الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTROTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

16/86
1ea

PROJET DE FIN D'ETUDES

En vue de L'obtention du diplome d'ingenieur d'état

SUJET

LOGICIEL POUR LA COMMANDE NUMERIQUE PAR CALCULATEUR

Proposé par : Dr SCHREYER

Etudié par : M. DjOUDER

Dirigé par : Dr SCHREYER
M.S BOUCHERIT

PROMOTION : JUIN 1986

E.N.P. 10, Avenue Hacen Badi – EL-HARRACH – ALGER

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

-oOo- R E M E R C I E M E N T S -oOo-

Je suis profondément reconnaissant à Monsieur SCHREYER pour m'avoir guidé avec tant de compétence et de clairvoyance, dans la réalisation de ce travail.

Je tiens particulièrement à remercier Monsieur BOUCHERIT pour ses conseils et pour les encouragements qu'il m'a prodigués.

J'adresse également mes remerciements à tous les professeurs qui ont contribué à ma formation.

Que tous, ceux qui ont participé de prés ou de loin à l'élaboration de ce travail, trouvent ici l'expression de ma profonde reconnaissance.

-=-=-=-=-=-=-=-=-o-oOo-o-=-=-=-=-=-=-=-

-oOo- ///) E D I C A C E S -oOo-

Je dédie ce travail :

- A la mémoire de mon père.
- A ma mère.
- A mes frères et soeurs.
- A tous mes amis.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

الموضوع: إعداد برامج للتحكم العددي بواسطة العقل الالكتروني -

الملخص: هذا المشروع يحتوي على إعداد برامج للتحكيم العددي بواسطة العقل الالكتروني و يتكون من :

- وصف البنية الداخلية للعقل الالكتروني .
- دراسة لغة التجميع (Langage d'assemblage) على الميكروبروسيسور Z 80
- اعداد برامج بلغة التجميع لمختلف المنظمات . (PI, PD, PID)-

SUJET : "LOGICIEL POUR LA COMMANDE NUMERIQUE PAR CALCULATEUR".

RESUME: Ce projet consiste à l'établissement de programmes pour la commande numérique par calculateur, il se résume par :

- La description de la structure interne d'un ordinateur.
- L'Etude du langage d'assemblage sur le microprocesseur Z 80.
- l'Etablissement de programmes en langage d'assemblage pour les régulateurs standards (PI, PD, PID).

SUBJECT : "SOFTWARE FOR DIGITAL-CONTROL BY CALCULATOR".

ABSTRACT: This project enclosed the preparation of programs for digital-control by calculator and consist of :

- The description of the internal structure of computer.
- The study of assemblage langage on the microprocessor Z 80.
- Programs preparation on assemblage langage for differents standards regulators (PI, PD, PID).

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

SOMMAIRE

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

G E N E R A L I T E S

On commande à l'aide d'ordinateurs des processus de plus en plus divers:le fonctionnement des centrales nucléaires, la distribution d'électécité ,le pilotage automatique et les autres c commandes des avions,le fonctionnement des ascenseurs , les robots ou les macines-outils.

Ces ordinateurs doivent travailler en temps réél c'est -à dire , ils doivent suivre instantanément les variations des para--mêtres du processus commandé afin de générer ainsi des ordres d'action sur ces derniers.

On utilise par exemple les ordinateurs de commande dans les raffineries de pétrole,notamment pour piloter le fonctionne--ment des colonnes à distillation fractionnée,où l'on sépare les composés chimiques lègers des produits plus lourds.

L'ordinateur piloté par le logiciel,reçoit les valeurs des quantités,débits,températures et pressions des différents fluides de la colonne;il envoie des commandes afin de modifier ces grandeurs et d'ajuster la quantité et les caractéristiques des produits fabriqués en fonction des objectifs de production. On peut en outre programmer le système de commande pour mini--miser la consommation d'énergie.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

I N T R O D U C T I O N

La dernière decennie a vu une évolution considérable de l'électronique et de l'informatique,sous l'influence du dévelop--pement technologique des circuits intégrés.Ces derniers étaient au départ monofonctionnels,l'introduction de la notion propre à l'informatique de programme enregistré dans une mémoire,a permis de concevoir des circuits intégrés multifonctionnels:ils se com--portent comme des automates electroniques à programme enregistré c'est-à-dire comme des ordinateurs désignés sous le nom de: microprocesseurs.

Plusieurs générations existent déjà,les différentes améliorations portent sur la vitesse et la capacité de traitement, sur la capacité d'adressage des mémoires et des entrées-sorties.

LES MICROPROCESSEURS DANS LA COMMANDE:

Le premier problème qui a retardé le développement de la commande numérique par ordinateur est celui de la très grande rapidité des phénoménes à contrôler.

Il ya environ 15 ans,les gros calculateurs n'étaient pas assez rapides pour effectuer les tâches de régulation en temps réél;c'est seulement grâce aux progrés réalisés par la micro-infor--matique que de telles réalisations deviennent possibles.

D'autre part,les ordinateurs de ce type ne peuvent travailler à leur rythme:les logiciels de commande doivent leur permettre de réagir rapidement à toutes les variations des systèmes commandés.

Prenons l'exemple où l'ordinateur commande un avion à réaction,l'ordinateur doit prendre des décisions très rapidement face à de multiples demandes simultanées(système multitâches) et syncroniser les différentes tâches qu'il doit accomplir car un bref changement des paramètres du réacteur sans que l'ordinateur ne soit informé peut mettre en danger des vies humaines.

Doncle temps d'exécution des logiciels de commande doit être très petitpar rapport à la constante prépondérante de la chaîne à automatiser.

Afin d'accelérer l'exécution des programmes,il faut choisir un langage le plus rapide qu'il soit sur un ordinateur.

Le langage le plus rapide sur un ordinateur est celui qui est directement compréhensible par celui-ci:c'est le langage machine; il utilise le code binaire des" 0 " et des " 1 ".

Avant d'aborder directement l'étude de la programmation en langagemachine,certains rappels sont nécessaires:ils portent prin-cipalement sur les systèmes des numérations dans lesquels s'effectuent les opérations en langage machine;sur la manière dont les opérations sont menées par l'ordinateur, Enfin sur la struc-ture interne de l'ordinateur:c'est le but du premier chapitre.

Remarque: le travail a été effectué sur un micro-ordinateur " ZX -SPECTRUM+"

C H A P I T R E I

1-SYSTEME DE NUMERATION:

Introduction:

L'ordinateur est une machine qui ne comprend,ne stocke et ne réalise des opérations que sur des informations qui sont repré--sentées sous forme numérique,par des grandeurs physiques à deux états que l'on représente en génèral par "0" et " 1 " .

Chacune de ces unités physiques s'appelle un bit(binary digit).Un bit est donc la plus petite unité d'information contenue dans un ordinateur et il peut prendre :" 0 " ou " 1 ".

Les bits sont regroupés en unité de mémoire;celles-ci pos--sédent le même nombre de bits:c'est une des caractéristiques importantes d'un ordinateur.

Pour les ZX Spectrum +,ce nombre est huit et une telle unité de memoire est appelée :octet (byte).
l'octet peut contenir une donnée numérique entière comprise entre 0 et 255.

Il résulte de ceci que le système binaire est le système de numération le plus adéquat pour traiter les informations;on utilise aussi la base 16 (hexa décimale)pour indiquer la valeur d'un octet.Cette valeur variera donc entre 0 et FF .

1-1 Système binaire:

Etant donné un nombre N et un systéme de numération de la base b ,on montre qu'il existe une suiteunique : a_0 ,a_1 ,a_2 ,... a_{n-1} d'élèments de l'ensemble des chiffres de ce système telle que:

$$N = a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \ldots\ldots\ldots\ldots a_1 \cdot b_1 + a_0$$

Dans le binaire, b=2, les ocefficients a_i sont pris parmi les chiffres du système: 0 et 1

exemple:

Soit: N = 101101 ce qui veut dire:

$N = 1.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0$

$N = 2^5 + 2^3 + 2^2 + 1 = 32 + 8 + 4 + 1$

N=45 dans le systéme décimal

donc: $(101101)_2 = (45)_{10}$

Le passage d'un nombre décimal au nombre binaire peut s'effectuer par une décomposition en puissances de 2.

Ex : $(61)_{10}$

La plus grande puissance de 2 contenue dans:

61	est	32 (2^5)	reste 29
29		16 (2^4)	13
13		8 (2^3)	5
5		4 (2^2)	1
1		1 (2^0)	0

donc $(61)_{10} = 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 1.2^5 + 1.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0$

$= (111101)_2$

Ce nombre peutêtre mis dans un octet :

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

Les opérations sur les nombresbinaires s'effectuent de la manière suivante donnée par les tables d'addition et de multiplication :

+	0	1
0	0	1
1	1	10

x	0	1
0	0	0
1	0	1

Ex 1 : 5 + 6 = 11 en décimal

$5 = 101_2$

$6 = 110_2$

d'où

$$\begin{array}{r} 101 \\ +\ 110 \\ \hline 1011 \end{array}$$

$1011 = 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 2^3 + 2 + 1 = 11_{10}$

Ex 2 : 5 x6 = 30 en décimal

en binaire 1 0 1

x 1 1 0

0 0 0

1 0 1 .

1 0 1 .

1 1 1 1 0 = $2^4 + 2^3 + 2^2 + 2$ = 30

Les nombres négatifs:

On reserve un bit pour le signe.

Ce bit est à 0 si le nombre est positif et à 1 sui le nombre est négatif.

Ex: sur 8 bits + 5 -------- 0000101

- 5 -------- 1000101

Représentation par le complément à 2: (complément vrai)

On obtient le complément à 2en ajoutant 1 au complément restreint (inversion de tous les bits du nombre considéré).

Ex: Sur 8 bits

10	00001010
Complément restreint	11110101
Complément vrai	11110110

On peut représenter ainsi sur 8 bits (cas du ZX SPECTRUM +) tous les nombres compris entre -128 à +127 . C'est à dire un nombre négatif est représenté par un nombre égal à 256 plus sa valeur.

Pour un nombre codé sur 2 octets (16 bits), on peut représenter en complément à 2 tous les nombres compris entre

- 32768 et + 32767 .

1-2 Notation hexadécimale:

La numération hexadécimale , de base 16 , posséde seize symboles.On utilise les symboles suivants:

0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , A , B , C , D , E , F

Les symboles A,B,C,D,E,F ayant pour valeur décimale respective 10,11,12,13,14,15.

Conversion décimale -héxadécimale:

La plus simple méthode de conversion se fait par des divsions entières par 16 successives,le reste représente l'élèment héxadécimal de plus faible poids.

Ex: 34 en décimal

34	16		
2	2	16	
2		0	

$(34)_{10} = (22)_{16}$

Conversion héxadécimale -décimale:

$$N = a_{n-1} \cdot 16^{n-1} + a_{n-2} \cdot 16^{n-2} + \ldots\ldots + a_1 \cdot 16 + a_0$$

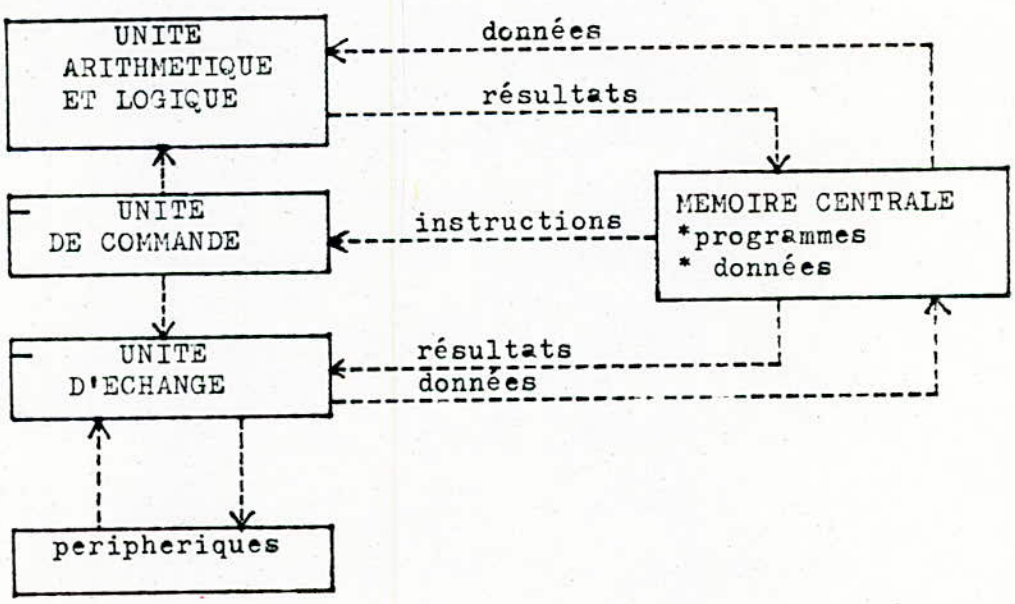
Ex: AB en héxadécimal

$$AB = A.16 + B = 10.16 + 11 = (171)_{10}$$

2- STRUCTURE INTERNE D'UN ORDINATEUR:

2-1 Organisation génèrale:

L'organisation génèrale d'un ordinateur est représentée par la figure suivante:



On distingue:

2-1-1- La mémoire centrale:

Physiquement,elle se présente comme un ensemble d'éléments bistables(0 et 1) permettant chacun de conserver un bit d'infor--mation.

Génèralement,on effectue des regroupements de ces bits en octets ou en mots.

La taille du mot pourra être :

- de 8 bits (cas du microordinateur ZX -SPECTRUM +)
- de 16 bits(sur beaucoup de micro-ou mini-calculateur)
- de 32 bits(sur beaucoup de moyens et gros calculateurs) IBM)

Pour exprimer les tailles des mémoires centrales,on emploie souvent des unités kilo-mot (Kmot) ou méga-mot (M mot).

1 Kmot = 2^{10} mots = 1024 mots, 1 Mmot= 2^{20} mots= 1024 Kmots.

La capacité mémoire varie de quelques Kmots (micro-ordinateur) à plusieurs Mmots (gros calculateurs).

Chaque mot est repéré par un nombre entier compris entre 1 et N (N étant la taille de lamémoire),nommé son "ADRESSE".

Caractéristiques d'une mémoire:

- La capacité
- La taille de l'unité mémoire
- La vitesse de fonctionnement
- Le temps d'accés.

Ex: pour le ZX SPECTRUM + : la capacité de la mémoire centrale est de 64 Kmctets,la taille de l'unité mémoire est de 8 bits(octets) et les adresses mémoires vont de 0 à 65535.

On peut distinguer deux types de mémoires:la mémoire vive et la mémoire morte.

