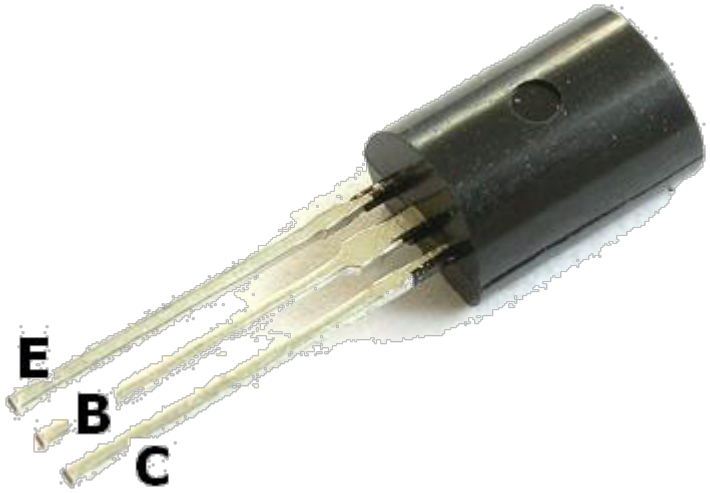
Architecture des ordinateurs
et
Systèmes d'exploitation

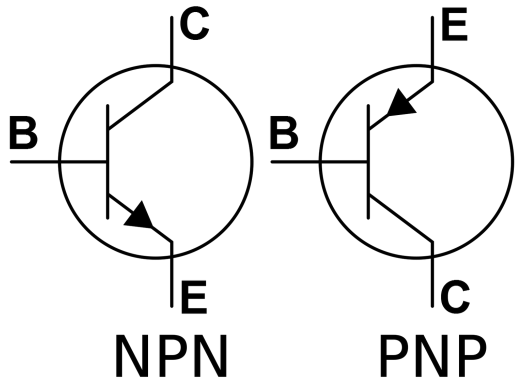
Raymond Namyst
Université de Bordeaux

Au niveau électronique

Des transistors… beaucoup de transistors

- Transistor = télérupteur miniature
  - Sorte de robinet de courant électrique
    - On peut commander le passage du courant





Représentation de l'information

Des histoires de 0 et de 1

- Au niveau électrique, on distingue deux états
  - Courant qui passe / courant qui ne passe pas

- Représentation *binaire* de l'information
  - Deux chiffres seulement: 0 et 1
    - 0 : absence de courant
    - 1 : présence de courant
  - BIT = *Binary Digit*

- Un ordinateur est capable
  - De stocker une grande quantité de 0 et de 1
    - Dont certains sont reçus depuis l'extérieur

      CDROM, clé USB, clavier, etc.
  - D'effectuer très rapidement des calculs sur des ensembles de bits

Représentation de l'information

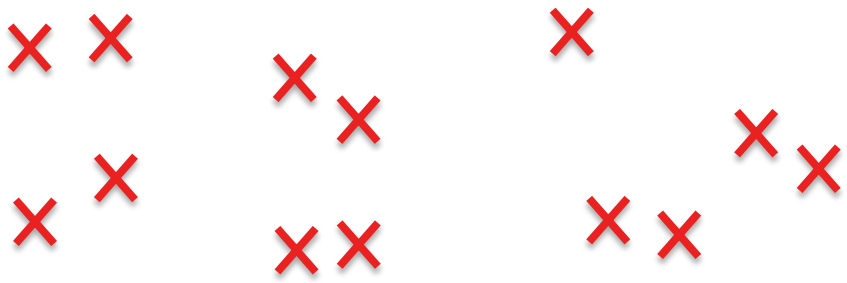
Du binaire au codage en base 2

- Pour manipuler des objets plus complexes, les ordinateurs *regroupent* les BITs par paquets indivisibles
  - Octet = paquet de 8 BITs
  - Mot = ensemble de plusieurs octets
    - Exemple : « un ordinateur 32 bits » est un ordinateur qui manipule des mots de 4 octets (4x8 = 32)
    - Aujourd'hui, la mode est plutôt aux ordinateurs « 64 bits »

- Représentation des entiers positifs
  - 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, etc.
  - Exemple : le nombre décimal 13 est « codé » 1101 en binaire (donc 00000000 00000000 00000000 00001101 sur une machine 32 bits)

