1100 Circuits logiques

Portes logiques, circuits combinatoires et séquentiels, additionneurs et registres

Jean Privat

Université du Québec à Montréal

INF2171 - Organisation des ordinateurs et assembleur
v241

Rappel

Niveaux d'un ordinateur (vers le bas)

- Assembleur: pour les humains
- Langage machine: architecture, jeu d'instructions
- Microarchitecture: l'intérieur d'un processeur
- Circuits logiques: algèbre de Boole
- Microélectronique: transistors
- Physique quantique: nombres complexes

Plan

1. Algèbre de Boole
2. Circuits logiques
3. Circuit combinatoire
4. Circuit séquentiel
5. Processeur entier
6. Conclusion

Algèbre de Boole

Algèbre de Boole

Domaines

- Maths, logique, informatique, électronique

2 valeurs

- Vrai, 1, $\top$
- Faux, 0, $\bot$

Opérateurs

- et, and, $\cdot$, $\wedge$, $\times$, `&`, `&&`
- ou, or, $+$, $\vee$, `|`, `||`
- non, not, $\overline{x}$, $\neg x$, x', `~x`, `!x`
- xor, $\oplus$, `^`, `!=`
- Et plein d'autres...

Opérations booléennes

Table de vérité

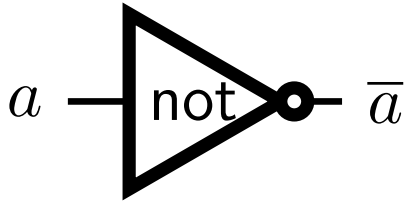
a	b	$\overline{a}$ not	$a+b$ or	$a \cdot b$ and	$a \oplus b$ xor	$\overline{a+b}$ nor	$\overline{a \cdot b}$ nand
0	0	1	0	0	0	1	1
0	1	1	1	0	1	0	1
1	0	0	1	0	1	0	1
1	1	0	1	1	0	0	0

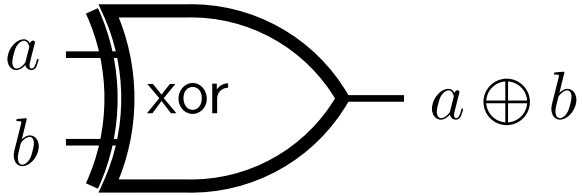
- Les détails en INF1132

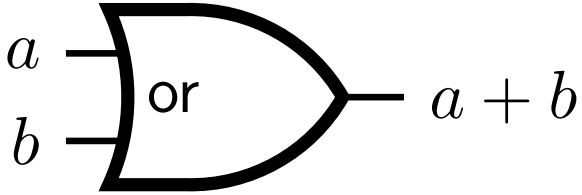
Circuits logiques

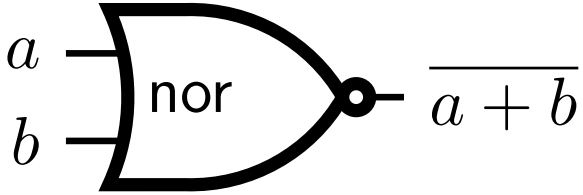
Portes logiques des circuits logiques

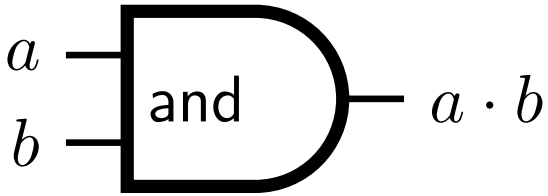
- Circuits logiques : utilise l'algèbre de Boole
  - Fil: 0 ou 1
- Porte logique: élément de base (opérateur de Boole)

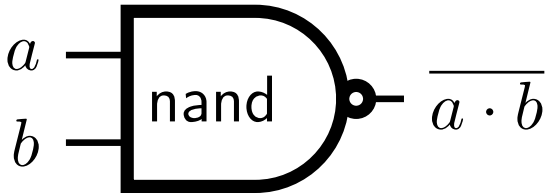












Transistors

- Les circuits logiques sont fait avec des **transistors**
- Le voltage des « fils » code les bits

Types de transistors

- Nombreux détails physiques, électriques et électroniques
- Plusieurs technologies: MOSFET, JFET, BJT, FinFET…
- Plusieurs conceptions de circuits: CMOS, NMOS…

