

Histoire et actualité de l'informatique

Évolution, problèmes, perspectives

Sacha Krakowiak

Université de Grenoble

Séance 2
Naissance du logiciel
Nouvelles générations d'ordinateurs

1954-1965

Où en est-on au début des années 50 ?

❖ Les avancées

Turing (et les autres) : définition et limites de l'algorithme, la machine universelle

von Neumann (et les autres): le modèle de base, toujours valable
les premiers ordinateurs commerciaux

❖ Les limites

une «technologie» déficiente, les tubes (peu fiables) ; les mémoires à lignes à retard (complexes, réglage délicat)

une programmation à très bas niveau, sans outils conceptuels, fastidieuse et sujette aux erreurs

un champ restreint d'applications

l'informatique est encore (pour longtemps) une affaire de professionnels

1954-1965 : une décennie féconde

❖ La naissance du logiciel

Les langages de programmation

Les systèmes d'exploitation

❖ Les nouvelles générations d'ordinateurs

Des tubes aux transistors, puis aux circuits intégrés

Du tambour magnétique aux tores de ferrite

❖ L'extension des domaines d'application

Calcul scientifique

Commande de procédés industriels

Gestion des entreprises

Recherche opérationnelle

❖ Un modèle d'organisation

Le centre de calcul

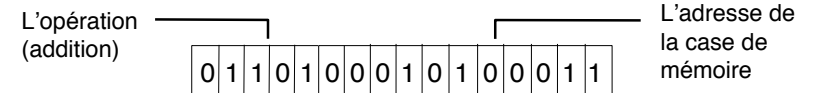
Du binaire à l'assembleur

❖ Qu'est-ce qu'une instruction pour un processeur ?

La description d'une action à exécuter

Représentée par une suite de bits dans la mémoire selon certaines conventions

Exemple de l'addition : ajouter le contenu d'une case de mémoire à celui de l'accumulateur



❖ Une notation plus commode ...

Donner des noms aux opérations et aux cases de mémoire

répertoire de la machine → **add toto** ← nom choisi par le programmeur

L'assembleur est le programme qui traduit cette notation en binaire

Qu'est ce qu'un langage de programmation ?

❖ Motivation

Un programme en assembleur décrit un algorithme en termes de *ce que sait faire la machine*

On souhaite une expression en termes de *ce que veut l'utilisateur*

❖ Qu'attend-on d'un langage de programmation ?

«Haut niveau» d'expression

proche du domaine d'application
indépendant de la machine d'exécution

Puissance d'expression

capable de décrire tous les traitements envisagés

Facilité d'apprentissage et commodité d'utilisation

Sûreté

réduit les risques d'erreur

```
si Z > 0
  X[3] = N + 2
```

```
trier(Tableau)
```

Le premier langage de programmation

❖ Fortran (*Formula Translation*)

Développé chez IBM par l'équipe de John Backus (1954-57)

Langage orienté vers le calcul scientifique

Évaluation de formules

Définition de fonctions réutilisables

Toujours utilisé aujourd'hui (mais a beaucoup évolué)

❖ Une révolution dans la pratique de la programmation

Ouvre la programmation aux utilisateurs ...

... sans sacrifier l'efficacité

❖ Les limites du premier Fortran

Manque de rigueur dans la définition

Manque de sûreté : mauvaise détection des erreurs

Exécuter un programme ...

Problème : combler l'écart entre le programme en langage de haut niveau et le code binaire exécutable de la machine

```
...
DO i=1,10
  DO j=1,50
    A(i,j)=(B(i,j))+C(j,i)
  END DO
END DO
...
```

```
...
DO i=1,10
  DO j=1,50
    A(i,j)=(B(i,j))+C(j,i)
  END DO
END DO
...
```

On "descend" le programme au niveau de la machine

Machine virtuelle
On "remonte" la machine au niveau du programme

Compilateur
(traduit le programme)

Machine virtuelle
(exécute le programme)

Interprète
(exécute les instructions du programme)

```
0010110001110101001011011010
1000011101011010011101010001
0100000110110011101011011010
11011001111101011 ...
```

Machine

exécute du code binaire

Les langages influents des années 60

❖ Fortran (1954) - John Backus

Le premier langage de haut niveau, toujours utilisé

❖ Algol 60 (1958-61) - comité de scientifiques (Peter Naur, rapporteur)

Définition d'un langage sur une base rigoureuse

objectif atteint, malgré définition par un comité ...

