

3 Le fonctionnement d'un ordinateur

Peter Schlagheck

Université de Liège

Ces notes ont pour seule vocation d'être utilisées par les étudiants dans le cadre de leur cursus au sein de l'Université de Liège. Aucun autre usage ni diffusion n'est autorisé, sous peine de constituer une violation de la Loi du 30 juin 1994 relative au droit d'auteur.

3 Le fonctionnement d'un ordinateur

3.1 La machine de Turing

3.2 Le fonctionnement d'un processeur

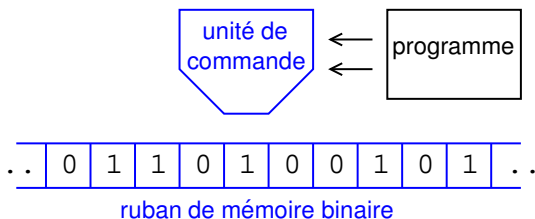
3.3 La réalisation en pratique

3.1 La machine de Turing

Alan Turing (1912 - 1954), mathématicien

- grande contribution au déchiffrement des codes allemands “Enigma” pendant la deuxième guerre mondiale (ordinateur électromécanique “bombe”, 1940)
 - conception de la **machine de Turing** (1936)
= une machine de calcul universelle qui sait effectuer tous les algorithmes numériques possibles
- ... afin de démontrer qu'il n'est pas possible de déterminer si de tels algorithmes s'arrêtent ou non

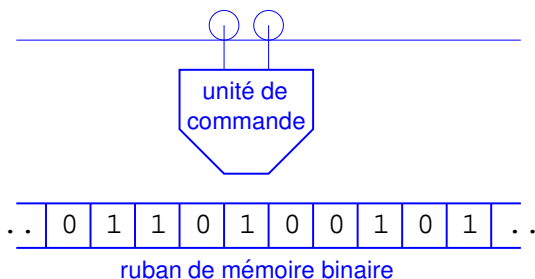
La machine de Turing



La machine de Turing est constitué

- ▶ d'un ruban de mémoire binaire infini et
- ▶ d'une unité de commande chargée d'un programme

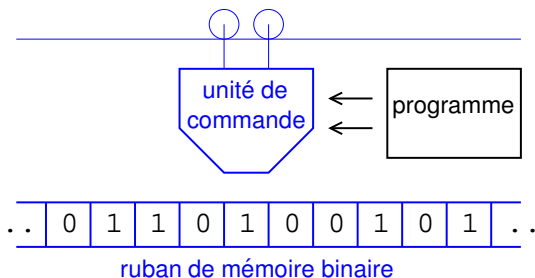
La machine de Turing



L'unité de commande peut

- ▶ lire le contenu (0 ou 1) de la cellule actuelle du ruban
- ▶ écrire 0 ou 1 dans la cellule actuelle du ruban
- ▶ se déplacer à gauche ou à droite par une cellule

La machine de Turing



Le programme définit plusieurs “états” de l’unité de commande.
Chaque état se compose

- ▶ d’une lecture et écriture dans la cellule actuelle
- ▶ d’un déplacement à gauche ou à droite
- ▶ d’un lien/saut à un autre état à exécuter à la suite

... tout dépendant du contenu de la cellule actuelle

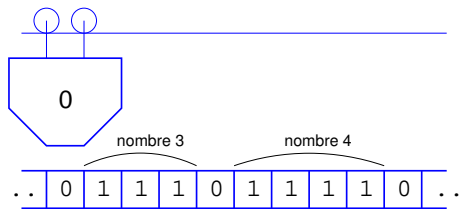
La machine de Turing

Exemple d'un programme (additions) pour la machine de Turing:

état actuel	cellule actuelle		étape prochaine	
	lecture	écriture	déplacement	état suivant
état 0	0	0	droite	état 0
	1	1	droite	état 1
état 1	0	1	droite	état 2
	1	1	droite	état 1
état 2	0	0	gauche	état 3
	1	1	droite	état 2
état 3	0	0	droite	STOP
	1	0	droite	STOP

La machine de Turing

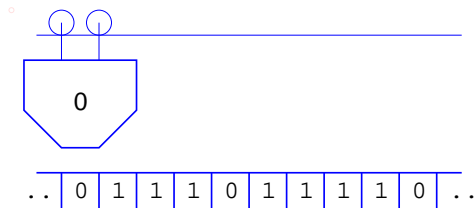
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

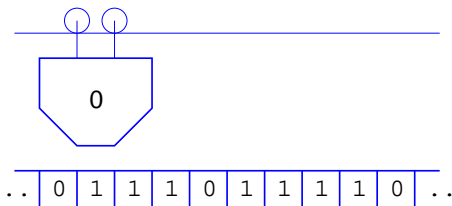
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

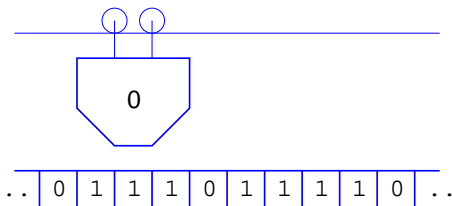
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

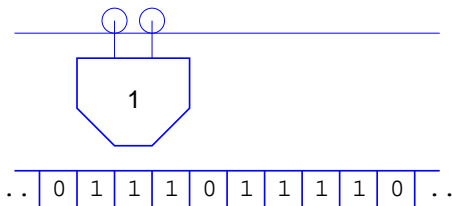
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

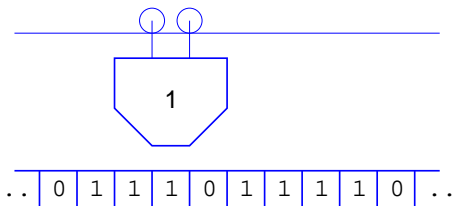
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

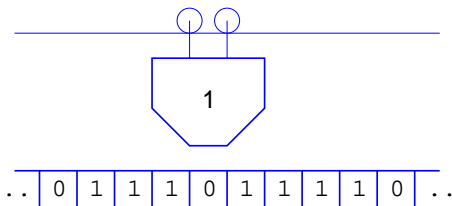
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

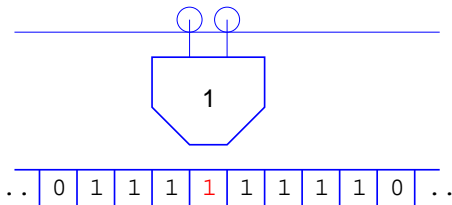
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

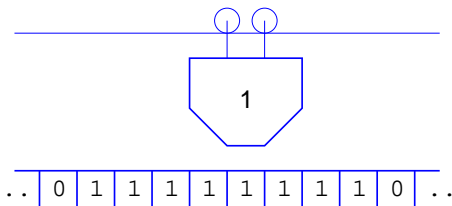
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

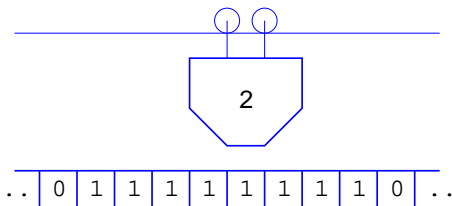
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

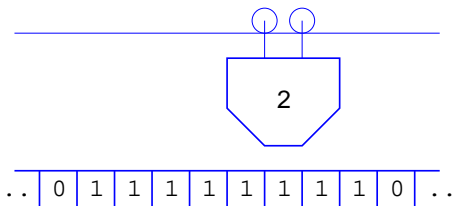
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

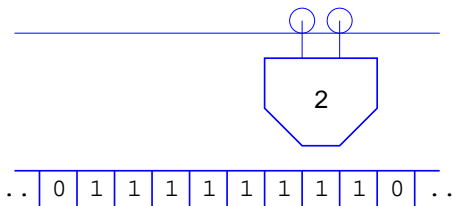
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

