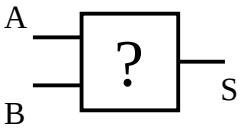
Comparateur de différence :

Différence de deux valeurs booléennes A et B (Sortie à 1 si différence)

Etape 1 : Table de vérités

	A	B	S
Ligne 1	0	0	0
Ligne 2	0	1	1
Ligne 3	1	0	1
Ligne 4	1	1	0



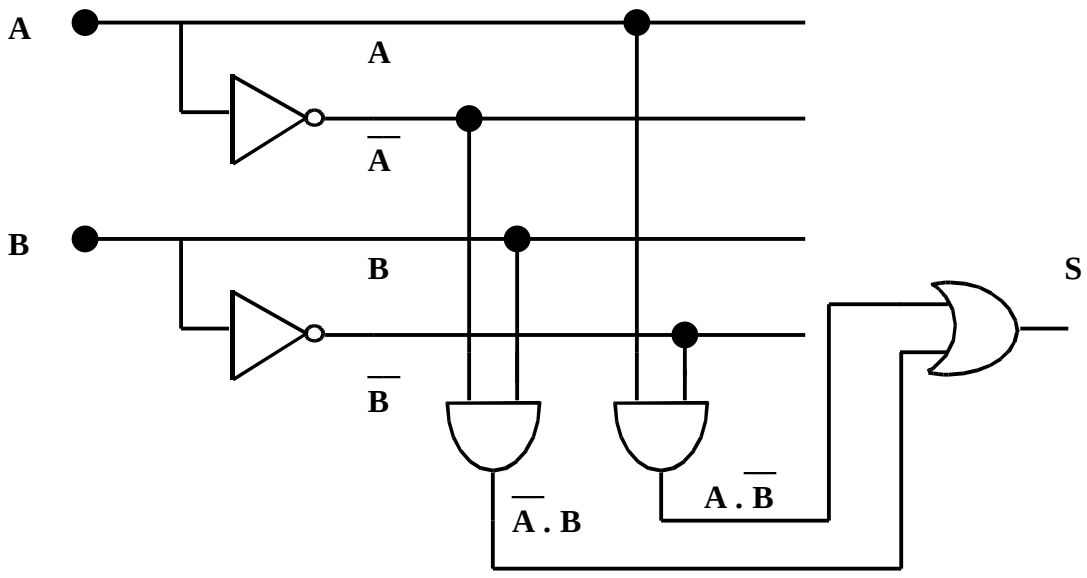
Etape 2 : Ecriture Algébrique (S = f(a , b)

S = Ligne 2 + Ligne 3

S = [(A = 0) ET (B = 1)] OU [(A = 1) ET (B = 0)]

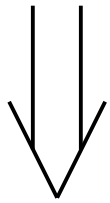
$$S = \overline{A} \cdot B + A \cdot \overline{B}$$

Etape 3 : Réalisation du circuit



Simplification des fonctions binaires :

	A	B	S
/ A	0	0	
	0	1	
A	1	0	
	1	1	



S		A	
		0	1
B	0		
	1		

L'agencement dans les tableaux doit se faire selon un ordre (Voir Page 13) pour que les termes soient logiquement adjacents (00, 01, 11,10)

Et les colonnes des extrémités étant logiquement adjacentes, elles doivent être vues comme étan géométriquement adjacentes.

00 01 11 10

Exemple : (Page 14)

F(a,b,c) est vrai si A.B.C = 1 OU A./B.C = 1

Si dans le tableau on obtient deux UN l'un à côté de l'autre, on peut simplifier $\Rightarrow$ On supprime la variable qui change.

Page 15

F(a, b, c) = A + /A.C
F(a, b, c) = C + A./C
F(a, b, c) = A + C

Codage / Décodage / Transcodage :

Bits : Position binaire
Transcodage : Passage d'un code à un autre (Exemple : UC → Ecran)

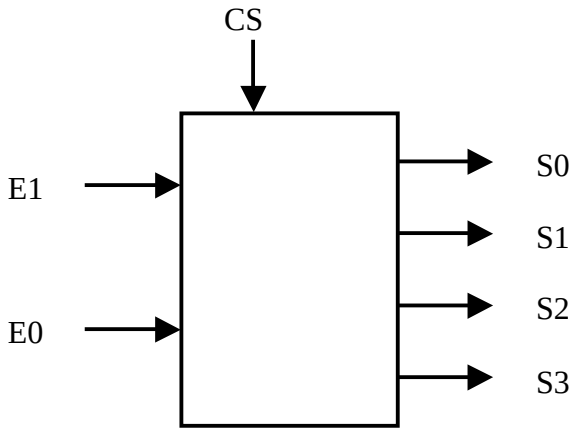
Les différents codes :

