

# Architecture et Système

Stefan Schwoon

Cours L3, 2020/21, ENS Cachan

# Organisation

---

Cours: Lundi 13:45–15:45, 1E18 (Stefan Schwoon)

TP: Vendredi 13–16, 1S53 (Farzad Jafarrahmani et Gabriel Hondet)

Contrôles de connaissances : 2 Projets + Examen (un tiers chacun)

Seconde session : rattrapage d'examen

Transparents etc : <http://www.lsv.fr/~schwoon/>

E-mail: [schwoon@lsv.fr](mailto:schwoon@lsv.fr)

Bureau: 2S54 (LSV)

# Contenu du cours

---

## Contenu

Comment fonctionne un ordinateur ? Que se passe-t-il à l'intérieur ?

Aspects matériel (architecture) et logiciel (système)

**Exemples (matériel)** : Circuits, assembler, représentation de données, mémoire, interruptions, ...

**Exemples (logiciel)** : shell, programmation système, système fichiers, réseau, ...

# Contenu du cours

---

## Objectifs:

Connaissances pratiques pour vos tâches quotidiennes, en expérimentation, programmation, ...

Systeme d'exploitation : Unix (standard POSIX)

Langages utilisés : assembleur (un petit peu) et surtout C

# Littérature

---

## Architecture :

John P. Hayes, *Computer Architecture and Organization*, McGraw Hill (3rd edition)

## Système d'exploitation :

Andrew S. Tanenbaum, *Operating systems*, Prentice Hall

# Contenu d'aujourd'hui

---

Historique (très) abrégée des ordinateurs

Développement de l'architecture

Début sur les circuits logiques

# Les premières machines de calcul mécaniques

---

Au 17e siècle : Wilhelm Schickard / Blaise Pascal / Gottfried Leibniz

Machines mécaniques, basées sur le système décimal

Pascaline : addition, subtraction (par complément)

Machine de Leibniz : multiplication en plus (grandes problèmes mécaniques)

motivation : astronomique (S.), calcul financier, impôts (P.), philosophique (L.)

production en masse : à partir du 19e siècle (Thomas de Colmar)