2-1-1-a- La mémoire vive ou RAM :(RANDOM ACCESS MEMORY)

Cette partie de mémoire est reservée aux programmes utilisateurs. Cependant, les programmes peuvent être modifiés ou supprimés par l'utilisateur.

Cette mémoire travaille en lecture et en écriture.

La capacité de la RAM dans le "ZX SPECTRUM +" est de 48 K.O et s'étend de l'adresse 16384 à 65535.

2-1-1-b- La mémoire morte ou ROM :(Read Only Mémory)

Cette deuxième partie contient le programme moniteur qui permet l'usage de plusieurs périphériques et un interpréteur du langage évolué utilisé par l'ordinateur(BASIC ,FORTRAN ...).

Son contenu est mémorisé une fois pour toutes;il ne peut être modifié par l'utilisateur,c'est à dire qu'elle ne fonctionne qu'en lecture.cependant, une écriture peut provoquer l'arrêt de l'ordinateur sans que son contenu ne soit modifié.

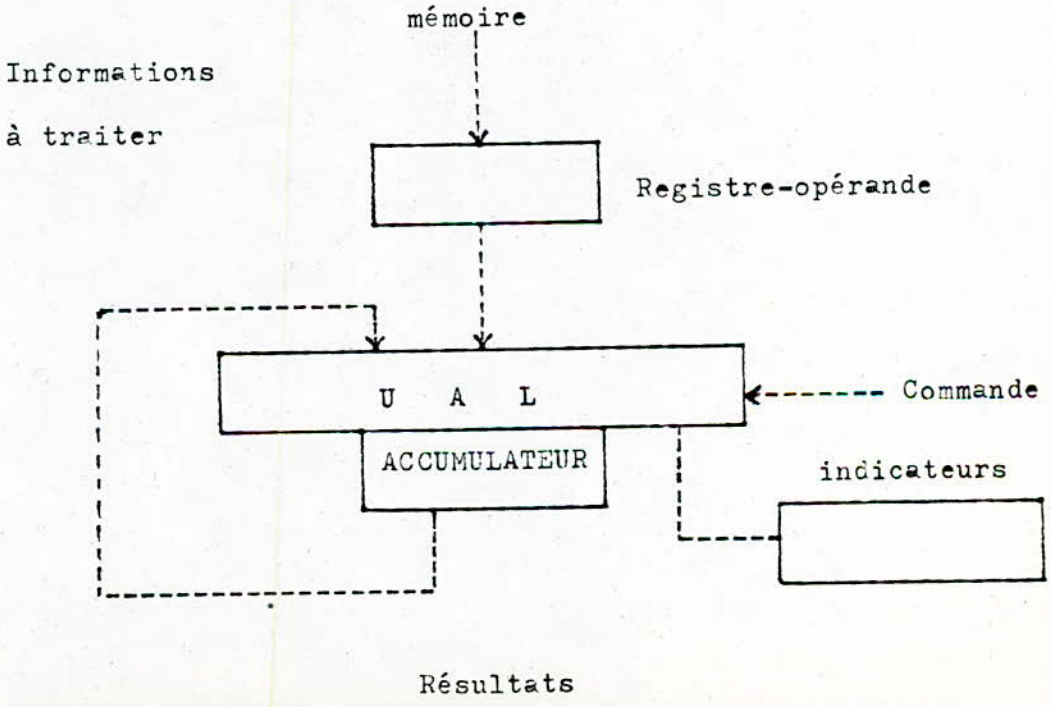
La mémoire ROM contient plusieurs routines en code machine qui sont appelées par l'interpréteur, au fur et à mesure de ses besoins.

La taille de cette mémoire sur le "ZX SPECTRUM +" est de 16 K.O et s'étend de l'adresse O jusqu'à 16384.

2-2- L'unité de traitement et logique: (UAL)

Elle contient en particulier les opérateurs arithmétiques et logiques qui traitent les informations issues de la mémoire centrale.

Le fonctionnement de l'UAL se résume par la figure suivante:



Cette unité posséde des "entrées " et des "sorties".

Les informations en entrée sont:

- Les opérandes provenant en génèral de la mémoire centrale et qui sont présentés à l'UAL dans les registres.

- La commande en provenance de l'unité de commande,indiquant l'opération à effectuer.

En sprtie,on distingue:

-Le résultat du calcul,présenté dans un registre(le registre peut être également une entrée de l'UAL).

-Les indicateurs qui indiquent le résultat de l'opération effectuée (résultat positif,négatif,débordement...).Ce sont ces indicateurs qui servent lors des instructions de branchement conditionnel.

2-3-L'unité de commande (ou de contrôle):

C'est ce bloc qui gère le déroulement du programme.

On y trouve:

2-3-1- Le registre-instruction:

Ce registre permet de stocker l'instruction au cours d'exécution qui vient d'être extraite de la mémoire centrale.

On dictingue principalement:-le code-opération

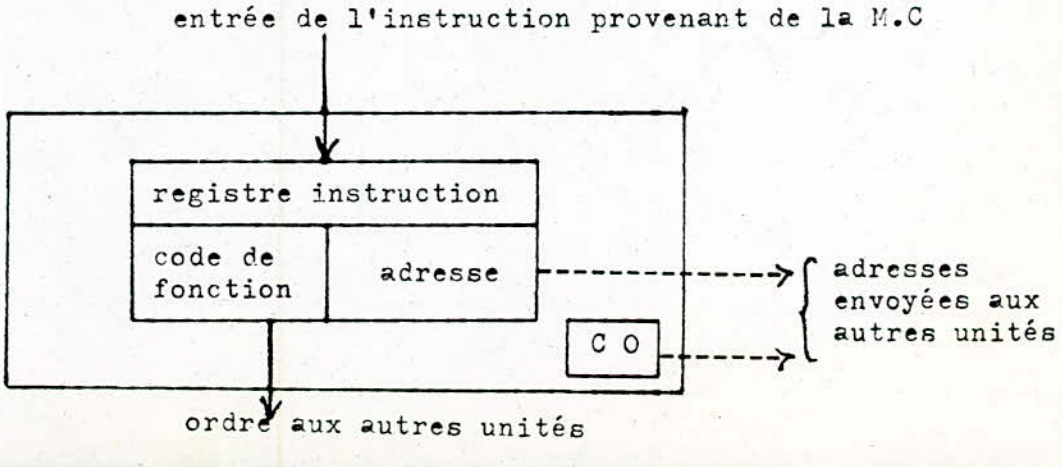
-le code-adresse

2-3-2- Le compteur ordinal :(CO)

C'est un registre dans lequel se trouve l'adresse où a été extraite,en mémoire,l'instruction en cours d'éxécution;son contenu est modifié ensuite pour passer à l'appel de l'instruction suivante.

2-3-3- Le décodeur de fonction:

Il permet d'interpréter la fonction de l'instruction à exécuter.



2-3-4- La base de temps:

Elle délivre des impulsions de façon périodique appelées : Teps d'horloge et destinées à rythmer le travail de l'unité de commande ; c'est à dire elle permet de distribuer les temps néces--saires à la commande des opérations élémentaires à effectuer pendant les différentes phases de déroulement d'une instruction.

C'est donc l'unité de commande qui est chargée de mettre en oeuvre l'éxécution des instructions d'un programme.Elle déclenche les échanges d'information entre la mémoire centrale et elle-même, entre l'unité centrale et l'unité de traitement,;.....

L'unité arithmétique et logique et l'unité de commande forment ensemble l'unité centrale de l'ordinateur (CPU) qui est le "coeur de la"machine" .

On définit également ,autour de l'unité centrale, d'autres blocs fonctionnels appelés périphériques.

2-4- Les périphériques:

Les périphériques sontdes organes de liaison entre l'ordinateur et le monde extérieur.

Ils ne sont pas fondamentalement nécessaires au fonctionnement de l'ordinateur,maissans eux l'ordinateur ne peut communiquer ses résultats à l'utilisateur.

Parmi ces périphériques,on distingue: le clavier,l'écran de visualisation,l'imprimante,table traçante,convertisseurs numériques analogiques et analogiques numériques ...ect.

2-5- L'unité d'échange:

Elle perm et les échanges d'infármation entre la mémoire centrale et les périphériques.

3- LANGAGES DE PROGRAMMATION:

3-1- Langages évolués: ce sont des langages de haut niveau dont l'apprentissage est rapide.

Parmi ces langages , on distingue:

- FORTRAN: premier langage en date,apparu en 1951,son nom est une abréviation de " FORmula TRANslation". Il est destiné essentiel--lement à une utilisation scientifique et technique,il reste dans ces

domaines,le langage le plus utilisé.

-ALGOL: il est tout particulièrement adapté à la mise en oeuvre d'algorithme mathématique.

Enthéorie,ALGOL couvre les mêmes applications que FORTRAN. Toutefois,il est d'un emploi moins aisé qt il produit des programmes moins performants (sur le plan vitesse d'éxécution).

-COBOL: (CO mmun Business Oriented Langage)

Apparu en 1959,il est à la gestion ce que FORTRAN est au domaine scientifique et technique.Dans ce domaine,il est le langage le plus répandu.

-BASIC:à l'origine,BASIC était un lan gage simple destiné a un apprentissage rapide et utilisé en mode conversationnel sur de petites machines.Depuis l'apparition des microprocesseurs,ce langage a repris de la vigueur en même temps que de nouvelles possibilités sont apparues.

-PASCAL: c"est par excellence,le langage de la programmation structurée.Assez répandu parmi les microprocesseurs,il reste encore peu utilisé dnas le domaine scientifique.

3-2- Les langages machine et d'assemblage:

Ils seront détaillés profondément dans le chapitre suivant (vu leurs importances dans ce travail).

C H A P I T R E II

1- LE LANGAGE MACHINE:

Introduction:

Comme on a déjà evoqué dans le chapitre précédent que le seul langage compréhensible par l'ordinateur était la langage machine des 0 et 1 . cependant, le langage BASIC est incompréhensible par l'ordinateur; pour utiliser ce langage,l'ordinateur dispose d'un interface entre le langage binaire et le langage BASIC manipulé par l'utilisateur. Cet interface est un programme écrit en langage machine et stocké dans la ROM;on appelle cet interface aussi "interpréteur BASIC ". Ainsi, tout programme en langage machine autre que l'interpréteur BASIC devra s'implanter en mémoire vive.

1-1- Avantages du langage machine par rapport au BASIC:

Le langage BASIC est interprété,c'est-à-dire que durant l'éxécution d'un programme, chaque instruction est examinée pour déterminer sa nature et les paramètres qu'elle contient,puis elle est éxécutée.Cette phase de recherche ralentit l'éxécution du programme,surtout si l'instruction à exécuter se trouve dans une boucle FOR....NEXT, parce qu'elle est décodée à chaque passage dans la boucle.

Les instructions de branchement(GOTO , GOSUB ...) ralentissent également l'exécution du programme,puisque l'interpréteur doit parcourir la mémoire où est stoké le programme pour trouver l'endroit où se trouve la ligne vers laquelle on désire se brancher.

Pour diminuer le temps d'excécution d'un programme,on doit utiliser le langage machine qui est directement compréhensible par le microprocesseur.Cependant,la programmation en langage machine n'est pas une chose facile;pour cela,au lieu d'écrire tout le programme dans ce langage,il est souvent plus aisé de l'écrire en BASIC puis de remplacer les boucles de traitement les plus longues par des petits sous-programmes en langage machine appelés par la fonction USR.

On peut dire qu'unordinateur passe preqque 99% de son temps à préparer xxxune opération et 1% seulement à l'éxecuter;donc en évitant cette phase de préparation,on gagne une rapidité de plusieurs dizaines de fois.

Un autre avantage du langage machine sur le BASIC est que: le programme en langage machine occupe moins de place (moins d'octets) en mémoire qu'un programme écrit en BASIC.

On peut résumer ainsi les avantages du langage machine:

- éxécution plus rapide du programme.
- utilisation plus efficace de l'espace mémoire.

1-2- Inconvénients du langage machine:

La programmation en langage machine est réservée auxprogrammeurs qui ont acquis une certaine expérience en programmation puisqu'elle demande la connaissance de la structure interne du microprocesseur (différents registres utilisateurs,unité de contrôle,UAL ...).

Les erreurs de syntaxe du BASIC peuvent être signalées par le programme moniteur tandis que cellesdu programmeur en langage machine peuvent provoquées l'effacement complet de la mémoire en bloquant complétement le système,ce qui en traine la perte complète du travail effectué si on n'avait pas pris la précaution de sauvegarder le programme sur une bande magnétique.Cet inconvénient est très important vu qu'il n'échappe pas aux programmeurs débutants en langage machine.

On peut ainsi résumer les inconvénients du langage machine:

-programmes difficiles à lire et à mettre au point.
- prégrammes non adaptables à d'autres ordinateurs.
- programmes nécessitant plus de lignes.
- difficultés d'effectuer les calculs arithmétiques compliqués.

2- ETUDE DU LANGAGE D'ASSEMBLAGE:

2-1- Introduction:

Pour faire exécuter une instruction au microprocesseur,il suffit de lui envoyer son code;comme il est difficile de se rappeler à quel code correspond une instruction,d'autant plus qu'il ya plusieurs cent es d'instructions en langage machine(696 instrructions sur le Z80).

On a eu l'idée alors, de substituer à chaque code un mnémonique constitué de deux ou trois lettres faciles à retenir et qui renseigne sur le rôle de cette instru ction;ce langage a reçu le nom de " Langage d'assemblage ".

Ex: JP (Jump en anglais qui veut dire saut)

CP (Compare)

AND (et)

Chaque constructeur a défini des mnémoniques pour toutes les ins--tructions de son microprocesseur.

2-2- L'assembleur:

Les mnémoniques n'étant pas directement éxécutables par le microprocesseur, on utilise alors un programme appelé " ASSEMBLEUR " qui effectue la conversion entre les mnémoniques et les codes.

Pour le micro-ordinateur " ZX SPECTRUM + " ,il existe sur cas--sette "EDITEUR / ASSEMBLEUR " distribué par SINCLAIR (appelé "PROGRAMME GENS "); chaque cassette contient un éditeur de textes qui nous permet d'écrire nos propres programmes en utilisant les mnémoniques du Z80 et de créer ainsi ce qu'on appelle la version"source".

Une fois le texte écrit,nous pouvons l'assembler c'est-à-dire le traduire en code machine et on peut sauvegarder le programme sur une bande magnétique.

Avant la traduction,il procède dans une première partie à un certain nombre de vérifications qui permettent l'élimination immédiate des erreurs.

Comme le programme assembleur est écrit en langage machine, il doit être implanté en mémoire vive "RAM";cecé diminue l'espace mémoire destiné au programme utilisateur.

