Formation à enseigner l'ISN
Séance 1: Machines

Manuel Gobillot et Cyrille Guieu

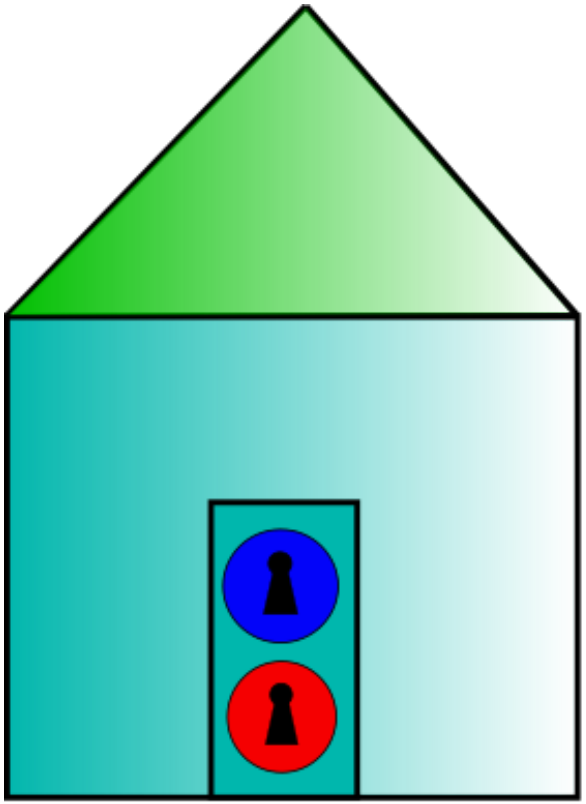
Rectorat de la Guyane

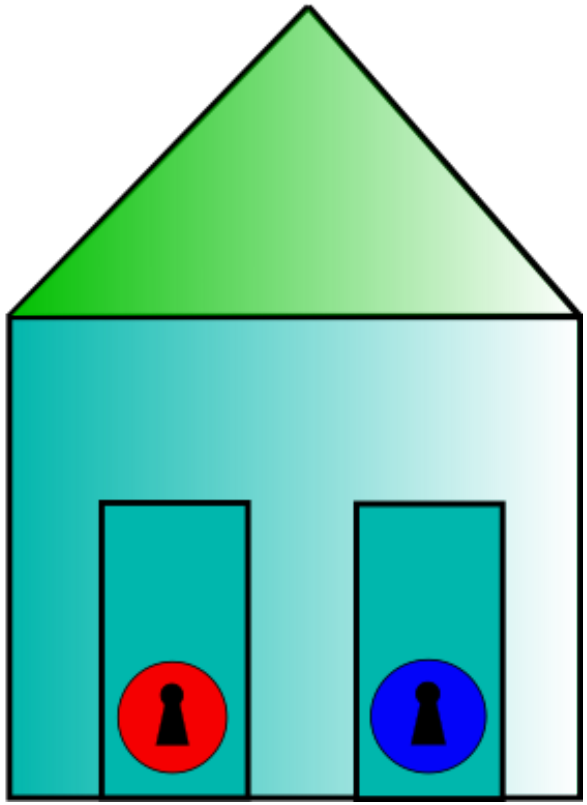
11 octobre 2107

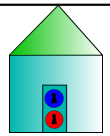
L'outil informatique est directement issue d'un besoin (pour l'homme) de résoudre des problématiques de plus en plus complexes.

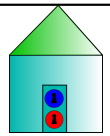
L'outil informatique est directement issue d'un besoin (pour l'homme) de résoudre des problématiques de plus en plus complexes. On peut, par exemple, considérer la création d'un local verrouillé comme un premier pas dans cette évolution :
Exemple de cas : une pièce équipée d'une porte à 2 serrures / une pièce équipée de 2 portes avec chacune sa serrure.

