



# ESIR SYS1 CC2 2023–2024

## 21 décembre 2023

Nom et Prénom

**X X****numéro d'étudiant : 99999991**

Durée : 1h, Notation : sur 20 points

## 1 Première partie : QCM (10 points)

### Instructions :

Cochez clairement la case de la réponse que vous pensez être juste. Il y a **une seule réponse** juste par question.

### Barème :

+0.5 pour chaque réponse correcte

−0.5/ $m$  pour chaque réponse fausse (où  $m+1$  est le nombre de réponses possibles)

### Question 1 Soit le code suivant

```
#include <stdio.h>
```

```
void foo(int* i) {  
    *i = 10;  
}
```

```
int main( int argc, char** argv) {  
    int i = 5;  
    foo(&i);  
    printf("%i\n",i);  
}
```

Quelle valeur ce code imprime-t-il sur la console ?

☐ 10☐ le contenu de la mémoire à l'adresse 10☐ 5

**Question 2** Dans un processeur utilisant la technologie de *pipelining*, l'unité de chargement d'instruction (Instruction Fetch Unit) et l'unité de décodage d'instruction (Instruction Decode Unit) d'un même coeur peuvent-elles être actives simultanément ?

☐ oui☐ non

**Question 3** Un code assembleur est directement exécutable par un processeur.

☐ vrai☐ faux

**Question 4** L'appel suivant est-il possible en assembleur nasm x86-64 ?

```
call foo(RAX)
```

☐ oui☐ non



**Question 5** La mémoire physique d'un ordinateur personnel est typiquement intégrée au processeur principal (CPU)

- ☐ vrai  
☐ faux

**Question 6** Où sont allouées les variables automatiques ?

- ☐ sur la pile  
☐ sur le tas  
☐ sur le segment BSS

**Question 7** Si deux programmes A et B sont compilés avec la même librairie dynamique L, le code de la librairie L se retrouve-t-il dupliqué dans chacun des fichiers exécutables de A et de B ?

- ☐ oui  
☐ non

**Question 8** En C, combien d'arguments prend la fonction `printf` ?

- ☐ aucun  
☐ toujours 1  
☐ toujours 2  
☐ 1 ou plus selon les cas  
☐ 2 ou plus selon les cas

**Question 9** Comment désalloue-t-on explicitement de la mémoire sur le tas en Java ?

- ☐ On ne peut pas désallouer explicitement la mémoire du tas en Java.  
☐ avec l'opérateur `delete`  
☐ avec la méthode `free()`

**Question 10** Dans un processeur moderne :

- ☐ le cache L2 est plus rapide que le cache L1.  
☐ le cache L1 est plus rapide que le cache L2.

**Question 11** En assembleur, utilise-t-on toujours une opération `cmp` avant une opération de saut conditionnel (p.ex. `je`, `jg`, `jle`, etc.) ?

- ☐ non  
☐ oui



**Question 12** Le code suivant compile-t-il en C ?

```
#include <stdio.h>

int main( int argc, char** argv) {
    int i = "Hi" - "Bye";
    printf("%i\n",i);
}
```

☐ non

☐ oui

**Question 13** En assembleur x86-64, l'instruction 'call foo' modifie-t-elle la valeur du registre **rsp** ?

☐ non

☐ oui

**Question 14** En assembleur, quel est le registre par rapport auquel se font les accès aux variables locales à une fonction ?

☐ rbp

☐ rbx

☐ rsp

☐ rsx

**Question 15** On considère l'extrait de code en C suivant :

```
char s[] = "bonjour";
s[3]=0;
printf("%s\n", s );
```

La ligne `printf("%s\n", s );` permet d'imprimer une chaîne de caractères. Quelle sortie produit cet extrait de programme ?

☐ bon

☐ bonJour

☐ bon our

☐ bon0our



**Question 16** Soit le code assembleur suivant

```
mov  rbx,1
mov  rcx,2
push rbx
push rcx
pop  rbx
pop  rax
```

À la suite de ce code :

- ☐ rax==1, rbx==1
- ☐ rax==2, rbx==1
- ☐ rax==1, rbx==2
- ☐ rax==2, rbx==2

**Question 17** Comment peut-on compiler le fichier `test.c` avec `gcc` sans réaliser l'édition de lien (*linking*) ?

- ☐ `gcc -c test.c`
- ☐ `gcc test.c`
- ☐ `gcc -o test.c`

**Question 18** On considère le nombre hexadécimal `0xAABBCCDDEE`. Combien d'octets sont au minimum nécessaires pour le stocker ?

