THÈSE

présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE

en vue de l'obtention du titre de

DOCTEUR D'UNIVERSITÉ

Spécialité "Automatique"

par

Benwei YONG

Ingénieur de l'INSTITUT TECHNIQUE DE TRAITEMENT DE DONNEES de PEKIN

DEFINITION ET SIMULATION D'UNE UNITE DE COMMANDE POUR PROCESSUS SIMULTANES.

Soutenue le 26 février 1982 devant la Commission d'Examen

Messieurs	Pierre VIDAL	Président
	Jean-Marc TOULOTTE	Rapporteur
	Georges MANESSE	Examinateur
	Pierre FLOTAT	Invité

AVANT - PROPOS

Le travail présenté dans ce mémoire a été effectué au Centre d'Automatique de l'Université des Sciences et Techniques de Lille 1.

C'est avec la plus grande bienveillance que Monsieur le Professeur Pierre VIDAL, Directeur du Centre d'Automatique m'a permis d'effectuer ma recherche au sein de son Laboratoire, je lui adresse ici ma plus profonde gratitude.

Que Monsieur Jean-Marc TOULOTTE, Professeur à l'Université des Sciences et Techniques de Lille 1, veuille bien trouver ici l'expression de ma sincère reconnaissance pour sa direction sympatique et efficace tout au long de cette recherche.

Je remercie sincèrement Monsieur Georges MANESSE, Maître-Assistant au Laboratoire d'Electronique de l'Université des Sciences et Techniques de Lille 1, pour sa participation à ce Jury.

Je remercie également Monsieur Pierre FLOTAT, Président de l'URCEN-SERIME à Seclin, pour sa participation à ce Jury de Thèse.

Mes remerciements vont aussi à mes Professeurs chinois, en particulier ceux de l'Université QING-HUA et de l'Institut Technique de Traitement de Données de Pékin, pour les enseignements qu'ils m'ont dispensés.

J'exprime ma sincère reconnaissance à Monsieur Christian VASSEUR, Maître-Assistant à l'Université des Sciences et Techniques de Lille 1, qui m'a guidé dans mes travaux grâce à de nombreuses discussions.

Les chercheurs et le Personnel Administratif du Centre d'Automatique, surtout Monsieur Bernard CEURSTEMONT et Madame Annick PIGNON, ont toujours répondu amicalement à mes sollicitations. Je les remercie très sincèrement pour leur accueil inoubliable.

Je tiens aussi à remercier Madame Michèle LELONG pour la réalisation dactylographique de ce mémoire.

SOMMAIRE

INTRODUCTION

Augmenter la vitesse de traitement d'un système informatique et aborder des problèmes de complexité croissante sont deux objectifs permanents des cherches.

Pour satisfaire ces exigences, il a été fait de grands efforts à deux niveaux : soit celui de la technologie, soit celui de la méthodologie. Le travail présenté dans ce mémoire est une contribution à l'étude des structures de commande et des méthodologies de réalisation.

Un premier chapitre discute de la notion de parallélisme et a pour but de définir notre système de calcul parallèle.

Le deuxième chapitre donne les bases de notre travail. La description par GRAFCET amène les choix tout au niveau matériel que logiciel pour tout le système que nous avons proposé. Nous introduisons donc les notions fondamentales et leurs applications essentielles dans notre application. Toutefois nous nous contentons d'utiliser les résultats d'ouvrages /1//2/ sans explication détaillée pour allèrger notre mémoire.

Le troisième chapitre aborde l'implantation du GRAFCET et la réalisation. En ce qui concerne l'implantation, nous precisons diverses structures de données représentant le GRAFCET et nous concevons concrètement l'organigramme du programme de gestion. La réalisation comporte l'architecture de l'unité de commande et le microprogramme qui réalise la gestion.

Le chapitre IV est consacré a la simulation du parallélisme que nous proposons et à celle du microprogramme de façon à vérifier la validation de la conception.

CHAPITRE I : UNE UNITE DE COMMANDE POUR CONTROLER DES TRAITEMENTS PARALLELES.

I.1. Les Divers types de parallélismes :

Un calcul scientifique complexe, un modèle biologique, écologique ou sociologique associent, souvent, au traitement un très grand nombre d'équations simultanées et donc un parallélisme naturel se produit. Il est impossible d'exposer tous les types de machine qui travaillent avec parallélisme.

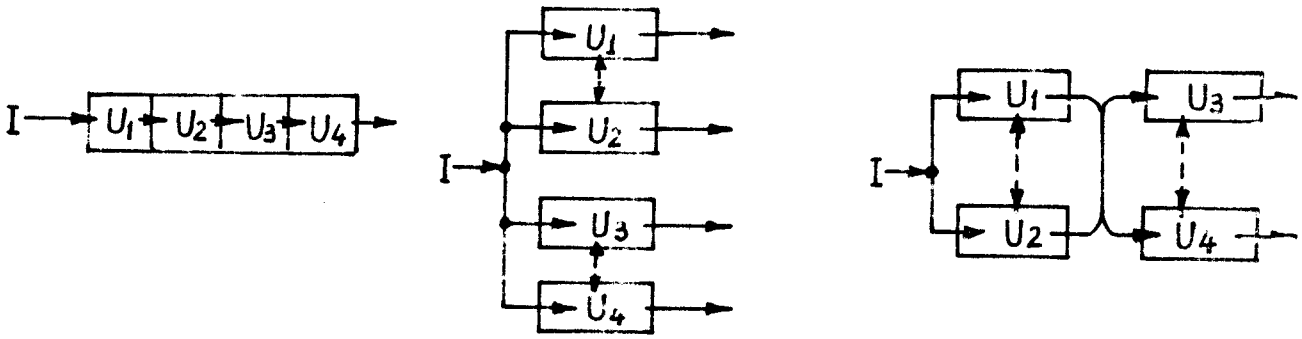
En effet, un microprocesseur à 4 bits est une machine parallèle au niveau le plus bas - parallélisme des bits. A l'opposé, un réseau d'ordinateurs se trouve au niveau le plus haut - parallélisme des traitements et des échanges. Cependant, quelques machines, par exemple STARAN /3/, comportent quelques centaines de processeurs 1 bit qui travaillent en parallèle. L'information traitée se trouve donc au niveau le plus bas (bit), mais son unité de traitement se trouve au niveau assez haut (traitement). Il y a encore d'autres propriétés qui distinguent les divers parallèlismes, par exemple, les structures avec parallèlisme, les informations traitées et traitantes, etc. Nous nous proposons donc d'essayer de classifier le parallélisme pour préciser notre système.

I.1.1. Essai de classification des parallélisme

I.1.1.1. Les structures avec parallélisme :

D'après le mode de calcul, le parallélisme se divise en trois catégories : **pipe-line, parallèle, mixte** (pipe-line et parallèle).

On peut les schématiser par la figure I.1.



a) Pipe line b) Parallèle c) mixte

Figure I-1 Structure de parallélisme

I.1.1.2. Unité de traitement :

C'est l'unité de traitement qui exécute diverses opérations sur l'information d'entrée. Elle fonctionne de façon parallèle à différents niveaux, soit celui des circuits logiques, soit celui des opérateurs, soit celui des processeurs. Dans n'importe quel ordinateur, même un microordinateur, les circuits logiques fonctionnent parallélement sans gros problème de conception. Mais la gestion des unités de traitement parallèle au niveau processeur (ou calculateur) n'est pas du tout une chose simple sur le plan du sytème d'exploitation qui souvent ralentit la vitesse de parallélisme.

I.1.1.3. Information traitante et information traitée :

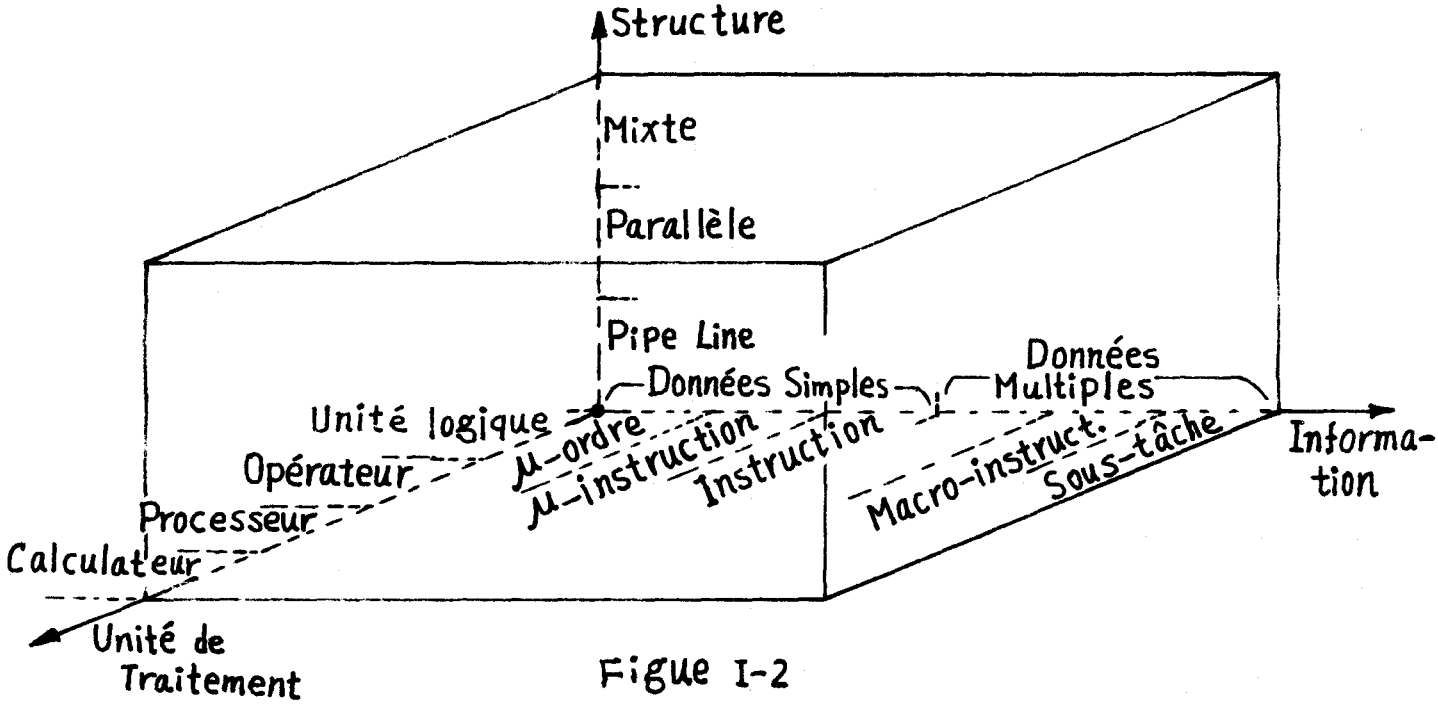
Les opérations, exécutées par l'unité de traitement, sont l'information traitante qui apparait aux divers niveaux : microordre, microinstruction, superinstruction (macroinstruction) jusqu'à sous-tâche (processus). On peut considérer l'exécution d'une microinstruction horizontale comme un certain parallélisme dont l'information traitante se trouve au niveau du microordre.

L'information traitée peut-être aux niveaux : bit, octet, mot jusqu'à un tableau.

Les machines SIMD (Simple-Instruction, Multiple-Données) et MIMD (Multiple-Instruction, Multiple-Données) ne diffère que par l'information traitante.

I.1.1.4. Classification des parallélismes :

Nous avons tenté de représenter les parallélismes dans un espace à trois dimensions : Unité de traitement, structure et information traitante (fig. I.2.).



Figue I-2

Nous distinguons la sous-tâche de la macroinstruction. La dernière ne peut pas être divisée par le programmeur, mais la sous-tâche est un segment de programme qui peut être déterminé au gré du programmeur.

La machine ILLIAC-IV /4/ est du type SIMD, c'est-à-dire que l'information traitante est à instruction simple, l'information traitée est à données multiples. Elles sont exécutées parallélement par plusieurs processeurs, unités de traitement.

La machine LAU système /5/ est du type MIMD - chaque processeur exécute différentes instructions sur différentes données. Les processeurs marchent parallélement et la mémoire est organisée en pipe-line implicite.

La machine MUNAP /6/ est organisée en parallèle au niveau microinstruction.

Les gros ordinateurs IBM 360/91, 370/165 possèdent plusieurs opérateurs (additionneur en virgule fixe, multiplicateur flottant, etc...) comme unités de traitement. Elles sont organisées en pipe-line au niveau instruction et microinstruction /7/.

Au contraire, les machines CDC-6600, 7600 fonctionnent en principe mixte. Leur processeur central est organisé en pipe-line au niveau instruction, mais le processeur central et les processeurs périphériques fonctionnent parallèlement sous la gestion du sysmtème d'exploitation /7/.

I.1.2. Les problèmes de parallèlisme dans la machine pipe-line

Le principe pipe-line consiste à réaliser une exécution simultanée de plusieurs sous-tâches dans leur forme originale. Donc l'approche algorithmique et l'expression programmée du problème ne sont pas modifiées et le langage évolué universel (BASIC, PASCAL, FORTRAN etc) peut-être appliqué à cette forme de parallélisme. Ceci est certainement vrai pour les machines organisées en pipe-line au niveau instruction, microinstruction, microordre. En effet, beaucoup de grosses machines marchent jusqu'ici à ce niveau là. Par exemple, IBM-360/91 et 195, CDC-6600 et 7600, CDC STAR 100 etc. Nous allons maintenant discuter quelques problèmes concernant les machines pipe-line au niveau instruction et au niveau plus élevé.

I.1.2.1. Décomposition du problème :

Comme la figure I.3., une tâche T est décomposée en plusieurs sous-tâche $T = \{T_0, T_1, \ldots T_{n-1}\}$, qui sont exécutées par plusieurs unités de traitement $U = \{U_0, U_1, \ldots U_{n-1}\}$. Les temps de traitement sont respectivement $t = \{t_0, t_1, \ldots t_{n-1}\}$. Les temps de déplacement sont respectivement $\tau = \{\tau_0, \tau_1, \cdots \tau_{n-1}\}$.

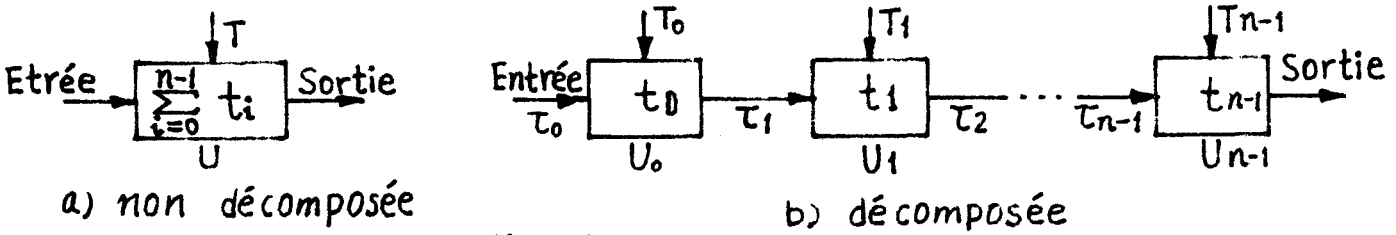


Figure I-3

$\sum_{i=0}^{n-1} t_i$, donc Avant la décomposition, le temps de traitement est égal à le débit du système est Dc :

$$D_c = 1/(\sum_{i=0}^{n-1} t_i)$$

Après la décomposition, le débit du système devient Dp :

$$Dp = 1/Max(t_i + \tau_i)$$

Un cas particulier apparaît si :

$$t_i = t_j = \frac{1}{n} \sum_{i=0}^{n-1} t_i \qquad \forall\ i \text{ et } j$$

$$\tau_i = \tau_j = 0$$

Il vient alors : Dp = n Dc

Il apparaît que si plus n est grand, plus l'amélioration de Dp est forte.

Mais une grande valeur de n entraine des inconvénients :

- Le Max $[t_i + \tau_i]$ est limité pour le moins par la théorie de relativité : Une impulsion électrique parcourt au mieux 30 centimètres en une nanoseconde ! Jusqu'à maintenant aucune machine pipe-line ne déclenche une sous-tâche, une instruction ou une microinstruction à moins de 10 ns /7/. Cela signifie qu'il est préférable d'organiser une machine pipe-line à un niveau plus élevé, par exemple au niveau sous-tâche, pour atténuer l'effet de

la restriction de la théorie de relativité.

- Le nombre n est la quantité d'unités de traitement et aussi le nombre de répétitions de traitement. La machine au niveau sous-tâche signifie que l'unité de traitement est un processeur, et chaque sous-tâche doit se répéter au moins n fois pour ne pas vider le pipe-line. Plus n est grand et plus le niveau est haut, plus le système coûte cher et plus il est mal utilisé.

Au contraire, la machine pipe-line au niveau microinstruction demande un circuit logique comme unité de traitement. Un problème quelconque comprend toujours un grand nombre de répétitions de microinstruction. C'est pourquoi les machines universelles sont organisées en pipe-line au niveau le plus haut qui est celui de l'instruction. Seules les machines spécialisées sont au niveau sous-tâche.

- Un branchement dans un programme d'utilisateur provoque fatalement une rupture dans le fonctionnement du pipe-line. Plus le niveau est haut et plus n est grand, plus la rupture est longue.

I.1.2.2. Contrôle du flot d'information :

La progression simultanée de plusieurs opérations (sous-tâche, instruction, microinstruction, etc...), lancées successivement par l'unité de commande en chaque cycle mineur et susceptible d'avancer par différents chemins à différentes vitesses, demande le contrôle du flot d'information. Par exemple la figure I.4. illustre des instructions différentes, déclenchées par l'unité contrôle chaque 30 ns (cycle mineur). Elles parcourent différents chemins dans un pipe-line.

L'instruction 3 n'a pas besoin de calcul d'adresse et de recherche d'opérande. Les instructions 2 et 1 passent par différentes unités arithmétiques. Le contrôle du flot d'information au niveau instruction est une question délicate, mais pas insurmontable en utilisant des codes d'opérations différents. Au niveau sous-tâche, le contrôle est assez difficile pour la machine universelle, car il y a de nombreuses instructions et données à contrôler.

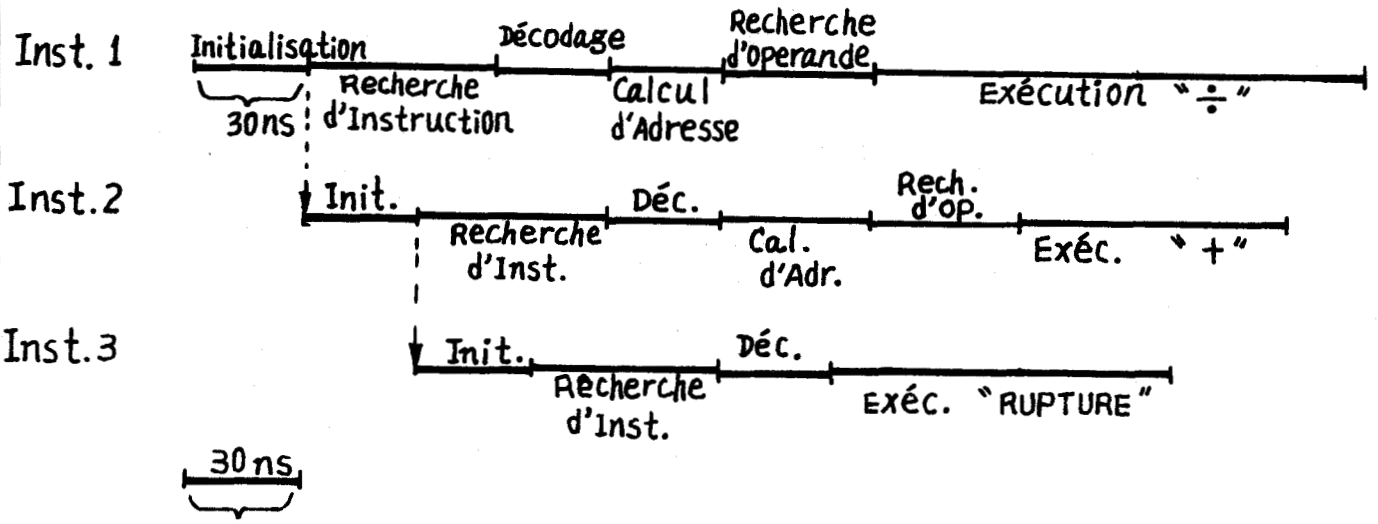


Figure I-4

I.1.2.3. Les conflits :

Le problème de conflit se pose dans deux cas : le conflit d'accès à une même unité et le conflit de dépendance de données.

Puisque chaque opération simultanée progresse par différents chemins à différentes vitesses, il est toujours possible que plusieurs unités en amont envoient respectivement une demande de transfert de données vers une même unité en aval qui est occupée par une demande antérieure. Pour éviter de bloquer le pipe-line, il faut intercaler des fils d'attente entre les unités en amont et celles en aval et introduire des circuits d'arbitrage. La figure I.5. illustre une solution.

Si on considère une machine pipe-line au niveau sous-tâche, le problème est beaucoup plus complexe que dans le cas de l'instruction. Par exemple, la file d'attente devient en fait un bloc mémoire intermédiaire.

Le conflit de dépendance de données vient du branchement conditionnel, de la dépendance d'opérande et de la dépendance de résultat intermédiaire.

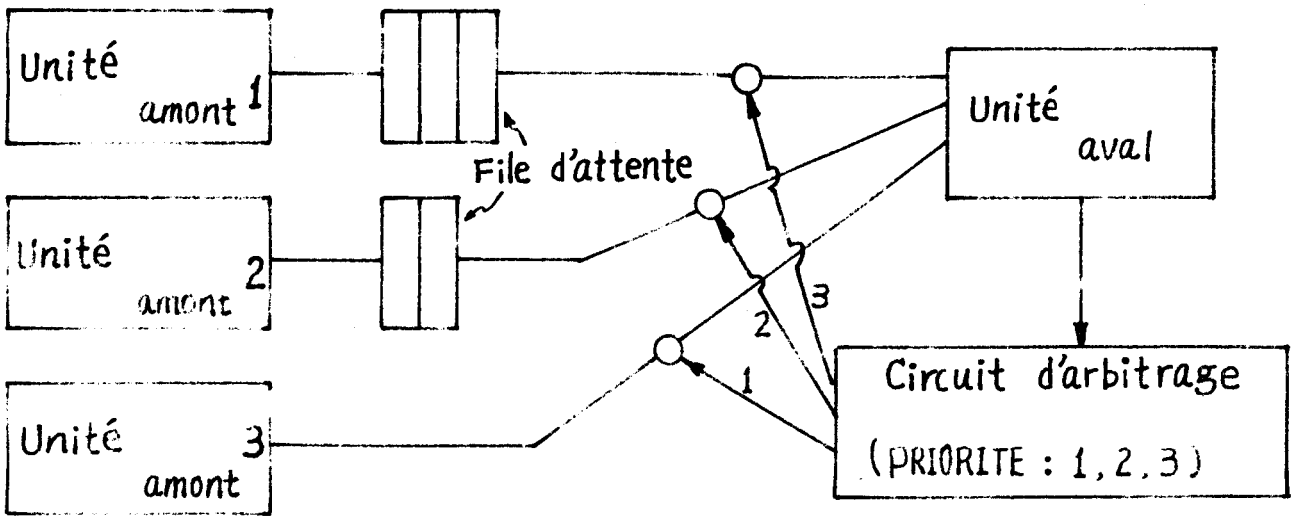


Figure I-5

Dans le cas de branchement conditionnel, tant que la condition n'a pas été produite, le résultat du test de cette condition et par conséquent le branchement sont indéterminés même incorrects. La résolution de ce conflit est soit de rompre le pipe-line pour respecter l'enchaînement logique du programme, soit de parier une possibilité, soit de dérouler toutes les branches possibles, enfin choisir une branche correcte.

Dans le cas où une opération va modifier des données et où l'autre opéation a besoin de ces données, la seule manière de résoudre le conflit est de respecter l'enchaînement logique du programme, même au prix de rupture de l'ordre naturel du pipe-line.

Dans le cas de dépendance de résultat intermédiaire, une opération produit des résultats dans une mémoire intermédiaire, l'autre prend l'opérande de cette mémoire. Donc, l'écrivain et le lecteur sont en conflit sur une même mémoire intermédiaire. Si l'écrivain fournit immédiatement le résultat au lecteur, le conflit est réglé.

En bref, il s'agit de plusieurs producteurs (écrivains) et de plusieurs consommateurs (lecteurs) qui de façon dépendante accèdent à une même ressource (données). Au niveau instruction, des solutions ont été proposées. A la machine pipe-line au niveau sous-tâche, on ne peut pas dire que le problème soit résolu /8/.

I.1.3. Les problèmes de parallélisme dans les machines SIMD

Jusqu'à maintenant, trois catégories principales ont été proposées : La Machine de Holland, La Machine de Solomon et le Multiprocesseur Distribué comme une machine cellulaire.

L'organisation de Holland a une tendance plus théorique que pratique.Il faut une très grande quantité de matériel qui est mal utilisé et il est difficile de définir un langage de commande efficace.

Le Multiprocesseur Distribué est en plein développement. Il semble avoir un certain avenir.

La machine de Solomon est déjà développée par plusieurs sociétés. Les plus célèbres sont **ILLIAC IV, BURROGS BSP, STARAN, PROPAL 2*, etc...**Elles semblent ouvrir une des rares voies permettant, à technologie égale, de gagner en performances par rapport aux machines pipe-line actuelles pour des problèmes particuliers. Mais il y a beaucoup de problèmes posés par son organisation - SIMD (Simple-Instruction Multiple - Données). Nous les récapitulons ci-dessous :

1) Le type de traitement dans lequel un flot d'instructions contrôle plusieurs flots de données, ne se rencontre guère que dans les problèmes de type matriciel. En effet, seulement dans les opérations matricielles (ou vectorielles) les trois conditions d'utilisation des processeurs cellulaires à très haute puissance se réalisent :

- **Opérations identiques** : elles peuvent donc être commandées par une instruction unique globale.

- **Opérations indépendantes** : toutes les opértions, exécutées par les processeurs cellulaires, peuvent se dérouler sans attendre le résultat de l'une pour exécuter l'autre.

- Opérandes "bien rangées" en mémoires : les éléments successifs des matrices (ligne et colonne) sont rangés dans des blocs successifs à des adresses calculées simplement pour qu'une seule instruction puisse les localiser par son champ d'adresse.

Ces machines fatalement ont perdu leur caractère **"universel"**.

2) Le nombre de processeurs est toujours difficile à définir, même si la machine traite seulement de problèmes matriciels.

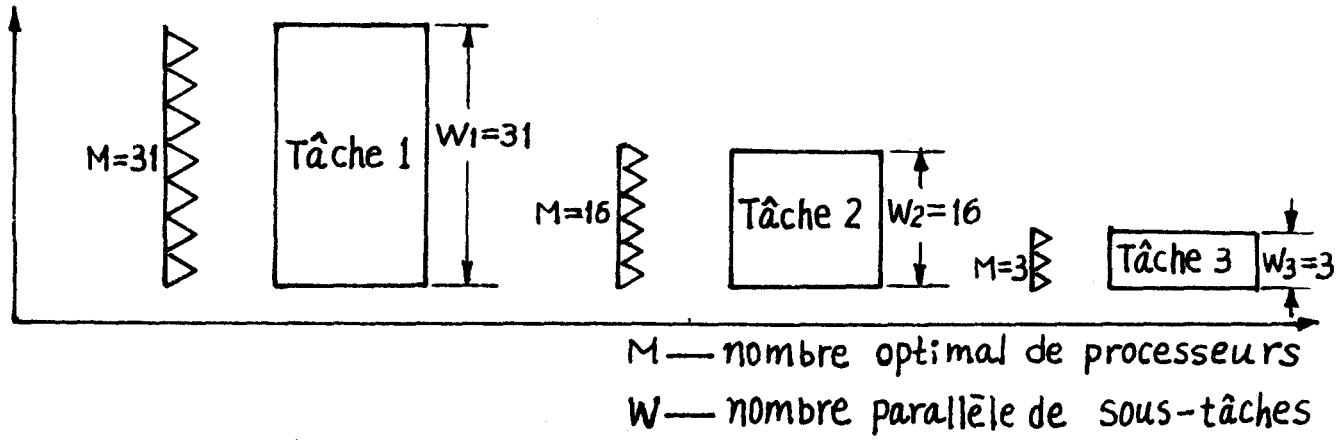


Figure I-6

La figure I.6. illustre un cas de décision délicate. La machine va rencontrer trois tâches, chacune travaille avec une largeur différente de matrice. Si l'on utilise 31 processeurs cellulaires, l'efficacité est très base pour la tâche 3. Si l'on utilise 3 processeurs, la vitesse de traitement est très lente pour la tâche 1. Donc la performance en vitesse et l'efficacité d'utilisation des processeurs sont toujours en conflit.

I.2. Définition du parallélisme dans notre système :

I.2.1. Le principe de fonctionnement :

Le but du travail dans ce mémoire est de tenter d'établir la structure d'un système, schématisé figure I.7., avec d'importantes possibilités de parallélisme.

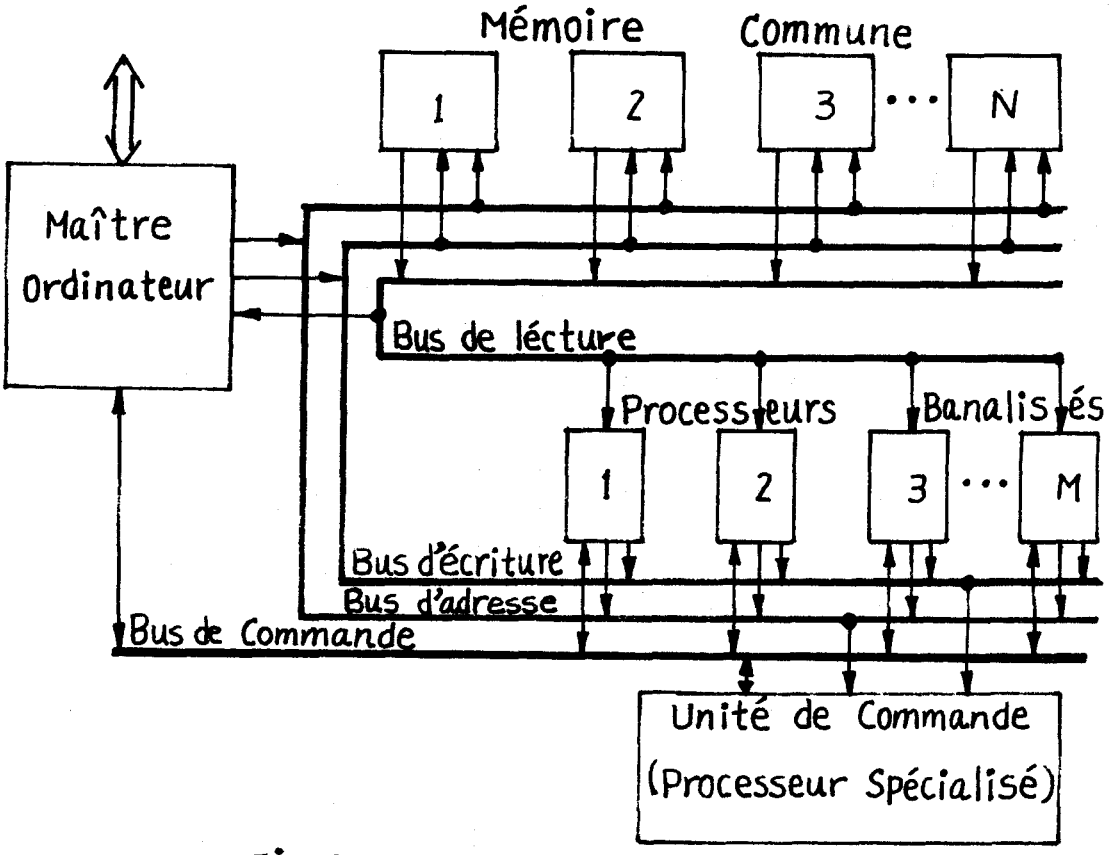


Figure I-7

I.2.1.1. Le maître ordinateur :

Le système d' exploitation réside dans le maître ordinateur.

Il joue donc un rôle important :

- Toute compilation et translation de programme d'utilisateur sont exécutées par lui.
- Les échanges d'information avec l'extérieur sont réalisés par lui.
- Il organise le flot de tâches, teste le système, etc....

I.2.1.2. La mémoire commune :

Le programme objet en binaire est chargé de cette mémoire. Pour obtenir une haute vitesse d'accès, elle est divisée en N blocs physiques indépendants, appelés Blocs. On l'appelle encore mémoire entrelacée. On est obligé d'ajouter des circuits d'arbitrage afin de régler le conflit dans le cas où plusieurs processeurs accèdent simultanément à un même bloc.

I.2.1.3. Les processeurs banalisés :

Il y a M processeurs identiques qui possèdent UAL (Unité Arithmétique et Logique) et UC (Unité de Contrôle). Chaque processeur peut accepter une sous-tâche, précisée par l'adresse initiale fournie par l'unité de commande de calcul parallèle (UCP) à condition qu'il ne soit pas actif. Il travaille alors sur cette sous-tâche jusqu'à sa fin où il envoie un prédicat correspondant à l'UCP et attend un nouvelle sous-tâche.

C'est donc un système MIMD (Multiple Instructions Multiple Données) au niveau processeur - processus (sous-tâche).

I.2.1.4. L'Unité de commande de calcul parallèle :

Le but de ce mémoire est surtout d'examiner la conception d'une unité de commande pour contrôler le calcul parallèle. Nous le décrivons donc aux paragraphes suivants.

I.2.2. Le cahier des charges de l'unité de commande

L'unité de commande doit satisfaire les demandes suivantes

- Toutes les sous-tâches qui sont rangées en mémoire commune sont dépendantes ou indépendantes. L'unité de commande doit contrôler le flot de tâches (ou sous-tâches) et assumer les dépendances logiques entre elles afin d'obtenir le résultat correct.

- N'importe quelle sous-tâche qui est exécutable peut être exécutée par n'importe quel processeur inactif. Le système possède donc une haute disponibilité.

- Si plusieurs sous-tâches sont en conflit sur un seul processeur inactif, l'unité de comande doit régler le conflit d'après des règles de priorité.

- Si plusieurs processeurs inactifs sont en conflit sur une seule sous-tâche exécutable, l'unité de commande doit le régler également.

- L'unité de commande peut organiser les sous-tâches et le travail des processeurs d'après un principe pipe-line, parallèle ou mixte sans avoir à effectuer de changement matériel. Le système est donc de nature universelle.

I.2.3. La description et la réalisation

Nous pouvons succinctement réduire le système à la figure

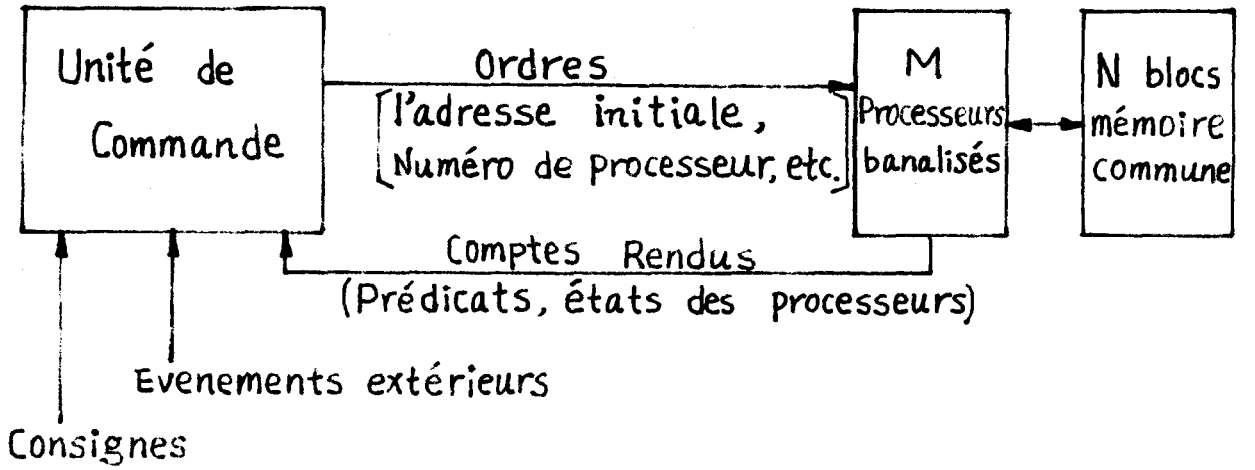


Figure I-8

Cette décomposition nous permet de traiter les processeurs et les exécutions des sous-tâches comme des processus physiques à automatiser et l'unité de commande comme un automatisme qui élabore en sortie des ordres destinés aux processus en fonction des compte-rendus venant des processus et en fonction des événements extérieurs.

Le GRAFCET étant un outil puissant pour décrire le genre d'automatismes logiques, sera envisagé comme outil de description de la solution des problèmes à résoudre.

Nous supposons définis et réalisés la mémoire, les processeurs banalisés etc...et consacrons le reste de notre mémoire à la réalisation de l'unité de commande.

Ce problème se décompose en deux parties : implantation du GRAFCET de contrôle en circuit matériel, programme de gestion de l'évolution du GRAFCET.

I.3. Conclusion :

Nous nous sommes proposés de concevoir une unité de commande qui contrôle un système de calcul parallèle, possédant des performances élevées :

- Le parallélisme est du type pipe-line, parallèle et mixte.

- L'utilisation de processeur banalisé fait que quand un processeur quelconque tombe en panne, il y a seulement dégradation des performances du système. La disponibilité est assez grande.

- La gestion du parallélisme est fait en GRAFCET. Le changement de problème à traiter n'entraine qu'un changement du programme représentant le GRAFCET.

- C'est un système universel de haut niveau dont nous avons précisé les difficultés de mise en oeuvre dans ce chapitre.

CHAPITRE II : OUTIL DE DESCRIPTION

Dans la phase de description d'un nouveau système de traitement numérique,il est important de disposer d'un outil de description fonctionnelle qui ne nécessite pas la connaissance des détails matériels de réalisation.En effet l'ingénieur ne precise la repartition entre partie cablée et partie programmée qu'après simulation et émulation du système à chacun des niveaux.Le GRAFCET "GRAphe Fonctionnel de Commande Etape-Transition" nous semble être un outil général de description de la partie contrôle d'un système quel que soit le mode de réalisation : cablé, programmé ou cablé programmé (firmware).

II.1 - Le GRAFCET

II.1.1 - Définition du GRAFCET

LE GRAFCET est un graphe orienté défini par un quadruplet $\{E,T,A,M\}$. On appelle respectivement E l'ensemble des étapes,T l'ensemble des transitions,A un ensemble d'arcs orientés et M un sous ensemble d'étapes initiales.

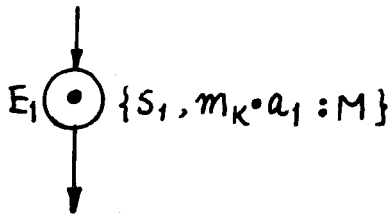
Le graphe est muni d'une inteprétation c'est-à-dire qu'aux étapes,sont associées des actions et aux transitions des réceptivités.

II.1.1.1. Les étapes E

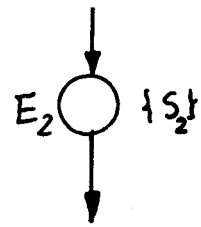
$E=\{E1,E2,...Ep\}\neq 0$ est un ensemble fini,non vide,d'étapes représentées par des cercles.Une étape peut être active ou inactive.On met un point (marqueur) dans le cercle d'étape active.Initiativement une étape au moins est active.Au cours du temps,l'activité des étapes se modifie senon les règles d'évolution définies plus loin.

La figure II.1. illustre la notion d'étape (active ou inactive) et les actions associées.

En effet aux étapes sont associées des actions conditionnelles et/ou inconditionnelles à effecter.Les conditions sont des fonctions logiques complexes des entrées du système,eventuellement d'autres commandes et également de activité des étapes du GRAFCET.Par exemple,à l'étape E1 de la figure II.1. est associée une action inconditionnelle S1 et une action conditionnelle M (Si $m_K \cdot a_i=1$ alos M=1) où m_K est une variable interne caractérisent l'activité de l'étape Ek et a une variable d'entrée.



1) Etape 1 active :
$S_1 = 1$
$M = 1$ (Si $m_K \cap a_1 = 1$)



2) Etape 2 inactive :
$S_2 = 0$

Figure II-1 Etape active et étape inactive

II.1.1.2. Les Transitions T

$T = \{t_1, t_2, \ldots t_q\} \neq 0$ est un ensemble fini, non vide, de transitions représentées par des tirets, auquelles sont associées des réceptivités, fonctions logiques de variables d'entrée et de sortie du système (conditions ou evenements) mais aussi de l'activité du GRAFCET.

Aux transitions peuvent être également associées des actions inpulsionnelles inconditionnelles qui lancent ou arrêtent des processus lors du franchissement de la transition.

Par exemple, à la transition t1 de la figure II.2 est associée une réceptivité EV1 et un ensemble d'actions impulsionnelles At1 où

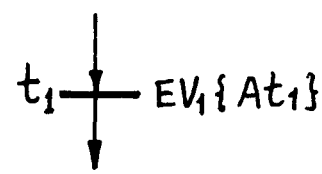


Figure II-2 Une transition

$$EV_1 = m_K \cap a \cap \overline{b} \cap \uparrow c$$
$$\{At_1\} = \{r_i, s_j, \overline{m}_K, OP_1\}$$

mk est une variable interne, représentant l'activité de l'étape EK et $\uparrow c$ est un evenement, variation d'état de 0 à 1 de la variable d'entrée c (front montant). $\downarrow C$ sera la notation d'un front decsentant.

L'action impulsionnelle $\overline{m_K}$ signifie la mise à zéro de mk lors du franchissement de t1. Ri, Sj et OP1 sont des mises à 1 d'actions.

II.I.I.3. Les Arcs A

$A = \{a1, a2, \ldots ar\} \neq 0$ est un ensemble fini, non vide, d'arcs orientés qui lie respectivement une étape à une transition et une transition à une étape. La figure II.3. montre l'alternance obligatoire étape-transition.

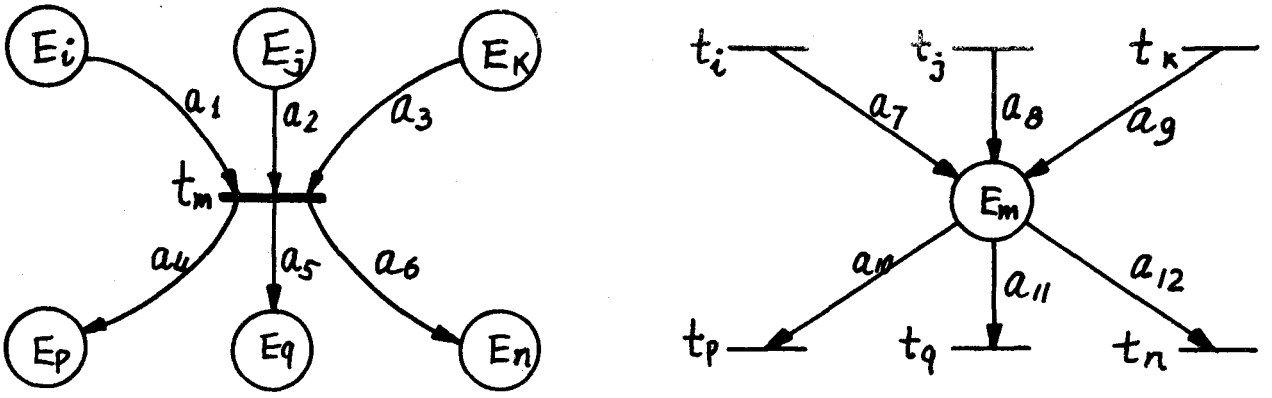


Figure II-3 Des arcs orientés : $\{a_1, a_2 \cdots a_{12}\}$

Les notations habituelles des relations entre étapes,transitions et arcs sont les suivantes,par exemple,pour la figure II.3. :

a_1={Ei,tm} indique une liaison allant de l'étape Ei à la transition tm

a_4={tm,Ep} indique une liaison allant de la transition tm à l'étape Ep

$^{\bullet}$Tm={Ei,Ej,Ek} représente les étapes d'entrée de tm

Tm$^{\bullet}$={Ep,Eq,En} représente les étapes de sortie de tm

$^{\bullet}$Em={ti,tj,tk} représente les transitions d'entrée de Em

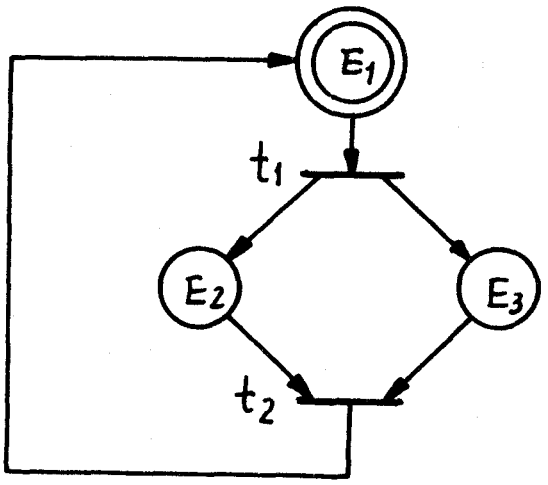
Em$^{\bullet}$={tp,tq,tn} représente les transitions de sortie de Em.

II.1.1.4 L'activité M

M={m1,m2,...mp} où mi=M(Ei)={0,1}

Si l'étape Ei est active,mi=1,sinon mi=0.L'activité M est donc la distribution des marqueurs dans un GRAFCET à un moment donné.

M0 est la distribution initiale des marqueurs:



$M_0 = \{1, 0, 0\}$

Figure II-4 Un GRAFCET marqué initialement

II.1.2 - Règles D'évolution du GRAFCET

L'évolution de l'activité dans le GRAFCET decrit le com-

portement du système représenté par le GRAFCET.

Les règles d'évolution sont suivantes:

II.1.2.1. Validation d'une transition

Une transition t est validée pour une activité M donnée si toutes les étapes d'entrée •t sont actives, c'est-à-dire:

$$\forall E \in {}^\bullet t \ , \ M(E) = 1$$

II.1.2.2. Franchissement d'une transition

Une transition peut être franchie si elle est validée et si la réceptivité associée est vraie.

Le franchissement d'une transition t entraine la désactivation de toutes les étapes d'entrée •t et l'activation de toutes les étapes de sortie t•; donc après le franchissement de t:

$$\forall E \in {}^\bullet t \ , \ M_j(E) = 0$$
$$\forall E \in t^\bullet \ , \ M_j(E) = 1$$
$$\forall E \notin \{{}^\bullet t \cup t^\bullet\} \ , \ M_j(E) = M_i(E)$$
$$\text{Si} \quad M_i \xrightarrow{t} M_j$$

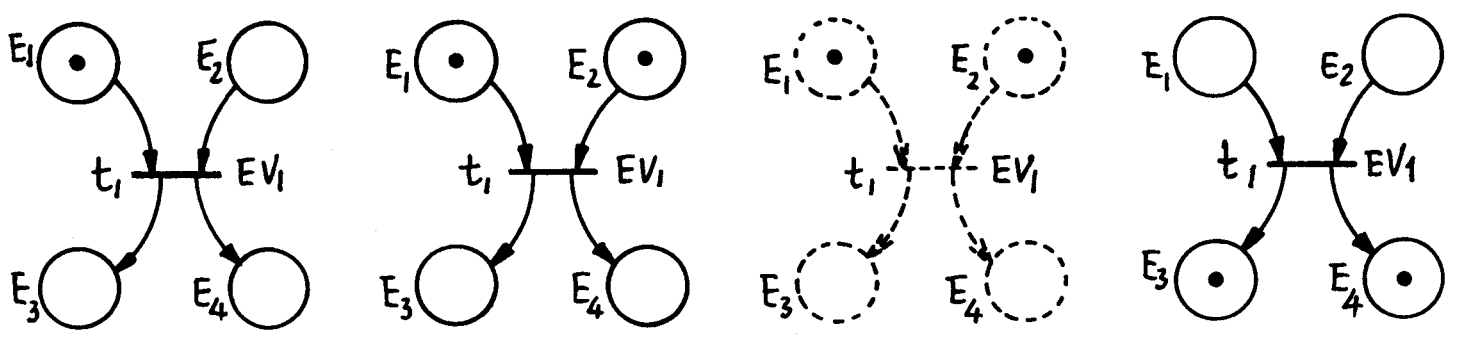


Figure II-5 Validation et franchissement d'une transition

Figure II.5a transition t1 non validée, l'étape 2 étant inactive.

Figure II.5b transition t1 validée, les étapes d'entrée •t1={E1,E2} sont actives, mais elle ne peut être franchie, la réceptivité étant EV1=0.

Figure II.5c si •t1 sont actives et EV1=1, la transition t_1 doit être franchie.

FIgure II.5d. Le franchissement de la transition t1 entraine l'activation des étapes $t_1^{\bullet}$={E3,E4} et la désactivation des étapes d'entrée.

Il y a de plus les deux règles suivantes:

--Plusieurs transitions simultanément franchissables sont simultanément franchies.

--Si une étape doit être désactivée et activée simultanément,elle reste active.

L'utilisation des règles complementaires nécessite une grande habitude,mais permet le parallélisme interprété qui donne des représentations plus concises.Les proprietés du GRAFCET peuvent être mis en détail en /1/ et /2/.

II.2 - Description Des Systèmes Informatiques En Terme De GRAFCET

II.2.1 - GRAFCET et Graphe D'état

Les méthode traditionnelles de synthèse des machines sequentielles traitent des graphes d' état.

II.2.1.1.Le RGAFCET comprend la notion de graphe d'état

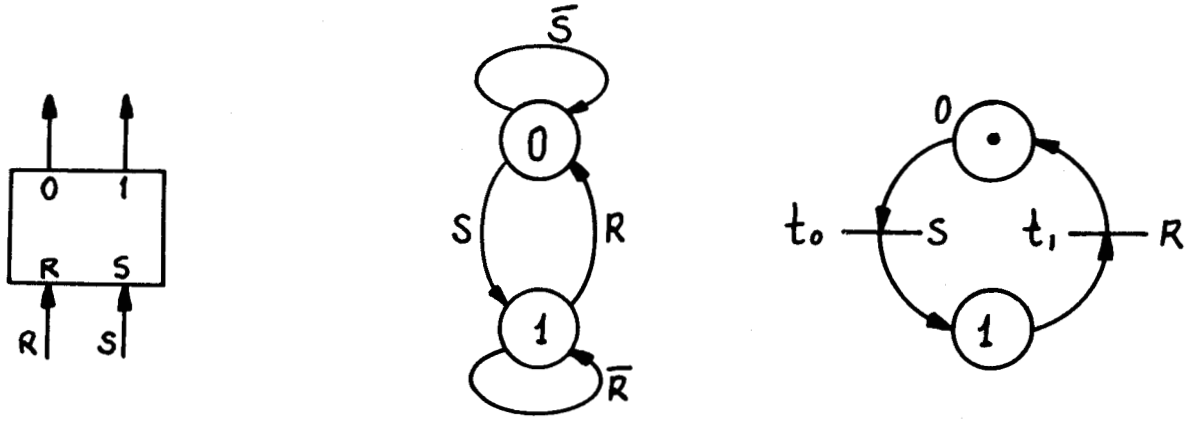
Dans la litterature,il est possible de trouver deux classes de graphes d'état correspondant aux deux définitions suivantes:

--Un graphe d'état est un GRAFCET à étape initiale unique tel qu'une transition ne possède qu'une étape d'entrée et une étape de sortie et qu'à un instant donné une étape et une seule soit activée.

--Un graphe d'état est un GRAFCET tel qu'à un instant donné une étape au plus soit active.Cette définition autorise donc les transitions sources et les transitions puits.

La grande différence entre les graphes Booléenes traditionnels,obtenus par la méthode d'Huffman et le GRAFCET tient dans la notion réceptivité.

