Troisième partie

La programmation des microprocesseurs

Chapitre 10

La programmation des microprocesseurs

La programmation d'un ordinateur répondant à une architecture de von Neumann s'effectue *a priori* en langage machine, comme nous l'avons vu. Le langage machine est le seul compréhensible par la machine mais il est cependant peu ergonomique pour un être humain, aussi a-t-on cherché à remplacer ce langage par d'autres langages plus proches de nous. Ceci est si vrai qu'il est difficile de programmer directement en langage machine sur un système informatique moderne.

Les systèmes d'exploitation tels que Windows, Linux ou MacOs ne permettent pas de programmer pleinement en langage machine car ils utilisent ce qui est appelé le *mode protégé* des microprocesseurs. Un microprocesseur en mode protégé distingue plusieurs niveaux d'utilisation, au minimum deux : le niveau *noyau* qui a le droit d'utiliser toutes les ressources et le niveau *utilisateur* (y compris le super-utilisateur) qui transmet au noyau les paramètres prévus par celui-ci pour exécuter les appels système prévus au niveau noyau. Si le concepteur du système d'exploitation n'a pas prévu l'accès à une instruction du microprocesseur, ou la refuse pour une raison qui lui semble bonne, l'utilisateur ne pourra pas l'utiliser à travers ce système d'exploitation. En particulier, l'utilisateur ne peut pas en général accéder comme bon lui semble à la mémoire vive (pour éviter d'y détruire des données vitales au bon fonctionnement du système d'exploitation).

Pour programmer pleinement en code machine tout en utilisant tous les services d'un système d'exploitation, on doit en choisir un qui n'utilise pas le mode protégé des microprocesseurs. C'est le cas de MS-DOS (pour *MicroSoft Disk Operating System*) conçu pour le micro-ordinateur PC (*Personal Computer*) d'IBM de 1981, à base du 8088, qui ne possède pas de mode protégé de toute façon.

Par politique de compatibilité chez *Intel*, tous les microprocesseurs démarrent comme s'il s'agissait d'un 8086/8088 (mais plus rapidement que l'originel) et on passe en mode protégé ensuite (on parle de *mode réel* pour ce démarrage). On peut donc installer MS-DOS sur n'importe quel compatible PC, ce qui va nous aider dans la suite. Il n'est pas question de rétrograder votre super ordinateur : nous verrons comment démarrer, de façon optionnelle, sous MS-DOS.

Nous allons donc utiliser un compatible PC avec MS-DOS 5 (disponible gratuitement) ou MS-DOS 6.22 (la dernière version fonctionnant en mode réel).

10.1 Microprocesseur, mémoire et périphériques

Nous avons vu dans la deuxième partie les principes sur lesquels reposent les ordinateurs électroniques. De nos jours, la partie essentielle se trouve concentrée dans un circuit intégré appelé *microprocesseur*. La première étape consiste donc à savoir programmer celui-ci.

10.1.1 Une modélisation des ordinateurs

Un micro-ordinateur peut être modélisé de la façon suivante, pour la programmation qui nous intéresse. La partie essentielle en est le *microprocesseur* qui permet d'effectuer des calculs. Celui-ci contient des **registres internes** [1] (*register* en anglais) : ce sont des éléments de mémoire de capacité très limitée qui permettent de sauvegarder quelques valeurs. Le microprocesseur a donc besoin d'accéder à de la **mémoire vive** (ou **mémoire centrale** ; *main memory* en anglais), de plus grande capacité, dans laquelle sont stockées les données, le programme lui-même et les données auxiliaires. Il a également besoin d'**entrée-sortie** pour pouvoir entrer les données et sortir les résultats (et également pour communiquer avec la **mémoire de masse**).

10.1.2 Les registres

Notion.- La notion de *mémoire* est importante pour un ordinateur. On distingue, d'un point de vue matériel, les *registres* qui sont des éléments de mémoire situés sur le microprocesseur lui-même, la *mémoire vive* installés sur la carte mère et la *mémoire de masse* située sur des périphériques (disquettes, disque dur, CD-Rom...). Bien entendu cette distinction, d'origine matérielle, va avoir des retombées sur la programmation.

Nombre de registres.- Seuls deux registres sont indispensables en théorie : l'**accumulateur**, qui contient les résultat des calculs, et le **compteur ordinal**, qui contient le numéro de la prochaine instruction à exécuter. Mais, en général, un microprocesseur comporte beaucoup plus de registres pour accélérer la vitesse de l'ordinateur : ceci évite un passage incessant entre l'accumulateur et un élément de mémoire vive.

Numérotation des bits.- Par convention, on interprète les bits d'un élément de mémoire, et donc d'un registre, de façon à ce que le bit numéroté i contienne le i-ième chiffre binaire (celui correspondant à 2^i) si le contenu est considéré comme étant un entier naturel. Ils sont donc numérotés de 0 à 7 pour un octet.

Registres cachés.- Un registre est un élément mémoire inclus dans le microprocesseur. Les concepteurs d'un microprocesseur peuvent donc décider d'en placer beaucoup pour accélérer tel ou tel point. Certains registres peuvent être utiles pour accélérer les calculs de façon matérielle sans que le programmeur puisse y accéder explicitement. Les registres sur lesquels le programmeur peut agir explicitement reçoivent un nom. Les autres sont appelés **registres cachés**.

