En hoc 1 hay una pila cual es?

1.- La pila de yacc

En hoc 4 hay 2 pilas cuales son?

1.-De yacc

2.-De máquina virtual

En hoc 6 hay 3 pilas cuales son?

1.-De yacc

2.-De máquina virtual

3.-Pila de llamadas a funciones

**Falso o verdadero (F/V)**

1.-Una gramática no es ambigua si existe alguna cadena de terminales que pueda obtenerse mediante árboles de análisis sintáctico distintos (dos árboles distintos dan la misma cadena) **( F )**

2.-Dos gramáticas son equivalentes si generan el mismo lenguaje.\_\_\_\_\_\_\_\_ **( V )**

3.-Dos bloques basicos son equivalentes sicalculan el mismo conjunto de expresiones \_\_\_\_\_\_\_\_\_ **( V )**

4.-En el Análisis sintáctico ascendente el árbol de análisis sintáctico la construcción se inicia en la raíz y avanza hacia las hojas **( F )**

5.-En el Análisis sintáctico descendente se construye el árbol de análisis sintáctico de la cadena de desdelas hojas y avanza hacia las raíz **( F )**

6.-el arbol sintactico no es una version condensada del arbol de analisis sintactico **( F )**

7.-Un Esquema de traducción asocia a cada símbolo de una GLC un conjunto de atributos y a cada producción, un conjunto de reglas semánticas para calcular los valores de los atributos asociados con los símbolos que aparecen en esa producción. **( F )**

8.-Definición dirigida por la sintaxis es una GLC en la que se encuentran intercalados, en los lados derechos de las producciones, fragmentos de programa llamados acciones semánticas. **( F )**

9.-Los valores de los atributos sintetizados se calculan a partir de los valores de atributos de su nodo padre o sus nodos hermanos. **( F )**

10.-El código de tres direcciones consiste en una secuencia de instrucciones, cada una de las cuales tiene como máximo tres operandos. **( V )**

11.-En lenguaje C los parámetros formales son como variables locales que ya fueron inicializadas en el momento de la llamada a la función o procedimiento **( V )**

12.-En lenguaje C las variables locales (no estáticas) se crean cuando se entra a una función y se destruyen cuando se sale de la función **( V )**

13.-En hoc los parámetros usados dentro de la definicion de las funciones no tienen nombre\_\_\_\_\_\_\_\_\_\_\_ **( V )**

14.-No es posible definir funciones recursivas en hoc\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **( F )**

15.-En hoc no hay variables locales\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **( V )**

16.-En hoc cuando una función termina su ejecución se saca su marco de la pila de llamadas. **( V )**

17.-En hoc los parámetros reales son listas de expresiones \_\_\_\_\_\_\_\_\_\_\_\_ **( V )**

18.-En hoc el código que ejecuta la maquina virtual de pila esta en prefijo (considere como se ejecuta una operacion de suma **( F )**

19.-En hoc el tipo de las variables es entero **( V )**

20.-En hoc los parámetros reales se meten a la pila **( V )**

1.-Indica gráficamente cómo del símbolo inicial de una gramática deriva una cadena del lenguaje. a) árbol de análisis sintáctico con anotaciones b) árbol sintáctico

c) árbol de análisis sintáctico d) Ninguno de los anteriores 2.-El código de tres direcciones se usa en a) El análisis sintáctico b) El análisis léxico c) Generación de código intermedio d) Generación de código 3.-Un \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ es [A→ α . β ,a] donde A→ αβ es una producción y a es un terminal o $. a) mango b) prefijo viable c) elemento LR (1) d) elemento LR (0) 4.-Es una producción de G con un punto en cierta posición del lado derecho. a) mango b) prefijo viable c) elemento LR (1) d) elemento LR (0) 5.-Son prefijos de las formas de frase derecha que pueden aparecer en la pila a) mango b) elemento LR (1) c) prefijo viable d) elemento LR (1)

