

# Fonctionnement d'un ordinateur

M. Dubacq

S1D 2009

## 1 Histoire des ordinateurs

**Objectif :** *Savoir préparer un exposé, connaître l'histoire de l'informatique.*

Préparez un petit rapport (2 à 4 pages, 4000 à 5000 signes) sur l'histoire des débuts de l'ordinateur. Sujets proposés : les bouliers, la règle à calcul, les machines à engrenages, les machines à compter, la calculatrice mécanique, le métier à tisser, la machine de Babbage, le Z3, la Bombe de Turing, l'ENIAC, la machine de Harvard, le PDP1, l'Apple II, le ZX81, le Commodore 64.

Vous pourrez présenter l'objet de l'étude, le mécanisme de fonctionnement, les personnages historiques qui y ont contribué, la réussite commerciale de l'objet ou pas, des exemples de programme ou d'algorithmes qui sont faits avec.

Les étudiants s'attarderont à respecter les règles élémentaires : mention des sources (images, documents, textes), liste de références bibliographiques, etc.

## 2 Démontage d'une machine

**Objectif :** *Reconnaître les différents composants d'une machine, connaître les précautions élémentaires.*

L'enseignant va vous apporter des (vieilles) machines. Sous sa direction, démontez-les, puis remontez-les.

Reconnaissez les éléments mentionnés dans le cours. Vous pourrez retirer le ventilateur de son emplacement pour voir le processeur (et la pâte thermique qui les joint). Démontez une barrette mémoire et remettez-la en place. Démontez une carte d'extension, repérez sur une carte mère les condensateurs aux points stratégiques qui servent à stabiliser le courant des pièces plus délicates. Démontez un disque dur, remettez-le en place (en repérant les connecteurs Molex).

Les précautions à connaître : pas de cheveux longs non attachés (comme en cuisine) ou de pendentifs et autres objets divers qui pourraient être pris dans un ventilateur, pas de pulls en laine ou autres vêtements faisant de l'électricité statique. La plupart des machines se défont avec un simple tournevis ou simplement avec les doigts.

## 3 Jeu du loup, de la chèvre et du chou

**Objectif :** *Comprendre une modélisation à l'aide d'un automate.*

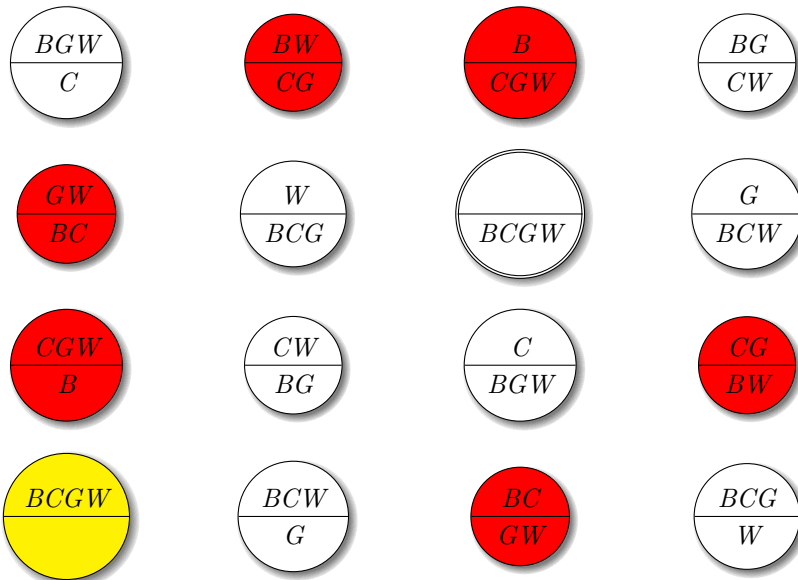
Le jeu est le suivant : sur la rive d'un cours d'eau, il y a une Barque, Walter le loup, Gontran la chèvre et une Caisse de choux. Vous devez faire traverser les trois, sachant que vous ne pouvez prendre qu'un seul élément à la fois dans la barque, le but étant de faire traverser tout le monde (barque y compris). Mais si on laisse sur une même rive (où vous n'êtes pas) le loup et la chèvre ou la chèvre et la caisse de choux, on a perdu (Walter attaquerait immédiatement Gontran, et Gontran attaquerait immédiatement la caisse de choux).

