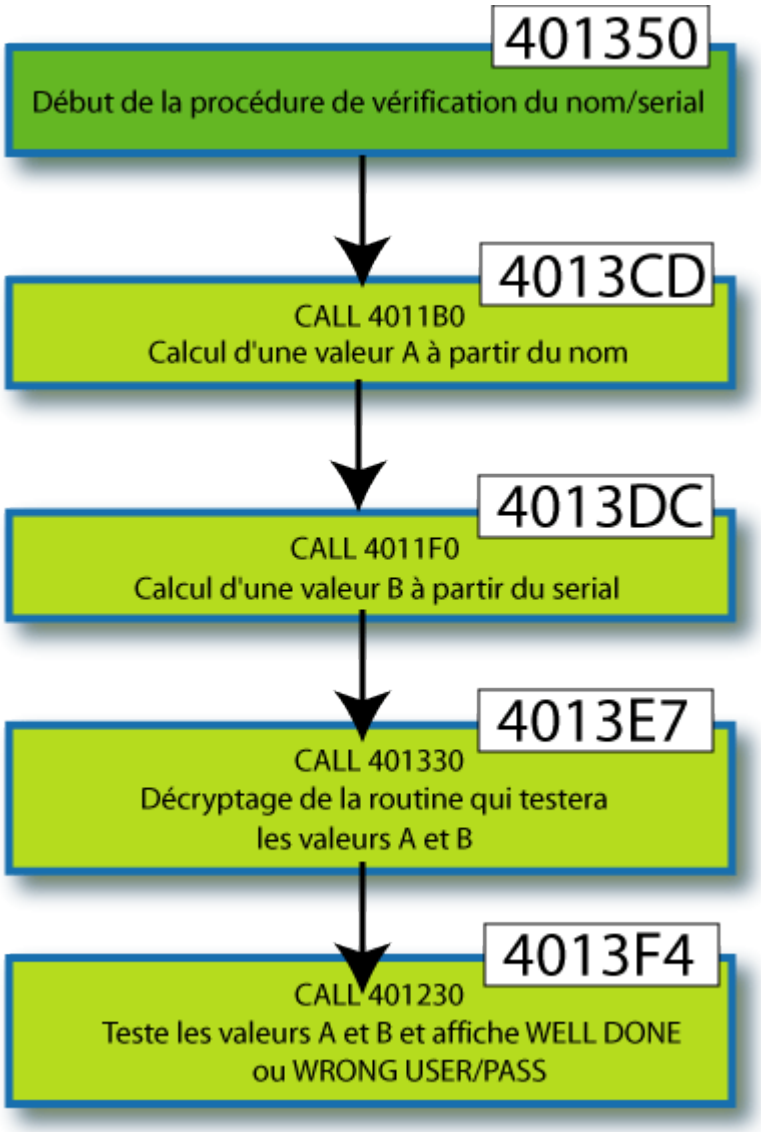
CRACKME 5 de _JULIEN_

1 . Analyse du programme

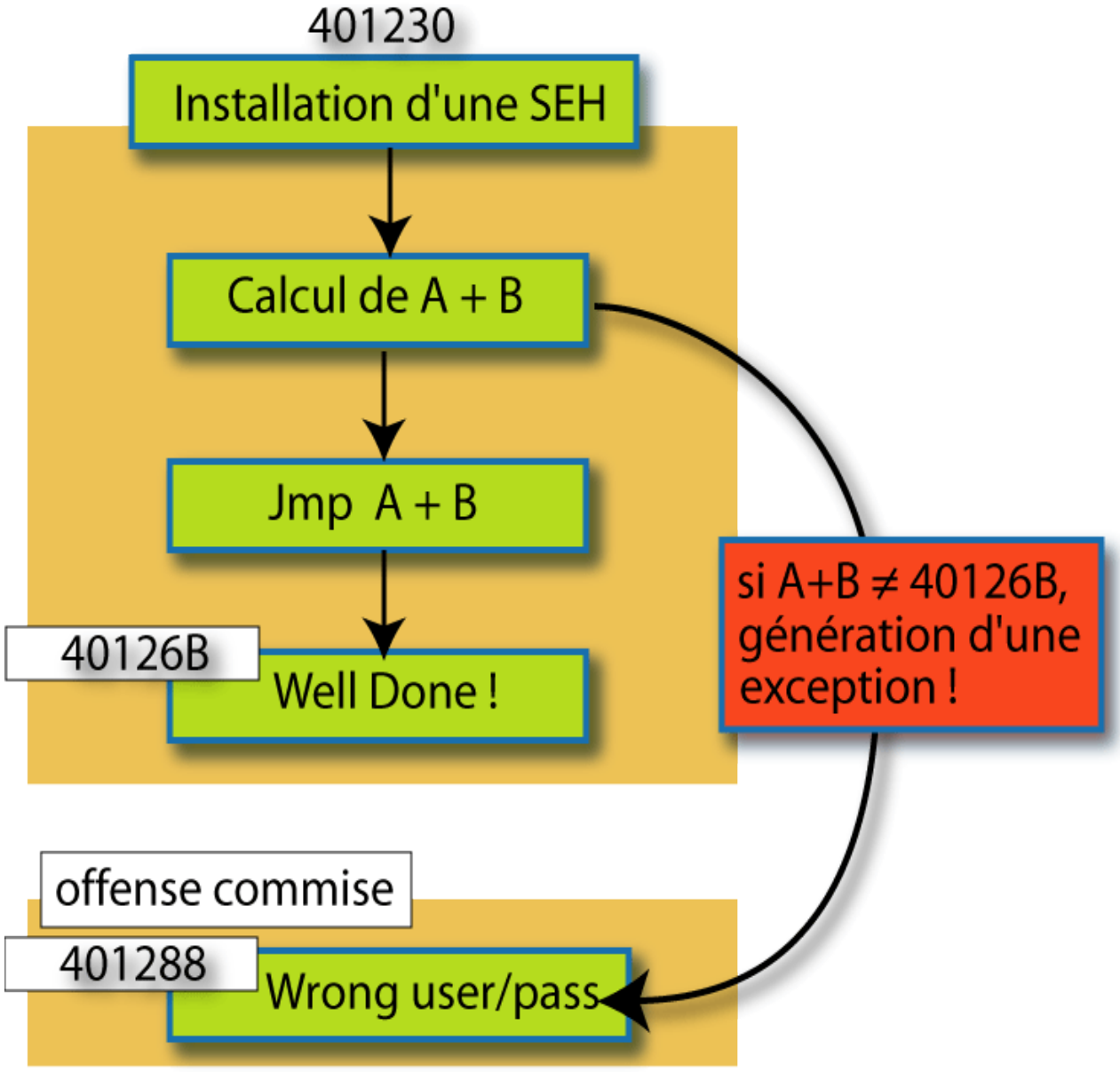
On entre tout de suite dans le vif du sujet. La procédure de traitement du nom et du serial se trouve en 401350. Voici un schéma pour simplifier l'explication de cette procédure :



Nous verrons plus loin comment sont calculés ces nombres A et B. Regardons tout de suite ce qui se passe en 401230, le dernier Call :

1) ça commence par l'installation d'une SEH qui va s'occuper de vous envoyer sur Wrong user/pass si vous ne saisissez pas le bon nom/serial.
2) Puis, on calcule A+B pour finir par un Jmp A+B....si les valeurs sont les bonnes, A+B = 40126B où se trouve le message Well Done !. Si A+B a une autre valeur, le saut va

