Université des sciences et de la technologie d'Alger

Département d'Electronique et d'Electrotechnique

Filiere d'Ingénieur en Electronique

PROJET DE FIN D'ETUDES

APPLICATIONS AU MICROPROCESSEUR

de la Carte TM 990/189

proposé par: M. Sureshchander

étudié par: M.T. Fradjia
E.H. Atmane

JUIN 1981

Université des sciences et de la technologie d'Alger

Département d'Electronique et d'Electrotechnique

Filiere d'Ingénieur en Electronique

PROJET DE FIN D'ETUDES

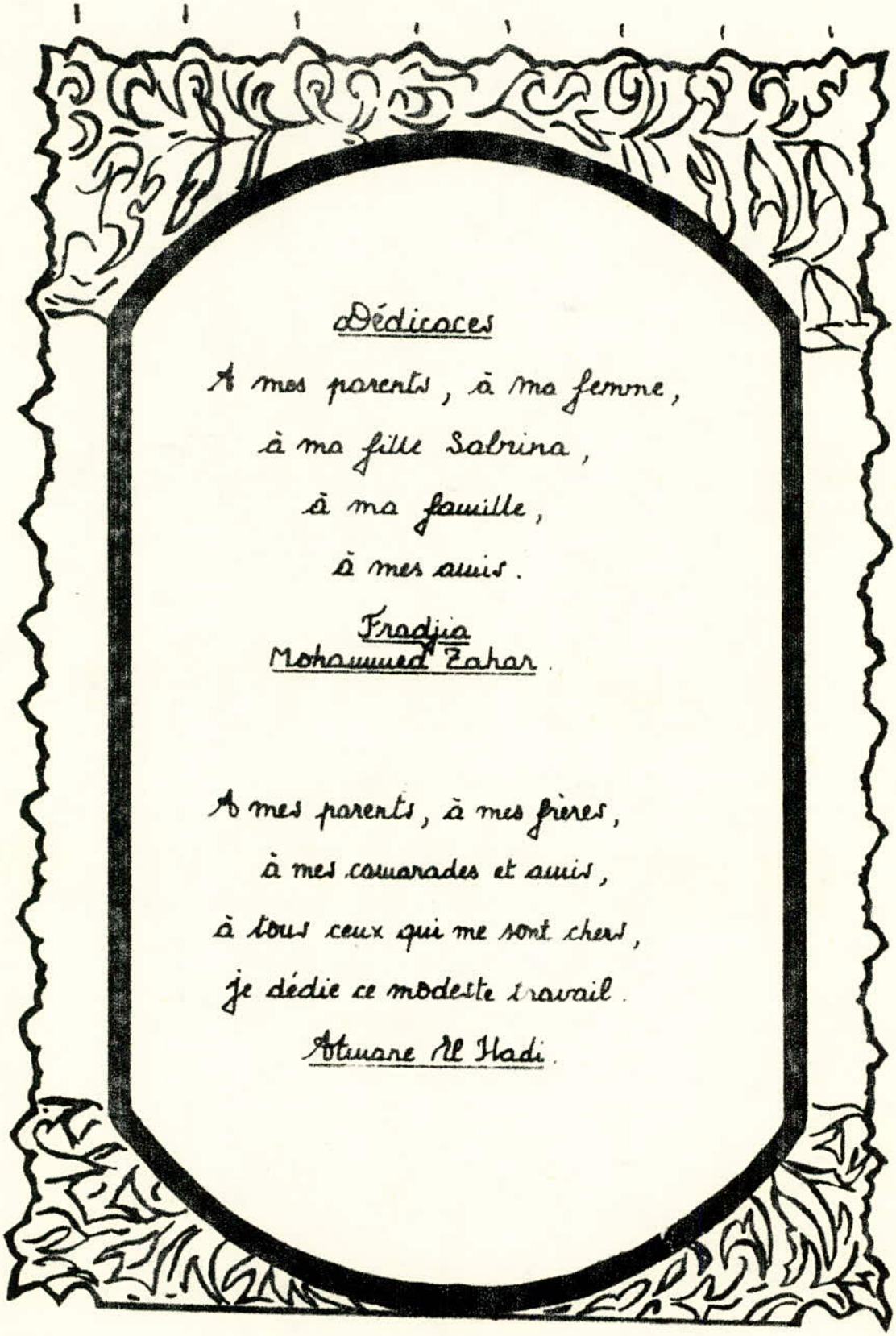
APPLICATIONS AU MICROPROCESSEUR

de la Carte TM 990/189

proposé par: M. Sureshchander

étudié par: M.T. Fradjia
E.H. Atmane

JUIN 1981



Dédicaces

A mes parents, à ma femme,
à ma fille Sabrina,
à ma famille,
à mes amis.

Fradjia
Mohammed Zahar.

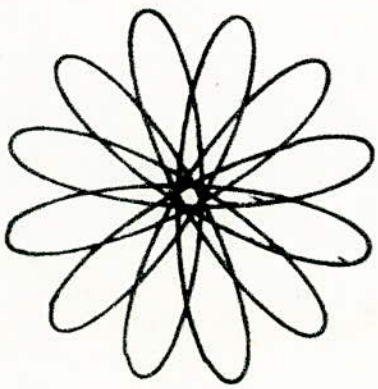
A mes parents, à mes frères,
à mes camarades et amis,
à tous ceux qui me sont chers,
je dédie ce modeste travail.

Attmane El Hadi.

REMERCIEMENTS

Nous tenons à remercier tous ceux qui, de près ou de loin, ont contribué à notre formation.

Nos remerciements vont plus particulièrement à notre promoteur, Mr. SURESHCHANDER qui, d'une part nous a proposé ce sujet, et d'autre part nous a suivi tout au long de sa préparation.



SOMMAIRE

Introduction

L'automatisation de certains travaux, conduit à utiliser des micro-processeurs. Le but essentiel est d'avoir un système simple, de gestion d'entrées-sorties aussi bien pour effectuer des échanges d'informations avec des périphériques qu'avec des appareillages électroniques classiques. L'arrivée des microprocesseurs a été une grande révolution dans la technique de l'automatisme, rendant bien souvent inutiles les anciennes méthodes d'études (notons que tout système en logique cablée peut être remplacé par son homologue en logique programmée).

L'application des micro-processeurs s'étend chaque jour, l'utilisation de la famille des TMS 9900 en est un exemple; grâce à sa haute performance, cette famille est capable de piloter des applications très diverses :

- Alarme
- Horloge
- Chauffage
- enregistrement sur magnétophone
- exécution d'une partition de musique

etc ...

Dans ce sujet nous allons étudier l'application d'1 type de micro-processeur de la famille TMS 9900, le TMS 9980A, utilisé dans la table TM 990/189

CHAPITRE I

Initiation à la famille 9900

La famille 9900 est une série compatible de microprocesseurs et de micro-ordinateurs, soutenue par des dispositifs périphériques et des systèmes de développement, et un software. Grâce à des dispositifs d'interface, cette famille couvre un domaine d'application très large. Elle est établie avec une unique architecture flexible permettant une adaptation technologique facilement incorporable quand il s'agit de réduire l'encombrement.

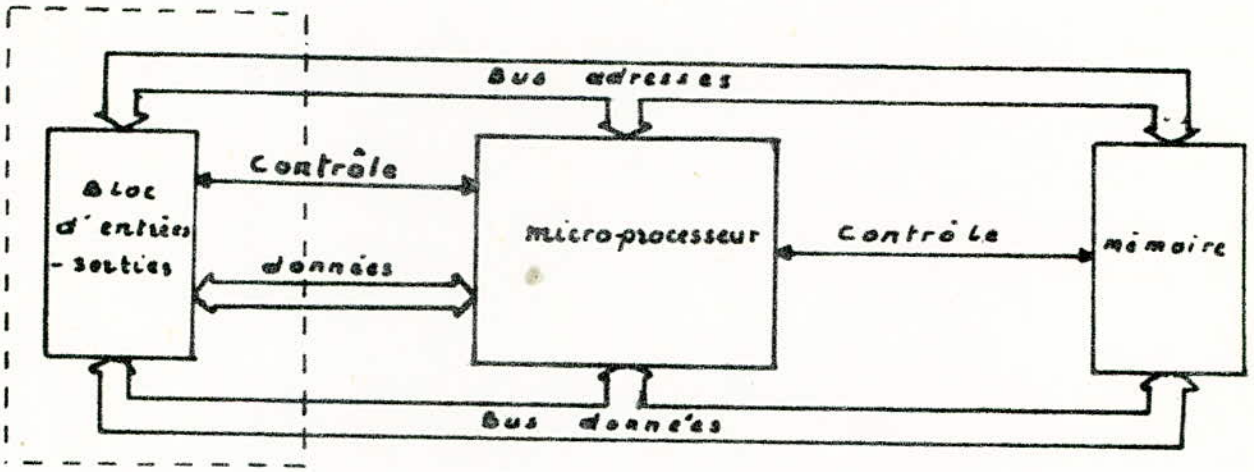
1. Le CPU : (Unité centrale de traitement) :

Le CPU est la partie d'un système informatique qui contient l'unité arithmétique, un groupe de registres spécialisés et les principaux éléments de stockage. Le tableau suivant fournit les caractéristiques de base de certains CPU de la famille TMS 9900.

µ-processeurs / caractéristiques	SN 545 481 SN 745 481	SBP 9900A	TMS 9900/ TMS 9900-40	TMS 9980 A TMS 9981	TMS 9985
Nbre d'octets adressables	65 K	65 K	65 K	16 K	65 K
Nbre d'interruptions	16	16	16	5	5
Nbre de blocks	48	64	64	40	40
Alimentation	+5	Résistance programmable	+12, ±5	+12, ±5	+5
Technologie	Schott Ky TTL	I^2L	N MOS	N MOS	N MOS
environnement (Température °C)	-55 à 125	-55 à 125	0 à 70	0 à 70	0 à 70
fréquence d'Horloge	10 MHz	3 MHz	3,3 MHz / 4 MHz	10 MHz	5 MHz
Nbre de lignes de bus adresses	15	15	15	14	16
Nbre de lignes de bus donnée	16	16	16	8	8

2. Bloc entrée-sortie :

Le bloc d'entrées-sorties se comporte comme un intermédiaire entre le processeur et le monde extérieur, permettant de se communiquer et de se transmettre leur réaction. Son utilisation permet une application très variée fonctionnant avec le monde extérieur. La figure suivante montre l'implantation du bloc d'entrées-sorties dans un système micro-processeur; on distingue 3 parties : le processeur, la mémoire et le bloc d'entrées-sorties.



2.1 Catégories d'entrées-sorties :

Le bloc d'entrées-sorties peut fonctionner selon 3 façons différentes :

2.1.a Entrées-sorties controllées par programme

Dans ce cas, le programme ordonne au processeur d'émettre l'adresse d'un périphérique sur le bus adresse, puis un signal de contrôle vers l'interface d'entrées-sorties pour prévenir le bloc d'entrées-sorties qu'on veut communiquer avec un périphérique. L'interface décode l'adresse pour savoir quel est le périphérique choisi et le prévenir.

2.1.b Entrées-sorties pilotées par interruption.

Dans le cas précédent, on ne pouvait émettre un caractère que si le programme lui en demandait un ; par contre dans ce cas, on peut interrompre à tout moment le programme qui est en train de se dérouler dès qu'on appuie sur une touche.

2.1.c Entrées-sorties par accès direct à la mémoire

Dans ce 3e cas, le processeur n'intervient plus dans l'échange des données entre la mémoire et les périphériques d'entrées-sorties ; l'existence d'un contrôleur DMA qui est conçu spécialement pour l'échange des données permet un transfert de données à très haute vitesse.

2-2. Options utilisant le bus de données :

La famille 9900 permet de traiter les entrées-sorties selon 3 méthodes :

- entrelaçage avec la mémoire.
- DMA
- CRU

Dans les 2 premières méthodes, l'information est transmise en parallèle (plusieurs bits à la fois), cela peut présenter certains inconvénients, ce qui nécessite la réalisation sur le bus données d'un démultiplexage des données à destination de la mémoire et à destination de périphérique ; d'autre part, les adresses réservées aux périphériques ne sont plus utilisables par la mémoire, ce qui réduit la capacité du système. L'utilisation du CRU (Entrées-sorties séparées) permet de résoudre les problèmes posés par les 2 méthodes précédentes.

Le CRU comprend une logique interne à base de registre de décalage, assure les entrées-sorties sur 3 broches du micro-processeur utilisé : CRUIN, CRUOUT et CRUCLK (fig. 1.2), utilise des composants tout prêts, supprime le démultiplexage des données et permet un adressage individuel du bit.

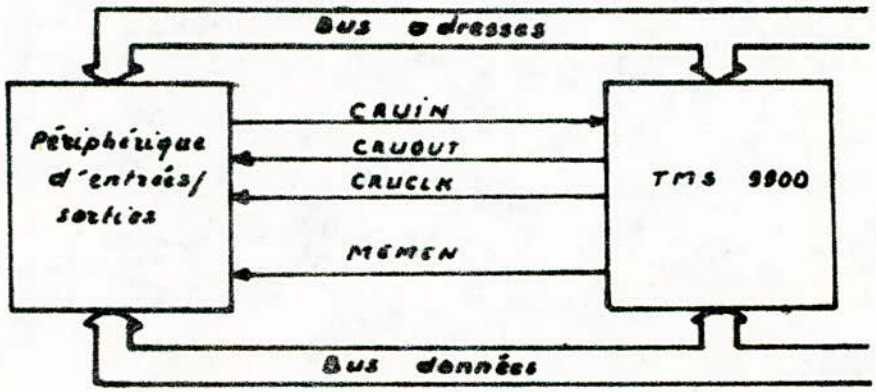


Fig. 1.2

CRUIN : Sert à introduire des données en série (1 ou plusieurs bits)
CRUOUT : Sert à émettre un ou plusieurs bits en série
CRUCLK : Sert à émettre des signaux de validation et d'échantillonnage pour piloter l'échange des données.

Le DMA présente une importance particulière lors d'échanges des données à très grande vitesse (1 million d'octets par seconde) [lecteurs de bandes, magnétophones, disques ...], citons comme exemple le TMS 9911 qui peut assurer cette tâche.

3. Partie mémoire :

La partie mémoire contient l'espace nécessaire accessible en lecture et en écriture pour un stockage de résultats intermédiaires ou des données temporaires qui seront communiqués aux périphériques, ainsi on peut lire et ranger en mémoire des données à partir d'un périphérique.

Une mémoire fonctionne sous le contrôle des signaux en provenance du CPU (ou du contrôleur DMA), ces fonctions principales sont :

* une mémorisation des instructions
* mémorisation et fourniture des données en provenance ou à destination du CPU.

4. Interruptions :

La famille 9900 possède la propriété d'abandonner momentanément une tâche pour aller en exécuter une autre, et de sauvegarder toutes les informations nécessaires pour reprendre ultérieurement la 1ère tâche dans les mêmes conditions qu'à l'instant où elle était interrompue.

5. Les périphériques :

Ce sont des dispositifs externes au microprocesseur, fonctionnant en liaison avec le CPU et permettant l'échange des données avec le monde extérieur, exemples de périphériques :

* haut parleur
* Lecteur-enregistreur
* terminal extérieur

etc ....

6. Les interfaces :

L'interface est une logique située à la frontière entre le processeur et le monde extérieur destinée à réaliser l'adaptation entre le système central et ses périphériques.

La gestion d'entrées-sorties s'effectue selon 2 techniques (parallèle ou série).

7. Jeux d'instructions :

Il est difficile de faire une comparaison entre les jeux d'instruction des micro-processeurs de familles différentes ; dans le tableau suivant, on en compare 3 :

Instructions \ Micro-processeurs	8080	6800	9900
Nbre d'instructions données par le constructeur	78	72	69
Instructions distinctes	27	26	36
Nbre de combinaisons max	237	169	62.235

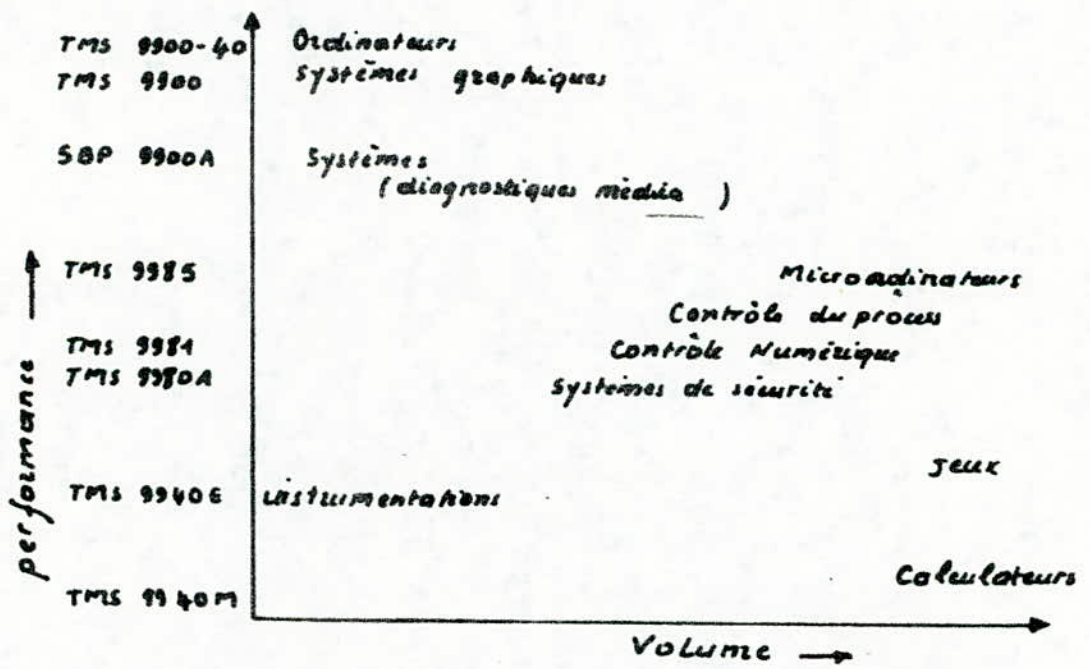
La 1ère rangée représente le nombre d'instructions données par le constructeur, la 2ème représente le nombre d'instructions distinctes dans ce jeu et la 3ème donne le nombre maximum de combinaisons possibles de chaque famille ; Ce dernier nombre est dérivé du fait que certaines instructions laissent plusieurs bits non spécifiés pour prévenir une variété de modes d'adressage, par exemple dans le TMS 9900, 12 instructions (Addition, soustraction, ...) laissent 12 bits non spécifiés, ce qui donne 2^{12} (4096) variations, alors que les jeux d'instructions d'octet des MC 6800 ne possèdent pas ce degré de flexibilité.

8. Applications :

Deux systèmes inondent actuellement le marché ;

* le système à tâche unique, caractérisé par une basse performance et un grand volume.
* le système à multi-tâches, caractérisé par une haute performance et un volume bas.

Le schéma suivant montre le domaine d'application en fonction du volume et de la performance.



Types d'applications de la famille 9900 :

Simple tâche

* Système d'alarme
* automatisme
* Jeux
* Processeur de contrôle
* équipements de navigation

multi-tâches

* Contrôleur vidéo
* instruments électroniques
* équipement médical
* Système de sécurité
* Contrôle des machines

9. Apperçu sur la famille 9900 :

a) Microprocesseurs utilisés comme CPU :

TMS 9900	NMOS µp à 16 bits, 64 broches
TMS 9900-40	Version de 9900 haute fréquence
SBP 9900-A	I^2L Domaine de température étendu
TMS 9980A 9981	40 broches, NMOS, µp à 16 bits avec 8 bits de bus données
TMS 9940 M	40 broches, NMOS, µ-ordinateur à simple tâche
TMS 9985	40 broches, NMOS, µp à 16 bits avec une unique alimentation de 5V et 256 bits de mémoire RAM
TMS 9940 E	40 broches, NMOS, version EPROM µ-ordinateur à simple tâche

b) Microprocesseurs utilisés comme dispositifs périphériques :

TMS 9901	Programmable System interface
TMS 9901-40	Higher frequency version of 9901
TMS 9902	Asynchronous communications controller
TMS 9902-40	Higher frequency version of 9902
TMS 9903	Synchronous communications controller
TMS 9904	4-phase clock driver
TMS 9905	8 to 1 Multiplexer

TMS 9906	3 bits Latch
TMS 9907	8 to 3 Priority Encoder
TMS 9908	8 to 3 Priority Encoder
TMS 9909	Floppy Disk Controller w/ tri-state outputs
TMS 9911	Direct Memory Access Controller
TMS 9914	GPIB Adaptateur
TMS 9915	Dynamic RAM Controller Chip Set
TMS 9916	92 K Magnetic Bubble Memory Controller
TMS 9922	250K Magnetic Bubble Controller
TMS 9923	250K Magnetic Bubble Controller
TMS 9927	Video Timer / Controller
TMS 9932	Combination ROM / RAM Memory
TMS 9960	I/O Expander
TMS 9961	Interrupt-Controller / Timer
SBP 9964	SBP 9900A Timing génération
SBP 9965	Peripheral interface Adapter

* TMS 9901 utilisé dans la carte TM990/189
* TMS 9902 permet la gestion des entrées-sorties parallèles asynchrones
* TMS 9903 permet la gestion des entrées-sorties parallèles synchrones.
* TMS 9904 Horloge à 4 phases fournie avec des µ-processeurs de la famille 9900 ou autres ;
L'oscillateur interne du TMS 9904 peut être contrôlé par une base de temps externe d'un circuit oscillat ou autre.
* TMS 9909 Le TMS 9909 facilite l'interfaçage des disques souples qui sont en train de remplacer rapidement

les cassettes pour des raisons de coût, de performance et de fiabilité. Le micro-processeur permet par exemple de lire ou écrire des données sur une disquette.

* TMS 9911 (DMAC) : Direct Memory Access Controller
C'est un micro-processeur d'intégration à haute densité [LSI ; Large Scale Integration], générant les signaux de contrôle des mémoires et la séquence des adresses mémoires pour des voies indépendantes du DMA

* TMS 9927 (VTC) : Vidéo Timer Controller
Ce micro-processeur contient toute la logique nécessaire pour générer les signaux du timing ; il possède des registres de contrôle à 8 bits utilisés pour la programmation, des registres pour les balayages horizontaux et verticaux et des registres pour un adressage rapide

10. Avantages :

L'utilisation des micro-processeurs de la famille 9900 apporte un grand avantage ; cet avantage provient du fait qu'ils travaillent sur 16 bits, possèdent des opérations internes [division, soustraction, multiplication, décalage, etc...] et permettent un changement de contexte souple et rapide.

Grâce à leur compatibilité technologique complète du point de vue logiciel ou matériel et leur haute performance à un prix bas, cette famille couvre un champ d'application très large.

CHAPITRE II

A. Le TMS 9980A

1. Description :

Le TMS 9980A est un micro-processeur dont le jeux d'instructions est compatible avec ceux des micro-processeurs et micro-ordinateurs de cette famille ; ses caractéristiques sont choisies de manière à minimiser le prix de revient des systèmes relativement petits, son CPU travaille sur 16 bits et possède une horloge intégrée dans le boitier et son bus de données a une capacité de 8 bits. L'architecture (mémoire à mémoire) caractérisant la famille 9900 permet de définir en mémoire plusieurs bancs de registres de travail, ce qui donne plus de souplesse à la programmation et accélère la prise en compte des interruptions.

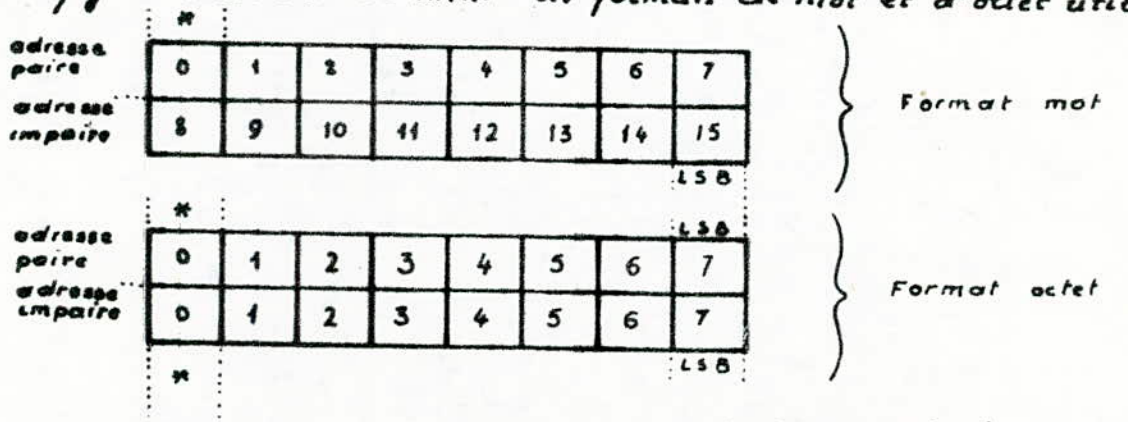
Caractéristiques principales du TMS 9980A :

- mono-boitier 40 broches
- instructions mot (16 bits)
- Capacité de bus de données 8 bits
- capacité mémoire extensible jusqu'à 16384 octets (16K)
- Architecture évoluée mémoire à mémoire.
- 16 registres de travail
- 4 niveaux d'interruptions hiérarchisés
- Générateur d'horloge à 4-phases intégré dans le boitier
- bit d'entrées-sorties adressable individuellement à l'aide du CRU

2. Architecture :

Les mémoires du TMS 9980A sont adressables en tant qu'octet de 8 bits un mot est un groupe de 16 bits (2 octets consécutifs) situé en mémoire d'adresse paire. L'octet le plus significatif d'un mot est situé à une adresse paire, le moins significatif à l'adresse impaire suivante.

Les figures suivantes donnent les formats de mot et d'octet utilisés :



* : MSB (Bit de poids fort)
ou bit de signe
LSB (Bit de poids faible)

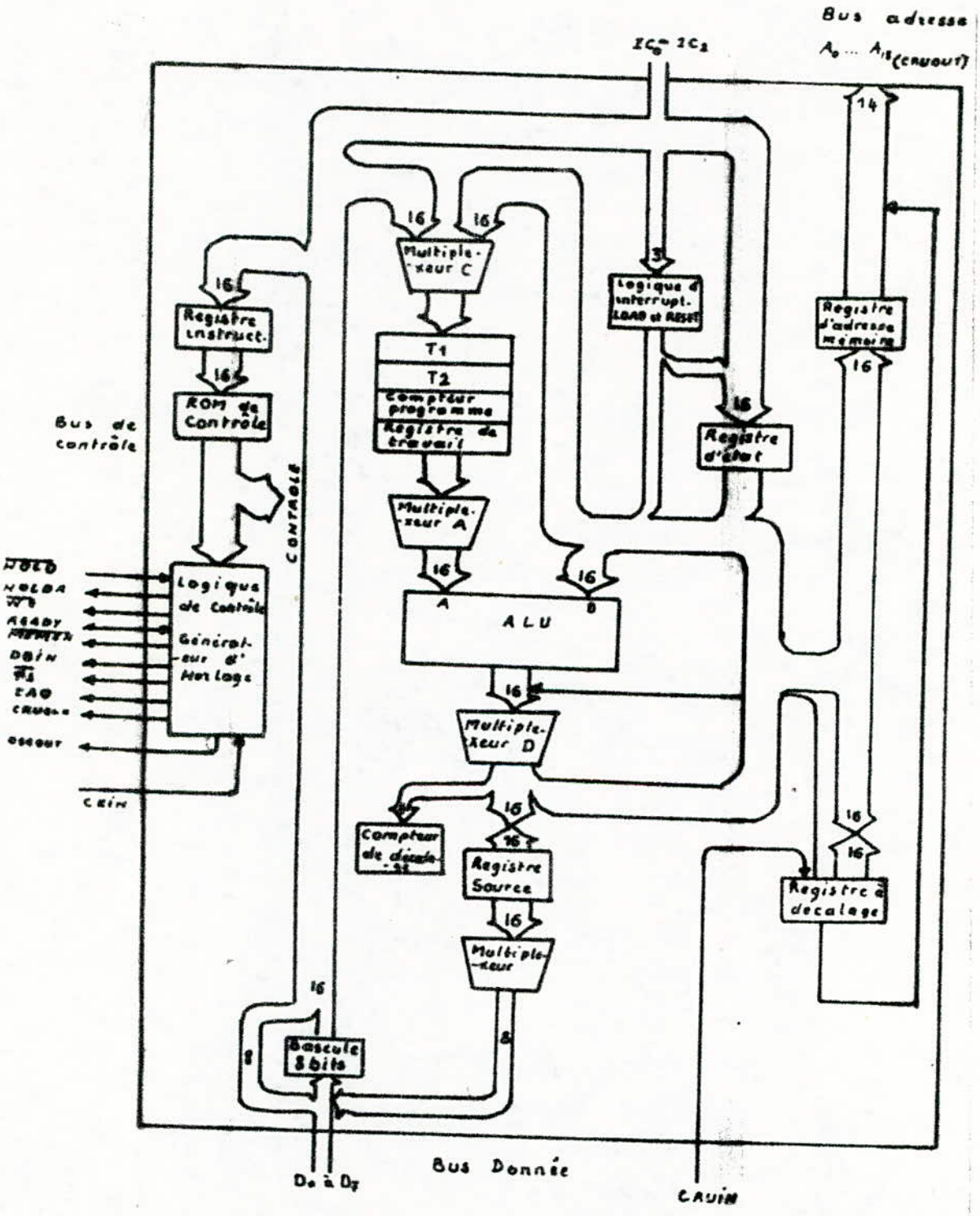


Fig 2 : ARCHITECTURE DU TMS 9980A

* Registre d'adresse mémoire :

L'adressage mémoire est limité par 14 bits (A_0 à A_{13}) d'où la capacité de sortie est de $2^{14} = 16384$ octets.

