Source	**flashtro.com**
Tutorials	**AmigaCracking : R-TYPE II**
Protection	**MFM Protection**
(source) Original Author	**AlphaOne**
Submitted by (on flashtro.com)	**2013-06-05 04h13**
Version FR	**22/03/2016 Gi@nts v1.0**

R–TYPE 2
* CRACK TUTORIAL *

Matériels nécessaire

1) Un Amiga 500 avec 1 Mega de Chip ou l'émulateur WINUAE
2) Une Carte ACTION REPLAY MKIII (ou ça ROM Image)
3) Le jeu Original R-TYPE II ou son image CAPS (SPS 2065)
4) le logiciel Xcopy Pro en disquette ou image disk.
5) le logiciel ASM-ONE v1.20
6) Le binaire rmcloader.bin du SectorLoader de Rob Northen

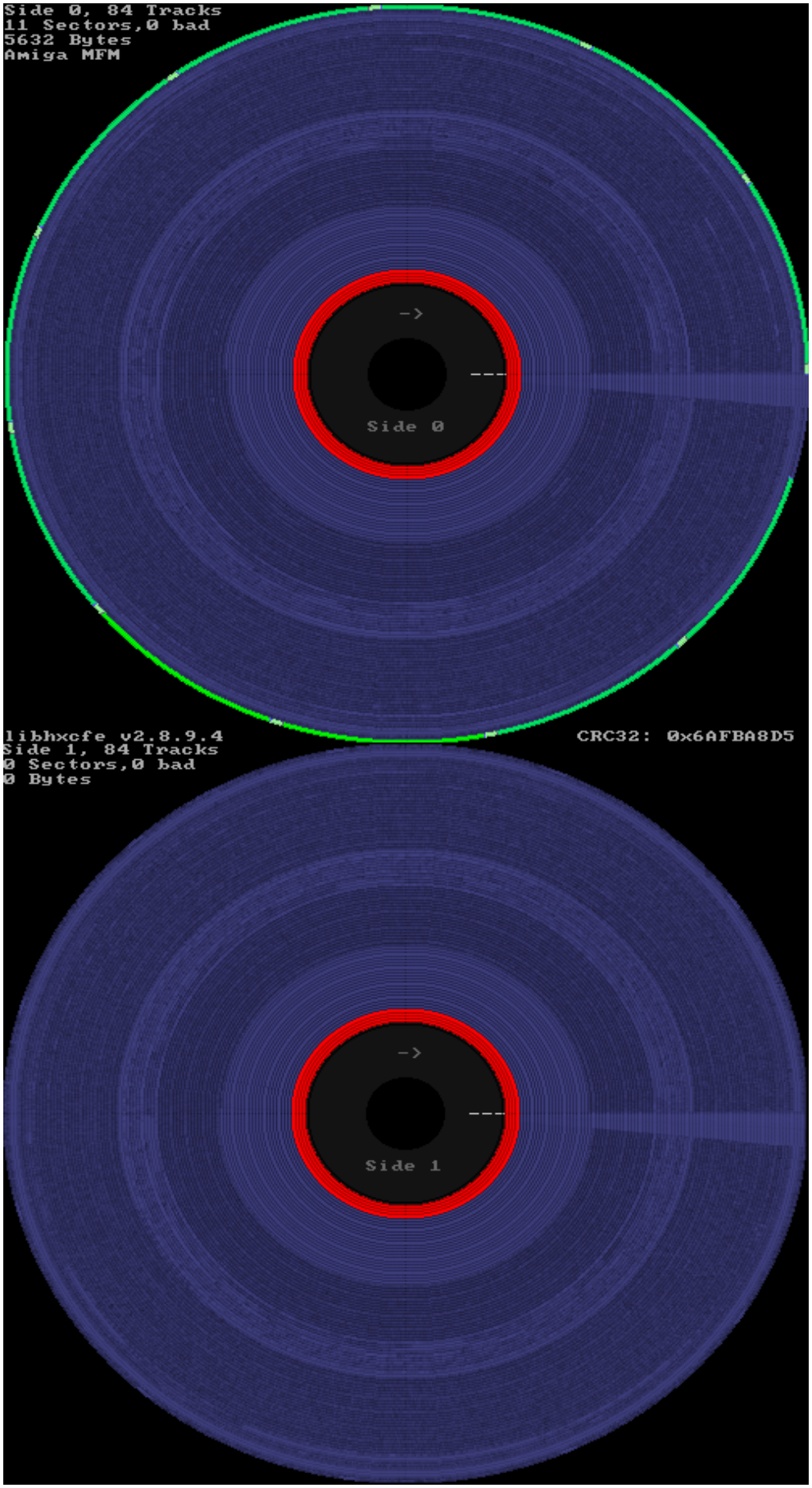
Général Info

Ce tutoriel Français est basé sur le tutoriel original de Alpha One .
Néanmoins, ce document n'est PAS une traduction mot par mot de celui-ci mais plus une nouvelle version.
Basé sur l'original mais avec des nouvelles informations et parfois une autre approche de la problématique.

Bon tuto.

Gi@nts

Table des matières


Side 0, 84 Tracks
11 Sectors,0 bad
5632 Bytes
Amiga MFM
->

Side 0
libhxcfe v2.8.9.4
Side 1, 84 Tracks
0 Sectors,0 bad
0 Bytes
CRC32: 0x6AFBA8D5
->

Side 1

Copiable ?

Comme dans Tous bon hack qui se respecte, on va commencer par essayer de copier le disk original.
Démarrer votre logiciel de copie préféré, à savoir Xcopy Pro
Choisissez le mode **DOSCOPY+**, **insérez** une **disquette vierge** en **DF1** et la **disquette du jeu original** en **DF0**
Lancer la copie

Cela semble assez clair, la copie ne fonctionnera pas et il semble que l'on est là un format spécifique.
On notera tout de même que l'ensemble des erreurs est de l'ordre de :

ERROR 5 '*Error in header/format long*'
ERROR 1 '*Less or more than 11 sectors*'
ERROR 3 *'No sync after gap found'*

Lancement du jeu

Insérez la disquette original du jeux dans le lecteur et **allumer** votre Amiga.
Si vous passez par un émulateur comme WinUAE, il est assez facile de connaître la piste en court.
Sur un amiga réel, à moins d'avoir un lecteur ou un montage spécifique affichant les pistes...
Quoi qu'il en soit, voilà ce que l'on peu noter sur les chargements du jeu.

Pistes	Information
00 -> 1	Micro Chargement, RAS, écran noir
2 -> 13	Ras, écran noir
13 -> 16	Image jeu r-type
57 -> 59	Image copyright #1
59 -> 60	Image copyright #2
16 -> 18	Ras
18 -> 20	Image écran de départ du jeu
21 -> 26	Démo du jeu L1 qui commence après semble t'il une phase de décompression
27 -> 32	Démo du jeu L2 qui commence après semble t'il une phase de décompression
33 -> 43	Démo du jeu L3 qui commence après semble t'il une phase de décompression

Analyse bootblock

Toujours avec la disquette original du jeux dans le lecteur de l'Amiga, nous allons regarder ce qui se passe dans le *bootblock*, **entrez** dans votre **AR**

#RT alias Read Track, permet le chargement de track

Tapez : RT 0 1 50000 (puis touche ESC à la seconde erreur) puis **D 50000**

```
rt 0 1 50000
Disk ok
-
d 50000
~050000 NEG.W   A7
~050002 SUBQ.B  #1,D0
~050004 CMP.B   -(A7),D0
~050006 BCLR    #4445,21(A4,D5.L)
~05000C LEA     50016(PC),A0
~050010 MOVE.L  A0,00000080.S
~050014 TRAP    #0
~050016 MOVE.W  #2700,SR
~05001A LEA     00080000,A7
~050020 MOVE.W  #7FFF,D0
~050024 MOVE.W  D0,00DFF096
~05002A MOVE.W  D0,00DFF09A
~050030 CLR.W   00DFF180
~050036 LEA     5004E(PC),A0
~05003A LEA     00078000,A1
~050040 LEA     5043E(PC),A2
~050044 MOVE.L  A1,-(A7)
~050046 MOVE.B  (A0)+,(A1)+
~050048 CMPA.L  A0,A2
~05004A BNE     00050046
~05004C RTS
'-------------------------------------
~05004E MOVEQ   #FFFFFFFF,D0
~050050 LEA     0007C000,A0
~050056 BSR     00050072
~05005A MOVEQ   #0,D0
~05005C LEA     00000500.S,A0
~050060 MOVE.L  A0,-(A7)
~050062 BSR     00050076
~050066 LEA     00000E2C.S,A0
~05006A MOVE.L  #AE3B9CE3,(A0)
~050070 RTS
'-------------------------------------
```

On peu voir une boucle de copie de donnée de l'adresse **$5004E** vers **$78000** et pour finir un **RTS**

La valeur **$7C000** est mise dans **A0** et on effectue un **BSR 50072**
puis on met la valeur **$500** dans **A0** et on effectue un **BSR 50076**
suivi d'une modification de **A7** et finalement le **RTS exécute** le code en **$500**

Il est donc fort probable que le code en **$50072 et/ou $50076** charge des données en mémoire **$500**
On notera aussi la valeur **AE3B9CE3** mis dans **A0** à l'adresse **$5006A**
C'est une valeur... peu commune et juste avant, le chargement de donnée de **$E2C** dans **A0**...

