Organisation des ordinateurs
Examen de première session 2026
Énoncés et solutions

Énoncés

1. Un vaisseau spatial est équipé de trois gyroscopes mesurant sa vitesse angulaire par rapport aux axes de référence. La plage de mesure de chaque gyroscope s'étend de -10 à $+10$ degrés par seconde. Dans cet intervalle, les vitesses angulaires présentent une distribution de probabilité uniforme, celles relatives à différents axes étant indépendantes les unes des autres. La résolution de chaque gyroscope est de 0,001 degré par seconde, sauf pour les vitesses angulaires comprises entre -1 et $+1$ degré par seconde pour lesquelles elle est de de 0,00025 degré par seconde.

   (a) Calculer la quantité d'information moyenne d'une mesure produite par **un** gyroscope.

   (b) La centrale inertielle du vaisseau spatial mémorise les mesures fournies pendant 24 h par les trois gyroscopes, au rythme de 10 mesures par seconde pour chacun d'entre eux. Combien de temps lui faut-il pour transmettre l'ensemble de ces mesures au centre de contrôle, si le canal de communication employé possède une capacité de 600 Mbits/s ?

2. (a) Donner tous les entiers strictement négatifs dont les représentations par valeur signée et par complément à deux sur 8 bits sont identiques.

   (b) Calculer la somme $(-\frac{7}{4}) + \frac{3}{16}$ à partir de la représentation des nombres par complément à deux en virgule fixe, avec 4 bits avant et 4 bits après la virgule.

   (c) Calculer $-18 - 29$ à partir de la représentation des nombres par complément à un sur 8 bits.

   (d) Encoder le nombre $-2^{15} - 2^{-15}$ selon le standard IEEE754 en simple précision. La représentation obtenue est-elle exacte ? Si ce n'est pas le cas, quel nombre représente-t-elle exactement ?

3. (a) À quoi sert le registre `RSP` dans l'architecture x86-64 ? Donner un exemple d'instruction qui modifie ce registre sans que le nom de celui-ci apparaisse explicitement parmi ses opérandes.

(b) Quelle est la technologie de mémoire morte actuellement prédominante, et quelles en sont les caractéristiques principales ?

(c) Expliquer étape par étape comment se déroulera l'exécution du fragment de code assembleur x86-64 suivant, en supposant que tous les emplacements de mémoire adressés correspondent à de la mémoire vive. Quelle sera la valeur finale de `EAX` ?

```
MOV RBX, 0x400
MOV EAX, 0xA1B2C3D4
MOV dword ptr[RBX], EAX
OR  byte ptr[RBX + 1], AL
AND byte ptr[RBX + 2], 0x07
ADD BL, byte ptr[RBX + 2]
XOR byte ptr[RBX + 1], AH
AND word ptr[RBX], AX
AND BL, byte ptr[RBX - 2]
MOV EAX, dword ptr[RBX]
```

4. On souhaite programmer une fonction `f` acceptant en arguments un tableau d'entiers signés `t` et le nombre d'éléments `n` de ce tableau. Cette fonction doit retourner le plus grand nombre d'éléments strictement négatifs consécutifs contenus dans `t`. Par exemple, si `t` contient $[-2, -1, 0, -2, -3, -4]$, alors la fonction doit retourner 3. Si `t` contient $[0, 1, 4]$ ou si `t` est vide, alors la fonction 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) Les valeurs mesurées présentent une distribution de probabilité uniforme dans l'intervalle $[-10, 10]$. Les probabilités qu'une mesure appartienne aux intervalles $[-10, -1]$, $[-1, -1]$ ou $[1, 10]$ sont donc respectivement égales à 0,45, 0,1 et 0,45.

   Dans l'intervalle $[-1, 1]$, il y a 8000 ou 8001 mesures équiprobables, selon la façon dont on les arrondit. Elle présentent donc chacune une probabilité d'occurrence approximativement égale à

$$0{,}1 \cdot \frac{1}{8000} = \frac{1}{80000},$$

ce qui fournit une quantité d'information de

$$\log_2 80000 \approx 16{,}288 \text{ bits.}$$

(Notons que cette approximation aurait été la même si l'on avait considéré 8001 plutôt que 8000 valeurs.)

