

# EJERCICIOS SISTEMAS BASADOS EN MICROPROCESADORES

(05/03/2021)

## Ejercicio 1: Cálculo de factoriales

1.  $4! \times 5! = 2880$  (B40 en hexadecimal)

The screenshot shows a DOSBox emulator window titled "DOSBox 0.70, Cpu Cycles: 3000, Frameskip: 0, Program: TD". The window contains an assembly code editor with the following code:

```
cs:0026 ZEBB1E0000  * MOV BX, FACT_DATO_1
cs:002B F7E3        * MUL BX ; EN AX ESTA EL RESULTADO DE
cs:002D 26A30000    * MOV RESULT, AX
cs:0031 2689160200  * MOV RESULT*2, DX
cs:0036 B8004C      * MOV AX, 4C00H
cs:0039 CD21        * INT 21H

#factor#factor
cs:003B B80100      * MOV AX, 1
cs:003E 32ED        * XOR CH, CH
cs:0040 83F900      * CMP CX, 0
cs:0043 7405        * JE FIN

#factor#ir
cs:0045 F7E1        * MUL CX
cs:0047 49          * DEC CX
cs:0048 75FB        * JNE IR
```

The registers window on the right shows the following values:

Register	Value
ax	0B40
bx	0018
cx	0000
dx	0000
si	0000
di	EEEE
bp	0100
sp	0040
ds	480E
es	480F
ss	4815
cs	4810
ip	0036

The memory window at the bottom shows the following values:

Address	Value
es:0000	40 0B 00 00 00 00 00 00
es:0008	00 00 00 00 00 00 00 00
es:0010	18 00 B8 0E 48 8E D8 B8
es:0018	15 48 8E D0 B8 0F 48 8E
es:0020	C0 BC 40 00 B8 0E 00 00
ss:0048	0009
ss:0046	0000
ss:0044	0040
ss:0042	0209
ss:0040	7302

Los cambios realizados en el código han sido sustituir DATO\_1 por el número 4 y DATO\_2 por el número 5.

El resultado es B40, como puede apreciarse el programa proporciona el resultado correcto tras buscar es:00 (puesto en Little Endian).

En este caso, el valor de FACT\_DATO\_1 es 0018, que coincide con el valor de 4! (24 en decimal).

2.  $8! = 40320$  (9D80 en hexadecimal)

Los cambios realizados en el código han sido sustituir DATO\_1 por el número 8 y DATO\_2 por el número 1.

El resultado es 9D80, como puede apreciarse en el programa proporciona el resultado correcto tras buscar es:00 (escrito en Little Endian).

En este caso, el valor de FACT\_DATO\_1 es 9D80, que coincide con el valor de  $8!$  (40320 en decimal).

3.  $9! = 362880$  (58980 en hexadecimal)

4.  $8! \times 7! = 203212800$  (C1CC800 en hexadecimal)

Valor de FACT\_DATO\_1 es 18

## Ejercicio 2: Modificación del Programa “Factor”

Lo que hemos cambiado en el código ha sido: primero realizar la multiplicación de DATO\_1 y DATO\_2 y, a continuación, llamar a factor con el resultado, adjuntamos a continuación el código de la rutina principal:

```
MOV AL, DATO_1
MOV CL, DATO_2
MUL CL
MOV CL, AL
CALL FACTOR
MOV RESULT, AX
MOV RESULT+2, DX
MOV AX, 4C00H
INT 21H
```

1.  $(2 \times 3)! = 6! = 720$  (2D0 en hexadecimal)

```

DOSBox Status Window
DOSBox 0.70, Cpu Cycles: 3000, Frameskip 0, Program: TD
File View Run Breakpoints Data Options Window Help
[CPU 80486]
#factor#start
cs:0002 B0E4B * MOV AX, DATOS
cs:0005 8ED8 * MOV DS, AX
cs:0007 B8144B * MOV AX, PILA
cs:000A 8ED0 * MOV SS, AX
cs:000C B0F4B * MOV AX, EXTRA
cs:000F 8EC0 * MOV ES, AX
cs:0011 BC4000 * MOV SP, 64
cs:0014 A00000 * MOV AL, DATO_1
cs:0017 8A0E0100 * MOV CL, DATO_2
cs:001B F6E1 * MUL CL
cs:001D 8ACB * MOV CL, AL
cs:001F E0E00 * CALL FACTOR
cs:0022 26A30000 * MOV RESULT, AX
cs:0026 2689160200 * MOV RESULT+2, DX
ax 0000 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0042 d=0
ds 47FE
es 47FE
ss 4814
cs 4810
ip 0002
ds:0000 CD 20 FF 9F 00 EA FF FF = f 0
ds:0008 AD DE E0 01 4E 15 AA 01 i10E8~0
ds:0010 4E 15 7C 02 5B 0F 2C 01 N510[*],0
ds:0018 01 01 01 00 02 FF FF FF 0000
ds:0020 FF FF FF FF FF FF FF FF
ss:0044 0040
ss:0042 0209
ss:0040 52FB
ss:003E 0000
ss:003C 0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

El resultado es 2D0, como puede apreciarse en el programa proporciona el resultado correcto tras buscar es:00 (escrito en Little Endian).

2.  $(2 \times 4)! = 8! = 40320$  (9D80 en hexadecimal)

```

DOSBox Status Window
DOSBox 0.70, Cpu Cycles: 3000, Frameskip 0, Program: TD
File View Run Breakpoints Data Options Window Help
[CPU 80486]
cs:0022 26A30000 * MOV RESULT, AX
cs:0026 2689160200 * MOV RESULT+2, DX
cs:002B B8004C * MOV AX, 4C00H
cs:002E CD21 * INT 21H
#factor#factor
cs:0030 B80100 * MOV AX, 1
cs:0033 32ED * XOR CH, CH
cs:0035 83F900 * CMP CX, 0
cs:0038 7405 * JE FIN
#factor#ir
cs:003A F7E1 * MUL CX
cs:003C 49 * DEC CX
cs:003D 75FB * JNE IR
#factor#fin
cs:003F C3 * RET
ax 9D80 c=0
bx 0000 z=1
cx 0000 s=0
dx 0000 o=0
si 0000 p=1
di 0000 a=0
bp 0000 i=1
sp 0040 d=0
ds 480E
es 480F
ss 4814
cs 4810
ip 002B
es:0000 80 9D 00 00 00 00 00 00 C0
es:0008 00 00 00 00 00 00 00 00
es:0010 00 00 B8 0E 4B D8 B8 00 00 00
es:0018 14 4B 8E D0 B8 0F 4B 8E 00 00 00
es:0020 C0 BC 40 00 A0 00 00 BA 00 00 00
ss:0048 0009
ss:0046 0030
ss:0044 0040
ss:0042 0209
ss:0040 7302
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

El resultado es 9D80, como puede apreciarse en el programa proporciona el resultado correcto tras buscar es:00 (escrito en Little Endian).

3.  $(3 \times 3)! = 9! = 362880$  (58980 en hexadecimal)
4.  $(5 \times 2)! = 10! = 3628800$  (375F00 en hexadecimal)