Processeur et OS
Matthieu Boyer
Planche A

Khôlle MP2I
23/09

Question de Cours

Calculer $255 \wedge 71$.

Bitwise Tricks

Dans cet exercice, les réponses seront données en pseudo-code, puis traduites en formules logiques utilisant les opérateurs de shift (» et «), la soustraction et les opérations logiques traditionnelles ($\neg, \wedge, \vee, \oplus$). On utilisera `sizeof(int)` pour la taille de la représentation mémoire d'un entier et on pourra utiliser les opérations `int` et `unsigned int` qui considèrent un entier signé ou non. On pourra aussi utiliser `CHAR_BIT`, le nombre de bits dans un octet. On pourra utiliser toutes les opérations précédemment construites.

1. Calculer le signe d'un entier. Il doit valoir 1 si $v \geq 0$ et -1 sinon.
2. En déduire une méthode pour détecter si deux entiers ont même signe.
3. Calculer la valeur absolue d'un entier sans branchement (`if statement`).
4. Donner une méthode pour déterminer si un nombre est une puissance de deux.
5. Donner une méthode pour compter le nombre de bits valant 1 dans un nombre. On pourra d'abord faire une itération par bit dans le nombre, puis on cherchera une méthode pour faire autant d'itération que de bits valant 1 dans le nombre.
6. Donner une méthode pour fusionner des bits de deux valeurs selon un masque. C'est à dire, étant donné un masque, on prend les bits de a si les bits du masque valent 0, sinon on prend ceux de b.

Solution: Bitwise Tricks

Voir `https://graphics.stanford.edu/~seander/bithacks.html` pour les corrections et d'autres trucs du genre.

1. On a : `sign = -(v < 0)` . Ce qui peut causer des branchements sur certains processeurs. On propose donc :

```c
int v; // we want to find the sign of v
        int sign; // the result goes here
// or, to avoid branching on CPUs with flag registers (IA32):
sign = -(int)((unsigned int)((int)v) >> (sizeof(int) * CHARBIT - 1));
```

   Dans la suite on ne donne plus un code `c` fonctionnel complet à chaque question.

2. On propose : `bool f = ((x ^ y) < 0);`
3. On propose : `v + (v >> sizeof(int) * CHAR_BIT - 1) ^ (v >> sizeof(int) * CHAR_BIT - 1)`
4. On propose : `v && !(v & (v - 1))`
5. On propose : `for (c = 0; v; c++) { v &= v -1; }`
6. On propose : `a ^ ((a ^ b) & mask)`

Processeur et OS
Matthieu Boyer
Planche B

Khôlle MP2I
23/09

Question de Cours

Calculer $\neg(184 \oplus 37)$

Sur la Turing-Complétude de la Porte NAND

Si P et Q sont deux assertions, on note $P \uparrow Q$ et on appelle barre de Sheffer de P et Q l'assertion $\neg(P \wedge Q)$. A l'aide uniquement de la barre de Sheffer, construire des assertions équivalentes à :

1. $\neg P$
2. $P \wedge Q$
3. $P \vee Q$
4. $P \oplus Q$
5. $P \Rightarrow Q$
6. $P \Leftrightarrow Q$

Dessiner les portes logiques équivalentes. Faire le même exercice avec la porte NOR. En déduire que savoir construire une porte NAND ou une porte NOR suffit pour reconstruire n'importe quel fonnction logique. On essaiera de formaliser le résultat avec une procédure similaire à une récurrence.

Solution: Sur la Turing-Complétude de la Porte NAND

1. $\neg P = P \uparrow P$
2. Si $R = P \uparrow Q$, $P \wedge Q = R \uparrow R$.
3. $P \vee Q = \neg (\neg P \wedge \neg Q)$.
4. $P \Rightarrow Q = \neg (P \vee Q)$.
5. $P \Rightarrow Q = P \Longrightarrow Q \wedge Q \Longrightarrow P$.

Processeur et OS
Matthieu Boyer
Planche C

Khôlle MP2I
23/09

Question de Cours

Calculer $\neg\,(193 \oplus \neg 17)$

Incrémentatation

L'opérateur le plus pratique est le multiplexeur : un circuit permettant de concentrer sur une même voie de transmission différents types de liaisons (informatique, télécopie, téléphonie, télétex) en sélectionnant une entrée parmi N. Il possédera donc une sortie et N entrées, ainsi qu'une entrée C de commande de $\log_2 N$ bits permettant de choisir quelle entrée sera sélectionnée.

1. Donner la table de vérité d'un multiplexeur à 2 entrées.
2. Écrire une fonction booléenne représentant le multiplexeur.
3. Réaliser un multiplexeur comme un circuit logique.

Une grande partie des additions d'un ordinateur consistent à ajouter 1 à une autre valeur x.

4. Réaliser un incrémenteur 3 bits : Entrées A0A1A2, Sorties RI0I1I2.
5. Réaliser un décrémenteur 3 bits : Entrées A0A1A2, Sorties D0D1D2.
6. Réaliser la fonction D0 avec un Mux de 3 entrées.
7. Réaliser la fonction D1 avec un DEMux de 3 entrées.
8. Réaliser la fonction D2 avec un DEMux de 1×4 et un Mux de 4×1.

Solution: Incrémentatation

`https://mlichouri.wordpress.com/wp-content/uploads/2013/10/circombrevsol.pdf`