pour

OC 2000

MANUEL DE L'UTILISATEUR

ELECTRONIQUE OCCITANE
3 rue Colomiés Z.I. Thibaud - 31084 TOULOUSE

TABLE DES MATIERES.

H O B B Y C O M P U T E R

MANUEL DE L'UTILISATEUR

I-Introduction

Le HOBBY COMPUTER est une cassette spéciale de l'OC 2000 permettant de concevoir et de réaliser soi-même des programmes de jeu, de dessin, de musique, sur son écran de télévision . Il est un excellent moyen d'aborder le domaine de la micro-informatique et de la programmation, par sa facilité et donc, par une initiation très rapide et surtout par ses possibilités audio-visuelles très puissantes .

Les capacités du HOBBY COMPUTER permettent en effet une programmation personnelle sur écran TVC de 4 objets, avec 4 tailles pour chacun des objets, 8 couleurs , le décor, toute la gamme de fréquences audibles avec 4 niveaux sonores plus le bruit d'explosion, et un affichage de 4 scores . Les programmes peuvent être sauvegardés en les enregistrant sur bandes magnétiques . Les claviers de 12 touches montés sur chacune des commandes à distance permettent d'entrer des données et de réaliser diverses fonctions que nous verrons plus loin .

Avant d'entrer dans le détail de l'utilisation du HOBBY COMPUTER, voyons comment faire fonctionner son appareil et écrivons notre premier programme .

1 - Mise en service et affichage initial

Brancher votre OC 2000 comme il est indiqué sur la notice d'emploi de l'appareil . Enclencher la cassette HOBBY COMPUTER . Faire une Remise à Zéro (RAZ touche et touche 0/0). Il doit apparaitre en bas à gauche de l'écran "IIII" . Si ces signes ne sont pas affichés, vérifier que la cassette est bien enfoncée et faire plusieurs fois "RAZ" .

2 - Modes de fonctionnement

Le HOBBY COMPUTER possède deux principaux modes de fonctionnement :
- le mode MONITEUR et
- le mode d'EXECUTION

Il entre automatiquement en mode MONITEUR après la mise sous tension en affichant "IIII" sur l'écran .
Une fois en mode MONITEUR, on peut :

- entrer ou modifier un programme
- relire un programme précédemment enregistré sur cassette audio
- visualiser et modifier le contenu des registrès de travail et/ou du mot d'Etat (PSW : Program Status Word : nous verrons plus loin la signification de ces termes) .
- examiner et modifier le contenu d'emplacements mémoire .
- examiner et modifier le contenu du compteur de programme (PC : Program Counter)

- spécifier et examiner un ou deux points d'arrêt de programme.
- sauvegarder un programme sur cassette magnétique.

Le mode EXECUTION permet l'exécution du programme à partir de l'adresse spécifiée par le compteur du programme (PC).

3 - Le langage du HOBBY COMPUTER = le système HEXADECIMAL.

Dans la vie courante, nous employons pour compter le système de numération à base 10 : c'est le système décimal. Dans le langage d'un microordinateur il lui est plus facile de par sa conception de compter en système binaire (base 2).
Le Hobby Computer utilise des mots de 8 bits ou octets (BIT : BInary digiT).
Le BIT est un élément simple pouvant prendre la valeur 0 ou la valeur 1.
Familiarisons-nous avec les chiffres.

Exemple :

Soit en base 10 le nombre = 234
N s'écrit 234 parce que $N = 2 \times 10^2 + 3 \times 10^1 + 4$

$N = 234$

Soit en base 2 le nombre M = 11101010
M s'écrit ainsi parce que M (base 10) =

$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0$

M = 1 1 1 0 1 0 1 0

et M : 234

Pour simplifier l'écriture de M, on sépare le mot de 8 bits en deux mots de quatre bits. La valeur maximale que prendra chacun des deux mots est 1111 soit 15 en base 10.
On en vient donc à prendre comme base de numération la base 16 = c'est le système hexadécimal. Le tableau ci-dessous résume les équivalences.

DECIMAL	HEXADECIMAL	BINAIRE
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	b	1011
12	C	1100
13	d	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
etc ...	etc...	etc...

Note : les caractères b et d sont écrits en minuscule pour ne pas confondre B avec le chiffre 8 et D avec le chiffre 0

donc si M = 1 1 1 0 1 0 1 0 M = EA

M s'écrira E A

pour passer de base décimale à base hexadécimale, procédons comme suit

Soit à convertir N = 42 (base 10) en base 16

N = 42 = 2 X 16 + 10

N = 2 A (base 16)

2 A

Donc pour entrer une instruction ou une donnée dans la machine, on utilise le système hexadécimal (en abrégé "hexa") qui désigne une méthode d'écriture en raccourci d'un groupe de quatre bits consécutifs en un seul caractère (ou digit).

Opérations dans le système HEXA.

4 - Première utilisation des claviers.

Enfiler les cartes gauche et droite dans leur clavier respectif et positionner les deux commandes côte à côte de façon à pouvoir utiliser sans difficulté les 2 X 12 = 24 touches.

L'écriture des données et adresses dans le HOBBY COMPUTER se fait par le clavier HEXA (composé par les touches 0 à F). Pour entrer une adresse on entre QUATRE digits hexa en commençant par celui le plus significatif (poids le plus fort) de l'adresse. Il n'est pas nécessaire d'entrer les zéros en tête ; si on entre moins de quatre digits, les digits de tête sont automatiquement mis à zéro. Les valeurs des données consistent en DEUX digits hexa, le digit le plus significatif étant le premier.

Additions et soustractions se font comme en base 10.

Exemple :

1 - 8 A
+ 3 7

C 1

car $A_{16} = 10_{10}$ $7_{16} = 7_{10}$ donc $A_{16} + 7_{16} = 10_{10} + 7_{10} = 17_{10} = 16_{10} + 1_{10}$

on pose 1 et on retient 1 (CARRY = 1)

d'où $8_{10} + 3_{10} + 1_{10} = 12_{10} = C_{16}$

Finalement 8 A + 37 = C 1

2 - 8 A
- 3 E
---
4 C

A < E il faut alors faire $(1A - E)_{16} = (26 - 14)_{10} = 12_{10} = C_{16}$

On pose C et on retient 1 (borrow = 1)
D'où 8 - (3 + 1) = 4 Finalement 8 A - 3 E = 4 C

Correction d'erreurs d'entrée.

L'affichage des nombres d'une adresse entrée au clavier se décale vers la gauche à chaque nouveau caractère entré, et les caractères qui disparaissent du champ d'affichage sont perdus. Seuls les derniers digits entrés sont retenus, de sorte qu'une erreur d'entrée peut être corrigée facilement en entrant l'adresse correcte.
Par exemple, si on a entré l'adresse 921 au lieu de la valeur correcte 920, sur l'écran sera affiché : Ad : 921.
Pour corriger, il suffit d'entrer simplement la valeur correcte en appuyant sur les touches suivantes :

0 9 2 0

La valeur correcte sera alors affichée Ad : 0920.
Pour afficher une donnée à une adresse déterminée on entre une valeur comprise en 00 et FF. Pour corriger une erreur d'entrée de donnée on doit afficher à nouveau l'adresse où a été commise l'erreur. Pour ce faire on appuie sur -, ce qui décrémente et positionne l'adresse . Puis on appuie sur +, ce qui incrémente et positionne de nouveau à l'adresse désirée. On entre alors les 2 valeurs hexa de la donnée.

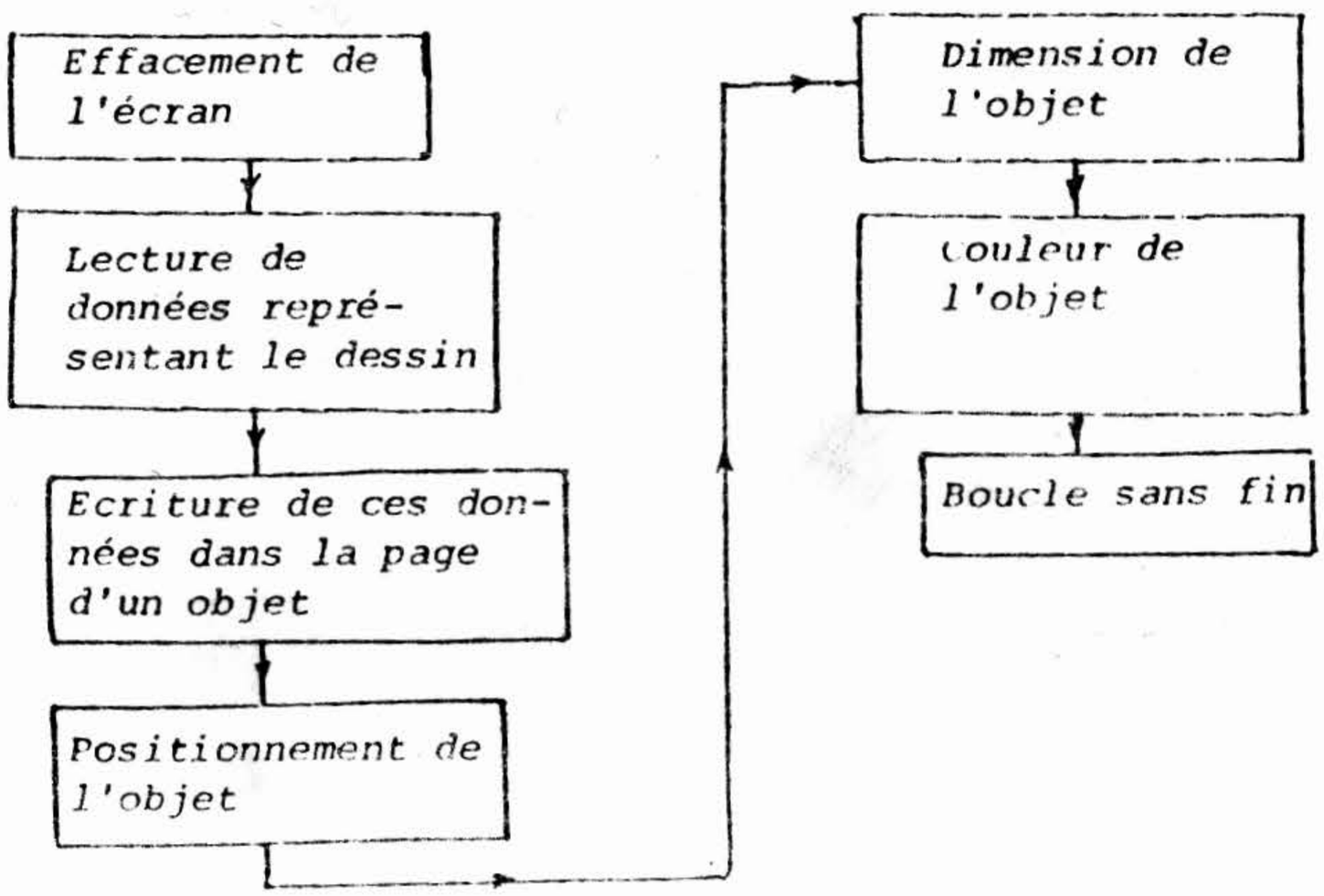
Exemple : à l'adresse 0900 on veut entrer 3 F mais on a écrit 3 E.

Pour corriger il suffit d'appuyer sur les touches suivantes :

Touches	Affichage	Commentaires
-	08FF XX	XX représente la valeur à l'adresse 08FF
+	0900 3E	L'ancienne valeur 3E réapparait.
3	0900 03	Le chiffre 3 est précédé par un "0"
F	0900 3 F	La correction est réalisée.

5 - Entrée et exécution d'un programme simple.

L'exemple choisi consiste à visualiser un objet au milieu de l'écran.
L'objet représente ici un verre mais l'utilisateur pourra modifier à volonté la forme de son objet.
L'organigramme du programme se compose ainsi :

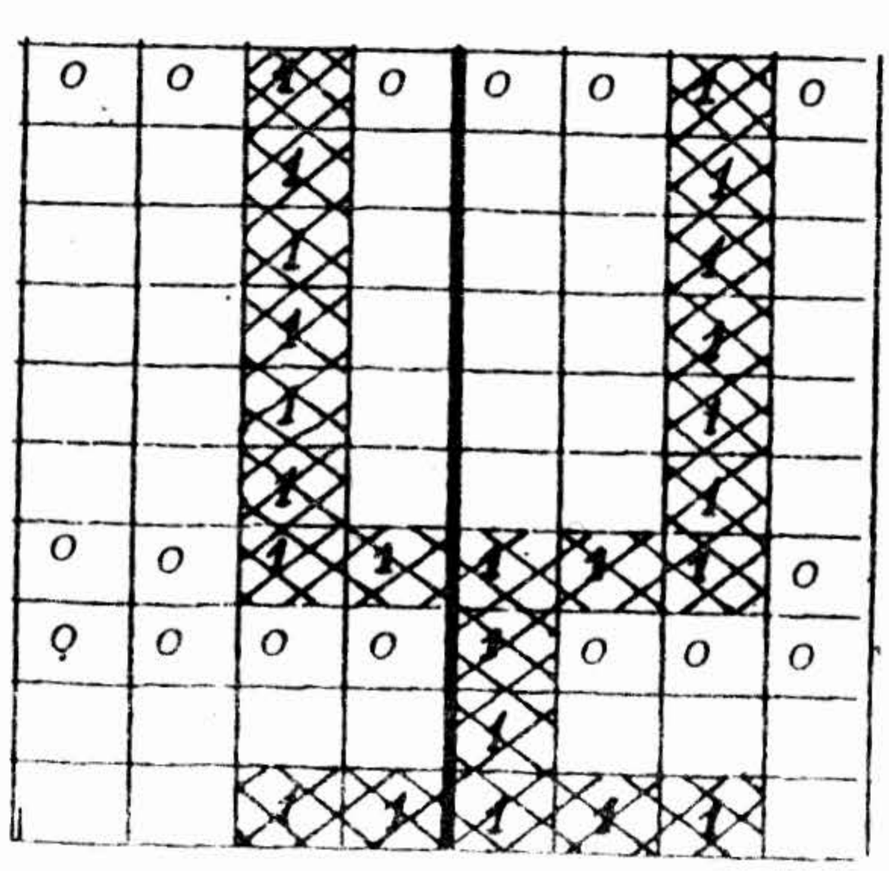


Ecrivons le programme :

ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTION	COMMENTAIRES
0900	3F016E	START	BST, UN	Effacement objets
0903	050A		LODI,R1,0A	
0905	060A		LODI,R2,0A	
0907	0D4A00	OBJ1	LODA,RØ,R1	Lecture et écriture
090A	CE5F00		STRA,RØ,R2	données
090D	5A78		BRN,R2,OBJ1	
090F	0450		LODI,RØ,50	Positionnement hori-
0911	CC1F0A		STRA, RØ	zontal de l'objet
0914	0480		LODI,RØ	Positionnement verti-
0916	CC1F0C		STRA,RØ	cal de l'objet.
0919	04F0		LODI FO	Positionnement hors
091B	CC1F0B		STRA,RØ	écran duplicata
091E	0402		LODI,RØ,02	Dimension de l'objet
0920	CC1FC0		STRA,RØ	
0923	0420		LODI,RØ	Couleur de l'objet
0925	CC1FC1		STRA,RØ	
0928	1B7E	LOOP	BRN,UN LOOP	Boucle sans fin

ADRESSE	VALEUR HEXA	COMMENTAIRES.
0A00	22	10 lignes de données représentant le dessin
0A01	22	de l'objet, ici, un verre à pied.
0A02	22	
0A03	22	
0A04	22	
0A05	22	
0A06	3E	
0A07	08	
0A08	08	
0A09	3E	

Vous pouvez à loisir modifier le dessin de l'objet en changeant les valeurs hexa des cases mémoire d'adresses 0A00 à 0A09. Pour cela, il faut vous aider d'une grille représentant les 10 mots mémoires (voir ci-dessous).



Pour entrer le programme dans l'appareil, voyons la procédure à suivre :

TOUCHE	AFFICHAGE	COMMENTAIRES
ou 0 0	IIII	RAZ du système
Ad	Ad =	Passage en fonction MEMOIRE
9 0 0	Ad = 900	Adresse 0900 de départ
+	0900 XX	Visualisation du contenu de l'adresse 900 (XX)
3 F	0900 3F	Début programme. Après avoir appuyé sur 3 il apparait 03
0 1	0901 01	
6 E	0902 6E	
0 5	0903 05	
0 A	0904 0A	
0 6	0905 06	
0 A	0906 0A	
0 D	0907 0D	
4 A	0908 4A	
0 0	0909 00	
C E	090A CE	
5 F	090B 5F	
0 0	090C 00	
5 A	090D 5A	
7 8	090E 78	
0 4	090F 04	
5 0	0910 50	
C C	0911 CC	
1 F	0912 1F	
0 A	0913 0A	
0 4	0914 04	
8 0	0915 80	
C C	0916 CC	
1 F	0917 1F	
0 C	0918 0C	
0 4	0919 04	
F 0	091A F0	
C C	091B CC	
1 F	091C 1F	
0 B	091D 0B	
0 4	091E 04	
0 2	091F 02	
C C	0920 CC	
1 F	0921 1F	
C 0	0922 C0	
0 4	0923 04	
2 0	0924 20	
C C	0925 CC	
1 F	0926 1F	
C 1	0927 C1	
1 B	0928 1B	
7 E	0929 7E	

Le programme étant écrit, il faut le vérifier en commençant par l'adresse 0900.

TOUCHE	AFFICHAGE
Ad	Ad =
9 O O	Ad = 900
+	0900 3F
+	0901 01
etc...	Jusqu'à l'adresse 0929

Pour corriger une erreur d'entrée, se reporter au paragraphe 4.
Pour entrer les données correspondantes au dessin de l'objet on procède comme pour entrer le programme :

TOUCHE	AFFICHAGE.
Ad	Ad =
A O O	Ad = AOO
+	OAOO XX
2 2	OAOO 22
2 2	OAO1 22
2 2	OAO2 22
2 2	OAO3 22
2 2	OAO4 22
3 E	OAO5 22
O 8	OAO6 3E
O 8	OAO7 O8
3 E	OAO8 O8
	OAO9 3E

Il faut de nouveau vérifier les données à partir de l'adresse OAOO.
On procède comme pour la vérification du programme.

Tout est prêt maintenant pour expérimenter le programme.
Voyons la procédure pour son EXECUTION.

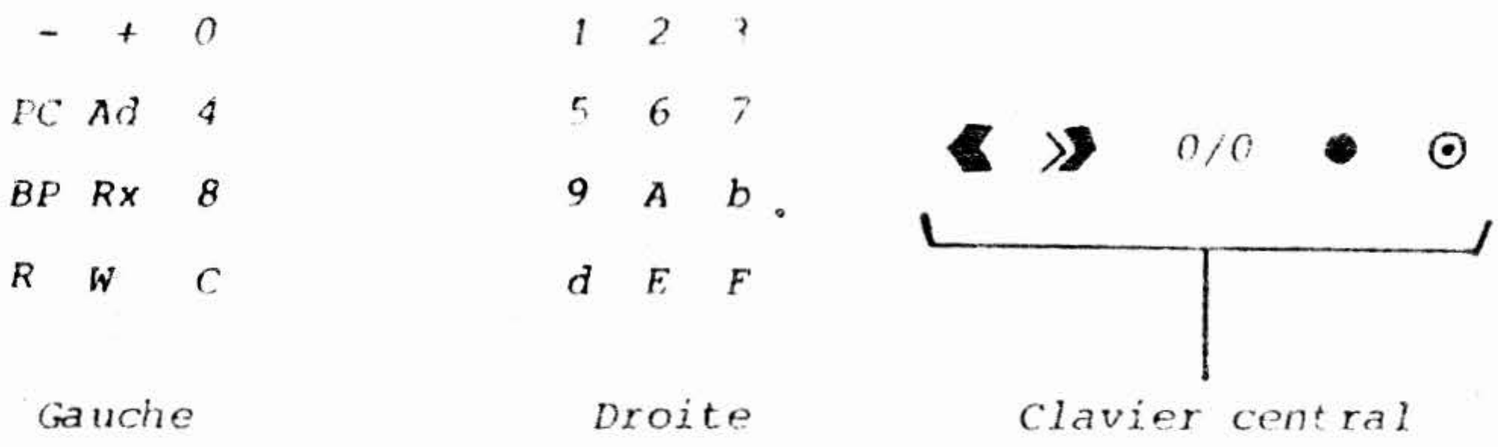
TOUCHE	AFFICHAGE	COMMENTAIRES.
PC	PC = 0000	Programme counter = adresse de
9 O O	PC = 900	début de programme.
+	Le dessin que vous avez choisit. Dans l'exemple donné un verre à pied est affiché au milieu de l'écran.	Mode Exécution

La boucle sans fin qui termine le programme permet de maintenir l'affichage de l'objet sur l'écran indéfiniment. Pour arrêter la boucle, faire une RAZ sur la touche ◀ (insister plusieurs fois) vous revenez au programme Moniteur et sur la dernière ligne est affiché l'adresse du programme Counter. On peut revenir aussi au programme Moniteur en appuyant sur O/O, il apparait IIII et l'adresse du programme Counter (PC) est perdue.

II - DESCRIPTION DES DIFFERENTES COMMANDES

I - Le clavier

Composé de 2 X 12 touches + 5 du clavier central, soit 24 + 5 touches = 29 touches, le clavier permet une programmation complète du HOBBY COMPUTER . Détaillons les diverses fonctions :



0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F = les 16 chiffres du code HEXA .

Les fonctions du MONITEUR sont controlées à partir du clavier dans le seul mode "MONITEUR" . Elles se décomposent suivant :

- ◀ RAZ = Remise à zéro du système ou arrêt du programme utilisateur .
- 0/0 = Initialisation du moniteur et de l'état du microprocesseur .
- PC = Adresse de début de programme à exécuter (Program Counter) .
- Ad = Adresse de mémoire à examiner ou à changer .
- BP = Points d'arrêt du programme utilisateur (Break Points)
- R = Lecture (Read) de cassette à bande magnétique .
- W = Ecriture sur cassette à bande magnétique .
- Rx = Touche de fonction Registres .
- -/+ = Touches d'entrée .

1) ◀ RAZ

Après mise sous tension, le système doit être remis à zéro manuellement RAZ est utilisé aussi pour arréter des programmes utilisateur ou des lectures/écritures de cassette . . RAZ provoque aussi l'initialisation automatique du système et affiche alors "IIII" sur l'écran .

Note : La valeur du registre R0 est perdue lors d'une RAZ . Celui-ci se positionnera toujours à "09" .

2) 0/0

Cette touche permet de remettre à zéro l'état du registre de l'Unité Centrale et positionne PC = 0000 . Les deux points d'arrêts BP1 et BP2 sont aussi à zéro .
L'écran est effacé et "IIII" apparait .

3) Touches d'entrée +/-

Elles sont utilisées pour entrer les données et/ou lister les adresses . Excepté durant la fonction "Ad" les touches +/- sont toujours utilisées pour entrer des données . Pour la fonction "BP", la touche "-" efface l'adresse du point d'arrêt .

4) "PC" début du programme utilisateur

Sur l'écran est affiché "PC = XXXX" après avoir appuyé sur la touche PC . La valeur du PC peut être changée en commençant par le bit de plus fort poids .
Lorsque l'adresse du début du programme utilisateur est fixée dans le PC, pour lancer le programme dans le mode "EXECUTION", appuyer sur "+" ou "-" .

5) "Ad" Fonction mémoire

Le moniteur affiche "Ad" . Il faut alors entrer l'adresse de la mémoire à examiner ou à changer, puis appuyer sur "+" ou "-" pour lire cette mémoire dans laquelle est écrit un nombre composé de deux chiffres HEXA . Si l'on veut changer cette valeur, on entre 2 autres chiffres HEXA, le premier entré étant précédé par un zéro sur l'écran . Après avoir entré ces 2 digits, l'adresse sera automatiquement incrémentée en appuyant sur les 2 prochains digits formant la valeur suivante à entrer .
Exemple :

TOUCHE	AFFICHAGE	COMMENTAIRES
Ad	Ad =	
0 9 0 0	Ad = 0900	
+	0900 XX	XX représente l'ancienne valeur .
3 F	0900 3F	
0 1	0901 01	Il suffit d'appuyer sur 0 et 1 pour incrémenter l'adresse et écrire 01 à cette nouvelle adresse .

La touche "+" permet aussi d'incrémenter les adresses et donc d'examiner leur contenu tandis que la touche "-" décrémente les adresses .

6) "Rx" - Fonction Registre

La programmation du HOBBY COMPUTER est basée sur la programmation du micro-processeur 2650, qui possède un jeu de 7 registres R0, R1, R2, R3, R'1, R'2, R'3 + 2 demi-mots d'état permettant d'uiliser 75 instructions que nous verrons en détail dans le prochain chapitre .
La touche Rx donne immédiatement la valeur de R0 . Si une nouvelle valeur est entrée le signe "=" disparait ., indiquant que cette opération n'est pas encore effectuée . Pour valider l'opération appuyer sur "+" ou "-" et le prochain (+) ou précédent (-) registre est affiché .
L'identification des registres est donnée suivant :

Affichage	Correspondance	
R0	R0	
R1	R1	
R2	R2	Banque de registres 0 (RS = 0)
R3	R3	
R4	R'1	
R5	R'2	Banque de registres 1 (RS = 1)
R6	R'3	
R7	PSL	Program Status Lower - Mot d'état de poids faible
R8	PSU	Program Status Upper - Mot d'état de poids fort .

Nous verrons dans le prochain chapitre comment utiliser les 7 registres et les 2 demi-mots d'état formant le PSW (Program Status Word)

7) "BP" - La fonction Point d'Arrêt

Deux points d'arrêt sont disponibles pour le programme utilisateur . Ces points d'arrêt servent à suivre le programme utilisateur et à visualiser le contenu des registres R0 à R8 afin de détecter une erreur dans l'utilisation des instructions ou dans l'organigramme .
En appuyant une première fois sur la touche "BP", le moniteur affiche l'ancienne valeur de BP1, puis la seconde fois, celle de BP2 . La nouvelle valeur entrée correspond à une adresse et doit être validée par "+" . Le signe "=" apparaitra alors pour indiquer que le moniteur a validé le point d'arrêt .
L'adresse d'un point d'arrêt doit correspondre à celle d'un premier octet d'une instruction .
En mode EXECUTION, le programme s'arrêtera à l'adresse du point d'arrêt, le moniteur affichera la valeur de ce point d'arrêt et le PC prendra l'adresse du point d'arrêt .

II - Utilisation d'un magnétophone - Les touches "R" et "W" .

Le HOBBY COMPUTER comporte une interface cassette à bande magnétique qui permet à l'utilisateur de sauvegarder son programme grâce à un magnétophone . Tout enregistreur de cassette audio de bonne qualité peut être utilisé .

1) Procédure

Mise en place :
- Connecter la sortie cassette du HOBBY COMPUTER à la prise DIN du magnétophone à l'aide du cordon fourni avec le HOBBY COMPUTER .
- S'assurer que la bande est positionnée de telle sorte que des enregistrements précédents ne soient pas détruits par la commande "W" .
- Régler, si possible, le niveau d'entrée du magnétophone à un niveau d'enregistrement normal .

2) Fonctionnement

Nous le voyons avec un exemple :

TOUCHE	AFFICHAGE	COMMENTAIRES
W	bEG =	Le moniteur demande l'adresse de départ de file à écrire sur la bande .
0 8 C 0	bEG = 08C0	L'adresse est entrée
+	ENd =	Le moniteur demande l'adresse de fin de file .
9 F F	ENd = 9FF	L'adresse est entrée .
+	SAd =	Le moniteur demande l'adresse de début de programme .
9 0 0	SAd = 900	L'adresse est entrée .
+	FIL =	Le moniteur demande le numéro de file à écrire sur la bande . Cette possibilité permet de reconnaitre l'identité du programme à enregistrer .
7	FIL = 7	Le numéro est entré . Il faut maintenant faire défiler la bande sur la position "enregistrement" On appuie alors sur la prochaine touche qui est : +
+	FIL = 7	L'affichage se fait en haut de l'écran .

Les données sont écrites sur la bande pendant toute la durée de "FIL = 7" en haut de l'écran . L'écriture est terminée lorsque l'image précédente apparaît .

Remarque : Chaque"file"peut être précédée par quelques commentaires parlés sur la bande sonore .

3) Vérification de l'enregistrement

Sans couper le courant et donc, sans faire perdre le programme en mémoire, on peut et on doit vérifier que l'enregistrement qui vient d'être effectué est correct . Cette opération de vérification ne détruit pas le bon programme contenu dans les mémoires du HOBBY COMPUTER .

Procédure : nous continuons sur l'exemple précédent

TOUCHE	AFFICHAGE	COMMENTAIRES
R	FIL =	Le moniteur demande le numéro de file . Si celui-ci n'est pas spécifié, l'opération se fera sur le premier programme enregistré qu'il rencontrera .

R	FIL = 7	Numéro de file entré .
-	FIL - 7	L'affichage se fait en haut de l'écran. Il faut maintenant lire la bande que nous venons d'enregistrer . Lorsqu'on arrive au bon numéro de file, le moniteur fait clignoter sous le signe - le signe -- à vitesse lente . Si c'est un autre numéro de file, le clignotement est plus rapide .
	FIL == 7	

Si durant l'opération une erreur est détectée, la fonction "R" est arrétée . Le moniteur affiche "Ad = XXXX" où "XXXX" est l'adresse de l'erreur . Il faut alors re-enregistrer le programme .
A la fin d'une vérification sans erreur le moniteur retourne à la fonction PC . "PC = XXXX" est affiché où "XXXX" est l'adresse de début de programme spécifiée dans la fonction "W" (à SAd = XXXX) .

4) Lecture d'un enregistrement

La lecture d'un programme sur une bande magnétique est une opération qui charge les mémoires du HOBBY COMPUTER par des données binaires préalablement enregistrées . La procédure est la même que pour la vérification d'un enregistrement sauf en un point : on utilise la touche "+" au lieu de "-" . La lecture se termine par l'affichage de "PC = XXXX" où XXXX est l'adresse de début de programme . Le programme démarre en entrant un "+" ou un "-" .

- Le tableau du jeu d'instructions

	MNEMO-NIQUE		DESCRIPTION DE L'OPERATION	Code opération R ou CC 3	2	1	0	OCTETS	CYCLES
CHARGEMENT - STOCKAGE	LOD	Z	Load register zero	03	02	01	-	1	2
		I	Load immediate	07	06	05	04	2	2
		R	Load relative	08	0A	09	08	2	3
		A	Load absolute	0F	0E	0D	0C	3	4
	STR	Z	Store register zero	C3	C2	C1	-	1	2
		R	Store relative	CB	CA	C9	C8	2	3
		A	Store absolute	CF	CE	CD	CC	3	4
ARITHMETIQUE	ADD	Z	Add to register zero w/wo carry	83	82	81	80	1	2
		I	Add immediate w/wo carry	87	86	85	84	2	2
		R	Add relative w/wo carry	8B	8A	89	88	2	3
		A	Add absolute w/wo carry	8F	8E	8D	8C	3	4
	SUB	Z	Substract from register zero w/wo borrow	A3	A2	A1	A0	1	2
		I	Substract immediate w/wo borrow	A7	A6	A5	A4	2	2
		R	Substract relative w/wo borrow	AB	AA	A9	A8	2	3
		A	Substract absolute w/wo borrow	AF	AE	AD	AC	3	4
	DAR		Decimal adjust register	97	96	95	94	1	3
LOGIQUE	AND	Z	AND to register zero	43	42	41	-	1	2
		I	AND immediate	47	46	45	44	2	2
		R	AND relative	4B	4A	49	48	2	3
		A	AND absolute	4F	4E	4D	4C	3	4
	IOR	Z	Inclusive-OR to register zero	63	62	61	60	1	2
		I	Inclusive-OR immediate	67	66	65	64	2	2
		R	Inclusive-OR relative	6B	6A	69	68	2	3
		A	Inclusive-OR absolute	6F	6E	6D	6C	3	4
	EOR	Z	Exclusive-OR to register zero	23	22	21	20	1	2
		I	Exclusive-OR immediate	27	26	25	24	2	2
		R	Exclusive-OR relative	2B	2A	29	28	2	3
		A	Exclusive-OR absolute	2F	2E	2D	2C	3	4
ROTATION - COMPARAISON	COM	Z	Compare to register zero arithmetic/logical	E3	E2	E1	E0	1	2
		I	Compare immediate arithmetic/logical	E7	E6	E5	E4	2	2
		R	Compare relative arithmetic/logical	EB	EA	E9	E8	2	3
		A	Compare absolute arithmetic/logical	EF	EE	ED	EC	3	4
	RRR		Rotate register w/wo carry	53	52	51	50	1	2
	RRL		Rotate register left w/wo carry	D3	D2	D1	D0	1	2
BRANCHEMENT	BCT	R	Branch on condition true relative	1B	1A	19	18	2	3
		A	Branch on condition true absolute	1F	1E	1D	1C	3	3
	BCF	R	Branch on condition false relative	-	9A	99	98	2	3
		A	Branch on condition false absolute	-	9E	9D	9C	3	3
	BRN	R	Branch on register non-zero relative	5B	5A	59	58	2	3
		A	Branch on register non-zero absolute	5F	5E	5D	5C	3	3
	BIR	R	Branch on incrementing register relative	DB	DA	D9	D8	2	3
		A	Branch on incrementing register absolute	DF	DE	DD	DC	3	3

Le tableau du jeu d'instructions (suite)

	MNEMO - NIQUE	DESCRIPTION DE L'OPERATION	Code opération R ou CC				OCTETS	CYCLES
			3	2	1	0		
BRANCHEMENT	BDR R	Branch on decrementing register relative	FB	FA	F9	F8	2	3
	BDR A	Branch on decrementing register absolute	FF	FE	FD	FC	3	3
	ZBRR	Zero branch relative unconditional	9B	-	-	-	2	3
	BXA	Branch indexed absolute unconditional .	9F	-	-	-	3	3
BRANCHEMENT ET RETOUR DE SOUS-PROGRAMMES	BST R	Branch to subroutine on condition true, relative	3B	3A	39	38	2	3
	BST A	Branch to subroutine on condition true, absolute	3F	3E	3D	3C	3	3
	BSF R	Branch to subroutine on condition false, relative	-	BA	B9	B8	2	3
	BSF A	Branch to subroutine on condition false, absolute	-	BE	BD	BC	3	3
	BSN R	Branch to subroutine on non-zero register, relative	7B	7A	79	78	2	3
	BSN A	Branch to subroutine on non-zero register, absolute	7F	7E	7D	7C	3	3
	ZBSR	Zero branch to subroutine relative, unconditionnal	BB	-	-	-	2	3
	BSXA	Branch to subroutine, indexed, absolute unconditional	BF	-	-	-	3	3
	RET C	Return from subroutine conditional	17	16	15	14	1	3
	RET E	Return from subroutine and enable interrupt, conditional	37	36	35	34	1	3
ENTREES - SORTIES	WRTD	Write data	F3	F2	F1	F0	1	2
	REDD	Read data	73	72	71	70	1	2
	WRTC	Write control	B3	B2	B1	B0	1	2
	REDC	Read control	33	32	31	30	1	2
	WRTE	Write extended	D7	D6	D5	D4	2	3
	REDE	Read extended	57	56	55	54	2	3
DIVERS	HALT	Halt, enter wait state	-	-	-	40	1	1
	NOP	No operation	-	-	-	C0	1	1
	TMI	Test under mask immediate	F7	F6	F5	F4	2	3
MOT D'ETAT	LPS U	Load program status, upper	92				1	2
	LPS L	Load program status, lower	93				1	2
	SPS U	Store program status, upper	12				1	2
	SPS L	Store program status, lower	13				1	2
	CPS U	Clear program status, upper, masked	74				2	3
	CPS L	Clear program status, lower, masked	75				2	3
	PPS U	Preset program status, upper, masked	76				2	3
	PPS L	Preset program status, lower, masked	77				2	3
	TPS U	Test program status, upper, masked	B4				2	3
	TPS L	Test program status, lower, masked	B5				2	3

III - COMMENT PROGRAMMER LE HOBBY COMPUTER

Le jeu d'instructions du microprocesseur 2650

I - Introduction

Le HOBBY COMPUTER est un microordinateur qui se programme en code machine avec le jeu d'instructions du microprocesseur 2650 qui est le centre de calcul de l'OC 2000 .
Les instructions sont au nombre de 75, qui possèdent 4 modes d'adressage qui sont :

- l'adressage implicite noté Z
- l'adressage immédiat noté I
- l'adressage relatif noté R
- l'adressage absolu noté A

II - Les principaux registres

Les instructions définissent l'opération à effectuer par le microprocesseur dans l'un des 4 modes d'adressage choisi , à l'aide de registres contenant les valeurs des opérandes . Ces registres sont au nombre de 7, comme on l'a vu au chapitre précédent .

Exemple : les 7 registres sont des mots de 8 bits : R0, R1, R2, R3, R'1, R'2, R'3
On choisit une des 2 banques
en positionnant RS à 0 ou à 1 . RS = 0 RS = 1
Soit RS = 0 .
On veut additionner 2 nombres se trouvant dans des mémoires d'adresse X et Y et mettre le résultat dans une mémoire d'adresse W :
On charge le contenu de la mémoire d'adresse X dans R0 : $X \longrightarrow R0$
On charge le contenu de la mémoire d'adresse Y dans R1 : $Y \longrightarrow R1$
On additionne R0 avec R1 et le résultat se trouve dans R0 :

$$(R1) + (R0) \longrightarrow R0$$

On stocke le résultat à l'adresse W :

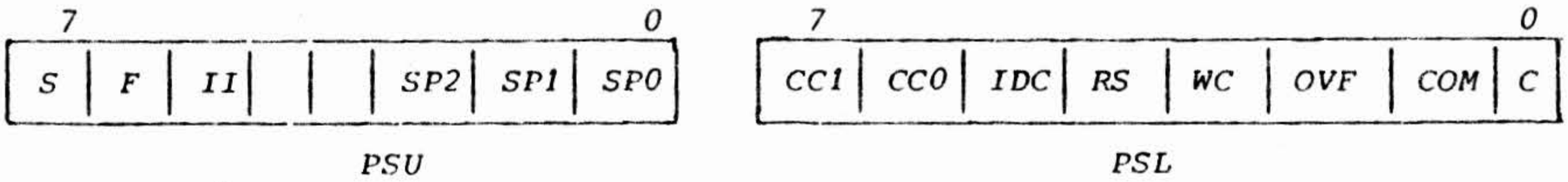
$$(R0) \longrightarrow W$$

Cet exemple nous montre le genre d'opérations qu'on peut effectuer avec les registres . Ici, le code de chaque opération (chargement, addition, stockage) n'a pas été mentionné . Nous le verrons dans les prochains paragraphes avec le détail de chaque instruction .

Le Mot d'Etat (PSW)

C'est un registre spécialisé qui contient des informations d'état et des bits de contrôle .
Il est composé de 2 octets : PSU partie haute
PSL partie basse

44
50



Les différents bits du PSW peuvent être testés, stockés, chargés, mis à un, mis à zéro à l'aide des instructions affectant le PSW . La répartition des bits est la suivante :

PSU

S — *Sense = ce bit est à l'état 1 pendant le retour trame et à 0 pendant la durée d'une image .*

F — *Flag = Ce bit sert à utiliser la commande en manche à balai dans le plan X-Y . F = 0 commande en X, F = 1 commande en Y .*

II — *Sert à invalider des demandes d'interruption*

SP2
SP1
SP0 — *Pointeur de la pile d'adresses de retour . Ces 3 bits servent à localiser dans cette pile l'adresse de retour d'un sous-programme . Cette pile peut donc recevoir $2^3 = 8$ adresses de retour ; on dispose donc de 8 niveaux de sous-programmes .*

PSL

CC1
CC0 — *Codage des conditions . Ces deux bits sont affectés par les instructions de comparaison et de test arithmétiques et logiques .*

IDC — *Inter Digit Carry = retenue intermédiaire entre digits .*

RS — *Register Select = sert à identifier la banque de registres utilisés :*
RS = 0 R1, R2, R3
RS = 1 R'1 (= R4), R'2 (=R5), R'3 (=R6)

WC — *With Carry = avec retenue . Précise si la retenue doit être utilisée ou non lors d'instructions arithmétiques ou de rotation .*

OVF — *Overflow = débordement . Ce bit est mis à 1 si un débordement en complément à deux se produit .*

COM — *Compare = précise si la comparaison à effectuer doit être logique ou arithmétique (grandeurs non signées ou signées)*

C — *Carry : Représente la retenue de l'unité arithmétique et logique ou le bit sortant lors d'une rotation .*

IV - Répertoire du jeu d'instructions

(Extrait du "Répertoire du jeu d'instructions du microprocesseur 2650" édité par R.T.C. - la Radiotechnique - Compelec)

1 - Format des instructions

Page 18

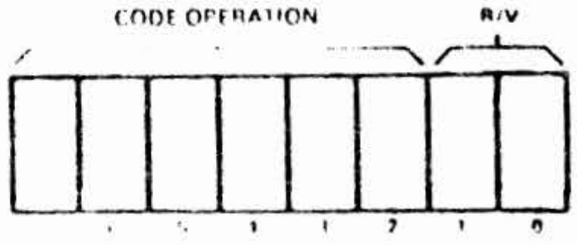
2 - Les divers types d'adressage

Pages 19 à 33

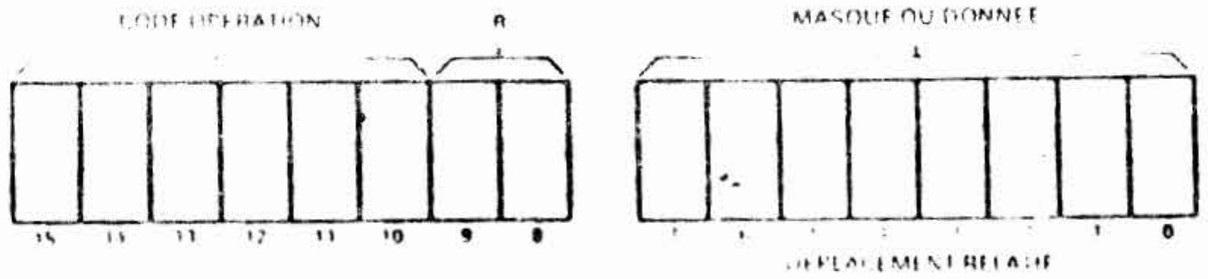
3 - Les 75 instructions du HOBBY COMPUTER

format des instructions

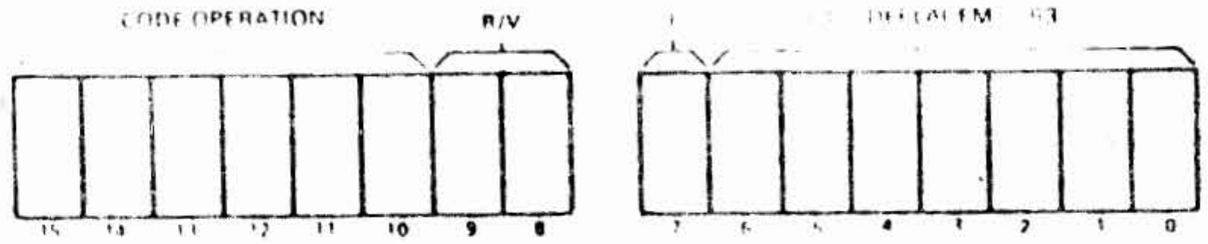
(Z) ADRESSAGE IMPLICITE



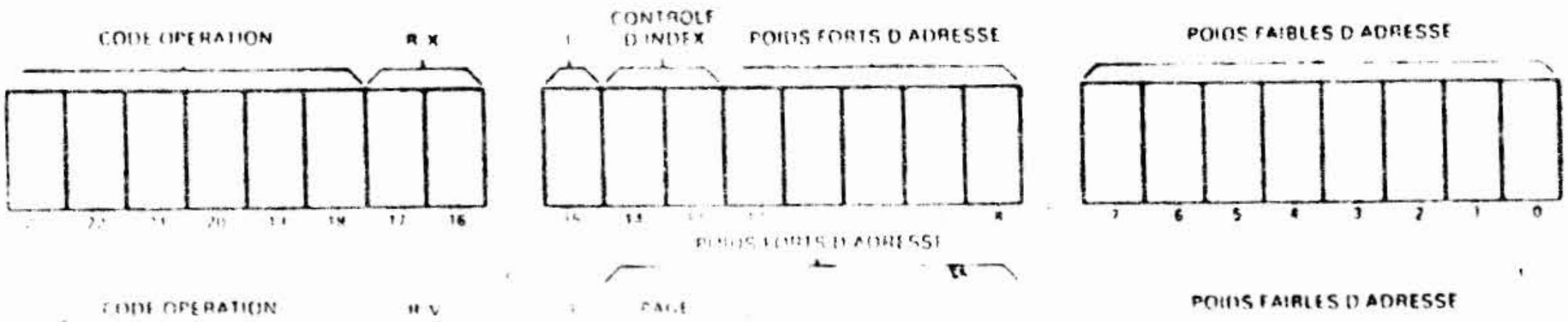
(I) ADRESSAGE IMMEDIAT



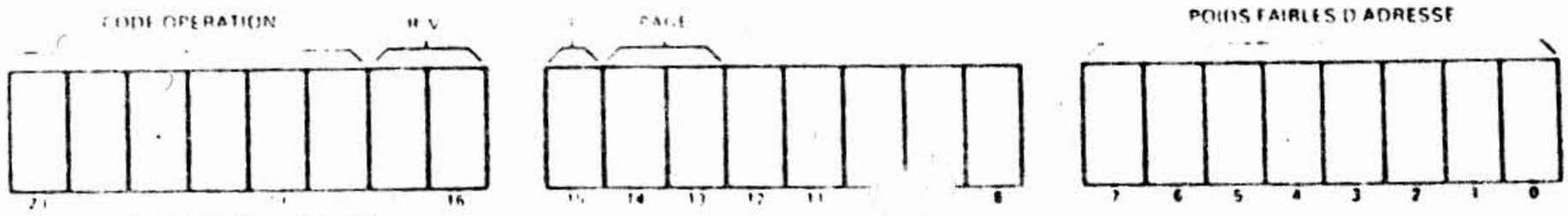
(R) ADRESSAGE RELATIF



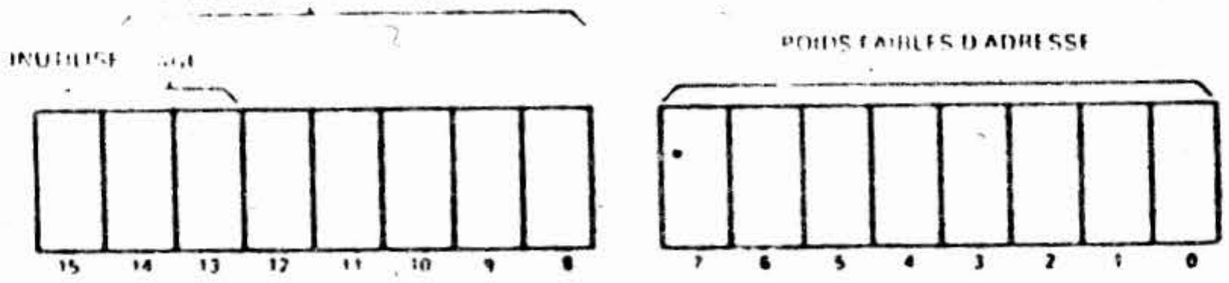
(A)



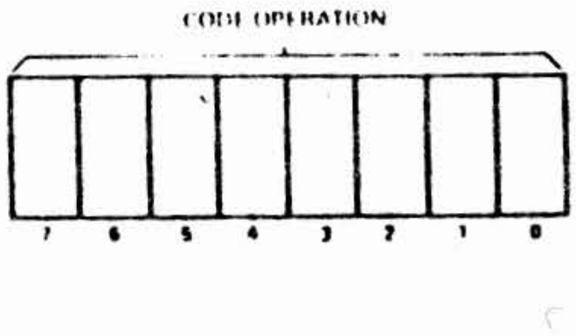
(B) ADRESSAGE ABSOLU (INSTR DE BRANCHEMENT)



ADRESSAGE INDIRECT



(E) INSTRUCTIONS DIVERSES



SYMBOLES
R = NUMERO DE REGISTRE
V = VALEUR OU CONDITION
X = NUMERO DU REGISTRE D INDEX
I = BIT D INDIRECTION

CONTROLE D INDEX
00 = NON INDEXE
01 = INDEXE AVEC PRE INCREMENT
10 = INDEXE AVEC PRE DECREMENT
11 = SIMPLEMENT INDEXE

2 - Les divers types d'adressage

a) ADRESSAGE IMMEDIAT

OPERANDES : - Registre
- Valeur

OCCUPATION MEMOIRE : 2 octets

CODE BINAIRE :

DESCRIPTION : La donnée à accéder se trouve dans l'instruction elle-même .