provoquer une exception qui va faire fonctionner la SEH



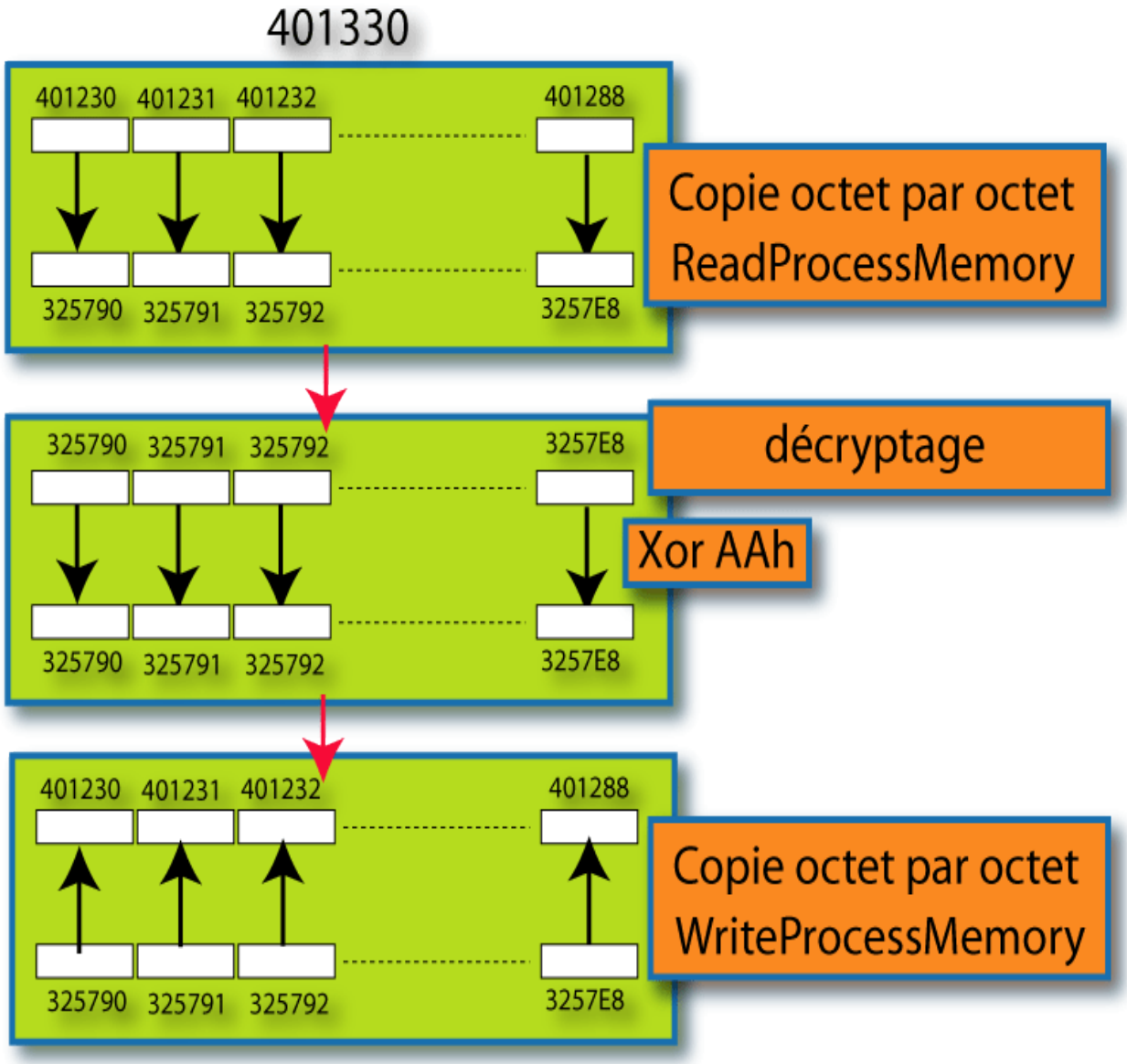
Remontons un peu et voyons maintenant la procédure de décryptage qui se charge de décrypter la procédure que nous venons de voir. J'en profite au passage pour dire que les strings du type Well Done n'apparaissent pas au début du programme à cause de cryptage. Voici le procédé de décryptage :

1) On copie dans un buffer le contenu de la portion de code à décrypter.

2) On « xor » chaque octet par Aah dans le buffer.

3) On recopie dans l'emplacement d'origine les octets xorés.

C'est le cryptage/décryptage de base, il ne présente aucune difficulté particulière.



Nous allons à ce stade pouvoir répondre à la première étape du défi : Patcher le crackme pour que n'importe quel nom/serial fasse l'affaire.

