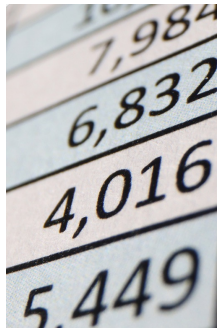
Information, Calcul et Communication

CS-119(g) ICC – Théorie
Semaine 2

Rafael Pires
rafael.pires@epfl.ch

Lausanne, 19.09.2025

Précédemment, dans ICC-T 01

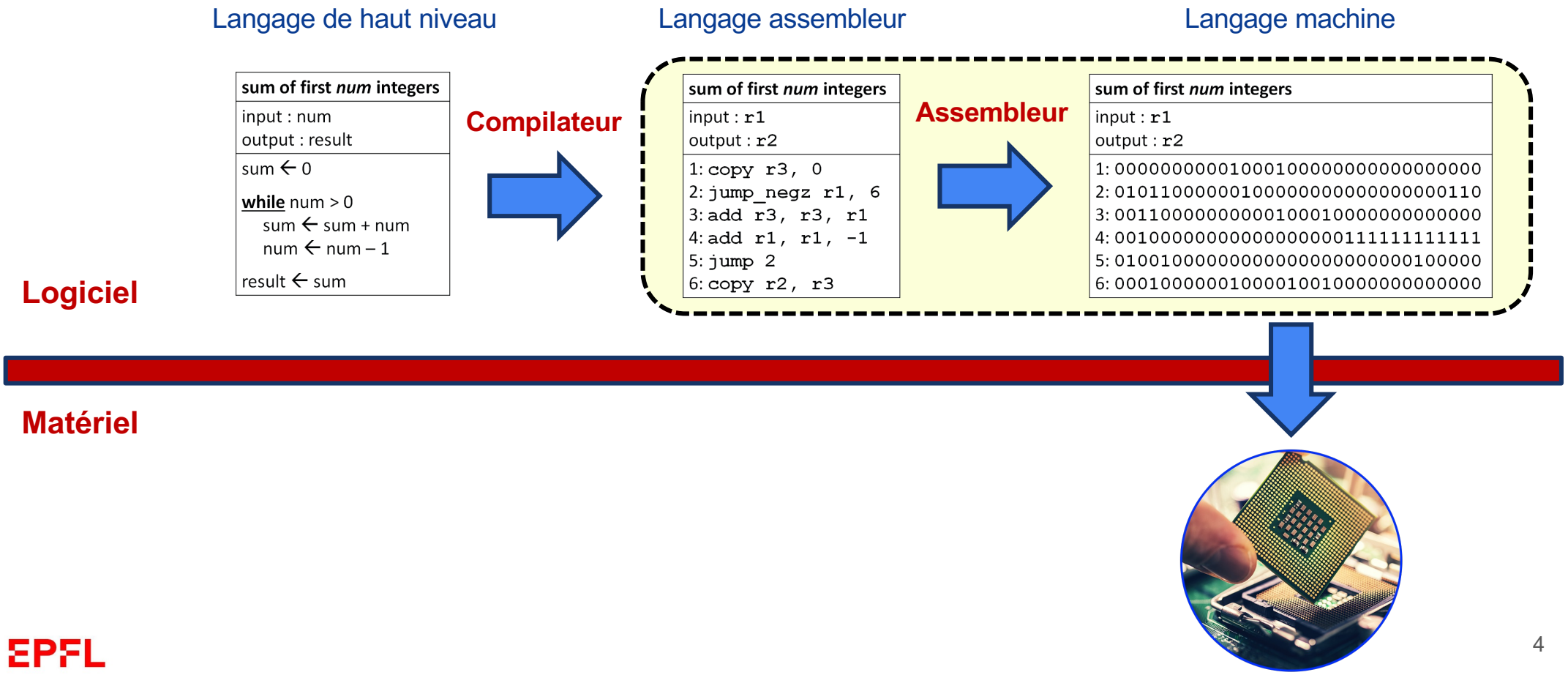


- Représentation de l'information
- Représentation binaire des nombres entiers
  - Nombres positifs, négatifs (complément à deux)
  - Opérations (addition, soustraction, multiplication, division)
  - Dépassement de capacité (overflow)
- Représentation binaire des nombres réels
  - Virgule fixe : erreur relative inévitable
  - Virgule flottante : signe, exposant, mantisse
- Représentation de textes et images

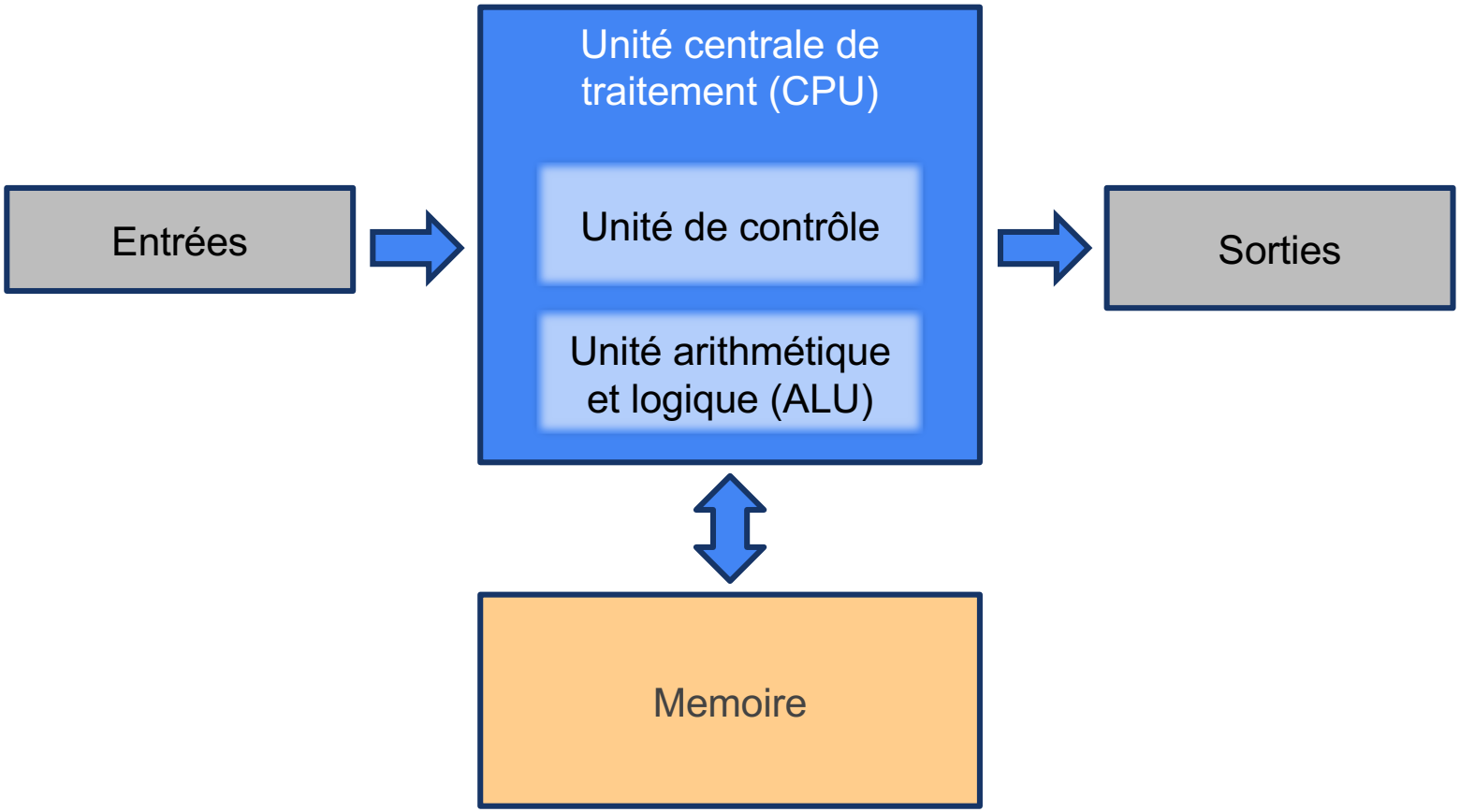
Aujourd’hui

- **Introduction à l’architecture des ordinateurs**
- Circuits logiques
- Transistors
- Introduction au langage assembleur