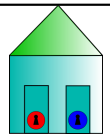
Un exemple en mécanique
Étude du cas





	Ouverte	Fermée	Ouverte	Fermée
	Ouverte	Ouverte	Fermée	Fermée
	Ouverte	Fermée	Fermée	Fermée

	Ouverte	Fermée	Ouverte	Fermée
	Ouverte	Ouverte	Fermée	Fermée
	Ouverte	Fermée	Fermée	Fermée

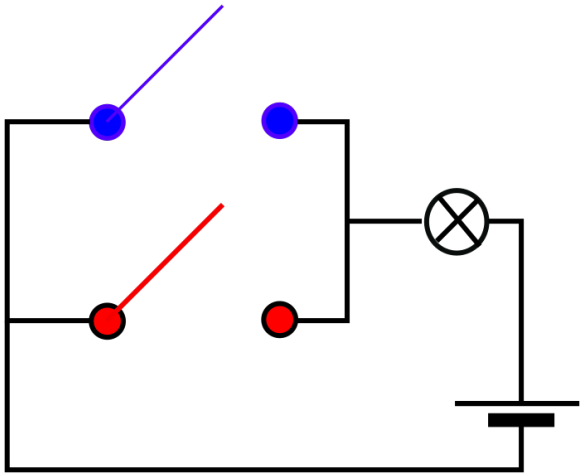
	Ouverte	Fermée	Ouverte	Fermée
	Ouverte	Ouverte	Fermée	Fermée
	Ouverte	Ouverte	Ouverte	Fermée

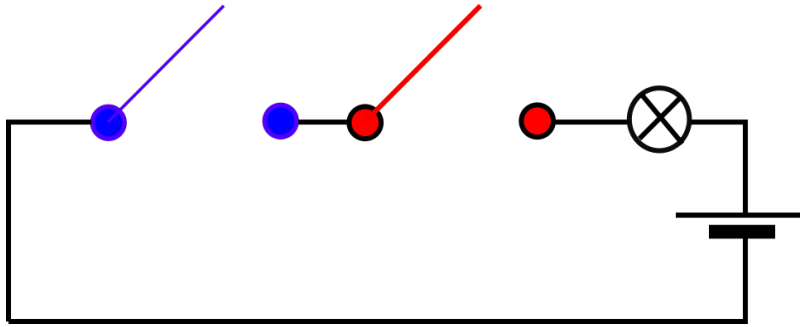
Étude du cas : Bien que le nombre de serrures soit identique (2), il faut disposer des clés des 2 serrures pour entrer dans la première pièce (ET Logique), tandis qu'une seule des 2 clés suffit dans le second cas (OU Logique).

A Noter : Les mathématiques sont dès lors un facteur d'amélioration évident menant à la création d'outils de plus en plus complexes jusqu'à l'ordinateur d'aujourd'hui.