10.1.3 Mémoire de masse

Votre expérience de la manipulation des ordinateurs, même si elle est très courte, montre que, sur nos ordinateurs actuels, même la mémoire vive n'est pas suffisante. Et ceci pour deux raisons : d'une part, certaines applications, telles que les bases de données, exigent une très grande

1. Il ne faut pas confondre la notion de *registre* que nous avons vue lors de la modélisation des ordinateurs et ces éléments mémoire. La tradition bien établie de part et d'autre fait que ces deux concepts portent le même nom mais ils n'ont pas grand chose à voir : les registres au sens de la modélisation sont représentés par la mémoire vive.

capacité mémoire et, d'autre part, la technologie utilisée actuellement fait que le contenu de la mémoire vive est perdue après chaque session de travail. Il faut donc utiliser des **mémoires de masse**. Le microprocesseur doit donc posséder un jeu d'instructions pour accéder à celle-ci. Nous verrons que, du point de vue du microprocesseur, il s'agit du même jeu d'instructions que pour les entrées-sorties.

Remarque sur la terminologie.- Nous avons conservé le nom de *mémoire vive* qui est traditionnel mais le qualificatif de *vive* fait référence à une caractéristique des technologies utilisées actuellement. En effet les éléments de mémoire sont réalisés de façon telle que lorsqu'il n'y a plus de courant électrique, ils perdent leur contenu. Ce n'est évidemment pas ce qui est recherché, mais c'est comme ça. Rien ne dit que cette caractéristique perdurera dans l'avenir, par contre on aura toujours besoin de mémoire centrale.

Le qualificatif « *de masse* » correspond également à une caractéristique : la capacité d'une mémoire de masse est plus importante que celle de la mémoire vive (mais d'accès beaucoup plus lent).

On parle aussi quelquefois de **mémoire primaire** et de **mémoire secondaire** au lieu de mémoire vive et de mémoire de masse, mais cela ne change pas grand chose. On ne voit pas *a priori*, d'un point de vue théorique, pourquoi utiliser deux types de mémoire.

FIGURE 10.1 – Boîtier du 8088

10.1.4 Première classification des instructions d'un microprocesseur

On peut donc considérer que les instructions d'un microprocesseur sont de trois sortes, comme dans notre modèle du chapitre un :

- les **instructions de transfert**, à la fois en entrée, en sortie et en ce qui concerne les *registres internes* ;
- les **instructions de calcul**, c'est en général pour elles que l'on utilise un ordinateur ;
- d'autres instructions, plus spécialisées et non théoriquement indispensables, dont le jeu dépend fortement du microprocesseur utilisé.

10.2 Un exemple de microprocesseur : Intel 8088

Nous avons dit que nous prendrions en exemple n'importe quel PC, à base de microprocesseur *Intel*. La compatibilité ascendante des microprocesseurs *Intel* fait qu'ils doivent tous démarrer en mode dit réel, qui se comporte comme un 8088.

Comme nous l'avons vu, la société *Intel* a créé le premier microprocesseur, le 4004 en 1971, avec un bus de largeur 4 (car cela est suffisant pour un chiffre). Elle a conçu ensuite des microprocesseurs 8 bits : le 8008 en 1972 puis le 8080 en 1974. Elle est ensuite passée aux microprocessseurs 16 bits : le 8086, conçu entre 1976 et 1978, et sa variante, le 8088, introduit le premier juillet 1979 (avec un bus des données de 8 bits au lieu de 16, ce qui permet d'en réduire le prix).

