
Architecture des systèmes à microprocesseurs

Maryam Siadat & Camille Diou

I – Introduction

L'architecture d'un ordinateur constitue l'ensemble des sous-systèmes réalisant différentes fonctionnalités, la manière de les relier et de les faire communiquer.

Au premier sens du terme, une architecture informatique est définie par des paramètres tels que les instructions du microprocesseur, les jeux de registres, la méthodologie de gestion de la mémoire, et d'autres fonctions ...

I.1 L'architecture d'un ordinateur simple

Un simple ordinateur est composé de :

- mémoire (RAM/ROM) ;
- un CPU ;
- quelques circuits d'entrée/sortie.

Les trois parties sont connectées par les trois bus (adresse, données, contrôle) créés par le CPU. Le CPU est le maître unique dans les ordinateurs simples, et c'est lui qui contrôle tout le système.

La séquence des instructions (ou programme) est stockée en mémoire à des emplacements successifs et en code binaire.

Le rôle du CPU se décompose en quatre étapes :

1. aller chercher l'instruction suivante en mémoire → cycle de chargement ;
2. la décoder pour déterminer l'action qu'il doit effectuer ;
3. l'exécuter → cycle d'exécution ;
4. revenir à l'étape 1.

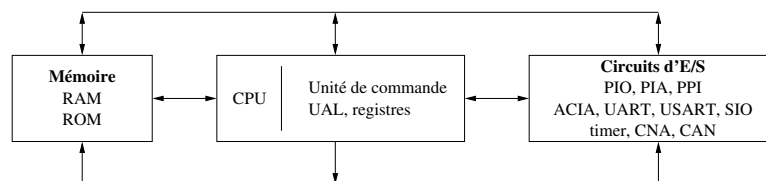


FIG. I.1: Schéma synoptique d'un ordinateur simple

1) L'unité centrale : (UC, CPU, MP)

Elle se compose essentiellement de l'unité de commande, de l'UAL et d'un ensemble de registres. L'UC est organisée le long d'un bus (bus interne du microprocesseur, cf figure I.2).

Remarque : les registres de l'UC n'ont pas tous la même taille.

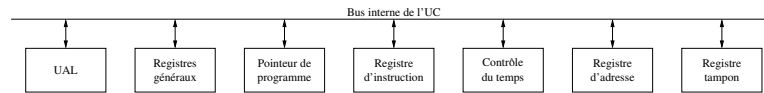


FIG. I.2: Schéma synoptique d'une unité centrale

1.a) L'UAL

Elle permet d'effectuer les additions et soustractions binaires, OR, AND, XOR...

Exemple de circuit : 74LS181, UAL 4 bits (cf figure I.3).

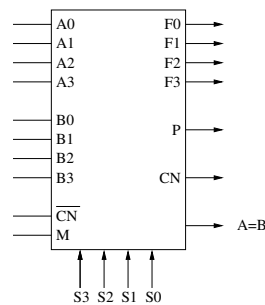


FIG. I.3: Schéma d'un 74LS181

Ce circuit réalise des fonctions arithmétiques et logiques sur A et B, avec le résultat dans F. L'opération est déterminée par M et par les entrées de sélection S0, S1, S2, et S3 :

- M=1 → 16 fonctions logiques ;
- m=0 → 16 fonctions arithmétiques.

L'intérêt d'une UAL c'est qu'elle peut être programmée avec des instructions binaires : c'est un circuit programmable. L'UAL est la partie la plus importante d'un processeur, donc d'un ordinateur.

Réalisation d'une UAL :

- elles sont souvent réalisées à l'aide des réseaux logiques programmables (PLA) ;
- avec l'arrivée des circuits LSI et VLSI, des réalisations très structurées ont été possibles à l'aide de MUX (sélecteurs de données).

Exemple : additionneur complet de 1 bit à l'aide de 2 MUX à 3 entrées de sélection (cf figure I.4).

Remarques :

1. pour réaliser une UAL traitant des mots de n bits, il suffit de brancher en cascade n paires de sélecteurs de données (exemple additionneur à 4 bits) ;
2. pour réaliser une autre opération, il suffit de changer les entrées des MUX.

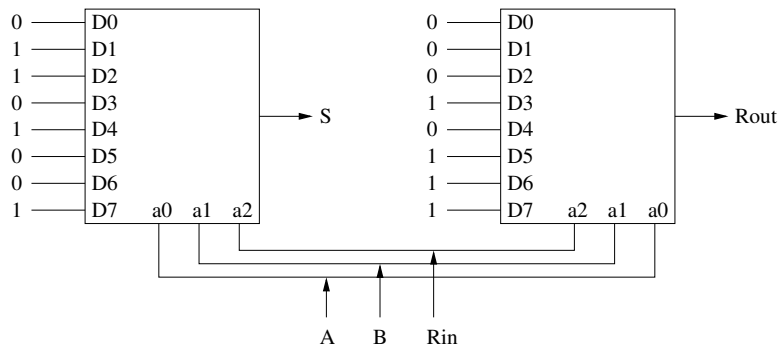


FIG. I.4: Additionneur complet

1.b) L'unité de commande

Son rôle est :

- d'envoyer des signaux de contrôle pour connecter les registres au bus ;
- de superviser le fonctionnement de l'UAL ;
- de donner des signaux d'horloge à l'ensemble de l'ordinateur.

Toutes les actions de l'unité de commande concernent les cycles de chargement et d'exécution des instructions. Par exemple, il y a besoin de signaux de contrôle pour relier les registres au bus, pour superviser le fonctionnement de l'UAL, et pour donner des signaux d'horloge à l'ensemble de l'ordinateur. La plupart de ces signaux provient de la section de contrôle de l'UC (voir la réalisation des séquenceurs).

Cycle de chargement (cf figure I.5) :

Les instructions sont lues une par une, puis décodées et exécutées.

Pour aller chercher une instruction en mémoire, il faut exécuter les opérations suivantes :

1. charger le contenu du pointeur de programme dans le registre d'adresse mémoire qui est relié au bus ;
2. demander à la mémoire de lire la donnée, et de la placer sur le bus (*signal read memory*) ;
3. ranger la valeur lue sur le bus de données, dans le buffer de l'UC ;
4. transférer le buffer dans le registre d'instruction ;
5. incrémenter le pointeur de programme d'une unité pour aller chercher l'instruction suivante.

Pour effectuer ces opérations, il faut envoyer un certain nombre de signaux de contrôle aux divers registres de l'UC et à la mémoire. Par exemple, pour la première opération, le système de contrôle doit fermer toutes les sorties de registres en direction du bus interne, ouvrir la sortie du pointeur de programme en direction du bus, et charger

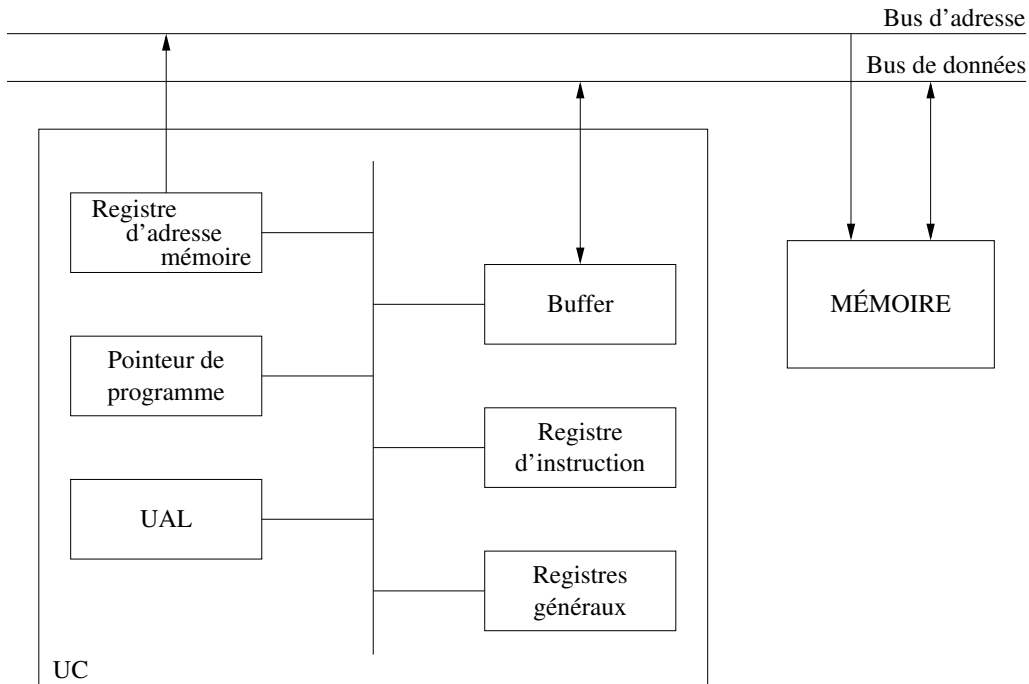


FIG. I.5: Unité de commande

le contenu du bus dans le registre d'adresse mémoire. Il faut évidemment envoyer tous ces signaux dans un ordre correct et avec un décalage temporel correct entre eux.

Cycle d'exécution :

L'ensemble des opérations précédentes se répète à chaque cycle de chargement. Mais, selon l'architecture et la teneur de chaque instruction, la façon dont le cycle d'exécution se déroule est très variable.

Prenons comme exemple l'addition de l'accumulateur et d'un autre registre avec le résultat dans l'accumulateur.

Puisque les registres sont dans l'UC, la séquence d'exécution est :

1. transfert du contenu de l'accumulateur dans l'UAL ;
2. transfert du contenu du registre adressé vers l'autre entrée de l'UAL ;
3. déclenchement de l'addition (en envoyant les signaux nécessaires dans l'UAL) ;
4. retour du résultat dans l'accumulateur.

Remarque : si une instruction fait appel à la mémoire le cycle est plus long.

1.c) La micro-programmation

On peut réaliser le système de commande à l'aide des techniques classiques portant sur la logique séquentielle (séquenceur câblé). Mais il est de plus en plus fréquent d'opérer à l'aide de la micro-programmation (séquenceur micro-programmé), c'est-à-dire que l'ensemble des opérations décrites précédemment (cycles de chargement et d'exécution) peuvent être considérées comme un programme qui tourne sur un processeur très simple. En programmant à ce niveau, on demande que les instructions soient exécutées dans un ordre correct et que chaque instruction donne la combinaison correcte des signaux de contrôle. En fait la section de commande est « micro-microprocesseur ».

I.2 Les processeurs spécialisés

1) Micro-contrôleurs

Ils contiennent un CPU, de la RAM, de la ROM, quelques ports d'E/S parallèles, des ports séries, des compteurs programmables (timers), des CAN/CNA, des interfaces pour réseaux de terrain ...

Ils sont en général utilisés pour contrôler des simples machines (appareils électroménagers, lecteurs de carte à puce...)

Exemple de circuits :

- 80C186XX (80186, 16 bits, Intel)
- 68HC11, 68HC12 (6809, 8 bits, Motorola)
- 68HC16 (68000, 16 bits, Motorola)
- C167XX (Infineon, ex Siemens)

2) Digital Signal Processor (sorties)

Ce sont des processeurs dédiés aux traitements des signaux numériques. Une architecture particulière leur permet un traitement efficace des fonctions complexes telles que FFT, convolution, filtrage numérique ...

Exemples :

- TMS320 (Texas Instrument)
- 2100 et 21000 (Analog Device)
- 56000 (Motorola)

3) Processeurs de traitement d'image

4) Processeurs spécialisés d'entrées/sorties

1.2. Les processeurs spécialisés

II – Les différentes architectures

II.1 Architecture uniprocasseur

1) Architecture Von Neumann (1946)

Architecture conventionnelle, la plus utilisée dans le domaine des ordinateurs. Elle repose sur quatre entités principales (cf figure II.1) :

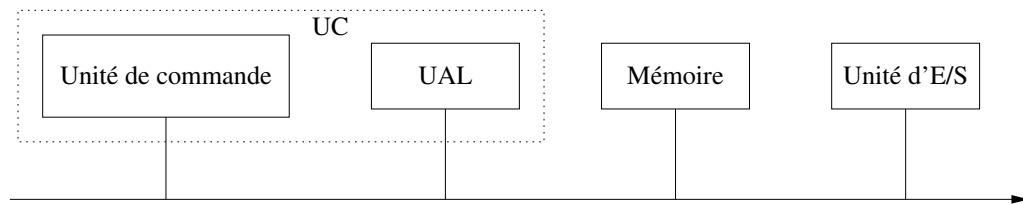


FIG. II.1: Architecture de Von Neumann

Un seul processeur (unité centrale) travaille de manière séquentielle sur des informations en mémoire qui constituent à la fois les données et les programmes : une seule unité de commande traitant une seule séquence d'instruction (*Single Instruction Stream*) et une seule unité d'exécution (UAL) traitant une unique séquence de données (*Single Data Stream*). Cette architecture est donc appelée SISD (*Single Instruction Single Data*). Un bus unique relie les différents modules.

