β
```
-(void) endAssignment
{

        b = [sas pop];
        boff = 0;
        if ( b = (int) subListType )
                [self evalSublistB];
        a = [sas pop];  // E2's value
        aoff = 0;
        if ( a = (int) subListType )
                [self evalSublistA];

        [imCode genquad:(int) assign
                        arg1: b
                        arg1Offset: boff
                        arg2: 0
                        arg2Offset: 0
                        results: a
                        resultOffset: aoff ];
}
-(void) evaluation:(int) op
{
        aoff = 0;
        boff = 0;
        c = [myST gentemp];
        a = [sas pop];  // E2's value

        if ( a = (int) subListType )
                [self evalSublistA];

        b = [sas pop];  // E1's value
        if ( b =  (int) subListType )
                [self evalSublistB];


        [imCode genquad: op
                        arg1: b
                        arg1Offset:aoff
                        arg2: a
                        arg2Offset:boff
                        results: c
                        resultOffset: 0
                        ];
        [sas push: c];
}
```

X
```
-(void) endElseStatement
{
        b = [sas pop];
        [imCode codeAt:b
                results: [imCode nextQuad]];
}
```

μ
```
-(void) endWhenCondition
{
        a = [sas pop]; // Conditional result
        b = [sas pop]; // end of statement
        c = [sas pop]; // beginning of the statement (at the jump).
        [imCode genquad:(int) jtrue
                        arg1: a
                        arg2: 0
                        results: c + 1 ];
        [imCode codeAt:c
                        results: b +1 ];
        [imCode codeAt:b
                        results: [imCode nextQuad] ];
}
```

```
prog(x)
    x:array[10];
    max,i:integer;
γ {
    := max x[1];
    := i 2;
γ {

γ { := max x[i];} when < max x[i];
    := i + i 1;
    }^ when 9;
    } -> (max)
```

γ
```
-(void) beginStatement
{
        [sas push:[imCode nextQuad]];
        [imCode genquad:(int) jmp];
}
```

η
```
-(void) beginElseStatement
{
        a = [sas pop]; // Conditional result
        b = [sas pop]; // end of statement
        c = [sas pop]; // beginning of the statement (at the jump).
        [imCode genquad:(int) jtrue
                        arg1: a
                        arg2: 0
                        results: c + 1 ];
        [imCode codeAt:c
                        results: b +1 ];
        [sas push:b];
}
```

ι
```
-(void) endCarrotCondition
{
        c = [sas pop]; // Results of conditional
        b = [sas pop]; // Jump coordinates after statement
        a = [sas pop]; // Jump coordinates before statement


        [imCode codeAt:a
                        results: b +1 ];
        [imCode codeAt:b
                        results: [imCode nextQuad] ];
        [imCode genquad:(int) jtrue
                        arg1: c
                        arg2: 0
                        results: a + 1 ];
}
```

κ
```
-(void) endCarrotExpression
{
        c = [sas pop]; // Results of conditional
        b = [sas pop]; // Jump coordinates after statement
        a = [sas pop]; // Jump coordinates before statement


        d = [imCode nextQuad];
        [imCode genquad:(int) jmp
                        arg1: 0
                        arg2: 0
                        results: 0 ];
        [imCode genquad:(int) minusType
                        arg1: c
                        arg2: 1
                        results: c ];
        e = [myST gentemp];
        [imCode genquad:(int) lessEqType
                        arg1: c
                        arg2: 0
                        results: e ];
        [imCode genquad:(int) jtrue
                        arg1: e
                        arg2: 0
                        results: a + 1 ];
        [imCode codeAt:a
                        results: b + 1 ];
        [imCode codeAt: b
                        results: d + 1 ];
        [imCode codeAt: d
                        results: d + 2 ];
}
```