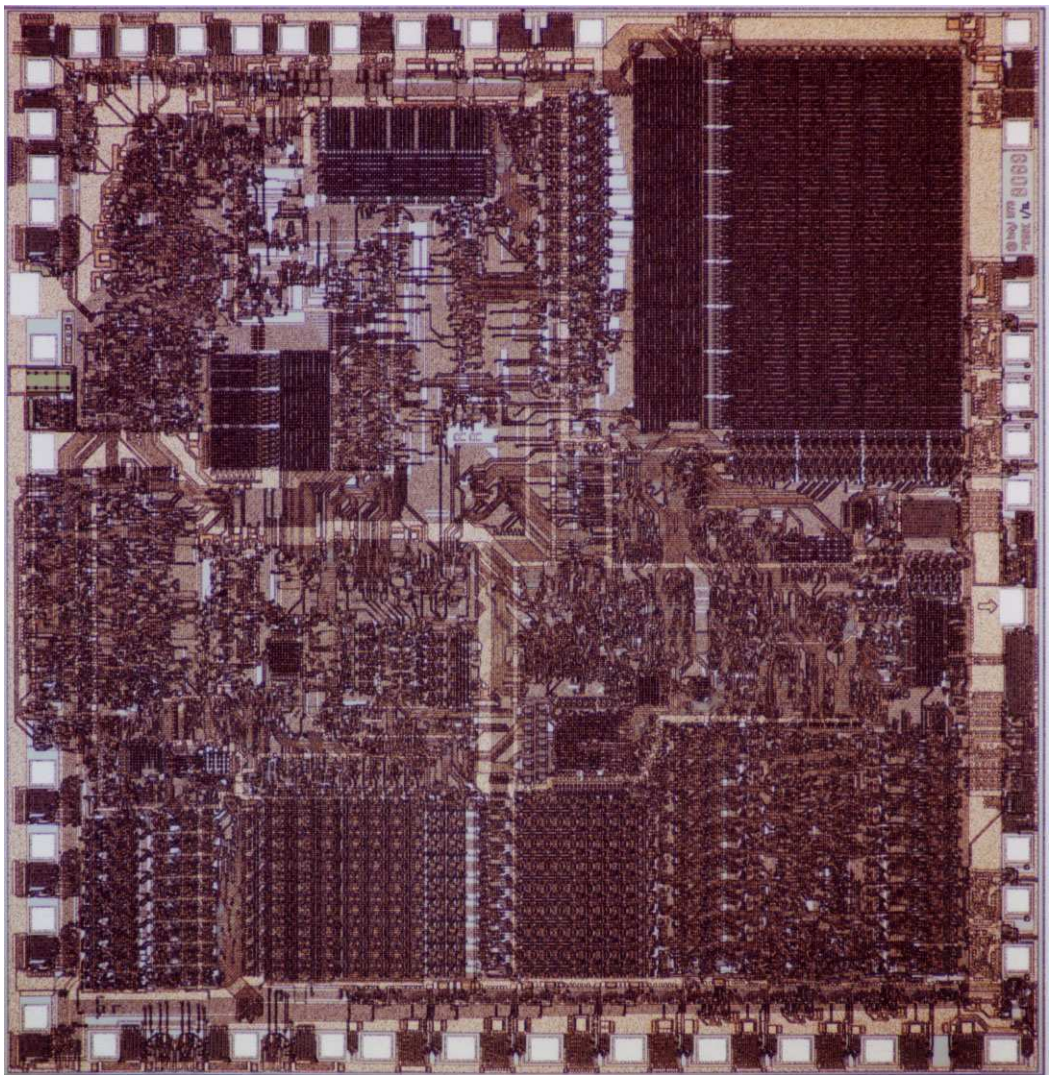

FIGURE 10.2 – Cœur du 8088

10.2.1 Aspect extérieur

Le microprocesseur est placé dans un boîtier duquel sortent 40 **broches**, comme le montre la photo 10.1. On pourra remarquer le détrompeur (l'encoche en forme de demi-disque à gauche), qui aide à bien positionner le microprocesseur.

10.2.2 Les broches

Le lien avec le monde extérieur est réalisé à l'aide de 40 broches (figure 10.3) dont les rôles peuvent être décomposés de la façon suivante :

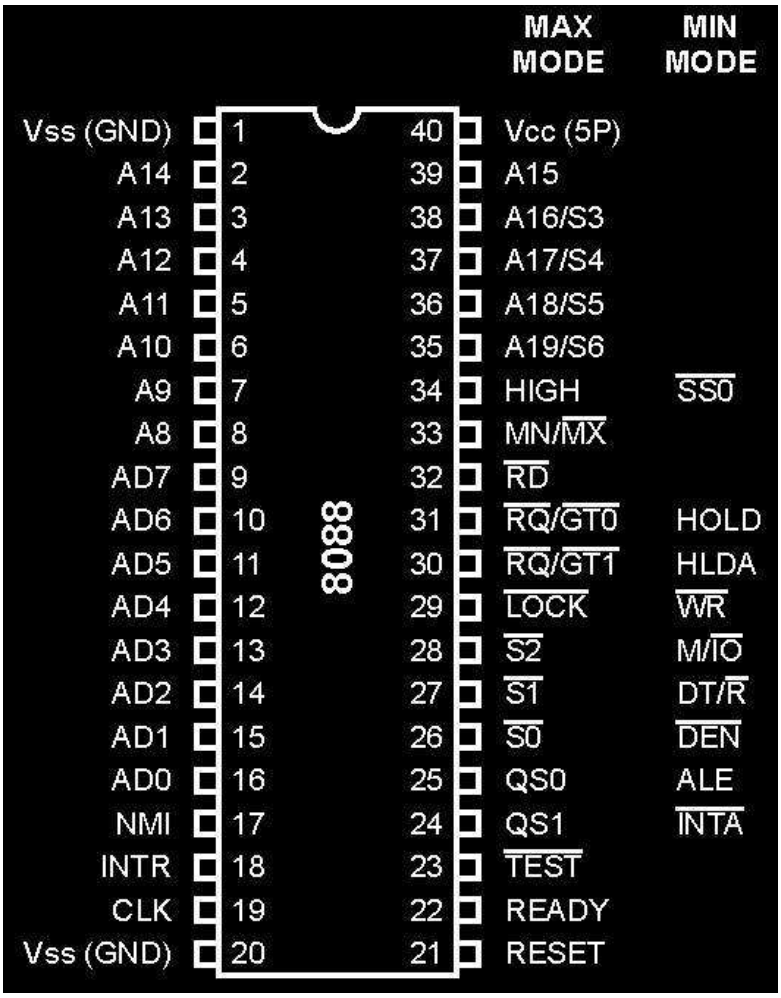

FIGURE 10.3 – Broches du 8088

– Le microprocesseur possède une interface avec le **bus des adresses** de largeur 20 bits (et donc 20 broches), ce bus étant extérieur au microprocesseur. Il permet d'indiquer la case mémoire de la ROM, de la RAM ou le port d'un périphérique dont le contenu doit être lu ou modifié. Il s'agit des broches 2 à 16 (intitulées A0 à A14) et 35 à 39 (intitulées A15 à A19), avec 'A' pour *Address*.