Effectue un **redémarrage** de votre Amiga et attendez la fin de la 1er phase de chargement
(piste 00->13, voir tableau plus haut) puis **entrez** dans notre **AR**

Tapez : D

```
d
~078174 BEQ     0007816C
~078176 MOVE.W  D4,9C(A1)
~07817A MOVE.W  #4000,24(A1)
~078180 RTS
```

Comme prévue, le code en court d'exécution est dans la zone **$78000**

Tapez : **R** puis **D 500**

```
r
D0=FC8603FE 00008000 00000000 00000000  00000001 FFFF001A 00000000 00000003
A0=00000AFE 00DFF000 0007FEE2 00000CA4  2F3B7F57 D04698F6 000008C8 0007FED0
PC = 0007FEEC USP = 00C014B2 SR = 2704 T=0 S=1 I=111 X=0 N=0 Z=1 V=0 C=0

d 500
~000500 MOVEQ   #0,D0
~000502 MOVEQ   #0,D1
~000504 MOVEQ   #0,D3
~000506 PEA     512(PC)
~00050A MOVE.L  (A7)+,00000010
~000510 ILLEGAL
~000512 MOVEM.L D0-D7/A0-A7,-(A7)
~000516 PEA     532(PC)
~00051A MOVE.L  (A7)+,00000010
~000520 MOVEA.L A7,A0
~000522 LINEF
~000524 ORI.B   #40,D2
```

Cela ressemble à un petite routine de *'copylock'*
Peu être pouvons nous gagner du temps sur la compréhension de celle-ci en allant directement regarder à l'adresse notée préalablement : **$E2C** (voir code plus haut sur image du *bootblock* en **$50066**)

Tapez : D E2A

```
d e2a
~000E2A CMP.L   #AE3B9CE3,D0
~000E30 BNE     000021EA
~000E34 LEA     E3C(PC),A0
~000E38 MOVE.L  A0,00000020.S
~000E3C MOVE.W  #2000,SR
~000E40 LEA     00DFF000,A0
~000E46 MOVE.W  #7FFF,D0
```

Il est fort probable que ce soit le code qui soit exécuté après la routine de *'copylock'*
Le test en **$E2A** est intéressant car l'on compare la valeur **AE3B9CE3** avec **D0**
On notera au passage que c'est la même valeur que dans notre *bootblock* préalablement chargé en $**50000**
voir **$5006A**

Redémarrer votre Amiga et des la fin de la première phase de chargement (jusqu'à piste 13) **entrer** dans votre **AR**
Tapez : `a e2a` puis

```
^000E2A move.w #$f00,$dff180
^000E32 bra $e2a
^000E34
x
```

```
a e2a
^000E2A move.w #$f00,$dff180
^000E32 bra $e2a
^000E34

x
```

Le code continue son petit chemin puis, l'écran passe au rouge. Il est temps pour nous **d'entrer** dans notre **AR**
Tapez : `r`

```
r
D0=AE3B9CE3 00000000 0000FFFF 00000000  55555555 00000000 00001AF9 00001051
A0=00000E2C 00DFF000 00BFD000 0007CC10  00001558 00C014B6 00C00276 00080000
PC = 00000E32 USP = 00C014B2 SR = 2700 T=0 S=1 I=111 X=0 N=0 Z=0 V=0 C=0
```

On peu donc connaître la valeur de **D0**, à savoir : **AE3B9CE3**
C'est la valeur qu'il s'attend à avoir avant lors du test en **$E2A** et si cella n'est pas le cas, il effectue un **BNE 21EA**
Regardons d'ailleurs ce qui se passe en **$21EA**

Tapez : `d 21ea`

```
d 21ea
~0021EA MOVE.W  #F0,00DFF180
~0021F2 BRA     000021EA
;--------------------------------------
```

Code assez simple à comprendre, on met le fond d'écran en **VERT** et on boucle sur ce code
Cela ressemble à 100% à une procédure de fin de parcourt :)
Check du disk, bon = on continue, pas bon.... on boucle sur un jolie écran vert.

Passer ce système de protection est assez facile, il suffit d'écraser la valeur de **D0** suivi d'un JUMP vers la fin de la routine de *'copylock'*

Pour être sur qu'il n'existe pas d'autre vérification sur cette valeur, le mieux est rechercher en mémoire si elle est présente à une autre adresse.

Tapez : `f AE 3B 9C E3`

```
f AE 3B 9C E3
Search from: 000000 to: C80000
000060  07801E  07FF0C  07FF24  07FFBA
Ready.
```

$7801E étant très proche de **$78000** est à coup sur la 1er copie que nous avons vue juste au dessus.
Il semblerait donc qu'il existe pour l'instant 4 autres vérifications au zone mémoire soulignées en rouge dans l'image ci dessous.

$7801E
$7FFF0C
$7FF24
$7FFBA

Nous reviendrons plus tard sur cette problématique.

Fonctionnement du trackloader part1

Continuons l'exécution de notre code
Tapez : `G e34`

Un *trackloader* fonctionne avec le contrôleur de disquette, on va donc chercher si les adresses des I/O disk sont utilisées.
Tapez : `FA bfd000` puis `FA bfe001`

```
fa bfd000
Search from: 000000 to: C80000
00F9FC LEA      00BFD000,A2
Searched up to adr: C80000
Ready.

fa bfe001
Search from: 000000 to: C80000
001356 BTST     #6,00BFE001
00EC6C BTST     #6,00BFE001
00EC86 BTST     #7,00BFE001
00EE58 LEA      00BFE001,A0
00F998 BTST     #4,00BFE001
00FA38 BTST     #5,00BFE001
0159A6 BSET     #1,00BFE001
015CF2 BSET     #1,00BFE001
Searched up to adr: C80000
Ready.
```

```
$bfe001         bits 6  /Fire0          Aucun intérêt pour nous dans le cadre du trackloader
$bfe001         bits 7  /Fire1          Aucun intérêt pour nous dans le cadre du trackloader
$bfe001         bits 4  /TK0            Intéressant     ($F998)
$bfe001         bits 5  /RDY            Intéressant     ($FA38)
$bfe001         bits 1  /LED            Aucun intérêt pour nous dans le cadre du trackloader
```

On devrait donc trouver notre *trackloader* au dessus de l'adresse **$F998**

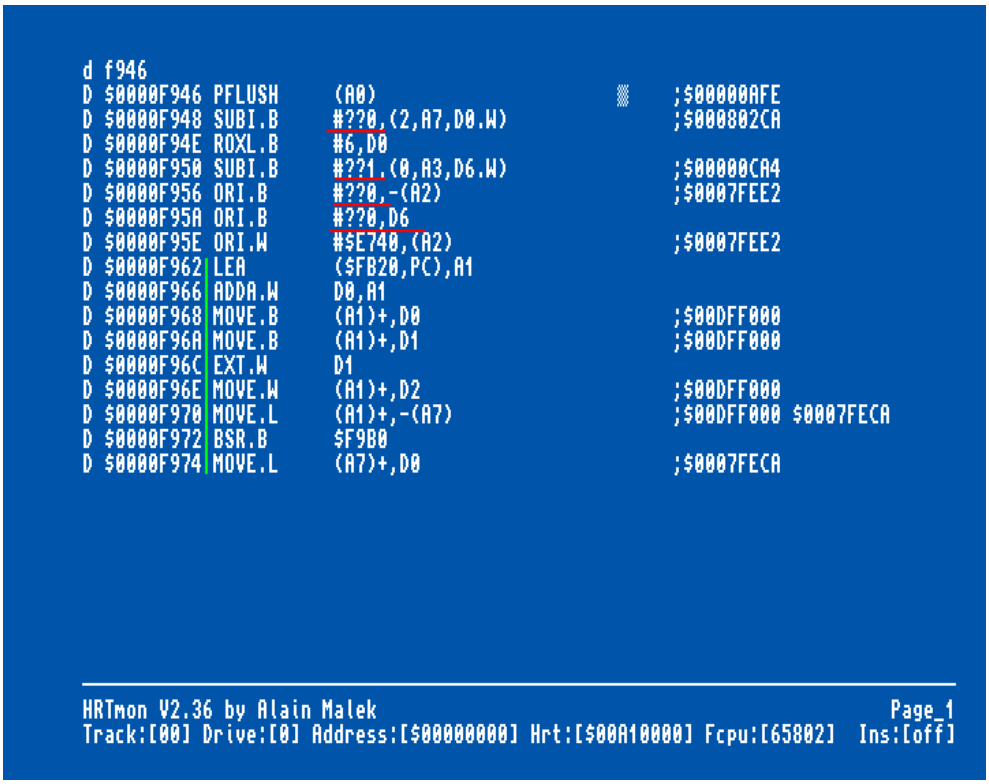
Tapez : `D F998` et remonter le code jusqu'à trouver quelque chose d'intéressant.

```
d f998
~00F998 BTST    #4,00BFE001
```

```
~00F940 SUBI.B  #0,00000002.S
~00F946 LINEF
~00F948 SUBI.B  #0,2(A7,D0.W)
~00F94E ROXL.B  #6,D0
~00F950 SUBI.B  #1,0(A3,D6.W)
~00F956 ORI.B   #0,-(A2)
~00F95A ORI.B   #0,D6
~00F95E ORI.W   #E740,(A2)
~00F962 LEA     FB20(PC),A1
~00F966 ADDA.W  D0,A1
~00F968 MOVE.B  (A1)+,D0
~00F96A MOVE.B  (A1)+,D1
~00F96C EXT.W   D1
~00F96E MOVE.W  (A1)+,D2
~00F970 MOVE.L  (A1)+,-(A7)
~00F972 BSR     0000F9B0
~00F974 MOVE.L  (A7)+,D0
~00F976 RTS
;=====================================
~00F978 LEA     FB1A(PC),A2
~00F97C TST.W   D0
~00F97E BNE     0000F982
~00F980 MOVE.W  (A2),D0
~00F982 MOVE.W  D0,(A2)+
~00F984 MOVE.L  A0,(A2)+
~00F986 TST.W   D0
~00F988 BMI     0000F98C
~00F98A RTS
;=====================================
~00F98C BSR     0000F9F6
~00F98E ORI.B   #2,100(A2)
```

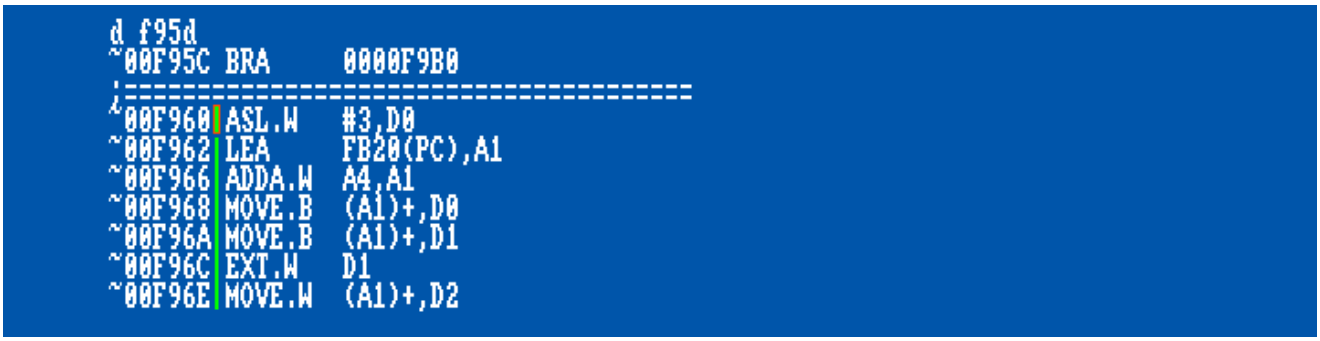
$F956 à **$F962** ne ressemble pas à du code normal, d'ailleurs **$F946** valide ceci.