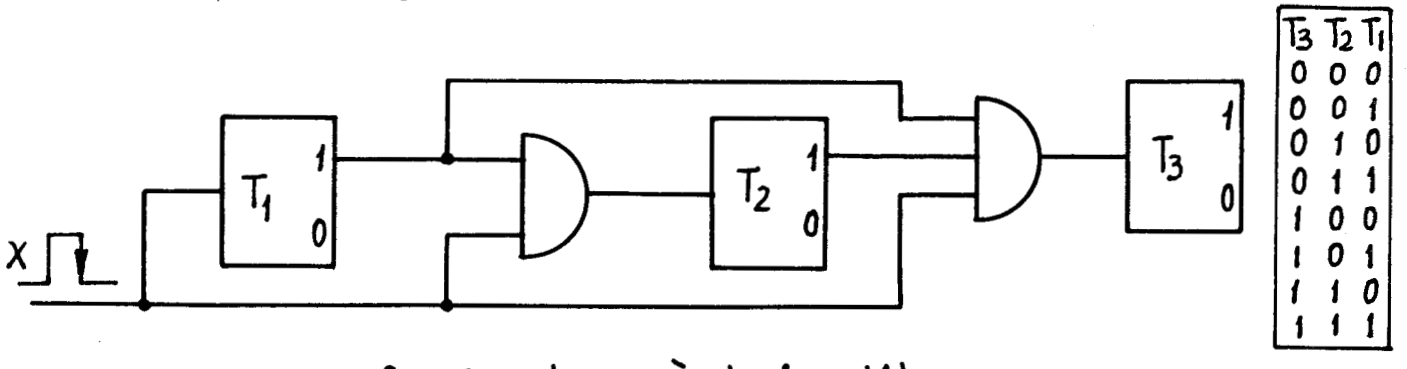
LES DEUX EXEMPLES SUIVANTS illustreront la différence de point de vue.La figure II.6. montre la description du fonctionnement d'une simple bascule R-S.



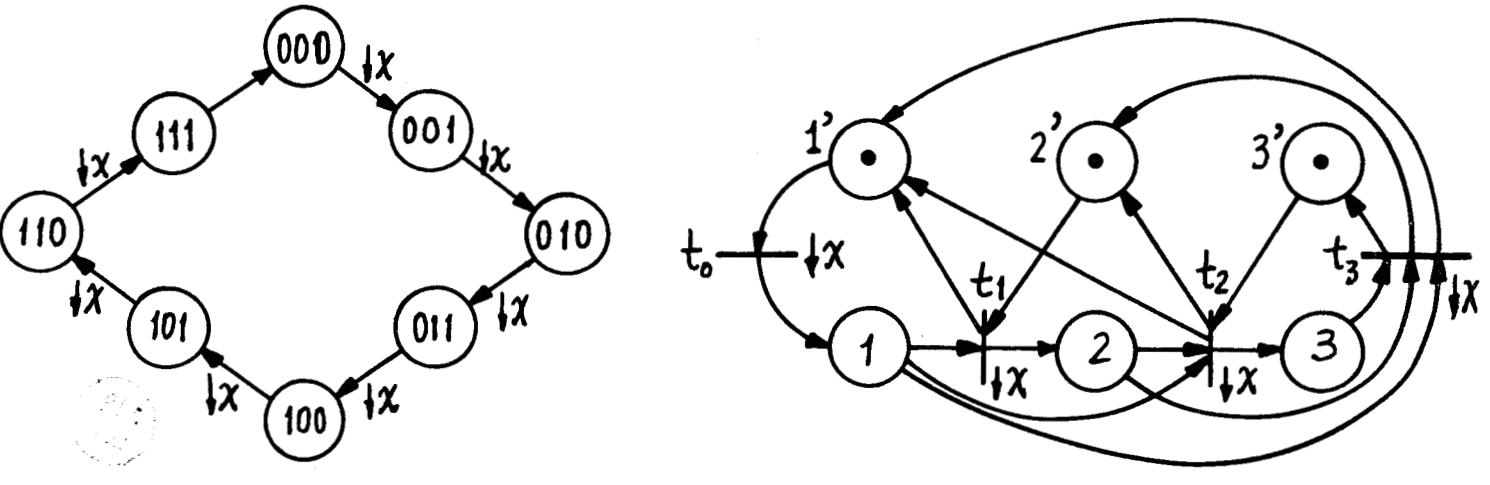
a) R-S bascule b) graphe d'état (R∩S=0) c) GRAFCET (R∩S=0)

Figure II-6 Description d'une R-S bascule

La figure II.7. decrit un compteur sur 3 bits.On peut remarquer l'analogie entre le GRAFCET et le schéma de réalisation.



a) Compteur à trois bits



b) description en GRAPHE d'état c) déscription en GRAFCET

Figure II-7 Description d'un compteur à trois bits

II.2.1.2. Le GRAFCET permet aussi le parallélisme

Le FRAFCET comporte les quatre principales structures de la figure II.8. dont c) et d) fondamentales pour decrire les systèmes avec parallélisme de fonctionnement.

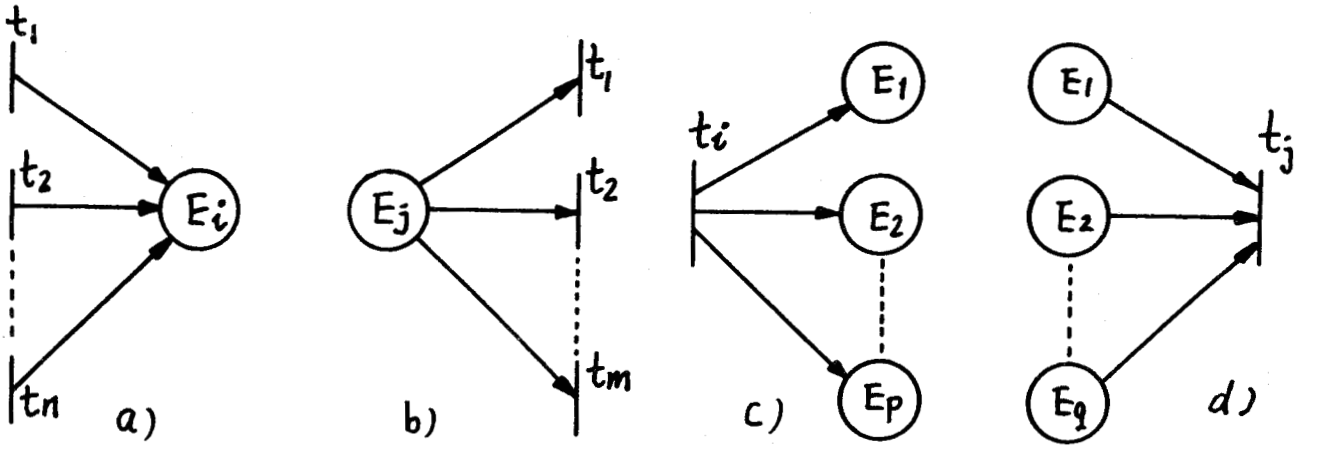


Figure II-8 Configurations Principales

II.2.2 - GRAFCET et Organigramme/1//9/

L'organigramme classique de l'informatique peut se ramener à un graphe d'état et donc facilement traduit en terme de GRAFCET. Par exemple les symboles de box de la figure II.9.se traduisent respectivement en ceux de la figure II.10.

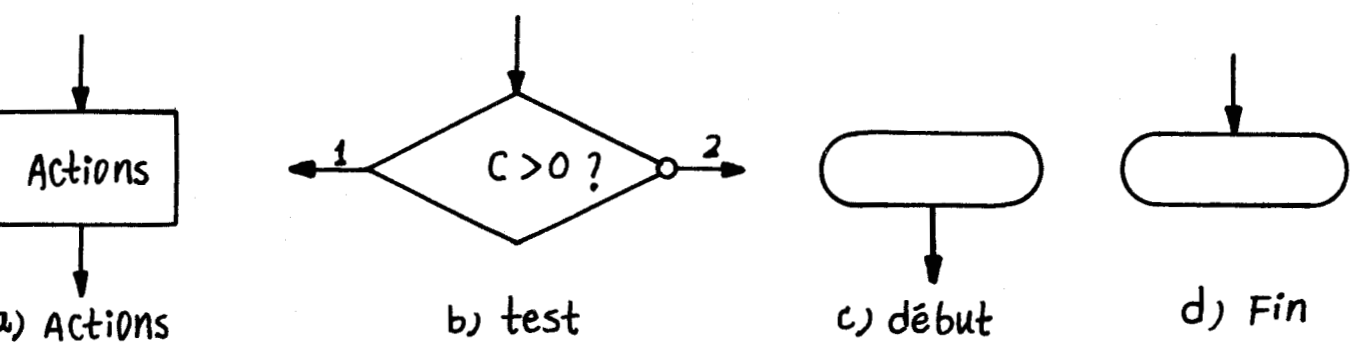


Figure II-9 Symboles essentiels d'un organigramme

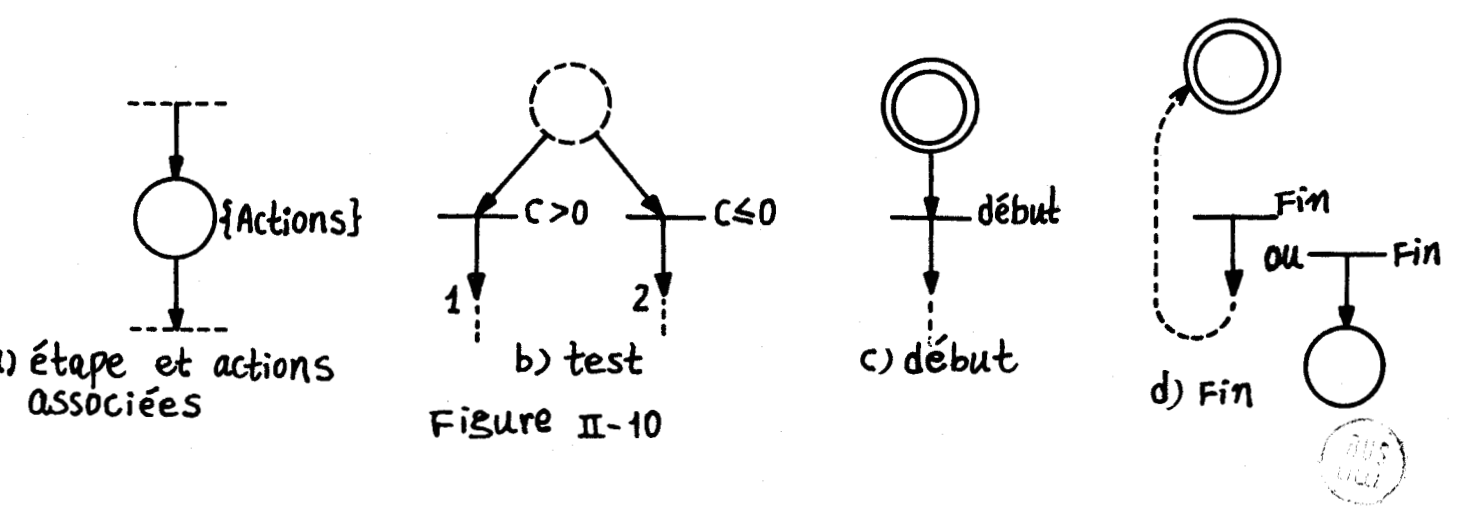


Figure II-10

Remarquons toutefois que la notion de test avec attente doit se faire avec bouclage sur le test (figure II.11)

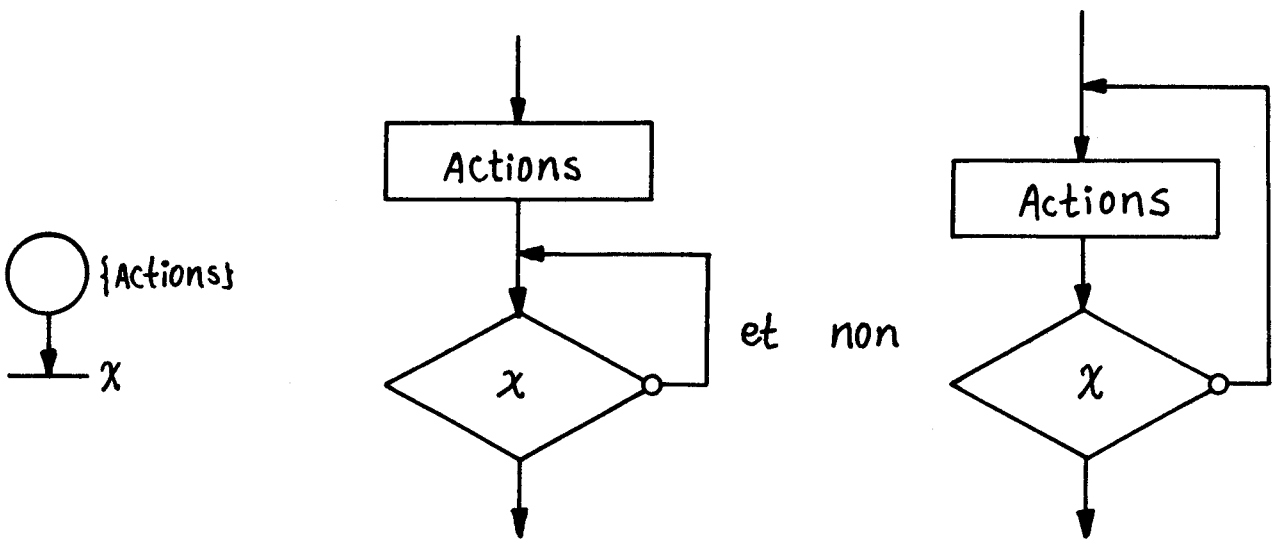
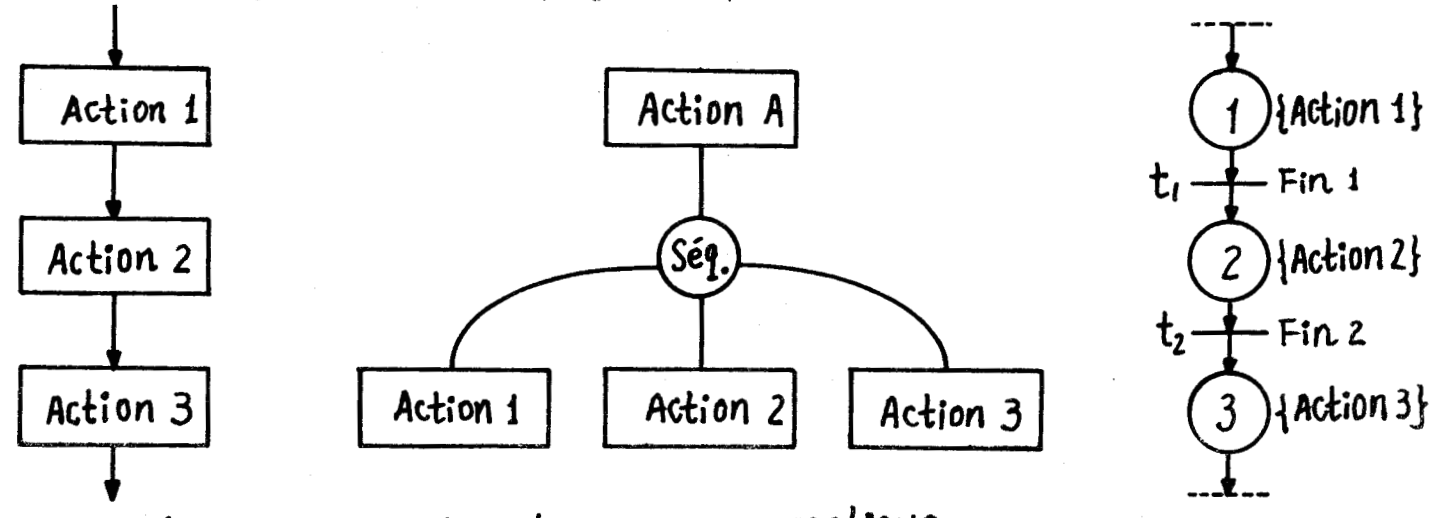


Figure II – 11

Considérons maintenant quelques configurations classiques intervenant dans les programmes.

II.2.2.1. Sequence d'actions

La figure II.12. montre une sequence de trois actions successives A_1,A_2,A_3constituant l'action A,suivant les trois écritures de l'organigramme,de l'arbre programmatique et du GRAFCET.



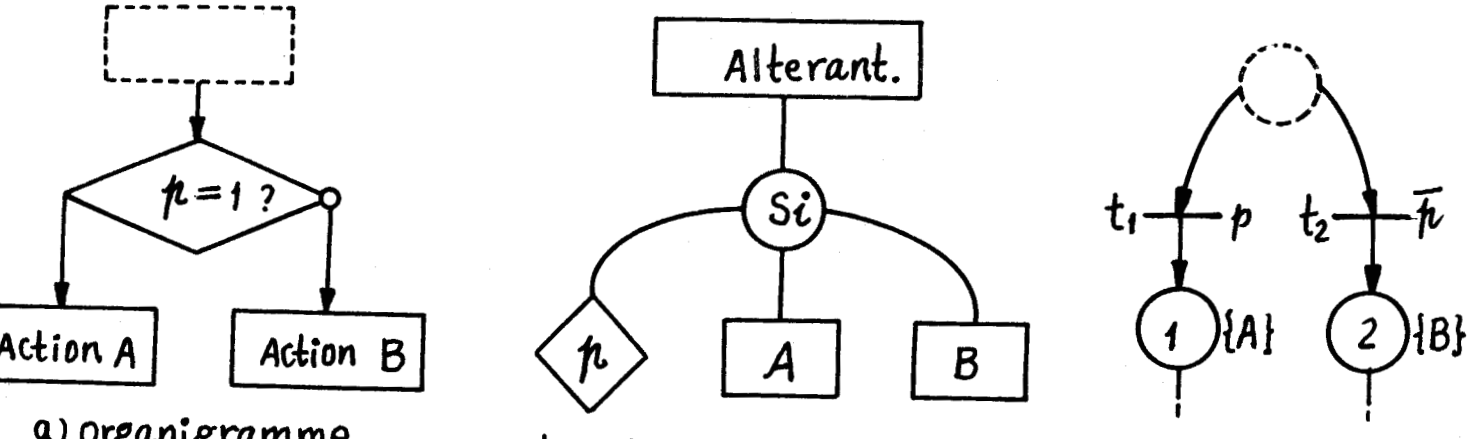
a) Organigramme b) arbre programmatique c) GRAFCET

Figure II-12 Séquence d'actions

Les réceptivités associées à $t_1$et t_2 sont les indications de fin d'action.

II.2.2.2. Alterative

Si le predicat P est vrai,il faut réaliser l'action A sinon réaliser l'action B. La figure II.13. donne les représentations.



a) Organigramme b) arbre programmatique c) GRAFCET

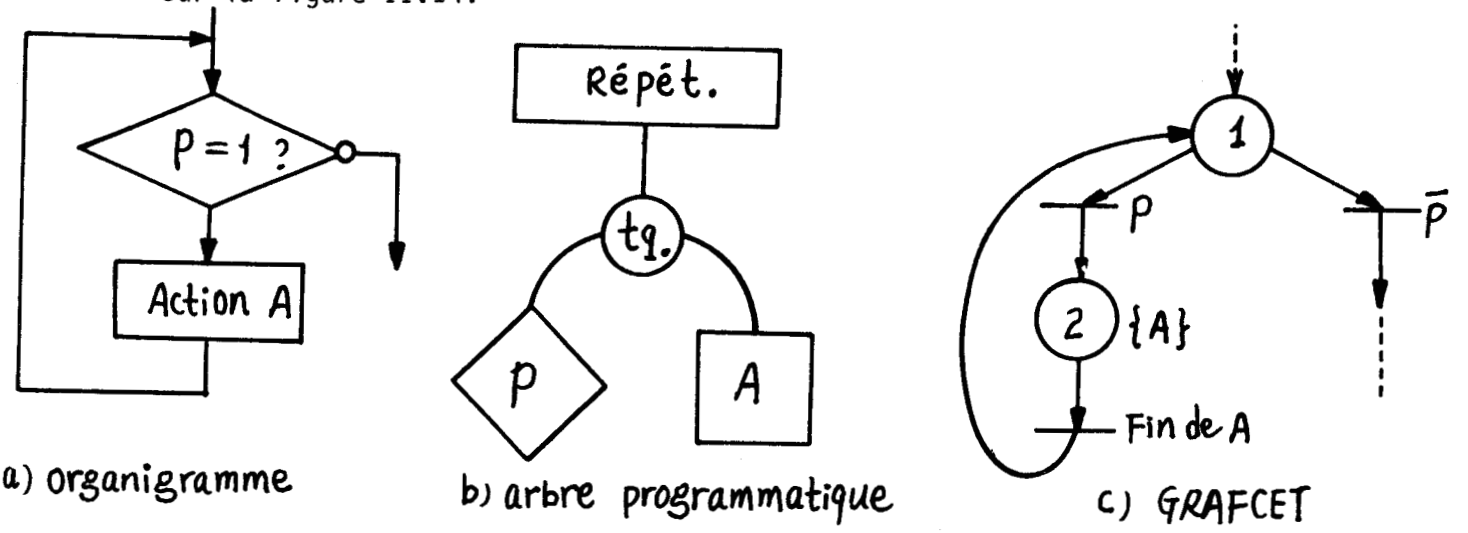
Figure II-13 Alternative

II.2.2.3. Itération (ou répétition)

L'itération peut se présenter sous deux formes:

--Tant_que

Faire l'action A tant que le predicat p est vrai comme on peut le voir sur la figure II.14.

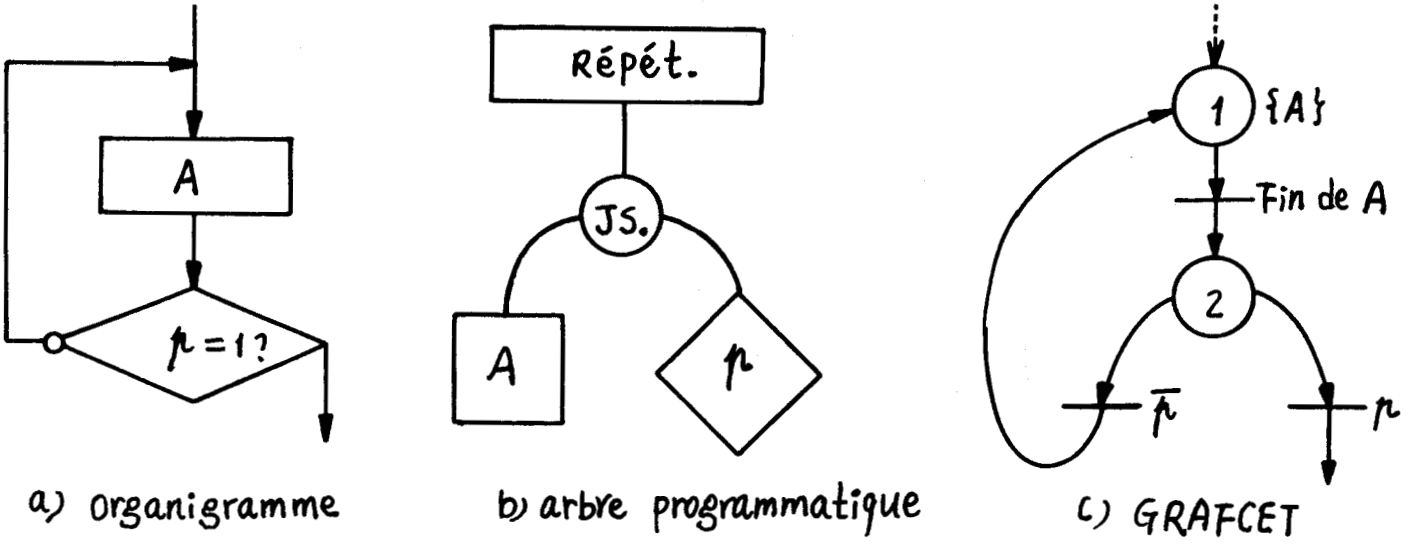


a) organigramme b) arbre programmatique c) GRAFCET

Figure II-14 Répétition — "tant que"

--Jusqu'à_ce_que

Faire l'action A jusqu'à ce que le predicat p soit vrai comme le montre la figure II.15

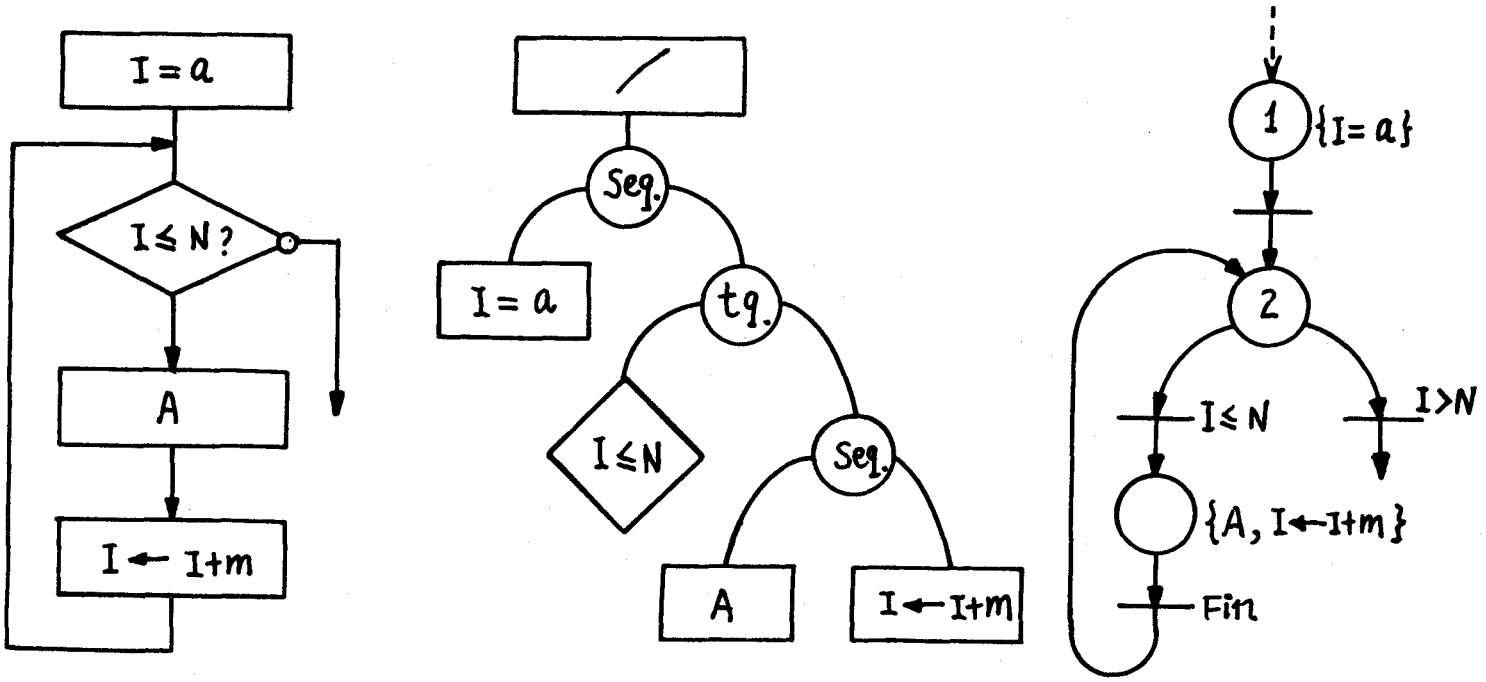


a) organigramme b) arbre programmatique c) GRAFCET

Figure II-15 Répétition — " Jusqu'à "

Les quatre figures : sequence,alternative,tant que,jusqu'à ce que sont les décomposeurs habituels de la programmation structurée.Dans certains cas, on admet une variante supplementaire de l'itération (figure II.16.)

Pour i=a à n, par pas m,faire l'action A:



a) organigramme b) arbre programmatique c) GRAFCET

Figure II-16 Répétition — "pour"

II.2.2.4. Sousprogramme et partage de ressource

Les divers sous-programmes, les données communes et les processeurs sont des ressources pour chaque programme utilisateur. Le partage de ressource entre plusieurs utilisateurs peut entrainer des conflits et donc des problèmes d'arbitrage.

La figure II.17. représente un appel de sous-programme,qui habituellement comporte une sauvegarde automatique de l'adresse de retour et se termine par une restitution de cette adresse.Sur la figure II.17.,l'étape E_2 sert de mémorisation et peut donc être omise en cas de sauvegarde inplicite.

Dans le cas du partage d'une ressource, d'un sous-programme non réentrant, on rencontre la configuration de la figure II.18.

Une étape Ed initialement active sert d'arbitre et indique la disponibilité de la ressource. Quand il y a appels simultanés, il convient d'introdruire des conditions d'arbitrage de façon à éviter le franchissement simultané de t_1 et t_2 .

Avec le GRAFCET , nous avons vu qu'il était possible de decrire les programmes classiques . Nous allons maintenant aborder quelques structures speciales de gestion de données .

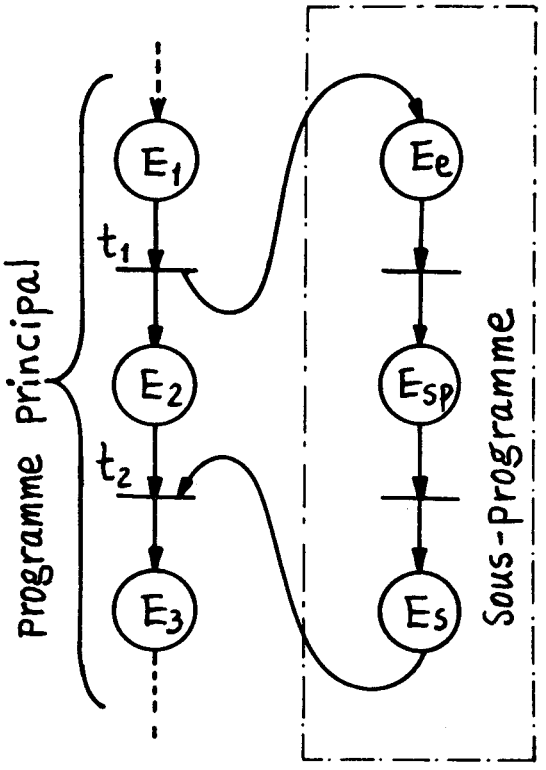


Figure II-17 l'appel de Sous-programme

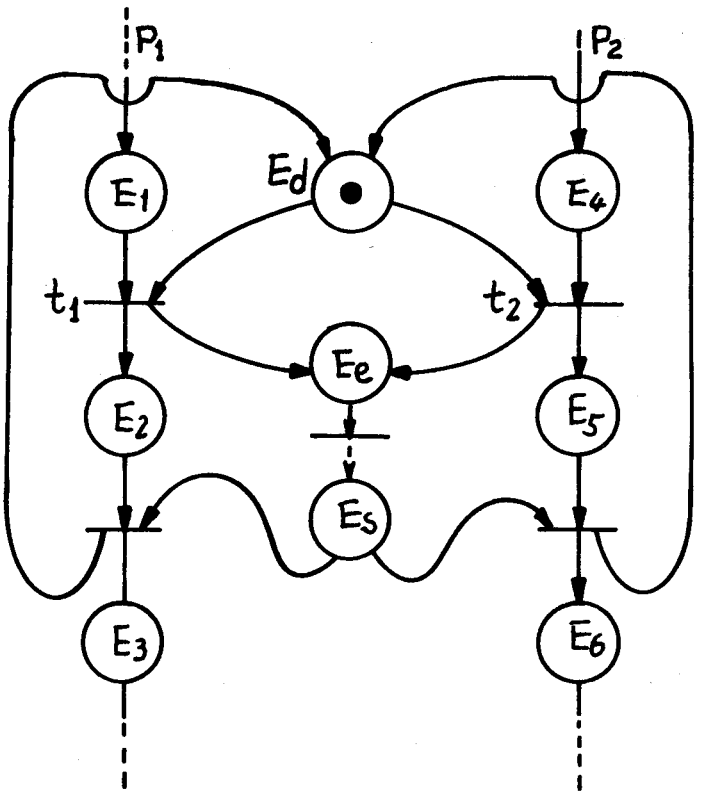


Figure II-18 Partage d'un Sous programme

II.2.3 - GRAFCET et Structure de Données /9/ /10/

II.2.3.1. Pile

Fondamentalement, une pile est mécanisme permettant de ranger des informations de telle sorte que l'information disponible soit toujours la dernière à avoir été rangée (LIFO) . Elle sert fréquemment et est réalisée soit par circuit logique, soit en logiciel. Sa description en terme de GRAFCET est donnée figure II.19.

Nous décomposons le mécanisme de la pile en deux sous-programmes " empiler" et "dépiler". Un programme utilisateur attend à l'étape E_1 pour empiler et à l'étape E_2 pour dépiler. Une étape E_d indique la disponibilité de la pile. Lorsque M est petit,le compteur de pile est intégré à la structure du GRAFCET (figure II.20).

Dans le cas , si E_d est active, on se trouve avec une pile vide et on ne peut plus dépiler. Si D_m est active , la pile est pleine et on ne peut empiler.

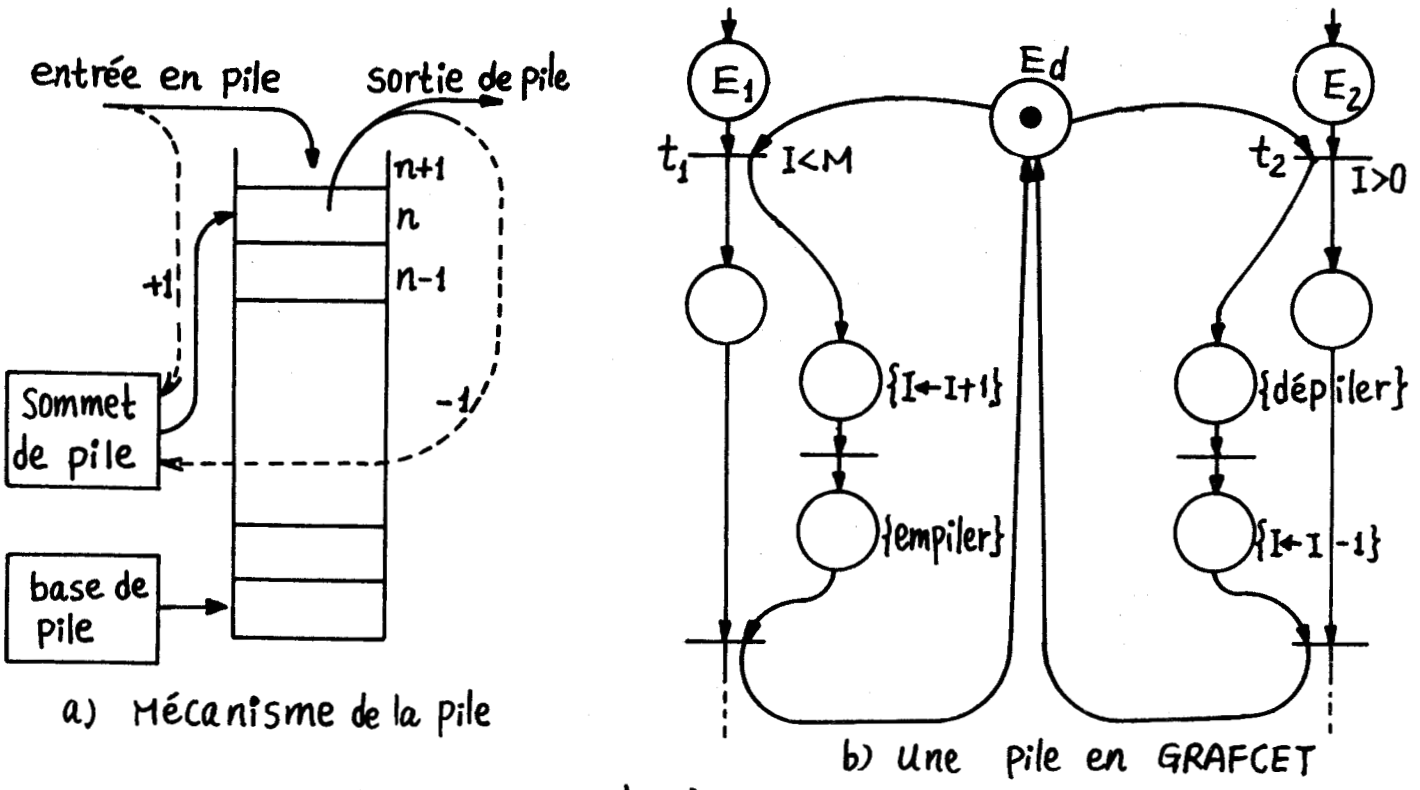


Figure II-19 une pile à M cases

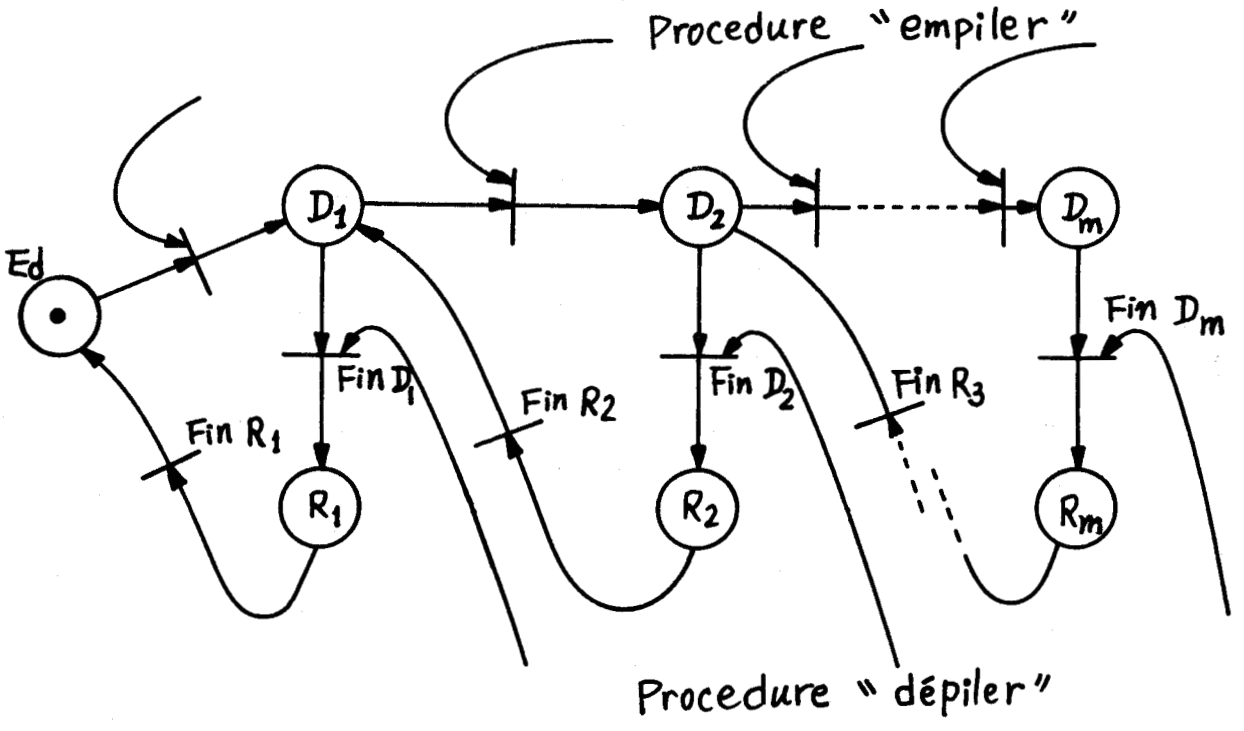


Figure II-20 Une pile à M cases

II.2.3.2. File d'attente circulaire

La file d'attente est aussi une succession de cases mémoires mais la discipline suivie est de type FIFO (premier entré , premier sorti) .

Comme le montre la figure II.21., un producteur ajoute un élément au bout de la file, indiqué par le pointeur P_2 tandis qu'un cosommateur prend l'élément en tête de file, pointée par P_1. Le GRAFCET est donné figure II.22.

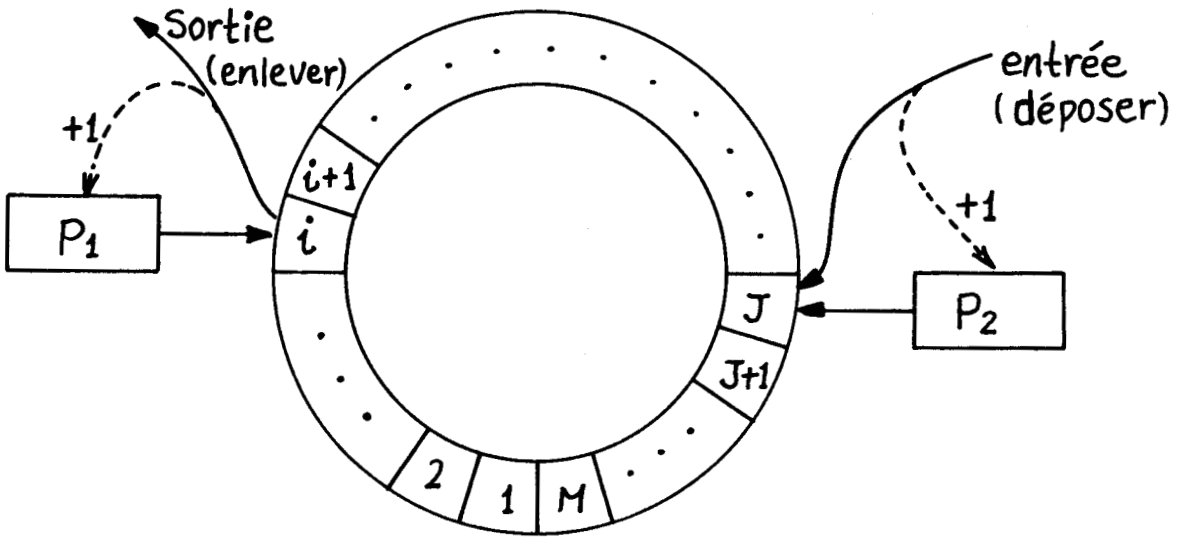
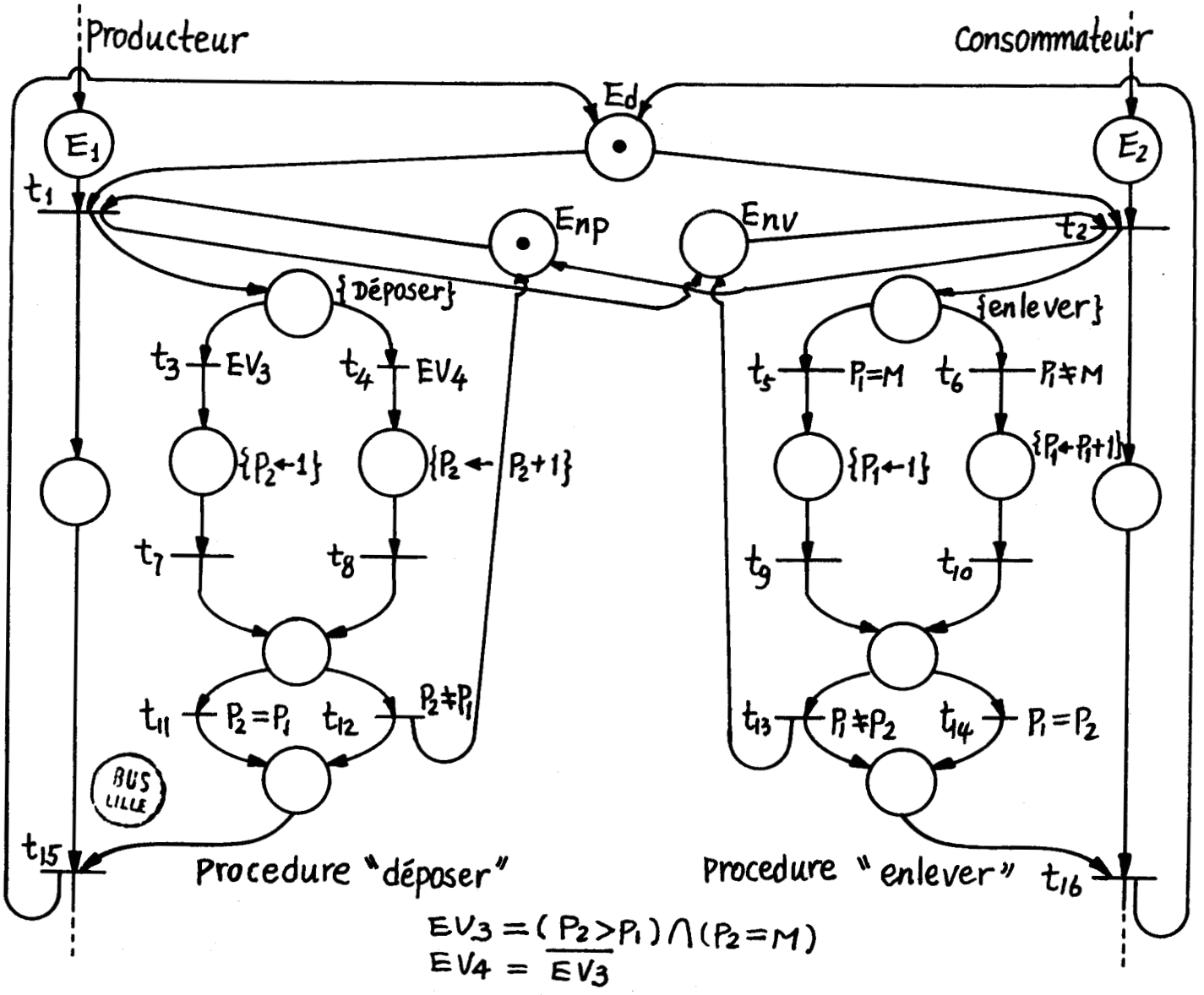


Figure II-21



$$EV3 = (P_2 > P_1) \wedge (P_2 = M)$$
$$EV4 = \overline{EV3}$$

Figure II-22 Une file d'attente à M cases

Sur cette figure, un programme d'utilisateur attend à l'étape E_1 pour déposer un élément et à l'étape E_2 pour enlever un élément . M est la capacité de la file . Une étape marquée E_d indique la disponibilité de la file. Quand l'étape E_{np} est active , la file n'est pas pleine et on peut déposer un élément. L'étape E_{nv} active correspond à une file non vide , on peut donc enlever un élément. Les étapes E_{np} et E_{nv} ne sont pas sauves.

Nous pouvons aussi établir un GRAFCET, pour la même file d'attente, avec compteur de file intégré au graphe (figure II.23.)

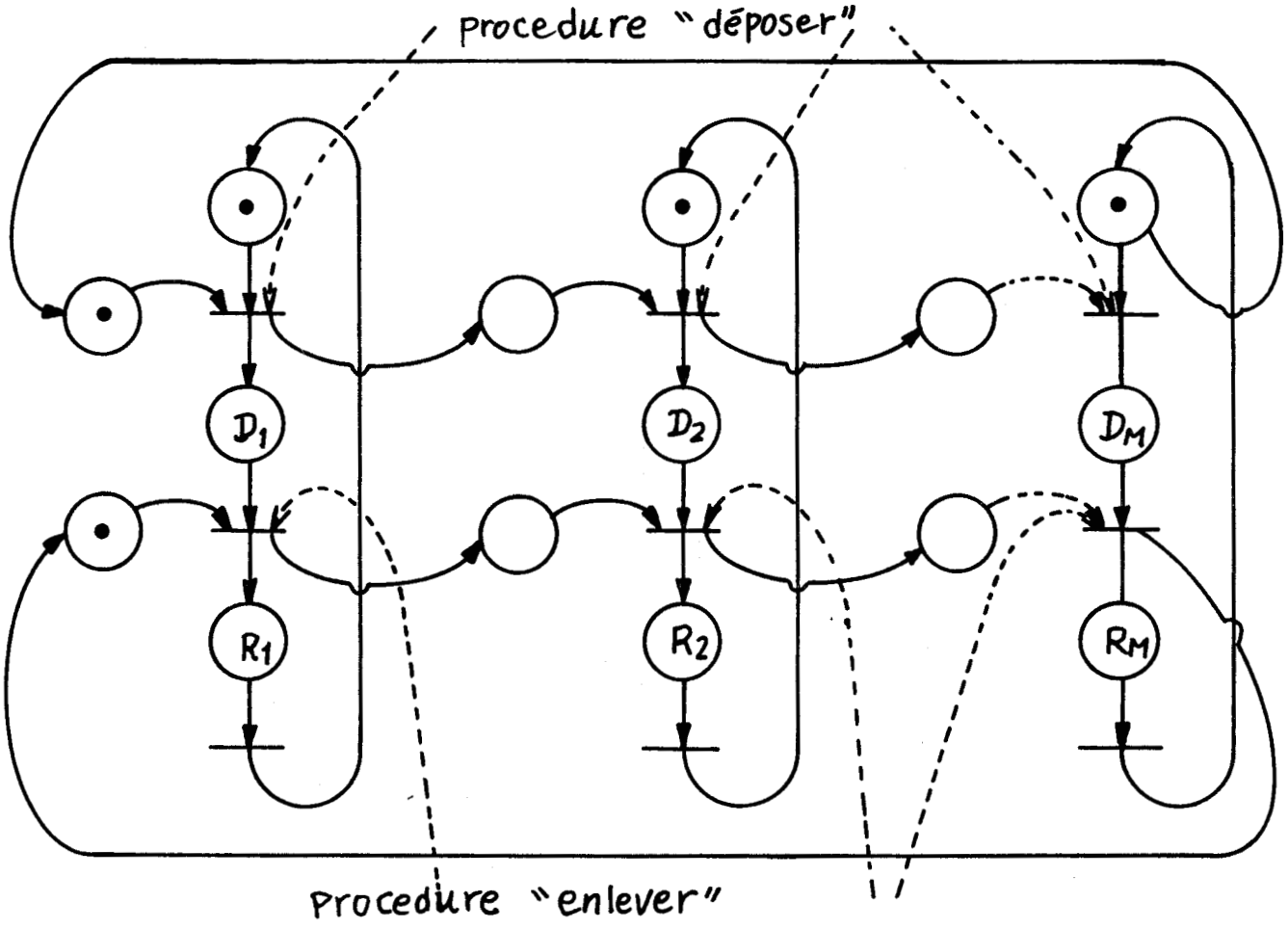


Figure II-23 Une file d'attente à M cases

Si toutes étapes $D_1 \ldots D_m$ sont actives, la file est pleine et on ne peut plus déposer. Si toutes les étapes $D_1 \ldots D_m$ sont vides, on ne peut plus enlever.

II.2.3.3. File d'attente en pipe-line

La figure II.24 représente un pipe-line à compteur intégé au graphe. T_i ne peut faire le transfert que si T_{i+1} est terminé .

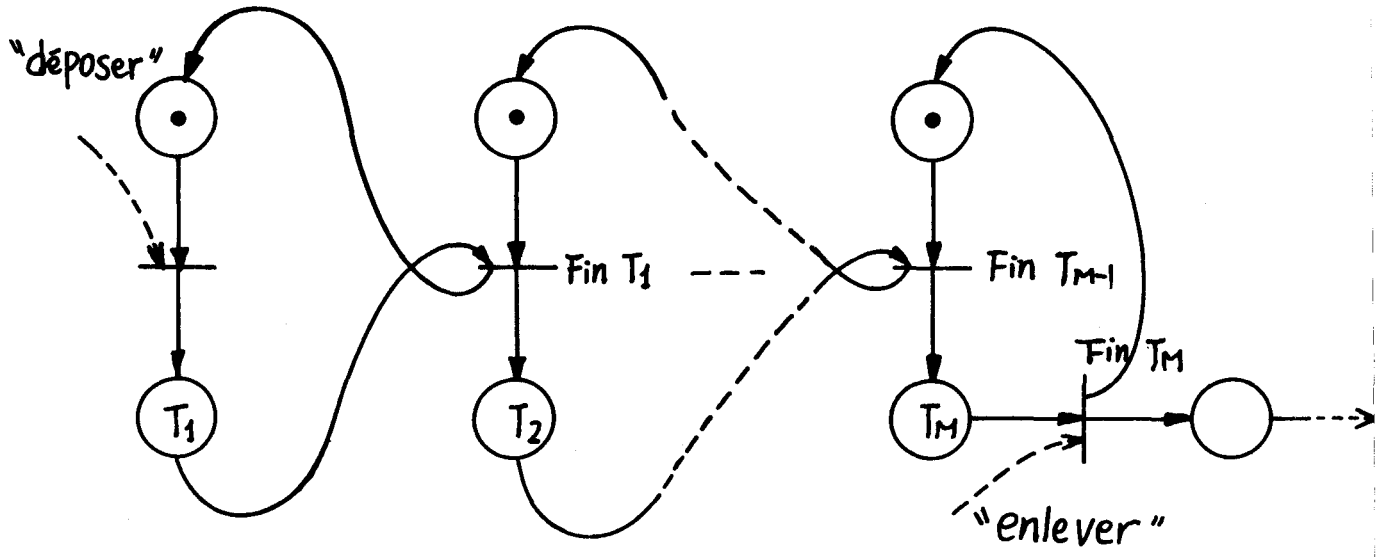


Figure II-24 Une file d'attente en pipe line

Un élément est déposé à une extrémité et enlevé à l'autre bout.