De même, dans chacun des intervalles $[-10, -1]$ et $[1, 10]$, il y a 9000 ou 9001 valeurs. La quantité d'information d'une mesure appartenant à un de ces intervalles est donc approximativement égale à

$$0{,}45 \,.\, \frac{1}{9000} = \frac{1}{20000},$$

ce qui conduit à une quantité d'information égale à

$$\log_2 20000 \approx 14{,}288 \text{ bits.}$$

La quantité d'information moyenne d'une mesure vaut donc

$$0{,}1 \,.\, 16{,}288 + 2 \,.\, 0{,}45 \,.\, 14{,}288 \approx 14{,}488 \text{ bits.}$$

(b) Il y a trois gyroscopes et chacun d'eux fournit 10 mesures par seconde, ce qui correspond à un total de

$$3 \,.\, 10 \,.\, 24 \,.\, 3600 = 2592000$$

mesures sur 24 h, représentant

$$2592000 \,.\, 14{,}488 \approx 37552896 \text{ bits}$$

d'information. Le temps nécessaire à la transmission de ces mesures via un canal à 600 Mbits/s s'élève donc environ a

$$\frac{37552896}{600 \,.\, 2^{20}} \approx 0.060 \text{ s,}$$

c'est-à-dire 60 ms.

2. (a) Considérons une séquence $w = 1a_6a_5 \ldots a_0$ de 8 bits commençant par le bit de signe 1, puisqu'elle doit représenter un nombre négatif. Les interprétations $[w]_{vs}$ et $[w]_{c2}$ de cette séquence par valeur signée et par complément à deux sont respectivement égales à

$$[w]_{vs} = -\sum_{i=0}^{6} a_i \, 2^i$$

et

$$[w]_{c2} = -2^7 + \sum_{i=0}^{6} a_i\, 2^i.$$

Pour que ces deux nombres soit égaux, on doit avoir

$$-\sum_{i=0}^{6} a_i\, 2^i = -2^7 + \sum_{i=0}^{6} a_i\, 2^i$$
$$\sum_{i=0}^{6} a_i\, 2^i = 2^6,$$

qui correspond au nombre -64, qui est la seule solution de ce problème.

(b)

$$\begin{array}{rccccccccc} & 1 & 1 & 1 & 0 & , & 0 & 1 & 0 & 0 \\ + & 0 & 0 & 0 & 0 & , & 0 & 0 & 1 & 1 \\ \hline & 1 & 1 & 1 & 0 & , & 0 & 1 & 1 & 1 \end{array}$$

Le résultat représente bien $-\dfrac{25}{16}$, comme attendu.

(c)

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

Le résultat représente bien -47.

(d) Ce nombre est négatif, donc le bit de signe de la représentation vaut $\boxed{1}$.

On commence par chercher une mantisse normalisée, ce qui revient à déterminer un exposant e pour lequel on a

$$1 \leq \frac{2^{15} + 2^{-15}}{2^e} < 2.$$

Cette condition est satisfaite par l'exposant $e = 15$, dont la représentation est l'encodage non signé sur 8 bits de $15 + 127 = 142$, c'est-à-dire $\boxed{10001110}$.

Avec cette valeur de l'exposant, on obtient une mantisse égale à

$$\frac{2^{15} + 2^{-15}}{2^{15}} = 1 + 2^{-30},$$

qui possède la représentation $\boxed{000 \cdots 0}$, composée de 23 bits tous égaux à 0. On voit que cette représentation est approximée, et est celle d'une mantisse précisément égale à 1.

En assemblant les trois parties, on obtient donc en résumé la représentation

$$\boxed{1 \quad 10001110 \quad 000 \cdots 0},$$

qui est celle du nombre -2^{15}.

3. (a) Ce registre sert à indiquer la position en mémoire du sommet de la pile. Plus précisément, il pointe en permanence vers le dernier octet qui a été empilé, c'est-à-dire celui dont l'adresse est la plus petite parmi les octets situés sur la pile.

Un exemple d'instruction modifiant implicitement ce registre est `PUSH 0`, dont l'effet est (entre autres) de décrémenter `RSP` de 8 unités.

