الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ÉCOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTROTECHNIQUE

PROJET DE FIN D'ETUDES

SUJET

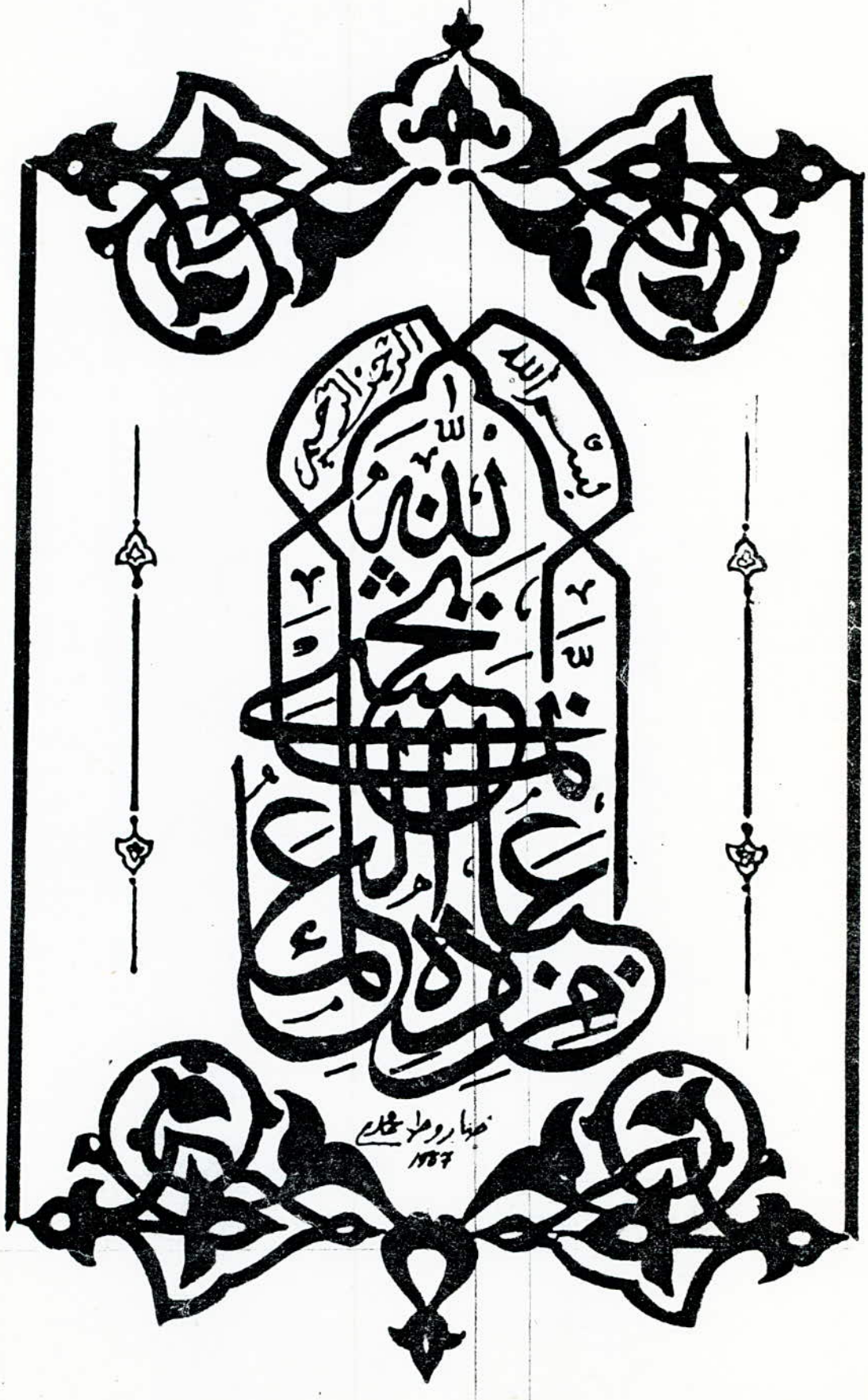
- CONTRIBUTION A L'ELABORATION - D'UN MANUEL DE TRAVAUX - PRATIQUES SUR CARTE MT-80Z -

Proposé par : Tchouketchkebir

Etudié par : OUADAH.A
SAROUTE.M

Dirigé par : Tchouk tchekir

PROMOTION : 01/88



(إنما يخشى الله من عباده العلماء).

CONTRIBUTION
A
L'ELABORATION
D'UN MANUEL
DE T-P
SUR MT.80Z

بسم الله الرحمن الرحيم

الإهداء

DEDICACES

Je dedie ce modeste travail en signe de reconnaissance a:

** Ma tres chere **MERE**
** Mon tres chere Pere si **ALLAL**
et **BACHIR**

* Mes freres et soeurs en particulier MOHAMED (KOUIDER).
* Mes amis en particulier:AISSA (B),BACHIR,MESSAOUD,AEK.
* Mon binome.

S.Mohamed.

اهدي هذا العمل المتواضع الى:

امي العزيزة: التي ضحت بكل شيء من أجلي لأبلغ ما بلغته والتي دربته إلى من ربّاني ودعمني معنويا وماديا وحرص كل الحرص على ان ابلغ هذه الغاية: امي العزيزة: علال فتحت

- A ma tres chere **Mere** , et a mon tres chere **Pere**
- A mes **Freres** et **Soeurs** ,temoignage de
  En temoignage de ma reconnaissance et d'affection.
- A mes **Proches** et **Amis** (es),
  convaincus de mes profonds sentiments.
- A mon **Binome.**
- Au **Miens** partout .

O.Abderrahim.

الى:

امي وابي: عرفانا بجميل صنيعهما بجلد لهما وثابرتهما في تعليمي.

الى الذين يذوبون كالشمع لينيروا الدرب للآخرين.

--------<<<***0***>>>--------

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

Nous remercions notre promoteur Monsieur TCHOUKETCH-KEBIR.A ,pour ses conseils,et ses encouragements durant l'elaboration de ce projet.

Que Mr ZOUAOUI EL-HADI (CEN), Mr IBTIOUEN (ENP), Mr Pr BARSKI (ENP), Mme DOUMA B.ZOULIKHA (ENP), trouvent ici l'expression de nos remerciements et notre profonde gratitude, pour nous avoir aides avec tant d'enthousiasme.

Nous tenons a adresser particulierement nos remerciement a Mr PODGOVSKY (ENP), Mr BOUCHERITE (ENP), Mme BENHAMZA et a Mr FEKKAR (Pdt CSI).

Nous remecions tous les enseignants qui ont contribuer de pres ou de loin a notre formation.

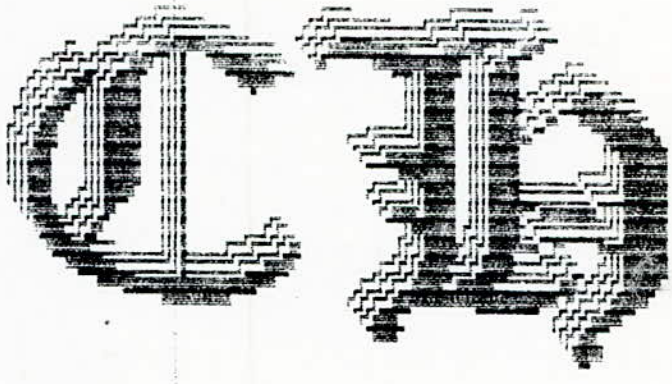
Nous tenons a remercier tous nos amis(es) qui nous ont tant aides de pres ou de loin a l'elaboration de ce travail et de le mettre sous cette forme.

Et en fin nous remeciements vont a tous les membres du CLUB SCIENTIFIQUE D'ELECTROTECHNIQUE et du CLUB INFORMATIQUE de l'ENP pour leur comprehension et leurs aides.

--------<<<***0***>>>--------

TABLE DES MATIERES

TABLES DES MATIERES



INTRODUCTION

INTRODUCTION

L'evolution de la technologie des circuits integres logiques pendant la decennie 1960/1970 a permis au fabriquants de composants de realiser une unite centrale simplifie d'ordinateur en un circuit.le uP qui comprent particulierement une unite de commande et une U.A.L(unite arithmetique et logique).(voir historique)

Notre projet de fin d'etudes est une etude comportant deux grandes parties:

1- Contribution a l'elaboration d'un manuel de T-P sur MT-80Z.
2- Etude des entrees-sorties en vue d'une commande eventuelle.

La carte MT80Z est un Kit fait a base d'un uP 8 bits, c'est le Z80 produit par ZILOG.Inc. D'ou l'interet manifeste au Z80 au debut de la premiere partie. Cette meme partie comporte les T-P, premier objectif de notre projet, ces derniers ont ete subdivises en plusieurs series pour des raisons dedactique; et ou chaque serie est constituee par plusieurs parties ceci est due aux volumes horaires resteints affectes aux travaux pratiques.

Quand a la seconde partie , qui comporte en premier lieu apres une introduction aux entrees-sorties une etude detaillee de la famile ZILOG (Peripherique du Z80),cette etude comporte la description et la programmation de chacun des composants et en fin nous avons donne quelques elements de la commande numerique d'un moteur a C-C alimente a partir d'un pont de Greatz.

Il est claire que nous avons voulu que ce modeste travail ne soit pas seulement un banal manuel de T-P, mais une timide introduction a la commande numerique, qui exige non seulement une parfaite connaissance des convertisseurs statiques et des machines electriques, mais aussi une maitrise des divers principes de la regulation (continue,discontinue...); et ce sans ometre l'organe principal qu'est le uP.

**********000**********

HISTOIRE DES MICROPROCESSEURS

Par rapport a ses predecesseurs (le boulier chinois ,la machine de pascal, par exemple) l'ordinateur a la particularite d'etre une machine a calculer programmable .En realite l'ordinateur ne sait pas calculer mais uniquement enchainer des operations elementaire combunees au sein de programmes.La succession rapide de ces operations simples,sans intervention humaine permet de realiser entre autres de puissant calculs,et la possibilite de les enchainer donne au traitement son caractere automatique.

*L'ordinateur est ne pour les besoins de la defense americaine lors de la seconde guerre mondiale.Il est compose de tubes et de lampes dont les principaux avantage etaient de facheuse tendances a la surchauffe et a la panne la derniere de cette espece fut construit en 1957.
*L'apparition du transistor apres sa mise au point des 1947 nous a permit de substantiels progres.Alors on pu disposer de cartes sur lequels se trouvaient des elements comme les diodes les transistors.etc... Les technique de fabrication et de miniautorisation des transistors amenerent certains constructeurs des les annees 1960/1965 a integrer dans de petit boitiers ce qu'il faut pour les fonctions logiques dans les calculateur au debut des annees 70 on arrivaient a integrer la totalite des fonctions logiques necessaire a un organe de traitement de l'information en un seul boitier,ainsi INTEL fabriqua en cette epoque le circuit integre 400 H qui reunissait toutes les fonctions de base d'un calculateur electronique le uP etait ne'.Il comprand principalement une unite de commande et une U-A-L (unite arithmetique et logique),en meme temps INTEL entreprend de devlopper un ensemble de circuit integres logiques complexes et a haute integration pour besoins specifiques des nouvelles calculatrices d'une societes specialisee (BUSICOM):c'est avec ces circuits que INTEL lance l'indistrie du uP vers la fin de l'annee 1971.Peut de temps apres 1972 ,INTEL presente le 8008.Il s'agit d'un uP PMOS (le 4004 est aussi en PMOS) qui taite des mots de 8 bits.Sa structure influenca fortement les constructeurs ainsi le 8008 marque le veritable coup d'envoi des uP.

*De1973 a 1878 de nombreux uP apparaissent sur le marche leurs capacité de traitement fixe a 8 bits leur structure fonctionnel est relativement uniforme .La technologie MOS a canal N (NMOS) qui est plus difficil a mettre au point et a fabriquer offre de meilleurs performances que le PMOS:un facteur au moins egal a 5.
* apartire de 78/79 des uP plus performants font leur apparitions(16 bits) et sont comparables a des mini ordinateur de haute gamme la technologie utilisee est une NMOS amelioree a haute performances (HMOS chezINTEL, XMOS chez national sem)
*Vers 1981 des uP plus performants (32bits) qui depassent les anciens et sont comparables aux gros ordinateurs.

LES GENERATIONS DE MICROPROCESSEURS:

*1r Generation: Sont ceux realises en PMOS.Ils sont peu performants et necessitent de nombreux boitiers auxiliaires les principaux types sont INTEL 4004 & 8008.Il effectuent une addition entre deux registres en 8 a 10 us.

*2nd Generation:(a partir 73),cette generation est marquee par la technologie NMOS la vitesse est d'environs 2 us pour l'addition entre registres,les pricipaux produits sont:INTEL 8008,MOTOROLA 6800,FAIRCHILD F8,(En 74/75) par la technologie NMOS amelioree ce qui a donnee des versions plus performantes (un facteur 1,5 a 2)des premiers produits tel que:INTEL 8080 A/A-1/A-2.MOTOROLA 68A00 et 68B00 et des produits nouveaux tels que ZILOG (Z80).8085 d'INTEL 6802 et 6809 de MOTOROLA.
*3 eme Generation:(A partir 79),avec la technologie NMOS au hautes performances ce qui a donne naissance a des uP de 16 bits et plus,avec une structure fonctionnelle de plus en plus complexe et une rapidite de traitement sensiblement egale a celle des produits 8 bits de nouvelle version ceci est du au fait que la technologie de fabrication est la meme en NMOS .Les produits typiques actuels sont l'INTEL 8086.MOTO 68000 ZILOG Z8000.

REMARQUE:

* Les produits de base sont ameliores entré ces generations. ce qui a fait profite ces produit des bienfaits de l'evolution technologique.
*L'apparition des vrais uP 32 bites marque probablement l'aube d'une 4 emme generation.

QUELQUE DOMAINES D'APPLICATION DES MICRO-PROCESSEURS:

L'automatisme indistriel:On peut distinguer dans ce domaine plusieur type d'application qui comprennent un ou plusieurs microprocesseurs agissant,en tant qu'unite centrale de traitement :

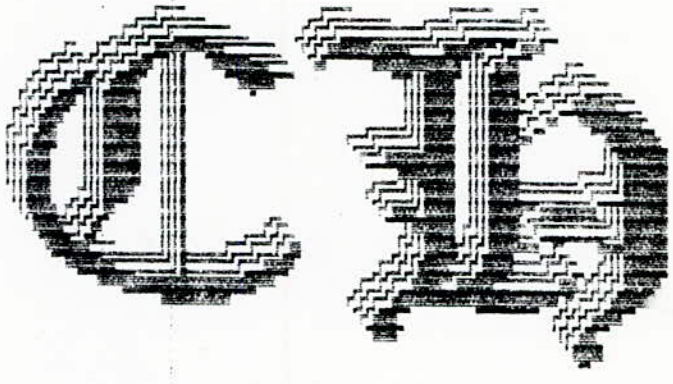
-Les systemes de production automatisee
-Les machines outils,
-Les robots indistriels
-Les protheses medicales destinees aux handicapees physiques.
-Les automates programmables

Dans un autre axe on remarque le remplacement de la logique cablee par une logique programmee c-a-d que le systeme logique cable est defini par le cablage des circuits execute par le concepteur par contre dans la logique programmee ce qui definit le fonctionnement c'est le programme ce qui se traduit par une plus grande souplesse de mise en point d'amelioration ulterieure apportee au fonctionnement...

-INDUSTRIELLE :c-a-d les capteurs les actionneurs...
-ELECTRIQUE et ELECTRONIQUE :oscilloscopes, multimetre.
-MEDICALE:appariels de diagnostique,de surveillance des patients
-SCIENTIFIQUE: analyseur de spectres,chronatograhes.

Dans ces domaines le microprocesseur permet d'amiliorer les performances accroitre la precision taitement des grandeurs non mesurables directement.

1re PARTIE



FICHES TECHNIQUES

FICHE TECHNIQUE MT80Z

****

Le MT-80Z est un Kit fait a base du uP:Z80 est comprenant les organes suivants:

* Une unite centrale UC Z80 avec une hortage de 3.579 MHZ

*Une section de logique de controle (+bus de donnees et d'adresses).

*Section de decodage.

*Section de commande de cycle unique.

*Indication logique :(port de sortie).

*Interrupteurs logiques:(port d'entrees).

*Barrette d'interconnection(port socket). (fig 2).

*Barrette de bus. (fig 1).

*Interface cassette BAR,MIC.

*Buzzer et diodes temoins.

*Memoires.

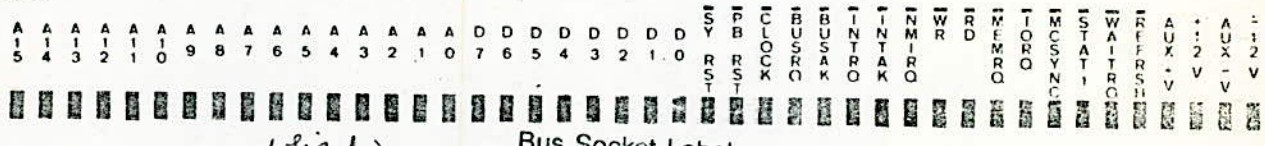
*Afficheur et adresses de donnees.

*Clavier.

*PPI(8255)

*P10.

*CTC.



(fig 1) Bus Socket Label

(fig-2) Port Socket Label

L'UNITE CENTRALE:

Est un microprocesseur Z80 de 40 broches.Se trouve a gauche en haut du KIT pour la description voir fiche technique du CPU.

LOGIQUE DE CONTROLE:

La section de logique de controle se trouve en haut a gouche de la carte .elle fournie les liaisons avec la Z80, les signaux de controle du uP . Le point INT (interruption) etant soude, le Z80 recoit son instruction NMI a partir de la touche BREAK(voir INT,BREAK plus loin)

ALIMENTATION +/-:

Se trouve pres de l'Unite Centrale et de logique de controle Elle a pour but de fournir des alimentations auxiliaires +/-.

BUFFER DE BUS:

Le micro utilise des bus de donnees et d'adresses.Il delivre des signaux de faible tension sur ses broches,le buffer amplifie ces signaux permettent la liaison de nombreux peripheriques autour des broches et assurant la protection du Z80.

CONNECTEUR DE STD:

Le connecteur de Bus STD, J1 se trouve sur le cote gauche du MT80-Z .Ce connecteur a 56 broches peut recevoir l'une des centaines de modules encartables utilisant ce bus.

J-2 BUS ACCESSOIRE:

J-2 se trouve au centre en haut du MT-80Z.Il se presente sous forme de deux alignements de trous.Son but est de permettre par soudure d'un connecteur approprie d'interfacer des accesoires.

SECTION DE DECODAGE:

La section de decodage est constituee de deux circuits situes entre le bas du bus STD et la barrette d'interconnexions, son objet est de generer les impultions de synchronisation necessaires au transfert de donnees en entree/sortie.

COMMANDE DE CYCLE UNIQUE:

La section de commande de cycle unique se trouve en bas a gauche du micro. Elle comprend deux interrupteurs,l'un un commutateur , l'autre un bouton possoir ,qui,utilises conjointement, permettent l'examen detaille du fonctionnent du Z80 et peripheriques accesoires.

INDICATEURS LOGIQUES /PORTS DE SORTIES:

Les indicateurs logiques (voir figure).Il sont constitues de deux rangees de 8 diodes libellees port 1 et 2.Ils permettent :

-Le suivi de valeurs logiques simples.

-Le suivi du bus de donnees lors du fonctionnement du KIT en cycle unique.

-La capture et l'affichage des donnees adressees par le Z80 sur des instruction de sorties.(voir T.P).

INTERRUPTEUR LOGIQUES/PORTS D'ENTREE:

Les interrupteurs logiques a droite des indicateurs logiques sur deux boitiers comptant chacun 8 commutateurs et libelles Port 1 (s-1), port 2 (s-2).(voir T.P)

BARRETTE D'INTERCONNEXIONS:

Elle comporte 33 rangees verticales de chacune 5 points interconnectes electriquement (voir figure) pour plus de detail voir la partie consernant le T.P.

BARRETTE DE BUS: (voir figure-1-)

elle est divisee en 3 sections ,elle est necessaire pour l'interfacage des composants avec le MT-80Z

1-) A15 a Ao :Bus d'adresses.
2-) D7 a Do : # de donnees.
3-) SYSRST a REFRESH: Bus de controle.

INTERFACE CASSETTE -EAR,MIC:

Deux mini jack-phono EAR ,MIC,ils permettent d'interfacer un lecteur enregistreur sur cassette pour conserver ou charger des programme.En utilisant les touches LOAD et DUMP situees sur le clavier, les informations binaires stockees en memoire sont transformees en son qui peuvent etre enregistres ou lus apartir d'un lecteur de cassette.

BUZZER ET DIODE TEMOIN:

Le buzzer et sa diode temoin verte sont interfaces au bus du Z80 par le biais du cicuit d'interface PPI 8255 (voir plus loin)

DIODE TEMOIN D'ARRET DE FONCTIONNEMENT:

La diode temoin d'arret de fonctionnement est la diode rouge.

MEMOIRE:

Les circuits de memoire utilises sur le MT-80Z sont marques U-6 , U-7,et U-8. L'emplacement U-7 est libre d'origine il peut recevoir un circuit destine a etendre la memoire du systeme.Le circuit U-6 est un EPROM(accessible en lecture).Il stocke toutes les informations du systeme du programme du Kit.(voir definition des memoires)

PPI 8255:

(voir definition du PPI).

AFFICHAGE DES ADRESSES ET DONNEES:

L'affichage des adresses et des donnees s'effectue sur 6 digits a 7 segments, les 4 premieres pour les adresses,les deux autres pour les donnees.

CLAVIER:

(voir definition du clavier)

******0******

DESCRIPTION DU CLAVIER :

Le clavier et divise en 3 groupes fonctionnels:

1-) Les touches d'interruptions - RESET ,BREAK ,INTER:

a) RESET:Commande la mise a "0" du Z80, ce qui stoppe le Z80 et lui commande d'aller prendre ses instructions L'adresse 0000. Le MT-80Z est initialise lors de la misesous tension et affiche le mot ready sur l'affi- cheur a 7 segment.
Le programme n'est pas efface,neomoins le pointeur de pile peut se trouver modifie et des donnees peu- vent etre entre sur la partie haute de la RAM.

b) BREAK:La touche BREAK permet une interruption de programme destinee a l'examen des registres et des memoires egalement afin de changer les registres et les memoires pour l'instruction MNI (interruption non masquable) du Z-80 equivalent a CALL 0066.Elle commande au MT-80Z de commancer l'execution d'instructions a l'adresse 0066 du programme moniteur.

c) INTER:Cette touche, connectee au signal broche INT du Z80 permet une interruption suos certaine conditions.Il s'agit d'une interruption masquable qui , lorsqu'elle est validee, entraine l'execution de l'instriction RST 38 selon les etapes suivants
1) le Z80 recherche et execute les instrictions a l'adresse 0038.
2) la routine du moniteur a l'adresse utilise les contenus des adresses stockees en 1FEE et 1FEF.Cette adresse s'appelle un vecteur.
3) Le MT-80 execute l'instriction a partir de l'adresse formee par le contenue de 1FEE et 1FEF.INTER elle peut etre validee soit apartir du clavier,soit a partir d'instriction du prog.

2-) Les touches de fonction:
ADDR,DATA,PREV,NEXT,RELA,INSERT,DELETE,COPY,SET BRK PT,CLR BRK PT,LOAD DUMP, PC,STEP,GO,USER.

1) ADDR: Cette touche selectionne la fonction adrese/memoire.
2) DATA: Elle permet d'introduire des donnees en memoire ou en registre.Elle est enfocee apres ADDR ou egalement apres la touche REG pour modifier les registres.
3) PREV: Elle permet de decrementer l'adresse d'une position memoire.Dans le mode REG la touche PREV decremente sur le registre ou le flag suivant.
4) NEXT: Elle permet d'incrementer l'adresse d'une position memoire Dans le mode REG la touche NEXT incremente sur le registre ou le flag suivant.

b) RELA: Calcule l'adresse relative et stoke le deplacement des ins-instructions JR et DJNZ. Format:appuyer sur RELA,entrer l'adresse de l'instruction JR ou DJNZ puis appuyer sur la touche NEXT,entrer l'adresse de destination du saut,puis appuiyer sur GO.

c) INSERT et DELETE: Permet l'insertion d'un byte en memoire a une adresse affichee +1.Decale les donnees suivante d'une position vers le haut. pour insrer une donnee (octet) a l'adresse (ADDR) il convient en premier lieu d'afficher l'adresse precedente (ADDR-1) en appuyant sur la touche INSERT qui incremente le sur l'adresse (ADDR).Entrer la donnee (octet) a l'adresse (ADDR) en appuyant sur DATA.

- DELETE: Permet d'effacer le contenu d'une instruction en memoire et decale toutes les donnees suivantes vers le bas .

Pour effacer une donnee en memoire (1 octet) a l'adresse ADDR en commence par selectionner l'adresse ADDR ,puis appuyer sur DELETE.Toutes les donnees suivantes sont decales vers le bas

*** REMARQUE: si on cherche a modifier le contenue d'une adresse memoire en dehors de la RAM utilisateur on ne peut apporter aucune modification .La zone accessible par les touches INSERT et DELETE et comprise entre 1800 et 1DFF.

d) COPY: Commande le transfert d'un segment entier en memoire vers une autre zone de memoire.Format:appuyer sur copy Entrer l'adresse de debut de zone memoire a transferer,appuyer sur NEXT,pour entrer l'adresse de fin de zone memoire a transferer,puis a nouveau sur NEXT pour indiquer l'adresse de fin de destination.Enfin apuyer sur GO.

e) SET BRK PT et CLR BRK PT: Elle permettent la mise en place d'un point d'arret pour la premiere ,effacement pour la deuxieme. Le point d'arretest une adresse dans un programmesur lequelle il est possible d'arreter temporairement l'execution de celui-ci.

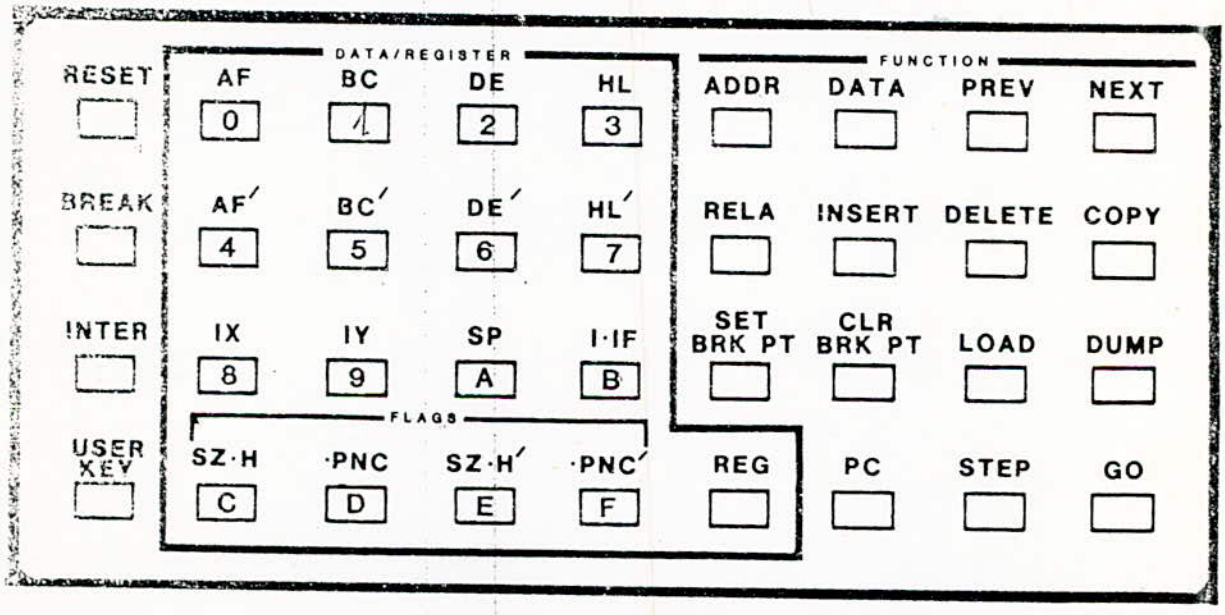
f) LOAD,DUMP: ces touche permettent le stockage ou la reprise de donnees ou de contenusde memoire sur un lecteur de cassette.un bloc quelconquede la memoire peut etre selectionne comme fichier de fonction DUMP le transfert en serie au lecteur de cassette par le biais des jack- phono d'interface.Plusieurs fichier peuvent etre stockes sur une seule cassette lors de l'enregistrement pour stockage en memoire,la fonction LOAD demandera le nom du fichier,son adresse de depart et son adresse finale.*Pour utiliser ces deux fonctions du MT-80Z il est necessaire de relier le lecteur de cassette au MT-80Z par l'intermediaire des jackphono EAR et MICet les cordons appropries.

g) PC: cette touche commande la remise a "0" du compteur de progra- mme de l'utilisateur.Ce compteur de programme est un registre 16 bits contenant l'adresse de la premiere instruction qui doit etre executee. La touche RESET initialise le systeme et branche le compteur a l'adresse la plus petite de la RAM(1800).PC initialise et GO execute le programme.

h) GO: commande l'execution d'un programme appartir de l'adresse afficher.Format:Appuyer sur PC (ou determine l'adresse de depart en la touche ADDR),puis sur la touche GO .

i) STEP:Cette touche est utile lors du deverminaminage d'un pro- gramme car elle permet de suivre pas à pas pour controler son execution. Au cas ou le programme affecterait le pointeur de pile dans les zones 1FAF a 1F9E (pile moniteur),la touche STEP entrane l'apparition de SYS-SP sur l'afficheur.

e) USER: cette touche utilsateur est reliee directement a la broche 38 du PPI 8255 qui se trouve immediatement au dessus de l'afficheur adresse/donnee.Sur le 8255,la brouche 38 est le bit 6 du port A.Le MT-80Z a son port A a l'adresse d'E/S 00. L'utilisation de la touche USEP necessite l'ecriture d'un programme qui entrera le port 00 sur l'accumulateur du Z80 et testera le bit 6.



MT-80Z Keypad

FICHE TECHNIQUE DU CPU Z80

* GENERALITES

Une equipe d'ingenieurs de la societe INTEL;ayant participe au projet du uP INTEL 8080 estimaient que la conception pouvait ameliorer ainsi ils fonderent ZILOG Inc.Dont le premier produit etait ZILOG (Z80) commercialise en 1976 comme il est le successeur du 8080 et du 6800 et possede la totalite des controles de ces deux uP.
Le uP Z80 est synchronise par une horloge monophase situee sur la broche dont la frequence est:3.569MHZ le Z80 est dote d'un certains nombre registres et de possibilites que n'offrent pas le 8080 et 6800 ce qui fait de lui un uP les plus complets du marche.

* CARACTERISTIQUES

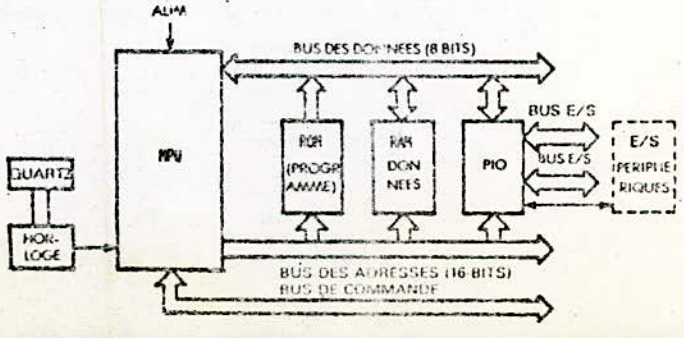
-Technologie:MOS a canal N.
-Tension d'alimentation:5V.
-Duree d'une instruction:1.6mS.
-capacite d'adressge:64 Koctets.
-Longueur des mots traites:8 bits.
-Modes d'adressage :Immediat;relatif;etendu;indexe;implicite;indirect;
a renistre.
-Interrption:3modes masquables et un non masquable.

* CONSTITUTION

L'ensemble des registres internes au microrprocesseur constitue une memoire RAM de 208 bits accessible a l'utulisateur et organisee de la facon suivante:
*Deux jeux de six registres generaux :A,B,C,D,E,H,L et B',C',D',E',H',L' programmables indivudiellement(8bits) ou par paires (16bits)
*Les deux accumulateurs A et A' munis de leur propre indicateur(flags) F et F' (le 6800 possede aussi 2 accumulateurs mais 1 seul indicateur.
*UN REGISTRE de raffraichissement de memoire (R.memory refresh) permettant de connecter directement aveccette unite centrale des memoires dynamique.
Un registre 1(interrpt vector) necessaire a la gestion rapide d'un niveau d'iterruption.Il stoke les 3 bits de poids fort de l'adresse dans le vecteur d'interruption (le dispositif d'interruption fournit les bits de poids faible).

*Deux registres d'index de 16 bits (Indexx Register:IX,IY).
*Un pointeur de pile de 16 bits (Stack Pointer:SP).
*Un compteur de programme de 16 bits (programm counter:PC).

— Un système Z80 standard



BROCHAGE DU Z80

Les 40 broches de l'unite centrale sont representees a la fig (I) leurs definitions sont donnees ci-apres:

A0-A15 : Sorties tri-state (3 etats). Bus d'adresses de 16 bits permet d'adresser 64 Koctets de memoire ou des dispositifs d'entree-sortie.

D0-D7 :Entrees-sorties tri-states actif au niveau logique "1" bus de donnees utiliser pour l'echange de donnees avec les memoires et ou les dispositifs d'entrees-sorties.

$\overline{M1}$: Sortie active a "0". Indique que le CPU est dans un cycle de recherche de l'instruction a executer.

$\overline{MREQ}$: Memory reqest sortie active a "0" validation de l'adresse presente sur le bus d'adresse pour une operation de lecture/ecriture en memoire.

$\overline{IORQ}$: (Input/output request) sortie tri-states active a "0", Les huit bits de poids faible presente sur le bus d'adresse sont valides pour une operation de lecture/ecriture.

$\overline{RD}$: (memory read) sortie tri-states active a "0",l'operation de lecture en memoire ou sur les dispositifs d'entrees-sorties.

$\overline{WR}$: (memory write) Sortie tri-state active a "0", l'ecriture en memoire ou sur les dispositifs d'entrees-sorties validation de donnees sur le bus de donnees.

$\overline{RFSH}$: (refresh) Sortie active a "0". Les 7 bits de poids faible du bus d'adresse contiennent l'adresse des memoires dynamiques a rafrichir.

$\overline{HALT}$: Sortie active a "0", indique que le uP execute l'instruction HALT et attend une instruction masquable ou non avant de poursuivre le traitement de l'instruction durant le HALT le CPU maintient son activite de rafraichissement des memeoire.

$\overline{INT}$: (interrupt request) Entree active a "0" signal genere par les dispositifs d'entrees-sorties et valide a la fin du traitement de l'instruction en cours, si la bascule d'interruption l'autorise et si le signal BUSRQ n'est pas active.

$\overline{NMI}$: (non masquable interrupt) Entree active a "0" la ligne de demande d'interruption non masquable a priorite sur la ligne INT. NMI positionne automatiquement le pointeur de pile a l'adresse 0066H.

$\overline{WAIT}$: Entree active a "0" indique au CPU que les memoires ou les peripherique ne sont pas pret au transfert de donnees.

$\overline{RESET}$:Entree active a "0", elle initialise le CPU comme suit : positionnement de la bascule de validation des interruptions remise a zero du compteur de programme et des registres I et R durant le temps de RESET les bus d'adresse et des donnees sont a l'etat haute impedance.

$\overline{BUSRQ}$:(bus request) entree active a "0" signal de controle de bus demande au CPU de placer le bus d'adresse, le bus de donnees et les signaux de controle a l'etat haute impedance.

$\overline{BUSAK}$:(bus acknowledge) Sortie active a "0", c'est unsignal d'acquitement du BUSRQ.

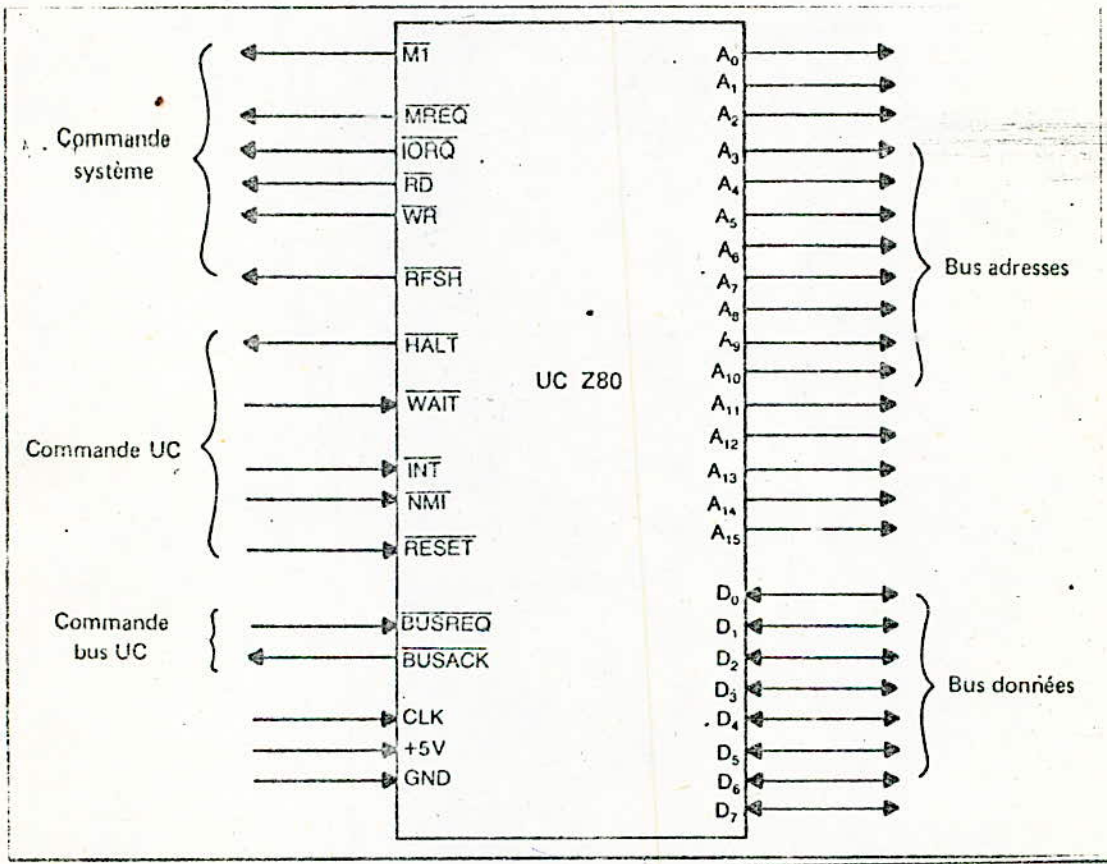
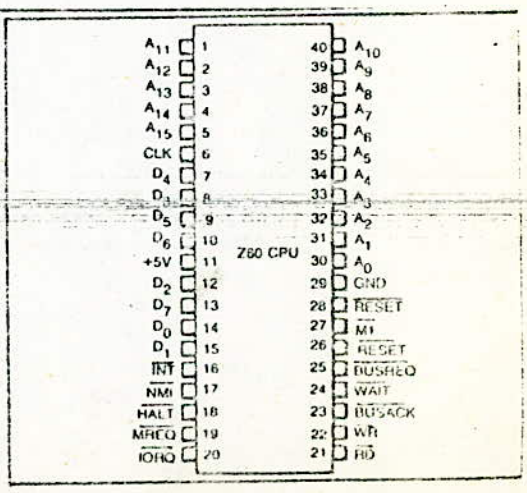


Schéma de brochage fonctionnel du microprocesseur Zilog Z80.



