CPC +

RASTER SUR CPC+

Cette fois, on va privilégier les possesseurs de CPC+, car mis a part le B-ASIC, je crois qu'ils n'ont pas grand-chose à se mettre sous la dent.

J'entends déjà certaines remarques désobligeantes : « Quoi ? Encore des Rasters ? ». Eh oui, mais cette fois sur le CPC +, et on peut dire que 4 096 couleurs à la place de 27, cela fait une grosse différence !
Il est à noter que les fonctionnalités de l'Asic par rapport au Gate Array sont beaucoup plus diverses et plus complexes. Cela ne veut pas forcément dire que la programmation en est plus compliquée, bien au contraire, comme nous allons le voir.
Essayons d'énumérer les différences - non visibles extérieurement par l'utilisateur mais visibles par le programmeur - qui existent entre l'Asic et le Gate Array.
Tout d'abord, la différence majeure, et la plus importante à mes yeux, se situe dans l'accès aux registres de l'Asic qui s'effectue en écrivant ou en lisant dans la mémoire centrale et non pas dans la zone d'Entrées/Sorties du Z80 (comme se fait habituellement l'accès aux périphériques). Pour ceux qui ne saisissent pas ce que cela signifie, je vais essayer d'être plus clair et de donner un exemple concret, ce qui ne m'empêchera pas de présenter le côté technique.

COTE TECHNIQUE

Sur un CPC de l'ancienne génération, lorsque l'on désire programmer le Gate Array, il est nécessaire d'utiliser des instructions d'Entrées/Sorties (OUT pour écrire dans la zone d'Entrées/Sorties et IN pour y lire). Le Z80 ne se comporte pas de la même manière lorsqu'on lit/écrit avec des instructions d'Entrées/Sorties que lorsqu'on lit/écrit dans la mémoire centrale. Dans le cas d'une instruction d'adressage de la mémoire centrale, le Z80 positionne l'adresse à laquelle doit s'effectuer l'opération sur son bus (groupe de fils sur lesquels transitent des valeurs binaires qui sont les adresses, broches A0 à A15) et ensuite active sa broche Read/Write (Lecture/Ecriture pour les anglophobes) en fonction de l'opération désirée. Si c'est une lecture, il va prendre la valeur présente sur son bus de données et va la mettre dans un registre (celui désigné par l'instruction). Si c'est une écriture, il va positionner la valeur d'un registre (celui désigné par l'instruction) sur son bus de données. Tout cela se déroule en une micro-seconde et est donc très rapide. Pour connaître la durée totale de l'instruction, il faut aussi tenir compte du temps que prend le décodage de l'instruction (c'est-à-dire le temps durant lequel le Z80 lit et comprend l'instruction à exécuter). Cela peut durer entre 1 et 4 micro-secondes et dépend de différentes choses : les registres entrant en jeu (1 micro-seconde de plus pour IX et IY), le mode d'adressage (indirect, indirect indexé, etc.) et la taille de la donnée à traiter (8 bits ou 16 bits).
Une instruction de Lecture/Ecriture dans la zone d'Entrées/Sorties se déroule presque de la même manière, à deux différences près. En premier lieu, pour signaler que l'instruction intervient sur les Entrées/Sorties, le Z80 active en plus sa broche IORQ (Input Output Request ou bien Demande d'Entrées/Sorties), et ensuite la phase Lecture/Eécriture proprement dite dure plus longtemps.
Le Z80 insère une micro-seconde d'attente, cela s'explique par le fait qu'à l'époque de la fabrication du Z80, les périphériques étaient plus lents que les mémoires, ce qui nécessitait une attente plus longue durant les accès.
Il est aussi à noter que les instructions d'Entrées/Sorties sont beaucoup moins souples que des instructions d'adressage à la mémoire centrale, donc moins facilement optimisables.

TOUT CELA POUR VOUS DIRE QUE...

Si je me suis bien fait comprendre, vous réalisez en ce moment même que pour lire/écrire dans les registres de l'Asic nous n'avons plus qu'à utiliser les instructions LD à la place des instructions IN et OUT qui sont beaucoup trop lentes.

