Microprocesseur - Micro-ordinateur

par Claude LACOMBE,
I.N.S.E.T., Abidjan (Côte-d'Ivoire).

Le microprocesseur est un composant électronique dont les domaines d'application ne cessent de s'étendre, souvent à notre insu, mais il est très souvent confondu et pris pour synonyme de micro-ordinateur.

Dans ce qui suit, nous avons essayé de montrer comment les microprocesseurs ont pu apparaître grâce aux progrès de l'intégration.

Nous nous sommes ensuite attachés à dégager les principes mis en application pour l'utilisation de ces composants, quel que soit le but de cette utilisation, pour finir en insistant un peu plus sur l'application informatique, le microprocesseur constituant alors le cœur d'un système appelé micro-ordinateur.

A) DE LA LOGIQUE CABLEE AUX CIRCUITS PROGRAMMABLES.

A.1) La logique câblée.

Supposons que l'on veuille réaliser une fonction déterminée comme une opération de calcul arithmétique ou une commande de mécanisme par exemple. Cette fonction peut être réalisée de différentes manières, et en particulier en représentation binaire par l'utilisation de portes logiques. Ces portes logiques peuvent pratiquement être réalisées en utilisant différentes technologies, les plus courantes à l'heure actuelle étant la TTL (Transistor Transistor Logic) et la CMOS (Complementary Metal Oxyde Semiconductor).

On peut aisément vérifier que le montage correspondant au schéma de la fig. 1 effectue bien l'addition binaire de deux nombres codés sous forme de niveaux de tension (voir Annexe 1).

On dit que la fonction addition est réalisée en logique câblée car il a fallu relier convenablement (par soudure...) les entrées et sorties des diverses portes.

Le montage réalisé est uniquement capable de faire une addition binaire mais il la fait bien et surtout vite : en logique TTL, il faudra environ 20 nanosecondes.

A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table de vérité de l'additionneur.

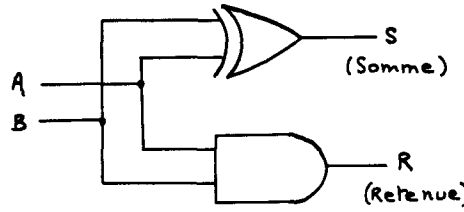


Fig. 1. — Additionneur logique.

La sortie S donne la somme booléenne des deux nombres A et B codés sur un bit, tandis que R indique éventuellement la présence d'une retenue.

A.2) **Réseaux logiques programmables.**

Ils sont plus connus sous le nom de PLA (Programmable Logic Arrays).

Pour la réalisation de fonctions complexes, le nombre de portes croît très vite et la notion de fiabilité du montage prend une importance capitale. Or, cette fiabilité diminue en fonction du nombre de connexions à établir. Les techniques d'intégration des composants étant de mieux en mieux maîtrisées, on a commencé à réaliser des circuits contenant un nombre de plus en plus important de portes logiques.

Mais il était hors de question de réaliser industriellement des fonctions particulières à la demande (à cause des coûts prohibitifs des séries trop petites), sauf pour quelques très gros demandeurs. D'où l'idée de circuits programmables par l'utilisateur en fonction de ses besoins propres.

Toute fonction logique pouvant être écrite sous forme de fonctions ET et OU correctement combinées, un PLA est constitué par un réseau de portes ET et un réseau de portes OU. La programmation de ces circuits consistant à fondre (à l'intérieur même du circuit intégré) les connexions entre les portes qui ne sont pas nécessaires à la réalisation de la fonction souhaitée (principe analogue aux fusibles protecteurs d'installations électriques domestiques).

L'inconvénient majeur de ces circuits est que leur programmation est définitive (gare aux erreurs au moment de cette programmation), les avantages restant ceux de la logique câblée.

A.3) **Les microprocesseurs.**

En poussant le raisonnement plus loin, on peut imaginer un circuit susceptible de réaliser des fonctions à la demande, éventuellement chaque fois différentes, c'est-à-dire réaliser une pro-

grammation momentanée du circuit, le temps de réaliser la fonction souhaitée. On obtient un microprocesseur.

On peut noter que, très naturellement, les microprocesseurs ont été conçus par des fabricants de composants (Intel 1970) et que l'utilisation « informatique » n'est apparue qu'à l'usage. Il faut aussi insister sur le fait que l'apparition de tels circuits est directement liée aux progrès technologiques en matière d'intégration.

B) CONCEPTS ASSOCIES A UN MICROPROCESSEUR.

B.1) Notion d'instruction.

La programmation d'un microprocesseur se fait en communiquant au circuit un certain nombre d'ordres qu'il est capable de comprendre. Chaque ordre exécutable par le microprocesseur est appelé instruction.

L'ensemble des instructions exécutables est le jeu d'instructions du microprocesseur. A titre d'exemple, on peut citer l'addition binaire avec ou sans retenue, instruction faisant partie du jeu de tous les microprocesseurs.

On peut aussi dire que, du point de vue de l'utilisateur, l'instruction est la fonction minimale réalisable.

Le microprocesseur étant un circuit logique, ne peut travailler que sur des informations par tout ou rien. L'unité d'information correspondant à un état logique (oui/non, 0/1 par exemple) est appelé bit (Binary digit).

Les instructions sont constituées par des « mots », les mots pouvant être définis comme des groupes de bits que le microprocesseur manipule simultanément.