L'inconvénient des programmes en langage d'assemblage c'est qu'ils sont longs à écrire.

2-3- Compilateur et interpréteur:

Introduction:

L'éxécution d'un programme source écrit par l'utilisateur en langage évolué s'effectue grâce à un interpréteur où un compilateur. Ce sont des programmes qui font partie du système d'exploitation du micro-ordinateur; recevant en entrée le programme source,il le traitent différemment;

COMPILATEUR:

Le compilateur est un programme de traduction du langage source en langage machine exécutable par le microprocesseur.

La compilation est faite une fois pour toutes et c'est le programme objet éxécutable très rapidement qui est sauvegardé en mémoire.Ce code objet compilé occupe environ la moitié de l'espace occupé par le code source .

INTERPRETEUR:

Un interpréteur permet une éxécution immédiate du programme source sans conversion en langage machine,il contrôle l'éxécution du programme source,instruction par instruction,en activant pour chacune d'elles un sous-programme en langage machine.C'est le programme source,seul,accessible ,qui est stocké en mémoire.L'interprétation etéant effectués à chaque éxécution.

Comparaison: unprogramme interpréte sera plus long que compilé,par contre les corrections d'erreurs sont immédiates.

Avec un compilateur,il faudra recompiler après chaque correction et de plus ne peut être mis en mémoire morte;de par ses dimensions (40 KO environ)il limiterait trop l'espace mémoire utilisateur.

Par contre, l'interprêteur n'occupe pas environ 10 Ko et peut facilement être mis en ROM avec le système opérateur .

2-4 LE MICRO-ORDINATEUR " ZX SPECTRUM + " :

L'unité centrale du ZX SPECTRUM + est un microprocesseur Z80.

Ce micro-ordinateur utilise comme langage évolué le BASIC ; cependant la programmation en assembleur est aussi possible.

Caractéristiques téchniques :

- Mémoire (ROM) : 16 KO
- Mémoire (RAM) : 48 KO
- Fréquence d'horloge : 3,5 M HZ

Remarque : Tous les exemples qui sont donnés en annexe sont vérifiés sur ce micro-ordinateur, cependant, ils resteront valables sur d'autres types de micro-ordinateur equipés d'un microprocesseur Z80 comme unité centrale.

2 - 4 - 1 Microprocesseur Z80 :

La structure interne du Z80 est constituée par cinq parties essentielles : une unité de contrôle, un compteur de programme, un registre d'instruction, une unité arithmétique et logique et 24 registres utilisateurs.

Le microprocesseur Z80 est une machine capable d'éxécuter un certain nombre d'instructions (appelé jeu d'instructions).Ces instructions sont des opérations arithmétiques et logiques.

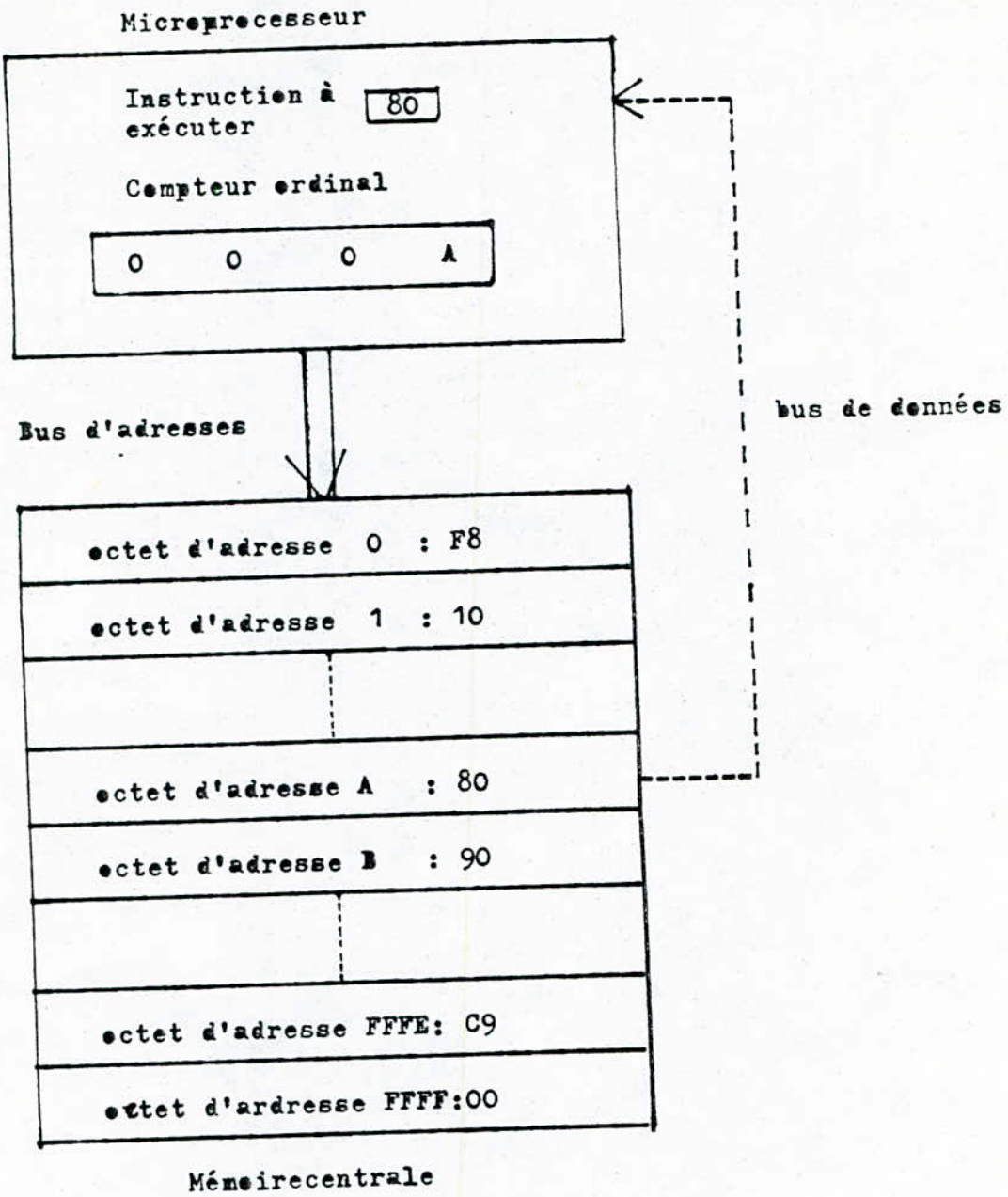
Le cycle complet d'exécution d'une instruction est le suivant.:

1. Le compteur ordinal est relié aux sorties adresses pour sélectionner l'octet contenant le code de l'instruction à exécuter.

2. L'octet seléctionné arrive sur le microprocesseur par le bus de données.

3. Le microprocesseur décode l'instruction et exécute l'action correspondante.

4. Le compteur ordinal s'incrémente pour préparer l'adresse de l'instruction suivante;



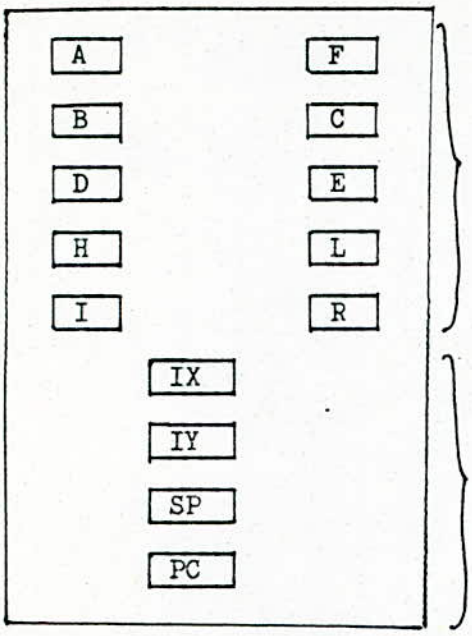
L'unité des commandes extrait de la mémoire centrale le code de l'instruction à exécuter(80)situé à l'adresse A indiquée par le compteur ordinal par l'intermédiaire de bus d'adresses. Le bus de données renvoie ce code au microprocesseur.Le décodeur de fonction interpréte ce code et l'unité de traitement et logique exécute cette instruction.

Le compteur ordinal s'incrémente de un pour préparer l'exécution de l'instruction suivante située à l'adresse mémoire B.

2-4-2- Les registres Z80:

Le microprocesseur dispose d'un certain nombre de registres qui sont des mémoires de travail ultra-rapides situées à l'intérieur de celui-ci.Le Z80 contient en tout 22 registres qui ont chacun des utilisations particulières.

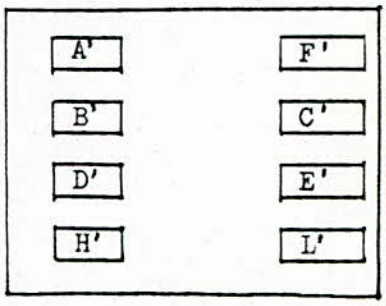
REGISTRES PRIMAIRES



Registres 8 bits

Registres 16 bits

REGISTRES SECONDAIRES



Registres 8 bits

Les registres A,B,C,D,F,H et L possèdent 8 bits et sont à usages généraux. On ne peut charger ou mémoriser ces registres qu'avec une donnée tenant sur un octet (0 à 255).

Le registre A est un registre particulier appelé accumulateur, dans lequel les calculs sont effectués.

Le registre B est souvant utilisé avec l'instruction DJNZ dans le programme comme compteur de boucle.

Les registres peuvent être regroupés deux à deux et former ainsi des registres de 16 bits BC, DE, HL. Le registre HL joue le rôle d'accumulateur pour les opérations arithmétiques sur 16 bits.

Le registre F est un registre contenant 6 indicateurs; selon les résultats des opérations qui ont été menées sur d'autres registres; chacun des bits de ce registre est modifié d'une certaine manière.

Les bits les plus usuels sont les bits 0,6,7 qui sont respectivement bits de carry, de zéro et de signe.

- Le bit 0 est l'indicateur de retenue, vaut 1 si la dernière opération sur le registre A á engendré une retenue et 0 sinon.

- Le bits 6 est l'indicateur de zéro, il est à un si le dernier résultat est zéro.

- Le bit 7 est l'indicateur de signe, vaut 1 si le bit 7 de A vaut 1 et 0 sinon.

Ces trois indicateurs sont souvent utilisés dans les branchements conditionnels.

Le registre I est exclusivement réservé pour gérer les interruptions.

Le registre R est utilisé pour le rafraichissement des mémoires; il ne peut-être utilisé comme registre utilisateur; réservé comme générateur de nombre aléatoire.

Les registres secondaires A', F', B', C', D', E', H', L' permettent seulement d'augmenter la capacité de mémorisation du microprocesseur. Ces registres ne sont pas manipulés par le Z80.

Les registres IX, IY sont des registres d'index à 16 bits et sont utilisés comme les autres registres classiques de 16 bits.

La pile :

Elle sert à l'utilisation de sous-programmes et de la sauvegarde temporelle de registres : PUSH (registre 16 bits) et POP (registre 16 bits) pour restaurer le registre. Elle fonctionne selon : la dernière donnée entrée est retirée la première, en anglais on appelle LIFO (Last In, First Out). Elle utilise le registre SP (Stack Poiter).

Remarque : Cette opération de sauvegarde de données dans la pile peut bloquer le micro-ordinateur en cas de mauvaise gestion de la pile.

Le dernier registre Z80 est PC (Program-Conter) s'appelle compteur ordinal; ce registre a été déjà évoqué dans le chapitre précédent.

2 - 5 Les modes d'adressages :

Représentation d'une instruction :

Pour représenter une instruction en code machine le Z80 utilise cinq zônes :

Adresse	Code	Etiquette	Mnémonique	Commentaires

- La première zône donne l'adresse mémoire de l'instruction en hexadécimal ou en décimal.

- La deuxième zône donne le code de l'instruction en hexadécimal.

- La troisième zône est réservée aux étiquettes dans le cas où on utilise les branchements dans le programme.

- La quatrième zône donne le mnémonique de l'instruction.

Et enfin, la cinquième zône est réservée aux commentaires afin de rendre lisible la fonction de l'instruction.

Ex :	Adresse	Code	Etiquette	Mnémonique	Commentaires
	30000	C68	-	ADD A,8	; Fait A= A+8

Les modes d'adressage indique la façon dont on doit prendre les données utilisées par une instruction. Pour bien assimiller l'utilisation de ces modes d'adressage, on prend comme exemple l'instruction de transfert de mnémonique LD (LOAD en anglais qui veut dire chargement).

L'instruction complète s'ecrit :

LD X,Y X,Y sont des opérandes de 8 ou 16 bits.

Cette instruction transfère la valeur Y dans X. Pour celà on appelle X : opérande destination, Y : opérande source.

2 - 5 - 1 Adressage immédiat :

L'opérande source est une contante de 8 ou 16 bits et l'opérande destination est un registre de 8 ou 16 bits.

Ex1 :	Adressage	Code	Mnémonique
	31000	3E08	LD A,8

Ainsi, le premier octet 3E indique le code de l'opération et le deuxième octet 08 correspond à la constante de 8 bits.

Ex2 :	30000	013A1E	LD BC, #1E3A

De même, le premier octet 01 indique le code de l'opération; tandis que le deuxième et le troisième octet donnent la valeur de la constante de 16 bits.

indique la notation héxadécimal.

Cette instruction de code opération 01 va charger le registre double BC par la valeur héxadécimale 1E3A.

L'ordre de stockage des deux octets de la valeur héxadécimale est le suivant:

L'octet de poids faible 3A est stocké à l'adresse 30001 et l'octet de poids fort 1E est stocké à l'adresse 30002.quant au code de l'instruction, il est stocké à l'adresse 30000;cette instruction occupera donc 3 octets en mémoire vive.

2-5-2- Adressage registre:

Les deux opérandes sont des registres de 8 bits.

ex: 78 LD A,B

Cette instruction transfère le contenu du regsitre B dans le registre A.

2-5-3- Adressage direct:

Dans ce type d'adressage,on donne l'adresse mémoire où se trouve l'opérande.

ex1: 3ABCDF LD A, (# DFBC)

Cette instruction charge le registre A avec la valeur de 8 bits située à l'adresse mémoire 57276.

ex2: 22BCDF LD (# DFBC), A

Cette instruction effectue le chargement inverse de la précédente.

ex3: 2ABCDF LD HL, (# DFBC)

Les contenus des adresses mémoire DFBC et DFBD sont tranférés dans le registre double HL.