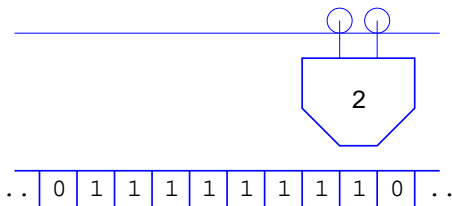
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

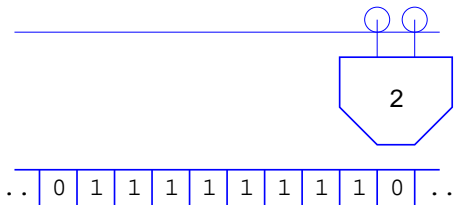
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

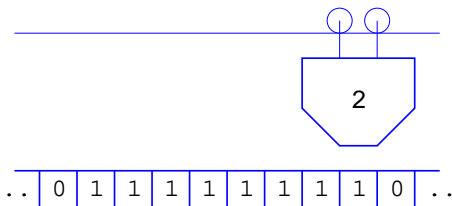
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

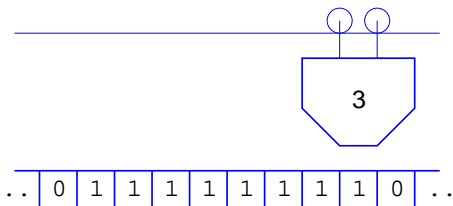
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

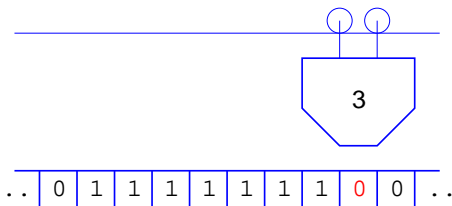
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

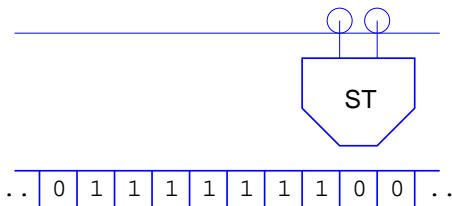
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

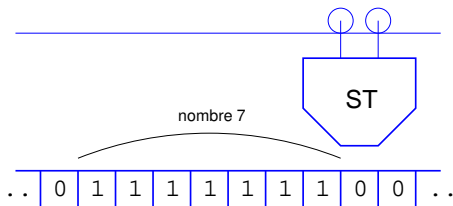
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

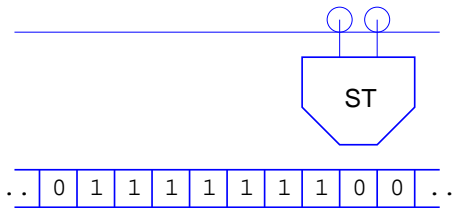
Additions avec la machine de Turing:



0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

La machine de Turing

Additions avec la machine de Turing:



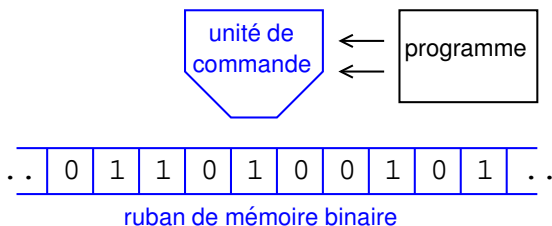
0	0	0	→	0
	1	1	→	1
1	0	1	→	2
	1	1	→	1
2	0	0	←	3
	1	1	→	2
3	0	0	→	ST
	1	0	→	ST

→ tous les algorithmes numériques peuvent être représentés sur une telle machine (thèse de Church-Turing)

(ce n'est pas forcément la meilleure façon de les exécuter ...)

Comment est-ce qu'on pourrait faire mieux

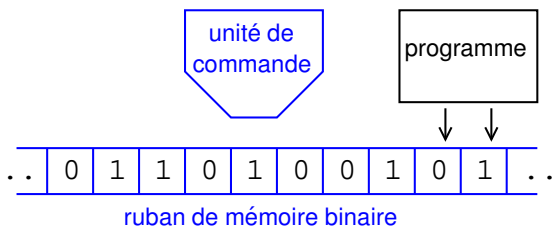
... après tout ce qu'on a déjà appris ?



- remplacer tous les bits par des octets
- déplacement arbitraire de l'unité de commande
 - nécessite l'adressage global des cellules du ruban

Comment est-ce qu'on pourrait faire mieux

... après tout ce qu'on a déjà appris ?



- remplacer tous les bits par des octets
- déplacement arbitraire de l'unité de commande
- effectuer des calculs élémentaires (additions des octets) à l'intérieur de l'unité de commande
- encoder le programme dans la mémoire

3.2 Le fonctionnement d'un processeur

John von Neumann (1903 - 1957), mathématicien

Contributions:

- ▶ théorie de logique
- ▶ physique quantique
- ▶ théorie des jeux
- ▶ Manhattan project
- ▶ informatique
→ "l'architecture de von Neumann"



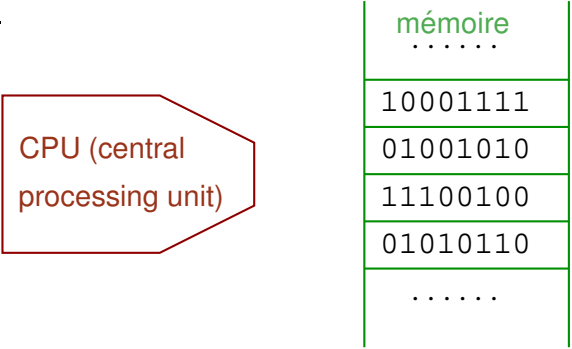
3.2 Le fonctionnement d'un processeur

Le principe de von Neumann:

Un ordinateur est constitué

- ▶ d'une unité de commande,
- ▶ d'une unité arithmétique et logique,
- ▶ d'une mémoire centrale et
- ▶ des unités d'entrée et de sortie
(clavier, souris, écran, imprimante, ...)

Le fonctionnement d'un processeur primitif



The diagram illustrates a primitive processor. On the left, a red-outlined pentagon represents the CPU, labeled "CPU (central processing unit)". On the right, a vertical green-outlined structure represents memory, labeled "mémoire" in green. The memory is divided into five horizontal sections. The first section contains an ellipsis "...". The second section contains the binary string "10001111". The third section contains "01001010". The fourth section contains "11100100". The fifth section contains "01010110". Below the fifth section is another ellipsis "...".

CPU (central
processing unit)

mémoire

.....

10001111

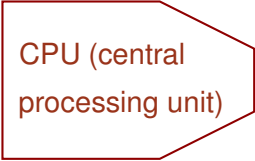
01001010

11100100

01010110

.....

