GRAMATICA ASCENDENTE

```
INIT:
        PROGRAM         {Return PROGRAM.val}
        ε               {Return Null}

PROGRAM:
        MAIN            { PROGRAM.val = program([MAIN.val])}
        MAIN LABELS     { PROGRAM.val = program([MAIN.val]+LABELS.val)}

MAIN:
        t_main t_dos_puntos INSTRUCTIONS     {MAIN.val = main(INSTRUCTIONS.val)}

LABELS:
        LABELS1 LABEL   {LABELS1.val.append(LABEL.val)
                        LABELS.val = LABELS1.val}
        LABEL           {LABELS.val = [LABEL.val]}

LABEL:
        t_label t_dos_puntos INSTRUCTIONS    {LABEL.val = label(INSTRUCTIONS.val)}

INSTRUCTIONS:
        INSTRUCTIONS1 INSTRUCTION   {INSTRUCTIONS1.val.append(INSTRUCTION.val)
                                    INSTRUCTIONS.val = INSTRUCTIONS1.val}
        INSTRUCTION                 {INSTRUCTIONS.val = [INSTRUCTION.val]}

INSTRUCTION:
        EXIT t_punto_coma           {INSTRUCTION.val = EXIT.val}
        GOTO t_punto_coma           {INSTRUCTION.val = GOTO.val}
        UNSET t_punto_coma          {INSTRUCTION.val = UNSET.val}
        PRINT t_punto_coma          {INSTRUCTION.val = PRINT.val}
        IF t_punto_coma             {INSTRUCTION.val = IF.val}
        SET t_punto_coma            {INSTRUCTION.val = SET.val}

EXIT:
        t_exit          {EXIT.val = exit()}

GOTO:
        t_goto t_label          {GOTO.val = label.lexval}

UNSET:
        t_unset t_par_izq VAR t_par_der         {UNSET.val = unset(VAR.val)}

PRINT:
        t_print t_par_izq VAR t_par_der         {PRINT.val = print(VAR.val)}

IF:
        t_if t_par_izq EXPRESSION t_par_der GOTO        {IF.val = if(EXPRESSION.val, GOTO.val)}

SET:
        VAR t_igual ASSIGNATION         {SET.val = set(VAR.val, ASSIGNATION.val)}

VAR:
        REGISTER                    {VAR.val = REGISTER.val}
        REGISTER POSITIONS          {VAR.val = array(REGISTER.val, POSITIONS.valww)}

REGISTER:
        t_temp          {REGISTER.val = register(t_temp.lexval)}
```

```
          t_params        {REGISTER.val = register(t_ params.lexval)}
          t_pila          {REGISTER.val = register(t_ pila.lexval)}
          t_return        {REGISTER.val = register(t_ return.lexval)}
          t_devuelto      {REGISTER.val = register(t_ devuelto.lexval)}
          t_puntero       {REGISTER.val = register(t_ puntero.lexval)}

POSITIONS:
          POSITIONS1 POSITION   {POSITIONS1.val.append(POSITION.val)
                                 POSITIONS.val = POSITIONS1.val}
          POSITION              {POSITIONS.val = [POSITION.val]}

POSITION:
          t_cor_izq CONT t_cor_der      {POSITION.val = index(CONT.val)}

CONDITION:
          EXPRESSION            {CONDITION.val = EXPRESSION.val}
          VAR                   {CONDITION.val = VAR.val}

PRIMARY:
          t_entero              {PRIMARY.val = primary(t_entero.lexval)}
          t_decimal             {PRIMARY.val = primary(t_ decimal.lexval)}
          t_cadena              {PRIMARY.val = primary(t_ cadena.lexval)}
          t_caracter            {PRIMARY.val = primary(t_ caracter.lexval)}

ASSIGNATION:
          DATA                  {ASSIGNATION.val = DATA.val}
          ARRAY                 {ASSIGNATION.val = ARRAY.val}
          READ                  {ASSIGNATION.val = READ.val}
          CAST                  {ASSIGNATION.val = CAST.val}
```

