Organisation des ordinateurs
Examen de juin 2025
Énoncés et solutions

Énoncés

1. Le réseau informatique d'une entreprise est analysé continuellement par un dispositif de sécurité. Celui-ci produit toutes les 5 millisecondes un signal de 3 bits indiquant l'état du réseau. Les codes utilisés sont les suivants :

   — `000` : Activité normale,
   — `001` : Activité suspecte faible,
   — `010` : Activité suspecte moyenne,
   — `011` : Activité suspecte grande,
   — `100` à `111` : Attaque détectée et son type.

   En moyenne, 93% des signaux concernent une activité normale, 2% une activité suspecte de chaque type, et 1% la détection d'une attaque. Les différents types d'attaques sont équiprobables.

   (a) Calculer la quantité d'information moyenne d'un signal.
   (b) De quelle quantité de mémoire, exprimée en mégaoctets, le dispositif de sécurité a-t-il besoin pour retenir l'ensemble de signaux produits pendant une semaine ? On suppose que l'analyse du réseau n'est jamais interrompue, et que les signaux sont mémorisés de façon optimale.

2. (a) Déterminer les nombres encodés sur 32 bits par 0xFFFFFF25 selon les représentations
      — non-signée,
      — par valeur signée,
      — par complément à un,
      — par complément à deux,
      — en virgule fixe par complément à deux, avec 16 bits avant et 16 bits après la virgule,
      — IEEE754 en simple précision.
   (b) Calculer le produit $(-\frac{7}{2}) \times \frac{5}{2}$ en complément à deux en virgule fixe. Vous êtes libre de choisir la taille des représentations des opérandes et du résultat, ainsi que la position de la virgule.
   (c) Calculer la somme $27 + (-22)$ en complément à un sur 6 bits.

(d) Une fonction arithmétique exprime son résultat dans le format IEEE754 en simple précision. Donner une représentation hexadécimale de ce résultat dans le cas où il correspond à un nombre plus grand que le plus grand nombre représentable de façon exacte.

3. (a) Qu'est-ce que le bus interne d'un processeur et à quoi sert-il ? Décrire au moins deux composants du processeur reliés au bus, et expliquer brièvement leur rôle.

(b) Quels sont les drapeaux levés par l'instruction `ADD EAX, EBX`, dans le cas où l'on a initialement `EAX` = 0x7FFFFFFF et `EBX` = 0x00000001 ? (Justifier votre réponse.)

(c) Expliquer étape par étape comment se déroulera l'exécution du fragment de code assembleur x86-64 suivant, en faisant l'hypothèse que les cellules de la mémoire qui sont adressées correspondent à de la mémoire vive. Quelle sera la valeur finale de `EAX` ?

```
MOV RBX, 0x200
MOV EAX, 0x12344321
MOV dword ptr[RBX], EAX
ADD word ptr[RBX + 2], AX
XOR byte ptr[RBX], AH
AND byte ptr[RBX + 1], AL
MOV AX, word ptr[RBX + 2]
ADD word ptr[RBX], AX
MOV EAX, dword ptr[RBX]
```

4. On souhaite programmer une fonction `compter_pairs` acceptant en arguments un tableau d'entiers signés `t` et le nombre d'éléments `n` de ce tableau. Cette fonction doit retourner le nombre de valeurs paires contenues dans `t`. Par exemple, si `t` contient $[-2, -1, 0, 1, -3, 4]$, alors la fonction doit retourner 3. Si `t` contient $[1, -1, 1]$ ou si `t` est vide, alors elle doit retourner 0.

(a) Écrire, en pseudocode ou en langage C (au choix), un algorithme permettant de résoudre ce problème.

(b) Traduire cet algorithme en un programme assembleur x86-64, en veillant à respecter la convention d'appel de fonctions des systèmes *Unix*.

Exemples de solutions

1. (a) La quantité d'information apportée par un signal est égale à

$$\log_2 \frac{1}{0{,}93} \approx 0{,}105 \text{ bit}$$

dans le cas d'une activité normale,

$$\log_2 \frac{1}{0{,}02} \approx 5{,}644 \text{ bits}$$

pour une activité suspecte de chacun des trois types, et

$$\log_2 \frac{1}{0{,}01} \approx 6{,}644 \text{ bits}$$

lors de la détection d'un attaque. En moyenne, le signal contient donc

$$0{,}93 \,.\, 0{,}105 + 3 \,.\, 0{,}02 \,.\, 5{,}644 + 0{,}01 \,.\, 6{,}644 \approx 0{,}502 \text{ bit}$$

d'information.

(b) En une semaine, il y a

$$200 \,.\, 3600 \,.\, 24 \,.\, 7 = 120{,}96 \,.\, 10^6$$

signaux émis. La quantité de mémoire nécessaire vaut donc

$$120{,}96 \,.\, 10^6 \,.\, 0{,}502 \approx 60{,}775 \,.\, 10^6 \text{ bits,}$$

c'est-à-dire

$$\frac{60{,}775 \,.\, 10^6}{8 \,.\, 2^{20}} \approx 7{,}245 \text{ MB.}$$

2. (a) — *Représentation non signée :* Il faut ajouter 0xDA = 218 à ce nombre pour obtenir 0xFFFFFFFF, qui correspond au plus grand nombre représentable sur 32 bits, c'est-à-dire $2^{32} - 1$. Le nombre recherché vaut donc $2^{32} - 219 = 4294967077$.

