ALM : Architectures logicielles et Matérielles

- Pascal.Sicard@imag.fr
- Bureau 313 Bâtiment D ENSIMAG

- Organisation de l'enseignement d'ALM
  - 10 * 1h30 heures de cours + 8* 3 Travaux dirigés de 2h
  - 2 séances de SOUTIEN sur machine en fin de semestre
  - Préparation de travaux pratiques en TD
  - Travail personnel sur machine en libre service
  - Comptes rendus (note de contrôle continu) par binôme
  - Note finale ALM1 : Examen coefficient 2 , Contrôle Continu 1

Organisation de l'enseignement

- **Partie hard (matériel):** éléments matériel nécessaire à la conception de processeurs et d'ordinateurs
- **Partie soft (logiciel):** Langage d'Assemblage, introduction à la traduction de programmes en langage de haut niveau permettant leurs exécutions par les circuits
- 1 TD "hard", 1 TD "soft", 1 TD "général" («hard bis»)
- **Attention à l'emploi du temps qui n'est pas "régulier" (supports de TDs)**
- **Contrôle continu:**
  - 3 comptes rendu de TPs Soft
  - 3 comptes rendu TPs Hard
  - Un compte rendu à rendre par binôme sur papier à l'enseignant concerné

Bibliographie et communication

- **Bibliographie**
  - P. Amblard, JC Fernandez, F. Lagnier, F. Maraninchi, P. Sicard, P. Waille Architectures logicielles et matérielles. Editions DUNOD 2000
  - En bibliothèque ou en ligne sur le WEB (dans le Moodle)
- **Outil pédagogique Moodle à l'UFR IMAG:**
  - http://imag-moodle.e.ujf-grenoble.fr

Objectifs

- Comprendre le fonctionnement des circuits composants les ordinateurs
- Comprendre comment un programme est exécuté par un ordinateur
- Notions indispensables à tout informaticien polyvalent
  - En particulier quand on «touche»
    - au système d'exploitation
    - aux périphériques et entrée/sortie
    - programmation «haute performance» (contrainte mémoire, temps réel ...)
  - Bases de la conception de circuit....
- 2ème semestre : Interface Logiciel/matériel et système (ALM2)
  - Processeur et son entourage, les entrées/sorties, les bases du système d'exploitation

Chapitre 1 : c'est quoi un ordinateur

Introduction

- Ethymologie : adjectif existant depuis longtemps :"Dieu qui met de l'ordre dans le monde"
- Computer : calculateur trop restrictif pour les premiers fabricants français

- Un ordinateur: une machine qui traite des informations
- Traitement : enchaînement "programmé" de calculs sur une représentation particulière de l'information
- On parle de traitement "numérique"
- Changements possibles des programmes pour effectuer des traitements à volonté

Ordinateurs



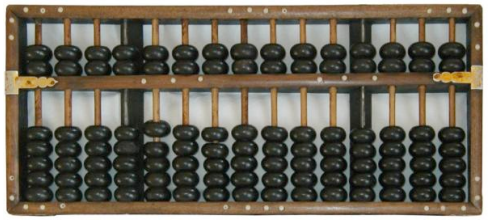


Brève historique

http://www.histoire-informatique.org

- La préhistoire

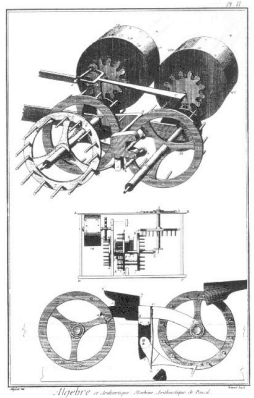
-3000 an AvJC : **Boulier**
- Appareil de calcul qui permet d'avoir la représentation du nombre (mémoire) et d'effectuer une opération sur ce nombre (calcul)



Première machine à calculer en bois et métal

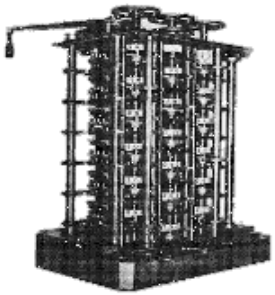
- 17e siècle premier calcul automatique :
  - Machine de Pascal : roues crantées pour effectuer des additions et des soustractions.


Une machine de Pascal conservée



Historique 19 ème

- 1820: **Machine à différences** de C. BABBAGE
- jamais finalisé, 25000 pièces, système décimal



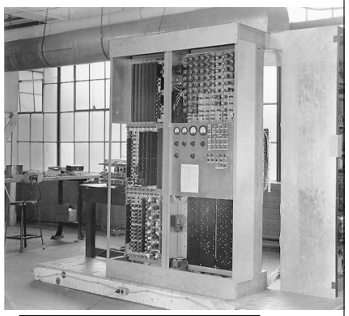
- 1833: **Machine analytique** de C. BABBAGE
  - **utilisé à l'aide de cartes *opérations* , de cartes *de variables* et de cartes *nombres*.**
  - **Ancêtre de l'ordinateur**

Historique début 20 ème siècle
Du mécanique à l'électrique

- Pas encore vraiment des ordinateurs: spécialisés dans des calculs particuliers (multiplication ...): 1 seul programme
- **1930 : *Enigma* : Premier Calculateur : 1 seul algorithme**
  - **Cryptage de message pendant la 2ème guerre mondiale**
  - **A. Turing participe à la construction de la "bombe" permettant de décrypter**
- **1938: *Z1***, premier calculateur à relais électriques par K. ZUSE.
- **1939: *ABC*** (J. ATANASOFF)**: premier calculateur binaire (l'information est représentée par des 0 et des 1)**
- **1943: ASCC** ou **Harvard MARK 1** par H. AIKEN : en collaboration avec IBM , à base de milliers de relais, 5 tonnes, refroidissement par glace

