = PC + déplacement**: le compteur programme est sauvegardé dans la pile, son contenu additionné au déplacement est chargé dans PC. Il n'y a pas de test préalable.

Registre d'état :

H		N	Z	V	C

Exemple :



28. BVC – Branchement si pas de débordement :

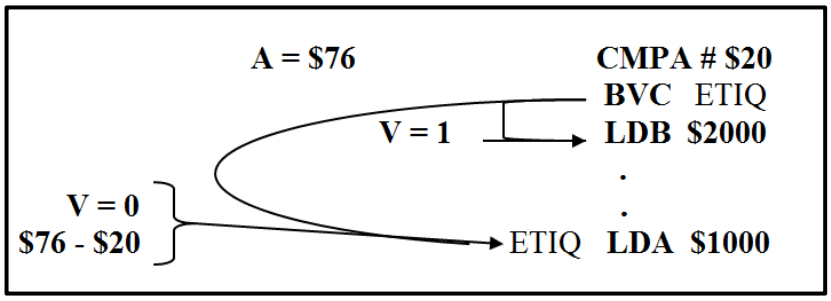
	RELATIF (LONG)		
	C_O	N_C	N_O
BVC	28	3	2
LBVC	10 28	5(6)	4

Si V = 0 => PC = PC + déplacement: effectue un branchement si l'opération précédente (en complément à 2) était valide.

Registre d'état :

H		N	Z	V	C

Exemple :



29. <u>BVS</u> – <u>Branchement si débordement :</u>

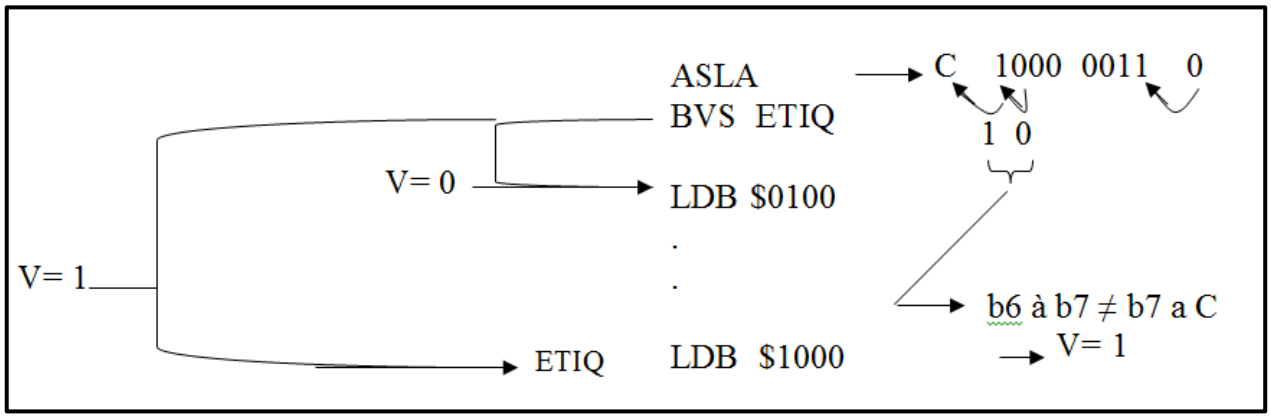
	RELATIF (LONG)		
	C_O	N_C	N_O
BVS	29	3	2
LBVS	10 29	5(6)	4

Si V = 1 => PC = PC + déplacement: effectue un branchement si l'opération précédente (en complément à 2) n'était pas valide.

Registre d'état :

H		N	Z	V	C

Exemple :



30. CWAI – Attente d'interruption :

	INHERENT		
	C_O	N_C	N_O
CWAY # $XX	***3C***	***20***	***2***

(CCR) . (XX) ⟶ **(CCR)**: l'instruction CWAY effectue un « ET logique » entre le contenu du registre d'état et l'octet mémoire le contenu du registre d'état et l'octet mémoire immédiat. On peut de cette façon effacer les marques d'interruptions. L'état entier du processeur est sauvegardé dans la pile système.

Registre d'état :

E	F	H	I	N	Z	V	C
E	**F**	**H**	**I**	**N**	**Z**	**V**	**C**

Remarque :
Tous les indicateurs peuvent être mis à zéro grâce à l'aide de l'octet immédiat.
Plus de détails sur le fonctionnement de cette instruction sont donnés dans le chapitre fonctionnement en interruption.