(Fig I)

ORGANISATION INTERNE DU Z80

Le uP (Z80):travaille de la maniere suivante:

Il lit dans la memoire de programme la premiere instruction, la decode,va eventuellement chercher en memoire les donnees necessaires a l'execution de l'instruction et traite cette instruction , est ainsi de suite jusqu'a la derniere instruction du programme.
Le uP communique avec les circuits de memoire et d'interface d'entree/ sortie generalement par trois lingnes ou bus :
-Le bus d'adresses est unidirectionnel,Il permet au uP de selectionner une position de memoire ou un registre a l'interieur d'un circuit d'interface d'entree/sortie.
-Le bus de donnees est bidirectionnel et assure l'echange d'information entre le uP d'une part et les circuits de memoire et d'interface d'entree/ sortie d'autre part.
Le bus de controle est un ensemble heterogene de signaux, qui permette au uP et aux circuits peripheriques de fonctionner en harmonie .Ces signaux comprennent la remise au point initial des horloges,les demandes d'interruption,etc...

Le uP comprend:
-L'unite de controle (CU) se charge de la decodification des instructions et engendre les signaux de commande necessaires a la bonne execution de l'instruction.
-L'unite arithmetique et logique (ULA):execute des operations arithmetiques et logiques.
-Un certain nombre de registre pour executer une instruction. Certains sont accessibles a la l'utilisateur , d'autres ne le sont pas.

Les registres accessible a l'utilisateur comprennent:
Les accumuateur,le compteur ordinale PC, le pointeur de pile le SP, registre de condition parfois des registres d'index et des registres generaux destines a des stockages temporaires.

DEFINITION DES REGISTRES DU Z80

*L'ensemble des registres internes du uP Z80 sont utilises pour des stockages provisoires d'information donc ilsconstitue une memoire RAM de 208 bits.Le Z80 contient 22 registres qui ont chacun des utilisations particulieres.Ils sont partagees en deux categories:Les registres d'usage general et Les registres specialises.

LES REGISTRES D'USAGE GENERAL

*Ce sont les registres A,B,C,D,E,H,L,A',B',C',D',E',H',L,'ont peut les considerer comme de simples cases memoires directement accessibles par le uP,ils peuvent etre associes deux a deux pour former des registres de 16 bits.Il est a noterqu'on ne peut pas former n'importe quelle paire de registres,enoutre on peut les associer comme suit:(BC),(DE),(HL) ou(B'C'), (D'E'),(H'L');d'ou il faut consederer que l'on possede deux jeux interchangables de registres.

UTILISATION DES REGISTRES

*Le registre A est appele ACCUMULATEUR,dans lequel les calculs sont effectues.
*Le registre B est souvant utilise avec l'instruction DJNZ dans les programme comme compteur de boucle.
*Les paires de registre jouent un role d'accumulateurs pour les operations arithmetiques sur 16 bits.
*La paire HL forme un registre pointeur primaire pour la uP.
*Les registres B,C et D sont appeles registres tompons,ils sont utilises pour le stockage temporaire de donnees.

LES REGISTRES SPECIALISES

*Les registres d'index(IX,IY):
*L'indexation est une technique d'adressage de la memoire qui n'est pas utilisee par tous les uP,elle permet d'acceder facilement aux elements successifs d'un bloc de donnees en memoire.

*Le Compteur Ordinal(PC):(Program counter)
*Il est present dans tous les uP,cette presence est fondamentale pour l'execution des programmes car le PC contient l'adresse de la prochaine instruction a executer,ainsi il est necessaire de la transferer de la memoire vers le uP sur le bus de donnees.

*Le Pointeur de pile(SP):(Stack Pointer)
*Avant d'executer un sous-programme,le uP stocke dans une partie de la memoire de donnees appelee PILE.L'adresse du programme (Adresse de retour) vers laquelle le uP devra se rendre apres avoir traite le sous-programme. Lors d'une demande d'interruption,le uP reconnait l'interruption et accepte la dite demande;apres avoir termine l'instruction en cours.Il sauvgarde automatiquement dans la pile l'adresse de retour apres traitement de l'interruption et parfois les contenu de ses registres internes (Accumulateur,Registres d'index,Registres generaux,Registre de conditions) et se branche au sous programme d'interruption.

Une fois ce sous-programme execute,le uP,recharge eventuellement,a partir de la pile , ses registres avec le contenu qu'ils avait au paravant(Avant l'interruption).Et se branche a l'adresse de retour du programme principal.
*La pile est organisee de facon telle que l'information entree la premiere sort la derniere (LIFO:last in first out).Quand un mot (Nombre de bits sur lequels le uP peut travailler directement)est stocke dans la pile il est inscrit a l'adresse,qui est contenue dans le pointeur de pile.Le pointeur est decremente automatiquement d'une unite apres chaque inscription d'un mot ou d'une information.Quand un mot est lu dans la pile le pointeur de pile est incremente automatiquement d'une unite.

*Le registre de condition F:
*Ce registre de 8 bits,contient 8 bascules qui se positionnent selon la derniere instruction a un ou a zero;chaque bits de ce registre a une signification bien particuliere ce qui sera explicite ci-apres a l'exception de deux:les bits 3 et 5 qui sont laisses par le constructeur sans indications.
BIT0 Carry C:C'est l'indicateur de retenu il est positione par les operations arithmetiques de decalage et de comparaison;les operations logiques (XOR,OR,AND) mettent C a zero,et l'instruction SCF le met a un(1).
BIT1 N:Ce bit indique si la derniere instruction est une soustraction dans ce cas il est positione a un (1),Ce bit est utilise avec l'indicateur H et l'operation d'ajustement decimal.
BIT2 P/V:Ce bit a un double sens;dans le cas d'operations logiques ou de decalages il sert d'indicateurs de parite.Et dans le cas des oprations arithmetique,d'increment ou de decrement il indique le depacement de capacite (Over flow).
BIT4 Half carry H:C'est l'indicateur de retenue au rang 4 c-a-d le 5eme bit,il fonctionnent comme C mais pour le premier quartet.
BIT5 Z:Ce bit indique que la derniere operation effectue est nulle,alors Z=1;de meme pour les situations suivantes:les operations logiques et arithmetique si le resultat est nul,Les operations de comparaison si les valeurs sont egales aussi.Ce bit est affecte par les instructions d'entree -sortie en bloc lorsque le decrement de B donne un resultat nul;(INDR,INI, IND,OUTI,OUTD...)
BIT7 S:Ce flag donne le signe de l'octet contenu dans l'accumulateur c'est a dire qu'il prend la valeur du bit7 de l'octet contenu dans l'accumulateur(A).

Le registre d'interruption I:
*Ce registre de 8bits contient le poids fort de l'adresse d'une memoire dont l'octet de poids faible est genere par l'organe ayant provoque l'interruption,cette case memoire et la juxtaposante contiennent l'adresse du sous-programme d'interruption,ainsi avec ce registre on peut avoir une table contenant 128 vecteur d'interruption.

Le registre de rafraichissement R:
*Il sert a stocker ler adresses de la memoire dynamique a rafraichir qui doivent etre rafraichies,il est evident qu'il lui est associe toute la logique necessaire au rafraichissement

*Ce sont les registres A,B,C,D,E,H,L,A',B',C',D',E',H',L,'ont peut les considerer comme de simples cases memoires directement accessibles par le uP,ils peuvent etre associes deux a deux pour former des registres de 16 bits.Il est a noterqu'on ne peut pas former n'importe quelle paire de registres,enoutre on peut les associer comme suit:(BC),(DE),(HL) ou(B'C'),(D'E'),(H'L');d'ou il faut consederer que l'on possede deux jeux interchangables de registres.

UTILISATION DES REGISTRES

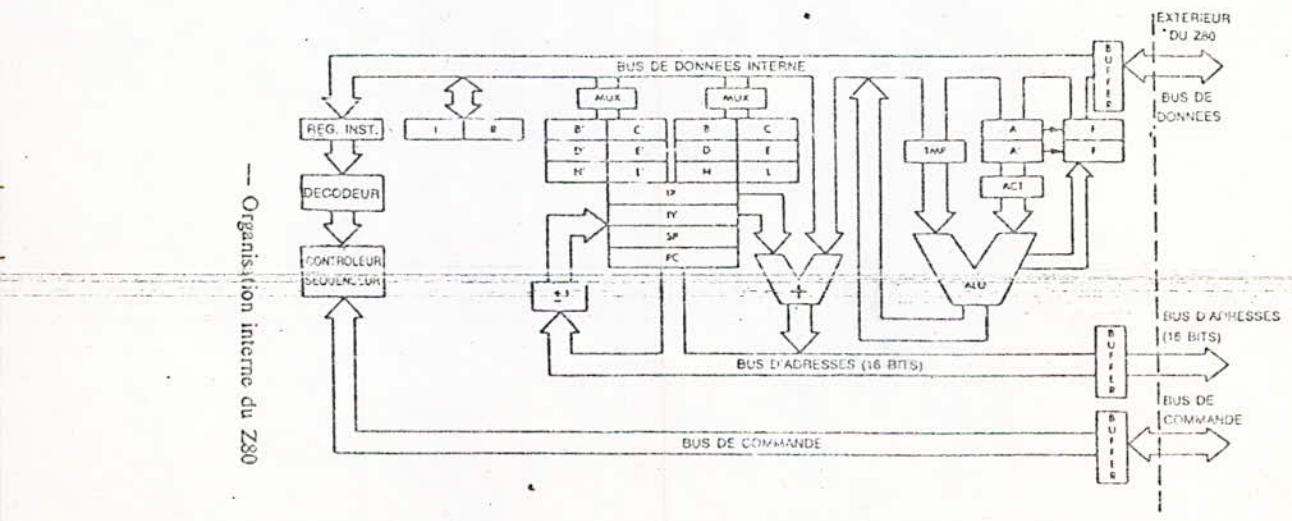
*Le registre A est appele ACCUMULATEUR,dans lequel les calculs sont effectues.
*Le registre B est souvant utilise avec l'instruction DJNZ dans les programme comme compteur de boucle.
*Les paires de registre jouent un role d'accumulateurs pour les operations arithmetiques sur 16 bits.
*La paire HL forme un registre pointeur primaire pour la uP.
*Les registres B,C et D sont appeles registres tompons,ils sont utilises pour le stockage temporaire de donnees.

LES REGISTRES SPECIALISES

*Les registres d'index(IX,IY):
*L'indexation est une technique d'adressage de la memoire qui n'est pas utilisee par tous les uP,elle permet d'acceder facilement aux elements successifs d'un bloc de donnees en memoire.

*Le Compteur Ordinal(PC):(Program counter)
*Il est present dans tous les uP,cette presence est fondamentale pour l'execution des programmes car le PC contient l'adresse de la prochaine instruction a executer,ainsi il est necessaire de la transferer de la memoire vers le uP sur le bus de donnees.

*Le Pointeur de pile(SP):(Stack Pointer)
*Avant d'executer un sous-programme,le uP stocke dans une partie de la memoire de donnees appelee PILE.L'adresse du programme (Adresse de retour) vers laquelle le uP devra se rendre apres avoir traite le sous-programme. Lors d'une demande d'interruption,le uP reconnait l'interruption et accepte la dite demande;apres avoir termine l'instruction en cours.Il sauvgarde automatiquement dans la pile l'adresse de retour apres traitement de l'interruption et parfois les contenu de ses registres internes (Accumulateur,Registres d'index,Registres generaux,Registre de conditions) et se branche au sous programme d'interruption.



— Organisation interne du Z80

DEFINITION DE LA PILE *

Une pile est une succession de cases memoires contigues dont l'acces est genere par le registre SP.Cet acces d'un type special peut ce comparer des assiettes dans une armoire Chaque donnee a ecrire dans la pile serait l'equivalent d'une nouvelle assiette placee au sommet de la pile .Chaque ecriture par le uP dans la pile ne ferai qu'augmente .le tas de donnees empilees en les conservant .Par contre chaque lecture de la pile equivau a retirer l'assiette superieure qui est la seule accessible sans risque de casse .On concoit ainsi que la premiere donnee lue dans la pile sera celle qui a ete ecrite en derniere est a chaque lecture en desempile pour acceder au donnees plus profondes.Ce mode d'accee s'apalle pile L.I.F.O "last In First Out" _dernier entre premier sortie.

La getion de cette pile par le registre SP est tres simple. Au depart lorsque la pile est vide SP contient l'adresse d'une case memoire .A chaque ecriture de la pile le registre SP est decremente de 1 puis la donnee est ecrite dans la case memoire dont l'adresse est cotenue dans le registre SP .Les donnees de la pile sont aussi rangees de facon contigues par adresses deroissantes.A chaque lecture dans la pile il se produit la demarche inverse.C'est a dire qu'on lit la case memoire pointee par SP puis on incremente de 1 le registre SP.

*Cette definition a ete tire du livre intetule "ASSEMBLEUR FACILE DU Z80"

LES MEMOIRES

LES GRANDES CATEGORIES DE MEMOIRE

*Pour realiser des memoires plusieurs techniques ont ete mises a profit dans le passe,aujourd'hui il ne reste plus que trois grandes categories:

LES MEMOIRES A SEMI-CONDUCTEUR: Nees vers 1970,avec le progre de l'integration des semi-conducteurs,c'est la categorie qui se developpe le plus.les boitiers les plus denses en 1985 memorisaient 256 kbites c-a-d 32 koctets,elles ont un temps d'acces a l'information tres rapide (quelques dizaines de nanosecondes).

LES MEMOIRES MAGNETIQUES: Elles remontent aux annees 1960,il s'agit des tombours,bandes et disques,elles stockent les elements binaires sous la forme d'un domaine aimante,et c'est par deplacement mecanique du support devant des tetes de lectures ou ecritures que l'on a acces a l'information.les capacites sont elevees,mais le temps d'acces est tres long,on les appelle aussi memoires de masse.

LES MEMOIRES A BULLES MAGNETIQUES: Assez recentes(fin des annees 1970), elles utilisent le deplacement d'un domaine magnetique sous l'influence d'un champ magnetique dans une structure metallique deposee sur support, ells fonctionnent comme un tres grand registre a decalage dont la sortie est rebouclee sur l'entree donc il n'ya aucun mouvement mecanique. actuellement les boitiers proposes sont aux megabits(1 octet=8 bits), elles se developpent rapidement et devraient concurencer les memoires magnetiques (pas de deplacement mecanique) beaucoups plus que les memoires a semi-conducteur(temps d'acces tres long)

TECHNOLOGIES DES MEMOIRES A SEMI-CONDUCTEUR

*On s'interessa plus particulierement a ce type de memoires car c'est le plus utilise,il y a quelques grandes technologies communes a la majorite des meneoires,les plus utilisees a titres d'informations:
*Technologie MOS (Metal Oxide Semi-Conductor).
*Technologie CMS ("MOS Complementaire).
*Technologie Bipolaire.

CLASSEMENT DES MEMOIRES SELON LEURS UTILISATIONS

*La memoire ideale est de grande capacite,consommant peu d'energie,de vitesse elevee,gardant son information en cas de coupure d'alimentation, cette memoire n'existe pas,celles qui existent sont classees comme suit:

RAM (Random Access Memory)
*Sur laquelle on peut ecrire ou lire une information,RAM sinifie:memoire a acces aliatoire c-a-d qu'on peut passer d'une case memoire a n'importe quelle autre case memoire sans contrainte contrairement aux memoires a acces serie ou apres avoir lu ou ecrit une case memoire la prochaine operation case devrait concerner immediatement la voisine.Le contenu d'une memoire vive s'efface si la tension d'alimentation disparait ainsi on les apppelle parfois :Memoire Volantaire.

ROM (Read only memory)
*On ne peut que lire cette memoire,l'information est incrite par le constructeur au moment de la fabrication:elle contient des programmes systeme et des programme de service utilitaire.La RAM est une memoire a acces aleatoire et consarve son contenu apres une coupure d'alimentation.

PROM (Progcammable Read Memory)
*Lorsqu'une information varie dans le temps la ROM ne convient pas;alors on utilise des memoires PROM,programmable par l'utilisateur avec un despositif appele Programmateur PROM.La PROM conservent son cotenu apres coupure d'alimentation et on ne peut ecrire dessus qu'une seul fois.

REPROM OU EPROM
*On les appelles ROM effacables,EPROM(electrically programmable read only memory)constituent une solution,du fait qu'elles sont effacees apres exposition a une source d'ultraviolets a travers une fenetre de quartz, l'effacement d'une vingtaine de minutes .l'integrite de l'information contenue dans ces memoires est conservee apres une coupure d'alimentation.

EPROM(ELECIRICALLY ERASABLE PROGRAMMABLE READ ONLY MEMORY)
*En plus des aventages qu'ont les ROM et les PROM. Les EPROM peuvent etre ecrites et effacees electriquement,cela permet d'effacer l'information inadequate sur la carte meme ou se situe le boitier sans l'extraire comme c'est le cas avec les EPROM.

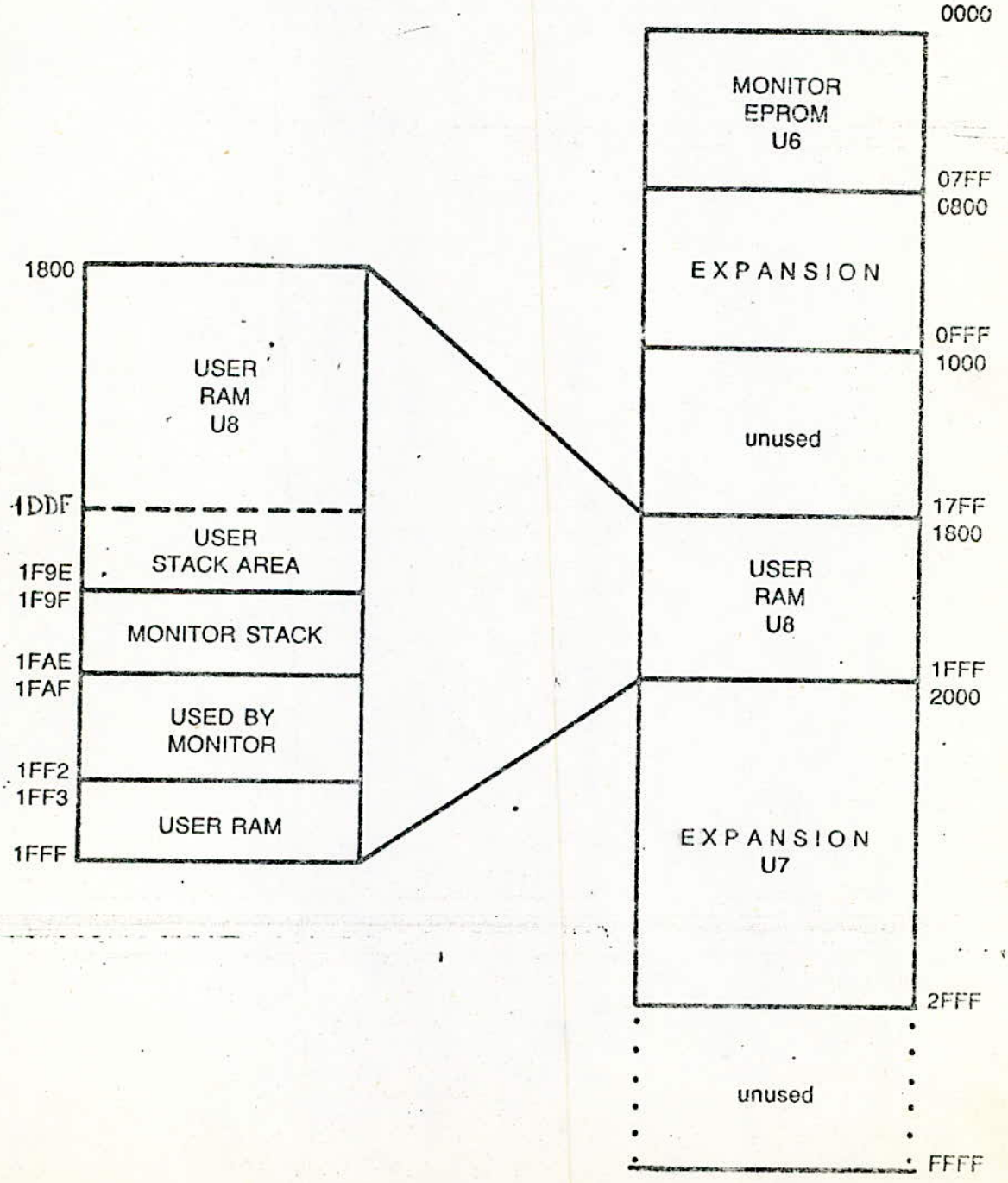
MEMOIRE FIFO (FIRST IN,FIRST OUT)
*Une memoire FIFO fonctionnent comme une file devant un guichet: Le premier arrive est le premier a etre servi l'ordre chronologique d'entree est respectee en sorte.

MEMOIRE LIFO (LAST IN,FIRST OUT)
*Une memoire LIFO fonctionne comme une pile d'assiettes: la derniere assiette posee sur le dessus de la pile est la premiere a en etre retiree.

INTEGRITE DE L'INFORMATION
*L'augmentation incessante de la capacite des memoires va de paire avec une reduction de la taille des elements constituant,ainssi la capacite electrique de stockage diminue et par conseqent la quantite d'electricite stockee,ce qui rend la memoire plus sensible aux conditions exterieures et aux effets de proximite;en outre la densite accrue se paie par une fragilite plus grande.

La figure ci dessous, represente les emplacements memoires du MT-80Z, Les programmes etablits et stockes en memoire sont entres entre les adresses 1800 et 1FFF.

* La zone accessible par les touches INSERT et DELETE est comprise entre 1800 et 1DDF.
* Le moniteur ajuste automatiquemenet le pointeur sur 1F9F..
* Chaque fois que l'on utilise les instructions PUSH,POP,CALL,RET, ainsi que les points d'interruption la memoire se trouve affectee'.



MODE D'ADESSAGE DU Z80

Le Z80 utilise comme modes d'adressages l':

-ADRESSAGE IMPLICITE.
-ADRESSAGE IMMEDIAT.
-ADRESSAGE ABSOLU (ETENDU).
-ADRESSAGE PAGE ZERO MODIFIE.
-ADRESSAGE RELATIF.
-ADRESSAGE INDEXE.
-ADRESSAGE INDIRECT.

**ADESSAGE* IMPLICITE:

Une instruction est dite implicite lorsq'elle ne contient pas explicitement l'adresse de loperande,sur le quel elle travaille .L'adressage implicite est essentiellement utilise par les instructions sur un seul octet travaillant sur les registres internes. Chaque fois que des instructions implicites travaillant exclusivement de cette facon leur execution ne dure pas plus d'un cycle:
Exemple:LD r,r'; ADD A,r; ADC A,s; INC r;CP s.

**ADRESSAGE IMMEDIAT:

Dans ce mode, le code operation sur huits bits est suivi d'un litteral (constante)sur 8 ou 16 bits. Ce type d'instructiones utile pour mettre une valeur donnee sur 8 bits dans un registre, de meme si on dispose d'instruction sur 16 bits il sera nessaire d'avoir recours a des litteraux eux memes sur 16 bits

Exemple : LD r,n (2 octets).
LD dd,nn (3 octets).
ADD A,n (2 octets).

**ADESSAGE* ABSOLU: <<OU ETENDU>> Z80.

Par definition,l'adressage absolu requert trois octets premier est celui du code operation,et les deux autres une adresse sur 16 bits (l'adresse absolue). Par opposition avec l'adressage court (adressage sur 8 bits) ce mode est aussi appele <<adressage etendue>>.

Exemples d'instructions utilisant l'adressage etendu:
°°°°°°°°° LD HL,(nn) et JP nn*
nn* represente une adresse memoire sur 16 bits.
(nn) represente son contenue.

**ADRESSAGE APAGE ZERO:

L'adressage de la page zero n'est pas disponible sur le Z80,sauf au travers des instructions RST (RESET).Le mode special d'adressage utilise par ces instructions est appele <<adressage page zero> L'instruction RST contiennent un champ de trois bits (b5 b4 b3) servant a designer l'une de huit adresses memoires en page zero.L'adresse chargee dans le PC est b5 b4 b3 000.
Ces instructions sont executees rapidement car elle ne demandent qu'un seul octet et sont facilement executes par le HARDWERE.Elle sont generalement utilisees pour repondre a des sources d'interruption multiples (jusqu'a 8).

Cette instruction est utilisee moins souvent, depuis l'apparution des controleurs de priorite d'interrption (PIC).

(* ADRESSAGE.)

**ADRESSAGE RELATIF:

Par definition,l'adressage relatif demande deux octets Le premier est le code operation de <<Saut relatif>>,le second represente le deplacement et son signe il est note JR pour le differencier du saut absolu.

Lors d'execution de l'instruction s'il n'y a pas de branchement,elle ne requert que sept cycles d'horloge ,parce que l'adresse de la prochaine instruction executable se trouve deja dans le compteur prdinal. Or l'orsque le branchement doit avoir lieu elle demande 12 cycles d'orloge une nouvel adresse doit,en effet etre calculee et charger dans le compteur ordinal.

Le probleme de dure n'existe pas pour le saut inconditionnel JR e,qui ne test aucune condition,et dure toujours 12 cycles d'horloge.

**ADRESSAGE INDEXE:

Le Z80 dispose d'instruction dont le code operation tient sur 16 bits. Ce mode d'adressage est tres utile lorsqu'il s'agit d'adresser l'une apres l'autre de nombreuses positions memoires situees a des adresses adjacentes.L'adresse de la premiere position-memoires sera chargee dans un registre specifique appele INDEX pour passer a une adresse a la suivante d'incrementer d'une unite le de l'index .

Exemple: LD,ADD,INC,RLC,BIT,SET.

**ADRESSAGE INDIRECT:

Le Z80 fournit un adressage indirect limite,appele adressage indirect par registre .Dans ce mode ,chaque pair [BC,DE,HL], peut etre utilisee en tant qu'adresse memoire .Lorsqu'elles pointent sur les donnees de 16 bits c'est toujours vers leur partie basse,la partie haute se trouve en memoire a l'adresse immediatement superieur.

INTRODUCTION

*Les Travaux Pratiques que nous proposons se divisent comme suit:

-SERIE N 1 :SUR LE CLAVIER.
-SERIE N 2 :SUR LES JEUX D'INSTRUCTIONS DU Z80.
-SERIE N 3 :SUR LES DIVERS OPERATIONS ARITHMETIQUES.
-SERIE N 4 :SUR LES ENTREES/SORTIES.

*REMARQUE: Nous laissons le libre choix a l'enseignant de dispatcher ces series de T.P selon le volume horaire .

La serie N 1 A pour but d'initier a l'utilisation des touches du clavier de la carte MTZ-80. Avant d'aborder cette serie il faut ce referer a la partie concernant la definition des touches.

La serie N 2 A pour but de manipuler les Instructions utilisees par le Z80 pour cela on la subdivise en 4 grandes parties:

-1er partie contient des instructions groupees de la facon suivante:
*Chargement 8 bits.
* # # 16 bits.
* # # Immediat.
*Echange.

-2eme partie contient des instructions groupees de la facon suivante:
*Arithmetiques 8 bits.
*Ajustement decimale.
*Logique 8 bits.
*Arithmetiques 16 bits.

-3eme partie contient des instructions groupees de la facon suivante:
*de Saut.
*Sous-programmes.
*de la manipulation de Pile.
*Sur les bits.
*de Decalage.

-4eme partie contient des instructions groupees de la facon suivante:
*Entree/Sortie.
*De Chaines.
*D'Usage general.
*Les Interruptions.
*Controle.

DEROULEMENT DES TP DE LA SERIE N_2:

La serie N 2 contient tout le jeu d'instruction du Z80 . Il est divises en 4 parties , chaque partie contient un ensemble de groupes d'instructions, de chaque groupe on a choisi une seule instruction exposee selon les etapes suivantes :

1-* L'instruction, avec definition.
2-* Un commentaire.
3-* Le chemin des donnees: indiquant par schemas comment s'effectue le deroulement des operations.
4-* La duree:de l'instruction en cycles Machines.
5-* Le mode d'adressage.
6-* L'indicateur (des flags) s'il indique quelque chose ou non.
7-* Un exemple detaille.

*Puis pour le reste des instructions d'un meme groupe, elles ont ete classes dans des tableau.

Manipulation: Nous suggerons le deroulement des manipulations de la facon suivante :

Faire la meme chose avec une partie du reste des instructions classees dans les tableaux comme dans l'etape N 7, et le rendre sur place. (le jour meme).
Tandis que pour les etapes 1,2,3,4,5 et 6 se feront sous forme de compte rendu a rendre en seance suivante .

**** REMARQUE: On a tout ce qu'il faut pour rediger,la duree, mode d'adressage et indicateur de flags. Pour le code objet il faut se referer a l'ANNEXE.
Pour le reste des groupes, les instructions sont classees de la meme facon et dans la meme disposition. Il y'a des tableaux qui contiennent une grande liste d'instructions qui vont prendre peut etre beaucoup de temps, nous laissons le choix a l'enseignant de subdiviser ces parties selon les seances et les volumes horaires qui leurs sont affectees. A la condition que l'etudiant effectue un nombre minimum d'instructions correspondant a deux ou trois instructions de chaque groupes ; ces instructions sont choisies soit par l'enseignant ou par l'etudiant.

La serie N 3 a pour but de traiter divers operations arithmetiques, ainsi que la manipulation des zones memoires, et traitement de tables de valeurs,etc...

DEROULEMENT DES TP DE LA SERIE N 3:

Pour les operations arithmetiques, on a propose des programmes de (+,-,x,:) , suivi de questions que l'etudiant peut y repondre facilement L'enseignant peut varier les nombres utilises dans les operations , par chaque binome d'etudiants (ou d'un etudiant).
Pour la partie concernant la Mise a zero d'une zone memoire, ou Trouver le plus grand element d'une table, Somme de N elements, Compter des zero, Comparaison de deux nombres signes sur 16 bits. Elles sont suivis de bonne questions faciles.

*** L'enseignant doit definir:
*Une table d'elements pour que les etudiants puissent trouver le plus grand element de cette table le programme correspondant.
*Une table d'elements qui contient au maximum 256 elements,qui doivent etre des entiers positifs,par la suite l'etudiant peut faire la somme.
*Une table qui contient des elements et des zero , et avec le programme correspondant l'etudiant peut compter les zero existant.
*Les deux nombres (signes sur 16 bits) par le programme de comparaison l'etudiant peut les compare .

***REMARQUE: Pour les tables et les nombres nous suggerons de les varier pour chaque binome d'etudiants (ou un etudiant).

****

La serie N 4 Cette serie est tres intressante de point de vue commande elle nous permet de savoir commment comminique le Z80 avec l'exterieur et comment s'effectuent les operations en les visualisants sur les leds,elle nous permet aussi de savoir comment utiliser les interrupteurs sans passer par le clavier.

**** REMARQUE: Pour cette partie on peut se referer a la deuxieme partie de notre travail a condition que le PIO et le CTC existent et d'executer les programmes mentionnes dans la partie programmation du PIO et programmation du CTC, dans cette remarque, on constate qu'on s'approche d'une partie tres interressante qui pourait nous aider dans la commande par uP elle peut aide ceux qui font la suite de notre trvail.

****

**** REMARQUE : Sur la serie n 2
Les notations utiliser dans les exemples pour definir les indicateur de falgs sont :
? : Indicateur modifie aleatoirement par l'operation
X ou * : Modifie selon le resultat de l'operation.
1 : Indicateur mis a un
0 : Indicateur mis a zero

r: registre.
d: deplacement.

SERIES

DES

SERIE -1-

I.P :N° SUR LE CLAVIER

1ere partie:

A) voici un programme sous-routine TONE 2K. Ce programme entraine un signal audio de frequence 2KHz sur le buzzer:

Programme:	ADRESSE	CODE	MNEMONIQUE
	1800	CD	CALL
	1801	E2	LO ADDR
	1802	05	HI ADDR
	1803	76	HALT

*Mettre le Kit sous tension.
*1-Pour charger et verifier le programme :
Utiliser les touches:ADDR,DATA,NEXT,PREV,PC,STEP,GO,RESET
Executer-le.
*2-Modifier le programme pour changer la duree du son emis.En modifiant HL en utilisant la touche REG par:

HL	duree(~)
0400	0,5 sec
0800	1 #
1000	2 #
2000	4 #
0000	15 #

B) INSERT DELETE: (voir definition)
****

Programme:		
	1800	11
	1801	22
	1802	33
	1803	44
	1804	55
	1805	66
	1806	77
	1807	88
	1808	99
	1809	AA

QUESTIONS:
1°-Inserere EE a l'adresse 1805.
2°-Effacer EE.

C) COPY:
***** Pour le programme precedant: 1800 11 etc...
# # #
-Appuyer sur COPY -->(-S) .Entrer l'adresse de depart.
NEXT -->(-E) .Demande l'adresse final
NEXT -->(-d) .Entrer l'adresse de destination.
GO le programme est transferer vers l'adresse voulu
-Faire l'inverse

D) RELA:

```
Programme:   1800    3E       LD A,00
             1801    00
             1802    3C       INC A
             1803    D3       OUT(FF),A
             1804    FF
             1805    18       JR dis   . Branchement relatif
             1806    ??                  a l'adresse 1802
             ++++++++++
```

Question: - Calculer l'etiqette dis de destination.

E) STEP:

```
Programme:   1800    00
             1801    97
             1802    D3
             1803    FF
             1804    0E
             1805    FF
             1806    10
             1807    FE
             1808    3C
             1809    18
             180A    F7
```

QUESTIONS:- Initialiser
Utiliser STEP et constater le saut.

Suite page 29

F) BRK PT:

Programme:	Adresse	Ins code	Mnemonique	Commentaire
	1800	3E	LD A,00	A=0
	1801	00		
	1802	47	LD B,A	B=A
	1803	2F	CPL	A=A
	1804	D3	OUT (FF),A	Sortie sur FF
	1805	FF		
	1806	3D	DEC A	A=A-1
	1807	04	INC B	B=B+1
	1808	18	JR dis	Branchement a l'adr 1804
	1809	FA		

Questions: 1-) Executer le programme
2-) Faire des points d'arret.
3-) Effacer-les

G) BREAK:

Programme:	Adresse	Ins code	Mnemonique
	1800	CD	CALL TONE 2K
	1801	E2	
	1802	05	
	1803	76	HALT

Questions: 1-) Executer le programme
2-) Utiliser la touche BREAK.
(constater qu'elle interrompe le programme.

H) INTER:

Programme:	Adresse	Ins code	Mnemonique
	1800	00	NOP
	1801	97	SUB A
	1802	D3	OUT (FF),A
	1803	FF	
	1804	0E	LD C,FF
	1805	FF	
	1806	10	DJNZ dis
	1807	FE	
	1808	3C	INC A
	1809	18	JR dis
	180A	F7	

Questions: 1-) Relier les connexions OUTFF et P1CL de la barrette d'interconnexion par un fil de 7.2 cm.
2-) Executer le programme
3-) appuyer sur INTER que constatez-vous?

**REMARQUE:
(appuyer sur BREAK et RES et entrer 1 1F l'afficheur indique 0000 1F le vecteur INTERRUPT est à 0 .Pour valider INTER appuyer sur DATA et 01 l'afficheur -------->000.1.1F.) .Refaite Q:2-) et 3-).

PARTIE <1>

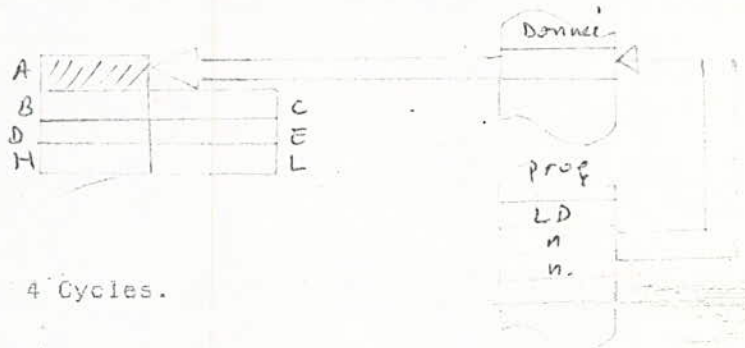
Serie -2-
Partie <1>

CHARGEMENT (8 bits) REGISTRES

LD A,(nn): Chardement de l'accumulateur apartir de l'emplacement memoire (nn).

COMMENTAIRE: Le contenue de l'emplacement memoire adresse par le contenue des deux emplacements memoire suivantes immediatement le code operatoire est charge dans l'accumilateur.l'octet de poids faible de l'adresse est situe juste derriere le code operatoire.

