

Ram Viewer *Hp 49*

Visitez la mémoire de votre machine de manière graphique grâce à un petit navigateur...

Tous les viewers de mémoires conçus pour les calculatrices HP permettent essentiellement de visualiser l'adresse, la mémoire sous forme ASCII et la transcription sous forme hexadécimale. Mais comment repérer des objets graphiques avec ce type de logiciel ? Impossible. Réparons cette lacune avec un petit viewer qui permet de survoler la mémoire en l'affichant sous forme graphique. Et une fois n'est pas coutume, nous utiliserons l'assembleur...

• mise en forme •

Tout d'abord, il faut savoir que les adresses vont de # 00000h à # FFFFFh, soient 1048576 quartets à balayer ou 512Ko : c'est la Ram. Ensuite, beaucoup de grobs ont une largeur de 131 pixels (ou 34 quartets) aussi bien au niveau de l'écran courant qu'au niveau des menus placés juste au dessus des touches de fonctions. Donc, nous afficherons la mémoire sur toute la largeur de l'écran (c'est à dire sur 34 quartets), mais aussi sur les cinquante premières lignes. Le reste de l'écran sera consacré à l'affichage de l'adresse courante, mais aussi au menu qui sera le tableau de bord de navigation. Schématisons un peu l'écran final :



• affichage de l'adresse courante •

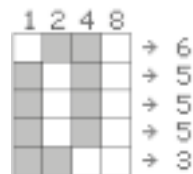
Pour afficher un texte, aussi simple soit-il, il faut une table de caractères sur laquelle s'appuyer pour en extraire les caractères dont on a besoin. Deux solutions s'offrent à nous : soit utiliser une table de caractères interne à la machine, soit en recréer une pour l'occasion. Nous choisirons la deuxième solution afin de savoir comment construire une table réduite à seize caractères, cas qui se généralise de manière fort simple à une table complète, qui est en fait une table ASCII. En assembleur, tout est à concevoir, même le pauvre petit caractère qu'il faut afficher à l'écran, outre l'utilisation de routines bien sûr. Tout d'abord, définissons les caractères qui feront partie de notre mini-table : ce sont les caractères qui font partie d'une chaîne hexadécimale (en base 16), c'est à dire 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F. Sur la Hp 49G, nous créerons des caractères de 3x5 pixels et ils seront donc définis par un grob de 64x5 pixels (ou 5 lignes de 16 quartets) dont voici la représentation telle qu'elle est affichée à l'écran :

0123456789ABCDEF

Voici chaque ligne sous sa forme hexadécimale :

6277576763236377
5344511455551511
5262773226731533
5214445154551511
3773436133536371

Chaque caractère est composé de cinq quartets, comme cet exemple de codage du premier caractère qui représente le chiffre zéro :



Pour le premier quartet 2+4=6, pour le second quartet 1+4=5, etc. On en fait de même pour les autres caractères, et c'est ainsi qu'on construit la table.

Un autre moyen beaucoup plus simple est de créer un petit programme (facile à comprendre) qui construit notre table tout seul :

```
« "0123456789ABCDEF" 1 ``GROB
  { # 0h # 0h } { # 3Fh # 4h } SUB
  ``H 21 100 SUB »
```

Il reste ensuite à insérer la chaîne hexadécimale obtenue dans notre source et le tour est joué ! Cette chaîne insérée dans la source ressemble à :

```
GOSUB GHEX %on saute la table de caractères
$6277576763236377
$5344511455551511
$5262773226731533
$5214445154551511
$3773436133536371
*GHEX %fin du saut
C=RSTK %on récupère l'adresse du début de la table
```

Le fait de réaliser un GOSUB stocke automatiquement l'adresse du quartet qui suit dans la pile des retours qu'on appelle RSTK (Return STack). En fin de saut, il suffit de récupérer cette adresse pour pouvoir accéder aux données graphiques de notre table de caractères.

