```
#*********************************************************************************
#  PARTE NO 1.
#
#  Utilizar el programa “HLA” para visualizar un mensaje
#*********************************************************************************

program helloWorld;
#include( "stdlib.hhf" );

begin helloWorld;

   stdout.put( "Hello, NOMBRE DEL ALUMNO", nl );

end helloWorld;


#*********************************************************************************
#  PARTE NO 2.
#
#  Ingresar variables, con valores previamente establecidos y no establecidos
#*********************************************************************************

Program DemoVars;
#include( "stdlib.hhf" )

static
   InitDemo:       int32 := DATO HEXADECIMAL ALUMNO;
   NotInitialized: int32;

begin DemoVars;

// Display the value of the pre-initialized variable:

stdout.put( "InitDemo value is ", InitDemo, nl );

// Imput an integer value from the user and display that value:

stdout.put( "Enter an integer value: " );
stdin.get( NotInitialized );
stdout.put( "You entered: ", NotInitialized, nl );

end DemoVars;


#*********************************************************************************
#  PARTE NO 3.
#
#  Ingresar datos hexadecimales. Sumar y restar, datos hexadecimales
#*********************************************************************************

program AddSub;
```

```
#include( "stdlib.hhf" )

begin AddSub;

    stdout.put( " Suma A + B y Resta A - B ", nl );
    stdout.put( "Meter dato hexadecimal  A  entre 00 y FF: " );
    stdin.get( al );

    stdout.put( "Meter dato hexadecimal  B  entre 00 y FF: " );
    stdin.get( bl );

// Suma A+B

    mov( al, cl );
    add( bl, cl );

    stdout.put( nl, "Valor A + B es: $", cl, nl );

// Resta A-B

    mov( al, cl );
    sub( bl, cl );

    stdout.put( nl, "Valor A - B es: $", cl, nl );

end AddSub;


#************************************************************************
#  PARTE NO 4.
#
#  Ingresar un dato decimal y convertirlo a hexadecimal
#************************************************************************

program ConvertToHex;
#include( "stdlib.hhf" )

static
    value: int32;

begin ConvertToHex;
    stdout.put( "Imput a decimal value:" );
    stdin.get( value );
    mov( value, eax );
    stdout.put( "The value ", value, " converted to hex is $", eax, nl );

end ConvertToHex;


#************************************************************************
#  PARTE NO 5.
```

```
#
#  Ingresar un dato hexadecimal y convertirlo a decimal
#********************************************************************************

program ConvertToDecimal;
#include( "stdlib.hhf" )
static
   value: int32;

begin ConvertToDecimal;

   stdout.put( "Imput a hexadecimal value: " );
   stdin.get( ebx );
   mov( ebx, value );
   stdout.put( "The value $", ebx, "convereted to decimal is ", value, nl );

end ConvertToDecimal;


#********************************************************************************
#  PARTE NO 6.
#
#  Ingresar datos hexadecimales. Operaciones lógicas And, Or, Xor y Not
#********************************************************************************

program LogicalOp;
#include( "stdlib.hhf" )

begin LogicalOp;

        stdout.put( "Imput left operand: " );
        stdin.get( eax );
        stdout.put( "Imput right operand: " );
        stdin.get( ebx );

        mov ( eax, ecx );
        and( ebx, ecx );
        stdout.put( "$", eax, " AND $", ebx, " = $", ecx, nl );

        mov( eax, ecx );
        or( ebx, ecx );
        stdout.put( "$", eax, "OR $", ebx, " = $", ecx, nl );

        mov( eax, ecx );
        xor( ebx, ecx );
        stdout.put( "$", eax, " XOR $", ebx, " =$", ecx, nl );

        mov( eax, ecx);
        not( ecx );
        stdout.put( "NOT $", ebx, " =$", ecx, nl );
```

```
        mov( ebx, ecx);
        not( ecx );
        stdout.put( "NOT $", ebx , "=$", ecx, nl );

end LogicalOp;


#********************************************************************************
#  PARTE NO 7.
#
#  Ingresar un dato decimal, convertirlo a hexadecimal y sacarle el complemento a dos
#********************************************************************************

program twosComplement;
#include( "stdlib.hhf" )

static
   PosValue:   int8;
   NegValue:   int8;

begin twosComplement;

   stdout.put( "Enter an integer between 0 and 127: " );
   stdin.get( PosValue );

   stdout.put( nl, "Value in hexadecimal: $" );
   stdout.putb( PosValue );

  mov( PosValue, al );
  not( al );
  stdout.put( nl, "Invert all the bits: $", al, nl );
  add( 1, al );
  stdout.put( "Add one: $", al, nl );
  mov( al, NegValue );
  stdout.put( "Result in decimal: ", NegValue, nl );

  stdout.put
  (
      nl,
      "Now do the same thing with the NEG instruction: ",
      nl
   );

   mov( PosValue, al );
   neg( al );
   mov( al, NegValue );

   stdout.put( "Hex result = $", al, nl );
   stdout.put( "Decimal result = ", NegValue, nl );

end twosComplement;
```

```
#*************************************************************************
#  PARTE NO 8.
#
#  Ingresar un dato hexadecimal de 8 bits, 16 bits y 32 bits, realizar la extensión de signo
#*************************************************************************

program SignExtension;
#include( "stdlib.hhf" )

static
        i8:     int8;
        i16:    int16;
        i32:    int32;

begin SignExtension;

        stdout.put( "Enter a small negative number: " );
        stdin.get( i8 );

        stdout.put( nl, "Sign extension using CBW and CWDE: ", nl, nl );

        mov( i8, al );
        stdout.put( "You entered ", i8, " ($", al, ")", nl );

        cbw();
        mov( ax, i16 );
        stdout.put( "16-bit sigh extension: ", i16, " ($", ax, ")", nl );

        cwde();
        mov( eax, i32 );
        stdout.put( "32-bit sign extension: ", i32, " ($", eax, ")", nl );

        stdout.put( nl, "Sign extension usign MOVSX:", nl, nl );

        movsx( i8, ax );
        mov( ax, i16 );
        stdout.put( "16-bit sign extension: ", i16, " ($", ax, ")", nl );

        movsx( i8, eax );
        mov( eax, i32 );
        stdout.put( "32-bit sign extension: ", i32, "($", eax, ")", nl );

end SignExtension;


#*************************************************************************
#  PARTE NO 9.
#
#  Ingresar un dato hexadecimal. Realiza: Shl(corrimiento a la izquierda con carry);
```

```
# Shr(corrimiento a la derecha con carry); Sar(corrimiento a la izquierda con carry y
# realimentación del bit más significativo); Rol(corrimiento a la izquierda con realimentación
# del bit más significativo al bit menos significativo); Ror(corrimiento a la derecha con
# realimentación del bit más significativo al bit menos significativo); Rcl(corrimiento a la
# izquierda con realimentación del bit más significativo al carry y al bit menos significativo);
# Rcr(corrimiento a la derecha con realimentación del bit más significativo al carry y al bit
# menos significativo
#********************************************************************************

program ShiftsRotation;
#include( "stdlib.hhf" )

begin ShiftsRotation;

    stdout.put( "Meter dato hexadecimal entre 00 y FF: " );
    stdin.get( al );

// "shl( cuantas rotaciones, registro a rotar) " corriemiento a la izquierda con carry
// ejemplo; shl( 1, bl ) rota una vez, shl( 2, bl ) rota dos veces

    mov( al, bl );
    shl( 1, bl );

    stdout.put( nl, "Valor original es: $", al, nl );
    stdout.put( nl, "Valor rotado a la izquierda es: $", bl, nl );

// "shr( cuantas rotaciones, registro a rotar) " corriemiento a la derecha con carry
// ejemplo; shr( 1, bl ) rota una vez, shr( 2, bl ) rota dos veces

    mov( al, bl );
    shr( 1, bl );

    stdout.put( nl, "Valor original es: $", al, nl );
    stdout.put( nl, "Valor rotado a la derecha es: $", bl, nl );

// "sar( cuantas rotaciones, registro a rotar) " corriemiento a la izquierda con carry y
// realimentación del bit más significativo
// ejemplo; sar( 1, bl ) rota una vez, sar( 2, bl ) rota dos veces

    mov( al, bl );
    sar( 1, bl );

    stdout.put( nl, "Valor original es: $", al, nl );
    stdout.put( nl, "Valor rotado a la izquierda es: $", bl, nl );

// "rol( cuantas rotaciones, registro a rotar) " corriemiento a la izquierda con realimentación
// del bit más significativo al bit menos significtivo
// ejemplo; rol( 1, bl ) rota una vez, rol( 2, bl ) rota dos veces

    mov( al, bl );
    rol( 1, bl );
```

```
    stdout.put( nl, "Valor original es: $", al, nl );
    stdout.put( nl, "Valor rotado a la izquierda es: $", bl, nl );

// "ror( cuantas rotaciones, registro a rotar) " corriemiento a la derecha con realimentación del
// bit más significativo al bit menos significtivo
// ejemplo; ror( 1, bl ) rota una vez, ror( 2, bl ) rota dos veces

    mov( al, bl );
    ror( 1, bl );

    stdout.put( nl, "Valor original es: $", al, nl );
    stdout.put( nl, "Valor rotado a la derecha es: $", bl, nl );

// "rcl( cuantas rotaciones, registro a rotar) " corriemiento a la izquierda con realimentación
// del bit más significativo al carry y al bit menos significtivo
// ejemplo; rcl( 1, bl ) rota una vez, rcl( 2, bl ) rota dos veces

    mov( al, bl );
    rcl( 1, bl );

    stdout.put( nl, "Valor original es: $", al, nl );
    stdout.put( nl, "Valor rotado a la izquierda es: $", bl, nl );

// "rcr( cuantas rotaciones, registro a rotar) " corriemiento a la derecha con realimentación del
// bit más significativo al carry y al bit menos significtivo
// ejemplo; rcr( 1, bl ) rota una vez, rcr( 2, bl ) rota dos veces

    mov( al, bl );
    rcr( 1, bl );

    stdout.put( nl, "Valor original es: $", al, nl );
    stdout.put( nl, "Valor rotado a la derecha es: $", bl, nl );

end ShiftsRotation;


#********************************************************************************
#  PARTE NO 10.
#
#  Direccionamiento Inmediato
#********************************************************************************

program DirecInm;
#include( "stdlib.hhf" )

 begin DirecInm;

     // Direccionamiento INMEDIATO

   stdout.put( nl, nl, "Direccionamiento INMEDIATO", nl );
```

```
    // Direccionamiento INMEDIATO a ocho bits

    mov( $DATO ALUMNO 8 BITS, al );
   stdout.put( nl, "mov($DATO ALUMNO 8 BITS, al)" );
   stdout.put( "     Valor de AL es: $", al, nl );

    // Direccionamiento INMEDIATO a diez y seis, bits

   mov( $DATO ALUMNO 16 BITS, ax );
   stdout.put( nl, "mov($DATO ALUMNO 16 BITS, ax)" );
   stdout.put( "    Valor de AX es: $", ax, nl );

    // Direccionamiento INMEDIATO a treinta y dos, bits

   mov( $DATO ALUMNO 32 BITS, eax );
   stdout.put( nl, "mov($DATO ALUMNO 32 BITS, eax)" );
   stdout.put( "   Valor de EAX es: $", eax, nl );

end DirecInm;


#*********************************************************************************
#  PARTE NO 11.
#
#  Direccionamiento de Registro
#*********************************************************************************

program DirecReg;

#include( "stdlib.hhf" )

 begin DirecReg;

     // Direccionamiento de REGISTRO

  stdout.put( nl, nl, "Direccionamiento de REGISTRO", nl );

    // Direccionamiento de REGISTRO a ocho bits

   mov( $DATO ALUMNO 8 BITS, bl );
   mov( bl, al);
   stdout.put( nl, "El valor de BL es: $DATO ALUMNO 8 BITS       mov(bl, al)" );
   stdout.put( "    Valor de AL es: $", al, nl );

    // Direccionamiento de REGISTRO a diez y seis, bits

   mov( $DATO ALUMNO 16 BITS, bx );
   mov( bx, ax);
   stdout.put( nl, "El valor de BX es: $DATO ALUMNO 16 BITS       mov(bx, ax)" );
   stdout.put( "    Valor de AX es: $", ax, nl );
```

```
    // Direccionamiento de REGISTRO a treinta y dos, bits

    mov( $DATO ALUMNO 32 BITS, ebx );
    mov( ebx, eax);
    stdout.put( nl, "El valor de EBX es: $ DATO ALUMNO 32 BITS     mov(ebx, eax)" );
    stdout.put( "   Valor de EAX es: $", eax, nl );

end DirecReg;


