Projet – Compression de données

Le but de ce projet est d'implémenter un algorithme de compression de données sans perte : l'algorithme `LZ77`. Il s'agit de réaliser deux codes : un code de compression et un code de décompression. Les codes seront réalisés en assembleur MIPS, et executés grâce à l'émulateur `Mars`. Le projet est à réaliser en binôme, et **à rendre avant le mercredi 9 décembre 2015 à minuit** en respectant les spécifications données dans la section 1.

L'archive `PROJET_ressources.tar.gz` contient :
- le présent document,
- Des fichiers de test : `Voltaire.txt`, `Pirouette.txt` et `Lepetitprince.txt`,
- l'archive java `Mars.jar`.

1 Spécifications

Vos deux codes, qui seront commentés, prendront chacun la forme d'un fichier `.s`, exécutable grâce à l'émulateur `Mars`. Lors de son exécution, chaque code demandera à l'utilisateur de rentrer une chaîne de caractère désignant un fichier au format texte ou au format compressé (`.txt`) (resp. `.lz77`). Le résultat de l'exécution sera un autre fichier nommé `.lz77` (resp. `.txt`), correspondant au résultat de la compression (resp. décompression) du fichier texte (resp. du fichier compressé).

Le rendu prendra la forme d'une archive nommée `Nom1_Nom2.tar.gz`, qui contiendra : les deux fichiers `.s`, l'émulateur `Mars`, Les fichiers de test, et un fichier `readme.txt` mentionnant les noms et prénoms de chaque étudiant du binôme, et comprenant les réponses aux questions de la section 3, ainsi que l'explication des fonctionnalités implémentées.

Cette archive sera déposée avant le mercredi 9 décembre 2015 à minuit sur la plateforme `Moodle`.

La suite du document présente les algorithmes de compression et de décompression (section 2), des questions, et quelques aides sur les appels systèmes disponibles avec l'émulateur `Mars` (section 4).

2 `LZ77`

`LZ77` est un algorithme de compression de données sans perte proposés par Abraham Lempel et Jacob Ziv en 1977 (d'où son nom). Cet algorithme pose les bases de la plupart des algorithmes de compression par dictionnaire. `LZ77` est notamment utilisé dans l'algorithme `deflate` (avant un codage de Huffman) et pour la compression des fichiers dans le système de fichier Windows NTFS (cf. `https://fr.wikipedia.org/wiki/LZ77_et_LZ78`).

2.1 Principe

La description ci-dessus est extraite de transparents à l'adresse `http://vernier.frederic.free.fr/Teaching/InfoTermS/InfoNumerique/rennes_28784_cours2_comp.pdf`. Voir aussi à l'adresse `http://d.nouchi.free.fr/TER/c6.html` pour une autre présentation plus détaillée de l'algorithme.

L'algorithme `LZ77` utilise une fenêtre glissante. On fait glisser une fenêtre de N symboles de la gauche vers la droite. La fenêtre est composée de 2 parties : à droite le tampon de lecture de F symboles dans lequel se trouvent les symboles en attente de compression, à gauche le tampon de recherche qui constitue le dictionnaire des symboles déjà lus et compressés. À chaque étape, on recherche (dans le tampon de recherche) la plus longue chaîne qui se répète au début du tampon de lecture. Cette répétition sera codée (p, l, c) où p est la distance entre le début du tampon de lecture et la position de la répétition (on prendra la plus grande possible), l est la longueur de la répétition et c est le symbole suivant dans le texte. Si $N - F = 2^{e1}$ et $F = 2^{e2}$, il faut $e1$ bits pour coder p et $e2$ pour l. Si il n'y a pas de répétition, on prend $p = l = 0$ (et c est le symbole lu).