- BCD
- AIKEN (Symétrie)
- Code Excédent (Auto-complémentaire)

Décodeurs :

C'est un circuit qui actionne seulement une sortie

Décodeur 2.4

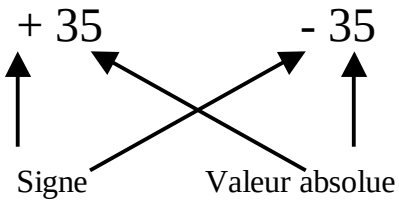


L'Unité Arithmétique et Logique est inc use dans l'Unité Centrale

Addition binaire : 1 + 1 = 0 et retenu de 1

1 + 1 = Somme 0 et Retenu 1

La représentation des nombres négatifs



La représentation en complément à 2 :

N = 10
E = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

CV(X) = -X

CV(X) soit appartenir à E

- X = (10 – X) modulo 10

Complément restreint

N = 2
E = { 0, 1 }

Complément restreint d'un nombre binaire :

```
      1 0 0 0 1 1     + 35
      0 1 1 1 0 0
+             1
-------------------------
      1 1 1 0 1       - 35


      1 0 0 0 1 1
+     0 1 1 1 0 0
-------------------------
    1 0 0 0 0 0 0
    1 0 0 0 0 0 0 modulo  =   0
```

Additionneur

	Ri	Ri-1		
	An-1 …	Ai	A1	A0
+	Bn-1 …	Bi	B1	B0
		Si		

Le transfert de l'information :

Réseau permettant de faire communiquer l'ensemble des modules entre eux.

- Transfert (Gestion des transfert
- Qu'est ce qu'une information ?

Notion d'aiguillage
- Convergent
- Divergent

Considérons une porte ET

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

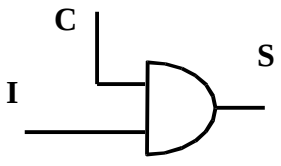
« Multiplicateur binaire »

C	I	S
0	0	0
0	1	0
1	0	0
1	1	1

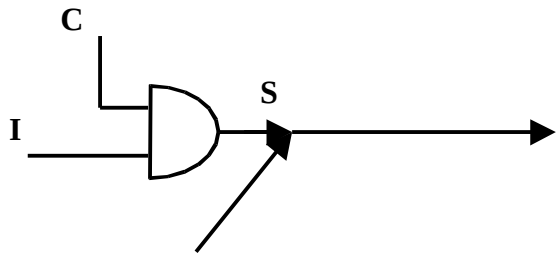
Lorsque C = 0 quelque soit I on a S = 0

C	I	S
0	-	0
1	-	I

C	S
0	0
1	I



Aiguillage convergent par l'utilisation d'un ET
Si C = 1, l'information peut passer
Si C = 0 pas de passage



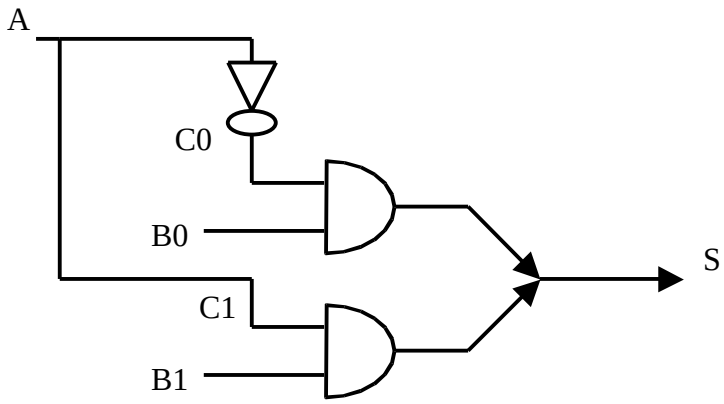
Multiplexeurs (P. 48)

N voies en entrée
1 voie en sortie

Le canal A désigne le bus qui doit avoir l'accès.
On y délivre une adresse binaire.

Le circuit doit activer les commandes à 1 du bus désigné par A et doit mettre 0 pour les commandes des autres bus d'entrées.