II.3 - Description de synchronisation

Dans ce paragraphe nous allons étudier la synchronisation et la coopération de plusieurs processus concurrents d'après leur dépendance à la fois dans le temps et dans l'espace. Deux processus doivent par exemple être exécutés l'un après l'autre car l'un utilise résultat de l'autre comme entrée. C'est une dépendance dans le temps.

Par ailleurs deux processus peuvent être exécutés indépendamment l'un de l'autre, mais sont seulement en conflit sur une même ressource; ils deviennent donc dépendants dans espace.

De nombeux outils de description /10/ 11/12/ existent pour decrire les problèmes de synchronisation; nous nous proposons d'utiliser le GRAFCET /1/ pour decrire la synchronisation d'evenements.

II.3.1 - Autosynchronisation et sychronisation extérieure

L'évolution dans un GRAFCET se fait par franchissement de transitions , auxquelles sont associées des réceptivités, fonctions de variables internes et externes. Les premières permettent la description des synchronisations entre étapes d'un GRAFCET ou entre différents GRAFCETs. Les autres assurent la synchronisation avec le monde extérieur.

Deux parties de GRAFCET se synchronisent suivant quelques-

unes des situations de la figure II.25. qui conduisent à des interactions momentanées ou permanentes, bidirectionnelles ou unidirectionnelles, parallèles ou sequentielles, compatibles ou exclusives.

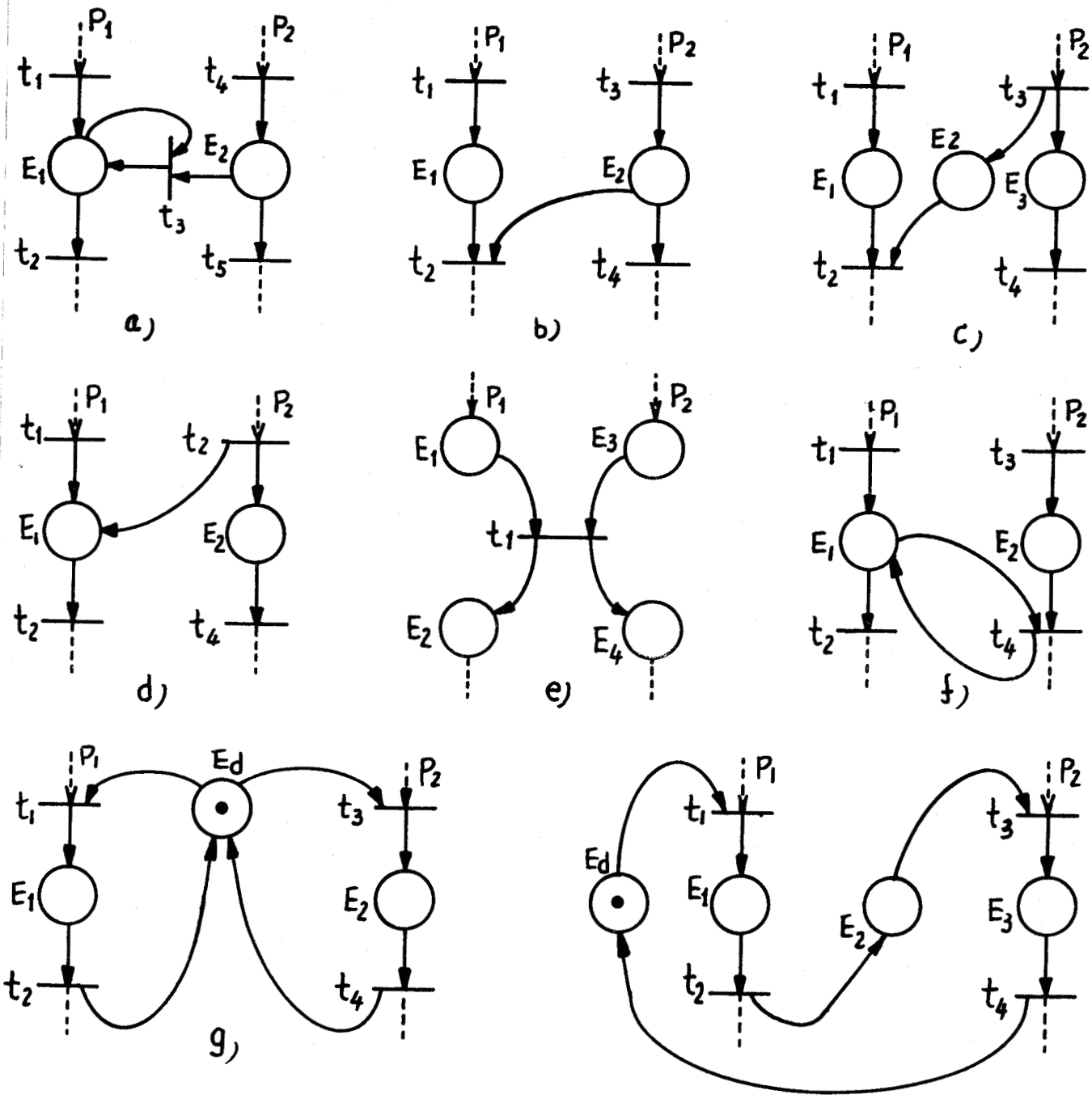


Figure II-25 quelques cas de synchronisation

La figure II.25 a correspond à la suppression d'un marqueur dans P_2, la figure II.25b à un contrôle du franchissement de la transition t_2 avec possibilité de conflit. Dans la figure II.25c, le franchissement de la transition t_3 autorise la validation de la transition t_2. La figure II.25$_d$

correspond au placement d'activité dans P_1. La situation de la figure II.25e est une autorisation mutuelle. La figure II.25f illustre le cas où le franchissement de la transition t_4 est momentanement autorisé par l'activité de l'étape E_1.

Dans le cas II.25g, on a deux sections critiques exclusives, quant à la figure II.25h, il y a section critique avec passage alterné.

Des mécanismes de synchronisation plus complexes peuvent être obtenus par combinaison de ces configurations de base.

II.3.2. - Synchronisation en temps réel

Les modeles que nous utilisons tiennent compte de l'opposition de signaux en provenance du monde extérieur. La notion de temps nécessite quelques precisions:

II.3.2.1. Synchronisation avec l'horloge en temps réel

--L'introduction d'horloges multiples ne pose pas de difficulté comme le montre la figure II.26.

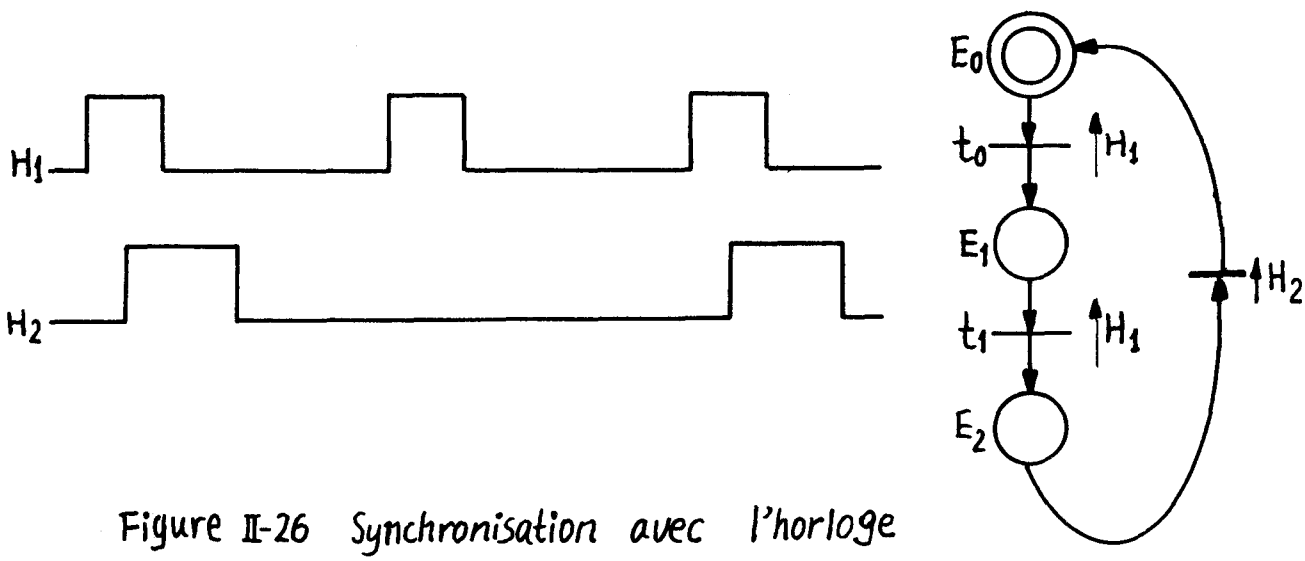


Figure II-26 Synchronisation avec l'horloge

--Synchronisation relativement à un intervalle de temps

La figure II.27 montre la mise en place de deux intervalles de temps τ_1 et τ_2.

La réceptivité $t/i/\tau$ signifie que la transition associée sera franchie τ unités de temps après l'activation de Ei. Nous pouvons utiliser les signaux $\uparrow H, \downarrow H$ et $t/i/\tau$ dans les diverses configurations de synchronisation de la figure II.25. Nous les appliquons par exemple aux cas de la figure II.25f et g. pour obtenir les figures II.28a et b.

En a), la transition t4 peut être franchie pendant τ unités de temps; en b) t_1 n'est plus conflit avec t_3.

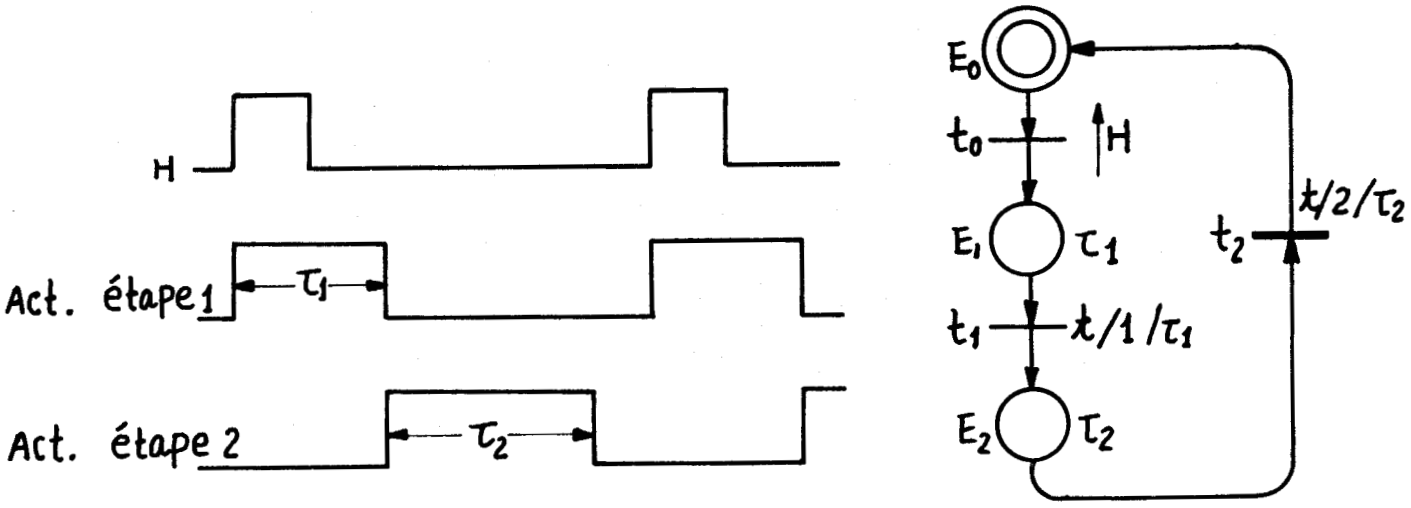


Figure II-27 Synchronisation avec intervalle de temps réel

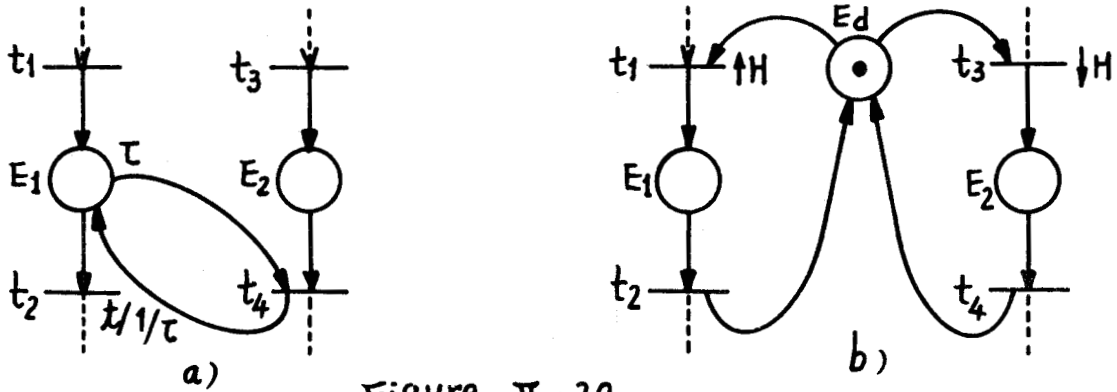


Figure II-28

II.3.2.2. Partage en temps réel

Soit un système d'exploitation avec partage du temps où N utilisateurs partagent une même ressource. Si la tâche Ti pose une demande de service (Ri), elle peut occuper la ressource pendant τ_i unités de temps, comme sur la figure II.29. Il s'agit ici d'une discipline à priorité fixée.

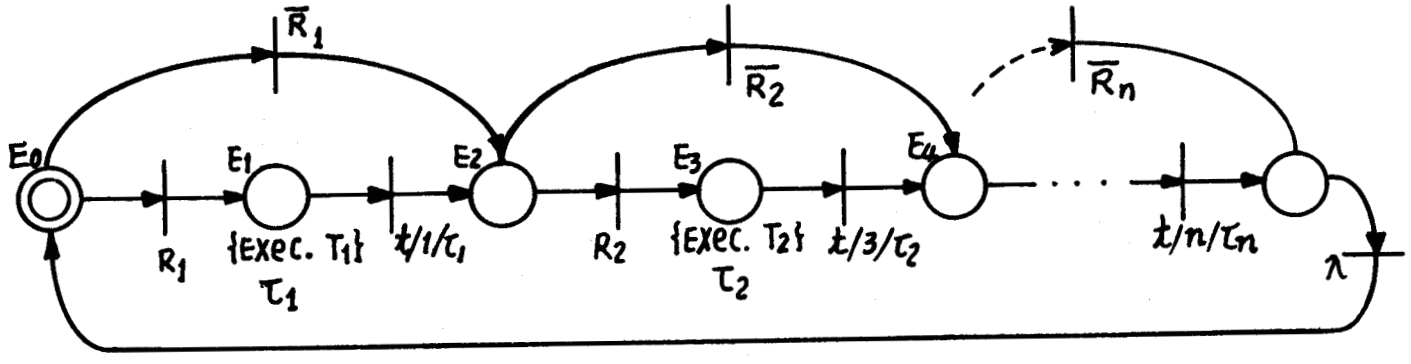


Figure II-29

II.4 - Description d'Arbitrage

II.4.1 - Le but de l'arbitrage dans notre système

Dans les problèmes de synchronisation de processus concurrents, il reste à résoudre les conflits, par exemple d'accès de plusieurs processus à une même ressource (ou ensemble de ressources). On pose alors une stratégie d'arbitrage /13/.

II.4.2 - Arbitrage à priorité fixée

L'arbitrage à priorité fixée est une discipline simple mais imparfaite, car elle peut provoquer soit la famine du processus soit celle de la ressource.

La figure II.30. montre trois processus P_1, P_2, P_3 se partagent une même ressource E_R. Nous donnons à chacun des priorités fixées. P_1 possède la priorité la plus haute, puis P_2 et enfin P_3. Donc si P_1 et P_2 sont toujours demandants, P_3 est en famine.

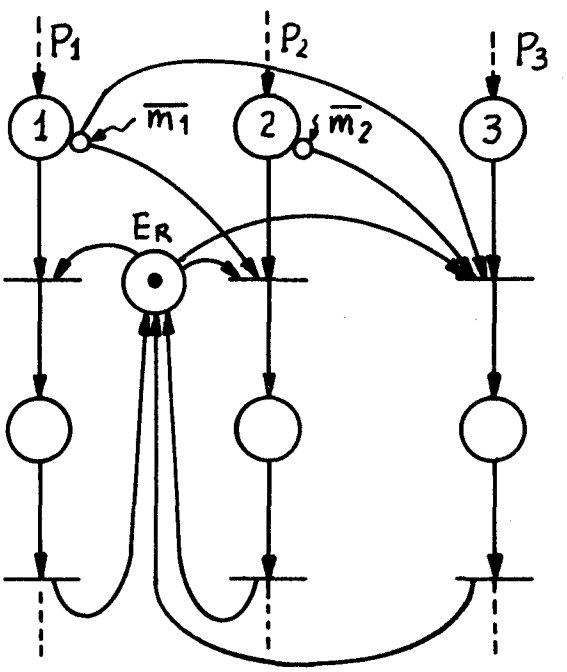


Figure II-30

II.4.3 - Arbitrage à priorité fixée avec contraintes

Pour éviter la famine on peut introduire des contraintes par exemple sur le nombre des services. La figure II.31. illustre une discipline d'arbitrage à priorité fixée avec les contraintes suivantes: le nombre de service de P_1, P_2 et P_3 sont respectivement $\varphi_1 \leq 3$, $\varphi_2 \leq 2$, $\varphi_3 \leq 1$. Les compteurs sont remis à zéro chaque fois que P_3 est servi une fois.

Nous supposons au préalable que les variables internes

m_i sont disponibles pour chaque étape.

D'autres conditions de contraintes sont également possible /14/.

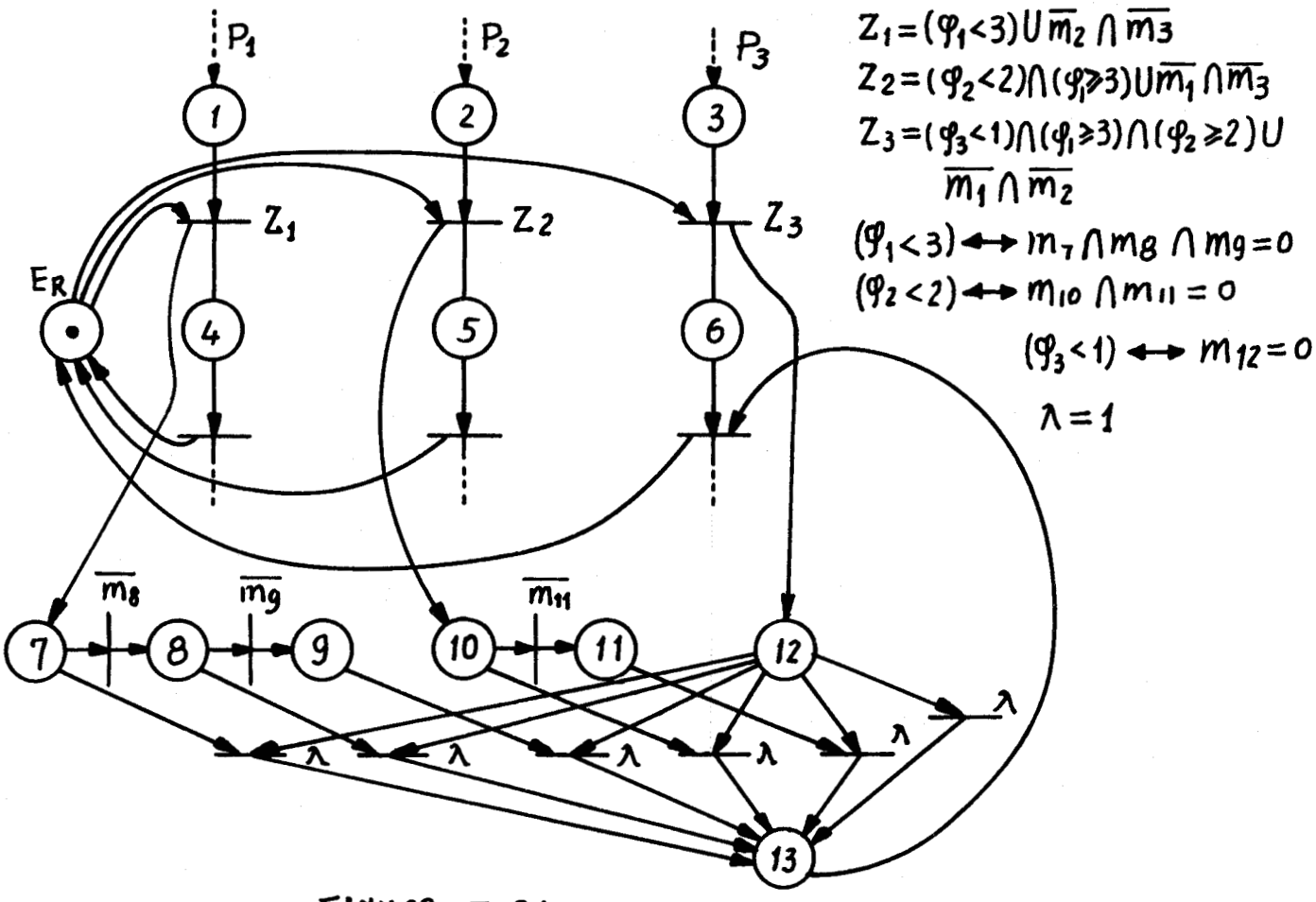


Figure II-31

II.4.4 - Arbitrage à priorité dynamique

L'arbitrage à priorité dynamique peut s'avèrer plus efficace que celui à priorité fixée. Il s'agit cette fois d'organiser des structures d'attente soit du type FIFO soit LIFO, soit à priorité relative circulaire (PRC) soit encore une combinaison avec diverses contraintes.

Nous decrivons maintenant les trois disciplines de base.

II.4.4.1. FIFO

Soit une d'attente circulaire servent d'ossature et illustrée par la figure II.32 pour trois processus.

Ceux-ci P_1,P_2,P_3 demandent une même ressource E_R pour passer dans la section critique E_s. Le predicat d'utiliser la FIFO dépend des requêtes R_1,R_2, et R_3. $Finp_1$,$FinP_2$,et $FinP_3$ sont les signaux de fin de service.

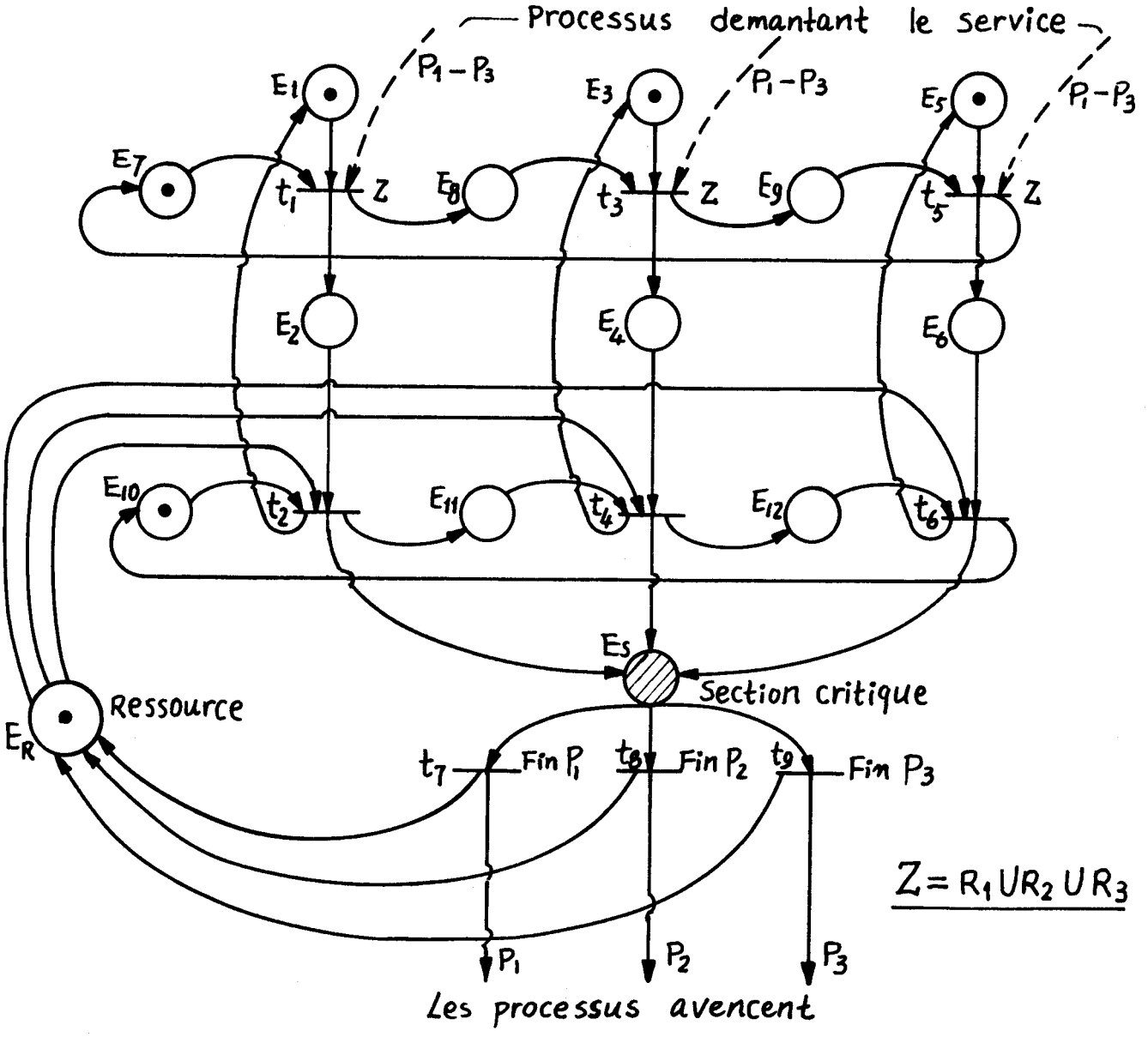


Figure II-32

II.4.4.2. LIFO

La figure II.33 illustre un système à base de trois processus entrent dans une pile et passer par une section critique E_S, en utilisant une même ressource E_R.

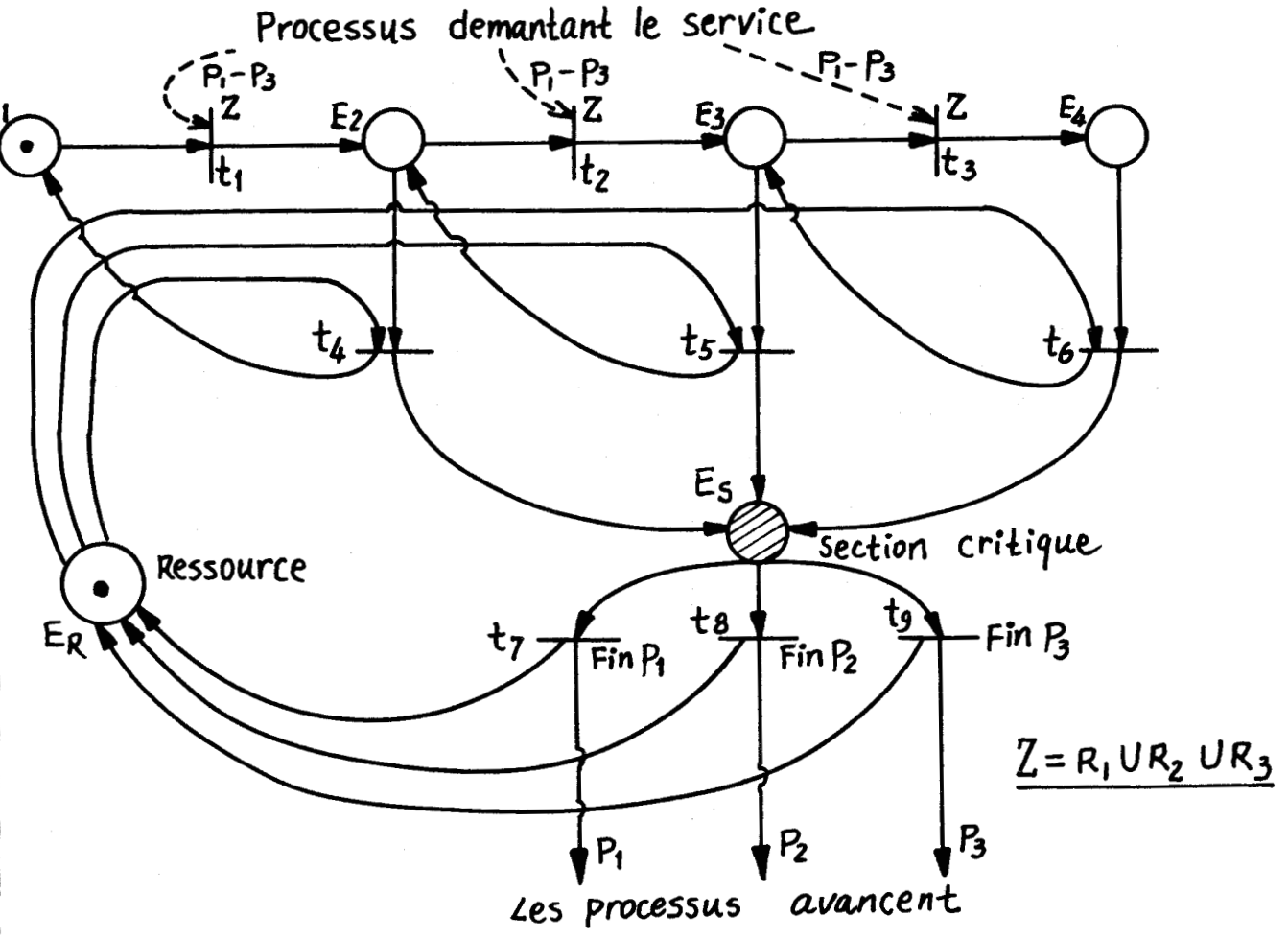


Figure II-33

II.4.4.3. PRC

Si un processus figure II.34. pose sa requête à son tour juste avant l'arivée de l'horloge H , il possède la priorité sinon il devient le dernier de la file. Personne ne possède une priorité plus haute que l'autre; la priorité est circulaire.

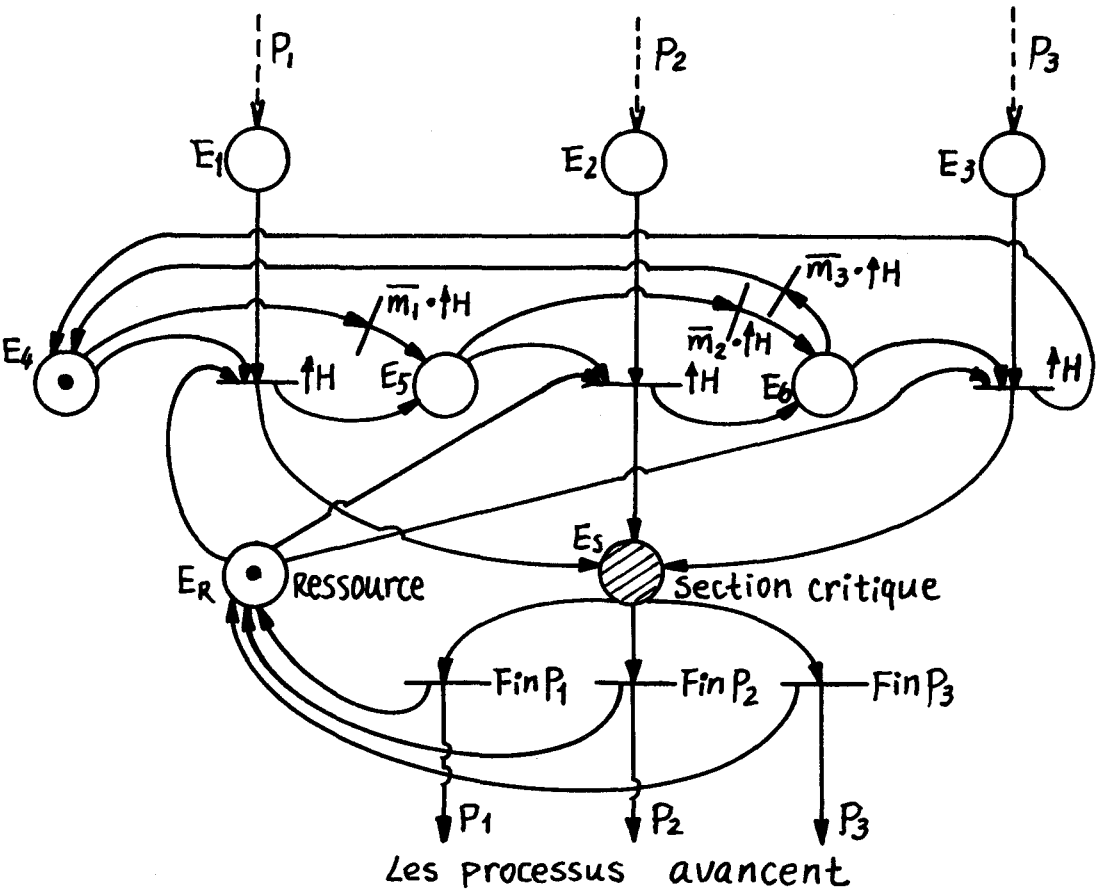


Figure II-34

II.5 - Description de Problèmes de Calcul Parllèle en Terme de GRAFCET

Nous avons envisagé jusqu'à présent le descriptions des cahier des charges en terme de processus et de ressource. Nous allons aborder maintenant la description des processus de calcul parallèle.

II.5.1 - Graphe de calcul /15/

Le graphe de calcul (ou de données) indique les connections entre opérandes et opérateurs. Nous voulons par exemple exécuter les calculs suivants (figure II.35).

OP_1 est simplement l'entrée des données X_1 et X_2 tandis que OP_{10} est assez complexe. Le graphe peut être plus ou moins décomposé au gré de l'utilisateur.

Le graphe de calcul montre également la dépendance logique entre opérations. Par exemple OP_6 a besoin que soient achevées OP_2,OP_4,OP_3.

$$
\begin{cases}
Y_2 = \mathrm{Sin}(X_1) \\
Y_3 = \mathrm{Cos}(X_2) \\
Y_4 = \mathrm{Exp}(X_1) \\
Y_5 = (Y_2)^2 \\
Y_6 = \mathrm{SQR}(Y_2) + Y_3 / Y_4 \\
Y_7 = 1 + \mathrm{Exp}(Y_4) \\
Y_8 = Y_5\, Y_6 / Y_7 \\
Y_9 = \mathrm{Sin}(Y_8) + \mathrm{Cos}(Y_8) \\
Y_{10} = Y_8 (Y_8 - 1)(Y_8 - 2) \cdots (Y_8 - 10)
\end{cases}
$$

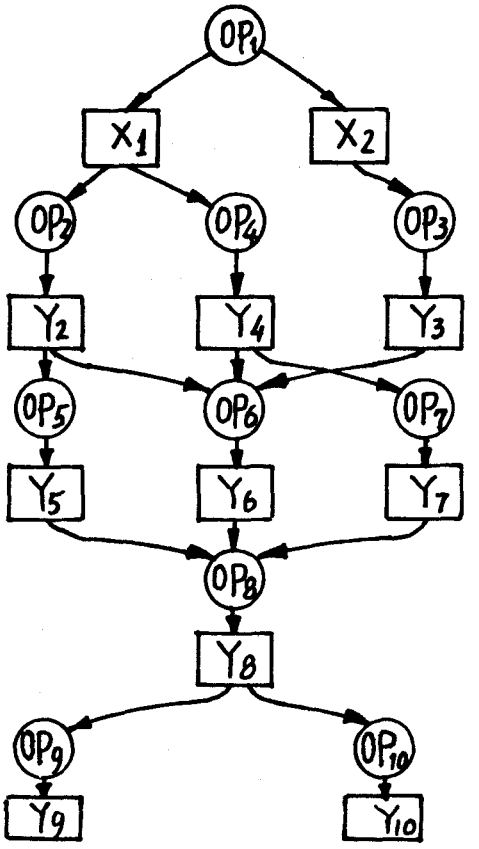


Figure II-35

II.5.2 - Graphe de commande

Le graphe de commande est en relation avec le graphe de calcul d'après les principes suivants :

--Les résultats des opérations sont associés aux transitions correspondantes. Sur la figure II.36.a) PXi et PYi sont des variables logiques, représentent la disponibilité du résultat d'opérations correspondantes.

Notons que chaque étape ne possède ici qu'une transition d'entrée et qu'une transition de sortie

--Chaque opération du graphe II.36a) placée en relation avec une étape est lancée par une action impulsionnelle mise sur la transition d'entrée de l'étape (figure II.36b). L'utilisation de cette phase dépend en fait de la méthodologie d'implantation.

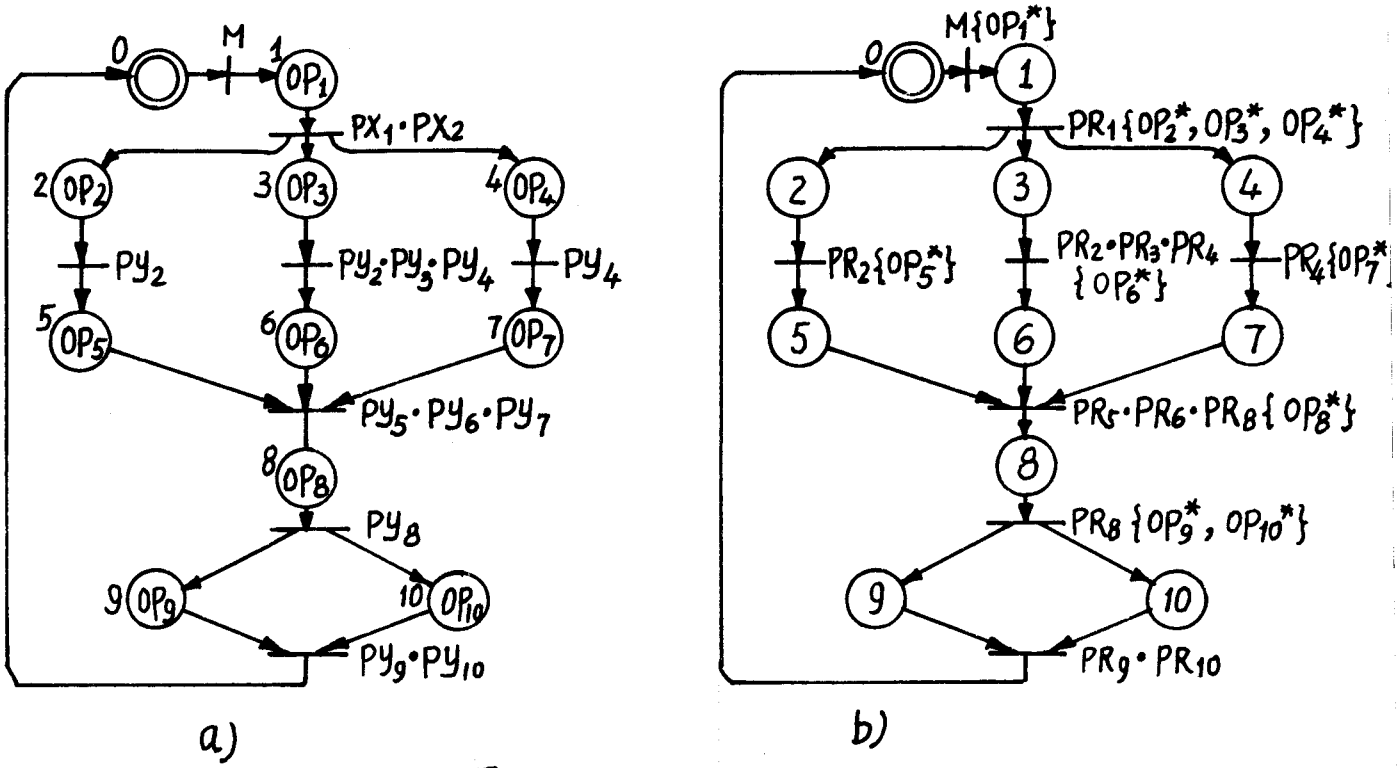


Figure II-36

II.5.3 - Graphe d'allocation de ressource

Dans notre système les processeurs sont les seules ressources. Comme le montre la figure II.37., l'opération exécutable et le processeur oisif sont liés par l'unité d'arbitrage. Les stratégies d'arbitrage ont déjà discutées.

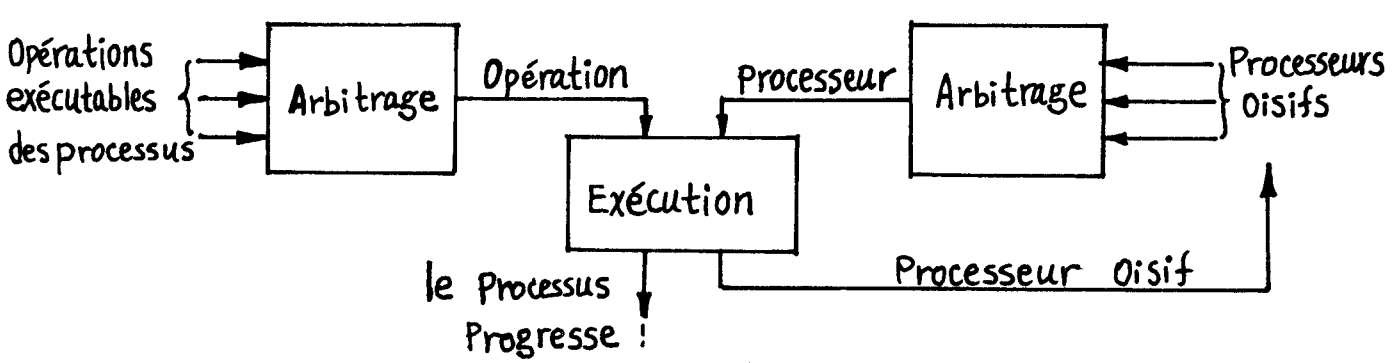
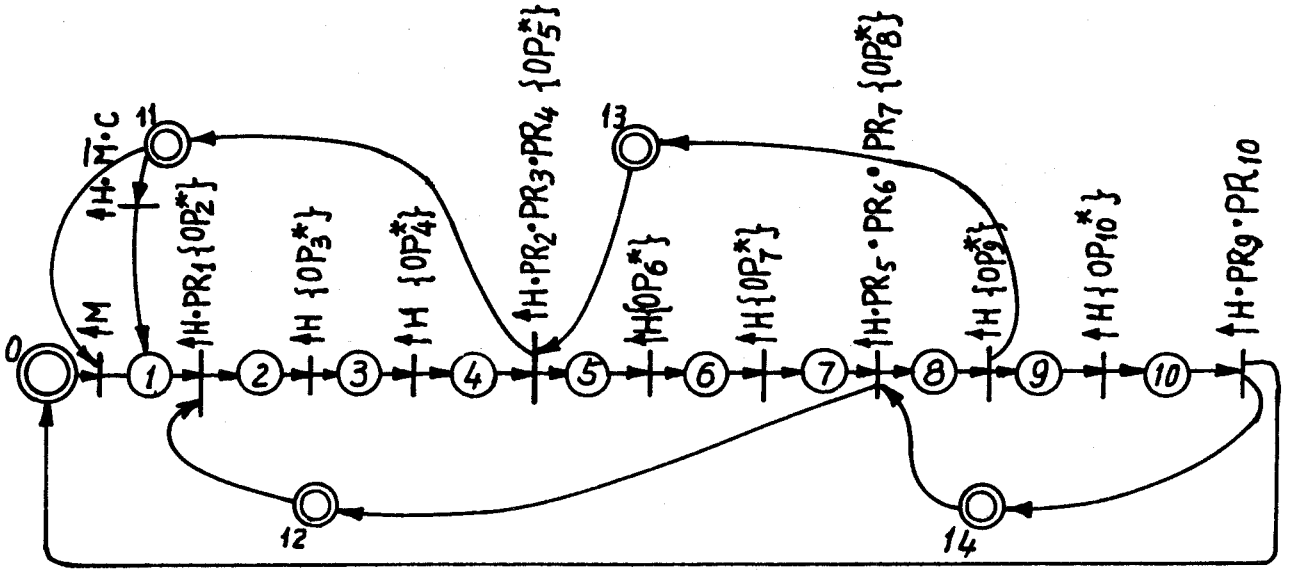


Figure II-37

II.5.4 - Description d'unité de commande en mode pipe-line

Si la tâche du paragraphe II.5.1. doit donner des résul-

tats sur un grand nombre de paires d'entrées $\{X_1, X_2\}$, nous avons intérêt à établir une configuration du type pipe-line (figure II.38.)



M — début , C — continue

H — horlog de cycle mineur du pipe line; à chaque cycle mineur, une opération est déclanchée.

Figure II-38

Lorsque les opérations OP_2, OP_3, OP_4 ont fourni leurs résultats, OP_1 peut reprendre une nouvelle valeur $\{X_1, X_2\}$. Les étapes 11 ,12,13 et 14 servent de protection.

II.5.5 - Description de l'unité de commande en mode parallèle

La figure II.39. est issue de la figure II.36. en y adjoignant le graphe d'allocation de ressource et des étapes ou les soustâches posent des requêtes et attendent l'accès d'une ressource. Nous avons associé EVi $\{At_i\}$ à chaque transition et $C_i\{AE_i\}$ à chaque étape.

EV_i est la réceptivité associée à la transition t_i

At_i est l'action associée à la transition t_i

C_i est le combinatoire lié à l'étape E_i

AE_i donne les actions associées à l'étape E_i.

Pour ne pas surcharger la figure II.39. nous faisons figurer les éléments dans les tableaux II.1. et II.2.

Nous utilisons ici deux processeurs banalisés qui sont alloués aux sous-tâches d'après une priorité relative circulaire. La priorité des soustâches sera precisée plus loin.

T_1 est un ensemble de transitions $\{t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10},$

$t_{11}\}$ utilisé pour simplifier les arcs sur la figure et aussi la rendre plus claire.

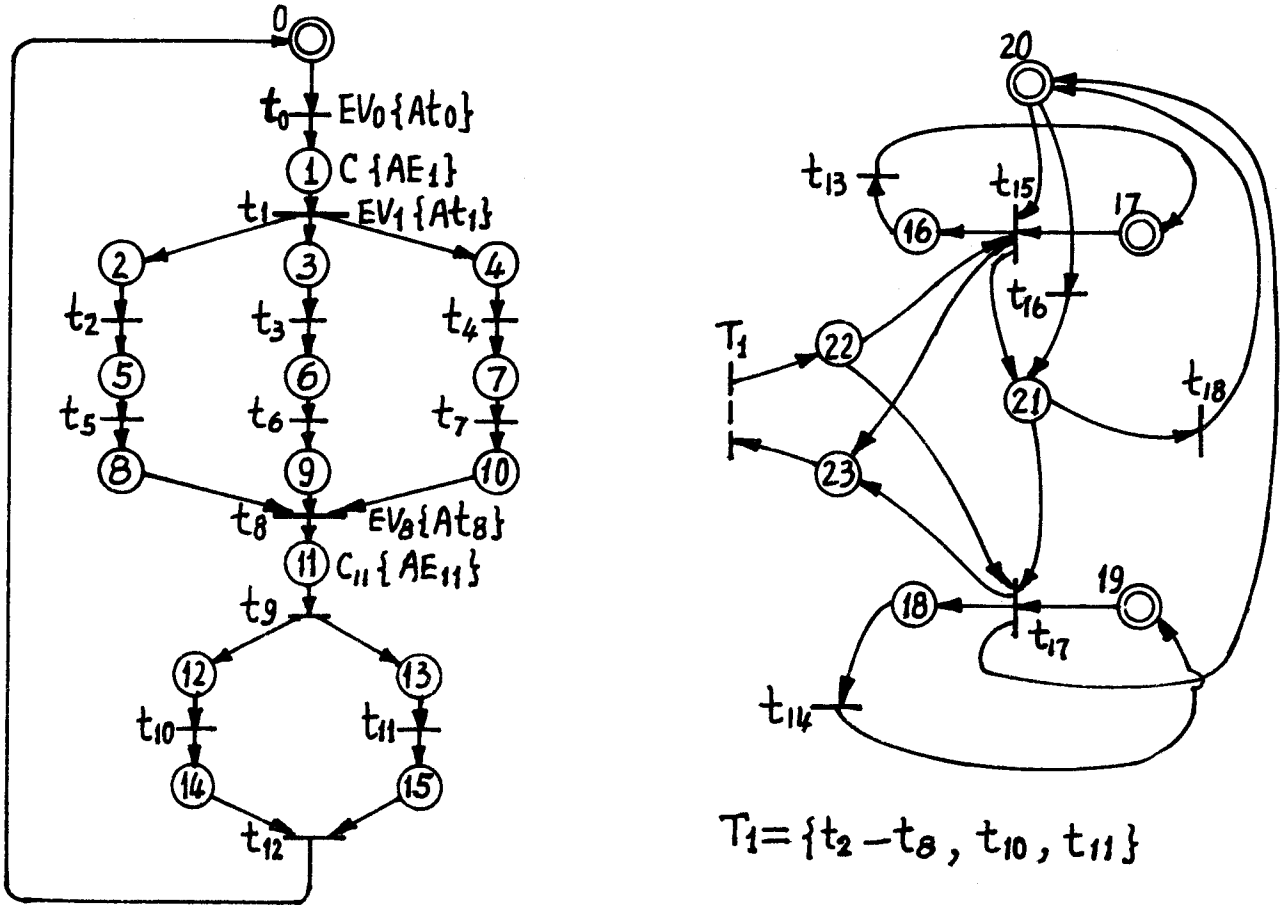


Figure II-39

Tableau II-1

Etape	Ci{AEi}	Indication de
E_0	$bt_0\{0\}$	état initial
E_1	$bt_1\{0\}$	exécution de l'entrée de X_1, X_2
E_5	$bt_2\{0\}$	exécution de OP_2
E_6	$bt_3\{0\}$	exécution de OP_3
E_7	$bt_4\{0\}$	exécution de OP_4
E_8	$bt_5\{0\}$	exécution de OP_5
E_9	$bt_6\{0\}$	exécution de OP_6
E_{10}	$bt_7\{0\}$	exécution de OP_7
E_{11}	$bt_8\{0\}$	exécution de OP_8
E_{14}	$bt_9\{0\}$	exécution de OP_9
E_{15}	$bt_{10}\{0\}$	exécution de OP_{10}
E_{16}	$bt_{11}\{0\}$	Processeur 1 est occupé
E_{18}	$bt_{12}\{0\}$	Processeur 2 est occupé

Tableau II-2

Transition t_i	EV_i { At_i }	Significations
t_0	M { OP_1^* }	M — depart du calcul.
t_1	PR_1 { $\overline{PR_1}$ }	OP_i^* — déclenchement de OP_i mais OP_1 et OP_{11} sont sans calcul.
t_2	m_{23} { $\overline{m_{23}}$, S_{22}, OP_2^* }	
t_3	m_{23} { $\overline{m_{23}}$, S_{22}, OP_3^* }	
t_4	m_{23} { $\overline{m_{23}}$, S_{22}, OP_4^* }	
t_5	$m_{23} \cdot PR_2$ { $\overline{m_{23}}$, S_{22}, OP_5^*, S_8 }	m_{23} — marguage d'etape 23, indique qu'un processeur est en train d'etre allouer.
t_6	$m_{23} \cdot PR_2 \cdot PR_3 \cdot PR_4$ { $\overline{m_{23}}$, $\overline{PR_3}$, S_{22}, OP_6^* }	
t_7	$m_{23} \cdot PR_4$ { $\overline{m_{23}}$, S_{22}, OP_7^*, S_{10} }	
t_8	$m_{23} \cdot PR_5 \cdot PR_6 \cdot PR_7$ { $\overline{m_{23}}$, $\overline{PR_2}$ $\overline{PR_4}$, $\overline{PR_5}$, $\overline{PR_6}$, $\overline{PR_7}$, r_8, r_{10}, S_{22}, OP_8^* }	{ $\overline{m_{23}}$ } — Un processeur a été alloué, l'étape E_{23} est désactivée immédiatement.
t_9	PR_8 { $\overline{PR_8}$ }	
t_{10}	m_{23} { $\overline{m_{23}}$, S_{22}, OP_9^*, S_{14} }	PR_i — fin de OP_i.
t_{11}	m_{23} { $\overline{m_{23}}$, S_{22}, OP_{10}^* }	{ $\overline{PR_i}$ } — Indique que PR_i ne joue plus son rôle, donc mise à zéro.
t_{12}	$RR_9 \cdot PR_{10}$ { $\overline{PR_9}$, $\overline{PR_{10}}$, r_{14}, OP_{11}^* }	
t_{13}	$\overline{REQ_1}$ { S_{17} }	REQ_i — le Processeur i pose sa requête.
t_{14}	$\overline{REQ_2}$ { S_{19} }	
t_{15}	$m_{17} \cdot m_{22} \cdot REQ_1$ { RES_1, r_{17}, r_{20}, r_{22}, S_{21}, S_{23} }	RES_i — Avertissement que le processeur i est alloué.
t_{16}	$\overline{REQ_1}$ { r_{20}, S_{21} }	
t_{17}	$m_{19} \cdot m_{22} \cdot REQ_2$ { RES_2, r_{19}, r_{21}, r_{22}, S_{20}, S_{23} }	r_i, S_i, m_i — Variables internes
t_{18}	$\overline{REQ_2}$ { r_{21}, S_{20} }	

II.5.6 - Description de l'unité de commande en mode mixte

La combinaison des figure II.38 et II.39 pourrait constituer un système plus efficace que chaque configuration prise isolement.