* Registres à décalage :

Ces registres permettent un décalage [à droite, à gauche ou Circulaire) de 1 à 16 bits.

* Registre d'état :

La configuration des bits du registre d'état dépend des résultats de la dernière opération effectuée par l'ALU ; elle permet ainsi des comparaisons arithmétiques et logiques.

* L'ALU (Unité arithmétique et Logique)

Cette unité peut effectuer des opérations arithmétiques ou logiques telles que : l'addition, le décalage, la comparaison, la remise à zéro...

* Compteur programme :

Ce registre contient l'adresse de la prochaine instruction à exécuter.

* Pointeur d'espace de travail :

C'est un registre modifiable par programme et servant de pointeur vers le début d'un bloc de 16 mémoires appartenant à un espace défini dans la RAM, cela implique que les registres de travail n'appartiennent plus au CPU, mais ils se trouvent en mémoire [fig. 2-1]. L'avantage principal de ce procédé est que le programmeur peut se définir plusieurs blocs de registres (espaces de travail) pour résoudre un problème ardu de programmation simplement en réinitialisant son espace de travail chaque fois que son programme demande un changement de contexte.

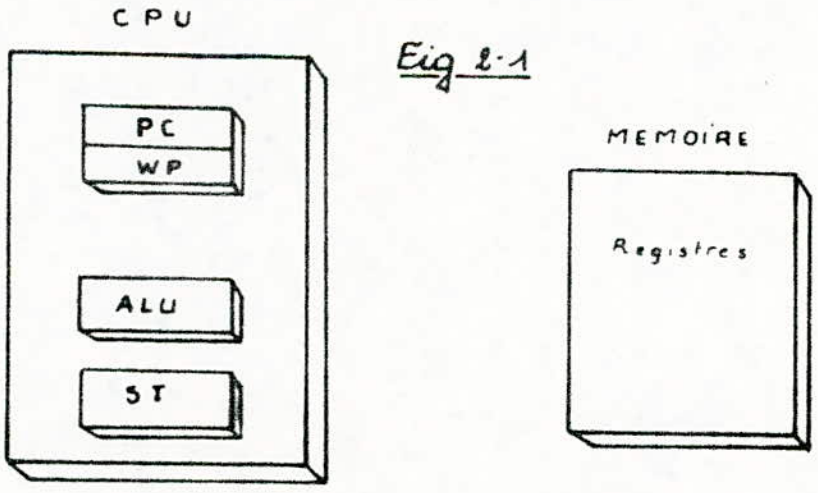


Fig 2-1

* Registre d'instructions :

Ce registre conserve l'instruction pendant que le CPU l'exécute ; la logique de décodage de CPU analyse les différents champs de l'instruction pour déterminer les opérations à exécuter.

* Les interruptions :

Le TMS 9980A reconnait 6 niveaux d'interruptions hiérarchisés vectorisés, dont le niveau 0 est le plus prioritaire ; Ces interruptions servent à un changement de contexte en sauvegardant toutes les informations nécessaires pour reprendre à nouveau le programme interrompu. Si une interruption se présente, et elle est permise par le masque contenu dans le registre d'état, le processeur répond de la façon illustrée par l'organigramme [fig 2.2].

* Décodage des Interruptions:

La voie d'I/O utilisateur (U_{IO}) et le generateur de type LOAD assurent l'encodage de niveau de priorité correspondant au signal d'interruption ; le code de ce signal d'interruption selectionnée est envoyé sur 4 bits $[IC_0, \ldots, IC_3]$ associés à un signal de demande d'Interruption sur la ligne INTREQ, mais le TMS 9980A n'en accepte que 3.

Code d'Interruption INT_0 ... INT_2	Fonction	adresse du vecteur
0 0 0	RESET	0 0 0 0
0 0 1	RESET	0 0 0 0
0 1 0	LOAD	3 F F F
0 1 1	INTERRUPT 1	0 0 0 4
1 0 0	= 2	0 0 0 B
1 0 1	= 3	0 0 0 C
1 1 0	= 4	0 0 1 0
1 1 1	—	

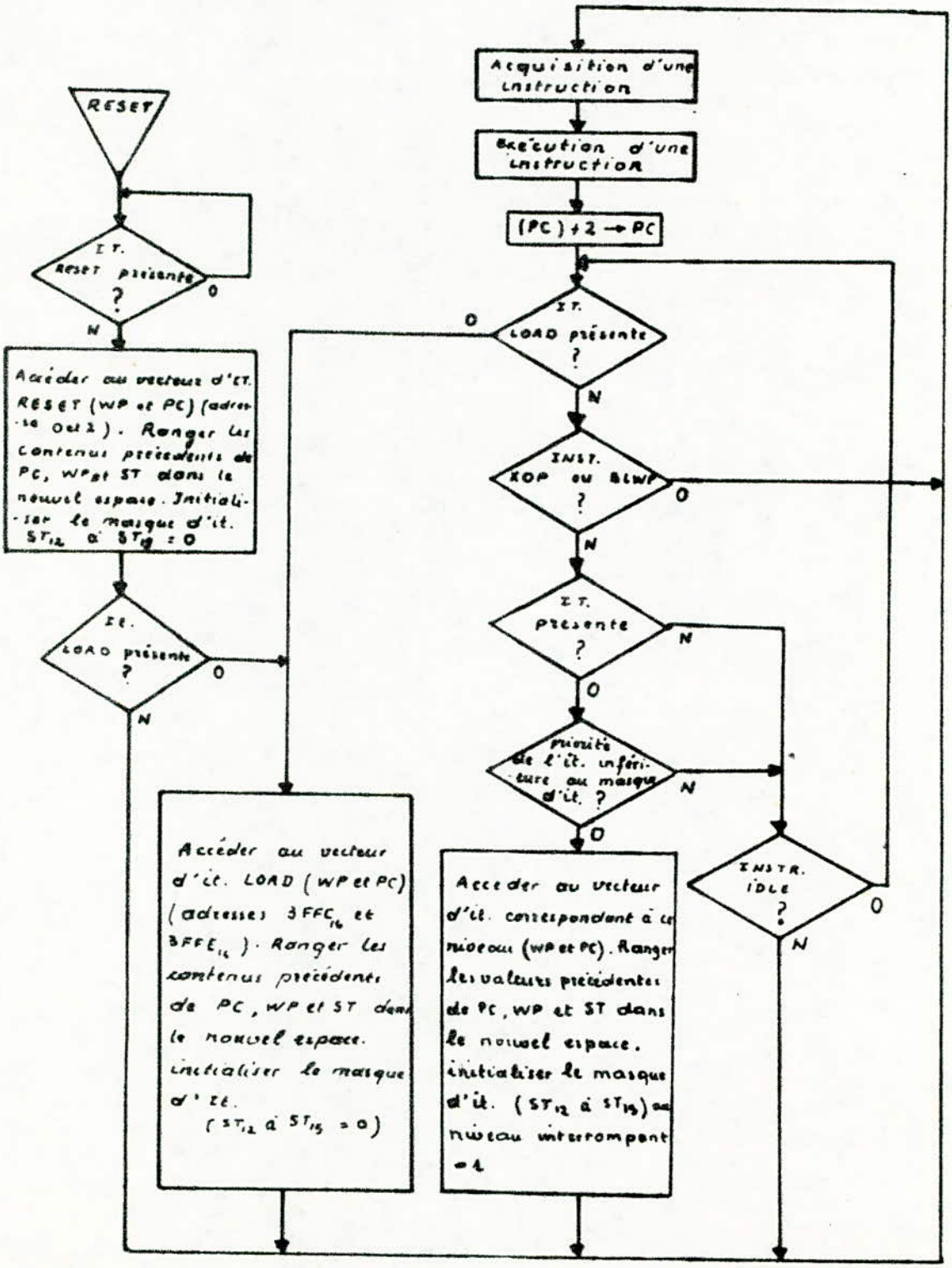
* cheminement du signal d'Interruptions :

Le cheminement des demandes d'interruptions causées par les entrées/sorties du Système ainsi que les demandes externes à travers P_5 arrivant sur U_{IO}, est illustré par le schéma [fig 2.3] U_{IO} a pour rôle de générer les codes d'Interruptions correspondants.

* Répartition-mémoires :

Le TMS 9980A permet un adressage sur 14 bits, un exemple d'organisation de la mémoire adressable par ce processeur utilisé dans la table TM 990/189 est donnée par la figure 2-4.

Fig 2.2 : Organigramme de l'unité centrale TMS 9980 A



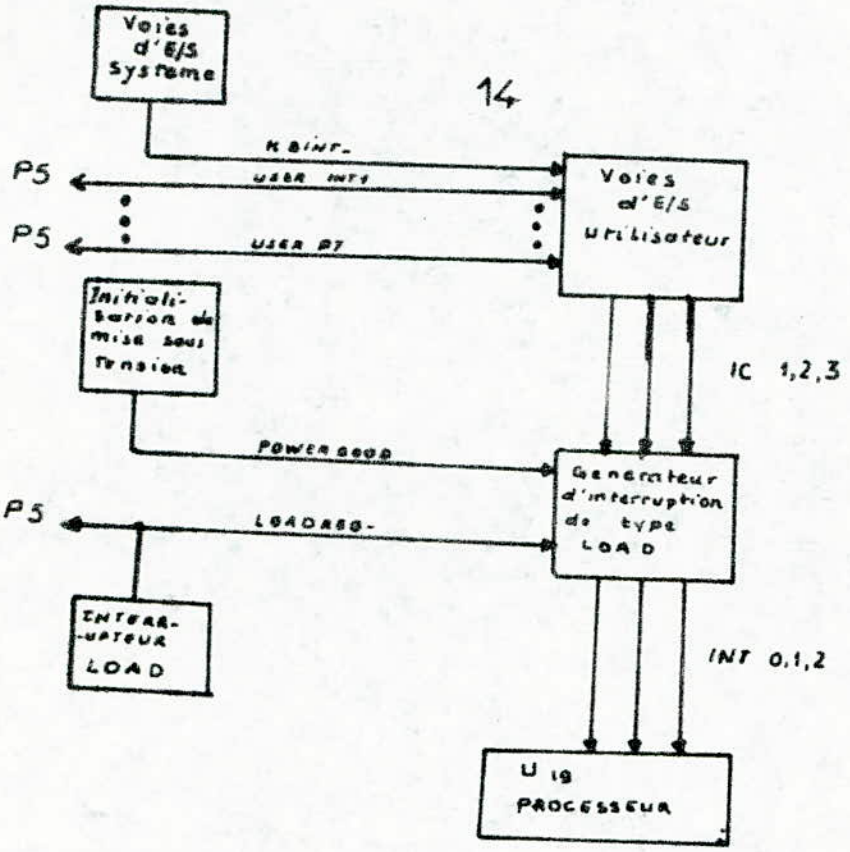


Fig 2-3 : CHEMINEMENT D'UN SIGNAL D'INTERRUPTION

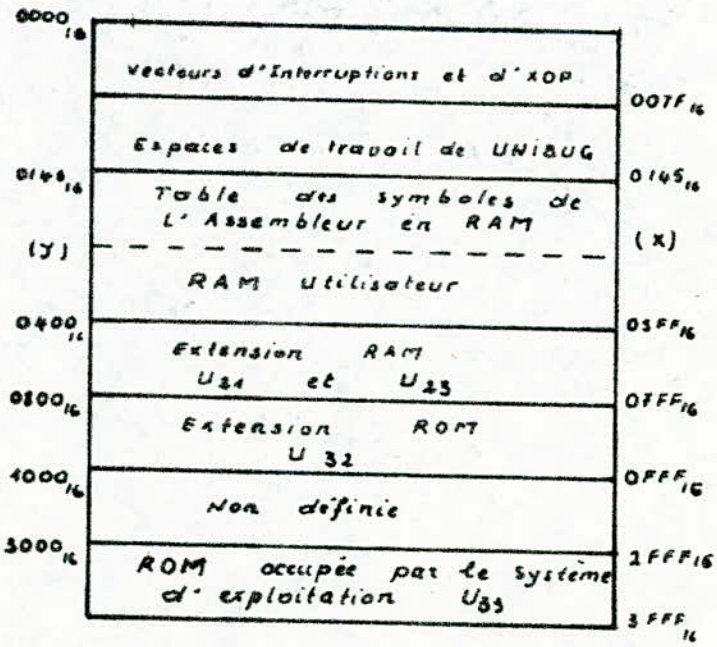


Fig 2-4 : Répartition de la mémoire de la carte TM 990/189.

B. Le TMS 9901

1. Introduction et présentation :

Le TMS 9901 est un circuit d'interface compatible avec les micro-processeurs de la famille 9900 pouvant servir d'un générateur d'horloge et hiérarchiser les interruptions d'un système micro-processeur basé sur le CRU.

Selon le schéma [2.5], le TMS 9901 Comprend 4 parties :

a. Interface unité centrale:

L'échange des données entre le CPU et le TMS 9901 peut utiliser 1 à 16 bits à l'aide d'une seule instruction CRU, cet échange s'effectue en serie, permettant ainsi une minimisation des interconnections.

b. Interface Système :

Le TMS 9901 permet de générer 22 lignes d'entrées-sorties réparties en 3 groupes :

- Groupe 1: Comprend 6 lignes prises individuellement comme entrée ou interruption seulement.
- Groupe 2. Comprend 9 lignes prises individuellement comme inter-ruption ou entrées-sorties
- Groupe 3 Comprend 7 lignes prises individuellement comme entrées ou sorties seulement.

c. Logique d'interruption:

Le TMS 9901 reconnait 15 interruptions, chaqu'une d'elles peut être masquée ou non par programme indépendemment des relations des niveaux de priorité.

d. Horloge temps réel :

L'Horloge temps réel est un compteur de 14 bits, se décrémentant automatiquement à une fréquence fixe à partir de la valeur programmée. Le passage à zéro de ce compteur provoque une demande d'interruption sur $\overline{INT_2}$.

A. Interface unité centrale :

A1: Interface CRU :

L'échange des données entre le CPU et le TMS 9901 se fait à travers le CRU à partir des 3 lignes [CRUIN, CRUCLK et CRUOUT]. Les 32 bits CRU sont contrôlés par les lignes d'adresses (S_0 à S_4)

A2: Description des bits CRU.

Le TMS 9901 a en entrée 5 lignes d'adresses (S_0 à S_4) permettant un décodage interne de 32 bits. Sur la figure [2-6], on a représenté la correspondance entre l'équivalent décimal de l'adresse S_0 à S_4 et la signification du signal concerné ainsi que le bit associé pour les 2 circuits TMS 9901 (U_{10} et U_{11}) utilisés dans la carte (TM 990/189).

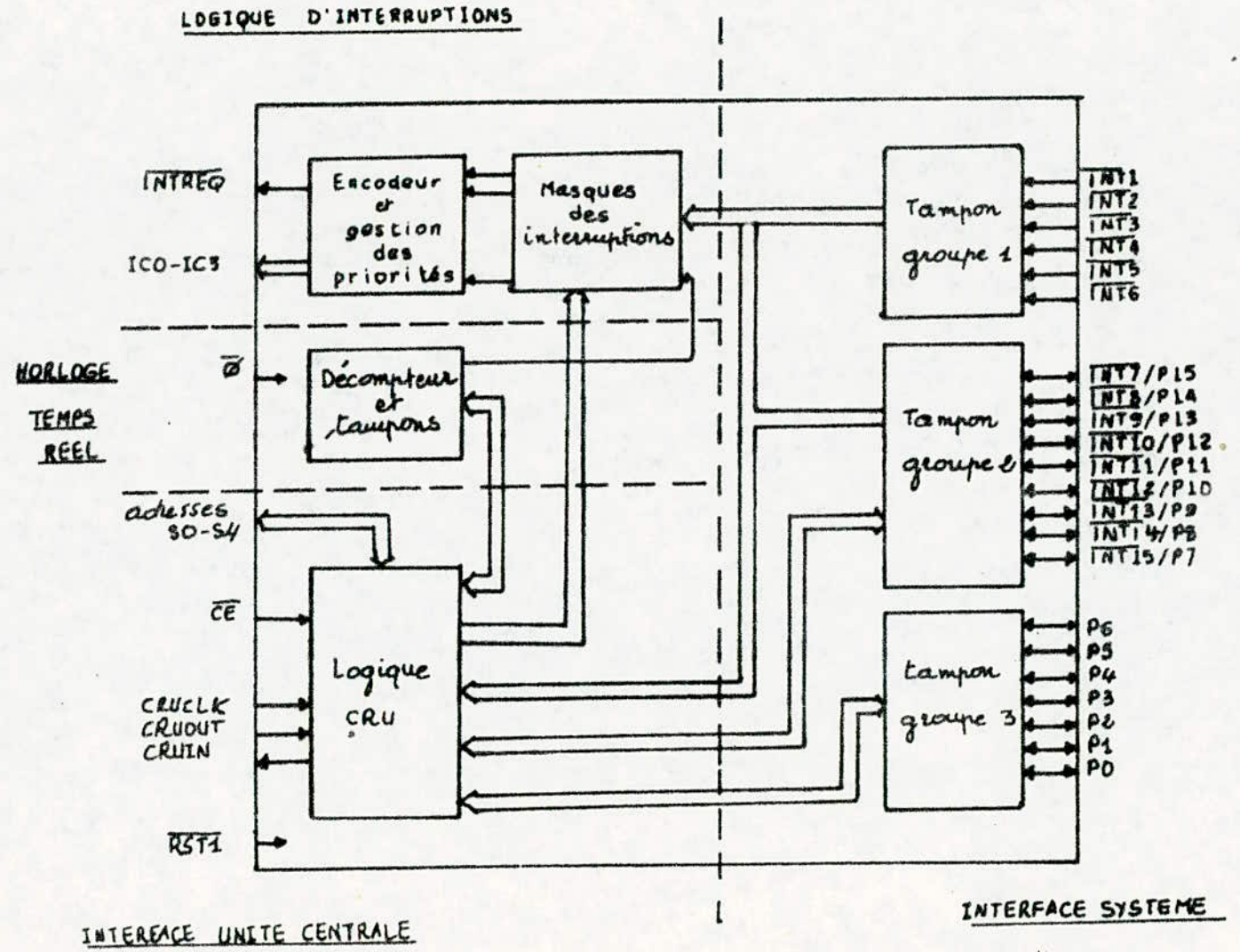


Fig 2-5 SCHEMA BLOC DU TMS 9901

Fig 1.6 Organisation des voies d'entrées-sorties

Utilisateur [U10] Système [U11]

		Repérage des bits du TMS 9901			Identification du Signal concerné	
U_{10}	U_{11}		Entrée	Sortie	U_{10}	U_{11}
000	400	0	Control bit	Control bit		
002	402	1	INT1-, CLK 1A	MASK 1, CLK 1B	UINT 1-	KB1-
004	404	2	INT2-, CLK 2A	MASK 2, CLK 2B	UINT 2-	KB2-
006	406	3	INT3-, CLK 3A	MASK 3, CLK 3B	UINT 3-	KB3-
008	408	4	INT4-, CLK 4A	MASK 4, CLK 4B	UINT 4-	KB4-
00A	40A	5	INT5-, CLK 5A	MASK 5, CLK 5B	UINT 5-	KB5-
00C	40C	6	INT6-, CLK 6A	MASK 6, CLK 6B	KBINT-	RDATA
00E	40E	7	INT7-, CLK 7A	MASK 7, CLK 7B	USER P 15	
010	410	8	INT8-, CLK 8A	MASK 8, CLK 8B	USER P 14	
012	412	9	INT9-, CLK 9A	MASK 9, CLK 9B	USER P 13	
014	414	10	INT10-, CLK 10A	MASK 10, CLK 10B	USER P 12	
016	416	11	INT11-, CLK 11A	MASK 11, CLK 11B	USER P 11	
018	418	12	INT12-, CLK 12A	MASK 12, CLK 12B	USER P 10	
01A	41A	13	INT13-, CLK 13A	MASK 13, CLK 13B	USER P 9	
01C	41C	14	INT14-, CLK 14A	MASK 14, CLK 14B	USER P 8	
01E	41E	15	INT15-, INTREQ-	MASK 15, RST2-	USER P 7	
020	420	16	P 0 INPUT	P 0 OUTPUT	USER P 0	DIGITSEL A
022	422	17	P 1 INPUT	P 1 OUTPUT	USER P 1	DIGITSEL B
024	424	18	P 2 INPUT	P 2 OUTPUT	USER P 2	DIGITSEL C
026	426	19	P 3 INPUT	P 3 OUTPUT	USER P 3	DIGITSEL D
028	428	20	P 4 INPUT	P 4 OUTPUT	USER P 4	SEGMENT A-
02A	42A	21	P 5 INPUT	P 5 OUTPUT	USER P 5	SEGMENT B-
02C	42C	22	P 6 INPUT	P 6 OUTPUT	USER P 6	SEGMENT C-
02E	42E	23	P 7 INPUT	P 7 OUTPUT	USER P 7	SEGMENT D-
030	430	24	P 8 INPUT	P 8 OUTPUT	USER P 8	SEGMENT E-
032	432	25	P 9 INPUT	P 9 OUTPUT	USER P 9	SEGMENT F-
034	434	26	P10 INPUT	P10 OUTPUT	USER P10	SEGMENT G-
036	436	27	P11 INPUT	P11 OUTPUT	USER P11	SEGMENT P-
038	438	28	P12 INPUT	P12 OUTPUT	USER P12	DSPLYTRIGR-
03A	43A	29	P13 INPUT	P13 OUTPUT	USER P13	SHIFTLIGHT
03C	43C	30	P14 INPUT	P14 OUTPUT	USER P14	SPKRDRIVE
03E	43E	31	P15 INPUT	P15 OUTPUT	USER P15	WDATA

Q1-1 Bit de contrôle

Ce bit peut être positionné à 1 ou à 0 par une simple instruction CRU

* à 0 : l'information lue (mode entrée) pour les adresses 1 à 15 correspond aux bits $\overline{INT}_1$ à $\overline{INT}_{15}$, tandis que l'information écrite (mode sortie) concerne les bits MASK1 à $MASK_{15}$
* à 1 : en mode entrées les adresses 1 à 15 correspondent aux bits CLK_{1A} à CLK_{14A} ainsi que $\overline{INTREQ}$, tandis qu'en mode sortie, ces adresses correspondent aux bits CLK_{1B} à CLK_{14B} ainsi que $\overline{RST}_2$.

En mode écriture ou en mode lecture, la valeur du bit de contrôle n'a aucune influence sur les bits d'adresse 16 à 31 correspondant à P_0 jusqu'à P_{15}.

Q1-2 $\overline{INT}_1$ à $\overline{INT}_{15}$

L'état de ces lignes est représenté par les bits 1 à 15, même si ces lignes ne sont pas utilisées comme interruption mais comme entrées ou sorties ; la lecture des bits $\overline{INT}_1$ à $\overline{INT}_{15}$ est équivalente à la lecture des bits P_{15} à P_1.

Q1-3 $MASK_1$ à $MASK_{15}$

Comme on vient de voir [paragraphe c] ; les lignes d'interruption (1 à 15) peuvent être masquées individuellement sans aucune relation de niveau de priorité. Donc pour valider ou masquer une interruption, on associe le bit de masque correspondant au niveau 1 ou 0 respectivement ; valider l'INT_5 équivaut à mettre à 1 le bit masque 5.

Q1-4 CLK_{1A} à CLK_{14A}

Ces bits 1 à 14 permettent de lire l'état du décompteur Horloge temps réel.

Q1-5 CLK_{1B} à CLK_{14B}

Ces 14 bits correspondent à une valeur de temporisation et définissent une période de comptage de l'horloge temps réel ; la valeur de temporisation est mémorisée dans le registre d'Horloge afin de pouvoir la charger cycliquement dans le décompteur.

Q1-6 $\overline{INTREQ}$

Ce bit indique, s'il est au niveau 0, qu'il y a au moins une demande d'interruption active ; il est équivalent au signal INTREQ disponible sur les broches du circuit, signalant que pour le TMS 9980A ce signal n'existe pas. Le TMS 9980A compare en permanence le code IC_0 à IC_2 avec le masque d'interruption contenu dans les bits ST_{12} à ST_{15} du registre d'état.

Q1-7 $\overline{RST}_2$

Le positionnement à 0 de ce bit provoque une réinitialisation logicielle du TMS 9901.

Q1-8 P_0 à P_{15} (sorties)

Ces bits positionnent à 1 ou à 0 les lignes d'entrée/sorties P_0 à P_{15}

Q1-9 P_0 à P_{15} (Entrées)

Ces bits reflètent l'état des lignes d'entrées-sorties P_0 à P_{15}

a3- Passage d'un mode à un autre.

Après une réinitialisation matérielle, toutes les broches fonctionnent en entrée jusqu'à ce qu'on essaie d'écrire ; à cet instant elles fonctionnent en sortie et restent dans ce mode jusqu'à ce qu'une réinitialisation générale se présente, alors elles passent en mode d'entrée ou qu'on exécute la fonction de réinitialisation logicielle $\overline{RST_2}$ qui assure ce passage. On revient au mode de fonctionnement normal du TMS 9901 en mettant à 0 le bit de contrôle.

b. Interface système :

On a vu que l'interface système se compose de 22 lignes réparties en 3 groupes :

Groupe 1 :

Ses 6 lignes peuvent être configurées individuellement comme entrée ou interruption. Après une réinitialisation matérielle par $\overline{RST_1}$, ces lignes sont configurées en mode d'entrée et les interruptions sont masquées [bits $MASK_1$ à $MASK_6$ sont au niveau 0], ces lignes même démasquées peuvent être utilisées comme entrées ou sorties et sont lues par l'intermédiaire de l'interface CRU.

La figure [2-7 Gr 1] donne un exemple de configuration associée :

2-7 a1. Après une réinitialisation matérielle

2-7 b1. Après l'exécution d'un programme validant $\overline{INT_1}$ et $\overline{INT_2}$ et masquant les $\overline{INT_3}$ à $\overline{INT_6}$

Si on applique $\overline{RST_1}$ à l'état b1, on provoque une réinitialisation complète (retour à l'état a1), par contre l'application de $\overline{RST_2}$ ne provoque aucun effet, car elle n'affecte pas les masques d'interruptions.

Groupe 2

Les 9 lignes de ce groupe sont configurées individuellement comme interruption ou ligne d'entrées-sorties. Après une réinitialisation matérielle, ces lignes sont configurées en mode d'entrée et les interruptions sont masquées [bits $MASK_7$ à $MASK_{15}$ sont au niveau 0] ; On envisage 2 types d'action :

* Démasquage (validation) d'une ligne d'interruption
* Passage d'une ligne en mode sortie.

ces 2 cas ont été traités précédemment.

