Republique Algerienne Democratique et Populaire

Ministere Des Universites

ECOLE NATIONALE POLYTECHNIQUE

Departement d'Electronique

PROJET DE FIN D'ETUDE

INGENIORAT D'ETAT EN ELECTRONIQUE

THEME:

CONTRIBUTION A LA REALISATION D'UN SYSTEME DE DEVELOPPEMENT POUR LA FAMILLE DES IAPX86

PROPOSE PAR :
- Mme BEDDEK
- Mr SAADOUN

ETUDIE PAR :
- BELKACEMI SAID
- MAHFOUDI MOURAD

Promotion 1992

Republique Algerienne Democratique et Populaire

Ministere Des Universites

ECOLE NATIONALE POLYTECHNIQUE

Departement d'Electronique

PROJET DE FIN D'ETUDE

INGENIORAT D'ETAT EN ELECTRONIQUE

THEME:

CONTRIBUTION A LA REALISATION D'UN SYSTEME DE DEVELOPPEMENT POUR LA FAMILLE DES IAPX86

PROPOSE PAR :
- Mme BEDDEK
- Mr SAADOUN

ETUDIE PAR :
- BELKACEMI SAID
- MAHFOUDI MOURAD

Promotion 1992

Bon courage !!

Dédicaces

à mes parents

Said

Dedicaces

à mes parents
à toute ma famille . . .

mourad .

REMERCIMENTS

Nous tenons à remercier nos deux promoteurs Mme BEDDEK et Mr SAADOUN pour l'aide et l'effort qu'ils nous ont fournis tout au long de ce projet.
Nous remercions également Mr MAHFOUDI KAMEL pour l'aide materielle qu'il nous a donnée.
Que tous ceux qui nous ont porté aide et soutient trouvent ici l'expression de nos sincères remerciements.

SOMMAIRE

INTRODUCTION

Il y'a une vainqtaine d'années, l'industrie éléctronique a permis l'intégration en un seul chip d'une unité arithmétique et loqique (UAL) et d'une unité de commande (UC) : C'est la naissance du microprocesseur.

Par ailleur les besoins accrus en vitesse, précision, et mémoire dans les applications les plus diverses (CAO, Résaux, Gestion de bases de données...) ont ammené à l'évolution de ces microprocesseurs qui passèrent trés rapidement de 4 (4004 d'INTEL) à 32 bits (80486 d'INTEL).Il en résulte une compléxité évidente dans le développement et l'analyse de cartes à base de ces microprocesseurs (nombre d'informations à analyser élevé :signaux, modes de fonctionnement, périphériques). Les systèmes de développements s'avèrent donc une éxigence incontournable pour pouvoir surmonter ces difficultés.

L'objectif de notre projet est la contribution à la réalisation d'un système de développement pour la famille des microprocesseurs 80X86 d'INTEL; Nous nous sommes limités à la conception d'une carte d'analyse loqique s'intégrant dans n'importe quel PC compatible, de son loqiciel qui permet la q e stion et l'émulation du 80286, Premier pas vers l'émulation de cartes à bases de microprocesseurs de la famille INTEL.

Avant de décrire la réalisation soft et hard dans les chapitres 4 et 5, nous présenterons dans le chapitre 2 les pricipes fonctionnels d'un système de développement, le chapitre 3 sera réservé aux microprocesseurs utilisés.

CHAPITRE 1

GENERALITES SUR LES SYSTEMES DE DEVELOPPEMENT

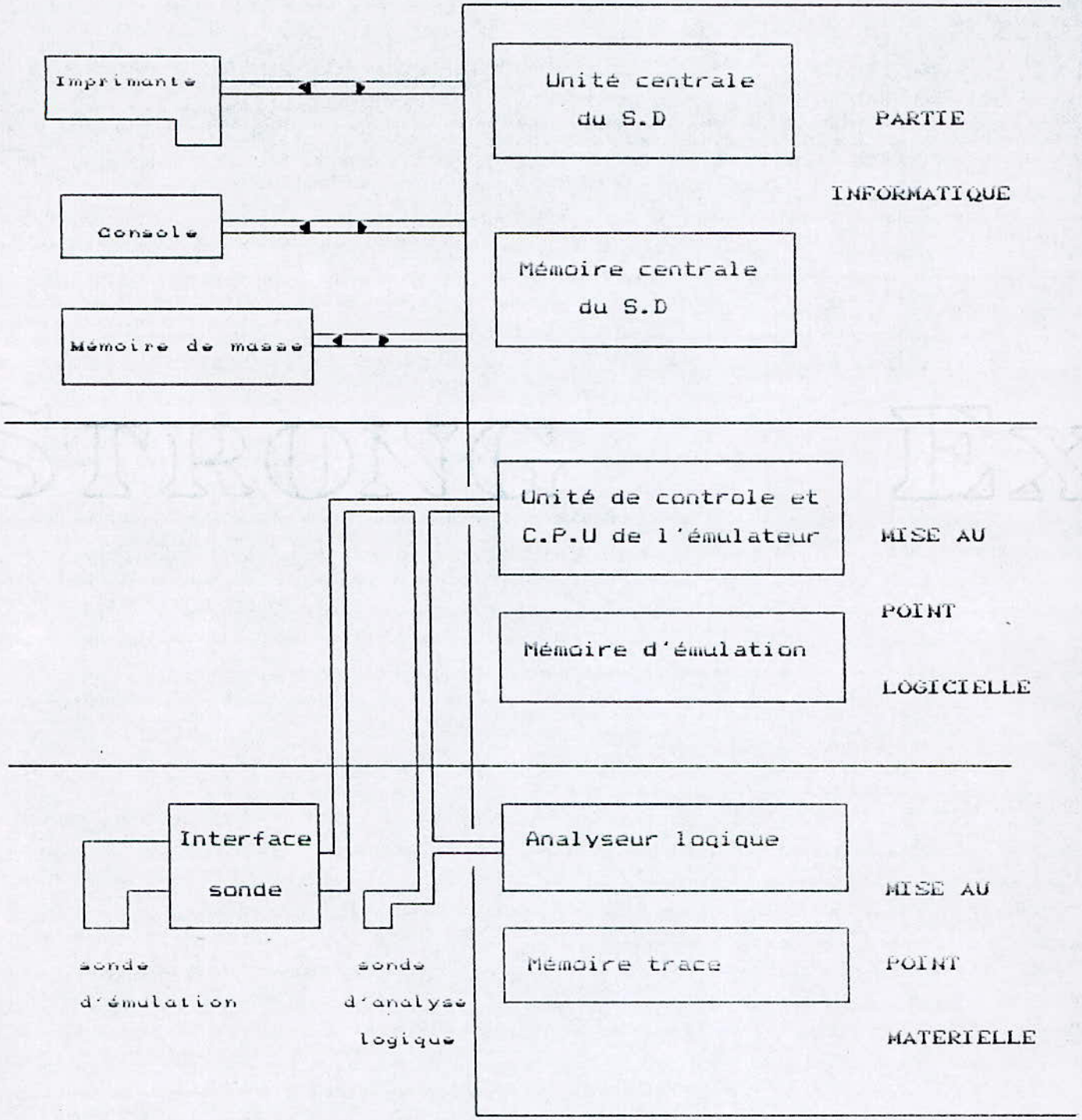
1_ QU'EST CE QU'UN SYSTEME DE DEVELOPPEMENT:

Un système de développement est un outil informatique qui permet de faciliter et d'accélerer la mise au point d'applications à microprocesseurs,puisque ces derniers se résument en fait à une éléctronique programmable,le système de développement doit donc servir à développer et à mettre au point cette éléctronique programmable.
Il fait la jonction entre l'instrumentation éléctronique et la programmation informatique.
Le système de développement est nécessaire dans un laboratoire de micro-informatique,sans ce,toute réalisation pourrait entrainer des difficultés énormes,insurmentables parfois.
Le système de développement permet d'élaborer les programmes d'application relatifs au produit développé c'est à dire:

1- De les introduire et les modifier dans sa mémoire par l'intèrmédiaire d'un clavier alpha-numérique.
2- De les afficher sur un écran de visualisation.
3- De les stocker dans la mémoire de masse.
4- De les traduire dans un langage éxécutable par le microprocesseur de l'application.
5- De faire le lien entre les différents programmes de l'application.
6- De les mettre au point par l'intermédaire d'un émulateur qui va remplacer le microprocessseur de l'application et en simuler dans un premier temps la mémoire et les entrées/sorties.

Il contribue aussi à la mise au point de modules matériels grâce à l'émulateur et à l'analyseur logique et facilite par ailleur l'intégration des programmes dans la partie matérielle.

2_ CONSTITUTION D'UN SYSTEME DE DEVELOPPEMENT:


Imprimante
Console
Mémoire de masse
Unité centrale du S.D
Mémoire centrale du S.D
PARTIE INFORMATIQUE
Unité de controle et C.P.U de l'émulateur
Mémoire d'émulation
MISE AU POINT LOGICIELLE
Interface sonde
Analyseur logique
Mémoire trace
sonde d'émulation
sonde d'analyse logique
MISE AU POINT MATERIELLE

2_1 PARTIE INFORMATIQUE:

La partie informatique est constituée par un ordinateur avec sa mémoire centrale, sa console, sa mémoire de masse, son imprimante et ses logiciels, son role est de saisir, manipuler, stocker le code programme de l'application et de le transformer pour le rendre utilisable par le microprocesseur cible,de plus il doit assurer le dialoque entre l'opérateur et la partie instrumentation.
Les logiciels associés à cette partie sont:

2-1-1 Le système d'éxploitation:

C'est le logiciel de base,il permet l'utilisation du système de développement.Son role est de gérer au mieu l'utilisation des ressources matérielles dont le logicel va disposer: processeurs, mémoires,entrées/sorties.
Pour cela il doit constituer une interface entre les ressources physiques et l'utilisateur en fournissant à ce dernier un langage de commande (avec intèrpréteur associé) et des utilitaires.
Les systémes d'éxploitation multi-tâche entre autre ont pour role de permettre le partage du processeur par plusieures programmes qui, vu,de l'utilisateur,se déroulent en même temps.
Le système d'éxploitation doit gérer les ressources partagées (mémoires,fichiers,ect...) et veiller à la cohérence de l'utilisation des données communes.

2-1-2 L'éditeur de textes:

C'est une pièce maitresse d'un système de développement,il permet la manipulation de programmes,de les modifier et de les stocker pour une utilisation future.

2-1-3 L'assembleur:

Dans les applications à microprocesseurs,l'utilisation du langage machine est trés courante,elle permet essentiellement la concèption de programmes rapides et peu encombrants en mémoire

2-1-4 Le(s) compilateure(s):

L'assembleur est un langage puissant,mais reste spécifique à un microprocesseur donné et son utilisation n'est pas pratique lorsque les programmes sont grands, on a recours alors à l'utilisation de languages de haut niveau qui ont pour avantage: la portabilité des programmes et la durée de mise au point plus courte qu'en assembleur.

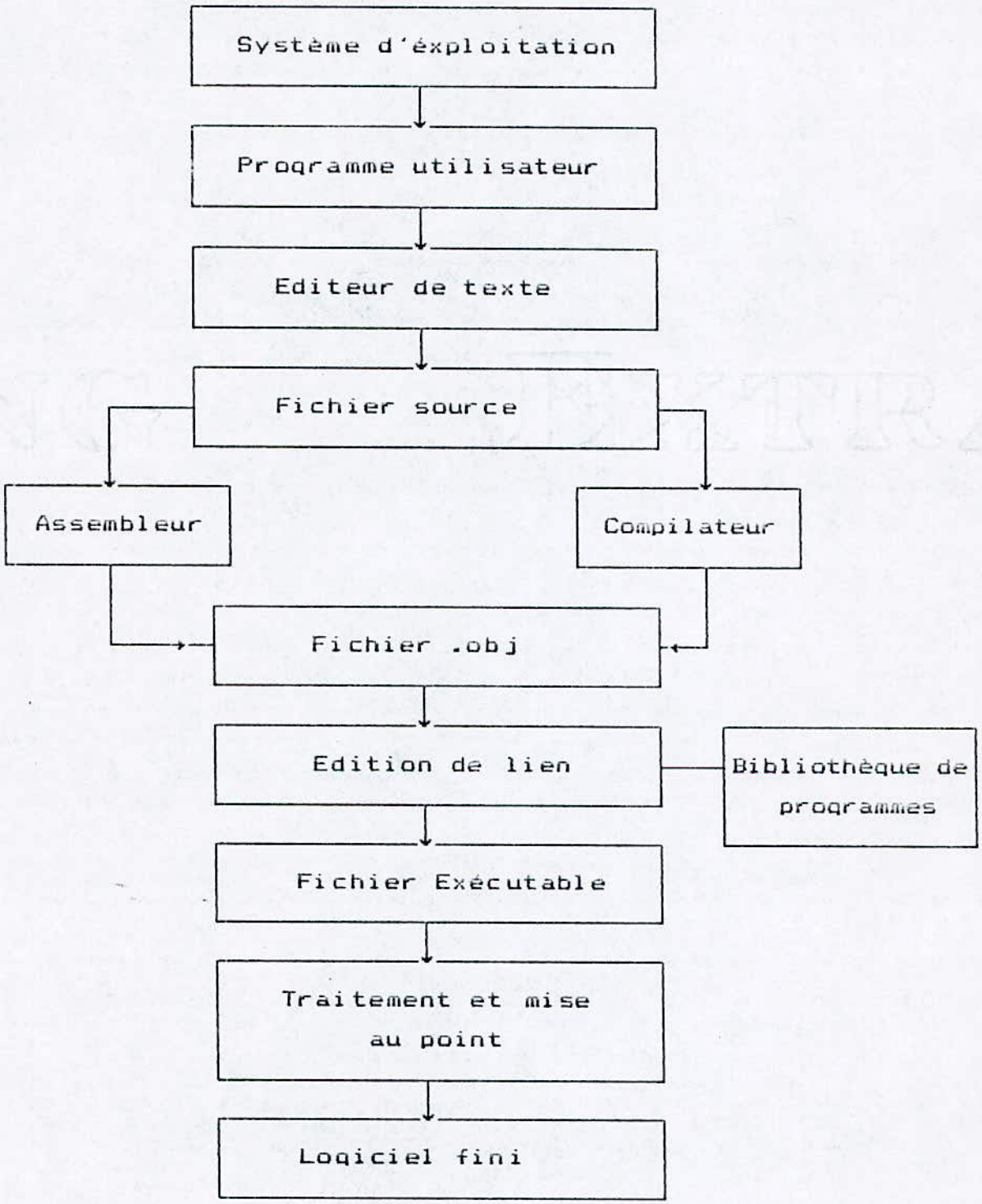
2-1-5 L'éditeur de lien:

Pour entreprendre l'écriture d'un programme long et complèxe, on divise ce programme en modules de petites taille.
L'avantage immédiat est une facilité de mise au point et une possibilité de réutiliser les modules dans d'autres applications.
L'éditeur de lien permet de "rattacher" ensemble les différents modules qui ont étés traduits au préalable en langage machine pour générer le code absolue éxécutable.

2-1-6 LE logiciel de mise au point:

Ce logiciel permet le test et la mise au point du programme, en le lancant, l'arrétant afin d'observer son comportement, pour cela on commande son éxécution par étapes.

OPERATION DE DEVELOPPEMENT DU LOGICIEL:

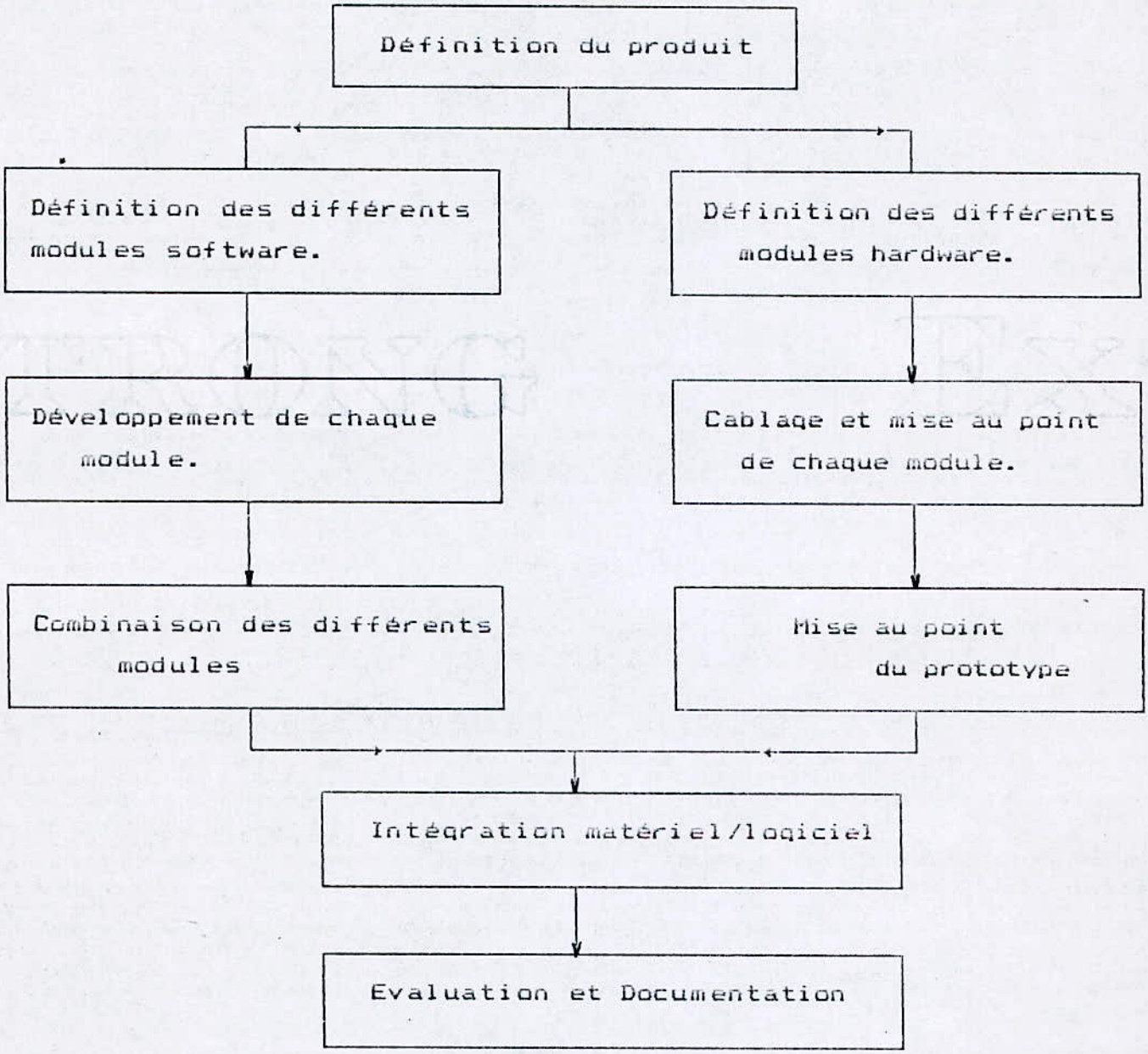


2-2 PARTIE INSTRUMENTATION:

Les outils uniquement logiciels sont des programmes chargés en même temps que le programme à tester en mémoire.On ne réspècte pas les conditions réelles d'éxécution du programme d'application qui est pèrturbé par le logiciel de mise au point que l'on éxécute aussi.
Certaines erreures ou mauvais comportement dû à la vitesse d'éxécution ne peuvent être décelées.
Pour palier à ce défaut on utilise deux instruments:l'analyseur logique qui obsèrve ce qui se passe en perturbant le moin possible le microprocesseur: c'est de l'éspionnage et l'émulateur qui commande son environnement.
Dans la phase finale de mise au point,il faut transférer les programmes du système de développement vers le prototype,on utilise pour cela des mémoires mortes qu'on écrit dessus à travers un programmateur d'éproms.

3 PHASES DE DEVELOPPEMENT D'UNE APPLICATION A BASE DE MICROPROCESSEUR

Pour développer un système à microprocesseur on doit définir le produit en analysant le cahier de charge,pour concevoire le matériel à utiliser et définir les différents modules après quoi on écrit le programme en assembleur ou en langage évolué.Ce programme doit étre téster et corrigé puis intégré dans l'environnement pour lequel il est déstiné.
Lorsque le produit est au point le programme est introduit dans une mémoire morte. Une fois toutes ces étapes tèrminées, le produit est évalué c'est à dire soumis à des tèsts pour connaitre ses performances. Ainsi vient l'étape de la documentation qui comprend les notices d'utilisation.



* PHASES DE DEVELOPPEMENT D'UN PRODUIT

CHAPITRE 2

ANALYSE LOGIQUE
ET
EMULATION

1-L'ANALYSE LOGIQUE:

1-1-INTRODUCTION:

Avant l'apparition d'équipements spécialisés, le développement,la mise au point et le dépannage des systémes numériques se sont avérés trés difficiles,on sentait de plus en plus la nécessité d'avoir des instruments capables de remédier à ces problèmes.

Les analyseurs logiques font partie de cette catégorie d'outils qui ont aidé le développement et la mise au point de systémes à microprocesseurs et ainsi la micro-informatique a pu faire des pas de géants.

L'analyseur logique est un outil d'analyse fonctionnel et d'invèstigation visuel, capable de tèster des circuits integrés complèxes et divers systèmes logiques.

L'opérateur doit prévoir la succéssion des états logiques, sur tout le système, ainsi, une simple comparaison des signaux récupérés et ceux prévues donne la possibilité de détécter une défaillance et localiser sa source.

1-2-UTILISATION:

La tâche essentielle d'un analyseur logique est de prélever plusieurs signaux à la fois pour pouvoir être informer de l'état du systéme sous test.

L'acquisition de ces informations est conditionnée par un ou plusieurs évènements particuliers ce qui permet de faire une certaine sélection d'information pour ne choisir que ce qui est intéressant.

Ces données seront mémorisées pour ensuite être restituées sous une forme éxploitable.
Donc l'enregistrement est fonction d'un déclenchement sur une combinaison binaire ou un mot prés-séléctionné qui indiquera le moment cible d'éspionnage et de collècte d'informations.
Il peut être question de faire apparaitre les évènements qui ont suivit, précédés ou qui entourent le déclenchement.Ce qui facilite rechèrche des causes d'un défaut éventuel.

1-3-MODES DE DECLENCHEMENT ET MEMORISATION:

1-3-1-Déclenchement:

Lors d'une analyse logique le déclenchement peut être provoqué par une combinaison d'évènements suivants:

-Adresse
-Données
-signaux de controle

Ainsi le mot ou l'évènement choisi intervient dans le déclenchement qui sera lié à la réalisation de cet évènement logique

.

1-3-2-Mémorisation:

Pour la collècte d'informations et leurs mémorisation,les analyseurs logiques proposent plusieurs modes de mémorisation pour que l'analyse soit plus souple et plus malliable.

1-3-2-1-Mode 1: Prédéclenchement:

On veut connaitre dans ce mode ce qui a précédé le mot choisi,l'analyse loqique dans ce cas enreqistre les états loqiques en pèrmanence, lorsque la mémoire sera remplie ou totalement saturée l'information suivante chasse la premiére et prendra sa place.La réalisation de l'évènement pré-séléctionné fait arrêter la mémorisation et ainsi on pourra éxploiter tous ce qu'on a pu emmaqasiner avant le siqnal de déclenchement.

1-3-2-2-Mode 2:Post Déclenchement

Dans ce mode on veut connaitre les états qui ont suivits la réalisation de l'évènement choisie, l'analyse loqique est inactif dans ce cas, mais reste éveiller, jusqu'à l'apparition du siqnal de déclenchement qui veut dire que le mot pré-sélectionné a été réalisé ainsi la mémoire, viérqe jusqu'ici, commencera à se remplir jusqu'à saturation.

1-3-2-3-Mode 3:Déclenchement

Dans ce mode ,on veut connaitre les états loqiques qui entourent la réalisation d'un évènement loqique pris comme référence.Dans un premier temps il y'aura mémorisation continuelle comme dans le mode 1 jusqu'à apparition du siqnal de déclenchement qui bloquera cette partie de mémoire dans sa dèrnière position, tandisque l'extension mémoire réservée à cet effet,et jusqu'ici dépourvue de toute information ,se voit validée et remplie jusqu'à sa capacité maximale.