Le fonctionnement d'un processeur primitif



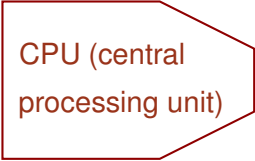
CPU (central
processing unit)

31	10001101

16	01001010
15	11100100

0	01100101

Le fonctionnement d'un processeur primitif



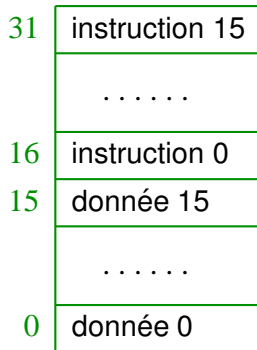
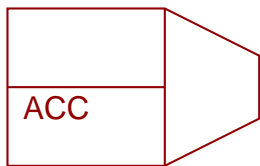
CPU (central
processing unit)

31	instruction 15

16	instruction 0
15	donnée 15

0	donnée 0

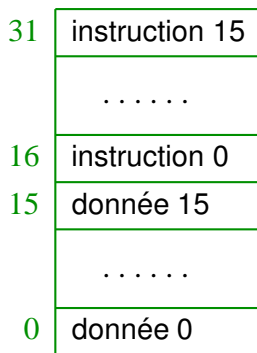
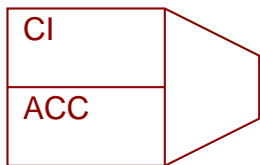
Le fonctionnement d'un processeur primitif



ACC: accumulateur (1 octet)

→ sauvegarde d'une donnée pour des opérations arithmétiques et logiques

Le fonctionnement d'un processeur primitif

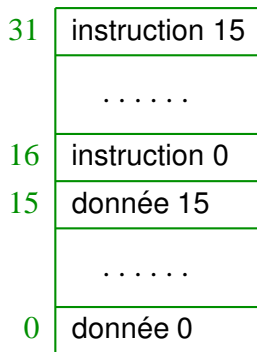
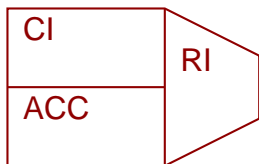


ACC: accumulateur (1 octet)

CI: compteur d'instructions (1 octet)

→ contient l'adresse de l'instruction actuelle

Le fonctionnement d'un processeur primitif

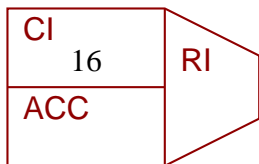


ACC: accumulateur (1 octet)

CI: compteur d'instructions (1 octet)

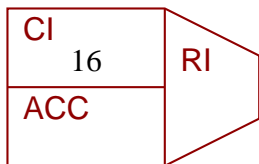
RI: registre d'instruction (1 octet)

Le fonctionnement d'un processeur primitif



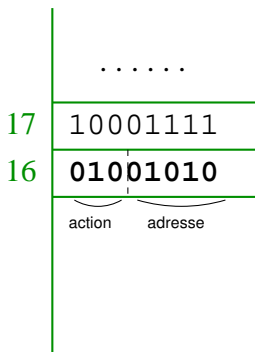
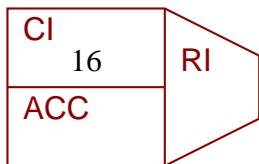
17	10001111
16	01001010
15	11100100

Le fonctionnement d'un processeur primitif

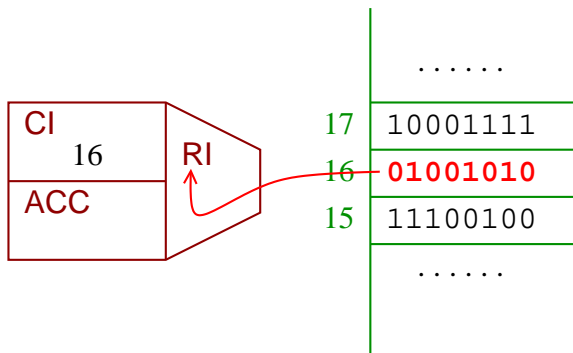


17	10001111
16	01001010
15	11100100

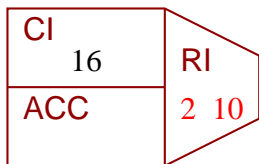
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

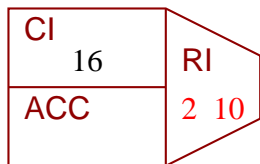


Le fonctionnement d'un processeur primitif



17	10001111
16	01001010
15	11100100

Le fonctionnement d'un processeur primitif

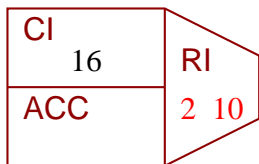


17	10001111
16	01001010
15	11100100

Opération 2:

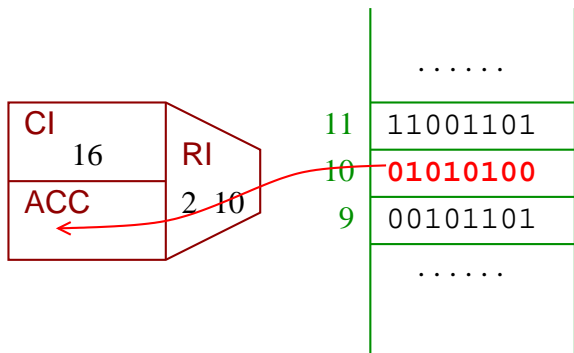
copier le contenu à l'adresse indiquée dans l'accumulateur

Le fonctionnement d'un processeur primitif

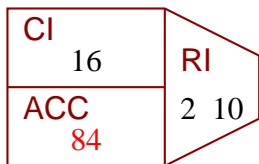


11	11001101
10	01010100
9	00101101

Le fonctionnement d'un processeur primitif

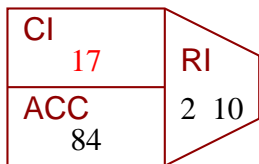


Le fonctionnement d'un processeur primitif



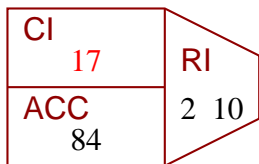
11	11001101
10	01010100
9	00101101

Le fonctionnement d'un processeur primitif



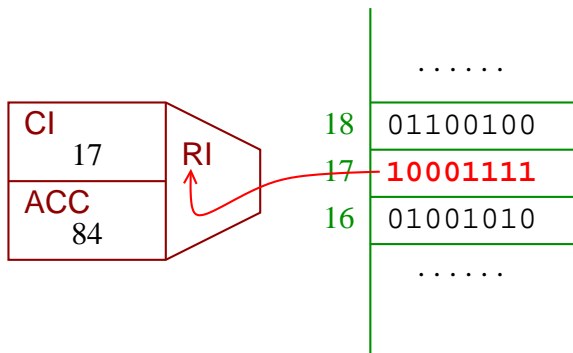
11	11001101
10	01010100
9	00101101

Le fonctionnement d'un processeur primitif

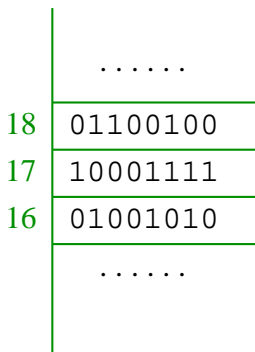
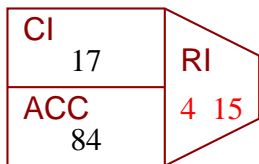


18	01100100
17	10001111
16	01001010

Le fonctionnement d'un processeur primitif



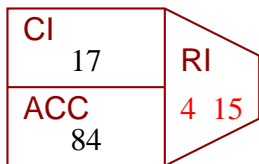
Le fonctionnement d'un processeur primitif



Opération 4:

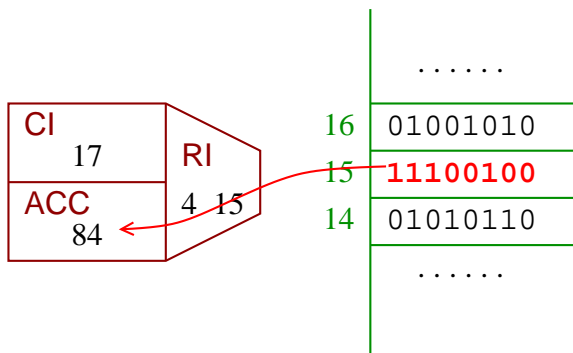
ajouter à l'accumulateur le contenu dans l'adresse indiquée

Le fonctionnement d'un processeur primitif

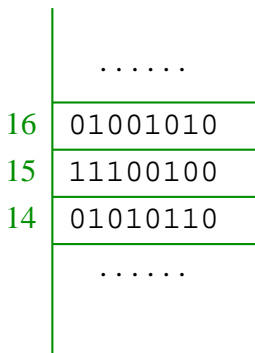
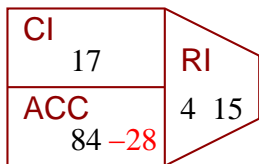