Donc il est impossible de faire passer une ligne en mode entrée sans affecter les autres lignes. La figure [2-7 Gr 2] donne un exemple de configuration logicielle associée à ce groupe :

2-7 a2. Après une réinitialisation matérielle

2-7 b2. Après l'exécution du programme suivant.

```
LI    R12, CRUBASE
SBO   0
LI    R12, CRUBASE + 16
LDCR  DMN, 3          ← 2 fois adresse masque 8
 :
```

Fig 2-1 Configuration des groupes

MN DATA >x7xx MASK$_8$ = 1, MASK$_9$ = 1, MASK$_{10}$ = 1 } validation des interruptions INT_8, ,

CRUBASE = >0000 ou = > 0400 Suivant qu'on utilise U_{10} ou U_{11} respectivement de la carte

2-7 L2. Après l'exécution du programme suivant :

```
LI    R12 , CRUBASE + 46
LDCR  @MS, 4
:
MS    DATA  >x5xx
```

l'application de $\overline{RST_1}$, à partir de l'état (2-7 b2) ou l'état (2-7 C2), ramène les 2 états à l'état (L2) [réinitialisation générale], tandis que l'application de $\overline{RST_2}$ à l'état (2-7 b2) ne modifie rien, mais son application à l'état (2-7 C2) ramène le système à l'état (2-7 b2), c'est-à-dire les lignes de sortie sont ramenées en mode entrée sans affecter les bits de masque.

Groupe 3

Les 7 lignes de ce groupe sont configurées en mode entrée ou sortie; elles ne peuvent pas être utilisées comme lignes d'interruption, comme en groupe 2 toute ligne configurée en mode sortie ne peut être remise en mode entrée que par suite d'une réinitialisation matériel-le ($\overline{RST_1}$) ou logicielle ($\overline{RST_2}$). La figure [2-7 Gr 3] donne une configuration de ce groupe :

2-7 a3, Après une réinitialisation materielle

2-7 b3, Après l'exécution du programme suivant

```
LI    R12 , CRUBASE + 38
LDCR  @MA, 2
:
MA    DATA  x2xx
```

LDCR positionne les P_i en mode sortie, l'application de $\overline{RST_1}$ ou $\overline{RST_2}$ à l'état [2-7 b3] provoque un retour à l'état [2-7 a3]

C. Interruptions :

Comme on vient de voir, toutes les lignes du premier et du second groupe peuvent être utilisées comme lignes d'interruption, et pour valider une interruption d'un niveau quelconque on positionne à un le bit de masque correspondant. Le TMS 9901 possède 15 lignes d'interruption dont le niveau 0 est le plus prioritaire.

traitement des interruptions :

Après échantillonnage par le signal $\overline{\phi}$, chaque ligne d'interruption est traitée en fonction du bit de masque positionné par le programme en cours [fig 2-8] : une interruption codée

par le circuit encodeur peut être transmise ou non suivant l'état du bit de masque ; si elle est transmise, elle provoque un changement de contexte et l'exécution d'un programme spécial où le processeur répond de la façon illustrée par l'organigramme [fig 2-4] et force ainsi le masque interne [bit ST_{12} à ST_{15}] du registre d'état à un niveau inférieur d'une unité au niveau d'interruption reconnu de façon à n'autoriser que les demandes d'interruption plus prioritaire . L'instruction RTWP restaure le registre d'état initial et par conséquent le masque initial, ainsi que le PC et le WP . Il est nécessaire donc que chaque programme ait son propre espace de travail pour sauvegarder l'environnement (PC, ST, WP) du programme interrompu.

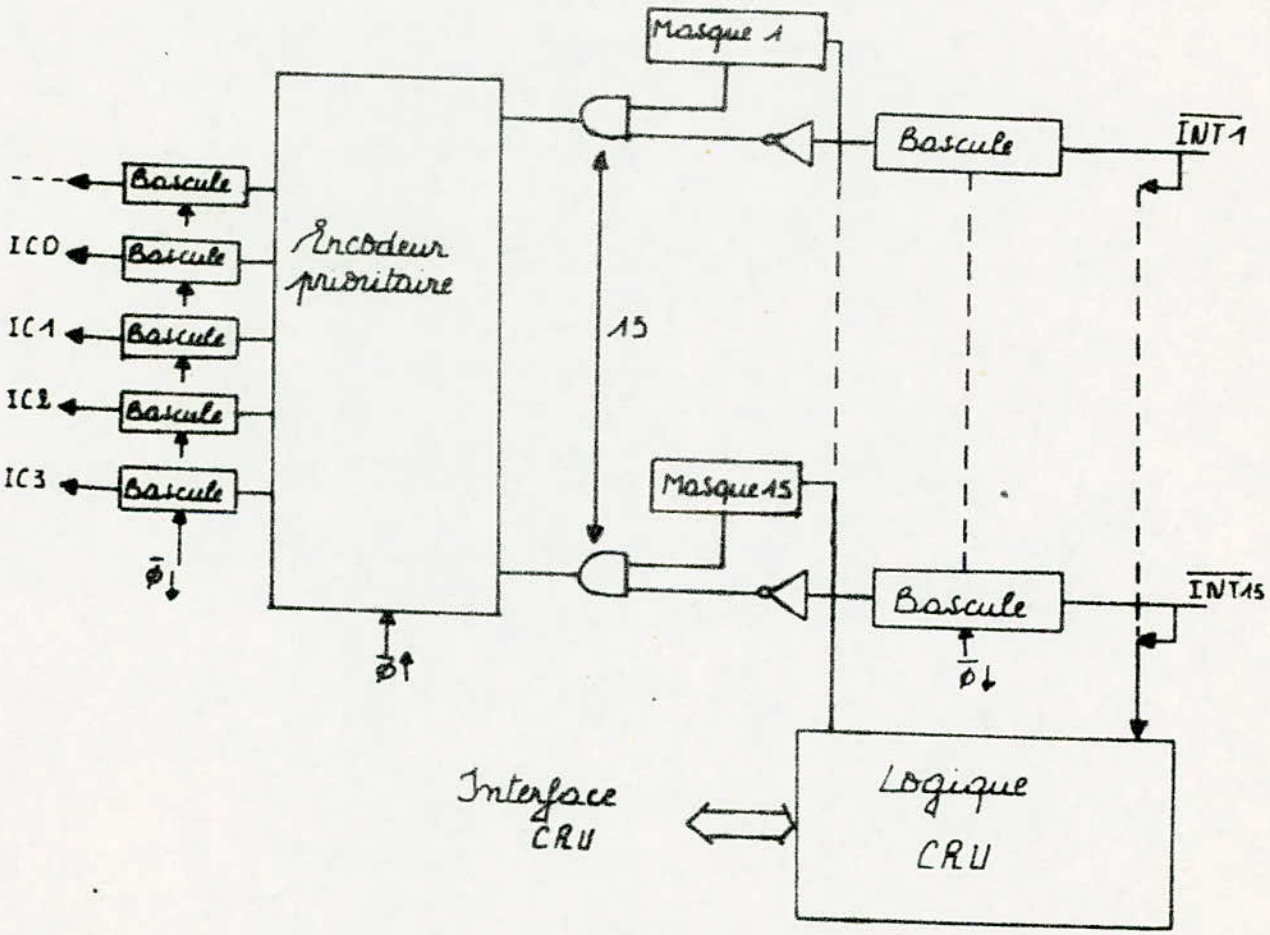


Fig 2-8 Logique de contrôle des interruptions

CHAPITRE III

A. La carte TM 990/189

1. Description et présentation :

La carte TM 990/189 est un micro-ordinateur destiné à l'étude et à la compréhension des connaissances de base nécessaires pour l'utilisation des microprocesseurs de la famille 9900, avec elle on découvre également la puissance de l'architecture à 16 bits de cette famille, en particulier le CPU TMS 9980A et l'interface programmable TMS 9901. La figure [3.1] montre l'implantation des composants de cette carte ; nous citons :

* Le microprocesseur (U_{19})
* 1 K d'octets de RAM (U_{20} et U_{22}) extensible en U_{21} et U_{23} à 2 K d'octets.
* 4 K d'octets de ROM (U_{33}) extensible en U_{32} et U_{31} à 6K d'octets
* Horloge à quartz
* Clavier 45 touches alphanumériques
* dispositifs de visualisation 10 digits à 7 segments et un point décimal chacun.
* indicateurs acoustiques (haut-parleur) et optique (LED)
* interface cassette audio.
* Connecteur de bus à 40 broches
* connecteur d'entrée/sortie à 40 broches

2. Schéma fonctionnel :

Un schéma fonctionnel de cette carte est donné par la figure [3.2].

2-1: Le clavier :

Le clavier sert à introduire les données et les commandes à destination du microprocesseur, il peut fonctionner en mode principal ou en mode secondaire. L'initialisation et la mise sous tension sont données par l'organigramme [fig 3.3].

2-2: L'interrupteur LOAD

L'utilisation de cet interrupteur ne modifie pas le contenu de la mémoire utilisateur, mais elle peut générer une interruption de chargement non masquable ou abandonner un programme en cours d'exécution pour se replacer sous le contrôle de UNIBUG.

2-3: Les afficheurs :

Le dispositif de visualisation, ayant une capacité de 10 Digits, permet l'affichage des commandes, des données et les messages d'erreur.

fig 3-1
Schéma d'implantation des composants sur la carte Université TM 990/189.

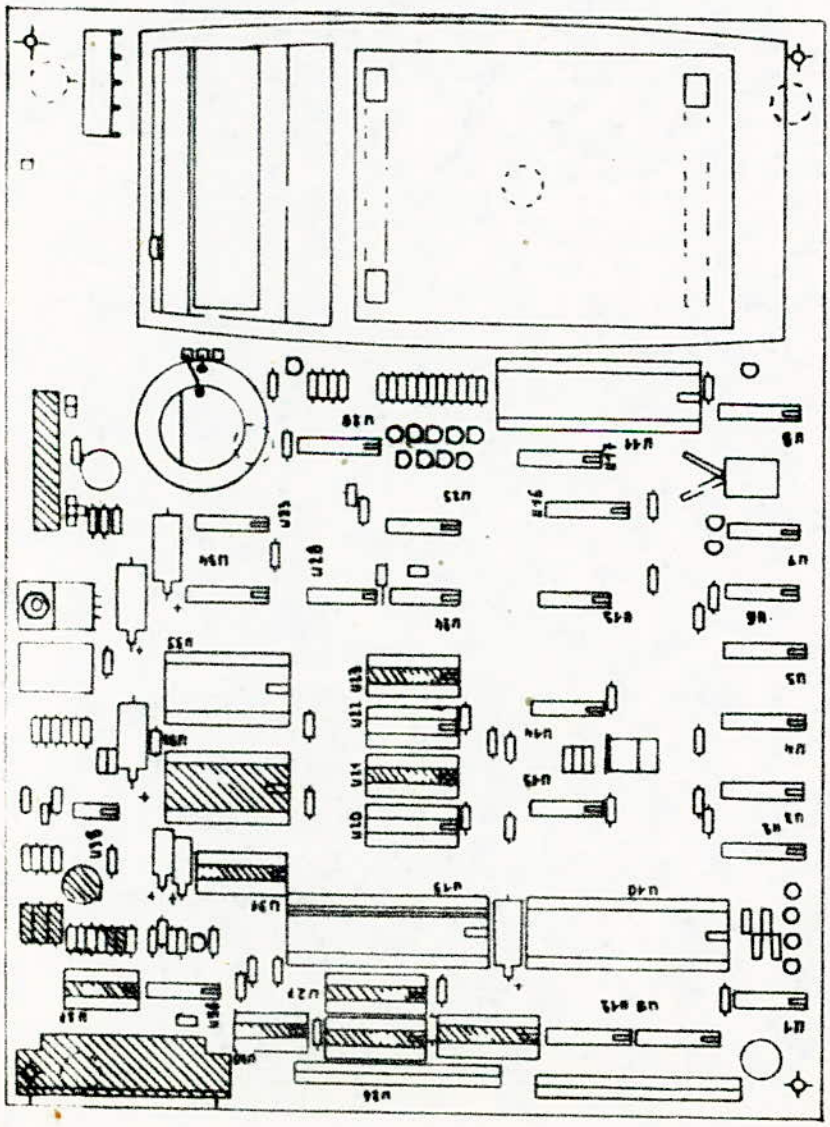


fig 3.2 Schéma fonctionnel du système.

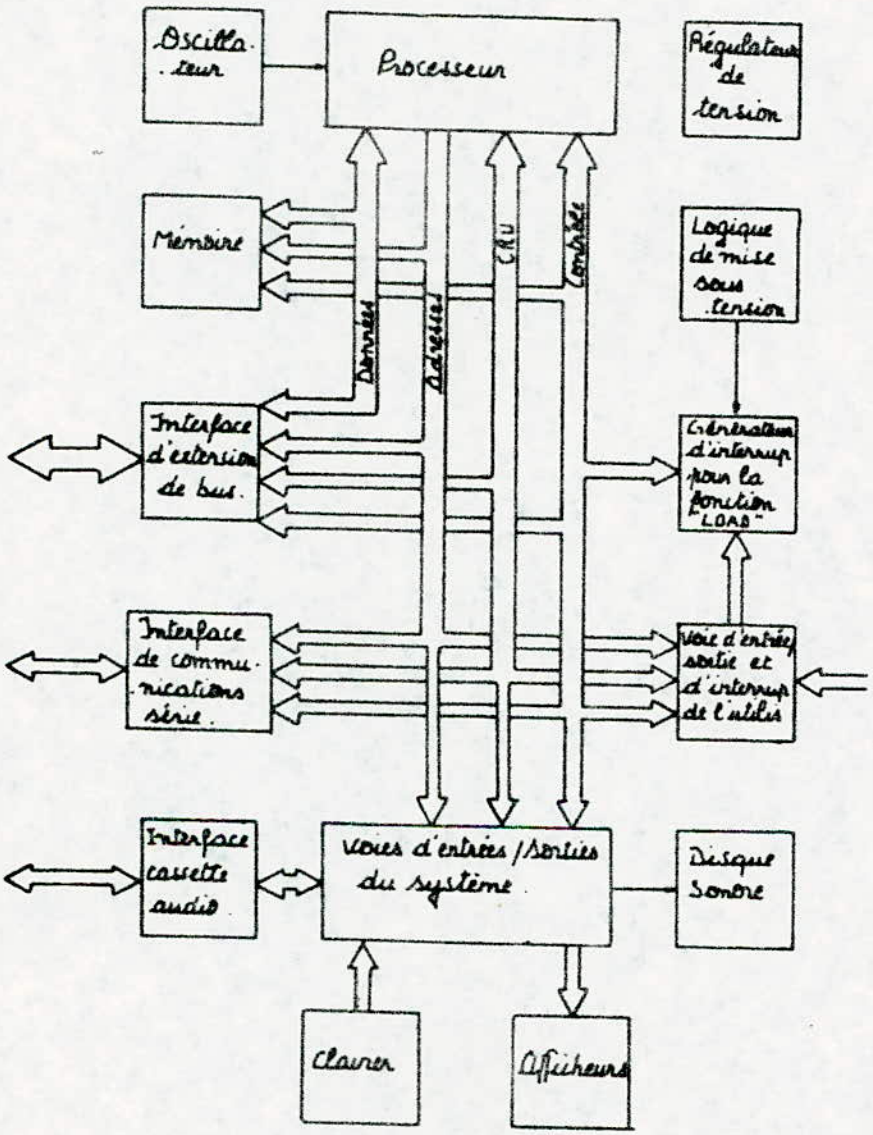
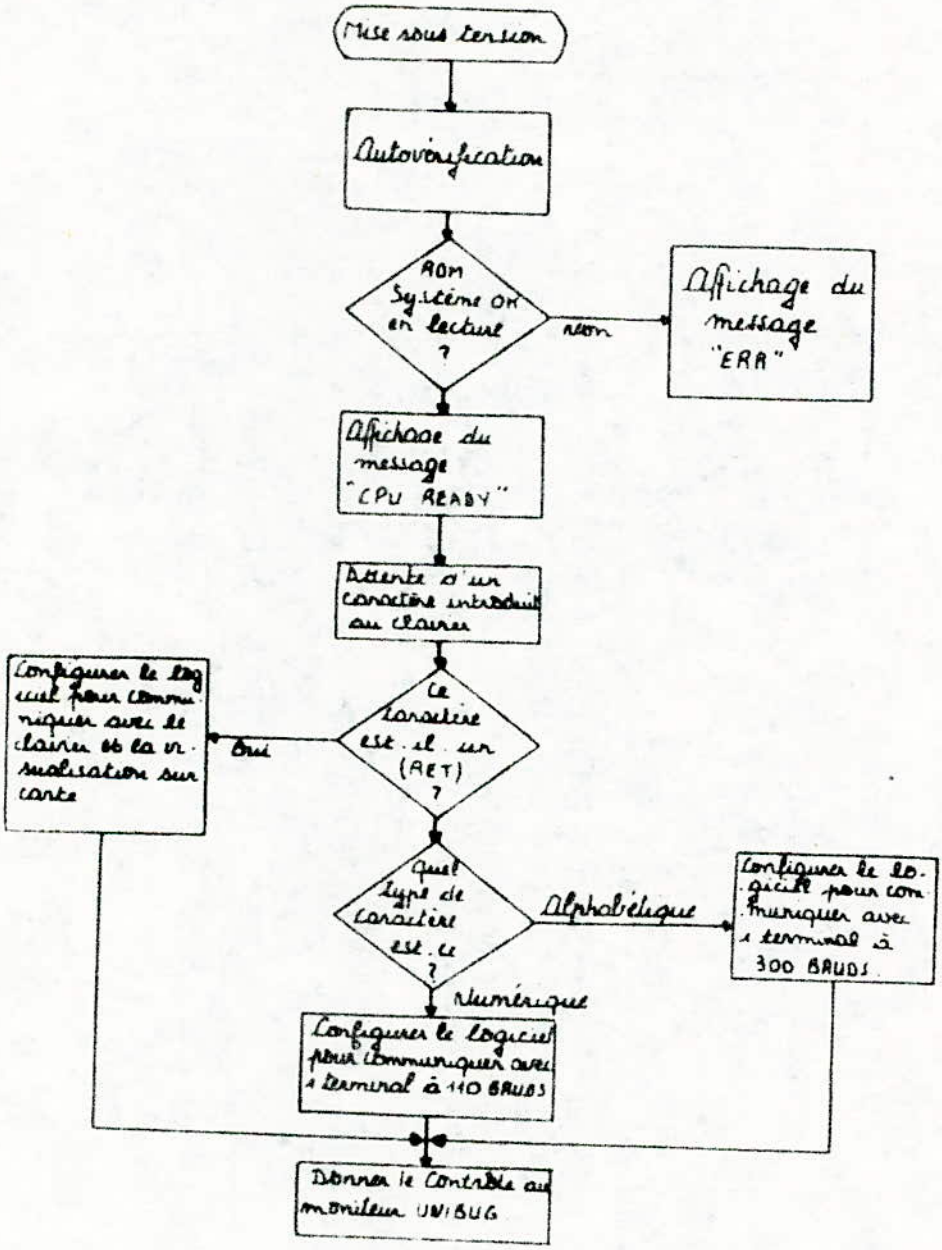


fig 3.3 Organigramme de l'initialisation à la mise sous tension



2.4 : interface cassette audio

Sur la carte, on trouve un interface adéquat pour la connection d'un lecteur / enregistreur de commerce permettant ainsi le chargement d'un programme d'une cassette en mémoire ou l'inverse.

2.5 : Indicateur acoustique

Sous le contrôle d'un programme et en utilisant ce haut-parleur On peut émettre des fréquences audibles et programmer des chansons.

2.6 mémoires :

la mémoire de la carte TM 990/189 se répartie en :

- RAM : mémoires actives utilisées pour ranger les programmes et les données.
- ROM : mémoires mortes contenant le moniteur et l'assembleur.

la répartition de la mémoire de la carte a été donnée précédemment [fig 2.4].

Registres

L'exécution d'un programme met en œuvre 3 registres principaux: le compteur programme, le pointeur d'espace de travail et le registre d'état.

3. Le moniteur Unibug :

L'Unibug est un programme général de supervision du système, capable d'exécuter certaines fonctions permettant à l'opérateur de communiquer avec lui.

Les commandes UNIBUG.

L'UNIBUG reconnait 15 commandes fournies par la table suivante :

commande	action
A	exécution de l'assembleur
B	exécution de l'assembleur avec table des symboles courants
C	inspection / modification du CRU
D	vidage
E	exécution jusqu'à point d'arrêt
F	inspection / modification du registre d'état
J	saut vers EPROM
L	chargement de la mémoire à partir d'une cassette
M	inspection / modification de la mémoire
P	inspection / modification du compteur programme
R	inspection / modification du registre de travail
S	exécution pas à pas
T	programme machine à écrire
W	inspection / modification du pointeur d'espace de travail
RET	Retour à la ligne.

Les sous-programmes fournis par UNIBUG:

N° du XOP	FONCTION
8	écrire sur le terminal un caractère hexadécimal
9	lire un mot hexadécimal sur le terminal
10	écrire sur le terminal 4 caractères hexadécimaux
11	lire un caractère avec écho
12	écrire sur le terminal un caractère
13	lire sur le terminal un caractère
14	écrire un message sur le terminal
15	réservé à UNIBUG

4. L'assembleur:

L'assembleur est un programme permettant de convertir le langage assembleur en code objet. La compréhension, l'exécution assez rapide et l'occupation efficace des mémoires de ce langage permet une grande utilité pour les instructions temps réel des microprocesseurs.

4.1: fonctionnement de l'assembleur:

L'assembleur de la carte TM990/189 assure les tâches suivantes:

* gestion des adresses: il se charge de repérer les emplacements mémoires à la place du programmeur.
* définition des constantes symboliques: l'utilisation d'une constante symbolique facilite la modification d'un programme; il suffit de la modifier en un seul endroit pour que l'assembleur se charge de la modifier partout où elle est utilisée.
* identification des erreurs: l'assembleur émet un message d'erreur lors d'une identification d'une erreur de syntaxe.
* contrôle des informations en sortie: l'assembleur fournit deux types d'informations en sortie: le code objet qui peut être lu par l'ordinateur, et le listing pour être lu par l'homme.

4.2: formats:

L'assembleur de cette carte regroupe 69 instructions et reconnaît 6 directives et une pseudo-instruction. La constitution du jeu complet de ces instructions nécessite l'utilisation de 9 formats permettant une modification rapide du programme en mémoire. La table suivante fournit ces 9 formats:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	OP code				B	TD	DR				TS		SR			
2	OP code								déplacement signé							
3	OP code						WR				TS		SR			
4	OP Code						C				TS		SR			
5	OP code								C				R			
6	OP Code										TS		SR			
7	OP code												Non utilisé			
8	OP Code											N	R			
9	OP code						DR				TS		SR			

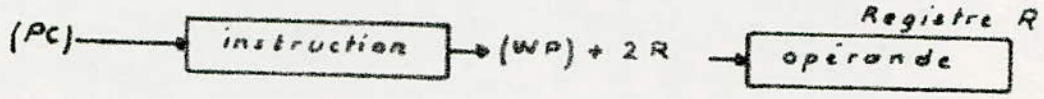
abréviations :

- B : indicateur d'octet
- TD : mode d'adressage pour l'opérande destination
- TS : mode d'adressage pour l'opérande Source
- DR : Registre destination
- SR : Registre Source
- C : Compteur de décalage ou d'échange CRU
- R : Registre
- N : non utilisé.

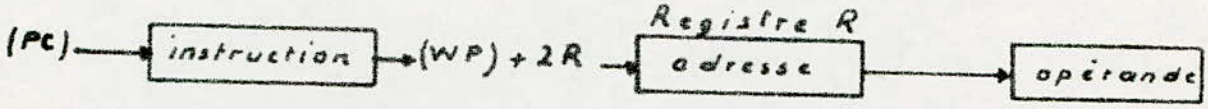
4.3 : modes d'adressage :

La carte TM 990/189 reconnait 7 modes d'adressage :

4.3.1 : adressage direct par registre : dans ce mode les opérandes sont des registres.

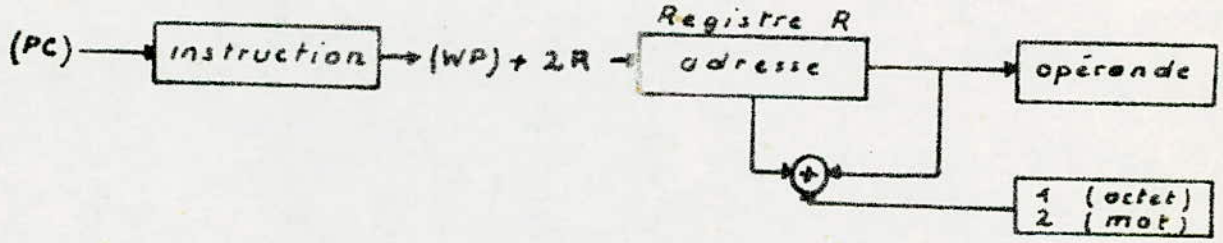


4.3.2 : adressage indirect par registre : le registre ne contient pas l'opérande mais l'adresse de celle-ci



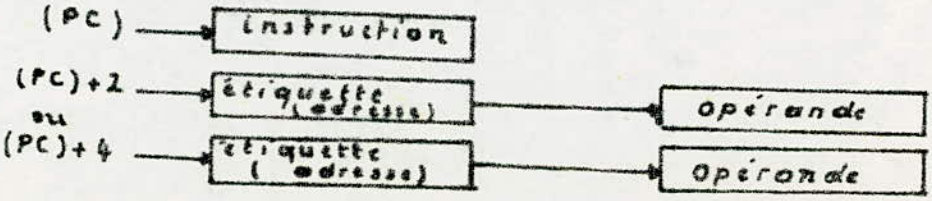
4.3.3 : adressage indirect par registre avec autoincrémentation :

Il diffère de son précédent par la propriété supplémentaire qu'a le registre d'incrémenter automatiquement son contenu par un ou deux selon qu'on travaille en octet ou en mot.



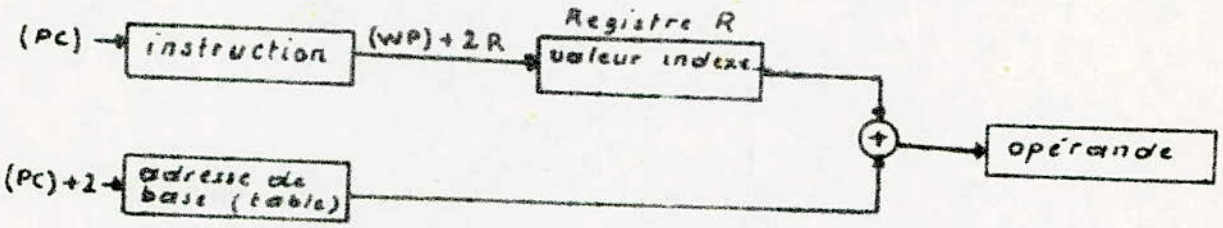
4.3.4 : adressage symbolique en mémoire :

dans ce cas les registres n'interviennent pas, et l'adresse est une valeur de 16 bits rangée dans le 2e ou le 3e mot de l'instruction.



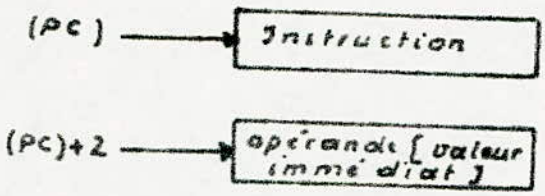
4-3-5: adressage indexé :

C'est la combinaison de l'adressage symbolique et l'adressage indirect par registre, l'adresse du mot mémoire est obtenue par l'addition du contenu du registre spécifié à l'opérande.

