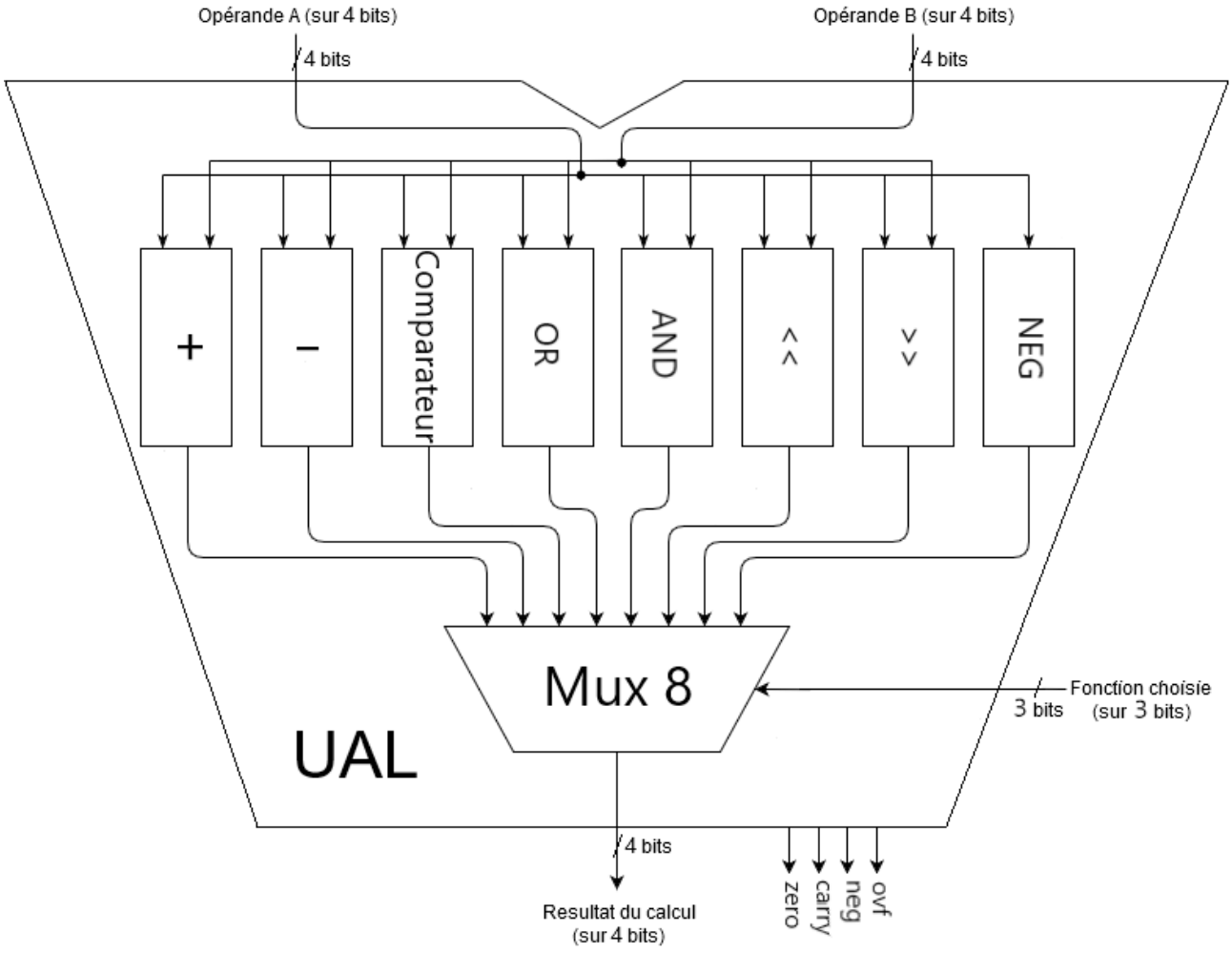
Université Sétif 1
Faculté des Sciences
Département d'Informatique

2-ième année(année2019/20)
Architecture des Ordinateurs
Enseignant Kara Abdelaziz

Solution Série 1
(Pré-requis et principaux composants d'un ordinateur)

Exercice 01 :

Schéma générique de l'UAL :

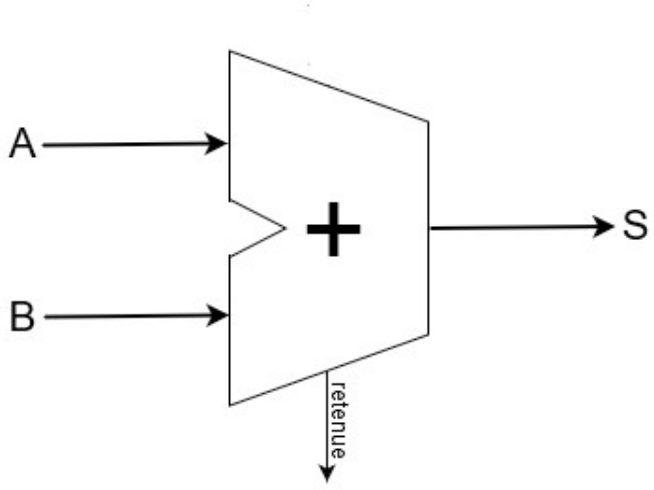


1)

Demi-Additionneur

Un demi-additionneur est un additionneur sur 1 bit qui renvoie en sortie le résultat sur 1 bit de la somme, en plus de la retenue de l'opération d'addition.

Étape 1 : Schéma global



Étape 2 : Table de Vérité

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

R : Retenue

Étape 3 : Fonctions Canoniques Disjonctives

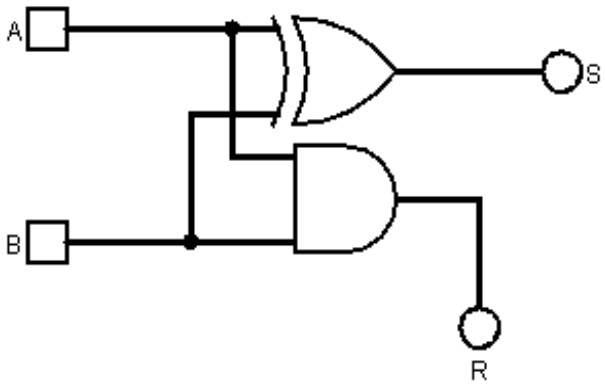
$S(A,B) = \overline{A}\cdot B + A\cdot\overline{B}$
$R_s(A,B) = A\cdot B$

