

Architecture des ordinateurs

IR1 Informatique 2010-2011

TP 4

Objectif de cette séance

Le but de cette séance est de réaliser un programme assembleur implantant le crible d'Ératosthène.

Cette fiche est à faire en binôme. Il faudra :

1. réaliser un — bref — rapport à rendre **au format pdf** contenant les réponses aux questions de cette fiche ;
2. réaliser une archive **au format zip** contenant les sources des programmes ;
3. envoyer le tout à l'adresse de votre chargé de TP

`Samuele.Giraud@univ-mlv.fr` ou `vcarnino@gmail.com`

par un courriel dont le sujet est sous la forme

IR1 Archi TP4 *NOM1* ; *NOM2*

Tous les fichiers nécessaires pour ce TP se trouvent à l'adresse

<http://igm.univ-mlv.fr/~giraud/Enseignements/IR1/Archi/Archi.html>.

1 Algorithme du crible d'Ératosthène

L'algorithme du *crible d'Ératosthène* permet de calculer tous les nombres premiers inférieurs à nombre N fixé. Il procède ainsi :

1. on commence par générer une liste d'entiers non soulignés $L := [2, 3, 4, \dots, N]$;
2. ensuite, tant qu'il existe des nombres non soulignés dans la liste L :
 - (a) on souligne le plus petit nombre k de L non encore souligné ;
 - (b) pour tout nombre $\ell > k$ de la liste L :
 - si k est un diviseur de ℓ , alors on supprime l'élément ℓ de la liste ;
3. la liste L – qui contient maintenant exclusivement des nombres soulignés – est la liste des nombres premiers inférieurs à N .

Par exemple, si on applique cet algorithme pour $N := 30$, on obtient successivement les valeurs de L suivantes :

```
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30],
[2,3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 25, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ],
[2,3, 5, 7, 11, 13, 17, 19, 23, 29 ].
```

2 Implantation

On peut se servir de la bibliothèque `asm_io.inc` pour gérer les entrées/sorties. Le programme doit définir une constante `N` de valeur 128. Ceci est réalisé par l'instruction :

```
1 %define N 128
```

Le programme doit également réserver `N-1` octets à partir d'une adresse `tab` et un octet à l'adresse `pgs`. Ceci est réalisé par les instructions :

```
1 tab resb N-1
2 pgs resb 1
```

Cette suite de `N` octets peut être vue comme un tableau. Celui-ci est utilisé pour mémoriser les nombres effacés pendant l'exécution de l'algorithme. L'octet à l'adresse `tab + i - 2` vaut 1 si le nombre `i` n'est pas effacé et 0 s'il est effacé. La variable `pgs` contient la valeur du plus grand nombre souligné (c'est à dire le dernier nombre souligné).

Important 1. Toutes les fonctions implantées par la suite doivent respecter la convention d'appel du C.

Question 1. Écrire une fonction `effacer` qui prend en argument un nombre `i` et positionne la case correspondante du tableau `tab` à 0, signifiant que le nombre `i` est effacé.

Question 2. Écrire une fonction `marquer` qui prend en argument un nombre `i` et positionne la case correspondante du tableau `tab` à 1, signifiant que le nombre `i` n'est pas effacé.

Question 3. Écrire une fonction `lire` qui prend en argument un nombre `i` et retourne dans `eax` la valeur de la case correspondante dans le tableau `tab`.

Important 2. Dorénavant, on utilisera *uniquement* les trois fonctions précédentes pour accéder au tableau `tab`.

Question 4. Écrire une fonction `initialiser` qui positionne tous les octets du tableau `tab` à 1. Cette fonction doit aussi positionner la variable `pgs` à la valeur 0 pour spécifier qu’aucun nombre du tableau n’est souligné.

Question 5. Écrire une fonction `effacer_multiples` qui prend un argument `p` et qui positionne à 0 les cases du tableau `tab` correspondant aux entiers qui sont multiples de `p`.

Question 6. Écrire une fonction `premier_suivant` qui prend un argument `i` et renvoie dans `eax` la valeur du premier nombre supérieur à `i` tel que l’octet correspondant dans le tableau `tab` vaut 1. Si une telle valeur n’existe pas, la fonction renvoie `N + 1`.

Question 7. Écrire une fonction `afficher` qui affiche dans l’ordre croissant les nombres dont la valeur dans le tableau `tab` est 1.

Question 8. Écrire un programme qui implante l’algorithme d’Ératosthène et utilisant les fonctions précédentes. Le programme doit afficher les nombres premiers inférieurs à `N`.

Question 9. Modifier le programme afin qu’il sorte de la boucle principale (instruction 2.) lorsque le plus grand nombre souligné dépasse \sqrt{N} .

Astuce 1. Il n’est pas nécessaire de savoir calculer la racine carrée de `N`.

Bonus 1. Améliorer le programme pour qu’il utilise de manière plus efficace la mémoire. On utilise le fait qu’un octet permet de conserver l’état – effacé ou non effacé – de huit nombres et non plus d’un seul. Si le programme est déjà bien écrit, il ne suffit de modifier que les trois premières fonctions.

Bonus 2. Améliorer le programme pour faire en sorte qu’il fonctionne pour des valeurs de `N` non fixées. L’utilisateur du programme saisit sur l’entrée standard une valeur pour `N` et le programme doit afficher tous les nombres premiers inférieurs à `N`.