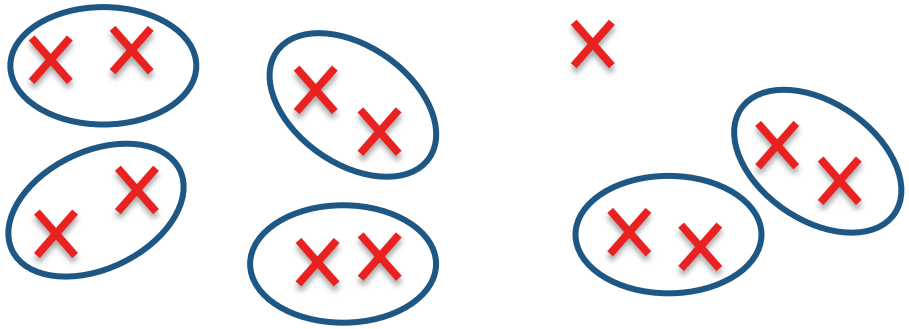
Représentation de l'information

À une époque, à l'école primaire…



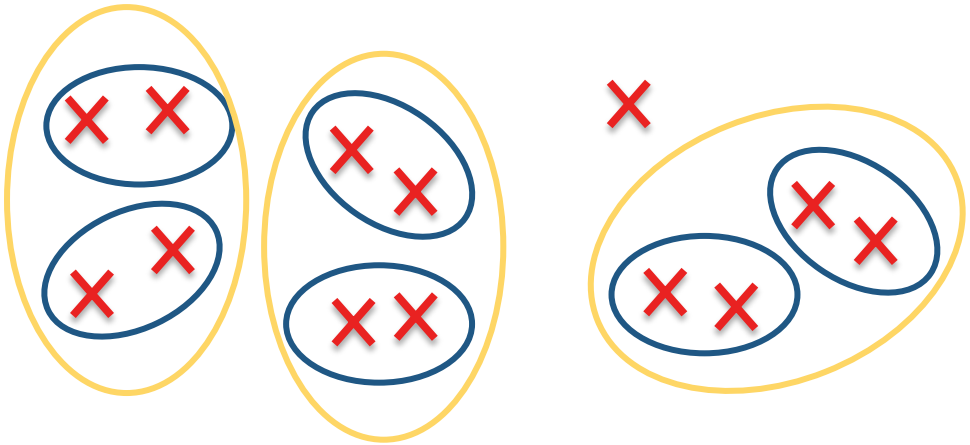
Représentation de l'information

À une époque, à l'école primaire…



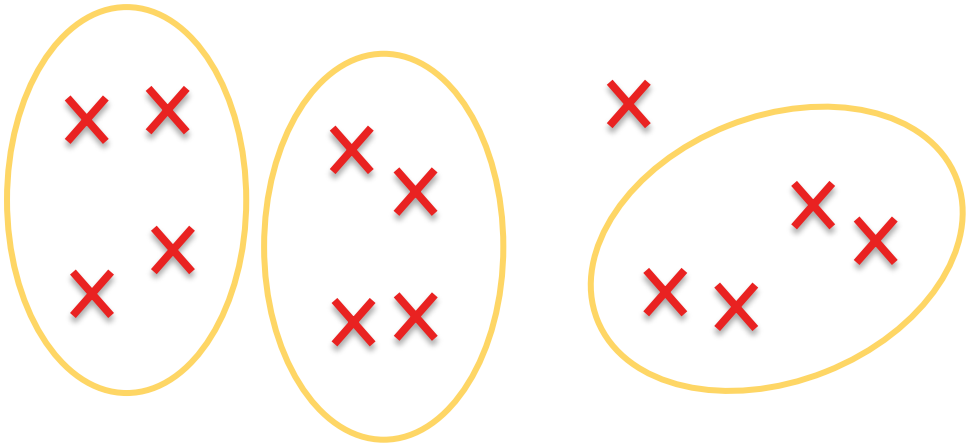
Représentation de l'information

À une époque, à l'école primaire…



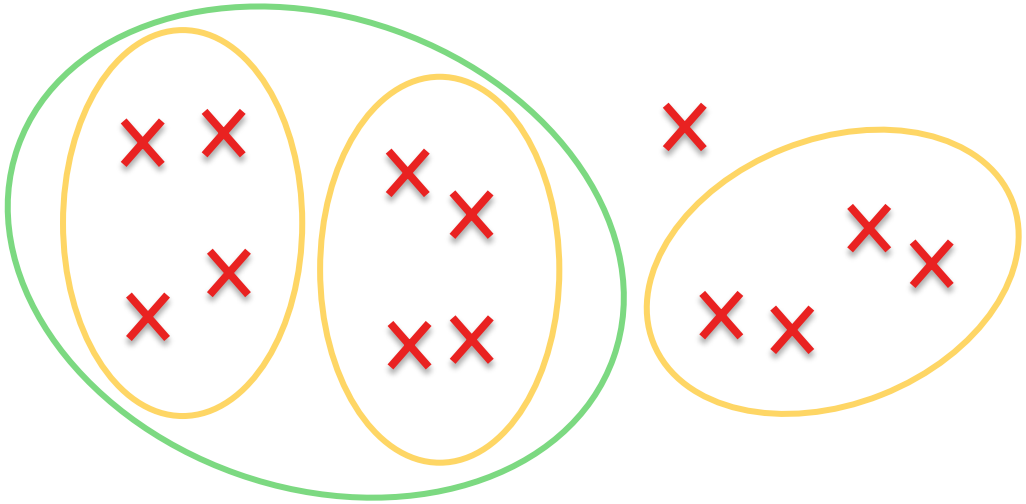
Représentation de l'information

À une époque, à l'école primaire…



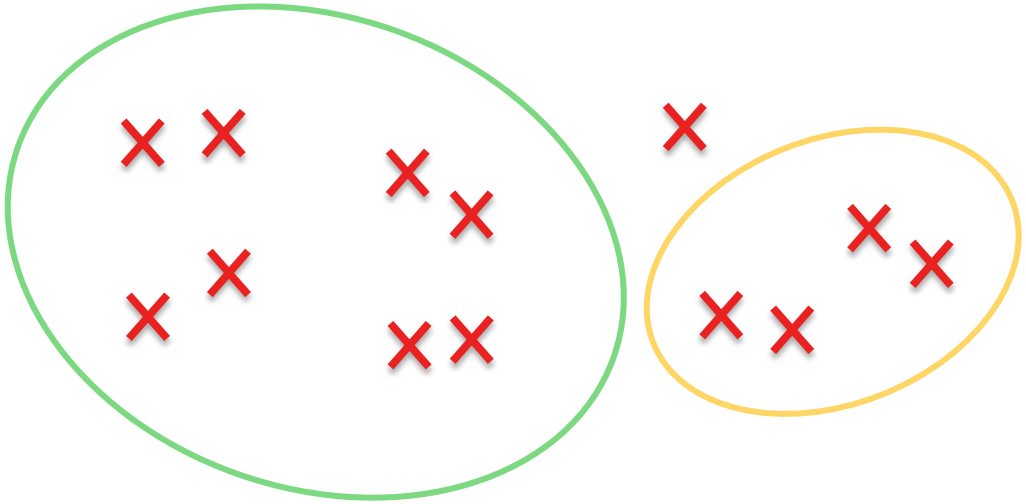
Représentation de l'information

À une époque, à l'école primaire…



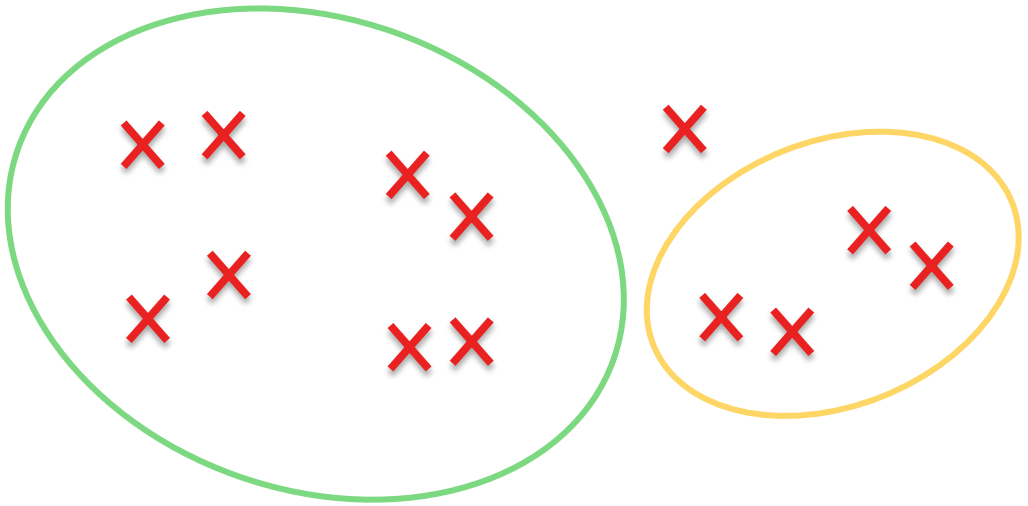
Représentation de l'information

À une époque, à l'école primaire…



Représentation de l'information

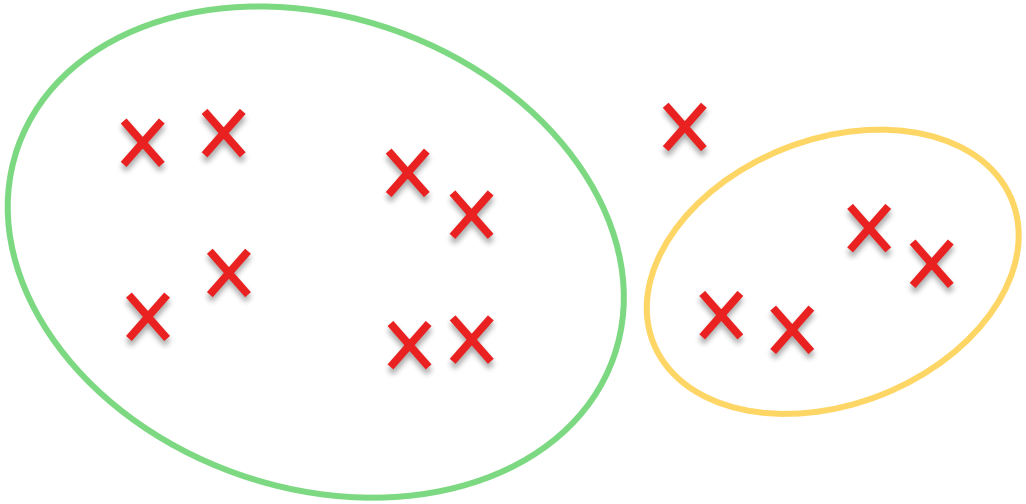
À une époque, à l'école primaire…



1101

Représentation de l'information

À une époque, à l'école primaire…



```
  1101
= 1000   (8)
+  100   (4)
+    1   (1)
```

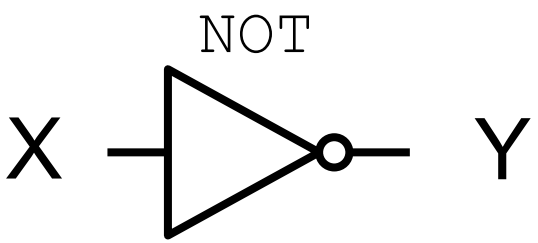
Principe de fonctionnement d'un ordinateur : les circuits élémentaires

Au niveau électronique

Des transistors… aux portes logiques

- En combinant plusieurs transistors, on peut former des circuits logiques élémentaires : les *portes* logiques

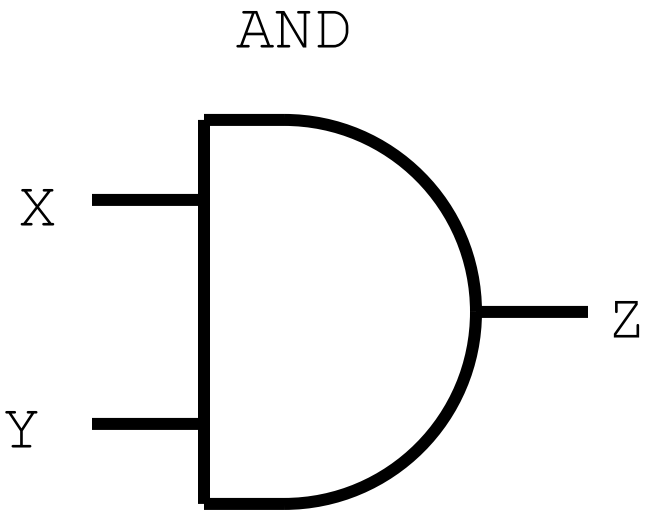
- Exemple : la porte « NON »
  - Inverseur



X	Y
0	1
1	0

Au niveau électronique (suite)

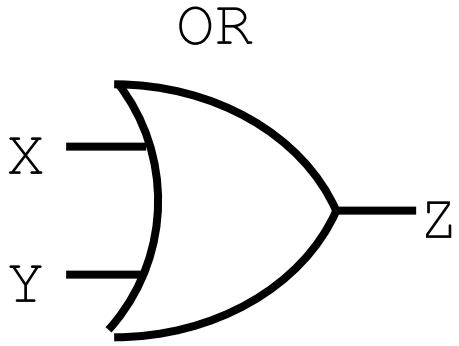
- La porte « ET »
  - La sortie vaut 1 *si et seulement si* les deux entrées valent 1



X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Au niveau électronique (suite)

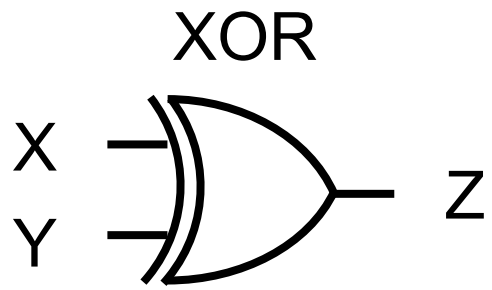
- La porte « OU »
  - La sortie vaut 1 si l'une des deux entrées vaut 1



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Au niveau électronique (suite)

- La porte « OU EXCLUSIF »
  - La sortie vaut 1 si et seulement si uniquement l'une des deux entrées vaut 1

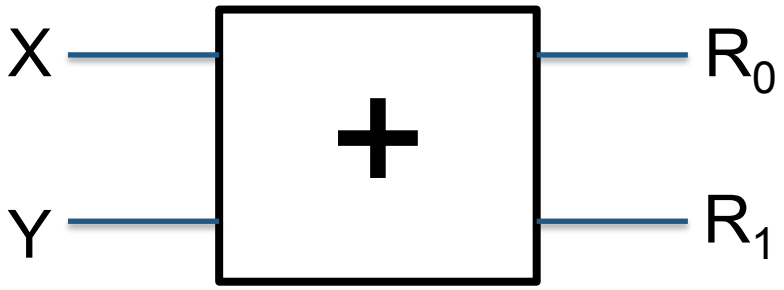


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Calculer en binaire

- En construisant des assemblages de portes logiques, on peut imaginer toutes sortes de circuits !

- Par exemple, un additionneur « 1 bit »
  - On veut un circuit à deux entrées (X et Y) et à deux sorties (les deux bits du résultat)

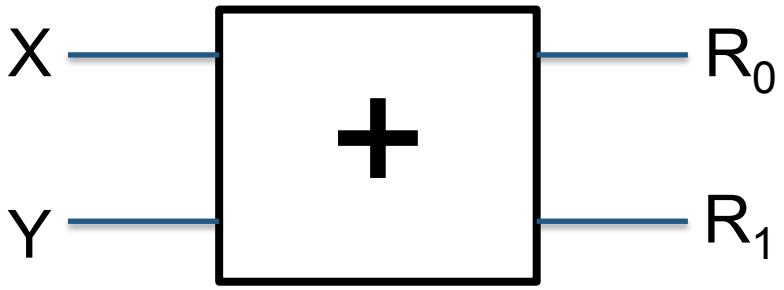


X	Y	R_1	R_0
0	0	0	0
0	1		
1	0		
1	1		

Calculer en binaire

- En construisant des assemblages de portes logiques, on peut imaginer toutes sortes de circuits !

- Par exemple, un additionneur de 2 bits X et Y
  - On veut un circuit à deux entrées (X et Y) et à deux sorties (les deux bits du résultat)

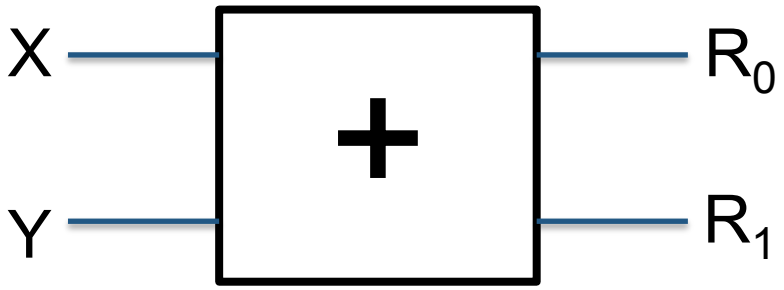


X	Y	R_1	R_0
0	0	0	0
0	1	0	1
1	0		
1	1		

Calculer en binaire

- En construisant des assemblages de portes logiques, on peut imaginer toutes sortes de circuits !

- Par exemple, un additionneur « 1 bit »
  - On veut un circuit à deux entrées (X et Y) et à deux sorties (les deux bits du résultat)

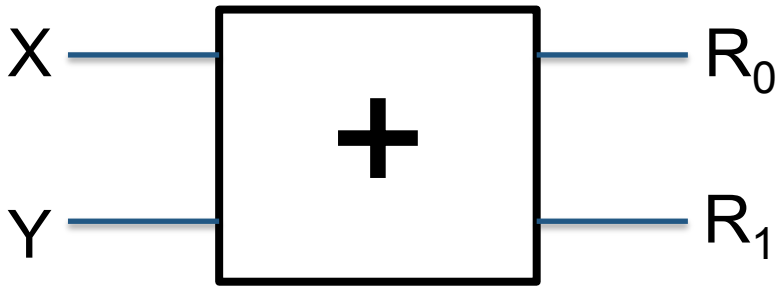


X	Y	R_1	R_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1		

Calculer en binaire

- En construisant des assemblages de portes logiques, on peut imaginer toutes sortes de circuits !

- Par exemple, un additionneur « 1 bit »
  - On veut un circuit à deux entrées (X et Y) et à deux sorties (les deux bits du résultat)



X	Y	R_1	R_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

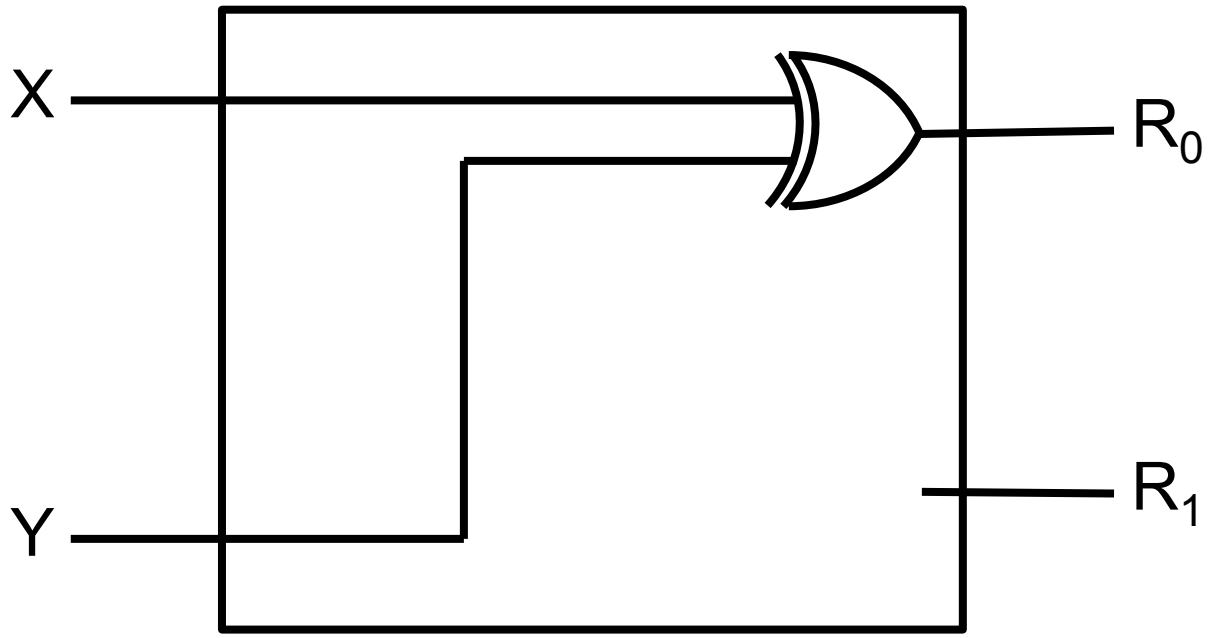
Calculer en binaire

- Un additionneur « 1 bit » = ?