2-5-4- Adressage indirect registre:

L'adresse de l'octet est contenue dans un registre double .

ex: 21409C LD HL,40000

7E LD A,(HL)

L'octet dont l'adresse mémoire est contenue dans le registre double HL est chargé dans l'accumulateur.Ces deux instructions peuvent se résumer en une seule qui est:

3A409C LD A, (40000)

2-5-5- Adressage indexé:

L' adresse de l'opérande est égale au contenu du registre d'index (IX ou IY) plus une valeur de 8 bits que l'on fournit.

ex: DD460A LD B, (IX+11)

Si le registre IX était chargé auparavant par la valeur 100.

DD2164 LD IX,100

Les deux instructions précédentes se résument en une seule:

LD B, (111)

mais cette instruction n'est pas compréhensible par le microprocesseur Z80.

2-5-6- Adresse relatif:

On utilise l'instruction de branchement relatif JR dans ce type d'adressage.L'instruction JR provoque un décalage positif ou négatif du compteur ordinal.

ex: JR $ +5

Cette instruction provoque un décalage de 5 octets en avant ($ valeur du compteur ordinal au début de l'instruction).

2-5-7- Agressage par bit:

Ce type d'adressage est utilisé pour les instructions opérantes sur 1 bit;il positionne en même temps les indicateurs.

ex: SET 5,B mise à 1 du bit 5 de B

RES 7,(HL) mise à 0 du bit 7 de (HL)

3-INSTRUCTIONS BASIC RELATIVES A L'UTILISATION DU LANGAGE MACHINE

3-1- Instruction POKE,PEEK :

Le BASIC du ZX SPECTRUM + utilise pour l'écriture d'un octet à une adresse mémoire l'instruction:POKE adresse, octet

ex; l'instruction:POKE 30000,255 est équivalente aux deux instructions en langage d'assemblage:

LD A, 255

LD (30000), 255

Pour la lecture d'un octet situé à une adresse mémoire,on utilise l'instruction: PEEK adresse,et pour afficher la valeur de l'octet à cette adresse par: PRINT PEEK adresse.

Si par exemple on veut lire l'octet situé à l'adresse 30000 (1er éxemple),on entre PRINT PEEK 30000 , et le micro-ordinateur affichera 255.

3-2- Exécution d'un programme en assembleur:

Pour exécuter un programme en langage d'assemblage,on utilise l'instruction; PRINT USR adresse,qui provoque le branchement du Z80 à l'adresse spécifiée.

Cette instruction renvoie le contenu du registre BC situé juste avant l'instruction RET qui le raménera au BASIC.

Remarque: pour les exemples de programmes donnés en annexe,le résultat de l'opération est toujours placé dans le registre BC juste avant l'instruction RET afin de pouvoir l'afficher(voir annexe).

4- CHARGEMENT D'UN PROGRAMME EN LANGAGE MACHINE EN UTILISANT LE BASIC:

Lorsqu'on ecrit un programme en assembleur , la phase suivante est de traduire les mnémoniques de chaque instruction et de le mettre en mémoire.Un assembleur fait génèralement cette opération automa-tiquement ,mais si on ne dispose pas de l'assembleur,on peut alors effectuer nous-mêmes la traduction de chaque instruction par son code

et de la placer en mémoire par l'instruction BASIC:POKE adresse,octet.

Cette méthode s'avère difficile surtout si les programmes à traduire sont longs et demande beaucoup de temps et d'attention pour l'entrée des codes en mémoire.

Remarque: les exemples de multiplication,de division et de l'addition des N premiers nombres qui sont donnés en annexe sont écrits une fois en langage d'assemblage,une autre fois traduits en langage machine et placés en mémoire en utilisant le BASIC.

5- MESURE DU TEMPS D'EXECUTION D'UN PROGRAMME:

Afin de vérifier la rapidité de l'assembleur sur le BASIC,il est nécessaire de savoir le temps d'exécution d'un même programme écrit une fois en langage d'assemblage et une autre fois en BASIC.

Pourcela,il existe une méthode très précise pour mesurer le temps.Elle utilise le contenu de certaines positions de mémoire qui s'incrémentent avec le fonctionnement d'une horloge.

L'horloge du ZX SPECTRUM+ a une fréquence de 3,5 MHZ , soit une période T égale à 0,2857μs et les positions de mémoire qui affectent cette horloge sont: 23672,23673,23674. Pour lire les contenus de ces adresses, on utilise l'instruction:PEEK adresse du BASIC;et pour avoir le temps en secondes écoulées depuis que l'ordinateur a été mis en fonction, on utilisera l'instruction:

```
PRINT (PEEK 23672+256.PEEK23673+65536.PEEK23674)/50
```

La case mémoire 23672 s'incrémente de 1 chaque 20ms; c'est le plus petit temps qui puisse être détecté.Par contre,le plus grand tempsest de 3 jours et 21 heures,moment auquel l'horloge revient à zéro.

~~revient.~~

Pourle BASIC,cette méthode est très efficace car le langage évolué doit être interprété chaque fois qu'il est exécuté.

Comme on avait évoqué au début de ce chapitre que le langage d'assemblage est traduit par l'assembleur une fois pour toutes en langage machine avant qu'il soit exécuté.Alors,son exécution est tellement rapide (de l'ordre da la microseconde)que l'horloge ne peut detecter;pour y remédier,on a eu l'idée de faire une boucle (100 environ) afin de augmenter le temps d'exécution.

Il existe une deuxième méthode pour calculer le temps d'exécution d'un programme en assembleur:c'est que pour chaque instruction correspond un nombre de cycles d'horloge mis par le Z80 pour l'exécuter. Ainsi,l'instruction la plus courte du microprocesseur Z80 dure 4 T ,soit 1,143 μs et la plus longue 23T soit 6,571 μs . Il suffit donc de calculer le nombre de cycles total du programme pour avoir son temps d'exécution;mais cette deuxième méthode n'est pas précise.

Ainsi,on montre pour l'addition des N premiers nombres que l'as-sembleur va environ 650 fois plus vite que le BASIC (voir annexe).

6- UTILISATION DES ROUTINES DU PROGRAMME MONITEUR:

Pour effectuer des opérations arithmétiques, on a toujours besoin de la multiplication et de la division;ces deux opérations n'existent pas sur le Z80.On est alors amené à faire des sous-programmes de multiplication et de division en virgule fixe (voir annexe);les calculs sont donc faits sur des entiers seulement,d'où une perte de précision.

Pour augmenter la précision dans les calculs(travailler en virgule flottante),on a eu l'idée d'utiliser des routines qui sont stockées en mémoire ROM à unecertaine adressse. ces routines sont écrites en langage machine par le constructeur et elles sont spécia-lisées pour diverses fonctions.

On peut alors avoir accès à ces routines en utilisant l'ins-truction RST adresse, qui est équivalente à CALL adresse.

Le moniteur dispose de 8 routines,mais nous nous interessons qu'à deux seulement:RST #10 et RST# 28 qui sont respectivement des routines d'affichage de caratères et de calculs en virgule flottante.

6-1- Routine d'affichage de caractères:RST#10

Pour afficher un caractère sur l'écran, il suffit de mettre son code dans le registre A et d'appeler la routine d'affichage de caract-tères par RST#10.

ex: pour afficher " GOTO " il suffit d'écrire le programme suivant:

```
ORG 44000        ;débute le programme à l'adresse 44000
LD A,236         ;code de "GOTO"dans le registre A
 RST #10
  RET
```

Avant de lancer l'exécution de ce petit programme,il est néces-saire d'introduire la position où on veut afficher ce caractère par le BASIC suivant:

```
10    PRINT    AT  11,15 ; " " ;
20    RANDOMISE  USR 44000 ; exécute le programme situé à
                               l'adresse 44000
```

Après éxécution,lemicro-ordinateur affichera au milieu de l'écran: GOTO .

6-2- Routine des calculs en virgule flottante:

Le RST # 28 permet l'accès au sous-programme qui contrôle les calculs en virgule flottante situé dans la mémoire ROM du micro-ordinateur. Cette routine extrait les octets qui suivent RST#28 dont la valeur correspond à une opération, un calcul;ces octets sont contenus dans une table.L'extraction des octets continue jusqu'à la rencontre de #38 qui DEFB de FIN des calculs.

Les codes des routines les plus interessantes sont donnés par le tableau suivant:

ROUTINE	Code opération(en héxa)
-	03
*	04
/	05
↑	06
+	0F
SIN	1F
COS	20
TAN	21
ASN	22
ACN	23
ATN	24
LN	25
EXP	26
SQR	28
ABS	2A

Cette routine de contrôle exploite les nombres,en virgule flottante,présents dans la pile du calculateur.Pour les y placer, on fait appel à une routine située à 11560.

On fait en trer un nombre de 8 bits(X) par:

```
LD A,X
CALL # 2D28
```

Pour un nombre de 16 bits(Y):

```
LD BC , Y
CALL # 2D2B
```

Une fois le calculateur a terminé las calculs qui lui sont demandés,le résultat se trouve dans la pile du calculateur.Nous pouvons le transférer dans le registre BC à l'aide d'une routine située à l'adresse 11682(#2DA2).

Un exemple est donné en annexe illustre bien l'utilisation de cette routine de calculs en virgule flottante.

Pour l'utilisation des graphiques en assembleur,voir annexe.

C H A P I T R E III

L O G I C I E L D E C O M M A N D E

1. REGULATEURS:

1-1 Introduction:

Dans beaucoup de cas d'application,on utilise des régulateurs STANDARds (régulateurs P,PI,PD,PID) en raison de la simplicité de l'algorithme de réglage et des expériences acquises dans le domaine des réglages continus.

1-2 Choix du régulateur:

Le choix et le dimensionnement d'un régulateur dans les circuits de réglage sont fondamentaux.Ce choix dépend non seulement de l'objet de commande mais aussi du paramètre que l'on veut contrôler et des conditions posées en régime établi.

Il existe beaucoup de théories permettant le choix et le dimensionnement d'un régulateur (compensation des pôles, lieu des racines ...).

Ainsi,par exemple si on exige que l'écart de réglage soit nul, il est indispensable que le régulateur possède une action intégrale (I), c'est-à-dire prendre un régulateur du type I , PI ou PID.

l'écart de réglage s'annule alors en régime établi pour une grandeur de consigne constante.

Dans le cas où le système à régler possède un comportement intégral,il n'est pas nécessaire que le régulateur présente l'action intégrale;on peut donc choisir un régulateur PD.

Exemple: pour la commande en position d'un moteur à courant continu ayant pour transmutance

$$H(p)=\frac{K}{p(1+Tmp)}$$

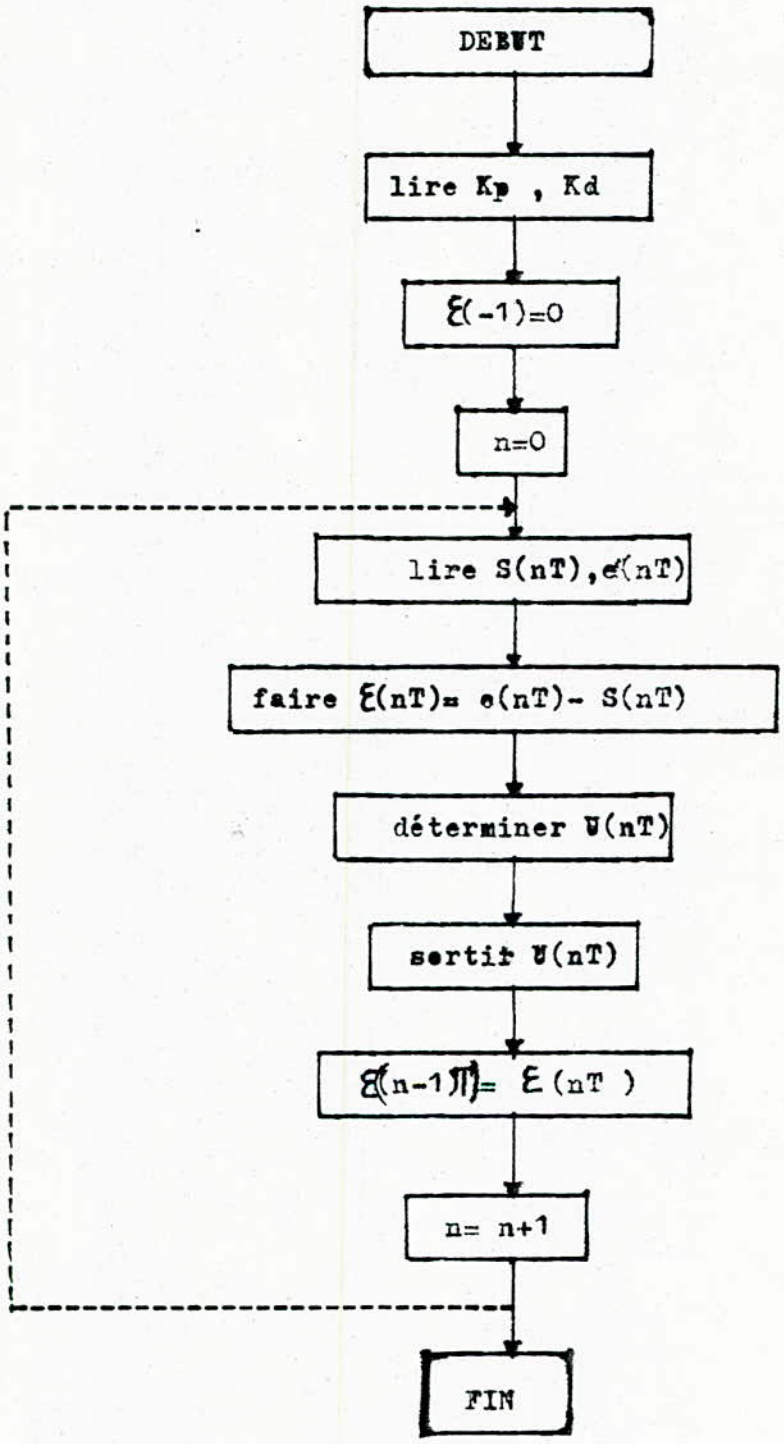
On choisira le régulateur PD et pour sa commande en vitesse,on choisira le régulateur PID.

