

POB

David Teofilo Garitagoitia Romero
SISTEMAS BASADOS EN MICROPROCESADORES POB

Contenido

EJERCICIO 1.....2

1. $4! \times 5! =$ 2

2. $8! =$ 3

3. $9! =$ 4

4. $8! \times 7! =$ 4

Ejercicio 25

1. $(2 \times 3)! =$ 5

2. $(2 \times 4)! =$ 6

3. $(3 \times 3)! =$ 6

4. $(5 \times 2)! =$ 7

EJERCICIO 1

1. $4! \times 5! =$

Para el ejercicio 1 basta con modificar los valores de DATO_1 y DATO_2 en data segment,

Para el primero muestra como resultado 0B40, convirtiendo ese valor de hexadecimal a decimal corresponde con 2.88×10^3 , que es el valor de $4! \times 5!$

The screenshot shows the DOSBox 0.70 interface with the CPU window displaying assembly code and registers. The program is titled 'CPU 80486' and is in the 'READY' state. The assembly code is as follows:

```
Modul CPU 80486
MOV #factor#73: MOV AX, 4C00H
MOV cs:0036 B8004C mov ax,4C00
#factor#74: INT 21H
; FI cs:0039 CD21 int 21
MOV #factor#factor: MOV AX, 1
INT cs:003B B80100 mov ax,0001
#factor#85: XOR CH,CH
START EN cs:003E 32ED xor ch,ch
; SUBRUT #factor#86: CMP CX, 0
; ENTRAD cs:0040 83F900 cmp cx,0000
; SALIDA #factor#87: JE FIN
; cs:0043 7405 je #factor#fin (
#factor#ir: MUL CX
es:0000 0B40 0000 0000 0000
MOV es:0008 0000 0000 0000 0000
XOR es:0010 0018 0EB8 8E48 B8D8
CMP es:0018 4815 D08E 0FB8 8E48
```

The registers window shows the following values:

Register	Value	Comment
ax	4C00	c=0
bx	0018	z=0
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	4815	
cs	4810	
ip	0039	

The Watches window shows the following values:

Address	Value
es:0000	0B40 0000 0000 0000
es:0008	0000 0000 0000 0000
es:0010	0018 0EB8 8E48 B8D8
es:0018	4815 D08E 0FB8 8E48
ss:0042	0209
ss:0040	7202

The bottom status bar shows the following keyboard shortcuts:

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

2. 8! =

Para este se puede, tanto cambiar el DATO_2 por un 1, como modificar el programa para que únicamente haga el factorial de 1 dato, como podemos ver el resultado es 9D80 por lo que es correcto

DOSBox 0.70, Cpu Cycles: 3000, Frameskip: 0, Program: TD

Module: factor File: C:\PRA0\FACTOR\factor.asm 74

```

MOV CL, DATO_1
CALL FACTOR
MOV     CPU 80486
;MOV    cs:0036 B8004C    MOV AX, 4C00H
MOV     cs:0039 CD21    INT 21H
CALL    #factor#factor
MOV     cs:003B B80100    MOV AX, 1
MUL     cs:003E 32ED    XOR CH, CH
        cs:0040 83F900    CMP CX, 0
; AL    cs:0043 7405    JE FIN
MOV     #factor#ir
MOV     cs:0045 F7E1    MUL CX
        cs:0047 49    DEC CX
; FI    cs:0048 75FB    JNE IR
MOV     #factor#fin
INT     cs:004A C3    RET
START EN

```

Watch

es:0000 80 9D 00 00 00 00 00 00 C0
es:0008 00 00 00 00 00 00 00 00 00
es:0010 01 00 B8 0E 48 8E D8 B8 0E 48 8E
es:0018 15 48 8E D0 B8 0F 48 8E 8E 48 8E

ss:0042 0209
ss:0040 7202

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

factor.asm
48 MOV AX, PILA
49 MOV SS, AX
50
51 MOV AX, EXTRA
52 MOV ES, AX
53
54 ; CARGA EL PUNTERO DE PILA CON EL VALOR MAS ALTO
55 MOV SP, 64
56
57 ; FIN DE LAS INICIALIZACIONES
58
59 ; COMIENZO DEL PROGRAMA
60 MOV CL, DATO_1
61 CALL FACTOR
62 MOV FACT_DATO_1, AX ; ALMACENA EL RESULTADO
63 MOV RESULT, AX
64 ; MOV CL, DATO_2
65 ; CALL FACTOR
66 ; MOV BX, FACT_DATO_1
67 ; MUL BX ; EN AX ESTA EL RESULTADO DEL FACTORIAL DEL SEGUNDO
68
69 ; ALMACENA EL RESULTADO
70 ; MOV RESULT, AX
71 ; MOV RESULT+2, DX
72
73 ; FIN DEL PROGRAMA
74 MOV AX, 4C00H
75 INT 21H
76

```

3. 9!=

El resultado no coincide porque no entra el valor en la variable ya que como máximo llega a FFFF, por lo que ocurre overflow como se puede ver en la bandera O que se activa dentro de la función factor, lo mismo en el apartado 4 de este mismo ejercicio

4. 8! x 7! =

Ocorre lo mismo que en el caso anterior

En cada uno de los casos el valor de FACT_DATO_1 es el valor del factorial del dato 1