Multiplexeurs à deux voies



A est constitué d'un seul fil.

Deux bus d'entrées
Si A contient 0 alors C0 = 1 et C1 = 0
Si A contient 1 alors C0 = 0 et C1 = 1

A	C0	C1
0	1	0
1	0	1

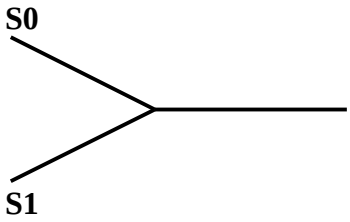
INFORMATIONS

« Troisième éta »

{ 0 , 1 Rien »}

Etats d'information

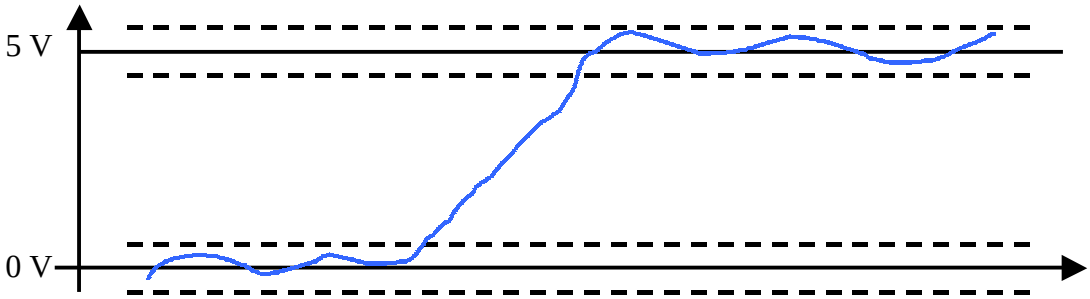
Source de courant



Il faut intercaler un dispositif permettant de « couper » le courant sur le fil

C	S
0	HI
1	I

Il faut qu'il n'y est plus de courant (Haute Impédance)



Loi d'ohm V = RI
Pour qu'il n'y est plus de courant il faut amener un courant qui est une intensité presque nulle.

I = V / R , On observe que pour diminuer la val eur de I on doit prendre un R >>1

Cet outil est appelé le « Buffer trois états » (p. 53) dans lequel lorsque C = 0 il n'y a pas de courant.

Les Buffers trois états et 3 états inverses ne peuvent pas se construire à partir de porte NAND

Théorème de l'information / Théorème de l'automate

{ 0 , 1 } / Rien

Information Etat technologique

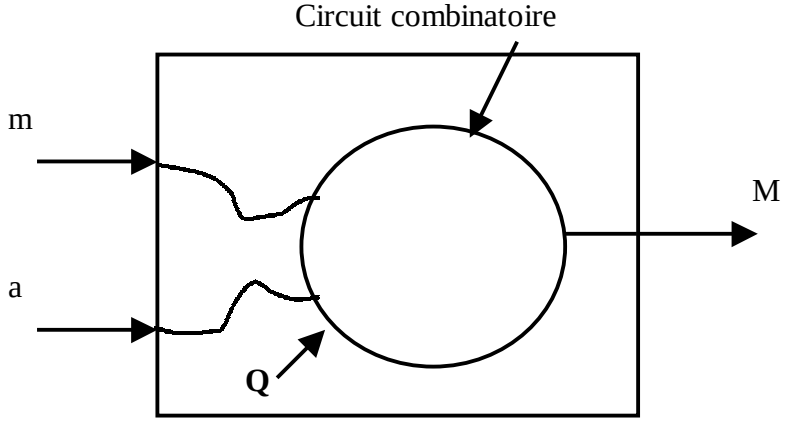
Caractérisée par un couran

LOGIQUE SEQUENTIELLE :

