1. Principe de fonctionnement

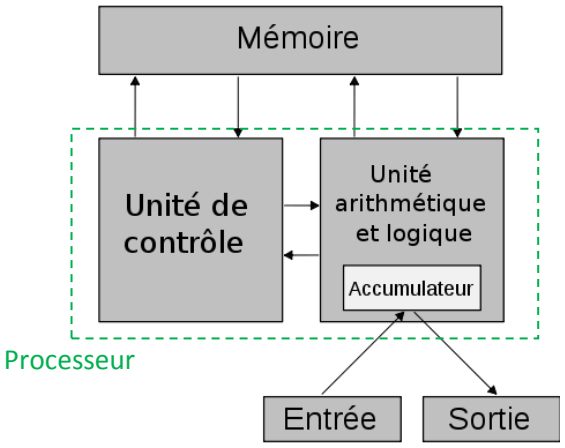
1.1. Machine de Von Neumann

1.1.1. Principe

1943 : Seconde guerre mondiale : le COLOSSUS 1, calculateur électromécanique conçu par Max Newman, voit le jour avec un premier langage de *programmation binaire* inventé par Alan Turing, afin de casser les codes allemands. Von Neumann (IBM) introduit les branchements conditionnels (rupture des instructions séquentielles) : en quelque sorte, le "GOTO" est né...

Les instructions d'un programme ?

Cartes => Rubans => relais électromécaniques = tubes => transistors => mémoire centrale



Wikipédia

Von Neumann propose en **1945** un principe d'architecture, modèle de base pour un ordinateur qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données requises ou générées par le calcul.

1.1.2. Architecture

L'architecture de Von Neumann est une circuiterie implémentant un ordinateur universel contenant les composants suivants :

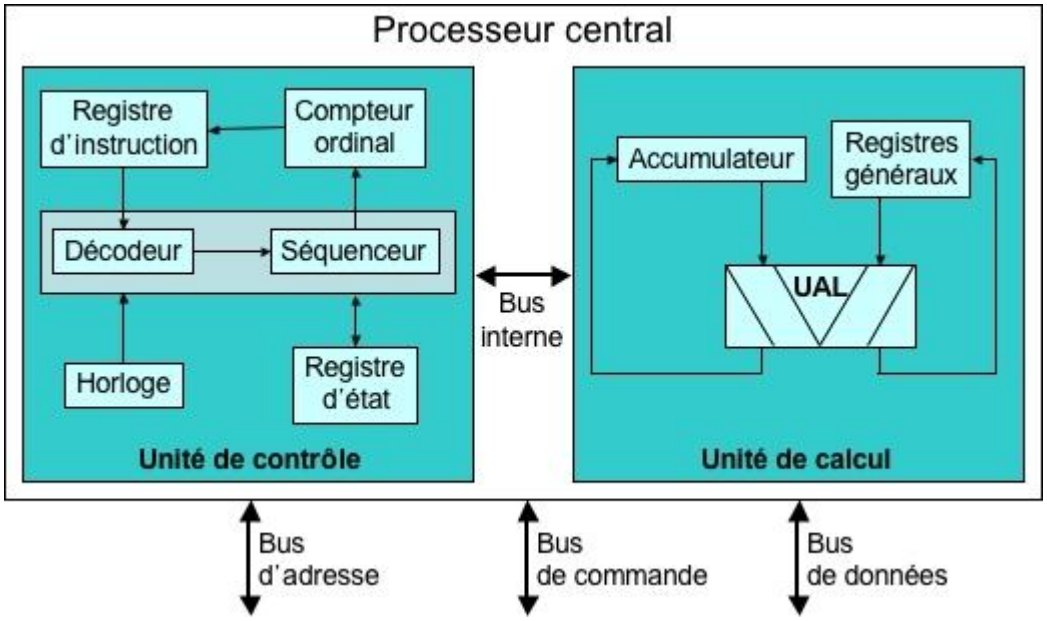
- Une unité arithmétique et logique qui procède à des opérations de calcul et des opérations logiques.
- Une unité de contrôle qui pilote l'exécution des programmes.
- Une mémoire dans laquelle sont stockés les programmes et les données.
- Une unité d'entrée qui permet l'acquisition de données en provenance du monde extérieur.
- Une unité de sortie qui permet l'envoi de résultats à destination du monde extérieur.