C'est plus visible sous le logiciel **HRTmon** pour info.



Il y a forte chance que le code commence en **$F962** et même... si on regarde bien :
$F95E semble autant suspect qu'en **$F956.**
Si on remonte donc notre désassemblage juste au dessus de cette dernière adresse bizarre :
$F95E - 1 = $F95D

Tapez : `D F95D`

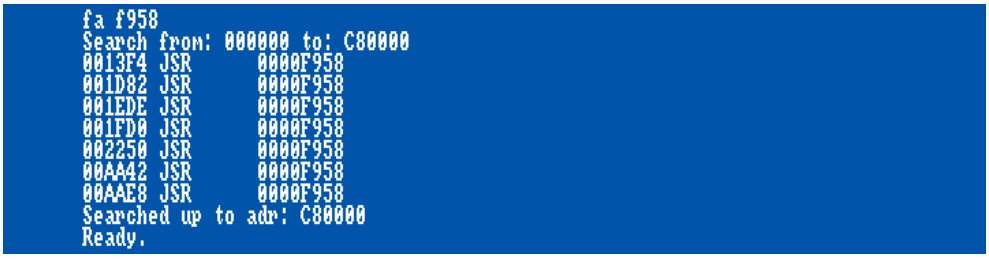


On retombe bien sur notre code (ligne verte image ci dessus) mais avec une instruction en plus.
Et on voie assez clairement que le début du code semble être **$F960**

Si on cherche en mémoire un branchement vers cette adresse, on trouve bien quelque chose.

Tapez : `D F960`

Tapez : `D F958`



Il y a du monde, ce qui est assez logique car R-type2 possède plusieurs niveaux.
D'ailleurs, rien qu'au début de ce tuto, on a compter pas mal de chargements.

`$13F4, $1D82, $1EDE, $1FD0, $2250, $AA42, $AAE8`

Redémarrer votre Amiga, laisser finir la 1er phase de chargement (jusqu'à piste 13) **entrez** dans votre **AR**

Taper : **`d f958`** puis

```
a f958
^00F958 BRA     F958
^00F95A
```

```
d f958
^00F958 BRA     0000F960
;=====================================


a f958
^00F958 bra f958
^00F95A

x
```

Assez vite, le jeu semble ne plus rien faire... **Entrer** dans votre **AR**

Taper : **`D`** (puis touche **ESC**) puis **`R`**
Noter les registres puis continuer l'exécution du code vers le *trackloader* en tapant : **`G F960`**

```
d
^00F958 BRA     0000F958

r
D0=00000001 00000001 0000FFFF 0000FFFF  55555555 00000000 AAAAF2ED 00005045
A0=0004CE58 00DFF000 00BFD000 00030188  00001558 00C014B6 000148F6 000004F4
PC = 0000F958 USP = 00C014B2 SR = 2000 T=0 S=1 I=000 X=0 N=0 Z=0 V=0 C=0

g f960_
```

Assez rapidement, le jeu va à nouveau entrer dans notre boucle et ne plus rien faire, **Entrez** de nouveau dans votre **AR**
Noter les registres puis continuer l'exécution du code vers le *trackloader* en tapant : **`G F960`**

Continuer ce mode opératoire jusqu'au chargement du niveau 1 (voir plus si affinité *)
et noter les informations de registre dans un tableau.

`* Mode Invulnérable =` Commencer une partie, mètre le jeu en pause puis appuyez sur le bouton gauche de la souris et la touche F1 du clavier. Relâcher la touche et le bouton, un écran vert confirme la manipulation. Appuyez sur P pour retourner au jeu.

Tableau des registres quand appel de la routine de *trackloader*

Tableau1

	1er chargement	Screen RTYPE2	Screen (c) #1	Screen (c) #2	N/A	Screen Start	
D0	N/A	00000001	0000000F	00000010	00000002	00000004	
D1	N/A	00000001	0000001F	0000001F	0000001F	0000001F	
D2	N/A	0000FFFF	0000FFFF	0000FFFF	0000FFFF	0000FFFF	
D3	N/A	0000FFFF	0000FFFF	0000FFFF	0000FFFF	0000FFFF	
D4	N/A	55555555	55555555	55555555	55555555	55555555	
D5	N/A	00000000	00000000	00000000	00000000	00000000	
D6	N/A	AAAAF2ED	0000FFFF	9C71FFFF	3A6EFFFF	399C093D	
D7	N/A	00005045	0000FFFF	1451FFFF	1044FFFF	114FFFF	FIRE
A0	N/A	0004CE58	0004CE58	0004CE58	0004CE58	00053000	
A1	N/A	00DFF000	000102F6	000102F6	000102F6	0005CF4C	
A2	N/A	00BFD000	00054658	00054658	00054658	0002FD80	
A3	N/A	00030188	00056E58	00056E58	00056E58	00037A80	
A4	N/A	00001558	00059658	00059658	00059658	0003F780	
A5	N/A	00C014B6	0007BA00	0007BA00	0007BA00	00047480	
A6	N/A	000148F6	000148F6	000148F6	000148F6	000148F6	
A7	N/A	000004F4	000004F4	000004F8	000004F8	000004F8	
(SP)	N/A	1EE4	1EE4	1EE4	AA48	13FA	

		Anim Intro	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5	Niveau 6
D0		00000003	00000005	00000006	00000007	00000008	00000009	0000000A
D1		00000078	0000000E	00063918	00063918	00063918	00063918	00063918
D2		0000000E	00063000	00063000	00063000	00063000	00063000	00063000
D3		0000FFFF	00007D00	00000000	00000000	00000000	00000000	00000000
D4		55555555	00000028	00000028	00000028	00000028	00000028	00000028
D5		00000000	00008FCA	00000001	00000001	00000001	00000001	00000001
D6		0000FFFF	82E10080	00000030	00000030	00000030	00000030	00000030
D7	FIRE	0000FFFF	0000FFFF	0000FFFF	0000FFFF	0000FFFF	0000FFFF	0000FFFF
A0		00053000	0004F180	0004F180	0004F180	0004F180	0004F180	0004F180
A1		00010378	0001039C	0001039C	0001039C	0001039C	0001039C	0001039C
A2		0000B4F6	0005058C	00007f89A	00007f89A	00007f89A	00007f89A	00007f89A
A3		000151F0	000796E8	00007F93A	00007F93A	00007F93A	00007F93A	00007F93A
A4		0003F780	0000F1A4	0007F9DA	0007F9DA	0007F9DA	0007F9DA	0007F9DA
A5		000150B8	00009348	000150C0	000150C0	000150C0	000150C0	000150C0
A6		000148F6	000148F6	000148F6	000148F6	000148F6	000148F6	000148F6
A7		000004F8	000004F8	00004F8	00004F8	00004F8	00004F8	00004F8
(SP)		1D88	AAEE	AAEE	AAEE	AAEE	AAEE	AAEE

Avec ce tableau, on va comprendre le fonctionnement du *trackloader*

Fonctionnement du Trackloader part2

Si on regarde de nouveau le code du *trackloader* on peu voir qu'une lecture mémoire en **$fb20** est effectuée.
Taper : `d f960` puis `m f920`

```
d f960
~00F960 ASL.W   #3,D0
~00F962 LEA     FB20(PC),A1
~00F966 ADDA.W  D0,A1
~00F968 MOVE.B  (A1)+,D0
~00F96A MOVE.B  (A1)+,D1
~00F96C EXT.W   D1
~00F96E MOVE.W  (A1)+,D2
~00F970 MOVE.L  (A1)+,-(A7)
~00F972 BSR     0000F9B0
~00F974 MOVE.L  (A7)+,D0
~00F976 RTS
;=====================================

m fb20
:00FB20 02 01 01 21 00 02 43 CC 1A 03 00 45 00 00 8B C5 ...!..C....E....
:00FB30 20 01 00 39 00 00 73 8F 24 0B 00 1C 00 00 38 AF  ..9..s.$.....8.
:00FB40 27 04 00 08 00 00 10 70 28 01 00 9F 00 01 3E CB '.......p(.....>.
:00FB50 35 05 00 7F 00 00 FE 22 40 01 01 0D 00 02 1A 96 5..▒...."@.......
:00FB60 56 07 00 7C 00 00 F8 5A 61 00 00 56 00 00 AD 35 V..|...Za..V...5
:00FB70 68 03 00 76 00 00 ED 9B 72 02 00 02 00 00 04 00 h..v....r.......
:00FB80 72 04 00 02 00 00 04 00 72 06 00 02 00 00 04 00 r.......r.......
:00FB90 72 08 00 02 00 00 05 75 72 0B 00 26 00 00 4D 5C r.......ur..&..M\
:00FBA0 76 02 00 27 00 00 4F EF AA AA AA AA AA AA AA AA v..'..O.........
:00FBB0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA ................
```

Cela ressemble à une *filetable*
Si on regarde le début du code du *trackloader*, ça nous donnes :

```
00F960   ASL.W    #3,D0            #Décalage de 3 bit vers la gauche de D0 ce qui nous crée au final D0=D0*8
00F962   LEA      FB20(PC),A1      #On met FB20 dans A1, c'est notre Adr. De départ de notre filetable
00F966   ADDA.W   D0,A1            #Ajoute D0 à A1, ce qui va donc nous positionner dans la filetable
00F968   MOVE.B   (A1)+,D0         #On récupère la premières valeur du tableau 'TA1' dans D0, ensuite A1=A1+1
00F96A   MOVE.B   (A1)+,D1         #On récupère la seconde valeur du tableau 'TA2' dans D1, ensuite A1=A1+1
00F96C   EXT.W    D1               #On efface les 2 premiers bytes de D1
00F96E   MOVE.W   (A1)+,D2         #On récupère les deux premières valeurs du tableau 'TA3' dans D2, ensuite A1=A1+2
00F970   MOVE.L   (A1)+,-(A7)      #On récupère les quatre premières valeurs du tableau 'TA4' dans SP
00F972   BSR      0000F9B0         #On se branche sur le sous-programme en $F9B0
00F974   MOVE.L   (A7)+,D0         #On restitue le contenue préalablement sauvé 2 lignes au dessus vers D0 (check?)
00F976   RTS                       #On revient
```

