TRADUCTORES DE LENGUAJES

19 de abril de 2018

Observaciones: 1. Las calificaciones del primer parcial se publicarán hacia el 25 de abril y la revisión será hacia el 27 de abril. Las fechas exactas se avisarán en la web de la asignatura.
2. La duración de este examen es de 1 hora.

Dado el siguiente fragmento de la gramática de un lenguaje de programación:

$E \rightarrow E_1 \# E_2 \mid id$
$S \rightarrow repeat\ S_1\ while\ E \mid S_1 ; S_2 \mid exit \mid call\ id\ (\ P\ )$
$P \rightarrow E \mid E , P_1$

Teniendo en cuenta que:

- Los identificadores son lógicos.
- La entrada recibida es léxica, sintáctica y semánticamente correcta.
- El operador lógico # devuelve falso si el primer operando es falso y el segundo es cierto; en caso contrario devuelve cierto. Hay que tener en cuenta que este operador no existe en el lenguaje intermedio.
- La sentencia repeat funciona de la siguiente manera. Se ejecuta el cuerpo del repeat. Se evalúa la expresión E. Si es cierta, se vuelve a ejecutar el cuerpo del repeat; si es falsa, se sale del repeat.
- La sentencia exit detiene la iteración en curso del repeat y temina el repeat.
- La sentencia call llama a la función id que recibe parámetros y no devuelve nada.

Explicando brevemente las funciones y atributos utilizados, se pide diseñar el Generador de Código Intermedio mediante un Esquema de Traducción, asumiendo una representación numérica de los valores lógicos.

TRADUCTORES DE LENGUAJES

4 de junio de 2018

Observaciones: 1. Las calificaciones se publicarán hacia el 25 de junio y la revisión será hacia el 27 de junio. Las fechas exactas se avisarán en la web de la asignatura.
2. La duración de este examen es de 1 hora.

1. Dado el siguiente fragmento de la gramática de un lenguaje de programación:

$E \rightarrow E_1$ % E_2 | id
$S \rightarrow$ do S_1 until E | S_1 ; S_2 | continue | call id (P)
$P \rightarrow E$ | E , P_1

Teniendo en cuenta que:

- Los identificadores son lógicos.
- La entrada recibida es léxica, sintáctica y semánticamente correcta.
- El operador lógico % devuelve cierto si el primer operando es falso y el segundo es cierto; en caso contrario devuelve falso. Hay que tener en cuenta que este operador no existe en el lenguaje intermedio.
- La sentencia do-until funciona de la siguiente manera. Se ejecuta el cuerpo de la sentencia. Se evalúa la expresión E. Si es falsa, se vuelve a ejecutar el cuerpo del do-until; si es cierta, se sale del do-until.
- La sentencia continue detiene la iteración en curso del do-until y vuelve a evaluar la expresión E (esta sentencia no termina la ejecución del do-until).
- La sentencia call llama a la función id que recibe parámetros y no devuelve nada.

Explicando brevemente las funciones y atributos utilizados, se pide diseñar el **Generador de Código Intermedio** mediante una **Definición Dirigida por Sintaxis**, asumiendo una representación numérica de los valores lógicos.

TRADUCTORES DE LENGUAJES

4 de junio de 2018

Observaciones: 1. Las calificaciones se publicarán hacia el 25 de junio y la revisión será hacia el 27 de junio. Las fechas exactas se avisarán en la web de la asignatura.
2. La duración de este examen es de 1 hora.

2. Dado el siguiente programa fuente correcto:

```
Programa Junio18
  Global a: entero
         b: real
  Procedimiento INSTRUCTOR (Ref x: entero, y: real, Ref z: real)
                                          // x, z por referencia, y por valor
    Procedimiento DECISOR (z: real)       // z por valor
        Local f: real
      Begin DECISOR
        f:= z / 2.0
        z:= z * y      //    ← c. Traducir a CI y CO
      End DECISOR
    Begin INSTRUCTOR
      If (y - z < 1)Then DECISOR (a)
                    Else a:= a + 2
    End INSTRUCTOR

  Procedimiento SELECTOR (Ref x: entero; y: real) // x por referencia, y por valor
    Begin SELECTOR
      En caso que y:
        >  5.0 ⇒ INSTRUCTOR (x, y, 3.0)
        <= 1.0 ⇒ INSTRUCTOR (x, 3.0, y)
        En otro caso ⇒ INSTRUCTOR (2, 3.0, 4.0)
      x:= x * 2
    End SELECTOR

Begin Junio18
  a:= 2
  b:= 6.0
  Mientras (a <= 8)
    SELECTOR (a, b)
    b:= b - 1
  End Mientras
  Print (a, b)  //    b. Resultado
End Junio18
```

Se pide:

a. Diseñar el **Registro de Activación** para este lenguaje, teniendo en cuenta (además de lo que se deduce del programa) que los enteros ocupan 2 bytes, los reales 4 y las direcciones 6. El lenguaje tiene conversión automática de tipos.

b. Realizar la **traza de ejecución** del programa mostrando la pila de Registros de Activación. Indicar el resultado de la última sentencia del programa: "`Print (a, b)`".

c. Traducir a **Código Intermedio** y a **Código Objeto** la instrucción marcada con ←, explicando muy brevemente el objetivo de cada instrucción de Código Ensamblador.

TRADUCTORES DE LENGUAJES

Examen Final, 27 de junio de 2018

Observaciones:	1. Fecha estimada de publicación de las calificaciones: 2 de julio. 2. Fecha estimada de la revisión: 4 de julio. 3. La duración de este examen será de 2 horas. 4. Cada ejercicio tiene la misma puntuación. 5. Cada ejercicio deberá entregarse en hojas separadas.

