Cryptographie avancée - TP 1
ZZ3 F5 - Réseaux et Sécurité Informatique
Manipulation de bits

L'objectif de ce TP est de développer une compréhension du fonctionnement du compilateur et d'évaluer son efficacité lorsqu'il s'agit d'effectuer des opérations de base sur des entiers. Pour ce faire, nous allons effectuer des opérations élémentaires tout en tenant compte de la représentation binaire des variables manipulés afin de réécrire les opérations sur les entiers. Lorsque vous répondez aux questions, votre but sera de fournir les réponses les plus optimales, c'est-à-dire celles qui requièrent le moins d'opérations pour votre processeur.

Prise en main

1. Au moyen de la fonction `sizeof` déterminer le nombre d'octet de plusieurs types (`int`, `unsigned int`, `short int`, `float`, `double`,...).

Permutation de registres

2. Ecrire une fonction permettant de la permuter les valeurs de deux variables en n'utilisant que deux registres. Pensez à utiliser l'opérateur XOR. Généraliser la méthode avec l'opérateur + d'un groupe.

Opération sur des entiers positifs

A présent nous allons nous passer des opération de bases `+`, `-`, `*`, `/` et `%`. Nous privilégieons les opérateurs `&`, |, `^` ainsi que `<<` et `>>`.

3. Écrire une fonction permettant de diviser par deux un nombre pair positif efficacement. Généraliser à la division par 2^i de nombres positifs multiples de 2^i.

4. Écrire une fonction permettant de calculer le reste de la division par deux d'un nombre entier positif efficacement. Généraliser à la division par 2^i d'un nombre entier positif.

5. Écrire une fonction permettant de multiplier par deux un nombre entier positif efficacement. Généralsier l'approche lorsque l'on veut multiplier un nombre entier positif par un nombre fixé creux en base binaire (exemple : 33 ou 34).

Déplacement de bits.

6. Écrire une fonction permettant d'effectuer la rotation d'un vecteur booléen de longueur 32. Pensez à utiliser un type non signé.

7. Écrire une fonction permettant d'inverser efficacement l'ordre des bits d'un vecteur booléen de longueur 32.

Multiplication matrice/vecteur binaires.

8. Déclarer un tableau à deux dimensions (carré) et un tableau à une dimension de même taille.

9. Écrire des fonctions permettant de remplir les objets avec des 0 et des 1 de façon aléatoire.

10. Écrire une fonction permettant de calculer les produite matrice/vecteur de façon classique.

11. Écrire une fonction permettant de calculer le produit matrice/vecteur de ce vecteur par un très grande puissance de la matrice, de façon classique.

Si l'ordre de la matrice est inférieur à 32 (ou 64), il est possible d'être plus efficace en stockant

les lignes de la matrice dans une variable de type `unsigned int`. La matrice complète peut donc être dans un tableau dont les éléments sont de type `unsigned int`. Le vecteur peut également être stocké dans une variable de même type.

Génération de matrices et de vecteurs.

12. Écrire une macro modifiant le $i^{\text{ème}}$ (en partant de la droite) d'un élément de type `unsigned int`.
13. Écrire une macro modifiant le $i^{\text{ème}}$ (en partant de la droite) du $k^{\text{ème}}$ d'un tableau dont les éléments sont de type `unsigned int`.

Calcul de la parité d'un vecteur.

14. Écrire une fonction permettant de calculer la parité d'un vecteur de longueur 32 (64) bits.
15. Écrire une fonction permettant de calculer un tableau renvoyant la parité d'un vecteur 0/1 de longueur 16.
16. Écrire une fonction permettant de calculer la parité d'un vecteur de longueur inférieur à 32 (64) bits en utilisant le tableau précédemment généré.

Affichage des matrices et vecteurs.

17. Écrire une macro renvoyant le $i^{\text{ème}}$ bit (en partant de la droite) d'un élément de type `unsigned int`.
18. Écrire une macro renvoyant le $i^{\text{ème}}$ (en partant de la droite) du $k^{\text{ème}}$ d'un tableau dont les éléments sont de type `unsigned int`.
19. Écrire une fonction affichant les éléments de la matrice.
20. Écrire une fonction affichant les éléments du vecteur.

Produit matrice/vecteur.

21. Écrire une fonction permettant de calculer le produit matrice/vecteur en utilisant la fonction de parité.
22. Écrire une fonction permettant de calculer le produit matrice/vecteur en utilisant le tableau qui renvoie la parité (cette fonction ne s'appliquera qu'à des éléments de taille 16).
23. Écrire une fonction permettant de calculer le produit matrice/vecteur en utilisant la fonction de parité qui se base sur le tableau qui renvoie la parité.

Comparaison entre les produits mactrice/vecteur.

Écrire une fonction permettant de clauler le produit matrice/vecteur d'un vecteur par une très grande puissance d'une matrice donnée de façon classique (en multipliant successivement) :

24. en utilisant la fonction de parité écrite plus haut.
25. en utilisant le tableau qui renvoie la parité (cette fonction ne s'appliquera qu'à des élément de taille 16).
26. en utilisant la fonction de parité qui se base sur le tableau qui renvoie la parité.

Tester ces fonctions avec les différentes options de compilation (sans option, -O1, -O2, -O3) et pour des tailles de matrice 16 et 32 (64). Que constatez-vous ? Afin de faciliter les tests, on pourra tirer parti des arguments de la fonction main et utiliser la fonction `time` lorsque vous lancez l'exécutable. Expliquez pourquoi il y a une différence entre les temps "Real", "User", "Sys".