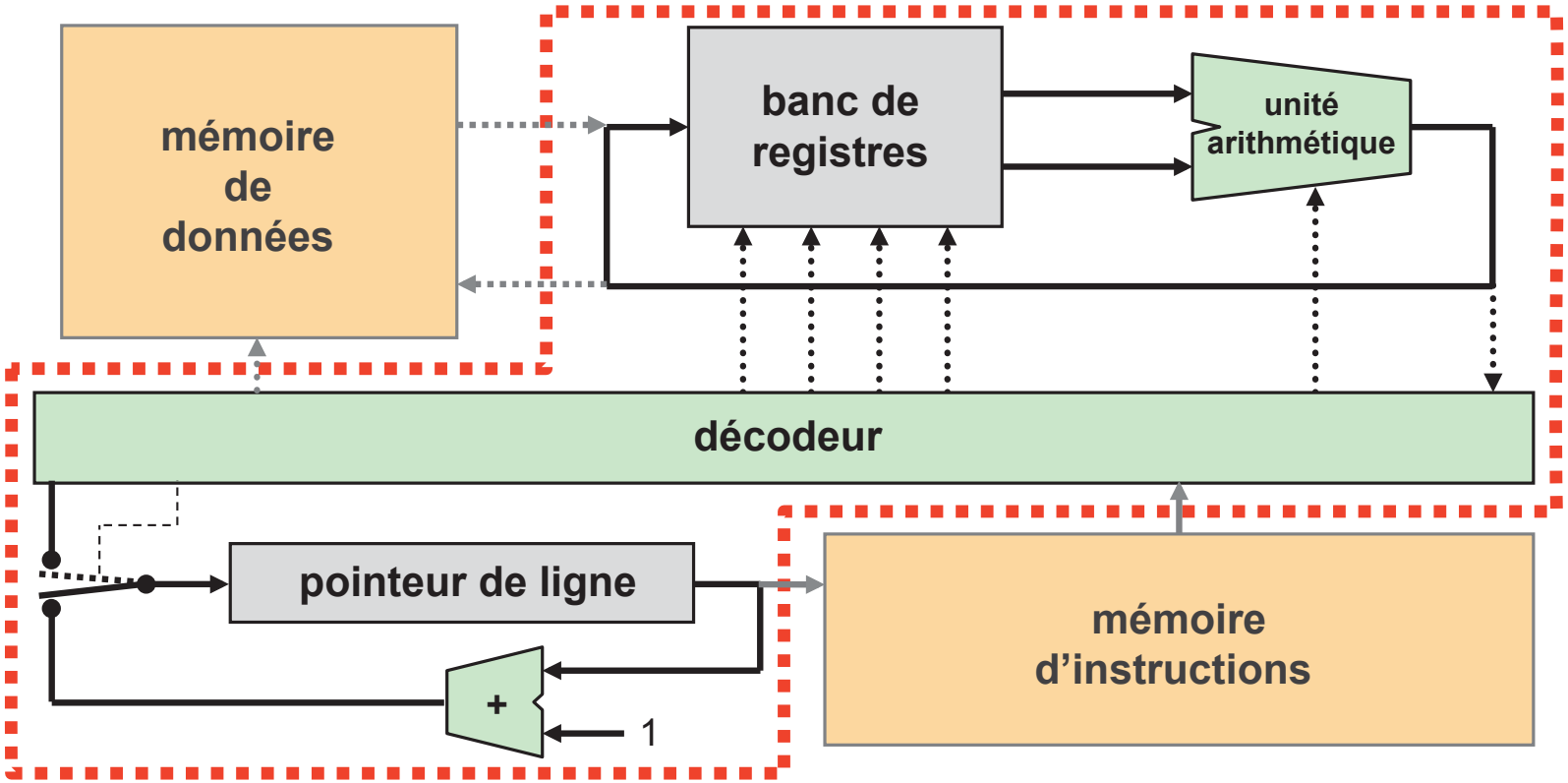
Introduction à l'architecture des ordinateurs



L’architecture de von Neumann



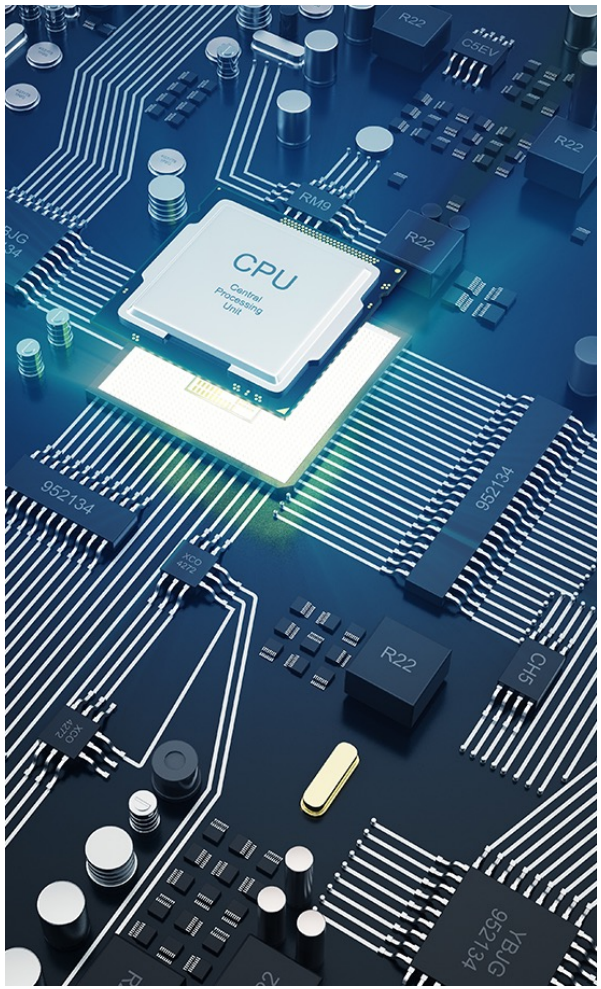
Le processeur



Aujourd’hui

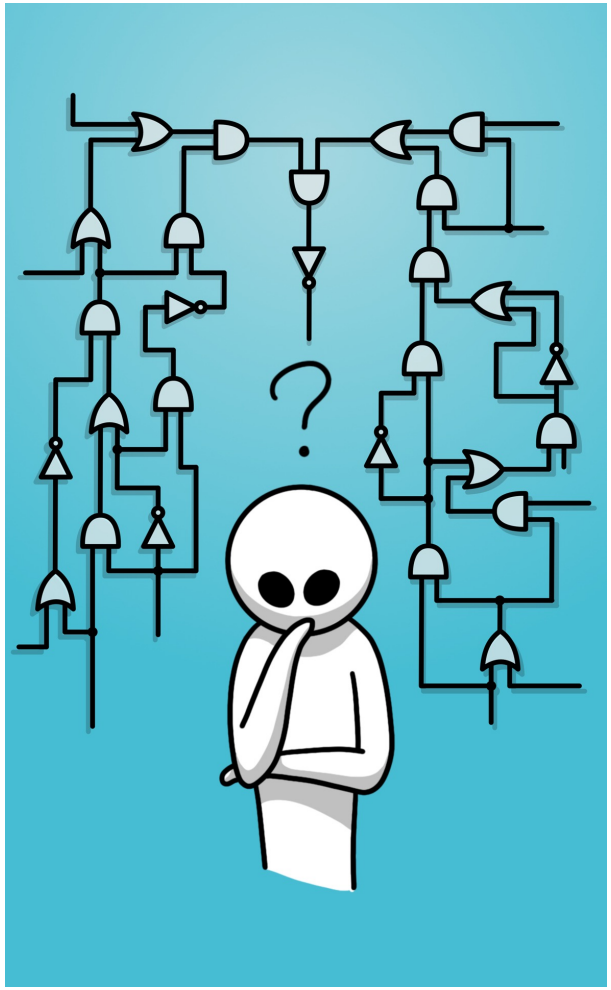
- Introduction à l’architecture des ordinateurs
- **Circuits logiques**
- Transistors
- Introduction au langage assembleur

Introduction : Transistors et circuits logiques



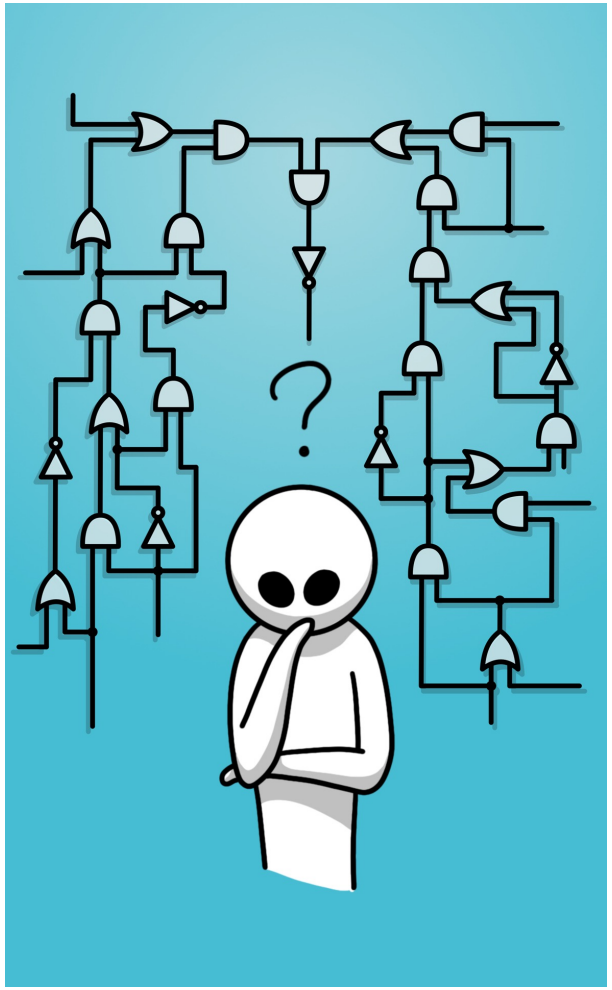
- **Le transistor** : brique élémentaire, agit comme un interrupteur pour représenter 0 et 1
- **Portes logiques** : associations de transistors pour effectuer des opérations binaires (ET, OU, NON…)
- **Processeurs** : assemblage géant de ces portes logiques (à l'échelle de milliards de transistors), réalisant calculs et contrôles

