Nom :	Prénom :
N° Étudiant :	Groupe de TD :

TME Solo 2021 – 2022 – Sujet n°1

Architecture des ordinateurs 1 – LU3IN029

Durée : 0h55

Documents autorisés : Aucun document ni machine électronique n'est autorisé à l'exception du mémento MIPS.

Le barème indiqué pour chaque question n'est donné qu'à titre indicatif. Le barème total est lui aussi donné à titre indicatif. Merci de rendre la feuille.

Étapes préliminaires à suivre scrupuleusement :

1. Créez un répertoire pour le TME solo à la racine de votre compte qui devra contenir les codes réalisés, en tapant une à une et dans l'ordre les commandes suivantes (ce qui est après le signe > ci-dessous) dans un terminal :
   `> cd`
   `> mkdir TMESolo_<nom>` (`<nom>` est à remplacer par votre nom en minuscule sans espace ni accent)
   `> chmod -R go-rwx .`
   Attention : cette dernière commande est très importante car elle empêche d'autres utilisateurs d'accéder à vos fichiers. Si vous ne la faites pas correctement et qu'un autre étudiant copie vos fichiers, vous risquez d'obtenir la note de 0.
   Remarque : la détection de plagiat sera faite automatiquement par logiciel.
2. Ouvrez un éditeur de texte et sauvegardez le fichier dans le répertoire créé (nommé `TMESolo_<nom>`) sous le nom `rapport.txt`. Ce fichier contiendra votre rapport de TME solo. Vous devrez y copier-coller tous vos codes réponses. Vous indiquerez bien le numéro de la question auquel se rapporte chaque code.
   Important : ce fichier doit se trouver dans le répertoire `TMESolo_<nom>` et s'appeler `rapport.txt`
3. Lancez Mars (commande `mars`) et composez le TME solo en répondant aux questions ci-dessous. nregistrez bien tous vos codes dans le répertoire `TMESolo_<nom>`.

Soumission de votre devoir à la fin du TME solo

1. Créez une archive contenant vos codes et votre rapport avec les commandes suivantes :
   `> cd`
   `> tar -cvf tmesolo_<nom>.tar TMESolo_<nom>/`
2. Déposez l'archive dans Moodle : dans la section "TME solo : information et organisation" il y a une remise de devoir intitulée TME solo de 8h30. Deux tentatives sont autorisées au cas où vous vous tromperiez de fichiers. Attention vous devez soumettre avant 9h30 (sauf tiers-temps qui doivent soumettre avant 10h), si vous modifiez votre rendu après cette heure vous serez en retard et pénalisé.

Consigne importante : il vous est demandé de mettre des commentaires dans vos codes pour indiquer la correspondance entre les registres utilisés et les variables des programmes.

Le TME solo est sur 20 points + 5 points de bonus : la première question est sur 14 points et la deuxième question est sur 11 points. Vous devez répondre aux questions dans l'ordre.

Exercice 1 : Occurrences et nombre de chiffres décimaux – 25 points

Question 1.1 : 14 points

On considère le programme C donné ci-dessous. Ce programme lit un entier `user_num` au clavier puis appelle la fonction `occ_num_chiffre` qui calcule le nombre de chiffres dans la représentation décimale d'un entier ainsi que le nombre d'occurrences de chacun des chiffres enregistré dans un tableau passé en paramètre. Le nombre de chiffres renvoyé par la fonction est affiché.

```
int user_num;             // entier non initialisé (0 par défaut)
int tab_chiffre[10];      // tableau de 10 entiers (mis à 0 par défaut)

int occ_num_chiffre(int n, int t[]){
  int nb_c = 1;           // nombre de chiffres
  int q = n;
  int r;
  while (q >= 10){
    r = q % 10;           // reste de la divison par 10
    q = q / 10;           // quotient de la divison par 10
    t[r] += 1;            // +1 du nombre d'occurrences du chiffre r
    nb_c += 1;            // nombre de chiffres + 1
  }
  t[q] += 1;
  return nb_c;
}

void main(){
  int num_ch;               // nombre de chiffres décimaux
  scanf("%d", &user_num);   // lire la valeur de user_num au clavier

  num_ch = occ_num_chiffre(user_num, tab_chiffre);
  printf("%d", num_ch);     // affichage du nombre de chiffres dans l'écriture
                            // décimale de user_num
  exit();
}
```

Dans un fichier nommé `Q1.s` (et enregistré dans le répertoire pour le TME solo), écrivez un programme assembleur correspondant au code C ci-dessus.