31. DA – Ajustement décimal:

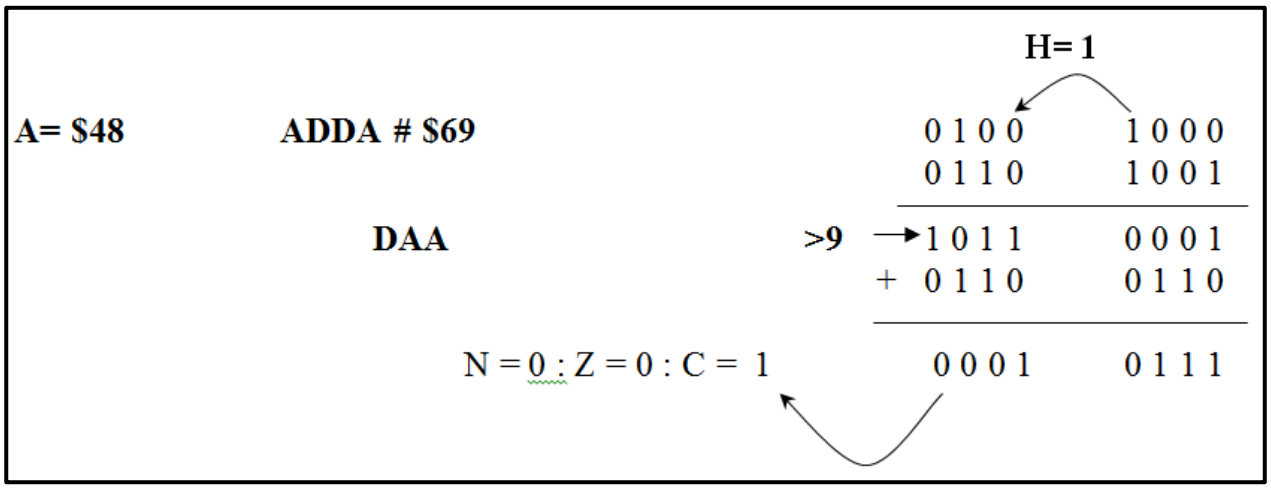
	INHERENT		
	C_O	N_C	N_O
DAA	3C	20	2

(A) ⟵ **(A) + (FC poids faibles ; FC poids forts)**: permet de réaliser des opérations en BCD, FC est facteur correcteur qui permet d'obtenir des résultats corrects.
Quartet poids faibles ⟶ **FC = 6 => H = 1** ou poids faibles > 9.
Quartet poids forts ⟶ **FC = 6 => C= 1** ou poids forts > 9

Registre d'état :

			H		N	Z	V	C
					N	**Z**		

Exemple



32. DEC – Décrémentation:

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
DEC				0A	6	2	7A	7	3	6A	6^+	2^+
DECA	4A	2	1									
DECB	5A	2	1									

M⟵M – 1 : soustraction un de l'opérande qui peut être un octet mémoire ou un registre interne. L'indicateur de retenu n'est pas affecté, DEC peut être utilisée comme compteur. De boucles dans les calculs à précision multiple.

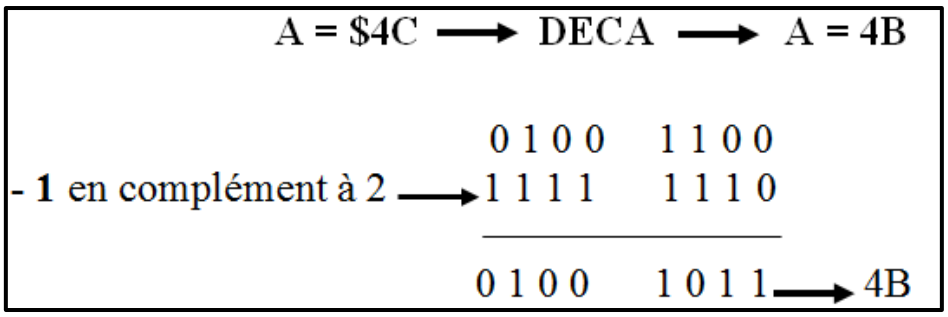
Branchements
- Si les valeurs sont non signées, seules BEQ et BNE peuvent être utilisées.
- Si les valeurs sont signées, tous les branchements signés sont utilisables.

Registre d'état :

H				N	Z	V	C
				N	Z	*	

* V= 1 ⟶ Opérande initiale = $ 80
* V= 0 ⟶ autre cas.

Example:



33. EQR – OU exclusif:

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
EQRA	88	2	2	98	4	2	B8	5	3	A8	4+	2+
EQRB	C8	2	2	D8	4	2	F8	5	3	E8	4+	2+

