

LANGUAGE PROCESSORS

TRANSLATOR C TO ASSEMBLER

Authors

Juan Ruiz Bonald

Rafael Román Aguilar

INDEX

- GRAMMAR
- THINGS SUPPORTED
- DATA STRUCTURE
- EXAMPLES

GRAMMAR

HEAD

ENTRADA → INSTRUCCION ENTRADA | EPSILON
INSTRUCCION → FUNCION | ASIGNACION ';' | CONDICIONAL | INICIALIZACION ';'

BODY

FUNCION → TIPO ID '(' INICIALIZACION_CABECERA ')' '{' CONTENIDO '}'
CONDICIONAL → IF '(' OPERACIONLOG ')' '{' CONTENIDO '}' | IF '(' OPERACIONLOG ')' '{' CONTENIDO '}' ELSE '{' CONTENIDO '}' |
WHILE '(' OPERACIONLOG ')' '{' CONTENIDO '}'
CONTENIDO → ASIGNACION ';' CONTENIDO | INICIALIZACION ';' CONTENIDO | CONDICIONAL CONTENIDO | RETORNO ';' CONTENIDO |
LEER ';' CONTENIDO | IMPRIMIR ';' CONTENIDO | EPSILON
LEER → SCANF '(' STRING ';' '&' ID LEER_OPC ')'
LEER_OPC → ';' '&' ID LEER_OPC | EPSILON
IMPRIMIR → PRINTF '(' STRING IMPRIMIR_OPC ')'
IMPRIMIR_OPC → ';' '&' ID IMPRIMIR_OPC | ';' ID IMPRIMIR_OPC | EPSILON
RETORNO → RETURN DEVUELVE
DEVUELVE → OPERACION | EPSILON
ASIGNACION → ID ASSIGN OPERACION

OPERATIONS

OPERACION → OPERACIONARI | OPERACIONLOG
OPERACIONLOG → EXPRLOG
EXPRLOG → FACTOR AND EXPRLOG | FACTOR EQ EXPRLOG | FACTOR (EQ, LT, LE, GT, GE, NE, AND, OR, NOT) EXPRLOG | FACTOR
OPERACIONARI → SUM PLUS OPERACIONARI | SUM MINUS OPERACIONARI | SUM
SUM → EXPRARI | EXPRARI TIMES EXPRARI | EXPRARI DIVIDE EXPRARI
EXPRARI → '(' OPERACIONARI ')' | FACTOR
FACTOR → MINUS NUMBER | NUMBER | ID

INITIALIZATION

INICIALIZACION_CABECERA → TIPO ID RESTO_CABECERA | EPSILON
RESTO_CABECERA → ';' TIPO ID RESTO_CABECERA | EPSILON
INICIALIZACION → TIPO ID RESTO | TIPO ID ASIGNACION_INI RESTO
ASIGNACION_INI → ASSIGN OPERACION

RULES DERIVED TO TOKEN

TIPO → INT
RESTO → ';' ID RESTO | ';' ID ASIGNACION RESTO | EPSILON

THINGS SUPPORTED

- ARITHMETIC AND LOGICAL OPERATIONS
- VARIABLES (GLOBAL AND LOCAL)
- FUNCTIONS INT (WITH ANY PARAMETERS IN THE CALLING OF THE FUNCTION)
- ASSIGNMENT AND INITIALIZATION
- *CONDITIONAL FUNCTIONS (IF/ELSE AND WHILE), SCANF AND PRINTF*

DATA STRUCTURE

- **UP IMPLEMENTATION**
- WE CREATED A “TABLE” (IN PYTHON) TO SAVE THE VARIABLES AND WE DISTINCT THEM BY GLOBAL OR LOCAL. ALSO, AN ARRAY STORES THE STRINGS (ID AND POSITION IN STACK (OR IF IT IS GLOBAL, ITS ID))
- WE HAVE GLOBAL VARIABLES (IN PYTHON) FOR CONTROLLING THE INITIALIZATION OF GLOBAL AND LOCAL VARIABLES, THE NUMBER OF THE STACK AND THE PARAMETERS THAT RECEIVE THE FUNCTIONS
- WE HAVE USED SEVERAL NODES IN THE IMPLEMENTATION OF THE AST AS NODOFUNCION, NODOINIC, NODOVARIABLES AND NODOOPERACIONES.

EXAMPLE

```
INT A;  
  
INT MAIN()  
{  
    A = 1 && 0;  
    A = 4 + 1;  
    INT B;  
    B = A * 2;  
    RETURN B/A;  
}
```

```
.text  
.globl main  
.type main, @function  
main:  
    pushl %ebp  
    movl %esp, %ebp  
  
    cmp $1,$0  
    je AND  
    cmp $0,$0  
    je AND  
    movl $1,%eax  
    jmp F-AND  
AND:  
    movl $0,%eax  
F-AND:  
    movl %eax, a  
    movl $4, %eax  
    addl $1, %eax  
    movl %eax, a
```

```
    subl $4, %esp  
    movl $2, %eax  
    imull a, %eax  
    movl %eax, -4(%ebp)  
    movl -4(%ebp), %eax  
    cdq  
    movl a, %ecx  
    divl %ecx  
    movl %ecx, %eax  
  
    movl %ebp, %esp  
    popl %ebp  
    ret
```


EXAMPLE

```
#INCLUDE <STDIO.H>
```

```
INT A;
```

```
INT MAIN()
```

```
{
```

```
    IF(1 == 1)
```

```
    {
```

```
        WHILE(A!=1)
```

```
        {
```

```
            A = A + 1;
```

```
        }
```

```
    }
```

```
    RETURN 0;
```

```
}
```

```
.section .rodata.text  
.globl main  
.type main, @function  
main:
```

```
    pushl %ebp  
    movl %esp, %ebp
```

```
    cmpl $1,$1  
    je EQ  
    movl $0,%eax  
    jmp F-EQ
```

```
EQ:  
    movl $1,%eax
```

```
F-EQ:  
    cmpl $0, %eax  
    je final
```

```
start:  
    cmpl a,$1  
    jne NE  
    movl $0,%eax  
    jmp F-NE
```

```
NE:  
    movl $1,%eax
```

```
F-NE:
```

```
    cmpl $0, %eax  
    je final  
    movl $1, %eax  
    addl a, %eax  
    movl %eax, a  
    jmp start:
```

```
final:
```

```
final:
```

```
    movl $0, %eax
```

```
    movl %ebp, %esp  
    popl %ebp  
    ret
```

EXAMPLE

```
#INCLUDE <STDIO.H>

INT A;

INT MAIN()
{
    SCANF("RESULTADO: %i",&A);
    PRINTF("FINAL DEL EJEMPLO");
    RETURN 0;
}
```

```
.section .rodata
.LC0:
    .string "Resultado: %i"

.LC1:
    .string "Final del ejemplo"
.text
.globl main
.type main, @function
main:
    pushl %ebp
    movl %esp, %ebp

    pushl a
    pushl $s1
    call scanf
    addl $(8), %esp
    pushl $s2
    call printf
    addl $(4), %esp
    movl $0, %eax

    movl %ebp, %esp
    popl %ebp
    ret
```


THANKS FOR THE ATTENTION!