- ☐ 3
- ☐ 5
- ☐ 8
- ☐ 10

**Question 19** Comment réserver 16 octets (10h en hexadécimal) sur la pile en assembleur x86 64 bits ?

- ☐ `sub rbp,10h`
- ☐ `add rsp,10h`
- ☐ `sub rsp,10h`
- ☐ `add rbp,10h`



**Question 20** Soit le code assembleur nasm suivant :

```
SECTION    .data
message:   db      "foo is executing", 10      ; note the newline at the end
msgLen:    equ $-message                      ; compute the length of message string

SECTION    .text

foo:
    mov     rax, 1                          ; system call for write
    mov     rdi, 1                          ; file handle 1 is stdout
    mov     rsi, message                    ; address of string to output
    mov     rdx, msgLen                    ; number of bytes
    syscall                                ; invoke operating system to do the write
    ret

_start:
    call foo

    mov     rax, 60                        ; system call for exit
    mov     rdi, 0                        ; exit code 0, xor rdi, rdi is used (faster, shorter)
    syscall                                ; invoke operating system to exit

GLOBAL     _start
```

Ce code est censé imprimer la chaîne de caractère 'foo is executing', puis terminer. Ce code va-t-il compiler et s'exécuter correctement ?

☐ non

☐ oui



## 2 Deuxième partie : Questions ouvertes et problèmes (10 points)

**Question 21** L'on souhaite imprimer 5 fois de suite la chaîne de caractère 'In the loop!' en utilisant une boucle en assembleur. Un premier essai livre le code assembleur suivant : (Seules les parties du code pertinentes sont montrées.)

```
SECTION      .data
message:     db "In the loop!", 10      ; note the newline at the end
msgLen:      equ $-message

SECTION      .text
[.]
mov          rax, 0

begin:
mov          rax, 1                    ; system call for write
mov          rdi, 1                    ; file handle 1 is stdout
mov          rsi, message              ; address of string to output
mov          rdx, msgLen               ; number of bytes
syscall

inc          rax
cmp          rax, 5
jl           begin
[.]
```

Ce code compile, mais l'exécution n'imprime qu'une seule fois le message :

```
$ ./loop_buggy
In the loop!
$
```

