COMO PODER LLENAR MAS MEMORIA RAM

Si hojea las páginas de Microhobby Amstrad comprobará que una gran parte de los listados de los programas que habitualmente publicamos son bastante cortos. Teniendo cerca de 40K de RAM libres para llenarlos de instrucciones Basic podemos decir que realmente utilizamos muy poca memoria de nuestro Amstrad cada vez que escribimos y ejecutamos uno de estos programas

¿Qué le parece la idea de poder almacenar varios programas Basic a la vez en la memoria? Y no sólo eso, sino que si además pudiéramos relacionar unos con otros y hacer que se ejecutara cualquiera de ellos... sería una maravilla.

Tendríamos la posibilidad de reunir en la memoria, y al mismo tiempo, todos los programas de la serie de Primeros Pasos, por ejemplo, de una semana. Podríamos seleccionar uno, ejecutarlo, seleccionar otro, ejecutarlo a su vez y comparar los resultados. Y también conseguiríamos evitar teclear repetidamente un listado o tener que acceder al disco o a la cinta con demasiada frecuencia.

Estas pudieran ser algunas de las múltiples aplicaciones que tendríamos al alcance de nuestra mano una vez que podamos disponer de esta facilidad. Nos abre todo un nuevo y extenso campo de posibilidades a la hora de utilizar nuestro ordenador.

Por ejemplo, podríamos emplear un programa para modificar otro. No nos sería nada difícil escribir una utilidad para resumir un programa, eliminando todos los REMs que hubiera en su listado. Otra cosa que se nos ocurre es hacer una de búsqueda y sustitución que nos permita cambiar los nombres de las variables, etc.

Hemos citado algunas ideas que se nos han ocurrido, pero seguro que en este momento estarán pasando por su mente muchas más.

El Programa que acompaña este artículo nos permitirá almacenar y ejecutar programas Basic en cualquier dirección de la memoria. Esto quiere decir que vamos a poder cargar un programa en la posición &1000, por ejemplo, otro en la &2000 y un tercero en la &3000. Y además nos será posible seleccionar uno de ellos y ejecutarle.

Tenemos la facilidad de elegir cualquier cifra como dirección de comienzo de carga pero precisamente por eso le aconsejamos que no lo haga alegremente. Escoja unas cantidades semejantes a &1000, que sean facilitas, ya que si así lo hacemos va a resultarnos relativamente simple mantenernos siempre al tanto de donde hemos colocado cada uno de nuestros programas.

Es necesario, también, que nos aseguremos de que al escribir un programa no lo hagamos sobre alguno anterior que esté colocado en una determinada dirección. Sin embargo, con tantos Ks de memoria RAM como dispone nuestro ordenador, no va a resultarnos difícil encontrar un espacio libre.

El Programa I es un listado en Basic de la utilidad y el Programa II es otro listado de la misma, pero esta vez en lenguaje ensamblador. Elija el que mejor le parezca.

Cuando los ejecute verá como se le añaden algunos nuevos comandos al Basic de su **Amstrad.** En la Tabla I le proporcionamos una lista de los mismos.

Analicemoslos con detenimiento. El primero es:

|PRINT.PAGE

Nos vas a imprimir el valor de una de las variables del sistema que hemos llamado PAGE. Está en la dirección &AE64 si posee el Basic 1.1, en la &AE81 si tiene el Basic 1.0.

El segundo que nos encontramos es:

|SET.PAGE, entero

y, como probablemente ya habrá adivinado, establece el valor que va a tener la variable PAGE.

Cuando introduzca en memoria un programa o lo cargue desde una cinta o un disco, el código generado por las instrucciones Basic se almacenará a partir de la posición siguiente a la indicada por PAGE. Si escribimos el comando SAVE, el ordenador nos va a salvar el programa que se encuentre en la memoria colocado donde nos indique PAGE. Y lo mismo ocurre si tecleamos RUN. Lo que se nos va a ejecutar es el señalado por la variable PAGE.