MOSFET et CMOS

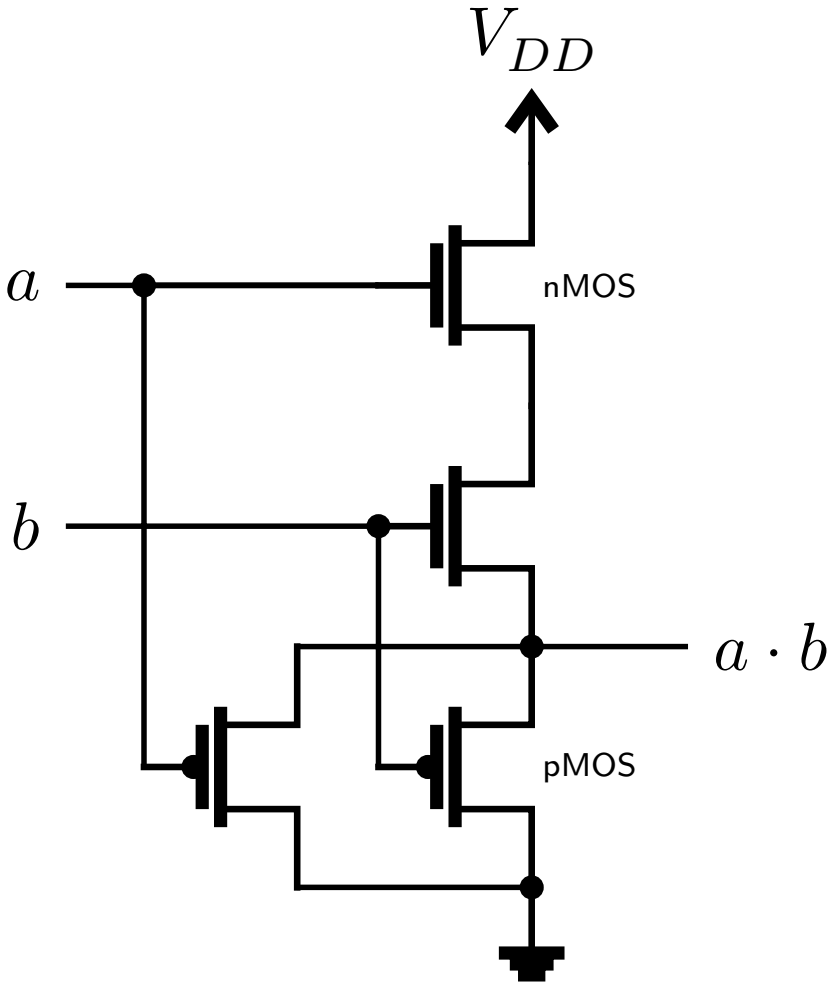
MOSFET

- *Metal-oxide-semiconductor field effect transistors* (1925)
  - Semi-conducteur à base de silicium
- Codage des bits
  - 1 $\rightarrow V_{DD}$, exemple 5V, 3.3V, 1.2V... (D pour *drain*)
  - 0 $\rightarrow$ la terre (*ground*), 0V (parfois notée V_{SS}, S pour *source*)
- Deux types de transistors MOS
  - nMOS: laisse passer le courant si la base est sous tension
  - pMOS: laisse passer le courant si la base n'est pas sous tension

CMOS

- *Complementary MOS* (1963)
- On combine les deux types complémentaires de transistor MOS
- Circuit plus complexes, mais plus économes en énergie
- $\rightarrow$ La majorité actuelle des circuits intégrés

Exemple: porte « et » avec transistors CMOS



Types de circuits logiques

Circuits combinatoires

- Signaux d'entrée
- Signaux de sortie
  dépendent des signaux d'entrée présents
- Pas de mémoire

Circuits séquentiels

- Signaux d'entrée
- Signaux de sortie
  dépendent des signaux d'entrée présents et passés
- L'état du circuit contient une mémoire du passé