- Le microprocesseur possède une interface avec le **bus des données** de largeur 8 bits (et donc 8 broches), ce bus étant également extérieur au microprocesseur. C'est par lui que transite l'octet à lire ou à écrire. Les broches sont en fait communes (on dit *multiplexées*) avec les huit premières broches du bus des adresses : il s'agit des broches 9 à 16, intitulées AD0 à AD7, avec 'AD' pour *Address/Data*.
- Le microprocesseur possède également une interface avec le **bus de contrôle** de largeur 16 bits (et donc 16 broches, 17, 18 et 21 à 34) qui permet de piloter les liens avec l'extérieur. Par exemple, une des broches indique qu'une donnée cohérente [2] se trouve sur les broches de données.
- Les quatre dernières broches servent, d'une part, à l'alimentation électrique du microprocesseur (la broche 40, intitulée Vcc, et les broches 1 et 20, intitulées GND, pour *GrouND*, terre en anglais) et, d'autre part, au cadencement de celui-ci (réception des tops d'horloge), il s'agit de la broche 19, intitulée CLK pour *CLocK*, horloge en anglais.

10.2.3 Les registres internes

Description.- Le microprocesseur 8088 possède 14 registres internes visibles par le programmeur, tous de capacité 16 bits, mais on peut accéder octet par octet pour les registres de données :

- L'**accumulateur A** est l'un des deux registres importants avec le pointeur de programme. Nous en avons vu le rôle fondamental dans notre modélisation des ordinateurs. On peut accéder directement aux 16 bits qui le constitue, sous la dénomination **AX** (avec un 'X' pour *eXtended*). On peut aussi accéder à l'octet de poids faible (sous la dénomination **AL** avec un 'L' pour *Low*) et à l'octet de poids fort (sous la dénomination **AH** avec un 'H' pour *High*).
- Le pointeur de programme PC (renommé **IP** pour l'anglais *Instruction Pointer* car il exige une autre donnée, le contenu du registre CS pour déterminer l'emplacement de l'instruction), l'autre registre fondamental avec l'accumulateur, n'est pas visible par l'utilisateur.
- On a, par contre, outre l'accumulateur, trois autres **registres de données**, dénommés **B**, **C** et **D**, qui permettent de détenir des données auxiliaires. On y accède par les noms **BX**, **BL**, **BH**, **CX**, **CL**, **CH**, **DX**, **DL** et **DH**.
- Le **registre des indicateurs** (*Flags*, drapeaux en anglais), sans nom car on ne peut pas y accéder directement, permet de détenir des indicateurs pour certains événements qui se produisent lors des instructions, par exemple le dépassement de capacité lors d'une addition.
- Quatre registres servent pour les pointeurs et les index. Ils sont spécialisés :
  - Le **pointeur de pile de sauvegarde SP** (pour l'anglais *Stack Pointer*) permet, comme nous le verrons plus tard, le stockage provisoire dans la RAM des données et des adresses de retour des sous-programmes.
  - Le **pointeur de base BP** (pour l'anglais *Base Pointer*) permet de déterminer le premier élément d'un tableau.
  - L'**index de source SI** (pour l'anglais *Source Index*) permet de repérer le premier élément d'une zone mémoire à copier.
  - L'**index de destination DI** (pour l'anglais *Destination Index*) permet de repérer le premier élément d'une zone mémoire dans laquelle on veut copier une autre zone mémoire.
- Nous verrons que la gestion de la mémoire exige quatre registres supplémentaires :

2. Ce qui n'est pas le cas en permanence. Nous avons vu, lors de l'étude de la réalisation des registres, que ceux-ci contiennent une donnée sauf durant une période transitoire.

– Le **registre de segment de code CS** (pour l'anglais *Code Segment*).
– Le **registre de segment de données DS** (pour l'anglais *Data Segment*).
– Le **registre de segment de pile SS** (pour l'anglais *Stack Segment*).
– Le **registre de segment supplémentaire ES** (pour l'anglais *Extra Segment*).

Origine des noms des registres.- Les noms des registres de données ne sont pas donnés par hasard. Ils rappellent les contextes essentiels dans lesquels ils sont utilisés : les lettres **B**, **C** et **D**, outre le jeu sur l'ordre alphabétique, sont les initiales de 'Base', 'Compteur' (en fait *Counter*) et 'Données' (en fait *Data*).

10.2.4 Les unités du microprocesseur

D'un point de vue fonctionnel, le microprocesseur 8088 peut être représenté comme sur la figure 10.4.