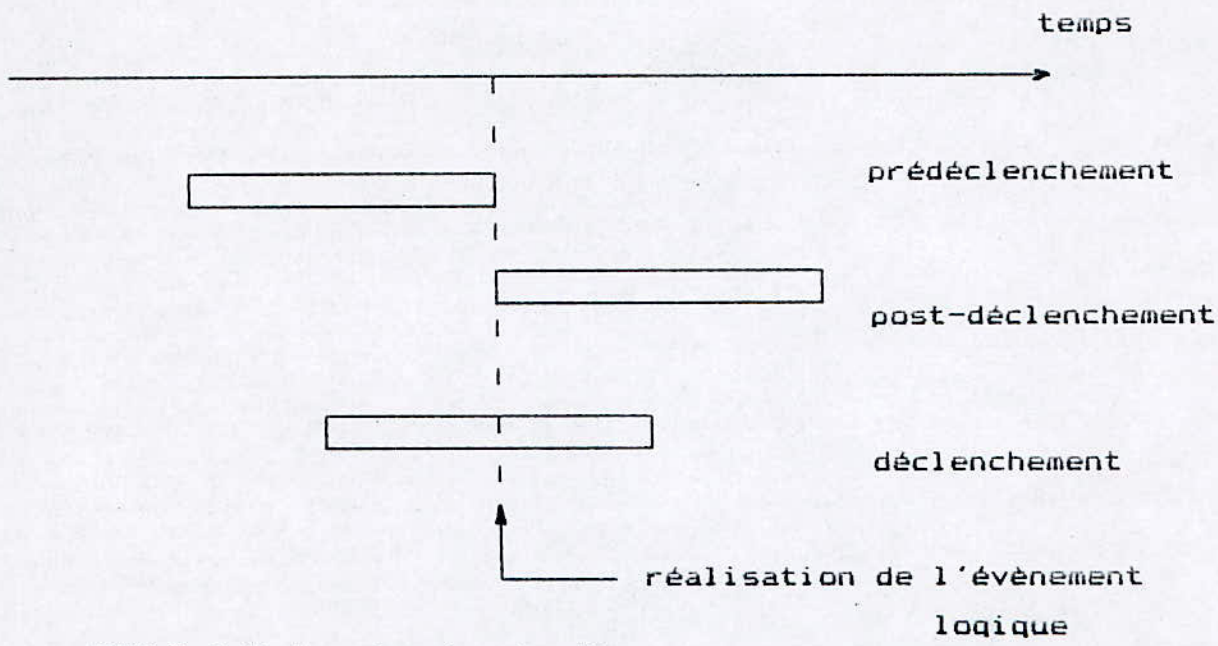


FIG 3-1 Modes de mémorisation

1-4-MODES DE VISUALISATION DES INFORMATIONS:

Les informations receuillis et mémorisées seront visualisées sur écran selon trois possibilités:

1-4-1-Mode état:

dans ce mode, le contenue des differentes voies, apparrait en notation héxadécimale, octale ou binaire.

1-4-2-Mode temporel:

L'analyseur logique se comporte dans ce mode comme un oscilloscope à plusieurs canaux,la forme des signaux de chaque voie est affichée sur une ligne de l'écran, donc on aura une représentation temporelle des résultats qui permettera de détecter les erreurs matérielles.

1-4-3-Mode carte graphique:

Dans ce mode on représentera de facon globale le déclenchement d'un programme ou d'une séquence en visualisant chaque mot sous forme d'un point unique sur l'écran.
EXEMPLE:Pour une donnée de 16 bits,on aura les 8 bits de poids faible en abscisse et les 8 bits de poids fort en ordonnée.
Dans ce mode une séquence de programme se traduit par un graphe constitué par plusieurs segments résultant du passage de la donnée d'une valeur à une autre.

2-L'EMULATION:

2-1-PRINCIPE:

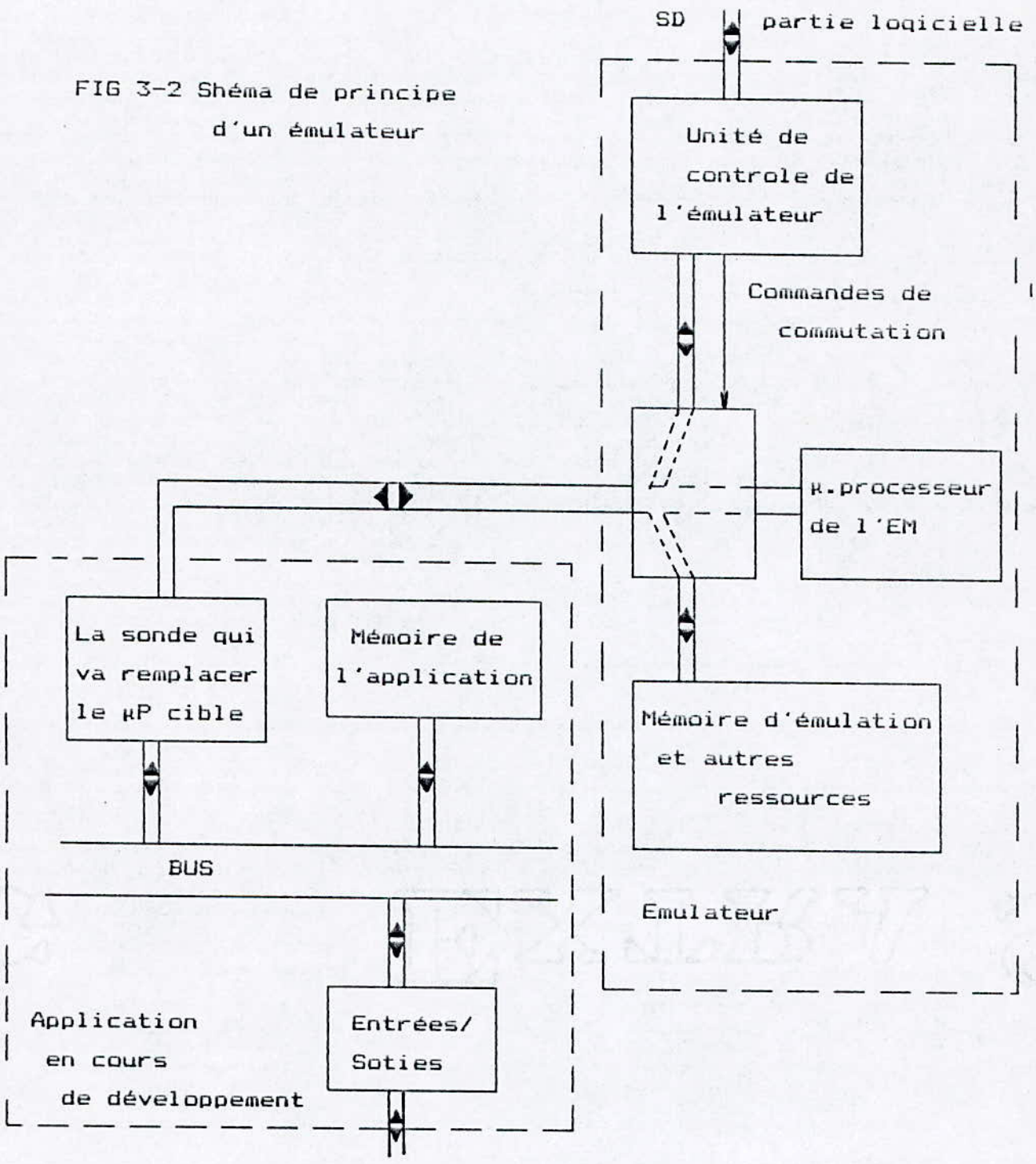
Un systéme à microprocesseurs peut être identifier par son comportement GLOBAL, c.a.d par la relation entre ses entrées et ses sorties.
Par contre sa mise au point nécessite d'avoir accés aux fonctions intèrnes qui contribuent à ce comportement global.
L'émulateur est un dispositif qui remplace le microprocesseur de

l'application développée(Microprocesseur cible)par un systéme ayant un comportement identique(en général,il s'agit d'un microprocesseur identique au microprocesseur cible) relié au systéme de développement.
De cette maniére les fonctions intèrnes du prototype sont accessibles lors de la mise au point sans modifier les performances.
De plus tout ou partie de la mémoire ou des entrées/sorties du prototype peut être simuler dans la mémoire d'émulation du systéme de développement .
De facon imagée,on projette le microprocesseur cible et à travers lui le prototype dans le systéme de developpement.
le schéma de principe d'un émulateur est représenté à la FIG.3.1, l'unité de controle de l'émulateur, pilotée par le SD(logiciel d'emulation)a pour role de gérer les connexions entre les différents éléments de l'émulateur (sonde,memoire,...ect) et d'éxécuter les commandes correspondantes à la fonction d'émulation.

2-2-CONDITIONS POUR UNE BONNE EMULATION:

1-L'émulation ne doit pas utiliser les ressources du systéme sous test par éxemple les mémoires, les lignes d'interruption ou DMA
2-L'émulation doit réspécter la vitesse d'éxécution du programme .

FIG 3-2 Shéma de principe
d'un émulateur



L'émulation se présente en trois étapes ou modes:

1-Emulation d'ordre zero:

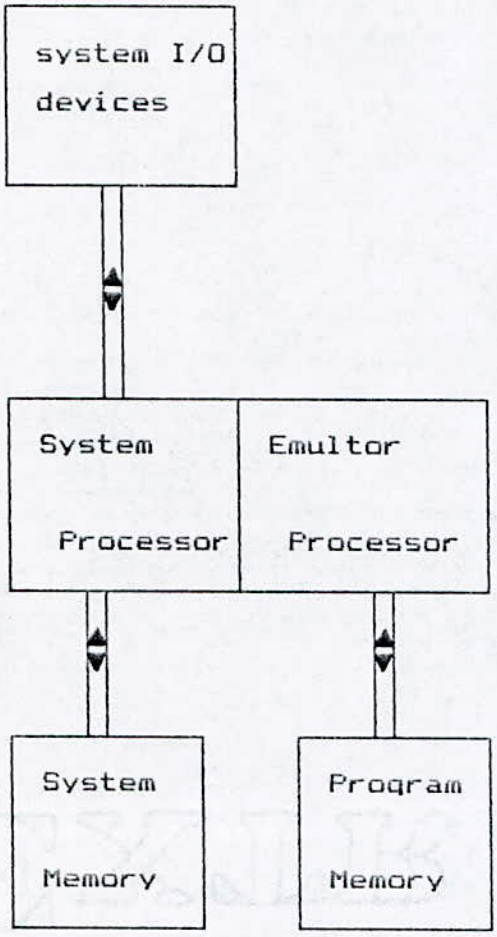


FIG 3-3 Emulation logicielle

L'émulation d'ordre zéro ou l'émulation logicielle consiste en un développement du logiciel de la carte prototype dans la mémoire de l'émulateur,sans aucune interaction avec le prototype. Il s'agit donc de mettre au point des programmes d'application relatifs à la carte prototype, il faut pouvoir les charger, les lancer, les arrêter, afin d'observer leurs comportements.
Pour cela on commande leur éxécution par étapes(instruction par instruction, ou bien jusqu'à une instruction donnée), aprés l'éxécution de chaque morceau,on examine les résultats obtenus (dans les registres,dans les mémoires,en I/O) que l'on comparera avec ce qui était prévue pour mettre en évidence les erreurs éventuelles.
Les outils logicielles utilisés sont des programmes chargés en mémoire en même temps que le programme à tester, ils permettent les manipulations utiles pour la mise au point,lire-écrire des mémoires, des registres, placer des points d'arrêt etc..., Les conditions réelles d'éxécution des programmes d'application ne sont pas réspéctés à cause des pèrturbations dûes au logiciel de mise au point que l'on éxécute aussi,certaines erreurs ou comportements liées au temps ou à la vitesse de déroulement du programme, ne peuvent alors être mises en évidence. On notera que Les E/S du protoype seront simulées entièrement.

2-Emulation d'ordre 1:

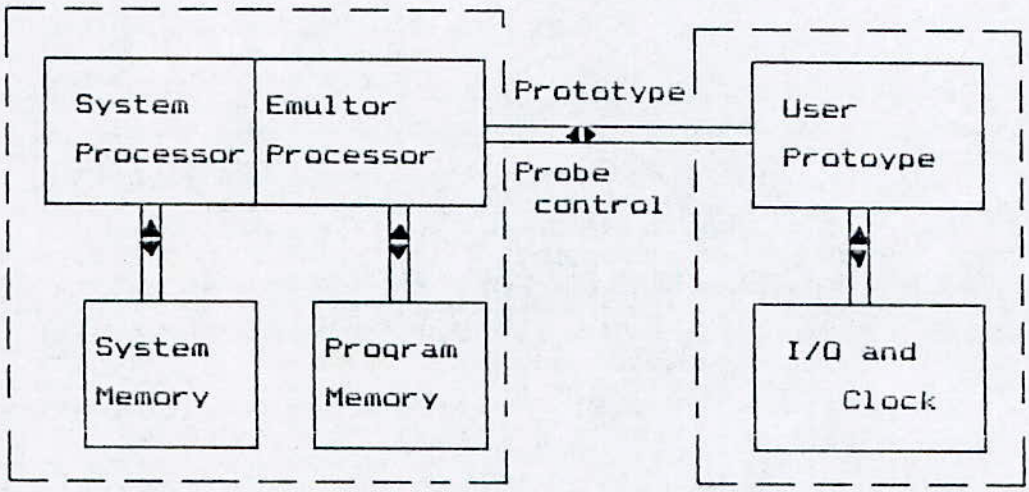


FIG 3-4 Emulation d'ordre 1

L'émulation d'ordre 1 consiste en l'éxécution du logiciel de la carte à développer dans le prototype mais toujours sous le contröle de l'émulateur avec toutes les possibilités de controle et de correction . le logiciel est éxécuté à partir de la mémoire programme (mémoire de l'émulateur).L'horloge cette fois est fournie par la carte à développer.

3-Emulation d'ordre 2:

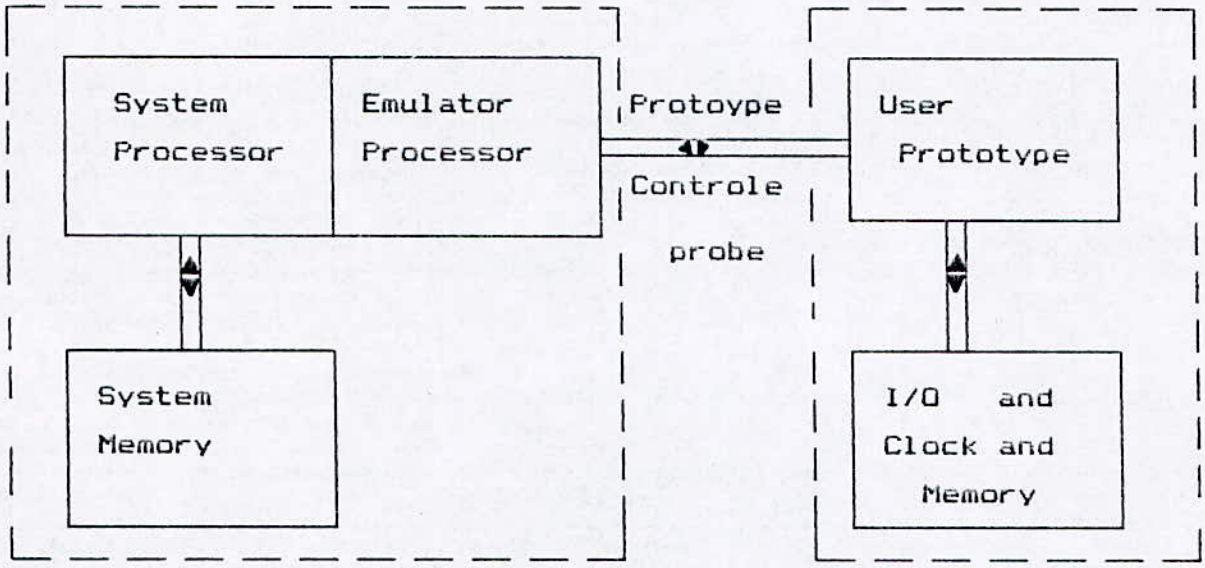


FIG 3-5 Emulation d'ordre 2

Dans l'étape précédante le loqiciel de l'utilisateur est testé dans un matériel qui n'est pas celui auquel il a été déstiné (car éxécuté sur le matériel du systéme de dvpt). Le principe de cet ordre d'émulation est de tester l'ensemble du loqiciel dans son environnement véritable mais sous contröle de l'émulateur à travers la sonde.Comme dans le mode 2 l'horloqe est fournie par le système sous tèste.

CHAPITRE 3

PRESENTATION DU 8086 ET DU 80286

1-PRESENTATION DU 8086:

1-1 GENERALITES:

Le 8086 concu par Intel,est un microprocesseur 16 bits en technologie HMOS, il peut adresser un million d'octets,organisés en 16 pages de 64 KO,l'accès à une position mémoire se fait en deux temps : accès au segment puis déplacement à l'intérieur du segment. Son horloge de base est de 5MHZ, il éxiste des versions plus rapides à 8 et même 10MHZ.

Les caractéristiques générales du 8086 sont :

1-Un bus Adresse/Données multipléxé 16 bits.

2-Un adressage paginé simplifié, un circuit de géstion mémoire rudimentaire est intégré sur la puce du microprocesseur.

3-Il a 4 éspaces d'adressages séparés: "Programme", "Données", "Pile", et "Données supplémentaires".

4-Deux modes de fonctionnement sont possibles:

a-<u>mode minimal:</u> Le 8086 travaille dans ce cas d'une manière analogue au 8085 et ne peut adresser que 64KO.

b-<u>mode maximal:</u> Le CPU peut dans ce cas adresser 1MO de mémoire physique.

5-Deux structures d'entrée/sortie possibles une structure E/S par instruction E/S et une structure E/S par instruction mémoire,dans le premier cas le 8086 permet d'adresser 64 Kports de 8 bits en dehors de l'éspace mémoire.

1-2-LES REGISTRES INTERNES DU 8086:

Vu de l'utilisateur Le 8086 comprend 3 groupes de 4 registres 16 bits et un registre flag. Il dispose également d'un compteur ordinal non accessible par l'utilisateur.

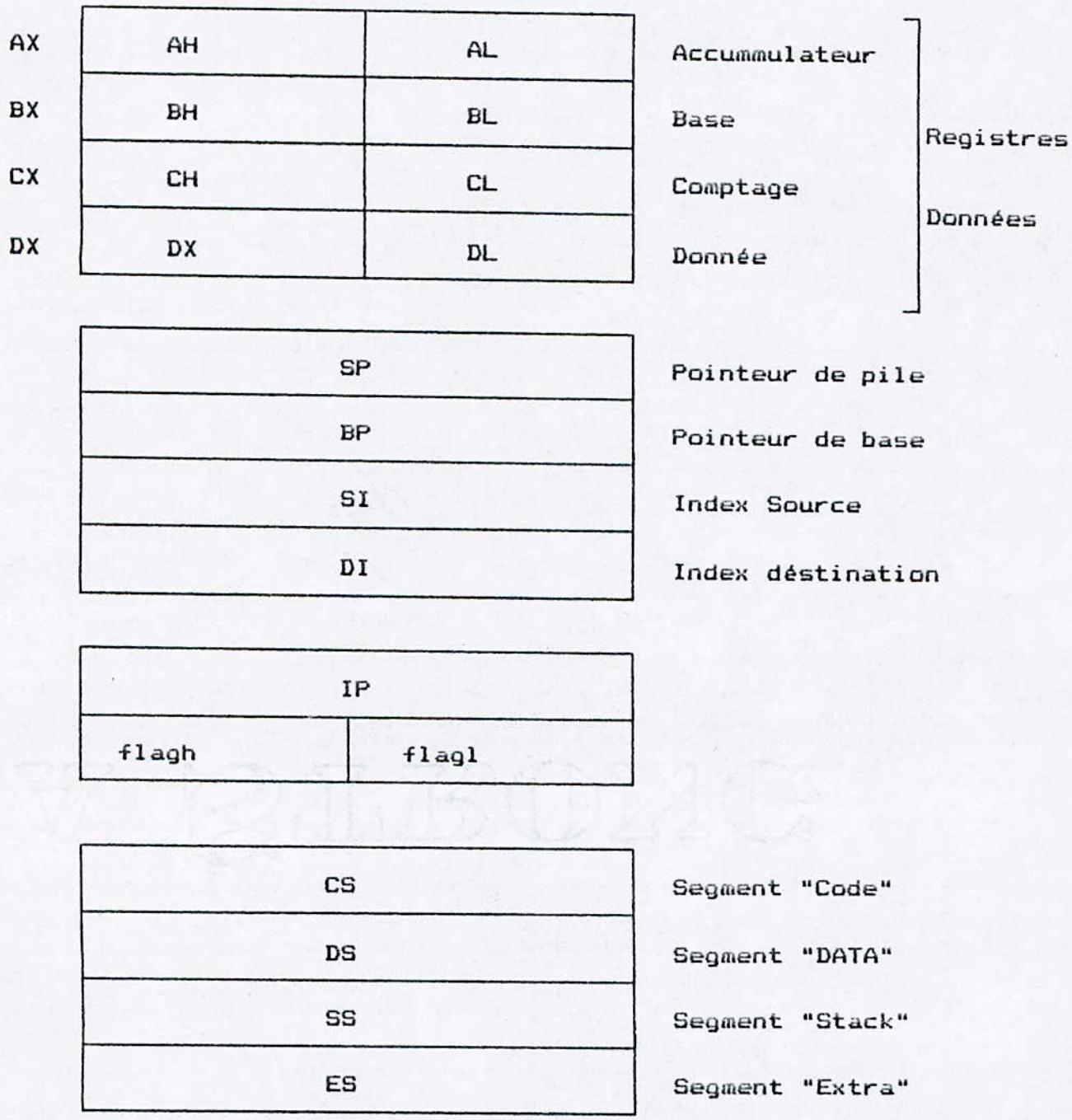


FIG 4-1 Les Registres Internes du 8086

1-2-1-Les registres de données:

Ils se comportent comme des Accumulateurs, ils participent aux opérations arithmétiques et logiques que doit efféctuer le 8086.
Il est à noter que les quatres registres de 16 bits peuvent se comporter comme 8 registres de 8 bits.

1-2-2-Les registres de segments:

Permettent à tous moments,au microprocesseur,d'adresser 256 KO de mémoire, il peut accéder aux autres parties de la mémoire en changeant les contenues des registres segments.

1-2-3-Les registres pointeurs:

Comme les regitsres de données, ils participent aux opérations arithmétiques et logiques.
SP et BP sont pris par défaut pour éxprimer un déplacement à l'intérieur du segment pile, SI et DI sont pris par défaut pour éxprimer un déplacement à l'intérieur du segment données sauf pour les opérations sur les chaines de données.

1-2-4-Le registre d'état:

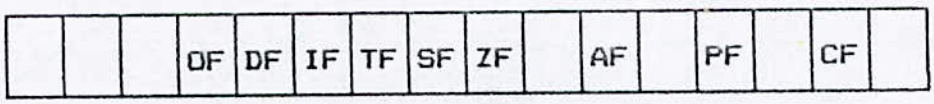


FIG 4-2 Le registre flag

Le registre flag est un registre dont chacun des bits qui le composent a un sens :

CF: Retenue
PF: Parité
AF: Retenue Intermédiaire
ZF: Zéro
TF: Bit indiquant le mode trap
SF: Bit signe
OF: Bit overflow
DF: Bit de direction
IF: Bit autorisation d'interruption

1-3-ORGANISATION DE LA MEMOIRE:

1-3-1-organisation des données en mémoire:

Le 8086 a 20 lignes d'adresse, ceci signifie qu'il a 1 million emplacements de stockage distincts adressable. Comme son bus de données est de 16 bits, la mémoire du systéme à base du 8086 a une longueur de 16 bits (soit 2 octets), ces 2 octets sont appelés octet supérieur et octet inférieur.

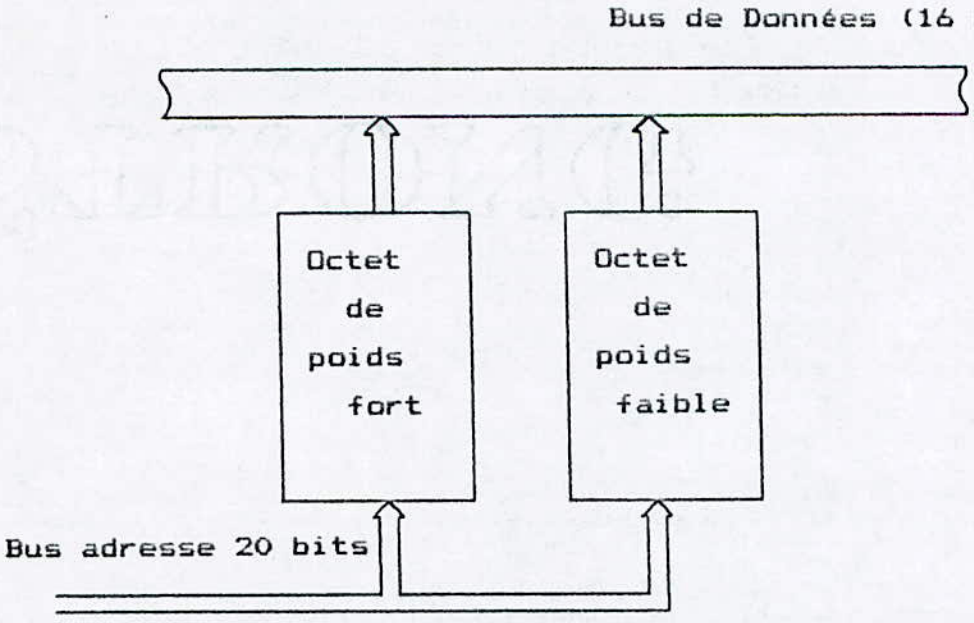


FIG 4-3

Le microprocesseur peut accéder à n'importe lequel du million d'octets possibles de la mémoire, comme chaque mot de la mémoire emploie 2 octets, ceci donne une mémoire système de 2 adresses de 16 bits.Il est à noter que le 8086 peut accéder aussi bien à des octets en mémoire qu'à des mots.
Deux signaux déterminent l'octet de la mémoire qui fait l'objet de l'accés : A0 et BHE.