Les microprocesseurs les plus utilisés utilisent des mots de 8 bits (Z80, 8080 ou 6800), mais la taille peut aller de 4 à 32 (voire davantage) bits. Dans le cas de mots de 8 bits, on parle alors d'octets (byte en anglo-saxon).

Dans la suite, nous supposerons que le microprocesseur travaille sur des octets. Dans ce cas, une instruction correspond à un code de 8 bits. Le nombre maximum d'instructions sera donc de 256 (2 puissance 8).

Pour fixer les idées, le 6800 a un jeu de 72 instructions. Il semblera que l'on soit très en dessous des possibilités, mais en fait, la plupart des instructions peuvent être réalisées de diverses manières selon la façon dont le microprocesseur travaille avec son environnement, d'où l'existence de plusieurs codes différents attachés à une même instruction.

Il faut enfin indiquer tout de suite que l'ordre de grandeur du temps d'exécution d'une instruction est sans commune me-

sure avec la logique câblée : pour les microprocesseurs utilisés dans les micro-ordinateurs de grande diffusion, le temps d'exécution d'une instruction est de l'ordre de quelques microsecondes, soit deux ordres de grandeur de différence à l'avantage de la logique câblée.

B.2) Programme.

Il incombe à l'utilisateur de concevoir une suite d'instructions que le microprocesseur exécutera successivement les unes après les autres, ce qui conduira à la réalisation de la fonction souhaitée.

Cette suite d'instructions constitue le programme.

Il est théoriquement possible de faire exécuter des programmes aussi complexes qu'on le veut et dans le domaine de la souplesse d'utilisation, le microprocesseur marque des points vis-à-vis des fonctions logiques câblées.

B.3) Mémoires.

Comment le microprocesseur peut-il prendre connaissance du programme qu'on lui demande d'exécuter ?

La suite des instructions constituant ce programme est stockée dans des mémoires, dispositifs susceptibles de conserver (mémoriser) des informations binaires. Nous ne parlerons pas de la technologie des mémoires.

Le microprocesseur devra donc être relié à cette mémoire de telle sorte qu'il puisse aller chercher successivement les instructions à exécuter : on dit que le microprocesseur adresse et lit la mémoire.

Il faut toutefois préciser qu'il existe deux grands types de mémoires :

— Les mémoires vives (RAM = Random Access Memory) mémoires que le processeur peut lire, mais également écrire. En effet, au cours du déroulement d'un programme, il est nécessaire de pouvoir stocker des données ou résultats intermédiaires, raison pour laquelle tout système bâti autour d'un microprocesseur comporte de la RAM.

— Les mémoires mortes (ROM = Read Only Memory) qui ne peuvent qu'être lues par le microprocesseur. La différence fondamentale par rapport aux RAM étant qu'elles conservent leur information même lorsqu'elles ne sont pas électriquement alimentées. Les informations contenues dans les ROM sont écrites au cours d'une opération spéciale appelée programmation de la mémoire. Certaines ROM peuvent être effacées puis reprogrammées plusieurs fois : c'est le cas des UVPROM que l'on efface à l'aide d'un rayonnement ultraviolet

intense (facilement reconnaissables à leur fenêtre en quartz permettant d'admirer la puce (Chip) de silicium).

Cette notion de mémoire est capitale :

— D'une part, elle montre que pour fonctionner, le microprocesseur nécessite un environnement que l'on peuplera un peu dans le paragraphe suivant.

— D'autre part, cette interconnexion avec d'autres composants soulève quelques problèmes, dont les plus importants sont ceux de synchronisation des échanges (Timing) dont nous dirons un mot dans l'analyse de l'architecture d'un système.

B.4) **Les Entrées/Sorties (Input/Output ou I/O).**

Un microprocesseur associé à de la seule mémoire serait totalement inutilisable parce que sourd et muet : il ne pourrait ni recevoir, ni donner d'informations à l'utilisateur (plus généralement au milieu extérieur). Il ne serait même pas possible de placer en RAM les programmes à réaliser.

Il est impératif d'établir des liaisons avec le milieu extérieur : ce sont les Entrées/Sorties.

Par ailleurs on appelle périphérique tout organe reliant le processeur au monde extérieur. On peut, assez arbitrairement, classer les périphériques en trois catégories selon les relations qu'ils permettent d'établir :

— avec l'opérateur : claviers, imprimantes, visu, télétypes,

— avec l'environnement : capteurs de toutes sortes, convertisseurs A/N et N/A, actionneurs,

— avec les mémoires de masse : bandes magnétiques, disques magnétiques (disquettes ou disques durs), bandes perforées, servant à stocker de grandes quantités d'information.

A propos des mémoires de masse, il faut faire ici la différence avec les RAM et ROM qui constituent la mémoire dite centrale (ou interne) directement accessible par le processeur et rapide (mais chère), alors que les mémoires de masse sont de capacité beaucoup plus grande, mais plus lentes d'accès : il faudra, avec une disquette, un temps typique de 2 ms pour trouver et lire un octet déterminé, alors qu'il suffit de quelques centaines de nanosecondes avec une RAM.

Un périphérique quel qu'il soit ne peut être connecté directement à un microprocesseur, ce dernier ne pouvant comprendre que des informations binaires organisées en mots (par exemple octets). Le travail de mise en forme des informations échangées est dévolu à des circuits (ou groupes de circuits) formant l'interface entre le périphérique et le processeur. Les fabricants de microprocesseurs ont le plus souvent développé des interfaces

en un seul boîtier d'une grande souplesse d'emploi, qui portent le nom explicite de Port d'E/S.