Étape 4 : Minimisation Algébrique

$S(A,B) = \overline{A}\cdot B + A\cdot\overline{B}$
$S(A,B) = A \oplus B$

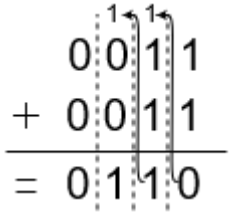
$R(A,B) = A\cdot B$

Étape 5 : Logigramme



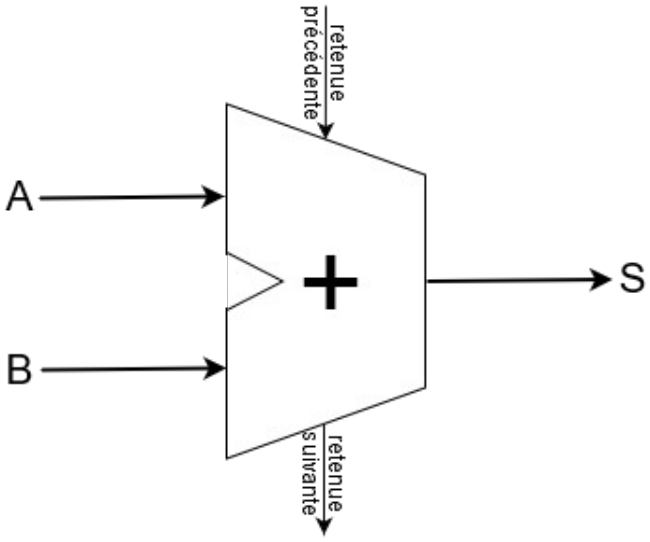
Soustracteur Complet :

La construction d'un additionneur 4 bits Complément-à-2/Valeur Absolue à partir de d'additionneur 1 bit**.**



Si on observe l'opération d'addition Complément-à-2/Valeur Absolue sur un seul bit, par exemple sur le 2-ème bit (2-ème colonne sur le schéma), on peut construire un additionneur sur 1 bit qui doit avoir 3 entrées qui sont le 2-ème chiffre dans A, le 2-ème chiffre dans B et la retenue du calcul du bit précédent (du 1-ier bit/1-iere colonne). et 2 sorties qui sont la Somme et la retenue pour le calcul du bit suivant (pour le 3-ème bit/3-ème colonne).

Étape 1 : Schéma global



Étape 2 : Table de Vérité

A	B	R_p	S	R_s
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

R_p : Retenue précédente
R_s : Retenue suivante

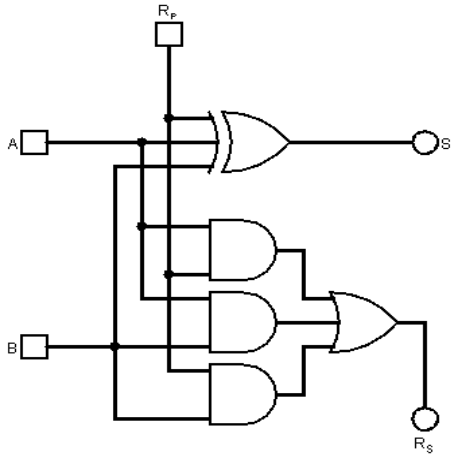
Étape 3 : Fonctions Canoniques Disjonctives

$S(A,B,R_p) = \overline{A}\cdot\overline{B}\cdot R_p + \overline{A}\cdot B\cdot\overline{R}_p + A\cdot\overline{B}\cdot\overline{R}_p + A\cdot B\cdot R_p$
$R_s(A,B,R_p) = \overline{A}\cdot B\cdot R_p + A\cdot\overline{B}\cdot R_p + A\cdot B\cdot\overline{R}_p + A\cdot B\cdot R_p$

Étape 4 : Minimisation Algébrique

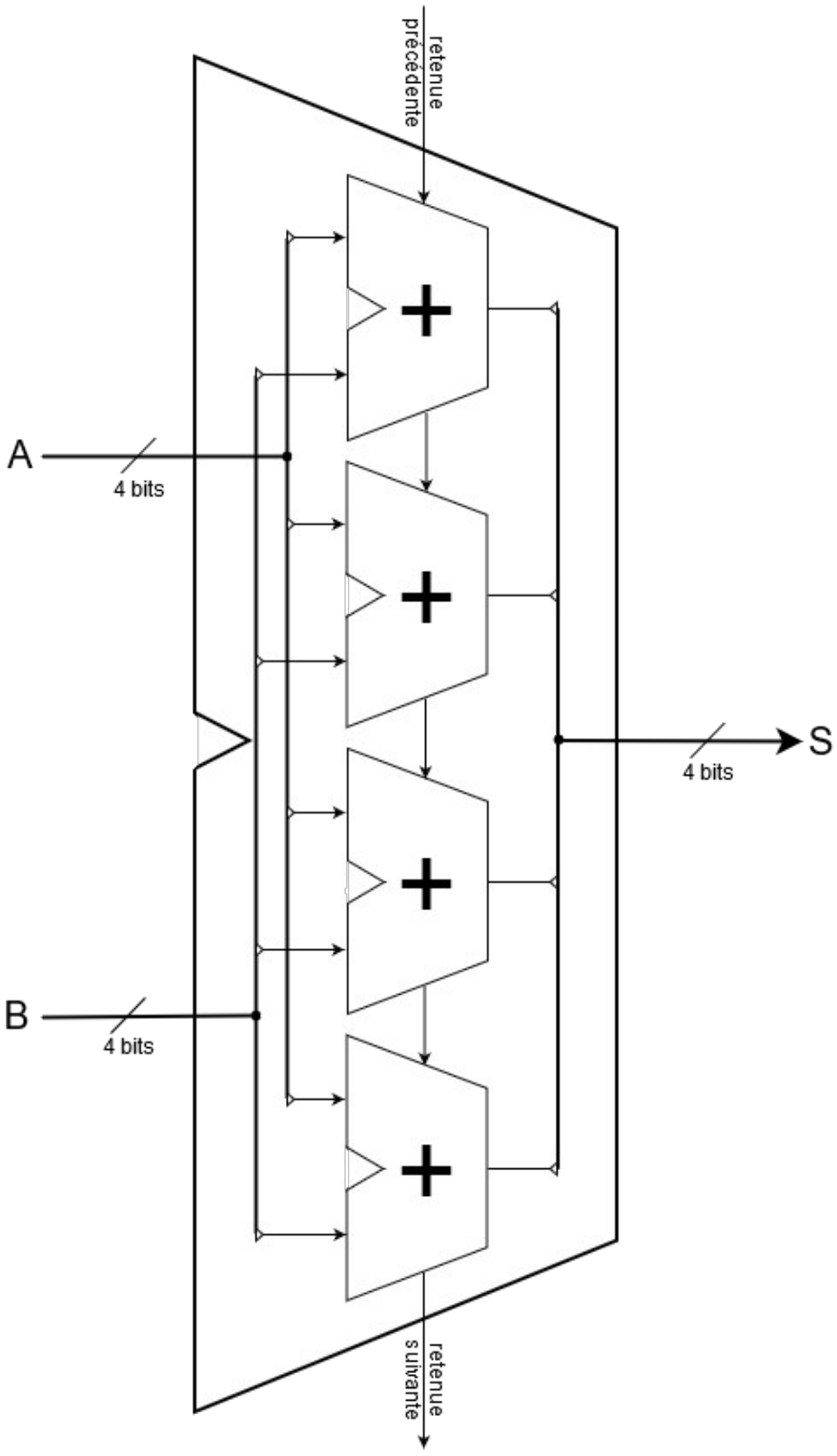
$S(A,B,R_p) = \overline{A}\cdot\overline{B}\cdot R_p + \overline{A}\cdot B\cdot\overline{R}_p + A\cdot\overline{B}\cdot\overline{R}_p + A\cdot B\cdot R_p$
$S(A,B,R_p) = \overline{A}\cdot(\overline{B}\cdot R_p + B\cdot\overline{R}_p) + A\cdot(\overline{B}\cdot\overline{R}_p + B\cdot R_p)$
$S(A,B,R_p) = \overline{A}\cdot(B\oplus R_p) + A\cdot(B\otimes R_p)$
$S(A,B,R_p) = \overline{A}\cdot(B\oplus R_p) + A\cdot\overline{(B\oplus R_p)}$
$S(A,B,R_p) = A\oplus(B\oplus R_p) = A\oplus B\oplus R_p$