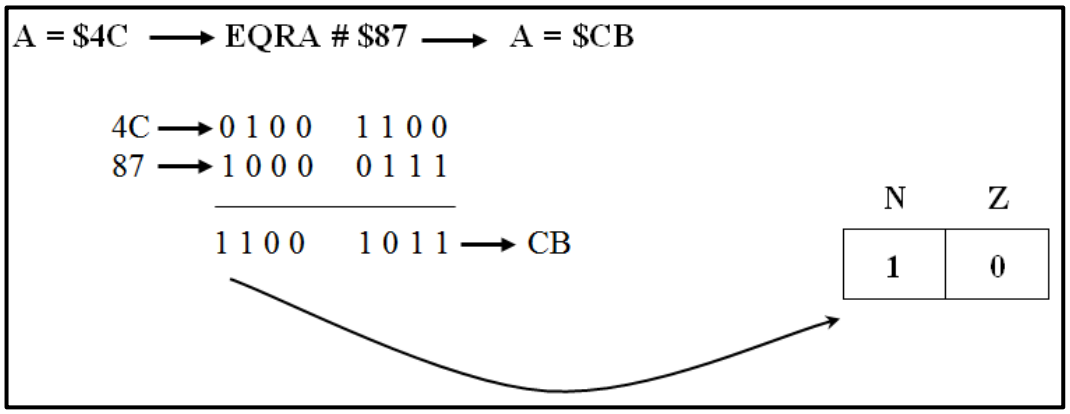
R ← (R) ⊕ (M) : Cette instruction effectue un OU exclusif entre le contenu mémoire ou l'octet immédiat, et l'accumulateur, le résultat est rangé dans l'accumulateur.

Registre d'état :

H				N	Z	V	C
				N	Z	*	

***V =0**

Exemple :



34. EXG – Echange de registre:

	INHERENT		
	C_O	N_C	N_O
EXG	1E	7	2

$(R_1) \rightleftarrows (R_2)$: Echange le contenu de 2 registres. Le mode d'adressage est inhérent, codé sur 2 octets. Le code opératoire est suivi d'un post-octet désignant les registres sources et destination

Post-octet

b7	b6	b5	b4	b3	b2	b1	b0

Source **Destination**

Code	Registre
0	D
1	X
2	Y
3	U
4	S
5	PC
8	A
9	B
A	CCR
B	DP

Registre d'état :

H		N	Z	V	C

Exemple :

A = $20, B = $4C → EXG A, B → A = $4C, B = $20

L'échange ne peut se faire que sur des registres de même taille

35. INC – Incrémentation

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
INC				0C	6	2	7C	7	3	6C	6+	2+
INCA	4C	2	1									
INCB	5C	2	1									

M ⟶ (M) + 1: Additionne 1 a l'opérande, qui peut être un octet mémoire ou un registre interne. L'indicateur de retenue n'est pas affecté, INC peut être utilisé comme compteur de boucles dans les calculs à précision multiple.

Branchements {
- Si les valeurs sont non signées, seules **BEQ** et **BNE** sont utilisable
- Si les valeurs sont non signées, touts les branchements signés sont utilisable.

Registre d'état :

H				N	Z	V	C
				N	Z	*	

* V = 1 ⟶ Opérande initiale = $7F
* V = 0 ⟶ Autre cas

Exemple : A = $32 ⟶ INCA ⟶ A= $32 ⟶ (N=0, Z=0, V=0)

36. JMP – Saut inconditionnel à une adresse effective :

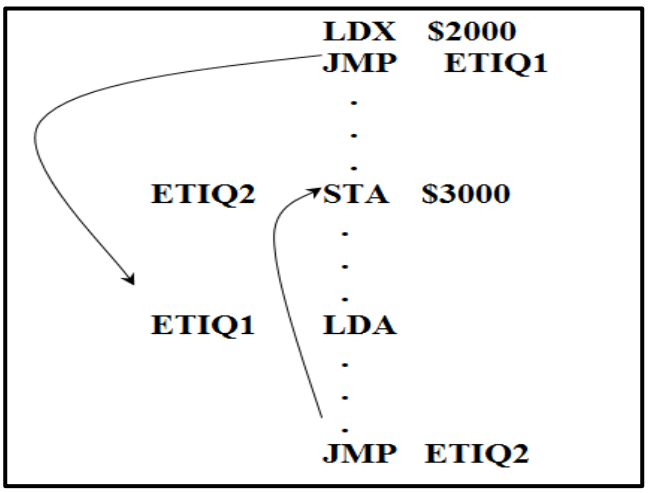
	DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
JMP	**7E**	**4**	**2**	**7E**	**4**	**3**	**6E**	**3+**	**2+**

PC ⟵ A.E : Transfère sans condition l'exécution du programme à un emplacement désigné par l'adresse effective qui suit le code opératoire.

Registre d'état :

H		N	Z	V	C

Exemple :



37. JSR – Saut à un sous- programme :

	DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
JSR	9D	7	2	BD	8	3	AD	7+	2+

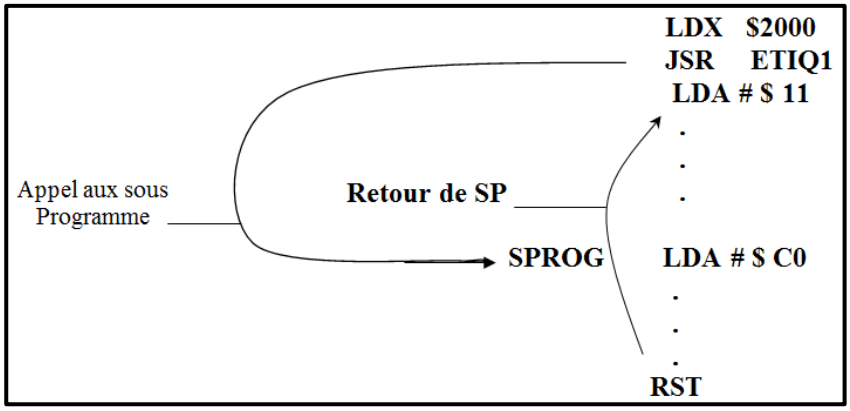
$PC_L \longrightarrow SP - 1$:
$PC_H \longrightarrow SP - 2$:
A.E $\longrightarrow$ PC :