1.2. Architectures CISC et RISC

Un microprocesseur à jeu d'instruction étendu, ou *complex instruction set computer (CISC)* en anglais, désigne un microprocesseur possédant un jeu d'instructions comprenant de très nombreuses instructions mixées à des modes d'adressages complexes (plusieurs cycles d'horloge).

Le microprocesseur à jeu d'instruction réduit *(RISC)*ou *reduced instruction-set computer* en anglais est une architecture matérielle de microprocesseurs. On l'a opposé à la fin des années 1980 et au début des années 1990 à l'architecture CISC (*complex instruction-set computer*). La sortie d'architectures hybrides comme le *Pentium* (CISC émulé par du RISC) a mis fin, par disparition de repères.

2. Structure interne d'un microprocesseur



INFORX2ECF

Architecture simplifiée d'un processeur central

- L'*unité de contrôle* (en anglais : Control Unit - CU) est la partie la plus complexe du processeur. Elle se décompose en plusieurs parties dont elle doit assurer la coordination :
  - Le processeur exécute une à une les instructions stockées dans la mémoire centrale. Pour cela, les instructions doivent être chargées dans le processeur. Elles le sont dans le *registre d'instruction*. Ce dernier contient donc l'instruction courante à exécuter.
  - L'instruction à exécuter, chargée dans le registre d'instruction, est interprétée par le *décodeur*.
  - Le *séquenceur* est alors capable d'ordonnancer les diverses opérations élémentaires du processeur, nécessaires pour exécuter l'instruction, grâce à un microprogramme.
  - Le *compteur ordinal* est un registre particulier qui contient à tout instant, l'adresse de l'instruction suivante à exécuter.
  - L'*horloge* est un dispositif qui détermine le rythme dans lequel sont exécutées les instructions. Elle fournit un signal régulier au processeur.
  - Le *registre d'état* est, comme le compteur ordinal, un registre particulier. Il représente à tout moment l'état du processeur. En effet, divers évènements peuvent créer des situations « exceptionnelles ». Par exemple, l'addition de deux valeurs peut dépasser les capacités de représentation du processeur. Dans ce cas, un bit du registre d'état signale que le processeur est dans l'état « overflow ».
  - Le processeur contient encore d'autres registres spéciaux présentés dans le paragraphe suivant.
- L'*unité de calcul*, comme son nom l'indique, effectue tous les calculs au sein du processeur. À côté des opérations arithmétiques, elle peut aussi procéder à des opérations logiques. C'est grâce à

cette fonctionnalité que l'ordinateur est capable d'exécuter des structures de contrôle contenant des conditions.

- L'*accumulateur* est le registre de calcul par excellence. C'est par lui en effet que transitent toutes les données devant faire l'objet d'une opération ainsi que tous les résultats produits par calcul arithmétique.
- Comme ces opérations mettent généralement en jeu plusieurs termes et/ou états intermédiaires, l'accumulateur est secondé par un nombre variable de *registres généraux* destinés à la réalisation matérielle des calculs.
- Comme son nom l'indique, l'*unité arithmétique et logique* (en anglais : Arithmetic and Logic Unit - ALU) se charge de réaliser les opérations arithmétiques et logiques.

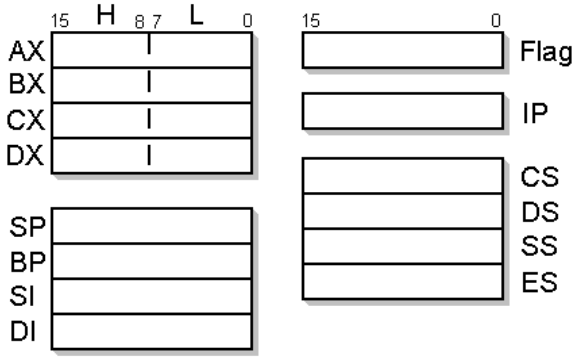
- Pour communiquer avec son environnement, le processeur dispose d'un ensemble de « bus » :
  - Le *bus d'adresses* permet au processeur de désigner l'adresse d'un octet en mémoire. Selon le cas, il peut s'agir de l'adresse d'un ou de plusieurs octets à charger dans le processeur ou à stocker en mémoire.
  - Le *bus de données* permet de désigner la valeur à charger dans le processeur ou à stocker en mémoire.
  - Enfin, le *bus de commande* permet au processeur de désigner l'opération à effectuer, chargement dans le processeur ou stockage en mémoire.

