UNIVERSITE DE GRENOBLE
Service de Mathématiques Appliquées

PROGRAMMATION FONDAMENTALE SUR P.D.P. 8

MANUEL DE REFERENCE

Jean-Pierre PEYRIN

SCALP - M3 - I97I

UNIVERSITE DE GRENOBLE

Service de Mathématiques Appliquées

PROGRAMMATION FONDAMENTALE SUR P.D.P. 8

MANUEL DE REFERENCE

Jean-Pierre PEYRIN

SCALP - M3 - I97I

Je tiens à remercier Jean-Pierre PAILLARD et Jacques VOIRON pour l'aide efficace qu'ils m'ont apportée.

Le P.D.P. 8 (Programmed Data Processor) est une réalisation de la "Digital Equipment Corporation".

DOCUMENTATION

SMALL COMPUTER HANDBOOK

I966/67	P.D.P. 8 Users Handbook
I968/69/70	P.D.P. 8/I Users Handbook

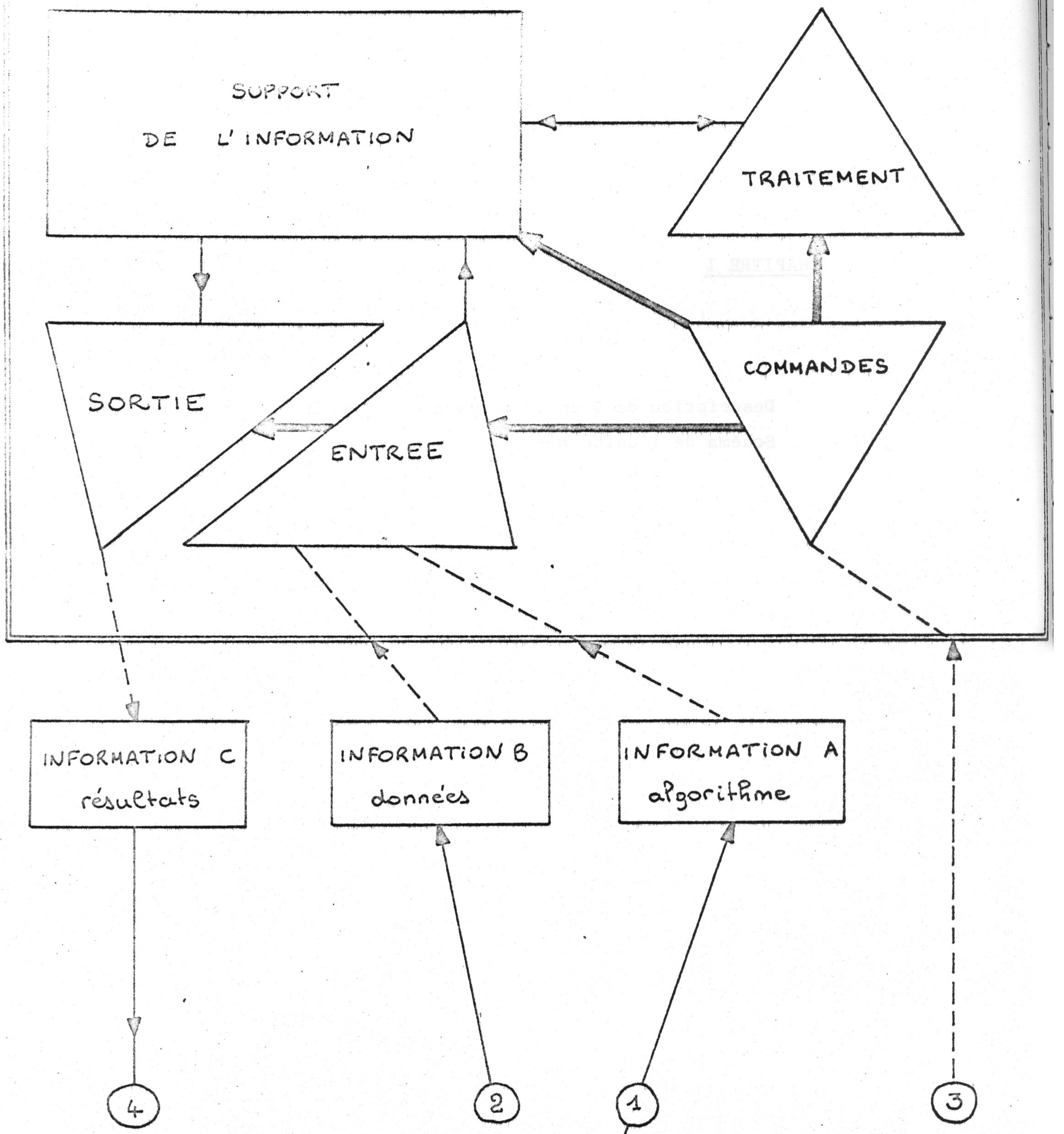
INTRODUCTION TO PROGRAMMING

I969	The P.D.P. 8 Family Computers

édités par D.E.C.

CHAPITRE I

ORDINATEUR


SUPPORT
DE L'INFORMATION
TRAITEMENT
SORTIE
ENTREE
COMMANDES
INFORMATION C
résultats
INFORMATION B
données
INFORMATION A
algorithme
4
2
1
3
PROBLEME

L'UNITE MEMOIRE DU P.D.P. 8

1°) la mémoire centrale

- l'information élémentaire : le support d'une information élémentaire est un tore de ferrite. Un tore de ferrite possède deux états stables : ceux correspondant aux deux valeurs opposées possibles de l'induction rémanente qui y règne après saturation. Ils sont représentés conventionnellement par 1 et 0 (bit).
- l'information fondamentale : le support d'une information fondamentale est un ensemble de 12 tores de ferrite (le mot).
- la mémoire centrale est composée de 4096 × 12 tores de ferrite, soit 4096 mots de 12 bits.
- l'adresse d'un mot : les 4096 mots sont "numérotés" de 0 à 4095. le "numéro" d'un mot s'appelle son adresse.

2°) les registres de l'unité mémoire

Pour "gérer" l'information de la mémoire centrale, deux "mémoires" particulières, distinctes des 4096 mots, sont utilisées. Du fait qu'il s'agit de "mémoires à bascules électroniques", ces "mémoires" sont qualifiées de registres. Ce sont:

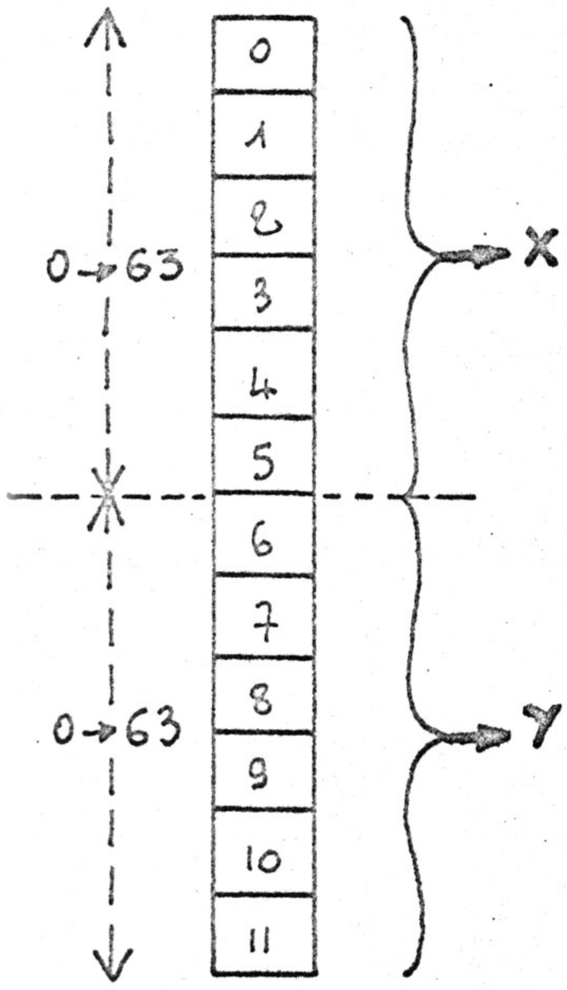
- MA : Memory Address Register (registre d'adresse mémoire).
- MB : Memory Buffer Register (registre tampon pour la mémoire).

3°) les circuits de sélection, de lecture et d'écriture.

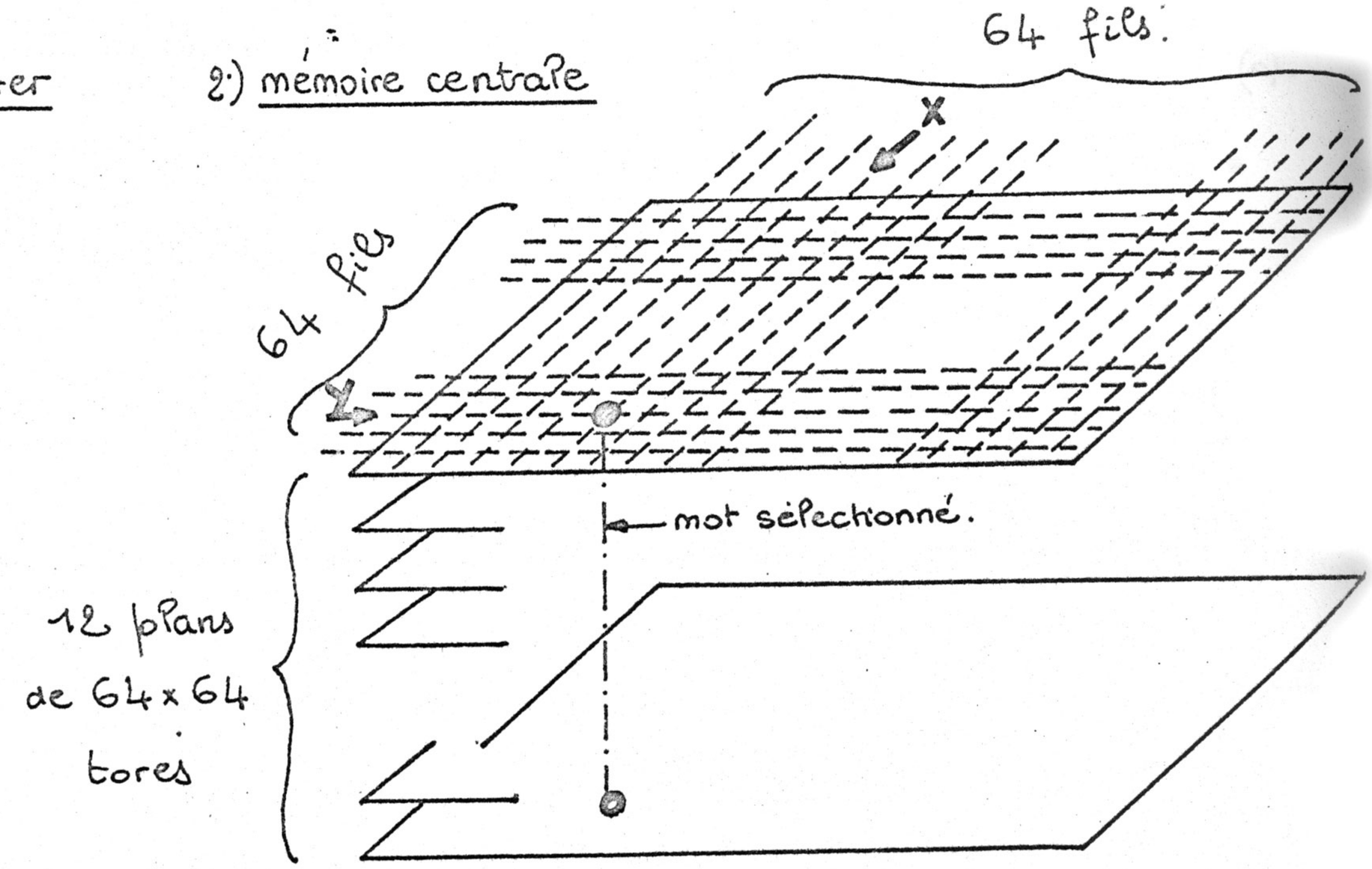
- une opération de lecture dans la mémoire centrale consistera en la recopie dans le MB du mot dont l'adresse se trouve dans le MA.
- une opération d'écriture dans la mémoire centrale consistera en la recopie dans le mot dont l'adresse se trouve dans le MA du contenu du MB.
- pour adresser (sélectionner) l'un des 4096 mots de la mémoire centrale, il faut pouvoir représenter les nombres de 0 à 4095 dans le MA.
- $4095_{(10)} = 7777_{(8)} = 111\ 111\ 111\ 111_{(2)}$. Il faut donc 12 chiffres binaires pour représenter une adresse quelconque.
  le MA est un registre de 12 bits.
- le MB est aussi un registre de 12 bits, du fait qu'il doit contenir les mots en provenance ou à destination de la mémoire.

• SELECTION

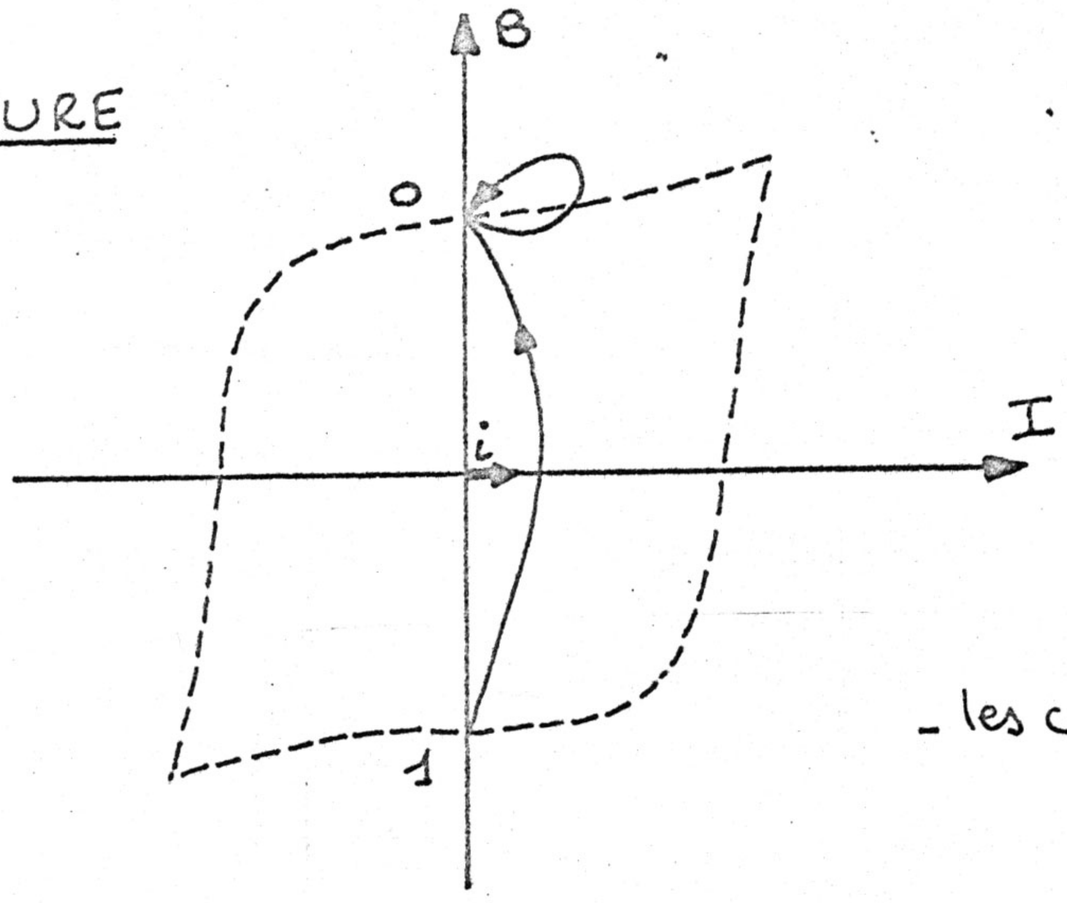
1°) memory address register



2°) mémoire centrale



• LECTURE



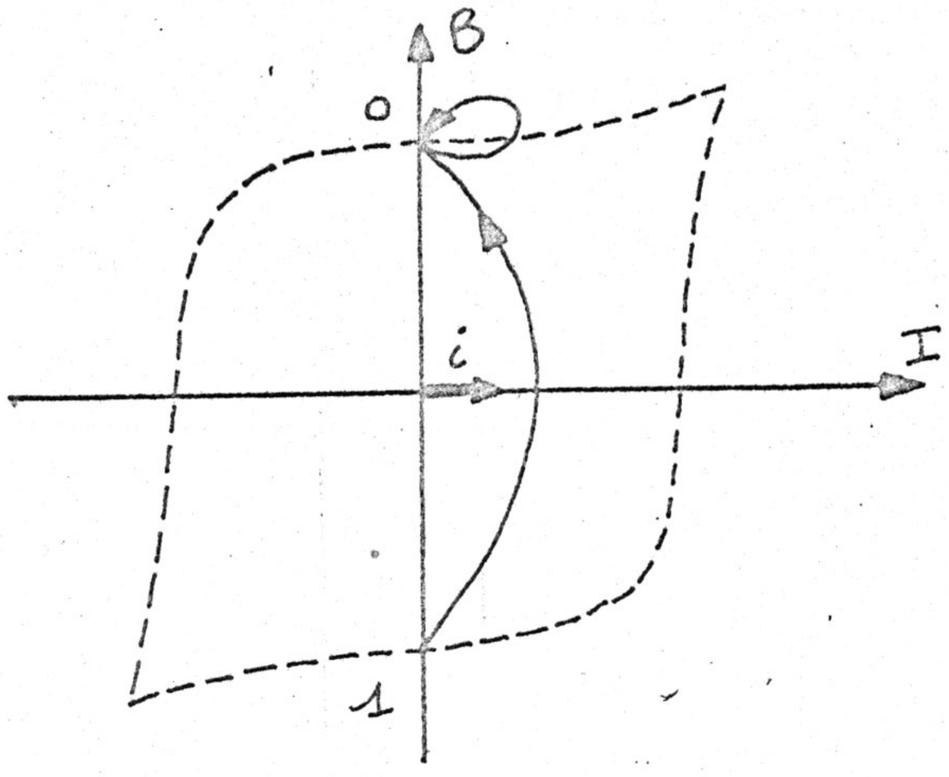
Si 0 :. aucune variation de flux
. aucun courant induit (0)

Si 1 :. une variation de flux
. un courant induit (1).

