






✖ CONSTRUCCIÓN DEL ANALIZADOR SEMÁNTICO CON BISON

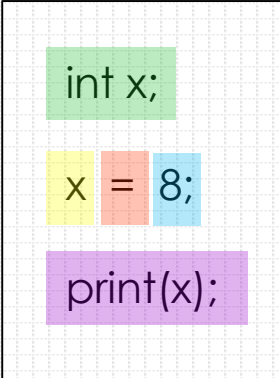
- ✖ SEMÁNTICO: transparencias 4 a 13
- ✖ GENERACIÓN DE CÓDIGO: nada

- ✖ Se va a utilizar una estrategia de desarrollo incremental con el objetivo de no esperar a tener todo el compilador desarrollado para empezar a probarlo.
- ✖ Se debe aprovechar lo desarrollado en la práctica 0 (Nasm)
- ✖ El objetivo inicial es compilar un programa muy sencillo:

```
main
{
    int x;
    x = 8;
    print(x);
}
```

- ✗ Para compilar el programa anterior es necesario implementar las partes del analizador semántico y el generador de código relativas a:

- ✗ Declaración de variables 
- ✗ Uso de variables 
- ✗ Gestión de constantes enteras 
- ✗ Asignación 
- ✗ Impresión de variables 



```
int x;  
x = 8;  
print(x);
```

- ✗ Además también será necesario generar código para las partes inicial y final del fichero ensamblador.
- ✗ Si se ejecuta la práctica de análisis sintáctico se pueden ver las producciones que se reducen cuando se compila el programa anterior.

✗ DECLARACIÓN DE VARIABLES LOCALES Y GLOBALES

- ✗ SEMÁNTICO: transparencias 14-33
- ✗ GENERACIÓN DE CÓDIGO: transparencias 7-12

✗ USO DE IDENTIFICADORES

- ✗ SEMÁNTICO: Transparencias 40-42 del fichero
- ✗ GENERACIÓN DE CÓDIGO: nada

✗ GESTIÓN DE CONSTANTES ENTERAS

- ✗ SEMÁNTICO: transparencias 51, 52, 54, 55 y 57
- ✗ GENERACIÓN DE CÓDIGO: Transparencias 18 y 20

✗ ASIGNACIÓN

- ✗ SEMÁNTICO: transparencias 63 y 64
- ✗ GENERACIÓN DE CÓDIGO: Transparencias 47 y 48
- ✗ NOTA: si se quiere se puede explicar la siguiente transparencia .

✖ OPERACIONES DE SALIDA

- ✖ SEMÁNTICO: transparencia 72
- ✖ GENERACIÓN DE CÓDIGO: transparencias 52-54

✖ CÓDIGO DEL FINAL DEL PROGRAMA ENSAMBLADOR

- ✖ La última instrucción del programa ensamblador se puede escribir en la producción correspondiente al axioma.
- ✖ La instrucción ensamblador es "ret"

✖ NOTAS

- ✖ Para depurar errores en la generación de código, es recomendable, cada vez que se escribe código en el fichero ensamblador, escribir en formato comentario, la línea de código ALFA que corresponde al código escrito.

ANÁLISIS SEMÁNTICO Y GENERACIÓN DE CÓDIGO

asignacion: TOK_IDENTIFICADOR '=' exp

{

 Buscar en la tabla de símbolos (todos los ámbitos abiertos) el identificador \$1.lexema

 Si no existe

 {

 MOSTRAR MENSAJE ERROR SEMÁNTICO

 TERMINAR CON ERROR

 }

 En caso contrario hacer lo correspondiente a la producción

 {

 COMPROBACIONES SEMÁNTICAS

 Si el identificador es una función → ERROR SEMÁNTICO

 Si el identificador es un vector → ERROR SEMÁNTICO

 Si el identificador y la expresión son de distinto tipo → ERROR SEMÁNTICO

 GENERACIÓN DE CÓDIGO: ACCESO AL VALOR DE exp

 Si exp es una constante, ya se tiene su valor, si no, hay que acceder a él

 GENERACIÓN DE CÓDIGO: DIRECCIÓN DE TOK_IDENTIFICADOR

 Si el identificador es una variable global, su dirección es su lexema

 Si el identificador es un parámetro o una variable local, su dirección se expresa en función de ebp y la posición del parámetro o variable local

 GENERACIÓN DE CÓDIGO: HACER EFECTIVA LA ASIGNACIÓN

 Una vez que se dispone del valor de exp y de la dirección de TOK_IDENTIFICADOR la asignación ya se puede hacer efectiva (es un movimiento)

 }

}