Un exemple en électricité

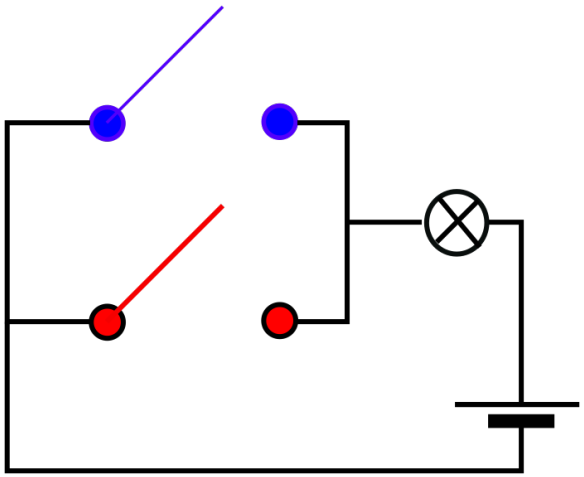
Deux interrupteurs, trois solutions

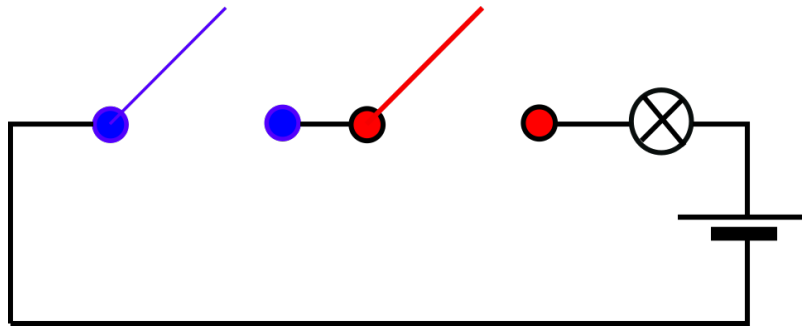




Un exemple en électricité

Deux interrupteurs, trois solutions

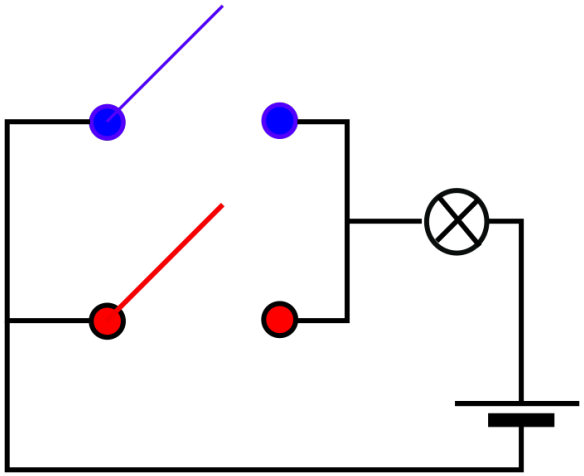


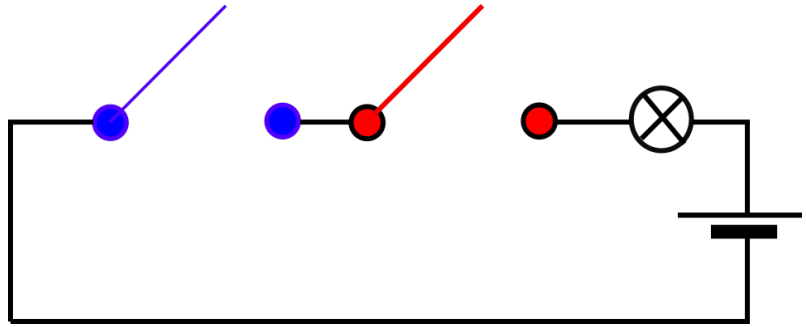


Les circuits électriques parallèles(OU Logique) ou en série(ET Logique) sont connus

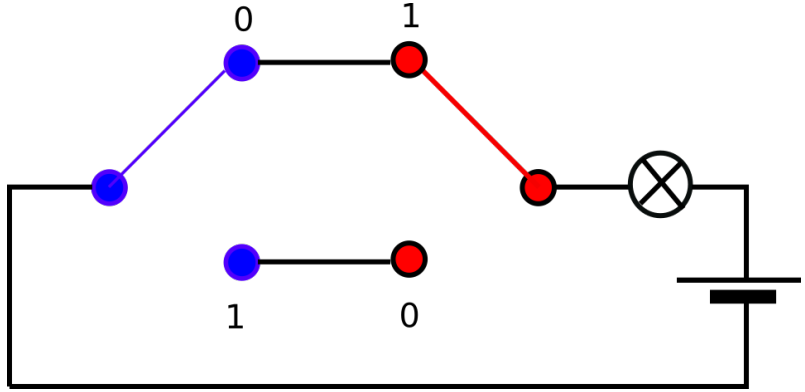
Un exemple en électricité

Deux interrupteurs, trois solutions





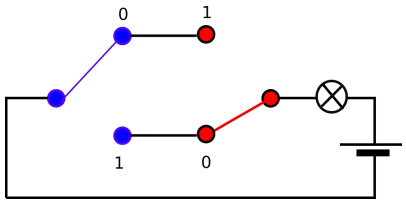
Les circuits électriques parallèles(OU Logique) ou en série(ET Logique) sont connus .... *mais qu'en est-il d'un va-et-viens ?*



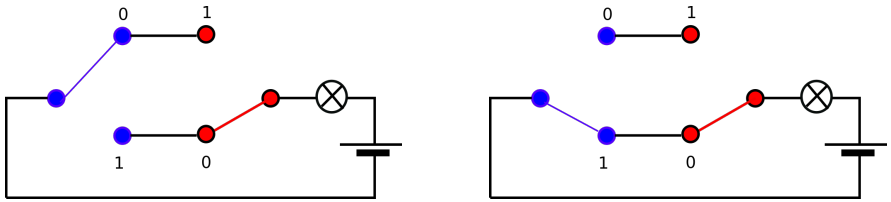
Un moyen d'aborder le sujet est d'effectuer ce qu'on appelle "un jeu d'essai" ou jouer à essayer ... (mais quand s'arrêter ?)

