Initiation à l'Assembleur (2)

par André Chéramy et tous ceux qui voudront bien y participer...

Si vous n'avez pas eu le temps de potasser la première partie de cette série, il est temps de vous y mettre, car le matériel ne va pas tarder à s'accumuler. Quel est le programme du jour ? Plutôt que de repartir sur un autre exemple, nous allons tourner encore un peu autour de l'affichage de 'Salut les gars !'. En effet, nous avons vu comment organiser le texte du message par rapport au code du programme et comment marquer la fin de la chaîne de caractères à afficher. Nous avons eu un petit aperçu de ce qui peut se passer si les instructions ne sont pas dans le bon ordre. En fait, c'est en déboguant qu'on apprend le mieux, car trouver la raison d'un disfonctionnement, c'est comprendre encore mieux comment ça marche.

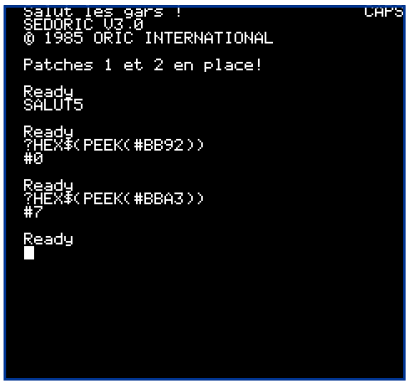
Variante 5

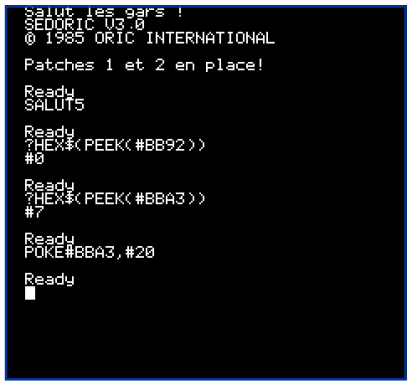
Allez, triturons encore ce pauvre bout de code ! Voyons un peu ce qui se passe lorsqu'on déplace l'instruction INY. Chouette, on peut encore gagner une instruction de branchement (2 octets) :

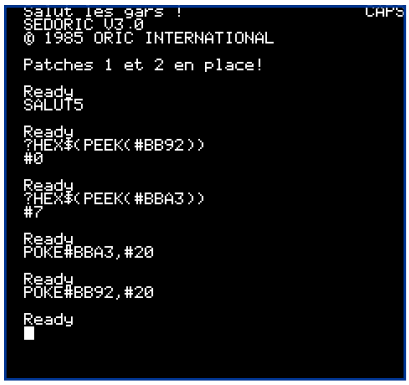
```
9801 53 61 6C 75 74 20 6C 65 73 20 67 61 72 73 20 21 00 Salut les gars !
9812 A0 FF    LDY #FF     Y=#FF
9814 C8       INY         Y=Y+1
9815 B9 01 98 LDA 9801,Y  lit l'octet présent à l'adresse 9801+Y
9818 99 82 BB STA BB82,Y  et le copie en BB82+Y
981B D0 F7    BNE 9814    si cet octet n'était pas #00, reboucle en 9814
981D 60       RTS         sinon, c'est fini !
```

On sauve ce petit programme avec SAVE «SALUT5»,A#9801,E#981D,T#9812.

Il peut sembler curieux d'initialiser Y=#FF, puis à la ligne suivante de l'incrémenter avec Y=Y+1. Vous ne savez pas ce que vous voulez ou quoi ? Et pourtant, c'est une pratique des plus courantes. En effet, après affichage du caractère (STA BB82,Y), on reboucle en 9814 pour incrémenter l'index et lire le caractère suivant. Remarquez que le premier caractère doit être lu en 9801 (avec Y=0) et que la boucle commence justement par un INY. Il est donc nécessaire d'initialiser correctement Y avant