16	01001010
15	11100100
14	01010110

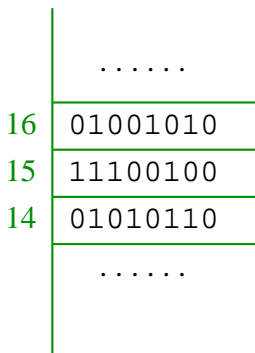
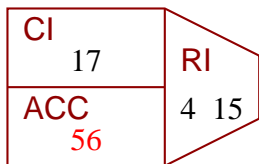
Le fonctionnement d'un processeur primitif



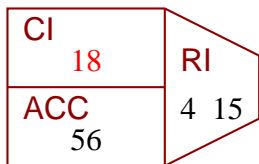
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

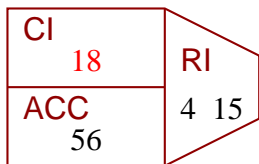


Le fonctionnement d'un processeur primitif



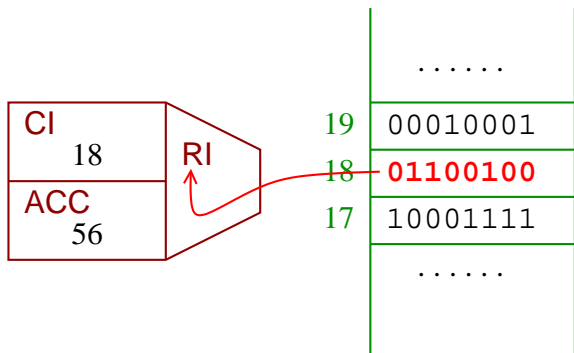
16	01001010
15	11100100
14	01010110

Le fonctionnement d'un processeur primitif

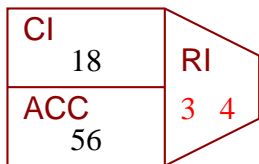


19	00010001
18	01100100
17	10001111

Le fonctionnement d'un processeur primitif

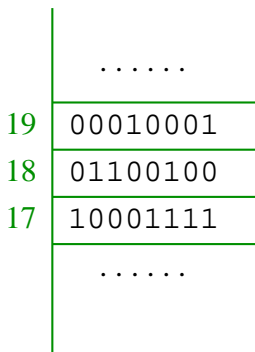
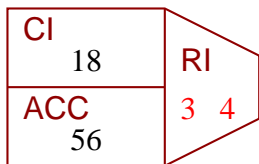


Le fonctionnement d'un processeur primitif



19	00010001
18	01100100
17	10001111

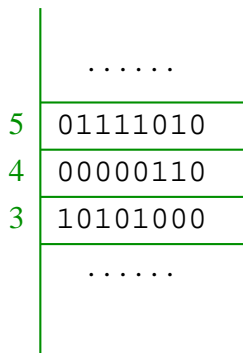
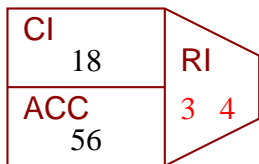
Le fonctionnement d'un processeur primitif



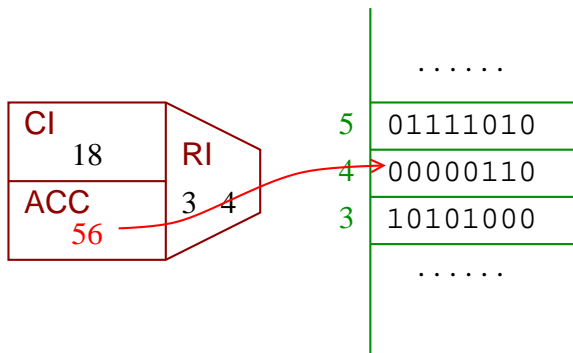
Opération 3:

écrire le contenu de l'accumulateur dans l'adresse indiquée

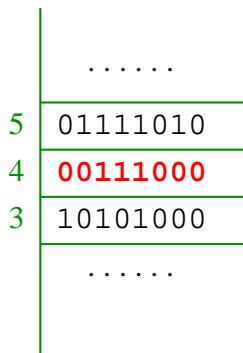
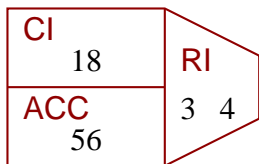
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

Les instructions élémentaires:

2	010	LOAD	copier la valeur de la mémoire dans l'accumulateur
3	011	STORE	copier la valeur de l'accumulateur dans la mémoire
4	100	ADD	ajouter la valeur de la mémoire à l'accumulateur
5	101	MPY	multiplier la valeur dans l'accumulateur avec la valeur de la mémoire

Le fonctionnement d'un processeur primitif

Les instructions élémentaires:

2	010	LOAD	copier la valeur de la mémoire dans l'accumulateur
3	011	STORE	copier la valeur de l'accumulateur dans la mémoire
4	100	ADD	ajouter la valeur de la mémoire à l'accumulateur
5	101	MPY	multiplier la valeur dans l'accumulateur avec la valeur de la mémoire
1	001	BR	copier l'adresse indiquée dans le compteur d'instructions

→ interruption de la séquence,
continuation avec une autre partie du programme

Le fonctionnement d'un processeur primitif

Les instructions élémentaires:

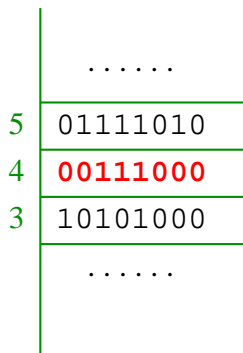
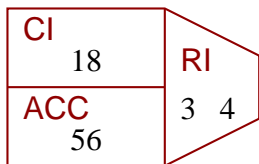
2	010	LOAD	copier la valeur de la mémoire dans l'accumulateur
3	011	STORE	copier la valeur de l'accumulateur dans la mémoire
4	100	ADD	ajouter la valeur de la mémoire à l'accumulateur
5	101	MPY	multiplier la valeur dans l'accumulateur avec la valeur de la mémoire
1	001	BR	copier l'adresse indiquée dans le compteur d'instructions
0	000	BRG	copier l'adresse indiquée dans le compteur d'instructions si l'accumulateur ne vaut pas zéro

Le fonctionnement d'un processeur primitif

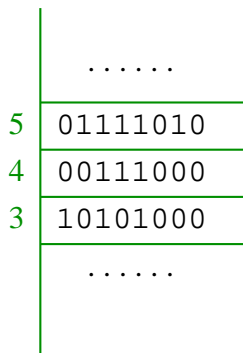
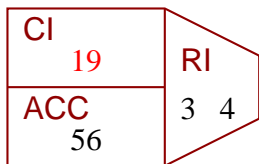
Les instructions élémentaires:

2	010	LOAD	copier la valeur de la mémoire dans l'accumulateur
3	011	STORE	copier la valeur de l'accumulateur dans la mémoire
4	100	ADD	ajouter la valeur de la mémoire à l'accumulateur
5	101	MPY	multiplier la valeur dans l'accumulateur avec la valeur de la mémoire
1	001	BR branchement	copier l'adresse indiquée dans le compteur d'instructions
0	000	BRG branchement conditionnel	copier l'adresse indiquée dans le compteur d'instructions si l'accumulateur ne vaut pas zéro

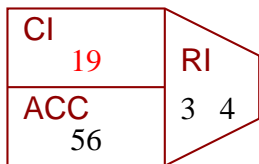
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