Encapsulation et abstraction

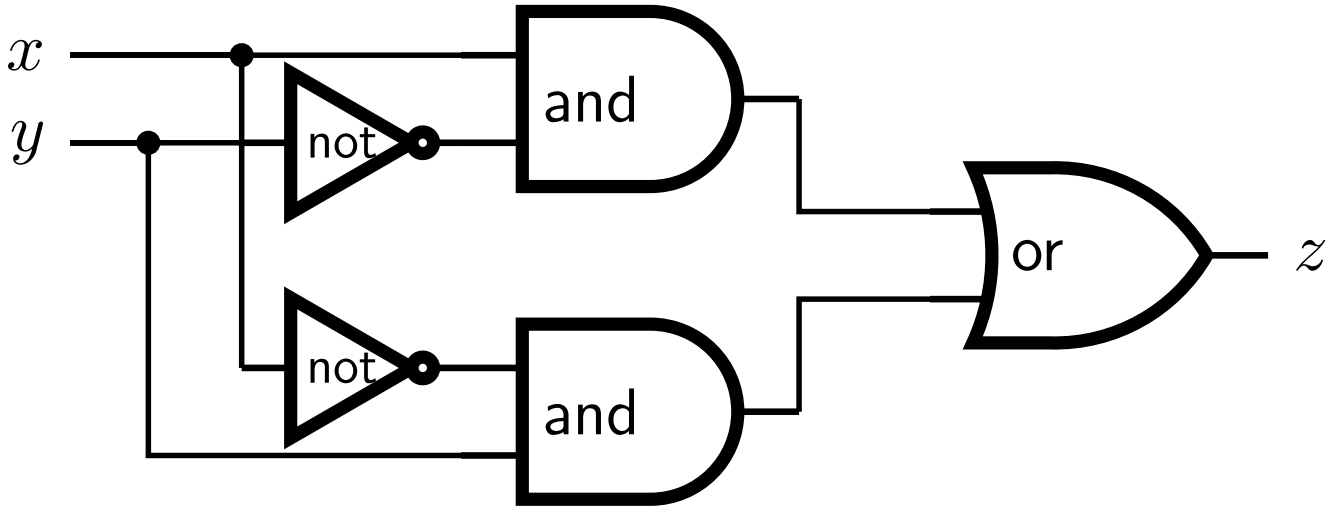
- On fabrique des circuits complexes
- En **réutilisant** et **combinant** des circuits plus simples

Circuit combinatoire

Circuit combinatoire

Exemple

- $z = x \cdot \overline{y} + \overline{x} \cdot y$



Exercice

- Dessiner la table de vérité du circuit

Arithmétique

Représentation des nombre

- Un nombre est représenté par une séquence de bits
- Une opération arithmétique est réalisée par un circuit logique

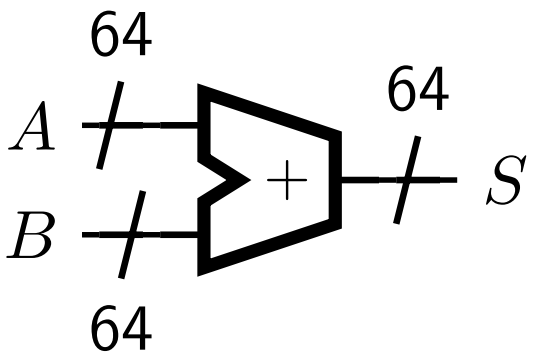
Circuits complexes

- Beaucoup de signaux d'entrée et de sortie (un par bit)
- Opérations de plus en plus complexe : additionneur, multiplicateur, diviseur, flottants IEEE 754, etc.

Addition

Objectif : addition 64 bits

- Entrée : 128 signaux (un par bit)
- Sortie : 64 signaux (un par bit)

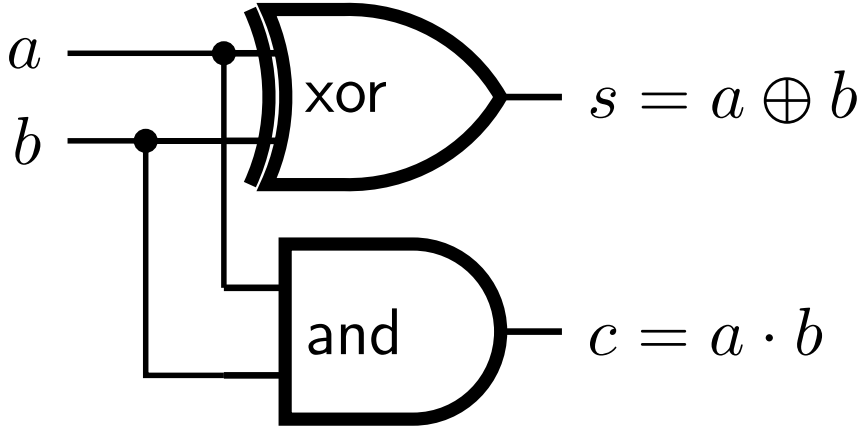


Étape 1 : Demi-additionneur (très simple)