1-3- Régulateur (PD):

Relation de base:

$$U(nT) = Kp.\varepsilon(nT) + Kd.\ (\varepsilon(nT) - \varepsilon((n-1)T))$$

Organigramme:



Programme (PD)

```
                ORG 44000
                LD IX,1000      LD A,0      ε(-1)=0
                LD (30850),A
H4              LD A,0                      lécture de S
                IN A,(63)
                NEG
                LD HL,30126                 consigne:C
                LD E,(HL )
                ADD A,E                     fait ε=C-S
                LD D,0
                LD E,A
                LD A,(30130)
                CALL MULT
                LD (30200),HL
                LD (30250),DE
                LD H,D                      calcul de U(nT)
                LD L,E
                LD BC,(30850)
                ADD A,0
                SBC HL,BC
                LD D,H
                LD E,L
                LD A,(30160)
                CALL MULT
                LD BC,(30200)
                ADD HL,BC
                LD BC,31
                LD A,L
                OUT (C),A                   sortie de U(nT)
                LD HL ,(30250)
                LD (30850),HL               ε((n-1)T)=ε(nT)
                DEC IX
                LD (30760),IX
                LD BC,(30760)               refaire l'éxécution tant que
                LD A,0                            IX≠0
                CP B
                JR NZ ,H4
                CP C
                JR  NZ ,H4
                RET                         ;Sous-programme demultiplication
MULT            LD HL,0
                LD B,8
  MO            ADD HL,HL
                SLA A
                JR NC,M1
                ADD HL,DE
  M1            DJNZ MO
                RET
```

Exécution par le programme BASIC suivant:

```
10  INPUT " Kp=";Kp
20  INPUT " Kd=";Kd
30  POKE 30130,Kp
40  POKE 30160,Kd
50  INPUT " C=";C
60  POKE 30126,C
70  PRINT(PEEK 23672+256.PEEK23673+65536.PEEK23674)/50
80  RANDOMISE USR 44000
90  PRINT(PEEK 23672+256.PEEK23673+65536.PEEK23674)/50
```

Temps d'exécution du programmme (PD):

$T_e \simeq 0{,}36$ ms

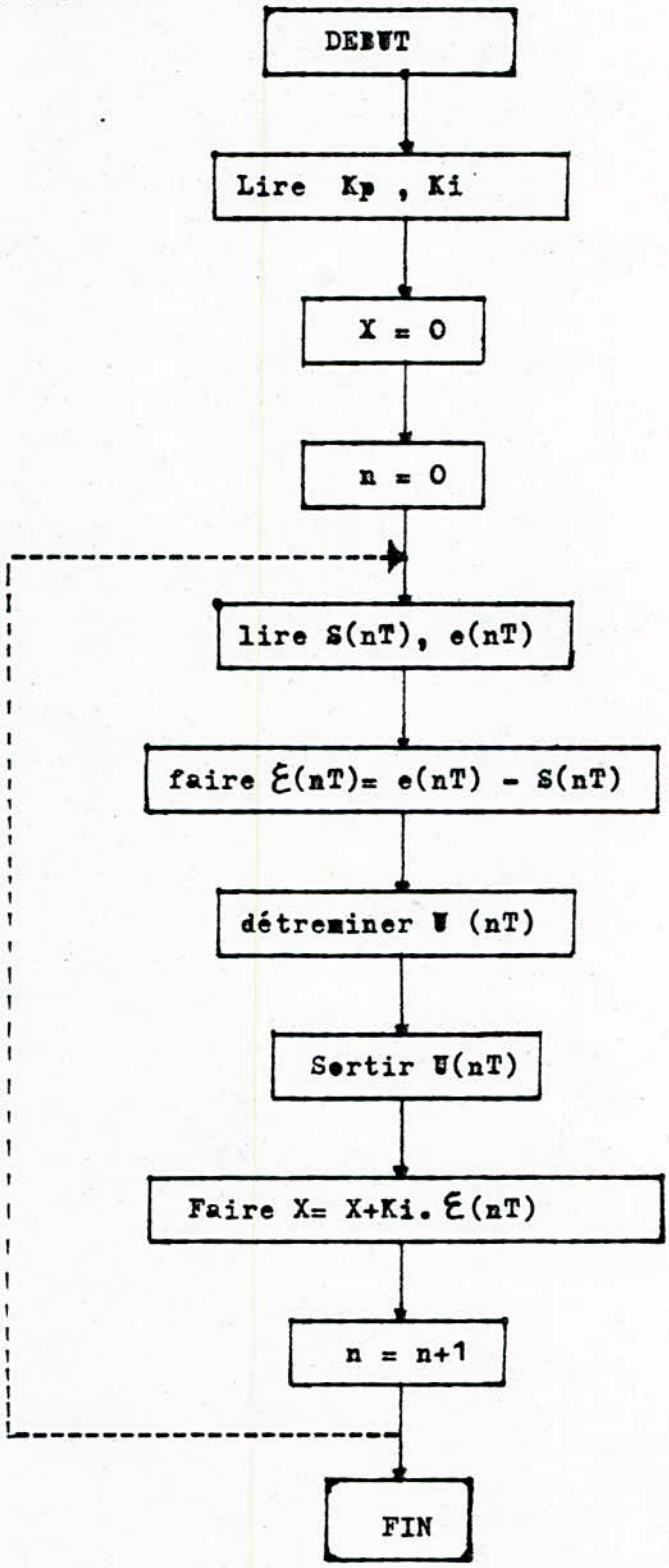
1-4- Régulateur (PI):

Relation de base:

$$U(nT) = Kp.\varepsilon(nT)+Ki.\sum_{k=0}^{n}\varepsilon(Kt)$$

ou $\begin{cases} U(nT) = Kp\cdot\varepsilon(nT)+X+Ki;\ \varepsilon(nT) \\ X=0 \quad , \quad X= X+Ki.\varepsilon(nT) \end{cases}$

Organigramme:



Programme: COMMENTAIRES

```
        ORG 44000
        LD IX,1000
        LD HL,0          ┐
        LD (30950),HL    ┘  X=0
H4      LD A, 0          ┐
        IN A,(63)        ┘  lécture S
        NEG
        LD HL,30126      ..... consigne:C
        LD E,(HL)
        ADD A , E        ..... fait ε=c-S
        LD D ,0          ┐
        LD E, A          │
        LD A,(30130)     │
        CALL MULT        │
        LD (30200),HL    │
        LD A,(30150)     │
        CALL MULT        │
        LD BC,(30950)    │  calcul deU(nT)
        ADD HL,BC        │
        LD (30800),HL    │
        LD BC,(30200)    │
        ADD HL,BC        ┘
        LD BC,31
        LD A,L
        OUT (C),A        ..... sortie deU(nT)
        LD HL ,(30800)   ┐
        LD (30950),HL    ┘  fait X=X+Ki.ε(nT)
        DEC IX           ┐
        LD (30760),IX    │
        LD BC ; (30760)  │
        LD A,0           │
        CP B             │  reprendre l'éxecution ,tant que IX≠0
        JP NZ ,H4        │
        CP C             │
        JP NZ , H4       │
        RET              ┘
MULT    LD HL ,0         ┐
        LD B ,0          │
  MO    ADD HL,HL        │
        SLA A            │  sous-programme de multiplication
        JR NC ,M1        │
        ADD HL,DE        │
  M1    DJNZ MO          │
        RET              ┘
```

Exécution par le programme BASIC:

```
10  INPUT "Kp = " ; Kp
20  INPUT "Ki = " ; Ki
30  POKE 30130 , Kp
40  POKE 30150 , Ki
50  INPUT "C=";C
60  POKE 30126 , C
70  PRINT (PEEK 23672 + 256.PEEK
    23673+65536.PEEK 23674 )/50
80  RANDOMISE USR 44000
90  PRINT (PEEK 23672 + 256.PEEK
    23673+65536.PEEK 23674 )/50
```

Temps d'exécution du programme "PI":

Te = 0,36 ms

1-5- Régulateur (PID):

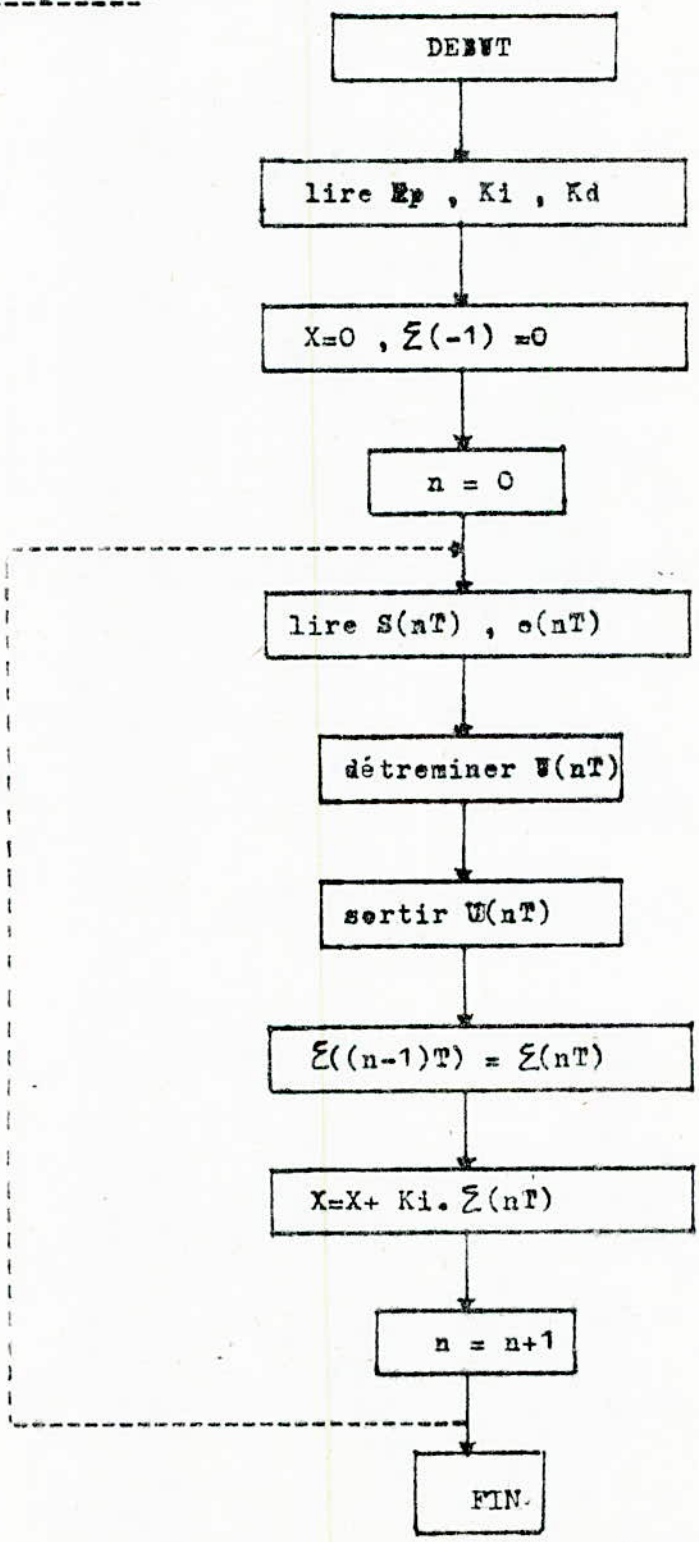
Relation de base:

$$U(nT) = Kp.\,\varepsilon(nT) + X + Ki.\,\varepsilon(nT) + Kd.(\varepsilon(nT) - \varepsilon((n-1)T))$$

$$\varepsilon(-1)=0 \quad , \quad \varepsilon((n-1)T) = \varepsilon(nT)$$

$$X = 0 \quad , \quad X = X + Ki.\,\varepsilon(nT)$$

Organigramme:



PROGRAMME (PID):

```
        ORG 44000
        LD IX,1000
        LD HL,0
        LD (30950),HL      LD (30850),HL
        LD A,0
H4      IN A,(63)
        NEG
        LD HL,30126
        LD E,(HL)
        ADD A,E
        LD D,0
        LD E,A
        LD A,(30130)
        CALL MULT
        LD (30200),HL
        LD A,(30150)
        CALL MULT
        LD BC,(30950)
        ADD HL,BC
        LD (30800),HL
        LD BC,(30200)
        ADD HL,BC
        LD (30700),HL
        LD (30250),DE
        LD H,D
        LD L,E
        LD BC,(30850)
        ADD A,0
        SBC HL,BC
        LD D,H
        LD E,L
        LD A,(30160)
        CALL MULT
        LD BC,(30700)
        ADD HL,BC
        LD BC,31
        LD A,L
        OUT (C),A
        LD HL,(30800)
        LD (30950),HL
        LD HL,(30250)
        LD (30850),HL
        DEC IX
        LD (30760),IX
        LD BC,(30760)
        LD A,0
        CP B
        JP NZ,H4
        CP C
        JP NZ,H4
        RET
MULT    LD HL,0
        LD B,8
HO      ADD HL,HL
        SLA A
        JR NC,H1
        ADD HL,DE
H1      DJNZ HO
        RET
```

COMMENTAIRES:

..... Compteur de boucle

..... X=0

..... $\varepsilon(-1)=0$

..... lécture de S

..... consigne: C

..... fait $\varepsilon=C-S$

calcul de U(nT)

..... sortie de U(nT)

..... X=X + Ki. ε(nT)

..... $\varepsilon((n-1)T)=\varepsilon(nT)$

reprendre l'éxécution du programme tant que IX≠0

sous-programme de multiplication.

Exécution par le programme BASIC suivant:

```
 10  INPUT "Kp=";Kp
 20  INPUT "Ki=";Ki
 30  INPUT "Kd=";Kd
 40  POKE 30130,Kp
 50  POKE 30150,Ki
 60  POKE 30160,Kd
 70  INPUT "C=";C
 80  POKE 30126,C
 90  PRINT(PEEK 23672+256.PEEK 23673+65536.PEEK 23674)/50
100  RANDOMISE USR 44000
110  PRINT(PEEK 23672+256.PEEK 23673+65536.PEEK 23674)/50
```

Temps d'exécution du programme (PID):

Te = 0,5 ms

1-6- Régulateur optimal:

C'est un régulateur qui rend le temps de réponse minimal; il est totalement déterminé par l'objet que l'on veut contrôler.