L'ASIC : BIEN SOUS TOUS RAPPORTS

Un des autres avantages de l'Asic, non visible par l'utilisateur mais rendant bien des services aux programmeurs, se situe dans le fait que les registres de l'Asic se trouvent tous (je dis bien tous) à des adresses différentes. Ils sont donc directement accessibles en une seule instruction, contrairement à certains registres (pour ne citer qu'eux) du Gate Array ou du CRT 6845, qu'il faut préalablement choisir par l'intermédiaire d'un registre de sélection (exemple: OUT &BC00 puis OUT &BD00 pour le CRT 6845), dans lequel on écrit le numéro de registre courant. De cet avantage découle la diminution du nombre d'instructions nécessaires à la programmation des registres Asic et l'optimisation s'en trouve facilitée.
Une fois que l'on a bien saisi les différences qu'implique la programmation de l'Asic, on va pouvoir songer à nos fameux Rasters en 4 096 couleurs. Pour comprendre le fonctionnement des registres de couleurs (je ne pense pas me tromper en disant que ce sont ceux-là qu'il faut utiliser pour faire des Rasters), il ne faut pas faire appel à nos connaissances sur le Gate Array, en se disant qu'avec telle valeur nous obtiendrons telle couleur. On ne doit pas considérer des valeurs exactes, mais des groupes de bits qui, une fois réunis, donnent une valeur.
Plus clairement, on ne peut pas directement savoir quelle couleur donnera la valeur 703.
Tout d'abord, il faut savoir que les registres de couleurs sont des registres 16 bits dont 12 bits sont utilisés. Ces 12 bits se découpent en trois groupes de 4 bits chacun. Ces trois groupes correspondent à chacune des composantes (rouge, vert, bleu) nécessaires pour obtenir une couleur ; plus la valeur sur 4 bits est élevée, plus la composante sera intense. Le mélange des trois composantes donne les couleurs intermédiaires.
Donc, pour donner un exemple simple, si nous mettons &0000 dans un registre de couleur, toutes les composantes sont au minimum : ce qui donne du noir ; si par contre nous mettons &0FFF (sur 12 bits utiles, ne pas oublier !), nous aurons toutes les composantes au maximum, lesquelles, une fois mélangées, donneront du blanc brillant.
Pour vérifier (est-ce bien nécessaire ?) le nombre de couleurs disponibles, il suffit de faire le calcul suivant : 2 élevé à la puissance 12, car nous avons 12 bits et 2 valeurs possibles pour chaque bit, le résultat est effectivement 4 096 (ouf ! je suis rassuré, on ne sait jamais, ils auraient pu se tromper chez Amstrad !).
L'Asic dispose de 16 registres de couleurs pour les encres, d'un registre pour la couleur du bord, ainsi que de 15 registres de couleurs pour les encres des sprites, il n'y a que

15 encres pour les sprites, car l'encre 0 correspond à l'encre transparente à laquelle on ne peut pas affecter de couleur (ce qui est logique, non ?).

AU BOULOT

Mis à part la possibilité d'ajouter un côté esthétique non négligeable à vos démos, le fait de disposer de beaucoup de couleurs vous permet de créer de nouveaux effets, comme par exemple donner une impression de profondeur dans les déplacements des Rasters. Cela n'a rien de très nouveau de voir des Rasters se promener sur l'écran, même si ceux-ci suivent une douce courbe sinusoïdale. C'est plat et terne. Ce que je vous propose comme programme d'illustration de cet article, ce sont des Rasters qui se déplacent à la fois sur l'axe Y (de bas en haut ou de haut en bas, c'est comme vous voulez...) et en même temps sur un axe Z (pour la profondeur). La gestion de l'axe Y n'est pas une nouveauté, il nous suffit d'utiliser une table de cosinus pour le déplacement. Si vous avez régulièrement suivi les articles Logon, je pense que ce genre de chose ne doit pas vous poser de problèmes.
Pour ce qui est de l'axe Z, c'est une autre paire de manches. Il faut tout d'abord songer à l'effet que l'on désire obtenir. Dans notre cas, nous souhaitons avoir un Raster qui donne l'impression de se rapprocher ou de s'éloigner de l'écran. Lorsqu'il s'éloigne, le Raster est sombre et plus petit, par contre, lorsqu'il se rapproche, il est plus grand et plus clair.
Nous prévoyons en mémoire une zone dans laquelle seront stockées différentes tables de couleurs qui correspondent aux différents niveaux de profondeur visible.
Dans notre programme d'exemple, il y a 16 niveaux de profondeur.