Deux modes fondamentaux d'échange des informations entre processeur et environnement sont utilisés :

— *Transmission parallèle :*

Tous les bits d'un mot sont transmis simultanément (en parallèle). L'interface porte alors le nom de PIA (Parallel Interface Adapter) ou PIO (Parallel Input Output) selon le fabricant.

Ce type de transmission est utilisé par exemple pour connecter un clavier.

— *Transmission série :*

Les bits d'un mot sont dans ce cas transmis les uns à la suite des autres (en série). Ce type de transmission sera utilisé dans le cas d'une liaison avec une bande magnétique, ou une télétype par exemple. Un interface souvent utilisé est l'ACIA (Asynchronous Communication Interface Adapter).

A ce stade, nous pouvons schématiser un système construit autour d'un microprocesseur par la fig. 2, ce schéma ne constituant pas un synoptique exact, à cause de la notion de BUS dont nous allons maintenant parler.

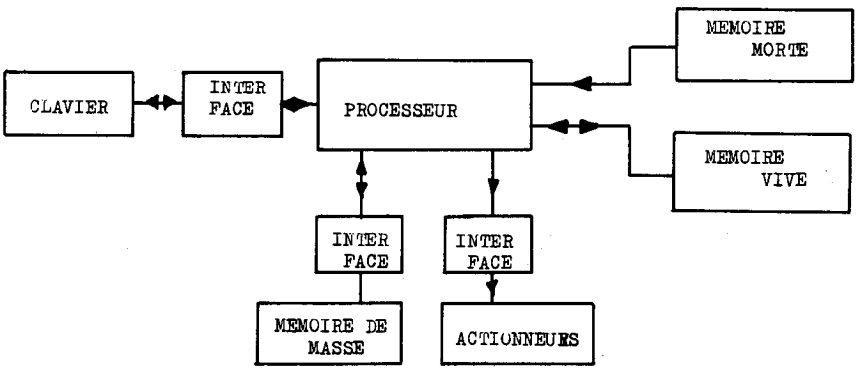


Fig. 2. — Exemple de liaisons entre le processeur et son environnement.

Ce synoptique est seulement théorique et la structure réelle d'un système fait intervenir la notion de BUS illustrée par la fig. 5.

C) ARCHITECTURE D'UN SYSTEME.

Il nous est maintenant possible de préciser comment on conçoit les interconnexions entre microprocesseur, mémoires et

interfaces pour aboutir à un système qu'il soit à vocation d'automate ou informatique. Dans le cas d'un système informatique, on parle alors de micro-ordinateur.

La structure du système constitue son architecture (ou Hardware).

La fig. 2 montre que si la construction du système devait être faite comme l'indique le schéma, le microprocesseur devrait avoir un très grand nombre de sorties, situation qui n'est pas du tout satisfaisante pour de multiples raisons (entre autres, limitation du nombre de périphériques, coût à la fabrication...).

On a donc pensé à articuler la conception des systèmes autour de la notion de BUS.

C.1) Bus de données (Data Bus).

Soit un microprocesseur 8 bits. Son boîtier comportera (entre autres) 8 pattes permettant l'entrée (ou la sortie) des octets. Ces pattes sont fréquemment appelées D0 à D7 (D pour Data et 0 à 7 correspondant aux puissances successives de 2 en représentation binaire).

Il est clair que ces 8 pattes devront être reliées aux pattes correspondantes de toute mémoire ou interface devant échanger des informations avec le processeur.

Cet ensemble de 8 lignes porte le nom de Bus de données (Data Bus). Il est dit bidirectionnel car les informations peuvent circuler soit à partir du processeur soit vers lui.

Il est ainsi théoriquement possible de raccorder autant de boîtiers qu'on le souhaite sur le Bus de données. Théoriquement, car il faut en réalité tenir compte de divers phénomènes électriques, et en particulier de l'affaiblissement des signaux et de leur dégradation le long des lignes. On remédie à ces défauts en intercalant des amplificateurs (Buffers) entre le processeur et les divers interfaces. Dans le cas du bus de données, ces buffers doivent être bidirectionnels. Cette remarque vaut également pour les deux bus suivants.

C.2) Bus d'Adresses (Address Bus).

Lorsque le microprocesseur envoie un octet sur le bus de données, cette information est destinée à un receveur bien précis (tel périphérique ou telle mémoire) à l'exclusion de tous les autres.

Il faut donc que le processeur envoie en plus des données, l'adresse du destinataire, et que ce dernier puisse se reconnaître. Le problème est résolu de manière très souple, mais au prix d'une petite complication matérielle, par la notion de Bus d'adresses.

Tout comme on dispose des 8 pattes du bus de données, il existe pour tout microprocesseur un certain nombre de pattes qui sont les lignes d'adresses. Le plus souvent, ce nombre est de 16 pour les processeurs 8 bits. Ceci correspond donc à

$2^{16} = 65536 = 64$ K adresses possibles (1 K = $2^{10} = 1024$).

On dira que l'espace adressable du processeur est de 64 K, une adresse correspondant à un état déterminé du bus d'adresse.