Un fragmento de un lenguaje tiene la siguiente sintaxis:

S → id:= E | return E | id (E) | C
C → case id of L end
L → cte_ent: S; L | D
D → default: S | λ
E → E * E | E - E | id | cte_ent

El lenguaje tiene las siguientes características:

- Las variables (id) tienen que estar declaradas previamente y son siempre enteras.
- Todas las variables temporales se almacenan en la tabla de símbolos.
- Todos los campos del registro de activación se almacenan en la pila.
- Todas las funciones del lenguaje tienen un parámetro entero (que puede pasarse por valor o por referencia según su declaración) y devuelven un entero.
- El compilador no realiza ningún tipo de optimización.
- La sentencia case es una sentencia de selección múltiple que permite ejecutar la sentencia S asociada a un determinado valor (cte_ent) o, si existe, la sentencia S por defecto (default), en función del valor de una variable (id).

1. Construir una Definición Dirigida por la Sintaxis para **generar código intermedio** de tres direcciones para la gramática dada, indicando todos los accesos a la Tabla de Símbolos.

2. Representar la **pila** de **registros de activación** completa, detallándola al máximo, durante la ejecución del siguiente fragmento de programa e indicar el código intermedio y final de la instrucción marcada con **(*)**:

```
Procedure prueba;
   x, y: Integer;
   Function resultado(Val y: Integer): Integer;
   {  Case y of
         1: x:= x * y – x;
         2: x:= x * y;
         3: x:= x – y;
         default: Return x – y – y;
      End;
      Return x;  (*)
   }
   Function calcula(Ref y: Integer): Integer;
      x, z: Integer;
   {  z:= 3;
      x:= y – z;
      y:= y – 1;
      x:= resultado (x);      // x se pasa por valor
      Return x;
   }
{  y:= 5;
   x:= y;
   y:= calcula (x);    // x se pasa por referencia
   Print y;
}
```

TRADUCTORES DE LENGUAJES

3 de abril de 2019

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 24 de abril. 2. Fecha **estimada** de la revisión: 26 de abril. 3. En la web se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 4. La duración de este examen será de 1 hora.

De un lenguaje de programación se ha extraído el siguiente fragmento de gramática:

```
S → For id := E To E Step E Do S Next ; | id := E ; | S S
E → id | núm | E + E | id ( L )
L → E | E , L
```

Se pide diseñar para este lenguaje el **Generador de Código Intermedio** de tres direcciones mediante un **Esquema de Traducción**, teniendo en cuenta que:

- El lenguaje solo tiene variables y datos de tipo entero y se asume que el Análisis Semántico está realizado y es correcto
- Las variables y funciones exigen haber sido declaradas previamente a su uso
- La expresión puede ser una variable, una constante entera, una suma entera o una llamada a una función (siendo `L` los parámetros actuales de la función)
- El funcionamiento del bucle `For` es como sigue:
  1. Se evalúan las tres expresiones
  2. Se inicializa la variable entera índice (`id`) con el valor de la primera expresión
  3. Si la variable índice es mayor al valor obtenido al evaluar la segunda expresión en el paso 1, se abandona la ejecución del bucle
  4. Se ejecutan las sentencias `S`
  5. Se incrementa el valor de la variable índice en tantas unidades como indique el valor obtenido al evaluar la tercera expresión en el paso 1 y se vuelve al paso 3
- Se deben explicar brevemente cada uno de los atributos y funciones utilizadas en el Esquema de Traducción.

TRADUCTORES DE LENGUAJES

Examen GCI. 12 de junio de 2019

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 18 de junio. 2. Fecha **estimada** de la revisión: 20 de junio. 3. En la web se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 4. La duración de este examen será de 1 hora.

De un lenguaje de programación se ha extraído el siguiente fragmento de gramática:

S → For E To E Do S Next ; | id := E ; | S S
E → id | cte | E Or E | id (L)
L → E | L , E

Se pide diseñar para este lenguaje el **Generador de Código Intermedio** de tres direcciones mediante una **Definición Dirigida por la Sintaxis**, teniendo en cuenta que:

- El lenguaje solo tiene variables y datos de tipo lógico y se asume que el Análisis Semántico está realizado y es correcto
- Debe utilizarse representación numérica para los valores lógicos
- Las variables y funciones exigen haber sido declaradas previamente a su uso
- La expresión puede ser una variable, una constante lógica, un *or* lógico, o una llamada a una función (siendo L los parámetros actuales de la función)
- El funcionamiento del bucle For es como sigue:
  1. Se evalúa las primera expresión
  2. Si la primera expresión se evalúa como falsa, se abandona la ejecución del bucle
  3. Se ejecutan las sentencias S
  4. Se evalúa la segunda expresión
  5. Si la segunda expresión se evalúa como verdadera, se vuelve al paso 3; en caso contrario, finaliza el bucle.
- Asúmase que en el código intermedio no se dispone de operadores lógicos,
- Se deben explicar brevemente cada uno de los atributos y funciones utilizadas en el Esquema de Traducción.

TRADUCTORES DE LENGUAJES

Examen EE+GC. 12 de junio de 2019

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 18 de junio. 2. Fecha **estimada** de la revisión: 20 de junio. 3. En la web se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 4. La duración de este examen será de 1 hora.