Prenons comme exemple,la commande de position d'un moteur à courant continu et à excitation indépendante ,ayant les caractérsi-tiques suivantes:

Excitation: tension : 12 V

COURANT :0,25 A

Coefficient de f.c;e.m :Ke=0,053 MKSA

Induit : resistance: 4,2 Ω

moment d'inertie(total):Jt= $133,8.10^{-6}$ $Kg.m^2$

coefficient de couple: Kc=0,053 MKSA

Constante du moteur:

$$Km=\frac{1}{K\,e}= 18,87$$

Constante de temps mécanique:

$$Tm=\frac{R.Jt}{Ke.\,Kc}= 0,2\ s$$

La tranmitance d'un tel moteur est donnée par:

$$H(p)=\frac{K}{P(1+Tmp)}$$ (commande en position)

La transformée en Z du régulateur optimal est donnée par:

$$D(Z)=\frac{b_2 Z^2 + b_1 Z + b_0}{a_2 Z^2 + a_1 Z + a_0}$$

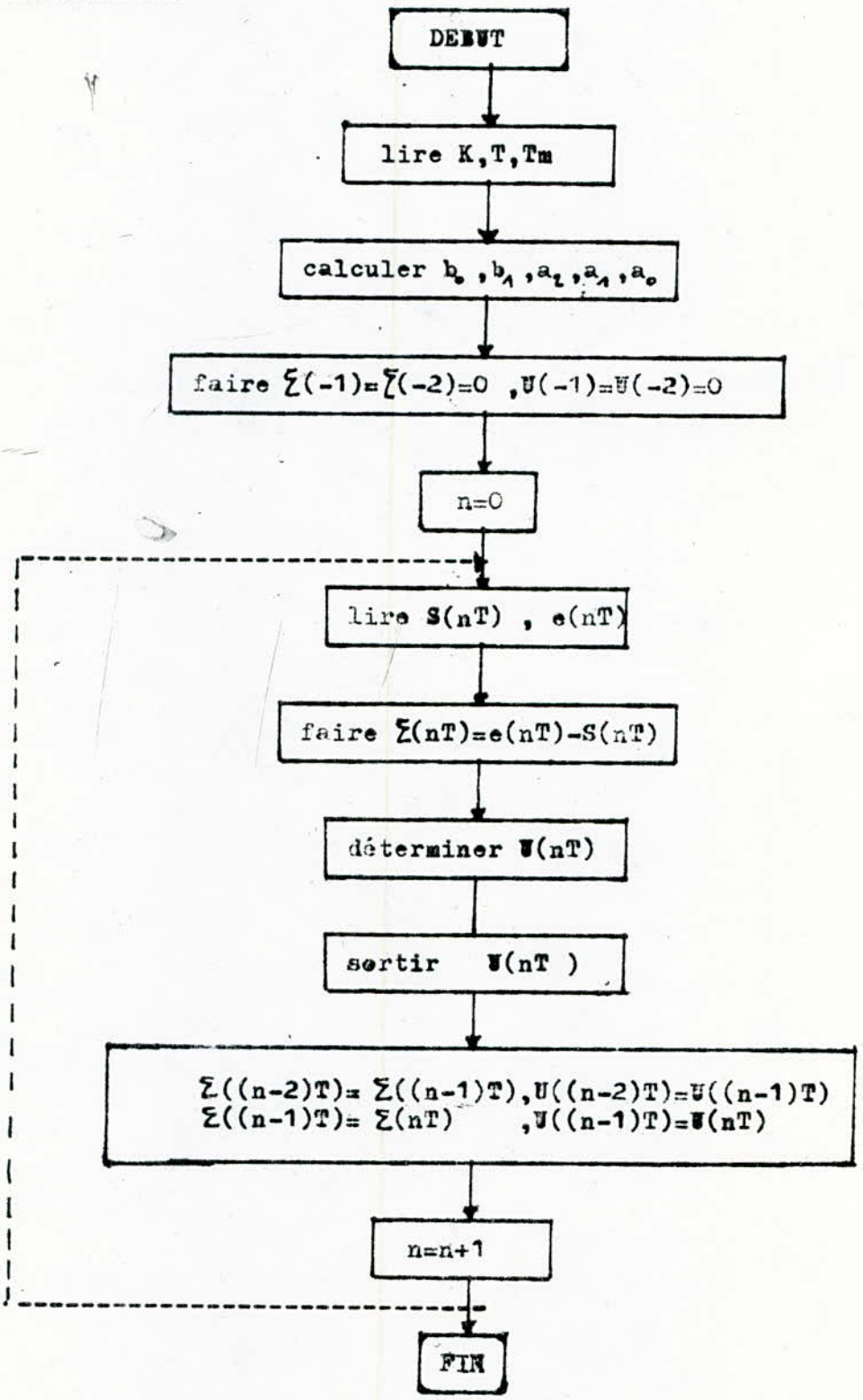
avec $b_2 = 1$
$b_1 = -(Z_0 + 1)$
$b_0 = Z_0$
$a_2 = K.T(1-Z_0)$
$a_1 = -K(T+TmZ_0 - Tm)$
$a_0 = -K(Tm-TmZ_0-TZ_0)$

et $z_0 = EXP\ (-T/Tm)$
$T = 0,1\ s$
$Tm = 0,2\ s$
$K = 1,428$

Algorithme de réglage:

$$U(nT)=(\ \varepsilon(nT)+b_1\cdot\varepsilon((n-1)T)+\ b_0\cdot\varepsilon((n-2)T)-a_1\cdot U((n-1)T)-a_0\cdot U((n-2)T))/a_2$$

Organigramme:



Programme:(régulateur optimal)

```
        ORG 44000
        LD IX,1000
        LD A,0
        LD (30108),A ..... Σ(-1)=0
        LD (30116),A ..... Σ(-2)=0
        LD (30124),A ..... U(-1)=0
        LD (30132),A ..... U(-2)=0
H8      LD A,0
        IN A,(63) ..... lécture de S
        NEG
        LD HL,30140 ..... consigne:C
        LD E,(HL)
        ADD A,E ..... Σ=C-S
        LD (30600),A
        LD A,(30132) .....
        CALL #2D28
        LD A,3
        CALL #2D28
        LD A,100
        CALL #2D28
        RST #28
        DEFB 5
        DEFB 4
        DEFB #38
        LD A,(30124)
        CALL#2D28
        LD A,3
        CALL #2D28
        LD A,100
        CALL #2D28
        RST #28
        DEFB 5
        DEFB 4
        DEFB #0F
        DEFB #38
        LD A,(30116)
        CALL#2D28
        LD A,6
        CALL#2D28
        LD A,10
        CALL #2D28
        RST #28
        DEFB 5
        DEFB 4
        DEFB #0F
        DEFB #38
        LD A,(30600)
        CALL #2D28
        LD A,(30108)
        CALL#2D28
        LD A,16
        CALL#2D28
        LD A,10
        CALL #2D28
        RST#28
        DEFB 5
        DEFB 4
        DEFB 3
        DEFB#0F
        DEFB#38
        LD A,6
        CALL#2D28
        LD A,100
        CALL #2D28
        RST#28
        DEFB 5
        DEFB 5
        DEFB #38
        CALL#2DA2 .....
        LD A,C
        LD BC,31
        OUT (C),A ..... sortie de U(nT)
        LD HL,(30108) ..... Σ((n-2)T)=Σ(n-1)
        LD(30116),HL .....
        LD HL,(30600) ..... Σ((n-1)T)= Σ(nT)
        LD (30108),HL .....
        LD HL,(30124) ..... U((n-2)T)=U((n-1)
        LD (30132),HL .....
        LD HL,(30880) ..... U((n-1)T)=U(nT)
        LD (30124),HL .....
        DEC IX
        LD (30700),IX
        LD BC,(30700)
        LD A,0
        CP B
        JP NZ,H8
        CP C
        JP NZ,H8
        RET
```

calcul de U(nT)

reprendre tant que IX≠0

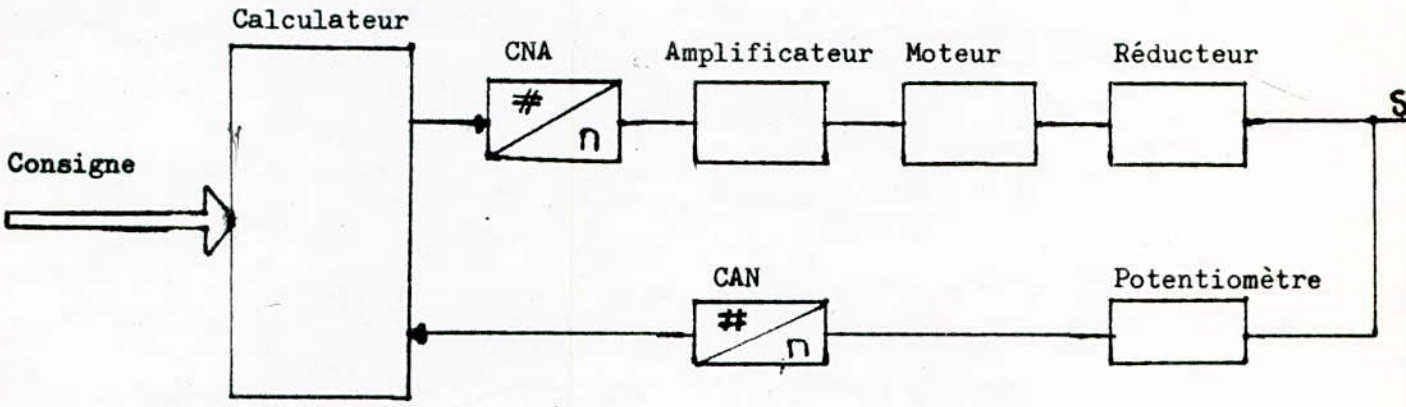
L'exécution se fait par le programme BASIC:

```
10 INPUT "C=";C
20 POKE 30140 ,C
30 PRINT(PEEK 23672+256.PEEK 23673+65536.PEEK 23674)/50
40 RANDOMISE USR 44000
50 PRINT(PEEK 23672+256/PEEK 23673+65536.PEEK 23674)/50
```

2 - INFLUENCE DU TEMPS DE CALCUL SUR LA QUALITE DE REGLAGE

Cette influence sera montrée à partir d'un exemple de commande en position d'un moteur à courant continu et à excitation indépendante dont les caractéristiques ont été mentionnées précedemment.

Le schéma de principe est le suivant :



La commande de ce moteur a été déjà faite antérieurement (voir /6/ de la bibliographie). Le langage utilisé était le BASIC, le régulateur est du type (PD), la constante de tempsmécanique du moteur était Tm = 0,2s.

La période d'échantillonage T relative au critère de " SHANNON " doit vérifier la condition : $T \leqslant \frac{Tm}{2} = 0,1s$, sa limite inférieure est le temps de calcul t_c. Les deux conditions se résument donc en une seule : $t_c \leqslant T \leqslant 0,1s$

En faisant varier le temps de calcul relatif ($Er = \frac{t_c}{T}$) pour des valeurs de T constantes, on peut voir les comportements du système d'après les réponses indicielles.

De plus, on exige au système les performances suivantes :

- Réponse du système la plus rapide.
- Amortissement relatif optimum (de 5 %)
- Stabilité

Les réponses indicielles correspondant aux critères de performances précedentes en fonction du temps de calcul relatif Er sont données par les figures 1 à 4 (pour plus de détails voir (9) de la bibliographie).

