UNIVERSITÉ DE BOURGOGNE
Département Informatique

Architecture des systèmes

Module INFO-TC-ASR2

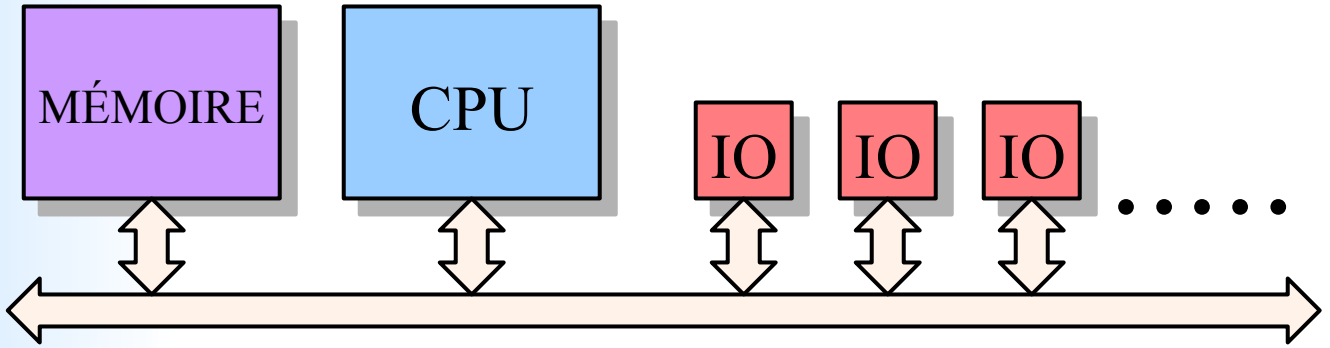
Fonctionnement des ordinateurs

Cours n°1

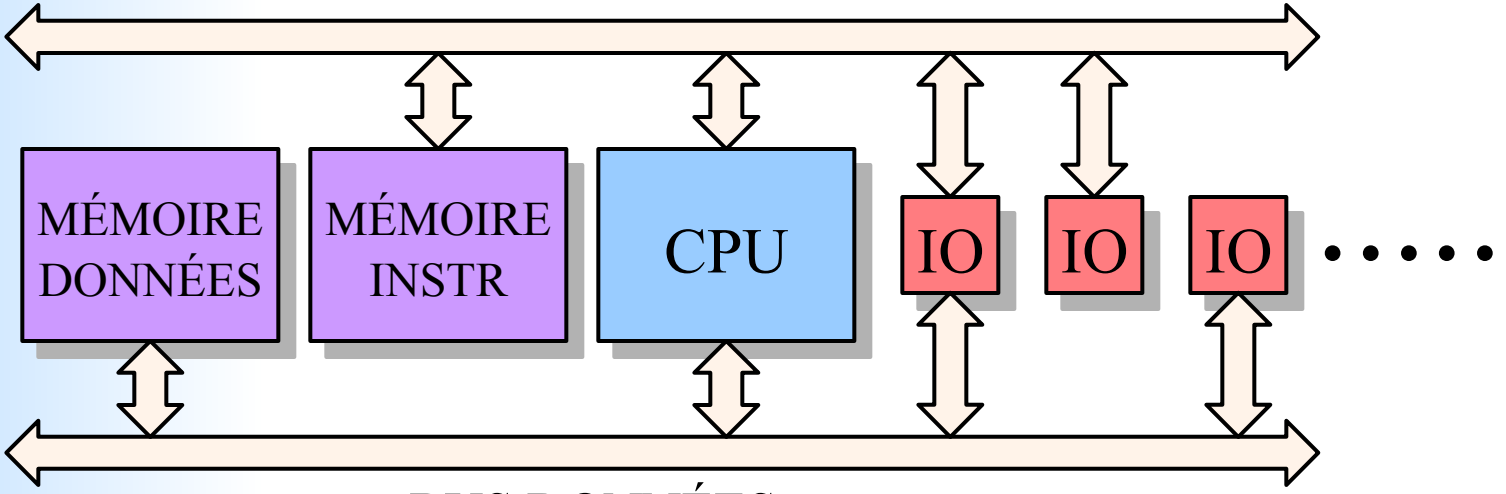
Description d'un microprocesseur : le 386

