Architecture des systèmes

Chapitre 1 : Fonctionnement d'un ordinateur

Objectifs du chapitre

- Fonctionnement d'un ordinateur
  - stockage : des bits aux valeurs
  - traitement
    - par des circuits simples (sans notion de temps et sans stockage)
    - par des circuits complexes (avec notions de temps et de stockage)
    - entre circuits complexes

Plan

- 1. Représentation des valeurs
- 2. Circuits logiques
- 3. Circuits séquentiels
- 4. Architecture d'un ordinateur

1. Représentation des valeurs

1. Représentation des valeurs

- Bit : unité de stockage de l'information
  - 0 ou 1
- Stockage de valeurs plus complexes
  - nombres, textes, images, sons, vidéos, etc.
- Comment sont stockées ces valeurs ?
  - entiers positifs, entiers naturels
  - nombres réels
  - caractères, chaînes de caractères

1.1 Entiers positifs

- Base = nombre de chiffres différents autorisés
- Base 2 (binaire)
  - valeurs : 0 et 1
- Base 10 (décimale)
  - valeurs : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9
- Base 16 (hexadécimale)
  - valeurs : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F

1.1 Entiers positifs

- Remarque
  - pour éviter les confusions, on utilise la base en indice
  - un nombre en base hexadécimale peut aussi être précédé de 0x
  - par défaut, la base est décimale
- Exemple
  - 25 est un nombre en base 10
  - 100 est un nombre en base 10
  - 100_2 est un nombre en base 2
  - AB_{16} est un nombre en base 16
  - 0xAB est un nombre en base 16

1.1 Entiers positifs

- Comment connaître la valeur décimale d'un nombre écrit dans une base *b* ?
  - soit un nombre binaire $(a_{n-1},a_{n-2},\ldots,a_2,a_1,a_0)$, chaque a_i étant écrit dans la base *b*
  - la valeur de $(a_{n-1},a_{n-2},\ldots,a_2,a_1,a_0)$ est $sum(i=0..n-1)a_i.b^i$
- Exemple : quelle est la valeur décimale de 110101_2 ?
  - $1.2^5+1.2^4+0.2^3+1.2^2+0.2^1+1.2^0=32+16+4+1=53$

1.1 Entiers positifs

- Comment connaître l'écriture dans une base *b* d'un nombre *n* écrit en base décimale ?
  - écrire les puissances de *b*
  - calculer *c* et *k* tel que :
    - *c* est compris entre 0 et *b*-1
    - $c.b^k \leq n < (c+1).b^k$
  - retirer du nombre actuel $c.b^k$: *c* est le *k*-ème chiffre de l'écriture de *n* dans la base *b*
  - recommencer jusqu'à obtenir 0

1.1 Entiers positifs

- Exemple : quelle est l'écriture binaire de 39 ?
  - puissances de 2 : 1, 2, 4, 8, 16, 32, 64
  - $1.32 \leq 39 < 2.32$, avec $32=2^5$, donc le 5-ème chiffre est 1, et il reste 7
  - $1.4 \leq 7 < 2.4$, avec $4=2^2$, donc le 2-ème chiffre est 1, et il reste 3
  - $1.2 \leq 3 < 2.2$, avec $2=2^1$, donc le 1-ème chiffre est 1, et il reste 1
  - $1.1 \leq 1 < 2.1$, avec $1=2^0$, donc le 0-ème chiffre est 1, et il reste 0
  - on obtient 100111_2

1.1 Entiers positifs

- Comment passer d'une base b_1 à une base b_2 ?
  - passer de la base b_1 à la base 10
  - passer de la base 10 à la base b_2
- Comment passer d'une base 16 à une base 2 rapidement ?
  - traduire chaque chiffre hexadécimal en 4 bits
- Comment passer d'une base 2 à une base 16 rapidement ?
  - lire les bits par blocs de 4, en commençant à droite
  - traduire chaque bloc de 4 bits indépendemment (car $16=2^4$)