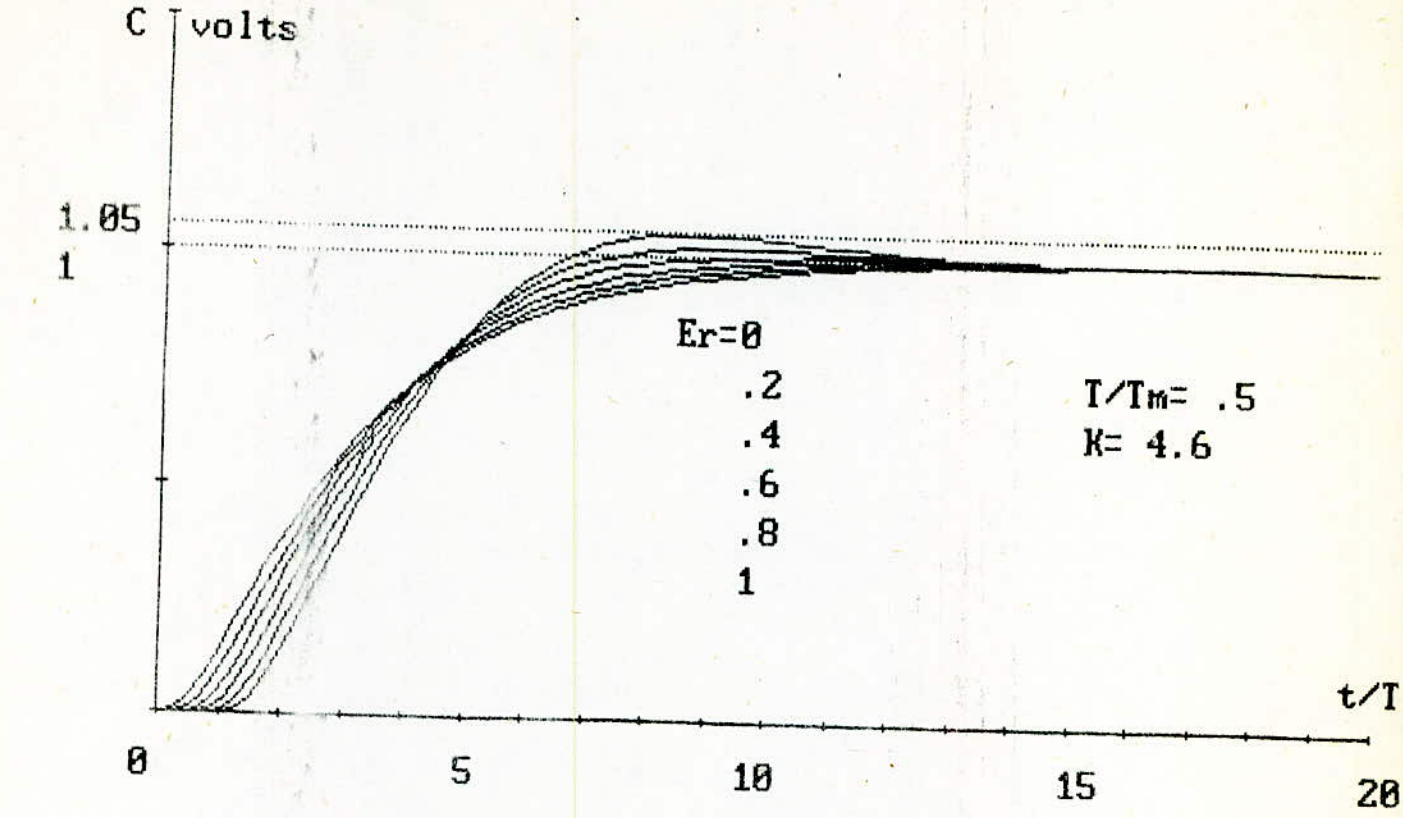

C volts
1.05
1
Er=0
.2
.4
.6
.8
1
T/Tm= .5
K= 4.6
t/T
0
5
10
15
20

fig:1

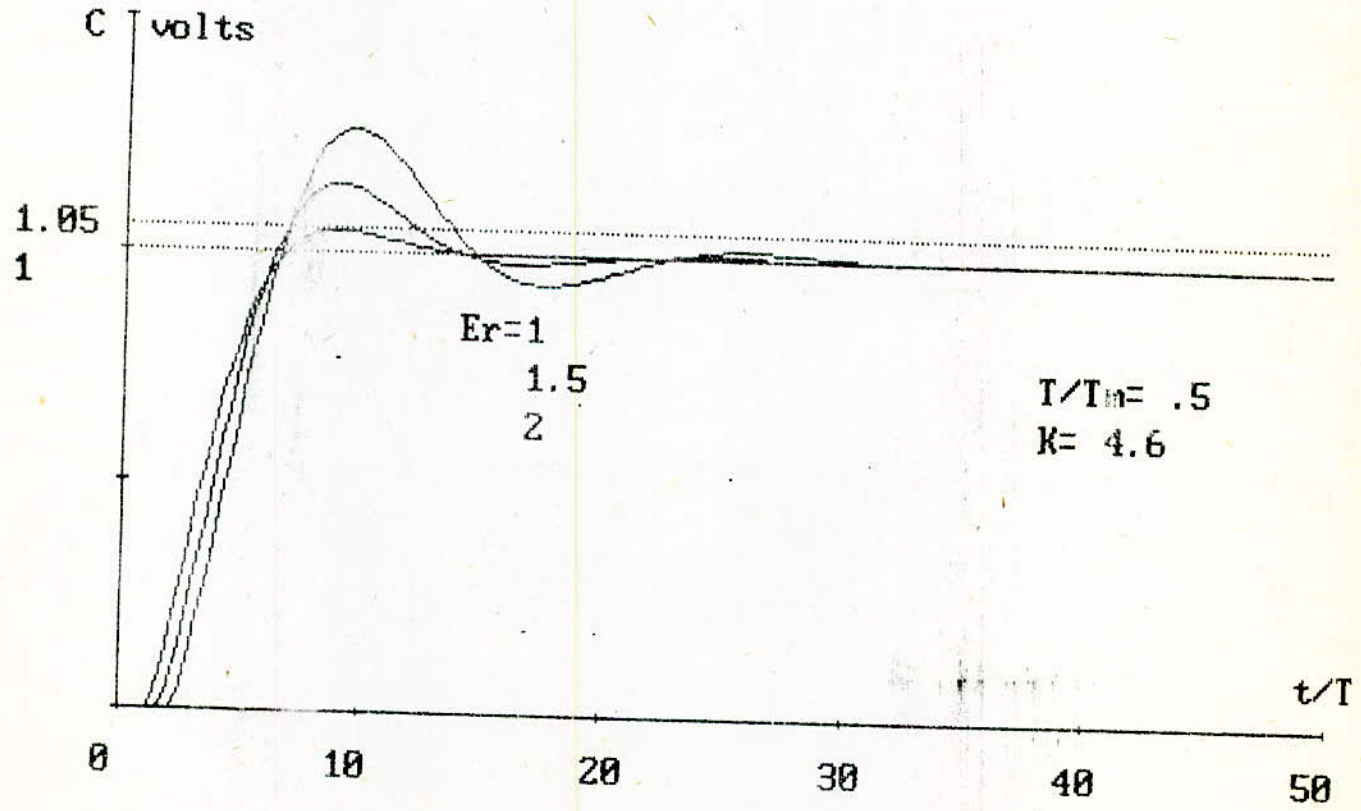

C volts
1.05
1
Er=1
1.5
2
T/Tm= .5
K= 4.6
t/T
0
10
20
30
40
50

fig : 2

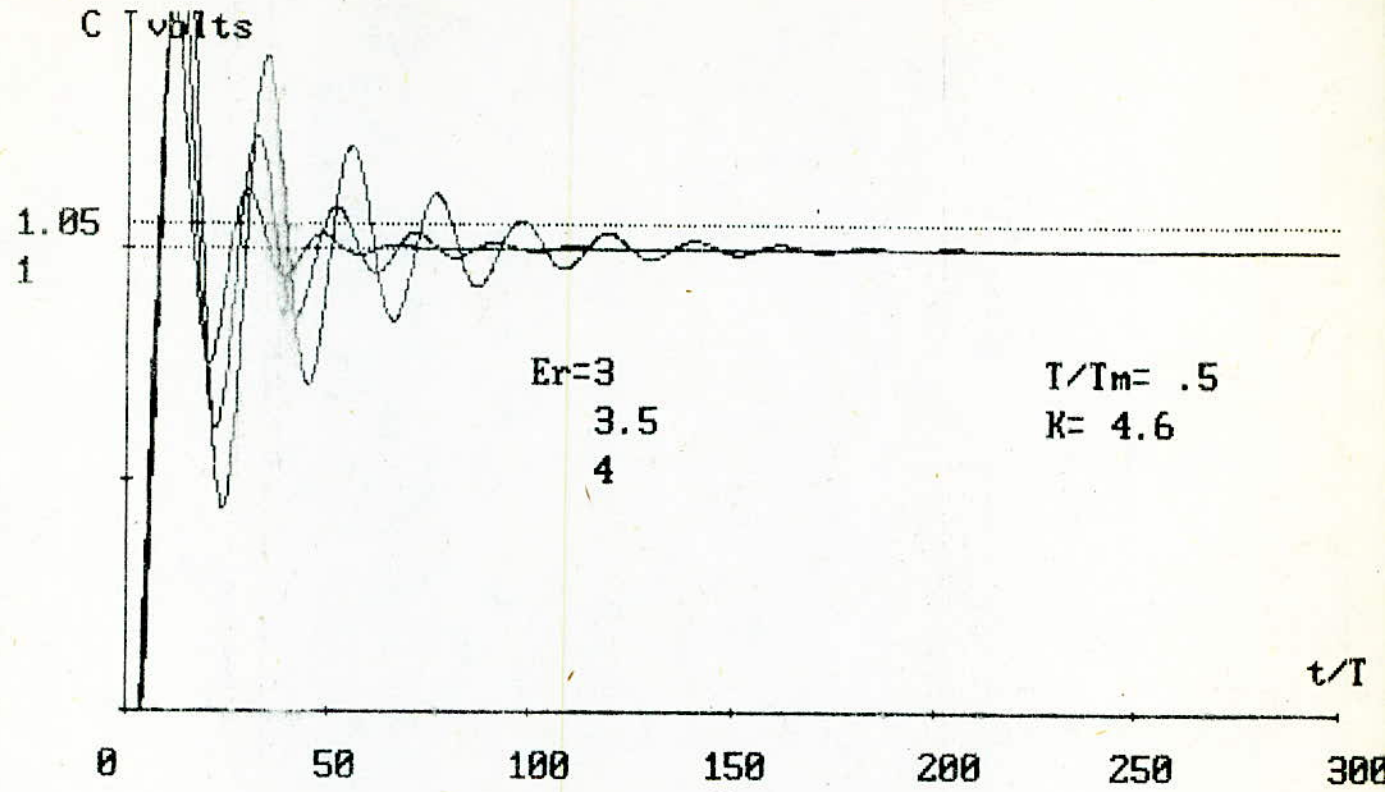
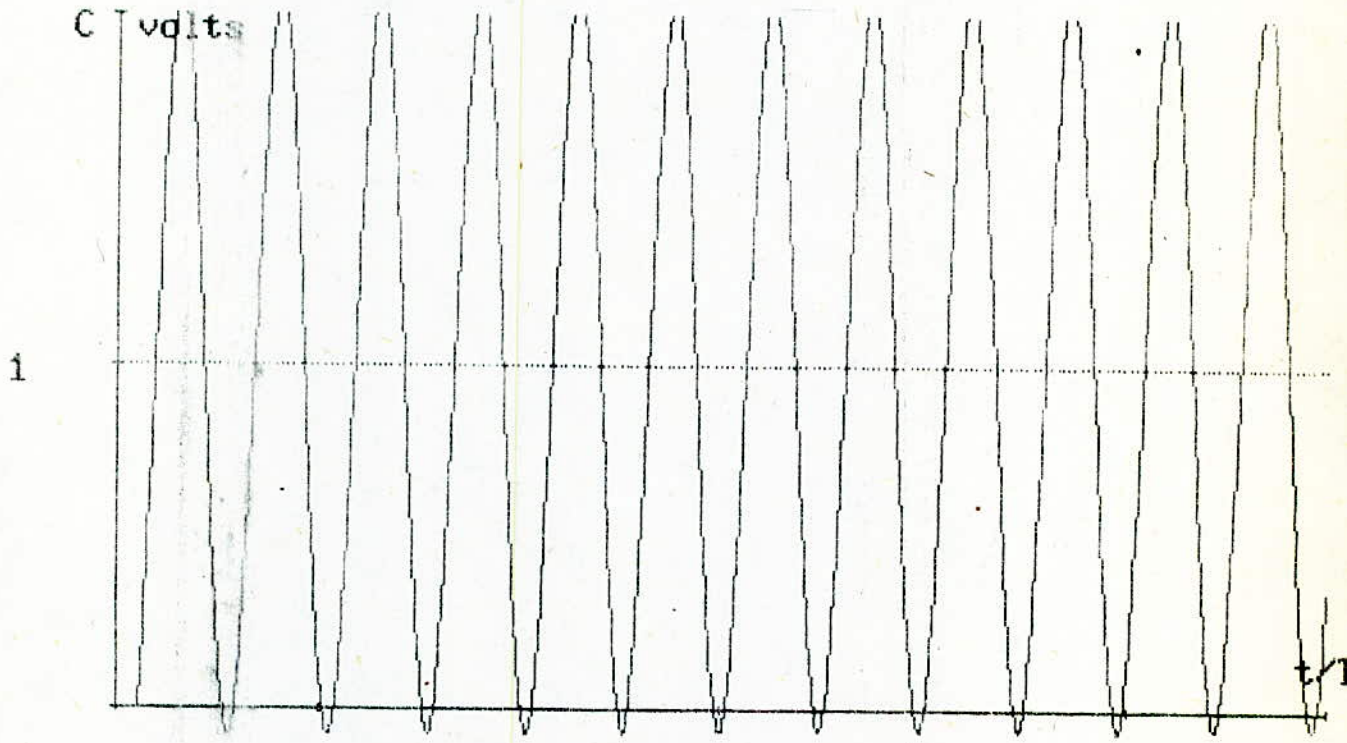


fig:3



T/Tm=.5 ;K=4.6 ; Er=5.15

fig:4

INTERPRETATION DES COURBES :

- Les meilleurs réponses sont celles qui sont données par la figure 1 puisque tous les critères de performance exigés sont obtenus. De plus, le critère de " SHANNON " est vérifié ($T \leq 0,1s$).

- Les figures 2 et 3 montrent les pertes de plus en plus des performances du système (système lent, dépassement) en fonction du temps de calcul.

- Quant à la figure 4, elle présente la limite supérieure du temps de calcul (Er = 5,15) où le système perd totalement les performances souhaitées.

Afin de comparer les avantages entre l'assembleur et le BASIC dans la commande numèrique par calculateur, on est amené à comparer leur temps de calcul sur un même micro-ordinateur (ZX SPECTRUM +).

Les tâches demandées au micro-ordinateur sont :

T1 : Calcule et envoie les grandeurs de commande sur les actionneurs.

T2 : Visualisation des courbes (consigne, sortie).

COMPARAISON :

			T1	T2	T1 + T2
$T_e = 10$ ms	BASIC	t_c (ms)	37	60	97
		$E_r = \frac{t_c}{T_e}$	3,7	6	9,7
	Assembleur	t_c (ms)	0,36	50	50,36
		$E_r = \frac{t_c}{T_e}$	0,036	5	5,36

On remarque que : - Les graphiques sont lents pour les deux types de langage.

- Seulement l'assembleur permet la commande de la tâche 1 en respectant les performances exigées.

- Le système est incontrolable par le BASIC (pour T T = 10 ms).

- L'assembleur va environ 100 fois plus rapide que le BASIC.

Pour T = 60 ms, on aura :

			T1	T2	T1 + T2
T = 60 ms	BASIC	t_c(ms)	37	60	97
		$E_r = \frac{t_c}{T}$	0,6	1	1,6
	Assembleur	t_c(ms)	0,36	50	50,36
		$E_r = \frac{t_c}{T}$	0,006	0,83	0,839

- La commande de la tâche 1 est possible pour les deux langages; par contre, les deux tâches ne sont commandables simultannement qu'avec l'assembleur.

Pour T = 100 ms.

			T1	T2	T1 + T2
T = 100 ms	BASIC	t_c(ms)	37	60	97
		$E_r = \frac{t_c}{T}$	0,37	0,6	0,97
	Assembleur	t_c(ms)	0,36	50	50,36
		$E_r = \frac{t_c}{T}$	0,0036	0,5	0,503

Dans ce cas, les deux tâches sont réalisées par les deux types de langage.

CONCLUSION

Il résulte de cette application que :

-Pour atteindre des performances encore plus élevées dans la commande numérique par calculateur, il devient nécessaire d'utiliser l'assembleur comme langage de programmation .

-Le choix de la période d'echantillonnage, du langage de programmation et du calculateur dépend essentiellement de la rapidité du phénomène à contrôler et des performances souhaitées .

-Plus les tâches à exécuter sont nombreuses ,plus elles nécessitent une capacité de calcul suffisante (capacité mémoire , nombre de bits par mot , rapidité). L'utilisation alors ,de plusieurs calculateurs permettra une grande souplesse dans la résolution de ces tâches .

A N N E X E

Adresse	Code		Mnémonique	Commentaire
7530	-		ORG 30000	;Adresse de dé-but
7530	3AF875		LD A, (30200)	;Place N dans A
7533	47		LD B, A	;Compteur de boucle.
7534	210000		LD HL, 0	
7537	1600		LD D, 0	
7539	58	H1	LD E, B	
753A	19		ADD HL, DE	; fait S=S+I
753B	10FC		DJNZ H1	
753D	44		LD B, H	; Resultat placé dans le registre BC.
753E	4D		LD C, L	
753F	C9		RET	; Fin et retour au BASIC

L'execution de ce programme se fait à partir d'un petit programme en BASIC permettant d'entrer le nombre N :

```
10    INPUT "N="; N          ; Entrée du nombre N
20    POKE  30200, N
30    LET U= USR 30000       ; éxécute le programme situé à l'adresse 30000.
                               et renvoie le contenu du registre BC.
40    PRINT "U="; U
```

La somme des N premiers nombres peut se faire par le programme BASIC suivant :

```
10    INPUT "N="; N
20    LET S= 0
30    FOR I= 1 TO N
40    LET S= S+I
50    NEXT I
60    PRINT "S="; S
```

On peut déterminer le temps d'exècution de ce programme une fois ecrit en assembleur et une autre fois écrit en BASIC et les comparer .

