Ejemplo A (if con and) Arbol

```
tdeclare
    real var1,var2;
tenddeclare

var1:=5

tif var1 >4 and var1!=8
      var1:=8
inelse
      var1:=10
outelse

fintif
```

```
nodo 1: --, var1, --
nodo 0: --, _5, --
nodo 2: 1, :=, 0
nodo 3: --, var1, --
nodo 4: --, _4, --
nodo 5: 3, >, 4
nodo 6: --, var1, --
nodo 7: --, _8, --
nodo 8: 6, !=, 7
nodo 9: 5, and, 8
nodo 11: --, var1, --
nodo 10: --, _8, --
nodo 12: 11, :=, 10
nodo 14: --, var1, --
nodo 13: --, _10, --
nodo 15: 14, :=, 13
nodo 16: 12, cuerpoif, 15
nodo 17: 9, tif, 16
nodo 18: 2, sentencia, 17

Raiz : nodo 18
```

```
.CODE
                    ;INICIALIZA EL SEGMENTO DE DATOS
                    MOV EAX, @DATA
                    MOV DS, EAX

                    ;CODIGO DEL PROGRAMA EN ASM
                    ;asignacion
                    mov eax, cte0
                    mov varasm_var1, eax
                    fld  varasm_var1
                    fld  cte1
                    fcomp   ;compara y popea el operando sobrante
                    ffree St(0)
                    fstsw ax
                    sahf
                    JB CONDIC_VERDADERA0    Jump if Below
                    mov result_comp1, 0
                    jmp CONDIC_FALSA1
CONDIC_VERDADERA0:
                    mov result_comp1, 1
CONDIC_FALSA1:
                    fld  varasm_var1
                    fld  cte2
                    fcomp   ;compara y popea el operando      sobrante
                    ffree St(0)
                    fstsw ax
                    sahf
                    JNE CONDIC_VERDADERA2
                    mov result_comp2, 0
                    jmp CONDIC_FALSA3
CONDIC_VERDADERA2:
                    mov result_comp2, 1
CONDIC_FALSA3:
                    mov eax, result_comp1
                    AND eax, result_comp2
                    mov result_logico3, eax
                    mov eax, result_logico3
                    cmp eax, 0
                    JE ELSE_IF4
                    ;asignacion
                    mov eax, cte2
                    mov varasm_var1, eax
                    JMP FIN_IF5
ELSE_IF4:
                    ;asignacion
                    mov eax, cte3
                    mov varasm_var1, eax
```

		FIN_IF5:
		
		;Espera pulsar una tecla
		MOV AH, 01
		iNT 21h
		
		;TERMINA LA EJECUCION DEL PROGRAMA
		TERMINAPROG:
		MOV EAX, 4C00h
		INT 21h
		.DATA
		varasm_var1 dd ?
		varasm_var2 dd ?
		cte0 dd 5
		cte1 dd 4
		cte2 dd 8
		cte3 dd 10
		result_comp1 dd ?
		result_comp2 dd ?
		result_logico3 dd ?
		END

Ejemplo B	**(inlist dentro de un if)**	**Arbol**

tdeclare
real var1;
tenddeclare
 	
var1:=3
 	
var1:=5
 	
tif tinlist(var1;[1;4;2])
var1:=39
 	
inelse
var1:=6
outelse
 	
fintif
 	
// Inlist es una sentencia que toma una expresión pivot y una lista de expresiones; y devuelve V o F si el resultado de la expresión pivot está o no
	nodo 1: --, var1, --
	nodo 0: --, _3, --
	nodo 2: 1, :=, 0
	nodo 4: --, var1, --
	nodo 3: --, _5, --
	nodo 5: 4, :=, 3
	nodo 6: 2, sentencia, 5
	nodo 13: --, var1, --
	nodo 11: --, _2, --
	nodo 9: --, _4, --
	nodo 7: --, _1, --
	nodo 8: 7, lista, --
	nodo 10: 9, lista, 8
	nodo 12: 11, lista, 10
	nodo 14: 13, tinlist, 12
	nodo 16: --,
		.CODE
		;INICIALIZA EL SEGMENTO DE DATOS
		MOV EAX, @DATA
		MOV DS, EAX
		