#////////////////////////////////////////////////////////////////////////
#  PARTE NO 12.
#
#  Direccionamiento Directo
#////////////////////////////////////////////////////////////////////////

program DirecDir;

#include( "stdlib.hhf" )

static
        M8b0:     int8;
        M8b1:     int8;
        M8b2:     int8;
        M8b3:     int8;
        M8b4:     int8;
        M8b5:     int8;
        M8b6:     int8;
        M8b7:     int8;
        M8b8:     int8;
        M8b9:     int8;
        M8bA:    int8;
        M8bB:    int8;
        M8bC:    int8;
        M8bD:    int8;
        M8bE:    int8;
        M8bF:    int8;

        M16b0:     int16;
        M16b1:     int16;
        M16b2:     int16;
        M16b3:     int16;
        M16b4:     int16;
        M16b5:     int16;
        M16b6:     int16;
        M16b7:     int16;
        M16b8:     int16;
        M16b9:     int16;
        M16bA:    int16;
        M16bB:    int16;
```

```
            M16bC:    int16;
            M16bD:    int16;
            M16bE:    int16;
            M16bF:    int16;

            M32b0:    int32;
            M32b1:    int32;
            M32b2:    int32;
            M32b3:    int32;
            M32b4:    int32;
            M32b5:    int32;
            M32b6:    int32;
            M32b7:    int32;
            M32b8:    int32;
            M32b9:    int32;
            M32bA:    int32;
            M32bB:    int32;
            M32bC:    int32;
            M32bD:    int32;
            M32bE:    int32;
            M32bF:    int32;

begin DirecDir;

   // PONER TABLAS DE DATOS

   stdout.put( nl, nl, "TABLAS DE DATOS EN MEMORIA", nl );

   // Cargar la tabla de ocho bits

   stdout.put( nl, "TABLA DE 8 BITS", nl );

   // Dirección:  M8b0    Dato: $DATO ALUMNO 8 BITS + 0
   // Dirección:  M8b1    Dato: $DATO ALUMNO 8 BITS + 1
   // Dirección:  M8b2    Dato: $DATO ALUMNO 8 BITS + 2
   // Dirección:  M8b3    Dato: $DATO ALUMNO 8 BITS + 3
   // Dirección:  M8b4    Dato: $DATO ALUMNO 8 BITS + 4
   // Dirección:  M8b5    Dato: $DATO ALUMNO 8 BITS + 5
   // Dirección:  M8b6    Dato: $DATO ALUMNO 8 BITS + 6
   // Dirección:  M8b7    Dato: $DATO ALUMNO 8 BITS + 7
   // Dirección:  M8b8    Dato: $DATO ALUMNO 8 BITS + 8
   // Dirección:  M8b9    Dato: $DATO ALUMNO 8 BITS + 9
   // Dirección:  M8bA    Dato: $DATO ALUMNO 8 BITS + A
   // Dirección:  M8bB    Dato: $DATO ALUMNO 8 BITS + B
   // Dirección:  M8bC    Dato: $DATO ALUMNO 8 BITS + C
   // Dirección:  M8bD    Dato: $DATO ALUMNO 8 BITS + D
   // Dirección:  M8bE    Dato: $DATO ALUMNO 8 BITS + E
   // Dirección:  M8bF    Dato: $DATO ALUMNO 8 BITS + F

   mov( $DATO ALUMNO 8 BITS + 0, al );
   mov( al, M8b0 );
```

```
mov( &M8b0, ebx );
stdout.put( nl, "La direccion de M8b0 es: $", ebx );
mov( M8b0, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 1, al );
mov( al, M8b1 );
mov( &M8b1, ebx );
stdout.put( nl, "La direccion de M8b1 es: $", ebx );
mov( M8b1, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 2, al );
mov( al, M8b2 );
mov( &M8b2, ebx );
stdout.put( nl, "La direccion de M8b2 es: $", ebx );
mov( M8b2, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 3, al );
mov( al, M8b3 );
mov( &M8b3, ebx );
stdout.put( nl, "La direccion de M8b3 es: $", ebx );
mov( M8b3, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 4, al );
mov( al, M8b4 );
mov( &M8b4, ebx );
stdout.put( nl, "La direccion de M8b4 es: $", ebx );
mov( M8b4, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 5, al );
mov( al, M8b5 );
mov( &M8b5, ebx );
stdout.put( nl, "La direccion de M8b5 es: $", ebx );
mov( M8b5, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 6, al );
mov( al, M8b6 );
mov( &M8b6, ebx );
stdout.put( nl, "La direccion de M8b6 es: $", ebx );
mov( M8b6, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 7, al );
mov( al, M8b7 );
mov( &M8b7, ebx );
stdout.put( nl, "La direccion de M8b7 es: $", ebx );
```

```
mov( M8b7, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 8, al );
mov( al, M8b8 );
mov( &M8b8, ebx );
stdout.put( nl, "La direccion de M8b8 es: $", ebx );
mov( M8b8, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 9, al );
mov( al, M8b9 );
mov( &M8b9, ebx );
stdout.put( nl, "La direccion de M8b9 es: $", ebx );
mov( M8b9, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + A, al );
mov( al, M8bA );
mov( &M8bA, ebx );
stdout.put( nl, "La direccion de M8bA es: $", ebx );
mov( M8bA, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + B, al );
mov( al, M8bB );
mov( &M8bB, ebx );
stdout.put( nl, "La direccion de M8bB es: $", ebx );
mov( M8bB, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + C, al );
mov( al, M8bC );
mov( &M8bC, ebx );
stdout.put( nl, "La direccion de M8bC es: $", ebx );
mov( M8bC, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + D, al );
mov( al, M8bD );
mov( &M8bD, ebx );
stdout.put( nl, "La direccion de M8bD es: $", ebx );
mov( M8bD, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + E, al );
mov( al, M8bE );
mov( &M8bE, ebx );
stdout.put( nl, "La direccion de M8bE es: $", ebx );
mov( M8bE, al );
stdout.put( ", El dato almacenado es: $", al, nl );
```

```
  mov( $DATO ALUMNO 8 BITS + F, al );
  mov( al, M8bF );
  mov( &M8bF, ebx );
  stdout.put( nl, "La direccion de M8bF es: $", ebx );
  mov( M8bF, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  // Cargar la tabla de diez y seis, bits

 stdout.put( nl, "TABLA DE 16 BITS", nl );

  // Dirección:  M16b0    Dato: $DATO ALUMNO 16 BITS + 0
  // Dirección:  M16b1    Dato: $DATO ALUMNO 16 BITS + 1
  // Dirección:  M16b2    Dato: $DATO ALUMNO 16 BITS + 2
  // Dirección:  M16b3    Dato: $DATO ALUMNO 16 BITS + 3
  // Dirección:  M16b4    Dato: $DATO ALUMNO 16 BITS + 4
  // Dirección:  M16b5    Dato: $DATO ALUMNO 16 BITS + 5
  // Dirección:  M16b6    Dato: $DATO ALUMNO 16 BITS + 6
  // Dirección:  M16b7    Dato: $DATO ALUMNO 16 BITS + 7
  // Dirección:  M16b8    Dato: $DATO ALUMNO 16 BITS + 8
  // Dirección:  M16b9    Dato: $DATO ALUMNO 16 BITS + 9
  // Dirección:  M16bA    Dato: $DATO ALUMNO 16 BITS + A
  // Dirección:  M16bB    Dato: $DATO ALUMNO 16 BITS + B
  // Dirección:  M16bC    Dato: $DATO ALUMNO 16 BITS + C
  // Dirección:  M16bD    Dato: $DATO ALUMNO 16 BITS + D
  // Dirección:  M16bE    Dato: $DATO ALUMNO 16 BITS + E
  // Dirección:  M16bF    Dato: $DATO ALUMNO 16 BITS + F

  mov( $DATO ALUMNO 16 BITS + 0, ax );
  mov( ax, M16b0 );
  mov( &M16b0, ebx );
  stdout.put( nl, "La direccion de M16b0 es: $", ebx );
  mov( M16b0, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 1, ax );
  mov( ax, M16b1 );
  mov( &M16b1, ebx );
  stdout.put( nl, "La direccion de M16b1 es: $", ebx );
  mov( M16b1, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 2, ax );
  mov( ax, M16b2 );
  mov( &M16b2, ebx );
  stdout.put( nl, "La direccion de M16b2 es: $", ebx );
  mov( M16b2, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 3, ax );
```

```
mov( ax, M16b3 );
mov( &M16b3, ebx );
stdout.put( nl, "La direccion de M16b3 es: $", ebx );
mov( M16b3, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 4, ax );
mov( ax, M16b4 );
mov( &M16b4, ebx );
stdout.put( nl, "La direccion de M16b4 es: $", ebx );
mov( M16b4, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 5, ax );
mov( ax, M16b5 );
mov( &M16b5, ebx );
stdout.put( nl, "La direccion de M16b5 es: $", ebx );
mov( M16b5, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 6, ax );
mov( ax, M16b6 );
mov( &M16b6, ebx );
stdout.put( nl, "La direccion de M16b6 es: $", ebx );
mov( M16b6, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 7, ax );
mov( ax, M16b7 );
mov( &M16b7, ebx );
stdout.put( nl, "La direccion de M16b7 es: $", ebx );
mov( M16b7, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 8, ax );
mov( ax, M16b8 );
mov( &M16b8, ebx );
stdout.put( nl, "La direccion de M16b8 es: $", ebx );
mov( M16b8, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 9, ax );
mov( ax, M16b9 );
mov( &M16b9, ebx );
stdout.put( nl, "La direccion de M16b9 es: $", ebx );
mov( M16b9, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + A, ax );
mov( ax, M16bA );
mov( &M16bA, ebx );
```

```
  stdout.put( nl, "La direccion de M16bA es: $", ebx );
  mov( M16bA, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + B, ax );
  mov( ax, M16bB );
  mov( &M16bB, ebx );
  stdout.put( nl, "La direccion de M16bB es: $", ebx );
  mov( M16bB, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + C, ax );
  mov( ax, M16bC );
  mov( &M16bC, ebx );
  stdout.put( nl, "La direccion de M16bC es: $", ebx );
  mov( M16bC, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + D, ax );
  mov( ax, M16bD );
  mov( &M16bD, ebx );
  stdout.put( nl, "La direccion de M16bD es: $", ebx );
  mov( M16bD, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + E, ax );
  mov( ax, M16bE );
  mov( &M16bE, ebx );
  stdout.put( nl, "La direccion de M16bE es: $", ebx );
  mov( M16bE, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + F, ax );
  mov( ax, M16bF );
  mov( &M16bF, ebx );
  stdout.put( nl, "La direccion de M16bF es: $", ebx );
  mov( M16bF, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  // Cargar la tabla de treinta y dos, bits

 stdout.put( nl, "TABLA DE 32 BITS", nl );

  // Dirección:  M32b0    Dato: $DATO ALUMNO 32 BITS + 0
  // Dirección:  M32b1    Dato: $DATO ALUMNO 32 BITS + 1
  // Dirección:  M32b2    Dato: $DATO ALUMNO 32 BITS + 2
  // Dirección:  M32b3    Dato: $DATO ALUMNO 32 BITS + 3
  // Dirección:  M32b4    Dato: $DATO ALUMNO 32 BITS + 4
  // Dirección:  M32b5    Dato: $DATO ALUMNO 32 BITS + 5
  // Dirección:  M32b6    Dato: $DATO ALUMNO 32 BITS + 6
  // Dirección:  M32b7    Dato: $DATO ALUMNO 32 BITS + 7
```

```
// Dirección:  M32b8    Dato: $DATO ALUMNO 32 BITS + 8
// Dirección:  M32b9    Dato: $DATO ALUMNO 32 BITS + 9
// Dirección:  M32bA    Dato: $DATO ALUMNO 32 BITS + A
// Dirección:  M32bB    Dato: $DATO ALUMNO 32 BITS + B
// Dirección:  M32bC    Dato: $DATO ALUMNO 32 BITS + C
// Dirección:  M32bD    Dato: $DATO ALUMNO 32 BITS + D
// Dirección:  M32bE    Dato: $DATO ALUMNO 32 BITS + E
// Dirección:  M32bF    Dato: $DATO ALUMNO 32 BITS + F

mov( $DATO ALUMNO 32 BITS + 0, eax );
mov( eax, M32b0 );
mov( &M32b0, ebx );
stdout.put( nl, "La direccion de M32b0 es: $", ebx );
mov( M32b0, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 1, eax );
mov( eax, M32b1 );
mov( &M32b1, ebx );
stdout.put( nl, "La direccion de M32b1 es: $", ebx );
mov( M32b1, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 2, eax );
mov( eax, M32b2 );
mov( &M32b2, ebx );
stdout.put( nl, "La direccion de M32b2 es: $", ebx );
mov( M32b2, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 3, eax );
mov( eax, M32b3 );
mov( &M32b3, ebx );
stdout.put( nl, "La direccion de M32b3 es: $", ebx );
mov( M32b3, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 4, eax );
mov(eax, M32b4 );
mov( &M32b4, ebx );
stdout.put( nl, "La direccion de M32b4 es: $", ebx );
mov( M32b4, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 5, eax );
mov( eax, M32b5 );
mov( &M32b5, ebx );
stdout.put( nl, "La direccion de M32b5 es: $", ebx );
mov( M32b5, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );
```

```
mov( $DATO ALUMNO 32 BITS + 6, eax );
mov( eax, M32b6 );
mov( &M32b6, ebx );
stdout.put( nl, "La direccion de M32b6 es: $", ebx );
mov( M32b6, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 7, eax );
mov( eax, M32b7 );
mov( &M32b7, ebx );
stdout.put( nl, "La direccion de M32b7 es: $", ebx );
mov( M32b7, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 8, eax );
mov( eax, M32b8 );
mov( &M32b8, ebx );
stdout.put( nl, "La direccion de M32b8 es: $", ebx );
mov( M32b8, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 9, eax );
mov( eax, M32b9 );
mov( &M32b9, ebx );
stdout.put( nl, "La direccion de M32b9 es: $", ebx );
mov( M32b9, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + A, eax );
mov( eax, M32bA );
mov( &M32bA, ebx );
stdout.put( nl, "La direccion de M32bA es: $", ebx );
mov( M32bA, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + B, eax );
mov( eax, M32bB );
mov( &M32bB, ebx );
stdout.put( nl, "La direccion de M32bB es: $", ebx );
mov( M32bB, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + C, eax );
mov( eax, M32bC );
mov( &M32bC, ebx );
stdout.put( nl, "La direccion de M32bC es: $", ebx );
mov( M32bC, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + D, eax );
mov( eax, M32bD );
```

```
    mov( &M32bD, ebx );
    stdout.put( nl, "La direccion de M32bD es: $", ebx );
    mov( M32bD, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + E, eax );
    mov( eax, M32bE );
    mov( &M32bE, ebx );
    stdout.put( nl, "La direccion de M32bE es: $", ebx );
    mov( M32bE, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + F, eax );
    mov( eax, M32bF );
    mov( &M32bF, ebx );
    stdout.put( nl, "La direccion de M32bF es: $", ebx );
    mov( M32bF, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    // Direccionamiento DIRECTO

   stdout.put( nl, nl, "Direccionamiento DIRECTO", nl );

    // Direccionamiento DIRECTO a ocho bits

    mov( M8b1, al );
    stdout.put( nl, "mov(M8b1, al)" );
    stdout.put( "     El dato en AL es: $", al, nl );

    // Direccionamiento DIRECTO a diez y seis, bits

    mov( M16b3, ax );
    stdout.put( nl, "mov(M16b3, ax)" );
    stdout.put( "     El dato en AX es: $", ax, nl );

    // Direccionamiento DIRECTO a treinta y dos, bits

    mov( M32bA, eax );
   stdout.put( nl, "mov(M32bA, eax)" );
    stdout.put( "     El dato en EAX es: $", eax, nl );

end DirecDir;