Un exemple en électricité
Étude du cas

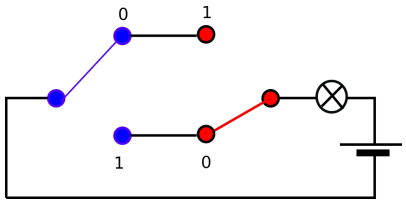
Un moyen d'aborder le sujet est d'effectuer ce qu'on appelle "un jeu d'essai" ou jouer à essayer ... (mais quand s'arrêter ?)

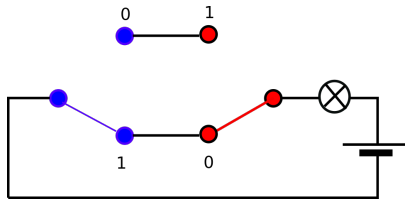


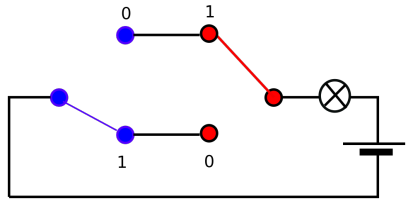
Un moyen d'aborder le sujet est d'effectuer ce qu'on appelle "un jeu d'essai" ou jouer à essayer ... (mais quand s'arrêter ?)



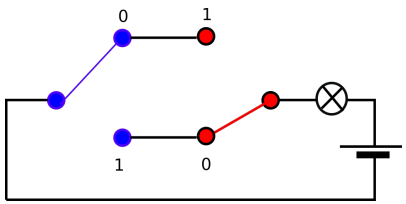
Un moyen d'aborder le sujet est d'effectuer ce qu'on appelle "un jeu d'essai" ou jouer à essayer ... (mais quand s'arrêter ?)

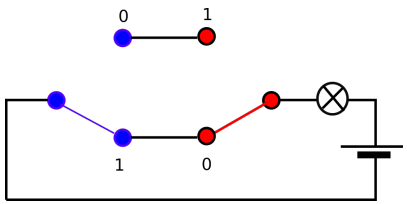


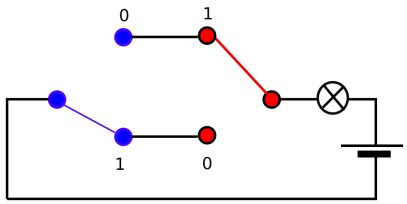


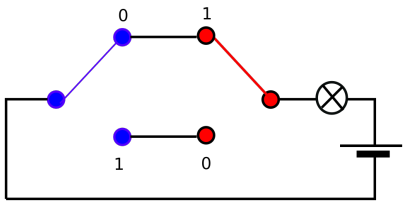


Un moyen d'aborder le sujet est d'effectuer ce qu'on appelle "un jeu d'essai" ou jouer à essayer ... (mais quand s'arrêter ?)

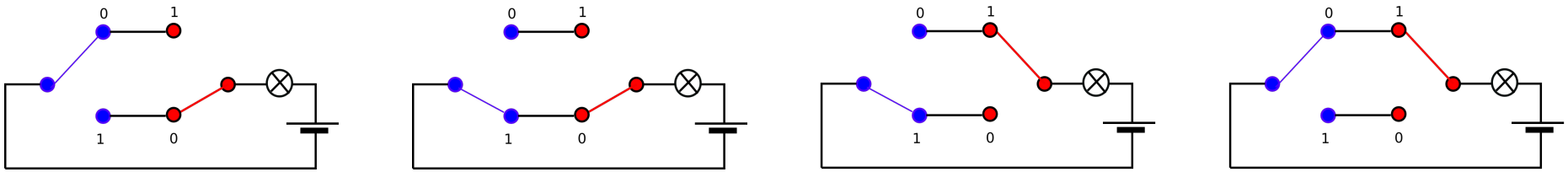








Un moyen d'aborder le sujet est d'effectuer ce qu'on appelle "un jeu d'essai" ou jouer à essayer ... (mais quand s'arrêter ?)