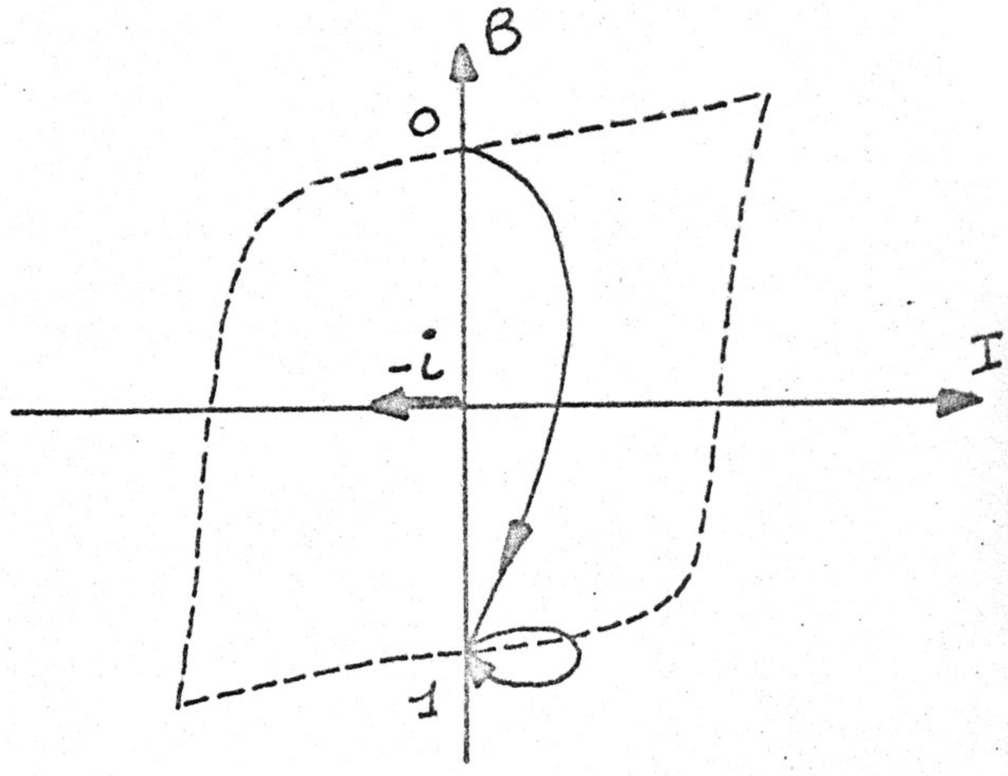
- les courants induits positionnent les bascules du MB

• ECRITURE

écriture d'un 0

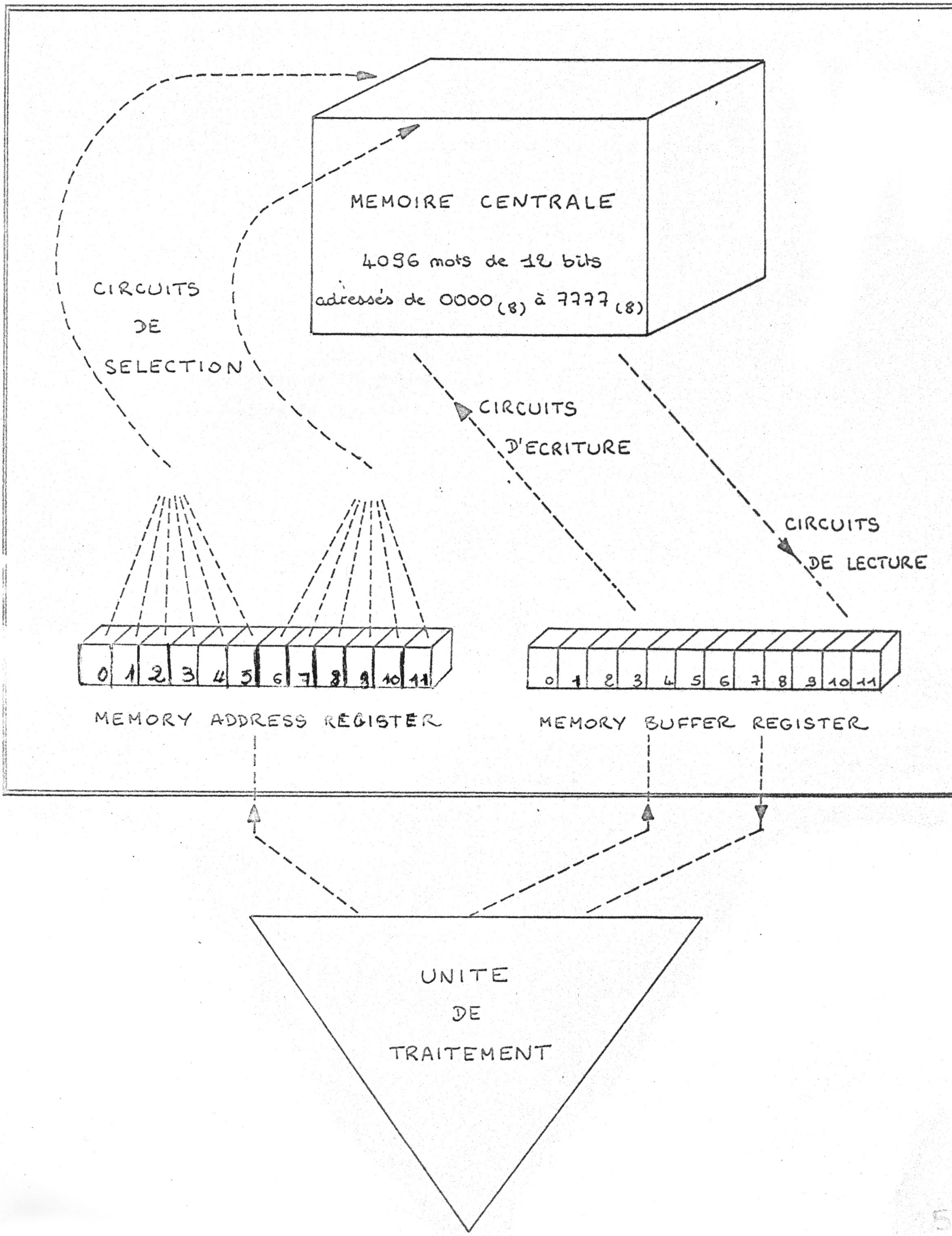


écriture d'un 1



les courants i (0) et -i (1) sont les informations élémentaires du MB

SCHEMA DE : L'UNITE MEMOIRE



CHAPITRE II

UTILISATION DU P.D.P. 8

L'ensemble des mots de 12 bits, transcription d'un algorithme en langage machine (langage binaire propre à la machine), constitue le programme associé à cet algorithme, chaque mot constituant une instruction du programme. Rien ne distingue dans la mémoire, les mots instructions des mots valeurs. La transcription de l'algorithme s'appelle sa programmation, celui qui effectue la transcription s'appelle le programmeur.

PROGRAMMATION DE L'ALGORITHME

le programmeur décide de placer tels ensembles de 12 bits à telles adresses.

↓

CHARGEMENT DU PROGRAMME

le programmeur active les circuits de sélection et d'écriture de l'unité mémoire au moyen du pupitre de contrôle.

↓

EXECUTION DU PROGRAMME

le programmeur active l'unité de traitement au moyen du pupitre de contrôle. Les instructions sont exécutées l'une après l'autre

↓

UTILISATION DES RESULTATS

le programmeur active les circuits de sélection et de lecture de l'unité mémoire au moyen du pupitre de contrôle.

L'UNITE DE TRAITEMENT (PROCESSEUR) DU P.D.P. 8

La mémoire à tores, qui est un organe passif où sont enregistrées valeurs et instructions, sera utilisée par les circuits et les registres du processeur pour répéter les opérations suivantes :

- lecture d'un mot dans la mémoire à tores à une adresse déterminée
- considération de ce mot en tant qu'instruction, analyse de cette instruction et exécution des actions qu'elle spécifie.
- détermination de l'adresse du mot à considérer comme instruction suivante.

1°) LES CIRCUITS DU PROCESSEUR

- des circuits propres à chaque opération élémentaire permettant l'exécution de n'importe quelle instruction.
- des circuits permettant l'enchainement des différentes opérations.

2°) LES REGISTRES DU PROCESSEUR

MA 12	MB 12

- Les deux registres de l'unité mémoire : lieu entre processeur et mémoire centrale.

MEMORY ADDRESS REGISTER - 12 bits = adresse d'un mot à sélectionner.

MEMORY BUFFER REGISTER - 12 bits = information lue ou à écrire.

PC 12

- un registre permettant l'enchainement de l'exécution de deux instructions.

PROGRAM COUNTER - 12 bits - (compteur ordinal) = à la fin de l'exécution d'une instruction, le PC contient l'adresse de l'instruction suivante.

IR 3

- un registre caractérisant le type de l'instruction.

INSTRUCTION REGISTER - 3 bits = Ces 3 bits (les trois bits de gauche de l'instruction) décident du type des actions du processeur pour l'exécution de l'instruction.

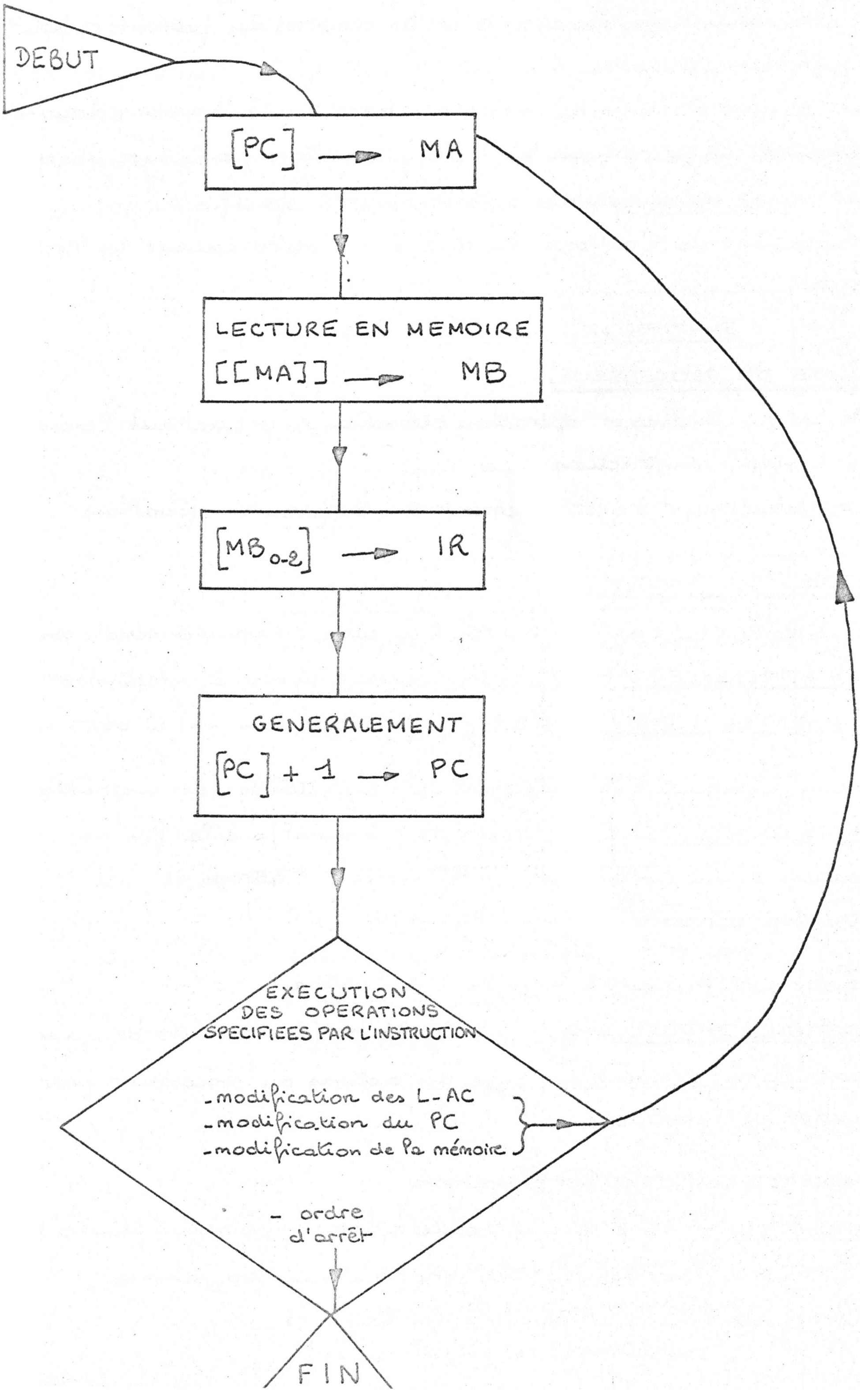
L 1	AC 12

- deux registres modifiables par programme.

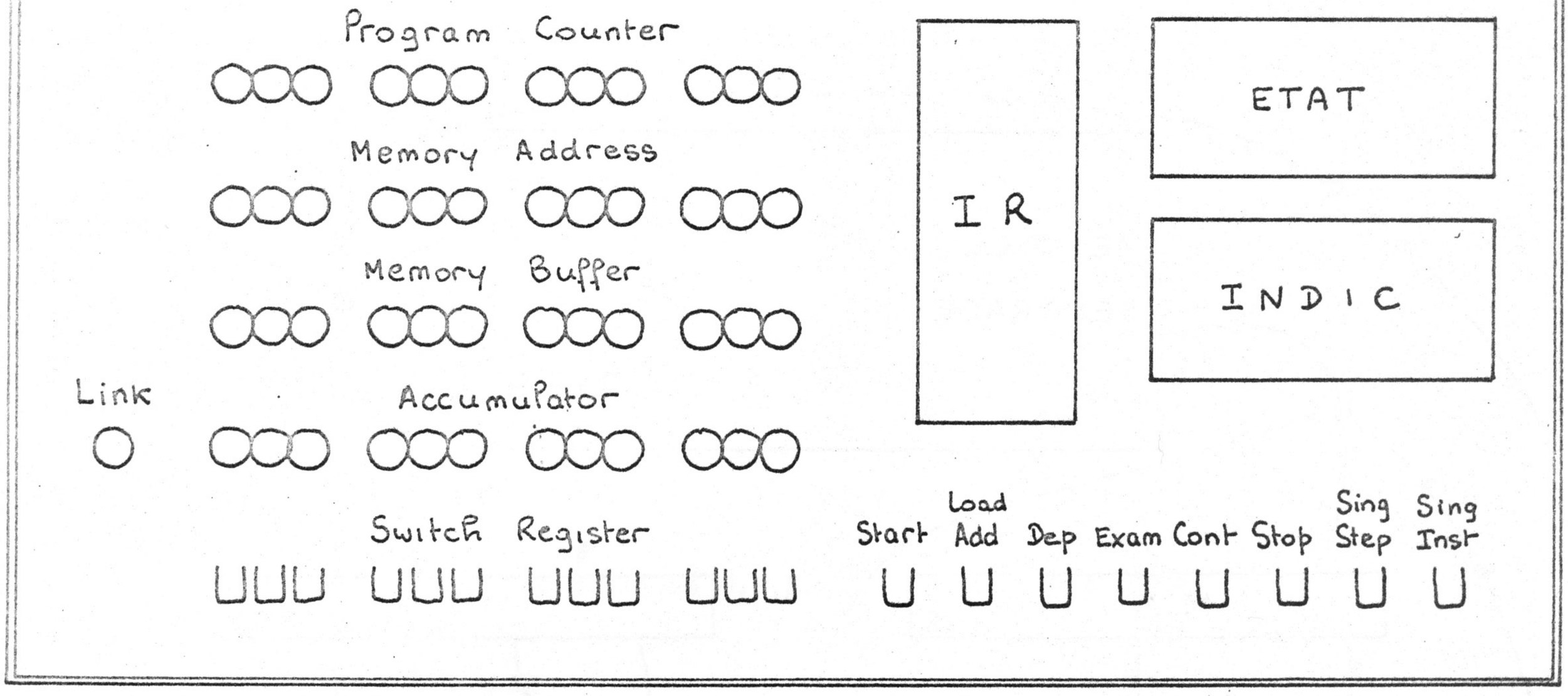
ACCUMULATOR - 12 bits = ce premier registre est "prolongé à gauche" par le

LINK - 1 bit : Ces deux registres permettent au programmeur d'effectuer des calculs arithmétiques et logiques.