- Deux bits d'entrée : a et b
- Deux bits de sortie : la somme s et la retenue c
- Exercice
  - Dessiner la table de vérité
  - Trouver les formules logique de s et c
  - Dessiner le circuit

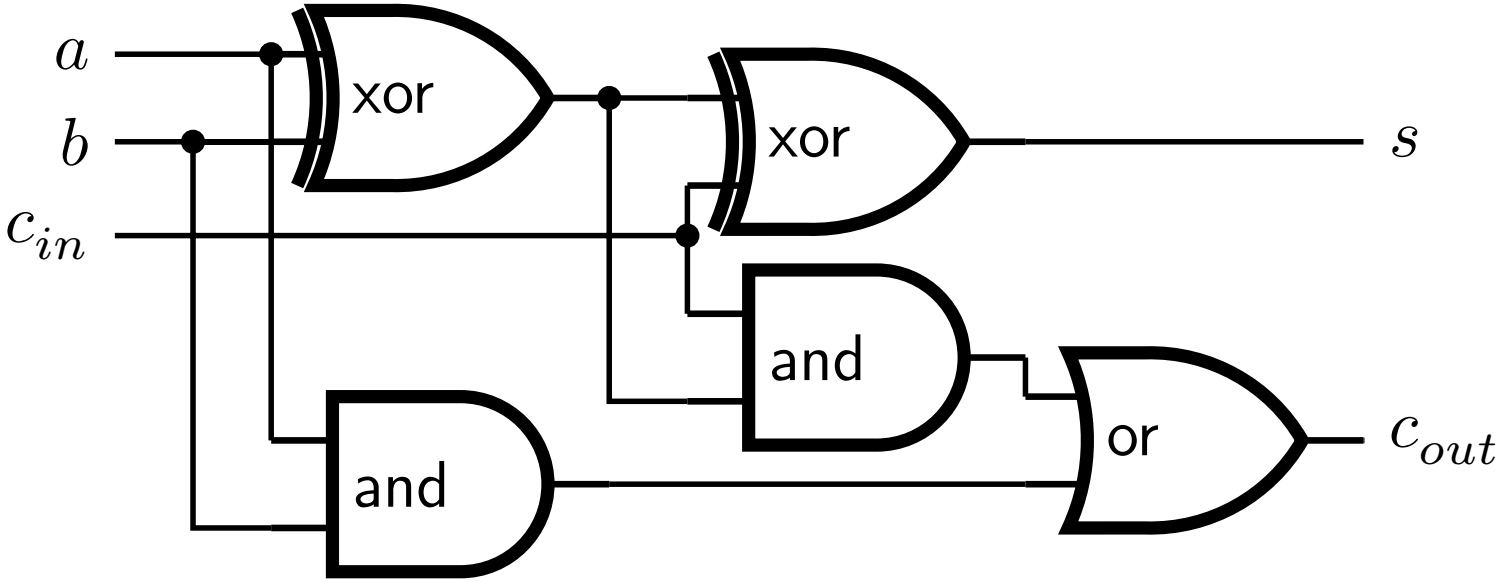
Demi-additionneur (étape 1)

a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



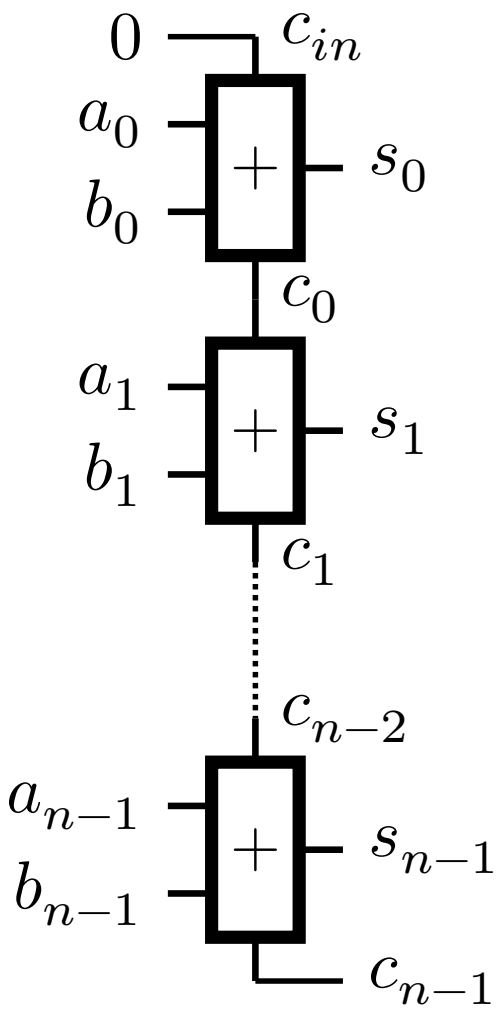
Additionneur complet (étape 2)