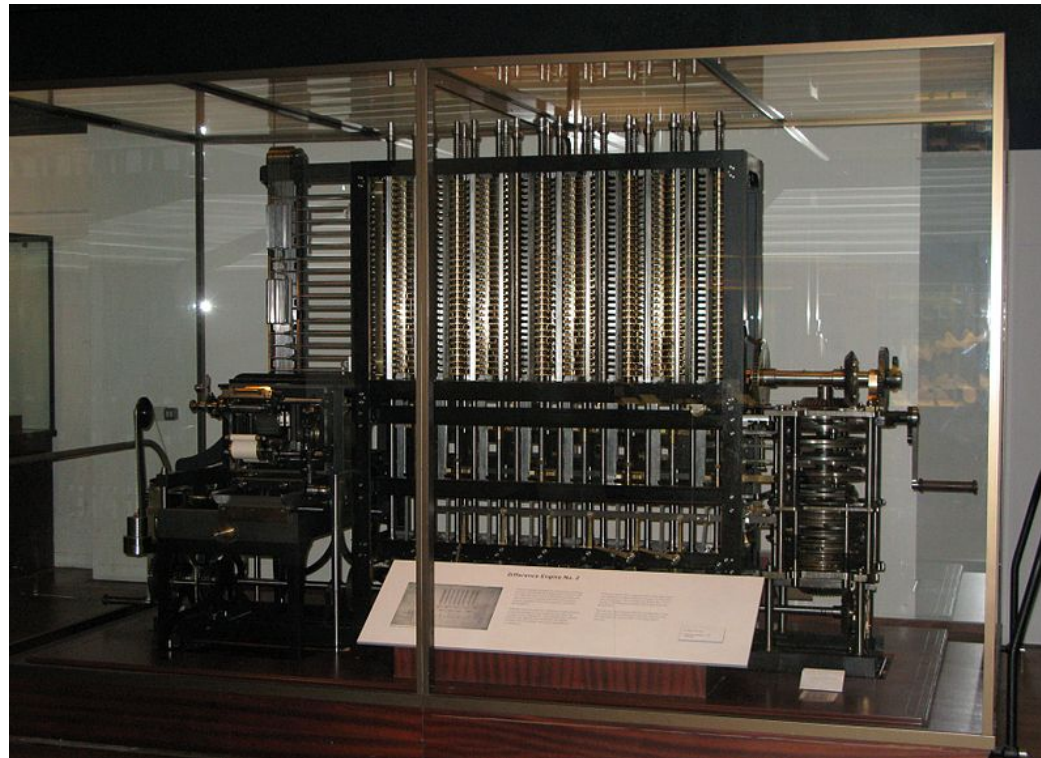
# Première machine dite programmable

---

**Difference Engine** (1822-1832) de Charles Babbage

Une opération (l'addition) appliquée simultanément à plusieurs **registres**

“Programmation” consiste en déterminant les valeurs initiales





# Fonctionnement de la Difference Engine

---

Fonctionnement (permet de calculer des polynômes) :

pour  $n$  registres, fixer les valeurs initiales  $x_1^{(0)}, \dots, x_n^{(0)}$ ;

appliquer la règle  $x_{i+1}^{(k+1)} := x_i^{(k)} + x_{i+1}^{(k)}$ , pour  $1 \leq i < n$  et  $k \geq 0$

La machine réellement construit avait 3 registres à 6 chiffres.

Motivation : calcul automatique des tableaux mathématiques

Problèmes mécaniques empêchent une construction efficace

Innovations : mémoire, itération

# Machine théorique conçue par Babbage

---

**Analytical Engine** (travail théorique, jamais mis en production)

Opérations multiples (addition/multiplication/. . .)

**Séquence** d'opérations programmable avec branchement conditionnel

# Fin 19e/début 20e siècle

---

Améliorations mécaniques et en utilisabilité

Des machines construites pour un but précis :

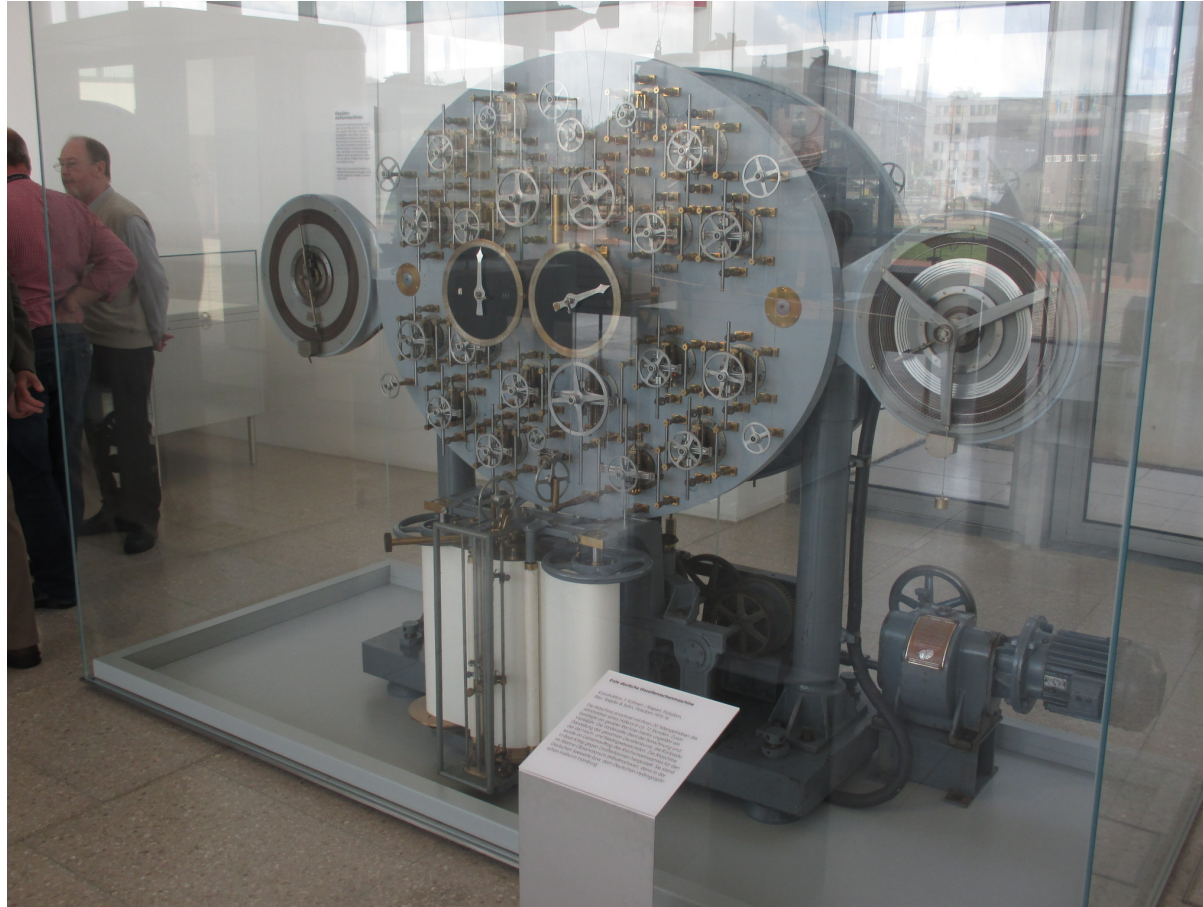
*calculatrices* utilisés par les ingénieurs, navigateurs, militaires, ...