X	Y	R_1	R_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Calculer en binaire

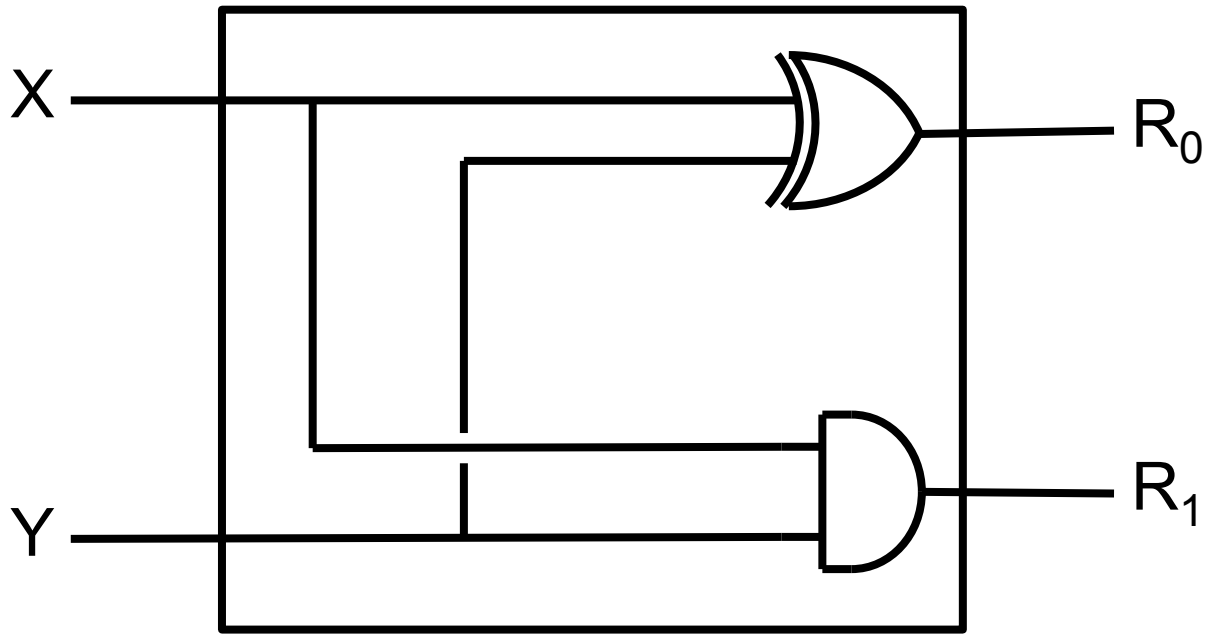
- Un additionneur « 1 bit » = une porte OU EXCLUSIF + ?



X	Y	R_1	R_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Calculer en binaire

- Un additionneur « 1 bit » = une porte OU EXCLUSIF + une porte ET



X	Y	R_1	R_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

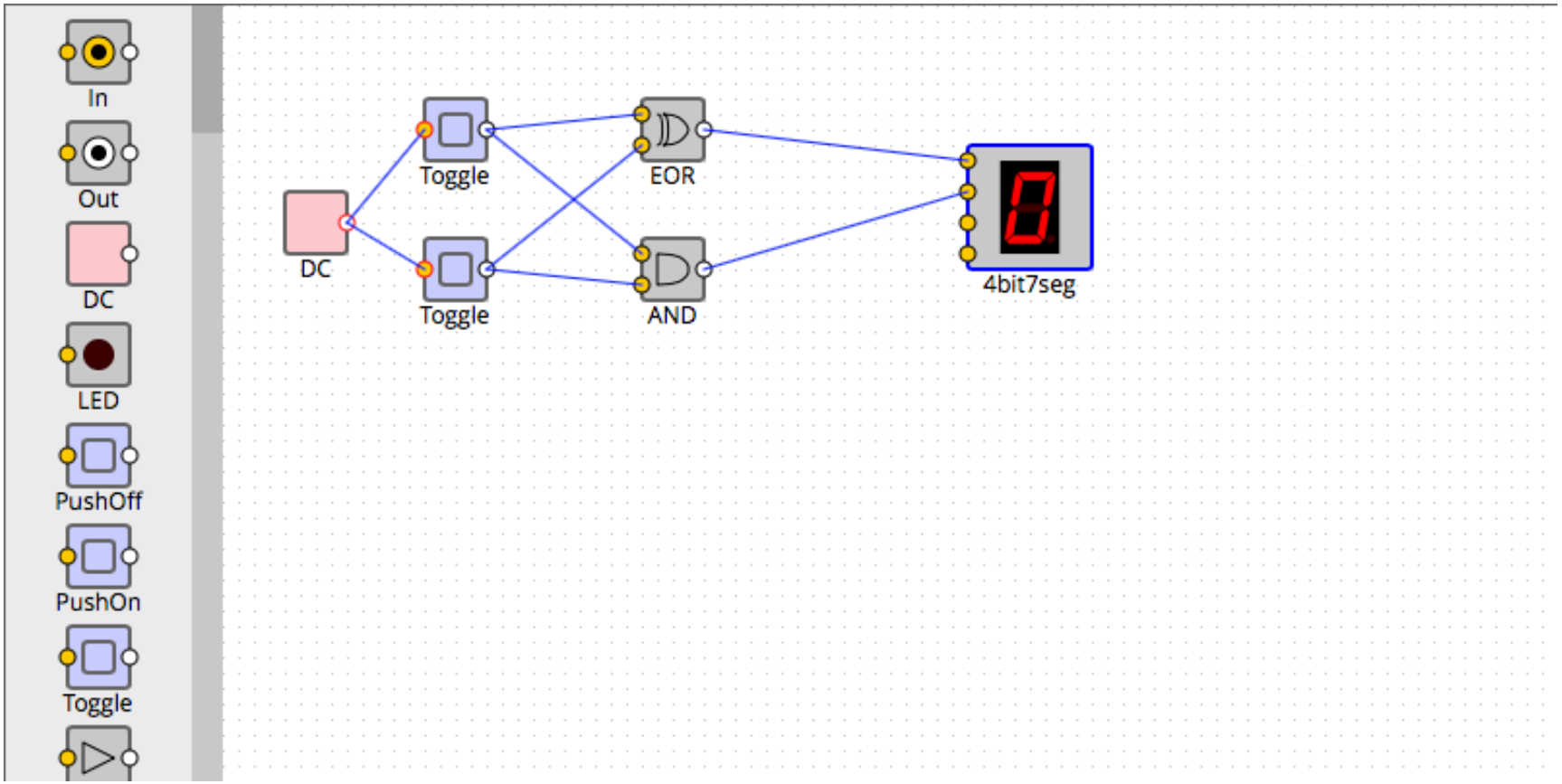
Portes, circuits… circuits plus complexes

On joue aux LEGO !

- Par assemblage de portes logiques et de circuits, on parvient à construire des circuits de plus en plus complexes
  - Unités Arithmétiques et Logiques
    - Addition
    - Soustraction
    - Multiplication
    - Division
    - Etc.
  - Multiplexeurs, démultiplexeurs
    - a*iguillage* de l'information
  - Microprocesseur

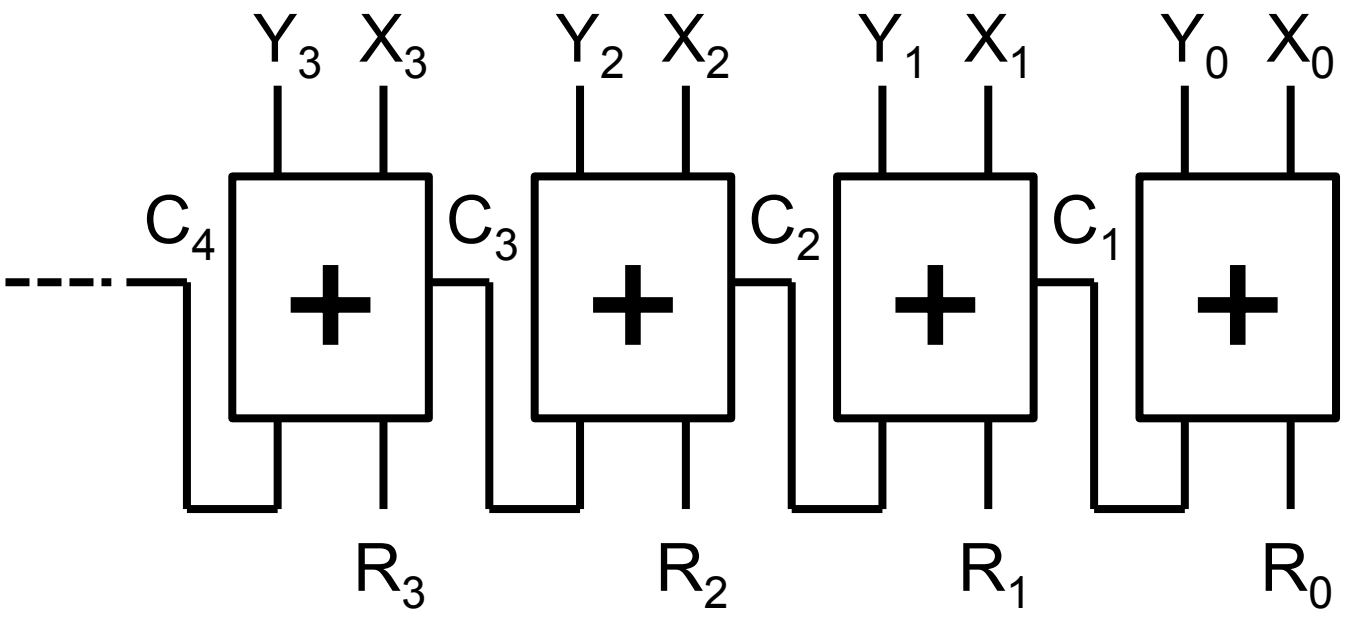
Un simulateur de circuits très pratique

simcirJS (https://kazuhikoarase.github.io/simcirjs/)



Calculer en binaire

- Pour additionner des nombres de plusieurs bits, il faut cascader des additionneurs de 3 bits
  - Exemple : addition de $X_3X_2X_1X_0$ et de $Y_3Y_2Y_1Y_0$



Calculer en binaire

- Et les entiers négatifs alors ?
  - Les entiers négatifs sont codés de manière à ce que l'addition telle que vue précédemment fonctionne encore
    - 3 + -3 doit donner 0
- De manière intuitive, on cherche comment on pourrait coder -1 de telle façon que l'addition avec 1 donne 0. Exemple sur 4 bits :

```
  0001          1
+ ????        + -1
--------      -----
= 0000        =  0
```