VOUS SUIVEZ ?

Pour savoir quel Raster choisir à un instant précis, nous gérons une table de cosinus de 16 valeurs d'amplitude et dont l'indice sera lié à celui de la table des cosinus du déplacement sur l'axe des Y. Etant donné que nous utilisons un axe Z, il devient très simple de gérer plusieurs Rasters avec différentes priorités d'affichage en fonction de la position des Rasters sur l'axe Z.
En effet, un Raster éloigné sur l'axe Z doit être affiché « sous » un autre Raster plus proche.
En fait, il doit être affiché avant, ce qui nous fait comprendre que l'ordre d'affichage des Rasters dépend de leur position sur l'axe Z.

ENCORE DEUX MOTS

Pour déterminer l'ordre des Rasters avant l'affichage, il nous faudra préalablement les trier, et ceci en fonction de leur position sur l'axe Z. Il existe plusieurs manières de trier un tableau. On peut scruter les éléments les uns après les autres, en sortir le plus petit, et cela autant de fois qu'il y a d'éléments. On peut optimiser cette méthode en tenant compte des inversions durant la recherche du plus petit. S'il n'y a plus d'inversion entre les éléments, cela signifie que le tableau est entièrement trié, donc plus la peine de continuer.

La solution pour laquelle j'ai opté consiste à remplir un tableau avec la coordonnée Z de chaque Raster. Ensuite, on cherche le plus petit du tableau et on l'affiche.
Le plus petit trouvé doit être supprimé du tableau. On continue le tri jusqu'à que l'on soit sûr d'avoir affiché tous les Rasters.
Voici à peu près toutes les explications que je peux donner. Il ne vous reste plus qu'à taper les deux listings : le chargeur en Basic qui crée les tables et lance le programme binaire, et le listing en assembleur que vous sauvegarderez sous le nom RASTER+.BIN.

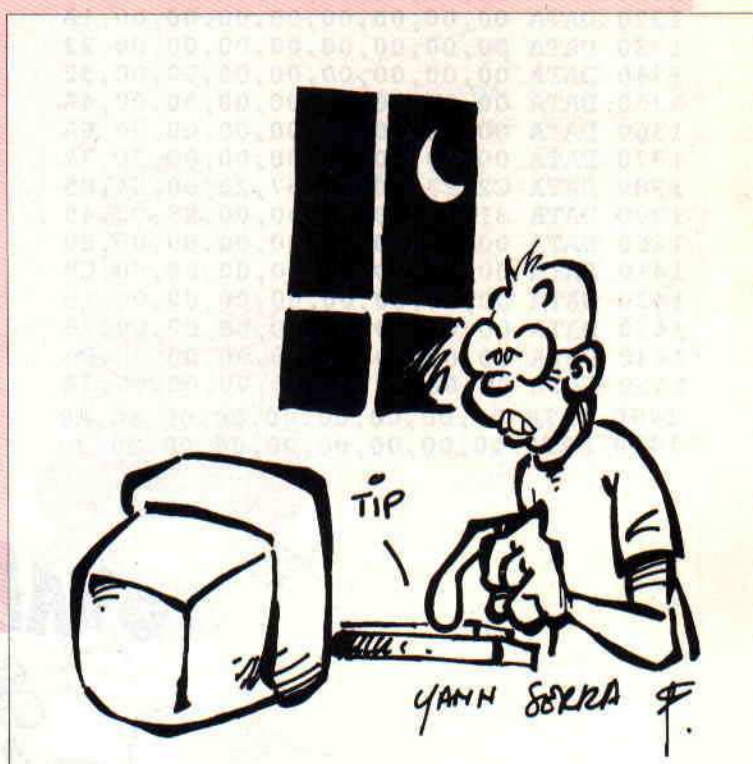
Pour les non-programmeurs, saisissez le lanceur Basic et les data. Lancez les datas qui généreront le fichier binaire et regardez le résultat en lancant le... lanceur.
- Et avec ça Madame ?
- Ce sera tout, merci.