automation dans la production des tableaux résultants

Encore des machines mécaniques basées sur le système décimal.

# Exemple d'une machine analogue

Calculatrice de marées de 1915



Calcule les marées d'un port donnée (dans la mer du nord) pour un an dans 12h

# La “machine” de Turing

---

Alan Turing (1936)

Conception théorique d'une machine universelle :

ruban infini pour stocker des données + un état interne

la machine lit un symbole à la fois, se déplace par une position et modifie son état selon des règles fixes

*Machine de Turing universelle*: peut simuler d'autres machines de Turing

⇒ un programme est une donnée

Modèle reconnu comme étant équivalent en pouvoir à un ordinateur quelconque;  
terminaison indécidable

# Les années 1930

---

Les machines de Konrad Zuse : Z1 (1938) and Z3 (1941)

fonctionnement électro-mécanique

calcul **binaire**, virgule flottante

programmation avec des **boucles** (mais pas de branchement conditionnel)



# Pendant la guerre

---

**Colossus** (britannique, 1943, pour briser des codes)

existence connue publiquement depuis les années 1970

construit pour un objectif précis, pas universelle, pas vraiment programmable

calcul binaire

**Mark I** (américaine, 1944, calcul ballistique)

calcul décimal

programmable par bande perforée

Turing-complet

# Les premiers ordinateurs électriques

---

## ENIAC (1946)

construit à l'Université de la Pennsylvanie

Poids : 30 tonnes; 18.000 tubes électroniques

encore décimal (20 registres à 10 chiffres)

programmable en (dé)branchant des câbles

temps requis pour une multiplication: 3 ms

utilisé pour des calculs ballistiques



# Les ordinateurs construits par von Neumann

---

## EDVAC (1951)

Calcul binaire

programme stocké en mémoire

instructions de la forme  $(a_1, a_2, a_3, a_4, op)$ :

appliquer  $op$  sur les données aux adresses  $a_1$  et  $a_2$ , stocker le résultat à  $a_3$ ,  
prochaine instruction à  $a_4$ .

branchement conditionnel: comparer les données à  $a_1, a_2$ , continuer soit à  
 $a_3$  ou à  $a_4$ .