$R_s(A,B,R_p) = \overline{A}\cdot B\cdot R_p + A\cdot\overline{B}\cdot R_p + A\cdot B\cdot\overline{R}_p + A\cdot B\cdot R_p$
$R_s(A,B,R_p) = \overline{A}\cdot B\cdot R_p + A\cdot\overline{B}\cdot R_p + A\cdot B\cdot\overline{R}_p + A\cdot B\cdot R_p + A\cdot B\cdot R_p + A\cdot B\cdot R_p$
$R_s(A,B,R_p) = (\overline{A}\cdot B\cdot R_p + A\cdot B\cdot R_p) + (A\cdot\overline{B}\cdot R_p + A\cdot B\cdot R_p) + (A\cdot B\cdot\overline{R}_p + A\cdot B\cdot R_p)$
$R_s(A,B,R_p) = (B\cdot R_p) + (A\cdot R_p) + (A\cdot B)$



Additionneur 4 bits :

La construction de l'additionneur 4 bits en utilisant les additionneurs complets ce fait par ce qu'on appelle la composition en cascade, dans laquelle chaque additionneur 1 bit représente une colonne (de 1 bit) sur l'opération de l'addition, la retenue suivante d'un additionneur en sortie est passée comme entrée en retenue précédente pour le bit suivant comme la représentation d'une cascade.



Remarque 1: Le petit triangle sur les bus à 4 bits A, B et S sont des indicateur pour dénoter qu'un seul fil parmi les 4 fils a été extirpé du bus.

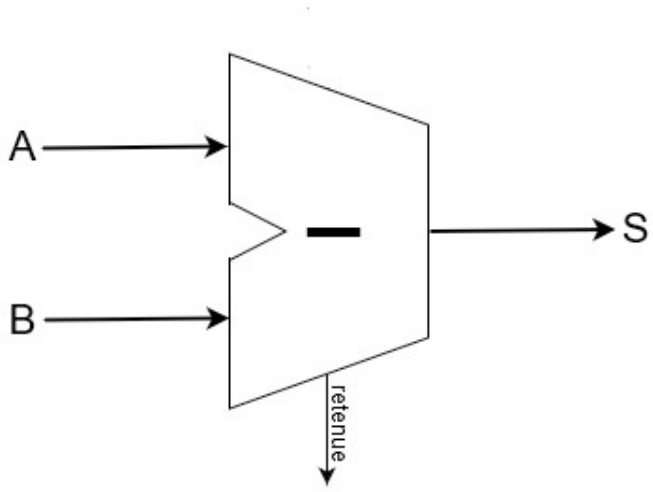
Remarque 2: La condition pour que l'additionneur sur 4 bits fonctionne correctement, mis tout seul ou en cascade avec d'autres additionneur 4 bits, est que l'entrée retenue précédente du premier additionneur doit être mise à 0, le premier n'a pas de retenue précédente.

2)
a/

Demi-Soustracteur

même principe que l'additionneur est appliqué au soustracteur

Étape 1 : Schéma global



Remarque : La notion de retenue ici est différente de celle de l'addition, c'est une retenue prêtée de la colonne suivante qui doit être rendue.

Étape 2 : Table de Vérité

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

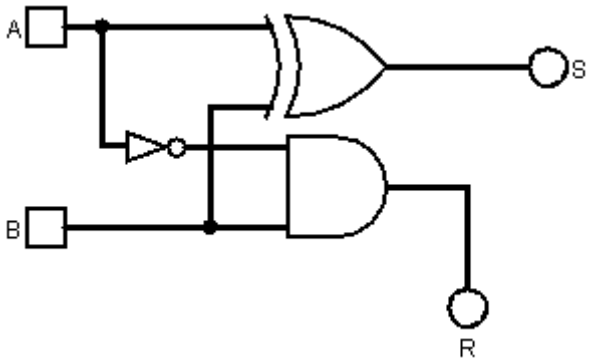
Étape 3 : Fonctions Canoniques Disjonctives

$S(A,B) = \overline{A}\cdot B + A\cdot\overline{B}$
$R_s(A,B) = \overline{A}\cdot B$

Étape 4 : Minimisation Algébrique

$S(A,B) = \overline{A}\cdot B + A\cdot\overline{B}$
$S(A,B) = A \oplus B$

$R(A,B) = \overline{A}\cdot B$
Étape 5 : Logigramme



Soustracteur-complet

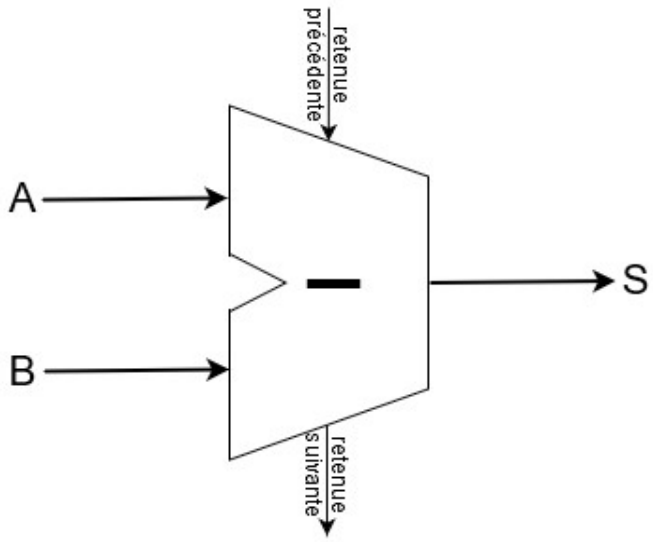
La construction d'un soustracteur 4 bits Complément-à-2/Valeur Absolue à partir de d'additionneur 1 bit.