#////////////////////////////////////////////////////////////////////////
#  PARTE NO 13.
#
#  Direccionamiento Indirecto de Registro
#////////////////////////////////////////////////////////////////////////

program DirecIndReg;
```

```
#include( "stdlib.hhf" )

static
        M8b0:     int8;
        M8b1:     int8;
        M8b2:     int8;
        M8b3:     int8;
        M8b4:     int8;
        M8b5:     int8;
        M8b6:     int8;
        M8b7:     int8;
        M8b8:     int8;
        M8b9:     int8;
        M8bA:    int8;
        M8bB:    int8;
        M8bC:    int8;
        M8bD:    int8;
        M8bE:    int8;
        M8bF:    int8;

        M16b0:     int16;
        M16b1:     int16;
        M16b2:     int16;
        M16b3:     int16;
        M16b4:     int16;
        M16b5:     int16;
        M16b6:     int16;
        M16b7:     int16;
        M16b8:     int16;
        M16b9:     int16;
        M16bA:    int16;
        M16bB:    int16;
        M16bC:    int16;
        M16bD:    int16;
        M16bE:    int16;
        M16bF:    int16;

        M32b0:     int32;
        M32b1:     int32;
        M32b2:     int32;
        M32b3:     int32;
        M32b4:     int32;
        M32b5:     int32;
        M32b6:     int32;
        M32b7:     int32;
        M32b8:     int32;
        M32b9:     int32;
        M32bA:    int32;
        M32bB:    int32;
        M32bC:    int32;
```

```
        M32bD:    int32;
        M32bE:    int32;
        M32bF:    int32;

begin DirecIndReg;

  // PONER TABLAS DE DATOS

  stdout.put( nl, nl, "TABLAS DE DATOS EN MEMORIA", nl );

  // Cargar la tabla de ocho bits

  stdout.put( nl, "TABLA DE 8 BITS", nl );

  // Dirección:  M8b0    Dato: $DATO ALUMNO 8 BITS + 0
  // Dirección:  M8b1    Dato: $DATO ALUMNO 8 BITS + 1
  // Dirección:  M8b2    Dato: $DATO ALUMNO 8 BITS + 2
  // Dirección:  M8b3    Dato: $DATO ALUMNO 8 BITS + 3
  // Dirección:  M8b4    Dato: $DATO ALUMNO 8 BITS + 4
  // Dirección:  M8b5    Dato: $DATO ALUMNO 8 BITS + 5
  // Dirección:  M8b6    Dato: $DATO ALUMNO 8 BITS + 6
  // Dirección:  M8b7    Dato: $DATO ALUMNO 8 BITS + 7
  // Dirección:  M8b8    Dato: $DATO ALUMNO 8 BITS + 8
  // Dirección:  M8b9    Dato: $DATO ALUMNO 8 BITS + 9
  // Dirección:  M8bA    Dato: $DATO ALUMNO 8 BITS + A
  // Dirección:  M8bB    Dato: $DATO ALUMNO 8 BITS + B
  // Dirección:  M8bC    Dato: $DATO ALUMNO 8 BITS + C
  // Dirección:  M8bD    Dato: $DATO ALUMNO 8 BITS + D
  // Dirección:  M8bE    Dato: $DATO ALUMNO 8 BITS + E
  // Dirección:  M8bF    Dato: $DATO ALUMNO 8 BITS + F

  mov( $DATO ALUMNO 8 BITS + 0, al );
  mov( al, M8b0 );
  mov( &M8b0, ebx );
  stdout.put( nl, "La direccion de M8b0 es: $", ebx );
  mov( M8b0, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  mov( $DATO ALUMNO 8 BITS + 1, al );
  mov( al, M8b1 );
  mov( &M8b1, ebx );
  stdout.put( nl, "La direccion de M8b1 es: $", ebx );
  mov( M8b1, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  mov( $DATO ALUMNO 8 BITS + 2, al );
  mov( al, M8b2 );
  mov( &M8b2, ebx );
  stdout.put( nl, "La direccion de M8b2 es: $", ebx );
  mov( M8b2, al );
  stdout.put( ", El dato almacenado es: $", al, nl );
```

```
mov( $DATO ALUMNO 8 BITS + 3, al );
mov( al, M8b3 );
mov( &M8b3, ebx );
stdout.put( nl, "La direccion de M8b3 es: $", ebx );
mov( M8b3, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 4, al );
mov( al, M8b4 );
mov( &M8b4, ebx );
stdout.put( nl, "La direccion de M8b4 es: $", ebx );
mov( M8b4, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 5, al );
mov( al, M8b5 );
mov( &M8b5, ebx );
stdout.put( nl, "La direccion de M8b5 es: $", ebx );
mov( M8b5, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 6, al );
mov( al, M8b6 );
mov( &M8b6, ebx );
stdout.put( nl, "La direccion de M8b6 es: $", ebx );
mov( M8b6, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 7, al );
mov( al, M8b7 );
mov( &M8b7, ebx );
stdout.put( nl, "La direccion de M8b7 es: $", ebx );
mov( M8b7, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 8, al );
mov( al, M8b8 );
mov( &M8b8, ebx );
stdout.put( nl, "La direccion de M8b8 es: $", ebx );
mov( M8b8, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 9, al );
mov( al, M8b9 );
mov( &M8b9, ebx );
stdout.put( nl, "La direccion de M8b9 es: $", ebx );
mov( M8b9, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + A, al );
```

```
    mov( al, M8bA );
    mov( &M8bA, ebx );
    stdout.put( nl, "La direccion de M8bA es: $", ebx );
    mov( M8bA, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + B, al );
    mov( al, M8bB );
    mov( &M8bB, ebx );
    stdout.put( nl, "La direccion de M8bB es: $", ebx );
    mov( M8bB, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + C, al );
    mov( al, M8bC );
    mov( &M8bC, ebx );
    stdout.put( nl, "La direccion de M8bC es: $", ebx );
    mov( M8bC, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + D, al );
    mov( al, M8bD );
    mov( &M8bD, ebx );
    stdout.put( nl, "La direccion de M8bD es: $", ebx );
    mov( M8bD, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + E, al );
    mov( al, M8bE );
    mov( &M8bE, ebx );
    stdout.put( nl, "La direccion de M8bE es: $", ebx );
    mov( M8bE, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + F, al );
    mov( al, M8bF );
    mov( &M8bF, ebx );
    stdout.put( nl, "La direccion de M8bF es: $", ebx );
    mov( M8bF, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    // Cargar la tabla de diez y seis, bits

  stdout.put( nl, "TABLA DE 16 BITS", nl );

    // Dirección:  M16b0    Dato: $DATO ALUMNO 16 BITS + 0
    // Dirección:  M16b1    Dato: $DATO ALUMNO 16 BITS + 1
    // Dirección:  M16b2    Dato: $DATO ALUMNO 16 BITS + 2
    // Dirección:  M16b3    Dato: $DATO ALUMNO 16 BITS + 3
    // Dirección:  M16b4    Dato: $DATO ALUMNO 16 BITS + 4
    // Dirección:  M16b5    Dato: $DATO ALUMNO 16 BITS + 5
```

```
// Dirección:  M16b6    Dato: $DATO ALUMNO 16 BITS + 6
// Dirección:  M16b7    Dato: $DATO ALUMNO 16 BITS + 7
// Dirección:  M16b8    Dato: $DATO ALUMNO 16 BITS + 8
// Dirección:  M16b9    Dato: $DATO ALUMNO 16 BITS + 9
// Dirección:  M16bA    Dato: $DATO ALUMNO 16 BITS + A
// Dirección:  M16bB    Dato: $DATO ALUMNO 16 BITS + B
// Dirección:  M16bC    Dato: $DATO ALUMNO 16 BITS + C
// Dirección:  M16bD    Dato: $DATO ALUMNO 16 BITS + D
// Dirección:  M16bE    Dato: $DATO ALUMNO 16 BITS + E
// Dirección:  M16bF    Dato: $DATO ALUMNO 16 BITS + F

mov( $DATO ALUMNO 16 BITS + 0, ax );
mov( ax, M16b0 );
mov( &M16b0, ebx );
stdout.put( nl, "La direccion de M16b0 es: $", ebx );
mov( M16b0, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 1, ax );
mov( ax, M16b1 );
mov( &M16b1, ebx );
stdout.put( nl, "La direccion de M16b1 es: $", ebx );
mov( M16b1, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 2, ax );
mov( ax, M16b2 );
mov( &M16b2, ebx );
stdout.put( nl, "La direccion de M16b2 es: $", ebx );
mov( M16b2, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 3, ax );
mov( ax, M16b3 );
mov( &M16b3, ebx );
stdout.put( nl, "La direccion de M16b3 es: $", ebx );
mov( M16b3, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 4, ax );
mov( ax, M16b4 );
mov( &M16b4, ebx );
stdout.put( nl, "La direccion de M16b4 es: $", ebx );
mov( M16b4, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 5, ax );
mov( ax, M16b5 );
mov( &M16b5, ebx );
stdout.put( nl, "La direccion de M16b5 es: $", ebx );
mov( M16b5, ax );
```

```
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 6, ax );
mov( ax, M16b6 );
mov( &M16b6, ebx );
stdout.put( nl, "La direccion de M16b6 es: $", ebx );
mov( M16b6, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 7, ax );
mov( ax, M16b7 );
mov( &M16b7, ebx );
stdout.put( nl, "La direccion de M16b7 es: $", ebx );
mov( M16b7, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 8, ax );
mov( ax, M16b8 );
mov( &M16b8, ebx );
stdout.put( nl, "La direccion de M16b8 es: $", ebx );
mov( M16b8, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 9, ax );
mov( ax, M16b9 );
mov( &M16b9, ebx );
stdout.put( nl, "La direccion de M16b9 es: $", ebx );
mov( M16b9, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + A, ax );
mov( ax, M16bA );
mov( &M16bA, ebx );
stdout.put( nl, "La direccion de M16bA es: $", ebx );
mov( M16bA, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + B, ax );
mov( ax, M16bB );
mov( &M16bB, ebx );
stdout.put( nl, "La direccion de M16bB es: $", ebx );
mov( M16bB, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + C, ax );
mov( ax, M16bC );
mov( &M16bC, ebx );
stdout.put( nl, "La direccion de M16bC es: $", ebx );
mov( M16bC, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );
```

```
  mov( $DATO ALUMNO 16 BITS + D, ax );
  mov( ax, M16bD );
  mov( &M16bD, ebx );
  stdout.put( nl, "La direccion de M16bD es: $", ebx );
  mov( M16bD, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + E, ax );
  mov( ax, M16bE );
  mov( &M16bE, ebx );
  stdout.put( nl, "La direccion de M16bE es: $", ebx );
  mov( M16bE, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + F, ax );
  mov( ax, M16bF );
  mov( &M16bF, ebx );
  stdout.put( nl, "La direccion de M16bF es: $", ebx );
  mov( M16bF, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  // Cargar la tabla de treinta y dos, bits

 stdout.put( nl, "TABLA DE 32 BITS", nl );

  // Dirección:  M32b0    Dato: $DATO ALUMNO 32 BITS + 0
  // Dirección:  M32b1    Dato: $DATO ALUMNO 32 BITS + 1
  // Dirección:  M32b2    Dato: $DATO ALUMNO 32 BITS + 2
  // Dirección:  M32b3    Dato: $DATO ALUMNO 32 BITS + 3
  // Dirección:  M32b4    Dato: $DATO ALUMNO 32 BITS + 4
  // Dirección:  M32b5    Dato: $DATO ALUMNO 32 BITS + 5
  // Dirección:  M32b6    Dato: $DATO ALUMNO 32 BITS + 6
  // Dirección:  M32b7    Dato: $DATO ALUMNO 32 BITS + 7
  // Dirección:  M32b8    Dato: $DATO ALUMNO 32 BITS + 8
  // Dirección:  M32b9    Dato: $DATO ALUMNO 32 BITS + 9
  // Dirección:  M32bA    Dato: $DATO ALUMNO 32 BITS + A
  // Dirección:  M32bB    Dato: $DATO ALUMNO 32 BITS + B
  // Dirección:  M32bC    Dato: $DATO ALUMNO 32 BITS + C
  // Dirección:  M32bD    Dato: $DATO ALUMNO 32 BITS + D
  // Dirección:  M32bE    Dato: $DATO ALUMNO 32 BITS + E
  // Dirección:  M32bF    Dato: $DATO ALUMNO 32 BITS + F

  mov( $DATO ALUMNO 32 BITS + 0, eax );
  mov( eax, M32b0 );
  mov( &M32b0, ebx );
  stdout.put( nl, "La direccion de M32b0 es: $", ebx );
  mov( M32b0, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

  mov( $DATO ALUMNO 32 BITS + 1, eax );
  mov( eax, M32b1 );
```

```
mov( &M32b1, ebx );
stdout.put( nl, "La direccion de M32b1 es: $", ebx );
mov( M32b1, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 2, eax );
mov( eax, M32b2 );
mov( &M32b2, ebx );
stdout.put( nl, "La direccion de M32b2 es: $", ebx );
mov( M32b2, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 3, eax );
mov( eax, M32b3 );
mov( &M32b3, ebx );
stdout.put( nl, "La direccion de M32b3 es: $", ebx );
mov( M32b3, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 4, eax );
mov(eax, M32b4 );
mov( &M32b4, ebx );
stdout.put( nl, "La direccion de M32b4 es: $", ebx );
mov( M32b4, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 5, eax );
mov( eax, M32b5 );
mov( &M32b5, ebx );
stdout.put( nl, "La direccion de M32b5 es: $", ebx );
mov( M32b5, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 6, eax );
mov( eax, M32b6 );
mov( &M32b6, ebx );
stdout.put( nl, "La direccion de M32b6 es: $", ebx );
mov( M32b6, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 7, eax );
mov( eax, M32b7 );
mov( &M32b7, ebx );
stdout.put( nl, "La direccion de M32b7 es: $", ebx );
mov( M32b7, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 8, eax );
mov( eax, M32b8 );
mov( &M32b8, ebx );
stdout.put( nl, "La direccion de M32b8 es: $", ebx );
```

```
mov( M32b8, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 9, eax );
mov( eax, M32b9 );
mov( &M32b9, ebx );
stdout.put( nl, "La direccion de M32b9 es: $", ebx );
mov( M32b9, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + A, eax );
mov( eax, M32bA );
mov( &M32bA, ebx );
stdout.put( nl, "La direccion de M32bA es: $", ebx );
mov( M32bA, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + B, eax );
mov( eax, M32bB );
mov( &M32bB, ebx );
stdout.put( nl, "La direccion de M32bB es: $", ebx );
mov( M32bB, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + C, eax );
mov( eax, M32bC );
mov( &M32bC, ebx );
stdout.put( nl, "La direccion de M32bC es: $", ebx );
mov( M32bC, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + D, eax );
mov( eax, M32bD );
mov( &M32bD, ebx );
stdout.put( nl, "La direccion de M32bD es: $", ebx );
mov( M32bD, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + E, eax );
mov( eax, M32bE );
mov( &M32bE, ebx );
stdout.put( nl, "La direccion de M32bE es: $", ebx );
mov( M32bE, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + F, eax );
mov( eax, M32bF );
mov( &M32bF, ebx );
stdout.put( nl, "La direccion de M32bF es: $", ebx );
mov( M32bF, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );
```

```
    // Direccionamiento INDIRECTO DE REGISTRO

   stdout.put( nl, nl, "Direccionamiento INDIRECTO DE REGISTRO", nl );

    // Direccionamiento INDIRECTO DE REGISTRO a ocho bits

    mov( &M8b5, ebx );
    mov( [ebx], al );
    stdout.put( nl, "El dato en EBX es: M8b5     mov([ebx], al)" );
    stdout.put("     El dato en AL es: $", al, nl );

    // Direccionamiento INDIRECTO DE REGISTRO a diez y seis, bits

    mov( &M16b0, ebx );
    mov( [ebx], ax );
    stdout.put( nl, "El dato en EBX es: M16b0     mov([ebx], ax)" );
    stdout.put( "    El dato en AX es: $", ax, nl );

    // Direccionamiento INDIRECTO DE REGISTRO a treinta y dos, bits

    mov( &M32bF, ebx );
    mov( [ebx], eax );
    stdout.put( nl, "El dato en EBX es: M32bF     mov([ebx], eax)" );
    stdout.put( "    El dato en EAX es: $", eax, nl );

 end DirecIndReg;


#*********************************************************************************
#  PARTE NO 14.
#
#  Direccionamiento de Registro Relativo
#*********************************************************************************

program DirecRegRel;

#include( "stdlib.hhf" )

static
          M8b0:      int8;
          M8b1:      int8;
          M8b2:      int8;
          M8b3:      int8;
          M8b4:      int8;
          M8b5:      int8;
          M8b6:      int8;
          M8b7:      int8;
          M8b8:      int8;
          M8b9:      int8;
          M8bA:     int8;
```

```
        M8bB:     int8;
        M8bC:     int8;
        M8bD:     int8;
        M8bE:     int8;
        M8bF:     int8;

        M16b0:     int16;
        M16b1:     int16;
        M16b2:     int16;
        M16b3:     int16;
        M16b4:     int16;
        M16b5:     int16;
        M16b6:     int16;
        M16b7:     int16;
        M16b8:     int16;
        M16b9:     int16;
        M16bA:     int16;
        M16bB:     int16;
        M16bC:     int16;
        M16bD:     int16;
        M16bE:     int16;
        M16bF:     int16;

        M32b0:     int32;
        M32b1:     int32;
        M32b2:     int32;
        M32b3:     int32;
        M32b4:     int32;
        M32b5:     int32;
        M32b6:     int32;
        M32b7:     int32;
        M32b8:     int32;
        M32b9:     int32;
        M32bA:     int32;
        M32bB:     int32;
        M32bC:     int32;
        M32bD:     int32;
        M32bE:     int32;
        M32bF:     int32;

begin DirecRegRel;

  // PONER TABLAS DE DATOS

  stdout.put( nl, nl, "TABLAS DE DATOS EN MEMORIA", nl );

  // Cargar la tabla de ocho bits

  stdout.put( nl, "TABLA DE 8 BITS", nl );

  // Dirección:  M8b0    Dato: $DATO ALUMNO 8 BITS + 0
```

```
// Dirección:  M8b1    Dato: $DATO ALUMNO 8 BITS + 1
// Dirección:  M8b2    Dato: $DATO ALUMNO 8 BITS + 2
// Dirección:  M8b3    Dato: $DATO ALUMNO 8 BITS + 3
// Dirección:  M8b4    Dato: $DATO ALUMNO 8 BITS + 4
// Dirección:  M8b5    Dato: $DATO ALUMNO 8 BITS + 5
// Dirección:  M8b6    Dato: $DATO ALUMNO 8 BITS + 6
// Dirección:  M8b7    Dato: $DATO ALUMNO 8 BITS + 7
// Dirección:  M8b8    Dato: $DATO ALUMNO 8 BITS + 8
// Dirección:  M8b9    Dato: $DATO ALUMNO 8 BITS + 9
// Dirección:  M8bA    Dato: $DATO ALUMNO 8 BITS + A
// Dirección:  M8bB    Dato: $DATO ALUMNO 8 BITS + B
// Dirección:  M8bC    Dato: $DATO ALUMNO 8 BITS + C
// Dirección:  M8bD    Dato: $DATO ALUMNO 8 BITS + D
// Dirección:  M8bE    Dato: $DATO ALUMNO 8 BITS + E
// Dirección:  M8bF    Dato: $DATO ALUMNO 8 BITS + F

mov( $DATO ALUMNO 8 BITS + 0, al );
mov( al, M8b0 );
mov( &M8b0, ebx );
stdout.put( nl, "La direccion de M8b0 es: $", ebx );
mov( M8b0, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 1, al );
mov( al, M8b1 );
mov( &M8b1, ebx );
stdout.put( nl, "La direccion de M8b1 es: $", ebx );
mov( M8b1, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 2, al );
mov( al, M8b2 );
mov( &M8b2, ebx );
stdout.put( nl, "La direccion de M8b2 es: $", ebx );
mov( M8b2, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 3, al );
mov( al, M8b3 );
mov( &M8b3, ebx );
stdout.put( nl, "La direccion de M8b3 es: $", ebx );
mov( M8b3, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 4, al );
mov( al, M8b4 );
mov( &M8b4, ebx );
stdout.put( nl, "La direccion de M8b4 es: $", ebx );
mov( M8b4, al );
stdout.put( ", El dato almacenado es: $", al, nl );
```

```
mov( $DATO ALUMNO 8 BITS + 5, al );
mov( al, M8b5 );
mov( &M8b5, ebx );
stdout.put( nl, "La direccion de M8b5 es: $", ebx );
mov( M8b5, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 6, al );
mov( al, M8b6 );
mov( &M8b6, ebx );
stdout.put( nl, "La direccion de M8b6 es: $", ebx );
mov( M8b6, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 7, al );
mov( al, M8b7 );
mov( &M8b7, ebx );
stdout.put( nl, "La direccion de M8b7 es: $", ebx );
mov( M8b7, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 8, al );
mov( al, M8b8 );
mov( &M8b8, ebx );
stdout.put( nl, "La direccion de M8b8 es: $", ebx );
mov( M8b8, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 9, al );
mov( al, M8b9 );
mov( &M8b9, ebx );
stdout.put( nl, "La direccion de M8b9 es: $", ebx );
mov( M8b9, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + A, al );
mov( al, M8bA );
mov( &M8bA, ebx );
stdout.put( nl, "La direccion de M8bA es: $", ebx );
mov( M8bA, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + B, al );
mov( al, M8bB );
mov( &M8bB, ebx );
stdout.put( nl, "La direccion de M8bB es: $", ebx );
mov( M8bB, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + C, al );
mov( al, M8bC );
```

```
  mov( &M8bC, ebx );
  stdout.put( nl, "La direccion de M8bC es: $", ebx );
  mov( M8bC, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  mov( $DATO ALUMNO 8 BITS + D, al );
  mov( al, M8bD );
  mov( &M8bD, ebx );
  stdout.put( nl, "La direccion de M8bD es: $", ebx );
  mov( M8bD, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  mov( $DATO ALUMNO 8 BITS + E, al );
  mov( al, M8bE );
  mov( &M8bE, ebx );
  stdout.put( nl, "La direccion de M8bE es: $", ebx );
  mov( M8bE, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  mov( $DATO ALUMNO 8 BITS + F, al );
  mov( al, M8bF );
  mov( &M8bF, ebx );
  stdout.put( nl, "La direccion de M8bF es: $", ebx );
  mov( M8bF, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  // Cargar la tabla de diez y seis, bits

stdout.put( nl, "TABLA DE 16 BITS", nl );

  // Dirección:  M16b0    Dato: $DATO ALUMNO 16 BITS + 0
  // Dirección:  M16b1    Dato: $DATO ALUMNO 16 BITS + 1
  // Dirección:  M16b2    Dato: $DATO ALUMNO 16 BITS + 2
  // Dirección:  M16b3    Dato: $DATO ALUMNO 16 BITS + 3
  // Dirección:  M16b4    Dato: $DATO ALUMNO 16 BITS + 4
  // Dirección:  M16b5    Dato: $DATO ALUMNO 16 BITS + 5
  // Dirección:  M16b6    Dato: $DATO ALUMNO 16 BITS + 6
  // Dirección:  M16b7    Dato: $DATO ALUMNO 16 BITS + 7
  // Dirección:  M16b8    Dato: $DATO ALUMNO 16 BITS + 8
  // Dirección:  M16b9    Dato: $DATO ALUMNO 16 BITS + 9
  // Dirección:  M16bA    Dato: $DATO ALUMNO 16 BITS + A
  // Dirección:  M16bB    Dato: $DATO ALUMNO 16 BITS + B
  // Dirección:  M16bC    Dato: $DATO ALUMNO 16 BITS + C
  // Dirección:  M16bD    Dato: $DATO ALUMNO 16 BITS + D
  // Dirección:  M16bE    Dato: $DATO ALUMNO 16 BITS + E
  // Dirección:  M16bF    Dato: $DATO ALUMNO 16 BITS + F

  mov( $DATO ALUMNO 16 BITS + 0, ax );
  mov( ax, M16b0 );
  mov( &M16b0, ebx );
  stdout.put( nl, "La direccion de M16b0 es: $", ebx );
```

```
mov( M16b0, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 1, ax );
mov( ax, M16b1 );
mov( &M16b1, ebx );
stdout.put( nl, "La direccion de M16b1 es: $", ebx );
mov( M16b1, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 2, ax );
mov( ax, M16b2 );
mov( &M16b2, ebx );
stdout.put( nl, "La direccion de M16b2 es: $", ebx );
mov( M16b2, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 3, ax );
mov( ax, M16b3 );
mov( &M16b3, ebx );
stdout.put( nl, "La direccion de M16b3 es: $", ebx );
mov( M16b3, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 4, ax );
mov( ax, M16b4 );
mov( &M16b4, ebx );
stdout.put( nl, "La direccion de M16b4 es: $", ebx );
mov( M16b4, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 5, ax );
mov( ax, M16b5 );
mov( &M16b5, ebx );
stdout.put( nl, "La direccion de M16b5 es: $", ebx );
mov( M16b5, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 6, ax );
mov( ax, M16b6 );
mov( &M16b6, ebx );
stdout.put( nl, "La direccion de M16b6 es: $", ebx );
mov( M16b6, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 7, ax );
mov( ax, M16b7 );
mov( &M16b7, ebx );
stdout.put( nl, "La direccion de M16b7 es: $", ebx );
mov( M16b7, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );
```

```
mov( $DATO ALUMNO 16 BITS + 8, ax );
mov( ax, M16b8 );
mov( &M16b8, ebx );
stdout.put( nl, "La direccion de M16b8 es: $", ebx );
mov( M16b8, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 9, ax );
mov( ax, M16b9 );
mov( &M16b9, ebx );
stdout.put( nl, "La direccion de M16b9 es: $", ebx );
mov( M16b9, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + A, ax );
mov( ax, M16bA );
mov( &M16bA, ebx );
stdout.put( nl, "La direccion de M16bA es: $", ebx );
mov( M16bA, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + B, ax );
mov( ax, M16bB );
mov( &M16bB, ebx );
stdout.put( nl, "La direccion de M16bB es: $", ebx );
mov( M16bB, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + C, ax );
mov( ax, M16bC );
mov( &M16bC, ebx );
stdout.put( nl, "La direccion de M16bC es: $", ebx );
mov( M16bC, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + D, ax );
mov( ax, M16bD );
mov( &M16bD, ebx );
stdout.put( nl, "La direccion de M16bD es: $", ebx );
mov( M16bD, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + E, ax );
mov( ax, M16bE );
mov( &M16bE, ebx );
stdout.put( nl, "La direccion de M16bE es: $", ebx );
mov( M16bE, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + F, ax );
```

```
  mov( ax, M16bF );
  mov( &M16bF, ebx );
  stdout.put( nl, "La direccion de M16bF es: $", ebx );
  mov( M16bF, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  // Cargar la tabla de treinta y dos, bits

 stdout.put( nl, "TABLA DE 32 BITS", nl );

  // Dirección:  M32b0    Dato: $DATO ALUMNO 32 BITS + 0
  // Dirección:  M32b1    Dato: $DATO ALUMNO 32 BITS + 1
  // Dirección:  M32b2    Dato: $DATO ALUMNO 32 BITS + 2
  // Dirección:  M32b3    Dato: $DATO ALUMNO 32 BITS + 3
  // Dirección:  M32b4    Dato: $DATO ALUMNO 32 BITS + 4
  // Dirección:  M32b5    Dato: $DATO ALUMNO 32 BITS + 5
  // Dirección:  M32b6    Dato: $DATO ALUMNO 32 BITS + 6
  // Dirección:  M32b7    Dato: $DATO ALUMNO 32 BITS + 7
  // Dirección:  M32b8    Dato: $DATO ALUMNO 32 BITS + 8
  // Dirección:  M32b9    Dato: $DATO ALUMNO 32 BITS + 9
  // Dirección:  M32bA    Dato: $DATO ALUMNO 32 BITS + A
  // Dirección:  M32bB    Dato: $DATO ALUMNO 32 BITS + B
  // Dirección:  M32bC    Dato: $DATO ALUMNO 32 BITS + C
  // Dirección:  M32bD    Dato: $DATO ALUMNO 32 BITS + D
  // Dirección:  M32bE    Dato: $DATO ALUMNO 32 BITS + E
  // Dirección:  M32bF    Dato: $DATO ALUMNO 32 BITS + F

  mov( $DATO ALUMNO 32 BITS + 0, eax );
  mov( eax, M32b0 );
  mov( &M32b0, ebx );
  stdout.put( nl, "La direccion de M32b0 es: $", ebx );
  mov( M32b0, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

  mov( $DATO ALUMNO 32 BITS + 1, eax );
  mov( eax, M32b1 );
  mov( &M32b1, ebx );
  stdout.put( nl, "La direccion de M32b1 es: $", ebx );
  mov( M32b1, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

  mov( $DATO ALUMNO 32 BITS + 2, eax );
  mov( eax, M32b2 );
  mov( &M32b2, ebx );
  stdout.put( nl, "La direccion de M32b2 es: $", ebx );
  mov( M32b2, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

  mov( $DATO ALUMNO 32 BITS + 3, eax );
  mov( eax, M32b3 );
  mov( &M32b3, ebx );
```

```
stdout.put( nl, "La direccion de M32b3 es: $", ebx );
mov( M32b3, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 4, eax );
mov(eax, M32b4 );
mov( &M32b4, ebx );
stdout.put( nl, "La direccion de M32b4 es: $", ebx );
mov( M32b4, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 5, eax );
mov( eax, M32b5 );
mov( &M32b5, ebx );
stdout.put( nl, "La direccion de M32b5 es: $", ebx );
mov( M32b5, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 6, eax );
mov( eax, M32b6 );
mov( &M32b6, ebx );
stdout.put( nl, "La direccion de M32b6 es: $", ebx );
mov( M32b6, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 7, eax );
mov( eax, M32b7 );
mov( &M32b7, ebx );
stdout.put( nl, "La direccion de M32b7 es: $", ebx );
mov( M32b7, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 8, eax );
mov( eax, M32b8 );
mov( &M32b8, ebx );
stdout.put( nl, "La direccion de M32b8 es: $", ebx );
mov( M32b8, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 9, eax );
mov( eax, M32b9 );
mov( &M32b9, ebx );
stdout.put( nl, "La direccion de M32b9 es: $", ebx );
mov( M32b9, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + A, eax );
mov( eax, M32bA );
mov( &M32bA, ebx );
stdout.put( nl, "La direccion de M32bA es: $", ebx );
mov( M32bA, eax );
```

```
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + B, eax );
    mov( eax, M32bB );
    mov( &M32bB, ebx );
    stdout.put( nl, "La direccion de M32bB es: $", ebx );
    mov( M32bB, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + C, eax );
    mov( eax, M32bC );
    mov( &M32bC, ebx );
    stdout.put( nl, "La direccion de M32bC es: $", ebx );
    mov( M32bC, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + D, eax );
    mov( eax, M32bD );
    mov( &M32bD, ebx );
    stdout.put( nl, "La direccion de M32bD es: $", ebx );
    mov( M32bD, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + E, eax );
    mov( eax, M32bE );
    mov( &M32bE, ebx );
    stdout.put( nl, "La direccion de M32bE es: $", ebx );
    mov( M32bE, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + F, eax );
    mov( eax, M32bF );
    mov( &M32bF, ebx );
    stdout.put( nl, "La direccion de M32bF es: $", ebx );
    mov( M32bF, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

// Direccionamiento REGISTRO RELATIVO

   stdout.put( nl, nl, "Direccionamiento REGISTRO RELATIVO", nl );

    // Direccionamiento REGISTRO RELATIVO a ocho bits

    mov( &M8b1, ebx );
    mov( [ ebx + $00000002 ], al );
    stdout.put( nl, "El dato en EBX es: M8b1     mov([ebx+02], al)", nl );
    stdout.put( "Direccion final es:M8b3      El dato en AL es: $", al, nl );

    // Direccionamiento REGISTRO RELATIVO a diez y seis, bits

    // Múltiplos de dos las direcciones.
```

```
    stdout.put( nl, "Las direcciones son multiplos de dos", nl );

    mov( &M16b2, ebx );
    mov( [ ebx + $00000008 ], ax );
    stdout.put( "El dato en EBX es: M16b2      mov([ebx+$08], ax)", nl );
    stdout.put( "Direccion final es:M16b6      El dato en AX es: $", ax, nl );

    // Direccionamiento REGISTRO RELATIVO a treinta y dos, bits

    // Múltiplo de cuatro las direcciones.

    stdout.put( nl,"Las direcciones son multiplos de cuatro", nl );

    mov( &M32b5, ebx );
    mov( [ ebx + $0000000C ], eax );
    stdout.put( "El dato en EBX es: M32b5      mov([ebx+$0C], eax)", nl );
    stdout.put( "Direccion final es:M32b8      El dato en EAX es: $", eax, nl );

 end DirecRegRel;


#********************************************************************************
