Algorithmes et Pensée Computationnelle

Architecture des ordinateurs

Le but de cette séance est de comprendre le fonctionnement d'un ordinateur. La série d'exercices sera axée autour de de conversions en base binaire, décimale ou hexadécimal, de calcul de base en suivant le modèle Von Neumann.

1 Conversions

Question 1: (*5 minutes*) **Conversion** $Base_{10}$ **-** $Base_2$

1. Convertir le nombre $10_{(10)}$ en base 2.
2. Convertir le nombre $45_{(10)}$ en base 2.
3. Convertir le nombre $173_{(10)}$ en base 2.

Conseil

Vous pouvez utiliser la table de puissance de 2 pour vous aider !
TODO : Ajouter la table de puissance de 2.

Solution

Il existe 2 méthodes pour simplifier la conversion d'un nombre de la base 10 à la base 2. Ici le nombre 45 sera entièrement développé.

Méthode 1 :

Prenez votre nombre et divisez le par 2 en colonne. S'il y a un reste, notez le à coté, sinon notez 0. Répétez l'opération avec le résultat que vous venez d'obtenir, et ce jusqu'à arriver à 0. Pour lire votre nombre en binaire, prenez la suite de 0 et 1 correspondants aux différents restes, mais prenez les de bas en haut.

nombre	**reste**
45	-
22	1
11	0
5	1
2	1
1	0
0	1

Ici, on obtient $101101_{(2)}$

Méthode 2 :

Cette deuxième méthode consiste à utiliser une table de puissances de 2 que voici, ensuite on regarde la plus grande valeur de cette table qui peut être soustraite à notre nombre. Une fois trouvé, on le soustrait, on ajoute 1 sous la case correspondante et on répète l'opération avec le résultat obtenu. Répètez l'opération jusqu'à obtenir 0

Ici, 45 - 32 = 13, puis 13- 8 = 5, puis 5-4 = 1, puis 1-1 = 0

puissance	2^5	2^4	2^3	2^2	2^1	2^0
valeur	32	16	8	4	2	1
binaire	1	0	1	1	0	1

On obtient donc $101101_{(2)}$

Les réponses sont les suivantes :

1. $10_{(10)} = 1010_{(2)}$
2. $45_{(10)} = 101101_{(2)}$
3. $173_{(10)} = 10101101_{(2)}$

Question 2: (*15 minutes*) **Conversion** $Base_{10}$ **-** $Base_3$**,** $Base_8$**,** $Base_{16}$

1. Convertir le nombre $40_{(10)}$ en base 8.
2. Convertir le nombre $52_{(10)}$ en base 3.
3. Convertir le nombre $254_{(10)}$ en base 16.

Conseil

Vous pouvez réutiliser les méthodes présentées dans la solution du premier exercice.

N'oubliez pas qu'en hexadécimal, A vaut 10, B vaut 11, C vaut 12, D vaut 13, E vaut 14 et F vaut 15 !

Solution

Dans cette solution, la méthode 1 sera utilisée pour 52 en base 3, et la méthode 2 pour 254 en base 16.

Méthode 1 pour 52 (ici, nous sommes en base 3 donc on divise par 3) :

nombre	reste
52	-
17	1
5	2
1	2
0	1

Ici, on obtient $1221_{(3)}$

Méthode 2 :

Ici, il faut prendre en compte le nombre de fois qu'on peut multiplier le nombre par la valeur de la puissance. Ici on a 256 qui est trop grand, il faut donc partir sur 16. On voit qu'on peut aller jusqu'à 15*16 qui vaut 240. Donc on entre 15 sous la valeur 16 et on soustrait. 254-240 = 14. Il nous reste donc 14*1, on met alors 14 sous la valeur 1.

puissance	16^2	16^1	16^0
valeur	256	16	1
hexa	0	15	14

On obtient donc 15 14 (qui s'écrit $FE_{(16)}$)

Les réponses sont les suivantes :

1. $40_{(10)} = 50_{(8)}$
2. $52_{(10)} = 1221_{(3)}$
3. $254_{(10)} = FE_{(16)}$

Question 3: (*15 minutes*) **Conversion** $Base_3$ **-** $Base_{16}$ **en** $Base_8$

1. Convertir le nombre $10110_{(2)}$ en base 10.
2. Convertir le nombre $4321_{(5)}$ en base 10.
3. Convertir le nombre $ABC_{(16)}$ en base 10.