EXEMPLES

1) Chargement de 34 hexa dans R3

LODI, R3 H'34' R3 ⟵ H'34'

Code binaire : | 0 0 0 0 0 1 1 1 | | 0 0 1 1 0 1 0 0 |

Code hexa : 07 34

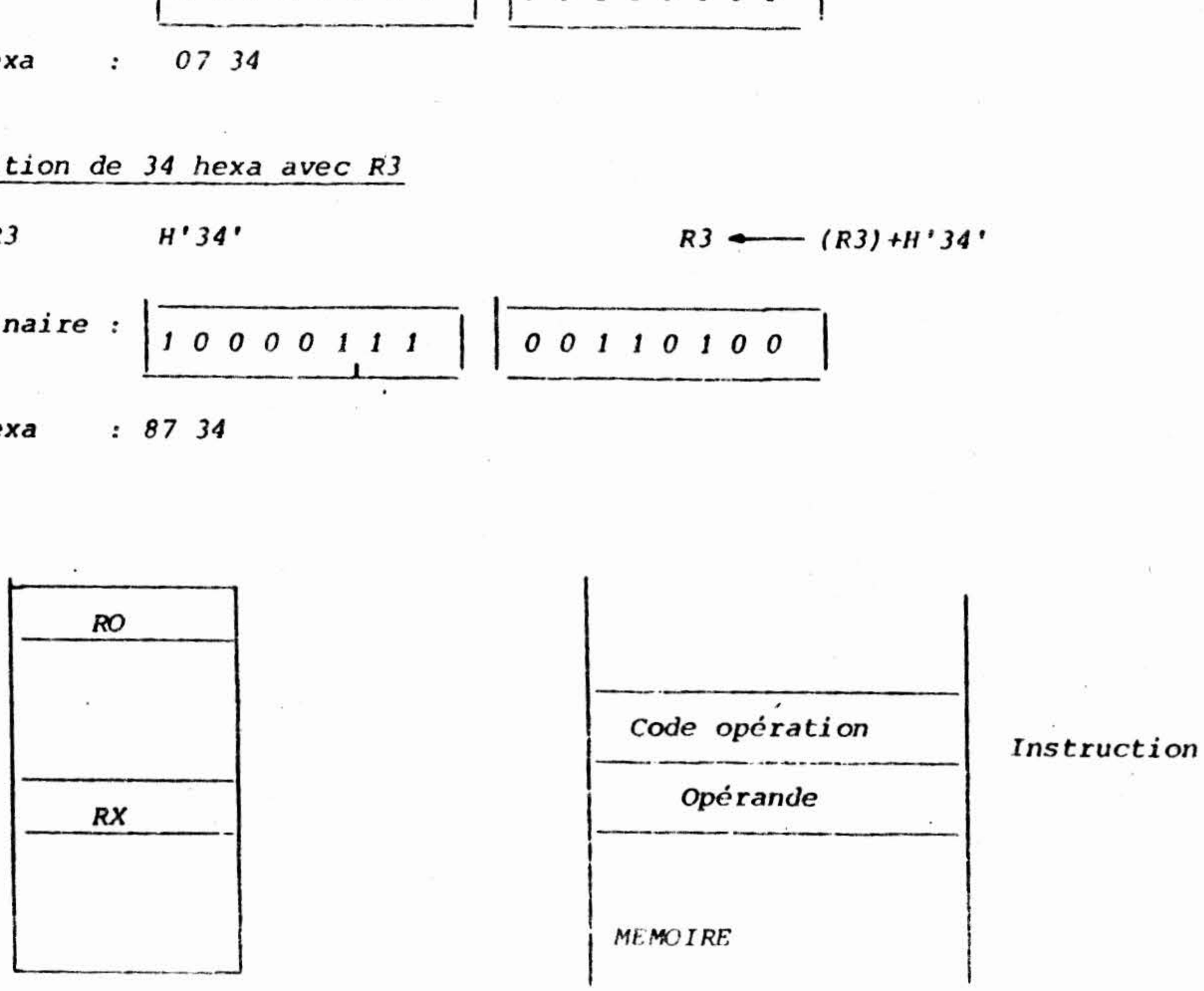
2) Addition de 34 hexa avec R3

ADDI, R3 H'34' R3 ⟵ (R3)+H'34'

Code binaire : | 1 0 0 0 0 1 1 1 | | 0 0 1 1 0 1 0 0 |

Code hexa : 87 34

SCHEMA

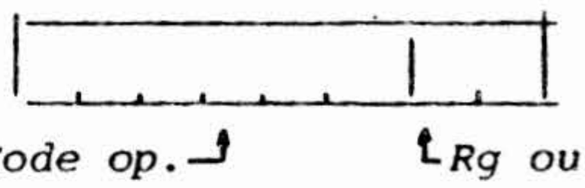


b) ADRESSAGE "REGISTRE" IMPLICITE

OPERANDES : - R0 implicitement
- Registre

OCCUPATION MEMOIRE : 1 octet

CODE BINAIRE :



DESCRIPTION ---------- C'est une instruction dont le premier opérande est toujours le registre R0, et le second un autre registre .

EXEMPLES
----

1) Chargement de R2 dans R0

LODZ R2 R0 ⟵ (R2)

Code binaire : | 0 0 0 0 0 0 1 0 |

Code hexa : 02

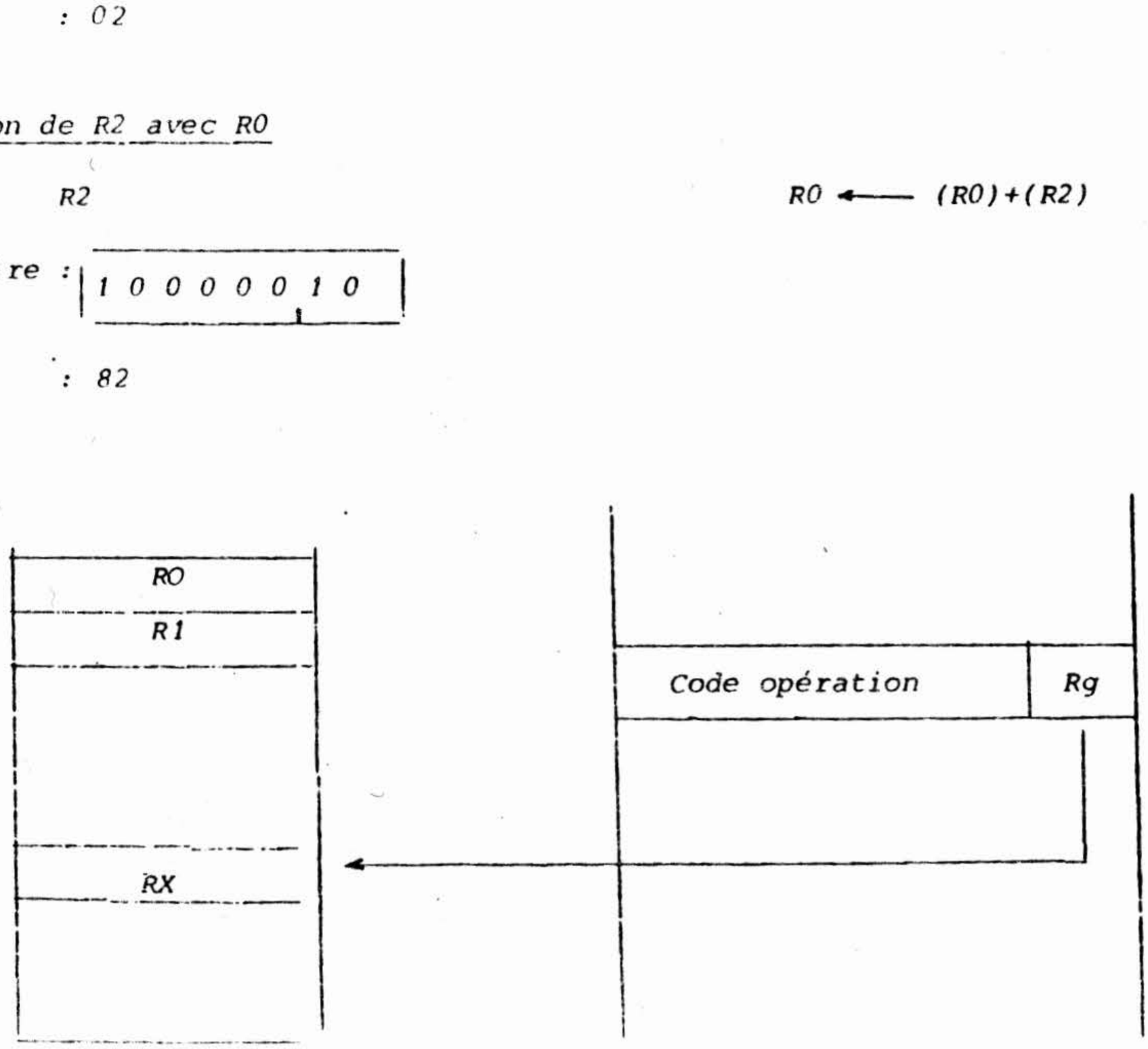
2) Addition de R2 avec R0

ADDZ R2 R0 ⟵ (R0)+(R2)

Code binaire : | 1 0 0 0 0 0 1 0 |

Code hexa : 82

SCHEMA

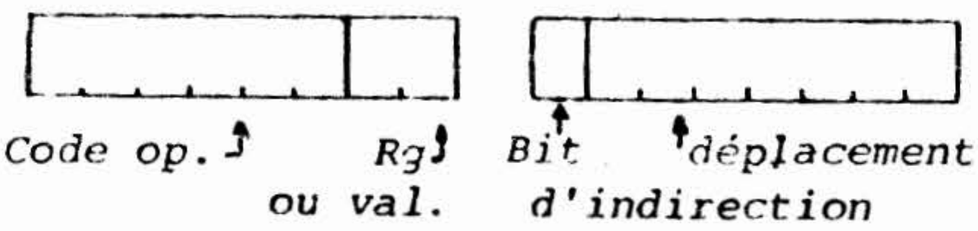


c) ADRESSAGE RELATIF

OPERANDES : - Registre ou valeur
- Adresse calculée comme un déplacement

OCCUPATION MEMOIRE : 2 octets

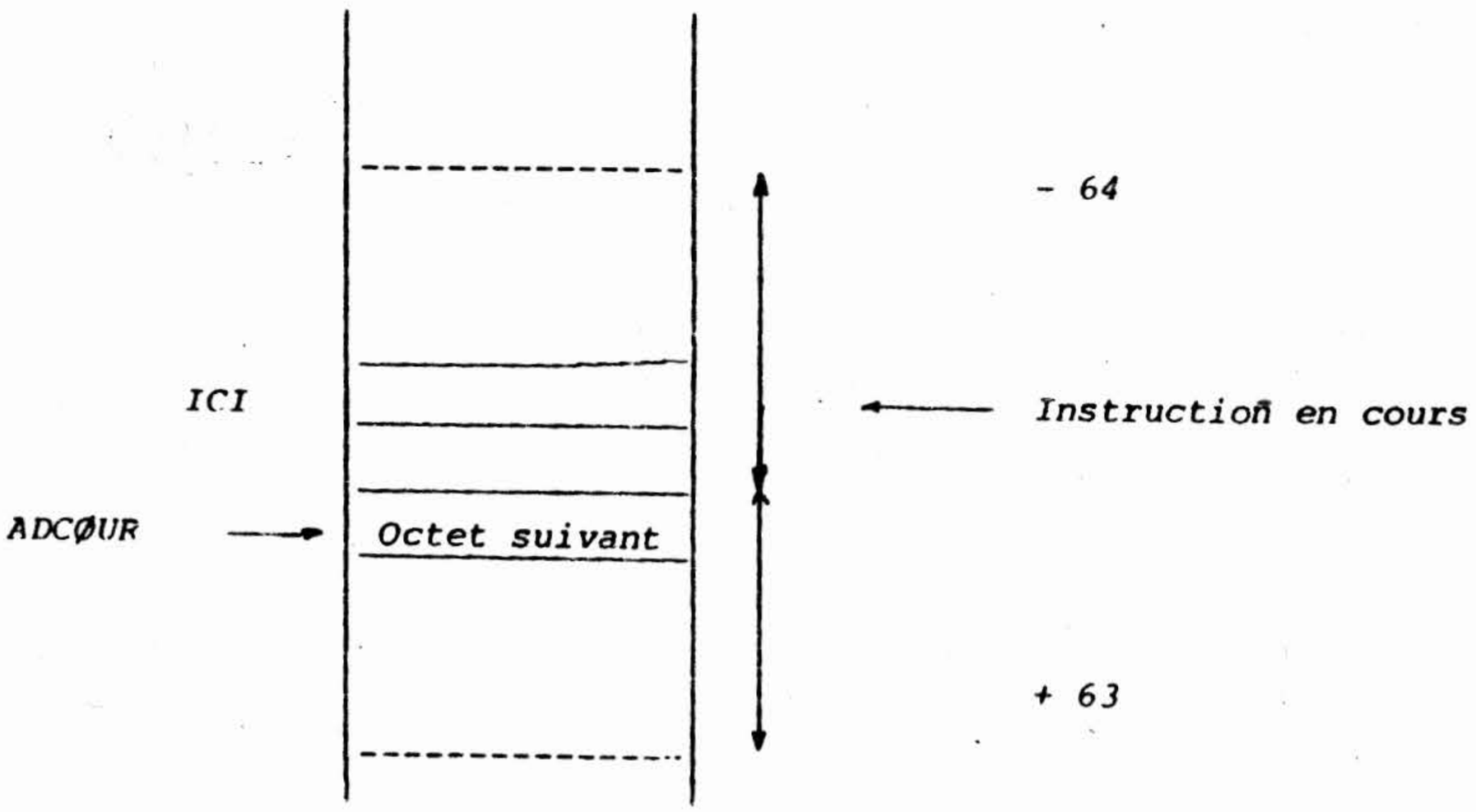
CODE BINAIRE :



DESCRIPTION

L'adresse est calculée en effectuant un déplacement de -64 à +63 octets par rapport à l'octet suivant l'instruction exécutée . On appelle adresse courante (ADCOUR) l'adresse de la prochaine instruction à exécuter, c'est à dire, celle qui suit l'instruction en cours d'exécution . En général c'est le contenu du compteur ordinal ou Program Counter (PC) .

Indirection : possible



EXEMPLES :
---

1) Branchement en avant si code condition = 2

Adr. hexa	Adr. symbolique	Instruction	
009C		BCTR,2	ADR2
→ 009E		ADDI,R3	H'34'
00A0		LODZ	R3
→ 00A1	ADR2	ADDR,R1	TAB

Adresse courante : 009E
Adresse de l'instruction à se brancher : 00A1
Déplacement : 00A1 - 009E = 03

En pratique, pour trouver la valeur du déplacement, compter en hexa le nombre d'octets en commençant par celui qui vient juste après l'octet de déplacement et en finissant juste avant l'instruction à se brancher . Par exemple ici, il faut compter 3 adresses : 009E, 009F et 00A0 .

BCTR , 2 ADR2

Code binaire : 0 0 0 1 1 0 1 0 | 0 0 0 0 0 0 1 1

Code hexa : 1A 03

2) Branchement en arrière si code condition = 2

Adr. hexa	Adr. Symbolique	Instruction	
→ 0056	ADR1	LODI,R1	10
0058		LODZ	R1
0059		SUBI,R3	10
005B		SUBR,R3	X
005D		LODZ	R3
005E		BCTR,2	ADR1
→ 0060			

Adresse courante : 0060
Adresse de l'instruction à se brancher : 0056
Déplacement : 0056 - 0060 = -0A

Pour exprimer un nombre binaire négatif, on utilise la règle du complément à 2 .

Expression en complément à 2 sur 7 bits de : -0A

0A	0 0 0 0 1 0 1 0
C1(0A) (Complément à 1)	1 1 1 1 0 1 0 1
+1	1 1 1 1 0 1 1 0
C2(0A) = -0A	1 1 1 0 1 1 0 = H "76"

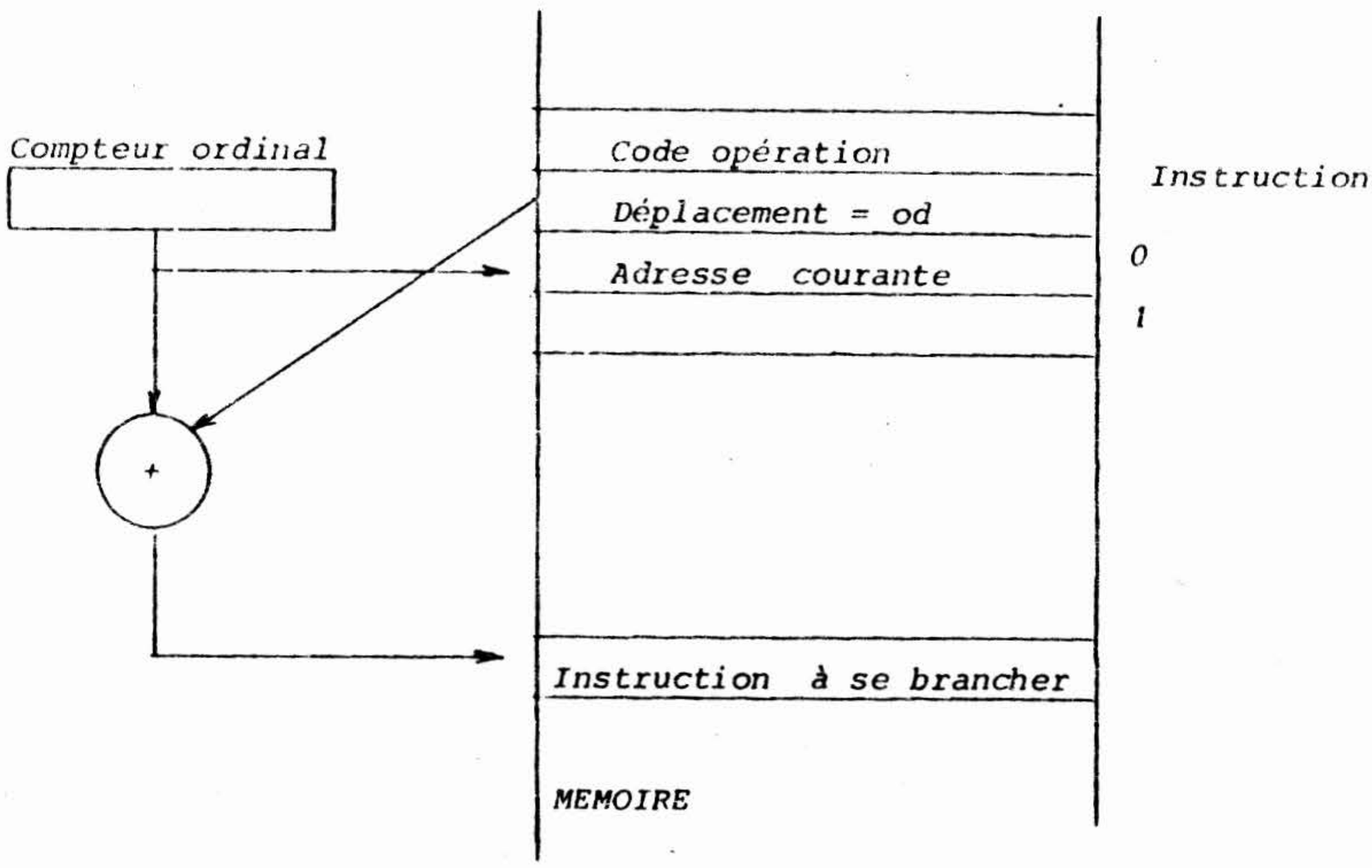
En pratique, pour trouver la valeur du déplacement, compter (en décimal, puis convertir en hexa) le nombre d'octets en commençant par l'octet de déplacement et en finissant sur l'instruction à se brancher . Puis transformer en complément à 2 .

BCTR,2 *ADR1*

Code binaire : | 0 0 0 1 1 0 1 0 | | 0 1 1 1 0 1 1 0 |

Code hexa : 1A 76

SCHEMA



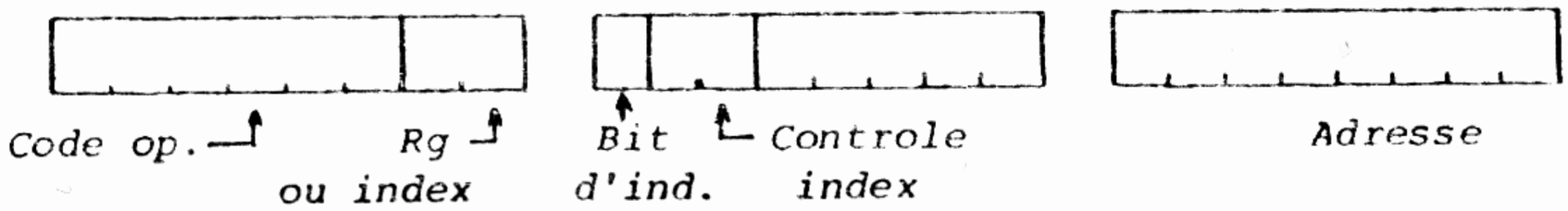
Adressage Relatif

d) ADRESSAGE ABSOLU (Non Branchements)

OPERANDES : - Registre ou Registre 0
- Adresse
- Rien ou Registre index

OCCUPATION MEMOIRE : 3 octets

CODE BINAIRE :



DESCRIPTION ----------
Ces instructions permettent d'adresser une donnée située dans la même page (c'est à dire à l'intérieur des 8K octets) .
L'adresse est codée sur 13 bits .
(2^{13} = 8192)

Indirection : possible

Indexation : possible

EXEMPLES
----

1) Adressage absolu : chargement dans R3

LODA,R3 TAB R3←(TAB) adr. TAB = 53B

Code binaire : 0 0 0 0 1 1,1 1 | 0,0 0,0 0 1 0 1 | 0 0 1 1 1 0 1 1

Code hexa : OF 05 3B

2) Adressage absolu indirect : chargement dans R3

LODA,R3 * TAB R3←((TAB))

Code binaire : 0 0 0 0 1 1,1 1 | 1,0 0,0 0 1 0 1 | 0 0 1 1 1 0 1 1

A l'adresse de TAB se trouve l'adresse de la valeur à charger dans R3

Code hexa : 0F 85 3B

3) Adressage absolu indexé avec incrémentation : addition avec R0 (implicite)

ADDA,R0 TAB,R2,+ R0←(R0)+(TAB+(R2)+1)

Code binaire : 1 0 0 0 1 1,1 0 | 0,0 1,0 0 1 0 1 | 0 0 1 1 1 0 1 1

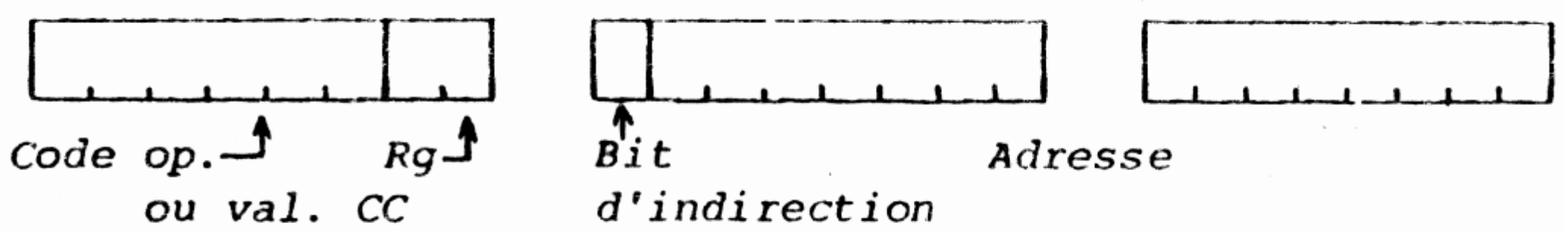
Code hexa : 8E 25 3B

e) ADRESSAGE ABSOLU (Branchements)

OPERANDES : - Registre ou valeur code condition
- Adresse

OCCUPATION MEMOIRE : 3 octets

CODE BINAIRE :



DESCRIPTION

Ces instructions permettent un branchement sur tout l'espace adressable, c'est à dire 32K octets .
L'adresse est codée sur 15 bits .
($2^{15} = 32768$)

Indirection : possible

EXEMPLES
---

1) Branchement si code condition = 0

BCTA,0	SORT	Si CC=0 aller à SORT adr. SORT = H'31F'

Code binaire : 0 0 0 1 1 1,0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 1 1 1 1 1

Code hexa : 1 C03 1F

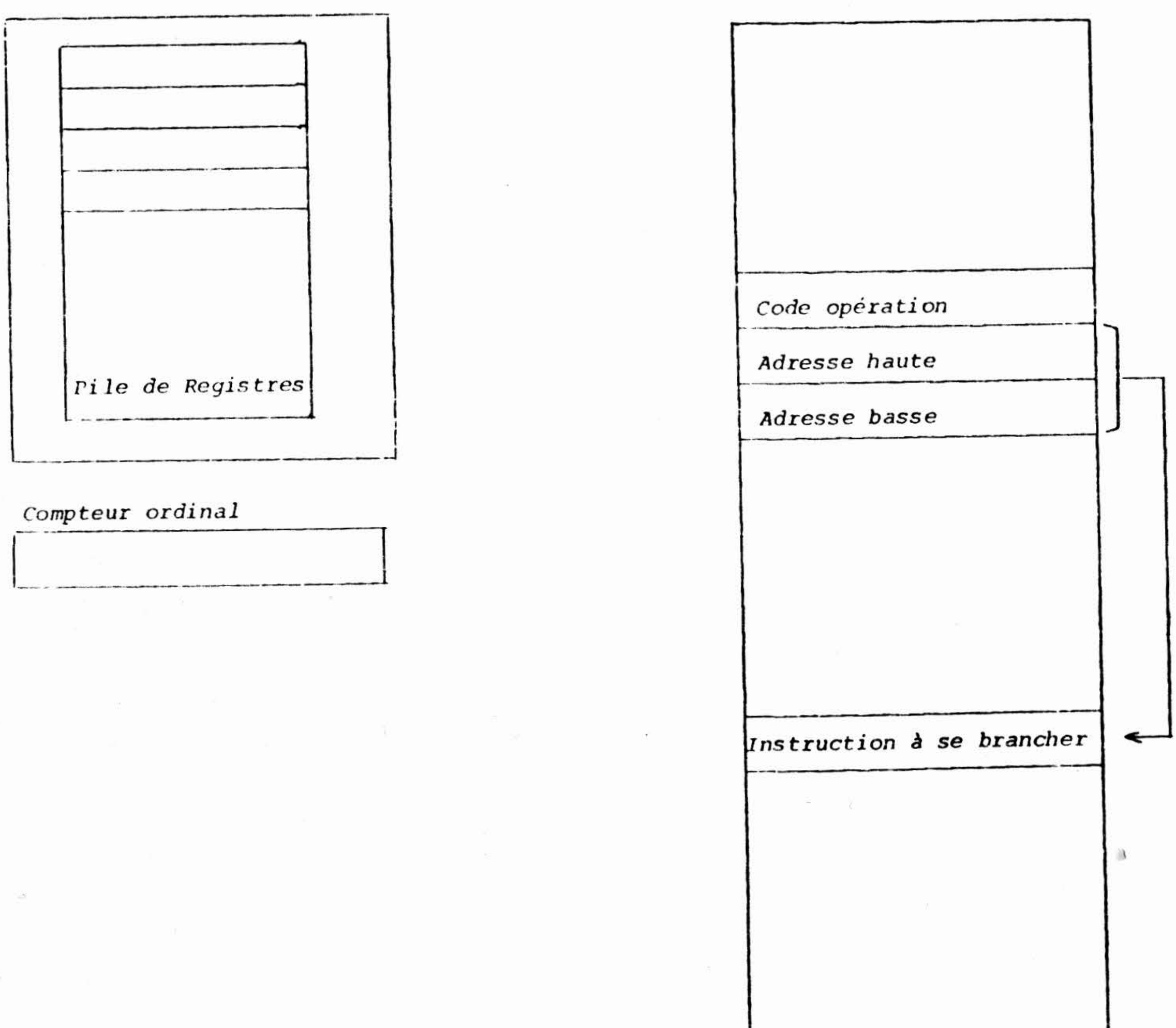
2) Branchement si code condition ≠ 2 avec indirection

BCFA,2	* TEST	Si CC≠2 aller à l'adresse contenue dans TEST . adr. TEST = H'40D'

Code binaire : 1 0 0 1 1 1,1 0 | 1 0 0 0 0 1 0 0 | 0 0 0 0 1 1 0 0

Code hexa : 9E 84 OD

SCHEMA



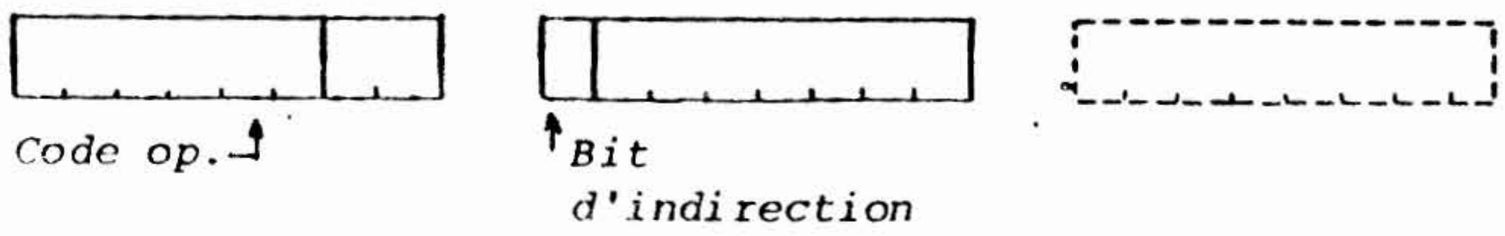
Adressage absolu

f) ADRESSAGE INDIRECT

OPERANDES : Les mêmes qu'avec l'adressage relatif ou l'adressage absolu

OCCUPATION MEMOIRE : 2 (adr. relatif) 3 (adr. absolu)

CODE BINAIRE :



DESCRIPTION

L'adresse de l'opérande se trouve à l'adresse désignée par l'instruction .
L'indirection est représentée :

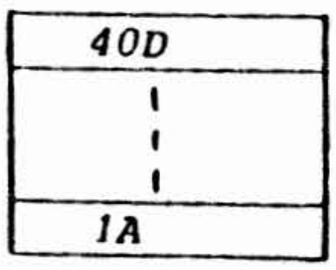
- par une étoile (*) dans les instructions symboliques
- par le bit d'indirection à 1 dans le code binaire

Indexation : possible avec l'adressage absolu

EXEMPLES
---

1) Chargement dans R1 du contenu de la mémoire dont l'adresse est dans VAR

LODA,R1 * VAR R1 ← ((VAR)) adr. VAR=H'400'



R1 ← (40D)

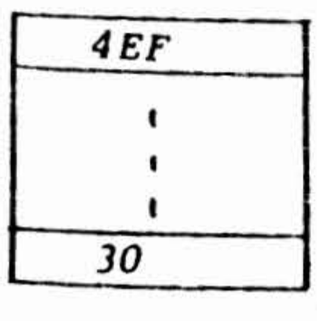
R1 1A

Code binaire : 0 0 0 0 1 1,0 1 | 1,0 0 0 0 0 1 0 0 | 0 0 0 0 0 0 0 0

Code hexa : 0D 84 00

2) Addition avec R2 du contenu de la mémoire dont l'adresse est dans VAR

ADDA,R2 * VAR R2 ← (R2) + ((VAR))



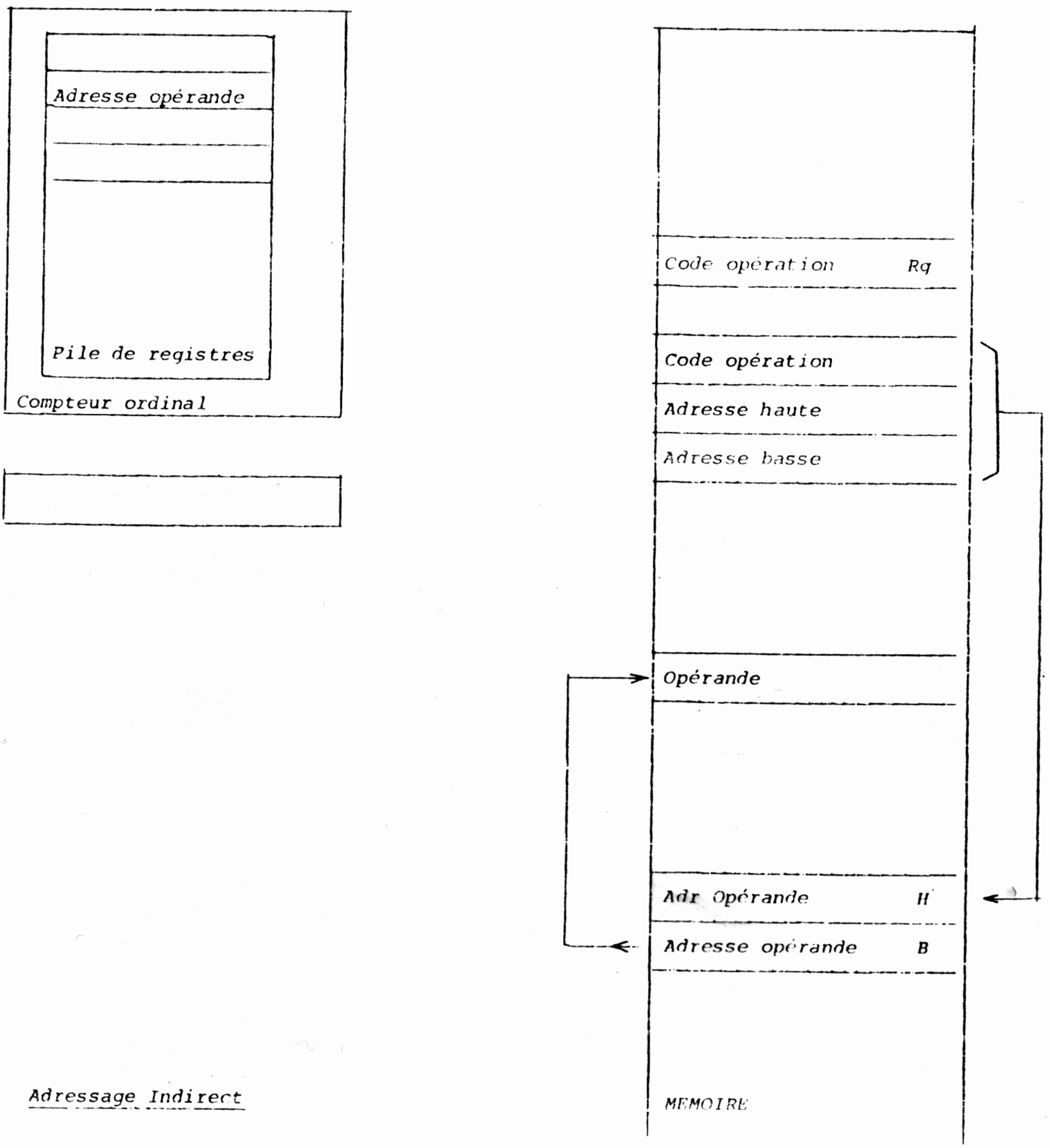
R2 ← (R2) + (4EF)

R2 ← (R2) + 30

Code binaire : 1 0 0 0 1 1,1 0 | 1,0 0 0 0 0 1 0 0 | 0 0 0 0 0 0 0 0

Code hexa : 8E 84 00

SCHEMA



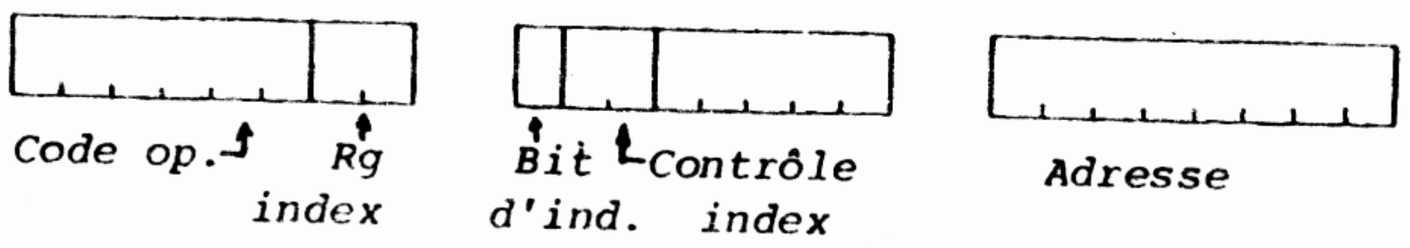
Adressage Indirect

g) ADRESSAGE INDEXE (Employé avec adr. absolu)

OPERANDES : - Registre 0 obligatoirement
- Adresse
- Registre index

OCCUPATION MEMOIRE : 3 octets

CODE BINAIRE :



DESCRIPTION

L'adresse de l'opérande est calculée en ajoutant le contenu du registre d'index à l'adresse effective contenue dans l'instruction .