Introduction

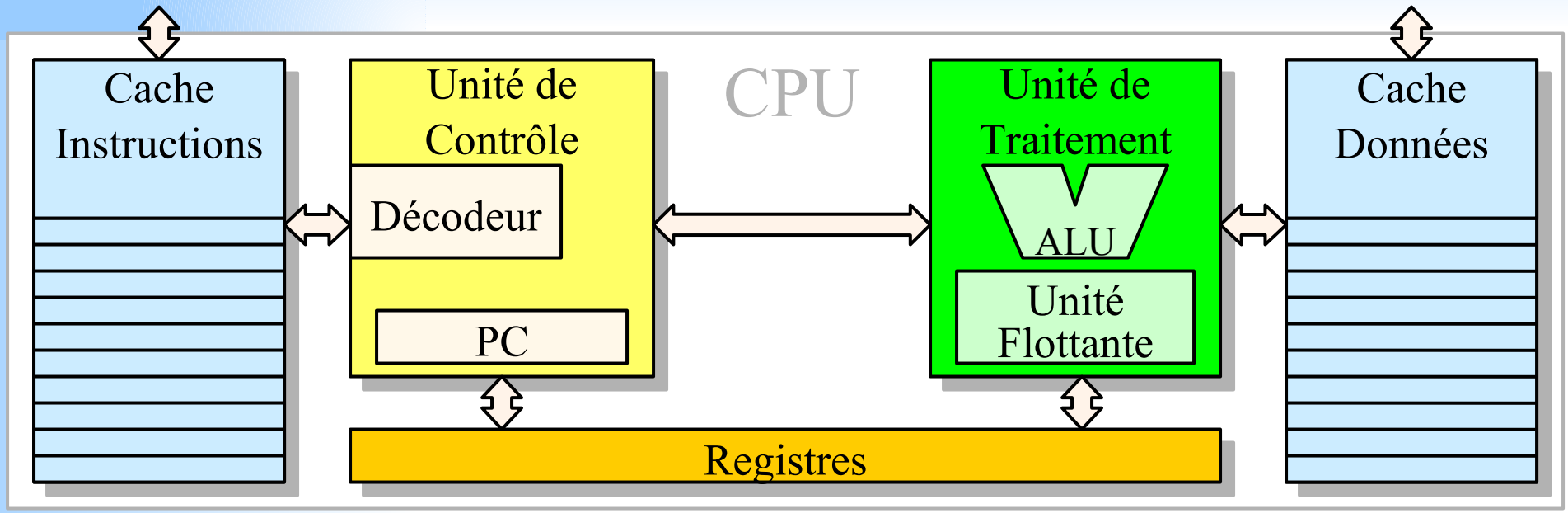
Un système informatique classique possède l'architecture suivante, dite **architecture de Von Neumann** :



On trouve également une architecture légérement différente :
l'architecture de Harvard



Le microprocesseur



En anglais CPU (*Central Processing Unit*), c'est le **coeur** de l'unité centrale

Il **exécute** les instructions en provenance de la **mémoire** à l'aide de ses différents composants.

Le micro i386

Historique

Ce microprocesseur est apparu en 1985. Il succède au 286, au 186, au 8086 et au 4004. Il hérite certaines de leurs caractéristiques et garde une certaine compatibilité avec eux.

Caractéristiques générales

Bus de données (et registres accumulateurs) de 32 bits.
Bus d'adresses externe de 32 bits (donc 4 GO adressables)
Bus d'adresse interne de 48 bits (donc 64 TO de mémoire virtuelle)
Une ALU entière de 32 bits (pas d'unité de flottants)
Pipeline de 6 unités
Jeu d'instruction étendu (processeur de type RISC)
3 modes de fonctionnement : **réel**, **protégé**, **virtuel**.
Prévu pour des OS multitâches

Les registres du 386

Registres généraux (accumulateurs) de 32 bits:
EAX, EBX, ECX, EDX

Registres de pointeurs et d'index (32 bits) :
ESI, EDI, EBP, ESP, EIP

Registres de segments (16 bits) :
CS, DS, DS, ES, FS, GS

Registres spéciaux
EFLAGS, CR0, CR1, CR2, CR3, GDTR, IDTR, LDTR, etc...