Conseil

Ici utilisez les tables de puissances utilisées pour la méthode 2 (présentée dans la solution du premier exercice).

N'oubliez pas qu'en hexadécimal, A vaut 10, B vaut 11, C vaut 12, D vaut 13, E vaut 14 et F vaut 15 !

Solution

Reprenons la table des puissances utilisée précédemment (exemple avec $ABC_{(16)}$) :

Ici, A vaut 10, B vaut 11 et C vaut 12

puissance	16^2	16^1	16^0
valeur	256	16	1
hexa	10	11	12

Ici on obtient donc 10*256 + 11*16 + 12*1 = 2560 + 176 + 12 = $2748_{(10)}$

1. Convertir le nombre $10110_{(2)} = 22_{(10)}$
2. Convertir le nombre $4321_{(5)} = 586_{(10)}$
3. Convertir le nombre $ABC_{(16)} = 2748_{(10)}$

2 Arithmétique binaire

Question 4: (*15 minutes*) **Addition de nombres binaires**

1. Additionner $\mathbf{01010101}_{(2)}$ et $\mathbf{10101010}_{(2)}$
2. Additionner $\mathbf{01011111}_{(2)}$ et $\mathbf{10000001}_{(2)}$
3. Additionner $\mathbf{01110100}_{(2)}$ et $\mathbf{00011010}_{(2)}$

Conseil

Table d'addition binaire :

a	b	s=a+b	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Solution

Il faut utiliser la table d'addition. Pour commencer, entrez a et b, puis commencez à additionner chaque ligne (si vous additionnez de haut en bas, n'oubliez pas d'entrer les nombres à additionner de bas en haut et de lire le résultat de bas en haut également, étant donné qu'on commence par additionner la fin des nombres, comme lors d'une addition en colonne). Si l' addition vaut 1, le résultat est 1 et le reste est 0. Si l'addition vaut 2, le résultat est 0 et le reste est 1. Si l'addition vaut 3, le résultat est 1 et le reste est 1. N'oubliez pas d' additionner également le reste à chaque ligne !

Voici un exemple pour le deuxième point :

a	b	s=a+b	r
1	1	0	1
1	0	0	1
1	0	0	1
1	0	0	1
1	0	0	1
0	0	1	0
1	0	1	0
0	1	1	0

On obtient donc $11100000_{(2)}$

1. Additionner $01010101_{(2)} + 10101010_{(2)} = 11111111_{(2)}$
2. Additionner $01011111_{(2)} + 10000001_{(2)} = 11100000_{(2)}$
3. Additionner $01110100_{(2)} + 00011010_{(2)} = 10001110_{(2)}$

Question 5: (*15 minutes*) **Soustraction de nombres binaires**

Effectuer les opérations suivantes :

1. $01111111_{(2)}$ - $01000000_{(2)}$
2. $10000000_{(2)}$ - $00000001_{(2)}$
3. $10101010_{(2)}$ - $01010101_{(2)}$

Conseil

Table de soustraction binaire :

a	b	s=a-b	r
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Solution

Il faut utiliser la table de soustraction. Pour commencer, entrez a et b, puis commencez à soustraire chaque ligne (si vous soustrayez de haut en bas, n'oubliez pas d'entrer les nombres à soustraire de bas en haut et de lire le résultat de bas en haut également, étant donné qu'on commence par soustraire la fin des nombres, comme lors d'une soustraction en colonne). Si la soustraction vaut 0, le résultat est 0 et le reste est 0. Si la soustraction vaut 1, le résultat est 1 et le reste est 0. Si la soustraction vaut -1, le résultat est 1 et le reste est 1. N'oubliez pas de soustraire le reste !