Réponse : -1 doit être codé 1111

Calculer en binaire

Représentation des entiers relatifs

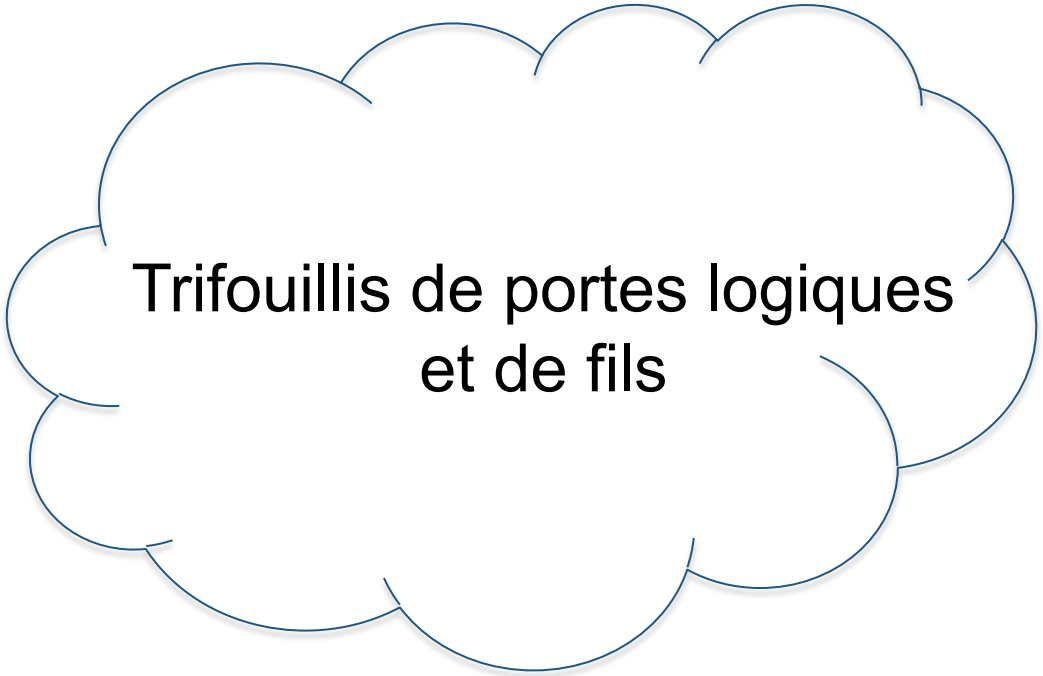
- De manière plus formelle, le bit le plus à gauche possède un poids « négatif »
  - Un nombre X représenté $X_nX_{n-1}…X_1X_0$ a pour valeur :

    $$-X_n.2^n + X_{n-1}.2^{n-1} + ... + X_1.2^1 + X_0.2^0$$

  - Par exemple :
    - -8 s'écrit 1000 (soit -2^3)
    - -7 s'écrit 1001 (soit $-2^3 + 2^0 = -8 + 1$)
    - -1 s'écrit 1111 (soit $-2^3 + 2^2 + 2^1 + 2^0 = -8 + 7$)
- L'ordinateur effectue des additions sans savoir si un entier est codé de manière relative ou pas : ça ne fait aucune différence pour lui !
  - Sur 4 bits, on peut manipuler des entiers positifs entre 0 et 15
  - Ou des entiers relatifs entre -8 et 7 !
  - C'est uniquement le langage de programmation qui interprête le codage lors de l'affichage des nombres, par exemple...

Principe de fonctionnement d'un ordinateur : le microprocesseur

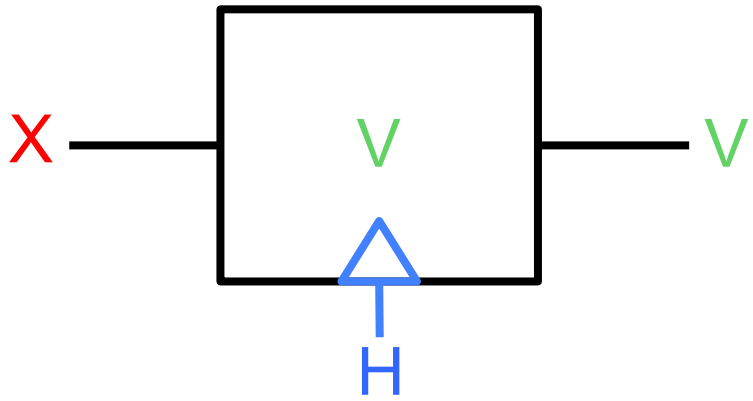
Plan de cablage détaillé d'un microprocesseur

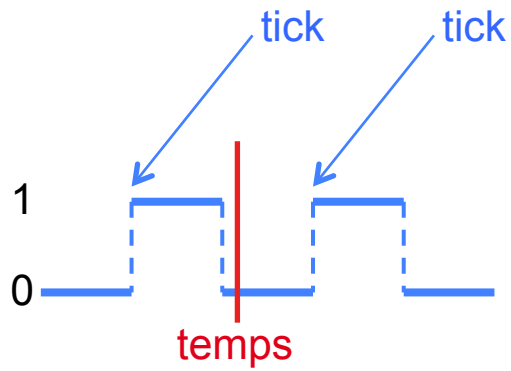


Circuits « mémoire »

Vers un fonctionnement synchrone

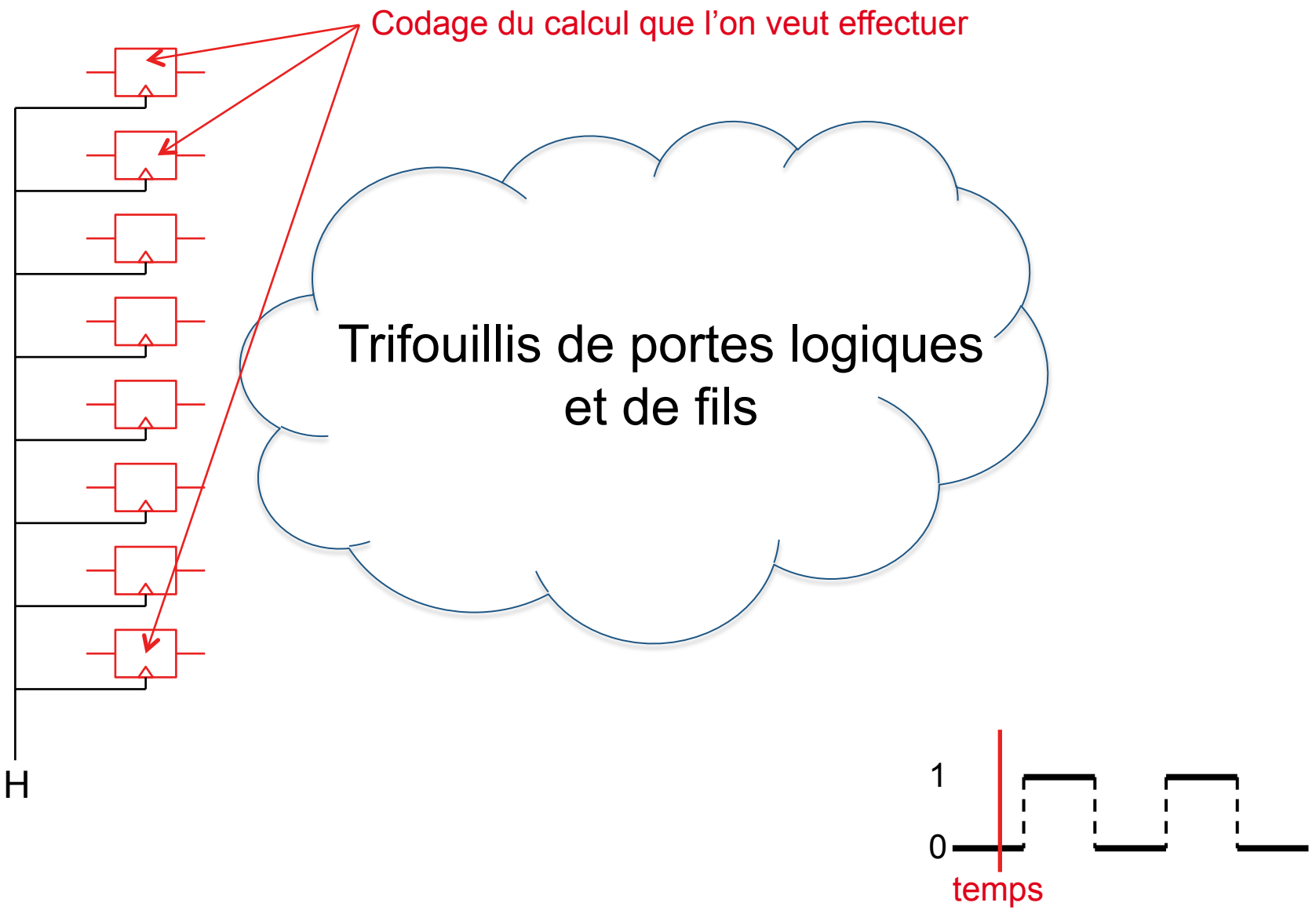
- Circuits « mémoires » qui ne changent d'état que lors d'un « tick d'horloge »





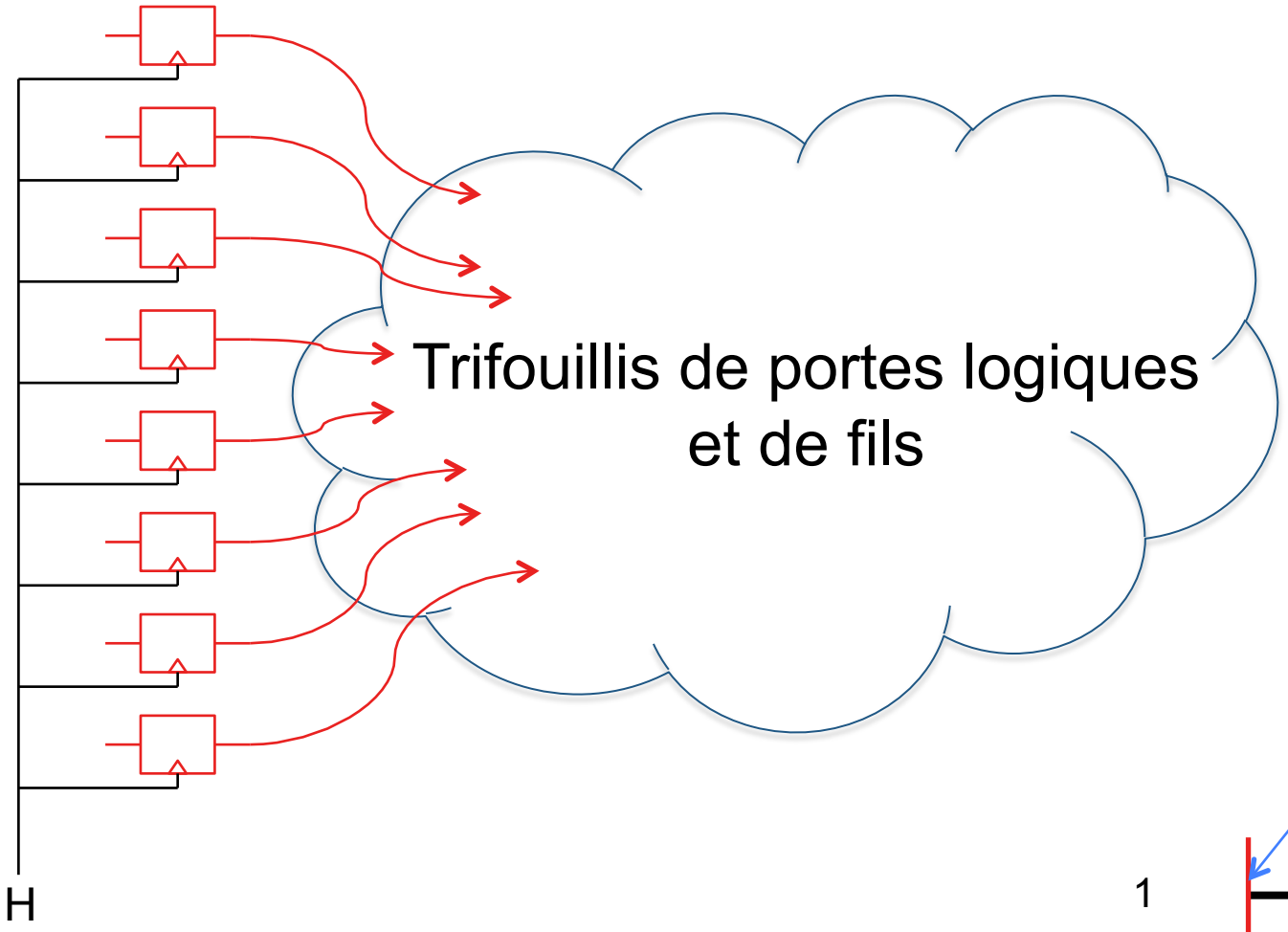
Propagation d'un « calcul » dans le processeur