Les deux bits du contrôle d'index indiquent :

- 0 0 pas d'indexation
- 0 1 indexation avec pré-incrément
- 1 0 indexation avec pré-décrement
- 1 1 indexation simple

Indirection : possible

EXEMPLES
---

1) Chargement dans R0 du contenu de l'adresse TAB + (R2)

LODA,R0 TAB,R2 R0 ← (TAB +(R2))

Code binaire : 0 0 0 0 1 1 1 0 | 0 1 1 0 0 0 0 0 | 0 0 0 0 1 0 1 0

Code hexa : 0E 60 0A

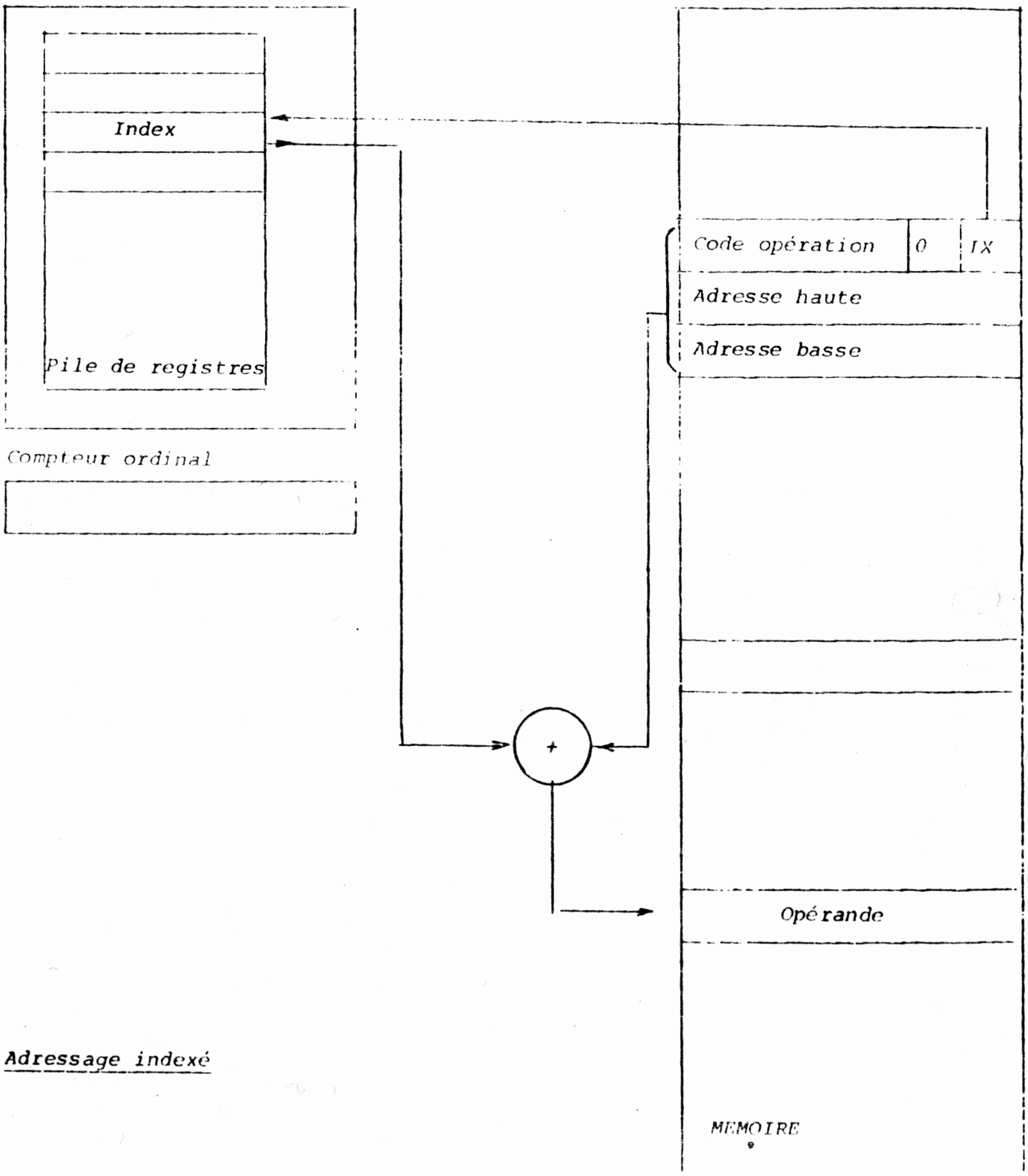
2) Addition avec R0 du contenu de l'adresse TAB = (R2) - 1

ADDA,R0 TAB,R2,- R0 (TAB +(R2) - 1)

Code binaire : 1 0 0 0 1 1 1 0 | 0 1 0 0 0 0 0 0 | 0 0 0 0 1 0 1 0

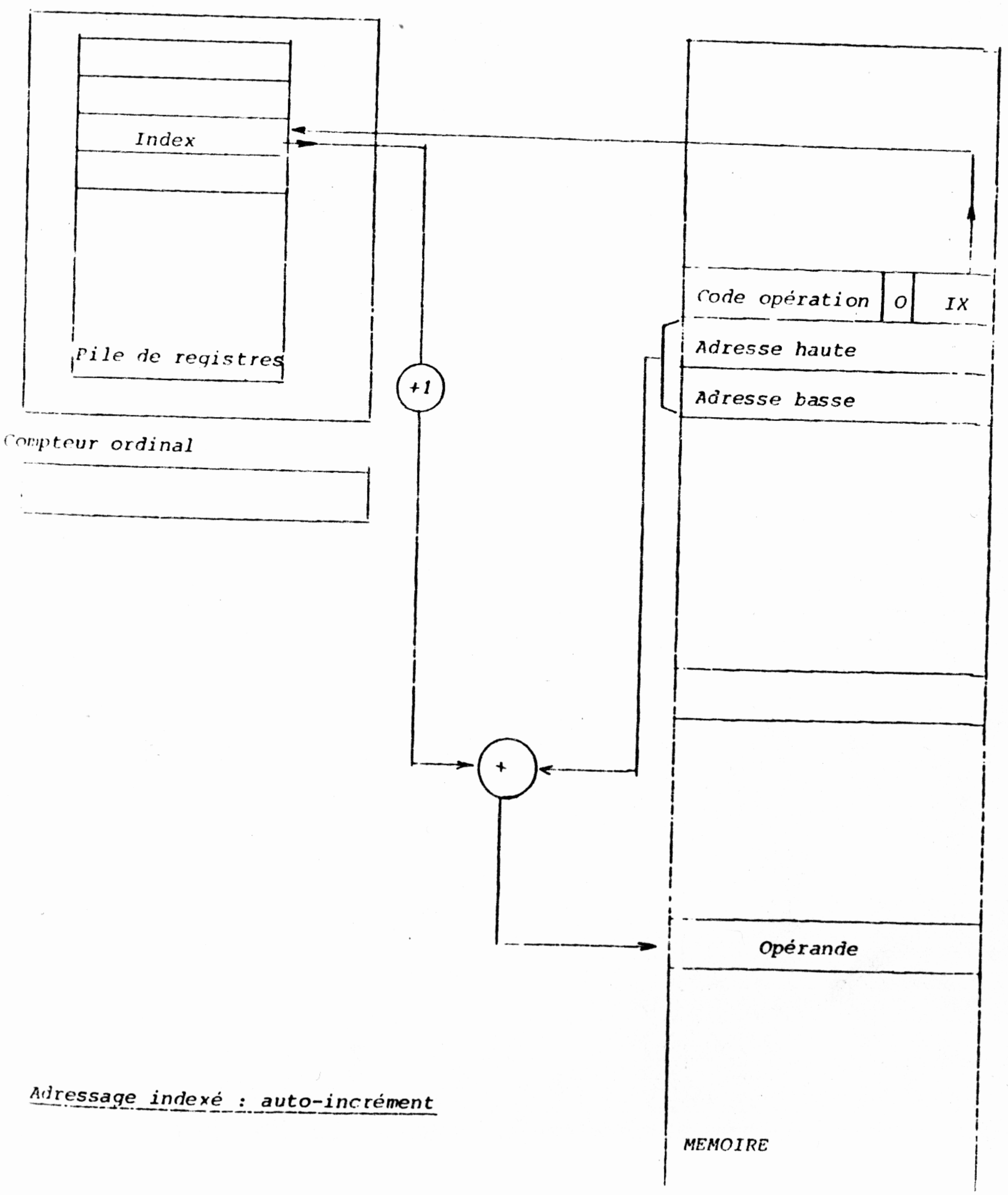
Code hexa : 8D 40 0A

SCHEMA



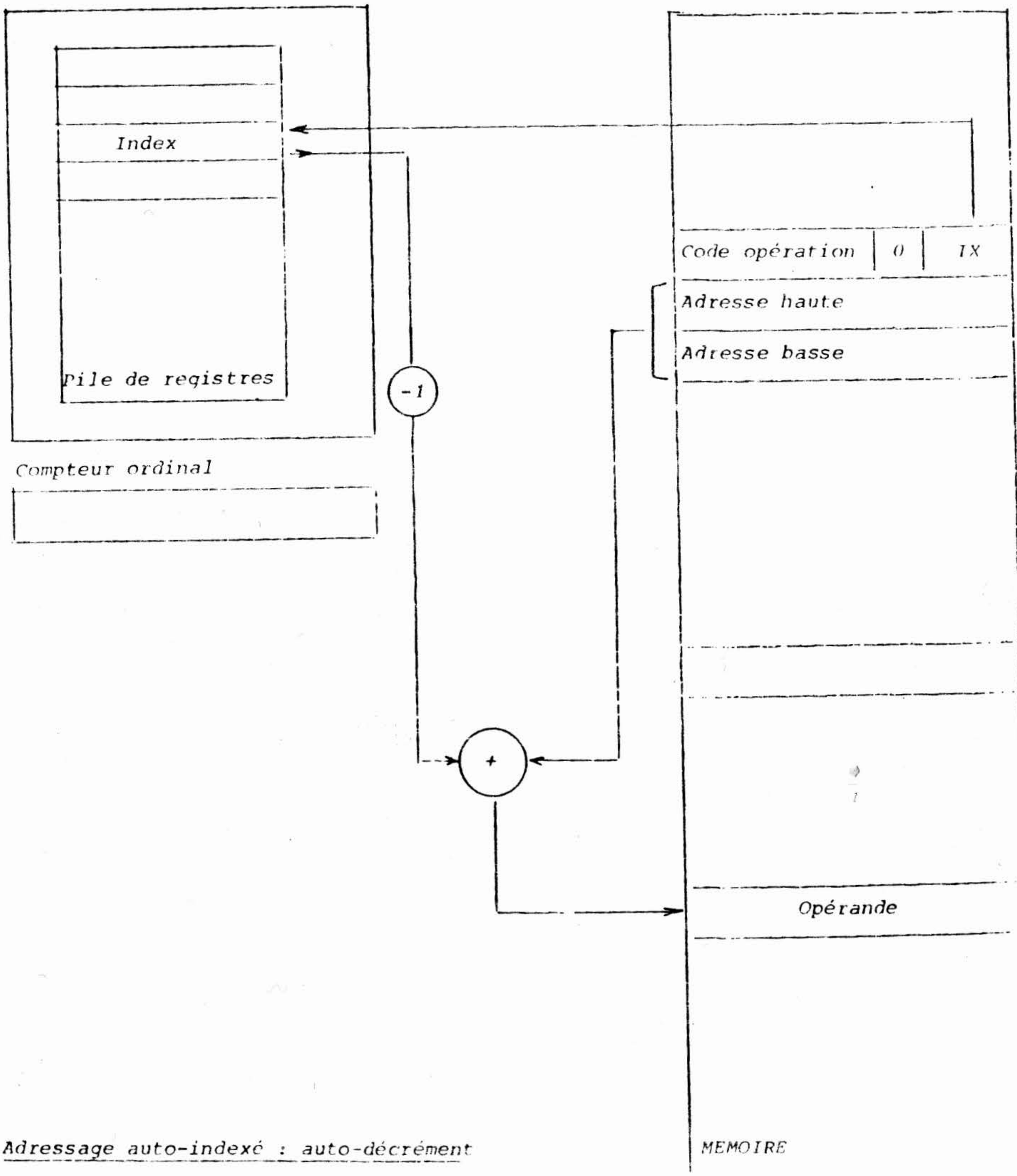
Adressage indexé

SCHEMA



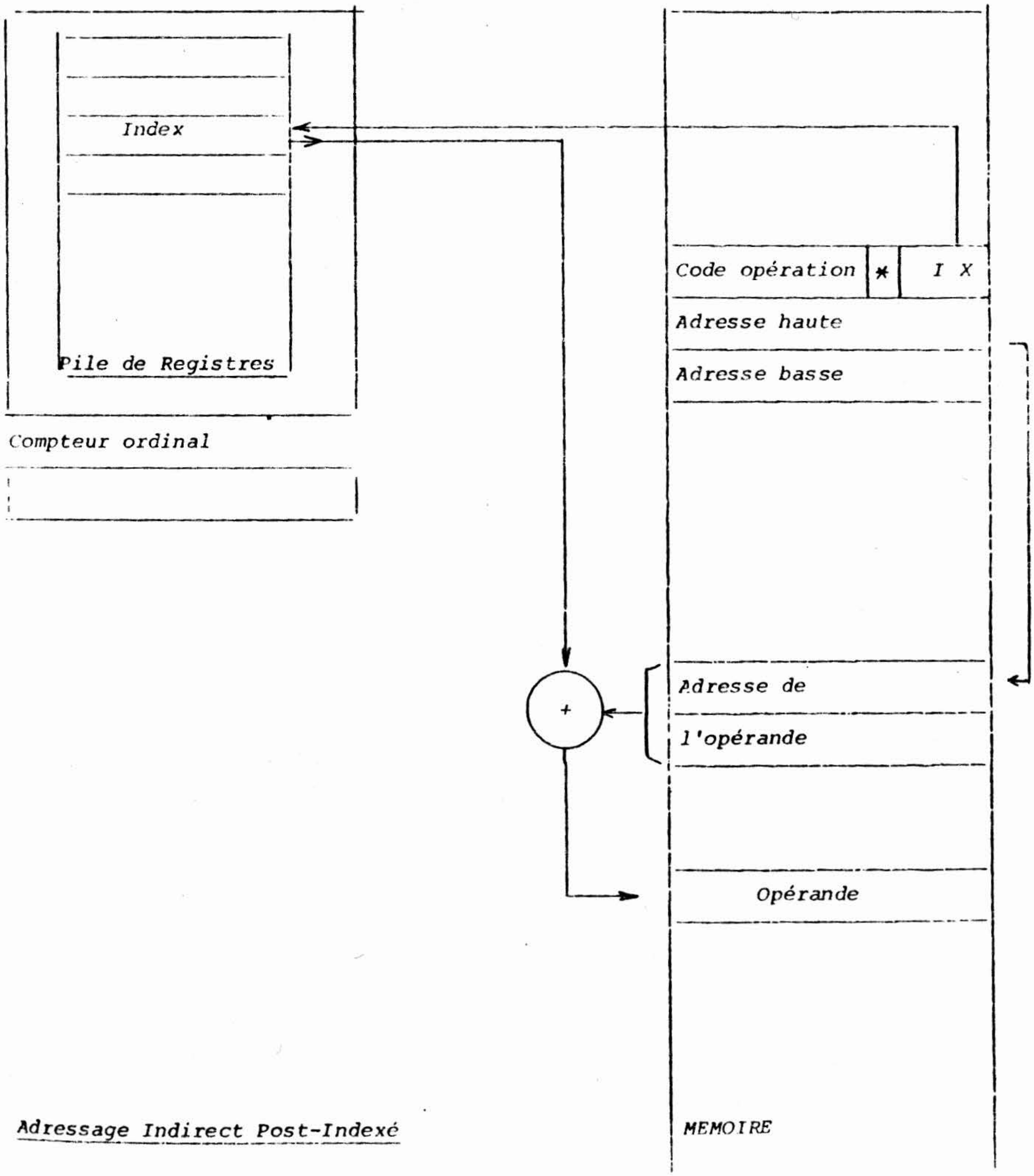
Adressage indexé : auto-incrément

SCHEMA



Adressage auto-indexé : auto-décrément

SCHEMA



Adressage Indirect Post-Indexé

INSTRUCTIONS DE STOCKAGE

ET

DE CHARGEMENT

1 - CHARGEMENT

LOD

CHARGEMENT D'UN REGISTRE AVEC UNE VALEUR

RX ◄—— VALEUR

2 - STOCKAGE

STR

STOCKER LE CONTENU D'UN REGISTRE

RX ——►

CHARGEMENT REGISTRE ZERO (LOAD REGISTER ZERO)

MNEMONIQUE : LODZ R
NOMBRE DE CYCLES : 2 (le nombre de cycles indique la durée de l'instruction : 1 cycle = 1,1 microseconde)
TYPE D'ADRESSAGE : Registre

DESCRIPTION

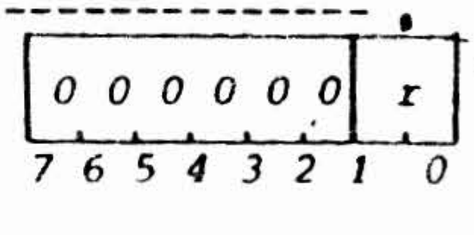
Cette instruction simple longueur (elle ne prend qu'un octet) charge le registre zéro avec le contenu du registre spécifié

REGISTRES DU PROCESSEUR AFFECTES CC

Registre zéro	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

Le registre peut prendre les valeurs arithmétiques H"00" à H"FF", c'est-à-dire en décimal de + 127 à - 128 : 0 à 127 correspond à 7F ; (-1) à (-128) correspond à FF à 80 (nombres en complément à 2) .

CODE BINAIRE DE L'INSTRUCTION



STOCKAGE REGISTRE ZERO (STORE REGISTER ZERO)

MNEMONIQUE : STRZ R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

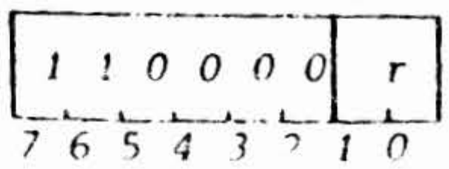
DESCRIPTION

Cette instruction simple longueur charge le contenu du registre zéro dans le registre R spécifié .
On ne peut spécifier le registre zéro puisque ce code opération est réservé au NOP. (Code hexa = C0)

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



CHARGEMENT IMMEDIAT (LOAD IMMEDIATE)

MNEMONIQUE : LODI,R V
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Immédiat

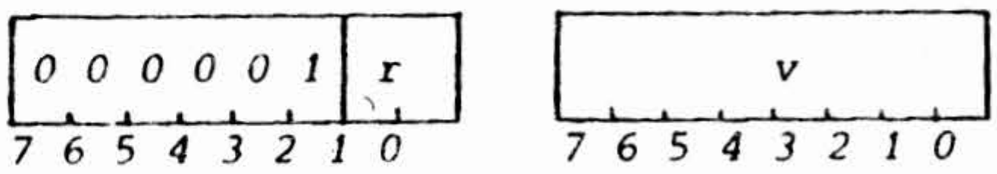
DESCRIPTION

Cette instruction double longueur (2 octets) charge le registre spécifié R avec le contenu du second octet de l'instruction .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



CHARGEMENT RELATIF (LOAD RELATIVE)

MNEMONIQUE : LODR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : relatif

DESCRIPTION

- Cette instruction double longueur charge le registre spécifié avec un octet de données . Cet octet de données est cherché à l'emplacement de la mémoire référencée par l'adresse de l'instruction suivante à laquelle on ajoute la valeur du déplacement a .
  Le contenu préalable du registre R est perdu .
- Si le bit I est positionné, alors il s'agit d'une indirection .
  Dans ce cas on charge dans le registre R le contenu de la case mémoire pointée par le contenu de la case mémoire pointée par le contenu de celle située à l'adresse Adcour + a .

EXEMPLE

Adresse hexa	Code op.	Mnémonique
0910	0897	LODR,RO * 17
0912	⋮	⋮
⋮	⋮	⋮
→ 0919	0A28	Adresse 0A28
⋮	⋮	⋮
0A28	67	Data 67

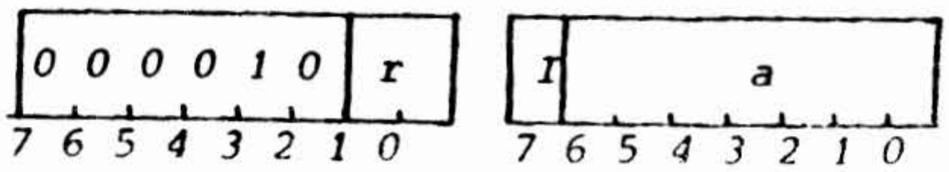
Ici Adcour = 0912 a = 17 d'où 0912 + 17 = 0919

Dans l'exemple ci-dessus c'est la donnée H'67' qui sera chargée dans le registre Ro .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



STOCKAGE RELATIF (STORE RELATIVE)

MNEMONIQUE : STRR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

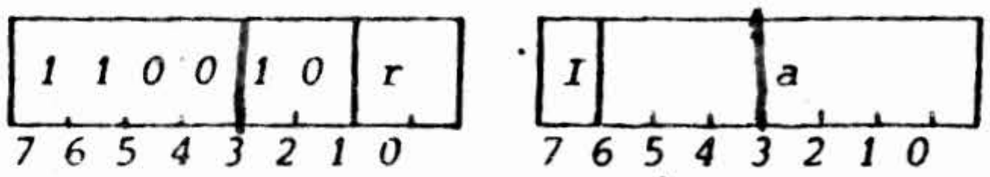
DESCRIPTION

- Cette instruction double longueur range le contenu du registre spécifié à l'emplacement de la mémoire pointée par l'adresse effective .
  Cette adresse effective est obtenue en ajoutant à l'adressé de l'instruction suivante, la valeur du déplacement a .

- De la même manière que pour le LODR, on peut spécifier une indirection .

REGISTRES DU PROCESSEUR AFFECTES : aucun

CODE BINAIRE DE L'INSTRUCTION



CHARGEMENT ABSOLU (LOAD ABSOLUTE)

MNEMONIQUE : LODA,R (*) a (,x)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

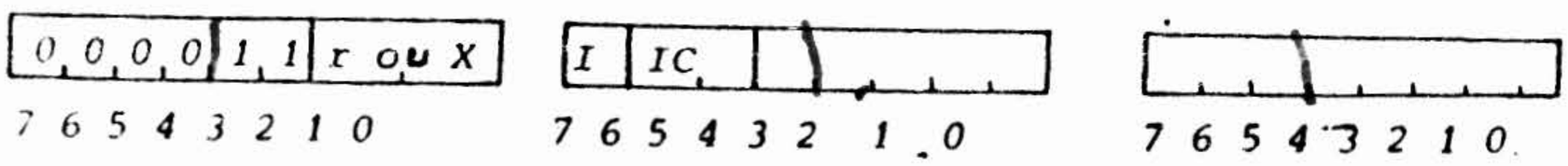
DESCRIPTION

- Cette instruction triple longueur (3 octets) charge le registre spécifié avec un octet de données .
  L'adresse de cet octet de données est fournie par les bits 0-4 octet 1 et les bits 7-0 octet 2 de l'instruction .
- Si on spécifie adressage indexé, le registre spécifié devient registre d'index et la destination est implicitement le registre zéro .
- On peut également spécifier l'indirection .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



STOCKAGE ABSOLU (STORE ABSOLUTE)

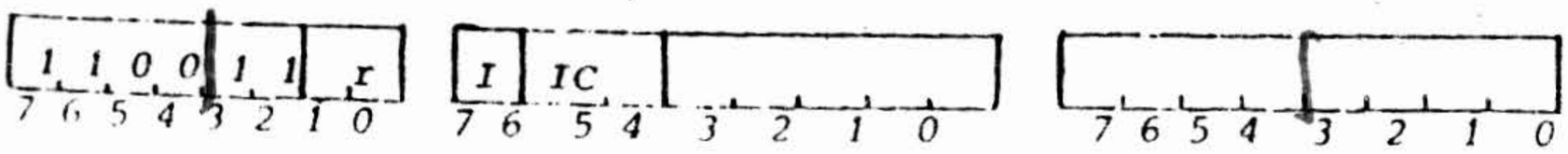
MNEMONIQUE : STRA,R (*) a (,X)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

- Cette instruction triple longueur transfère le contenu du registre R à l'emplacement mémoire pointé par le champ a .
- De la même manière que pour le LODA on peut spécifier l'indexation ou l'indirection .
- Si l'indexation est spécifiée le registre émetteur devient implicitement le registre zéro .

REGISTRES DU PROCESSEUR AFFECTES : aucun

CODE BINAIRE DE L'INSTRUCTION



INSTRUCTIONS ARITHMETIQUES

ET

LOGIQUES

1 - ADDITION : ADD

L'addition consiste à effectuer l'opération suivante :

RX ← (RX) + VALEUR

La valeur peut être le contenu d'un autre registre, la valeur d'un octet de mémoire .

L'addition est faite avec ou sans carry selon la valeur du bit WC .

Si l'on fait des additions en chaine on positionnera :

WC = 1
C = 0

2 - SOUSTRACTION : SUB

La soustraction consiste à effectuer l'opération suivante :

RX ← (RX) -VALEUR

La valeur peut être le contenu d'un autre registre ou celle d'un octet de mémoire .

La soustraction est faite avec ou sans carry selon la valeur du bit WC .

Si WC = 1 on retranchera le COMPLEMENT DU CARRY donc si l'on soustrait en chaine on positionnera :

WC = 1
C = 1

Exemple :

WC = 1 C = 0

1A ⟶ Ro

Ro - OD → Ro

Ro = 1A - OD - 1 (= $\bar{C}$)

= OD - 1 = OC

WC = 1 C = 1

1A ⟶ Ro

Ro - OD → Ro

Ro = 1A - OD - 0 (= $\bar{C}$)

= OD - 0 = OD

3 - AJUSTEMENT DECIMAL : DAR

Utilisé pour effectuer des calculs en décimal;si l'on additionne ou l'on soustrait 2 octets en décimal on fera à la suite un DAR pour être certain que le résultat est signifiant .

4 - ET LOGIQUE : AND

RX ← RX ET VALEUR

A ET B

TABLES DE VERITE

A \ B	1	0
1	1	0
0	0	0

5 - OU INCLUSIF : IOR

RX ← RX OU VALEUR

A OU B

A \ B	1	0
1	1	1
0	1	0

6 - OU EXCLUSIF : EOR

RX ← RX ⊕ VALEUR

A ⊕ B

A \ B	1	0
1	0	1
0	1	0

ADDITION AU REGISTRE ZERO (ADD TO REGISTER ZERO)

MNEMONIQUE : ADDZ R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

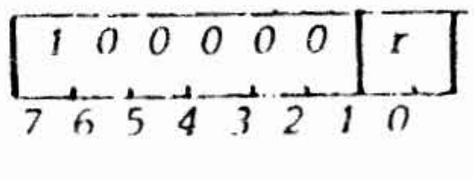
DESCRIPTION

Cette instruction simple longueur effectue l'addition binaire du **registre** spécifié avec le registre zéro . Le résultat est mis dans le registre zéro .
Le contenu du registre R reste inchangé .
L'opération peut s'effectuer avec ou sans carry .

REGISTRES DU PROCESSEUR AFFECTES C, CC, IDC, OVF

Registre zéro	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



SOUSTRACTION AU REGISTRE ZERO (SUBSTRACT FROM REGISTER ZERO)

MNEMONIQUE : SUBZ R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

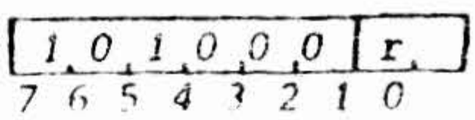
DESCRIPTION

Cette instruction simple longueur effectue la soustraction du registre R ou registre zéro . Le résultat de cette opération est mis dans le registre zéro .
Cette opération est réalisée en prenant le complément à 2 du registre R et en l'additionnant au registre R0 .
Cette opération peut s'effectuer avec ou sans le bit CARRY du PSW .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre zéro	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



ADDITION IMMEDIATE (ADD IMMEDIATE)

MNEMONIQUE : ADDI,R V
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Immédiat

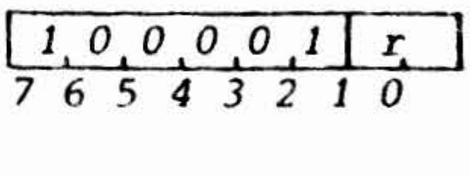
DESCRIPTION

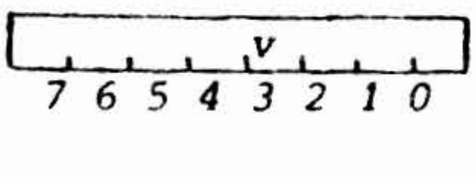
Cette instruction double longueur effectue la somme de la valeur spécifiée avec le registre indiqué . La somme trouvée est rangée dans le registre indiqué . Cette opération peut être effectuée avec ou sans carry .

REGISTRES DU PROCESSEUR AFFECTES C, CC, IDC, OVF

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION





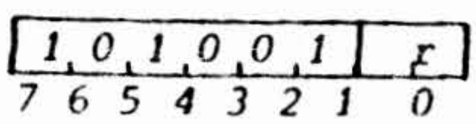
SOUSTRACTION IMMEDIATE (SUBSTRACT IMMEDIATE)

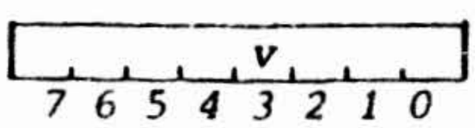
MNEMONIQUE : SUBI,R V
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

Cette instruction double longueur effectue la soustraction au registre indiqué de la valeur V . Le résultat de la soustraction est rangé dans le registre R . Cette opération peut s'effectuer avec ou sans le bit CARRY du PSW .

CODE BINAIRE DE L'INSTRUCTION





ADDITION RELATIVE (ADD RELATIVE)

MNEMONIQUE : ADDR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

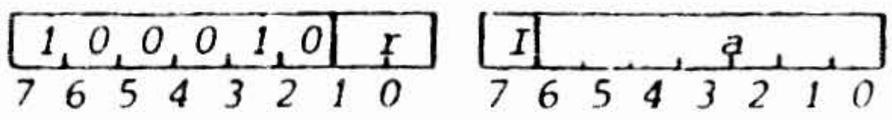
DESCRIPTION

- Cette instruction double longueur effectue la somme du contenu de la case mémoire pointée par l'adresse effective avec le registre spécifié .
- L'adresse effective étant calculée comme la somme de l'adresse de l'instruction suivante avec la valeur a .
- Le résultat de la somme des deux opérandes remplace le contenu du registre .
- On peut spécifier l'indirection .
- On peut effectuer cette opération avec ou sans carry .

REGISTRES DU PROCESSEUR AFFECTES C, CC, IDC, OVF

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



SOUSTRACTION RELATIVE (SUBSTRACT RELATIVE)

MNEMONIQUE : S U BR,R ( ) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

- Cette instruction double longueur soustrait le contenu de la case mémoire pointée par l'adresse effective au contenu du registre R spécifié, le résultat est rangé dans R .
- L'adresse effective étant calculée comme la somme de l'adresse de l'instruction suivante avec la valeur du déplacement a
- On peut spécifier l'indirection
- La soustraction peut être effectuée avec ou sans le bit CARRY du PSW .

REGISTRES DU PROCESSEUR AFFECTES C, CC, IDC, OVF

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION

ADDITION ABSOLUE (ADD ABSOLUTE)

MNEMONIQUE : ADDA,R (*) a (, X)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

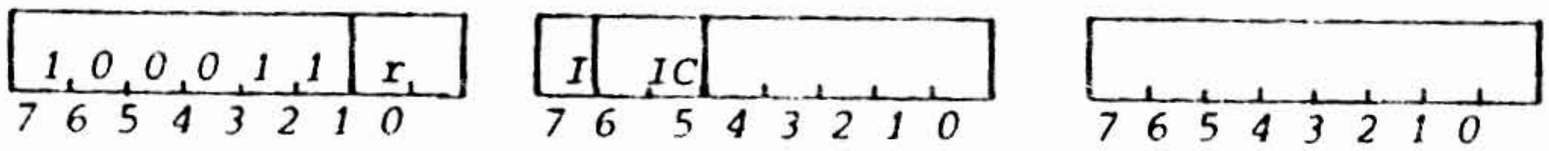
DESCRIPTION

- Cette instruction triple longueur effectue la somme du contenu du registre R avec le contenu de la case mémoire pointée par l'adresse effective .
  L'adresse effectivc étant donnée par le champ a .
  On peut spécifier indirection ou indexation ou les deux .
  Si on indique l'indexation, le résultat est rangé implicitement dans le registre zéro, sinon il est rangé dans le registre R spécifié .
  Cette opération peut être effectuée avec ou sans carry .

REGISTRES DU PROCESSEUR AFFECTES C, CC, IDC, OVF

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



SOUSTRACTION ABSOLUE (SUBSTRACT ABSOLUTE)

MNEMONIQUE : SUBA,R (*) a (, X)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

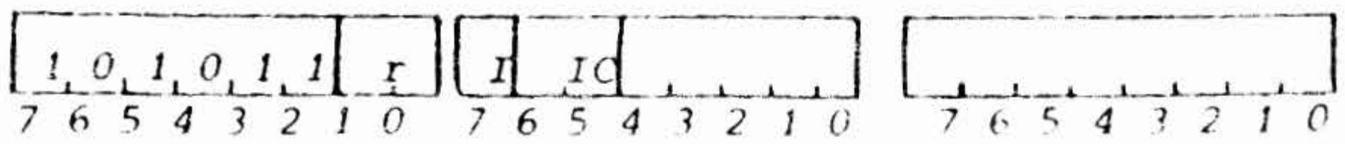
DESCRIPTION

Cette instruction triple longueur soustrait un octet de données au contenu du registre R . Le résultat est rangé dans R .
L'adresse de la donnée est donnée par le champ a .
On peut spécifier l'indirection ou l'indexation ou les deux .
Si on indique l'indexation, le registre zéro devient implicitement le registre opérande .
Cette soustraction peut s'effectuer avec ou sans le bit Carry du PSW .

REGISTRES DU PROCESSEUR AFFECTES C, CC, IDC, OVF

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



AJUSTEMENT DECIMAL (DECIMAL ADJUST REGISTER)

MNEMONIQUE : DAR,R
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Registre

DESCRIPTION

- Cette instruction de simple longueur permet d'opérer des opérations d'addition et de soustraction sur des nombres décimaux .
- Suivant les valeurs du carry et du carry intermédiaire la table de vérité ci-dessous indique la quantité que l'on ajoute au registre R . Les carry ne sont pas modifiés .
- On prendra garde à prendre le signe du résultat avant de faire un DAR car celui-ci opère sur des valeurs absolues .

Table : Carry C	Carry intermédiaire IDC	Quantité
0	0	H 'AA'
0	1	H 'A0'
1	1	H '00'
1	0	H '0A'

L'addition effectuée sur le registre R ne tient pas compte des retenues de cette addition .

Exemple : Si l'on effectue un DAR sur Ro = 57 on aura alors dans Ro :

pour C = 0 IDC = 0

```
57
AA
--
F1 ---> Ro
```

REGISTRES DU PROCESSEUR AFFECTES CC

CC contient une valeur non significative .

CODE BINAIRE DE L'INSTRUCTION

1	0	0	1	0	1	r	r
7	6	5	4	3	2	1	0

ADDITION DECIMALE DE DEUX OCTETS

1) Ajouter au premier octet H'66'
2) Ajouter le second octet au premier
3) Effectuer un DAR sur le second octet

Exemple : Ajout de R0 et de R1

```
ADDI,R0     H'66'
ADDZ        R1
DAR,R0
```

SOUSTRACTION DECIMALE DE DEUX OCTETS

1) Soustraire le second octet du premier
2) Effectuer un DAR sur le premier octet

Exemple : Soustraction R0 - R1

```
SUBZ        R1
DAR,R0
```

"ET" AU REGISTRE ZERO (AND TO REGISTER ZERO)

MNEMONIQUE : ANDZ R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

Cette instruction simple longueur effectue un ET logique entre le registre zéro et le registre R spécifié .
Le résultat de l'opération remplace le contenu du registre zéro . Le registre R demeure inchangé .
La table de vérité est la suivante :

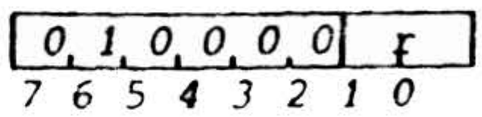
Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	0
1	1	1
1	0	0

Attention : il ne faut pas spécifier R0 comme registre ; le code opération 40 étant réservé pour l'instruction HALT

REGISTRES DU PROCESSEUR AFFECTES CC

Registre zéro	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"OU INCLUSIF" AU REGISTRE ZERO (INCLUSIVE OR TO REGISTER ZERO)

MNEMONIQUE : IORZ R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

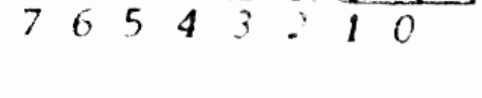
Cette instruction simple longueur effectue un ou inclusif logique entre le registre R et le registre zéro .
Le résultat de l'opération est rangé dans le registre zéro . Le contenu du registre R demeure inchangé .
La table de vérité est la suivante :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	1
1	1	1
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

	CC1	CC0
Registre zero		
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"OU EXCLUSIF" AU REGISTRE ZERO (EXCLUSIVE OR TO REGISTER ZERO)

MNEMONIQUE : EORZ R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

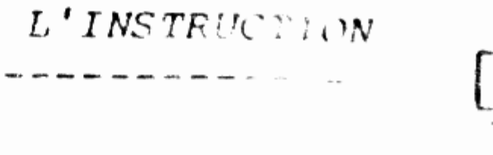
Cette instruction simple longueur effectue un ou exclusif entre le registre zéro et le registre R .
Le résultat est rangé dans le registre zéro ; le contenu du registre R demeure inchangé.
La table de vérité est la suivante :

Bit (0-7)	Bit(0-7)	Résultat
0	0	0
0	1	1
1	1	0
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

	CC1	CC0
Registre zéro		
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"ET" IMMEDIAT (AND IMMEDIATE)

MNEMONIQUE : ANDI,R V
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

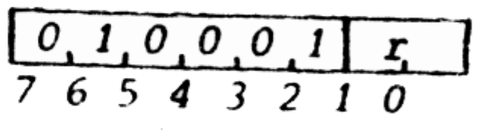
Cette instruction double longueur effectue un ET logique entre le registre R et la valeur V (2ème mot de l'instruction) .
Le résultat est rangé dans R .
La table de vérité est la suivante .

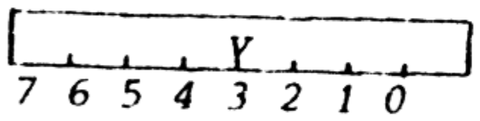
Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	0
1	1	1
1	0	0

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION





"OU INCLUSIF" IMMEDIAT (INCLUSIVE OR IMMEDIATE)

MNEMONIQUE : IOR,R V
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

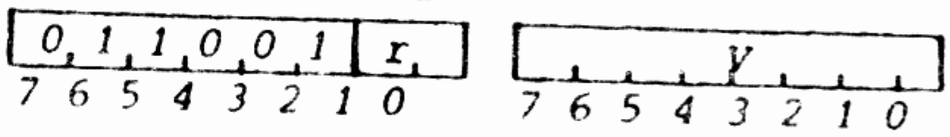
Cette instruction double longueur effectue un OU inclusif logique entre le registre R et la valeur V . Le résultat de cette opération est rangé dans le registre R . La table de vérité est la suivante :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	1
1	1	1
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"OU EXCLUSIF" IMMEDIAT (EXCLUSIVE OR IMMEDIATE)

MNEMONIQUE : EORI,R V
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

Cette instruction double longueur effectue un OU exclusif logique entre le registre R et la valeur V . Le résultat est rangé dans le registre R . La table de vérité est la suivante :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	1
1	1	0
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION

"ET" RELATIF (AND RELATIVE)

MNEMONIQUE : ANDR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

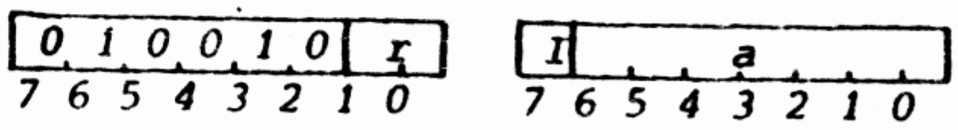
Cette instruction double longueur effectue un ET logique entre le registre R et le contenu de la case mémoire pointée par l'adresse effective .
L'adresse effective est ici la somme de l'adresse de l'instruction suivante avec la valeur du déplacement a .
Le résultat est rangé dans le registre R . On peut spécifier l'indirection avec cette opération .
La table de vérité est la suivante :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	0
1	1	1
1	0	0

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"OU INCLUSIF" RELATIF (INCLUSIVE OR RELATIVE)

MNEMONIQUE : IORR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

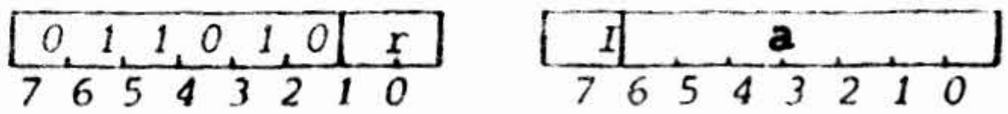
Cette instruction double longueur effectue un OU inclusif logique entre le registre R et le contenu de la case mémoire pointée par l'adresse effective .
Le résultat est rangé dans le registre R .
L'adresse effective est calculée comme la somme de l'instruction suivante et de la valeur du déplacement a . On peut spécifier l'indirection .
La table de vérité est la suivante :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	1
1	1	1
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"OU EXCLUSIF" RELATIF (EXCLUSIVE OR RELATIVE)

MNEMONIQUE : EORR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

Cette instruction double longueur effectue un OU exclusif logique entre le registre R et le contenu de la case mémoire pointée par l'adresse effective .
Le résultat est rangé dans R .
L'adresse effective est la somme de l'adresse de l'instruction suivante avec a .
On peut spécifier l'indirection .
La table de vérité est la suivante :

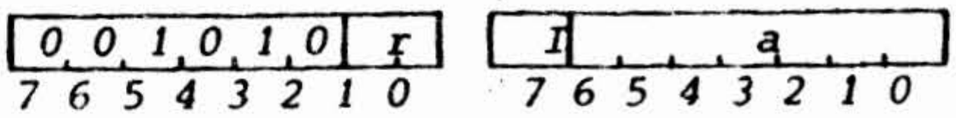
Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	1
1	1	0
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1

Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"ET" ABSOLU (AND ABSOLUTE)

MNEMONIQUE : ANDA,R (*) a (,X)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

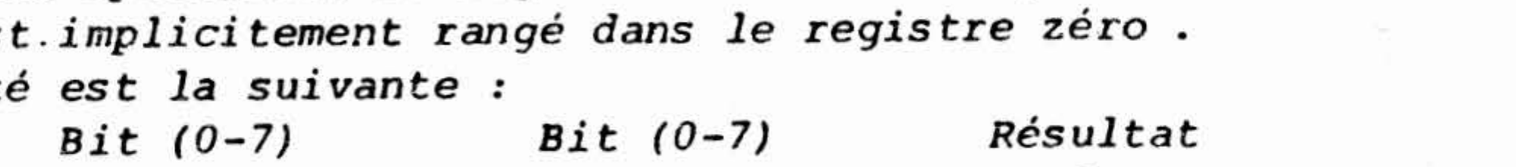
Cette instruction triple longueur effectue un ET logique entre le registre R et le contenu de la case mémoire pointée par l'adresse effective .
Le résultat est rangé dans le registre R .
L'adresse effective est donnée par la valeur a .
On peut également spécifier l'indirection ou l'indexation ou les deux .
Si l'indexation est spécifiée, le registre R sert de registre d'index et le résultat de l'opération est implicitement rangé dans le registre zéro .
La table de vérité est la suivante :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	0
1	1	1
1	0	0

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



"OU INCLUSIF" ABSOLU (INCLUSIVE OR ABSOLUTE)

MNEMONIQUE : IORA,R (*) a (,X)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

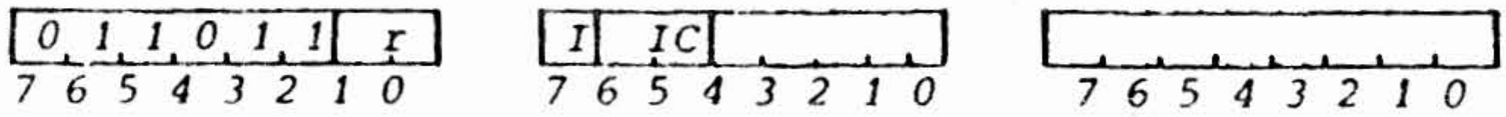
Cette instruction de triple longueur effectue un OU inclusif entre le contenu du registre R et le contenu de la case mémoire pointée par l'adresse effective .
Le résultat de l'opération est rangé dans le registre R .
L'indirection ou l'indexation ou les deux peuvent être spécifiées .
Dans le cas où l'indexation est spécifiée, c'est le registre zéro qui devient destinataire du résultat .
La table de vérité est la suivante :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	1
1	1	1
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

	CC1	CC0
Registre zéro	0	1
Positif	0	0
Nul	1	0
Négatif		

CODE BINAIRE DE L'INSTRUCTION



"OU EXCLUSIF" ABSOLU (EXCLUSIVE OR ABSOLUTE)

MNEMONIQUE : EORA,R (*) a (,X)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

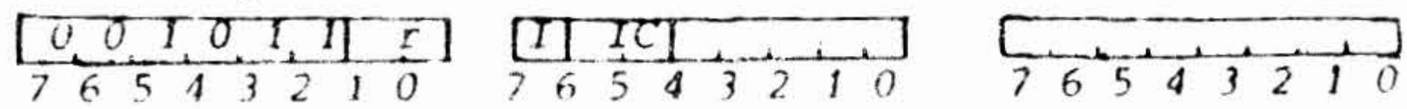
Cette instruction de triple longueur effectue un OU exclusif entre le contenu du registre R et la case mémoire pointée par l'adresse effective .
Le résultat de l'opération est rangé dans le registre R ; l'indirection ou l'indexation ou les deux peuvent être spécifiées .
Dans le cas où l'indexation est spécifiée, le résultat est rangé implicitement dans le registre zéro .
Table de vérité :

Bit (0-7)	Bit (0-7)	Résultat
0	0	0
0	1	1
1	1	0
1	0	1

REGISTRES DU PROCESSEUR AFFECTES CC

	CC1	CC0
Registre R		
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



COMPARAISONS ARITHMETIQUES

ET LOGIQUES

L'INSTRUCTION EST : CØM

ELLE PERMET DE COMPARER LA VALEÚR CONTENUE DANS UN REGISTRE AVEC UNE AUTRE VALEUR

RX : VALEUR

LE CODE CONDITION REND COMPTE DU RESULTAT ET IL EST POSSIBLE EN UTILISANT LE BIT COM DE SPECIFIER SI

- LA COMPARAISON EST LOGIQUE (entre valeurs absolues)

- LA COMPARAISON EST ARITHMETIQUE (entre valeurs relatives : nombres signe en complément à 2)

COMPARAISON AU REGISTRE ZERO (COMPARE TO REGISTER ZERO)

MNEMONIQUE : COMZ R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

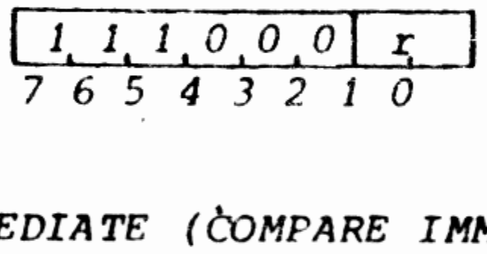
Cette instruction de simple longueur permet de comparer de manière logique ou arithmétique les contenus des registres R et 0 .

Si COM = 1 Le mode de comparaison est logique
Si COM = 0 Le mode de comparaison est arithmétique .
Les deux registres ne sont pas modifiés ; seul le code condition CC est positionné .

REGISTRES DU PROCESSEUR AFFECTES CC

			CC1	CC0
Registre 0	>	Registre R	0	1
"	=	"	0	0
"	<	"	1	0

CODE BINAIRE DE L'INSTRUCTION



COMPARAISON IMMEDIATE (COMPARE IMMEDIATE)

MNEMONIQUE : COMI,R V
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

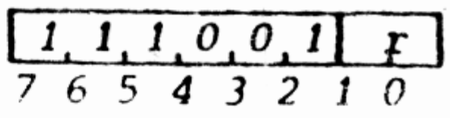
Cette instruction de double longueur permet de comparer logiquement ou arithmétiquemen
suivant la valeur du bit CØM les contenus du registre R et de la valeur immédiate V .

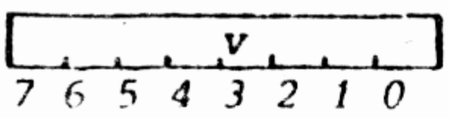
Si COM = 1 Le mode de comparaison est logique
Si COM = 0 Le mode de comparaison est arithmétique
Il n'y a pas de modification du registre ; seul le contenu du CC est modifié .

REGISTRES DU PROCESSEUR AFFECTES CC

			CC1	CC0
Registre R	>	(V)	0	1
"	=	(V)	0	0
"	<	(V)	1	0

CODE BINAIRE DE L'INSTRUCTION





COMPARAISON RELATIVE (COMPARE RELATIVE)

MNEMONIQUE : COMR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

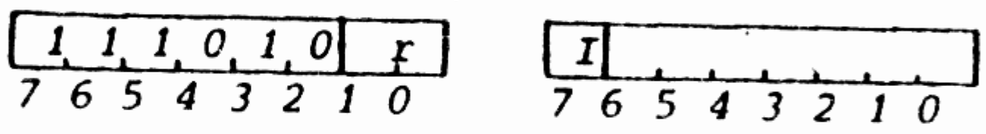
Cette instruction de double longueur permet de comparer logiquement ou arithmétiquement suivant la valeur du bit COM les contenus du registre R et de la case mémoire pointée par l'adresse effective .

Si COM = 1 Le mode de comparaison est logique
Si COM = 0 Le mode de comparaison est arithmétique
Il n'y a pas de modification du registre et de la mémoire ; seul le contenu du CC est modifié .

REGISTRES DU PROCESSEUR AFFECTES CC

	CC1	CC0
Registre R > (mémoire)	0	1
" = (mémoire)	0	0
" < (mémoire)	1	0

CODE BINAIRE DE L'INSTRUCTION



COMPARAISON ABSOLUE (COMPARE ABSOLUTE)

MNEMONIQUE : COMA,R (*) a (,X)
NOMBRE DE CYCLES : 4
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

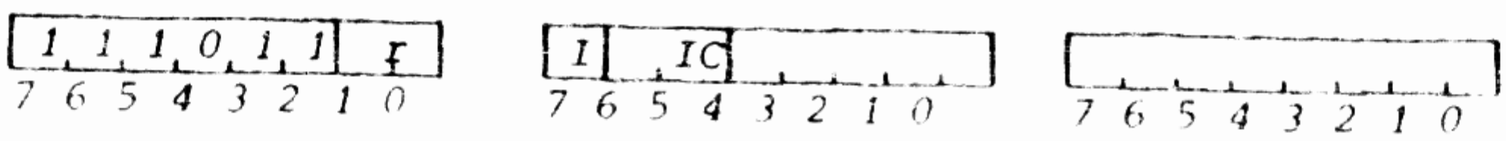
Cette instruction de triple longueur permet de comparer logiquement ou arithmétiquement suivant la valeur du bit COM les contenus du registre R et de la case mémoire pointée par la valeur a .

Si COM = 1 Le mode de comparaison est logique
Si COM = 0 Le mode de comparaison est arithmétique .
L'adressage peut de plus être indirect et/ou indexé,s'il y a indexation.c'est le conten du registre 0 qui est comparé à la mémoire .
Il n'y a pas de modification ni du registre ni de la mémoire seul le contenu du CC est modifié .