Digit

```
10 MEMORY &7FFF:MODE 1:LOAD"raster+.bin"
:MODE 1:PRINT"Calculs en cours..."
20 DEG:AMP=(128-32)/2:AMPZ=14/2:adr=&900
0
30 FOR i=0 TO 359 STEP 360/256
40 POKE adr,AMP+AMP*COS(I):POKE adr+256,
1+AMPZ+AMPZ*COS(I):adr=adr+1
50 NEXT
60 gris=&9200:bleu=gris+&400:vert=gris+&
800:rouge=gris+&C00
70 FOR i=0 TO 15:FOR i1=0 TO 179 STEP 18
0/32
80 a=i*SIN(i1):POKE gris,INT(a)+INT(a)*1
6:POKE gris+1,INT(a):gris=gris+2
90 POKE bleu,INT(a):bleu=bleu+2
100 POKE vert+1,INT(a):vert=vert+2
110 POKE rouge,INT(a)*16:rouge=rouge+2
120 NEXT:NEXT
130 OUT &BC00,7:OUT &BD00,34:OUT &BC00,1
:OUT &BD00,50:OUT &BC00,2:OUT &BD00,50
140 MODE 1:CALL &8000
```


```
                ORG #8000

;****************************************
; DEFINITION DES CONSTANTES DU PROGRAMME
;****************************************
NBR_RASTER    EQU  4
TAILLE_RASTER EQU  32
NBR_LIGNE     EQU  128

;****************************************
; DEFINITION DES POINTEURS DU PROGRAMME
;****************************************

COSINUS1      EQU  #9000
COSINUS2      EQU  #9100
GRIS          EQU  #9200
BLEU          EQU  GRIS+#400
VERT          EQU  GRIS+#800
ROUGE         EQU  GRIS+#C00

;****************************************
; DEFINITION DES REGISTRES DE L'ASIC
;****************************************

INK0          EQU #6400
INK1          EQU INK0+2
INK2          EQU INK1+2
INK3          EQU INK2+2
INK4          EQU INK3+2
INK5          EQU INK4+2
INK6          EQU INK5+2
INK7          EQU INK6+2
INK8          EQU INK7+2
INK9          EQU INK8+2
INK10         EQU INK9+2
INK11         EQU INK10+2
INK12         EQU INK11+2
INK13         EQU INK12+2
INK14         EQU INK13+2
INK15         EQU INK14+2
BORDER        EQU #6420

;  PROGRAMABLE RASTER INTERRUPT
PRI           EQU #6800

;  SCREEN SPLIT SCAN LINE
SPLT          EQU #680

; SCREEN SPLIT SECONDARY START ADRESS
SSA           EQU #6802
```

```
;       SOFT SCROLL CONTROL REGISTER
SSCR        EQU #6804

;       INTERRUPT VECTOR
IVR         EQU #6805

;*************************************
; DEBUT DE LA PARTIE INSTRUCTIONS
;*************************************

INIT_ASIC   DI
            IM  1
            LD  BC,#7F84
            OUT (C),C
            LD  E,#11
            LD  HL,SEQUENCE
            LD  B,#BC
DELOCK      LD  C,(HL)
            OUT (C),C
            INC HL
            DEC E
            JR  NZ,DELOCK
            LD  BC,#7F88
            OUT (C),C

;       TOUTES LES ENCRES EN NOIR

            LD  HL,INK0
            LD  DE,INK0+1
            LD  BC,#3F
            LD  (HL),L
            LDIR
            LD  BC,#F782
            OUT (C),C
            LD  BC,#7F84
            OUT (C),C
            LD  BC,#7F88
            OUT (C),C
;       INITIALISE DIFFERENTS REGISTRES
            XOR A
            LD  (PRI),A
            LD  (SPLT),A
            LD  (SSCR),A
            LD  HL,#C9FB
            LD  (#38),HL
            LD  HL,CODE
            LD  DE,COPY
            LD  BC,NBR_LIGNE-1
            LDIR
            EI