Premiers ordinateurs
"On peut changer les programmes"

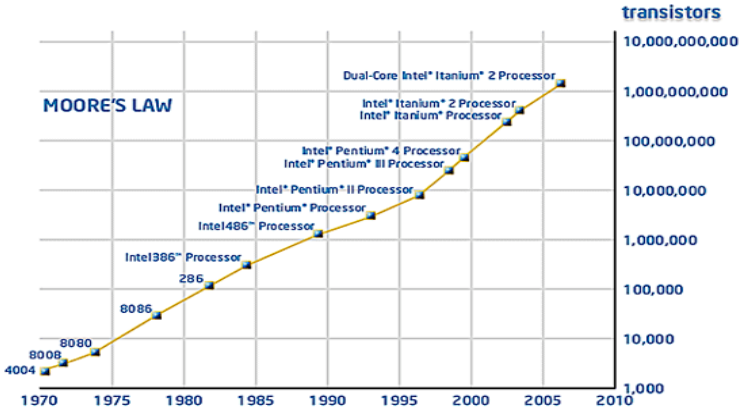
- 1948: Machine de Von Neuman (EDVAC)
- Architecture identique dans les ordinateurs d'aujourd'hui
- Mémoire pouvant contenir un programme et ses données
- Une unité centrale permettant de réaliser des calculs
  - 2096 mots de 40 bits
  - 1 seul registre accumulateur
  - Instructions sur 20 bits



Jusqu'aux premiers ordinateurs personnels

- **1955** : Premiers ordinateurs à transistors (1948 : invention du transistor)
- **1970** : Premiers ordinateurs personnels (1000 transistors sur un circuit)
- **1981** : Premiers PC
- **1982** : A la faculté de Grenoble : Un ordinateur (Multics) mémorisation sur bande, programmation en assembleur sur cartes perforées (1 carte = une instruction). Mais aussi en langage de haut niveau.
- **1990** : RAM en kilo octets, Disque dur en Méga octets, fréquence horloge en Méga hertz
- **2000** : fois 1000 en 10 ans (Loi de Moore double tous les 18 mois)

Evolution des processeurs
Loi de Moore (source Intel)



Taille du transistor (micro: 10^{-6} mètre, nano 10^{-9} mètre)

La représentation de l'information

- Une information: nombre, caractère, texte, image, son, couleur...
- Dans une machine ces informations sont codées
  - Différents codes possibles:
    - un entier: des lignes de boules déplacées, des roues crantées positionnées, du courant électrique...
    - Du son: une gravure sur du vinyle, du courant électrique, vibration d'une membrane...
- **Acquisition de l'information :** Clavier, micro, caméra, réseau
- **Restitution de l'information :** écran, haut-parleur, commande de robots, réseau

Codage de l'information dans un ordinateur

- Deux niveaux de tension électriques (0 volt et 5 volts)
- Circuits plus "simple" à concevoir
- Codage binaire : 1 et 0
- **Toute information est représentée par une suite de 1 ou 0**
- On parle de **codage digital** (circuit digital)
  - Vient de digit : doigt
- **Bit (contraction de Binary Digit)** code élémentaire : 1 ou 0 **(Chiffre binaire)**
- **Octet** : 8 bits
  - S'écrit souvent en **base 16 (chiffre hexadécimal)**
  - 4 bits = 1 chiffre hexadécimal (0 à 9, A, B, C, D, E, F)
  - Exemple : 0011 1010 (base 2)= 3A (base 16)

Quantité d'information

- Sur 8 bits : 256 valeurs. Sur n bits : 2^n valeurs
- On parle de kilo/méga/giga/téra octets (ou bits)
- Par habitude les quantités de stockage sont en octet (Byte); Les débits en bit
- On parle de Poids fort -Poids faible
  - Exemple : 1000 1011 :
    - 1000 : 4 bits de poids forts,
    - 1011 : 4 bits de poids faible
- Pour les quantités de stockage les ordres de grandeurs font référence au puissance de 2 :
  - Kilo : $2^{10}=1024 \neq 10^3 = 1000$
  - Méga : $2^{20}=1\,048\,576 \neq 10^6$
  - Giga : : $2^{30}=1\,073\,741\,824 \neq 10^9$
  - Téra : : $2^{40}=1\,099\,511\,627\,776 \neq 10^{12}$

Interprétation de l'information binaire

- Il peut y avoir différents codages numériques de la même information
- Exemple: 0 1 0 0 0 0 0 1 peut représenter l'entier 65 (en base 2), un caractère ("A" en code ASCII), une couleur, une instruction machine ....
- **C'est une question d'interprétation**
- **Exemple courant de mauvaise interprétation:**

  -Un fichier contenant des instructions machines est interprété comme des codes ASCII