$$\begin{array}{r} {\scriptstyle 1}0\,{\scriptstyle 1}1\,{\scriptstyle 1}0\,1 \\ -\,0\,1\,1\,0 \\ \hline = \;1\,1\,1\,1 \end{array}$$

On observe que l'opération de osustraction Complément-à-2/Valeur Absolue sur un seul bit, par exemple sur le 2-ème bit (2-ème colonne sur le schéma), on peut appliquer comme pour l'additionneur un soustracteur sur 1 bit qui doit avoir 3 entrées qui sont le 2-ème chiffre dans A, le 2-ème chiffre dans B et la retenue du calcul du bit précédent (du 1-ier bit/1-iere colonne), et 2 sorties qui sont la Somme et la retenue pour le calcul du bit suivant (pour le 3-ème bit/3-ème colonne).

Remarque : Comme pour l'additionneur, le soustracteur peut être utilisé pour l'encodage Complément-à-2 et Valeur Absolue, mais pour le soustracteur en encodage Valeur Absolue, il est impérative que A doit être supérieure ou égale a B.

Étape 1 : Schéma global



Étape 2 : Table de Vérité

A	B	R_p	S	R_s
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

R_p : Retenue précédente
R_s : Retenue suivante

Étape 3 : Fonctions Canoniques Disjonctives

$S(A,B,R_p) = \overline{A}\cdot\overline{B}\cdot R_p + \overline{A}\cdot B\cdot\overline{R_p} + A\cdot\overline{B}\cdot\overline{R_p} + A\cdot B\cdot R_p$
$R_s(A,B,R_p) = \overline{A}\cdot\overline{B}\cdot R_p + \overline{A}\cdot B\cdot\overline{R_p} + \overline{A}\cdot B\cdot R_p + A\cdot B\cdot R_p$

Étape 4 : Minimisation Algébrique

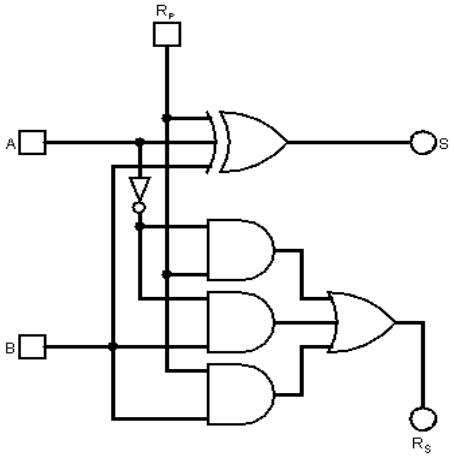
$S(A,B,R_p) = \overline{A}\cdot\overline{B}\cdot R_p + \overline{A}\cdot B\cdot\overline{R_p} + A\cdot\overline{B}\cdot\overline{R_p} + A\cdot B\cdot R_p$
$S(A,B,R_p) = \overline{A}\cdot(\overline{B}\cdot R_p + B\cdot\overline{R_p}) + A\cdot(\overline{B}\cdot\overline{R_p} + B\cdot R_p)$
$S(A,B,R_p) = \overline{A}\cdot(B\oplus R_p) + A\cdot(B\otimes R_p)$
$S(A,B,R_p) = \overline{A}\cdot(B\oplus R_p) + A\cdot(\overline{B\oplus R_p})$
$S(A,B,R_p) = A\oplus(B\oplus R_p) = A\oplus B\oplus R_p$

$R_s(A,B,R_p) = \overline{A}\cdot\overline{B}\cdot R_p + \overline{A}\cdot B\cdot\overline{R}_p + \overline{A}\cdot B\cdot R_p + A\cdot B\cdot R_p$

