# Architecture des ordinateurs

Hélène Collavizza
http://www.polytech.unice.fr/~helen/ARCHI

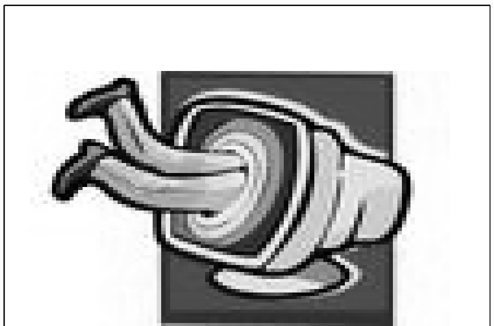
2006/2007 SI 1



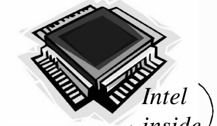


# Introduction

J'ai un problème !

0001011011110110
0001011101101101
....



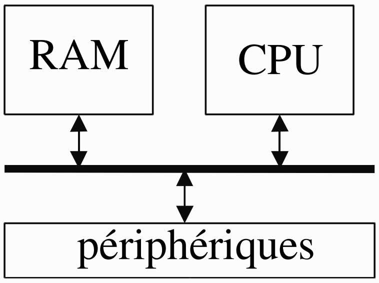
J'ai un problème !!! *Analyse, Conception, Algorithmique*

programme Java, C, C++, … *Programmation*

code binaire exécutable

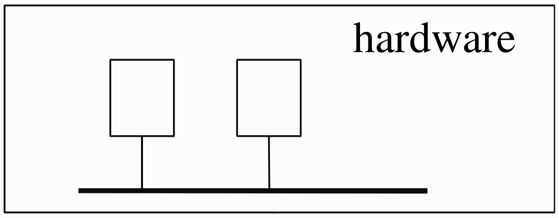
Assembleur

RAM = ?
CPU = ?




Architecture

TDO




Électronique numérique





# Vous avez dit **Architecture**

**IBM 360 (1964)**

**Structure d'un ordinateur que le "programmeur en langage machine" doit connaître pour écrire un programme correct (indépendamment du temps) pour cet ordinateur.**

**Glossaire Intel 2002**
www.polytech.unice.fr/~helen/glossaireIntel.html

**The architecture of a processor refers to the instruction set, registers, and memory-resident data structures that are public to a programmer and are maintained and enhanced from one generation of architecture to the next.**

**The micro-architecture of a processor refers to implementation of a processor architecture in silicon. Within a family of processors, the micro-architecture typically changes from one processor generation to the next, while implementing the same public processor architecture.**

# Programme de haut niveau

Informatique = traitement automatique de données

Exécuter un programme

- Acquérir des données
- Socker des données
- Faire des calculs sur les données
- Emettre des données





Où sont les programmes ?

- Programmes système

  Ex: ls en linux, clic-clic sur « Poste de travail » en windows, ...
- Programmes « utilitaires »

  Ex: mozilla, word, warcraft, excel, gcc, …
- Outils de développement qui permettent eux-mêmes d'exécuter des programmes (ex: VisualBasic, eclipse, interpréteur scheme, ...)
- Programmes utilisateur

En Java : c'est la JVM qui exécute le programme, le processeur lui exécute la JVM.

Javac JoliProgramme.java ← ***Compilation***
***Le processeur exécute /usr/java/jdk1.5.0/bin/javac***

java JoliProgramme ← ***Exécution de JoliProgramme par la JVM***
***Le processeur exécute /usr/java/jdk1.5.0/bin/java***

En C ou C++ :

**programme source « vacances.c »**

↓ ***compilation*** — ***code binaire avec adresses symboliques***

**code objet « vacances.o »**

↓ ***édition de lien*** — ***code binaire avec adresses réelles + assemblages de modules***

**code exécutable « vacances.exe »**

Exécution du programme vacances en invoquant « vacances.exe » sous l'interpréteur de commande





Lors de l'exécution d'un programme, son code binaire est chargé en mémoire centrale et est traité par l'unité centrale.

C'est la façon de traiter ce code que nous allons étudier.

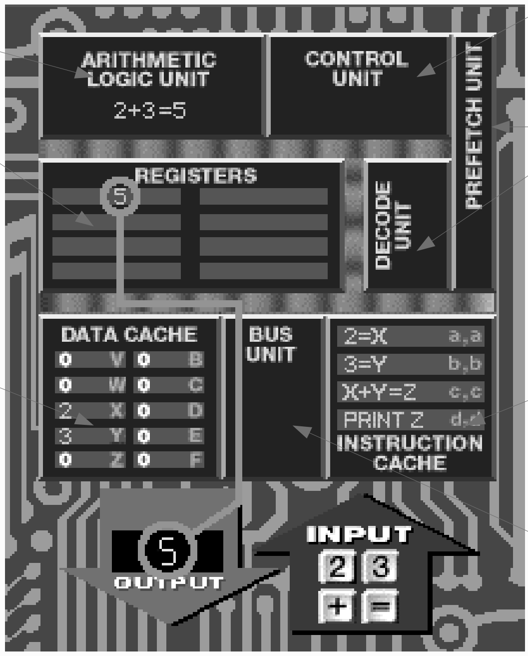
## Un exemple : Le tutorial d'Intel

calculs

ordonnancement des tâches

stockage interne

interprétation des instructions




cache des données (copie partielle de la mémoire centrale)

cache des instructions (copie partielle de la mémoire centrale)

gestion des transferts entre les unités

http://www.intel.com/education/mpworks/





# Objectifs de ce cours d'architecture

- Comprendre comment un programme de haut niveau s’exécute sur un assemblage de composants électroniques
  - Comment est codé un programme ?
  - Comment sont exécutées les instructions ?
  - Le parallélisme : exécution pipeline
  - Lien avec l'électronique numérique : que se passe-t-il sur chaque top d'horloge ?
  - Tour d'horizon d'architectures
- Prendre conscience du coût des accès mémoire : principe des caches mémoire