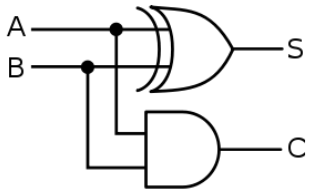


Considérons une lumière éteinte : [0] [0] = éteint
On allume d'un côté : [1] [0] = allumé
On appuie sur l'autre interrupteur : [1] [1] = éteint
On appuie sur le premier interrupteur : [0] [1] = allumé
Quelle que soit l'action suivante, nous retomberons sur un état déjà étudié, le jeu est donc terminé.

L'électronique permet entre autres, la création de « circuits logiques » effectuant le même type de travaux (ET/OU ...) et bien plus.

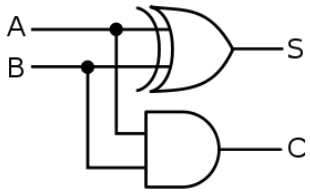
L'électronique permet entre autres, la création de « circuits logiques » effectuant le même type de travaux (ET/OU ...) et bien plus.

Additionneur deux bits :

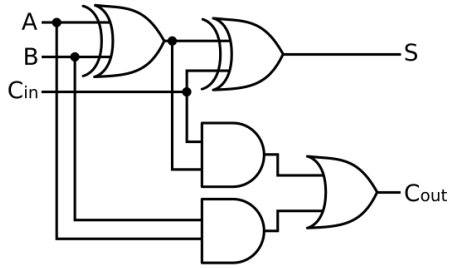


L'électronique permet entre autres, la création de « circuits logiques » effectuant le même type de travaux (ET/OU ...) et bien plus.

Additionneur deux bits :



Additionneur deux bits avec retenue :



Elle est avant tout utilisée pour effectuer des opérations complexes prédéfinies(ou pré-programmées) : une radio (réglage volume et fréquence) par exemple ...

Elle est avant tout utilisée pour effectuer des opérations complexes prédéfinies(ou pré-programmées) : une radio (réglage volume et fréquence) par exemple ...
L'ordinateur (à programme enregistré) naît ainsi de la volonté d'intégrer un ensemble de circuits logiques (ET/OU...) et arithmétiques (1+2=3...) au sein d'un même composant (l'unité arithmétique et logique) sans connaître par avance l'utilisation qui en sera faîte.

Dans les faits, un lien étroit existe entre circuits logiques (ET/OU ...) et circuits arithmétiques (calculs).
La représentation binaire, qui n'autorise que 2 états (0/1) par rang (bit), convient parfaitement aux traitements logiques (ET/OU ...) basés sur le Vrai(1) et le Faux(0). Le binaire représente de plus la petite base arithmétique permettant l'écriture de nombres réels.
Plus simplement : on peut traduire toute forme de données numériques en binaire comme dans les autres bases.

Exemple de conversion de base 2 en base 10 :
Soit le nombre 10011 écrit en base 2.
On peut le noter par exemple b10011.

Exemple de conversion de base 2 en base 10 :
Soit le nombre 10011 écrit en base 2.
On peut le noter par exemple b10011.
Pour le convertir en base 10 on attribue à chaque chiffre égal à 1 sa valeur en puissance de deux :

$$\text{b10011} = 2^0 + 2^1 + 2^4 = 1 + 2 + 16 = 19$$

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.
Exemple : convertir 18 en base 2

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.
Exemple : convertir 18 en base 2
$18/2 = 9$(reste 0) donc 18 = b.....0

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.
Exemple : convertir 18 en base 2
$18/2 = 9$(reste 0) donc 18 = b.....0
$9/2 = 4$(reste 1) donc 18 = b....10

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.
Exemple : convertir 18 en base 2
$18/2 = 9$(reste 0) donc 18 = b.....0
$9/2 = 4$(reste 1) donc 18 = b....10
$4/2 = 2$(reste 0) donc 18 = b....010

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.