Processeur Intel Core 2 Extreme Dual-Core

2.1. Registre de données et d'adresse

Dans l'architecture x86, le processeur comporte quatre registres de travail, six registres d'offset, six registres de segment, 1 registre d'état (*flags*) et un pointeur d'instruction.

Les registres de travail



Les registres du microprocesseur Intel 8086

Les quatre registres de travail sont principalement utilisés pour stocker des résultats :

1. EAX : registre accumulateur (*accumulator register*). Utilisé pour les opérations arithmétiques et le stockage de la valeur de retour des appels systèmes.
2. EDX : registre de données (*data register*). Utilisé pour les opérations arithmétiques et les opérations d'entrée/sortie.
3. ECX : registre compteur (*counter register*)
4. EBX : registre de base (*base register*). Utilisé comme pointeur de donnée (située dans DS en mode segmenté).

Ce sont des registres 32 bits; pour des raisons historiques, les 16 bits de poids faible sont constitués respectivement des registres AX, DX, CX et BX.

Ces 4 registres 16 bits sont également décomposés en 8 registres de 8 bits :

1. AL : octet de poids faible de AX
2. AH : octet de poids fort de AX
3. BL : octet de poids faible de BX
4. BH : octet de poids fort de BX
5. CL : octet de poids faible de CX
6. CH : octet de poids fort de CX
7. DL : octet de poids faible de DX
8. DH : octet de poids fort de DX

Les registres d'offset

Les registres d'offset sont utilisés lors de l'adressage indirecte de la mémoire (pointeurs). Ces registres complémentaires sont :

1. EBP : (*Extended Base Pointer*) pointeur de base
2. ESP : (*Extended Stack Pointeur*) pointeur de pile
3. ESI : (*Extended Source Index*) pointeur source
4. EDI : (*Extended Destination Index*) pointeur destination

Le nom des deux derniers registres vient du fait qu'ils sont utilisés pour la copie d'une zone mémoire vers une autre.

Les registres de segment

La gestion de la mémoire dans l'architecture x86 est particulière : celle-ci est divisée en segments. Les registres de segment permettent d'accéder, soit au segment de programme qui est la zone mémoire des instructions de programme, soit au segment de données (zone mémoire contenant les données du programme), ou encore au segment de pile.

1. CS: pointe vers les instructions du programme (*code segment*).
2. DS : pointe vers les données du programme (*data segment*).
3. SS : pointe vers la pile programme (*stack segment*).
4. ES : pointe vers les données du programme multi-segments (*extra segment*).
5. FS : pointe vers les données du programme multi-segments en mode protégé.
6. GS : pointe vers les données du programme multi-segments en mode protégé.

La gestion de mémoire en multi-segments permet d'utiliser tous les registres d'offset. La plupart des systèmes d'exploitation actuels utilisent un mode protégé où tous les registres de segment pointent vers le même segment.

Le registre d'états

Chaque bit du registre EFLAGS est un indicateur d'état qui peut être modifié à chaque instruction exécutée :

- retenue (addition ou soustraction),
- dépassement,
- comparaison,
- autoriser les interruptions,
- ...