ORGANIGRAMME GLOBAL DU FONCTIONNEMENT DU PROCESSEUR



LE PUPITRE DE CONTROLE



VISUALISATION

- les registres PC, MA, MB, L, AC (en binaire).
- le registre IR (en symbolique).
- l'état du processeur et des indications de fonctionnement.

ENTREE

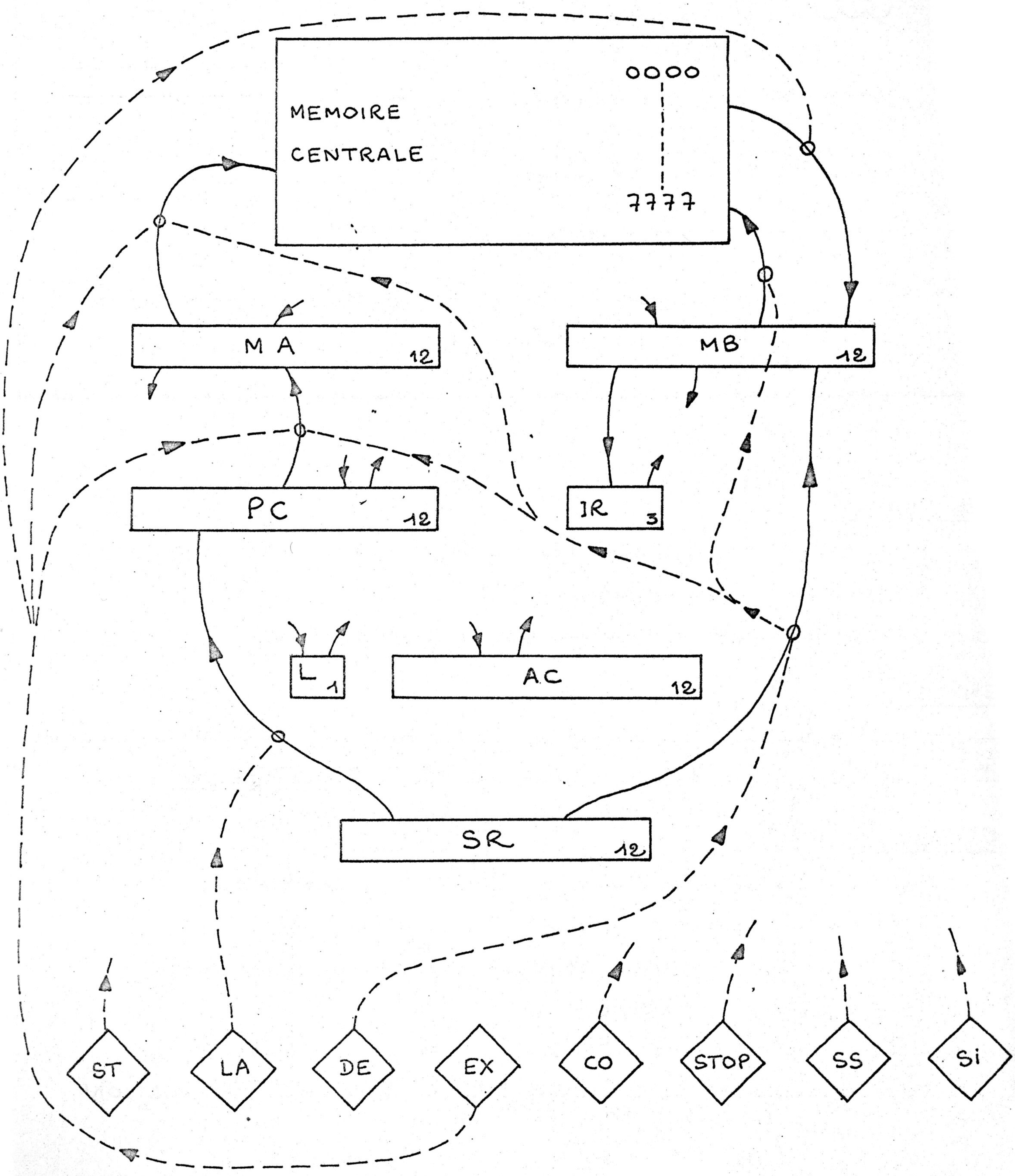
SR 12

le pupitre comporte une rangée de 12 interrupteurs permettant de positionner chaque bit d'un registre de 12 bits, le SR (Switch Register).

COMMANDES

- Mise en route du calculateur : START & CONTINUE
- Arrêt du calculateur : STOP
- [SR] → PC : LOAD ADDRESS
- [SR] → [PC] : DEPOSIT
- [[PC]] → MB : EXAMINE
- Mode d'exécution : SINGLE STEP & SINGLE INSTRUCTION

SCHEMA DE L'ORDINATEUR



CHAPITRE III

INSTRUCTION A OPÉRANDES IMPLICITES

Type 7 : OPeRate instruction

GROUPE OPR1

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	0	0 / CLA	0 / CLL	0 / CMA	0 / CML	0 / R.R	0 / R.L	A / T	0 / IAC

code opération : 7 — contient 0 pour spécifier le groupe OPR1 — CLA — CLL — CMA — CML — RAR RTR — RAL RTL — • Si 0 rotation de 1 position • Si 1 rotation de 2 positions — IAC

GROUPE OPR 2

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0 / CLA	0 / AC<0	0 / AC=0	0 / L=1	S / $\overline{S}$	0 / OSR	0 / HLT	0

code opération : 7 — contient 1 pour spécifier le groupe OPR 2 — CLA — SMA — SZA — SNL — reverse skip • Si 0 test spécifié en 5,6,7 • Si 1 test inverse du test spécifié en 5,6,7 • Si 1 (avec 0 en 5,6,7) saut inconditionnel — OSR — HLT — contient 0 pour spécifier le groupe OPR 2 (1 pour le groupe OPR3

Rappel : $\overline{S1 + S2} = \overline{S1} \cdot \overline{S2}$

GROUPE OPR 1

NOP (No OPeration)

- Code octal : 7000
- Event time : -
- Indicateurs : OPR, FETCH
- temps d'éxécution : 1,5 µs
- opération : Cette instruction provoque un retard d'un cycle (1,5µs) dans le programme avant l'exécution de l'instruction suivante.
  Elle est utilisée pour réaliser des synchronisations.

IAC (Increment ACcumulator)

- Code octal : 7001
- Event time : 2
- Indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : $[L-AC] + 1 \longrightarrow L-AC$

RAL (Rotate Accumulator Left)

- code octal : 7004
- Event time : 2
- Indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération :
  - $[AC_0] \longrightarrow L$
  - $[AC_i] \longrightarrow AC_{i-1}$ $\quad 1 \leq i \leq 11$
  - $[L] \longrightarrow AC_{11}$

RTL (Rotate Two Left)

- Code octal : 7006
- Event time : 2
- Indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération :
  - $[AC_0] \longrightarrow AC_{11}$
  - $[AC_1] \longrightarrow L$
  - $[AC_i] \longrightarrow AC_{i-2}$ $\quad 2 \leq i \leq 11$
  - $[L] \longrightarrow AC_{10}$

RAR (Rotate Accumulator Right)

- Code octal : 7010
- Event time : 2
- Indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération :
  - $[AC_i] \longrightarrow AC_{i+1}$ $\quad 0 \leq i \leq 10$
  - $[AC_{11}] \longrightarrow L$
  - $[L] \longrightarrow AC_0$

RTR (Rotate Two Right)

- Code octal : 7012
- Event time : 2
- Indicateurs : OPR, FETCH
- Temps d'exécution : 1,5 µs
- opération :
  - $[AC_i] \longrightarrow AC_{i+2}$ $\quad 0 \leq i \leq 9$
  - $[AC_{10}] \longrightarrow L$
  - $[AC_{11}] \longrightarrow AC_0$
  - $[L] \longrightarrow AC_1$

CML (CoMplement Link)

- code octal : 7020
- Event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : $\overline{[L]} \longrightarrow L$

CMA : (CoMplement Accumulator)

- code octal : 7040
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : $\overline{[AC_i]} \longrightarrow AC_i \qquad 0 \leq i \leq 11$

CLL (CLear Link)

- code octal : 7100
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : $0 \longrightarrow L$

CLA (CLear Accumulator)

- code octal : 7200
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : $0 \longrightarrow AC_i \qquad 0 \leq i \leq 11$

GROUPE OPR2

HLT (HaLT)

- code octal : 7402
- event time : 1
- indicateurs : OPR, FETCH, pas RUN
- temps d'exécution : 1,5 µs
- opération : 0 ⟶ RUN

OSR (Or with Switch Register)

- code octal : 7404
- event time : 2
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : $[AC_i] \vee [SR_i] \longrightarrow AC_i \qquad 0 \leq i \leq 11$

SKP (unconditional SKiP)

- code octal : 7410
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : [PC] + 1 ⟶ PC

SNL (SKip on Non-zero Link)

- code octal : 7420
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : si [L] = 1, alors [PC] + 1 ⟶ PC

SZL (Skip on Zero Link)

- code octal : 7430
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : si [L] = 0, alors [PC] + 1 → PC

SZA (Skip on Zero Accumulator)

- code octal : 7440
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : si [AC] = 0, alors [PC] + 1 → PC

SNA (Skip on Non-zero Accumulator)

- code octal : 7450
- Event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : si [AC] ≠ 0, alors [PC] + 1 → PC

SMA (Skip on Minus Accumulator)

- code octal : 7500
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : si $[AC_0] = 1$, alors [PC] + 1 → PC

SPA (Skip on Positive Accumulator)

- code octal : 7510
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : si $[AC_0] = 0$, alors $[PC] + 1 \longrightarrow PC$

CLA (CLear Accumulator)

- code octal : 7600
- event time : 1
- indicateurs : OPR, FETCH
- temps d'exécution : 1,5 µs
- opération : $0 \longrightarrow AC_i$ $\quad 0 \leq i \leq 11$

MICROPROGRAMMATION D'INSTRUCTIONS OPR

	Symbolique	Code octal	Symbolique	
OPR 1	CMA IAC	7041	CIA	Complement and Increment Ac.
	CLL RAL	7104		
	CLL RTL	7106		
	CLL RAR	7110		
	CLL RTR	7112		
	CLL CML	7120	STL	SeT Link
	CLA iAC	7201		
	CLA RAL	7204	GLK	Get LinK
	CLA CLL	7300		
	CLA CMA	7240	STA	SeT Accumulator
OPR 2	SZA SNL	7460		
	SNA SZL	7470		
	SMA SNL	7520		
	SPA SZL	7530		
	SMA SZA	7540		
	SPA SNA	7550		
	CLA OSR	7604	LAS	Load Ac with Sr
	SZA CLA	7640		
	SNA CLA	7650		
	SMA CLA	7700		
	SPA CLA	7710		

EXEMPLE

ETAT INITIAL DE LA MEMOIRE

adresse	contenu	adresse	contenu
0513	7001	0522	7404
0514	7006	0523	7440
0515	7430	0524	7410
0516	7000	0525	7600
0517	7200	0526	7510
0520	7040	0527	7420
0521	7010	0530	7402

ACTIONS : 0513 → SR ; LA ; ST ;

PC	MA	MB	L	AC
0514	0513	7001	0	0001
0515	0514	7006	0	0004
0517	0515	7430	0	0004
0520	0517	7200	0	0000
0521	0520	7040	0	7777
0522	0521	7010	1	3777
0523	0522	7404	1	3777
0524	0523	7440	1	3777
0526	0524	7410	1	3777
0530	0526	7510	1	3777
0531	0530	7402	1	3777

CHAPITRE IV

L'ADRESSAGE

1ère convention : ADRESSAGE DIRECT & ADRESSAGE INDIRECT

La convention utilisée consiste à prévoir deux possibilités - la distinction de ces deux possibilités nécessite un bit (le bit 3 de l'instruction, ou bit d'adressage), ce qui laisse huit autres bits.

TYPE :	A.D / A.I	4	5	6	7	8	9	10	11

- Dans le cas de l'adressage direct ($[MB_3]=0$), ces 8 bits permettront de choisir, parmi $2^8 = 256$ mots de la mémoire, un mot qui sera l'opérande.
- Dans le cas de l'adressage indirect ($[MB_3]=1$), ces 8 bits permettront de choisir, parmi $2^8 = 256$ mots de la mémoire, un mot qui contiendra l'adresse de l'opérande

2ème convention : PAGINATION

La convention est la suivante : les $256 = 400_8$ mots sont répartis en deux groupes de 200_8 mots ; la distinction de ces deux groupes nécessite un bit (le bit 4 de l'instruction, ou bit de page).

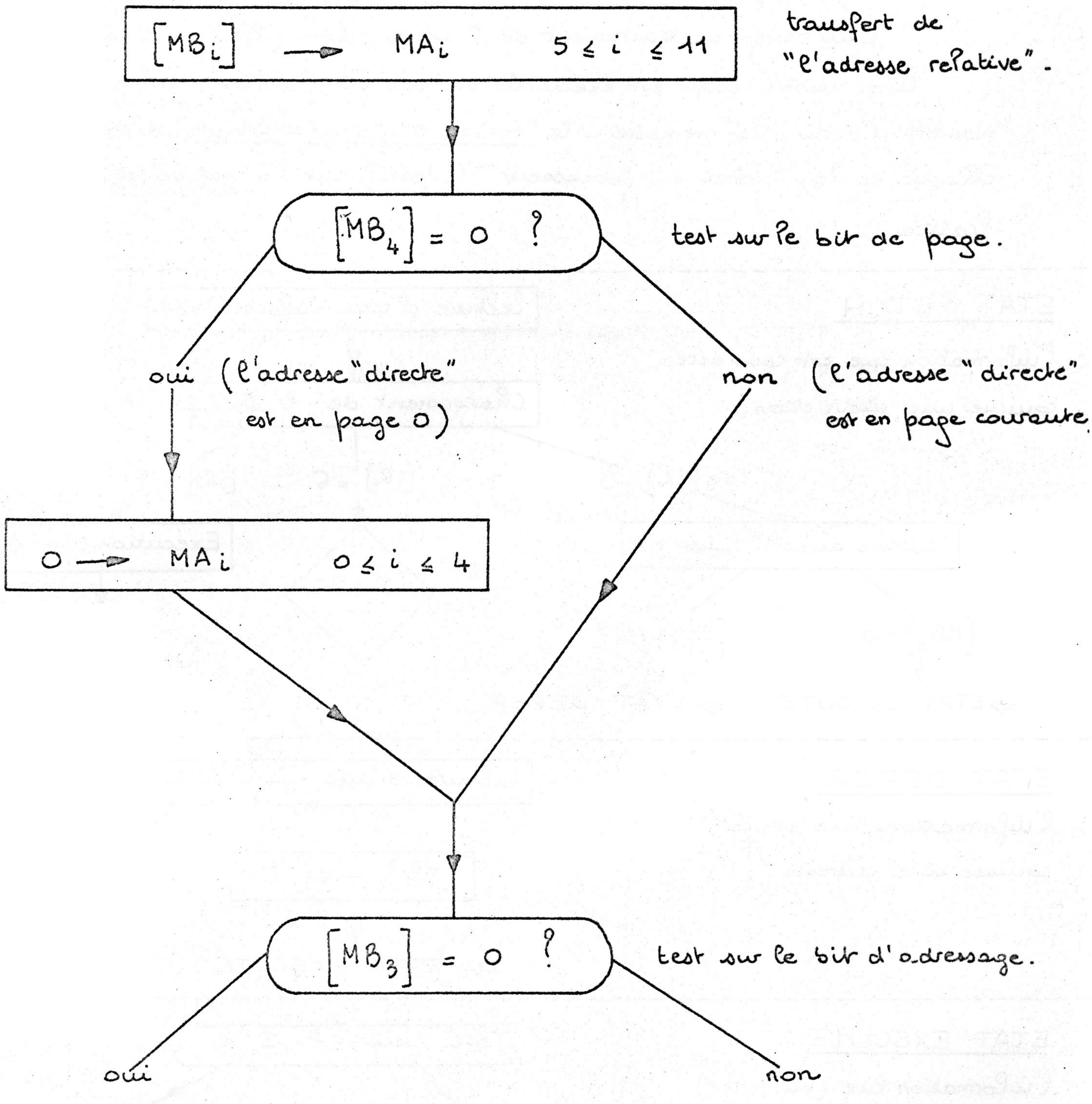
TYPE	A.D / A.I	P / P'	5	6	7	8	9	10	11

- les sept bits de droite sont toujours considérés comme les sept bits de droite de l'adresse de l'un de ces 400_8 mots.
- les cinq bits de gauche de cette adresse sont :
  - soit 0 si $[MB_4] = 0$
  - soit ceux de l'adresse de l'instruction si $[MB_4] = 1$.