La petite complication matérielle provient du fait que les mémoires (ou les périphériques) possèdent moins de lignes d'adresse que le microprocesseur. Prenons un exemple :

Nous voulons relier une mémoire de 1 K octet (1024 octets) ou processeur. Cette mémoire comptera donc 10 lignes d'adresse. Supposons de plus que cette mémoire réponde aux adresses allant de 0000 à 0400 en numération hexadécimale (voir Annexe 2). Une conversion hexadécimale binaire donne :

X X X X X X X X X X 0 0 0 0 0 0 (X = état indifférent).
A0 ... A9 A10 ... A15

Les dix premières lignes du bus d'adresse reliées aux pattes correspondantes de la mémoire permettront de lire ou écrire les 1024 mots de cette mémoire, mais si on veut que ces adresses correspondent à 0000 à 0400 (hexadécimal) pour le processeur, il faudra que la mémoire ne réponde que lorsque les 6 autres lignes (A10 à A15) sont simultanément à 0.

A cet effet, les boîtiers disposent d'une ou plusieurs broches de validation, généralement appelées CS (Chip Selct) qui valident le boîtier lorsqu'elles sont soit à l'état bas (CS) soit à l'état haut (CS). Si nous supposons que notre mémoire exemple possède un CS, nous aurons le schéma de branchement de la fig. 3.

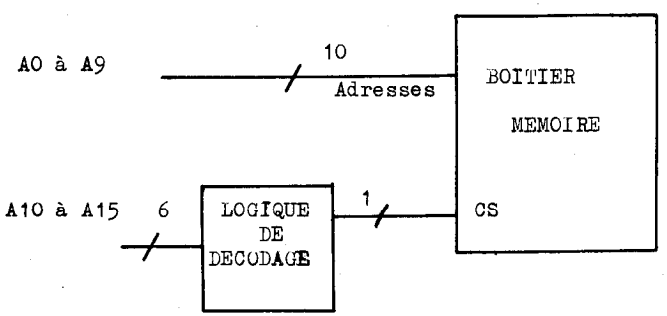


Fig. 3. — Principe d'un décodage d'adresse.

La logique de décodage positionne CS à 1 lorsque l'état des lignes d'adresse A10 à A15 correspond à l'adresse voulue.

La logique de décodage devra prendre en compte les lignes A10 à A15 du bus d'adresse et donner en sortie un 0 logique lorsque les 6 lignes sont à 0.

Cette notion de logique de décodage s'applique à tout l'environnement du microprocesseur.

C.3) **Bus de Contrôle (Control Bus).**

Un troisième bus est nécessaire pour que des échanges corrects puissent se faire dans le système. C'est le bus de contrôle, qui comporte un nombre variable de lignes selon le microprocesseur utilisé.

Ce bus comporte toujours une ligne dite R/W (Read/Write). En effet, le processeur peut vouloir recueillir un octet en provenance d'une mémoire ou périphérique (on dit alors qu'il lit) ou, au contraire, vouloir stocker cet octet dans ladite mémoire (on dit qu'il écrit). Le microprocesseur indique sa volonté en positionnant la ligne R/W (1 pour une lecture et 0 pour une écriture). Il faudra bien sûr que chaque partie du système reçoive cette information et soit connectée à cette ligne.

Une autre ligne de contrôle très importante est la ligne VMA (Valid Memory Address). Lorsque le microprocesseur a positionné correctement le bus d'adresse, donc lorsque l'adresse est valide (utilisable), il l'indique en positionnant la ligne VMA à 1 logique. Cette procédure est rendue nécessaire par l'existence de régimes transitoires sur les lignes lors des changements d'états logiques qui pourraient conduire à des erreurs d'interprétation.

Il faut également parler d'une autre ligne (mais qui n'existe pas sur tous les processeurs). Il s'agit de Ø2, ligne liée à l'horloge qui pilote tout le système. Ceci nous amène à parler des synchronisations (Timing) à respecter impérativement pour assurer un bon fonctionnement de l'ensemble.

Quand elle existe, cette ligne indique (par exemple quand elle est à l'état haut) que des données peuvent être envoyées sur le bus de données et que le processeur les acceptera.

En pratique Ø2 et VMA sont prises en compte par la logique de décodage de chaque mémoire ou interface.

C.4) **Horloge et timing.**

Toutes les opérations se déroulant dans un système (et à l'intérieur même du microprocesseur) sont rigoureusement cadencées, raison pour laquelle il y a toujours une horloge (généralement à quartz pour assurer une bonne stabilité). Cette horloge est de plus en plus souvent intégrée au boîtier du processeur, seul le quartz restant alors à l'extérieur.

L'utilisateur du microprocesseur n'a pas à se préoccuper du cadencement des opérations se déroulant à l'intérieur de celui-ci (il peut seulement avoir à se soucier de la durée d'exécution des instructions si le temps de déroulement du programme est un facteur important), par contre il devra faire attention en ce qui concerne le transfert des octets sur le bus de données.

Par construction, le processeur interroge son environnement de la manière suivante :

— il positionne les adresses et R/W = 1,

— puis VMA = 1 indique que tout est prêt.

A partir de ce moment-là, il attend que les données se présentent (fournies par la mémoire ou le périphérique ayant répondu à l'adresse sollicitée) pendant un laps de temps déterminé et surtout limité. Il faut impérativement, s'il s'agit par exemple de la lecture d'une mémoire, que celle-ci soit assez rapide pour répondre pendant l'attente du processeur. On appelle temps d'accès le délai de réponse d'une mémoire ou d'un périphérique. Le temps d'accès doit donc être inférieur au temps d'attente du processeur.

Ceci explique le fait que les mémoires de masse ne puissent être directement adressées par le microprocesseur, leur temps d'accès étant beaucoup trop long.

Avec une horloge de fréquence 1 MHz, le temps d'attente du processeur est de l'ordre de 500 à 600 nanosecondes.

