FACULTÉ DES SCIENCES

Formulaire pour reprographie d'examen

Le questionnaire d'examen doit parvenir au Centre de reprographie 5 jours ouvrables avant la date de l'examen.

INFORMATIONS POUR LE SECRÉTARIAT DE LA FACULTÉ
(Attacher à chaque questionnaire)

☒ **INTRA**
☐ **FINAL**

Sigle : IFT209 (H-15)

Titre : Programmation système

Professeur : ST-DENIS, Richard

Date et heure de l'examen : Mercredi 25 février, 8h30 à 10h20, D3-2038,2039

Nombre de pages : 9

Nombre d'étudiants : 43

☒ **Je désire prendre possession des questionnaires d'examen la veille de l'activité et je me rends responsable de leur mise en sécurité.**

Signature : _____

L'étudiant devra répondre

☐ dans un cahier d'examen
☒ sur le questionnaire

Veuillez inclure

_____ feuilles blanches additionnelles
_____ feuilles de papier graphique

Je consens à ce que deux (2) copies du questionnaire de cet examen soient remises à l'AGES (Association générale des étudiants en sciences) après la fin de la période des examens.

☒ **OUI** ☐ **NON**

Signature : _____

UNIVERSITÉ DE SHERBROOKE

DÉPARTEMENT D'INFORMATIQUE

IFT 209

Programmation système

EXAMEN PÉRIODIQUE[1]

Le mercredi 25 février 2015 de 8 h 30 à 10 h 20 Professeur : Richard St-Denis

— Toute documentation est permise.

— **Tout appareil électronique, incluant le téléphone cellulaire et l'ordinateur, est interdit.**

— Ne dégrafez pas ce questionnaire.

— Répondez dans les espaces prévus à cet effet.

— **Personne ne peut quitter la salle d'examen avant 8 h 45.**

— **Personne ne peut quitter son siège entre 10 h 10 et 10 h 20** (-1 point).

— La correction est, entre autres, basée sur le fait que chacune de vos réponses soit :

  — claire, c'est-à-dire lisible et compréhensible pour le correcteur ;

  — précise, c'est-à-dire exacte ou sans erreur ;

  — complète, c'est-à-dire que toutes les étapes de résolution du problème sont présentes ;

  — concise, c'est-à-dire que la méthode de résolution est la plus courte possible.

Nom : _____ Prénom : _____

Signature : _____ Matricule : _____

Question	Barème
1	/6
2	/6
3	/6
4	/6
5	/6
total :	/30

1. Ce questionnaire comporte neuf (9) pages.

1. (6 points) Répondez aux questions suivantes.

   a) (2 points) Convertissez le nombre binaire $111101,1001_2$ en un nombre octal et en un nombre décimal. Donnez les principales étapes de résolution de ce problème.

   b) (2 points) Convertissez le nombre décimal $172,53_{10}$ en base 4 et en base 16. Donnez les principales étapes de résolution de ce problème.

c) (2 points) Quel est le nombre minimum de bits requis pour représenter le nombre 353 ? Pourquoi ?

2. (6 points) Répondez aux questions suivantes.

a) (1 point) L'adresse $1A0000666_{16}$ est-elle une adresse (avec justification)

d'un octet

d'un demi-mot

d'un mot

d'un mot étendu

b) (1 point) Dans quelles situations un programmeur doit-il tenir compte du format *little-endian* ou *big-endian* ?

c) (1 point) Quelle est la principale raison de la présence d'instructions synthétiques comme `setx`, `mov` et `ret` dans le langage d'assemblage *SPARC* ?

d) (1 point) Combien y a-t-il de registres dans six fenêtres de registres d'une machine *SPARC* ? Pourquoi ?

e) (1 point) Combien de registres généraux (les registres `r`) un programmeur dispose-t-il pour écrire un sous-programme ? Pourquoi ?

f) (1 point) Quel est le mode d'adressage utilisé dans les instructions de branchement ?

3. (6 points) Effectuez chacune des opérations suivantes dans la représentation complément à 2 et, pour chacune des opérations, indiquez s'il y a débordement puis indiquez s'il y a report (les nombres sont sur quatre bits).