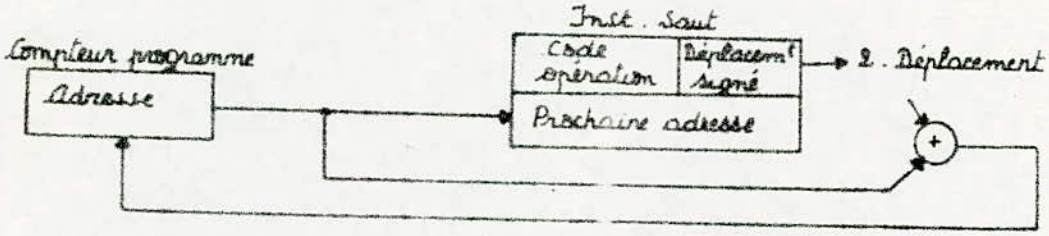


4-3-b: modes d'adressages particuliers :

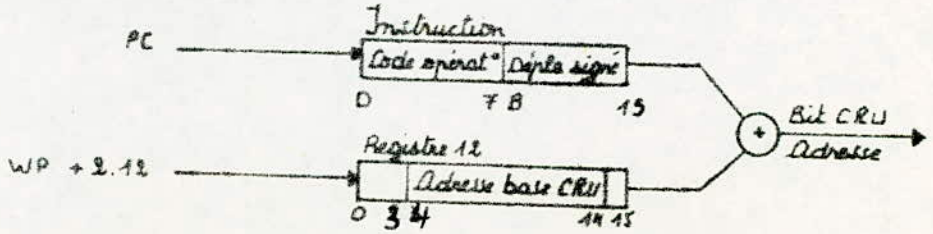
* adressage immédiat : Les instructions utilisant ce mode, contiennent la donnée pour être utilisée comme une partie de l'instruction; le premier mot est le code objet de l'instruction, le second est la donnée à utiliser.



* adressage relative au compteur programme : Ce mode d'adressage est utilisé pour modifier directement par instruction le compteur programme ; les instructions conditionnelles de branchement et de saut utilisent ce mode d'adressage.



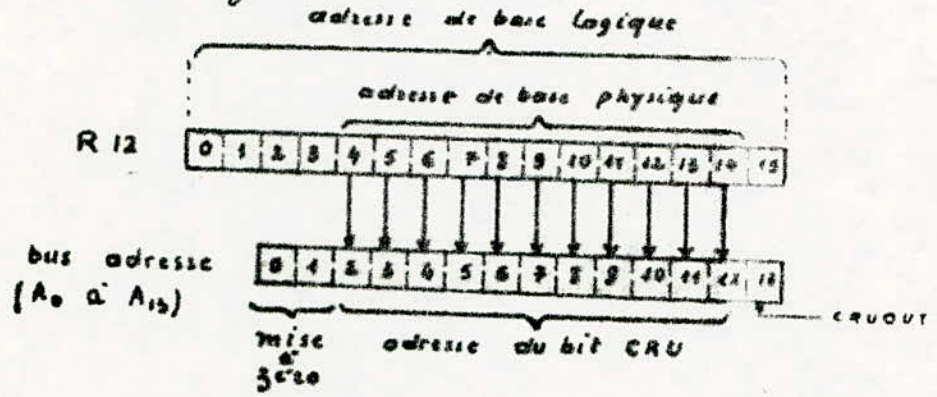
* adressage CRU : Le déplacement signé de 8 bits contenu dans l'octet droit d'une instruction CRU utilisant le format 2 (SBO, SBZ et TB) est additionné à l'adresse de base CRU (bits 4 à 14 du registre 12); le résultat constitue l'adresse CRU du bit sélectionné.



La figure suivante nous montre les 2 adresses de base CRU physique et logique:

l'adresse de base CRU physique est la valeur présente sur les lignes d'adresses A_2 à A_{12} ; le CPU utilise les bits 4 à 14 du registre R_{12} pour obtenir cette adresse, donc on a une capacité d'adressage de 2^{11}: 2048 périphériques différents

l'adresse de base CRU logique est la valeur donnée par les 16 bits du registre 12.



La table suivante fournit l'implantation des adresses CRU pilotées par le TMS 9901 de la carte TM 990/189:

contenu de R_{12} bits 0 à 15	contenu de R_{12} bits 4 à 14	Fonction
0000 à 003E	0000 à 001F	TMS 9901 utilisateur (U_{10}) (fonctionne avec P5)
0400 à 043E	0200 à 021F	TMS 9901 Système (U_{11})
0800 à 083E	0400 à 041F	TMS 9902 Système (interface EIA sur P1)
0C00 à 0C3E	0600 à 061F	adresse CRU à la disposition de l'utilisateur extensible à 512 bits

L'utilisation de cette table avec la figure [2-6] fournit l'adresse CRU des périphériques présents dans la carte (diodes, haut parleur, diode shift ...)

La table suivante récapitule les principaux modes d'adressage en utilisant l'instruction MOV (transférer)

mode d'adressage	opérande		exemple	résultat	TD ou TS
Direct par registre	n	registre de travail R_n	MOV 3,5	$R_3 \rightarrow R_5$	0 0
indirect par registre	*n	l'adresse est donnée par le contenu du registre de travail R_n : M (R_n)	MOV *R_3, *R_5	$M(R_3) \rightarrow M(R_5)$	0 1
indirect par registre autoincrémenté	*n+	Le contenu du registre R_n est incrémenté automatiquement après chaque opération par 1 ou 2	MOV #3+, #5+ (mot) MOVB #3+, #5+ (octet)	$M(R_3) \rightarrow M(R_5)$ $R_3 + 2 \rightarrow R_3$ $R_5 + 2 \rightarrow R_5$ $M(R_3) \rightarrow M(R_5)$ $R_3 + 1 \rightarrow R_3$ $R_5 + 1 \rightarrow R_5$	1 1
adressage Symbolique	@MN	l'adresse est donnée par la valeur de MN : M (MN)	MOV @AA, @BB	$M(AA) \rightarrow M(BB)$	1 0
Symbolique en mémoire indexé	@MN(n)	l'adresse est donnée par la somme de contenu de R_n et la valeur de MN : M (R_n + MN)	MOV @AA(R_4), @BB(R_3)	$M(R_4 + AA) \rightarrow M(R_3 + BB)$	1 0

n : représente le numéro du registre de travail $0 \leq n \leq 15$
n = 0 ne peut pas être utilisé pour un adressage indexé

MN : peut être un symbole, nbre ou expression.

4-4 : Jeux d'instructions

langage assem. - Bleur	Code opération	Octets	Action	C	Bits d'état affectés	Description
A	1 0 1 0	1	Addition (mots)	Oui	0 - 4	(SA) + (DA) ⟶ (DA)
AB	1 0 1 1	1	Addition (octets)	Oui	0 - 5	(SA) + (DA) ⟶ (DA)
C	1 0 0 0	1	Comparaison (mots)	non	0 - 2	Comparer (SA) à (DA) et positionner les bits d'état concernés.
CB	1 0 0 1	1	Comparaison (octets)	non	0 - 2,5	Comparer (SA) à (DA) et positionner les bits d'état concernés.
MOV	1 1 0 0	1	Transfert (mots)	Oui	0 - 2	(SA) ⟶ (DA)
MOVB	1 1 0 1	1	Transfert (octets)	Oui	0 - 2,5	(SA) ⟶ (DA)
S	0 1 1 0	1	Soustraction (mots)	Oui	0 - 4	(DA) - (SA) ⟶ (DA)
SB	0 1 1 1	1	Soustraction (octets)	Oui	0 - 5	(DA) - (SA) ⟶ (DA)
SOC	1 1 1 0	1	Mettre à un les bits correspondants (mots)	Oui	0 - 2	(DA) ou (SA) ⟶ (DA)
SOCB	1 1 1 1	1	Mettre à un les bits correspondants (octets)	Oui	0 - 2,5	(DA) ou (SA) ⟶ (DA)
SZC	0 1 0 0	1	Mettre à zero les bits correspondants (mots)	Oui	0 - 2	(DA) et $(\overline{SA})$ ⟶ (DA)
SZCB	0 1 0 1	1	Mettre à zero les bits correspondants (octets)	Oui	0 - 2,5	(DA) et $(\overline{SA})$ ⟶ (DA)
JEQ	0 0 0 1 0 0 1 1	2	Saut si égalité (arithmétique)			ST2 = 1
JGT	0 0 0 1 0 1 0 1	2	Saut si supérieur à (arithmétique)			ST1 = 1
JH	0 0 0 1 1 0 1 1	2	Saut si supérieur à (logique)			ST0 = 1 et ST2 = 0
JHE	0 0 0 1 0 1 0 0	2	Saut si supérieur ou égal (logique)			ST0 = 1 ou ST2 = 1
JL	0 0 0 1 1 0 1 0	2	Saut si inférieur à (logique)			ST0 = 0 et ST2 = 0
JLE	0 0 0 1 0 0 1 0	2	Saut si inférieur ou égal (logique)			ST0 = 0 ou ST2 = 1
JLT	0 0 0 1 0 0 0 1	2	Saut si inférieur à (arithmétique)			ST1 = 0 et ST2 = 0
JMP	0 0 0 1 0 0 0 0	2	Saut inconditionnel			Inconditionnel

JNC	0 0 0 1 0 1 1 1	2	Saut si pas de retenue			ST3 = 0
JNE	0 0 0 1 0 1 1 0	2	Saut si inégalité			ST2 = 0
JND	0 0 0 1 1 0 0 1	2	Saut si pas de dépas-sement			ST4 = 0
JDC	0 0 0 1 1 0 0 0	2	Saut si retenue			ST3 = 1
JOP	0 0 0 1 1 1 0 0	2	Saut si parité impaire			ST5 = 1
SBO	0 0 0 1 1 1 0 1	2	Mettre le bit à un			Mettre à un le bit choisi dans le registre CRU
SBZ	0 0 0 1 1 1 1 0	2	Mettre le bit à zéro			Mettre à zéro le bit choisi dans le registre CRU
TB	0 0 0 1 1 1 1 1	2	Vérifier l'état du bit			Si le bit d'entrée CRU est à un mettre ST2 = 1
COC	0 0 1 0 0 0	3	Comparer les bits à un correspondants	non	2	Contrôler (DR) pour savoir si chacun de ses bits correspondant (en position) aux bits à un de (SA) sont à un. Si oui, mettre ST2 à un
CZC	0 0 1 0 0 1	3	Comparer les bits à zéro correspondants	non	2	Contrôler (DR) pour savoir si chacun de ses bits correspondant (en position) aux bits à un de (SA) sont à zéro. Si oui, mettre ST2 = 1.
XOR	0 0 1 0 1 0	3	Ou exclusif	Ou	0. 2	(DR) ⊕ (SA) ⟶ (DR)
MPY	0 0 1 1 1 0	3	Multiplication	non		Multiplier le contenu (considéré comme un nombre non signé) de DR par le contenu non signé de SA; placer le produit non signé (sur 32 bits) dans DR (mot de poids fort) et dans DR + 1 (mot de poids faible). Si DR est le registre 15, on chargera la moitié de poids faible du produit dans le mot suivant en mémoire

DIV	0 0 1 1 1 1	3	Division	non	4	Si le contenu (non signé) de SA est inférieur ou égal au contenu non signé de DR, on n'effectue pas la division et l'on met ST4 à un. Sinon on divise (DR) et (DR+1) par (SA). Le quotient est mis dans DR et le reste est mis dans DR+1. Si DR = 15, le reste est mis dans le mot suivant en mémoire le registre 15.
LDCR	0 0 1 1 0 0	4	Chargement du registre de communication CRU (émission)	oui	0.2,5	On transfère le nombre de bits indiqué par C, de (SA) vers le CRU, en commençant par le bit de poids faible de (SA).
STCR	0 0 1 1 0 1	4	Rangement du registre de communication CRU (réception)	oui	0.2,5	On range dans (SA) le nombre de bits indiqué par C, et qui est lu sur le CRU, en commençant par le bit de poids faible de (SA). Les positions de SA non utilisées sont initialisées à zéro
SLA	0 0 0 0 1 0 1 0	5	Décalage arith à gauche	oui	0.4	Décaler (R) à gauche. Les positions libérées sont mises à zéro.
SRA	0 0 0 0 1 0 0 0	5	Décalage arith à droite	oui	0.3	Décaler (R) à droite. Les positions libérées ont la même valeur que le bit de signe (MSB de R).
SRC	0 0 0 0 1 0 1 1	5	Décalage à droite circulaire.	oui	0.3	Décaler (R) à droite. Le LSB sortant est réintroduit dans le MSB.

SRL	0 0 0 0 1 0 0 1	5	Décalage à droite logique.	ou	0..3	Décaler (R) à droite Mettre à zero les bits libérés.
B	0000010001	6	Branchement	non		SA ⟶ PC
BL	0000011010	6	Branchement avec chainage	non		(PC) ⟶ (R_{11}) ; SA ⟶ (PC)
BLWP	0000010000	6	Branchement avec changement de context	non		(SA) ⟶ (WP), (SA+2) ⟶ (PC) (ancien WP) ⟶ (nouveau Reg 13) (ancien PC) ⟶ (nouveau Reg 14) (ancien ST) ⟶ (nouveau Reg 15) L'entrée intreq n'est pas prise en compte jusqu'à l'exécution complète de l'instruction BLWP.
CLR	0000010011	6	Remise à zero de l'opérande	non		0000 ⟶ (SA)
SETO	0000011100	6	Mise à un de tous les bits de l'opérande	non		$FFFF_{16}$ ⟶ (SA)
INV	0000010101	6	Complémentation	ou	0..2	$\overline{(SA)}$ ⟶ (SA) (complement à un)
NEG	0000010100	6	Négation	ou	0..4	-(SA) ⟶ (SA) Complément à deux
ABS	0000011101	6	Valeur absolue	non	0..4	|(SA)| ⟶ (SA)
SWPB	0000011011	6	Echange d'octets	non	0..4	(SA), bits 0 à 7 ⟶ (SA), bits 8 à 15. (SA), bits 8 à 15 ⟶ (SA), bits 0 à 7.
INC	0000010110	6	Incrémentation	ou	0..4	(SA) + 1 ⟶ (SA)
INCT	0000010111	6	Incrémentation par 2	ou	0..4	(SA) + 2 ⟶ (SA)
DEC	0000011000	6	Décrémentation	ou	0..4	(SA) - 1 ⟶ (SA)
DECT	0000011001	6	Décrémentation par 2	ou	0..4	(SA) - 2 ⟶ (SA)
X	0000010010	6	Exécution	non		Exécuter l'instruction située à SA. On suspend le travail du microprocesseur jusqu'à l'apparition d'une interruption du type LOAD ou RESET.
INE	0000001101 0	7	Mise à l'état de repos			
RSET	0000001101 1	7	A définir par l'utilisateur		12..15	0 ⟶ ST12 à ST15

CKOF	0000001111 0	7	À définir par l'utilisateur			
CKON	0000001111 01	7	À définir par l'utilisateur			
LREX	0000001111 11	7	Interruption de type LOAD			Le contrôle est rendu à Unibus (CPU READY)
RTWP	0000001111 00	7	Retour du sous-programme		0..15	Restauration du Contexte: (R13) → (WP), (R14) → (PC) (R15) → (ST)
AI	0000001000 01	8	Addition immédiate	ou	0..4	(R) + IOP → (R)
ANDI	0000001001 10	8	ET logique immédiat	ou	0..2	(R) ET IOP → (R)
CI	0000001010 00	8	Comparaison immédiate	ou	0..2	Comparer (R) à IOP Positionner les bits d'état en conséquence
LI	0000001000 0	8	Chargement immédiat	ou	0..2	IOP → (R)
ORI	0000001001 11	8	OU logique immédiat	ou	0..2	(R) OU IOP → (R)
LWPI	0000001011 11	8	Chargement immédiat du pointeur d'espace de travail			IOP → (WP). Les bits du registre d'état ne sont pas affectés
LIMI	0000001100 0	8	Chargement du masque d'interruption			IOP (bits 12 à 15 → ST12 à ST15)
STST	0000001011 0	8	Sauvegarde du registre d'état			(ST) → (R)
STWP	0000001010 1	8	Sauvegarde du pointeur d'espace de travail			(WP) → (R)
XOP	001011	9	Opération étendue		6	

Passage du Langage assembleur en langage machine :

Exemple :

Addition :

Format A G_s , G_d

opération : la donnée d'adresse G_s est additionnée à la donnée d'adresse G_d , le résultat est placé dans cette dernière adresse et comparé à 0

$$M(G_s) + M(G_d) \longrightarrow M(G_d) \quad ;$$

$$M(G_d) : 0 \quad (\text{comparé à } 0)$$

bits d'état affectés : les bits 0 (LGT) ; 1 (AGT)
2 (EQ) ; 3 (C)
4 (OV) sont positionnés en conséquence.

exemples : a) **addition mot**

A R_1, R_2

opération : additionner le contenu de R_1 au contenu de R_2, placer le résultat dans R_2 et la comparer à 0.

Format : l'addition utilise le format 1, son code opération est donnée par la table du jeu d'instruction : (101)

addition mot (B) : B = 0
registre source (SR) : SR = 0001
adressage source (TS) : TS = 00 (adressage direct par Registre)
registre destination (DR) : DR = 0010
adressage destination (TD) : TD = 00 (adressage direct par Registre)

d'où :

en binaire :

en hexadécimal > A021

Application :

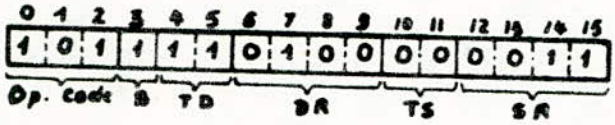
M(Gs)	M(Gd)	Résultat M(Gd)	bits d'état affectés				
			LGT	AGT	EQ	C	OV
1000	0001	1001	1	1	0	0	0
F000	1000	0000	0	0	1	1	0
F000	8000	7000	1	1	0	1	1
4000	4000	8000	1	0	0	0	1

b) addition (octets)

AB R3, *R4+ additionne l'octet de gauche contenu dans R3 à l'octet de gauche d'adresse R4, cette adresse est autoincrémentée par 1 après chaque opération.

Code opération	:	101
addition octet	B :	1
registre source (SR)	SR :	0011
adressage source (TS)	TS :	00
registre destination (DR)	DR :	0100
adressage destination (TD)	TD :	11 (adressage indirect par registre avec auto-incrémentation

d'où en binaire



en hexadécimal : > BD03

4-5 : Les directives :

Les directives sont des instructions destinées à l'assembleur seulement.

différentes directives :

* AORG : Contrôle de l'origine
  Format : AORG <espace> <origine>
  Application : Détermine l'adresse d'implantation en mémoire du code objet (initialisation de PC)
* CANC : Annulation d'une ligne :
  SHIFT / X
  Application : annule une fausse ligne pour la corriger ensuite.
* BSS : Déclaration d'une zone :
  Format : BSS <espace> <C>
  Application : réserve une zône de mémoires RAM destinée à recevoir des données diverses (les registres d'un espace de travail par exemple).
* DATA : initialisation des constantes :
  Format : [étiquette] <espace> DATA <espace> [constantes séparées par des ,]

Application: initialise des constantes, la première peut être définie ou pas encore.

* TEXT : initialisation d'une chaîne de caractères.

Format: [étiquette] <espace> TEXT <espace> ' <chaîne de caractères> '

Application: permet de ranger en mémoire une chaîne de caractères en code ASCII.

* EQU : définition d'une constante symbolique.

Format : <étiquette> <espace> EQU <espace> <constante définie>

Application : permet à l'utilisateur d'assigner une valeur d'1 symbole défini à un autre symbole encore indéfini.

* END : Fin de l'assemblage.

Format : END <espace> [point d'arrêt]

Application : permet de sortir de l'assembleur pour se mettre sous le contrôle d'UNIBUG.

4-6 : Pseudo-instruction :

L'assembleur reconnait une pseudo-instruction NOP (NO-Operation) qui n'est pas une instruction supplémentaire, mais une mnémonique supplémentaire représentant conventionnellement un membre du jeu d'instructions. Elle peut être incluse dans une routine pour forcer un temps d'exécution supplémentaire (temporisation) ou être utilisée pour remplacer une instruction devenue inutile dans un programme ; cette instruction est équivalente à JMP $+2 ayant pour code objet (1000_{16}).

B. Techniques de programmation :

Le processeur contient certains éléments de base comme il est indiqué sur la figure 2 [Chapitre II]. le timing et la section de contrôle sont les éléments les plus importants dans le système matériel qui doit faire en sorte que les evènements du système se présentent dans un ordre et un temps correct. Le système logiciel est intéressé par les opérations fournies par l'ALU et les registres déterminant l'instruction et l'adresse des données. Ces registres sont: le compteur programme (PC), le pointeur d'espace de travail (WP) et le registre d'état (ST).

a. Le compteur Programme (PC):

Comme l'indique la figure [3.4], le PC est un registre qui pointe vers l'adresse de l'instruction à exécuter. Après l'exécution d'une instruction, le PC est incrémenté automatiquement par 2, localisant ainsi l'adresse de l'instruction suivante. Son contenu peut être contrôlé par le programmeur en utilisant des instructions de branchement ou de saut.

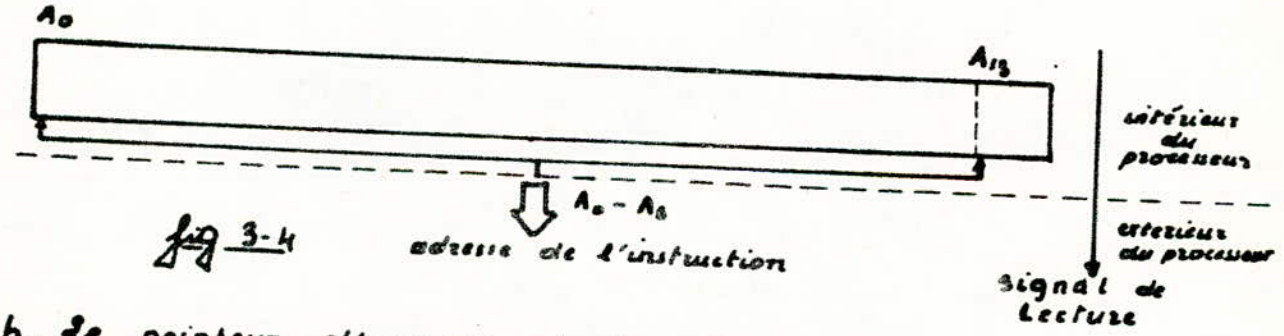


fig 3-4

b. Le pointeur d'espace de travail (WP):

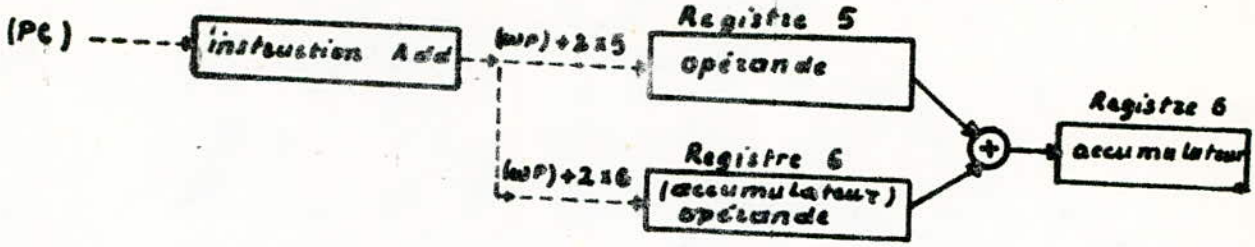
Nous avons vu précédemment que le WP est un registre modifiable par programme servant à pointer vers le début d'un bloc de 16 registres; ces différents registres peuvent être utilisés comme: accumulateur lors des opérations arithmétiques, ou mémoires sauvegardant certaines données très utiles d'un programme.

b-1 : Utilisation de l'espace de travail:

* registre utilisé comme opérande: Ce registre contient une donnée pour des opérations arithmétiques ou logiques.
* registre utilisé comme accumulateur: Ce registre garde les résultats intermédiaires d'une opération arithmétique.

Exemple:

A R_5, R_6 additionner le contenu de R_5 au contenu de R_6, résultat dans R_6.

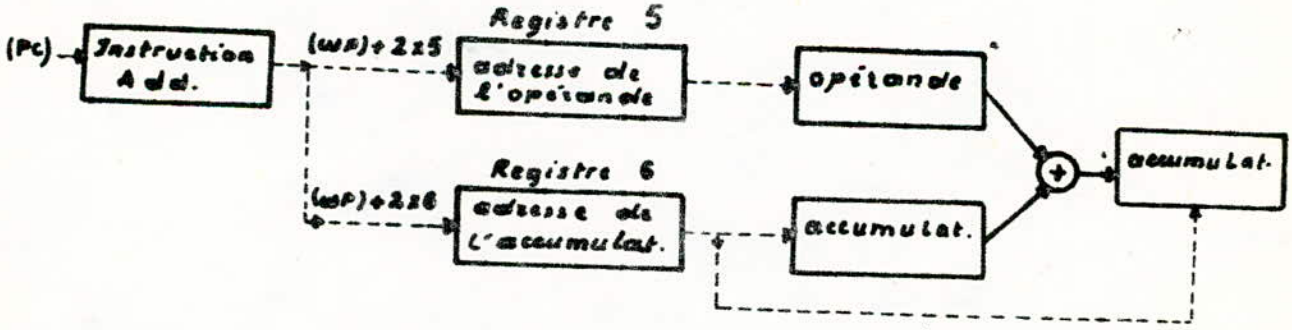


Le contenu de PC pointe vers l'instruction dans La ROM, le code de l'instruction indique les registres utilisés dans La RAM dont les adresses sont calculées automatiquement [(WP)+2R] pour localiser les données utilisées par l'instruction.

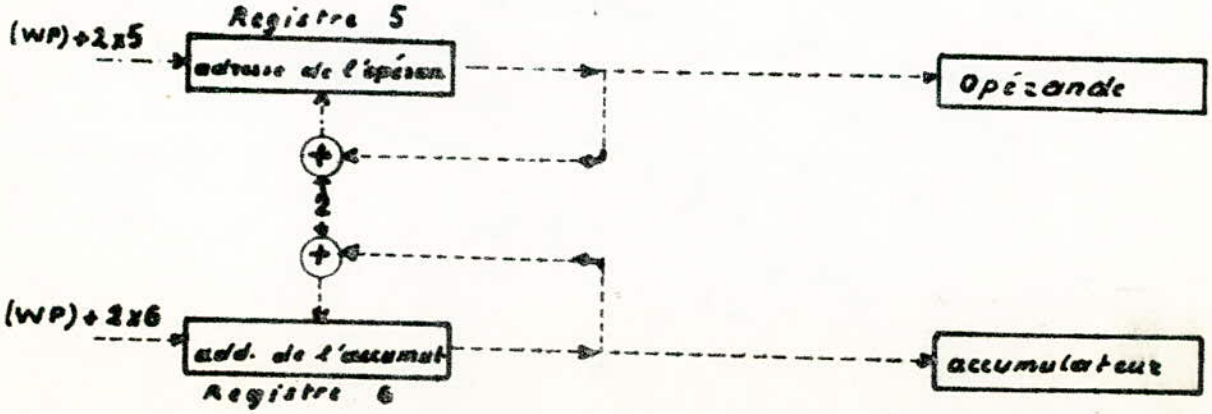
registre adresse : Ces registres sont utilisés pour spécifier l'adresse mémoire d'une opérande, ou l'adresse d'un accumulateur dans un programme. Ces registres sont accédés par un adressage indirect avec ou sans autoincrémentation; si l'autoincrémentation n'est pas utilisée, le contenu du registre de travail n'est pas modifié par l'opération; si elle est utilisée, l'adresse contenue dans le registre de travail est incrémentée automatiquement par 1 ou 2 selon l'instruction utilisée (octet ou mot).