Les temps d'accès des mémoires à semi-conducteurs (RAM ou ROM) varient en gros de 150 à 450 nanosecondes (parfois plus), ce temps étant de l'ordre de la milliseconde pour une disquette.

Toutes les informations relatives au timing d'un composant sont souvent données sous forme de diagramme, la fig. 4 en donnant un exemple simplifié. Cette figure correspond à une horloge à 1 MHz et nous montre que les adresses sont valides 300 ns maximum après la descente à 0 de Ø2, cet état se traduisant par la montée à 1 de VMA. On voit que les données sur le bus de données ne peuvent être lues que lorsque Ø2 = 1 et, au plus tard, 100 ns avant la descente de Ø2.

Il faudra donc que la mémoire puisse présenter des données valides sur le bus correspondant en moins de :

$$200 + 500 - 100 = 600 \text{ ns.}$$

Cette valeur sera son temps d'accès maximal pour qu'elle puisse fonctionner correctement avec ce microprocesseur. En pratique, on ne dépassera pas 400 à 450 ns pour conserver une bonne marge de sécurité.

Pour conclure ce paragraphe, la fig. 5 montre le synoptique général d'un système illustrant la notion de bus.

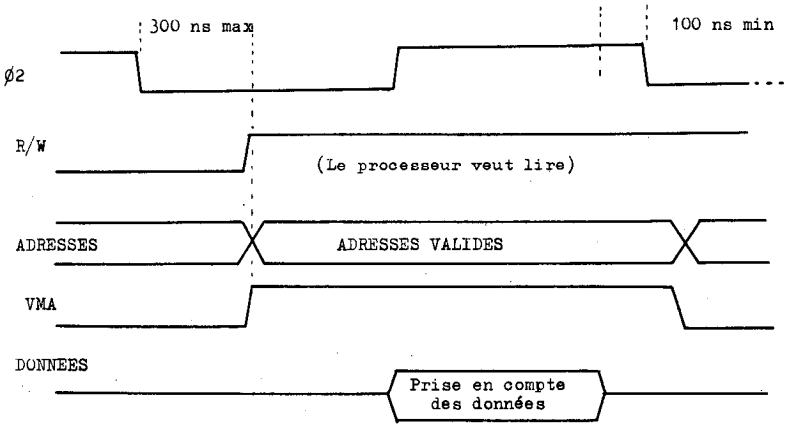


Fig. 4. — Procédure de lecture par le processeur.

Le processeur prend en compte les données présentes sur les lignes correspondantes lorsque R/W = 1, VMA = 1 et Ø2 = 1 et, au plus tard, 100 ns avant la descente de Ø2.

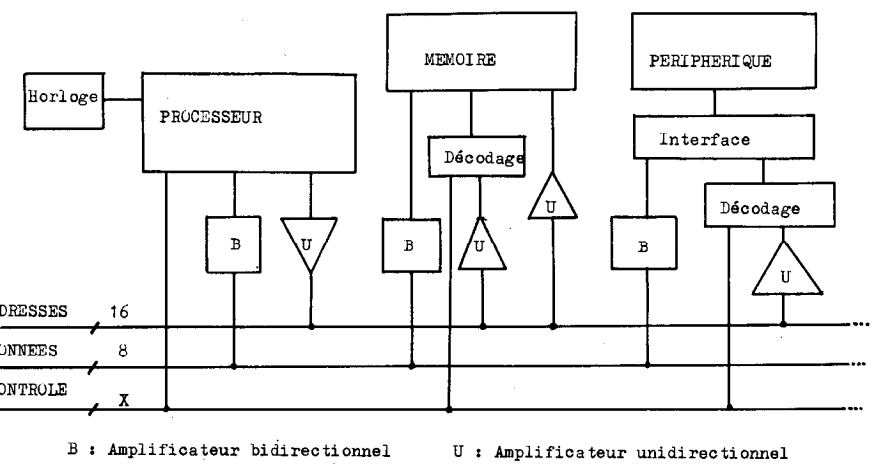


Fig. 5. — Synoptique d'un système à microprocesseur.

Cette figure illustre la notion de Bus. Il est possible de connecter sur ce Bus un nombre quelconque de mémoires ou périphériques.

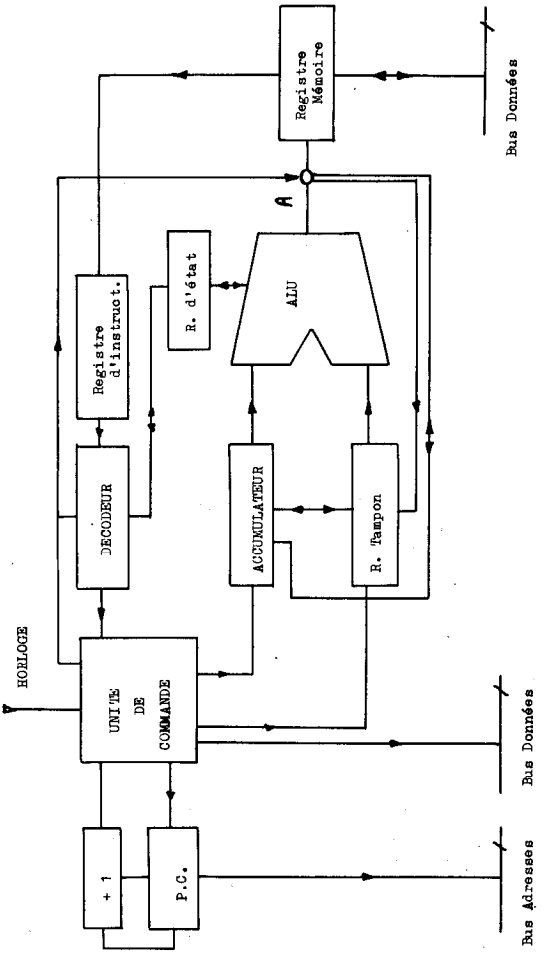


Fig. 6. — Synoptique simplifié d'un microprocesseur.