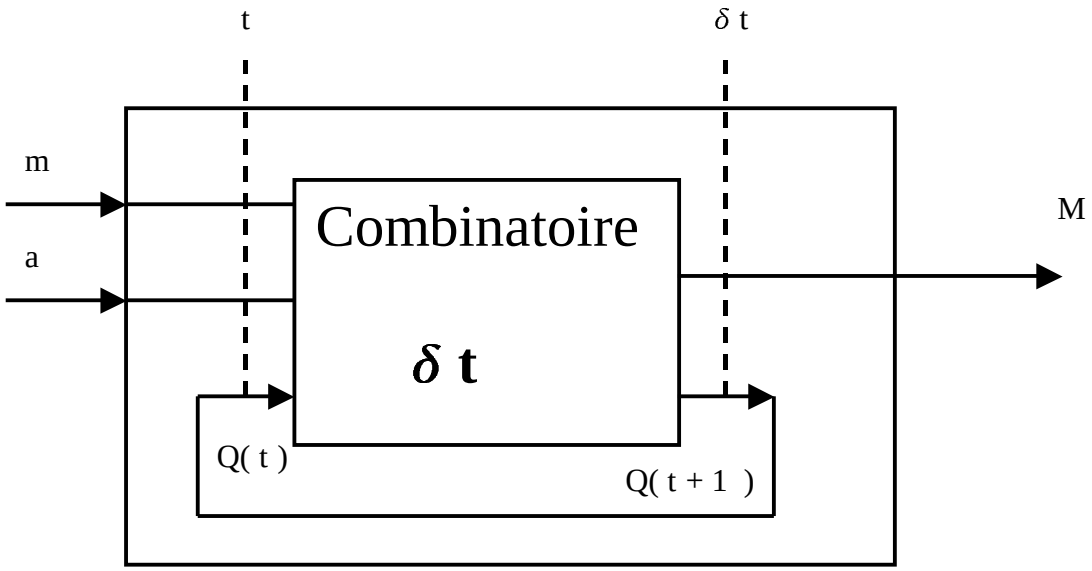
Rappel :

Combinatoire : A **une** fonction d'entrée on associ **une** valeur de sortie

Variable primaire
Variable secondaire



C'est le circuit lu -même qui gère la variable Q. Concept de boucle Autogestion



Un circuit séquentiel est un circuit combinatoire avec une ou plusieurs sorties qui viennent sur les entrées. Autogestion des états internes.

M

Q(t) \ m . A	0	1	11	10
0	0	0	-	1
1	1	0	-	1

M = Q (T + 1) = m + /a.. Q(t)

Circuit séquentiel :
circuit combinatoire

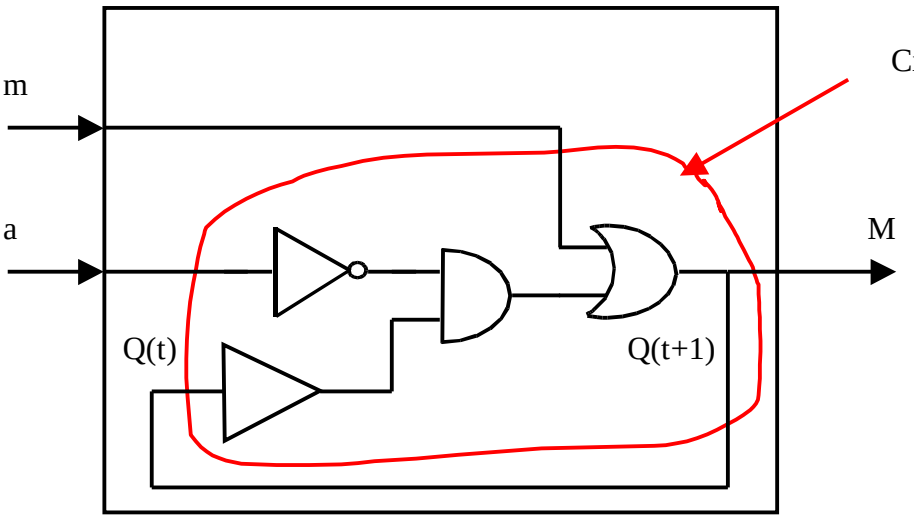


Tableau de vérité périodique : Temps de traver ée du circuit

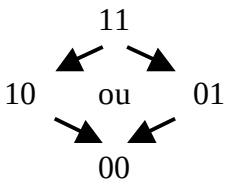
Les bascules

Bascul e R / S

On bascule d'un état à un autre (Bistable : on peut mettre deux états stables)

Que se passe t'il si je met R = S = 1
Q (T +1) = 0

Le passage de 11 à 00 n'est pas synchronisé (La notion de simultanéit n'existe pas)



On regarde les états de façon continue

Concept de synchronisation

Deux types de circuit :

- Asynchrone
- Synchrone (On ajoute un autre signal qui donne les tops (moment durant lesquels on observe l'état des entrées)