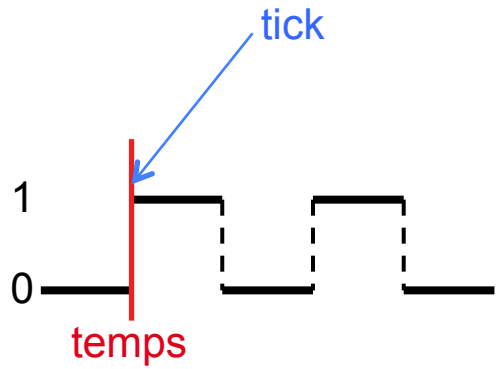
Notion de cycle



Propagation d'un « calcul » dans le processeur

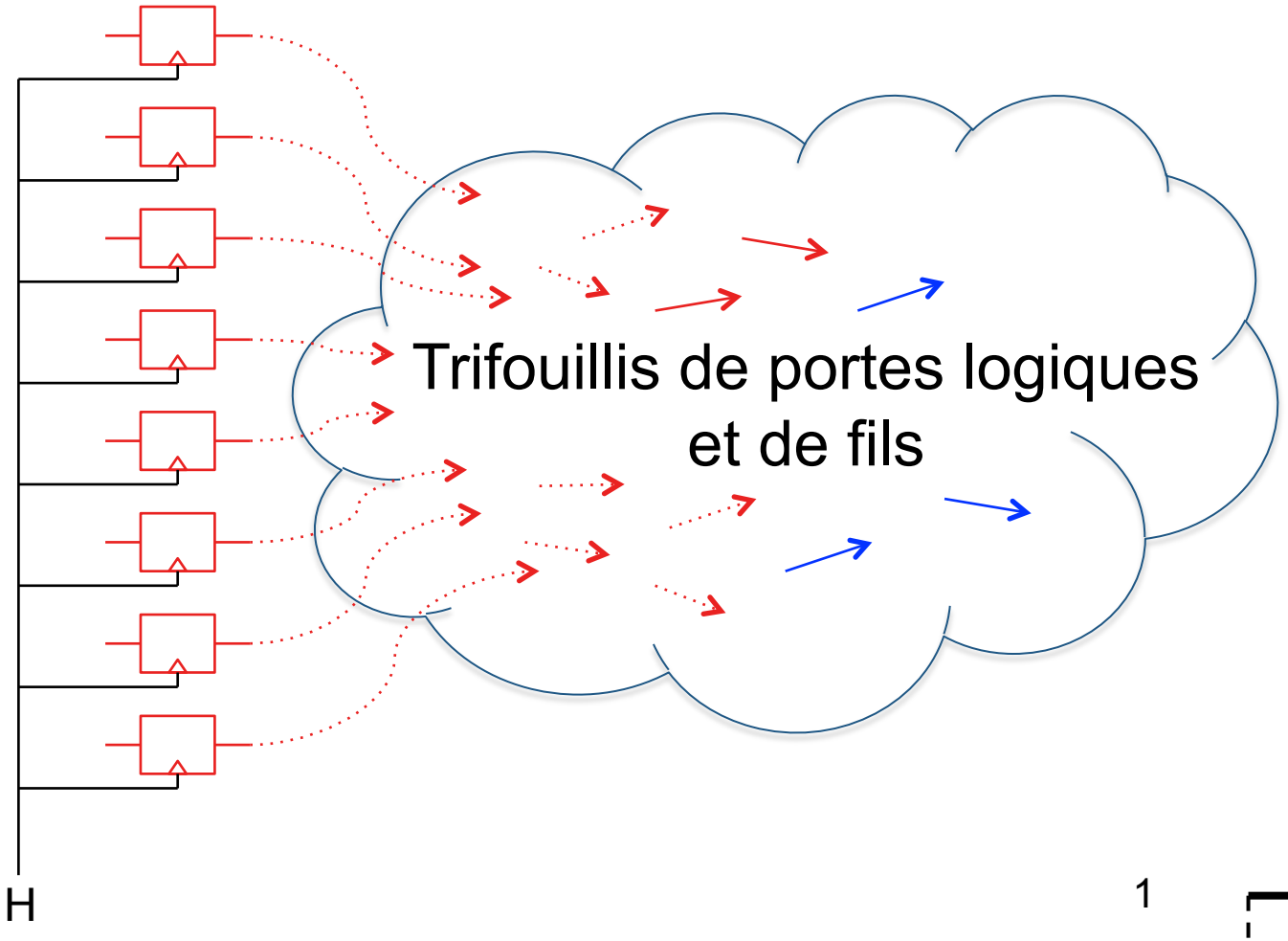
Notion de cycle

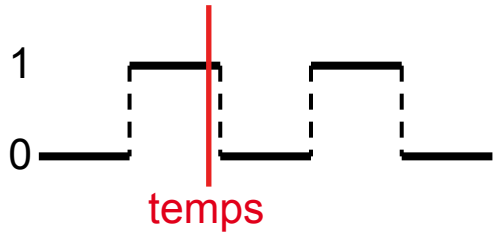




Propagation d'un « calcul » dans le processeur

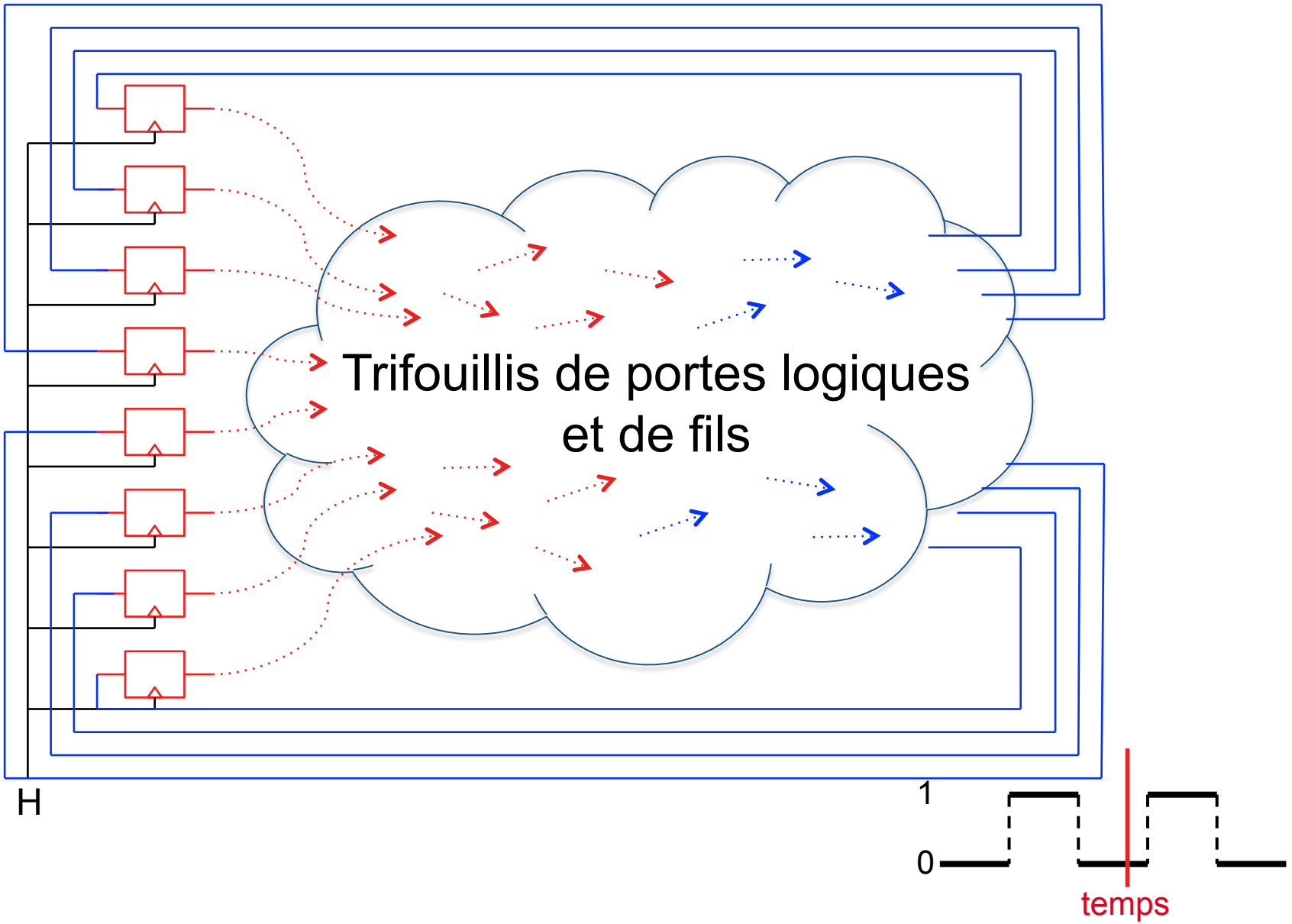
Notion de cycle





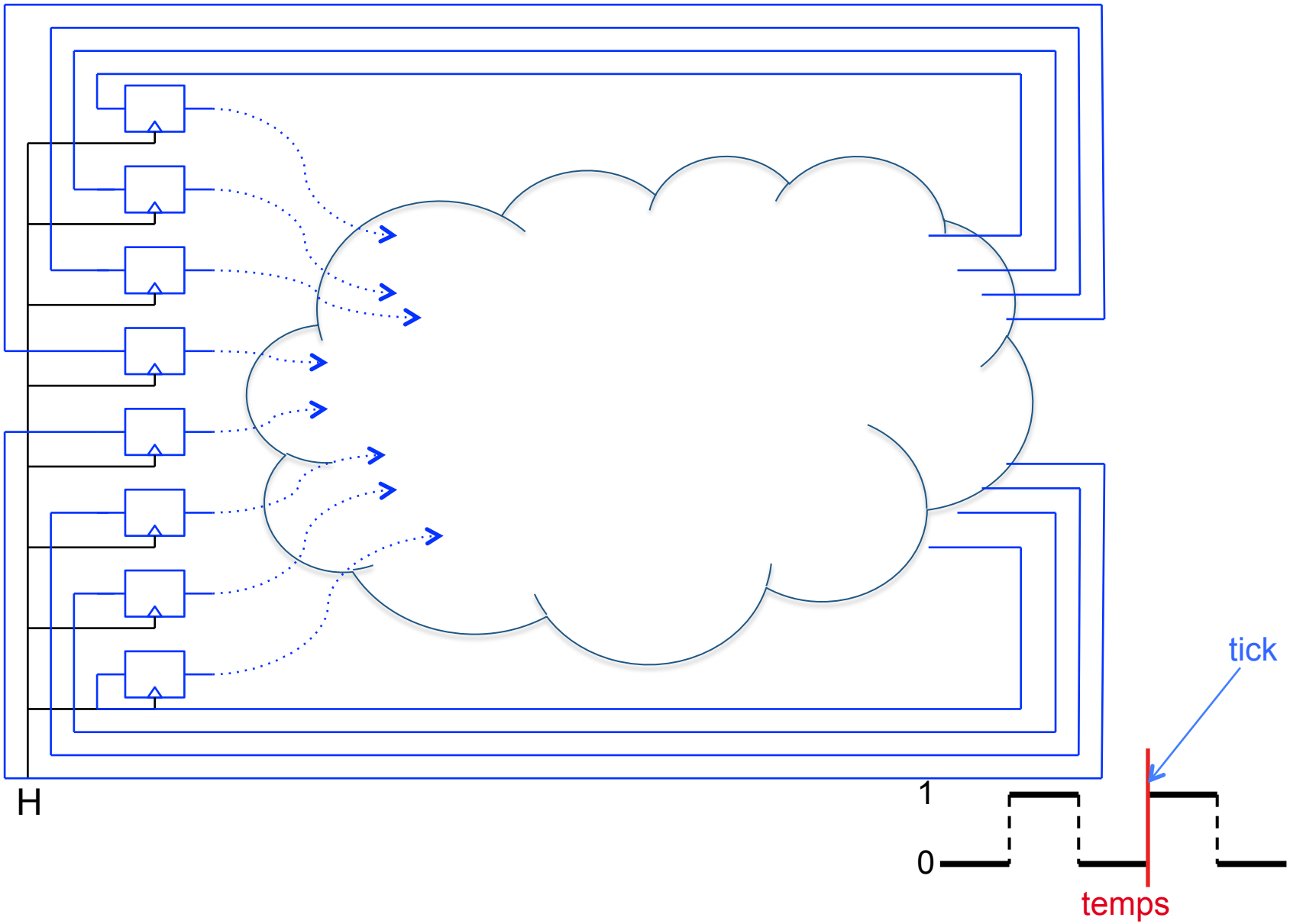
Propagation d'un « calcul » dans le processeur

