Organisation des ordinateurs

Examen de première session 2019

Énoncés et solutions

Énoncés

1. En Belgique, un numéro de téléphone mobile est formé
   — d'un préfixe de la forme $04c_1c_2$, où c_1 est un chiffre de 5 à 9 et c_2 un chiffre quelconque, et
   — d'un suffixe composé de 6 chiffres, dont le premier est non nul.

[1/20] (a) En supposant que tous les numéros de téléphone mobile valables sont équiprobables, calculez la quantité d'information contenue dans un numéro.

[1/20] (b) Un opérateur téléphonique gère 2 millions de numéros de téléphone mobile belges, regroupés dans une base de données qui associe à chaque numéro le code postal de son titulaire. En sachant qu'il y a 2825 codes postaux distincts, supposés équiprobables et indépendants des numéros de téléphone, calculez la quantité de mémoire nécessaire au stockage de cette base de données.

[1/20] (c) Quelle quantité d'information un disque dur vendu comme possédant une capacité de 1 TB permet-il de mémoriser ?

[4/20] 2. (a) Quel est le nombre représenté par la constante `0x80000001` dans les représentations
   — par valeur signée ?
   — par complément à un ?
   — par complément à deux ?
   — IEEE754 simple précision ?

[1/20] (b) Quels sont les nombres dont les représentations sur n bits par complément à un et par complément à deux sont égales ? (Justifiez votre réponse.)

[1/20] (c) En utilisant la notation hexadécimale, donnez une représentation d'une valeur indéterminée (*Not A Number*) dans le système IEEE754 en double précision.

[1/20] 3. (a) À quoi servent les drapeaux d'un processeur ? Comment un programmeur peut-il les utiliser ?

(b) Décrivez, le plus simplement possible, l'effet des fragments de code assembleur x86-64 suivants. (On suppose que la pile est correctement configurée avant leur exécution.)

[2/20] — Fragment 1 :

```
PUSH RBX
PUSH RAX
MOV  AL, byte ptr[RSP+0]
MOV  BL, byte ptr[RSP+7]
MOV  byte ptr [RSP+0], BL
MOV  byte ptr [RSP+7], AL
POP  RAX
POP  RBX
```

[2/20] — Fragment 2 :

```
   PUSH RDX
   CMP  AX, 0
   JGE  p
n: MOV RDX, -1
   MOV DX, AX
   JMP f
p: MOV RDX, 0
   MOV DX, AX
f: MOV RAX, RDX
   POP RDX
```

[1/20] (c) Expliquez comment une valeur est représentée en mémoire dans le système petit-boutiste.

4. Dans le cadre du développement d'une application de calcul statistique, on souhaite programmer une fonction `histogramme` capable de calculer l'histogramme d'un tableau d'octets donné. Cette fonction prend pour entrée
   — l'adresse d'un tableau d'octets `ts`,
   — la taille `n` de ce tableau, exprimée sur 64 bits,
   — l'adresse d'un tableau `th` contenant 256 entiers de 64 bits, non initialisés.

   À l'issue de l'exécution de cette fonction, chaque case de `th` doit contenir le nombre d'octets de `ts` égaux à son indice. Par exemple, la cinquième case `th[4]` de `th` contiendra le nombre d'octets de `ts` égaux à 4. La fonction ne retourne rien.

[1/20] — Écrire, en pseudocode ou en langage C (au choix), un algorithme permettant de résoudre ce problème.

[4/20] — Traduire cet algorithme en assembleur x86-64, en veillant à respecter la convention d'appel de fonctions des systèmes *Unix*.

Exemple de solutions

1. (a) Le chiffre c_1 peut prendre 5 valeurs possibles et c_2 10 valeurs possibles, indépendamment l'un de l'autre. Toutes les valeurs étant équiprobables, la quantité d'information contenue dans un préfixe vaut donc

$$\log_2 5 + \log_2 10 \approx 5{,}644 \text{ bits.}$$

Le premier chiffre du suffixe peut prendre 9 valeurs possibles, et les 5 autre chiffres 10 valeurs, indépendamment et de façon équiprobable. La quantité d'information contenue dans un suffixe vaut donc

$$\log_2 9 + 5\log_2 10 \approx 19{,}780 \text{ bits.}$$

La quantité d'information totale d'un numéro vaut donc

$$5{,}644 + 19{,}780 \approx 25{,}423 \text{ bits.}$$

(b) La réponse à la question dépend du procédé utilisé pour représenter le contenu de la base de données. Une solution simple serait de mémoriser un tableau à 2 millions d'entrées, contenant chacune un numéro de télephone et le code postal qui lui est associé.