• affichage de la barre des menus •

Comme une adresse est constituée de cinq quartets, il serait préférable d'agir sur chacun d'eux. C'est à dire que nous allons avoir la possibilité d'incrémenter ou décrémenter directement le quartet qui nous intéresse, ceci grâce à cinq touches. De plus, pour choisir le sens de défilement de l'adresse, il nous faut une sixième touche. On en déduit donc facilement la forme de la barre du menu :

00001|00010|00100|01000|10000| +

Comme pour la construction de la table de caractères, on considérera cette barre de menu comme un grob qu'on insérera dans notre source, un peu comme ceci :

```
GOSUB MTXT %on saute le graphique
$0000000000. . . %le grob (un peu coupé... !)
*MTXT %fin du saut
C=RSTK %on récupère l'adresse de début du grob
```

• sens de navigation •

Le sens de navigation permet de faire défiler l'adresse dans le sens positif ou négatif, ceci grâce à une seule touche : [F6]. De ce fait le signe du dernier icône bascule entre + et - :



Le sens de défilement est géré par un seul flag (drapeau in french), qui incrémente l'adresse s'il est à zéro et qui la décrémente s'il est à un. C'est le flag ST5.

• la source •

Voici la source complète entièrement commentée et 100% assembleur. Pour l'assembler, utilisez la commande ASM de la library de développement.

```
"SAVE I NTOFF2 %sauve registres et pointeurs, interdit les interruptions
A=0. A R4=A. A %adresse à zéro
SCREEN R3=A. A %adresse d'écran
D1=A LC 00770 ZEROMEM %efface l'écran
MENU C=0. A LC 84 A+C. A R2=A. A %adresse du menu

%affichage du menu
GOSUB MTXT %on saute le graphique
$0000000000000000000000000000000000
$FFFFFFDFFFF7FFFFFFDFFFF7FFFFFFDFFFF70
$3333BDCCCE433B33DCECC4B3333DFFFF70
$555595555655595556555955565559555DFFEF70
$5555B555D6555B555D6555B555B5555DF7CF70
$5555B555D6555B555D6555B555B5555DFFEF70
$999915666469919956466619999DFFFF70
$FFFFFFDFFFF7FFFFFFDFFFF7FFFFFFDFFFF70
*MTXT %fin du saut
C=RSTK D1=C LC 10 %on récupère l'adresse et on pointe
{ A=DAT1. W DATO=A. W DO+16 D1+16 C-1. B UPNC } %boucle
d'affichage du menu

*BOUCLE %début de boucle
GOSUB AFFI CHE %routine d'affichage
LC 020 OUT=C=I N A=0. A %appel des touches du menu
?CBI T=1. 0 -> { LA 00001 } %touche [F1]: +/- 1h
?CBI T=1. 1 -> { LA 00010 } %touche [F2]: +/- 10h
?CBI T=1. 2 -> { LA 00100 } %touche [F3]: +/- 100h
?CBI T=1. 3 -> { LA 01000 } %touche [F4]: +/- 1000h
?CBI T=1. 4 -> { LA 10000 } %touche [F5]: +/- 10000h
?CBI T=1. 5 -> { GOSUB PLUSMOI NS } %touche [F6]: sens + ou -
B=A. A A=R4. A %on récupère la valeur de l'adresse
?ST=0. 5 -> { A+B. A } %dans ce cas, on l'incréménte
?ST=1. 5 -> { A-B. A } %sinon on la décrémente
R4=A. A %et on sauve le résultat
LC 001 OUT=C=I N ?CBI T=1. 6 -> FI N %touche [<=]: pour quitter
LC 03000 { C-1. A UPNC } %petite pause
GOTO BOUCLE %et on reboucle

*FI N %fin du programme
I NTON2 GOSBVL FI ush LOADRPL %réautorise les interruptions, vide
le buffer clavier, restaure registres et pointeurs, et retourne au Rpl

*PLUSMOI NS %test du sens de navigation
?ST=0. 5 -> { ST=1. 5 RTN } % si c'est positif, il devient négatif
```

```
?ST=1. 5 -> { ST=0. 5 RTN } % et vice-versa
```

```
*AFFI CHE %affichage de l'ensemble
```

```
%modification du sens de comptage
```

```
A=R2. A DO=A LC 00044 A+C. A D1=A %on pointe la dernière icône
```

```
?ST=0. 5 -> { LC E } %positif : on choisit +
```

```
?ST=1. 5 -> { LC F } %négatif : on choisit -
```

```
DATO=C. 1 DAT1=C. 1 %et on l'écrit
```

```
%affichage de la partie graphique
```

```
A=R4. A DO=A %adresse en RAM
```

```
A=R3. A D1=A %adresse de l'écran
```

```
LC 351 { A=DATO. B DAT1=A. B DO+2 D1+2 C-1. X UPNC }
```

```
%on affiche les 50 lignes de RAM
```

```
%affichage de l'adresse
```

```
P=15 LC 4 P=0 %compteur de caractères
```

```
A=R3. A LC 006CA A+C. A B=A. A DO=A %on pointe l'endroit où
écrire l'adresse
```

```
C=R4. A D=C. A %on charge la valeur de l'adresse
```

```
{ C=0. A LC F C&D. A GOSUB HEXA %on affiche un caractère
```

```
B-1. A A=B. A DO=A DSR. A %on décrémente
```

```
C-1. S UPNC } %on passe au caractère suivant
```

```
RTN %fin de routine
```

```
*HEXA %affiche un caractère
```

```
A=C. A %valeur du caractère
```

```
GOSUB GHEX %on saute la table de caractères
```

```
$6277576763236377
```

```
$5344511455551511
```

```
$5262773226731533
```

```
$5214445154551511
```

```
$3773436133536371
```

```
*GHEX %fin du saut
```

```
C=RSTK A+C. A D1=A LC 04 %adresse de début de table, et début
de boucle
```

```
{ A=DAT1. 1 DATO=A. 1 DO+34 D1+16 C-1. B UPNC } %on
affiche le caractère quartet par quartet
```

```
RTN %fin de routine
```

```
@"
```