1. Est-ce que la barque peut être à un endroit où vous n'êtes pas ? Combien d'états sont possibles pour le problème ?

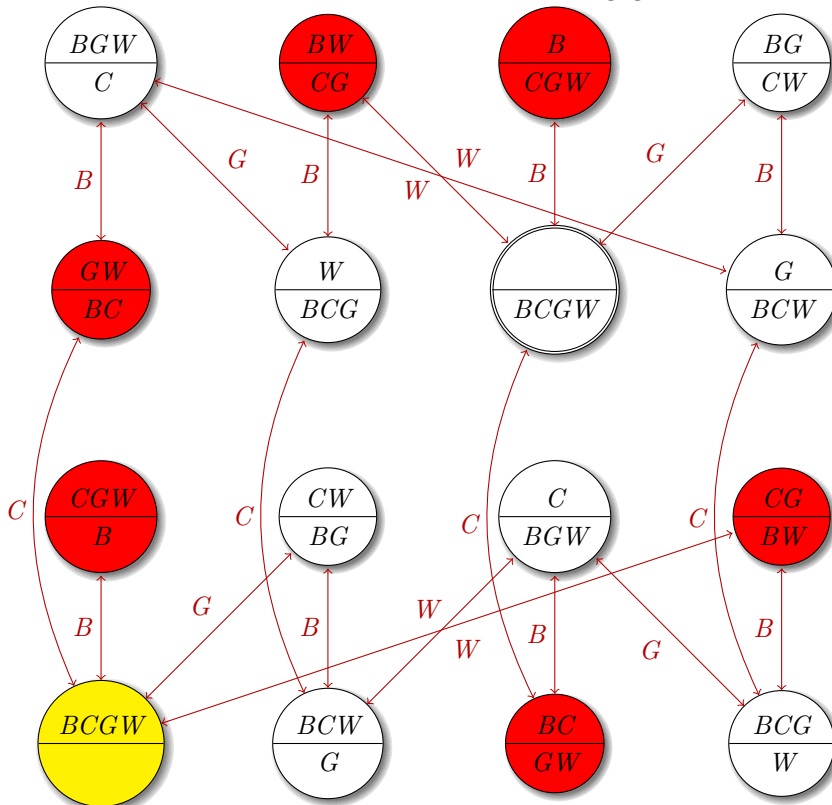
*Évidemment non (la caisse de choux fait la grève, le loup est trop fainéant pour le faire, et la chèvre refuse de toucher des rames en bois). Donc il y a quatre variables indépendantes ayant chacune 2 valeurs possibles (rive de la barque, rive du loup, rive de la chèvre, rive de la caisse), soit 16 états possibles.*

*On utilisera les lettres B, C, G et W pour représenter le bateau, la caisse, Gontran la chèvre et Walter le loup.*

2. Représentez les états possibles. Quel est l'état initial ? Repérez les états qui mènent à une partie perdue.



- On étiquettera les transitions par une lettre BCGW, B représentant un voyage à vide et CGW représentant le voyage avec le passager correspondant dans la barque. Vérifiez qu'il n'y a pas d'ambiguïté dans les transitions (qu'on a jamais une même étiquette vers deux états différents). *C'est vrai*
- Faites toutes les transitions. Existe-t-il une solution gagnante ? Donnez-la.

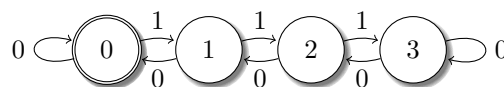


*Il existe donc une solution (et même plusieurs) par exemple : BCGW/CW/BCW/W/BGW/G/BG/fini.*

#### 4 Réalisation d'un incrémenteur

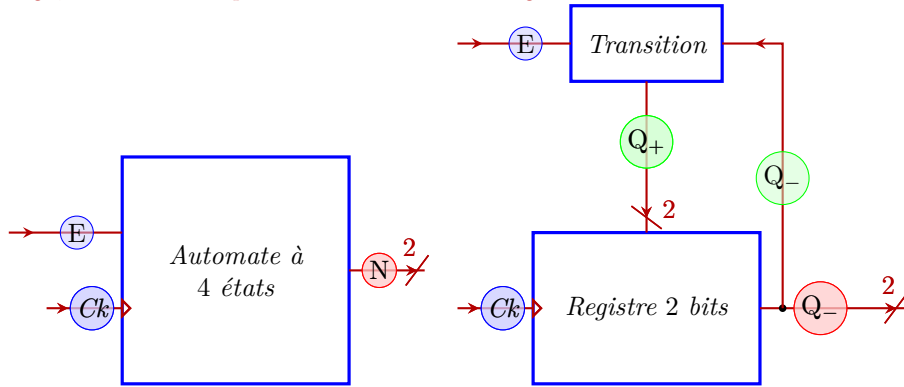
**Objectif :** Écrire une table de vérité, passer d'une table de vérité à un système d'équations, passer d'un système d'équations à une représentation graphique.