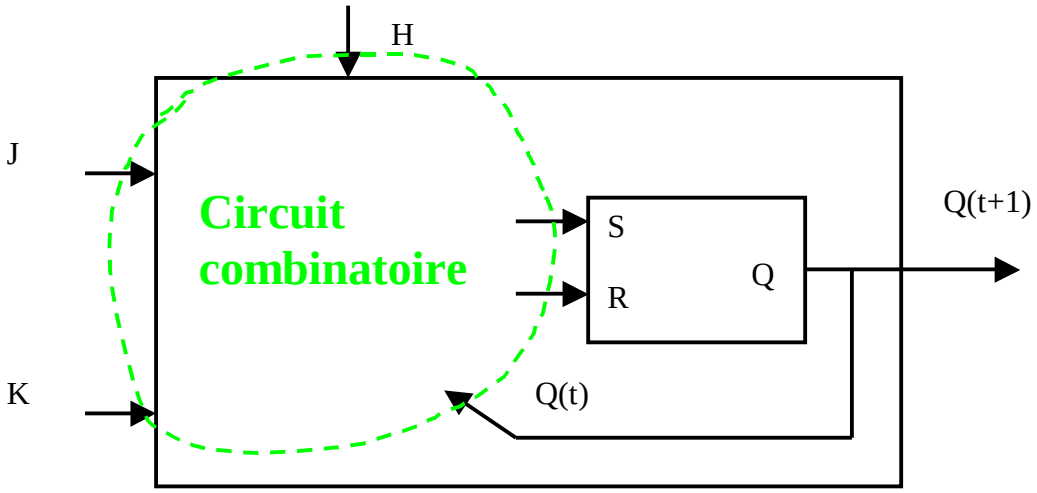
Le laps de temps durant lequel on observe les états doit être très cour

Front montant / Front descendant : Espace de temps le plus petit possible

Circuit actif sur front montant
Circuit actif sur front descendan

Dans les deux cas c -dessus, on regarde l'état des entrées durant la front.

Bascule JK à partir d'un RS



Q(t)	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	0	-	0
1	-	0	1

K Q(t)

HJ \ S	00	01	11	10
00				
01				
11			1	
10	-		1	-

S = H . J ./ Q(t)

R = H . K . Q(t)

Circuit asynchrone:

On regarde en continu le signal

Circuit synchrone :

On regarde à intervalle régulier (discret)
On ajoute un signal appelé horloge

Un front est le plus petit espace de temps où il se passe « quelque chose »
⇒ Signal électrique sous états : Haut, Bas, Front Montant, Front Descendant.

Une horloge est à UN sur un front.

S

HJ \ K Q(t)

S	00	01	11	10
00		-	-	
01			-	
11	1	-	0	1
10	0	-	0	

S = H . J . / Q(t)

S

HJ \ K Q(t)

S	00	01	11	10
00	-			-
01	-			-
11			1	
10	-		1	-

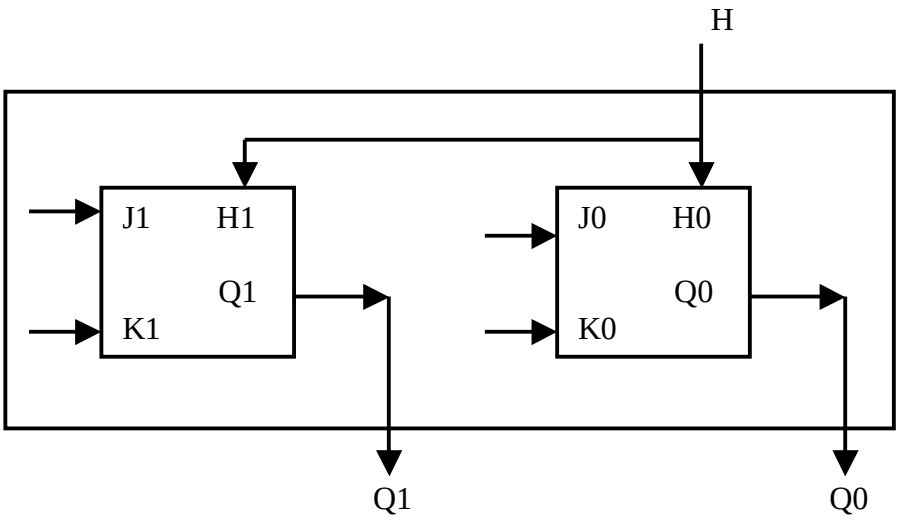
R = H . K . Q(t)

Le comptage :

On utilise des bascules T (Modulo T)

Compteur synchrone :

Avec deux bascules on compte de 0 à 3.