REGISTRES DU PROCESSEUR AFFECTES CC

	CC1	CC0
Registre R > (mémoire)	0	1
" = (mémoire)	0	0
" < (mémoire)	1	0

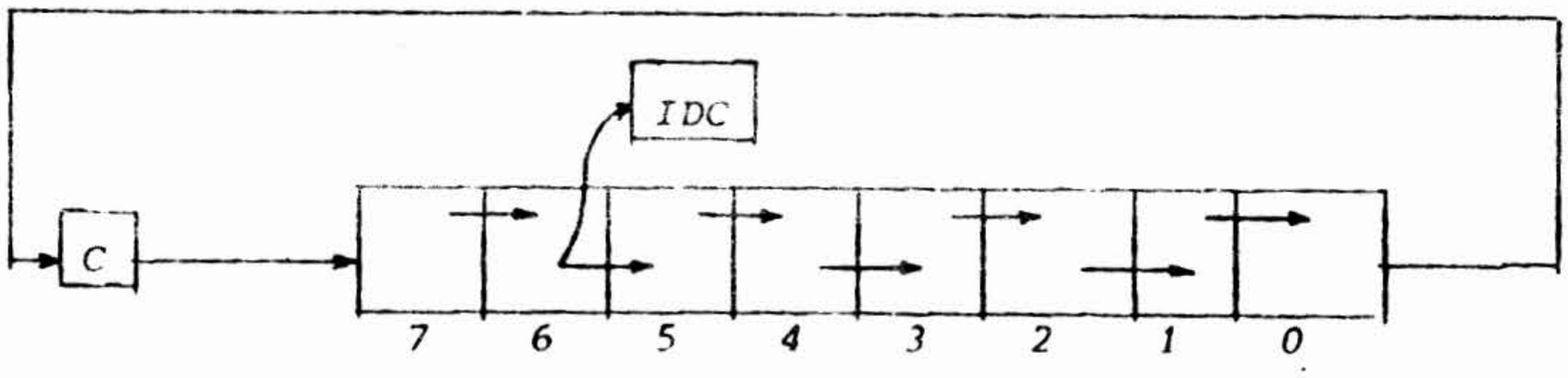
CODE BINAIRE DE L'INSTRUCTION



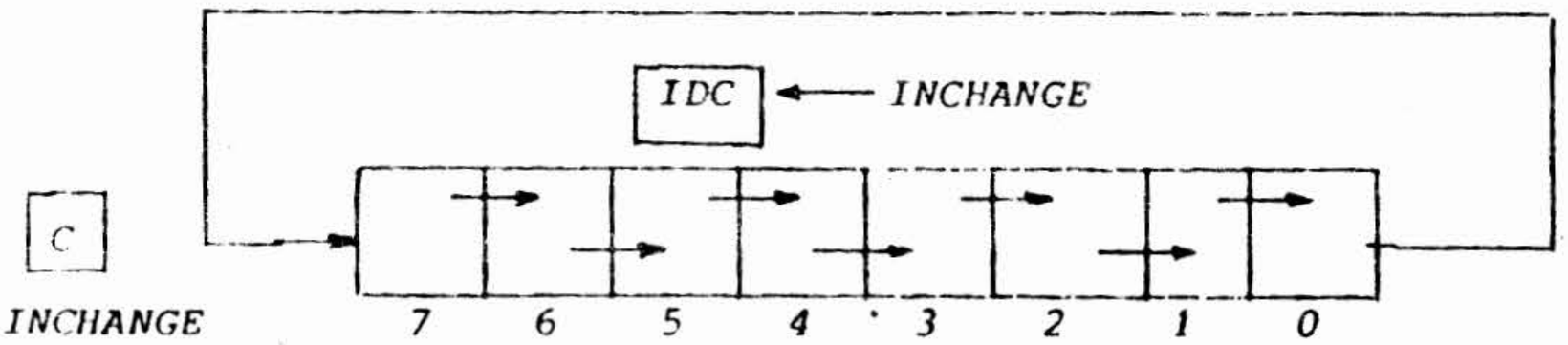
INSTRUCTIONS DE ROTATIONS

LES ROTATIONS SE FONT SUR LE REGISTRE SPECIFIE VERS LA DROITE OU VERS LA GAUCHE .

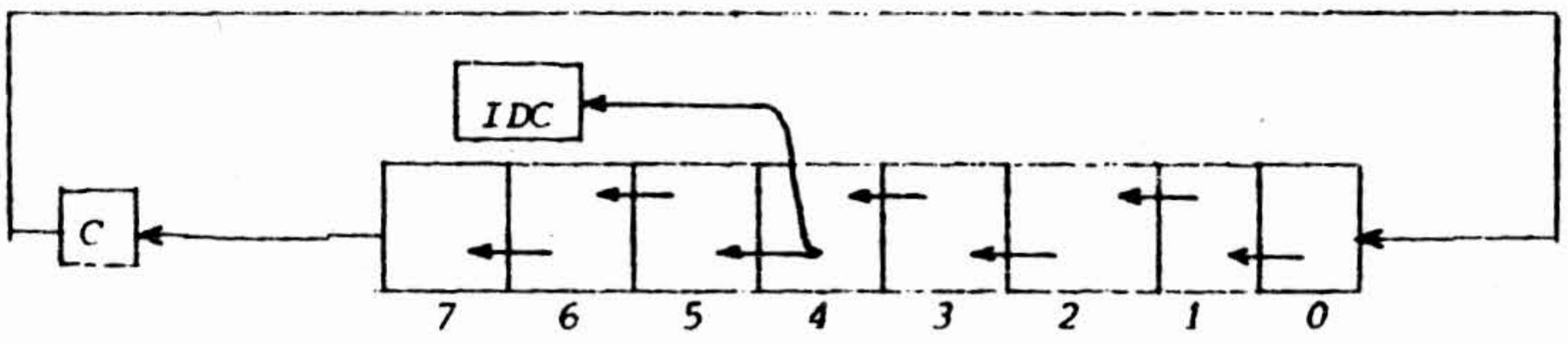
LA ROTATION S'EFFECTUE AVEC OU SANS LES CARRY (C, IDC) SUIVANT LA VALEUR DU BIT WC .



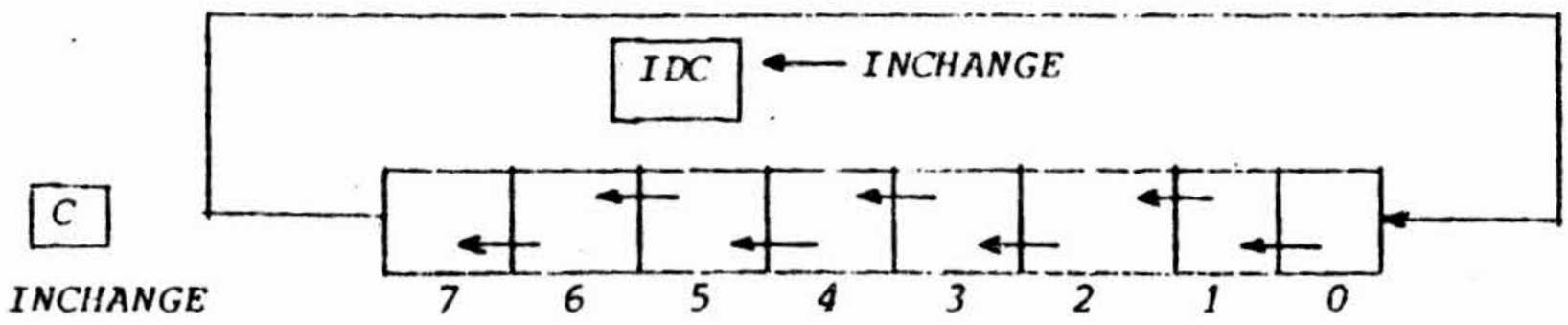
WC = 1 : ROTATION DROITE AVEC CARRY



WC = 0 : ROTATION DROITE SANS CARRY



WC = 1 : ROTATION GUACHE AVEC CARRY



WC = 0 : ROTATION GAUCHE SANS CARRY

ROTATION A GAUCHE (ROTATE REGISTER LEFT)

MNEMONIQUE : RRL,R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

Cette instruction de simple longueur permet d'effectuer une rotation du registre R de un bit vers la gauche, la rotation inclut ou non le carry suivant la valeur du bit WC du PSWL .

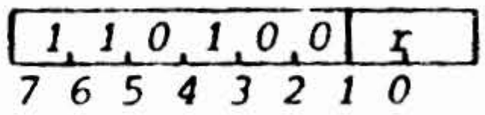
Si ce bit = 0 Bit 0 Bit 7
Si ce bit = 1 Bit 0 carry carry Bit 7

Attention : il peut y avoir modification du bit OVF .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



ROTATION A DROITE (ROTATE REGISTER RIGHT)

MNEMONIQUE : RRR,R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

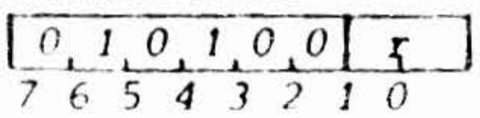
Cette instruction de simple longueur permet d'effectuer une rotation de 1 bit vers la droite sur le registre R . La rotation est effectuée suivant la valeur du bit WC du PSWL .

Si ce bit = 0 elle n'inclut pas le carry
Si ce bit = 1 elle inclut le carry
Attention : le bit OVF peut être positionné .

REGISTRES DU PROCESSEUR AFFECTES C, CC, IDC, OVF

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



MANIEMENT DU PSW

LES INSTRUCTIONS QUI PERMETTENT DE MANIPULER LE PSW SONT :

1 - CHARGEMENT DU PSW : LPS

R0 → PS | U / L

2 - STOCKAGE DU PSW : SPS

PS | U / L → R0

3 - MISE A ZERO SELECTIVE DE BITS : CPS

LES BITS SELECTIONNES PAR LE MASQUE SONT MIS A 0 .

4 - MISE A UN SELECTIVE DE BITS : PPS

LES BITS SELECTIONNES PAR LE MASQUE SONT MIS A 1 .

5 - TEST PAR MASQUE DU PSW : TPS

LES BITS A 1 DU MASQUE PERMETTENT DE SELECTIONNER LES BITS CORRESPONDANTS DU PSW .
LE CODE CONDITION REFLETE LE RESULTAT DU TEST .

REMARQUE : LE PSW EST ACCESSIBLE PAR MOITIE

PARTIE BASSE	PSL
PARTIE HAUTE	PSU

CHARGEMENT PSU (LOAD PROGRAM STATUS UPPER)

MNEMONIQUE : LPSU
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

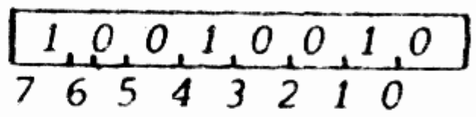
DESCRIPTION

Cette instruction de simple longueur permet de stocker dans la partie haute du PSW le contenu du registre 0 .
Les bits 3 et 4 du PSWU contiendront toujours 0 .

REGISTRES DU PROCESSEUR AFFECTES : F, II, SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



CHARGEMENT PSL (LOAD PROGRAM STATUS LOWER)

MNEMONIQUE : LPSL
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

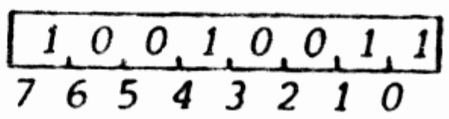
DESCRIPTION

Cette instruction de simple longueur permet de stocker dans la partie basse du PSW le contenu du registre 0 .

REGISTRES DU PROCESSEUR AFFECTES CC, IDC, RS, WC, OVF, CØM, C

CC Il prend les valeurs des bits 6 et 7 du registre 0

CODE BINAIRE DE L'INSTRUCTION



STOCKAGE PSU (STORE PROGRAM STATUS, UPPER)

MNEMONIQUE : SPSU
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

Cette instruction de simple longueur permet de stocker la partie haute du PSW dans le registre 0 .
Les bits 4 et 3 du PSW sont stockés comme zéros .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre 0	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION

0	0	0	1	0	0	1	0
7	6	5	4	3	2	1	0

STOCKAGE PSL (STORE PROGRAM STATUS, LOWER)

MNEMONIQUE : SPSL
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

Cette instruction de simple longueur permet de stocker la partie basse du PSW dans le registre 0 .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre 0	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION

0	0	0	1	0	0	1	1
7	6	5	4	3	2	1	0

MISE A UN SELECTIF PSU (PRESET PROGRAM STATUS UPPER SELECTIVE)

MNEMONIQUE : PPSU V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

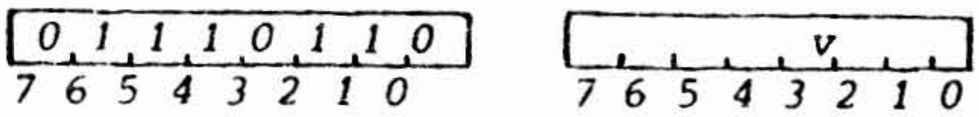
DESCRIPTION

Cette instruction de double longueur permet de positionner à 1 sélectivement suivant un masque V certains bits du PSU . En fait, cette instruction réalise un OU inclusif entre la valeur V et le registre PSU .
Les bits à un de V positionnent à un les bits correspondants du PSU .
Les bits à zéro de V laissent les bits correspondants du PSU inchangés .

REGISTRES DU PROCESSEUR AFFECTES F, II, SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



MISE A UN SELECTIF PSL (PRESET PROGRAM STATUS LOWER SELECTIVE)

MNEMONIQUE : PPSL V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

Cette instruction double longueur permet de positionner à 1 sélectivement suivant un masque V certains bits du PSL . En fait, cette instruction réalise un OU inclusif entre la valeur V et le registre PSL .
Les bits à un de V positionnent à un les bits correspondants du PSL .
Les bits à zéro de V laissent les bits correspondants du PSL inchangés .

REGISTRES DU PROCESSEUR AFFECTES CC, IDL, RS, WC, OVF, COM, C

CC Il peut être touché suivant la valeur V .

CODE BINAIRE DE L'INSTRUCTION

MISE A ZERO SELECTIF PSU (CLEAR PROGRAM STATUS UPPER, SELECTIVE)

MNEMONIQUE : CPSU V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

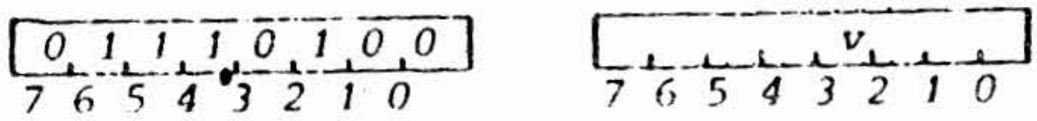
DESCRIPTION

Cette instruction de double longueur permet de positionner à 0 sélectivement suivant un masque V certains bits du PSU .
Les bits à un du masque V positionnent les bits correspondants du PSU à 0 .
Les autres bits ne sont pas modifiés .

REGISTRES DU PROCESSEUR AFFECTES F, II, SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



MISE A ZERO SELECTIF PSL (CLEAR PROGRAM STATUS LOWER, SELECTIVE)

MNEMONIQUE : CPSL V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

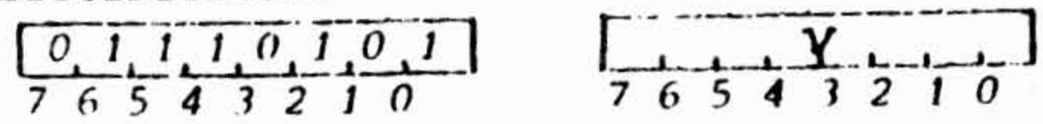
DESCRIPTION

Cette instruction de double longueur permet de positionner à 0 sélectivement suivant un masque V certains bits du PSL .
Les bits à 1 du masque V positionnent les bits correspondants du PSL à 0 .
Les autres bits ne sont pas modifiés .

REGISTRES DU PROCESSEUR AFFECTES CC, IDC, RS, WC, OVF, COM, C

CC Les bits du CC peuvent être modifiés .

CODE BINAIRE DE L'INSTRUCTION



TEST SELECTIF PSU (TEST PROGRAM STATUS UPPER, SELECTIVE)

MNEMONIQUE : TPSU V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

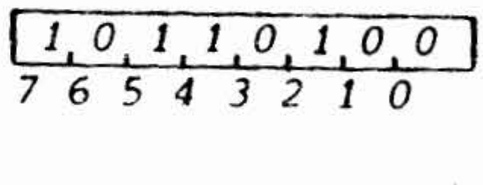
DESCRIPTION

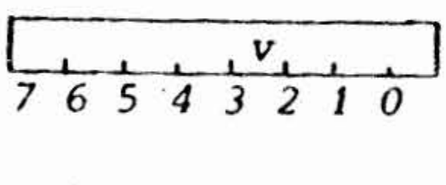
Cette instruction de double longueur teste certains bits de la partie haute du PSW . Le test est sélectif en ce sens que les bits à 1 de V pointent sur les bits correspondants du PSWU et suivant le résultat de l'opération le code condition est positionné .

REGISTRES DU PROCESSEUR AFFECTES CC

		CC1	CC0
CC	Tous les bits sélectionnés dans le PSU sont à 1 :	0	0
	Un des bits sélectionnés dans le PSU est à 0 :	1	0

CODE BINAIRE DE L'INSTRUCTION





TEST SELECTIF PSL (TEST PROGRAM STATUS LOWER, SELECTIVE)

MNEMONIQUE : TPSL V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

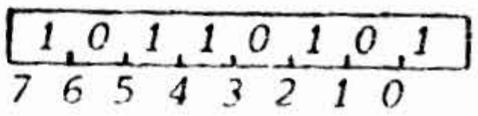
DESCRIPTION

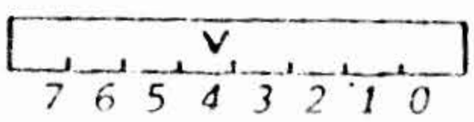
Cette instruction de double longueur teste certains bits de la partie basse du PSW . Le test est sélectif en ce sens que les bits à 1 de V pointent sur les bits correspondants du PSWL et suivant le résultat de l'opération le code condition est positionné .

REGISTRES DU PROCESSEUR AFFECTES CC

		CC1	CC0
CC	Tous les bits sélectionnés dans le PSL sont à 1 :	0	0
	Un des bits sélectionnés dans le PSL est à 0 :	1	0

CODE BINAIRE DE L'INSTRUCTION





LES DIVERS TYPES DE BRANCHEMENTS

SYNTAXE :

ETIQUETTE	CODE OPERATION	VALEUR REGISTRE	ADRESSE

LES BRANCHEMENTS PERMETTENT D'ACCEDER

SOIT A UNE ADRESSE RELATIVE (DEPLACEMENT -64 à + 63)

SOIT A UNE ADRESSE ABSOLUE (SUR TOUT L'ESPACE MEMOIRE)

SOIT A UN DEPLACEMENT PAR RAPPORT A L'ADRESSE '0000' HEXA

LES BRANCHEMENTS S'EFFECTUENT DE 7 MANIERES

SOIT SUR UNE CONDITION LOGIQUE FAUSSE

BCFA (absolu)
BCFR (relatif)

SOIT SUR UNE CONDITION LOGIQUE VRAIE

BCTA (absolu)
BCTR (relatif)

BRANCHEMENT AVEC INCREMENT DE REGISTRE

BIRR
BIRA

BRANCHEMENT AVEC DECREMENT DE REGISTRE

BDRR
BDRA

BRANCHEMENT SUR REGISTRE NON NUL

BRNA
BRNR

BRANCHEMENT INDEXE ABSOLU

BRANCHEMENT RELATIF A LA PAGE 0

BRANCHEMENT SUR LE CONTENU D'UN REGISTRE

LE BRANCHEMENT S'EFFECTUE DANS CES CAS SI

1 - BRANCHEMENT SI LE REGISTRE SPECIFIE EST NON NUL

```
BRNA,RX     ADR
BRNR,RX     ADR
```

ALLER A ADR SI RX ≠ 0
SINON EN SEQUENCE

2 - BRANCHEMENT SI LE REGISTRE SPECIFIE EST NON NUL APRES AVOIR ETE INCREMENTE

```
BIRA,RX     ADR
BIRR,RX     ADR
```

RX ⟵ (RX) + 1
ALLER A ADR SI RX ≠ 0
SINON EN SEQUENCE

3 - BRANCHEMENT SI LE REGISTRE SPECIFIE EST NON NUL APRES AVOIR ETE DECREMENTE

```
BDRA,RX     ADR
BDRR,RX     ADR
```

RX ⟵ (RX) - 1
ALLER A ADR SI RX ≠ 0
SINON EN SEQUENCE

BRANCHEMENT SUR CODE CONDITION

1 - BRANCHEMENT SI LA VALEUR INDIQUEE EST CELLE DU CODE CONDITION

BCTA, VAL	ADR
BCTR, VAL	ADR

SE BRANCHER A ADR SI CC = VAL
SINON EN SEQUENCE

2 - BRANCHEMENT SÎ LA VALEUR INDIQUEE EST DIFFERENTE DE CELLE DU CODE CONDITION

BCFA, VAL	ADR
BCFR, VAL	ADR

SE BRANCHER A ADR SI CC ≠ VAL
SINON EN SEQUENCE

REMARQUE :

CERTAINS BRANCHEMENTS INCONDITIONNELS SE FONT EN CODANT VAL = 3

BRANCHEMENT ZERO RELATIF (ZERO BRANCH RELATIVE)

MNEMONIQUE : ZBRR (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

Cette instruction de double longueur permet d'effectuer un branchement relatif par rapport à la page zéro adresse zéro .
La valeur d'adresse que nous spécifions est considérée comme un déplacement entre - 64 et + 63 octets modulo 8192 par rapport à page 0 adresse 0 .
Le processeur remet à 0 les bits de page (Adresse bus 13 et 14)
Il est possible de spécifier un adressage indirect .

NE PAS UTILISER CETTE INSTRUCTION POUR LE HOBBY COMPUTER

REGISTRES DU PROCESSEUR AFFECTES : aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION

BRANCHEMENT SUR CONDITION VRAIE, RELATIF (BRANCH ON CONDITION TRUE, RELATIVE)

MNEMONIQUE : BCTR,V (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

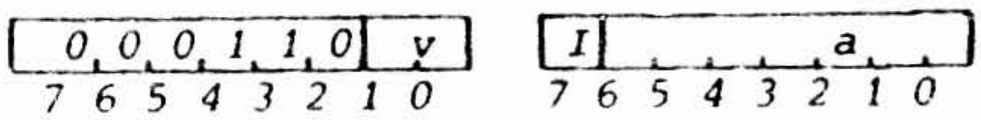
DESCRIPTION

Cette instruction de double longueur permet d'effectuer un branchement relatif si le contenu du code condition CC égale la valeur immédiate V .
L'adressage indirect est possible .
Si V = 3 **branchement inconditionnel .**

REGISTRES DU PROCESSEUR AFFECTES aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT SUR CONDITION FAUSSE, RELATIF (BRANCH ON CONDITION FALSE, RELATIVE)

MNEMONIQUE : BCFR,V (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

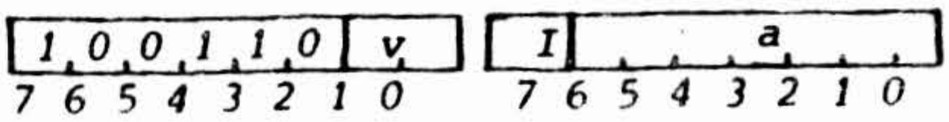
DESCRIPTION

Cette instruction de double longueur permet d'effectuer un branchement relatif si la valeur du code condition CC n'est pas égale à la valeur immédiate V .
L'adressage indirect est possible .
V ne peut valoir 3, car dans ce cas le code opération est celui de l'instruction ZBRR .

REGISTRES DU PROCESSEUR AFFECTES aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT SUR CONDITION VRAIE, ABSOLU (BRANCH ON CONDITION TRUE, ABSOLUTE)

MNEMONIQUE : BCTA,V (*) a
NOMBRE DE CYCLES. : 3
TYPE D'ADRESSAGE : Absolu

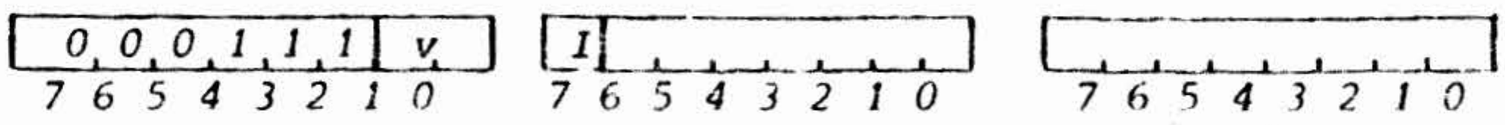
DESCRIPTION

Cette instruction de triple longueur permet d'effectuer un branchement absolu si la valeur du code condition CC égale la valeur immédiate V .
L'adressage indirect est possible .
Si V = 3 branchement inconditionnel .

REGISTRES DU PROCESSEUR AFFECTES : aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT SUR CONDITION FAUSSE, ABSOLU (BRANCH ON CONDITION FALSE, ABSOLUTE)

MNEMONIQUE : BCFA,V (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Absolu

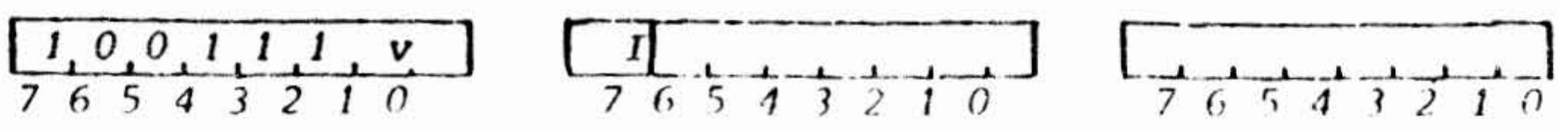
DESCRIPTION

Cette instruction de triple longueur permet un branchement absolu si le code condition CC diffère de la valeur immédiate V .
L'adressage indirect est possible .
La valeur V ne doit pas valoir 3 sinon le code opération correspond à celui de BXA .

REGISTRES DU PROCESSEUR AFFECTES aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT AVEC REGISTRE INCREMENTE, RELATIF (BRANCH ON INCREMENTING REGISTER, RELATIV

MNEMONIQUE : BIRR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

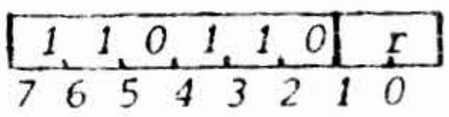
Cette instruction de double longueur permet d'effectuer un branchement relatif si le contenu du registre R préalablement incrémenté est différent de 0 .

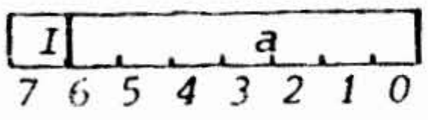
L'adressage indirect est possible .

REGISTRES DU PROCESSEUR AFFECTES aucun

CC non affecté

CODE BINAIRE DE L'INSTRUCTION





BRANCHEMENT AVEC REGISTRE DECREMENTE, RELATIF (BRANCH ON DECREMENTING REGISTER, RELATI

MNEMONIQUE : BDRR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

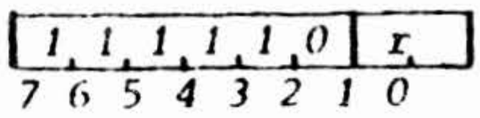
Cette instruction de double longueur permet d'effectuer un branchement relatif si le contenu du registre R est différent de 0, celui-ci ayant été préalablement décrémenté

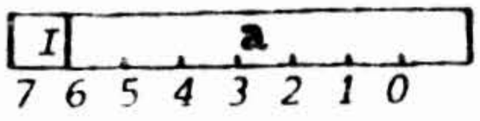
L'adressage indirect est possible .

REGISTRES DU PROCESSEUR AFFECTES Aucun

CC Non affecté .

CODE BINAIRE DE L'INSTRUCTION





BRANCHEMENT AVEC REGISTRE INCREMENTE, ABSOLU (BRANCH ON INCREMENTING REGISTER,ABSOLUTE)

MNEMONIQUE : BIRA,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

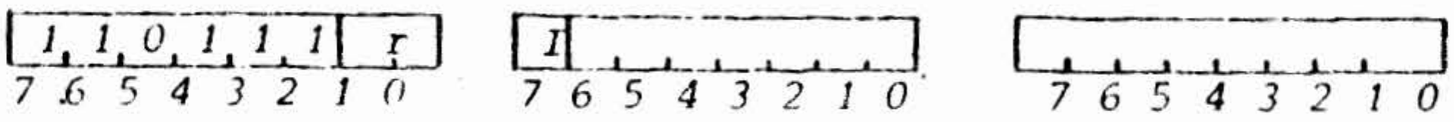
Cette instruction de triple longueur permet d'effectuer un branchement absolu si le contenu du registre R au préalable incrémenté est différent de 0

L'adressage indirect est possible .

REGISTRES DU PROCESSEUR AFFECTES Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT AVEC REGISTRE DECREMENTE, ABSOLU (BRANCH ON DECREMENTING REGISTER, ABSOLUTE)

MNEMONIQUE : BDRA,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Absolu .

DESCRIPTION

Cette instruction en triple longueur permet d'effectuer un branchement absolu si le contenu du registre R est non nul après que celui-ci ait été décrémenté .

L'adressage indirect est possible .

REGISTRES DU PROCESSEUR AFFECTES Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION

BRANCHEMENT AVEC REGISTRE NON NUL, RELATIF (BRANCH ON REGISTER NON-ZERO,RELATIVE)

MNEMONIQUE : BRNR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

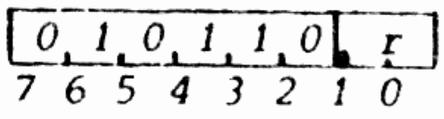
DESCRIPTION

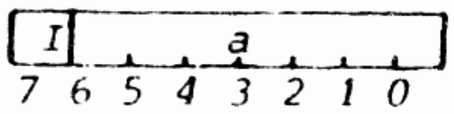
Cette instruction de double longueur permet de se débrancher si le registre R est non nul .
L'adressage indirect est possible .

REGISTRES DU PROCESSEUR AFFECTES Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION





BRANCHEMENT AVEC REGISTRE NON NUL, ABSOLU (BRANCH ON REGISTER NON-ZERO, ABSOLUTE)

MNEMONIQUE : BRNA,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Absolu

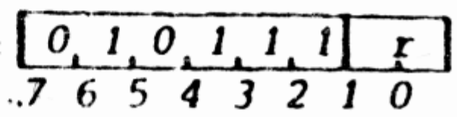
DESCRIPTION

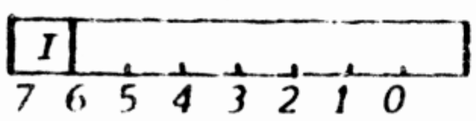
Cette instruction de triple longueur effectue un branchement à l'adresse effective si le contenu du registre R est différent de zéro .
Si le registre est égal à zéro l'exécution du programme continue en séquence .
L'adressage indirect est possible .

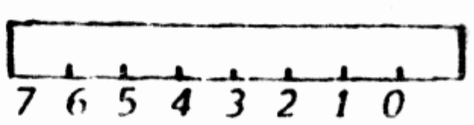
REGISTRES DU PROCESSEUR AFFECTES Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION







LE BRANCHEMENT INDEXE ABSOLU

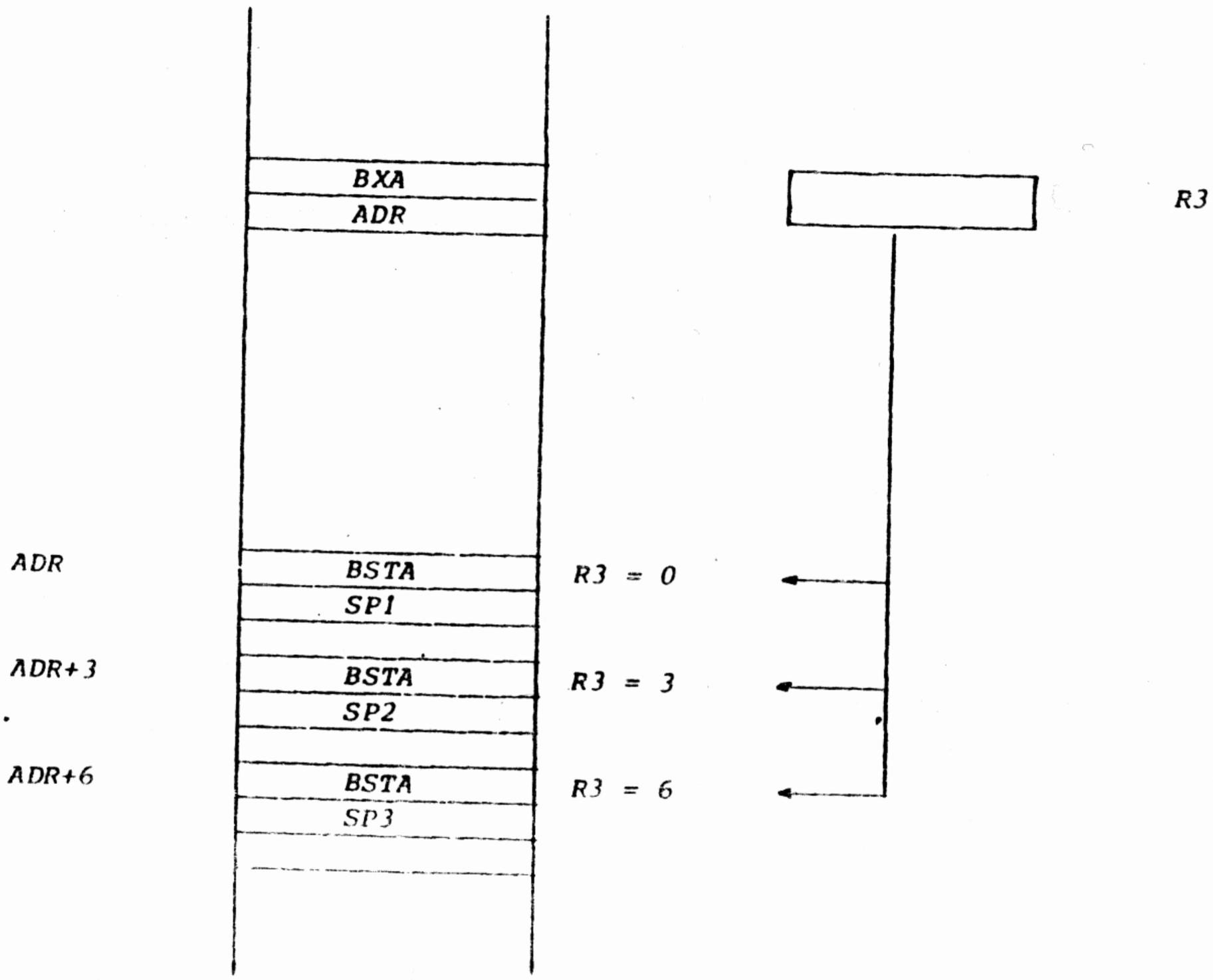
C'EST UNE AUTRE MANIERE D'INSTITUER UN VECTEUR DE TRANSFERT .
L'INSTRUCTION CONSIDEREE :

BXA ADR,R3

PERMET DE SE BRANCHER EN CALCULANT L'ADRESSE EFFECTIVE COMME SOMME DE ADR ET DU CONTENU D'UN INDEX QUI DOIT ETRE R3

ADRE = ADR + (R3)

ON PEUT DONC L'UTILISER AINSI



BRANCHEMENT INDEXE ABSOLU (BRANCH INDEXED ABSOLUTE)

MNEMONIQUE : BXA (*) a,X
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

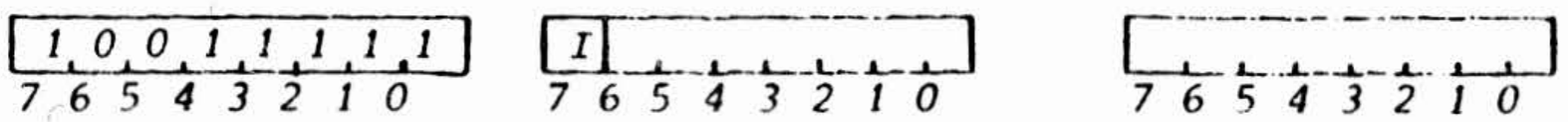
Cette instruction triple longueur permet de réaliser un branchement inconditionnel . L'indexage est obligatoire, et le registre 3 doit alors contenir l'index .

Si l'adressage indirect est spécifié, la valeur de l'index est ajoutée à la valeur indirecte pour calculer l'adresse effective de branchement ..

REGISTRES DU PROCESSEUR AFFECTES : Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT VERS LES SOUS-PROGRAMMES

LES INSTRUCTIONS DE BRANCHEMENT PERMETTENT D'ATTEINDRE UN SOUS-PROGRAMME PAR

- UNE ADRESSE RELATIVE
- UNE ADRESSE ABSOLUE

LES BRANCHEMENTS VERS LES SOUS-PROGRAMMES S'EFFECTUENT DE LA MANIERE SUIVANTE :

CODE CONDITION = VALEUR
CODE CONDITION ≠ VALEUR
INCONDITIONNELS
REGISTRE NON NUL .

DEUX BRANCHEMENTS PARTICULIERS SONT EGALEMENT DISPONIBLES

BRANCHEMENT RELATIF PAR RAPPORT A LA PAGE ZERO
BRANCHEMENT ABSOLU INDEXE

LE BRANCHEMENT RELATIF PAR RAPPORT A L'ADRESSE ZERO PAGE ZERO PERMET EN SPECIFIANT UNE INDIRECTION D'EFFECTUER UN VECTEUR DE TRANSFERT VERS DES SOUS-PROGRAMMES .

BRANCHEMENT A SOUS-PROGRAMME RELATIF A PAGE 0 (ZERO BRANCH TO SUBROUTINE RELATIVE)

MNEMONIQUE : ZBSR (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

Cette instruction double longueur permet de transférer inconditionnellement le contrôle à un sous-programme .

L'adressage est relatif à l'adresse 0 de la page 0 et (a) spécifie un déplacement entre - 64 et + 63 modulo 8192_{10} .

- Si l'on code ZBSR - 10 on se branche en 8182_{10}

Le processeur remettra à 0 les lignes 13 et 14 du bus adresse .

Cette instruction peut être exécutée n'importe où .

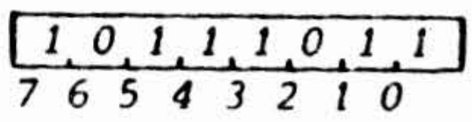
- Il est possible de spécifier un adressage indirect .

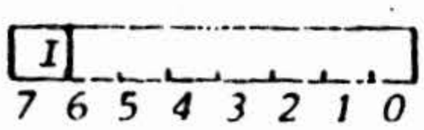
NE PAS UTILISER CETTE INSTRUCTION POUR LE HOBBY COMPUTER

REGISTRES DU PROCESSEUR AFFECTES SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION .





BRANCHEMENT A SOUS-PROGRAMME SUR CONDITION VRAIE, RELATIF (BRANCH TO SUBROUTINE ON CONDITION TRUE, RELATIVE)

MNEMONIQUE : BSTR,V (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

Cette instruction double longueur permet de transférer le contrôle à un sous-programme si la valeur du code condition CC égale V .

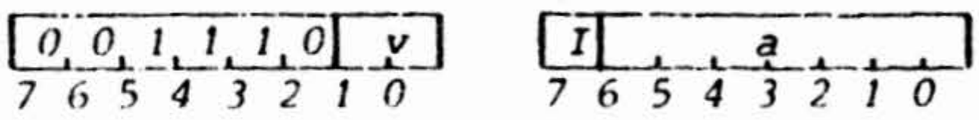
L'adressage est relatif et l'on peut spécifier une indirection .

Si V = 3 le branchement est inconditionnel .

REGISTRES DU PROCESSEUR AFFECTES SP

SP Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT A SOUS-PROGRAMME SUR CONDITION FAUSSE, RELATIF (BRANCH TO SUBROUTINE ON CONDITION FALSE, RELATIVE)

MNEMONIQUE : BSFR,V (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

Cette instruction en double longueur permet de transférer le contrôle à un sous-programme si seulement le code condition CC n'est pas égal à la valeur V .

L'adressage est relatif et l'on peut spécifier une indirection .

V ne peut valoir 3 car dans ce cas le processeur reconnaitrait l'instruction ZBSR .

REGISTRES DU PROCESSEUR AFFECTES SP

SP Non affecté

CODE BINAIRE DE L'INSTRUCTION

BRANCHEMENT A SOUS-PROGRAMME SUR CONDITION VRAIE,ABSOLU (BRANCH TO SUBROUTINE ON CONDITION TRUE, ABSOLUTE)

MNEMONIQUE	:	BSTA,V (*) a
NOMBRE DE CYCLES	:	3
TYPE D'ADRESSAGE	:	Absolu

DESCRIPTION

Cette instruction en triple longueur permet de transférer le contrôle à un sous-prog si seulement la valeur du code condition CC égale V .

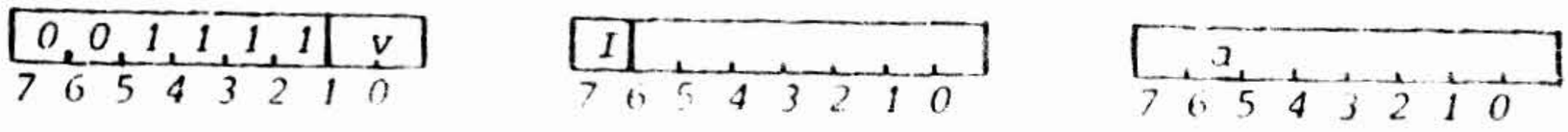
L'adressage est absolu si l'on peut spécifier une indirection .

Si V = 3 le branchement est inconditionnel .

REGISTRES DU PROCESSEUR AFFECTES SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT A SOUS-PROGRAMME SUR CONDITION FAUSSE, ABSOLU (BRANCH TO SUBROUTINE ON CONDITION FALSE, ABSOLUTE)

MNEMONIQUE	:	BSFA, V (*) a
NOMBRE DE CYCLES	:	3
TYPE D'ADRESSAGE	:	Absolu

DESCRIPTION

Cette instruction de triple longueur permet de transférer le contrôle à un sous-progra seulement si le code condition CC est différent de la valeur V .

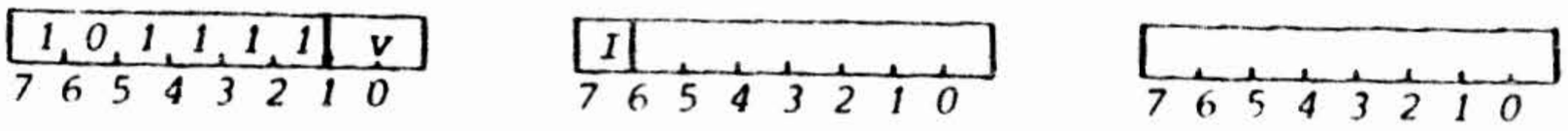
L'adressage est absolu et peut être fait avec une indirection .

V ne peut valoir 3 car dans ce cas le processeur reconnaitra l'instruction BSXA .

REGISTRES DU PROCESSEUR AFFECTES SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



BRANCHEMENT A SOUS-PROGRAMME SUR REGISTRE NON-NUL, RELATIF (BRANCH TO SUBROUTINE ON NON-ZERO REGISTER, RELATIVE)

MNEMONIQUE : BSNR,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

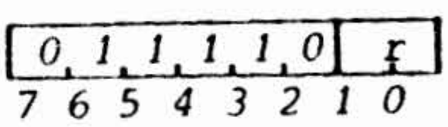
Cette instruction de double longueur permet de se débrancher à un sous-programme si le contenu du registre R est différent de zéro .

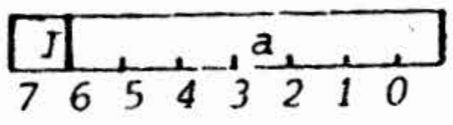
L'adressage est relatif et l'on peut de plus spécifier une indirection .

REGISTRES DU PROCESSEUR AFFECTES SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION





BRANCHEMENT A SOUS-PROGRAMME SUR REGISTRE NON-NUL, ABSOLU (BRANCH TO SUBROUTINE ON NON-ZERO REGISTER, ABSOLUTE)

MNEMONIQUE : BSNA,R (*) a
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Relatif

DESCRIPTION

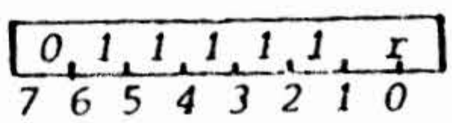
Cette instruction de triple longueur permet de se brancher à l'adresse "a" qui est l'adresse d'un sous-programme lorsque le contenu du registre 0 est différent de zéro .

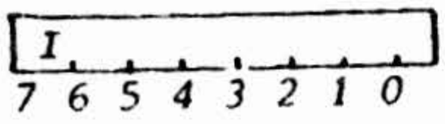
Il est possible d'utiliser un adressage indirect .

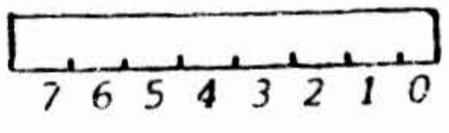
REGISTRES DU PROCESSEUR AFFECTES SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION







BRANCHEMENT A SOUS-PROGRAMME, INDEXE, ABSOLU, INCONDITIONNEL (BRANCH TO SUBROUTINE INDEXED, ABSOLUTE, UNCONDITIONAL)

MNEMONIQUE : BSXA (*) a,X
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Absolu

DESCRIPTION

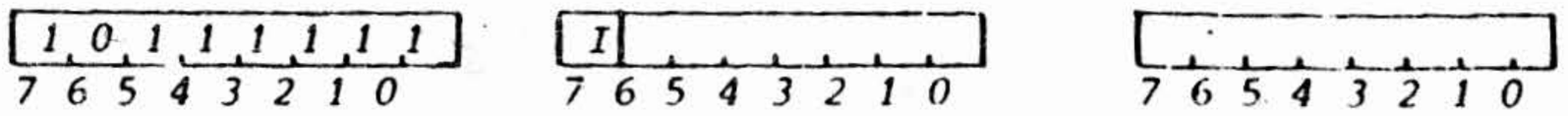
Cette instruction de triple longueur permet au processeur d'effectuer un branchement inconditionnel à un sous-programme . L'indexation est obligatoire et le registre 3 est obligatoirement celui qui contient l'index .

Si l'on spécifie un adressage indirect (*) la valeur de l'index est ajoutée à l'adresse indirecte pour calculer l'adresse effective du branchement .