(texte):<FE><ED><FA><CE>^@^@^@^R^@^@^@^@^@^@^B^@^@^@^K^@^@^D<F0>^@^@^@<

^@8__PAGEZERO^@^@^@^@^@^@^@^@^@^@^@^P^@^@^@^@^@^@^@^@^@^@^@^@^@^@^

^@^@^@__picsymbol_stub__TEXT^@^@^@^@^@^@^@^@^@^@^@^@^@3<D4>^@^@^Fx^@^@#<D4>^

g^@^@^@^@^@^@^@__TEXT^@^@^@^@^@^@^@^@^@^@^@^@^@:L^@^@^E<B4>^@^@*L^@^@^@
^B^@^@^@^@

Différents codages d'une même information

- Un caractère tapé au clavier
  - Une touche appuyée -> contact électrique particulier
    - Génération du code ASCII représentant ce caractère
    - Traitement par l'ordinateur de cette donnée, éventuellement transformation du code ASCII en valeur en base 2 (ou en complément à 2) si le caractère est un chiffre
- Affichage à l'écran :
  - Transformation du code ASCII en vecteurs de bits qui représentent les coordonnées des pixels à allumer à l'écran
  - Envoi d'électrons sur l'écran

Analogique / numérique

- **Représentation analogique :**
  - Sens physique, continu
  - Exemple : disque 33 tours vibration plus ou moins rapide d'un diamant qui va donner une variation de tension électrique
- **Représentation numérique ;**
  - Disque CD : trou au laser représentant des 1 (trou) et des 0 (pas de trou)
- Dans le cas d'un CD, l'information (le son ici) n'a pas besoin de changer de codage en entrée, elle peut être traitée directement
- Par contre à la restitution, il faudra la rendre analogique (tension électrique sur les hauts parleurs).

Codages binaires des caractères

- Code **ASCII** (American Standard Code for Information Interchange) sur 7 bits
- 1 bit pour les accents, non normalisé: problème classique du passage des caractères accentués entre différents types de d'ordinateur
- Exemple:
  - Code ASCII de a = 97 (base 10) = 61 en héxadécimal
  - Code ASCII de b= 98 (base 10)
- ***man ascii (sous unix)***

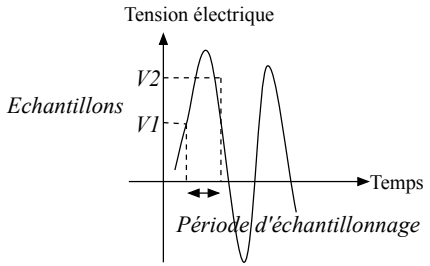
Codages des entiers

- Entiers naturels : **Base 2**
- Entiers relatifs : **complément à 2** (voir en TD) qui permet d'utiliser les mêmes opérations (et circuits) que pour la représentation en base 2
- Taille dépend du matériel, on parle de processeur 32 bits, 64 bits : taille des opérandes
- Taille supérieur possible mais opérations "décomposées" (Poids faibles - poids fort)

Codages des réels

- **Virgule flottante** : signe, mantisse, exposant
- Exemple en base 10:
  - 10,3 : $0{,}103.10^3$: signe +, mantisse entre 0 et 1: 0,103; exposant: 3
- Exemple en base 2 : norme 754 sur 32 bits
  - Signe s: 1 bit, mantisse sur 23 bits, exposant sur 8 bits
- Opération en virgule flottante:
  - Addition/soustraction:
    - Cadrage avant l'opération: même exposant
    - Addition/soustraction des mantisses
    - Post normalisation : premier chiffre après la virgule de l'exposant non nul
- Problème de **précision**: (normalisation) choix entre la taille de la mantisse et de l'exposant

Codage du son



- La qualité de restitution dépend de la fréquence d'échantillonnage et du nombre de bits pour coder la valeur des échantillons
- Exemple:
  - Qualité téléphone: 8000 fois par seconde, échantillon sur 1 octet -> 64 kilobits/s
  - CD Audio: 44100 Hertz, échantillon sur 2 octets : 700 kilobits/s (stéréo * 2)
- Compression possible :
  - Exemple de principe de compression: codage des différences entre les échantillons
  - exemple MP3: Suppression des fréquences non audibles pour l'oreille humaine: gain variable de 10 à 30 fois

Codage des images

- 1 pixel: point sur l'écran
- 1 image qualité télévision: 625*700 = 437 kilo-pixels
- 1 pixel: une couleur: 3 niveaux rouge/vert/bleu ou valeur dans une palette prédéterminée
  - Exemple: 1 octet par couleur: 3 octets : $\mathbf{2^{24}}$ nuances de couleurs (16 millions)
- 1 image couleur: 3*437= 1,3 mega octets pour une image
- On parle de format bit map (BMP) ou encore GIF, TIFF
- Compression possible : exemple JPEG: gain jusqu'à 50 fois
- Logiciel de compression/décompression

Codage de la vidéo

- **Vidéo : suite d'images + son**
  - film : 25 images /secondes
  - 1,3 méga octets * 25 = 33 mégaoctets / seconde !
- **Compression MPEG (***Moving Picture Experts Group)*
  - *Succession de différentes normes : MPEG 1, 2 et 4*
  - *MPEG1 :*
    - 352 x 288 à 25 images par seconde
    - Inclut le format audio MPEG-1 Layer 3 (MP3).
    - *codage JPEG puis codage des différences entre les images*
    - *Qualité télé: 1,5 Méga bit/s*
  - *MPEG4 :*
    - *Télé sur ADSL, Télé haute définition , TNT*
    - *5 à 7 mégabit/s*

Dans l'ordinateur "personnel"

- **Carte mère**

Type de circuits

•Microprocesseurs : calculs généraux

•Mémoires: RAM et ROM

•Digital Signal Processor (DSP) : calculs numériques généraux, spécialisés : addition / multiplication, matrices

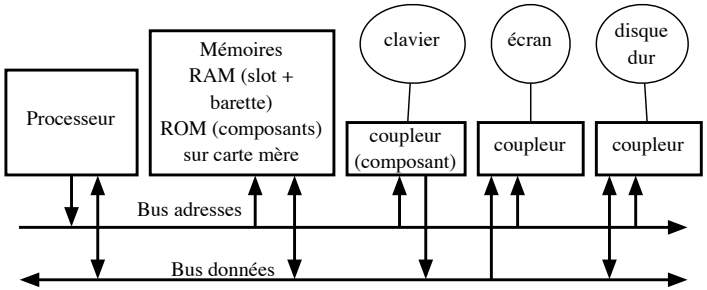
•Accélérateurs divers : compression image / son, cryptographie, ...