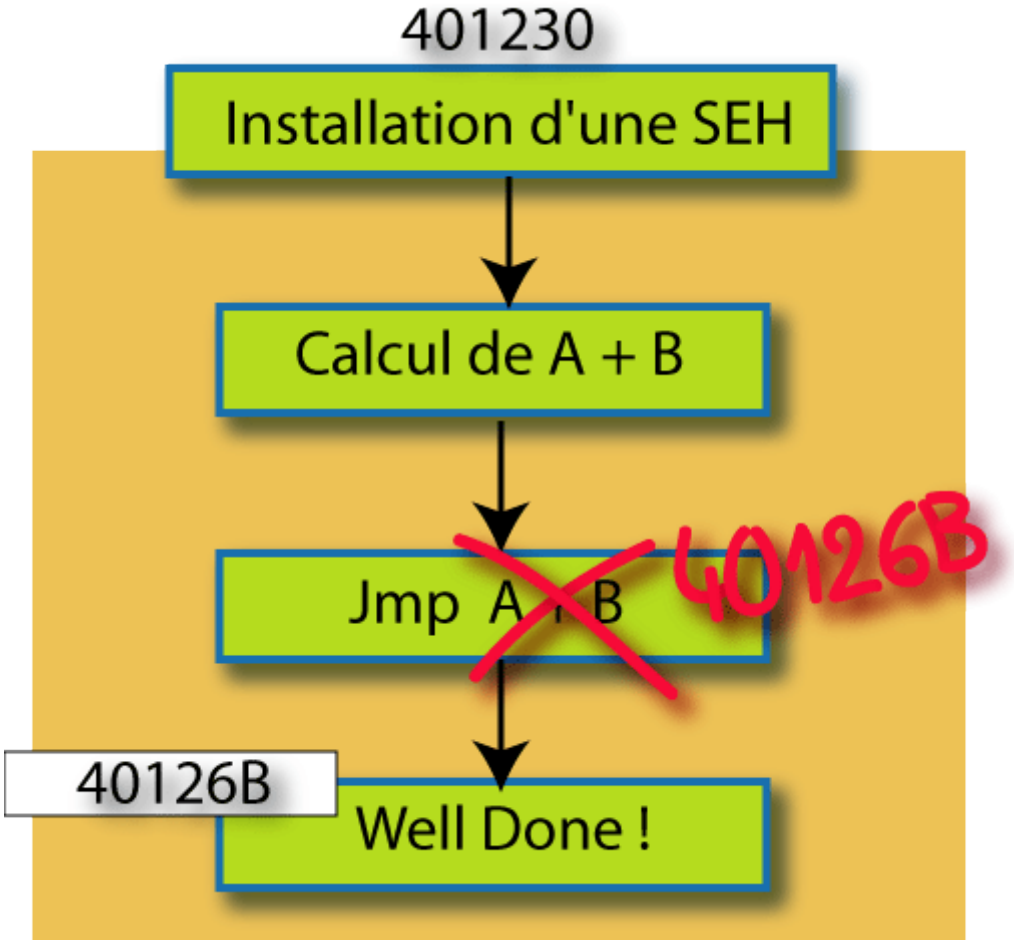
2 . Patcher le crackme.

Voilà ce que nous allons faire : obliger le programme à sauter en 40126B. Dans le programme, le Jmp A+B se présente comme ceci :

```
401268        . FF65 E4              JMP DWORD PTR SS:[EBP-1C]
```

Nous allons le modifier en ceci :

```
401268        . EB01                 JMP 40126B
40126B        . 90                   NOP
```



Pour le patcher définitivement, il faut savoir quelles sont les valeurs de nos trois octets avant le cryptage. C'est-à-dire :

?? XOR AAh = EBh
?? XOR AAh = 01h
?? XOR AAh = 90h

Il faut donc inverser les XOR. L'opérateur inverse de XOR est .... XOR lui-même ! Donc, les valeurs précédentes sont :

EBh XOR AAh = 41h
01h XOR AAh = ABh
90h XOR AAh = 3Ah

Finalement, en 401268, on a 41h, en 401269, on a AB et en 40126A, ona 3A.