Exemples:

1. A * R_5 , * R_6 additionner le contenu dont l'adresse est donnée par R_5 au contenu dont l'adresse est dans R_6, l'adresse du résultat est donnée par R_6



2. A * R5+ , * R6+

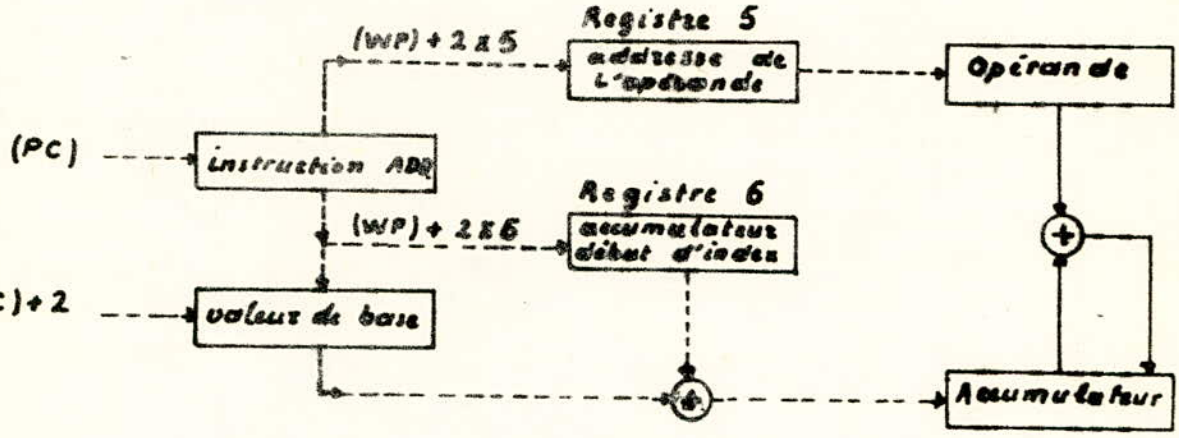


* registre utilisé comme registre index :

Quand un registre de travail est utilisé comme registre d'index, son contenu spécifié un début d'une adresse de base, la somme de ce début et l'adresse de base contenue dans l'instruction, définissent l'adresse mémoire de la donnée du programme. Seul le registre 0 ne peut être utilisé comme registre d'index.

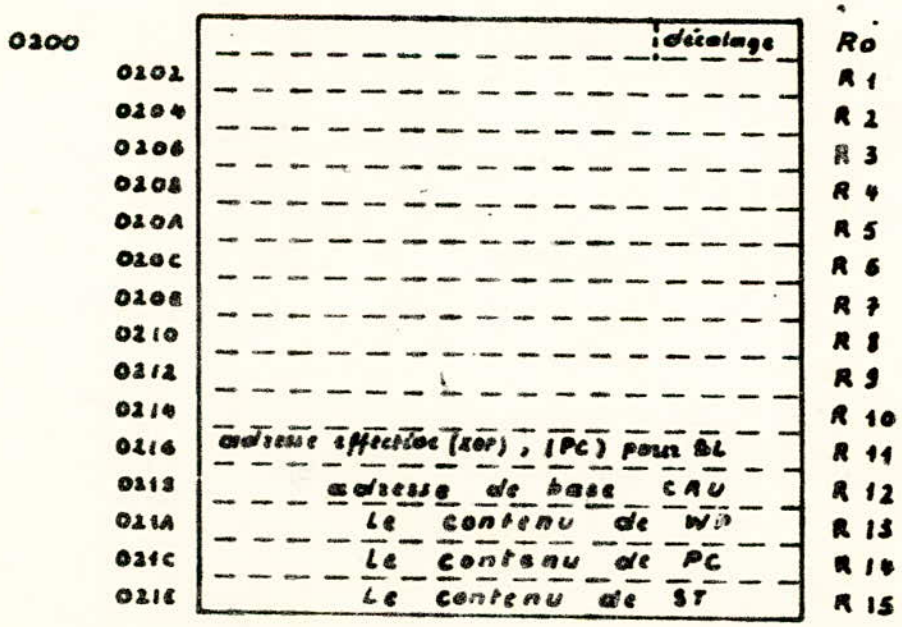
Exemple :

A *R5, @MN(R6)



b-2 : Zone d'espace de travail :

Le schéma suivant montre le bloc de 16 registres d'un espace de travail où le pointeur d'espace pointe vers l'adresse >0200



* registre 0 (R0) : les bits 12 à 15 de R0 indiquent le nombre de positions à décaler.

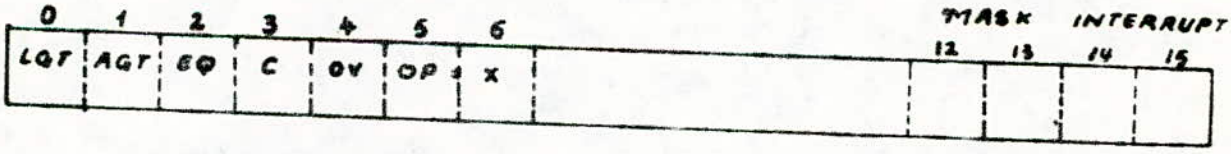
* Registre 11 (R_{11}) : Ce registre sauvegarde l'adresse de l'instruction suivant immédiatement BL (branchement avec chaînage) ou l'adresse source d'une instruction XOP. Le retour se fait à l'aide de l'instruction B *11

* Registre 12 (R_{12}) : Ce registre est utilisé pour l'adressage CRU.

* Registres 13, 14 et 15 (R_{13}, R_{14}, R_{15}) : Lors d'une éxécution d'une instruction de changement de contexte [BLWP, XOP, interruption] les contenus courants de WP, PC et ST sont rangés dans les registres 13, 14 et 15, respectivement, du nouvel espace de travail. L'adresse effective de l'opérande source d'une instruction XOP, est placée dans le registre 11 de cet espace, pour l'instruction BLWP, l'opérande source définie un vecteur de deux mots ; le 1er contient le nouveau (WP), le second le nouveau (PC).

C. Le registre d'état :

A la suite de l'éxécution d'une opération arithmétique ou logique, le CPU positionne d'une certaine façon les bits du registre d'état, conditionnant l'éxécution de la suite du programme en cours. Les 4 derniers bits de ce registre sont reservés au masquage des interruptions.



LGT	:	Supérieur à (logique)	[Logical Greater than]
AGT	:	Supérieur à (arithmétique)	[Arithmétic Greater than]
EQU	:	égalité	[Equal]
C	:	retenue	[Carry]
OV	:	dépacement	[overflow]
OP	:	parité impaire	[odd parity]
X	:	Opération étendue	[extended operation]

La programmation par segments :

D'une façon très classique, les programmes sont organisés en plusieurs segments, chacun d'eux assure l'exécution d'une tâche particulière au sein d'un programme principal d'application.

1. intérêt :

Cette méthode de programmation a plusieurs avantages :

* une occupation de mémoire réduite, un seul sous-programme peut être appelé à n'importe quel moment au lieu d'être répété maintes fois.
* une facilité de mise au point des sous programmes ; ces sous-programmes peuvent être facilement vérifiés et exécutés avant d'être insérés dans un programme principal.
* une grande possibilité de modifier les sous-programmes au lieu du programme tout entier.

2. caractéristiques des sous-programmes :

Les sous-programmes sont généralement caractérisés par :

* une taille limitée,
* Conçus pour une fonction bien précise.
* la possession des points d'entrées et de sorties

3. l'instruction BL (Branchement avec chaînage)

Cette instruction est utilisée pour faire appel à un sous-programme, l'adresse source de l'opérande fournit la valeur du nouveau PC alors que l'ancienne est sauvegardée dans R_{11} (voir figure [3-5]). le retour du sous-programme se fait par l'exécution d'une instruction de branchement indirect au registre 11 (B @11).

4. changement de contexte :

Un changement de contexte équivaut à un passage d'un environnement (WP, PC, ST) à un autre, Ce changement s'accompagne d'une modification des 3 registres internes (PC, WP, ST), en sauvegardant toutes les informations nécessaires pour le retour au programme interrompu (ancien environnement).

4-1 : utilité : lors d'un changement de contexte il n'est pas nécessaire de sauvegarder tous les registres de travail, ce qui permet un temps de réponse assez rapide.

4-2 : Origines d'un changement de contexte :

L'origine d'un changement de contexte peut être matérielle ou logicielle ; chacun d'eux met en œuvre un vecteur de deux mots servant à identifier l'adresse du programme (PC) et de l'espace de travail (WP) associés figure [3-6].

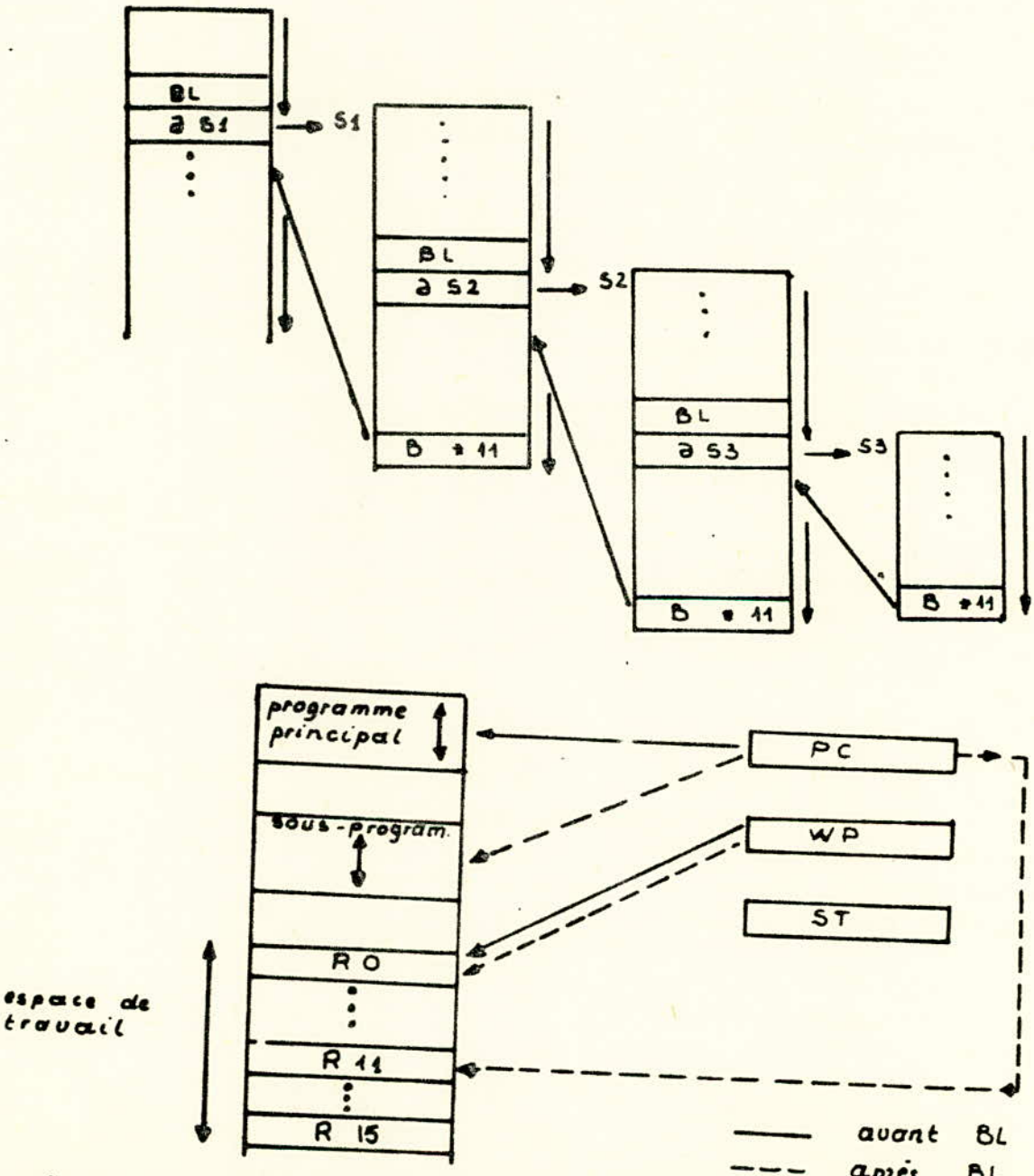
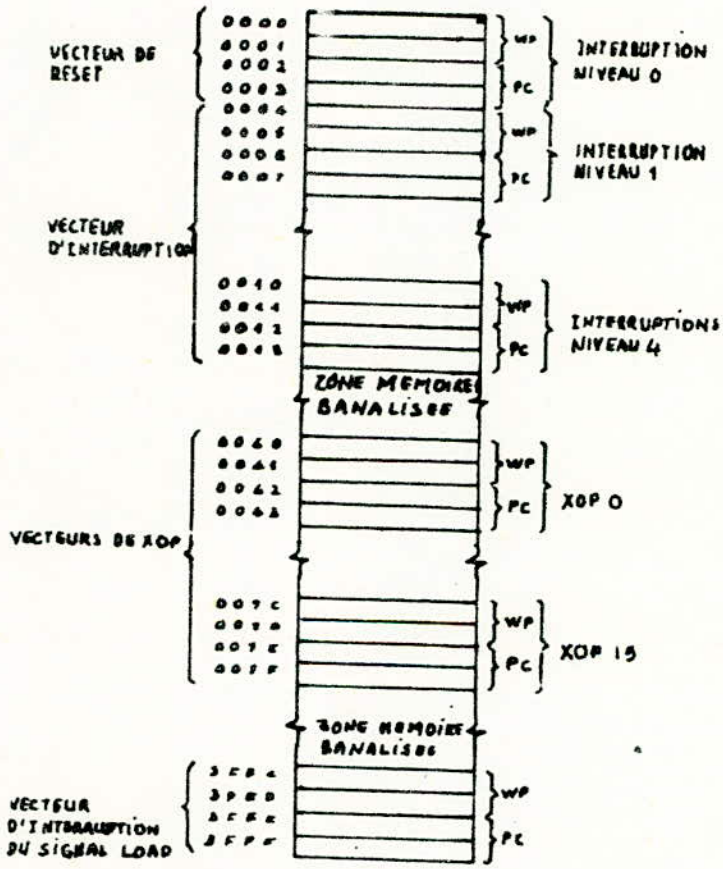


Fig 3.5 : Exécution de l'Instruction BL



MEMORY MAP ("CARTE DE LA MEMOIRE")

Fig 3.6

4-2-1: Changement de contexte d'origine matérielle :

Ce changement est provoqué par des interruptions externes qui exigent du processeur une réaction immédiate ; on a vu précédemment que la logique du TMS 9980A se sert de 3 entrées (IC_0, IC_1 et IC_2) pour identifier une demande d'interruption ; celle-ci est prise en compte si son niveau est inférieur ou égal au contenu des 4 bits (bits 12 à 15) du registre d'état (ST), alors un changement de contexte d'origine matérielle aura lieu.

4-2-2: Changement de contexte d'origine Logicielle :

Un changement de contexte d'origine logicielle se produit lorsqu'on exécute les instructions XOP et BLWP.

* XOP (opération étendue)

Le XOP admet deux opérandes ; la première est l'opérande source, la seconde est le Numéro du XOP. A ce dernier on associe un vecteur de 2 mots situés à des adresses mémoires bien définies fig [3-6], ces deux mots contiennent respectivement les nouveaux WP et PC. Les XOP n'ont aucun effet sur le masque d'interruption (bits 12 à 15 de ST) et ne sont affectées d'aucune priorité. La valeur définie par la première opérande du XOP est placée dans le registre 11 du nouvel espace de travail.

Fig 3-7

* BLWP

On associe à cette instruction (comme la précédente) un vecteur de 2 mots définis par l'opérande associée à celle-ci, et contenant les nouveaux WP et PC ; figure [38].

4-3: Restauration de l'environnement :

L'exécution de l'instruction RTWP permet la restauration de l'environnement d'un programme interrompu après un changement de contexte. Les contenus des registres 13, 14 et 15 sont transférés respectivement dans WP, PC et ST.

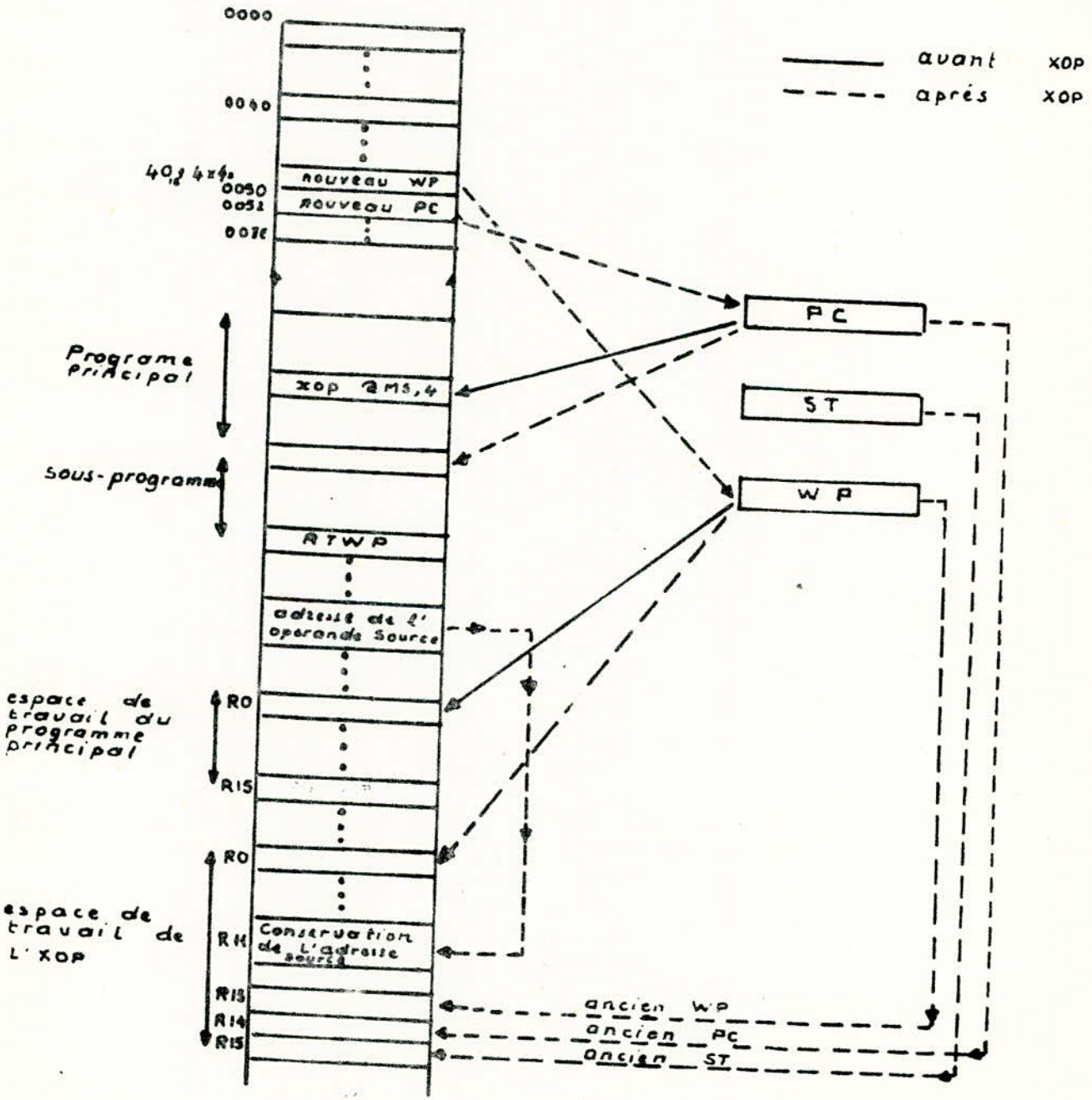


Fig 3-7: Exécution de L'instruction XOP (XOP @MS,4)

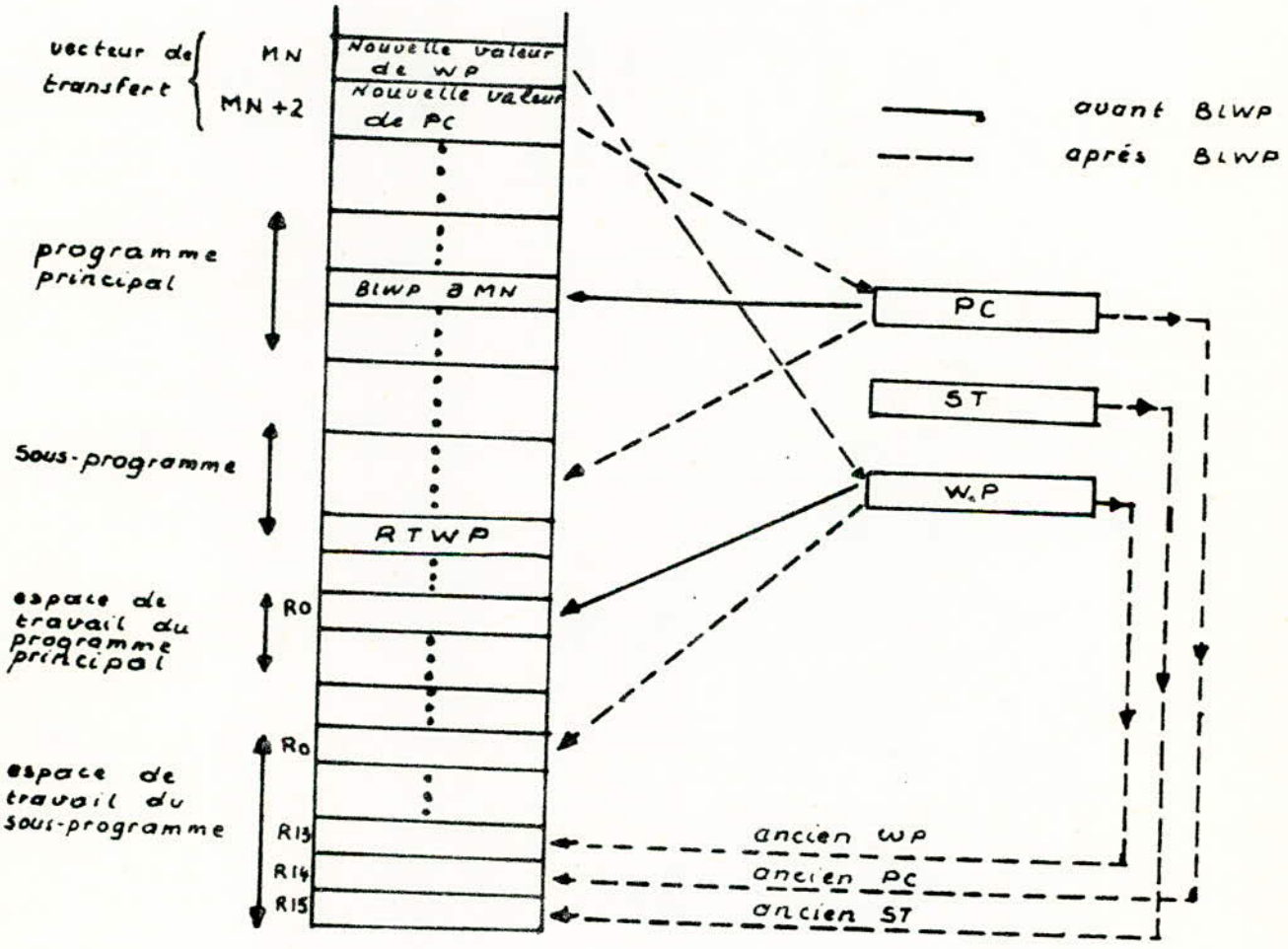
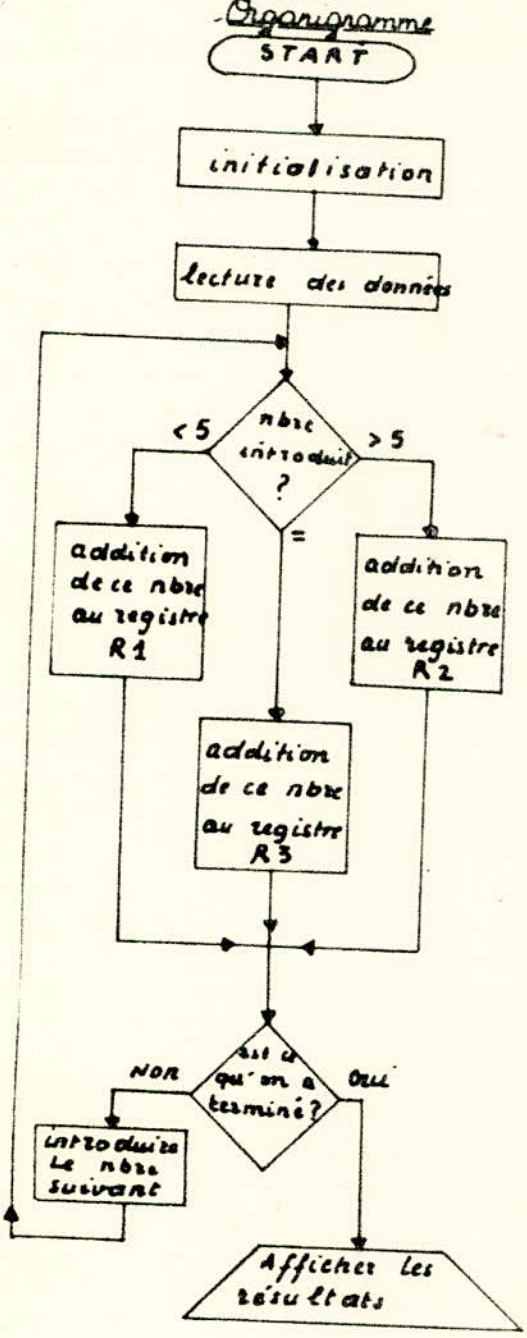


Fig 3-8 : Exécution de l'instruction BLWP (BLWP @MN)

C. Applications

Exemple 1 :

Organigramme



	programme		Commentaires
GO	LWPI	>0300	chargement de WP
	LI	R0,9	chargement immédiat de R0
	CLR	R1	mise à zéro des registres
	CLR	R2	R_1 , R_2 , R_3
	CLR	R3	
	LI	R4, NU	Chargement de R_4 avec l'adresse
LP	MOV	*R4+, R5	de la suite des données
			transferer le contenu de R3 avec autoincrementation dans R5
	CI	R5, 5	Comparer immédiatement le Contenu de R5 à 5
	JL	LS	Saut si inferieur
	JEQ	LN	Saut s'il y a égalité
	A	R5, R2	> 5 , additionner R5 a R2
	JMP	LA	Saut inconditionnel (introduire le nbre suivant?)
LS	A	R5, R1	<5 , additionner R5 a R1
	JMP	LA	introduire le nbre suivant ?
LN	A	R5, R3	= 5 , additionner R5 à R3
LA	DEC	R0	decrementer le compteur
	JGT	LP	la liste n'estpas finie
	XOP	1, 10	visualiser le contenu de R1
	XOP	1, 13	alimenter la visualisation
	XOP	2, 10	visualiser le nontenu de R2
	XOP	1, 13	alimenter la visualisation
	XOP	3, 10	visualiser le contenu de R3
	XOP	1, 13	alimenter la visualisation
NU	DATA	3 , 5 , 2 , 7 , 5 , 8 , 0 , 3 , 9	

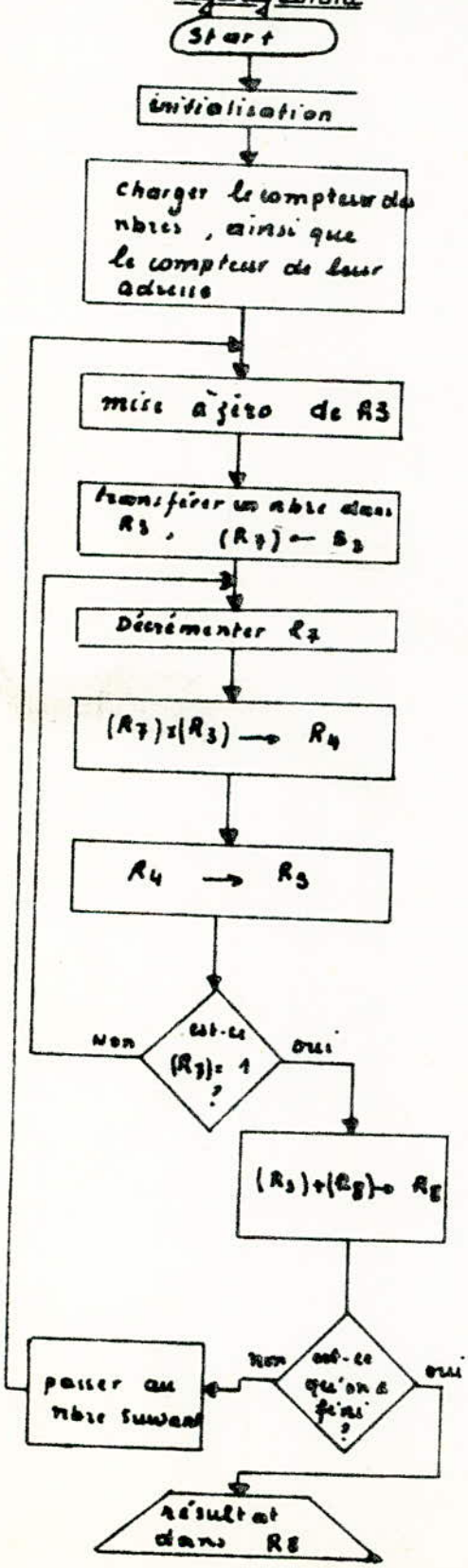
addition des nbres < 5 résultat dans R1
= = 5 résultat dans R3
> > 5 résultat dans R2

Les contenus du compteur R0 et de DATA peuvent être changés à volonté.