Exemple : convertir 18 en base 2

$18/2 = 9$(reste 0) donc 18 = b.....0

$9/2 = 4$(reste 1) donc 18 = b....10

$4/2 = 2$(reste 0) donc 18 = b....010

$2/2 = 1$(reste 0) donc 18 = b...0010

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.

Exemple : convertir 18 en base 2

$18/2 = 9$(reste 0) donc 18 = b.....0

$9/2 = 4$(reste 1) donc 18 = b....10

$4/2 = 2$(reste 0) donc 18 = b....010

$2/2 = 1$(reste 0) donc 18 = b...0010

$1/2 = 0$(reste 1) donc 18 = b10010

Réciproquement : on divise successivement par deux et on écrit les restes obtenus de droite à gauche. On s'arrête quand le quotient est nul.

Exemple : convertir 18 en base 2

$18/2 = 9$(reste 0) donc 18 = b.....0

$9/2 = 4$(reste 1) donc 18 = b....10

$4/2 = 2$(reste 0) donc 18 = b....010

$2/2 = 1$(reste 0) donc 18 = b...0010

$1/2 = 0$(reste 1) donc 18 = b10010

Vérification : $2^1 + 2^4 = 2 + 16 = 18$

Taille des données

bit, octet, mot

Le terme "**bit**" signifie "Binary Information uniT". En fonction du contexte cette information peut être codée par "Vrai/Faux" ("True"/"False") ou par "1"/"0".

Taille des données
bit, octet, mot

Le terme "**bit**" signifie "Binary Information uniT". En fonction du contexte cette information peut être codée par "Vrai/Faux" ("True"/"False") ou par "1"/"0".
Un **octet** est un ensemble de huit bits.

Taille des données

bit, octet, mot

Le terme "**bit**" signifie "Binary Information uniT". En fonction du contexte cette information peut être codée par "Vrai/Faux" ("True"/"False") ou par "1"/"0".
Un **octet** est un ensemble de huit bits.
Plus généralement un **mot** est un ensemble de bits de taille déterminée (usuellement 8,16,32 ou 64 bits).

On peut appliquer des opérations sur des mots :
Non(11001)=

On peut appliquer des opérations sur des mots :
Non(11001)= 00110
101 ET 110=

On peut appliquer des opérations sur des mots :
Non(11001)= 00110
101 ET 110=100
101 OU 110=

On peut appliquer des opérations sur des mots :
Non(11001)= 00110
101 ET 110=100
101 OU 110=111
$\rightarrow$ (11001)=

On peut appliquer des opérations sur des mots :
Non(11001)= 00110
101 ET 110=100
101 OU 110=111
$\rightarrow$ (11001)=1100
$\leftarrow$ (11001)=

On peut appliquer des opérations sur des mots :
Non(11001)= 00110
101 ET 110=100
101 OU 110=111
$\rightarrow$ (11001)=1100
$\leftarrow$ (11001)=110010

On peut appliquer des opérations sur des mots :
Non(11001)= 00110
101 ET 110=100
101 OU 110=111
$\rightarrow$ (11001)=1100
$\leftarrow$ (11001)=110010
L'opération "décalage à gauche" a une particularité. Laquelle ?

Elles facilitent la compréhension humaine, notamment des développeurs de programmes. Elles ont notamment la particularité d'être des puissances de 2 ($8 = 2^3$, $16 = 2^4$) et par conséquent, d'être plus simplement converties avec une représentation binaire.