Important : Les variables locales peuvent être optimisées en registre et globalement votre code peut être optimisé **mais** vous devez suivre scrupuleusement les conventions habituelles d'utilisation des registres et du cours. **Toute allocation en pile devra être assortie d'un commentaire justifiant le nombre d'octets alloués.**

Testez votre programme.

Par exemple si on entre la valeur 1010_{10}, le programme doit afficher 4, et à la fin de l'exécution le tableau d'occurrences en mémoire doit contenir 2 dans la case d'indice 0 et 2 dans la case d'indice 1.

Par exemple, si on entre la valeur 111222333_{10}, le programme doit afficher 9, et à la fin de l'exécution le tableau d'occurrences en mémoire doit contenir 3 dans les cases d'indice 1, 2 et 3.

N'oubliez pas de copier-coller votre code assembleur dans votre rapport.

Solution:

Programme principal et les données globales :

```
.data
user_num:
    .align 2
    .space 4
```

```
tab_chiffre:
    .align 2
    .space 40  # alloue 10 mots initialisés par défaut à 0

.text
main:
    addiu $29, $29, -12 # nv = 1 + na = 2
    ori   $2,  $0, 5
    syscall
    lui   $8, 0x1001
    sw    $2, 0($8)     # ecriture user_num
    lw    $4, 0($8)     # user_num = arg 1 de occ_num_chiffre
    lui   $5, 0x1001
    ori   $5, $5, 4     # tab_chiffre = arg 2 de occ_num_chiffre
    jal   occ_num_chiffre

    ori   $4, $2, 0     # affichage du resultat (num_ch)
    ori   $2, $0, 1
    syscall

    addiu $29, $29, 12
    ori   $2, $0, 10
    syscall
```

Solution:

Fonction `occ_num_chiffre` :

```
occ_num_chiffre:
    addiu $29, $29,-16    # nv = 3 + na = 0  + nr = 0 + $31
    sw    $31, 12($29)

    or    $8, $0, $4      # q = n
    or    $9, $0, $0      # r = 0
    ori   $2, $0, 1       # nb_c = 0
    ori   $10, $0, 10
while:
    slt   $11, $8, $10  # $11 = 1 si q < 10
    bne   $11, $0, fin_while
    divu  $8, $10         # division par 10
    mfhi  $9              # r
    mflo  $8              # q
    sll   $12, $9, 2      # r * 4
    addu  $12, $12, $5    # adresse de tab[r]
    lw    $13, 0($12)     # lecture tab[r]
    addiu $13, $13, 1     # incrementation
    sw    $13, 0($12)     # tab[r] <- tab[r] + 1
    addiu $2,  $2, 1      # nb_c + 1
    j while
fin_while:
    sll   $12, $8, 2      # q * 4
    addu  $12, $12, $5    # adresse de tab[q]
    lw    $13, 0($12)     # lecture tab[q]
    addiu $13, $13, 1     # incrementation
    sw    $13, 0($12)     # tab[q] <- tab[q] + 1

    lw    $31, 12($29)
    addiu $29, $29, 16
    jr    $31
```

Question 1.2 : 11 points

On souhaite ajouter au code précédent la fonction `affichage` permettant d'afficher le contenu du tableau d'occurrences. Cette fonction est appelée dans le `main` **avant** l'affichage du nombre de chiffres dans la réprésentation décimale de `user_num`. Le code C résultant est donné ci-dessous.