→ Les 12 bits ainsi obtenus constituent "l'adresse directe" associée à l'adressage d'une instruction en cours d'exécution.

Les sept bits de droite d'une instruction à opérande en mémoire, pour une certaine valeur du bit 4, correspondent donc à 200_8 mots. Pour le programmeur, la mémoire apparait donc partagée en groupes de 200_8 mots d'adresses consécutives à partir des adresses 0000_8, 0200_8, 0400_8, 0600_8, 1000_8, 1200_8, ..., 7400_8, 7600_8. Les groupes sont appelés des pages.

DETERMINATION DE L'ADRESSE DE L'OPERANDE



LES ETATS DU PROCESSEUR

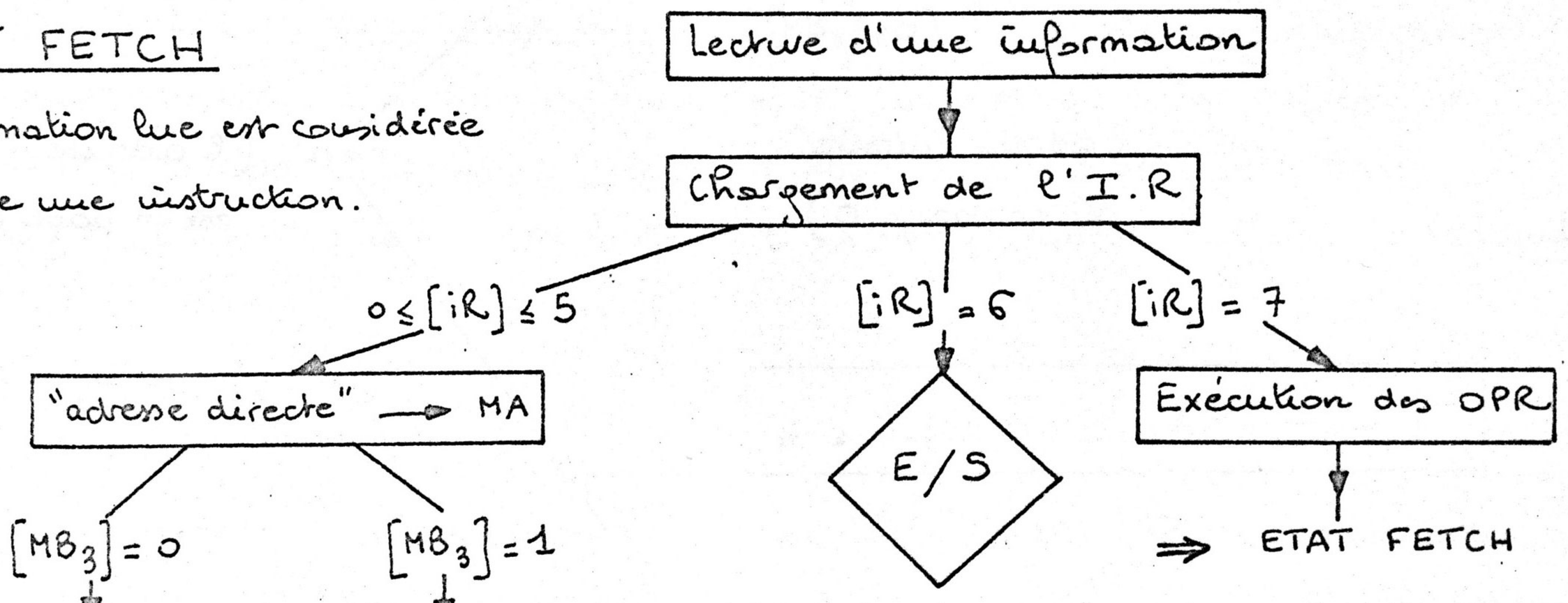
Le fonctionnement du processeur est cyclique - Un cycle (1,5 µs) comporte toujours deux temps :

- un temps d'accès à la mémoire (lecture ou écriture d'une information)
- un temps de traitement de l'information (lue ou à écrire).

Une instruction est exécutée en un ou plusieurs cycles selon le nombre d'accès à la mémoire - le "MAJOR STATE GENERATOR" détermine, pour chaque cycle, "l'état du processeur" (spécifiant la "nature" de l'information traitée).

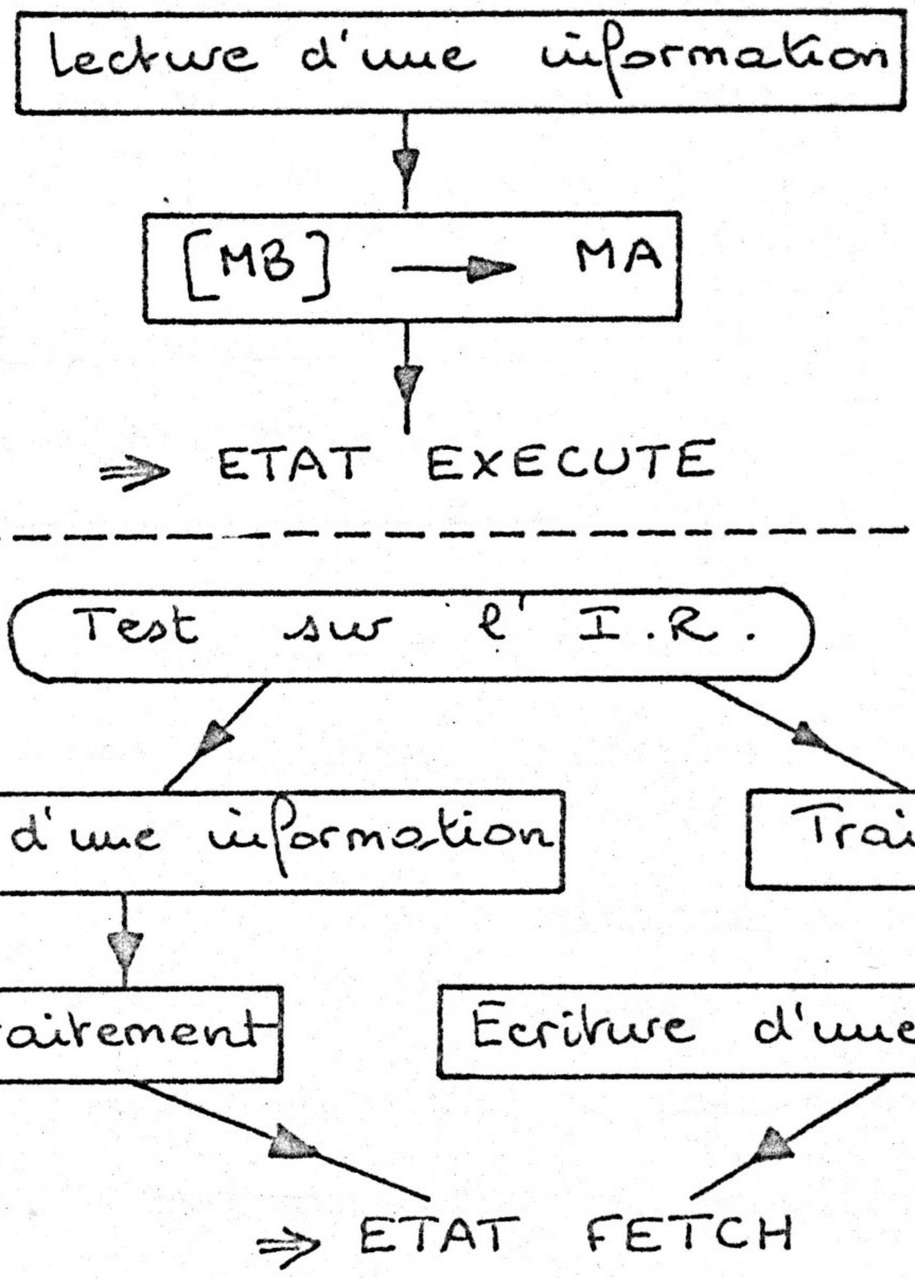
ETAT FETCH

l'information lue est considérée comme une instruction.



ETAT DEFER

l'information lue est considérée comme une adresse.



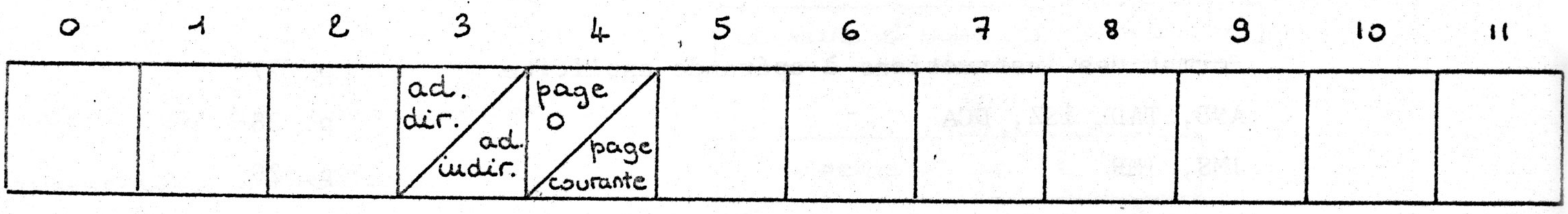
ETAT EXECUTE

l'information lue (ou écrite) est considérée comme une valeur (opérande).

CHAPITRE V

INSTRUCTIONS A OPERANDE EXPLICITE

type 0 : logical AND -
type 1 : Two's complement ADd -
type 2 : Increment and Skip if Zero -
type 3 : Deposit and Clear Accumulator.
type 4 : JuMp to Subroutine -
type 5 : JuMP -



code opération : bit d'adressage ; bit de page

000 : AND
001 : TAD
010 : iSZ
011 : DCA
100 : JMS
101 : JMP

- si 0 l'adressage est direct
- si 1 l'adressage est indirect
- si 0 l'adresse "directe" est en page 0
- si 1 l'adresse "directe" est en page courante

adresse relative

- nombre binaire de sept chiffres représentant l'adresse "relative" de l'opérande "direct" dans la page désignée par le bit 4.
- la mémoire centrale de 4096 mots de 12 bits est conventionnellement divisée en 32 pages de 128 mots.
- pages : 0 → 37 (8).
  adresses relatives : 0 → 177 (8).

---

les bits 4, 5, 6, 7, 8, 9, 10 et 11 permettent la détermination de l'adresse "directe" a.

- si [bit 3) = 0 :
  a est l'adresse de l'opérande en mémoire.
- si [bit 3) = 1 :
  a est l'adresse d'un mot de la mémoire dont le contenu est l'adresse de l'opérande en mémoire.

AND Y (logical AND).

. code octal : 0 . . .

. indicateurs : AND, FETCH, EXECUTE

. temps d'exécution : 3 µs (+ 1,5 µs en cas d'adressage indirect).

. opération : $[AC_i] \wedge [Y_i] \longrightarrow AC_i$ $\quad 0 \leq i \leq 11$

TAD Y (Two's complement ADd).

. code octal : 1 . . .

. indicateurs : TAD, FETCH, EXECUTE

. temps d'exécution : 3 µs (+ 1,5 µs en cas d'adressage indirect).

. opération : $[L.AC] + [Y] \longrightarrow L.AC$

ISZ Y (Increment and Skip if Zero).

. code octal : 2 . . .

. indicateurs : ISZ, FETCH, EXECUTE

. temps d'exécution : 3 µs (+ 1,5 µs en cas d'adressage indirect).

. opération :
- $[Y] + 1 \longrightarrow Y$
- si $[Y] = 0$, alors $[PC] + 1 \longrightarrow PC$

DCA Y (Deposit and Clear Accumulator).

. code octal : 3 . . .

. indicateurs : DCA, FETCH, EXECUTE .

. temps d'exécution : 3 µs (+ 1,5 µs en cas d'adressage indirect).

. opération :
- $[AC] \longrightarrow Y$
- $0 \longrightarrow AC_i$ $\quad 0 \leq i \leq 11$

JMS Y (JuMp to Subroutine)

. code octal : 4 . . .

. indicateurs : JMS, FETCH, EXECUTE

. temps d'exécution : 3 µs (+ 1,5 µs en cas d'adressage indirect).

. opération : • [PC] + 1 ⟶ Y

• Y + 1 ⟶ PC

JMP Y : (JuMP to Y).

. code octal : 5 . . .

. indicateurs : JMP, FETCH

. temps d'exécution : 1,5 µs (+ 1,5 µs en cas d'adressage indirect).

. opération : Y ⟶ PC

EXEMPLE

ETAT INITIAL de la MEMOIRE

adresse	contenu
0201	1200
0202	2205
0203	7006
0204	5600
0205	7777

adresse	contenu
0400	4606
0401	7041
0402	0204
0403	3100
0404	4606
0405	7402
0406	0200

ACTIONS : 0400 → SR ; LA ; ST ;

MSG	iR	PC	MA	MB	L	AC
F	4	0400	0400	4606	0	0000
D	4	0400	0406	0200	0	0000
E	4	0201	0200	0401	0	0000
F	1	0202	0201	1200	0	0000
E	1	0202	0200	0401	0	0401
F	2	0203	0202	2205	0	0401
E	2	0204	0205	0000	0	0401
F	5	0204	0204	5600	0	0401
D	5	0204	0200	0401	0	0401
F	7	0402	0401	7041	0	7377
F	0	0403	0402	0204	0	7377
E	0	0403	0404	4606	0	4206
F	3	0404	0403	3100	0	4206
E	3	0404	0100	4206	0	0000
F	4	0404	0404	4606	0	0000
D	4	0404	0406	0200	0	0000
E	4	0201	0200	0405	0	0000
F	1	0202	0201	1200	0	0000
E	1	0202	0200	0405	0	0405
F	2	0203	0202	2205	0	0405
E	2	0203	0205	0001	0	0405
F	7	0204	0203	7006	0	2024
F	5	0204	0204	5600	0	2024
D	5	0204	0200	0405	0	2024
F	7	0406	0405	7402	0	2024

CHAPITRE VI

/1

FETCH

[PC] → MA

[[MA]] → MB

lecture en mémoire

[MB 0,1,2] → iR

[iR] = 0 ? — oui → AND F /2

non

[iR] = 1 ? — oui → TAD F /2

non

[iR] = 2 ? — oui → iSZ F /2

non

[iR] = 3 ? — oui → DCA F /2

non

[iR] = 4 ? — oui → JMS F /3

non

[iR] = 5 ? — oui → JMP F /4

non

[iR] = 6 ? — oui → iOT F

non

[iR] = 7 → OPR F /5

JMS Y (JuMp to Subroutine).

. code octal : 4...

. indicateurs : JMS, FETCH, EXECUTE

. temps d'exécution : 3 µs (+ 1,5 µs en cas d'adressage indirect).

. opération : • [PC] + 1 ⟶ Y

• Y + 1 ⟶ PC

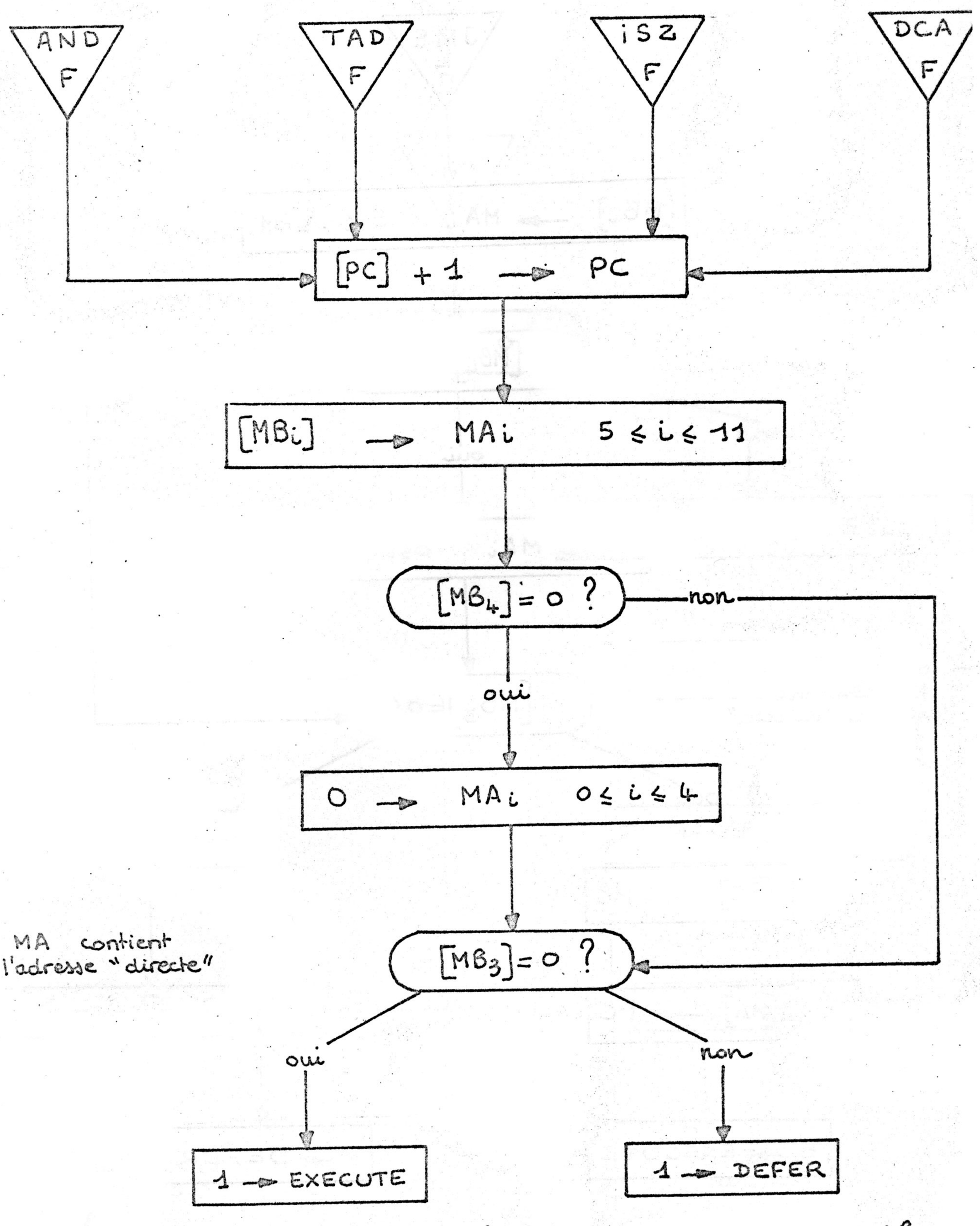
JMP Y : (JuMP to Y).