Sea el siguiente fragmento de un programa correcto:

```
Global p, x
Function uno (ref a, copyres b, val c)
{
   b:= c
   x:= 1
}
Function dos (ref n, ref z)
{
   p:= p - n
   If (p ≠ 0) Then uno (5 + x, x, 3)
              Else dos (z, n)
   n:= x
}
Function main ()
{
   Local n
   n:= 200
   p:= 4
   x:= p
   dos (p, n)
}
```

Teniendo en cuenta que el lenguaje tiene las siguientes características:

- Todas las variables son enteras
- Las direcciones y los enteros ocupan 1 palabra
- Las funciones no se pueden anidar
- Hay tres modos de paso de parámetros:
  - Por valor (`val`)
  - Por referencia (`ref`)
  - Por copia y restauración (`copyres`). Este modo es un híbrido de los dos anteriores y funciona de la siguiente manera:
    - En la llamada, se evalúa el parámetro actual. Tanto su valor como su dirección se guardan en el parámetro formal
    - Durante la ejecución de la función, se trabaja únicamente con el valor del parámetro formal (igual que si hubiera sido un parámetro por valor)
    - Al regresar, se restaura, es decir, se copia el valor del parámetro formal en la dirección guardada en la llamada

Se pide:

a. La **traza de ejecución** de este programa, detallando todo el contenido de la memoria.

b. La traducción a **código intermedio** y a **código ensamblador** de la instrucción "`p:= p - n`" que aparece en la función `dos`.

TRADUCTORES DE LENGUAJES

12 de julio de 2019

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 22 de julio. 2. Fecha **estimada** de la revisión: 24 de julio. 3. En la web se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 4. La duración de este examen será de 2 horas.

1. De un lenguaje de programación se ha extraído el siguiente fragmento de gramática:

```
S → For E To E Do S ;  |  id := E ;  |  S S
E → id  |  cte  |  E And E  |  id ( L )
L → E  |  L , E
```

Se pide diseñar para este lenguaje el **Generador de Código Intermedio** de tres direcciones mediante un **Esquema de Traducción**, teniendo en cuenta que:

- El lenguaje solo tiene variables y datos de tipo lógico y entero y existe conversión automática de tipos; se asume que el Análisis Semántico está realizado y es correcto
- Debe utilizarse representación numérica para los valores lógicos (0 es falso, distinto de 0 es verdadero)
- Las variables y funciones exigen haber sido declaradas previamente a su uso
- La expresión puede ser una variable, una constante entera, un *and* lógico, o una llamada a una función (siendo `L` los parámetros actuales de la función)
- El funcionamiento del bucle `For` es como sigue:
  1. Se evalúa las primera expresión
  2. Si la primera expresión se evalúa como verdadera, se abandona la ejecución del bucle
  3. Se ejecutan las sentencias `S`
  4. Se evalúa la segunda expresión
  5. Si la segunda expresión se evalúa como falsa, se vuelve al paso 3; en caso contrario, finaliza el bucle.
- Asúmase que en el código intermedio no se dispone de operadores lógicos.
- Se deben explicar brevemente cada uno de los atributos y funciones utilizadas en el Esquema de Traducción.

2. Sea el siguiente fragmento de un programa correcto:

```
Program Junio
Var  Array [1..5] of Integer: Resumen
     Integer: x
   Procedure C (REF p2: Integer) /* por referencia
      Function D (pb2: Integer): Integer /* por valor
      BEGIN/* D
         pb2:= pb2 + 111
         Return pb2
      END  /* D
   BEGIN/* C
      If (p2 <= 3)Then p2:= D(p2)
                  Else C(p2 - 2)
   END  /* C
   Function A (p1: Integer): Integer /* por valor
   Var var1: Integer
      Procedure B (REF pb1: Integer) /* por referencia
      Var var2: Integer
      BEGIN/* B
         var1:= pb1 + 3      /* ×××
         var2:= var1 + 1
         pb1:= var2 * var2
         If (var2 < 6) Then Resumen[var2]:= var2
      END  /* B
   BEGIN/* A
      var1:= 10 + p1
      If (p1 <= 1)Then B(p1)
                  Else C(p1)
      Return p1
   END  /* A
BEGIN/* Junio
   Resumen:= [0,0,0,0,0]
   x:= 1
   Print A(x)
   Print A(Resumen[5])
END  /* Junio
```

El compilador y el lenguaje tienen las siguientes características:

- Las expresiones lógicas se evalúan mediante control de flujo
- El lenguaje tiene estructura de bloques
- Los enteros ocupan 2 bytes; las direcciones ocupan 4 bytes (cada palabra son 2 bytes)
- Los parámetros se pasan por valor, salvo que se indique la palabra **REF**, en cuyo caso se pasan por referencia.

Se pide:

a. Diseñar y describir brevemente el **Registro de Activación** general para este lenguaje.

b. Realizar una **traza de ejecución** del programa representando el contenido completo de la memoria.

c. Indicar el **código intermedio** generado para la instrucción marcada con ***×××***. Indicar la traducción de ese código intermedio a **código ensamblador**.

TRADUCTORES DE LENGUAJES

6 de marzo de 2020

Observaciones:	1. La fecha estimada de publicación de las calificaciones es el 17 de marzo. 2. La fecha estimada de la revisión es el 19 de marzo. 3. En la web se anunciarán las fechas exactas. 4. La duración de este examen es de 1 hora.

Se dispone del siguiente fragmento de gramática de un lenguaje de programación:

$S \rightarrow$ do S while E | id += E | S S
$E \rightarrow$ id | E between E and E | id (L)
$L \rightarrow$ E | L , E

El lenguaje tiene las siguientes características:

- Dispone únicamente de los tipos entero y lógico.
- La sentencia do while ejecuta el cuerpo S, comprueba la condición E, y repite la ejecución del cuerpo mientras que la condición sea cierta.
- Cuando el operador += tiene como operandos dos enteros, se realiza la suma de ambos operandos y su resultado se guarda en el id. Cuando el operador += tiene como operandos dos lógicos, se realiza la operación *or* entre ambos operandos y su resultado se guarda en el id.
- En la expresión E_1 between E_2 and E_3, el operador devuelve cierto si el valor de la expresión E_1 está entre los valores de E_2 y E_3, ambos inclusive, y devuelve falso en caso contrario. Las tres expresiones son siempre enteras (se asume que el valor de E_2 no es nunca mayor que el valor de E_3).
- id (L) es una llamada a una función que devuelve un valor.
- En la llamada a una función, el primer parámetro siempre se pasa por referencia, y todos los demás se pasan por valor.