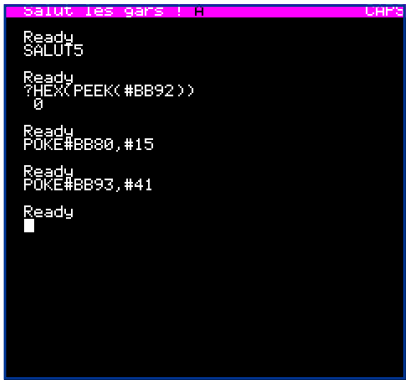


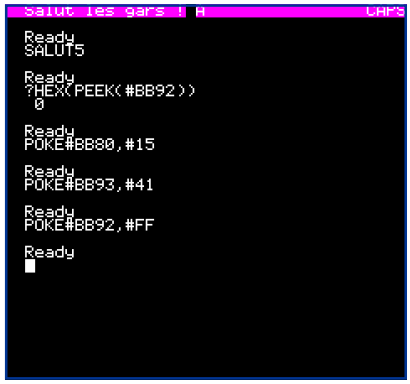


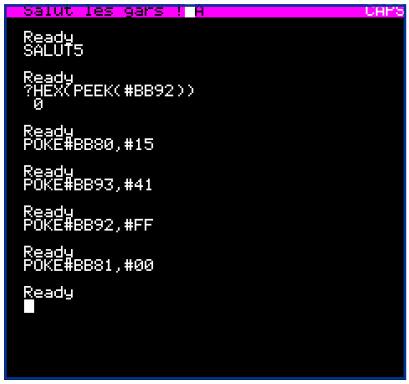
d'aborder la boucle. Note : Quand on incrémente un registre 8 bits, on passe de #FF à #00, ce qui explique le code un peu surprenant de cette 5e variante.

Le commentaire de la ligne 981B devrait vous alarmer. En clair on y explique que lorsque l'octet lu était le #00, on l'a copié sur la ligne service avant même de tester s'il fallait le faire. Résultat, le #00 (attribut encre noire) se retrouve sur la ligne service à la suite du message 'Salut les gars !' (en BB92), ce qui pourrait bien être fâcheux. Dans ce cas précis, la présence de cet octet parasite peut rester invisible, parce que d'une part la couleur du papier est noire (on a donc encre noire sur papier noir) et que d'autre part, il y a plus loin sur la ligne un attribut d'encre blanche (#07) juste devant le 'C' de 'CAPS', c'est à dire en BBA3.

La bogue peut donc rester inaperçue, jusqu'à ce que vous réutilisiez cette petite routine dans un autre contexte... Comme quoi, on n'a pas toujours raison de vouloir gagner 2 petits octets !