•Contrôleurs de périphériques : Ecran, disques, réseau, ...

•Gestionnaires internes : horloges, puissance, ...

Composants de l'ordinateur

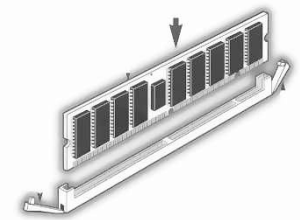
- **Description fonctionnelle**
  - **Bus** : nappe de fils
  - **Processeur** (ou CPU, unité centrale de traitement):
    - coeur de l'ordinateur ; exécute les calculs
  - **Périphériques** :
    - stockage, interfaces avec le monde extérieur
    - vu par le processeur comme des cases mémoires (à travers les coupleurs)



Mémoires

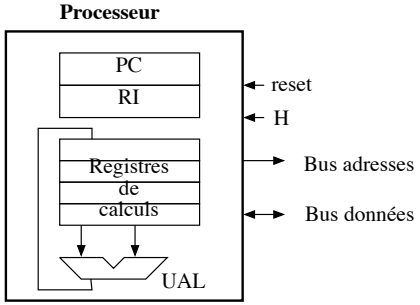
- Mémoire centrale/principale (Vive) : **RAM (Random Acces Memory)**,
  - rapide (quelques **dizaines de nano-secondes**),
  - contient les programmes en cours d'exécution, électrique donc non permanente
- Mémoire morte **ROM (Read only Memory)**:
  - contient les premiers programmes exécutés à la mise sous tension de l'ordinateur
- Mémoire **cache** :
  - encore plus petite et plus rapide (**quelques nano-secondes**),
  - contient aussi une partie des programmes en cours d'exécution, on l'oublie pour l'instant
- Mémoire **secondaire** :
  - disque dur,
  - beaucoup plus lente (**quelques milli-secondes**, facteur 10 puissance 6) mais sauvegarde permanente (magnétique)
  - Depuis quelques années: technologie SSD (mémoire flash comme les clés USB)

Mémoires RAM

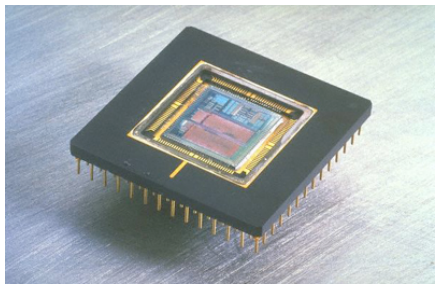


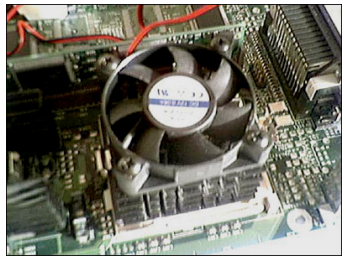
Processeur

- Contient quelques cases de mémoires appelées "registres" plus rapide que la RAM (facteur 10)
- Effectue des calculs grâce à une Unité Arithmétique et Logique (**UAL**)
- 1 registre particulier : **PC** (compteur programme) qui contient l'adresse de l'instruction en cours d'exécution
- 1 registre qui contient l'instruction machine en cours d'exécution (**RI**)
- **Reset** pour recommencer au début
- L'Horloge (**H**) cadence le rythme de calcul du processeur



Processeur





Différentes "formes" des programmes

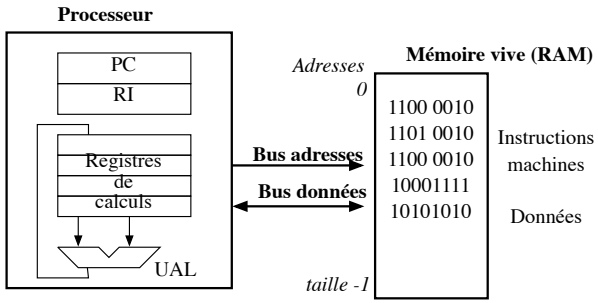
- **Programme** : suite d'**actions** (décidée par un **algorithme**) sur des **données**
- Programme sous différentes formes/langages :
  - Programme **source** dans un langage de haut niveau (ADA, C, PASCAL, JAVA…
  - Programme en **langage machine** (binaire) seul **exécutable** par le matériel
  - **Programme d'assemblage** : langage intermédiaire lisible (par l'homme) des instructions machines
- Passage d'un à l'autre grâce à un programme particulier : le **compilateur**
- Sur les premiers ordinateurs, les compilateurs n'existaient pas, les gens programmaient en binaire à l'aide d'interrupteurs.
- Les trois langages (source, assemblage, machine) sont codés en binaire : codes ASCII ou instructions machines