1.1 Entiers positifs

- Comment faire une addition entre deux nombres ?
  - comme sur les décimaux
  - attention aux retenues
  - attention aux débordements de capacité (le résultat peut avoir plus de chiffres que les opérandes)

1.1 Entiers positifs

- Exemple : faire l'addition de 39 et de 46 en base 2
  - $39 = 100111_2$ et $46 = 101110_2$
  - on obtient bien $1010101_2 = 85$

Addition de 39 et 46							
39		1	0	0	1	1	1
46		1	0	1	1	1	0
retenue	1	0	1	1	1	0	-
résultat	1	0	1	0	1	0	1

1.2 Entiers naturels

- Problème : il faut un moyen pour stocker les nombres négatifs
- Solution initiale : un bit de signe (le plus à gauche)
- Exemple : représenter 39 et -39 en base 2, avec un bit de signe, sur 8 bits
  - 39 = 100111
  - 39 sur 8 bits : (0)0100111
  - -39 sur 8 bits : (1)0100111

1.2 Entiers naturels

- Problèmes
  - les additions et les soustractions sont à gérer différemment
  - 0 et -0 ont deux représentations différentes
- Solution intermédiaire : complément logique (ou complément à 1)
  - pour les entiers négatifs, tous les bits sont inversés
- Exemple : représenter 39 et -39 en base 2, en complément logique, sur 8 bits
  - 39 = 100111
  - 39 sur 8 bits : 00100111
  - -39 sur 8 bits : 11011000

1.2 Entiers naturels

- Remarque : les nombres positifs commencent par 0, les nombres négatifs par 1
- Comment ajouter deux entiers naturels en complément logique ?
  - faire l'addition en base 2
  - si une retenue déborde (dépasse de la capacité prévue), il faut la réajouter au résultat
- Problèmes
  - addition complexe
  - 0 et -0 ont toujours deux représentations différentes
- Solution : complément arithmétique (ou complément à 2)
  - obtenu en ajoutant 1 au complément logique pour les nombres négatifs

1.2 Entiers naturels

- Exemple : représenter 39 et -39 en base 2, en complément arithmétique sur 8 bits
  - 39 = 100111
  - 39 sur 8 bits : 00100111
  - -39 sur 8 bits : 11011000+1 = 11011001
- Comment ajouter deux entiers naturels en complément arithmétique ?
  - faire une addition en base 2
  - si une retenue déborde, il faut l'ignorer

1.3 Réels

- Problème : il faut un moyen pour représenter les nombres à virgule
- Solution : format IEEE 754, simple ou double précision
  - signe : bit de signe
  - valeur absolue : mantisse normalisée et exposant $(m.2^e)$, avec $0.5 \leq m < 1$ et e entier
- Mantisse à bit caché
  - le premier bit de la mantisse, qui vaut toujours 1, n'est pas stocké
  - zéro est stocké avec un exposant maximum

1.3 Réels

- Exposant biaisé
  - l'exposant est décalé d'un biais pour que la valeur stockée soit toujours positive
- Remarques
  - simple précision : 1 bit de signe, 23 bits de mantisse, 8 bits d'exposant (biaisé de -126)
  - double précision : 1 bit de signe, 52 bits de mantisse, 11 bits d'exposant (biaisé de -1022)

1.3 Réels

- Exemple : quelle est la valeur décimale de $11000000\ 00110000\ 00000000\ 00000000_2$?
  - bit de signe : 1, donc nombre négatif
  - exposant : 10000000=128 (valeur biaisée), 128-126=2 (valeur non biaisée)
  - mantisse : 01100...0 (avec bit caché), 101100...0 (sans bit caché) = 1/2+1/8+1/16 = 0,5+0,125+0,0625 = 0,6875
  - résultat : $-\ 0{,}6875.2^2 = -2{,}75$