3 . Trouver un nom/serial valide.

Deuxième étape du défi : trouver un nom et un serial valide. Il faut donc décortiquer les deux procédures du calcul de A et de B.

CALCUL de A à partir du nom :

Je ne rentre pas dans les détails de la procédure, ce n'est pas très intéressant. Ce qui importe, c'est le résultat. Prenons au hasard BeatriX comme nom. Nous obtenons :

A = 3 x (B) x (e) x (a) x (t) x (r) x (i) x (X) **(1)**

(B) signifie valeur ascii de B.

CALCUL de B à partir du serial :

Le calcul de B se fait de la façon suivante. Supposons que le serial soit abc,

B = FFDEADF1h / (a x b x c) **(2)**

En réalité, dans la procédure de calcul, on divise FFDEADF1h successivement par a, b et c.

LA SOMME :

Dernière condition :

A + B = 40126Bh **(3)**

BON ! Maintenant, on attaque dans quel sens, hein ? Je vous propose d'attaquer le plus embêtant, le calcul de B ! Pour calculer B, il nous faut le serial !!

Exemple pour comprendre : Je prends comme serial ***abc***. (99h 98h 97h) Appliquons les 3 divisions successives :

1) FFDEADFh / 99 = 1AC1F41 (le reste est abandonné)
2) 1AC1F41 / 98 = 2D10C
3) 2D10C / 97 = 4C6.

Donc B = 4C6h. Avec la condition (3), on obtient : A = 40126Bh – 4C6h = 4000DA5h.
Il faut maintenant vérifier que A est factorisable par des coefficients inférieurs à 7Dh (ascii). Il faut qu'il soit également divisible par 3....or dans notre cas, il ne l'est pas.

donc le serial *abc* ne donnera jamais de nom valide !

Deuxième exemple : Je prends comme nom abc.

A = 3 x 99 x 98 x 97 = A0C038h.

B = 40126B – A0C038 = FF9F5233h.

FFDEADF1 / ???? = FF9F5233 ? Cette équation n'a pas de solution ! Il suffit de diviser par 2 pour s'en convaincre.

donc le nom *abc* ne donnera jamais de serial valide !

Il faut donc Bruteforcer le calcul, c'est-à-dire utiliser une méthode « exhaustive » pour tester tous les serials et vérifier si à chaque fois on peut obtenir un nom valide.

Nous allons donc programmer une petite procédure à cet effet. Pour ce faire, nous allons procéder ainsi :

1) Choisir un serial. Il suffit de boucler en incrémentant les valeurs des caractères qui composent le serial. On se fixera arbitrairement pour ce cas un serial de 4 caractères.

2) Calculer B. Toujours très simple, il faut juste manipuler les QWORD.

3) Calculer A. **Attention à la subtilité** !!! On sait que :

A + B = 40126Bh (3)

Donc, me direz-vous, pour connaître A il suffit de faire 40126Bh – B....et bien pasforcément !!!! L'adresse 40126Bh est contenue dans un DWORD, mais la somme A+B ne l'ai pas forcément.

Si A + B = 10040126Bh alors en DWORD, ça donne aussi 40126Bh
Si A +B = 20040126Bh idem
Si A + B = 345120040126Bh idem etc.....

DONC, lorsqu'on s'est donné B, il existe de très nombreuses possibilités pour A, très exactement 2 milliards 147 millions 483 mille 647 !

On espère quand même que sur ces 2 milliards de possibilités, il y en aura une qui se factorisera convenablement ! Pour faire simple, nous fixerons arbitrairement la somme à 10040126Bh.

4) Factoriser A et vérifier qu'il est divisible par 3 et que ses autres facteurs sont compris entre 20h et 7Dh. Simples divisions euclidiennes !