Compilateurs et exécutables

- **Langage machine propre à un processeur**.
- On ne peut pas utiliser un « exécutable » sur différents processeurs (PC et MAC par exemple)
- Il faut recompiler le programme source
- "Binaire universel" ? : contient les différents exécutables
- **Compilation**
  - Source -> Binaire/exécutable
  - Si plusieurs sources étape particulière : "Editions de liens"
- Interprétation : transformation faite pendant l'exécution du programme.
- Interprétation n'existe quasiment plus (ou après une transformation intermédiaire)
- Exemple d'interpréteur: Machine JAVA: programme qui interprète du "Bytecode Java" (forme intermédiaire entre source et exécutable)

Principe de l'exécution des programmes

- Instructions machine et données sont mises en mémoire vive (RAM)
  - C'est déjà un programme qui fait cela
- Le processeur lit en mémoire successivement les instructions machines en mettant sur le bus adresse l'adresse de l'instruction à exécuter
- Ces instructions peuvent être de différents types : lecture/écriture des données, calculs
- Enchaînement dans le temps séquentiel suivant l'écriture en mémoire, sauf instruction spéciale de branchement



Exécution des programmes

- **Interpréteur de commande:** programme permettant de lancer d'autres programmes (fait partie du système d'exploitation). Lancement d'un programme en donnant le nom de l'exécutable (ou double click sur une icône)
- **Chargeur** : programme particulier qui s'occupe de charger en mémoire l'exécutable lors de son lancement
- **Processus**: Programmes en cours d'exécution, il peut y en avoir plusieurs en même temps : entrelacement dans le temps de leur exécution par le processeur

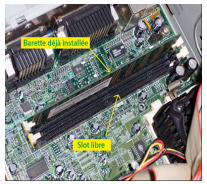
Programme source:
While I=0 do
a=a+b;

Compilation

Programme machine (exécutable):
(disque dur)
1000001010
1001010110
1010101011

Exécutable
sur Disque dur

Chargement



Exécutable
dans la RAM

Contenu de la mémoire pendant une compilation

- Système (Langage Machine) (en particulier interpréteur de commande)
- Données du système
- Compilateur (exécutable en Langage Machine)
- Données du compilateur ;
  - Source du programme à compiler (codes ASCII)
  - Exécutable du Programme compilé en LM (en construction)

Logiciels

- Programmes dans l'ordinateur
- Classement suivant leurs fonctionnalités
- Système d'exploitation :
  - Programmes basiques pour gérer les différents composants.
  - Propre à une machine
    - Interpréteur de commande (shell UNIX)
    - Chargeur
    - Dialogue avec les périphériques
    - …
- Compilateur : transformation source -> exécutable
- Applications diverses (éditeurs de texte, navigateur Web...)

Algèbre de Boole et fonctions booléennes

- Introduction
- Boole : mathématicien anglais du 19ème siècle qui s'intéresse aux propositions logiques
  - Proposition Vraie/Fausse
  - Et, Ou, complément
- Ordinateur : manipule des 0 et 1 (2 tensions électrique 0v et 5v)
  - 0 : faux
  - 1: vrai
  - L'étude des fonctions booléennes permet de construire des circuits électroniques

Définition d'un Algèbre de Boole

- Soit un ensemble A possédant 2 éléments particuliers notés 0 et 1
- A muni des opérations . (produit), + (somme) et $\overline{\ }$ (complémentation) est un algèbre de Boole si et seulement si il respecte les axiomes suivants :
  - A1 : . et + commutatives et associatives
    - $a.b= b.a,\ a+b= b+a,\ a.(b.c)= (a.b).c,\ a+(b+c)= (a+b)+c$
  - A2 : 0 est l'élément neutre pour +, 1 est l'élément neutre pour .
    - $0+a= a,\ 1.a= a$
  - A3 : . est distributive par rapport à + et inversement
    - $a.(b+c)= a.b+a.c\ ,\ a+(b.c)= (a+b).(a+c)$
  - A4 : pour tout a élément de A : $a+\overline{a}= 1$ et $a.\overline{a} = 0$

Un algèbre de Boole

- Il existe de nombreux algèbres de Boole
- L'exemple le plus simple d'algèbre de Boole est celui pour lequel A= {0, 1}. On notera B= {0,1}
- D'après les axiomes nous avons:
  - 1.0= 0, 1.1= 1, 0.0= 0
  - 1+0=1, 1+1=1, 0+0=0
  - $\overline{1}= 0, \overline{0}= 1$
- Propositions logiques :
  - produit: et
  - somme: ou
  - 1: vrai; 0: faux

Notations

- Le . est prioritaire sur le +. Le complément est prioritaire sur le + et le .
- On omet souvent les .
- Exemple :
  - ab+c= a.b+c= (a.b)+c
  - $\overline{ab} = (\overline{ab})$

Propriétés et Théorèmes

- **Dualité :**
  - Si (A, 0, 1, +, ., -) est un algèbre de Boole alors (A, 1, 0, ., +,-) est aussi un algèbre de Boole
  - Tous les axiomes (ou propriétés) sont toujours vrais si on remplace les + par des . et inversement , et les 0 par des 1 et inversement
  - Exemple: a.1= a ; dualité: a+ 0= a
- **Règles de De Morgan** :
  - $\overline{(a+b)} = \overline{a} \,.\, \overline{b}$
  - $\overline{(a.b)} = \overline{a} + \overline{b}$