Introduction : Circuits logiques



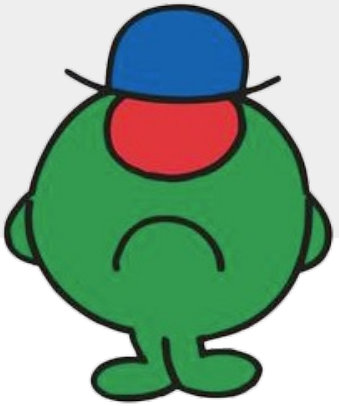
- Un circuit logique est un ensemble de portes logiques reliées entre elles.
- Ces portes logiques permettent de réaliser des opérations élémentaires sur des bits.
- Chaque porte logique est caractérisée par une table de vérité établissant une correspondance entre les entrées et les sorties de cette porte.

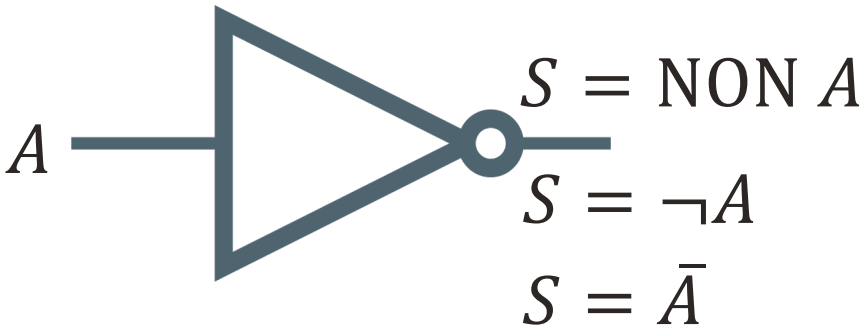
Introduction : Circuits logiques



- Chaque porte logique est également représentée par un symbole.
- Nous verrons que l'on peut combiner plusieurs portes logiques ensemble pour faire tout type d'opération, comme un additionneur, par exemple.

La porte NON (NOT)





A	S = NOT A
0	1
1	0

- Elle possède une seule entrée
- La porte NON donne en sortie, la négation de la valeur du bit d'entrée
- Notez que le cercle à la sortie d'une porte logique signifiera toujours la négation

La porte ET (AND)

A	B	S = A AND B
0	0	0
0	1	0
1	0	0
1	1	1

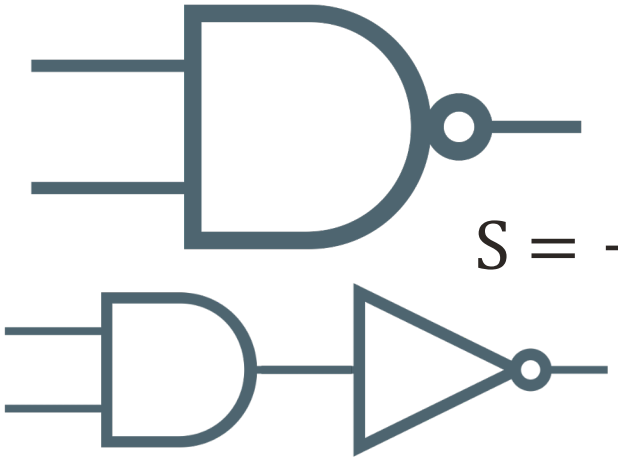
- Elle comporte deux ou plusieurs entrées.
- La porte ET génère un 1 en sortie si et seulement si tous les bits en entrée sont égaux à 1. Dans le cas contraire, la sortie vaut 0.
- Notez que la valeur de la sortie S correspond au produit des valeurs d'entrées $A{\cdot}B$.

La porte OU (OR)

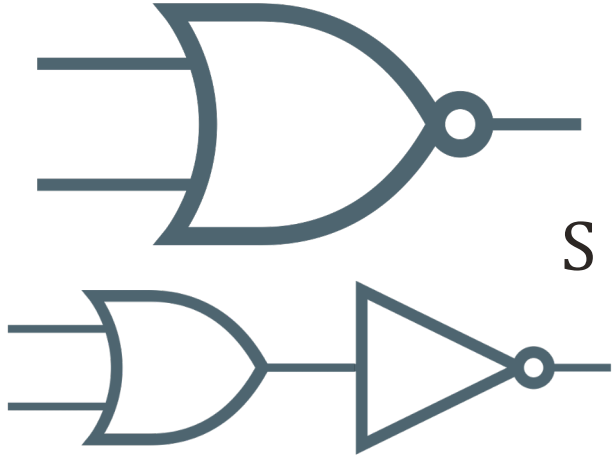
A	B	S = A OR B
0	0	0
0	1	1
1	0	1
1	1	1

- Elle comporte deux ou plusieurs entrées.
- La porte OU génère un 1 en sortie si au moins un des bits en entrée vaut 1. La sortie vaut donc 0 en sortie si et seulement si tous les bits en entrée valent 0.
- Notez que la valeur de sortie S vaut 1 quand $A + B \geq 1$ (mais n'est donc pas égale à A+B).

Les portes NON ET (NAND) et NON OU (NOR)



$S = \neg(A \wedge B)$

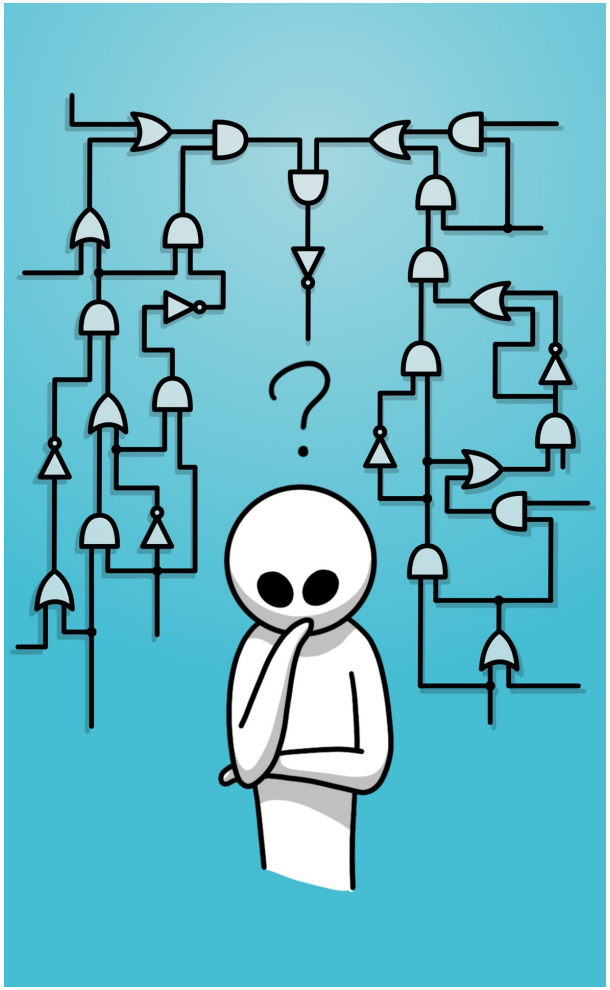


$S = \neg(A \vee B)$

A	B	S = A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

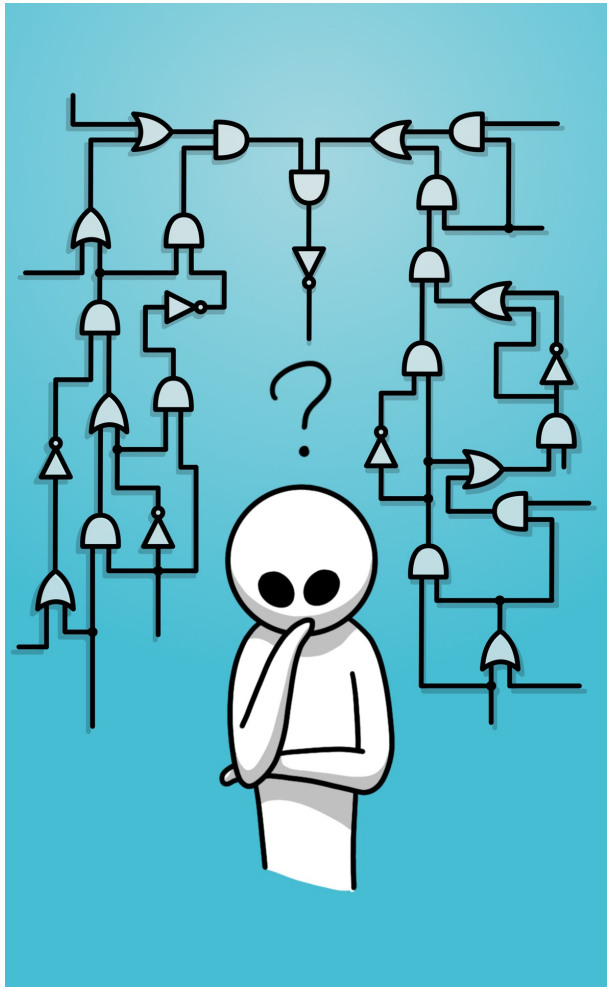
A	B	S = A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