Il faut que toutes les tensions soient stabilisées



Propagation d'un « calcul » dans le processeur

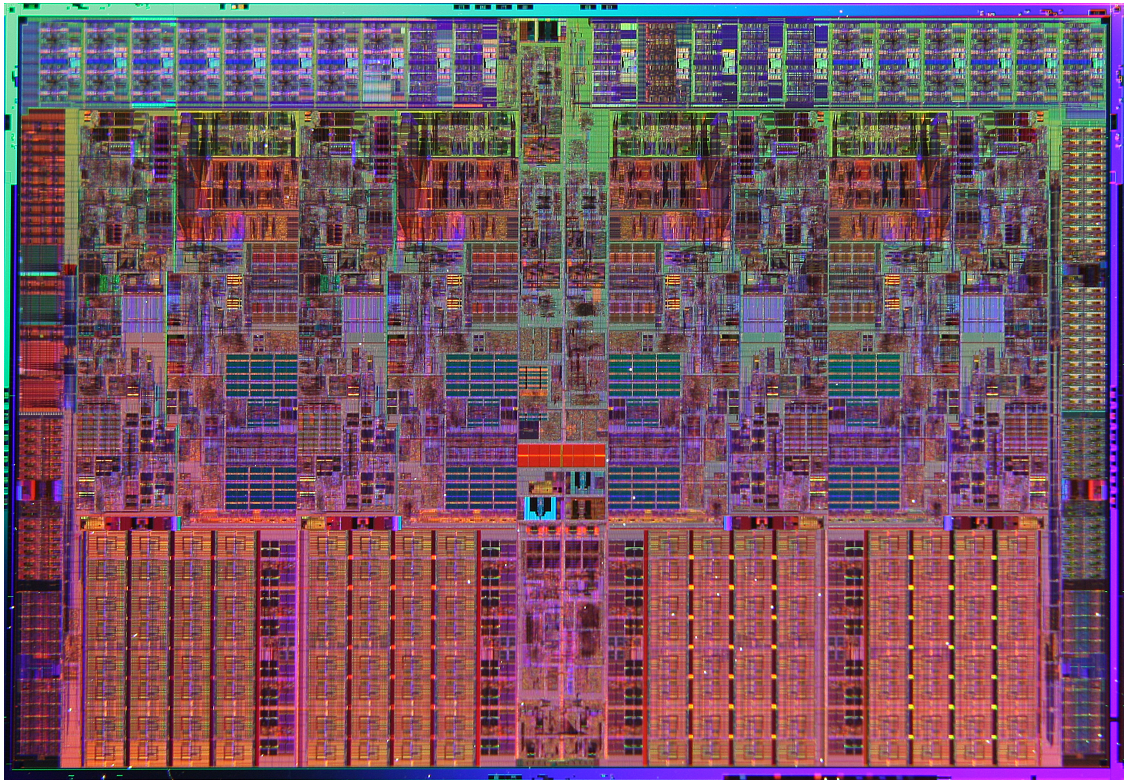
On démarre un nouveau calcul !



Au niveau électronique (suite)

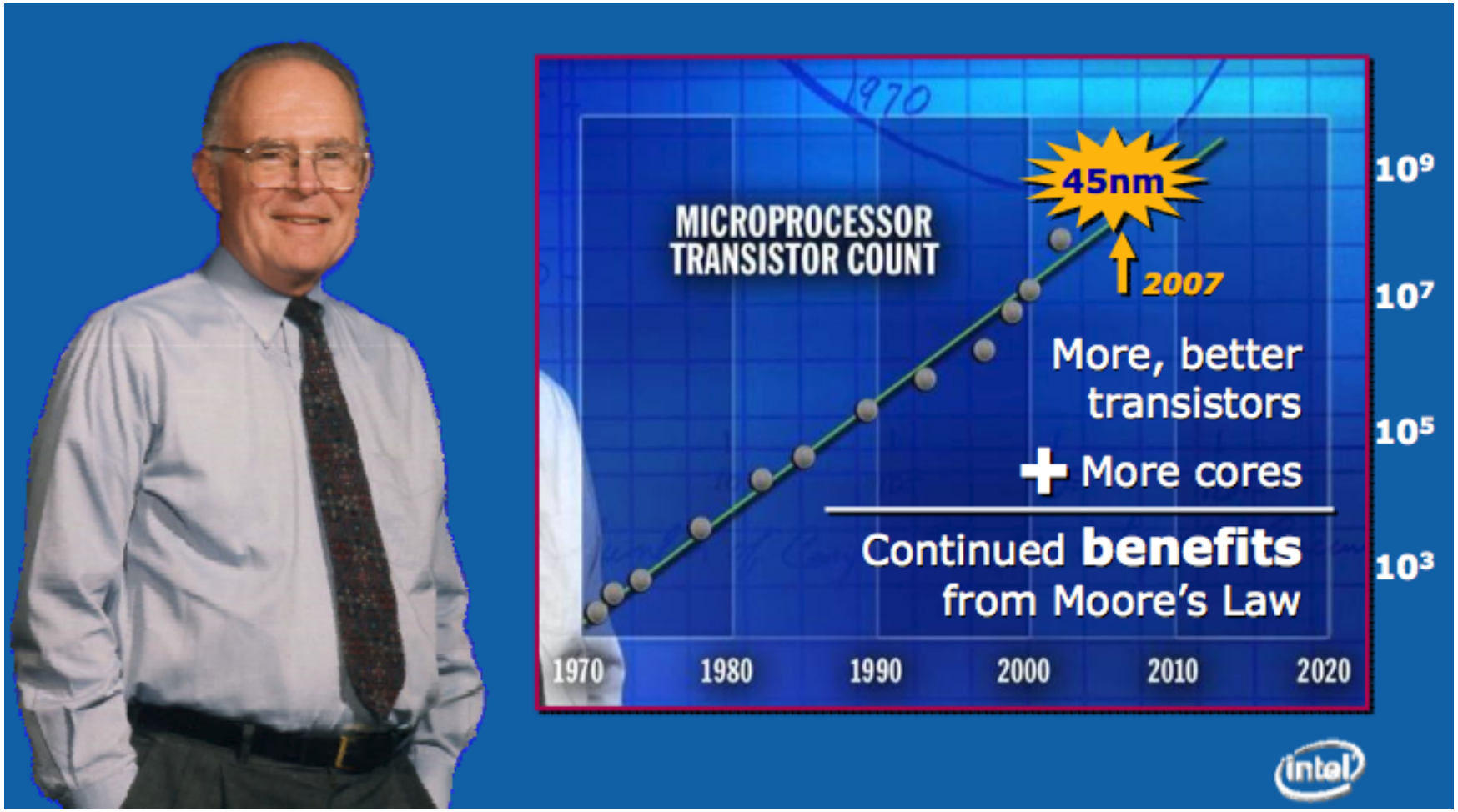
En définitive

- Aujourd'hui, les microprocesseurs sont constitués de plusieurs milliards de transistors
  - Plus personne n'est capable d'examiner leur architecture en détail ;)
    - Ça n'est pas grave puisqu'on procède par assemblage de boîtes !



Un prophète

Gordon Moore, co-fondateur d'Intel



« Le nombre de transistors par unité de surface double tous les deux ans »

Loi de moore

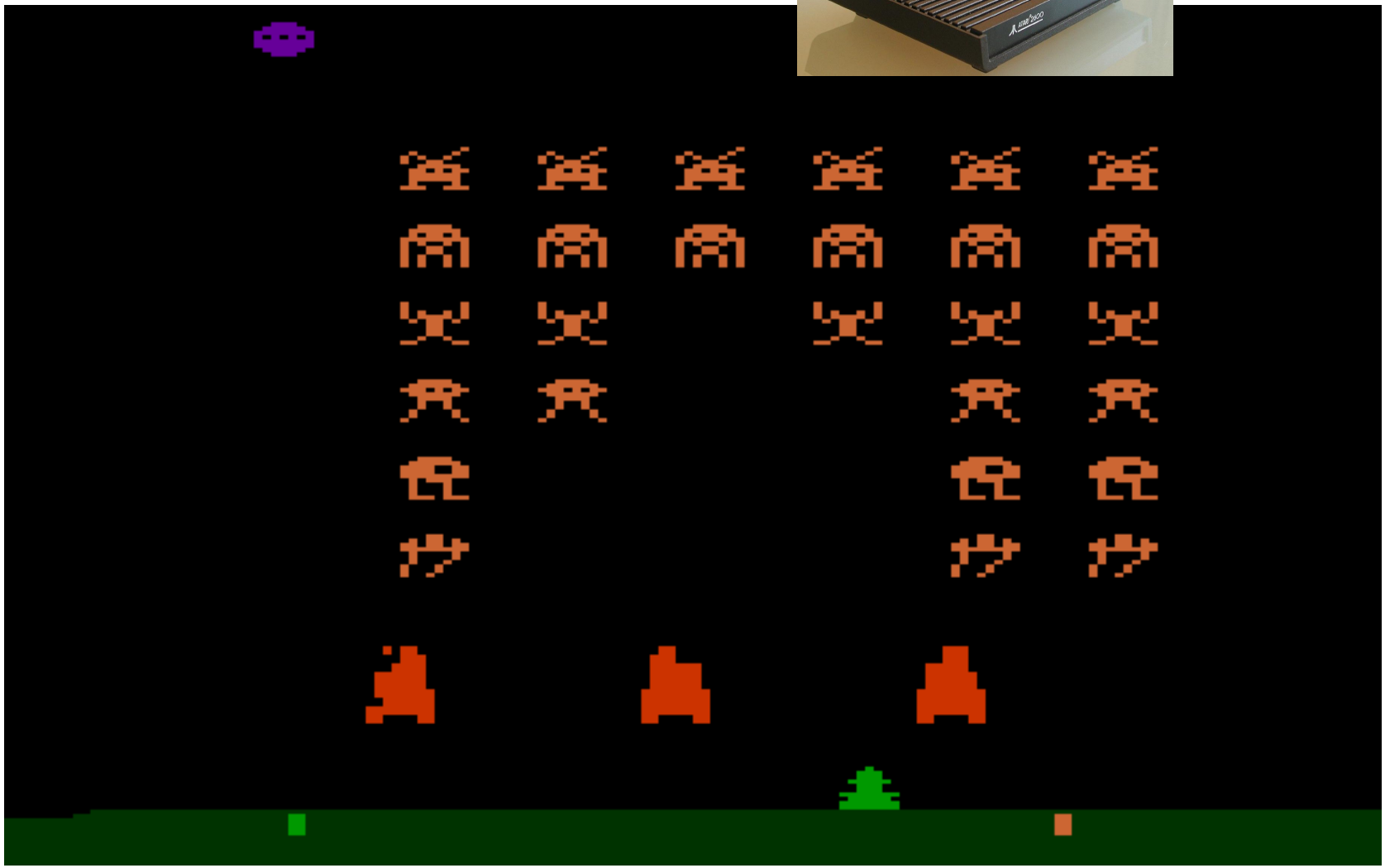
- Si les transistors étaient des personnes

2300	134000	32 millions	1,3 milliards
Opéra	Stade de foot	Pop. de Tokyo	Pop. Chine

Toujours sur la surface d'un opéra...