. code octal : 5...

. indicateurs : JMP, FETCH

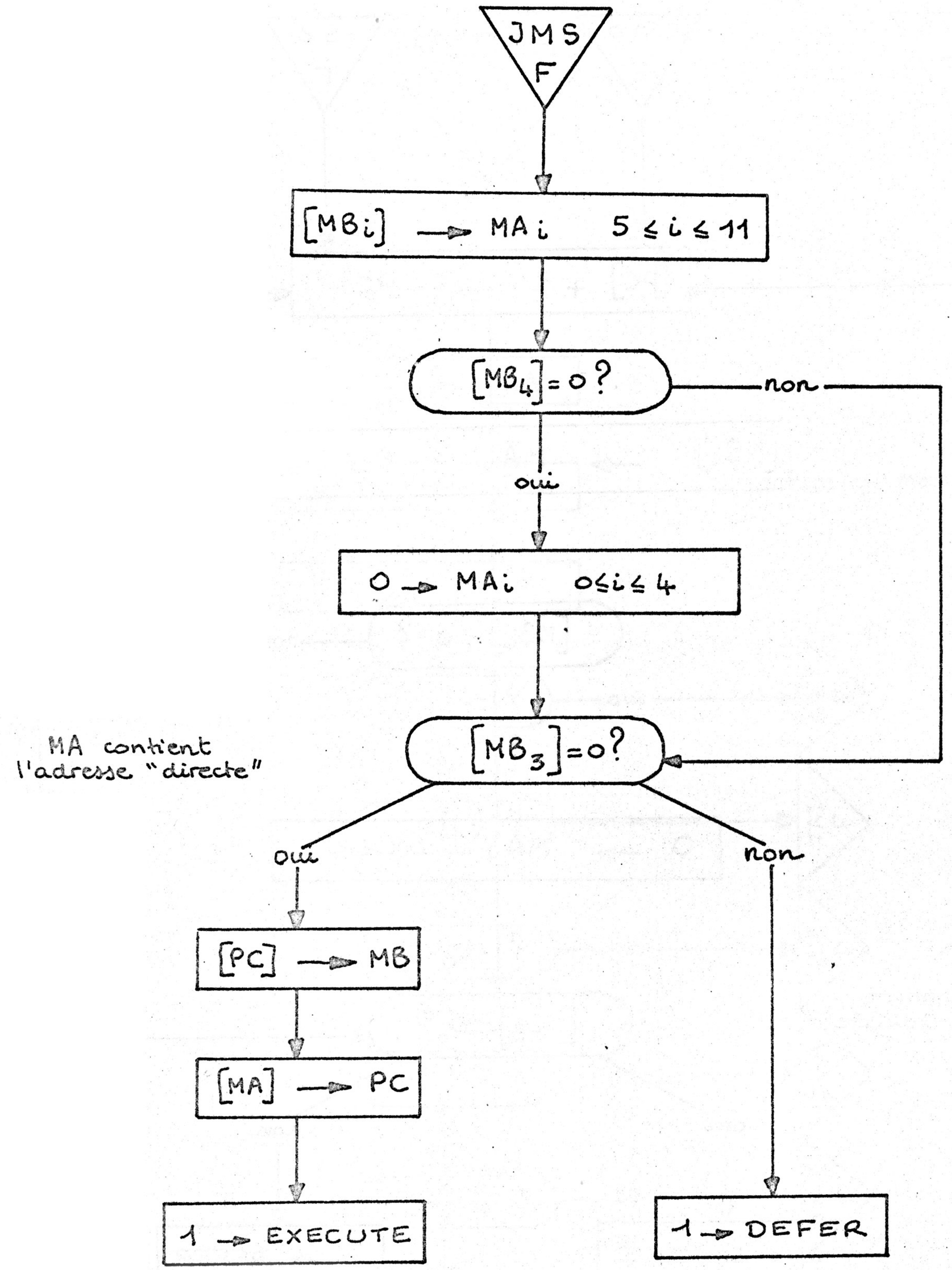
. temps d'exécution : 1,5 µs (+ 1,5 µs en cas d'adressage indirect).

. opération : Y ⟶ PC



adressage direct

adressage indirect



/9

/8

adressage direct

adressage indirect

JMP F

$[MB_3] = 0$?

oui

$[MB_i] \longrightarrow PC_i \quad 5 \leq i \leq 11$

$[MB_4] = 0$?

non

oui

$0 \longrightarrow PC_i \quad 0 \leq i \leq 4$

$1 \longrightarrow$ FETCH

/ 1

adressage direct

PC contient l'adresse "directe"

non

$[MB_i] \longrightarrow MA_i \quad 5 \leq i \leq 11$

$[MB_4] = 0$?

non

oui

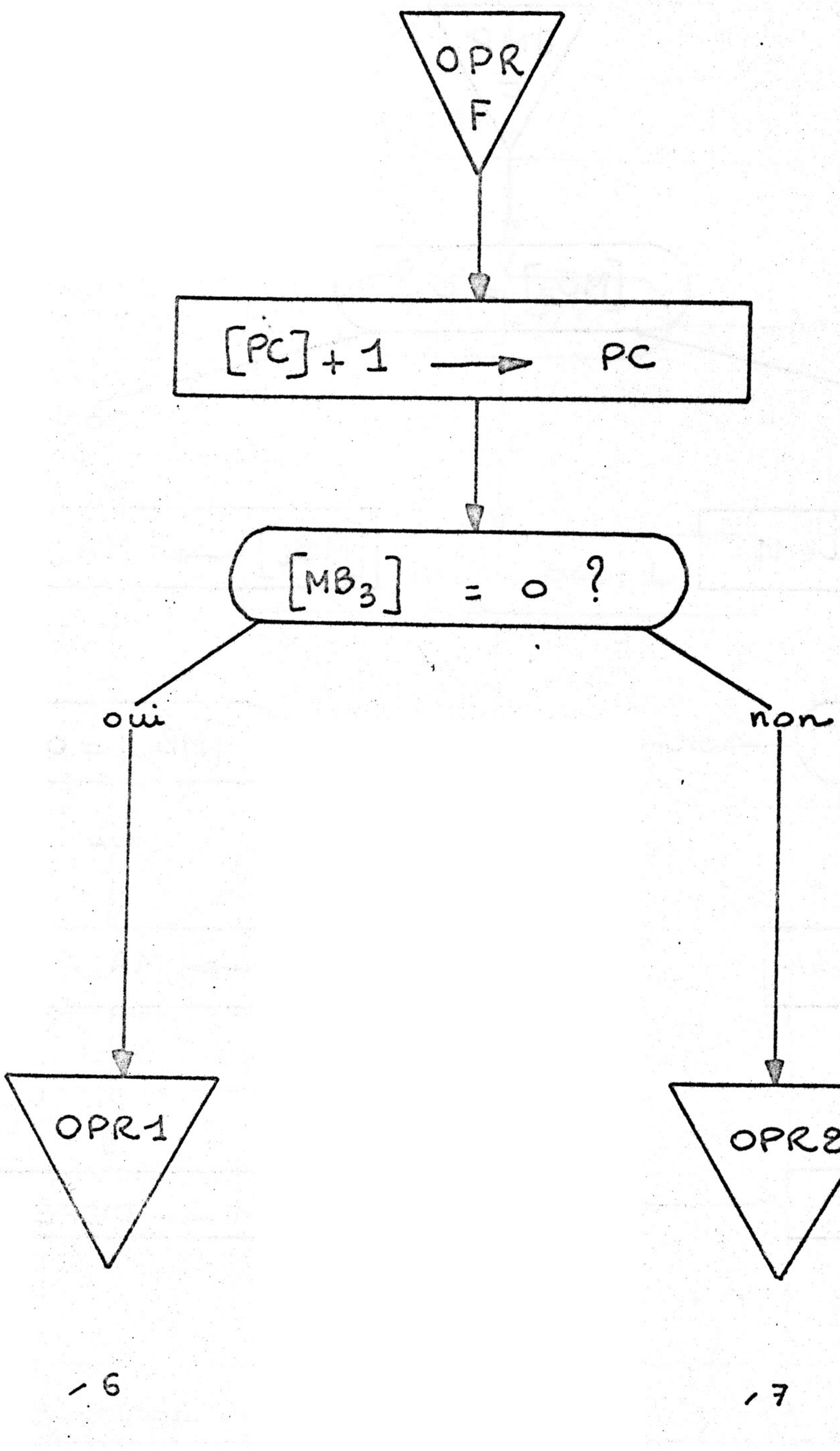
$0 \longrightarrow MA_i \quad 0 \leq i \leq 4$

$1 \longrightarrow$ DEFER

/ 8

adressage indirect

MA contient l'adresse "directe"


OPR
F
[PC] + 1 ⟶ PC
[MB3] = 0 ?
oui
non
OPR1
OPR2

OPR1

[MB4] = 1 ? — oui → 0 → ACi 0≤i≤11

non

[MB5] = 1 ? — oui → 0 → L

non

[MB6] = 1 ? — oui → [ACi] → ACi 0≤i≤11

non

[MB7] = 1 ? — oui → [L] → L

non

[MB8] = 1 ? — non → [MB9] = 1 ? — non → [MB11] = 1 ?

oui

[MB10] = 1 ? — oui

non

0 ≤ i ≤ 10
[ACi] → ACi+1
[AC11] → L
[L] → AC0

oui

[MB10] = 1 ? — oui

non

0 ≤ i ≤ 9
[ACi] → ACi+2
[AC10] → L
[AC11] → AC0
[L] → AC1

oui

[MB10] = 1 ? — oui

non

1 ≤ i ≤ 11
[AC0] → L
[ACi] → ACi-1
[L] → AC11

oui

2 ≤ i ≤ 11
[AC0] → AC11
[AC1] → L
[ACi] → ACi-2
[L] → AC10

oui

[L.AC]+1 → L.AC

non

1 → FETCH

OPR2

[MB11] = 1? — oui → OPR3

non

[MB4] = 1 ? — oui → 0 → ACi 0≤i≤11

non

[MB8] = 0 ? — non → [MB5] = 1 ? — non

oui.

[MB5] = 1 ? — non → [MB6] = 1 ? — non → [MB7] = 1 ? — non

oui

[AC]<0 ? — non

oui

[AC] = 0 ? — non

oui

[L] = 1 ? — non

oui

[PC]+1 → PC

[MB10] = 1 ?

non

oui

0 → RUN

[MB9] = 1 ? — non

oui

[ACi] v [SRi] → ACi 0≤i≤11

1 → FETCH

[MB5] = 1 ? — non

oui

[MB6] = 1 ?

oui

[AC] ≠ 0 — non

oui

[MB7] = 1 ?