- Trois bits d'entrée : a, b et la retenue précédente c_{in}
- Deux bits de sortie : la somme s et la retenue c_{out}
- On combine 2 demi-additionneurs



Additionneur de nombres (étape 3)

- Deux nombres de n bits : A et B
- $2n + 1$ entrées
  - a_0 à a_{n-1}
  - b_0 à b_{n-1}
  - Une retenue initiale c_{in} à 0
- $n + 1$ sorties
  - s_0 à s_{n-1}
  - Une retenue finale c_{n-1}
- On chaîne n additionneurs complets



Dans la vrai vie

- Si beaucoup de bits, le calcul est lent
  - La retenue doit se propager
- Des astuces pour être plus efficace
  - Additionneur parallèle à retenue anticipée

Circuit séquentiel

Circuit séquentiel

Principe

- Le circuit mémorise des trucs
- On utilise des feed-backs :
  - Sorties connectées à des entrées

Utilisation

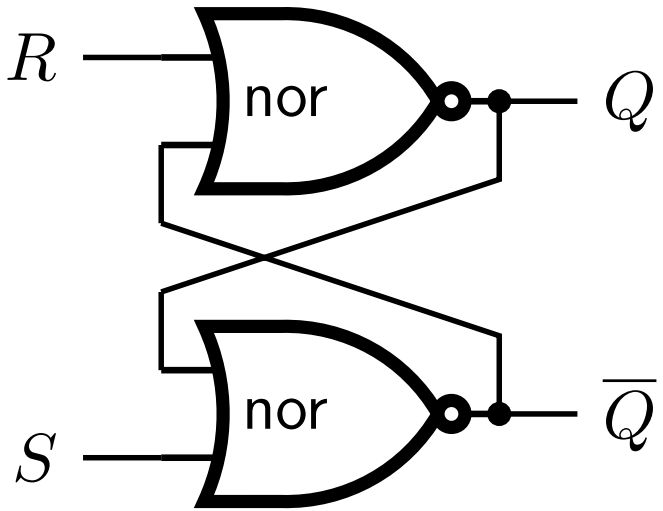
- Mémoire: SRAM, Registres

Pièges

- Temps de propagation et stabilisation
- Besoin de synchronisation : horloge
- Valeurs interdites : résultats indéfinis

Bascule asynchrone SR

- Ou verrou SR, ou *SR latch*
- Verrou (*latch*) = Asynchrone = pas d'horloge
- La sortie Q maintient la dernière valeur ; $\overline{Q}$ est sa négation
- Le bit « set » S met Q à 1 ; le bit « reset » R met Q à 0



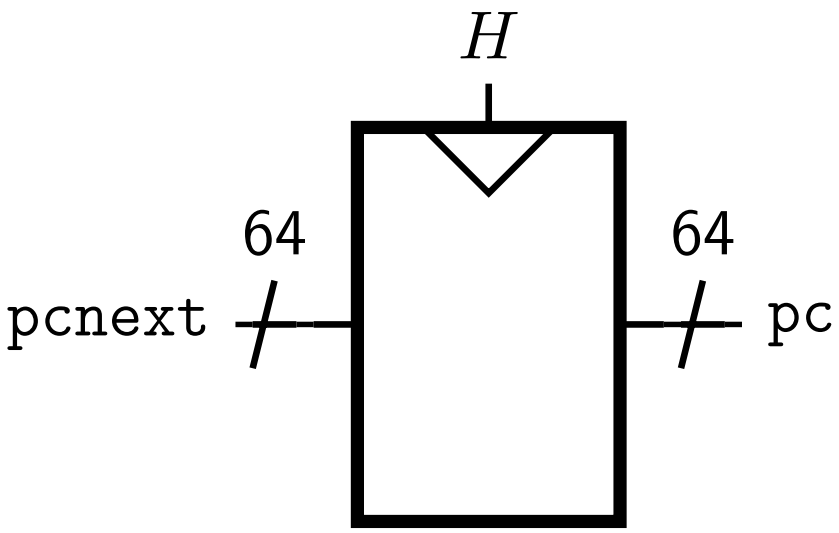
Exercice SR

- On commence avec $S = R = Q = 0$ et $\overline{Q} = 1$
- On met S à 1... S à 0... R à 1... R à 0...
- On met S et R a 1 en même temps... puis à 0 en même temps...

