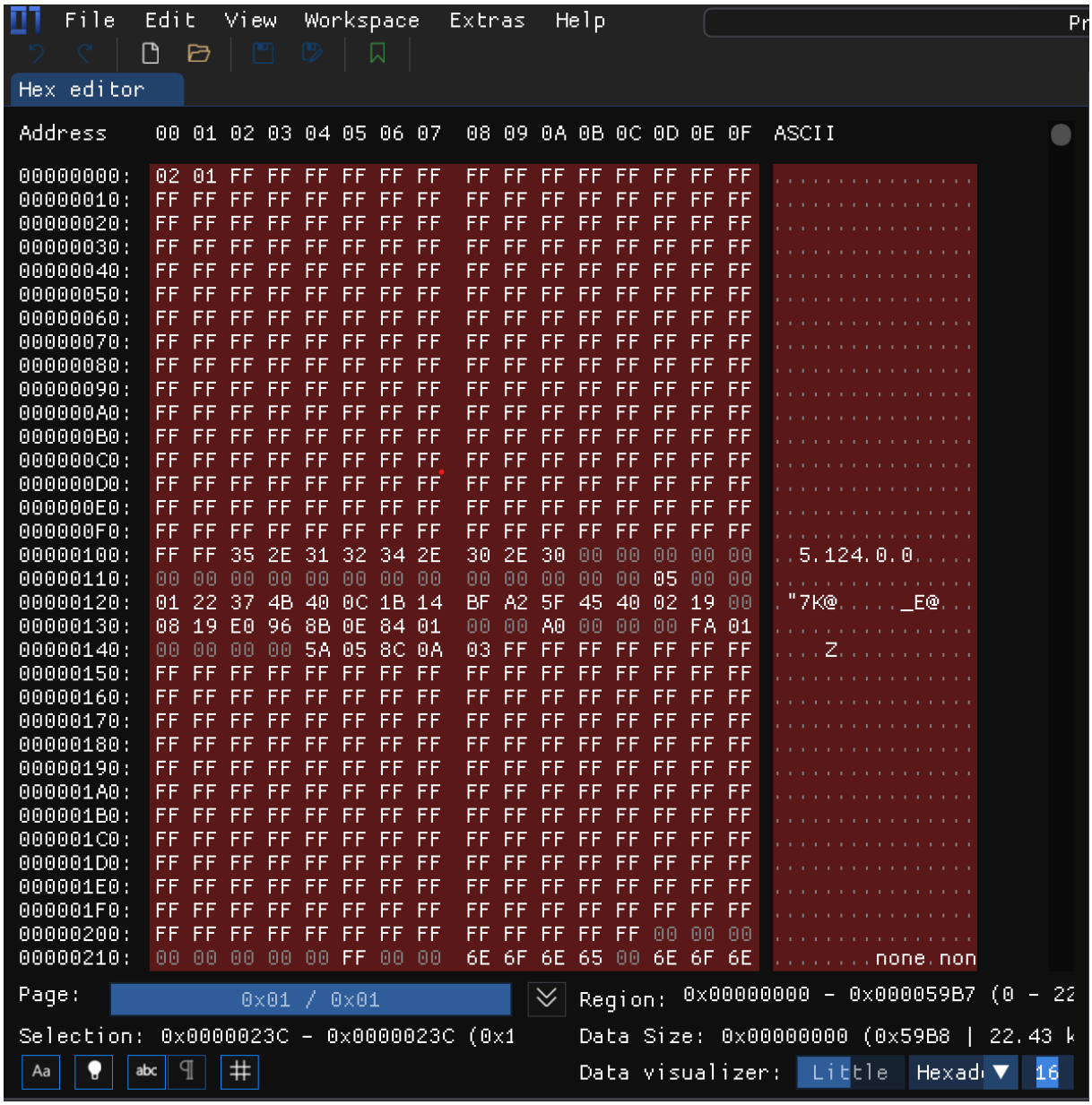
TP 2 – EL ARIF AMINE

I- Lire le fichier en ImHex



La première partie était de définir les partie intéressente dans le fichier binaire et pour faire cela on a définie une struct

```
struct fileStruct{

char metadata[0x248];
char data[0x56E6] ;
char endOfFile[0x8A] ;

};
```

La meta-data c’est la première partie du fichier contenant des informations sur le fichiers.

Le data est le contenue du ficher qu’on cherche à décoder.

Le endOfFile est la partie finale du fichier qui commence à partir de STOP.

On suite on passe au contenu de data, on cherche à trouver les cordonnée du GPS (latitude, longitude), grâce à l’indice donné par nos profs, j’ai pu définir cet partie on utilisant le struct suivant :

```
struct dataframe
{
    u8 offset[26];
    u8 longitude[4];
    u8 latitude[4];
    u8 offsetTwo[15];
};
```