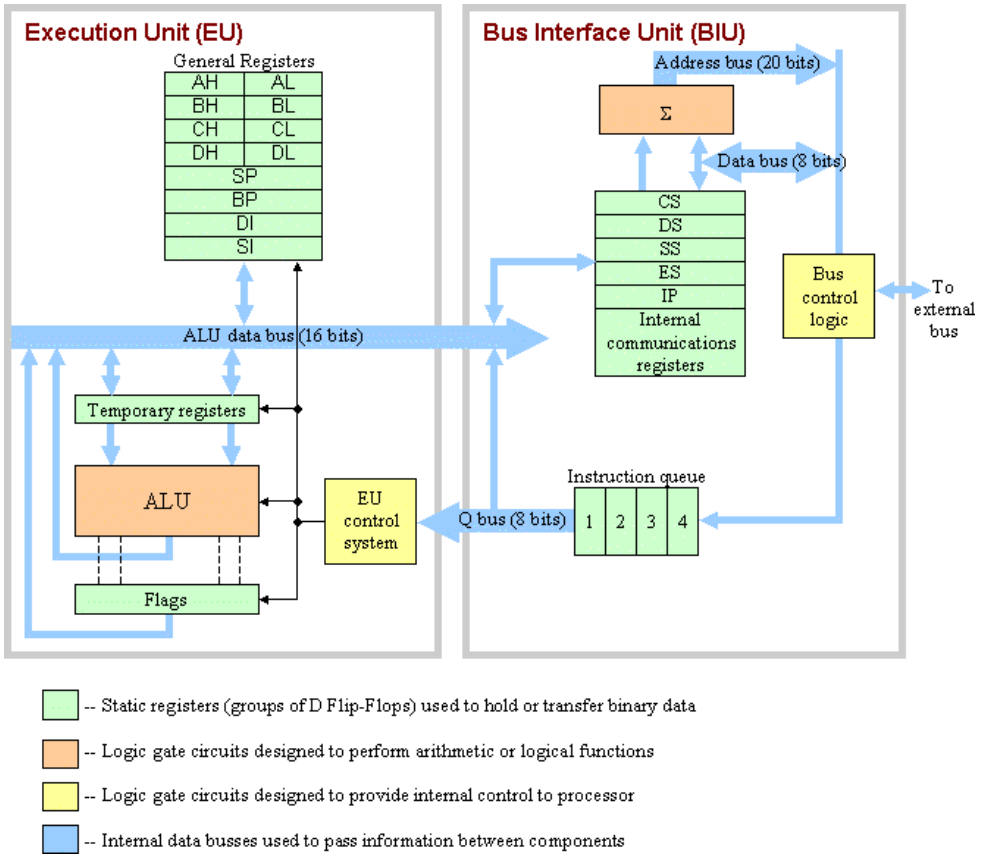

FIGURE 10.4 – Structure fonctionnelle du 8088

Le 8088 possède deux unités fonctionnelles, fonctionnant en parallèle :
– pendant que l'**unité d'exécution EU** (*Execution Unit* en anglais) exécute une instruction,
– l'**unité d'interfaçage de bus BIU** (pour l'anglais *Bus Interface Unit*) appelle la suivante.

L'EU contient une **unité arithmétique et logique UAL** (pour l'anglais *Arithmetic and Logical Unit*), chargée d'exécuter ces types d'opérations.

10.3 Façons de programmer un microprocesseur

Vous avez acquis un microprocesseur. Comment le programmer ? Ceci n'est pas simple dans la mesure où le microprocesseur est un circuit électronique conçu pour fonctionner avec d'autres circuits électroniques et non de façon isolée.

10.3.1 Programmation physique du microprocesseur

Si on ne dispose que du microprocesseur, il n'y a pas de miracle : on envoie des impulsions électriques bien choisies sur les broches qu'il faut et on recueille les impulsions sur les broches qu'il faut.

Ce cours est orienté programmation et non construction d'ordinateur. Nous allons donc donner le principe de programmation physique sans entrer dans les détails. Les étudiants en électronique y consacreraient, par contre, une séance de travaux pratiques.

10.3.2 Kit spécial

En général on ne se contente pas de n'utiliser que le microprocesseur. On lui adjoint de la mémoire et quelques périphériques qui permettent de placer le programme et les données en mémoire et de dire d'exécuter le programme. On récupère les résultats en mémoire. Il faut un dispositif pour aller lire le contenu de la mémoire.

Rappelons que nous avons présenté au chapitre 9 une telle carte (figure 9.21) appelée KIM-1, développée en 1977 par *MOS Technology* et équipée d'un microprocesseur 6502.

10.3.3 Programmation sur un système informatique

On appelle **système informatique** l'ensemble constitué d'un ordinateur, de ses logiciels (système d'exploitation, compilateurs...) et de ses périphériques (imprimantes...). Le plus simple pour programmer un microprocesseur est d'utiliser un système informatique à base de ce microprocesseur.

Malgré cela, la programmation en langage machine n'est plus la priorité de nos jours, aussi est-ce quand même assez difficile d'y avoir accès, y compris sur un système informatique.

10.3.3.1 Accès grâce au BIOS : lancement d'une disquette ou d'une clé USB

Le logiciel minimum est le BIOS, programme lancé lors du démarrage du micro-ordinateur. Celui-ci recherche quelque chose (un système d'exploitation de nos jous, un interpréteur BASIC au tout début des micro-ordinateurs) sur un périphérique : lecteur de disquette, disque dur, CD-ROM, clé USB..., plus exactement sur le 'premier secteur' de celui-ci.