Cette architecture à commande unique constitue un frein au traitement parallèle.

2) Architecture de Harvard (1940, la première employée pour les ordinateurs)

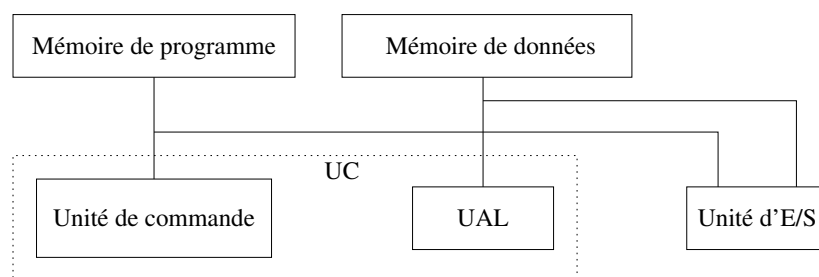


FIG. II.2: Architecture de Harvard

Elle est caractérisée par la séparation des données et programmes en mémoire. L'accès à chacun s'effectue par un bus séparé, différent et indépendant. Donc un accès si-

multané aux instruction et aux données est possible, ce qui implique une exécution plus rapide.

Cette architecture est abandonnée sur les ordinateurs universels en raison de sa complexité matérielle, mais elle est avantageuse pour les systèmes de traitement numérique de s signaux. Le 4004 (premier processeur Intel) relève de cette architecture, de même que la plupart des processeurs DSP.

3) Temps partagé ou multitâche

C'est l'exécution simultanée de plusieurs programmes (par exemple compiler un programme pendant qu'on travaille sur un manuscrit sous traitement de texte).

Cette possibilité repose sur une technique de commutation rapide entre les différents programmes. Le temps du processeur est alors partagé en petits incréments (par exemple 20 ms). Les tâches sont exécutées les unes après les autres, chacune pendant ce temps. Ce fonctionnement peut être réalisé par un logiciel (par exemple Linux), mais c'est plus efficace lorsque cela est directement géré par l'unité centrale.

Inconvénients :

- si le processeur ne marche pas, toutes les tâches sont en attente ;
- il y a saturation si le nombre d'utilisateurs ou de tâches augmente.

II.2 Architectures parallèles

1) Architectures avec cadencement des données

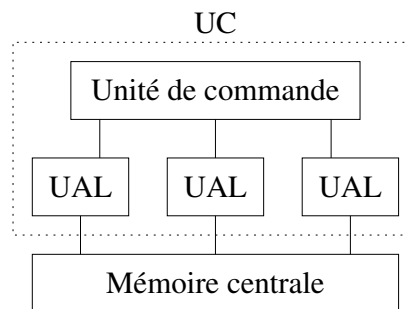


FIG. II.3: Architecture avec cadencement de données

Cette architecture à flot de données s'oppose à une architecture traditionnelle dite à flot d'instructions. Cette configuration est appelée SIMD (*Single Instruction Multiple Data*) et ce type de processeur « processeur vectoriel ».

On peut classer dans cette catégorie les « processeurs vectoriels » disposant des instructions vectorielles (exemple d'application : addition de deux vecteurs en virgule

flottante). L'unité de commande envoie une instruction à toutes les UAL qui exécutent l'instruction pas à pas sur des données locales. Le réseau d'interconnexions permet aux résultats d'être envoyés vers une autre UAL qui pourra les utiliser comme opérande dans une instruction suivante. Les processeurs DSP utilisent cette architecture.

2) Architecture multiprocesseurs (MIMD)

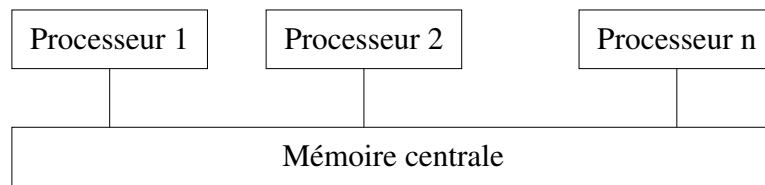


FIG. II.4: Architecture MIMD

Plusieurs processeurs partagent la même mémoire.

Chaque unité centrale (processeur) dispose de son propre programme indépendant.

Plusieurs distributions architecturales sont possibles :

- architecture centralisée (cf figure II.5) :

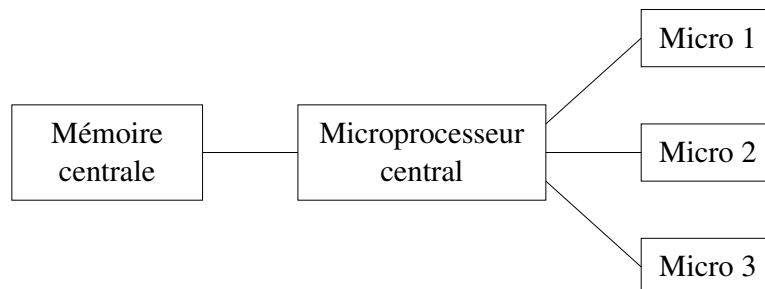


FIG. II.5: Architecture centralisée

- architecture horizontale (cf figure II.6) :

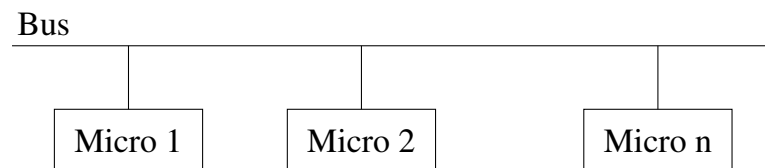


FIG. II.6: Architecture horizontale

chaque microprocesseur est soit un processeur central, soit un processeur spécialisé (E/S par exemple).

Chaque processeur constitue une ressource pour le système.

- structure hybride (cf figure II.7) :

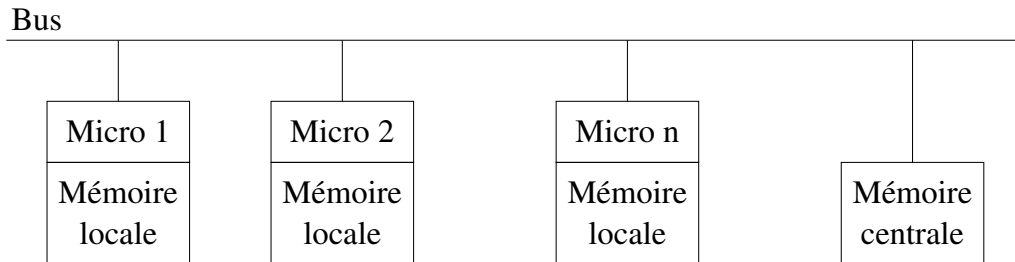


FIG. II.7: Architecture hybride

Dans les structures précédentes la charge du bus (transfert de données, communication entre les processeurs) freine l'efficacité de l'architecture multiprocesseurs. L'introduction de mémoire locale associée à chaque processeur permet d'éliminer les accès à la mémoire centrale

- hiérarchie de bus (cf figure II.8) :

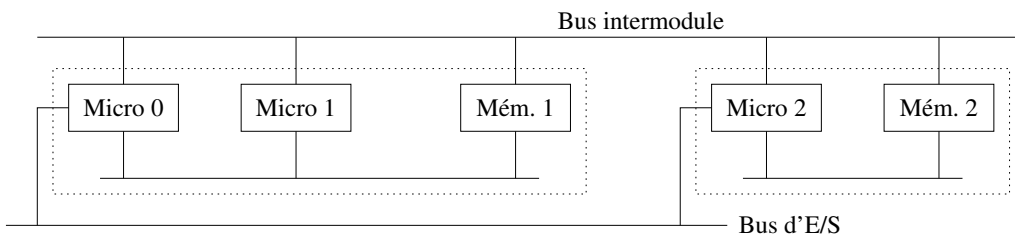


FIG. II.8: Hiérarchie de bus

Une autre méthode de décharger le bus consiste à réaliser une hiérarchie de bus. Celle-ci peut être étendue aux E/S.

3) Processeur multi-unités de traitement

Cette architecture n'a de sens que si le temps de chargement et de décodage est faible par rapport à la durée de son exécution.

4) Processeur pipeline

L'idée est inspirée de l'organisation du travail à la chaîne : cette technique permet d'effectuer davantage de travail par unité de temps.

chaque étape de l'exécution de l'instruction (chargement, décodage, recherche des données) se fait dans une unité séparée :

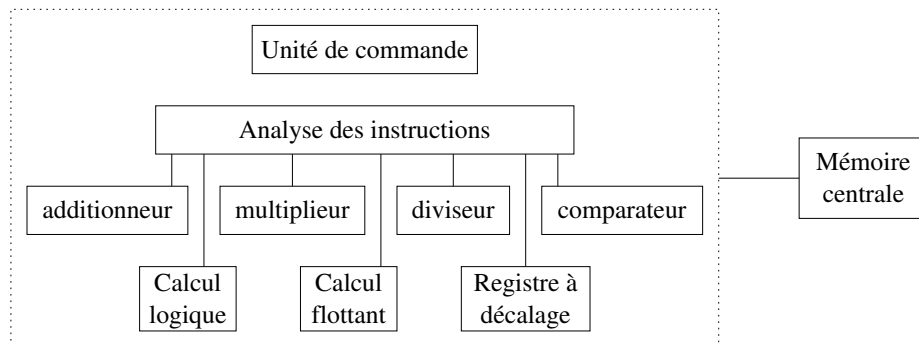


FIG. II.9: Processeur multi-unités de traitement

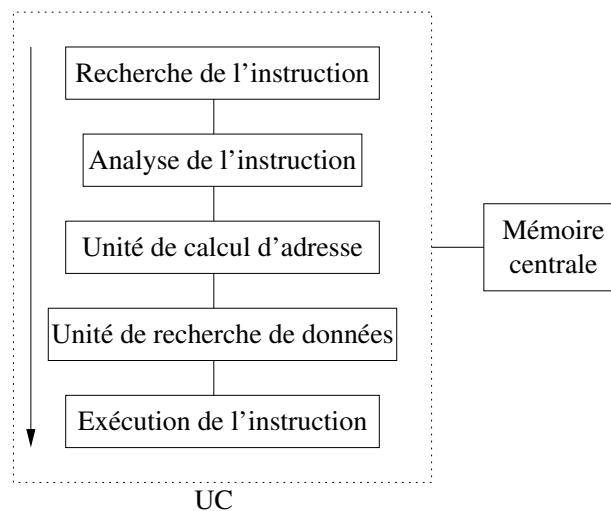


FIG. II.10: Processeur pipeline

- la première est chargée dans la première unité ;
- la deuxième unité commence à décoder l'instruction pendant que la première travaille à charger la deuxième instruction ;
- un peu plus tard, la troisième unité travaille sur le calcul d'adresse pendant que la deuxième unité décode la deuxième instruction, et que la première unité charge l'instruction suivante (exemple : IBM 360/195, CDC STAR).

II.3 Architecture RISC (*Reduced Instruction Set Computer*)

Architecture dans la quelle les instructions sont en nombre réduit (chargement, branchement, appel sous-programme) et elle sont fréquemment utilisées. Le but est d'éliminer les instructions rarement employées et de consacrer les ressources matérielles à exécuter les instructions relativement simples en un cycle d'horloge et à émuler les autres instructions à l'aide des séquences basées sur les instructions élémentaires. On trouve donc une meilleure performance à une vitesse donnée (le gain en performance envisageable est important mais dépend de la qualité du compilateur).

Caractéristiques des machines RISC :

1. un cycle d'horloge au maximum pour l'exécution d'une instruction ;
2. une dépendance plus forte par rapport aux compilateurs ;
3. toutes les instructions ont la même longueur ;
4. une utilisation systématique des registres (généralement plus nombreux que sur les processeurs CISC) ;
5. séquenceur câblé plutôt que programmé ;
6. une limitation des accès mémoire aux deux opérations LOAD et STORE.

Stations de travail équipées de processeurs à architecture RISC : SUN, HP, IBM.

Processeurs RISC : Alpha (DEC), PowerRISC (IBM), SPARC (SUN), PA-RISC (HP).

II.4 Architecture CISC (*Complex Instruction Set Computer*)

C'est une architecture avec un grand nombre d'instructions. Le processeur doit exécuter des tâches complexes par instruction unique. Donc, pour une tâche donnée, une machine CISC exécute un petit nombre d'instructions mais chacun nécessite un plus grand nombre de cycles d'horloge (Pentium et PowerPC). Actuellement les deux technologies convergent : les processeurs CISC (Pentium par exemple) utilisent des instructions de plus en plus simples et exécutent parfois plusieurs instructions en un cycle d'horloge.

