Université libre de Bruxelles

Faculté des Sciences

Départment d'informatique

Langages de programmation

(INFO-F105)

Première année du programme de bachelier en Science Informatique

Examen — août 2023

- L'examen comprend quatre questions, chacune notée sur 10 points
- La pondération associée à chaque sous-question est fournie
- Répondez à chaque question sur une feuille séparée fournie à cet effet
- Indiquez vos nom, prénom et numéro d'étudiant sur chaque feuille de réponse
- Toutes les notes de cours, transparents et syllabus sont autorisés
- Calculatrices, ordinateurs, téléphones interdits
- Durée de l'examen : deux heures trente
- Pour rappel, la note de cours est consituée de soit
  - 2/3 examen + 1/3 projet
  - 100% examen

  (la note la plus favorable des deux).

Question 1

Soit le code suivant écrit en ASM x86 muni des macros SASM vues lors des TPs :

```
1  %include "io.inc"
2  CPU 386
3  global CMAIN
4
5  section .data
6      V    DW 63, 14955, 9514, 3179, 13290,
7               10346, 16186, 16171, 13119, 11307
8      N    DB 10
9      M    DB 42
10
11 section .text
12 CMAIN:
13     xor   ebx, ebx
14     movzx ecx, BYTE[N]
15     dec   ecx
16     movzx edx, BYTE[M]
17 label_0:
18     movzx eax, WORD [V + 2*ecx]
19     and   eax, edx
20     cmp   eax, edx
21     je    label_1
22     inc   ebx
23     jmp   label_2
24 label_1:
25     loop  label_0
26 label_2:
27     xor   ebx, 1
28     PRINT_UDEC 4, ebx
29     ret
```

Pour vous simplifier la tâche, voici les nombres en binaire (codés sur 16 bits)

63	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
14955	0	0	1	1	1	0	1	0	0	1	1	0	1	0	1	1
9514	0	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0
3179	0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	1
13290	0	0	1	1	0	0	1	1	1	1	1	0	1	0	1	0
10346	0	0	1	0	1	0	0	0	0	1	1	0	1	0	1	0
16186	0	0	1	1	1	1	1	1	0	0	1	1	1	0	1	0
16171	0	0	1	1	1	1	1	1	0	0	1	0	1	0	1	1
13119	0	0	1	1	0	0	1	1	0	0	1	1	1	1	1	1
11307	0	0	1	0	1	1	0	0	0	0	1	0	1	0	1	1

On vous demande :

1. d'expliquer la ligne 25, « **`loop`** `label_0` », en incluant les labels et les registres utilisés. **[2 points]**
2. quelles sont les valeurs que peut prendre **ebx** pour d'autres valeurs de V ; **[2 points]**
3. d'expliquer ce que fait le code ASM ; **[4 points]**
4. d'indiquer ce que le programme imprime sur l'output ; **[2 points]**

Question 2

Répondez aux questions suivantes sur base du code C++ ci-dessous :

1. Qu'affiche ce programme ? Justifiez vos réponses. **[5 points]**
2. À quoi servent les lignes 3-5 ? Comment s'appellent ces déclarations de fonctions ? Sont-elles nécessaires dans ce cas-ci (justifiez) ? **[2 points]**
3. En quoi la fonction g montre-t-elle une différence entre *durée de vie* et *portée* ? **[2 points]**
4. Comment s'appelle le concept utilisé dans la fonction h (ligne 33) et en quoi consiste-t-il ? **[1 point]**

```
1  #include <iostream>
2
3  void f(int&);
4  int& g(int);
5  void h(int&, int);
6
7  int main() {
8      int y = 5, v;
9      const int& x = g(y);
10     std::cout << x << std::endl;
11     int& z = ++g(100);
12     std::cout << x << std::endl;
13     std::cout << z << std::endl;
14     z = 0;
15     h(y, z);
16     std::cout << z << std::endl;
17     switch(y) {
18     default: v = 5;
19     case 0: v = 0;
20     case 5: v = 1;
21     case 10: v = 2;
22     }
23     std::cout << v << std::endl;
24     return 0;
25 }
26
27 void f(int& x) { ++x; }
28 int& g(int x) {
29     static int y = x;
30     return y;
31 }
32 void h(int& a, int b) {
33     { int a = 5; }
34     { a = b = 0; }
35 }
```

Question 3

Étant un développeur hors pair et un avide utilisateur de XYZ, le nouveau nom du réseau social Qwiteur, vous réalisez que la plupart de vos *followers* ont un urgent besoin de gérer leurs crypto-monnaies. Vous décidez donc d'implémenter une application de gestion de portefeuilles numériques, et de la mettre à disposition de vos *followers* sur XYZ.