1977 : Console Atari 2600

1,19 MHz



2006 : Sony PS3

Processeur Cell 3.2 GHz

Processeur graphique Nvidia RSX

2006 : Sony PS3

Processeur Cell 3.2 GHz

Processeur graphique Nvidia RSX

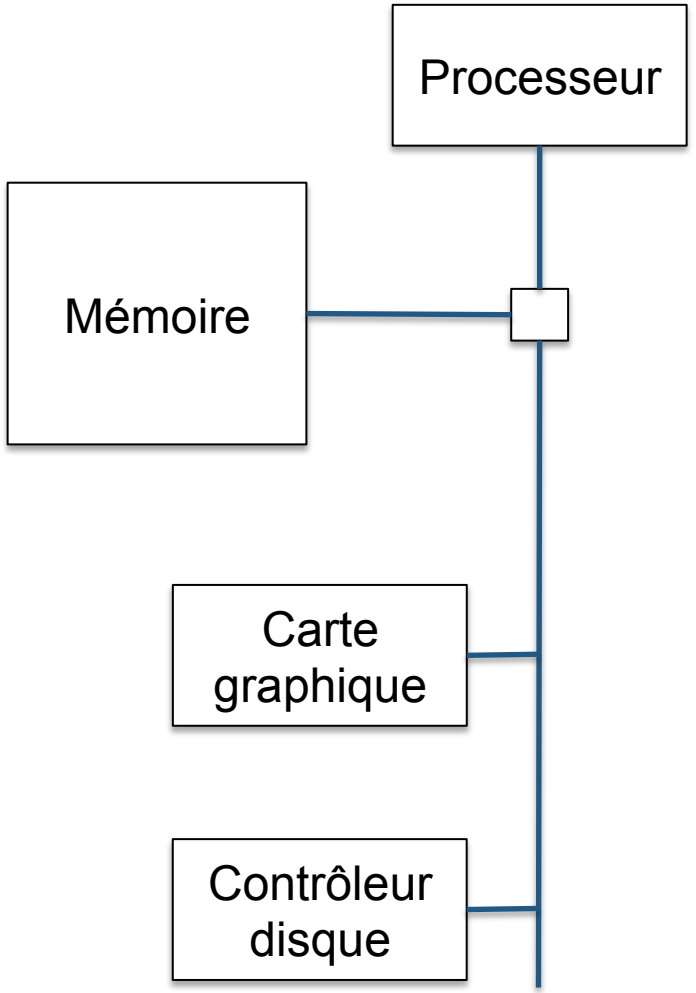
La PS3 calcule des millions de fois plus vite !

Architecture des ordinateurs

Vue d'ensemble

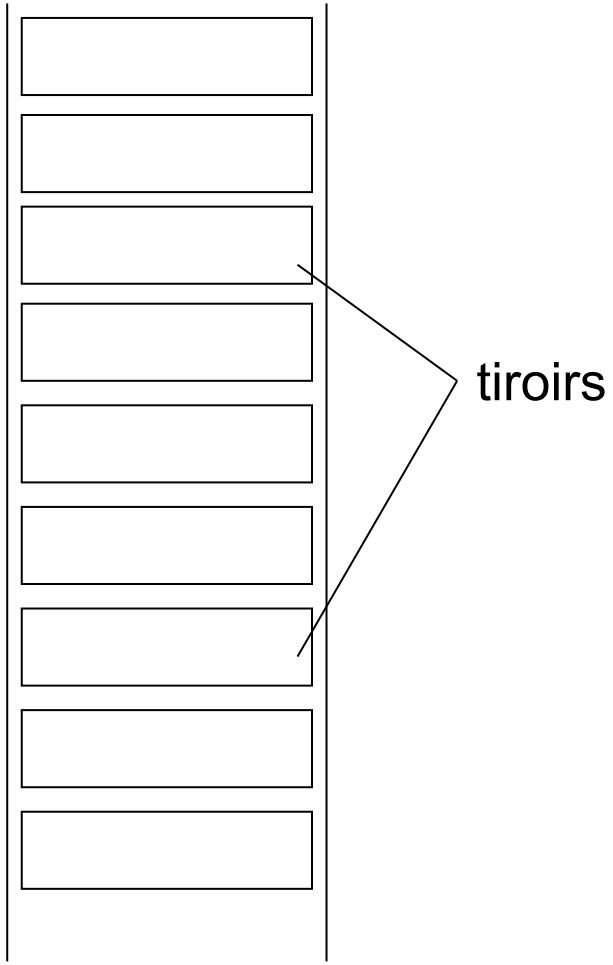
- Processeur
  - Unité arithmétique & logique
  - Registres
- Mémoire

- Périphériques d'Entrée-Sortie
  - Clavier, Souris
  - Contrôleur disque
  - Carte graphique
  - Carte réseau



Principe de fonctionnement d'un ordinateur

microprocesseur

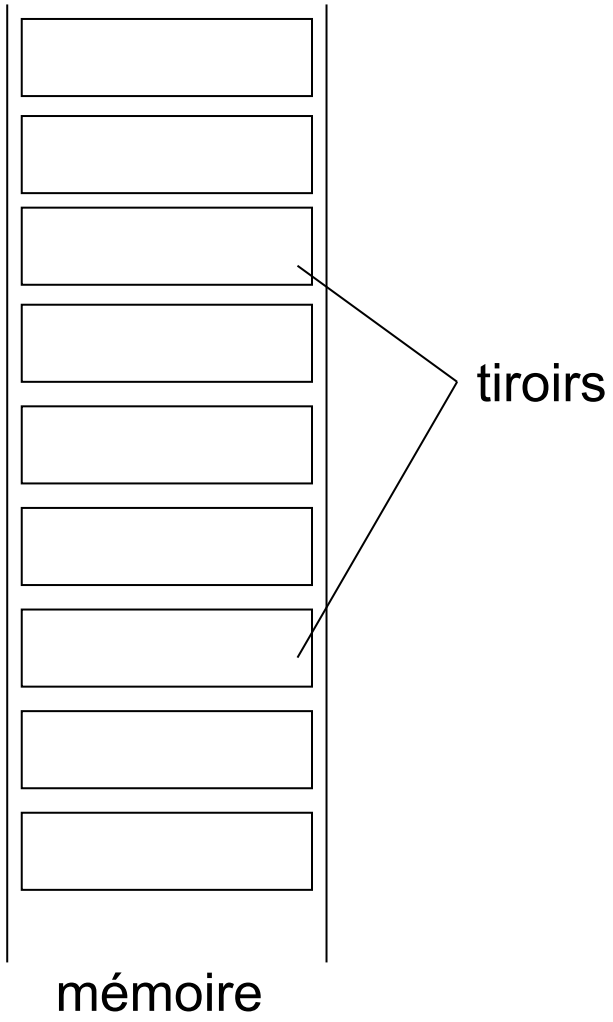


mémoire

Principe de fonctionnement d'un ordinateur

microprocesseur

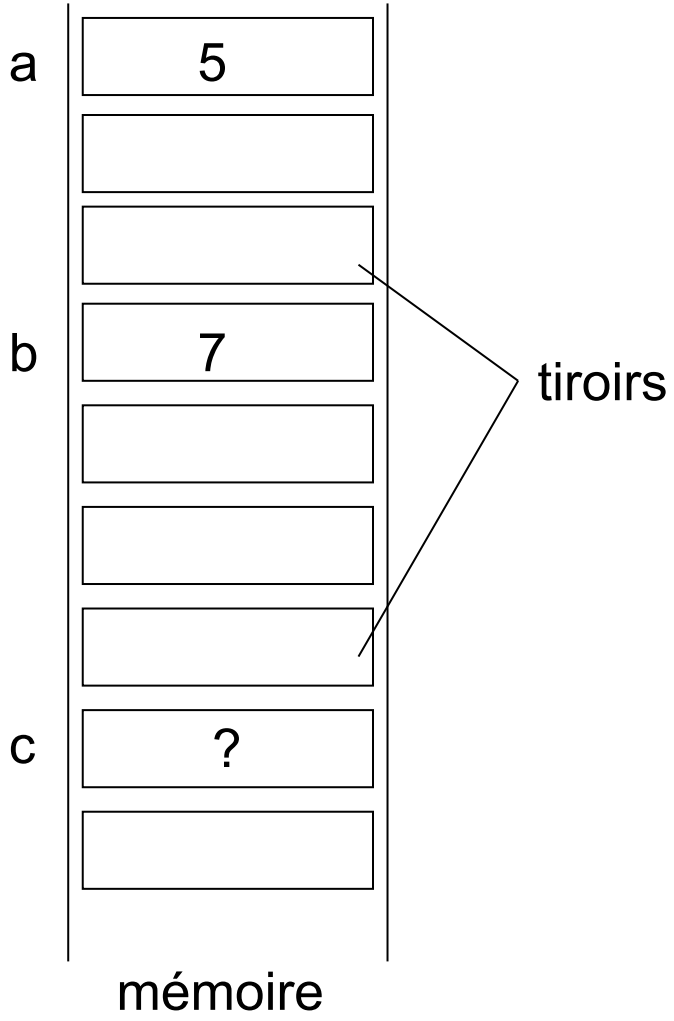
Il sait :
- aller chercher un nombre depuis la mémoire
- faire des opérations simples avec le contenu de ses *registres*
- ranger des nombres en mémoire



Principe de fonctionnement d'un ordinateur

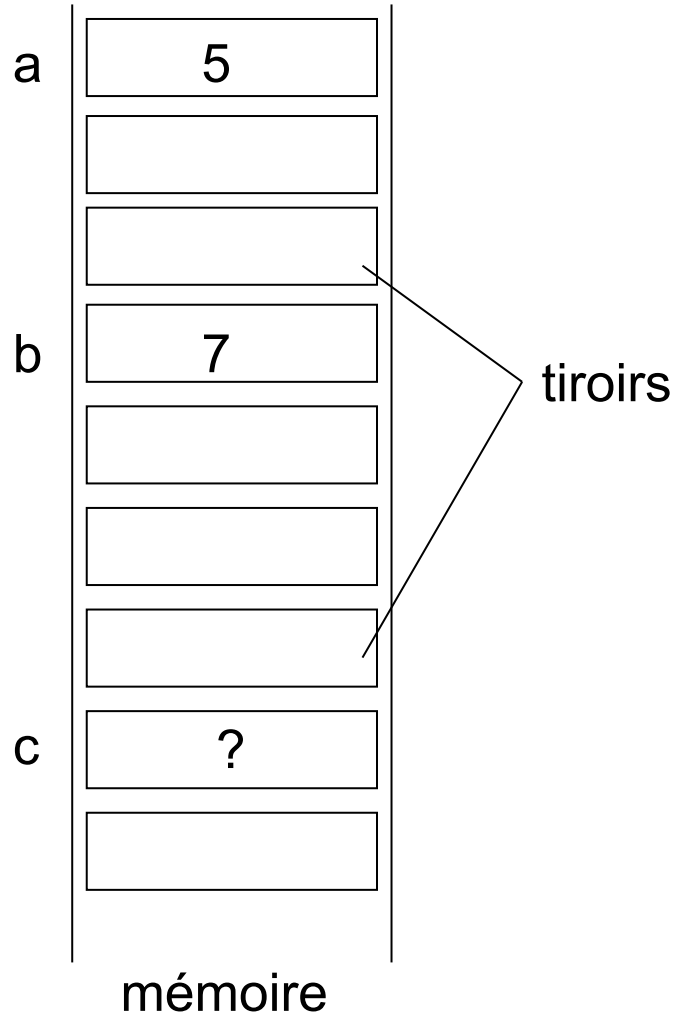
microprocesseur

Pour calculer l'opération c = 2 x a + b, il faut :



Principe de fonctionnement d'un ordinateur

microprocesseur



Pour calculer l'opération c = 2 x a + b, il faut :

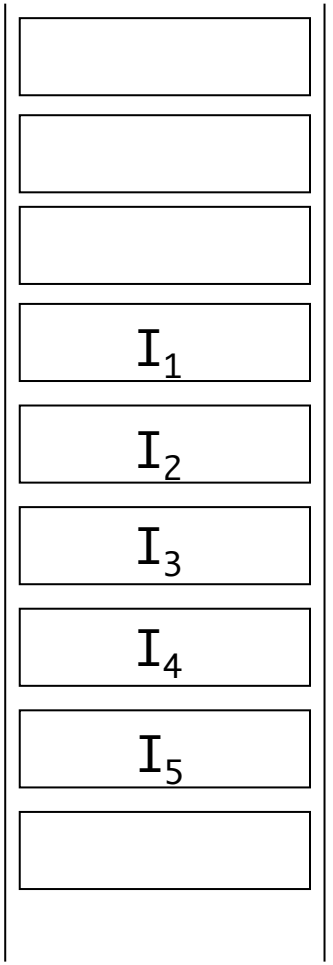
- charger a dans le registre r1
- calculer r1 = r1 * 2
- charger b dans le registre r2
- calculer r1 = r1 + r2
- ranger r1 dans c