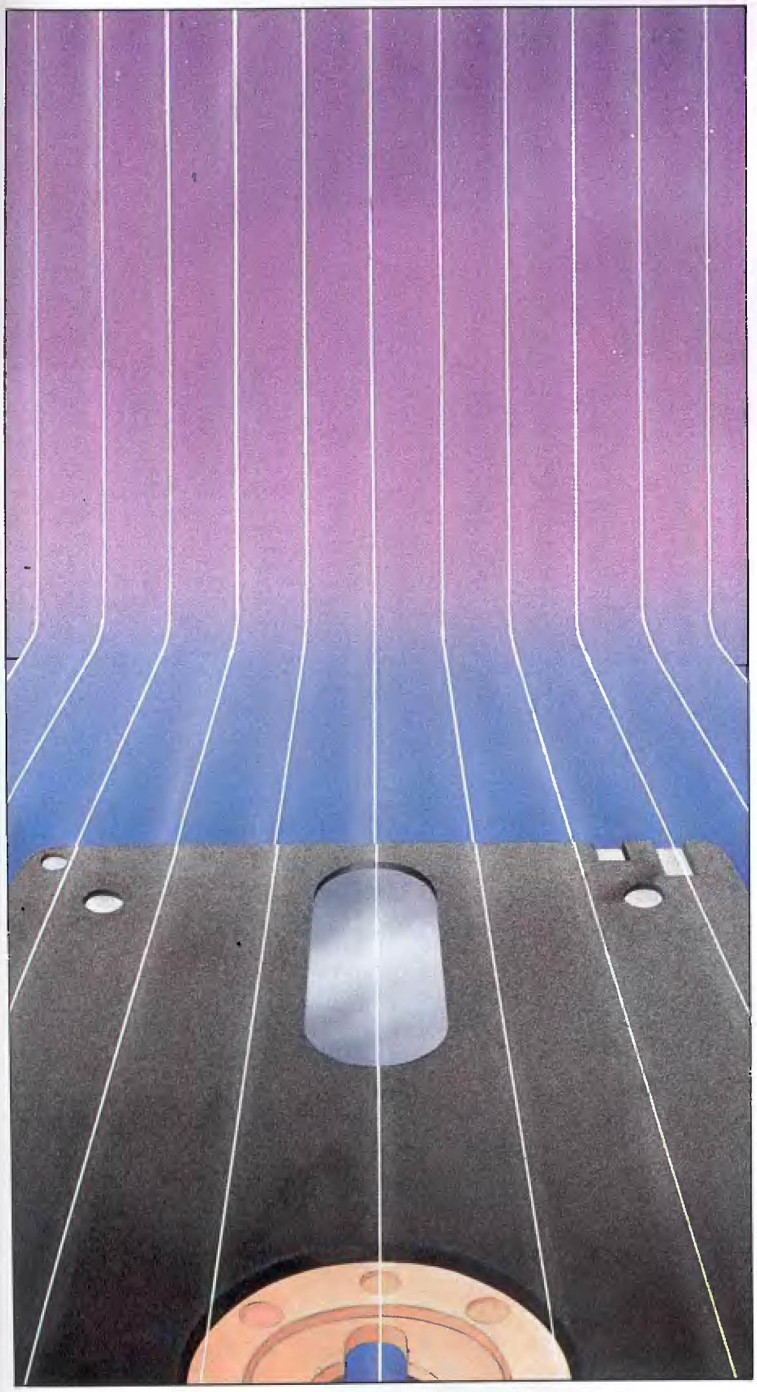
Si cambiamos su valor, podemos meter en memoria otro programa sin alterar el que ya teníamos en la misma siempre que, naturalmente, no lo escribamos encima.

No es cuestión más que de cambiar una de las variables del sistema. Si queremos volver a nuestro programa primitivo y devolver a page su valor original, no podremos hacerlo mágicamente. Es necesario que se lo indiquemos correctamente al ordenador.

Además de conocer dónde comienza un programa, en Basic también puede sernos necesario saber dónde termina. Esta dirección almacena, igualmente, en otra de las variables del sistema a la que llamamos TOP.

El Basic necesita conocer su valor, ya que a partir de esta posición es donde se almacenan las variables cuando el programa se está ejecutando.