Registre

Objectif : registre compteur ordinal 64 bits

- Stocke (et maintient) `pc`
  - Valeur binaire 64 bits
- Accepte `pcnext`
  - Nouvelle valeur 64 bits
- Synchronisé avec une horloge
  - Nouvelle instruction à chaque cycle

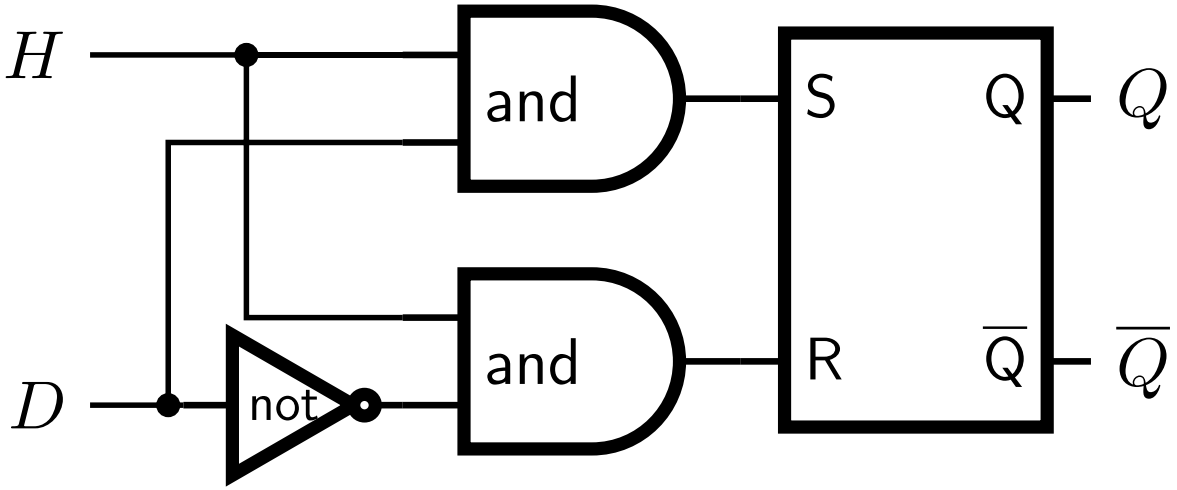


Étape 1 : bascule asynchrone SR

- Déjà fait !
- On stocke (et maintient) 1 bit (Q)
- On peut modifier ce bit avec S et R

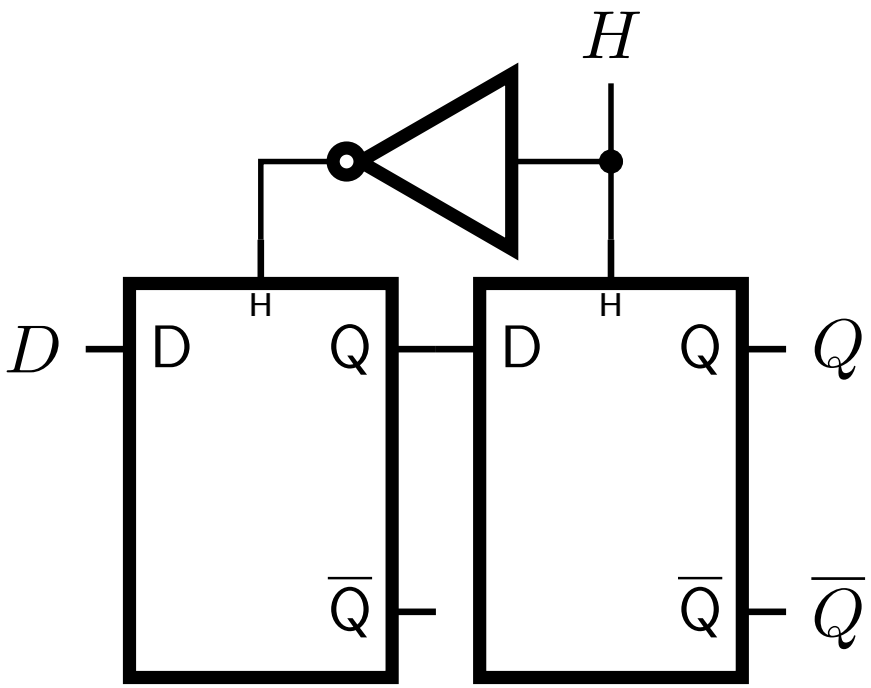
Bascule asynchrone D (étape 2)