CHEMIN DES DONNEES:



DUREE : 4 Cycles.

MODE D'ADRESSAGE: Direct

INDICATEUR: Aucun effet

EXEMPLE:

Programme:	Code objet	Avant			Apres		
LD A,(1900)	3A	A	K*	K*	A	B	A
	00						
	19	1900	B	A	1900	B	A
	76						

K*:Valeur H q/q

CHARGEMENT REGISTRE

INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS
LD A,(nn)	Direct	4 cycles	Aucun effet
LD A,(dd)	Indirect/Reg	2 cycles	# # # #
LD r,(HL)	Indirect/Reg	# #	# # # #
LD A,(IX+d)	Indexe	# #	# # # #
LD r,(IX+d)	Indexe	5 cycles	# # # #
LD A,(IY+d)	Indexe	5 cycles	# # # #

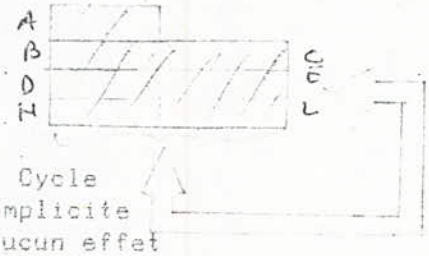
TRANSFERT : 8 bits

LD r,r': Chargement du registre r a partir du registre du registre r.

r <—— r'

COMMENTAIRE : Le contenue du registre source spesifier est charge dans le registre de destinstion specifie,r et r' peuvent etre n'import lequel parmis B,C,D,E,H,L

CHEMIN DES DONNEES:



DUREE: 1 Cycle
MODE D'ADRESSAGE: Implicite
INDICATEUR: Aucun effet

EXEMPLE:

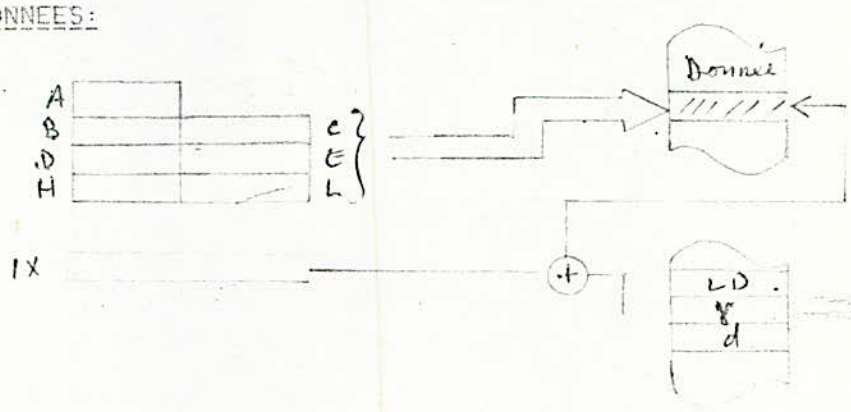
Programme		Code Objet	Avant	Apres
LD H,A	1800	67	A B E	B E
HALT	1801	76	H F 0	B E

STOCKAGE REGISTRE

LD (IX+d),r Chargement de l'emplacement memoire d'adresse indexe (IX+d) a partir du registre r (IX+d)<----- r

COMENTAIRE: Le contenu du registre r est charge dans l'emplacement memoire adresse par le contenu du registre d'index IX plus le deplacement fourni "d",le registre r peut etre n'importe lequel de A,B,C,D,E,H,L.

CHEMIN DES DONNEES:



DUREE: 5 Cycles machines

MODE D'ADRESSAGE: Indexe

INDICATEURS: Aucun effet

EXEMPLE:

Programme:	Code objet		Avant	Apres
LD(IY+D),A	FD,77,03	(A)	3 E	3 E
HALT	76	(IY)	19 20	19 00
		(1900)	2 1	2 1
		(1923)	5 A	3 E

INSTRUCTION	MODE D'ADRESSAGE	DUREE(Cycle)	INDICATEURS
LD (nn),A	Direct	4	Sans effet
LD (dd),a	Indirect/registre	2	# #
LD (HL),r	# # # #	2	# #
LD (IX+d),r	Indexe	5	# #

REM: Le registre IY a le meme role que IX.

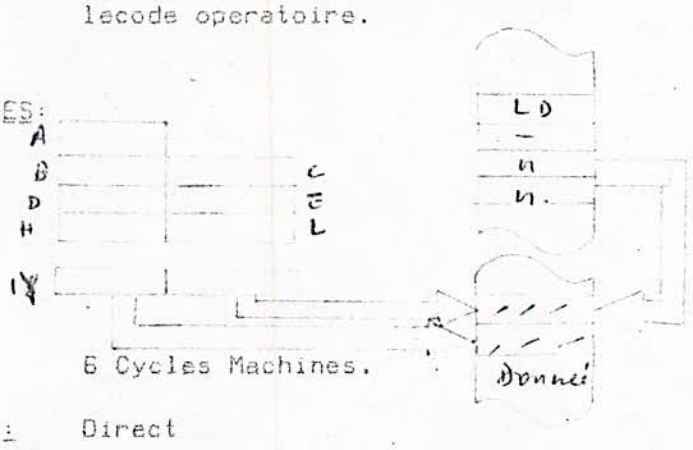
CHANGEMENT 16 BITS

*Le Z80 possede des registres de 16 bits,il y a la possibilite de coupler des registres de 8 bits pour ne former qu'un seul registre de 16 bits, (BC,DE,HL).La memoire etant organisee en mots de 8 bits un tel chargement 16 bits necessite deux octets de memoire,un mot memoire de 16 bits range en memoire contient l'octet de poids faible a l'adresse du mot (nn) et l'octet de poids fort a l'adresse suivant (nn+1).

LD (nn),IY — Chargement de l'emplacemment memoire d'adresse (nn) a partir de IY.
(nn)<---IY(faible);(nn+1)<---IY.

COMMENTAIRE: Le contenu du partie basse du registre IY est charge dans l'emplacement memoire adresse par le Contenu des emplacements memoires suivants immediatement le code operatoire,Le contenu de la partie haute du registre IY est charge dans l'emplacement memoire suivant immediatement celui charge a partir de la partie de poids faible de l'adresse est situe juste derriere lecode operatoire.

CHEMIN DES DONNEES:



DUREE: 6 Cycles Machines.

MODE D'ADRESSAGE: Direct

INDICATEURS: Aucun effet

EXEMPLE:

Programme	Code Objet		Avant	Apres
LD (1945),IY	FD,22,45,19	(IY)	18 43	18 43
HALT	76			
		(1945)	AB	43
		(1946)	CD	18

*L'ensemble des Instructions suivantes ont une structure identique a celles des chargements de 8 bits,d'ou en distingue:

Chargement registre (16 bits): Memoire ----->Registre.
Stockage # # # # # # : Registre ----->Memoire.
Transfert # # # # # # : Registre ----->Registre.

	INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS
chargement	LD dd,(nn)	Etendu	5	Aucun effet
	LD IX,(nn)	## ##	6	# # # #
	LD SP,(nn)	## ##	6	# # # #
Stockage	LD (nn),dd	Etendu	HL:5	Aucun effet
	LD (nn),IX	# #	BC,DE	# # # #
	LD (nn),SP	# #	et IX :6	# # # #
Transfert	LD SP,HL	Par registre	1	Aucun effet
	LD SP,IX	# # #	2	# # # #
	LD SP,IX	# # #	2	# # # #

REMARQUE:
*Seul le registre SP peut etre mis en distination.

INSTRUCTION DE CHARGEMENT IMMEDIAT

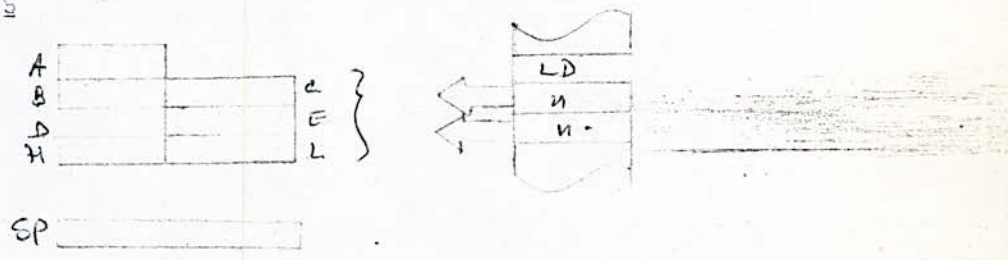
**On peut appeler cette instruction :chargement d'une constante

LD dd,nn — Chargement du registre double dd avec la donnee immediat nn.

dd <----- nn

COMMENTAIRE — Le contenu des deux emplacements memoires suivants immediatement le code operatoire est charge dans le registre double specifie l'octet de poids faible de la donnee est situe juste derriere le code operatoire dd peut etre n'importe quel paire de ces registre. (BC);(HL);(DE);(SP)

CHEMIN DES DONNEES



DUREE: 3 Cycles machines

MODE D'ADRESSAGE: Immdiat

INDICATEURS: Aucun effet

EXEMPLE:

Programme:	Code objet	Avant	Apres
LD BC,(1900)	01,00,19	(B) AF CD (C)	19 00
HALT	76		

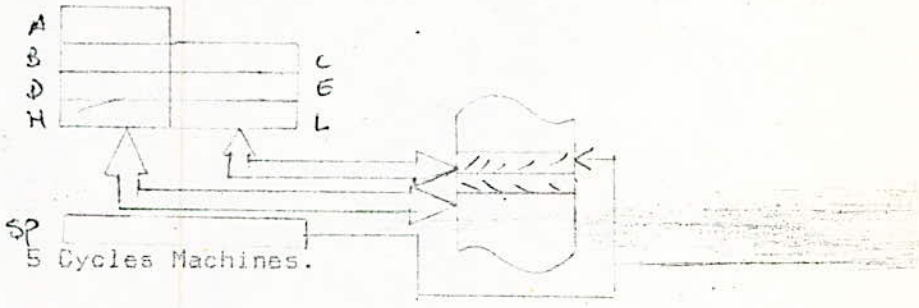
	Instruction	Mode d'adressage	Duree	Indicateurs
8 BITS	LD A,n	Immediat	2 cycles	AUCUN EFFETS
	LD r,n	# #	2 cycles	# # # #
	LD (HL),n	Indirect/registre	3 # #	# # # #
	LD (IX+d),n	Indexe	5 # #	# # # #
8 BITS	LD dd,nn	immediat	3 cycles	AUCUN EFFETS
	LD sp,nn	# #	3 # #	# # # #
	LD IX,nn	# #	4 # #	# # # #

INSTRUCTIONS D'ECHANGE

EX (SP),HL — Echange de HL avec le sommet de la pile.
SP<--->L , (SP+1) <--->L .

COMMENTAIRE: Le contenu du registre L est echange avec le contenu de l'emplacement memoire adresse par SP,Contenu de l'emplacement memoire suivant immediatement celui adresse par le pointeur de pile.

CHEMIN DES DONNEES:



DUREE: 5 Cycles Machines.

MODE D'ADRESSAGE: Indirect

INDICATEURS: Aucun effet

EXEMPLE:

Programme	Code Objet		Avant	Apres
EX (SP),HL	E3	(H)	19 00	6B 3A
HALT	76			
		(SP)	19 50	19 50
		(1950)	3A	00
		(1951)	63	19

INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEUR
EX DE,HL	Implicite	1	Aucun effet
EX AF,AF'	## ##	1	F echange avec F'
EX (SP),HL	Indirect/registre	5	Aucun effet
EX (SP),IX	## ## ## ##	6	## ## #

REMARQUE:
*Il existe une autre Instruction d'echange entre les deux jeux de registre d'usage general est :EXX qui echange:
BC avec B'C' ;DE avec D'E' ;HL avec H'L'.

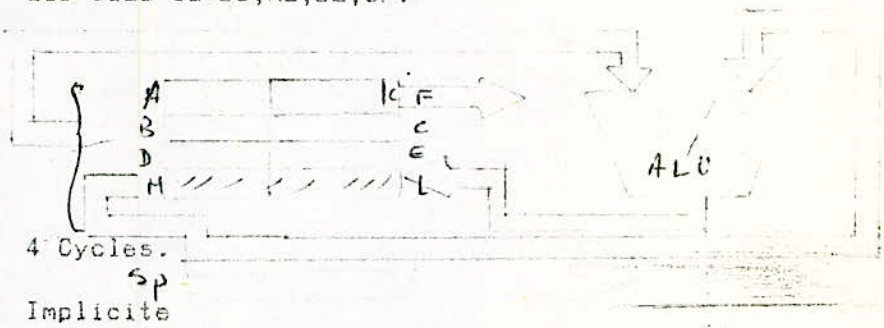
PARTIE (2)

INSTRUCTION ARITHMETIQUE 16 BITS

SBC HL,ss — Soustraction de HL le registre double ss et le report
HL <--- HL-ss-C

COMMENTAIRE: Le contenue du registre double specifie augmente du contenue du registre double HL et le resultat est range a nouveau dans HL,ss peut etre n'importe lequel des ceau la:BC,HL,DE,SP.

CHEMIN DES DONNEES:



DUREE: 4 Cycles.

MODE D'ADRESSAGE: Implicite

INDICATEURS: * * ? * 1 *

EXEMPLE: SBC HL,DE

Programme:

			Avant:			Apres:	
SBC HL,DE	ED		66	F		06	F
	52	D	19 00	E	D	19 00	E
HALT	76	H	19 6A	L	H	00 6A	L

	INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS
ADD	ADD A,r	par registre	1 Cyc	S,Z,H,V,C,N=0
8 bits	ADD A,n	Immediat	2 #	# #
A =A+s	ADD A,(HL)	Indirect/Reg	2 #	# #
	ADD A,(IX+d)	Indexe	5 #	# #
ADD	ADC A,d	par registre	1 #	# #
8 bits	ADC A,n	Immediat	2 #	# #
A=A+s+Cy	ADC A,(HL)	Indirect/Reg	2 #	# #
	ADC A,(IX+d)	Indexe	5 #	# #

*** REMARQUE: Le bit CY est additionne ceci permet de propager la retenue lors d'additions successives de plusieur octets.

Soustrac-	SUB A,r	par registre	1 #	# #
tion	SUB A,n	Immediat	2 #	# #
8 bits	SUB A,HL	Indirect/Reg	2 #	# #
A=A+s	SUB A,(IX+d)	Indexe/Reg	5 #	# #

Soustrac-	SBC A,r	par registre	1	#	#	#
tion	SBC A,n	Immediat	2	#	#	#
8 bits	SBC A,(HL)	Indirect /Reg	2	#	#	#
A=A-s-Cy	SBC A,(IY+d)	Indexe/Reg	5	#	#	#
Increment	INC r	par registre	1	#	#	#
	INC (HL)	indirect/Reg	3	#	#	# (C) n'est pas affecté
S=S+1	INC (IX+d)	Index	6	#	#	#

*** REMARQUE: Cette serie d'instruction n'utilise pas l'accumulateur toute fois l'instruction "INC" peut etre consideree comme le condense de 3 instructios utilisant l'acculaleur

LD A,HL ; *la seule differance c'est que l'instruction
ADD A,1 ; INC ne modifie pas le contenue de l'accumula-
LD (HL),A ; teur mis a part INCA

DEC 8 bits	DEC r	par registre	1	#	S,Z,H,V,N=1	
S=S-1.	DEC (HL)	Indirect/Reg	3	#	#	
	DEC (IX+d)	Indexe/IX	6	#	#	
	CP r	par registre	1	#	#	
Comparai-	CP n	Immediat	2	#	#	
son	CP (HL)	Indirect/Reg	2	#	#	
A-S	CP (IY+d)	Indexe/IY	5	#	#	
A=-A	NEG	Implicite	2	#	#	

INSTRUCTION D'AJUSTEMENT DECIMAL

*Cette instruction permet de faire de l'arithmetique decimale avec le Z80. L'octet est separe en deux mots de quatre (4)bits,chaque mot code l'un des chiffres de 0 a 9 ce qui permet de compter de 0 a 9 avec un seul octet.Cette representation n'est pas compatible avec l'arithmetique binaire du Z80.L'instruction d'ajustement decimal (DAA) permet de retourner a la convertion BCD a partir du binaire.

Exemple: addition de 8 et 6: ###### 8: 0000 1000 = 0 8
6: 0000 0110 = 0 6

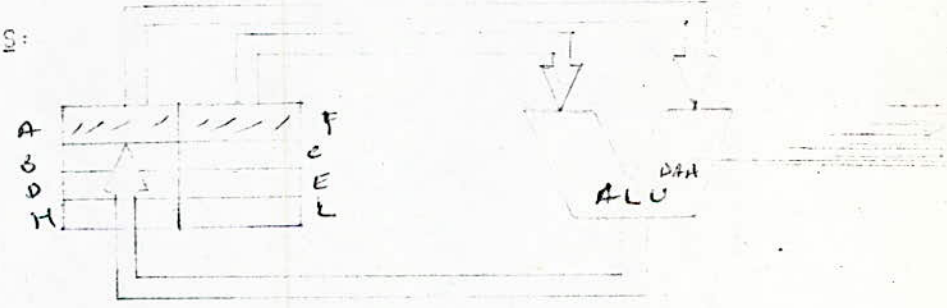
14: 0000 1110 = 0 E H
[non cmpatible]
le resultat est 0E H pour l'instruction DAA si en [avec BCD]
effectue le resultat: 0001 0100
1 4

Lorsque les bits 0 a 3 ont une valeur > 9 ou que le bit H est ,DDA additionne 6 a l'accumulateur.Ensuite si les bits 4 et 7 ont une valeur >9 ou que le bit C est,ces 4 bits sont incrementes de 6.

DAA . Ajustement Decimal de l'accumulateur.

COMMENTAIRE: Selon le registre d'indicateurs:cette instruction ajoute conditionnellement <<6>> au quartet de poids fort et au faible de l'accumulateur,pour la conversion DCB apres les operations arithmetiques.

CHEMIN DES DONNEES:



DUREE: 1 Cycle Machine.

MODE D'ADRESSAGE: Implicite

INDICATEURS:

?	?		?			?	!	?
S	Z		H			PV	N	C

EXEMPLE:

Programme	Code Objet		Avant	Apres
DAA	27	(A)	B2 94	18 05
HALT	76			

REM: L'instruction DAA onctionne uniquement sur l'accumulateur. (Adressage Imlplicite).

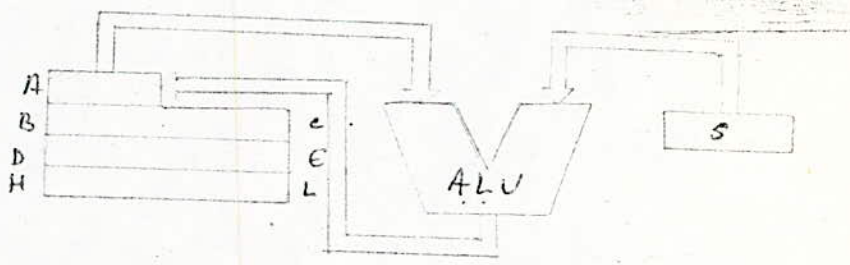
INSIRUCTIONS LOGIQUES (8BITS)

**L'ensemble de ses instructions s'apparente aux instructions arithmetique les operations effectuees font partie d'une arithmetique speciale qui n'existe qu'avec le binaire (Algebre de boule).Dans ce cas l'indicateurs P/v indique la parite du resultat.Toutes ces instructions utilisent l'accumulateur et s'ecrivent: OPR A,SOURCE

XOR s — Ou exlusif logique bentre l'accumulateur et l'operande. A<---AVS

COMENTAIRE: Le OU exlusif logique de l'accumulateur et de l'operande s specifie est effectue le resultat est range dans l'accumulateur.

CHEMIN DES DONNEES:



MODE D'ADRESSAGE & DUREE

L'opecande	Ducee en cycles	Moded'adressage
r	1	Implicite
n	2	Immediat
(HL)	2	Indirect
(IX+d)	5	Indexe

INDICATEURS:

X	X	!	0	!	X	0	0

EXEMPLE: XOR ,B1

Programme:	Avant	Apres
EE		
B1	(A) 36	87
76		

	INSTRUCTION	MODE D'ADRES	DUREE	INDICATEURS
	AND A,r	Implicite/registre	3	
AND	AND A,n	Immediat	3	S,Z,H=1
A=AΛA	AND A,(HL)	Indirect/registre	3	C,P,N=0
	AND A,(IX+d)	Indexe	3	
OR	OR A,r	Implicite/registre		
A=AVs	OR A,N	Immediat		S,Z,H=0
	OR A,(HL)	Indirect/registre		C,P,N=0
	OR A,(IX+d)	Indexe		
XOR	XOR A,r	Implicite/registre		
A=A+s	XOR A,n	Immediat		S,Z,H=0
	XOR A,(HL)	Indirect/registre		C,P,N=0
	XOR A,(IX+d)	Indexe		
NO	CPL	Implicite	1	H=1,N=1
A=Ā				

REM: Cette instruction ne se fait que sur l'accumulateur.

INSTRUCTION ARITHMETIQUE 16 bits

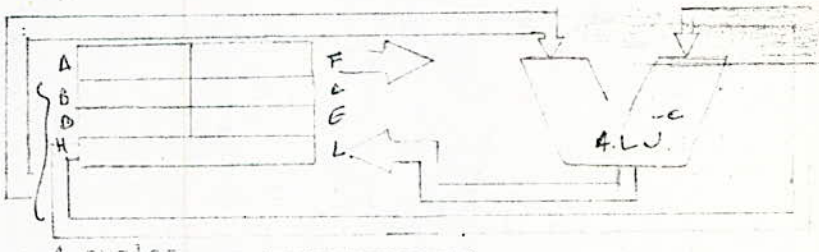
Certaine instructions arithmetique du Z80 sont realisable en utilisant les registres 16 bits ou les paires de registres 8 bits,les operandes ont donc une capacite de 16 bits Cet ensemble d'instructions resemble donc aux instructions arithmetiques 8 bits .l'indicateur P/V indique un debordement .

l'accumilateur ne possedent que 8 bits il n'est pas utilise par ces instruction ,c'est a dire que l'arithmetique 16 bits definit dans les operandes a la fois la source et a la destination.

OPR. Source, destination

SBC HL,ss — le contenue du registre double specifie augment du contenue de l'indicateur de report est soustrait du contenue du registre double HL et le resultat est range a nouveau dans HL .ss peut etre n'importe lequel de :BC,HL,DE,SP.

CHEMIN DES DONNEES :



DUREE: 4 cycles.

MODE D'ADRESSAGE: Implicite

INDICATEUR: * * ? * 1 *

S Z H P/V N C

H est positionne s'il y a report du bits 12

C est positionne s'il y a report.

EXEMPLE: SBC HL,DE

Programme :		Avant :			Apres:		
1800	ED	0110	0110	F	0000	0110	F
1801	52						
1802	76	D 06	89	E	D 06	89	E
		H 31	42	L	D 2A	89	L

***Add 16 bits :PP=PP+ss : dd=BC,DE,HL,et SP.

	INSTRUCTION	MODE D'ADRESSAGE	DUREE	FLAGS AFFECTES
De	ADD HL,dd	IMPLICITE /REG	3 Cycles	N=0,C
meme	ADD IX,BC	# # # #	4 # #	#
pour	ADD IX,IX	# # # #	# # #	#
IY	ADD IX,SP	# # # #	# # #	#

***Add 16 bits avec retenue :HL=HL+ss+CY :

INSTRUCTION	MODE D'ADRESSAGE	DUREE	FLAGS AFFECTES
ADC HL,dd	par registre	4 Cycles	S,Z,V,N=0,C
ADC HL	# # #	# #	# # #
ADC IX	# # #	# #	# # #

***Soustraction 16 bits avec retenue :HL=HL-ss-CY:

INSTRUCTION	MODE D'ADRESSAGE	DUREE	FLAGS AFFECTES
SBC HL,dd	par registre (imp)	4 Cycles	S,Z,V,N=0,C.

***Incrementation 16 bits : ss=ss+1:

INSTRUCTION	MODE D'ADRESSAGE	DUREE	FLAGS AFFECTES
INC dd	Implicite	1 Cycle	aucun
INC IX	# # #	2 #	#
INC IY	# # #	2 #	#

***Decrement 16 bits ss=ss+1:

INSTRUCTION	MODE D'ADRESSAGE	DUREE	FLAGS AFFECTES
DEC dd	Implicite	1 Cycle	aucun
DEC IX	# # #	2 Cycles	#

Soustraction 16 bits:

```
LD HL,(VAR1)    Initialisation
LD DE,(VAR1)    des registres
OR A,A          remise a du flag
SBC HL,DE       Soustraction:VAR1-VAR2
LD (RESULT) .   Rangement du resultat.
```

**Les operations 16 bits ne 'effectuent q'en registres .Il est interessant de notrer que l'instruction OR A,A n'effecte pas l'accumulateur et mais le bit C a zero.

PARTIE (3)

INSTRUCTIONS DE SAUT

***Ces instructions sont equivalentes au GOTO et IF..THEN du BASIC,elles forment les squelettes du proprogramme et utilisent trois mode de branchement: Relatif ,Indirect ,Page zero.

***Le saut peut etre soit conditionnel (Equivalent de IF..THEN..GOTO),ou incoditionnel(GOTO).Les sauts conditionnels utilisent la position des flagspour decider si le branchement doit etre effectue ou non.Il y'a huit condition possibles reperees par un mnemonique:

NZ : Z=0 Resultat non nul

Z : Z=1 Resultat nul

NC : C=0 Pas de retenue

C : C=1 Retenue presente

PO : P/V=0 Parite impaire

PE : P/V=1 Parite paire

P : S=0 Signe positif

M : S=1 Signe negatif

***La syntaxe d'un branchement conditionnel est la suivante:
SAUT Condition ,Etiquette

*SAUT: Soit un branchement direct soit un branchement relatif:

(Direct) Instruction 3 Octet (Branchement sur toute la zone memoire)

(Relatif)Instruction 2 Octet(# # # # 256 adresses(-128,127))

*CONDITION: Ce sont l'un des mnemonique cites en haute.

*ETIQUETTE: Sont l'une des etiquettes de branchement du programme situe dans la zone etiquette ou define par une directive.

JP cc,pq Saut conditionnel a l'adresse pq,si cc est vraie.
PC<--pq

COMENTAIRE: Si la condition specifiee est vraie,les deux octets d'adresse qui suivent immediatement le code operatoire sont charges dans le compteur ordinal,le premier octet suivant le code operatoire etant charge

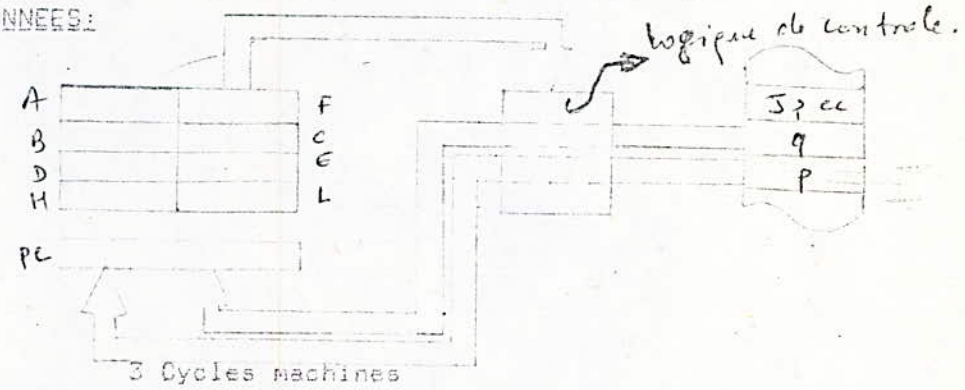
INSTRUCTIONS DE SAUT

JP cc,pq — Saut conditionnel a l'adresse pq,si cc est vraie.
PC<---pq

COMENTAIRE: Si la condition specifiee est vraie,les deux octets d'adresse qui suivent immediatement le code operatoire sont charges dans le compteur ordinal,le premier octet suivant le code operatoire etant charge dans les poids faibles de PC si la condition n'est pas verifiee,l'adresse est ignoree cc peut etre n'importe lequel de:

NZ: non nul	PO: parite impaire
Z: nul	PE: parite paire
NC: pas de report	P: plus
C: report	M: moins

CHEMIN DES DONNEES:



DUREE: 3 Cycles machines

MODE D'ADRESSAGE: Immediat

INDICATEURS: Aucun effet

EXEMPLE:

Programme	Code Objet		Avant	Apres
JP C,19AE	DA,24,3B	F	51	51
HALT	76			
		PC	00 31	19 AE

INSTRUCTION	CONDITION	DUREE	INSTRUCTION	CONDITION	DUREE
(Cas des branchements directs)			(Cas des branchements relatifs)		
JP pq	Incond	3	JR pq	Incond	3
JP NZ,pq	Z=0	#	JR C,pq	C=1	#
JP Z,pq	Z=1	#	JR NC,pq	C=0	#
JP NC,pq	C=0	#	JR Z,pq	Z=1	#
JP C,pq	C=1	#	JR NZ,pq	Z=0	#
JP PO,pq	P/V=0	#			
JP PE,pq	P/V=1	#			
JP P,pq	S=0	#			
JP M,pq	S=1	#			
JP (HL)	Incond	1			
JP (IX)	# #	2			
JP (IL)	# #	3			

**REM : C'est le cas des branchements indirects ou l'adresse est dans un un registre de 16 bits.

BRANCHEMENT ET DECREMENTATION

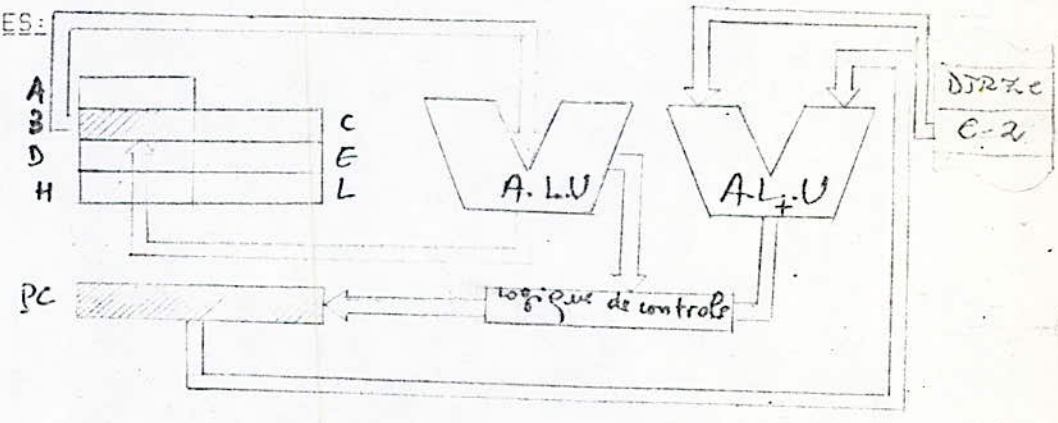
Le Z80 possede une insruction speciale pour faire des boucles.Cette instruction est l'association d'un saut et d'une incrementation de B.
On peut le resumer par:Decrement de B.Si B=0 continuer.Si non se brancher a l'etiquette specifiee.
Le branchement est relatif et effectue sur une plage de 255 adresses.
L'instruction qui effectuee ceci est: DJNZ pq
Dans ce cas d'instruction on utilise B comme compteur de boucle.

**********.

DJNZ pq — Decrementation de B et saut relatif de e si B non nul
B<---B-1 ;Si Best nul PC<---PC+e .

COMENTAIRE: Le registre B est decrementa,si le resultat n'est pas zero,le deplacement immediat est ajoute au compteur ordinal,en utilisant l'arithmetique en complement a 2 de facon a permettre a la fois des sauts en amont et en aval.Le deplacement est ajoute a la valeur de PC+2(apres le saut).Ainsi le deplacement reel va de 129 a -126 Octets.L'assembleur soustait automatiquement 2 du deplacement source pour generer le code hexadecimal.

CHEMIN DES DONNEES:



DUREE: Si b=0 :2 Cycles machines ;Si non 3 Cycles .

MODE D'ADRESSAGE: Immediat

INDICATEURS: Aucun effet

EXEMPLE:

Programme	Code Objet		Avant	Apres
DJNZ PC,-5	10,~~2E~~ F9	(B)	21	20
HALT	76			
		PC	18 E1	18 DC

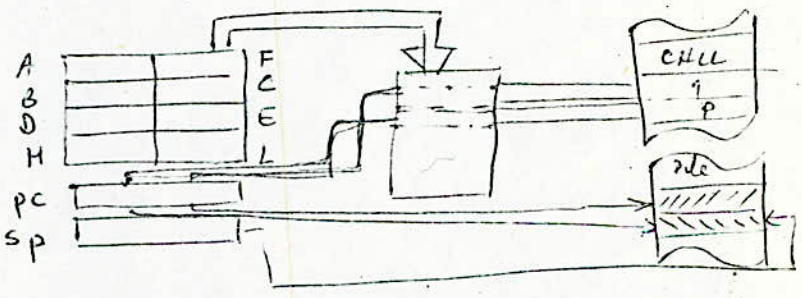
APPELS DE SOUS-PROGRAMME

*Ces instructions fonctionne comme le GOSUB du BASIC :
Appel : CALL GOSUB et Retour: RET RETURN
*L'appel de sous-programme peut etre conditionnel,(IF..THEN..GOSUB).
*Les conditions sont les memes que pour les instructions de saut,la santaxe de ces instructions sont: CALL Condition,Etiquette Appel
RET Condition Retour
*L'etiquette est le label attribue au sous-programme.

CALL cc,pq — Appel conditionnel de sous-programme
Si cc est vraie:(SP-1)<---PC (poids fort)
(SP-2)<---PC (poids faible)
Si cc est fausse :PC<---(PC+3)

COMMENTAIRE: Si la condition est remplie;le contenu du compteur ordinal est empile comme c'est decrit pour les instructions PUSH.Ensuite,le contenu de l'emplacement memoire suivant est charge dans le poids faible et le contenue du second emplacement memoire,apres le code operatoireest charge dans la moitie de poids fort du PC.L'instruction suivante est cherchee a cette nouelle adresse.Si la condition n'est pas remplise l'adresse pq est ignoree est l'instruction suivante est executee,cc peut erte n'importe laquelle de: NZ,Z,NC ,CPO,PE,P,M.L'instruction RET est utilisee a la fin du sous-prog appele pour restaurer le PC .

CHEMIN DES DONNEES:



DUREE: Si cc est vraie,5 Cycles;sinon 3 Cycles Machines.
MODE D'ADRESSAGE: Immediat
INDICATEURS: Aucun effet

EXEMPLE:

Programme	Code Objet		Avant	Apres
CALL Z,1900	CC,00,19	(F)	85	85
HALT	76			
		(PC)	18 AA	18 AD
		(SP)	19 54	19 54
		(1952)	8F	8F
		(1953)	04	04
		(1954)	32	32

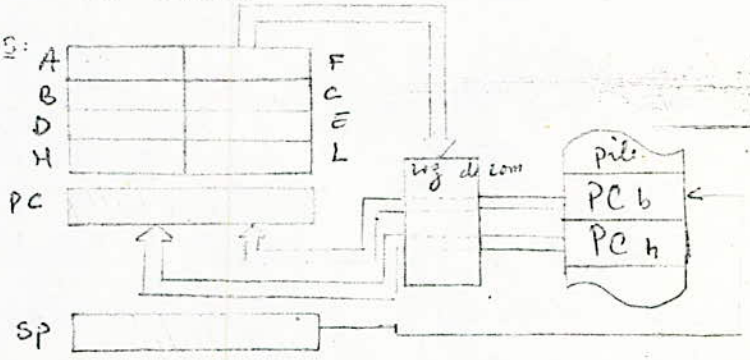
***(F):85 ie:que la condition est verifie c-a-d Z=1.

RETOUR DE SOUS-PROGRAMME

RET cc — Retour conditionnel de sous-programme
Si cc est vraie:PC(---(SP+1) (poids fort)
(PC)(---(SP) (poids faible);SP=SP+2

COMMENTAIRE: Si la condition est remplie,le compteur ordinal est depile comme c'est decrit pour l'instruction POP. Ensuite,L'instruction suivante est cherchee a l'adressa contenue dans le PC.Si la condition n'est pas remplie,l'execution des instructions continue en sequence.
cc peut etre n'importe qu'elle parmis les conditions deja citees.

CHEMIN DES DONNEES:



DUREE: Si cc est vraie,3 Cycles;sinon 1 Cycles Machines.

MODE D'ADRESSAGE: Indirect

INDICATEURS: Aucun effet

EXEMPLE:

Programme	Code Objet		Avant	Apres
RET NC	D0	(F)	00	00
HALT	76			
		(PC)	01 24	10 00
		(SP)	10 00	10 02
		(1000)	00	00
		(1001)	10	10

APPEL DE SOUS-PROGRAMME EN PAGE ZERO

***Le Z80 possede une instruction speciale appelee RESTART,qui permet d'effectuer un branchement sur un sous-programme residant a une adresse speciale en page zero,l'interet de RST est de n'utiliser qu'un seul octet. contrairement a CALL qui en necessite 3 octets.

***L'adresse de branchement en page zero sont en nombre de huit.

RST p	Cette instruction produit la meme chose que CALL p.
COMMENTAIRE:	L'instruction RST p est tres utilisees par le systeme d'exploitation donc elle est tres delicate a manier.
DUREE:	3 Cycles Machines.
MODE D'ADRESSAGE:	Indirect.
INDICATEURS:	Sans effet.
REMARQUE:	L'operateur peut etre l'une des adresses suivantes: 00H,08H,10H,18H,20H,28H,30H,38H.

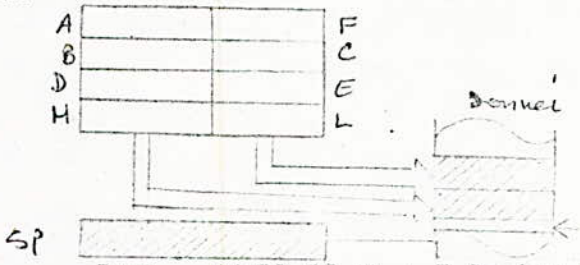
INSTRUCTION DE MANIPULATION DE LA PILES

*Le Z80 possede un ensemble d'instructions qui permetent de placer et de retirer ces registres de la pile,cette sauvgarde sur la pile se fait toujopurs par paire de registres ou par registre 16 bits .
*L'action d'empiler un registre s'appel PUSH,l'action de desempiler s'appela POP.