III – Les mémoires à semi-conducteur

La mémoire contient le programme (sous forme d'instructions codées en binaire) et les données (informations utilisables au cours de l'exécution).

On trouve la mémoire soit dans un CI soit dans une partie du microprocesseur.

La mémoire est un ensemble de cellules de même taille (en général 8 bits). Chaque cellule est atteignable par une adresse unique qui lui est propre. On dit alors que la mémoire est à accès aléatoire.

À l'intérieur de chaque circuit mémoire un décodeur active le mot sélectionné, une logique de contrôle définit le sens des échanges. Cette adresse permet à tous les sous-systèmes de l'ordinateur d'accéder à l'information qui y est stockée.

La structure de base (cf figure III.1) :

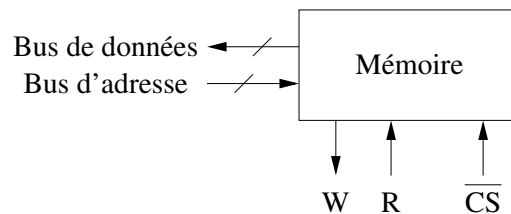


FIG. III.1: Structure de base d'une mémoire

Deux opérations sont possibles : lecture ou écriture.

Les caractéristiques de choix :

Taille physique, vitesse (environ quelques dizaines de ns), consommation et niveaux logiques (caractéristiques idéales : faible prix, vitesse élevée).

III.1 Classification des mémoires

1) Mémoires vives : RAM

Elles sont souvent qualifiées de RAM (*Random Access Memory*), mais les mémoires mortes sont également à accès aléatoire, donc de type RAM.

Elle nécessite une énergie pour stocker et garder les données. Il existe deux types de mémoires vives : statiques et dynamiques.

1.a) RAM Statique (SRAM)

La réalisation interne est essentiellement faite par une matrice de bascules :

- m bascule sde large, où m est la taille du mot ;
- 2^n bascules de long, où 2^n est la capacité en mots de la mémoire.

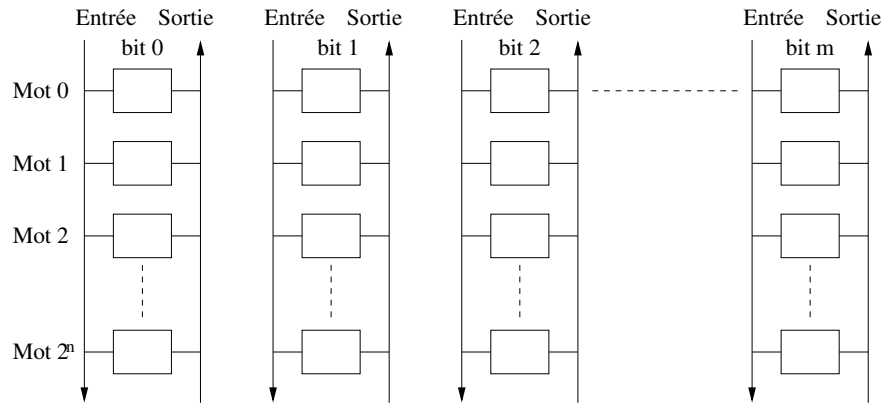


FIG. III.2: Matrice organisée en mots avec sélection linéaire : 2^n mots $\rightarrow 2^n$ lignes, m bits $\rightarrow m$ colonnes.

Cette réalisation est simple mais elle nécessite un décodeur interne à 2^n sorties, donc une surface de boîtier importante. Elle a cependant un accès rapide.

Remarque : il existe aussi la méthode de décodage à deux niveaux (*two level decoding*).

1.b) RAM dynamique (DRAM)

Les 1 et les 0 sont stockés comme une charge électrique dans un condensateur. Pour maintenir l'information il faut la réécrire régulièrement (rafraîchissement environ toutes les 2 ms), car le condensateur se décharge dans les résistances de fuite.

Une opération de lecture rafraîchit une ligne de la mémoire, par conséquent toute la mémoire est rafraîchie en 2^n lectures.

On peut utiliser des composants tels que des dRAM *refresh controller*, mais depuis l'utilisation de technologie VLSI, la plupart des DRAM ont leur propre circuit de rafraîchissement.

Les DRAM sont constituées selon une matrice (2^n lignes et 2^n colonnes) dont l'adressage des cellules se fait en deux temps : adressage des lignes puis adressage des colonnes.

L'avantage des DRAM par rapport aux SRAM est que leur surface est environ quatre fois plus petite à capacité identique.

Exemple de boîtier : Intel 2118, 16384×1 bits, 16 kbits (cf figure III.3).

Remarques :

- les DRAM sont des boîtiers de $x \times 1$ bits.
- SRAM : il existe des boîtiers de 64 Mbits configurables en 8, 16 ou 32 bits ;
- DRAM : bientôt des 16 Mbits.

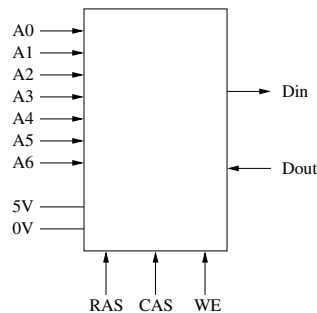


FIG. III.3: Boitier Intel 2118

Actuellement les SRAM sont surtout utilisées pour l'antémoire et les DRAM pour la mémoire principale. Il existe :

- **DRAM FPM** (*fast page mode* asynchrone jusqu'à 25 ou 33 MHz ;
À une certaine période, la mémoire FPM était la forme la plus courante de DRAM dans les ordinateurs. Elle était si fréquente que l'on parlait simplement de « DRAM », en oubliant « FPM ». La technologie de mémoire FPM offrait un avantage sur les précédentes car elle permettait un accès plus rapide aux données situées sur une même rangée.[\[RAM\]](#)
- **EDO DRAM** (*extended data out*) jusqu'à 66 MHz ;
Apparue en 1995, la mémoire EDO représentait une nouvelle innovation dans ce domaine. Similaire à la FPM, elle comportait pourtant une légère modification, autorisant des accès mémoire consécutifs bien plus rapides. Le contrôleur de mémoire gagnait du temps en supprimant quelques étapes dans le processus d'adressage. Avec une EDO, l'adressage de la mémoire par la CPU s'effectue à une vitesse supérieure de 10 à 15 % par rapport à une FPM.[\[RAM\]](#)
- **SDRAM** : DRAM synchrone jusqu'à 200 MO/s
Fin 1996, les SDRAM ont commencé à équiper les systèmes. À la différence des technologies antérieures, la SDRAM se synchronise elle-même avec la CPU. Ainsi, le contrôleur de mémoire connaît le cycle d'horloge exact où les données seront disponibles. Donc, la CPU n'attend plus entre les accès mémoire. Les puces SDRAM bénéficient des modes « entrelacement » et « rafale », qui accélèrent également la recherche en mémoire. Les modules SDRAM sont disponibles en différentes fréquences, assurant ainsi la synchronisation avec la vitesse d'horloge du système où elles sont implantées. Par exemple, une SDRAM PC66 est cadencée à 66 MHz, une SDRAM PC100 à 100 MHz, une SDRAM PC133 à 133 MHz, et ainsi de suite... Des valeurs supérieures tels que 200 MHz et 266 MHz sont actuellement en cours de développement.[\[RAM\]](#)
- **DDR SDRAM** : fonctionne sur les deux fronts de l'horloge
La DDR SDRAM représente la génération suivante de la technologie SDRAM. Elle permet à la puce mémoire d'effectuer des transactions à la fois durant la phase montante et durant la phase descendante du cycle d'horloge. Par exemple, avec une DDR SDRAM, un bus

mémoire à 100 ou 133 MHz gère un débit de données réel de 200 MHz ou 266 MHz. Des systèmes dotés de DDR SDRAM sont apparus fin de l'an 2000.[RAM]

– **RAMBUS**

Direct Rambus est une nouvelle norme d'architecture et d'interface de DRAM qui représente un défi par rapport à la configuration classique de la mémoire principale. Comparée aux anciennes technologies, Direct Rambus est extraordinairement plus rapide. Elle est capable de transférer les données à une vitesse atteignant 800 MHz via un bus étroit (16 bits), appelé Direct Rambus Channel. Cette vitesse d'horloge élevée est rendue possible grâce à un dispositif "double horloge," qui autorise les transactions à la fois durant la phase montante et durant la phase descendante du cycle d'horloge. Donc chaque dispositif de mémoire d'un module RDRAM génère une bande passante atteignant 1,6 giga-octet par seconde - le double de celle disponible sur les SDRAM 100 MHz courantes.[RAM]

Autres technologies de mémoires [RAM]

– **ESDRAM : Enhanced SDRAM**

Afin d'augmenter la vitesse et l'efficacité des modules mémoire standard, certains fabricants ont incorporé une petite quantité de SRAM directement sur la puce, créant ainsi un cache intégré. Une ESDRAM est donc essentiellement une SDRAM plus une petite quantité de cache SRAM qui autorise un fonctionnement en rafale jusqu'à 200 MHz. Comme avec un cache externe, la DRAM place les données le plus fréquemment utilisées dans le cache SRAM afin de réduire les accès à la DRAM, moins rapide. L'un des avantages de la SRAM sur puce est qu'elle permet la mise en place d'un bus plus large entre la SRAM et la DRAM, augmentant ainsi la bande passante et la vitesse de la DRAM.

– **FCRAM : Fast Cycle RAM**

La FCRAM, développée conjointement par Toshiba et Fujitsu est destinée à des applications spécifiques comme les serveurs évolués, les imprimantes ou les systèmes de commutation dans les télécommunications. Elle inclut une segmentation et un pipelining interne, qui accélèrent les accès aléatoires et réduisent la consommation électrique.

– **SLDRAM : Synclink DRAM**

Bien qu'elle soit considérée comme obsolète aujourd'hui, la SLDRAM a été développée par un groupe de fabricants de DRAM comme variante de la technologie Rambus, en fin d'années 1990.

– **VCM : Virtual Channel Memory**

Développée par NEC, la VCM permet à différents « bancs » de mémoire d'établir de manière autonome une interface avec le contrôleur de mémoire, grâce à un tampon. Il est ainsi possible d'attribuer à différentes tâches système leurs propres « canaux virtuels » ; de plus, les informations concernant une fonction ne partagent pas l'espace tampon avec d'autres tâches simultanées, ce qui rend le fonctionnement plus efficace.

2) Mémoires mortes : ROM

Elles sont réalisées en technologie MOS et bipolaire. Ces mémoires trouvent leur application dans la conversion de code, la génération de caractères pour l'affichage sous forme de matrice de points, le stockage de certains programmes système.

Elle ne nécessite pas d'énergie pour conserver leur information. En général, seule la lecture est possible. Il en existe plusieurs tps :

- **ROM : Read only memory**
Elles sont programmées en usine et ne peuvent pas être reprogrammées ;
- **PROM : Programmable read only memory**
Programmables par l'utilisateur, elle ne peuvent pas être reprogrammées ;
- **EPROM : erasable programmable read only memory**. Encore appelées UV-PROM, elles sont programmables par l'utilisateur et effaçables par effet photo-électrique à l'aide d'UV par la fenêtre placée au-dessus du boîtier.
Remarque : les EPROM sont programmable par octet, mais effaçables dans l'ensemble. Il existe une variable OTP (*one time programmable*). Leur capacité va jusqu'à 16 Mbits.
Exemple de boîtier : 2716. Lors de la programmation, il faut des impulsions de 50 ms sur prog (cf figure III.4).

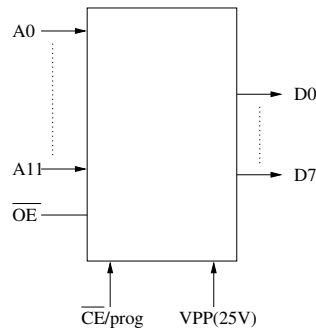


FIG. III.4: Boitier 2716

- **EEPROM : Electrically erasable programmable read only memory**.
Elle peut être effacée électriquement et reprogrammée dans l'application par octet.
Remarque : les RAM sont accessibles bit par bit. Sa capacité atteint 4 Mbits. L'endurance est limitée contrairement à celle des RAM qui est infinie. Elle est moins rapide en écriture que les RAM.
- **Flash EPROM :**
La mémoire flash est une mémoire à semiconducteurs, non volatile et réinscriptible, qui fonctionne comme la combinaison d'une RAM et d'un disque dur. La mémoire flash stocke les bits de données dans des cellules de mémoire, comme une DRAM, mais elle fonctionne comme un disque dur, dans la mesure où les données sont conservées en mémoire lorsque l'alimentation électrique est coupée. En raison de sa vitesse élevée, de sa durabilité et de sa faible consommation, la mémoire flash est idéale pour de nombreuses applications - comme les appareils photos numériques, les téléphones cellulaires, les imprimantes, les ordinateurs portables, les récepteurs d'ondes radio de poche et les dispositifs d'enregistrement sonore.[\[RAM\]](#)
Elle est programmable et effaçable électriquement comme les EEPROM. La seule différence provient de la structure interne et de la technologie utilisée qui permet

un meilleur compromis vitesse/capacité.