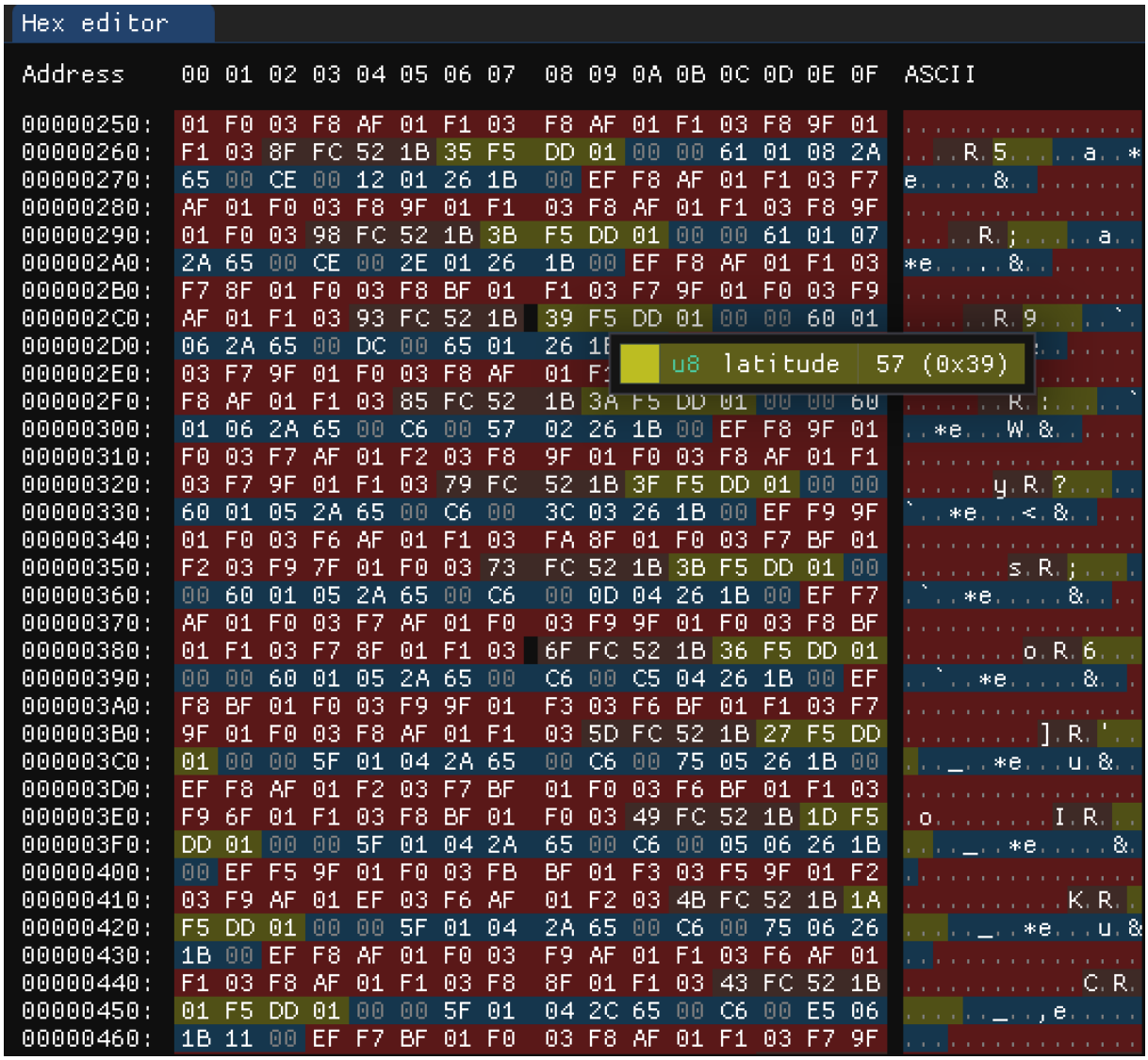
Cet partie va se répéter tout au long du partie data, grâce à cet partie :

```
dataframe frame[454] @ 0x248;
```

et finalement on doit définir la structure en disant que ca commence à partir de l’adresse 0x00 :

```
fileStruct var @ 0x00;
```

résultalt :



II- Script python pour lire les en csv

```
with open('13caaf25c97e29c8b45ae2e4f9f8ac3b_T0_2e526', 'rb') as file:
    file.seek(0x248)
    trames_bin = file.read()
    trames_size = len(trames_bin)
    trames_count = trames_size // frame_struct.size
   trames=[frame_struct.unpack(trames_bin[i*frame_struct.size:(i+1)*frame_stru
ct.size]) for i in range(trames_count)]
```

Cette partie du code lit le fichier binaire T0, se déplace à la position 0x248 (ignore la méta-data), lit le reste du fichier, calcule la taille et le nombre de trames, puis décode chaque trame du fichier.

```
with open('gpsFile.csv', 'w', newline='') as csvfile:
    fieldnames = ['Latitude', 'Longitude']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    writer.writeheader()
```

ensuite je crée un fichier CSV nommé ‘gpsFile.csv’, définit les en-têtes de colonnes comme ‘Latitude’ et ‘Longitude’, puis j’écris ces en-têtes dans le fichier.

```
for trame in trames:
        latitude =  '{:.6f}'.format(trame[1] / 10000000)
        longitude = '{:.6f}'.format(trame[2] / 10000000)

        writer.writerow({'Latitude': latitude, 'Longitude': longitude})
```

je formate ensuite les coordonnées à la bonne format.

Résultat! :

Leaflet Maps output

Your GPS data has been processed. Your Leaflet Map should be displayed below, and it's also **temporarily** av
GPSVisualizer.com. If something doesn't look like you expected it to, please contact me and explain the probl