**Université de Strasbourg**
**UFR de Mathématique et d'Informatique**
**Semestre 3 – L2**

**RAPPORT DE PROJET – UE : ARCHITECTURE**

# TRAITEMENT D'IMAGES EN MIPS

*Réalisé par :*

ALCINDOR Marc Jodel D.

MOHAMED YONIS Ahmed

Grâce à ce projet sur les traitements d'images, nous avons eu l'opportunité de développer nos connaissances théoriques et pratiques sur le langage assembleur MIPS.
En effet, les différents types d'implémentations dont nous avons fait le choix nous ont demandé des heures de réflexions. Cela va sans dire que cela nous a permis de maitriser davantage ce langage qui parfois peut paraitre impossible à cerner.
L'objectif final du projet étant d'appliquer un filtre sur une image d'extension bmp, nous avions dû élaborer un plan de travail assez rigide, tout en essayant d'optimiser le plus possible nos lignes de codes. Dans les lignes qui suivent, nous allons présenter les différentes étapes que nous avons suivies, et qui nous ont amené au résultat convoité.
Il est également à noter qu'en termes d'outils techniques utilisés, nous avons procédé à la création d'un serveur discord lors de nos échanges à distance, mais également un dépôt git pour l'échange et la comparaison des calculs que nous avions décidé de faire chacun de notre côté.
Enfin, la bibliothèque l'Alinéa qui se trouve sur le campus universitaire a été tout au long de ce projet un endroit que nous avons particulièrement fréquenté pour les discussions d'implémentation, mais aussi l'écriture du code en elle-même.
En effet, initialement, l'idée était une répartition équitable des tâches. Au final, nous avions dû nous poser à chaque fois pour discuter, corriger ensemble les erreurs que nous avions fait chacun de notre côté.
Il est également à noter qu'initialement, étant tous deux des passionnés du langage C, qui est lui-même près du langage machine, nous avions en tout premier lieu réfléchi en code C pour enfin réfléchir à comment implémenter tout cela en Mips.
Dans les lignes qui suivront, nous décrirons les différentes étapes que nous avons suivies.

Dans un premier temps, nous avons ouvert le fichier lena.bmp grâce à l'utilitaire linux hexdump. Cela nous a permis de mieux comprendre le sujet, mais aussi l'écrire de l'image en binaire.

Ensuite, nous avons découpé l'image. En effet, nous avons calculé en fonction des données qui nous avaient été données dans le sujet, l'espace mémoire qu'il faudrait pour chaque sous-partie (l'entête du fichier, l'entête de l'image, la palette de couleur ...)

Dans un troisième temps, nous avons défini les lignes de codes pour l'ouverture d'un fichier en Mips, la fermeture d'un fichier et l'écriture dans un fichier.

Ce qui a été évident pour nous lors de nos discussions, c'est qu'il fallait recopier dans le nouveau fichier le début du fichier précédent (De l'entête jusqu'à la palette de couleur).

Ensuite, nous avions procédé aux calculs qui nous paraissaient au début les plus périlleux.
-Dans un premier temps, nous avons récupéré dans des variables l'adresse des masques *fx* puis par la suite *fy* pour procéder aux calculs de la convolution.
Il fallait les matrices *fx* et *fy*, mais en même temps, nous devions nous positionner sur les pixels de l'image en elle-même (Ce que nous avons considéré et appelé la matrice de l'image) et non sur les autres pixels comme l'entête ou la palette...

Ensuite, toujours suivant le sujet qui a été donné, en faisant en attention à bien prendre les valeurs absolues de *Gx* et de *Gy*, nous les avons additionnées avant de seuiller le résultat obtenu.

Enfin, il nous a fallu recopier ces pixels dans l'image résultante, puis la fermer.

En réalité, les deux implémentations qui nous ont pris le plus de temps lors du projet et qui ont parfois été au cœur de nos discussions, ont été les implémentations du nom du fichier final et du calcul de la convolution des matrices.
Au début, le nom du fichier résultant avait été codé en dur dans une variable (fichier_result : .asciiz lenaContour.bmp ). Elle était donc prédéfinie. Nous avons par la suite compris qu'elle était loin d'être avantageuse dans la mesure où l'on déciderait d'appliquer le filtre de sobel sur une image autre que lena.bmp. Cette nouvelle image aurait un nom prédéfini lenaContour.bmp.
Nous avons fini par opter pour une implémentation en dur du mot « Contour » dans une variable qui sera à ajouter à chaque fois à la suite du nom du fichier entrant, sans oublier l'extension bmp à la fin. Une implémentation qui a été faite en MIPS par Alcindor.
Les codes pour les calculs de la convolution des matrices ont été implémentés par Mohammed Yonis.
En apprenant chacun de nos erreurs et de ce que nous avons fait chacun de notre côté, nous avons fini par implémenter les autres calculs et ligne de code ensemble au fil du temps.

Toutefois, quoiqu'il ait fallu être minutieux dans l'écriture de ce projet, nous avons fait en sorte d'optimiser dans la mesure qu'il nous semblait possible toutes nos lignes de codes.