Toujours pour ceux qui préfèrent la source sous forme hexadécimale, saisissez-la sans espace ni saut et compilez la avec la commande H'' de la library de développement (887 octets, CRC # CDCEh) :

```
CCD20 D6300 8FB97 608F1 9762D 081AF 048FC 77628
1AF03 13134 07700 8FC57 608F3 8762D 23148 CA81A
F0270 11000 00000 00000 00000 00000 00000 00000
```

```

OFFF FDFD F7FF FFDF FFF7 FFF7 333B
DCCCE 433B3 3DCEC C4B33 33DFF FF705 55595 55565
55955 55655 59555 5DFFE F7055 55B55 5D655 5B555
D6555 B5555 DF7CF 70555 5B555 D6555 B555D 6555B
5555D FFEF7 09999 15666 46991 99564 66619 999DF
FFF70 FFFFF DFFFF 7FFFF FDFD F7FF FDFD F700
71353 10115 37150 716F1 7FA6E 5EE76 D0320 208FF
0200D 0808A 0C080 82410 00080 8A1C0 80824 01000
808A2 C0808 24001 00808 A3C08 08240 00108 08A4C
08082 40000 1808A 56074 50D88 1AF14 87540 C0865
40E08 1AF04 32100 8FF02 00808 B6213 40003 0CE5D
F625F 8F767 628F6 A7628 D3415 08757 08550 18657
08450 181AF 12130 34440 00CA1 31875 5030E 86550
30F15 C015D 081AF 14130 81AF1 31313 21531 4A149
16117 1A3E5 0F2F3 04208 1AF13 34AC6 00CAD 81308
1AF1C D7D23 0FOEF 77110 CDD41 30F7A 4E56E 01DA7
05062 77576 76323 63775 34451 14555 51511 52627
73226 73153 35214 44515 45515 11377 34361 33536
37107 CA131 31401 5B015 8016F 16F16 117FA 6E58E
01

```

• l'utilisation du logiciel •

Une fois la source assemblée, une fois l'objet "Code" stocké et une fois le programme lancé, la machine affiche l'écran suivant :



Nous obtenons donc le résultat attendu avec les trois zones définies au départ.