También van a ser muy importantes otro tipo de apuntadores. Son los que le van a decir al interpretador Basic el punto donde termina la zona de memoria destinada a contener las distin-



tas variables que, como anteriormente dijimos, van a estar situadas al final del programa a partir de la dirección señalada por TOP.

Al resultar todos estos apuntadores, restablecemos al mismo tiempo el programa original, que podemos ejecutar otra vez.

Las variables, sin embargo, se perderán, y le aconsejamos que teclee CLEAR siempre que cambie el valor de PAGE.

La utilidad realiza todo esto automáticamente. Así que si alteramos PAGE, la rutina examina si hay almacenado algún programa en la nueva dirección. Si no hay ninguno, o no podemos elegir el programa existente debido a que lo hayamos «manchado», se ejecuta inmediatamente el comando:

INEW.PROGRAM

Esta orden suprimirá el programa que haya en la dirección señalada por PAGE sin destruir ninguno de los otros que comparten en ese momento la memoria. La instrucción Basic NEW limpia todo lo que haya en la misma, así que utilícela con cuidado.

Un programa puede necesitar conocer en un determinado momento que PAGE, TOP y LOMEM tiene. Veamos qué es esta última palabra: LOMEM es la dirección más baja de la zona de memoria que está libre. ¿Correcto?

Pero, ¿cómo vamos a saber el valor de cada una de estas tres variables? La utilidad PAGE nos va a proporcionar tres nuevos comandos para que lo podamos hacer con toda facilidad.

IGET.PAGE
IGET.TOP

y

IGET.LOMEM

toman el valor actual de cada una de estas variables del sistema y lo coloca en una, definida como entera, del lenguaje Basic.

Si hacemos:

a%=0

y a continuación:

IGET.PAGE,¶a%

la variable entera contendrá ahora el valor que tiene en este momento PAGE.

Podemos hacer los mismo para TOP y LOMEM. ¿Se atreve?

''PAGE'' es una utilidad extremadamente usada y de una gran ayuda a la hora de programar. Nos permite conseguir todas las ventajas de la generosa cantidad de memoria RAM que posee nuestro **Amstrad**.

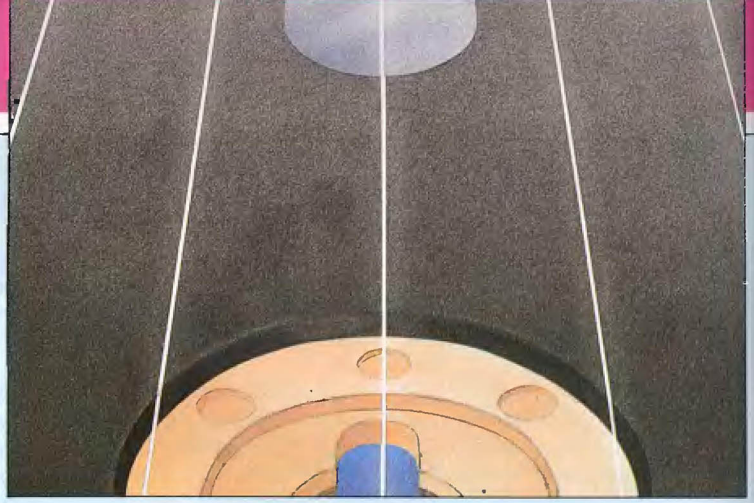
Anímese. Coja una de nuestras revistas y seleccione varios programas pequeños. Cargue cada uno de ellos en una posición seleccionada por el comando:

ISET.PAGE

siguiendo las indicaciones que le hemos dado y se encontrará con que la memoria de su ordenador está compartida por todos ellos y utilizada con un rendimiento bastante mayor. ¡Suerte!

```
IPRINT.PAGE
ISET.PAGE,integer
IGET.PAGE,¶variable%
IGET TOP,¶variable%
IGET.LOMEM,¶variable%
INEW.PROGRAM
```