PUSH dd — Empilement du registre qq
(SP-1)<---dd(poids fort);(SP-2)<---dd(poids faible)
(SP)<---(SP-2)

COMMENTAIRE: Le pointeur de pile est decremente et le contenu de poids fort du registre double specifie est change dans l'emplacement memoire adressee par le pointeur de pile .Le pointeur de pile est de nouveau decremente et le contenu de poids faible du registre double est charge dans l'emplacement memoire adresse alors par le pointeur de pile,dd peut etre:BC,HL,DE.

CHEMIN DES DONNEES:



DUREE: Pour dd=BC,DE,HL : 3 Cycles Machines.
Pour dd=IX,IY : 4 " " " " .

MODE D'ADRESSAGE: Indirect

INDICATEURS: Aucun effet

EXEMPLE:

Programme	Code Objet		Avant	Apres
PUSH BC	C5	(BC)	01 AB	01 AB
HALT	76			
		(SP)	10 00	10 FE
		(10FE)	00	AB
		(10FF)	10	01
		(1000)	34	34

L'instruction PUSH possede les meme caracteristiques que PUSH,sauf pour le chemin des donnees.

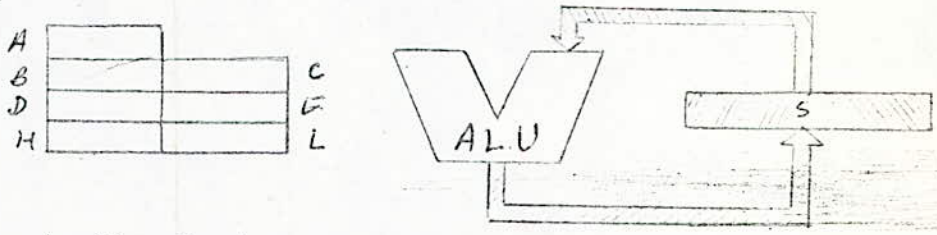
INSTRUCTION SUR LES BITS SET/RESET

*L'instructions SET-RESET permet de manupilation.Un bits sur un registre ou une case memoire.
*Les operations effectuer la mise de bits a 1(SET);et leur mise a 0(RESET)
*La syntaxe :OPR (numero de bit),source.

RES b,s — Effacement du bit b de l'operande s.
(S)<---0

COMMENTAIRE: Le bit specifie de l'amplacement determine par s est mis a zero.s est defini dans la description des instructions BIT similaire.

CHEMIN DES DONNEES:



INDICATEURS: Le flag Z est uniquement affecte.

DUREE:

S	r	(HL)	(IX+d)
Cycles	2	4	6

MODE D'ADRESSAGE:

	Implicite	Indirect	Indexe

EXEMPLE:

Programme	Code Objet	Avant	Apres
RES 1,H	CB,BC	(H) 0100 0010	0100 0000
HALT	76	(H) 4 2	4 0

INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS
BIT b,r	Implicite/registre	2	Z,H=1,N=0
BIT b,(HL)	Indirect/registre	3	Z,H=1,N=0
BIT b,(IX+d)	Indexe /IX	5	Z,H=1,N=0
SET b,r	Implicite	2	Aucun
SET b,(HL)	Indirect/registre	4	Aucun
SET b,(IY+d)	Indexe/IY	6	Aucun
RES b,r	Implicite	?	Aucun
RES b,r	Indirect	?	Aucun
RES b,r	Indexe/registre	?	Aucun

INSTRUCTIONS DE DECALAGE

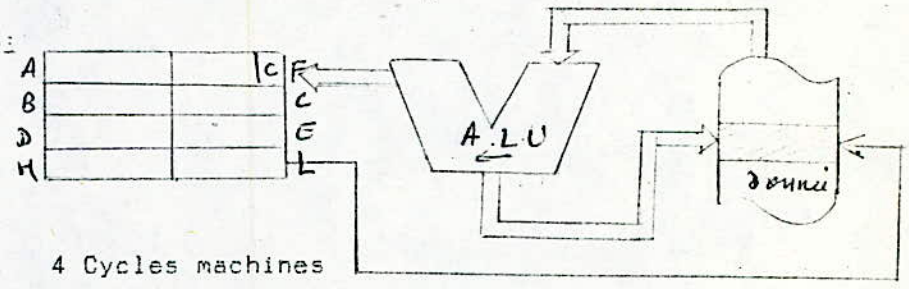
IL existe un tres grand nombre de decalage possible,la plupart de ces decalage utilisant le bit de retenue C qui de ce fait peut etre consedere comme un neuvieme bit participant au decalage.

ROTATION CIRCULAIRE
****
*Il en existe deux types une a gauche l'autre a droite.

RLC (HL) — Rotation a gauche sans le report de l'emplacement memoire (HL).

COMMENTAIRE: Le contenu de l'emplacement memoire adresse par HL est decalale circulairement vers la gauche d'un bit et le resultat est range a nouveau a cet emplacement le contenu du bit 7 va dans l'indicateur de report en meme temps que dans le bit 0.

CHEMIN DES DONNEES:



DUREE: 4 Cycles machines

MODE D'ADRESSAGE: Indirect

INDICATEURS: Le flag C est positione par le bit7 de l'octet.

EXEMPLE:

Programme	Code Objet		Avant	Apres
RLC (HL)	CB,06	(F)	00	00
HALT	76			
		(HL)	19 00	19 00
		(1900)	81	81

INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS
RLC r	Implicite/registre	4	C=bit 7
RLC (HL)	Indirect	4	C=bit 7
RLC (IX+d)	Indexe/IX	6	C=bit 7
RRC r	Implicite/registre	4	C=bit 7
RRC (HL)	Indirect	4	C=bit 7
RRC (IY+d)	Indexe IY	6	C=bit 7

*Les instructions RLC et RRC effectuent la meme chose sur l'accumulateur.

ROTATION CIRCULAIRE A TRAVERS LA RETENUE
****

INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS
RL	Implicite/registre		C=bit 7
RL	Indirect		C=bit 7
RL	Indexe IY		C=bit 7
RR	Implicite/registre		C=bit 7
RR	Indirect		C=bit 7
RR	Indexe IY		C=bit 7

ROTATION CIRCULAIRE A TRAVERS LA RETENUE:
****

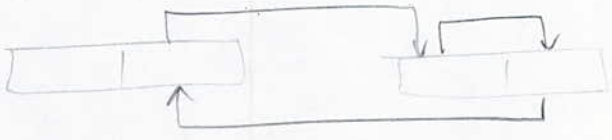
Rotation arithmetique decimale .L'octet est separe en deux mois de 4 bits le DCB operent sur des mots de 4 bits.

La destination est l'accumulateur ,le scond mot source est designe par un adressage indirect par registre:

RLD:

a gouche

RRD:



a droite

INSTRUCTION	MODE D'ADRESSAGE	DUREE	FLAGS AFFECTES
RLD	Indirect/Reg	18 Cycs	S,Z,H,=0,P,N=0,C
RRD	#	18 #	# # #

****

PARTIE (4)

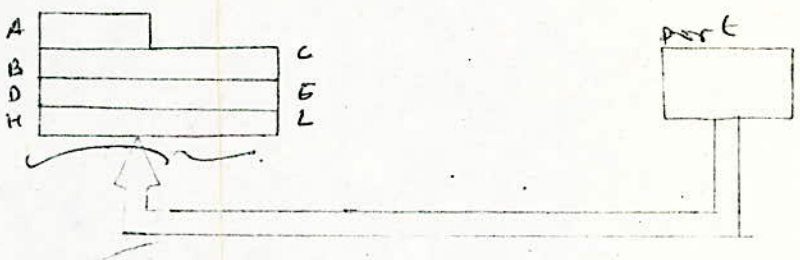
Partie <4>

ENTREE-SORTIE DE CHAINE

IN r,(c) — Chargement du registre r a partir du port (c).
r<---(C)

COMENTAIRE: L'organe peripherique adresse par le registre(c)est lu,le resultat est charge dans le registre specifie (c) fournit les bits A0 a A7 B fournit les bits A8 a A15 du bus d'adresse.

CHEMIN DES DONNEES:



DUREE: 5 Cycles machines

MODE D'ADRESSAGE: Externe

INDICATEURS:

X	X	!	X	!	X	Ø	!

REM: Il a noter que IN A,(n) n'a aucun effet sur les indicateurs alors que IN r,(c) en a.

EXEMPLE: IN D,(C)

Programme:		Avant		Apres
ED				
50		A5 (c)		A5 (c)
76				
	(D)	19	BA (L)	19 BB A5 BA

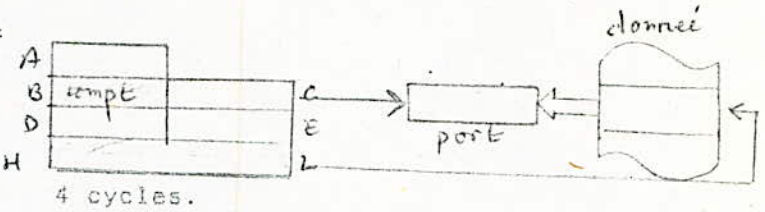
INSTRUCTION	MODE D'ADRESSAGE	DUREE(cycles)	INDICATEURS
IN A,(c)	Direct	3	Aucun effet
IN r,(c)	Indirect/registre	3	S,Z,H,P,N=Ø
OUT (n),A	Direct	3	Aucun effect
OUT (c),r	Indirect/registre	3	S,Z,H,P,N=Ø

ENTREE SORTIE DE CHAINE

OUTI: Sortie avec incrementation.
(c)<---(HL);B<---B-1;HL<---HL+1

COMMENTAIRE: Le contenue de l'emplacement memoire adresse par le registre double HLest ecrit dans l'organe peripherique adresse par le registre C .Le registre B est decremente et registre double HL est incremente C fournit les bits A0 a A7 du bus adresse B(apres decrementation) fournie les bits A8 a A15.

CHEMIN DE DONNEES:



DUREE: 4 cycles.

MODE D'ADRESSAGE :

S	Z	H	P/V	N	C
?	X	?	?	1	

——Positionne si B=0 apres l'execution,efface si non

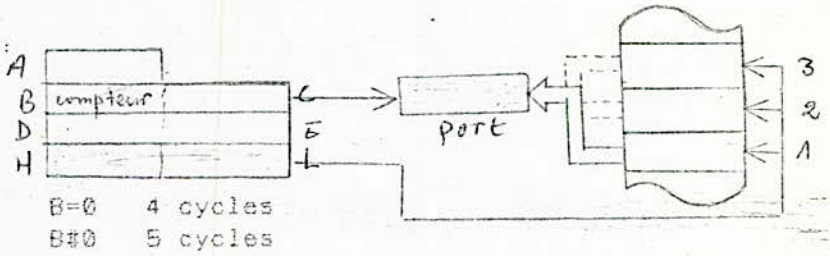
EXEMPLE: OUTI

	Avant:		Apres:	
Programme:				
1800 ED	B 9A	BB C	B 99	1801 A
1802 7B	H 1A	9A L	H 1A	9B L
		02 Port		0A Port
		BB		BB

OTDR: Sortie par bloc avec decrementation.
(C) <--- (HL); B <---B-1;HL <---HL-1;Repetr jusqu'a B=0.

COMMENTAIRE: Le contenue de l'emplacement memoire adresse par le registre double HL est ecrit dans l'organe periphérique adresse par le contenue du registre C.
Le registre B et le registre double HL sont tous deux decrementes.Si B#0 le compteur ordinal est decremente de deux et l'instruction est executee .C fournit les bits A0 a A7. du bus adresse et B fournit (apres decrementation les bits A8 a A15.

CHEMIN DES DONNEES:



DUREE: B=0 4 cycles
B#0 5 cycles

MODE D'ADRESSAGE: Externe

INDICATEURS: ? 1 ? ? 1

EXEMPLE: OTDR

Programme:

1800	ED
1801	BB
1802	76

	Avant:				Apres:		
B	02	E5	C	B	00	E5	C
H	19	05	L	H	19	03	L
		32	Port			68	Port
		E5				E5	

1903	02
1904	68
1905	9A

1903	02
1904	68
1905	9A

****

INSTRUCTIONS DE CHAINES

Ces instructions tres puissantes du Z80 travaillant sur un espace memoire de plusieur octets contigues.Elle se scindent en trois groupes:

-Instructions de transfert de chaine
-Instructions de recherche d'octet dans une chaine.
-Instructions d'entree-sortie de chaine.

	Instruction	Adressage	Duree	Flag affectee	Remarque
Transfert de chaine	(DE)=(HL) DE =DE+1 HL =HL+1 BC =BC-1 **LDI**	Indirect	4 Cys	H=0,N=0 P/V=0 si BC-1=0	
	(DE)=(HL) DE =DE+1 HL =HL+1 BC =BC-1 **LDIR**	Indirect	4 Cycs BC=0 5 Cycs BC#0	H=0,P/V=0 N=0	BC=0
	(DE)=(HL) DE =DE-1 HL =HL-1 BC =BC-1 **LDD**	Indirect	4 Cycs	H=0,N=0 P/V=0 si BC-1=0	
	(DE)=(HL) DE =DE-1 HL =HL-1 BC =BC-1 **LDDR**	Indirect	4 Cycs BC=0 5 Cycs BC#0	H=0,P/V=0 N=0	Instruction se repete BC=0
Recherche d'octet dans une chaine	A-(HL) HL=HL+1 BC=BC-1 **CPI**	Indirect	4 Cycs	S,H,N=1,Z=1 si A=(HL),P/V=0 si B-A=0	
	A-(HL) HL=HL+1 BC=BC-1 **CPIR**	Indirect	4 BC=0 5 BC#0	# # #	BC=0 ou A=(HL).
	A-(HL) HL=HL-1 BC=BC-1 **CPD**	Indirect	4 Cycs	# # #	
	A-(HL) HL=HL-1 BC=BC-1 **CPDR**	Indirect	4 BC=0 5 BC#0	# # #	instruction --> BC=0

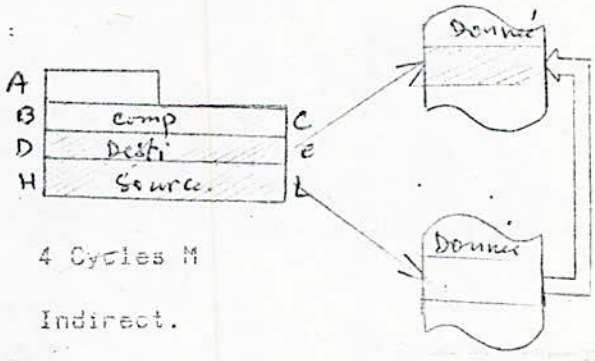
LDI: Transfert de bloc avec incrementation.
(DE) <----(HL);DE <---DE+1;HL<---HL+1;BC <---BC-1

COMMENTAIRE: Le contenue de l'emplacement memoire adresse par HL est charger dans l'emplacement memoire adresse par DE
Ensuite DE et HL sont tous deux incrementes et le

LDI: Transfert de bloc avec incrementation.
(DE) <----(HL);DE <---DE+1;HL<---HL+1;BC <-- BC-1

COMMENTAIRE: Le contenue de l'emplacement memoire adresse par HL est charger dans l'emplacement memoire adresse par DE Ensuite DE et HL sont tous deux incrementes et le registre double BC est decremente.

CHEMIN DES DONNEES :



DUREE: 4 Cycles M

MODE D'ADRESSAGE: Indirect.

INDICATEUR:

S	Z	H	P/V	N	C
		0	X	0	

*Positionne si BC=0 apres l'execution
*Effacer si non

EXEMPLE: LDI

Programme:

1800	ED
1801	A0
1802	76

Avant:

B	00	05	C
D	19	30	E
H	19	0A	L

1930 0A

190A 48

Apres:

B	00	04	C
D	19	31	E
H	19	0B	L

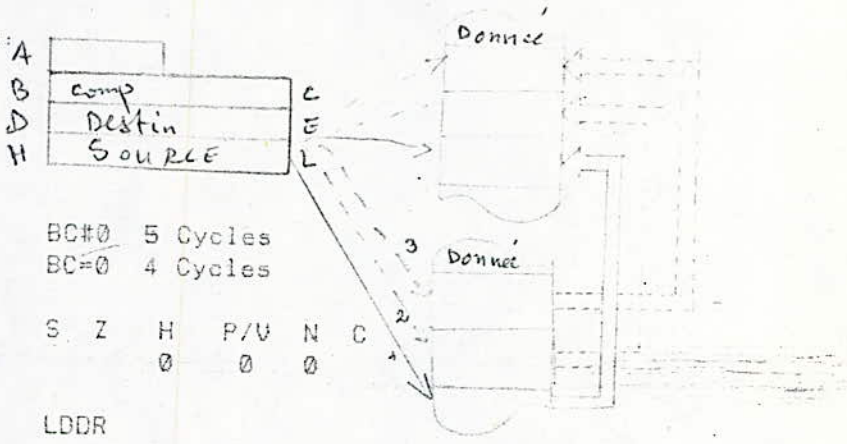
1930 38

190A 48

LDDR: Transfert relatif de bloc avec decrementation
(DE) <---(HL);DE <---DE-1;HL <---HL-1
BC <--- BC -1;repeter jusqu'a BC=0

COMMENTAIRE: le contenue de l'emplacement memoire adresse par HL et BC sont tous trois decrementes si B#0, le compteur ordinal est decremente de 2 et l'instruction est reexecutee.

CHEMIN DES DONNEES:



DUREE: BC#0 5 Cycles
BC=0 4 Cycles

MODES D'ADRESSAGE:

S	Z	H	P/V	N	C
		0	0	0	

EXEMPLE: LDDR

Programme:

1800	ED
1801	B8
1802	76

Avant:

B	00	03	C
D	19	08	E
H	1A	35	L

1905	B1
1906	04
1907	DF
1908	36
1A32	92
1A33	DE
1A34	E1
1A35	BF

Apres:

B	00	00	C
D	19	05	E
H	1A	32	L

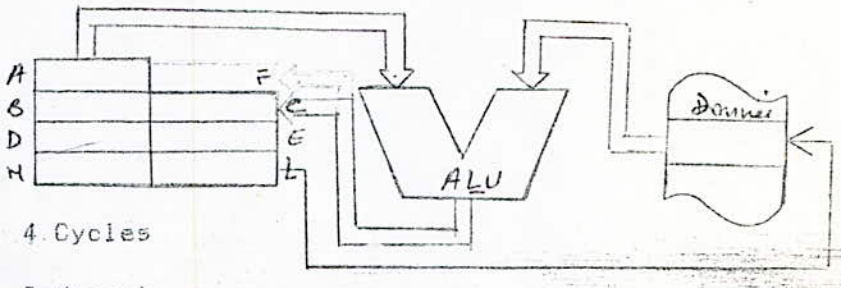
1905	B1	
1906	DE	
1907		
1908		
1A32	92	1A32
1A33	DE	↓
1A34	E1	↓
1A35	BF	↓

EXEMPLE DE RECHERCHE D'OCTET DANS UNE CHAINE

CPI: Comparaison avec incrementation
A-(HL);HL <--- HL+1;BC <---BC+1

COMMENTAIRE: Le contenue de l'emplacement memoire adresse par le registre double HL est soustrait du contenue de l'accumulateur et le resultat n'est pas conservee Le registre double HL est incrementeet le registre BC est decremente.

CHEMIN DES DONNEES:



DUREE: 4 Cycles

MODE D'ADRESSAGE: Indirect

INDICATUER:

S	Z	H	P/V	N	C	
						Effacesi BC=0 apres l'execution.
			!			! positionne si non
	!					! positionne si A=(HL)

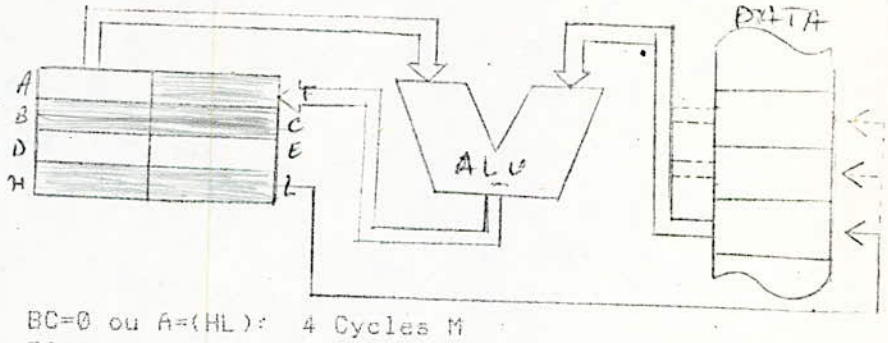
EXEMPLE: CPI

Progamme:		Avant:				Apres:			
1800	ED	A	09	00	F	A	09	16	F
1801	A1	B	05	10	C	B	05	0F	C
1802	76								
		H	1B	B9	L	H	1B	BA	L
			1BB9	9B			1BB9	9B	

CPDR: Comparaison par bloc avec decrementation
A-(HL),HL <--- HL-1;BC <--- BC-1;
Repeter jusqu'a BC ou A=(HL)

COMMENTAIRE: Le contenue de l'emplacement memoire adresse par le registre double HL est soustrait du contenu de l'accumulateur et le resultat n'est pas conserve Ensuite chacun des deux registres doubles BC et HL est decremente.Si BC#0 et A#(HL),le compteur ordinal est decremente de deux et l'instruction est reexecute

CHEMIN DES DONNEES:



DUREE:

BC=0 ou A=(HL): 4 Cycles M
BC#0 et A#(HL): 5 Cycles M

INDICATEURS:

S	Z	H	P/V	N	C	
*	x	*	x	1	!	- Efface si BC=0 apres l'execution;positionne si non
	!					Positionne si N=(HL)

EXEMPLE:

CPDR

Programme:

```
1800   ED
1801   B9
1802   76
```

	Apres:			Apres:		
A	9A	00	F	A	9A	B2	F
B	00	02	C	B	00	00	C
H	19	00	L	H	18	FE	L
	18FE	08			18FE	08	
	18FF	00			18FF	00	
	1900	2A			1900	2A	

ENTREE-SORIE DE CHAINE

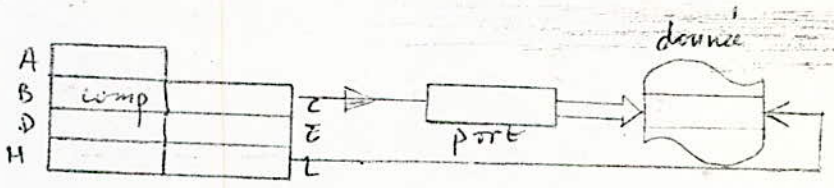
INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS	REMARQUE
INI (HL)=(c) B=B+1 HL=HL+1	Externe	4	Si B-1=0. N=1,Z=1	R-A-S
INIR (HL)=(c) B=B-1 HL=HL-1	Externe	B#0:5 B=0:4	N=1;Z=1	Jusqu'a B=0
IND (HL)=(c) B=B-1 HL=HL-1	Externe	4	N=1;Z=1 Si B-1=0	R-A-S
INDR (HL)=(c) B=B-1 HL=HL-1	Externe	B#0:5 B=0:4	N=1;Z=1	Jusqu'a B=0
OUTI (c)=(HL) HL=HL-1 B=B-1	Externe	4	N=1;Z=1 Si B-1=0	R-A-S
OTIR (c)=(HL) HL=HL+1	Externe	B#0:5 B=0:4	N=1;Z=1	Jusqu'a B=0
OUTD (HL)=(c) HL=HL-1 B =B-1	Externe	4	N=1;Z=0 Si B-1=0	R-A-S
OTDR (c)=(HL) HL=HL-1(c) B =B-1	Externe	B#0:5 B=0:4	N=1:Z=0	Jusqu'a B=0

ENTREE-SORTIE DE CHAINE

INI — Entree avec incrementation
(HL)<---(C);(B)<---(B-1);(HL)<---(HL+1)

COMENTAIRE: L'organe peripherique adresse par le registre est lu et le resultat est charge dans l'emplacement memoire adresse par le registre (HL) . Le registre (B) est decremente et le registre double (HL) est incremente Le contenu de (C) est place sur la moitie basse du bus adresse.Le contenu de (B) est place sur la moitie haute,La selection des ports d'entree-sortie est generalement faite par (C),c'est a dire A0-A7.B sert de compteur.

CHEMIN DES DONNEES:



DUREE: 4 Cycles machines

MODE D'ADRESSAGE: Externe

INDICATEURS:

?	X	!	?	!	?	1	!

REM: Le flag Z est positione si B=0 apres l'execution,efface si non.

EXEMPLE: INI

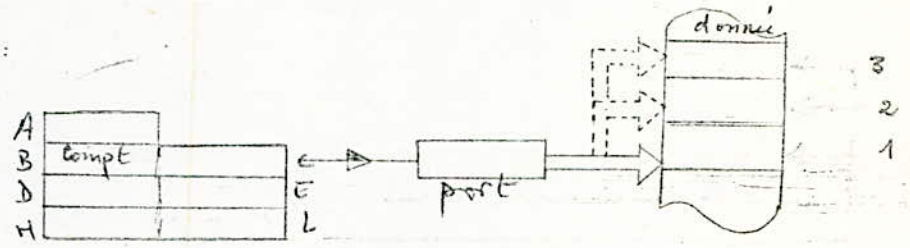
Progcamme:		Avant			Apres	
ED						
BA	(B)	09	21	(C)	08	21
76	(H)	19	BA	(L)	19	BB
	(PORT)	86			86	
	(19BA)	09			09	

ENTREE-SORTIE DE CHAINE (SUITE)

INDR — Entree par bloc avec decrementation
(HL)<---(C);(B)<---(B-1);(HL)<---(HL-1)

COMENTAIRE: L'organe peripherique adresse par le registre (C) est lu et le resultat est charge dans l'emplacement memoire adressee par le registre(HL).Le registre (B) et le registre double (HL) sont incrementes.si le registre(B) est non nul,le compteur ordinal est decremente de deux(2) et l'instruction est reexecutee.

CHEMIN DES DONNEES:



DUREE: 4 Cycles machines Si B ==0
5 # # # # Si B=/=0

MODE D'ADRESSAGE: Externe

INDICATEURS:

?	1	X	?	X	?	1	!

EXEMPLE: INDR

Programme:
ED
BA
76

	Avant		Apres
(B)	03	(C)	00 56
(H)	19	(L)	19 03
(PORT) (1)	B6		BF
(1903)	6A		6A
(1904)	EB		BF
(1905)	48		2A
(1906)	9A		BB

INSTRUCTION D'USAGE GENERAL

*Les instructions que nous donnons ici sont d'un usage tres particulier et ne sont pas toujours facilement utilisables nous en donnons la liste sans detailler.

INSTRUCTIONS SUR LES INTERRUPTIONS

INSTRUCTION	MODE D'ADRESSAGE	DUREE	INDICATEURS
DI	Implicite	1	Aucun effet
EI	# #	1	# # #
IM0	# #	2	# # #
IM1	# #	2	# # #
IM2	# #	2	# # #
RET.	# #	4	# # #
RETN	# #	4	# # #

INSTRUCTIONS DE CONTROLE

*Elles ne sont que deux et ne dure chacune qu'un cycle machine de plus elles n'affectent pas les indicateurs,et ne possedent pas de mode d'adressage.

DI: Interduction des interruptions
IFF <--- 0

COMMENTAIRE: Les bascules d'interruption sont mises a zero interdisant ainsi toutes les interruptions masquables elle reautorisee par l'instruction EI

DUREE: 1 Cycle M

MODE D'ADRESSAGE: Implicite

INDICATEURS: Aucun effet

++

HALT: HALT de l'unite centrale
Suspension du fonctionnement et execute des NOP pour continuer les cycles de rafraichissement de la memoire jusqu'a recevoir une interruption ou une reinitialisation.

DUREE: 1 Cycles M

MODE D'ADRESSAGE: Implicite

INDICATEURS: (AUCUN EFFET).

****

SERIE -3-

IP N°: SUR L'ADDITION 16 BITS:

Tout report genere par l'addition des deux octets sera automatiquement memorise dans le registre d'etat C:
Soit les adresse suivantes: POLY 1
POLY 2
POLY 3
Ni operande i=1-3.

```
LD A,(POLY1)      Charger la partie basse de POLY1
LD HL,POLY2       Adresse de la partie basse de N2
ADD A,(HL)        Additionner les partie basses de N1 et N2
LD (POLY3),A      Ranger la partie basse du resultat
LD A,(POLY1-1)    Charger la partie haute de N1
DEC HL            Adresse partie haute de N2
ADC A,(HL)        (N1+N2) partie haute+report eventuel
LD (POLY3-1),A    Ranger la partie haute du resultat
HALT
```

Questions:

1) Ecrire le programme en langage machine .
2) Introduire les valeurs N1 et N2 en Hex dans les adresses: POLY1 et POLY2 resp.
3) Executer le p rogramme et verifier le resultat dans l'adresse POLY3

IP N°: SUR LA SOUSTRACTION.16 BITS:

Soustraire des nombres de 16 bits:
Soit les deux nombres N1 et N2 sont ranger aux adresses: OUA1 et OUA2 la soustraction recourra a l'instruction SBC:

Soit le programme:

```
LD HL ,(OUA1)     N1 dans HL
LD DE,(OUA2)      N2 dans DE
AND A             Met l'indicateur <<C>> a zero "0"
SBC HL,DE         N1-N2
LD (OUA3),HL      Resultat en OUA3
HALT              Stop
```

Questions:

1) Ecrire le programme en langage machine
- Choisir les nombres N1 et N2 ranger-les aux adresses correspondantes
- Executer le programme
- Verifier le programme dans l'adresse OUA3
- Verifier le carry "C"

2) Reecrire le programme de soustraction sans utiliser les instructiions sur 16 bits

3) Ecrire le programme de soustraction pour des operandes sur 8 bits

*** Remarque: Il faut se souvenir que dans le cas de l'arithmetique en complement a deux la valeur finale de l'indicateur de report n'a pas de signification si un debordement a lieu a la suite de la soustraction, le bit indicateur de debordement (bit V) est positionne. Ilpeut etre teste.

IP N: SUR LA MULTIPLICATION 8 BITS PAR [redacted] 8 BITS

Ce programme permet d'effectuer la multiplication d'un mot de 8 bits par un mot de 8 bits .Le multiplicateur est dans l'adresse HEWLET et la et le multiplicande dans l'adresse PACKARD,le resultat dans l'adresse IBM soit MPR:Multiplicateur ;.MPD:Multiplicande.

Programme:

```
        LD  BC,(HEWELETT)   Multiplicateur dans C
        LD  B,8             B sera utiliser comme compteur
        LD  DE,(PACKARD)    Multiplicande dans E
        LD  D,0             Initialise D a "0"
        LD  HL,0            Initialise le resultat a "0"
Ho      SRL C               Decale un bit du MPR dans l'indicateur
                            de report
        JR  NC,H1           Test l'indicateur de report
        ADD HL,DE           Additionne MPD dans au resultat
H1      SLA E               Decale MPD vers la gauche
        RL  D               En sauvant le bit sorti dans D
        DEC B               Decremente le compteur de bit
        JP  NZ,Ho           Et recommence tant qu'il est # de "0"
        LD  (IRM),HL        Range le resultat
        HALT                Stop
```

Questions:

1) Ecrire le programme en langage machine
2) Choisir le MPR et MPR chargez-les dans leurs adresses correspondantes.
3) Verifier le resultat dans l'adresse IBM.
4) Reecrire le meme programme de multiplication en utilisant l'instriction BIT a la place de l'instruction SRL C.
   -QUels sont ici les desavantage?
5) Peut-on substituer l'instruction JR a l'instruction JP a la fin du programme ? Si oui,quel est l'avantage?
6) Pour les instructions :LD D,0 et LD HL,0 au debit du programme.Peut-on leur substituer la sequence suivante:

```
XOR  A
LD   D,A
LD   H,A
LD   L,A
```

   -Si oui,quel est l'impact de cette substitution sur la taille du programme (nombre d'octets) et sur sa vitesse?

IP N: SUR LA MULTIPLICATION 16 BITS PAR [redacted] 16 BITS

Ce programme permet d'effectuer la multiplication d'un mot de 16 bits par un mot de 16 bits.Le multiplicateur est dans l'adresse HEWLETT et HEWLETT+1 et le multiplicande dans l'adresse PACKARD.
Soit: MPR:Multiplicateur ; MPD:Multiplicande.

Programme:

```
        LD  A,(HEWLETT+1)   MPR partie haute
        LD  C,A
        LD  A,(HEWLETT)     MPR partie basse
        LD  B,16D           Compteur
        LD  DE,(PACKARD)    MPD
        LD  HL,0
Ho      SRL C               Decalage a droite de MPR haut
        RRA                 Rotation a droite de MPR bas
        JR  NC,H1           Tester l'indicateur C
        ADD HL,DE
H1      EX  DE,HL
        ADD HL,HL           Decalage a gauche de MPD
        EX  DE,HL
        DJNZ  Ho
        HALT                Stop
```

Questions:

1) Ecrire le programme en langage machine
2) Choisir le MPR et MPR chargez-les dans leurs adresses correspondantes.
3) Verifier le resultat
4) Mettre le multiplicateur dans les registre B et C,et le compteur dans A.
   -Ecrivez le programme de multiplication correspondante, et discutez des avantages et des inconvenients de cette allocation de registres.
5) Proposer un moyen de decaler le MPD contenue dans les registre D et E,sans transferer dans la paire HL.
6) Ecrivez un programme de multiplication 16*16 bits qui qui precise si le resultat tient sur plus de 16 bits (Rem:il s'agit d'une simple amelioration du programme)
7) Ecrire un programme de multiplication 16*16 dont le resultat tient sur 32 bits

****

IP N: SUR LA DIVISION 16 BITS PAR 8 BITS

Ce programme permet la division 16 par 8 qui produira un cotient sur 8 bits, et un reste sur 8 bits egalement.MSX est l'adresse du diviseur YAMAHA est l'adresse de dividande .Soit:DVS:Diviseur;DVD:Dividanded

Programme:

```
          LD  A,(MSX)        Diviseur
          LD  D,A            dans D
          LD  E,0
          LD  HL,(YAMAHA)    Dividende sur 16 bits
          LD  B,8
Ho        XOR A              Indicateur C a zero
          SBC HL,DE          DVD-DVS
          INC HL             Quotiont=Quotont+1
          JP  P,Ho           Rest positif?
          ADD HL,DE          Restaurer si necessaire
          DEC HL
H1        ADD HL,HL          Decaler dividende
          DJNZ Ho            Boucler jusqu'a B=0
          HALT               Stop
```

Questions:

1) Ecrire le programme en langage machine
2) Choisir le DVS et DVD chargez-les dans leurs adresses correspondantes.
3) Verifier le resultat
4) Verifier le fonctionnement du programme a la main en remplissant le tableau suivant :

Etiquette	Instruction	B	H	L

SERIE DE T.P N:

1: MISE A ZERO D'UNE ZONE DE MEMOIRE

Notre butet de mettre a zero d'une le contenue des emploicements memoires de l'adresse:par exemple appele ADR a l'adresse ADR+d, telque d etant inferieur a 256.

```
Programme:      Mise a zero  LD B,d      d dans B
                             LD A,Ø
                             LD HL,ADR   pointe sur la zone zero
                H0           LD (HL),A   dans la case memoire
                             INC HL      Pointe sur la caze suivante
                             DEC B       Decremente le compteur
                             JR NZ,HØ    Fin de la zone
                             HALT        Stop
```

QUESTIONS:

-1) Mettre le programme en langage machine.
-2) Executer-le et verifier si les cases memoire sont a zero.
-3) -Modifier le programme pour la mise a "1" d'une zone memoire
-Verifier S'il sont a "1".

*2 TROUVER LE PLUS GRAND ELEMENT D'UNE TABLE

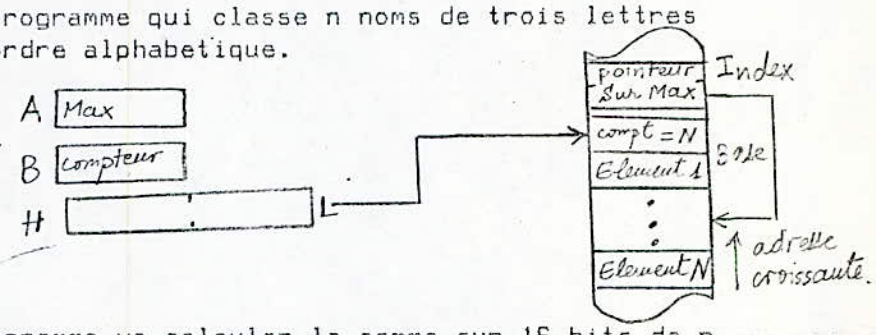
L'adresse du debut de la table se trouve a l'adresse memoire "ADR" en page zero, sa premiere entree est le nombre d'octets qu'elle contient.
Ce programme decouvrira le plus grand element d'une table dans un mot situe a l'adresse SAROU.
Il utilise les registres A,B,H,et L et l'adressage indirect,de maniere a pouvoir travailler sur des tables situees a n'importe quel endroit de la memoire.

```
PROGRAMME:      MAX     LD HL,ADR       Adresse de la table
                        LD B,(HL)       Nombre d'elements de la table
                        LD A,Ø          Initialiser la plus grande valeur
                        INC HL          HL pointe sur la premere entree de
                                        la table.
                        LD (SAROU),HL   Initialise le SAROU.
                H1      CP (HL)         Compare avec entree suivante
                        JR NC,HØ        Saut si inferieur au maximum
                        LD A,(HL)       Prendre nouvelle valeur  max
                        LD (SAROU),HL   Conserver pointeur sur nouveax max
                HØ      INC HL          Pointe sur valeur suivante
                        DJNZ H1         Continuer si n'est pas fini
                        HALT            Fin
```

**IL faut d'abord tester la n'ieme entree.Si elle est superieur a zero elle est mise dans A et son adresse.Conserver dans SAROU.L'entree (n+1) est ensuite testee etc...
**Ce programme <<marche>> pour les entiers positifs.