Peu utilisé mais très forte influence sur les langages futurs

❖ Cobol (1959) - comité d'industriels

Langage pour applications de «gestion»

Toujours utilisé, malgré son manque d'élégance

❖ Lisp (1958) - John Mc Carthy, MIT

Premier langage «fonctionnel»

Structure de base, pour programmes et données : la *liste*

Le langage favori pour l'intelligence artificielle

❖ Basic (1963) - John Kemeny

Langage simple (dérivé de Fortran, interactif)

Adapté au temps partagé

Naissance des systèmes d'exploitation

❖ Le problème

- Gérer l'accès des utilisateurs à la machine
- Assurer la bonne exécution des programmes

❖ Les origines

- Utilisation individuelle sur réservation
- Usage de «sous-programmes» pour les entrées-sorties
- Traitement par lots
 - La file d'attente physique devient virtuelle
 - Le «moniteur» gère les ressources et les programmes
- Les premiers systèmes d'exploitation viennent des clients !

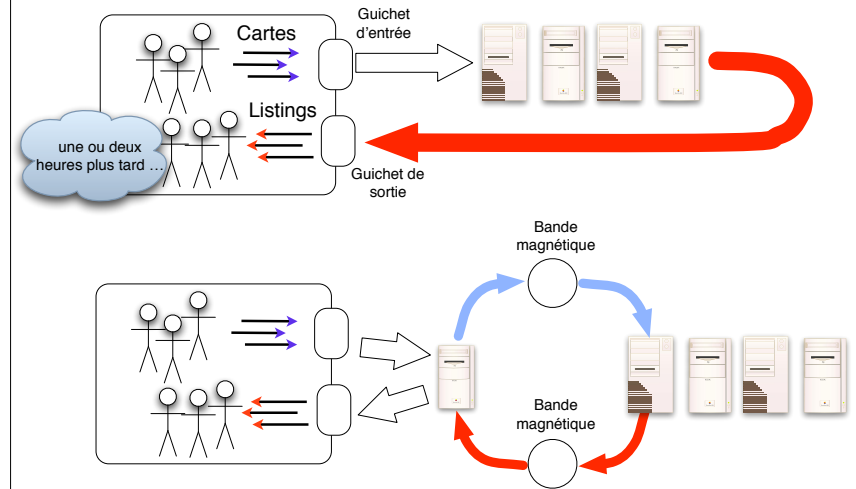
❖ De nouveaux métiers pour l'exploitation

- Opérateur, pupitreux, perfo-vérif.

Deux exigences qui s'opposent ...

- l'utilisation optimale du matériel
- le confort (et l'efficacité) des utilisateurs

L'exploitation par lots (*batch processing*)



Partage des ressources de l'ordinateur

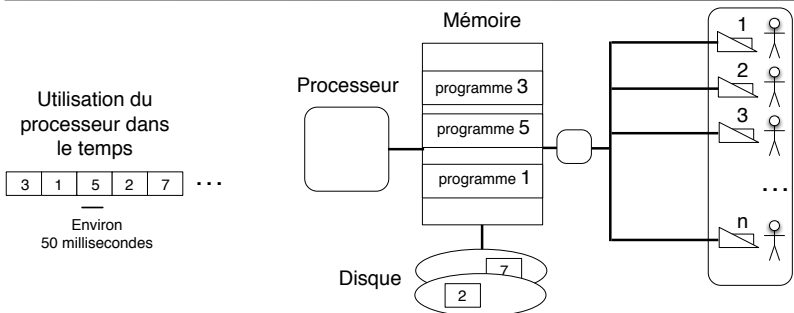
❖ Partager la mémoire

- Multiprogrammation : plusieurs programmes partagent la mémoire
- Intérêt : utiliser les temps morts des différents programmes
 - Applications : OS 360 MFT, MVT
- Extensions : la pagination et la mémoire virtuelle
 - Expériences : Atlas (Manchester, 1962), IBM 360/67 (1966)

❖ Partager le processeur

- Restaurer le confort des usagers : le mode conversationnel
- Exploiter la différence des temps de réaction
 - Usagers : quelques secondes
 - Processeur : quelques millisecondes (à l'époque)
- Premières expériences
 - CTSS (Fernando Corbató, MIT, 1961) : montre que c'est faisable