Les flash sont programmables par octet et effaçables soit dans leur ensemble (comme les EPROM) soit par secteur (secteur > 1 octet). Leur capacité va de 16 à 128 Mbits en 2002, des capacités de 256 à 512 Mbits doivent apparaître courant 2003. Il existe des mémoires flash monotension (pour l'alimentation de la lecture et de l'effacement) : elles remplaceront peut-être à terme les DRAM.

Remarque générale : les flash EPROM prennent actuellement plus de 50% du marché européen des mémoires non volatiles.

– **FRAM : Ferroelectric RAM.**

Elle combine les performances de L/Eillimitée des DRAM et la non volatilité. Sa capacité est de 256 Kbits (très faible).

– **PLA : Programmable logic array**

Les PLA sont similaires aux ROM mais différents par leur structure interne. Ils permettent de produire des fonctions logiques spéciales. Ils sont *mask programmable* (MPLA) ou *field programmable* (FPLA). Le PLA de base possède deux matrices ET et OU permettant de réaliser directement des sommes de produit : matrice ET en entrée, matrice OU en sortie.

III.2 Interface mémoire

Les lignes de données sont bidet à 3 états (sinon conflit sur le bus) : lors de la lecture seulement un des CI mémoire doit être validé à un instant donné.

Les circuits INS8202 ou 74LS244 sont des octuples buffer/drivers de lignes et permettent d'interfacer les lignes de données.

Les décodeurs, qui permettent de décoder les lignes d'adresse, sont un autre type de circuits d'interface mémoire.

III.3 Les différents type de mémoires vives d'un ordinateur

On distingue plusieurs types de mémoires vives en fonction de :

- leur utilisation ;
- leur vitesse ;
- leur mode d'adressage ;
- leur mode d'accès aux données.

1) Les registres

Un registre est un ensemble ordonné de bascules dans lequel on peut écrire puis mémoriser un mot binaire. Ce sont des mémoires particulières car l'information n'y est pas adressée.

Les registres se situent à l'intérieur de l'unité centrale, alors que la mémoire centrale forme un bloc fonctionnel à part entière en dehors de l'unité centrale. Ils offrent un temps d'accès en lecture/écriture beaucoup plus faible (rapport 10) que la mémoire centrale.

Conceptuellement, les registres et mémoire centrale sont semblables : ce sont le lieu d'implantation dans la machine, la capacité de stockage et le temps d'accès aux informations qu'ils contiennent qui les différencient.

- les registres accessibles aux programmeurs constituent une mémoire très petite (en général 16 registres d'un mot) comparés aux mémoires centrales, mais ils sont beaucoup plus rapide.
- au sein de la couche microprogrammée, les registres sont utilisés pour assurer le stockage d'informations nécessaires à l'exécution en cours de traitement.

2) Les piles

- **piles FIFO** : elles sont réalisées à partir d'un ensemble de registres dont l'accès aux données s'effectue de manière séquentielle sans l'aide d'une adresse.

La première information mémorisée sera la première lue. Une lecture est possible tant que toutes informations n'ont pas été lues. Une écriture est possible tant que la pile n'est pas saturée.

Application : utilisation d'une FIFO asynchrone avec deux horloges pour adapter l'échange d'information entre deux systèmes de vitesse différente.

Exemple : transmission de l'ordinateur vers l'imprimante en adaptant la vitesse de communication (cf figure III.5).



FIG. III.5: Adaptation de la vitesse de communication par une FIFO

Remarque : il y a une différence importante entre une pile FIFO et un registre à décalage conventionnel.

- **piles FILO** : ces piles sont en général implantées sous forme logicielle. La mémoire utilisée est une RAM dont l'adresse est générée par un pointeur qui est incrémenté ou décrémenté selon l'écriture ou la lecture des informations. Ces piles sont utilisées par le CPU pour mémoriser rapidement un ensemble de données ou d'adresses (retour de sous-programmes par exemple).

III.4 Mémoire centrale

C'est la zone de stockage des données et des programmes. Elle peut être considérée comme un ensemble de registres accessibles individuellement grâce à une adresse spécifique : mémoire à accès aléatoire.

La capacité de la mémoire centrale est plus ou moins importante selon le système. Elle est en général formée de plusieurs boîtiers de RAM : c'est une association de circuits en parallèle pour créer la longueur du mot désiré (traitable par le processeur) et de circuits en série pour créer la capacité désirée.

Exemple : constitution de 6 kmots de 8 bits à l'aide de boîtiers de $2k \times 4$ bits.

III.5 La mémoire cache, ou antémémoire

Les microprocesseurs 8 et 16 bits travaillent à des vitesses environ égales aux temps d'accès mémoire : la performance d'un système était donc due en général aux processeurs eux-mêmes.

Le problème majeur avec l'apparition des microprocesseurs rapides est de synchroniser l'accès à la mémoire avec la vitesse du processeur (la vitesse des microprocesseurs augmente vite alors que celle des mémoires évolue lentement). Autrement dit, il faut pouvoir satisfaire assez vite les demandes en écriture/lecture en mémoire pour ne pas laisser le processeur tourner à vide. La solution est la mémoire cache.

Principe : on place, entre le processeur et la mémoire centrale classique, quelques circuits de mémoire ultra-rapide (type SRAM) mais coûteux ; ce bloc renfermera les éléments les plus utilisés lors d'une tâche (cf figure III.6).

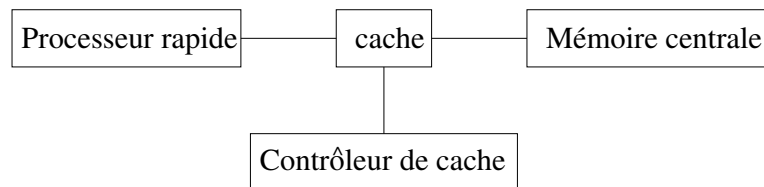


FIG. III.6: Principe de la mémoire cache

Chaque fois que le processeur doit lire une donnée, il commence par la recherche dans la mémoire cache. Si elle s'y trouve, il n'y a alors aucune attente, sinon le système charge dans le cache non seulement la donnée demandée mais aussi toute la zone environnante de la mémoire centrale. Grâce à cela la requête suivante du processeur a toutes les chances d'être effectivement dans le cache (il y a duplication de l'information mais ceci est transparent pour l'utilisateur).

Remarque 1 : certains processeurs ont une mémoire cache interne (486 : 8 ko, Pentium : 16 ko, Celeron 32 ko L1 + 128 ko L2, Pentium III : 32 ko L1 + 256 ko L2, Athlon : 128 ko L1 + 256 ko L2, Pentium 4 : 32 ko L1 + 256 ko L2)

Remarque 1bis : le Pentium 4 possède en fait un cache L1 de 8 ko associé à un cache de trace d'exécution L1 de 12000 opération : on a donc un cache données et un cache instructions distincts. Le cache L2 est de 256 ou 512 ko selon la version.

Remarque 2 : le 486 travaille en 32 bits (interne et externe) et le pentium en 64 bits interne et 32 bits externe. Pour le Pentium certains constructeurs ont conçu un bus sur 64 bits pour relier le cache et le contrôleur de cache ; d'autres ont adapté des solutions existantes et transfèrent les 64 bits en 2 fois 32 bits.

Remarque 3 : le Pentium pose un autre problème : ce circuit est si rapide que le temps de chargement du cache avec l'ensemble des données présentes en mémoire vive devient lui aussi critique. Quelques constructeurs proposent de relier cache et RAM par un bus

de 128 bits.

Structure générale d'un cache :

La mémoire cache n'est pas adressée comme toutes les mémoires, c'est-à-dire en faisant correspondre une adresse à une cellule mémoire.

Dans un cache sont rangées en même temps les données ainsi que leur adresse (la clé) correspondante dans la mémoire centrale (ou seulement une partie).

Ce type de mémoire, appelé « mémoire associative », opère selon deux étapes :

- test de présence d'adresse dans la mémoire adresse ;
- si oui, mettre la donnée sur le bus de donnée.

Remarque : la recherche par clé (adresse) dans la mémoire associative ne s'effectue pas de manière séquentielle, mais en parallèle sur toutes les cases de la mémoire associative. En un accès, on sait si l'instruction cherchée se trouve ou non dans l'antémémoire.

Structure générale d'un cache : recherche par clef (cf figure III.7).

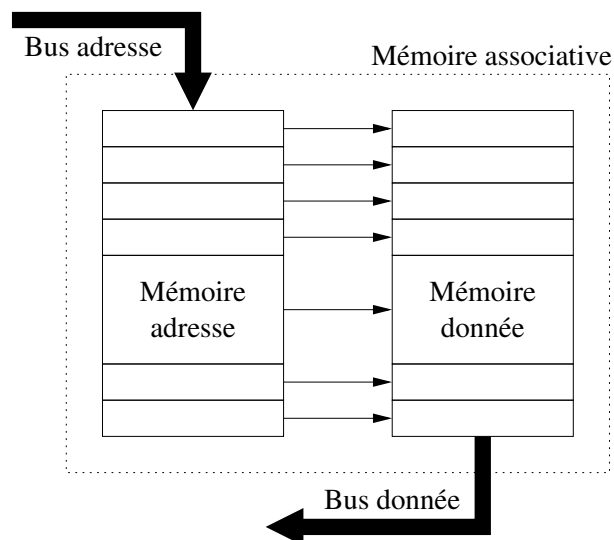


FIG. III.7: Recherche par clef dans une mémoire cache

Structure générale d'un cache : recherche par adresse (cf figure III.8).

Le contrôleur de cache 80385 :

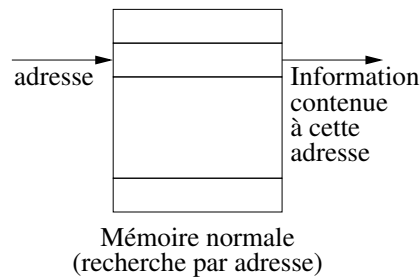


FIG. III.8: Recherche par adresse dans une mémoire centrale

La performance de mémoire cache se mesure à l'aide du taux de succès (nombre d'accès fructueux à l'antémémoire divisé par le nombre d'accès total à l'antémémoire). Il est obtenu expérimentalement et parfois dépasse 90%. Il dépend de la taille de l'antémémoire et de la localité du programme.

III.6 Mémoire virtuelle

Pour permettre l'utilisation de programmes volumineux et multi-utilisateurs, on fait appel à un système de gestion de la mémoire appelé « mémoire virtuelle ».

Elle permet de disposer de plus de mémoire que n'en offrent les circuits RAM de l'ordinateur, l'idée étant de séparer les concepts de la mémoire adressable et de la mémoire (RAM) disponible.

Fonctionnement : les parties de programmes ou de données qui excèdent la mémoire de l'ordinateur sont stockées sur une mémoire de masse (auxiliaire) et sont rechargées dans la RAM quand cela est nécessaire.

Rappel : hiérarchie de la mémoire dans un ordi :

- registres ;
- cache ;
- mémoire principale (RAM) ;
- mémoire auxiliaire (disque dur).

1) Principe de la mémoire virtuelle

Cela consiste à autoriser l'exécution d'un processus dont la capacité mémoire est supérieur à la RAM disponible.

Le dispositif pour la gestion de la mémoire virtuelle se compose de la RAM et d'une mémoire de masse (exemple : disque).

Exemple : soit un système à 16 lignes d'adresse, donc de 64 ko d'espace adressable, mais avec seulement 4 ko de mémoire centrale (RAM). Si le programme logé dans les 4 ko

fait appel à un processus dont l'adresse est hors des 4 ko, alors un défaut sera détecté. Le processeur va alors chercher un bloc de programme dans la mémoire auxiliaire et le place dans la zone disponible de la RAM. À ce moment, à l'adresse d'appel ne correspond toujours aucun programme. Il faut donc par une méthode de table, réaliser une translation des adresses afin que l'adresse spécifiée rentre dans le cadre des adresses disponibles – c'est-à-dire les adresses RAM (cf figure III.9).