Le pointeur d'instruction

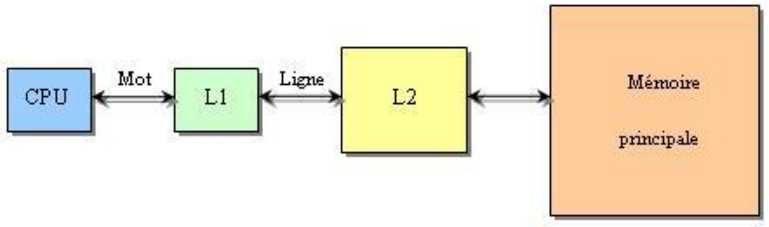
Le registre EIP est utilisé avec le segment du registre CS par le processeur pour connaitre la prochaine instruction à exécuter. Ce registre est donc modifié implicitement par le processeur (instruction suivante, saut à l'adresse indiquée, appel d'une fonction, interruption ...).

2.2. Mémoire cache

Une **mémoire cache** ou **antémémoire** est, en informatique, une mémoire relativement petite et rapide qui stocke les informations les plus utilisées d'une autre mémoire plus grande et plus lente. Elle sert à accélérer les traitements, afin de retrouver des données plus rapidement. Les données peuvent par exemple être un programme, un bloc d'image à traiter, etc... La source peut par exemple être un disque dur, la mémoire centrale, etc... Elle n'affecte pas le nombre d'opérations par seconde qu'un processeur est capable d'effectuer à moins que l'algorithme à exécuter implique des accès répétitifs à de petites zones de mémoires (un bout de programme qui se répète, un travail sur une sous partie d'un fichier son, etc...) ou qu'on soit capable de prédire les besoins futurs pour remplir la mémoire cache en parallèle d'un calcul (*prefetching*), de sorte qu'elle contiendra au moment venu une copie locale des données à accès beaucoup plus rapide. Pour des raisons de coût, les ordinateurs bas de gamme (consoles, ...) possèdent généralement un cache de taille moins élevée.

La mémoire cache est notamment utilisée entre le processeur et la mémoire vive, mais on peut en trouver entre tout fournisseur de données (réseau informatique, disque dur, mémoire principale) et le consommateur de ces données. Wikipédia

Elle est très rapide, mais aussi très chère. Il s'agit souvent de SRAM.



Différents niveaux de mémoire d'un microprocesseur

La présence de mémoire cache permet d'accélérer l'exécution d'un programme. De ce fait, plus la taille de la mémoire cache est grande, plus la taille des programmes accélérés peut être élevée. C'est ainsi un élément souvent utilisé par les constructeurs pour faire varier les performances d'un produit sans changer d'autres matériels. Par exemple, pour les microprocesseurs, on trouve des séries bridées (avec une taille de mémoire cache volontairement réduite), tels que les Duron chez AMD ou Celeron chez Intel, et des séries haut de gamme avec une grande mémoire cache comme les processeurs Opteron chez AMD, ou Pentium 4EE chez Intel. Autrement dit, la taille de la mémoire cache résulte d'un compromis coût/performance.

En programmation, pour profiter de l'accélération fournie par cette mémoire très rapide, il faut que les parties de programme tiennent le plus possible dans cette mémoire cache. Comme elle varie suivant les processeurs, ce rôle d'optimisation est souvent dédié au compilateur. Cela dit, un programmeur chevronné peut écrire son code d'une manière qui optimise l'utilisation du cache.

C'est le cas des *boucles très courtes* qui tiennent entièrement dans les caches de données et d'instruction, par exemple le calcul suivant (écrit en langage C) :

```
long i;    double s;    s=0;    for (i = 1; i<50000000;i++) s+=1./i;
```

2.3. Pipeline d'instruction

Dans la microarchitecture d'un processeur, un ***pipeline*** est une technique de conception des processeurs où l'exécution des instructions est découpée en étages, et où à un instant donné, chaque étage peut exécuter une instruction. Le premier ordinateur à utiliser cette technique est l'IBM Stretch, conçu en 1958.

Soit un processeur où 5 cycles sont nécessaires pour accomplir une instruction[1] :

1. IF (Instruction Fetch) charge l'instruction à exécuter dans le pipeline.
2. ID (Instruction Decode) décode l'instruction et adresse les registres.
3. EX (Execute) exécute l'instruction (par la ou les unités arithmétiques et logiques).
4. MEM (Memory), dénote un transfert depuis un registre vers la mémoire dans le cas d'une instruction du type STORE (accès en écriture) et de la mémoire vers un registre dans le cas d'un LOAD (accès en lecture).
5. WB (Write Back) stocke le résultat dans un registre. La source peut être la mémoire ou bien un registre.