Bien entendu ce qui est recherché est écrit en langage machine. On peut donc écrire un programme en langage machine, le placer sur le premier secteur d'une disquette (d'une clé USB de nos jours) et relancer l'ordinateur. Il exécutera ce qu'on aura placé sur le média.

Cette façon de faire présente deux inconvénients. Les lecteurs de disquettes n'existent plus sur les ordinateurs actuels : on peut cependant y substituer une clé USB. Il faut utiliser des outils (logiciels) pour placer le programme sur les secteurs adéquats du média.

Nous réservons l'étude de cette façon de faire à plus tard.

10.3.3.2 Accès grâce à MS-DOS

Les premiers micro-ordinateurs étaient tous munis d'un interpréteur d'un langage de programmation alors populaire : le BASIC. Ce BASIC permettait d'écrire des sous-programmes en langage machine.

Il s'agit d'ene des façons les plus simples d'accèder à la programmation en langage machine. Ceci ne peut pas se faire avec les systèmes d'exploitation modernes (Linux ou Windows) car ils utilisent le mode dit protégé du microprocesseur qui ne permet pas d'accéder directement à la mémoire. C'est pourquoi nous allons revenir au premier système d'exploitation de l'IBM-PC, le MS DOS.

10.4 Historique

Comme nous l'avons vu dans un chapitre antérieur, la société *Intel* a créé le premier microprocesseur : le 4004 en 1971. Elle a conçu ensuite le 8008 en 1972 puis le 8080 en 1974. Le 8080 présente quelques inconvénients : il nécessite trois tensions différentes et deux circuits intégrés supplémentaires pour la production des signaux de commande et de synchronisation. La société *Zilog*, créée par des concepteurs du 8080 démissionnaires de la société *Intel*, développe le Z80 en 1974 en rémédiant à ces deux défauts tout en restant compatible avec le 8080 et en ajoutant des fonctionnalités nouvelles : les 12 opcodes non utilisés sur le 8080 concernent maintenant des instructions sur les chaînes de caractères. C'est une réussite puisqu'il devient alors très rapidement le microprocesseur 8 bits le plus répandu, talonné par le 6502.

Intel se retrouve alors un des concepteurs de microprocesseurs parmi d'autres. Le 8086 va le propulser comme le premier concepteur mondial. Les espoirs de la société reposent alors sur le projet du 8800 (devenu plus tard iAPX 432), avec un bus de données de 32 bits alors que les processeurs d'alors ont tous un bus de 8 bits. Mais le projet a beaucoup de retard car la conception complexe est difficile à implémenter avec la technologie des circuits intégrés d'alors. Tout en gardant espoir dans ce projet, la direction d'*Intel* prend conscience qu'elle doit répondre à *Zilog* et demande en mai 1976 à Stephen Morse, né en 1940 et engagé en mai 1975, qui l'a impressionnée par un examen critique des défauts de conception du 8800, de concevoir un microprocesseur 16 bits, ou plus exactement son architecture, l'implémentation étant confiée à un autre ingénieur, McKewitt. La dernière révision de l'évaluation du 8800 par Morse est datée du 14 avril 1976. Le 13 août, trois mois après avoir débuté sur le projet, est publiée la version 0 du jeu d'instructions, ainsi que la structure des registres, l'espace des entrées-sorties, le mécanisme d'interruption et les modes d'adressage de la mémoire. Ces spécifications sont écrites à un haut niveau. Morse écrit un document intulé *8086 Architectural Specifications* et McKewitt un document compagon appelé *8086 Device Specifications*.

Morse quitte *Intel* en mars 1979. Quelques semaines après son départ, *Intel* sort le 8088, l'analogue du 8086 mais avec un bus des données de 8 bits pour pouvoir concurrencer favorablement le Z80. Lorsque, deux ans plus tard, IBM commence à travailler à son modèle 5150, le premier PC de la firme ne comprend que des composants à bas coût. IBM veut un microproceseur 16 bits pour lequel elle a trois candidats : le Motorola 68000 (le puissant microprocesseur au cœur du premier MacIntosh), le 8086 et le 8088. David J. Bradley, l'un des concepteurs du PC, explique que le Motorola est éliminé car IBM est plus familier avec les microprocesseurs d'*Intel*. De plus *Microsoft* a un interpréteur BASIC disponible pour le 8086 et donc, puisqu'ils partagent le même jeu d'instructions, pour le 8088, qui sera choisi car le moins onéreux.

10.5 Bibliographie

[Edw-08] Benj EDWARDS, *Stephen Morse : Father of the 8086 Processor*, **PCWorld**, June 17, 2008 7 :00 am. Disponible en ligne :

`http://www.pcworld.com/article/146917/stephen_morse_father_of_the_8086_processor.html`

10.6 Appendice : MS-DOS sur une clé USB

Comme nous l'avons dit, nous allons étudier les instructions du microprocesseur 8088 en se plaçant sur un PC muni du système d'exploitation MS-DOS. Si vous disposez déjà d'un système informatique avec MS-DOS (en mode réel, c'est-à-dire jusqu'à la version 6.22), que ce soit un ordinateur auxiliaire ou un double démarrage, il est inutile d'installer MS-DOS. Mais c'est rare de nos jours.