- Ou verrou D, ou *D latch*
- Un bit « data » D et un bit H de contrôle (ou *CLK*)
- Quand $H = 0$, Q ne change pas (D est ignoré)
- Quand $H = 1$, Q prend la valeur de D
- Avantage sur SR: plus pratique, pas de valeurs interdites



Bascule synchrone D (étape 3)

- Ou juste bascule D, ou *D flip-flop*
- H est une horloge, avec un signal régulier 0→1→0→1...
- D est copié dans Q seulement quand H passe de 0 à 1
- → On parle de **front d'horloge** (*rising edge of the clock*)
- *flip-flop* = synchrone = front d'horloge



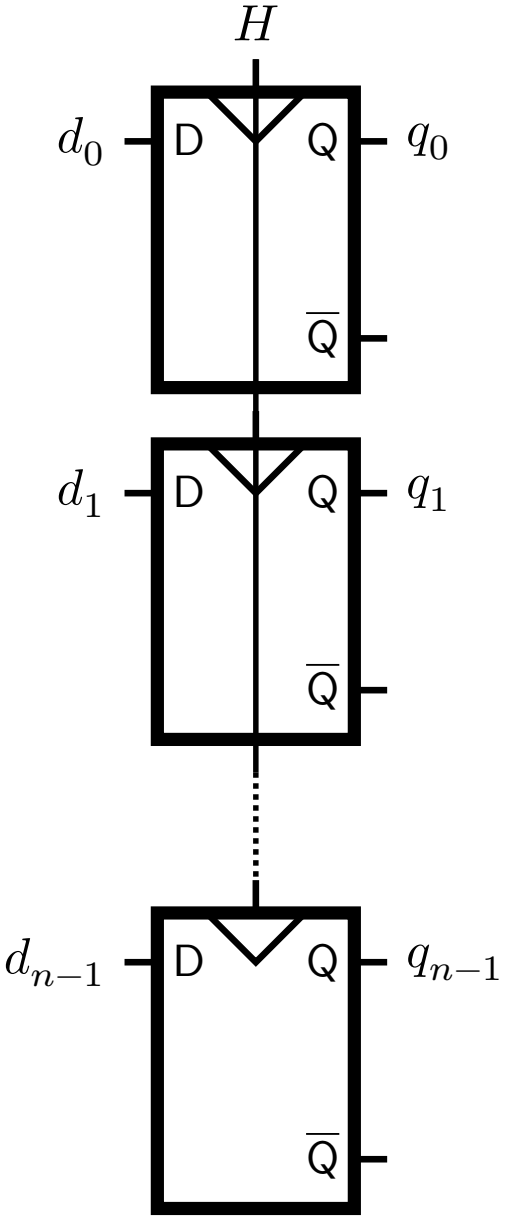
Registre (étape 4)

Rappel de l'objectif

- Maintenir la valeur de n bits
- Permettre de changer la valeur du registre (de temps en temps)

