Ejercicio 6 - Memoria dinámica

Con este ejercicio se pretende probar el funcionamiento de memoria dinámica. Se solicitan varias celdas de memoria, se liberan algunas y solicitan nuevamente otras. Se guardan las direcciones obtenidas en el DS y luego se recorren los encabezados de cada nodo del ES para verificar la correcta implementación.

6.1 Traducción

Ejecutar:

```
>mvc 6.asm 6.bin
```

```
; probando la memoria dinámica
[00 00 00 00]: B5 00 E0 0E          1:        xor         ex,         ex
; NEW 4 celdas -> DS[1]
[00 00 00 01]: 04 00 C0 04          2:        mov         cx,          4
[00 00 00 02]: F0 00 00 05          3:        sys         %5
[00 00 00 03]: 14 00 E0 01          4:        add         ex,          1
[00 00 00 04]: 0D 00 E0 0D          5:        mov       [ex],         dx
; NEW 8 celdas -> DS[2]
[00 00 00 05]: 04 00 C0 08          6:        mov         cx,          8
[00 00 00 06]: F0 00 00 05          7:        sys         %5
[00 00 00 07]: 14 00 E0 01          8:        add         ex,          1
[00 00 00 08]: 0D 00 E0 0D          9:        mov       [ex],         dx
; NEW 10 celdas -> DS[3]
[00 00 00 09]: 04 00 C0 0A         10:        mov         cx,         10
[00 00 00 0A]: F0 00 00 05         11:        sys         %5
[00 00 00 0B]: 14 00 E0 01         12:        add         ex,          1
[00 00 00 0C]: 0D 00 E0 0D         13:        mov       [ex],         dx
; NEW 5 celdas -> DS[4]
[00 00 00 0D]: 04 00 C0 05         14:        mov         cx,          5
[00 00 00 0E]: F0 00 00 05         15:        sys         %5
[00 00 00 0F]: 14 00 E0 01         16:        add         ex,          1
[00 00 00 10]: 0D 00 E0 0D         17:        mov       [ex],         dx
; NEW 5 celdas -> DS[5]
[00 00 00 11]: 04 00 C0 05         18:        mov         cx,          5
[00 00 00 12]: F0 00 00 05         19:        sys         %5
[00 00 00 13]: 14 00 E0 01         20:        add         ex,          1
[00 00 00 14]: 0D 00 E0 0D         21:        mov       [ex],         dx
; FREE 10 celdas <- DS[3]
[00 00 00 15]: 06 00 D0 03         22:        mov         dx,        [3]
[00 00 00 16]: F0 00 00 06         23:        sys         %6
; FREE 4 celdas <- DS[1]
[00 00 00 17]: 06 00 D0 01         24:        mov         dx,        [1]
[00 00 00 18]: F0 00 00 06         25:        sys         %6
; NEW 6 celdas -> DS[6]
[00 00 00 19]: 04 00 C0 06         26:        mov         cx,          6
[00 00 00 1A]: F0 00 00 05         27:        sys         %5
[00 00 00 1B]: 14 00 E0 01         28:        add         ex,          1
[00 00 00 1C]: 0D 00 E0 0D         29:        mov       [ex],         dx
; FREE 5 celdas <- DS[4]
[00 00 00 1D]: 06 00 D0 04         30:        mov         dx,        [4]
```

```
[00 00 00 1E]: F0 00 00 06         31:        sys         %6
[00 00 00 1F]: 14 00 E0 01         32:        add         ex,          1
[00 00 00 20]: 0C 00 EF FF         33:        mov       [ex],         -1      ;marcador
[00 00 00 21]: B5 00 F0 0F         34:        xor         fx,         fx
[00 00 00 22]: 64 00 E0 0E       copy:        cmp         ex,         14
[00 00 00 23]: F7 00 00 2A         36:        jnn        fin
[00 00 00 24]: 14 00 F0 01         37:        add         fx,          1
[00 00 00 25]: 07 00 D0 0F         38:        mov         dx,       [fx]
[00 00 00 26]: 24 00 D0 01         39:        sub         dx,          1
[00 00 00 27]: 14 00 E0 01         40:        add         ex,          1
[00 00 00 28]: 0F 00 E0 0D         41:        mov       [ex],       [dx]
[00 00 00 29]: F1 00 00 22         42:        jmp       copy
[00 00 00 2A]: 04 00 A0 08        fin:        mov         ax,         %8
[00 00 00 2B]: 04 00 C0 0E         44:        mov         cx,         14
[00 00 00 2C]: 04 00 D0 01         45:        mov         dx,          1
[00 00 00 2D]: F0 00 00 02         46:        sys         %2
[00 00 00 2E]: F0 00 00 0F         47:        sys         %f
[00 00 00 2F]: FF 10 00 00         48:       stop
```