#  PARTE NO 15.
#
#  Direccionamiento Indice
#********************************************************************************

program DirecInd;

#include( "stdlib.hhf" )

static
          M8b0:      int8;
          M8b1:      int8;
          M8b2:      int8;
          M8b3:      int8;
          M8b4:      int8;
          M8b5:      int8;
          M8b6:      int8;
          M8b7:      int8;
          M8b8:      int8;
          M8b9:      int8;
          M8bA:     int8;
          M8bB:     int8;
          M8bC:     int8;
          M8bD:     int8;
          M8bE:     int8;
          M8bF:     int8;

          M16b0:      int16;
```

```
        M16b1:      int16;
        M16b2:      int16;
        M16b3:      int16;
        M16b4:      int16;
        M16b5:      int16;
        M16b6:      int16;
        M16b7:      int16;
        M16b8:      int16;
        M16b9:      int16;
        M16bA:     int16;
        M16bB:     int16;
        M16bC:     int16;
        M16bD:     int16;
        M16bE:     int16;
        M16bF:     int16;

        M32b0:      int32;
        M32b1:      int32;
        M32b2:      int32;
        M32b3:      int32;
        M32b4:      int32;
        M32b5:      int32;
        M32b6:      int32;
        M32b7:      int32;
        M32b8:      int32;
        M32b9:      int32;
        M32bA:     int32;
        M32bB:     int32;
        M32bC:     int32;
        M32bD:     int32;
        M32bE:     int32;
        M32bF:     int32;

begin DirecInd;

  // PONER TABLAS DE DATOS

  stdout.put( nl, nl, "TABLAS DE DATOS EN MEMORIA", nl );

  // Cargar la tabla de ocho bits

  stdout.put( nl, "TABLA DE 8 BITS", nl );

  // Dirección:  M8b0    Dato: $DATO ALUMNO 8 BITS + 0
  // Dirección:  M8b1    Dato: $DATO ALUMNO 8 BITS + 1
  // Dirección:  M8b2    Dato: $DATO ALUMNO 8 BITS + 2
  // Dirección:  M8b3    Dato: $DATO ALUMNO 8 BITS + 3
  // Dirección:  M8b4    Dato: $DATO ALUMNO 8 BITS + 4
  // Dirección:  M8b5    Dato: $DATO ALUMNO 8 BITS + 5
  // Dirección:  M8b6    Dato: $DATO ALUMNO 8 BITS + 6
  // Dirección:  M8b7    Dato: $DATO ALUMNO 8 BITS + 7
```

```
// Dirección:  M8b8    Dato: $DATO ALUMNO 8 BITS + 8
// Dirección:  M8b9    Dato: $DATO ALUMNO 8 BITS + 9
// Dirección:  M8bA    Dato: $DATO ALUMNO 8 BITS + A
// Dirección:  M8bB    Dato: $DATO ALUMNO 8 BITS + B
// Dirección:  M8bC    Dato: $DATO ALUMNO 8 BITS + C
// Dirección:  M8bD    Dato: $DATO ALUMNO 8 BITS + D
// Dirección:  M8bE    Dato: $DATO ALUMNO 8 BITS + E
// Dirección:  M8bF    Dato: $DATO ALUMNO 8 BITS + F

mov( $DATO ALUMNO 8 BITS + 0, al );
mov( al, M8b0 );
mov( &M8b0, ebx );
stdout.put( nl, "La direccion de M8b0 es: $", ebx );
mov( M8b0, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 1, al );
mov( al, M8b1 );
mov( &M8b1, ebx );
stdout.put( nl, "La direccion de M8b1 es: $", ebx );
mov( M8b1, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 2, al );
mov( al, M8b2 );
mov( &M8b2, ebx );
stdout.put( nl, "La direccion de M8b2 es: $", ebx );
mov( M8b2, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 3, al );
mov( al, M8b3 );
mov( &M8b3, ebx );
stdout.put( nl, "La direccion de M8b3 es: $", ebx );
mov( M8b3, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 4, al );
mov( al, M8b4 );
mov( &M8b4, ebx );
stdout.put( nl, "La direccion de M8b4 es: $", ebx );
mov( M8b4, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 5, al );
mov( al, M8b5 );
mov( &M8b5, ebx );
stdout.put( nl, "La direccion de M8b5 es: $", ebx );
mov( M8b5, al );
stdout.put( ", El dato almacenado es: $", al, nl );
```

```
mov( $DATO ALUMNO 8 BITS + 6, al );
mov( al, M8b6 );
mov( &M8b6, ebx );
stdout.put( nl, "La direccion de M8b6 es: $", ebx );
mov( M8b6, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 7, al );
mov( al, M8b7 );
mov( &M8b7, ebx );
stdout.put( nl, "La direccion de M8b7 es: $", ebx );
mov( M8b7, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 8, al );
mov( al, M8b8 );
mov( &M8b8, ebx );
stdout.put( nl, "La direccion de M8b8 es: $", ebx );
mov( M8b8, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 9, al );
mov( al, M8b9 );
mov( &M8b9, ebx );
stdout.put( nl, "La direccion de M8b9 es: $", ebx );
mov( M8b9, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + A, al );
mov( al, M8bA );
mov( &M8bA, ebx );
stdout.put( nl, "La direccion de M8bA es: $", ebx );
mov( M8bA, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + B, al );
mov( al, M8bB );
mov( &M8bB, ebx );
stdout.put( nl, "La direccion de M8bB es: $", ebx );
mov( M8bB, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + C, al );
mov( al, M8bC );
mov( &M8bC, ebx );
stdout.put( nl, "La direccion de M8bC es: $", ebx );
mov( M8bC, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + D, al );
mov( al, M8bD );
```

```
    mov( &M8bD, ebx );
    stdout.put( nl, "La direccion de M8bD es: $", ebx );
    mov( M8bD, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + E, al );
    mov( al, M8bE );
    mov( &M8bE, ebx );
    stdout.put( nl, "La direccion de M8bE es: $", ebx );
    mov( M8bE, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + F, al );
    mov( al, M8bF );
    mov( &M8bF, ebx );
    stdout.put( nl, "La direccion de M8bF es: $", ebx );
    mov( M8bF, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    // Cargar la tabla de diez y seis, bits

  stdout.put( nl, "TABLA DE 16 BITS", nl );

    // Dirección:  M16b0    Dato: $DATO ALUMNO 16 BITS + 0
    // Dirección:  M16b1    Dato: $DATO ALUMNO 16 BITS + 1
    // Dirección:  M16b2    Dato: $DATO ALUMNO 16 BITS + 2
    // Dirección:  M16b3    Dato: $DATO ALUMNO 16 BITS + 3
    // Dirección:  M16b4    Dato: $DATO ALUMNO 16 BITS + 4
    // Dirección:  M16b5    Dato: $DATO ALUMNO 16 BITS + 5
    // Dirección:  M16b6    Dato: $DATO ALUMNO 16 BITS + 6
    // Dirección:  M16b7    Dato: $DATO ALUMNO 16 BITS + 7
    // Dirección:  M16b8    Dato: $DATO ALUMNO 16 BITS + 8
    // Dirección:  M16b9    Dato: $DATO ALUMNO 16 BITS + 9
    // Dirección:  M16bA    Dato: $DATO ALUMNO 16 BITS + A
    // Dirección:  M16bB    Dato: $DATO ALUMNO 16 BITS + B
    // Dirección:  M16bC    Dato: $DATO ALUMNO 16 BITS + C
    // Dirección:  M16bD    Dato: $DATO ALUMNO 16 BITS + D
    // Dirección:  M16bE    Dato: $DATO ALUMNO 16 BITS + E
    // Dirección:  M16bF    Dato: $DATO ALUMNO 16 BITS + F

    mov( $DATO ALUMNO 16 BITS + 0, ax );
    mov( ax, M16b0 );
    mov( &M16b0, ebx );
    stdout.put( nl, "La direccion de M16b0 es: $", ebx );
    mov( M16b0, ax );
    stdout.put( ", El dato almacenado es: $", ax, nl );

    mov( $DATO ALUMNO 16 BITS + 1, ax );
    mov( ax, M16b1 );
    mov( &M16b1, ebx );
    stdout.put( nl, "La direccion de M16b1 es: $", ebx );
```

```
mov( M16b1, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 2, ax );
mov( ax, M16b2 );
mov( &M16b2, ebx );
stdout.put( nl, "La direccion de M16b2 es: $", ebx );
mov( M16b2, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 3, ax );
mov( ax, M16b3 );
mov( &M16b3, ebx );
stdout.put( nl, "La direccion de M16b3 es: $", ebx );
mov( M16b3, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 4, ax );
mov( ax, M16b4 );
mov( &M16b4, ebx );
stdout.put( nl, "La direccion de M16b4 es: $", ebx );
mov( M16b4, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 5, ax );
mov( ax, M16b5 );
mov( &M16b5, ebx );
stdout.put( nl, "La direccion de M16b5 es: $", ebx );
mov( M16b5, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 6, ax );
mov( ax, M16b6 );
mov( &M16b6, ebx );
stdout.put( nl, "La direccion de M16b6 es: $", ebx );
mov( M16b6, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 7, ax );
mov( ax, M16b7 );
mov( &M16b7, ebx );
stdout.put( nl, "La direccion de M16b7 es: $", ebx );
mov( M16b7, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 8, ax );
mov( ax, M16b8 );
mov( &M16b8, ebx );
stdout.put( nl, "La direccion de M16b8 es: $", ebx );
mov( M16b8, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );
```

```
mov( $DATO ALUMNO 16 BITS + 9, ax );
mov( ax, M16b9 );
mov( &M16b9, ebx );
stdout.put( nl, "La direccion de M16b9 es: $", ebx );
mov( M16b9, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + A, ax );
mov( ax, M16bA );
mov( &M16bA, ebx );
stdout.put( nl, "La direccion de M16bA es: $", ebx );
mov( M16bA, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + B, ax );
mov( ax, M16bB );
mov( &M16bB, ebx );
stdout.put( nl, "La direccion de M16bB es: $", ebx );
mov( M16bB, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + C, ax );
mov( ax, M16bC );
mov( &M16bC, ebx );
stdout.put( nl, "La direccion de M16bC es: $", ebx );
mov( M16bC, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + D, ax );
mov( ax, M16bD );
mov( &M16bD, ebx );
stdout.put( nl, "La direccion de M16bD es: $", ebx );
mov( M16bD, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + E, ax );
mov( ax, M16bE );
mov( &M16bE, ebx );
stdout.put( nl, "La direccion de M16bE es: $", ebx );
mov( M16bE, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + F, ax );
mov( ax, M16bF );
mov( &M16bF, ebx );
stdout.put( nl, "La direccion de M16bF es: $", ebx );
mov( M16bF, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

// Cargar la tabla de treinta y dos, bits
```

```
stdout.put( nl, "TABLA DE 32 BITS", nl );

// Dirección:  M32b0    Dato: $DATO ALUMNO 32 BITS + 0
// Dirección:  M32b1    Dato: $DATO ALUMNO 32 BITS + 1
// Dirección:  M32b2    Dato: $DATO ALUMNO 32 BITS + 2
// Dirección:  M32b3    Dato: $DATO ALUMNO 32 BITS + 3
// Dirección:  M32b4    Dato: $DATO ALUMNO 32 BITS + 4
// Dirección:  M32b5    Dato: $DATO ALUMNO 32 BITS + 5
// Dirección:  M32b6    Dato: $DATO ALUMNO 32 BITS + 6
// Dirección:  M32b7    Dato: $DATO ALUMNO 32 BITS + 7
// Dirección:  M32b8    Dato: $DATO ALUMNO 32 BITS + 8
// Dirección:  M32b9    Dato: $DATO ALUMNO 32 BITS + 9
// Dirección:  M32bA    Dato: $DATO ALUMNO 32 BITS + A
// Dirección:  M32bB    Dato: $DATO ALUMNO 32 BITS + B
// Dirección:  M32bC    Dato: $DATO ALUMNO 32 BITS + C
// Dirección:  M32bD    Dato: $DATO ALUMNO 32 BITS + D
// Dirección:  M32bE    Dato: $DATO ALUMNO 32 BITS + E
// Dirección:  M32bF    Dato: $DATO ALUMNO 32 BITS + F

mov( $DATO ALUMNO 32 BITS + 0, eax );
mov( eax, M32b0 );
mov( &M32b0, ebx );
stdout.put( nl, "La direccion de M32b0 es: $", ebx );
mov( M32b0, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 1, eax );
mov( eax, M32b1 );
mov( &M32b1, ebx );
stdout.put( nl, "La direccion de M32b1 es: $", ebx );
mov( M32b1, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 2, eax );
mov( eax, M32b2 );
mov( &M32b2, ebx );
stdout.put( nl, "La direccion de M32b2 es: $", ebx );
mov( M32b2, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 3, eax );
mov( eax, M32b3 );
mov( &M32b3, ebx );
stdout.put( nl, "La direccion de M32b3 es: $", ebx );
mov( M32b3, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 4, eax );
mov(eax, M32b4 );
mov( &M32b4, ebx );
```

```
stdout.put( nl, "La direccion de M32b4 es: $", ebx );
mov( M32b4, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 5, eax );
mov( eax, M32b5 );
mov( &M32b5, ebx );
stdout.put( nl, "La direccion de M32b5 es: $", ebx );
mov( M32b5, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 6, eax );
mov( eax, M32b6 );
mov( &M32b6, ebx );
stdout.put( nl, "La direccion de M32b6 es: $", ebx );
mov( M32b6, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 7, eax );
mov( eax, M32b7 );
mov( &M32b7, ebx );
stdout.put( nl, "La direccion de M32b7 es: $", ebx );
mov( M32b7, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 8, eax );
mov( eax, M32b8 );
mov( &M32b8, ebx );
stdout.put( nl, "La direccion de M32b8 es: $", ebx );
mov( M32b8, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 9, eax );
mov( eax, M32b9 );
mov( &M32b9, ebx );
stdout.put( nl, "La direccion de M32b9 es: $", ebx );
mov( M32b9, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + A, eax );
mov( eax, M32bA );
mov( &M32bA, ebx );
stdout.put( nl, "La direccion de M32bA es: $", ebx );
mov( M32bA, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + B, eax );
mov( eax, M32bB );
mov( &M32bB, ebx );
stdout.put( nl, "La direccion de M32bB es: $", ebx );
mov( M32bB, eax );
```

```
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + C, eax );
    mov( eax, M32bC );
    mov( &M32bC, ebx );
    stdout.put( nl, "La direccion de M32bC es: $", ebx );
    mov( M32bC, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + D, eax );
    mov( eax, M32bD );
    mov( &M32bD, ebx );
    stdout.put( nl, "La direccion de M32bD es: $", ebx );
    mov( M32bD, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + E, eax );
    mov( eax, M32bE );
    mov( &M32bE, ebx );
    stdout.put( nl, "La direccion de M32bE es: $", ebx );
    mov( M32bE, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + F, eax );
    mov( eax, M32bF );
    mov( &M32bF, ebx );
    stdout.put( nl, "La direccion de M32bF es: $", ebx );
    mov( M32bF, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

// Direccionamiento CON ÍNDICE

   stdout.put( nl, nl, "Direccionamiento CON INDICE", nl );

    // Direccionamiento CON ÍNDICE a ocho bits

    mov( &M8b0, edi );
    mov( [ edi ], al );
    stdout.put( nl, "El dato en EDI es: M8b0     mov([edi], al)" );
    stdout.put( "      El dato en AL es: $", al, nl );

   // Direccionamiento CON ÍNDICE a diez y seis, bits

    // Múltiplos de dos las direcciones.

    mov( &M16bB, edi );
    mov( [ edi ], ax );
    stdout.put( nl, "El dato en EDI es: M16bB     mov([edi], ax)" );
    stdout.put( "     El dato en AX es: $", ax, nl );

   // Direccionamiento CON ÍNDICE a treinta y dos, bits
```

```
    // Múltiplo de cuatro las direcciones.

    mov( &M32b7, edi );
    mov( [ edi ], eax );
    stdout.put( nl, "El dato en EDI es: M32b7    mov([edi], eax)" );
    stdout.put( "     El dato en EAX es: $", eax, nl );

end DirecInd;


#********************************************************************************
#  PARTE NO 15.
#
#  Direccionamiento Base más Indice
#********************************************************************************

program DirecBaseMasInd;

#include( "stdlib.hhf" )

static
        M8b0:      int8;
        M8b1:      int8;
        M8b2:      int8;
        M8b3:      int8;
        M8b4:      int8;
        M8b5:      int8;