Comme le temps d'exécution du programme en assembleur est inferieur à 20 ms (temps d'incrémentation de l'horloge), on est amené alors à faire une exécution en boucle afin d'augmenter le temps .

Le programme en assembleur éxécuté en boucle de 1000 est le suivant :

```
        ORG  30000
        LD IX, 1000      ;Compteur de boucle
H2      LD A, (30200)
        LD B, A
        LD HL, 0
        LD D, 0
H1      LD E, B
        ADD HL, DE
        DJNZ H1
        DEC IX
        LD (30110), IX
        LD BC, (30110)
        LD A, 0
        CP B
        JR NZ, H2
        CP C
        JR NZ, H2
        LD B, H
        LD C, L
        RET
```

Reprendre l'exécution du programme tant que IX ≠ 0.

Place le resultat de l'opération dans le registre BC et FIN du programme.

On éxécute les deux programmes (assembleur, BASIC) successivement afin de comparer leur temps d'exécution .

```
10      INPUT "N="; N
20      POKE 30200, N
30      PRINT "ASSEMBLEUR"
40      PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
50      LET U = USR 30000
60      PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
70      PRINT "U ="; U
80      PRINT "BASIC"
90      PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
100     FOR T = 1 TO 1000
110     LET S = 0
120     FOR I = 1 TO N
130     LET S = S+I
140     NEXT I
150     NEXT T
160     PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
170     PRINT "S ="; S
```

EX : N = 10

Boucle de répitition = 1000

aprés éxécution, le micro-ordinateur affichera :

```
ASSEMBLEUR
Te = 0,16 s
U = 55
BASIC
Te = 107 s
S = 55
```

Soit environ 650 fois plus rapide que le BASIC pour cet exemple .

Il éxiste une méthode permettant de programmer en langage machine; elle consiste à faire charger en mémoire RAM les codes machines de chaque instruction du programme.

La traduction de chaque instruction en code de machine est assurée par l'utilisateur (jouant le rôle de l'assembleur).

Remarque : Cette methode est fastidieuse et demande plus d'attention à l'entrée des codes machines .

Le programme précédent deviendra :

```
10      REM "ADDITION N PREMIERS NOMBRES"
20      DATA 58, 48, 117, 71, 33, 0, 0, 22, 0, 88, 25, 16, 252, 68,
            77, 201
30      FOR J = 0 TO 15
40      READ C
50      POKE (33333 + J), C
60      NEXT J
70      INPUT "N ="; N
80      POKE 30000, N
90      LET U = USR 33333
100     PRINT "U ="; U
```

EXEMPLE 2 : "Multiplication d'un nombre tenant sur deux octets (16 bits) par un nombre de 8 bits .

Le multiplicande est placé dans le registre double DE à l'adresse mémoire 30000.

Le multiplicateur est placé dans le registre A (accumulateur) à l'adresse mémoire 30100.

Adresse	Code		Mnemonique
7D00	-		ORG 32000
7D00	ED5B3075		LD DE, (30000)
7D04	349475		LD A, (30100)
7D07	210000		LD HL, 0
7D0A	0608		LD B, 8
7D0C	29	H0	ADD HL, HL
7D0D	CB27		SLA A
7D0F	3001		JR NC, H1
7D11	19		ADD HL, DE
7D12	10F8	H1	DJNZ H0
7D14	44		LD B, H
7D15	4D		LD C, L
7D16	C9		RET

Pour éxécuter ce programme en assembleur, on utilise le BASIC pour faire entrer le multiplicande et le miltiplicateur.

```
10      INPUT "D ="; D          ;- multiplicande
20      INPUT "A ="; A          ;- multiplicateur
30      POKE 30000, D
40      POKE 30100, A
50      LET U = USR 32000
60      PRINT "U ="; U
```

Temps d'éxécution du programme :

Programme en assembleur éxécuté en boucle de 1000 :

```
                ORG 32000
                LD IX, 1000
H3              LD DE, (30000)
                LD A, ( 30100)
                LD HL, 0
                LD B, 8
H0              ADD HL, HL
                SLA A
                JR NC, H1
                ADD HL, DE
H1              DJNZ H0
                DEC IX
                LD (30200), IX
                LD BC, (30200)
                LD A, 0
                CP B
                JR NZ, H3
                CP C
                JR NZ, H3
                LD B, H
                LD C, L
                RET
```

_ Chargement en mémoire des codes machines du programme précédent:

```
10  DATA 237,91,48,117,58,148,117,33,0,0,6,8,41,203,39,48,1,25
       ,16,248,68;77,201
20  FOR J=0 TO 22
30  READ C
40  POKE (32000+I),C
50  NEXT J
60  INPUT "D=";D          ; Multiplicande
80  INPUT "A=";A          ; Multiplicateur
90  POKE 30000,D
100 POKE 30100,A
110 LET  K=USR 32000
120 PRINT "K=",K
```

Exécution successive des deux programmes afin de comparer leur temps d'éxécution :

```
 10      INPUT "D ="; D
 20      INPUT "A ="; A
 30      POKE 30000, D
 40      POKE 30100, A
 50      PRINT "ASSEMBLEUR"
 60      PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
 70      LET U = USR 32000
 80      PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
 90      PRINT "U ="; U
100      PRINT"BASIC"
110      PRINT (PEEK 23672 + 256. PEEK 23673 +65536. PEEK 23674)/50
120      FOR T = 1 TO 1000
130      LET M = D.A
140      NEXT T
150      PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
160      PRINT "M ="; M
```

EX : D = 255
A = 255
Boucle de 1000

Le micro-ordinateur affichera :

```
ASSEMBLEUR
Te = 0,18 s
U = 65025
BASIC
Te = 9,32 s
M = 65025
```

L'assembleur va environ 50 fois plus vite que le BASIC pour le cas de la multiplication .

EXEMPLE 3 : " DIVISION ENTIERE "

Le programme suivant éxécute une division entière :

- d'un nombre 16 bits (Allant à 65535) placé dans un registre HL .
- Par un nombre se tenant sur 8 bits (allant à 255) placé dans l'accumulateur A.

Adresse	Code		Mnémonique	
7530	-		ORG 30000	
7530	219475		LD HL, 30100	] Placement du dividende dans le registre HL.
7533	66		LD H, (HL)	
7534	3A9575		LD A, (30101)	
7537	6F		LD L, A	]
7538	3A9E75		LD A, (30110)	→ Placement du diviseur dans A.
753B	ED44		NEG	
753D	4F		LD C, A	
753E	06FF		LD B, 255	
7540	110000		LD DE, 0	
7543	09	H1	ADD HL, BC	
7544	3004		JR NC, H2	
7546	13		INC DE	; DE contient le quotion
7549	C34375		JP H1	
754A	42	H2	LD B, D	] Place le quotion dans le registre BC.
754B	4B		LD C, E	
754C	C9		RET	

Chargement en mémoire des codes machines du programme précédant :

```
10    DATA  33, 148, 117, 102, 58, 149, 117, 111, 58, 158, 117, 237, 68, 79, 6,
            255, 17, 0, 0, 9, 48, 4, 19, 195, 67, 117, 66, 75, 201.
20    FOR J = 0 TO 28
30    READ  C
40    POKE (30000+J), C
50    NEXT J
60    INPUT "H ="; H
70    INPUT "L ="; L
80    INPUT "A ="; A
90    POKE 30100, H
100   POKE 30101, L.
110   POKE 30110, A
120   LET DE = USR 30000
130   PRINT "DE ="; DE
```

HL contient le dividende

A contient le diviseur

EX. : 36000/36 = 1000 (en décimal)

en héxadécimal on aura : 36000(D) = 8CA0Hex

A = 36(D) = 24Hex soit H = 8C = 140 (D)

L = A0 = 160 (D)

A = 24 = 36 (D)

Après RUN le micro-ordinateur affichera : DE = 1000

Les temps d'éxécution des programmes (assembleur et BASIC) sont donnés après éxécution du programme suivant :

```
10    INPUT "L ="; L
20    INPUT "A ="; A
30    POKE 30100, 0
40    POKE 30101, L
50    POKE 30110; A
60    PRINT "ASSEMBLEUR"
70    PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
80    LET U = USR 30000
90    PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
100   PRINT "U ="; U
110   PRINT "BASIC"
120   PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
130   FOR T = 1 TO 10000
140   LET D = L/A
150   NEXT T
160   PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
170   PRINT "D ="; D
```

EX. : L = 250
A = 5
Boucle de 10000

Après RUN, on aura : ASSEMBLEUR
Te = 3,34 s
U = 50
BASIC
Te = 117,38
D = 50

L'assembleur va environ 35 fois plus vite que la BASIC dans le cas de la division .

EXEMPLE 4 : " CALCULS EN VIRGULE FLOTTANTE "

```
 10    ORG 44000
 20    LD BC, 550
 30    CALL # 2D28
 40    LD A, 13
 50    CALL # 2D28
 60    RST # 28
 70    DEFB 5
 80    DEFB # 38
 90    CALL # 2DA2
100    RET
```

Ce petit programme effectue la division du nombre 550 par 13. On peut afficher le resultat par l'instruction : PRINT USR 44000

Le résultat exact de cette division est 42,30; par contre le resultat affiché est 42 .

Cela se justifie par : Le registre BC ne peut contenir que des valeurs entières comprises entre 0 et 65535 .

Pour avoir la valeur exacte qui se trouve dans la pile du calculateur, on la transférè dans une variable qui aura été déclarée dans un programme BASIC et éxécuté .

Ainsi, pour avoir le resultat exact de la division précédante, il suffit d'élliminer les lignes 90 et 100 et d'ajouter les lignes suivantes :

```
106    LD HL, (23627).      ; 23627 est l'adresse de VARS
108    INC HL
110    LD (HL), A .
120    INC HL
130    LD (HL), E
140    INC HL
150    LD (HL), D
160    INC HL
170    LD (HL), B
180    INC HL
190    RET
```

Ce programme séxécute à partir du programme BASIC :

```
10    LET U = 0                    ; Première valeur de VARS.
20    RANDOMISE USR 44000          ; Lance l'exécution du programme
                                     situé à l'adresse 44000.
30    PRINT "U =";U
```

Le résultat affiché est alors 42,3076

LES GRAPHIQUES

L'equivalent du PLOT X,Y du BASIC :

En assembleur, il suffit de placer l'abscisse X dans le registre C et l'ordonnée Y dans le registre B et d'appeller une routine chargée des graphiques située à l'adresse 8927 (#22DF) par CALL #22DF .

EX. 5 : " TRACE D'UNE DROITE D'EQUATION Y = X "

Le Programme BASIC qui permet de tracer cette droite est le suivant :

```
10    FOR X = 0 TO 170
20    LET Y = X
30    PLOT X , Y
40    NEXT  X
```

Programme en assembleur équivalent :

```
          ORG 44000
   X      EQU 0
          DEFB X
   H4     LD A, (44000)
          CALL # 2D28
          LD A, 1
          CALL # 2D28
          RST # 28
          DEFB 4
          DEFB # 38
          CALL # 2D28
(44018)   RET
44019) GRAPHE LD D, 0
          LD E, 0
   H8     LD A, E
          LD (44000), A
          LD (30200), DE
          CALL H4
          LD B, C      ; met Y dans B
          LD A, E
          LD C, A      ; met X dans C
          CALL # 22DF  ; Appel de la routine des graphiques.
          LD DE, (30200)
          INC DE
          CP 170
          JR NZ, H8
          RET
```

Calcule Y = 1.X et place le résultat Y dans le registre BC.

L'éxécution du programme par RANDOMISE USR 44019 permet de tracer la droite Y = X

Remarque: On gagne un peu de temps dans le tracé avec le programme en assembleur .

EXEMPLE 6 : " TRACE D'UNE FONCTION SINUSOIDALE D'EQUATION S = 60.SIN (X/10) "

```
            ORG 44000
        X   EQU 0
       H4   LD A, 60
            CALL # 2D28
            LD A;(44000)
            CALL # 2D28
            LD A, 1
            CALL # 2D28
            RST # 28
            DEFB 5
            DEFB 4
            DEFB # 1F
            DEFB 4
            DEFB # 38
            CALL # 2D28
  (44031)   RET
(44032) GRAPHE
            LD D, 0
            LD E, 0
       H8   LD A, E
            LD (44000), A
            LD (30200), DE
            CALL H4
            LD B, C          ; met S dans B
            LD A, E
            LD C, A          ; met X dans C
            CALL # 22DF      ; Appel de la routine des graphiques.
            LD DE; (30200)
            INC DE
            LD A, E
            CP 253
            JR NZ, H8
            RET
```

Calcule S=60.SIN(X/10)
et place S dans le registre BC.

Exécution par : RANDOMISE USR 44032

Comparaison des temps d'éxécution des deux programmes (Assembleur et BASIC)

```
10   PRINT "BASIC"
20   PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
30   FOR X = 0 TO 253
40   LET S = 60.SIN (X/10)
50   PLOT X, S
60   NEXT X
70   PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
80   PAUSE 200
90   CLS
100  PRINT "ASSEMBLEUR"
110  PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
120  RANDOMISE USR 44032
130  PRINT (PEEK 23672 + 256. PEEK 23673 + 65536. PEEK 23674)/50
```

L'éxécution donne les temps suivants :

BASIC
Te = 15,64 s
ASSEMBLEUR
Te = 13,38 s

B I B L I O G R A P H I E:

/1/ "Structure et organisation
des systémes (calculatoires)".
Université d'Alger(département INFORMATIQUE)
(1974) VALERI A.DOLIATOVSKI

/2/ "Langage machine,trucs et astuces
sur ZX SPECTRUM".
Micro-ordinateurs. PASCAL PELIER

/3/ "La pratique du ZX SPECTRUM et SPECTRUM +".
Tome 2:Programmation en langage machine.
Micro-ordinateurs. MARCEL HENROT

/4/ "L'assembleur facile du Z80".
Micro-ordinateurs.OLIVIER LEPAPE

/5/ "Réglages échantillonnés".
Volume 1:Traitement par la transformation en Z.
Ecole polytéchnique de LAUSANNE.H.BUHLER

/6/ "Commande numérique d'un moteur à
courant continu par micro-ordinateur".
Projet de fin d'etudes(promotion:juin 1985).

/7/ "Les microprocésseurs".
(Structure et fonctionnement des circuits
intégrés programmables.)

/8/ "La commande des machines élèctriques
par microprocèsseurs".
J.F AUBRY,R.HUSSON,C.LUNG et G.PFITSHER

/9/ "Analyse de la qualité de réglage numérique ".
Projet de fin d'études(promotion:juin 1986).