

Nom :
N° Étudiant :

Prénom :
Groupe de TD :

TME Solo 2022 – 2023 – 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 et consignes 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)
> cd TMEsolo_<nom> (<nom> est à remplacer par votre nom en minuscule sans espace ni accent)
> chmod -R go-rwx . (copier strictement cette commande, le "." inclus)
```

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. **Important :** indiquez en commentaire dans tous vos fichiers assembleur vos nom, prénom et numéro d'étudiant.
3. Lancez Mars (commande `mars` ou commande `java -jar /usr/local/mars/Mars4_5.jar`) et composez le TME solo en répondant aux questions ci-dessous. Enregistrez 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 réponses 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 "Remise TME solo de 8h30..." Deux tentatives sont autorisées au cas où vous vous tromperiez de fichier. 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 27 points : la première question est sur 12 points et la deuxième question est sur 15 points. Vous devez répondre aux questions dans l'ordre. La note finale sera calculée en donnant quelques points bonus.

Exercice 1 : Chiffrement de César – 27 points

Question 1.1 : 12 points

On considère le programme C donné ci-dessous. Ce programme affiche la chaîne de caractère `ch` puis la chiffre selon le chiffrement de Cesar dans la chaîne `ch2`. La chaîne résultante est affichée. On suppose des chaînes de caractères ne contenant que des lettres minuscules.

```
unsigned char ch[] = "monsecret";
unsigned char ch2[10];
unsigned char decalage = 1;

int cipher_cesar(unsigned char src[], unsigned char dst[], unsigned char
    decal){
    int i = 0;
    unsigned char tmp;

    while (src[i] != 0){
        tmp = src[i] - 'a' + decal;    // 'a' vaut 0x61
        dst[i] = 'a' + (tmp % 26);
        i++;
    }
    return i;
}

void main() {
    printf("%s", ch); // affichage chaîne de caractères
    cipher_cesar(ch, ch2, decalage);
    printf("%s", ch2); // affichage chaîne de caractères
    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. Il n'est pas demandé de sauvegarder les registres persistants, ni \$31, dans le main. **Toute allocation en pile devra être assortie d'un commentaire justifiant le nombre d'octets alloués.**

Testez votre programme. Il doit afficher "monsecret" puis "npotfdsfu".

Solution:

Programme principal et les données globales :

```
.data
ch: .ascii "monsecret"
ch2: .space 10
decalage: .byte 1

.text
    addiu $29, $29, -12 # nv = 0 + na = 3
    lui   $4, 0x1001    # adresse ch
    ori   $2, $0, 4
    syscall              # affichage ch

    lui   $4, 0x1001    # 1er param = ch
    ori   $5, $4, 0xA   # 2eme param = ch2
    ori   $10, $4, 20   # adresse decalage
    lbu   $6, 0($10)    # 3eme param = decalage
    jal   cipher_cesar
```

```

lui    $4, 0x1001
ori    $4, $4, 0xA    # ch2
ori    $2, $0, 4
syscall                # affichage

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

```

Solution:

Fonction cipher_cesar :

```

cipher_cesar:
    addiu $29, $29, -12    # nv = 2 + nr = 0 + 1 + na = 0
    sw    $31, 8($29)
    xor   $10, $10, $10    # i = 0
    # $4 = src, $5 = dst, $6 = decal
while:
    # ch[i] != 0

    addu  $9, $10, $4    # adresse src[i]
    lbu   $9, 0($9)      # src[i]
    beq   $9, $0, finwhile

    # tmp = src[i] - 0x61 + decal
    addiu $9, $9, -0x61    # src[i] - a
    addu  $12, $9, $6      # tmp = src[i] - a + decal

    # dst[i] = 'a' + tmp%26
    ori   $13, $0, 26
    div   $12, $13
    mfhi  $13              # tmp%26
    addiu $13, $13, 0x61    # tmp%26 + 'a'

    addu  $9, $10, $5      # adresse dst[i]
    sb    $13, 0($9)       # dst[i] = tmp%26 + 'a'

    # i++
    addiu $10, $10, 1
    j     while

finwhile:

    or    $2, $0, $10      # val retour = i
    lw    $31, 8($29)
    addiu $29, $29, 12
    jr    $31