— *Représentation par valeur signée :* Le nombre recherché est l'opposé de 0x7FFFFF25, qui par un raisonnement similaire au point précédent vaut $2^{31} - 219$. La réponse est donc $-2^{31} + 219 = -2147483429$.

— *Représentation par complément à un :* Il s'agit d'un nombre négatif. Son complément à un possède l'encodage 0x000000DA qui représente 218. Le nombre vaut donc -218.

— *Représentation par complément à deux :* Pour les nombres négatifs, le nombre représenté par complément à deux est inférieur d'une unité à celui représenté par complément à un. Le nombre recherché vaut donc -219.

— *Représentation en virgule fixe :* Il y a 16 chiffres après la virgule, donc le nombre recherché est 2^{16} fois plus petit que celui obtenu pour le complément à deux. Ce nombre vaut donc

$$-\frac{219}{2^{16}} \approx -0{,}003342.$$

— *Représentation IEEE754 :* Le bit de signe est égal à 1, l'exposant est le plus grand possible, et la mantisse n'est pas nulle. On a donc une valeur indéfinie négative $-NaN$.

(b) Chacun des deux opérandes se représente avec 3 bits avant et 1 bit après la virgule. Le résultat de l'opération a donc besoin de 6 bits avant et 2 bits après la virgule. On choisit alors d'effectuer l'opération à l'aide de nombres représentés sur 8 bits, avec 1 bit après la virgule pour chaque opérande.

	1	1	1	1	1	0	0	1
$\times$	0	0	0	0	0	1	0	1
	$\boxed{1}$	$\boxed{1}$						
	1	1	1	1	1	0	0	1
$+$	1	1	1	0	0	1		
	1	1	0	1	1	1	0	1

Le résultat 110111,01 représente bien le nombre $-8{,}75$.

(c)

$\boxed{1}$	$\boxed{1}$	$\boxed{1}$		$\boxed{1}$	$\boxed{1}$		
	0	1	1	0	1	1	
$+$	1	0	1	0	0	1	
	0	0	0	1	0	0	
$+$	0	0	0	0	0	1	
	0	0	0	1	0	1	

Le résultat représente bien le nombre 5.

(d) Pour un dépassement arithmétique vers le haut, le bit de signe vaut 0, l'exposant est le plus grand possible, et tous les bits de la mantisse sont nuls. On a donc la représentation hexadécimale 0x7F800000.

3. (a) Le bus interne est un canal de communication qui relie les composants du processeur qui sont amenés à s'échanger des données. Il est connecté en particulier à la banque de registres, qui est une petite quantité de mémoire vive utilisable par les opérations du processeur, et à l'unité arithmétique et logique (ALU) qui, comme son nom l'indique est le composant du processeur qui effectue les opérations de traitement des données, en particulier les opérations arithmétiques et logiques.

(b) Cette opération ne produit pas de report à la position 32 et ne conduit pas à un résultat nul, donc CF et ZF sont baissés. Le résultat 0x80000000 de cette opération possède un bit de signe égal à 1, donc SF est levé. Enfin, si les nombres sont considérés signés, cette opération additionne deux nombres positifs pour obtenir un résultat négatif, ce qui indique un dépassement arithmétique. Cette opération lève donc OF.

(c) — Les deux premières instructions placent respectivement les valeurs 0x12344321 dans les 32 bits de poids faible de `RAX`, et 0x200 dans `RBX`.

— L'instruction suivante écrit successivement les quatre octets 0x21, 0x43, 0x34 et 0x12 à partir de l'adresse 0x200 en mémoire.

— L'instruction `ADD word ptr[RBX + 2], AX` ajoute 0x4321 à la valeur de 16 bits située à l'adresse 0x202, qui vaut initialement 0x1234. Les octets situés aux adresses 0x202 et 0x203 deviennent donc tous deux égaux à 0x55.

— L'instruction `XOR` s'applique aux deux octets 0x21 et 0x43, et écrit donc 0x62 à l'adresse 0x200 de la mémoire.

— L'instruction `AND` s'applique aux deux valeurs 0x43 et 0x21, et écrit donc l'octet 0x01 à l'adresse 0x201. On a donc a ce stade à partir de l'adresse 0x200 les octets 0x01, 0x63, 0x55 et 0x55.

— L'instruction `MOV` qui suit copie donc la valeur 0x5555 dans les 16 bits de poids faible de `RAX`.

— L'instruction suivante ajoute cette valeur à l'entier de 16 bits présent à l'adresse 0x200 de la mémoire, qui vaut initialement 0x0162. Cet entier devient donc égal à 0x56B7.

— La dernière instruction copie dans `EAX` le nombre de 32 bits situé à partir de l'adresse 0x200 de la mémoire, qui vaut 0x555556B7.

4. (a) Pour déterminer si un nombre est pair, il suffit de regarder si le bit de poids faible de sa représentation est nul.

```
unsigned compter_pairs(signed char t[], unsigned n)
{
  unsigned i, nb;

  for (i = 0, nb = 0; i < n; i++)
    if ((t[i] & 0x01) == 0x00)
      nb++;

  return nb;
}
```

(b)

```
        .intel_syntax noprefix
        .text
        .global compter_pairs
        .type compter_pairs, @function
compter_pairs:
        PUSH RBP
        MOV  RBP, RSP
        MOV  RAX, 0
        MOV  RCX, 0
boucle: CMP  ECX, ESI
        JAE  fin
        MOV  DL, byte ptr[RDI + RCX]
        INC  ECX
        AND  DL, 0x01
        JNZ  boucle
        INC  EAX
        JMP  boucle
fin:    POP  RBP
        RET
```