R_p \ AB	00	01	11	10
0	0	1	0	0
1	1	1	1	0

$R_s(A,B,R_p) = (B\cdot R_p) + (\overline{A}\cdot R_p) + (\overline{A}\cdot B)$

Étape 5 : Logigramme

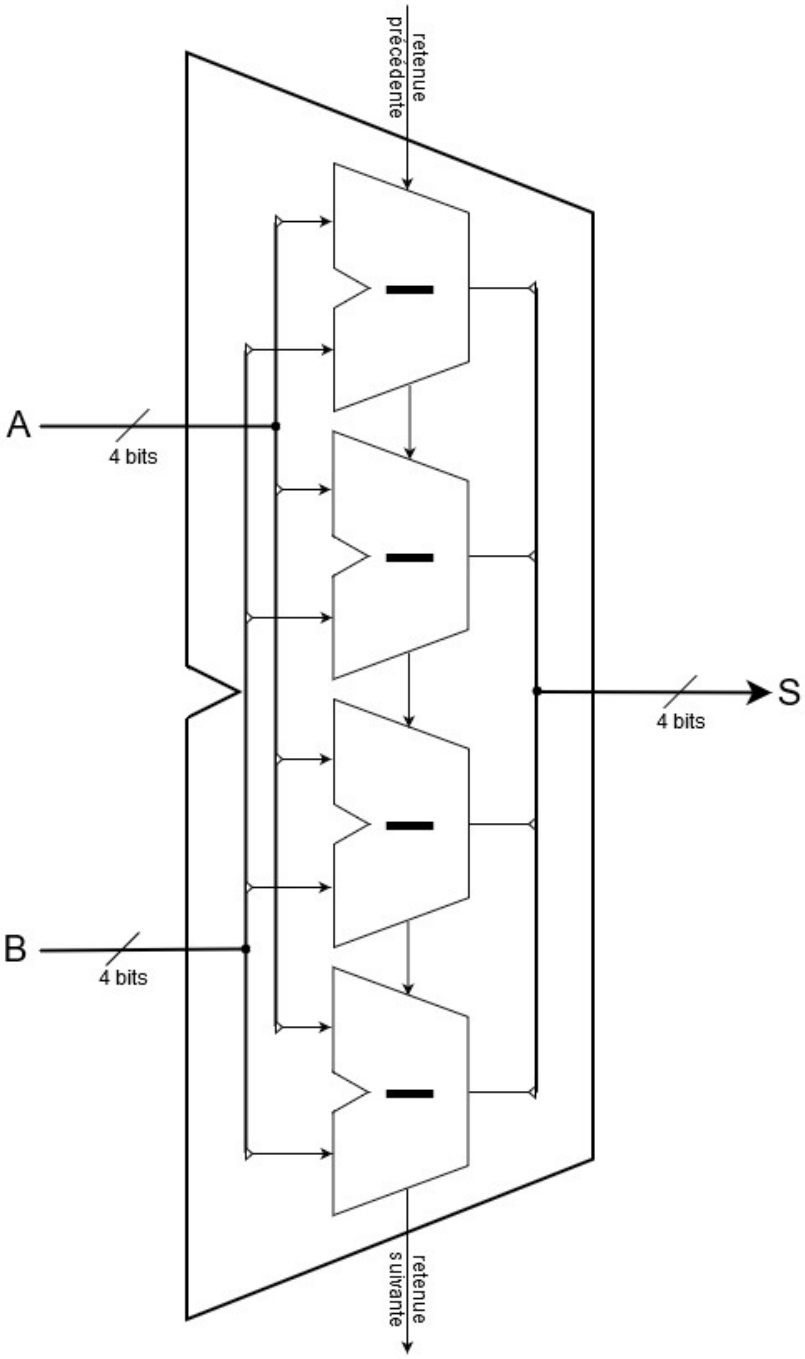


Soustracteur 4 bits :

La construction du soustracteur 4 bits en utilisant les soustracteurs complets ce fait exactement de la même manière que l'additionneur 4 bits, en utilisant le principe de la composition en cascade.

Remarque 1: Pour fonctionner correctement la retenue du 1^{-ier} bit du soustracteur doit être mise à 0.

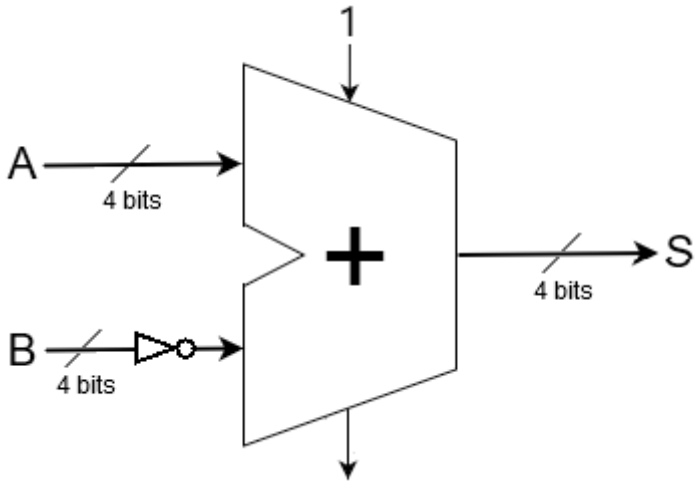
Remarque 2: Les soustracteurs 4 bits comme pour les additionneurs 4 bits peuvent être mis en cascade pour former des additionneurs/soustracteurs de 8, 12, 16, 20... bits.



b/ Construction d'un soustracteur 4 bits en utilisant un additionneur 4 bits

La construction d'un soustracteur à partir d'un additionneur permet de réduire le nombre de portes et la complexité de l'UAL, puisque au-lieu de créer 2 circuits différents l'un pour l'addition et l'autre pour la soustraction, un unique additionneur suffit pour faire le travail des 2.

Sachant qu'on utilise l'encodage Complément-à-2, on peut modifier l'opération de la soustraction comme suite : A – B = A + (-B), et en Complément-à-2 (-B) = $\overline{B}$ + 1($\overline{B}$ étant le Complément-à-1). Donc la formule devient A – B = A + $\overline{B}$ + 1, et on peut la réaliser comme suite :

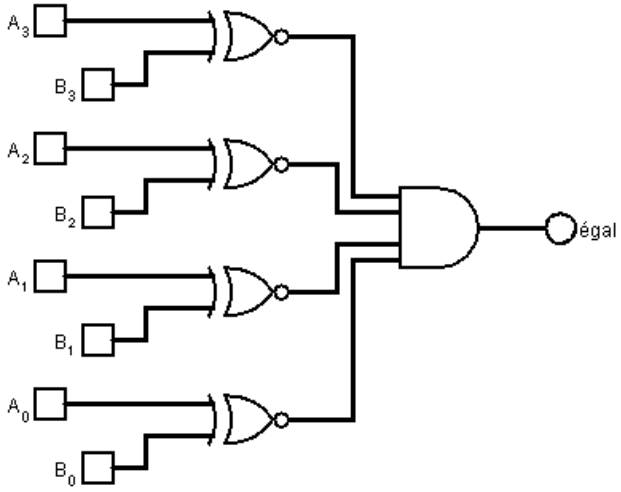


Remarque : le +1 est réalisé par la mise-à-1 de la retenue précédente de l'additionneur.

3)

Une solution simple est de faire une comparaison de l'égalité de 2 nombres en utilisant des portes XNOR. On observant la table de vérité en bas à gauche de la porte XNOR on constate que XNOR retourne 1 que si A = 1 et B = 1 ou A = 0 et B = 0, ainsi une porte XNOR permet de comparer 2 bits et dire s'ils sont égaux ou pas. Le circuit en bas à droite fait une comparaison entre 2 valeurs sur 4 bits.

A	B	A ⊗ B
0	0	1
0	1	0
1	0	0
1	1	1



Remarque 1: Un comparateur complet doit contenir des sorties pour supérieur-à et inférieur-à. La solution pour construire un comparateur sur 4 bits en utilisant la méthode à 5 étapes est trop longue est fastidieuse, le mieux est de construire un comparateur à base de composition en cascade en utilisant des comparateurs sur 1 bit.

Remarque 2: Pour une UAL, il est préférable de ne pas construire un comparateur du tout, la soustraction permet de le remplacer, en faisant la soustraction de A-B les signaux Zéro et Négative de méta-information permettent de déduire la comparaison entre A et B. Ainsi si A-B = 0 ça implique A = B (Zéro = 1), et si A-B est négatif (Négative = 1) ça implique A < B,...et ainsi de suite.

Exercice 02 :

a/

- Combien de bits d'information peuvent être transmis simultanément sur une ligne électrique ?

R: 1 bit d'information peut être transmis simultanément sur une ligne électrique.

- Comment peut-on transmettre simultanément 4 bits ?

R: Il nous faut 4 lignes (fils) électriques pour transmettre simultanément 4 bits en parallèle.

- Soit un bus de données de 8 bits. Quel est le plus petit nombre binaire que l'on peut y représenter ? Et le
plus grand ?

R: Soit un bus de données de 8 bits:
Le plus petit nombre binaire que l'on peut y représenter = 00000000 (0)
Le plus grand nombre binaire que l'on peut y représenter = 11111111 $(255)_{VA}$

- Soit un bus d'adresse de 2 bits. Combien d'adresses différentes peut-on y représenter ? Et pour 20 bits?