Se pide, diseñar el **Generador de Código Intermedio** de tres direcciones, mediante un Esquema de Traducción, manejando los lógicos por representación numérica (0 para falso y 1 para verdadero), teniendo en cuenta que el análisis semántico está ya realizado y el programa que se recibe es correcto.

TRADUCTORES DE LENGUAJES

12 de marzo de 2021

Observaciones: 1. La fecha estimada de publicación de las calificaciones es el 23 de marzo.
2. La fecha estimada de la revisión es el 25 de marzo.
3. En la web se anunciarán las fechas exactas.
4. La duración de este examen es de 1 hora.

Se dispone del siguiente fragmento de gramática de un lenguaje de programación:

$S \rightarrow$ Loop S_1 Cond E | Skip | id := E | S_1 S_2 | If E Then Begin S_1 End R
$R \rightarrow$ ElseIf E Then Begin S End | λ
$E \rightarrow$ id | E_1 # E_2

El lenguaje tiene las siguientes características:

- Dispone de los tipos real y lógico.
  - Realiza conversión de tipos automática únicamente de real a lógico en la asignación.
  - Se considera que un número real es falso cuando está próximo a cero (entre -0.5 y +0.5, ambos exclusive); en caso contrario, es verdadero.
- Las expresiones lógicas se manejan por control de flujo.
- El bucle Loop repite la ejecución del cuerpo S_1 hasta que la expresión E sea cierta.
- La sentencia Skip, salta al final de la iteración del Loop para pasar a evaluar la condición del bucle.
- Se ha de usar en el diseño el atributo heredado *.siguiente*.
- El operador lógico # tiene la siguiente tabla de verdad:

E_1	E_2	E
V	F	V
V	V	F
F	F	V
F	V	V

Se pide, diseñar el **Generador de Código Intermedio** de tres direcciones, mediante una Definición Dirigida por la Sintaxis, teniendo en cuenta que el Análisis Semántico está ya realizado y el programa que se recibe es correcto.

TRADUCTORES DE LENGUAJES

21 de mayo de 2021

Observaciones:

1. Las calificaciones se publicarán hacia el 28 de mayo y la revisión será hacia el 1 de junio
2. La duración de este examen es de 60 minutos

Sea un lenguaje de programación basado en Javascript-PDL, pero con la posibilidad de definir funciones anidadas y paso de parámetros por referencia. Se considerará representación numérica para las expresiones lógicas. Las direcciones de memoria, valores enteros y lógicos ocupan siempre 1 palabra. *alert* es una instrucción del lenguaje y no implica una llamada a función. Se usarán los registros IX (apunta a la cima del RA actual) e IY (apunta al inicio de la zona de Datos Estáticos).

```
let number var1;  //global
x=10; //no declarada, se considera global y entera
function number geo (number n, number p) //parámetros por valor
        function number menor (ref number z, number y)
             // z por referencia, y por valor
        {
          z = 2 + y;       // **1
          y = 0;
          return geo (z, y);
        }
        function number mayor (ref number z, number y)
             //parámetro z por referencia, y por valor
        {
          let number v2;  //local
          z = z * 3;
          y = z + 2;
          return menor (z, y);
        }
  {                        //inicio del cuerpo de geo
    if (n * p > 0)
    {  mayor (x-5, p); }
    else {
            if (n * p < 0)
            { menor (n, p); }
            else
            {  return n; }
     }
    return 8;
  }                        //fin del cuerpo de geo
{                          //inicio Programa Principal
  var1 = x – 3;
  x = geo (var1, x);
  alert (var1, x);
}
```

Se pide:

1) Realizar el diseño del Registro de Activación para este lenguaje [1 pt.]
2) Realizar la traza de ejecución de este programa detallando el contenido de los Registros de Activación. [7 puntos]
3) Traducir la instrucción marcada con **1 a Código Objeto [2 pt.]

TRADUCTORES DE LENGUAJES

7 de junio de 2021

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 14 de junio. 2. Fecha **estimada** de la revisión: 17 de junio. 3. En la web se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 4. El tiempo para responder cada pregunta es de 1 hora.

1. Sea el siguiente fragmento de una gramática de un lenguaje de programación:

$S \rightarrow$ for id = C_1 to C_2 do S_1
$S \rightarrow$ id *= E
$S \rightarrow S_1$; S_2
$S \rightarrow$ continue
$S \rightarrow$ ifnt E then S_1
$E \rightarrow C$
$E \rightarrow$ id
$E \rightarrow E_1$ # E_2
$C \rightarrow$ cte_ent

Y teniendo en cuenta las siguientes características del lenguaje:

- El bucle for funciona de la siguiente manera: se asigna a la variable índice id el valor de la primera constante (C_1); si el valor del id no coincide con el de la segunda constante (C_2), se ejecuta el cuerpo del for y se actualiza el valor de la variable índice id (incrementando o decrementado, como se explicará a continuación). La ejecución del bucle termina cuando el valor del id coincide con la segunda constante (C_2); es decir, si ambos valores son iguales, no se ejecuta el cuerpo del for. La variable índice (id) se actualiza incrementando en una unidad su valor, si la primera constante (C_1) es menor o igual que la segunda constante (C_2); en caso contrario, la variable índice se actualiza decrementando su valor en una unidad.
- El lenguaje tiene solo enteros y lógicos.
- Los lógicos se tratarán por representación numérica.
- El código intermedio no dispone de la operación *and*.
- La asignación con operación *= se corresponde con una multiplicación entre enteros (equivale a id = id * E) o un *AND* cuando el id es lógico (equivale a id = id and E). En este último caso, hay conversión implícita de tipos (de entero a lógico) para la expresión (E).
- La sentencia ifnt ejecuta el cuerpo (S_1) si la expresión E es falsa.
- El operador lógico # tiene la siguiente tabla de verdad:

E_1	E_2	#
V	F	V
V	V	F
F	F	V
F	V	V

Se pide diseñar el **Generador de Código Intermedio** mediante un **Esquema de Traducción**, para obtener código de 3 direcciones, detallando todos los accesos a la Tabla de Símbolos.