Mise en œuvre

- On combine n bascules D
- Une par bit à mémoriser
- L'horloge est partagée



Processeur entier

Processeur

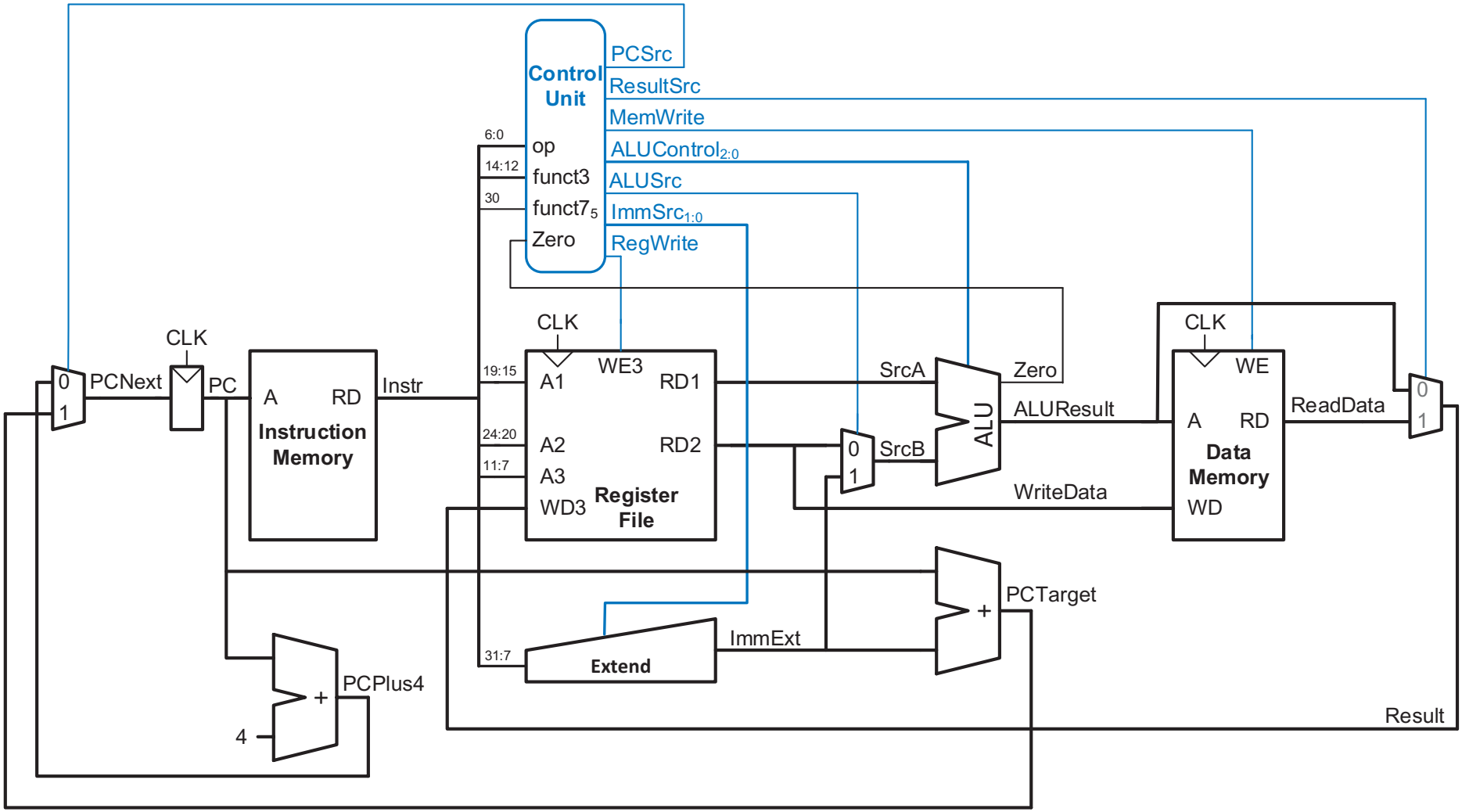
Milliards de transistors

- Intel 8086 (1978) : 29 000 transistors
- Apple M2 Ultra (2023) : 82 000 000 000 transistors
- Loi de Moore (1965)
  - Doublement du nombre de transistors d'un microprocesseur tous les deux ans
  - Gordon E. Moore (un des fondateurs d'Intel)

Les détails

- Dans INF4170 - Architecture des ordinateurs

Processeur RISC-V à un cycle



Source: Harris, D et Harris, S. *Digital Design and Computer Architecture RISC-V Edition* (2021). Fig. 7.12, p. 407

Conclusion

Résumé

Vers le haut

- Grâce aux mathématiques et à l'électronique
  - On peut construire des circuits complexes
  - Qui traitent et mémorisent des bits
- Grâce à ces circuits
  - On peut construire des microprocesseurs
  - Qui offrent des jeux complexes d'instructions et de registres
- Grâce à ces instructions et registres
  - On peut écrire des programmes
  - Et les faire s'exécuter sur un ordinateur

Conclusion du cours

- 1- Tout n'est que des bits
- 2- Il n'y a pas de magie

- « *Any sufficiently advanced technology is indistinguishable from magic* » — Arthur C. Clarke, *Profiles of the Future: An Inquiry into the Limits of the Possible* (1962)
- « *Any technology distinguishable from magic is insufficiently advanced* » — Barry Gehm (1991)