Soit le graphe d'états d'un automate fini suivant :



- On veut que le circuit change sur le front descendant de l'horloge. Faites le schéma-bloc d'un circuit réalisant cet automate; montrez comment on peut le réaliser avec des circuits séquentiels et combinatoires.

1 entrée, 1 horloge, 2 sorties. On peut le réaliser avec un registre à 2 bits et un circuit combinatoire (voir cours).



2. Faites la table de transition de cet automate.

$Q_1^-$	$Q_0^-$	I	$Q_1^+$	$Q_0^+$
0	0	1	0	1
0	0	0	0	0
0	1	1	1	0
0	1	0	0	0
1	0	1	1	1
1	0	0	0	1
1	1	1	1	1
1	1	0	1	0

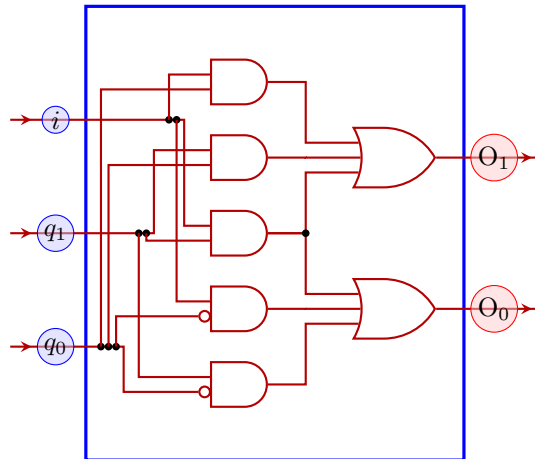
3. Faites le schéma éclaté du circuit combinatoire aidant à réaliser cet automate.

La table de vérité du circuit combinatoire : même chose que la table de transition, mais l'entrée I est une entrée et non plus une étiquette de transition. J'appelle  $O_0$  et  $O_1$  les sorties.

Équations des sorties à écrire et à simplifier :

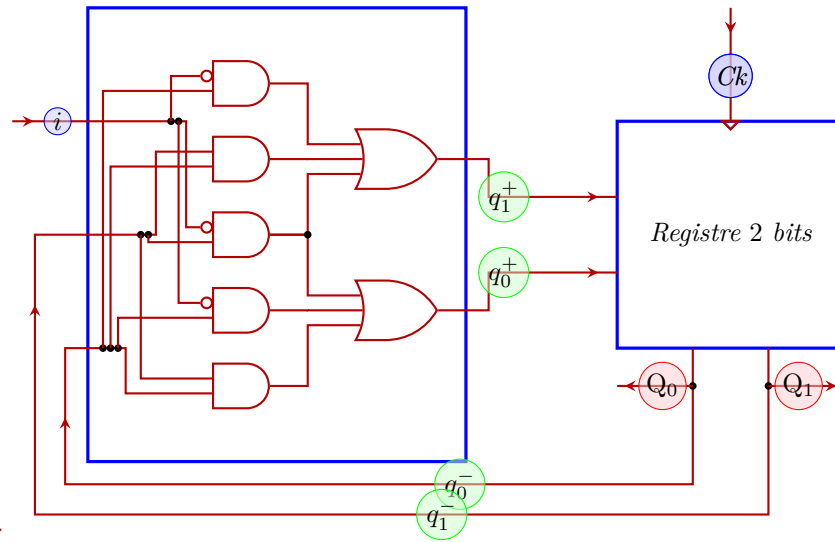
$$O_0 = q_1 \bar{q}_0 + q_1 \bar{I} + \bar{q}_0 \bar{I}$$

$$O_1 = q_1 q_0 + q_1 \bar{I} + q_0 \bar{I}$$



Soit le schéma suivant :

4. Completez votre schéma avec un ou plusieurs circuits séquentiels pour obtenir le schéma d'un circuit qui fait l'incrémenteur.