---

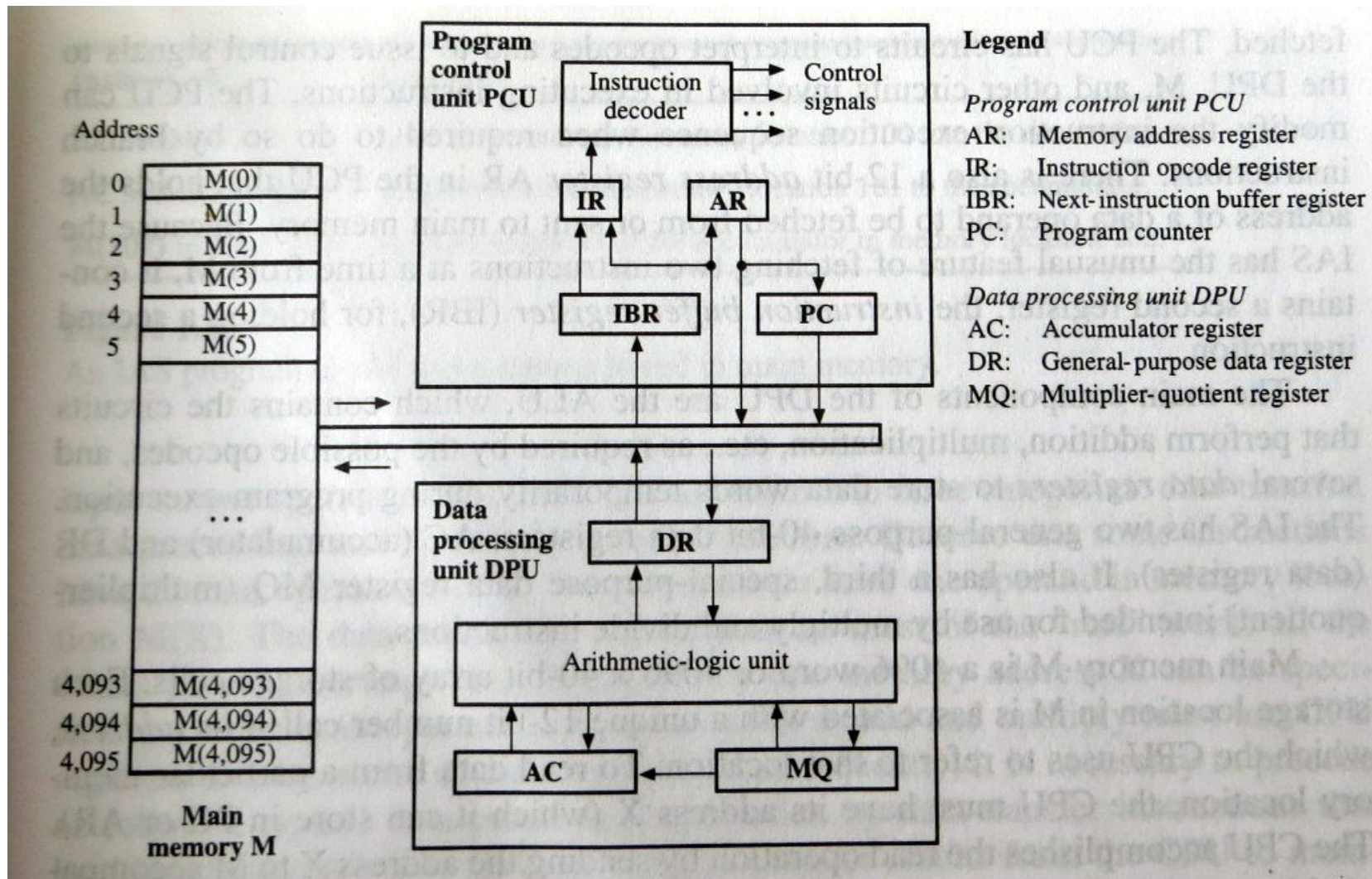
La **machine de l'IAS** (construit à Princeton)

mémoire:  $4096 = 2^{12}$  cellule ('mots') de 40 bit

notion d'un *processeur central* et *unité arithmétique-logique* (CPU et ALU)

considérée comme le prototype de l'architecture moderne

# Architecture de l'IAS



# Instructions de l'IAS

---

Un mot dans la mémoire peut être interprété soit comme une instruction, soit comme une valeur numérique.

Interpretation numérique : entier ou virgule fixe entre -1/+1

Instruction : un mot = 2 instructions de 20 bits

format  $(op, a)$  pour manipuler mémoire et registres

– op: code d'opération (8 bit) / a = adresse de 12 bit

transfert de données entre mémoire et registres/entre registres

addition/multiplication

opérations de contrôle (branchement conditionnel, code automodifiant)

# Ordinateurs de “deuxième génération” (50s-60s)

---

Couche physique :

remplacement de tubes par des transistors

plus petit, moins cher, plus rapide, plus fiable

Architecture:

instructions plus puissantes : adressage indirect, registres d'index

registres et opérations pour les virgules flottantes

réursion (opération sur pile)

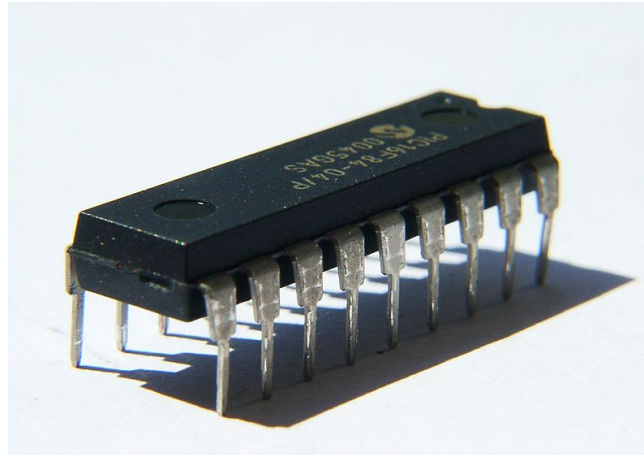
# Ordinateurs de “troisième génération” (1960s-70s)

---

Arrivée des **circuits intégrés** (IC)

beaucoup de transistors sur un espace énormément réduit

→ opérations plus rapides



Mémoires à vitesses différentes (cache); intégrée soit sur IC, soit ailleurs

# Circuits logiques

---

Composants pour construire un ordinateur, physiquement réalisés par des transistors, se comportant comme des portes logiques.