Si vous disposez d'un ordinateur avec un lecteur de disquettes, vous pouvez partitionner votre disque dur et installer MS-DOS sur la première partition. Mais c'est également rare de nos jours. Cela peut également se faire, en jonglant un peu, si l'ordinateur dispose d'un lecteur de CD-ROM.

Le cas le plus fréquent est un ordinateur avec un PC avec un microprocesseur *Intel* puissant, le plus souvent 64 bits, et, pour ce qui nous intéresse, des connecteurs USB.

Nous allons donc installer MS-DOS sur une clé USB. On choisira soit MS-DOS 5.0 disponible gratuitement, soit MS-DOS 6.22, toujours sous licence Microsoft mais disponible gratuitement, comme tous les autres outils de développement Microsoft, pour les étudiants dans un département abonné au système MSDN (ou par d'autres moyens, illicites cependant).

10.6.1 Première étape : récupérer MS-DOS

Il faut récupérer le fichier `en_msdos22.exe` sur MSDN (ou de façon illicite). Placer-le dans un répertoire vide, par exemple de nom '`en_MSDOS622`', et exécuter (sous Windows) ce fichier auto-extractible. On obtient deux sous-répertoires : '`DISKS`' et '`UPGRADE`'.

Ces répertoires contiennent à la fois des fichiers au nom bizarre et des fichiers exécutables qui nous intéresseront.

10.6.2 Seconde étape : créer une clé USB bootable

Une **clé bootable** est une clé USB qui, lorsqu'on démarre l'ordinateur avec la clé dans un des connecteurs USB (et en spécifiant au BIOS de commencer par chercher un système d'exploitation sur une telle clé, avant d'aller chercher sur un disque dur ou un CD-ROM), charge le système d'exploitation qui se trouve sur celle-ci.

Nous verrons plus tard comment on crée une clé bootable. Utilisons pour le moment un utilitaire pour cela. Spécifier '`HP Drive Key Boot Utility v2.1.8 download`' sur votre moteur de recherche favori. Ceci ne renvoie plus au site de Hewlett-Packard car l'utilitaire a évolué (mais la nouvelle version ne convient pas). On peut le récupérer, par exemple sur le site Web :

`http://files.extremeoverclocking.com/file.php?f=197`

en cliquant sur '*Primary Download Site*', ce qui permet de télécharger le fichier '`SP27608.exe`' de 1,97 MiO. Placer-le dans un répertoire vide de votre Windows, par exemple '`HP DriveKey`' et exécuter-le. On se retrouve avec quatre fichiers, dont `HPUSBFW.exe` pour *HP USB Disk Storage Format Tool FreeWare*.

Préparer un autre répertoire vide de votre Windows, nommé par exemple '`MSDOS622`' et y placer les trois fichiers '`command.com`', '`io.sys`' et '`msdos.sys`' trouvés dans le sous-répertoire '`UPGRADE`' de '`en_MSDOS622`'. Attention ! ces fichiers sont « cachés » car considérés comme fichiers système : il faut aller dans '`Panneau de configuration`', '`Apparence et personnalisation`' puis '`Afficher les fichiers et dossiers cachés`') pour désélectionner, tout au moins de façon temporaire, l'option '`Masquer les fichiers protégés du système d'exploitation (recommandé)`' afin de rendre visible les fichiers système.

Insérez une clé USB (de moins de 2 GiO) dans un connecteur USB et exécuter l'utilitaire `HPUSBFW.exe` en tant qu'administrateur. On obtient la fenêtre de la figure 10.5. Vérifier que le volume indiqué dans '`Device`' est bien celui de la clé que vous voulez rendre bootable. Choisir le système de fichier FAT16, c'est-à-dire 'FAT'. Choisir éventuellement un nom de volume autre que celui indiqué par défaut. Cliquer sur '*Create a DOS startup disk*' puis sur '*using DOS system files located at :*'. Appuyez sur les trois petits points pour rechercher le dossier '`MSDOS622`' précédemment préparé puis sur le bouton '*Start*'. La fenêtre de la figure 10.6 s'ouvre. Après une dernière vérification qu'il s'agit bien de la clé USB voulue, appuyer sur le bouton '*Oui*'.