QUESTIONS: -1) Choisir une TABLE d'elements (entier positif)
-2) Mettre le programme en langage machine
-3) Executer-le.
-4) Verifier s'il est bien le plus grand element
-5) Modifier le programme pour qu'il marche aussi avec des nombres negatifs,en complement a deux
-6) Ecriver un programme qui classe n nombre en ordre croissant
-7) Marcherait-il avec des caracteres ASCII?
-8) Ecriver un programme qui classe n noms de trois lettres chacun par ordre alphabetique.

VOICI LE SCHEM DE PRINCIPE:



SOMME DE N ELEMENTS:

********** Ce programme va calculer la somme,sur 16 bits,de n elementsd'une table.L'adresse de debut de cette derniere se trouve a l'adresse memoire ADR, en page zero sa premere entree contient son nombre d'elements.
La somme sur 16 bits sera disposer dans les adresse memoires SOMHIGH,et SOMLOW.
Si elle est tres grande pour tenir sur 16 bits ,seul les 16 bits de poids faibles seront conserves (les 16 bitsde poids forts sont tronques).
Les registres A,B,C,D,et L IX sont modifies.
*On suppose que la table con tien au maximum 256 elements.

PROGRAMME:

```
SOMME   LD HL,ADR       Pointe sur debut TABLE
        LD B,(HL)       Nombre d'elements sur le compteur
        INC HL          HL pointe sur premiere entree
        LD IX,SOMLOW    Pointe sur partie basse du resultat
        LD A,0          Initialise...
        LD (IX+0),A     ...La partie Basse(LOW) et...
        LD (IX+1),A     ...La partie haute (HIGH) du resulat
H0      LD A,(HL)       Lit l'entree
        ADD A,(IX+0)    Calcul la somme partielle
        LD (IX+0),A     Et la range
        JR NC,H1        Teste la presence d'une retenue
        INC (IX+1)      Tient compte de la retenue si non
H1      INC HL          Pointe sur entree suivante
        DEC B           Decremente le compteur
        JR NZ,H0        Et continuer jusqu'a la fin
        HALT            Stop
```

QUESTIONS: -1) Choisir une table qui contient au maximum 256 elements
-2) Ecrire le programme en langage machine.
-3) Executer le programme.
-4) Modifier ce programme pour:
a)-Calculer une somme sur 24 bits
b)-## ## ## ## ## ## 32 bits
c)- Detecter si un debordement a lieu.

COMPTER DES ZERO:

Ce programme compte le nombre de zero dans une table qu'il faut la definir avant, et met le resultat a l'adresse RAH
Il modifie A,B,C,H,L.

PROGRAMME:

```
        LD HL,ADR      Pointe sur la table
        LD B,(HL)      Longueur de la table
        LD C,0         Initialiser RAH
        INC HL         Pointe sur premiere partie
H1      LD A,(HL)      Charger l'element dans A
        OR 0           Positionne l'indicateur Z
        JR NZ,Ho       Saut si different de zero
        INC C          Si non ajouter 1 au RAH
Ho      INC HL         Pointe sur entree suivante
        DEC B          Decremente le compteur
        JR NZ,H1
        LD A,C
        LD (RAH),A     Sauver le resultat
        HALT           Stop
```

QUESTIONS:

1)- Ecrire le programme en langage machine
2)- Executez-le
3)- Modifier le programme pour compter:
-a) Le nombre d'etoiles de la table
-b) # # de lettres de l'alphabet.
-c) # # de chiffres compris entre "0" et "9"

**********0********

COMPARAISON DE DEUX NOMBRES SIGNES SUR 16 BITS :

IX Pointe sur le premier nombre N1.
IY Pointe sur N2.

PROGRAMME:

```
        LD B,(IX+1)    Chercher le signe de N1.
        LD A,B
        AND 80H        Tester le signe,met l'indicateur C a "0"
        JR NZ,H0       Saut si N1 est negatif
        BIT 7,(IY+1)
        RET NZ         N2 est negatif
        LD A,B
        CP (IY+1)      Les signes sont tous les 2 positifs
        RET NZ
        LD A,(IX)
        CP (IY)
        RET
H0      XOR (IY+1)
```

(Suite page suivante)

(suite)

```
RLA             Bit de signe dans l'indicateur C
RET C           Les signes sont differents
CP (IY+1)       Les deux signes sont negatifs.
RET NZ
LD A, (IX)
CP (IY)
HALT            Stop
```

QUESTIONS:

1)- Choisir deux nombres N1 et N2
2)- Ecrire le programme en langage machine
3)- Introduire les deux nombres N1 et N2
    N1 ----> IX, N2 ----> IY
4)- Execute le programme en verifiant N1 et N2.

SERIE -4-

A) SERIES DE I.P N°4

I/ 1) Strapper OUT FF avec P1CL.
2) Charger le programme suivant:

ADDR	DATA	Mnemonique	Commentaire
1800	3E	LD A,55	A=55
1801	55		
1802	D3	OUT (FF),A	Appliquer le registre A
1803	FF		a la sortie FF.
1804	76	HALT	Stop.

* Appuyer sur PC ,GO et en fin BREAK.
* La valeur 55 est affiche en binaire sur le port 1.

Questions:
———————
*- Changer la valeur "55" d'adresse 1801 par une autre.
*- de meme changer le port de sortie en changeant l'adresse 1803 (1°-par FD, 2°-par FE)

II/ Visualisation des bits du bus de donnees,par le port deux "2"

*Strapper OUT FF avec P2X CL.
* # # P2X CL # P2Y CL.
*Strapper (Do ——>D7) avec (Lo ——>L7) respectivement.

Questions:
———————
*-Charger le programme precedent.
*-Lancer # # # en appuyant sur PC,GO,puis sur BREAK
*-La valeur choisie a l'adresse 1801 sera affectee au port 2, changer cette adresse plusieur fois et constater le changement sur le port 2 .

III/ Avec le meme montage precedent et avec le meme programme operer le changement suivant :

*Debrancher le strap P2X CL ——> P2Y CL
ainsi seules les adresses (Do — D3) sont connectees au port P2XCL c-a-d (Lo— L3),ainsi le quartet (L4— L7) et dans le troisieme etat (haute impedance),pour un octet donne a l'adresse 1801, on ne peut visualiser que le quartet de poids faible;pour visualiliser l'octet de poids fort il suffit de strapper (P2XCL—>OUT FF) et d'enlever le strap (P2Y CL ——> OUT FF).

IV/ Apres avoir visualiser l'octet ou l'un de ces deux quartet,on peut visualiser n'import quel bit individuelement en utilisant des instructions logiques.
1/* Faire le meme strappage que dans la partie (II)

2/* Charger le programme suivant:

```
1800   3E   LD A,55      A=55
1801   55   AND A,N      Masquer des bits de l'octet
1802   E6
1803   40
1804   D3   OUT(FF),A    Appliquer le registre A a la
1805   FF                Sortie FF
1806   76    HALT        Stop
```

N=40 ====> 0100 0000.
A AND N <===>55 AND 40

```
          0101 0101
and       0100 0000
          ---------
          0100 0000
```

**REM: Tous les bits sont a 0 sauf.Le bit N°6 qui n'a pas changer et qui apparaitre sur le port 2,on peut jouer sur la valeur de N pour balayer tous les bits de la valeur contenue dans le REG A.

SERIE DE T.P N°

Le but de cette experimentation a pour but d'enseigner l'utilisation des interrupteurs logiques:

-En utilisant les interrupteurs du port 2,adresser des niveaux logiques 1 ou 2.Sur les terminaux So a S7 de la barrette d'interconnaixions (port socket).

*Remarque:ces terminaux peuvent etre utilises pour appliquer des signaux sur les circuits d'interface monter sur le support de cablage sans soudure.

-Utiliser les ports 1 et 2 pour adresser des donnees au Z80.

*Avant T.P:preparer { 8 straps de 15,2 cm mono brin de 0,8 }
{ 3 # # # 7,6 cm # ## ## ## }

I)- Utilisation des interrupteurs logiques du port 2
1-)En srappant par les long fils en liant So a Lo ————>S7 a L7.
2-)En agissant sur les interrupteurs du port 2 en constate que les diodes temoins du port 2 afficheent des valeurs logiques correspondantes.

Q1:en jouant sur les interrupteurs afficher quelques valeurs.

II)-Utilisation des interrupeurs du port 2 pour definir deux ports de sortie independantes de chacun de 4 bits :Port 2X et Port 2Y.
-Comment utilises comment utiliser P2X EN et P2Y EN pour definir un 3eme etat sur lo port socket.

**Strapper avec un strap court relier P2X EN a P2Y EN.
Q1: Faire apparaitre la valeur 1111 1111 sur les diodes.

**En strappant P2Y EN a IN FD avec un fil court.
Q2: Que constatez vous sur diodes temoins ?.
Commentez?

III) Retirer le strap reliant P2Y EN
Q1: Que constatez-vous? commentez?

**Strapper P2Y EN avc P2Y EN .
Q2: Faite votre remarque .

IV)-Utilisation des interrupteurs logiques du port 2 comme port d'enter les adresse disponibles,sont pour le port 2:FD et FE.
**debrancher le MT-80Z en effectant le strappage suivant:
— P2X EN et P2Y EN
— P2Y EN et IN FD
— P1 CL et OUT FF
— So--S7 sur le port socket et Do--D7

*Les interrupteurs du port 2 sont maintenant interfaces au bus de donnees du MT-80Z en tant que port d'entree FD.

*Les indicateurs logiques du port 1 sont inferfaces au bus de donnees en tant que port de sortie FF.

*Mettre le MT-80Z sous tention.

*Charger le programme suivant:

```
1800   DB   'IN A,(FD)   entrer la donnee de l'interr sur l'ACC
1801   FD
1802   D3   OUT(FF),A
1803   FF
1804   18   JR dis       Branchement relatif sur l'accumulateur
1805   FA
```

Q1: executez le programme.
*Mettre les interrupteurs logiques du port 2 sur les positions 0000 1111. - qu'affichent les diodes?
Q2: Si on retirant le strap reliant D7 a S7. Que.constatez-vous? Faites vos commentaires.

V)- *Debrancher le MT-80Z.
*Strapper comme suivant:
-a) Strapper de S0 a S7 et D0 a D7 respectivement (S0-D0 ....)
*Mettre sous tension charger et executer le programme liste ci-dessus

Q1: En agissant sur les interrupteurs 4,5,6,7 du port 2.
-Que constatez-vous ? Commentez?
Q2: Si en change l'instruction IN A(FD) en IN A(FE)
-Que constatez-vous ? Commentez?

VI)-*Debrancher le MT-80Z.
*Retirer tous les straps
*Utiliser les interrupteurs du port 1 comme port d'entrer FF.
*Srapper sur le port socket.(avec des fils long)
-1) P1 CL et OUT FF
-2) P1 EN et IN FF
*Mettre suos tesion ,et charger le programme:

```
1800   DB     ( Meme programme sauf qu'on a charger
1801   FF       INA(FD), en INA,(FF)>
1802   D3
1803   FF
1804   18
1805   FA
```

Q1: Executer-le .Que constatezr-vous en ajussant sur les interrupteurs du port 1.

Etablissement d'un programme en visualisant par l'intermediaire des leds en commence par la valeur 0000 0000 en augmentant jusqu'a la valeur FFH.

**Strapper avec un court fil (7 cm) PiCL-OUTFF.

Voici le programme:

```
XOR A              Initialisation de A
LD B,0             Charger B,0. B=0
OUT (FF)           Sortir A sur le port FF
LD E,FF            Charger E par FF
DJNZ etiquette 0   Boucle
DEC E              Decrementation de E
JR NZ etiquette 1  Boucle
INC A              Incrementation de A
JR NZ etiquette 2  Saut a l'etiquette 2
JR etiquette 3      #          #      3
HALT               Stop.
```

QUESTIONS:

1)- Chercher l'etiquette 0,1,2,et 3.
2)- Executez le programme ,que constatez-vous sur les leds
3)- Refaite le programme en commancant par FF est decremente jusqu'a 0000 000.
4)- Faite par le programme deja cite,et votre programme la somme pour avoir un seul programme qui incremente et decremente.
5)- Faite un programme qui incremente , puis il se stabilise puis il decremente .
6)- Faite un programme qui permet de visualiser des 1 et des 0 les leds .

****

CH

3

ENTREES/SORTIES

INTRODUCTION

On appelle entree,l'action d'amener des donnees dans le systeme (MPU et memoire),depuis des peripheriques exterieurs (clavier...
on appelle sortie l'action de trnsferer des donnees depuis le uP(ou la memoire)jusqu' des peripheriques exterieur,tels que imprimante disque,...

Notre but dans cette partie est d'essayer de generer, des signaux simples telque des impulsions ,et mesure de la duree des signaux .

Les instructions des entree/sortie du Z80 (voir jeux d'instruction).

GENERATION D'UN SIGNALE:

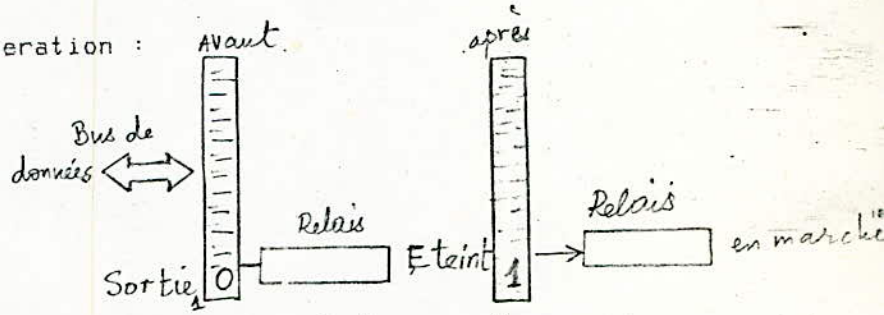
Pour activer ou desactiver un peripherique on doit faire passer un niveau de l'etat <0> a l'etat <1> ou l'inverse. Exemple:on suppose qu'on dispose d'un relai connecte au bit <0> d'un registre nomme <regi1> pour le mettre <ON> nous ecrivons un <1> dans dans le bit conserne du registre <regi1> voici un programme pour mettre <on> :

```
LD A,0000 0001H B    Charger la valeur 1 dans A
OUT (regi1),A
```

L'objectif est de fermer le relai (mettre on) sans changer l'etat d'aucun autre bit du registre le programme devient:

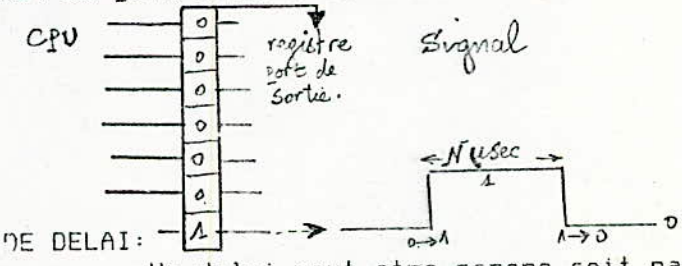
```
IN   A,(regi1)       Lit le contenu de regi1
OR   0000 0001B      Force le bit 0 a 1
OUT  (OUT1),A
```

Voici comment s'effectue l'operation :



IMPULSIONS:

La generation d'impulsion s'effectue de la meme facon une une sortie,plus un petit probleme c'est celui de la duree de l'impulsion (voir ci-desous) la generation d'un delai connu voir le paragraphe suivant.



GENERATION ET MESURE DE DELAI:

Un delai peut etre genere,soit par programme soit par un moyen hardware,on choisi le premier le 2eme est dans la definition du CTC.
Generation d'un delais au moyen d'un comptage.Un registre de comptage de comptage est d'abord charge avec une certaine valeur,puis decremente le programme boucle sur lui meme est decremente jusqu'a ce que la valeur atteint "0".

Voici un programme d'un delai de 130 cycles d'horloge:

```
        LD A,8          A sert de compteur
Ho      DEC A
        JR NZ,Ho
```

*(pour ces insructions le nombre de cycle machine voir la parite jeux d'instructions)
LD 7 cycles ; DEC 4 cycles ; JR 12 cycles (sauf pendant la derniere iteration,ou elle utilise 7)-si le branchement n'a pas lieu JR 7 cycles
-si il a lieu 12 cycles seront alors requis
Le delai est donc 7 pour LD,plus 16 pour les deux suivantes,fois le nbr d'execution de la boucle,moins un delai de 8 cycles pourJR,puisqu'aucun branchment n'a lieu: DELAI=7+16*8-5=130 cycles.
Si les cycles sont de .133us doc le delai est de 17,29 us
*Il est possible de generer des delais plus longs,en utilisant des compteur plus grands.

****

GESTION DES ENIREES/SORIIES

Plusieurs demandes d'E/S peuvent etre emises simultanement,trois technique de base sont emploiyees pour fixer l'ordre dans lequel les demandes deveront etre satisfaites.Ce sont l'interrogation,les interruptions; DMA.

INIERROGATION:C'est la methodes de getion de plusieurs peripheriques la plus simple.Dans cette strategie, le processeur interroge succe- ssivement les peripheriques connectes a ses bus.Si un peripherique demande un cervice;il le sert. S'il ne demande rien, le processeur examine le suivant La methode d'interrogation n'est pas utiliser pour les seuls peri-pheriques,mais aussi pour toutes les routines de service de peri-pheriques.

LES INTERRUPTIONS SUR LE Z80: Une interruption est un signal,une demande de service, envoye au uP.Elle est asynchrone au deroulement du programme.Lorsqu'un programme se branche a un sous-programme il est dit synchrone a l'execution du programme puisqu'il est prevu et ordonne' par lui une interruption suspend generalement l'execution du programme en cours sans que ce dernier le sache c'est pour cela qu'elle est dite asynchrone.

3 (trois) Interruptions cohabitent sur le Z80:

-Demande de Bus (BUSRQ,bus request).
-Interruption non masquable.(NMI non masquable interrupt)
-Interruption ordinaire INT.

**Remarque: Aucune interruption ne sera prise en compte avant la fin du cycle machine en cour.Les Interruptions MNI,INT ne puvant etre prises encompte qúa la fin de l'execution d'une instruction. Par contre l'interruption BUSRQ sera prise en compte des la fin du cycle machine en cours,sans attendre la fin de l'instruction La demandede bus (BUSQRQ) est le mecanisme d'interruption prio- ritaire du Z80.

BUSRQ (Bus request): Elle est utilise, pour l'accees direct memoire (DMA) en mode DMA le Z80 suspend son travail et met son bus de donnees et son bus d'adresse en haute impedance (Ce mode est utilise par un controler de DMA pour transferer des donnees agrande vitesse,entre la memoire et un peripherique en utilisant le bus d'adresses et le bus de donnees).Si un des deux signaux NMI ou INT est actif en meme temps avec BUSRQ son etat est memoriser par deux bascules internes:la bascule INT et NNI.

MNI: Elle ne peut etre empechee par le programmeur (sans masque) elle est toujours accepte par le Z80 a la fin d'execution de l'instruc-tion en cours en admettant que le signal BUSRQ ne sont pas actif. si une NMI est recuse pendant un BUSQRQ elle positionnera la bascule interne MNI et sera traitee a la fin du BUSRQ.

L' MNI: Provoque un empilage automatique du compteur ordinal PC et un branchement a l'adresse 0066H . Il n'est utilise que dans les cas de tres grande urgence. Le programme d'interruptiondoit etre charger dans l'adresse 0066H avant l'utilisation de l'MNI .L'MNI se branche automatiquement a 0066H .

Voici la sequence: (1) PC --->Pile (Sauve garde du compteur ordinal)
IFF1 --->IFF2 (sauve garde de IFF)
0 --->IFF1 (mise a Zero de IFF)
(2) Saut en 066H (executer la retine de taitement)

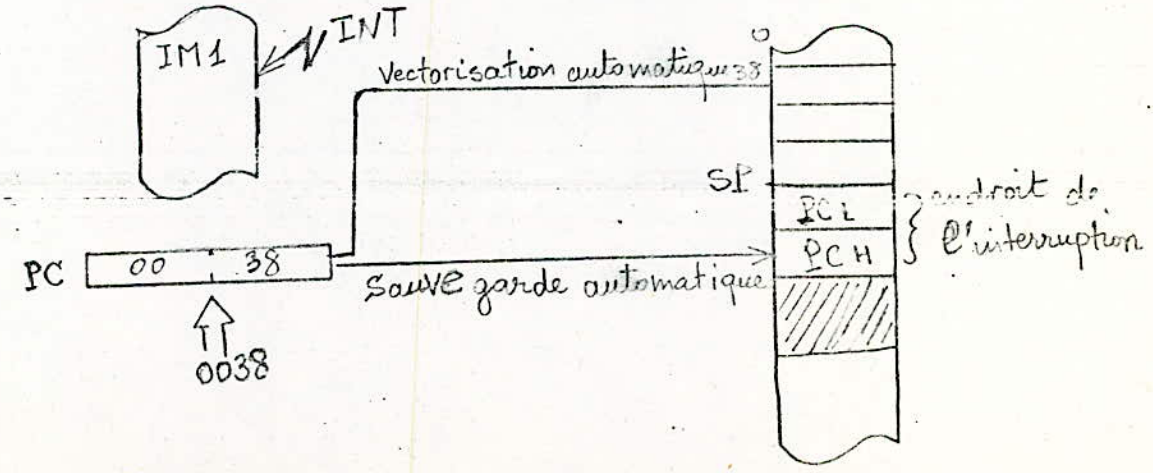
Le retour de la routine de gestion de l'MNI est assure par une instruction spetiale RETN (Return from Nom Masckable interrupt).

INTERRUPTIONS ORDINAIRES: * L'interruption normale;masquable,INT peut etre mise en oeuvre dans trois modes ,mode "0","1","2" (elle peut etre masquee par le programmeur). Mettre les bascules IFF1,et IFF2 a "1" autorise la prise en compte des interruptions INT; mettre a "0" empeche la detection des INT (sans masque).
- L'instruction EI permet de mettre ces bascules a "1".
- L'instruction DI permet de mettre ces bascules a "0".

MODE ZERO "0": Le Z80 travail en mode "0" lorsqu'il est initialise (lorsque le signal RESET est active') ou lorsque l'instruction IM0 (Int- erruption mode "0") a ete execute' INT ne pourra etre prise en compte que si la bascule IFF1 est a "1" de meme qu'un BUSRQ ou une MNI ne soit presente simultanement. Le role du Z80 lorsqu'il detecte une INT est d'activer les sig- naux M1 et IORQ,puis attendre.

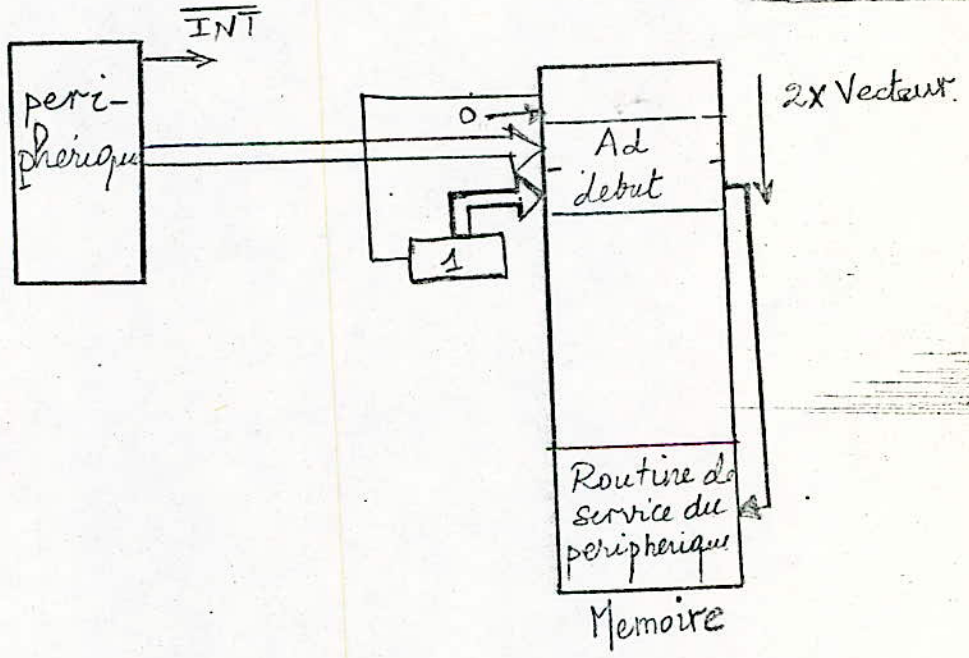
MODE UN "1": Mode "1" entre en vigueur avec l'execution de l'instruction IM1, il provoque un branchement automatique a l'adresse 0038H (pour toute les interruptions et permet de minimuser les circuits externes necessaire) apres sauve garde du compteur ordinal sur la pile.Il est comparable a celui du MNI a ceci presque l'interruption peut ici etre masquee.

**REMARQUE:Ici le Z80 doit ingnorer les presentes sur le bus pendant le cycle qui suit l'intrruption (le cycle de prise en compte de l'interruption the interrupt acknowledge cycle).

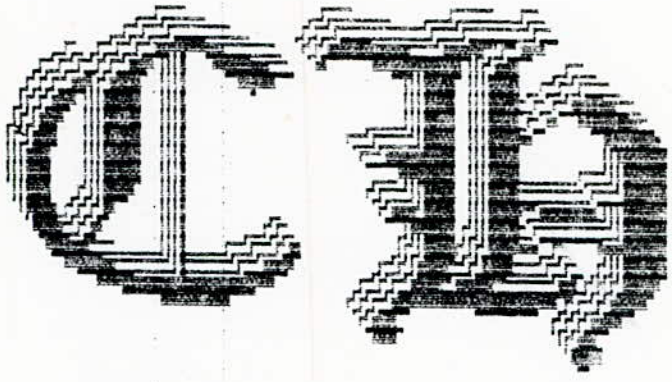


MODE DEUX "2": Entre en vigueur avec l'execution de IM2 tres puissant il permet de trouver directement l'adresse de la routine de traitement de l'interruption, grace a un vecteur d'interruption Le vecteur d'interruption est une adresse fournie par le peripherique generateur de l'interruption .Elle est utilisee comme pointeur sur l'adresse du debut de la routine de traitement de l'interruption.Le mecanisme d'adressage fourni par le mode d'interruption 2 est indirect.

Le peripherique interrompant fournie une adresse sur 7 bits qui est mise a droite des 8 bits du registre I . Un "0" est ensuite ajoute' a droite de cet ensemble de 15 bits fournissant ainsi une adresse sur 16 bits qui pointe quelque part en memoire Chacune d'entre elles est un double mot, qui est lui meme l'adresse du debut d'une routine de traitement d'intrruption.



2me PARTIE



L'INTERFACE PERIPHERIQUE PROGRAMMABLE INTEL 8255A

INTRODUCTION

L'INTEL 8255A est un boitier d'entree-sortie parallele tres simple qui ne possede que quatre registres sans particularites.dans le cas de notre KIT il gere le clavier, l'afficheur, le buzzer et les deux LEDS (rouge.et verte). Pour plus de detaille se referer au manuel du KIT dans les deux langues; neanmoins on va le presenter avec quelques schemas explicatives pour plus de detaille.Commencons avec la fig(1) du brochage du 8255A. Le boitier comprend une broche de selection "CS",deux broches d'adresses (A0 et A1), avec trois broches de commande de lecture "RD*",d'ecriture, (WR*) et de (RESET),et huit broches de donnees bidirectionnelles (D0 a D7).

CARACTERISTIQUES DU 8255A

Il possede quatre registres, puisqu'il possede deux entrees adresse. Possede un bus de donnees de huit bits.
Certaines combinaisons de niveaux sur les broches du 8255A donne au uP la possibilite d'ecrire et de lire dans les registres internes du boitier. Le 8255A possede trois modes de fonctionnement selectionnes avec l'octet placee dans le registre de commande, ces modes sont les suivants:

MODE 0 : Entree et sortie de base ou E/S de bits.
MODE 1 : Entree/sortie echantillonnee.
MODE 2 : Bus bidirectionnel.

Les ports A et B peuvent etre mis a des modes differents.Mais le port C est configure comme le port A pour son quartet de poids fort, et configure comme le port B pour son quartet de poids faible. On peut voir les different modes sur le schema explicatif, fig(4).

On peut definir tel ou tel mode avec une programation adequate du registre de commande du 8255A. Ainsi la figure (2) montre les differentes possibilites du mot de commande du registre de commande. Le bit 7 doit etre toujours a 1 pour annoncer que c'est un mot de commande, les bit 5 et 6 commandent le mode du port A;le bit 4 determine le sens des donnees dans ce meme port c'et a dire qu'il configure le port A en entree ou en sortie. Le bit 3 determine le sens des broches du quartet de poids fort port C. Le bit 2 selectionne le mode du port B, sachant que ce port ne peut etre configure qu'en mode deux,c-a-d bus bidirectionnel. Le bit 1 determine le sens des donnees pour les broches du port B et le bit 0 pour celles du quartet de poids faible du port C. De plus on peut programmer le port C d'une autre facon a travers le mot de commande comme le montre la figure (3), ou les broches du port C sont directement mises a UN ou a ZERO. Le bit 7 doit etre a ZERO pour specifier ce type de configuration.Les bits 1 et 3 specifient quel bit du port C sera manipule et le bit 0 indique l'etat du bit en question. Les bits 1,3 et 6 sont indifferants.

REM:Ce resume a ete tire de l'ouvrage intetule:"manuel des interfaces"

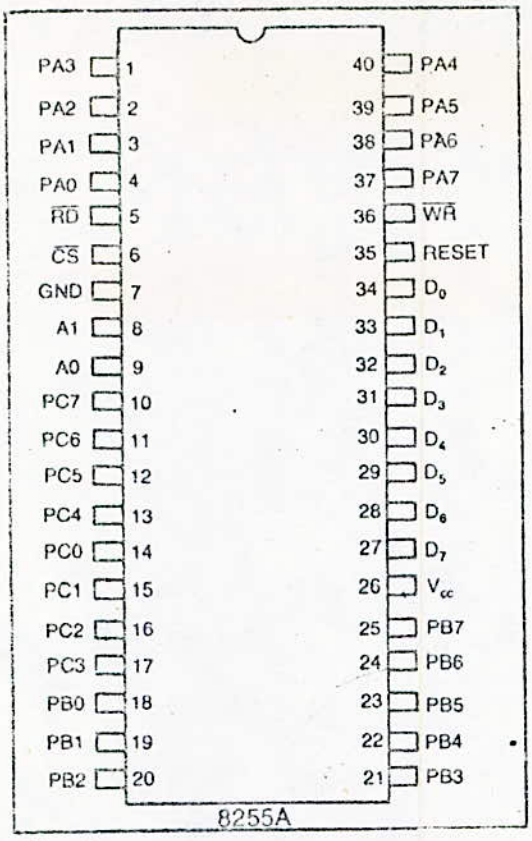


FIG 1 Schéma de brochage du 8255A.

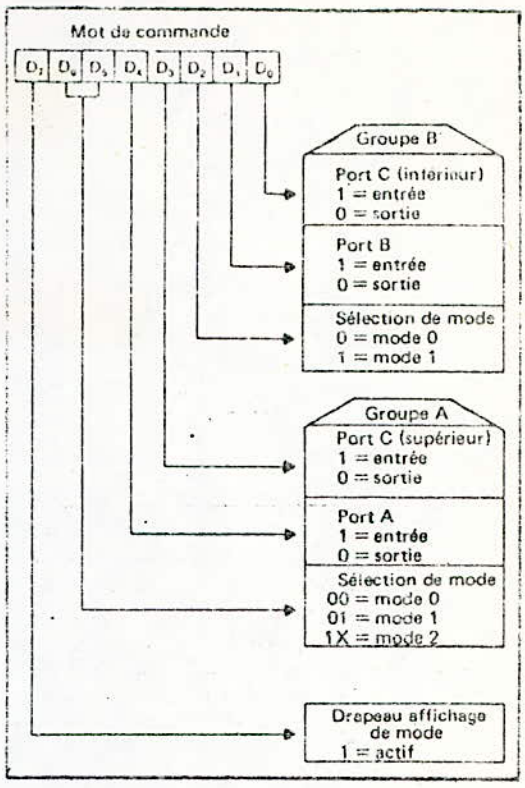
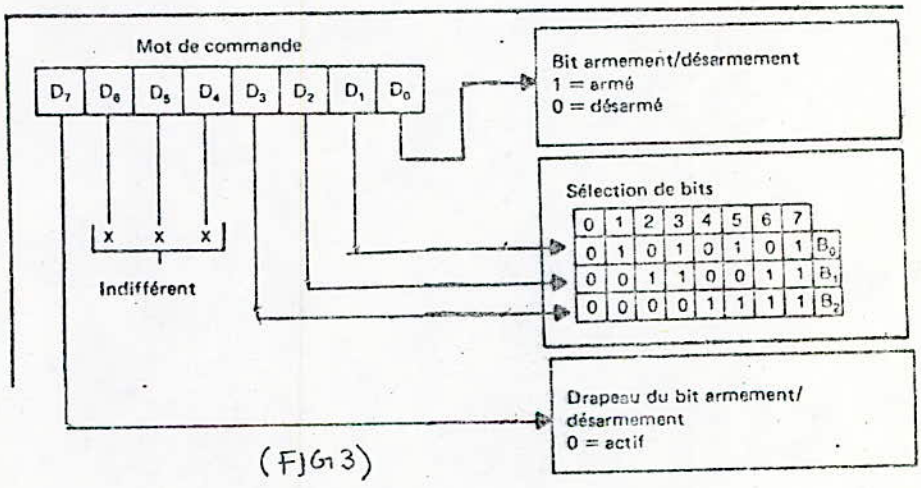


FIG 2. Définitions des bits du registre de commande de 8255A.



(FIG 3)

FIG 4 Configurations du mode 0 du 8255A

Bits	A D4	A D3	B D1	B D0	Groupe A Port A	Groupe A Port C (supérieur)	Groupe B Port B	Groupe B Port C (inférieur)
	0	0	0	0	Sortie	Sortie	Sortie	Sortie
	0	0	0	1	Sortie	Sortie	Sortie	Entrée
	0	0	1	0	Sortie	Sortie	Entrée	Sortie
	0	0	1	1	Sortie	Sortie	Entrée	Entrée
	0	1	0	0	Sortie	Entrée	Sortie	Sortie
	0	1	0	1	Sortie	Entrée	Sortie	Entrée
	0	1	1	0	Sortie	Entrée	Entrée	Sortie
	0	1	1	1	Sortie	Entrée	Entrée	Entrée
	1	0	0	0	Entrée	Sortie	Sortie	Sortie
	1	0	0	1	Entrée	Sortie	Sortie	Entrée
	1	0	1	0	Entrée	Sortie	Entrée	Sortie
	1	0	1	1	Entrée	Sortie	Entrée	Entrée
	1	1	0	0	Entrée	Entrée	Sortie	Sortie
	1	1	0	1	Entrée	Entrée	Sortie	Entrée
	1	1	1	0	Entrée	Entrée	Entrée	Sortie
	1	1	1	1	Entrée	Entrée	Entrée	Entrée

Note : Les bits D4, D3, D1 et D0 sont des bits du registre de commande du 8255A.

PERIPHERIE D'ENTREE-SORTIE

PIO (Programmable Input Output) :

** Il n'existe pas en realite de PIO standard,mais tous se ressemblent plus ou moins.La fonction d'PIO est de fournir plusieurs ports pour la connection des peripheries d'entree-sortie.
Chaque PIO possede ,au moins deux ensembles de 8 lignes ,et requiert un "tampon" de donnees (DATA BUFFER) pourstabiliser le contenu du bus de donnees au moins en sortie.Il sera donc equipe d'au moins un tampon pour chaque port.
*Le microprocesseur utilise des interruptions ,pour communiquer avec un peripherique d'entree -sortie (E-S).
Le PIO en fait de meme,et doit donc posseder au moins deux lignes de controle par port pour etablir le dialogue avec le peripherique qui lui est connecte.Le microprocesseur devra pouvoir lire l'etat de chaque port.
*Chaque port comprendra donc un ou plusieurs bits d'etats.
Chaque PIO comportera un certain nombres d'options,suivant l'utilisation qui sera faite de ces rssources.
*Le programmeur devra un acces au registre special du PIO,afin de lui preciser les options qu'il choisit.
*C'est le registre de commande (control registr:CR),dans certains cas les informations d'etat font partie du registre de commande.
*Une propriete essentielle du PIO est que chaque ligne doit pouvoir etre utilisee soit en entree soit en sortie <).
*Le programmeur a la possibilite de preciser les lignes qu'il desire utiliser respectivement en entree et en sortie.
*Pour programmer la direction des lignes 'chaque port possede un registre de direction de donnees (DDR ,data direction register) .Un "1" dans un bits du registre de direction de donnees indique que la ligne correspondante est en entree.Un "0 indique une sortie INPUT et 1,OUTPUT et 0.Lorsque le systeme est mis sous tension,il est tres important que toutes les lignes soient configurees en entrees,si en effet ,le uP est connecte a un peripherique dangreux il risquerait,dans le cas contraire,de l'activer accidentelement.Apres l'initialisation (Reset),tous les registres sont normalement mis a 1,et par consequent toutes les lignes en entrees.Le PIO se connecte naturellement au bus de donees (8bits),au bus d'adresses et au bus de controle.
*Il faut simplement se charge de preciser l'adresse des registres du PIO auquel il veut avoir accees.

LE REGISTRE INTERNE DE COMMANDE

** Le registre de commande du PIO permet de choisir differentes options de fonctionnement,de generer ou tester interruptions et de fournir un protocole d'echange automatique chaque fois que le systeme est initialise, le programmeur doit charger dans le registre de commande les valeurs correspondant a l'utillisation souhaitee du PIO.

LE PIO DU Z80:

Comporte deux ports, son brochage son brochage est represnte a la figure (---) avec son organisation interne.
Chaque port du PIO contient 6 registres:

-Un registre d'entree 8 bits .
- # # de sortie de 8 bits.
- # # de choix de mode de fonctionnemnet de 2 bits.
- # # masque de 8 bits.
- # # de direction de donnees de 8 bits.
- # # de 2 bits qui controle le fonctionnement de du registre masque.