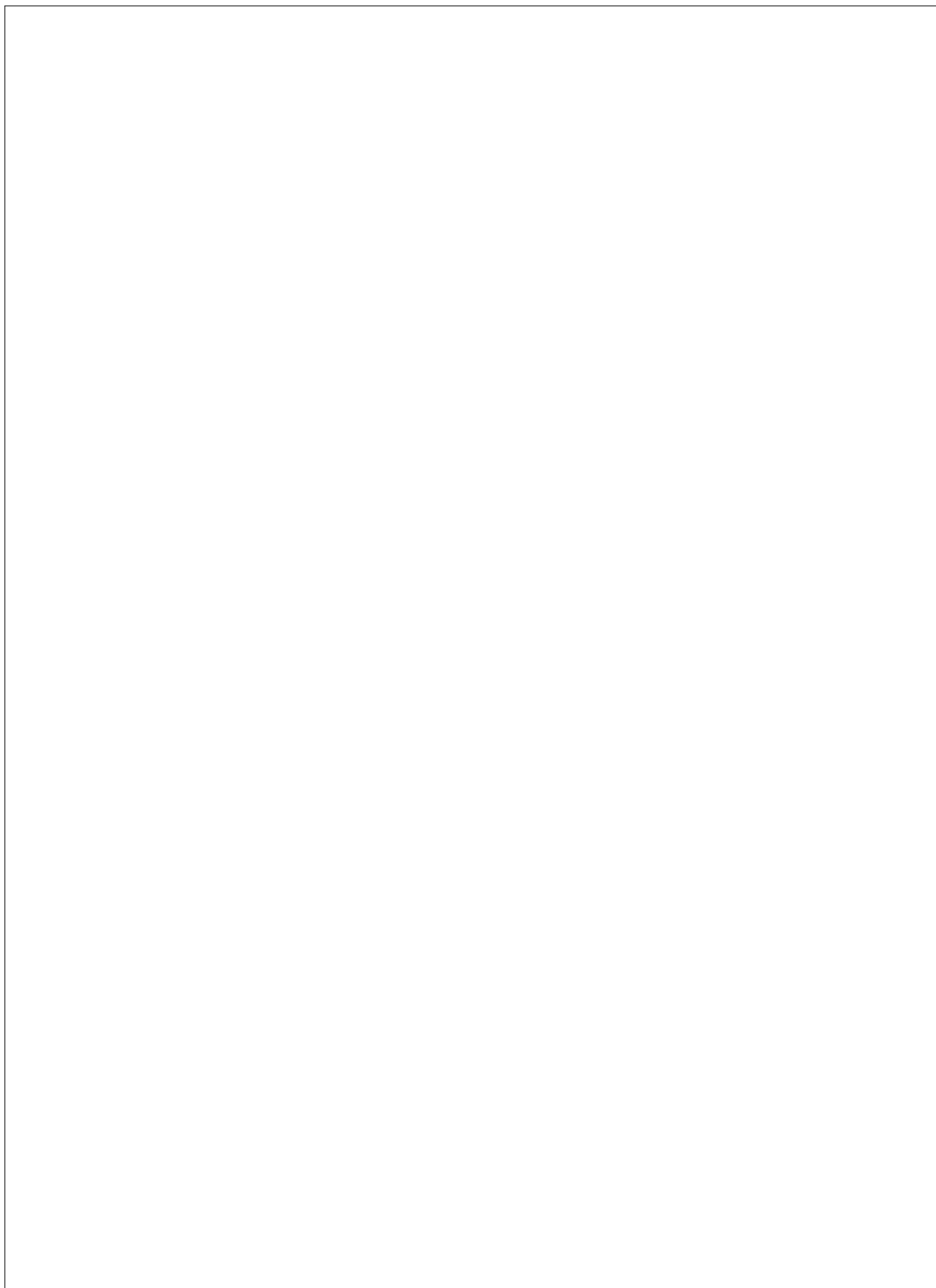
1. Pourquoi observe-t-on ce comportement ?
2. Comment corrigeriez-vous le code pour que la boucle s'exécute bien 5 fois. Vous proposerez *deux* solutions : l'une utilisant une variable supplémentaire dans la section `.data`, et l'autre s'appuyant sur la pile. Dans les deux cas vous prendrez soin de bien expliquer votre solution.

[5 points]

☐ A+ ☐ A ☐ B ☐ C ☐ E ☐ F



+55/7/54+





**Question 22** On considère le programme C suivant (certains `#include` et tests d'erreur ont été retirés pour plus de lisibilité).

```
int main( int argc, char** argv) {
    FILE * f = fopen("mon_fichier.txt","r+");
    //... tests d'erreur retirés
    int my_char ;
    while ( (my_char = fgetc(f)) != EOF ) {
        if (my_char == ' ' && ftell(f)>=2) {
            fseek(f,-2,SEEK_CUR);
            char c = fgetc(f);
            if (c>='a' && c<='z') {
                c = c - 'a' + 'A';
            }
            fseek(f,-1,SEEK_CUR);
            fputc(c,f);
            fseek(f,+1,SEEK_CUR);
        } // EndIf
    } // EndWhile
    fclose(f);
    return 0;
} // EndMain
```

L'on compile ce programme avec gcc.

1. L'on suppose que le fichier `mon_fichier.txt` contient le texte suivant.

La Terre est bleue comme une orange

Quel sera le contenu de `mon_fichier.txt` après l'exécution du programme ci-dessus? Vous expliquerez votre raisonnement. **[2 points]**

2. L'on réinitialise le fichier `mon_fichier.txt` à son contenu d'origine. L'on souhaite maintenant modifier le programme pour que `mon_fichier.txt` contienne le texte suivant après l'exécution du programme modifié