Contenu :

construction de fonctions logiques et arithmétiques

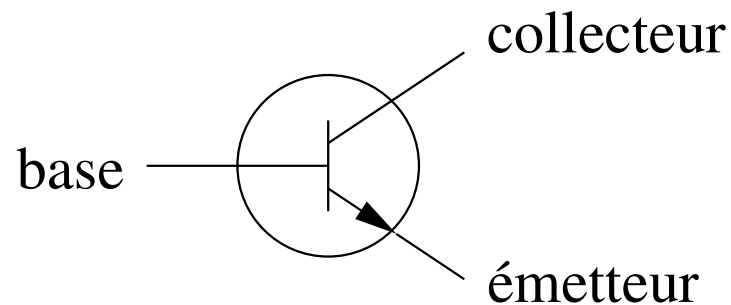
Aspects dynamiques : stockage, mémoire, transfert, microprogrammation

# Transistor

---

Découverte théorique 1925, utilisation dans les ordinateurs à partir des mi-1950s.

Il existe une grande variété de transistors utilisables pour réaliser des circuits logiques. Exemple : transistor bipolaire NPN :



Flux d'électrons possible entre émetteur et collecteur lorsque le voltage de base excède un certain seuil.



# Transistor réalisant une manipulation logique

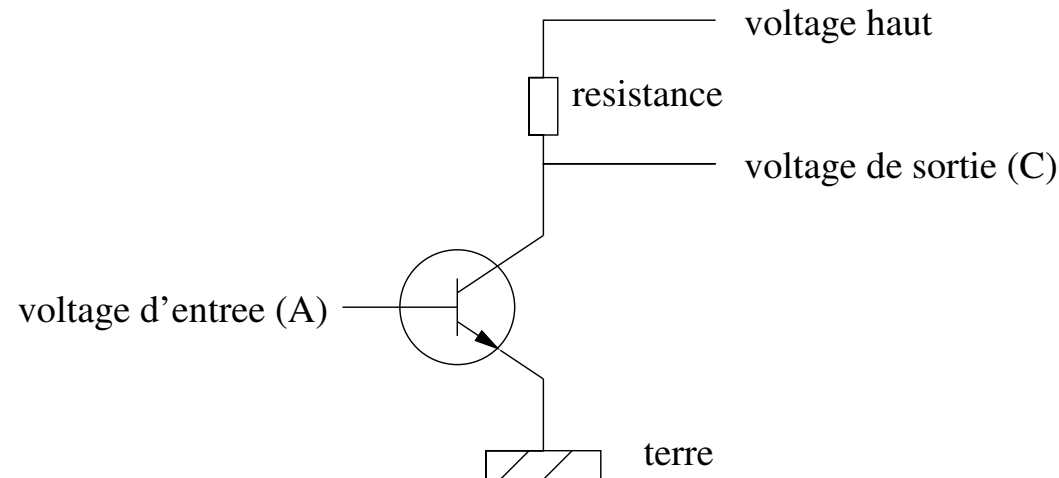
---

Le comportement du transistor permet de réaliser une logique binaire :

soit le voltage est en-dessous du seuil, alors le flux  $E \rightarrow C$  est interrompu;

soit le voltage est en-dessus du seuil, alors il y a un flux.