Ça nous découpe la *filetable* comme ceci :

	D0	D1	D2		A7			
	TA1	TA2	TA3		TA4			
$00FB20	02	01	01	21	00	02	43	CC
$00FB28	1A	03	00	45	00	00	8B	C5
$00FB30	20	01	00	39	00	00	73	8F
$00FB38	24	0B	00	1C	00	00	38	AF
$00FB40	27	04	00	08	00	00	10	70
$00FB48	28	01	00	9F	00	01	3E	CB
$00FB50	35	05	00	7F	00	00	FE	22
$00FB58	40	01	01	0D	00	02	1A	96
$00FB60	56	07	00	7C	00	00	F8	5A
$00FB68	61	00	00	56	00	00	AD	35
$00FB70	68	03	00	76	00	00	ED	9B
$00FB78	72	02	00	02	00	00	04	00
$00FB80	72	04	00	02	00	00	04	00
$00FB88	72	06	00	02	00	00	04	00
$00FB90	72	08	00	02	00	00	05	75
$00FB98	72	0B	00	26	00	00	4D	5C
$00FBA0	76	02	00	27	00	00	4F	EF

N/A
SCREEN START
Niveau 1
Niveau 2
Niveau 3
Niveau 4
Niveau 5
Niveau 6
SCREEN R-TYPE2
Anim Intro
Screen (c) #1
Screen (c) #2

TA1 varie de **$02** à **$76**, peu être indique t'il la piste de départ.
TA2 varie de **$01** à **$0B**, **!11** Vue que le format n'est pas standard **AMIGADOS** et que l'on a des **ERROR 1** '*Less or more than 11 sectors*' sous X-Copy, **TA2** pourrait du coup indiquer un nbr de secteurs

TA3 varie de $0002 à $0121, il s'agit sûrement d'une taille en secteur car trop petit pour une taille direct et trop gros pour autre chose.
TA3, au vue de sa taille, on imagine plus aussi une taille mais en quoi...

Ce qui serait intéressante serait dans un premier temps de connaître l'adresse de destination mémoire, il est impossible que ce soit la même tout le temps donc elle doit être passé aussi via les registres.

Au vue du tableau des registres plus haut, il est impossible qu'elle se trouve en **D0-D7** car
D0 à **D2** sont déjà utilisé et au vue des valeurs de **D3** à **D7** cela semble improbable, on va donc commencer nos tests avec **A0**

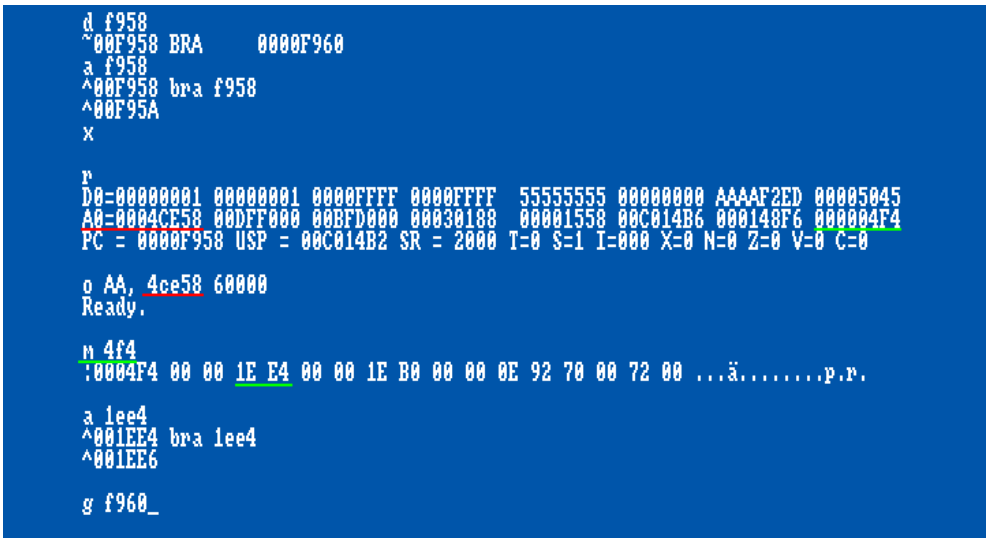
Redémarrer votre Amiga et des la fin de la première phase de chargement (jusqu'à piste 13) **entrez** dans votre **AR**
Taper : `a f958` puis

```
^00F958 BRA     F958
^00F95A
X
```

Très vite l'Amiga semble ne plus rien faire.
On va entrer dans l'AR, afficher les registres et remplir la zone mémoire que l'on pense de destination avec la valeur **AA**
puis regarder dans la pile l'adresse de retour et poser une boucle à celle-ci.

Entrez dans l'**AR** et **Taper :**
`R` puis `o AA, 4ce58 60000`
`M 4f4` *(pour afficher l'adresse de retour stockée dans la pile)* puis
`A 1EE4`

```
^001EE4 bra 1ee4
^001EE6

G f960
```



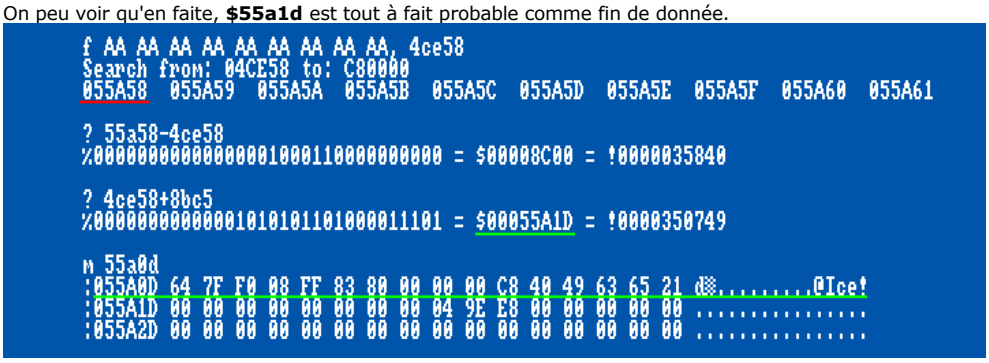
L'Amiga va charger quelques pistes et assez vite, ne plus rien faire. **Entrez** dans l'**AR** et **Taper :**
`f AA AA AA AA AA AA AA AA, 4ce58` *(et appuyer sur ESC des le 1er affichage)*

La fin des données est située en **$55a58**, ce qui nous donne comme taille : **$55A58-$4ce58 = $8c00**
Hummm, ca ne ressemble pas à ce que l'on a dans notre *filetable*...

Si **TA4** est la taille de nos donnée chargée par le *trackloader*, la fin des données devrait se trouver en **$4ce58+$8bc5 = $55a1d**
Si on regarde de plus prêt...

Taper :
`m 55a0d`

On peu voir qu'en faite, **$55a1d** est tout à fait probable comme fin de donnée.



TA4 serait donc plutôt la taille des données sans extra-Info

D'ailleurs, si on prends pour référence la taille de ce qui a été lue, à savoir : **$8C00**
Que l'on divise par **TA3** de '*screen RTYPE2*' de notre *filetable*, a savoir **$45**

$8C00 = !35840
$45 =!69 mais en faite, si on part de Zero comme 1er valeur, cella nous donne un total de !70 valeurs possibles.
Donc : !35840 /!70 =!512

Tiens donc, exactement la taille d'un secteur AMIGADOS.
Cela reste à vérifier mais il semblerait que **TA3** indique la taille des données lue en nombre de secteur de 512 bytes

Taille d'une piste

Pour mieux ripper l'ensemble des données il est préférable (mais pas obligatoire) de connaître la taille d'une piste.
Sur une disquette au format AmigaDOS, elle est de $1600 mais quand est t'il sur notre jeux R-Type II ?

Au vue des erreurs affichées lors de la phase de copie avec le logiciel X-copy pro. L'ensemble des pistes semblent non omogene...
On va quand même vérifier quelques pistes. La 1er piste étant AmigaDOS, on va faire nos tests sur la *Seconde* et *Troisième* **TRACK**

Redémarrer votre Amiga et des la fin de la première phase de chargement (jusqu'à piste 13), ou plus exactement, dés le début de la seconde phase, **entrer** dans votre **AR**
Taper : `a f958` puis

```
^00F958 BRA     F958
^00F95A
X
```

Très vite l'amiga semble ne plus rien faire.
On va **entrer** dans **l'AR**, afficher les registres, regarder dans la pile l'adresse de retour et poser une boucle à celle-ci
puis modifier table *filetable* pour lire le début de la 3eme **TRACK**

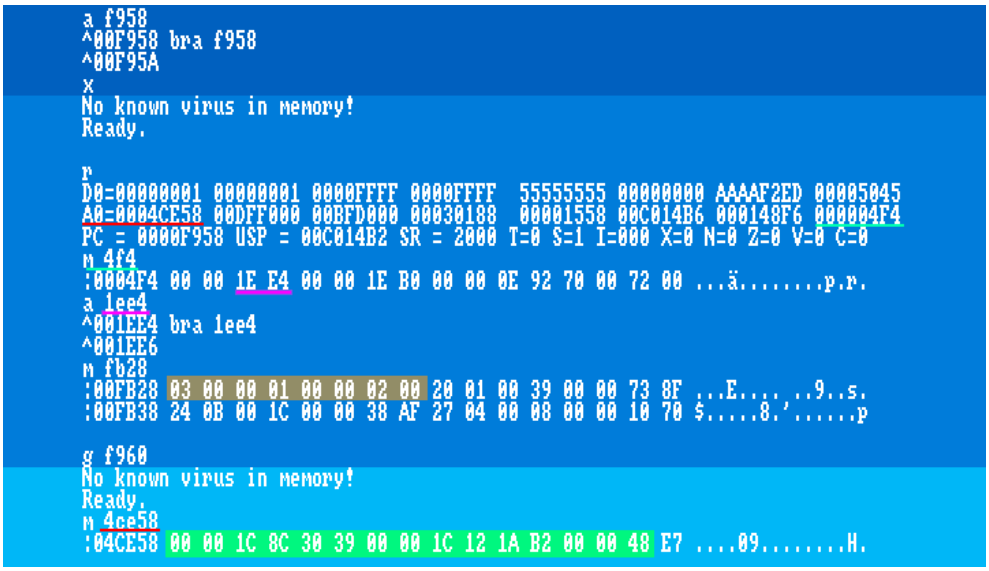
Entrez dans l'**AR** et **Taper :**
`R` *(on note **A0** et **A7**)*
`M 4f4` *(On affiche le contenue à l'adresse de **A7**, a savoir l'adresse de retour stockée dans la pile, ici **1EE4**)* puis

```
A  1EE4
^001EE4 bra 1ee4
^001EE6
```