La figure II.40 illustre une structure permettant la même tâche que précédemment mais en traitant divers cas $\{X_1, X_2\}$ en parallèle et en pipe-line, suivant l'utilisation des quatre processeurs.

Les étapes $E_{26}, E_{27}, E_{28}, E_{29}, E_{30}$ contrôlent le pipe-line. La stratégie d'allocation de ressources et de sous-tâches est à priorité fixée.

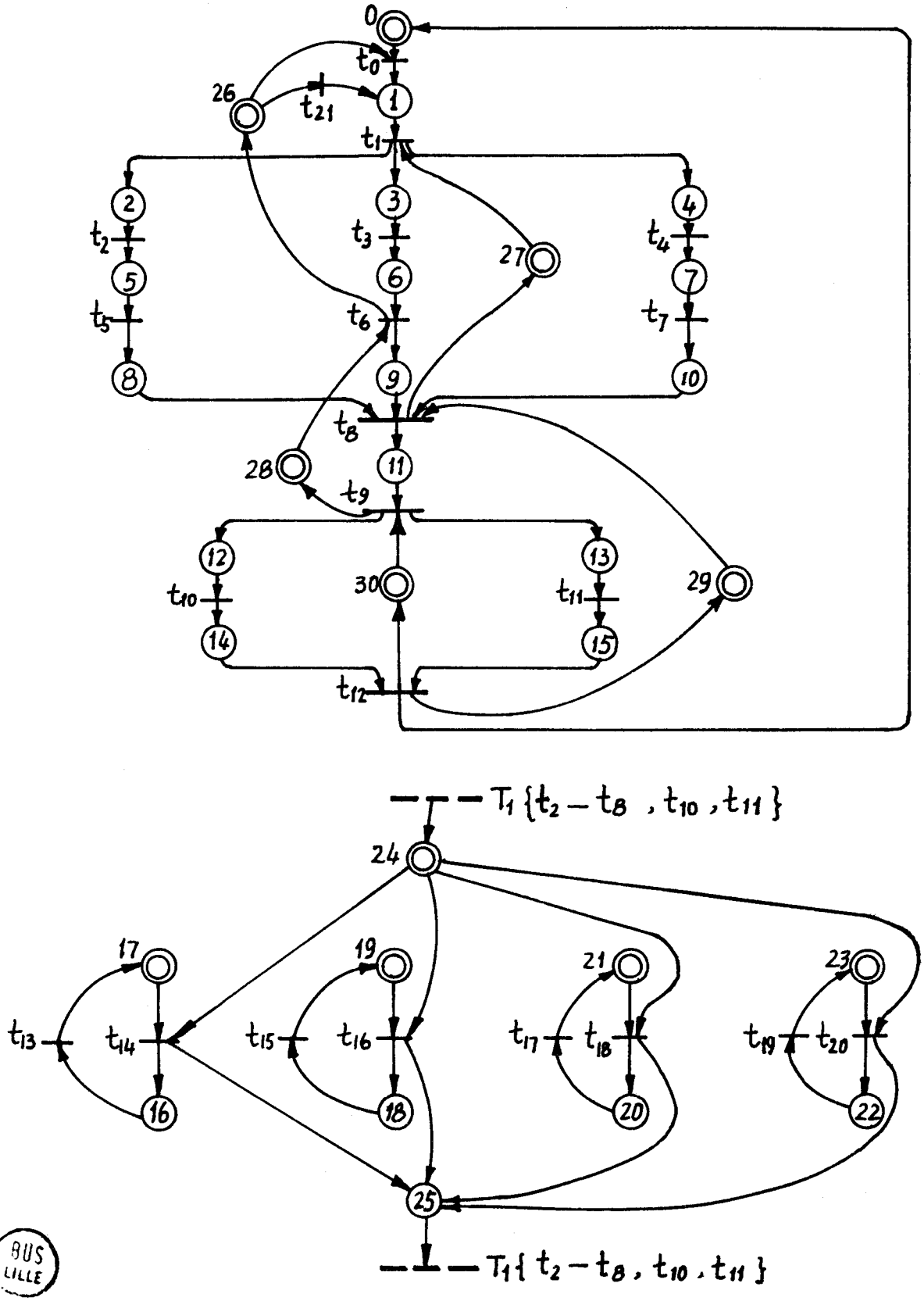


Figure II-40

Les table II.3 et II.4 donnent les $EV_i\{At_i\}$ et $C_i\{AE_i\}$.

Tableau II-3

Etape E_i	$C_i\{AE_i\}$	significations
E_0 — E_{15}	même chose que tableau II-1	

Tableau II-4

Transition t_i	$EV_i\{At_i\}$	significations
t_0	$M \cdot m_0 \{\overline{M}, r_0, r_{26}, OP_1^*\}$	M — depat du calcul
t_1	$m_{27} \cdot PR_1 \{\overline{PR_1}, r_{27}\}$	m_{25} — Un processeur est à allouer ($m_{25}=1$).
$t_2, t_3, t_4, t_5, t_7, t_{10}, t_{11}$ sont pareils au tableau II-2, mais : $\overline{m_{23}} \leftarrow \overline{m_{25}}$, $S_{22} \leftarrow S_{24}$, $m_{23} \leftarrow m_{25}$		$\{m_{25}\}$ — Un processeur a été alloué, l'étape E_{25} est désactivée immédiatement.
t_6	$m_{25} \cdot m_{28} \cdot PR_2 \cdot \overline{PR_3} \cdot PR_4 \{\overline{m_{25}}, \overline{PR_2}, \overline{PR_3}, \overline{PR_4}, r_{28}, S_{24}, S_{26}, OP_6^*\}$	
t_8	$m_{25} \cdot m_{29} \cdot PR_5 \cdot PR_6 \cdot PR_7 \{\overline{m_{25}}, \overline{PR_5}, \overline{PR_6}, \overline{PR_7}, r_8, r_{10}, r_{29}, S_{24}, S_{27}, OP_8^*\}$	
t_9	$m_{30} \cdot PR_8 \{\overline{PR_8}, r_{30}, S_{28}\}$	REQi — Le proccesseur i pose sa requête.
t_{12}	$PR_9 \cdot PR_{10} \{\overline{PR_9}, \overline{PR_{10}}, r_{14}, S_{29}, S_{30}, OP_{11}^*\}$	
t_{13}	$\overline{REQ_1}\{0\}$	RESi — Avertissement que le processeur i est alloué.
t_{14}	$m_{24} \cdot REQ_1 \{\overline{m_{24}}, S_{25}, RES_1\}$	
t_{15}	$\overline{REQ_2}\{0\}$	C — Pipe line en fonctionnement (dans la simulation du chapitre IV, supposons $C \equiv 1$)
t_{16}	$m_{24} \cdot REQ_2 \{\overline{m_{24}}, S_{25}, RES_2\}$	
t_{17}	$\overline{REQ_3}\{0\}$	
t_{18}	$m_{24} \cdot REQ_3 \{\overline{m_{24}}, S_{25}, RES_3\}$	
t_{19}	$\overline{REQ_4}\{0\}$	
t_{20}	$m_{24} \cdot REQ_4 \{\overline{m_{24}}, S_{25}, RES_4\}$	
t_{21}	$C \cdot \overline{M} \{r_{26}, OP_1^*\}$	

II.2.6 - Conclusion

Dans ce chapitre nous avons étudié le concept de GRAFCET et nous l'avons utilisé à la description de systèmes de traitement numérique aussi bien au niveau matériel que logiciel. Nous avons surtout envisagé les problèmes de synchronisation et d'arbitrage. Enfin nous avons proposé trois configurations pour réaliser une unité de commande soit le mode pipe-line, le mode parallèle et le mode mixte.

CHAPITRE III : IMPLANTATION ET REALISATION

III.1 - Implantation du grafcet

Deux nombreuses méthodes d'implantation de grafcet existent tant en logique cablée que programmée. |1| |2|.

Dans les réalisations cablées, les étapes, transitions et arcs sont respectivement matérialisés par des mémoires, des circuits combinatoires et des cables. A titre d'exemple, nous présentons l'approche modulaire, puis nous détaillerons les implantations programmées utilisées dans ce mémoire.

III.1.1. - Approche modulaire

La figure III.1 présente un module "jonction" et sa réalisation

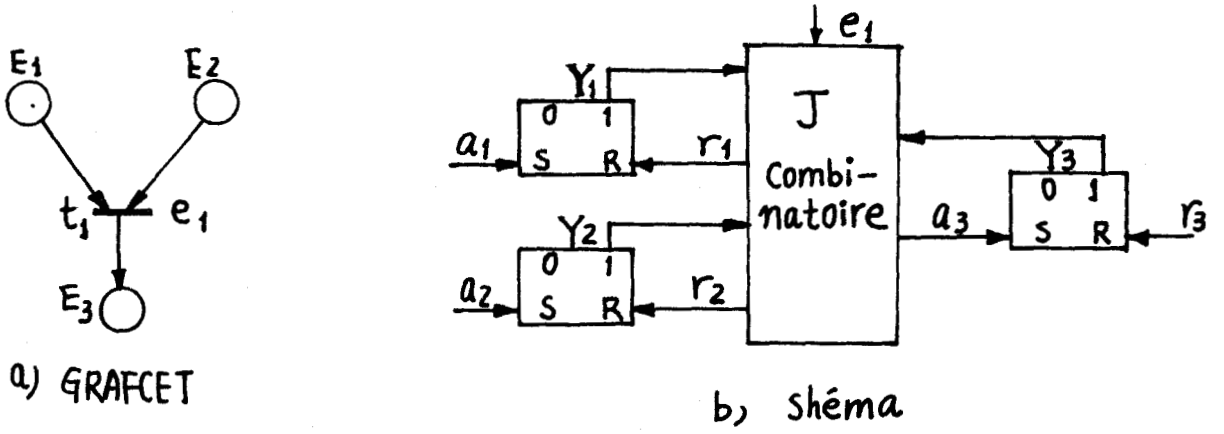


Figure III-1 Module "Jonction"

On peut trouver d'autres modules : Distribution, Sélection, Attribution, Transfert |2|. Cette méthode possède une grande vitesse d'exécution mais d'une part elle est moins simple qu'en programmé et ne s'applique pas à tous les types de grafcet car elle est asynchrone.

La mémoire Y_3 devra être mise à un lorsque les mémoires Y_1 et Y_2 étant actives, la réceptivité (entrée e_1) devient vraie. La mise à un de Y_3 provoque alors la mise à 0 de Y_1 et Y_2. Autrement dit :

$a_3 = e_1 . Y_1 . Y_2$ condition d'appel

$r_1 = r_2 = e_1 Y_3$ condition de reprise.

III.1.2. - Approche programmée

Parmi toutes les méthodes, les plus intéressantes utilisent un programme moniteur invariant et des données représentatives du grafcet. Par simple modification de quelques informations dans la table de données, on peut corriger un grafcet. Ceci donne une grande souplesse à ces méthodes qui sont toutefois plus lente. Il reste malgré tout possible de découper le grafcet en plusieurs sous-graphes exécutées parallèlement |2| .|16|.

Compte tenu de nos possibilités matérielles, nous choisissons un traitement monoprocesseur d'un grafcet général en disposant d'instruction de saut général.

III.2 - Une méthode d'implantation programmée

III.2.1. - Synchronisme lié aux entrées et sorties

Dans cette implantation, les valeurs des entrées sont figées avant de commencer le traitement. Par ailleurs les nouvelles valeurs des commandes ne sont affectées aux lignes de sortie que globalement à la fin de tous les traitements.

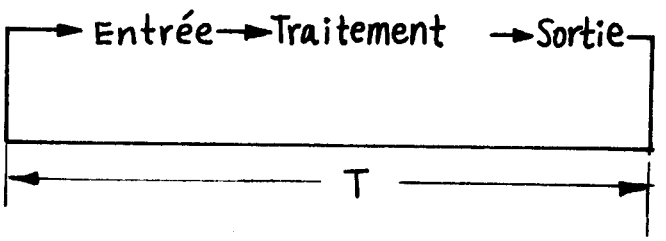


Figure III-2 Un cycle de traitement

La figure III.2 montre le cycle de traitement avec entrées et sorties synchrone.

III.2.2. - Table de données du grafcet

Notre système étant orienté vers des traitements très variés, le grafcet à implanter sera amené à être changé. C'est pourquoi nous utilisons une méthode orientée "table de données" avec un programme de gestion invariant.

III.2.2.1. - Table de données autour des étapes

On peut décrire le grafcet dans la table de données en prenant comme base soit les étapes, soit les transitions. La description autour des étapes semble plus simple pour notre système qui possède des transitions en conflit, des étapes d'arbitrage et des combinatoires locaux. Ce cas consiste à établir une liste d'étapes comprenant successivement le traitement des actions associées, les conditions de franchissement de ses transitions de sortie, les activations et désactivations lors d'un franchissement.

La présence de transitions du type "jonction" entraîne une redondance de description et de traitement. Pour pallier cet inconvénient, on utilise la notion d'étape clé et d'étape de synchronisme. Comme base de la création de la table de façon à ce que chaque transition ne soit décrit qu'une seule fois.

Une étape d'entrée unique d'une transition est obligatoirement choisie comme étape clé. Cette première phase peut entraîner la définition d'étapes de synchronisme. Plusieurs étapes clés, ainsi définies peuvent posséder une même transition de sortie. Une seule étape parmi elles est choisie comme étape clé, les autres sont alors étapes clés de synchronisme.

Une étape clé E_K est traitée de façon ordinaire :

$$\forall t_i \in E_K^{\bullet} \cap EV_i = 1,\ M(E_K) = 0$$

$$\forall t_j \in {}^{\bullet}E_K \cap EV_j = 1,\ M_+(E_K) = 1$$

$$M_+(E_K) = M(E_K) \cup M_+(E_K).$$

Une étape clé de synchronisme suit la règle suivante :

$$\forall t_i \in E_K^{\bullet} \cap EV_i = 1,\ r_k = 1$$
$$\forall t_j \in {}^{\bullet}E_K \cap EV_j = 1,\ s_k = 1$$
$$m_{k+} = m_k \cap \overline{r}_k \cup s_k,\ M_+(E_k) = m_{k+}$$

r_k, s_k, m_k sont des variables internes de l'étape E_k. $M(E_k)$ est l'activité précédente, $M_+(E_k)$ est la nouvelle activité.

L'étape de synchronisme, représentant une ressource non réentrante obéit à d'autre règle spécialisée pour éviter la redondance d'allocation d'une même ressource :

$$\forall t_i \in E_k^{\bullet} \cap EV_i = 1,\ m_k = 0$$
$$\forall t_j \in {}^{\bullet}E_k \cap EV_j = 1,\ s_k = 1$$
$$m_k = m_k \cup s_k$$

Pour illustrer la procédure de choix, la figure III.3 donne un exemple. L'étape 1 est choisie comme étape clé, l'étape 2 est clé de synchronisme, l'étape 3 étape de synchronisme.

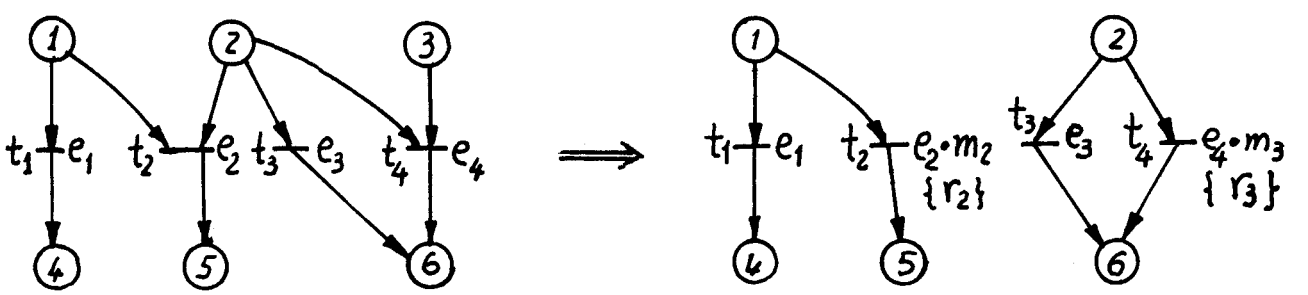
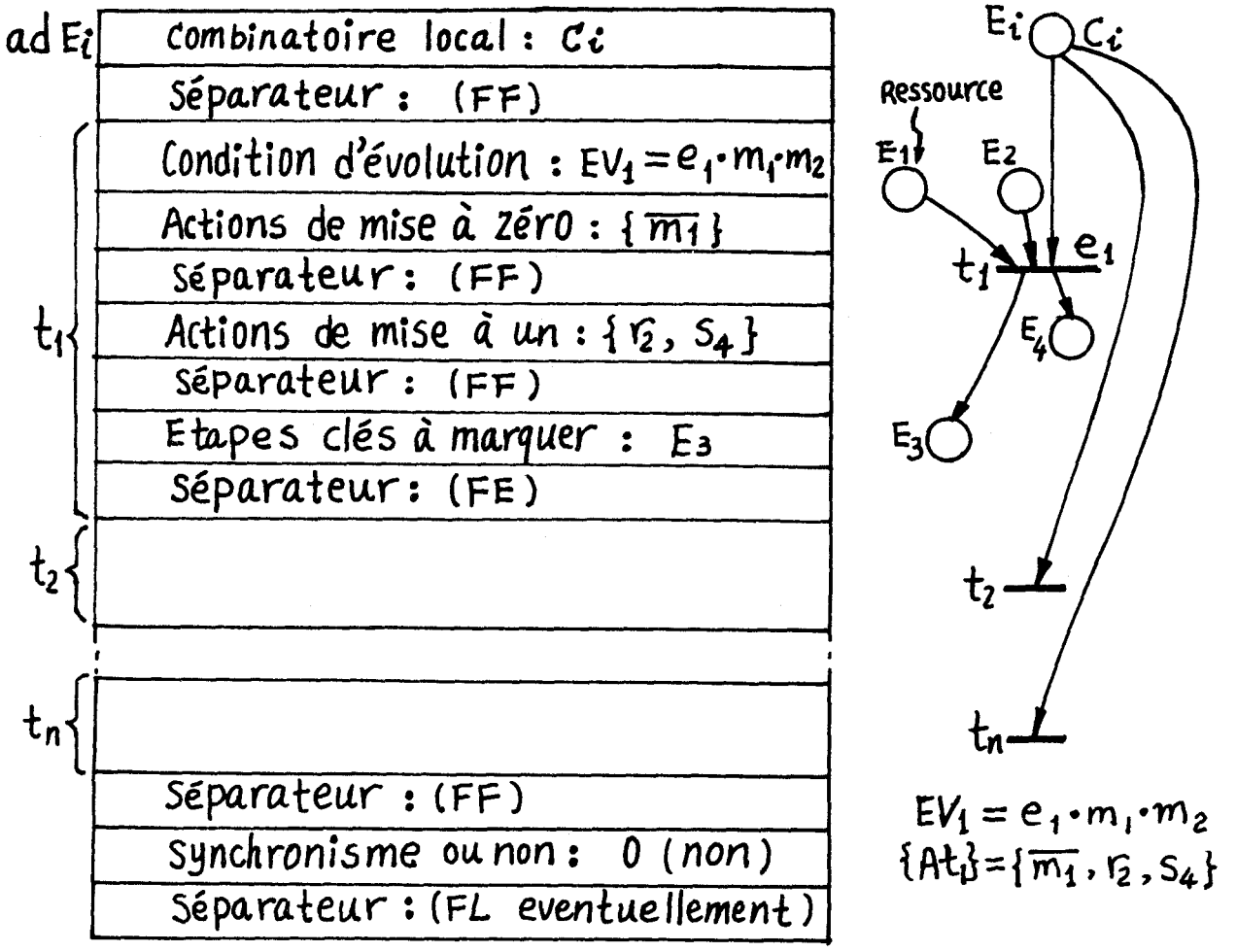


Figure III-3

La figure III.4 donne la structure générale de la liste d'une étape E_i qui possède n transitions de sortie.



Figre III-4 liste d'une étape E_i

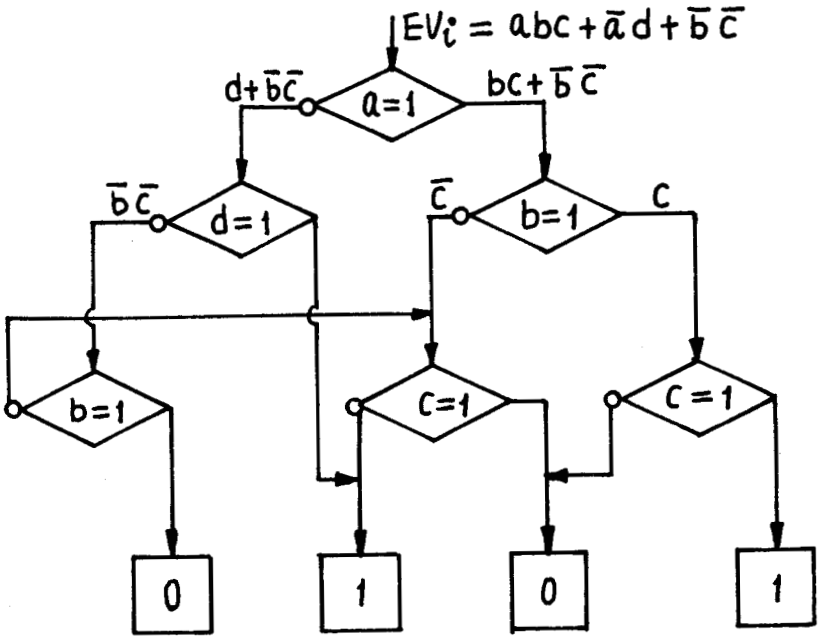
La transition t_1 comporte deux étapes de synchronisme $\{E_1, E_2\}$ et deux étapes de sortie $\{E_3$ et $E_4\}$ parmi lesquelles E_3 est une étape clé.

Nous convenons d'appeler LRDP la liste de toutes les étapes clés d'un grafcet. La table de données comprend également d'autres listes que nous présentons maintenant.

III.2.2.2. - Les autres listes

LISTE D'ACTIVITE

Deux listes d'activité sont dynamiquement établies : l'une s'appelle la liste des étapes activées en cours (LEAC) qui sert pour le cycle en cours de traitement, l'autre (LEAS) est la liste des étapes actives pour le cycle suivant. A chaque cycle, le rôle des listes est interverti. L'étape est repérée directement par son adresse



a) l'arbre de décision logique

Adresse	Contenu
a_1	Test a
	Saut à d_1
b_1	Test b
	Saut à C_2
C_1	Test C
	Mise à 0
	Mise à 1
d_1	Test d
	Saut à b_2
	Mise à 1
C_2	Test C
	Mise à 1
	Mise à 0
b_2	Test b
	Saut à C_2
	Mise à 0

b) liste

Figure III-5

Adresse	Contenu
1	a
2	8
3	b
4	11
5	C
6	Séparateur FF
7	Séparateur FE
8	d
9	14
10	Séparateur FE
11	C
12	Séparateur FE
13	Séparateur FF
14	b
15	11
16	Séparateur FF

Figure III-6

dans les listes. Notons par ailleurs que la façon d'inscrire les étapes dans LEAS peut constituer une discipline d'arbitrage.

LISTE DES CONDITIONS D'EVOLUTION

La réceptivité de chaque transition est traitée comme un sous-programme de façon à simplifier l'écriture de la liste principale du grafcet (LRDP) et à modifier facilement les réceptivités. Dans LRDP un pointeur donne l'adresse d'entrée de ce sous-programme écrit dans une autre liste LCE.

Les réceptivités sont supposées **mises** sous la forme d'une somme de produit. La figure III.5 donne un exemple de mise en place pour la réceptivité $EV_i = a\ b\ c + \overline{a}\ d + \overline{b}\ \overline{c}$ avec l'arbre de décision logique et la liste correspondante. Celle-ci signifie implicitement que si un test est faux, l'opération suivante est exécutée et si le test est vrai, l'opération suivante est sautée. Les opérations étant faites par le système de gestion, la liste réelle devient celle de la figure III.6.

Pour implanter toutes les données relatives à un grafcet, nous établissons encore les listes suivantes :

LC liste des constantes

LI liste de l'activité initiale

LE liste des variables d'entrée

LM liste des masques d'entrée

LS liste de sortie

LVI liste des variables internes

LCG liste du combinatoire générale

Nous pouvons les expliciter sur l'exemple de la figure III.7

III.2.2.3. - Exemple

Pour illustrer l'implantation, nous supposons une mémoire avec trois pages de 256 mots chacune.

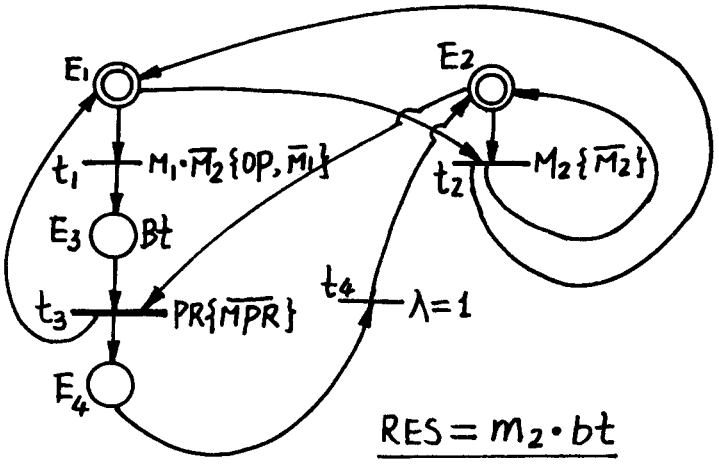


Figure III-7

La figure III.8 donne toutes les listes

Adresse	Contenu	Code
Page 0 :		
	LC	
0	Vide	0
1	Nº de page de LRDP	2
2	Nº de page de LCE	1
3	Début de LE	48
4	Début de LS	96
5	Début de LVI	144
6	Début de LI	20
7	Pointeur B de LEAC	2
8	Pointeur H de LEAC	1
9	Pointeur B de LEAS	17
10	Pointeur H de LEAS	1
11	Début de LLG	32
12	Nº de page de LLG	1
13	Nombre de V.E.	3
14	Indicateur de franchissement	INF
15	Début de LM	72
16	Séparateur FF	255
17	Séparateur FF	255
	LI	
20	ad E_1	1
21	FF	255
22	ad m_1	146
23	ad m_2	150
24	FF	255
25	FF	255
	LE	
48	M_1	0
49	M_2	0
50	PR	0
51	FF	255
52	FF	255

Adresse	Contenu	code
Page 0 :		
	LM	
72	MM_1	1
73	MM_2	1
74	MPR	1
75	FF	255
76	FF	255
	LS	
96	RES	0
97	OP	0
98	Bt	0
99	FF	255
100	FF	255
	LVI	
144	r_1	0
145	S_1	0
146	m_1	0
147	adE_1	1
148	r_2	0
149	S_2	0
150	m_2	0
151	FF	255
152	FF	255

* Pour montrer la manière de traitement, l'étape clé E_1 est choisie comme une étape clé de synchronisme.

Figure III-8 a)

Adresse	contenu		code
Page 1 :			
	LCG		
32	ad SP_5		63
33	ad RES		96
34	FF		255
35	FF		255
	LCE		
48	ad M_1		48
49	FF		255
50	ad M_2	SP_1	49
51	FE		254
52	FF		255
53	ad M_2		48
54	FF	SP_2	255
55	FE		254
56	ad PR		50
57	FF		255
58	ad m_2	SP_3	150
59	FF		255
60	FE		254
61	FE	SP_4	254
62	Vide		0
63	ad m_2		150
64	FF		255
65	ad Bt	SP_5	98
66	FF		255
67	FE		254

	Adresse	Contenu	Code
	Page 2 :		
ad E_1 →	1	FF	255
	2	ad SP1	48
	3	ad MM1	72
	4	FF	255
	5	ad r_1	144
t_1	6	ad OP	97
	7	FF	255
	8	ad E_3	21
	9	FE	254
	10	ad SP2	53
	11	ad MM2	73
	12	FF	255
	13	ad r_1	144
t_2	14	ad r_2	148
	15	ad S_1	145
	16	ad S_2	149
	17	FF	255
	18	FE	254
	19	FF	255
	20	Synchronisme	1
ad E_3 →	21	ad Bt	98
	22	FF	255
	23	ad SP3	56
	24	ad MPR	74
	25	FF	255
t_3	26	ad r_2	148
	27	ad S_1	145
	28	FF	255
	29	ad E_4	33
	30	FE	254
	31	FF	255
	32	non Synch.	0
ad E_4 →	33	FF	255
	34	ad SP4	61
	35	FF	255
t_4	36	ad S_2	149
	37	FF	255
	38	FE	254
	39	FF	255
	40	non Synch.	0

* ad X indique l'adresse de X,
X indique la valeur Booléene de X.

Figure III-8 b)

Cette liste peut paraître rébarbative et ne peut être établie qu'à partir d'un langage de haut niveau, proche de la représentation graphique initiale.

III.2.3. - Gestion de la table de données du grafcet

La méthodologie choisie est asynchrone relativement au franchissement des transitions. Ceci implique que nous nous restreignons à une classe particulière de grafcet,dans lesquels il ne peut y avoir franchissement simultané de transitions. Il reste toutefois possible d'utiliser la notion de structure implicite.

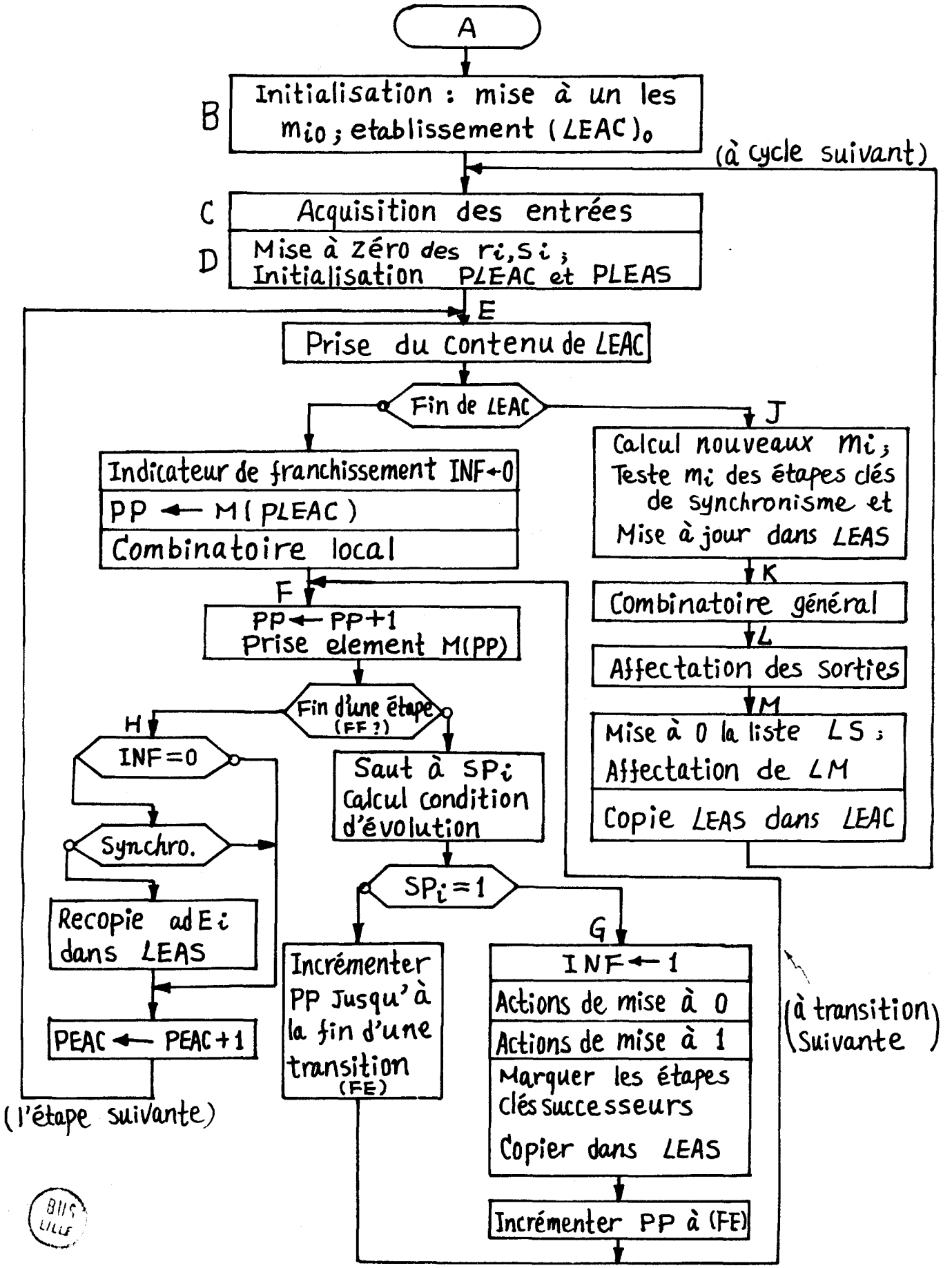
Ce choix se justifie d'une part par le fait que les procédures synchrones sont plus lentes et plus complexes et d'autre part parce qu'il est nécessaire d'introduire dans un certain nombre de cas une relation d'ordre sur le franchissement des transitions.

Nous avons ainsi réalisé une extension de cette classe de grafcet en donnant un ordre au traitement des transitions, de façon à tenir compte des règles de priorité précédemment introduites. Les réceptivités de certaines transitions sont donc parfois égales mais les franchissements sont rendus disjoints par l'introduction de ces règles. Un traitement asynchrone rend compte tout naturellement de ces règles dans le cas où l'ensemble des transitions est ordonné.

Le programme de gestion correspondant est donné figure III.9, pour laquelle nous précisons les points suivants :

III.2.3.1. - Initialisation

Le nombre d'étapes initialement actives peut être quelconque. Initialement, les adresses des étapes clés actives sont inscrites dans LEAC. Les variables internes m_i des étapes de synchronisme, activées initialement, sont mises à 1 dans LVI. La liste des activités initiales (LI) est divisée en deux parties séparées par FF une pour les étapes clés, l'autre pour les variables internes.



PLEAC — Pointeur de la liste LEAC
PLEAS — Pointeur de la liste LEAS
PP — Pointeur du programme

Figure III–9

III.2.3.2. - Acquisition des entrées

Au début d'un cycle, les variables d'entrée sont acquises dans LE et figées pour toute la durée du cycle en cours de façon à réaliser le traitement synchrone vis à vis des entrées.

La disponibilité des variables d'entrée peut parfois être nécessaire pour quelques cycles dans le cas par exemple d'un traitement en mode pipe-line. Nous établissons donc la liste des masques d'entrée (LM) qui enregistre la fin de disponibilité des entrées. (Par exemple, le franchissement de la transition t_3 de la figure III.7 supprime la disponibilité du prédicat PR, représenté par $\{\overline{MPR}\}$, c'est-à-dire, la mise à zéro de MPR dans la liste LM de la figure III.8a). A la fin d'un cycle, nous faisons la mise à jour correspondante de LM.

III.2.3.3. - Actions de mise à zéro

Dans la figure III.9 au point G, nous ajoutons, lors du franchissement d'une transition, des actions de mise à zéro de façon à éviter des conflits sur certaines ressources qui ne peuvent être réutilisées immédiatement. Ceci se fait en mettant directement m_j à zéro.

D'autres variables m_i sont calculées à la fin du cycle, selon la règle $m_i = m_i \overline{r_i} + s_i$ pour traiter le cas des étapes clés de synchronisme.

III.2.3.4. - Activation ou désactivation d'une étape

Une étape clé E_i sera active au cycle suivant si une de ses transitions d'entrée est franchie au cycle en cours (d'où figure III.9 point G on trouve : " marquer les étapes clés successeurs, copier dans LEAS") ou si elle est active et si aucune de ses transitions de sortie n'est franchie (figure III.9 H, si "INF = 0 et si l'étape n'est pas de synchronisme, on recopie AdE_i dans LEAS). Une étape clé E_i sera désactivée au cycle suivant si une de ses transitions de sortie est franchie et si aucune de ses transitions d'entrée n'est franchie.

L'activation et la désactivation d'une étape clé de synchronisme ne diffèrent que par la mise à jour dans LEAS qui est traitée en fin de cycle (figure III.9 J).

III.3 - Réalisation de l'unité de commande

L'unité de commande a pour rôle de traiter cycle par cycle la table de données du grafcet selon la stratégie de la figure III.9. Nous considérons en fait l'unité de commande comme un processeur spécialisé. La table de données du grafcet est stockée dans une mémoire vive afin de permettre une adaptation au changement de GRAFCET. Le programme de gestion est exécuté par le processeur sous le contrôle d'unité de commande, réalisée soit en circuits logiques cablés soit en technique microprogrammée.

III.3.1. - L'unité de commande cablée

Le programme de gestion peut être décrit en utilisant des instructions standard |2| et des circuits logiques sont cablés pour matérialiser ces instructions. Avec un jeu d'instructions bien choisi, il n'y a pas de problème majeur sauf celui de la vitesse d'exécution, car le passage par un cadre prédéfini risque d'alourdir l'écriture de certains programmes et par ailleurs, un changement de stratégie de gestion est impossible à ceux du cablage.

III.3.2. - L'unité de commande microprogrammée

L'unité de commande devient maintenant un processeur spécialisé permettant l'exécution d'instruction spécialisée, écrite sous la forme de microprogramme d'après la stratégie de la figure III.9.

Un changement de stratégie devient un simple changement de microprogramme. L'adjonction d'instructions revient à ajouter des micro-programmes. Nous réservons une page 1 de la mémoire de contrôle pour ces adjonctions, la page 0 étant destinée au microprogramme spécialisé.

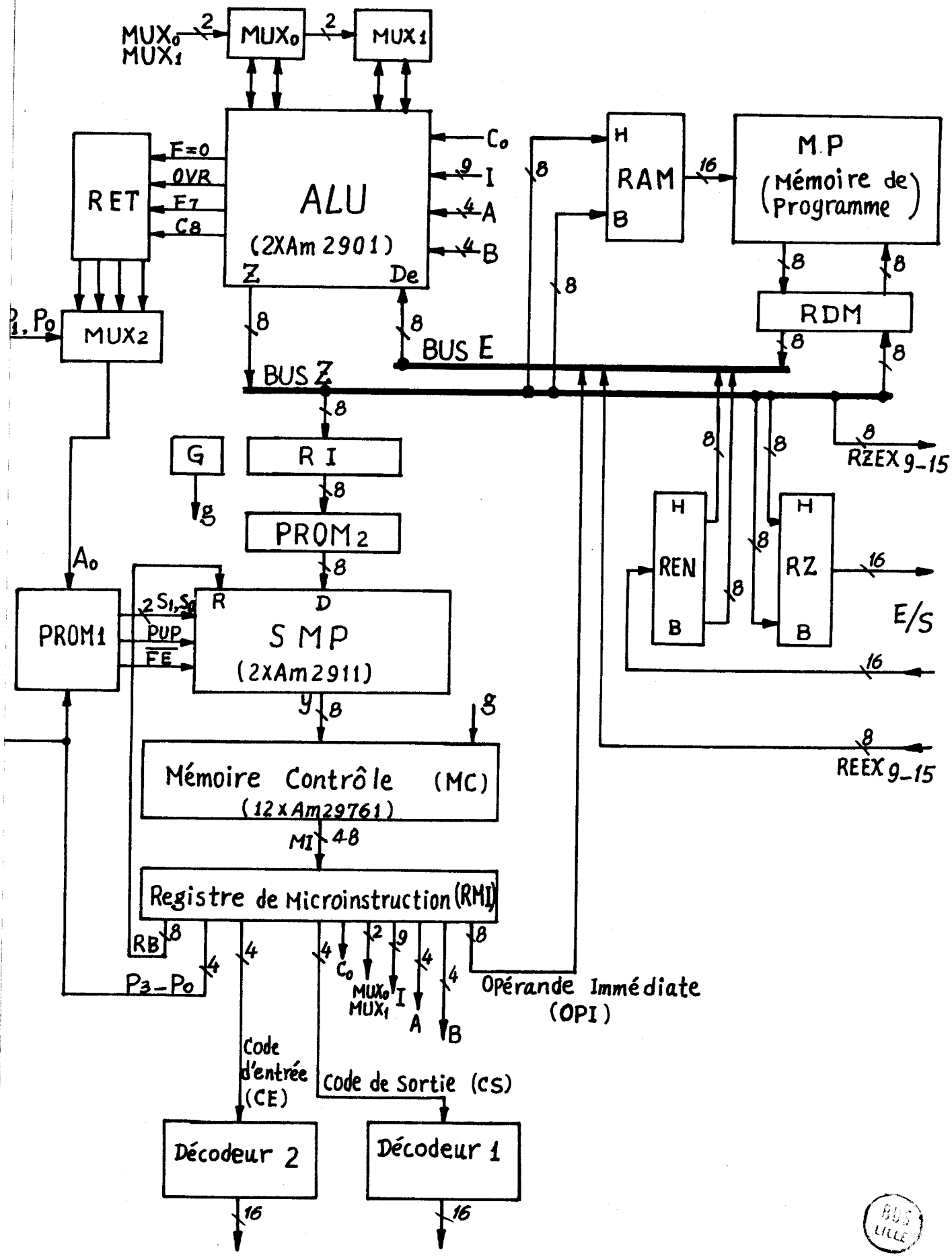


Figure III-10

III.3.2.1. - Architecture de l'unité de commande

L'unité de commande est schématisée figure III.10. Toutes les opérations arithmétiques et logiques sont exécutées dans l'ALU |17|. Elle acquiert l'opérande par le BUS E et délivre le résultat de l'opération par le BUS Z.

Le séquenceur de microprogramme (SMP) donne l'adresse de la microinstruction suivante Y : soit à partir du contenu de RB, soit en prenant la valeur de l'entrée D, soit à partir du sommet de la pile intérieure au SMP, soit par simple incrémentation de la valeur actuelle de Y. Le choix dépend du microcode P_3 - P_0 de la microinstruction actuelle et de l'état de l'ALU |18|. Les microinstructions résident dans la mémoire de contrôle; la microinstruction en cours d'exécution est stockée dans le registre RMI.

La mémoire de programme MP dispose d'un registre d'adresse RAM, d'un registre de données RDM et travaille de façon habituelle.

Les registres d'entrée REN et de sortie RZ servent aux échanges avec l'extérieur.

Le registre G est mis à 1 si l'unité de commande fonctionne comme processeur général, c'est à-dire quand les microprogrammes sont stockés en page 1 de la mémoire de contrôle. Dans notre étude, nous nous limiterons à la conception du microprogramme de traitement du grafcet et travaillerons uniquement en page 0.

III.3.2.2. - Description fonctionnelle des composants

De nombreux langages de description matérielle ont été développées, chacun oriente vers une application spécifique |19|. Le langage de description utilisé en |20| nous a paru plus clair et direct. Les registres et les mémoires sont mises à part pour bien les différencier des autres composants. En effet le contenu d'un registre ou d'une mémoire est gardé jusqu'au changement suivant, tandis que pour les autres composants combinatoires, les valeurs en sortie sont définies lorsque les valeurs d'entrée sont fixées. Ce point est important aussi bien dans la phase de simulation de la description que dans la phase de compilation.

Le symbole " ← " indique l'affectation ou le chargement d'un registre ou d'une mémoire; le symbole "=" sera utilisé dans les autres cas. La simultanéité sera précisée par ";".

III.3.2.3. - L'unité arithmétique et logique (ALU)

Nous utilisons deux tranches Am 2901 pour constituer une ALU de 8 bits. Dans le tableau III.1 qui présente cette ALU, la condition est indifférente quand il n'y a aucun chiffre ou quand il y a Ø.

NOTION : Q(0-7) - Registre de 8 bits, représenté par Q

R(0-15,0-7) - 16 registres de 8 bits, R_a représente la $a^{ième}$ registre dont $j^{ième}$ bit est R_a (j)

III.3.2.4. - Séquenceur de microprogramme (SMP)

Le séquenceur (figure III.12) est défini par le tableau III.2 et utilisé les notions suivantes :

MPC(0-7), REG (0-7), PSTK(0-7) - Registres de 8 bits
P(0-3) est une pile de 4 mots de 8 bits, son sommet est STKO = P(PSTK)

III.3.2.5 - PROM 1 et MUX 2

Il s'agit d'une mémoire morte programmable de 32 x 8 bits (figure III.13). Nous en donnons une description simplifiée dans le tableau III.3.

III.3.2.6. - MUX_0 et MUX_1

Ce sont deux multiplexeurs A_m 25 LS 253 servent à réaliser un décalage (Figure III.14 et tableau III.4).