Un système en temps partagé



Grâce au partage du processeur :

Chaque utilisateur a l'impression qu'il dispose seul de l'ordinateur

Grâce à l'utilisation du disque :

Chaque utilisateur a l'impression qu'il dispose d'une mémoire pratiquement illimitée (mémoire virtuelle)

Multics, un jalon dans l'histoire des systèmes d'exploitation

❖ Un projet ambitieux

General Electric (1964-69), Bell Labs (1964-70), MIT
Un service (centralisé) pour des centaines d'utilisateurs
Un accent fort sur la sécurité

❖ Des innovations techniques

Conception conjointe machine-système
Mémoire virtuelle, système de fichiers, protection, ...
Écrit en langage de haut niveau (PL/1)

❖ Un bilan contrasté

Carrière commerciale tardive et limitée ...
... mais influence forte et durable
Élaboration des concepts de base
Création d'Unix par des transfuges des Bell Labs

Un circuit du GE-645

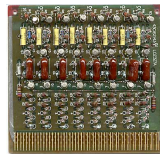


Photo Bill Eaton

Nouvelles générations d'ordinateurs (1)

❖ Des tubes aux transistors et aux circuits intégrés

Inventions : 1947, transistor ; fin années 50 : circuits intégrés

À partir de 1955

circuits : les transistors remplacent les tubes

mémoires : les tores de ferrite remplacent les tambours magnétiques

À partir des années 60 : introduction des circuits intégrés

Avantages

Fiabilité, vitesse, consommation d'énergie, encombrement

❖ La microprogrammation

Inventée par Wilkes en 1951, utilisée dans les années 60

Idee : étendre la notion de programme à la conception interne des processeurs

Chaque instruction est une séquence de micro-instructions

Avantages : facilité de conception, adaptabilité, réutilisation

Nouvelles générations d'ordinateurs (2)

❖ La première «famille» : la série IBM 360 (1964)

Une ligne d'ordinateurs compatibles
grâce à la microprogrammation

Des avancées techniques

Le cache (IBM 360/85, 1968)

Les machines virtuelles (IBM 360/67, 1966)

Un système d'exploitation gros et complexe (OS/360)

Traitement par lots, premier système utilisant les disques

Un succès commercial

❖ Le premier «super-calculateur» : CDC 6600

Seymour Cray (1964)

Processeurs spécialisés, traitement parallèle

❖ Le premier mini-ordinateur : DEC PDP-8 (1965)



IBM 360/40

Acont



CDC 6600



DEC PDP 8

Les premières applications (1)

❖ Le calcul scientifique

L'objectif : trouver des solutions numériques à des problèmes de nature mathématique

Les bases : calcul numérique, puis analyse numérique

Une discipline en très fort développement

Les domaines d'application

Recherche : physique, chimie, mécanique, astrophysique, ...

Médecine et biologie peu impliquées au début

Sciences de l'ingénieur : électronique, électrotechnique, génie civil, nucléaire, spatial ...

Premier outil graphique : Sketchpad (Ivan Sutherland, 1961)

❖ La commande de procédés industriels

L'objectif : piloter un processus évolutif pour atteindre un but fixé

Les bases : l'automatique

Les applications : processus de fabrication, transports

Les premières applications (2)

❖ Les applications de gestion

Objectif : automatiser les opérations de comptabilité, gestion des stocks, gestion de personnel, ...

Bases :

analyse fonctionnelle (le «quoi ?» : besoins et contraintes)

analyse organique (le «comment ?» : modèle de réalisation)

Limites

méthodes encore peu évoluées, difficulté sous-estimée

pas de notion globale de «système d'information»

❖ La recherche opérationnelle

Objectif : aide à la décision, notamment en environnement incertain

Bases : mathématiques «discrètes» (combinatoire, ...)

Applications : initialement militaires, puis approvisionnement, implantation, logistique, ...

❖ Une tentative prématurée : la traduction automatique

Où en est-on vers la fin des années 60 ?

❖ Les avancées

Les langages de programmation de haut niveau

Les systèmes d'exploitation

Les nouvelles générations d'ordinateurs

Le développement d'une industrie informatique

Constructeurs de machines

Sociétés de service

❖ Les limites

Le défi de la production de logiciel

Programmes incorrects, délais non respectés, budgets dépassés ...

Des domaines d'application encore restreints

Encore peu d'impact sociétal

Une organisation centralisée