** Mots alignés et mots non alignés:

Quand le C.P.U accède à un mot situé à une adresse paire, il accède à un mot aligné. Le mot est aligné car 2 octets sont tous deux situé à la même adresse mot et peuvent être lus ou écrits en un seul cycle bus.
Quand le C.P.U accède maintenant à un mot situé à une adresse impaire on dit qu'il accède à un mot non aligné ceci est dQ au fait que les deux octets du mot ne résident pas à la même adresse mot. Il faut donc 2 cycles mémoire pour lire le mot complet.

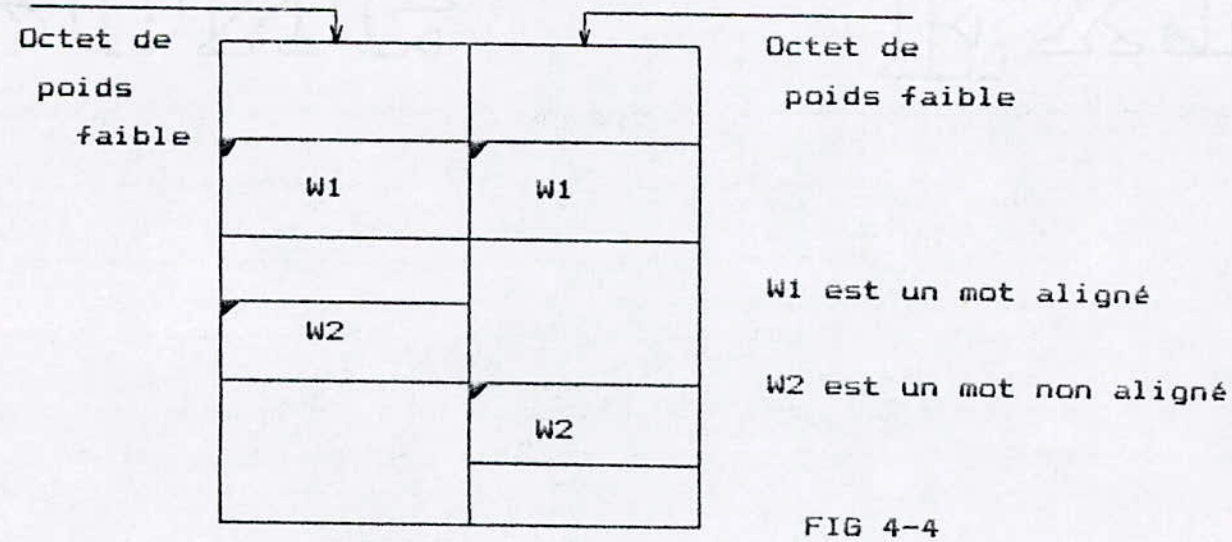
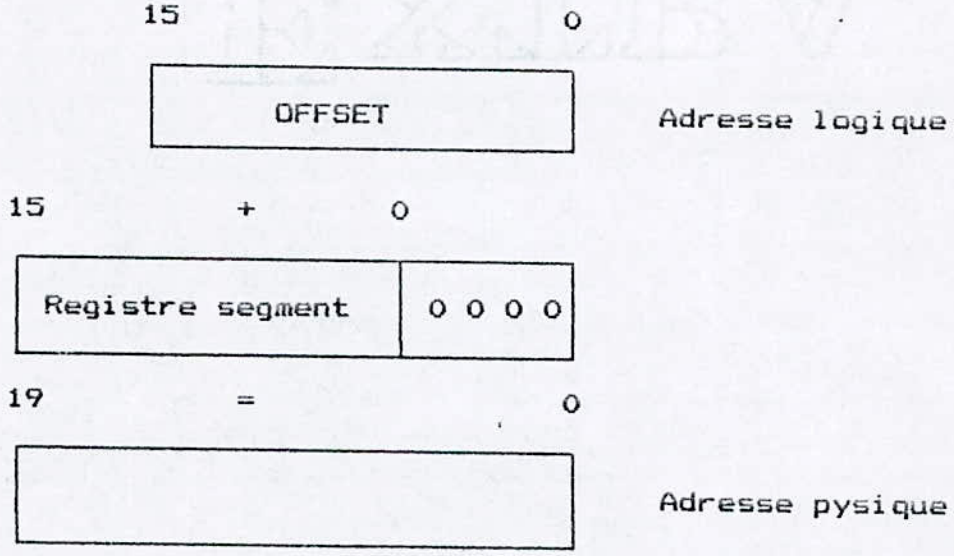


FIG 4-4

1-3-2-La Segmentation de l'éspace mémoire:

a-Adresse logique et adresse physique:

Le 8086 possède un bus adresse de 20 bits, cependant pour diminuer la longueur des instructions en limitant le champs adresse à 16 bits, l'éspace mémoire est divisé en N segments, les emplacements physiques en mémoire de 4 des N segments, c'est à dire les premieres adresses physiques des 4 segments, sont indiqués par le contenu des 4 registres segments.
L'adresse d'une position mémoire à l'intérieure d'un segment est l'adresse relative (déplacement ou offset) de cette position mémoire par rapport à la premiére position du segment.
Cette adresse relative, éxprimée sur 16 bits, est une adresse logique. L'adresse physique correspondante est calculée en ajoutant l'adresse logique à l'adresse de base du segment. Cette dèrnière est obtenue en multipliant par 16 le contenue du registre segment correspondant (ce qui revient à ajouter 4 zéros à droite du contenu du registre segment).



b-La gestion des registres segments:

Le C.P.U efféctue automatiquement, pour tout accès mémoire et grâce son circuit de géstion mémoire intégré, le calcul de l'adresse physique, il affèctera par défaut un registre segment pour tout accés mémoire.Mais,par programme et grâce à des pseudo_instructions du langage assembleur, il est possible de choisir l'un des registres segments CS,DS,SS,ES.
Il est à noter que les programmes qui ne mentionnent pas les registres segments sont relogable : voilà l'avantage le plus important des microprocesseurs 16 bits par rapport au 8 bits.

1-3-3-L'implantation de la pile en RAM:

Le segment pile courant est implanté en mémoire à partir de l'adresse de base contenue dans le registre segment. Il est à noter que la pile se développe vers les adresses décroissantes, l'écriture dans la pile est toujours précédée d'une décrémentation de deux unités du registre pointeur de pile.

2-PRESENTATION DU 80286:

2-1-ARCHITECTURE INTERNE:

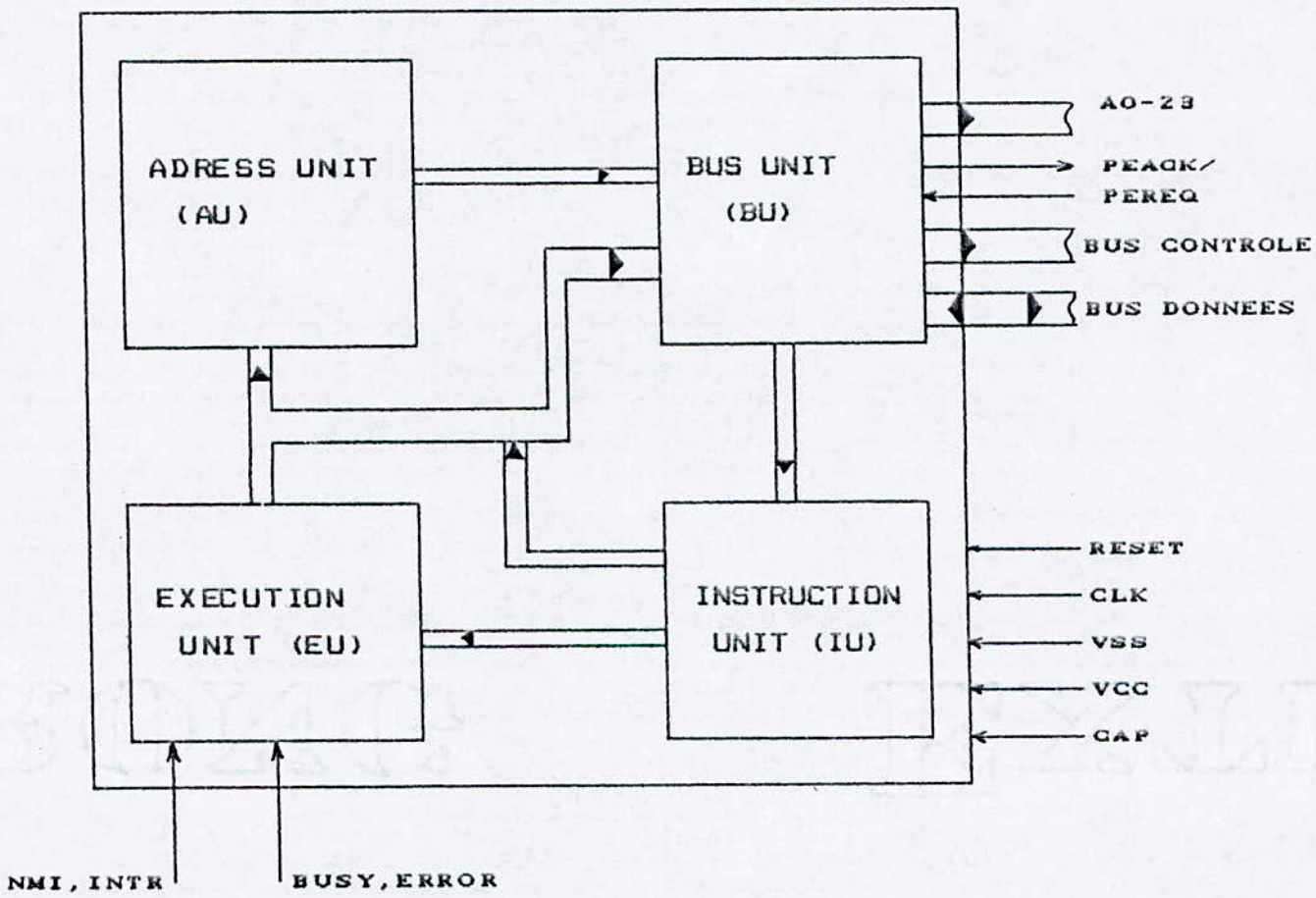
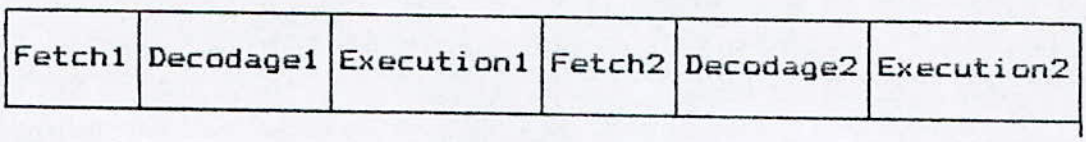


FIG 3-2-1

Le microprocesseur 80286 est constitué de 4 unités indépandantes qui travaillent en prallèl , cette structure est dite pipeline.

L'intérêt de cette structure parallèle apparait en figure 3-2 , le gain de temps est mis en évidence au niveau des éxécutions.

* Processeur à structure séquentielle:



* Processeur à structure parallèle:

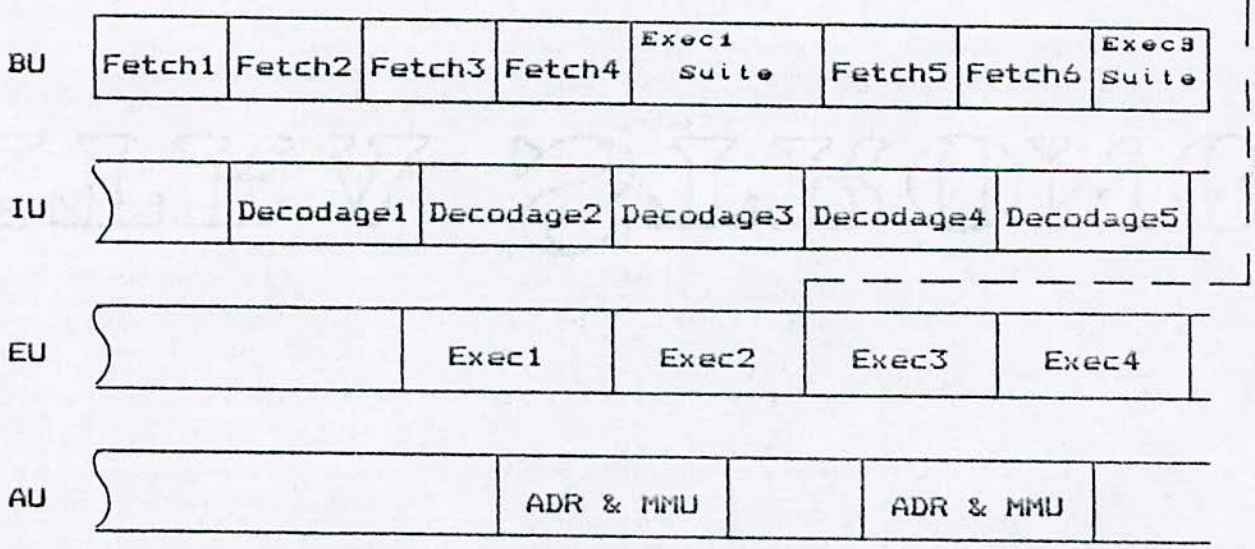


FIG 3-2-2

(Cas ou l'instruction 2 n'utilise pas l'unité bus)

2-1-1 L'Unité Bus: (U.B)

Cette unité gère les adresses, les données, les signaux de controle afin de réaliser les accès à la mémoire et aux E/S.

2-1-2 L'Unité d'instruction: (I.U)

Cette unité recoit les instructions de l'unité bus, les décodes et les places dans une file d'attente.

2-1-3 L'Unité d'éxécution: (E.U)

Les instructions de la file d'attente de l'unité d'instruction sont éxécutés dans cette unité. L'U.E utilise l'unité bus pour tous les transferts de données de/vers la mémoire (ou E/S).

2-1-4 L'Unité Adresse: (A.U)

Cette unité assure la gestion de la mémoire, la protection et la translation des adresses logiques en adresses physiques pour être utiliser par l'unité bus.

2-2-LES SIGNAUX DU 80286:

Le 80286 se présente dans un boitier de type pin gray array, il possède 68 Broches dont 63 sont utilisées.
Le bus adresse est un bus de 24 lignes, indépendant du bus données formé lui de 16 lignes.Il possède 2 entrées d'interruptions (NMI,INTR).

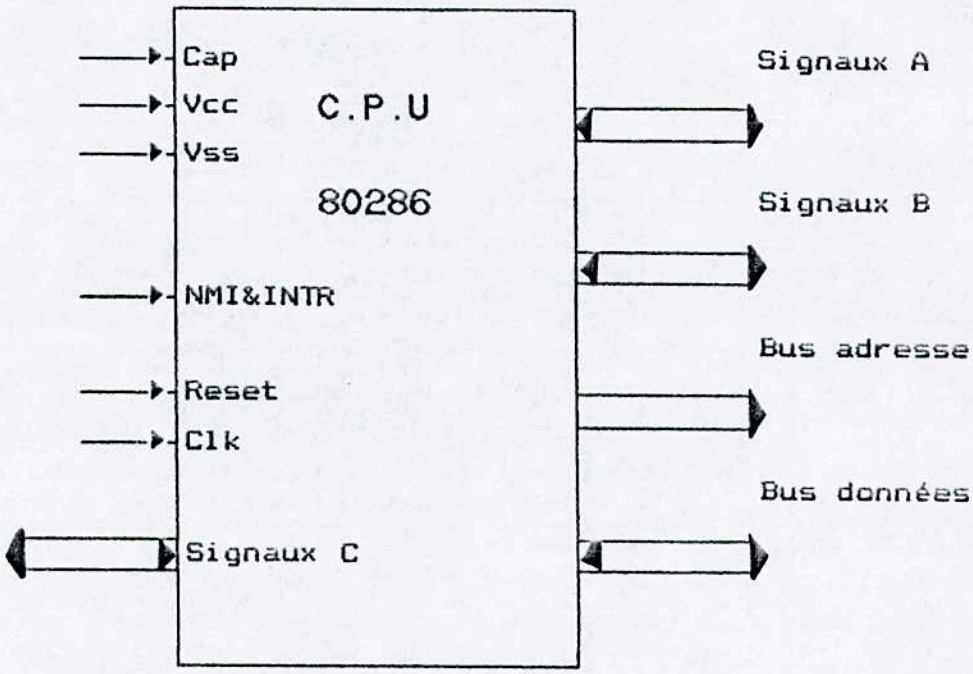


FIG 3-2-3 Signaux du 80286

Les signaux A sont au nombre de 6 et forment les signaux d'état et de définition des cycles bus, Les signaux B sont au nombres de 3 et forment les signaux d'attribution et de verrouillage du bus, les signaux C au nombre de 4 forment les signaux de controle du processeur d'éxtention.

A l'entrée Cap vient se brancher une capacité de filtrage dont la valeur est donnée par le concépteur (de l'ordre de 0.047 µf).

2-3-GESTION MEMOIRE ET PROTECTION :

Les mécanismes intégrés du 80286 qui le distinguent des autres familles de processeurs permettent de répondre à :

1-Etablir une protection entre les différentes entités résidants simultanément en mémoire (système d'éxploitation, tâche, segments).

2-Implanter et controler de facon éfficace l'allocation de la ressource mémoire physique.

** SEGMENTATION ET MODELE DE PROTECTION :

A-MODE REEL:

Le 80286 présente dans ce mode un fonctionnement similaire au 8086, si ce n'est sa vitesse d'éxécution qui en fait un "super 86". Un segment, dans ce mode, est caractérisé par sa base et une taille fixe de 64 KO, tout élément accédé en mémoire (instruction, pile, donnée) sera référencé dans le programme par une adresse logique à deux composantes 16 bits : Le sélécteur identifiant le segment et l'offset, adresse relative à l'intérieure du segment. Ceci sans aucune protection puisque quelque soit la combinaison sélécteur-offset peut être proposée sans guarantit de validité et d'utilisation correcte de l'information accèdée.

Le mode réel a deux intérêts :

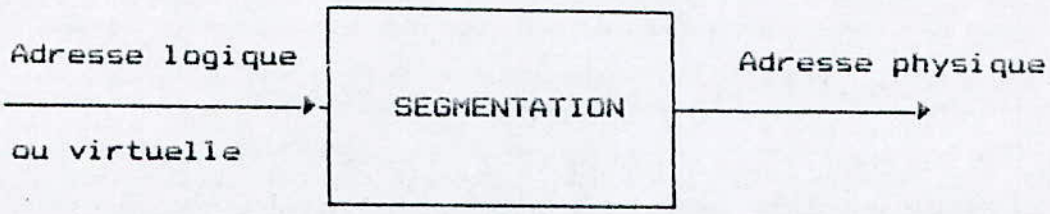
1- A l'initialisation le processeur est toujours en mode réel, pour préparer le mode protégé.

2- Le 80286 peut être utilisé en mode réel seulement, il offre une compatibilité totale avec le 8086 (C'est le cas de son utilisation sous MS-DOS).

B-MODE PROTEGE:

Le passage au mode protégé se fait par la mise à 1 du bit PE (Protected Enable) du registre MSW interne du 80286.
C'est le mécanisme d'adresse qui constitue la différence essentielle entre le mode réel et le mode protégé.
Dans le mode protégé on distingue 2 étapes pour l'adresse référencant un objet :

1- Adresse logique ou virtuelle.
2- Adresse physique (par segmentation).



B-1 LA SEGMENTATION :

B-1-1 ATTRIBUT D'UN SEGMENT :

En mode protégé un segment est caractérisé par :

1- Sa base.
2- Sa limite ou sa taille.
3- Ses droits d'acées:
   - Type de segment (éxécutable,accéssible en lécture, écriture).
   - Niveau de prévilège (0-3) et des indicateurs pour une géstion de mémoire virtuelle.

B-1-2 Déscripteurs et tables de déscripteurs:

Tout segment est définie par son déscripteur :

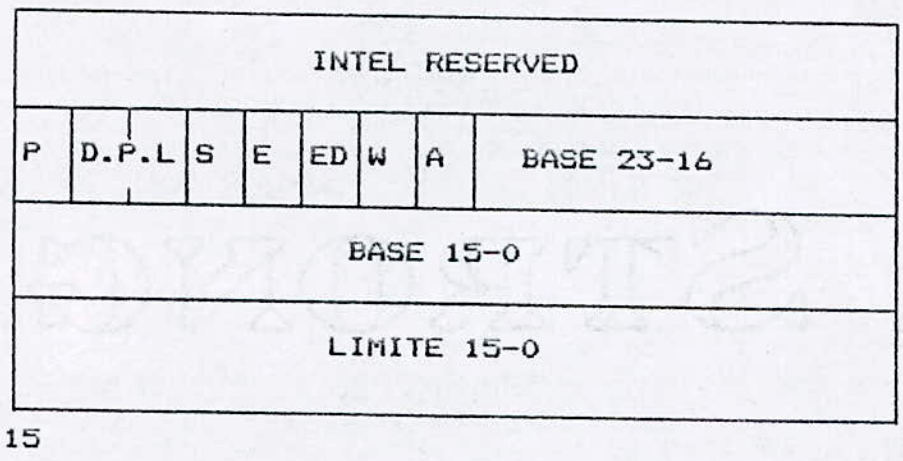


FIG 3-2-4 Format d'un déscripteur.

Champs type : Donne la nature du segment.

E = 0 Segment code.
E = 1 Segment données
ED =0 Le segment se développe de l'adresse basse vers la limite.
ED = 1 Le segment est de progression inverse.
W = 1 Le segment est accessible en écriture.
W = 0 Le segment n'est pas accessible en écriture.

Le Bit A: Est un indicateur, il est positioné à 1 automatiquement dés que le déscripteur est accédé.

Le Champs DPL: Identifie le niveau de prévilège du segment à accéder. (0 à 3), niveau zéro plus préviligié.

Le bit P: Bit de présence mis à 1 si le segment adressé par le déscripteur est en mémoire physique.

Base du segment: Adresse physique de début de segment.

Limite du segment: Dimension du segment (Max 64 KO).

Les déscripteurs segments sont résidants en mémoire dans des tables de déscripteurs : GDT (Global Table Descriptor) et LDT (local Table Descriptor).
Ces tables de déscripteurs sont elles même des segments, du type système et d'accès réservé au système d'éxploitation.
La GDT est pointée par le registre interne GDTR, le système implantera dans cette table les déscripteurs de segments qui doivent réster accessible quelque soit la tâche active.
La LDT est pointée par le registre interne LDTR, C'est une table local spécifique à la tâche en cours, tous changement de tâche provoquera un changement automatique de LDT,par modification de LDTR

2-4-NOTION DE TACHE ET PROTECTION :

Le concept de tâche permet la cohabitation simultanément de plusieurs programmes dans la mémoire du système.
Un contèxte est associé à chaque tâche et le système d'éxploitation doit guarantir l'isolation des contèxtes des différentes tâches.
Le contèxte de chaque tâche est définie par :

- L'état actuel des registres internes.
- Une LDT spécifique.
- Des pointeurs de piles pour les différents niveaux de prévilège.
- Un lien éventuel sur une autre tâche.
- Une chaine de bits éventuelle définissant les permissions

Ce contexte est décrit en mémoire dans un segment, de type système, TSS (Task State Segment).
La tâche active, unique, est pointée par le registre TR (Task Register).
La commutation de tâche se décompose en 2 phases :

1- Sauvegarde de l'état courant des registres du processeur dans le TSS en mémoire pointé par TR.

2- Activation d'une nouvelle tâche en modifiant TR.

Cette opération peut être réalisée par l'instruction "JMP" ou "CALL" ou sur interruption.

2-5-SYNTHESE DES CRITERES DE PROTECTION :

La protection agit sur 3 niveaux distincts:

1- Protection des fonctions système vis-à-vis des applications.

2- Isolation des tâches : effectué par l'affectation globale ou locale des segments sous la forme de table GDT et LDT .

3- Protection des applications vis-à-vis d'elles mêmes réalisé par la limite et le type de segment.

2-6-LES INTERRUPTIONS :

La prise en compte des interruptions en mode réel s'effectue éxactement de la même manière que dans le cas du 8086.
En mode protégé, une source d'interruption est toujours identifiée par son type mais dans ce cas la table consultée (IDT: Interrupt Descriptor Table) sera interprtée de manière différente que dans le cas du 8086 où cette table contient les vécteurs d'interruptions.
Dans la IDT est associé un déscripteur à chaque type d'interruptions
Comme la GDT, la IDT est une table Globale, elle peut contenir un maximum de 256 déscripteurs. Elle est pointée par le registre interne IDTR, elle peut résider n'importe où en mémoire physique.
Le registre IDTR contient la base et la limite de la IDT et sera

initialisé en mode reel.

2-7-LES INSTRUCTIONS 80286 :

Le jeu d'instruction du 80286 se compose de celui du 8086 plus un bon nombre d'instructions dont la plupart sont déstinées uniquement au mode réel et protégé tandisque d'autres sont déstinées uniquement au mode protégé.