`M fb28` *(Comme vue plus haut, c'est l'adresse fb28 qui est adressé en 1er donc on modifie celle-ci)*
`:00FB28 03 00 00 01 00 00 02 00` *(3eme track, sector 0, 1 secteur, taille 200)*

`G f960`

Très vite l'amiga semble ne plus rien faire. On re-**entrer** dans **l'AR**, puis
Taper : `M 4CE58` *(4ce58 étant l'adresse de destination du trackloader, contenue dans A0 lors de l'appel de la routine)*



On note, le début de la TRACK03 est : `00 00 1C 8C 30 39 00 00 1C 12 1A B2 00 00 48 E7`

On refait la même manipulation mais pour lire le début de la 4eme track donc modification de $fb28 par
`:00FB28 04 00 00 01 00 00 02 00` *(Début 4eme track, sector 0 avec une taille de 1 secteur, taille $200 bytes)*

On note, le début de la TRACK04 est : `00 40 3D 40 00 CA 3D 41 00 CC 41 F9 00 02 76 00`

Maintenant, on va refaire cette manipulation mais en commençant 1 TRACK avant (donc 2 au lieu de 3 **et** 3 au lieu de 4)
mais en lisant le plus de donnée possible d'une piste.
Sur une disquette AmigaDOS il y a 11 secteurs, on va donc prendre un peu plus pour 'déborder' sur la prochaine TRACK

Refaite donc la manipulation mais avec
`:00FB28 02 00 00 0F 00 00 1E 00` *(Début 2eme track, sector 0 avec une taille de 15 secteurs, taille $1E00 bytes)*

Puis on recherche en mémoire le début que l'on a noté de la **TRACK03** en **Tapant :**
`F 00 00 1C 8C 30 39 00 00 1C 12 1A B2 00 00 48 E7, 4CE58` *(Début le Premier affichage, **appuyer** sur **ESC**)*
Cela nous donnes **$4E616** donc **$4E616 - $4CE58 = $17BE**
La taille de la **TRACK02** est donc de **$17B0** au lieu des **$1600** Standard AmigaDOS

Refaite la manipulation mais avec
`:00FB28 03 00 00 0F 00 00 1E 00` *(Début 3eme track, sector 0 avec une taille de 15 secteurs, taille $1E00 bytes)*
`F 00 40 3D 40 00 CA 3D 41 00 CC 41 F9 00 02 76 00, 4CE58` *(Des le début du Premier affichage, **appuyer** sur **ESC**)*
Cela nous donnes **$4E658** donc **$4E658 - $4CE58 = $1800**

Bingo, pas la même taille. La taille de la **TRACK03** est donc de **$1800** au lieu des **$1600** Standard AmigaDOS

Taille d'un secteur

On ne sait jamais... Même si le 'standard' nous appel vers une taille de 512 bytes, il est préalablement de tester.

On va se passer de copie d'écran vue que ce sont les mêmes manipulations que précédemment mais en travaillant sur les secteurs.

Refaite donc les manipulation mais avec
`:00FB28 02 01 00 01 00 00 20 00` *(Début 2eme track, sector 1 avec une taille de 1 secteur, taille $200 bytes)*

On note, le début du secteur 1 en TRACK02 est : `70 00 72 00 76 00 48 7A 00 0A 23 DF 00 00 00 10`

Refaite donc la manipulation mais avec
`:00FB28 02 00 00 02 00 00 40 00` *(Début 2eme track, sector 0 avec une taille de 2 secteurs, taille $400 bytes)*

Puis on recherche en mémoire le début que l'on a noté du secteur 1 en TRACK02
`F 70 00 72 00 76 00 48 7A 00 0A 23 DF 00 00 00 10, 4CE58` *(Début le Premier affichage, appuyer sur ESC)*

Cela nous donnes **$4D058** donc **$4D058 - $4CE58 = $200**
La taille d'un secteur en track2 est donc de $200 bytes à savoir 512 bytes en décimal

Calcul avant Rip des pistes

Afin de faire tenir notre RIP de pistes sur une disquette AmigaDOS, on se doit de ripper uniquement les données.
Pour cela nous devons calculer exactement ce que l'on doit copier.

La premier track est de type AmigaDos (voir Ecran Xcopy) on commencera donc le début de notre rip à la piste $2 et secteur $0
Si on regarde la *filetable* on peu remarquer que la plus grosse valeur en **TA1** est de **$76** avec une taille de donnée en **TA4** de **$4FEF**
A savoir **Screen © #2**

On va déjà regarder de quoi à l'air la fin de ces données.

Redémarrer votre Amiga et des la fin de la première phase de chargement (jusqu'à piste 13) **entrez** dans votre **AR**
Taper : `a f958` puis

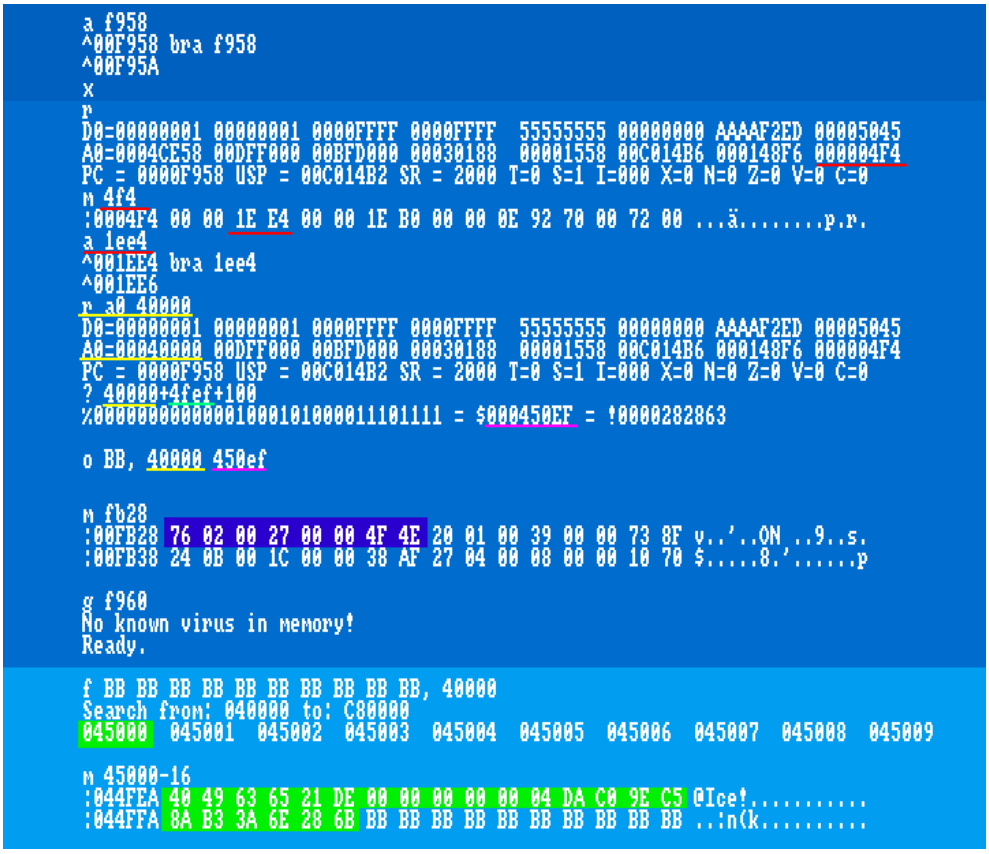
```
^00F958 BRA     F958
^00F95A
X
```

Très vite l'amiga semble ne plus rien faire.
Entrez dans l'**AR** et **Taper :**

```
R  (on note A7)
M 4f4 (On affiche le contenue à l'adresse de A7, a savoir l'adresse de retour stockée dans la pile, ici 1EE4) puis
A  1EE4
^001EE4 bra 1ee4
^001EE6

R  a0 40000    (On positionne A0 à 40000 pour l'adresse de destination)

O BB, 40000 450EF  (On remplie la zone mémoire 40000 à 450EF du patern BB)
                   (450EF étant le somme de la longueur dans la filetable de Screen © #2 + $4000 (adr de depart) + delta de $100)


M fb28 (Comme vue plus haut, c'est l'adresse fb28 qui est adressé en 1er donc on modifie celle-ci)
:00FB28 76 02 00 27 00 00 4F 4E          (Début 76 eme track, sector 2 avec une taille de $27 secteurs, taille $4F4E bytes)

G F960
```

Très vite l'amiga semble ne plus rien faire. **Entrez** dans l'**AR** et **Taper :**

```
F BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB, 40000 (Des le début du Premier affichage, appuyer sur ESC)
```



Fin de donnée a riper ==> `40 49 63 65 21 DE 00 00 00 00 00 04 DA C0 9E C5 8A BB 3A 6E 28 6B`

Rip des pistes

Bon, il n'y a plus qu'a comme on dit.
On a vue que la taille d'une **TRACK** n'était pas constant et que celle-ci pouvait atteindre une taille de $1800 ce qui, à ma connaissance est le maximum que l'on puisse avoir sur Amiga.
On va donc partir de cette base pour nos calcul et ajuster si besoin à la fin.

Au vue de ce que l'on sait et de notre *filetable*, notre début de rip va commencer en TRACK2 et aller jusqu'à la TRACK 76 + 27 secteurs
$75 – $2 = $73
$73 * $1800 = $AC800
$AC800 + $4E4F = $B164F

Le *trackloader* fonctionne ici en taille de secteur donc : **$B164F / $200 = $58B**

Comme je suis d'une nature méfiante, on va ajouter encore $F à celui-ci ce qui nous donnes : **$58B + $F = $59A**
De toute façon, on à noté la fin des données en hexa donc, on pourra ajuster si nécessaire.

N'oublions pas que le *trackloader* a aussi besoin d'une taille de donnée donc : **$59A * $200 = $B3400**

C'est partie, je passe la section commentaire qui reste exactement la même que précédemment.