Le PIO peut fonctionner dans l'un des 4 modes selectionnables a l'aide du registre de choix de mode.Qui sont les suivant:

sortie octet,entree octet,bidirectionnel et bit.

Les 8 bits de selection de direction des lignes fait travailler chaque ligne en entree ou en sortie,lorsqu'on travail dans ce mode.

BROCHAGE DU PIO:

Le PIO du Z80 a 40 broches l'alimentation est de +5 V et la masse.comment on peut connecter le PIO au microprocesseur Z80 (voir schema).

D7-D0: Ce sont les d'entree et la sortie de donnees.

B/A Select (selection B/A) cette ligne d'entree destincte determine si les donnees du bus de donnees sont en communication avec le port B (1 logique) ou avec le port A (0 logique).Generalement connectee au bit Ao.

C/D Select (selection commande/donnees)La valeur logique de cette entree indique au PIO si si les donnees en cours d'ecriture dans le PIO constituent reellement des donnees ou bien un mot de commande. Les mots de commande servent a programmer le PIO dans les differents mode de fonctionnement

CE (chip eanable input,entree de validation de boitier)chque fois que cette ligne est au 0 logique actif,les donnees peuvent etre ecrites ou lues dans le dispositif d'E/S.L'entree CE est une adresse decode apartir des 8 bits de poids faible du bus d'adreese du systeme.

CLOCK (horloge):une entree d'orloge monophasee pour synchroniser certaines operations internes.

M1 C'est la sortie M1 du Z80.Le PIO utilise cette ligne d'entree pour commander plusieur fonctionnement internes.Quand M1 et RD sont tous les deux actifs le Z80 et entrain de chercher un code operation Il faut que le PIO reconnaitre cet etat,le parce que le dispositif reconnaitra automatiquement un certain code operation.C'est l'instruction RETI (retour d'interruption) Le signal M1 va aussi effectuer:

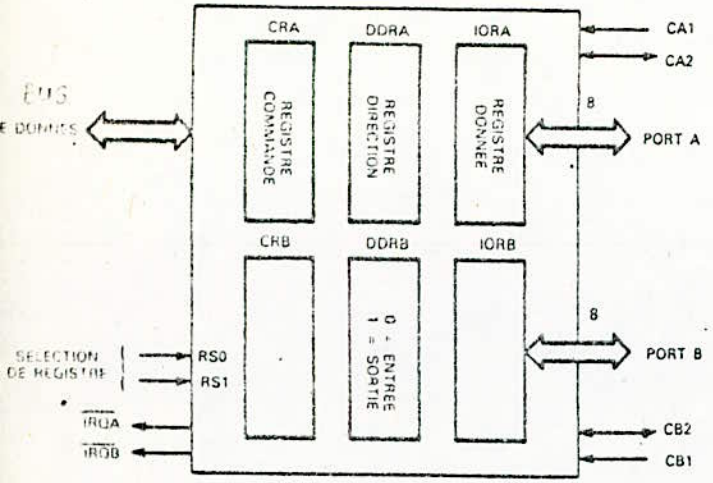
- *Il va synchroniser la logique d'interruption.
- *Quand M1 a lieu sans RD ou IORQ actifs le PIO est mis en etat de reinitialisation interne,

IORQ (Input out put request,demande d'entree sortie)C'est une entree du PIO qui se connect directement a la sortie IORQ du uP Z80 ce signal est employe avec B/A select, C/D select et CE pour pour transferer commandes et donnees entre Z80 et PIO quand l'entree IORQ et CE sont toutes deux actives (0 logique),le PIO communique electriquement avec Z80.

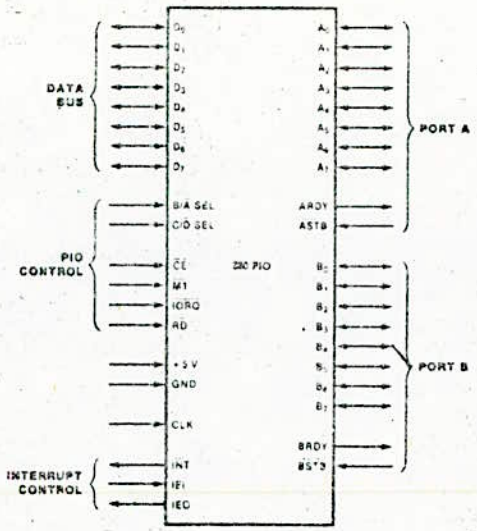
RD (read,lecture) Cette entre est connecte directement a la ligne d'entree du Z80 .Quand l'entree IORQ du PIO est active de meme pour les entrees RD et CE le sont aussi,le Z80 lit les donnees dans le PIO.Le PIO n'a pas d'entree WR (Writ) si IORQ et CE sont tous deux actifs mais que RD n'est pas actif le Z80 doit ecrire dans le PIO.

IEI et IEO Ce sont les lignes Interrupt Enable Input et output(entree et sortie d'autorisation)On les utilise dans le chainage de priorite d'interruption du Z80

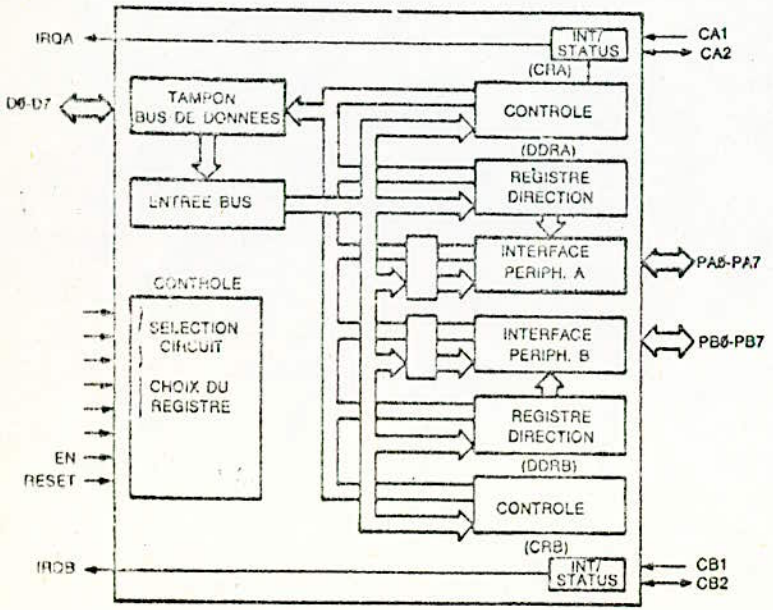
INT Cette sortie est une connexion a drain ouvert, active chaque foisque le PIO a ete programme pour des onterruptions et en train de demander activementune interruption au uP Z80.



. — PIO typique

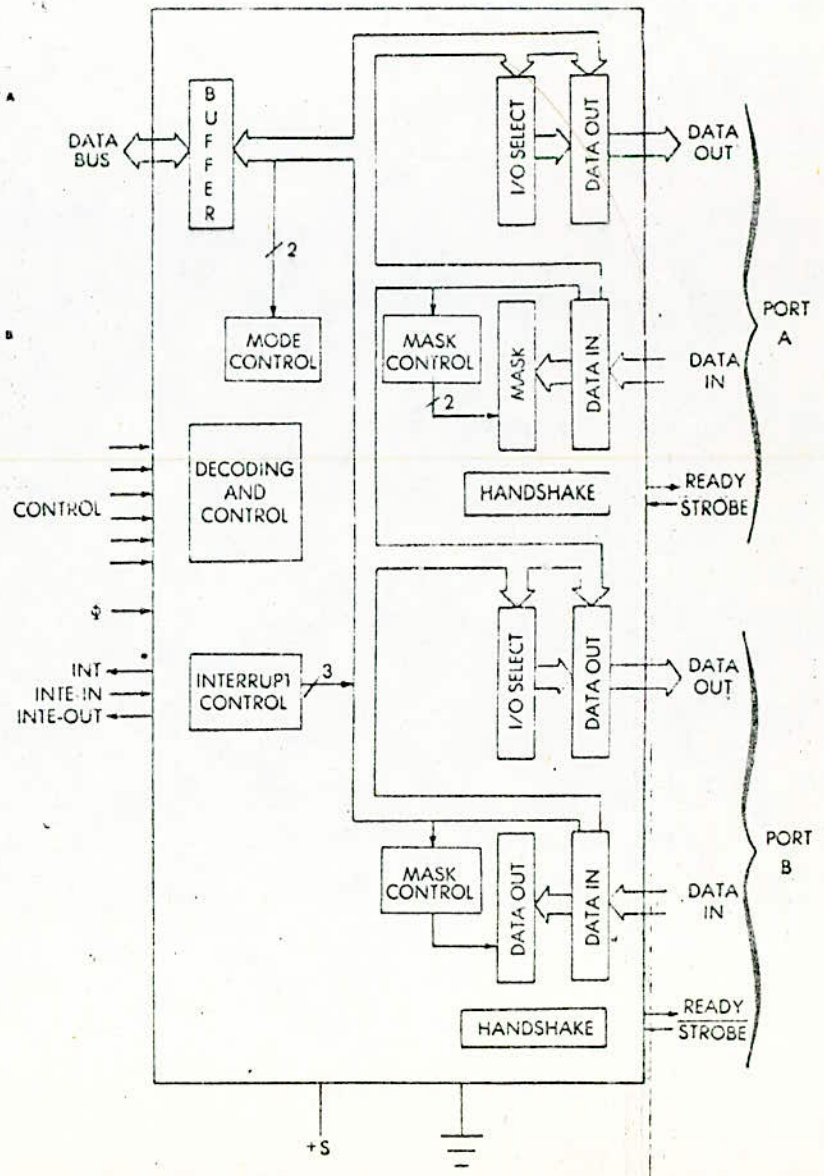


Logic Functions



— PIO : Lecture d'un octet de données

Pin Configuration



— Schéma fonctionnel du Z80 PIO

Ao-A7 Ce sont les ligne d'entree-sortie du port A. C'est par ces lignes que le PIO communique avec un peripherique.Ao est le bit de poids faible de ce bus d'E/S.

ASTB Cette entree recoit les singnaux d'un peripherique pour effec- une procedure de handsheking dans le registre du port A.

ARDY (register A Ready,registre A pret) La fonction de cette sortie depond du mode de fonctionnement du PIO.

Bo-B7 Ce sont les lignes d'E/S qui relient le port B au peripherique.

BSTB (port B strobe pulse input,entree d'impulsion d'echantillonnage du port B)La fonction de cette ligne est la meme que pour le port ASTB a l'exeption du point suivant: Quand le port A est programme en mode bidirectionnel,le signal va echantiollonner les donnees provenant du peripherique dans l'entree du registre A ceci implique que le port B n'a pas de capacite bidirectionnel.

BRDY (registre B ready,registre B pret)C'est une sortie active au 1 logique.

****

CONNEXION:

La ligne de selection B/A du PIO du Z80 est connecte a la ligne de sortie d'adresse Ao du Z80.La ligne C/D est connectree a la ligne de sortie d'adresse A1 du Z80:

A1	Ao	Port	Selection
0	0	A	Donnees
0	1	B	Donnees
1	0	A	Commande
1	1	B	Commande

INITIALISATION DU PIO:

Processus de mise a l'etat initial:

*1)Les registres masques des deux ports sont remis a zero pour inhiber tous les bits de donnees des ports.

*2)Les lignes de bus de donnees de ports sont mises en etat de haute impedance;les lignes de handshaking READY(pret)sont innactifs Le mode 1 (entree) .

*3)Les registres d'adressages de vecteurs d'interruptions ne sont pas remis a l'etat initial.

*4)Les bascules d'autorisation d'interruption des deux port sont remises a l'etat initial ceci interdit au PIO toute reponse a une interruption jusqu'a ce qu'il soit programme.

*5)Les registres de sortie des deux ports sont remit a l'etat initial

PROGRAMMATION DE PIO

Ce mode permet d'employer les ports en sortie seulement on peut programmer les deux ports independamment.Par exemple on programme le port A seulement Le premier registre programme pour etablir le mode c'est le registre de commande. Ao=0 pour le port A et C/D=1 pour la commande l'adresse de port 2E sera celle du registre de commande du port A. *Voici un programme de Z80 qui valide le port A en sortie,puis ecrit un compte binaire dans le port de sortie:

```
(**) 1800   3E          LD A,0FH
     1801   0F
     1802   D3          OUT(2EH),A    ;Sortir le mot de commande
     1803   2E
        * Port A=Sortie,B/A=0,C/D=1
        * Ecrire 53H dans le port de sortie A
     1804   3E          LD A,53H
     1805   53
     1806   D3          OUT(2CH),A    ;sortire les donnees
     1807   2C
        * Programmerle port B en port de sortie
     1808   3E          LD A,0FH
     1809   0F
     180A   D3          OUT(2FH),A    ;sortire le mot de commande
     180B   2F
        * Port B=sortie,B/A=1,C/D=1
```

*Le port B peut alors etre employer en port generale de sortie en ecrivant les donnees de sortie en adresse de port 2DH.

++++++++++++++++

MODE DE PROGRAMMATION:

Dans ce mode le PIO va etre mis en port d'entree.Le mot de commande qui valide le PIO dans ce mode est 4F.Ce mot pourrait etre ecrit dans le port 2F ou 2E .En programmant le port B en port d'entree en utilisant le port 2F. *Voici le programme qui va initialiser le PIO en mettant le port A en sortie et le port B en entree,permettant ainsi au programme de lire les donnees dans le port A et d'en ecrire dans le port B:

```
(**) 1800   3E          LD A,0FH      ;A reg=mot de commande
     1801   0F
     1802   D3          OUT(2EH),A    ;ecrire mot de commande dans PIO
     1803   2E
       *Le port A est maintenant un port de sortie
     1804   3E          LD A,4FH      ;A reg= mot de commande
     1805   4F
     1806   D3          OUT(2FH)      ;Ecrire mot de commande dans PIO
     1807   2F
       *Le port B est maintenant un port d'entree
     1808   DB          IN A,(2DH)    ;lire donnee dans portB
     1809   2D
     180A   D3          OUT (2CH),A   ;sortie de donnees dans port A
     180B   2C
```

****

MOI DE COMMANDE D'INIERRUPTION:

On peut programmer et utiliser les interruptions dans le cas du mode 1et 2 Pour etablir le mot de commande d'interruption il faut ecrire dans le registre de commande d'un port particulier les 4 bits de poids faible D3 D0 sont mit a 0111.Ce qui indique que les 4 bits de poids fort vont definir le mot de commande d'interruption voir figure ( ).Si le bit D7 du mot de commande d'INT est mit au 1 logique la bascule d'autorisation du d'INT du PIO est mise a l'etat initial et le dispositif peut produire des INT .Si le bit est efface ou mis a 0 les INT sont interdites.
Si le bit D4 du mot de commande d'INT est mis au 1 logique toute INT en attenteva etre mise a 0 dans le PIO.
Les bits D6,D5 et D4 ne servent qu'en mode 3 .Pour les interruptions en modes 0 et 1 il suffit d'autoriser ou d'interdire les circuits d'INT du PIO. La bscule d'autorisation d'INT peut etre validee ou inhibee dans le registre de commande par les mots: AUTORISATION D'INT ---->87H
INHIBITION D'INT ------>07H
On peut aussi autoriser ou inhiber sans modifier le REG de commande par 83H pour autoriser,03H pour inhiber .
Si les interruptions sont autorisees le vecteur d'intrruption doit etre charge dans le bus de donnees et employe par le Z80 en mode d'INT 2.
Voici un programme qui autorise les INTs du port

PIO EN MODE BIDIRECTIONNEL (MODE 2)

Le mode de fonctionnement bidirectionnel est combinaison des modes 1 et 2 Il utilise les 4 lignes de handshaking du PIO le mode d'E/S n'exist pas sur le port A, le port B doit etre misten mode de commende.Le vecteur d'INT de sortie doit etre programme dans le port A, et le vecteur d'INT d'entree doit etre programme dans le port B.
Voici le programme pour etablir le PIO en mode 2.Les routines de service d'INT sont empolyees pour entree et mettre des donnees avec le PIO:

```
        *Ce programme va etablir le port A en port d'E/S

(**) 1800   3E,8F      LD A,8F        ;CEST LE MOT DE MEMOIRE
     1802   D3,2E      OUT(2EH),A     ;ECRIRE MOT DE MODE DANS P/A
     1804   3E,00      LD A,00        ;VECTEUR D'INTERRUPT POUR P/A
     1806   D3,2E      OUT(2EH)       ;ECRIRE VECTEUR DANS P/A

     1808   3E,CF      LD A,CF        ;MOT DE MODE POUR POUR PORT/A
     180A   D3,2F      OUT(2FH)       ;ECRIRE MOT DE MODE DANS PORT/B

        *Mot de mode etablit port B en mode 3 les definitions de bits
         pour le port B suivant
(**) 180C   3E,FF      LD A,FFH       ;TOUS LES BITS SONT DES ENTREES
     180E   D3,2E      OUT(2FH),A                ;ECRIRE DANS PORT/B
     1810   3E,17      LD A,17H       ;MOT DE COMMANDE D'INT DU PORT/B
     1812   D3,2E      OUT(2EH),A          ;INHIBER INT ,MASQUE SUIT
     1814   3E,FF      LD A,FFH               ;TOUS LES BITS INHIBES
     1816   D3,2F      OUT(2FH),A         ;ECRIRE MASQUE DANS PORT/B
     1818   3E,02      LD A,02H                 ; VECTEUR INT PORT/B
     181A   D3,2F      OUT(2FH)                 ; ECRIRE DANS PORT/B
```

```
181C   3E,3A      LD A,3AH          ;ETABLIR OCTET HAUT DE TABLE
181E   ED,47      LD I,A                        DES VECTEURS

   *Table de vecteur en 3A00 pour le port A
                .   et en 3A02   #      #   B
   *Adresse des routines de service en FC81 pour le port A
                                      et FD00   #      #   B
L'INT du port A est pour emettre des donnees
L'INT du port B est pour entrer  des donnees

(* *) 1820   ED,5E      IM 2                  ;ETABLIR MODE 2 D'INT
    1822   3E,83      LD A,83H               ;VALIDER INT,SUR PIO
    1824   D3,2E      OUT(2EH)                    ;VALIDER PORT A
    1826   D3,2F      OUT(2F),A                   ;VALIDER PORT B
    1828   FB         EI                     ;VALIDER INT DU Z80

    1829   C3,29,18   JP BOUCLE          ;ATTENDRE INTERRUPTIONS
                  *************************
```

MODE 3 DU PIO:

Le fonctionnement du PIO en mode 3 est prevu en configuration sans handshaking .les huits bits de chaque port peuvent se voir attribuer individuellement une fonction d'entree ou de sortie dans n'importe quel ordre .Les bits Bo,B4 et B5 peuvent,par exemple,etre des donnees et tous les autres bits de donnees du port B des sorties.Pour selectionner le mode 3 ,les donnees ecrites dans le registre de commande sont 0CFH.Le mot suivant immediatement ce mot de selection de mode va definir l'emploi des 8 bits de port.Un "1" va mettre la ligne de donnee en entree.Un "0" en sortie.Les instructions suivantes vont mettre le PIO en mode 3:

```
(* *)  LD A,CFH                  Charger CF dans le registre A
     OUT (2FH),A    Selectionner le mode 3 pour le registre B
     LD A,49H                          Bits Do,D3 et D6=1
     OUT (2FH),A               Mettre Bo,B3 et B6 en entrees
```

Quand on lit des donnees dans le port d'entree,on lit la valeur logique des bits d'entree programmes.
Ce mode peut fonctionner avec le PIO comme suit :soit l'exemple

1) On met d'abord le port B en mode 3.
2) On programme ensuite en entrees les lignes de donnees B2, B3 et B4 du port B.
3) on programme ensuite en sorties de donnees B0,B1,B5,B6 B7 du port B
4) On surveille ensuite une interruption dans les lignes de donnees B3,et B4 du port B.On masque toutes les autres lignes du port B
5) L'etat actif de B3 et B4 sera un 0 logique.
6) Un mot se produira quand les deux bits seront au logique actif.C'est la fonction ET des etats actifs.
7) Le vecteur d'interruption sera 08.

Pour cela on suppose que le PIO est defini avec les ports d'E/S 2C,2D,2E et 2F soit le programme qui permet cette configuration:

```
LD A,CFH
OUT (2FH)     Etablir mode 3 dans le PIO pour le port B
LD A,1CH
OUT (2FH),A   Mettre bits B2,B3,B4 en entrees
LD A,08H
OUT  (2FH),A Vecteur d'INT=08
LD A,D7H
OUT (2FH),A    Autorisation INT,bas ,masque suit
LD A,E7h
OUT (2FH),A    Les bits B3,B4 ne sont pas masques
```

Une caracteristique seduisante du PIO est qu'il surveille le bus de donnees du Z80 quand une instruction RETI est cherchee en memoire,il supprime automatiquement l'INT.
Quand le PIO est mis en configuration initiale une demande d'INT exterieure indesirable pour inhiber eliminer sa, il faut inhiber les entrees d'INT provenant du Z80 .
On peut la faire avec les instructions suivantes :

```
LD A,03
DI                              Inhiber INT du Z80
OUT (2EH),A          Inhiber INT sur port A du PIO
OUT (2FH),A          Inhiber INT sur port B du PIO
EI                             Valider INT sur Z80
```

Examinons les 3 lignes de commande d'INT du PIO: $\overline{INT}$,IEO,IEI.La ligne $\overline{INT}$ est une sortie elle est relie a la ligne d'entree INT du Z80 .IEI est la ligne d'autorisation d'intrruption.
On utilisant ces lignes (IEO et IEI)on peut chainer jusqu'a quatre PIO en configuration de priorite d'INT,

**REM : Pour que nos programmes "marche" Sur Le KIT, Il faut changer les adresses comma suit :

PIO A DATA	2C	80
PIO B DATA	2D	81
PIO A CONTROL	2E	82
PIO B CONTROL	2F	83

↑ Utiliser par notre Kit.

COUNTER TIMER CIRCUIT (CTC)

*Le CTC du Z80 est un boitier LSI specialement concu pour etre utilise avec le Z80,il est d'un emploi tres souple.

SCHEMA DU CTC

Ce circuit tire son nom de deux fonctions:comptage est temporisation il quatre cannaux de comptage-decomptage independants nommes Ch0,Ch1,Ch2 et Ch3,connectes a trois cannaux (0,1 et 2),deux lignes de signaux appelees compte zero/fin du temps (sortie) et declenchement d'horloge (entree) s'interfacent avec un perifirique,le canal 3 n'a que la ligne d'entree de declenchement ceci est du au nombre limite de broches du boitier qui sont de 24 broches.

CONSTITUTION

*La logique de commande interne : Assure la synchronisation appropriee de tous les transfert de donnees a l'interieur du circuit.
*La logique d'interruption : C'est un dispositif tres puissant,il sera detaille plus loin.
*L'entree-sortie du CPU: Cette logique donne l'interface entre Z80 & CTC il y a huit lignes de donnees et six lignes de commandes connectees a ce bloc,c'est par ce bloc que le Z80 communique electriquement avec le CTC,de sorte qu'il peut etre programme pour agir exactement comme le veut le le programmeur.

EXAMEN DETAILLE DU BLOC DE CANAUX

La figure (1) montre le schema d'un canal,on peut voir qu'il comporte un registre de commande,un de constante de temps et un decompteur Le programme ecrit des informations dans le registre de commande ,il definit le foctionnemt du canal.Le registre constante de temps (8bits),ce nombre sert a initialiser le bloc decompteur.Ce dernier est a 8 bits,sa sortie est referencee compte zero/fin de temps,quand il atteint zero,sa sortie devient active d'une facon deja programme.Le compteur de predetermination (8bits) predetermine ou devise l'entree d'horloge du decompteur par 16 ou 256,suivant la programmation du boitier.Une autre entree du decompteur nommee horloge externe/declenchement de temporisateur cette ligne peut etre une entree d'horloge directe du decompteur a 8 bits ou elle peut servir de ligne de validation a l'horloge de sortie du prediviseur,la fonction de cette ligne depend de la programation.

BROCHAGE DU CTC-Z80

La figure (2) montre le brochage et les signaux d'entrees et de sorties.

LE CTC EN MODE COMPTEUR

PROGRAMATION DU REGISIRE DE COMMANDE DE CANAL

Examinons les bits de donnees de programmation du registre de canal

D0 : Il est a un (1) indique que le mot du registre de commande du canal.

D1 :Si ce bit est a UN le canal cesse de compterou de mesurer le temps aucun bits du registre de canal ne sera change,puis il reprend lorsqu' une constante de temps est chargee.

D2 :Chaque fois qu'une constante de temps doit etre chargee,l faut ecrire deux mots dans le CTC le premier avec cette configuration de bit est le second,la valeur de la constante.Ainsi un UN (1) dans ce bit informe que le mot suivant est une constante de temps.

D3 :Si c'est un UN (1) ie;le declencheur exterieur demmare le temporisateur,si c'est un zero (0) ie;le temporisateur demmare des que la constante de temps est chargee,ce bit sert pour le mode temporisa-teur seulement.

D4 : Ce bit determine le front actif de l'entree:Declenchement/horloge externe,ainsi nous avons dans les deux modes:

En temporisateur: D4=1 => Declanchement par front montant.
D4=0 => " " " " " " " descendant.

En compteur : D4=1 => Front montant de decrementation de l'horloge externe.
D4=0 => Front descendant de decrementation de l'horloge externe.

D5 :Il choisit la valeur de predivition,un UN (1) donne une division par 256 et un zero (0) donne une devision par 16.Ce bit ne sert qu'en mode temporisateur.

D6 :C'est le bit de selection,UN (1) signifie que le canal fonctionne en compteur,dans le cas contraire il fonctionne en temporisateur.

D7 :Un UN (1) signifie que l'iterruption est validee,le vecteur d'interruption est ecrit dans le Z80 la valeur zero est atteinte par le decompteur,de plus le vecteur d'interruption doit etre charge avant que le canal ne fonctionne .un ZERO (0) dans ce bit inhibe les inter-ruptions interne.

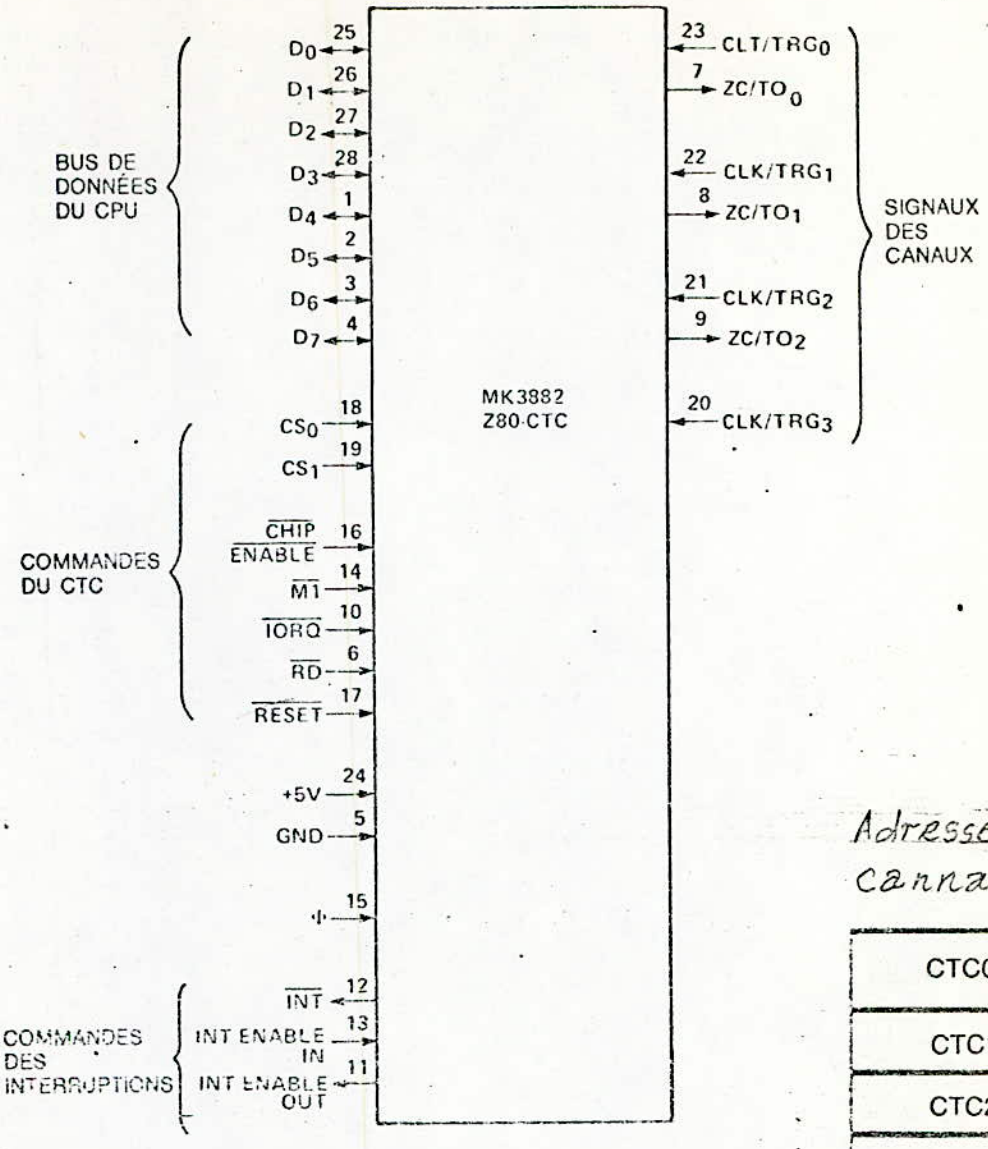
PROGRAMMATION DU REGISIRE DE CONSTANTE DE TEMPS

Le canal ne peut fonctionner avant que le registre de constante de temps n'ait ete charge.Il faut que le bit D2 du mot de commande soit a UN apres ce mot on charge la constante de temps,si le registre de la constante est charge pendant le comptage,la nouvelle valeur n'est utilisee qu'apres que la valeur en cour soit etteinte.

PROGRAMMATION DU VECTEUR D'INTERRUPTION

Le CTC est prevu pour emploi avec le Z80 quand il fonctionne en mode 2 d'interruption,un seul vecteur d'interruption est charge pour le CTC. Le bit D0 est a "0":Il s'agit d'un vecteur d'interruption qui est charge dans lecanal 0 seulement.Les bits D1 et D2 sont determine par le canal qui a interrompu le uP.Les cinq bits de poids fort sont mis par le programmeur par exemple:

CANAL 0	0	1	0	1	1	0	0	0
CANAL 1	0	1	0	1	1	0	1	0
CANAL 2	0	1	0	1	1	1	0	0
CANAL 3	0	1	0	1	1	1	1	0

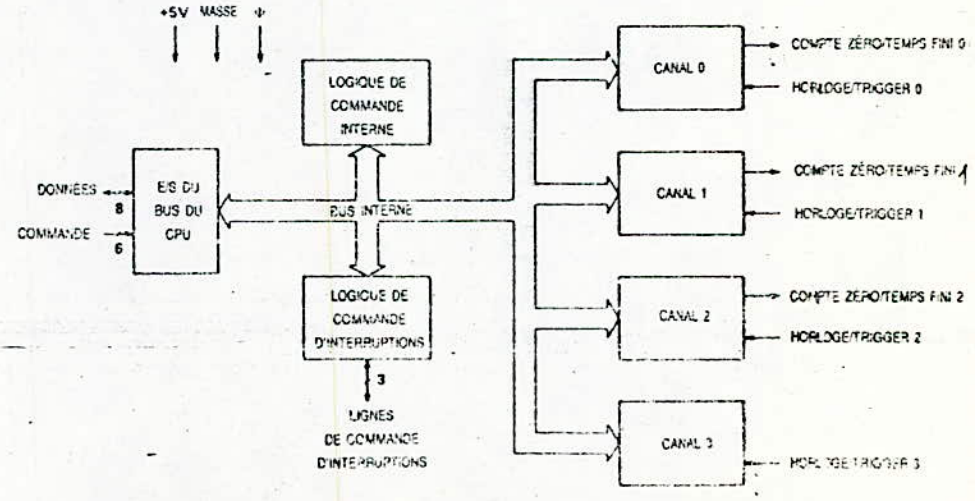


Adresses des cannaux .C.T.C

CTC0	40
CTC1	41
CTC2	42
CTC3	43

: Brochage et définition des signaux du CTC du Z80.

SCHÉMA DU CTC DU Z80



**Il est possible qu'un canal interrompe le CTC .Il y a une priorite interne,predeterminee dans le CTC .Le canal 0 a la priorite la plus elevee et le canal 3 a la plus basse.

PROGRAMMATION DU CTC EN COMPTEUR

***Pour etudier l'utilisation du CTC en mode compteur,on commance par un systeme sans interruption.prenons l'exemple suivant:

*Le systeme compte les impulsions,le front montant est actif a l'entree d'horloge exterieure du canal

*On utilise le canal 1 du CTC.

*On utilise le CTC en mode de scrutation,quand le compte atteint 46D,le uP effectue une action donne .

Il faut definir le mode de fonctionnement en programmmant le registre de commande comme suit:

BIT 7=0 : Inhibe les interruptions.
BIT 6=1 : Mode compteur selectionne.
BIT 5=0 : Sans importance car on l'utilise dans le cas du temporisateur.
BIT 4=1 : Le front montant de l'horloge externe va compter .
BIT 3=0 : Sans importance,concerne le temporisateur.
BIT 2=1 : Les donnees de constante de temps vont suivre.
BIT 1=1 : Reinitialiser le canal .
BIT 0=1 : C'est un mot de commande.

Ce qui se traduit par :

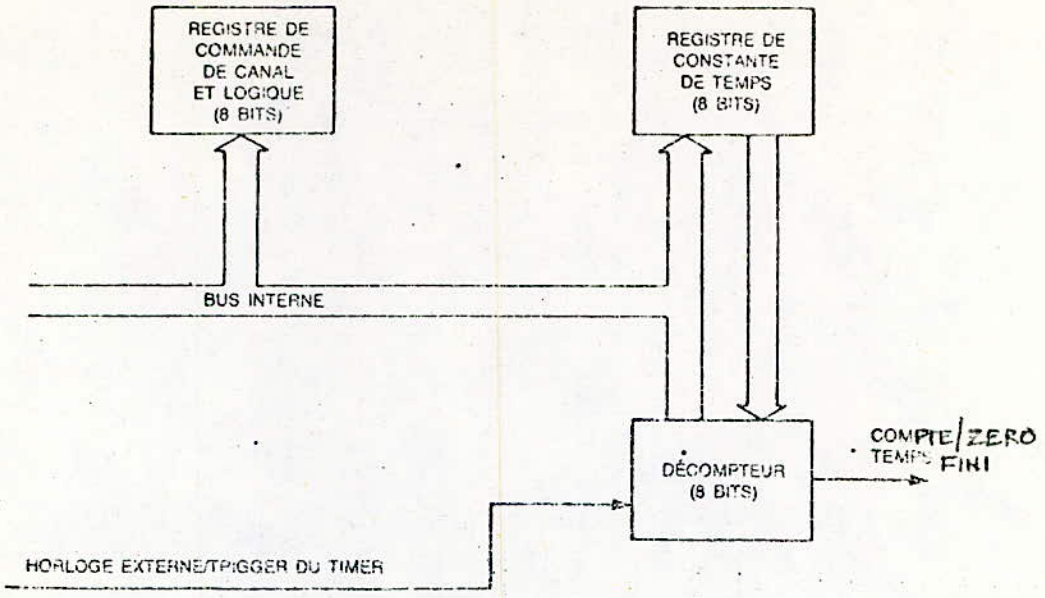
```
LD   A ,57H
OUT (Ch1),A
```

Le CTC s'attend a ce que le prochain mot ecrit dans le port (Ch1) soit la constante de temps,car le bit D2 a ete mit a UN (1).Le nombre 46D=2EH,alors on ecrit la constante de temps comme suit:

```
LD   A ,2EH
OUT (Ch1),A
```

Aussitot que le registre de constante de temps est envoyee,le canal commance a fonctionner.Ainsi pendant que le Z80 est en train d'effectuer son action,le CTC recharge automatiquement sa constante de temps et recommence a compter.voici a titre d'exemple un programme qui va surveiller en mode de scrutation:

```
        LD  A,57H
        OUT (Ch1),A     Mot de commande
        LD  A,2EH
        OUT (Ch1),A     Constante de temps
INT:    IN  A,(Ch1)     Lire le compte
        CP  00H         Verifier le compte final
        JP  NZ,INT      S'il n'est pas nul continuer a interroger
                        jusqu'au nbre 46D,a ce moment le Z80 fera
                        autre chose.
```



CANAL - MODE TEMPORISATEUR

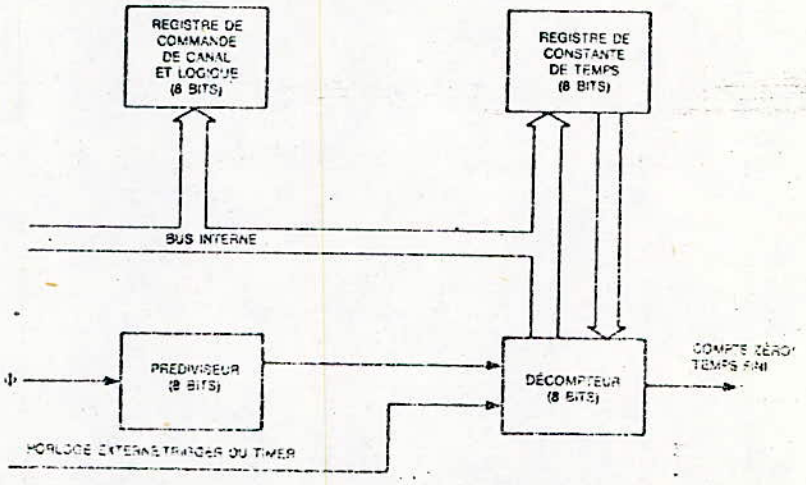
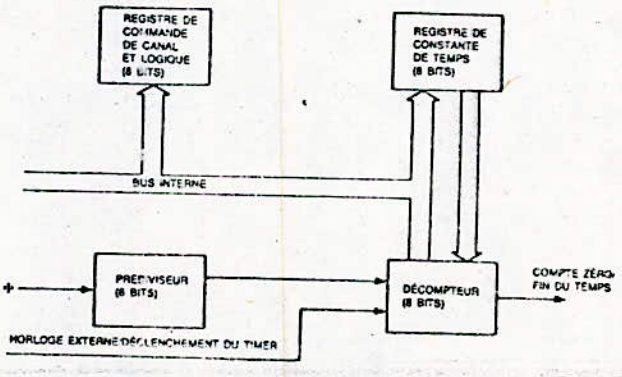


SCHÉMA DE CANAL



***Presantons maitenant le meme programme,avec des interruptions.Le Z80 entre dans une boucle d'attente d'interruption et ce apres confeguration du CTC,ainsi le mode de commande va changer pour indiquer l'utilisation ds interruptions.

```
            DI              Pour eviterles fausses interruptions
            IM  2           Etablir mode 2 d'interruption
            LD  A,80H       Dans le registre de commande
            LD  I,A         Charger l'octet de poids fort du tableau
                              d'interruption
            LD  A,D7H       Mot de commande pour le canal 1
            OUT (Ch1),A     Vers le canal 1
            LD  A,2EH       La constante de temps
            OUT (Ch1),A     vers le canal 1
            LD  A,31        Charger le vecteur d'intrruption
            OUT (Ch1),A
            EI              Revalider les interruptions
      ATT   JP  ATT         Attendre ici une interruption
```

*L'adresse de routine d'interruption va etre chargee dans l'emplacement 1FEE et 1FEF.