REGISTRES DU PROCESSEUR AFFECTES SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



RETOUR DES SOUS-PROGRAMMES

2 TYPES DE RETOUR :

- LE RETOUR NORMAL CONDITIONNEL OU INCONDITIONNEL

- LE RETOUR AVEC AUTORISATION DES INTERRUPTIONS CONDITIONNEL OU INCONDITIONNEL

LES CONDITIONS DE RETOUR CONCERNENT LE CODE CONDITION, LE RETOUR INCONDITIONNEL EST OBTENU EN CODANT UNE VALEUR 3 POUR LE CODE CONDITION .

RETOUR DE SOUS-PROGRAMME, CONDITIONNEL (RETURN FROM SUBROUTINE, CONDITIONAL)

MNEMONIQUE : RETC,V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Registre

DESCRIPTION

Cette instruction en simple longueur est utilisée par un sous-programme pour effectuer un retour conditionnel .

Condition de retour :

Le retour s'effectue si la valeur V égale le code condition, sinon exécution de l'instruction en séquence .

Si V = 3 un retour inconditionnel s'effectue .

REGISTRES DU PROCESSEUR AFFECTES SP

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION

7	6	5	4	3	2	1	0
0	0	0	1	0	1	v	

RETOUR DE SOUS-PROGRAMME AVEC AUTORISATION D'INTERRUPTION, CONDITIONNEL (RETURN FROM SUBROUTINE AND ENABLE INTERRUPT CONDITIONAL

MNEMONIQUE : RETE,V
NOMBRE DE CYCLES : 3 .
TYPE D'ADRESSAGE : Registre

DESCRIPTION

... de simple longueur est utilisée pour effectuer un retour conditionnel depuis un sous-programme et si le retour se fait le bit d'interruption (II) est positionné à zéro autorisant de nouvelles interruptions .

On utilise surtout cette instruction comme retour de sous-programmes de traitement d'interruptions .

Conditions de retour

Le retour s'effectue si la valeur V égale le code condition .

Si V = 3 le retour est inconditionnel .

REGISTRES DU PROCESSEUR AFFECTES SP, II

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION

7	6	5	4	3	2	1	0
0	0	1	1	0	1	v	

INSTRUCTIONS

ENTREES/SORTIES

ELLES PRENNENT L'OCTET DANS UN REGISTRE POUR LE DEPOSER SUR LE BUS DONNEES

OU

ACQUIERENT UN OCTET DEPUIS LE BUS DONNEE DANS UN REGISTRE

LECTURE DONNEES (READ DATA)

MNEMONIQUE : REDD,R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

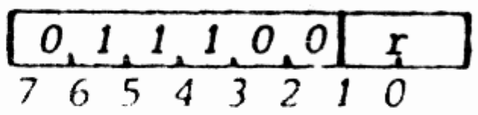
DESCRIPTION

Cette instruction de simple longueur permet d'acquérir dans le registre R l'octet de donnée disponible sur le bus de données . L'octet peut être considéré comme une donnée .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



LECTURE CONTROLE (READ CONTROL)

MNEMONIQUE : REDC,R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

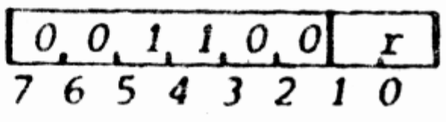
DESCRIPTION

Cette instruction de simple longueur permet d'acquérir dans le registre l'octet de donnée disponible sur le bus de données .
L'octet peut être considéré comme une commande .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION



LECTURE ETENDUE (READ EXTENDED)

MNEMONIQUE : REDE,R V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

Cette instruction de double longueur permet d'acquérir dans le registre R l'octet circulant sur le bus données . Durant l'exécution de cette instruction l'octet immédiat "V" est disponible sur le bus adresses .

REGISTRES DU PROCESSEUR AFFECTES CC

Registre R	CC1	CC0
Positif	0	1
Nul	0	0
Négatif	1	0

CODE BINAIRE DE L'INSTRUCTION

ECRITURE DONNEES (WRITE DATA)

MNEMONIQUE : WRTD,R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

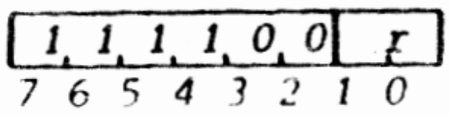
Cette instruction de simple longueur permet d'envoyer un octet de donnée vers un périphérique . L'octet à écrire est contenu dans le registre R .

REGISTRES DU PROCESSEUR AFFECTES

Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



ECRITURE CONTROLE (WRITE CONTROL)

MNEMONIQUE : WRTC,R
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE : Registre

DESCRIPTION

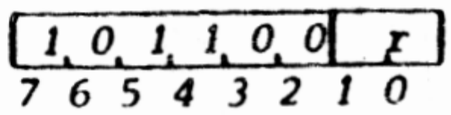
Cette instruction de simple longueur permet d'envoyer un octet considéré comme une commande vers un périphérique .
considéré se trouve dans le registre R et est envoyé sur le bus de donnée .

REGISTRES DU PROCESSEUR AFFECTES

Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



ECRITURE ETENDUE (WRITE EXTENDED)

MNEMONIQUE : WRTE,R V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

Cette instruction de double longueur permet de donner un octet de donnée à un organe externe .
L'octet à sortir se trouve dans le registre R et est disponible sur le bus de données; simultanément la donnée immédiate V est disponible sur le bus adresse ; elle peut être considérée comme une adresse de périphérique .

REGISTRES DU PROCESSEUR AFFECTES aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION

INSTRUCTIONS DIVERSES

NON OPERATION (NO OPERATION)

MNEMONIQUE : NOP
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE :

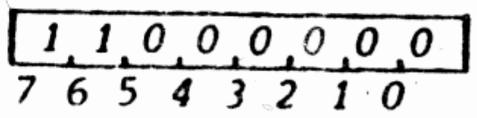
DESCRIPTION

Cette instruction de longueur simple permet au processeur de consommer 2 cycles en ne faisant rien .

REGISTRES DU PROCESSEUR AFFECTES Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



TEST AVEC MASQUE, IMMEDIAT (TEST UNDER MASK IMMEDIATE)

MNEMONIQUE : TMI,R V
NOMBRE DE CYCLES : 3
TYPE D'ADRESSAGE : Immédiat

DESCRIPTION

Cette instruction de double longueur permet de sélectionner par le masque certains bits du registre R et de regarder s'ils sont ou non tous à 1 .

Les bits à 1 du masque servent à désigner les bits à considérer dans R .

Si tous les bits sélectionnés sont à 1 le code condition vaut 0
Si l'un des bits sélectionnés vaut 0 le code condition vaut 2

REGISTRES DU PROCESSEUR AFFECTES CC

	CC1	CC0
Tous les bits à 1	0	0
Un des bits à zéro	1	0

CODE BINAIRE DE L'INSTRUCTION

HALTE (HALT)

MNEMONIQUE : HALT
NOMBRE DE CYCLES : 2
TYPE D'ADRESSAGE :

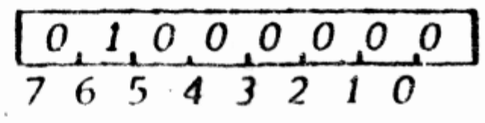
DESCRIPTION

- Cette instruction de longueur simple force le processeur à arrêter son exécution et à entrer dans le mode d'attente
- La seule manière de sortir de cet état pour entrer en mode normal est de faire une RAZ ou d'interrompre le processeur .

REGISTRES DU PROCESSEUR AFFECTES Aucun

CC Non affecté

CODE BINAIRE DE L'INSTRUCTION



IV - DESCRIPTION DES MEMOIRES PROGRAMMABLES ET DE L'ESPACE MEMOIRE DISPONIBLE

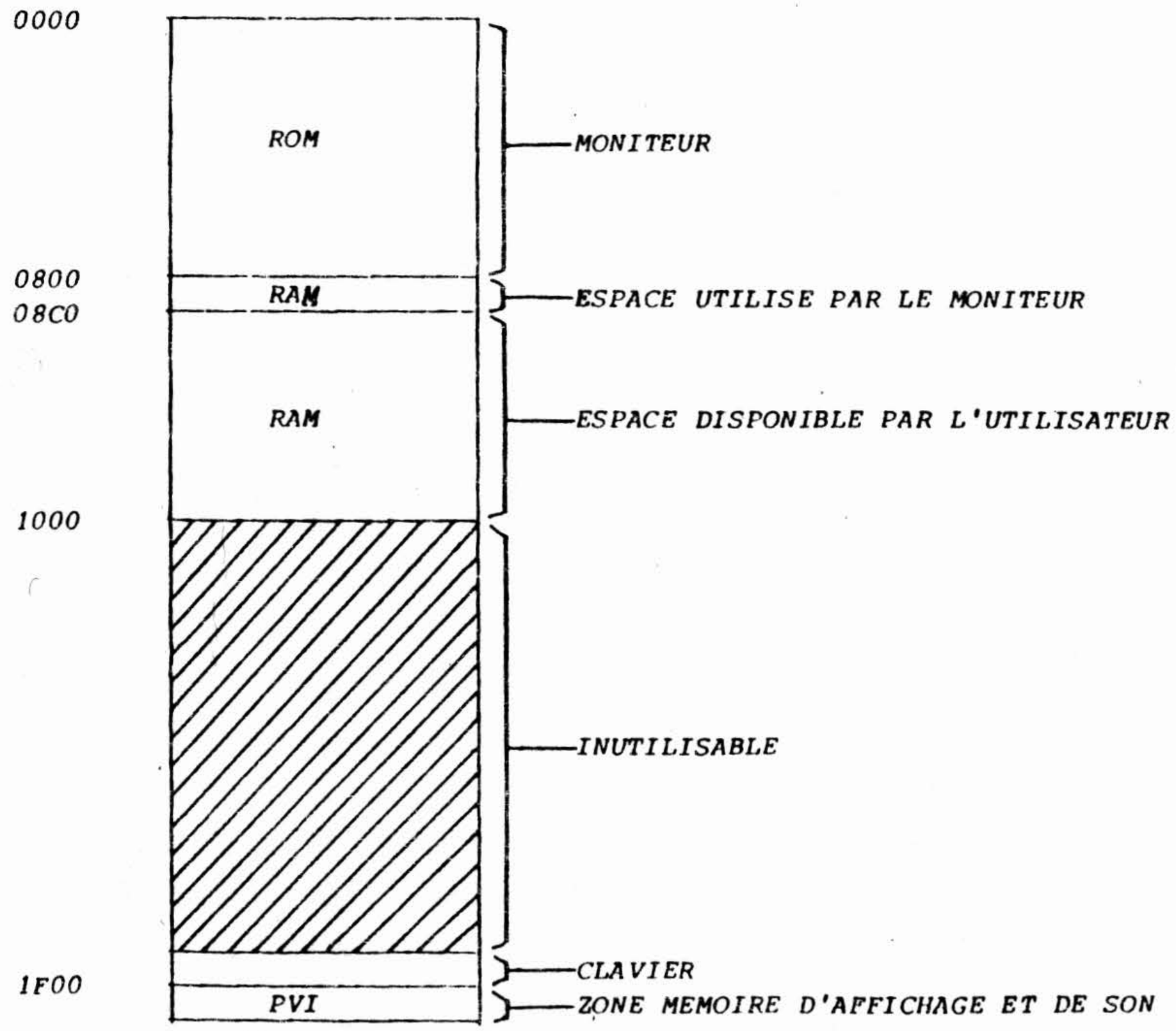
I - Configuration du HOBBY COMPUTER

1) Introduction

Le HOBBY COMPUTER possède 2K (2048) octets de mémoires vives (RAM) dans lesquelles on peut écrire et lire les informations en code machine . Les RAM (Random Access Memory) sont des mémoires qui ne peuvent garder leurs informations lorsqu'on coupe le courant, au contraire des mémoires mortes (ROM = Read Only Memory) qui ne permettent que la lecture . C'est pourquoi on doit utiliser des cassettes à bande magnétique pour sauvegarder le contenu des RAM .

2) Espace Mémoire Adressable

La configuration du système est donnée par le code mémoire ci-dessous :



Détaillons cette organisation mémoire :

- de l'adresse 0000 à l'adresse 0800, le programme moniteur occupe 2K octets d'espace mémoire . (En effet $0800_{16} = 0400_{16} + 0400_{16}$ et $0400_{16} = 4 \times 256_{10} = 1024_{10} = 1K$) . Le programme moniteur est un programme de base permettant d'utiliser le HOBBY COMPUTER .

- adresse 0800 à adresse 08C0 (non comprise) : zone occupant 12 X 16 = 192 octets de mémoire utilisée par le moniteur . L'utilisateur n'y a pas accès, cette zone est protégée .

- adresse 08C0 à adresse 1000 (non comprise) : zone occupant 2048 - 192 = 1856 octets de mémoire disponible par l'utilisateur pour y écrire ses programmes

- adresse 1E88 à adresse 1E8E : 7 octets réservés à l'identification des claviers .

- adresse 1F00 à adresse 2000 (non comprise) : zone occupée par le PVI (Programmable Video Interface) . le paragraphe suivant nous explicite le PVI

II - Le PVI

Le PVI est une RAM spécialisée permettant de programmer et d'afficher sur le téléviseur 4 objets, leurs positions, leurs tailles, leurs couleurs, le décor, le son et le score .
Analysons toutes ses possibilités . Voir Figure 1 .

1) Video-génération des objets

Comme le montre la figure 1, chacun des 4 objets est décrit par cinq éléments de données :

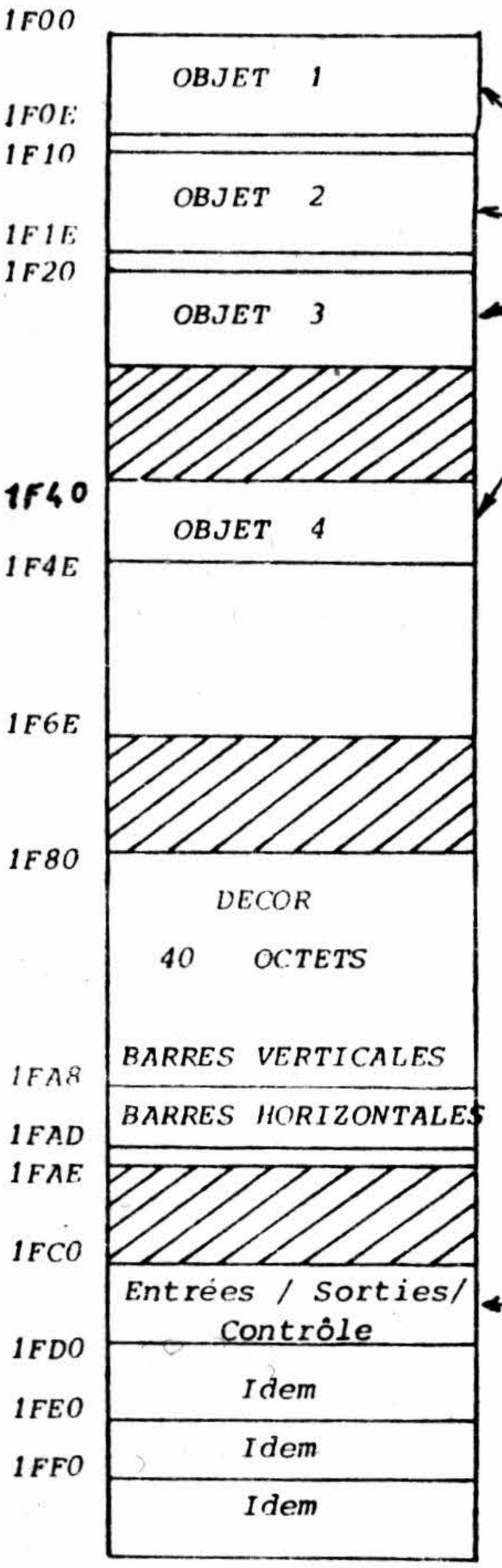
. Dimension de l'objet - 2 bits
. Forme de l'objet - 80 bits
. Position de l'objet - 16 bits
. Position des duplicata-16 bits
. Couleur de l'objet - 3 bits

- Dimensions de l'objet

4 dimensions sont disponibles pour chacun des 4 objets . Elles sont décrites à la Figure 2

- Position de l'objet

Les octets de positionnement d'un objet CH et CV représentent les coordonnées horizontales et verticales de l'objet . L'origine des coordonnées se trouve en haut et à gauche sur l'écran . Le point le plus bas à droite a ses coordonnées CH autour de 170 et CV = 254 .
Les changements de valeurs de CH et CV s'opèrent durant le retour trame (Voir Chapitre V, parag. III).



Ces zones sont des images de 1FC0-1FCF

Les cases mémoires d'adresse 1F0E, 1F0F, 1F1E, 1F1F, 1F4E à 1F6F, 1FAD (en tout 37 octets) sont des mémoires tampon .

FIGURE 1

Octet

Octet		
0 1 2 3 4 5 6 7 8 9	FORME DE L'OBJET	PROGRAMMEE SUR UNE MATRICE DE 10 LIGNES SUR 8 COLONNES
A	CH	Coordonnée Horizontale de l'o
B	CHD	Coordonnée Horiz. des Duplica
C	CV	Coordonnée Verticale de l'obj
D	CVD	Coordonnée Vertic. des Duplic

E = Ecriture
L = Lecture

	N° BIT								L ou E	DESCRIPTION
Octet	7	6	5	4	3	2	1	0		
1FC0	DIM 4		DIM 3		DIM 2		DIM 1		E	Dimension des 4 objets
1FC1			C1 C2 C3 Couleur 1		C1 C2 C3 Couleur 2				E	Couleur des 4 objets
1FC2			C1 C2 C3 Couleur 3		C1 C2 C3 Couleur 4					
1FC3							FORM	POS	E	Format et position du sco
1FC4										
1FC5										
1FC6	Coul. décor			BVD	Coul. fond				E	Couleurs fond et décor
1FC7	SON								E	Fréquence du son
1FC8	N1				N2				E	Valeur des 4 digits de score
1FC9	N3				N4					
1FCA	OBJ/DEC 1 2 3 4				ETAT V OBJ 1 2 3 4				L	2 octets de reconnaissanc des collisions et de l'ét de visualisation des obje
1FCB		VALIDE	COLLISION INTER OBJET 1/2	1/3	1/4	2/3	2/4	3/4		
1FCC	POT 1								L	2 octets de valeurs de po tentiomètres permettant d'utiliser les commandes
1FCD	POT 2									
1FCE										
1FCF										

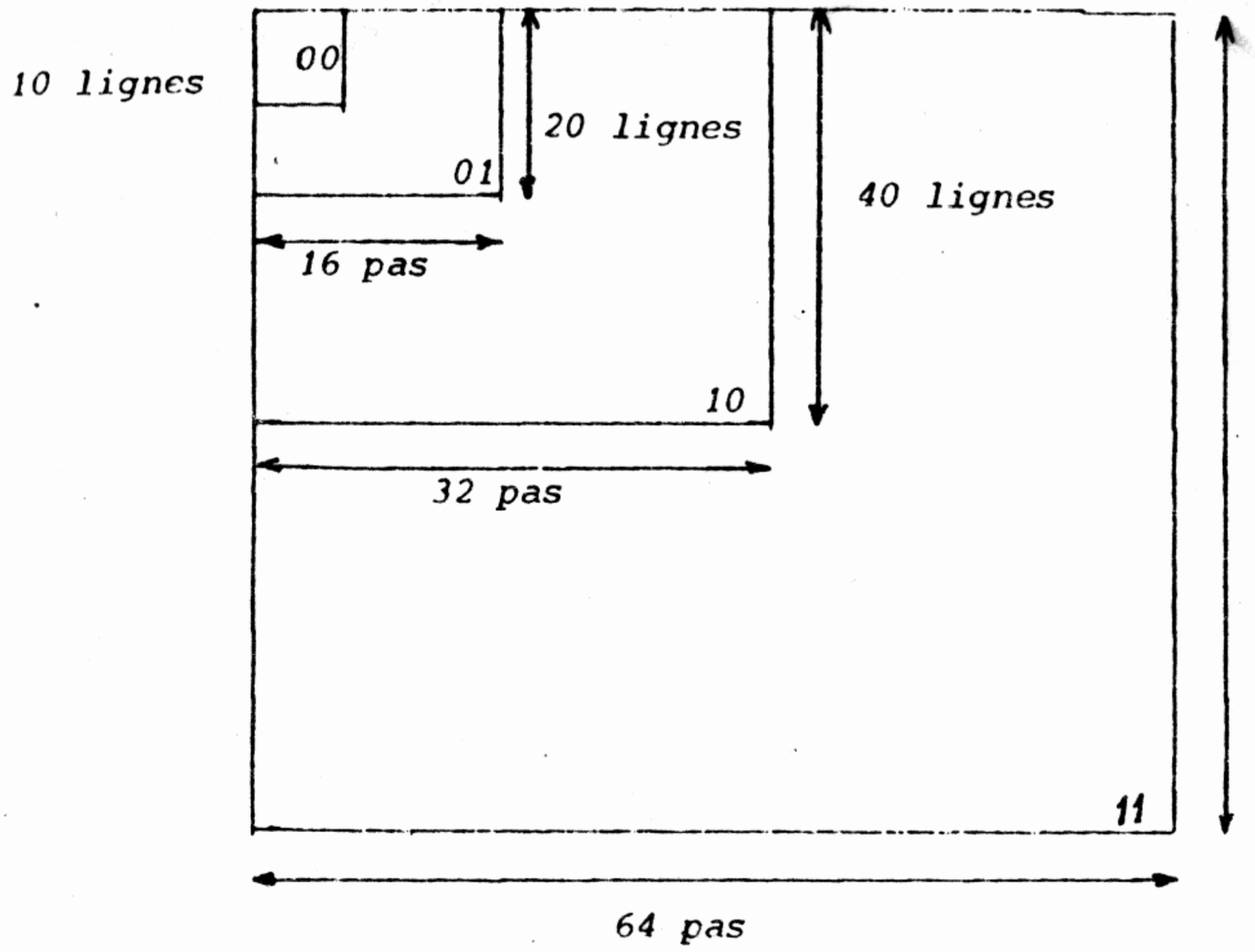


FIGURE 2

XX = valeur de la dimension

FORME DE L'OBJET

EXEMPLE

Octet	7	6	5	4	3	2	1	0	Valeur de l'octet Binaire	Hexa
0			▨				▨		00100010	22
1			▨				▨		00100010	22
2			▨				▨		00100010	22
3			▨				▨		00100010	22
4			▨				▨		00100010	22
5			▨				▨		00100010	22
6			▨	▨	▨	▨	▨		00111110	3E
7					▨				00001000	08
8					▨				00001000	08
9			▨	▨	▨	▨	▨		00111110	3E

La forme de l'objet est décrite dans un champ de 8 X 10 = 80 bits (8 bits, 10 lignes) . Les bits mis à un dessinent l'objet .

- Position des duplicata

Une ou plusieurs copies d'un objet peuvent être visualisées en chargeant CHD par une coordonnée horizontale et CVD par un nombre N correspondant à (N+1) lignes à sauter après l'affichage de l'original . Dans le cas où aucune ligne ne serait à sauter, N = 255 .
La Figure 3 détaille 2 objets avec duplicata de taille = 00 .
L'objet 1 commence à 42H,36V et sa 1ère copie à 30H,56V ,
donc N+1 = 56 - (36 + 10) = 10 d'où N = 9 = CVD .
La duplication continue jusqu'à la fin de l'image .

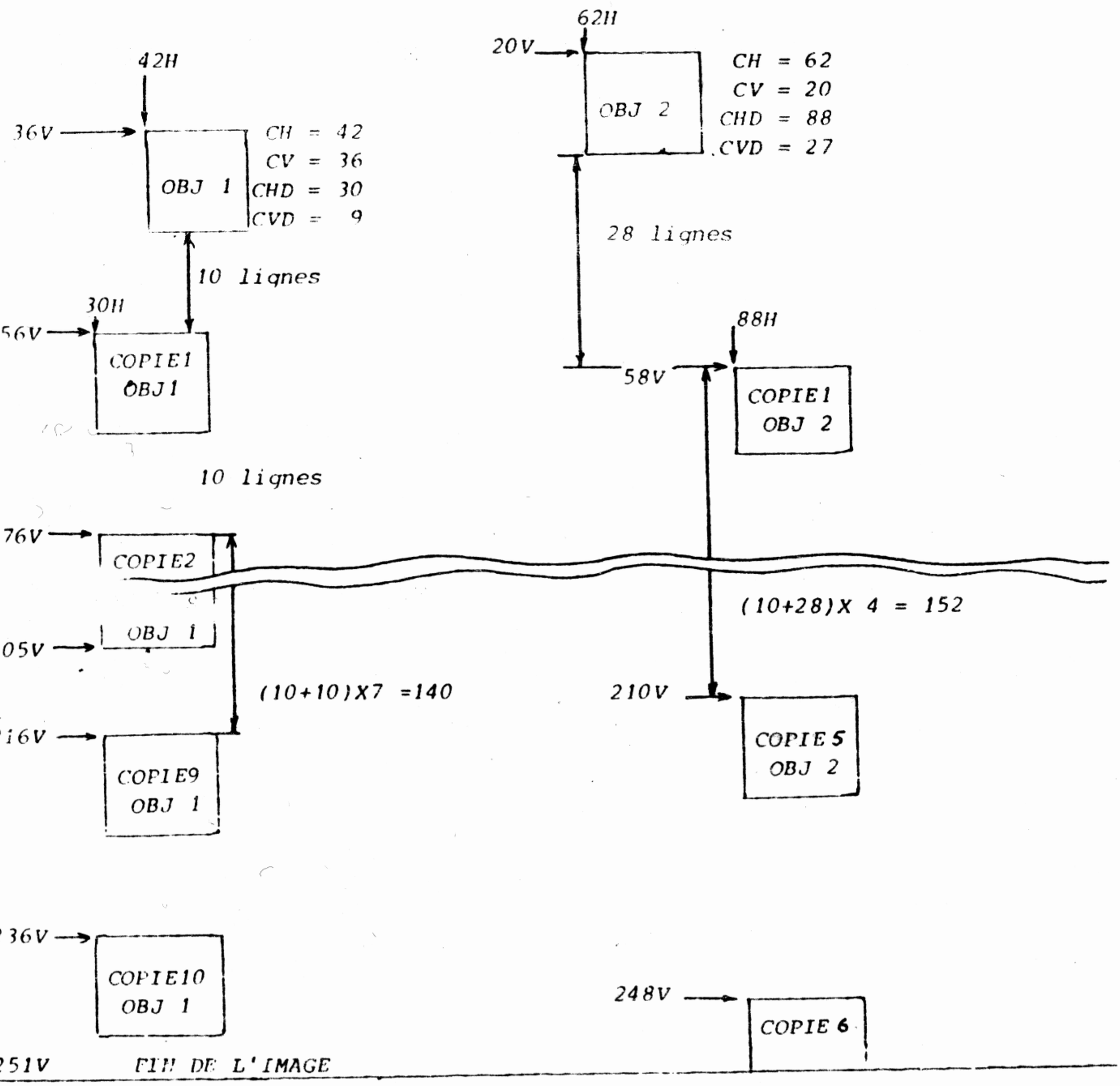


FIGURE 3

Contrairement à CH et CV, CHD et CVD peuvent être changés pendant l'image . CHD doit être changé APRES que le bit d'état de visualisation d'objet indique l'affichage complet de l'objet, tandis que CVD doit être changé AVANT .

- Couleur de l'objet

C1, C2, C3 déterminent la couleur de chaque objet . 8 couleurs sont possibles :

C1	C2	C3	Couleurs
0	0	0	blanc
0	0	1	jaune
0	1	0	magenta
0	1	1	rouge
1	0	0	cyan
1	0	1	vert
1	1	0	bleu
1	1	1	noir

- Etat de visualisation des objets

4 bits indiquent l'état d'affichage de chaque objet (complet - incomplet) . Le signal est disponible sur la dernière ligne de l'objet ou des duplicata .

- Collision Objet-Décor

4 bits d'état fournissent les indications de collision objet ou duplicata - décor . Lors d'une collision objet-décor, le bit correspondant se met à 1 . Il revient à zéro lorsque l'octet est accédé ou lors du retour trame (Voir Chapitre V,parag. III)

- Collision Inter Objet

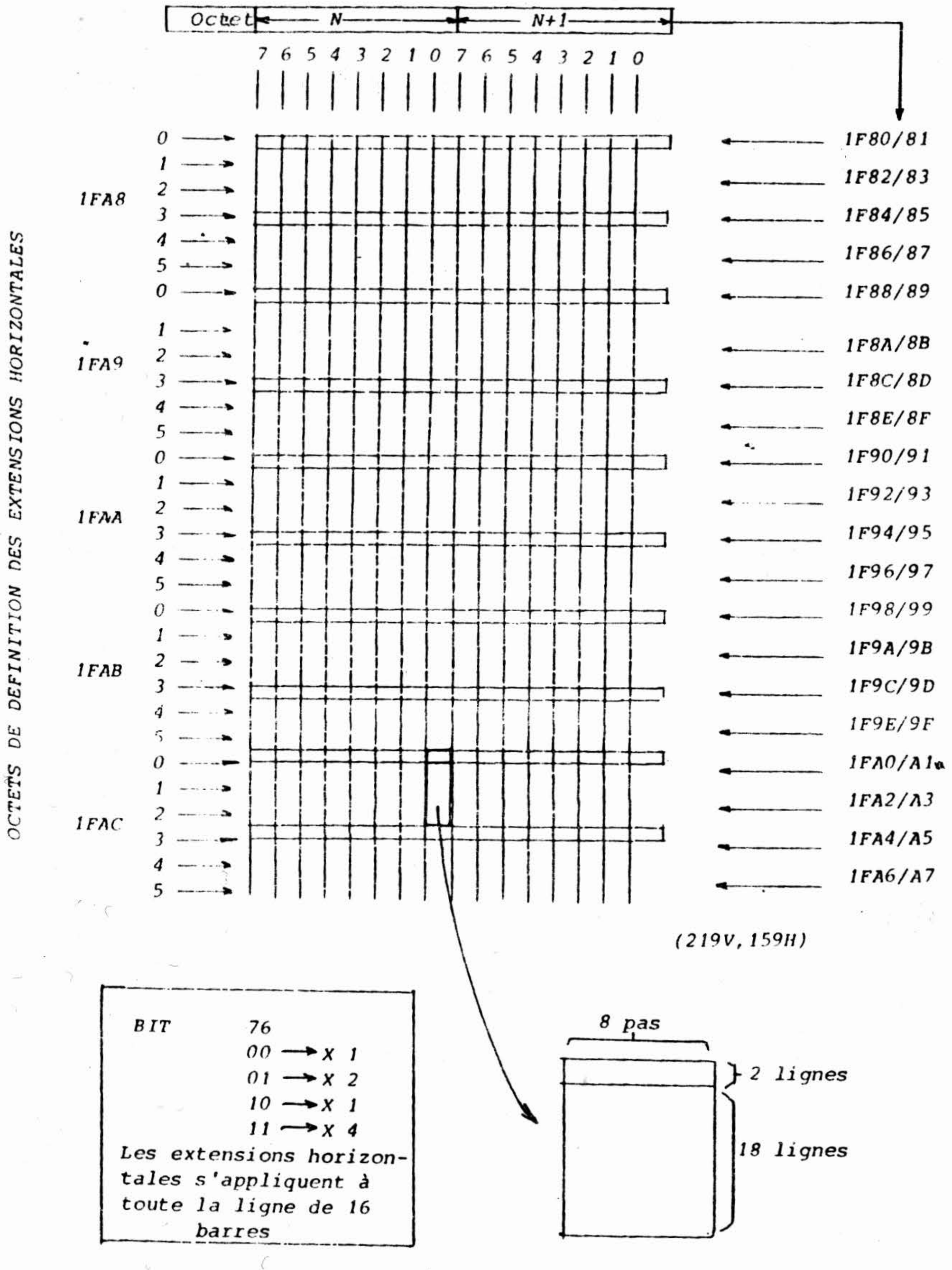
6 bits d'état fournissent la détection de collision inter-objet (ou duplicata). Lors d'une collision inter-objet, le bit correspondant se met à 1 . Il revient à zéro lorsqu'on accède à l'octet d'adresse 1FCB ou lors du retour trame . Le bit VRLE suit le niveau du retour trame ou revient à zéro lorsqu'on accè à l'octet 1FCB .

2) Vidéo Génération du Décor

Comme le montre la Figure 1, le décor est décrit par cinq éléments de données :
- Définition des barres verticales -320 bits .
- Définition des barres horizontales - 40 bits
- Validation Décor (BVD) - 1 bit
- Couleur Décor et Fond - 2 groupes de 3 bits chacun

- Définition des barres verticales

370 barres verticales distinctes peuvent être visualisées . Les barres sont arrangées en 20 groupes de 16 barres chacun (Voir Figure 4) . Chaque barre a une largeur de 1 pas d'horloge (point - unité) et dans chaque groupe, 2 barres consécutives sont espacées de 7 pas d'horloge . Les 20 groupes sont structurés en 10 groupes de paires de barres verticales .



DEFINITION DU DECOR

FIGURE 4

Chaque paire de barres verticales comprend une barre haute de 2 lignes et une barre haute de 18 lignes .
Une zone mémoire de 40 octets d'adresses 1F80 à 1FA7 définissent ces barres verticales .

- *Définition des barres horizontales*

Les barres horizontales se définissent par l'extension des barres verticales programmées . Les 40 octets de définition des barres verticales décrivent des barres de largeur d'un pas d'horloge .
Cinq octets supplémentaires permettent d'étendre cette largeur en 2,4 ou 8 pas .
Les adresses de ces octets sont :
1FA8, 1FA9, 1FAA, 1FAB, 1FAC ,
et la figure 4 montre l'affectation de ces octets sur la grille de décor .

Analysons la fonction de ces octets bit par bit :

- *Bit n° 7 et bit N° 6 : ces bits servent à définir la largeur intermédiaire des barres :*

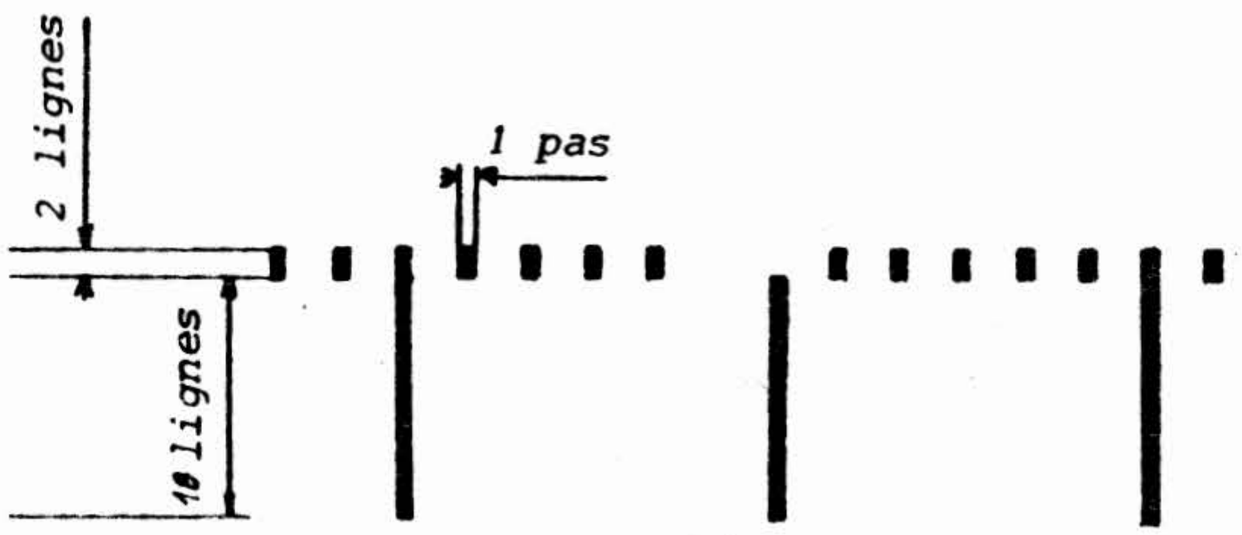
Bit 7	Bit 6	largeur des barres
0	0	1
0	1	2
1	0	1
1	1	4

- *Les bits n° 0 à n° 5 servent à définir la largeur de 8 pas d'horloge (bit positionné à 1) ou de 1, 2, ou 4 pas d'horloge, suivant les bits 7 et 6 (bit positionné à 0) .*
Le découpage des zones d'affectation des bits 0 à 5 est montré à la Figure 4 .
On peut voir que :
le bit 0 et le bit 3 affectent chacun 2 lignes entières du décor, les bits 1, 2, 4 et 5 affectent chacun 4 lignes entières du décor . L'extension n'est effective que pour les barres verticales programmées .

Exemple :

	7	6	5	4	3	2	1	0
1F80	1	1	1	1	1	1	1	0
1F82	0	0	1	0	0	0	0	0

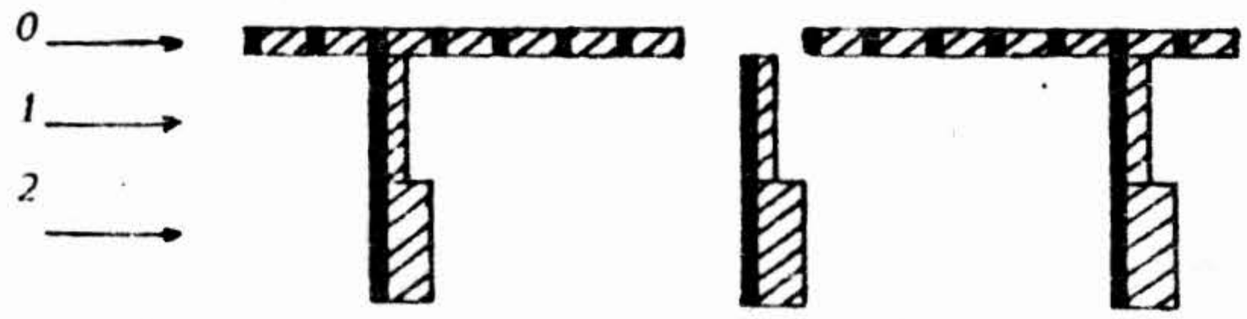
	7	6	5	4	3	2	1	0
1F81	0	1	1	1	1	1	1	1
1F83	1	0	0	0	0	0	1	0



DEFINITION DES BARRES VERTICALES

	7	6	5	4	3	2	1	0
1FA8	1	1	0	1	0	1	0	1

EXTENSIONS HORIZONTALES



Remarque : Les parties horizontales étendues sont hachurées pour faciliter l'observation .

Dans cet exemple, nous avons choisi la 1ère paire de barres d'adresses verticales 1F80/81 et 1F82/83 .

L'extension horizontale est générée par le contenu de la mémoire d'adresse 1FA8 . Pour ces 2 paires de barres verticales, seuls les bits 0, 1, 2, 6 et 7 sont utilisés .

Le bit 0 = 1 : les barres des 2 premières lignes s'étendent à une largeur de 8 pas d'horloge .
Le bit 1 = 0 : les barres des 4 lignes suivantes s'étendent à une largeur de 4 pas d'horloge car bit 7 et 6 = 11
Le bit 2 = 1 : les barres des 4 lignes suivantes s'étendent à une largeur de 8 pas d'horloge .

- Bit de Validation du Décor (BVD)

On autorise la génération du décor en positionnant à 1 le bit 3 de 1FC6 . (Voir figure 1)

- Couleur du Décor et du Fond

La couleur du décor est définie par les bits 4 à 6 de 1FC6 . La couleur du fond (derrière le décor) est définie par les bits 0 à 2 ; lorsque le décor n'est pas autorisé, une couleur de "111" est générée à la fois pour le décor et pour le fond . Le codage des couleurs C1 C2 C3 est inversé par rapport à celui des objets et du score .

C1	C2	C3	Couleurs
0	0	0	noir
0	0	1	bleu
0	1	0	vert
0	1	1	cyan
1	0	0	rouge
1	0	1	magenta
1	1	0	jaune
1	1	1	blanc

Un exemple de programmation du décor est exposé au Chapitre V, paragraph 1 : affichage d'un échiquier .

3) Score

Quatre digits de score peuvent être affichés soit en 2 parties de 2 digits, soit en un bloc de 4 digits . Les 4 digits peuvent être affichés soit au bas soit au haut de l'écran . Le format et la position du score sont controlés par les bits FORM et POS de 1FC3 (Voir Figure 1) .
FORM (bit1 de 1FC3) mis à zéro détermine 2 parties de 2 digits .
FORM mis à 1 détermine un bloc de 4 digits .
POS (Bit 0 de 1FC3) mis à zéro détermine l'affichage au haut de l'écran .
POS mis à 1 détermine l'affichage au bas de l'écran .
Les valeurs des scores sont écrites aux adresses 1FC8 et 1FC9 dans les zones N1, N2, N3 et N4 . Elles sont affichées dans cet ordre sur l'écran .

Les valeurs de A à F pour N1 à N4 ne donnent pas d'affichage de score . La couleur des quatre digits est la même que la couleur du décor .
Les 2 positions peuvent être affichées simultanément en changeant l'octet de contrôle POS entre les lignes 40 et 199 et pendant le retour trame .

4) Son

La fréquence du son est programmée par la valeur N stockée à l'adresse 1FC7 . Une valeur de zéro annule le son . Les autres valeurs N donnent un son de période égale à 2 (N+1)TH où TH est la période ligne (TH = 64 microsecondes)
Valeurs interdites pour N =

20 à 24
40 à 44
60 à 64
A0 à A4

5) Convertisseurs Analogiques/Numériques (A/N)

Deux convertisseurs internes A/N permettent l'utilisation des potentiomètres (commandes à distance type manche à balai) .Les conversions se font pendant la durée de l'image et sont stockées à 1FCC et 1FCD . Elles ne sont disponibles que pendant le retour trame .

V - QUELQUES EXEMPLES DE PROGRAMMES

Nous étudierons quelques programmes (en anglais ROUTINES) d'affichage de décor (EX : échiquier), d'utilisation de sous-programmes (SUBROUTINES) contenus dans le moniteur, puis nous verrons un programme de jeu de la goutte d'eau qui permettra de bien situer les possibilités de programmation de HOBBY COMPUTER . Enfin, quelques programmes utiles seront exposés dans ce chapitre .

I - Sous-Programmes disponibles dans le Moniteur

C'est le programme moniteur qui génère les fonctions de commandes PC, Ad, +/-, R, W, RX et d'utilisation des claviers, contrôle et gère l'écran et permet d'accèder aux zones mémoires utilisables . Il est localisé de l'adresse 0000 à l'adresse 0800 et occupe pour ses données la zone 0800 - 08BF ..

Cinq sous-programmes contenus dans le moniteur peuvent être réutilisés :

1) KBSCAN = (Keyboard Scan) porgramme d'analyse et de détection des claviers .
Adresse d'entrée : 0181
Données : aucune (Remarque : ce programme doit être utilisé pendant le retour trame)
Registres utilisés : R0, Banque 0 (RS = 0) : R1,R2, R3
Niveaux de sous-programmes imbriqués : 2
Sorties : RKBST adresse 089D clavier droit
LKBST adresse 089E clavier gauche
MKBST adresse 089F , R1 tous claviers réunis .
Un exemple utilisant KBSCAN est traité dans ce chapitre, paragraphe IV .

2) CONVRT = programme de conversion ligne à image .
Adresse d'entrée : 020E
Données : LINE 8 adresses de codes caractères 0890 à 0897
Les caractères disponibles sont donnés avec leur code dans le tableau ci-dessous :

Caractères	Code Hexa	Caractères	Code Hexa	Caractères	Code Hexa
0	00	A	0A	P	14
1	01	b	0b	r	15
2	02	c	0c	=	16
3	03	d	0d	Espace	17
4	04	E	0E	+	18
5	05	F	0F	-	19
6	06	G	10	:	1A
7	07	L	11	x	1b
8	08	I	12		
9	09	n	13		

Registres utilisés : R0, R1, R2, R3 .
Niveaux de sous-programmes imbriqués : 2
Sortie : l'image des caractères stockée dans une zone tampon du moniteur est affichable sur écran par l'intermédiaire du sous-programme OUTPUT (voir ci-après)

3) OUTPUT : affichage de caractères sur l'écran
Adresse d'entrée : 0055
Données : la sortie du programme précédent CONVRT
Remarque : ce programme doit être appelé pendant le retour trame .
Registres utilisés : R0, R1, R2 .
Niveaux de sous-programmes utilisés : 1
Sortie : Affichage d'une ligne de caractères au bas de l'écran

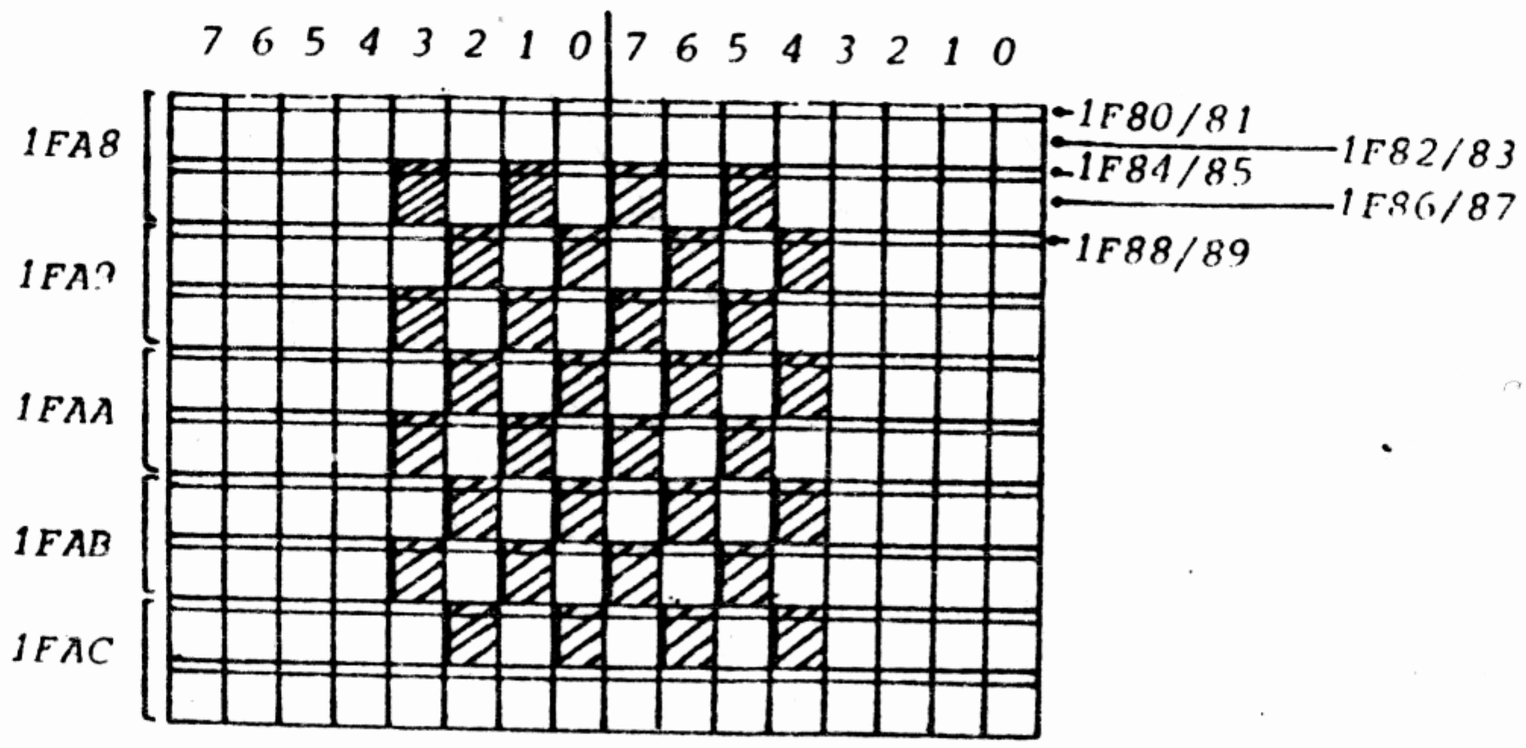
4) SCROLL : saut d'une ligne de caractères vers le haut de l'écran
Adresse d'entrée : 02CF
Données : Aucune
Niveaux de sous-programmes utilisés : 1
Sortie : Saut d'une ligne vers le haut . La ligne suivante affichée est initialisée par des espaces (code hexa : 17, 17, 17, 17, 17, 17 17, 17)

5) CLROBJ : (Clear Object) - Sous-programme d'effaçage des objets du PVI .
Adresse d'entrée : 01EE
Données : Aucune
Registres utilisés : R0, R2
Niveaux de sous-programmes utilisés : 1
Sortie : Effaçage des 4 objets du PVI et de leurs coordonnées .