SYNCHRO     LD  B,#F5
            IN  A,(C)
            RRA
            JP  NC,SYNCHRO+2
            DI
            LD  HL,#0000
            LD  (INK0),HL           COULEURS CODEES SUR 16 BITS !!!

            LD  IX,TAB_TRI
            LD  SP,TAB_PTR          TABLE DES POINTEURS DE COSINUS
            LD  C,NBR_RASTER

CAL_ADR     POP HL                  TABLE COSINUS POUR MOUVEMENT Y
            LD  E,(HL)
            LD  D,0
            INC L                   AUGMENTE INDICE COSINUS POUR COORD Y
            PUSH HL
            INC SP
            INC SP
            EX  DE,HL

            ADD HL,HL               MULTIPLIE PAR DEUX CAR VALEUR 16 BITS
            LD  DE,RASTER           DEBUT RASTER APPARAISSANT A L'ECRAN
            ADD HL,DE               POSITION DANS LA TABLE PRINCIPALE
            LD  (IX+1),L            RANGE POINTEUR DESTINATION
            LD  (IX+2),H

PTR_COS2    POP HL                  TABLE COSINUS POUR PROFONDEUR Z
            LD  A,(HL)
            INC L                   AUGMENTE INDICE COSINUS POUR COORD Z
            PUSH HL
            INC SP
            INC SP
            LD  (IX+0),A            RANGE COORDONNE Z
            LD  L,A
            LD  H,0
            ADD HL,HL               *2
            ADD HL,HL               *4
            ADD HL,HL               *8
            ADD HL,HL               *16
            ADD HL,HL               *32
            ADD HL,HL               *64  NOMBRE D'OCTETS PAR RASTER
            POP DE                  CHOIX DE LA TABLE DES TAILLES DU RASTER
            ADD HL,DE               HL POINTE SUR LE RASTER SOURCE

            LD  (IX+3),L            RANGE POINTEUR SOURCE
            LD  (IX+4),H
            LD  DE,5
            ADD IX,DE
            DEC C
            JP  NZ,CAL_ADR          ON RECOMMENCE POUR CHAQUE RASTER

            LD  B,NBR_RASTER
RE_TRI      LD  HL,TAB_TRI
            LD  C,NBR_RASTER-1      CAR ON PREND DEJA LE PREMIER
            LD  D,H
            LD  E,L                 POINTEUR AFFICHAGE=POINTEUR DEBUT
            LD  A,(HL)              PREND PREMIERE VALEUR Z
RE_COMP     INC HL
            INC HL
            INC HL
            INC HL
            CP  (HL)                COMPARE AVEC SUIVANT
            JP  C,COMP_SUIVANT      LE SUIVANT EST PLUS GRAND
            LD  A,(HL)              LE SUIVANT EST PLUS PETIT ON LE STOCKE
            LD  E,L                 ON MEMORISE LE POINTEUR DANS 'DE'
            LD  D,H
COMP_SUIVANT DEC C
            JP  NZ,RE_COMP

            EX  DE,HL               'HL'=POINTEUR MEMORISE
            LD  (HL),#FF            ON SUPPRIME POUR PROCHAIN TOUR
            INC HL
            LD  SP,HL
            EXX
            POP DE                  ON RECUPERE POINTEUR DESTINATION
            POP HL                  PUIS ON RECUPERE POINTEUR SOURCE
            LD  BC,TAILLE_RASTER*2
            LDIR                    ET ON COPIE DANS LA TABLE AFFICHEE
            EXX
            DJNZ RE_TRI
            LD  SP,#100
            EI
