# Rapport technique

Fait par : Alexandr Ivanov

**Description de la machine.**

La machine de Turing peut être décrite de façon suivante :

Notre machine de Turing se compose des éléments suivants :

La bande (B) : représente une bande finie avec les « 0 » sauf le point initial qui est « 1 ».
La tête de lecture (T) : le signe de la tête de lecture est « X » qui montre le positionnement sur la bande. Peut bouger à gauche et à droite.
La pile pour enregistrer l'adresse de l'entrée et sortie d'une boucle : garde l'adresse d'entrée et de sortie de la boucle.
Le compteur d'instructions : compte chaque instruction.
Le lecteur d'état : permet à la machine de récupérer la valeur de la case courante.

**Les éléments interagissent de manière suivante :**

La tête de lecture se déplace à gauche et à droite sur la bande de lecture qui se divise en cases qui peuvent avoir une valeur à la fois (soit 1 soit 0). Le compteur d'instructions compte chaque instruction exécutée par le processeur et permet de sauvegarder la valeur du nombre de l'instruction dans la pile pour savoir l'entrée en boucle. Le lecteur d'état vérifie si la condition est vraie pour la valeur de la case sur laquelle est positionnée la tête de lecture.

Le langage de notre programme qui peuvent être interprétés par notre interprétateur :

| Instruction | Sémantique |
|---|---|
| D | Fait bouger la tête de lecture à droite |
| G | Fait bouger la tête de lecture à gauche |
| 1 | Ecrit le « 1 » dans la case du positionnement de la tête de lecture et la déplace une fois à droite |
| 0 | Ecrit le « 0 » dans la case du positionnement de la tête de lecture et la déplace une fois à droite |
| boucle : | Commence la boucle (implémentation incomplète) |
| fin | Sortie de la boucle (implémentation incomplète) |
| if | Vérification de l'état de cellule |
| I | Montre l'état courant de la machine sur la sortie standard |

Les instructions sont séparées par un saut de ligne. La seule instruction qui s'écrit sur toute la ligne est le teste « si ». Exemple : si 1 fin.

**Implémentation en Python3**

La pile correspond à la liste « adresse=[] » qui se remplit par la fonction append() pour garder les valeurs. La variable « compteur_bouc » correspond au compteur des instructions à l'intérieur de boucle. La variable « compt » est un compteur d'instructions et est incriminée à chaque lecture de l'instruction par l'interpréteur.
La position de la tête est représenté par la variable tete_pos qui est initialisée à 19 ce qui correspond à l'index de la liste qui représente la bande : bande=[x= « 0 » for e in 50]. La tête de lecture est représentée par une valeur « X » dans une liste avec les espaces d'un nombre correspondant à la longueur de la bande : tete=[x for e in range(0,len(bande))]. Les deux listes sont imprimées toujours ensemble et dans la version « « .join() pour faciliter la visibilité de la bande. Pour le travail à l'intérieur de l'interprétateur les listes gardent leur type d'origine et ne sont jamais converties en chaîne de caractères.
Le programme prend en entrée les fichiers.txt avec le code en langage machine décrite ci-dessus et les chiffres initialisés par l'utilisateur qui doivent comporter un espace entre eux.
Les instructions dans le fichiers sont converties en liste et stockées dans la variable read_data.
Les chiffres sont initialisées sur la bande :

```
for i in range(19,20+int(chiffres[0])):
        bande[i]="1"

for c in range(22+int(chiffres[0]),22+int(chiffres[0])+int(chiffres[1])):
        bande[c]="1"
```

Le bloque d'instructions suivant permet d'implémenter les instructions : G, D, 1, 0, I. Chaque instruction est comptée.

read_data est une liste des instructions pour chaque instruction il y a une exécution immédiate lors de l'itération sur la liste. Chaque commande change la position de la tête de lecture.

```
for d in read_data:
        compt+=1
        if d!="boucle":
                if d=="I":
                        print("".join(bande))
                        print("".join(tete))
                if d=="G":
                        tete[tete_pos]=" "
                        tete_pos=tete_pos-1
                        tete[tete_pos]="X"
                if d=="D":
                        tete[tete_pos]=" "
                        tete_pos=tete_pos+1
                        tete[tete_pos]="X"

                if d=="1":
                        bande[tete_pos]="1"
                        tete[tete_pos]=" "
                        tete_pos=tete_pos+1
                        tete[tete_pos]="X"

                if d=="0":
```

```
bande[tete_pos]="0"
tete[tete_pos]=" "
tete_pos=tete_pos+1
tete[tete_pos]="X"
```

Les commandes représentées ci-dessus sont faciles à exécuter et implémenter et ne présentent pas de problèmes particulières.

**Difficultés :**

Les boucles représentent des difficultés particulières et bien que l'implémentation des boucle ne représente pas de mystère au niveau théorique, elle est difficile à implémentée en Python. Bien que cette fonctionnalité ait été prévue dans notre langage, elle n'a pas pu être implémentée complètement à cause du manque de temps pour finaliser le projet et pour bien tester le programme et enlever les erreurs.

**Argumentation :**

Nous sommes parti du principe que le langage doit être concis et simple avec une structure claire. C'est pour cela que nous avons pris la décision de séparer chaque instruction par un saut de ligne ce qui facilite l'interprétation et le parsing du code.

La machine contient un fichier instructions.txt avec l'exemple de code en langage machine, mais elle n'est pas capable de faire les calculs vu l'absence des boucles. Le langage fait bouger la tête de lecture à droite et écrit les « 1 » en imprimant le résultat à chaque étape.

**Les boucles :**

Il est nécessaire d'avoir :

- un compteur d'instructions
- une pile

Nous avons implémenté ces éléments dans notre code, mais ils ne sont pas fonctionnels pour le moment.
Le pas suivant est de séparer la liste composant les instructions en slice qui [boucle : fin]. Cette liste comportera toutes les instructions elle sera séparée par les espaces avec la fonction split(). Chaque instruction va suivre la même procédure que les autres instructions à l'extérieur de la boucle ( si 1 alors on écrit 1 etc.), mais leur exécution va se trouver dans la boucle python while dont la condition d'arrêt sera la valeur qui se trouve avant « fin ».

**Preuve de Turing completeness**

Soit la machine de Turing M qui peut être représentée de manière suivante sur le langage M={1,0}:

M=<T, Σ, Γ, B, Q, Δ(δ) , δn...>

B est la bande .
T est la tête de lecture.
Γ la pile pour enregister les adresses de la boucle
Q est le compteur d'instructions.
Σ est la somme des états.
Δ est l'ensemble des instructions où Δ(δ) est une instruction des boucles.

Notre machine de Turing peut être représentée de manière suivante :

M=<T, Σ, Γ, B, Q, δn...>

L'absence des instructions de boucle rend notre machine incapable d'avoir le même pouvoir expressif pour le rendre équivalent à la machine de Turing. Donc nous n'avons pas tous les éléments nécessaires à la simulation d'une machine de Turing universelle ce qui fait que notre machine n'est pas Turing complète.

# Bibliographie


1. Bonnay D. "La philosophie des mathématiques". 2011. hal-00617305

2. Sipser, M. "Introduction to the Theory of Computation". volume 27. Thomson Course Technology Boston, MA. 2006.

3. Turing A.  "Computability and λ-definability", *J. Symbolic Logic*, vol. 2, 1937, p. 153-163

4. Wolper P. "Introduction à la calculabilité", Collection: Sciences Sup, Dunod, 2006 - 3ème édition - 240 pages