En pratique



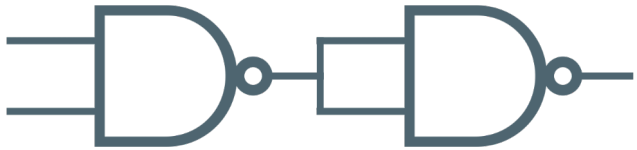
- Avec les trois portes de base (NON, ET, OU), on peut créer tous les circuits possibles et donc effectuer toutes les opérations possibles.
- Il est possible de représenter une porte logique comme étant la composition d'autres portes logiques.
- En électronique, la porte NON ET est la plus simple à réaliser du point de vue technologique. Pour cette raison, elle sert souvent de brique de base aux circuits intégrés. On peut reconstituer toutes les fonctions logiques uniquement à l'aide de portes NON ET.

En pratique



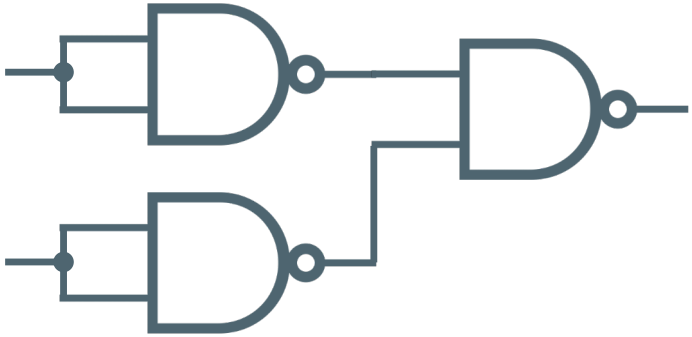
NON

$$\bar{A} = \overline{A \cdot A}$$



AND

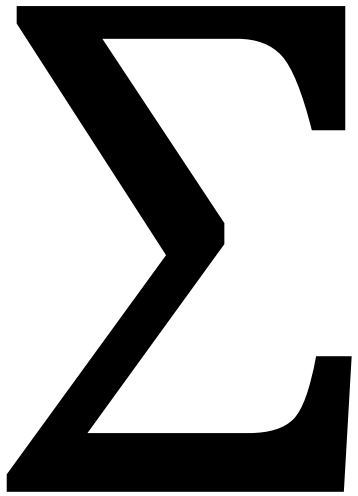
$$A \cdot B = \overline{\overline{A \cdot B} \cdot \overline{A \cdot B}}$$



OR (loi de De Morgan)

$$A + B = \overline{\bar{A} \cdot \bar{B}}$$

Additionner deux bits (sans retenue)



- On aimerait créer un circuit avec entrées A et B et sortie S dont la table de vérité soit :

A	B	(A+B)%2
0	0	0
0	1	1
1	0	1
1	1	0

- Pour créer ce circuit, remarquez que $S=1$ si et seulement si :

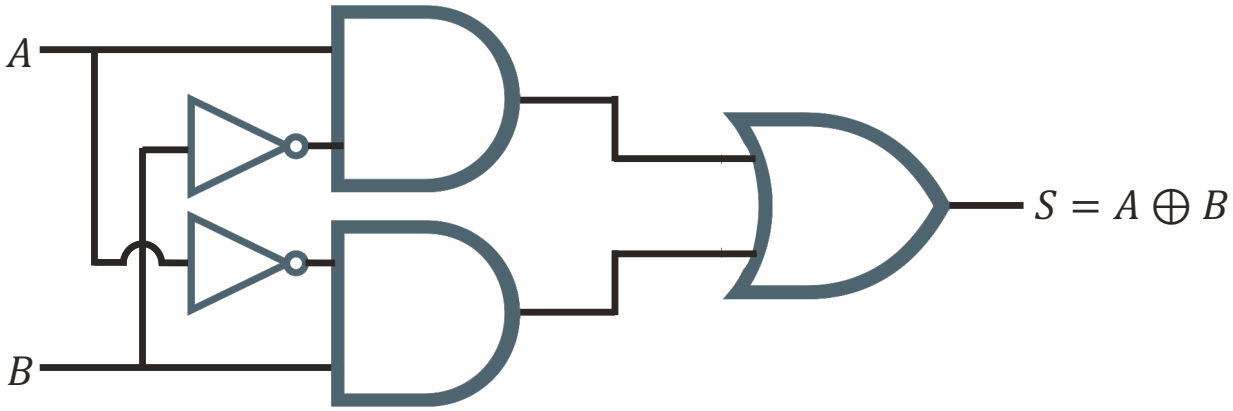
$$(\ A=1 \text{ ET } B=0\ ) \text{ OU } (\ A=0 \text{ ET } B=1\ )$$

- Autrement dit:

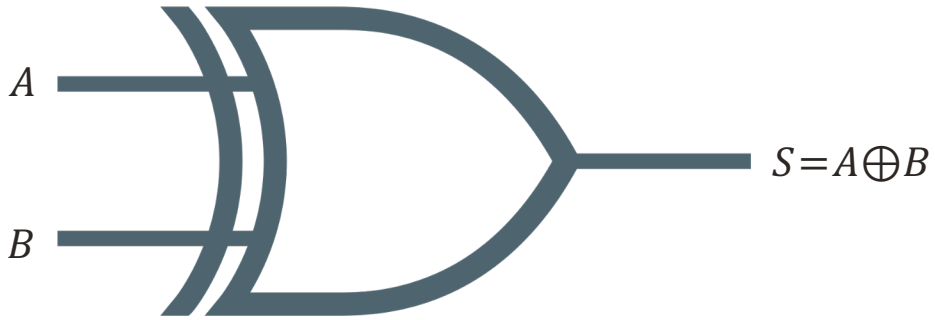
$$S=(\ A \text{ ET NON } B\ ) \text{ OU } (\ \text{NON } A \text{ ET } B\ )$$

Additionner deux bits (sans retenue)

S=(A ET NON B) OU (NON A ET B)



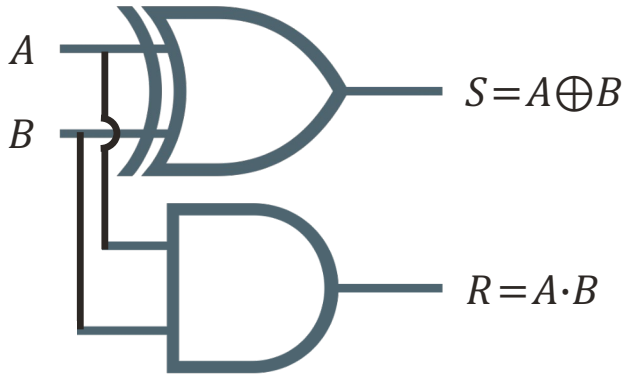
La porte OU Exclusif (XOR)

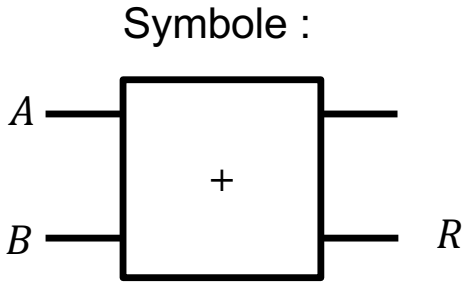


A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Additionner 2 bits (avec retenue)

- On aimerait maintenant créer un circuit avec entrées A et B et sortie $\mathrm{S}=\mathrm{A} \oplus \mathrm{B}$, ainsi qu'une retenue $\mathrm{R}=1$ si et seulement si $\mathrm{A}=1$ et $\mathrm{B}=1$.