            HALT
            HALT                    ON LAISSE PASSER LES MICROS-SECONDES
            DEFS  32                POUR DEBUTER LA OU IL FAUT
            DI
            LD  SP,RASTER
            LD  A,NBR_LIGNE
BOUCLE1     POP HL                  3 NOPS EN TEMPS
            LD  (INK0),HL           5 "   "
            DEFS 64-12              POUR UN TOTAL DE 64 NOPS
            DEC A                   1 NOPS EN TEMPS
            JP  NZ,BOUCLE1          3 "   "
            LD  L,A
            LD  H,A
            LD  (INK0),HL
            LD  SP,NBR_LIGNE-1*2+RASTER  ON POINTE SUR LA FIN DE LA TABLE
            LD  HL,00
CODE        PUSH HL                 POUR EFFACER VITE FAIT
COPY        DEFS NBR_LIGNE-1        ON DELIMITE LA PLACE POUR LES AUTRES 'PUSH'
            LD  SP,#100             ON REMET 'SP' POUR PAS ETRE DERANGE
            EI                      DES NOUVEAUX LES INTERRUPTIONS
            JP  SYNCHRO             ET HOP LA !!

            ; LA FAMEUSE SEQUENCE (HA HA !!)
SEQUENCE    DB #FF,#00,#FF,#77,#B3,#51,#A8,#D4,#62,#39,#9C,#46,#2B,#15
            DB #8A,#CD,#EE,#3F

            ;TABLE PREDEFINIE DES POINTEURS DE COSINUS ET DE COULEURS
TAB_PTR     DEFW COSINUS1+192       POINTEUR POUR COORDONNEES Y
            DEFW COSINUS2           POINTEUR POUR COORDONNEES Z
            DEFW GRIS               COULEUR DU RASTER
            DEFW COSINUS1           ET AINSI DE SUITE
            DEFW COSINUS2+64
            DEFW BLEU

            DEFW COSINUS1+64
            DEFW COSINUS2+128
            DEFW VERT

            DEFW COSINUS1+128
            DEFW COSINUS2+192
            DEFW ROUGE

TAB_TRI     DEFS NBR_RASTER*5       ZONE UTILISEE POUR LE TRI

RASTER      DEFS 128*2              ZONE DE RASTER AFFICHE
```



```
100 'COPYRIGHT AMSTRAD 100 %
110 ADR= 32768:FOR I=0 TO 61
120 FOR J=1 TO 8:READ A$
130 A=VAL("&"+A$)
140 B=(B+I+A*J) AND 255
150 POKE ADR+I*8+J-1,A:NEXT J
160 READ B$:IF B=VAL("&"+B$) THEN 180
170 PRINT "ERREUR EN ";1000+I*10:STOP
180 NEXT I:SAVE"RASTER+.BIN",B,ADR,496
190 '
1000 DATA F3,ED,56,01,84,7F,ED,49,24
1010 DATA 1E,11,21,C5,81,06,BC,4E,20
1020 DATA ED,49,23,1D,20,F9,01,B8,C9
1030 DATA 7F,ED,49,21,00,64,11,01,70
1040 DATA 64,01,3F,00,75,ED,B0,01,62
1050 DATA 82,F7,ED,49,01,84,7F,ED,E3
1060 DATA 49,01,B8,7F,ED,49,AF,32,32
1070 DATA 00,68,32,01,68,32,04,68,64
1080 DATA 21,FB,C9,22,38,00,21,3E,8D
1090 DATA 81,11,3F,81,01,7F,00,ED,A0
1100 DATA B0,FB,06,F5,ED,78,1F,D2,56
1110 DATA 54,80,F3,21,00,00,22,00,4D
1120 DATA 64,DD,21,EF,81,31,D7,81,7E
1130 DATA 0E,04,E1,5E,16,00,2C,E5,E1
1140 DATA 33,33,EB,29,11,03,82,19,0C
1150 DATA DD,75,01,DD,74,02,E1,7E,29
1160 DATA 2C,E5,33,33,DD,77,00,6F,97
1170 DATA 26,00,29,29,29,29,29,29,8E
1180 DATA D1,19,DD,75,03,DD,74,04,15
1190 DATA 11,05,00,DD,19,0D,C2,6A,A5
1200 DATA 80,06,04,21,EF,81,0E,03,8C
1210 DATA 54,5D,7E,23,23,23,23,23,D6
1220 DATA BE,DA,B7,80,7E,5D,54,0D,75
1230 DATA C2,AB,80,EB,36,FF,23,F9,36
1240 DATA D9,D1,E1,01,40,00,ED,B0,53
1250 DATA D9,10,D8,31,00,01,FB,76,F3
