Exercice à rendre 1

Convertissez en assembleur Zorglub33 les fonctions C ci-après.

```
// x² − 2xy − y²
int calcul (int x, int y)
{
    return x*x - 2*x*y - y*y ;
}

// produit scalaire de deux vecteurs
int prodscal (const int v1 [], const int v2 [], int n)
{
    int p = 0 ;
    for (int i = 0 ; i < n ; i++)
        p += v1 [i] * v2 [i] ;
    return p ;
}

// valeur approchée de ⌊√n⌋ par dichotomie
int racine (int n)
{
    int inf = 1, sup = n, r ;
    r = inf + (sup - inf)/2 ;
    while (r*r > n || (r+1)*(r+1) <= n)
    {
        if (r*r > n)
            sup = r  ;
        else
            inf = r ;
        r = inf + (sup - inf)/2 ;
    }
    return r ;
}
```

Vous mettrez ces fonctions traduites en assembleur dans un fichier unique `exo1.s`, auquel vous adjoindrez des programmes `main_calcul`, `main_prodscal` et `main_racine` pour vous permettre de tester les appels à vos fonctions.

Vous apporterez un soin particulier à la mise en forme et aux commentaires de façon à rendre un code lisible, et vous utiliserez l'émulateur Zorglub33 (disponible sur Moodle) pour la mise au point de votre programme.

Un script de test est mis à votre disposition sur Moodle. Celui-ci exécute votre programme sur des jeux de tests qui serviront de base à l'évaluation de votre rendu. La commande suivante permet de lancer les tests : `sh ./test1.sh`

N'hésitez pas à contacter votre enseignant si vous constatez un comportement anormal ou si vous souhaitez ajouter un test.

Vous devrez rendre sur Moodle un *unique* fichier nommé `exo1.s` (Moodle sait qui vous êtes, il est inutile d'appeler votre programme `Jean-Claude_Dusse_exo1.s`, et il est interdit de rendre un fichier d'un autre nom ou une archive au format du jour). De plus, assurez-vous de rendre des fichiers utilisant l'encodage UTF-8, il y aura de sévères pénalités sinon.

Ce TP à rendre est **individuel**. On rappelle que la copie ou le plagiat sont sévèrement sanctionnés.