Voici un exemple pour le deuxième point :

a	b	s=a-b	r
0	1	1	1
0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1
1	0	0	0

On obtient donc $1111111_{(2)}$

1. $01111111_{(2)}$ - $01000000_{(2)}$ = $111111_{(2)}$
2. $10000000_{(2)}$ - $00000001_{(2)}$ = $111111_{(2)}$
3. $10101010_{(2)}$ - $01010101_{(2)}$ = $1010101_{(2)}$

Exemple :

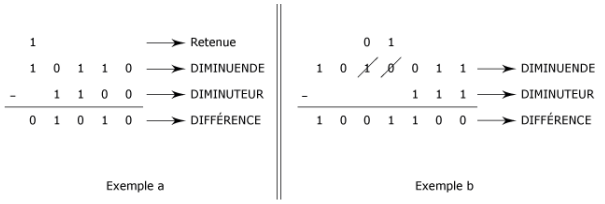

FIGURE 1 – Exemple de soustraction de nombres binaires

3 Conversion et arithmétique

Question 6: (*20 minutes*) **Conversion et addition :**
Effectuer les opérations suivantes :

1. $111101_{(2)} + 110_{(2)} = ..._{(10)}$
2. $111111_{(2)} + 000001_{(2)} = ..._{(10)}$
3. $127_{(10)} + ABC_{(16)} = ..._{(8)}$

Conseil

TODO : Conseils pour convertir puis additionner des nombres de bases différentes.

Solution

Présenter les étapes détaillées permettant d'aboutir à la solution.

Question 7: (*20 minutes*) **Conversion et soustraction :**
Effectuer les opérations suivantes :

1. $101010_{(2)} - 010101_{(2)} = ..._{(10)}$
2. $64_{(10)} - 001000_{(2)} = ..._{(10)}$
3. $FFF_{(10)} - 127_{(10)} = ..._{(2)}$

Conseil

TODO : Conseils pour convertir puis soustraire des nombres de bases différentes.

Solution

Présenter les étapes détaillées permettant d'aboutir à la solution.

4 Modèle de Von Neumann

Dans cette section, nous allons simuler une opération d'addition dans le **modèle de Van Neumann**, il va vous être demandé à chaque étape (FDES) de donner la valeur des registres.

État d'origine :
A l'origine, notre **Process Counter (PC)** vaut **00100000**.

Dans la mémoire, les instructions sont les suivantes :

Adresse	Valeur
00011111	00100100
00100000	10110110
00100001	11101101

Les registres sont les suivantes :

Registre	Valeur
00	11100011
01	01101100
10	00100101
11	00000000

Les opérations disponibles pour l'unité de contrôle sont les suivantes :

Numéro	Valeur
00	MOV
01	XOR
10	ADD
11	SUB

Question 8: (*5 minutes*) **Fetch**
À la fin de l'opération **FETCH**, quelles sont les valeurs du **Program Counter** et de l'**Instruction Register** ?

Conseil

Pour rappel, l'**unité de contrôle (Control Unit)** commande et contrôle le fonctionnement du système. Elle est chargée du **séquençage** des opérations. Après chaque opération **FETCH**, la valeur du **Program Counter** est incrémentée (valeur initiale + 1).

Solution

À la fin de l'opération **Fetch**, le **Program Counter** vaudra **00100001** tandis que l'**Instruction Register** vaudra **10110110**, ce qui correspond à la valeur de l'adresse mémoire **00100000**.

Question 9: (*5 minutes*) **Decode**

1. Quelle est la valeur de l'opération à exécuter ?
2. Quelle est l'adresse du registre dans lequel le résultat doit être enregistré ?
3. Quelle est la valeur du premier nombre de l'opération ?
4. Quelle est la valeur du deuxième nombre de l'opération ?

Conseil

Pensez à décomposer la valeur de l'**Instruction Register** pour obtenir toutes les informations demandées.
Les données issues de la décomposition de l'**Instruction Register** ne sont pas des valeurs, mais des références. Trouver les tables concordantes pour y récupérer les valeurs.

Solution

En décomposant l'**Instruction Register** (10110110), on obtient les données suivantes :

- **10**, correspond à l'opération à effectuer,
- **11**, correspond à **l'adresse** du registre où sera sauvegardé le résultat,
- **01**, correspond à **l'adresse** du premier nombre,
- **10**, correspond à **l'adresse** du deuxième nombre.

À partir de ces informations, on peut répondre aux questions posées :

1. Valeur de l'opération : **ADD**
2. Adresse du registre dans lequel le résultat doit être enregistré : **11**
3. Premier nombre de l'opération : **01101100**
4. Deuxième nombre de l'opération : **00100101**

Question 10: (*5 minutes*) **Execute**
Quel est résultat de l'opération ?

Conseil

Toutes les informations permettant d'effectuer l'opération se trouvent dans les données de l'**Instruction Register**.

Solution

$$\begin{array}{cccccccc} & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ + & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{array}$$