Q0(t)	J0	K0	Q0(t + 1)
0	1	0	1
	1	1	
1	0	1	0
	1	1	

Q1(t)	J1	K1	Q1(t + 1)
0	0	0	0
	0	1	
0	1	0	1
	1	1	
1	0	0	1
	1	0	
1	0	1	0
	1	1	

Mémoire Ensemble de récipients tous identiques numérotés de 0 à N-1

Octet : 8 bits

- La taille d'un mot : p bits
- Capacité de la mémoire : N * p
  avec $N = 2^n$ où n est le nombre de fil du bus d'adresse

  128 M : $2^7 . 2^{20} = 2^{27}$ 27 fils pour adresser 128 M

p : taille du bus de données

Pour une mémoire de 64 k + 1
Bus de données = 1 fil
$2^6 . 2^{10} = 2^{16}$ 16 fils d'adresse

32 M * 4
Bus de données de 4 fils d0 → d3

32 M = $2^5 . 2^{20} = 2^{25}$ 25 fils d'adresse

1 G * 8
8 fils sur le bus de données
30 fils pour le bus d'adresse

Deux types de mémoires :

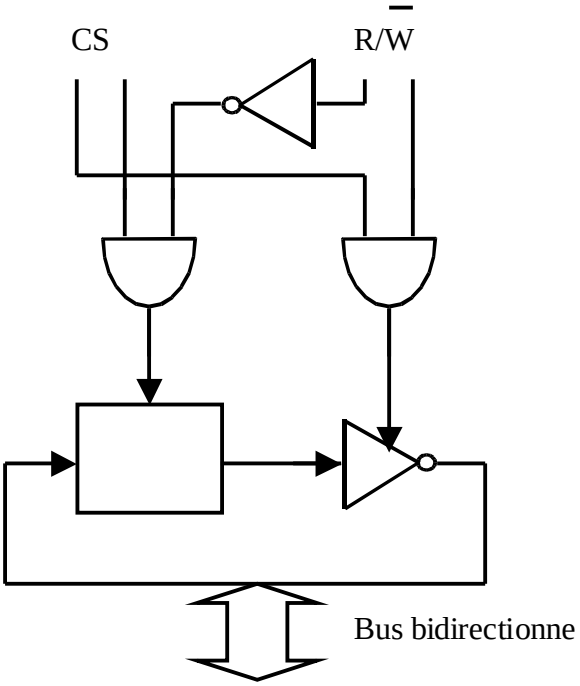
RAM
- SRAM : Static (Pas de rafraichissement)
- DRAM : Dynamic (Rafraichissement de la mé oire)
- SDRAM : Synchronous (Synchronisation des bus entre le processeur et l mémoire)

ROM (Voir page 80)

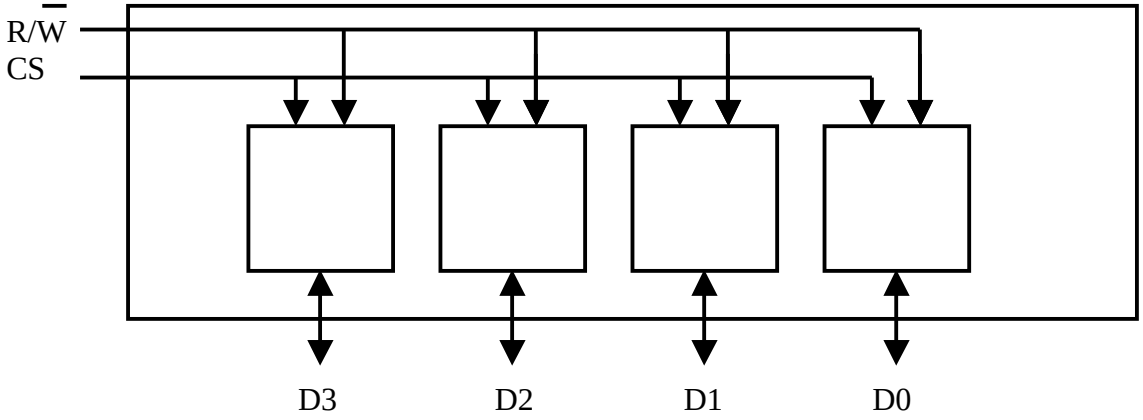
Exemple : **RAM 64 K * 8**

Type Capacité taille du mot

Schéma d'une mémoire :