1. Implémentez une `enum class` **Currency** avec les valeurs constantes :
   — **ETH**, représentant une monnaie *Ethereum*
   — **BTC**, représentant une monnaie *Bitcoin*

   [1 point]

2. Implémentez une première `struct` **Transaction** avec les attributs suivants :
   — **currency**, l'attribut représentant le type de **Currency** de la *transaction*.
   — **value**, la valeur numérique de la transaction.

   [2 points]

3. Implémentez une seconde `struct` **Wallet** avec les attributs suivants :
   — **owner**, la chaîne de caractères représentant le nom du propriétaire du portefeuille.
   — **transactions**, un tableau représentant un **maximum de 100 transactions** effectuées avec le portefeuille.
   — **size**, représentant le nombre d'éléments du tableau **transactions**.

   [2 points]

4. Implémentez une fonction **balance** prenant en paramètre un **Wallet** et une **Currency**. Cette fonction retourne la somme des transactions du portefeuille (*wallet*) pour la monnaie (*currency*) donnée. **[2 points]**

5. Implémentez le surcharges suivantes :

   (a) Surchargez l'opérateur << pour l'`enum class` **Currency**, permettant de faire :
   ```
   cout << currency << endl;
   ```

   Pour `Currency::ETH`, cela affichera *ETH* et pour `Currency::BTC`, cela affichera *BTC*.

   (b) Surchargez l'opérateur << pour la structure **Wallet**. Par exemple, le code suivant :
   ```
   Wallet wallet = {
     .owner = "Hakim",
     .transactions = {
       {.currency=Currency::ETH, .value=0.5},
       {.currency=Currency::BTC, .value=2.12},
       {.currency=Currency::BTC, .value=0.4},
     },
     .size = 3
   };
   cout << wallet << endl;
   ```

Affichera : *Wallet of Hakim with 0.5 ETH and 2.52 BTC.*

Astuce : utilisez la fonction **balance** pour effectuer les calculs.

[2 points]

6. Vous avez remarqué que votre code mérite d'être amélioré, et pensez à utiliser l'encapsulation pour vos structures **Transaction** et **Wallet**. Proposez 2 modifications pour protéger l'état des objets de ces structures. **[1 point]**

Question 4

Considérez le programme C++ suivant.

```
1  #include <iostream>
2  using namespace std;
3
4  int* myfunction(int* x){
5      int* a = x ;
6      (*a)+=1;
7      int b = *a ;
8      b+=10;
9      int &c = b ;
10     c+=1000;
11     int* d = new int(c) ;
12     (*d)+=10000;
13     cout<<"*a="<<*a<<" b="<<b<<" c="<<c<<" *d="<<*d<<endl;
14
15     int e [] = {-4,-3,-2,-1,0,1,2,3,4} ;
16     d = e+1 ;
17     int f = d[(&d[4] + 2)[-3]+1]+2 ;
18     cout<<"f="<<f<<endl;
19     int* g = new int(f) ;
20     delete g ;
21     //int h =*g ;
22     return x;
23 }
24
25 int mathFunction(int x){
26     if (x <0)
27         x=-x ;
28     return x;
29 }
30
31 int otherFunction(){
32     int a = 1;
33     int p[2] = {2,3} ;
34 //  p[-1]=0;
35     cout<<a<<endl;
36     return 0;
37 }
38
39 int main() {
40     int*a= new int(1);
41     myfunction(a);
42     cout<<mathFunction(-17)<<" "<<mathFunction(23)<<endl;
43     cout<<mathFunction(-2147483648)<<endl; //2147483648=2**31
```

```
44      otherFunction();
45      return 0;
46  }
```

Répondez aux questions suivantes en supposant que le programme est exécuté sur une architecture 32-bit.