2-7-1 Les instructions 80286 utilisées dans le mode reel et protégé:

LGDT	Load GDTR.
LIDTR	Load IDTR.
LMSW	Load MSW.
PUSH	Push immediate.
PUSHA	Push All.
POPA	Pop All.
IMUL	Integer immediate multiply.
SHIFT/ROTATE INSTRUCTION	REGISTER/Memory by count.
INS	Input byte/word from DX port.
OUTS	Output byte/word from dx port.
INS	Input String.
OUTS	Output String.
ENTER	Enter procedure.
Leave	Leave procedure.
BOUND	Detect value out of range.
CLTS	Clear Task Switched flag.
SGDT	Store Global Descriptor Table register.
SIDT	Store Interrupt Descriptor Table register.
SMSW	Store Machine Statu word.

2-7-2 Instructions 80286 utilisées uniquement dans le mode protégé:

LLDT	Load Local Descriptor table register from register memory
SLDT	Store Local Descriptor table register to register/memory
LTR	Load Local Task register from register/memory.
STR	Store Task register to register/memory.
LAR	Load acces rights from register/memory.
LSL	Load Segment Limit from register/memory.
ARPL	Adjust Requested Previlege Level from register/memory.
VERR	Verify Read access.
VERW	Verify write access.

CHAPITRE 4

DESCRIPTION DE LA CARTE D'ANALYSE LOGIQUE PROPOSEE

1-ETUDE DESCRIPTIVE DE L'ANALYSEUR LOGIQUE PROPOSE:

L'analyseur logique proposé assure la mémorisation des informations recueillies par la sonde sur la carte prototype en donnant oreille fine au signal de déclenchement,ce dernier est conditionné par la réalisation de l'évènement logique.Il assure par ailleur l'analyse de ces informations pour les rendre sous une forme éxploitable à l'opérateur.

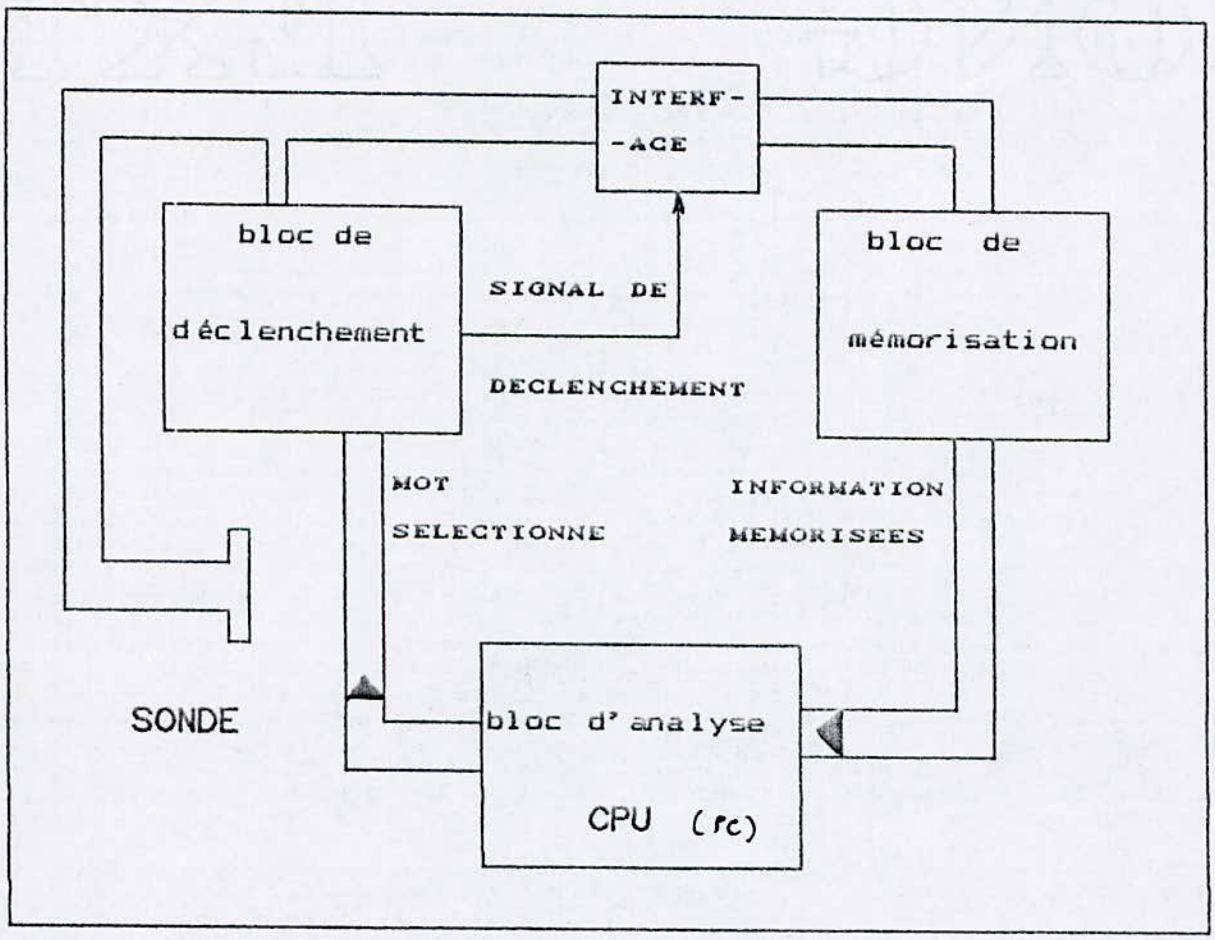


FIG. 4. 1:shéma synoptique général de la carte d'analyse logique

Au lieu d'avoir les blocs de déclenchements et de mémorisation distincts,on a pensé à l'utilisation d'un CONTROLEUR DE DMA LE 80258 qui permet d'éviter d'avoir le bloc de délenchement,çela aurait été intéressant puisque le 80258 permet d'être programmé,ainsi il comparera à chaque fois le mot préséléctionné à la donnée qui lui est offerte et si le test est bon il déclenche la mémorisation, seulement, le problème réside d'une part au niveau de la fréquence car le controleur DMA consomme quatre périodes d'horloge pour faire le transfert d'un octet,donc si on veut qu'il pilote un 80286 à 16MHZ il doit avoir une fréquence de l'ordre de 100 MHZ ce qui n'éxiste pas sur le marché; d'autre part dans le mode 2 et 3 ce controleur de DMA doit aprés réalisation de l'évènement charger ses registres de la mémoire; pour avoir le nouveau contexte de travail,il pèrdera un temps fou, et les allentours du point critique ne seront pas mémorisés.

1-1-BLOC DE DECLENCHEMENT:

Ce bloc doit assurer la reconaissance de l'évènement logique et le signaler quand il se produit.
Le mot séléctionné,est choisie par l'opérateur sur n'importe quel combinaison entre le bus controle,le bus données ou le bus d'adresse un masque a été prévu à cet effet pour choisir la combinaison séléctionnée relatif au mot de déclenchement.

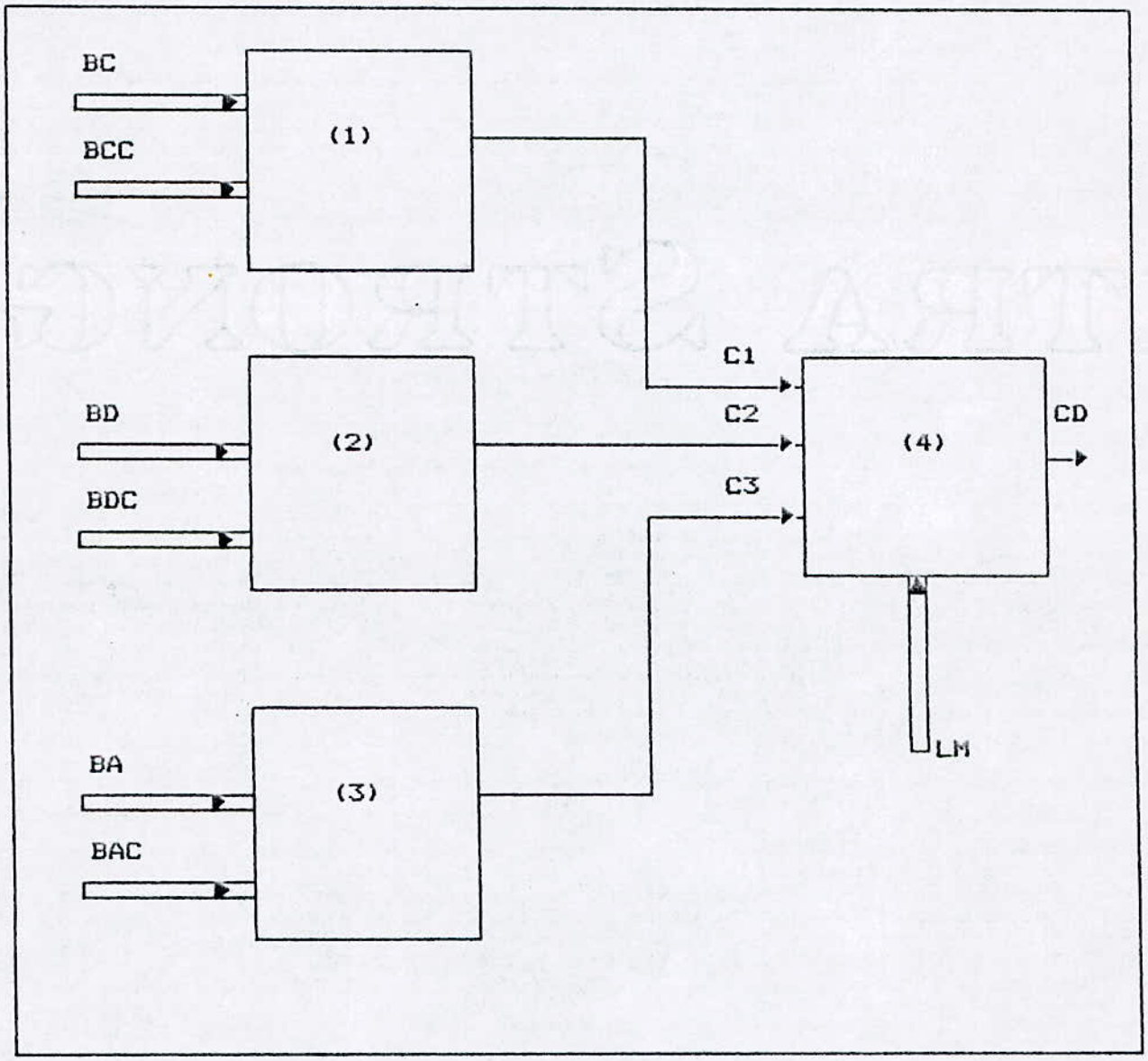


FIG.4.2.schéma synoptique du reconaisseur de l'évènement logique.

(1) Reconaisseur de l'évènement logique(R.E.L) relatif au bus controle.

(2) R.E.L relatif au bus Données.

(3) R.E.L relatif au bus Adresse.

(4) Masque.

BA: Bus Adresse ,BAC:Bus Adresse Choisie ,BD:Bus de Données

BC: Bus de Controle, LM : ligne de commande.

La reconaissance de l'évènement logique s'établit par une comparaison des signaux logiques d'entrées avec le mot préséléctionné.

On propose l'utilisation de COMPARATEURS qui assurent cette fonction.La comparaison se fait au niveau du bus controle,de données et d'adresse indépendament pour pouvoir choisir la combinaisondu combinaison du masque.

Le choix du mot est réalisé en utilisant des tampons pour verrouiller la donnée,ils sont adressés comme le montre la figure suivante par un décodeur 74LS138 à partir d'une combinaison des trois lignes d'adresses A_1,A_2,A_3.
M/IO indique, quand il est bas,que l'adressage est un adressage d'entée/sortie et non un adressage mémoire,alors que DEN signale l'éxistance d'une donnée valide sur le bus.

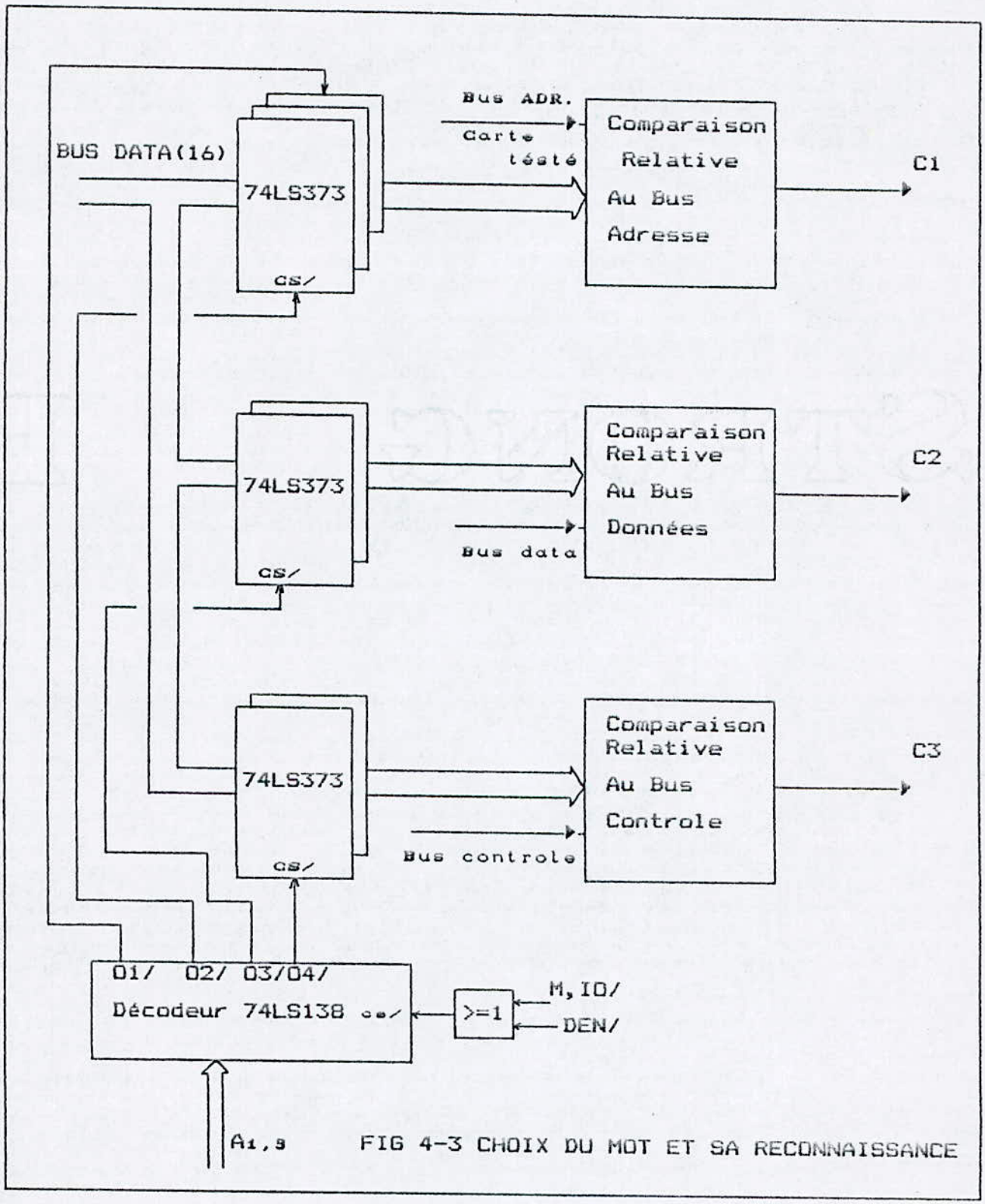


FIG 4-3 CHOIX DU MOT ET SA RECONNAISSANCE

Les 24 lignes d'adresses, 16 lignes de données et 16 lignes de controle vont alors être comparés avec le contenu des verrous qui représente le mot logique ou l'évènement attendu, par les comparateurs .
Les signaux C1,C2,C3 doivent subir un masquage qui permettera la séléction de la combinaison voulu de l'événement .

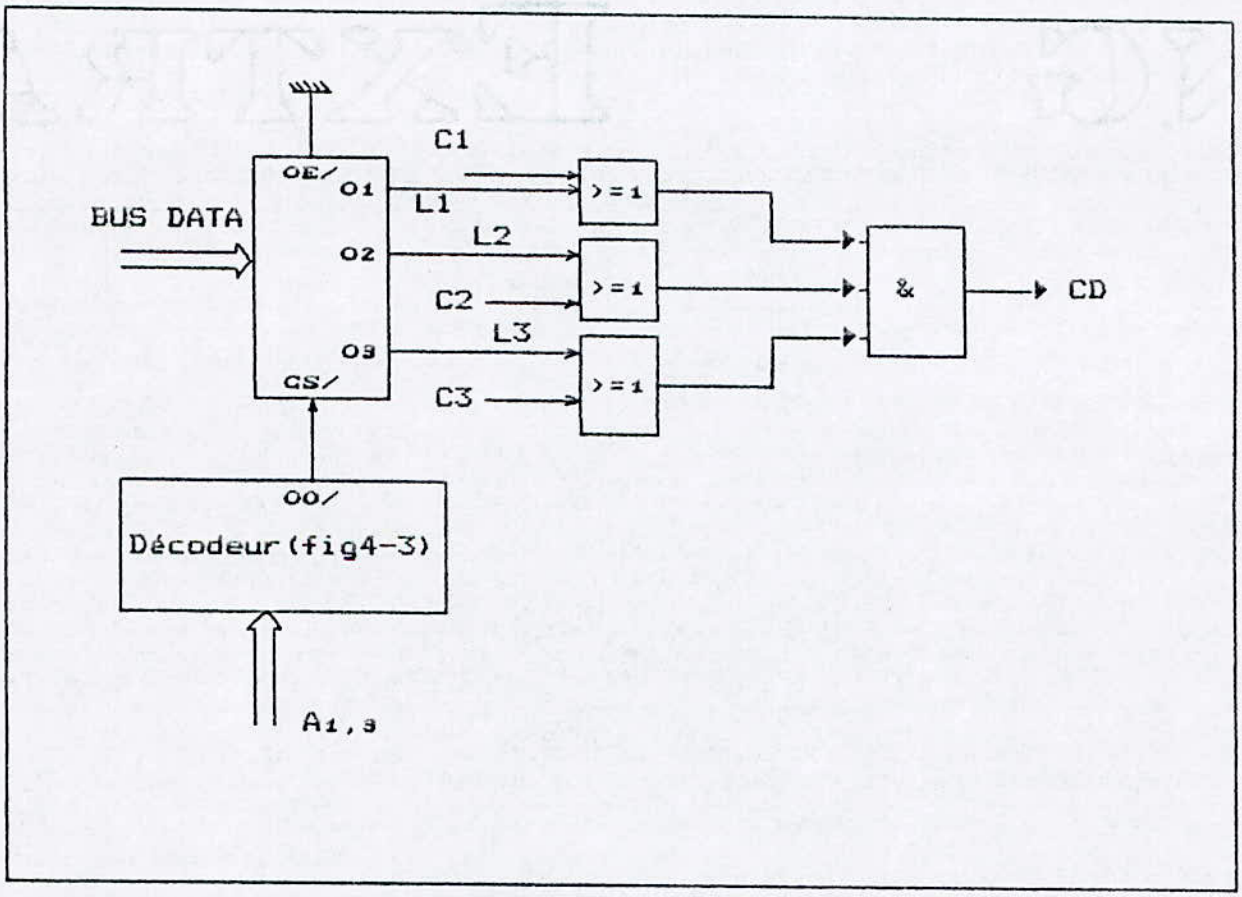


FIG 4-4 MASQUE

L3	L2	L1	COMBINAISON CHOISIE
0	0	0	bus controle-donnée-adresse
0	0	1	bus controle-donnée
0	1	0	bus controle-adresse
0	1	1	bus controle
1	0	0	bus adresse-donnée
1	0	1	bus donnée
1	1	0	bus adresse
1	1	1	aucune combinaison

TABLE.1.sens de L1,L2 et L3.

CD représente la sortie de ce bloc qui signalera , la reconaissance du mot séléctionné (l'évènement attendu), il represente le signal de déclenchement.

1-2-BLOC DE MEMORISATION:

La mémorisation des évènements logiques parvenues de la carte sous test via la sonde est assurée par l'adjonction de compteurs synchronisés par le signal de déclenchement, et, de mémoires rapides à temps d'accés trés court (de l'ordre de 10ns) puisque la fréquence de mémorisation atteint les 25MHZ pour suivre les variations trés rapides des signaux d'entrées.
On a alors utilisé, à cet effet, deux compteurs binaires, qui travaillent non simultanément suivant le mode de mémorisation choisie, commandés par deux (2) lignes programmables P1 et P2.

P1	P2	MODE
1	0	Mode 1 : Prédéclenchement
0	1	Mode 2 : Post-Déclenchement
1	1	Mode 3 : Déclenchement
0	0	Réservé à la phase d'analyse

Table.2.Sens de P1,P2.

1-2-1-MODE 1: P1=1 et P2=0

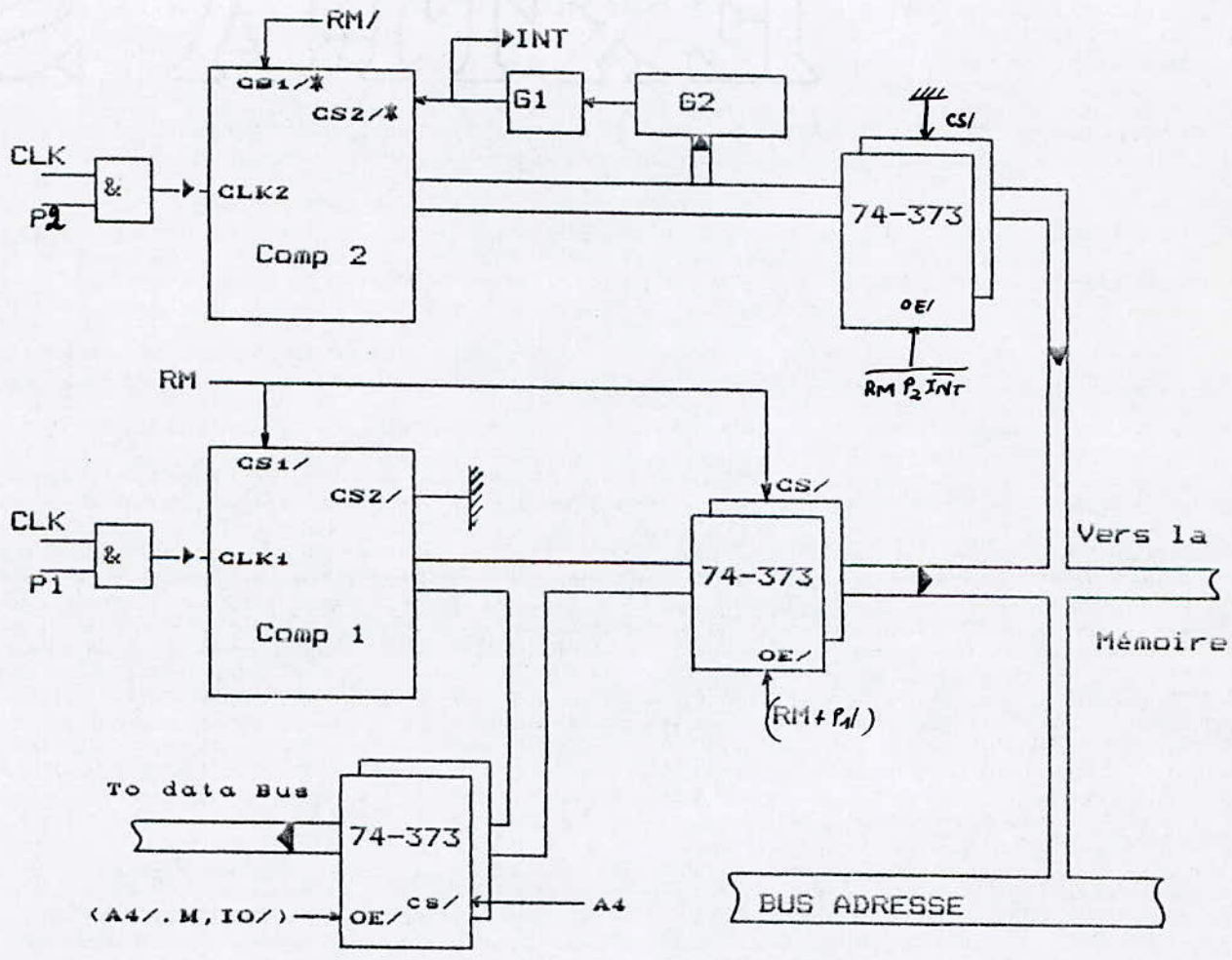
La mémorisation se fait avant l'avènement de l'évènement, la reconnaissance du mot préséléctionné arrête la mémorisation commencée par le compteur1 qui garde la dérnière adresse de mémorisation, lue,par la suite par le microprocesseur à travers un verrou.Le compteur2 dans ce mode étant inactif puisquu'il ne recoit aucun top d'horloge.