2. Sea el siguiente fragmento de un programa correcto:

```
var b: integer; // global
Procedure principal;
 var a: integer;     // local
 function integer primer (x, y: integer);      // parámetros por valor
     var a: integer;      // local
     function integer segun (x, z: integer ref);     // parámetros por referencia
     begin      // segun
        if (z > 0) then y:= segun (x, -z);
        return x * z;
     end;
 begin     // primer
  a:= x / y;         // división entera (resultado truncado sin decimales)
  a:= x – (a * y);
  if (a < 5)
     then a:= segun (a, b);
     else a:= segun (b, a);
  return y * b;           // ◄◄
 end;
begin      // principal
 b:= 10;
 print "Introduce los dígitos de tu número de matrícula";
 input a;
 print primer (a, b);
end.
```

El compilador y el lenguaje tienen las siguientes características:

- Las expresiones lógicas se evalúan mediante control de flujo
- El lenguaje tiene estructura de bloques con funciones anidadas
- Los enteros, lógicos y direcciones ocupan 1 palabra
- El registro IY apuntará al inicio de la zona de Datos Estáticos y el registro IX apuntará al inicio del Registro de Activación activo.

Se pide:

a. Diseñar y describir brevemente el **Registro de Activación general** para este lenguaje.

b. Realizar una **traza de ejecución** del programa representando el contenido completo de la memoria.

c. Indicar el **código intermedio** que se generaría para la instrucción marcada con ◄◄ e indicar la traducción de ese código intermedio a **código ensamblador**.

TRADUCTORES DE LENGUAJES

Examen Final - 24 de junio de 2021

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 1 de julio. 2. Fecha **estimada** de la revisión: 5 de julio. 3. En la web se avisará la fecha exacta de publicación de las calificaciones y la fecha y hora definitiva de la revisión. 4. El tiempo para responder el examen es de 2 horas. 5. No se responderán preguntas sobre el enunciado debido a la situación sanitaria. Deberá describirse cualquier decisión de diseño asumida no especificada en el enunciado.

1. Dado el siguiente fragmento de gramática de un lenguaje de programación:

P → D S
D → var int L = I D | λ
L → id | id , L
I → E | E , I
S → if E < E then S | id += E | S S
E → id | cte_ent | E + E

Se pide diseñar el **Generador de Código Intermedio** mediante una Definición Dirigida por la Sintaxis para esta grámatica, teniendo en cuenta que:

- El análisis semántico ya ha sido realizado y no hay errores (entre otras cosas, se ha comprobado ya que todas las variables se han declarado previamente a su uso y que la lista de valores en una declaración con inicialización tiene la longitud correcta).
- En una declaración múltiple, el tipo se aplica a toda la lista de variables.
- El operador += equivale a una asignación con suma (`a+=b` ⇔ `a=a+b`).
- Un ejemplo de un programa válido sería:

```
var int x, y, z = 1+2+3+4, x, x+99
var int h = 7
var int f, g = h, 8
if g < 5 then y+= f
if 2 + h < f + 4 then if x < 9 then y+= g
```

2. Dado el siguiente programa correcto:

```
Program Principal; // principal
   VarG a:=3: integer;     // VarG define una variable global
   Var b: integer;         // local
   Function integer SUMA (Ref x: integer; y: integer)  // x por referencia, y por valor
      Var a;         // local
   Begin SUMA
      a:= x + y;     // ***
      Return a;
   End SUMA;
   Function integer ASIGNA ()
      Var b;         // local
      Function integer PRODUCTO (x: integer)  // x por valor
      Begin PRODUCTO
         Return 2 * x;
      End PRODUCTO;
   Begin ASIGNA
      b:= SUMA (PRODUCTO (1), a);
      a:= b;
      Return PRODUCTO (a);
   End ASIGNA;
Begin Principal
   a:= ASIGNA ();
   Print (a);
End Principal;
```

El compilador y el lenguaje tienen las siguientes características:

- El lenguaje tiene estructura de bloques con funciones anidadas.
- Se considerará que los enteros ocupan 1 palabra y las direcciones de memoria 2 palabras.
- El registro IY apuntará al inicio de la zona de Datos Estáticos y el registro IX apuntará al inicio del Registro de Activación activo.

Se pide:

a. El diseño general del **Registro de Activación** para cualquier función de este lenguaje

b. La **Traza de Ejecución** de este programa, detallando el contenido de la pila y los registros de activación.

c. El **Código Objeto** correspondiente a la sentencia ***. Se recomienda escribir antes el código intermedio y explicar brevemente cada instrucción del código objeto resultante.

d. Indicar los **cambios** que habría que hacer al código objeto del apartado anterior si la sentencia a traducir fuera `b:= x + y` (situada en el mismo lugar del programa).

TRADUCTORES DE LENGUAJES

11 de marzo de 2022

Observaciones:	1. La fecha estimada de publicación de las calificaciones es el 25 de marzo. 2. La fecha estimada de la revisión es el 29 de marzo. 3. En la web se anunciarán las fechas exactas. 4. La duración de este examen es de 1 hora.