Maintenant pour naviguer, l'utilisation des touches est la suivante :

- [F1] : déplace l'adresse d'un quartet (# 1h),
- [F2] : déplace l'adresse de 16 quartets (# 10h),
- [F3] : déplace l'adresse de 256 quartets (# 100h),
- [F4] : déplace l'adresse de 4096 quartets (# 1000h),
- [F5] : déplace l'adresse de 65536 quartets (# 10000h),
- [F6] : change le sens de navigation,
- [C=] (touche DEL) : quitte le logiciel.

Quand on scrute la mémoire, on peut retrouver (sur une Hp 49 disposant d'une Rom 1.16 officielle), un écran comme celui-ci :



Par ailleurs, on peut retrouver des grobs qui sont noyés dans des logiciels stockés en Ram, ou même des morceaux d'images qu'on a préalablement utilisés (avant redémarrage à chaud, bien sûr).

Une particularité de ce logiciel est qu'il rafraîchit toujours l'écran même si l'on ne se déplace pas. C'est à dire que quand on est à l'adresse # 00000h, on peut voir évoluer l'horloge en temps réel.

• petite amélioration •

Ce petit logiciel permet de visiter la Ram, ce qui veut dire que quand vous posez un objet dans la pile, il serait également possible de le visualiser. Une petite amélioration de ce programme serait de visiter l'objet qui se trouve sur le niveau 1 de la pile, ou commencer au quartet # 0h si la pile est vide. Pour cela, il suffit de remplacer la deuxième ligne de la source par celle-ci :

A=DAT1. A R4=A. A %adresse de l'objet placé sur le niveau 1 de la pile

• la version Hp 48 série G •

Les utilisateurs de Hp 48 pourront eux aussi faire un tour d'horizon de la Ram de leur machine grâce à cette chaîne hexadécimale qu'il faut saisir sans espace ni saut et compiler avec le programme GASS48 présenté juste après :

```

CCD20 97300 8FB97 608F5 1110D 081AF 048F1 3C108
1AF03 13134 07700 8FC57 608F8 5C10D 23148 CA81A
F0270 11000 00000 00000 00000 00000 00000 00000
OFFF FDFD F7FF FFDF FFF7 FFF7 333B
DCCCE 433B3 3DCEC C4B33 33DFF FF705 55595 55565
55955 55655 59555 5DFFE F7055 55B55 5D655 5B555
D6555 B5555 DF7CF 70555 5B555 D6555 B555D 6555B
5555D FFEF7 09999 15666 46991 99564 66619 999DF
FFF70 FFFFF DFFFF 7FFFF FDFD F7FF FDFD F700
71353 10115 37150 716F1 7FA6E 5EE76 E0322 008FC
EE10D 0808A 4C080 82410 00032 0018F CEE10 808A4
C0808 24010 00808 A3C08 08240 01008 08A2C 08082
40001 0808A 1C080 82400 00180 8A060 7450D 881AF
14875 40C08 6540E 081AF 04320 108FC EE108 08B02
13400 030CE 5DF66 4F8F5 E0108 F75D0 08D34 15087
57085 50186 57084 50181 AF121 30344 4000C A1318
75503 0E865 5030F 15C01 5D081 AF141 3081A F1313
13215 314A1 49161 171A3 E50F2 F3042 081AF 1334A
C600C AD813 081AF 1CD7D 230F0 EF771 10CDD 4130F
7A4E5 6E01D A7050 62775 76763 23637 75344 51145
55515 11526 27732 26731 53352 14445 15455 15113
77343 61335 36371 07CA1 31314 015B0 15801 6F16F
16117 FA6E5 8E01

```

'GASS48' (86,5 octets - CRC # 1DB3h)

```

« "GROB 8 " OVER SIZE 2 / + " " + SWAP + OBJ"
# 4017h SYSEVAL # 56B6h SYSEVAL DROP NEWOB »

```

Philippe Pamart
[phpamart@nordnet.fr]