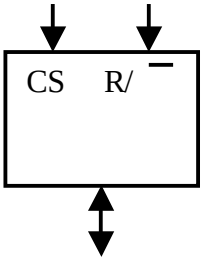


Construction d'un mot de 4 bits



Résumé mémoire :

1.Cellule élémentaire de mémoire :



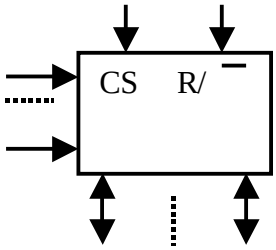
CS	R / $\overline{W}$	
1	1	Lecture
1	0	Ecriture
0	-	mémorisation

Buffer trois Etats : Permet de limiter les circuits logiques

2. Mot mémoire : Rassemble des cellules élémentaires commandées en parallèle.

3. Boîtier mémoire : Regroupement d'un grand nombre de mot mémoire

Adresse



INTEGRATION

Bus de données

4. IMPRESSION / ASSEMBLAGE

$A = qN + r$

r : Adresse dans le boîtier
q : Numéro du boîtier

Langage symbolique / Langage machine :

Il faut adapter le Langage de Haut niveau au langage machine.

Instruction machine

A = B + C (Une opération + ; des données : A, B, C)
Source Destination
B + C → A (+ ; B ,C ,A)

Notions de statut :

- Variables
- Constantes

Pour distinguer A = B + C de A = B + Constante, on utilise le statut

A = B + C est codé

1	+	B	C	A

A = B + 13 est codé

1	+	B	13	A

Instruction machine :

Statut	Opération	@Ah	@Ab	Constante

Un octet

L'adresse est portée par deux octets @Ah et @Ab car on a une mémoire de 64 ko. Les constantes sur un octet.

Exemple :

2A	0B	12	13	A1	F3

0B00 0B05

Décryptage :

2 : Statut (X = Y opérateur Constante)
A : Code de multiplication (X = Y * Constante)
0B12 : Adresse de Y (Valeur à l'adresse 0B12 : $1E = 30)
13 : Constante (19)
A1F3 : Adresse de X

30 * 19 → A l'adresse $A1F3

Notion d'étiquette :

(Vers ET) ⇒ Rupture de Séquence

Associations :

- Les objets ont des emplacements en mémoire
- Toute instruction virtuelle a une correspondance dans la mémoire
- La Source est située avant la des ination dans les instructions machines (Car avant de manipuler les données il faut connaître l'opération que l'on doit réaliser.

Programme :

10 05 05 05 20 1A 2B AF 0C 42 1A 2B AF 0C 33 05 05 01 05 05 67 05 05 00 00 09 00 1C

@ Val1 : $ 05 05

Val1 ← 05

@Val2 : $ 1A 2B

@Val3 : $ AF 0C

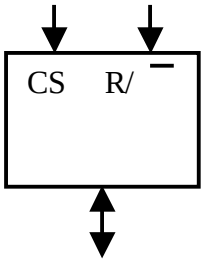
B → C

B + C → C (@ $00 09)

A – 1 → A

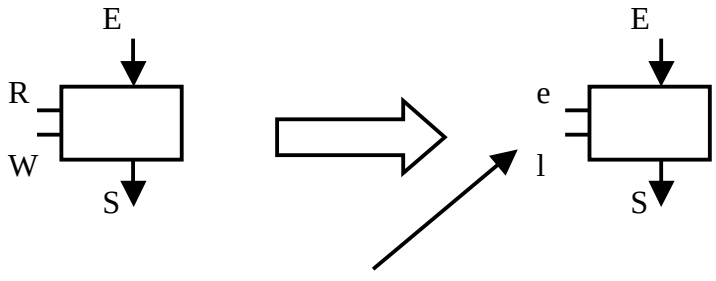
SI A ≠ 0 VERS @ $00 09

Cellule élémentaire de mémoire :

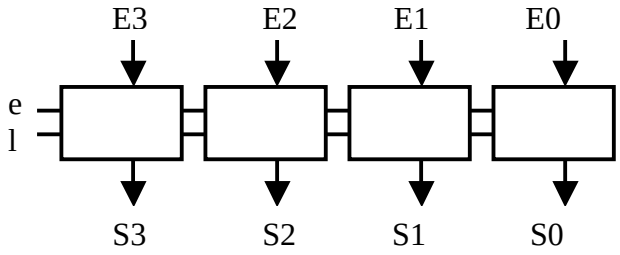


Registres : (Il faut avoir une entrée et une sortie décalée pour la mémoire)