Soit le schéma suivant :

## 5 La machine à accumulateur

**Objectif :** Comprendre le fonctionnement d'une machine élémentaire, comprendre comment faire un programme complexe avec des instructions élémentaires.

Voici le jeu complet des instructions de la machine à accumulateur décrite en cours (OP représente l'opérande) :

000	ADDI	ACCU ← ACCU + OP, PC ← PC + 1
001	ADD	ACCU ← ACCU + MEM[OP], PC ← PC + 1
010	SET	ACCU ← OP, PC ← PC + 1
011	LOAD	ACCU ← MEM[OP], PC ← PC + 1
100	GOTO/STOP	PC ← OP
101	SUB	ACCU ← ACCU - MEM[OP], PC ← PC + 1
110	SAVE	MEM[OP] ← ACCU, PC ← PC + 1
111	NUL ?	Si ACCU = 0, PC ← OP, sinon PC ← PC + 1

La machine s'arrête si l'instruction est un

GOTO qui ne change pas PC. Elle imprime alors tout le contenu de sa mémoire.

1. Traduisez en instructions les 7 premières cases mémoires dans le programme suivant (PC=00000 au départ) :

Adr	Adr+0	Adr+1	Adr+2	Adr+3
0	01000000	00110100	00110101	00000010
4	10110110	11010000	10000110	00000000
8	00000000	00000000	00000000	00000000
12	00000000	00000000	00000000	00000000
16	00000000	00000000	00000000	00000000
20	10011001	01001100	00011000	00000000
24	00000000	00000000	00000000	00000000
28	00000000	00000000	00000000	00000000

Le programme se lit :

0 : SET 00000

1 : ADD 20

2 : ADD 21

3 : ADDI 2

4 : SUB 22

5 : SAVE 16

6 : GOTO 6 (=STOP)

2. Représentez à chaque cycle d'instruction les valeurs de PC et de ACCU, ainsi que les cases mémoires qui ont changées.

Temps	PC	ACCU	MEM
0	00000	????	—
1	00001	00000000	—
2	00002	10011001	—
3	00003	11100101	—
4	00004	11100111	—
5	00005	11001111	—
6	00006	11001111	MEM[16]←11001111
7	STOP	—	—

3. Que fait donc ce programme ?

Avec 20 qui contient 0x99, 21 qui contient 0x4C et 22 qui contient 0x18. Le calcul consiste à mettre dans la case mémoire numéro 16 la somme (MEM[20]+MEM[21]+2-MEM[22]), c'est-à-dire la valeur 0x83+0x4C, soit 0xCF (soit 207 en décimal).

4. Voici le listing d'un programme pour cette machine : que fait-il ?

```

0: LOAD 22          8: GOTO 8 (=STOP)
1: ADD 20           ...
2: SAVE 22         20: un nombre
3: LOAD 21         21: un autre nombre
4: SUB 23          22: 00000000
5: SAVE 21         23: 00000001
6: NUL? 8         ...
7: GOTO 0
    
```

Ce programme multiplie le premier nombre (adresse 20) par le deuxième (adresse 21). En fait, il fait  $MEM[22] \leftarrow MEM[22] + MEM[21]$ ,  $MEM[21] \leftarrow MEM[21] - MEM[23] = MEM[21] - 1$ , et si  $ACCU = MEM[21]$  est nul, il s'arrête, sinon il recommence au début. Soit : Tant que  $B \neq 0$ , faire  $\{ C = C + A, B = B - 1 \}$ . À la fin, C contient bien  $B \times A$ .

## 6 Modélisation d'un feu rouge

**Objectif :** Comprendre une modélisation à l'aide d'un automate.

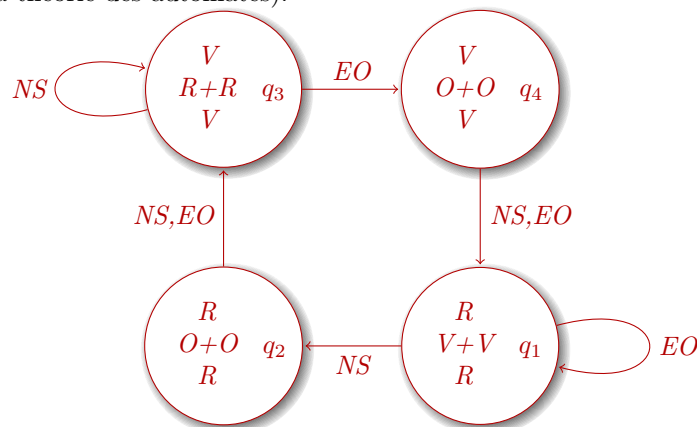
On veut modéliser à l'aide d'un automate les feux de circulation à un croisement de deux routes. Ces feux sont habituellement commandés par une minuterie (qui en fonction du trafic ou autre dit quelle route doit circuler et quelle autre doit être arrêtée ; parfois, un gendarme peut intervenir manuellement pour contrôler la minuterie).