Se dispone del siguiente fragmento de gramática de un lenguaje de programación:

$S \rightarrow$ for id = E to E do S when E | id = E | S S
$E \rightarrow E >> E \mid E - E \mid E$ nor $E \mid$ id

Teniendo en cuenta las siguientes características:

- Dispone únicamente de los tipos entero y lógico y no existe conversión automática de tipos.
- El lenguaje exige declaración previa de variables
- La sentencia for funciona de la siguiente manera: se inicializa la variable índice con la primera expresión E; si el valor de la variable índice es menor o igual que el de la segunda expresión, se ejecuta el cuerpo del for; si la tercera expresión (la que acompaña al when) se evalúa como cierta, la variable índice se actualiza incrementándose en una unidad (pero si la expresión es falsa, no hay actualización); a continuación, se vuelve a comprobar la condición para ver si hay que volver a ejecutar el cuerpo del for; el bucle termina cuando la variable índice es mayor que la segunda expresión.
- El operador mucho mayor que (>>) recibe dos enteros y devuelve cierto si el valor del primer operando es al menos el doble que el del segundo operando.
- El operador lógico nor recibe dos operandos lógicos y devuelve cierto solo si ambos operandos son falsos.
- El operador de resta binaria (–) recibe dos operandos enteros y devuelve un entero cuyo valor es la resta de ambos operandos.
- Se debe usar el atributo heredado *.sig* en la solución.

Se pide diseñar el **Generador de Código Intermedio** de tres direcciones, mediante un Esquema de Traducción, manejando los lógicos por control de flujo, teniendo en cuenta que el análisis semántico está ya realizado y el programa que se recibe es correcto.

TRADUCTORES DE LENGUAJES

Examen GCI. 8 de junio de 2022

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 16 de junio. 2. Fecha **estimada** de la revisión: 20 de junio. 3. En la web se avisarán las fechas exactas. 4. La duración de este examen será de 1 hora.

De un lenguaje de programación se ha extraído el siguiente fragmento de gramática:

S → when E do S until id = E | id := E | S; S | λ

E → E // E | E pot E | E xor E | id

Se pide diseñar, para este fragmento de lenguaje, el **Generador de Código Intermedio** para obtener código de tres direcciones mediante un Esquema de Traducción, representando numéricamente los valores lógicos y teniendo en cuenta que:

- El lenguaje exige declaración previa de variables.
- Los tipos del lenguaje son entero y lógico.
- Hay conversión automática entre enteros y lógicos.
- El bucle when-do-until funciona de la siguiente manera: Se evalúa la primera expresión (la E del when); si es falsa, se termina la ejecución del bucle, pero, si es cierta, se ejecuta el cuerpo del bucle (S). A continuación, se evalúa la condición de final (la comparación id = E del until) y, si es cierta, se termina el bucle, pero, si es falsa, se vuelve al inicio del bucle para volver a comprobar la primera expresión y volver a ejecutar el cuerpo S si fuera cierta. El bucle termina, por tanto, cuando la expresión del when es falsa o cuando la condición del until es cierta.
- El operador 'es divisible por' (//) recibe dos enteros y devuelve cierto si el resto de dividir el primero entre el segundo es cero, y falso en caso contrario. El operador módulo existe en el lenguaje intermedio.
- El operador lógico xor recibe dos operandos lógicos y devuelve cierto solo si uno de los operandos es cierto, pero no ambos.
- El operador potencia (pot) recibe dos operandos enteros (que toma como base y exponente de una exponenciación, respectivamente) y devuelve un entero cuyo valor es la base elevada al exponente. Se asume que siempre se recibe como exponente un número superior a cero. El operador potencia no existe en el lenguaje intermedio.

TRADUCTORES DE LENGUAJES

Examen EE+GC. 8 de junio de 2022

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 16 de junio. 2. Fecha **estimada** de la revisión: 20 de junio. 3. En la web se avisarán las fechas exactas 4. La duración de este examen será de 1 hora.

Sea el siguiente fragmento de un programa correcto:

```
Global int u;    // variables globales
Global int a;
Global char s[13]= "IAMGEOCENTRIC"; // s es una cadena (vector 1..13) de caracteres

imprime (int b)  // paso de parámetros siempre por valor
{
   if (b > 0) printc (s[b]);  // imprime el carácter que está en la posición b de s
}

int formula (int i)
{
   Local int x= a;      // variable local a la función
   while (i < a)
   {
      a= a - u;
      x= x + formula (i + u);
   }
   if (x > 0) printc (s[x]);  // imprime el carácter que está en la posición x de s
   return x - a;        // instrucción que hay que traducir
}

int valor (int d)
{
   input (u);    // lee un número del teclado: el dígito introducido por el usuario
   if (u > 9) return valor(d);
   return u + d;
}

main ()   // las funciones no se pueden anidar
{
   u= 2;     // prints imprime una cadena:
   prints ("Introduce el último dígito de tu número de matrícula");
   a= 5;
   u= valor (u) - 1 - u;
   imprime (formula (u));
}
```

Además de lo indicado en el programa, el lenguaje tiene las siguientes características:

- Las direcciones, caracteres y enteros ocupan 1 palabra.
- Una variable de tipo cadena se declara como un vector de caracteres (`char [13]`), indicando su tamaño (13, en este caso), inicializándose con los caracteres de la cadena. Se puede acceder a cada carácter de la cadena usando la notación de vector indicando su posición (`s[10]` es '`T`'). Los índices van de 1 hasta el tamaño del vector. Acceder a un índice no válido es un error de ejecución, pero nunca se produce en la ejecución de este programa.

Se pide:

a. Justificar qué campos tendría el **Registro de Activación** de este lenguaje y cuáles no.

b. Detallar una **traza de ejecución** de este programa, detallando todo el contenido de la memoria e indicando qué palabra imprime por pantalla el programa.

c. La traducción a **código intermedio** y a **código ensamblador** de la instrucción "`return x - a;`".