Elles facilitent la compréhension humaine, notamment des développeurs de programmes. Elles ont notamment la particularité d'être des puissances de 2 ($8 = 2^3$, $16 = 2^4$) et par conséquent, d'être plus simplement converties avec une représentation binaire.
Exemple de conversion base 2 en base 16 :
h11000011 = h 1100 0011 = h B 3

En informatique, ces dimensions sont des puissances de 2. il est d'usage de noter ces dimensions comme suit :

- 1 b (b) = 1 bit

En informatique, ces dimensions sont des puissances de 2. il est d'usage de noter ces dimensions comme suit :

- 1 b (b) = 1 bit
- 1 o (B) = 1 Byte = 8 bits

En informatique, ces dimensions sont des puissances de 2. il est d'usage de noter ces dimensions comme suit :

- 1 b (b) = 1 bit
- 1 o (B) = 1 Byte = 8 bits
- 1 Kio (KiB) = 1024 octets (2^{10})

En informatique, ces dimensions sont des puissances de 2. il est d'usage de noter ces dimensions comme suit :

- 1 b (b) = 1 bit
- 1 o (B) = 1 Byte = 8 bits
- 1 Kio (KiB) = 1024 octets (2^{10})
- 1 Mio (MiB) = 1024^2 octets (2^{20})

En informatique, ces dimensions sont des puissances de 2. il est d'usage de noter ces dimensions comme suit :

- 1 b (b) = 1 bit
- 1 o (B) = 1 Byte = 8 bits
- 1 Kio (KiB) = 1024 octets (2^{10})
- 1 Mio (MiB) = 1024^2 octets (2^{20})
- 1 Gio (GiB) = 1024^3 octets (2^{30})

En informatique, ces dimensions sont des puissances de 2. il est d'usage de noter ces dimensions comme suit :

- 1 b (b) = 1 bit
- 1 o (B) = 1 Byte = 8 bits
- 1 Kio (KiB) = 1024 octets (2^{10})
- 1 Mio (MiB) = 1024^2 octets (2^{20})
- 1 Gio (GiB) = 1024^3 octets (2^{30})
- 1 Gib (Gib) = 1024^3 bits = 1024^3 / 8 B = 2^{27} (octets)

La confusion entre normes de grandeurs binaires et décimales entraîne des erreurs fondamentales notamment dans les grandes dimensions.

La confusion entre normes de grandeurs binaires et décimales entraîne des erreurs fondamentales notamment dans les grandes dimensions.
Les débits sont le plus souvent exprimés en b/s (bits/s) tandis que les espaces de mémoire ou de stockage (disques durs) sont mesurés en octets.

La confusion entre normes de grandeurs binaires et décimales entraîne des erreurs fondamentales notamment dans les grandes dimensions.
Les débits sont le plus souvent exprimés en b/s (bits/s) tandis que les espaces de mémoire ou de stockage (disques durs) sont mesurés en octets.
La taille mémoire est le plus souvent exprimée en Gio, les espaces disque en Tio...

La confusion entre normes de grandeurs binaires et décimales entraîne des erreurs fondamentales notamment dans les grandes dimensions.
Les débits sont le plus souvent exprimés en b/s (bits/s) tandis que les espaces de mémoire ou de stockage (disques durs) sont mesurés en octets.
La taille mémoire est le plus souvent exprimée en Gio, les espaces disque en Tio...
Exemple Calculer le temps de nécessaire au téléversement de 10 images non compressées (3 octets par couleur) de résolution full-hd depuis une connexion adsl 512kib/s (512kib/s descendant, 128kib/s montant) ?

Ordinateur (à programme enregistré)

Première approche

L'ordinateur « à programme enregistré » permet l'exécution successive d'opérations ou « instructions » provenant de l'extérieur (c.a.d inconnues lors de la conception de cet ordinateur).

Ordinateur (à programme enregistré)

Première approche

L'ordinateur « à programme enregistré » permet l'exécution successive d'opérations ou « instructions » provenant de l'extérieur (c.a.d inconnues lors de la conception de cet ordinateur).
Ces instructions aujourd'hui exécutées par le « microprocesseur » nécessitent le plus souvent la conservation au moins temporaire de données.