6.-Un \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ de una forma de frase derecha g es una producción S → y una posición de g donde la cadena podría encontrarse y sustituirse por A para producir la forma de frase derecha previa en una derivación por la derecha de g. a) prefijo viable b) mango c) elemento LR (0) d) elemento LR (1) PROBLEMAS /\*1.- Dibuje el arbol para la expresiones de tipo de la estructura de abajo : struct agregado { int m, n, o; }; typedef struct agregado Agregado; 2.- Cual es la expresion de tipo para : char \*g(char \*, char \*, char); 3.-Expresense, utilizando variables de tipos el tipo de la funcion ref que toma como argumento una variable de cualquier tipo y devuelve un apuntador a dicho objeto. \*/ Genere el codigo de 3 direcciones de 4.- a := b + c \* d; t1:= b t2:= c\*d a:= t1+t2 5.- x := y + z - ( u \* w ) t1:= y+z

t2:=u\*w x:=t1-t2 6.- a \* - (b+c ) y a:= ( b + c ) \* ( e + f ) t1=b+c t2=-t1 a:=t1\*t2 t1:= b+c t2:= e+f a:= t1\*t2 7.-(a \* b + h) - j \* k + 1 t1:= a\*b t2:= t1+h t3:= j\*k t4:= t3+1 t5:=t3-t4 8.- a > b + h or b == d R= t1:=b + h If a>t1 go to Ltrue goto L1 L1 if b==d goto Ltrue goto Lfalse

9.-Genere el codigo de 3 direcciones de las siguientes expresiones booleanas I) a < b or c < d If a< b or c < d or e < f If a< b or c < d or e < f or g < h If a

IV) a < b and c < d If a< b and c < d and e < f If a< b and c < d and e < f and g < h if a

goto Lfalse VIII) a < b or c < d and e < f If a< b and c < d or e < f If a y ) A; else B; x:=2

y:=1 L1: if x y) { A; } x:=2 y:=1 L1:= if x>y goto L2 goto Lfalse L2:= A ------------------- a = 0 while ( a <= 5) { a =a + 1 } A:= 0 L1: if a

a:=t1 gotoL1 --------------------------- for ( i = 0; i < 5; i=i+1){ A; } i:= 0 L1:= if i<= p + 2 ) { a := p[4+i\*2] ; } T1:= 2\*x A: t1+ 10 T2:= p+2 If a

If a=t2 goto L3 goto L4 L3: t3 t3:= i\*2 t4:= 4+t3 a:= p[t4] goto L1 --------------------------- a := a + 1; for ( i = 0; i < 5; i=i+1){ a=p[2\*i]; } i:= 0 L1: if i

goto Lfalse L2: stmt E3 goto L1 Lfalse … Generacion Codigo Objeto Ensamblador generico 12.-Si una instrucción de asignación de la forma x=y+z . Se traduce a: mov y, R0 add z, R0 mov R0 , x Como se traducen a = b + c; mov b.R0 add c,R0 mov R0,a d = a + c; mov a,R0

add c,R0 mov R0,d a = a + 1; mov a,R0 add #1,R0 mov R0,a 13.-Si una instrucción de asignación de la forma a=a+1 . Se traduce a: mov a , R0 add #1, R0 mov R0, a Como se traducen a = c + 2 mov c,R0 add #2,R0 mov R0,a d = a + 3 mov a,R0 add #3,R0 mov R0,a /\*En NASM (Netwide Assembler) se tienen las siguientes instrucciones para mover

mov destino, fuente saltos condicionales jl (<=), je (==), jge (>=), jg (>), jne (!=) 14.-Si una instrucción de la forma if ( m > 5 ) x = 6 ; else x = 8 ; Se traduce a: mov eax, dword [m] mov ebx, 5 cmp eax, ebx ; compar jg etq1 ; salta si mayor jmp etq2 ; salto incondicional etq1: mov eax , 6 jmp etq3 ;salto incondicional etq2: mov eax , 8 etq3:

mov eax, dword [x] Como se traduce la de abajo? if ( var1 <= var2 ) y = 99 ; else y = 22 ; 15.-Para el siguiente codigo genere el código de 3 direcciones y divida el codigo generado en bloques basicos w = 0; x = x + y; y=0; if( x > z ){ y = x; x++; } else { y = z; z++; } w = x + z;\*/