Description:

Tableau III-1

Opération	Description	Conditions (en décimal)						
		B	A	I(6-8)	I(3-5)	I(0-2)	$\overline{OE}$	H
Source	$R = R_a$; $S = Q$	ø	ø	ø	ø	0	ø	ø
	$R = R_a$; $S = R_b$					1		
	$R = 0$; $S = Q$					2		
	$R = 0$; $S = R_b$					3		
	$R = 0$; $S = R_a$					4		
	$R = De$; $S = R_a$					5		
	$R = De$; $S = Q$					6		
	$R = De$; $S = 0$					7		
Fonction	$F = R + S + C_0$	ø	ø	ø	0	ø	ø	ø
	$F = S - R + C_0$				1			
	$F = R - S + C_0$				2			
	$F = R \cup S$				3			
	$F = R \cap S$				4			
	$F = \overline{R} \cap S$				5			
	$F = R \oplus S$				6			
	$F = \overline{R \oplus S}$				7			
Destination	$Q \leftarrow F$; $Y = F$	ø	ø	0	ø	ø	0	1
	$Y = F$			1				
	$R_b \leftarrow F$; $Y = R_a$			2				
	$R_b \leftarrow F$; $Y = F$			3				
	$R_b \leftarrow F/2$; $Q \leftarrow Q/2$; $Y = F$			4				
	$R_b \leftarrow F/2$; $Y = F$			5				
	$R_b \leftarrow 2F$; $Q \leftarrow 2Q$; $Y = F$			6				
	$R_b \leftarrow 2F$; $Y = F$			7				
Sortie	$Z = Y$	ø	ø	ø	ø	ø	1	ø

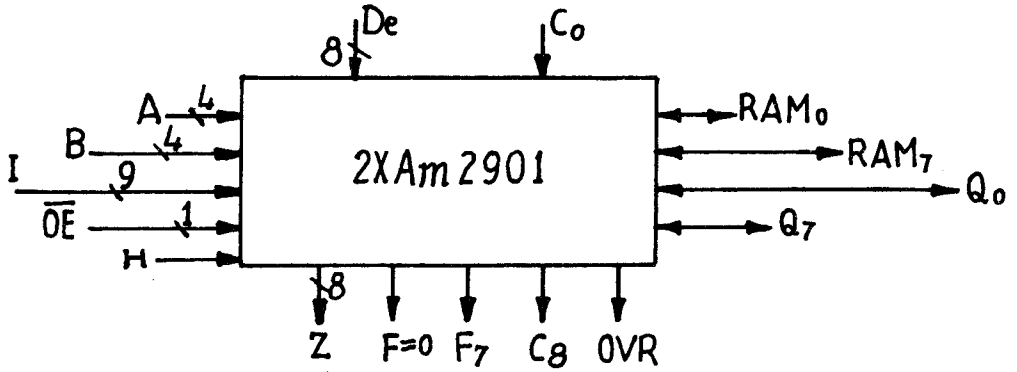


Figure III-11

Description:

Tableau III-2

Opération	Description	Conditions (en décimal)						
		S	$\overline{FE}$	PUP	$\overline{Zéro}$	$\overline{OE}$	$\overline{RE}$	H
Affectation de REG	REG ← R	Ø	Ø	Ø	Ø	Ø	0	1
	REG ← REG						1	
Contrôle de la pile	PSTK ← PSTK-1 ,	Ø	0	0	Ø	Ø	Ø	1
	P(PSTK) ← MPC , PSTK ← PSTK+1		0	1				
	PSTK ← PSTK		1	Ø				
l'Adresse Suivante	Y = MPC , MPC ← MPC+1	0	Ø	Ø	1	0	Ø	1
	Y = REG , MPC ← REG+1	1						
	Y = P(PSTK), MPC ← P(PSTK)+1	2						
	Y = D , MPC ← D+1	3						

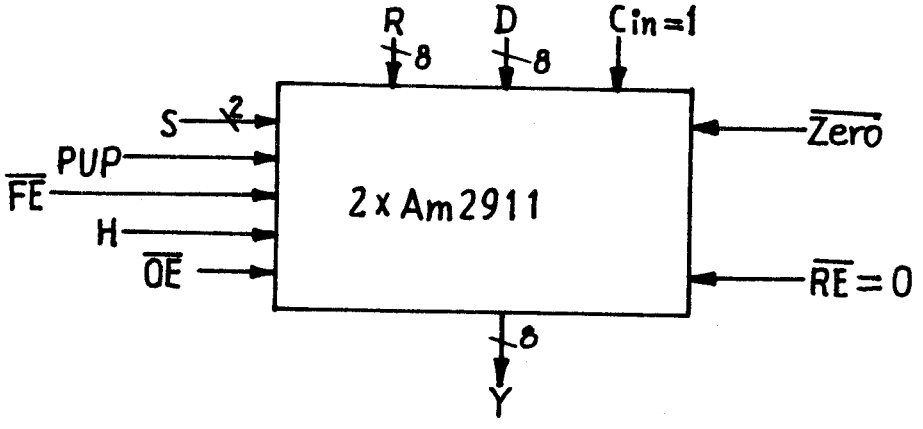


Figure III-12

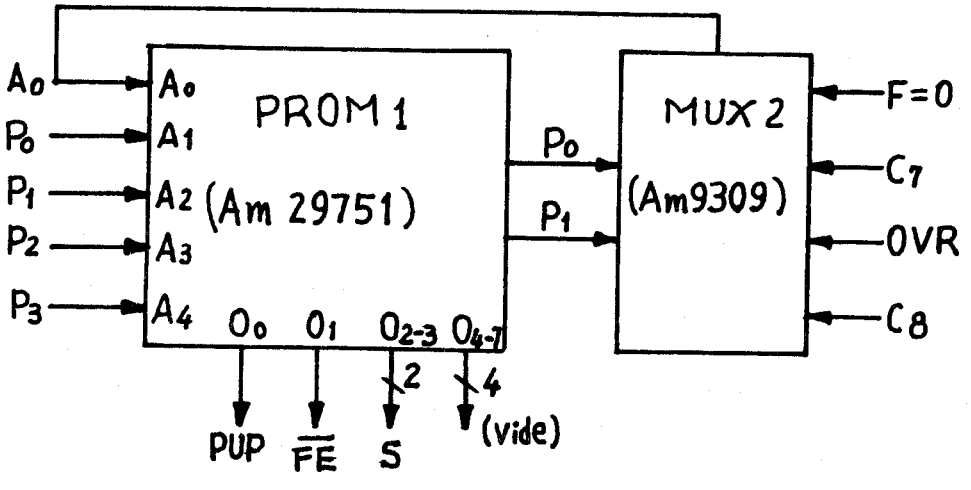


Figure III-13

Description:

Tableau III-3

Opération	Description	Conditions (en déc.)	
		P	C, OVR, F_7, F
Branch. à RB Conditionnel	$A_0 = F$, Si $F \neq 0$, $(S, \overline{FE}, PUP) = (1, 1, \emptyset)$ Sinon $(S, \overline{FE}, PUP) = (0, 1, \emptyset)$	0	$\emptyset$
Branch. à RB Inc.	$(S, \overline{FE}, PUP) = (1, 1, \emptyset)$	1	$\emptyset$
Continue	$(S, \overline{FE}, PUP) = (0, 1, \emptyset)$	2	$\emptyset$
Brach. à D Incond.	$(S, \overline{FE}, PUP) = (3, 1, \emptyset)$	3	$\emptyset$
Branch. à S.P. RB cond.	$A_0 = F$, Si $F \neq 0$, $(S, \overline{FE}, PUP) = (1, 0, 1)$ Sinon $(S, \overline{FE}, PUP) = (0, 1, \emptyset)$	4	$\emptyset$
Branch. à SP. RB In.	$(S, \overline{FE}, PUP) = (1, 0, 1)$	5	$\emptyset$
Retour du S.P.	$(S, \overline{FE}, PUP) = (2, 0, 0)$	6	$\emptyset$
Branch. à STKO Inc.	$(S, \overline{FE}, PUP) = (2, 1, \emptyset)$	7	$\emptyset$
Test de Fin de boucle et POP	$A_0 = F$, Si $F = 0$, $(S, \overline{FE}, PUP) = (0, 0, 0)$ Sinon $(S, \overline{FE}, PUP) = (2, 1, \emptyset)$	8	$\emptyset$
Continue et PUSH	$(S, \overline{FE}, PUP) = (0, 0, 1)$	9	$\emptyset$
Continue et POP	$(S, \overline{FE}, PUP) = (0, 0, 0)$	10	$\emptyset$
Test de Fin de boucle et POP	$A_0 = C_8$, Si $C_8 = 1$, $(S, \overline{FE}, PUP) = (0, 0, 0)$ Sinon $(S, \overline{FE}, PUP) = (2, 1, \emptyset)$	11	$\emptyset$
Branch. à RB Conditionnel	$A_0 = F$, Si $F = 0$, $(S, \overline{FE}, PUP) = (1, 1, \emptyset)$ Sinon $(S, \overline{FE}, PUP) = (0, 1, \emptyset)$	12	$\emptyset$

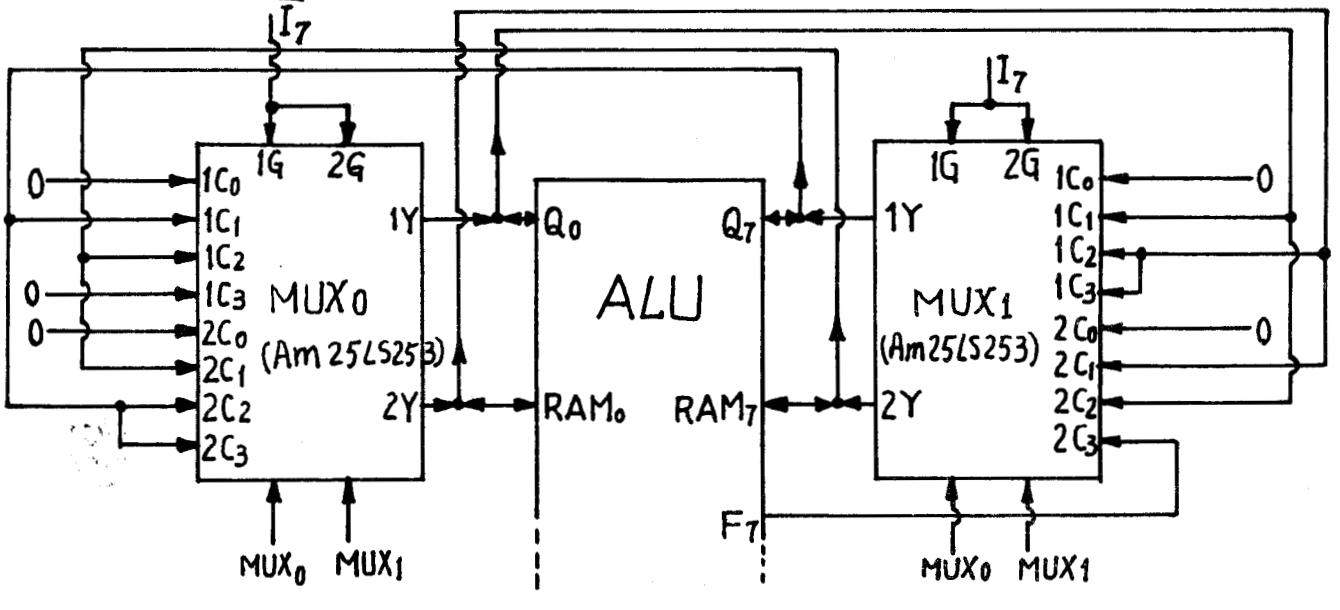


Figure III-14

Description:

Tableau III-4

Opération	Description ($i = 1\_7$)	Conditions		
		I_7	MUX_1	MUX_0
Décalage Simple à gauche	$Q(i) \leftarrow Q(i-1)$; $Q(0) \leftarrow 0$; $RAM(i) \leftarrow RAM(i-1)$; $RAM(0) \leftarrow 0$	1	0	0
Décalage Simple à droite	$Q(i-1) \leftarrow Q(i)$; $Q(7) \leftarrow 0$; $RAM(i-1) \leftarrow RAM(i)$; $RAM(7) \leftarrow 0$	0	0	0
Rotation Simple à gauche	$Q(i) \leftarrow Q(i-1)$; $Q(0) \leftarrow Q(7)$; $RAM(i) \leftarrow RAM(i-1)$; $RAM(0) \leftarrow RAM(7)$	1	0	1
Rotation Simple à droite	$Q(i-1) \leftarrow Q(i)$; $Q(7) \leftarrow Q(0)$; $RAM(i-1) \leftarrow RAM(i)$; $RAM(7) \leftarrow RAM(0)$	0	0	1
Rotation double à gauche	$Q(i) \leftarrow Q(i-1)$; $Q(0) \leftarrow RAM(7)$; $RAM(i) \leftarrow RAM(i-1)$; $RAM(0) \leftarrow Q(7)$	1	1	0
Rotation double à droite	$Q(i-1) \leftarrow Q(i)$; $Q(7) \leftarrow RAM(0)$; $RAM(i-1) \leftarrow RAM(i)$; $RAM(7) \leftarrow Q(0)$	0	1	0
Décalage double arithmétique à G.	$Q(i) \leftarrow Q(i-1)$; $Q(0) \leftarrow 0$; $RAM(i) \leftarrow RAM(i-1)$; $RAM(0) \leftarrow Q(7)$	1	1	1
Décalage double arithmétique à D.	$Q(i-1) \leftarrow Q(i)$; $Q(7) \leftarrow RAM(0)$; $RAM(i-1) \leftarrow RAM(i)$; $RAM(7) \leftarrow F(7)$	0	1	1

Description:

Tableau III-5

Opération	Description	Conditions	
		CS	CE
Non		0	Ø
Sortie aux Registres	RZB ← Z	1	
	RZH ← Z	2	
	RDM ← Z	3	
	RAMB ← Z	4	
	RAMH ← Z	5	
	RI ← Z	6	
	RENB ← Z	7	
	RENH ← Z	8	
Sortie à l'extérieur	$RZEX_{9\_15}$ ← Z	9–15	
Non		Ø	0
Entrée des Registres	E ← OPI(0-7)		1
	E ← RDM		2
	E ← RENB		3
	E ← RENH		4
Début de l'écriture	W ← 1		5
Fin de l'écriture	W ← 0		6
Début de lecture	R ← 1		7
Fin de lecture	R ← 0		8
Entrée de l'extérieur	E ← Extérieur $RENEX_{9\_15}$		9–15

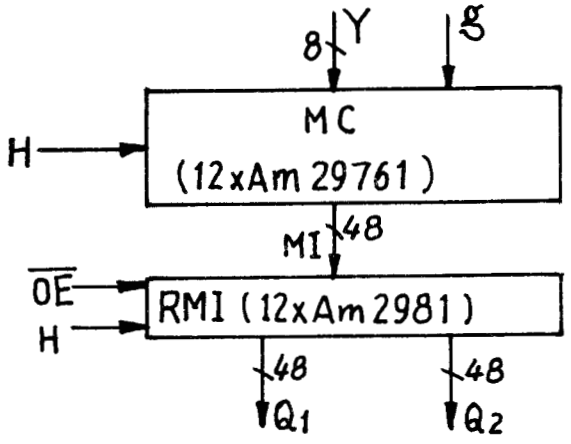


Figure III-15

III.3.2.7. - Décodage

Le décodeur 1 reçoit le code de sortie du registre de microinstruction, RMI (44-47) et fournit les signaux de contrôle de sortie, le décodeur 2 reçoit le code d'entrée RMI (40-43) et fournit les signaux de contrôle d'entrée, de lecture et d'écriture en mémoire principale(MP).

NOTION DU TABLEAU III.5

RENB = REN(0-7), RENH = REN(8-15), OPI = RMI(0-7), RDM, RZB = RZ(0-7), RZH = RZ(8-15), RAMB = RAM(0-7), RAMH = RAM(8-15), RI(0-7) - Registre de 8 bits.
CS = RMI(44-47), CE = RMI(40-43) - Sous registres de RMI.

III.3.2.8. - La mémoire de contrôle (MC) (Figure III.15)

C'est une mémoire morte de deux pages de 256 mots de 48 bits. Nous ne considérons ici que la page 0 pour stocker le microprogramme de gestion du grafcet.

NOTION

MC(0-255,0-47) - mémoire de 256 x 48 bits
RMI (0-47) - Registre de microinstruction de 48 bits

Description:

Tableau III-6

Opération	Description	Conditions			
		H	Y	g	$\overline{OE}$
Lecture	MI ← MC(Y) , RMI ← MI	1	ø	0	ø
Sortie	Q_1 = RMI ; Q_2 = RMI	ø	ø	ø	0
	Q_1 = RMI ; Q_2 = HI				1

Le terme HI signifie haute impédance

D'autres composants mineurs interviennent encore,nous n'en serons pas la description.

III.3.3 - Le microprogramme

Il nous reste à réaliser maintenant l'instruction microprogrammée qui sera exécuter par la machine structurée comme sur la figure III.10 et mise en oeuvre suivant la stratégie de la figure III.9.

Les méthodes de microprogrammation ont été largement exposées [18] [21] [22], nous détaillons simplement au dessous les résultats de notre travail.

Les séquences de microopérations sont données par les figures A-1 à A-8. En regard de chaque microopération, il y a un numéro, indiquant l'adresse de la microinstruction correspondante de la figure A-9.

A côté du sous-microprogramme (SMPi), il y a deux indications : la première indique le numéro de la microinstruction appelante, la seconde représente l'adresse de début du sous-microprogramme.

Le microprogramme et tous les sous-microprogrammes sont donnés à la figure A-9 en décimal. Le premier nombre est l'adresse de chaque microinstruction, les autres chiffres sont les codes de microinstruction, dont le format est précisé à la figure III.16.

Le sous-microprogramme de lecture SMP1 et le sous-microprogramme d'écriture SMP2 ne sont pas concrétisés car la mémoire principale(MP) n'est pas fixée. C'est pourquoi à la figure A-7a) la microinstruction correspondante n'est pas numérotée.

Sur la figure A-9, le numéro 191 désigne le début de lecture, 195 le début d'écriture et 198 un retour. Les microinstructions 192-194,196-197 sont vides. Cette procédure a pour but de poursuivre le microprogramme pour la phase de simulation.

Nº de Champ	11				10				9	8	7				6			
Nº de bit	47	46	45	44	43	42	41	40	39←36	35←32	31	30	29	28	27	26	25	24
Définition des bits	CS_3	CS_2	CS_1	CS_0	CE_3	CE_2	CE_1	CE_0	RB(4-7)	RB(0-3)	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6
Définition du Champ	Destination de BUS Z				Source de BUS E				RB(0-7)		PROM 1 du séquenceur				MUX_1	Destination de AUL		

	5				4				3				2				1	0
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7←4	3←0
MUX_0	I_2	I_1	I_0	C_0	I_5	I_4	I_3	a_3	a_2	a_1	a_0	b_3	b_2	b_1	b_0	OPI(4-7)	OPI(0-3)
MUX_0	Source de ALU			C_0	Fonction de ALU			l'adresse de Ra : A(0-3)				l'adresse de Rb : B(0-3)				OPI (0-7)	

Figure III-16

III.4 - Conclusion

Après avoir discuté sur les modèles de représentation de nos systèmes et sur les méthodologies d'implantation sur machine programmée, nous allons vérifier au chapitre IV par une simulation la valeur du travail présenté dans ces trois premiers chapitres.

CHAPITRE IV : SIMULATION

Le rôle de la simulation est d'effectuer un certain nombre de test sur une nouvelle conception d'un produit de façon à éliminer le maximum d'anomalies avant la mise en fabrication. Toutefois il ne faut pas attendre tout de la simulation car un examen exhaustif de tous les cas possibles est illusoire.

Dans le système que nous proposons, la simulation doit permettre d'affiner deux questions importantes :

. la méthode d'implantation et de réalisation satisfait-elle le cahier des charges au niveau du système global?

. le microprogramme et l'architecture de l'unité de commande réalisent-ils les fonctions du programme de gestion au niveau du microcode ?

Nous avons fait trois expériences de simulation pour répondre à ces deux questions. Il est possible de mettre en oeuvre d'autres simulations à des niveaux divers entre le niveau système et le niveau microcode soit en utilisant une méthode descendante soit une méthode ascendante |23|.

Le langage BASIC a été utilisé compte tenu de sa disponibilité, même si d'autres langages plus orientés simulation |24| auraient été préférables.

IV.1 - Traduction du langage de description en BASIC

Nous nous limiterons dans ce paragraphe à la présentation de la traduction d'un nombre limité d'instructions.

IV.1.1. - Les opérations arithmétiques

	Langage de description	Basic correspondant	Commentaire
1	A ← a	A = a	A est un registre, a est un nombre
2	A ← B	A = B	A,B sont deux registres
3	A(0-n)←B(0-n)	FOR I=0 To n A(I) = B(I) NEXT I	A,B sont deux registres de n+1 bits
4	A ← B $\pm$ C	A = B $\pm$ C	A,B,C sont trois registres
5	A ← M(B)	A = M(B)	M(B) est le contenu de la mémoire M à l'adresse B, M est un tableau : DIM M(n).
6	A ← M(B,C)	A = M(B,C)	M(B,C) est le contenu de la mémoire M à l'adresse C de la page B, M est un tableau à deux dimensions : DIM M(m,n)
7	M(B) ← A	M(B) = A	même chose que 5
8	M(B,C) ← A	M(B,C) = A	même chose que 6

IV.1.2. - Les opérations logiques

Le langage BASIC n'est pas très riche en opération logique. La procédure utilisée n'est certes pas la meilleure mais est utilisable dans notre simulation.

	Langage de description	BASIC Correspondant	Commentaire
1	Q ← 0	Q = 0	0 est "faux"
2	Q ← 1	Q = 1	1 est "vrai"
3	Q ← $\overline{Q}$	IF Q = 0 THEN Q=1 (ELSE) Q=0	
4	F ← R∩S	F = R * S	"*" est la multiplication arithmétique.
5	F ← RUS	F = R+S IF F=0 THEN F=0 (ELSE) F=1	"+" est l'addition arithmétique
6	F ← R⊕S	IF R<>S THEN F=1 (ELSE) F=0	"<>" est ≠ en BASIC

IV.1.3. - D'autres opérations

Langage de description	BASIC correspondant	Commentaire
1 $Q \leftarrow Q/2$	soit Q=Q/2 soit FOR I=0 TO n-1 {Q(I) = Q(I+1)/NEXT I Q(n) = 0	Q est un registre à n+1 bits
2 $Q \leftarrow 2Q$	soit Q = 2Q soit FOR I=0 TO n-1 Q(I+1) = Q(I)/NEXT I Q(0) = 0	
3 Saut à I_k	GOTO EI_k	I_k est une instruction, EI_k est l'étiquette de I_k dans le programme en BASIC.
4 Saut à SP_k	GOSUB ESP_k	SP_k est un sous-programme, ESP_k est l'étiquette de SP_k en BASIC.
5 Si C est vrai Saut à I_k	IF C=1 GOTO EI_k	C est la condition de branchement
6 Si C est vrai Saut à SP_k	IF C=1 GOSUB ESP_k	
7 Si C=1 à I_k C=2 à I_1 C=3 à I_m	ON C GOTO EI_k,EI_1,EI_m	

IV.2 - Simulation de calcul en contrôle parallèle

Cette expérience de simulation a pour but de vérifier les principes de parallélisme proposés au chapitre II au niveau système et de vérifier l'implantation définie au chapitre III.

Les points suivants ont été mis en évidence :

. la dépendance logique entre les sous tâches doit être respectée. N'importe quel processeur traite n'importe quelle sous-tâche et chaque sous-tâche a une durée propre

. les sous-tâches indépendantes peuvent être traitées parallélement si les processeurs sont disponibles.

. les conflits entre les sous-tâches doivent être arbitrés selon un principe de priorité fixée.

. les règles d'utilisation des processeurs sont faites sur le principe d'une priorité relative circulaire (PRC).

IV.2.1. - Le programme de simulation

Le programme est donné par la figure A·10 que nous commentons :

IV.2.1.1. - Les variables et les registres simulés

En BASIC un symbôle est constitué d'une lettre suivie d'un chiffre. On ne peut donc pas écrire directement les noms des variables et des registres du système simulé, quand ils sortent de la norme BASIC. Pour les éléments de la figure A·10, la correspondance est la suivante :

$Q_1(16)$ correspond au registre d'entrée REN de la figure III.10. Les variables d'entrée sont stockées dans la liste d'entrée LE de la mémoire. Nous avons donc $Q_1(0\text{-}12)$ = REN (0-12) = {M,REQ_1,REQ_2,PR_1,PR_2,PR_3, PR_4, PR_5,PR_6,PR_7,PR_8,PR_9,PR_{10}} et M (0,48-62) = {REN(0-12),FF,FF}. La liste d'entrée de la page A.18 donne les valeurs initiales.

$Q_2(16)$ simule la concaténation des registtes R_0,Q, de l'ALU. d'où $Q_2 = R_0Q$.

$Q_3(16)$ représente le registre de sortie RZ de la figure III.10.

$Q_4(16)$ est vide

P(16) enregistre chaque sous-tâche activée et l'occupation de son processeur.

Par exemple P(8) = 2 et P(9) = 1 signifient que OP_7 soit $Y_7 = 1 + EXP\ (Y_4)$, et OP_8 soit $Y_8 = Y_5\ Y_6/Y_7$, sont en cours d'exécution respectivement par les processeurs 2 et 1.

$Y_0(16)$ est un tableau intermédiaire pour calculer Y_{10}.

M(3,256) correspond à la mémoire principale MP de trois pages de 256 mots.

R_i simule le registre R_i correspondant de l'ALU, i=0-9.

S_j simule le registre $R_{10} + j$ correspondant de l'ALU, j=0-5.

Par exemple S_o simule R_{10} et S_5 simule R_{15}.

D représente le registre RDM

Z_9 simule un compteur extérieur qui compte le nombre de mots de sortie. Ce compteur peut être RZEX9 du tableau III.5 . Les variables à sortir ont en effet plus de 16 bits et sont regroupées en trois mots séparés par FF. Dans la liste de sortie (LS) de la mémoire , nous avons : M(0,96-124)= {RES1,RES2,FF; OP_1,OP_2,OP_3,OP_4,OP_5,OP_6,OP_7,OP_8,OP_9,OP_{10},OP_{11}, FF; Bt_1,Bt_2,Bt_3,Bt_4,Bt_5,Bt_6,Bt_7,Bt_8,Bt_9,Bt_{10},Bt_{11},Bt_{12},FF; FF}.
La liste de sortie de la page A.18 donne les valeurs initiales.

D'autres listes de variables (figure A·11) contiennent les valeurs initiales des variables ci-dessous :
LC = M(0,0-17) ne diffère de la figure III-8 a) que par M(0,13)=13 qui est le nombre de variables d'entrée.

LI = M(0,20-28) = {ad E_0, ad E_{20}, FF; ad m_{17}, adm_{19}, adm_{20}, adm_{22}, FF;FF}

LVI = M(0,144-180) = {r_8, S_8,m_8,FF; r_{10},S_{10},m_{10},FF; r_{14}, S_{14},m_{14}, FF; r_{17},S_{17},m_{17},FF; r_{19},S_{19},m_{19},FF; r_{20}, S_{20}, m_{20},ad E_{20}; r_{21},S_{21},m_{21},ad E_{21}; r_{22},S_{22},m_{22},FF; r_{23},S_{23},m_{23},FF;FF}.

LCG = M(1,32-43) = {ad SP_{20},ad Bt_5, FF; ad SP_{21}, ad Bt_7,FF; ad SP_{22}, ad Bt_9, FF; ad SP_{23}, ad RES_1,FF; ad SP_{24}, ad RES_2,FF;FF}.

Les listes LCE et LRDP sont beaucoup plus longues, c'est pourquoi nous ne donnons que leurs adresses principales c'est-à-dire l'adresse de départ du sous-programme réceptivité et l'adresse de chaque étape clé.

LCE est stockée à la page 1 : ad SP_0 = 48, ad SP_1 = 53, ad $SP_{2,3,4,10,11}$ = 56, ad SP_5 = 61, ad SP_6 = 68, ad SP_7 = 77, ad SP_8 = 84, ad SP_9 = 93, ad SP_{12} = 96, ad $SP_{13,16}$ = 101, ad $P_{14,18}$ = 106, ad SP_{15} = 113, ad SP_{17} = 120, ad SP_{20} = 127, ad SP_{21} = 130, ad SP_{22} = 133, ad SP_{23} = 136.

LRDP est mémorisée à la page 2 : ad E_0 = 2, ad E_1 = 11, ad E_2 = 24, ad E_3 = 36, ad E_4 = 48, ad E_5 = 60, ad E_6 = 72, ad E_7 = 87, ad E_9 = 99, ad E_{11} = 118, ad E_{12} = 130, ad E_{13} = 142, ad E_{115} = 153, ad E_{16} = 166, ad E_{18} = 176, ad E_{20} = 186, ad E_{21} = 207.

Utilisant les adresses et les figures III.4 et III.6, il est possible de comprendre la table de données du GRAFCET (figure A·11).

IV.2.1.2. - Le programme de gestion simulée

Le programme de gestion est simulé en pas à pas. La ligne 21 de la figure A·10 correspond au début de la figure A·1 a. L'initialisation, l'acquisition des entrées sont ensuite réalisées d'après la figure III.9 et les figures A·1 à A·8. Quelques sous-microprogrammes devenus trop élémentaires en BASIC ne sont pas mis en sous-programme. (par exemple SMP1, SMP6,SMP9 - SMP11 - SMP14). Par contre SMP7 (à la ligne 420), SMP8 (à la ligne 427) et SMP10 (à la ligne 435) sont traités dans le programme de simulation comme des sous-programmes.

Un cycle de simulation est compris entre les lignes 90 et 419.

IV.2.1.3. - Simulation de traitement par les processeurs

Le système simulé sur un calculateur unique comporte trois processeurs : un spécialisé (l'unité de commande programmée), deux banalisés. Le temps de traitement est partagé entre les trois processeurs.

A chaque fois que le programme de gestion atteint la ligne 375, un mot de sortie est affecté. Le programme passe alors au sous-programme exécution et simule l'exécution d'opérations extérieures en processeur banalisé. Le premier mot de sortie est $\{RES_1, RES_2\}$ qui indique si les processeurs P1 et/ou P2 sont alloués.(lignes 495 et 497 de la figure A·10). Le second mot de sortie est $\{OP_1, OP_2,\}$ qui met P(I) à un ou à deux suivant que P_1 ou P_2 est à 1; il enclenche l'opération correspondante (lignes 501-521).

Pendant la sortie du second mot, si une certaine opération, par exemple OP_3 est enclenchée, elle est exécutée immédiatement (lignes 615-623). Toutefois elle ne peut se terminer qu'en fonction du résultat d'une fonction aléatoire (lignes 535-541). Si T > 5 par exemple le prédicat PR3 ne peut être envoyé et cette opération poursuit son traitement.

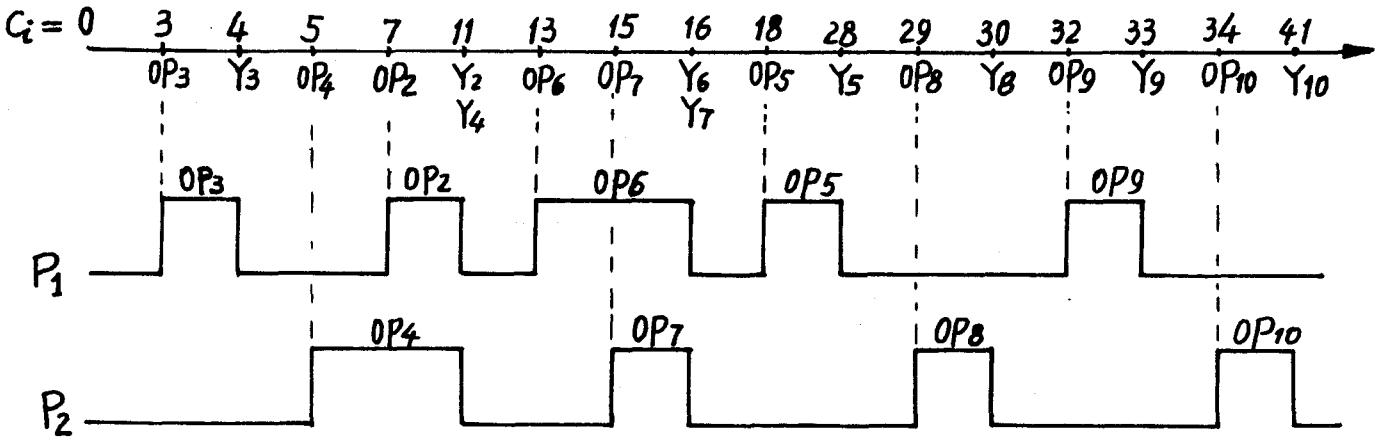
L'ordinateur saute alors à l'exécution du programme suivant ou au début d'un nouveau cycle du programme de gestion. Une opération peut donc durer plusieurs cycles de traitement jusqu'à rencontrer T < 5.
A ce moment, l'opération s'achève, le résultat Y_3 est imprimé, le prédicat PR3 est transmis et le processeur redevient libre.

La fonction aléatoire joue trois rôles dans la simulation:

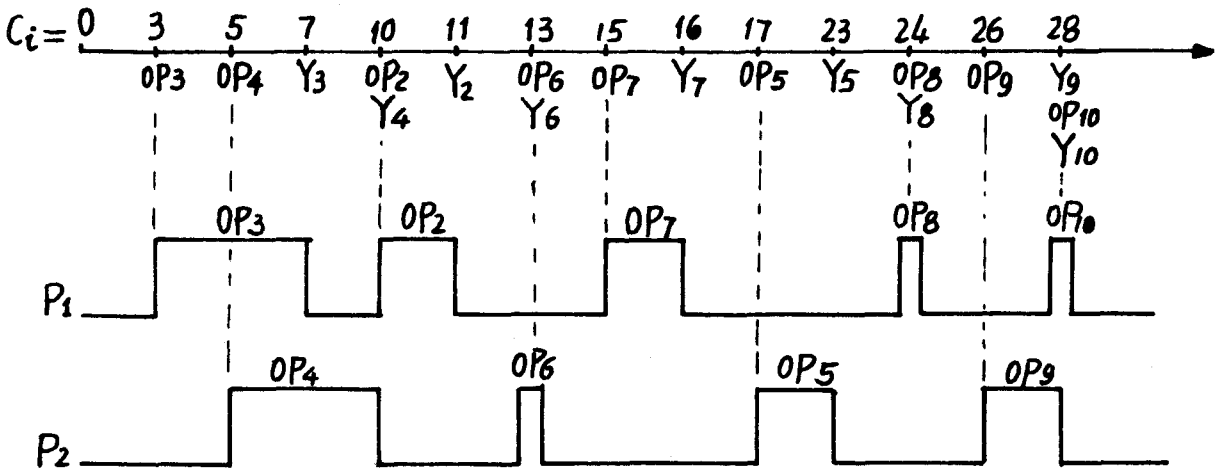
- elle permet la simulation des traitements parallèles car il n'est pas nécessaire de réaliser des traitements réels mais simplement de faire une simulation stochastique des tâches.
- elle fournit en fait la durée d'exécution d'une opération.
- il est enfin possible en jouant sur le test T de faire varier le rapport de la durée du traitement d'une opération relativement à la durée du cycle.

IV.2.1.4. - La priorité des sous-tâches

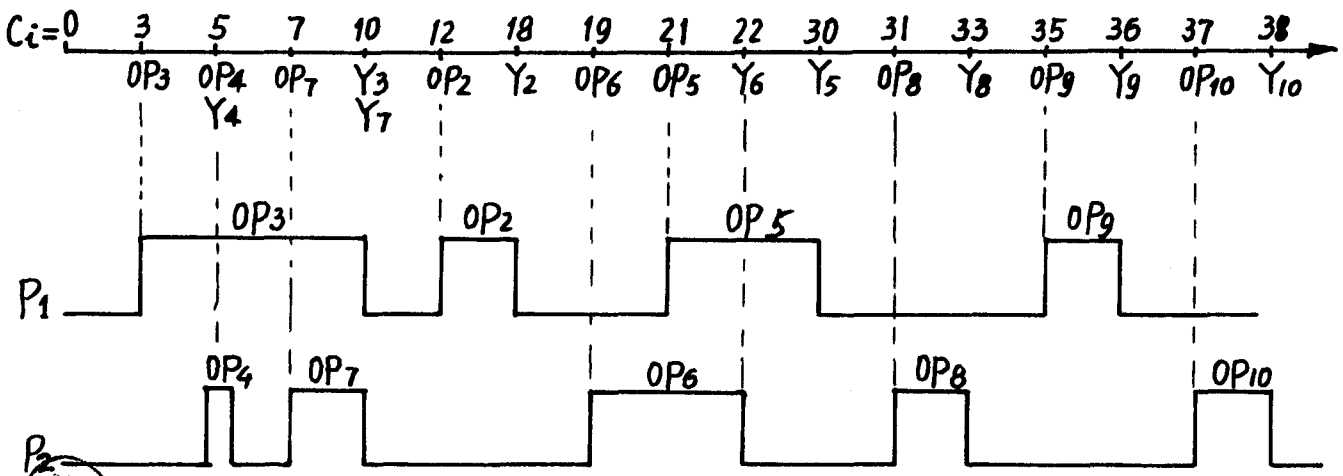
Le grafcet donne les règles d'assignation des processeurs banalisés . La priorité des sous-tâches est fixée par la liste LRDP. Si, par exemple, OP_2 OP_3 et OP_4 sont lancées respectivement au franchissement des transitions t_2 t_3 et t_4, les étapes E_2, E_3, E_4, successeurs de la transition t_1, sont rangés dans l'ordre E_3, E_4, E_2 pour avoir la priorité $OP_3 > OP_4 > OP_2$.



a) Premier cas



b) Deuxième cas



c) Troisième cas

Figure IV.1

De la même façon, nous fixons $OP_9 > OP_{10}$. Les autres opérations sont écrites sans relation d'ordre dans LEAC et LEAS.

IV.2.2. - L'analyse des résultats

Nous ne donnons ici que les résultats de simulation avec la condition de test aléatoire T > 2 (figure A-12).

Les trois cas de simulation montrent les points suivants:

1) la dépendance logique entre les opérations (sous-tâches) a été respectée. La figure IV.1 donne les ordres d'apparition de OP_i, Y_j et P_k.

L'opération OP_8 n'est lancée qu'après production des résultats de OP_5, OP_6, OP_7 c'est-à-dire Y_5,Y_6 et Y_7. Les opérations OP_9 et OP_{10} suivent obligatoirement Y_8; OP_5 vient après Y_2, OP_6 après Y_3, OP_7 après Y_4.

2) Les sous-tâches indépendantes peuvent être traitées en parallèle. En effet les sous-ensembles $\{OP_3, OP_4\}$ $\{OP_4,OP_2\}$, $\{OP_3,OP_7\}$ $\{OP_6,OP_7\}$, $\{OP_6,OP_5\}$ $\{OP_9,OP_{10}\}$ ont été traités, en parallèle dans la simulation.

3) les conflits sont réglés correctement. Par exemple quand OP_2 OP_3 OP_4 partagent deux processeurs, le résultat est en faveur de OP_3 puis de OP_4 et enfin de OP_2.

Les processeurs sont par ailleurs alternativement alloués s'ils sont en conflit sur une sous-tâche.

4) Les processeurs sont bien banalisés. Les opérations OP_5 - OP_{10} peuvent être exécutées aussi bien par le processeur 1 ou 2.

IV.3 - Simulation de calcul en contrôle mixte

Cette simulation a pour but de prouver la possibilité d'organiser un calcul en contrôle mixte sans aucun changement du programme de gestion.

IV.3.1. - Le programme de simulation

Ce programme est donné à la figure A·13. Le programme de gestion est le même qu'auparavant les lignes 21 à 469 sont donc identiques.

IV.3.1.1. - Les variables et les registres simulés

$Q_1(16)$, $Q_2(16)$, $Q_3(16)$, $Q_4(16)$, P(16), $Y_0(16)$, M(3,256), R_i S_i, D, Z_9 ont la même définition que sur la figure A.10 (paragraphe IV.2.11); toutefois leurs contenus sont modifiés.

$Q_1(16)$, simule REN et $Q_1(0\text{-}14)$ = REN (0-14) = (M,REQ_1, REQ_2, REQ_3, REQ_4, PR_1, PR_2, PR_3, PR_4, PR_5, PR_6, PR_7, PR_8, PR_9, PR_{10}). Les entrées sont stockées aux cases de la mémoire M(0,48) - M(0,62). La liste d'entrée de la page A.26 contient les valeurs initiales des variables d'entrée.

Z_9 simule un compteur extérieur qui compte le nombre de mots de sortie. Les valeurs initiales sont données par la figure A·14.

$Z_9 = 1$: $Q_3(0\text{-}3)$ = M(0,96-99) = (RES1, RES2, RES3, RES4)

$Z_9 = 2$: $Q_3(0\text{-}10)$ = M(0,101-111) = (OP_1 - OP_{11})

$Z_9 = 3$: $Q_3(0\text{-}9)$ = M(0,113-122) = (Bt_1 - Bt_{10})

Entre chaque mot il y a le Séparateur FF.

La liste LC a le même contenu que celle de la figure III.8 a) sauf M(0,13) = 15, (il y a 15 variables d'entrée) et M(0,5) = 140. Les bits LE et LS sont déjà décrites sous les références $Q_1(16)$ et Z_9.

La liste LVI s'écrit : M(0,140-184) = (r_0,S_0,m_0,FF; r_{26}, r_8,S_8,m_8,FF; r_{10},S_{10},m_{10},FF; r_{14},S_{14},m_{14},FF; r_{24},S_{24},m_{24},FF; r_{25},S_{25},m_{25},FF; S_{26},m_{26},ad E_{26}; r_{27},S_{27},m_{27},FF; r_{28},S_{28},m_{28},FF; r_{29},S_{29},m_{29},FF; r_{30},S_{30},m_{30},FF;FF).

La liste LI s'écrit : M(0,20-34) = (ad E_{17},ad E_{19}, ad E_{21}, ad E_{23}, ad E_{26}, FF, ad m_0, ad m_{24}, ad m_{26}, ad m_{27}, ad m_{28}, ad m_{29}, ad m_{30}, FF, FF).

La liste LCG s'écrit : M(1,32-41) = (ad SP_{23}, ad Bt_5,FF, ad SP_{24}, ad Bt_7, FF, ad SP_{25}, ad Bt_9, FF,FF).

La liste LCE se trouve à la page 1 de la mémoire principale. Les adresses de début de chaque sous-programme sont les suivantes :

80 - SP_0, 87 - SP_1, 94 -{SP_2, 3,4,10,11}, 99 - SP_5, 106 - SP_6, 117 - SP_7, 124 - SP_8, 135 - SP_9, 142 - SP_{12}, 147 - SP_{13}, 150 - SP_{14}, 155 - SP_{15}, 158 - SP_{16}, 163 - SP_{17}, 166 - SP_{18}, 171 - SP_{19}, 174 - SP_{20}, 181 - SP_{21}, 186 - SP_{23}, 189 - SP_{24}, 192 - SP_{25}.

La liste LRDP est stockée à la page 2 de la mémoire principale. Chaque étape clé est repérée par l'adresse : 2-E_1, 15-E_2, 26-E_3, 37-E_4, 48-E_5, 60-E_6, 78-E_7, 91-E_9, 110-E_{11}, 123-E_{12}, 134-E_{13}, 145-E_{15}, 159-E_{16}, 167-E_{17}, 178-E_{18}, 186-E_{19}, 197-E_{20}, 205-E_{21}, 216-E_{22}, 224-E_{23}, 235-E_{26}.

A partir ce des adresses et des figures III.4, III.6, nous pouvons comprendre la table de données du GRAFCET RDP3 de la figure A-14.

Chaque paire d'entrée < X_1, x_2> correspond à un cas que nous notons dans N_0(16). Si N_0 (i) = j, l'opération OP_i est en cours d'exécution au j-ième cas d'entrée.

IV.3.1.2. - Le programme de gestion simulé

Il est fréquent de vouloir modifier le principe de contrôle, pipe line, parallèle ou mixte, au gré d'une utilisation. Toutefois ici cela peut se faire sans modification du programme de gestion.

IV.3.1.3. - La simulation du traitement par les processeurs

Le système simulé comporte cinq processeurs : l'unité de commande et quatre processeurs banalisés. Entre les figures A-10 et A-13, la différence réside dans les priorités. Nous fixons ici : Processeur 1 > Processeur 2 > Processeur 3 > processeur 4.

La méthodologie de simulation a été expliquée au paragraphe IV.2.1.3.

IV.3.1.4. - La priorité des sous-tâches

Le principe de priorité des sous-tâches est identique à celui du paragraphe IV.2.1.4.; l'ordre Y a simplement été changé : $OP_4 > OP_2 > OP_3$ et $OP_{10} > OP_9$.

IV.3.2. - Remarque sur les résultats

Nous avons considéré 7 cas. Par exemple au cycle 26 il y a quatre opérations en cours d'exécution, OP_2,OP_3,OP_4 et OP_9. OP_2 est exécutée par le processeur 1 sur les données du deuxième cas, OP_3 par le processeur 4 avec les données du deuxième cas, OP_4 par le processeur 2 sur les données du deuxième cas et OP_9 par le processeur 3 sur les données du premier cas.

Nous échantillonons les résultats de la figure A·15 ce qui donne l'illustration de la figure IV.2. Quelques points sont intéressants :

1) la dépendance logique entre les sous-tâches est respectée. Les premiers et quatrième cas sont exécutés sur les mêmes entrées que la figure A·12. L'ordre des calculs n'est pas identique mais les résultats égaux.

2) le principe de contrôle mixte est valable. Nous donnons aux variables d'entrée les valeurs suivantes :

$\{x_1,x_2\}=\{1,1\},\{0.8,0.8\},\{0.5,0.5\},\{1,1\},\{0.8,0.8\}\ \{0.5,0.5\}$

Les données circulent de façon très mélangées. Par exemple au cycle 41 dans la mémoire commune il y a les résultats de OP_9 et OP_{10} du cas 1, les résultats OP_2,OP_3,OP_4,OP_5,OP_6 et OP_7 du cas 2, les résultats intermédiaires de OP_8 du cas 2 et les données d'entrée du cas 3 (figure IV.2).

3) les conflits sont réglés correctement.

Sur la figure A·14, les priorités données au paragraphe IV.3.1.4. sont toujours respectées. Le principe d'allocation des processeurs étant Proc 1 > Proc 2 > Proc 3 > Proc 4, la figure IV.2 montre que le processeur 1 est très occupé et le processeur 4 presque en état de famine.

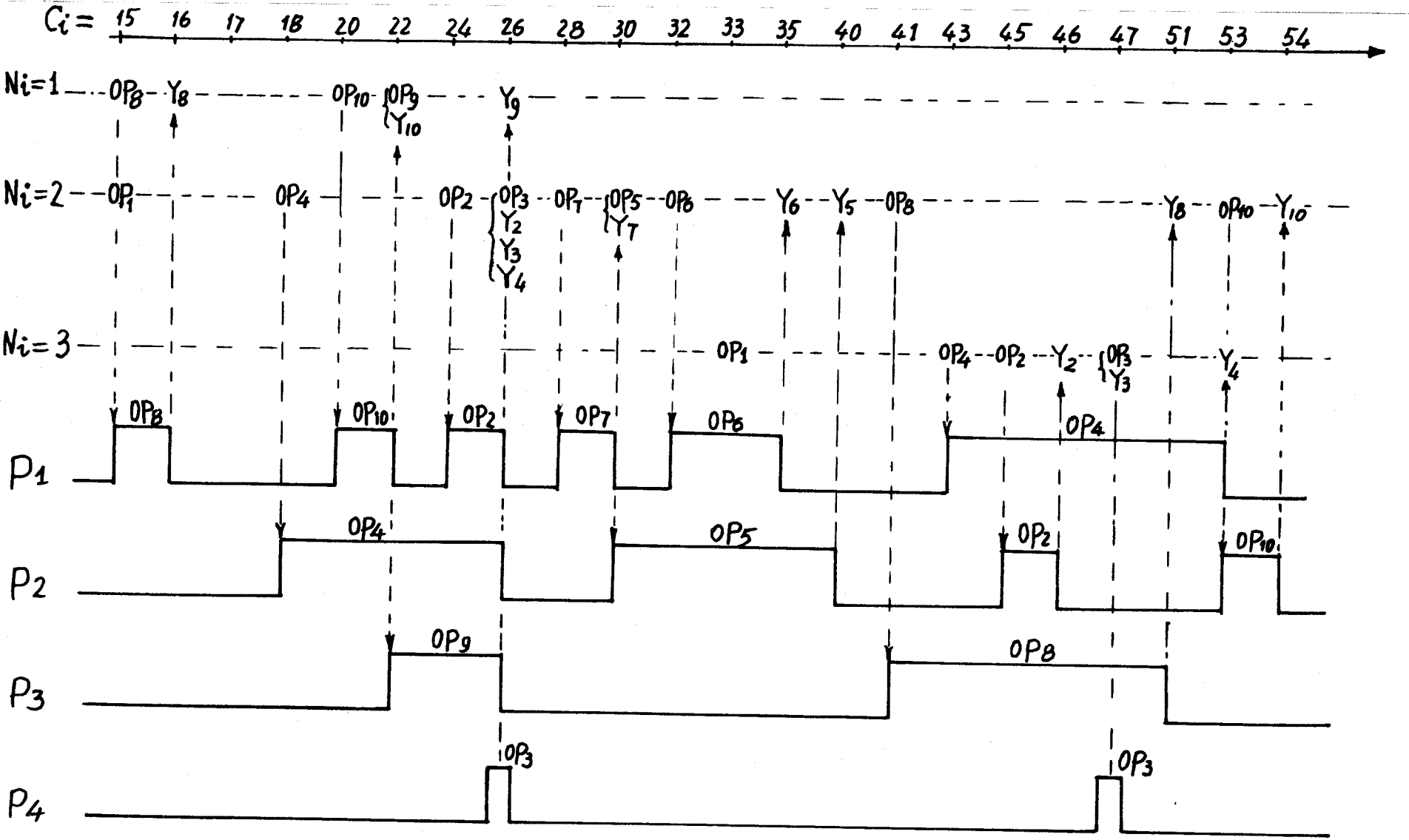

Ci = 15 16 17 18 20 22 24 26 28 30 32 33 35 40 41 43 45 46 47 51 53 54
Ni=1
Ni=2
Ni=3
P1
P2
P3
P4

Figure IV.2

4) Les processeurs sont bien banalisés. La disponibilité du système est très haute. Par exemple si P_1 est en panne, P_2 et P_3 deviennent plus occupés et le processeur 4 moins affamé.

IV.4 - Vérification du microprogramme

Les deux expériences de simulation ne se déroulent avec succès que si le programme de gestion coïncident strictement avec le microprogramme des figures A·1 à A·8. La mémoire de contrôle contiendra les codes du microprogramme de la figure A·9. Le problème est donc de vérifier la coïncidence des codes et des microopérations des figures A·1 à A.8.

IV.4.1. - Explication de la méthode

IV.4.1.1. - Choix correct du grafcet traité

Une manière simple de valider une instruction micro-programmée est de regarder la trace du microprogramme lors de l'exécution de chaque microinstruction.

Pour tester notre gestion minoprogrammée, il faut choisir un grafcet suffisamment général mais le plus simple possible. Nous utilisons le grafcet de la figure III.7.

Le programme de simulation est illustré figure A·16. Il prend à la ligne 35 une microinstruction et l'interprête, puis il exécute chaque microordre à partir de la table de données du grafcet RDP2 (figure III.8) puis il prépare l'adresse de la microinstruction suivante (lignes 285-379) et continue.

IV.4.1.2. - Les variables et les registres

Q(16) simule la concaténation des registres R_0 et Q, de l'ALU.

$Q_1(16)$ correspond au registre d'entrée REN. Q_1(0-2) = REN(0-2)= $=\{M_1, M_2, PR\}$.

La liste LE a été donnée à la figure III.8.

$Q_3(16)$ représente le registre de sortie RZ. $Q_3(0\text{-}2)$= = RZ(0-2) = {RES,OP,Bt}.

La liste LS a été montrée à la figure III.8.

$Q_5(2)$ simule le registre d'adresse de la mémoire RAM. $Q_5(0)$ = RAMB,

$Q_5(1)$ = RAMH. $M(Q_5(1), Q_5(0))$ = x signifie l'écriture de x à la case mémoire correspondante.

$Q_6(10)$ simule le registre RMI qui contient la microinstruction exécutée. $B_0(8)$ contient le code binaire de $Q_6(6)$.

$D_7(3)$ simule la sortie de PROM1, $D_7(0\text{-}3)$ = {PUP,$\overline{FE}$,S}

I(3) contient le code décimal codé binaire de I_0-I_2, I_3-I_5 et I_6-I_8.

$M_3(4)$ simule la pile du séquenceur SMP.

$R_0(8)$, $S_0(8)$, Z(8), E(8), F(8) représentent respectivement R, S, BUS Z, BUS E et F. Si l'opérande est en décimal, nous utilisons seulement $R_0(0)$, $S_0(0)$, Z(0), Z(0) et F(0). Si l'opérande est en binaire, nous utilisons les 8 bits.

{K_0, Y, T_2} simulent {STKO, Y, PSTK}

IV.4.1.3. - Le cadencement

Une microinstruction horizontale comprend plusieurs microopérations exécutées parallèlement par rapport au cycle de la microinstruction. Dans le cadre d'un micro-cycle, chaque microopération possède sa cadence propre, par exemple, le tableau III.1 montre que l'opération Q ← F ne peut fonctionner qu'après le choix des sources d'opérande.

L'introduction en BASIC de temporisation n'est pas aussi facile directement que dans certains langages (25) (26). Il faudrait d'ailleurs définir la valeur des retards introduits par chaque composant. Nous nous contentons donc de simuler le cadencement relatif implicitement compris dans le programme de la figure A·16.

Par exemple, les lignes 39-55 sont exécutées avant la ligne 171, ce qui définit le choix des opérandes avant l'exécution de Q ← F.

IV.4.2. - La trace du microprogramme

La trace du microprogramme est donnée figure A·17 par les numéros des microinstructions exécutées. Nous avons analysé la trace par rapport aux figures A·1 à A·8. Les résultats sont satisfaisants. Nous allons les analyser globalement et détailler un seul bloc, le bloc J.

IV.4.2.1. - Analyse globale

Après avoir traité les blocs A et B, l'adresse de l'étape E1 (Ad E1 = 1) est initialement inscrite dans la liste LEAS et m_1 et m_2 (ad m_1 = 146, ad m_2 = 150) dont initialement mis à un dans la liste LVI.

Au cycle 1, il y a une seule étape E_1 à traiter. La trace parcourt également les blocs C D E F G H E J K L M. Le premier traitement du bloc E correspond à la prise de Ad E_1. Le deuxième E correspond au séparateur FF, fin de la liste LEAC. Après l'exécution du cycle 1, l'étape E_3 est activée (ad E_3 = 21) et l'étape E_2 reste active (ad m_2 = 150). Donc il y a " LEAS : 21" et "LMEU : 150".

Au cycle 2, il y a une seule étape clé E_3 à traiter. La trace globale est aussi C,D,E,F,G,H,E,J,K,L,M. Compte tenu des différentes étapes à traiter d'un cycle à l'autre, le détail de chaque bloc montre des traces locales différentes. Après exécution du cycle 2, l'étape E_4 (ad E_4 = 33) est activée, l'étape E_1 est de nouveau activée (ad E_1 = 1 et ad m_1 = 146). Donc LEAS est 33, 1 et LMEU est 146, **donc** la **trace** globale est correcte.

IV.4.2.2. - Analyse du bloc J

Nous analysons ici la trace du bloc J compte tenu de sa complexité. Les numéros 109, 110, 111, 112, 113 et 122 doivent apparaître

deux fois à chaque cycle car il y a deux ensembles de variables internes à traiter : $\{r_1, S_1, m_1\}$ et $\{r_2, S_2, m_2\}$.

Au cycle 1, la transition t_1 est franchie : $m_{1+} = m_1 . \overline{r_1} + S_1 = 0$ Donc $s_1 = 0$ correspond au passage de 114 à 115 et $m_1 \overline{r_1} = 0$ correspond au passage de 117 à 122. La transition t_2 n'est pas franchie : $m_{2+} = m_2 \overline{r_2} + s_2 = 1$. Donc $s_2 = 0$ (deuxième passage 114 à 115) et $m_2 \overline{r_2} = 1$ correspond au passage de 115 à 118.

Au cycle 2, la transition t_3 est franchie ce qui conduit à l'activation des étapes E_1 et E_4 et à la désactivation de l'étape E_2. Pour l'étape E_1 : $m_{1+} = m_1 \overline{r_1} + s_1 = 1$; donc s=1 correspond au passage de 114 à 118. Pour l'étape E_2: $m_{2+} = m_2 \overline{r_2} + s_2 = 0$ donc $m_2 \overline{r_2} = 0$ (passage de 117 à 122) et $S_2 = 0$ (passage de 114 à 115).

Les résultats sont donc corrects.

IV.5 - Conclusion

Les première et deuxièmes expériences de simulation montrent la possibilité d'organiser un système de traitement qui travaille soit en mode pipe-line, soit en mode parallèle ou mixte, sans aucun changement de programme de gestion. Quel que soit le problème à résoudre, l'utilisateur peut toujours trouver une procédure optimale pour le contrôle d'un processus de calcul en fonction de critères :

- soit de vitesse de traitement
- soit de disponibilité
- soit du taux d'utilisation des processeurs.

La troisième expérience de simulation montre que les codes du microprogramme ont réalisé effectivement les fonctions décrites par le langage de description du chapitre III. La validation des codes du microprogramme n'est pas obligatoire s'ils proviennent d'un microcompilateur ou d'un microassembleur.

CONCLUSION GÉNÉRALE

Dans ce mémoire nous avons proposé une réflexion sur une architecture multiprocesseur - multitraitement et sur le principe de contrôle en modes pipe-line, parallèle et mixte au gré de l'utilisateur. Nous avons fait l'étude de l'unité de contrôle et recommandé une méthode de réalisation et d'implantation.