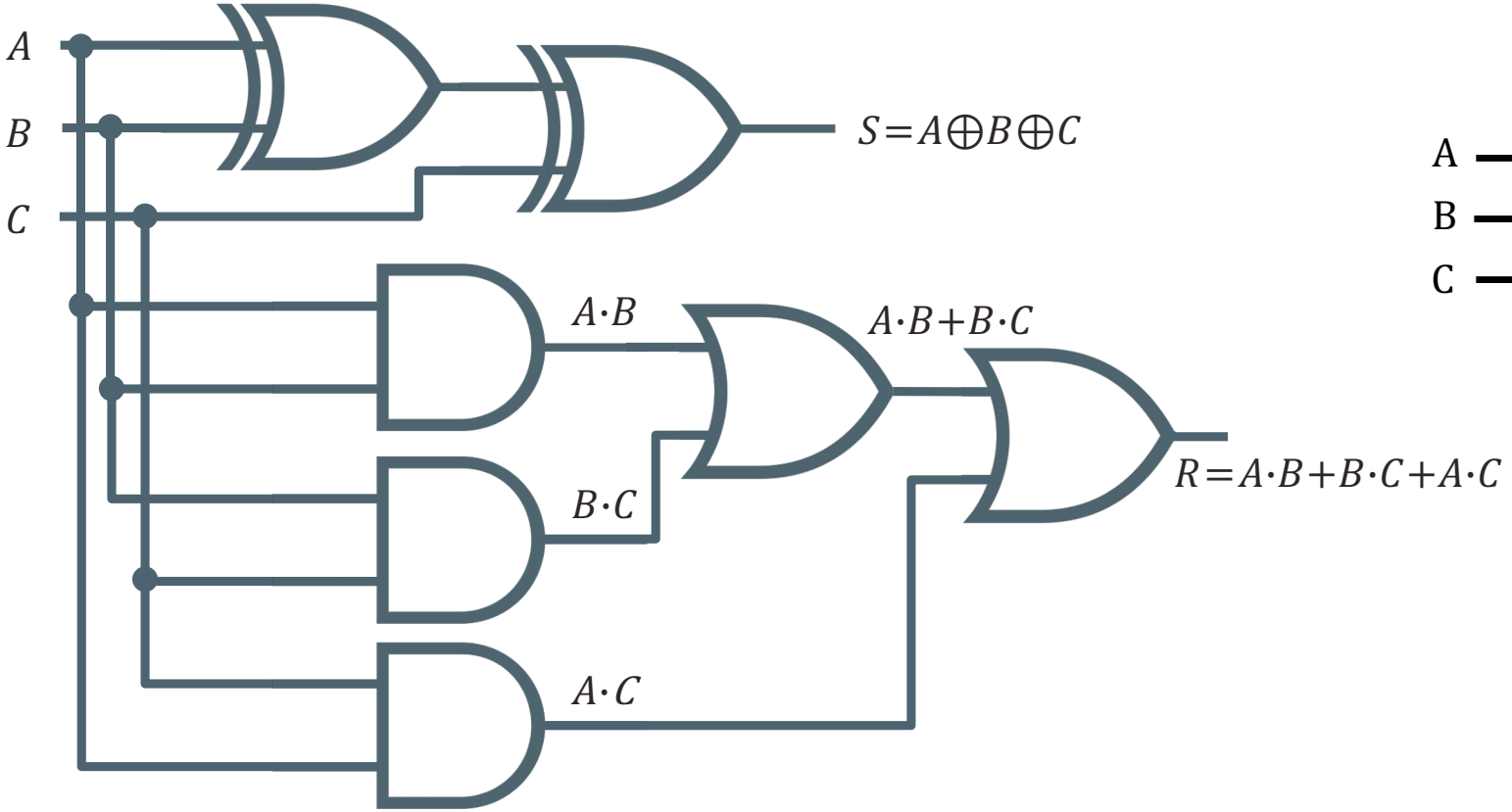
ICC-T 01 : Op. binaires : addition et soustraction

- **Addition :**

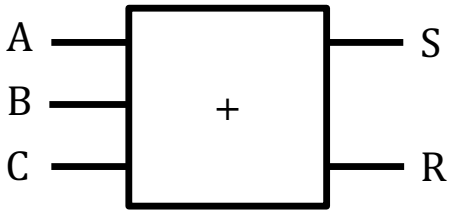
	111	
	0111	7
+	0011	3
	1010	10

0+0=0 0+1=1
1+1=10 1+1+1=11

Additionner 3 bits (avec retenue)



Symbole :



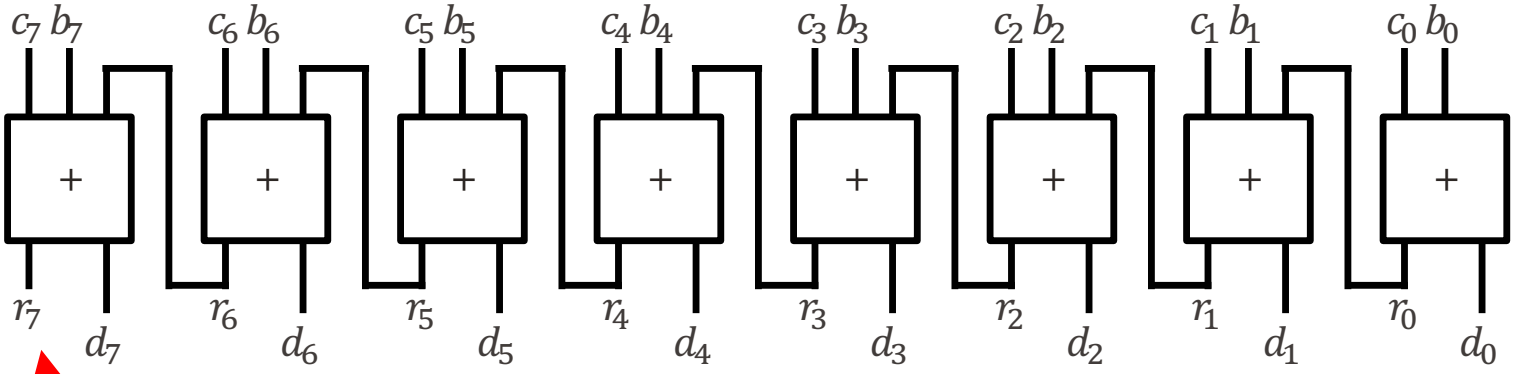
Additionneur sur 8 bits

Effectuer :

$$\begin{array}{r} b_7 b_6 b_5 \cdots b_1 b_0 \\ +c_7 c_6 c_5 \cdots c_1 c_0 \\ \hline = d_7 d_6 d_5 \cdots d_1 d_0 \end{array}$$

Exemple :

$$\begin{array}{r} \,111\;\;11 \\ \,00111001 \\ +\;\;00101011 \\ \hline = 01100100 \end{array}$$



Si $r_7 = 1 \Rightarrow$ overflow !

Mémoire vive

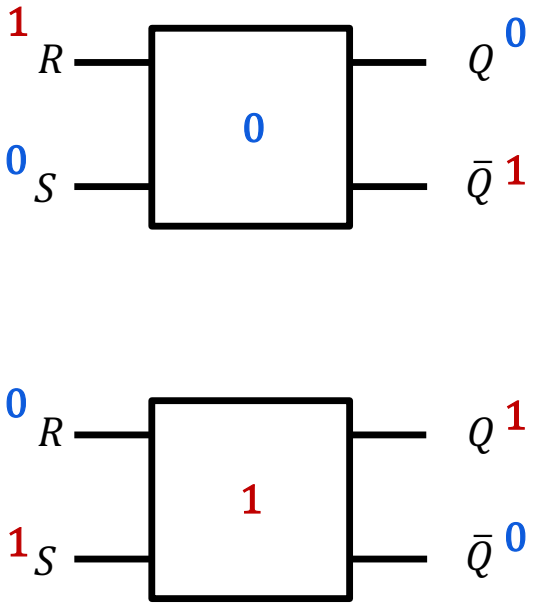
Jusqu'ici...

- Nous avons étudié des circuits combinatoires : leur sortie dépend uniquement des entrées à l'instant donné.
- Aucune mémoire : impossible de « retenir » un état.