1. Qu'est-ce que le programme affichera en ligne 13 ? Justifiez votre réponse. **[2 points]**
2. Qu'est-ce que le programme affichera en ligne 18 ? Justifiez votre réponse. **[1 point]**
3. Si on décommente la ligne 21, le programme produira-t-il une erreur à la compilation et/ou à l'exécution ? Justifiez vos réponses. **[1 point]**
4. Donnez une ligne du code provoquant une "fuite de mémoire" (memory leak). Justifiez votre réponse. **[1 point]**
5. Quelle fonction mathématique est-elle implémentée par la fonction *mathFunction* ? Justifiez votre réponse. Qu'est-ce que le programme affichera en ligne 42 ? **[1 point]**
6. En ligne 43, le programme affiche la valeur -2147483648. Expliquez ce résultat. **[1 point]**
7. Quelles sont les valeurs imprimées en lignes 35 et 36 ? Justifiez vos réponses. **[1 point]**
8. Si on décommente la ligne 34, le programme produira-t-il une erreur à la compilation et/ou à l'exécution ? Justifiez vos réponses. **[1 point]**
9. Si on décommente la ligne 34, le programme produira-t-il une erreur à la compilation et/ou à l'exécution ? Justifiez vos réponses. **[1 point]**

Examen INFO-F105 Langages de Programmation 1 ; examen – août 2023

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 1

SOLUTION

1. Le code parcourt les éléments du vecteur V en partant du dernier élément jusqu'au premier. Pour chaque élément, on réalise un masque du nombre traité avec le nombre M qui vaut 42. Si le résultat du masque est 42, le code passe à l'élément suivant (donc précédent dans le vecteur). Si un résultat après le masque ne correspond pas à 42, on s'arrête d'itérer.
   D'une façon plus simple, on peut dire que le code vérifie pour chaque nombre qu'en binaire les bits 2, 4 et 6 sont bien à 1.
2. Le code imprimera 1 sur l'output puisque tous les nombres de la liste répondent au test
3. Le code ne peut afficher que 0 ou 1, car :
   — soit tous les nombres répondent au test. Dans ce cas, `ebx=0`, le `xor` inverse le bit et on imprimera 1 sur l'output ;
   — soit un nombre ne répond pas au test. Dans ce cas, on incrémente `ebx` qui vaut alors 1. Le `xor` inverse le bit et le code affichera 0 sur l'output.
4. l'instruction **`loop`** `label_0` permet de réaliser une boucle. L'instruction consulte le registre `ecx` vérifie qu'il est bien positif, le décrémente et réassigne le pointeur d'instruction vers le label mentionné, ici `label_0`.

Examen INFO-F105 Langages de Programmation 1 ; examen – août 2023

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 2

SOLUTION

1. Le programme affiche les valeurs suivantes :

```
5
6
6
6
2
```

   En effet, voici quelques commentaires indiquant le déroulement de la fonction `main` :

```
1  int main() {
2      int y = 5, v;
3      const int& x = g(y);   // g::y = 5
4      std::cout << x << std::endl;
5      int& z = ++g(100);   // ++(g::y = 5) -> g::y == 6
6      std::cout << x << std::endl;   // main::x *est* g::y
7      std::cout << z << std::endl;   // main::z *est* g::y
8      h(y, z);   // main::y == 0, main::z inchangé (copie)
9      std::cout << z << std::endl;
10     // aucun break donc les cas 0, 5 et 10 sont exécutés
11     switch(y) {
12     default: v = 5;
13     case 0: v = 0;
14     case 5: v = 1;
15     case 10: v = 2;
16     }   // main::v == 2
17     std::cout << v << std::endl;
18     return 0;
19 }
```

2. Ces déclarations sont des *prototypes*. Ils sont nécessaires ici puisque les fonctions `f`, `g` et `h` sont définies *après* la fonction `main`. Il faut alors dire au compilateur que de telles fonctions (avec leur signature complète) existent mais que leur contenu n'est pas encore disponible.
3. La variable *locale* `y` dans `g` est *statique*. Elle a donc une *durée de vie* globale mais une *portée* locale : elle n'est en effet accessible qu'à l'intérieur de la fonction `g`

(portée locale) mais ne cesse pas d'exister lorsque la fonction est terminée (durée de vie globale).

4. La fonction `h` montre un exemple de *shadowing* : une variable `b` de type **`int`** existe en tant que paramètre mais une autre variable du même nom (et du même type accessoirement) est définie dans un bloc au sein de cette fonction. Dans ce second bloc, seule la *nouvelle* variable est accessible et le paramètre est masqué.