		;CODIGO DEL PROGRAMA EN ASM
		;asignacion
		mov eax, cte0
		mov varasm_var1, eax
		;asignacion
		mov eax, cte1
		mov varasm_var1, eax
		fld Dword Ptr ds:[varasm_var1]
		fld Dword Ptr ds:[cte2]
		fcomp ;compara y popea el operando sobrante
		ffree St(0)
		fstsw ax
		sahf
		JE CONDIC_VERDADERA0
		mov result_comp2, 0
		jmp CONDIC_FALSA1
		CONDIC_VERDADERA0:
		mov result_comp2, 1
		CONDIC_FALSA1:
		fld Dword Ptr ds:[varasm_var1]

en la lista	var1, --
	nodo 15: --, _39, --
	nodo 17: 16, :=, 15
	nodo 19: --, var1, --
	nodo 18: --, _6, --
	nodo 20: 19, :=, 18
	nodo 21: 17, cuerpoif, 20
	nodo 22: 14, tif, 21
	nodo 23: 6, sentencia, 22
		fld Dword Ptr ds:[cte2]
		fcomp ;compara y popea el operando sobrante
		ffree St(0)
		fstsw ax
		sahf
		JE CONDIC_VERDADERA2
		mov result_comp3, 0
		jmp CONDIC_FALSA3
		CONDIC_VERDADERA2:
		mov result_comp3, 1
		CONDIC_FALSA3:
		mov eax, result_comp3
		OR eax, result_comp2
		mov result_logico1, eax
		fld Dword Ptr ds:[varasm_var1]
		fld Dword Ptr ds:[cte3]
		fcomp ;compara y popea el operando sobrante
		ffree St(0)
		fstsw ax
		sahf
		JE CONDIC_VERDADERA4
		mov result_comp5, 0
		jmp CONDIC_FALSA5
		CONDIC_VERDADERA4:
		mov result_comp5, 1
		CONDIC_FALSA5:
		mov eax, result_comp5
		OR eax, result_logico1
		mov result_logico4, eax
		fld Dword Ptr ds:[varasm_var1]
		fld Dword Ptr ds:[cte4]
		fcomp ;compara y popea el operando sobrante
		ffree St(0)
		fstsw ax
		sahf
		JE CONDIC_VERDADERA6
		mov result_comp7, 0
		jmp CONDIC_FALSA7
		CONDIC_VERDADERA6:
		mov result_comp7, 1
		CONDIC_FALSA7:
		mov eax, result_comp7
		OR eax, result_logico4
		mov result_logico6, eax
		mov eax, result_logico6
		cmp eax, 0
		JE ELSE_IF8
		;asignacion
		mov eax, cte5
		mov varasm_var1, eax

		JMP FIN_IF9
		ELSE_IF8:
		;asignacion
		mov eax, cte6
		mov varasm_var1, eax
		FIN_IF9:
		
		;Espera pulsar una tecla
		MOV AH, 01
		INT 21h
		
		;TERMINA LA EJECUCION DEL PROGRAMA
		TERMINAPROG:
		MOV EAX, 4C00h
		INT 21h
		.DATA
		printBuffer db 14 dup (0);necesario para el uso interno de la rutina formatoReal
		varasm_var1 dd ?
		cte0 dd 3
		cte1 dd 5
		cte2 dd 1
		cte3 dd 4
		cte4 dd 2
		cte5 dd 39
		cte6 dd 6
		result_logico1 dd ?
		result_comp2 dd ?
		result_comp3 dd ?
		result_logico4 dd ?
		result_comp5 dd ?
		result_logico6 dd ?
		result_comp7 dd ?
		END

Ejemplo C	**(round y trunc)**	**Tercetos**

PROGRAM
  DECLARE
  REAL var1, var2, var3, var4;
  ENDDECLARE
 	
var1 := .9;
var2 := 6.9;
var3 := ROUND(var1);
var4 := TRUNC(var2);
 	
ENDPROGRAM
	1 (:=, var1, _cte_1)
	2 (:=, var2, _cte_2)
	3 (ROUND, var1, --)
	4 (:=, var3, [3])
	5 (TRUNC, var2, --)
	6 (:=, var4, [5])
		.MODEL LARGE
		.386
		.STACK 200h
		
