

IFT209 –Programmation système

Laboratoire 2

Enseignant: Rosa Garcia

Date de remise: indiquée sur **Turnin**

Modalités de remise: **Turnin**

À réaliser: personne seule ou en équipe de deux

Le but de ce laboratoire est de vous familiariser avec l'outil de déverminage **gdb**.

Problème. Trois programmes dont on vous donne le code source font des erreurs à l'exécution. Vous devez corriger ces programmes jusqu'à ce qu'ils produisent les bons résultats. L'utilisation du débuggeur est essentielle.

Tâche. Corrigez les programmes se trouvant dans les fichiers `prog1.as`, `prog2.as` et `prog3.as`. Le code de ceux-ci ainsi que leur script de compilation vous sont fournis sur **Turnin**.

Programme 1. Le fichier `prog1.as` contient un programme qui calcule la partie entière de la racine carrée d'un nombre entré au clavier.

Pour le compiler, entrez la commande `make prog1` dans le terminal.

Tests. Voici les résultats attendus pour les valeurs de test 40, 25 et -27 :

Entrez une valeur:40

La racine carree est: 6

Entrez une valeur:25

La racine carree est: 5

Entrez une valeur:-27

Valeur inadmissible.

Conseil. L'utilisation des commandes `info reg`, `break`, `stepi` et `continue` est recommandée pour déverminier ce programme.

Programme 2. Le fichier `prog2.as` contient un programme qui cherche un nombre entier entré au clavier dans une liste de nombres prédéterminée. Il retourne la position du nombre dans la liste s'il le trouve, ou affiche un message comme quoi le nombre n'est pas présent dans la liste.

Pour le compiler, entrez la commande `make prog2` dans le terminal.

Tests. Voici les résultats attendus pour les valeurs de test 768, 13 et 0 :

Entrez la valeur qu'on cherche:768
La valeur n'est pas dans la liste.

Entrez la valeur qu'on cherche:13
La valeur est à la position 5

Entrez la valeur qu'on cherche:0
La valeur n'est pas dans la liste.

Conseil. L'utilisation de la commande `x` devient pertinente pour ce programme. Elle sert à afficher le contenu de la mémoire. Par exemple, la commande `x/2dg &liste` afficherait les deux premiers éléments de la liste (ceux-ci occupent des double-mots).

Vous pouvez également procéder à l'affichage directement à partir d'une adresse. Si l'adresse correspondant au symbole `liste` est `0x411018`, alors `x/d 0x411018` affichera le premier élément de la liste, `x/d 0x411020` affichera le second (la distance entre `0x411018` et `0x411020` est exactement 8).

Programme 3. Le fichier `prog3.as` lit deux nombres entrés au clavier et vérifie si le premier est divisible par le second, en utilisant un sous-programme nommé `Divisible`. Selon le résultat du sous-programme, il affiche le message approprié.

Pour le compiler, entrez la commande `make prog3` dans le terminal.

Tests. Voici les résultats attendus pour les paires de valeurs de test 4 et 3 ainsi que 16 et 4 :

Entrez deux valeurs:4
3
Le nombres ne sont pas divisibles.

Entrez deux valeurs:16
4
Le nombres sont divisibles.

Conseil. L'utilisation de la commande `disass` devient pertinente pour ce programme. Cette commande permet de voir où se trouve le compteur ordinal dans la liste des instructions présentement.

La commande `break` peut aussi être utilisée pour mettre un point d'arrêt à la ligne de votre choix, comme ceci: `break prog3.as:44` mettrait un point d'arrêt à la ligne 44 de votre fichier de code source.

Rappels.

- Un sous-programme est appelé avec l'instruction `b1`;
- On commence un sous-programme avec `SAVE`;
- On termine un sous-programme avec `RESTORE`, puis `ret`;
- On termine un programme principal avec un appel au sous-programme `exit`;
- Les valeurs numériques basses (autour de 0) ne sont pas des adresses valides;
- Lorsque vous exécutez pas à pas (`stepi`) et qu'une instruction a un comportement étrange, il est utile d'aller lire sa description complète dans la feuille d'instructions et de regarder les valeurs des registres au moment de son exécution.

Directives.

- Chaque programme doit être une version corrigée du code fourni sur **Turnin**;
- Votre remise doit comporter le code source des trois programmes: `prog1.as`, `prog2.as` et `prog3.as`.

Pointage. Vous pouvez obtenir jusqu'à 10 points répartis ainsi:

- 3 points si `prog1` réussit ses trois tests;
- 3 points si `prog2` réussit ses trois tests;
- 4 points si `prog3` réussit ses deux tests.