A : aiguillage : il est commandé par l'unité de commande et permet le transfert des octets, soit vers le registre d'instruction, soit vers l'accumulateur ou, au contraire, de l'ALU vers le bus de données.

ALU : unité arithmétique et logique : cette unité effectue les calculs arithmétiques (additions...) ou les fonctions logiques.

Registre d'état : il informe sur l'état du processeur après l'exécution d'une instruction (existence d'une retenue après une opération par exemple).

D) STRUCTURE INTERNE D'UN MICROPROCESSEUR.

Nous allons très brièvement parler de la structure interne d'un microprocesseur.

Quelle que soit l'instruction à exécuter, le travail du processeur se décompose toujours en 3 étapes :

- Aller chercher l'instruction (Fetch).
- La décoder (Decode).
- Enfin l'exécuter (Execute).

L'analyse sommaire de ces étapes va nous permettre de comprendre quelle est la structure générale du microprocesseur.

D.1) Fetch.

On suppose que l'on se trouve dans le cas le plus simple de déroulement linéaire d'un programme, c'est-à-dire tel que les instructions du programme sont rangées à des adresses successives dans une mémoire. Pour savoir à tout instant où il en est, le microprocesseur comporte un compteur de programme (PC = Program Counter) ou compteur ordinal, qui est un simple registre (c'est-à-dire une mémoire d'un seul mot) contenant à chaque instant l'adresse qui vient d'être lancée sur le bus + 1. Ce registre est de 16 bits puisque c'est la longueur des mots d'adresse.

Piloté par l'unité de commande (Command Unit) ou séquenceur, véritable cerveau du circuit, le contenu du PC est envoyé sur le bus d'adresse, et le bus de contrôle est positionné. Le PC est incrémenté de 1.

Lorsque l'octet attendu se présente sur le bus de données, il est réceptionné dans le registre mémoire. S'il s'agit d'une instruction, il est envoyé dans le registre d'instruction, toujours sous contrôle de l'unité de commande qui sait à tout moment (en fonction du travail précédemment effectué) si l'octet reçu correspond à une instruction ou une donnée (voir Annexe 3).

D.2) Decode.

Le mot contenu dans le registre d'instruction est analysé par le décodeur qui définit la séquence d'opérations à effectuer pour réaliser l'instruction reçue.

Si le mot reçu est une donnée, relative à une précédente instruction, l'unité de commande peut, selon l'opération à effectuer, l'envoyer dans des registres de stockage (accumulateurs) ayant accès à l'unité arithmétique et logique (ALU) si une telle opération doit être faite (cas par exemple d'une addition ou d'un ET logique...).

D.3) **Execute.**

Ensuite l'unité de commande assure le séquencement défini lors du décodage de l'instruction.

Les mêmes opérations sont ensuite reprises pour traiter l'instruction suivante, jusqu'à la fin du programme.

Ce qui précède est schématique et le nombre d'accumulateurs et registres peut varier d'un microprocesseur à un autre, ainsi d'ailleurs (dans une certaine mesure) que l'organisation interne.

E) LOGICIEL. LANGAGES DE PROGRAMMATION.

Par opposition au matériel (Hardware), on appelle logiciel (Software) tout ce qui touche à la programmation du microprocesseur ou du micro-ordinateur. En parlant de programmes définis comme suites d'instructions, nous parlions déjà de logiciel.

E.1) *Les niveaux de langage.*

Les langages de programmation peuvent être classés en trois catégories appelées niveaux de programmation.

1) LE LANGAGE MACHINE.

C'est le seul que la machine comprenne. On parle aussi de code machine. Chaque instruction exécutable par le processeur correspond à un mot binaire (octet si 8 bits). Par exemple pour le 6800 (microprocesseur 8 bits), le code :

00001001

sera compris comme étant l'ordre de décrémenter un registre interne particulier en l'occurence le registre d'index.

Ce simple exemple montre les inconvénients que représente le dialogue en binaire entre l'utilisateur et la machine :

— lenteur de la frappe au clavier,

— très grand risque d'erreur,

— les codes ne « parlent » pas, aggravant les risques d'erreur,

— chaque microprocesseur a ses propres codes (même si on retrouve des instructions identiques, les codes ne seront pas les mêmes)...

En fait, même avec les plus petits systèmes (tel que les cartes dites d'évaluation), il n'est pas nécessaire de travailler en binaire. Dans ce cas, on continue d'utiliser directement les codes machine (langage machine), mais en les écrivant (et en les tapant au clavier) sous forme hexadécimale (quelquefois aussi sous forme octale). Ainsi l'instruction exemple précédente (décrément du registre d'index) sera-t-elle écrite 09 (traduction de 00001001 en

hexadécimal). Il faudra que l'interface entre le clavier et le processeur génère le code 0000 quand on frappera 0, et 1001 quand on frappera 9 par un moyen quelconque. Ce moyen peut être logiciel, c'est-à-dire être un programme résidant en permanence dans la machine (donc en ROM).