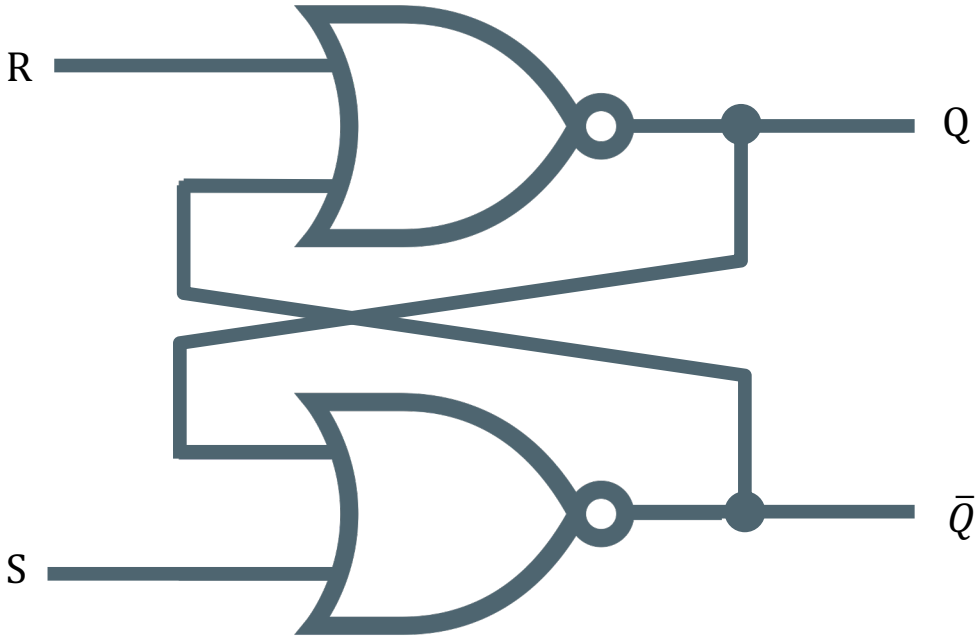
Et maintenant...

- Nous avons parfois besoin de conserver un bit de donnée (pour une commande, un signal d'erreur, etc.).
- Pour cela, on utilise des circuits séquentiels, c'est-à-dire avec mémoire.

Mémoire vive : SR Latch

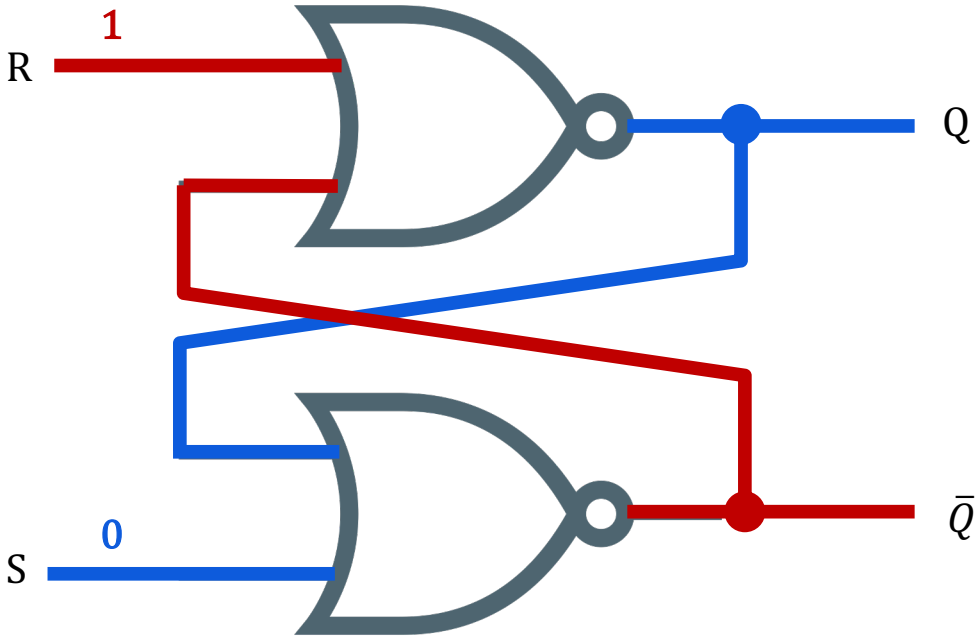


Mémoire vive : SR Latch



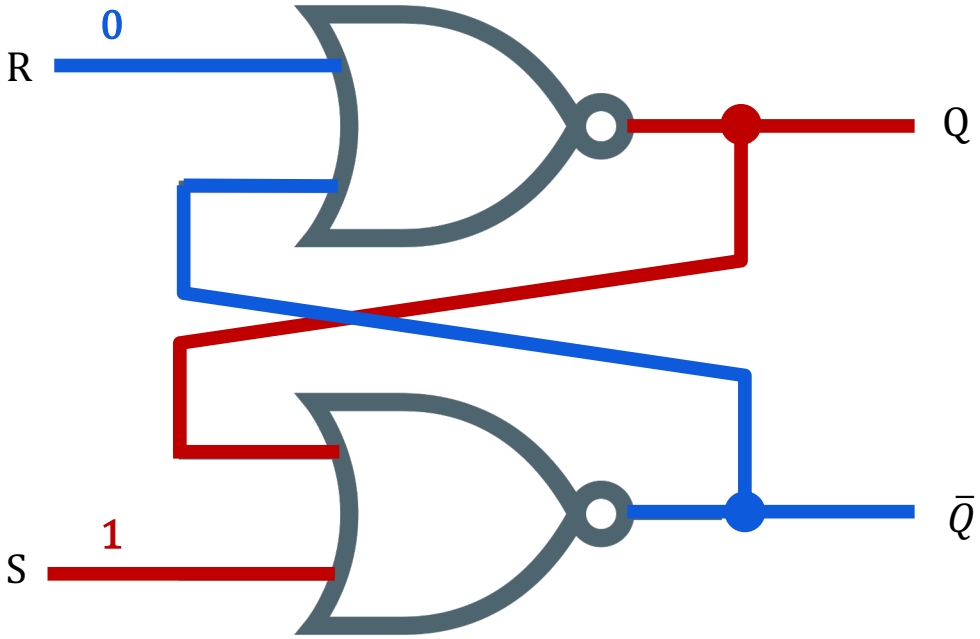
S	R	Q	$\bar{Q}$
0	0	1	0
		0	1
0	1	0	1
1	0	1	0
1	1	indéfini	

Mémoire vive : SR Latch



S	R	Q	$\bar{Q}$
0	0	1	0
		0	1
0	1	0	1
1	0	1	0
1	1	indéfini	

Mémoire vive : SR Latch

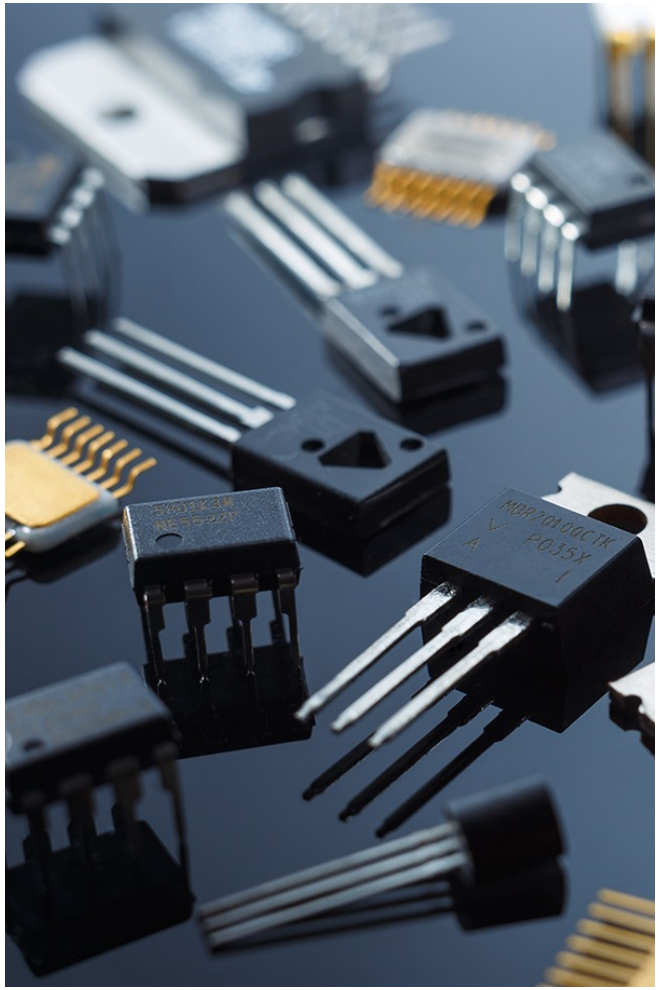


S	R	Q	$\bar{Q}$
0	0	1	0
		0	1
0	1	0	1
1	0	1	0
1	1	indéfini	

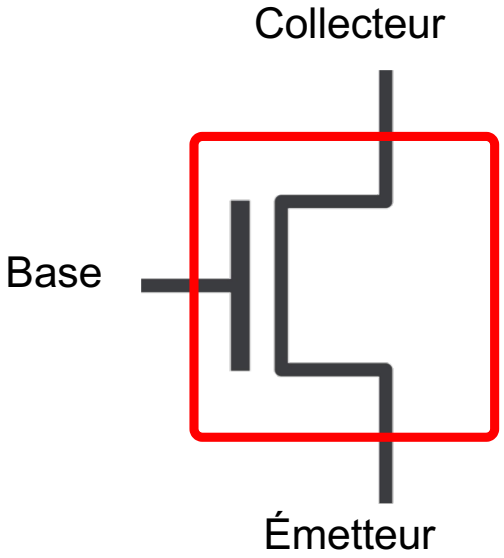
Aujourd'hui

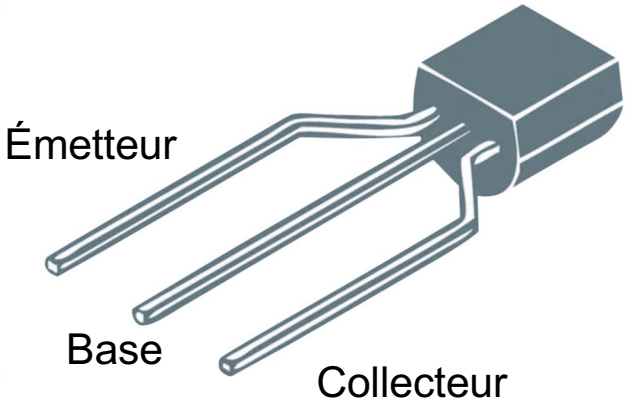
- Introduction à l'architecture des ordinateurs
- Circuits logiques
- **Transistors**
- Introduction au langage assembleur