Règles de simplification

- $\overline{\overline{a}} = a$ **duale:** $\overline{\overline{a}} = a$
- **a+1= 1 duale: a.0= 0**
- **a+a= a duale: a.a= a**
- **a + a.b = a duale: a.(a+b)= a**
- **a.b+** $\overline{a}$**.b = b duale: (a+b).(**$\overline{a}$**+b) = b**
- **a.b +** $\overline{a}$ **= b +** $\overline{a}$ **duale: (a+b).** $\overline{a}$ **= b.** $\overline{a}$
- **a.b +** $\overline{a}$**.c + b.c = a.b +** $\overline{a}$**.c**
- **A démontrer à partir des axiomes**

Fonctions booléennes

- **Définition :**
  - On appelle fonction booléenne à n variables une application de B^n dans B :
  - f : B^n -> B
  - $(x_1, x_2, \ldots, x_n) \rightarrow f(x_1, x_2, \ldots, x_n)$
- **Exemple à 2 variables :**

x_1, x_2	$f(x_1, x_2)$
0 0	0
0 1	0
1 0	0
1 1	1

Représentation des fonctions booléennes

- **Table de vérité**
  - Tableau: valeurs de la fonction pour chaque combinaison des valeurs des n variables
  - 2^n lignes
- **Expressions algébriques**
  - à partir des noms des variables et des opérateurs de l'algèbre de Boole
    - Exemple : $f(x_1,x_2) = x_1.\overline{x_2} + \overline{x_1}.(x_1 + x_2)$
  - Une fonction : une infinité d'expressions algébriques
  - Simplification des fonctions grâce aux règles de l'algèbre
    - Exemple : $f = ab + \overline{a}\, b\, c + \overline{c} + \overline{b} = ab + \overline{a}\, b + \overline{c} + \overline{b} = 1 + \overline{c} = 1$

Vocabulaire

- Monôme : produit de variables sous forme normale ou complémentée
  - -Exemple: $x_1.x_2.\overline{x_3}.x_4$
- Monal : somme de variables sous forme normale ou complémentée
  - -Exemple: $\overline{x_1}+x_2+x_3+\overline{x_4}$
- Forme polynomiale : somme de monômes
- Monômes canoniques : toutes les variables de la fonction apparaissent (complémentée ou non)
- Forme normal conjonctive: somme de monômes canoniques
- Forme normal disjonctive: somme de monaux canoniques

Expression algébrique à partir de la table de vérité

- **Théorème de Shannon :**

$f(x_1,x_2,\ldots,x_n)= \overline{x_i}\,.\, f(x_1,x_2,\ldots,0,\ldots,x_n) + x_i\,.\, f(x_1,x_2,\ldots,1,\ldots,x_n)$

- Exemple avec n=2 :
  - $f(x_1,x_2)= \overline{x_1}\,.\, f(0,x_2) + x_1\,.\, f(1,x_2)$
  - $f(x_1,x_2)= \overline{x_1}\,.\,(\overline{x_2}\,.\, f(0,0)+ x_2\,.\, f(0,1)\,)+ x_1\,.\,(\overline{x_2}\,.\, f(1,0)+ x_2\,.\, f(1,1)\,)$
  - $f(x_1,x_2)= \overline{x_1}\,.\overline{x_2}\,.\, f(0,0)+ \overline{x_1}.x_2\,.\, f(0,1) + x_1.\overline{x_2}\,.\, f(1,0)+ x_1.x_2\,.\, f(1,1)$
- **Passage évident de la table de vérité à une expression algébrique sous forme de somme de produits (forme normale)**

Exemple sur une table de vérité

- x_1, x_2 $f(x_1, x_2)$
- 0 0 $f(0,0)= 1$
- 0 1 $f(0,1)= 0$
- 1 0 $f(1,0)= 1$
- 1 1 $f(1,1)= 1$

- $f(x_1,x_2)= \overline{x_1}\,.\overline{x_2}\,.\, f(0,0)+ \overline{x_1}.x_2\,.\, f(0,1) + x_1.\overline{x_2}\,.\, f(1,0)+ x_1.x_2\,.\, f(1,1)$
- $f(x_1,x_2)= \overline{x_1}\,.\overline{x_2}\,.\, 1 + \overline{x_1}.x_2\,.\, 0 + x_1.\overline{x_2}\,.\, 1+ x_1.x_2\,.\, 1$
- $f(x_1,x_2)= \overline{x_1}\,.\overline{x_2} + x_1.\overline{x_2} + x_1.x_2 = \overline{x_1}\,.\overline{x_2} + x_1.\overline{x_2} + x_1.\overline{x_2} + x_1.x_2$
- $f(x_1,x_2)= \overline{x_2} + x_1$

Méthode

- On regarde les lignes où la fonction vaut 1
- On écrit les monômes avec la variable sous forme normal si elle vaut 1 , complémentée si elle vaut 0
- Possibilité d'obtenir la forme normale disjonctive (par dualité)
  - On regarde les lignes où la fonction vaut 0 et on écrit un produit de somme canonique en prenant la variable sous forme normale si elle vaut 0 et complémentée si elle vaut 1
- Exemple forme normale disjonctive:

x_1, x_2 $f(x_1, x_2)$

0 0 $f(0,0)= 1$

0 1 $f(0,1)= 0$ $\quad f(x_1, x_2)= (x_1 + \overline{x_2}).(\overline{x_1} + \overline{x_2})$