R: Soit un bus d'adresse de 2 bits: On peut y représenter 4 adresses différentes.
Avec un bus d'adresse de 20 bits, On peut représenter 2^{20} adresses différentes, un espace de 1 Méga mots.

b/

- Quel est l'espace adressable par un processeur 16 bits à 32 bits d'adresse ?

R: L'espace adressable par un processeur 16 bits à 32 bits d'adresse = 2^{32} mots mémoire = 4 Giga mots.

- Combien de pattes " adresse " y-a-t-il sur un module de mémoire de 1 Mo (mots de 8 bits) ?

R: Sur un module mémoire de 1 Mo (mots de 8 bits), il y a $Log_2(2^{20}) = 20$ pattes adresses.

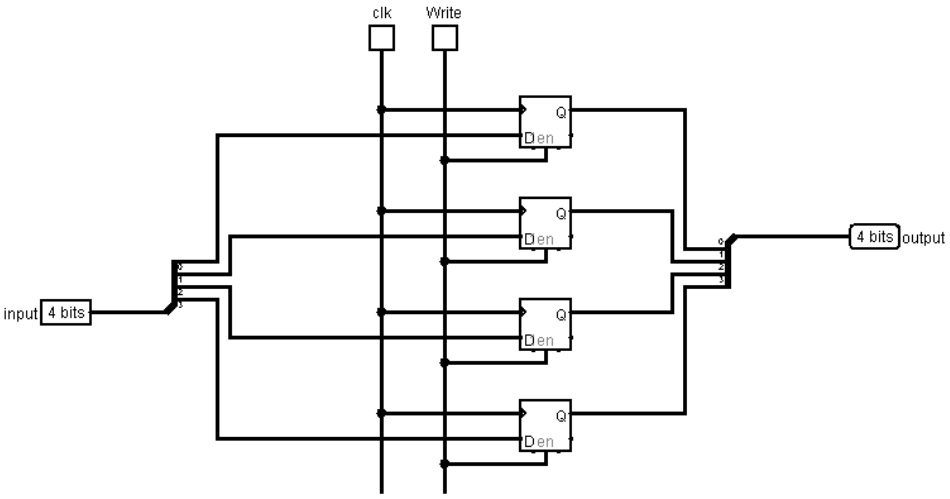
c/

- Dans une architecture Von Neumann : où sont les données ? où sont les programmes ?

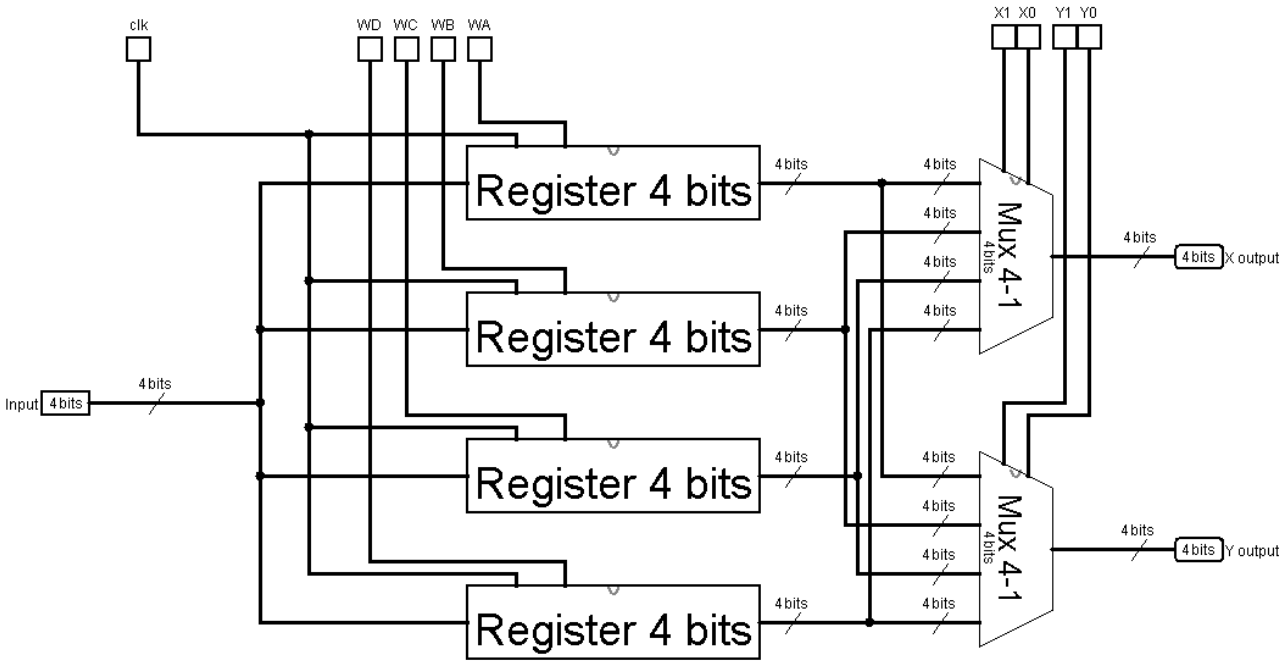
R: Dans une architecture Von Neumann : les données et les programmes sont dans la même mémoire, contrairement à l'architecture de Harvard où les données et les programmes sont séparés sur des mémoires différentes.

Exercice 03 :

1. registre 4 bits avec une commande d'écriture :



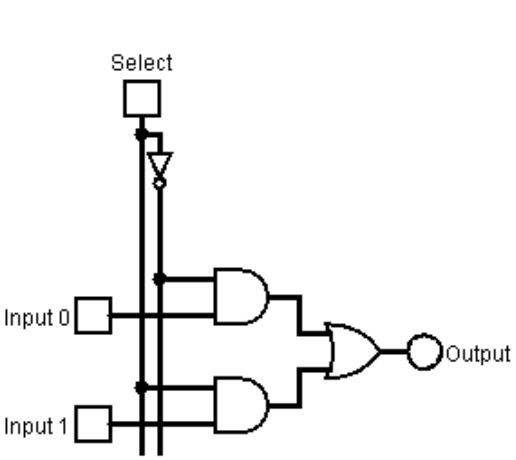
2. le Banc de Registres (Register File) :