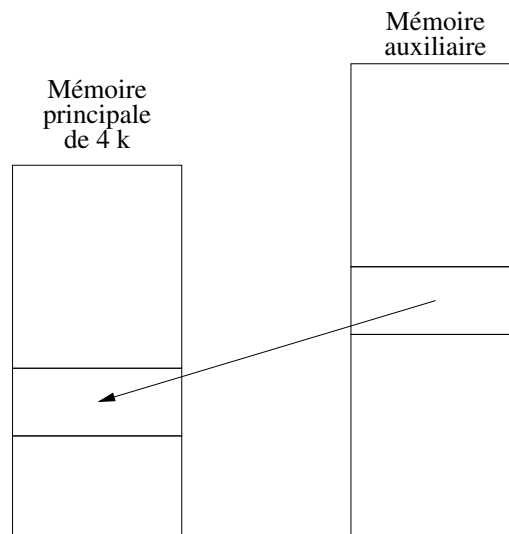


FIG. III.9: Principe de la mémoire virtuelle

Remarque : le principe de la mémoire virtuelle peut être comparé au principe de la mémoire cache.

2) Pagination

La pagination permet la gestion de la mémoire virtuelle. Elle consiste en une fragmentation de la mémoire auxiliaire en blocs de dimensions fixes appelées « pages ». Chaque page est chargée en mémoire centrale dès qu'elle est appelée par un processus. Elle sera alors implantée à une adresse disponible qui sera référencée dans une table de processus (remarque : la mémoire centrale est aussi divisée en pages, cf figures III.10 et III.11).

La gestion de la mémoire virtuelle est réalisée par le système d'exploitation qui exécutera les instructions suivantes :

- l'appel de la page auxiliaire ;
- son implantation en mémoire principale ;
- la mise à jour de la table des pages.

Cette méthode de gestion est transparente pour l'utilisateur (contrairement au principe de la segmentation). Le programmeur peut travailler comme si l'ensemble de son

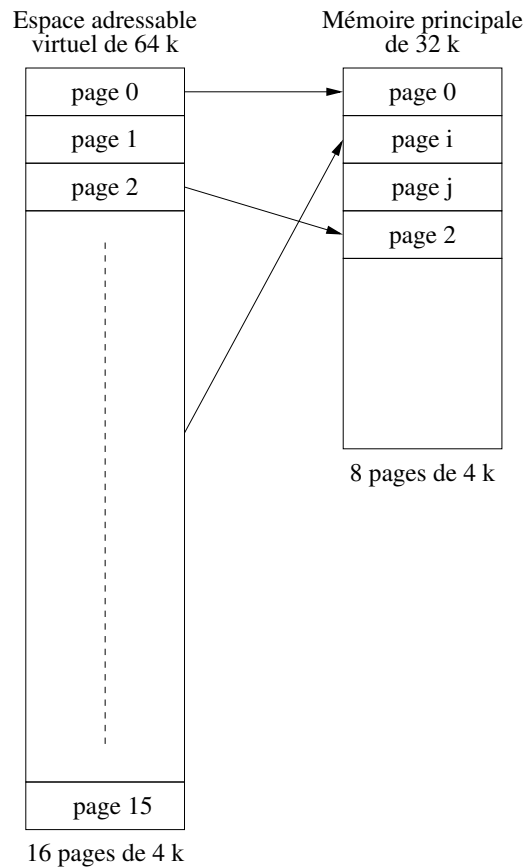


FIG. III.10: Exemple de pagination

programme était constitué d'une adresse linéaire.

La mémoire virtuelle et la mémoire cache ont un aspect dynamique : c'est sur demande instantanée (non planifiée) que le bloc d'information est amené dans le niveau inférieur. La mémoire centrale est comme une mémoire cache vis à vis de la mémoire virtuelle.

3) Segmentation

Dans ce cas la fragmentation de la mémoire est effectuée en zones de tailles variables contenant en général des entités homogènes (tableaux, piles, programmes ...)

La segmentation est également géré par une mémoire topologique qui précise la localisation des segments, mais aussi leur taille et leur présence ou non en mémoire principale.

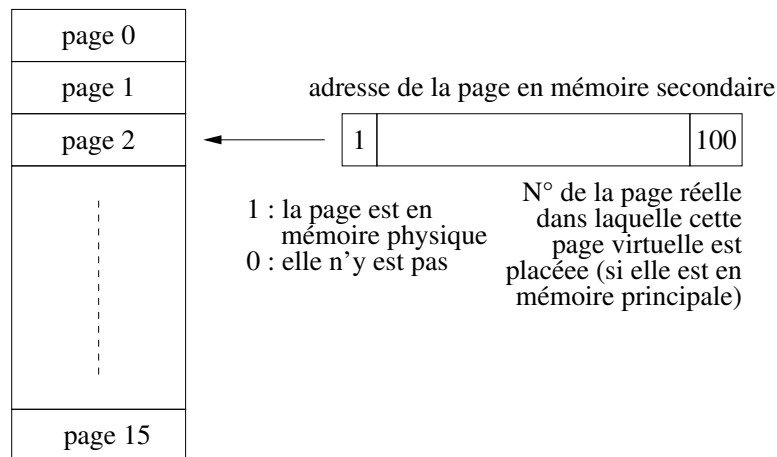


FIG. III.11: Fonctionnement d'une mémoire topologique

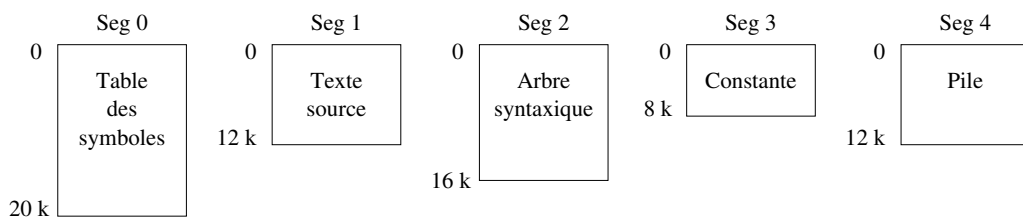


FIG. III.12: Exemple de segmentation utilisée par un compilateur

Remarque : chaque segment est géré indépendamment des autres.

4) Gestion de la mémoire virtuelle

Nous avons vu que la mémoire virtuelle est gérée par l'intermédiaire d'une mémoire topologique qui réalise le transcodage.

Il y a un va-et-vient des pages entre la mémoire principale et la mémoire secondaire durant l'exécution d'un programme. Si le contenu d'une page n'est pas modifié, il n'est pas nécessaire de la réécrire lorsqu'elle doit être retirée de la mémoire principale. Dans le cas contraire, il y a réécriture (et donc nécessité de mémoriser si il y a eu ou pas eu de modification).

En général, un système utilisant la mémoire virtuelle travaille en multiprogrammation, c'est-à-dire avec plusieurs processus activables. Dans ce cas, à chaque processus correspond une table de transcodage.

III.7 DMA (*Direct memory access*)

Le transfert de données entre un périphérique et la mémoire se fait souvent par bloc de caractères. Chaque transfert de caractère nécessite une intervention du processeur, donc une perte de temps pour le processeur, même en mode « interruption ».

La solution est donc de remplacer le traitement logiciel par un traitement matériel, en utilisant un circuit spécialisé : le contrôleur DMA.

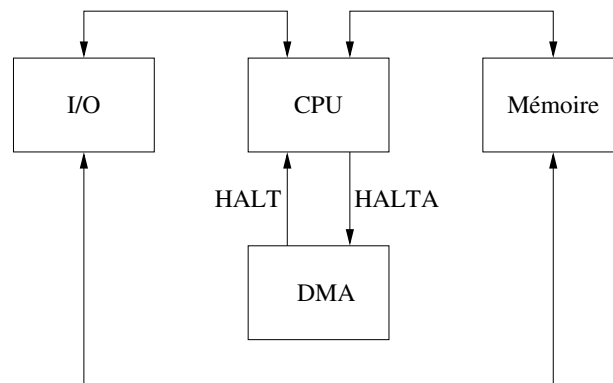


FIG. III.13: Principe du DMA

Les contrôleurs DMA sont assez coûteux. Il existe plusieurs philosophies de DMA ; par exemple, un DMAC ¹ peut suspendre le processeur, l'arrêter, voler des cycles mémoire au processeur, ou encore rallonger les périodes d'horloge.

1) Matériel pour la gestion mémoire

La réalisation des outils de gestion mémoire requiert de nombreux éléments matériels. Ceci peut inclure les mémoires associatives, les circuits de transformation d'adresse et les mémoires rapides pour les tables de pages. De tels équipements sont disponibles dans de nombreux systèmes.

Pour les micro-ordinateurs, ce matériel est fourni généralement sous forme d'un circuit intégré compatible avec le processeur utilisé (le *chipset*). De tels CI acceptent les adresses logiques du processeur et génèrent les adresses physiques en utilisant leurs tables internes. Des tables séparées peuvent être fournies pour les espaces utilisateurs et les espaces système. Le contenu de ces tables – qui peuvent donner des informations de protection et de droit d'accès – est chargé par le processeur.

La plupart des processeurs 16 et 32 bits disposent de circuits de gestion mémoire.

Remarques générales :

¹ *Direct memory access controller* : contrôleur d'accès direct à la mémoire, ou contrôleur DMA

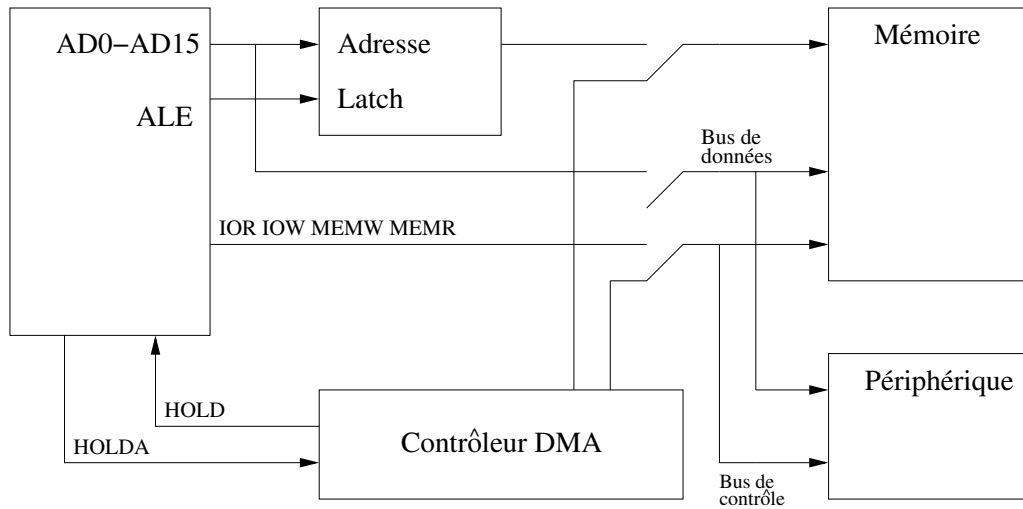


FIG. III.14: Fonctionnement du DMA

La mémoire principale est l'un des composants principaux des ordinateurs : leur taille et leur vitesse fixent la capacité des ordinateurs.

La mémoire cache est réalisée en technologie bipolaire, alors que la mémoire principale en technologie CMOS (coût plus faible).

III.7. *DMA* (Direct memory access)

IV – Le processeur

Rôle dans l'ordinateur : interprète et exécute les instructions d'un programme mémorisé en mémoire centrale.

Il se décompose en deux grandes entités :

- le séquenceur : automate exécutant de façon répétitive les séquences de :
 1. recherche d'instruction ;
 2. décodage de l'instruction ;
 3. exécution de l'instruction.
- le chemin de données : l'ensemble des composants et moyens empruntés par une information au cours du traitement (registres, mémoire, ALU, bus ...). Le chemin de données est aussi appelé la partie opérative.

IV.1 Le séquenceur

Le système apte à réaliser une succession d'événements afin de parvenir à un traitement désiré sur une ou plusieurs données ou déclencher des actions de commande d'un processus.

Deux problèmes se posent lors de l'élaboration d'un système séquentiel :

1. présentation du système (pour éditer et simuler son fonctionnement) ;
2. réalisation pratique.

Pour la présentation d'un système séquentiel on dispose de plusieurs méthodes :

- GRAFCET, élaboré en 1977 ;
- réseaux de Petri, élaborés en 1970 ;
- graphes cartésiens ou organigrammes, élaborés en 1970.

1) GRAFCET (graphe étape-transition)

C'est une représentation d'un système séquentiel basée sur l'alternance étape-transition : une étape suit une transition et inversement.