FONCTIONNEMENT DU CTC EN MODE TEMPORISATEUR

*Pour que le CTC fonctionne en mode temporisateur,il faut le configuration en sortie d'hologe donnant , par exemple, des impulsions a la frequence 2500 Hz,on prend comme frequence d'entree de l'horloge du systeme est egale a 4 MHz.

*Il faut calculer le nombre par lequel il faut diviser la frequence du signal d'entree pour obtenir une frequence de 2500 Hz.Ce est 16x100 donc le prediviseur du CTC doit etre mis soit a 16 ou 256.Donc la constante de temps sera 100D soit 64H ce qui donne un diviseur egal a 1600D.A la sortie nous aurons un (1)logique puis un zero (0)logique a une frequence egale a 2500.La programmation relative a cet exemple commence par etablir le mot de commande comme suit:

D7=0:Pas d'INT dans ce cas.
D6=0:Mode temporisateur.
D5=0:Facteur de predivision"16".
D4=0:Sans importance.
D3=0:Demarrage qd la cste est charge.
D2=1:La cste va suivre ce mot.
D1=1:Reinitialiser le canal .
D0=1:Il s'agit d'un mot de commande.

*Le programme sera comme suit:

```
            LD A,07H    Mot de commande
           OUT (Ch1),A  Vers le canal 1
            LD A,64H    Cste de temps
           OUT (Ch1),A  Vers le canal
```

*Le temporisateur fonctionne sur le canal 1 avec une frequence de 2500Hz.

COMMUNICATIONS SERIE

DEFINITIONS

Une communication serie consiste a transmettre des donnees sous forme d'un flot de bits,un bit a la fois.Cette transmission a lieu sous forme sequentielle dans le temps.
Un des points chaud de la communication serie est la frequence du flot de bits de donnees transmis c-a-d la vitesse de transfert:ie:Le nombre de bits transmis par seconde en serie dans une ligne unique.Les vitesses de transfert types sont:110,150,300,1200,2400,4800,9600.

CONVERSION DE DONNEES EN PARALLELE EN DONNEES EN SERIE

La communication serie doit recevoir les donnees du microprocesseur qui sont en parallele et les transformer en un flot de bits.Cette conversion passe par les etapes suivantes:
-Stocker le mot de 8 bits en parallele dans un registre a decalage.
-Decaler les 8 bits en sortie du registre,un bit a la fois a la vitesse de transfert appropriee.

BIT DE DEPART

Les donnees transmises en serie doivent etre recues et intrpretees electriquement,alors un bit de depart (start bit)est ajoute automatiquement au flot des bits de donnees.ayant pour role d'informer electriquement le circuit de reception qu'un nouveau flot de donnees arrive afin de lui permettre de synchroniser son horloge sur le flot de donnee d'entree.L'etat d'attente d'une ligne de transmission serie quand elle ne comporte aucune donnee est apelle "marquage".Si le marquage est le 1 logique ,le bit de depart sera a l'etat logique oppose a celui du marquage et vis-versa.Le circuit de reception va detecter ce bit et autoriser l'ensemble recepteur a saisir les nouvelles donnees.

BIT DE PARITE

Ce bit est ajoute par l'emetteur et utilise par le recepteur.Quand un mot est pret a etre transmis,il contient un certin nombre de 1 logiques,ce nombre peut etre,soit paire,soit impaire.Le circuit de reception est prevu pour recevoir les donnees et detecter un nombre de 1,soit pair,soit impair. Supposons que le circuit de reception soit prevu pour recevoir un nombre de 1 logique qui soit pair a chaque flot de bits serie.Le nombre 55H conviendra,part contre 55H ne l'est pas .Le circuit va donc ajouter un bit supplementaire au flot de donnees :Ce sera soit un 1 logique, soit un 0 logique,de facon que le nombre total de 1 soit un nombre pair.La parite est utilisee comme controle d'erreur au premier niveau des donnees transmises.

BIT D'ARRET (STOP BIT)

Le circuit de reception est prevu pour detecter un bit d'arret a la fin du flot de donnees.Les bits d'arret ont le meme niveau logique que le marquage de la ligne de transmission serie.

SERIAL OUTPUT INPUT (SIO)

Le dispositif d'entree-sortie serie appele SIO, possede plusieurs mode de fonctionnement differents et peut etre employe dans plusieurs applications

SCHEMA DU SIO

Le SIO comporte deux caneaux de communication serie qui sont independants. Chaque canal est associe a un bloc de commande et d'etatUII sont relies a plusieurs lignes d'entree et de sortie. Les registres de lecture/ecriture du canal A et du canal B, sont-les logiques internes qui permettent la programmation appropriee du SIO.
Tous les modes de fonctionnement du SIO existent sous controle du logiciel Des mots de commandes doivent etre envoyes a ces blocs pour configurer le composant.
La logique de commande d'interruptipon :Le SIO a une architecture d'interruption tres puissante.Ce bloc logique permet au SIO de s'interfacer directement a la structure d'interruption du Z80.
La logique de commande interne :Sert a lire et a ecrire des donnees dans le bus de donnees interne .Ce bloc synchronise de facon apropriee tous les transferts de donnees internes du SIO.
Le bus d'entree-sortie du CPU sert d'adaptation et d'interface electrique entre CPU du Z80 et SIO.

BROCHAGE DU SIO-Z80

La figure( ) montre la brochage et les signaux d'entrees et de sorties

REGISTRE D'ECRITURE

*Le SIO contient huit registres (WR0-WR7) par le canal;ces registres sont programmes separement par le programme du systeme pour mettre chaque canal dans sa configuration specifique.A l'exception de WR0,la programmation des registres d'ecriture necessite deux octets.Les trois bits du premier octet (D0-D2)pointent le registre choisi;le second octet est le mot de commande effectif ecrit dans le registre pour configurer le SIO.
*Le programmeur apres pointage du registre choisi,peut lire ou tester le registre de lecture,comme il peut ecrire pour initialiser le registre d'ecriture.En elaborant un programme pour initialiser le SIO de facon modulaire est structuree,il peut utiliser des blocs d'instructions d'entree-sortie.
*WR0 est un cas tres special;car on peut acceder a toutes les commandes avec un seul octet.Reset(interne ou externe)initialise les bits du pointeur(D0-D2) pour qu'il pointe vers WR0.

LES REGISTRES DE LECTURE

Le SIO contient trois registres RR0,RR1,RR2 qui peuvent etre lus pour obtenir l'information d'etat de chaque canal (excepte canal B de RR2).
*L'information d'etat comporte les conditions d'erreur,le vecteur d'interruption et les signaux standard de communications interfaces.
*Pour lire le contenu d'un registre de lecture selectionne autre que RR0, le programme du systeme doit d'abord ecrire l'octet pointeur dans WR0 exactement de la maniere qu'une operation d'ecriture en registre.Puis,par execution d'une instruction d'entree,le contenu du registre de lecture adresse peut etre lu par le CPU.
*Les bits d'etat de RR0 et RR1 sont soigneusement groupes pour simplifier la suveillance des etats.Par exemple,quand le vecteur d'interruption indique qu'une interruption de condition de reception distinct (RR1).

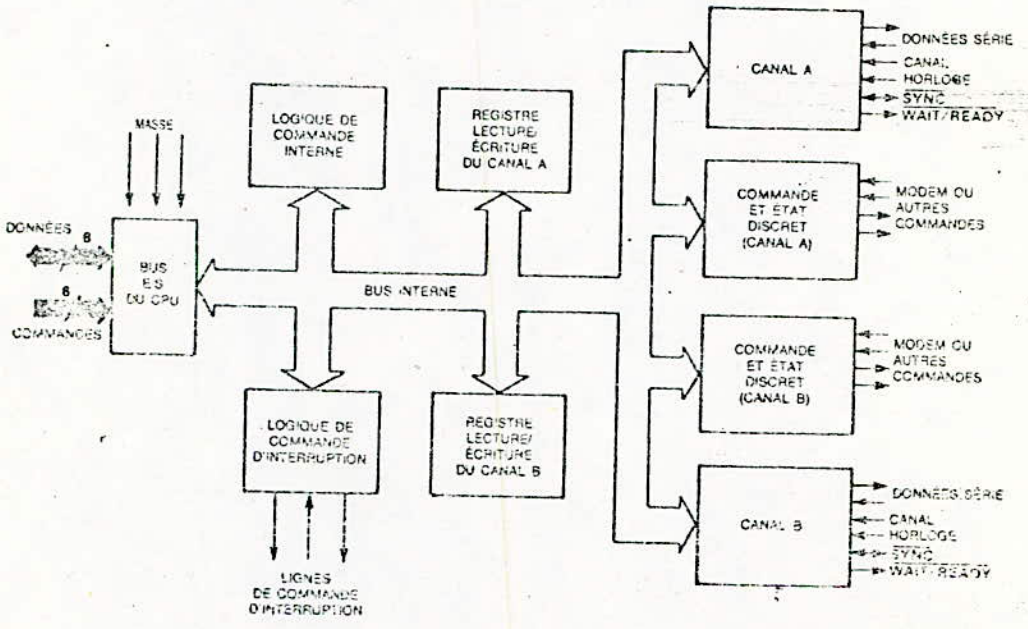
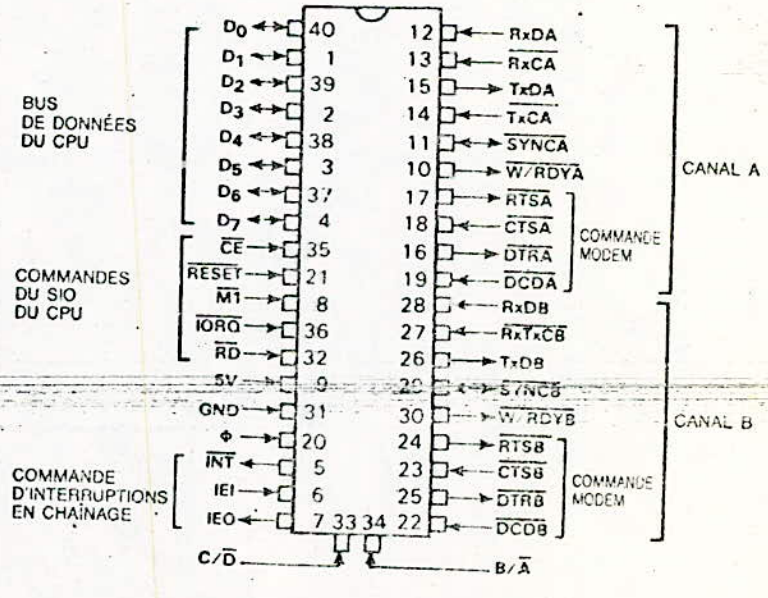
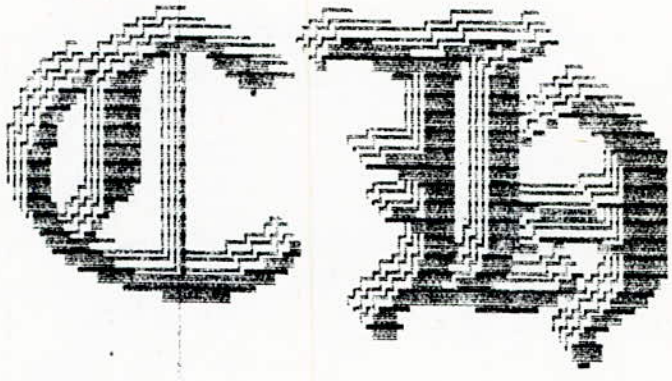


Schéma du Z80-SIO.



Brochage du Z80-SIO



ELEMENTS DE LA COMMANDE NUMERIQUE D'UN MOTEUR A COURANT CONTINU

INTRODUCTION

*L'etude des microprocesseurs en Electrotechnique se fait dans un but bien precis, c'est le domaine de la commande numerique. Vous trouveriez dans ce chapitre quelqués elements de la commande numerique qui donneront un apercu un peu succsent sur la structure de cette commande du point de vue HARD et du point de vue SOFT .

*Il y a quelques annees, les convertisseurs statiques industriels etait realises et commandes par des circuits analogiques ou logiques a faibles ou moyen degre d'integration. Le developpement de la micro-informatique et de la micro-electronique a donne des circuits a tres haut degre d'integration avec plus de possibilites et avec une programmation plus facile. L'utilisation judicieuse des nouvelles technologies, et de ces nouveaux concepts associes a ces dernieres, nous oblige a revoire nos anciens concepts de la commande et de la regulation.

* L'utilisation d'un microprocesseur permet d'obtenir la reponse du moteur a des echelons de consigne autour de differents points de fonctionnement .

*L'utilisation d'uP permet d'envisager, plusieurs controles de la machine.

*Le Kit MT80Z nous permet de concevoir diferents programmes de commande de divers convertisseurs.

ANALYSE DE COMMANDE DES CONVERTISSEURS STATIQUES

PRINCIPE

Ces convertisseurs permettent de modifier la forme ,la valeur,des grandeurs electriques;et ce en ouvrant et fermant des interrupteurs electroniques unidirectionnels, aux temps de reponse tres courts, aux pertes faibles et a encombrement reduit, ils different par la souplesse de la commande des instants d'ouvertures et de fermetures.

ROLES

*Assurer l'alimentation correcte du declencheur de l'interrupteur electronique avec un signal de qualite sans danger pour ce dernier.

*Faire apparaitre correctement le signal de commande selon les criteres requis par le systeme.

LES MODES DE COMMUTATION

La commutation naturelle: Ou l'ouverture est imposee par la reference qui est la tension alternative d'aimentation.

La commutation forcee: Le choix des instants d'ouverture est imposee par une reference independante de la tension d'alimentation qui est generalement continue.

CIRCUITS A COMMUTATION NATURELLE

C'est cette commande qui nous interesse, elle se base sur les conditions suivantes:

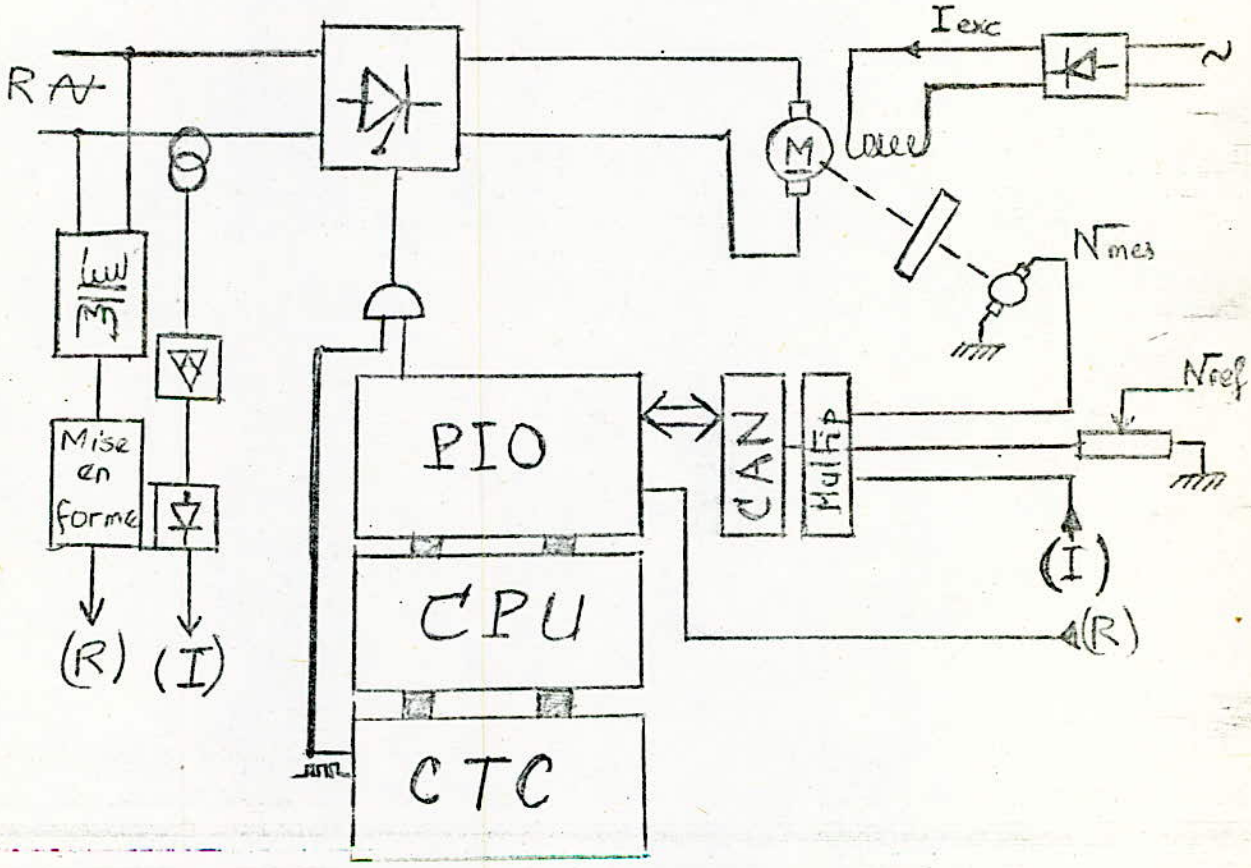
*Le signal de commande n'est applique a l'interrupteur que si la tension a ses bornes est positive.
*Il est constitue d'un train d'impulsions de largeur suffisante pour fixer avec precision l'instant de fermeture.
*Pour eviter les risques d'imprecision sur l'instant de la commande on realise une synchronisation de sa reference sur la phase de l'alimentation alternetive correspondante.

STRUCTURE DE LA COMMANDE NUMERIQUE

ROLE DU MICROPROCESSOR

*Le uP doit se charger de deux taches distinctes:
-La gestion des signaux de commande.
-Le traitement numerique des gandeurs de sortie (courant,vitesse,tension, securite ...).

*Ainsi pour commander le pont de GREATZ,il faut :
-Des signaux de commande delivres par l'interface du uP.
-Un oscillateur a frequence fixe pour generer des trains d'impulsions.
-Un circuit de mise en forme et d'isolement du reseau pour la synchronisation.



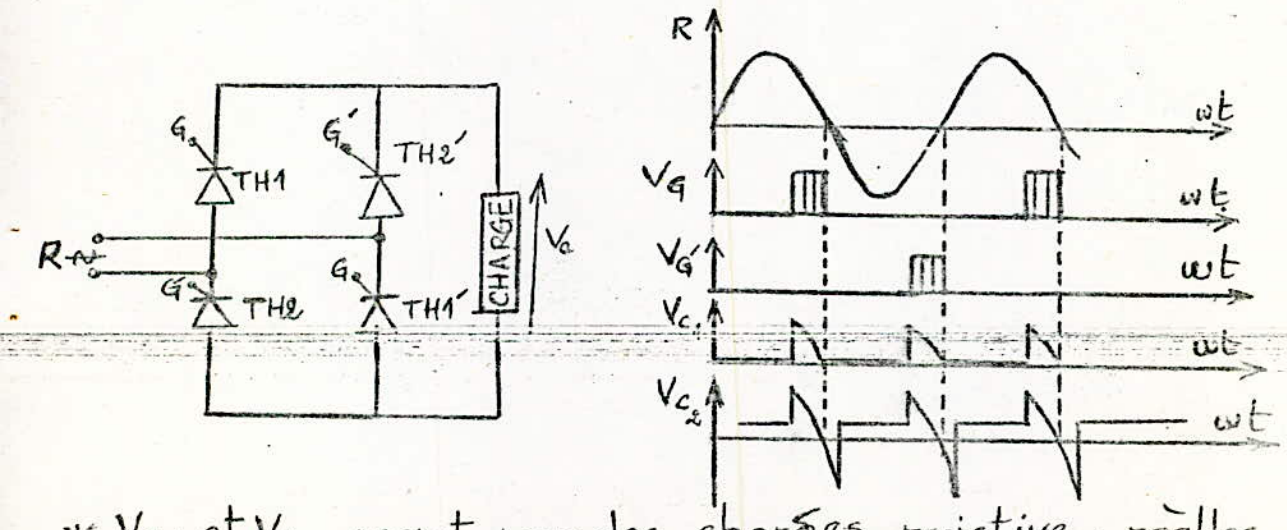
FONCTIONNEMENT D'UN PONT DE GRAETZ MONOPHASE

DETECTION DES PASSAGES PAR ZERO

*Pour permettre au thyristor de conduire pendant les alternances positives du reseau, il faut synchroniser les signaux de commande envoyes aux thyristors avec le reseau.Pour cela il faut detecter les passages par zero et les envoyer au microprocesseur pour qu'il puisse choisir l'instant d'amorcage du thyristor. Comme la tension du reseau est trop elevee par rapport au uP, donc il faut l'isoler, l'attenuer et la transformer pour qu'il soit possible de l'introduire a travers la carte d'acquisition(voir la carte plus loin) .

AMORCAGE DES THYRISTORS:

Pour amorcer un Thyristor il faut :
*Que la tension de l'anode soit superieure a celle de la cathode.
*Qu'il y un signal positif sur sa gachette.Cette meme derniere est attaquee par un train d'impulsions, l'obtention de ce train s'effectue a l'aide d'un signal carree donne par l'oscillateur (CTC). Ce signal est multiplie a l'aide d'une porte AND avec un signal genere a l'aide de la programmation du le PIO . Le signal resultant est amplifie en courant et parfois en tension aussi, puis passe un composant ayant pour role la separation galvanique entre la partie commande (uP) et la partie puissance (Th & Moteur) par exemple les PHOTO-COUPLEURS .



* V_{c1} et V_{c2} respct pour des charges resistive ; reelles

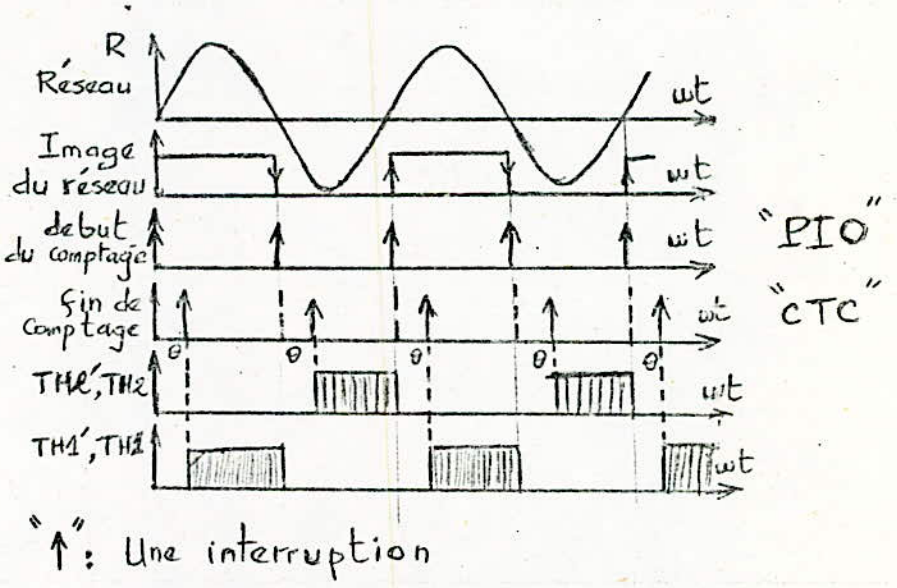
GESTION DES SIGNAUX DE COMMANDE

Pour gerer ses signaux de commande, le uP doit fonctionner en mode "interrompu" et ce pour les aventages suivants :

*Elimination des boucles d'attente pour la synchronisation au reseau.
*Minimisation du nombre de circuits externes.

On distingue deux types d'interruptions succeptibles d'etre generer par les peripheriques (CTC,PIO):
*Une interruption due aux flancs du signal carre (Image du reseau).
*Une interruption due a la fin de comptage.

Les circuits d'entree-sortie CTC, PIO ont pour roles respectivement:
*Generer une interruption en fin de comptage de l'angle d'amorcage. Generation d'un signal carre a une frequence donnee pour attaquer les gachettes des thyristors.
*Prendre en compte les instants de passage par zero a partir de l'image du reseau (signal carre TTL). Generation des signaux de commande. Lecture des mesures de parametres de retour (vitesse, courant...).



DESCRIPTION DE LA CARTE D'INTERFACE

L'ACQUISITION DES DONNEES

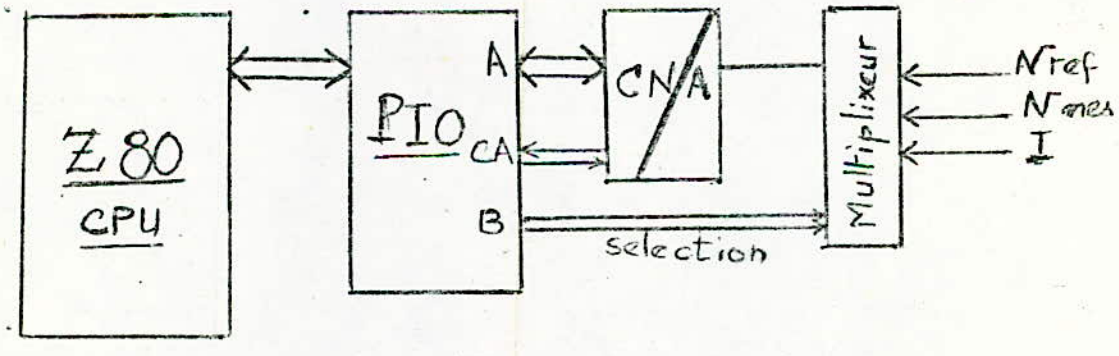
**Comme la tension du'reseau est elevee par raport a celle du uP,donc il faut l'attenuer et la transformer en un signal compatible TTL (0-5V),ou l'alternance positive est representee par le niveau logique 1 et l'alternance negative est representee par le niveau logique 0,comme ceci est montre a la Fig suivante.Pour arriver a cela nous proposons le montage suivant la figure explicative:

*Lorsque le programme a besoin de l'information vitesse, le uP envoie au convertisseur analogique-digital l'ordre "debut de conversion", et attend le signal de "fin de conversion"; il lit alors la donnee sur un des deux ports du PIO .

*L'information vitesse <N>, apres attenuation, est filtree et additionnee a une composante continue afin que toutes les valeurs possibles, negatives ou positives, soient ramenees a un niveau de tension compris entre "0" et "N" a l'entree du convertisseur analogique-numerique; ce qui correspondrait a une valeur numerique comprise entre "00" et "FF" en hexadecimal

*La vitesse de reference est donnee soit par programme et elle est stockee dans une case memoire donnee. Ou elle est fournis a l'aide d'une tension analogique continue numerisee a l'aide d'un convertisseur A/D, dans ce cas et vu que le PIO ne possede que deux ports de 8 bits chacun, on aura besoin d'un multiplixeur analogique ou digital commande par le uP a travers le PIO .

*Dans le cas ou on aurai besoin de l'information courant <I>, apres isolement a travers un transformateur d'intensite, et amplification, il est redressee et filtre. Le courant moyen obtenu est donc l'image du courant circulant dans la machine a courant continu.



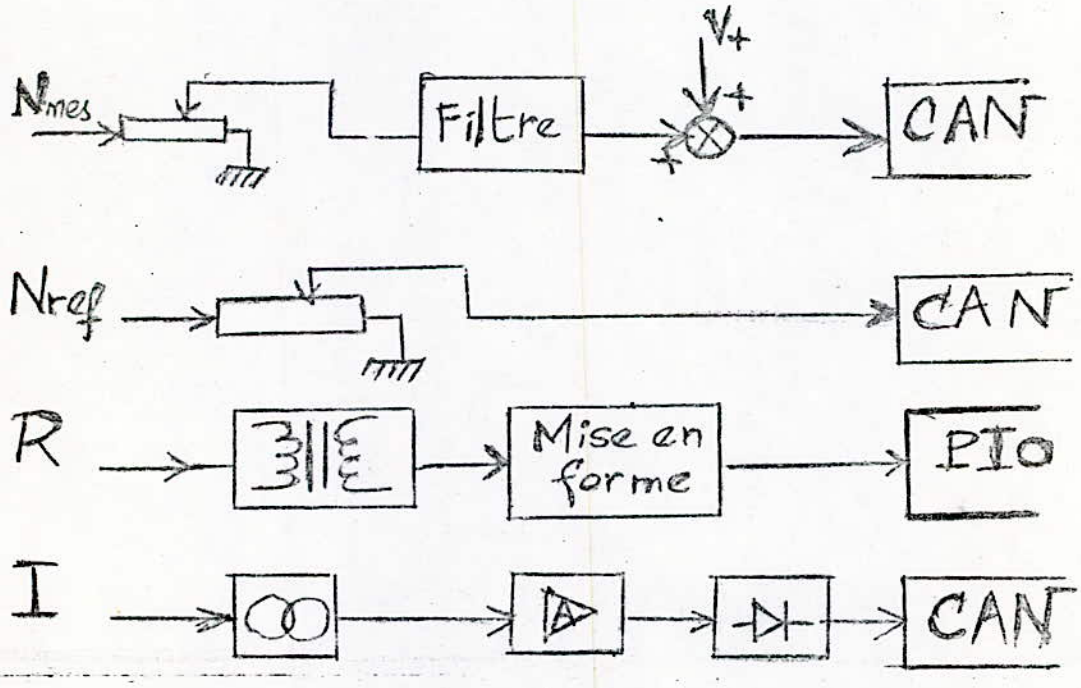
TRAITEMENT NUMERIQUE DES GRANDEURS DE SORTIE

Les echanges entre le uP et un systeme exterieur necessaitent deux interfaces, l'une hardware comprenant les circuits d'interface et l'autre software constituee par les programmes.

Dans le cas d'une regulation de la vitesse par generatrice tachymetrique ou n'importe quel autre capteur de vitesse, les signaux qui doivent etre pris par le uP sont :
*La tension de sortié du capteur de vitesse.
*La tension de reference de la regulation.

Les signaux d'entree (vitesse, reference) necessitent des signaux de controle; dans le cas de la lecture de la vitesse on doit passer par un convertisseur analogique-numerique transformant la tension de sortie du capteur en un signal numerique; Ainsi deux signaux de commande sont necessaire pour commander le convertisseur, l'un pour l'ordre de conversion l'autre pour la fin de conversion.

*L'information vitesse <N>, apres attenuation, est filtree et additionnee a une composante continue afin que toutes les valeurs possibles, negatives ou positives, soient ramenees a un niveau de tension compris entre "0" et "N" a l'entree du convertisseur analogique-numerique; ce qui correspondrait a une valeur numerique comprise entre "00" et "FF" en hexadecimal



PROGRAMMATION DU MICROPROCESSOR

La programmation est constitue par un programme principal et plusieurs sous-programme. Ainsi la programmation se fait en plusieurs etapes. On en cite quelques unes dans le desordre a titre d'exemple.

*Initialisation des peripheriques (Voir Chp "Peripheriques du Z80).
*Sous-programme de synchronisation.
*Sous-programme de regulation.
*Traitement des interruptions.

SOUS-PROGRAMME DE SYNCHRONISATION

*L'apparition du front (Montant ou Descendant) de ce signal provoque une interruption faisant appel a un sous-programme qui a pour role d'inhiber les interruption pas mesure de precaution puis le chargement d'une constante situe dans une case memoire (qui n'est autre que la valeur de l'angle d'amorcage) dans un registre d'etat du CTC,puis le uP autorise les interruptions,alors que le CTC s'autodecremente et generera une interruption en fin de comptage. pour faire appel a un sous-programme d'allumage qui commence par inhiber les intruptions puis determiner la paire de thyristors a allumer et en fin generer le signal d'allumage sur le bit concerne .

SOUS-PROGRAMME DE REGULATION:

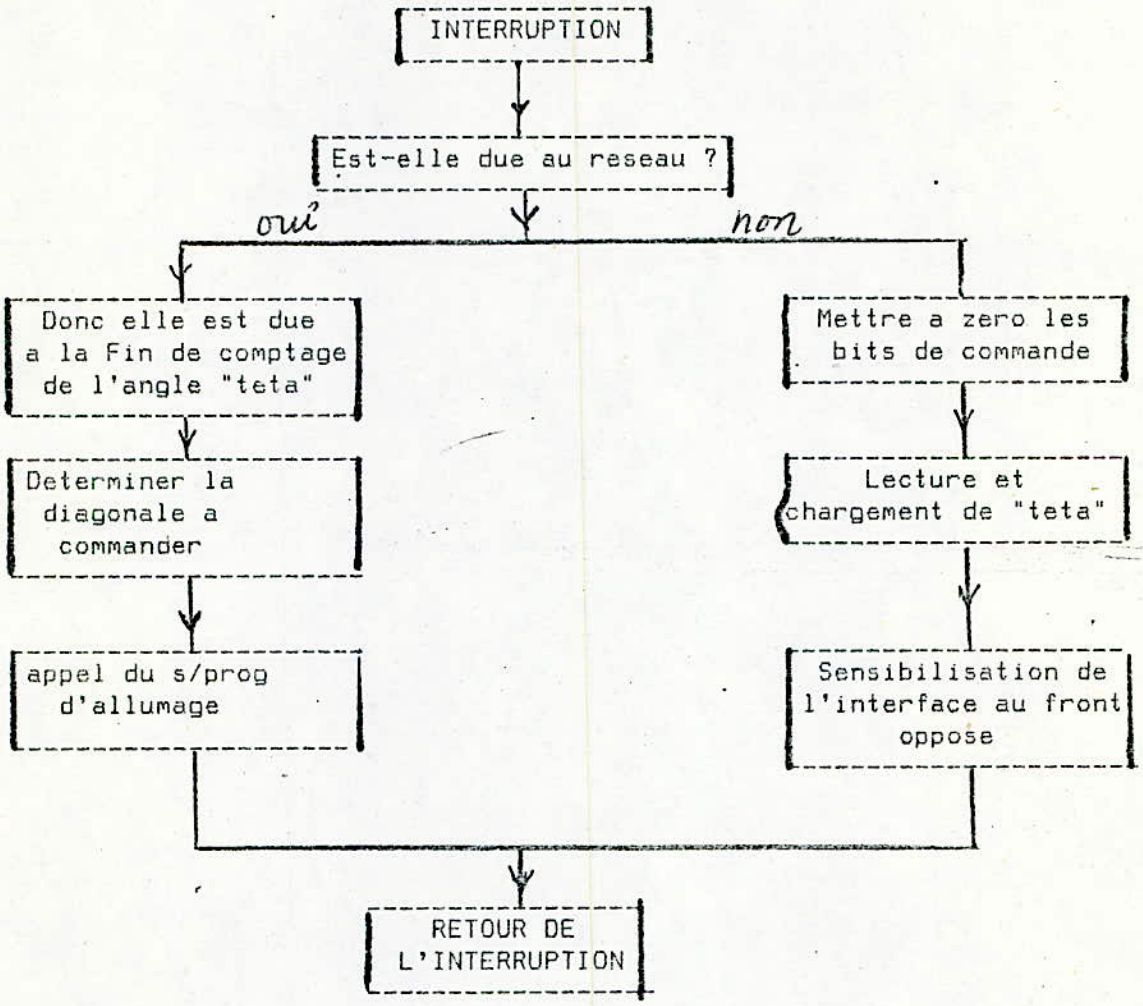
*La loi de commande permet de determiner l'angle de commande du thyristor qui ramenerait la vitesse mesuree a la vitesse de reference choisie, le calcul de l'angle d'amorcage se fait par un programme (qui est un lui meme le regulateur). pour plus de dettille, on vous renvoie aux differents projets de fin d'etudes deja soutenus a notre l'ecole ayant pour theme "la commande numerique " et ce selon les differentes methodes.
*Les methodes de reglage des regulateurs peuvent etre abordees par l'etude detaillee de la fonction de transfert echantillonee du systeme.