Exemple : Calculer $3^2 + 4^2$

Ordinateur (à programme enregistré)

Première approche

L'ordinateur « à programme enregistré » permet l'exécution successive d'opérations ou « instructions » provenant de l'extérieur (c.a.d inconnues lors de la conception de cet ordinateur).
Ces instructions aujourd'hui exécutées par le « microprocesseur » nécessitent le plus souvent la conservation au moins temporaire de données.
Exemple : Calculer $3^2 + 4^2$
Les registres du microprocesseur et la « mémoire vive » sont ainsi des éléments clé de l'ordinateur.
L'ordinateur est aussi doté, dès ses débuts, de dispositifs d'entrée/sortie pour communiquer avec l'extérieur et en premier lieu permettre la lecture de programmes (bandes perforées, clé usb ...).
Cf : Les architectures Harvard et Von Neumann

- Une mémoire « adressable » : les éléments situés dans cette mémoire peuvent être retrouvés grâce à leur adresse tout comme une personne peut être trouvée grâce à l'adresse de son domicile.

- Une mémoire « adressable » : les éléments situés dans cette mémoire peuvent être retrouvés grâce à leur adresse tout comme une personne peut être trouvée grâce à l'adresse de son domicile.
- Une unité de contrôle équipée d'une horloge : elle produit des impulsions électriques régulières entraînant la lecture « séquentielle » des instructions situées en mémoire.

- Une mémoire « adressable » : les éléments situés dans cette mémoire peuvent être retrouvés grâce à leur adresse tout comme une personne peut être trouvée grâce à l'adresse de son domicile.
- Une unité de contrôle équipée d'une horloge : elle produit des impulsions électriques régulières entraînant la lecture « séquentielle » des instructions situées en mémoire.
- Une unité arithmétique et logique : elle est chargée de l'exécution de chaque instruction et dispose de zones de mémoire internes (registres, accumulateurs, piles). Cette mémoire « temporaire » autorise notamment l'enchaînement « séquentiel » des instructions.

- Une mémoire « adressable » : les éléments situés dans cette mémoire peuvent être retrouvés grâce à leur adresse tout comme une personne peut être trouvée grâce à l'adresse de son domicile.
- Une unité de contrôle équipée d'une horloge : elle produit des impulsions électriques régulières entraînant la lecture « séquentielle » des instructions situées en mémoire.
- Une unité arithmétique et logique : elle est chargée de l'exécution de chaque instruction et dispose de zones de mémoire internes (registres, accumulateurs, piles). Cette mémoire « temporaire » autorise notamment l'enchaînement « séquentiel » des instructions.
- Des « contrôleurs » d'entrée/sortie : ils permettent les échanges avec l'extérieur. Ils peuvent notamment interrompre l'enchaînement séquentiel et détourner le processeur vers une autre action.

Le langage « machine » est le langage numérique utilisé par le microprocesseur. Il représente une boîte à « outils numérotés ». Chaque outil peut posséder ses propres paramètres et peut dépendre de l'utilisation d'autres outils.

Le langage « machine » est le langage numérique utilisé par le microprocesseur. Il représente une boîte à « outils numérotés ». Chaque outil peut posséder ses propres paramètres et peut dépendre de l'utilisation d'autres outils.
L'assembleur est une représentation humaine du langage machine. L'assembleur doit donc être traduit en langage machine avant d'être compris par l'ordinateur. Il peut aussi être déduit du langage machine dans certaines limites. Les langages « évolués » les plus optimisés s'appuient souvent sur l'assembleur.

Le langage « machine » est le langage numérique utilisé par le microprocesseur. Il représente une boîte à « outils numérotés ». Chaque outil peut posséder ses propres paramètres et peut dépendre de l'utilisation d'autres outils.
L'assembleur est une représentation humaine du langage machine. L'assembleur doit donc être traduit en langage machine avant d'être compris par l'ordinateur. Il peut aussi être déduit du langage machine dans certaines limites. Les langages « évolués » les plus optimisés s'appuient souvent sur l'assembleur.
Il existe une multitude d'architectures et par conséquent, une multitude de jeux d'instructions.

Conclusion

L'augmentation des fréquences d'horloge et des capacités d'intégration sont des moteurs dans l'essor du numérique mais les relations et transcriptions homme-machine sont devenues complexes. Elles s'appuient sur l'utilisation de langages de programmation plus "évolués" ainsi que sur une représentation fidèle et efficace de l'information.