Redémarrer votre Amiga et des la fin de la première phase de chargement (jusqu'à piste 13) **entrez** dans votre **AR**
Taper : `a f958` puis

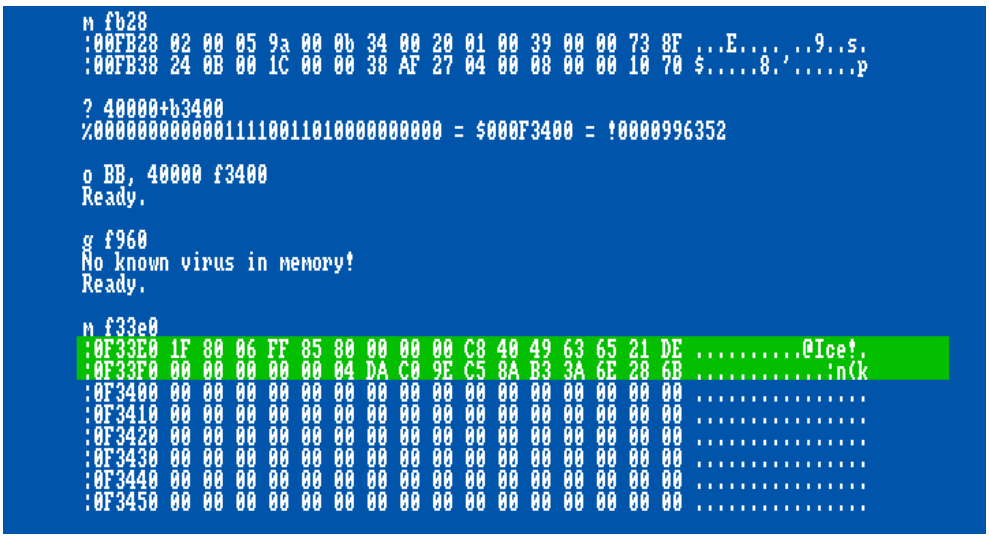
```
^00F958 BRA     F958
^00F95A
X
```

Très vite l'amiga semble ne plus rien faire. **Entrez** dans l'**AR** et **Taper :**

```
A  1EE4
^001EE4 bra 1ee4
^001EE6

R  A0 40000
M fb28
:00FB28 02 00 05 9A 00 0B 34 00

O BB, 40000 f3400
G f960
```

Bon, pour une raison que j'ignore, la fonction de recherche de chaîne dans la MKIII ne fonctionne pas bien en mémoire haute... -_-'
On vas donc aller voir à la main.

Taper : `M F3400` (et remonter jusqu'à apercevoir la vrai fin des données, en l'occurrence en **$F33E0** on a quelque chose)



On a donc une des données (qui colle au passage EXACTEMENT à ce que l'on a noté de la fin des DATA du Screen © 2)
en **$F33FF** (plus exactement en $F33EF mais bon, soyons prudent, on prefere en garder un peu plus que pas assez)

Ce qui nous donnes une taille final de : **$F33FF - $40000 = $B33FF**
Fichtre, on était pas loin :)

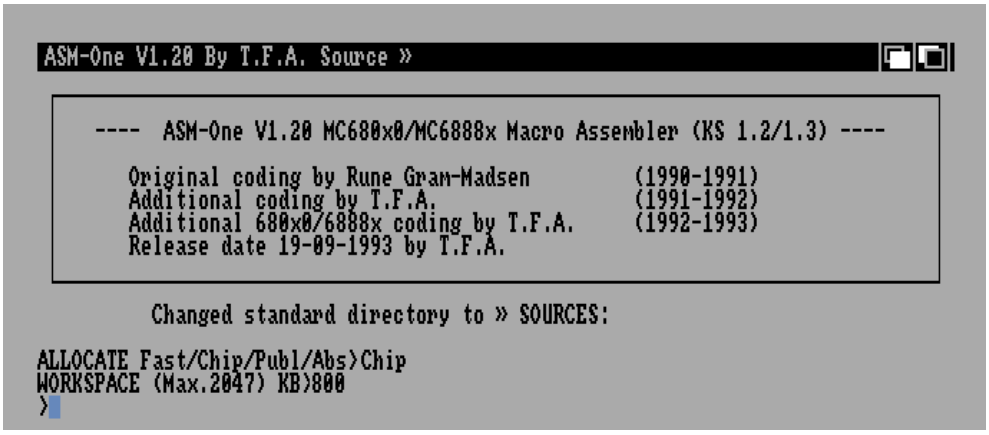
et $B33FF en décimal nous donnes : 734 207 Bytes soit environ **717 Ko**, on devrait pouvoir stocker ça sur une disquette standard Amiga :)

Inserez dans votre Amiga une disquette fraîchement formatée et **Tapez**

```
SM dump, 40000 F33FF
```

Loader Maison

On va reprendre exactement ce qui existe dans le tuto original d'AlphaOne à savoir, le *sectorLoader* de Rob Northen
Insérez votre disk **ASM One 1.20** ou son image adf dans votre **Amiga**
Une fois notre assembleur préféré chargé, **remplacer** la Disquette d'***ASM one*** par notre disquette de **Sauvegarde** en **DF0**
Celle contenant le fichier dump de sauvegarde.



Après avoir alloué environ **800 Ko** de **Chipmen**, **Tapez** le code suivant
Il est très bien commenté et ne nécessite à mon sens aucun addon d'explication.

```
; DISKIMAGE_RNC.s
; ###################################################################
; ## R-TYPE II (C) ACTIVISON                                       ##
; ## DISKIMAGE (133 TRACKS) FOR FLASHTRO TUTORIAL.                 ##
; ## !! USE CHIPMEM FOR THIS FOR WRITEBACK !!                      ##
; ## ------------------------------------------------------------- ##
; ## USAGE:                                                        ##
; ## 1. <A>  ASSEMBLE SOURCECODE                                   ##
; ## 2. <J>  EXECUTE SOURCE TO MAKE NECCESSARY PATCHES IN DISKIMAGE ##
; ## 3. <WT> WRITE BACK 133 TRACKS FROM LABEL #DISK_START          ##
; ## 4. <CC> CALCULATE BOOTBLOCK-CHECKSUM                          ##
; ## ------------------------------------------------------------- ##
; ###################################################################

DISKOFFSET_COPYLOCK_CODE       = 2*$1800+$200
DISKOFFSET_ORIGINAL_LOADER     = 2*$1800+$200+$F960-$500
DISKOFFSET_FILETABLE           = 2*$1800+$200+$FB20-$500

          ; OVERWRITE OLD COPYLOCK CODE WITH OUR NEW LOADER
          ; -----------------------------------------------

          LEA       NEW_COPYLOCK_CODE(PC),A0
          LEA       DISK_START+DISKOFFSET_COPYLOCK_CODE,A1
          MOVE.L    #(COPYLOCK_CODE_END-NEW_COPYLOCK_CODE)-1,D0
COPY_CL:  MOVE.B    (A0)+,(A1)+
          DBF       D0,COPY_CL

          ; OVERWRITE FIRST INSTRUCTION OF ORIGINAL LOADER WITH
          ; A "JMP $500" -> SO OUR LOADER IS EXECUTED!
          ; ------------------------------------------------------

          LEA       DISK_START+DISKOFFSET_ORIGINAL_LOADER,A0
          MOVE.W    #$4EF9,(A0)+
          MOVE.L    #$500,(A0)+

          ; THE FILETABLE IS LOCATED AT TRACK #2 SECTOR #0 ON DISK
          ; AND IS LOADED SEPERATELY BY THE ORIGINAL CODE. WE SKIP THIS PART
          ; AND COPY IT INTO THE RIGHT PLACE HERE ($FB20 IN MEMORY).
          ; ------------------------------------------------------------------

          LEA       DISK_START+2*$1800,A0                       ; TR #2 SEC #0
          LEA       DISK_START+DISKOFFSET_FILETABLE,A1
          MOVE.L    #$90,D0                                     ; SIZE OF FILETABLE
COPY_FT:  MOVE.B    (A0)+,(A1)+
          DBF       D0,COPY_FT

          MOVEQ     #0,D0                                       ; END OF PATCHING!!
          RTS
; ------------------------------------------------------------------------
```

```
; ========================================================================
; THE NEW TRACKLOADER
; -------------------
; HERE COMES OUR NEW LOADER THAT  IS WRITTEN TO THE PLACE OF THE
; ORGINAL COPYLOCK, IN MEMORY AT $500, ON DISK LOCATED ON TRACK #2, SECTOR #1
; ========================================================================


NEW_COPYLOCK_CODE:                                          ; <- $500 IN MEMORY !

          MOVEQ     #0,D1                                   ; HANDLE THE FILES LIKE THE ORIGINAL
          MOVEQ     #0,D2
          ASL.W     #3,D0
          LEA       $FB20,A1
          MOVE.B    (A1,D0.W),D1                            ; D1 = TRACKNUM
          MULS      #$1800,D1                               ; * TRACKSIZE $1800
          MOVE.B    1(A1,D0.W),D2                           ; D2 = SECTORNUM
          MULS.W    #512,D2                                 ; * SECTORSIZE $200
          ADD.L     D2,D1                                   ; GET CORRECT BYTEOFFSET
          DIVS.W    #512,D1                                 ; AND TURN IT INTO AN SECTOROFFSET
          MOVE.W    2(A1,D0.W),D2                           ; GET FILESIZE IN SECTORS
          ADDQ.W    #1,D2                                   ;
          MOVEQ     #0,D3
          MOVE.L    4(A1,D0.W),-(A7)                        ; SAVE REAL BYTESIZE OF FILE ON STACK
          LEA       $2FD80,A1                               ; MFM POINTER
AGAIN:    MOVEQ     #0,D0                                   ; DRIVENUMBER (DF0:)
          BSR.W     TLOADER                                 ; CALL RNC TRACKLOADER
          TST.B     D0                                      ; LOADING OK?
          BEQ.B     LGUT                                    ; YEAH -> END TLOADER
BLINKER:BTST        #6,$BFE001                              ;
          BNE.B     BLINK                                   ; OTHERWISE MAKE SCREEN FLASH
          BRA.B     AGAIN                                   ; AND WAIT FOR LEFT MOUSE TO
BLINK:    MOVE.W    D0,$DFF180                              ; REPEAT PROCEDURE :P
          SUBQ.W    #1,D0                                   ;
          BRA.B     BLINKER                                 ;
LGUT:     MOVE.L    (A7)+,D0                                ; RESTORE BYTESIZE OF FILE
          RTS                                               ; FOR FURTHER GAME-USAGE

TLOADER:  INCBIN    "RNCLOADER.BIN"                         ; ROB NORTHEN SECTORLOADER

COPYLOCK_CODE_END:

; ==<DISKIMAGE>==========================================================

DISK_START:

          DC.B      "DOS",0                                 ; TRACK 0 (BOOT)
          DC.L      0
          DC.L      0

          MOVE.W    #$2,$1C(A1)                             ; READTRACK
          MOVE.L    #2*$1800+$200,$2C(A1)                   ; DISKPOSITION
          MOVE.L    #$24400,$24(A1)                         ; SIZE
          MOVE.L    #$50000,$28(A1)                         ; ADDRESS
          JSR       -$1C8(A6)                               ; DOIO()

          LEA       COPY_C(PC),A0                           ; COPY NEXT PART OF THE
          LEA       $78000,A1                               ; BOOTCODE TO $78000
          MOVE.L    #(BOOTEND-COPY_C)-1,D0                  ; SO THAT WE DONT OVERWRITE
COPY:     MOVE.B    (A0)+,(A1)+                             ; OUR OWN BOOTCODE WHEN
          DBF       D0,COPY                                 ; COPYING GAMEDATA TO $500 !!

          MOVE.W    #$7FFF,D0                               ; KILL DMA/INTERRUPTS
          MOVE.W    D0,$DFF096
          MOVE.W    D0,$DFF09A
          MOVE.L    #$78000,$80.W                           ; CONTINUE AT $78000 IN
          TRAP      #0                                      ; SUPERVISOR MODE

COPY_C:   LEA       $50000,A0                               ; COPY OUR GAMEDATA DOWN
          LEA       $500.W,A1                               ; FROM $50000 TO $500 !!
          MOVE.L    #$24400,D0
COPY_D:   MOVE.B    (A0)+,(A1)+
          MOVE.W    D0,$DFF180
          SUBQ.L    #1,D0
          BNE.B     COPY_D

          MOVE.L    #$4E714E71,$E86.W                       ; NOP OUT LOADING OF FILETABLE
          MOVE.W    #$4E71,$E8A.W                           ;

          MOVE.L    #$AE3B9CE3,D0                           ; MOVE COPYLOCK KEY TO D0
          MOVE.L    D0,$60.W                                ; AND TO ALL MEMORYLOCATIONS
          MOVE.L    D0,$E2C.W
          MOVE.L    D0,$7FF0C
          MOVE.L    D0,$7FF24
          MOVE.L    D0,$7FFBA
          JMP       $E2A.W                                  ; NO NEED FOR A JMP TO $500.
                                                            ; WE GO DIRECTLY TO $E2A!

BOOTEND:BLK.B       $1800-(BOOTEND-DISK_START),0            ; ZEROES REST OF LONGTRACK 0

          BLK.B     $1800,0                                 ; TRACK 1 (EMPTY)

DUMP:     INCBIN    "DUMP"                                  ; TRACK 2 - END

          BLK.B     2577,0                                  ; ZERO TO GET AN EVEN DISKSIZE
                                                            ; 133 DOS TRACKS IN ALL

DISK_END:
```