		.CODE
		
		;CODIGO ASSEMBLER
		;----------
		
		mov ax,@DATA ;limpia buffer
		mov ds,ax
		mov es,ax
		
		FLD _cte_1
		FSTP var1
		FLD _cte_2
		FSTP var2
		FLD var1
		FRNDINT
		FSTP _aux_1
		FLD _aux_1
		FSTP var3
		FLD _uno
		FLD var2
		FPREM
		FSTP _resto_trunc
		FLD var2
		FSUB _resto_trunc
		FSTP _aux_1
		FLD _aux_1
		FSTP var4
		mov dx,OFFSET msgPRESIONE ;imprimir mensaje de espera
		mov ah,09
		int 21h
		mov ah,1 ;pausa, espera que oprima una tecla
		int 21h
		mov ax, 4C00h ;fin de ejecucion
		int 21h
		
		.DATA
		
		;DEFINICION DE VARIABLES
		;----------
		var1 dd 0
		var2 dd 0
		var3 dd 0
		var4 dd 0
		_cte_1 dd 0.900000
		_cte_2 dd 6.900000
		_aux_1 dd 0

		_aux_a dw 0 _aux_b dw 0 _uno dd 1.000000 _resto_trunc dd 0 msgPRESIONE db 0DH,0AH,"Presione una tecla para continuar...",'$' _NEWLINE db 0Dh,0Ah,'$' ;--------------------------------------- ;FIN DEFINICION DE VARIABLES END
Ejemplo D	**(STRING)**	**Polaca**

DECLARE
string pepe, pepe2;
ENDDECLARE
 	
pepe = 'HOLA '
 	
concatenate(pepe , 'MUNDO ')
 	
pepe2 = pepe
	HOLA
	pepe
	=
	MUNDO
	pepe
	CONCATENATE
	pepe
	pepe2
	=
		.MODEL LARGE
		.386
		.STACK 200h
		
		MAXTEXTSIZE equ 50
		
		.DATA
		
		pepe_ db MAXTEXTSIZE dup (?),'$'
		pepe2_ db MAXTEXTSIZE dup (?),'$'
		zv2 db "HOLA ",'$', 42 dup (?)
		zv3 db "MUNDO ",'$', 41 dup (?)
		aux db ?
		
		.CODE
		mov AX,@DATA ; inicializa el segmento de datos
		mov DS,AX
		mov es,ax ;
		mov si,OFFSET zv2
		mov di,OFFSET pepe_
		CALL COPIAR ; copia los strings
		mov si,OFFSET zv3
		mov di,OFFSET pepe_
		CALL CONCAT ; cat los strings
		mov si,OFFSET pepe_
		mov di,OFFSET pepe2_
		CALL COPIAR ; copia los strings

		mov ax, 4C00h ; Indica que debe terminar la ejecución int 21h
Ejemplo E	**(for)**	**Polaca**

DECLARE
real var2 , var1;
ENDDECLARE
 	
var1 = 4
var2 = 30
 	
for var1 = 1 to 3
{
var2 = var2 - 1
 	
}
next var1
	4
	Var1
	=
	30
	Var2
	=
	1
	Var1
	=
	_eti9:
	3
	Var1
	<>
	_eti27
	BF
	Var2
	1
	-
	Var2
	=
	Var1
	1
	+
	Var1
	=
	_eti9
	BI
	_eti27:
		.MODEL LARGE
		.386
		.STACK 200h
		
		MAXTEXTSIZE equ 50
		
		.DATA
		
		var2_ dd ?
		var1_ dd ?
		zv2 dd 040800000h;valor=4
		zv3 dd 041f00000h;valor=30
		zv4 dd 03f800000h;valor=1
		zv5 dd 040400000h;valor=3
		aux db ?
		
		.CODE
		mov AX,@DATA ; inicializa el segmento de datos
		mov DS,AX
		mov es,ax ;
		fld zv2
		fstp var1_
		fld zv3
		fstp var2_
		fld zv4
		fstp var1_
		_eti9:
		fld zv5

		fld var1_ fxch fcomp fstsw ax ffree st(0) sahf je _eti27 fld var2_ fld zv4 fsub fstp var2_ fld var1_ fld zv4 fadd fstp var1_ jmp _eti9 _eti27: mov ax, 4C00h ; Indica que debe terminar la ejecución int 21h
Ejemplo F	**(while con or)**	**Polaca**

DECLARE
real var2,var1;
 	
ENDDECLARE
 	
var1 = 4
var2 = 3
 	
while var2 <> 1 AND var1 > var2
{
var2 = var2 - 1
 	
if var2 == var1
{
WRITE('IF2 ')
}
endif
WRITE('POSTIF2 ')
 	
}
endwhile
	4
	var1
	=
	3
	Var2
	=
	_eti6:
	Var2
	1
	<>
	_eti34
	BF
	Var1
	Var2
	>
	_eti34
	BF
	Var2
	1
	-
	Var2
	=
	Var2
	var1
		.MODEL LARGE
		.386
		.STACK 200h
		
		MAXTEXTSIZE equ 50
		
		.DATA
		
		var2_ dd ?
		var1_ dd ?
		zv2 dd 040800000h;valor=4
		zv3 dd 040400000h;valor=3
		zv4 dd 03f800000h;valor=1
		zv5 db "IF2 ",'$', 43 dup (?)
		zv6 db "POSTIF2 ",'$', 39 dup (?)
		aux db ?
		
		.CODE
		mov AX,@DATA ; inicializa el segmento de datos
		mov DS,AX
		mov es,ax ;
		fld zv2
		fstp var1_
		fld zv3

	==
	_eti29
	BF
	“IF2”
	WRITE
	_eti29:
	“POSTIF2”
	WRITE
	_eti6
	BI
	_eti34:
		fstp var2_
		_eti6:
		fld var2_
		fld zv4
		fxch
		fcomp
		fstsw ax
		ffree st(0)
		sahf
		je _eti34
		fld var1_
		fld var2_
		fxch
		fcomp
		fstsw ax
		ffree st(0)
		sahf
		jbe _eti34
		fld var2_
		fld zv4
		fsub
		fstp var2_
		fld var2_
		fld var1_
		fxch
		fcomp
		fstsw ax
		ffree st(0)
		sahf
		jne _eti29
		mov DX, OFFSET zv5
		mov ah, 9 ; AH=9 es el servicio de impresion
		int 21h ; imprime
		_eti29:
		mov DX, OFFSET zv6
		mov ah, 9 ; AH=9 es el servicio de impresion
		int 21h ; imprime
		jmp _eti6
		
		_eti34:
		
		mov ax, 4C00h ; Indica que debe terminar la ejecución
		int 21h

Ejemplo G	**(average)**	**Polaca**

start
{
declare
a : real;
enddeclare
avg([1,4,0.4,3.2]);
}
end
	[0] rsv: avg, [1] 1, [2] 4, [3] +, [4] 0.4, [5] +, [6] 3.2, [7] +, [8] 4, [9] /,
	
	Tabla de Simbolos:
	Pos Nombre
	1 a
	2 _cte1
	3 _cte2
	4 _cte3
	5 _cte4
		.MODEL LARGE ; tipo del modelo de memoria usado.
		.386
		.STACK 200h ; bytes en el stack
		
		include fontdat.inc
		.CODE ;comienzo de la zona de codigo
		include fontaux.inc
		
		start:
		MOV EAX,@DATA ;inicializa el segmento de datos
		MOV DS,EAX
		MOV ES,EAX
		FINIT ;inicializamos la FPU
		FLD Dword Ptr ds:[_cte1]
		FLD Dword Ptr ds:[_cte2]
		
		; suma
		FADD
		FLD Dword Ptr ds:[_cte3]
		
		; suma
		FADD
		FLD Dword Ptr ds:[_cte4]
		
		; suma
		FADD
		FLD Dword Ptr ds:[_cte2]
		
		; division
		FDIV
		
		MOV EAX, 4C00h ; termina la ejecución.
		INT 21h
		END start;
		;

Ejemplo H	**(if y while)**	**Tercetos**
MAIN DEC int a,b real c string d ENDEC BEGINPROG a := 2 b := 1 while(a <= 5 AND b <= 1)	1 (_2 , _ , _) 2 (a , [1] , :=) 3 (_1 , _ , _) 4 (b , [3] , :=) 5 (_ , _ , WHILE) 6 (a , _ , _) 7 (_5 , _ , _) 8 ([6] , [7] , CMP) 9 (_ , _ , AND)	MODEL LARGE .386 .STACK 200h .DATA ;variables de la tabla de simbolos a dw ? msg_a db "a:",'$' aux_a db 10 dup(?),'$' b dw ? msg_b db "b:",'$'

do	10 (b , _ , _)	aux_b db 10 dup(?),'$'
<< 1	11 (_1 , _ , _)	c dd ?
if(b <= 2)	12 ([10] , [11] ,	msg_c db "c:",'$'
<< 1	CMP)	aux_c db 10 dup(?),'$'
b := 6	13 ([12] , [32] ,	d db 32 dup(?),'$'
<< b	BGT)	msg_d db "d:",'$'
fi	14 ([8] , [32] ,	MAXTEXTSIZE EQU 32
a := a + 1	BGT)	NEW_LINE db 0Dh,0Ah,'$'
<< a	15 (_1 , _ , <<)	_1_ dd 1; tiene que ser dd para operaciones aritmeticas
endwhile	16 (_ , _ , IF)	r0 dd ?
a:=0	17 (b , _ , _)	r1 dd ?
b:=0	18 (_2 , _ , _)	r2 dd ?
while(a <> 3 AND b<>3)	19 ([17] , [18] ,	r3 dd ?
do	CMP)	r4 dd ?
a := a + 1	20 ([19] , [25] ,	r5 dd ?
b := b + 1	BGT)	r6 dd ?
endwhile	21 (_1 , _ , <<)	r7 dd ?
<< a	22 (_6 , _ , _)	r8 dd ?
<< b	23 (b , [22] , :=)	r9 dd ?
ENDPROG	24 (b , _ , <<)	r10 dd ?
	25 (_ , _ , FI)	r11 dd ?
	26 (a , _ , _)	r12 dd ?
	27 (_1 , _ , _)	r13 dd ?
	28 ([26] , [27] ,	r14 dd ?
	+)	r15 dd ?
	29 (a , [28] , :=)	r16 dd ?
	30 (a , _ , <<)	r17 dd ?
	31 ([8] , _ , BI)	r18 dd ?
	32 (_ , _ ,	r19 dd ?
	ENDWHILE)	r20 dd ?
	33 (_0 , _ , _)	r21 dd ?
	34 (a , [33] , :=)	r22 dd ?
	35 (_0 , _ , _)	r23 dd ?
	36 (b , [35] , :=)	r24 dd ?
	37 (_ , _ ,	r25 dd ?
	WHILE)	r26 dd ?
	38 (a , _ , _)	r27 dd ?
	39 (_3 , _ , _)	r28 dd ?
	40 ([38] , [39] ,	r29 dd ?
	CMP)	r30 dd ?
	41 (_ , _ , AND)	r31 dd ?
	42 (b , _ , _)	r32 dd ?
	43 (_3 , _ , _)	r33 dd ?
	44 ([42] , [43] ,	r34 dd ?
	CMP)	r35 dd ?
	45 ([44] , [56] ,	r36 dd ?
	BEQ)	r37 dd ?
	46 ([40] , [56] ,	r38 dd ?
	BEQ)	r39 dd ?
	47 (a , _ , _)	r40 dd ?

	48 (_1 , _ , _) 49 ([47] , [48] , +) 50 (a , [49] , :=) 51 (b , _ , _) 52 (_1 , _ , _) 53 ([51] , [52] , +) 54 (b , [53] , :=) 55 ([40] , _ , BI) 56 (_ , _ , ENDWHILE) 57 (a , _ , <<) 58 (b , _ , <<)	r41 dd ? r42 dd ? r43 dd ? r44 dd ? r45 dd ? r46 dd ? r47 dd ? r48 dd ? r49 dd ? auxEstado dw ? _raux0 dd ? op0 dd ? _raux1 dd ? op1 dd ? _raux2 dd ? op2 dd ? _raux3 dd ? op3 dd ? _raux4 dd ? op4 dd ? _raux5 dd ? op5 dd ? _raux6 dd ? op6 dd ? _raux7 dd ? op7 dd ? _raux8 dd ? op8 dd ? _raux9 dd ? op9 dd ? _raux10 dd ? op10 dd ? _raux11 dd ? op11 dd ? _raux12 dd ? op12 dd ? _raux13 dd ? op13 dd ? _raux14 dd ? op14 dd ? _raux15 dd ? op15 dd ? _raux16 dd ? op16 dd ? _raux17 dd ? op17 dd ? _raux18 dd ? op18 dd ? _raux19 dd ? op19 dd ?

		_raux20 dd ? op20 dd ? _raux21 dd ? op21 dd ? _raux22 dd ? op22 dd ? _raux23 dd ? op23 dd ? _raux24 dd ? op24 dd ? _raux25 dd ? op25 dd ? _raux26 dd ? op26 dd ? _raux27 dd ? op27 dd ? _raux28 dd ? op28 dd ? _raux29 dd ? op29 dd ? _raux30 dd ? op30 dd ? _raux31 dd ? op31 dd ? _raux32 dd ? op32 dd ? _raux33 dd ? op33 dd ? _raux34 dd ? op34 dd ? _raux35 dd ? op35 dd ? _raux36 dd ? op36 dd ? _raux37 dd ? op37 dd ? _raux38 dd ? op38 dd ? _raux39 dd ? op39 dd ? _raux40 dd ? op40 dd ? _raux41 dd ? op41 dd ? _raux42 dd ? op42 dd ? _raux43 dd ? op43 dd ? _raux44 dd ? op44 dd ?

		_raux45 dd ? op45 dd ? _raux46 dd ? op46 dd ? _raux47 dd ? op47 dd ? _raux48 dd ? op48 dd ? _raux49 dd ? op49 dd ? _raux_cad1_ db 32 dup(?),'$' _raux_cad2_ db 32 dup(?),'$' _raux_cad3_ db 32 dup(?),'$' aux db 32 dup(?),'$' rS0 db 32 dup(?),'$' rS1 db 32 dup(?),'$' rS2 db 32 dup(?),'$' rS3 db 32 dup(?),'$' rS4 db 32 dup(?),'$' rS5 db 32 dup(?),'$' rS6 db 32 dup(?),'$' rS7 db 32 dup(?),'$' rS8 db 32 dup(?),'$' rS9 db 32 dup(?),'$' rS10 db 32 dup(?),'$' rS11 db 32 dup(?),'$' rS12 db 32 dup(?),'$' rS13 db 32 dup(?),'$' rS14 db 32 dup(?),'$' rS15 db 32 dup(?),'$' rS16 db 32 dup(?),'$' rS17 db 32 dup(?),'$' rS18 db 32 dup(?),'$' rS19 db 32 dup(?),'$' rS20 db 32 dup(?),'$' rS21 db 32 dup(?),'$' rS22 db 32 dup(?),'$' rS23 db 32 dup(?),'$' rS24 db 32 dup(?),'$' rS25 db 32 dup(?),'$' rS26 db 32 dup(?),'$' rS27 db 32 dup(?),'$' rS28 db 32 dup(?),'$' rS29 db 32 dup(?),'$' rS30 db 32 dup(?),'$' rS31 db 32 dup(?),'$' rS32 db 32 dup(?),'$' rS33 db 32 dup(?),'$' rS34 db 32 dup(?),'$' rS35 db 32 dup(?),'$'

```
rS36 db 32 dup(?),'$'
rS37 db 32 dup(?),'$'
rS38 db 32 dup(?),'$'
rS39 db 32 dup(?),'$'
rS40 db 32 dup(?),'$'
rS41 db 32 dup(?),'$'
rS42 db 32 dup(?),'$'
rS43 db 32 dup(?),'$'
rS44 db 32 dup(?),'$'
rS45 db 32 dup(?),'$'
rS46 db 32 dup(?),'$'
rS47 db 32 dup(?),'$'
rS48 db 32 dup(?),'$'
rS49 db 32 dup(?),'$'

.CODE

JMP MAIN
;************************************************************
; devuelve en BX la cantidad de caracteres que tiene un string
; DS:SI apunta al string.
;
STRLEN PROC
mov bx,0
STRL01:
cmp BYTE PTR [SI+BX],'$'
je STREND
inc BX
jmp STRL01
STREND:
ret
STRLEN ENDP
;*********************************************************************
; copia DS:SI a ES:DI; busca la cantidad de caracteres
;
COPIAR PROC
call STRLEN ; busco la cantidad de caracteres
cmp bx,MAXTEXTSIZE
jle COPIARSIZEOK
mov bx,MAXTEXTSIZE
COPIARSIZEOK:
mov cx,bx ; la copia se hace de ?X?caracteres
cld ; cld es para que la copia se realice
; hacia adelante
rep movsb ; copia la cadea
mov al,'$' ; car?ter terminador
mov BYTE PTR [DI],al ; el registro DI quedo apuntando al
; final
ret
COPIAR ENDP
```

```
;*****************************************************
; concatena DS:SI al final de ES:DI.
;
; busco el size del primer string
; sumo el size del segundo string
; si la suma excede MAXTEXTSIZE, copio solamente MAXTEXTSIZE caracteres
; si la suma NO excede MAXTEXTSIZE, copio el total de caracteres que tiene el
segundo string
;
CONCAT PROC
push ds
push si
call STRLEN ; busco la cantidad de caracteres del
;2do string
mov dx,bx ; guardo en DX la cantidad de caracteres
;en el origen.
mov si,di
push es
pop ds
call STRLEN ; tama#o del 1er string
add di,bx ; DI ya queda apuntando al final del
;primer string
add bx,dx ; tama#o total
cmp bx,MAXTEXTSIZE ; excede el tama#o maximo?
jg CONCATSIZEMAL
CONCATSIZEOK: ; La suma no excede el maximo, copio
; todos
mov cx,dx ; los caracteres del segundo string.
jmp CONCATSIGO
CONCATSIZEMAL: ; La suma de caracteres de los 2 strings
; exceden el maximo
sub bx,MAXTEXTSIZE
sub dx,bx
mov cx,dx ; copio lo maximo permitido el resto
;se pierde.
CONCATSIGO:
push ds
pop es
pop si
pop ds
cld ; cld es para que la copia se realice
;hacia adelante
rep movsb ; copia la cadea
mov al,'$' ; car?ter terminador
mov BYTE PTR [DI],al ; el registro DI quedo apuntando al
;final
ret
CONCAT ENDP

;********FIN*****FUNCIONES*****************
```

```
;*********************************************************
CARGAR PROC
GOBAK1:
MOV    AH,01H
INT    21H
CMP    AL,0DH
JE     BAK1
MOV    [SI],AL
INC    SI
JMP    GOBAK1
BAK1:
MOV    AL,'$'
MOV    [SI],AL
ret
CARGAR ENDP
;*********************************************************
;=============== Convertir numero a cadena ========ITOA=======
; Parametros
; ax: valor
; bx: donde guardar la cadena final
; Retorna
; cadena
itoa proc near
xor cx,cx  ;CX = 0

itoa_1:
cmp ax,0   ; El ciclo itoa_1 extrae los digitos del
je itoa_2  ; menos al mas significativo de AX y los
; guarda en el stack. Al finalizar el
xor dx,dx  ; ciclo el digito mas significativo esta
push bx    ; arriba del stack.
mov bx,10  ; CX contiene el numero de digitos
div bx
pop bx
push dx
inc cx
jmp itoa_1

itoa_2:
cmp cx,0    ; Esta seccion maneja el caso cuando
ja itoa_3   ; el numero a convertir (AX) es 0.
mov ax,'0'  ; En este caso, el ciclo anterior
mov [bx],ax ; no guarda valores en el stack y
inc bx      ; CX tiene el valor 0
jmp itoa_4

itoa_3:
pop ax      ; Extraemos los numero del stack
add ax,30h  ; lo pasamos a su valor ascii
```

```
mov [bx],ax ; lo guardamos en la cadena final
inc bx
loop itoa_3

itoa_4:
mov ax,'$'  ; terminar cadena con '$' para
mov [bx],ax ; imprimirla con la INT21h/AH=9
ret
itoa endp
;*************************************************
            ; ========= Convertir cadena a numero ===============ATOI======
; Parametros
; si: offset inicial de la cadena con respecto a DS
; Retorna
; bx: valor
atoi proc
 xor bx,bx   ;BX = 0

 atoi_1:
 lodsb      ;carga byte apuntado por SI en AL
           ;e incrementa si
 cmp al,'0'  ;es numero ascii? [0-9]
 jb noascii  ;no, salir
 cmp al,'9'
 ja noascii  ;no, salir

 sub al,30h  ;ascii '0'=30h, ascii '1'=31h...etc.
 cbw        ;byte a word
 push ax
 mov ax,bx   ;BX tendra el valor final
 mov cx,10
 mul cx      ;AX=AX*10
 mov bx,ax
 pop ax
 add bx,ax
 jmp atoi_1  ;seguir mientras SI apunte a un numero ascii
 noascii:
 ret        ;BX tiene el valor final
atoi endp
;FIN funciones

MAIN:
MOV AX,@DATA
MOV DS,AX
FINIT; Inicializa el coprocesador

MOV r0,2
FILD r0
FIST a
FFREE st(0)          ;FIN ASIGNACION
```

```
MOV r0,1
FILD r0
FIST b
FFREE st(0)         ;FIN ASIGNACION

WHILE0:             ;INICIO WHILE0
FILD a
FIST r0
FFREE st(0)
MOV r1,5
FILD r0
FILD r1
FCOMP
FSTSW ax
FWAIT
SAHF
FFREE st(0)
MOV auxEstado, ax
FILD b
FIST r2
FFREE st(0)
MOV r3,1
FILD r2
FILD r3
FCOMP
FSTSW ax
FWAIT
SAHF
FFREE st(0)
JB FIN_WHILE0
MOV ax,auxEstado
FWAIT
SAHF
JB FIN_WHILE0

;imprimiendo entero
mov ax,1
mov bx,offset _raux_cad1_
  call itoa
  MOV DX,OFFSET _raux_cad1_
  MOV Ah,9
  INT 21h
  MOV DX,OFFSET NEW_LINE
  MOV Ah,9
  INT 21h
FILD b
FIST r4
FFREE st(0)
MOV r5,2
```

```
FILD r4
FILD r5
FCOMP
FSTSW ax
FWAIT
SAHF
FFREE st(0)
JB FIN_IF0

;imprimiendo entero
mov ax,1
mov bx,offset _raux_cad1_
  call itoa
  MOV DX,OFFSET _raux_cad1_
  MOV Ah,9
  INT 21h
  MOV DX,OFFSET NEW_LINE
  MOV Ah,9
  INT 21h
MOV r6,6
FILD r6
FIST b
FFREE st(0)          ;FIN ASIGNACION


;imprimiendo entero
mov ax,b
mov bx,offset _raux_cad1_
  call itoa
  MOV DX,OFFSET _raux_cad1_
  MOV Ah,9
  INT 21h
  MOV DX,OFFSET NEW_LINE
  MOV Ah,9
  INT 21h
FI0:
FIN_IF0:             ;si no hay else
FILD a
FIST r6
FFREE st(0)
MOV r7,1
FILD r7
FILD r6
FADD
FIST r6
FFREE st(0)
FILD r6
FIST a
FFREE st(0)          ;FIN ASIGNACION
```

```
;imprimiendo entero
mov ax,a
mov bx,offset _raux_cad1_
  call itoa
  MOV DX,OFFSET _raux_cad1_
  MOV Ah,9
  INT 21h
  MOV DX,OFFSET NEW_LINE
  MOV Ah,9
  INT 21h
JMP WHILE0
FIN_WHILE0:     ;FIN_WHILE0
MOV r6,0
FILD r6
FIST a
FFREE st(0)     ;FIN ASIGNACION

MOV r6,0
FILD r6
FIST b
FFREE st(0)     ;FIN ASIGNACION

WHILE1:         ;INICIO WHILE1
FILD a
FIST r6
FFREE st(0)
MOV r7,3
FILD r6
FILD r7
FCOMP
FSTSW ax
FWAIT
SAHF
FFREE st(0)
MOV auxEstado, ax
FILD b
FIST r8
FFREE st(0)
MOV r9,3
FILD r8
FILD r9
FCOMP
FSTSW ax
FWAIT
SAHF
FFREE st(0)
JE FIN_WHILE1
MOV ax,auxEstado
FWAIT
```

```
SAHF
JE FIN_WHILE1
FILD a
FIST r10
FFREE st(0)
MOV r11,1
FILD r11
FILD r10
FADD
FIST r10
FFREE st(0)
FILD r10
FIST a
FFREE st(0)          ;FIN ASIGNACION

FILD b
FIST r10
FFREE st(0)
MOV r11,1
FILD r11
FILD r10
FADD
FIST r10
FFREE st(0)
FILD r10
FIST b
FFREE st(0)          ;FIN ASIGNACION

JMP WHILE1
FIN_WHILE1:       ;FIN_WHILE1

;imprimiendo entero
mov ax,a
mov bx,offset _raux_cad1_
  call itoa
  MOV DX,OFFSET _raux_cad1_
  MOV Ah,9
  INT 21h
  MOV DX,OFFSET NEW_LINE
  MOV Ah,9
  INT 21h

;imprimiendo entero
mov ax,b
mov bx,offset _raux_cad1_
  call itoa
  MOV DX,OFFSET _raux_cad1_
  MOV Ah,9
  INT 21h
  MOV DX,OFFSET NEW_LINE
```

		MOV Ah,9
		INT 21h
		
		FINAL:
		mov ah, 1 ; pausa, espera que oprima una tecla
		int 21h ; AH=1 es el servicio de lectura
		MOV AX, 4C00h ; Sale del Dos
		INT 21h ; Enviamos la interripcion 21h
		END ; final del archivo.