oui

[L] = 0? — non

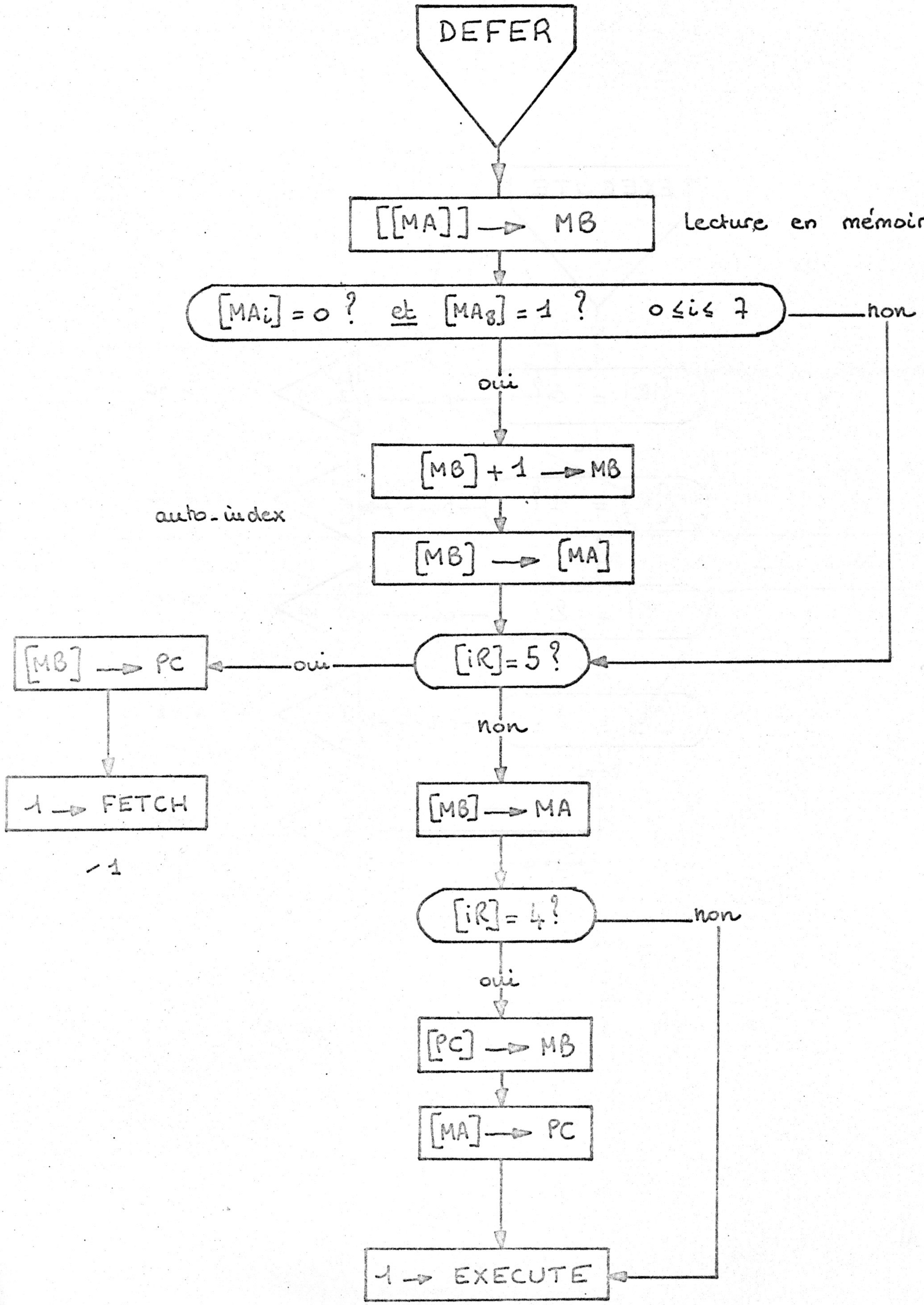
oui

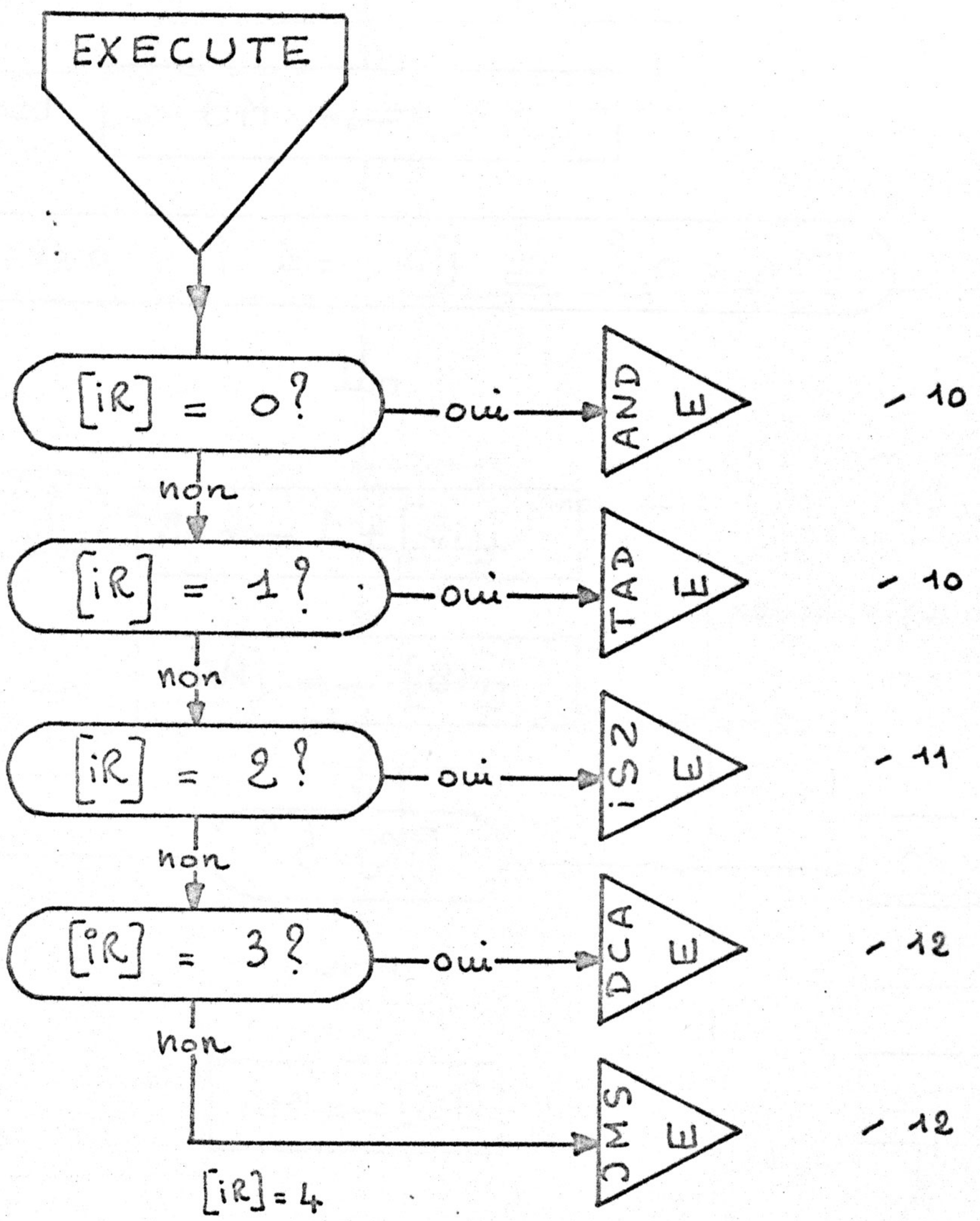
[PC]+1 → PC

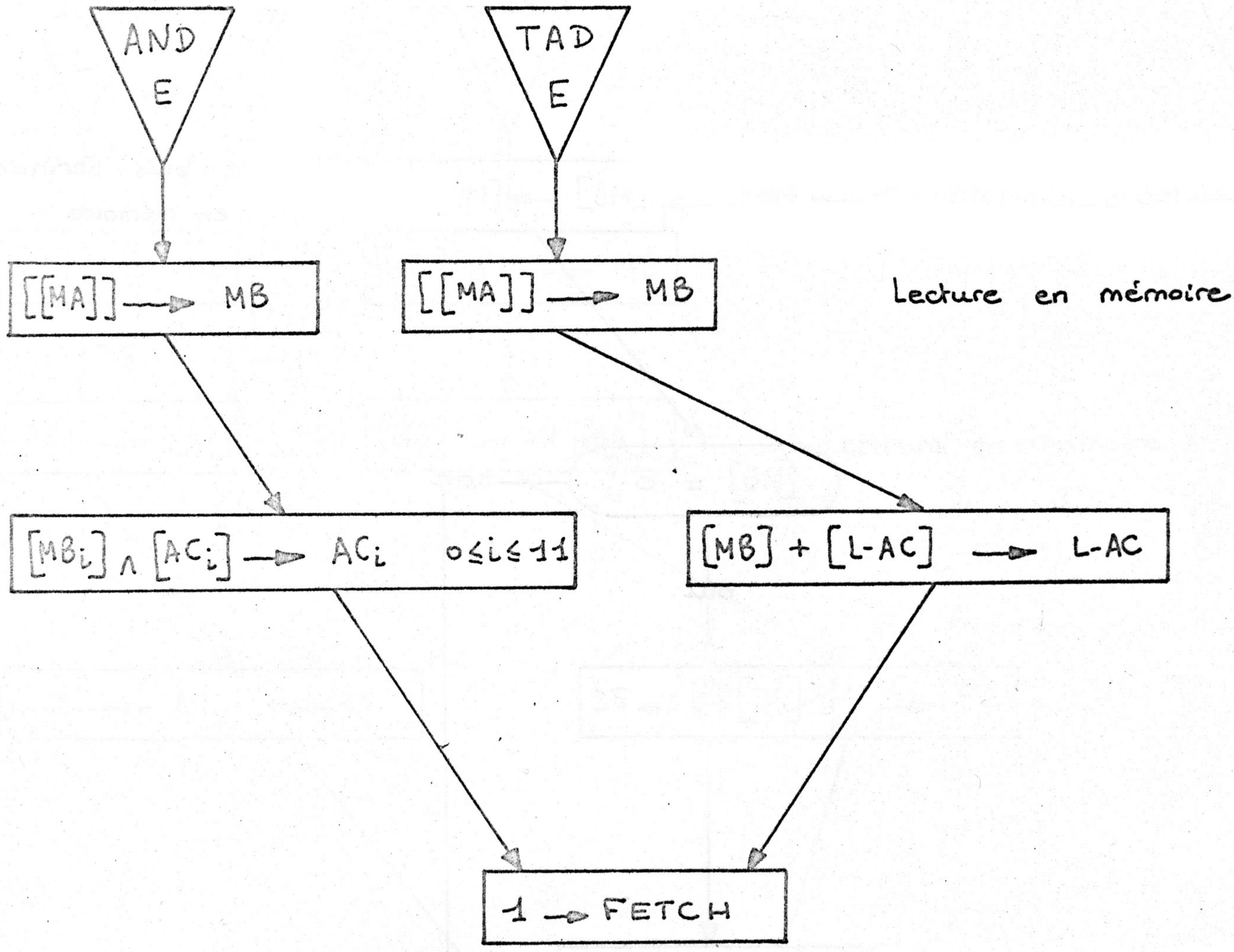
non

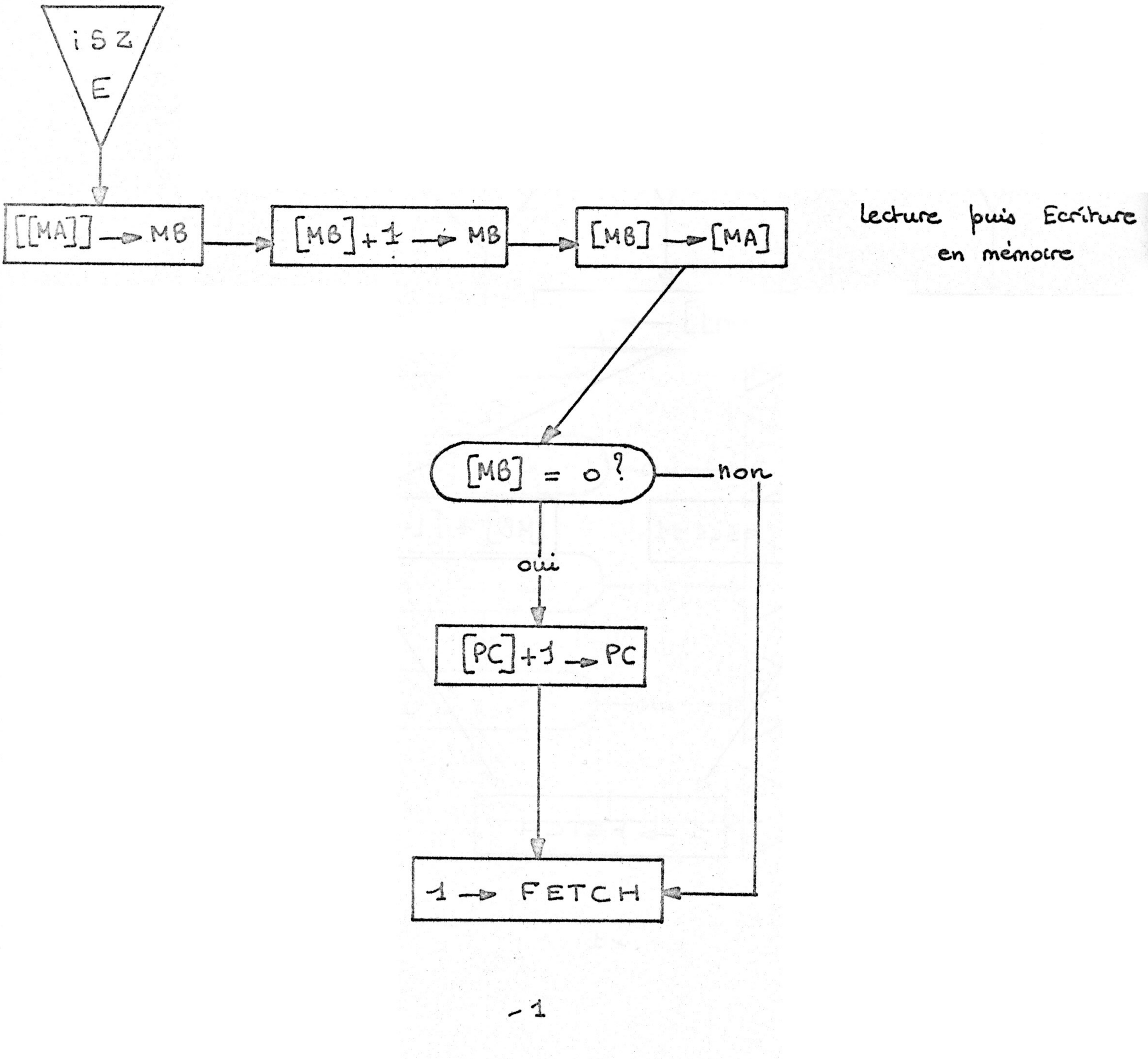
[AC] ≥ 0 ?

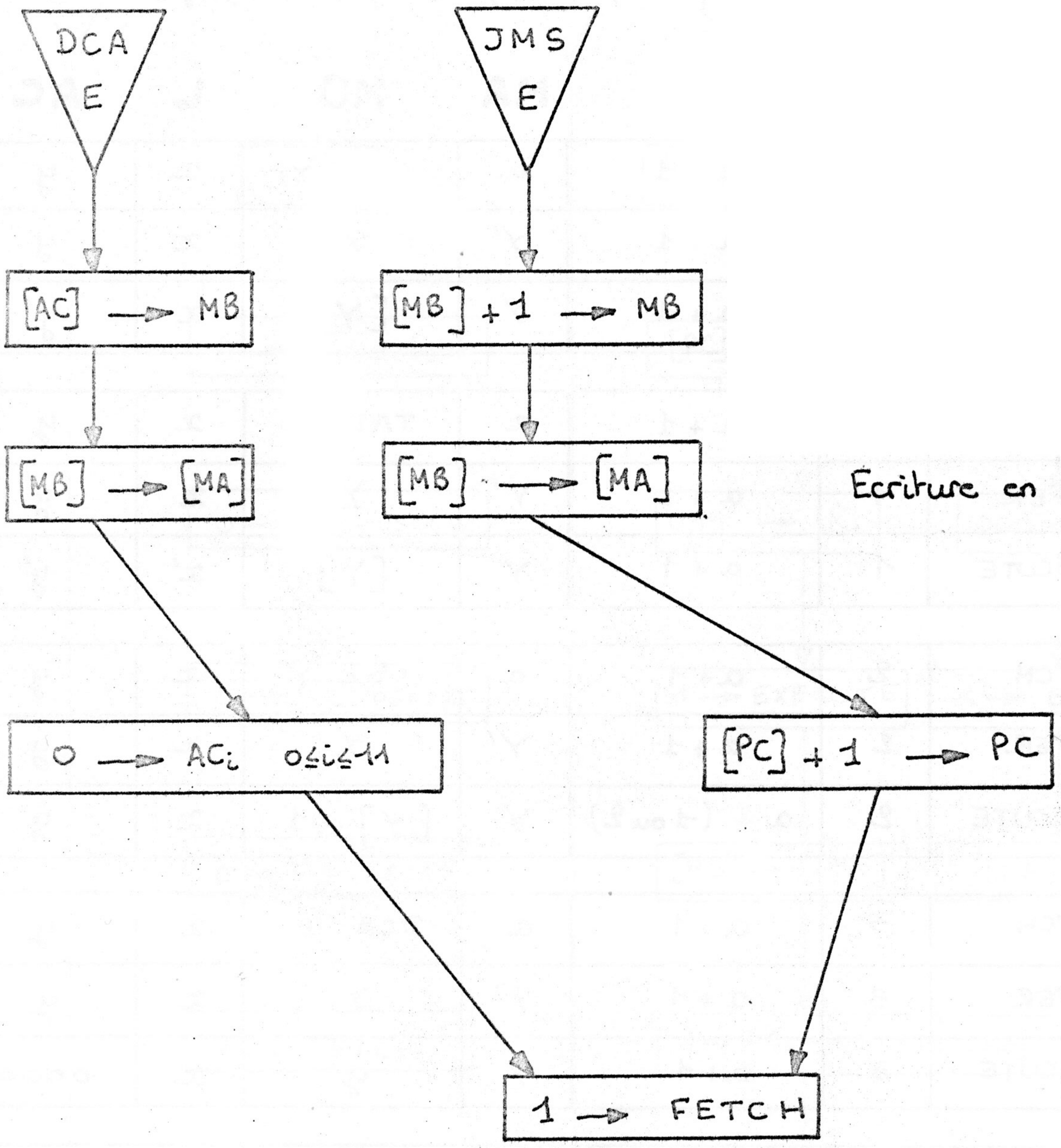
oui


DEFER
[[MA]] → MB
Lecture en mémoire
[MAi] = 0 ? et [MA8] = 1 ? 0 ≤ i ≤ 7
non
oui
[MB] + 1 → MB
auto-index
[MB] → [MA]
[iR] = 5 ?
[MB] → PC
oui
1 → FETCH
non
[MB] → MA
[iR] = 4 ?
non
oui
[PC] → MB
[MA] → PC
1 → EXECUTE


EXECUTE
[iR] = 0 ?
oui
AND W
- 10
non
[iR] = 1 ?
oui
TAA W
- 10
non
[iR] = 2 ?
oui
ISZ W
- 11
non
[iR] = 3 ?
oui
DCA W
- 12
non
[iR] = 4
CMS W
- 12


AND
E
TAD
E
[[MA]] → MB
[[MA]] → MB
Lecture en mémoire
[MBi] ∧ [ACi] → ACi 0≤i≤11
[MB] + [L-AC] → L-AC
1 → FETCH


iSz
E
[[MA]] → MB
[MB] + 1 → MB
[MB] → [MA]
lecture puis Ecriture en mémoire
[MB] = 0 ?
non
oui
[PC] + 1 → PC
1 → FETCH


DCA
E
[AC] → MB
[MB] → [MA]
JMS
E
[MB] + 1 → MB
[MB] → [MA]
Ecriture en mémoire
0 → ACi 0≤i≤11
[PC] + 1 → PC
1 → FETCH

VISUALISATION

Notations: a : adresse de l'instruction -
Y' : adresse "directe" - Y : adresse de l'opérande -
x, y, x', y' : valeurs numériques ; $0 \leq x, x' \leq 1$; $0000 \leq y, y' \leq 7777$ (8)

AND Y

MSG	IR	PC	MA	MB	L	AC
FETCH	0	a+1	a	AND Y	x	y
DEFER	0	a+1	Y'	Y	x	y
EXECUTE	0	a+1	Y	[Y]	x	y'

TAD Y

FETCH	1	a+1	a	TAD Y	x	y
DEFER	1	a+1	Y'	Y	x	y
EXECUTE	1	a+1	Y	[Y]	x'	y'