Exemple 2

Calcul de la somme des factorielles de N nbres.

Organigramme

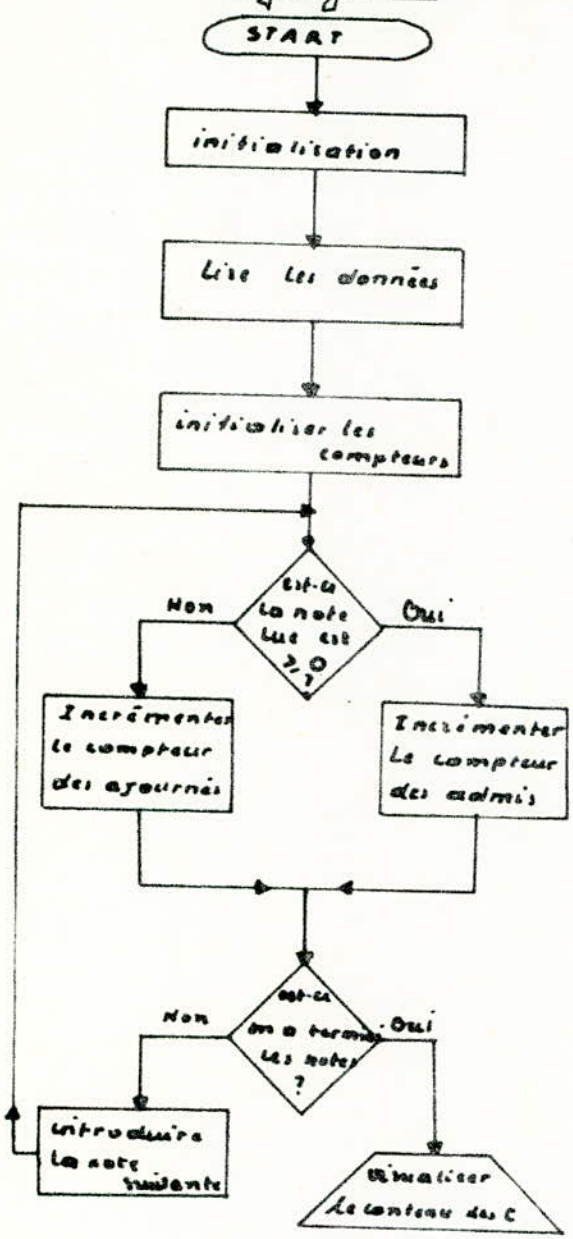


Programme

```
GO    LWPI  >0300
      LI    R1, N       Compteur des nbres.
      LI    R2, MU      Compteur de leur adresse
      CLR   R8
LT    CLR   R3
      MOV   *R2, R3     transfert des nbre N
      MOV   R3, R7      Sauvegarder ce nbre dans R3
LP    DEC   R7          N-1 → R7
      MPY   R7, R3      N.(N-1) → R4 (petits nbres)
      MOV   R4, R3      (R4) → R3
      CI    R7, 1       N-1 = 1 ?
      JEQ   LA          oui sauter à l'adresse Symbolique LA
      JMP   LP          Non : Sauter à l'adresse Symbolique LP
LA    A     R3, R8      additionner le contenu
      CI    R1, 1       de R3 à l'accumulateur R8
      JEQ   LB          Si on a fini, sauter
      DEC   R1          Si non, passer au
      INCT  R2          nbre suivant
      JMP   LT
LB    XOP   8, 10       visualiser le résultat
      XOP   4, 13       alimenter la visualisation
MU    DATA  . . .       données du programme
      END   GO
```

Exemple 3 :

Organigramme



programme

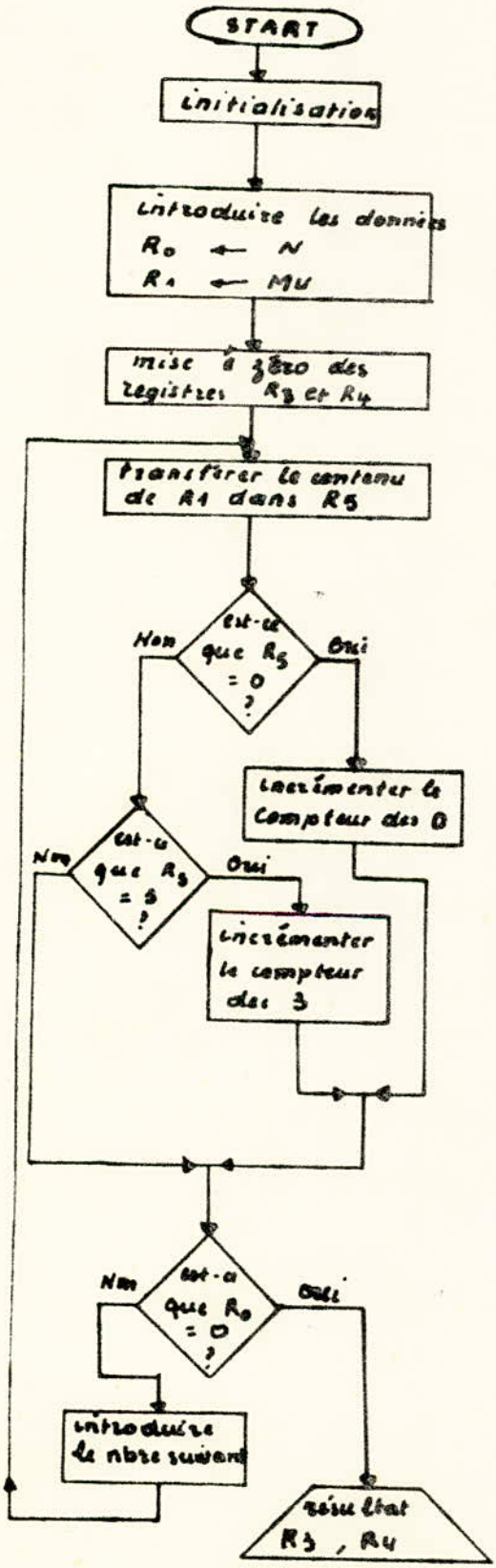
Commentaires

```
GO   LWPI  >0300
     CLR   R0        compteur des ajournés
     CLR   R1        Compteur des admis.
     LI    R2,       Compteur des notes
     LI    R3,10     valeur de comparaison
     LI    R4,NU     Compteur des adresses
LP   MOV   *R4+,R5   Un nombre dans R5
     S     R3, R5
     JGT   LT        nbre ≥10
     JMP   LA        non
LT   INC   R1        incrémenter le compteur
     JMP   LD        des admis

LA   INC   R0        incrémenter le compteur
                     des ajournés
LD   DEC   R2
                     passer à la note
     JGT   LP        suivante

     XOP   1,10      visualiser le contenu de
     XOP   1,13      R1 et alimenter la visu-
     XOP   0,10      visualiser le contenu de R0
     XOP   1,13      alimenter la visualisation

NU   DATA  ...       données du
                     programme (notes)
     END   GO
```

On donne les notes de 30 étudiants (notes/20), et on veut afficher le nombre des admis et des ajournés.

Exemple 4 :

Organigramme

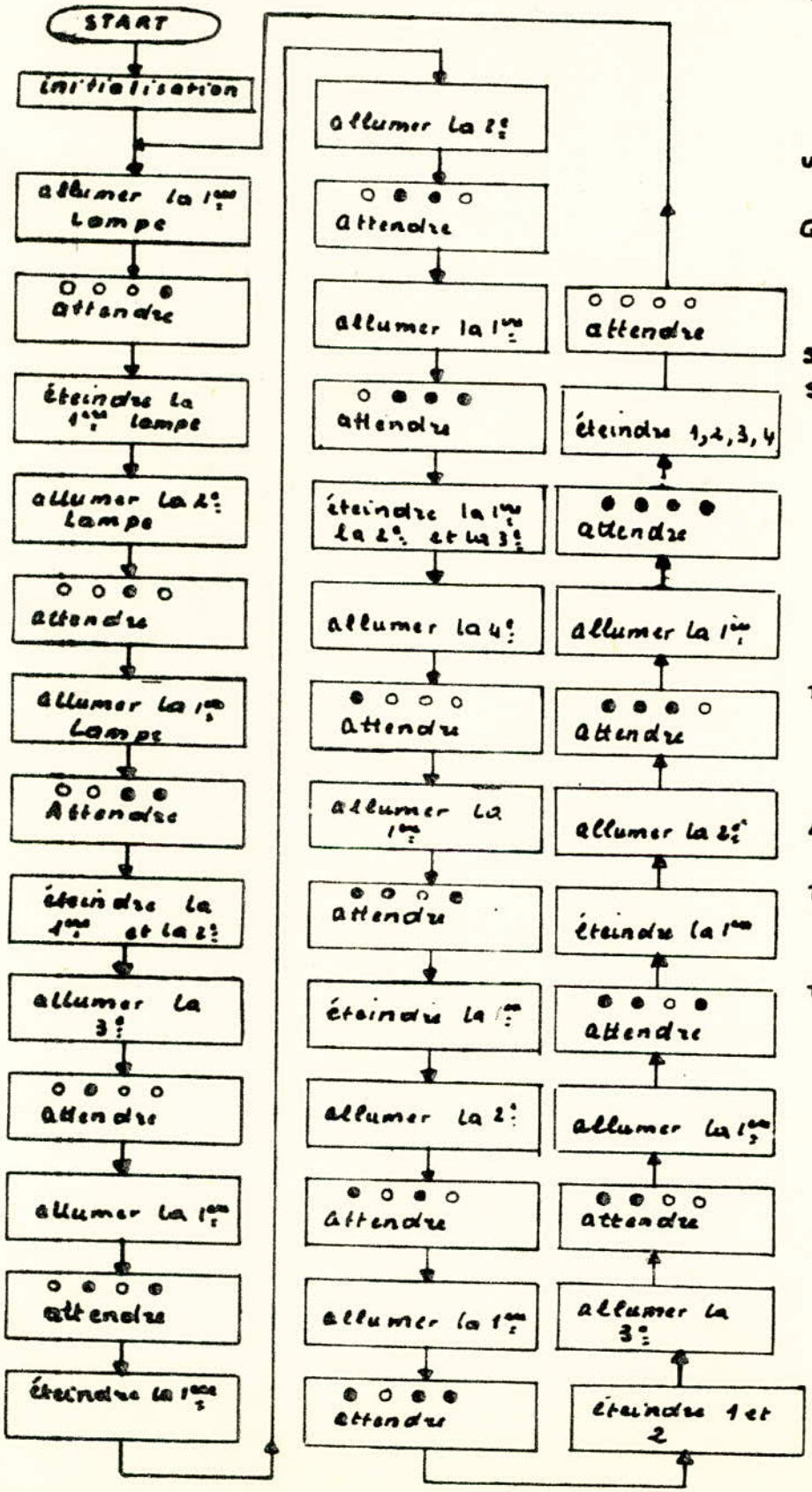


	programme		Commentaires
GO	LWPI	>0300	chargement du WP
	LI	R0, N	chargement du compteur
	LI	R1, MU	Chargement de R1 avec l'adresse de la liste des nombres
	CLR	R3	mise à zéro des registres
	CLR	R4	R3 et R4
LP	MOV	*R1, R5	transférer les nombres dans R5
	CI	R5, 0	est-ce que ce nbre est égal à 0 ?
	JEQ	LA	Oui
	CI	R5, 3	non ; est-ce que ce nbre est égal à 3?
	JEQ	LB	Oui
	JMP	LC	non
LA	INC	R3	incrémenter le compteur des 0
	JMP	LC	
LB	INC	R4	incrémenter le compteur des 3
LC	CI	R0, 0	est-ce qu'on a fini ?
	JEQ	LD	Oui
	DEC	R0	non
	INCT	R1	passons au nbre suivant
	JMP	LP	
LD	XOP	3, 10	visualiser le contenu de R3
	XOP	1, 13	alimenter la visualisation
	XOP	4, 10	visualiser le contenu de R4
	XOP	1, 13	alimenter la visualisation
MU	DATA		
	END	GO	

Compter le nombre des zéros et des trois dans une liste de N nombres.

Exemple 5 : Compteur binaire

Organigramme

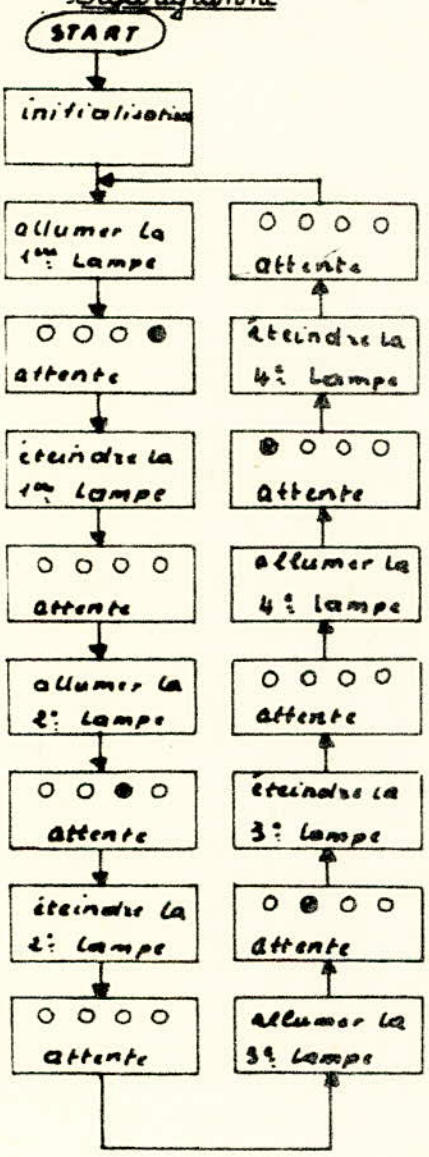


Programme

```
WS   BSS 32

GO   LWPI WS
     LI  R12,>20

S1   CLR R0
S2   CLR R1
     CI  R0,16
     JEQ S1
     SWPB R0
     XOR R0,R1
     LDCR R1,4
     JEQ AA
     LI  R2,>FFFF
                 (modifiable)
T2   DEC R2
     JNE T2
     JMP T4

AA   LI  R3,>FFFF
                 (modifiable)
T3   DEC R3
     JNE T3

T4   SWPB R0
     INC R0
     JMP S2

     END GO
```

Exemple 6 :

Organigramme



programme

```
GO    LWPI  >0300
      LI    R1,WS
      LI    R12,>0020

ST    SBO   0
      BL    *R1
      SBZ   0
      BL    *R1
      SBO   1
      BL    *R1
      SBZ   1
      BL    *R1
      SBO   2
      BL    *R1
      SBZ   2
      BL    *R1
      SBO   3
      BL    *R1
      SBZ   3
      BL    *R1
      JMP   ST
WS    TB    4
      JEQ   T0
      LI    R3,>3FFF   (modifiable)
T1    DEC   R3
      JNE   T1
      B     *11
T0    LI    R3,>3FFF   (modifiable)
T2    DEC   R3
      JNE   T2
      B     *11
      END   GO
```

Application : indicateur de position d'un ascenseur.

CHAPITRE III

A. TIMING (temps d'exécution des instructions du TMS 9980 A):

Le temps d'exécution d'une instruction du TMS 9980A est fonction de :

1) La période de l'horloge $t_c(\phi)$
2) Le mode d'adressage utilisé
3) Le nbre de temps d'attente requis par chaque accès mémoire

La figure [4-1] donne le nombre de périodes d'Horloge et des accès mémoires nécessaires à l'exécution de chaque instruction du TMS 9980A. Pour les instructions possédant un mode d'adressage pour chacun des deux opérandes, la même figure fournit ces paramètres dans le cas d'un adressage direct par registre pour les deux opérandes. Le Calcul des périodes et accès mémoires supplémentaires se fait en additionnant les valeurs données par les tables correspondantes A ou B.

Le temps total d'une instruction est donné par la formule:

$$T = t_c(\phi)\ [C + W.M] \qquad (1)$$

avec :

- T : durée totale d'exécution.
- $t_c(\phi)$: période d'Horloge.
- C : nbre de périodes d'Horloge nécessaires à l'exécution et la modification d'adresse.
- W : nombre de temps d'attentes nécessaires à chaque accès mémoire pour l'exécution de l'instruction et la modification d'adresse.
- M : nombre d'accès mémoire.

pour une fréquence de 2 MHz, la période d'Horloge est

$$t_c(\phi) = \frac{10^{-6}}{f(\text{MHz})} = \frac{1}{2.10^{+6}} = 0,5\ \mu s$$

exemples Calculons la durée d'exécution des instructions

a) A R0, R1 l'instruction utilise un mode d'adressage direct par registre pour les deux opérandes. Aucun temps d'attente n'étant introduit lors de l'adressage de la mémoire, en utilisant la relation [(1)]

On trouve :

$$T = 0,5\ (22 + 0 \times 8) = 11\ \mu s$$

Si l'on introduit deux temps d'attente par accès mémoire, le temps d'exécution devient :

$$T = 0,5\ (22 + 2 \times 8) = 19\ \mu s$$

b) A *R0, *R1

Dans ce cas les 2 opérandes utilisent le mode d'adressage indirect par registre. Si l'on introduit deux temps d'attente par accès mémoire, en utilisant les tables correspondantes et la relation précédente on trouve :

$$C = 6 + 22 + 6 = 34$$
$$M = 2 + 8 + 2 = 12$$

d'où $T = t_c(\Phi)\ [C + W.M] = 0,5\ (34 + 2 \times 12) = 29\ \mu s$

c) <u>Synthétiseur de fréquence</u> :

L'examen des boucles d'émission (SB0 — JMP A1) et d'arrêt d'émission (SB2 — JMP D1) montre la façon dont la temporisation est obtenue. Comme on a précisé précédemment, chaque instruction s'exécute en un temps fini déterminé par certains paramètres : (nombre de Cycles d'Horloge, mode d'adressage, nombre d'accès mémoire, temps d'attente et la fréquence d'Horloge elle-même).

$$T = t_c(\Phi)\ [C + W.M]$$

Boucle d'émission

Instr. \ Temps	$t_c(\Phi)$ µs	C	W	M	T µs
SB0	0,5	16	0	4	8
MOV	0,5	22	0	8	11
DEC	0,5	16	0	6	8
JEQ	0,5	10	0	2	5
SRC	0,5	24	0	8	12
JMP	0,5	12	0	2	6

Boucle d'arrêt d'émission

Instr. \ Temps	$t_c(\Phi)$ µs	C	W	M	T µs
SB2	0,5	16	0	4	8
MOV	0,5	22	0	8	11
DEC	0,5	16	0	6	8
JEQ	0,5	10	0	2	5
SRC	0,5	24	0	8	12
JMP	0,5	12	0	2	6

A - conditions normales

Instr. \ Param.	$t_c(\phi)$ µs	C	W	M	T µs
SBO	0,5	16	0	4	8
MOV	0,5	22	0	8	11
DEC	0,5	16	0	6	8
JEQ	0,5	12	0	2	6
SRC	0,5	24	0	8	12
JMP	0,5	12	0	2	6

Instr. \ Param.	$t_c(\phi)$ µs	C	W	M	T µs
SBZ	0,5	16	0	4	8
MOV	0,5	22	0	8	11
DEC	0,5	16	0	6	8
JEQ	0,5	12	0	2	6
SRC	0,5	16	0	8	8
JMP	0,5	12	0	2	6

* La conception de la carte TM 990/189 n'introduit aucun temps d'attente d'accès mémoire (W = 0)

R_0 et R_1 contiennent respectivement le nombre de temps élémentaires de 50 µs d'émission et d'arrêt d'émission. Dans la majorité des cas (tableau [A. cond. normales]) le cycle d'émission est égal au cycle d'arrêt d'émission, d'où la période:

$$T = 50 + 50 = 100\ \mu s = 10^{-4}\ s$$

pour une fréquence (f) donnée, par exemple 1KHz, on a un nombre (x) de périodes de 10^{-4} s d'où:

$$10^{-4} \times x = \frac{1}{f(Hz)} \Rightarrow \quad x = 10$$

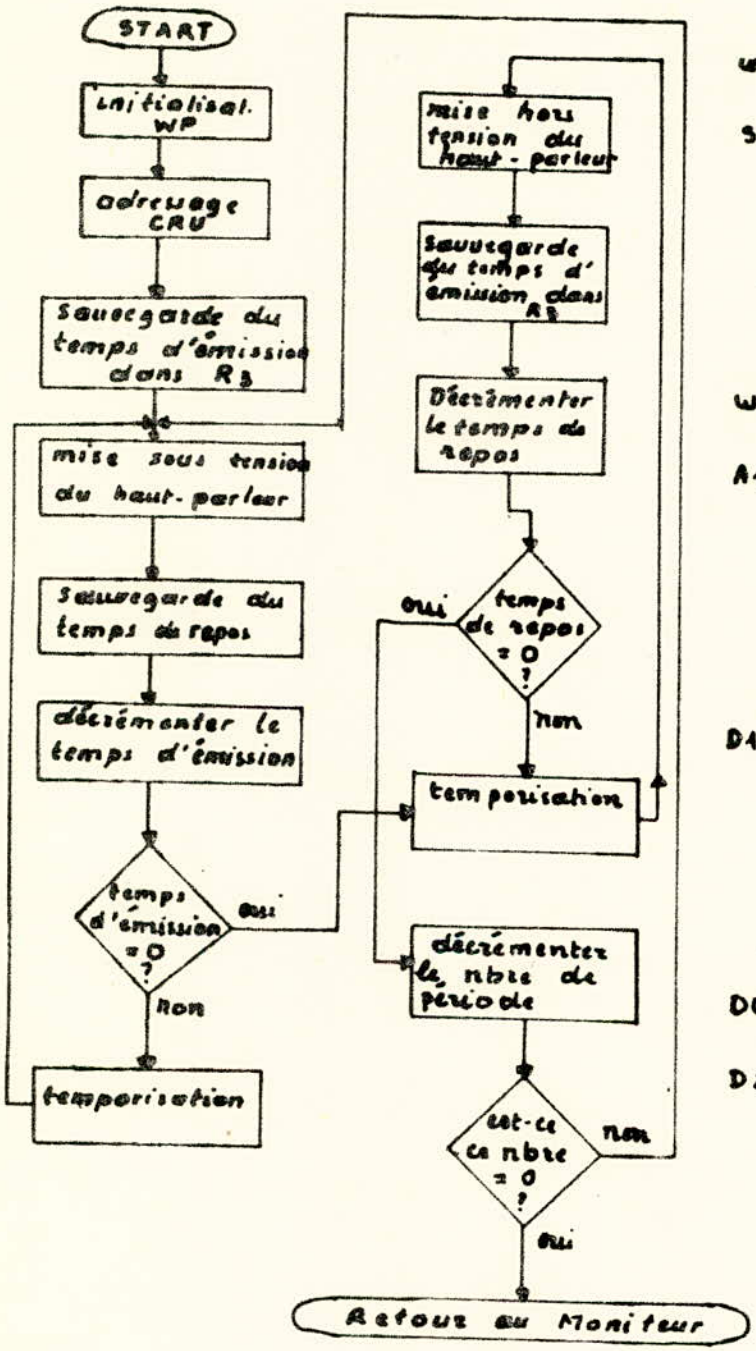
donc R_0 et R_1 contiennent (chacun) la valeur 10 pour une fréquence de 1KHz et pour une fréquence (f) quelconque :

$$x = \frac{10^4}{f(Hz)}$$

R_2 contient le nombre de cycle:
pour une durée d'émission t_s, pour une fréquence f on a un nombre de cycles y
d'où $y \times \frac{1}{f} = t_s \Rightarrow y = f t$

Si f = 1KHz et t_s = 1 s on a $y = 1000$

donc R_2 contient 1000




```
WS   BSS   32          zône réservée à l'espace
                       de travail
ST   LWPI  WS          initialiser WP
     LI    R0,10       durée de l'émission
     LI    R1,10       durée de l'arrêt
                       d'émission
     LI    R2,>2E8     nombres de cycles
     LI    R12,>43C    initialiser l'adresse
                       de base CRU
WC   MOV   R0,R3       sauvegarder la durée
                       d'émission
A1   SBO   0           mettre le haut parleur
     MOV   R1,R4       Sous tension
                       sauvegarder d'arrêt
                       d'émission ^ la durée
     DEC   R3          décrémenter la durée
                       d'émission
     JEQ   D0          a-t-on fini d'émettre?
     SRC   R3,3        non on temporise
     JMP   A1
D1   SBZ   0           Couper le haut-parleur
     MOV   R0,R3       Sauvegarder la durée
                       d'émission
     DEC   R4          décrémenter la durée
                       d'arrêt d'émission
     JEQ   D2          a-t-on épuisé la durée
                       de repos ?
D0   SRC   R3,3        temporisation
     JMP   D1
D2   DEC   R2          décrémenter le
                       Compteur de Cycles
     JNE   A1
     B     @>3000      On rend le
                       Contrôle au moniteur
     END   ST
```

Organigramme et programme de Synthétiseur de fréquence.

Fig 4-1 : Temps d'exécution des instructions du TMS 9980 A

Instruction	CLOCK CYCLES C	MEMORY ACCESS M	adres modification... SOURCE	Destination
A	22	8	A	A
AB	22	8	B	B
ABS (MSB = 0)	16	4	A	-
(MSB = 1)	20	6	A	-
AI	22	8	-	-
ANDI	22	8	-	-
B	12	4	A	-
BL	18	6	A	-
BLWP	38	12	A	-
C	20	6	A	A
CB	20	6	B	B
CI	20	6	-	-
CKOF	14	2	-	-
CKON	14	2	-	-
CLR	16	6	A	-
COC	20	6	A	-
CZC	20	6	A	-
DEC	16	6	A	-
DECT	16	6	A	-
DIV (ST_4 is set)	22	6	A	-
DIV (ST_4 is reset)	104 - 136	12	A	-
IDLE	14	2	-	-
INC	16	6	A	-
INCT	16	6	A	-
INV	16	6	A	-
Jump (PC is changed)	12	2	-	-
(PC is not changed)	10	2	-	-
LDCR (C = 0)	58	6	A	A
(1 ≤ C ≤ 8)	26 + 2C	6	B	B
(9 ≤ C ≤ 15)	26 + 2C	6	A	-
LI	18	6	-	-
LIMI	22	6	-	-
LREX	14	2	-	-
LWPI	14	4	-	-
MOV	22	8	A	A
MOVB	22	8	B	B
MPY	62	10	A	-
NEG	18	6	A	-
ORI	22	8	-	-
RSET	14	2	-	-

Fig 4.1 (suite)

Instruction	clock cycles C	MEMORY ACCESS M	adress modification Source	Destination
RTWP	22	8	-	-
S	22	8	A	A
SB	22	8	B	B
SBO	16	4	-	-
SBZ	16	4	-	-
SETO	16	6	A	-
Shift (C ≠ 0)	48 + 2C	6	-	-
(C≠0, Bits 12-15 of WR_0 = 0)	60	8	-	-
(C = 0, Bits 12-15 of WRP = N ≠ 0)	28 + 2N	8	-	-
SOC	22	8	A	A
SOCB	22	8	B	B
STCR (C = 0)	68	8	A	-
(1 ≤ C ≤ 7)	50	8	B	-
(C = 8)	52	8	B	-
(9 ≤ C ≤ 15)	66	8	A	-
STST	12	4	-	-
STWP	12	4	-	-
SWPB	16	6	A	-
SZC	22	8	A	A
TB	16	4	-	-
X	12	4	A	-
XOP	52	16	A	-
XOR	22	8	A	-
SZCB	22	8	B	B
RESET Function	36	10	-	-
LOAD Function	32	10	-	-
Interrupt context switch	32	10	-	-
Undefined op codes 0000-01FF, 0820 03FF, 0C00-0FFF 0780-07FF	8	2	-	-

A

Mode d'adressage	pénalités C d'attentes	Accès Mém. M
WR (TS ou TD = 00)	0	0
WR (TS ou TD = 01)	6	2
WR (TS ou TD = 11)	12	4
(TS ou TD = 10, S ou D = 0)	10	2
(TS ou TD = 10, S ou D ≠ 0)	12	4

B

Mode d'adressage	pénalités C d'attentes	Accès Mém. M
WR (TS ou TD = 00)	0	0
WR (TS ou TD = 01)	6	2
WR (TS ou TD = 11)	A0	4
(TS ou TD = 10, S ou D = 0)	A0	2
(TS ou TD = 10, S ou D ≠ 0)	A2	4

Applications :

- Initiation à la musique :

La présence d'un haut parleur permet de programmer certains morceaux de musique connus, Ces programmes présentent un aspect plus attrayant que les réalisations habituelles, l'appareil se chargera de déchiffrer la partition choisie et un changement de données seulement permet de changer de partition.