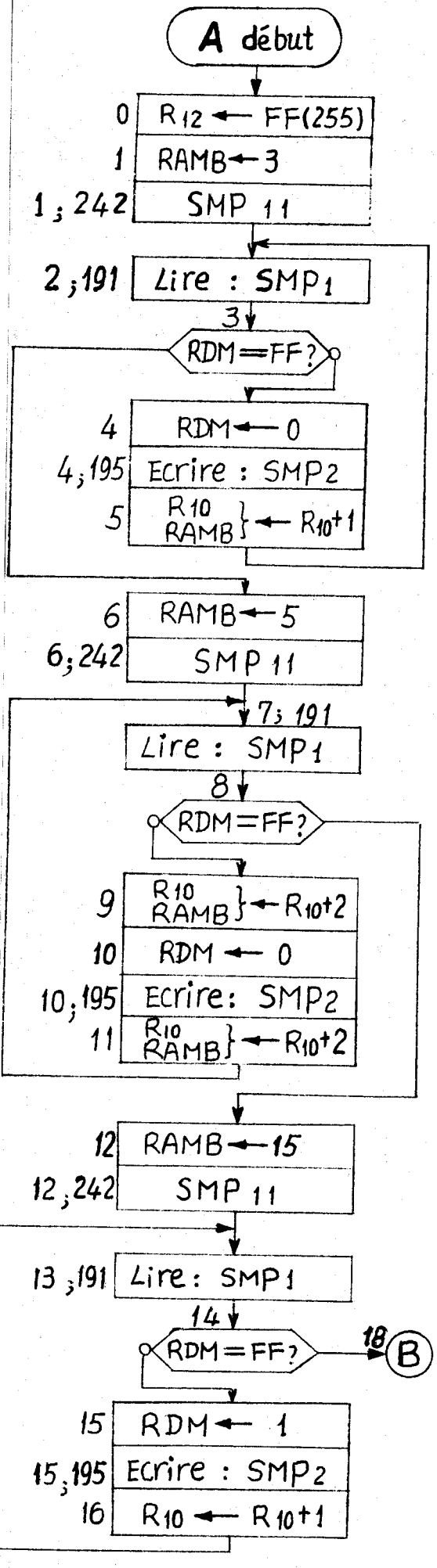
Des expériences de simulation ont été menées au niveau système et au niveau du microcode et ont montré les performances possibles d'un tel système.

D'une part la souplesse du contrôle rend le système utilisable pour une grande diversité de problèmes. Ces sous-tâches à traiter peuvent être dépendantes et/ou indépendantes, et organisées suivant n'importe quel mode. L'utilisateur peut déterminer la priorité de chaque sous-tâche et ainsi s'adapter à des impératifs de vitesse.

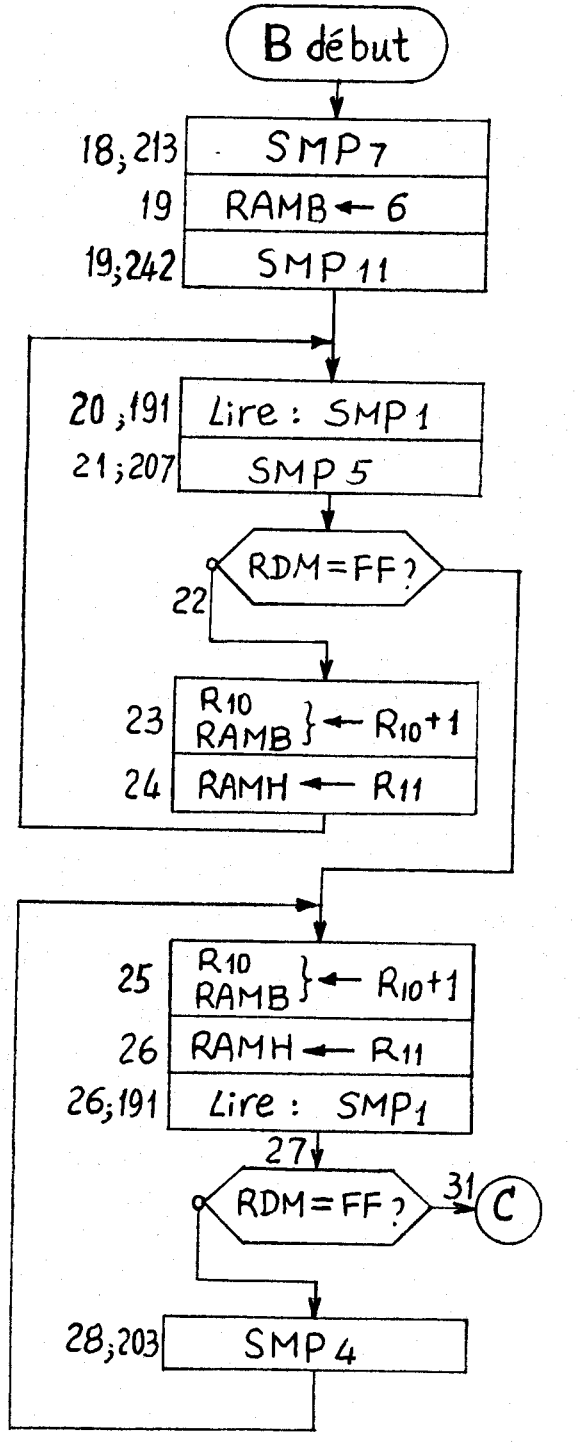
Les nombreux processeurs banalisés fournissent une grande disponibilité, importante pour de nombreux processus industriels.

Le travail que nous avons fait, est un commencement. Pour réaliser un tel système, il y a encore fort à faire. Un point important est le développement d'un langage évolué. Le programme de gestion pourrait encore être simplifié et la possibilité de traitement asynchrone envisagée. La stratégie d'allocation des processeurs pourrait être améliorée de façon à accélérer le travail.

Notre stage à Lille a été trop court pour parfaire cette étude, toutefois l'intérêt de ces premiers résultats est pour nous une incitation à poursuivre ces travaux.

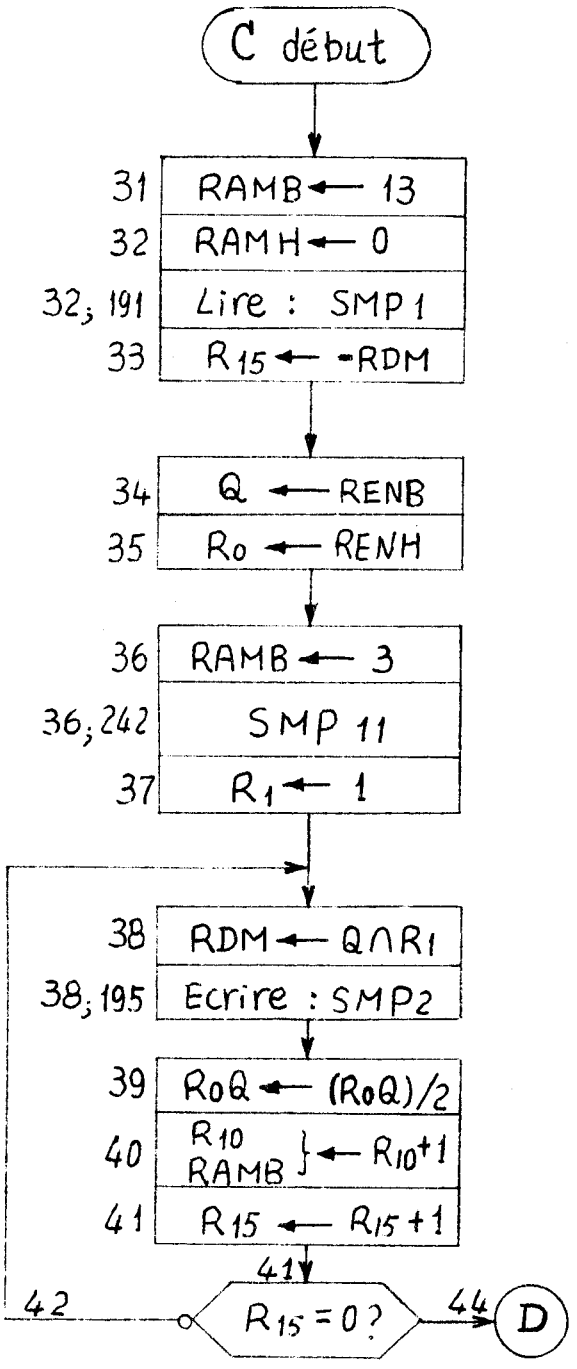


a) Début

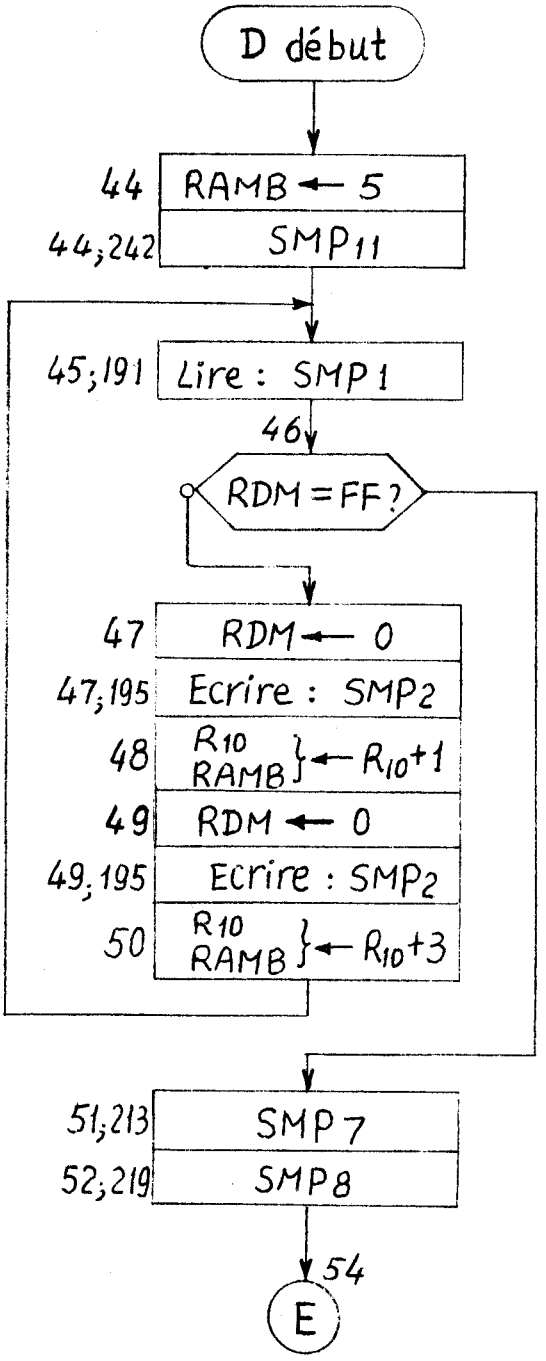


b) Initialisation

Figure A-1

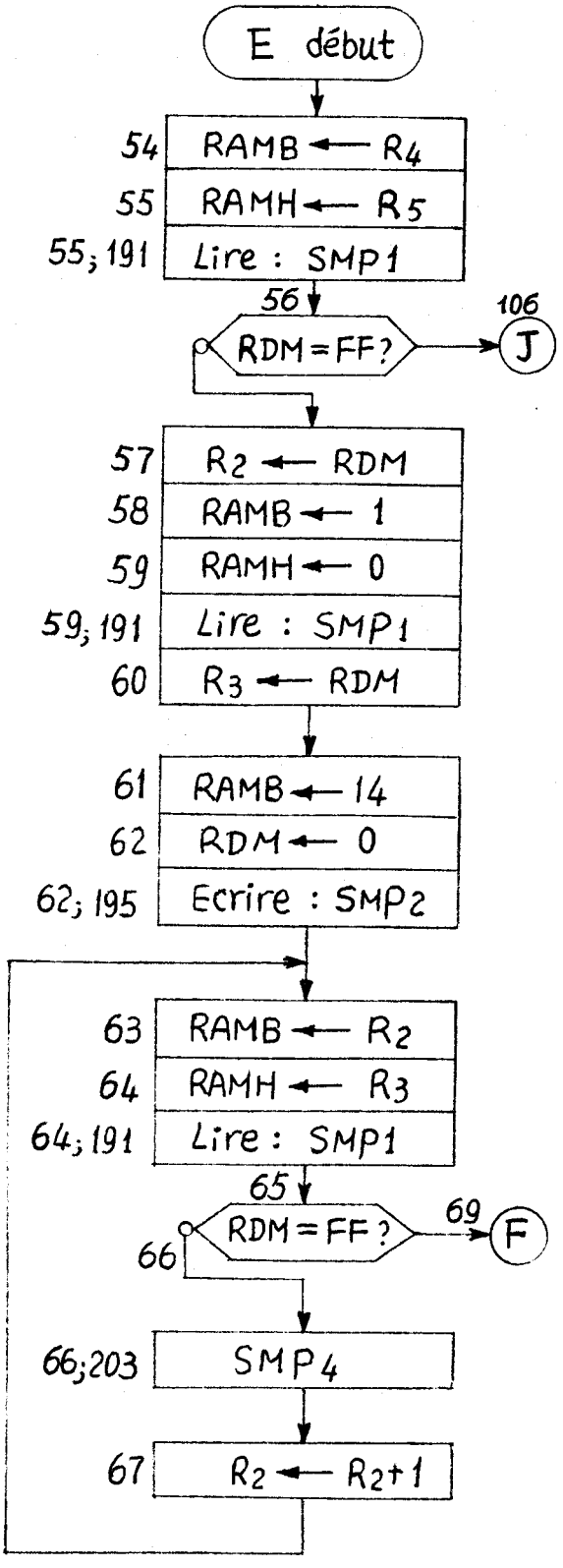


a) Acquisition des entrées

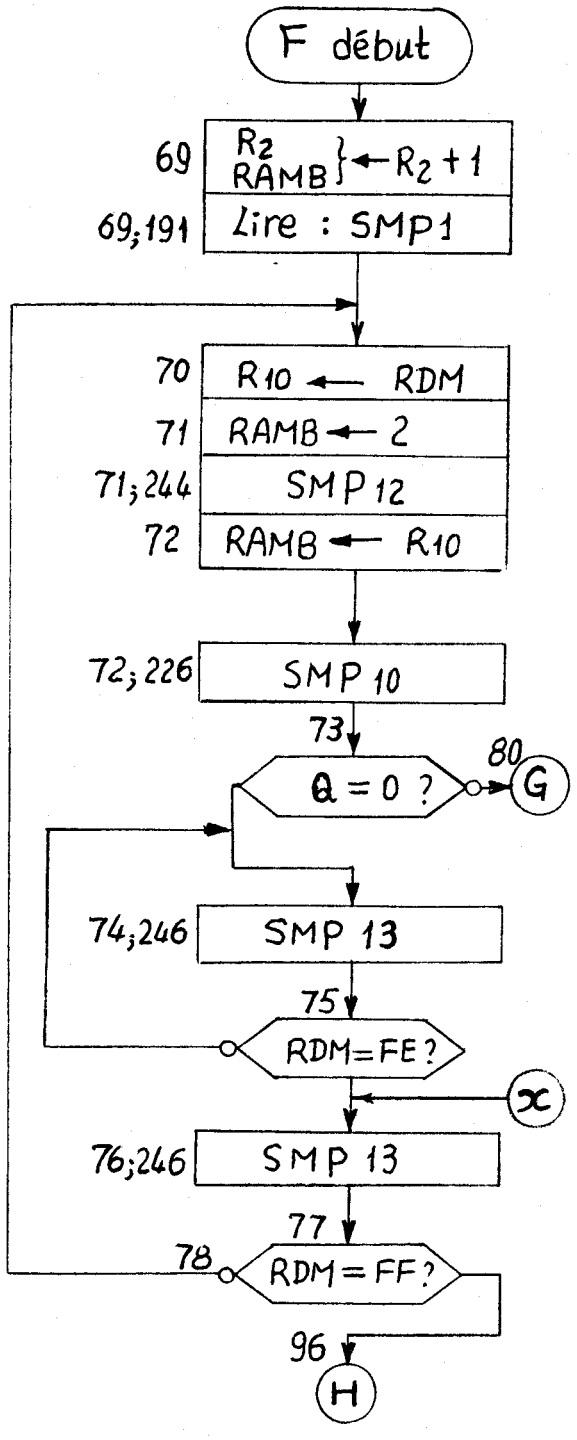


b) Mise à zéro des r_i, s_i et initialisation PLEAC, PLEAS

Figure A-2

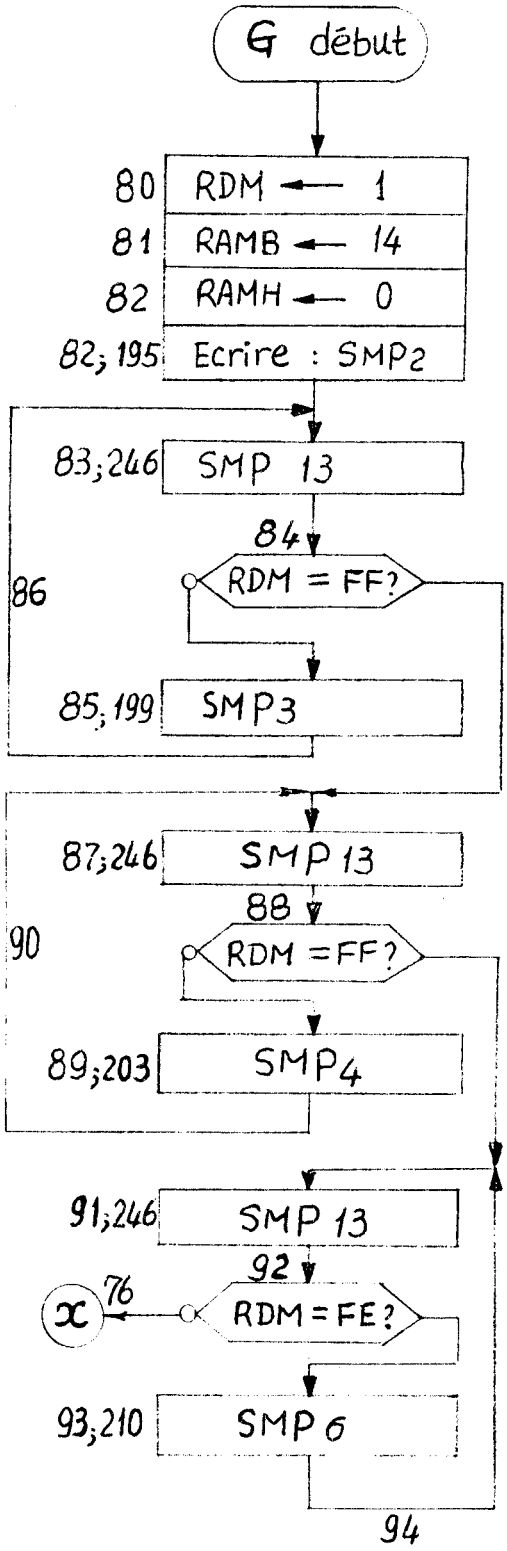


a) Prise du contenu de LEAC INF←0 et Combinatoire local

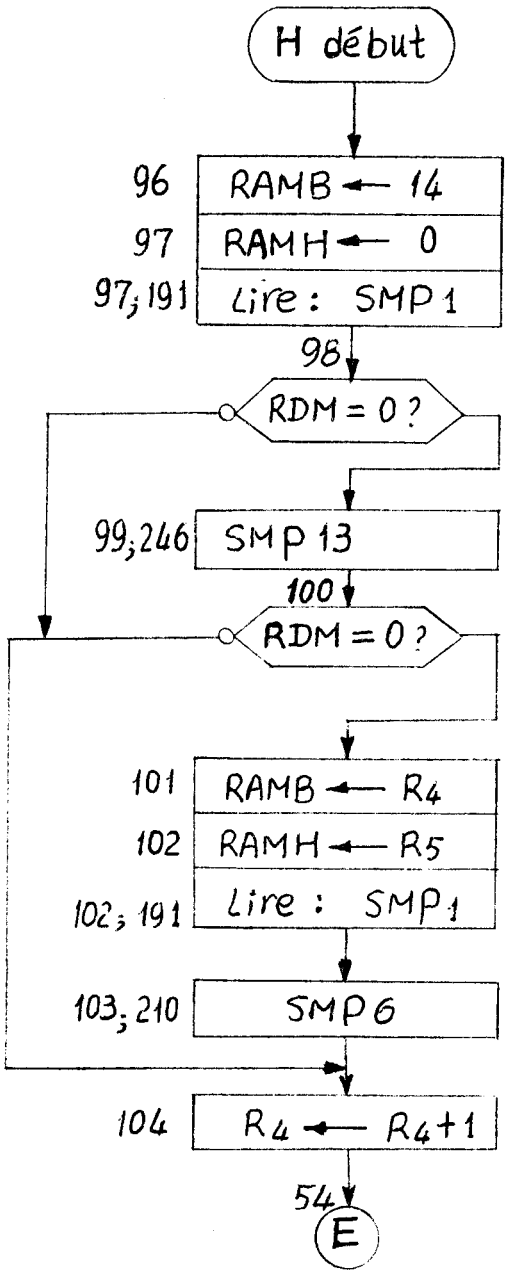


b) Calcul condition d'évolution

Figure A-3

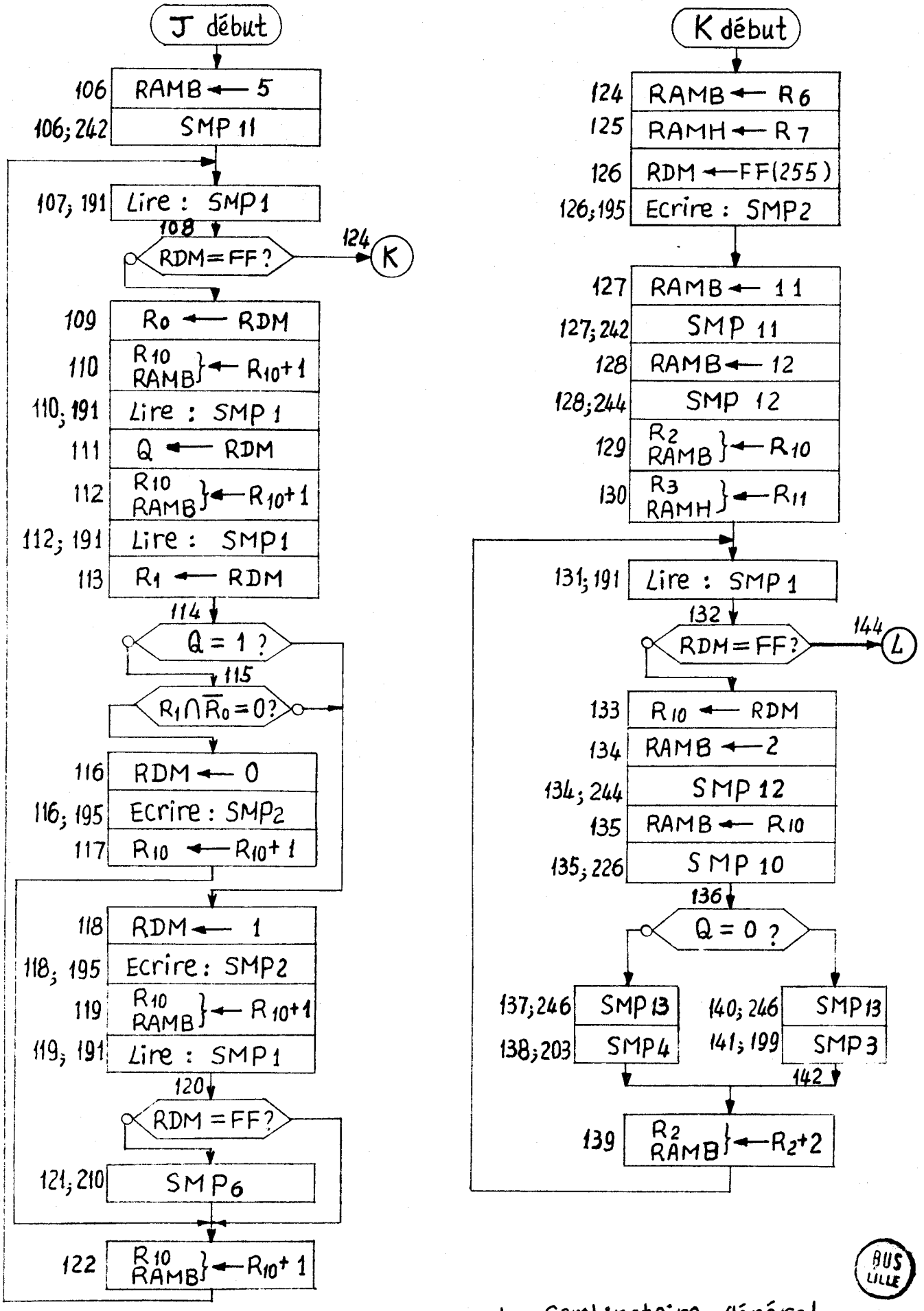


a) INF ← 1 et Actions impulsionnelles



b) Traitement d'étape de Synchronisme

Figure A-4



a) Calcul nouveaux m_i

b) Combinatoire général

Figure A-5

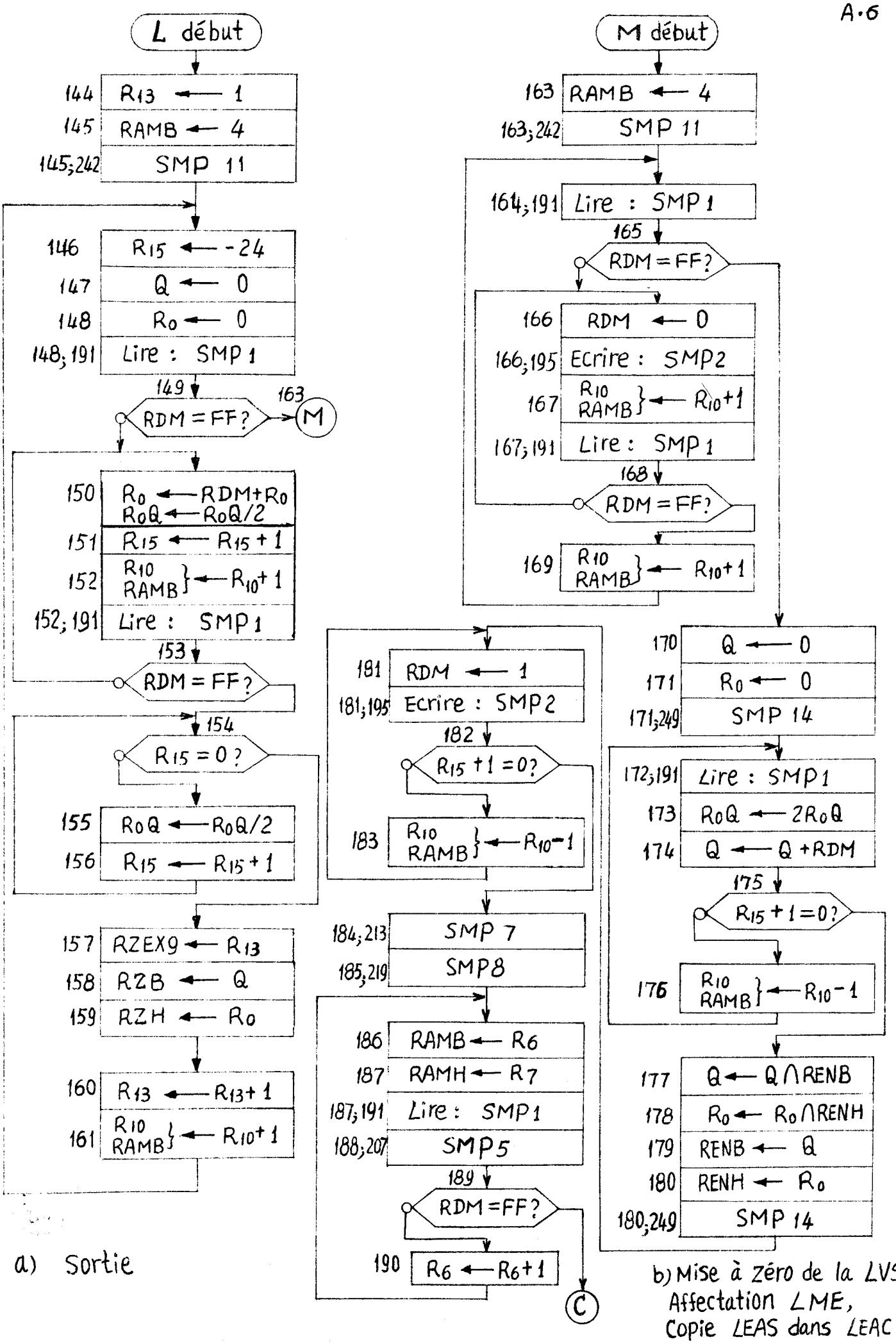

L début
144 R13 ← 1
145 RAMB ← 4
145;242 SMP 11
146 R15 ← -24
147 Q ← 0
148 R0 ← 0
148;191 Lire : SMP 1
149
RDM = FF ?
163
M
150 R0 ← RDM+R0
R0Q ← R0Q/2
151 R15 ← R15 + 1
152 R10 RAMB ← R10+1
152;191 Lire : SMP 1
153
RDM = FF ?
154
R15 = 0 ?
155 R0Q ← R0Q/2
156 R15 ← R15 + 1
157 RZEX9 ← R13
158 RZB ← Q
159 RZH ← R0
160 R13 ← R13+1
161 R10 RAMB ← R10+1
a) Sortie
M début
163 RAMB ← 4
163;242 SMP 11
164;191 Lire : SMP 1
165
RDM = FF ?
166 RDM ← 0
166;195 Ecrire : SMP2
167 R10 RAMB ← R10+1
167;191 Lire : SMP 1
168
RDM = FF ?
169 R10 RAMB ← R10+1
170 Q ← 0
171 R0 ← 0
171;249 SMP 14
172;191 Lire : SMP 1
173 R0Q ← 2R0Q
174 Q ← Q +RDM
175
R15 + 1 = 0 ?
176 R10 RAMB ← R10 - 1
177 Q ← Q ∩ RENB
178 R0 ← R0 ∩ RENH
179 RENB ← Q
180 RENH ← R0
180;249 SMP 14
181 RDM ← 1
181;195 Ecrire : SMP2
182
R15 + 1 = 0 ?
183 R10 RAMB ← R10 - 1
184;213 SMP 7
185;219 SMP8
186 RAMB ← R6
187 RAMH ← R7
187;191 Lire : SMP 1
188;207 SMP 5
189
RDM = FF ?
190 R6 ← R6+1
C
b) Mise à zéro de la LVS,
Affectation LME,
Copie LEAS dans LEAC.

Figure A-6

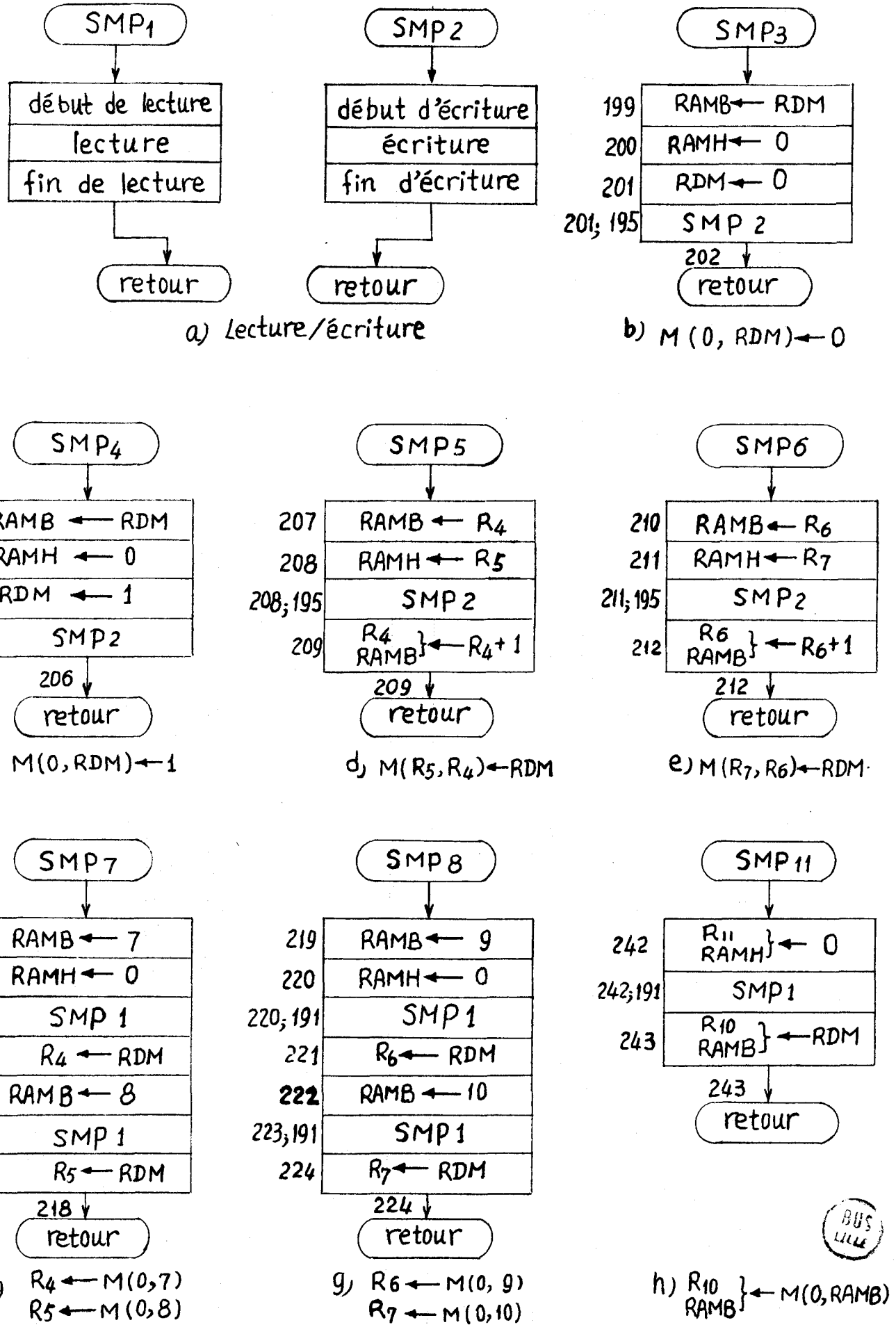
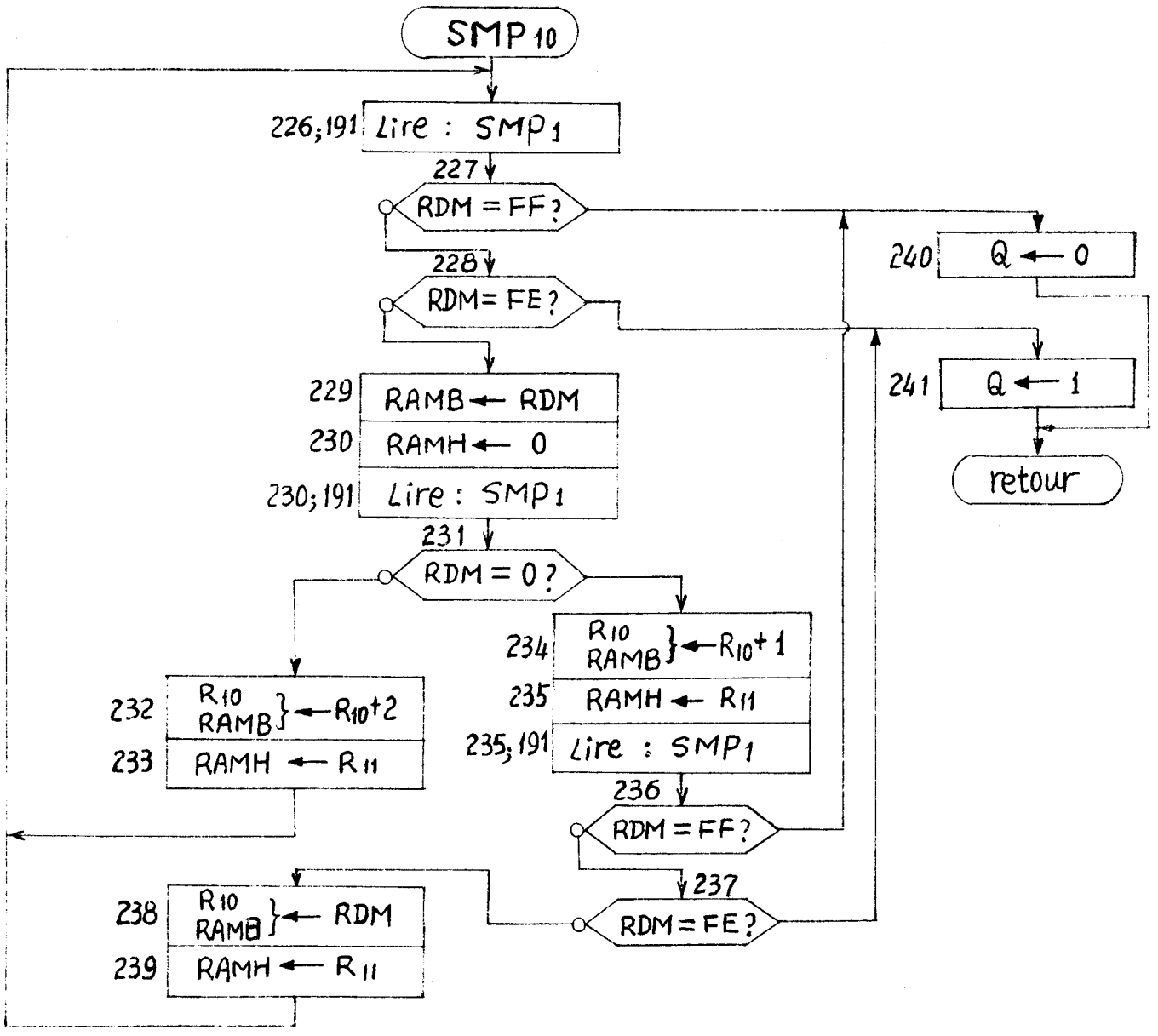
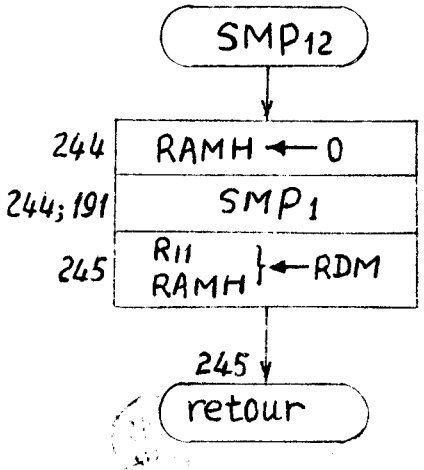

SMP1
début de lecture
lecture
fin de lecture
retour
SMP2
début d'écriture
écriture
fin d'écriture
retour
a) Lecture/écriture
SMP3
199 RAMB ← RDM
200 RAMH ← 0
201 RDM ← 0
201; 195 SMP 2
202
retour
b) M(0, RDM) ← 0
SMP4
203 RAMB ← RDM
204 RAMH ← 0
205 RDM ← 1
205; 195 SMP2
206
retour
c) M(0, RDM) ← 1
SMP5
207 RAMB ← R4
208 RAMH ← R5
208; 195 SMP 2
209 R4 RAMB } ← R4 + 1
209
retour
d) M(R5, R4) ← RDM
SMP6
210 RAMB ← R6
211 RAMH ← R7
211; 195 SMP2
212 R6 RAMB } ← R6+1
212
retour
e) M(R7, R6) ← RDM
SMP7
213 RAMB ← 7
214 RAMH ← 0
214; 191 SMP 1
215 R4 ← RDM
216 RAMB ← 8
217; 195 SMP 1
218 R5 ← RDM
218
retour
f) R4 ← M(0,7)
R5 ← M(0,8)
SMP8
219 RAMB ← 9
220 RAMH ← 0
220; 191 SMP 1
221 R6 ← RDM
222 RAMB ← 10
223; 191 SMP 1
224 R7 ← RDM
224
retour
g) R6 ← M(0, 9)
R7 ← M(0,10)
SMP11
242 R11 RAMH } ← 0
242; 191 SMP 1
243 R10 RAMB } ← RDM
243
retour
h) R10 RAMB } ← M(0, RAMB)

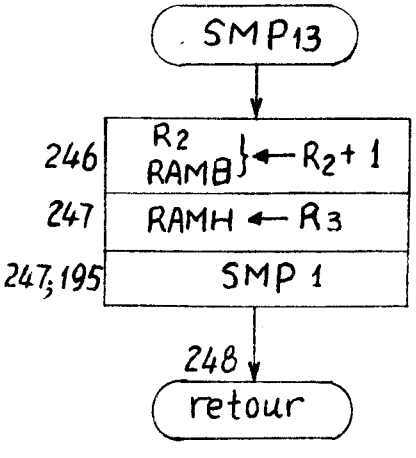
Figure A-7



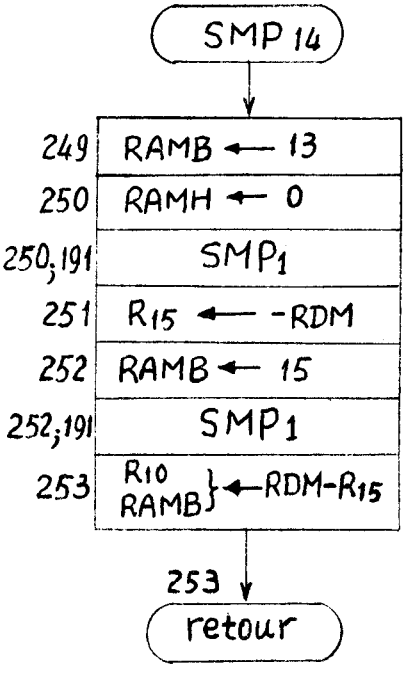
a) Calcul des conditions d'évolution ou du combinatoire général



b) R11 RAMH ← M(0, RAMB)



c) Prise de l'élément suivant de LEAC



d) Pointer le dernier élément de LME

Figure A-8

LE CODE DU MICROPROGRAMME EN DEC.

```
0 __ 0  1  0  2  3  7  0  0  12  255 ;
1 __ 4  1  242  5  1  7  0  0  0  3 ;
2 __ 0  0  191  5  0  0  0  0  0  0 ;
3 __ 0  2  6  12  1  5  2  12  0  0 ;
4 __ 3  0  195  5  1  4  4  0  0  0 ;
5 __ 4  0  2  1  3  4  8  10  10  0 ;
6 __ 4  1  242  5  1  7  0  0  0  5 ;
7 __ 0  0  191  5  0  0  0  0  0  0 ;
8 __ 0  2  12  12  1  5  2  12  0  0 ;
9 __ 4  1  0  2  3  5  0  10  10  2 ;
10 __ 3  0  195  5  1  4  4  0  0  0 ;
11 __ 4  1  7  1  3  5  0  10  10  2 ;
12 __ 4  1  242  5  1  7  0  0  0  15 ;
13 __ 0  0  191  5  0  0  0  0  0  0 ;
14 __ 0  2  18  12  1  5  2  12  0  0 ;
15 __ 3  1  195  5  1  7  0  0  0  1 ;
16 __ 4  0  13  1  3  4  8  10  10  0 ;
17 __ 0  0  0  0  0  0  0  0  0  0 ;
18 __ 0  0  213  5  0  0  0  0  0  0 ;
19 __ 4  1  242  5  1  7  0  0  0  6 ;
20 __ 0  0  191  5  0  0  0  0  0  0 ;
21 __ 0  0  207  5  0  0  0  0  0  0 ;
22 __ 0  2  25  12  1  5  2  12  0  0 ;
23 __ 4  0  0  2  3  4  8  10  10  0 ;
24 __ 5  0  20  1  1  4  0  11  0  0 ;
25 __ 4  0  0  2  3  4  8  10  10  0 ;
26 __ 5  0  191  5  1  4  0  11  0  0 ;
27 __ 0  2  31  12  1  5  2  12  0  0 ;
28 __ 0  0  203  5  0  0  0  0  0  0 ;
29 __ 0  0  25  1  0  0  0  0  0  0 ;
30 __ 0  0  0  0  0  0  0  0  0  0 ;
31 __ 4  1  0  2  1  7  0  0  0  13 ;
32 __ 5  0  191  5  1  4  4  0  0  0 ;
33 __ 0  2  0  2  3  7  1  0  15  0 ;
34 __ 0  3  0  2  0  7  3  0  0  0 ;
35 __ 0  4  0  2  3  7  3  0  0  0 ;
36 __ 4  1  242  5  1  7  0  0  0  3 ;
37 __ 0  1  0  2  3  7  0  0  1  1 ;
38 __ 3  0  195  5  1  0  4  1  0  0 ;
39 __ 0  0  0  2  12  11  0  0  0  0 ;
40 __ 4  0  0  2  3  4  8  10  10  0 ;
41 __ 0  0  44  12  3  4  8  15  15  0 ;
42 __ 0  0  38  1  1  0  0  0  0  0 ;
43 __ 0  0  0  0  0  0  0  0  0  0 ;
44 __ 4  1  242  5  1  7  0  0  0  5 ;
45 __ 0  0  191  5  0  0  0  0  0  0 ;
46 __ 0  2  51  12  1  5  2  12  0  0 ;
47 __ 3  0  195  5  1  4  4  0  0  0 ;
48 __ 4  0  0  2  3  4  8  10  10  0 ;
49 __ 3  0  195  5  1  4  4  0  0  0 ;
50 __ 4  1  45  1  3  5  0  10  10  3 ;
51 __ 0  0  213  5  0  0  0  0  0  0 ;
52 __ 0  0  219  5  0  0  0  0  0  0 ;
53 __ 0  0  0  2  0  0  0  0  0  0 ;
54 __ 4  0  0  2  1  4  0  4  0  0 ;
55 __ 5  0  191  5  1  4  0  5  0  0 ;
56 __ 0  2  106  12  1  5  2  12  0  0 ;
57 __ 0  2  0  2  3  7  0  0  2  0 ;
58 __ 4  1  0  2  1  7  0  0  0  1 ;
59 __ 5  0  191  5  1  4  4  0  0  0 ;
```

```
60 __ 0  2  0  2  3  7  0  0  3  0 ;
61 __ 4  1  0  2  1  7  0  0  0  14 ;
62 __ 3  0  195  5  1  4  4  0  0  0 ;
63 __ 4  0  0  2  1  4  0  2  0  0 ;
64 __ 5  0  191  5  1  4  0  3  0  0 ;
65 __ 0  2  69  12  1  5  2  12  0  0 ;
66 __ 0  0  203  5  0  0  0  0  0  0 ;
67 __ 0  0  63  1  3  4  8  2  2  0 ;
68 __ 0  0  0  0  0  0  0  0  0  0 ;
69 __ 0  0  246  5  0  0  0  0  0  0 ;
70 __ 0  2  0  2  3  7  0  0  10  0 ;
71 __ 4  1  244  5  1  7  0  0  0  2 ;
72 __ 4  0  226  5  1  4  0  10  0  0 ;
73 __ 0  0  80  0  1  2  0  0  0  0 ;
74 __ 0  0  246  5  0  0  0  0  0  0 ;
75 __ 0  2  74  0  1  5  10  12  0  0 ;
76 __ 0  0  246  5  0  0  0  0  0  0 ;
77 __ 0  2  96  12  1  5  2  12  0  0 ;
78 __ 0  0  70  1  0  0  0  0  0  0 ;
79 __ 0  0  0  0  0  0  0  0  0  0 ;
80 __ 3  1  0  2  1  7  0  0  0  1 ;
81 __ 4  1  0  2  1  7  0  0  0  14 ;
82 __ 5  1  195  5  1  7  0  0  0  0 ;
83 __ 0  0  246  5  0  0  0  0  0  0 ;
84 __ 0  2  87  12  1  5  2  12  0  0 ;
85 __ 0  0  199  5  0  0  0  0  0  0 ;
86 __ 0  0  83  1  0  0  0  0  0  0 ;
87 __ 0  0  246  5  0  0  0  0  0  0 ;
88 __ 0  2  91  12  1  5  2  12  0  0 ;
89 __ 0  0  203  5  0  0  0  0  0  0 ;
90 __ 0  0  87  1  0  0  0  0  0  0 ;
91 __ 0  0  246  5  0  0  0  0  0  0 ;
92 __ 0  2  76  12  1  5  10  12  0  0 ;
93 __ 0  0  210  5  0  0  0  0  0  0 ;
94 __ 0  0  91  1  0  0  0  0  0  0 ;
95 __ 0  0  0  0  0  0  0  0  0  0 ;
96 __ 4  1  0  2  1  7  0  0  0  14 ;
97 __ 5  0  191  5  1  4  4  0  0  0 ;
98 __ 0  2  104  0  1  7  0  0  0  0 ;
99 __ 0  0  246  5  0  0  0  0  0  0 ;
100 __ 0  2  114  0  1  7  0  0  0  0 ;
101 __ 4  0  0  2  1  4  0  4  0  0 ;
102 __ 5  0  191  5  1  4  0  5  0  0 ;
103 __ 0  0  210  5  0  0  0  0  0  0 ;
104 __ 4  0  54  1  3  4  8  4  4  0 ;
105 __ 0  0  0  0  0  0  0  0  0  0 ;
106 __ 4  1  242  5  1  7  0  0  0  5 ;
107 __ 5  0  191  5  1  4  4  0  0  0 ;
108 __ 0  2  124  12  1  5  2  12  0  0 ;
109 __ 0  2  0  2  3  7  0  0  0  0 ;
110 __ 4  0  191  5  3  4  8  10  10  0 ;
111 __ 0  2  0  2  0  7  0  0  0  0 ;
112 __ 4  0  191  5  3  4  8  10  10  0 ;
113 __ 0  2  0  2  3  7  0  0  1  0 ;
114 __ 0  0  118  0  1  2  0  0  0  0 ;
115 __ 0  0  118  0  1  1  5  0  1  0 ;
116 __ 3  0  195  5  1  4  4  0  0  0 ;
117 __ 4  0  122  1  3  4  8  10  10  0 ;
118 __ 3  1  195  5  1  7  0  0  0  1 ;
119 __ 4  0  191  5  3  4  8  10  10  0 ;
120 __ 0  2  122  12  1  5  2  12  0  0 ;
```

```
121 __ 0  0  210  5  1  0  0  0  0  0 ;
122 __ 4  0  107  1  3  4  8  10  10  0 ;
123 __ 0  0  0  0  0  0  0  0  0  0 ;
124 __ 4  0  0  2  1  4  0  6  0  0 ;
125 __ 5  0  0  2  1  4  0  7  0  0 ;
126 __ 3  0  195  5  1  4  0  12  0  0 ;
127 __ 4  1  242  5  1  7  0  0  0  11 ;
128 __ 4  1  244  5  1  7  0  0  0  12 ;
129 __ 4  0  0  2  3  4  0  10  2  0 ;
130 __ 5  0  0  2  3  4  0  11  3  0 ;
131 __ 5  0  191  5  1  4  0  3  0  0 ;
132 __ 0  2  144  12  1  5  2  12  0  0 ;
133 __ 0  2  0  2  3  7  0  0  10  0 ;
134 __ 4  1  244  5  1  7  0  0  0  2 ;
135 __ 4  0  226  5  1  4  0  10  0  0 ;
136 __ 0  0  140  12  1  2  0  0  0  0 ;
137 __ 0  0  246  5  0  0  0  0  0  0 ;
138 __ 0  0  203  5  0  0  0  0  0  0 ;
139 __ 4  1  131  1  3  5  0  2  2  2 ;
140 __ 0  0  246  5  0  0  0  0  0  0 ;
141 __ 0  0  199  5  0  0  0  0  0  0 ;
142 __ 5  0  139  1  1  4  0  3  0  0 ;
143 __ 0  0  0  0  0  0  0  0  0  0 ;
144 __ 0  1  0  2  3  7  0  0  13  1 ;
145 __ 4  1  242  5  1  7  0  0  0  4 ;
146 __ 0  1  0  2  3  7  1  0  15  24 ;
147 __ 0  0  0  2  0  4  4  0  0  0 ;
148 __ 0  0  191  5  3  4  4  0  0  0 ;
149 __ 0  2  163  12  1  5  2  12  0  0 ;
150 __ 0  2  0  2  12  5  0  0  0  0 ;
151 __ 0  0  0  2  3  4  8  15  15  0 ;
152 __ 4  0  191  5  3  4  8  10  10  0 ;
153 __ 0  2  150  0  1  5  2  12  0  0 ;
154 __ 0  0  157  12  1  4  0  15  0  0 ;
155 __ 0  0  0  2  12  4  0  0  0  0 ;
156 __ 0  0  154  1  3  4  8  15  15  0 ;
157 __ 9  0  0  2  1  4  0  13  0  0 ;
158 __ 1  0  0  2  1  2  0  0  0  0 ;
159 __ 2  0  0  2  1  4  0  0  0  0 ;
160 __ 0  0  0  2  3  4  8  13  13  0 ;
161 __ 4  0  146  1  3  4  8  10  10  0 ;
162 __ 0  0  0  0  0  0  0  0  0  0 ;
163 __ 4  1  242  5  1  7  0  0  0  4 ;
164 __ 0  0  191  5  0  0  0  0  0  0 ;
165 __ 0  2  170  12  1  5  2  12  0  0 ;
166 __ 3  0  195  5  1  4  4  0  0  0 ;
167 __ 4  0  191  5  3  4  8  10  10  0 ;
168 __ 0  2  166  0  1  5  2  12  0  0 ;
169 __ 4  0  164  1  3  4  8  10  10  0 ;
170 __ 0  0  0  2  0  4  4  0  0  0 ;
171 __ 0  0  249  5  3  4  4  0  0  0 ;
172 __ 0  0  191  5  1  0  0  0  0  0 ;
173 __ 0  0  0  2  14  12  0  0  0  0 ;
174 __ 0  2  0  2  0  6  0  0  0  0 ;
175 __ 0  0  177  12  3  4  8  15  15  0 ;
176 __ 4  1  172  1  3  5  1  10  10  1 ;
177 __ 0  3  0  2  0  6  4  0  0  0 ;
178 __ 0  4  0  2  3  5  4  0  0  0 ;
179 __ 7  0  0  2  1  2  0  0  0  0 ;
180 __ 8  0  249  5  1  4  0  0  0  0 ;
181 __ 3  1  195  5  1  7  0  0  0  1 ;
```

```
182 __ 0  0   184  12  3  4  8  15  15  0 ;
183 __ 4  1   181  1  3  5  1  10  10  1 ;
184 __ 0  0   213  5  0  0  0  0  0  0 ;
185 __ 0  0   219  5  0  0  0  0  0  0 ;
186 __ 4  0   0  2  1  4  0  6  0  0 ;
187 __ 5  0   191  5  1  4  0  7  0  0 ;
188 __ 0  0   207  5  0  0  0  0  0  0 ;
189 __ 0  2   31  12  1  5  1  12  0  0 ;
190 __ 4  0   186  1  3  4  8  6  6  0 ;
191 __ 0  7   195  1  1  0  0  0  0  0 ;
192 __ 0  0   0  0  0  0  0  0  0  0 ;
193 __ 0  0   0  0  0  0  0  0  0  0 ;
194 __ 0  0   0  0  0  0  0  0  0  0 ;
195 __ 0  5   198  1  1  0  0  0  0  0 ;
196 __ 0  0   0  0  0  0  0  0  0  0 ;
197 __ 0  0   0  0  0  0  0  0  0  0 ;
198 __ 0  6   0  6  1  0  0  0  0  0 ;
199 __ 4  2   0  2  1  7  0  0  0  0 ;
200 __ 5  1   0  2  1  7  0  0  0  0 ;
201 __ 3  0   195  5  1  4  4  0  0  0 ;
202 __ 0  0   0  6  0  0  0  0  0  0 ;
203 __ 4  2   0  2  1  7  0  0  0  0 ;
204 __ 5  1   0  2  1  7  0  0  0  0 ;
205 __ 3  1   195  5  1  7  0  0  0  1 ;
206 __ 0  0   0  6  0  0  0  0  0  0 ;
207 __ 4  0   0  2  1  4  0  4  0  0 ;
208 __ 5  0   195  5  1  4  0  5  0  0 ;
209 __ 4  0   0  6  3  4  8  4  4  0 ;
210 __ 4  0   0  2  1  4  0  6  0  0 ;
211 __ 5  0   195  5  1  4  0  7  0  0 ;
212 __ 4  0   0  6  3  4  8  6  6  0 ;
213 __ 4  1   0  2  1  7  0  0  0  7 ;
214 __ 5  0   191  5  1  4  4  0  0  0 ;
215 __ 0  2   0  2  3  7  0  0  4  0 ;
216 __ 4  1   0  2  1  7  0  0  0  8 ;
217 __ 0  0   191  5  0  0  0  0  0  0 ;
218 __ 0  2   0  6  3  7  0  0  5  0 ;
219 __ 4  1   0  2  1  7  0  0  0  9 ;
220 __ 5  0   191  5  1  4  4  0  0  0 ;
221 __ 0  2   0  2  3  7  0  0  6  0 ;
222 __ 4  1   0  2  1  7  0  0  0  10 ;
223 __ 0  0   191  5  0  0  0  0  0  0 ;
224 __ 0  2   0  6  3  7  0  0  7  0 ;
225 __ 0  0   0  0  0  0  0  0  0  0 ;
226 __ 0  0   191  5  1  0  0  0  0  0 ;
227 __ 0  2   240  12  1  5  2  12  0  0 ;
228 __ 0  2   241  12  1  5  10  12  0  0 ;
229 __ 4  2   0  2  1  7  0  0  0  0 ;
230 __ 5  0   191  5  1  4  4  0  0  0 ;
231 __ 0  2   234  12  1  7  0  0  0  0 ;
232 __ 4  1   0  2  3  5  0  10  10  2 ;
233 __ 5  0   226  1  1  4  0  11  0  0 ;
234 __ 4  0   0  2  3  4  8  10  10  0 ;
235 __ 5  0   191  5  1  4  0  11  0  0 ;
236 __ 0  2   240  12  1  5  2  12  0  0 ;
237 __ 0  2   241  12  1  5  10  12  0  0 ;
238 __ 4  2   0  2  1  7  0  0  0  0 ;
239 __ 5  0   226  1  1  4  0  11  0  0 ;
240 __ 0  0   0  6  0  4  4  0  0  0 ;
241 __ 0  1   0  6  0  7  0  0  0  1 ;
242 __ 5  0   191  5  3  4  4  0  11  0 ;
```

```
243 __  4   2   0   6   3   7   0   0   10   0 ;
244 __  5   0  191   5   1   4   4   0   0   0 ;
245 __  5   2   0   6   3   7   0   0   11   0 ;
246 __  4   0   0   2   3   4   8   2   2   0 ;
247 __  5   0  191   5   1   4   0   3   0   0 ;
248 __  0   0   0   6   0   0   0   0   0   0 ;
249 __  4   1   0   2   1   7   0   0   0   13 ;
250 __  5   0  191   5   1   4   4   0   0   0 ;
251 __  0   2   0   2   3   7   1   0   15   0 ;
252 __  4   1  191   5   1   7   0   0   0   15 ;
253 __  4   2   0   6   3   5   2   15   10   0 ;
254 __  0   0   0   0   0   0   0   0   0   0 ;
255 __  0   0   0   0   0   0   0   0   0   0 ;
```