Un système de contrôle du carrefour reçoit les instructions de la minuterie et doit s'assurer de lui obéir mais en respectant des règles et délais minimaux :

- Les feux Nord et Sud sont tout le temps identiques, les feux Est et Ouest de même.
- Si un feu est à l'orange, il doit forcément passer au rouge après (même si le gendarme fait n'importe quoi).
- Si la minuterie demande à ce qu'un sens circule et qu'il ne circule pas, il faut d'abord mettre l'autre feu au rouge.
- Dans un premier temps, une direction passe au vert dès que l'autre passe au rouge.

Le système de contrôle ne lit les instructions de la minuterie que toutes les dix secondes : ça donne assez de souplesse dans les rapports entre les durées des deux feux (par exemple à certaines heures 80 secondes pour un et 60 pour l'autre, et de nuit 160 secondes pour un et 30 secondes pour l'autre) ; c'est également la durée d'un feu orange.

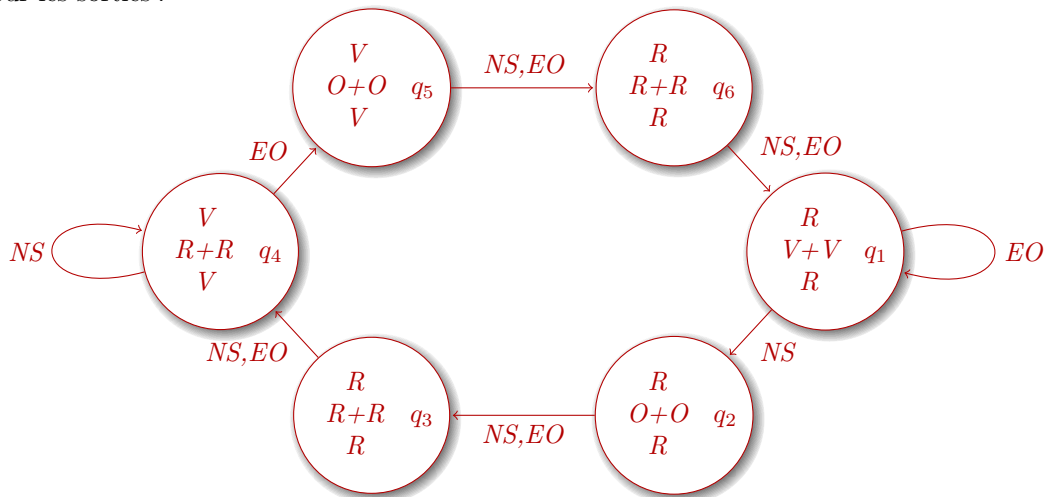
1. Modélisez ce problème par un automate fini complet déterministe. Rappelez la définition de chacun de ces mots (dans le contexte de la théorie des automates).



Sur les transitions: NS=0, EO=1.

Un automate fini a un nombre fini d'états. Un automate fini complet a une transition pour toute entrée possible. Un automate fini déterministe a une seule transition possible pour une entrée donnée.

2. Pour plus de sécurité, on veut qu'entre chaque changement de direction, il y ait une période (dix secondes) où les voitures ne circulent dans aucun sens (tous les feux sont rouges). Combien d'états doit-on avoir ? Que doit-on faire pour les sorties ?



*On remarque que deux états sont en apparence identique (rouge partout) mais ils sont en fait différents, avec des sorties qui sont rendues identiques. Dans beaucoup de cas, on a identité entre la sortie d'un automate et ses états internes, mais là ce n'est pas le cas (les représentations d'états sont moins nombreuses que les états). C'est le cas avec un ordinateur : c'est un automate fini (avec énormément d'états) mais ce qu'on en voit ne correspond qu'à une partie de ces états (il n'y a pas une indication visuelle de chaque bit de mémoire de l'ordinateur, seulement d'une toute partie qui est la mémoire écran).*