1 0 $f(1,0)= 1$ $\quad = \overline{x_1}.\,\overline{x_2} + x_1.\overline{x_2}$

1 1 $f(1,1)= 0$

Simplification

- Pour concevoir un circuit , on peut partir d'une expression algébrique
- Forme simplifiée permet d'obtenir le circuit le plus petit ou le plus rapide
- Formes et critères de simplification des expressions algébriques dépendent de la technologie utilisée, on comprendra plus tard
- Exemple: Forme polynomiale
  - Critère de simplification:
    - »Minimum de monômes et (en deuxième lieu) minimum de variables par monômes

Exemple de fonctions

- x_1, x_2 $f(x_1, x_2)$
- 0 0 $f(0,0)= 0$
- 0 1 $f(0,1)= 1$
- 1 0 $f(1,0)= 1$
- 1 1 $f(1,1)= 1$

- $f(x_1,x_2)= \overline{x_1}.x_2 + x_1.\overline{x_2} + x_1.x_2 = \overline{x_1}.x_2 + x_1.x_2 + x_1.\overline{x_2} + x_1.x_2$
- $f(x_1,x_2) = x_2 + x_1$
- Fonction Ou (Or)

Exemple de fonctions

- x_1, x_2 $f(x_1, x_2)$
- 0 0 $f(0,0)= 1$
- 0 1 $f(0,1)= 0$
- 1 0 $f(1,0)= 0$
- 1 1 $f(1,1)= 0$

- $f(x_1,x_2)= \overline{x_1}.\overline{x_2}$
- $f(x_1,x_2) = \overline{x_2 + x_1}$ par De Morgan
- Fonction Ou (Or)

Exemple de fonctions

- x_1, x_2 $f(x_1, x_2)$
- 0 0 $f(0,0)= 0$
- 0 1 $f(0,1)= 0$
- 1 0 $f(1,0)= 0$
- 1 1 $f(1,1)= 1$

- $f(x_1,x_2)= x_1.x_2$
- Fonction Et (And)

Exemple de fonctions

- x_1, x_2 $f(x_1, x_2)$
- 0 0 $f(0,0)= 1$
- 0 1 $f(0,1)= 1$
- 1 0 $f(1,0)= 1$
- 1 1 $f(1,1)= 0$

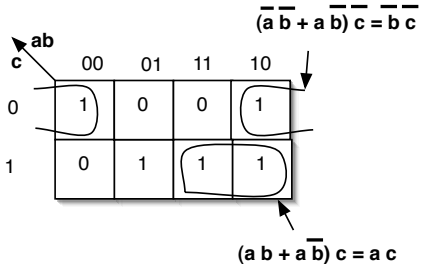
- $f(x_1,x_2)= \overline{x_1} + \overline{x_2} = \overline{x_1.x_2}$ par De Morgan
- Fonction NonEt (NAnd)

Quelques fonctions particulières

- Fonctions à 1 variable (4):
  - $f(x)= \overline{x}$: complément
  - $f(x)= x$
  - $f(x) = 1$ tautologie (toujours vrai)
  - $f(x) = 0$ (toujours faux)
- Fonctions à 2 variables (16):
  - $f(x,y) =x.y$: Et and
  - $f(x,y) =x+y$: Ou or
  - $f(x,y) = \overline{x}\ y + x\ \overline{y}$: Ou exclusif, appelé aussi XOR, noté $\oplus$
  - $f(x,y) = x\ y + \overline{x}\ \overline{y}$: Equivalence logique, coïncidence, noté $\odot$
  - $f(x,y) = \overline{(x.y)}$: non-et , nand
  - $f(x,y) = \overline{(x+y)}$: non-ou , nor
  - Même nom à n variables
- 2 puissance (2 puissance n) fonctions à n variables

Tableau de Karnaugh

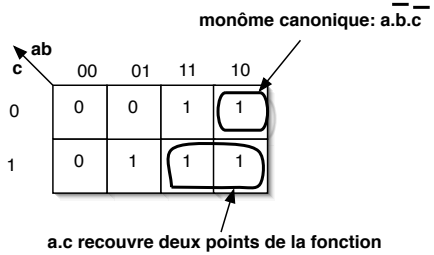
- Outil "manuel" de simplification des fonctions booléennes
- **Principe**
  - Table de vérité sous forme de tableau
  - Principe du regroupement de 2^n cases adjacentes
  - Exemple f(a,b,c)



  - Attention à l'ordre des colonnes et des lignes

Monômes et points d'une fonction booléenne

- Un point d'une fonction: un monôme canonique: une case à 1 du tableau de Karnaugh
- On dit qu'un monôme "recouvre" des points de la fonction
- Un monôme à k variables (parmi n) couvre 2^{n-k} point



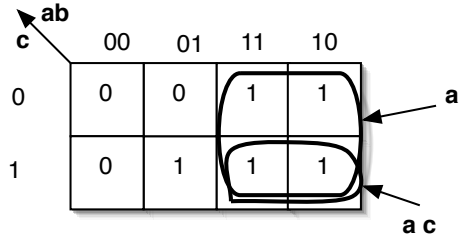
Principe de la minimisation d'une forme polynomiale

•**Définitions :**

- Relation d'ordre sur les expressions algébriques booléennes : f≤g ssi f.g = f
- Exemple : a.c < a car a.c.a = a. c

•**Monôme premier** d'une fonction f :

- m ≤ f et il n'existe pas monôme m' tel que m'≤ f et m'≤ m
- Monôme premier: "plus gros regroupement" de 2^k points sur le tableau de Karnaugh