1.a) Représentation

Chaque étape est représentée par un carré dans lequel est spécifié un symbole associé à l'étape (en général un numéro). À chaque étape est associée un rectangle pour préciser l'action correspondante.

Remarque : une étape active peut être repérée par un ' '.

La transition à franchir est représentée par un segment placé sur la liaison entre deux étapes.

Remarque : à côté de ce segment on définit une expression (en général logique) qui représente la validité de la transition.

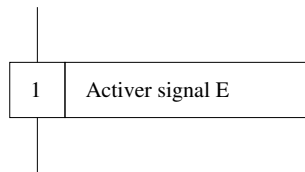


FIG. IV.1: Représentation d'une étape

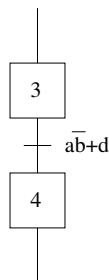


FIG. IV.2: Représentation d'une transition

Cette transition est appelée « réceptivité », elle peut être fonction de :

- l'état des variables d'entrée ;
- l'évolution de ces variables (front montant ou descendant) ;
- l'activité d'étapes ;
- l'état des variables internes (compteur, horloge).

1.b) Les règles d'évolution régissant le fonctionnement

- Règle 1 : une étape d'initialisation précise le début du fonctionnement.

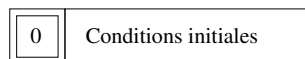


FIG. IV.3: Représentation d'une condition initiale

- Règle 2 : une transition ne peut être franchie que lorsque toutes les étapes immédiatement précédentes sont actives et que la réceptivité (la transition) est vraie.
- Règle 3 : le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement suivantes et la désactivation des étapes immédiatement précédentes.
- Règle 4 : si au cours du fonctionnement une même étape doit être désactivée et activée simultanément, elle reste active.

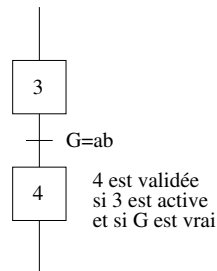


FIG. IV.4: Condition de franchissement d'une transition

1.c) Les différents cas de figures possibles

La figure ci-dessous présente les différents cas de figure de la sélection de séquence d'un GRAFCET.

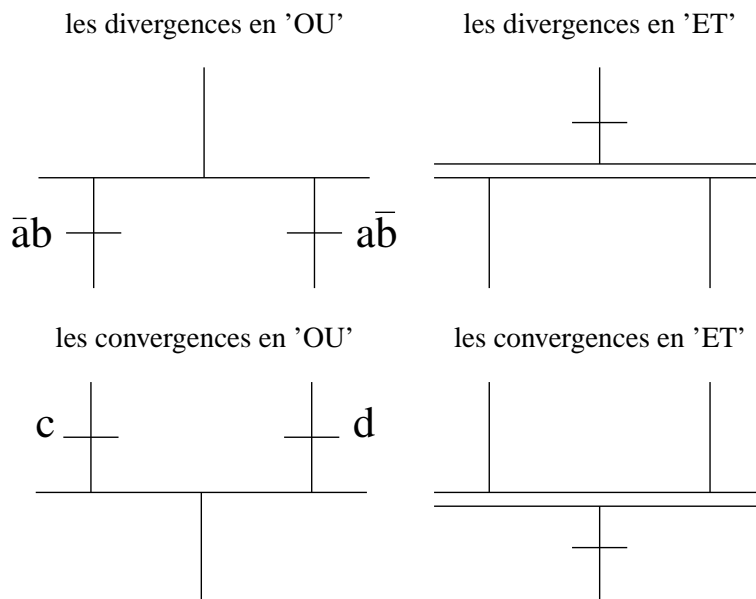


FIG. IV.5: Les différents types de divergences et convergences

1.d) Exemple : système de transfert de matériaux par chariot

Le système se compose d'une trémie dont l'activation s'effectue par un actionneur T durant le temps nécessaire pour le remplissage défini par le capteur P . Le chariot peut se déplacer à gauche et à droite à l'aide des moteurs G et D . Les capteurs f et d définissent la localisation du chariot (f : fin, d : début). Un actionneur V permet de

verser les matériaux dans le précipice. Le bouton m réalise la mise en route du système.

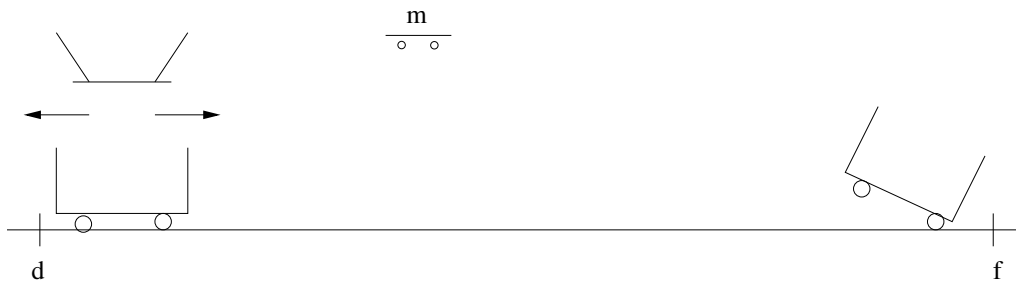


FIG. IV.6: Exemple d'utilisation du grafcet : tâche à automatiser

Présenter cet exemple par un grafcet.

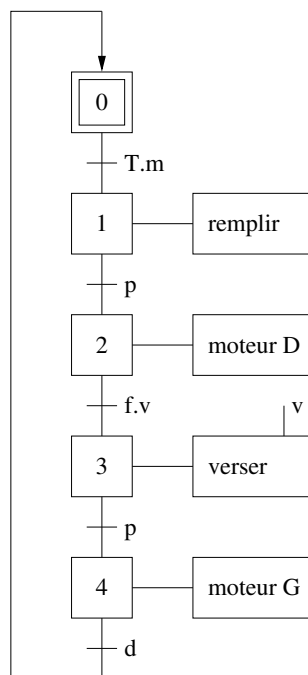


FIG. IV.7: Exemple d'utilisation du grafcet

2) L'organigramme

Comme le GRAFCET, l'organigramme est une méthode de représentation d'une suite séquentielle d'actions par un graphique symbolique.

- les actions sont décrites dans un rectangle ;
- les suites conditionnelles sont représentées par des figures à plusieurs sorties (losange ou hexagone) ;
- d'autres symboles sont utilisés pour des descriptions de sous-programmes, des étapes d'initialisation, etc.

Exemple : présentation de l'exemple précédent par un organigramme.

3) Comparaison entre les deux méthodes

Le GRAFCET permet l'activation de plusieurs branches simultanément, ce que l'organigramme exclut. Un tel mode de fonctionnement demande l'écriture de plusieurs organigrammes synchronisés par des variables secondaires.

IV.2 Réalisation pratique des circuits séquentiels définis par un GRAFCET ou un organigramme

1) Les séquenceurs asynchrones

Un système à évolution séquentielle se compose d'un certain nombre d'étapes au cours desquelles les actions sont stables, ce qui se traduit par un phénomène de mémoire (circuits utilisables : bascules D, RS, JK, cf figure IV.8).

Pour respecter les règles du GRAFCET, une étape (mémoire) est activée si la mémoire précédente est active et si la réceptivité associée est vraie.

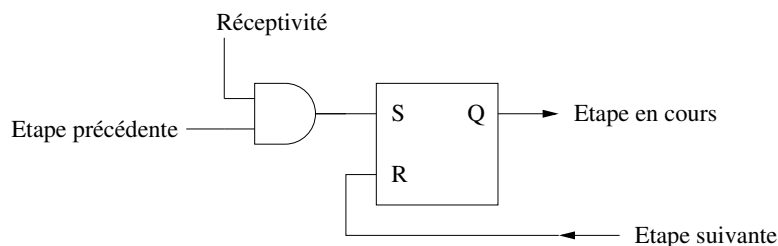


FIG. IV.8: Réalisation d'un GRAFCET à l'aide de logique asynchrone

La désactivation d'une étape en cours est réalisée par l'activation de l'étape suivante, par action sur l'entrée R.

Exemple :

Les figures IV.9 et IV.10 présentent un GRAFCET à implanter et le schéma logique de son implantation, respectivement.

IV.2. Réalisation pratique des circuits séquentiels définis par un GRAFCET ou un organigramme

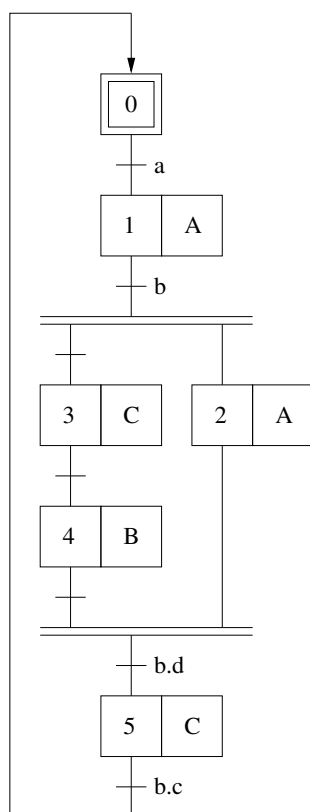


FIG. IV.9: GRAFCET à implanter à l'aide d'un circuit logique asynchrone

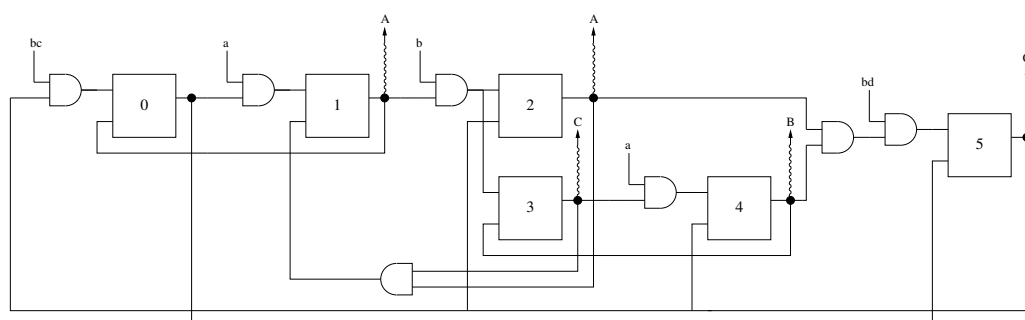


FIG. IV.10: Schéma électrique associé au GRAFCET précédent

2) Les séquenceurs synchrones câblés

- il s'agit de séquenceurs synchrones réalisés par le câblage de circuits logiques évoluant de façon synchrone sous la commande d'une horloge.
- le séquenceur étudié précédemment a une évolution asynchrone (une étape est activée dès que les réceptivités sont vraies) : les étapes surviennent de manière aléatoire et un tel circuit est très sensible aux parasites.
- solution : synchroniser chaque bascule sur un front d'horloge : séquenceur synchrone.

2.a) Exposé de la méthode

Les activation et désactivation associées devant être simultanées, on prend la même fonction logique pour l'activation des modules où l'on va et la désactivation des modules que l'on quitte.

Exemple :

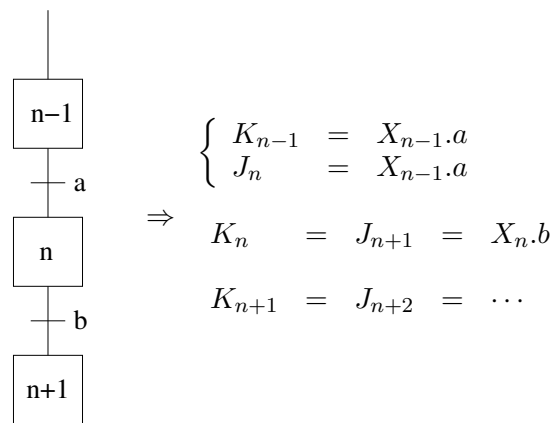


FIG. IV.11: GRAFCET à implanter ...