1260 DATA 76,00,00,00,00,00,00,00,39
1270 DATA 00,00,00,00,00,00,00,00,11
1280 DATA 00,00,00,00,00,00,00,00,F1
1290 DATA 00,00,00,00,00,00,00,00,D9
1300 DATA 00,F3,31,03,82,3E,80,E1,D4
1310 DATA 22,00,64,00,00,00,00,00,1A
1320 DATA 00,00,00,00,00,00,00,00,1A
1330 DATA 00,00,00,00,00,00,00,00,22
1340 DATA 00,00,00,00,00,00,00,00,32
1350 DATA 00,00,00,00,00,00,00,00,4A
1360 DATA 00,00,00,00,00,00,00,00,6A
1370 DATA 00,00,00,00,00,00,00,3D,7A
1380 DATA C2,F7,80,6F,67,22,00,64,85
1390 DATA 31,01,83,21,00,00,E5,00,40
1400 DATA 00,00,00,00,00,00,00,00,80
1410 DATA 00,00,00,00,00,00,00,00,C8
1420 DATA 00,00,00,00,00,00,00,00,18
1430 DATA 00,00,00,00,00,00,00,00,70
1440 DATA 00,00,00,00,00,00,00,00,D0
1450 DATA 00,00,00,00,00,00,00,00,38
1460 DATA 00,00,00,00,00,00,00,00,A8
1470 DATA 00,00,00,00,00,00,00,00,20
1480 DATA 00,00,00,00,00,00,00,00,A0
1490 DATA 00,00,00,00,00,00,00,00,28
1500 DATA 00,00,00,00,00,00,00,00,B8
1510 DATA 00,00,00,00,00,00,00,00,50
1520 DATA 00,00,00,00,00,00,00,00,F0
1530 DATA 00,00,00,00,00,00,00,00,98
1540 DATA 00,00,00,00,00,00,00,00,48
1550 DATA 00,00,00,00,00,00,31,00,57
1560 DATA 01,FB,C3,52,80,FF,00,FF,11
1570 DATA 77,B3,51,A8,D4,62,39,9C,28
1580 DATA 46,2B,15,8A,CD,EE,3F,C0,49
1590 DATA 90,00,91,00,92,00,90,40,2E
1600 DATA 91,00,96,40,90,80,91,00,28
1610 DATA 9A,80,90,C0,91,00,9E,00,81
1620 DATA 00,00,00,00,00,00,00,00,71
1630 DATA 00,00,00,00,00,00,00,00,69
1640 DATA 00,00,00,00,00,00,00,00,69
1650 DATA 00,00,00,00,00,00,00,00,71
1660 DATA 00,00,00,00,00,00,00,00,81
1670 DATA 00,00,00,00,00,00,00,00,99
1680 DATA 00,00,00,00,00,00,00,00,B9
1690 DATA 00,00,00,00,00,00,00,00,E1
1700 DATA 00,00,00,00,00,00,00,00,11
1710 DATA 00,00,00,00,00,00,00,00,49
1720 DATA 00,00,00,00,00,00,00,00,89
1730 DATA 00,00,00,00,00,00,00,00,D1
1740 DATA 00,00,00,00,00,00,00,00,21
1750 DATA 00,00,00,00,00,00,00,00,79
1760 DATA 00,00,00,00,00,00,00,00,D9
1770 DATA 00,00,00,00,00,00,00,00,41
1780 DATA 00,00,00,00,00,00,00,00,B1
1790 DATA 00,00,00,00,00,00,00,00,29
1800 DATA 00,00,00,00,00,00,00,00,A9
1810 DATA 00,00,00,00,00,00,00,00,31
1820 DATA 00,00,00,00,00,00,00,00,C1
1830 DATA 00,00,00,00,00,00,00,00,59
1840 DATA 00,00,00,00,00,00,00,00,F9
1850 DATA 00,00,00,00,00,00,00,00,A1
1860 DATA 00,00,00,00,00,00,00,00,51
1870 DATA 00,00,00,00,00,00,00,00,09
1880 DATA 00,00,00,00,00,00,00,00,C9
1890 DATA 00,00,00,00,00,00,00,00,91
1900 DATA 00,00,00,00,00,00,00,00,61
1910 DATA 00,00,00,00,00,00,00,00,39
1920 DATA 00,00,00,00,00,00,00,00,19
1930 DATA 00,00,00,00,00,00,00,00,01
1940 DATA 00,00,00,00,00,00,00,00,F1
1950 DATA 00,00,00,00,00,00,00,00,E9
1960 DATA 00,00,00,00,00,00,00,00,E9
1970 DATA 00,00,00,00,00,00,00,00,F1
1980 DATA 00,00,00,00,00,00,00,00,01
1990 DATA 00,00,00,00,00,00,00,00,19
2000 DATA 00,00,00,00,00,00,00,00,39
2010 DATA 00,00,00,00,00,00,00,00,61
2020 DATA 00,00,00,00,00,00,00,00,91
2030 DATA 00,00,00,00,00,00,00,00,C9
2040 DATA 00,00,00,00,00,00,00,00,09
2050 DATA 00,00,00,00,00,00,00,00,51
2060 DATA 00,00,00,00,00,00,00,00,A1
```