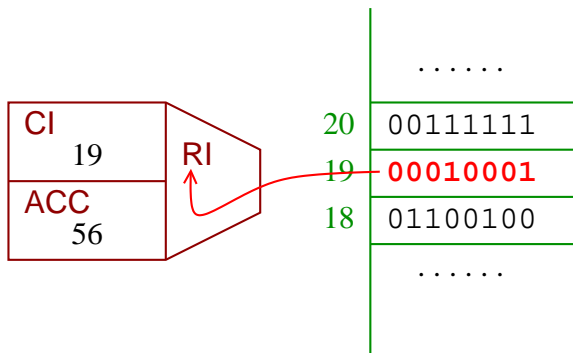


Le fonctionnement d'un processeur primitif

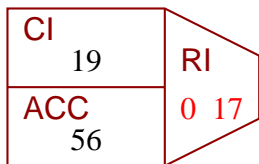


20	00111111
19	00010001
18	01100100

Le fonctionnement d'un processeur primitif

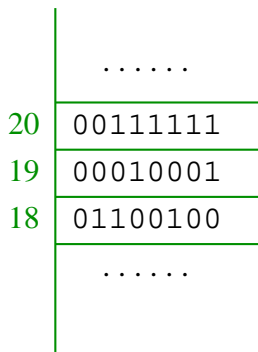
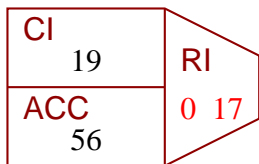


Le fonctionnement d'un processeur primitif



20	00111111
19	00010001
18	01100100

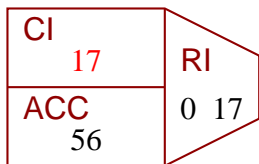
Le fonctionnement d'un processeur primitif



Opération 0:

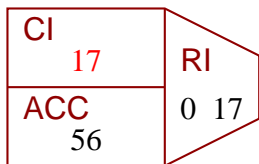
continuer avec l'instruction dans l'adresse indiquée
si le contenu de l'accumulateur ne vaut pas zéro

Le fonctionnement d'un processeur primitif



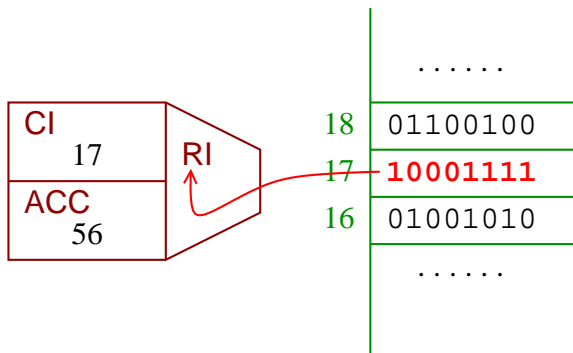
20	00111111
19	00010001
18	01100100

Le fonctionnement d'un processeur primitif

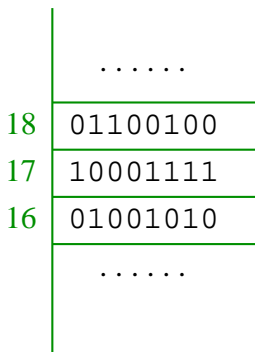
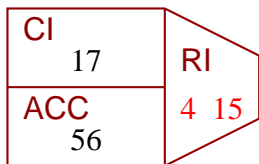


18	01100100
17	10001111
16	01001010

Le fonctionnement d'un processeur primitif



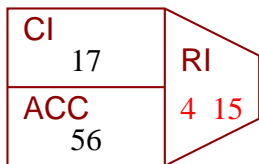
Le fonctionnement d'un processeur primitif



Opération 4:

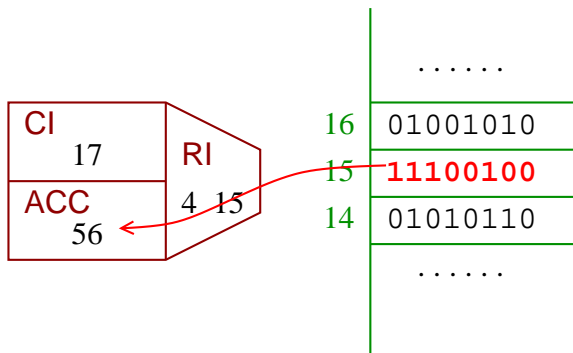
ajouter à l'accumulateur le contenu dans l'adresse indiquée

Le fonctionnement d'un processeur primitif

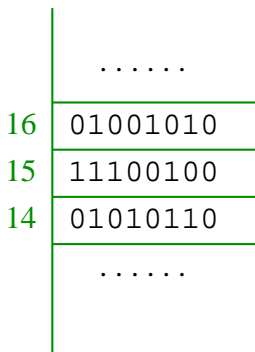
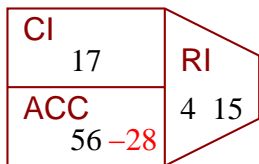


16	01001010
15	11100100
14	01010110

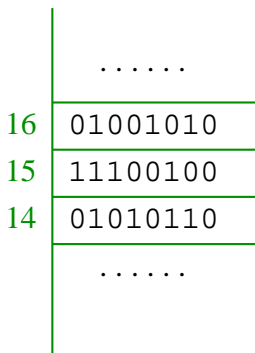
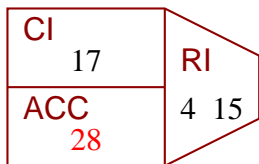
Le fonctionnement d'un processeur primitif



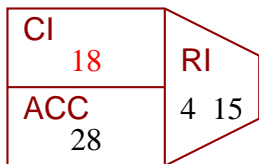
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

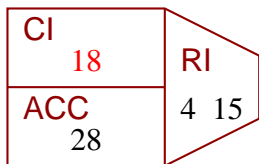


Le fonctionnement d'un processeur primitif



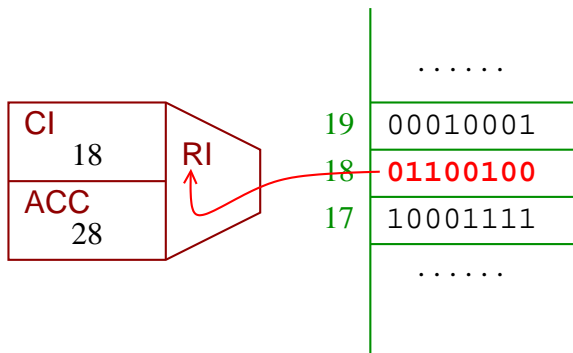
16	01001010
15	11100100
14	01010110

Le fonctionnement d'un processeur primitif

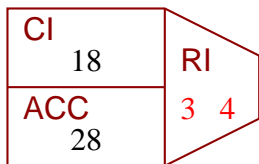


19	00010001
18	01100100
17	10001111

Le fonctionnement d'un processeur primitif

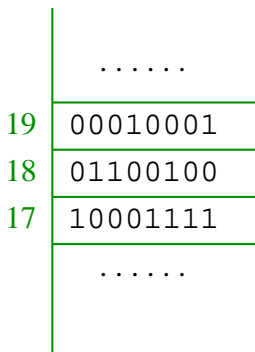
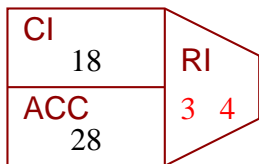


Le fonctionnement d'un processeur primitif



19	00010001
18	01100100
17	10001111

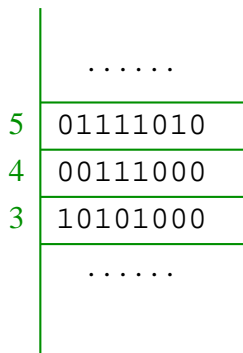
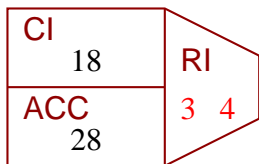
Le fonctionnement d'un processeur primitif