(b) Il s'agit de la mémoire Flash. Celle-ci permet un nombre limité d'opérations de réécriture de son contenu, et un nombre illimité d'opérations de lecture. Ces operations sont significativement plus lentes que pour la mémoire vive, surtout pour l'écriture. Par rapport à l'EEPROM, l'écriture dans une mémoire Flash ne nécessite d'effacer préalablement qu'une partie et non la totalité de son contenu.

(c) — Les deux premières instructions placent respectivement dans le registre `RBX` et dans les 32 bits de poids faible de `RAX` les constantes 0x400 et 0xA1B2C3D4.
— L'instruction suivante écrit successivement les octets 0xD4, 0xC3, 0xB2 et 0xA1, dans cet ordre, à partir de l'adresse 0x400 de la mémoire.
— L'instruction `OR` suivante calcule le *ou logique* de l'octet situé à l'adresse 0x401, qui est égal à 0xC3, et des 8 bits de poids faible de `RAX`, égaux à 0xD4. Le résultat 0xD7 est placé en mémoire à l'adresse 0x401.

— L'instruction `AND` suivante met à 0 tous les bits de l'octet situé à l'adresse 0x402, sauf les 3 bits de poids faible qui restent inchangés. Cet octet prend donc la valeur 0x02.
— L'instruction `ADD` qui suit ajoute cette valeur 0x02 à l'octet de poids faible de `RBX`. Cet octet devient donc égal à 0x02, et `RBX` prend donc la valeur 0x402.
— Ensuite, l'instruction `XOR` calcule le *ou exclusif* de `AH`, qui vaut 0xC3, et de l'octet situé à l'adresse 0x403 de la mémoire, égal à 0xA1. Le résultat 0x62 est écrit à l'adresse 0x403.
— L'instruction `AND` suivante calcule le *et logique* de `AX`, qui contient actuellement 0xC3D4, et de la valeur de 16 bits lue à partir de l'adresse 0x402 de la mémoire, qui vaut 0x6202. Le résultat 0x4200 se décompose en deux octets 0x00 et 0x42 qui sont successivement écrits aux adresses 0x402 et 0x403.
— L'avant-dernière instruction effectue un *et logique* entre le contenu du registre `BL`, qui est égal à 0x02, et l'octet situé à l'adresse 0x400, qui vaut 0xD4. Le registre `BL`, correspondant aux 8 bits de poids faible de `RBX`, devient donc égal à 0x00. Le registre `RBX` prend donc pour valeur 0x400.
— La dernière instruction lit les 4 octets situés en mémoire à partir de l'adresse 0x400 et les place dans `EAX`, qui prend donc la valeur 0x4200D7D4.

4. (a)
```
unsigned  neg_consecutifs(int t[], unsigned n)
{
  unsigned nb, max_nb, i;

  nb = 0;
  max_nb = 0;

  for (i = 0; i < n; i++)
    if (t[i] >= 0)
      nb = 0;
    else
      if (++nb > max_nb)
        max_nb = nb;

  return max_nb;
}
```

(b) On choisit d'associer les variables du programme du point (a) aux registres suivants : `t` : `RDI`, `n` : `ESI` (conformément à la convention d'appel de fonctions), `max_nb` : `EAX` (afin que la valeur de retour soit présente dans le registre approprié), `nb` : `ECX`, `i` : `EDX` (dont le contenu ne doit pas être préservé). On obtient alors le code suivant.

```
        .intel_syntax noprefix
        .text
        .global neg_consecutifs
        .type neg_consecutifs, @function
neg_consecutifs:
        PUSH RBP
        MOV  RBP, RSP
        MOV  ECX, 0
        MOV  EAX, 0
        MOV  RDX, 0
boucle: CMP  EDX, ESI
        JAE  fin
        CMP  dword ptr[RDI + 4 * RDX], 0
        JL   suite
        MOV  ECX, 0
        JMP  fin_boucle
suite:  INC  ECX
        CMP  ECX, EAX
        JBE  fin_boucle
        MOV  EAX, ECX
fin_boucle:
        INC  RDX
        JMP  boucle
fin:    POP  RBP
        RET
```