Tabla 1. Comandos nuevos



```
10          ORG  #A000
20 ;
30 ;***** INICIALIZAR RSX *****
40 ;
50          LD   HL,flags
60          BIT  1,(HL)
70          RET  NZ
80          SET  1,(HL)
90          LD   BC,tablasal
100         LD   HL,workspace
110         CALL #BCD1
120         CALL #B900
130         PUSH AF
140         LD   A,(#C002)
150         AND  A
160         JR   Z,inicializar
170         LD   HL,#AE64
180         LD   (PAGIN1),HL
190         LD   (PAGIN2),HL
200         LD   (PAGIN3),HL
210         LD   (PAGIN4),HL
220         LD   HL,#AE17
230         LD   (DATO1),HL
240         LD   (DATO2),HL
250         LD   HL,#AE5E
260         LD   (HIMEM),HL
270         LD   HL,#AE66
280         LD   (SUP1),HL
290         LD   (SUP3),HL
300         LD   HL,#AE68
310         LD   (SUP2),HL
320         LD   HL,#AE6A
330         LD   (ARRAY),HL
340         LD   HL,#AE6C
350         LD   (MEMAL1),HL
360         LD   (MEMAL2),HL
370         LD   HL,#AE1D
380         LD   (EJELIN),HL
390 inicializar:
400         POP  AF
410         CALL #B90C
420         CALL cadena
430         DEFB "U","t","i"
440         DEFB "l","i","d"
450         DEFB "a","d"," "
460         DEFB "P","A","G"
470         DEFB "E"," ","a"
480         DEFB "c","t","i"
490         DEFB "v","a","d"
500         DEFB "a",".",7
510         DEFB 13,10,0
520         RET
530 ;
540 ;* DAR NUEVO VALOR A PAGE *
550 ;
560 pagina:
570         DEC  A
580         JP   NZ,errorpar
590         CALL chequeo
600         LD   E,(IX+0)
610         LD   D,(IX+1)
620         LD   HL,(#AE7B)
630 HIMEM:  EQU  $-2
640         DEC  H
650         PUSH HL
660         AND  A
670         SBC  HL,DE
680         JP   C,nositio
690         LD   HL,#016C
700         SBC  HL,DE
710         EX   DE,HL
720         POP  DE
730         JP   NC,nopuedo
740         LD   (#AE81),HL
750 PAGIN1: EQU  $-2
760         LD   (#AE30),HL
770 DATO1:  EQU  $-2
780         LD   A,(HL)
790         AND  A
800         JR   NZ,nueva
810         INC  HL
820 linueva:
830         LD   C,(HL)
840         INC  HL
850         LD   B,(HL)
860         LD   A,B
870         AND  A
880         JR   NZ,nueva
890         OR   C
900         JR   Z,varpointer
910         DEC  HL
920         ADD  HL,BC
930         PUSH HL
940         AND  A
950         SBC  HL,DE
960         POP  HL
970         JR   C,linueva
980 ;
990 ;********** NEW **********
1000 ;
1010 nueva:
1020        LD   HL,(#AE81)
1030 PAGIN2: EQU $-2
1040        LD   (#AE30),HL
1050 DATO2: EQU  $-2
1060        LD   (HL),#00
1070        INC  HL
1080        LD   (HL),#00
1090        INC  HL
1100        LD   (HL),#00
1110 varpointer:
1120        INC  HL
1130        LD   (#AE83),HL
1140 SUP1:  EQU  $-2
1150        LD   (#AE85),HL
1160 SUP2:  EQU  $-2
1170        LD   (#AE87),HL
1180 ARRAY: EQU  $-2
1190        LD   (#AE89),HL
1200 MEMAL1: EQU $-2
1210        CALL cadena
1220        DEFB "o","k",13
1230        DEFB 10,7,0
1240        RET
1250 ;
1260 ;* PONER PAGE EN VARIABLE *
1270 ;
1280 cogerpag:
1290        DEC  A