```

;COMIENZO DEL PROGRAMA
MOV CL, DATO_1
CALL FACTOR
MOV FACT_DATO_1, AX ; ALMACENA EL RESULTADO

```

Ejercicio 2

Para que calcule la multiplicación y después haga el factorial, basta con eliminar una llamada a factor y hacer la multiplicación antes.

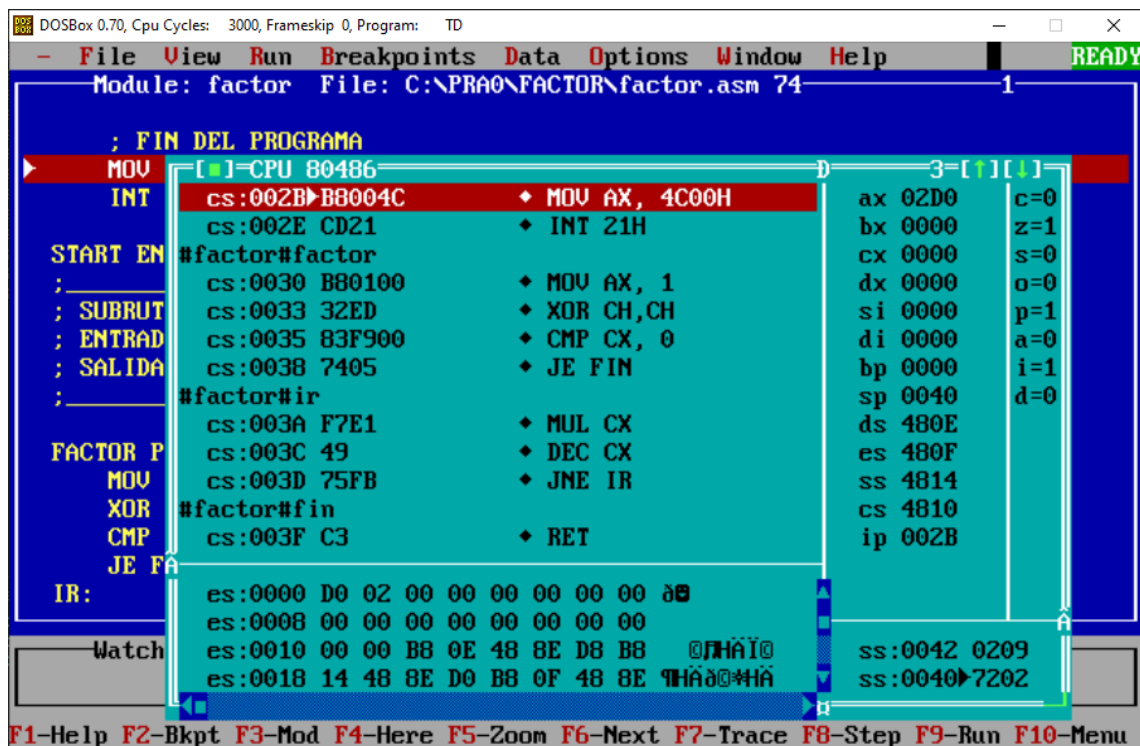
```
;COMIENZO DEL PROGRAMA
MOV AL, DATO_1
;CALL FACTOR
;MOV FACT_DATO_1, AX      ; ALMACENA EL RESULTADO
MOV CL, DATO_2 ;Tiene que ser en L porque es little endian
;CALL FACTOR
;MOV BX, FACT_DATO_1
MUL CX                ; EN AX ESTA EL RESULTADO DEL FACTORIAL DEL SEGUNDO
MOV CX, AX ;Se pasa el valor de la multiplicación al registro C que es el que se usa en factor
; ALMACENA EL RESULTADO
CALL FACTOR |
MOV RESULT, AX ;Se pasa AX que contiene el valor del resultado a RESULT
MOV RESULT+2, DX

; FIN DEL PROGRAMA
MOV AX, 4C00H
INT 21H

START ENDP
```

1. $(2 \times 3) !=$

Se muestra como resultado 02D0 que traducido de hexadecimal en decimal corresponde con el valor 720= $5!=(2 \times 3)!$



2. $(2 \times 4) !=$

El resultado de la operación es 4.032×10^4 , que en hexadecimal corresponde con 9D80

3. $(3 \times 3) !=$

Overflow, nuevamente volvemos al caso de 9! cuyo resultado no entra en FFFF

4. $(5 \times 2) \neq$

Nuevamente el resultado es incorrecto, ya que se produce overflow

DOSBox 0.70, Cpu Cycles: 3000, Frameskip 0, Program: TD

File View Run Breakpoints Data Options Window Help

Module: factor File: C:\PRAO\FACTOR\factor.asm 74

MOV RESULT, AX ;Se pasa AX que contiene el valor del resultado a RESULT
MOV RESULT+2, DX

[CPU 80486] 3=

Address	Instruction	Comment	Register/Value
cs:002B	MOV AX, 4C00H		ax 5F00
cs:002E	INT 21H		bx 0000
cs:0030	MOV AX, 1		cx 0000
cs:0033	XOR CH, CH		dx 0000
cs:0035	CMP CX, 0		si 0000
cs:0038	JE FIN		di 0000
cs:003A	MUL CX		bp 0000
cs:003C	DEC CX		sp 0040
cs:003D	JNE IR		ds 480E
cs:003F	RET		es 480F

Watch

ss:0042 0209
ss:0040 7202

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu