

# Organisation des ordinateurs

## Énoncés et solutions de l'examen de seconde session 2018

### Énoncés

- [1/20] 1. (a) Quelle est la quantité d'information fournie par  $k$  signaux pouvant chacun prendre  $n$  valeurs équiprobables ?
- [2/20] (b) Une bande magnétique mesurant 154 m est composée de 1200 pistes parallèles. Sur chacune de ces pistes, l'information est représentée par l'orientation de domaines magnétiques disposés séquentiellement. Chaque domaine mesure  $6,8 \mu\text{m}$  de longueur et possède quatre orientations possibles. On demande de calculer la quantité d'information totale mémorisée par une telle bande magnétique.
- [3/20] 2. (a) Quelle est la représentation sur  $n$  bits (avec  $n \geq 2$ ) du nombre  $-1$  par les procédés
- i. par complément à un ?
  - ii. par complément à deux ?
  - iii. en virgule fixe par complément à deux avec 1 bit après la virgule ?
- [1/20] (b) Dans le procédé de représentation IEEE 754 en simple précision, quel est le plus petit nombre strictement positif représentable ?
- [1/20] (c) Calculer la somme  $-2 + (-1)$  à l'aide de la représentation par complément à un des entiers sur 4 bits.
- [1/20] (d) Quels sont les avantages du procédé par complément à deux pour représenter les nombres entiers signés ?
- [1/20] 3. (a) Qu'est-ce que le code machine ? En quoi diffère-t-il du code assembleur ?
- [2/20] (b) Dans un programme assembleur x86-64, on souhaite écrire la valeur  $v$  dans la  $k$ -ème case d'un tableau d'entiers (de 32 bits) pointé par le registre RAX. Le premier élément du tableau correspond à  $k = 1$ , le second à  $k = 2$ , et ainsi de suite. Les valeurs de  $v$  et de  $k$  sont respectivement disponibles dans les registres R8D et RSI. On demande d'écrire une instruction x86-64 réalisant cette opération d'écriture.
- [2/20] (c) Le plus simplement possible, décrivez l'effet des instructions x86-64 suivantes :
- i. `IMUL AX`
  - ii. `SUB R8D, -1`
  - iii. `POP qword ptr[0x100]`
  - iv. `CALL qword ptr[8 * RAX]`

4. On souhaite programmer une fonction `facteur_deux(n)` chargée de calculer le plus grand entier  $k$  tel que  $2^k$  divise  $n$ , où  $n$  est un nombre de 64 bits supposé strictement positif. Par exemple, `facteur_deux(96)` doit retourner 5, car  $96 = 2^5 \cdot 3$ . Lorsque  $n$  est impair, `facteur_deux(n)` doit retourner 0.

[2/20] (a) Écrire, en pseudocode ou en langage C (au choix), un algorithme permettant de résoudre ce problème.

*Suggestion :* En exploitant les instructions logiques, compter le nombre de bits nuls situés à la fin de la représentation binaire de  $n$ .

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

## Exemples de solutions

1. (a) Chaque signal apporte  $\log_2 n$  bits d'information. En supposant les  $k$  signaux indépendants les uns des autres, ces signaux fournissent donc au total  $k \log_2 n$  bits d'information.

(b) Chaque piste contient

$$\frac{154}{6,8 \cdot 10^{-6}} \approx 22647058$$

domaines, et chacun d'entre eux apporte  $\log_2 4 = 2$  bits d'information. La quantité d'information totale mémorisée par une bande magnétique est donc égale à

$$1200 \times 22647058 \times 2 \approx 50,620 \text{ Gbits.}$$

2. (a) i.  $\overbrace{11 \dots 1}^{n-1} 0.$   
 ii.  $\overbrace{11 \dots 1}^n.$   
 iii.  $\overbrace{11 \dots 1}^{n-1} 0.$

(b) Ce nombre possède un exposant égal à  $-127$ , et une mantisse (dénormalisée) égale à  $2^{-22}$ . Il s'agit donc du nombre

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

(c)

$$\begin{array}{rcccc} \boxed{1} & \boxed{1} & & & \\ & 1 & 1 & 0 & 1 \\ + & 1 & 1 & 1 & 0 \\ \hline & 1 & 0 & 1 & 1 \\ + & & & & 1 \\ \hline & 1 & 1 & 0 & 0 \end{array}$$

- (d) Ce procédé permet d'additionner les nombres à l'aide du même algorithme que celui développé pour la représentation non signée. Un autre avantage est que le nombre 0 possède une représentation unique.
3. (a) Le code machine est une représentation numérique des instructions d'un programme, directement décodable par un processeur. Le code assembleur est quant à lui une représentation textuelle lisible de ces instructions.
- (b) `MOV dword ptr[RAX + 4 * RSI - 4], R8D.`
- (c)
- i. Cette instruction calcule, sur 32 bits, le carré de la valeur de `AX`, représentée de façon signée. Les 32 bits de poids fort du résultat sont placés dans `DX`, et les 32 bits de poids faible dans `AX`.
  - ii. Cette instruction ajoute 1 au contenu du registre `R8D` (correspondant aux 32 bits de poids faible de `R8`).
  - iii. Cette instruction extrait de la pile une valeur de 64 bits, qu'elle écrit ensuite à l'adresse `0x100` de la mémoire.
  - iv. Cette instruction appelle une fonction dont l'adresse (sur 64 bits) est extraite d'un tableau situé à l'adresse 0 et indexé par `RAX`. (La première case du tableau correspond à `RAX = 0`, la deuxième à `RAX = 1`, etc.)
4. (a) `unsigned facteur_deux(unsigned long n)`
- ```
{
    unsigned    f;
    unsigned long i;

    for (f = 0, i = 1; (n & i) == 0; i *= 2)
        f++;

    return f;
}
```
- (b)
- ```
.text
facteur_deux: PUSH    RBP
               MOV     RBP, RSP
               MOV     EAX, 0
               MOV     RCX, 1
boucle:        MOV     RDX, RDI
               AND     RDX, RCX
               JNZ     fin
               INC     EAX
               ADD     RCX, RCX
               JMP     boucle
fin:           POP     RBP
               RET
```