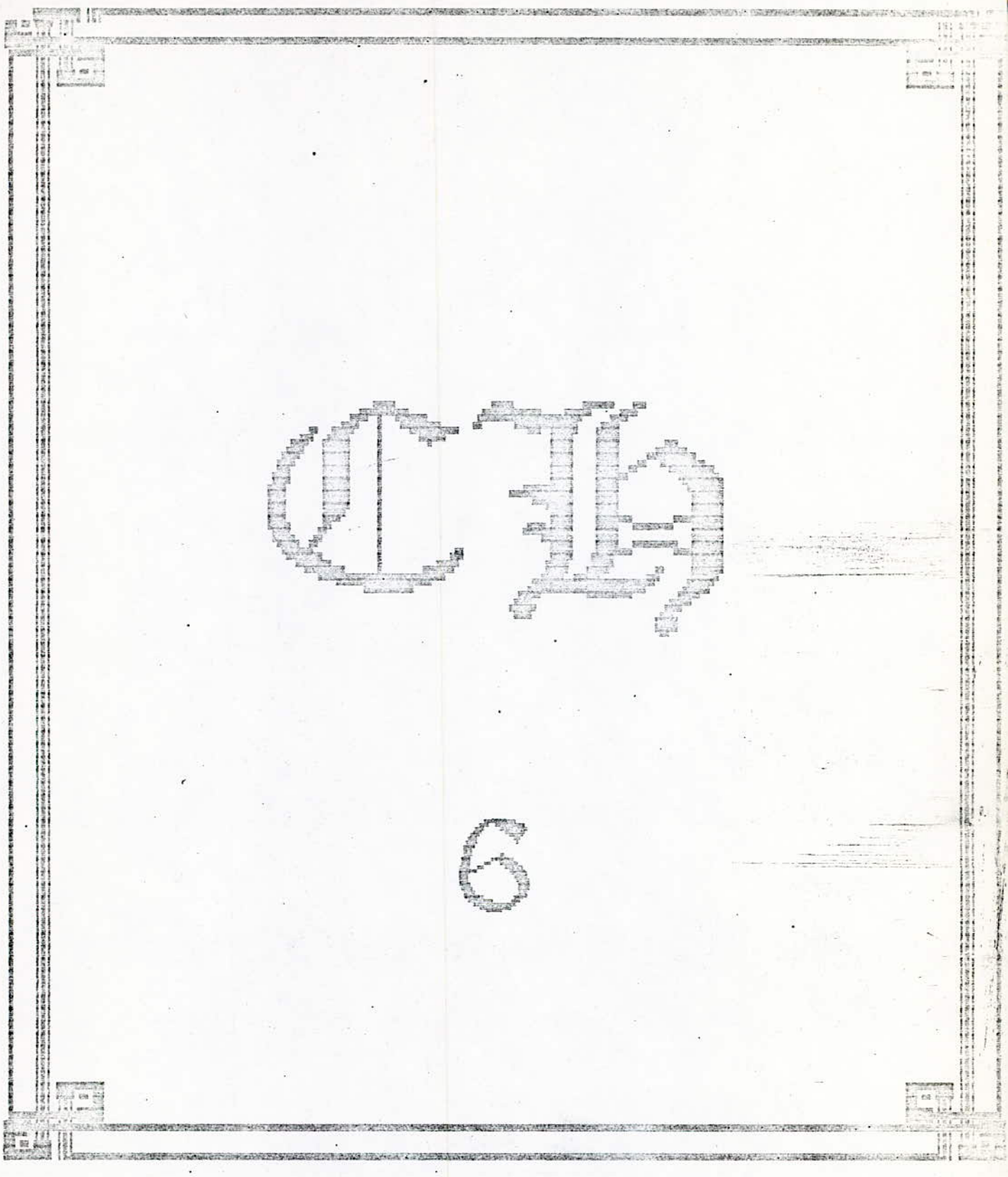
*REMARQUE:

La vitesse en fonction de l'angle d'amorcage est une fonction non lineaire , il serai interressant de lineariser son propre "gain statique" pour simplifier la programmation.

TRAITEMENT DES INTERRUPTIONS

Quand le uP detecte une interruption, il finit l'instruction en court puis s'occupe de l'interruption comme suit:





CONCLUSION

CONCLUSION

L'elaboration d'un manuel de T-P sur une carte a microprocesseur necessite la determination au prealable de l'emploi eventuel de cette derniere.

C'est ce qui a ete fait, il s'agit d'une possible utilisation dans le domaine de la commande numerique des machines sachant que dans le domaine analogique de grands pas ont ete fait dans notre departement.

Vu que le cours sur les microprocesseurs ne fait pas partie du cursus de notre formation (Le systeme modulaire), ce qui nous a pris trop de temps; de plus les peripheriques "PIO" et "CTC" ne font pas partie du Kit. Ainsi nous pensons que les etudiants dans le nouveau systeme seront plus a l'aise avec la carte MT-80Z et puissent en faire un meilleur usage .

BIBLIOGRAPHIE

BIBLIOGRAPHIE

(Par ordre alphabetique)

$ LIVRES

*L'ASSEMBLEUR FACILE DU Z80.	O. LEPAPE
*AU COUER DU MICROPROCESSEUR.	D. GIROD
*COURS D'ELECTRONIQUE (TOME 5)	. MILSANT
*DEPANAGE DES ORDINATEURS PERSONNELS	A. MARGOLIS
*L'EMPLOI DES MICROPROCESSEURS.	M. AUMIAUX
*COMMANDE DE VITESSE ET OPTIMISATION ENERGETIQUE PAR MICROPROCESSEUR (These de Dr-ING 1979 GRENOBLE)	R. FEUILLET
*IDENTIFICATION ET COMMANDE NUMERIQUE DIRECTE PAR MONOCHIP D'UN MOTEUR a CC (These de Dr-ING 1985 NANCY)	Mme.BENHAMZA
*MANUEL DES INTERFACES	S. LEIBSON
*LES MICROPROCESSEURS 8 BITS	E. HORLAIT
*LES SYSTEMES A MICROPROCESSEUR	M. AUMIAUX
*LES SYSTEMES A MICROPROCESSEUR (TOME 3)	L. CLEMENT
*PRATIQUE DE L'ELECTRONIQUE (TOME 1)	M. AUMIAUX
*PRATIQUE DES CIRCUITS LOGIQUES	J. BERNARD
*COURS DU Z80 (Ensigne' a l'N.P.A)	. PODGOVSKY

$ THESES DE FIN D'ETUDES

* LOGITIEL POUR LA COMMANDE NUMERIQUE PAR CALCULATEUR	M. DJOUDER
* COMMANDE D'UN CHARGEUR AUTOMATIQUE PAR MICRO-ORDINATEUR	M. ATOUI/N.BOUFASSA
* TECHNIQUE D'INTERFACE	L. MOUSS HAYET
* COMMANDE D'UNE TABLE TRACENTE	DIAF (79)

ANNEXE

CODES DES INSTRUCTIONS Z 80

DANS LE CODE OBJET, LE LITTÉRAL D EST PRIS ÉGAL A 05

CODE OBJET	INSTRUCTION	
8E	ADC	A,(HL)
DD8E05	ADC	A,(IX+d)
FD8E05	ADC	A,(IY+d)
8F	ADC	A,A
88	ADC	A,B
89	ADC	A,C
8A	ADC	A,D
8B	ADC	A,E
8C	ADC	A,H
8D	ADC	A,L
CE20	ADC	A,n
ED4A	ADC	HL,BC
ED5A	ADC	HL,DE
ED6A	ADC	HL,HL
ED7A	ADC	HL,SP
86	ADD	A,(HL)
DD8605	ADD	A,(IX+d)
FD8605	ADD	A,(IY+d)
87	ADD	A,A
80	ADD	A,B
81	ADD	A,C
82	ADD	A,D
83	ADD	A,E
84	ADD	A,H
85	ADD	A,L
C620	ADD	A,n
09	ADD	HL,BC
19	ADD	HL,DE
29	ADD	HL,HL
39	ADD	HL,SP
DD09	ADD	IX,BC
DD19	ADD	IX,DE
DD29	ADD	IX,IX
DD39	ADD	IX,SP
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD29	ADD	IY,IY
FD39	ADD	IY,SP
A6	AND	(HL)
DDA605	AND	(IX+d)
FDA605	AND	(IY+d)
A7	AND	A
A0	AND	B
A1	AND	C
A2	AND	D

CODE OBJET	INSTRUCTION	
E620	AND	n
CB46	BIT	0,(HL)
DDCB0546	BIT	0,(IX+d)
FDCB0546	BIT	0,(IY+d)
CB47	BIT	0,A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB4E	BIT	1,(HL)
DDCB054E	BIT	1,(IX+d)
FDCB054E	BIT	1,(IY+d)
CB4F	BIT	1,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB56	BIT	2,(HL)
DDCB0556	BIT	2,(IX+d)
FDCB0556	BIT	2,(IY+d)
CB57	BIT	2,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB5E	BIT	3,(HL)
DDCB055E	BIT	3,(IX+d)
FDCB055E	BIT	3,(IY+d)
CB5F	BIT	3,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB66	BIT	4,(HL)
DDCB0566	BIT	4,(IX+d)
FDCB0566	BIT	4,(IY+d)
CB67	BIT	4,A

CODE OBJET	INSTRUCTION	
CB63	BIT	4,E
CB64	BIT	4,H
CB65	BIT	4,L
CB6E	BIT	5,(HL)
DDCB056E	BIT	5,(IX+d)
FDCB056E	BIT	5,(IY+d)
CB6F	BIT	5,A
CB68	BIT	5,B
CB69	BIT	5,C
CB6A	BIT	5,D
CB6B	BIT	5,E
CB6C	BIT	5,H
CB6D	BIT	5,L
CB76	BIT	6,(HL)
DDCB0576	BIT	6,(IX+d)
FDCB0576	BIT	6,(IY+d)
CB77	BIT	6,A
CB70	BIT	6,B
CB71	BIT	6,C
CB72	BIT	6,D
CB73	BIT	6,E
CB74	BIT	6,H
CB75	BIT	6,L
CB7E	BIT	7,(HL)
DDCB057E	BIT	7,(IX+d)
FDCB057E	BIT	7,(IY+d)
CB7F	BIT	7,A
CB78	BIT	7,B
CB79	BIT	7,C
CB7A	BIT	7,D
CB7B	BIT	7,E
CB7C	BIT	7,H
CB7D	BIT	7,L
DC8405	CALL	C,nn
FC8405	CALL	M,nn
D48405	CALL	NC,nn
C48405	CALL	NZ,nn
F48405	CALL	P,nn
EC8405	CALL	PE,nn
E48405	CALL	PO,nn
CC8405	CALL	Z,nn
CD8405	CALL	nn
3F	CCF	
BE	CP	(HL)
DDBE05	CP	(IX+d)
FDBE05	CP	(IY+d)
BF	CP	A
B8	CP	B
B9	CP	C
BA	CP	D
BB	CP	E
BC	CP	H
BD	CP	L

CODE OBJET	INSTRUCTION	
EDB1	CPIR	
EDA1	CPI	
2F	CPL	
27	DAA	
35	DEC	(HL)
DD3505	DEC	(IX+d)
FD3505	DEC	(IY+d)
3D	DEC	A
05	DEC	B
0B	DEC	BC
0D	DEC	C
15	DEC	D
1B	DEC	DE
1D	DEC	E
25	DEC	H
28	DEC	HL
DD2B	DEC	IX
FD2B	DEC	IY
2D	DEC	L
3B	DEC	SP
F3	DI	
102E	DJNZ	e
FB	EI	
E3	EX	(SP),HL
DDE3	EX	(SP),IX
FDE3	EX	(SP),IY
08	EX	AF,AF'
EB	EX	DE,HL
D9	EXX	
76	HALT	
ED46	IM	0
ED56	IM	1
ED5E	IM	2
ED78	IN	A,(C)
ED40	IN	B,(C)
ED48	IN	C,(C)
ED50	IN	D,(C)
ED58	IN	E,(C)
ED60	IN	H,(C)
ED68	IN	L,(C)
34	INC	(HL)
DD3405	INC	(IX+d)
FD3405	INC	(IY+d)
3C	INC	A
04	INC	B
03	INC	BC
0C	INC	C
14	INC	D
13	INC	DE
1C	INC	E
24	INC	H
23	INC	HL
DD23	INC	IX
FD23	INC	IY

CODE OBJET	INSTRUCTION	
EDAA	IND	
EDBA	INDR	
EDA2	INI	
EDB2	INIR	
C38405	JP	nn
E9	JP	(HL)
DDE9	JP	(IX)
FDE9	JP	(IY)
DA8405	JP	C,nn
FA8405	JP	M,nn
D28405	JP	NC,nn
C28405	JP	NZ,nn
F28405	JP	P,nn
EA8405	JP	PE,nn
E28405	JP	PO,nn
CA8405	JP	Z,nn
382E	JR	C,e
302E	JR	NC,e
202E	JR	NZ,e
282E	JR	Z,e
182E	JR	e
02	LD	(BC),A
12	LD	(DE),A
77	LD	(HL),A
70	LD	(HL),B
71	LD	(HL),C
72	LD	(HL),D
73	LD	(HL),E
74	LD	(HL),H
75	LD	(HL),L
3620	LD	(HL),n
DD7705	LD	(IX+d),A
DD7005	LD	(IX+d),B
DD7105	LD	(IX+d),C
DD7205	LD	(IX+d),D
DD7305	LD	(IX+d),E
DD7405	LD	(IX+d),H
DD7505	LD	(IX+d),L
DD360520	LD	(IX+d),n
FD7705	LD	(IY+d),A
FD7005	LD	(IY+d),B
FD7105	LD	(IY+d),C
FD7205	LD	(IY+d),D
FD7305	LD	(IY+d),E
FD7405	LD	(IY+d),H
FD7505	LD	(IY+d),L
FD360520	LD	(IY+d),n
328405	LD	(nn),A
ED438405	LD	(nn),BC
ED538405	LD	(nn),DE
228405	LD	(nn),HL
DD228405	LD	(nn),IX
FD228405	LD	(nn),IY
ED738405	LD	(nn),SP

CODE OBJET	INSTRUCTION	
DD7E05	LD	A,(IX+d)
FD7E05	LD	A,(IY+d)
3A8405	LD	A,(nn)
7F	LD	A,A
78	LD	A,B
79	LD	A,C
7A	LD	A,D
7B	LD	A,E
7C	LD	A,H
ED57	LD	A,I
7D	LD	A,L
3E20	LD	A,n
ED5F	LD	A,R
46	LD	B,(HL)
DD4605	LD	B,(IX+d)
FD4605	LD	B,(IY+d)
47	LD	B,A
40	LD	B,B
41	LD	B,C
42	LD	B,D
43	LD	B,E
44	LD	B,H
45	LD	B,L
0620	LD	B,n
ED4B8405	LD	BC,(nn)
018405	LD	BC,nn
4E	LD	C,(HL)
DD4E05	LD	C,(IX+d)
FD4E05	LD	C,(IY+d)
4F	LD	C,A
48	LD	C,B
49	LD	C,C
4A	LD	C,D
4B	LD	C,E
4C	LD	C,H
4D	LD	C,L
0E20	LD	C,n
56	LD	D,(HL)
DD5605	LD	D,(IX+d)
FD5605	LD	D,(IY+d)
57	LD	D,A
50	LD	D,B
51	LD	D,C
52	LD	D,D
53	LD	D,E
54	LD	D,H
55	LD	D,L
1620	LD	D,n
ED5B8405	LD	DE,(nn)
118405	LD	DE,nn
5E	LD	E,(HL)
DD5E05	LD	E,(IX+d)
FD5E05	LD	E,(IY+d)
5F	LD	E,A
58	LD	E,B

CODE OBJET	INSTRUCTION	
58	LD	E,E
5C	LD	E,H
5D	LD	E,L
1E20	LD	E,n
66	LD	H,(HL)
DD6605	LD	H,(IX+d)
FD6605	LD	H,(IY+d)
67	LD	H,A
60	LD	H,B
61	LD	H,C
62	LD	H,D
63	LD	H,E
64	LD	H,H
65	LD	H,L
2620	LD	H,n
2A8405	LD	HL,(nn)
218405	LD	HL,nn
ED47	LD	I,A
DD2A8405	LD	IX,(nn)
DD218405	LD	IX,nn
FD2A8405	LD	IY,(nn)
FD218405	LD	IY,nn
6E	LD	L,(HL)
DD6E05	LD	L,(IX+d)
FD6E05	LD	L,(IY+d)
6F	LD	L,A
68	LD	L,B
69	LD	L,C
6A	LD	L,D
6B	LD	L,E
6C	LD	L,H
6D	LD	L,L
2E20	LD	L,n
ED4F	LD	R,A
ED7B8405	LD	SP,(nn)
F9	LD	SP,HL
DDF9	LD	SP,IX
FDF9	LD	SP,IY
318405	LD	SP,nn
EDA8	LDD	
EDB8	LDDR	
EDA0	LDI	
EDB0	LDIR	
ED44	NEG	
00	NOP	
B6	OR	(HL)
DDB605	OR	(IX+d)
FDB605	OR	(IY+d)
B7	OR	A
B0	OR	B
B1	OR	C
B2	OR	D
B3	OR	E
B4	OR	H
B5	OR	L
F620	OR	n

CODE OBJET	INSTRUCTION	
EDB3	OTIR	
ED79	OUT	(C),A
ED41	OUT	(C),B
ED49	OUT	(C),C
ED51	OUT	(C),D
ED59	OUT	(C),E
ED61	OUT	(C),H
ED69	OUT	(C),L
D320	OUT	(n),A
EDAB	OUTD	
EDA3	OUTI	
F1	POP	AF
C1	POP	BC
D1	POP	DE
E1	POP	HL
DDE1	POP	IX
FDE1	POP	IY
F5	PUSH	AF
C5	PUSH	BC
D5	PUSH	DE
E5	PUSH	HL
DDE5	PUSH	IX
FDE5	PUSH	IY
CB86	RES	0,(HL)
DDCB0586	RES	0,(IX+d)
FDCB0586	RES	0,(IY+d)
CB87	RES	0,A
CB80	RES	0,B
CB81	RES	0,C
CB82	RES	0,D
CB83	RES	0,E
CB84	RES	0,H
CB85	RES	0,L
CB8E	RES	1,(HL)
DDCB058E	RES	1,(IX+d)
FDCB058E	RES	1,(IY+d)
CB8F	RES	1,A
CB88	RES	1,B
CB89	RES	1,C
CB8A	RES	1,D
CB8B	RES	1,E
CB8C	RES	1,H
CB8D	RES	1,L
CB96	RES	2,(HL)
DDCB0596	RES	2,(IX+d)
FDCB0596	RES	2,(IY+d)
CB97	RES	2,A
CB90	RES	2,B
CB91	RES	2,C
CB92	RES	2,D
CB93	RES	2,E
CB94	RES	2,H
CB95	RES	2,L
CB9E	RES	3,(HL)
DDCB059E	RES	3,(IX+d)

CODE OBJET	INSTRUCTION	
CB9F	RES	3,A
CB98	RES	3,B
CB99	RES	3,C
CB9A	RES	3,D
CB9B	RES	3,E
CB9C	RES	3,H
CB9D	RES	3,L
CBA6	RES	4,(HL)
DDCB05A6	RES	4,(IX+d)
FDCB05A6	RES	4,(IY+d)
CBA7	RES	4,A
CBA0	RES	4,B
CBA1	RES	4,C
CBA2	RES	4,D
CBA3	RES	4,E
CBA4	RES	4,H
CBA5	RES	4,L
CBAE	RES	5,(HL)
DDCB05AE	RES	5,(IX+d)
FDCB05AE	RES	5,(IY+d)
CBAF	RES	5,A
CBA8	RES	5,B
CBA9	RES	5,C
CBAA	RES	5,D
CBAB	RES	5,E
CBAC	RES	5,H
CBAD	RES	5,L
CBB6	RES	6,(HL)
DDCB05B6	RES	6,(IX+d)
FDCB05B6	RES	6,(IY+d)
CBB7	RES	6,A
CBB0	RES	6,B
CBB1	RES	6,C
CBB2	RES	6,D
CBB3	RES	6,E
CBB4	RES	6,H
CBB5	RES	6,L
CBBE	RES	7,(HL)
DDCB05BE	RES	7,(IX+d)
FDCB05BE	RES	7,(IY+d)
CBBF	RES	7,A
CBB8	RES	7,B
CBB9	RES	7,C
CBBA	RES	7,D
CBBB	RES	7,E
CBBC	RES	7,H
CBBD	RES	7,L
C9	RET	
D8	RET	C
F8	RET	M
D0	RET	NC
C0	RET	NZ
F0	RET	P
E8	RET	PE

CODE OBJET	INSTRUCTION	
ED4D	RETI	
ED45	RETN	
CB16	RL	(HL)
DDCB0516	RL	(IX+d)
FDCB0516	RL	(IY+d)
CB17	RL	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
17	RLA	
CB06	RLC	(HL)
DDCB0506	RLC	(IX+d)
FDCB0506	RLC	(IY+d)
CB07	RLC	A
CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
07	RLCA	
ED6F	RLD	
CB1E	RR	(HL)
DDCB051E	RR	(IX+d)
FDCB051E	RR	(IY+d)
CB1F	RR	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
1F	RRA	
CB0E	RRC	(HL)
DDCB050E	RRC	(IX+d)
FDCB050E	RRC	(IY+d)
CB0F	RRC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
0F	RRCA	
ED67	RRD	
C7	RST	00H
CF	RST	08H
D7	RST	10H
DF	RST	18H
E7	RST	20H
EF	RST	28H
F7	RST	30H

CODE OBJET	INSTRUCTION	
9E	SBC	A,(HL)
DD9E05	SBC	A,(IX+d)
FD9E05	SBC	A,(IY+d)
9F	SBC	A,A
98	SBC	A,B
99	SBC	A,C
9A	SBC	A,D
9B	SBC	A,E
9C	SBC	A,H
9D	SBC	A,L
ED42	SBC	HL,BC
ED52	SBC	HL,DE
ED62	SBC	HL,HL
ED72	SBC	HL,SP
37	SCF	
CBC6	SET	0,(HL)
DDCB05C6	SET	0,(IX+d)
FDCB05C6	SET	0,(IY+d)
CBC7	SET	0,A
CBC0	SET	0,B
CBC1	SET	0,C
CBC2	SET	0,D
CBC3	SET	0,E
CBC4	SET	0,H
CBC5	SET	0,L
CBCE	SET	1,(HL)
DDCB05CE	SET	1,(IX+d)
FDCB05CE	SET	1,(IY+d)
CBCF	SET	1,A
CBC8	SET	1,B
CBC9	SET	1,C
CBCA	SET	1,D
CBCB	SET	1,E
CBCC	SET	1,H
CBCD	SET	1,L
CBD6	SET	2,(HL)
DDCB05D6	SET	2,(IX+d)
FDCB05D6	SET	2,(IY+d)
CBD7	SET	2,A
CBD0	SET	2,B
CBD1	SET	2,C
CBD2	SET	2,D
CBD3	SET	2,E
CBD4	SET	2,H
CBD5	SET	2,L
CBD8	SET	3,B
CBDE	SET	3,(HL)
DDCB05DE	SET	3,(IX+d)
FDCB05DE	SET	3,(IY+d)
CBDF	SET	3,A
CBD9	SET	3,C
CBDA	SET	3,D
CBDB	SET	3,E

CODE OBJET	INSTRUCTION	
DDCB05E6	SET	4,(IX+d)
FDCB05E6	SET	4,(IY+d)
CBE7	SET	4,A
CBE0	SET	4,B
CBE1	SET	4,C
CBE2	SET	4,D
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBEE	SET	5,(HL)
DDCB05EE	SET	5,(IX+d)
FDCB05EE	SET	5,(IY+d)
CBEF	SET	5,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBF6	SET	6,(HL)
DDCB05F6	SET	6,(IX+d)
FDCB05F6	SET	6,(IY+d)
CBF7	SET	6,A
CBF0	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBFE	SET	7,(HL)
DDCB05FE	SET	7,(IX+d)
FDCB05FE	SET	7,(IY+d)
CBFF	SET	7,A
CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CB26	SLA	(HL)
DDCB0526	SLA	(IX+d)
FDCB0526	SLA	(IY+d)
CB27	SLA	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB2E	SRA	(HL)
DDCB052E	SRA	(IX+d)
FDCB052E	SRA	(IY+d)
CB2F	SRA	A

CODE OBJET	INSTRUCTION	
CB28	SRA	B
CB2C	SRA	H
CB2D	SRA	L
CB3E	SRL	(HL)
DDCB053E	SRL	(IX+d)
FDCB053E	SRL	(IY+d)
CB3F	SRL	A
CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
96	SUB	(HL)
DD9605	SUB	(IX+d)
FD9605	SUB	(IY+d)
97	SUB	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
D620	SUB	n
AE	XOR	(HL)
DDAE05	XOR	(IX+d)
FDAE05	XOR	(IY+d)
AF	XOR	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
EE20	XOR	n

ÉQUIVALENTS 8080 DU Z 80

Z80	8080
ADC A, (HL)	ADC M
ADC A, n	ACI [B2]
ADC A, r	ADC r
ADD A, (HL)	ADD M
ADD A, n	ADI [B2]
ADD A, r	ADD r
ADD HL, BC	DAD B
ADD HL, DE	DAD D
ADD HL, HL	DAD H
ADD HL, SP	DAD SP
AND n	ANI [B2]
AND r	ANA r
AND (HL)	ANA M
CALL C, nn	CC [B2] [B3]
CALL M, nn	CM [B2] [B3]
CALL NC, nn	CNC [B2] [B3]
CALL nn	CALL
CALL NZ, nn	CNZ [B2] [B3]
CALL P, nn	CP [B2] [B3]
CALL PE, nn	CPE [B2] [B3]
CALL PO, nn	CPO [B2] [B3]
CALL Z, nn	CZ [B2] [B3]
CCF	CMC
CP r	CMP r
CP (HL)	CMP M
CPL	CMA
CP n	CPI [B2]
DAA	DAA
DEC BC	DCX B
DEC DE	DCX D
DEC HL	DCX H
DEC r	DCR r
DEC SP	DCX SP
DEC (HL)	DCR M
DI	DI
EI	EI
EX DE, HL	XCHG

Z80	8080
EX (SP), HL	XTHL
HALT	HLT
IN A, (n)	IN [B2]
INC BC	INX B
INC DE	INX D
INC HL	INX H
INC r	INR r
INC SP	INX SP
INC (HL)	INR M
JP C, nn	JC [B2] [B3]
JP M, nn	JM [B2] [B3]
JP NC, nn	JNC [B2] [B3]
JP nn	JMP [B2] [B3]
JP NZ, nn	JNZ [B2] [B3]
JP P, nn	JP [B2] [B3]
JP PE, nn	JPE [B2] [B3]
JP PO, nn	JPO [B2] [B3]
JP Z, nn	JZ [B2] [B3]
JP (HL)	PCHL
LD A, (DE)	LDAX
LDA, (nn)	LDA [B2] [B3]
LD DE, nn	LXID, [B2] [B3]
LD SP, nn	LXI SP, [B2] [B3]
LD (BC), A	STAX B
LD (DE), A	STAX D
LD (HL), r	MOV M, r
LD (nn), A	STA [B2] [B3]
LD (nn), HL	SHLD [B2] [B3]
LD A, (BC)	LDAX B
LD BC, nn	LXIB, [B2] [B3]
LD HL, (nn)	LHLD [B2] [B3]
LD HL, nn	LXI H [B2] [B3]
LD r, (HC)	MOV r, M
LD r, n	MVI r, [B2]
LD r, r'	MOV r1, r2
LD SP, HL	SPHL
NOP	NOP

Z80	8080
OR n	ORI [B2]
OR r	ORA r
OR (HL)	ORA M
OUT (n), A	OUT [B2]
POP AF	POP PSW
POP BC	POP B
POP DE	POP D
POP HL	POP H
PUSH AF	PUSH PSW
PUSH BC	PUSH B
PUSH DE	PUSH D
PUSH HL	PUSH H
RET	RET
RET C	RC
RET M	RM
RET NC	RNC
RET NZ	RNZ
RET P	RP
RET PE	RPE
RET PO	RPO
RET Z	RZ
RLA	RAL
RLCA	RLC
RRA	RAR
RRCA	RRC
RST P	RST P
SBC A, (HL)	SBB M
SBC A, n	SBI [B2]
SBC A, r	SBB r
SCF	STC
SUB n	SUI [B2]
SUB r	SUB r
SUB (HL)	SUB M
XOR n	XRI [B2]
XOR r	XRA r
XOR (HL)	XRA M

SUMMARY OF FLAG OPERATION

Instruction	D7 S	Z		H		P/V	N	D0 C	Comments
ADD s; ADC s	‡	‡	X	‡	x	v	0	‡	8-bit add or add with carry
SUB s;SBC s; CP s; NEG	‡	‡	X	‡	X	V	1	‡	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	‡	‡	X	‡	X	P	0	0	Logical operations
OR s; XOR s	‡	‡	X	‡	X	P	0	0	
INC s	‡	‡	X	‡	X	V	0	•	8-bit increment
DEC s	‡	‡	X	‡	X	V	1	•	8-bit decrement
ADD DD,SS	•	•	X	X	X	•	0	‡	16-bit add
ADC HL,SS	‡	‡	X	X	X	V	0	‡	16-bit add with carry
SBC HL,SS	‡	‡	X	X	X	V	1	‡	16-bit subtract with carry
RLA; RLCA; RRA; RRCA	•	•	X	0	X	•	0	‡	Rotate accumulator
RL s; RLC s; RR s; RRC s; SLA s; SRA s; SRL s	‡	‡	X	0	X	P	0	‡	Rotate and shift locations
RLD; RRD	‡	‡	X	0	X	P	0	•	Rotate digit left and right
DAA	‡	‡	X	‡	X	P	•	‡	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	‡	Complement carry
IN r; (C)	‡	‡	X	0	X	P	0	•	Input register indirect
INI; IND; OUTI; OUTD	X	‡	X	X	X	X	1	•	Block input and output
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	•	Z = 0 if B ≠ 0 otherwise Z = 1
LDI; LDD	X	X	X	0	X	‡	0	•	Block transfer instructions
LDIR; LDDR	X	X	X	0	X	0	0	•	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI; CPIR; CPD; CPDR	X	‡	X	X	X	‡	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A,I; LD A,R	‡	‡	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
Bit b,s	X	‡	X	1	X	X	0	•	The state of bit b of location is copied into the Z flag

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z = 1 if the result of the operation is zero.
S	Sign flag. S = 1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even. P/V = 0 if the result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
‡	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range V 0.255 V .
nn	16-bit value in range V 0.65535 V .

8-BIT LOAD GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments r, s	Reg.
LD r, s	r ← s	•	•	X	•	X	•	•	•	01 r s		1	1	4	r, s	Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	000	B
															001	C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010	D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 110 ← d →	DD	3	5	19	011 100 101	E H L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	111	A
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7		
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19		
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19		
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10		
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19		
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	ED 36	4	5	19		
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7		
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7		
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13		
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7		
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7		
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13		
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9		
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9		
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9		
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9		

Notes: r, s means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

16-BIT LOAD GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n → ← n →		3	3	10	dd Pair 00 BC 01 DE
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 ← n → ← n →	DD 21	4	4	14	10 HL 11 SP
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 ← n → ← n →	FD 21	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	X	•	X	•	•	•	00 101 010 ← n → ← n →	2A	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	ED	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 ← n → ← n →	DD 2A	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 ← n → ← n →	FD 2A	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	X	•	X	•	•	•	00 100 010 ← n → ← n →	22	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	ED	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 ← n → ← n →	DD 22	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 ← n → ← n →	FD 22	4	6	20	
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	10 HL 11 AF
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
e.g. BC_L = C, AF_H = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ flag is affected according to the result of the operation.

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE↔HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF↔AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	(BC↔BC' DE↔DE' HL↔HL')	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔(SP+1) L ↔(SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_H ↔(SP+1) IX_L ↔(SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_H ↔(SP+1) IY_L ↔(SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE)←(HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	•	•	X	0	X	① ↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE)←(HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE)←(HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE)←(HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A – (HL) HL ← HL+1 BC ← BC-1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A – (HL) HL ← HL+1 BC ← BC-1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A – (HL) HL ← HL-1 BC ← BC-1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A – (HL) HL ← HL-1 BC ← BC-1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: ① P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

8-BIT ARITHMETIC AND LOGICAL GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 [000] r		1	1	4	r Reg.
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 [000] 110 ← n →		2	2	7	000 B 001 C 010 D
ADD A, (HL)	A ← A+(HL)	↕	↕	X	↕	X	V	0	↕	10 [000] 110		1	2	7	011 E 100 H
ADD A, (IX+d)	A ← A+(IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101 10 [000] 110 ← d →	DD	3	5	19	101 L 111 A
ADD A, (IY+d)	A ← A+(IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101 10 [000] 110 ← d →	FD	3	5	19	
ADC A, s	A ← A+s+CY	↕	↕	X	↕	X	V	0	↕	[001]					s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction. The indicated bits replace the [000] in the ADD set above.
SUB s	A ← A - s	↕	↕	X	↕	X	V	1	↕	[010]					
SBC A, s	A ← A - s - CY	↕	↕	X	↕	X	V	1	↕	[011]					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	[100]					
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	[110]					
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	[101]					
CP s	A - s	↕	↕	X	↕	X	V	1	↕	[111]					
INC s	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r [100]		1	1	4	
INC (HL)	(HL) ← (HL)+1	↕	↕	X	↕	X	V	0	•	00 110 [100]		1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	↕	↕	X	↕	X	V	0	•	11 011 101 00 110 [100] ← d →	DD	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d)+1	↕	↕	X	↕	X	V	0	•	11 111 101 00 110 [100] ← d →	FD	3	6	23	
DEC s	s ← s - 1	↕	↕	X	↕	X	V	1	•	[101]					s is any of r, (HL), (IX+d), (IY+d) as shown for INC. DEC same format and states as INC. Replace [100] with [101] in OP Code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow, P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

GENERAL PURPOSE AF OPERATIONS

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

MISCELLANEOUS CPU CONTROL

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM0'	ED 46	8080A MODE
SET INT MODE 1 'IM1'	ED 56	CALL TO LOCATION 0038_H
SET INT MODE 2 'IM2'	ED 5E	INDIRECT CALL USING REGISTER I AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (One's complement)
NEG	$A \leftarrow \overline{A}+1$	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI*	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI*	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

*Interrupts are not sampled at the end of EI or DI

16 BIT ARITHMETIC

DESTINATION		SOURCE: BC	DE	HL	SP	IX	IY
'ADD'	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
INCREMENT 'INC.'		03	13	23	33	DD 23	FD 23
DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

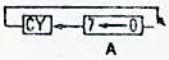
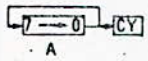
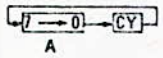
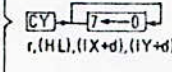
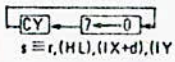
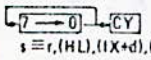
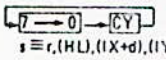
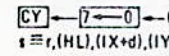
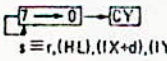
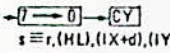
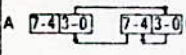
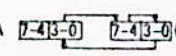
16-BIT ARITHMETIC GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL, ss	HL ← HL+ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL ← HL-ss-CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 00 pp1 001	DD	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss ← ss - 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX - 1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY - 1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

ROTATE AND SHIFT GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 r	CB	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 110	CB	2	4	15	r Reg. 000 B 001 C
RLC (IX+d)	 r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 – d – 00 000 110	DD CB	4	6	23	010 D 011 E 100 H 101 L 111 A
RLC (IY+d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 – d – 00 000 110	FD CB	4	6	23	
RL s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	010					Instruction format and states are as shown for RLC's. To form new Op-Code replace 000 of RLC's with shown code
RRC s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	001					
RR s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	011					
SLA s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	100					
SRA s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	101					
SRL s	 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	111					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

BIT SET, RESET AND TEST GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z \leftarrow \overline{r_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX+d)$_b$	$Z \leftarrow \overline{(IX+d)_b}$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b, (IY+d)$_b$	$Z \leftarrow \overline{(IY+d)_b}$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 – d – 11 b 110	DD CB	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 – d – 11 b 110	FD CB	4	6	23	
RES b, s	$s_b \leftarrow 0$ $s \equiv r$, (HL), (IX+d), (IY+d)	•	•	X	•	X	•	•	•	10					To form new Op-Code replace 11 of SET b, s with 10. Flags and time states for SET instruction

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	Bit Tested
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Notes: The notation s_b indicates bit b (0 to 7) or location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

JUMP GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n ← ← n ←	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n ← ← n ←		3	3	10	cc | Condition 000 | NZ non zero 001 | Z zero 010 | NC non carry 011 | C carry 100 | PO parity odd 101 | PE parity even 110 | P sign positive 111 | M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 ← e-2 ←	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e-2 ←	38	2	2	7	If condition not met
	If C = 1, PC ← PC+e											2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e-2 ←	30	2	2	7	If condition not met
	If C = 0, PC ← PC+e											2	3	12	If condition is met
JR Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e-2 ←	28	2	2	7	If condition not met
	If Z = 1, PC ← PC+e											2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 ← e-2 ←	20	2	2	7	If condition not met
	If Z = 0, PC ← PC+e											2	3	12	If condition is met
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B-1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e-2 ←	10	2	2	8	If B = 0
	If B ≠ 0, PC ← PC+e											2	3	13	If B ≠ 0

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <126, 129>

e-2 in the op-code provides an effective address of pc+e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation.

CALL AND RETURN GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC \leftarrow nn$	•	•	X	•	X	•	•	•	11 – –	001 n n	101 – –	CD	3	5	17	
CALL cc, nn	If condition cc is false	•	•	X	•	X	•	•	•	11 –	cc n	100 –		3	3	10	If cc is false
	continue, otherwise same as CALL nn									–	n	–		3	5	17	If cc is true
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$	•	•	X	•	X	•	•	•	11	001	001	C9	1	3	10	
RET cc	If condition cc is false	•	•	X	•	X	•	•	•	11	cc	000		1	1	5	If cc is false
	continue, otherwise same as RET													1	3	11	If cc is true
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 01	101 001	101 101	ED 4D	2	4	14	
RETN[1]	Return from non maskable interrupt	•	•	X	•	X	•	•	•	11 01	101 000	101 101	ED 45	2	4	14	
RST p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow p$	•	•	X	•	X	•	•	•	11 •	t	111		1	3	11	

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

[1] RETN loads $IFF_2 \rightarrow IFF_1$

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

INPUT AND OUTPUT GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 – n –	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	X	↕ ①	X	X	X	X	1	X	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	↕ ①	X	X	X	X	1	X	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	X	↕ ①	X	X	X	X	1	X	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	X	↕ ①	X	X	X	X	1	X	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	X	↕ ①	X	X	X	X	1	X	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 111 011	ED BB	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Notes: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.