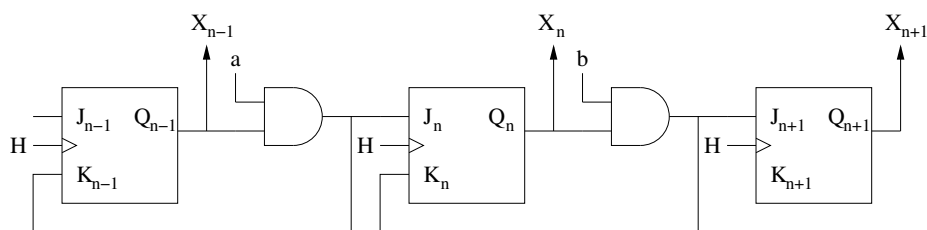


FIG. IV.12: ... et son implantation à l'aide de logique synchrone

Exercice : écrire les expressions des commandes des bascules JKT réalisant le GRAF-

CET suivant :

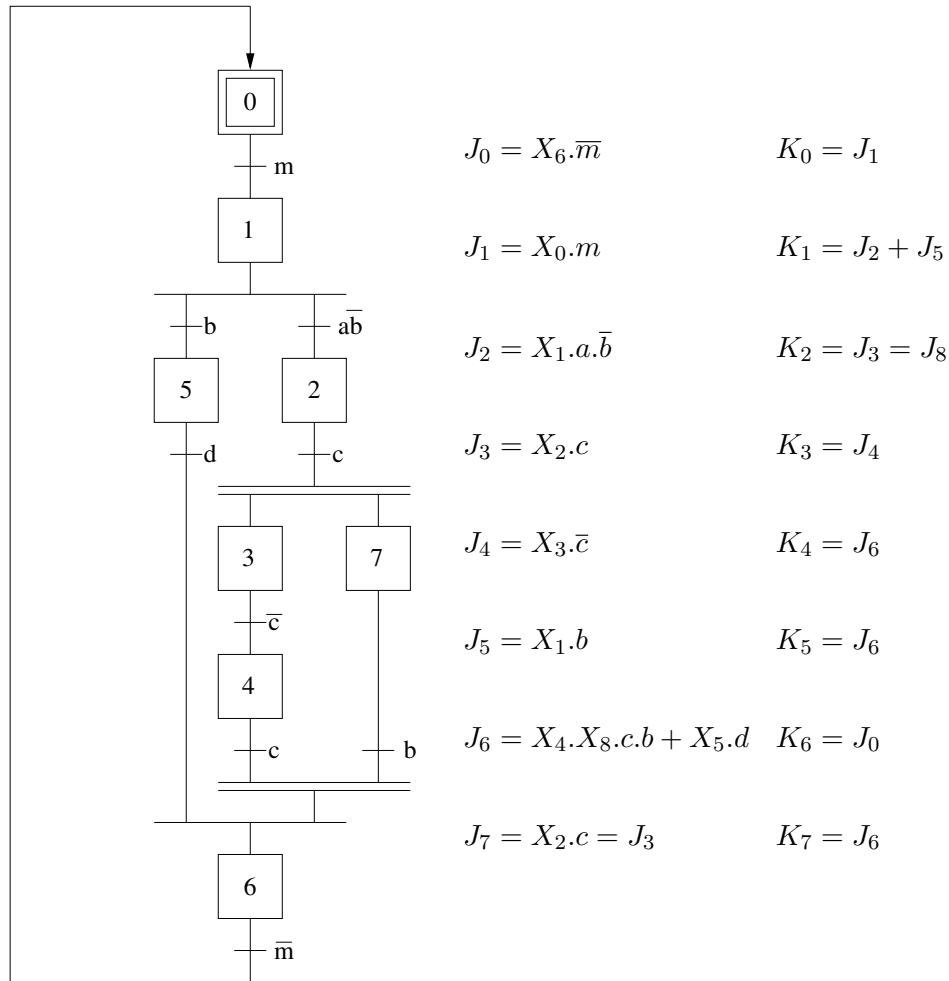


FIG. IV.13: Exemple de GRAFCET

2.b) Réalisation à partir des compteurs-décodeurs

Lorsque le nombre d'étapes est petit (inférieur à 4), on peut affecter une bascule à chaque étape (le câblage reste de dimension raisonnable). Si le nombre d'étapes augmente (supérieur à 8), on limite l'accroissement du câblage en utilisant un registre compteur et un décodeur.

Plusieurs cas de figure à traiter :

- enchaînement d'étapes : en faisant compter le compteur, on peut représenter 2^n

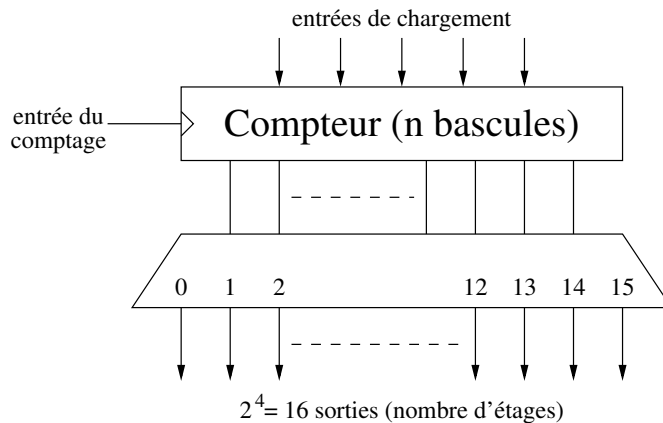


FIG. IV.14: Réalisation d'un séquenceur à l'aide d'un compteur

étapes successives, de numéro K , $K + 1$, $K + 2$, ...

Par exemple, soit le compteur dans l'état $K = 7$, seule la sortie 7 du décodeur est activée, et elle commande les actions à réaliser durant cette étape. Lorsque l'évolution du séquenceur fait passer le compteur dans l'état $K + 1 = 8$, la sortie 7 est désactivée ainsi que l'action de l'étape 7, et l'action de l'étape 8 est activée.

- tests et sauts d'étapes (transitions conditionnelles)
- test : le compteur doit patiner tant que la condition n'est pas vraie ;
- saut : en forçant le compteur à une valeur comprise entre 0 et $2^n - 1$, on peut passer de n'importe quelle étape de numéro i à une autre de numéro j .

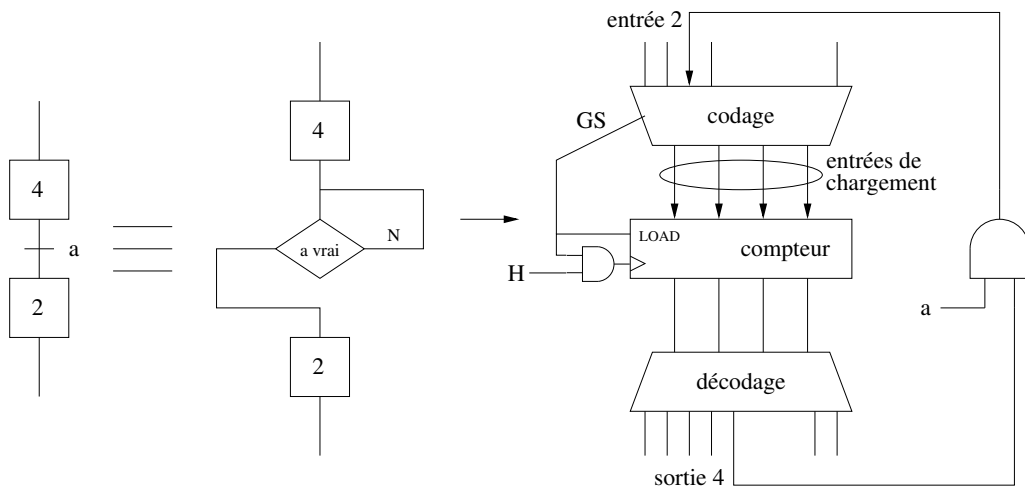


FIG. IV.15: GS indique si une entrée du codeur est activée ou non

- activation de plusieurs branches : l'activation simultanée de plusieurs états du compteur est impossible. Or, la traduction d'un GRAFCET en organigramme nécessite autant d'organigrammes que de branches actives simultanément, synchronisés par des variables secondaires : introduction de variable secondaires (telle que *start flag*, *end flag*).

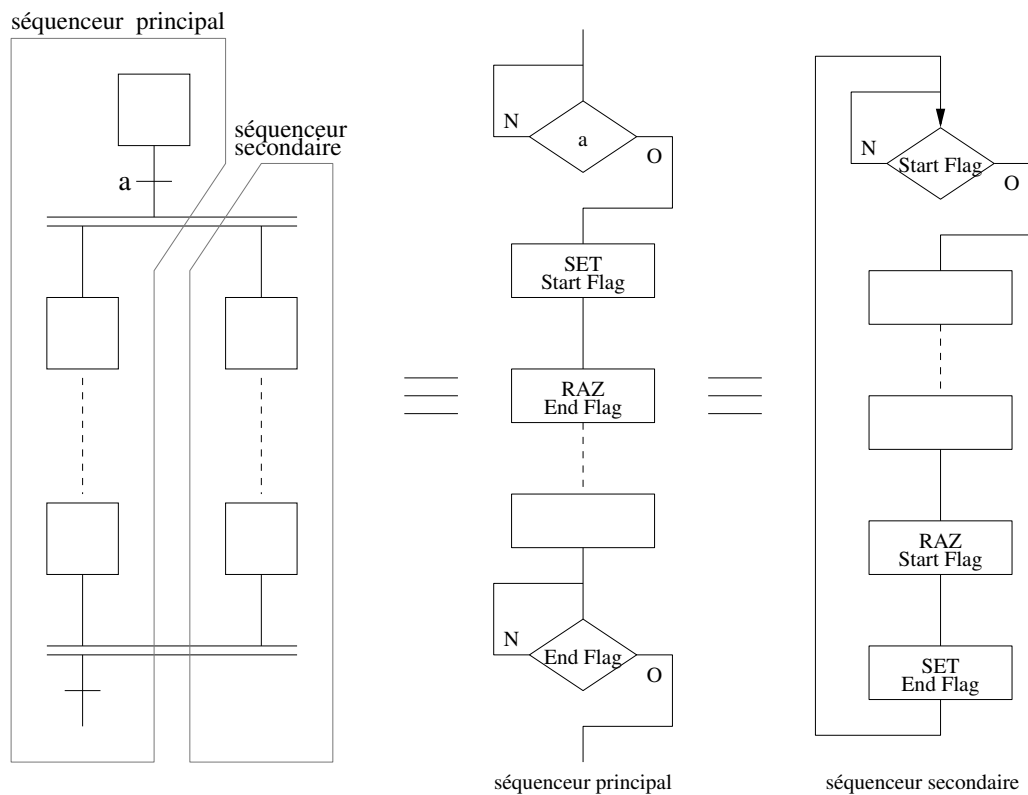


FIG. IV.16: Activation de plusieurs branches à l'aide de deux séquenceurs

avec :

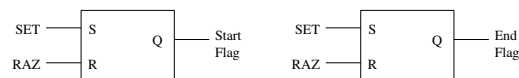


FIG. IV.17: Génération des signaux *Start Flag* et *End Flag*

2.c) Schéma global du séquenceur synchrone câblé

2.d) Exemple d'un séquenceur câblé

Réalisation d'un multiplieur 8 bits par 8 bits signé en complément à deux. Le résultat est donné sous forme de module et signe.

La partie opérative contient des circuits tels que :

- des registres universels assurant le chargement, le décalage à droite et à gauche, commandés par 2 bits S1 et S2 ;

S1	S2	
0	0	rien
0	1	décalage à droite
1	0	décalage à gauche
1	1	chargement

- un registre résultat de 15 bits pour le module et 1 bit pour le signe, validé sur front ;
- une UAL sur 16 bits dont la fonction à exécuter est codée sur 4 bits S_3 , S_2 , S_1 , et S_0 :

	S_3	S_2	S_1	S_0
A	0	0	0	0
A+1	1	1	1	1
zéro	0	0	1	1
A+B	1	0	0	1
⋮	⋮	⋮	⋮	⋮

- un compteur de type 74160

PT	LOAD	
0	1	comptage invalidé
X	0	chargement synchrone
1	1	comptage validé

On dispose du circuit de la partie opérative. Il est représenté sur le schéma suivant :

- la partie opérative est le circuit sur lequel va être réalisée l'opération désirée ;
- au départ, nous supposons que :
 - OP1 \rightarrow Reg1 (multiplicande)
 - OP2 \rightarrow Reg3 (multiplicateur)

Deux opérations sont à effectuer :

- validation du signe du résultat et transformer les opérandes négatifs en valeurs positives (par le complément à deux) ;
- ensuite, réalisation de la multiplication.

Algorithme 8 bits \times 8 bits

Le multiplicateur est analysé bit par bit à partir du LSB. S'il faut '1', le multiplicande est additionné au résultat puis décalé à gauche, sinon, il est seulement décalé à gauche.

Exemple :

										1	1	0	0	0	1	0	1	
									×	0	0	1	1	0	1	0	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	← résultat intermédiaire initialisé à 0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	bit 0 = 1
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	← R0
0	0	0	0	0	0	0	1	1	0	0	0	1	0	1				bit 2 = 1
0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1			← R2
0	0	0	0	1	1	0	0	0	1	0	1							bit 4 = 1
0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	1			← R4
0	0	0	1	1	0	0	0	1	0	1								bit 5 = 1
0	0	1	0	1	0	0	0	1	1	0	0	1	0	0	1			← R5 = résultat final, fin du traitement

Organigramme



Reg1 et Reg2 contiennent OP1 sur deux octets pour permettre l'addition des résultats intermédiaires sur deux octets.