Charge le contenu du PC dans la pile (adresse de retour) puis transfère sans transfère sans condition l'exécution du programme a un emplacement désigné par l'adresse effective qui suit le code opératoire.

Registre d'état :

H		N	Z	V	C

Exemple :



38. LD : Chargement d'un registre

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
LDA	86	2	2	96	4	2	B6	5	3	A1	4+	2+
LDB	C6	2	2	D6	4	2	F6	5	3	E1	4+	2+
LDD	CC	3	3	DC	5	2	FC	6	4	10.A3	7+	3+
LDS	10.CE	4	4	10.DE	6	3	10.FE	7	4	11.AC	7+	3+
LDU	CE	3	3	DE	5	2	FE	6	4	11.A3	7+	3+
LDX	8E	3	3	9E	5	2	BE	6	3	AC	6+	2+
LDY	10.8E	4	4	10.9E	6	3	10.BE	7	4	10.AC	7+	3+

(R) ← (M) : Charge dans le registre interne spécifié (8 ou 16 bits), la valeur
(R) ← (M:M+1) immédiate (8 ou 16 bits) ou le contenu mémoire (1 ou 2 octets)

Registre d'état :

H				N	Z	V	C
				N	Z	*	

* V = 0

Exemple : X = $1000 ⟶ LDX # $8000 ⟶ X= $8000 ⟶ (N=1, Z=0)

39. LEA : Chargement d'une adresse effective :

	INDEXE		
	C_O	N_C	N_O
LEAS	**32**	**4+**	**2+**
LEAU	**33**	**4+**	**2+**
LEAX	**30**	**4+**	**2+**
LEAY	**31**	**4+**	**2+**

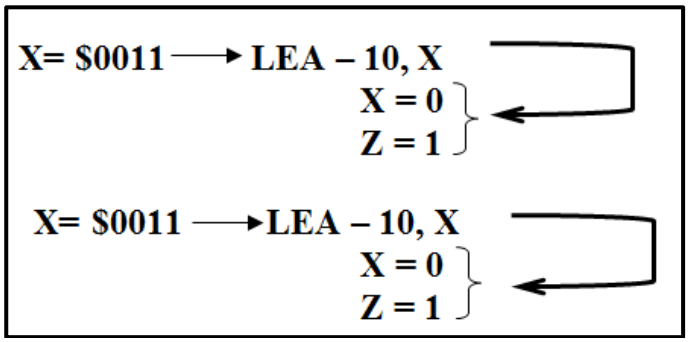
R ⟵ AE : Calcule l'adresse effective en fonction du mode d'adressage indexé et charge cette valeur dans le pointeur désigné.

Registre d'état : * **LEAX** et **LEAY** affectent **Z** en compatibilité avec **INX/DEX** du 6809. **X** et **Y** sont utilisables en compilateur avec arrêt sur zéro.
* **LEAU** et **LEAS** n'affectent pas **Z**, mais assurent la comptabilité avec **INS/DES** du 6800. La valeur zéro pourra être utilisée comme paramètre pour libérer la pile lors du retour à un sous-programme appelant.

Registre d'état :

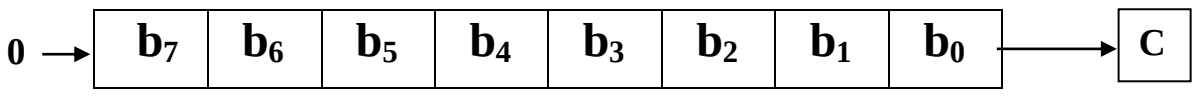
H		N	Z	V	C
			Z		

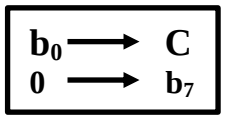
Exemple :



40. LSR Décalage logique à droite

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
LSR				04	6	2	74	7	3	64	6+	2+
LSRA	44	2	1									
LSRB	54	2	1									





Décale le contenu de l'accumulateur ou de l'emplacement mémoire de un bit vers la droite (décalage logique). Le bit 7 prend la valeur 0, le bit 0 va dans l'indicateur de retenue.