Nous verrons dans le premier exemple une utilisation du programme CLROBJ et dans le second une utilisation des programmes CONVRT, OUTPUT et SCROLL .

II - Exemples de programmes d'application

1) Le décor : un échiquier
Dessinons-le dans le cadre qui est réservé au décor



On voit qu'il y a 3 valeurs différentes pour les couples d'adresses associés aux lignes verticales ; ce sont :
- 00/00 pour 1F80/81
82/83
A4/A5
A6/A7

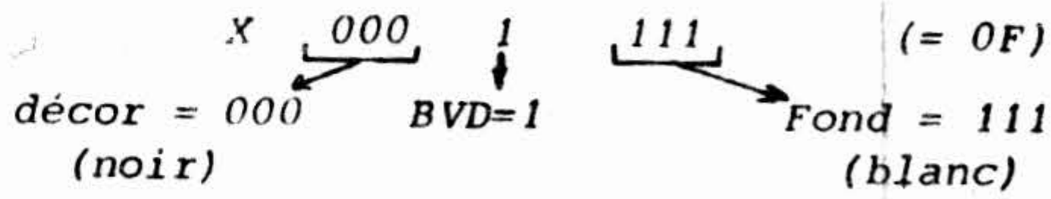
- 0A/A0 pour 1F84/85
  86/87
  8C/8D
  8E/8F
  94/95
  96/97
  9C/9D
  9E/9F

- 05/50 pour 1F88/89
  8A/8B
  90/91
  92/93
  98/99
  9A/9B
  A0/A1
  A2/A3

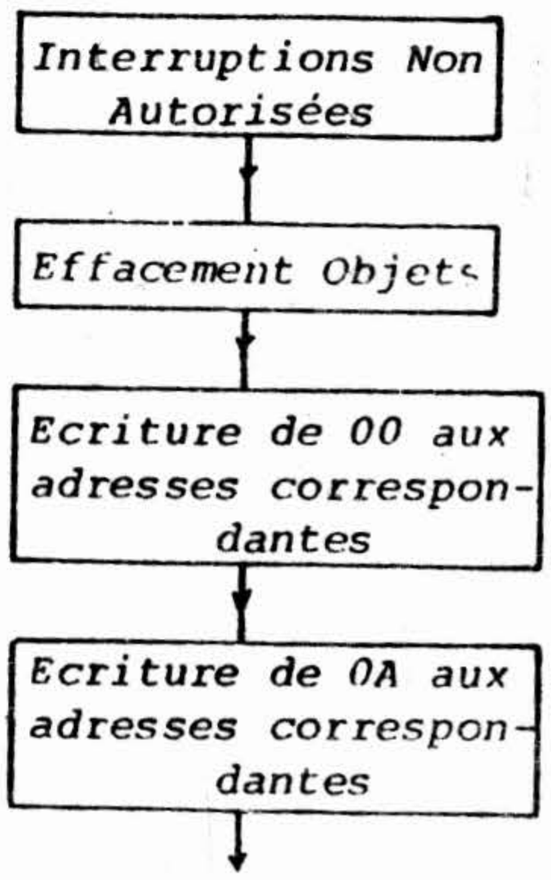
Nous devons donc écrire les valeurs 00, 0A, A0, 05 et 50 dans les zones mémoires correspondantes (1F80 à 1FA7) .
Pour les extensions horizontales (Voir Figure 4 du chapitre IV), les octets d'adresse 1FA9, 1FAA, 1FAB doivent avoir leurs bits N° 0, 1, 2, 3, 4 et 5 mis à 1 pour une largeur de 8 pas d'horloge . Les bits 6 et 7 définissant les largeurs intermédiaires (1, 2 ou 4 pas d'horloge) n'ont ici pas d'importance car on prend la plus grande largeur (8 pas) . Donc les octets d'adresse 1FA9, 1FAA, 1FAB s'écrivent :

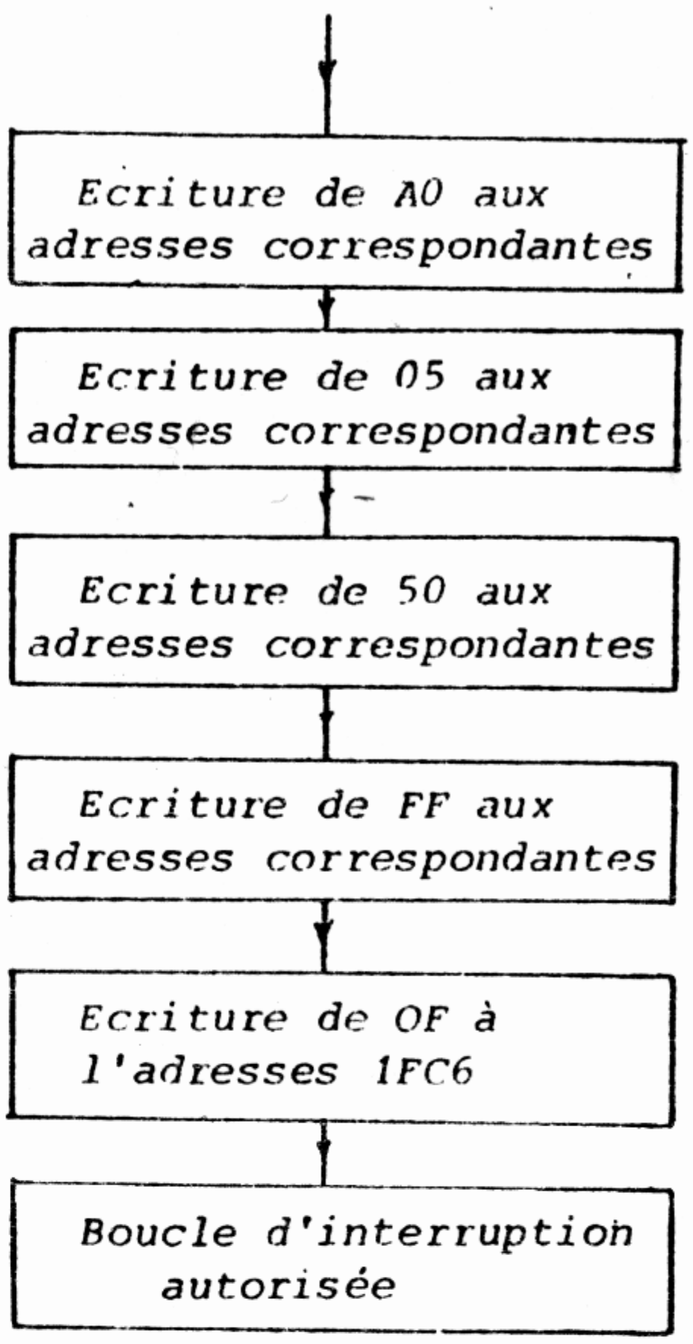
X X 1 1 1 1 1 1

(XX correspondant à une valeur quelconque) .
De même, pour les octets d'adresse 1FA8 et 1FAC, ils vont s'écrire respectivement X X 1 1 1 X X X et X X X X X 1 1 1 .
Nous voyons que pour des raisons de commodité et de facilité d'écriture, on peut prendre la valeur commune 1 1 1 1 1 1 1 1 = FF à écrire à ces 5 adresses
Enfin il ne nous reste plus qu'à trouver la valeur à stocker à l'adresse 1FC6 positionnant les couleurs du décor et du fond et permettant l'affichage de l'échiquier . Pour des carrés noirs sur un fond blanc, on a la combinaison :



Pour d'autres couleurs voir Chapitre IV, paragraphe II, 2 ,
Ecrivons l'organigramme de ce programme





Ecrivons le programme "ECHIQUIER"

ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTION	COMMENTAIRES
0900	7620		PPSU,II	II=1 pas d'interruption
0902	3F016E		BSTA,UN CLROBJ	Effacement objet
0905	20		EORZ,R0	R0 = 0
0906	0504		LODI,R1 04	04 R1
0908	CD5F80	LOOP Ø	STRA,R0 DC0 1,R1-	Stockage de 00
090B	597B		BRNR,R1 LOOP Ø	aux adresses 1F80
090D	0504		LODI,R1 04	à 1F83 et dans
090F	CD5FA4	LOOP 1	STRA,R0 DCOA,R1-	1FA4 à 1FA7
0912	597B		BRNR,R1 LOOP 1	
	040A		LODI,R0 0A	0A ⟶ R0
	C1F84		STRA,R0 DC0	┐
0919	CC1F86		STRA,R0 DC0	│
091C	CC1F8C		STRA,R0 DC0	│ Stockage de 0A aux
091F	CC1F8E		STRA,R0 DC0	│ adresses
0922	CC1F94		STRA,R0 DC0	│ correspondantes
0925	CC1F96		STRA,R0 DC0	│
0928	CC1F9C		STRA,R0 DC0	│
092B	CC1F9E		STRA,R0 DC0	┘
092E	04A0		LODI,R0 A0	A0 ⟶ R0
0930	CC1F85		STRA,R0 DC0	┐
0933	CC1F87		STRA,R0 DC0	│
0936	CC1F8D		STRA,R0 DC0	│ Stockage de A0 aux
0939	CC1F8F		STRA,R0 DC0	│ adresses
093C	CC1F95		STRA,R0 DC0	│ correspondantes
093F	CC1F97		STRA,R0 DC0	│
0942	CC1F9D		STRA,R0 DCO	│
0945	CC1F9F		STRA,R0 DC0	┘

```
0948    0405        LODI,R0 05      05 ⟶ R0
094A    CC1F88      STRA,R0 DC0  ┐
094D    CC1F8A      STRA,R0 DC0  │
0950    CC1F90      STRA,R0 DC0  │  Stockage de 05 aux
0953    CC1F92      STRA,R0 DC0  │     adresses
0956    CC1F98      STRA,R0 DC0  │  correspondantes
0959    CC1F9A      STRA,R0 DC0  │
095C    CC1FA0      STRA,R0 DC0  │
095F    CC1FA2      STRA,R0 DC0  ┘
0962    0450        LODI,R0 50      50 ⟶ R0
0964    CC1F89      STRA,R0 DC0  ┐
0967    CC1F8B      STRA,R0 DC0  │
096A    CC1F91      STRA,R0 DC0  │  Stockage de 50 aux
096D    CC1F93      STRA,R0 DC0  │     adresses
0970    CC1F99      STRA,R0 DC0  │  correspondantes
0973    CC1F9B      STRA,R0 DC0  │
0976    CC1FA1      STRA,R0 DC0  │
0979    CC1FA3      STRA,R0 DC0  ┘
097C    04FF        LODI,R0 FF      FF ⟶ R0
097E    0505        LODI,R1 05      05 ⟶ R1
0980    CD5FA8  LOOP 2  STRA,R0 DCOH,R1-  Stockage de FF
0983    597B        BRNR,R1 LOOP 2  dans 1FA8 à 1FAC
0985    040F        LODI,R0 OF      0F ⟶ R0
0987    CC1FC6      STRA,R0 CLD     Stockage de 0F à 1FC6
098A    7420    WAIT    CPSU,II     II=0 autorise les interrup-
098C    1B7C        BCTR,UN WAIT    tions . Boucle d'attente
                                    d'interruption ( Touche
                                    RAZ)
```

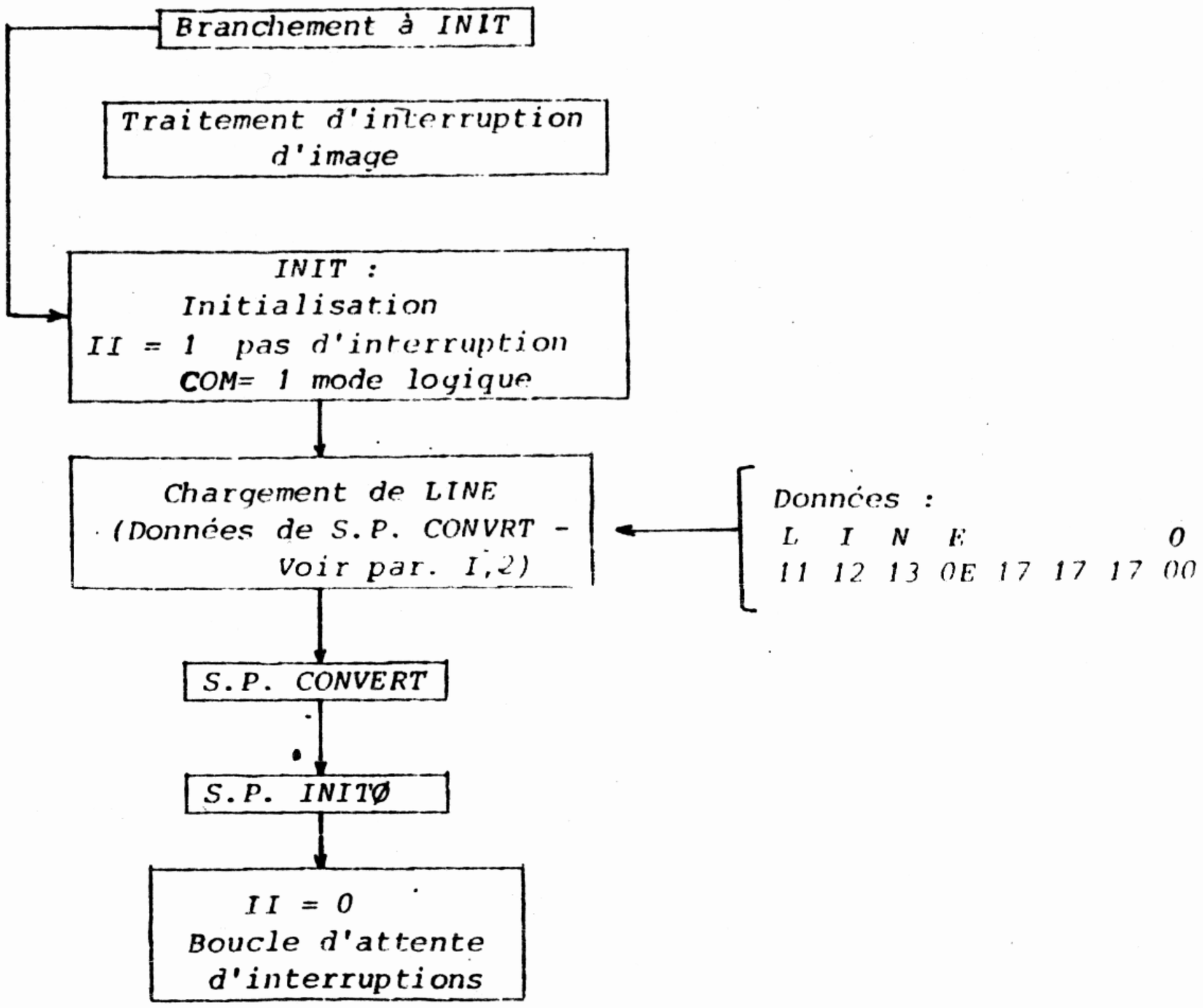
Le lecteur entrera les valeurs HEXA de l'adresse 0900 à l'adresse 098D, puis fera exécuter le programme à partir de PC = 0900 . Pour revenir en mode MONITEUR appuyer sur RAZ . Pour sauvegarder ce programme sur cassette, placer bEG =0900, END=0990 et SAd=0900 .

La comprhension de ce programme ne pose pas de problème particulier sauf, peut-être aux boucles LOOP Ø, LOOP 1 et LOOP 2 . Elles sont bâties sur une même structure, c'est -à-dire qu'une même valeur chargée dans R0 est stockée à quatre ou cinq adresses consécutives . R1 est le registre d'index permettant une décrémentation des adresses jsuqu'à ce que R1 soit nul (Voir Adressage Indexé Chapitre IV, paragraphe IV,2) .

2) Programme d'application utilisant les sous - programmes CONVRT, OUTPUT et SCROLL

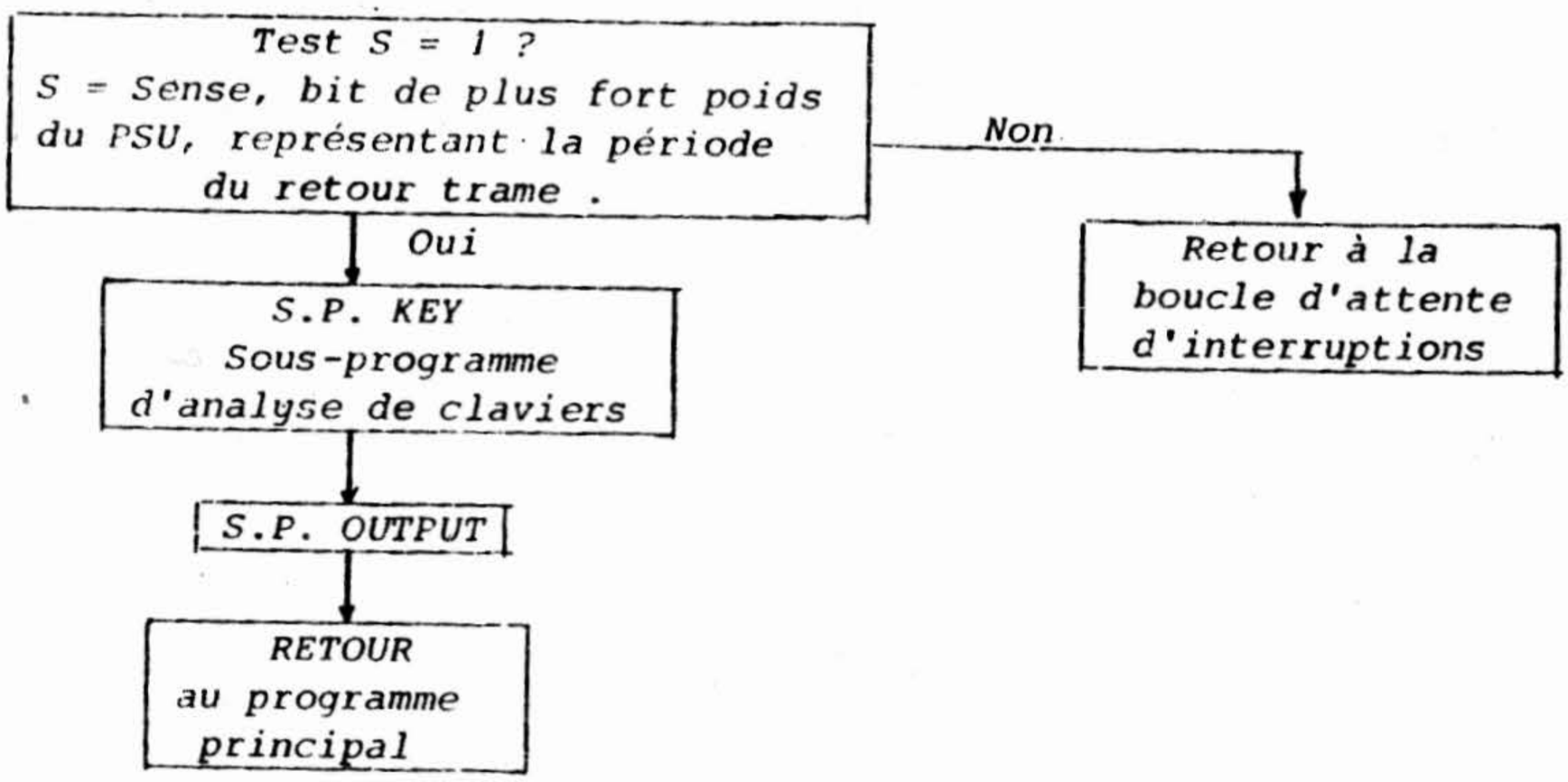
AFFICHAGE de tous les CARACTERES ALPHANUMERIQUES contenus dans le MONITEUR par action de la touche »

Organigramme du programme principal



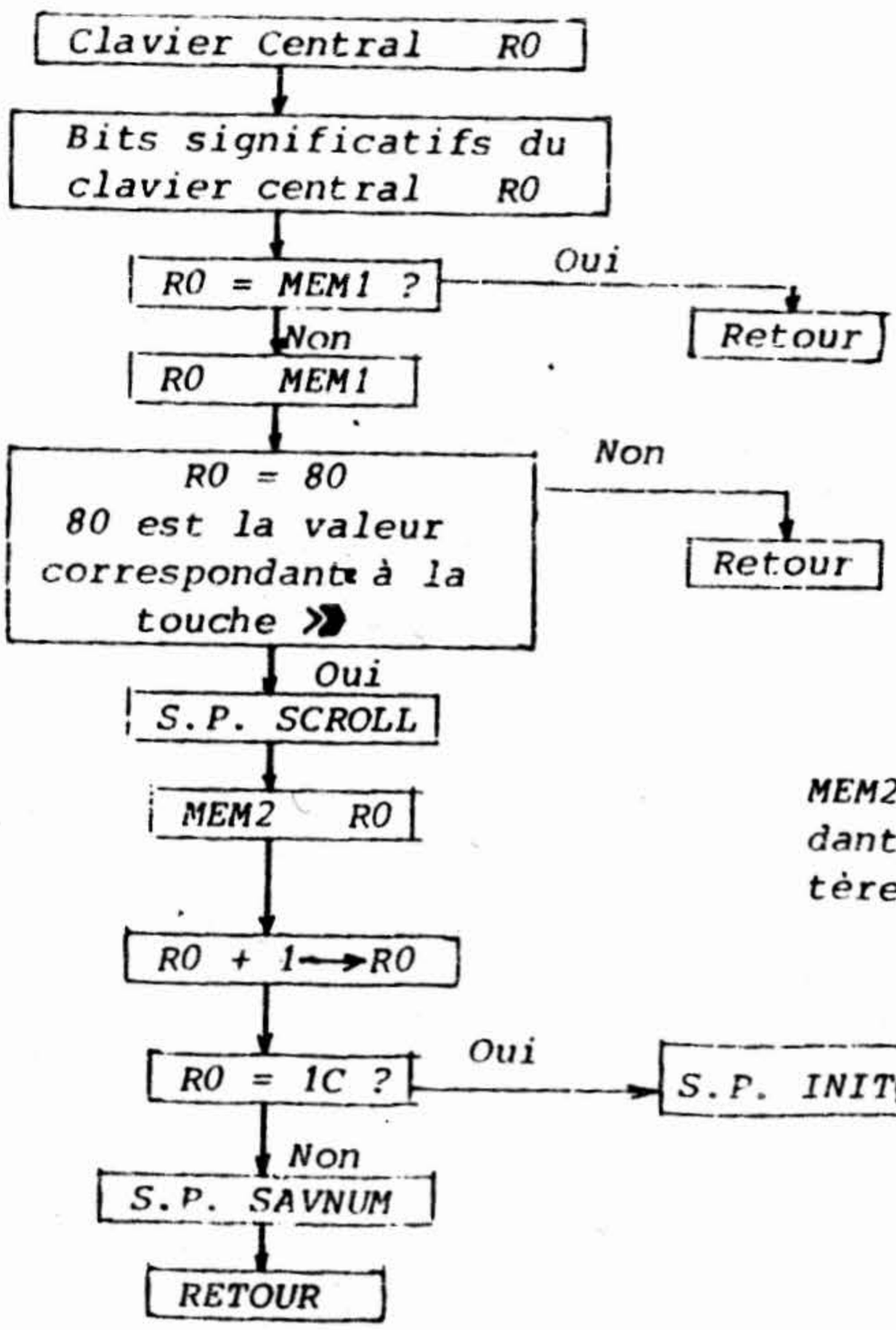
Traitement d'interruption d'Image (Voir plus loin Paragraphe III)

Les interruptions d'image se font toutes les 20 ms pendant le retour trame C'est pendant cette période que doivent se faire l'analyse des claviers et l'appel du sous-programme OUTPUT .



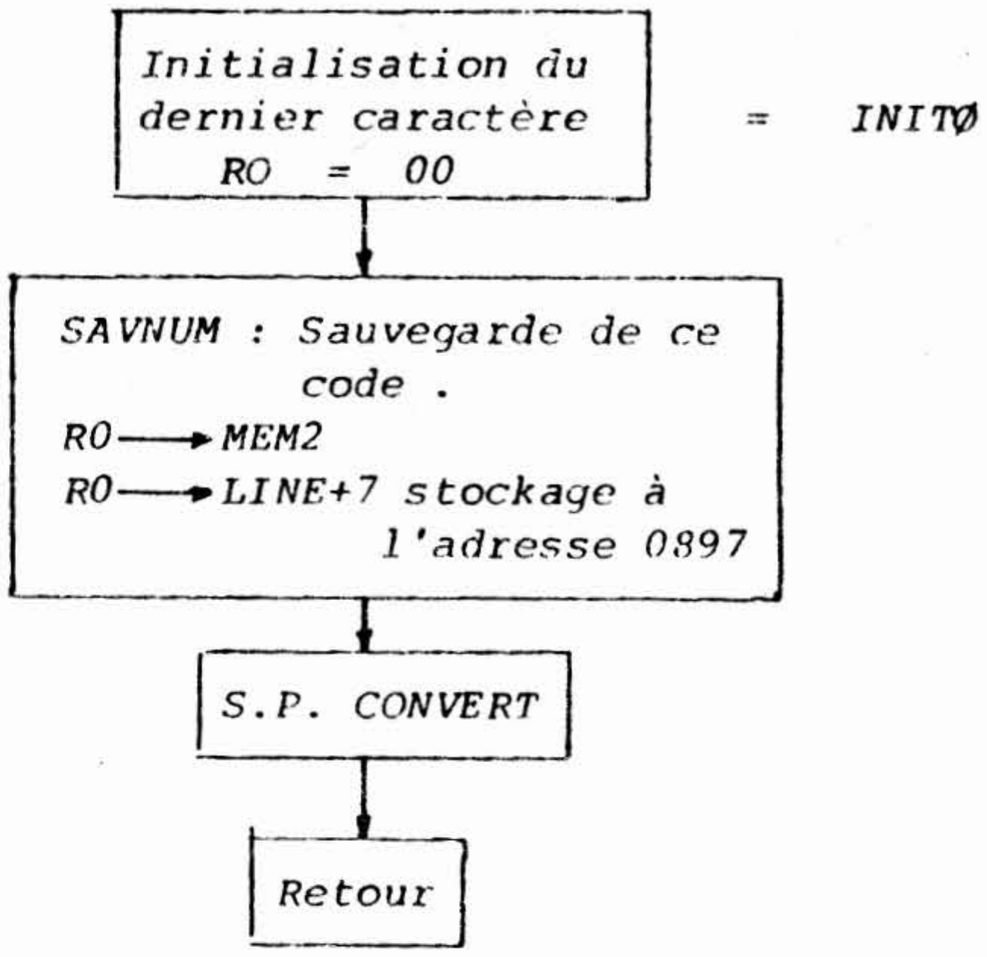
S.P. KEY

KEY est un programme d'analyse du clavier central . Pour une meilleure compréhension de l'utilisation des claviers, voir par. IV,1; l'action de la touche permet de remonter la page d'une ligne (S.P. SCROLL) et d'écrire un caractère sur la 8ème colonne .



MEM2 est une mémoire tampon gardant la valeur hexa du code caractère à écrire sur la 8ème colonne .

Les sous-programmes INITØ et SAVNUM permettent de convertir les dernières données de codes caractères pour aller ensuite s'afficher sur l'écran grâce au programme OUTPUT .



Ecrivons le Programme :

ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTION	COMMENTAIRES
0900	1F09D		BCTA, UN INIT	Branchement à INIT
0903	B480		TPSU, SENSE	Test S = 1 ?
0905	16		RETC, 1	Retour si S = 0
0906	3F0925		BSTA,UN KEY	Branchement à KEY
0909	3F0055		BSTA,UN OUTPUT	Branchement à OUTPUT
090C	17		RETC,UN	Retour
090D	7620	INIT	PPSU,II	II=1 pas d'interruption
090F	7702		PPSL,COM	COM=1 mode logique
0911	0508		LODI,R1 8	8 ⟶ R1
0913	0D494B	LOOP Ø	LODA,R0 MESØ,R1+	Chargement des valeurs
0916	CD6890		STRA,R0 LINE,R1	de MESØ dans
0919	5978		BRNR,R1 LOOP Ø	LINE
091B	3F020E		BSTA,UN CONVRT	
091E	3F0940		BSTA,UN INITØ	
0921	7420	WAIT	CPSU,II	Boucle d'attente
0923	1B7C		BCTR,UN WAIT	d'interruptions
0925	0C1E8B	KEY	LODA,R0	]
0928	44F0		ANDI,R0 F0	] Traitement
092A	EC08C0		STRA,R0 MEM1	] des rebonds
092D	14		RETC,EQ	]
092E	CC08C0		STRA,R0 MEM1	
0931	F480		TMI,R0 80	Test touche »
0933	16		RETC,N1	
0934	3F02CF		BSTA,UN SCROLL	
0937	0C08C1		LODA,R0 MEM2	R0 ⟶ MEM2
093A	8401		ADDI,R0 1	R0 + 1 ⟶ R0
093C	E41C		COMI,R0 1C	charger LINE+7

```
093E    9801                    BCFR,EQ SAVNUM      de R0 si
0940    20          INIT0       EORZ  R0            R0 < 1C
0941    CC08C1                  STRA,R0 MEM2
0944    CC0897                  STRA,R0 LINE+7
0947    3F020E                  BSTA,UN CONVRT
094A    17                      RETC,UN
094B    1112130E    MES0        DATA                Données
094F    17171700
```

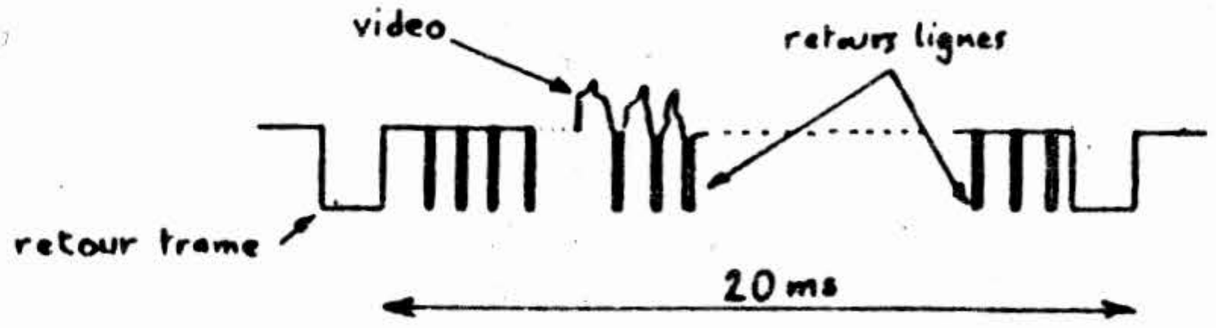
Mode EXECUTION :

Après PC = 0900 et "+", il apparait sur la dernière ligne LINE 0 . En appuyant sur » , la page remonte d'une ligne et il apparait 1 . Puis 2, 3, etc... jusqu'à revenir à 0 . Pour revenir dans le mode "MONITEUR", actionner la touche «

III - La technique de Programmation sur votre Téléviseur

Regardons quelques notions de télévision qui vont nous permettre de maîtriser notre logiciel .

Le jeu OC 2000 fonctionne sur le standard 625 lignes UHF couleur SECAM ; il émet un signal vidéo qui module une porteuse HF, ce qui permet de le recevoir sur tous les téléviseurs en France .
Ce signal vidéo a la forme suivante :



signal vidéo composite schématisé

Une image sur un téléviseur est formée par le balayage de faisceaux d'électrons traçant 625 lignes toutes dans un même sens et de haut en bas de l'écran , le spot doit revenir à son point de départ sans perturber l'image : c'est la période de retour trame .
Pendant ce retour trame, le PVI lance une interruption et le microprocesseur, s'il tient compte de cette interruption (II=0), exécute le programme d'interruption de l'utilisateur .
Le bit SENSE(S) du PSU suit l'évolution du signal de retour trame (S = 1 pendant le retour trame) .
Le bit II (N° 5 du PSU) inhibe les interruptions lorsque II = 1 .
Le moniteur prend compte de l'interruption de retour trame à l'adresse 0903 pour II = 0 . Donc, le traitement des interruptions de retour trame de l'utilisateur commencera à cette adresse .

Exemple :

ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTIONS	COMMENTAIRES
0900	1FXXXX		BCTA,UN INIT	Branchement au programme principal
0903	B480	INTERR	TPSU,SENSE	Test S+1 ?
0905	16		RETC,CC=10	Retour si S+0
				S=1 programme d'interruption .
XXXX	7620	INIT	PPSU,II	Programme principal
				II=1

Le PVI génère un autre type d'interruption qui se déroule pendant l'image, au moment du retour ligne, à la dernière ligne de génération d'un objet ou de ses duplicata (utilisation des 4 bits de poids faible d'adresse 1FCA) . Pour accéder aux objets pendant l'image on procède comme sur l'exemple suivant avec l'objet N° 1 :

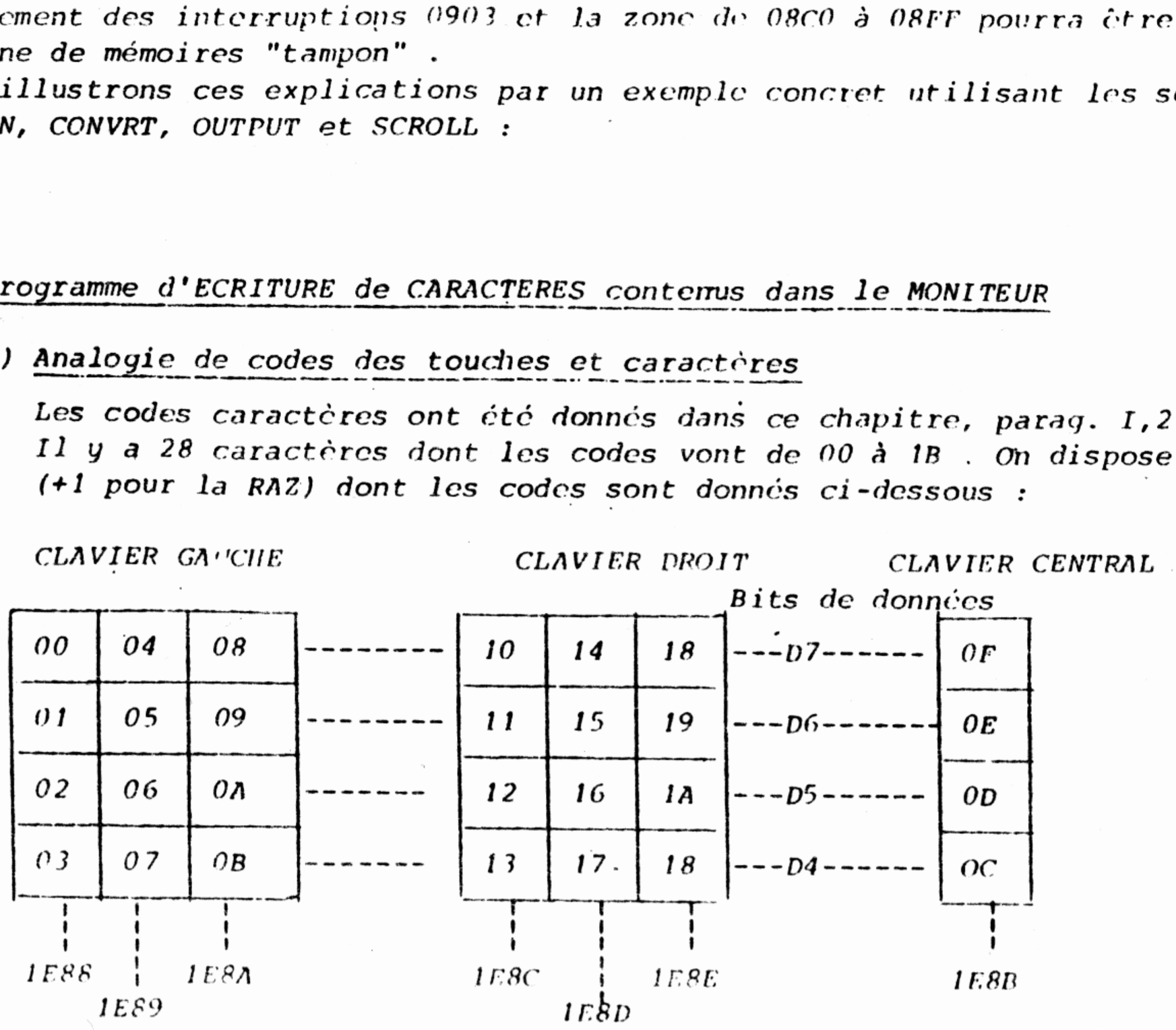
ADRESSE	CODE HEXA	ETIQUETTE	INSTRUCTIONS	COMMENTAIRES
0900	1FXXXX	START	BSTA,UN INIT	
0903	B480		TPU,SENSE	Si retour trame
0905	18XX		BCTR,CC=00 INTERT	Branchement à INTERT
0907	0C1FCA		LODA,R0 1FCA	
090A	F408		TMI,R0 8 = OBJ1	
090C	16		RETC,CC = 10	

En général, l'adresse de début de programme utilisateur sera 0900, l'adresse de traitement des interruptions 0903 et la zone de 08C0 à 08FF pourra être utilisée comme zone de mémoires "tampon" .
Nous illustrons ces explications par un exemple concret utilisant les sous-programmes KBSCAN, CONVRT, OUTPUT et SCROLL :

IV - Programme d'ECRITURE de CARACTERES contenus dans le MONITEUR

1) Analogie de codes des touches et caractères

Les codes caractères ont été donnés dans ce chapitre, parag. I,2 (CONVRT) .
Il y a 28 caractères dont les codes vont de 00 à 1B . On dispose de 28 touches (+1 pour la RAZ) dont les codes sont donnés ci-dessous :



L'utilisateur a le choix entre 2 possibilités d'utilisation des claviers :

- Soit en scrutant parmi les 7 adresses (1E88 à 1E8E) la bonne valeur hexa (10, 20, 40 ou 80) .

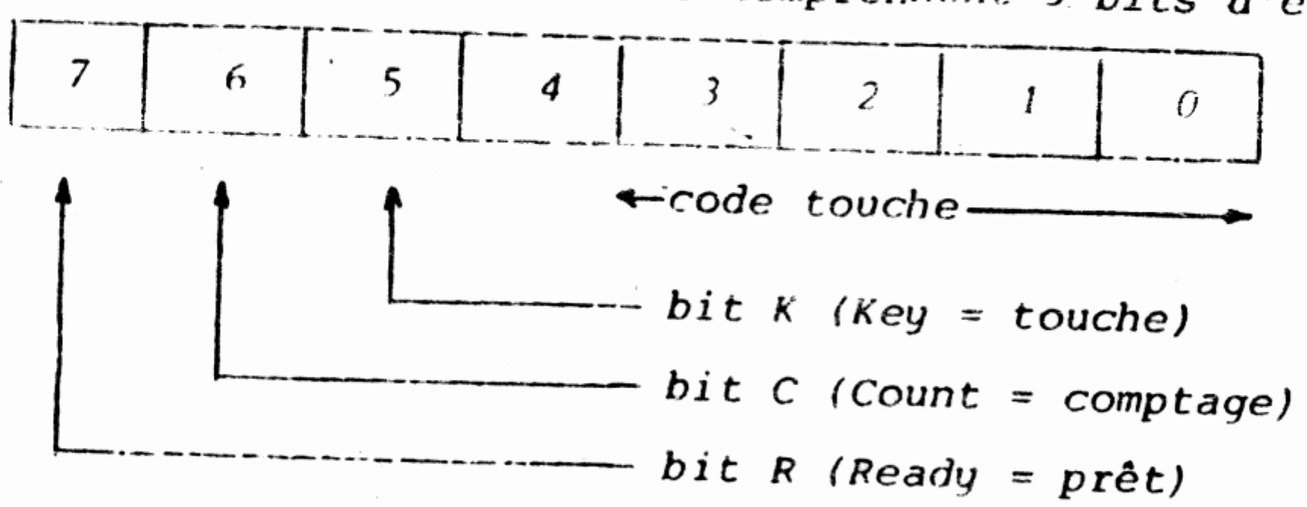
- Soit en utilisant le sous-programme KBSCAN qui donne le code (00 à 1B) d'une touche enfoncée . Nous remarquons que les codes des touches sont les mêmes que ceux des caractères . Nous utilisons cette remarque pour construire notre programme .

2) Le sous-programme KBSCAN (KEYBOARD SCAN : balayage des claviers)

KBSCAN doit être appelé pendant le retour trame dans le programme d'interruption . Il délivre 3 mots d'état reipésentant le code et l'état de la touche enfoncée . Ces 3 mots sont aux adresses :

089D	=	RKBST	Valeur de l'état du clavier droit .
089E	=	LKBST	Valeur de l'état du clavier gauche
089F	=	MKBST	Valeur de l'état des 3 claviers réunis .
R1	=	MKBST	

Les 8 bits de chacun des 3 mots comprennent 3 bits d'état et 5 bits de code :



5 bits de code touche correspondent aux codes 00 à 1B

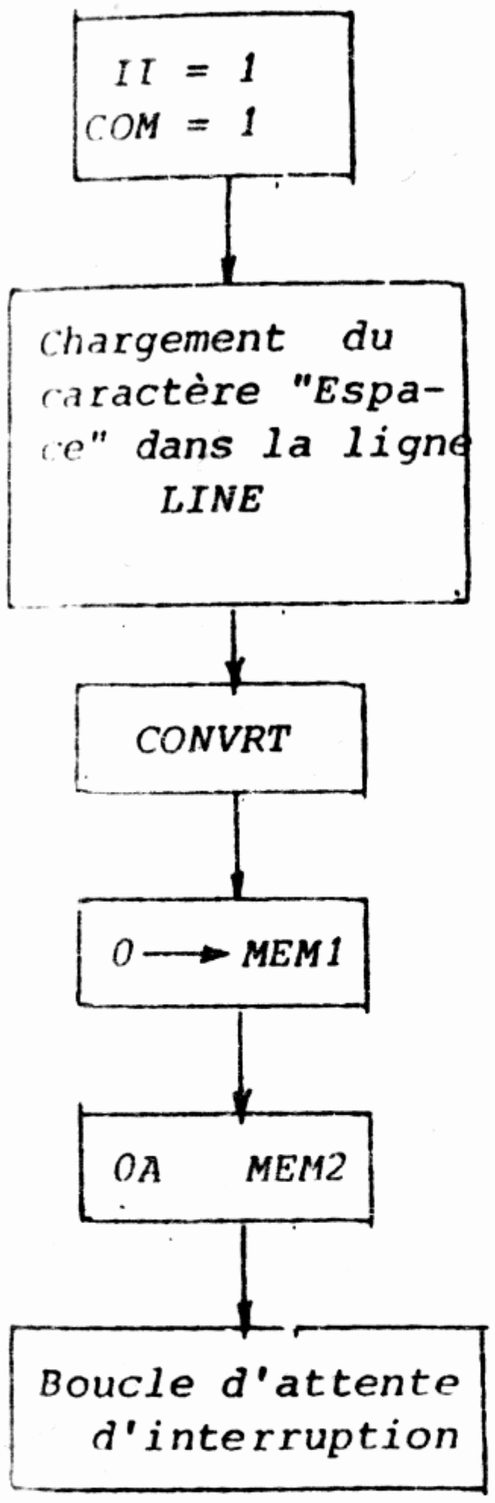
- Le bit R doit être remis à zéro si une nouvelle touche est demandée par le programme . Le nouveau code est valide si après le programme KBSCAN, le bit R est à un .

- Le bit C indique le premier et le second passage du balayage KBSCAN . Ce bit sert à éliminer le rebond .