1300        JP   NZ,errorpar
1310        LD   DE,(#AE81)
1320 PAGIN4: EQU $-2
1330 cp:
1340        LD   L,(IX+0)
1350        LD   H,(IX+1)
1360        LD   (HL),E
1370        INC  HL
1380        LD   (HL),D
1390        RET
1400 ;
1410 ;* PONER TOP EN VARIABLE *
1420 ;
1430 superior:
1440        DEC  A
1450        JP   NZ,errorpar
1460        LD   DE,(#AE83)
1470 SUP3:  EQU  $-2
1480        JR   cp
1490 ;
1500 ;* PONER LOMEM EN VARIABLE *
1510 lomem:
1520        DEC  A
1530        JP   NZ,errorpar
1540        LD   DE,(#AE89)
1550 MEMAL2: EQU $-2
1560        JR   cp
1570 ;
1580 ;***** ESCRIBIR CADENA *****
1590 ;
1600 cadena:
1610        POP  HL
1620 sp1:
1630        LD   A,(HL)
1640        CALL #BB5A
1650        INC  HL
1660        OR   A
1670        JR   NZ,sp1
1680        JP   (HL)
1690 ;
1700 ;***** ESCRIBIR PAGE *****
1710 ;
1720 escribir:
1730        LD   A,#26
1740        CALL #BB5A
1750        LD   HL,(#AE81)
1760 PAGIN3: EQU $-2
1770 ;
1780 ;** ESCRIBIR PALABRA HEX **
1790 ;
1800 hexpal: LD  A,H
1810        CALL hex
1820        LD   A,L
1830 ;
1840 ;*** ESCRIBIR BYTE HEX ***
1850 ;
1860 hex:   PUSH AF
1870        RRCA
1880        RRCA
1890        RRCA
1900        RRCA
1910        CALL hex1
1920        POP  AF
1930 hex1:  AND  #0F
1940        ADD  A,#90
1950        DAA
1960        ADC  A,#40
1970        DAA
1980        JP   #BB5A
1990 ;
2000 ;******** ERRORES ********
2010 ;
2020 nopuedo:
2030        CALL cadena
2040        DEFB "N","o"," "
2050        DEFB "p","u","e"
2060        DEFB "d","o"," "
2070        DEFB "h","a","c"
2080        DEFB "e","r","l"
2090        DEFB "o","!",13
2100        DEFB 10,7,0
2110        RET
2120 ;
2130 nositio:
2140        POP  HL
2150        CALL cadena
2160        DEFB "N","o"," "
2170        DEFB "h","a","y"
2180        DEFB " ","s","i"
```

```
2190            DEFB "t","i","o"
2200            DEFB ".",13,10
2210            DEFB 7,0
2220            RET
2230 ;
2240 error:     CALL cadena
2250            DEFB "E","r","r"
2260            DEFB "o","r"," "
2270            DEFB "R","S","X"
2280            DEFB 13,10,7
2290            DEFB 0
2300            RET
2310 ;
2320 chequeo:
2330            LD   HL,(#AE36)
2340 EJELIN:    EQU  $-2
2350            LD   A,H
2360            OR   L
2370            RET  Z
2380            POP  HL
2390            CALL cadena
2400            DEFB "P","r","o"
2410            DEFB "g","r","a"
2420            DEFB "m","a"," "
2430            DEFB "e","n"," "
2440            DEFB "e","j","e"
2450            DEFB "c","u","c"
2460            DEFB "i","o","n"
2470            DEFB "!",13,10
2480            DEFB 7,0
2490            RET
2500 ;
2510 errorpar:
2520            CALL cadena
2530            DEFB "O","l","v"
2540            DEFB "i","d","o"
2550            DEFB " ","d","e"
2560            DEFB " ","P","A"
2570            DEFB "G","E","!"
2580            DEFB 13,10,7
2590            DEFB 0
2600            RET
2610 ;
2620 ;**** TABLA DE SALTOS ****
2630 ;
2640 tablasal:
2650            DEFW nomtabla