6.2 Ejecución

Ejecutar:

`>mvx 6.bin`

```
[0001]: %20001
[0002]: %20006
[0003]: %2000f
[0004]: %2001a
[0005]: %20020
[0006]: %2000f
[0007]: %ffffffff
[0008]: %40015
[0009]: %8000e
[0010]: %6001f
[0011]: %50025
[0012]: %50005
[0013]: %6001f
[0014]: %0
```

Ejercicio 7 - Strings

Con este ejercicio se pretende probar el trabajo con strings, además de los breakpoints, desensamblado y muestra de memoria.

7.1 Traducción

Ejecutar:

```
>mvc 7.asm 7.bin
```

Es necesario modificar el tamaño del ES

```
;Inicializa HEAP (manual)
[00 00 00 00]: F9 00 00 02          1:        ldh          2
[00 00 00 01]: F8 00 00 00          2:        ldl          0
[00 00 00 02]: 0D 00 90 09          3:        mov       [ac],         ac
[00 00 00 03]: 1C 00 90 01          4:        add       [ac],          1
; Escribe mensaje al usuario
msg     equ     "Escriba palabras seguidas de ENTER (en blanco para terminar)"
[00 00 00 04]: F9 00 00 03          5:        ldh          3
[00 00 00 05]: F8 00 00 2D          6:        ldl        msg
[00 00 00 06]: 04 00 A8 00          7:        mov         ax,       %800
[00 00 00 07]: 04 00 C3 E8          8:        mov         cx,       1000
[00 00 00 08]: 05 00 D0 09          9:        mov         dx,         ac
[00 00 00 09]: F0 00 00 04         10:        sys         %4
[00 00 00 0A]: 04 00 C0 01         11:        mov         cx,          1
; Lee una palabra en DS[0]
[00 00 00 0B]: F9 00 00 02         12:        ldh          2
[00 00 00 0C]: F8 00 00 00         13:        ldl          0
[00 00 00 0D]: 04 00 A9 00        ini:        mov         ax,       %900
[00 00 00 0E]: 04 00 C3 E8         15:        mov         cx,       1000
[00 00 00 0F]: 04 00 D0 00         16:        mov         dx,          0
[00 00 00 10]: F0 00 00 03         17:        sys         %3
[00 00 00 11]: C6 00 C0 00         18:       slen         cx,        [0]
[00 00 00 12]: 64 00 C0 00         19:        cmp         cx,          0      ;Si está
vacía...
[00 00 00 13]: F2 00 00 19         20:         jz     finlee
;...termina la lectura
[00 00 00 14]: 07 00 D0 09         21:        mov         dx,       [ac]      ;Sino,
muevo el HEAP a DX
[00 00 00 15]: 1D 00 90 0C         22:        add       [ac],         cx
;Incremento el HEAP para reservar la memoria
[00 00 00 16]: 1C 00 90 01         23:        add       [ac],          1      ;Uno más
por el \0
[00 00 00 17]: DE 00 D0 00         24:       smov       [dx],        [0]      ;Agrega la
palabra en el ES
[00 00 00 18]: F1 00 00 0D         25:        jmp        ini
[00 00 00 19]: F8 00 00 01     finlee:        ldl          1
[00 00 00 1A]: 05 00 D0 09         27:        mov         dx,         ac
;Inicializa 1 para comenzar el recorrido
[00 00 00 1B]: F8 00 00 00         28:        ldl          0
[00 00 00 1C]: 07 00 C0 09         29:        mov         cx,       [ac]
[00 00 00 1D]: 24 00 C0 01         30:        sub         cx,          1      ;Posiciona
a cx en el último \0
[00 00 00 1E]: 65 00 D0 0C       next:        cmp         dx,         cx      ;Termina
cuando DX llega al último \0
[00 00 00 1F]: F2 00 00 26         32:         jz        fin
[00 00 00 20]: 9C 00 D0 DF        may:        and       [dx],        %DF      ;Pasa a
MAYUSCULAS
[00 00 00 21]: 6C 00 D0 00         34:        cmp       [dx],          0      ;Si
encuentra un \0 ...
[00 00 00 22]: F5 00 00 24         35:        jnz        sig
[00 00 00 23]: 0C 00 D0 20         36:        mov       [dx],        %20      ;... pone
un espacio para contactenar
[00 00 00 24]: 14 00 D0 01        sig:        add         dx,          1
[00 00 00 25]: F1 00 00 1E         38:        jmp       next
; Muestra cadena concatenada y pasada a mayusculas
[00 00 00 26]: F0 00 00 0F        fin:        SYS         %F                ;mostrar
el ES
```

```
[00 00 00 27]: F8 00 00 01        40:        ldl         1
[00 00 00 28]: F9 00 00 02        41:        ldh         2
[00 00 00 29]: 05 00 D0 09        42:        mov        dx,         ac
[00 00 00 2A]: 04 00 A9 00        43:        mov        ax,       %900
[00 00 00 2B]: F0 00 00 04        44:        sys        %4
[00 00 00 2C]: FF 10 00 00        45:       stop
```