1-2-2-MODE 2: P1=0 et P2=1

La mémorisation dans ce cas ne se déclenche qu'aprés que le mot n'ait été détécté,dans ce cas le compteur2 n'est actif qu'aprés réception du signal de déclenchement. La mémorisation annoncée se termine lorsque la mémoire arrive à sa fin.dans ce mode le compteur1 est inactif.

1-2-3-MODE 3: P1=1 et P2=1

Dans ce cas la mémorisation s'éffectue avant l'avènement de l'événement,la reconaissance du mot préséléctionné arrête la mémorisation, commencée par le compteur1 devenu inactif, pendant que le compteur2 se valide et commence le comptage pour mémoriser les informations survenues aprés l'évènement dans une mémoire d'éxtension qui se valide aussitot que le mot soit reconnue.



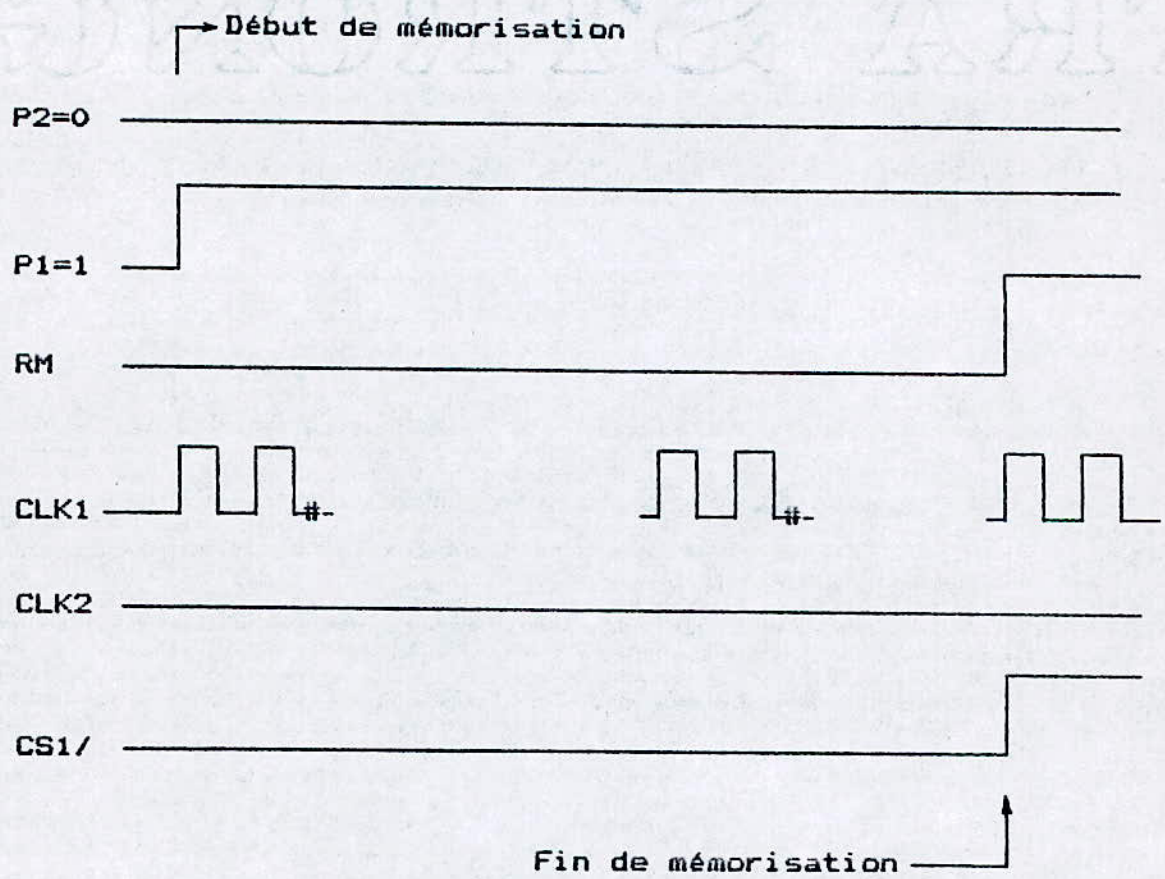
Remarque: G1 = Bistable (bascule RS).

G2 = Décode la valeur max atteinte par Comp2.

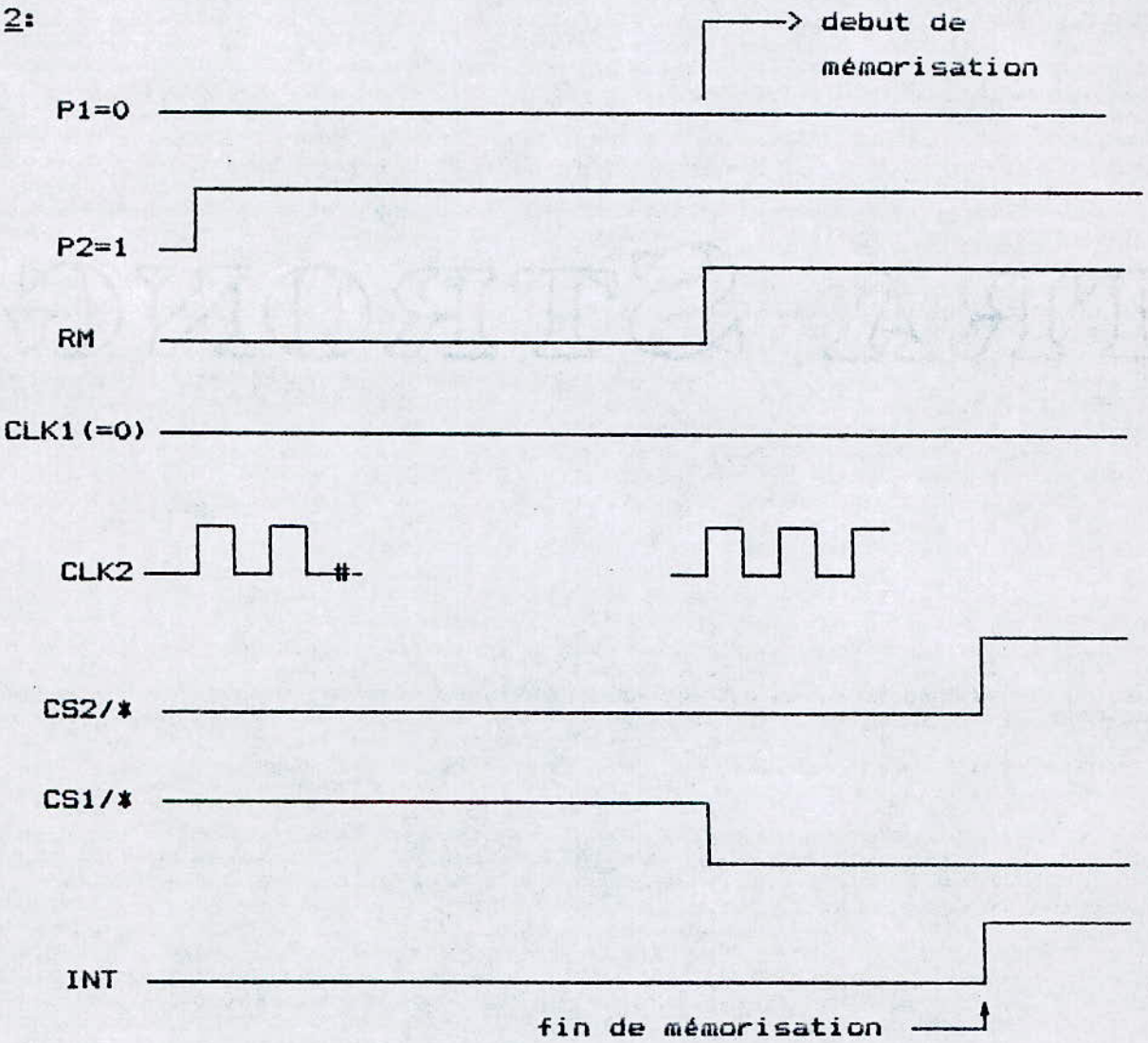
RM est le signal CD passé à travers un bistable.

LES CHRONOGRAMMES

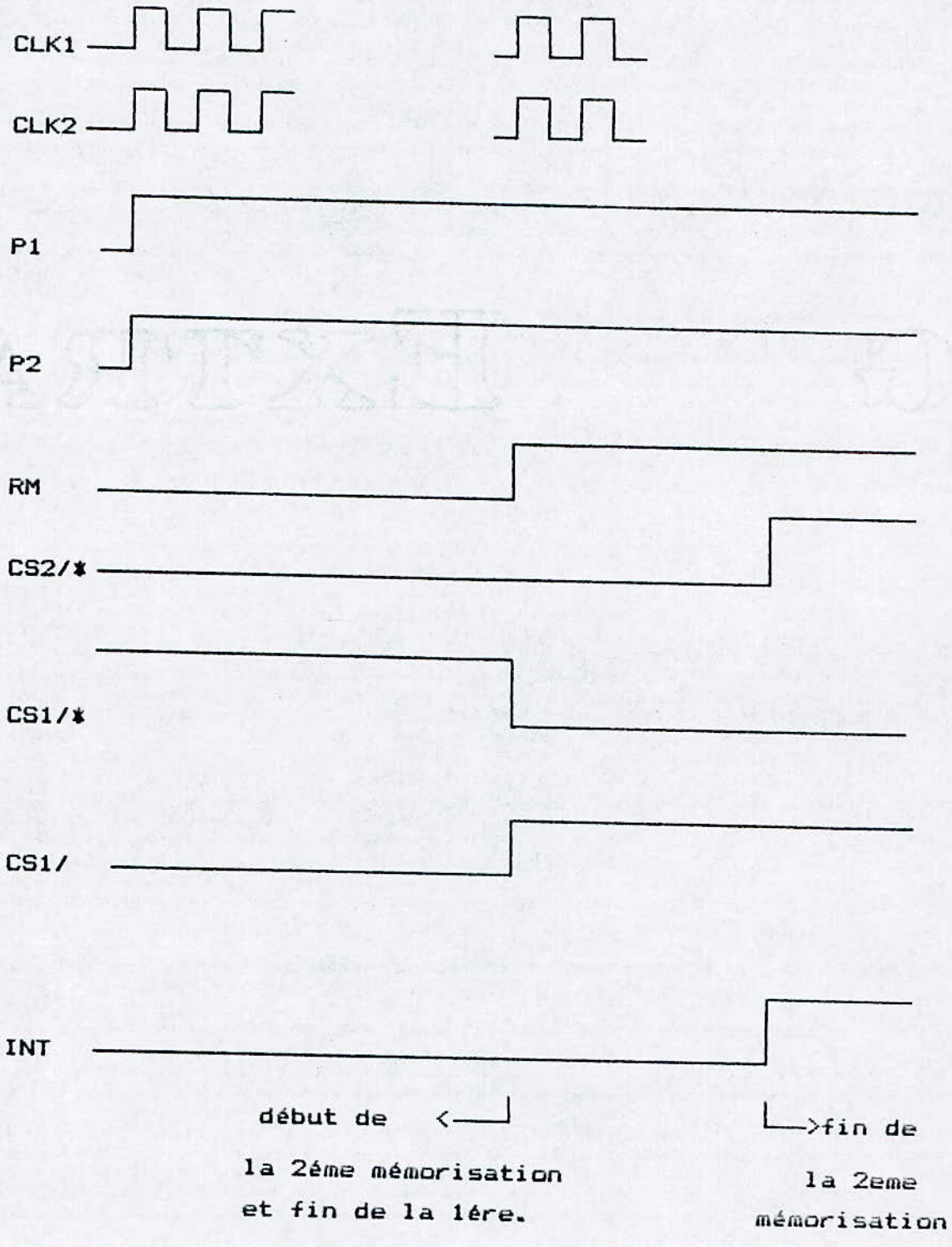
Mode 1:



Mode 2:


debut de
mémorisation
P1=0
P2=1
RM
CLK1 (=0)
CLK2
CS2/*
CS1/*
INT
fin de mémorisation

Mode 3:


CLK1
CLK2
P1
P2
RM
CS2/*
CS1/*
CS1/
INT
début de
la 2éme mémorisation
et fin de la 1ére.
fin de
la 2eme
mémorisation

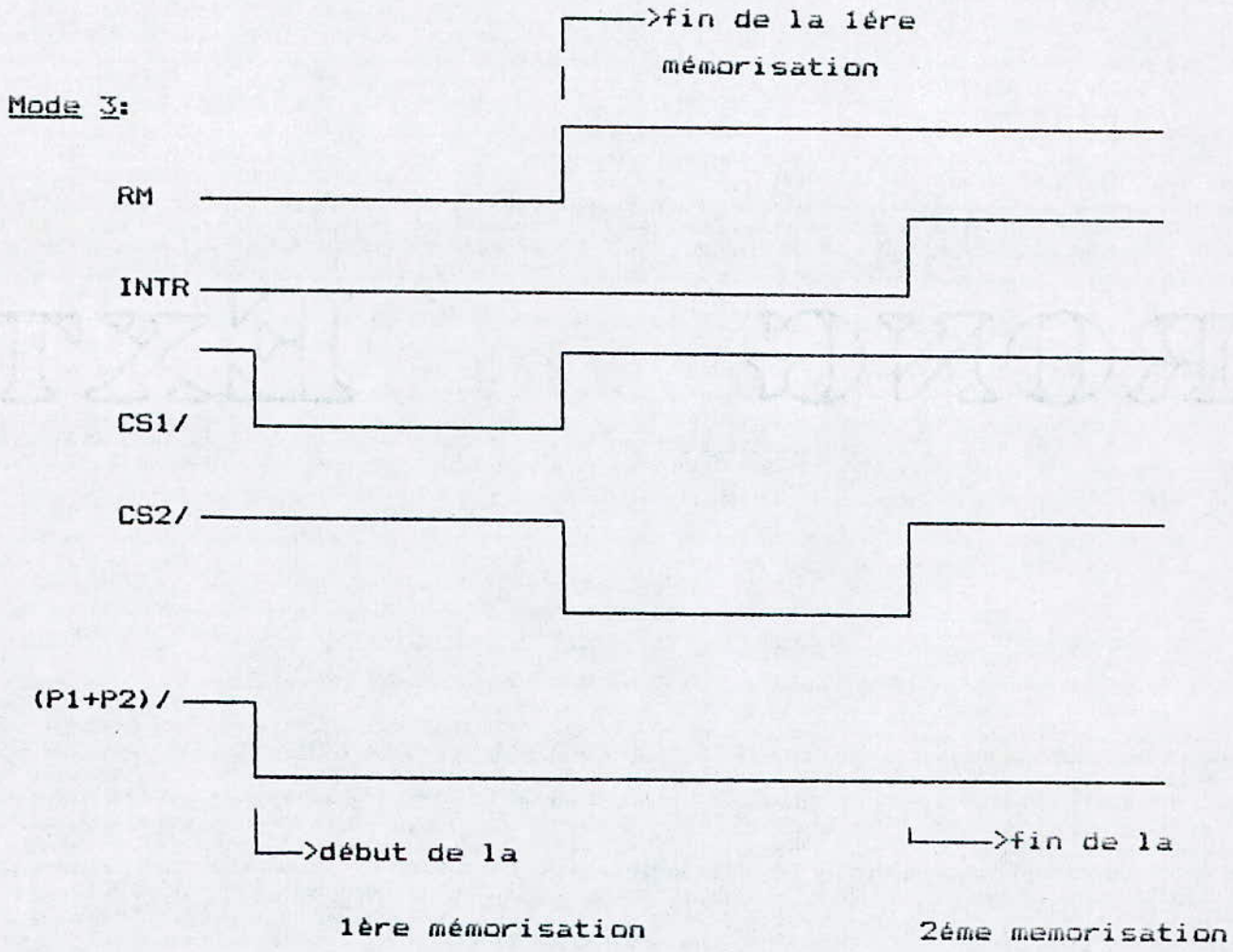
2-INTERFACAGE DES MEMOIRES:

Les mémoires doivent être utilisées de telle façon que pendant l'étape de collècte d'informations,la mémorisation des 56 signaux d'entrées doit s'effèctuer simultanément .
Ainsi la mémoire doit être validée pendant toute cette opération
Pendant l'étape de l'analyse ,le microprocesseur ne peut accéder à plus d'un mot de 2 octets, et puisque nos mots sont de 7 octets, alors on doit utiliser des verrous à double sens (TRANSEIVERS) qui font que l'accés ne se fait jamais à plus de 2 octets simultanément.

2-1-SELECTION DES BOITIERS:

La séléction de la mémoire d'éxtention ne se fait qu'aprés la fin de la première mémorisation et doit être non valide en fin de mémorisation dans le mode 3 et 2. Dans le mode 1 cette mémoire n'est pas valide puisqu'elle ne sera pas utilisée.
Les chronogrammes détaillées seront éxposés sur la page suivante.

LES CHRONOGRAMMES



RM: Signal de reconnaissance du mot.

INTR: Signal d'interruption envoyé au CPU indiquant la fin de mémorisation.

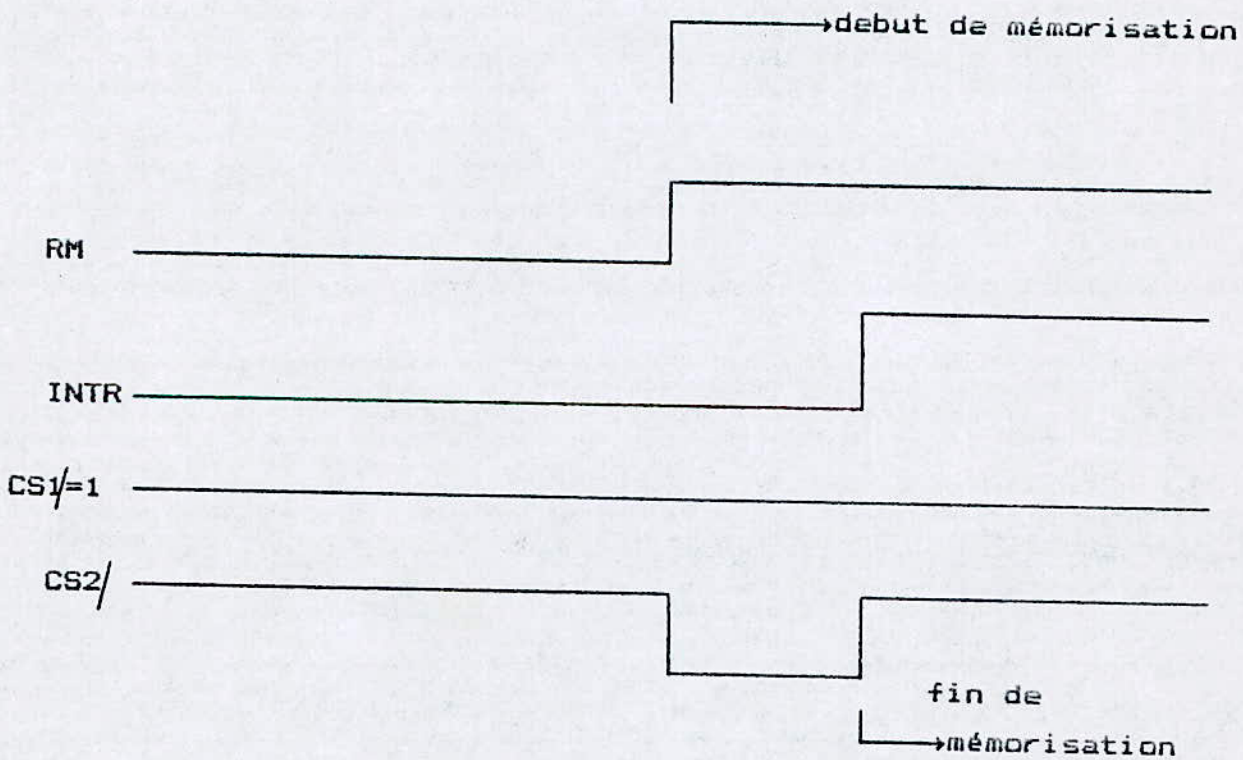
CS1: Chip select de la mémoire principale.

CS2: Chip select de la mémoire d'éxtension.

CS1/= RM + (P1+P2)/ ——————>En mode Analyse (P1+P2)/=1 et donc ses mémoires ne seront validées que par le CPU

CS2/= RM/.INTR/ +INTR.RM +(P1+P2)/

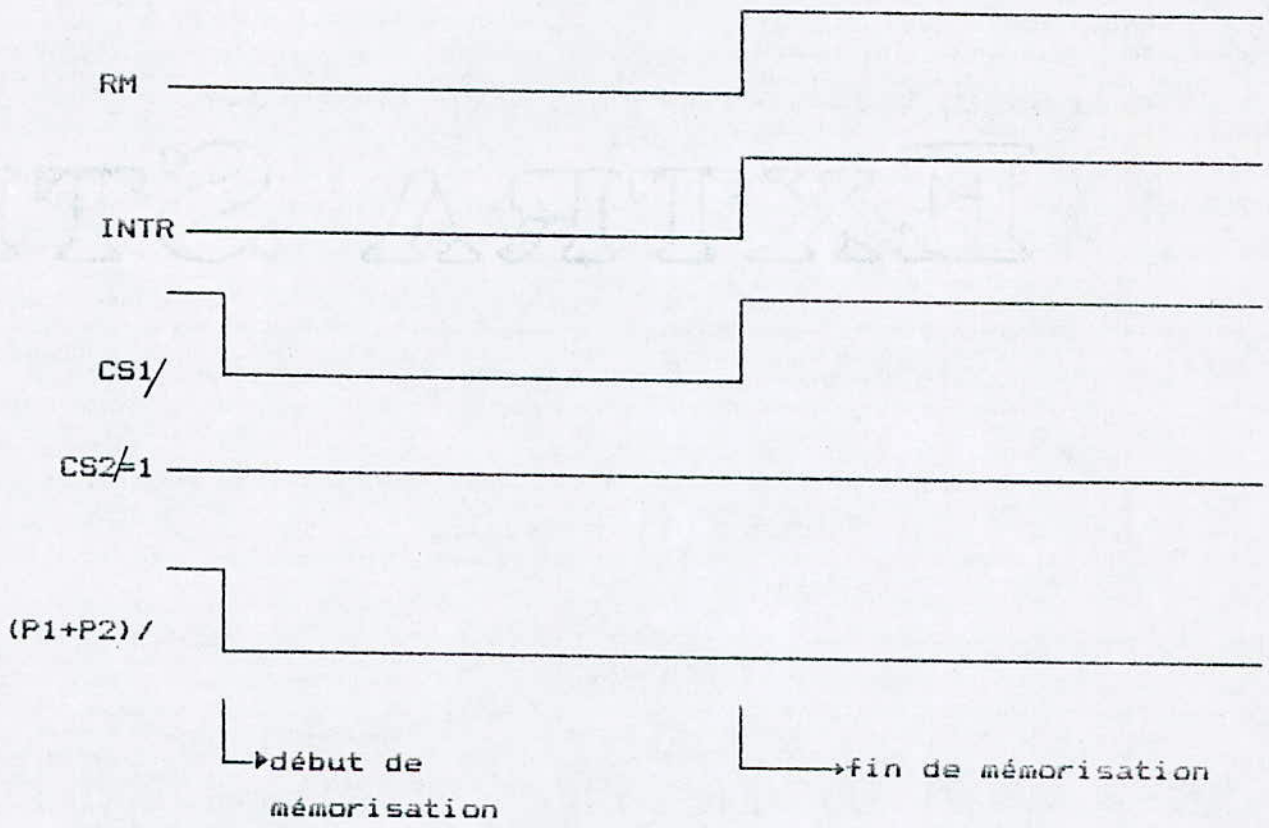
MODE 2:



CS2/= RM/.INTR/ + INTR.RM + (P1+P2)/

CS1/= P1/.P2 + (P1+P2)/ ; P1/P2 : Mode 2

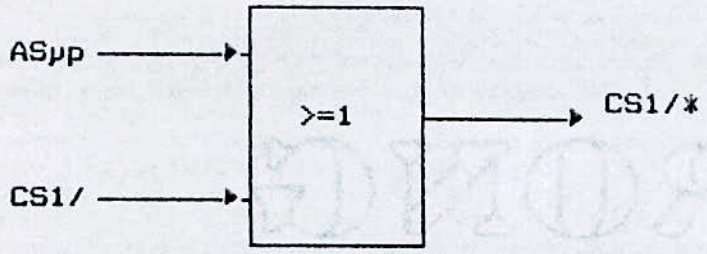
Mode 1:



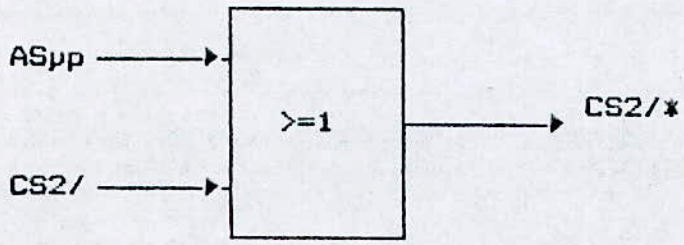
CS1/= RM + (P1+P2)/

CS2/= RM/.INTR/ +INTR.RM + (P1+P2)/

Sur ce qui à précéder il en découle:



CS1/= RM +P1/P2+(P1+P2)/



a-signal write:

Le signal de lécture est activé pendant toute l'étape de mémorisation, pour permettre la saisie d'informations.Il sera absent quand la mémorisation serait finie.