Minimisation

•**Base d'une fonction**

-Somme de monômes premiers tel que tous les points de f sont couverts par au moins un de ces monômes

•**Base complète :** Tous les monômes premiers de la fonction

•**Base irrédondante d'une fonction** :

-Si on enlève un de ses monômes ce n'est plus une base de f

-Il peut y avoir plusieurs bases irrédondantes

•**But de la minimisation:**

-Critère : nombre minimal de monômes et de variables par monômes

-Trouver une base irrédondante possédant le moins possible de monôme

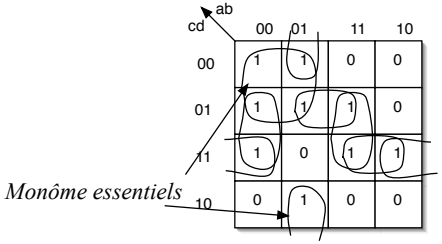
-Problème NP complet : il faut trouver toutes les bases irrédondantes pour trouver la (ou les) minimales

Algorithme de minimisation

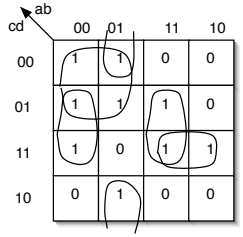
- Trouver la base complète
- Garder les monômes essentiels (seul à couvrir au moins un point)
- Eliminer les monômes inutiles (tous les points qu'il couvre sont couverts par des essentiels
- Essayer tous les cas pour couvrir les points non couverts par les essentiels
- Heuristique : prendre d'abord ceux qui couvrent le plus de points non encore couverts par les monômes déjà choisis

Exemple sur tableau de Karnaugh

•Base complète



• Une base irrédondante non minimale (5)



Une base irrédondante minimale (4)

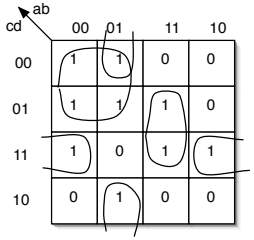
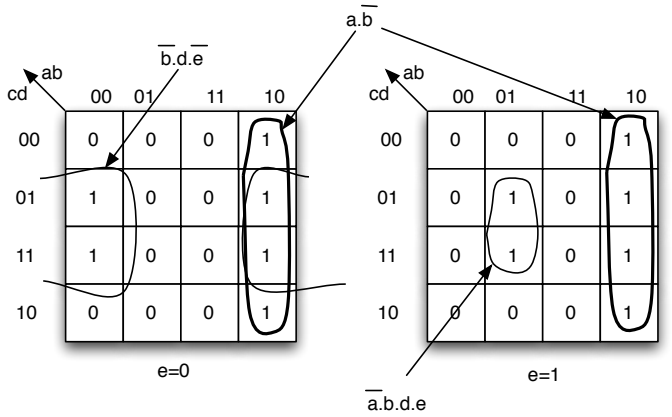


Tableau de Karnaugh à 5 variables

On peut le voir comme deux tableaux à 4 variables superposés



Fonctions phi-booléennes

•**Définition**:

- On appelle fonction phi-booléenne à n variables une application de B^n dans $\{0,1,\Phi\}$:
  - $f : B^n \rightarrow \{0, 1, \Phi\}$
  - $(x_1,x_2, \ldots,x_n) \rightarrow f(x_1,x_2, \ldots,x_n)$

•**Remarque :** phi : 0 superposé à 1 : Φ

•Les points à Φ de la fonction représentent des cas « non-existants »

•Les phis n'existent pas dans les circuits, seulement des 1 ou des 0 (réalisation de fonctions booléennes)

•On veut trouver une expression algébrique minimale d'une fonction booléenne égale à la fonction phi-booléenne où l'on a remplacé les phi par des 1 ou des 0

Exemple

•**Exemple :** Une fonction de 4 variables (x_4,x_3, x_2, x_1) qui vaut 1 si l'entier compris entre 0 et 9 représenté en base 2 par les 4 variables est impair (x_1 poids faible)

x2x1 \ x4x3	00	01	11	10
00	0	0	phi	0
01	1	1	phi	1
11	1	1	phi	phi
10	0	0	phi	phi

Bornes d'une fonction phi-booléenne

- **Borne inférieure d'une fonction phi-booléenne f** :
  - la fonction booléenne où tous les phi ont été remplacés par des 0
- **Borne supérieure d'une fonction phi-booléenne f** :
  - la fonction booléenne où tous les phi de f ont été remplacés par des 1
- **Minimisation d'une fonction phi-booléenne :**
  - But obtenir une fonction booléenne en remplaçant les phi au choix par 0 ou 1

•Les monômes premiers d'une fonction phi-booléenne sont ce de sa borne supérieure

•Une base d'une fonction phi-booléenne est une somme de monômes premiers (de la borne supérieure) qui couvrent tous les points de la borne inférieure.

•Certains phis sont remplacés par des 0 et d'autres par des 1

Exemple de minimisation phi-booléenne

- Fonction phi-booléenne

cd \ ab	00	01	11	10
00	Phi	Phi	0	0
01	0	1	1	0
11	0	0	1	Phi
10	0	0	1	1

- Borne supérieure

cd \ ab	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	0	0	1	1

Borne inférieure

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	0	1	0
10	0	0	1	1

cd \ ab	00	01	11	10
00	phi ->0	phi ->0	0	0
01	0	1	1	0
11	0	0	1	phi ->1
10	0	0	1	1