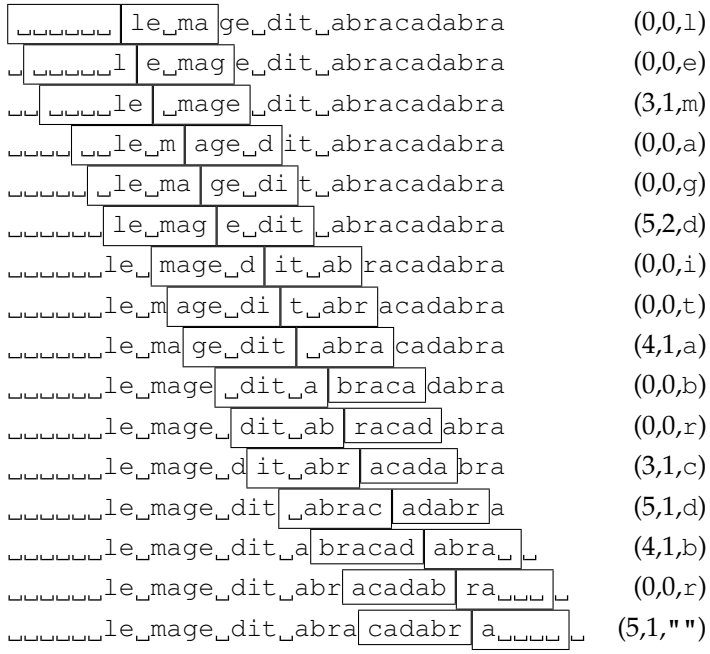
Pour la décompression, on reconstruit le dictionnaire en faisant coulisser la fenêtre de gauche à droite puis la séquence initiale.

2.2 Algorithme simpliste de la compression

Tant que (le tampon de lecture n'est pas vide) faire
{
Trouver la plus longue chaîne du tampon de recherche
identique à une chaîne située au début du tampon de lecture ;
Si la longueur de la chaîne > 0
{
On retourne le triplet (p,l,c) où
p est la position de la chaîne dans le tampon de recherche,
l est sa longueur, et c est le caractère suivant dans le tampon de lecture ;
On décale la fenêtre glissante de $l + 1$ positions ;
}
Sinon
{
On renvoie le triplet (0,0,c) où
c est le premier caractère du tampon de lecture ;
On décale la fenêtre glissante de 1 position ;
}
}

2.3 Exemple

On veut coder `le mage dit abracadabra` avec une fenêtre de taille $N = 11$ dont $F = 5$ pour le tampon de lecture.



2.4 Algorithme simpliste de la décompression

Tant que (il reste un triplet (p,l,c)) faire
{
Recopier les l caractères (de positions p ... $p + l - 1$) du tampon de recherche au début du tampon de lecture ;
Ajouter le caractère c au tampon de lecture ;
Décaler la fenêtre de lecture de $l + 1$ positions ;
}

2.5 Exemple

	␣␣␣␣␣␣	l␣␣␣␣	(0,0,l)
	␣␣␣␣␣l	e␣␣␣␣	(0,0,e)
	␣␣␣␣le	␣m␣␣␣	(3,1,m)
	␣␣le␣m	a␣␣␣␣	(0,0,a)
	␣le␣ma	g␣␣␣␣	(0,0,g)
	le␣mag	e␣d␣␣	(5,2,d)
le␣	mage␣d	i␣␣␣␣	(0,0,i)
le␣m	age␣di	t␣␣␣␣	(0,0,t)
le␣ma	ge␣dit	␣a␣␣␣	(4,1,a)
le␣mage	␣dit␣a	b␣␣␣␣	(0,0,b)
le␣mage␣	dit␣ab	r␣␣␣␣	(0,0,r)
le␣mage␣d	it␣abr	ac␣␣␣	(3,1,c)
le␣mage␣dit	␣abrac	ad␣␣␣	(5,1,d)
le␣mage␣dit␣a	bracad	ab␣␣␣	(4,1,b)
le␣mage␣dit␣abr	acadab	r␣␣␣␣	(0,0,r)
le␣mage␣dit␣abra	cadabr	a␣␣␣␣	(5,1,"")

3 Évaluation des performances

Il est recommandé de tester l'algorithme avec différents fichiers textes, notamment avec ceux fournis dans ce projet. Le fichier `readme.txt` sera complété par les réponses aux questions suivantes :

- Le taux de compression est-il le même pour les trois fichiers `.txt` ? pourquoi ?
- La compression peut-elle être négative ? Si oui, dans quels cas ?
- Peut-on obtenir un meilleur taux de compression avec d'autres valeurs de N et/ou de F ?
- Quels sont les points forts/faibles de l'algorithme `LZ77` ?

4 Mars et appels systèmes

L'émulateur Mars se lance à partir de l'archive java Mars.jar grâce à la commande `java -jar Mars.jar`.

On peut effectuer des appels système grâce au mot clé `syscall`. La liste complète des appels système disponibles ainsi que leur utilisation est décrite à l'adresse :
`http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html`
On notera que pour ouvrir un fichier, (appel système 13), l'adresse du fichier à passer est l'adresse depuis le répertoire duquel a été lancé `Mars`.