TRADUCTORES DE LENGUAJES

28 de junio de 2022

Observaciones: 1. Las calificaciones se publicarán hacia el 4 de julio. La revisión será hacia el 6 de julio. En la web se avisará de las fechas exactas.
2. La duración de este examen es de 2 horas.
3. No se responderán preguntas sobre el enunciado. Deberá describirse cualquier decisión de diseño que no esté especificada.
4. Las dos preguntas tienen la misma puntuación y se entregan por separado.

1. Dado el siguiente fragmento de gramática de un lenguaje de programación:

```
S → for id := E X E do begin S end | A | S ; S
X → to | downto
A → L := I
L → id | id , L
I → E | E , I
E → id | cte_ent | E + E
```

Se pide diseñar el **Generador de Código Intermedio** mediante una **Definición Dirigida por la Sintaxis** para esta grámatica, teniendo en cuenta que:

- En una asignación múltiple (A), se tienen dos listas de igual longitud: en el lado izquierdo se tiene una lista de variables (L) y en el lado derecho una lista de expresiones (I). Al identificador i-ésimo se le asigna la expresión i-ésima.
- El índice (id) del for varía unidad a unidad. Se puede ir aumentando (to) o disminuyendo (downto).
- El bucle for funciona de la siguiente manera: el índice se inicializa con la primera expresión (E); el bucle for ejecuta su cuerpo (S) siempre que el índice tenga un valor distinto al valor de la segunda expresión (E). Hay que tener en cuenta que esta segunda expresión se evalúa en cada iteración del bucle.
- Se debe asumir que todos los identificadores son enteros.
- Un ejemplo de un programa válido es:

```
h:= 7;
x, y, z:= 3, 2 + h, 9;
x, z:= z, x;
for x:= x + 8 downto h do
begin
  y, z:= 3 + 5 + x, y + y
end
```

2. Dado el siguiente programa correcto:

```
Function principal;
Var x, y: integer;
     Function cantidad (x, y: integer): integer;
     begin
          return x + y
     end;
     Function sumatorio (x: integer): integer;
     begin
          y:= x;     (***)
          if (x<0)  then write ("El parámetro no debe ser negativo")
                    else if (x=0) then return 1
                         else return x + sumatorio (x-1)
     end;
begin (*principal*)
     y:= cantidad (1,1);
     x:= sumatorio (cantidad (1,1));
     write ("El resultado es ", x)
end (*principal*).
```

Se tienen las siguientes características:

- El lenguaje tiene estructura de bloques con funciones anidadas.
- Se considerará que los enteros y las direcciones de memoria ocupan 1 palabra.
- El paso de parámetros es siempre por valor.

Se pide:

a. El diseño general del **Registro de Activación** para este lenguaje

b. La **Traza de Ejecución** de este programa, detallando el contenido de la pila y los registros de activación.

c. El **Código Objeto** correspondiente a la sentencia `y:= x (***)`. Se recomienda explicar brevemente cada instrucción del código objeto resultante.

TRADUCTORES DE LENGUAJES

10 de marzo de 2023

Observaciones:	1. La fecha estimada de publicación de las calificaciones es el 23 de marzo. 2. La fecha estimada de la revisión es el 28 de marzo. 3. En la web se anunciarán las fechas exactas. 4. La duración de este examen es de 1 hora.

Se dispone del siguiente fragmento de gramática de un lenguaje de programación:

$S \rightarrow$ for id = E_1 to E_2 begin S_1 end else begin S_2 end
$S \rightarrow$ break ;
$S \rightarrow$ if E then begin S_1 end
$S \rightarrow$ id = E ;
$S \rightarrow S_1\ S_2$
$E \rightarrow$ id
$E \rightarrow E_1$ / E_2
$E \rightarrow E_1$ nand E_2

El lenguaje tiene las siguientes características:

- Dispone únicamente de los tipos entero y lógico, y no existe conversión automática de tipos
- Exige declaración previa de variables
- El operador de división (/) recibe dos enteros y devuelve el resultado de la división entera
- El operador lógico nand devuelve 'falso' si ambos operandos son 'verdaderos'; en caso contrario, devuelve 'verdadero'
- La sentencia for-else tiene dos partes bien diferenciadas: una iterativa, que afecta al conjunto de sentencias S_1 dependiendo del índice del for, y una alternativa, que afecta a la parte del else y que se ejecuta una o ninguna vez. Funciona de la siguiente manera:
    1. Se asigna a la variable índice id del for el valor de la primera expresión E_1; si esta variable índice es menor o igual que la segunda expresión E_2, se ejecutan las sentencias S_1; sin embargo, si la variable índice es mayor que la segunda expresión E_2 se para de iterar y no se ejecuta S_1 (la expresión E_2 se evalúa solamente una vez al principio del bucle).
    2. Tras cada iteración, la variable índice del for se incrementa en una unidad y se vuelve a comprobar si su valor supera o no al de la segunda expresión E_2, para decidir si se vuelve a iterar.
    3. Si en alguna iteración del for, la ejecución encuentra una sentencia break, se termina el for-else.
    4. Si al finalizar la última iteración del for, no se ha ejecutado ninguna vez la sentencia break, entonces se ejecutarán las sentencias del else.

Se pide diseñar el **Generador de Código Intermedio** de tres direcciones, mediante un Esquema de Traducción, manejando los lógicos por control de flujo, teniendo en cuenta que las comprobaciones semánticas están ya realizadas y el programa que se recibe es correcto.

TRADUCTORES DE LENGUAJES

Examen GCI. 6 de junio de 2023

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 14 de junio. 2. Fecha **estimada** de la revisión: 19 de junio. 3. En la web se avisarán las fechas exactas. 4. La duración de este examen será de 1 hora.