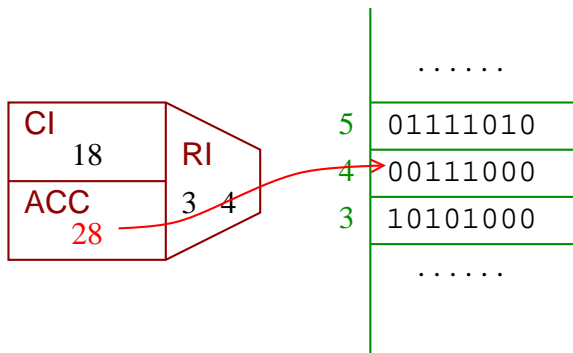
Opération 3:

écrire le contenu de l'accumulateur dans l'adresse indiquée

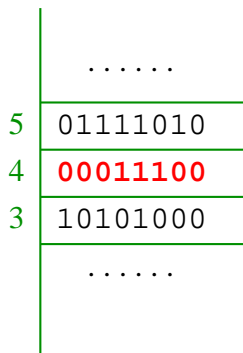
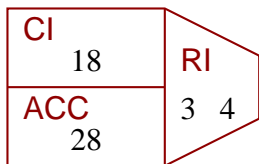
Le fonctionnement d'un processeur primitif



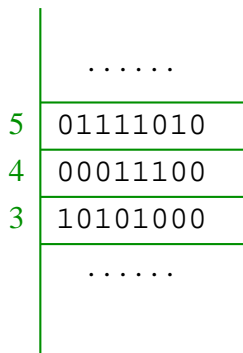
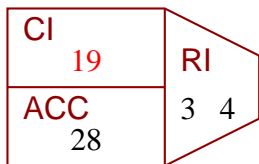
Le fonctionnement d'un processeur primitif



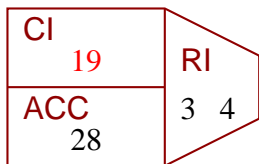
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

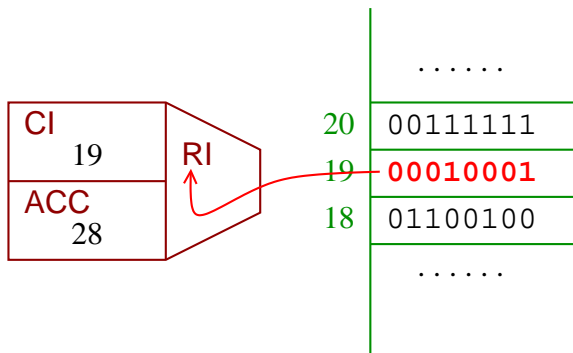


Le fonctionnement d'un processeur primitif

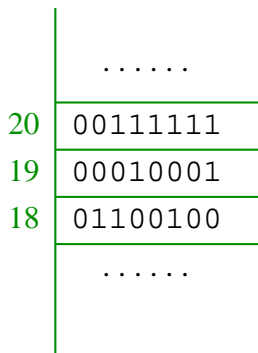
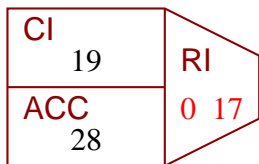


20	00111111
19	00010001
18	01100100

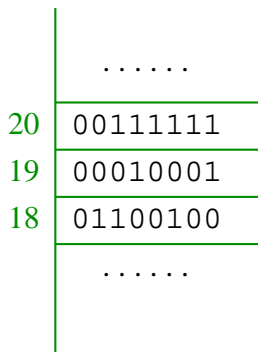
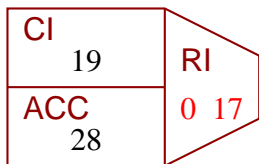
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif



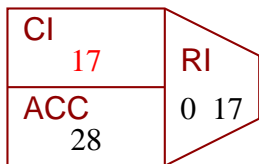
Le fonctionnement d'un processeur primitif



Opération 0:

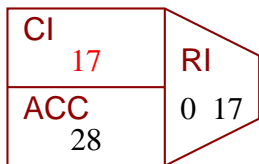
continuer avec l'instruction dans l'adresse indiquée
si le contenu de l'accumulateur ne vaut pas zéro

Le fonctionnement d'un processeur primitif



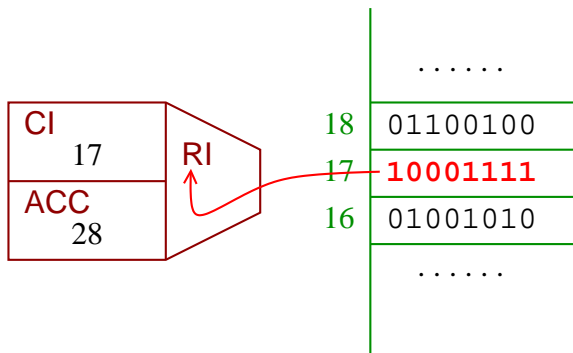
20	00111111
19	00010001
18	01100100

Le fonctionnement d'un processeur primitif

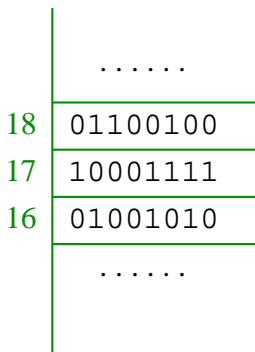
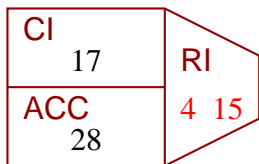


18	01100100
17	10001111
16	01001010

Le fonctionnement d'un processeur primitif



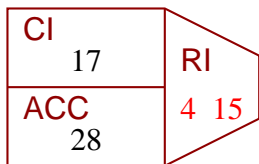
Le fonctionnement d'un processeur primitif



Opération 4:

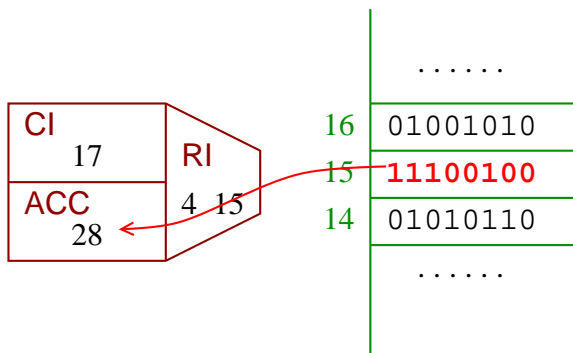
ajouter à l'accumulateur le contenu dans l'adresse indiquée

Le fonctionnement d'un processeur primitif

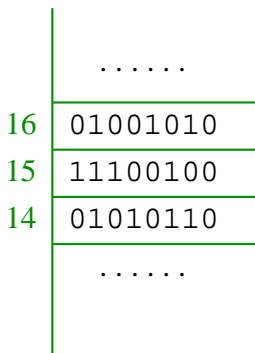
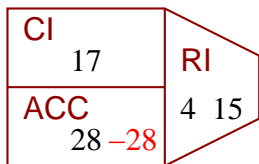


16	01001010
15	11100100
14	01010110

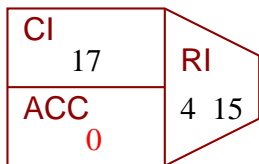
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

