Implementar con Java, JFlex y Cup el compilador del lenguaje fuente PyPL que se describe a continuación, traduciéndolo a código ensamblador de tres direcciones

EL CÓDIGO FUENTE:

El lenguaje fuente tiene una sintaxis similar a Python y consiste en un conjunto de sentencias. El lenguaje solo admite variables globales de tipo entero. El lenguaje puede contener expresiones aritméticas simples, expresiones relacionales, sentencias de asignación; así como sentencias de control ***if-elif-else***, ***while y for***. Por ejemplo, el siguiente es un programa válido en PyPL:

MiProg.pypl

```
x = 4
i = 1
factorial = 1
while i<x+1 :
  factorial = factorial * i
  i = i+1
print factorial
```

Las características mas destacadas de PyPL son las siguientes:

Declaración de variables:

No es necesario declarar las variables, pero si inicializarlas antes de usarlas

Comentarios:

A partir del carácter #, se ignora el resto del texto, considerándolo un comentario.

```
print("Esto no es un comenttario")
#print("Esto si")
```

Se pueden hacer comentarios de múltiples líneas escribiendo tres apóstrofos (“single quote”) seguidos, y terminando de la misma forma.

```
'''
print("Esto esa dentro de un comentario")
print ("y esto tambien")
'''
print("esto no esta dentro de un comentario")
```

Operadores:

Se consideran siete operadores binarios: suma, resta, multiplicación, división, módulo, potencia y división entera; y el operador menos unario. A los efectos de este ejercicio, puesto que se consideran solo operaciones entre enteros, el resultado de la división y la división entera es el mismo.

```
print (3 + 4)
print (3 - 4)
print (3 * 4)
print (3 / 4)  # resultado 0 (en Pyton 0.75)
print (3 % 2)
print (3 ** 4) # 3 elevado a 4 = 81
print (3 // 4) # resultado 0
```

Sentencia-***if-elif-else***:

La sentencia ***if*** requiere de una operación condicional, los operadores condicionales son los mismos que en otros lenguajes: igual, distinto, mayor, mayor o igual, menor, menor o igual. En una sentencia ***if***, puede contener cero, una o mas partes ***elif***, y opcionalmente una sección ***else***.

```
a = 20
if a >= 22:
    print(a-22)
elif a >= 21:
    print(a-21)
elif a >= 10 :
    print(a-10)
else:
    print(a)
```

Anidamiento:

En PyPL no existen las llaves, pero las sentencias pueden anidarse, y las sentencias pueden formar bloques de sentencias.. El anidamiento se obtiene a partir de la identación, de manera que dos líneas consecutivas que están identadas de la misma forma se supone que forman un bloque de sentencias. Se admite cualquier nivel de anidamiento

```
a = 5
if a > 0:
    print (a+a)
    print (a*a)
    print (a/a)
print(a)
```

```
a = 5
if a > 0:
    print (a+a)
    if (a > 10) :
       print (a*a)
       print (a/a)
print(a)
```

Sentencia-***while***:

La sentencia ***while*** tiene una sintaxis similar a la sentencia ***if***, (pero sin parte ***elif***, ni ***else***). La semántica es la misma que la sentencia ***while*** de otros lenguajes, repetir la instrucción o instrucciones, mientras se cumpla la condición:

```
i = 3
while i < 5:
    print(i)
    i = i+1
print(i)
```

Sentencia-***for***:

A los efectos de esta practica, la sentencia ***for*** se utilizara siempre con un rango fijo definido por dos números enteros, que indican el valor inicial y el valor de comparación del limite superior (es decir el bucle se ejecuta desde que la variable toma el valor inicial y mientras sea estrictamente menor que el máximo, incrementando en 1 en cada iteración):

```
for i in range(1,3):
    print (i)
```

NOTA:

Para una descripción del lenguaje Python, puede usarse la referencia:
https://wiki.python.org/moin/BeginnersGuide/Programmers

EL CÓDIGO OBJETO:

El código objeto es a su vez una extensión del código intermedio utilizado en la practica principal de la asignatura. Todas las variables del código intermedio se considera que están previamente definidas y que su valor inicial es 0.

El conjunto de instrucciones del código ensamblador, y su semántica son las siguientes:

Instrucción	Acción
`x = a ;`	Asigna el valor de a en la variable x
`x = a + b ;`	Suma los valores de a y b, y el resultado lo asigna a la variable x
`x = a - b ;`	Resta los valores de a y b, y el resultado lo asigna a la variable x
`x = a * b ;`	Multiplica los valores de a y b, y el resultado lo asigna a la variable x
`x = a / b ;`	Divide (div. entera) los valores de a y b , y el resultado lo asigna a la variable x
`goto l ;`	Salto incondicional a la posición marcada con la sentencia "`label l`"
`if (a == b) goto l ;`	Salta a la posición marcada con la sentencia "`label l`", si y solo si el valor de a es igual que el valor de b
`if (a < b) goto l ;`	Salta a la posición marcada con la sentencia "`label l`", si y solo si el valor de a es estrictamente menor que el valor de b.
`l:`	Indica una posición de salto.
`label l ;`	Indica una posición de salto. Es otra forma sintáctica equivalente a la anterior.
`print a ;`	Imprime el valor de a
`error ;`	Indica una situación de error, pero no detiene la ejecución.
`halt ;`	Detiene la ejecución. Si no aparece esta instrucción la ejecución se detiene cuando se alcanza la última instrucción de la lista.
`# ….`	Cualquier línea que comience con un # se considera un comentario.

En donde `a,b` representan tanto variables como constantes enteras, `x,y` representan siempre una variable y `l` representa una etiqueta de salto.

EVALUACIÓN DE ESTE EJERCICIO:

Para la corrección de este ejercicio, se tendrán en cuenta resultados parciales según el número de casos de prueba que supere el compilador, de acuerdo a los siguientes criterios:

a) Implementación de la sentencia print que contenga solamente expresiones con números enteros. Se deben implementar los operadores aritméticos +,-,*,/,%,//, operador menos unario, y paréntesis. (6 puntos)
b) Eliminación de comentarios. (4 puntos)
c) Sentencia `if-elif-else`. La parte "*elif*" y "*else*" es opcional, al igual que en Java. Se deben implementar los seis operadores relacionales <=, <, ==, !=, >, >= (4 puntos)
d) Sentencia `while` (4 puntos)
e) Sentencia `for` (4 puntos)
f) Anidamiento de sentencias según la identación (Sentencias compuestas) (8 puntos)

NOTAS IMPORTANTES:

1. Toda práctica debe contener al menos tres ficheros denominados "`PYPLC.java`", "`PYPLC.flex`" y "`PYPLC.cup`", correspondientes respectivamente al programa principal y a las especificaciones en Jflex y Cup. Para realizar la compilación se utilizarán las siguientes instrucciones:

```
cup PYPLC.cup
jflex PYPLC.flex
javac *.java
```