On sait déjà que chaque numéro de téléphone représente 25,423 bits d'information. La quantité d'information nécessaire à un code postal vaut quant à elle

$$\log_2 2825 \approx 11{,}464 \text{ bits.}$$

La quantité totale d'information nécessaire au stockage de la base de données vaut donc

$$2\,10^6(25{,}423 + 11{,}464) \approx 73{,}775 \text{ Mbits.}$$

(c) Un tel disque permet de mémoriser 10^9 octets, c'est-à-dire

$$\frac{10^9}{2^{30}} \approx 0{,}931 \text{ TB.}$$

2. (a) Il s'agit d'un nombre représenté sur 32 bits, dont seuls le bit de poids fort et le bit de poids faible sont égaux à 1.
   — Par valeur signée, le nombre représenté vaut -1.
   — Par complément à un, il s'agit de

$$-\sum_{i=1}^{30} 2^i = -2^{31} + 2.$$

   — Par complément à deux, on obtient

$$-2^{32} + 2^{31} + 1 = -2^{31} + 1.$$

— Dans le format IEEE754 en simple précision, on a un bit de signe égal à 1, un exposant égal à -127 (ce qui indique une mantisse dénormalisée), et une mantisse égale à 2^{-22}. Le nombre représenté vaut donc

$$-2^{-127}\,2^{-22} = -2^{-149}.$$

(b) On sait qu'un nombre positif (y compris le zéro positif) se représente de façon identique par complément à un et par complément à deux. En revanche, la représentation d'un nombre négatif v par complément à deux correspond à la représentation de $v + 1$ par complément à un. Seuls les nombres positifs (y compris le zéro positif) possèdent donc des représentations identiques par complément à un et par complément à deux.

(c) Pour obtenir une valeur indéterminée, il faut choisir un bit de signe quelconque, la valeur la plus grande possible de l'exposant (c'est-à-dire 1024 pour la double précision), et une mantisse dont un moins un bit de la représentation est égal à un. En choisissant un bit de signe égal à 0 et une mantisse dont seul le premier bit est égal à 1, on obtient la représentation binaire

$$0\,11111111111\,10000\,\cdots\,000,$$

qui correspond à

0x7FF8000000000000

en notation hexadécimale.

3. (a) Les drapeaux d'un processeur fournissent certaines informations sur le résultat d'une opération précédemment effectuée, par exemple le fait qu'une opération arithmétique a produit un report, un dépassement ou un résultat nul.

Le programmeur peut utiliser ces drapeaux par le biais d'une instruction de saut conditionnel, qui effectue un saut à condition que l'état des drapeaux satisfasse une condition donnée.

(b) — Fragment 1 : Ce code sauvegarde (en les empilant) le contenu des registres `RBX` et `RAX`, et puis permute les octets de poids fort et de poids faible dans le bloc de 8 octets situé en sommet du pile, qui correspond donc à la valeur sauvegardée de `RAX`. Les valeurs de `RAX` et de `RBX` sont ensuite restaurées.
L'effet de ce code est donc de permuter les 8 bits de poids fort et les 8 bits de poids faible du registre `RAX`.

— Fragment 2 : Ce code détermine si le contenu de `AX` (considéré signé) est positif ou négatif. S'il est positif ou nul, le programme construit une valeur de 64 bits dont les 16 bits de poids faible sont égaux au contenu de `AX`, et les autres bits sont nuls. Si le contenu de `AX` est en

revanche négatif, on construit une valeur de 64 bits dont les 16 bits de poids faible sont égaux au contenu de `AX`, et les autres bits sont égaux à 1. Dans les deux cas, la valeur obtenue est recopiée dans `RAX`.
Ce code a donc pour effet d'étendre sur les 64 bits de RAX la valeur signée contenue dans ses 16 bits de poids faible.

(c) La valeur à mémoriser est découpée en octets. L'octet de poids faible est placé à l'adresse la plus petite, puis l'octet suivant est placé à l'adresse suivante, et ainsi de suite jusqu'à l'octet de poids fort qui sera placé à l'adresse la plus élevée.

4. — Une solution simple consiste à d'abord initialiser à zéro toutes les cases du tableau `th`. Ensuite, il suffit de parcourir le tableau `ts` et, pour chacune des valeurs qu'il contient, d'incrémenter la case correspondante de `th`.

```
void histogramme(unsigned char *ts, unsigned long n,
                 unsigned long *th)
{
  unsigned long  i;

  for (i = 0; i < 256; i++)
    th[i] = 0;

  for (i = 0; i < n; i++)
    th[ts[i]]++;
}
```

5. Le code assembleur suivant est une traduction directe de ce programme C :

```
              .text
histogramme:  PUSH  RBP
              MOV   RBP, RSP

              MOV   RAX, 0
boucle1:      CMP   RAX, 0x100
              JAE   suite1
              MOV   qword ptr[RDX + 8*RAX], 0
              INC   RAX
              JMP    boucle1

suite1:       MOV   RAX, 0
boucle2:      CMP   RAX, RSI
              JAE   suite2
              MOV   RCX, 0
```

```
                MOV   CL, byte ptr[RDI + RAX]
                INC   qword ptr[RDX + 8*RCX]
                INC   RAX
                JMP   boucle2

suite2:         POP   RBP
                RET
```