```

Question 1.2 : 15 points

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

On souhaite désormais coder différemment la fonction de chiffrement de César. On considère donc le code C suivant :

```
unsigned char ch[] = "monsecret";           // chaine à chiffrer
unsigned char ch2[10];                     // chaine chiffrée
unsigned char decalage = 1;                 // decalage chiffrement de Cesar

int cipher_cesar(unsigned char src[], unsigned char dst[], unsigned char
    decal){
    // fonction de la question précédente
}

unsigned char encode_char_cesar(unsigned char c, unsigned char decal){
    unsigned char tmp;
    tmp = (c - 'a') + decal ; // 'a' vaut 0x61
    return ('a' + (tmp % 26));
}

int cipher_cesar2(unsigned char src[], unsigned char dst[], unsigned char
    decal){
    int i = 0;
    unsigned char tmp;

    while (src[i] != 0){
        tmp = encode_char_cesar(src[i], decal);
        dst[i] = tmp ;
        i++;
    }
    return i;
}

void main() {
    printf("%s", ch);           // affichage chaine de caractères
    cipher_cesar(ch, ch2, decalage);
    printf("%s", ch2);         // affichage chaine de caractères
    cipher_cesar2(ch, ch2, decalage);
    printf("%s", ch2);         // affichage chaine de caractères
    exit();
}
```

Un appel à la fonction `cipher_cesar2` est ajouté dans le programme principal qui affiche ensuite de nouveau la chaîne `ch2`. La fonction `cipher_cesar2` calcule aussi une chaîne chiffrée selon le chiffrement de César mais elle fait appel à la fonction `encode_char_cesar`.

Donner le code assembleur correspondant à ce programme en modifiant le code réponse de la question précédente (attention à bien composer dans le fichier `Q2.s`). Il est conseillé de commencer par la fonction `cipher_cesar2`, puis de coder la fonction `encode_char_cesar` avant le programme principal.

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. Ici encore, il n'est pas demandé de sauvegarder les registres persistants, ni `$31`, dans le programme principal. **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. Il doit afficher la chaîne "monsecret" puis afficher

deux fois "npotfdsfu".

Solution:

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

```
.text
    addiu $29, $29, -12 # nv = 0 + na = 3
    lui   $4, 0x1001    # adresse ch
    ori   $2, $0, 4
    syscall                               # affichage ch

    lui   $4, 0x1001    # 1er param = ch
    ori   $5, $4, 0xA   # 2eme param = ch2
    ori   $10, $4, 20   # adresse decalage
    lbu   $6, 0($10)    # 3eme param = decalage
    jal   cipher_cesar

    lui   $4, 0x1001
    ori   $4, $4, 0xA   # ch2
    ori   $2, $0, 4
    syscall                               # affichage

    lui   $4, 0x1001    # 1er param = ch
    ori   $5, $4, 0xA   # 2eme param = ch2
    ori   $10, $4, 20   # adresse decalage
    lbu   $6, 0($10)    # 3eme param = decalage
    jal   cipher_cesar2

    lui   $4, 0x1001
    ori   $4, $4, 0xA   # ch2
    ori   $2, $0, 4
    syscall                               # affichage

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

Solution:

Fonction encode_char_cesar.

```
encode_char_cesar:
    addiu $29, $29, -8 # nv = 1 + nr = 0 + 1 + na = 0
    sw    $31, 4($29)

    addiu $2, $4, -0x61
    addu  $2, $2, $5
    ori   $8, $0, 26
    div   $2, $8
    mfhi  $2
    addiu $2, $2, 0x61

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

Solution:

Fonction cipher_cesar2 :

```
cipher_cesar2:
    addiu $29, $29, -36 # nv = 2 + nr = 4 + 1 + na = 2
    sw    $31, 32($29)
    sw    $19, 28($29)
```

```

sw    $18, 24($29)
sw    $17, 20($29)
sw    $16, 16($29)

xor    $16, $16, $16    # i = 0
# $4 = src, $5 = dst, $6 = decal
ori    $17, $4, 0        # src
ori    $18, $5, 0        # dst
ori    $19, $6, 0        # decal
while2:
# ch[i] != 0

addu   $9, $16, $17    # adresse src[i]
lbu    $9, 0($9)        # src[i]
beq    $9, $0, finwhile2

# encode char cesar(src[i], decal)
ori    $4, $9, 0        #1er param = src[i]
ori    $5, $19, 0        #2eme param = decal
jal    encode_char_cesar
addu   $9, $16, $18
sb     $2, 0($9)        # dst[i] = res de l'appel

# i++
addiu  $16, $16, 1
j     while2

finwhile2:

or     $2, $0, $16    # val retour = i
lw     $31, 32($29)
lw     $19, 28($29)
lw     $18, 24($29)
lw     $17, 20($29)
lw     $16, 16($29)
addiu  $29, $29, 32
jr     $31

```