ISZ Y

FETCH	2	a+1	a	ISZ Y	x	y
DEFER	2	a+1	Y'	Y	x	y
EXECUTE	2	a + (1 ou 2)	Y	[Y]+1	x	y

DCA Y

FETCH	3	a+1	a	DCA Y	x	y
DEFER	3	a+1	Y'	Y	x	y
EXECUTE	3	a+1	Y	y	x	0000

JMS Y

FETCH	4	a	a	JMS Y	x	y
DEFER	4	a	Y'	Y	x	y
EXECUTE	4	Y+1	Y	a+1	x	y

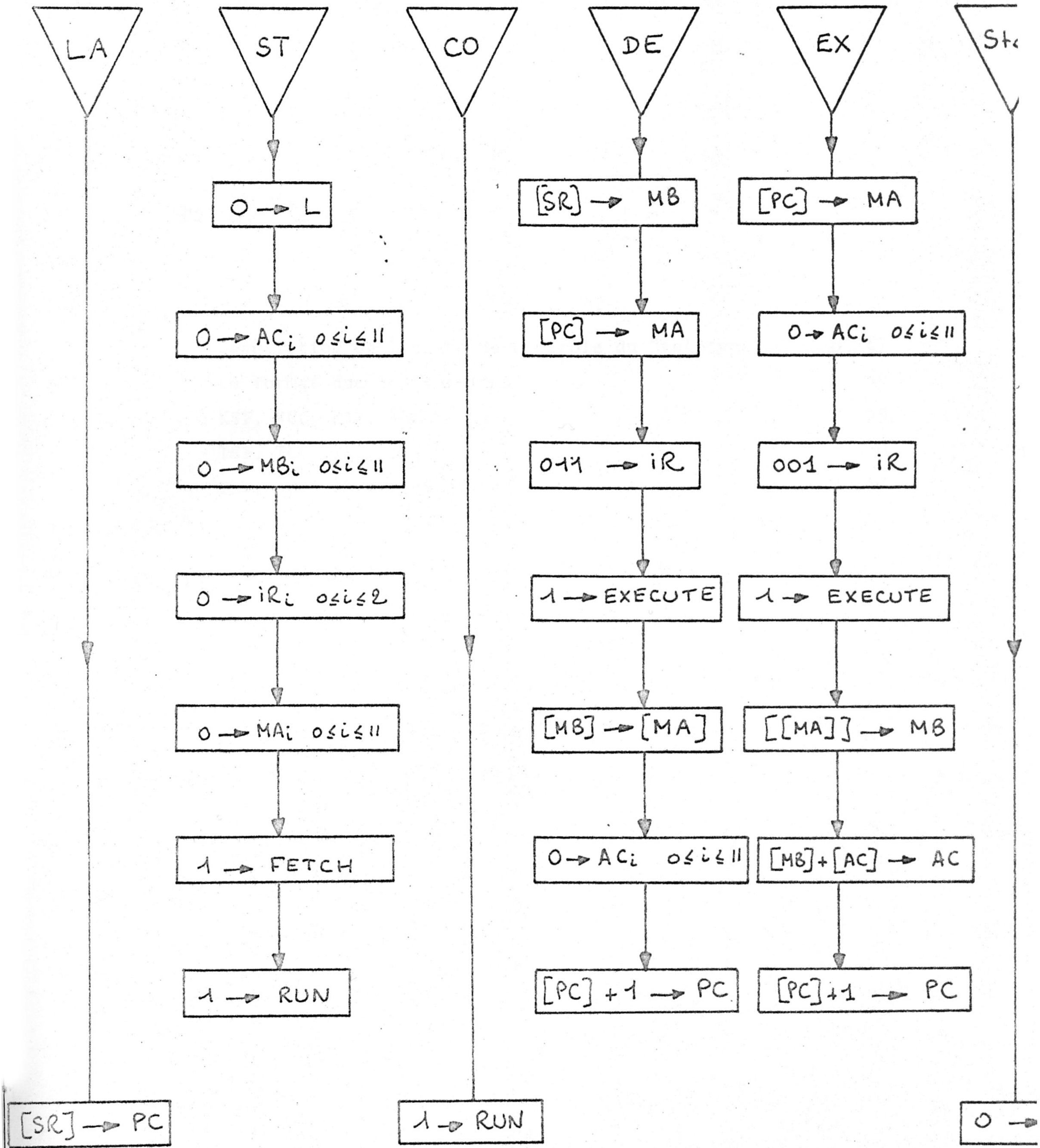
JMP Y

FETCH	5	a	a	JMP Y	x	y
DEFER	5	a	Y'	Y	x	y

OPR :

FETCH	7	a + (1 ou 2)	a	OPR ;	x'	y'

LES COMMANDES MANUELLES



CHAPITRE VII

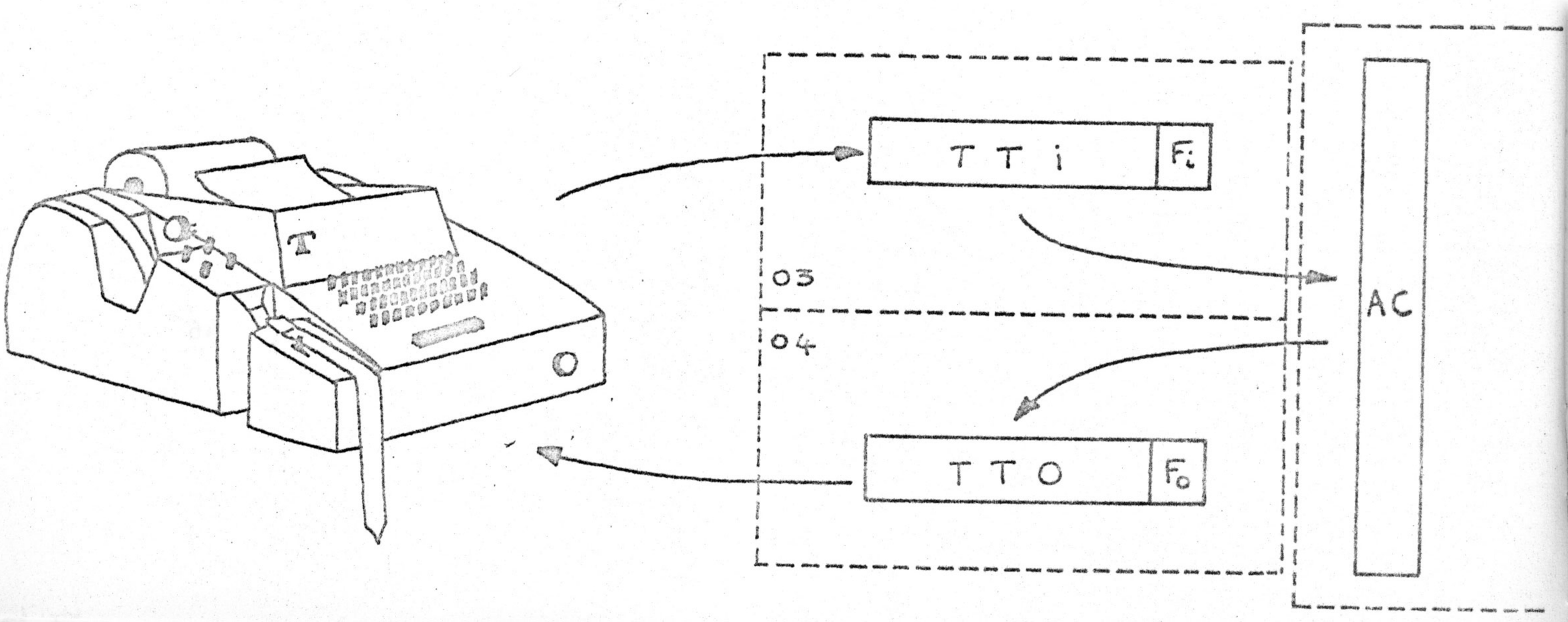
ENTREES-SORTIES A L'AIDE DU "TELETYPE"

A l'aide d'un registre de 8 bits qui joue ici le rôle de sas ("buffer"), l'unité de contrôle du "teletype" effectue des conversions série-parallèle ou parallèle-série - lorsqu'une conversion est effectuée, donc lorsque le registre est "plein" (ou "vide") suivant le sens de la transmission, un indicateur ("flag"), de un bit, sert au processeur à reconnaître qu'un mot de 8 bits peut être transmis vers l'accumulateur en bits 4 à 11 (:ou à partir de l'accumulateur bits 4 à 11).

L'unité de contrôle du "teletype" est formée en fait de deux "machines" respectivement codées $03_{(8)}$ et $04_{(8)}$.

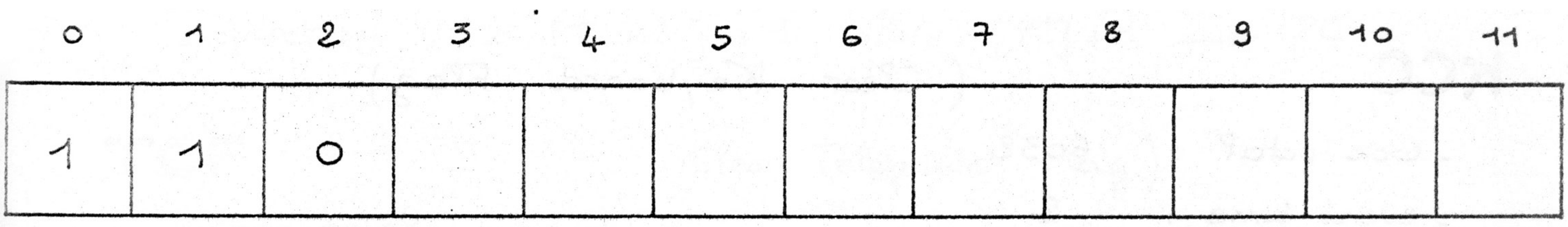
Machine 03 : les codes des caractères, provenant du lecteur de bande ou du téléscripteur, sont reçus par le "TELETYPE INPUT" en série. Le Flag de la machine 03 est positionné sur 1 quand un caractère est arrivé complètement dans le T.T.I. - A l'aide d'une commande programmée le transfert du T.T.I. est effectué en parallèle dans l'accumulateur.

Machine 04 : les codes des caractères dans le "TELETYPE OUTPUT" sont envoyés en série sur le perforateur de bandes et/ou sur le téléimprimeur le Flag de la machine 04 est positionné sur 1 lorsqu'un caractère vient d'être transmis vers le perforateur de bandes et/ou le téléimprimeur, c'est à dire lorsque le T.T.O. est disponible pour une nouvelle utilisation - Une commande programmée permet le transfert en parallèle entre l'accumulateur et le T.T.O.



LES INSTRUCTIONS

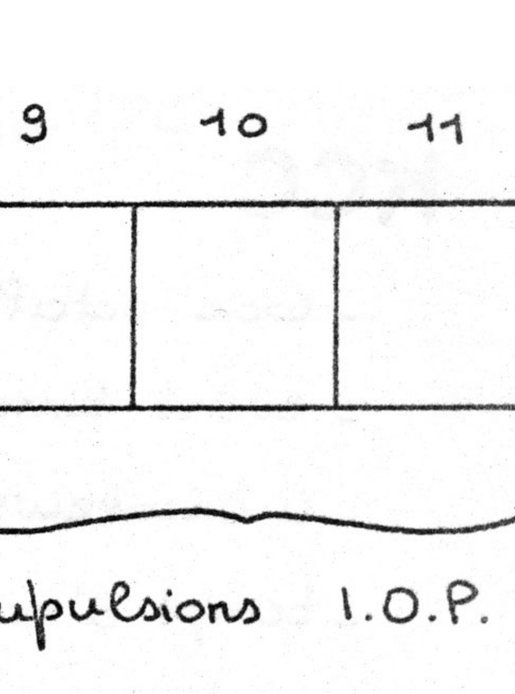
type 6 : Input - Output Transfer



i.O.T : 6

03 : entrée "teletype"

04 : sortie "teletype"



le programmeur gère directement, par programme, les deux fonctions de contrôle du "teletype" ($[AC_{4\text{-}11}] \longrightarrow TTO$; $[TTi] \longrightarrow AC_{4\text{-}11}$).

Ces fonctions sont synchronisées, comme toutes les autres fonctions d'entrée-sortie, par une horloge secondaire synchronisée à l'horloge centrale du processeur.

Quand le code opération 6 est détecté, une bascule "PAUSE" est positionnée sur 1 et un générateur d'impulsions IOP (input-output pulses) produit 3 impulsions : IOP1, IOP2, et IOP4 espacées d'une microseconde. Les intervalles sont appelés, respectivement, les Event Time 1, 2 et 3.

1°) <u>les instructions d'ENTREE</u>

KSF (Skip on Keyboard Flag).

- code octal : 6031
- event time : 1
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération : si [FLAG] = 1, alors [PC] + 1 ⟶ PC

KCC (Clear Keyboard Flag).

- code octal : 6032
- event time : 2
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération : . 0 ⟶ FLAG
  . 0 ⟶ AC_i $\quad 0 \leq i \leq 11$

KRS (Read Keyboard buffer Static).

- code octal : 6034
- event time : 3
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération : $[BUFFER_i] \vee [AC_j] \longrightarrow AC_j \quad 0 \leq i \leq 7 \; ; \; 4 \leq j \leq 11$

KRB (Read Keyboard Buffer dynamic)

- code octal : 6036
- event time : 2,3
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération : . 0 ⟶ FLAG
  . 0 ⟶ AC_i $\quad 0 \leq i \leq 11$
  . $[BUFFER_i] \vee [AC_j] \longrightarrow AC_j \quad 0 \leq i \leq 7 \; ; \; 4 \leq j \leq 11$

2°) Les instructions de SORTIE

TSF (Skip on Teleprinter Flag)

- code octal : 6041
- event time : 1
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération : si [FLAG] = 1, alors [PC] + 1 ⟶ PC

TCF (Clear Teleprinter Flag)

- code octal : 6042
- event time : 2
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération : 0 ⟶ FLAG

TPC (load Teleprinter and Print)

- code octal : 6044
- event time : 3
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération : $[AC_i] \longrightarrow BUFFER_j \quad 4 \leq i \leq 11 \; ; \; 0 \leq j \leq 7$

TLS (load Teleprinter Sequence)

- code octal : 6046
- event time : 2,3
- indicateurs : IOT, FETCH, PAUSE
- temps d'exécution : 3,75 µs
- opération :
  - 0 ⟶ FLAG
  - $[AC_i] \longrightarrow BUFFER_j \quad 4 \leq i \leq 11 \; ; \; 0 \leq j \leq 7$

TABLE DES CODES

TELETYPE 33 ASR/KSR

CODE OCTAL (ASCii)

A	:	301
B	:	302
C	:	303
D	:	304
E	:	305
F	:	306
G	:	307
H	:	310
i	:	311
J	:	312
K	:	313
L	:	314
M	:	315
N	:	316
O	:	317
P	:	320
Q	:	321
R	:	322
S	:	323
T	:	324
U	:	325
V	:	326
W	:	327
X	:	330
Y	:	331
Z	:	332
0	:	260
1	:	261
2	:	262
3	:	263
4	:	264
5	:	265
6	:	266
7	:	267
8	:	270
9	:	271

!	:	241
"	:	242
#	:	243
$	:	244
%	:	245
&	:	246
'	:	247
(	:	250
)	:	251
*	:	252
+	:	253
,	:	254
-	:	255
.	:	256
/	:	257
:	:	272
;	:	273
<	:	274
=	:	275
>	:	276
?	:	277
@	:	300
[	:	333
\	:	334
]	:	335
↑	:	336
←	:	337

amorce et fin de bande	:	200
avancement (line-feed)	:	212
retour-chariot (return)	:	215
espace	:	240
effacement (rub-out)	:	377
rien	:	000

EXEMPLE

ETAT INITIAL DE LA MEMOIRE

adresse	contenu
0400	7300
0401	6046
0402	6031
0403	5202
0404	6036
0405	6041
0406	5205
0407	5201

ACTIONS : 1 → Si ; 0400 → SR ; LA ; ST

ENTREES — SORTIES

	INDICATEURS	PC	MA	MB	L	AC	
	OPR, FETCH	0401	0400	7300	0	0000	
	iOT, FETCH, PAUSE	0402	0401	6046	0	0000	
	iOT, FETCH, PAUSE	0403	0402	6031	0	0000	
	JMP, FETCH	0403	0403	5202	0	0000	
	iOT, FETCH, PAUSE	0403	0402	6031	0	0000	
	JMP, FETCH	0403	0403	5202	0	0000	
	iOT, FETCH, PAUSE	0403	0402	6031	0	0000	
	JMP, FETCH	0403	0403	5202	0	0000	
A	iOT, FETCH, PAUSE	0404	0402	6031	0	0000	
	iOT, FETCH, PAUSE	0405	0404	6036	0	0301	
B	iOT, FETCH, PAUSE	0407	0405	6041	0	0301	
	JMP, FETCH	0407	0407	5201	0	0301	
	iOT, FETCH, PAUSE	0402	0401	6046	0	0301	A
	iOT, FETCH, PAUSE	0404	0402	6031	0	0301	
	iOT, FETCH, PAUSE	0405	0404	6036	0	0302	
	iOT, FETCH, PAUSE	0407	0405	6041	0	0302	
	JMP, FETCH	0407	0407	5201	0	0302	
	iOT, FETCH, PAUSE	0402	0401	6046	0	0302	B
	iOT, FETCH, PAUSE	0403	0402	6031	0	0302	
	JMP, FETCH	0403	0403	5202	0	0302	
	iOT, FETCH, PAUSE	0403	0402	6031	0	0302	
	JMP, FETCH	0403	0403	5202	0	0302	
	iOT, FETCH, PAUSE	0403	0402	6031	0	0302	
C	JMP, FETCH	0403	0403	5202	0	0302	
	iOT, FETCH, PAUSE	0404	0402	6031	0	0302	
	iOT, FETCH, PAUSE	0405	0404	6036	0	0303	
	iOT, FETCH, PAUSE	0407	0405	6041	0	0303	
	JMP, FETCH	0407	0407	5201	0	0303	
	iOT, FETCH, PAUSE	0402	0401	6046	0	0303	C
	⋮	⋮	⋮	⋮	⋮	⋮	

CHAPITRE VIII

LES INTERRUPTIONS

• Deux instructions de type 6 .