1.3 Réels

- Exemple : quel est l'encodage (en simple précision) de 4,25 ?
  - nombre positif, donc bit de signe : 0
  - $4{,}25 = 2{,}125*2^1 = 1{,}0625*2^2 = 0{,}53125*2^3$, arrêt car mantisse normalisée
  - exposant : 3 (valeur non biaisé), 3+126=129 (valeur biaisée), donc : 10000001
  - mantisse : 0,53125 = 0,5 + 0,03125 = 1/2 + 1/32 = 1000100...0 (sans bit caché), 000100...0 (avec bit caché)
  - résultat : $01000000\ 10001000\ 00000000\ 00000000_2$

1.4 Caractères

- Comment les caractères sont-ils stockés ?
  - règles associant chaque caractère à un numéro
  - exemple : ASCII, UTF-8, UTF-16, ISO 8859-1 (latin1), etc
- ASCII = American Standard Code for Information Interchange
  - table associant chaque caractère (parmi 128) à un numéro sur un octet
  - exemples : 'A' est 65, 'B' est 66, 'a' est 97, '+' est 43

1.4 Caractères

- Propriétés
  - les caractères minuscules se suivent
  - les caractères majuscules se suivent
  - les chiffres se suivent
- Problèmes
  - pas de caractères accentués
  - beaucoup (33) de caractères non affichables

1.5 Chaînes de caractères

- Deux méthodes
  - stocker le nombre de caractères puis la liste des caractères (approche utilisée en Pascal)
  - stocker un caractère terminal (approche utilisée en C ou sous DOS)

2. Circuits logiques

2. Circuits logiques

- Circuit logique
  - circuit électronique simplifié permettant de calculer une sortie en fonction d'entrées
  - exemple : allumer une lumière en fonction de la position de plusieurs interrupteurs
- Remarque
  - les entrées et les sorties sont composées de signaux binaires
  - la manipulation des bits se fait via des fonctions logiques

2. Circuits logiques

- Fonctions logiques
  - fonctions de base : NOT, OR, AND

Table de vérité de “NOT A”	
A=0	1
A=1	0

Table de vérité de “A OR B”		
	B=0	B=1
A=0	0	1
A=1	1	1

Table de vérité de “A AND B”		
	B=0	B=1
A=0	0	1
A=1	1	1

  - autres fonctions : XOR, NOR, NAND

Table de vérité de “A XOR B”		
	B=0	B=1
A=0	0	1
A=1	1	0

Table de vérité de “A NOR B”		
	B=0	B=1
A=0	1	0
A=1	0	0

Table de vérité de “A NAND B”		
	B=0	B=1
A=0	1	1
A=1	1	0

2. Circuits logiques

- Algèbre de Boole
  - opérations élémentaires : NOT (noté barre), OR (noté +), AND (noté * ou .)
  - permet de représenter des fonctions logiques, de les simplifier, et de faire du calcul (appelé calcul propositionnel)
- Quelques propriétés
  - $\overline{\overline{A}}=A$
  - lois de De Morgan : $\overline{A+B}=\overline{A}.\overline{B}$ et $\overline{A.B}=\overline{A}+\overline{B}$

2. Circuits logiques

- Table de vérité
  - liste exhaustive des sorties en fonction de toutes les entrées
- Exemple
  - la lumière L est allumée uniquement quand les interrupteurs A ou B sont ouverts, à moins que l'interrupteur de sécurité S soit ouvert
  - entrées : A, B, S
  - sortie : L

2. Circuits logiques

- Suite de l'exemple :