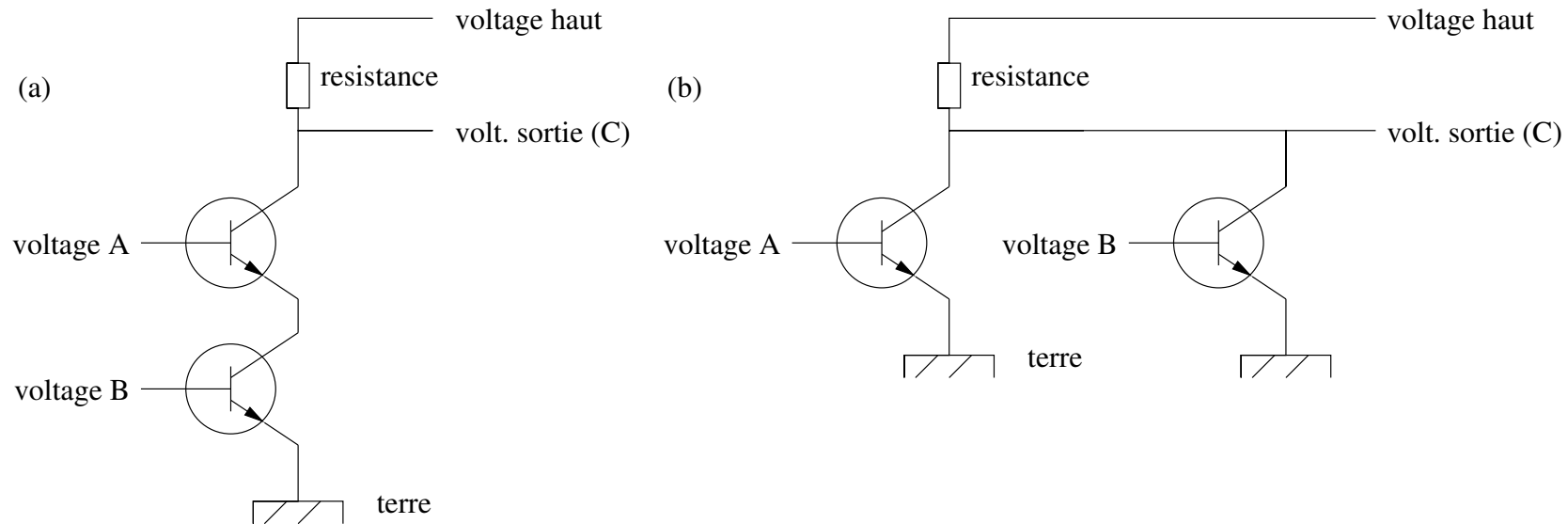
Utilisation d'un transistor pour la négation logique :



# Opérations binaires

---

Portes NON-ET (a) et NON-OU (b) réaliser avec des transistors:



Interprétation : voltage haut  $\hat{=}$  1, voltage bas  $\hat{=}$  0

# Importance de NON-ET et NON-OU

---

Les opérateurs NON-ET et NON-OU (NAND/NOR en anglais) sont importants pour deux raisons :

Ils sont facilement réalisable avec des transistors.

Toute autre fonction logique peut être exprimé avec soit NON-ET, soit NON-OU :

$$\neg A \equiv A \bar{A};$$

$$A \wedge B \equiv (A \bar{B}) \bar{(A \bar{B})};$$

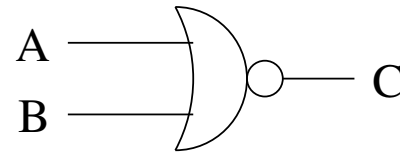
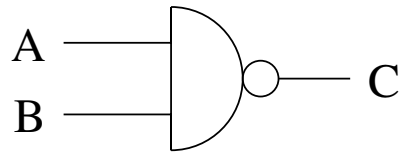
$$A \vee B \equiv (A \bar{A}) \bar{(B \bar{B})}.$$

Désormais, pour faire abstraction des détails physiques, on représentera les circuits en forme de diagrammes avec des **portes logiques** qui traitent des bits avec valeurs 0 et 1.

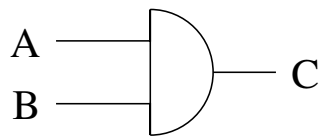
# Portes logiques

---

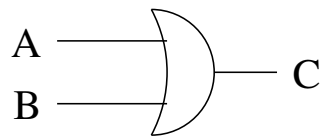
Diagrammes pour NON-ET (à gauche) et NON-OU (à droite):



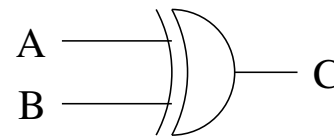
Diagrammes dérivés, réalise p.ex. par la combinaison de plusieurs portes NON-ET/OU.



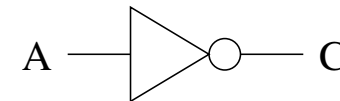
ET



OU



XOR



NON