

**EXERCICE A Assembleur (5 points)****Numéro :**

On cherche à écrire en assembleur un code calculant la partie entière de la racine carrée d'un nombre positif. Ce code est composé de deux fonctions. La première **sqr()** se contente d'invoquer la fonction récursive **sqrr()**.

```
int sqrr (int n, int i)
{
    if (n < i) return 0;
    return 1 + sqrr ((n - i), (i + 2));
}

int sqr (int n)
{
    return sqrr (n, 1);
}
```

Le code de la fonction **main()** utilisée par xspim pour l'invocation de la fonction **sqr** est le suivant :

```
msg:      .data
          .asciiz "\nEntrez un nombre: "

          .text
          .globl main

main:     addiu   $29, $29, -8
          sw      $31, 4 ($29)

main1:    la      $4, msg
          li      $2, 4
          syscall
          li      $2, 5
          syscall

          beq     $2, $0, main2

          ..... appel de la fonction sqr()

          add     $4, $0, $2
          li      $2, 1
          syscall
          j       main1

main2:    lw      $31, 4 ($29)
          addiu   $29, $29, 8
          jr      $31
```

**A1 (1 point)** Écrivez ici les instructions assembleur permettant l'appel de la fonction **sqr()**.

**A2 (1 point)** quoi sert la ligne **.globl main**

**A3 (1point)** Pour quelle raison ne peux-t-on pas écrire le code suivant alors que c'est fonctionnellement correct ?

```
addiu   $29, $29, 8
lw      $31, -4 ($29)
jr      $31
```

**A4 (1 point)** Écrire le code de la fonction **sqr**. Vous devez justifier la valeur du déplacement du pointeur de pile en indiquant les valeurs de (na, nv, nr) et l'usage de la pile.

**A5 (1 point)** Ecrire le code de la fonction **sqrr**

**Exercice B : Mémoires Cache****Numéro :**

On considère un cache de données de premier niveau, write-through, à correspondance directe, contenant 512 lignes, et possédant une capacité totale de 32Ko. Les adresses sont sur 32 bits. Le but de l'exercice est d'analyser le remplissage d'un cache de données de premier niveau puis d'estimer le nombre de cycles nécessaires à l'exécution d'un programme. On suppose que ce cache est initialement vide. On considère la fonction my\_sum suivante (on remarquera que 768 = 512+256)

```
int32_t X[768][16];
...
int32_t my_sum(int32_t X[768][16]) {
    register int32_t i, j;
    register int32_t sum = 0;
    for (i = 0; i < 16; i++) {
        for (j = 0; j < 768; j++) { sum += X[j][i]; }
    }
    return sum;
}
```

Dans tout l'exercice, on suppose que le tableau X est placé à l'adresse 0x10010000, et qu'il est passé en paramètre de la fonction my\_sum.

**Rappel (i)** le mot clé "register" est une directive passée au compilateur pour qu'il place la variable i dans un registre plutôt que sur la pile ; la variable i reste donc en registre durant toute la durée d'exécution de la fonction et ni sa lecture ni son écriture ne provoquent d'accès au cache de données. Un int32\_t est implanté sur 4 octets.

**Rappel (ii)** En langage C les éléments du tableau X à 2 dimensions sont rangés en mémoire dans l'ordre : X[0][0], X[0][1], X[0][2], ... , X[0][15], X[1][0], X[1][1], X[1][2], ... , X[1][15], X[2][0] ...

**B1 (1 point).** Déterminer la taille d'une ligne (en octets) et donner le nombre de bits des champs offset, index et étiquette d'une adresse.

**B2 (1 point).** Représenter dans le schéma ci-dessous les cases valides du cache après 2 itérations de la boucle interne, en précisant les indexes (il y a au plus 2 cases valides).

Index	Adresse	Mot0	Mot15	Mot14	Mot13	Mot2	Mot1	Mot0
.....								
.....								

**B3 (1 point).** On rappelle qu'une case du cache contient une ligne. Quels sont les éléments de X qui peuvent occuper le mot d'offset 12 de la case d'index 3 du cache ? Même question avec le mot d'offset 16 de la case d'index 256.

**B4 (1 point).** Calculer, en le justifiant, le nombre de miss sur le cache de données rencontrés lors de l'exécution de cette fonction. En déduire le taux de MISS sur le cache de données.

**B5 (0.5 point).** La compilation de ce code a produit une boucle interne contenant 8 instructions, qui mettrait donc 8 cycles à s'exécuter si le système mémoire était parfait. On néglige l'effet des miss sur le cache d'instructions. En utilisant le résultat de la question B4, calculer le nombre de cycles moyen par itération dans la boucle interne si un miss de données coûte 11 cycles.

**B6 (0.5 point).** Comment peut-on réduire le taux de miss ?