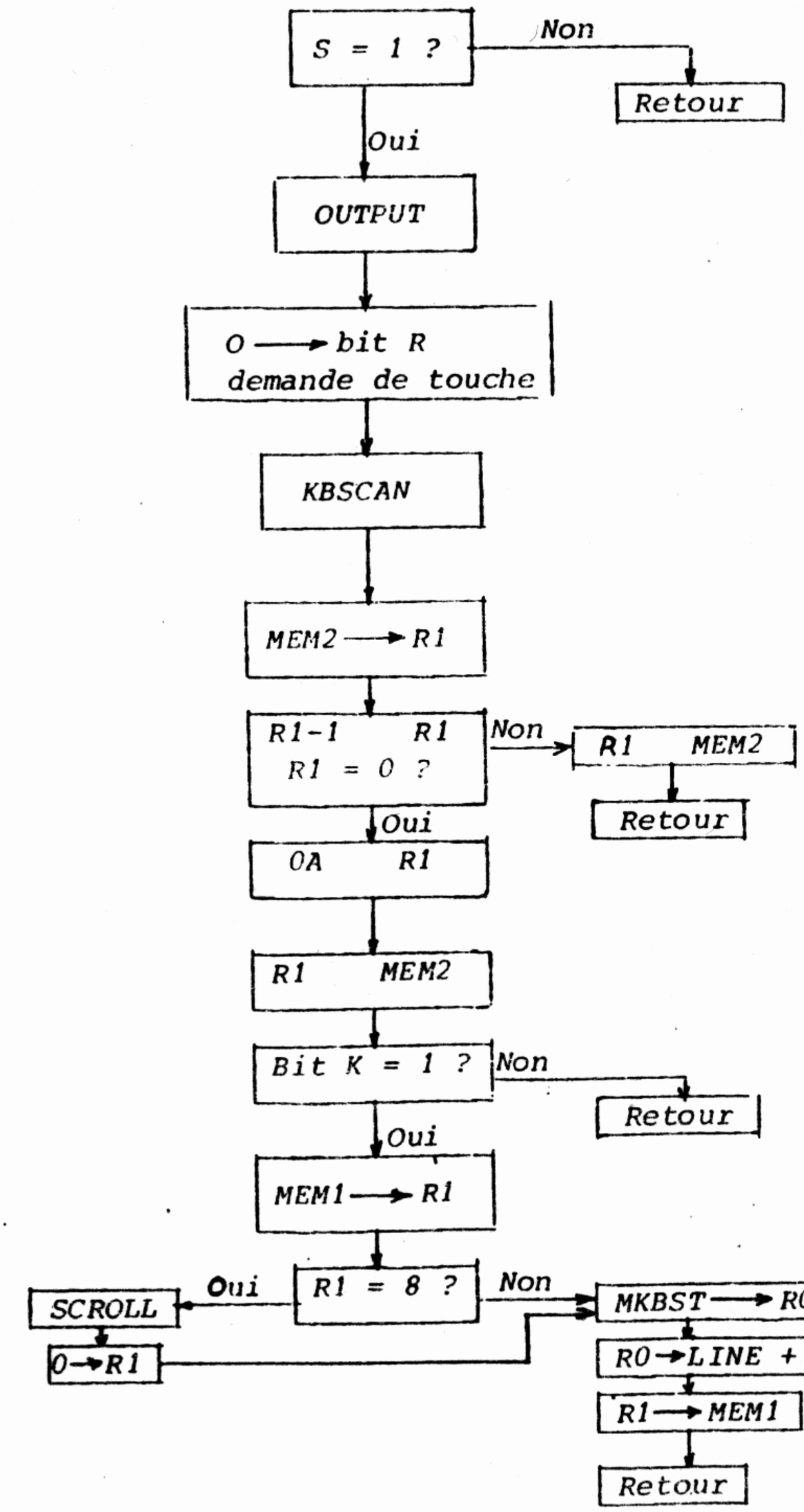
- Le bit K indique qu'une des touches est enfoncée .

3) Organigramme du programme d'ECRITURE de CARACTERES

Programme Principal



Programme d'interruption



MEM1 et MEM2 sont des mémoires tampon d'adresses: 08C0 et 08C1
MEM1 garde en mémoire la valeur de la position du caractère à afficher .
MEM2 sert à éviter les rebonds de la touche enfoncée pendant un laps de temps de 0A = 10 périodes trame .

Ecrivons le programme :

ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTIONS	COMMENTAIRES
0900	1F091C		BCTA,UN INIT	Branchement à INIT
0903	B480		TPSU,SENS	Test S = 1 ?
0905	16		RETC,N1	Retour si S = 0
0906	3F0055		BSTA,UN OUTPUT	Branchement à S.P. OUTPUT
0909	0503		LODI,R1 3	3 ⟶ R1
090B	0D489D	RLMKB	LODA,R0, R1-	Chargement et remise à
090E	447F		ANDI,R0 7F	zéro du bit R de
0910	CD689D		STRA,R0, R1	RKBST,LKBST,MKBST
0913	5976		BRNR,R1 RLMKB	
0915	3F0181		BSTA,UN KBSCAN	Branch. à S.P. KBSCAN
0918	3F0939		BSTA,UN KBCRT	Branch à KBCRT
091B	17		RETC,UN	Retour à la boucle d'attente .
091C	7620	INIT	PPSU,II	II = 1
091E	7702		PPSL,COM	COM = 1
0920	0417		LODI,R0 17	17 ⟶ R0
0922	0508		LODI,R1 8	8 ⟶ R1
0924	CD4890	LOOPØ	STRA,R0,LINE,R1-	Chargement de "ESPACE"
0927	597B		BRNR,R1,LOOPØ	dans.LINE
0929	3F020E		BSTA,UN CONVRT	Branch. à S.P. CONVRT
092C	20		EORZ,R0	0 ⟶ R0
092D	CC08C0		STRA,R0 MEM1	R0 ⟶ MEM1
0930	040A		LODI,R0 0A	0A ⟶ R0
0932	CC08C1		STRA,R0 MEM2	R0 ⟶ MEM2
0935	7420	LOOP1	CPSU,II	II=0 Boucle
0937	1B7C		BCTR,UN LOOP 1	d'attente d'interruptions
0939	0D08C1	KBCRT	LODA,R1 MEM2	MEM2 ⟶ R1
093C	F928		BDRR,R1 KBNV	(R1-1⟶R1)si R≠ 0 KBNV
093E	050A		LODI,R1 0A	0A ⟶ R1
0940	CD08C1		STRA,R1 MEM2	R1 ⟶ MEM2
0943	0C089F		LODA,R0 MKBST	MKBST ⟶ R0
0946	F420		TMI,R0 K	Test du bit K de MKBST
0948	16		RETC,N1	Retour si K≠ 1
0949	0D08C0	KBVLD	LODA,R1 MEM1	MEM1 ⟶ R1
094C	E508		COMI,R1 8	Test R1 = 8 ?
094E	180F		BCTR,N SCROLL	Branchement à SCROLL(R1=8)
0950	0C089F	STLIN	LODA,R0 MKBST	MKBST ⟶ R0
0953	441F		ANDI,R0 1F	On ne garde que le code de la touche
0955	CD288F		STRA,R0 LINE-1,R1+1	Code touche ⟶ LINE
0958	CD08C0		STRA,R1 MEM1	on mémorise la valeur de R1
095B	3F020E		BSTA,UN CONVRT	Branchement à S.P.CONVRT
095E	17		RETC,UN	Retour
095F	3F02CF	SCROLL	BSTA,UN SCROLL	Branch à S.P. SCROLL
0962	0500		LODI,R1 0	0 ⟶ R1
0964	1B6A		BCTR,UN STLIN	Branch. à début de ligne
0966	CD08C1	KBNV	STRA,R1 MEM2	R1 ⟶ MEM2
0969	17		RETC,UN	Retour

MODE EXECUTION

Après avoir lancé l'exécution du programme à partir de l'adresse 0900, la dernière ligne doit être effacée sur l'écran (écriture de 8 caractères "espace") . Grâce à l'analogie des codes caractères et des codes touches nous pouvons déterminer la correspondance entre touche et caractère :

CLAVIER GAUCHE

0	4	8
1	5	9
2	6	A
3	7	b

CLAVIER DROIT

G	P	+
L	r	-
I	=	:
n		X

ESPACE

CLAVIER CENTRAL

F	»
E	o/o
d	●
c	◎

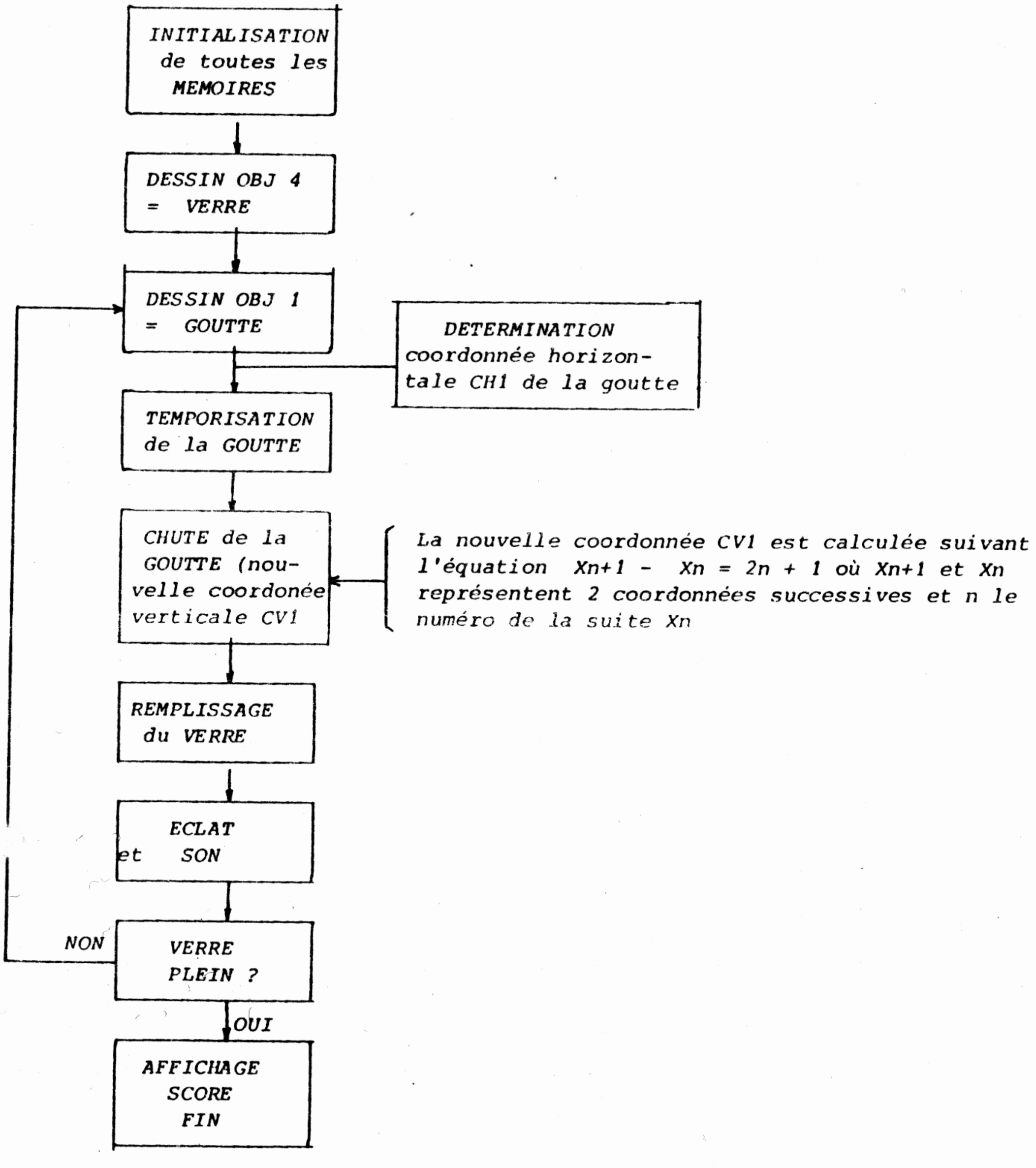
Ces tableaux nous permettent d'écrire et d'afficher des lignes de 8 caractères . Lorsque la ligne est complète, le caractère suivant s'affiche sur la gauche en décalant vers le haut les lignes précédentes (SCROLL) .
La valeur d'attente (H "0A") pour la prise en compte d'une touche a été déterminée arbitrairement . Le lecteur pourra la changer aux adresses 0931 et 093F afin d'ajuster le temps de réponse de la touche (pas trop long) et le temps de prise en compte des rebonds (pas trop court).

V - *Jeu de la Goutte d'eau*

Ce jeu consiste à recueillir dans un verre des gouttes d'eau tombant du haut de l'écran, la position horizontale de la goutte étant indéterminée . Après un remplissage du verre par 10 gouttes, le jeu s'arrête et le score s'affiche . Si aucune goutte n'est tombée à côté, le joueur marque 1000 points .
On peut augmenter l'intêrét du jeu en proposant 2 vitesses d'apparition des gouttes
Le programme décrit a une longueur légèrement supérieure à 1/2 K octets (> 512) . Il comporte un programme principal (de gestion du jeu et de la chute de la goutte d'eau), un sous-programme d'éclat de la goutte, un sous-programme de déplacement du verre, un sous-programme de son, et un sous-programme de comptage et d'affichage du score . Le traitement des interruptions de retour trame ne portera que sur un programme de fin de jeu, par exemple, le déplacement du verre .

I. Organigrammes Généraux

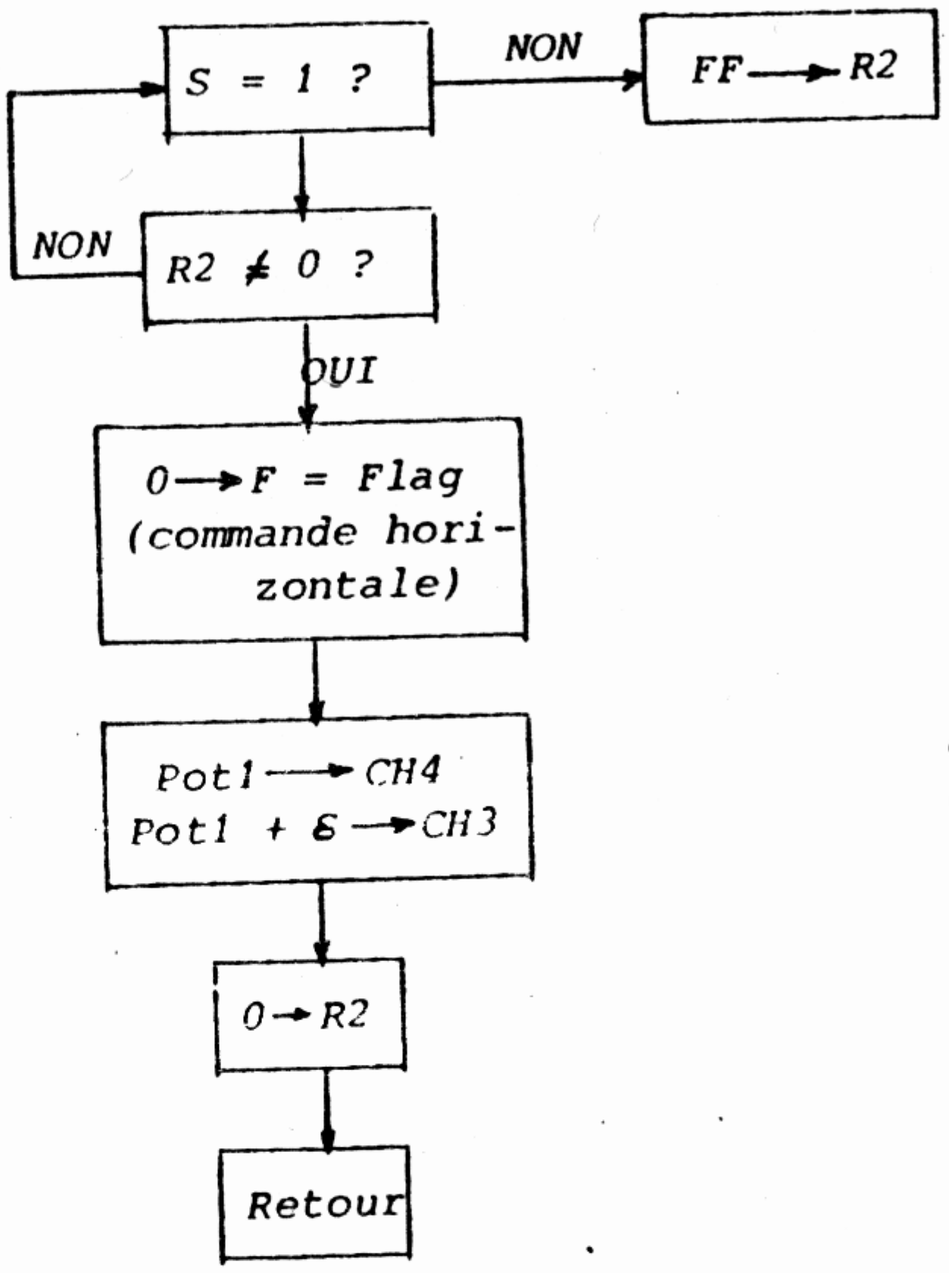
1) Programme Principal



Le programme de temporisation de la goutte est une boucle dont la longueur peut être choisie . Cela permettra d'avoir 2 vitesses d'apparition des gouttes . Ce programme doit inclure le programme de déplacement du verre car on n'utilise pas le traitement des interruptions en cours de partie .

2) Déplacement du verre

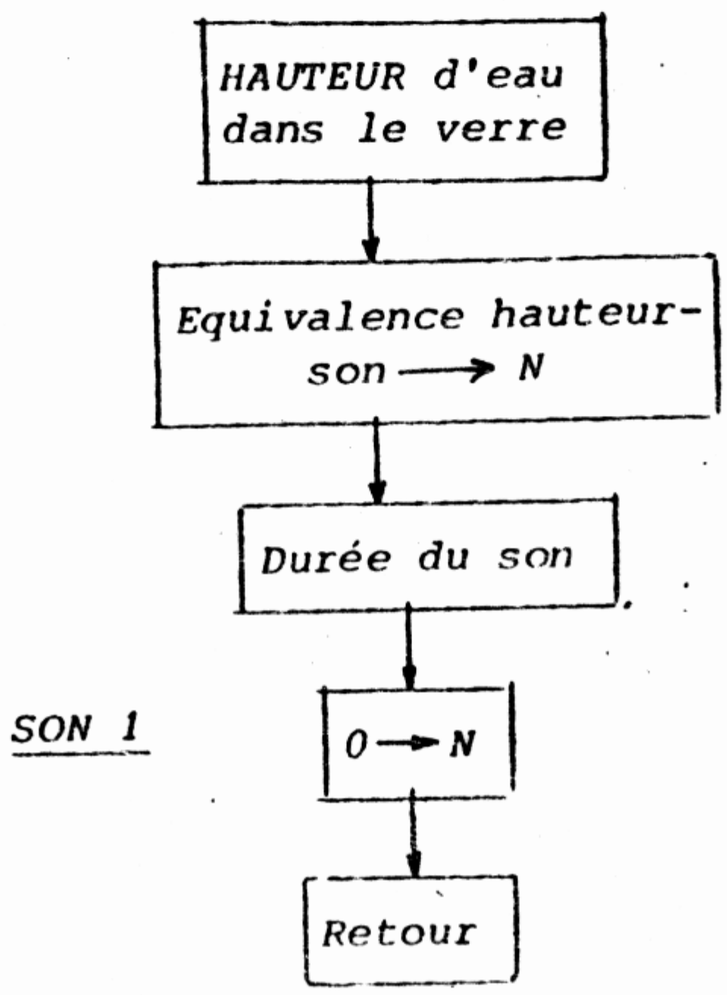
La valeur du potentiomètre de commande doit être stockée dans la case mémoire de coordonnée horizontale du verre CH4 . L'opération se déroule une seule fois pendant le retour trame, c'est-à-dire lorsque le bit SENSE S ± 1 . Le déplacement du verre fait intervenir la déplacement de l'eau contenue dans le verre . La coordonnée horizontale de l'eau (OBJ3) doit donc suivre celle du verre .



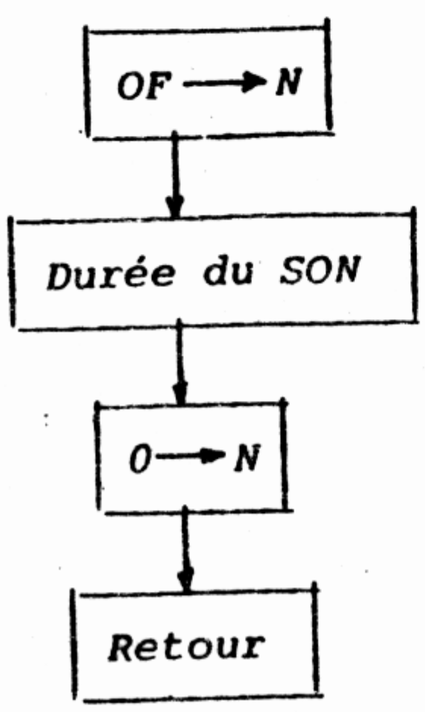
CH4 = Coordonnée Horizontale du Verre
CH3 = Coordonnée Horizontale de l'eau
δ = Différence d'alignement de CH3 avec CH4

Programme de SON

. Donnons 2 programmes de son, 1 pour l'éclat de la goutte sur le verre ou à côté, 1 pour le remplissage du verre . La valeur du son N doit être écrite à l'adresse 1FC7 (voir chapitre IV, para. II,4) . Pour générer le son on doit accéder à l'adresse 1E80, le temps d'accés correspond au temps de génération du son . Pour le remplissage du verre, le son devient de plus en plus aigu .



SON 1

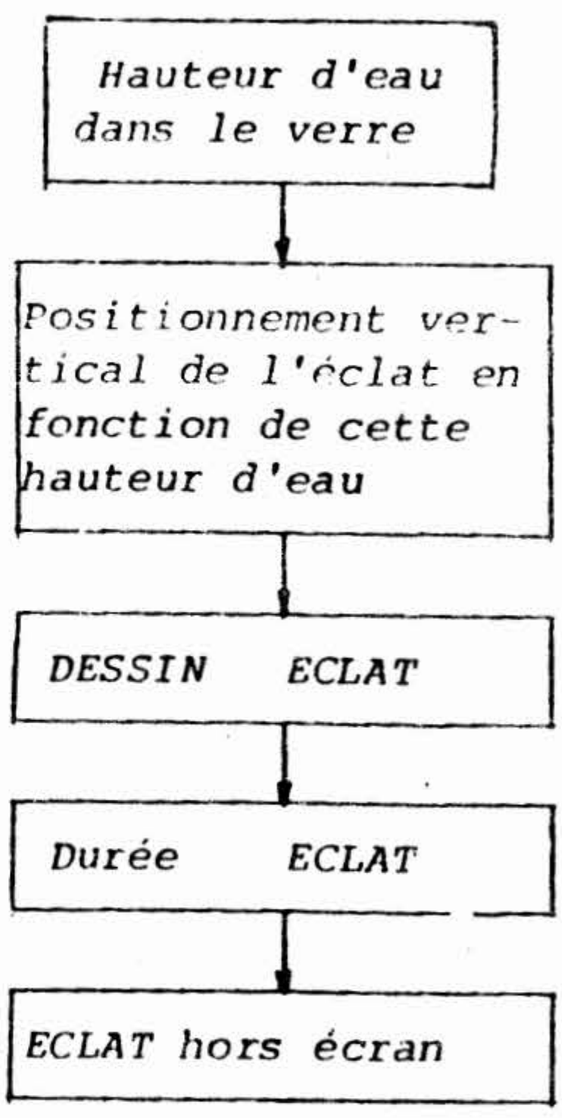


SON 2

4) Programme d'ECLAT

Le dessin de l'ECLAT necessite 2 objets du PVI . Comme 3 objets sont déjà occupés (GOUTTE, VERRE et EAU), il faut recharger l'objet N°1 (à la place de GOUTTE) et prendre OBJ2 pour l'éclat car le PVI ne dispose que de 4 objets . Ceci n'est pas génant car l'éclat et la goutte n'apparaissent pas simultanément .

Il y a 3 possibilités d'éclat :
- à côté du verre
- sur le verre
- dans le verre sur l'objet 3 (EAU) :

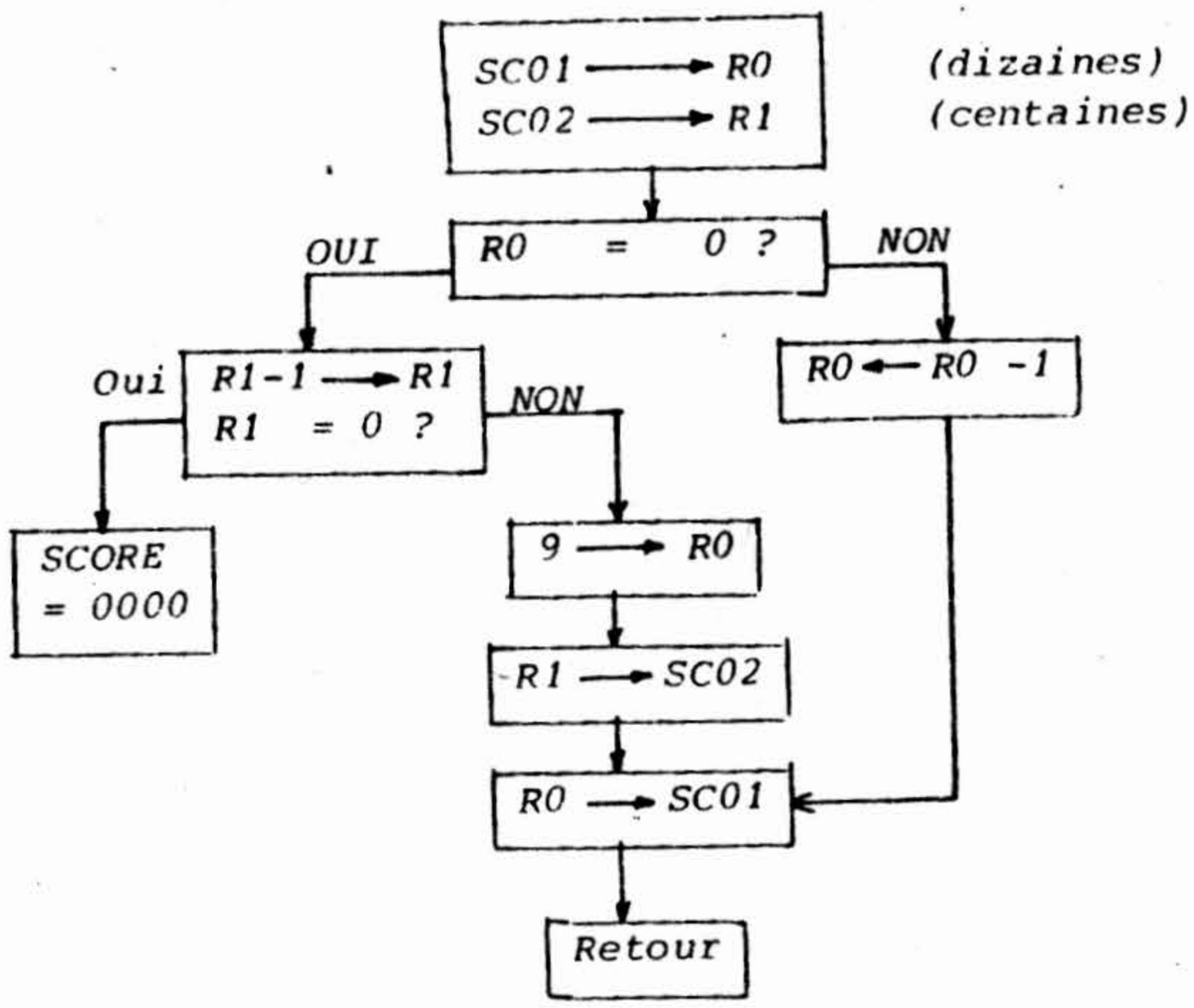


Les 2 premiers cas positionnent la coordonnée verticale de l'éclat et reprennent le dessin et la durée .

5) Programme de COMPTAGE

Il suit le programme d'éclat car à chaque éclat correspond une décrémentation du score . Pour gagner, il faut recueillir 10 gouttes sans en laisser échapper une . Cela correspond à 1000 points . On doit donc initialiser le score à 1100 points . Le jeu s'arrête si on a rempli le verre ou si on a atteint 0000 point

Initialisation de SC01 = 0A
SC02 = 0A dans programme principal

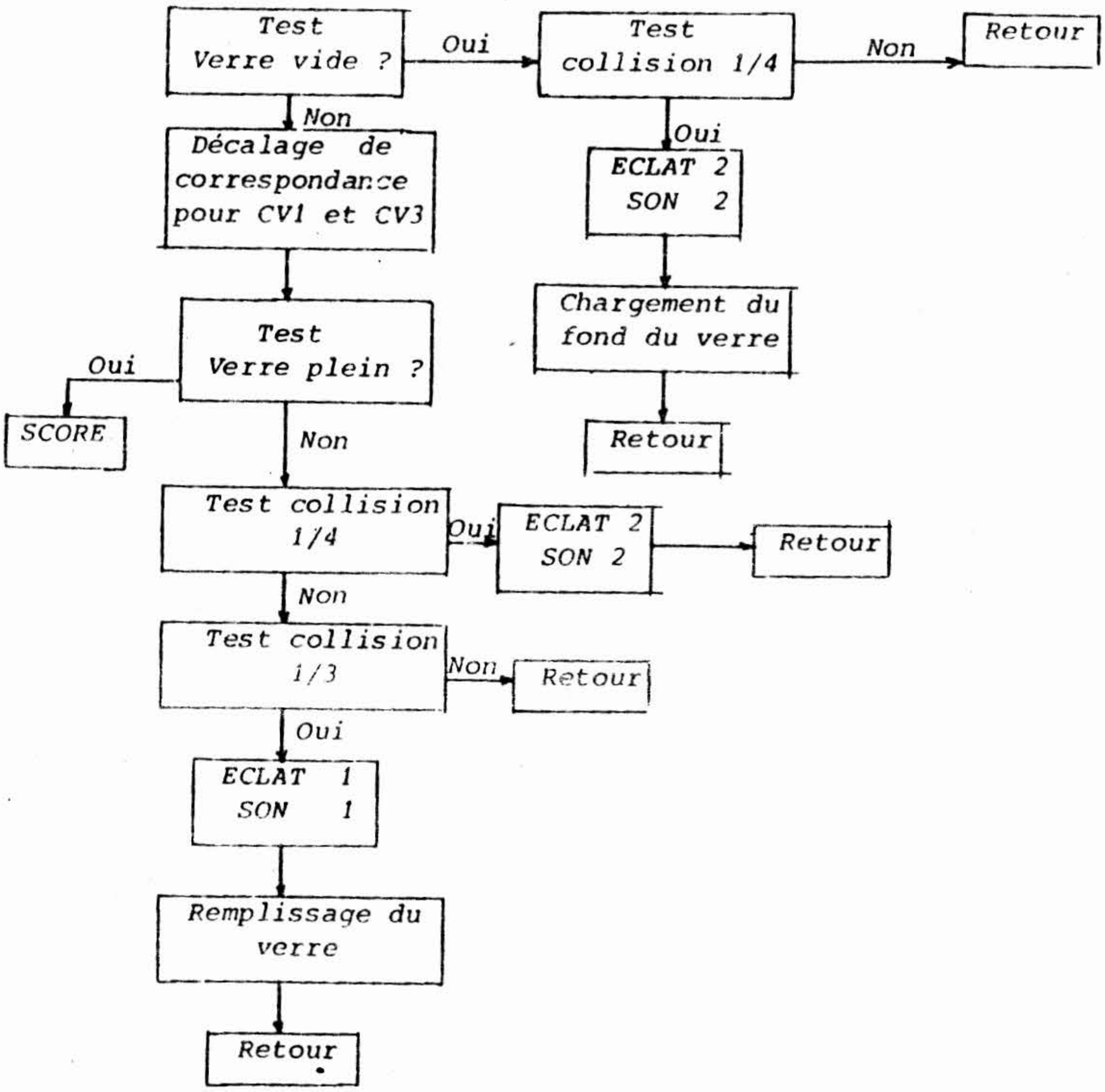


6) <u>*Programme de Gestion de jeu et de remplissage du verre*</u>

Le jeu présente 3 cas distincts :

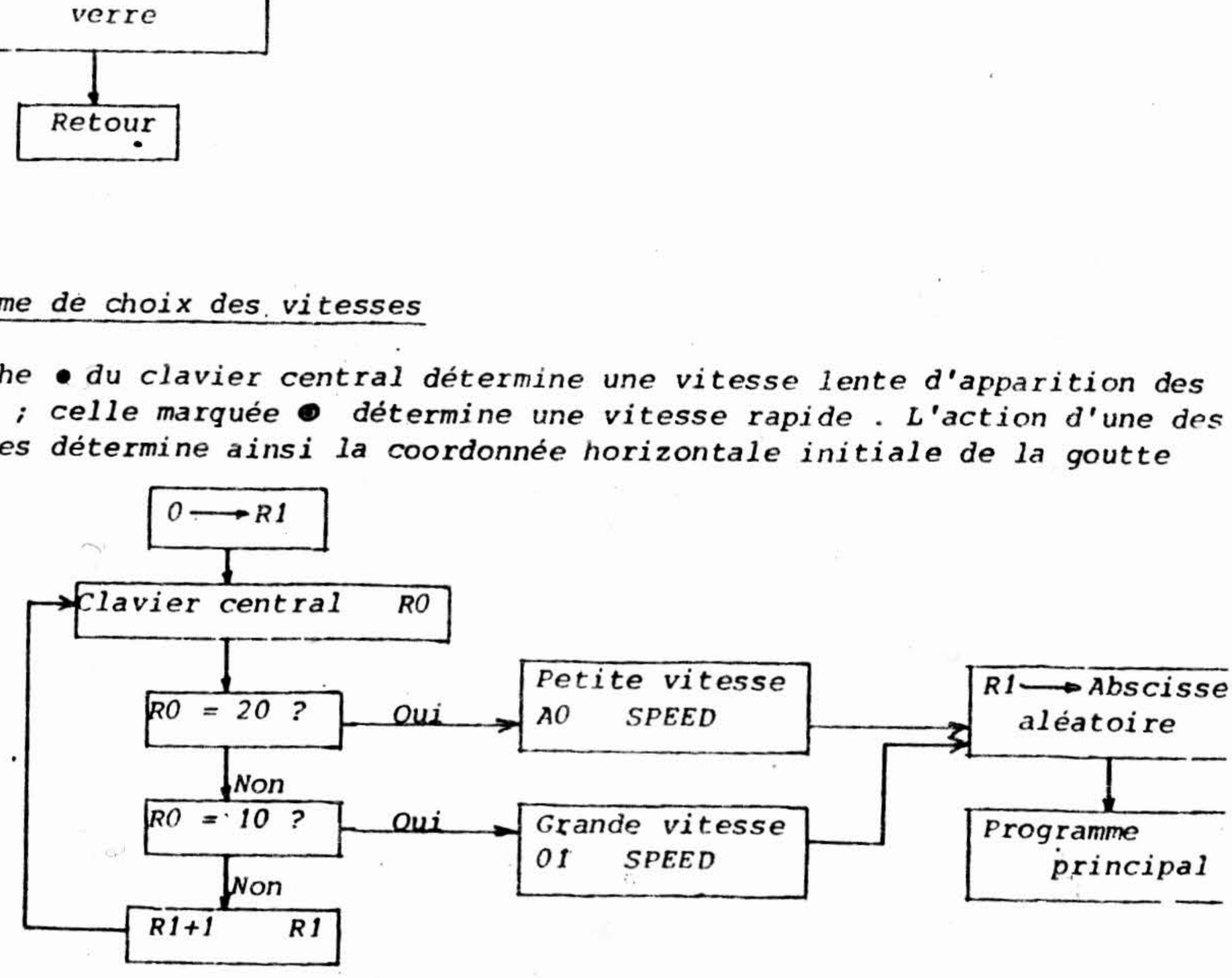
- *Collision entre GOUTTE et VERRE (COLL 1/4)*
- *Collision entre GOUTTE et EAU (COLL 1/3)*
- *Pas de collision .*

En début de jeu, le verre étant vide il n'y a pas de collision entre GOUTTE et EAU . Pour avoir collision entre GOUTTE et VERRE, il faut intersection et donc présence des 2 objets à un isntant donné . Pour le verre, il n'y a pas de problème, la zone étant suffisamment large pour intercepter la goutte . Pour l'objet 3 (EAU), lorsuq'on a le premier niveau de remplissage il faut faire correspondre les coordonnées verticales de l'EAU et de la GOUTTE après la première collision entre GOUTTE et VERRE . Donc, le premier remplissage du verre se fera avec une collision GOUTTE-VERRE en amenant ensuite le couple VERRE-EAU au niveau de correspondance des coordonnées verticales EAU-GOUTTE (Le décalage est presque imperceptible, seul le programmeur pourra l'identifier) .



7) Programme de choix des vitesses

La touche ● du clavier central détermine une vitesse lente d'apparition des gouttes ; celle marquée ● détermine une vitesse rapide . L'action d'une des 2 touches détermine ainsi la coordonnée horizontale initiale de la goutte



Les valeurs A0 et 01 correspondantes aux petites et grandes vitesses déterminent le nombre de boucles dans la temporisation de la goutte .

- Listing du jeu de la Goutte d'Eau

5 octets de mémoire sont utilisés pour garder les informations de score (2) de fin de remplissage (1) et de nombres aléatoires (2) pour la détermination de CH1 . Ces mémoires sont aux adresses :

ADRESSE	ETIQUETTE
08C0	SC01
08C1	SC02
08C2	FINRP
08C3	NBR1
08C4	NBRAL

Ecrivons et commentons les programmes .

1) Programme principal

ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTION	COMMENTAIRES
0900	1F090A	START	BCTA,UN INIT	Branchement à INIT
0903	B480		TPSU,SENSE	Test du bit S = 1 ?
0905	16		RETC,cc = 10	S ≠ 1 retour
0906	3F0B00		BSTA,UN DEPLV	Branchement à S.P. Déplacement verre .
0909	17		RETC	Retour
090A	7620	INIT	PPSU,II	II=1 . Interruptions inhibées
090C	040A		LODI,R0 0A	0A ⟶ R0
090E	CC08C0		STRA,R0 SC01	Initialisation de SC01
0911	CC08C1		STRA,R0 SC02	et de SC02
0914	CC08C3		STRA,R0 NBR1	0A ⟶ NBR1
0917	20		EORZ	0 ⟶ R0
0918	CC08C2		STRA,R0 NBRAL	0 ⟶ NBRAL
091B	7510		CPSL,RS	RS = 0 Banque 0
091D	CC1F0B		STRA,R0 CHD1	Effacement duplicata 1 .
0920	3F016E		BSTA,UN CLROBJ	Effacement Objets .
0923	04BF		LODI,R0 BF	BF ⟶ R0
0925	CC1F4C		STRA,R0 CV4	BF ⟶ CV4
0928	04C3		LODI,R0 C3	
092A	CC1F2C		STRA,R0 CV3	C3 ⟶ CV3
092D	0490		LODI,R0 90	90 ⟶ R0
092F	CC1FC0		STRA,R0 DIMOBJ	Dimensions des objets
0932	050A		LODI,R1 0A	
0934	0D4A40	LOOP Ø	LODA,R0 VERRE R1-	Dessin du verre
0937	CD7F40		STRA,R0 OBJ4 R1	= OBJ4
093A	5978		BRN,R1 LOOP Ø	
093C	040C		LODI,R0 0C	Chargement de 0C dans
093E	CC1FC2		STRA,R0 COLOR	COLOR, couleur des objets
0941	050A	CHUTG	LODI,R1 0A	
0943	0D4A05	LOOP1	LODA,R0 GOUTTE R1-	Dessin de la goutte
0946	CD7F00		STRA,R0 OBJ1 R1	= OBJ1
0949	5978		BRN,R1 LOOP1	

```
094B    070A
094D    CF1F0C
0950    7702
0952    0500
0954    CD08C4      ABSCIG  LODI,R3 0A          Coordonnée verticale
0957    0C08C3              STRA,R3 CV1         CV1 = 0A
095A    0D08C4              PPSL,COM            Mode comparaison logique
095D    81                  LODI,R1 V       <-- V est la valeur de la boucle
095E    E420                STRA,R1 NBRAL       d'attente de sélection de vi
0960    1905                LODA,R0 NBR1        tesse déterminée dans CHOVIT
0962    7509                LODA,R1 NBRAL
0964    51                  ADDZ,R1             NBR1 + NBRAL ---> R0
0965    1B6D                COMI,R0  20
0967    E4A0        CAD1    BCTR,CC = 01CAD1    Branchement si R0 > 20
0969    1A07                CPSL, WC + C        WC = 0   C = 0
096B    7508                RRR,R1              Rotation sans carry
096D    7501                BCTR,UN ABSCIG      Retour à ABSCIG
096F    51          CAD1    COMI,R0 A0
0970    1B62                BCTR,CC = 10        Branchement si R0 < A0
0972    CC1F0A              CPSL, WC            WC = 0
0975    0C08C3              CPSL, C             C = 0
0978    7508                RRR,R1              Rotation sans carry
097A    D0                  BCTR,UN ABSCIG      Retour à ABSCIG
097B    51                  STRA,R0 CH1         R0 ---> CH1
097C    CC08C3              LODA,R0 NBR1
097F    CD08C4              CPSL, WC            WC = 0
0982    1B03                RRL,R0              Rotation à gauche de NBR1
0984    3F0B00      DEPLV1  RRR,R1              Rotation à droite de NBRAL
0987    04A0        TEMPOG  STRA,R0 NBR1        Sauvegarde de NBR1 et NBRAL
0989    0600        LOOP2   STRA,R1 NBRAL       pour 1 autre calcul de CH1 .
098B    B480        TESTS1  BCTR,UN TEMPOG      Aller à TEMPOG
098D    1804                BSTA,UN DEPLV
098F    06FF                LODI,R0 A0      <-- A0 représente la valeur de
0991    1B78                LODI,R2 00          la vitesse qui peut être
0993    5802        UNPAS1  TPSU, SENSE         changée dans CHOVIT
0995    1B74                BCTR,CC=00 UNPAS1   Branchement si S = 1
0997    F870        DECRRØ  LODI,R2 FF          FF ---> R2 (R2 ≠ 0)
                            BCTR,UN TESTS1      Retour à TESTS
0999    FB69                BRN,R0 DECRRØ       Branchement à DECRRØ si R0≠0
099B    040A                BCTR,UN TESTS1
099D    B480        TESTS2  BDRR,R0 LOOP2       Branch. à LOOP2 jusqu'à ce q
099F    1804                                    R0 = 0
09A1    05FF                BDRR,R3 DEPLV1      Branch. à DEPLV
09A3    1B78                LODI,R0 0A      ]
09A5    5902        UNPAS2  TPSU, SENSE     |
09A7    1B74                BCTR,CC=0 UNPAS2|
09A9    83          EQUAT   LODI,R1 FF      |
09AA    83                  BCTR,UN TESTS2  |   Chute de la goutte . La
09AB    8401                BRN,R1 EQUAT    |   coordonnée verticale est
09AD    CC1F0C              BCTR,UN TESTS2  |   déterminée une fois à
09B0    8701                ADDZ,R3         |   chaque retour trame .
09B2    F70F                ADDZ,R3         |
09B4    1814                ADDI,R0 01      |
09B6    3F0B00              STRA,R0 CV1     |
                            ADDI,R3 01      |
                            TMI,R3 0F       ]
                            BCTR,CC=00 ECLAT30
                            BSTA,UN DEPLV
```

```
09B9    3F0C00              BSTA,UN REMPV     Branch. à la logique de remplis-
                                              sage
09BC    F6FF                TMI,R2 FF         Test R2 = FF ? R2 = FF après
09BE    1807                BCTR,CC=0 REP     chaque éclat . Aller à REP
09C0    0500                LODI,R1 00        Pas d'éclat 0⟶R1
09C2    0C1F0C              LODA,R0 CV1       CV1 ⟶ R0
09C5    1B56                BCTR,UN TEST2     Branch. à la boucle de chute
09C7    1F0941      REP     BCTA,UN CHUTG     Branch . en début de chute
09CA    3F0B5D      ECLAT 30 BSTA,UN ECLAT3   Branch. à ECLAT3
09CD    3F0A60              BSTA,UN SON2      Branch. à SON2
09D0    1B75                BCTR,UN REP       Retour à REP
09D2    20          SCORNUL EORZ              0⟶R0
09D3    CC1FC8              STRA,R0 N1N2      N1N2 = 00
09D6    CC1FC9              STRA,R0 N3N4      N3N4 = 00
09D9    1B18                BCTR,UN AFFSCO2
09DB    0C08C1      SCORE   LODA,R0 SC02
09DE    F40A                TMI,R0 0A         Test R0 = 0A ?
09E0    181A                BCTR,CC=0 TILT    Si R0= 0A aller à TILT
09E2    CC1FC8              STRA,R0 N1N2      R0 ⟶ N1N2
09E5    0C08C0              LODA,R0 SC01
09E8    7708                PPSL,WC           WC = 1 avec carry
09EA    7501                CPSL, C           C = 0
09EC    D0                  RRL,R0
09ED    D0                  RRL,R0            Décalage à gauche
09EE    D0                  RRL,R0             de 4 bits
09EF    D0                  RRL,R0             (dizaines)
09F0    CC1FC9      AFFSC01 STRA,R0 N3N4      R0⟶N3N4
09F3    0402        AFFS02  LODI,R0 02
09F5    CC1FC3              STRA,R0 FORPOS    Format et position du score
09F8    7420        LOOP3   CPSU,II           II = 0 Boucle d'attente
09FA    1B7C                BCTR,UN LOOP3       d'interruptions
09FC    0410        TILT    LODI,R0 10
09FE    CC1FC8              STRA,R0 N1N2      10⟶ N1N2
0A01    20                  EORZ              0⟶R0
0A02    1B6C                BCTR,UN AFFSCO 1
```