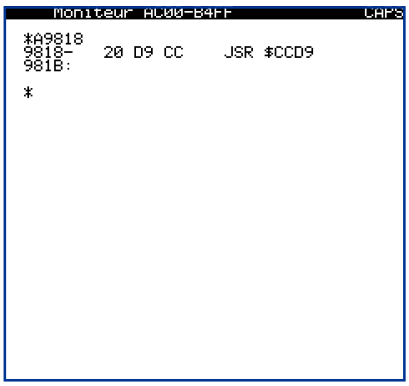


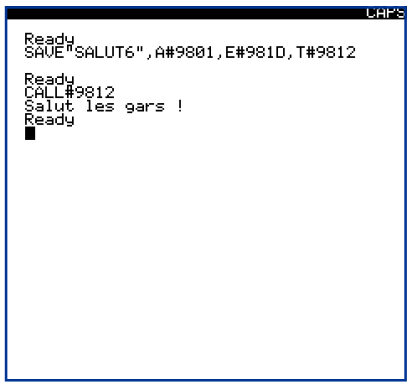


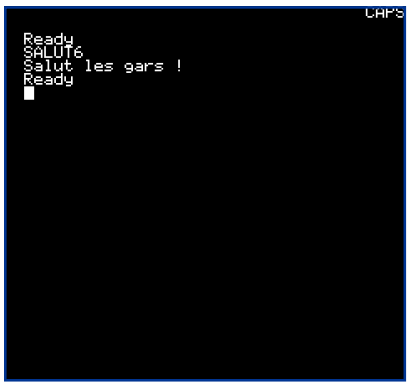
Il est facile de mettre en évidence la présence du #00 sur la ligne service. Le plus simple est de POKEr un espace sur l'octet encre blanche devant le 'CAPS', en BBA3. Ce qui entraîne la disparition de ce message. On peut ensuite poker un autre espace sur l'octet #00 en cause en BB92 et hop le message réapparaît ! Note : Un attribut d'encre est actif sur le reste de la ligne d'écran ou jusqu'à ce qu'un autre attribut d'encre soit rencontré.

On peut illustrer ce phénomène en changeant les couleurs par défaut de l'écran. Ici, il s'agit de papier noir (#10) et encre blanche (#07). Après un CLS, on relance SALUT5, on vérifie que le #00 parasite est bien en BB92, avec un ?HEX$(PEEK(#BB92)), on change la couleur papier de toute la ligne avec un POKE#BB80,#15 (couleur magenta, ainsi on pourra voir l'encre noire et l'encre blanche) et enfin on place une lettre 'A' juste après le #00 avec un POKE#BB93,#41. La recopie d'écran de gauche montre l'apparition d'un 'A' noir. Si on remplace le #00 par l'Ascii #FF (un carré), on voit apparaître ce carré. Tiens, il n'est pas de la couleur de l'encre (c'est à dire blanc), mais noir ! C'est parce que je vous ai un peu trompé : Le code Ascii du carré noir, utilisé pour le curseur texte, n'est pas #FF, mais #7F (127 en décimal). L'octet #7F s'écrit **0**111 1111 en binaire. Vous voyez que son bit 7 est à zéro. Pour passer en vidéo inverse, il suffit de mettre à un ce bit 7, ce qui donne **1**111 1111, soit #FF. En vidéo inverse, le carré blanc apparaît en noir ! Allez, on joue encore un peu, en initialisant une couleur d'encre noire pour toute la ligne service avec un POKE#BB81,#00, ce qui modifie bien la couleur du message 'Salut les gars !' qui passe en noir et celle du carré (toujours en vidéo inverse) qui passe en blanc. Et le CAPS reste imperturbablement en blanc, à cause de l'attribut encre blanche toujours présent en BBA3.