Le transistor



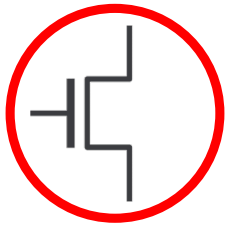
- Inventé en 1947 par trois américains : Bardeen, Shockley & Brattain
- Ce composant, qui est à la base de toute l'électronique moderne, a remplacé avantageusement les relais électromécaniques et les tubes à vide utilisés dans les premiers ordinateurs à la même époque → miniaturisation



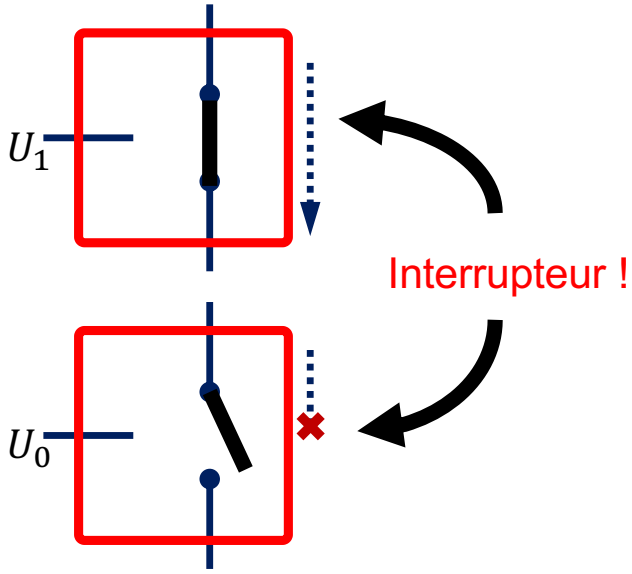


Principe de fonctionnement (n-mos)

- Symbole :



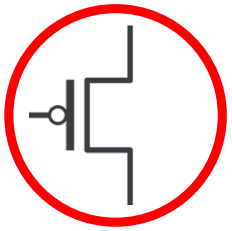
- Si la tension à la base est haute ($U_1=5V$) alors le courant passe entre l'émetteur et le collecteur :



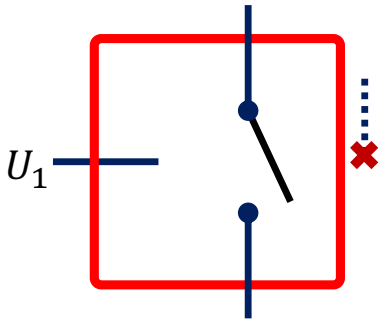
- Si la tension à la base est basse ($U_0=0V$) alors le courant ne passe pas entre l'émetteur et le collecteur :

Principe de fonctionnement (p-mos)

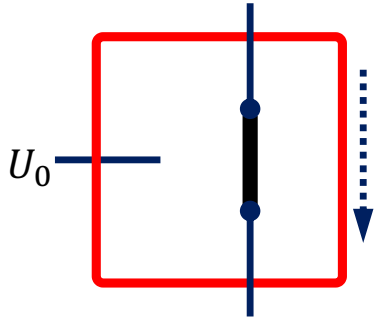
- Symbole :



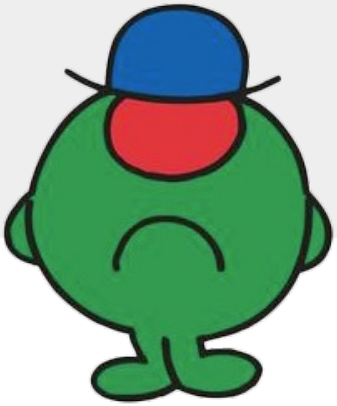
- Si la tension à la base est haute ($U_1 = 5V$) alors le courant ne passe pas entre l'émetteur et le collecteur :



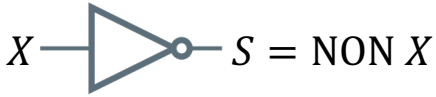
- Si la tension à la base est basse ($U_0 = 0V$) alors le courant passe entre l'émetteur et le collecteur :

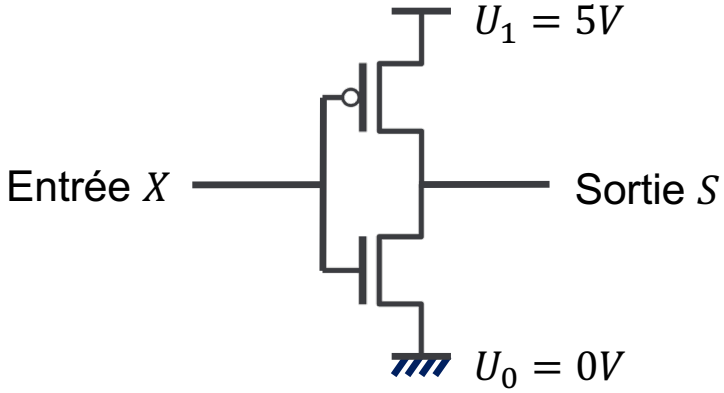


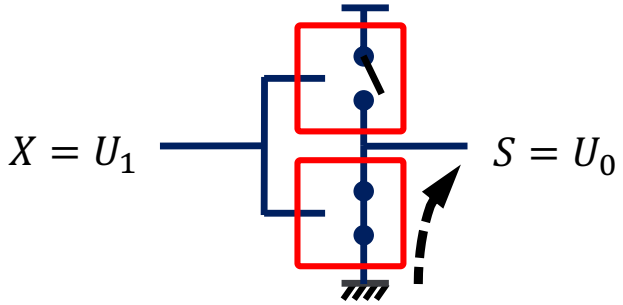
Création d'un inverseur

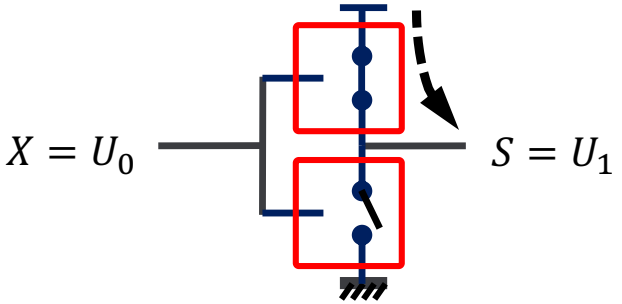


- Si on identifie U_0 comme 0 et U_1 comme 1, on peut créer un inverseur (porte NOT) à l'aide d'un transistor n-mos et d'un transistor p-mos