Modèle utilisé



e : Ecriture et l : Lecture



Création d'un compteur permettant d'atteindre tout l'espace mémoire
(⇒ 16 fils en sortie car 64 ko de mémoire)

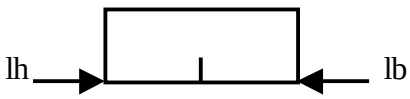
INIT est utilisé pour mettre les valeurs qui sont sur E15,…,E0 dans le compteur

Registres accumulateurs (8 bits)
Bus de données de 8 bits
Registre d'états (Retenue, NULL, >, < , = ,…)
Si opération logique alors le résultat est dans RE
Si opération arithmétique résultat est dans Accumulateur

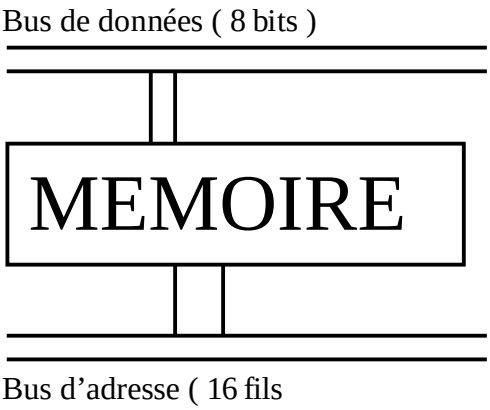
<u>Registre instruction :</u>

Il faut stocker le statut et l'opération dans un registre car ces deux valeurs commencent chaque programme.

On les stocke dans le registre instruction qui doit contenir le statut et l'opérati



Comment est lu un programme ?



Un programme est une suite d'instructions dans la mémoire. Pour lire un programme il faut avoi l'adresse de début de programme et ensuite il faut lire de case en case.

Donc le bus adresse doit avoir un compteur (Program counter) modulo 64 Ko pour se déplacer case par case dans la mémoire.

Dans le cas d'instruction « Vers », il faut mettre l'adresse de l'étiquette dans le compteur (celle-ci est contenue sur deux octets)
Il y a donc changement de niveau : niveau des données → niveau des adresses.

Il faut stocker chaque « morceau » de l'adresse grâce à deux accumulateurs de 8 bits avec pour chacun une entrée et une sortie.

Il s'agit d'un pointeur d'adresse de programme ou d'un registre d'adresse de programme.

La sorties des accumulateurs sont synchronisées pour initialiser le compteur ordinal.

Exemple :

32 αBh αBb 13 αAh αAb

Il faut un lien entre les cases d'adresse et les cases contenant les valeurs

Il faut donc ajouter deux registres pour stocker l'adresse (Exemple : αBh , αBb) ils se situent entre le bus de données et le bus d'adresse) : Appelé RAD (Registre Adresse de données)

Langage de déroulement :

32	@Bh	@Bb	13	@Ah	@Ab

Lecture de PC (lpc)
L'adresse sort sur le bus d'adresse
CS mémoire
Ouverture de la mémoire en lecture
32 sur le bus de données
RI ouvert en écriture

} **l_{pc} Csmem R/W l_{RI}**

Inc PC ← Incrément du PC

l_{pc} Csmem R/W eRADP

(lecture PC Sélection mémoire en lecture écrit dans RAD la partie Haute)

IncPC

l_{pc} Csmem R/W eRADPb

…

Codage du Statut des Instructions Machines :

A : = Cte	1	10	Cte	@Ah	@Ab			
A := B	2	20	@Bh	@Bb	@Ah	@Ab		
A := B Opa Cte	3	3 -	@Bh	@Bb	Cte	@Ah	@Ab	
A := B Opa C	4	4 -	@Bh	@Bb	@Ch	@Cb	@Ah	@Ab
Vers ET	5	5 -	@Eh	@Eb				
Si A Opl Cte Vers ET	6	6 -	@Ah	@Ab	Cte	@Eh	@Eb	
Si A Opl B Vers ET	7	7 -	@Ah	@Ab	@Bh	@Bb	@Eh	@Eb

Opa : Opération arithmétique **Opl : Opération logique** **Cte : Constante**

Codage des Opérations Arithmétiques et Logiques :

+	-	*	/	=	!=	>	<	>=	<=
2	3	4	5	6	7	8	9	A	B