Règles élémentaires de musique :

a) hauteur du son : la figure [4-2] montre la hauteur des notes sur une partie précédée par une clef de SOL

fig 4-2

DO RE MI FA SOL LA SI ↑DO ↑RE ↑MI ↑FA ↑SOL ↑LA

- notes de base :

On distingue Sept notes de base, les autres notes sont obtenues à partir des 7 premières, soit en multipliant successivement par deux la fréquence de la note obtenue, soit en divisant successivement par deux la fréquence de cette note (paragraphe 9)

b) durée d'une note : le tableau suivant donne la durée d'une note par rapport à une période choisie à volonté réglable par programme.

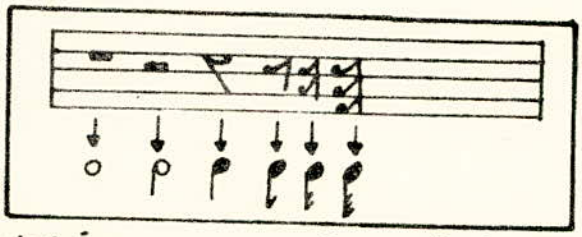
o : ronde	♩ : noire	𝅘𝅥𝅯 : double croche
𝅗𝅥 : blanche	♪ : croche	𝅘𝅥𝅰 : triple croche

o = 2 𝅗𝅥 = 4 ♩ = 8 ♪ = 16 𝅘𝅥𝅯 = 32 𝅘𝅥𝅰

c) dièse(♯) et bémol(♭) : un nombre de dièses ou bémols associé à la clef de Sol permet de transposer les notes de musique suivant le tableau suivant :

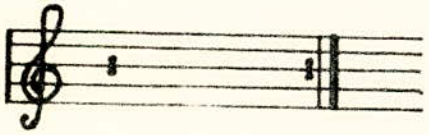
nbre de # à la clef de Sol	nbre de b à La clef de Sol	note à considérer comme DO
0	0	DO
1	6	Sol
2	5	RE
3	4	LA
4	3	MI
5	2	SI
6	1	FA

d) Silence (longueur de silence) : le tableau suivant donne la durée relative à une période choisie à volonté par programme.

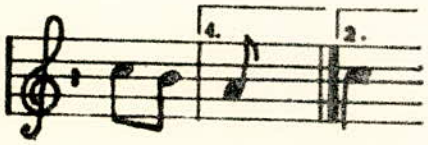


e) notes pointées : une note pointée à sa droite voit sa durée augmentée de sa demi-valeur.
exemple :
noire pointée = 3 crochets
Cette règle reste également applicable aux notes de silence.

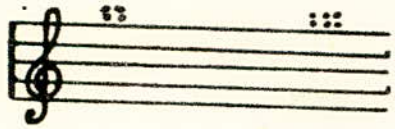
f) Sigles et notations :



① Passage à reprendre deux fois de Signe à Signe, quand il y a deux lignes de paroles superposées dans la même langue et sans indications spéciales, on chante la deuxième fois la deuxième ligne.

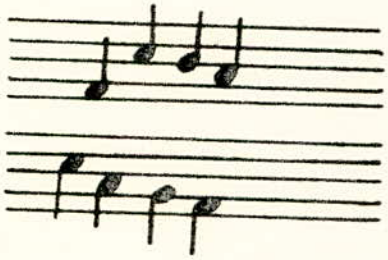


② à la deuxième reprise on saute le passage entre 1 et 2



③ passages particuliers à reprendre de Signe à Signe.

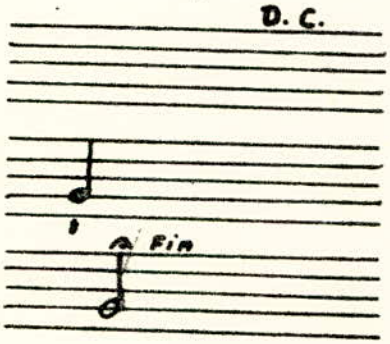
④ note ou suite de notes représentant simultanément un passage qui est répété avec une variante mélodique



⑤ 1ère fois

très rarement cela représente deux voies simultanées

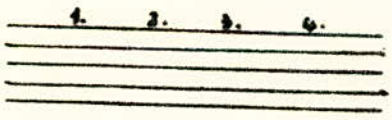
⑥ 2e fois



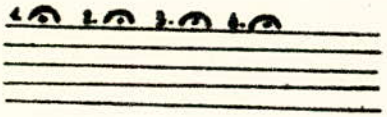
⑦ DA CAPO (reprenez " depuis le début")

AL Fin (" jusqu'à la fin ")

⑧ note à chanter à l'octave inférieur

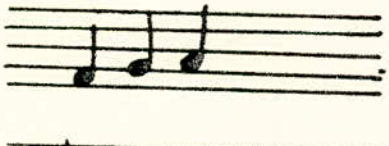


⑨ Quand la chanson ne finit pas sur la dernière note de musique, le mot fin et (ou) un point d'orgue (𝄐) l'indiquent en cours de route

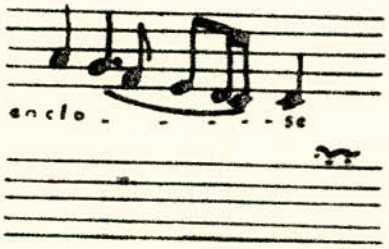


⑩ Canons : départ des voies successives. (dans le canon DA PACEM à 4 voies mixtes, on a noté en petit les notes de départ dans les différentes clés).

⑪ Canons : fin simultanée pour chaque voix (ex : si on arrête la voix 1 à cet endroit, les autres s'arrêteront à leurs chiffres respectifs simultanément)



⑫ Syllabes à replacer sous leurs notes



⑬ notes liées : à chanter sur la même syllabe.

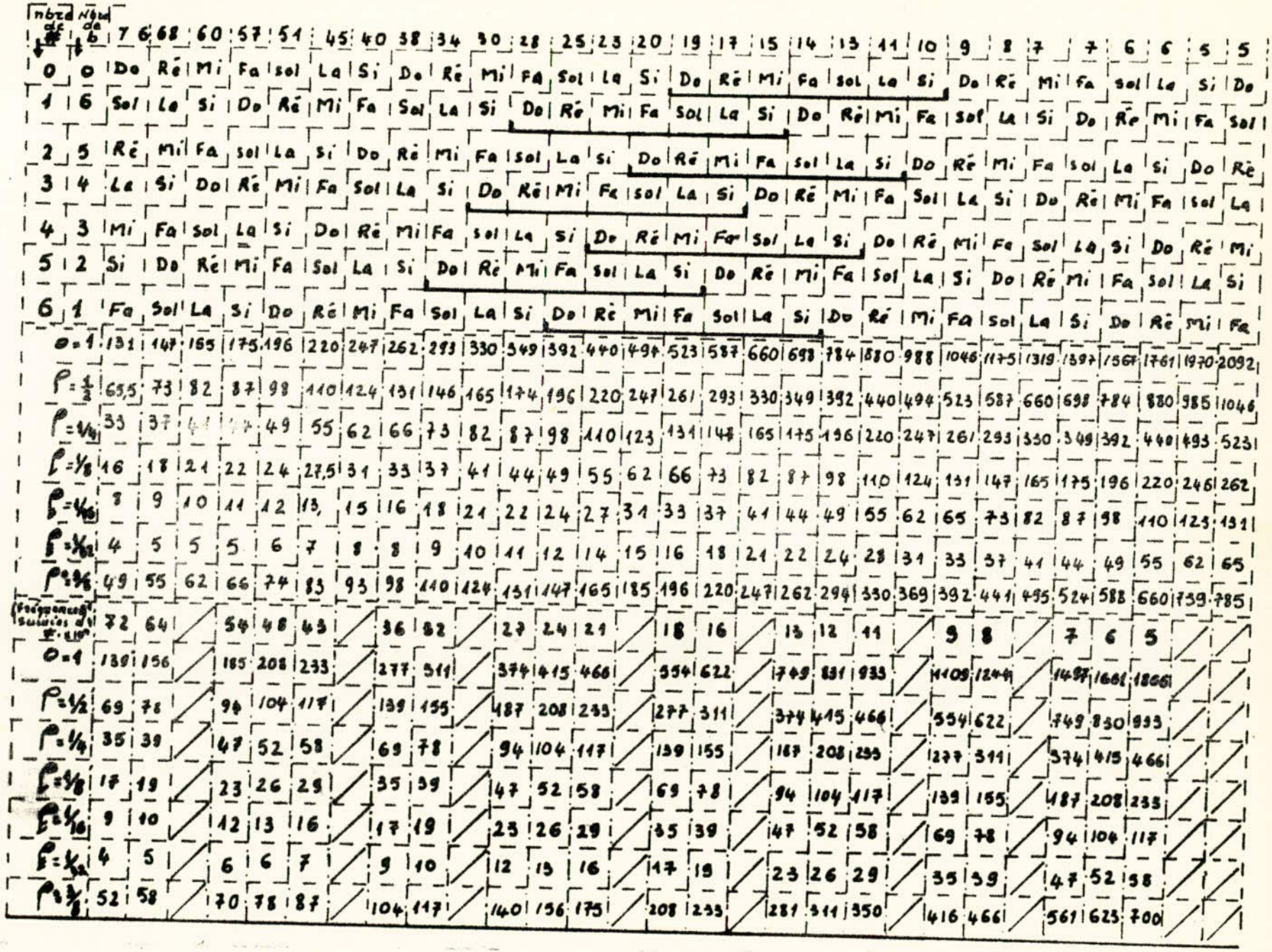
⑭ Signale les chansons drôles, rengaines humoristiques.

9) **Fréquences** : Le tableau suivant donne la correspondance notes - fréquences.

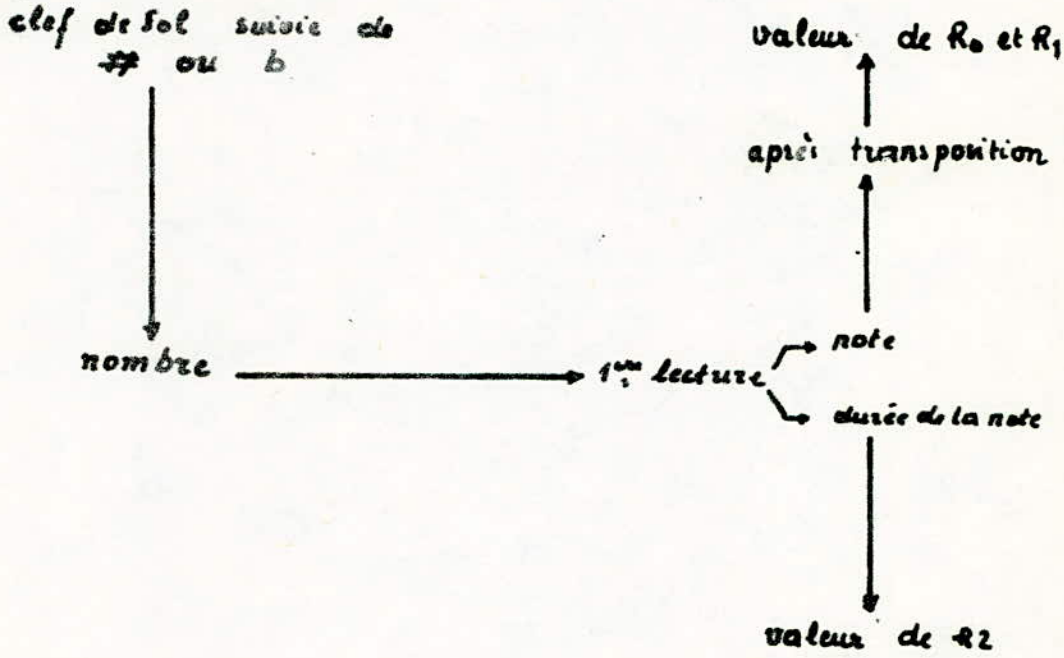
notes	fréquence (Hz)
DO 7	2092,0
Si 6	1970,0
LA ♯6	1865,6
La 6	1760,5
Sol ♯6	1661,1
Sol 6	1567,4
Fa ♯6	1497,3
Fa 6	1396,6
Mi 6	1319,2
Ré ♯6	1248,3
Ré 6	1173,7
Do ♯6	1108,6
Do 6	1046,0
Si 5	988,1
La ♯5	932,3
La 5	880,2
Sol ♯5	830,5
Sol 5	783,7
Fa ♯5	739,7
Fa 5	698,3
Mi 5	659,6
Re ♯5	624,9
Ré 5	586,8
Do ♯5	554,3
Do 5	523
Si 4	494
La ♯4	466
La 4	440

notes	fréquence (Hz)
Sol ♯4	415
Sol 4	392
Fa ♯4	370
Fa 4	349
Mi 4	330
Re ♯4	311
Ré 4	293
Do ♯4	277
Do 4	262
Si 3	247
La ♯3	233
La 3	220
Sol ♯3	208
Sol 3	196
Fa ♯3	185
Fa 3	175
Mi 3	165
Re ♯3	156
Ré 3	147
Do ♯3	139
Do 3	131
Si 2	123
La ♯2	117
La 2	110
Sol ♯2	104
Sol 2	98
Fa ♯2	92
Fa 2	87

A) tableau donnant la note et les valeurs correspondantes à R_0 et R_2.



• **Utilisation du tableau (R_2)**



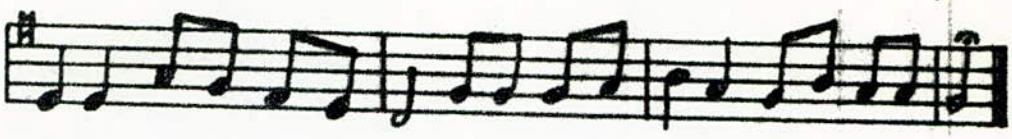
Remarque:

Lors de l'exécution du programme Synthétiseur de fréquence, le volume du son n'est pas contrôlé.

Application : programmation

Exemple 1 : "Au clair de la lune"

1ère Lecture = :	Sol	sol	sol	La	Si	La	sol	Si	La	La	Sol :	La	La		La	La		
Après transposition = :	Do	Do	Do	Ré	Mi	Ré	Do	Mi	Ré	Ré	Do :	Ré	Ré		Ré	Ré		
Durée t = :	1/8	1/8	1/8	1/8	1/4	1/4	1/8	1/8	1/8	1/8	1/2 :	1/8	1/8		1/8	1/8		
$(R_0)=(R_0)=\frac{10^9}{f}$ = :	38	38	38	34	30	34	38	30	34	34	38 :	34	34		34	34		
$(R_2)=tf$ = :	33	33	38	37	62	73	33	41	37	37	131 :	37	37		37	37		



1ère Lecture =	Mi	Mi	La	Sol	Fa	Mi	Ré	Sol	sol	sol	La	Si	La	sol	Si	la	La	sol				
Après transposition =	La	La	Ré	Do	Si	La	Sol	Do	Do	Do	Ré	Mi	Ré	Do	Mi	Ré	Ré	Do				
Durée t =	1/4	1/4	1/8	1/8	1/8	1/8	1/2	1/8	1/8	1/8	1/8	1/4	1/4	1/8	1/8	1/8	1/8	1/2				
$(R_0)=(R_0)=\frac{10^9}{f}$ =	45	45	34	38	40	45	51	38	38	38	34	30	34	38	30	34	34	38				
$(R_2)=tf$ =	55	65	87	85	81	28	98	33	33	33	37	82	73	33	41	37	37	131				

Conclusion :

MS DATA : 38 , 38 , 38 , 34 , 30, 34 , 38, 30, 34, 34, 38 ,
38 , 38 , 38 , 34 , 30, 34 , 38, 30, 34, 34, 38 ,
34 , 34 , 34, 34, 45, 45, 34, 38 , 40, 45 , 51 , 38,
38, 38, 34 , 30 , 34, 38, 30, 34 , 34, 38

MN DATA : 33, 33, 33 , 37, 82, 73, 33, 41, 37, 37, 131
33 , 33, 33 , 37, 82 , 73 , 33 , 41 , 37, 37 , 131
37, 37, 37, 37, 56, 55, 37, 85, 81, 28 , 98, 33, 33,
33, 37, 82, 73, 33, 41, 37, 37 , 131

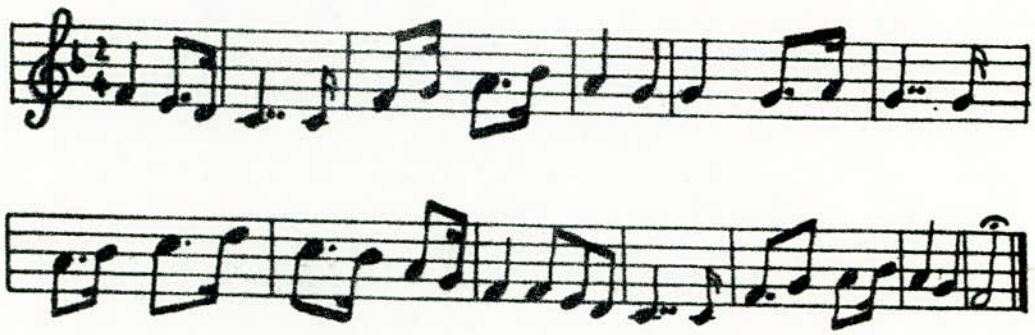
programme :

```
      A 0200

WS    BSS 32          réservation de la zone d'espace de travail
GØ    LWPI WS         initialiser le pointeur d'espace de travail
      LI  R12,>43C    adresse CRU du haut-parleur
AA    LI  R9,...      charger le compteur des notes
1     LI  R7, MN      charger R6 par l'adresse de la liste des notes
      LI  R6, MS      =  R7 par l'adresse de la liste des durées des notes
AB    MØV *R6+, R0
      MØV *R7+, R2
      CI  R0,0        est-ce qu'il y a une note de silence ?
      JEQ BB          Oui
      JMP $+4         non : correction des notes et leur durée
      SLA R0,1        s'il le faut
      SRA R2,1
      MØV R0,R1
      MØV R0,R3       Sauvegarder la durée d'émission
A1    SBØ 0           mettre le haut parleur sous tension
      MØV R1,R4       Sauvegarder la durée d'arrêt d'émission
      DEC R3
      JEQ DØ          a-t-on fini d'émettre ?
      SRC R5,3        non on temporise
      JMP A1
D1    SBZ 0           Couper le haut-parleur
      MØV R0,R3       Sauvegarder la durée d'émission.
      DEC R4          décrémenter la durée d'émission
      JEQ D2          a-t-on fini la durée de repos ?
DØ    SRC R5,3        temporisation
      JMP D1
D2    DEC R2          Décrémenter le compteur de cycle, si on n'a pas fini
      JNE A1          sauter à A1 pour remettre le haut parleur sous tension
      DEC R9          Décrémenter le compteur des notes
      JGT AB
      JMP AA          reprendre dès le début.
BB    DEC R2          boucle de la longueur de la note de
      JNE BB          silence.
      JMP AB          lire la note suivante

MS    DATA ...        données de la chanson.
MN    DATA ...

      END GØ
```

exemple 2 :

" trois jeunes tambours sont revenus de guerre"

Solfège :



de la même manière que précédemment, on obtient les données suivantes :

MS DATA 38 , 40 , 45 , 51 , 51 , 38 , 34 , 30 , 28 , 30 , 34 , 34 ,
34 , 30 , 34 , 34 , 30 , 28 , 25 , 23 , 25 , 28 , 30 , 34 ,
38 , 38 , 40 , 45 , 51 , 51 , 38 , 34 , 30 , 28 , 30 , 34 , 38

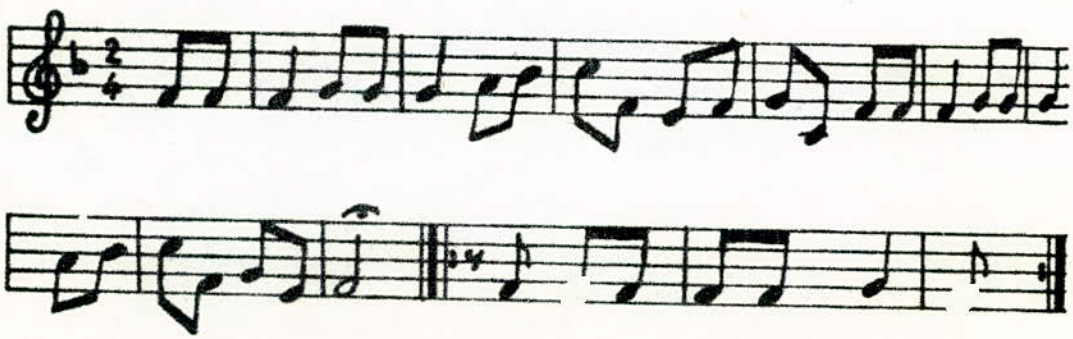
MN DATA 66 , 45 , 14 , 84 , 12 , 48 , 18 , 63 , 22 , 82 , 73 , 73 , 54
21 , 126 , 18 , 63 , 22 , 72 , 27 , 72 , 22 , 63 , 18 , 33 , 33 ,
31 , 28 , 84 , 12 , 48 , 18 , 63 , 22 , 82 , 73 , 131

Remarque : Le compteur des notes (R9) doit contenir la valeur 37.

exemple 3 :

" Sur le pont d'avignon "

Solfège :



résultat :

MS DATA 38, 38, 38, 34, 34, 34, 30, 28, 25, 38, 40, 38, 34,
51, 38, 38, 38, 34, 34, 34, 30, 28, 25, 38, 34, 40,
38, 0, 38, 38, 38, 38, 38, 34, 38, 0, 38, 38, 38,
38, 38, 34, 38

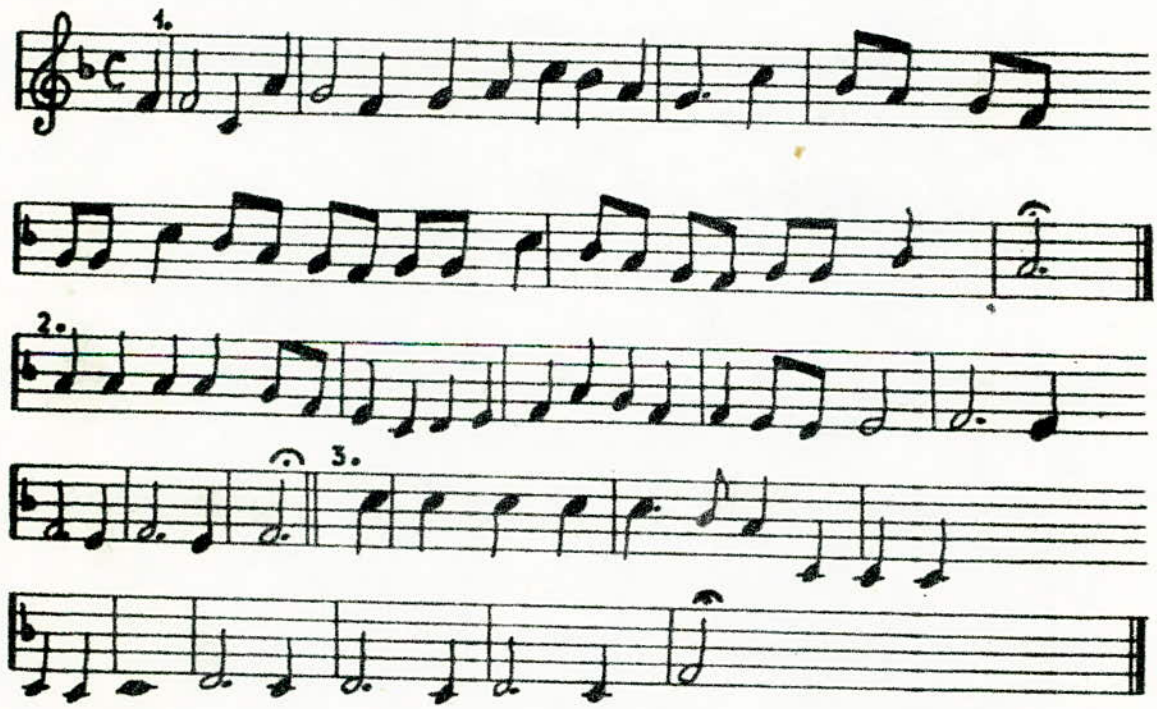
MN DATA 33, 33, 66, 37, 37, 73, 41, 44, 49, 33, 31, 33, 37, 24,
33, 33, 66, 37, 37, 73, 41, 44, 49, 33, 37, 31, 151,
8929, 33, 33, 33, 33, 33, 73, 33, 8929, 33, 33,
33, 33, 33, 73, 33

le compteur des notes (R9) doit contenir la valeur 43 pour une note de silence (0), la valeur de R_1 est calculée à partir de la boucle de Silence (DEC = 16, JNE = 12, $\gamma = \frac{1}{8}$)

Exemple 4 :

" Les cloches de HAARLEM "

Solfège:



MS DATA 38, 38, 51, 30, 34, 38, 34, 30, 25, 28, 30,
34, 25, 28, 30, 34, 38, 34, 34, 25, 28, 30,
34, 38, 34, 34, 25, 28, 30, 34, 38, 34, 34,
28, 30, 30, 30, 30, 30, 34, 38, 40, 51, 45,
40, 38, 30, 34, 38, 38, 40, 45, 40, 38, 40

38, 40, 38, 40, 38, 25, 25, 25, 25, 25, 28, 30, 51,
51, 54, 51, 54, 51, 45, 51, 45, 51, 45, 51, 38.

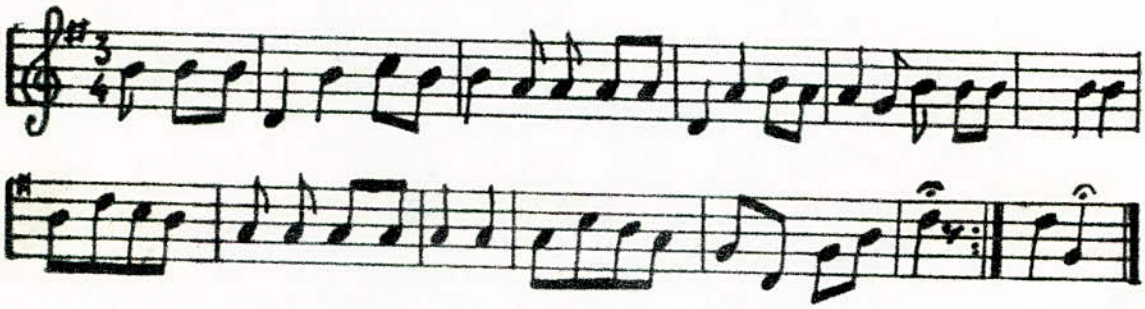
MN DATA 66, 131, 49, 82, 146, 66, 73, 82, 98, 87, 82, 219,
98, 44, 41, 37, 33, 37, 37, 98, 44, 41, 37, 33, 37, 37, 98,
44, 41, 37, 33, 37, 37, 87, 246, 82, 82, 82, 82, 37, 33,
62, 49, 55, 62, 66, 82, 73, 66, 66, 31, 28, 124, 198, 62,
198, 62, 198, 62, 198, 98, 196, 98, 98, 147, 44, 82,
49, 49, 49, 49, 49, 196, 165, 49, 165, 49, 165, 49,
198.

Remarque: le registre 9 (compteur des notes dans le programme Synthétiseur de fréquence) doit contenir la valeur 80

Exemple 5

' il était un petit navire'

Solfège



Résultat:

MS DATA (30, 30, 30, 51, 30, 28, 30, 30, 34, 34, 34, 34,
54, 34, 30, 34, 34, 38, 30, 30, 30, 30, 30, 30,
25, 28, 30, 34, 34, 34, 34, 34, 34, 34, 28, 30,
34, 38, 51, 38, 30, 25, 0) (25, 38.

MN DATA (41, 41, 41, 49, 82, 44, 41, 82, 37, 37, 37, 37,
49, 73, 41, 37, 73, 33, 41, 41, 41, 82, 82, 41,
49, 44, 41, 37, 37, 37, 37, 73, 73, 37, 44, 41,
37, 33, 24, 33, 41, 98, 43393) (98, 66.

Remarques: * les données comprises entre 2 parenthèses ( ) doivent être répétées 2 fois lors d'une programmation

* le registre R9 (compteur des notes dans le programme Synthétiseur de fréquence) doit contenir le nbre 88

* Quand on a une note de silence (les registres R0 et R1 contiennent la valeur 0), on utilise la boucle de silence (DEC, JNE) pour calculer la valeur de R2.

transformation des touches des 2 colonnes de droite du clavier en piano.

```
      A   0200

WS    BSS  32
GØ    LWPI   WS
      LI     12,>43C
      XØP    R9, 11
      CB     R9, ØBB
      JEQ    WC
      CB     R9, ØBC
      JEQ    WC
      CB     R9, ØBD
      JEQ    WC
      CB     R9, ØBE
      JEQ    WC
      CB     R9 , ØBF
      JEQ    WC
      CB     R9 , ØBG
      JEQ    WC
      CB     R9 , ØBH
      JEQ    WC
      CB     R9 , ØBI
      JEQ    WC
      CB     R9 , ØX1
      JEQ    WC
      CB     R9 , ØX2
      JEQ    WC
      CB     R9 , ØX3
      JEQ    WC
      CB     R9 , ØX4
      JEQ    WC
      CB     R9 , ØX5
      JEQ    WC
      CB     R9 , ØX6
      JEQ    WC
      CB     R9 , ØX7
      JEQ    WC
      CB     R9 , ØX8
      JEQ    WC
      JMP    AB
WC    SWPB   R9
      MØV    R9, R3
      LI     R2,
A1    SBØ    0
      MØV    R9, R4
      DEC    R3
      JEQ    DØ
      SRC    R5,3
      JMP    A1
D1    SBZ    0
      MØV    R9, R3
      DEC    R4
      JEQ    D2
DØ    SRC    R5,3
      JMP    D1
      DEC    R2
      JNE    A1
      JMP    AB
BB    DATA   >2000
BC    DATA   >3068
BD    DATA   >3500
BE    DATA   >4100
BF    DATA   >4600
BG    DATA   >4800
BH    DATA   >5000
BI    DATA   >5500
X1    DATA   >5400
X2    DATA   >4F00
X3    DATA   >4A00
X4    DATA   >4500
X5    DATA   >3900
X6    DATA   >3400
X7    DATA   >3E00
X8    DATA   >2400
      END    GØ
```

B. Horloge temps réel :

L'Horloge temps réel est définie précédemment comme étant un compteur sur 14 bits se décrémentant automatiquement à partir d'une valeur programmée, son schéma de principe est représenté sur la figure [4-3] . L'initialisation de ce compteur se produit en positionnant à 1 le bit de contrôle et les bits 1 à 14 à une valeur correspondante au comptage désiré, la modification de cette valeur entraîne le changement de la valeur d'intervalle de temps. Dans ce mode d'utilisation, l'horloge temps réel génère une interruption de façon périodique, ainsi l'intervalle de temps entre deux interruptions consécutives étant fixe. La fréquence de l'horloge de contrôle est 64 fois plus grande que la fréquence du signal pilotant cette temporisation ; par exemple : si le registre d'horloge est chargé par la valeur N telle que $1 \leq N \leq 16384$, l'intervalle du temps est donné par :

$$T = N . \frac{64}{f}$$

f étant la fréquence du signal $\bar{\Phi}$ appliqué au TMS 9901
f en Hz et T en seconde

donc une demande d'interruption de niveau 3 est générée toutes les T seconde.

valeurs extrêmes de cet intervalle pour $f = 2$ MHz.

$T_{min} = 1 \times \frac{64}{f} . 10^{-6}$ s $= 32\ \mu s$.

$T_{max} = N_{max} \frac{64}{f} . 10^{-6}$ s $= 524\ \mu s$

Les valeurs comprises dans cet intervalle [32 µs, 524 µs] sont disponibles par pas de 32 µs, mais cette méthode ne permet pas de mesurer directement les intervalles supérieurs à 524 µs, cependant plusieurs itérations avec incrémentation d'un registre permet d'étendre cet intervalle à volonté.

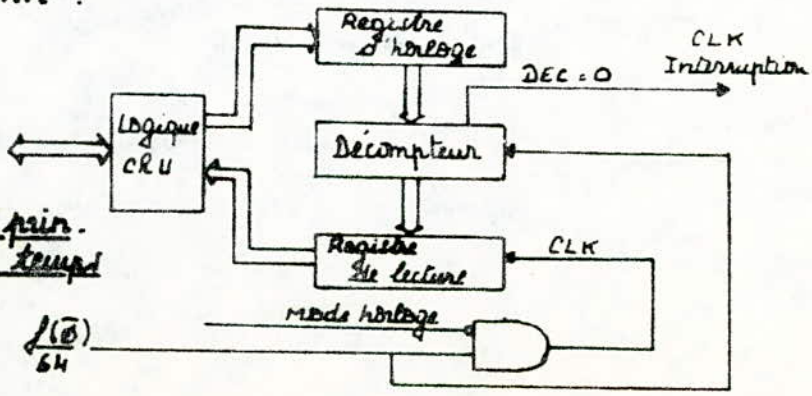
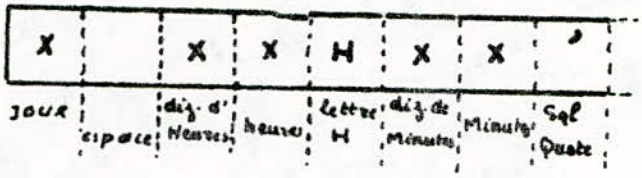


Fig 4-3 : Schéma de principe de l'horloge temps réel.

Application :

L'exemple choisi gère l'heure et le jour de la semaine et déclenche une alarme à chaque heure en un nombre de tops égal à l'heure affichée avec un changement de fréquence pour le dernier top. L'heure est obtenue par programme à partir des interruptions périodiques émises par une horloge temps réel, ce programme assure la fonction d'une horloge sur une semaine, il tient compte des heures écoulées sous forme d'un nombre de 0 à 23 et il affiche sur 8 bits le jour, les heures et les minutes.



La temporisation programmable du TMS 9901 réservée à la visualisation est initialisée pour générer une interruption toutes les 200 millisecondes, d'où la valeur de temporisation :

$$N = \frac{Tf}{64} = 6250 \qquad T = 200\,ms \quad f = 2\,MHz$$

$$(2N+1)_{16} = 30D5$$

L'horloge temps réel de 200ms, servant à comptabiliser l'heure est utilisée en combinaison avec la temporisation qui interrompt le TMS 9980 A, cette dernière interruption périodique qui se produit toutes les millisecondes sert à rafraichir régulièrement les caractères affichés sur le terminal.

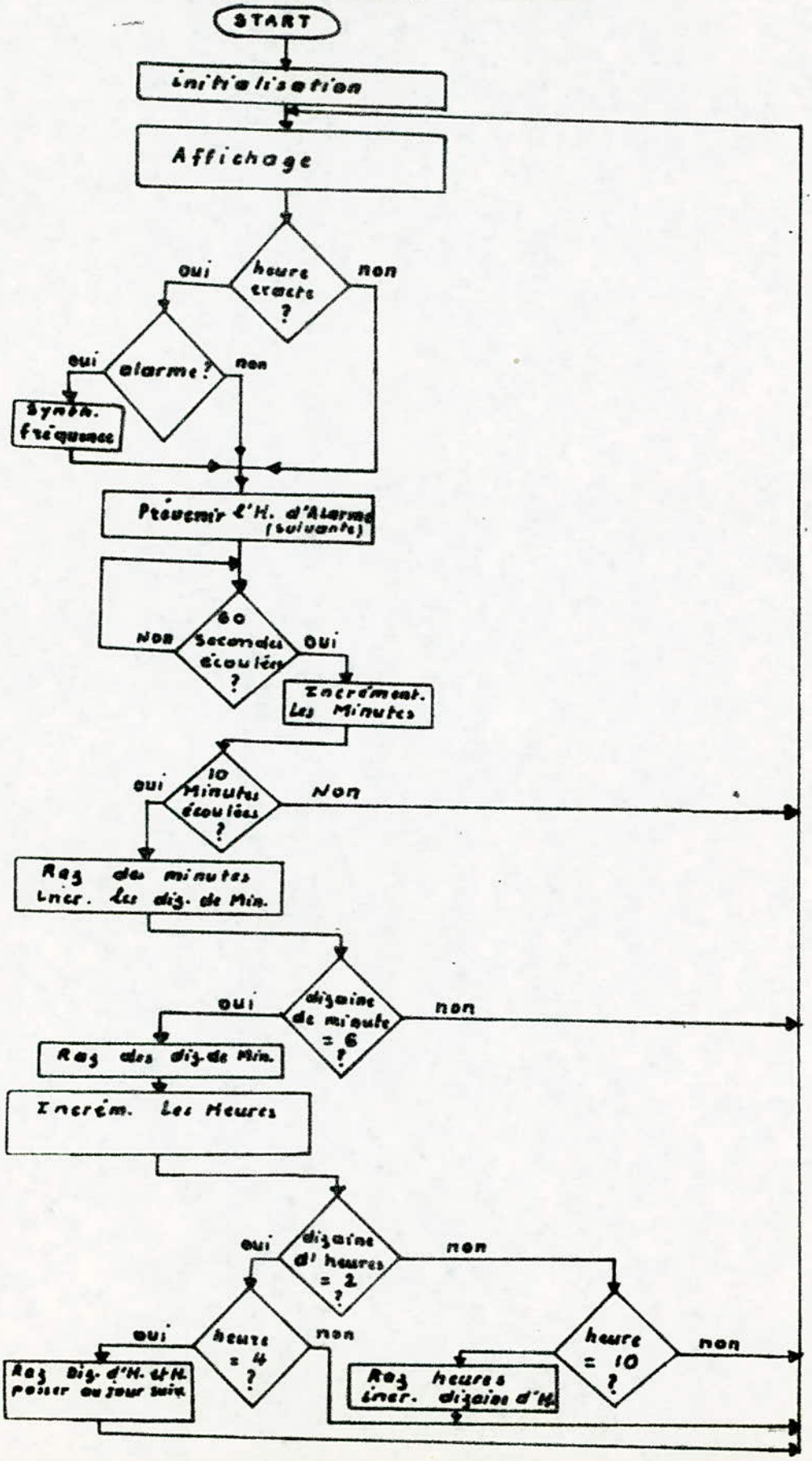
Valeur de la temporisation.

$$N = \frac{Tf}{64} = 31 \qquad T = 1\,ms \quad f = 2\,MHz$$

$$(2N+1)_{16} = 3F$$

Le programme exploite les XOP utilitaires disponibles dans le moniteur de la carte pour communiquer avec l'opérateur lorsqu'on effectue une remise à l'heure, le diagramme d'exécution est décrit à la page (82).

Organigramme



Programme

```
      A   0200
IW    EQU >1E0      adresse de l'espace de travail associé à l'interruption de Niveau 4
W4    EQU >10       adresse du vecteur WP pour l'IT 4
P4    EQU >12       adresse du vecteur PC de l'IT 4
CT    EQU >3005     Contrôle de la temporisation à    ms de U1
DT    EQU >3F       Contrôle de la temporisation à    ms de U1
WS    BSS 32
ST    LWPI WS       initialiser le WP ( entrée du Synthétiseur de fréquences)
      LI  R12,>43C
      CI  R8,0
      JEQ TT
T1    LI  R0,>A
      LI  R2,>150
      CI  R8,1      dernier top ?
      JGT WT        Non
      SLA R0,1      oui  modification de la note
      SRA R2,1
WT    MOV R0,R1
WC    MOV R0,R3
A1    SBO 0
      MOV R1,R4
      DEC R3
      JEQ D0                 } boucle d'émission
      SRC R5,3
      JMP A1
D1    SBZ 0
      MOV R0,R3
      DEC R4
      JEQ D2                 } boucle d'arrêt d'émission
D0    SRC R5,3
      JMP D1
D2    DEC R2
      JNE A1
      LI  R2,>3FFF
RR    DEC R2        } boucle d'intervalle de silence
      JNE RR          entre 2 Tops
      DEC R8
      JNE T1
TT    B   *11       retour au programme appelant

      AORG >26E
TC    LWPI >200     initialiser le WP
      LI   R0,IW    initialiser le vecteur d'IT de niveau 4
      MOV  R0,@W4
      LI   R0,IS
      MOV  R0,@P4
```

```
        CLR   R2
        CLR   R8
C1      XØP   ØJØ(R2),14      demander l'introduction
        XØP   ØDM(R8),9       introduire, le jour, l'heure, la minute
        DATA  C1
        DATA  C1
        AI    R2,8
        INCT  R8
        CI    R2,>20
        JNE   C1
        XØP   ØJF,14          Demande d'une fonction Alarme ou Non
        XØP   ØAA,11          introduire une lettre
        C     ØAA,ØXT         le caractère lu est-il un "A" ?
        JNE   TW              Non
C4      XØP   ØHE,14          Oui introduire l'heure
        XØP   ØTH,9
        DATA  C4
        DATA  C4
TW      CLR   ØTK             RAZ du compteur d'impulsion
        LI    R12,>400        adresse de base CRU, adresse du TMS 9901 (U11)
        LI    R1,CT           initialiser la valeur de la temporisation.
        LDCR  R1,15           démarrer la base du temps
        SBZ   0               placer le TMS 9901 en mode d'interruption
        SBØ   3
        LI    R12,0           adresse de base CRU ; adresse du TMS 9901 (U10)
        LI    R1,DT           initialiser la valeur du cycle de rafraichissement
        LDCR  R1,15           démarrer la base du temps.
        SBZ   0               placer le TMS 9901 dans le mode interruption.
        SBØ   3               valider les interruptions du niveau 1 et 4
        SBØ   6
        LIMI  4               valider le masque d'interruption jusqu'au niveau 4
        LI    R9,6
T4      CB    ØDM,ØDK(R9)     aligner la valeur d'index
        JEQ   T6
        DEC   R9
        JMP   T4
T6      XØP   ØBM,14          afficher le
        C     ØXT,ØAA         a-t-on programmé une alarme ?
        JNE   T5              non
        C     ØTL,ØTH         Oui heure exacte ?
        JNE   T5              non : pas encore
        MOV   ØTL,R8          nbre de TOP égal à l'heure
        BL    ØST             branchement au Synthétiseur de fréquence.
T5      MØV   ØTH,ØTL         sauvegarder l'heure incrémentée de 1
        INC   ØTL
        LI    R2,4
        LI    R4,>12C         nbre d'impulsions par minute.
```

```
T3    C     @TK,R4      80 secondes écoulées
      JL    T3          non
      CLR   @TK         Oui, mise à jour du compteur d'impulsion.
      LI    R3,>100
      AB    R3,@TM(R2)  incrementer les minutes
      CB    @TM(R2),@MV(R2)  dépassement ?
      JNE   T6          Non, aller à la visualisation.
      MOVB  @TJ,@TM(R2) Oui raz des minutes.
      DEC   R2
      AB    R3,@TM(R2)  +1 sur les dizaines de minutes
      CB    @TM(R2),@MV(R2)  dépassement ?
      JNE   T6          Non
      MOVB  @TJ,@TM(R2) Oui, RAZ des dizaines de minutes
      DECT  R2
      INC   @TH         +1 au compteur de TOP
      AB    R3,@TM(R2)  +1 sur les heures
      CB    @TM,@MV     dizaine d'heures = 2 ?
      JEQ   XY          Oui
      CB    @TM(R2),@MV(R2)  non, dépassement de 9 ?
      JNE   T6          Non
      MOVB  @TJ,@TM(R2) Oui RAZ des heures
      AB    R3,@TM      +1 sur les dizaines d'heure
      JMP   T6
XY    CB    @TM(R2),@HM  unité d'heures = 4 ?
      JNE   T6          Non
      CLR   @TH         Oui, RAZ du compteur de TOP
      MOV   @TJ,@TM           Ainsi que les heures et les dizaines d'heures
      CI    R9,0        passer au jour suivant.
      JNE   S1
      LI    R9,7
S1    DEC   R9
      MOVB  @DK(R9),@DM
      JMP   T6

AA    DATA  0
JB    DATA  >0D0A       retour chariot, saut de ligne
      TEXT  'J?'        demande à l'operateur d'introduire le jour
      DATA  0
HE    DATA  >0D0A
      TEXT  'H?'        demande à l'opérateur d'introduire l'heure
      DATA  0
      DATA  >0D0A
      TEXT  'D?'        demande à l'opérateur d'introduire les dizaines
      DATA  0             de minutes
      DATA  >0D0A
      TEXT  'M?'        demande à l'opérateur d'introduire les minutes
      DATA  0
```

```
JF      DATA  >0D0A
        TEXT  ' A OR N?'
        DATA  0
BM      DATA  >0D0A
DM      DATA  0
        DATA  0
        DATA  0
        DATA  0
        DATA  0
        DATA  0
MV      DATA  >323A
        DATA  >0036
        DATA  >3A00
TJ      DATA  >3030
TK      DATA  0
DK      DATA  'DS'
        DATA  'VJ'
        DATA  'LM'
        DATA  'L O'
TH      DATA  0
        DATA  0
AT      DATA  >4100
TL      DATA  0
        DATA  0
HM      DATA  >3400
```

Question à l'opérateur alarme ou Non

jour + espace
diz. d'H + H.
H + diz Minutes
Minutes + '

Valeurs de comparaison

valeurs pour la remise à zéro
Compteur d'impulsion
lettres représentant les jours de la semaine

Compteur des TOP

'A' en code ASCII
adresse recevant une valeur de comparaison.

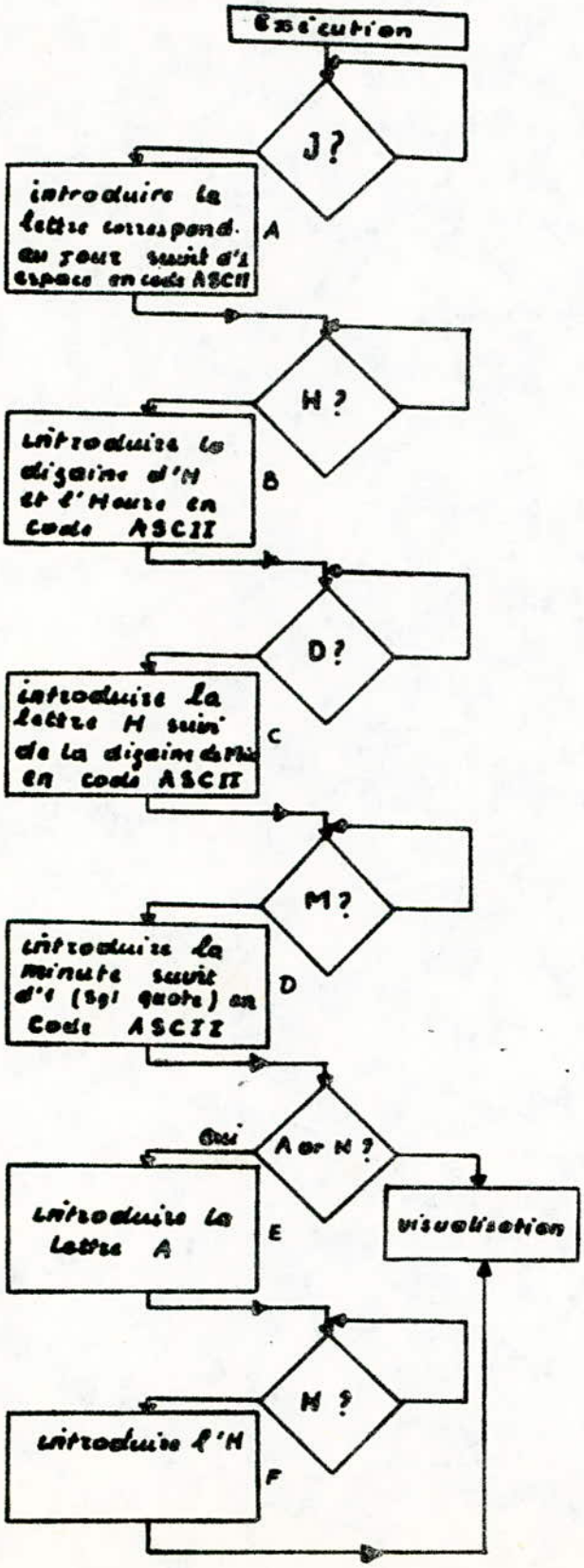
valeur de comparaison

Routine du traitement des interruptions

```
        ADRG  >3F4
IS      INC   @TK
        LI    R12, >400
        SBO   3
        RTWP
        END   TC
```

incrémenter le compteur d'impulsion
adresse de base de TMS 9901 U,,
RAZ et validation du 9901
restauration du contexte

Remarque : après l'exécution de ce programme, la touche shift permet l'accélération de changement d'heures, ainsi une correction rapide de ces heures.



A : lettres correspondants au jour suivies d'un espace en code ASCII

D :	Dimanche	4 4 20
S :	Samedi	5 3 20
V :	Vendredi	5 6 20
J :	Jeudi	4 A 20
W :	Mercredi	5 7 20
M :	Mardi	4 d 20
L :	Lundi	4 C 20

B : chiffres — correspondance en code ASCII

0	30
1	31
2	32
3	33
⋮	⋮
9	39

Ex: 14 → 31 34 en code ASCII

C)

H en code ASCII = 48

pour les chiffres même chose que B

Exemple

H 3 → 48 33

D) ' (Sgl quote) en code ASCII = 27

pour les chiffres même chose qu'en B et C

Ex: 2' → 32 27

F) : heure en décimal égale à l'heure introduite en code ASCII

Chronomètre

programme

```
     A  0200
IW   EQU  >160      adresse de l'espace de travail associé à l'IT de niveau 4
W4   EQU  >010      adresse du vecteur WP pour l'IT 4
P4   EQU  >012          :     :   PC  :   :   :
CT   EQU  >1868     Contrôle de temporisation à      ms de U4
DT   EQU  >3F           :     :   :   à      ms de U1
WS   BSS  32
G0   LWPI WS        initialiser le WP
     LI   R0,IW     initialisation des vecteurs d'IT de niveau 4
     MOV  R0,@W4
     LI   R0,IS
     MOV  R0,@P4
     CLR  @TK       mise à zéro du compteur d'impulsion
     LI   R12,>400  adresse de base CRU (adresse du TMS 9901 U44)
     LI   R4,CT     initialiser la valeur de la temporisation.
     LDCR R4,15     démarrer la base du temps.
     SBZ  0         placer le TMS 9901 en mode interruption
     SBO  3
     LI   R12,0     adresse de TMS 9901 U10
     LI   R4,DT     initialiser la valeur du cycle de rafraîchissement
     LDCR R4,15     démarrer la base du temps
     SBZ  0         TMS 9901 en mode interruption
     SBO  3         valider les interruptions de niveau 1 et 4
     SBO  6
     LIMI 4         valider le masque d'interruption jusqu'au niv 4
     MOVB @TB,@TM
     MOV  @TM,@NN
     LI   R2,1      } remise à zéro de la visualisation
     MOVB @TB,@MM(R2)
TG   XOP  @DM,14    visualiser le temps
     LI   R2,5
     LI   R4,5      valeur de comparaison
T4   C    @TK,R4
     JL   T4
     CLR  @TK
     LI   R5,>400
     LI   R5,
T2   AB   R5,@TM(R2)    + au dixième de seconde
     CB   @TM(R2),@MV(R2)  1 seconde écoulé
     JNE  TG               Non
```

```
        MOV     @TB,@TM(R2)
        DECT    R2
        AB      R3,@TM(R2)
        CB      @TM(R2),@MV(R2)
        JNE     T6
        MOVB    @TB,@TM(R2)
        DEC     R2
        AB      R3,@TM(R2)
        CB      @TM(R2),@MV(R2)
        JNE     T6
        MOVB    @TB,@TM(R2)
        DECT    R2
        AB      R3,@TM(R2)
        CB      @TM(R2),@MV(R2)
        JNE     T6
        MOVB    @TB,@TM(R2)
        INCT    R2
        MOV     @TM,@TM(R2)
        INCT    R2
        INC     R2
        MOVB    @TB,@TM(R2)
        JMP     T6
TB      DATA    >3000
TM      DATA    >3030
DM      DATA    >0D0A
        DATA    'CR'
TM      DATA    >0027
NN      DATA    0
MM      DATA    >2200
        DATA    >2000
MV      DATA    >3A20
        DATA    >363A
        DATA    >203A
TK      DATA    0

        AORG    >3100
IS      INC     @TK
        LI      R12,>0400
        SBO     3
        RTWP
        END     GO
```

Oui ; +1 au seconde
et remise à zéro du dixième de seconde

10 secondes écoulées ?
Non
Oui RAZ des secondes

+1 sur les dizaines de secondes
dizaines de secondes = 6 ?
Non
Oui RAZ des dizaines de secondes

+1 sur les minutes
minutes = 10 ?
Non
Oui . RAZ les minutes

RAZ secondes et dizaines de secondes

RAZ des dixièmes de secondes.

valeur pour la remise à zéro d'un octet
valeur pour la remise à zéro d'un mot ou d'1 octet

minutes suivies d'1 Sql. Quote
dizaines de secondes et secondes
d'blc. Quote suivi du dixième de seconde

valeurs de comparaison
valeurs de comparaison
valeurs de comparaison
compteur d'impulsions

incrementer le compteur d'impulsion
adresse de base de TAIS 9901 U,
RAZ et validation de TAIS 9901
restauration du contexte.

Conclusion

Cette étude nous a permis de distinguer l'utilité des microprocesseurs dans le domaine de la pratique et plus particulièrement dans l'économie du temps et le contrôle industriel. Au cours de notre sujet nous étions limités par la taille mémoire de la carte TM 990/189, ce qui nous a empêché de faire des applications plus intéressantes que celles qui sont données ; par exemple la réalisation d'un calendrier électronique ; en plus le manque de certains périphériques ne nous a pas permis d'utiliser des terminaux extérieurs (mis à part le lecteur-enregistreur); par exemple ; une machine à écrire ou un vidéo permettant la visualisation de l'écriture du programme.

Bibliographie

* Introduction aux microprocesseurs
(Texas Instruments 1980)

* Guide d'utilisation de la carte universite TM 990/189

* 9900 Family Systems Design and DATA Book
(1st Edition)

* Electronique Applications N° 13 1980

* Electronique pratique N° 17 Juin 79

* 1000 chants par Jean Edel Berthier
III: trimestre 73 edition N° 561

* Electronique applications N° 17 19 81

* Interface programmable TMS 9901
(Texas Instruments)