En supposant que chaque étape met 1 cycle d'horloge pour s'exécuter, il faut normalement 5 cycles pour exécuter une instruction, 15 pour 3 instructions :

wikipedia

Séquençage des instructions dans un processeur sans pipeline. Il faut 15 cycles pour exécuter 3 instructions.

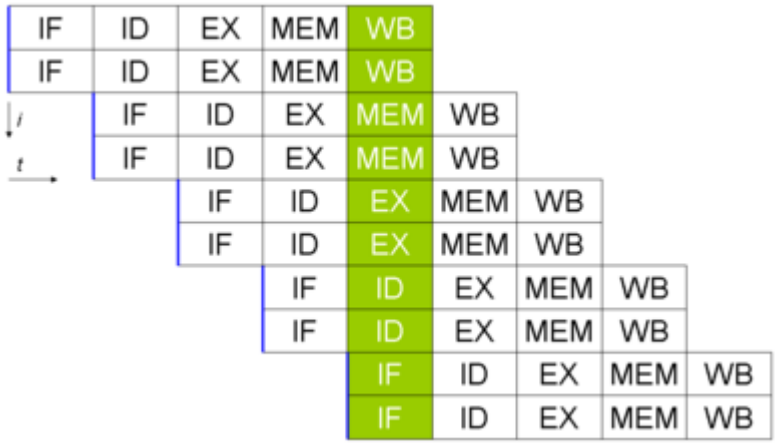
Si l'on insère des registres tampons (*pipeline registers*) entre chaque unité à l'intérieur du processeur, celui ci peut alors contenir plusieurs instructions, chacune à une étape différente.

Les 5 instructions s'exécuteront en 9 cycles, et le processeur sera capable de terminer une instruction par cycle à partir de la cinquième, bien que chacune d'entre elles nécessite 5 cycles pour s'exécuter complètement.

Séquençage des instructions dans un processeur doté d'un pipeline à 5 étages. Il faut 9 cycles pour exécuter 5 instructions. À $t = 5$, tous les étages du pipeline sont sollicités, et les 5 opérations ont lieu en même temps.

Au 5e cycle, tous les étages sont en cours d'exécution.

Une architecture superscalaire contient plusieurs pipelines en parallèle. Il est possible d'exécuter plusieurs instructions simultanément. Sur un processeur superscalaire de degré 2, deux instructions sont chargées depuis la mémoire simultanément. C'est le cas des processeurs récents conçus pour maximiser la puissance de calcul. Notons toutefois qu'en général, chaque pipeline est spécialisé dans le traitement d'un certain type d'instruction : aussi seules des instructions de types compatibles peuvent être exécutées simultanément.



Séquençage des instructions dans un processeur superscalaire de degré 2. Il faut 9 cycles pour exécuter 10 instructions. A *t* = 5, toutes les unités du processeurs sont sollicitées.

2.4. Pagination

La pagination est une technique, utilisée sur les ordinateurs, pour augmenter la taille de la mémoire virtuelle sans changer la mémoire vive. Cette technique divise la mémoire physique en *cadres de pages*, qui seront remplis par des pages chargées au moment de leur utilisation. Cette technique est complètement transparente au programmeur.

Elle consiste à découper la mémoire en "pages" de taille constante (Par exemple de 4ko). Ainsi le client dispose de blocs de mémoire constants qui évite la fragmentation externe de la mémoire.

Les pages mémoires sont principalement gérées par un mécanisme logiciel et peuvent faire intervenir un mécanisme matériel appelé Memory Management Unit (MMU) qui servira de "Mémoire cache" afin d'accélérer les requêtes ultérieures.

La fonction de la pagination permet d'obtenir un grand espace d'adressage linéaire sans avoir à acheter de la mémoire physique.

ANNEXE

Architecture externe du microprocesseur 8086 .