Registre d'état :

E	F	H	I	N	Z	V	C
				*	Z		C

* N = 0

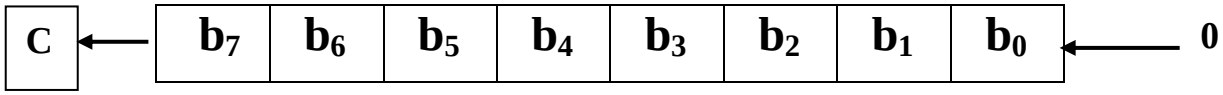
Exemple :

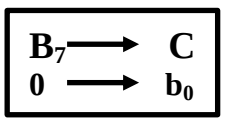
A=$C9 ⟶ LSRA ⟶ A=$64

N	Z	C
0	0	1

41. LSL Décalage logique à gauche :

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
LSL				08	6	2	78	7	3	68	6+	2+
LSLA	48	2	1									
LSLB	58	2	1									





Décale le contenu de l'accumulateur ou de l'emplacement mémoire de un bit vers la gauche (décalage logique). Le bit 7 vas dans l'indicateur de retenue, le bit zéro est lui mis a 0. C'est une 2eme forme de mnémonique de l'instruction **ASL**

Registre d'état :

E	F	H	I	N	Z	V	C
				N	Z	*	C

* V = N⊕C après décalage

Exemple :

A=$C9 ⟶ LSLA ⟶ A=$92

N	Z	V	C
1	0	0	1

42. MUL : Multiplication entre accumulateurs :

	INHERENT		
	C_O	N_C	N_O
MUL	3D	11	1

D ⟵ (A) x (B) : Multiplie contenu de **A** par celui de **B** (nombres binaires non signés) et range le résultat dans **D.** L'octet de poids fort correspond à A, celui de poids faible à B. on peut faire des multiplications à précision multiple

Registre d'état :

H			N	Z	V	C
				Z		*

*C=b7 du registre B.

Le travail de l'indicateur de retenu permet d'arrondir l'octet de poids fort.

Exemple :

A=$C7
B=$21
⟶ MUL
ADCA # 0
⟶ D=$19A7 ⟶ (Z=0,C=1)
⟶ A=$ 1A ⟶ poids fort de D arrondi

43. NEG – Complément à deux :

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
NEG				00	6	2	70	7	3	60	6(+)	2+
NEGA	40	2	1									
NEGB	50	2	1									

R ⟶ (-R) = (R+1)
M ⟶ (-M) = (M+1)

Remplace le contenu de l'accumulateur ou l'octet mémoire par son complément à 2. **C** représente une retenue de soustraction

Registre d'état :

H				N	Z	V	C
				N	Z	*	

*V=1 Si l'opération initiale = 80
*V=0 Autre cas

Retenue de **soustraction** (inverse du report binaire du résultat)

Exemple :

A = $37 → NEGA → A= $ C9

N	Z	V	C
1	0	0	1

- (N⊕V) = 1 → (M) ou (R) >0 (Signé)
- C = 1 → (M) ou (R) >0 (Non Signé)
- Z = 1 → (M) ou (R) =0

44. <u>NOP : Pas d'opération :</u>

	INHERENT		
	C_O	N_C	N_O
NOP	12	2	1

Pas d'opération : cette instruction codée sur un octet n'entraîne pas d'opération , elle incrémente seulement le compteur PC les autres registre ne sont pas modifiés

Registre d'état :

H				N	Z	V	C

Exemple :

```
LDA # $42
NOP
NOP
SLA $2000
```

→ Temporisation logicielle de 4 cycles du microprocesseur
Occupation de 2 octets mémoire

45. OR – « OU Logique » entre mémoire et registre :

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ORA	8A	2	2	9A	4	2	BA	5	3	AA	4+	2+
ORB	CA	2	2	DA	4	2	FA	5	3	EA	4+	2+
ORCC	4A	3	2									

R ← (R) + (M) : Effectue un « **OU** logique » entre le contenu de l'accumulateur (A**, B** ou **CCR)** et le contenu de la mémoire (**M**). Le résultat est rangé dans l'accumulateur spécifié.

Registre d'état :

H				N	Z	V	C
				N	Z	*	

* V=0

Exemple

CCR = 0 **ORCC # $40** **CCR = 40** Les instructions rapides $\overline{\textbf{FIRQ}}$ Sont masquées car **F=1**

46. PSH : Empilement des registres sur une pile :

	INHERENT		
	C_O	N_C	N_O
PSHS	34	5+	2+
PSHU	36	5+	2+

PSH (**U** ou **S**) nom des registres
⟶ (**S** ou **U**) = (**S** ou **U**) - **n**

Empile un, plusieurs ou l'ensemble des registres du microprocesseur dans la pile (système ou utilisateur). Les registres sont rangés dans l'ordre : **PC…….CCR**