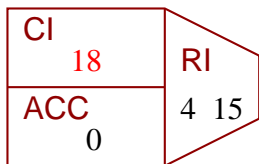


Le fonctionnement d'un processeur primitif



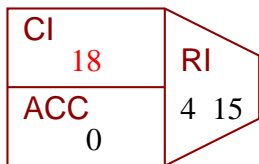
16	01001010
15	11100100
14	01010110

Le fonctionnement d'un processeur primitif



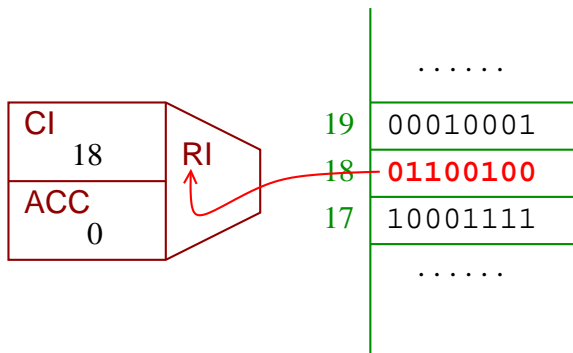
16	01001010
15	11100100
14	01010110

Le fonctionnement d'un processeur primitif

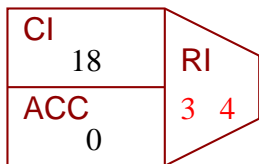


19	00010001
18	01100100
17	10001111

Le fonctionnement d'un processeur primitif

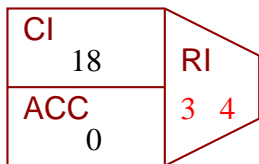


Le fonctionnement d'un processeur primitif



19	00010001
18	01100100
17	10001111

Le fonctionnement d'un processeur primitif

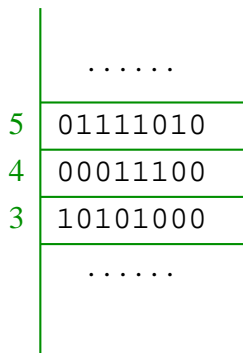
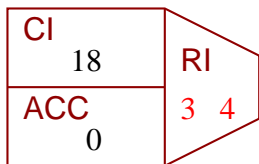


19	00010001
18	01100100
17	10001111

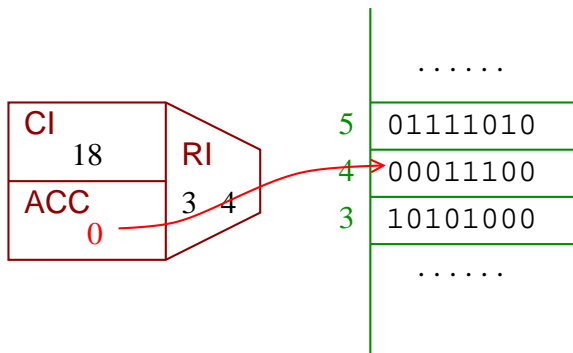
Opération 3:

écrire le contenu de l'accumulateur dans l'adresse indiquée

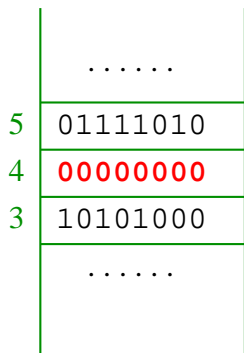
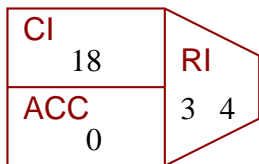
Le fonctionnement d'un processeur primitif



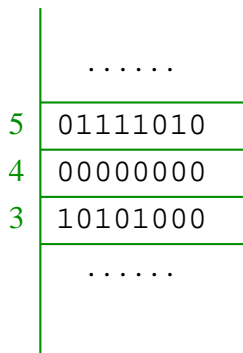
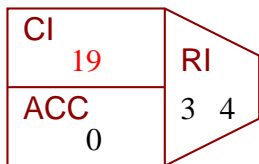
Le fonctionnement d'un processeur primitif



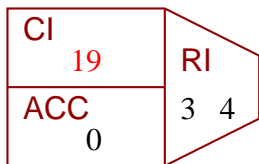
Le fonctionnement d'un processeur primitif



Le fonctionnement d'un processeur primitif

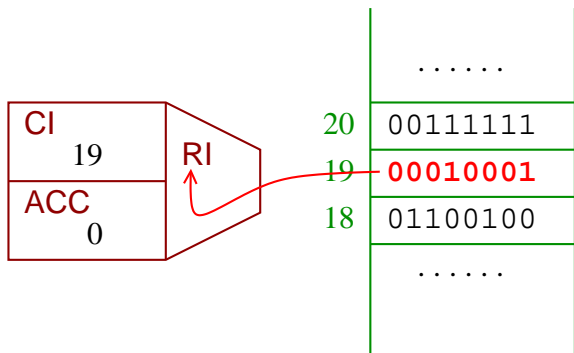


Le fonctionnement d'un processeur primitif

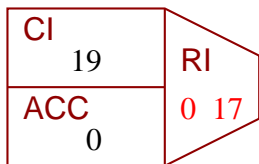


20	00111111
19	00010001
18	01100100

Le fonctionnement d'un processeur primitif

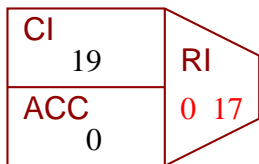


Le fonctionnement d'un processeur primitif



20	00111111
19	00010001
18	01100100

Le fonctionnement d'un processeur primitif

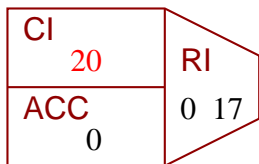


20	00111111
19	00010001
18	01100100

Opération 0:

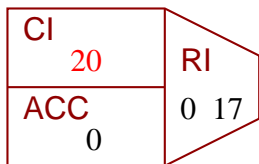
continuer avec l'instruction dans l'adresse indiquée
si le contenu de l'accumulateur ne vaut pas zéro

Le fonctionnement d'un processeur primitif



20	00111111
19	00010001
18	01100100

Le fonctionnement d'un processeur primitif



20	00111111
19	00010001
18	01100100

→ réalisation d'une **boucle**

Le fonctionnement d'un processeur primitif

Le programme exprimé dans un langage symbolique (assembleur):

16: LOAD *10

17: ADD *15

18: STORE *4

19: IF (ACC NOT 0) GOTO 17

20: ...

*10 = la valeur encodée à l'adresse 10

Le fonctionnement d'un processeur primitif

Le programme exprimé dans un langage symbolique (assembleur):

16: LOAD *10

17: ADD *15

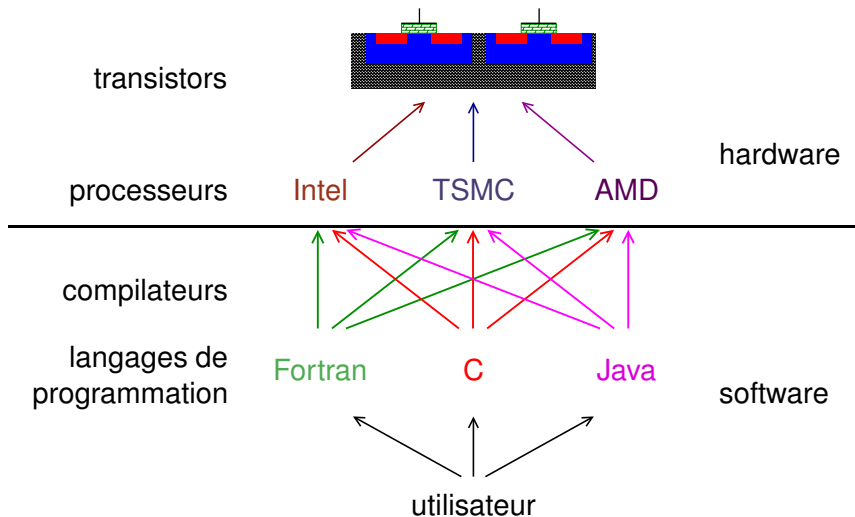
18: STORE *4

19: IF (ACC NOT 0) GOTO 17

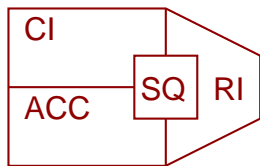
20: ...

→ toute complexité de la fonctionnalité des ordinateurs peut être introduite par la **programmation** (software)

“hardware” et “software”



3.3 Réalisation en pratique



mémoire
.....
10001111
01001010
11100100
01010110
.....