# Comment ?

- Etude des principes généraux : lien avec les familles de processeurs les plus courantes
- Application : PROCSI, le processeur de l'SI
  - Un exemple simple mais réaliste de CPU et de jeu d'instruction
  - On connait tous les détails et on pourrait le concevoir et le simuler avec Altera
  - Une version simple, puis une version pipeline

➜ Vous devrez comprendre en détail le fonctionnement de PROCSI et en déduire le fonctionnement d'architectures similaires

A compléter

page 8

Détails dans le document "PROCSI" page 8





# Plan

| | |
|---|---|
| 1. Chemin de données | *ce que contient le CPU* |
| 2. Instructions machine | *ce qu'exécute le CPU* |
| 3. Cycle d'exécution des instructions | *le pipeline* |
| 4. Partie contrôle | *ordonnancer les calculs* |
| 5. Hiérarchie mémoire | *la mémoire centrale s'accelère* |
| 6. Tour d'horizon d'architectures actuelles | *le CPU accelère* |

- QUIZZ ½h en TD la 1ère semaine (non noté)
- QUIZZ ½h en amphi en fin de parcours (pas de document)
- Contrôle final : exercices sur table (documents)

Cours d'assembleur ARM pendant 6 semaines

# Historique

- Lié à :
  - Evolution des composants électroniques (tubes à vide, transistors, LSI, VLSI)
  - Evolution du logiciel (écriture des compilateurs)
  - Evolution de l'architecture (principe du pipeline)
- Tendances :
  - vitesse
    - taille des composants ↘ donc distance↘ donc vitesse ↗
    - organisation avec pipeline, parallélisme, prédiction d'exécution -> le CPU est toujours occupé
  - taille puce ↘
  - taille mémoire ↗
- Difficulté : équilibre entre les différents compossants

  ex: technique du cache car la vitesse CPU augmente beaucoup plus vite que celle des accès mémoire





# Historique (évolution architecturale)

| | | |
|---|---|---|
| | **Calculettes** | |
| **< 1946** | **ENIAC 1ère machine programmable**<br>**(Univ. Pensylvannie, 30m, programmable à la main)** | |
| **1949** | **Machine programmable, modèle Von Neumann (programme stocké)**<br>**Machines à accumulateurs** | |
| **1956** | **Machines à registres généraux**<br>**(Pegassus, IBM 701)** | |
| **1963** | **Machines à pile**<br>**(B5000)** | |
| **1964** | **IBM 360**<br>**"Architecture des ordinateurs"**<br>**→ adressage par octet**<br>**→ inst. registre / mémoire** | **Control Data CDC 6600**<br><br>**→ inst. registre/registre**<br>**→ pipeline** |

| | | | |
|---|---|---|---|
| **1970** | **architectures orientées logiciel** | **micro-processeurs<br>Intel 4004, 8008** | |
| **1978** | **VAX DEC<br>architecture orthogonale<br>instruction mémoire / mémoire<br>1 architecture ↔ 1 système d'exploitation** | | |
| **1980** | **IBM PC 8086, 8088<br>architecture ouverte** | **Motorola Macintosh<br>68000** | **Projets architectures RISC<br>→ IBM 801<br>→ MIPS Stanford<br>→ RISC Berkeley** |
| **1986** | **Commercialisation architectures RISC<br>SPARC SUN, 88000, DECstation 3200** | | |
| **1993** | **Architectures superscalaires + traitement parallèle de flots de données (vidéo, son, …)<br>+ organisation mémoire<br>Intel Pentium, PowerPC, Ultra SPARC** | | |





**2000** **toujours plus de parallèlisme, exécution dans le désordre, prédiction des branchements,**
**DSP (Digital Signal Processor), multi-processeur**
**Pentium 4 : multi-threading**

**2015** **http://www.intel.com/technology/architecture/platform2015/**
**Processeurs dédiés (téléphonie mobile, reconnaissance et synthèse de la parole,**
**bio-médical, ...), parallélisme de haut niveau, des mémoires plus grandes**
**économies d'énergies !**

# Bibliographie

"*Computation Structures*"
**S. A. Ward, R. H. Halsted**
**MIT Press, McGraw-Hill Company**
**http://6004.lcs.mit.edu/**

"*Computer Organization & Architecture, Designing for Performance*"
**W. Stallings**
**Prentice-Hall International**

"*Organisation et architecture de l'ordinateur*"
**W. Stallings**
**Pearson Education France**

"*Computer Organization & Design, The hardware / software interface*"
**D. A. Patterson, J. L. Hennessy**
**Morgan Kaufmann Publishers**

"*Logic and Computer Design fundamentals*"
**M. M. Mano, C. R. Kime**
**Prentice-Hall International**