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

TME Solo 2023 – 2024 – 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.

Le TME solo est composé d'un exercice sur 21 points : la première question est sur 8 points, la deuxième sur 5 et la troisième sur 8. Vous devez répondre aux questions dans l'ordre.

Exercice 1 : Construction d'un programme assembleur MIPS déterminant si une chaîne de caractère forme un palindrome – 21 points

On considère le programme C donné ci-dessous. Ce programme déclare une chaîne de caractères, `str`, en calcule la longueur par appel à la fonction `len`, puis détermine si cette chaine forme un palindrome, par appel à la fonction `est_palin`, et enfin affiche "oui" si c'est un palindrome, et "non" dans le cas contraire. Ces affichages correspondent à des chaînes de caractères également implantées en variables globales (`ok` et `ko`).

Un palindrome est un mot (ou une phrase) qui se lit de la même façon de droite à gauche et de gauche à droite. Par exemple, les mots "abcba" et "abba" sont des palindromes, le mot "ab" ne l'est pas.

```c
char str[] = "abcba";
char ok[] = "oui";
char ko[] = "non";


int len(char * str) {
  // ... voir question 2

}


int est_palin(char * str, int idx_last_char) {
  // ... voir question 3
}


int main() {
    int n;
    int res;

    n = len(str);
    res = est_palin(str, n - 1);

    if (res == 1) {
        printf("%s", ok);
    }
    else {
        printf("%s", ko);
    }

    return 0;
}
```

On se propose d'écrire le programme assembleur MIPS associé, en différentes étapes.

Consignes importantes :

- Il vous est demandé de mettre des commentaires dans vos codes pour indiquer la correspondance entre les registres utilisés et les variables des programmes.
- 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.

Question 1.1 : 8 points

Dans un fichier nommé **`Q1.s`** (et enregistré dans le répertoire pour le TME solo), écrivez un programme assembleur correspondant à la section `data`, et à la section `text` comprenant le programme principal (`main`) uniquement. Lors des appels à `len` et `est_palin`, `str` désigne l'adresse du premier élément de la chaine `str`.

Testez votre programme pour différentes valeurs de la chaîne `str`. Pour cela, il est nécessaire d'implanter deux fonctions minimales pour `len` et `est_palin` (uniquement l'instruction de retour de fonction). Décrivez le contenu du segment de données, octet par octet, pour la zone correspondant à `str`.

Solution:

Programme principal et les données globales :

```
.data
   str : .asciiz "abcba"
   ok : .asciiz "oui"
   ko : .asciiz "non"

.text

main :   addiu $29, $29, -16  # res et n et 2 param

   lui $4, 0x1001
   jal len
   sw $2, 0($29)      # n = len(str)

   lui $4, 0x1001
   addu $5, $0,$2
   addiu $5, $5, -1
   sw $2, 4($29)      # res = est_palin(str, n-1)


   beq $2, $0, aff_ko
   or $5, $0, 6       # offset ok
   j aff

aff_ko:  or $5, $0, 10     # offset ko

aff : lui $4, 0x1001    # affichage résultat
   addu $4, $4, $5
   ori $2, $0, 4
   syscall

   addiu $29, $29, 16

   ori $2, $0, 10
   syscall
```

Le contenu de la mémoire à partir de l'adresse `0x10010000` est le suivant (par adresse croissante) : `0x61 0x62 0x63 0x62 0x61 0x0A 0x00`.

La fonction `len` est définie comme suit. La chaine est parcourue de son premier à son dernier élément, et un compteur est incrémenté à chaque nouvel élément rencontré. A l'issue du parcours, la valeur du compteur est renvoyée à l'appelant.

```
int len(char * str) {
    int c = 0;
    while (str[c] != '\0') {
        c += 1;
    }
```

```
    return c;
}
```

Question 1.2 : 5 points

Enrichissez votre programme en complétant la fonction `len`. Vous sauvegarderez cette version du programme dans le fichier **`Q2.s`**

Testez votre programme pour vérifier qu'il fonctionne. Vous décrirez les tests effectués.

Solution:

Fonction `len` :

```
len : addiu $20, $29, -8   # $31 et variable locale c
   sw $31, 4($29)
   sw $0, 0($29)      # c = 0

   xor $2, $2, $2     # optimisation de c dans le registre 2

boucle_len:
   lb $10, 0($4)
   beq $10, $0, fin_boucle_len
   addiu $4, $4, 1
   addiu $2, $2, 1
   j boucle_len

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

La fonction compte tous les octets parcourus à partir de l'adresse de base passée en argument, jusqu'à rencontrer le caractère de fin de chaine (0x00) exclu.

La fonction `est_palin` est définie comme suit. Les deux éléments extrémaux de la chaine sont repérés par leurs indices dans le tableau `idx_deb` et `idx_fin` ; ces éléments sont comparés, et s'ils sont égaux, le même procédé est réitéré sur les éléments internes. Si les éléments comparés ne sont pas égaux, on peut conclure immédiatement que le mot analysé ne forme pas un palindrome.

```
  int est_palin(char * str, int idx_last_char) {
    int idx_deb = 0;
    int idx_fin = idx_last_char;
    while (idx_deb <= idx_fin) {
        if (str[idx_deb] != str[idx_fin]) {
            return 0;
        }
        idx_deb += 1;
        idx_fin -= 1;
    }

    return 1;
}
```

Question 1.3 : 8 points

Enrichissez votre programme en complétant la fonction `est_palin`. Vous sauvegarderez cette version du programme dans le fichier **`Q3.s`**

Testez votre programme pour vérifier qu'il fonctionne. Vous préciserez les tests effectués.

Solution:

Fonction `est_palin` :

```
est_palin:
   addiu $29, $29, -12  #$31 et 2 variables locales idx_deb, idx_fin
   sw $31, 8($29)
   sw $5, 4($29)     # init de idx_fin
   sw $0, 0($29)     # init de idx_deb

   # $4 : str
   # optimisation en registre : $8 = idx_deb et $9 = idx_fin
   xor $8, $8, $8
   or $9, $5, $0

boucle_est_palin:
   slt $10, $9, $8      # $10 = 1 si fin < dbt
   bne $10, $0, fin_boucle_est_palin

   add $11, $4, $8      # adresse str+idx_deb
   add $12, $4, $9      # adresse str+idx_fin
   lb $13, 0($11)
   lb $14, 0($12)
   bne $13, $14, fin_ko
   addiu $8, $8, 1
   addiu $9, $9, -1
   j boucle_est_palin

fin_ko :
   ori $2, $0, 0
   j epilogue

fin_boucle_est_palin :
   ori $2, $0, 1

epilogue :
   lw $31, 8($29)
   addiu $29, $29, 12
   jr $31
```

Lors de la saisie de la chaîne "abcba", la valeur affichée par le programme `main` est "oui". Lors de la saisie de la chaîne "abca", la valeur affichée par le programme `main` est "non".