7.2 Ejecución

Ejecutar:

```
>mvx 7.bin -b -d
```

Cuando el programa solicite escribir palabras escriba (para finalizar presione ENTER sin escribir caracteres):
uno
dos
tres

Cuando el debug solicite cmd, escriba el rango de memoria correspondiente a las **celdas 1 a 13 del Extra segment**. Luego continúe con la ejecución. Si todo está correcto, el programa terminará escribiendo “UNO DOS TRES”.

A continuación pegue el estado de los registros, las direcciones de memoria solicitadas, el contenido del rango de memoria y la impresión final por pantalla:

```
Registros:
DS =   67108970 | SS =   67111018 | ES =   67109994 | CS =    6946816 |
HP =         -1 | IP =     196647 | SP =      66560 | BP =      65536 |
CC =          1 | AC =     131072 | AX =      -1792 | BX =          0 |
CX =     131085 | DX =     131085 | EX =          0 | FX =          0 |
[038] cmd: 1131 1143
[1131]: 00 00 00 55 U 85
[1132]: 00 00 00 4E N 78
[1133]: 00 00 00 4F O 79
[1134]: 00 00 00 20   32
[1135]: 00 00 00 44 D 68
[1136]: 00 00 00 4F O 79
[1137]: 00 00 00 53 S 83
[1138]: 00 00 00 20   32
[1139]: 00 00 00 54 T 84
[1140]: 00 00 00 52 R 82
[1141]: 00 00 00 45 E 69
[1142]: 00 00 00 53 S 83
[1143]: 00 00 00 00 . 0
[038] cmd:
UNO DOS TRES
```

Ejercicio 8 - Indirección

Con este ejercicio se pretende probar los operandos indirectos. La intención es escribir en DS, ES y SS. Al final imprime los valores de las 3 primeras celdas de ES, luego de DS y finalmente una celda del SS.

El SYS debe interpretar DX como relativo al segmento, al igual que en una indirección.