2) *Déplacement du verre et de son contenu : DEPLV*

```
ADRESSE  VALEUR HEXA  ETIQUETTE   INSTRUCTION          COMMENTAIRES

0B00     7710         DEPLV       PPSL,RS              RS=1 Banque 1
0B02     B480         RETRA       TPSU,SENSE           Test S = 1 ?
0B04     1804                     BCTR,CC=0 UNPAS 3
0B06     06FF                     LODI,R2 FF      ←┐   La fonction de R2 assure un
0B08     1B78                     BCTR,UN RETRA    │   seul passage pendant le
0B0A     5A02        UNPAS3       BRN,R2 CCX       │   retour trame
0B0C     1B74                     BCTR,UN RETRA    ┘
0B0E     7440        CCX          CPSU,F               F=0 commande horizontale
0B10     0D1FCC                   LODA,R1 POT1
0B13     CD1F4A                   STRA,R1 CH4          POT 1 ⟶ CH4
0B16     8508                     ADDI,R1 08
0B18     CD1F2A                   STRA,R1 CH3          POT 1 + 8 ⟶ CH3
0B1B     0500                     LODI,R1 00
0B1D     CD1F4B                   STRA,R1 CHD4         CHD4 = 0
0B20     05F0                     LODI,R1 F0
0B22     CD1F4D                   STRA,R1 CVD4         Pas de duplicata
```

```
OB25    CD1F2D      STRA,R1 CVD3      pas de duplicata
OB28    0600        LODI,R2 00
OB2A    7510        CPSL,RS           RS = 0 Banque 0
OB2C    17          RETC,UN
```

3) Programme de SON : SON 1 et SON 2

```
ADRESSE  VALEUR HEXA  ETIQUETTE  INSTRUCTION       COMMENTAIRES
0A50     050A         SON1       LODI,R1 0A
0A52     0D5F20       BOUCL1     LODA,R0 OBJ3,R1-  Recherche de la
0A55     F43F                    TMI,R0            hauteur d'eau dans
0A57     1879                    BCTR,CC=0,BOUCL1  le verre, puis décalage
0A59     8505                    ADDI,R1 05        de S = 5 pour le son
0A5B     CD1FC7                  STRA,R1 SON
0A5E     1B03                    BCTR UN DURSON1
0A60     CF1FC7       SON2       STRA,R3 SON
0A63     06FF         DURSON1    LODI,R2 FF        ]
0A65     0504         DURSON2    LODI,R1 04        |
0A67     CD1E8C                  STRA,R1 SONENA    | Durée du Son
0A6A     FA79                    BDRR,R2 DURSON2   ]
0A6C     20                      EORZ
0A6D     CC1FC7                  STRA,R0 SON
0A70     17                      RETC,UN
```

4) Programme d'ECLAT : ECLAT 1, ECLAT 2, ECLAT 3 suivi du comptage .

```
0B38    050A        ECLAT1     LODI,R1 0A
0B3A    0D5F20      BOUCL2     LODA,R0 OBJ3,R1-
0B3D    F43F                   TMI,R0              Recherche de la
0B3F    1879                   BCTR,CC=00,BOUCL2   hauteur d'eau dans
0B41    0C1F0C                 LODA,R0 CV1         le verre et recalage
0B44    A411                   SUBI,R0 11          de la position
0B46    81                     ADDZ,R1             verticale de l'éclat
0B47    81                     ADDZ,R1
0B48    CC1F0C                 STRA,R0 CV1
0B4B    CC1F1C                 STRA,R0 CV2
0B4E    1B13                   BCTR,UN DESECL
0B50    0C1F0C      ECLAT2     LODA,R0 CV1
0B53    A416                   SUBI,R0 16          Position verticale sur
0B55    CC1F0C                 STRA,R0 CV1         le bord du verre
0B58    CC1F1C                 STRA,R0 CV2
0B5B    1B06                   BCTR,UN DESECL
0B5D    0C1F0C      ECLAT 3    LODA,R0 CV1
0B60    CC1F1C                 STRA,R0 CV2
0B63    050A        DESECL     LODI,R1 0A
0B65    0D4A10      LOOP4      LODA,R0 DEMEC,R1-   ]
0B68    Cp7F10                 STRA,R0,OBJ2,R1     | Dessin de l'ECLAT
0B6B    .5978                  BRN,R1 LOOP4        ]
```

```
0B6D    050A                    LODI,R1 0A
0B6F    0D4A20      LOOP5       LODA,R0 DEMEC2,R1-
0B72    CD7F00                  STRA,R0 OBJ1,R1
0B75    5978                    BRN,R1 LOOP5
0B77    0C1F0A                  LODA,R0 CH1
0B7A    A404                    SUBI,R0 04
0B7C    CC1F1A                  STRA,R0 CH2
0B7F    8408                    ADDI,R0 D8
0B81    CC1F0A                  STRA,R0 CH1
0B84    050A                    LODI,R1 0A
0B86    B480        TESTS3      TPSU,SENSE
0B88    1804                    BCTR,CC=0 UNPAS
0B8A    06FF                    LODI,R2 FF
0B8C    1B78                    BCTR,UN TEST3
0B8E    5A02        UNPAS4
0B90    1B74                    BCTR,UN TESTS3
0B92    F902                    BDRR,R1 POINT1
0B94    1B04                    BCTR,UN POINT2
0B96    0600        POINT1      LODI,R2 D0
0B98    1B6C                    BCTR,UN TESTS3
0B9A    20          POINT2      EORZ
0B9B    CC1F0A                  STRA,RO CH1
0B9E    CC1F1A                  STRA,R0 CH2
0BA1    0C08C0                  LODA,R0 SC01
0BA4    0D08C1                  LODA,R1 SC02
0BA7    580E                    BRNR,R0 POINT4
0BA9    F903                    BDRR,R1 POINT3
0BAB    1F09D2                  BCTA,UN SCORNUL
0BAE    0409        POINT3.     LODI,R0 09
0BB0    CD08C1                  STRA,R1 SC02
0BB3    CC08C0      POINT5      STRA,R0 SC01
0BB6    17                      RETC,UN
0BB7    A401        POINT4      SUBI,R0 01
0BB9    1B78                    BCTR,UN POINT5
```

Dessin de l'ECLAT

Positionnement horizontal de l'ECLAT

Durée de l'ECLAT

Durée de l'ECLAT

Eclat hors écran

SC01 ⟶ R0

SC02 ⟶ R1

Comptage des points

5) <u>Programme de remplissage du verre REMPV</u>

ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTION	COMMENTAIRES
0C00	0D1F29	REMPV	LODA,R1 OBJ39	
0C03	F53F		TMI,R1	Test R1 = 3F ?
0C05	1819		BCTR,CC=00 VRNVD	Branch à Verre Non Vide
0C07	0D1FCB		LODA,R1 CI0	Collision Inter Objet ⟶ R1
0C0A	F508		TMI,R1	Test collision 1/4
0C0C	980F		BCFR,CC=00 RETOUR	Branch. si pas de collision 1/4
0C0E	3F0B50		BSTA,UN ECLAT2	Si collision 1/4
0C11	3F0A60		BSTA,UN SON2	
0C14	063F		LODI,R2 3F	
0C16	CE1F29		STRA,R2 OBJ39	Remplissage d'un niveau
0C19	06FF		LODI,R2 FF	détermine la fin de la goutte

```
0C1B    1B02                BCTR,UN RET
0C1D    0600        RETOUR  LODI,R2 00      Détermine la non fin de la
                                               goutte
0C1F    17          RET     RETC,UN
0C20    04C1        VRNVD   LODI,R0 C1   ]
0C22    CC1F4C              STRA,R0 CV4   ]  Ajustage des coordonnées
0C25    04C5                LODI,R0 C5    ]  verticales du verre
0C27    CC1F2C              STRA,R0 CV3   ]  et de son contenu
0C2A    0D1F20              LODA,R1 OBJ30
0C2D    F53F                TMI,R1          Test R1 = 3F ?
0C2F    9803                BCFR,CC=00 VRNPL  Branch. à VeRre Non PLein
0C31    1F09DB              BCTA,UN SCORE
0C34    0D1FCB      VRNPL   LODA,R1 CIO
0C37    F508                TMI,R1          Test Collision 1/4
0C39    9809                BCFR,CC=00 TSTCOL Branch. si pas de coll. 1/4
0C3B    3F0B50              BSTA,UN ECLAT2
0C3E    3F0A60              BSTA,UN SON2
0C41    06FF                LODI,R2 FF
0C43    17                  RETC,UN
0C44    F510        TSTCOL  TMI,R1          Test Collision 1/3
0C46    9855                BCFR,CC=00 RETOUR Branch. si pas de Coll. 1/3
0C48    3F0B38              BSTA,UN ECLAT1
0C4B    3F0A50              BSTA,UN SON1
0C4E    06FF                LODI,R2 FF
0C50    0C08C2              LODA,R0 FINRP
0C53    5815                BRN,R0 VPLEIN
0C55    050A                LODI,R1 0A
0C57    0D5F20      POINT6  LODA,R0 OBJ30,R1-
0C5A    F43F                TMI,R0          Test R0 = 3F ?
0C5C    1879                BCTR,CC=00,POINT6 Branch. si R0 = 3F
0C5E    043F                LODI,R0 3F
0C60    CD5F21              STRA,R0,OBJ31,R1-  Remplissage du Verre
0C63    5908                BRN,R1 RET1     Lorsque R1=0 le verre est
0C65    CC08C2              STRA,R0 FINRP   presque plein, il faut alors
0C68    1B03                BCTR,UN RET1    stocker 3F dans FINRP et
0C6A    CC1F20      VPLEIN  STRA,R0 OBJ30   attendre la dernière goutte
0C6D    17          RET1    RETC,UN         pour remplir le verre .
```

6) Programme de choix de Vitesse

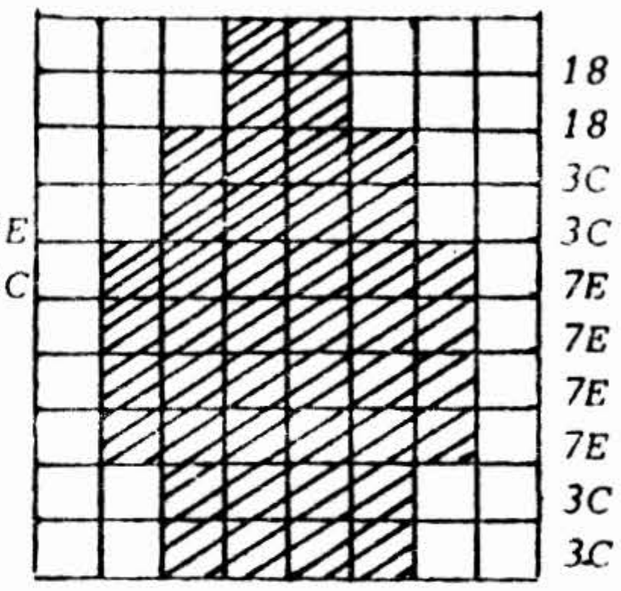
ADRESSE	VALEUR HEXA	ETIQUETTE	INSTRUCTION	COMMENTAIRES.
0CC0	7620	CHOVIT	PPSU,II	Interruptions Inhibées
0CC2	0500		LODI,R1 00	Initialisation de R1
0CC4	0C1E8B	RCHCLV	LODA,R0 CLACNT	R0 charge par CLAvierCeNTral
0CC7	44F0		ANDI,R0 F0	Bits significatifs seuls
0CC9	E420		COMI,R0 20	R0 = 20 ? Touche ●
0CCB	1813		BCTR,CC=00 VITLNT	Branch. si R0 = 20
0CCD	E410		COMI,R0 10	R0 = 10 ? Touche ◉
0CCF	1804		BCTR,CC=00 VITRPD	Branch. si R0 = 10

```
0CD1    8501            ADDI,R1 01      R1 + 1 ⟶ R1
0CD3    1B6F            BCTR,UN RCHCLV  Retour à balayage clavier
0CD5    0405    VITRPD  LODI,R0 05      Vitesse rapide
0CD7    CC0988  POINT7  STRA,R0 DATA    R0 donnée de vitesse
0CDA    CD0953          STRA,R1 V       R1 valeur aléatoire ⟶ V
0CDD    1F0900          BCTA,UN START   Programme lancé
0CE0    04A0    VITLNT  LODI,R0 A0      A0 ⟶ R0
0CE2    1B73            BCTR,UN POINT7  Aller à POINT 7
```

7) Ecriture des données . Dessin du verre, de la goutte et de l'éclat

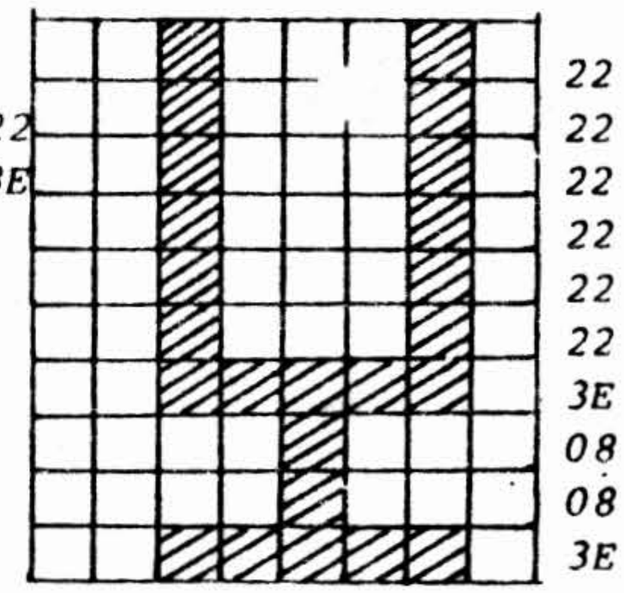
a) Dessin de la goutte

```
ADR        DONNEES
0A05  18 18 3C 3C 7E
0A0A  7E 7E 7E 3C 3C
```



b) Dessin du verre

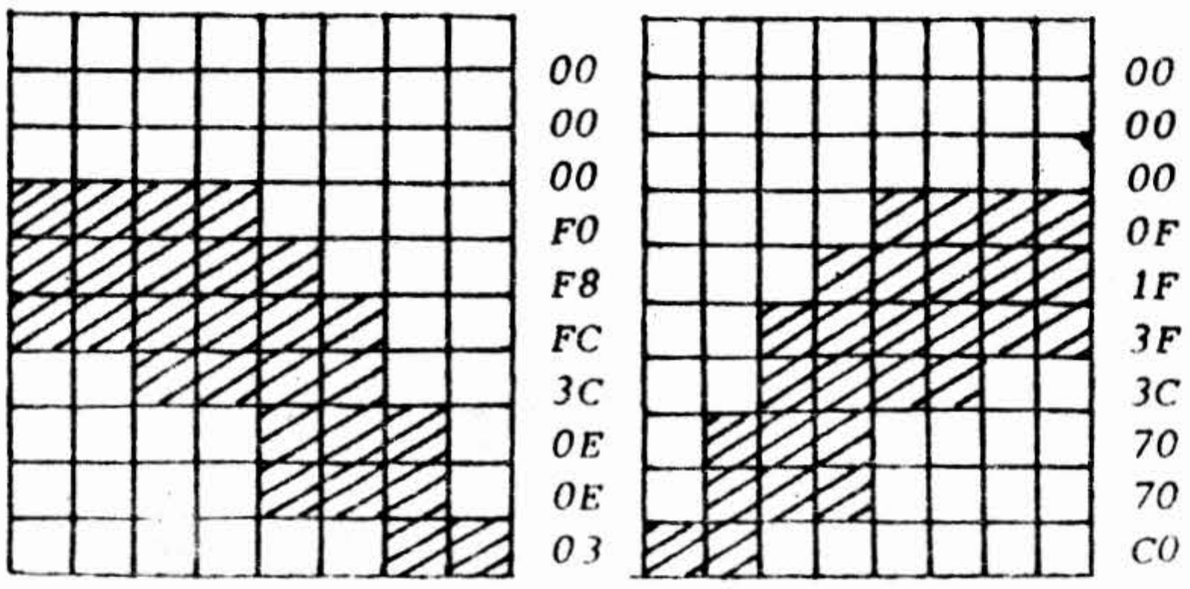
```
0A40  22 22 22 22 22
0A45  22 3E 08 08 3E
```



c) Dessin de la goutte

```
0A10  00 00 00 F0 F8
0A15  FC 3C 0E 0E 03

0A20  00 00 00 0F 1F
0A25  3F 3C 70 70 C0
```



8 - EXECUTION

Après avoir entré le programme et les données, puis vérifié son écriture, passons au mode EXECUTION . Il faut afficher PC = 0CC0 . En lançant le PC par la touche +, on entre dans la boucle du programme de choix de vitesse . PC = 0CC0 s'affiche en haut de l'écran . Pour commencer le jeu, il faut choisir la vitesse lente (touche ●) ou la vitesse rapide (touche ◉) . Le jeu étant lancé, pour rattraper la goutte, on déplace en X (direction gauche-droite) le manche gauche qui fait déplacer le verre . Le jeu se termine par l'affichage du score . Pour recommencer une partie, il faut faire une RAZ (touche ◀) et relancer le PC .

VI - SOUS-PROGRAMMES D'ADDITION, SOUSTRACTION, MULTIPLICATION et CONVERSION

Nous proposons pour terminer quelques programmes utiles qui vous aideront dans vos programmes de jeu .

1 - ADDITION/SOUSTRACTION Binaire de nombres signés de longueur 2 octets

Entrées = Opérande 1 aux adresses 0D2D et 0D2E
Opérande 2 aux adresses 0D2F et 0D30
Détermination d'Addition ou de Soustraction par le bit COM du PSL
COM = 0 Addition
COM = 1 Soustraction

Sorties = Résultat aux adresses 0D31 et 0D32

Remarque : Les adresses 0D2D,0D2F et 0D31 comportent les octets de poids fort d'opérande 1, opérande 2 et du résultat .

Listing :

```
0D00        7709        ADSB    PPSL,WC + C
0D02        0502                LODI,R1 2
0D04        B502                TPSL, COM
0D06        180F                BCTR,CC=00 LPSB   Branch. si soustraction .
0D08        7501                CPSL, C           Addition, C = 0
0D0A        0D4D2D      LPAD    LODA,R0 OPR1,R1-
0D0D        8D6D2F              ADDA,R0 OPR2,R1
0D10        CD6D31              STRA,R0 RSLT,R1
0D13        5975                BRNR,R1 LPAD
0D15        1B0B                BCTR,UN TEST
0D17        0D4D2D      LPSB    LODA,R0 OPR1,R1,-
0D1A        AD6D2F              SUBA,R0 OPR2,R1
0D1D        CD6D31              STRA,R0 RSLT,R1
0D20        5975                BRNR,R1 LPSB
0D22        9808        TEST    BCTR,Z RTRN
0D24        0C0D32              LODA,R0 RSLT+1
0D27        14                  RETC,
0D28        7580                CPSL, CC1
0D2A        7740                PPSL, CC0
0D2C        17          RTRN    RETC, UN
```

2 - MULTIPLICATION Binaire de 2 Nombres non-signés de longueur 1 Octet

Entrées = OPR1 contient le multiplieur d'adresse 08C0 .
OPR2 contient le multiplicande d'adresse 08C1 .

Sorties = RSLT octet de poids fort du résultat - Adresse 08C2
RSLT + 1 octet de poids faible du résultat - Adresse 08C3

```
0E00    0D08C0    MULT    LODA,R1 CPR1
0E03    0E08C1            LODA,R2 CPR2
0E06    7708      MPYU    PPSL, WC
0E08    20                EORZ, R0
0E09    0708              LODI, R3
0E0B    7501      LOOP    CPSL, C
0E0D    F501              TMI, R1
0E0F    9801              BCFR, CC=00 SHFT
0E11    82                ADDZ, R2
0E12    50        SHFT    RRR, R0
0E13    51                RRR, R1
0E14    FB75              BDRR, R3 LOOP
0E16    CC08C2            STRA,R0 RSLT
0E19    CD08C3            STRA,R1 RSCT + 1
0E1C    17                RETC, UN
```

3 - CONVERSION BINAIRE - DECIMAL

Fonction : Conversion d'un nombre binaire de longueur 1 octet en un nombre décimal de 3 digits .

Entrées = BINN contient le nombre binaire de 8 bits . Adresse 08C0

Sorties = Les registres R0, R1 contiennent le résultat en décimal (3 digits) .
R0 est l'octet de poids fort .
Le résultat maximum est 255 (conversion de FF)

```
0F00    770A      CONV    PPSL, WC + COM
0F02    7501              CPSL, C
0F04    0C08C0            LODA, R0 BINN
0F07    C1                STRZ, R1
0F08    450F              ANDI, R1 0F
0F0A    8566              ADDI, R1 66
0F0C    95                DAR, R1
0F0D    44F0              ANDI, R0 F0
0F0F    E410      LOOP    COMI, R0 10
0F11    1A09              BCTR,LT EXIT
0F13    A40F              SUBI,R0 10-1
0F15    857B              ADDI, R1 16+66-1
0F17    95                DAR, R1
0F18    8400              ADDI,R0 0
0F1A    1B73              BCTR, UN LOOP
0F1C    17        EXIT    RETC, UN
```

VI - LEXIQUE

ADRESSAGE ABSOLU

Dans ce mode d'adressage, l'opérande se trouve à l'adresse qui est spécifiée, celle-ci étant codée sur 13 bits, elle ne peut dépasser la page de 8 K octets .

ADRESSAGE IMMEDIAT

Dans ce mode d'adressage, l'opérande est donnée directement, sans référence à une adresse quelconque .

ADRESSAGE IMPLICITE

Mode d'adressage mettant en jeu, sans le désigner explicitement un des registres internes du microprocesseur (en général R0)

ADRESSAGE INDEXE

Mode d'adressage faisant intervenir un registre d'index dont le contenu est utilisé pour calculer une nouvelle adresse .

ADRESSAGE INDIRECT

Adressage d'un emplacement mémoire qui contient l'adresse de la donnée et non la donnée elle-même .

ADRESSAGE RELATIF

Mode d'adressage permettant de spécifier un emplacement mémoire en ajoutant ou en retranchant une certaine valeur à l'adresse courante .

BOUCLE (LOOP)

Répétition d'une séquence donnée d'instructions jusqu'à ce qu'une certaine condition ou valeur soit atteinte .

BRANCHEMENT (BRANCH)

Se rapporte à la possibilité pour un micrprocesseur de modifier la séquence de programme . Une telle modification peut dépendre de la valeur d'une donnée ou de la réalisation ou non d'une condition .

BUS de DONNEES (DATA BUS)

C'est l'ensemble de lignes de données qui permet au microprocesseur de communiquer à l'intérieur et à l'extérieur . Ce Bus est bidirectionnel et peut transférer les données vers le microprocesseur, les mémoires, les claviers, etc..

COMPLEMENT à 2

Le complément à 2 est une fonction binaire qui permet à l'Unité Arithmétique et Logique du microprocesseur (centre de calcul) de réaliser des additions binaires de nombres positifs et négatifs . Il s'obtient en ajoutant 1 au complément .

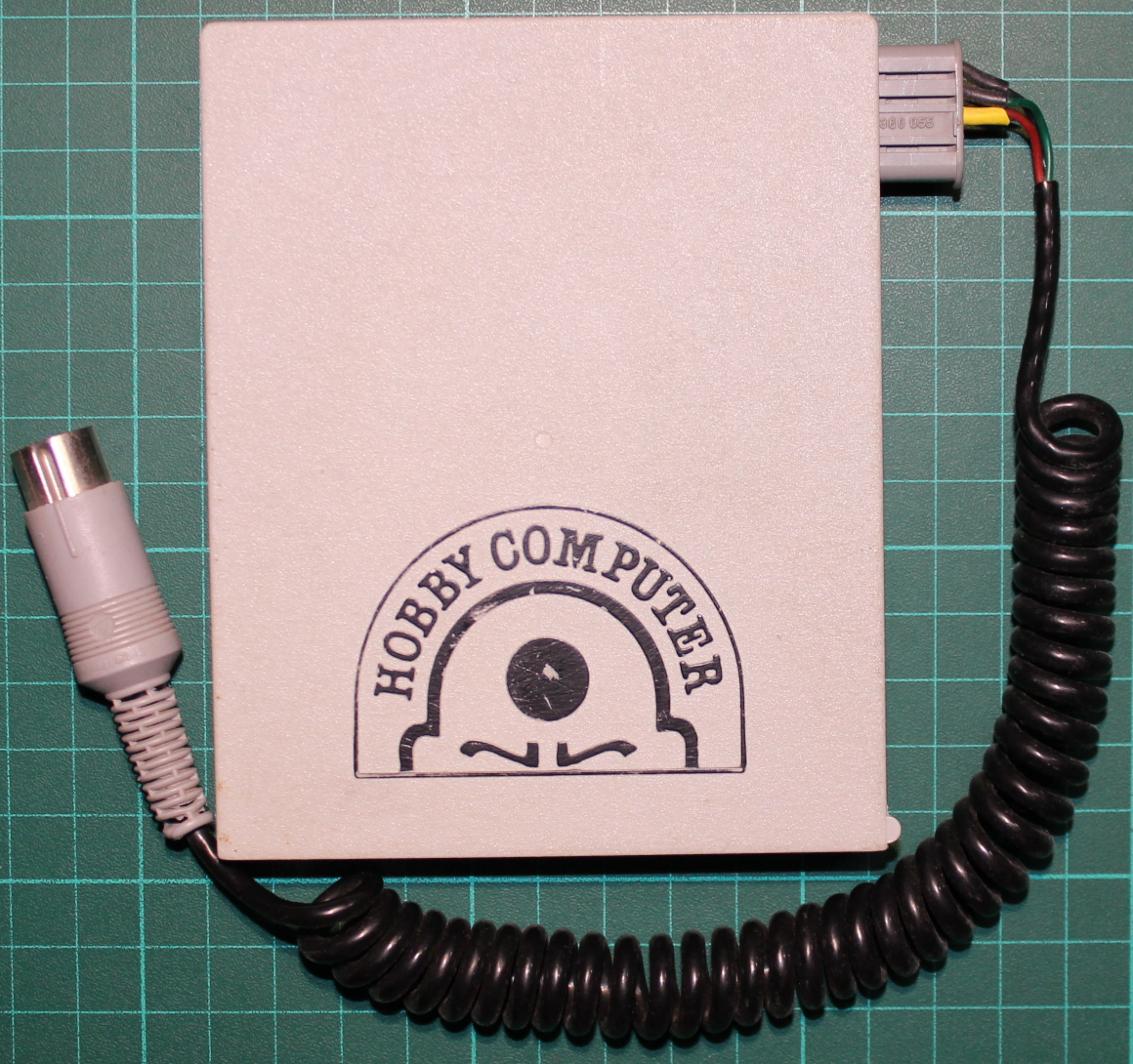
COMPTEUR DE PROGRAMME (PC)

Registre de l'Unité Centrale qui contient en permanence l'adresse de l'instruction suivante à exécuter . Au moment d'une interruption ou d'une instruction de branchement à un sous-programme ; le compteur de programme sauvegarde cette adresse de manière à pouvoir y retourner après traitement de l'interruption .

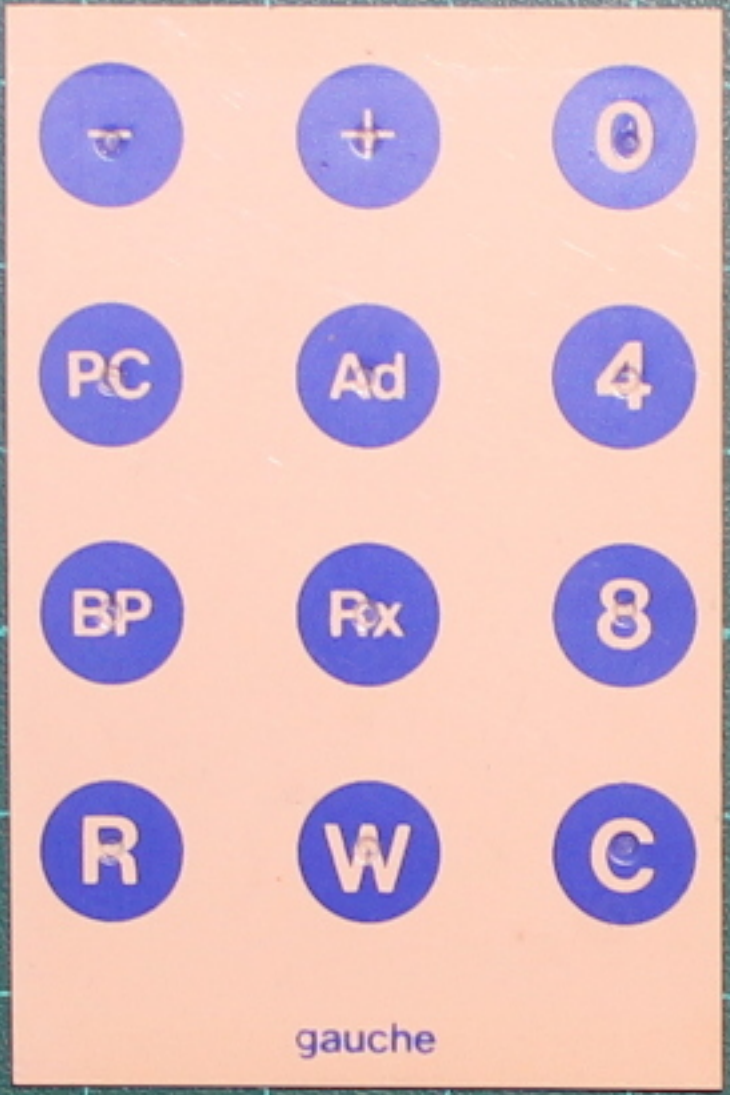
DECREMENT	*Instruction de programmation qui permet de faire décroitre le contenu d'un emplacement mémoire d'une unité .*
HEXADECIMAL	*Nombres à notation positionnelle utilisant la base 16 . Puisqu'il y a 16 digits hexadécimaux (0 à 15) et qu'il n'existe que 10 digits décimaux (0 à 9), on a introduit six digits supplémentaires (10 à 15) représentés par les six premières lettres de l'alphabet .*
HORLOGE (CLOCK)	*Générateur d'impulsions qui commande le séquencement de la commutation des circuits internes au microprocesseur . La fréquence d'horloge du 2650 de l'OC 2000 est de 3,55MHz*
INCREMENT	*Augmentation d'une unité du contenu d'un registre ou d'un compteur . Comme le décrément, ce mot définit une opération de logiciel (souvent associée à la pile d'adresses de retour ou au pointeur de pile) .*
INTERRUPTION	*Suspension de la séquence en cours d'exécution par le microprocesseur, de façon à répondre immédiatement à une demande extérieure .*
INTERRUPTION VECTORISEE	*Système de gestion des interruptions dans lequel chaque demande se traduit par la fourniture immédiate de l'adresse (vecteur) caractérisant le périphérique demandeur .*
JEU D'INSTRUCTIONS	*Liste de toutes les instructions que le microprocesseur sait interpréter et exécuter .*
LANGAGE MACHINE	*Le seul langage compréhansible par le microprocesseur est le langage binaire ou langage machine .*
LOGICIEL (SOFTWARE)	*C'est le manuel d'instructions ou de l'ensemble des programmes que sait exécuter l'ordinateur . L'écriture utilisée est le langage machine .*
MEMOIRE TAMPON ou BLOC-NOTE (SCRATCHPAD)	*Mémoire permettant le stockage temporaire de résultats ou de données intermédiaires .*
MICROPROCESSEUR	*C'est une unité centrale constituée d'un ou de deux circuits de base comprenant en général : une unité arithmétique et logique, un bloc de contrôle, un ensemble de registres . Le microprocesseur ne peut fonctionner que par l'adjonction d'un système mémoire, de dispositifs d'entrée-sortie et d'une horloge de séquencement, plus le jeu d'instructions appropriées .*
MNEMONIQUE	*Expressions étudiées pour aider la mémoire humaine . Le langage microprocesseur étant constitué par une suite de mots binaires qui sont des successions de 0 ou de 1, il est difficile de se rappeler la signification des instructions sous cette forme . Pour assister la mémoire humaine, à chaque code binaire d'instruction correspond un groupe de lettres, symbole mnémonique, qui définit l'instruction . Ce langage mnémonique utilise des termes anglais, largement consacrés par l'usage tels que, par exemple, LODI pour "LOaD Immediate", chargement immédiat .*

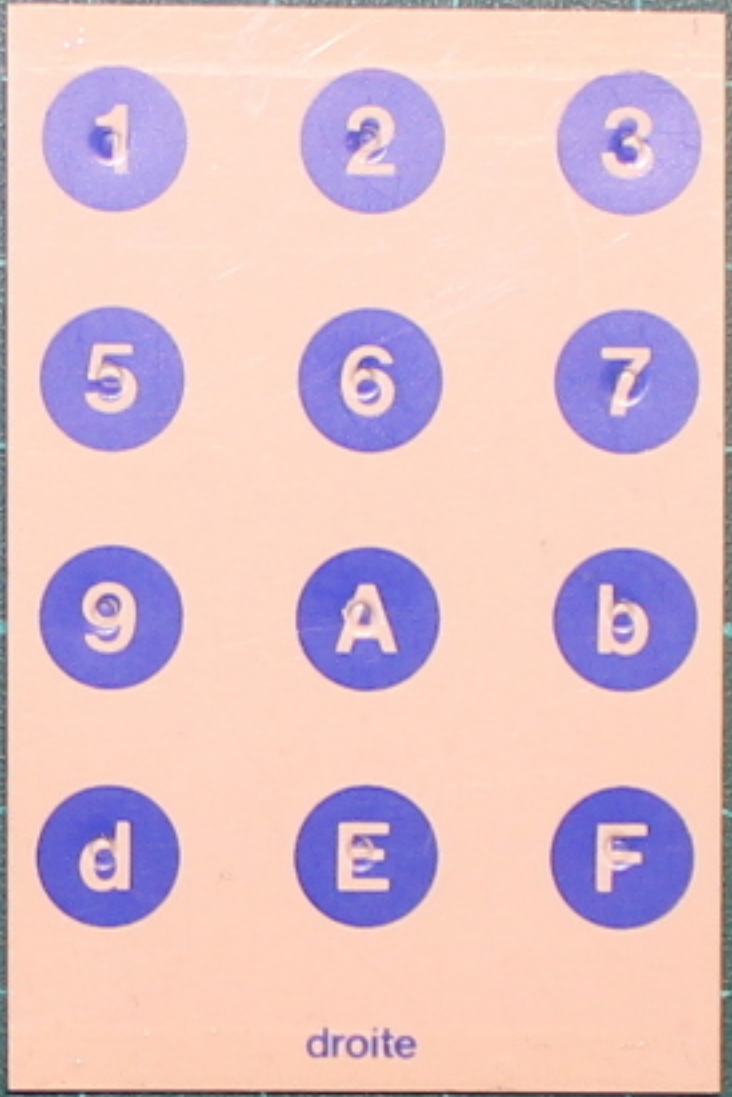
MODES D'ADRESSAGE	Une adresse est un message codé désignant l'emplacement d'une donnée ou d'une partie de programme dans une mémoire . Le code d'adresse lui-même peut être stocké en mémoire, de sorte que l'emplacement mémoire contient l'adresse de la donnée et non la donnée elle-même . Ce mode d'adressage, dit indirect, est classiquement utilisé dans les microprocesseurs .
MOT	Groupe de caractères traités comme un tout . Peut comporter des caractères et symboles numériques et/ou alphabétiques .
MOT D'ETAT	Groupe de nombres binaires qui renseigne l'utilisateur sur l'état dans lequel se trouve le microprocesseur et sur certains résultats d'opérations tels que : indication de dépassement, bit de retenue, nombres traités positifs ou négatifs, bit d'interruption, etc...
OPERANDE	Quantité sur laquelle une opération mathématique doit être effectuée . Désigne aussi un des champs d'instruction dans un adressage . Habituellement, l'expression consiste en un opérateur et une opérande . L'opérateur peut indiquer l'instruction (par ex : addition, ADD) et l'opérande ce qui doit être additionné .
ORGANIGRAMME ou DIAGRAMME DE CHEMINEMENT (FLOW CHART)	Séquence d'opérations décrite à l'aide de symboles, diagrammes ou autres représentations pour indiquer un programme d'exécution . Le diagramme de cheminement permet au concepteur de visualiser les conditions nécessaires à la réalisation de chaque étape du programme .
PILE	Bloc d'emplacements mémoires successifs, ou ensemble de registres d'empilements .
POINT D'ARRET	Etape dans le programme, désignée par un indicateur de point d'arrêt, qui interrompt le déroulement de la séquence pour permettre à l'utilisateur de vérifier le programme .
POINTEUR DE PILE	Compteur associé à une pile de registres, et dont le contenu indique en permanence l'emplacement où la dernière information a été stockée, ou l'emplacement libre suivant. Ce compteur est automatiquement incrémenté ou décrémenté .
PROGRAMME	Procédure utilisée pour résoudre un problème et fréquemment appelée "logiciel"
REGISTRE d'INDEX	Sert généralement à effectuer un certain nombre d'opérations soit en fonction de son contenu qui est testé, soit en utilisant directement ce contenu dans un calcul . Dans les 2 cas, le contenu peut être modifié par programmation

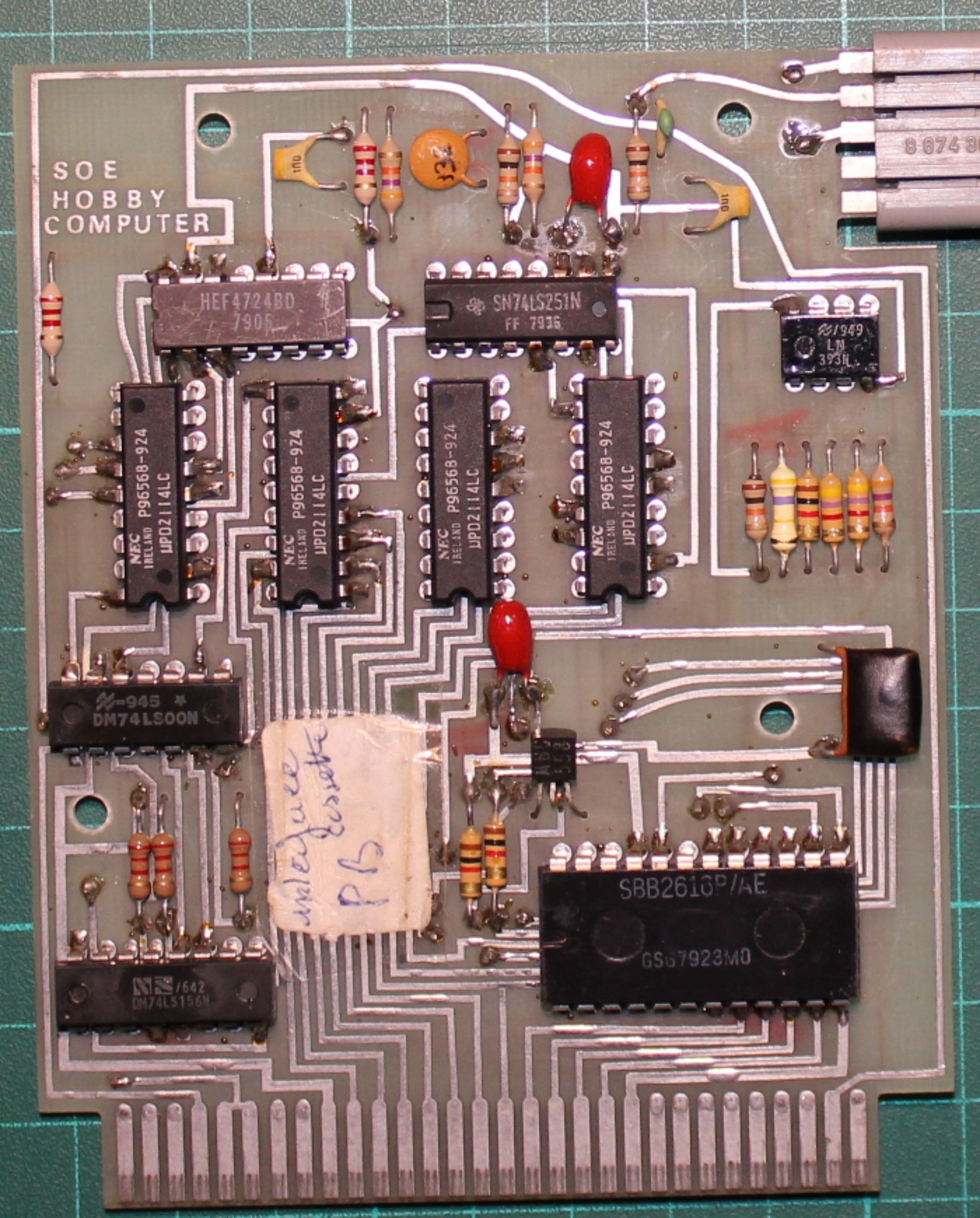
RETOUR TRAME	*En télévision, période pendant laquelle l'image est effacée pour permettre aux faisceaux d'électrons de revenir en début d'image et de décrire à nouveau la trame . Cette période , asservie sur la fréquence du réseau EDF, est imperceptible à l'oeil humain ; c'est pourquoi , on ne "voit" pas d'effacement image sur le téléviseur . Le retour ligne est similaire au retour trame : il permet de revenir à la ligne suivante sans"écrire" sur le tube . Ce sont 2 interruptions vidéo .*
SOUS-PROGRAMME (SUB ROUTINE)	*Partie de programme pouvant être utilisée plusieurs fois dans différents programmes . Fait l'objet d'instructions de saut ou de branchement .*
SYSTEME de BUS	*Réseau de lignes d'échange.d'informations, à l'intérieur du microprocesseur, qui assure la circulation des données . Les chemins d'échange les plus importants sont le chemin de données, le chemin d'adresse, le chemin de contrôle .*

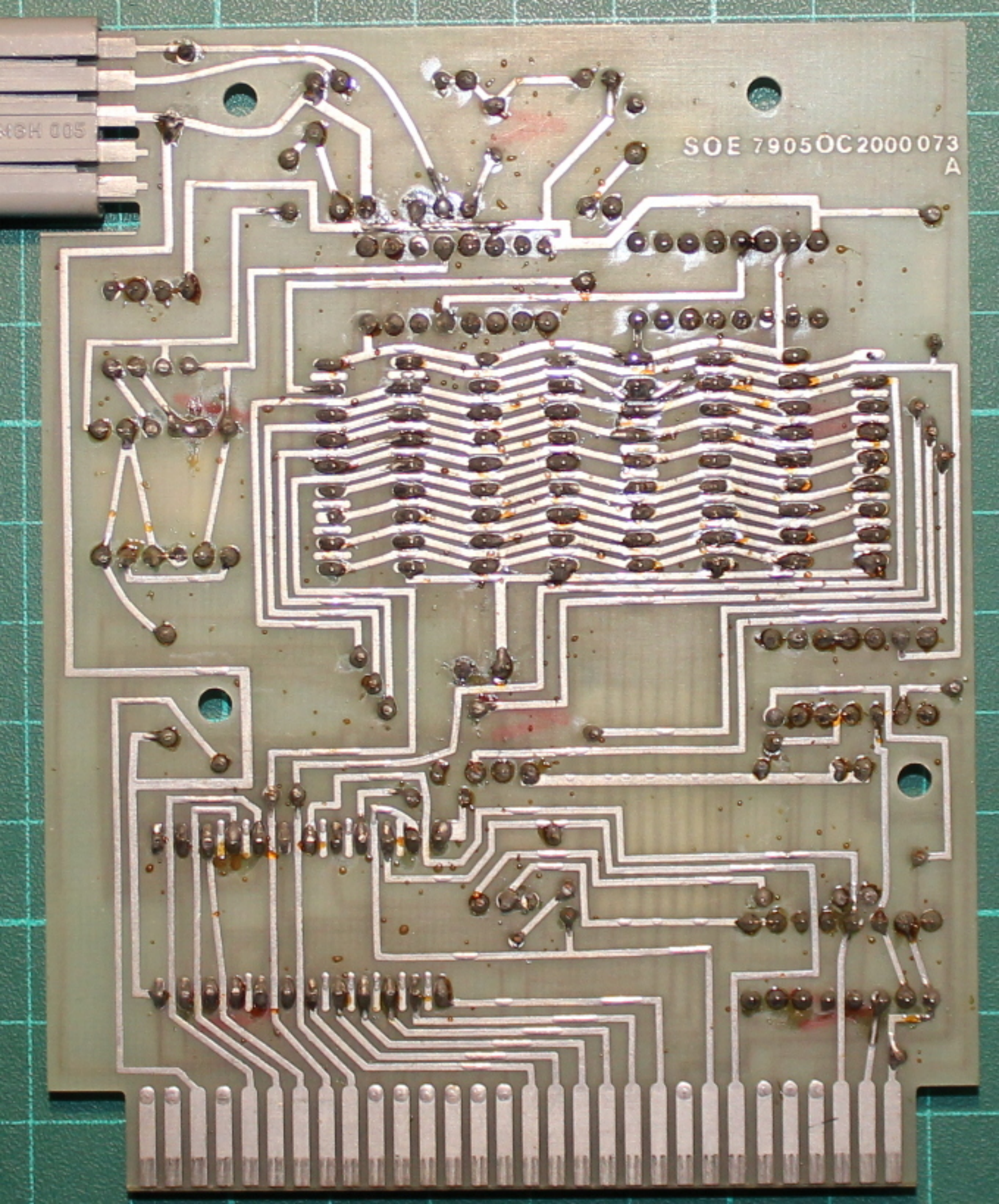

HOBBY COMPUTER


0
4
8
C
+
Ad
Ex
D
-
PC
BP
E


-
+
0
PC
Ad
4
BP
Rx
8
R
W
C
gauche


1
2
3
5
6
7
9
A
b
d
E
F
droite


SOE
HOBBY
COMPUTER
HEF4724BD
7905
SN74LS251N
FF 7936
NEC
IRELAND P96568-924
μPD2114LC
DM74LSOON
DM74LS156N
SBB2616P/AE


SOE 7905OC2000073
A