Ejercicio 1 - ¡Hola mundo!

Con este ejercicio se pretende probar la correcta impresión de valores de celdas en todos los formatos simultáneamente. Además de la correcta ejecución del jump.

1.1 Traducción

Ejecutar:

```
>mvc 1.asm 1.bin
```

```
[0000]: 04 00 A0 FF                    1:  MOV          AX, %FF
[0001]: 04 00 C0 01                    2:  MOV          CX, 1
[0002]: 04 00 D0 01                    3:  MOV          DX, 1
[0003]: 08 00 10 61                    4:  MOV        [%1], 'a'
[0004]: F0 00 00 02                UNOMAS:  SYS          %2
[0005]: 18 00 10 01                    6:  ADD         [1], 1
[0006]: 68 00 10 65                    7:  CMP         [1], 'e'
[0007]: F6 00 00 04                    8:  JNP       UNOMAS
[0008]: FF 10 00 00                    9:  STOP
```

*Requiere cambiar el rótulo “masuno” por “unomas”

1.2 Ejecución

Ejecutar:

```
>mvx 1.bin -c
```

```
a %00000061  @00000141  97
b %00000062  @00000142  98
c %00000063  @00000143  99
d %00000064  @00000144  100
e %00000065  @00000145  101
```

Ejercicio 2 - Manejo de números negativos

Con este ejercicio se pretende evaluar cómo se almacena y se opera con los números negativos. También el modo debug (breakpoint) para mostrar el resultado final de las celdas.

(Es importante que se muestran como quedaron las celdas de memoria y los registros)

2.1 Traducción

Ejecutar:

```
>mvc 2.asm 2.bin
```

```
[0000]: 04 00 A0 01                    1:  MOV           AX, 1
[0001]: 09 00 00 0A                    2:  MOV          [0], AX
[0002]: 06 00 B0 00                    3:  MOV           BX, [0]
[0003]: 24 00 A0 01                    4:  SUB           AX, 1
[0004]: 09 00 10 0A                    5:  MOV          [1], AX
[0005]: 06 00 E0 01                    6:  MOV           EX, [1]
[0006]: 24 00 A0 01                    7:  SUB           AX, 1
[0007]: 09 00 20 0A                    8:  MOV          [2], AX
[0008]: 06 00 E0 02                    9:  MOV           EX, [2]
[0009]: 44 00 AF FF                   10:  MUL           AX, %ffff
[0010]: 09 00 30 0A                   11:  MOV          [3], AX
[0011]: 06 00 F0 03                   12:  MOV           FX, [3]
[0012]: F9 00 FF FF                   13:  LDH       65535
[0013]: F8 00 FF FF                   14:  LDL       65535
[0014]: 09 00 40 09                   15:  MOV          [4], AC
[0015]: 84 00 90 20                   16:  SHR           AC, 32
[0016]: 04 00 C0 01                   17:  MOV           CX, 1
[0017]: 74 00 C0 1F                   18:  SHL           CX, 31
[0018]: 84 00 C0 1F                   19:  SHR           CX, 31
[0019]: 09 00 50 0C                   20:  MOV          [5], CX
[0020]: F0 00 00 0F                   21:  SYS          %F
[0021]: FF 10 00 00                   22:  STOP
```

2.2 Ejecución

Ejecutar:

```
>mvx 2.bin -d -b -c
CMD: 22 28
```

(Asegurarse de que sean visibles los valores de los registros AC, AX, BX, CX, EX, FX y las celdas de memoria DS:0 a DS:5 inclusive)

```
  [0013]: F8 00 FF FF LDL       -1
  [0014]: 09 00 40 09 MOV       [4],AC
  [0015]: 84 00 90 20 SHR       AC,32
  [0016]: 04 00 C0 01 MOV       CX,1
  [0017]: 74 00 C0 1F SHL       CX,31
```

```
  [0018]: 84 00 C0 1F SHR       CX,31
  [0019]: 09 00 50 0C MOV       [5],CX
  [0020]: F0 00 00 0F SYS       15
> [0021]: FF 10 00 00 STOP
DS =          22 |                    |                    |                    |
                 | IP =          21 |                    |                    |
CC = -2147483648 | AC =          -1 | AX =           1 | BX =           1 |
CX =          -1 | DX =           0 | EX =          -1 | FX =           1 |
[0021]: cmd: 22 28
[0022]: 1
[0023]: 0
[0024]: -1
[0025]: 1
[0026]: -1
[0028]: 0
```

Ejercicio 3 - Conversión de notación

Con este ejercicio se pretende mostrar una representación binaria de un número hexadecimal.