Octet Immédiat :

b7	b6	b5	b4	b3	b2	b1	b0
PC	**U/S**	**X**	**Y**	**DP**	**B**	**A**	**CCR**

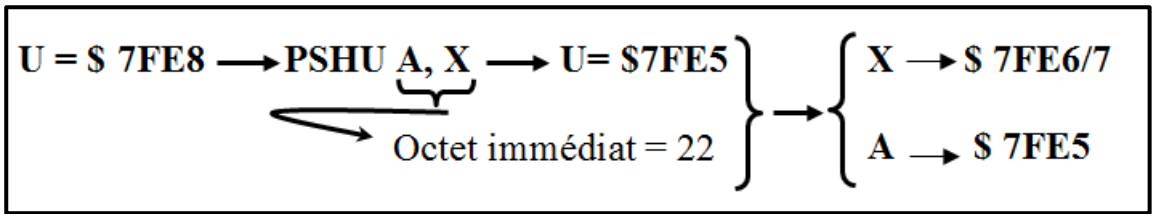
Ordre d'empilement ⟶

L'octet immédiat permet de sélectionner les registres a sauvegarder

Registre d'état :

H		N	Z	V	C

Exemple :



47. PUL : Dépilement de(s) registre(s) :

	INHERENT		
	C_O	N_C	N_O
PULS	35	5+	2+
PULU	37	5+	2+

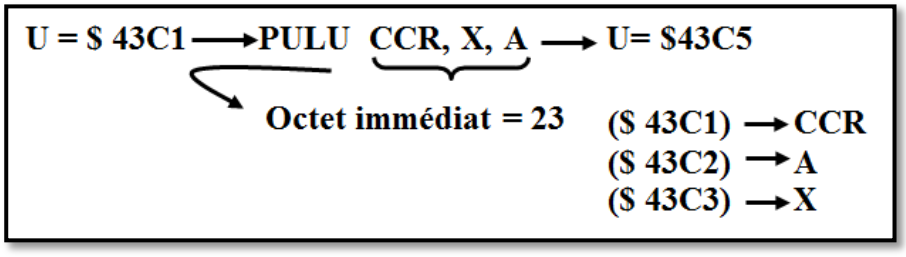
PUL (**U** ou **S**) nom des registres : Dépile un, plusieurs ou la totalité des registres du microprocesseur sauf le pointeur lui même (**S** ou **U**).
Les registres sont tirés dans l'ordre **PC, U/S, X, Y, DP, B, A** et **CCR**, indépendamment de l'ordre apparaissant dans l'instruction. L'octet immédiat permet de sélectionner les registres dépiler.

b7	b6	b5	b4	b3	b2	b1	b0
PC	U/S	X	Y	DP	B	A	CCR

Registre d'état :

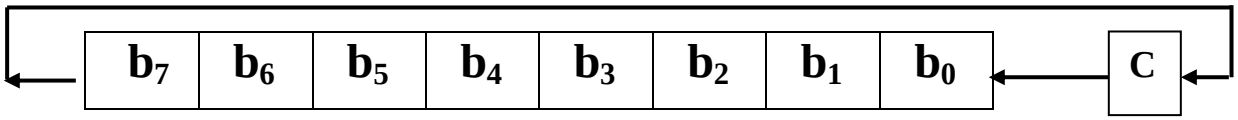
H		N	Z	V	C

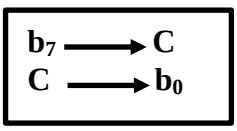
Exemple :



48. ROL :Rotation à gauche :

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ROL				09	6	2	79	7	3	69	6+	2+
ROLA	49	2	1									
ROLB	59	2	1									





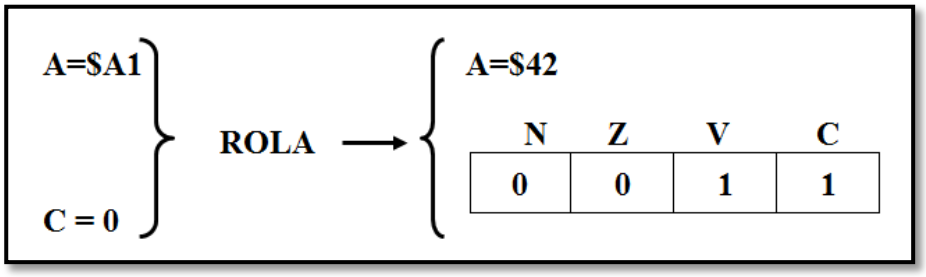
Effectue une rotation d'une position vers la gauche de tous les bits de l'accumulateur ou de l'octet mémoire spécifié par l'intermédiaire de l'indicateur de retenue **C** '(rotation sur 9 bts)

Registre d'état :

E	F	H	I	N	Z	V	C
				N	Z	*	C

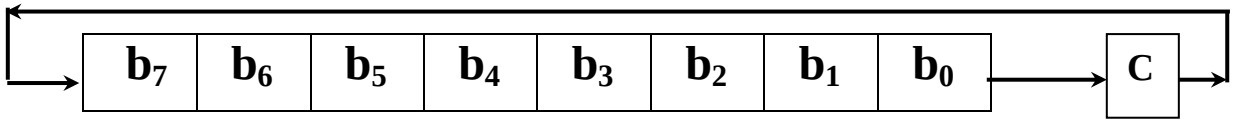
* V = N⊕C après décalage

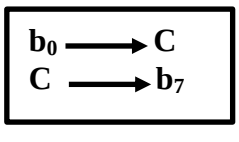
Exemple :



49. ROR : Rotation à droite :

	INHERENT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
ROR				06	6	2	76	7	3	66	6+	2+
RORA	46	2	1									
RORB	56	2	1									



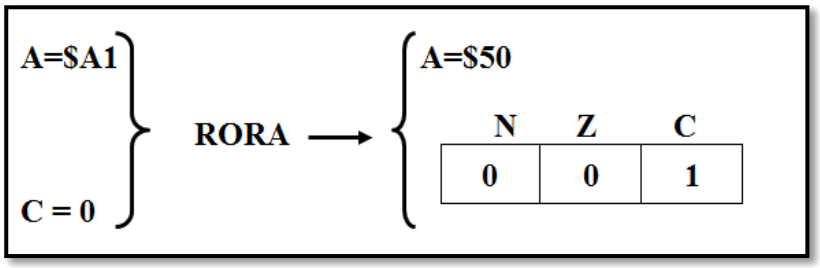


Effectue une rotation d'une position vers la droite de tous les bits de l'accumulateur ou de l'octet mémoire spécifié par l'intermédiaire de l'indicateur de retenue **C** '(rotation sur 9 bits)

Registre d'état :

E	F	H	I	N	Z	V	C
				N	Z	*	C

Exemple :



50. <u>SBC : Soustraction avec retenue :</u>

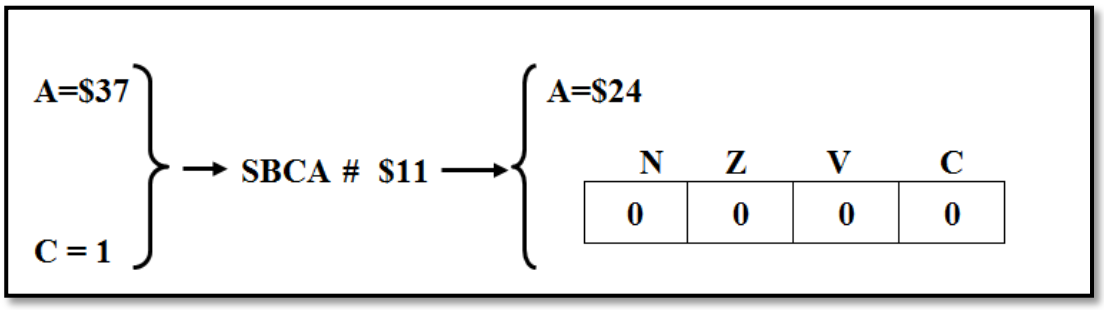
	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
SBCA	82	2	2	92	4	2	B2	5	3	A2	4+	2+
SBCB	C2	2	2	D2	4	2	F2	5	3	E2	4+	2+

R← (R) – (M) – (C) = (R) + $\overline{(M)}$ + 1 + $\overline{(C)}$ + 1 : Soustrait le contenue de l'octet mémoire **M** ou de l'octet immédiat et l'indicateur de retenue **C** a l'accumulateur choisi.
Le résultat est rangé dans cet accumulateur. **C** représente une retenue de soustraction (inverse du report binaire du résultat).

Registre d'état :

E	F	H	I	N	Z	V	C
				N	Z	V	C

Exemple :



51. SUB : Soustraction du contenu mémoire a l'accumulateur :

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
SUBA	80	2	2	90	4	2	B0	5	3	A0	4+	2+
SUBB	C0	2	2	D0	4	2	F0	5	3	E0	4+	2+
SUBD	83	4	3	93	6	2	B3	7	3	A3	6+	2+

R⟵ (R) – (M) => (M) = (R) + $\overline{(M)}$ + 1: Soustrait le contenue de la mémoire **(**1ou 2 octets) ou la valeur immédiat (8 ou 16 bits) du contenu d'un registre interne (8 ou 16 bits). Le résultat est rangé dans le registre interne. **C** représente une retenue de soustraction (inverse du report binaire du résultat).

Registre d'état :

H				N	Z	V	C
				N	Z	V	C

- $N \oplus V = 1 \longrightarrow$ (R) < (M) (Signé)
- $C = 1 \longrightarrow$ (R) < (M) (Non signé)
- $Z = 1 \longrightarrow$ (R) = (M)

Exemple :

A = $24 ⟶ SUBA # $22 ⟶ A = 02, (N=0, Z=0, V=0, C=0)

52. ST : Stockage d'un registre en mémoire :

	DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
STA	97	4	2	B7	5	3	A7	4+	2+
STB	D7	4	2	F7	5	3	E7	4+	2+
STD	DD	5	2	FD	6	3	ED	5+	2+
STS	10.DF	6	3	10.FF	7	4	10.EF	6+	3+
STU	DF	5	2	FF	6	3	EF	5+	2+
STX	9F	5	2	BF	6	3	AF	5+	2+
STY	10.9F	6	3	10.BF	7	4	10.AF	6+	3+

(M) ← (R) **:** Ecrit le contenu d'un des registres internes du microprocesseur (8 ou 16 bits) dans un emplacement mémoire (1 ou 2 octets).