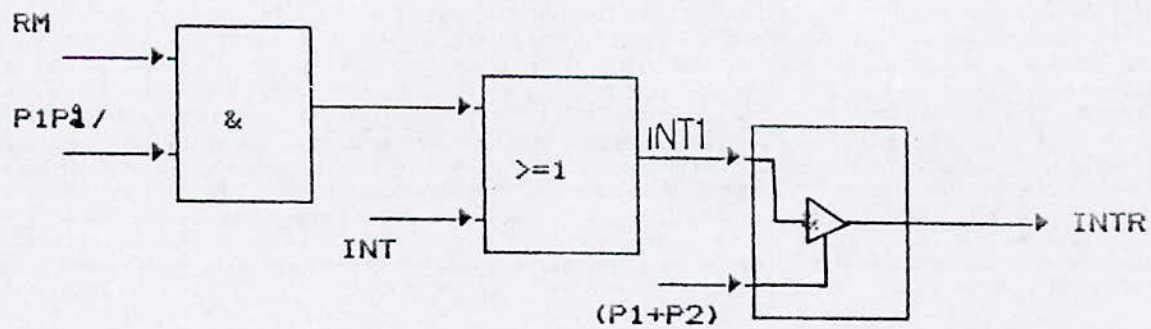
2-2-BLOC D'ANALYSE:

Aprés avoir positionner le mot de préséléction,les masques et bites de commande P1 et P2,le microprocesseur entre dans un état HALT ou il attend une interruption et se déconnècte du bus systéme.

P1 et P2 doivent donc étre retarder le temps qu'il faut à l'éxtraction et l'éxécution de cette instruction (HLT) sinon il y aurait conflit sur le bus entre le microprocesseur d'une part voulant éxécuter cette instruction et le compteur qui commence la mémorisation .

Il est à noter qu' avant l'éxécution de l'instruction il faut valider les interruptions par CLI ,sinon le microprocesseur rentrera dans un état dit de MORT,il ne pourra sortir de cet état que par un reset .

Aprés la fin de mémorisation un signal demande d'intérruption est générer par le bloc mémorisation vers le CPU pour indiquer la fin de cette étape.



* P1+P2 = (P1/.P2/)/

* P1/.P2/: Représente l'état d'analyse, il sera positionné à 1 par

le CPU juste aprés la fin de mémorisation ainsi la pin INTR ne sera plus actif.

* RM.P1.P2/ : Représente une fin de mémorisation dans le MODE 1

* INT : C'est le signal CS2 du compteur2 il indique la fin de mémorisation dans le mode 2 et 3 quand il est à l'état haut.

CHAPITRE 5

DESCRIPTION DU LOGICIEL ELABORE

INTRODUCTION:

L'élaboration du logiciel d'émulation nécessite une connaissance trés approfondie de la structure des instructions 80286, modes d'adressage, ainsi que les éspaces mémoires et registres utilisés par chaque instruction.
En plus, l'intégration de l'émulateur dans l'environnement PC éxige une maitrise des techniques de chargement des programmes dans la RAM et l'utilisation des interruptions DOS appropriées pour éxécuter ce dérnier en mode pas à pas, ainsi, une connaissance approfondie sur les programmes EXE et COM est nécessaire.

Nous aborderons alors les structures des instructions 286 et nous décrirons les programmes EXE et COM avant la déscriptions du logiciel d'émulation élaboré.
Il est à remarquer que les interruptions DOS les plus importantes utilisées lors du développement de notre logiciel sont mentionnées en Annexe.

5-1 STRUCTURE DES INSTRUCTIONS 80286 :

Le jeux d'instruction 80286 est concue à partir de deux types d'instructions:

1-Les instructions 80286 mode reel: Qui regroupent les instructions 8086 modifiées et les instructions 80286.

2-Les instructions 80286 mode protégé.

REMARQUE:
Il est à remarquer que la structure d'une instruction nous permet de connaitre d'avance quelles sont les registres segments affectés par défaut à cette instruction.

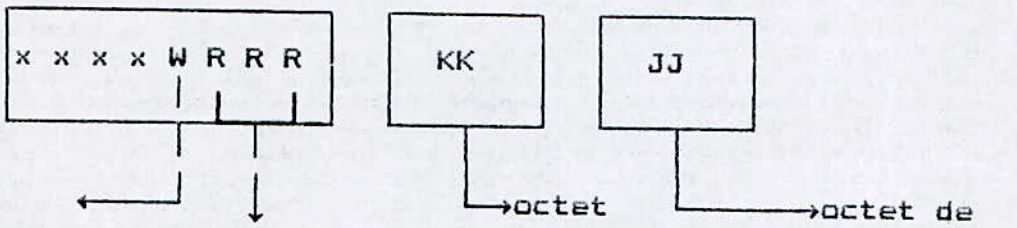
5-1-1-LES INSTRUCTIONS 80286 MODE REEL:

La structure du code objet de l'instruction 80286 varie d'une instruction à l'autre, cela est dû à la fonction de l'instruction, au mode d'adressage séléctionné et les espaces registres ou mémoires utilisés.
Il existe six modes d'adressages:

5-1-1-1-ADRESSAGE IMMEDIAT:

Consiste à charger une valeur immédiate dans un registre ou une mémoire. Les structures d'instructions possibles sont:

1-



w=0
opération
sur 8 bits
w=1
opération
sur 16 bits

spécifie
le registre
opération

octet
de poids
faible de
l'opérande

octet de
poids
fort de
l'opérande
si w=1

r r r	w=0	w=1
0 0 0	AL	AX
0 0 1	CL	CX
0 1 0	BL	BX
0 1 1	DL	DX
1 0 0	AH	SP
1 0 1	CH	BP
1 1 0	BH	SI
1 1 1	DH	DI

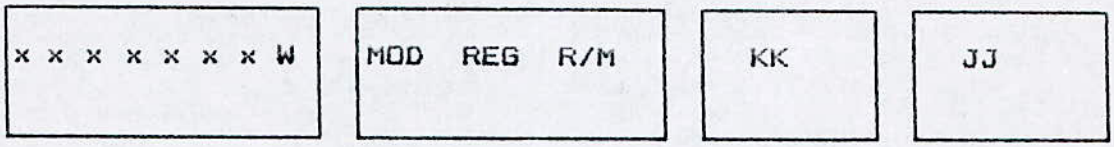
EXEMPLE:

MOV reg,data

reg: registre

data:donnée immédiate

2-



CODE DE LA FONCTION

EXEMPLE:

MOV mem/reg ,data

REMARQUE:

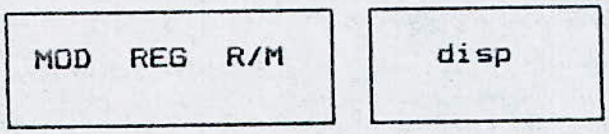
Cette remarque porte sur le deuxiéme octet de la structure précédente chaque terme a une signification relative au mode d'adressage et l'éspace déstination.

MOD:

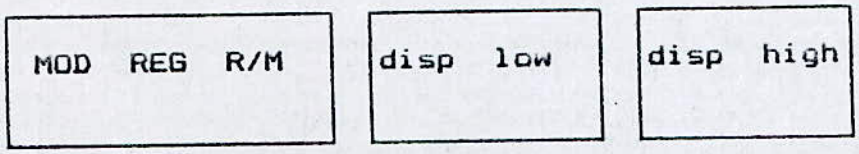
Ce sont deux bits qui sont utilisés lors de la distinction entre l'adressage mémoire et l'adressage registre,et dans le cas de l'adressage mémoire,ils spécifient les octets de déplacement.

mod = 00 ——————→adressage mémoire sans déplacement.

01 ——————→adressage mémoire avec déplacement sur 1 octet.



10 ——————→adressage mémoire avec déplacement sur 2 octets.

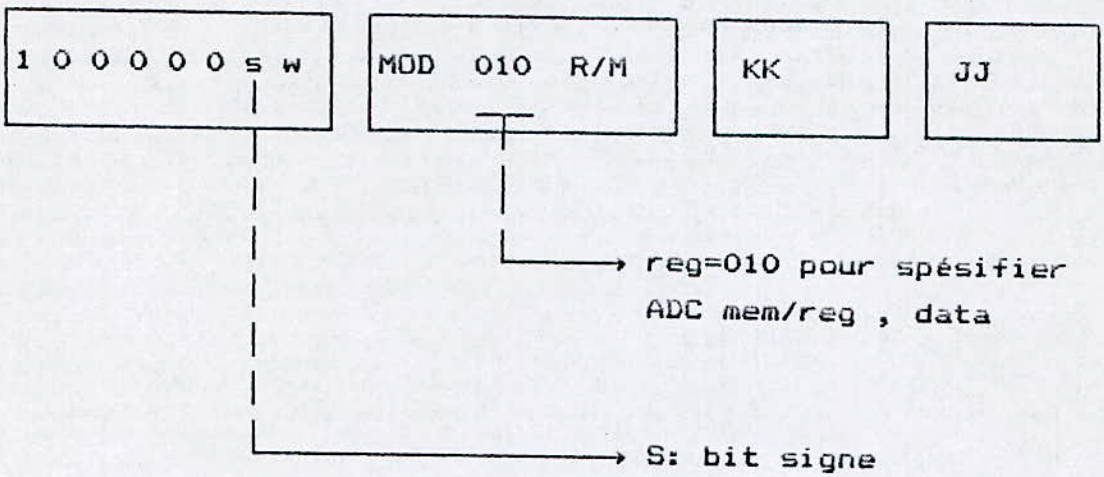


REG:

Utilisée avec w, REG séléctionne le registre utilisé dans l'opération spésifiée par la fonction.

REG peut être utiliser comme éxtension du code opération pour spécifier l'instruction désirée. C'est le cas de l'instruction ADC par éxemple.

ADC mem/reg , data



si w=0 ce bit S est ignoré
si w=1 alors:
si s=0 tous les 16bits du data seront présents
si s=1 seulement les 8 bits de poids faible seront présent

REG	W=0	W=1
0 0 0	AL	AX
0 0 1	CL	CX
0 1 0	DL	DX
0 1 1	BL	BX
1 0 0	AH	SP
1 0 1	CH	DP
1 1 0	DH	SI
1 1 1	BH	DI

R/M:

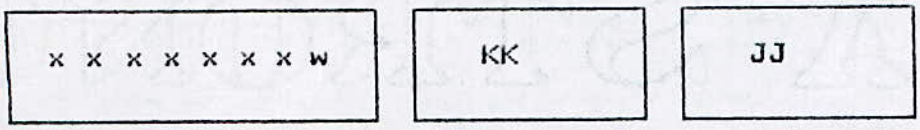
Spécifie le mode d'adressage en conjonction avec MOD:

R/M	MOD-00	MOD-01	MOD-10	MOD-11	
				W=0	W=1
000	BX + SI	BX +SI+DISP(b)	BX +SI+DISP(b)	AL	AX
001	BX + DI	BX +DI+DISP(b)	BX +DI+DISP(b)	CL	CX
010	BP + SI	BP +SI+DISP(b)	BP +SI+DISP(b)	DL	DX
011	BP + DI	BP +DI+DISP(b)	BP +DI+DISP(b)	BL	BX
100	SI	SI + DISP(b)	SI + DISP(b)	AH	SP
101	DI	DI + DISP(b)	DI + DISP(b)	CH	BP
110	DIRCT. ADR	BP + DISP(b)	BP + DISP(b)	DH	SI
111	BX	BX + DISP(b)	BX + DISP(b)	BH	DI

5-1-1-2-ADRESSAGE REGISTRE:

Le contenu d'un registre doit être écrit dans un registre ou une case mémoire.
L'architècture de l'instruction dans ce mode est donnée pa les représentation suivante:

1-

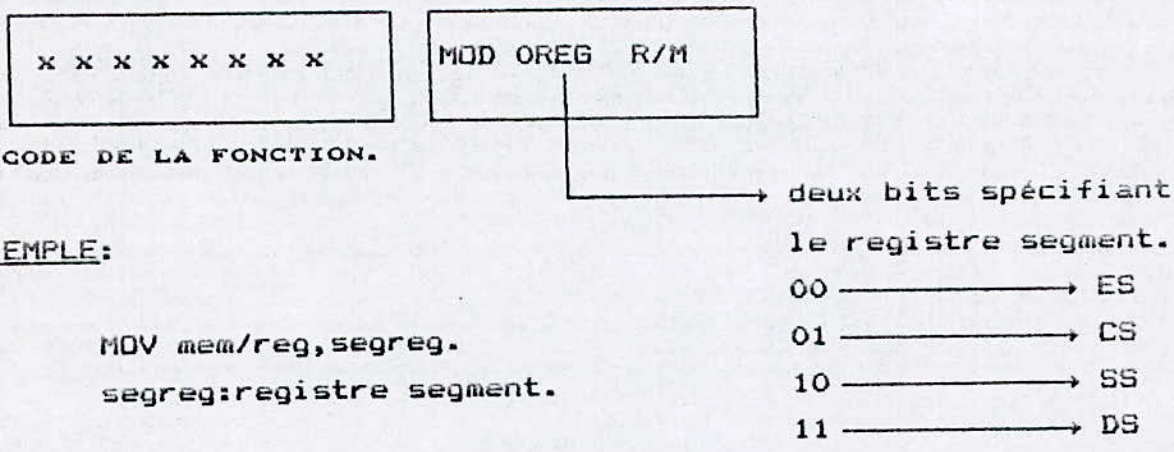


CODE DE LA FONCTION.

EXEMPLE:

MOV mem , ac.

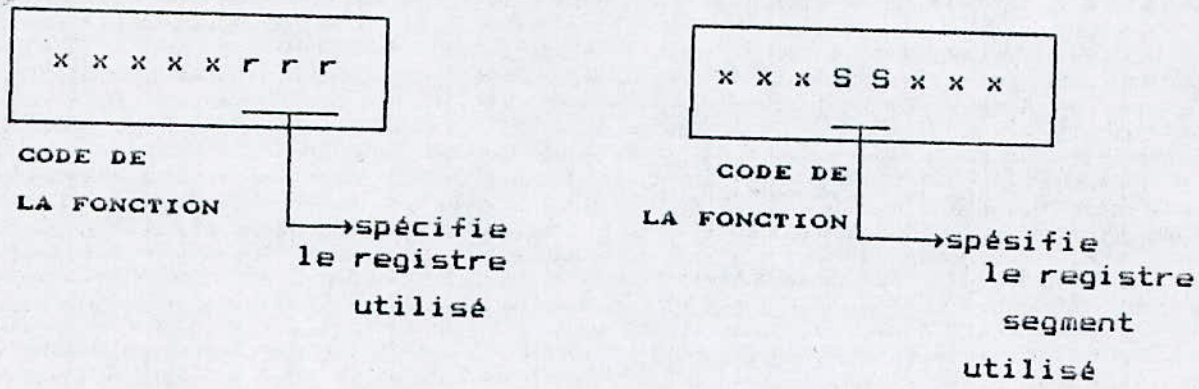
2-



EXEMPLE:

MOV mem/reg,segreg.
segreg:registre segment.

3-



EXEMPLE:

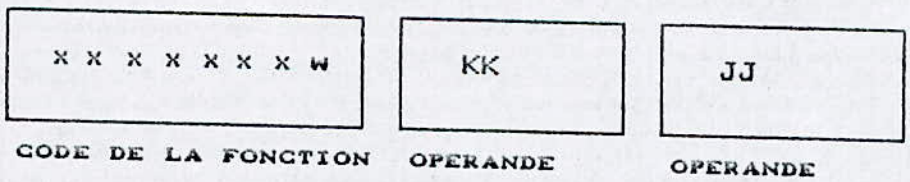
PUSH reg

PUSH segreg

5-1-1-3-ADRESSAGE INDIRECT:

Le contenu de l'adresse effféctive doit être transmis au registre ou à la mémoire correspondante.

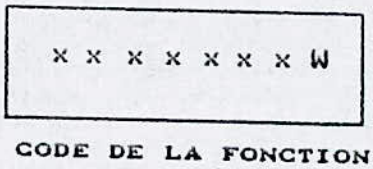
1-



EXEMPLE:

MOV ac,mem

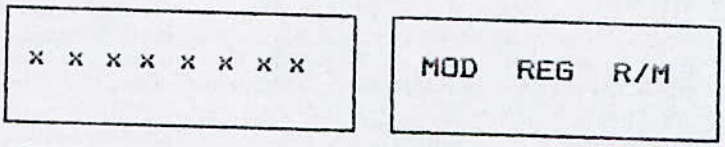
2-



EXEMPLE:

LODSB

3-



CODE DE LA FONCTION OPERANDE

EXEMPLE:

LEA reg,mem

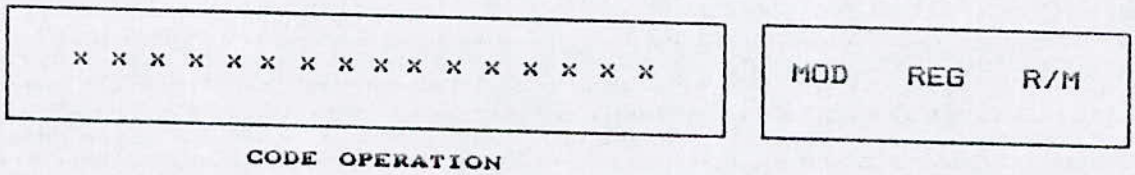
REMARQUE:

Pour ce qui est des autres modes d'adressage les même structures se répétent mais avec des differences au niveau du code opération ainsi que les valeurs de MOD,REG et R/M.

5-1-2-LES INSTRUCTIONS 80286 MODE PROTEGE:

Leur structure présente un code opération sur deux octets au lieu d'un , donc on aura à tester deux octets pour connaitre l'instruction.

Le troisième octet indiquera le mode d'adressage séléctionné ainsi que les éspaces et registres utilisés.



CODE OPERATION

5-2-PROGRAMMES *.EXE ET *.COM :

Le DOS soutient en plus des fichiers BATCH, deux types de fichiers programmes EXE et COM.

La plus grande différence entre ces deux programmes réside dans leur taille: un programme COM ne peut jamais éxcéder 64 KO alors qu'un programme EXE peut avoir une taille bien plus grande ,ainsi dans les programmes COM les REGISTRES SEGMENTS ont tous la même valeur contrairement aux programmes EXE.

Ces deux programmes peuvent être changé et lancé par la fonction EXEC du DOS qui place au début du programme en RAM une structure de donnée appelée PSP (Programme Segment Prefixe) le programme est chargé aprés cette structure de données. (Voir taableau 5-1)

Le PSP toujours d'une longueur de 256 octets, contient des informations importantes aussi bien pour le DOS que pour le programme à éxécuter.

STRUCTURE DU PSP : Program Segment Prefix

ADR.	CONTENU	TYPE
+00H	APPEL DE L'interruption 20H	2 BYTES
+02H	ADR.Segm.de fin mém occupée par prog	1 WORD
+04H	RESERVE	1 BYTE
+05H	FAR CALL de l'intrruption 21H	1 PTR
+0AH	Copie du vécteur d'interrup. 22H	1 PTR
+0EH	Copie du vécteur d'interrup. 23H	1 PTR
+12H	Copie du vécteur d'interrup. 24H	1 PTR
+16H	RESERVE	22 BYTE
+2CH	Adr.Seg. du bloc d'environnement	1 WORD
+2EH	RESERVE	46 BYTES
+5CH	FCB #1	16 BYTES
+6CH	FCB #2	16 BYTES
+80H	Nbr.de caract.dans la ligne de comde	1 BYTE
+81H	Ligne de commande	127 BYTES

Tableau 5-1

2-1-Les programmes *.COM :

Les fichiers COM sont sauvegardés sur la disquette sous une forme réflétant éxactement le contenue de la mémoire RAM lorsqu'ils sont chargés aucune information supplémentaire ne leur ait nécessaire donc ils se chargent et sont lancés plus rapidement que les programmes EXE. Dès qu' un programme COM a été chargé, juste à la suite du PSP, son éxécution commence.

2-2-Les programmes *.EXE :

Les programmes EXE présentent l'interêt, par rapport aux programmes COM, de ne pas être limiter à 64 KO de RAM mais peuvent avoir des tailles de l'ordre de la RAM elle même encore bien plus. Ces programmes ont des segments distincts pour le code, les données et la pile.

Le programme LINK met au début de chaque fichier EXE une structure de données qui contient entre autre les adresses de tous les références de segments.(Voir Tableau 5-2)

Lorsque le programme EXE est chargé par la fonction EXEC du DOS , celle ci connait les adresses auquelles sont chargés les différents segments du programme EXE .Elle peut ainsi inscrire les valeures appropriées dans les cellules mémoires enregistrées dans la tête du fichier EXE.Cette opération entraine un temps plus long entre l'appel et le lancement du programme que pour un programme COM.

Cette inconvénient ne représente rien devant la possibilitée qu'offre les programmes EXE à développer des programmes de taille superieur à 64 KO.

STRUCTURE DU HEADER D'UN FICHIER EXE :

ADR	CONTENU	TYPE
00H	Marque d'un programme EXE (5A4Dh)	1 WORD
+02H	Longueur du fichier mode 512	
+04H	Longueur du fichier Div 512	1 WORD
+06H	Nombre d'adresses de seg. à adapter	1 WORD
+08H	Taille du header en paragraphes	1 WORD
+0AH	Nbre min de paragraphes nécessaires en supplément.	1 WORD
+0EH	Nbre max de paragraphes nécessaires en supplément.	1 WORD
+10H	Contenu du registre SP au lancement du programme	1 WORD
+12H	Checksum sur l'en-tête du fichier	1 WORD
+14H	Contenu du registre IP au lancement du programme.	1 WORD
+16H	Début du sgmt code ds le fichier EXE	1 WORD
+18H	Adresse de la table de relogement dans le fichier EXE.	1 WORD
+1AH	Numéro d'overlay	1 WORD
+1CH	Mémoire-tampon	1 WORD
+??	Adresse des Adresses de segments à adapter (table de relogement)	VARIABLE
+??	Code progrme,segments données & pile	VARIABLE

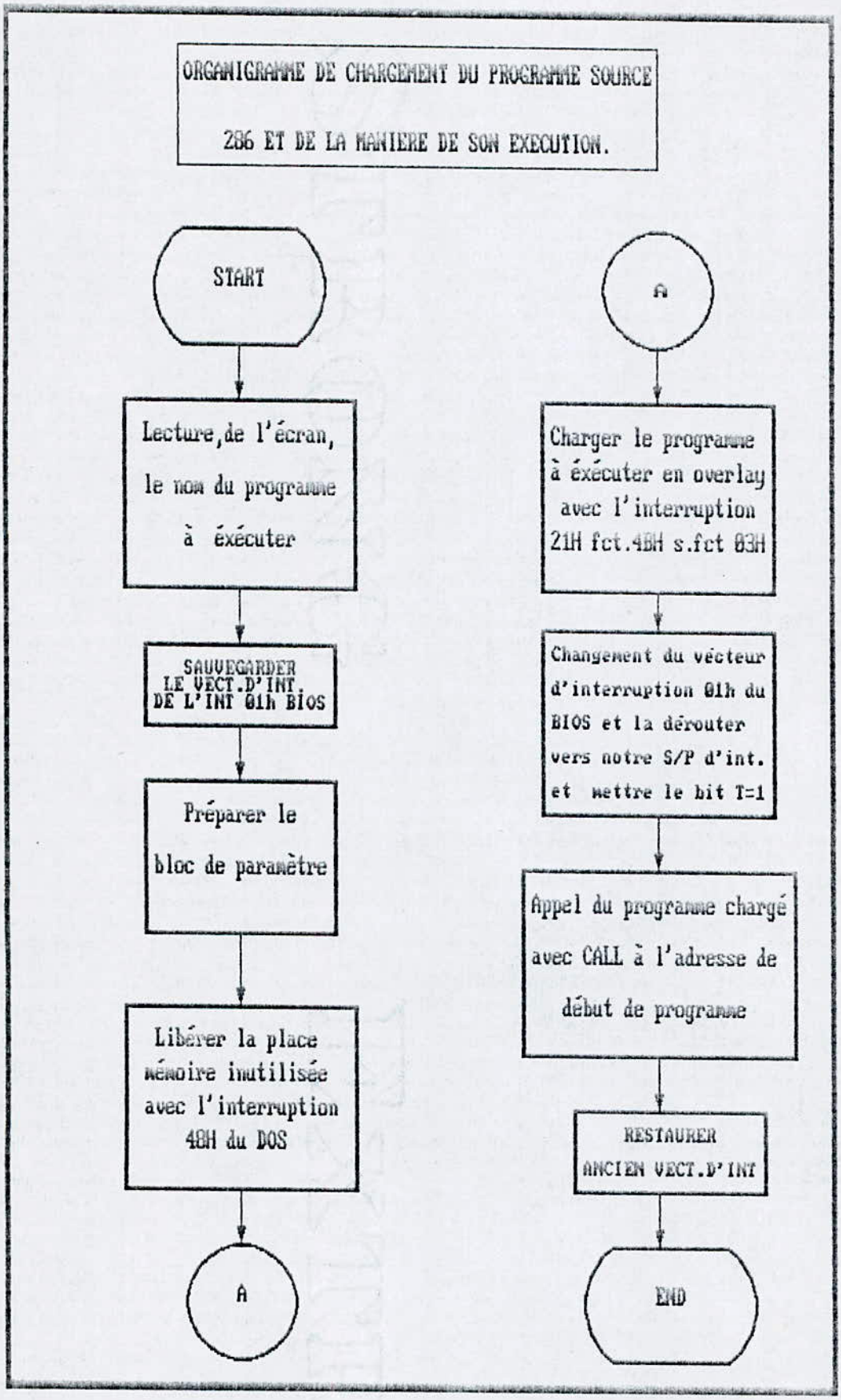
TABLEAU 5-2

5-2 DESCRIPTION ET ORGANIGRAMMES :

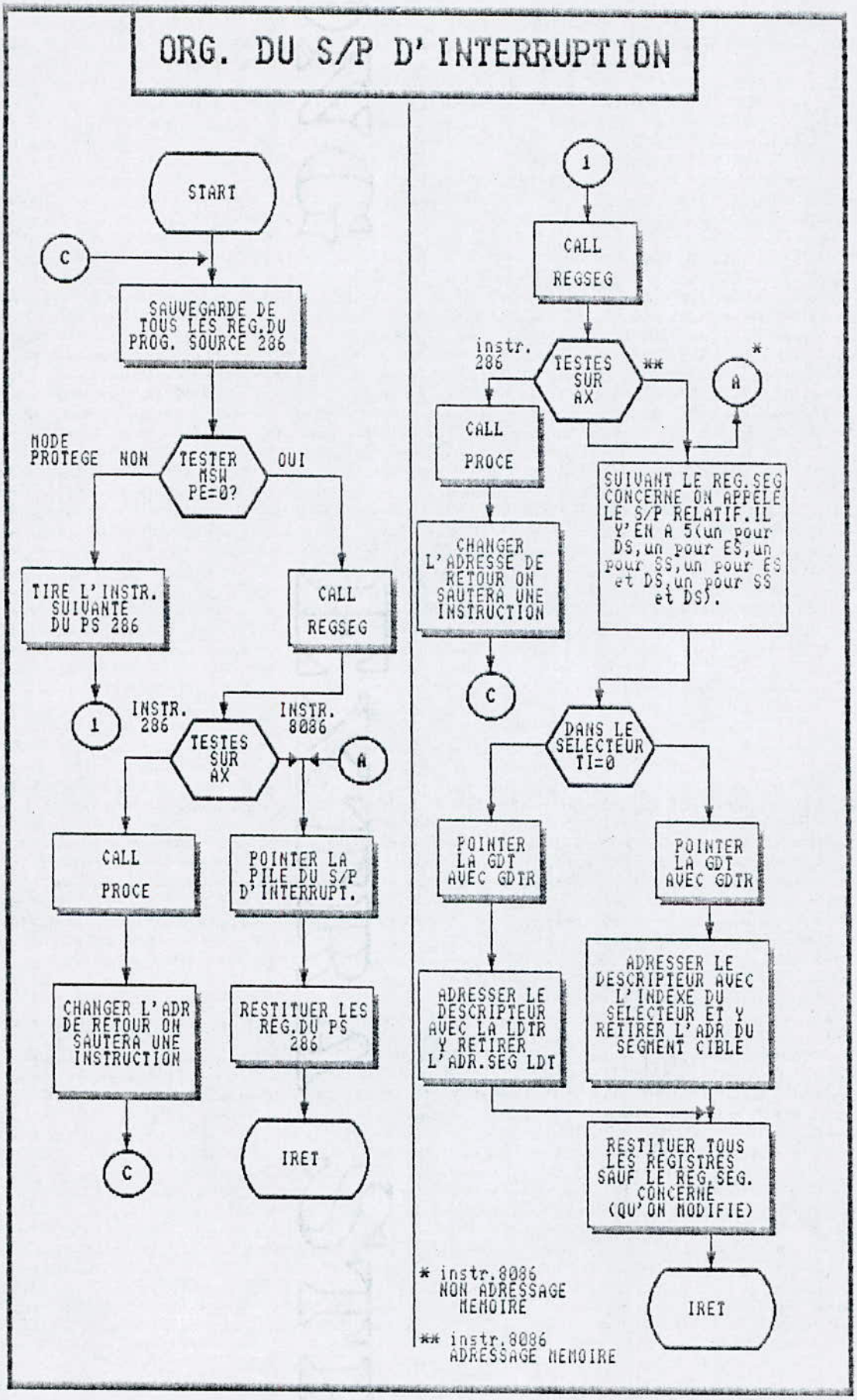
Le logiciel élaboré permet de charger le programme source 80286 choisie par l'utilisateur et de l'éxécuter, à l'aide de la fonction 4Bh de l'interruption 21h du DOS qui permet de charger le programme fils dans un emplacement mémoire définie par le programme père aprés qu'il n'ait libéré la place mémoire restante inutilisée.
La fonction EXEC attendra l'adresse de 2 paramètres dans la paire de registres DS:DX et ES:BX ; le premier paramètre représente le nom du programme à éxécuter ou seulement à charger (suivant la valeur de Al réspéctivement 00h ou 03h) nom qui doit figurer en mémoire comme chaine de charactères terminée par le code ASCII "0" marque de fin.
Ce nom sera introduit par l'opérateur ainsi qu'un chemin de recherche, le second paramètre représente un bloc de paramètres dont la structure est définie par convention et contient toute une série d'informations.
Mais pour éxécuter ce programme 286, on doit préparer le contèxte de chaque instruction 8086 en mode protégé et de dérouter les instructions 80286 vers leurs sous programmes relatifs où chaque instruction 286 est traité à part.
On travaillera alors en mode TRAP, c'est à dire aprés l'éxécution de chaque instruction du programme source 286, le CPU se branchera à un sous programme d'interruption, chargé aussi par la fonction EXEC en overlay (interruption 21h,fonction 4Bh,sous fonction 3),dont le vecteur d'interruption réside aux adresses 00004h et 00005h pour le registre segment CS et 00006h et 00007h pour le poiteur d'instruction IP. Vecteur dont la valeur doit pointer le programme d'interruption qui sera appelé à chaque fois qu'une instruction 286 sera éxécutée.

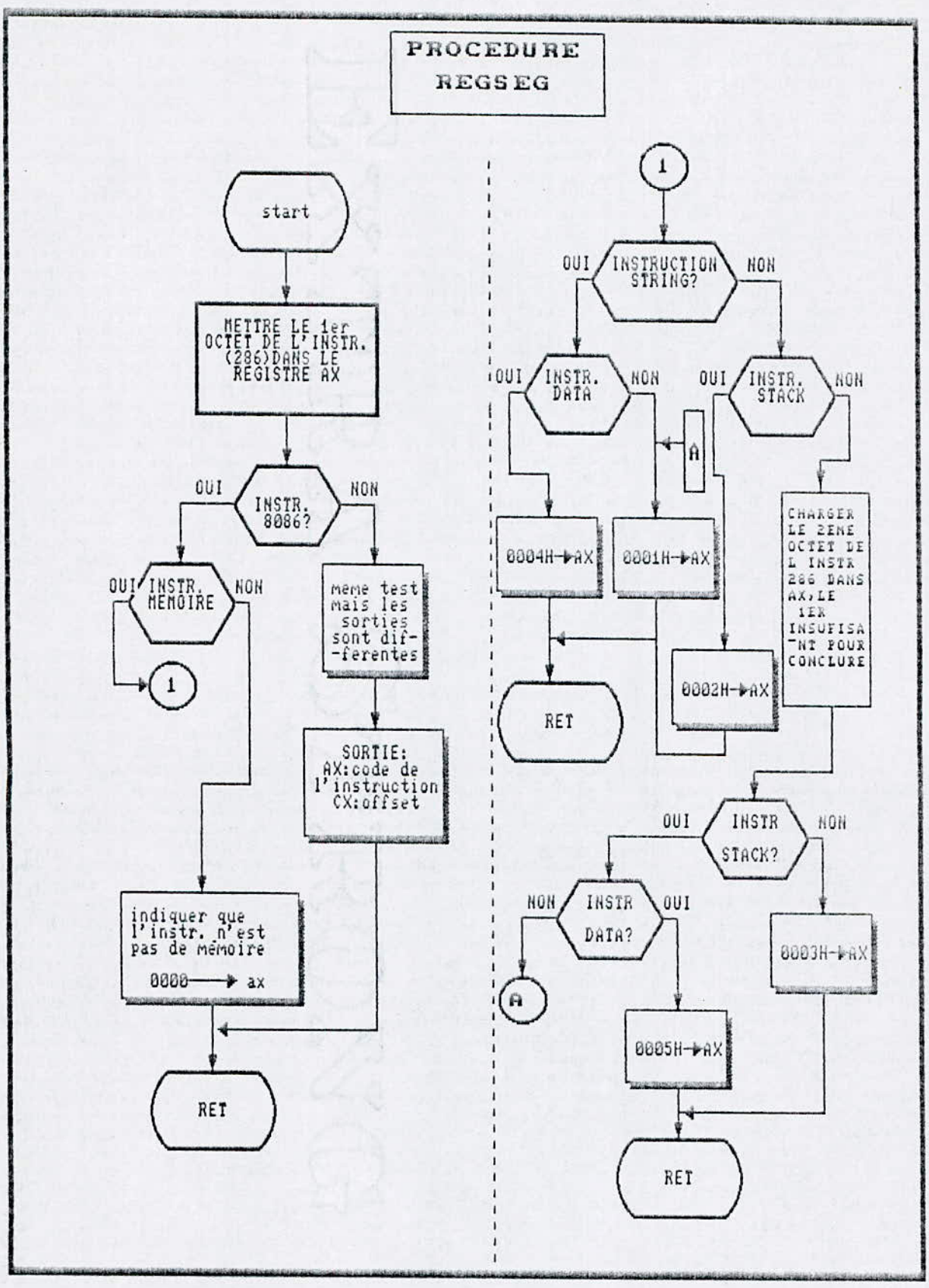
Cette interruption générée aprés chaque éxécution d'une instruction 286 n'est rien d'autre que l'interruption 01h du bios dont le vécteur d'interruption sera modifié par l'interruption 25h du DOS pour la dérouter vers notre sous-programme.
Le sous programme d'interruption éxaminera l'instruction suivante à éxécuter du programme fils et réagira en conséquence pour dérouter l'instruction vers son sous programme relatif, il sera transparent face à une instruction 8086 en mode réel et préparera le contèxte de travail pour une instruction 8086 en mode protégé pour qu'elle soit éxécutée corrèctement.

Remarque:
Il est à noter que la fonction EXEC (en sous fonction 3) ne charge pas le PSP du programme enfant, de sorte qu'il est chargé directement à l'adresse de de segment indiqué (aprés libération de la mémoire) avec une adresse d'offset 0000h.


ORGANIGRAMME DE CHARGEMENT DU PROGRAMME SOURCE
286 ET DE LA MANIERE DE SON EXECUTION.
START
Lecture,de l'écran,
le nom du programme
à éxécuter
SAUVEGARDER
LE VECT.D'INT.
DE L'INT 01h BIOS
Préparer le
bloc de paramètre
Libérer la place
mémoire inutilisée
avec l'interruption
48H du DOS
A
A
Charger le programme
à éxécuter en overlay
avec l'interruption
21H fct.4BH s.fct 03H
Changement du vecteur
d'interruption 01h du
BIOS et la dérouter
vers notre S/P d'int.
et mettre le bit T=1
Appel du programme chargé
avec CALL à l'adresse de
début de programme
RESTAURER
ANCIEN VECT.D'INT
END

ORG. DU S/P D'INTERRUPTION




PROCEDURE
REGSEG
start
METTRE LE 1er OCTET DE L'INSTR. (286)DANS LE REGISTRE AX
OUI
INSTR. 8086?
NON
OUI
INSTR. MEMOIRE
NON
1
même test mais les sorties sont dif-ferentes
SORTIE: AX:code de l'instruction CX:offset
indiquer que l'instr. n'est pas de mémoire
0000 → ax
RET
1
OUI
INSTRUCTION STRING?
NON
OUI
INSTR. DATA
NON
OUI
INSTR. STACK
NON
A
0004H→AX
0001H→AX
CHARGER LE 2ENE OCTET DE L INSTR 286 DANS AX,LE 1ER INSUFISA-NT POUR CONCLURE
RET
0002H→AX
OUI
INSTR STACK?
NON
NON
INSTR DATA?
OUI
A
0003H→AX
0005H→AX
RET

La procédure REGSEG permet de faire la distinction entre les instructions 8086 et 80286, des instructions mémoire et des instructions registres et détermine les registres segments affectés à chaque instruction mémoire.
Les programmes 80286 sont orientés selon 2 modes distincts:

1-Mode réel: Les instructions formant le programme pendant ce mode sont les instructions 8086 et/ou les instructions 8086 modifiées en plus de quelques instructions 80286.

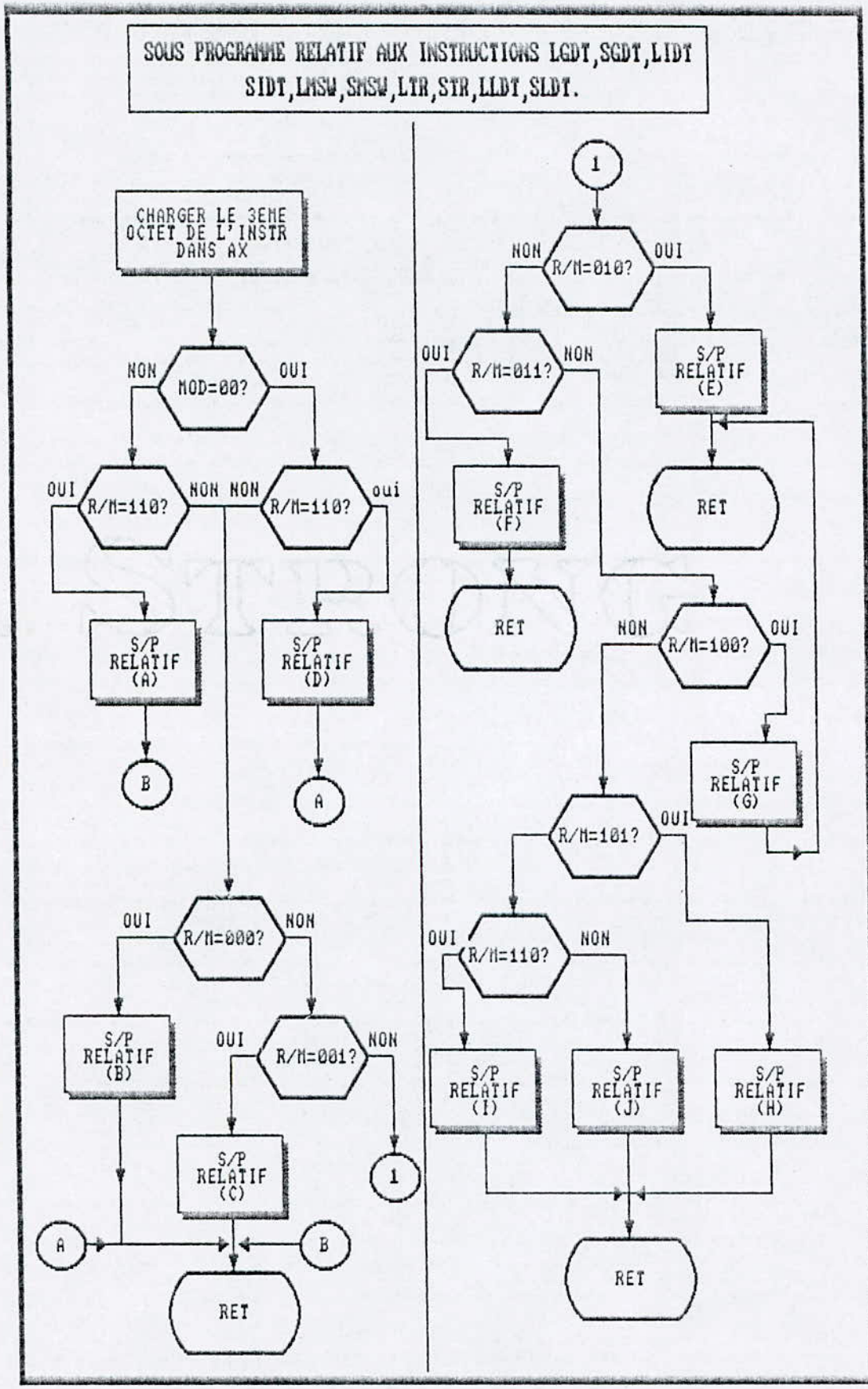
2-Mode protégé: Les instructions formant le programme pendant ce mode sont les instructions du mode reel plus les instructions 80286 caractérisant ce mode.

Pour trouver les registres segments relatifs à une instruction donnée il faut d'abord:

1- Tester à le 1er octet pour connaitre l'instruction 8086 ou 80286.

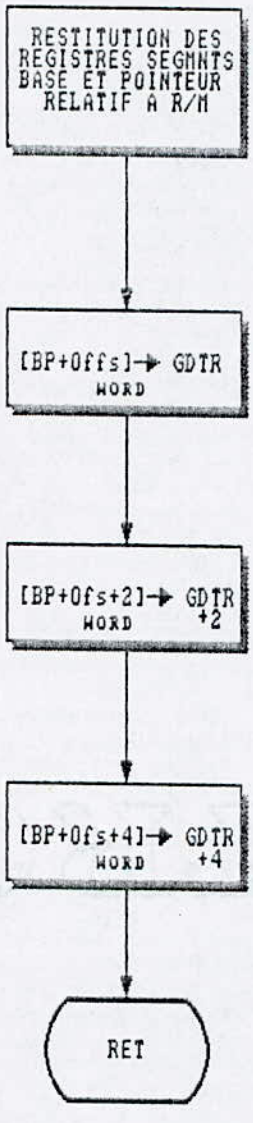
2- Puis tester si c'est une instruction registre ou memoire puis enfin connaitre le registre segment qui lui est affecté, AX comme sortie de cette procédure l'indiquera.

Remarque: Il est à noter que la procédure PROCE contiendra toutes les procédures relatifs à l'éxecution des instructions 80286


SOUS PROGRAMME RELATIF AUX INSTRUCTIONS LGDT,SGDT,LIDT SIDT,LMSW,SMSW,LTR,STR,LLDT,SLDT.
CHARGER LE 3EME OCTET DE L'INSTR DANS AX
MOD=00?
NON
OUI
R/M=110?
OUI
NON NON
R/M=110?
oui
S/P RELATIF (A)
S/P RELATIF (D)
B
A
R/M=000?
OUI
NON
S/P RELATIF (B)
R/M=001?
OUI
NON
S/P RELATIF (C)
1
A
B
RET
1
R/M=010?
NON
OUI
R/M=011?
OUI
NON
S/P RELATIF (E)
S/P RELATIF (F)
RET
RET
R/M=100?
NON
OUI
S/P RELATIF (G)
R/M=101?
OUI
R/M=110?
OUI
NON
S/P RELATIF (I)
S/P RELATIF (J)
S/P RELATIF (H)
RET

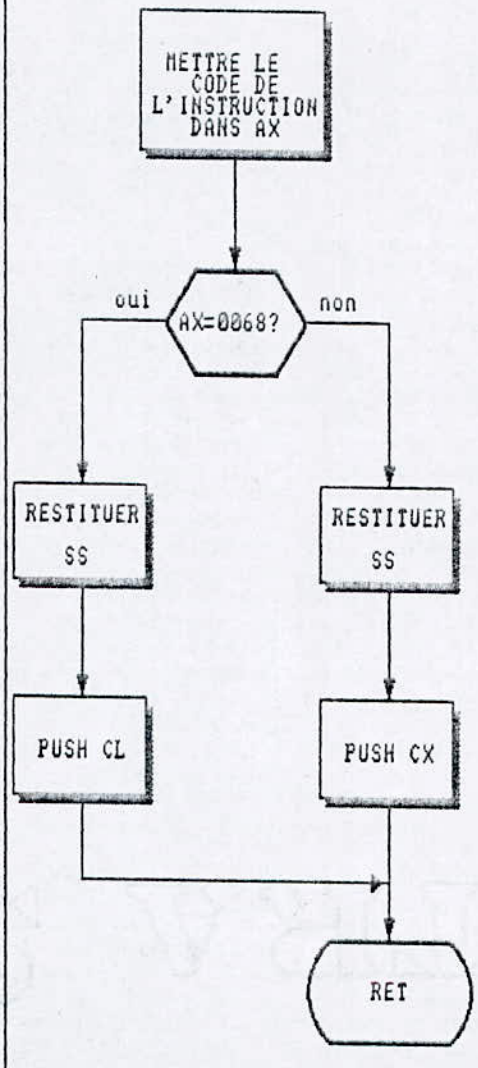
EXEMPLE DU S/P RELATIF A L'INSTRUCTION LGDT

LGDT[BP+Offset]

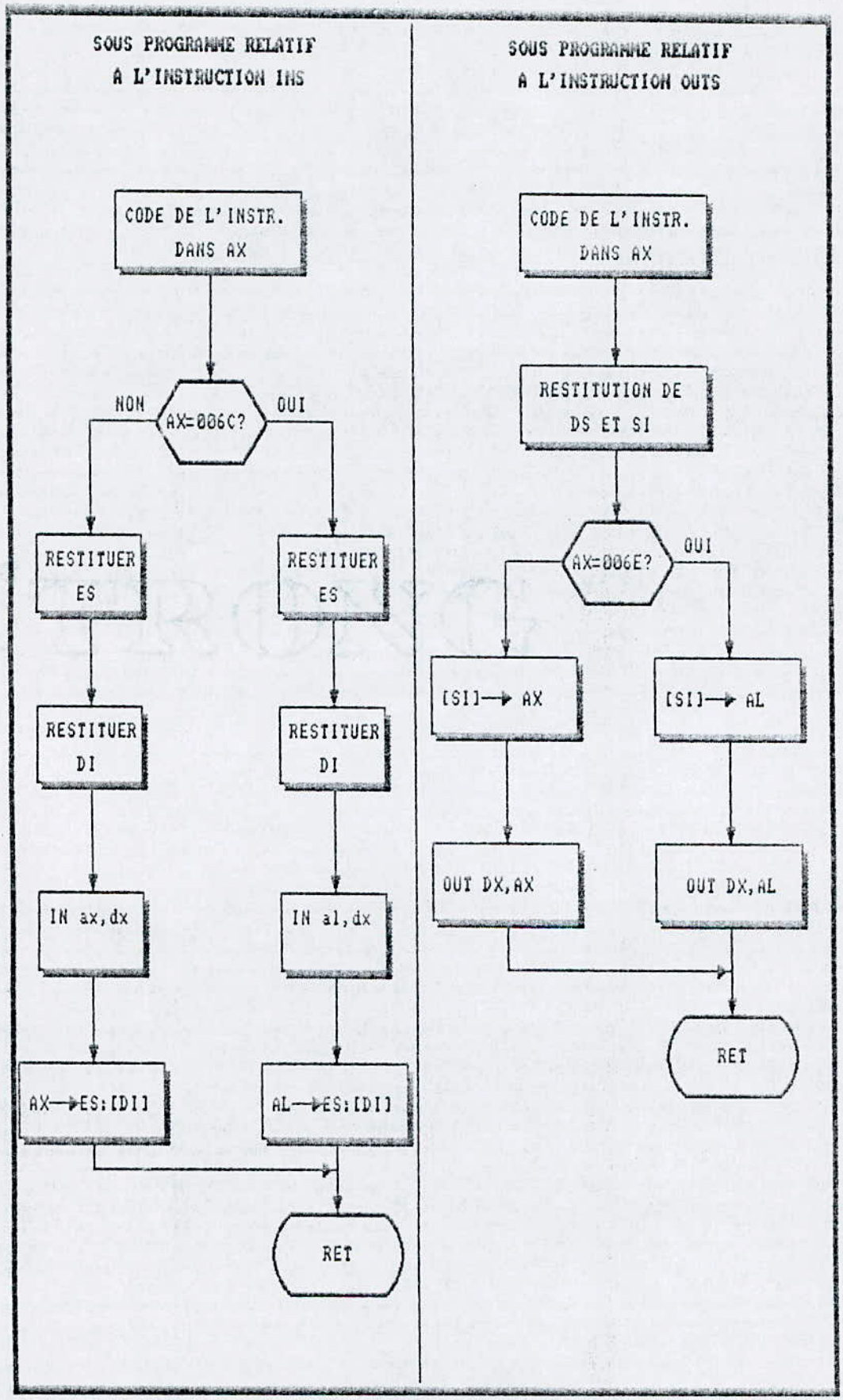


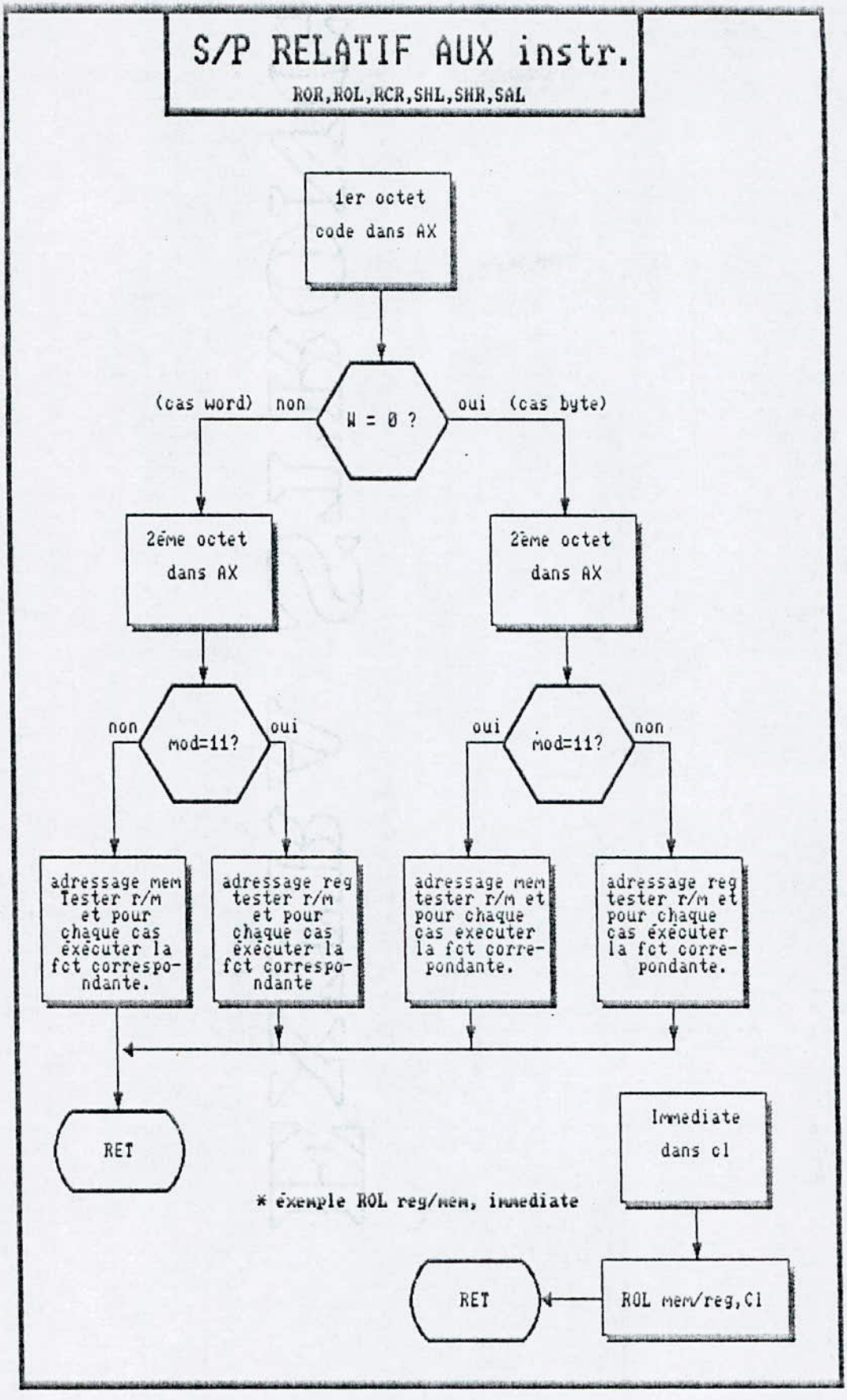
NB:Chaque S/P relatif (A,B,C,D,E,F,G,H J) lui correspond un S/P similaire