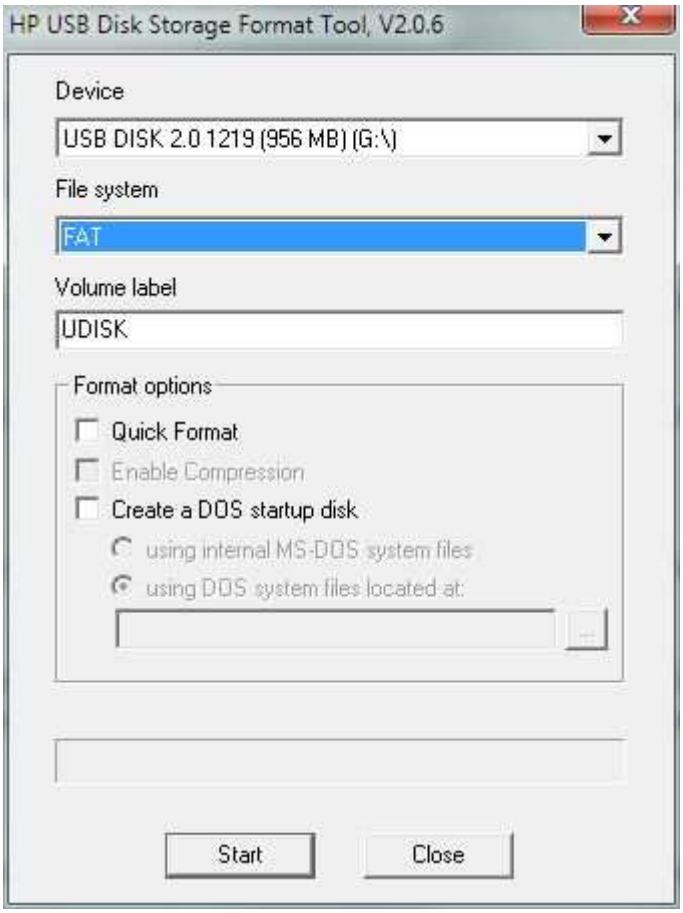

FIGURE 10.5 – Fenêtre de `HP USB Disk Storage Format Tool`

Une fois l'opération terminée, redémarrer l'ordinateur en laissant la clé USB dans le connecteur. Il va falloir indiquer au BIOS de booter sur une clé USB en changeant l'ordre du choix de périphérique de boot. Si tout s'est bien passé, on se retrouve avec MS-DOS (avec clavier Qwerty). Plus exactement on voit apparaître :

```
Starting MS-DOS...

Current date is Wed 01-09-2013
Enter new date (mm-dd-yy):
```

Appuyer sur le touche '`retour`'. On a alors :

```
Starting MS-DOS...

Current date is Wed 01-09-2013
Enter new date (mm-dd-yy):
Current time is 11:01:58.32a
Enter new time:
```

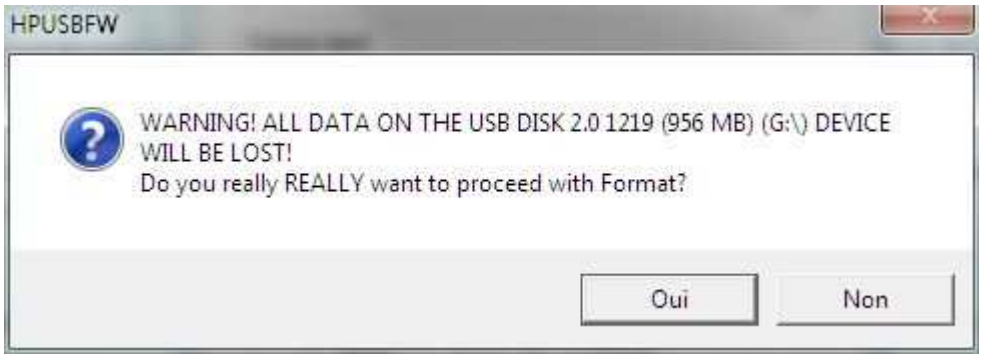

FIGURE 10.6 – Fenêtre d'avertissement de `HP USB Disk Storage Format Tool`

Appuyer sur le touche '`retour`'. On se retrouve avec :

```
Starting MS-DOS...

Current date is Wed 01-09-2013
Enter new date (mm-dd-yy):
Current time is 11:01:58.32a
Enter new time:


Microsoft(R) MS-DOS(R) Version 6.22
            (C)Copyright Microsoft Corp 1981-1994

C:\>
```

On se retrouve avec le prompteur que l'on trouve sur l'émulateur de ligne de commande '`command.com`' de Windows.

On peut commencer à explorer MS-DOS mais on se trouve dans la version anglaise avec clavier querty.

10.6.3 Troisième étape : version française complète

Pour la suite on va se placer dans la version avec clavier azerty et ajouter les utilitaires avec lesquels nous allons travailler, en particulier '`debug`'.

Enlever la clé USB er redémarrer l'ordinateur pour se retrouver avec Windows. Placer la clé dans un connecteur USB. Créer un répertoire '`DOS`' sur la clé et y placer les fichiers '`COUNTRY.SYS`', '`DEBUG.EXE`', '`EDIT.COM`', '`KEYB.COM`', '`KEYBOARD.SYS`', '`MORE.COM`' et '`QBASIC.EXE`' du sous-répertoire '`UPGRADE`' de '`en_MSDOS622`'.

Préparer un fichier de nom '`AUTOEXEC.BAT`' à placer à la racine de la clé USB :

```
@ECHO OFF
PROMPT $p$g
PATH C:\DOS
KEYB FR,,C:\DOS\KEYBOARD.SYS
```

Préparer aussi un fichier de nom '`CONFIG.SYS`' à placer également à la racine de la clé USB :

```
COUNTRY=033,850,C:\DOS\COUNTRY.SYS
```

Redémarrer l'ordinateur en laissant la clé USB dans le connecteur. On voit maintenant apparaître :

```
Starting MS-DOS...

C:\>
```

On peut maintenant utiliser MS-DOS avec clavier azerty.