S	A	B	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

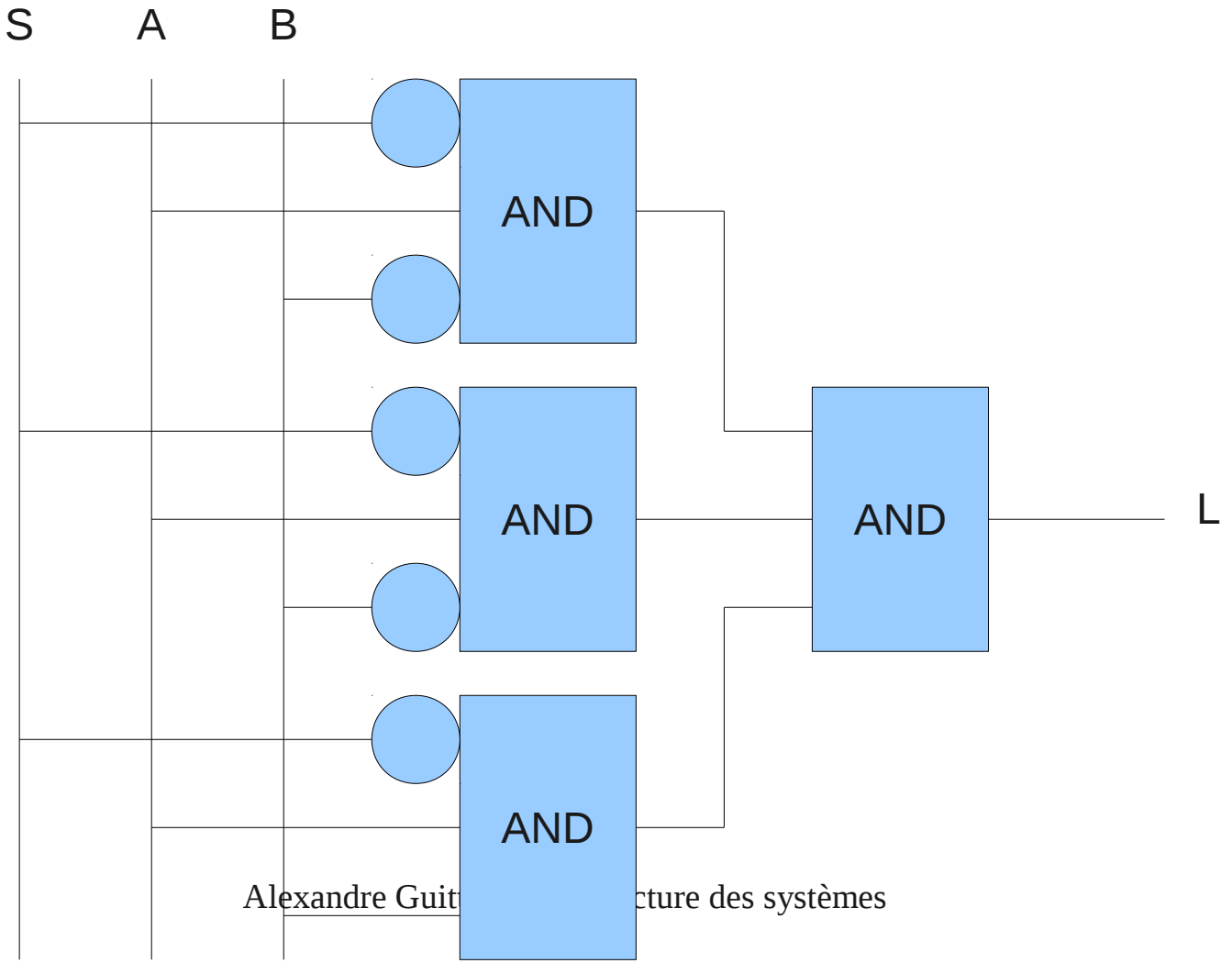
- fonction logique : $L=\overline{S}.A.\overline{B}+\overline{S}.A.\overline{B}+\overline{S}.A.B$

2. Circuits logiques

- Synthèse
  - construction d'un circuit logique à partir d'une fonction logique
- Étapes de la synthèse
  - construction de la table de vérité
  - obtention d'une expression logique
  - réalisation du circuit