Figure A·9

LE PROGRAMME DE SIMULATION DU RdP1

```
1 REM__PROGRAMME DE GESTION_____SIMUL
3 DIM Q1(16),Q2(16),Q3(16),Q4(16),P(16),YO(16)
5 OPEN "SY1:MFD" AS FILE #1
6 OPEN "LP:" FOR OUTPUT AS FILE #9
7 DIM #1,M(3,256)
9 FOR I=0 TO 15 \ P(I)=0 \ NEXT I
21 REM__MISE A ZERO LES ENTREES ET V.INT.
23 S2=255 \ S0=M(0,3) \ S1=0
29 D=M(S1,S0) \ IF D=S2 THEN 39
33 M(S1,S0)=0 \ S0=S0+1 \ GO TO 29
39 S0=M(0,5)
41 D=M(S1,S0) \ IF D=S2 THEN 53
51 S0=S0+2 \ M(S1,S0)=0 \ S0=S0+2 \ GO TO 41
53 S0=M(0,15)
55 D=M(S1,S0) \ IF D=S2 THEN 61
57 M(S1,S0)=1 \ S0=S0+1 \ GO TO 55
60 REM__INITIALISATION DE Eio et Mio
61 GOSUB 421 \ Q1(1)=1 \ Q1(2)=1 \ PRINT #9,"LES ETAPES INIT.:";
65 S0=M(0,6) \ C=0 \ S1=0
67 D=M(S1,S0) \ M(R5,R4)=D \ R4=R4+1
71 PRINT #9,D; \ IF D=S2 THEN 77
73 S0=S0+1 \ GO TO 67
77 S0=S0+1 \ D=M(S1,S0) \ IF D=S2 THEN 85
81 PRINT "adM:";D \ M(0,D)=1 \ GO TO 77
85 PRINT
86 PRINT "SI VOUS VOULEZ COMMENCER,ENTREZ M:" \ INPUT Q1(0)
88 PRINT #9, \ PRINT #9,"M=";Q1(0)
90 REM__ACQUISITION D'ENTREE
91 C=C+1 \ PRINT
93 S5=-M(0,13) \ PRINT "DEBUT D'UNE CYCLE";"_____CYCLE Ci=";C
95 FOR I=0 TO 15 \ Q2(I)=Q1(I) \ NEXT I
99 S0=M(0,3) \ S1=0 \ R1=1 \ PRINT "ENTREE:";
103 M(S1,S0)=Q2(0)*R1 \ PRINT M(S1,S0);
107 FOR I=0 TO 15 \ Q2(I)=Q2(I+1) \ NEXT I
113 S0=S0+1 \ S5=S5+1 \ IF S5=0 THEN 121
119 GO TO 103
120 REM__MISE A ZERO LES Ri et Si,INIT. PLEAC,PLEAS
121 S0=M(0,5) \ S1=0
123 D=M(S1,S0) \ IF D=S2 THEN 135
125 M(S1,S0)=0 \ S0=S0+1 \ M(S1,S0)=0 \ S0=S0+3 \ GO TO 123
133 D=M(S1,S0) \ GO TO 123
135 GOSUB 421 \ GOSUB 429 \ PRINT  \ PRINT "LES ETAPES ACT. EN COURS:"
140 REM__TRAITEMENT D'UNE ETAPE
141 D=M(R5,R4) \ IF D=S2 THEN 251
143 PRINT D; \ R2=D \ R3=M(0,1) \ M(0,14)=0
149 D=M(R3,R2) \ IF D=S2 THEN 159
151 S0=D \ M(0,S0)=1 \ R2=R2+1 \ GO TO 149
159 R2=R2+1
161 S0=M(R3,R2) \ PRINT "adSP:";S0
170 REM__CALCUL DE CONDITION D,EVOLUTION
171 S1=M(0,2) \ GOSUB 437 \ PRINT "EVt:";Q \ IF Q<>0 THEN 191
177 R2=R2+1 \ IF M(R3,R2)<>S2-1 THEN 177
181 R2=R2+1 \ IF M(R3,R2)=S2 THEN 231
183 GO TO 161
190 REM__ACTIONS ASSOCIEES A TRANSITION
191 M(0,14)=1 \ PRINT
195 R2=R2+1 \ D=M(R3,R2) \ IF D=S2 THEN 205
201 M(0,D)=0 \ GO TO 195
205 R2=R2+1 \ D=M(R3,R2) \ IF D=S2 THEN 215
211 M(0,D)=1 \ GO TO 205
215 R2=R2+1 \ D=M(R3,R2) \ IF D=S2-1 THEN 181
```

```
220 M(R7,R6)=D \ R6=R6+1 \ GO TO 215
230 REM__TRAITEMENT D'ETAPE CLE SYNCHRONE
231 IF M(0,14)<>0 THEN 241
235 R2=R2+1 \ IF M(R3,R2)<>0 THEN 241
237 D=M(R5,R4)
238 M(R7,R6)=D \ R6=R6+1
241 R4=R4+1 \ GO TO 141
250 REM__CALCUL DE Mi
251 S1=0 \ S0=M(0,5)
255 IF M(S1,S0)=S2 THEN 301
257 R0=M(S1,S0) \ S0=S0+1 \ Q=M(S1,S0) \ S0=S0+1
259 PRINT "Ri:";R0;"Si:";Q;
265 R1=M(S1,S0) \ PRINT "Mc:";R1; \ IF Q<>0 THEN 281
269 IF R0=0 THEN 273
271 R0=0 \ GO TO 275
273 R0=1
275 IF R1*R0<>0 THEN 281
277 M(S1,S0)=0 \ PRINT M(S1,S0); \ S0=S0+1 \ GO TO 291
281 M(S1,S0)=1 \ PRINT M(S1,S0); \ S0=S0+1 \ D=M(S1,S0)
287 IF D=S2 THEN 291
289 M(R7,R6)=D \ R6=R6+1
291 S0=S0+1 \ GO TO 255
300 REM__CALCUL DE C. G.
301 M(R7,R6)=S2 \ REM_MISE SEPARAT. FF(255)DANSLAES
303 R2=M(0,11) \ R3=M(0,12) \ REM__POINTE A LA LISTE DE C.G.
307 IF M(R3,R2)=S2 THEN 341
309 S0=M(R3,R2) \ S1=M(0,2)
313 GOSUB 437 \ IF Q=0 THEN 327
319 R2=R2+1 \ D=M(R3,R2) \ M(0,D)=1
323 R2=R2+2 \ GO TO 307
327 R2=R2+1 \ D=M(R3,R2) \ M(0,D)=0 \ GO TO 323
340 REM__AFFECTION DES SORTIES
341 S3=1 \ S0=M(0,4) \ S1=0 \ REM_POINTE A LA LISTE SE SORTIE
345 PRINT  \ PRINT "SORTIE:"
347 S5=-24 \ FOR I=0 TO 15 \ Q2(I)=0 \ NEXT I
355 D=M(S1,S0) \ IF D=S2 THEN 391
359 Q2(8)=D \ GOSUB 769 \ PRINT Q2(7);
363 S5=S5+1 \ S0=S0+1 \ D=M(S1,S0) \ IF D<>S2 THEN 359
368 IF S5=0 THEN 375
371 GOSUB 769 \ S5=S5+1 \ GO TO 368
375 Z9=S3 \ FOR I=0 TO 15 \ Q3(I)=Q2(I) \ NEXT I \ GOSUB 473
385 S3=S3+1 \ PRINT  \ S0=S0+1 \ GO TO 347
390 REM__MISE A ZERO LA LISTE DE SORTIE ET COPIE LEAS
391 S0=M(0,4) \ S1=0
393 IF M(S1,S0)=S2 THEN 402
395 M(S1,S0)=0 \ S0=S0+1 \ IF M(S1,S0)<>S2 THEN 395
401 S0=S0+1 \ GO TO 393
402 S5=M(0,13)-1 \ S0=M(0,15)
403 FOR I=0 TO S5 \ Q2(I)=M(0,S0) \ M(0,S0)=1 \ S0=S0+1
404 Q1(I)=Q1(I)*Q2(I) \ NEXT I
405 REM__COPIE LEAS DANS LEAC:
407 GOSUB 421 \ GOSUB 429 \ PRINT
409 PRINT "LES ETAPES ACTIVEES SUIVANTES:"
410 D=M(R7,R6) \ M(R5,R4)=D \ R4=R4+1
414 IF D=S2 THEN 418
416 PRINT D; \ R6=R6+1 \ GO TO 410
418 PRINT  \ PRINT "FIN D'UN CYCLE"
419 GO TO 90
420 REM__SOUSPROGRAMME__SMP7
421 R4=M(0,7) \ R5=M(0,8) \ RETURN
427 REM__SMP8
```

```
429 R6=M(0,9) \ R7=M(0,10) \ RETURN
435 REM__SMP10
437 D=M(S1,S0) \ IF D=S2 THEN 463
439 IF D=S2-1 THEN 467
443 D=M(0,D) \ IF D=0 THEN 451
445 S0=S0+2 \ GO TO 437
451 S0=S0+1 \ D=M(S1,S0) \ IF D=S2 THEN 463
457 IF D=S2-1 THEN 467
459 S0=D \ GO TO 437
463 Q=0
465 RETURN
467 Q=1
469 GO TO 465
471 REM__SOUSPROGRAMME__"EXECUT."
473 ON Z9 GO TO 479,501,477
477 RETURN
479 IF Q3(0)=1 THEN 495
491 IF Q3(1)=1 THEN 497
493 GO TO 477
495 P1=1 \ Q1(1)=0 \ PRINT "P1=1" \ GO TO 491
497 P2=1 \ Q1(2)=0 \ PRINT "P2=1" \ GO TO 477
501 IF Q3(0)=1 THEN  GOSUB 601
503 IF Q3(9)=1 THEN  GOSUB 685
505 IF Q3(8)=1 THEN  GOSUB 675
507 IF Q3(7)=1 THEN  GOSUB 665
509 IF Q3(6)=1 THEN  GOSUB 655
511 IF Q3(5)=1 THEN  GOSUB 645
513 IF Q3(4)=1 THEN  GOSUB 635
515 IF Q3(3)=1 THEN  GOSUB 625
517 IF Q3(2)=1 THEN  GOSUB 615
519 IF Q3(1)=1 THEN  GOSUB 605
521 IF Q3(10)=1 THEN  GOSUB 702
535 FOR I=1 TO 9 \ IF P(I)<>0 THEN 539
537 NEXT I \ GO TO 477
539 RANDOMIZE \ T=INT(RND*10+1)
541 IF T>5 THEN 537
542 PRINT #9, \ PRINT #9,"CYCLE C1="$C,
543 ON I GOSUB 551,553,555,557,559,561,563,565,567
545 ON P(I) GO TO 547,549
547 P(I)=0 \ Q1(1)=1 \ GO TO 537
549 P(I)=0 \ Q1(2)=1 \ GO TO 537
551 Q1(4)=1 \ PRINT #9,"Y2="$Y2 \ RETURN
553 Q1(5)=1 \ PRINT #9,"Y3="$Y3 \ RETURN
555 Q1(6)=1 \ PRINT #9,"Y4="$Y4 \ RETURN
557 Q1(7)=1 \ PRINT #9,"Y5="$Y5 \ RETURN
559 Q1(8)=1 \ PRINT #9,"Y6="$Y6 \ RETURN
561 Q1(9)=1 \ PRINT #9,"Y7="$Y7 \ RETURN
563 Q1(10)=1 \ PRINT #9,"Y8="$Y8 \ RETURN
565 Q1(11)=1 \ PRINT #9,"Y9="$Y9 \ RETURN
567 Q1(12)=1 \ PRINT #9,"Y10="$Y0(11) \ RETURN
601 INPUT X1,X2 \ PRINT #9, \ PRINT #9,"X1="$X1,"X2="$X2
603 Q1(3)=1 \ P(0)=0 \ RETURN
605 IF P1=1 THEN 609
606 IF P2=1 THEN 613
607 RETURN
609 P(1)=1 \ P1=0
611 Y2=SIN(X1) \ GOSUB 705 \ GO TO 607
613 P(1)=2 \ P2=0 \ GO TO 611
615 IF P1=1 THEN 619
616 IF P2=1 THEN 623
617 RETURN
```

```
619 P(2)=1 \ P1=0
621 Y3=COS(X2) \ GOSUB 705 \ GO TO 617
623 P(2)=2 \ P2=0 \ GO TO 621
625 IF P1=1 THEN 629
626 IF P2=1 THEN 633
627 RETURN
629 P(3)=1 \ P1=0
631 Y4=EXP(X1) \ GOSUB 705 \ GO TO 627
633 P(3)=2 \ P2=0 \ GO TO 631
635 IF P1=1 THEN 639
636 IF P2=1 THEN 643
637 RETURN
639 P(4)=1 \ P1=0
641 Y5=Y2^2 \ GOSUB 705 \ GO TO 637
643 P(4)=2 \ P2=0 \ GO TO 641
645 IF P1=1 THEN 649
646 IF P2=1 THEN 653
647 RETURN
649 P(5)=1 \ P1=0
651 Y6=SQR(Y2)+Y3/Y4 \ GOSUB 705 \ GO TO 647
653 P(5)=2 \ P2=0 \ GO TO 651
655 IF P1=1 THEN 659
656 IF P2=1 THEN 663
657 RETURN
659 P(6)=1 \ P1=0
661 Y7=1+EXP(Y4) \ GOSUB 705 \ GO TO 657
663 P(6)=2 \ P2=0 \ GO TO 661
665 IF P1=1 THEN 669
666 IF P2=1 THEN 673
667 RETURN
669 P(7)=1 \ P1=0
671 Y8=Y5*Y6/Y7 \ GOSUB 705 \ GO TO 667
673 P(7)=2 \ P2=0 \ GO TO 671
675 IF P1=1 THEN 679
676 IF P2=1 THEN 683
677 RETURN
679 P(8)=1 \ P1=0
681 Y9=SIN(Y8)+COS(Y8) \ GOSUB 705 \ GO TO 677
683 P(8)=2 \ P2=0 \ GO TO 681
685 IF P1=1 THEN 689
686 IF P2=1 THEN 695
687 RETURN
689 P(9)=1 \ P1=0
691 Y0(0)=1 \ FOR I=0 TO 10 \ Y0(I+1)=Y0(I)*(Y8-1) \ NEXT I
693 GOSUB 705 \ GO TO 687
695 P(9)=2 \ P2=0 \ GO TO 691
702 C=0 \ PRINT #9,
703 PRINT #9,"_____"
704 PRINT #9, \ PRINT #9,"M=";Q1(0) \ GO TO 477
705 PRINT #9, \ PRINT #9,"CYCLE Ci=";C, \ PRINT #9,"OPERAT. PARAL.:";
709 FOR I=0 TO 10 \ PRINT #9,P(I); \ NEXT I \ RETURN
769 L=Q2(0) \ FOR I=0 TO 15 \ Q2(I)=Q2(I+1) \ NEXT I
779 Q2(15)=L \ RETURN
781 CLOSE #1 \ CLOSE #9
783 END
```


Figure A-10

DONNEES DU STRUCTURE DE RdP1

```
PAGE  0 :
LISTE DES CONSTANTES:
 0 - 0 ; 1 - 2 ; 2 - 1 ; 3 - 48 ; 4 - 96 ; 5 - 144 ;
 6 - 20 ; 7 - 2 ; 8 - 1 ; 9 - 17 ; 10 - 1 ; 11 - 32 ;
 12 - 1 ; 13 - 13 ; 14 - 0 ; 15 - 72 ; 16 - 255 ; 17 - 255 ;

LISTE INITIALE:
 20 - 2 ; 21 - 186 ; 22 - 255 ; 23 - 158 ; 24 - 162 ; 25 - 166 ;
 26 - 174 ; 27 - 255 ; 28 - 255 ;
LISTE D'ENTREE:
 48 - 0 ; 49 - 0 ; 50 - 0 ; 51 - 0 ; 52 - 0 ; 53 - 0 ;
 54 - 0 ; 55 - 0 ; 56 - 0 ; 57 - 0 ; 58 - 0 ; 59 - 0 ;
 60 - 0 ; 61 - 255 ; 62 - 255 ;
MASQUE D'ENTREE:
 72 - 1 ; 73 - 1 ; 74 - 1 ; 75 - 1 ; 76 - 1 ; 77 - 1 ;
 78 - 1 ; 79 - 1 ; 80 - 1 ; 81 - 1 ; 82 - 1 ; 83 - 1 ;
 84 - 1 ; 85 - 255 ; 86 - 255 ;
LISTE DE SORTIE:
 96 - 0 ; 97 - 0 ; 98 - 255 ; 99 - 0 ; 100 - 0 ; 101 - 0 ;
 102 - 0 ; 103 - 0 ; 104 - 0 ; 105 - 0 ; 106 - 0 ; 107 - 0 ;
 108 - 0 ; 109 - 0 ; 110 - 255 ; 111 - 0 ; 112 - 0 ; 113 - 0 ;
 114 - 0 ; 115 - 0 ; 116 - 0 ; 117 - 0 ; 118 - 0 ; 119 - 0 ;
 120 - 0 ; 121 - 0 ; 122 - 0 ; 123 - 255 ; 124 - 255 ;
LISTE DE V. INTER.:
 144 - 0 ; 145 - 0 ; 146 - 0 ; 147 - 255 ; 148 - 0 ; 149 - 0 ;
 150 - 0 ; 151 - 255 ; 152 - 0 ; 153 - 0 ; 154 - 0 ; 155 - 255 ;
 156 - 0 ; 157 - 0 ; 158 - 0 ; 159 - 255 ; 160 - 0 ; 161 - 0 ;
 162 - 0 ; 163 - 255 ; 164 - 0 ; 165 - 0 ; 166 - 0 ; 167 - 186 ;
 168 - 0 ; 169 - 0 ; 170 - 0 ; 171 - 207 ; 172 - 0 ; 173 - 0 ;
 174 - 0 ; 175 - 255 ; 176 - 0 ; 177 - 0 ; 178 - 0 ; 179 - 255 ;
 180 - 255 ;
PAGE  1:
LISTE DE C.G. :
 32 - 127 ; 33 - 115 ; 34 - 255 ; 35 - 130 ; 36 - 117 ; 37 - 255 ;
 38 - 133 ; 39 - 119 ; 40 - 255 ; 41 - 255 ; 42 - 255 ; 43 - 255 ;
 44 - 0 ; 45 - 0 ; 46 - 0 ; 47 - 0 ;
LISTE DE CONDITION D'EVOLUT. :
 48 - 48 ; 49 - 255 ; 50 - 254 ; 51 - 0 ; 52 - 0 ; 53 - 51 ;
 54 - 255 ; 55 - 254 ; 56 - 178 ; 57 - 255 ; 58 - 254 ; 59 - 0 ;
 60 - 0 ; 61 - 178 ; 62 - 255 ; 63 - 52 ; 64 - 255 ; 65 - 254 ;
 66 - 0 ; 67 - 0 ; 68 - 178 ; 69 - 255 ; 70 - 52 ; 71 - 255 ;
 72 - 53 ; 73 - 255 ; 74 - 54 ; 75 - 255 ; 76 - 254 ; 77 - 178 ;
 78 - 255 ; 79 - 54 ; 80 - 255 ; 81 - 254 ; 82 - 0 ; 83 - 0 ;
 84 - 178 ; 85 - 255 ; 86 - 55 ; 87 - 255 ; 88 - 56 ; 89 - 255 ;
 90 - 57 ; 91 - 255 ; 92 - 254 ; 93 - 58 ; 94 - 255 ; 95 - 254 ;
 96 - 59 ; 97 - 255 ; 98 - 60 ; 99 - 255 ; 100 - 254 ; 101 - 49 ;
 102 - 254 ; 103 - 255 ; 104 - 0 ; 105 - 0 ; 106 - 50 ; 107 - 254 ;
 108 - 255 ; 109 - 0 ; 110 - 0 ; 111 - 0 ; 112 - 0 ; 113 - 49 ;
 114 - 255 ; 115 - 158 ; 116 - 255 ; 117 - 174 ; 118 - 255 ; 119 - 254 ;
 120 - 50 ; 121 - 255 ; 122 - 162 ; 123 - 254 ; 124 - 174 ; 125 - 255 ;
 126 - 254 ; 127 - 146 ; 128 - 255 ; 129 - 254 ; 130 - 150 ; 131 - 255 ;
 132 - 254 ; 133 - 154 ; 134 - 255 ; 135 - 254 ; 136 - 178 ; 137 - 255 ;
 138 - 121 ; 139 - 255 ; 140 - 254 ; 141 - 178 ; 142 - 255 ; 143 - 122 ;
 144 - 255 ; 145 - 254 ;
PAGE  2:
 2 - 255 ; 3 - 48 ; 4 - 255 ; 5 - 99 ; 6 - 255 ; 7 - 11 ;
 8 - 254 ; 9 - 255 ; 10 - 0 ; 11 - 111 ; 12 - 255 ; 13 - 53 ;
 14 - 75 ; 15 - 255 ; 16 - 255 ; 17 - 36 ; 18 - 48 ; 19 - 24 ;
 20 - 254 ; 21 - 255 ; 22 - 0 ; 23 - 0 ; 24 - 255 ; 25 - 56 ;
```

```
26 - 178 ; 27 - 255 ; 28 - 173 ; 29 - 100 ; 30 - 255 ; 31 - 60 ;
32 - 254 ; 33 - 255 ; 34 - 0 ; 35 - 0 ; 36 - 255 ; 37 - 56 ;
38 - 178 ; 39 - 255 ; 40 - 173 ; 41 - 101 ; 42 - 255 ; 43 - 72 ;
44 - 254 ; 45 - 255 ; 46 - 0 ; 47 - 0 ; 48 - 255 ; 49 - 56 ;
50 - 178 ; 51 - 255 ; 52 - 173 ; 53 - 102 ; 54 - 255 ; 55 - 87 ;
56 - 254 ; 57 - 255 ; 58 - 0 ; 59 - 0 ; 60 - 112 ; 61 - 255 ;
62 - 61 ; 63 - 178 ; 64 - 255 ; 65 - 173 ; 66 - 103 ; 67 - 145 ;
68 - 255 ; 69 - 254 ; 70 - 255 ; 71 - 0 ; 72 - 113 ; 73 - 255 ;
74 - 68 ; 75 - 178 ; 76 - 77 ; 77 - 255 ; 78 - 173 ; 79 - 104 ;
80 - 255 ; 81 - 99 ; 82 - 254 ; 83 - 255 ; 84 - 0 ; 85 - 0 ;
86 - 0 ; 87 - 114 ; 88 - 255 ; 89 - 77 ; 90 - 178 ; 91 - 255 ;
92 - 173 ; 93 - 105 ; 94 - 149 ; 95 - 255 ; 96 - 254 ; 97 - 255 ;
98 - 0 ; 99 - 116 ; 100 - 255 ; 101 - 84 ; 102 - 178 ; 103 - 76 ;
104 - 78 ; 105 - 79 ; 106 - 80 ; 107 - 81 ; 108 - 255 ; 109 - 144 ;
110 - 148 ; 111 - 173 ; 112 - 106 ; 113 - 255 ; 114 - 118 ; 115 - 254 ;
116 - 255 ; 117 - 0 ; 118 - 118 ; 119 - 255 ; 120 - 93 ; 121 - 82 ;
122 - 255 ; 123 - 255 ; 124 - 130 ; 125 - 142 ; 126 - 254 ; 127 - 255 ;
128 - 0 ; 129 - 0 ; 130 - 255 ; 131 - 56 ; 132 - 178 ; 133 - 255 ;
134 - 173 ; 135 - 107 ; 136 - 153 ; 137 - 255 ; 138 - 254 ; 139 - 255 ;
140 - 0 ; 141 - 0 ; 142 - 255 ; 143 - 56 ; 144 - 178 ; 145 - 255 ;
146 - 173 ; 147 - 108 ; 148 - 255 ; 149 - 153 ; 150 - 254 ; 151 - 255 ;
152 - 0 ; 153 - 120 ; 154 - 255 ; 155 - 96 ; 156 - 83 ; 157 - 84 ;
158 - 255 ; 159 - 152 ; 160 - 109 ; 161 - 255 ; 162 - 2 ; 163 - 254 ;
164 - 255 ; 165 - 0 ; 166 - 121 ; 167 - 255 ; 168 - 101 ; 169 - 255 ;
170 - 157 ; 171 - 255 ; 172 - 254 ; 173 - 255 ; 174 - 0 ; 175 - 0 ;
176 - 122 ; 177 - 255 ; 178 - 106 ; 179 - 255 ; 180 - 161 ; 181 - 255 ;
182 - 254 ; 183 - 255 ; 184 - 0 ; 185 - 0 ; 186 - 255 ; 187 - 113 ;
188 - 255 ; 189 - 96 ; 190 - 156 ; 191 - 164 ; 192 - 172 ; 193 - 169 ;
194 - 177 ; 195 - 255 ; 196 - 166 ; 197 - 254 ; 198 - 101 ; 199 - 255 ;
200 - 164 ; 201 - 169 ; 202 - 255 ; 203 - 254 ; 204 - 255 ; 205 - 1 ;
206 - 0 ; 207 - 255 ; 208 - 120 ; 209 - 255 ; 210 - 97 ; 211 - 160 ;
212 - 168 ; 213 - 172 ; 214 - 165 ; 215 - 177 ; 216 - 255 ; 217 - 170 ;
218 - 254 ; 219 - 106 ; 220 - 255 ; 221 - 168 ; 222 - 165 ; 223 - 255 ;
224 - 254 ; 225 - 255 ; 226 - 1 ; 227 - 0 ;
```

Figure A-11

```
          T>2  ----------  8-JUILLET-81
LES ETAPES INIT.: 2  186  255
M= 1

X1= 1          X2= 1

CYCLE Ci= 3    OPERAT. PARAL.: 0  0  1  0  0  0  0  0  0  0  0
CYCLE Ci= 4    Y3= .540302

CYCLE Ci= 5    OPERAT. PARAL.: 0  0  0  2  0  0  0  0  0  0  0
CYCLE Ci= 7    OPERAT. PARAL.: 0  1  0  2  0  0  0  0  0  0  0
CYCLE Ci= 11   Y2= .841471

CYCLE Ci= 11   Y4= 2.71828

CYCLE Ci= 13   OPERAT. PARAL.: 0  0  0  0  0  1  0  0  0  0  0
CYCLE Ci= 15   OPERAT. PARAL.: 0  0  0  0  0  1  2  0  0  0  0
CYCLE Ci= 16   Y6= 1.11608

CYCLE Ci= 16   Y7= 16.1543

CYCLE Ci= 18   OPERAT. PARAL.: 0  0  0  0  1  0  0  0  0  0  0
CYCLE Ci= 28   Y5= .708074

CYCLE Ci= 29   OPERAT. PARAL.: 0  0  0  0  0  0  0  2  0  0  0
CYCLE Ci= 30   Y8= .0489201

CYCLE Ci= 32   OPERAT. PARAL.: 0  0  0  0  0  0  0  0  1  0  0
CYCLE Ci= 33   Y9= 1.0477

CYCLE Ci= 34   OPERAT. PARAL.: 0  0  0  0  0  0  0  0  0  2  0
CYCLE Ci= 41   Y10=-.575953

-------------------------------------------------------------

M= 1

X1= 1          X2= 1

CYCLE Ci= 3    OPERAT. PARAL.: 0  0  1  0  0  0  0  0  0  0  0
CYCLE Ci= 5    OPERAT. PARAL.: 0  0  1  2  0  0  0  0  0  0  0
CYCLE Ci= 7    Y3= .540302

CYCLE Ci= 10   OPERAT. PARAL.: 0  1  0  2  0  0  0  0  0  0  0
CYCLE Ci= 10   Y4= 2.71828

CYCLE Ci= 11   Y2= .841471

CYCLE Ci= 13   OPERAT. PARAL.: 0  0  0  0  0  2  0  0  0  0  0
CYCLE Ci= 13   Y6= 1.11608

CYCLE Ci= 15   OPERAT. PARAL.: 0  0  0  0  0  0  1  0  0  0  0
CYCLE Ci= 16   Y7= 16.1543

CYCLE Ci= 17   OPERAT. PARAL.: 0  0  0  0  2  0  0  0  0  0  0
CYCLE Ci= 23   Y5= .708074

CYCLE Ci= 24   OPERAT. PARAL.: 0  0  0  0  0  0  0  1  0  0  0
CYCLE Ci= 24   Y8= .0489201

CYCLE Ci= 26   OPERAT. PARAL.: 0  0  0  0  0  0  0  0  2  0  0
CYCLE Ci= 28   OPERAT. PARAL.: 0  0  0  0  0  0  0  0  2  1  0
```

```
CYCLE Ci= 28   Y9= 1.0477

CYCLE Ci= 28   Y10=-.575953

-------------------------------------------------------------

M= 1

X1= 1          X2= 1

CYCLE Ci= 3    OPERAT. PARAL.: 0  0  1  0  0  0  0  0  0  0  0
CYCLE Ci= 5    OPERAT. PARAL.: 0  0  1  2  0  0  0  0  0  0  0
CYCLE Ci= 5    Y4= 2.71828

CYCLE Ci= 7    OPERAT. PARAL.: 0  0  1  0  0  0  2  0  0  0  0
CYCLE Ci= 10   Y3= .540302

CYCLE Ci= 10   Y7= 16.1543

CYCLE Ci= 12   OPERAT. PARAL.: 0  1  0  0  0  0  0  0  0  0  0
CYCLE Ci= 18   Y2= .841471

CYCLE Ci= 19   OPERAT. PARAL.: 0  0  0  0  0  2  0  0  0  0  0
CYCLE Ci= 21   OPERAT. PARAL.: 0  0  0  0  1  2  0  0  0  0  0
CYCLE Ci= 22   Y6= 1.11608

CYCLE Ci= 30   Y5= .708074

CYCLE Ci= 31   OPERAT. PARAL.: 0  0  0  0  0  0  0  2  0  0  0
CYCLE Ci= 33   Y8= .0489201

CYCLE Ci= 35   OPERAT. PARAL.: 0  0  0  0  0  0  0  0  1  0  0
CYCLE Ci= 36   Y9= 1.0477

CYCLE Ci= 37   OPERAT. PARAL.: 0  0  0  0  0  0  0  0  0  2  0
CYCLE Ci= 38   Y10=-.575953

-------------------------------------------------------------
```

Figure A-12

LE PROGRAMME DE SIMULATION DU RdP3

```
1 REM_PROGRAMME DE SIMULATION DU RdP3_____PSRDP3
3 DIM Q1(16),Q2(16),Q3(16),Q4(16),P(16),Y0(16),P1(16),N0(16),C0(16),Y(10)
5 OPEN "SY1:MFD" AS FILE #1
6 OPEN "LP:" FOR OUTPUT AS FILE #9
7 DIM #1,M(3,256)
9 FOR I=0 TO 15 \ P(I)=0 \ P1(I)=0 \ N0(I)=0 \ C0(I)=0 \ NEXT I
21 REM__MISE A ZERO LES ENTREES ET V.INT.
23 S2=255 \ S0=M(0,3) \ S1=0
29 D=M(S1,S0) \ IF D=S2 THEN 39
33 M(S1,S0)=0 \ S0=S0+1 \ GO TO 29
39 S0=M(0,5)
41 D=M(S1,S0) \ IF D=S2 THEN 53
51 Y0=S0+2 \ M(S1,S0)=0 \ S0=S0+2 \ GO TO 41
53 S0=M(0,15)
59 D=M(S1,S0) \ IF D=S2 THEN 61
57 M(S1,S0)=1 \ S0=S0+1 \ GO TO 55
60 REM__INITIALISATION DE Ei0 et Mi0
61 GOSUB 421 \ FOR I=1 TO 4 \ Q1(I)=1 \ NEXT I
63 PRINT #9,"LES ETAPES INIT. :";
65 S0=M(0,6) \ C=0 \ S1=0
67 D=M(S1,S0) \ M(R5,R4)=D \ R4=R4+1
71 PRINT #9,D; \ IF D=S2 THEN 77
73 S0=S0+1 \ GO TO 67
77 S0=S0+1 \ D=M(S1,S0) \ IF D=S2 THEN 85
81 PRINT "adM:";D \ M(0,D)=1 \ GO TO 77
85 PRINT
86 PRINT "SI VOUS VOULEZ COMMENCER,ENTREZ M:" \ INPUT Q1(0)
88 PRINT #9, \ PRINT #9,"M=";Q1(0)
90 REM__ACQUISITION D'ENTREE
91 C=C+1 \ PRINT
93 S5=-M(0,13) \ PRINT "DEBUT D'UNE CYCLE";"_____CYCLE Ci=";C
95 FOR I=0 TO 15 \ Q2(I)=Q1(I) \ NEXT I
99 S0=M(0,3) \ S1=0 \ R1=1 \ PRINT "ENTREE:";
103 M(S1,S0)=Q2(0)*R1 \ PRINT M(S1,S0);
107 FOR I=0 TO 15 \ Q2(I)=Q2(I+1) \ NEXT I
113 S0=S0+1 \ S5=S5+1 \ IF S5=0 THEN 121
119 GO TO 103
120 REM__MISE A ZERO LES Ri et Si,INIT. PLEAC,PLEAS
121 S0=M(0,5) \ S1=0
123 D=M(S1,S0) \ IF D=S2 THEN 135
125 M(S1,S0)=0 \ S0=S0+1 \ M(S1,S0)=0 \ S0=S0+3 \ GO TO 123
133 D=M(S1,S0) \ GO TO 123
135 GOSUB 421 \ GOSUB 429 \ PRINT  \ PRINT "LES ETAPES ACT. EN COURS:"
140 REM__TRAITEMENT D'UNE ETAPE
141 D=M(R5,R4) \ IF D=S2 THEN 251
143 PRINT D; \ R2=D \ R3=M(0,1) \ M(0,14)=0
149 D=M(R3,R2) \ IF D=S2 THEN 159
151 S0=D \ M(0,S0)=1 \ R2=R2+1 \ GO TO 149
152 R2=R2+1
161 S0=M(R3,R2) \ PRINT "adSP:";S0
170 REM__CALCUL DE CONDITION D,EVOLUTION
171 S1=M(0,2) \ GOSUB 437 \ PRINT "EVt:";Q \ IF Q<>0 THEN 191
177 R2=R2+1 \ IF M(R3,R2)<>S2-1 THEN 177
181 R2=R2+1 \ IF M(R3,R2)=S2 THEN 231
183 GO TO 161
190 REM__ACTIONS ASSOCIEES A TRANSITION
191 M(0,14)=1 \ PRINT
195 R2=R2+1 \ D=M(R3,R2) \ IF D=S2 THEN 205
201 M(0,D)=0 \ GO TO 195
205 R2=R2+1 \ D=M(R3,R2) \ IF D=S2 THEN 215
211 M(0,D)=1 \ GO TO 205
```

```
215 R2=R2+1 \ D=M(R3,R2) \ IF D=S2-1 THEN 181
220 M(R7,R6)=D \ R6=R6+1 \ GO TO 215
230 REM__TRAITEMENT D'ETAPE CLE SYNCHRONE
231 IF M(0,14)<>0 THEN 241
235 R2=R2+1 \ IF M(R3,R2)<>0 THEN 241
237 D=M(R5,R4)
240 M(R7,R6)=D \ R6=R6+1
241 R4=R4+1 \ GO TO 141
250 REM__CALCUL DE M1
251 S1=0 \ S0=M(0,5)
255 IF M(S1,S0)=S2 THEN 301
257 R0=M(S1,S0) \ S0=S0+1 \ Q=M(S1,S0) \ S0=S0+1
259 PRINT "R1:";R0;"S1:";Q;
260 R1=M(S1,S0) \ PRINT "Mc:";R1; \ IF Q<>0 THEN 281
269 IF R0=0 THEN 273
271 R0=0 \ GO TO 275
273 R0=1
275 IF R1*R0<>0 THEN 281
277 M(S1,S0)=0 \ PRINT M(S1,S0); \ S0=S0+1 \ GO TO 291
281 M(S1,S0)=1 \ PRINT M(S1,S0); \ S0=S0+1 \ D=M(S1,S0)
287 IF D=S2 THEN 291
289 M(R7,R6)=D \ R6=R6+1
291 S0=S0+1 \ GO TO 255
300 REM__CALCUL DE C. G.
301 M(R7,R6)=S2 \ REM__MISE FF(255)DANSLAES
303 R2=M(0,11) \ R3=M(0,12) \ REM__POINTE A LA LISTE DE C.G.
307 IF M(R3,R2)=S2 THEN 341
309 S0=M(R3,R2) \ S1=M(0,2)
313 GOSUB 437 \ IF Q=0 THEN 327
312 R2=R2+1 \ D=M(R3,R2) \ M(0,D)=1
323 R2=R2+2 \ GO TO 307
327 R2=R2+1 \ D=M(R3,R2) \ M(0,D)=0 \ GO TO 323
340 REM__AFFECTION DES SORTIES
341 S5=1 \ S0=M(0,4) \ S1=0 \ REM_POINTE A LA LISTE SE SORTIE
345 PRINT  \ PRINT "SORTIE:"
347 S5=-24 \ FOR I=0 TO 15 \ Q2(I)=0 \ NEXT I
355 D=M(S1,S0) \ IF D=S2 THEN 391
359 Q2(8)=D \ GOSUB 769 \ PRINT Q2(7);
363 S5=S5+1 \ S0=S0+1 \ D=M(S1,S0) \ IF D<>S2 THEN 359
368 IF S5=0 THEN 375
371 GOSUB 769 \ S5=S5+1 \ GO TO 368
375 Z9=S3 \ FOR I=0 TO 15 \ Q3(I)=Q2(I) \ NEXT I \ GOSUB 473
385 S3=S3+1 \ PRINT  \ S0=S0+1 \ GO TO 347
390 REM__MISE A ZERO LA LISTE DE SORTIE ET COPIE LEAS
391 S0=M(0,4) \ S1=0
393 IF M(S1,S0)=S2 THEN 402
395 M(S1,S0)=0 \ S0=S0+1 \ IF M(S1,S0)<>S2 THEN 395
401 S0=S0+1 \ GO TO 393
402 S5=M(0,13)-1 \ S0=M(0,15)
403 FOR I=0 TO S5 \ Q2(I)=M(0,S0) \ M(0,S0)=1 \ S0=S0+1
404 M1(I)=Q1(I)*Q2(I) \ PRINT "MASQ.:";M(0,S0-1); \ NEXT I
407 GOSUB 421 \ GOSUB 429 \ PRINT
409 PRINT "LES ETAPES ACTIVEES SUIVANTES:"
410 D=M(R7,R6) \ M(R5,R4)=D \ R4=R4+1
414 IF D=S2 THEN 418
416 PRINT D; \ R6=R6+1 \ GO TO 410
418 PRINT  \ PRINT "FIN D'UN CYCLE"
419 GO TO 76
420 REM__SOUSPROGRAMME__SMP7
421 R4=M(0,7) \ R5=M(0,8) \ RETURN
427 REM__SMP8
```

```
429 R6=M(0,9) \ R7=M(0,10) \ RETURN
435 REM__SMP10
437 D=M(S1,S0) \ IF D=S2 THEN 463
439 IF D=S2-1 THEN 467
443 D=M(0,D) \ IF D=0 THEN 451
445 S0=S0+2 \ GO TO 437
451 S0=S0+1 \ D=M(S1,S0) \ IF D=S2 THEN 463
457 IF D=S2-1 THEN 467
459 S0=0 \ GO TO 437
463 Q=0
465 RETURN
467 Q=1
469 GO TO 465
471 REM__SOUSPROGRAMME__"EXECUT."
473 ON Z9 GO TO 479,501,477
477 RETURN
479 FOR I=0 TO 3 \ IF Q3(I)=0 THEN 483
481 P1(I)=Q3(I) \ Q1(I+1)=0
483 NEXT I \ GO TO 477
501 IF Q3(0)=1 THEN  GOSUB 601
503 IF Q3(9)=1 THEN  GOSUB 685
505 IF Q3(8)=1 THEN  GOSUB 675
507 IF Q3(7)=1 THEN  GOSUB 665
509 IF Q3(6)=1 THEN  GOSUB 655
511 IF Q3(5)=1 THEN  GOSUB 645
513 IF Q3(4)=1 THEN  GOSUB 635
515 IF Q3(3)=1 THEN  GOSUB 625
517 IF Q3(2)=1 THEN  GOSUB 615
519 IF Q3(1)=1 THEN  GOSUB 605
521 IF Q3(10)=1 THEN  GOSUB 702
535 FOR I=1 TO 9 \ IF P(I)<>0 THEN 539
537 NEXT I \ GO TO 477
539 RANDOMIZE \ T=INT(RND*10+1) \ IF T>2 THEN 537
541 PRINT #9 \ PRINT #9,"CYCLE:Ci="$C$","$
542 PRINT #9,"CADRE Ni="$NO(I)$":",
543 S1(I+5)=1 \ PRINT #9,"Y"$I+1$"="$Y(I+1)
545 Q1(P(I))=1 \ P(I)=0 \ GO TO 537
601 INPUT X1,X2 \ Q1(5)=1 \ P(0)=0 \ NO(0)=NO(0)+1
603 PRINT #9 \ PRINT #9,"CYCLE Ci="$C$","$"CADRE Ni="$NO(0)$":",
604 PRINT #9,"X1="$X1$"X2="$X2 \ RETURN
605 FOR I=0 TO 3 \ IF P1(I)<>0 THEN 609
607 NEXT I \ RETURN
609 P(1)=I+1 \ P1(I)=0 \ NO(1)=NO(1)+1
611 Y(2)=SIN(X1) \ GOSUB 705
615 FOR I=0 TO 3 \ IF P1(I)=0 THEN 623
619 P(2)=I+1 \ P1(I)=0 \ NO(2)=NO(2)+1
621 Y(3)=COS(X2) \ GOSUB 705
623 NEXT I \ RETURN
625 FOR I=0 TO 3 \ IF P1(I)=0 THEN 633
629 P(3)=I+1 \ P1(I)=0 \ NO(3)=NO(3)+1
631 Y(4)=EXP(X1) \ GOSUB 705
633 NEXT I \ RETURN
635 FOR I=0 TO 3 \ IF P1(I)=0 THEN 643
639 P(4)=I+1 \ P1(I)=0 \ NO(4)=NO(4)+1
641 Y(5)=Y(2)^2 \ GOSUB 705
643 NEXT I \ RETURN
645 FOR I=0 TO 3 \ IF P1(I)=0 THEN 653
649 P(5)=I+1 \ P1(I)=0 \ NO(5)=NO(5)+1
651 Y(6)=SQR(Y(2))+Y(3)/Y(4) \ GOSUB 705
653 NEXT I \ RETURN
655 FOR I=0 TO 3 \ IF P1(I)=0 THEN 663
```

```
659 P(6)=I+1 \ P1(I)=0 \ NO(6)=NO(6)+1
661 Y(7)=1+EXP(Y(4)) \ GOSUB 705
663 NEXT I \ RETURN
665 FOR I=0 TO 3 \ IF P1(I)=0 THEN 673
669 P(7)=I+1 \ P1(I)=0 \ NO(7)=NO(7)+1
671 Y(8)=Y(5)*Y(6)/Y(7) \ GOSUB 705
673 NEXT I \ RETURN
675 FOR I=0 TO 3 \ IF P1(I)=0 THEN 683
679 P(8)=I+1 \ P1(I)=0 \ NO(8)=NO(8)+1
681 Y(9)=SIN(Y(8))+COS(Y(8)) \ GOSUB 705
683 NEXT I \ RETURN
685 FOR I=0 TO 3 \ IF P1(I)=0 THEN 695
689 P(9)=I+1 \ P1(I)=0 \ NO(9)=NO(9)+1
691 Y0(0)=1 \ FOR J=0 TO 10 \ Y0(J+1)=Y0(J)*(Y(8)-1) \ NEXT J
693 Y(10)=Y0(11) \ GOSUB 705
695 NEXT I \ RETURN
702 RETURN
705 PRINT #9, \ PRINT #9,"Ci=";C;
707 PRINT #9,"OPERAT. PARAL.:";
709 FOR J=0 TO 10 \ PRINT #9,P(J); \ NEXT J
711 PRINT #9 \ PRINT #9,"_______CADRE  CORESP.:";
713 FOR J=0 TO 10 \ PRINT #9,NO(J); \ NEXT J \ RETURN
769 L=Q2(0) \ FOR I=0 TO 15 \ Q2(I)=Q2(I+1) \ NEXT I
779 Q2(15)=L \ RETURN
781 CLOSE #1 \ CLOSE #9
783 END
```