Rappels :
L'appel système pour afficher un caractère en Mars a le numéro 11.
L'instruction `ori $10, $0, 'x'` met la valeur ASCII du caractère `x` dans `$10`.

```
int user_num;           // entier non initialisé (0 par défaut)
int tab_chiffre[10];    // tableau de 10 entiers (mis à 0 par défaut)

void affichage(int t[]){
  int j;
  for (j=0; j<10; j++){
    printf("%d", j);    // affichage j en décimal : appel système numero 1

    printf(":");        // affichage du caractère ':' (codage ASCII 0x3A)
                        // appel système numéro 11, argument dans $4

    printf("%d", t[j]); // affichage t[j] en décimal : appel système numero 1

    printf(";");        // affichage caractère ';' (codage ASCII 0x3B)
                        // appel systeme numéro 11, argument dans $4
  }
  printf("\n");         // affichage du caractère '\n' (codage ASCII 0xA)
                        // appel systeme numéro 11, argument dans $4
  return;
}

int occ_num_chiffre(int n, int t[]){
  // la fonction est inchangée par rapport à la question précédente
}

void main(){
  int num_ch;
  scanf("%d", &user_num);

  num_ch = occ_num_chiffre(user_num, tab_chiffre);
  affichage(tab_chiffre); // affichage du tableau d'occurrences
  printf("%d", num_ch); // affichage du nombre de chiffres dans l'écriture
                        // décimale de user_num
  exit();
}
```

Copiez votre fichier contenant le code réponse de la question précédente en l'enregistant sous le nom `Q2.s` (dans le répertoire du TME solo) dans Mars.

Modifiez le programme principal pour ajouter un appel à la fonction `affichage` **avant** l'affichage de la valeur de la variable `num_ch` (2,5 points). Puis, programmez la fonction `affichage` (8,5 points).

Important : Ici encore, les variables locales peuvent être optimisées en registre et globalement votre code peut être optimisé. Les conventions habituelles d'utilisation des registres et du cours doivent toujours être suivies. **Toute allocation en pile (nouvelle ou modifiée) doit être assortie d'un commentaire justifiant le nombre d'octets alloués.**

Testez votre programme pour vérifier qu'il fonctionne.

N'oubliez pas de copier-coller votre code assembleur dans votre rapport.

Solution:

Programme principal modifié (sans les données globales identiques ici) :

```
.text
main:
    addiu $29, $29, -12 # nv = 1 + na = 2
    ori   $2,  $0, 5
    syscall
    lui   $8, 0x1001
    sw    $2, 0($8)     # ecriture user_num
    lw    $4, 0($8)     # user_num = arg 1 de occ_num_chiffre
    lui   $5, 0x1001
    ori   $5, $5, 4     # tab_chiffre  = arg 2 de occ_num_chiffre
    jal   frequence_num_chiffre
    ori   $16, $2, 0    # num_ch = resultat, à sauvegarder dans un reg persistant ou en pile

    lui   $4, 0x1001
    ori   $4, $4, 4     # arg 1 = tab_chiffre, à recharger
    jal   affichage

    ori   $4, $16, 0    # affichage de num_ch
    ori   $2, $0, 1
    syscall
```

Solution:

Fonction `affichage` :

```
 affichage:
    addiu $29, $29, -16  # nv = 1, nr = 2 + $31, na = 0
    sw    $31, 12($29)
    sw    $17, 8($29)
    sw    $16, 4($29)

    xor   $16, $16, $16  # j = 0 dans registre persistant car appels systeme
    ori   $17, $4, 0     # sauvegarde de t à car appels systeme
for:
    slti  $8, $16, 10
    beq   $8, $0, fin_for
    ori   $4, $16, 0
    ori   $2, $0,  1
    syscall              # affichage j
    ori   $4, $0,  ':'
    ori   $2, $0,  11
    syscall              # affichage :
    sll   $4, $16, 2     # j * 4
    addu  $4, $4, $17    # @t[j]
    lw    $4, 0($4)      # t[j]
    ori   $2, $0,  1
    syscall               # affichage t[j]
    ori   $4, $0,  ';'
    ori   $2, $0,  11
    syscall               # affichage ;
    addiu $16, $16, 1     # j++
    j for
fin_for:
    ori   $4, $0, '\n'    # affichage \n
    ori   $2, $0,  11
    syscall

    lw    $31, 12($29)
    lw    $17, 8($29)
    lw    $16, 4($29)
    addiu $29, $29, 16
    jr    $31
```