3.1 Traducción

Ejecutar:

```
>mvc 3.asm 3.bin
```

```
[0000]: 04 00 A0 08                1:  MOV          AX, %008
[0001]: 04 00 C0 01                2:  MOV          CX, 1
[0002]: 04 00 D0 00                3:  MOV          DX, 0
[0003]: F0 00 00 01                4:  SYS         1
[0004]: 04 00 A0 20                5:  MOV          AX, 32
[0005]: BA 00 10 01                6:  XOR         [1] , [1]
[0006]: 06 00 E0 00            SIGUE:  MOV          EX, [0]
[0007]: 78 00 00 01                8:  SHL         [0] , 1
[0008]: 84 00 E0 1F                9:  SHR          EX, 31
[0009]: 94 00 E0 01               10:  AND          EX, 1
[0010]: 48 00 10 0A               11:  MUL         [1] , 10
[0011]: 19 00 10 0E               12:  ADD         [1] , EX
[0012]: 24 00 A0 01               13:  SUB          AX, 1
[0013]: F5 00 00 06               14:  JNZ        SIGUE
[0014]: 04 00 A0 01               15:  MOV          AX, %001
[0015]: 04 00 C0 01               16:  MOV          CX, 1
[0016]: 04 00 D0 01               17:  MOV          DX, 1
[0017]: F0 00 00 02               18:  SYS          2
```

```
[0018]: FF 10 00 00                  19:  STOP
[0019]: 08 00 10 01                  20:  MOV         [1] , 1 // ojo esto no.
[0020]: F0 00 00 02                  21:  SYS           2
[0021]: FF 10 00 00                  22:  STOP
```

*Requiere cambiar la instrucción ***AMD*** por ***AND***

3.2 Ejecución

Ejecutar:

```
>mvx 3.bin
```

```
[0000]: 101
```

```
[0000]: 101
100000001
```

```
>mvx 3.bin
```

```
[0000]: 87
```

```
[0000]: 87
10000111
```

```
>mvx 3.bin
```

```
[0000]: AA
```

```
[0000]: AA
10101010
```

```
>mvx 3.bin
```

```
[0000]: CAE
```

```
[0000]: CAE
-1659048586
```

Ejercicio 4 - Prueba de escritura y ciclos

Con este ejercicio se pretende evaluar la llamada al sistema para escritura en un ciclo variando el formato a intervalos regulares (utilizando el módulo del número de impresión).

Ejecutar:

```
>mvc 4.asm 4.bin
```

```
[0000]: BA 0F F0 FF                  1:  XOR        [%FF], [%FF] ; cambia FFF por FF
[0001]: 18 0F F0 0F                  2:  ADD        [%FF], %F
[0002]: 04 00 A0 08                  3:  MOV           AX, %8
[0003]: 04 00 C0 01                  4:  MOV           CX, 1
[0004]: 04 00 D0 FF                  5:  MOV           DX, %FF
[0005]: 04 00 F0 1F                  6:  MOV           FX, 31
[0006]: 64 00 F0 00               CMP1:  CMP           FX, 0
[0007]: F4 00 00 13                  8:  JN           FIN
[0008]: 05 00 E0 0F                  9:  MOV           EX, FX
[0009]: 24 00 F0 01                 10:  SUB           FX, 1
[0010]: 54 00 E0 04                 11:  DIV           EX, 4
[0011]: 64 00 90 00                 12:  CMP           AC, 0
[0012]: F2 00 00 11                 13:  JZ          LINE
[0013]: A4 00 A1 00                 14:  OR            AX, %100
[0014]: F0 00 00 02               SYS2:  SYS           2
[0015]: 78 0F F0 01                 16:  SHL        [%FF], 1
[0016]: F1 00 00 06                 17:  JMP         CMP1
[0017]: 94 00 AE FF               LINE:  AND           AX, %EFF
[0018]: F1 00 00 0E                 19:  JMP         SYS2
[0019]: FF 10 00 00                FIN:  STOP
```