2. Circuits logiques

- Exemple : $L=\overline{S}.A.\overline{B}+\overline{S}.A.\overline{B}+\overline{S}.A.B$



2. Circuits logiques

- Analyse
  - construction d'une fonction logique à partir d'un circuit logique
- Simplification des fonctions logiques (cf TD)
  - par observation de la table de vérité
  - par propriétés de l'algèbre de Boole
  - par la méthode des tableaux de Karnaugh

3. Circuits séquentiels

3. Circuits séquentiels

- Circuit séquentiel
  - circuit logique qui intègre une notion de temps et de mémoire
  - temps : horloge fournissant un signal (à une certaine fréquence)
  - mémoire : bascule RS
- Bascule RS
  - une entrée R (pour reset) : met la sortie à 0
  - une entrée S (pour set) : met la sortie à 1

3. Circuits séquentiels

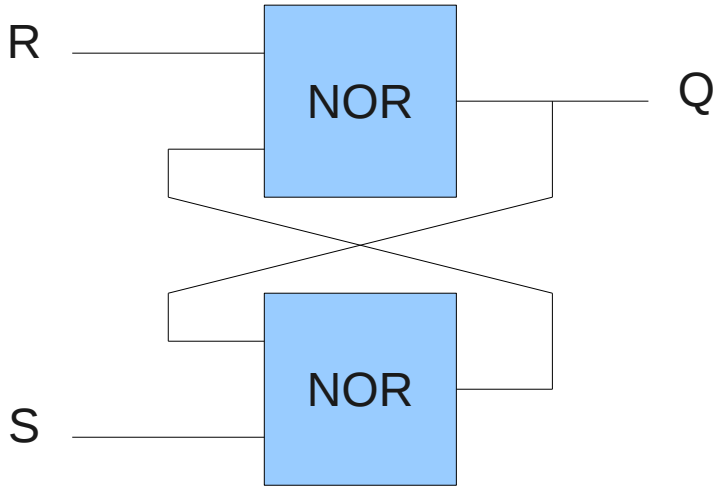
- Table de vérité de la bascule RS

Q (ancien)	R	S	Q (nouveau)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	???
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	???

  - fonction logique : $Qnew=\overline{R}.S+\overline{R}.Qold=\overline{R}.(S+Qold)=R+\overline{(S+Qold)}$

3. Circuits séquentiels

- Circuit séquentiel de la bascule RS
  - fonction logique : $Q=\overline{R+\overline{(S+Q)}}$