Avant toute exécution, sauvegarder le code sur votre disquette de sauvegarde.
Celle-ci doit maintenant contenir le fichier **dump**, le fichier **DISKIMAGE_RNC.s** et bien sur le binaire **rncloader.bin** du *sectorloader*

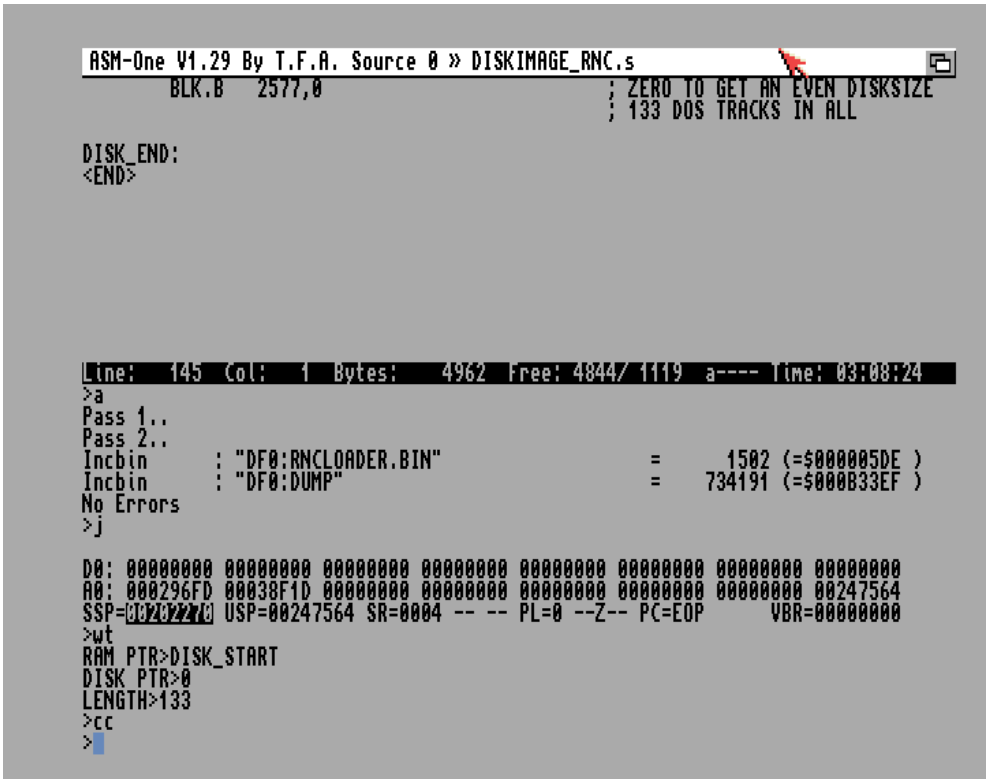
Basculez en mode **'command line'** en appuyant sur **ESC** puis compilez le tout à l'aide de la commande **A**,
puis exécutez le code avec la commande **J**

Insérez ensuite une nouvelle disquette vierge et **Taper :**

```
WT
RAM PTR>DISK_START
DISK PTR>0
LENGTH>133
```

Sans oublier la commande pour calculer le nouveau checksum du bootblock (sinon cela ne bootera pas)

```
CC
```



Une fois le disk écrie, **redémarrer** votre Amiga avec cette même disquette.
Apprécier le jeu.