VARIANTE 6

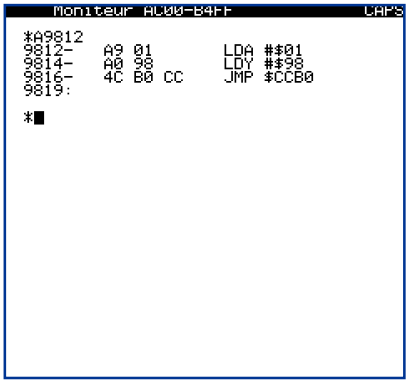


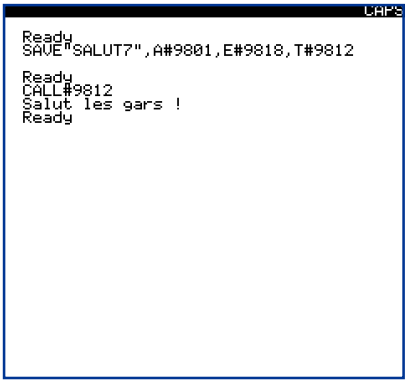


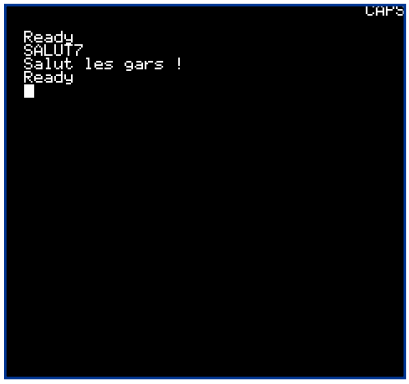


Bon, voilà bientôt 8 pages occupées avec l'affichage du message 'Salut les gars !'. Et si nous voulions l'envoyer, non pas à une adresse prédéfinie, mais tout simplement au curseur, comme lorsqu'on fait un PRINT ? Il faudrait alors gérer le déplacement du curseur et la mise à jour de plusieurs variables, ce qui allongera considérablement notre exemple. Dans la Rom, il y a une routine de base qui fait cela très bien. Elle est située en CCD9 et il suffit de mettre le caractère à afficher au curseur dans le registre A avant de l'appeler avec un JSR CCD9. Voilà ce que ça donne :

```
9801 53 61 6C 75 74 20 6C 65 73 20 67 61 72 73 20 21 00 Salut les gars !
9812 A0 FF    LDY #FF
9814 C8       INY
9815 B9 01 98 LDA 9801,Y
9818 20 D9 CC JSR CCD9
981B D0 F7    BNE 9814
981D 60       RTS
```







Pour utiliser une routine de la Rom, il suffit de connaître ce qu'on doit mettre en entrée et ce qu'on récupère en sortie. 'L'Oric à nu' de Fabrice Broche nous dit : En entrée, A contient le caractère à afficher. En sortie, le caractère est affiché sur l'écran ou sur l'imprimante selon [le drapeau] #2F1; A, X, Y sont inchangés; Z et N sont positionné selon A. Lorsqu'on utilise une routine (que ce soit de la Rom ou d'ailleurs), il est important de connaître aussi les modifications annexes que cette routine va apporter aux registres du 6502, sinon cela peut perturber la suite du programme principal. Ici, par paresse, j'ai gardé la trame de la variante 5, que j'ai juste un peu modifiée (remplacement du STA BB82,Y par JSR CCD9). J'ai donc conservé la bogue du #00, en me disant que pour notre essai, ce ne serait pas grave. Mais j'ai eu de la chance, car le test de fin de message, qui se fait sur le drapeau N, se fait non pas après le LDA 9801,Y mais après l'appel à la routine de la Rom. Or cette routine revient avec N positionné selon A et donc ça marche ! Ouf ! Il n'en reste pas moins que la structure de la variante 6 est doublement boguée, puisque à la bogue de l'affichage du #00, j'ai ajouté une bogue potentielle sur le drapeau N. Supposons que plus tard, nous utilisions une autre routine à la place de CCD9. Il se pourrait alors qu'un problème apparaisse. Je ne vous conseille donc pas ce genre de fantaisie. Il aurait mieux valu repartir de la variante 1, qui était plus propre !

VARIANTE 7 (c'est la dernière, promis !)

C'est bien beau d'avoir utilisé ce JSR CCD9 à la place du STA BB82,Y mais il y avait beaucoup mieux dans la Rom. La routine CCB0 permet d'afficher au curseur une chaîne de caractères pointée par AY et terminée par #00.

```
9801 53 61 6C 75 74 20 6C 65 73 20 67 61 72 73 20 21 00 Salut les gars !
9812 A9 01    LDA #01     les registres AY pointent sur l'adresse
9814 A0 98    LDY #98     9801 de la chaîne (octet de poids faible devant)
9816 4C B0 CC JMP CCB0    affiche la chaîne terminée par zéro au curseur
```

On sauve cette 7e variante avec SAVE «SALUT7»,A#9801,E#9818,T#9812 La recopie d'écran montre ce que son exécution donne. Notre exemple est localement devenu très court, mais en fait son temps d'exécution a augmenté, car la routine appelée en Rom est assez complexe et fait elle-même appel à plusieurs autres routines.

à suivre...