2660            JP   escribir
2670            JP   nueva
2680            JP   pagina
2690            JP   cogerpag
2700            JP   superior
2710            JP   lomem
2720 ;
2730 ;**** TABLA DE NOMBRES ****
2740 ;
2750 nomtabla:
2760            DEFB "P","R","I"
2770            DEFB "N","T","."
2780            DEFB "P","A","G"
2790            DEFB #C5
2800            DEFB "N","E","W"
2810            DEFB ".","P","R"
2820            DEFB "O","G","R"
2830            DEFB "A",#CD
2840            DEFB "S","E","T"
2850            DEFB ".","P","A"
2860            DEFB "G",#C5
2870            DEFB "G","E","T"
2880            DEFB ".","P","A"
2890            DEFB "G",#C5
2900            DEFB "G","E","T"
2910            DEFB ".","T","O"
2920            DEFB #D0
2930            DEFB "G","E","T"
2940            DEFB ".","L","O"
2950            DEFB "M","E",#CD
2960            DEFB #00
2970 ;
2980 workspace:
2990            DEFW #00
3000            DEFW #00
3010 ;
3020 flags:
3030            DEFB #0
```

```
10 REM UTILIDAD "PAGINA"
20 REM DE R.A.WADDILOVE
30 REM (c) MICROHOBBY AMSTRAD
40 REM CALL &A000 para ???????
50 MEMORY &9FFF:direccion=&A000
60 FOR i=1 TO 44
70 suma=0:READ codigo$,chequeo$
80 FOR j=1 TO 21 STEP 2
90 byte=VAL("&"+MID$(codigo$,j,2))
100 POKE direccion,byte
110 suma=suma+byte:direccion=direcc
ion+1
120 NEXT j
130 IF suma<>VAL("&"+chequeo$) THEN
 PRINT "ERROR DE DATOS EN LA LINEA
";140+i*10
140 NEXT I
150 DATA 21E7A1CB4EC0CBCE0199A1,656
160 DATA 21E3A1CDD1BCCD00B9F53A,6B4
170 DATA 02C0A728422164AE229CA0,464
180 DATA 22BAA02214A122E4A02117,431
190 DATA AE229FA022BDA0215EAE22,4DD
200 DATA 87A02166AE22C9A022F6A0,59F
210 DATA 2168AE22CCA0216AAE22CF,4EF
220 DATA A0216CAE22D2A02200A121,453
230 DATA 1DAE2267A1F1CD0CB9CD04,549
240 DATA A1506167652057469 6C69,445
250 DATA 7479206F6B2E070D0A00C9,2FC
260 DATA 3DC285A1CD66A1DD5E00DD,611
270 DATA 56012A7BAE25E5A7ED52DA,574
280 DATA 45A1216C01ED52EBD1D22F,570
290 DATA A12281AE2230AE7EA72014,44B
300 DATA 234E234678A7200CB12817,315
310 DATA 2B09E5A7ED52E138ED2A81,5B0
320 DATA AE2230AE36002336002336,296
330 DATA 00232283AE2285AE2287AE,422
340 DATA 2289AECD04A16F6B0D0A07,3C3
350 DATA 00C93DC285A1ED5B81AEDD,642
360 DATA 6E00DD6601732372C93DC2,482
370 DATA 85A1ED5B83AE18EC3DC285,627
380 DATA A1ED5B89AE18E2E17ECD5A,6A0
390 DATA BB23B720F8E93E26CD5ABB,5DC
400 DATA 3A81AE7CCD1BA17DF50F0F,4EE
410 DATA 0F0FCD24A1F1E60FC69027,513
420 DATA CE4027C35ABBCD04A14361,523
430 DATA 6E277420646F2074686174,3CD
440 DATA 210D0A0700C9E1CD04A14E,3A9
450 DATA 6F20726F6F6D0D0A0700C9,333
460 DATA CD04A1525358206572726F,447
470 DATA 720D0A0700C92A36AE7CB5,398
480 DATA C8E1CD04A150726F677261,586
490 DATA 6D2072756E6E696E67210D,3BC
500 DATA 0A0700C9CD04A1466F7267,3DA
510 DATA 6F742050414745210D0A07,25F
520 DATA 00C9ADA1C30EA1C3B9A0C3,668
530 DATA 79A0C3DEA0C3F0A0C3FAA0,80A
540 DATA 5052494E542E504147C54E,3A6
550 DATA 45572E50524F475241CD53,3B5
560 DATA 45542E504147C54745542E,372
570 DATA 504147C54745542E544FD0,41E
580 DATA 4745542E4C4F4D45CD0000,308
```