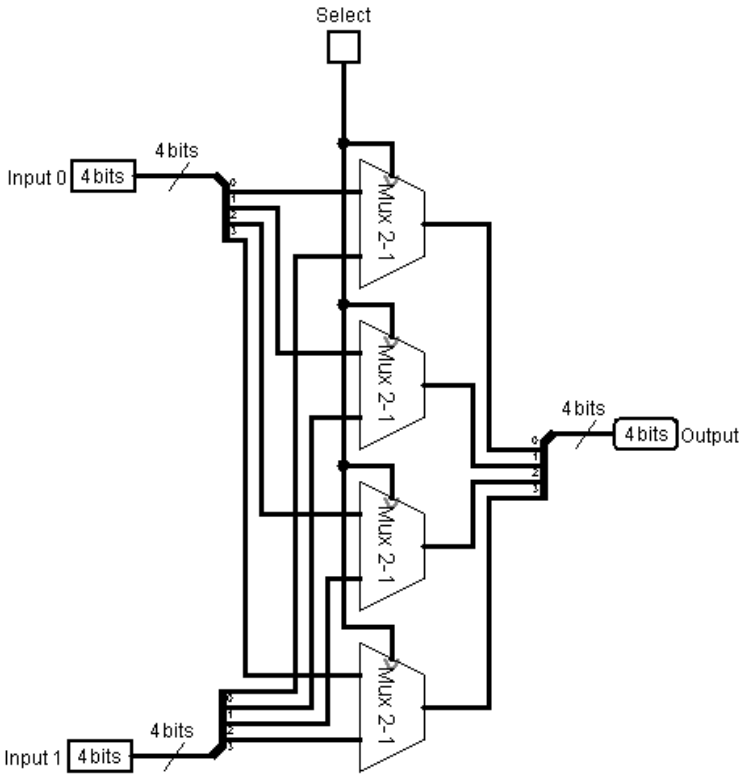


Le banc de registres est une collection ou un regroupement de registres, il se compose de 4 registres à 4 bits, avec une entrée pour l'écriture sur l'un des 4 registres, et 2 sorties X et Y qui sont normalement destinées a être raccordées aux entrées A et B de l'UAL. Les 2 multiplexeurs 4-1 permettent de choisir un parmi les 4 registres pour prendre une sortie donnée, un pour la sortie X et l'autre pour la sortie Y. les commandes X0, X1, Y0, Y1 permettent de choisir la sortie d'un registre sur l'une des 2 sorties X et Y, et les commandes WA, WB, WC, WD en suivant le schéma en haut, permettent de choisir sur quelle registre l'écriture de l'entrée doit se faire.

Sur le schéma en peut apercevoir que les 2 multiplexeurs 4-1 sont sur 4 bits, ils sont un peu différents des multiplexeurs 4-1 normaux (sur 1 bit) dans le sens où ils font passer 4 bits en même temps sur une seul entrée, leurs conception est un peu différente.

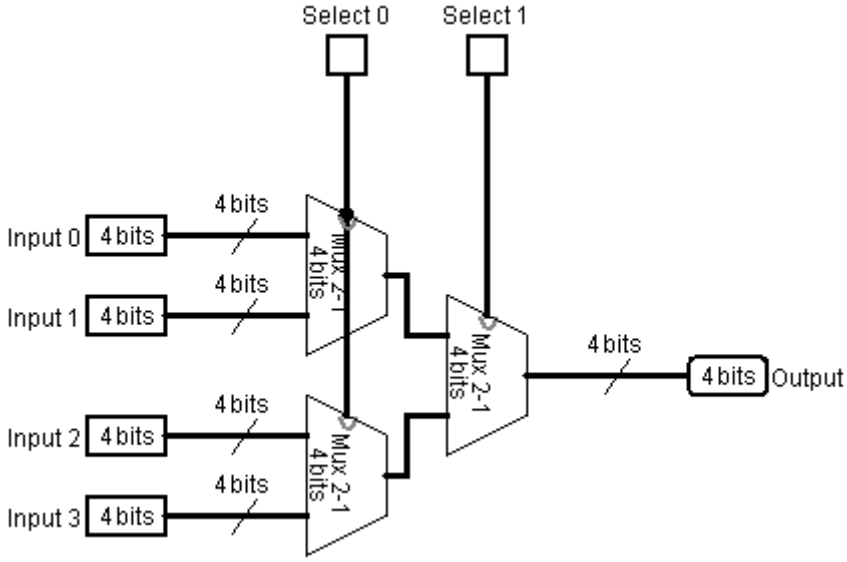
Pour concevoir un multiplexeurs 4-1 sur 4bits en va utiliser la composition en cascade, pour cela en commence par un multiplexeur 2-1, comme sur le schéma en bas à gauche avec la méthode à 5 étapes, et ensuite l'augmente en multiplexeur 2-1 sur 4 bits simplement par la duplication de 4 multiplexeur 2-1, chacun pour 1 seul bit et de partager le même sélecteur pour les 4 multiplexeur 2-1 comme sur le schéma en bas à droite.





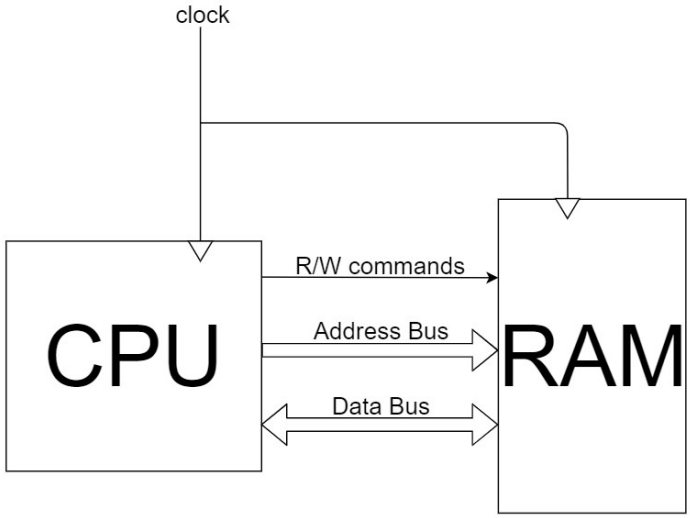
Remarque : Notez la forme du spliteur sur le schéma à droite (capturé de Logisim), à ne pas confondre avec un fil/bus avec plusieurs sorties, comme sur le bus de l'entrée du banc de registres. Un spliteur permet de répartir/amasser plusieurs fils du même bus.

Un multiplexeurs 4-1 4 bits est construit en composition en cascade à partir de multiplexeur 2-1 4bits selon le schéma en bas.