8.1 Traducción

Ejecutar:

```
>mvc 8.asm 8.bin
```

Pegar el resultado mostrado por consola:

```
;DIRECCIONAMIENTO INDIRECTO
ant             EQU     -1
sig             EQU             1
;inicializo los registros
[00 00 00 00]: 04 00 A0 01          1:        mov         ax,          1
[00 00 00 01]: 04 00 B0 05          2:        mov         bx,          5
[00 00 00 02]: 04 00 C0 02          3:        mov         cx,          2
;relleno la memoria de 1 a 10
[00 00 00 03]: 64 00 A0 07       otro:        cmp         ax,          7
[00 00 00 04]: F2 00 00 08          5:         jz      sigue
[00 00 00 05]: 0D 00 A0 0A          6:        mov       [ax],         ax
[00 00 00 06]: 14 00 A0 01          7:        add         ax,          1
[00 00 00 07]: F1 00 00 03          8:        jmp       otro
;recupero registros y multiplico x10
[00 00 00 08]: 4C FF C0 0A      sigue:        mul   [CX+ant],         10
[00 00 00 09]: 4C 00 C0 0A         10:        mul       [Cx],         10
[00 00 00 0A]: 4C 01 C0 0A         11:        mul   [cx+sig],         10
[00 00 00 0B]: 4C FF B0 0A         12:        mul     [bx-1],         10
[00 00 00 0C]: 4C 00 B0 0A         13:        mul       [Bx],         10
[00 00 00 0D]: 4C 01 B0 0A         14:        mul   [BX-ant],         10
[00 00 00 0E]: 0F 00 AF FC         15:        mov       [ax],   [cx-sig]
[00 00 00 0F]: FD C0 00 0A         16:       push       [ax]
[00 00 00 10]: 05 00 70 06         17:        mov         BP,         SP
[00 00 00 11]: 1C 00 70 01         18:        add       [BP],          1
[00 00 00 12]: F8 00 00 01         19:        LDL          1
[00 00 00 13]: F9 00 00 02         20:        LDH          2
[00 00 00 14]: 05 00 D0 09         21:        mov         dx,         ac
[00 00 00 15]: F9 00 00 00         22:        LDH          0
[00 00 00 16]: 0F 00 D0 09         23:        mov       [dx],       [ac]
[00 00 00 17]: 0F 01 D0 19         24:        mov   [dx+sig],   [ac+sig]
[00 00 00 18]: 0F 02 D0 29         25:        mov     [dx+2],     [ac+2]
;imprime todo
[00 00 00 19]: 04 00 A0 01         26:        mov         ax,         %1
[00 00 00 1A]: 04 00 C0 03         27:        mov         cx,          3
[00 00 00 1B]: F0 00 00 02         28:        sys         %2
[00 00 00 1C]: 04 00 D0 01         29:        mov         dx,          1
[00 00 00 1D]: 04 00 C0 07         30:        mov         cx,          7
```

```
[00 00 00 1E]: F0 00 00 02          31:         sys         %2
[00 00 00 1F]: 05 00 D0 07          32:         mov         dx,         bp
[00 00 00 20]: 04 00 C0 01          33:         mov         cx,          1
[00 00 00 21]: F0 00 00 02          34:         sys         %2
[00 00 00 22]: F0 00 00 0F          35:         sys         %F
[00 00 00 23]: FF 10 00 00          36:        stop
[00 00 00 24]: FF 10 00 00          37:        stop
[00 00 00 25]: FF 10 00 00          38:        stop
[00 00 00 26]: FF 10 00 00          39:        stop
[00 00 00 27]: FF 10 00 00          40:        stop
```

8.2 Ejecución

Ejecutar:

`>mvx 8.bin`

```
[0001]: 10
[0002]: 20
[0003]: 30
[0001]: 10
[0002]: 20
[0003]: 30
[0004]: 40
[0005]: 50
[0006]: 60
[0007]: 10
[0049]: 11
```

Ejercicio 9 - Fibonacci

Con este ejercicio se pretende evaluar el manejo de la pila.

9.1 Traducción

Ejecutar:

`>mvc 9.asm 9.bin`

Pegar el resultado mostrado por consola:

```
buffer  equ     0
[00 00 00 00]: 04 00 D0 00          1:         mov         dx,     buffer
[00 00 00 01]: 04 00 C0 01          2:         mov         cx,          1
[00 00 00 02]: 04 00 A0 01          3:         mov         ax,         %1
[00 00 00 03]: FD 00 00 01          4:        push          1
[00 00 00 04]: FD 00 00 01          5:        push          1
```

```
[00 00 00 05]: FD 00 00 0A          6:       push         10
[00 00 00 06]: FC 00 00 0D          7:       call       fibo
[00 00 00 07]: 14 00 60 03          8:        add         sp,          3
[00 00 00 08]: 08 00 00 01          9:        mov   [buffer],          1
[00 00 00 09]: F0 00 00 02         10:        sys         %2
[00 00 00 0A]: F0 00 00 02         11:        sys         %2
[00 00 00 0B]: F0 00 00 0F         12:        sys         %F
    ;pop     ax
[00 00 00 0C]: FF 10 00 00         13:       stop
[00 00 00 0D]: FD 40 00 07       fibo:       push         bp
[00 00 00 0E]: 05 00 70 06         15:        mov         bp,         sp
[00 00 00 0F]: 24 00 60 01         16:        sub         sp,          1
[00 00 00 10]: 6C 02 70 00         17:        cmp     [bp+2],          0
[00 00 00 11]: F2 00 00 1C         18:         jz       finf
[00 00 00 12]: 2C 02 70 01         19:        sub     [bp+2],          1
[00 00 00 13]: 0F FF 70 37         20:        mov     [bp-1],     [bp+3]
[00 00 00 14]: 1F FF 70 47         21:        add     [bp-1],     [bp+4]
[00 00 00 15]: FD C0 0F F7         22:       push     [bp-1]
[00 00 00 16]: FD C0 00 47         23:       push     [bp+4]
[00 00 00 17]: FD C0 00 27         24:       push     [bp+2]
[00 00 00 18]: FC 00 00 0D         25:       call       fibo
[00 00 00 19]: 14 00 60 03         26:        add         sp,          3
[00 00 00 1A]: 0B 00 0F F7         27:        mov   [buffer],     [bp-1]
[00 00 00 1B]: F0 00 00 02         28:        sys         %2
[00 00 00 1C]: 05 00 60 07       finf:        mov         sp,         bp
[00 00 00 1D]: FE 40 00 07         30:        pop         bp
[00 00 00 1E]: FF 00 00 00         31:        ret
```

9.2 Ejecución

Ejecutar:

```
>mvx 9.bin -d -b
```