4.2 Ejecución

Ejecutar:

```
>mvx 4.bin
```

```
[0255]:  %0000000F [0255]:  %0000001E [0255]:  %0000003C [0255]:  %00000078
```

```
[0255]:  %000000F0 [0255]:  %000001E0 [0255]:  %000003C0 [0255]:  %00000780
[0255]:  %00000F00 [0255]:  %00001E00 [0255]:  %00003C00 [0255]:  %00007800
[0255]:  %0000F000 [0255]:  %0001E000 [0255]:  %0003C000 [0255]:  %00078000
[0255]:  %000F0000 [0255]:  %001E0000 [0255]:  %003C0000 [0255]:  %00780000
[0255]:  %00F00000 [0255]:  %01E00000 [0255]:  %03C00000 [0255]:  %07800000
[0255]:  %0F000000 [0255]:  %1E000000 [0255]:  %3C000000 [0255]:  %78000000
[0255]:  %F0000000 [0255]:  %E0000000 [0255]:  %C0000000 [0255]:  %80000000
```

Ejercicio 5 - Valores inmediatos y saltos

En este ejercicio se pretende evaluar el uso de operandos inmediatos y el correcto funcionamiento de los saltos.

5.1 Traducción

Ejecutar:

```
>mvc 5.asm 5.bin
```

```
[0000]: 08 00 00 62                1:  MOV          [0], %62
[0001]: 08 00 10 75                2:  MOV          [1], @165
[0002]: 08 00 20 65                3:  MOV          [2], 101
[0003]: 08 00 30 6E                4:  MOV          [3], #110
[0004]: 08 00 40 6F                5:  MOV          [4], 'o'
[0005]: 04 00 E0 20                6:  MOV           EX, ' '
[0006]: FB 40 00 0E                7:  NOT           EX
[0007]: 60 00 00 00              CMP1:  CMP           0, 0
[0008]: F5 00 00 0A                9:  JNZ         CMP2
[0009]: 99 00 00 0E              COM1:  AND          [0], EX
[0010]: 60 00 F0 10              CMP2:  CMP           15, %10
[0011]: F7 00 00 12               12:  JNN         COM3
[0012]: F4 00 00 0E               13:  JN          COM2
[0013]: F1 00 00 15               14:  JMP         COM4
[0014]: 99 00 10 0E              COM2:  AND          [1], EX
[0015]: 60 02 00 20              CMP3:  CMP          %20, '
[0016]: F2 00 00 12               17:  JZ          COM3
[0017]: F1 00 00 0F               18:  JMP         CMP3
[0018]: 99 00 20 0E              COM3:  AND          [2], EX
[0019]: 60 02 80 29              CMP4:  CMP          '(', ')'
[0020]: F3 00 00 19               21:  JP          COM5
[0021]: 99 00 30 0E              COM4:  AND          [3], EX
[0022]: 60 00 10 02              CMP5:  CMP           1, 2
[0023]: F6 00 00 19               24:  JNP         COM5
[0024]: F1 00 00 16               25:  JMP         CMP5
[0025]: 99 00 40 0E              COM5:  AND          [4], EX
[0026]: 04 00 A1 10               27:  MOV           AX, %110
[0027]: 04 00 C0 05               28:  MOV           CX, 5
[0028]: 04 00 D0 00               29:  MOV           DX, 0
[0029]: F0 00 00 02               30:  SYS           2
```

5.2 Ejecución

Ejecutar:

`>mvx 5.bin`

```
[0000]: BUENO
```

*Diferencia por detalles de implementación:

```
[0000]: B [0000]: U [0000]: E [0000]: N [0000]: O
```

No está explícito en la documentación como se debe reaccionar ante la situación de tener que escribir el prompt en las iteración, pero en realidad el prompt debería escribirse una sola vez.