De un lenguaje de programación se ha extraído el siguiente fragmento de gramática:

$E \rightarrow E_1$ out { E_2 , E_3 } | E_1 + E_2 | id | E_1 >> E_2

$S \rightarrow$ if E then S_1 | loop E S_1 endloop | S_1 ; S_2

Se pide diseñar, para este fragmento de lenguaje, el **Generador de Código Intermedio** mediante sendas **Definiciones Dirigidas por la Sintaxis**, la primera asumiendo una representación numérica de los valores lógicos y, la segunda, asumiendo una representación mediante control de flujo para los lógicos.

- El lenguaje exige declaración previa de variables.
- Los tipos del lenguaje son entero y lógico.
- No hay conversión automática entre enteros y lógicos.
- El operador 'suma' (+) realiza una suma entre enteros.
- El operador lógico out recibe tres expresiones enteras y devuelve cierto solo si E_1 es menor que E_2 y mayor que E_3.
- El operador lógico >> devuelve cierto si E_1 es falso; en caso contrario, devuelve el valor lógico de E_2.
- El bucle loop funciona de la siguiente manera: Se evalúa la expresión E, que es un entero; si es un número mayor que cero, se ejecuta el cuerpo (S_1) del loop y se vuelve a evaluar la expresión; si vale cero o menos, se sale del loop.

TRADUCTORES DE LENGUAJES

Examen EE+GC. 6 de junio de 2023

Observaciones:	1. Fecha **estimada** de publicación de las calificaciones: 14 de junio. 2. Fecha **estimada** de la revisión: 19 de junio. 3. En la web se avisarán las fechas exactas 4. La duración de este examen será de 1 hora.

Sea el siguiente fragmento de un programa correcto:

```
1.  Global p: array [4] of char = "♦♠♥♣";
2.  Global v: array [30] of char = "0AbcKingQueenJack123456789";
3.  procedure main ();
4.     Local z: integer;
5.     function H (Val a: integer): integer;          // paso de parámetro por valor
6.        Local x: integer;
7.        procedure p (Ref a: integer);           // paso de parámetro por referencia
8.        Begin p
9.           a:= a * 2;
10.          print v [a];      // imprime el carácter que está en la posición a de v
11.       End p
12.    Begin H
13.       x:= 3;
14.       if (a < x) then
15.       Begin if
16.          if (a = z) then  a:= H (a + 1);
17.          p (a);
18.          H (a);
19.       End if
20.       return a * x;
21.    End H
22. Begin main
23.    z:= 0;
24.    print p [H (z) / 12];                     // imprime un carácter del vector p
25. End main
```

Además de lo indicado en el programa, el lenguaje tiene las siguientes características:

- Las direcciones, caracteres y enteros ocupan 1 palabra.
- El lenguaje permite declarar variables globales (`Global`) y variables locales (`Local`) a procedimientos y funciones.
- Una variable cadena puede declararse como un vector de caracteres (`array [30] of char`), indicando su tamaño máximo (30, en este caso) e inicializándose con los caracteres de la cadena. Se puede acceder a cada carácter de la cadena usando la notación de vector indicando su posición teniendo en cuenta que los índices van de 1 hasta el tamaño del vector (`v[3]` es 'b').
- El Generador de Código Intermedio ha representado los valores lógicos por control de flujo.
- El compilador no realiza ningún tipo de optimización.

Se pide:

a. Justificar qué campos tendría el **Registro de Activación** de este lenguaje y cuáles no.

b. Detallar una **traza de ejecución** de este programa, detallando todo el contenido de la memoria e indicando qué imprime por pantalla el programa.

c. Explicar la traducción a **código ensamblador** de la comparación "`(a = z)`" de la línea 16, asumiendo que el código intermedio obtenido es "`if a = z goto EtV; goto EtF`".

TRADUCTORES DE LENGUAJES

8 de marzo de 2024

Observaciones:	1. La fecha estimada de publicación de las calificaciones es el 18 de marzo. 2. La fecha estimada de la revisión es el 20 de marzo. 3. En la web se anunciarán las fechas exactas. 4. La duración de este examen es de 1 hora.

Se dispone del siguiente fragmento de gramática de un lenguaje de programación:

$S \rightarrow \text{if } E_1 : S_1 \text{ elif } E_2 : S_2 \text{ else } S_3$
$S \rightarrow \text{id} := E_1 \ ? \ E_2 : E_3 \ ;$
$S \rightarrow S_1 \ S_2$
$E \rightarrow \text{id}$
$E \rightarrow E_1 > E_2$
$E \rightarrow E_1 \text{ nor } E_2$

El lenguaje tiene las siguientes características:

- Dispone únicamente de los tipos entero y lógico, y no existe conversión automática de tipos
- Exige declaración previa de variables
- El lenguaje tiene evaluación perezosa, por lo que las expresiones lógicas se deben representar por control de flujo
- El operador condicional (?) devuelve el resultado de E_2 si la condición E_1 es 'verdadero' y devuelve el resultado de E_3 en caso contrario. La variable id es siempre entera
- El operador relacional (>) solo se puede utilizar con expresiones enteras
- El operador lógico nor devuelve 'verdadero' si ambos operandos son 'falsos; en caso contrario, devuelve 'falso'
- La sentencia condicional if funciona de la siguiente manera: si la expresión E_1 es 'verdadera' se ejecutan las sentencias S_1; en caso contrario, se comprueba la expresión E_2; si E_2 es 'verdadera', se ejecutan las sentencias S_2; en caso contrario, se ejecutan las sentencias S_3.

Se pide diseñar el **Generador de Código Intermedio** de tres direcciones, mediante un Esquema de Traducción, teniendo en cuenta que las comprobaciones de tipos están ya realizadas y el programa que se recibe es correcto.