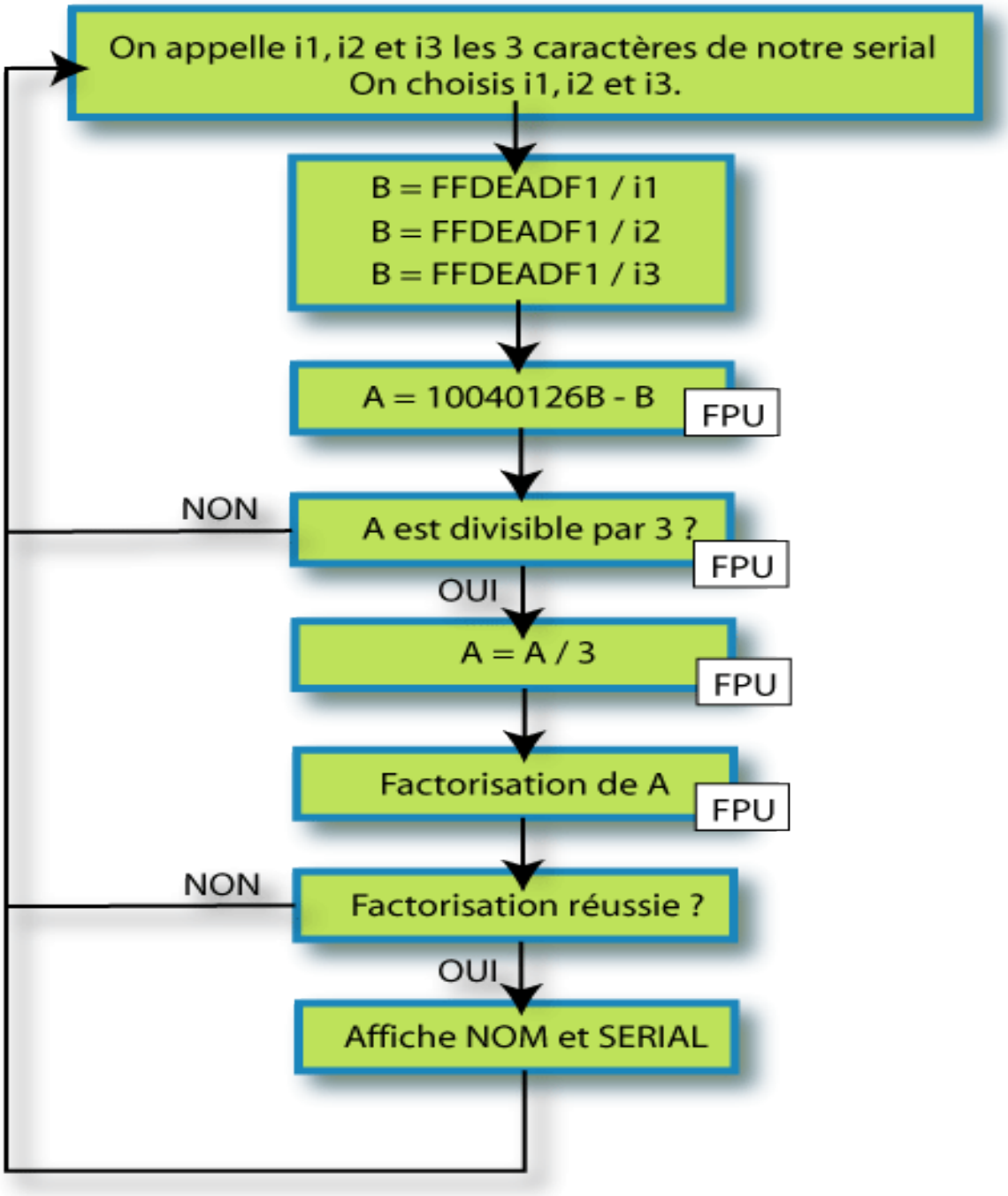
5) Retour à l'étape 1.

Vous pouvez voir l'algorithme simplifié sur le schéma ci-dessous. J'utilise la FPU (coprocesseur arithmétique) pour éviter de faire de la gymnastique avec les QWORD. C'est très pratique et très facile d'utilisation.

Exemple de NOM/SERIAL valide :

NOM : +Maa.
SERIAL : i/#

Evidemment, on peut modifier la valeur 10040126Bh pour obtenir d'autres résultats. On peut aussi changer le nombre de caractères du serial.



BeatriX - Août 2004.