Le seul avantage par rapport au binaire étant une forte diminution du risque d'erreur à la frappe ou à la lecture.

2) L'ASSEMBLEUR.

Pour mémoriser plus facilement les différentes instructions, on les note sous forme d'abréviation (le mnémonique) : l'instruction de code 09 de l'exemple précédent est notée DEX (pour décrémenter X).

On a alors pensé à écrire les programmes en utilisant systématiquement les mnémoniques au lieu des codes. On y gagne énormément en lisibilité des programmes qui sont bien plus faciles à analyser (ou à modifier ou corriger).

Si on frappe les mnémoniques au clavier pour entrer le programme, il faudra traduire ces mnémoniques en binaire avant de pouvoir faire exécuter le programme d'assemblage (à ne pas confondre avec l'assembleur qui est un langage et non un programme). Le programme d'assemblage devra, bien sûr, être présent en mémoire lorsqu'on voudra programmer, mais une fois que les mnémoniques auront été traduits (on dira que le programme a été assemblé), il sera devenu inutile.

Il faut noter que chaque ligne du programme écrit en assembleur correspond à une seule instruction exécutable par le microprocesseur.

L'assembleur est un progrès important par rapport au langage machine, mais il a l'inconvénient de nécessiter une bonne connaissance du jeu d'instructions du processeur de la machine que l'on utilise. En contrepartie, il a l'avantage, sur les langages de haut niveau dont nous allons parler, de permettre l'optimisation de l'utilisation du processeur et l'exécution très rapide des programmes.

3) LES LANGAGES ÉVOLUÉS.

Le fait d'avoir à connaître le jeu d'instruction du processeur de la machine est souvent un inconvénient (si l'on change de machine et que celle-ci soit architecturée autour d'un autre microprocesseur...). Le dernier stade consiste donc à s'affranchir du matériel et à concevoir un langage (dit de haut niveau ou évolué) qui en soit indépendant. C'est le cas de BASIC, LSE, FORTRAN, COBOL...

Une commande telle que PRINT (en BASIC) sera comprise par n'importe quelle machine et ne dépend plus du processeur utilisé.

Bien évidemment, le processeur ne comprendra pas l'ordre PRINT (impression sur le terminal vidéo), et il faudra que le logiciel de la machine comporte le programme de traduction du langage évolué en langage binaire. Par opposition à l'assembleur, une instruction d'un langage évolué sera traduite en une suite plus ou moins longue d'instructions exécutables par le processeur. Il faut aussi préciser que la puissance des instructions d'un langage évolué est souvent liée à un allongement parfois très important du temps d'exécution des programmes par rapport à l'assembleur.

Il existe deux grands principes de traduction :

- Les interpréteurs : ces programmes prenent une instruction du langage évolué, la traduisent en langage machine et la font exécuter immédiatement.
- Les compilateurs qui traduisent d'abord la totalité du programme et génèrent en mémoire un programme exécutable avant de faire éventuellement exécuter le programme.

Les compilateurs ont l'avantage de ne pas nécessiter leur présence en mémoire au moment de l'exécution du programme au contraire des interpréteurs, et surtout l'exécution d'un programme compilé sera beaucoup plus rapide. On retrouve là un des avantages de l'assembleur.

E.2) **Les utilitaires.**

On ne peut pas parler de logiciel de micro-ordinateur sans dire un mot des programmes particuliers qui assurent le fonctionnement de la machine. Ce sont les utilitaires ou programmes de service. Nous pouvons comprendre ce que peuvent être ces programmes à travers quelques exemples.

A la mise sous tension de la machine, il doit s'effectuer une série d'initialisations. Un programme sera chargé de ce travail. Il devra nécessairement être placé dans une mémoire non volatile (qui ne peut s'effacer) et en mémoire centrale, c'est-à-dire en ROM.

Le dialogue avec les périphériques nécessite aussi la mise en œuvre de programmes plus ou moins complexes : on a déjà parlé du codage d'un clavier. On peut aussi citer la gestion d'une unité de disquettes. Dans ce dernier cas, le programme de gestion porte le nom de DOS (Disk Operating System).

Sur des machines d'une certaine importance, on peut rencontrer des programmes de sauvegarde permettant par exemple en cas de coupure secteur, d'arrêter un programme en cours, puis de reprendre automatiquement sans perte d'information ou de contrôle après rétablissement de l'alimentation.

Dans le cas de tous petits systèmes, l'ensemble des utilitaires porte souvent le nom de moniteur. Si cet ensemble est plus important, il sera appelé système d'exploitation (Operating System).

Précisons pour terminer que les moniteurs contiennent très souvent des programmes d'aide à la programmation (Debugging Programs) particulièrement utiles, qui permettent la détection des erreurs et la mise au point des programmes écrits par l'utilisateur.

CONCLUSION.

Au terme de ces pages, nous n'avons pu que tenter de cerner un sujet particulièrement vaste, dans un domaine en fulgurant développement, et nous avons été obligé de schématiser quelque peu certaines notions ou même de passer sous silence de nombreuses choses comme les divers modes d'adressage, les éventuels multiplexages d'adresses, ou encore la notion de troisième état (Tree State).

Nous remercions M. Michel DEBERNARD pour la lecture critique qu'il a bien voulu faire de ce texte.

ANNEXE 1

PORTES LOGIQUES

Nous rappelons ici très brièvement les propriétés des fonctions logiques fondamentales, sous forme de table de vérité (tableau donnant l'état logique de la sortie en fonction des états des entrées).

