INF1600 — TP3
Programmation en assembleur et débogage

Giovanni Beltrame giovanni.beltrame@polymtl.ca
Imane Hafnaoui imane.hafnaoui@polymtl.ca
Karim Keddam karim.keddam@polymtl.ca

Polytechnique Montréal
Hiver 2016

Remise

Voici les détails concernant la remise de ce travail pratique :

— **Méthode :** sur Moodle (une seule remise par groupe).

— **Échéance :**

Groupes B2	08 & 09 Mars
Groupes B1	22 & 23 Mars

— **Format :** un seul fichier zip, dont le nom sera `<matricule1>-<matricule2>.zip`. Exemple : `0123456-9876543.zip`. L'archive doit contenir les fichiers `filter.s` et `tp3.c`.

— **Langue écrite :** français.

— **Distribution :** les deux membres de l'équipe recevront la même note.

Barème

Contenu	Points du cours
`filter.s`	6
Illisibilité du code (peu de commentaires, mauvaise structure...)	jusqu'à -1
Format de remise erroné (irrespect des noms de fichiers demandés, fichiers superflus, etc.)	jusqu'à -1
Retard	-0,025 par heure

Travail demandé

Vous êtes en charge d'implementer un filtre d'image en assembleur. Le filtre en question doit transformer une image couleurs en une images en niveaux de gris.
Des explications sur la maniére d'implémenter un tel filtre sont données plus bas.

Format BMP

Avant d'expliquer le filtre et la façon de l'appliquer, une présentation du format bitmap est nécessaire afin de vous faciliter la tâche.

Bitmap, souvent abrévié en BMP, est un format d'image développé par Microsoft et IBM. Il s'agit de l'un des formats les plus simples à utiliser dans la programmation.
Le format vient sous plusieurs types d'encodage et de compression. Pour ce travail, nous n'utilisons pas de compression et avons un encodage de 24 bits par pixel.
Dans ce type de BMP, le codage de l'image se fait en écrivant successivment les bits correspondant à chaque pixel, ligne par ligne en commençant par le pixel en bas à gauche.
Chaque pixel est représenté par 3 octets (24 bits) où chaque octet représente la valeur d'une des composantes de la couleur (format RGB). Les octets de chaque pixel doivent respecter l'ordre de l'alternance bleu, vert et rouge.

Naturellement, un fichier BMP possède une ou plusieurs entêtes qui contiennent divers informations sur l'image.
En réalité, les fichiers BMP qu'on utilise contiennent deux entêtes distinctes : une entête BMP et

une entête DIB. Afin de faciliter le travail, vous pouvez ignorer cette information et vous baser uniquement sur la représentation de l'entête fournie ci-bas. Si vous êtes curieux, vous pouvez toujours en apprendre plus sur le lien suivant : `https://en.wikipedia.org/wiki/BMP_file_format`.

Voici le contenu minimal d'une entête avec les adresses qui correspondent à chaque information :

0x00	'B'
0x01	'M'
0x02	Taille du Fichier
0x03	
0x04	
0x05	
0x06	0
0x07	
0x08	0
0x09	
0x0A	54
0x0B	
0x0C	
0x0D	
0x0E	40
0x0F	
0x10	
0x11	
0x12	Largeur
0x13	
0x14	
0x15	
0x16	Hauteur
0x17	
0x18	
0x19	
0x1A	1
0x1B	
0x1C	24
0x1D	
0x1E	0
0x1F	
0x20	
0x21	
0x22	Nombre de pixels
0x23	
0x24	
0x25	
0x26	11811
0x27	
0x28	
0x29	
0x2A	11811
0x2B	
0x2C	
0x2D	
0x2E	0
0x2F	
0x30	
0x31	
0x32	0
0x33	
0x34	
0x35	

Filtre

Le filtre que vous avez à implémentez est l'un des plus simples, il s'agit du filtre niveaux de gris.
La manière la plus facile de transformer une image couleurs en une image niveaux de gris est de remplacer, pour chaque pixel, les trois composantes par la même valeur. La valeur en question est la moyenne des trois composantes originales.

Voici un aperçu de la transformation que devrait causer votre filtre :


Fichiers fournis

Les fichiers nécessaires à la réalisation du TP sont dans l'archive `inf1600_tp3.zip`, disponible sur Moodle.

Voici la description des fichiers :

- `Makefile` : le *makefile* utilisé pour compiler et nettoyer le projet ;
- `tp3.c` : programme de test qui utilise les fonctions de référence et celle en assembleur. Ce fichier contient également du code C (non utilisé) qui permet d'écrire l'entête du fichier BMP, vous pouvez vous en inspirer.
- `filter.s` : fichier à compléter qui contient la fonction `filter_s()` qui produit l'image modifiée.
- `lena.bmp` : fichier de test.
- `poly.bmp` : fichier de test.
- `tux.bmp` : fichier de test.

Vous devez compléter le fichier `filter.s` et le remettre dans un archive zip.

Compilation et testing

Pour compiler le programme de (`tp3`), il est suffisant de taper :

```
$ make
```

et pour l'executer :

```
$ ./tp3 chemin_image_source chemin_resultat
```

par exemple :

```
$ ./tp3 ./images/tux.bmp ./monImage.bmp
```

Débogage

Vous pouvez déboguer votre programme avec `gdb`. Vous pouvez aller voir votre source avec `gdb` et insérer des points d'arrêt dans le code assembleur.

Si vous avez une erreur de segmentation, `gdb` vous indiquera à quelle ligne se produit celle-ci et vous pourrez alors observer le contexte (valeurs des registres, des variables, de la mémoire, de

la pile) et déterminer plus facilement la cause de cette erreur. Une erreur de segmentation arrive toujours lorsque le programme tente d'accéder à un emplacement mémoire invalide.

Une référence avec les commandes de `gdb` les plus utilisés se trouve ici : `http://web.cecs.pdx.edu/~apt/cs577_2008/gdb.pdf`.

Pour executer `gdb`, tapez :

```
$ gdb tp3
```

et puis cliquez sur *File*, *Target settings* et écrivez le nom du fichier de test à utiliser dans *Arguments*.