SOUS PROGRAMME RELATIF A L'INSTRUCTION PUSH immediat

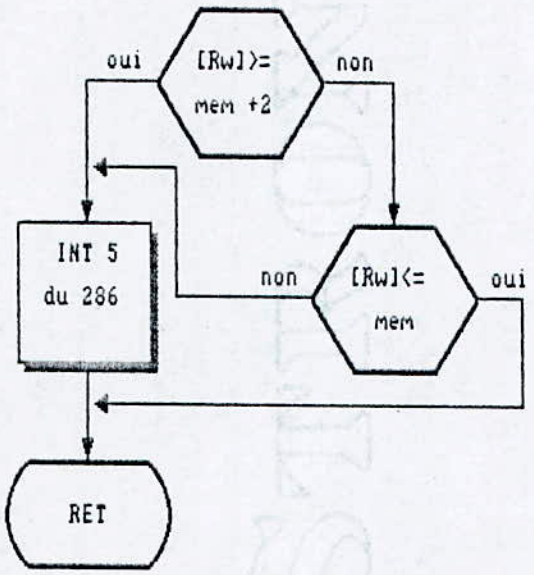


NB: CX contient avant l'appel de ce S/P la valeur immediate.


SOUS PROGRAMME RELATIF
A L'INSTRUCTION INS
CODE DE L'INSTR.
DANS AX
NON
AX=006C?
OUI
RESTITUER
ES
RESTITUER
ES
RESTITUER
DI
RESTITUER
DI
IN ax,dx
IN al,dx
AX→ES:[DI]
AL→ES:[DI]
RET
SOUS PROGRAMME RELATIF
A L'INSTRUCTION OUTS
CODE DE L'INSTR.
DANS AX
RESTITUTION DE
DS ET SI
AX=006E?
OUI
[SI]→ AX
[SI]→ AL
OUT DX,AX
OUT DX,AL
RET


S/P RELATIF AUX instr.
ROR,ROL,RCR,SHL,SHR,SAL
1er octet code dans AX
W = 0 ?
(cas word) non
oui (cas byte)
2ème octet dans AX
2ème octet dans AX
mod=11?
non
oui
mod=11?
oui
non
adressage mem Tester r/m et pour chaque cas éxecuter la fct correspondante.
adressage reg tester r/m et pour chaque cas éxecuter la fct correspondante
adressage mem tester r/m et pour chaque cas executer la fct correpondante.
adressage reg tester r/m et pour chaque cas éxecuter la fct correpondante.
RET
Immediate dans cl
* éxemple ROL reg/mem, immediate
RET
ROL mem/reg,Cl

Organ. instr. bound rw,mem



DESCRIPTION DU LOGICIEL DE GESTION DE LA CARTE D'ANALYSE LOGIQUE :

L'Analyse logique se divise en deux étapes :

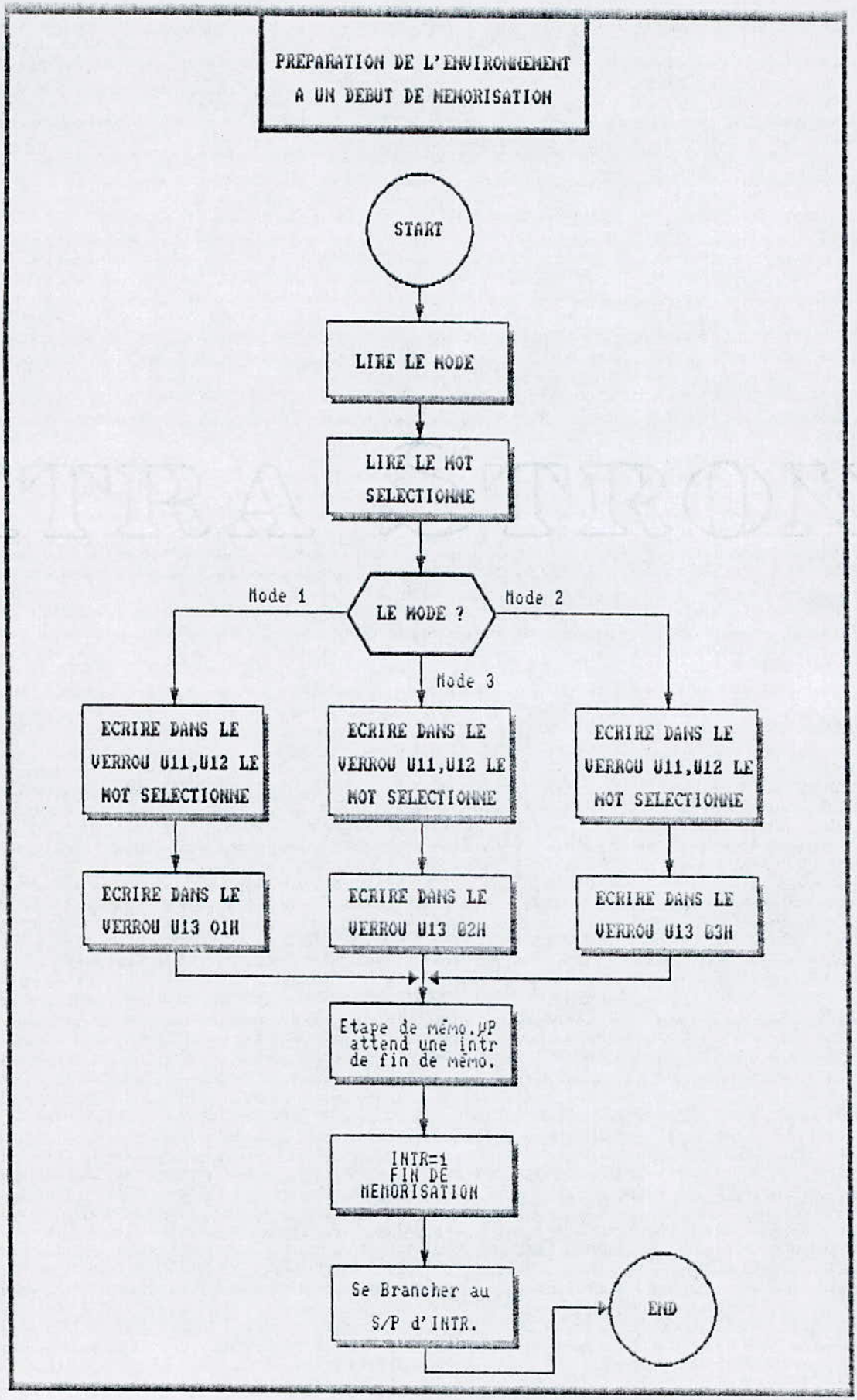
a-étape de mémorisation :

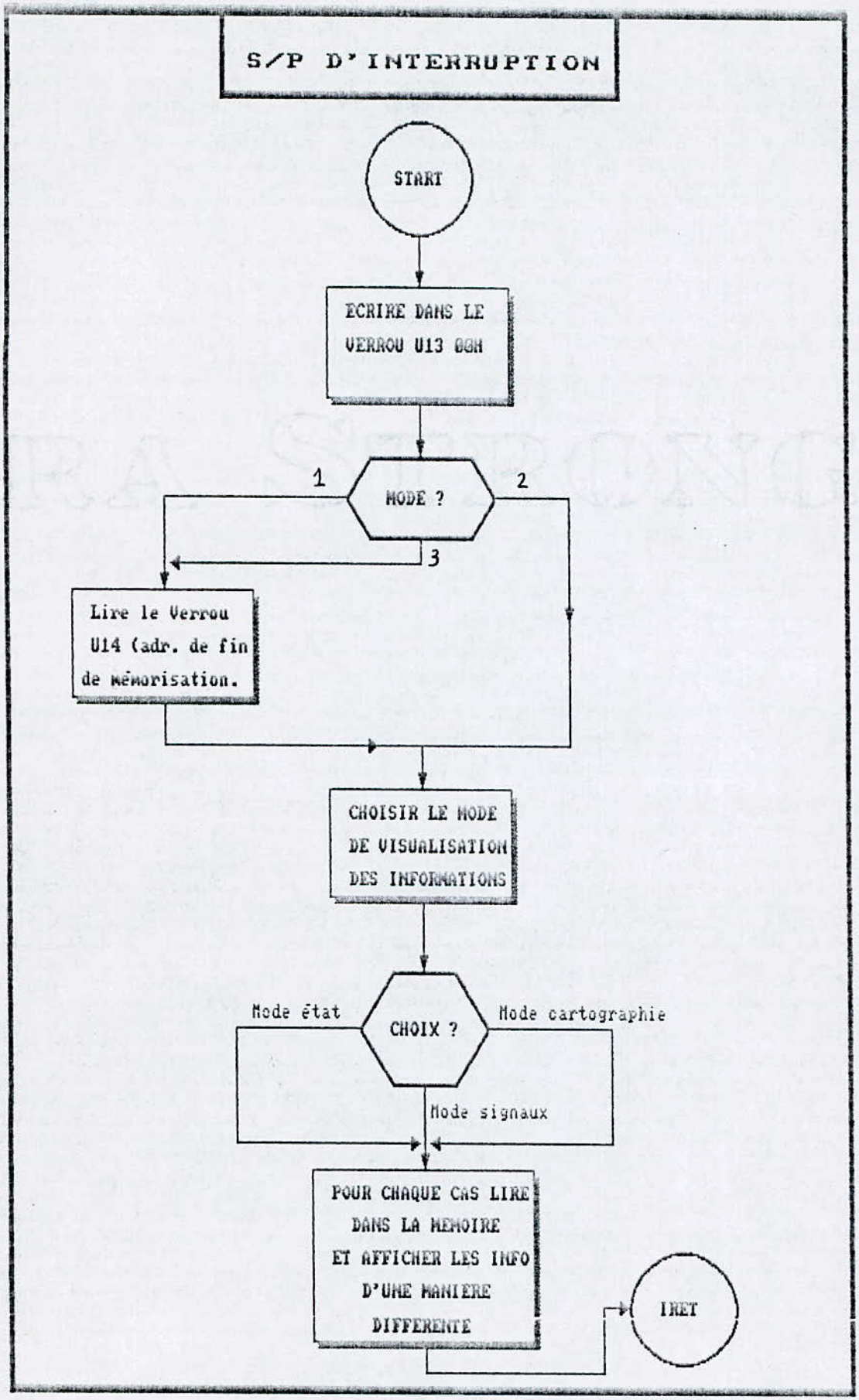
Avant cette étape, où la carte se prend en charge, le microprocesseur doit préparer l'environnement nécessaire à un bon fonctionnement de celle ci.
Il doit alors, lui transférer le mot séléctionné pour le déclenchement, et la commande relative au mode de mémorisation (P1 et P2); Pour enfin entrer dans une étape d'attente d'interruption, signal qui indique la fin de mémorisation.

b-étape d'analyse :

Le microprocesseur doit indiquer cette étape en mettant les lignes de commandes à zéro (P1=P2=0) et lit la mémoire concernée par la mémorisation pour enfin afficher les résultats sous la forme choisie.
Il est à signaler que pour les modes de pré-déclenchement et déclenchement, le verrou U14 mentionne l'adresse où le comptage c'est arrêté à l'apparition du mot pré-séléctioné.

Les organigrammes qui suivent montrent le programme de gestion et le sous programme d'interruption qui gèrent la carte d'analyse logique.


PREPARATION DE L'ENVIRONNEMENT
A UN DEBUT DE MEMORISATION
START
LIRE LE MODE
LIRE LE MOT
SELECTIONNE
LE MODE ?
Mode 1
Mode 2
Mode 3
ECRIRE DANS LE
VERROU U11,U12 LE
MOT SELECTIONNE
ECRIRE DANS LE
VERROU U11,U12 LE
MOT SELECTIONNE
ECRIRE DANS LE
VERROU U11,U12 LE
MOT SELECTIONNE
ECRIRE DANS LE
VERROU U13 01H
ECRIRE DANS LE
VERROU U13 02H
ECRIRE DANS LE
VERROU U13 03H
Etape de mémo.µP
attend une intr
de fin de mémo.
INTR=1
FIN DE
MEMORISATION
Se Brancher au
S/P d'INTR.
END


S/P D'INTERRUPTION
START
ECRIRE DANS LE VERROU U13 00H
1
MODE ?
2
3
Lire le Verrou U14 (adr. de fin de mémorisation.
CHOISIR LE MODE DE VISUALISATION DES INFORMATIONS
Mode état
CHOIX ?
Mode cartographie
Mode signaux
POUR CHAQUE CAS LIRE DANS LA MEMOIRE ET AFFICHER LES INFO D'UNE MANIERE DIFFERENTE
IRET

CONCLUSION

Le système de développement est un outil indispensable dans n'importe quel laboratoire de recherche et de développement ou de maintenance des cartes logiques.

La carte d'analyse logique proposée à 56 voies, et dont les tests n'ont été fait que sur un module de 16 voies seulement, peut être réaliser dans le cadre d'un autre projet ainsi que le logiciel élaboré puisque des routines supplémentaires peuvent être adjointes assurant la gestion des tâches, des interruptions et la protection.Se basant sur le squelette du logiciel on peut également élargir l'émulation à toute la famille des IAPX86.

Notre contribution, minime que soit elle , nous a permis d'approfondir nos idées dans le domaine de la micro-informatique et d'élargir nos connaissances dans le monde des PC.

ANNEXE

LES INTERRUPTIONS DOS UTILISEES :

1-INTERRUPTION 01h

Cette interruption est automatiquement générée aprés chaque instruction si l'indicateur TF (trap flag) est à 1.

Paramètres d'entrée:

NEANT

Paramètres de sortie:

NEANT

2-INTERRUPTION 21h FONCTION 25h

Cette fonction est utilisée pour modifier une adresse dans un vécteur d'interruption.

Paramètres d'entrée:

Ah = 25h

Al = Numéro de l'interruption dont le vécteur va changer.

DS:DX = Segment:Offset de la nouvelle Routine d'interruption.

Paramètres de sorties:

NEANT

3-INTERRUPTION 21h FONCTION 35h

Cette fonction renvoie le vécteur d'une interruption.

Paramètres d'entrée:

Ah = 35h

Al = Numéro de l'interruption.

Paramètres de sortie:

ES:BX = Segment:offset de la routine d'interruption.

4-INTERRUPTION 21h FONCTION 4Bh

Cette fonction permet de charger un programme en mémoire et éventuellement en lancer l'éxécution.

Paramètres d'entrée:

Ah = 4Bh

Al = 00h charger et éxécuter le programme.

03h charger en Overlay (Pas d'éxécution).

DS:DX = Segment:Offset du nom du programme à charger en mémoire.

ES:BX = Segment d'un Bloc de Paramètres.

Paramètres de sortie:

NEANT (Si le fichier est chargé correctement sinon le bit C indiquera une erreure s'il est à un.)

5-INTERRUPTION 21h FONCTION 48h

Cette fonction alloue un bloc de mémoire dont la taille est éxprimée en paragraphes (Un paragraphe = 16 Bytes).

Paramètres d'entrée:

Ah = 48h

BX = Nombre de paragraphes à allouer.

Paramètres de sortie :

AX:0000h Adresse de la zone mémoire allouée.

6-INTERRUPTION 21h FONCTION 4Ah

Cette fonction modifie la taille d'un bloc mémoire alloué précédemment par la fonction 48h.

Paramètres d'entrée:

Ah = 4Ah

BX = Nouvelle taille requise pour le bloc mémoire.

ES:0000 Adresse de la zone mémoire dont on souhaite changer la taille.

Paramètres de sortie:

NEANT (Si C=0 Pas d'erreurs)

BIBLIOGRAPHIE :

- R.RECTOR AND G.ALEXY
  "The 8086 book" MC-GRAW HILL 1980
- M.AUMIAU
  "Microprocesseurs à 16 bits" Ed.MASSON 1985
- Reference manual Intel
  "IAPX 286 Programmer's" INTEL 85
- C.VIEILLEFOND
  "Mise En Oeuvre De L'IAPX 286" Ed.SYBEX 1986
- HANDBOOK INTEL
  "MicroSystem Components" INTEL 86
- H.LILEN
  "80286 Assembleur IBM AT et compatible"
  Ed.RADIO
- MICHAEL TISCHER
  "La Bible Du PC Programmation Système"
  Ed.MICRO-APPLICATION
  1989
- M.TRIO
  "Utiliser MS-DOS à l'aide de l'assembleur"
  Ed.EYROLLES 1987
- PH.MERCIER
  "Les Interruptions du MS-DOS"
  Ed.MARABOUT 1990
- MICHEL.B & M.GAY
  "Comment Choisir un système de développement"
  EDITESTS 1983
- M.AUMIAU
  "Les Systèmes à microprocesseurs"
  Ed.MASSON 1982

- DRAI OMAR

"Etude et réalisation d'un système de développement pour µprocesseurs"

Option Hardwar INI 1985

- OUSSAID.R & HABDOUN.M

"Etude et mise en oeuvre de l'analyseur logique 7D02 de TEKTRONIX"

ENP Janvier 1985

- "Le Langage Machine sur PC"

Ed.MICRO-APPLICATION 1987

- "L'IAPX 286 INTEL"

P.TRUC MICRO-SYSTEMES Mai 1985

- "Fonctions d'émulation Assisté par IBM-PC"

FRANZETTI A. Eléctronique Industrielle N°75 Sept 1984

- "Un Emulateur Economique"

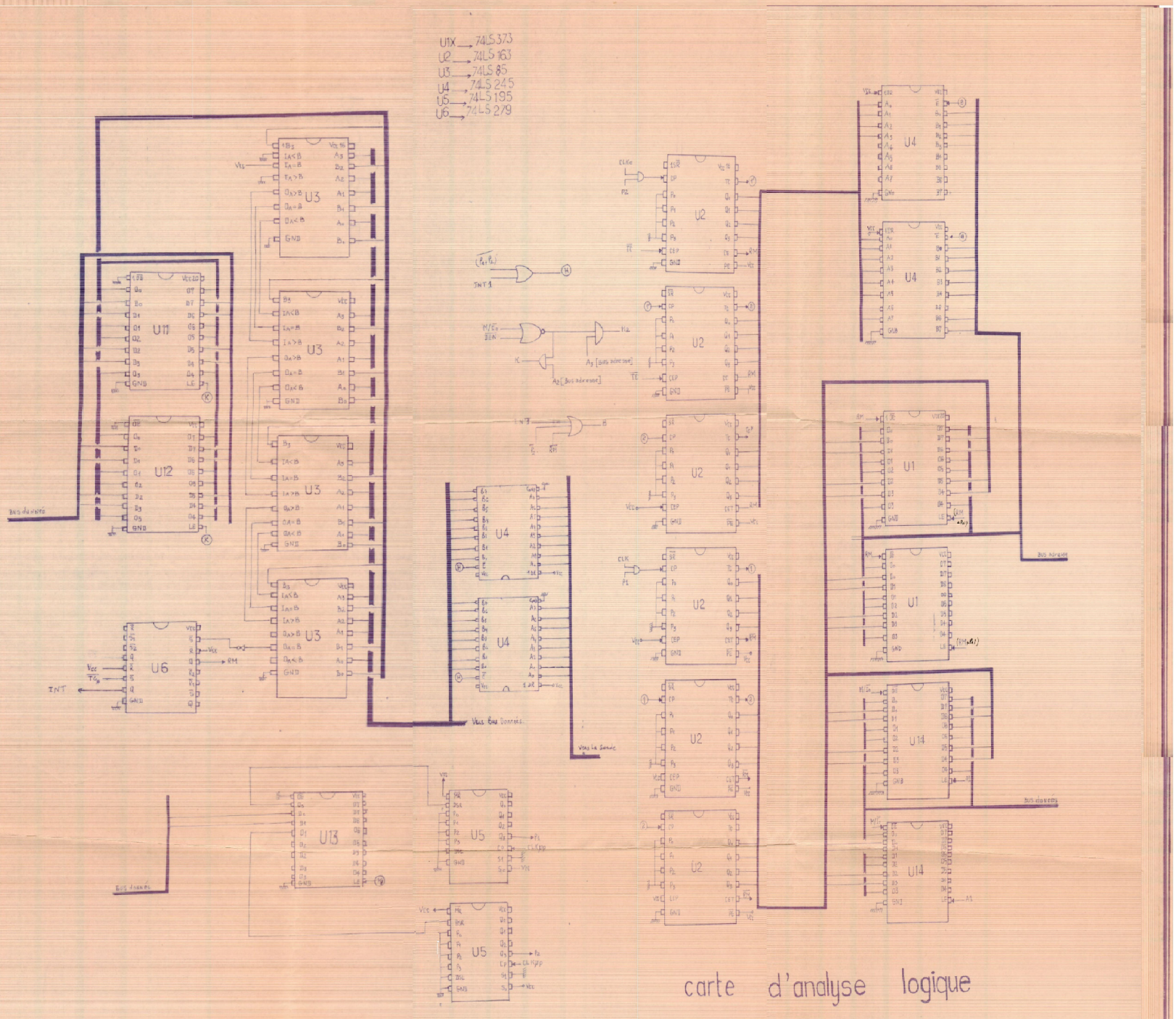
FAISANDIER Y. MICRO-SYSTEMES Dec 1985

- "L'émulation Automatique Des Cycles Bus"

PANTANO G. Eléctronique Industrielle N°83 Fev 1985

- "Outils De Développement 'intégré' Pour µP Sur IBM-PC"

FRANZETTI A. Eléctronique Industrielle N°82 Fev 1985


U1X → 74LS373
U2 → 74LS163
U3 → 74LS85
U4 → 74LS245
U5 → 74LS195
U6 → 74LS279
carte d'analyse logique
partie reconnaissance du mot et adressage