Registre d'état :

H				N	Z	V	C
				N	Z	*	C

* V = 0

Exemple :

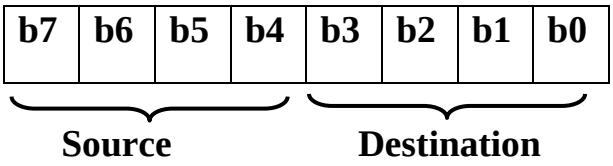
X = $3456 ⟶ STX $1000 ⟶ (N=0, Z=0) ⟶ { 1000 ⟶ 34 ; 1001 ⟶ 56 }

53. TFR – Transfert de registre a registre :

	INHERENT		
	C_O	N_C	N_O
TFR R1, R2	1F	7	2

R_1 ⟶ R_2 : Le 2eme octet de cet adressage précise le nom des registres a transposé

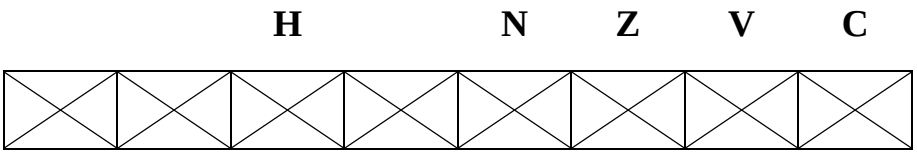
Post-octet



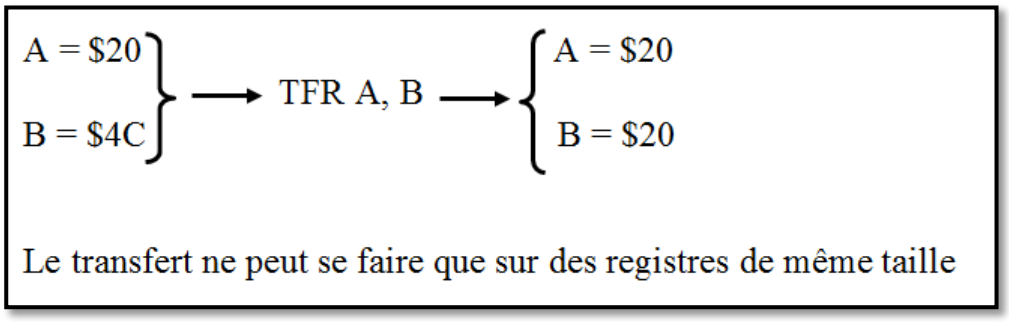
.

Code	Registre
0	D
1	X
2	Y
3	U
4	S
5	PC
8	A
9	B
A	CCR
B	DP

Registre d'état :



Exemple :



54. <u>TST : Test du contenu mémoire ou d'un accumulateur :</u>

	IMMEDIAT			DIRECT			ETENDU			INDEXE		
	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O	C_O	N_C	N_O
TST				0D	6	2	7D	7	3	6D	6+	2+
TSTA	4D	2	1									
TSTB	5D	2	1									

(M) ou (R) ≥ 0 : Positionne les indicateurs **N** et **Z** selon le contenu de l'octet mémoire **M** ou de l'accumulateur. L'indicateur **V** est remis a 0. Ce test se fait par rapport a 0 sur des valeurs non signées. Ce qui rend les instructions de branchement **BLO** et **BLS** parfaitement inutiles. Les branchements signés sont tous disponibles.

Registre d'état :

H				N	Z	V	C
				N	Z	*	

*V = 0

Exemple :

A = $45 ⟶ TSTA ⟶ (N=0, Z=0)

55. RTI – Retour d'interruption :

	INHERENT		
	C_O	N_C	N_O
RTI	3B	6/15	1

(SP) ⟶ CCR

E = 0 ⟶ SP + 3
ou
E = 1 ⟶ SP +

Apres un programme de traitement d'interruption, cette instruction est utilisée pour retourner au programme initial. Le **CCR** est dépilé, puis suivant le résultat du test sur **E**, tout le contexte du microprocesseur est dépilé (**E=1**) ou seulement le compteur programme (**E=0**)

Registre d'état : Prend la valeur du **CCR** qui était sauvegardé dans la pile.
Exemple : Voire fonctionnement en interruptions.

56. RTS – Retour de sous-programme :

	INHERENT		
	C_O	N_C	N_O
RTS	39	5	1

SP ⟶ PC_H :
SP + 1 ⟶ PC_L :
SP + 2

Le contrôle d'exécution retourne au programme initial.
L'adresse de retour est tirée de la pile système.

Registre d'état :

H				N	Z	V	C

Exemple :