Binaire du SectorLoader 'rncloader.bin' de Rob Northen

Pour ceux qui n'ont pas réussi à trouver le binaire **rncloader.bin** de *Rob Northen*, voilà la version hexa

```
00000000  48 E7 7F FC 4E 56 FF DC B6 7C 00 03 66 08 61 00  Hç.üNVÿÜ¶|..f.a.
00000010  00 DC 60 00 00 CE 38 00 02 44 00 03 3D 44 FF DC  .Ü`..Î8..D..=DÿÜ
00000020  3D 41 FF DE 3D 42 FF E0 3D 43 FF E2 2D 48 FF E4  =AÿÞ=Bÿà=Cÿâ-Hÿä
00000030  2D 49 FF E8 E4 58 02 40 00 01 52 40 3D 40 FF EC  -Iÿèä X.@..R@=@ÿì
00000040  70 00 36 02 67 72 70 1E D6 41 B6 7C 06 E0 6E 00  p.6.grp.ÖA¶|.àn.
00000050  00 92 02 81 00 00 FF FF 82 FC 00 0B 0C 6E 00 01  .'....ÿÿ,ü...n..
00000060  FF EC 67 02 D2 41 3D 41 FF EE 48 41 3D 41 FF F0  ÿìg.ÒA=AÿîHA=Aÿð
00000070  61 00 04 32 30 2E FF F0 72 0B 92 40 B2 6E FF E0  a..20.ÿðr.'@²nÿà
00000080  6F 04 32 2E FF E0 3D 41 FF F2 61 00 00 BC 66 28  o.2.ÿà=Aÿòa..¼f(
00000090  30 2E FF E0 90 6E FF F2 67 1E 3D 40 FF E0 30 2E  0.ÿà.nÿòg.=@ÿà0.
000000A0  FF F2 E1 88 D0 80 D1 AE FF E4 42 6E FF F0 30 2E  ÿòá^Ð€Ñ®ÿäBnÿð0.
000000B0  FF EC D1 6E FF EE 60 BC 2F 00 61 00 03 BA 20 1F  ÿìÑnÿî`¼/.a..° .
000000C0  67 20 72 00 32 2E FF EE 0C 6E 00 01 FF EC 67 02  g r.2.ÿî.n..ÿìg.
000000D0  E2 49 C2 FC 00 0B D2 6E FF F0 D2 6E FF FA 2F 41  âIÂü..ÒnÿðÒnÿú/A
000000E0  00 28 4E 5E 4A 80 4C DF 3F FE 4E 75 3D 7C 00 03  .(N^J€Lß?þNu=|..
000000F0  FF DC 7A 01 61 00 03 AE 61 00 03 8A 38 2E FF DC  ÿÜz.a..®a..Š8.ÿÜ
00000100  56 44 61 16 B0 BC FF FF FF FF 66 04 57 44 09 C5  VDa.°¼ÿÿÿÿf.WD.Å
00000110  53 6E FF DC 66 DE 20 05 4E 75 76 1F 70 00 61 0A  SnÿÜfÞ .Nuv.p.a.
00000120  D5 02 D1 80 51 CB FF F8 4E 75 72 FF 09 81 13 C1  Õ.Ñ€QËÿøNurÿ...Á
00000130  00 BF D1 00 08 39 00 05 00 BF E0 01 57 C2 09 C1  .¿Ñ..9...¿à.WÂ.Á
00000140  13 C1 00 BF D1 00 4E 75 78 02 42 6E FF FC 42 6E  .Á.¿Ñ.Nux.BnÿüBn
00000150  FF FA 42 6E FF F8 34 2E FF EE 61 00 03 60 66 00  ÿúBnÿø4.ÿîa..`f.
00000160  00 9E 70 1D 08 39 00 02 00 BF E0 01 67 00 00 90  .žp..9...¿à.g...
00000170  2A 6E FF E8 4B ED 04 00 2A BC AA AA AA AA 3B 7C  *nÿèKí..*¼ªªªª;|
00000180  44 89 00 04 61 00 01 D0 61 00 00 94 66 70 30 2E  D‰..a..Ða..”fp0.
00000190  FF F4 67 40 C0 FC 04 40 41 ED 00 06 61 00 02 62  ÿôg@Àü.@Aí..a..b
000001A0  49 F9 00 DF F0 1E 61 00 00 CE 66 6A 30 2E FF FA  Iù.ßð.a..Îfj0.ÿú
000001B0  90 6E FF F2 67 64 2A 6E FF E8 4B ED 04 00 30 2E  .nÿògd*nÿèKí..0.
000001C0  FF F4 C0 FC 04 40 DB C0 2A BC AA AA AA AA 3B 7C  ÿôÀü.@ÛÀ*¼ªªªª;|
000001D0  44 89 00 04 30 2E FF F6 67 18 C0 FC 04 40 41 ED  D‰..0.ÿög.Àü.@Aí
000001E0  00 06 61 00 02 1C 49 EE FF FE 42 54 61 00 00 88  ..a...IîÿþBTa..ˆ
000001F0  66 24 30 2E FF FA 90 6E FF F2 67 1E 70 1A 2F 00  f$0.ÿú.nÿòg.p./.
00000200  74 02 61 00 02 B8 61 00 03 02 20 1F 08 39 00 02  t.a..¸a... ..9..
00000210  00 BF E0 01 67 04 51 CC FF 32 60 00 02 46 74 0A  .¿à.g.QÌÿ2`..Ft.
00000220  41 ED 00 06 30 3C 00 40 61 00 01 D6 61 00 02 10  Aí..0<.@a..Öa...
00000230  66 36 61 00 01 6A 67 06 51 CA FF E6 60 2C 61 00  f6a..jg.QÊÿæ`,a.
00000240  01 28 66 2A B2 6E FF EE 66 24 B4 3C 00 0B 6C 1E  .(f*²nÿîf$´<..l.
00000250  B6 3C 00 0B 6E 18 53 03 3D 43 FF F4 3D 7C 00 0B  ¶<..n.S.=Cÿô=|..
00000260  FF F6 97 6E FF F6 70 00 4E 75 70 18 4E 75 70 1B  ÿö—nÿöp.Nup.Nup.
00000270  4E 75 70 19 4E 75 2A 6E FF E8 4B ED 04 00 30 2E  Nup.Nu*nÿèKí..0.
00000280  FF F8 C0 FC 04 40 DB C0 20 3C 00 00 17 70 61 00  ÿøÀü.@ÛÀ <...pa.
00000290  03 2C 08 2C 00 01 00 01 66 00 00 94 61 00 03 10  .,.,....f..”a...
000002A0  67 00 00 90 4A AD 04 40 67 E8 61 00 00 F2 66 BA  g...J..@gèa..òf°
000002B0  61 00 00 B6 66 B8 B2 6E FF EE 66 B2 36 02 B6 6E  a..¶f¸²nÿîf²6.¶n
000002C0  FF F0 6D 5C 30 2E FF F2 D0 6E FF F0 B6 40 6C 50  ÿðm\0.ÿòÐnÿð¶@lP
000002D0  08 2C 00 01 00 01 66 56 30 2E FF FC 07 00 66 40  .,....fV0.ÿü..f@
000002E0  41 ED 00 40 32 3C 04 00 61 00 00 C8 2F 00 41 ED  Aí.@2<..a..È/.Aí
000002F0  00 38 61 00 00 94 B0 9F 66 00 FF 78 08 2C 00 01  .8a..”°Ÿf.ÿx.,..
00000300  00 01 66 2A 61 30 41 ED 00 40 22 6E FF E4 D3 C1  ..f*a0Aí.@"nÿäÓÁ
00000310  61 00 00 C6 61 30 30 2E FF FA B0 6E FF F2 67 0E  a..Æa00.ÿú°nÿòg.
00000320  52 6E FF F8 0C 6E 00 0B FF F8 66 00 FF 4A 70 00  Rnÿø.n..ÿøf.ÿJp.
00000330  4E 75 70 FF 4E 75 22 03 92 6E FF F0 20 3C 00 00  NupÿNu".'nÿð <..
00000340  02 00 C2 C0 4E 75 30 2E FF FC 07 C0 3D 40 FF FC  ..ÂÀNu0.ÿü.À=@ÿü
00000350  52 6E FF FA 4E 75 20 4D 72 0A 70 00 41 E8 04 40  RnÿúNu Mr.p.Aè.@
00000360  20 80 51 C9 FF F8 4E 75 41 ED 00 08 61 1A 36 00   €QÉÿøNuAí..a.6.
00000370  02 43 00 FF 34 00 E0 4A 48 40 32 00 02 41 00 FF  .C.ÿ4.àJH@2..A.ÿ
00000380  E0 48 B0 3C 00 FF 4E 75 20 18 22 18 02 80 55 55  àH°<.ÿNu ."..€UU
00000390  55 55 02 81 55 55 55 55 D0 80 80 81 4E 75 61 0C  UU..UUUUÐ€€.Nua.
000003A0  2F 00 41 ED 00 30 61 E0 B0 9F 4E 75 41 ED 00 08  /.Aí.0aà°ŸNuAí..
000003B0  72 28 2F 02 E4 49 53 41 70 00 24 18 B5 80 51 C9  r(/.äISAp.$.µ€QÉ
000003C0  FF FA 24 1F 02 80 55 55 55 55 4E 75 08 39 00 06  ÿú$..€UUUUNu.9..
000003D0  00 DF F0 02 66 F6 4E 75 48 E7 F0 E0 70 7F 45 E8  .ßð.föNuHçðàp.Eè
000003E0  02 00 26 3C 55 55 55 55 22 18 24 1A C2 83 C4 83  ..&<UUUU".$.ÂƒÄƒ
000003F0  D2 81 82 82 22 C1 51 C8 FF F0 4C DF 07 0F 4E 75  Ò.,,"ÁQÈÿðLß..Nu
00000400  43 F9 00 DF F0 00 33 7C 40 00 00 24 33 7C 82 10  Cù.ßð.3|@..$3|,.
00000410  00 96 33 7C 66 00 00 9E 33 7C 95 00 00 9E 33 7C  .-3|f..ž3|•..ž3|
00000420  44 89 00 7E 23 48 00 20 33 7C 00 02 00 9C E2 48  D‰.~#H. 3|...œâH
00000430  00 40 80 00 33 40 00 24 33 40 00 24 4E 75 43 F9  .@€.3@.$3@.$NuCù
00000440  00 DF F0 00 20 3C 00 00 09 C4 61 00 01 70 08 29  .ßð. <...Äa..p.)
00000450  00 01 00 1F 66 0A 61 00 01 56 66 F2 70 FF 60 02  ....f.a..Vfòpÿ`.
00000460  70 00 33 FC 00 02 00 DF F0 9C 33 FC 40 00 00 DF  p.3ü...ßðœ3ü@..ß
00000470  F0 24 4A 80 4E 75 33 FC 04 00 00 DF F0 9E 4A 6E  ð$J€Nu3ü...ßðžJn
00000480  FF E2 6A 1E 72 FF 13 C1 00 BF D1 00 30 2E FF DC  ÿâj.rÿ.Á.¿Ñ.0.ÿÜ
00000490  56 80 01 81 13 C1 00 BF D1 00 01 C1 13 C1 00 BF  V€...Á.¿Ñ..Á.Á.¿
000004A0  D1 00 4E 75 72 FF 13 C1 00 BF D1 00 08 81 00 07  Ñ.Nurÿ.Á.¿Ñ.....
000004B0  61 D4 20 3C 00 00 00 C8 60 00 00 E2 48 E7 30 00  aÔ <...È`..âHç0.
000004C0  26 02 61 00 00 9E 30 2E FF DC D0 40 41 FA 01 08  &.a..ž0.ÿÜÐ@Aú..
000004D0  30 30 00 00 6A 04 61 32 66 2A E2 48 E2 4A 72 01  00..j.a2f*âHâJr.
000004E0  94 40 67 0E 6A 04 72 FF 44 42 70 03 61 50 53 42  ”@g.j.rÿDBp.aPSB
000004F0  66 F8 30 2E FF DC D0 40 41 FA 00 DC 31 83 00 00  fø0.ÿÜÐ@Aú.Ü1ƒ..
00000500  61 60 70 00 4C DF 00 0C 4E 75 48 E7 20 00 74 55  a`p.Lß..NuHç .tU
00000510  08 39 00 04 00 BF E0 01 67 0E 70 03 72 FF 61 1E  .9...¿à.g.p.rÿa.
00000520  51 CA FF EE 70 1E 60 10 30 2E FF DC D0 40 41 FA  QÊÿîp.`.0.ÿÜÐ@Aú
00000530  00 A6 42 70 00 00 70 00 4C DF 00 04 4E 75 2F 00  .¦Bp..p.Lß..Nu/.
00000540  61 2A 4A 01 6B 04 08 80 00 01 08 80 00 00 13 C0  a*J.k..€...€...À
00000550  00 BF D1 00 08 C0 00 00 13 C0 00 BF D1 00 20 1F  .¿Ñ..À...À.¿Ñ. .
00000560  60 3A 61 08 13 C0 00 BF D1 00 4E 75 48 A7 60 00  `:a..À.¿Ñ.NuH§`.
00000570  30 2E FF DC 14 39 00 BF D1 00 00 02 00 7F 56 00  0.ÿÜ.9.¿Ñ.....V.
00000580  01 82 57 00 D0 40 32 3B 00 4E 08 01 00 00 67 04  .‚W.Ð@2;.N....g.
00000590  08 82 00 02 10 02 4C 9F 00 06 4E 75 61 1E 08 39  .‚....LŸ..Nua..9
000005A0  00 00 00 BF DE 00 66 F6 53 80 66 F0 4E 75 08 39  ...¿Þ.föS€fðNu.9
000005B0  00 00 00 BF DE 00 66 1C 53 80 67 18 13 FC 00 08  ...¿Þ.f.S€g..ü..
000005C0  00 BF DE 00 13 FC 00 CC 00 BF D4 00 13 FC 00 02  .¿Þ..ü.Ì.¿Ô..ü..
000005D0  00 BF D5 00 4E 75 FF FF FF FF FF FF FF FF        .¿Õ.Nuÿÿÿÿÿÿÿÿ
```