On remarque que certaines actions sont actives sur des niveaux et d'autres sur des fronts. Pour éviter les problèmes de synchronisme, une horloge à plusieurs phases est utilisée (pour cet exemple, deux phases d'horloge sont utilisées : Φ_2 est associé à LOADR et Φ_1 active le séquenceur).

Après la numérotation des étapes, il faut réaliser le calcul des actions, le calcul des validations chargement et comptage, et le calcul des adresses de chargement.

(1) Calcul des actions

RAZ start flag = E17

SET end flag = E17

SIGN = E1

S11 = E16 + E4 + E6 VAL1 = E3 + E5 + E15

S12 = E16 VAL2 = E15

S21 = E16 VAL3 = E8 + E10

S22 = E16 VALR = E4 + E6 + E11 + E9

S31 = E9 + E11 LOADR = E3 + E5 + E8 + E10 + E12 + E15

S32 = E16

Les actions (fonctions) de l'ALU :

	S3	S2	S1	S0		
A	0	0	0	0	E3 + E8	\Rightarrow
A+1	1	1	1	1	E5 + E10	
zéro	0	0	1	1	E12	
A+B	1	0	0	1	E15	

$$\begin{aligned}
S3 &= E5 + E10 + E15 \\
S2 &= E5 + E10 + \\
S1 &= E5 + E10 + E12 \\
S0 &= E5 + E10 + E12 + E15
\end{aligned}$$

(2) Calcul des validations chargement et comptage

L'organigramme comporte 17 étapes, ce qui nous impose de prendre un compteur de 5 bits au minimum.

Le compteur dont nous disposons possède deux entrées LOAD et PT.

 Le chargement doit être validé (soit LOAD=0) à chaque fois qu'il y a un saut d'étape.

Calcul du patinage (arrêt du comptage) : patinage si $\overline{PT} = E0.start\ flag$

(3) Calcul des adresses de chargement

Définition des lignes de chargement du compteur A4, A3, A2, A1, A0.

A4	A3	A2	A1	A0		
0	0	1	1	1	E2 saut en E7	\Rightarrow
0	1	1	0	0	E7 saut en E12	
1	0	0	0	1	E13 saut en E17	
1	0	0	0	0	E14 saut en E16	
0	0	0	0	0	E17 saut en E0	

$$\begin{aligned}
A0 &= E2 + E13 \\
A1 &= E2 \\
A2 &= E2 + E17 \\
A3 &= E17 \\
A4 &= E13 + E14
\end{aligned}$$

3) Le séquenceur microprogrammé (microprogrammation)

Tout circuit séquentiel est constitué d'un circuit combinatoire rebouclé sur lui-même à travers un autre circuit.

Pour un séquenceur câblé, le circuit combinatoire se compose de fonctions logiques (constituées de circuits câblés) de manière à réaliser une succession d'actions bien déterminées et inchangées.

On peut réaliser le système de contrôle à l'aide de ces techniques classiques, mais il est de plus en plus fréquent d'opérer à l'aide de la microprogrammation (le logiciel associé reçoit le nom de *firmware*).

3.a) Constitution d'un séquenceur microprogrammé

- comme dans tout système programmé, une mémoire contient la suite séquentielle des opérations à effectuer (mémoire de contrôle).
- si nous regardons les représentations d'un enchaînement séquentiel (GRAFCET, organigramme), nous voyons qu'à chaque étape stable sont associées des informations telles que :
 - numéro ou code ou adresse de l'étape en cours ;
 - numéro ou code ou adresse de l'étape suivante ;

- type de transition avec l'étape suivante : inconditionnelle ou conditionnelle avec les conditions associées ;
- actions associées à l'étape.
- en fait, chaque étape est représentée par une fiche décrivant les informations précédentes. L'ensemble du fichier constitue le microprogramme et chaque fiche une microinstruction.

Les microinstructions définissent tous les signaux de contrôle pour le chemin de données et ont la possibilité de décider conditionnellement de l'instruction à exécuter à l'étape suivante.

L'invention de la microprogrammation permet de modifier le jeu d'instructions en changeant le contenu de la mémoire de contrôle sans modifier l'implantation matérielle.

Schéma de l'organisation du séquençement de microinstructions :

3.b) Les composants du séquenceur

(1) Circuit de décodage d'adresse

Un décodeur est utilisé pour générer l'adresse de début d'une microroutine donnée sur la base du code-opération contenu dans le registre d'instruction (IR).

(2) C.A.R.

Le CAR est un circuit apte à mémoriser le numéro (l'adresse, le code) de l'étape en cours.

Il peut être constitué de deux types de circuits :

- un simple registre (nécessite un chargement obligatoire à chaque fois) ;
- un compteur.

Dans les deux cas, les trois cas de figure suivants devront être exécutables :

- suite séquentielle
 - registre : chargement de l'adresse suivante ;
 - compteur : validation du comptage.
- patinage
 - registre : ne pas charger ;
 - compteur : invalider le compteur.
- branchement
 - registre : chargement de la nouvelle adresse ;
 - compteur : chargement de la nouvelle adresse.

(3) Mémoire de microprogramme

La mémoire de microprogramme est le circuit qui contient un code qui correspond à une étape donnée. En fait, cette mémoire est un circuit combinatoire qui fournit un ensemble d'états binaires en sortie en fonction d'un ensemble d'états en entrée (adresse).

Cette logique pourrait être réalisée à l'aide de circuits logiques câblés, mais ceci serait contraire à l'effet attendu, c'est-à-dire la souplesse de la programmation.

Le circuits programmables sont :

- les mémoires ;
- les PLA.

L'organisation des champs des microinstructions repose sur deux types :

- la microprogrammation horizontale ;
- la microprogrammation verticale.



Exemple de taille de la mémoire de microprogrammation : 4K microinstructions avec 50 à 80 bits par microinstruction.

La microprogrammation horizontale

Ce type de codage de la microinstruction consiste à employer un bit par microcommande, c'est-à-dire affecter un bit particulier à une action particulière et toujours la même. Donc, autant de bits que de bits de commande, ce qui aboutit à des instructions longues. L'intérêt est que les actions sont effectuées simultanément.

La microprogrammation verticale

On utilise un codage maximal des microcommandes. Seuls deux champs sont présents simultanément :

- un champ définissant le type du champ complémentaire ;
- un champ complémentaire définissant les microcommandes (concernant les actions, les adresses et les conditions).

champ complémentaire	Type de champ (code opération)
ACTIONS ADRESSES CONDITIONS	TYPE

En sortie du séquenceur, il faut que les microcommandes soient aiguillées vers différentes fonctions à effectuer. Ceci est réalisé à l'aide d'un multiplexeur commandé par le type de champ.

Le séquençement est alors découpé en plusieurs séquences de microcommandes et ceci pose un problème notamment lors de sauts conditionnels.

Lors des sauts conditionnels, il faut simultanément tester les variables externes, selon l'état recharger ou non une nouvelle adresse dans le CAR et générer l'ordre de chargement. Dans ce cas, il faudra réaliser ceci en deux étapes et mémoriser un état intermédiaire

L'organigramme de test se traduit en pratique par le schéma ci-dessus. D'où le schéma du séquenceur microprogrammé suivant :

Cette structure nécessite deux phases d'horloge décalées Θ_1 et Θ_2 . La condition de validation de chargement sera présente dans la bascule D après le calcul de la condition.

3.c) Exemple de séquenceur

Reprenons l'exemple du multiplieur câblé précédant pour une microprogrammation verticale.


(1) Organigramme

Il doit être modifié afin de créer deux étapes lors de chaque test.

Deux différences apparaissent par rapport à l'organigramme d'un séquenceur câblé :

- les tests se partagent en deux étapes ;
- les sauts inconditionnels génèrent des étapes à parts entières.

Donc, si nous décrivons le séquenceur d'un point de vue matériel, il nous faudra réaliser un saut conditionnel lorsque la variable secondaire est active.

 Dans certains organigrammes, nous voyons que des sauts conditionnels d'étapes sont effectués lorsque les tests sont faux (ex : sup) ou lorsque des tests sont vrais (ex : start flag).

Comme nous n'avons qu'une variable secondaire, il nous faut homogénéiser les tests et sauts : soit tous les sauts conditionnels sont valides pour un résultat VRAI du test, soit tous les sauts s'effectuent lors d'un test VRAI réalisé entre un bit du séquenceur et un état qui forcera la sortie à VRAI de la variable de chargement.

Ce qui donne au niveau du test matériel


(2) Organisation des microinstructions

Pour limiter le nombre de bits, nous travaillons en microprogrammation verticale. Nous allons donc créer plusieurs types de champs complémentaires :

- nous avons 28 étapes (d'après l'organigramme), ce qui nécessite un minimum de 5 bits pour son décodage ;
- lorsque nous analysons l'organigramme, nous nous apercevons que l'étape 25 nécessite 7 actions simultanées : 4 pour l'ALU, un chargement de registre (LOADR) et 2 validations de portes (VAL1, VAL2) ;
- nous disposons de deux manières d'agir : soit on augmente la taille des bits des champs complémentaires, soit on découpe les étapes en plusieurs étapes et crée des champs complémentaires différents (mais cela n'est pas toujours possible, par exemple lorsque la simultanéité est nécessaire comme dans notre problème). Par contre il existe un moyen en utilisant une mémorisation momentanée des actions.

Nous allons associer à une microinstruction contenant des actions de validation des portes, un registre (mémoire) actif sur front Θ_2 , la même que celle qui active LOADR. Ainsi l'écriture suivante sera valable.

On peut maintenant définir les différents types de microinstructions.

 Les modifications proposées accroissent le nombre des étapes. Il est nécessaire de créer un champ complémentaire de 6 bits au minimum.

Microinstruction de type 1

Dans une microinstruction de type 1, nous allons inclure les 4 bits de fonctions d'ALU (S3, S2, S1, S0), le chargement du registre résultat (LOADR) et l'ordre de validation (VALR).

VALR	LOADR	S3	S2	S1	S0	0	0	1
------	-------	----	----	----	----	---	---	---

type

Microinstruction de type 2

On y intègre les autres actions : SIGNF, VAL1, VAL2, VAL3, RAZSF, SETEF, CLEAR. Nous avons 7 actions pour 6 bits, donc codage simultané de SIGNF, CLEAR, VAL3.

SEF	REF	VAL1	VAL2			0	1	0
-----	-----	------	------	--	--	---	---	---

type

Microinstruction de type 3

On y inclut le reste des actions relatives aux commandes des registres. Nous y avons six actions (S11, S12, S21, S22, S31, S32) pour 6 bits : il n'est donc pas nécessaire de les coder.

S11	S12	S21	S22	S31	S32	0	1	1
-----	-----	-----	-----	-----	-----	---	---	---

type

Microinstruction de type 4

Dans cette microinstruction de type 4, on décrit les conditions de test. On dénombre 5 conditions : StartFlag, msbop1, msbop2, zero, lsop2. Il n'est pas besoin de les coder.

SF	msbop1	msbop2	zero	lsop2		1	0	0
----	--------	--------	------	-------	--	---	---	---

type

Microinstruction de type 5

Les adresses conditionnelles.

						1	0	1
--	--	--	--	--	--	---	---	---

type

Microinstruction de type 6

On y place les adresses incondionnelles.

						1	1	0
--	--	--	--	--	--	---	---	---

type

(3) Organigramme modifié

L'organigramme proposé plus haut va être légèrement modifié compte-tenu de nouveaux regroupements des actions.

(4) Le microprogramme

IV.2. Réalisation pratique des circuits séquentiels définis par un GRAFCET ou un organigramme

Maintenant que nous avons décrit la tâche à effectuer à l'aide d'un organigramme, nous pouvons procéder à l'écriture du microprogramme.

Il nous faut écrire une succession de microinstructions composées d'un ensemble de bits disposés selon le codage qui a été défini précédemment, ce qui est fastidieux et source d'erreurs.

Il existe des langages d'écriture de microprogrammes équivalents aux langages d'assemblage ou éventuellement aux langages de haut niveau de type interpréteur.

Nous allons rester ici, au niveau du langage binaire ardu mais intéressant pour la compréhension du fonctionnement du séquenceur.

N° étape	Champs complémentaires	type de champ
0	100000	100
1	000011	101
2	000001	110
3	000010	010
4	010000	100
5	000111	101
6	001111	110
7	001011	010
8	010000	001
9	100000	001
10	100000	011
11	001000	010
12	011111	001
13	100000	001
⋮	⋮	⋮

Bibliographie

[RAM] RAM Shopping Guide, <http://www.ramshopping.com/guide.php>