Figure A-13

```
_____DONNEES DU STRUCTURE DE  RdP3_____

PAGE  0 :
LISTE DES CONSTANTES:
 0 - 0 ; 1 - 2 ; 2 - 1 ; 3 - 48 ; 4 - 96 ; 5 - 140 ;
 6 - 20 ; 7 - 2 ; 8 - 1 ; 9 - 17 ; 10 - 1 ; 11 - 32 ;
 12 - 1 ; 13 - 15 ; 14 - 0 ; 15 - 72 ; 16 - 255 ; 17 - 255 ;

LISTE INITIALE:
 20 - 167 ; 21 - 186 ; 22 - 205 ; 23 - 224 ; 24 - 235 ; 25 - 255 ;
 26 - 142 ; 27 - 158 ; 28 - 166 ; 29 - 170 ; 30 - 174 ; 31 - 178 ;
 32 - 182 ; 33 - 255 ; 34 - 255 ;
LISTE D'ENTREE:
 48 - 0 ; 49 - 0 ; 50 - 0 ; 51 - 0 ; 52 - 0 ; 53 - 0 ;
 54 - 0 ; 55 - 0 ; 56 - 0 ; 57 - 0 ; 58 - 0 ; 59 - 0 ;
 60 - 0 ; 61 - 0 ; 62 - 0 ; 63 - 255 ; 64 - 255 ;
MASQUE D'ENTREE:
 72 - 1 ; 73 - 1 ; 74 - 1 ; 75 - 1 ; 76 - 1 ; 77 - 1 ;
 78 - 1 ; 79 - 1 ; 80 - 1 ; 81 - 1 ; 82 - 1 ; 83 - 1 ;
 84 - 1 ; 85 - 1 ; 86 - 1 ; 87 - 255 ; 88 - 255 ;
LISTE DE SORTIE:
 96 - 0 ; 97 - 0 ; 98 - 0 ; 99 - 0 ; 100 - 255 ; 101 - 0 ;
 102 - 0 ; 103 - 0 ; 104 - 0 ; 105 - 0 ; 106 - 0 ; 107 - 0 ;
 108 - 0 ; 109 - 0 ; 110 - 0 ; 111 - 0 ; 112 - 255 ; 113 - 0 ;
 114 - 0 ; 115 - 0 ; 116 - 0 ; 117 - 0 ; 118 - 0 ; 119 - 0 ;
 120 - 0 ; 121 - 0 ; 122 - 0 ; 123 - 255 ; 124 - 255 ;
LISTE DE V. INTER.:
 140 - 0 ; 141 - 0 ; 142 - 0 ; 143 - 255 ; 144 - 0 ; 145 - 0 ;
 146 - 0 ; 147 - 255 ; 148 - 0 ; 149 - 0 ; 150 - 0 ; 151 - 255 ;
 152 - 0 ; 153 - 0 ; 154 - 0 ; 155 - 255 ; 156 - 0 ; 157 - 0 ;
 158 - 0 ; 159 - 255 ; 160 - 0 ; 161 - 0 ; 162 - 0 ; 163 - 255 ;
 164 - 0 ; 165 - 0 ; 166 - 0 ; 167 - 235 ; 168 - 0 ; 169 - 0 ;
 170 - 0 ; 171 - 255 ; 172 - 0 ; 173 - 0 ; 174 - 0 ; 175 - 255 ;
 176 - 0 ; 177 - 0 ; 178 - 0 ; 179 - 255 ; 180 - 0 ; 181 - 0 ;
 182 - 0 ; 183 - 255 ; 184 - 255 ;
PAGE  1:
LISTE DE C.G. :
 32 - 186 ; 33 - 117 ; 34 - 255 ; 35 - 189 ; 36 - 119 ; 37 - 255 ;
 38 - 192 ; 39 - 121 ; 40 - 255 ; 41 - 255 ;
LISTE DE CONDITION D'EVOLUT. :
 80 - 48 ; 81 - 255 ; 82 - 142 ; 83 - 255 ; 84 - 254 ; 85 - 0 ;
 86 - 0 ; 87 - 170 ; 88 - 255 ; 89 - 53 ; 90 - 255 ; 91 - 254 ;
 92 - 0 ; 93 - 0 ; 94 - 162 ; 95 - 255 ; 96 - 254 ; 97 - 0 ;
 98 - 0 ; 99 - 162 ; 100 - 255 ; 101 - 54 ; 102 - 255 ; 103 - 174 ;
 104 - 255 ; 105 - 254 ; 106 - 174 ; 107 - 255 ; 108 - 162 ; 109 - 255 ;
 110 - 54 ; 111 - 255 ; 112 - 55 ; 113 - 255 ; 114 - 56 ; 115 - 255 ;
 116 - 254 ; 117 - 162 ; 118 - 255 ; 119 - 56 ; 120 - 255 ; 121 - 174 ;
 122 - 255 ; 123 - 254 ; 124 - 178 ; 125 - 255 ; 126 - 162 ; 127 - 255 ;
 128 - 57 ; 129 - 255 ; 130 - 58 ; 131 - 255 ; 132 - 59 ; 133 - 255 ;
 134 - 254 ; 135 - 182 ; 136 - 255 ; 137 - 60 ; 138 - 255 ; 139 - 254 ;
 140 - 0 ; 141 - 0 ; 142 - 61 ; 143 - 255 ; 144 - 62 ; 145 - 255 ;
 146 - 254 ; 147 - 49 ; 148 - 254 ; 149 - 255 ; 150 - 49 ; 151 - 255 ;
 152 - 158 ; 153 - 255 ; 154 - 254 ; 155 - 50 ; 156 - 254 ; 157 - 255 ;
 158 - 50 ; 159 - 255 ; 160 - 158 ; 161 - 255 ; 162 - 254 ; 163 - 51 ;
 164 - 254 ; 165 - 255 ; 166 - 51 ; 167 - 255 ; 168 - 158 ; 169 - 255 ;
 170 - 254 ; 171 - 52 ; 172 - 254 ; 173 - 255 ; 174 - 52 ; 175 - 255 ;
 176 - 158 ; 177 - 255 ; 178 - 254 ; 179 - 0 ; 180 - 0 ; 181 - 48 ;
 182 - 254 ; 183 - 255 ; 184 - 0 ; 185 - 0 ; 186 - 146 ; 187 - 255 ;
 188 - 254 ; 189 - 150 ; 190 - 255 ; 191 - 254 ; 192 - 154 ; 193 - 255 ;
 194 - 254 ;
PAGE  2:
```

```
2 - 113 ; 3 - 255 ; 4 - 87 ; 5 - 77 ; 6 - 255 ; 7 - 168 ;
8 - 255 ; 9 - 37 ; 10 - 15 ; 11 - 26 ; 12 - 254 ; 13 - 255 ;
14 - 0 ; 15 - 255 ; 16 - 94 ; 17 - 162 ; 18 - 255 ; 19 - 157 ;
20 - 102 ; 21 - 255 ; 22 - 48 ; 23 - 254 ; 24 - 255 ; 25 - 0 ;
26 - 255 ; 27 - 94 ; 28 - 162 ; 29 - 255 ; 30 - 157 ; 31 - 103 ;
32 - 255 ; 33 - 60 ; 34 - 254 ; 35 - 255 ; 36 - 0 ; 37 - 255 ;
38 - 94 ; 39 - 162 ; 40 - 255 ; 41 - 157 ; 42 - 104 ; 43 - 255 ;
44 - 78 ; 45 - 254 ; 46 - 255 ; 47 - 0 ; 48 - 114 ; 49 - 255 ;
50 - 99 ; 51 - 162 ; 52 - 255 ; 53 - 145 ; 54 - 157 ; 55 - 105 ;
56 - 255 ; 57 - 254 ; 58 - 255 ; 59 - 0 ; 60 - 115 ; 61 - 255 ;
62 - 106 ; 63 - 162 ; 64 - 78 ; 65 - 79 ; 66 - 80 ; 67 - 255 ;
68 - 172 ; 69 - 157 ; 70 - 165 ; 71 - 106 ; 72 - 255 ; 73 - 91 ;
74 - 254 ; 75 - 255 ; 76 - 0 ; 77 - 0 ; 78 - 116 ; 79 - 255 ;
80 - 117 ; 81 - 162 ; 82 - 255 ; 83 - 149 ; 84 - 157 ; 85 - 107 ;
86 - 255 ; 87 - 254 ; 88 - 255 ; 89 - 0 ; 90 - 0 ; 91 - 118 ;
92 - 255 ; 93 - 124 ; 94 - 162 ; 95 - 81 ; 96 - 82 ; 97 - 83 ;
98 - 255 ; 99 - 144 ; 100 - 148 ; 101 - 176 ; 102 - 157 ; 103 - 169 ;
104 - 108 ; 105 - 255 ; 106 - 110 ; 107 - 254 ; 108 - 255 ; 109 - 0 ;
110 - 120 ; 111 - 255 ; 112 - 135 ; 113 - 84 ; 114 - 255 ; 115 - 180 ;
116 - 173 ; 117 - 255 ; 118 - 134 ; 119 - 123 ; 120 - 254 ; 121 - 255 ;
122 - 0 ; 123 - 255 ; 124 - 94 ; 125 - 162 ; 126 - 255 ; 127 - 153 ;
128 - 157 ; 129 - 109 ; 130 - 255 ; 131 - 254 ; 132 - 255 ; 133 - 0 ;
134 - 255 ; 135 - 94 ; 136 - 162 ; 137 - 255 ; 138 - 157 ; 139 - 110 ;
140 - 255 ; 141 - 145 ; 142 - 254 ; 143 - 255 ; 144 - 0 ; 145 - 122 ;
146 - 255 ; 147 - 142 ; 148 - 85 ; 149 - 86 ; 150 - 255 ; 151 - 152 ;
152 - 177 ; 153 - 181 ; 154 - 111 ; 155 - 255 ; 156 - 254 ; 157 - 255 ;
158 - 0 ; 159 - 255 ; 160 - 147 ; 161 - 255 ; 162 - 255 ; 163 - 167 ;
164 - 254 ; 165 - 255 ; 166 - 0 ; 167 - 255 ; 168 - 150 ; 169 - 158 ;
170 - 255 ; 171 - 161 ; 172 - 96 ; 173 - 255 ; 174 - 159 ; 175 - 254 ;
176 - 255 ; 177 - 0 ; 178 - 255 ; 179 - 155 ; 180 - 255 ; 181 - 255 ;
182 - 186 ; 183 - 254 ; 184 - 255 ; 185 - 0 ; 186 - 255 ; 187 - 158 ;
188 - 158 ; 189 - 255 ; 190 - 161 ; 191 - 97 ; 192 - 255 ; 193 - 178 ;
194 - 254 ; 195 - 255 ; 196 - 0 ; 197 - 255 ; 198 - 163 ; 199 - 255 ;
200 - 255 ; 201 - 205 ; 202 - 254 ; 203 - 255 ; 204 - 0 ; 205 - 255 ;
206 - 166 ; 207 - 158 ; 208 - 255 ; 209 - 161 ; 210 - 98 ; 211 - 255 ;
212 - 197 ; 213 - 254 ; 214 - 255 ; 215 - 0 ; 216 - 255 ; 217 - 171 ;
218 - 255 ; 219 - 255 ; 220 - 224 ; 221 - 254 ; 222 - 255 ; 223 - 0 ;
224 - 255 ; 225 - 174 ; 226 - 158 ; 227 - 255 ; 228 - 161 ; 229 - 99 ;
230 - 255 ; 231 - 216 ; 232 - 254 ; 233 - 255 ; 234 - 0 ; 235 - 255 ;
236 - 181 ; 237 - 255 ; 238 - 164 ; 239 - 101 ; 240 - 255 ; 241 - 2 ;
242 - 254 ; 243 - 80 ; 244 - 72 ; 245 - 255 ; 246 - 140 ; 247 - 164 ;
248 - 101 ; 249 - 255 ; 250 - 2 ; 251 - 254 ; 252 - 255 ; 253 - 1 ;
254 - 0 ;
```

Figure A-14

```
          RdP3_________ T>2

LES ETAPES INIT. : 167  186  205  224  235  255
M= 1

CYCLE Ci= 1 ,CADRE Ni= 1 :  X1= 1 X2= 1

Ci= 3 OPERAT. PARAL.: 0  0  0  1  0  0  0  0  0  0  0
_____CADRE  CORESP.: 1  0  0  1  0  0  0  0  0  0  0
Ci= 5 OPERAT. PARAL.: 0  2  0  1  0  0  0  0  0  0  0
_____CADRE  CORESP.: 1  1  0  1  0  0  0  0  0  0  0
CYCLE:Ci= 5 ,CADRE Ni= 1 :  Y 4 = 2.71828

Ci= 7 OPERAT. PARAL.: 0  2  0  0  0  0  1  0  0  0  0
_____CADRE  CORESP.: 1  1  0  1  0  0  1  0  0  0  0
CYCLE:Ci= 7 ,CADRE Ni= 1 :  Y 2 = .841471

CYCLE:Ci= 7 ,CADRE Ni= 1 :  Y 7 = 16.1543

Ci= 9 OPERAT. PARAL.: 0  0  0  0  1  0  0  0  0  0  0
_____CADRE  CORESP.: 1  1  0  1  1  0  1  0  0  0  0
CYCLE:Ci= 9 ,CADRE Ni= 1 :  Y 5 = .708074

Ci= 11 OPERAT. PARAL.: 0  0  1  0  0  0  0  0  0  0  0
_____CADRE  CORESP.: 1  1  1  1  1  0  1  0  0  0  0
CYCLE:Ci= 13 ,CADRE Ni= 1 : Y 3 = .540302

Ci= 14 OPERAT. PARAL.: 0  0  0  0  0  2  0  0  0  0  0
_____CADRE  CORESP.: 1  1  1  1  1  1  1  0  0  0  0
CYCLE:Ci= 14 ,CADRE Ni= 1 : Y 6 = 1.11608

CYCLE Ci= 15 ,CADRE Ni= 2 : X1= .8 X2= .8

Ci= 16 OPERAT. PARAL.: 0  0  0  0  0  0  0  1  0  0  0
_____CADRE  CORESP.: 2  1  1  1  1  1  1  1  0  0  0
CYCLE:Ci= 17 ,CADRE Ni= 1 : Y 8 = .0489201

Ci= 18 OPERAT. PARAL.: 0  0  0  2  0  0  0  0  0  0  0
_____CADRE  CORESP.: 2  1  1  2  1  1  1  1  0  0  0
Ci= 20 OPERAT. PARAL.: 0  0  0  2  0  0  0  0  0  1  0
_____CADRE  CORESP.: 2  1  1  2  1  1  1  1  0  1  0
Ci= 22 OPERAT. PARAL.: 0  0  0  2  0  0  0  0  3  1  0
_____CADRE  CORESP.: 2  1  1  2  1  1  1  1  1  1  0
CYCLE:Ci= 22 ,CADRE Ni= 1 : Y 10 =-.575953

Ci= 24 OPERAT. PARAL.: 0  1  0  2  0  0  0  0  3  0  0
_____CADRE  CORESP.: 2  2  1  2  1  1  1  1  1  1  0
Ci= 26 OPERAT. PARAL.: 0  1  4  2  0  0  0  0  3  0  0
_____CADRE  CORESP.: 2  2  2  2  1  1  1  1  1  1  0
CYCLE:Ci= 26 ,CADRE Ni= 2 : Y 2 = .717356

CYCLE:Ci= 26 ,CADRE Ni= 2 : Y 3 = .696707

CYCLE:Ci= 26 ,CADRE Ni= 2 : Y 4 = 2.22554

CYCLE:Ci= 26 ,CADRE Ni= 1 : Y 9 = 1.0477

Ci= 28 OPERAT. PARAL.: 0  0  0  0  0  0  1  0  0  0  0
_____CADRE  CORESP.: 2  2  2  2  1  1  2  1  1  1  0
Ci= 30 OPERAT. PARAL.: 0  0  0  0  2  0  1  0  0  0  0
_____CADRE  CORESP.: 2  2  2  2  2  1  2  1  1  1  0
CYCLE:Ci= 30 ,CADRE Ni= 2 : Y 7 = 10.2585
```

```
Ci= 32 OPERAT. PARAL.: 0  0  0  0  2  1  0  0  0  0  0
_______CADRE  CORESP.: 2  2  2  2  2  2  2  1  1  1  0
CYCLE Ci= 33 ,CADRE Ni= 3 : X1= .5 X2= .5

CYCLE:Ci= 35 ,CADRE Ni= 2 : Y 6 = 1.16002

CYCLE:Ci= 40 ,CADRE Ni= 2 : Y 5 = .5146

Ci= 41 OPERAT. PARAL.: 0  0  0  0  0  0  0  3  0  0  0
_______CADRE  CORESP.: 3  2  2  2  2  2  2  2  1  1  0
Ci= 43 OPERAT. PARAL.: 0  0  0  1  0  0  0  3  0  0  0
_______CADRE  CORESP.: 3  2  2  3  2  2  2  2  1  1  0
Ci= 45 OPERAT. PARAL.: 0  2  0  1  0  0  0  3  0  0  0
_______CADRE  CORESP.: 3  3  2  3  2  2  2  2  1  1  0
CYCLE:Ci= 46 ,CADRE Ni= 3 : Y 2 = .479426

Ci= 47 OPERAT. PARAL.: 0  0  4  1  0  0  0  3  0  0  0
_______CADRE  CORESP.: 3  3  3  3  2  2  2  2  1  1  0
CYCLE:Ci= 47 ,CADRE Ni= 3 : Y 3 = .877583

CYCLE:Ci= 51 ,CADRE Ni= 2 : Y 8 = .0581904

Ci= 53 OPERAT. PARAL.: 0  0  0  1  0  0  0  0  0  2  0
_______CADRE  CORESP.: 3  3  3  3  2  2  2  2  1  2  0
CYCLE:Ci= 53 ,CADRE Ni= 3 : Y 4 = 1.64872

CYCLE:Ci= 54 ,CADRE Ni= 2 : Y 10 =-.517123

Ci= 55 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  1  0  0
_______CADRE  CORESP.: 3  3  3  3  2  2  2  2  2  2  0
CYCLE:Ci= 55 ,CADRE Ni= 2 : Y 9 = 1.05647

Ci= 57 OPERAT. PARAL.: 0  0  0  0  0  0  1  0  0  0  0
_______CADRE  CORESP.: 3  3  3  3  2  2  3  2  2  2  0
Ci= 59 OPERAT. PARAL.: 0  0  0  0  2  0  1  0  0  0  0
_______CADRE  CORESP.: 3  3  3  3  3  2  3  2  2  2  0
Ci= 61 OPERAT. PARAL.: 0  0  0  0  2  3  1  0  0  0  0
_______CADRE  CORESP.: 3  3  3  3  3  3  3  2  2  2  0
CYCLE Ci= 62 ,CADRE Ni= 4 : X1= 1 X2= 1

CYCLE:Ci= 62 ,CADRE Ni= 3 : Y 5 = .229849

CYCLE:Ci= 63 ,CADRE Ni= 3 : Y 6 = 1.22469

CYCLE:Ci= 68 ,CADRE Ni= 3 : Y 7 = 6.20033

Ci= 69 OPERAT. PARAL.: 0  0  0  0  0  0  0  4  0  0  0
_______CADRE  CORESP.: 4  3  3  3  3  3  3  3  2  2  0
CYCLE:Ci= 69 ,CADRE Ni= 3 : Y 8 = .0453997

Ci= 71 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  0  1  0
_______CADRE  CORESP.: 4  3  3  3  3  3  3  3  2  3  0
CYCLE:Ci= 72 ,CADRE Ni= 3 : Y 10 =-.599842

Ci= 73 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  2  0  0
_______CADRE  CORESP.: 4  3  3  3  3  3  3  3  3  3  0
CYCLE:Ci= 73 ,CADRE Ni= 3 : Y 9 = 1.04435

Ci= 75 OPERAT. PARAL.: 0  0  0  1  0  0  0  0  0  0  0
_______CADRE  CORESP.: 4  3  3  4  3  3  3  3  3  3  0
Ci= 77 OPERAT. PARAL.: 0  2  0  1  0  0  0  0  0  0  0
```

```
_______CADRE  CORESP.: 4  4  3  4  3  3  3  3  3  3  0
Ci= 79 OPERAT. PARAL.: 0  2  3  1  0  0  0  0  0  0  0
_______CADRE  CORESP.: 4  4  4  4  3  3  3  3  3  3  0
CYCLE:Ci= 81 ,CADRE Ni= 4 : Y 4 = 2.71828

Ci= 82 OPERAT. PARAL.: 0  2  3  0  0  0  4  0  0  0  0
_______CADRE  CORESP.: 4  4  4  4  3  3  4  3  3  3  0
CYCLE:Ci= 83 ,CADRE Ni= 4 : Y 2 = .841471

Ci= 84 OPERAT. PARAL.: 0  0  3  0  1  0  4  0  0  0  0
_______CADRE  CORESP.: 4  4  4  4  4  3  4  3  3  3  0
CYCLE:Ci= 84 ,CADRE Ni= 4 : Y 3 = .540302

Ci= 86 OPERAT. PARAL.: 0  0  0  0  1  2  4  0  0  0  0
_______CADRE  CORESP.: 4  4  4  4  4  4  4  3  3  3  0
CYCLE Ci= 87 ,CADRE Ni= 5 : X1= .8 X2= .8

CYCLE:Ci= 87 ,CADRE Ni= 4 : Y 5 = .708074

CYCLE:Ci= 87 ,CADRE Ni= 4 : Y 6 = 1.11608

CYCLE:Ci= 92 ,CADRE Ni= 4 : Y 7 = 16.1543

Ci= 93 OPERAT. PARAL.: 0  0  0  0  0  0  0  3  0  0  0
_______CADRE  CORESP.: 5  4  4  4  4  4  4  4  3  3  0
CYCLE:Ci= 93 ,CADRE Ni= 4 : Y 8 = .0489201

Ci= 95 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  0  1  0
_______CADRE  CORESP.: 5  4  4  4  4  4  4  4  3  4  0
Ci= 97 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  2  1  0
_______CADRE  CORESP.: 5  4  4  4  4  4  4  4  4  4  0
CYCLE:Ci= 97 ,CADRE Ni= 4 : Y 9 = 1.0477

CYCLE:Ci= 98 ,CADRE Ni= 4 : Y 10 =-.575953

Ci= 99 OPERAT. PARAL.: 0  0  0  2  0  0  0  0  0  0  0
_______CADRE  CORESP.: 5  4  4  5  4  4  4  4  4  4  0
Ci= 101 OPERAT. PARAL.: 0  1  0  2  0  0  0  0  0  0  0
_______CADRE  CORESP.: 5  5  4  5  4  4  4  4  4  4  0
CYCLE:Ci= 102 ,CADRE Ni= 5 :Y 4 = 2.22554

Ci= 103 OPERAT. PARAL.: 0  1  0  0  0  0  3  0  0  0  0
_______CADRE  CORESP.: 5  5  4  5  4  4  5  4  4  4  0
CYCLE:Ci= 104 ,CADRE Ni= 5 :Y 2 = .717356

Ci= 105 OPERAT. PARAL.: 0  0  0  0  2  0  3  0  0  0  0
_______CADRE  CORESP.: 5  5  4  5  5  4  5  4  4  4  0
CYCLE:Ci= 106 ,CADRE Ni= 5 :Y 5 = .5146

Ci= 107 OPERAT. PARAL.: 0  0  1  0  0  0  3  0  0  0  0
_______CADRE  CORESP.: 5  5  5  5  5  4  5  4  4  4  0
CYCLE:Ci= 107 ,CADRE Ni= 5 :Y 3 = .696707

Ci= 109 OPERAT. PARAL.: 0  0  0  0  0  1  3  0  0  0  0
_______CADRE  CORESP.: 5  5  5  5  5  5  5  4  4  4  0
CYCLE Ci= 110 ,CADRE Ni= 6 :X1= .5 X2= .5

CYCLE:Ci= 113 ,CADRE Ni= 5 :Y 6 = 1.16002

CYCLE:Ci= 114 ,CADRE Ni= 5 :Y 7 = 10.2585
```

```
Ci= 115 OPERAT. PARAL.: 0  0  0  0  0  0  0  2  0  0  0
_______CADRE  CORESP.: 6  5  5  5  5  5  5  5  4  4  0
Ci= 117 OPERAT. PARAL.: 0  0  0  1  0  0  0  2  0  0  0
_______CADRE  CORESP.: 6  5  5  6  5  5  5  5  4  4  0
Ci= 119 OPERAT. PARAL.: 0  3  0  1  0  0  0  2  0  0  0
_______CADRE  CORESP.: 6  6  5  6  5  5  5  5  4  4  0
CYCLE:Ci= 119 ,CADRE Ni= 6 :Y 4 = 1.64872

CYCLE:Ci= 119 ,CADRE Ni= 5 :Y 8 = .0581904

CYCLE:Ci= 120 ,CADRE Ni= 6 :Y 2 = .479426

Ci= 121 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  0  1  0
_______CADRE  CORESP.: 6  6  5  6  5  5  5  5  4  5  0
Ci= 123 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  2  1  0
_______CADRE  CORESP.: 6  6  5  6  5  5  5  5  5  5  0
CYCLE:Ci= 124 ,CADRE Ni= 5 :Y 10 =-.517123

Ci= 125 OPERAT. PARAL.: 0  0  0  0  0  0  3  0  2  0  0
_______CADRE  CORESP.: 6  6  5  6  5  5  6  5  5  5  0
CYCLE:Ci= 125 ,CADRE Ni= 5 :Y 9 = 1.05647

Ci= 127 OPERAT. PARAL.: 0  0  0  0  1  0  3  0  0  0  0
_______CADRE  CORESP.: 6  6  5  6  6  5  6  5  5  5  0
CYCLE:Ci= 127 ,CADRE Ni= 6 :Y 5 = .229849

Ci= 129 OPERAT. PARAL.: 0  0  1  0  0  0  3  0  0  0  0
_______CADRE  CORESP.: 6  6  6  6  6  5  6  5  5  5  0
CYCLE:Ci= 130 ,CADRE Ni= 6 :Y 7 = 6.20033

CYCLE:Ci= 131 ,CADRE Ni= 6 :Y 3 = .877583

Ci= 132 OPERAT. PARAL.: 0  0  0  0  0  2  0  0  0  0  0
_______CADRE  CORESP.: 6  6  6  6  6  6  6  5  5  5  0
CYCLE Ci= 133 ,CADRE Ni= 7 :X1= 1 X2= 1

CYCLE:Ci= 134 ,CADRE Ni= 6 :Y 6 = 1.22469

Ci= 135 OPERAT. PARAL.: 0  0  0  0  0  0  0  1  0  0  0
_______CADRE  CORESP.: 7  6  6  6  6  6  6  5  5  5  0
CYCLE:Ci= 135 ,CADRE Ni= 6 :Y 8 = .0453997

Ci= 137 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  0  1  0
_______CADRE  CORESP.: 7  6  6  6  6  6  6  6  5  6  0
CYCLE:Ci= 137 ,CADRE Ni= 6 :Y 10 =-.599842

Ci= 139 OPERAT. PARAL.: 0  0  0  0  0  0  0  0  1  0  0
_______CADRE  CORESP.: 7  6  6  6  6  6  6  6  6  6  0
Ci= 141 OPERAT. PARAL.: 0  0  0  2  0  0  0  0  1  0  0
_______CADRE  CORESP.: 7  6  6  7  6  6  6  6  6  6  0
Ci= 143 OPERAT. PARAL.: 0  3  0  2  0  0  0  0  1  0  0
_______CADRE  CORESP.: 7  7  6  7  6  6  6  6  6  6  0
CYCLE:Ci= 143 ,CADRE Ni= 7 :Y 4 = 2.71828

Ci= 145     . . .
```

Figure A-15

LE PROGRAMME POUR MONTRER LA TRACE DU MICROPROGRAMME

```
1 REM_"LE PROGRAMME DE SIMULATION DE MICRO_INSTRUCTION"
11 DIM Q(16),Q1(16),Q3(16),Q5(2),Q6(10),D4(8),D7(3),B0(8),I(3)
13 DIM R(16,8),R0(8),S0(8),Z(8),E(8),M3(4),F(8)
15 OPEN "LP:" FOR OUTPUT AS FILE #9
17 OPEN "SY1:MFD" AS FILE #1
19 DIM #1,M(3,256)
21 OPEN "SY0:MMFD" FOR INPUT AS FILE #2
23 DIM #2,M1(256,10)
25 OPEN "SY1:ROM" FOR INPUT AS FILE #3
27 DIM #3,M2(32,3)
28 PRINT #9,"_____TRACE DU MICROPROGRAMME_____"
29 PRINT #9,"A),B) INITIALISATION:" \ PRINT #9,"SEQUENCE DE Y:"
31 REM_TRAITEMENT D'UNE MICRO-INSTRUCTION
33 Y=0 \ T2=0 \ C=1 \ K1=0
35 FOR J=0 TO 9 \ Q6(J)=M1(Y,J) \ NEXT J \ A=Q6(2) \ B=Q6(1)
36 IF K1<15 THEN 38
37 K1=0 \ PRINT #9
38 PRINT #9,Y; \ K1=K1+1
39 REM_1) CHOIX D'ENTREE:
40 ON Q6(8)+1 GO TO 61,41,43,45,47,49,51,53,55
41 E(0)=Q6(0) \ GO TO 61
43 E(0)=D \ GO TO 61
45 FOR J=0 TO 7 \ E(J)=Q1(J) \ NEXT J \ GO TO 61
47 FOR J=0 TO 7 \ E(J)=Q1(J+8) \ NEXT J \ GO TO 61
49 M(Q5(1),Q5(0))=D \ GO TO 61
51 GO TO 61
53 D=M(Q5(1),Q5(0)) \ GO TO 61
55 GO TO 61
60 REM_2) CHOIX DE SOURCE D'OPERANDE:
61 D1=Q6(4)-8 \ IF D1<0 THEN 65
63 U0=1 \ I(0)=D1 \ GO TO 67
65 U0=0 \ I(0)=Q6(4)
67 ON I(0)+1 GO TO 69,73,77,81,85,89,93,97
69 FOR J=0 TO 7 \ R0(J)=R(A,J) \ S0(J)=Q(J) \ NEXT J \ GO TO 113
73 FOR J=0 TO 7 \ R0(J)=R(A,J) \ S0(J)=R(B,J) \ NEXT J \ GO TO 113
77 FOR J=0 TO 7 \ R0(J)=0 \ S0(J)=Q(J) \ NEXT J \ GO TO 113
81 FOR J=0 TO 7 \ R0(J)=0 \ S0(J)=R(B,J) \ NEXT J \ GO TO 113
85 FOR J=0 TO 7 \ R0(J)=0 \ S0(J)=R(A,J) \ NEXT J \ GO TO 113
89 FOR J=0 TO 7 \ R0(J)=E(J) \ S0(J)=R(A,J) \ NEXT J \ GO TO 113
93 FOR J=0 TO 7 \ R0(J)=E(J) \ S0(J)=Q(J) \ NEXT J \ GO TO 113
97 FOR J=0 TO 7 \ R0(J)=E(J) \ S0(J)=0 \ NEXT J \ GO TO 113
111 REM_3) LA FONCTION DE UAL:
113 D2=Q6(3)-8 \ IF D2<0 THEN 117
115 I(1)=D2 \ C0=1 \ GO TO 119
117 I(1)=Q6(3) \ C0=0
119 ON I(1)+1 GO TO 125,129,130,133,145,155,161,161
123 REM_3.1) I'OPERATION "+":
125 F(0)=R0(0)+S0(0)+C0 \ F=F(0) \ GO TO 161
127 REM_3.2) L'OPERATION "-":
129 F(0)=S0(0)-R0(0)+C0 \ F=F(0) \ GO TO 161
130 F(0)=R0(0)-S0(0)+C0 \ F=F(0) \ GO TO 161
131 REM_3.3) L'OPERATION "OU":
133 F=0 \ FOR J=0 TO 7 \ F(J)=R0(J)+S0(J) \ IF F(J)=0 THEN 139
137 F(J)=1 \ GO TO 141
139 F(J)=0
141 F=F+F(J) \ NEXT J \ GO TO 161
143 REM_3.4) L'OPERATION "ET":
145 F=0 \ FOR J=0 TO 7 \ F(J)=R0(J)*S0(J) \ IF F(J)=0 THEN 149
147 F(J)=1 \ GO TO 151
149 F(J)=0
151 F=F+F(J) \ NEXT J \ GO TO 161
```

```
153 REM_3.5) L'OPERATION "R' ET S":
155 F=0 \ FOR J=0 TO 7 \ IF R0(J)=0 THEN 157
156 R0(J)=0 \ GO TO 158
157 R0(J)=1
158 F(J)=R0(J)*S0(J) \ F=F+F(J) \ NEXT J
160 REM_4) LE CHOIX DE DESTINATION:
161 D3=Q6(5)-8 \ IF D3<0 THEN 165
163 U1=1 \ I(2)=D3 \ GO TO 167
165 U1=0 \ I(2)=Q6(5)
167 U4=2*U1+U0 \ ON I(2)+1 GO TO 171,173,175,177,181,201,213,231
171 FOR J=0 TO 7 \ Z(J)=F(J) \ Q(J)=F(J) \ NEXT J \ GO TO 243
173 FOR J=0 TO 7 \ Z(J)=F(J) \ NEXT J \ GO TO 243
175 FOR J=0 TO 7 \ Z(J)=R(A,J) \ R(B,J)=F(J) \ NEXT J \ GO TO 243
177 FOR J=0 TO 7 \ Z(J)=F(J) \ R(B,J)=F(J) \ NEXT J \ GO TO 243
179 REM_DECALAGE A DROITE:
181 L2=F(0) \ L3=Q(0)
183 FOR J=1 TO 7 \ R(B,J-1)=F(J) \ Q(J-1)=Q(J) \ NEXT J
187 ON D4+1 GO TO 189,191,193,195
189 R(B,7)=0 \ Q(7)=0 \ GO TO 243
191 R(B,7)=L2 \ Q(7)=L3 \ GO TO 243
193 R(B,7)=L3 \ Q(7)=L2 \ GO TO 243
195 R(B,7)=F(7) \ Q(7)=L2 \ GO TO 243
201 L2=F(0) \ FOR J=0 TO 7 \ R(B,J-1)=F(J) \ NEXT J
203 ON D4+1 GO TO 205,207,243,209
205 R(B,7)=0 \ GO TO 243
207 R(B,7)=L2 \ GO TO 243
209 R(B,7)=F(7) \ GO TO 243
211 REM_ DECALAGE A GAUCHE:
213 L2=F(7) \ L3=Q(7)
215 FOR J=0 TO 6 \ R(B,J+1)=F(J) \ Q(J+1)=Q(J) \ NEXT J
217 ON D4+1 GO TO 219,221,223,225
219 R(B,0)=0 \ Q(0)=0 \ GO TO 243
221 R(B,0)=L2 \ Q(0)=L3 \ GO TO 243
223 R(B,0)=L3 \ Q(0)=L2 \ GO TO 243
225 R(B,0)=L3 \ Q(0)=0 \ GO TO 243
231 L2=F(7) \ FOR J=0 TO 6 \ R(B,J+1)=F(J) \ NEXT J
233 ON D4+1 GO TO 235,237,243,235
235 R(B,0)=0 \ GO TO 243
237 R(B,0)=L2 \ GO TO 243
241 REM_5) LE CHOIX DE SORTIE:
243 ON Q6(9)+1 GO TO 263,245,247,249,251,253,263,255,257,259
245 FOR J=0 TO 7 \ Q3(J)=Z(J) \ NEXT J \ GO TO 263
247 FOR J=0 TO 7 \ Q3(J+8)=Z(J) \ NEXT J \ GO TO 263
249 D=Z(0) \ GO TO 263
251 Q5(0)=Z(0) \ GO TO 263
253 Q5(1)=Z(0) \ GO TO 263
255 FOR J=0 TO 7 \ Q1(J)=Z(J) \ NEXT J \ GO TO 263
257 FOR J=0 TO 7 \ Q1(J+8)=Z(J) \ NEXT J \ GO TO 263
259 Z9=Z(0) \ GO TO 263
261 REM_6) LE SEQUENCEMENT DU MICROPROGRAMME:
262 REM_6.1) LA PRISE DE LA CONDITION A0:
263 D4(0)=Q6(6) \ FOR J=0 TO 1 \ D4(J+1)=INT(D4(J)/2)
264 D2=D4(J+1)-D4(J)/2 \ IF D2=0 THEN 266
265 B0(J)=1 \ GO TO 267
266 B0(J)=0
267 NEXT J \ T1=2*B0(1)+B0(0) \ IF T1=0 THEN 273
271 A0=0 \ GO TO 281
273 IF F=0 THEN 271
275 A0=1
279 REM_6.2) LA GENERATION DES SIGNAUX CONTROLES:S,FE',PUP:
281 D6=2*Q6(6)+A0
```

```
283 FOR J=0 TO 2 \ D7(J)=M2(D6,J) \ NEXT J
285 REM___6,3) L'ADRESSE DE LA MICRO_INSTRUCTION SUIVANTE:
287 IF D7(1)=1 THEN 297
289 IF D7(0)=1 THEN 293
291 T2=T2-1 \ K0=M3(T2) \ GO TO 297
293 M3(T2)=Y+1 \ T2=T2+1
297 ON D7(2)+1 GO TO 299,301,303,353
299 Y=Y+1 \ GO TO 353
301 Y=Q6(7) \ GO TO 353
303 Y=K0 \ GO TO 353
350 REM_PROGRAMME AUXILIAIRE:
353 IF Y=31 GO TO 380
355 IF Y=54 GO TO 410
357 IF Y=69 GO TO 420
359 IF Y=80 GO TO 430
361 IF Y=96 GO TO 440
363 IF Y=106 GO TO 450
365 IF Y=44 GO TO 460
367 IF Y=124 THEN 463
369 IF Y=144 THEN 465
371 IF Y=163 THEN 467
379 GO TO 35
380 PRINT #9 \ PRINT #9,"LEAS:"; \ J=2
383 D8=M(1,J) \ IF D8=255 THEN 387
385 PRINT #9,D8, \ J=J+1 \ GO TO 383
387 PRINT #9 \ PRINT #9,"LMEU:"; \ J=146
389 IF M(0,J)<>1 THEN 393
391 PRINT #9,J;
393 J=J+2 \ IF M(0,J)=255 THEN 397
395 J=J+2 \ GO TO 389
397 PRINT #9 \ PRINT #9,"_____"
399 PRINT #9 \ PRINT #9,"DEBUT D'UN CYCLE NOUVEAU Ci=";C \ C=C+1
401 PRINT #9,"C) ACQUISITION D'ENTREE:"; \ INPUT Q1(0),Q1(1),Q1(2)
403 PRINT #9,Q1(0);Q1(1);Q1(2)
404 IF Q1(0)+Q1(1)+Q1(2)=3 THEN 472
405 PRINT #9,"SEQUENCE DE Y:" \ K1=0 \ GO TO 35
410 PRINT #9 \ PRINT #9,"E) PRISE LE CONTENU DE LEAC:" \ GO TO 405
420 PRINT #9 \ PRINT #9,"F) CALCUL LA CONDITION D'EVOLUTION:" \ GO TO 405
430 PRINT #9 \ PRINT #9,"G) TRAITEMENT DES ACTIONS IMPUL. :" \ GO TO 405
440 PRINT #9 \ PRINT #9,"H) TRAITEMENT D'ETAPE CLEE-SYNCH. :" \ GO TO 405
450 PRINT #9 \ PRINT #9,"J) CALCUL DES NOUVAUX Mi:" \ GO TO 405
460 PRINT #9 \ PRINT #9,"D) MISE A ZERO LES Ri,Si:" \ GO TO 405
463 PRINT #9 \ PRINT #9,"K) CALCUL C.G.:" \ GO TO 405
465 PRINT #9 \ PRINT #9,"L) AFFECTATION DE SORTIE:" \ GO TO 405
467 PRINT #9 \ PRINT #9,"M) REMISE LA SORTIE ET MASQUE:" \ GO TO 405
471 CLOSE #1 \ CLOSE #2 \ CLOSE #3
472 FOR I=0 TO 20 \ W=I \ PRINT #9 \ NEXT I
473 END
```

Figure A-16

```
_____TRACE DU MICROPROGRAMME_____
A).B) INITIALISATION:
SEQUENCE DE Y:
 0  1  242  191  195  198  243  2  191  195  198  3  4  195  198
 5  2  191  195  198  3  4  195  198  5  2  191  195  198  3
 4  195  198  5  2  191  195  198  3  6  242  191  195  198  243
 7  191  195  198  8  9  10  195  198  11  7  191  195  198  8
 9  10  195  198  11  7  191  195  198  8  12  242  191  195  198
 243  13  191  195  198  14  15  195  198  16  13  191  195  198  14
 15  195  198  16  13  191  195  198  14  15  195  198  16  13  191
 195  198  14  18  213  214  191  195  198  215  216  217  191  195  198
 218  19  242  191  195  198  243  20  191  195  198  21  207  208  195
 198  209  22  23  24  20  191  195  198  21  207  208  195  198  209
 22  25  26  191  195  198  27  28  203  204  205  195  198  206  29
 25  26  191  195  198  27  28  203  204  205  195  198  206  29  25
 26  191  195  198  27
LEAS: 1
LMEU: 146  150
_____

DEBUT D'UN CYCLE NOUVEAU Ci= 1
C) ACQUISITION D'ENTREE: 1  0  0
SEQUENCE DE Y:
 31  32  191  195  198  33  34  35  36  242  191  195  198  243  37
 38  195  198  39  40  41  42  38  195  198  39  40  41  42  38
 195  198  39  40  41
D) MISE A ZERO LES Ri,Si:
SEQUENCE DE Y:
 44  242  191  195  198  243  45  191  195  198  46  47  195  198  48
 49  195  198  50  45  191  195  198  46  47  195  198  48  49  195
 198  50  45  191  195  198  46  51  213  214  191  195  198  215  216
 217  191  195  198  218  52  219  220  191  195  198  221  222  223  191
 195  198  224  53
E) PRISE LE CONTENU DE LEAC:
SEQUENCE DE Y:
 54  55  191  195  198  56  57  58  59  191  195  198  60  61  62
 195  198  63  64  191  195  198  65
F) CALCUL LA CONDITION D'EVOLUTION:
SEQUENCE DE Y:
 69  246  247  191  195  198  248  70  71  244  191  195  198  245  72
 226  191  195  198  227  228  229  230  191  195  198  231  232  233  226
 191  195  198  227  228  229  230  191  195  198  231  234  235  191  195
 198  236  237  241  73
G) TRAITEMENT DES ACTIONS IMPUL. :
SEQUENCE DE Y:
 80  81  82  195  198  83  246  247  191  195  198  248  84  85  199
 200  201  195  198  202  86  83  246  247  191  195  198  248  84  87
 246  247  191  195  198  248  88  89  203  204  205  195  198  206  90
 87  246  247  191  195  198  248  88  89  203  204  205  195  198  206
 90  87  246  247  191  195  198  248  88  91  246  247  191  195  198
 248  92  93  210  211  195  198  212  94  91  246  247  191  195  198
 248  92  76  246  247  191  195  198  248  77  78  70  71  244  191
 195  198  245  72  226  191  195  198  227  228  229  230  191  195  198
 231  234  235  191  195  198  236  240  73  74  246  247  191  195  198
 248  75  74  246  247  191  195  198  248  75  74  246  247  191  195
 198  248  75  74  246  247  191  195  198  248  75  74  246  247  191
 195  198  248  75  74  246  247  191  195  198  248  75  74  246  247
 191  195  198  248  75  74  246  247  191  195  198  248  75  76  246
 247  191  195  198  248  77
H) TRAITEMENT D'ETAPE CLEE-SYNCH. :
SEQUENCE DE Y:
```

```
 54  55  191  195  198  56  57  58  59  191  195  198  60  61  62
 195  198  63  64  191  195  198  65  66  203  204  205  195  198  206
 67  63  64  191  195  198  65
F) CALCUL LA CONDITION D'EVOLUTION:
SEQUENCE DE Y:
 69  246  247  191  195  198  248  70  71  244  191  195  198  245  72
 226  191  195  198  227  228  229  230  191  195  198  231  232  233  226
 191  195  198  227  228  229  230  191  195  198  231  232  233  226  191
 195  198  227  228  241  73
G) TRAITEMENT DES ACTIONS IMPUL. :
SEQUENCE DE Y:
 80  81  82  195  198  83  246  247  191  195  198  248  84  85  199
 200  201  195  198  202  86  83  246  247  191  195  198  248  84  87
 246  247  191  195  198  248  88  89  203  204  205  195  198  206  90
 87  246  247  191  195  198  248  88  89  203  204  205  195  198  206
 90  87  246  247  191  195  198  248  88  91  246  247  191  195  198
 248  92  93  210  211  195  198  212  94  91  246  247  191  195  198
 248  92  76  246  247  191  195  198  248  77
H) TRAITEMENT D'ETAPE CLEE-SYNCH. :
SEQUENCE DE Y:
 96  97  191  195  198  98  104
E) PRISE LE CONTENU DE LEAC:
SEQUENCE DE Y:
 54  55  191  195  198  56
J) CALCUL DES NOUVAUX Mi:
SEQUENCE DE Y:
 106  242  191  195  198  243  107  191  195  198  108  109  110  191  195
 198  111  112  191  195  198  113  114  118  195  198  119  191  195  198
 120  121  210  211  195  198  212  122  107  191  195  198  108  109  110
 191  195  198  111  112  191  195  198  113  114  115  116  195  198  117
 122  107  191  195  198  108
K) CALCUL C.G.:
SEQUENCE DE Y:
 124  125  126  195  198  127  242  191  195  198  243  128  244  191  195
 198  245  129  130  131  191  195  198  132  133  134  244  191  195  198
 245  135  226  191  195  198  227  228  229  230  191  195  198  231  234
 235  191  195  198  236  240  136  140  246  247  191  195  198  248  141
 199  200  201  195  198  202  142  139  131  191  195  198  132
L) AFFECTATION DE SORTIE:
SEQUENCE DE Y:
 144  145  242  191  195  198  243  146  147  148  191  195  198  149  150
 151  152  191  195  198  153  150  151  152  191  195  198  153  150  151
 152  191  195  198  153  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  157  158  159  160  161  146
 147  148  191  195  198  149
M) REMISE LA SORTIE ET MASQUE:
SEQUENCE DE Y:
 163  242  191  195  198  243  164  191  195  198  165  166  195  198  167
 191  195  198  168  166  195  198  167  191  195  198  168  166  195  198
 167  191  195  198  168  169  164  191  195  198  165  170  171  249  250
 191  195  198  251  252  191  195  198  253  172  191  195  198  173  174
 175  176  172  191  195  198  173  174  175  176  172  191  195  198  173
 174  175  177  178  179  180  249  250  191  195  198  251  252  191  195
 198  253  181  195  198  182  183  181  195  198  182  183  181  195  198
 182  184  213  214  191  195  198  215  216  217  191  195  198  218  183
 219  220  191  195  198  221  222  223  191  195  198  224  186  187  191
 195  198  188  207  208  195  198  209  189  190  186  187  191  195  198
 188  207  208  195  198  209  189  190  186  187  191  195  198  188  207
```

```
 96  97  191  195  198  98  104
E) PRISE LE CONTENU DE LEAC:
SEQUENCE DE Y:
 54  55  191  195  198  56
J) CALCUL DES NOUVAUX Mi:
SEQUENCE DE Y:
 106  242  191  195  198  243  107  191  195  198  108  109  110  191  195
 198  111  112  191  195  198  113  114  115  116  195  198  117  122  107
 191  195  198  108  109  110  191  195  198  111  112  191  195  198  113
 114  115  118  195  198  119  191  195  198  120  122  107  191  195  198
 108
K) CALCUL C.G.:
SEQUENCE DE Y:
 124  125  126  195  198  127  242  191  195  198  243  128  244  191  195
 198  245  129  130  131  191  195  198  132  133  134  244  191  195  198
 245  135  226  191  195  198  227  228  229  230  191  195  198  231  232
 233  226  191  195  198  227  228  229  230  191  195  198  231  234  235
 191  195  198  236  240  136  140  246  247  191  195  198  248  141  199
 200  201  195  198  202  142  139  131  191  195  198  132
L) AFFECTATION DE SORTIE:
SEQUENCE DE Y:
 144  145  242  191  195  198  243  146  147  148  191  195  198  149  150
 151  152  191  195  198  153  150  151  152  191  195  198  153  150  151
 152  191  195  198  153  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  155  156  154  155  156  154
 155  156  154  155  156  154  155  156  154  157  158  159  160  161  146
 147  148  191  195  198  149
M) REMISE LA SORTIE ET MASQUE:
SEQUENCE DE Y:
 163  242  191  195  198  243  164  191  195  198  165  166  195  198  167
 191  195  198  168  166  195  198  167  191  195  198  168  166  195  198
 167  191  195  198  168  169  164  191  195  198  165  170  171  249  250
 191  195  198  251  252  191  195  198  253  172  191  195  198  173  174
 175  176  172  191  195  198  173  174  175  176  172  191  195  198  173
 174  175  177  178  179  180  249  250  191  195  198  251  252  191  195
 198  253  181  195  198  182  183  181  195  198  182  183  181  195  198
 182  184  213  214  191  195  198  215  216  217  191  195  198  218  185
 219  220  191  195  198  221  222  223  191  195  198  224  186  187  191
 195  198  188  207  208  195  198  209  189  190  186  187  191  195  198
 188  207  208  195  198  209  189
LEAS: 21
LMEU: 150
-------------------------------------------------------------------------

DEBUT D'UN CYCLE NOUVEAU Ci= 2
C) ACQUISITION D'ENTREE: 0  0  1
SEQUENCE DE Y:
 31  32  191  195  198  33  34  35  36  242  191  195  198  243  37
 38  195  198  39  40  41  42  38  195  198  39  40  41  42  38
 195  198  39  40  41
D) MISE A ZERO LES Ri,Si:
SEQUENCE DE Y:
 44  242  191  195  198  243  45  191  195  198  46  47  195  198  48
 49  195  198  50  45  191  195  198  46  47  195  198  48  49  195
 198  50  45  191  195  198  46  51  213  214  191  195  198  215  216
 217  191  195  198  218  52  219  220  191  195  198  221  222  223  191
 195  198  224  53
E) PRISE LE CONTENU DE LEAC:
SEQUENCE DE Y:
```

```
 208   195   198   209   189
LEAS: 33          1
LMEU: 146
```

```
DEBUT D'UN CYCLE NOUVEAU Ci= 3
C) ACQUISITION D'ENTREE: 1   1   1
```

:
:
:

Figure A-17

BIBLIOGRAPHIE

1/ S. THELLIEZ et J.-M. TOULOTTE , " GRAFCET et logique industrielle programmée", Editions Eyrolle, 1980

2/ E. DACLIN et M. BLANCHARD , "Synthèse des systèmes logiques", CEPADUES_ Editions, 1976

3/ ---, "STARAN parallel processor system hardware", Service Formation "LE TRAITEMENT PARALLELE", Ecole de l'Iria, Volume2,1978,PP.293-298

4/ G. H. BARNES et al, "The ILLIAC IV computer", IEEE Trans. Comput., vol. C-17, Aug. 1968, PP.746-757

5/ D. COMTE, N. HIFDI and J.C. SYRE, "The data driven LAU Multiprocessor system: results and perspectives", Information processing 80, S.H.Lavington (ed.), IFIP 1980, PP.175-180

6/ T. BABA et al, "MUNAP-A two-level microprogrammed multiprocessor architecture for nonnumeric processing", Information processing 80, S.H. Lavington (ed.), IFIP 1980, PP.169-174

7/ J.P. MEINADIER, "Structure et fonctionnement des ordinateurs", Larousse Editions, 1979

8/ PLAN, "Architecture pipe line", Service Formation " LE TRAITEMENT PARALLELE", Ecole de l'Iria, Volume 1, 1978 , PP.219-264

9/ D.-J. DAVIP et J.-L.DESHAMPS, "Programmer en PASCAL", Editions du P.S.I., 1980

10/ R.H. CAMPBELL, "The specification of process synchronisation by path expressions", Technical Report, University of Newcastle Upon Tyne, England, Dec. 1973

11/ ---, "La programmation et les langages parallèles" , Service Formation " LE TRAITEMENT PARALLELE ", Ecole de l'Iria, Volume 1, 1978, PP.1-31

12/ T.AGERWALA , "Some Extended semaphore primitives", Acta Infomatica 8.201-220, University of Texas Austin, 1977, 20P.

13/ B. DESCOTES-GENON, R. DAVID, " Arbitres asynchrones ", Labolatoire d'Automatique de Grenoble E.N.S.I.E.G.,1980, 16P.

14/ P.E. LAVER , "Abstract specification of ressource accessing disciplines: adequacy, starvation, priority and interrupts", University of Newcastle Upon Tyne, England, 1977

15/ B.V. HOWARD, " The manipulation and logical implementation of computation schemata for parallel processes, ", Journées AFCET, Paris 23-24 MARS 1977, PP.91-111

16/ K. QUOTB, " Systèmes Multi-automates programables:réalisation mtérielle et programmation a partir d'une description GRAFCET ", Thèse de Docteur-Ingénieur, L'Université de Lille 1, Juin 1981

17/ ---, " The Am2900 Family Data Book ", Advanced Micro Dvices, Inc.,1976

18/ J.R. MICK and J. BRICK , " Microprogramming Handbook ", Advanced Micro Devices, Inc., 1976

19/ G. ZIMMERMANN, " Cost performance analysis and optimisation of highly parallel computer structure: first results of a structured Top-Down method ", Proc. of 4th Intern. Symp. on C.H.D.L., Oct. 8-9 1979, PP.33-39

20/ A. STRAUFFER, " Méthode de synthèse des systèmes digitaux, première partie: l'unité de traitement ", Bull. ASE/UCS 71(1980), No 3, PP.143-150

21/ J.-M. TOULOTTE, " Dispositif de commande en temps réel ", © Bordas 1975

22/ ---, "Am2900 Evaluation and Learning KIT Instruction Manual ", Advanced Micro Devices, Inc., 1976

23/ H. EVEKING , " The application of register transfer languages in the design of real hardware ", Proc. of the 4th Intern. Symp. on C.H.D.L. , Oct. 1979, PP.169-177

24/ R.M. KLINE, " Digital Computer Design ", Prentice-Hall, Inc., 1977, PP. 241-253

25/ D.M. SCHULER, "A language for modeling the functional and timing characteristics of complex digital computers for logic simulation ", Proc. of the 4th Intern. Symp. on C.H.D.L., Oct. 1979 , PP.54-59

26/ H. ANLAUFF, al , " PHPL-A new computer hardware description language for modular description of logic and timing ", Proc. of the 4th Intern. Symp. on C.H.D.L. , Oct. 1979, PP.124-130