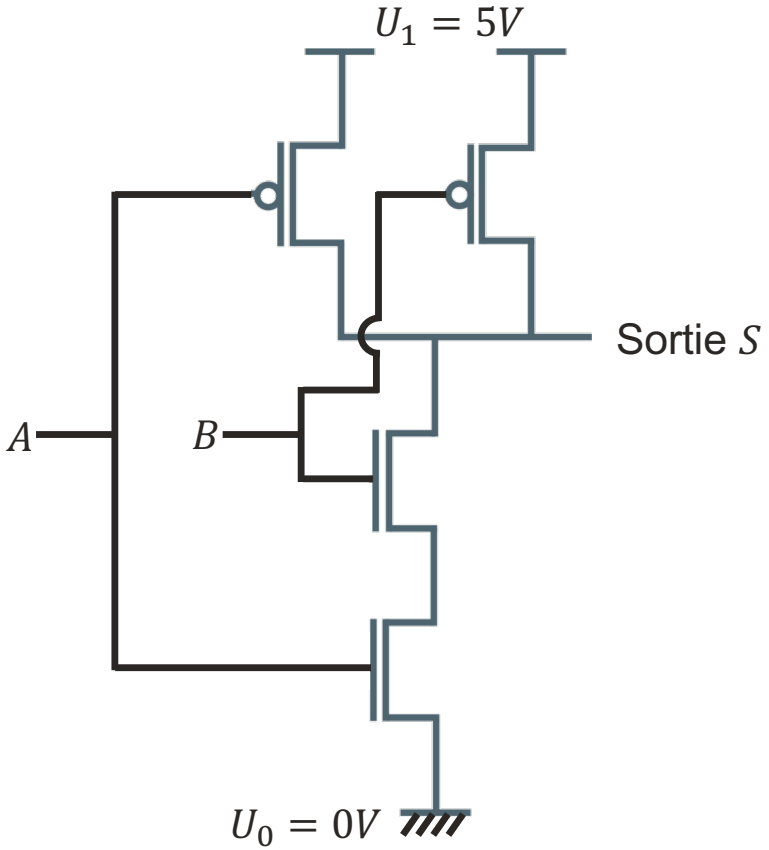






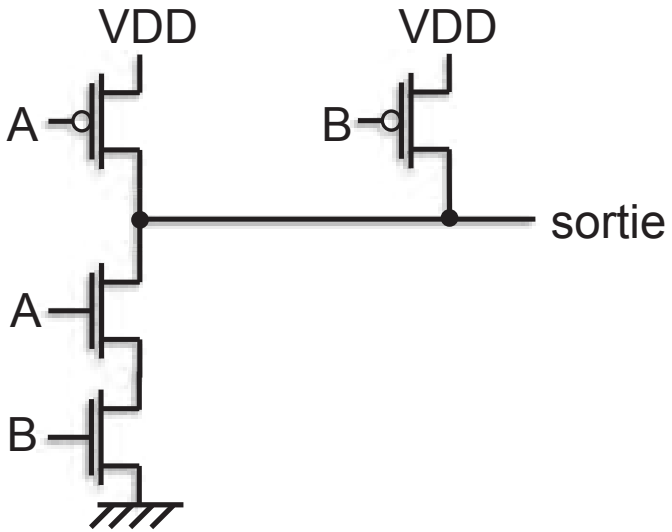


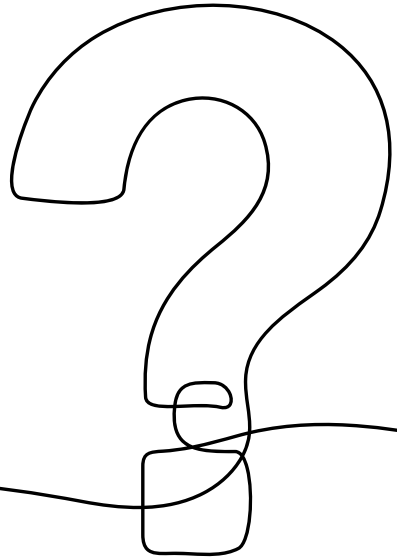
Création de la porte NAND



Question

Quelle est la sortie du circuit pour A = 1 et B = 0 ?

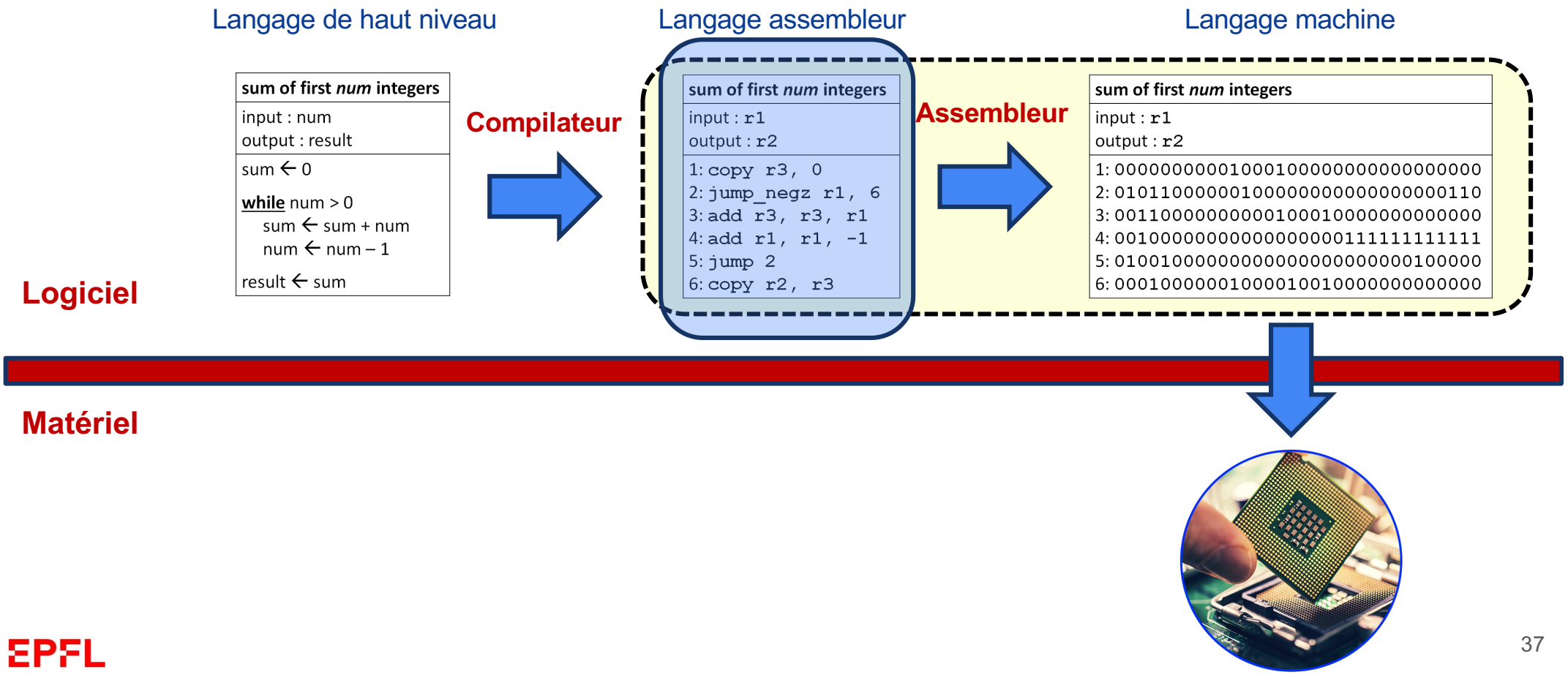




Aujourd’hui

- Introduction à l’architecture des ordinateurs
- Circuits logiques
- Transistors
- **Introduction au langage assembleur**

Introduction à l'architecture des ordinateurs



Langage assembleur

```
instruction registre_résultat, registre_opérande1, registre_opérande2
```

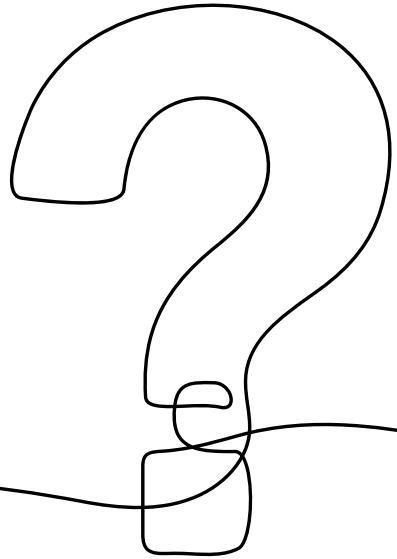
```
add
subtract
multiply
```

```
p. ex., r0 … r15
```

Typiquement, un processeur moderne dispose de 16 à 32 registres généraux.

Question

Écrivez un programme en assembleur pour calculer le discriminant d'une équation du second degré $\Delta = b^2 - 4ac$. Chaque coefficient est initialement stocké dans un registre : `r0` contient a, `r1` contient b, `r2` contient c. À la fin de l'exécution, le registre `r4` doit contenir la valeur de Δ. Utilisez uniquement les instructions de base (**`add`**, **`subtract`**, **`multiply`**).



Aujourd’hui

- Introduction à l’architecture des ordinateurs
- Circuits logiques
- Transistors
- Introduction au langage assembleur

Résumé Cours 2 – ICC-T

- Architecture de von Neumann : interaction entre mémoire, processeur et instructions.
- Les transistors n-mos et p-mos forment la base de l'électronique numérique en se comportant comme des interrupteurs commandés par la tension de grille.
- Les portes logiques (telles que l'inverseur ou la porte NAND) s'obtiennent en combinant ces transistors, et constituent les briques essentielles du calcul binaire.
- Un additionneur repose sur la porte XOR pour effectuer la somme binaire, et sur d'autres portes (AND, OR ...) pour gérer la retenue.
- Assembleur : instructions simples (`add`, `subtract`, `multiply`) et usage des registres

rafael.pires@epfl.ch

Merci