EXPRESSION	{ASSIGNATION.val = EXPRESSION.val}

DATA:

PRIMARY	{ DATA.val = PRIMARY.val}
VAR	{ DATA.val = VAR.val}

CONT:

PRIMARY	{ CONT.val = PRIMARY.val}
REGISTER	{ CONT.val = REGISTER.val}

ARRAY:

t_array t_par_izq t_par_der	{ARRAY.val = arrayDeclaration()}

READ:

t_read t_par_izq t_par_der	{READ.val = read()}

CAST:

t_par_izq TYPE t_par_der VAR	{CAST.val = cast(TYPE.val, VAR.val)}

TYPE:

t_float	{TYPE.val = t_float.lexval}
t_int	{TYPE.val = t_ int.lexval}
t_char	{TYPE.val = t_ char.lexval}

EXPRESSION:

ARITMETIC	{EXPRESSION.val = ARITMETIC.val}
LOGICAL	{EXPRESSION.val = LOGICAL.val}

BITXBIT	{EXPRESSION.val = BITXBIT.val}
RELATIONAL	{EXPRESSION.val = RELATIONAL.val}
POINTER	{EXPRESSION.val = POINTER.val}

ARITMETIC:

DATA t_suma DATA	{ARITMETIC.val = aritmetic(+, DATA1.val, DATA2.val)}
DATA t_resta DATA	{ARITMETIC.val = aritmetic(-, DATA1.val, DATA2.val)}
DATA t_mult DATA	{ARITMETIC.val = aritmetic(*, DATA1.val, DATA2.val)}
DATA t_div DATA	{ARITMETIC.val = aritmetic(/, DATA1.val, DATA2.val)}
DATA t_mod DATA	{ARITMETIC.val = aritmetic(%, DATA1.val, DATA2.val)}
t_abs t_par_izq DATA t_par_der	{ARITMETIC.val = aritmetic(abs, DATA1.val, Null)}
t_resta DATA	{ARITMETIC.val = aritmetic(minus, DATA1.val, Null)}

LOGICAL:

DATA1 t_and DATA2	{LOGICAL.val = logical(&&, DATA1.val, DATA2.val)}
DATA1 t_or DATA2	{LOGICAL.val = logical(||, DATA1.val, DATA2.val)}
DATA1 t_xor DATA2	{LOGICAL.val = logical(xor, DATA1.val, DATA2.val)}
t_not DATA	{LOGICAL.val = logical(!, DATA.val, Null)}

BITXBIT:

DATA1 t_and_bit DATA2	{BITXBIT.val = bitxbit(&, DATA1.val, DATA2.val)}
DATA1 t_or_bit DATA2	{BITXBIT.val = bitxbit(|, DATA1.val, DATA2.val)}
DATA1 t_xor_bit DATA2	{BITXBIT.val = bitxbit(^, DATA1.val, DATA2.val)}
DATA1 t_shift_der DATA2	{BITXBIT.val = bitxbit(>>, DATA1.val, DATA2.val)}
DATA1 t_shift_izq DATA2	{BITXBIT.val = bitxbit(<<, DATA1.val, DATA2.val)}
t_not_bit DATA	{BITXBIT.val = bitxbit(~, DATA.val, Null)}

RELATIONAL:

DATA1 t_es_igual DATA2	{RELATIONAL .val = relational(==, DATA1.val, DATA2.val)}
DATA1 t_no_igual DATA2	{RELATIONAL .val = relational(!=, DATA1.val, DATA2.val)}
DATA1 t_mayor DATA2	{RELATIONAL .val = relational(>, DATA1.val, DATA2.val)}
DATA1 t_menor DATA2	{RELATIONAL .val = relational(<, DATA1.val, DATA2.val)}
DATA1 t_mayor_igual DATA2	{RELATIONAL .val = relational(>=, DATA1.val, DATA2.val)}
DATA1 t_menor_igual DATA2	{RELATIONAL .val = relational(<=, DATA1.val, DATA2.val)}

POINTER:

t_and_bit VAR	{POINTER.val = pointer(VAR.val)}