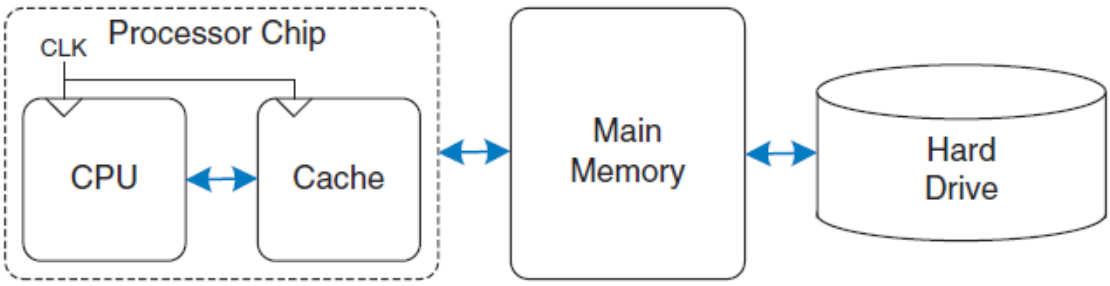


Exercice 04 :

a/ Une structure basique de l'architecture d'une machine est constituée d'un processeur et d'une RAM, comme sur le diagramme suivant :



Actuellement dans les architectures modernes, c'est devenu comme ceci :



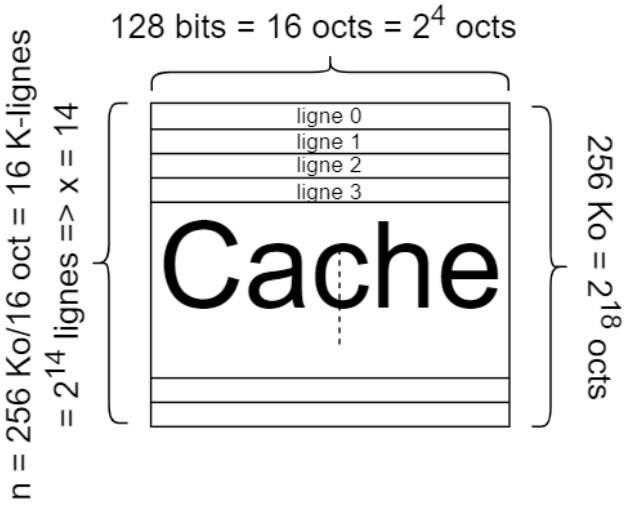
Le cache a vu le jour dans les années 90, principalement né à cause de l'évolution de vitesse non monotone des 2 composants, le CPU et la RAM. Le CPU avec au fur et à mesure est devenu beaucoup plus rapide que la RAM, et il lui fallait plusieurs cycles d'horloges d'attente pour accéder à une information dans la RAM, le cache était la réponse pour ce problème. Il représente réellement une mémoire tampon de petite taille avec la même vitesse que le CPU étant construite avec la même technologie, de telle sorte que les portions de programmes ou de données susceptible d'être utilisées par le processeur sont chargé à l'avance dans le cache et profiter ainsi de sa vitesse.

Remarque 1: Le disque dur représenté dans le schéma en haut représente la mémoire virtuelle (ou mémoire de pagination), il complémente la mémoire centrale lorsque celle-ci est insuffisante, mine de rien la mémoire virtuelle est beaucoup plus lente que la RAM.

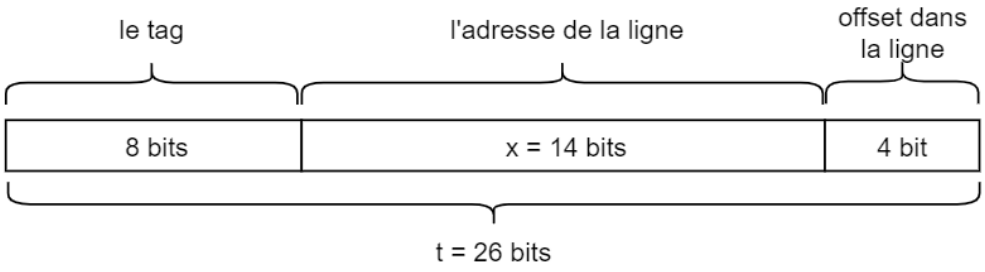
Remarque 2: Les 3 mémoires, mémoire cache, RAM et mémoire virtuel sont dans l'ordre de la plus rapide à la plus lente, et de la plus petite à la plus grande. Cette structuration est appelée la hiérarchie mémoire et l'accès à l'information dans cette hiérarchie par le processeur est géré par un circuit hardware dans le CPU appelé MMU (Memory Management Unit).

D'après l'image on peut déduire que la taille de la ligne est de 16 octets. Ce qui nécessite 4 bits pour les adresser dans une seule ligne.

N le nombre de lignes est de 16384 lignes selon le schéma, et il faut 14 bits pour les adresser dans le cache, donc x=14.



En ajoutant les 8 bits du tag mentionnés dans l'énoncé, ça nous mène à conclure comme sur le schéma en bas, que la taille de l'adresse mémoire est de t = 26 bits, et que la taille de la mémoire est de 2^{26} octs = 64 Mo.



Découpage de l'adresse

Remarque 1: il faut savoir qu'en général un processeur détient 2 caches, un cache de données pour contenir les données du programmes, et un cache du programme qui contient les instructions du programme.

Remarque 2: Le MMU est responsable de choisir les pages de la RAM qui devraient être chargées dans le cache, et celle qui devraient être supprimées du cache, et ça selon une stratégie bien précise.

Remarque 3: Il existe plusieurs stratégies de gestion de cache, et chaque processeur adopte l'une d'entre elles, la meilleurs stratégie est celle qui arrive a deviner à l'avance les informations que le processeur va demander, et c'est variable en fonction du type de l'application exécutée sur la machine.

Remarque 4: L'étude du modèle de fonctionnement de ces stratégies est complexe, et il est au-delà du niveau de ce module, quoique c'est excellemment bien expliqué dans le livre **Digital Design and Computer Architecture** dans le Chapitre 8 partie **8.3. caches**.

b/ Il tout a fait possible d'arranger les données d'une manière séquentiel dans un cache, et plus encore, c'est la manière normale et usuelle du fonctionnement du cache puisque il copies les données de la RAM par bloque (ligne de 16 octs). Dans le cas où les données consécutives se trouve sur la frontière entre 2 lignes, le MMU n'a qu'a charger les 2 lignes.

Le cache avec le pipeline sont considérés comme les 2 majeurs avancés technologiques des années 90 dans les architectures des ordinateurs. Le cache arrive a faire un booste énorme dans les performance de la machine, et ça en grande partie à cause de la nature des programmes informatiques, dans lesquels les instructions sont consécutives dans la plupart du temps ce qui augmente à plus de 90% le ration du Hit. De même pour les données, les tableaux sont généralement stockés d'une façon consécutives optimisant ainsi l'utilisation du cache.

Remarque : Dans la littérature de la technologie du cache, le *Hit* désigne l'étape lorsque le processeur demande une information et celle-ci se trouve dans le cache, donc c'est un accès immédiat. Et inversement le *Miss* est lorsqu'il ne trouve pas l'information dans le cache, ce qui implique que le MMU doit charger le bloque contenant l'information de la RAM, ce qui exige plusieurs cycles d'horloges perdus.