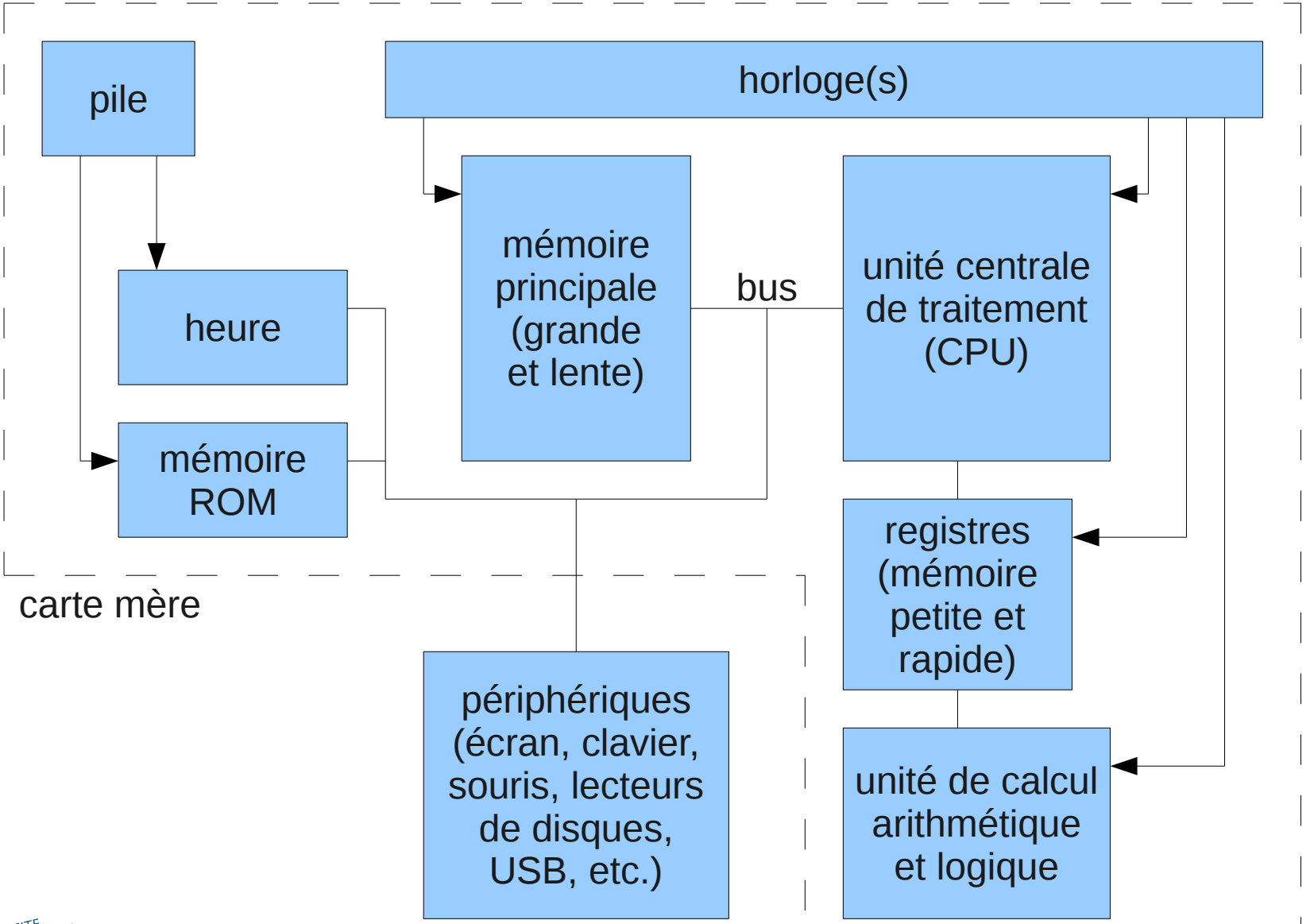


4. Architecture d'un ordinateur

4. Architecture d'un ordinateur

- Ordinateur
  - interconnexion de circuits séquentiels + horloge
  - bus : moyen de communication entre les circuits séquentiels
  - modèle de Von Neumann
- Types de circuits séquentiels
  - mémoire
  - unités de calculs
  - unités de traitement
  - périphériques externes

4. Architecture d'un ordinateur



4. Architecture des ordinateurs

- Quelques mots-clés
  - RAM *(Random Access Memory)* = mémoire principale, en lecture/écriture
  - ROM *(Read-Only Memory)* = mémoire en lecture seule
  - BIOS *(Basic Input/Output System)* = primitives de base de gestion des entrées/sorties, disponibles avant le démarrage du système d'exploitation, sauvegardées en ROM
  - CMOS = technologie mémoire historiquement utilisée pour sauvegarder quelques informations du BIOS en mémoire quand l'ordinateur est éteint

4. Architecture des ordinateurs

- Quelques mots-clés
  - circuit intégré = circuit logique ou séquentiel réalisant une ou plusieurs fonctions
  - processeur = composant électronique réalisant des traitements programmables
  - microprocesseur = processeur réalisé sur un seul circuit intégré

Résumé des principales connaissances à avoir

Résumé des principales connaissances à avoir

- Codage : entiers positifs, entiers naturels, réels, caractères
- Circuits logiques : synthèse, analyse
- Circuits séquentiels : principe
- Ordinateur : organisation