Ces fonctions peuvent se réaliser pratiquement de nombreuses manières, mais l'utilisation la plus fréquente actuellement est celle de portes intégrées en boîtiers de 14 ou 16 pattes DIL (Dual in Line), réalisées en technologie TTL (il s'agit alors de la célèbre série 74...) ou CMOS (série 4000).

Dans ces technologies, les états logiques correspondent à des niveaux de tension :

en TTL :

niveau logique 0 Tension inférieure à 0,8 volt

niveau logique 1 Tension supérieure à 2,4 volts

(logique dite positive).

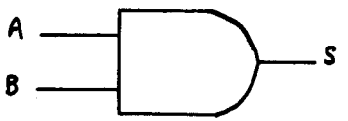
La tension d'alimentation étant impérativement de 5 volts en TTL.

en CMOS : les niveaux logiques sont pratiquement :

niveau logique 0 Tension inférieure à 25 % de la tension d'alim.
niveau logique 1 Tension supérieure à 75 % de la tension d'alim.

La tension d'alimentation pouvant aller de 3 à 18 volts.

PORTE ET (AND anglo-saxon) :



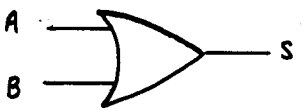
Schéma

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité

Pour une porte NON-ET (NAND), il faut inverser les états de S.

PORTE OU (OR) :

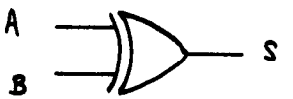


Schéma

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Table de vérité

PORTE OU EXCLUSIF (OR-EX) :



Schéma

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité

ANNEXE 2

BASE DE NUMERATION HEXADECIMALE

En base 16, les chiffres utilisés sont les suivants :

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

A titre d'exemple, nous avons :

$16_d = 10_H$ $256_d = 100_H$ $65535_d = FFFF_H$ d : décimal H : hexadécimal.

Cette base de numération est parfaitement adaptée aux microprocesseurs couramment utilisés.

En effet, les données correspondent à des mots de 8 bits. Ces données pourront s'écrire à l'aide de deux chiffres hexadécimaux compris entre 00_H et FF_H :

$$00000000_B = 0_d = 00_H$$
$$10110010_B = 178_d = B2_H$$
$$11111111_B = 255_d = FF_H.$$

De la même manière, les adresses pourront être très commodément écrites sous forme de 4 chiffres hexadécimaux puisqu'elles correspondent à des mots binaires de 16 bits. Nous aurons :

$$0000000000000000_B = 0_d = 0000_H$$
$$1111111111111111_B = 65535_d = FFFF_H.$$

Dans la littérature spécialisée, les nombres hexadécimaux sont précédés par le signe $: on écrira $E07F par exemple.

ANNEXE 3

FORMAT D'UNE INSTRUCTION EN LANGAGE MACHINE

Nous avons dit que les instructions exécutables par le microprocesseur étaient codées sur un octet.

Dans certains cas, ce seul code suffit pour que la machine sache exactement ce qu'elle a à faire. C'est le cas de l'instruction DEX citée en exemple dans le texte.

Mais supposons maintenant que notre processeur comporte un registre que nous appellerons A, et que l'on veuille transporter le contenu de ce registre dans la mémoire centrale à une adresse bien précise $XXXX.

Notre processeur possède une instruction dans son jeu qui permet de faire cela : supposons que son mnémonique soit STAA

(pour Store A). Lorsque le décodeur va recevoir cet ordre, cela ne lui suffira pas et il attendra (puisqu'il reçoit un octet après l'autre) l'adresse où doit être placé le contenu du registre A.

En définitive, et toujours pour cet exemple, l'instruction complète s'écrira :

STAA $XXXX

et correspondra dans le programme à 3 octets successifs. Le premier octet (code de STAA) est appelé code opération et les deux suivants sont les données.

Le code opération est tel que lorsque le décodeur recevra le code de STAA, il saura que deux octets de données doivent être réceptionnés avant de pouvoir exécuter l'instruction.

BIBLIOGRAPHIE

Les ouvrages sur la micro-informatique et les microprocesseurs sont maintenant très nombreux, et la liste suivante ne donne que quelques exemples.

OUVRAGES :

— H. LILEN. — *Du microprocesseur au micro-ordinateur.* Editions Radio.

— P. MELUSSON. — *Initiation à la micro-informatique : le microprocesseur.* Editions techniques et scientifiques françaises.

— R. ZAKS et P. LE BEUX. — *Les microprocesseurs.* Editions Sybex.

— Cl. PARIOT. — *Introduction aux microprocesseurs et aux micro-ordinateurs.* Editions Dunod Informatique.

— M. AUMIAUX. — *L'emploi des microprocesseurs.* Editions Masson.

— H. LILEN. — *Programmation des microprocesseurs.* Editions Radio.

ARTICLES :

En France, la revue *Micro-Systèmes* a publié quelques articles d'initiation et d'approfondissement sur les microprocesseurs. Citons :

— *L'unité arithmétique et logique.* N° 1 p. 79.

— *Initiation aux microprocesseurs.* N° 1 p. 11.

— *Le cheminement des informations dans un microprocesseur.* N° 2 p. 85.

— *Les trois niveaux de langage.* N° 3 p. 59.

— *L'unité de commande.* N° 3 p. 97.

Ces articles sont très orientés électronique et sont en général d'un niveau élevé.

Il existe également la revue mensuelle *L'ordinateur individuel* qui insiste plutôt sur l'aspect logiciel.