        M8b6:      int8;
        M8b7:      int8;
        M8b8:      int8;
        M8b9:      int8;
        M8bA:     int8;
        M8bB:     int8;
        M8bC:     int8;
        M8bD:     int8;
        M8bE:     int8;
        M8bF:     int8;

        M16b0:      int16;
        M16b1:      int16;
        M16b2:      int16;
        M16b3:      int16;
        M16b4:      int16;
        M16b5:      int16;
        M16b6:      int16;
        M16b7:      int16;
        M16b8:      int16;
        M16b9:      int16;
        M16bA:     int16;
        M16bB:     int16;
```

```
            M16bC:     int16;
            M16bD:     int16;
            M16bE:     int16;
            M16bF:     int16;

            M32b0:     int32;
            M32b1:     int32;
            M32b2:     int32;
            M32b3:     int32;
            M32b4:     int32;
            M32b5:     int32;
            M32b6:     int32;
            M32b7:     int32;
            M32b8:     int32;
            M32b9:     int32;
            M32bA:     int32;
            M32bB:     int32;
            M32bC:     int32;
            M32bD:     int32;
            M32bE:     int32;
            M32bF:     int32;

begin DirecBaseMasInd;

   // PONER TABLAS DE DATOS

   stdout.put( nl, nl, "TABLAS DE DATOS EN MEMORIA", nl );

   // Cargar la tabla de ocho bits

   stdout.put( nl, "TABLA DE 8 BITS", nl );

   // Dirección:  M8b0    Dato: $DATO ALUMNO 8 BITS + 0
   // Dirección:  M8b1    Dato: $DATO ALUMNO 8 BITS + 1
   // Dirección:  M8b2    Dato: $DATO ALUMNO 8 BITS + 2
   // Dirección:  M8b3    Dato: $DATO ALUMNO 8 BITS + 3
   // Dirección:  M8b4    Dato: $DATO ALUMNO 8 BITS + 4
   // Dirección:  M8b5    Dato: $DATO ALUMNO 8 BITS + 5
   // Dirección:  M8b6    Dato: $DATO ALUMNO 8 BITS + 6
   // Dirección:  M8b7    Dato: $DATO ALUMNO 8 BITS + 7
   // Dirección:  M8b8    Dato: $DATO ALUMNO 8 BITS + 8
   // Dirección:  M8b9    Dato: $DATO ALUMNO 8 BITS + 9
   // Dirección:  M8bA    Dato: $DATO ALUMNO 8 BITS + A
   // Dirección:  M8bB    Dato: $DATO ALUMNO 8 BITS + B
   // Dirección:  M8bC    Dato: $DATO ALUMNO 8 BITS + C
   // Dirección:  M8bD    Dato: $DATO ALUMNO 8 BITS + D
   // Dirección:  M8bE    Dato: $DATO ALUMNO 8 BITS + E
   // Dirección:  M8bF    Dato: $DATO ALUMNO 8 BITS + F

   mov( $DATO ALUMNO 8 BITS + 0, al );
   mov( al, M8b0 );
```

```
mov( &M8b0, ebx );
stdout.put( nl, "La direccion de M8b0 es: $", ebx );
mov( M8b0, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 1, al );
mov( al, M8b1 );
mov( &M8b1, ebx );
stdout.put( nl, "La direccion de M8b1 es: $", ebx );
mov( M8b1, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 2, al );
mov( al, M8b2 );
mov( &M8b2, ebx );
stdout.put( nl, "La direccion de M8b2 es: $", ebx );
mov( M8b2, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 3, al );
mov( al, M8b3 );
mov( &M8b3, ebx );
stdout.put( nl, "La direccion de M8b3 es: $", ebx );
mov( M8b3, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 4, al );
mov( al, M8b4 );
mov( &M8b4, ebx );
stdout.put( nl, "La direccion de M8b4 es: $", ebx );
mov( M8b4, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 5, al );
mov( al, M8b5 );
mov( &M8b5, ebx );
stdout.put( nl, "La direccion de M8b5 es: $", ebx );
mov( M8b5, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 6, al );
mov( al, M8b6 );
mov( &M8b6, ebx );
stdout.put( nl, "La direccion de M8b6 es: $", ebx );
mov( M8b6, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 7, al );
mov( al, M8b7 );
mov( &M8b7, ebx );
stdout.put( nl, "La direccion de M8b7 es: $", ebx );
```

```
mov( M8b7, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 8, al );
mov( al, M8b8 );
mov( &M8b8, ebx );
stdout.put( nl, "La direccion de M8b8 es: $", ebx );
mov( M8b8, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 9, al );
mov( al, M8b9 );
mov( &M8b9, ebx );
stdout.put( nl, "La direccion de M8b9 es: $", ebx );
mov( M8b9, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + A, al );
mov( al, M8bA );
mov( &M8bA, ebx );
stdout.put( nl, "La direccion de M8bA es: $", ebx );
mov( M8bA, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + B, al );
mov( al, M8bB );
mov( &M8bB, ebx );
stdout.put( nl, "La direccion de M8bB es: $", ebx );
mov( M8bB, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + C, al );
mov( al, M8bC );
mov( &M8bC, ebx );
stdout.put( nl, "La direccion de M8bC es: $", ebx );
mov( M8bC, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + D, al );
mov( al, M8bD );
mov( &M8bD, ebx );
stdout.put( nl, "La direccion de M8bD es: $", ebx );
mov( M8bD, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + E, al );
mov( al, M8bE );
mov( &M8bE, ebx );
stdout.put( nl, "La direccion de M8bE es: $", ebx );
mov( M8bE, al );
stdout.put( ", El dato almacenado es: $", al, nl );
```

```
  mov( $DATO ALUMNO 8 BITS + F, al );
  mov( al, M8bF );
  mov( &M8bF, ebx );
  stdout.put( nl, "La direccion de M8bF es: $", ebx );
  mov( M8bF, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  // Cargar la tabla de diez y seis, bits

 stdout.put( nl, "TABLA DE 16 BITS", nl );

  // Dirección:  M16b0    Dato: $DATO ALUMNO 16 BITS + 0
  // Dirección:  M16b1    Dato: $DATO ALUMNO 16 BITS + 1
  // Dirección:  M16b2    Dato: $DATO ALUMNO 16 BITS + 2
  // Dirección:  M16b3    Dato: $DATO ALUMNO 16 BITS + 3
  // Dirección:  M16b4    Dato: $DATO ALUMNO 16 BITS + 4
  // Dirección:  M16b5    Dato: $DATO ALUMNO 16 BITS + 5
  // Dirección:  M16b6    Dato: $DATO ALUMNO 16 BITS + 6
  // Dirección:  M16b7    Dato: $DATO ALUMNO 16 BITS + 7
  // Dirección:  M16b8    Dato: $DATO ALUMNO 16 BITS + 8
  // Dirección:  M16b9    Dato: $DATO ALUMNO 16 BITS + 9
  // Dirección:  M16bA    Dato: $DATO ALUMNO 16 BITS + A
  // Dirección:  M16bB    Dato: $DATO ALUMNO 16 BITS + B
  // Dirección:  M16bC    Dato: $DATO ALUMNO 16 BITS + C
  // Dirección:  M16bD    Dato: $DATO ALUMNO 16 BITS + D
  // Dirección:  M16bE    Dato: $DATO ALUMNO 16 BITS + E
  // Dirección:  M16bF    Dato: $DATO ALUMNO 16 BITS + F

  mov( $DATO ALUMNO 16 BITS + 0, ax );
  mov( ax, M16b0 );
  mov( &M16b0, ebx );
  stdout.put( nl, "La direccion de M16b0 es: $", ebx );
  mov( M16b0, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 1, ax );
  mov( ax, M16b1 );
  mov( &M16b1, ebx );
  stdout.put( nl, "La direccion de M16b1 es: $", ebx );
  mov( M16b1, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 2, ax );
  mov( ax, M16b2 );
  mov( &M16b2, ebx );
  stdout.put( nl, "La direccion de M16b2 es: $", ebx );
  mov( M16b2, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 3, ax );
```

```
mov( ax, M16b3 );
mov( &M16b3, ebx );
stdout.put( nl, "La direccion de M16b3 es: $", ebx );
mov( M16b3, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 4, ax );
mov( ax, M16b4 );
mov( &M16b4, ebx );
stdout.put( nl, "La direccion de M16b4 es: $", ebx );
mov( M16b4, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 5, ax );
mov( ax, M16b5 );
mov( &M16b5, ebx );
stdout.put( nl, "La direccion de M16b5 es: $", ebx );
mov( M16b5, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 6, ax );
mov( ax, M16b6 );
mov( &M16b6, ebx );
stdout.put( nl, "La direccion de M16b6 es: $", ebx );
mov( M16b6, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 7, ax );
mov( ax, M16b7 );
mov( &M16b7, ebx );
stdout.put( nl, "La direccion de M16b7 es: $", ebx );
mov( M16b7, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 8, ax );
mov( ax, M16b8 );
mov( &M16b8, ebx );
stdout.put( nl, "La direccion de M16b8 es: $", ebx );
mov( M16b8, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 9, ax );
mov( ax, M16b9 );
mov( &M16b9, ebx );
stdout.put( nl, "La direccion de M16b9 es: $", ebx );
mov( M16b9, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + A, ax );
mov( ax, M16bA );
mov( &M16bA, ebx );
```

```
  stdout.put( nl, "La direccion de M16bA es: $", ebx );
  mov( M16bA, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + B, ax );
  mov( ax, M16bB );
  mov( &M16bB, ebx );
  stdout.put( nl, "La direccion de M16bB es: $", ebx );
  mov( M16bB, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + C, ax );
  mov( ax, M16bC );
  mov( &M16bC, ebx );
  stdout.put( nl, "La direccion de M16bC es: $", ebx );
  mov( M16bC, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + D, ax );
  mov( ax, M16bD );
  mov( &M16bD, ebx );
  stdout.put( nl, "La direccion de M16bD es: $", ebx );
  mov( M16bD, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + E, ax );
  mov( ax, M16bE );
  mov( &M16bE, ebx );
  stdout.put( nl, "La direccion de M16bE es: $", ebx );
  mov( M16bE, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + F, ax );
  mov( ax, M16bF );
  mov( &M16bF, ebx );
  stdout.put( nl, "La direccion de M16bF es: $", ebx );
  mov( M16bF, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  // Cargar la tabla de treinta y dos, bits

 stdout.put( nl, "TABLA DE 32 BITS", nl );

  // Dirección:  M32b0    Dato: $DATO ALUMNO 32 BITS + 0
  // Dirección:  M32b1    Dato: $DATO ALUMNO 32 BITS + 1
  // Dirección:  M32b2    Dato: $DATO ALUMNO 32 BITS + 2
  // Dirección:  M32b3    Dato: $DATO ALUMNO 32 BITS + 3
  // Dirección:  M32b4    Dato: $DATO ALUMNO 32 BITS + 4
  // Dirección:  M32b5    Dato: $DATO ALUMNO 32 BITS + 5
  // Dirección:  M32b6    Dato: $DATO ALUMNO 32 BITS + 6
  // Dirección:  M32b7    Dato: $DATO ALUMNO 32 BITS + 7
```

```
// Dirección:  M32b8    Dato: $DATO ALUMNO 32 BITS + 8
// Dirección:  M32b9    Dato: $DATO ALUMNO 32 BITS + 9
// Dirección:  M32bA    Dato: $DATO ALUMNO 32 BITS + A
// Dirección:  M32bB    Dato: $DATO ALUMNO 32 BITS + B
// Dirección:  M32bC    Dato: $DATO ALUMNO 32 BITS + C
// Dirección:  M32bD    Dato: $DATO ALUMNO 32 BITS + D
// Dirección:  M32bE    Dato: $DATO ALUMNO 32 BITS + E
// Dirección:  M32bF    Dato: $DATO ALUMNO 32 BITS + F

mov( $DATO ALUMNO 32 BITS + 0, eax );
mov( eax, M32b0 );
mov( &M32b0, ebx );
stdout.put( nl, "La direccion de M32b0 es: $", ebx );
mov( M32b0, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 1, eax );
mov( eax, M32b1 );
mov( &M32b1, ebx );
stdout.put( nl, "La direccion de M32b1 es: $", ebx );
mov( M32b1, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 2, eax );
mov( eax, M32b2 );
mov( &M32b2, ebx );
stdout.put( nl, "La direccion de M32b2 es: $", ebx );
mov( M32b2, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 3, eax );
mov( eax, M32b3 );
mov( &M32b3, ebx );
stdout.put( nl, "La direccion de M32b3 es: $", ebx );
mov( M32b3, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 4, eax );
mov(eax, M32b4 );
mov( &M32b4, ebx );
stdout.put( nl, "La direccion de M32b4 es: $", ebx );
mov( M32b4, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 5, eax );
mov( eax, M32b5 );
mov( &M32b5, ebx );
stdout.put( nl, "La direccion de M32b5 es: $", ebx );
mov( M32b5, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );
```

```
mov( $DATO ALUMNO 32 BITS + 6, eax );
mov( eax, M32b6 );
mov( &M32b6, ebx );
stdout.put( nl, "La direccion de M32b6 es: $", ebx );
mov( M32b6, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 7, eax );
mov( eax, M32b7 );
mov( &M32b7, ebx );
stdout.put( nl, "La direccion de M32b7 es: $", ebx );
mov( M32b7, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 8, eax );
mov( eax, M32b8 );
mov( &M32b8, ebx );
stdout.put( nl, "La direccion de M32b8 es: $", ebx );
mov( M32b8, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 9, eax );
mov( eax, M32b9 );
mov( &M32b9, ebx );
stdout.put( nl, "La direccion de M32b9 es: $", ebx );
mov( M32b9, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + A, eax );
mov( eax, M32bA );
mov( &M32bA, ebx );
stdout.put( nl, "La direccion de M32bA es: $", ebx );
mov( M32bA, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + B, eax );
mov( eax, M32bB );
mov( &M32bB, ebx );
stdout.put( nl, "La direccion de M32bB es: $", ebx );
mov( M32bB, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + C, eax );
mov( eax, M32bC );
mov( &M32bC, ebx );
stdout.put( nl, "La direccion de M32bC es: $", ebx );
mov( M32bC, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + D, eax );
mov( eax, M32bD );
```

```
  mov( &M32bD, ebx );
  stdout.put( nl, "La direccion de M32bD es: $", ebx );
  mov( M32bD, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

  mov( $DATO ALUMNO 32 BITS + E, eax );
  mov( eax, M32bE );
  mov( &M32bE, ebx );
  stdout.put( nl, "La direccion de M32bE es: $", ebx );
  mov( M32bE, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

  mov( $DATO ALUMNO 32 BITS + F, eax );
  mov( eax, M32bF );
  mov( &M32bF, ebx );
  stdout.put( nl, "La direccion de M32bF es: $", ebx );
  mov( M32bF, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

 // Direccionamiento BASE MÁS ÍNDICE

 stdout.put( nl, nl, "Direccionamiento BASE MAS INDICE", nl );

  // Direccionamiento BASE MÁS ÍNDICE a ocho bits

  mov( &M8b3, edi );
  mov( $00000003, ebx );
  mov( [ edi + ebx ], al );
  stdout.put( nl, "El dato en EDI es: M8b3     El dato en EBX es:$03    mov([edi+ebx], al)", nl );
  stdout.put( "Direccion final es: M8b6      El dato en AL es: $", al, nl );

  // Direccionamiento BASE MÁS ÍNDICE a diez y seis, bits

  // Múltiplos de dos las direcciones.

  stdout.put( nl, "Las direcciones son multiplos de dos", nl );

  mov( &M16b1, edi );
  mov( $00000008, ebx );
  mov( [ edi + ebx ], ax );
  stdout.put( "El dato en EDI es: M16b1     El dato en EBX es:$08    mov([edi+ebx], ax)", nl );
  stdout.put( "Direccion final es: M8b5      El dato en AX es: $", ax, nl );

  // Direccionamiento BASE MÁS ÍNDICE a treinta y dos, bits

  // Múltiplo de cuatro las direcciones.

  stdout.put( nl,"Las direcciones son multiplos de cuatro", nl );

  mov( &M32b4, edi );
  mov( $0000000C, ebx );
```

```
    mov( [ edi + ebx ], eax );
    stdout.put( "El dato en EDI es: M32b4     El dato en EBX es:$0C    mov([edi+ebx], eax)", nl );
    stdout.put( "Direccion final es: M32b7      El dato en EAX es: $", eax, nl );

 end DirecBaseMasInd;


#********************************************************************************
#  PARTE NO 16.
#
#  Direccionamiento Base Relativa más Indice
#********************************************************************************

program DirecBaseRelMasInd;

#include( "stdlib.hhf" )

static
          M8b0:      int8;