La Terre Est Bleue Comme Une Orange

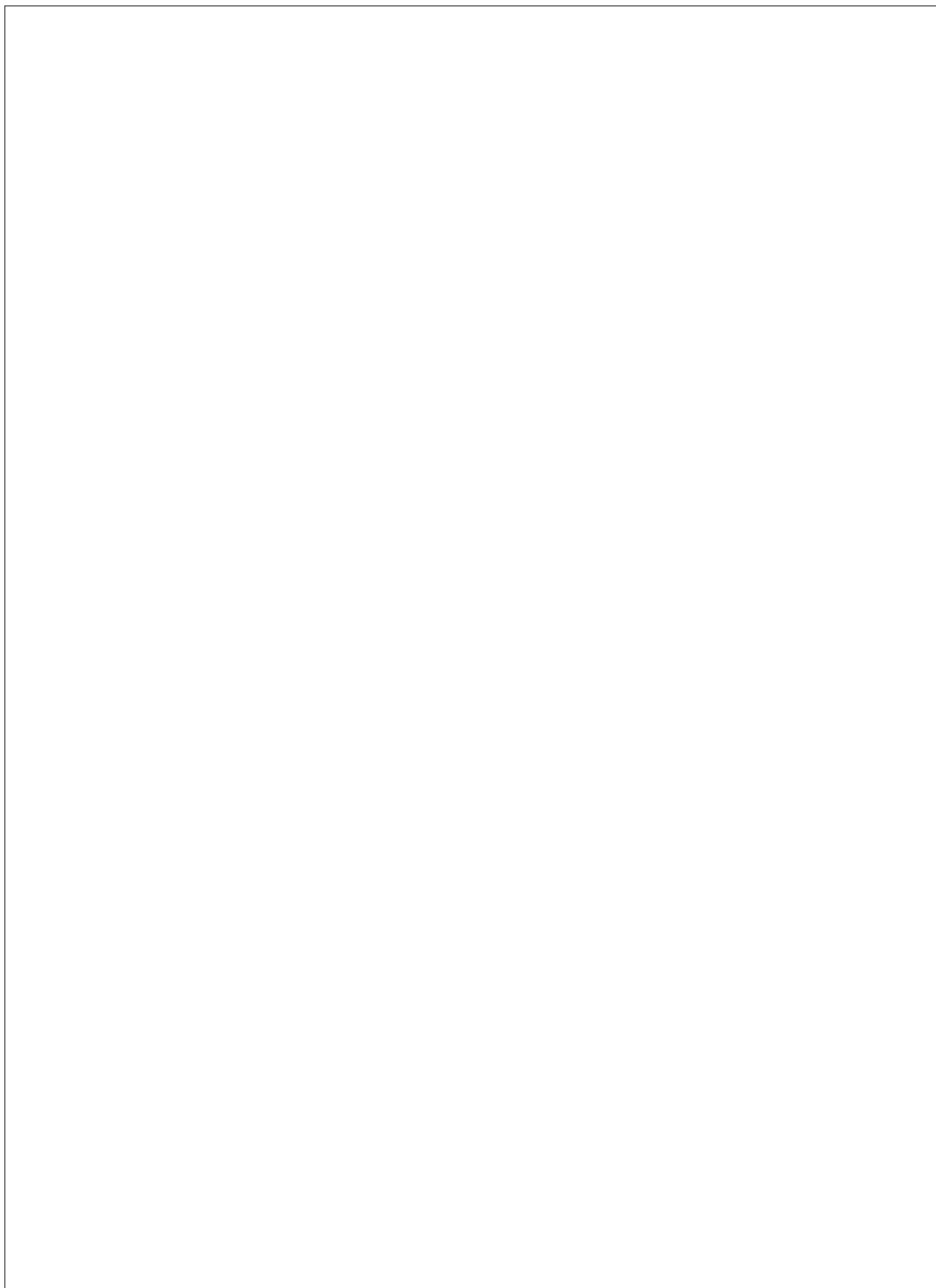
(C'est à dire que chaque lettre après un espace soit mise en majuscule.)

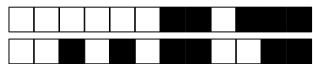
Quelles modifications apporteriez-vous au programme initial? Vous expliquerez votre raisonnement. (Le programme modifié doit pouvoir fonctionner sur n'importe quel texte ASCII, pas seulement sur la phrase '*La Terre Est Bleue Comme Une Orange*'.) **[3 points]**

☐ A+ ☐ A ☐ B ☐ C ☐ D ☐ F



+55/9/52+





+55/10/51+