Cuando se detenga la ejecución por breakpoint solicitar mostrar direcciones de memoria correspondientes para que muestre todo el contenido de la pila. Luego finalizar la ejecución. A continuación pegue el estado de los registros, las direcciones de memoria solicitadas, el contenido del rango de memoria y la impresión final por pantalla:

```
Registros:
DS =      65567 | SS =    4587552 | ES =         32 | CS =    2031616 |BP
HP =         -1 | IP =     196620 | SP =      65606 | BP =      65536 |
CC =          0 | AC =          0 | AX =          1 | BX =          0 |
CX =          1 | DX =          0 | EX =          0 | FX =          0 |
[011] cmd: 32 101
[0032]: 00 00 00 00 . 0
[0033]: 00 00 00 00 . 0
[0034]: 00 00 00 00 . 0
[0035]: 00 00 00 00 . 0
[0036]: 00 00 00 00 . 0
[0037]: 00 01 00 0B . 65547
[0038]: 00 03 00 19 . 196633
[0039]: 00 00 00 00 . 0
[0040]: 00 00 00 59 Y 89
[0041]: 00 00 00 90 . 144
```

```
[0042]: 00 00 00 90 . 144
[0043]: 00 01 00 11 . 65553
[0044]: 00 03 00 19 . 196633
[0045]: 00 00 00 00 . 0
[0046]: 00 00 00 37 7 55
[0047]: 00 00 00 59 Y 89
[0048]: 00 00 00 59 Y 89
[0049]: 00 01 00 17 . 65559
[0050]: 00 03 00 19 . 196633
[0051]: 00 00 00 01 . 1
[0052]: 00 00 00 22 " 34
[0053]: 00 00 00 37 7 55
[0054]: 00 00 00 37 7 55
[0055]: 00 01 00 1D . 65565
[0056]: 00 03 00 19 . 196633
[0057]: 00 00 00 02 . 2
[0058]: 00 00 00 15 . 21
[0059]: 00 00 00 22 " 34
[0060]: 00 00 00 22 " 34
[0061]: 00 01 00 23 # 65571
[0062]: 00 03 00 19 . 196633
[0063]: 00 00 00 03 . 3
[0064]: 00 00 00 0D . 13
[0065]: 00 00 00 15 . 21
[0066]: 00 00 00 15 . 21
[0067]: 00 01 00 29 ) 65577
[0068]: 00 03 00 19 . 196633
[0069]: 00 00 00 04 . 4
[0070]: 00 00 00 08 . 8
[0071]: 00 00 00 0D . 13
[0072]: 00 00 00 0D . 13
[0073]: 00 01 00 2F / 65583
[0074]: 00 03 00 19 . 196633
[0075]: 00 00 00 05 . 5
[0076]: 00 00 00 05 . 5
[0077]: 00 00 00 08 . 8
[0078]: 00 00 00 08 . 8
[0079]: 00 01 00 35 5 65589
[0080]: 00 03 00 19 . 196633
[0081]: 00 00 00 06 . 6
[0082]: 00 00 00 03 . 3
[0083]: 00 00 00 05 . 5
[0084]: 00 00 00 05 . 5
[0085]: 00 01 00 3B ; 65595
[0086]: 00 03 00 19 . 196633
[0087]: 00 00 00 07 . 7
[0088]: 00 00 00 02 . 2
[0089]: 00 00 00 03 . 3
[0090]: 00 00 00 03 . 3
[0091]: 00 01 00 41 A 65601
[0092]: 00 03 00 19 . 196633
[0093]: 00 00 00 08 . 8
[0094]: 00 00 00 01 . 1
[0095]: 00 00 00 02 . 2
[0096]: 00 00 00 02 . 2
[0097]: 00 01 00 00 . 65536
[0098]: 00 03 00 07 . 196615
[0099]: 00 00 00 09 . 9
[0100]: 00 00 00 01 . 1
[0101]: 00 00 00 01 . 1
[011] cmd:
```

Debe dar stack underflow

Ejercicio 10 - Hanoi

En este ejercicio se pretende realizar una evaluación integradora.

10.1 Traducción

Ejecutar:

```
>mvc A.asm A.bin
```

```
titulo equ  "--- TORRE DE HANOI ---"
msg    equ  "Ingrese la cantidad de discos:"
strA  equ "Torre A"
strB  equ "Torre B"
strC  equ "Torre C"
torreA equ 0
torreB equ 1
torreC equ 2
str    equ 1 ; ubicación de los str
step   equ 4
[00 00 00 00]: 08 00 40 00          1:        mov     [step],          0
[00 00 00 01]: F9 00 00 02          2:        ldh          2
[00 00 00 02]: F8 00 00 32          3:        ldl         50
[00 00 00 03]: 05 00 D0 09          4:        mov         dx,         ac
[00 00 00 04]: F9 00 00 03          5:        ldh          3
[00 00 00 05]: F8 00 00 82          6:        ldl        msg
[00 00 00 06]: DF 00 D0 09          7:       smov       [dx],       [ac]
[00 00 00 07]: F9 00 00 02          8:        ldh          2
[00 00 00 08]: F8 00 00 0A          9:        ldl         10
[00 00 00 09]: 05 00 D0 09         10:        mov         dx,         ac
[00 00 00 0A]: F9 00 00 03         11:        ldh          3
[00 00 00 0B]: F8 00 00 6B         12:        ldl     titulo
[00 00 00 0C]: DF 00 D0 09         13:       smov       [dx],       [ac]
[00 00 00 0D]: 04 00 A8 00         14:        mov         ax,       %800
[00 00 00 0E]: F0 00 00 04         15:        sys         %4
[00 00 00 0F]: 14 00 D0 28         16:        add         dx,         40
[00 00 00 10]: 04 00 A9 00         17:        mov         ax,       %900
[00 00 00 11]: F0 00 00 04         18:        sys         %4
[00 00 00 12]: 04 00 D0 00         19:        mov         dx,          0
[00 00 00 13]: 04 00 C0 01         20:        mov         cx,          1
[00 00 00 14]: 04 00 A8 01         21:        mov         ax,       %801
[00 00 00 15]: F0 00 00 01         22:        sys         %1
[00 00 00 16]: 04 00 C0 01         23:        mov         cx,        str
        ; Ubico los nombres de las torres
[00 00 00 17]: F9 00 00 03         24:        ldh          3
[00 00 00 18]: F8 00 00 A1         25:        ldl       strA
[00 00 00 19]: 0D 00 C0 09         26:        mov [cx+torreA],         ac
[00 00 00 1A]: F8 00 00 A9         27:        ldl       strB
[00 00 00 1B]: 0D 01 C0 09         28:        mov [cx+torreB],         ac
[00 00 00 1C]: F8 00 00 B1         29:        ldl       strC
[00 00 00 1D]: 0D 02 C0 09         30:        mov [cx+torreC],         ac

[00 00 00 1E]: FD 00 00 02         31:       push     torreC                 ;torre
auxiliar (aux)
```

```
[00 00 00 1F]: FD 00 00 01         32:       push     torreB                  ;torre
destino  (dtn)
[00 00 00 20]: FD 00 00 00         33:       push     torreA                  ;torre
origen   (org)
[00 00 00 21]: FD 80 00 00         34:       push        [0]                  ;cantidad
de discos a mover
[00 00 00 22]: FC 00 00 25         35:       call      hanoi
[00 00 00 23]: 14 00 60 05         36:        add         sp,          5
[00 00 00 24]: FF 10 00 00         37:       stop
; Hanoi
; parametros: (1) Cant discos, (2) origen, (3) destino, (4) aux
discos equ 2
origen equ 3
destino equ 4
aux equ 5
[00 00 00 25]: FD 40 00 07      hanoi:       push         bp
[00 00 00 26]: 05 00 70 06         39:        mov         bp,         sp
[00 00 00 27]: FD 40 00 0C         40:       push         cx

[00 00 00 28]: 07 00 C0 27         41:        mov         cx, [BP+discos]
[00 00 00 29]: 64 00 C0 00         42:        cmp         cx,          0      ;si la
cantidad a mover es 0...
[00 00 00 2A]: F2 00 00 3D         43:         jz       finh                  ;...no
hace nada
[00 00 00 2B]: 24 00 C0 01         44:        sub         cx,          1

[00 00 00 2C]: FD C0 00 47         45:       push [BP+destino]                ;destion
pasa a auxiliar
[00 00 00 2D]: FD C0 00 57         46:       push   [BP+aux]                  ;auxiliar
es el nuevo destino
[00 00 00 2E]: FD C0 00 37         47:       push [BP+origen]                 ;el origen
se mantiene
[00 00 00 2F]: FD 40 00 0C         48:       push         cx
[00 00 00 30]: FC 00 00 25         49:       call      hanoi
[00 00 00 31]: 14 00 60 04         50:        add         sp,          4

[00 00 00 32]: FD C0 00 47         51:       push [BP+destino]                ;destino
[00 00 00 33]: FD C0 00 37         52:       push [BP+origen]                 ;origen
[00 00 00 34]: FC 00 00 41         53:       call      print
[00 00 00 35]: 14 00 60 02         54:        add         sp,          2

[00 00 00 36]: FD C0 00 37         55:       push [BP+origen]                 ;el origen
pasa a auxiliar
[00 00 00 37]: FD C0 00 47         56:       push [BP+destino]                ;el
destino se mantiene
[00 00 00 38]: FD C0 00 57         57:       push   [BP+aux]                  ;el
auxiliar pasa a ser origen
[00 00 00 39]: FD 40 00 0C         58:       push         cx
[00 00 00 3A]: FC 00 00 25         59:       call      hanoi
[00 00 00 3B]: 14 00 60 04         60:        add         sp,          4
[00 00 00 3C]: F1 00 00 3D         61:        jmp       finh

[00 00 00 3D]: FE 40 00 0C       finh:        pop         cx
[00 00 00 3E]: 05 00 60 07         63:        mov         sp,         bp
[00 00 00 3F]: FE 40 00 07         64:        pop         bp
[00 00 00 40]: FF 00 00 00         65:        ret

desde  equ  2
hasta  equ  3
flecha equ " -> "
paso equ "PASO "
[00 00 00 41]: FD 40 00 07      print:       push         bp
[00 00 00 42]: 05 00 70 06         67:        mov         bp,         sp
[00 00 00 43]: FD 40 00 0A         68:       push         ax
```

```
[00 00 00 44]: FD 40 00 0B          69:       push         bx
[00 00 00 45]: FD 40 00 0C          70:       push         cx
[00 00 00 46]: FD 40 00 0D          71:       push         dx
[00 00 00 47]: F9 00 00 03          72:        ldh          3
[00 00 00 48]: F8 00 00 BE          73:        ldl       paso
[00 00 00 49]: 05 00 D0 09          74:        mov         dx,          ac
[00 00 00 4A]: 04 00 B0 01          75:        mov         bx,           1
[00 00 00 4B]: 04 00 A9 00          76:        mov         ax,        %900
[00 00 00 4C]: F0 00 00 04          77:        sys         %4
[00 00 00 4D]: 04 00 D0 04          78:        mov         dx,        step
[00 00 00 4E]: 1C 00 D0 01          79:        add       [dx],           1
[00 00 00 4F]: 04 00 C0 01          80:        mov         cx,           1
[00 00 00 50]: 04 00 A9 01          81:        mov         ax,        %901
[00 00 00 51]: F0 00 00 02          82:        sys         %2
[00 00 00 52]: 07 00 D0 27          83:        mov         dx, [bp+desde]
[00 00 00 53]: 14 00 D0 01          84:        add         dx,         str
[00 00 00 54]: 07 00 D0 0D          85:        mov         dx,        [dx]
[00 00 00 55]: 04 00 A9 00          86:        mov         ax,        %900
[00 00 00 56]: 04 00 B0 01          87:        mov         bx,           1
[00 00 00 57]: F0 00 00 04          88:        sys         %4
[00 00 00 58]: F9 00 00 03          89:        ldh          3
[00 00 00 59]: F8 00 00 B9          90:        ldl     flecha
[00 00 00 5A]: 05 00 D0 09          91:        mov         dx,          ac
[00 00 00 5B]: 04 00 A9 00          92:        mov         ax,        %900
[00 00 00 5C]: 04 00 B0 01          93:        mov         bx,           1
[00 00 00 5D]: F0 00 00 04          94:        sys         %4
[00 00 00 5E]: 07 00 D0 37          95:        mov         dx, [bp+hasta]
[00 00 00 5F]: 14 00 D0 01          96:        add         dx,         str
[00 00 00 60]: 07 00 D0 0D          97:        mov         dx,        [dx]
[00 00 00 61]: 04 00 A8 00          98:        mov         ax,        %800
[00 00 00 62]: 04 00 B0 01          99:        mov         bx,           1
[00 00 00 63]: F0 00 00 04         100:        sys         %4

[00 00 00 64]: FE 40 00 0D        finp:        pop         dx
[00 00 00 65]: FE 40 00 0C         102:        pop         cx
[00 00 00 66]: FE 40 00 0B         103:        pop         bx
[00 00 00 67]: FE 40 00 0A         104:        pop         ax
[00 00 00 68]: 05 00 60 07         105:        mov         sp,          bp
[00 00 00 69]: FE 40 00 07         106:        pop         bp
[00 00 00 6A]: FF 00 00 00         107:        ret
```

10.2 Ejecución

Ejecutar:

```
>mvx A.bin
```

```
Ingrese la cantidad de discos:3
```

Es necesario modificar el tamaño del ES

```
--- TORRE DE HANOI ---
Ingrese la cantidad de discos:3
PASO 1 Torre A -> Torre B
PASO 2 Torre A -> Torre C
PASO 3 Torre B -> Torre C
PASO 4 Torre A -> Torre B
```

```
PASO 5 Torre C -> Torre A
PASO 6 Torre C -> Torre B
PASO 7 Torre A -> Torre B
```

```
>mvx A.bin

Ingrese la cantidad de discos:9
```

```
--- TORRE DE HANOI ---
Ingrese la cantidad de discos:9
ERROR: Stack overflow
```