          M8b1:      int8;
          M8b2:      int8;
          M8b3:      int8;
          M8b4:      int8;
          M8b5:      int8;
          M8b6:      int8;
          M8b7:      int8;
          M8b8:      int8;
          M8b9:      int8;
          M8bA:     int8;
          M8bB:     int8;
          M8bC:     int8;
          M8bD:     int8;
          M8bE:     int8;
          M8bF:     int8;

          M16b0:      int16;
          M16b1:      int16;
          M16b2:      int16;
          M16b3:      int16;
          M16b4:      int16;
          M16b5:      int16;
          M16b6:      int16;
          M16b7:      int16;
          M16b8:      int16;
          M16b9:      int16;
          M16bA:     int16;
          M16bB:     int16;
          M16bC:     int16;
          M16bD:     int16;
          M16bE:     int16;
          M16bF:     int16;
```

```
            M32b0:      int32;
            M32b1:      int32;
            M32b2:      int32;
            M32b3:      int32;
            M32b4:      int32;
            M32b5:      int32;
            M32b6:      int32;
            M32b7:      int32;
            M32b8:      int32;
            M32b9:      int32;
            M32bA:     int32;
            M32bB:     int32;
            M32bC:     int32;
            M32bD:     int32;
            M32bE:     int32;
            M32bF:     int32;

begin DirecBaseRelMasInd;

   // PONER TABLAS DE DATOS

  stdout.put( nl, nl, "TABLAS DE DATOS EN MEMORIA", nl );

   // Cargar la tabla de ocho bits

  stdout.put( nl, "TABLA DE 8 BITS", nl );

   // Dirección:  M8b0    Dato: $DATO ALUMNO 8 BITS + 0
   // Dirección:  M8b1    Dato: $DATO ALUMNO 8 BITS + 1
   // Dirección:  M8b2    Dato: $DATO ALUMNO 8 BITS + 2
   // Dirección:  M8b3    Dato: $DATO ALUMNO 8 BITS + 3
   // Dirección:  M8b4    Dato: $DATO ALUMNO 8 BITS + 4
   // Dirección:  M8b5    Dato: $DATO ALUMNO 8 BITS + 5
   // Dirección:  M8b6    Dato: $DATO ALUMNO 8 BITS + 6
   // Dirección:  M8b7    Dato: $DATO ALUMNO 8 BITS + 7
   // Dirección:  M8b8    Dato: $DATO ALUMNO 8 BITS + 8
   // Dirección:  M8b9    Dato: $DATO ALUMNO 8 BITS + 9
   // Dirección:  M8bA    Dato: $DATO ALUMNO 8 BITS + A
   // Dirección:  M8bB    Dato: $DATO ALUMNO 8 BITS + B
   // Dirección:  M8bC    Dato: $DATO ALUMNO 8 BITS + C
   // Dirección:  M8bD    Dato: $DATO ALUMNO 8 BITS + D
   // Dirección:  M8bE    Dato: $DATO ALUMNO 8 BITS + E
   // Dirección:  M8bF    Dato: $DATO ALUMNO 8 BITS + F

   mov( $DATO ALUMNO 8 BITS + 0, al );
   mov( al, M8b0 );
   mov( &M8b0, ebx );
   stdout.put( nl, "La direccion de M8b0 es: $", ebx );
   mov( M8b0, al );
   stdout.put( ", El dato almacenado es: $", al, nl );
```

```
mov( $DATO ALUMNO 8 BITS + 1, al );
mov( al, M8b1 );
mov( &M8b1, ebx );
stdout.put( nl, "La direccion de M8b1 es: $", ebx );
mov( M8b1, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 2, al );
mov( al, M8b2 );
mov( &M8b2, ebx );
stdout.put( nl, "La direccion de M8b2 es: $", ebx );
mov( M8b2, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 3, al );
mov( al, M8b3 );
mov( &M8b3, ebx );
stdout.put( nl, "La direccion de M8b3 es: $", ebx );
mov( M8b3, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 4, al );
mov( al, M8b4 );
mov( &M8b4, ebx );
stdout.put( nl, "La direccion de M8b4 es: $", ebx );
mov( M8b4, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 5, al );
mov( al, M8b5 );
mov( &M8b5, ebx );
stdout.put( nl, "La direccion de M8b5 es: $", ebx );
mov( M8b5, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 6, al );
mov( al, M8b6 );
mov( &M8b6, ebx );
stdout.put( nl, "La direccion de M8b6 es: $", ebx );
mov( M8b6, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 7, al );
mov( al, M8b7 );
mov( &M8b7, ebx );
stdout.put( nl, "La direccion de M8b7 es: $", ebx );
mov( M8b7, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 8, al );
```

```
mov( al, M8b8 );
mov( &M8b8, ebx );
stdout.put( nl, "La direccion de M8b8 es: $", ebx );
mov( M8b8, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 9, al );
mov( al, M8b9 );
mov( &M8b9, ebx );
stdout.put( nl, "La direccion de M8b9 es: $", ebx );
mov( M8b9, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + A, al );
mov( al, M8bA );
mov( &M8bA, ebx );
stdout.put( nl, "La direccion de M8bA es: $", ebx );
mov( M8bA, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + B, al );
mov( al, M8bB );
mov( &M8bB, ebx );
stdout.put( nl, "La direccion de M8bB es: $", ebx );
mov( M8bB, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + C, al );
mov( al, M8bC );
mov( &M8bC, ebx );
stdout.put( nl, "La direccion de M8bC es: $", ebx );
mov( M8bC, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + D, al );
mov( al, M8bD );
mov( &M8bD, ebx );
stdout.put( nl, "La direccion de M8bD es: $", ebx );
mov( M8bD, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + E, al );
mov( al, M8bE );
mov( &M8bE, ebx );
stdout.put( nl, "La direccion de M8bE es: $", ebx );
mov( M8bE, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + F, al );
mov( al, M8bF );
mov( &M8bF, ebx );
```

```
  stdout.put( nl, "La direccion de M8bF es: $", ebx );
  mov( M8bF, al );
  stdout.put( ", El dato almacenado es: $", al, nl );

  // Cargar la tabla de diez y seis, bits

 stdout.put( nl, "TABLA DE 16 BITS", nl );

  // Dirección:  M16b0    Dato: $DATO ALUMNO 16 BITS + 0
  // Dirección:  M16b1    Dato: $DATO ALUMNO 16 BITS + 1
  // Dirección:  M16b2    Dato: $DATO ALUMNO 16 BITS + 2
  // Dirección:  M16b3    Dato: $DATO ALUMNO 16 BITS + 3
  // Dirección:  M16b4    Dato: $DATO ALUMNO 16 BITS + 4
  // Dirección:  M16b5    Dato: $DATO ALUMNO 16 BITS + 5
  // Dirección:  M16b6    Dato: $DATO ALUMNO 16 BITS + 6
  // Dirección:  M16b7    Dato: $DATO ALUMNO 16 BITS + 7
  // Dirección:  M16b8    Dato: $DATO ALUMNO 16 BITS + 8
  // Dirección:  M16b9    Dato: $DATO ALUMNO 16 BITS + 9
  // Dirección:  M16bA    Dato: $DATO ALUMNO 16 BITS + A
  // Dirección:  M16bB    Dato: $DATO ALUMNO 16 BITS + B
  // Dirección:  M16bC    Dato: $DATO ALUMNO 16 BITS + C
  // Dirección:  M16bD    Dato: $DATO ALUMNO 16 BITS + D
  // Dirección:  M16bE    Dato: $DATO ALUMNO 16 BITS + E
  // Dirección:  M16bF    Dato: $DATO ALUMNO 16 BITS + F

  mov( $DATO ALUMNO 16 BITS + 0, ax );
  mov( ax, M16b0 );
  mov( &M16b0, ebx );
  stdout.put( nl, "La direccion de M16b0 es: $", ebx );
  mov( M16b0, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 1, ax );
  mov( ax, M16b1 );
  mov( &M16b1, ebx );
  stdout.put( nl, "La direccion de M16b1 es: $", ebx );
  mov( M16b1, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 2, ax );
  mov( ax, M16b2 );
  mov( &M16b2, ebx );
  stdout.put( nl, "La direccion de M16b2 es: $", ebx );
  mov( M16b2, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 3, ax );
  mov( ax, M16b3 );
  mov( &M16b3, ebx );
  stdout.put( nl, "La direccion de M16b3 es: $", ebx );
  mov( M16b3, ax );
```

```
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 4, ax );
mov( ax, M16b4 );
mov( &M16b4, ebx );
stdout.put( nl, "La direccion de M16b4 es: $", ebx );
mov( M16b4, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 5, ax );
mov( ax, M16b5 );
mov( &M16b5, ebx );
stdout.put( nl, "La direccion de M16b5 es: $", ebx );
mov( M16b5, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 6, ax );
mov( ax, M16b6 );
mov( &M16b6, ebx );
stdout.put( nl, "La direccion de M16b6 es: $", ebx );
mov( M16b6, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 7, ax );
mov( ax, M16b7 );
mov( &M16b7, ebx );
stdout.put( nl, "La direccion de M16b7 es: $", ebx );
mov( M16b7, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 8, ax );
mov( ax, M16b8 );
mov( &M16b8, ebx );
stdout.put( nl, "La direccion de M16b8 es: $", ebx );
mov( M16b8, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 9, ax );
mov( ax, M16b9 );
mov( &M16b9, ebx );
stdout.put( nl, "La direccion de M16b9 es: $", ebx );
mov( M16b9, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + A, ax );
mov( ax, M16bA );
mov( &M16bA, ebx );
stdout.put( nl, "La direccion de M16bA es: $", ebx );
mov( M16bA, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );
```

```
mov( $DATO ALUMNO 16 BITS + B, ax );
mov( ax, M16bB );
mov( &M16bB, ebx );
stdout.put( nl, "La direccion de M16bB es: $", ebx );
mov( M16bB, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + C, ax );
mov( ax, M16bC );
mov( &M16bC, ebx );
stdout.put( nl, "La direccion de M16bC es: $", ebx );
mov( M16bC, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + D, ax );
mov( ax, M16bD );
mov( &M16bD, ebx );
stdout.put( nl, "La direccion de M16bD es: $", ebx );
mov( M16bD, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + E, ax );
mov( ax, M16bE );
mov( &M16bE, ebx );
stdout.put( nl, "La direccion de M16bE es: $", ebx );
mov( M16bE, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + F, ax );
mov( ax, M16bF );
mov( &M16bF, ebx );
stdout.put( nl, "La direccion de M16bF es: $", ebx );
mov( M16bF, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

// Cargar la tabla de treinta y dos, bits

stdout.put( nl, "TABLA DE 32 BITS", nl );

// Dirección:  M32b0    Dato: $DATO ALUMNO 32 BITS + 0
// Dirección:  M32b1    Dato: $DATO ALUMNO 32 BITS + 1
// Dirección:  M32b2    Dato: $DATO ALUMNO 32 BITS + 2
// Dirección:  M32b3    Dato: $DATO ALUMNO 32 BITS + 3
// Dirección:  M32b4    Dato: $DATO ALUMNO 32 BITS + 4
// Dirección:  M32b5    Dato: $DATO ALUMNO 32 BITS + 5
// Dirección:  M32b6    Dato: $DATO ALUMNO 32 BITS + 6
// Dirección:  M32b7    Dato: $DATO ALUMNO 32 BITS + 7
// Dirección:  M32b8    Dato: $DATO ALUMNO 32 BITS + 8
// Dirección:  M32b9    Dato: $DATO ALUMNO 32 BITS + 9
// Dirección:  M32bA    Dato: $DATO ALUMNO 32 BITS + A
// Dirección:  M32bB    Dato: $DATO ALUMNO 32 BITS + B
```

```
// Dirección:  M32bC    Dato: $DATO ALUMNO 32 BITS + C
// Dirección:  M32bD    Dato: $DATO ALUMNO 32 BITS + D
// Dirección:  M32bE    Dato: $DATO ALUMNO 32 BITS + E
// Dirección:  M32bF    Dato: $DATO ALUMNO 32 BITS + F

mov( $DATO ALUMNO 32 BITS + 0, eax );
mov( eax, M32b0 );
mov( &M32b0, ebx );
stdout.put( nl, "La direccion de M32b0 es: $", ebx );
mov( M32b0, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 1, eax );
mov( eax, M32b1 );
mov( &M32b1, ebx );
stdout.put( nl, "La direccion de M32b1 es: $", ebx );
mov( M32b1, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 2, eax );
mov( eax, M32b2 );
mov( &M32b2, ebx );
stdout.put( nl, "La direccion de M32b2 es: $", ebx );
mov( M32b2, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 3, eax );
mov( eax, M32b3 );
mov( &M32b3, ebx );
stdout.put( nl, "La direccion de M32b3 es: $", ebx );
mov( M32b3, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 4, eax );
mov(eax, M32b4 );
mov( &M32b4, ebx );
stdout.put( nl, "La direccion de M32b4 es: $", ebx );
mov( M32b4, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 5, eax );
mov( eax, M32b5 );
mov( &M32b5, ebx );
stdout.put( nl, "La direccion de M32b5 es: $", ebx );
mov( M32b5, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 6, eax );
mov( eax, M32b6 );
mov( &M32b6, ebx );
stdout.put( nl, "La direccion de M32b6 es: $", ebx );
```

```
mov( M32b6, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 7, eax );
mov( eax, M32b7 );
mov( &M32b7, ebx );
stdout.put( nl, "La direccion de M32b7 es: $", ebx );
mov( M32b7, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 8, eax );
mov( eax, M32b8 );
mov( &M32b8, ebx );
stdout.put( nl, "La direccion de M32b8 es: $", ebx );
mov( M32b8, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 9, eax );
mov( eax, M32b9 );
mov( &M32b9, ebx );
stdout.put( nl, "La direccion de M32b9 es: $", ebx );
mov( M32b9, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + A, eax );
mov( eax, M32bA );
mov( &M32bA, ebx );
stdout.put( nl, "La direccion de M32bA es: $", ebx );
mov( M32bA, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + B, eax );
mov( eax, M32bB );
mov( &M32bB, ebx );
stdout.put( nl, "La direccion de M32bB es: $", ebx );
mov( M32bB, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + C, eax );
mov( eax, M32bC );
mov( &M32bC, ebx );
stdout.put( nl, "La direccion de M32bC es: $", ebx );
mov( M32bC, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + D, eax );
mov( eax, M32bD );
mov( &M32bD, ebx );
stdout.put( nl, "La direccion de M32bD es: $", ebx );
mov( M32bD, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );
```

```
  mov( $DATO ALUMNO 32 BITS + E, eax );
  mov( eax, M32bE );
  mov( &M32bE, ebx );
  stdout.put( nl, "La direccion de M32bE es: $", ebx );
  mov( M32bE, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

  mov( $DATO ALUMNO 32 BITS + F, eax );
  mov( eax, M32bF );
  mov( &M32bF, ebx );
  stdout.put( nl, "La direccion de M32bF es: $", ebx );
  mov( M32bF, eax );
  stdout.put( ", El dato almacenado es: $", eax, nl );

 // Direccionamiento BASE RELATIVA MÁS ÍNDICE

  stdout.put( nl, nl, "Direccionamiento BASE RELATIVA MAS INDICE", nl );

  // Direccionamiento BASE RELATIVA MÁS ÍNDICE a ocho bits

  mov( &M8b4, edi );
  mov( $00000005, ebx );
  mov( [ edi + ebx + $00000002 ], al );
  stdout.put( nl, "El dato en EDI es: M8b4  El dato en EBX es:$05 mov([edi+ebx+$02], al)", nl );
  stdout.put( "Direccion final es: M8bB      El dato en AL es: $", al, nl );

  // Direccionamiento BASE RELATIVA MÁS ÍNDICE a diez y seis, bits

  // Múltiplos de dos las direcciones.

  stdout.put( nl, "Las direcciones son multiplos de dos", nl );

  mov( &M16b3, edi );
  mov( $00000004, ebx );
  mov( [ edi + ebx + $00000006 ], ax );
  stdout.put( "El dato en EDI es: M16b3   El dato en EBX es:$04   mov([edi+ebx+$06], ax)", nl );
  stdout.put( "Direccion final es: M16b8      El dato en AX es: $", ax, nl );

  // Direccionamiento BASE RELATIVA MÁS ÍNDICE a treinta y dos, bits

  // Múltiplo de cuatro las direcciones.

  stdout.put( nl,"Las direcciones son multiplos de cuatro", nl );

  mov( &M32b1, edi );
  mov( $00000008, ebx );
  mov( [ edi + ebx + $0000000C ], eax );
  stdout.put( "El dato en EDI es: M32b1  El dato en EBX es:$08 mov([edi+ebx+$0C], eax)", nl );
  stdout.put( "Direccion final es: M16b6      El dato en EAX es: $", eax, nl );
```

```
end DirecBaseRelMasInd;


#********************************************************************************
