INF1600

Contrôle périodique 1

Mon nom de famille est : _____

Mon prénom est : _____

Mon matricule est : _____

J'affirme sur mon honneur avoir fait ce devoir sans l'aide de personne.

Réécrivez la phrase ci-après :

Le contrôle périodique est accompagné de fichiers `.h`, `.c`, `.s` et d'un `makefile`. Dans votre remise, fournissez un fichier zip avec l'ensemble des fichiers `*.s` que vous aurez complétés ainsi qu'un fichier pdf comportant vos réponses aux questions.

Pour générer l'exécutable, il vous suffit d'exécuter `make` à la console. Le programme s'exécute en entrant à la console :

```
$> ./exec <votre matricule>
```

Pour chaque fonction `*_asm.s` que vous devez implémenter ci-après, une version `*_c.c` vous est fournie à titre de référence. Il n'est pas nécessaire de produire un code assembleur équivalent à celui que le compilateur `gcc` vous donnerait à partir du code c. Il est cependant interdit de produire le résultat de la décompilation du code C comme réponse aux questions.

Notez également qu'il est possible de tester votre implémentation `*_asm.s` en modifiant le fichier main.c de sorte que la constante associée retourne 1. Par exemple, pour tester la question 3, on mettra :

```
#define TEST_Q3 1
```

Q0) (obligatoire) Pour ce travail, il vous est demandé de programmer en assembleur des fonctions données en C. Le contrôle périodique est personnalisé suivant votre matricule. Pour ce faire, il vous est demandé de compléter le calcul suivant pour déterminer les valeurs de `A, B, C` et `D`.

Votre matricule = _____ ;

Votre matricule % 2 = `A` = __ ;

Votre matricule % 3 = `B` = __ ;

Votre matricule % 5 = `C` = __ ;

Votre matricule % 7 = `D` = __ ;.

Q1) **(0.5 point)** Reproduisez l'affichage obtenu en exécutant le programme avec votre matricule en paramètre :

Pour ce travail, on désire implémenter les chiffrements par substitution[1] que sont le chiffrement de César[2] et le chiffrement de Vigenère[3]. Vous êtes invités à consulter les pages Wikipedia données en note de bas de page pour plus d'informations sur ces méthodes de chiffrement. On vous demande pour ce faire de complétez un certain nombre de fonctions en assembleur.

La première fonction demandée est `is_alphabetic_asm` qui renvoie 1 si le caractère reçu en paramètre est une lettre entre A et Z, quelle soit majuscule ou minuscule, et qui retourne 0 autrement. Suivez les directives données en commentaire dans le fichier `is_alphabetic_asm.s`.

Notes :

- Référez-vous à l'implémentation C proposée de `is_alphabetic_c(…)` dans `c_functions.c`.
- Mettez `TEST_Q2-Q4` à `1` pour tester votre code.

Q2) **(0.5 point)** Selon que la valeur de A déterminée à la Q0 vaille 0 ou 1, vous devez décommenter ou pas les lignes 10 et 11. Ce changement nécessite un ajout d'au plus une instruction à ligne 45. Recopiez ci-après la ligne ajoutée. Justifiez brièvement votre réponse.

Q3) **(1 point)** Les valeurs de A et de B déterminées à la Q0 vont déterminer la ligne 22 que vous devez ajouter à `is_alphabetic_asm.s`. Cette ligne sert à recopier le paramètre reçu par `is_alphabetic_asm` dans le registre identifié pour vous (`%cl`, `%ch` ou `%dh`, selon la valeur de B). Recopiez ci-après la ligne ajoutée. Justifiez brièvement votre réponse.

Q4) **(1 point)** Complétez le fichier `is_alphabetic_asm.s` de sorte que la fonction `is_alphabetic_asm` réalise la tâche demandée. Incluez le fichier modifié dans le zip que vous remettrez.

Contraintes : 0.25 point est accordé si, entre les lignes 28 et 36 incluses, vous n'utilisez pas plus de 10 instructions d'assembleur IA-32 réaliser la fonction demandée.