ION (Interrupt turn ON)

- code octal : 6001
- event time : -
- indicateurs : IOT, FETCH, ION
- temps d'exécution : 1,5 µs
- opération : 1 → INT

IOF (Interrupt turn OFf)

- code octal : 6002
- event time : -
- indicateurs : IOT , FETCH
- temps d'exécution : 1,5 µs
- opération : 0 → INT

• Réalisation pratique d'une interruption :

Quand une information, dans le "buffer" d'entrée d'un périphérique, est prête à être transmise dans l'accumulateur, le "flag" correspondant est positionné sur 1. L'unité de contrôle du périphérique envoie à cet instant une "demande d'interruption" au processeur. le processeur n'en tient compte que s'il est en mode "interruption possible" (ION). Dans ce cas :

- le programme qui est en train de s'exécuter est arrêté à la fin de l'instruction en cours d'exécution.
- le contenu du PC est rangé en mémoire à l'adresse $0000_{(8)}$.
- Puis le contenu du PC est remplacé par $0001_{(8)}$; l'enchainement habituel d'exécution des instructions reprend, permettant l'exécution d'un programme assurant des actions associées à l'occurence de l'interruption.

La ligne de commande d'interruption étant commune à tous les dispositifs de transfert d'information, ce programme de "traitement de l'interruption" devra assurer les fonctions suivantes :

- chercher d'où provient l'interruption (en testant tous les "flags").
- Exécuter des actions associées à l'occurence de l'interruption -
- Redonner le contrôle au programme principal.

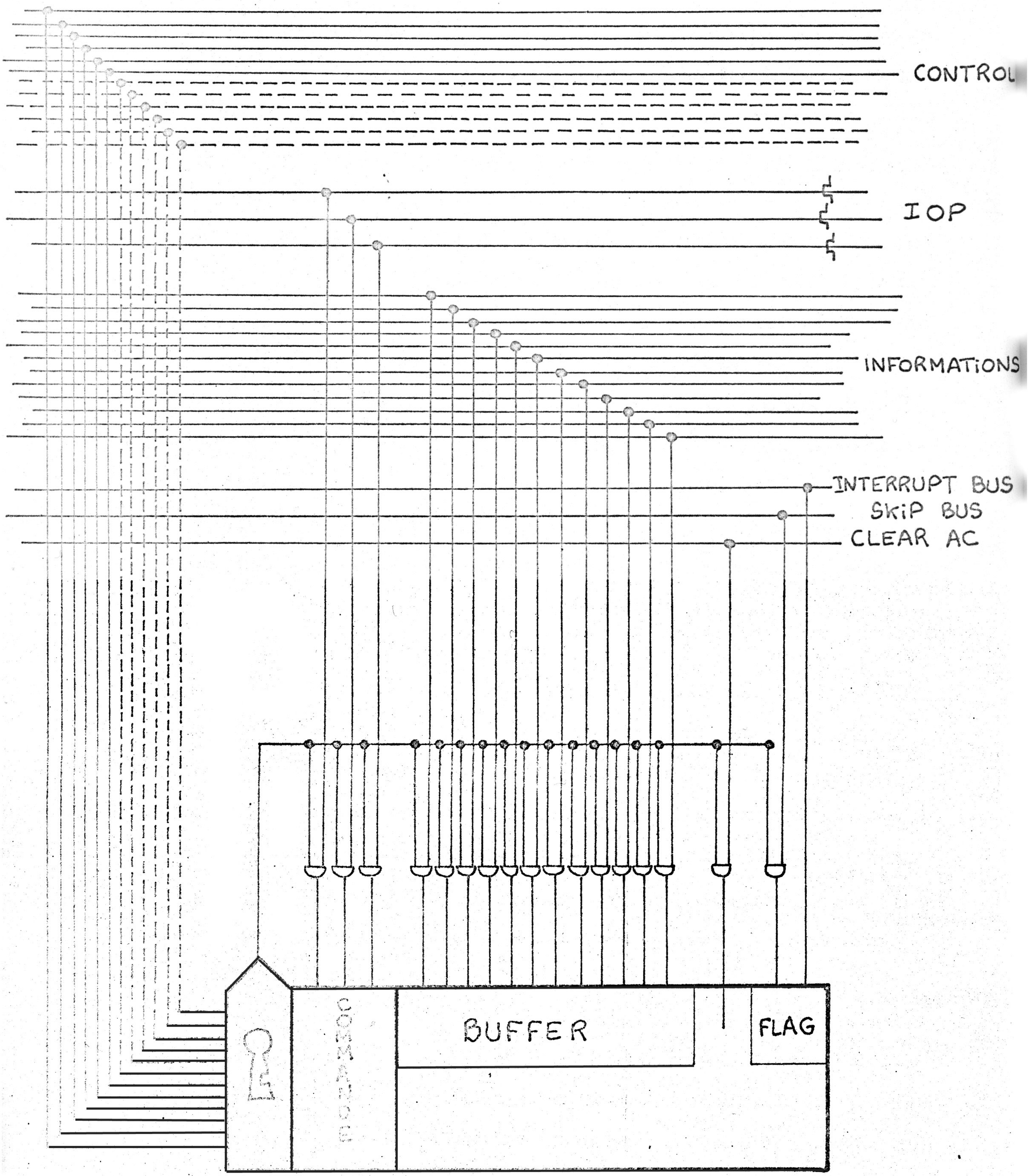
L'INTERFACE STANDARD

- Les organes de transfert-"sortie" reçoivent deux types d'information (- des signaux de données, - des signaux de contrôle).
  Les organes de transfert-"entrée" reçoivent des signaux de contrôle et envoient deux types d'information (- des signaux de données, - des signaux de contrôle).
- Electriquement, il existe deux types de signaux : - ceux qui durent un temps très court (relativement à la durée d'exécution d'une instruction) : les impulsions ; - ceux qui persistent un temps plus long : les niveaux -

SORTIE
- Les signaux de données sont en général des niveaux envoyés en parallèle sur 12 lignes. Les signaux sont reçus par tous les organes de transfert mais ne sont captés que par l'un d'entre eux en réponse à un signal de contrôle caractéristique de l'organe.
  Les signaux de contrôle sont de deux types :
  - six lignes de niveaux et leurs compléments fournissent un code caractéristique de chaque organe de transfert qui n'est capté que par l'organe de transfert intéressé et permet ainsi de sélecter un seul organe.
  - trois lignes fournissent des impulsions IOP qui permettent la synchronisation des divers travaux de l'organe de transfert. Ces impulsions ne sont captées que par l'organe de transfert qui a été sélecté auparavant.

ENTREE
- Les signaux de données sont envoyés en parallèle sur 12 lignes. Un organe de transfert ne peut envoyer ces signaux que s'il a été sélecté auparavant.
  Les signaux de contrôle sont de plusieurs types :
  - 6 lignes de niveaux et leurs compléments fournissent un signal caractéristique
  Toutes ces lignes sont sous le contrôle d'un programme écrit par le programme
  - 1 ligne d'interruption qui peut à tout moment être sollicitée par n'importe quel organe de transfert. Une demande d'interruption porte cette ligne à un niveau donné. Aucune demande d'interruption ne sera perdue.
  - 1 ligne permet de remettre l'AC à zéro. Elle n'est utilisée que par l'organe de transfert sélecté.
  - 1 ligne permet de tester le "flag" propre à l'organe de transfert dans lequel les données à transmettre sont rangées. Cette ligne ne peut être utilisée que pour l'organe de transfert sélecté.

SCHEMA DE L'iNTERFACE



DATA-BREAK

Certains organes de transfert qui travaillent à grande vitesse, unités de bandes magnétiques, tambours, etc... communiquent avec le calculateur en utilisant les possibilités de contrôle automatique des transferts appelées pour le P.D.P.8: "Data Break".

Lorsque le DB est utilisé par un organe de transfert, le processeur passera par une succession de un ou trois états différents de ceux par lesquels il passe habituellement lors de l'exécution d'une séquence d'instructions.

Pour pouvoir effectuer un transfert, il est nécessaire de connaître :

- l'adresse de la première mémoire à partir de laquelle les mots de données transférés pourront être rangés (ou pris).
- le nombre de mots que l'on veut transmettre.

Si ces deux informations sont contenues dans des registres appartenant à l'organe de transfert lui-même, le DB sera d'un cycle.

Si ces deux informations sont contenues dans deux mémoires particulières consécutives de la mémoire centrale, le DB sera de trois cycles.

Après avoir été demandé pendant l'exécution d'une instruction, le DB entrera en fonction à la fin de l'exécution de cette instruction à la place du prochain état Fetch.

DATA BREAK - 1 cycle : ("Break State").

1 cycle machine seulement est utilisé pour transférer un mot; ce DB sera utilisé pour un organe de transfert très rapide (tambour). L'adresse à laquelle on veut ranger (ou lire) le mot à transférer est chargée dans le MA à partir de l'organe de transfert - le mot est transféré par l'intermédiaire du MB.

DATA BREAK - 3 cycles : ("Word Count"; "Current Address State"; "Break State").

Ici, 3 cycles machine sont nécessaires pour le transfert d'un mot - les organes de transfert qui l'utiliseront seront un peu plus lents que précédemment (cas des bandes magnétiques).

Deux mots mémoire d'adresses fixes contiennent :

- le premier: le complément à 2 du nombre de mots à transmettre.
- le second : l'adresse du premier mot à transmettre (ou l'adresse du premier mot à partir duquel on doit ranger les données transférées).

CHAPITRE IX

UNITES d'ENTREES-SORTIES

UNITE ARITHMETIQUE & LOGIQUE

UNITE DE TRAITEMENT

UNITE MEMOIRE

PUPITRE de CONTROLE

P C

M A

L

AC

iR

MSG

MB

ORGANES DE TRANSFERT

MEMOIRE CENTRALE

4096 mots de 12 bits

SCHEMA DU P.D.P.8

FETCH

MEMORY STROBE

INT ACK | INT ACK

MB TRANSITIONS → iR

JMS → iR

OPR | iOT | AND + TAD + iS2 + DCA

MB3(0) | MB3(1) • MB11(0)

JMP + JMS

+1 → PC

JMP | JMP

MB4(1) • MB6(0)
0 → AC

MB5(1) • MB7(0)
0 → L

MB4(0) • MB6(1)
COMP → AC

MB5(0) • MB7(1)
COMP → L

MB4(1) • MB6(1)
ONES → AC

MB5(1) • MB7(1)
1 → L

MB5(1) • MB8(0) • AC0(1)
MB5(1) • MB8(1) • AC0(0)
MB6(1) • MB8(0) • AC = 0
MB6(1) • MB8(1) • AC ≠ 0
MB7(1) • MB8(0) • L = 1
MB7(1) • MB8(1) • L = 0
MB5(0) • MB6(0) • MB7(0) • MB8(1)
+2 → PC
OTHERWISE
+1 → PC

MB10(1)
0 → RUN

MB4(1)
0 → AC

+1 → PC

1 → PAUSE

MB11(1) : IOP1

MB10(1) : IOP2

MB9(1) : IOP4

T1

MB3(0) | MB3(1)

MB8(1) • MB10(0)
RAR

MB8(1) • MB10(1)
RTR

MB9(1) • MB10(0)
RAL

MB9(1) • MB10(1)
RTL

MB11(1)
+1 → AC

MB9(1)
SR → AC

0 → PAUSE

0 → MB
MB5-11 → PC

MB4(0) | MB4(1)

0 → PC 0-4

BRK RQST | BRK RQST

MB5-11 → MA

MB4(0) | MB4(1)

0 → MA 0-4

0 → iR

1 → F

BRK. RQST | BRK RQST

PC → MA

DATA ADDRESS → MA

0 → iR

1 CYCLE | 3 CYCLE

1 → B

1 → WC

MB5-11 → MA

MB4(0) | MB4(1)

0 → MA 0-4

MB3(0) | MB3(1)

JMS | JMS

0 → MB

MB5-11 → PC

PC → MB

MB4(0) | MB4(1)

0 → PC 0-4

1 → E

0 → MA

0 → PC

PC → MB

1 → E

0 → MB

1 → D

T2

DEFER

EXECUTE

MEMORY STROBE

JMS

DCA

ISZ + TAD + AND

MEMORY STROBE

AC → MB

0 → AC

+1 → PC

INT ACK INT ACK

+1 → MB

0 → INT ACK

MA0-7(0)·MA8(1)
+1 → MB

T 1

ISZ TAD AND

+1 → MB
MB0 (0)
TRANS
+1 → PC

MB HALF
ADD WITH AC

JMP JMP

MB → PC
0 → MB

MB → MA

BRK.RQST BRK RQST

MB → MA

DATA ADD → MA

1 CYCLE 3 CYCLES

0 → IR

1 → B

1 → WC

1 → F

JMS JMS

0 → MB

PC → MB
MB → PC

TAD AND

AC CARRY

MB 0 → AC

T 2

0 → MB

BRK RQST BRK RQST

PC → MA

DATA ADD → MA

1 CYCLE 3 CYCLES

0 → IR

1 → B

1 → WC

1 → F

WORD COUNT

MEMORY STROBE

+1 → MB

MB0 TRANSITION → 0
WC OVERFLOW PULSE

T1

T2

0 → MB

1 → MA-11

1 → CA

CURRENT ADDRESS

MEMORY STROBE

+1 → CA INHIBIT

+1 → CA INHIBIT

+1 → MB

0 → MB

MB → MA

1 → B

BREAK

DATA OUT

DATA IN

MEMORY STROBE

DATA → MB

0 → MB

BRK RQST

BRK RQST

PC → MA

DATA ADD → MA

1 CYCLE

3 CYCLES

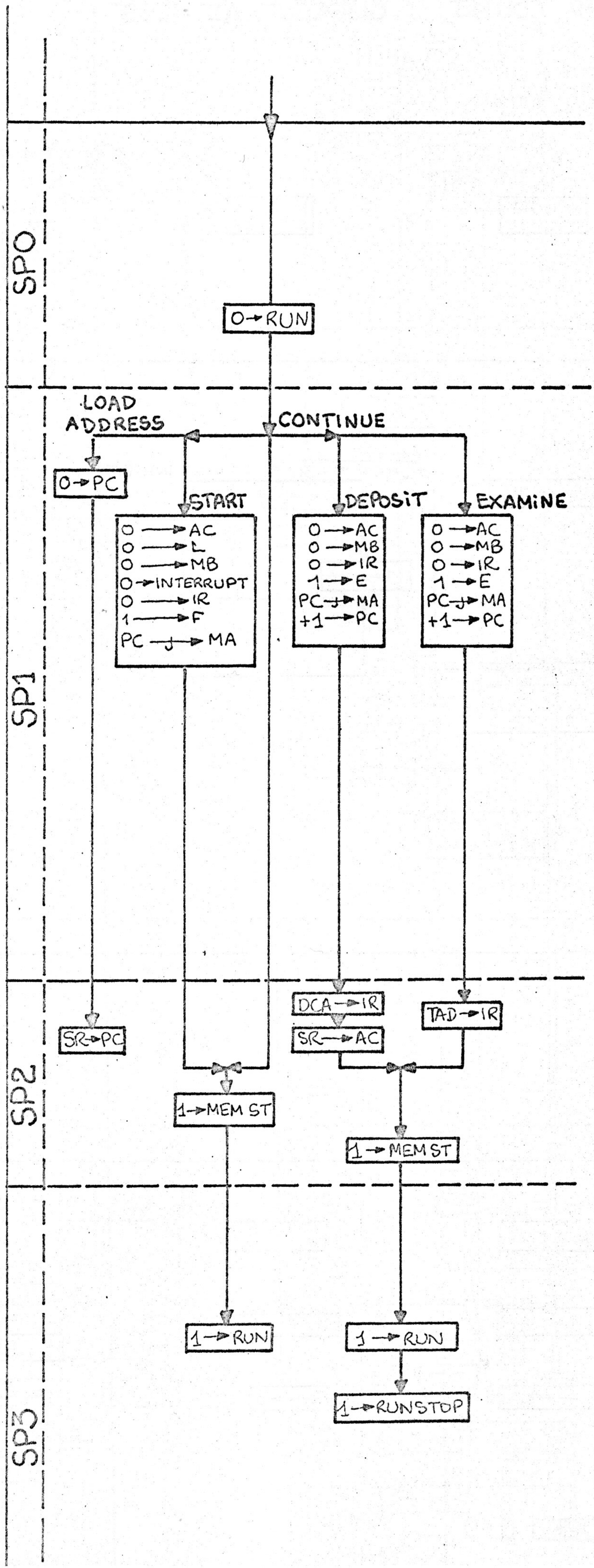
0 → IR

1 → B

1 → WC

1 → F

COMMANDES MANUELLES



S O M M A I R E