#  PARTE NO 17.
#
#  Direccionamiento Indice Escalado
#********************************************************************************

program DirecIndEsc;

#include( "stdlib.hhf" )

static
        M8b0:     int8;
        M8b1:     int8;
        M8b2:     int8;
        M8b3:     int8;
        M8b4:     int8;
        M8b5:     int8;
        M8b6:     int8;
        M8b7:     int8;
        M8b8:     int8;
        M8b9:     int8;
        M8bA:    int8;
        M8bB:    int8;
        M8bC:    int8;
        M8bD:    int8;
        M8bE:    int8;
        M8bF:    int8;

        M16b0:     int16;
        M16b1:     int16;
        M16b2:     int16;
        M16b3:     int16;
        M16b4:     int16;
        M16b5:     int16;
        M16b6:     int16;
        M16b7:     int16;
        M16b8:     int16;
        M16b9:     int16;
        M16bA:    int16;
        M16bB:    int16;
        M16bC:    int16;
        M16bD:    int16;
        M16bE:    int16;
        M16bF:    int16;

        M32b0:     int32;
        M32b1:     int32;
        M32b2:     int32;
```

```
            M32b3:      int32;
            M32b4:      int32;
            M32b5:      int32;
            M32b6:      int32;
            M32b7:      int32;
            M32b8:      int32;
            M32b9:      int32;
            M32bA:     int32;
            M32bB:     int32;
            M32bC:     int32;
            M32bD:     int32;
            M32bE:     int32;
            M32bF:     int32;

begin DirecIndEsc;

    // PONER TABLAS DE DATOS

   stdout.put( nl, nl, "TABLAS DE DATOS EN MEMORIA", nl );

    // Cargar la tabla de ocho bits

   stdout.put( nl, "TABLA DE 8 BITS", nl );

    // Dirección:  M8b0    Dato: $DATO ALUMNO 8 BITS + 0
    // Dirección:  M8b1    Dato: $DATO ALUMNO 8 BITS + 1
    // Dirección:  M8b2    Dato: $DATO ALUMNO 8 BITS + 2
    // Dirección:  M8b3    Dato: $DATO ALUMNO 8 BITS + 3
    // Dirección:  M8b4    Dato: $DATO ALUMNO 8 BITS + 4
    // Dirección:  M8b5    Dato: $DATO ALUMNO 8 BITS + 5
    // Dirección:  M8b6    Dato: $DATO ALUMNO 8 BITS + 6
    // Dirección:  M8b7    Dato: $DATO ALUMNO 8 BITS + 7
    // Dirección:  M8b8    Dato: $DATO ALUMNO 8 BITS + 8
    // Dirección:  M8b9    Dato: $DATO ALUMNO 8 BITS + 9
    // Dirección:  M8bA    Dato: $DATO ALUMNO 8 BITS + A
    // Dirección:  M8bB    Dato: $DATO ALUMNO 8 BITS + B
    // Dirección:  M8bC    Dato: $DATO ALUMNO 8 BITS + C
    // Dirección:  M8bD    Dato: $DATO ALUMNO 8 BITS + D
    // Dirección:  M8bE    Dato: $DATO ALUMNO 8 BITS + E
    // Dirección:  M8bF    Dato: $DATO ALUMNO 8 BITS + F

    mov( $DATO ALUMNO 8 BITS + 0, al );
    mov( al, M8b0 );
    mov( &M8b0, ebx );
    stdout.put( nl, "La direccion de M8b0 es: $", ebx );
    mov( M8b0, al );
    stdout.put( ", El dato almacenado es: $", al, nl );

    mov( $DATO ALUMNO 8 BITS + 1, al );
    mov( al, M8b1 );
    mov( &M8b1, ebx );
```

```
stdout.put( nl, "La direccion de M8b1 es: $", ebx );
mov( M8b1, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 2, al );
mov( al, M8b2 );
mov( &M8b2, ebx );
stdout.put( nl, "La direccion de M8b2 es: $", ebx );
mov( M8b2, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 3, al );
mov( al, M8b3 );
mov( &M8b3, ebx );
stdout.put( nl, "La direccion de M8b3 es: $", ebx );
mov( M8b3, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 4, al );
mov( al, M8b4 );
mov( &M8b4, ebx );
stdout.put( nl, "La direccion de M8b4 es: $", ebx );
mov( M8b4, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 5, al );
mov( al, M8b5 );
mov( &M8b5, ebx );
stdout.put( nl, "La direccion de M8b5 es: $", ebx );
mov( M8b5, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 6, al );
mov( al, M8b6 );
mov( &M8b6, ebx );
stdout.put( nl, "La direccion de M8b6 es: $", ebx );
mov( M8b6, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 7, al );
mov( al, M8b7 );
mov( &M8b7, ebx );
stdout.put( nl, "La direccion de M8b7 es: $", ebx );
mov( M8b7, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 8, al );
mov( al, M8b8 );
mov( &M8b8, ebx );
stdout.put( nl, "La direccion de M8b8 es: $", ebx );
mov( M8b8, al );
```

```
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + 9, al );
mov( al, M8b9 );
mov( &M8b9, ebx );
stdout.put( nl, "La direccion de M8b9 es: $", ebx );
mov( M8b9, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + A, al );
mov( al, M8bA );
mov( &M8bA, ebx );
stdout.put( nl, "La direccion de M8bA es: $", ebx );
mov( M8bA, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + B, al );
mov( al, M8bB );
mov( &M8bB, ebx );
stdout.put( nl, "La direccion de M8bB es: $", ebx );
mov( M8bB, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + C, al );
mov( al, M8bC );
mov( &M8bC, ebx );
stdout.put( nl, "La direccion de M8bC es: $", ebx );
mov( M8bC, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + D, al );
mov( al, M8bD );
mov( &M8bD, ebx );
stdout.put( nl, "La direccion de M8bD es: $", ebx );
mov( M8bD, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + E, al );
mov( al, M8bE );
mov( &M8bE, ebx );
stdout.put( nl, "La direccion de M8bE es: $", ebx );
mov( M8bE, al );
stdout.put( ", El dato almacenado es: $", al, nl );

mov( $DATO ALUMNO 8 BITS + F, al );
mov( al, M8bF );
mov( &M8bF, ebx );
stdout.put( nl, "La direccion de M8bF es: $", ebx );
mov( M8bF, al );
stdout.put( ", El dato almacenado es: $", al, nl );
```

```
  // Cargar la tabla de diez y seis, bits

 stdout.put( nl, "TABLA DE 16 BITS", nl );

  // Dirección:  M16b0    Dato: $DATO ALUMNO 16 BITS + 0
  // Dirección:  M16b1    Dato: $DATO ALUMNO 16 BITS + 1
  // Dirección:  M16b2    Dato: $DATO ALUMNO 16 BITS + 2
  // Dirección:  M16b3    Dato: $DATO ALUMNO 16 BITS + 3
  // Dirección:  M16b4    Dato: $DATO ALUMNO 16 BITS + 4
  // Dirección:  M16b5    Dato: $DATO ALUMNO 16 BITS + 5
  // Dirección:  M16b6    Dato: $DATO ALUMNO 16 BITS + 6
  // Dirección:  M16b7    Dato: $DATO ALUMNO 16 BITS + 7
  // Dirección:  M16b8    Dato: $DATO ALUMNO 16 BITS + 8
  // Dirección:  M16b9    Dato: $DATO ALUMNO 16 BITS + 9
  // Dirección:  M16bA    Dato: $DATO ALUMNO 16 BITS + A
  // Dirección:  M16bB    Dato: $DATO ALUMNO 16 BITS + B
  // Dirección:  M16bC    Dato: $DATO ALUMNO 16 BITS + C
  // Dirección:  M16bD    Dato: $DATO ALUMNO 16 BITS + D
  // Dirección:  M16bE    Dato: $DATO ALUMNO 16 BITS + E
  // Dirección:  M16bF    Dato: $DATO ALUMNO 16 BITS + F

  mov( $DATO ALUMNO 16 BITS + 0, ax );
  mov( ax, M16b0 );
  mov( &M16b0, ebx );
  stdout.put( nl, "La direccion de M16b0 es: $", ebx );
  mov( M16b0, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 1, ax );
  mov( ax, M16b1 );
  mov( &M16b1, ebx );
  stdout.put( nl, "La direccion de M16b1 es: $", ebx );
  mov( M16b1, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 2, ax );
  mov( ax, M16b2 );
  mov( &M16b2, ebx );
  stdout.put( nl, "La direccion de M16b2 es: $", ebx );
  mov( M16b2, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 3, ax );
  mov( ax, M16b3 );
  mov( &M16b3, ebx );
  stdout.put( nl, "La direccion de M16b3 es: $", ebx );
  mov( M16b3, ax );
  stdout.put( ", El dato almacenado es: $", ax, nl );

  mov( $DATO ALUMNO 16 BITS + 4, ax );
  mov( ax, M16b4 );
```

```
mov( &M16b4, ebx );
stdout.put( nl, "La direccion de M16b4 es: $", ebx );
mov( M16b4, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 5, ax );
mov( ax, M16b5 );
mov( &M16b5, ebx );
stdout.put( nl, "La direccion de M16b5 es: $", ebx );
mov( M16b5, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 6, ax );
mov( ax, M16b6 );
mov( &M16b6, ebx );
stdout.put( nl, "La direccion de M16b6 es: $", ebx );
mov( M16b6, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 7, ax );
mov( ax, M16b7 );
mov( &M16b7, ebx );
stdout.put( nl, "La direccion de M16b7 es: $", ebx );
mov( M16b7, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 8, ax );
mov( ax, M16b8 );
mov( &M16b8, ebx );
stdout.put( nl, "La direccion de M16b8 es: $", ebx );
mov( M16b8, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + 9, ax );
mov( ax, M16b9 );
mov( &M16b9, ebx );
stdout.put( nl, "La direccion de M16b9 es: $", ebx );
mov( M16b9, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + A, ax );
mov( ax, M16bA );
mov( &M16bA, ebx );
stdout.put( nl, "La direccion de M16bA es: $", ebx );
mov( M16bA, ax );
stdout.put( ", El dato almacenado es: $", ax, nl );

mov( $DATO ALUMNO 16 BITS + B, ax );
mov( ax, M16bB );
mov( &M16bB, ebx );
stdout.put( nl, "La direccion de M16bB es: $", ebx );
```

```
    mov( M16bB, ax );
    stdout.put( ", El dato almacenado es: $", ax, nl );

    mov( $DATO ALUMNO 16 BITS + C, ax );
    mov( ax, M16bC );
    mov( &M16bC, ebx );
    stdout.put( nl, "La direccion de M16bC es: $", ebx );
    mov( M16bC, ax );
    stdout.put( ", El dato almacenado es: $", ax, nl );

    mov( $DATO ALUMNO 16 BITS + D, ax );
    mov( ax, M16bD );
    mov( &M16bD, ebx );
    stdout.put( nl, "La direccion de M16bD es: $", ebx );
    mov( M16bD, ax );
    stdout.put( ", El dato almacenado es: $", ax, nl );

    mov( $DATO ALUMNO 16 BITS + E, ax );
    mov( ax, M16bE );
    mov( &M16bE, ebx );
    stdout.put( nl, "La direccion de M16bE es: $", ebx );
    mov( M16bE, ax );
    stdout.put( ", El dato almacenado es: $", ax, nl );

    mov( $DATO ALUMNO 16 BITS + F, ax );
    mov( ax, M16bF );
    mov( &M16bF, ebx );
    stdout.put( nl, "La direccion de M16bF es: $", ebx );
    mov( M16bF, ax );
    stdout.put( ", El dato almacenado es: $", ax, nl );

    // Cargar la tabla de treinta y dos, bits

  stdout.put( nl, "TABLA DE 32 BITS", nl );

    // Dirección:  M32b0    Dato: $DATO ALUMNO 32 BITS + 0
    // Dirección:  M32b1    Dato: $DATO ALUMNO 32 BITS + 1
    // Dirección:  M32b2    Dato: $DATO ALUMNO 32 BITS + 2
    // Dirección:  M32b3    Dato: $DATO ALUMNO 32 BITS + 3
    // Dirección:  M32b4    Dato: $DATO ALUMNO 32 BITS + 4
    // Dirección:  M32b5    Dato: $DATO ALUMNO 32 BITS + 5
    // Dirección:  M32b6    Dato: $DATO ALUMNO 32 BITS + 6
    // Dirección:  M32b7    Dato: $DATO ALUMNO 32 BITS + 7
    // Dirección:  M32b8    Dato: $DATO ALUMNO 32 BITS + 8
    // Dirección:  M32b9    Dato: $DATO ALUMNO 32 BITS + 9
    // Dirección:  M32bA    Dato: $DATO ALUMNO 32 BITS + A
    // Dirección:  M32bB    Dato: $DATO ALUMNO 32 BITS + B
    // Dirección:  M32bC    Dato: $DATO ALUMNO 32 BITS + C
    // Dirección:  M32bD    Dato: $DATO ALUMNO 32 BITS + D
    // Dirección:  M32bE    Dato: $DATO ALUMNO 32 BITS + E
    // Dirección:  M32bF    Dato: $DATO ALUMNO 32 BITS + F
```

```
mov( $DATO ALUMNO 32 BITS + 0, eax );
mov( eax, M32b0 );
mov( &M32b0, ebx );
stdout.put( nl, "La direccion de M32b0 es: $", ebx );
mov( M32b0, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 1, eax );
mov( eax, M32b1 );
mov( &M32b1, ebx );
stdout.put( nl, "La direccion de M32b1 es: $", ebx );
mov( M32b1, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 2, eax );
mov( eax, M32b2 );
mov( &M32b2, ebx );
stdout.put( nl, "La direccion de M32b2 es: $", ebx );
mov( M32b2, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 3, eax );
mov( eax, M32b3 );
mov( &M32b3, ebx );
stdout.put( nl, "La direccion de M32b3 es: $", ebx );
mov( M32b3, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 4, eax );
mov(eax, M32b4 );
mov( &M32b4, ebx );
stdout.put( nl, "La direccion de M32b4 es: $", ebx );
mov( M32b4, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 5, eax );
mov( eax, M32b5 );
mov( &M32b5, ebx );
stdout.put( nl, "La direccion de M32b5 es: $", ebx );
mov( M32b5, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 6, eax );
mov( eax, M32b6 );
mov( &M32b6, ebx );
stdout.put( nl, "La direccion de M32b6 es: $", ebx );
mov( M32b6, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 7, eax );
```

```
mov( eax, M32b7 );
mov( &M32b7, ebx );
stdout.put( nl, "La direccion de M32b7 es: $", ebx );
mov( M32b7, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 8, eax );
mov( eax, M32b8 );
mov( &M32b8, ebx );
stdout.put( nl, "La direccion de M32b8 es: $", ebx );
mov( M32b8, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + 9, eax );
mov( eax, M32b9 );
mov( &M32b9, ebx );
stdout.put( nl, "La direccion de M32b9 es: $", ebx );
mov( M32b9, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + A, eax );
mov( eax, M32bA );
mov( &M32bA, ebx );
stdout.put( nl, "La direccion de M32bA es: $", ebx );
mov( M32bA, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + B, eax );
mov( eax, M32bB );
mov( &M32bB, ebx );
stdout.put( nl, "La direccion de M32bB es: $", ebx );
mov( M32bB, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + C, eax );
mov( eax, M32bC );
mov( &M32bC, ebx );
stdout.put( nl, "La direccion de M32bC es: $", ebx );
mov( M32bC, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + D, eax );
mov( eax, M32bD );
mov( &M32bD, ebx );
stdout.put( nl, "La direccion de M32bD es: $", ebx );
mov( M32bD, eax );
stdout.put( ", El dato almacenado es: $", eax, nl );

mov( $DATO ALUMNO 32 BITS + E, eax );
mov( eax, M32bE );
mov( &M32bE, ebx );
```

```
    stdout.put( nl, "La direccion de M32bE es: $", ebx );
    mov( M32bE, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

    mov( $DATO ALUMNO 32 BITS + F, eax );
    mov( eax, M32bF );
    mov( &M32bF, ebx );
    stdout.put( nl, "La direccion de M32bF es: $", ebx );
    mov( M32bF, eax );
    stdout.put( ", El dato almacenado es: $", eax, nl );

  // Direccionamiento ÍNDICE ESCALADO

   stdout.put( nl, nl, "Direccionamiento INDICE ESCALADO", nl );

    // Direccionamiento ÍNDICE ESCALADO a ocho bits

    mov( &M8b3, ebx );
    mov( $00000002, edi );
    mov ( [ ebx + edi*2 + $00000004 ], al);
    stdout.put( nl, "El dato en EBX es: M8b3 El dato en EDI es:$02 mov([ebx+edi*2+$04], al)", nl );
    stdout.put( "Direccion final es: M8bB      El dato en AL es: $", al, nl );

    // Direccionamiento ÍNDICE ESCALADO a diez y seis, bits

    // Múltiplos de dos las direcciones.

    stdout.put( nl, "Las direcciones son multiplos de dos", nl );

    mov( &M16b2, ebx );
    mov( $00000003, edi );
    mov ( [ ebx + edi*2 + $00000006 ], ax);
    stdout.put( "El dato en EBX es: M16b2 El dato en EDI es:$03 mov([ebx+edi*2+$06], ax)", nl );
    stdout.put( "Direccion final es: M16b8      El dato en AX es: $", ax, nl );

    // Direccionamiento ÍNDICE ESCALADO a treinta y dos, bits

    // Múltiplo de cuatro las direcciones.

    stdout.put( nl,"Las direcciones son multiplos de cuatro", nl );

    mov( &M32b3, ebx );
    mov( $00000002, edi );
    mov ( [ ebx + edi*4 + $00000008 ], ax);
    stdout.put( "El dato en EBX es: M32b3 El dato en EDI es:$02 mov([ebx+edi*4+$08], eax)", nl );
    stdout.put( "Direccion final es: M32b7      El dato en EAX es: $", eax, nl );

end DirecIndEsc;


#********************************************************************************
```

```
#  PARTE NO 18.
#
#  Entra dato en hexadecimal, se incrementa y se decrementa
#*************************************************************************

program IncDec;
#include( "stdlib.hhf" )

begin IncDec;

// Incermento y Decremento de 8 bits

    stdout.put( nl, "Incermento y Decremento de 8 bits", nl );
    stdout.put( nl, "Meter dato hexadecimal entre 00 y FF: " );

    stdin.get( al );

    stdout.put( nl, "Valor original es: $", al, nl );

    mov( al, bl );
    inc( al );
    dec( bl );

    stdout.put( nl, "Valor incrementado es: $", al, nl );
    stdout.put( nl, "Valor decrementado es: $", bl, nl );

// Incermento y Decremento de 16 bits

    stdout.put( nl, nl, "Incermento y Decremento de 16 bits", nl );
    stdout.put( nl, "Meter dato hexadecimal entre 0000 y FFFF: " );

    stdin.get( ax );

    stdout.put( nl, "Valor original es: $", ax, nl );

    mov( ax, bx );
    inc( ax );
    dec( bx );

    stdout.put( nl, "Valor incrementado es: $", ax, nl );
    stdout.put( nl, "Valor decrementado es: $", bx, nl );

// Incermento y Decremento de 32 bits

    stdout.put( nl, nl, "Incermento y Decremento de 32 bits", nl );
    stdout.put( nl, "Meter dato hexadecimal entre 00000000 y FFFFFFFF: " );

    stdin.get( eax );

    stdout.put( nl, "Valor original es: $", eax, nl );
```

```
    mov( eax, ebx );
    inc( eax );
    dec( ebx );

    stdout.put( nl, "Valor incrementado es: $", eax, nl );
    stdout.put( nl, "Valor decrementado es: $", ebx, nl );


end IncDec;


#************************************************************************
#  PARTE NO 19.
#
#  Programa sin subrutina
#************************************************************************

program SinSubrutina;

#include( "stdlib.hhf" )

// Programa principal

    static
    VariableGlobal:      int8;

begin SinSubrutina;

    stdout.put( nl, "Programa sin subrutina", nl );
    mov( $DATO ALUMNO 8 BITS, VariableGlobal );
    mov( VariableGlobal, al );
    stdout.put( nl, "VariableGlobal=$", al, nl );

end SinSubrutina;


#************************************************************************
#  PARTE NO 20.
#
#  Programa con subrutina
#************************************************************************

program ConSubrutina;

#include( "stdlib.hhf" )

// Subrutina "Sub1"

        procedure Sub1;

        static
```

```
        VariableLocal:      int8;

        begin Sub1;

        stdout.put( nl,  nl, "Programa con subrutina", nl );
        mov( $DATO ALUMNO 8 BITS+1, VariableLocal );
        mov( VariableLocal, al );
        stdout.put( nl, "VariableLocal=$", al, nl );

        end Sub1;

// Programa principal

    static
    VariableGlobal:      int8;

begin ConSubrutina;

    stdout.put( nl, "Programa sin subrutina", nl );
    mov( $DATO ALUMNO 8 BITS+0, VariableGlobal );
    mov( VariableGlobal, al );
    stdout.put( nl, "VariableGlobal=$", al, nl );
    call Sub1;

end ConSubrutina;
```