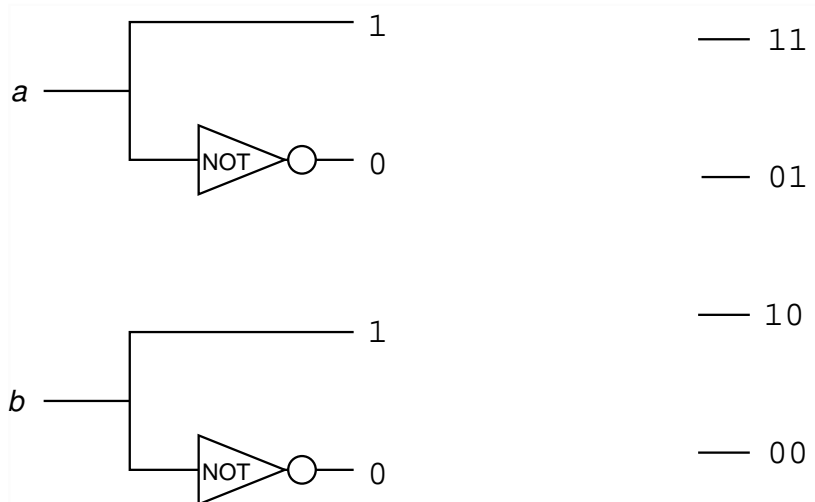
Installations supplémentaires:

→ séquenceur (cablé ou microprogrammé)

→ contrôle le déroulement de toutes les actions

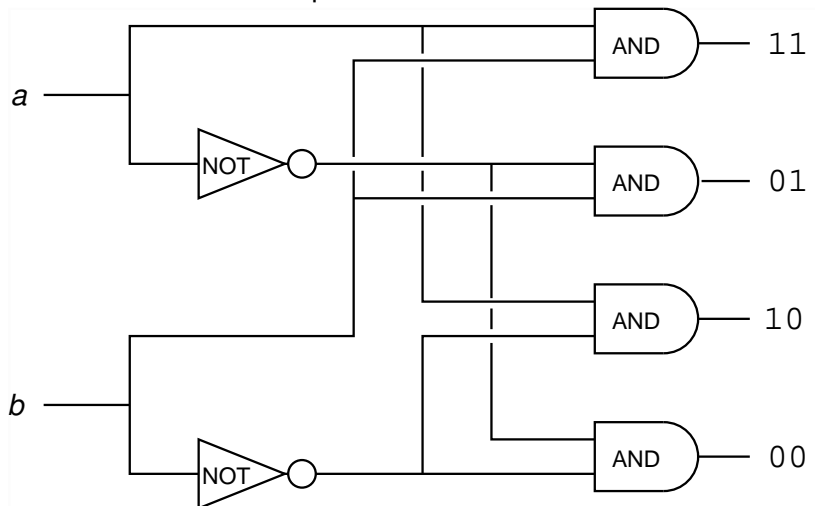
Comment décoder des codes d'instructions ?

... pour deux bits: a b

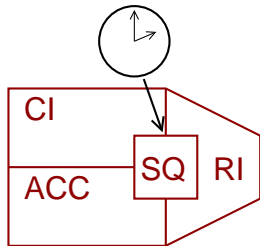


Comment décoder des codes d'instructions ?

... pour deux bits: a b



3.3 Réalisation en pratique

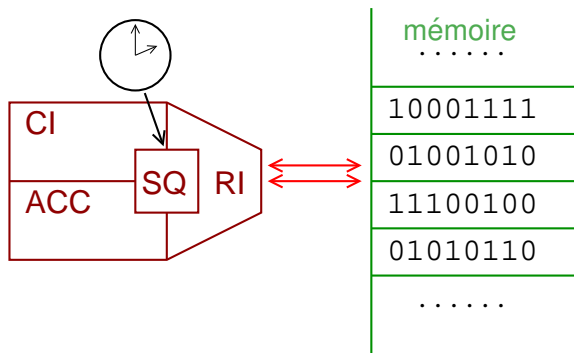


mémoire
.....
10001111
01001010
11100100
01010110
.....

Installations supplémentaires:

- séquenceur
- horloge (oscillateur de quartz)
 - donne des signaux de cadence réguliers (GHz) pour lancer des actions

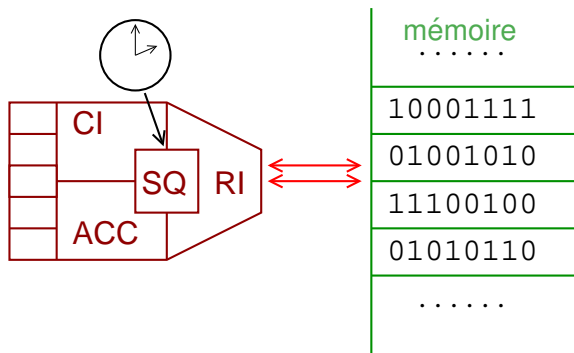
3.3 Réalisation en pratique



Installations supplémentaires:

- séquenceur
- horloge
- “bus” de transfert entre mémoire et processeur
... pour des adresses et des données

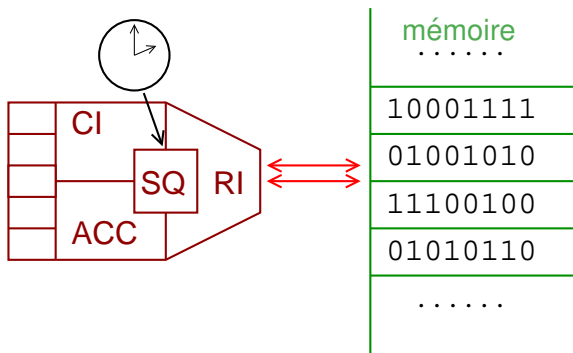
3.3 Réalisation en pratique



Installations supplémentaires:

- séquenceur
- horloge
- “bus” de transfert
- registres supplémentaires (SRAM)

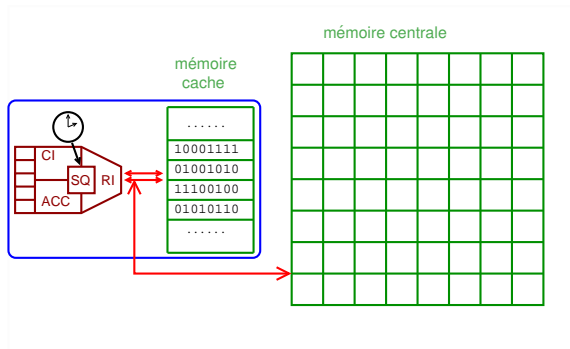
3.3 Réalisation en pratique



Optimisations:

→ execution parallèle des opérations indépendantes
(p.ex. copie des données et incrément du CI)

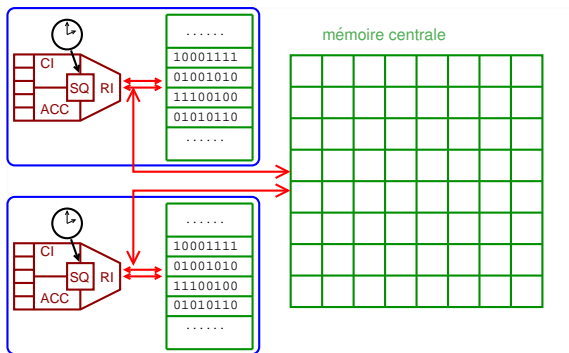
3.3 Réalisation en pratique



Optimisations:

- execution parallèle des opérations indépendantes
- mémoire "cache" et mémoire centrale
(réalisation avec SRAM et DRAM)

3.3 Réalisation en pratique



Optimisations:

- execution parallèle des opérations indépendantes
- mémoire “cache” et mémoire centrale
- opération parallèle de plusieurs processeurs
- ⇒ architecture “massivement parallèle” (p.ex. 256 processeurs)