Le registre des indicateurs (EFLAGS)

C'est le seul registre dont le sens est **bit à bit** et non pas **mot à mot**. On pourrait en fait le voir comme 32 registres de 1 bit.

Les bits utilisés par le 386 sont les suivants (du LSB au MSB) :

CF (retenue)
PF (parité)
AF (retenue auxilaire)
ZF (zéro)
SF (signe)
TF (piège)
IF (interruption)
DF (direction)
OF (dépassement)
IOPL (niveau de priviligège)
NT (emboîtage des tâches)
RF (reprise)
VM (mode virtuel)

Instructions

Le 386 reprend toutes les instructions de ses prédécesseurs, et en rajoute un grand nombre.

Les plus utilisés sont les suivantes :

Copie : MOV, MOVS
Opérations arithmétiques : ADD, SUB, MUL, DIV, IMUL, IDIV
Opérations logiques : AND, OR, XOR, NOT
Branchements : JMP, JL, JLE, JNE, JE, JG, JGE, JC, JNC, etc...
Sous-programmes : CALL, RET, IRET
Gestion de la pile : PUSH, POP, PUSHA, POPA, PUSHF, POPF

Accès aux registres

Les registres généraux sont accessibles au programmeur, ainsi que les registres index / de segment dans certains cas.

On peut lire ou écrire un registre.

En cas de lecture d'un registre, son contenu peut être transferé dans la mémoire centrale ou dans un autre registre **compatible**.

Le contenu d'un registre peut être modifié à partir d'une constante (**valeur immédiate**) ou d'un autre registre, ou à partir de la mémoire.

Le résultat d'une opération arithmétique et/ou logique **doit** se faire dans un des registres généraux (par exemple EAX) : ce registre doit dont contenir l'une des opérandes.

Exemple : affectation du nombre 0 au registre EAX
MOV EAX, 0

Exemple : recopie du registre EAX dans le registre EBX
MOV EBX, EAX

Interdit : affectation d'un registre à une constante
MOV 34, EAX

Ajout de EAX et EBX dans EAX :
ADD EAX, EBX

Ajout de EAX et EBX dans ECX :
MOV ECX, EAX
ADD ECX, EBX

Accès à la mémoire

L'accès à la mémoire centrale (la RAM) peut se faire en lecture, en écriture, **mais pas les deux en même temps**.

Pour accéder à une cellule mémoire, il faut fournir son **adresse**.

Cette adresse est soit fournie par une valeur constante (**immédiate**), soit contenue dans un **registre d'index** (ESI, EDI, EBP...)

On peut utiliser un adressage **direct**, **indirect**, avec **décalage**.

Lire une valeur mémoire

Lorsqu'on lit une valeur en mémoire, on ne peut que la stocker dans un registre général (accumulateurs, par exemple EAX).

Exemple : lecture du contenu de l'adresse 0x1000F000

```
MOV EAX, [0x1000F000]
```

Exemple : lecture du contenu d'une adresse indirecte

```
MOV EAX, [EBP]
```

Seuls EBP, EBX, ESI, EDI peuvent être utilisés ici

Exemple : lecture du contenu d'adresse avec déplacement

```
MOV EAX, [EBP + 0x04]
MOV EBX, [EBP - 0x03]
```

Exemple : lecture d'une adresse mémoire avec indexage

MOV EAX, [EBP + ESI]

Exemple : lecture d'une adresse mémoire avec indexage et déplacement

MOV EAX, [EBP + ESI + 4]

Les combinaisons valides sont :
EBP ± ESI
EBP ± EDI
EBX ± ESI
EBX ± EDI

Ecriture d'une valeur en mémoire

On peut écrire en mémoire à partir d'une valeur **immédiate** ou à partir d'un registre, **mais pas à partir d'une autre cellule mémoire**.

Comme pour la lecture, l'accès peut être direct, indirect, indexé, etc...

Ecriture à l'adresse 0xF000A000 de la valeur 10 :

```
MOV [0xF000A000], 10
```

Ecriture à l'adresse contenue dans EBP de la valeur contenue dans EAX :

```
MOV [EBP], EAX
```

Recopie à l'adresse 0x0000F100 de la valeur lue à l'adresse 0xF120A012 :

```
MOV EAX, [0xF120A012]
MOV [0x0000F100], EAX
```

Prochain cours :

La pile
Les sous-programmes

A la semaine prochaine !