	1010	1111	0111
	−0011	+1110	+1000
résultat			
débordement			
report			

4. (6 points) Voici un petit programme en langage de programmation C qui calcule la division (« / ») et le modulo (« % ») de deux nombres entiers signés :

```
#include <stdio.h>
int main()
{
 int x, y;
 scanf("%d %d", &x, &y);
 printf("Div: %d   Mod: %d\n", x / y, x % y);
 return 0;
}
```

Voici deux exemples d'exécution de ce programme sur *Tarin* :

```
dinf-tarin de localhost: a.out
79 11
Div: 7   Mod: 2
dinf-tarin de localhost: a.out
-79 11
Div: -7   Mod: -2
```

Voici les mêmes calculs, mais effectués à l'aide du langage de programmation *Python*.

```
dinf-tarin de localhost: python
Python 2.6.8 (unknown, Aug  8 2013, 10:21:13) [C] on sunos5
Type "help", "copyright", "credits" or "license" for more information.
>>> 79 / 11
7
>>> 79 % 11
2
>>> -79 / 11
-8
>>> -79 % 11
9
```

Vous pouvez remarquer que les résultats ne sont pas les mêmes dans le cas de nombres signés. En effet, le langage *Python* utilise la définition mathématique de la division entière signée suivante : $n \div m = \lfloor n/m \rfloor$, où le symbole « $\div$ » désigne la division entière, le symbole « / » désigne la division réelle et les symboles « $\lfloor \cdot \rfloor$ » désignent l'opération plancher ($\lfloor x \rfloor$ est le plus grand entier inférieur ou égal à x). Mais, le langage C utilise la troncature, tout comme l'instruction `sdivx`. Dans cet exemple, troncature de $-79/11 = -7,\overline{18}$ est -7, alors que $\lfloor -79/11 \rfloor = \lfloor -7,\overline{18} \rfloor$ est -8. Enfin, dans tous les cas, l'égalité suivante doit toujours être satisfaite : $n = (n \div m)m + (n \text{ modulo } m)$. Notez qu'il n'existe PAS d'instruction machine sous le *SPARC* pour le calcul du modulo. Il faut donc réaliser cette opération par programmation.

Complétez le sous-programme suivant qui calcule le modulo de deux entiers signés contenus dans les registres %i1 et %i2 selon la méthode retenue par *Python* (le registre %i0 contient alors 0) ou la méthode retenue par *C* (le registre %i0 contient alors 1).

```
/* Mod     : Sous-programme de calcul du modulo de deux entiers signes.
   Entree : %i0, 0 (pour la methode de Python) ou 1 (pour la methode de C);
            %i1, un entier signe de 64 bits;
            %i2, un entier signe de 64 bits.
   Sortie : %o0, le modulo.                                          */
        .global Mod
        .section ".text"
Mod:    save    %sp,-208,%sp
        sdivx   %i1,%i2,%l0   ! division entiere signee a la C
```

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

```
        ret
        restore
```

5. (6 points) Considérons le jeu *sudoku* classique avec une grille 9×9 et des sous-grilles de 3×3, appelées régions, dont les symboles sont des chiffres allant de 1 à 9. La règle du jeu est la suivante. Chaque ligne, colonne et région ne doit contenir qu'une seule fois tous les chiffres de 1 à 9. Formulé autrement, chacun de ces ensembles doit contenir tous les chiffres de 1 à 9. Voici une telle grille qui satisfait ces contraintes.

4	1	5	6	3	8	9	7	2
3	6	2	4	7	9	1	8	5
7	8	9	2	1	5	3	6	4
9	2	6	3	4	1	7	5	8
1	3	8	7	5	6	4	2	9
5	7	4	9	8	2	6	3	1
2	5	7	1	6	4	8	9	3
8	4	3	5	9	7	2	1	6
6	9	1	8	2	3	5	4	7

Supposons que la grille 9×9 est un tableau de dimension 2 de neuf rangées et neuf colonnes emmagasiné par rangée. Chaque composante du tableau est un entier stocké dans un demi-mot.

a) (2 points) Quelle est l'adresse de la première composante de la région au centre de la grille, reproduite ici

3	4	1
7	5	6
9	8	2

si le tableau est stocké en mémoire principale à l'adresse α ?

b) (2 points) Écrivez une suite d'instructions en langage d'assemble *SPARC* qui parcourt toutes les composantes de la région au centre de la grille sachant que l'adresse de sa première composante est dans le registre %10.

Une façon de vérifier si une région satisfait la contrainte du jeu (une région ne doit contenir qu'une seule fois tous les chiffres de 1 à 9), consiste à introduire un tableau de dimension 1 de neuf composantes à raison d'un octet par composante initialisé à 1. Lors du parcours d'une région, il suffit i) de mettre à 0 la i^e composante si i est le chiffre courant examiné de la région et ii) de vérifier à la fin du parcours si la somme des neuf composantes est nulle.

c) (2 points) Écrivez une suite d'instructions en langage d'assemble *SPARC* qui effectue le traitement décrit en i) dans le paragraphe précédent. Vous devez clairement indiquer ce que contiennent les registres que vous utilisez. Indiquez clairement à quel endroit dans la suite d'instructions de votre réponse en b) où vous placeriez la suite d'instructions suivante.

```
        .section ".data"
tmp:    .byte   1, 1, 1, 1, 1, 1, 1, 1, 1
        .section ".text"
```

FIN DE L'EXAMEN