[1] Voir https://fr.wikipedia.org/wiki/Chiffrement_par_substitution.
[2] Voir https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage.
[3] Voir https://fr.wikipedia.org/wiki/Chiffre_de_Vigen%C3%A8re.

La seconde fonction demandée est `substitute_asm` qui reçoit en paramètres un caractère et une clé et retourne un caractère de substitution préservant la casse si le caractère reçu est une lettre entre A et Z, sinon elle retourne le caractère reçu. On supposera que la clé est un entier positif entre 0 et 25.

Exemples :

- Si la lettre reçue est '|' et que la clé est 4, la fonctionne retourne '|' (le caractère n'est pas une lettre de l'alphabet, il n'est donc pas modifié).
- Si le caractère reçu est 'a' et que la clé est 4, la fonctionne retourne 'e' (4 lettres plus loin, casse préservée).
- Si la lettre reçue est 'A' et que la clé est 4, la fonctionne retourne 'E' (4 lettres plus loin, casse préservée).
- Si la lettre reçue est 'z' et que la clé est 4, la fonctionne retourne 'd' (4 lettres plus loin en reprenant à 'a', casse préservée).
- Si la lettre reçue est 'Z' et que la clé est 4, la fonctionne retourne 'D' (4 lettres plus loin en reprenant à 'A', casse préservée).

Notes :

- Référez-vous à l'implémentation `C` proposée de `substitute_c(…)` dans `c_functions.c`.
- Mettez `TEST_Q5-Q7` à `1` pour tester votre code.

Q5) **(1 point)** Considérez les lignes 76 et 77 de `c_functions.c` reproduites ci-après. À quoi servent ces lignes ?

```
76   else if( (src & 0x20) != (dst & 0x20) )
77      dst -= 26;
```

Q6) **(1 point)** Considérez les lignes 76 et 77 de `c_functions.c`. Donnez un exemple de caractère d'entrée et de clé pour lesquels la ligne 77 sera exécutée.

Q7) **(1 point)** Complétez le fichier `substitute_asm.s` de sorte que la fonction `substitute_asm` réalise la tâche demandée. Votre implémentation doit appeler la fonction `is_alphabetic_asm` ou `is_alphabetic_c` pour valider que le caractère reçu est une lettre entre A et Z inclus. Incluez le fichier modifié dans le zip que vous remettrez.

Contraintes : **0.25 point** est accordé au fait que votre code appelle `is_alphabetic_asm` ou `is_alphabetic_c` ; **0.75 point** est accordé à la fonctionnalité.

Q8) **(2 points)** Implémentez la fonction `caesar_encrypt_asm` qui utilise un chiffrement par décalage pour encrypter une chaine de caractères `src` en utilisant une clé non nulle `key`. La chaine de caractères encryptée est retournée dans `dst`. Votre implémentation doit appeler la fonction `substitute_asm` ou `substitute_c`. Incluez le fichier modifié dans le zip que vous remettrez.

Notes :

- Référez-vous à l'implémentation `C` proposée de `caesar_encrypt_c(…)` dans `c_functions.c`.
- Mettez `TEST_Q8` à `1` pour tester votre code.

Contraintes : **0.5 point** est accordé au fait que votre code appelle `substitute_asm` ou `substitute_c` ; **1.5 points** sont accordés à la fonctionnalité.

Q9) **(2 points)** Implémentez la fonction `vigenere_encrypt_asm` qui utilise un chiffrement de Vigenère pour encrypter une chaine de caractères `src` en utilisant une mot-clé `keyword`. La chaine de caractères encryptée est retournée dans `dst`. Votre implémentation doit appeler la fonction `substitute_asm` ou `substitute_c`. Incluez le fichier modifié dans le zip que vous remettrez.

Notes :

- Référez-vous à l'implémentation `C` proposée de `vigenere _encrypt_c(…)` dans `c_functions.c`.
- Mettez `TEST_Q9` à `1` pour tester votre code.

ontraintes : **0.5 point** est accordé au fait que votre code appelle `is_alphabetic_asm` ou `is_alphabetic_c` ; **0.5 point** est accordé au fait que votre code appelle `substitute_asm` ou `substitute_c` ; **1.0 point** est accordé à la fonctionnalité.