Examen INFO-F105 Langages de Programmation 1 ; examen – août 2023

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 3

SOLUTION

```cpp
#include <iostream>
#include <string>

using namespace std;

const short int MAX_TRANSACTIONS = 100;

// -0.5 per missing or incorrect constants
enum class Currency {
  ETH,
  BTC,
};

// -1 per missing or incorrect attributs
struct Transaction {
  Currency currency;
  double value;
};

// -1 per missing or incorrect attributs
struct Wallet {
  string owner;
  Transaction transactions[MAX_TRANSACTIONS];
  int size;
};

// -1 if logic is incorrect
// -1 if params and/or return are incorrect
double balance(const Wallet& wallet, const Currency& currency) {
  double amount = 0.;
  for (int i = 0; i < wallet.size; i++) {
    Transaction tx = wallet.transactions[i];
    if(tx.currency == currency) {
      amount += tx.value;
    }
  }
  return amount;
```

```
38 }
39
40 // -0.5 if params and/or return are incorrect
41 // -1 if logic is incorrect
42 ostream& operator<<(ostream& os, const Currency& currency){
43   switch (currency) {
44     case Currency::ETH:
45       os << "ETH";
46       break;
47     case Currency::BTC:
48       os << "BTC";
49       break;
50     }
51   return os;
52 }
53
54 // -1 if params and/or return are incorrect
55 // -1 if logic is incorrect
56 ostream& operator<<(ostream& os, const Wallet& wallet){
57   os << "Wallet of " << wallet.owner << " with " << "";
58   os << balance(wallet, Currency::ETH) << " " << Currency::ETH;
59   os << " and ";
60   os << balance(wallet, Currency::BTC) << " " << Currency::BTC;
61   return os;
62 }
63
64 int main() {
65   Wallet wallet = {
66     .owner = "Hakim",
67     .transactions = {
68       {.currency=Currency::ETH, .value=0.5},
69       {.currency=Currency::BTC, .value=2.12},
70       {.currency=Currency::BTC, .value=0.4},
71     },
72     .size = 3,
73   };
74   cout << wallet << endl;
75 }
```

Examen INFO-F105 Langages de Programmation 1 ; examen – août 2023

NOM, PRENOM (en majuscules) :
Numéro d'étudiant :

Numéro de question : 4

SOLUTION

1. *a=2 b=1012 c=1012 *d=11012
   Le pointeur a créé en ligne 40 est passé à la fonction *myfunction*, puis recopié dans le pointeur local a en ligne 5. La ligne 6 modifie la valeur pointée en l'incrémentant de 1.
   La ligne 7 récupère la valeur et la copie dans une nouvelle variable entière b, qui est ensuite incrémentée de 10 en ligne 8. A ce moment b vaut donc 12.
   La ligne 9 définit **c** comme un alias de **b**, et la ligne 10 incrémente cette variable de 1000. A ce moment on a donc $b = c = 1012$.
   La ligne 11 crée un nouveau pointeur vers une variable entière sur la tas, initialisée à la valeur de c càd 1012. La ligne 12 augmente cette valeur de 10000.
2. f=0
   d vaut l'adresse de e (càd l'adresse de son premier élément) plus un (càd que d vaut l'adresse du deuxième élément de e)
   On a &d[4] est l'adresse de e + 5, et &d[4] + 2 est l'adresse de e+7
   On a (&d[4] + 2)[-3] est la valeur en e[4] càd 0, et (&d[4] + 2)[-3]+1 vaut 1
   On a d[1]=e[2]=-2 et d[(&d[4] + 2)[-3]+1]+2=0
3. Pas d'erreur à la compilation : delete désalloue la mémoire mais le langage ne spécifie pas son action sur le pointeur. Possible erreur à l'exécution selon l'implémentation du delete
4. Ligne 16 : la mémoire allouée sur le tas en ligne 11 n'est plus accessible
5. La fonction implémente la valeur absolue (lorsque les entrées sont restreintes pour éviter les overflows)
   Affichage 17 23
6. On note que $2147483648 = 2^{31}$. Si les entiers signés sont codés sur 32 bits, alors ils prennent des valeurs entre -2^{31} et $2^{31} - 1$.
   La valeur de l'argument étant négative, le test est passé en ligne 26 et la ligne 27 est exécutée.
   La valeur 2^{31} ne pouvant pas être représentée, la ligne 27 provoque un *integer overflow* (dépassement d'entier), et cette valeur est codée comme la valeur égale modulo 2^{32} dans l'intervale admis, càd -2^{31}
7. 1
   la variable a est initialisée à 1 en ligne 32, et pas modifiée ensuite
   (la ligne 36 n'affiche rien)
8. Pas d'erreur à la compilation. Possible erreur à l'exécution. (Il est possible aussi qu'aucune erreur ne se produise, et que la zone de mémoire juste avant p sur la

pile soit modifiée. Souvent, cette zone de mémoire correspondra à la variable a, et la ligne 35 affichera alors 0 au lieu de 1.)

9. (Identique à la précédente ; ignorer.)

Le total sur 9 a été mis à l'échelle et arrondi au 1/4 supérieur pour obtenir une note sur 10.