Principe de fonctionnement d'un ordinateur

microprocesseur

Il s'agit d'un programme :

- charger a dans le registre r1 — Instruction 1
- calculer r1 = r1 * 2 — Instruction 2
- charger b dans le registre r2 — Instruction 3
- calculer r1 = r1 + r2 — Instruction 4
- ranger r1 dans c — Instruction 5

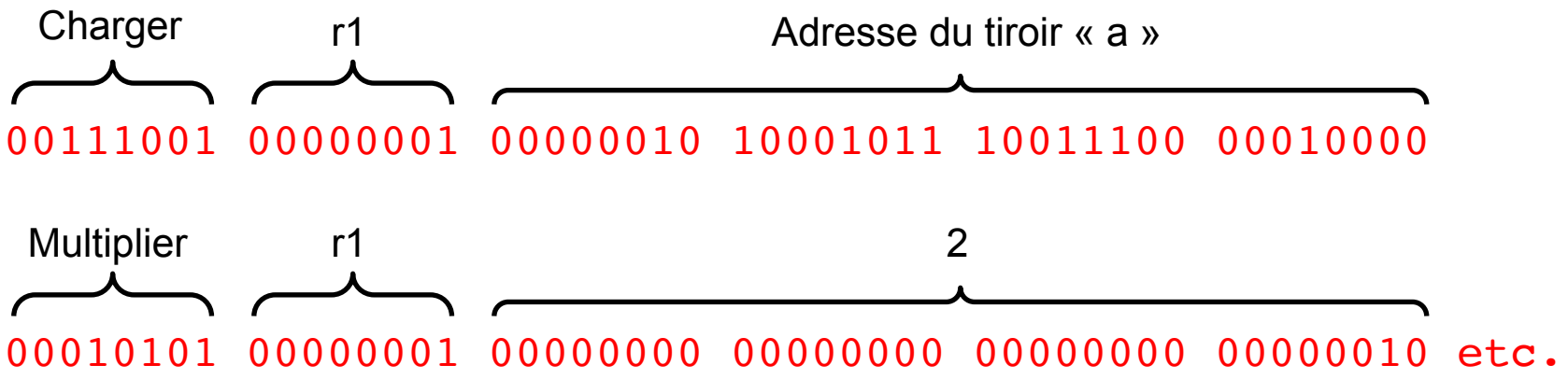


mémoire

Principe de fonctionnement d'un ordinateur

Notion de programme

- Vu que les instructions se trouvent en mémoire — qui ne contient de des nombres — chaque instruction est codée par un ou plusieurs nombres
  - Un programme est donc une suite de nombres binaires



  - Cela peut rapidement devenir rébarbatif pour un être humain
    - D'où l'invention de langages de programmation de « haut niveau »

Architecture des ordinateurs

Exemple de génération de code

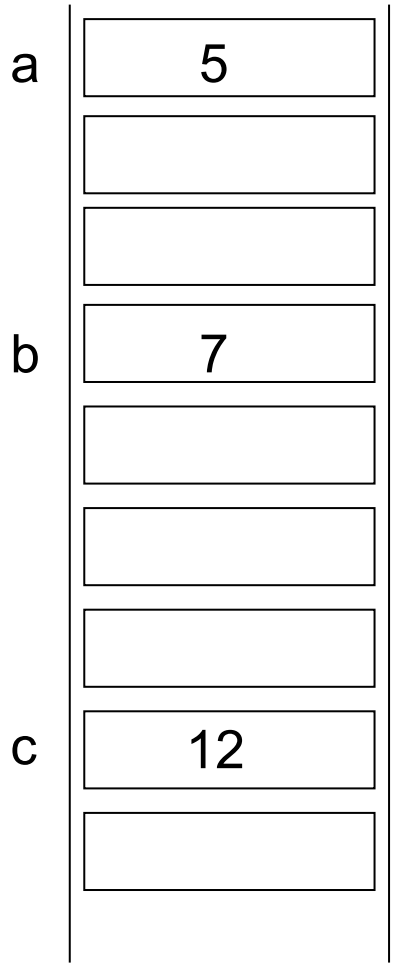
```
int x, y, n;
…
y = x;
n = 9;
while (n > 0) {
  y = y * x;
  n = n – 1;
}
```

```
  move 9, r1
  load @x, r2
  move r2, r3
start:
  mult r2, r3, r3
  decr r1
  jump_if_not_zero start
  store r3, @y
  halt
```

$$y = x^{10}$$

Les processeurs disposent d'un « cache »

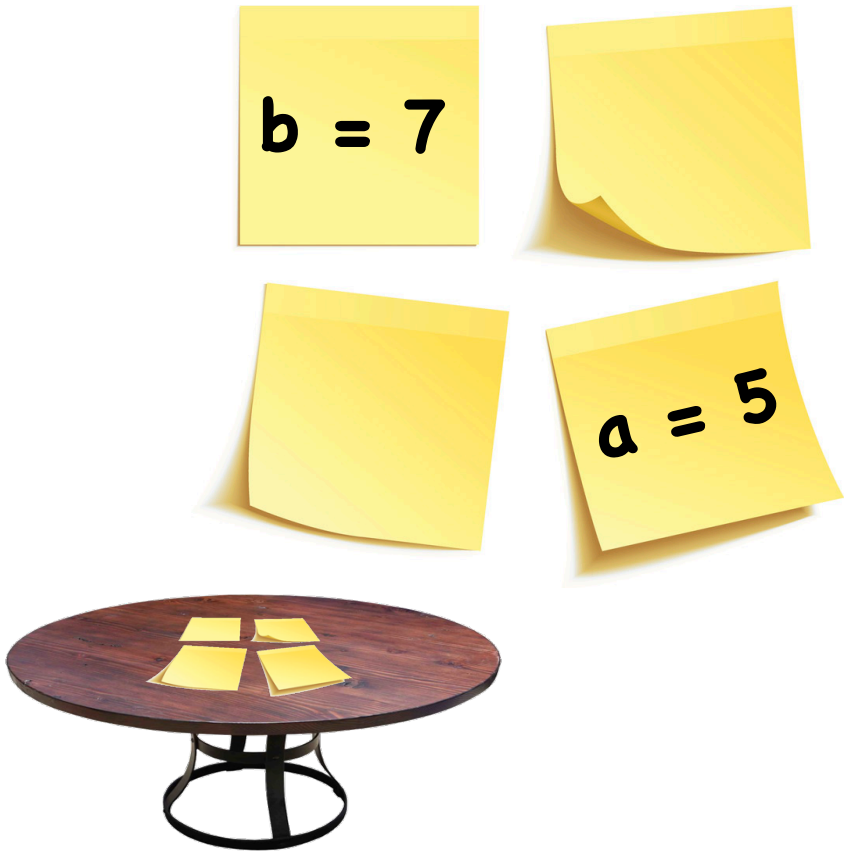
microprocesseur

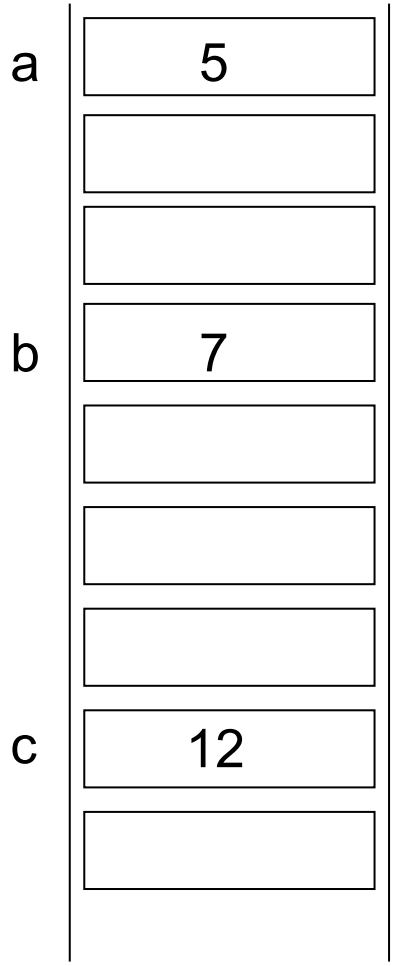


Les processeurs disposent d'un « cache »

Où ils conservent une copie des accès les plus … importants ☺

microprocesseur





mémoire

Le rôle
du
Système d’Exploitation

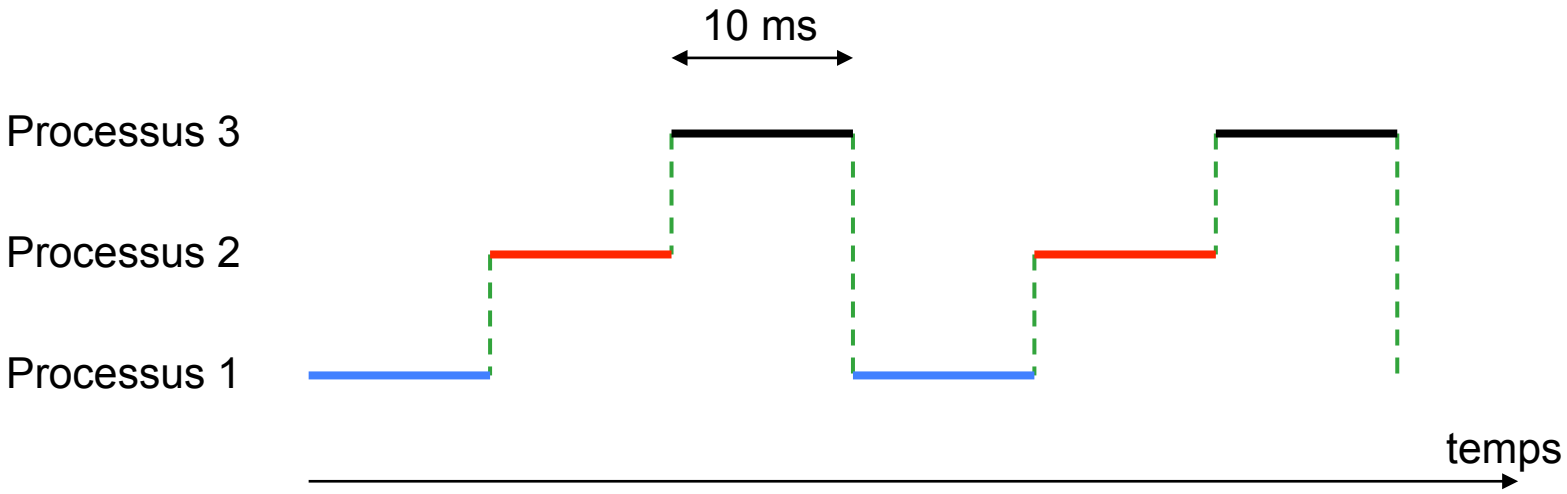
Rôle du système d'exploitation

- Les systèmes d'exploitation modernes (Windows, OS X, Linux) ont pour objectifs :
  - De factoriser du code commun à de multiples applications
    - Entrées-Sorties, allocation mémoire, etc.
  - D'abstraire le matériel en masquant sa diversité et sa complexité
    - Pilotes de périphériques, notion de fichier, etc.
  - D'arbitrer l'accès aux ressources
    - Les applications ont l'illusion d'avoir l'accès direct au processeur, à la mémoire… alors que ces ressources sont partagées

Arbitrage de l'accès aux ressources

Le processeur

- L'exécution des processus (applications) est régulièrement interrompue par un temporisateur matériel
  - Le système d'exploitation « reprend la main » un très court instant…
  - ... Et peut décider d'interrompre le processus en cours pour reprendre l'exécution d'un autre
    - Dans ce cas, les registres du processeurs sont sauvegardés/ restaurés



Arbitrage de l'accès aux ressources

La mémoire

- Les applications manipulent uniquement des « adresses mémoire » virtuelles
  - Le système d'exploitation établit une projection de ces adresses virtuelles vers des adresses physiques (réelles)
    - Le processeur effectue une traduction d'adresse à chaque accès
  - Lorsqu'on exécute simultanément plusieurs instances d'un même programme (e.g. Firefox), chaque variable possède la même adresse virtuelle…
    - ...mais pas la même adresse physique

Des questions ?

raymond.namyst@labri.fr