

---

ALBERT LUDWIGS UNIVERSITÄT FREIBURG

TECHNISCHE FAKULTÄT

# PicoC-Compiler

## Übersetzung einer Untermenge von C in den Befehlssatz der RETI-CPU

BACHELORARBEIT

*Abgabedatum:* 28<sup>th</sup> April 2022

*Author:*  
Jürgen Mattheis

*Gutachter:*  
Prof. Dr. Scholl

*Betreuung:*  
M.Sc. Seufert

---

Eine Bachelorarbeit am Lehrstuhl für  
Betriebssysteme

---

---

---

## **ERKLÄRUNG**

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

---

---

# Inhaltsverzeichnis

0.0.1	Umsetzung von Pointern . . . . .	9
0.0.1.1	Referenzierung . . . . .	9
0.0.1.2	Pointer Dereferenzierung durch Zugriff auf Arrayindex ersetzen . . . . .	11
0.0.2	Umsetzung von Arrays . . . . .	12
0.0.2.1	Initialisierung von Arrays . . . . .	12
0.0.2.2	Zugriff auf Arrayindex . . . . .	14
0.0.2.3	Zuweisung an Arrayindex . . . . .	17
0.0.3	Umsetzung von Structs . . . . .	19
0.0.3.1	Deklaration von Structs . . . . .	19
0.0.3.2	Initialisierung von Structs . . . . .	20
0.0.3.3	Zugriff auf Structattribut . . . . .	24
0.0.3.4	Zuweisung an Structattribut . . . . .	27
0.0.4	Umsetzung der Derived Datatypes im Zusammenspiel . . . . .	30
0.0.4.1	Einleitungsteil für Globale Statische Daten und Stackframe . . . . .	30
0.0.4.2	Mittelteil für die verschiedenen Derived Datatypes . . . . .	33
0.0.4.3	Schlusssteil für die verschiedenen Derived Datatypes . . . . .	36
0.0.5	Umsetzung von Funktionen . . . . .	43
0.0.5.1	Funktionen auflösen zu RETI Code . . . . .	43
0.0.5.1.1	Sprung zur Main Funktion . . . . .	45
0.0.5.2	Funktionsdeklaration und -definition . . . . .	48
0.0.5.3	Funktionsaufruf . . . . .	49
0.0.5.3.1	Ohne Rückgabewert . . . . .	49
0.0.5.3.2	Mit Rückgabewert . . . . .	51
0.0.5.3.3	Umsetzung von Call by Sharing für Arrays . . . . .	54
0.0.5.3.4	Umsetzung von Call by Value für Structs . . . . .	57
0.0.6	Umsetzung kleinerer Details . . . . .	59
0.1	Fehlermeldungen . . . . .	59
0.1.1	Error Handler . . . . .	59
0.1.2	Arten von Fehlermeldungen . . . . .	59
0.1.2.1	Syntaxfehler . . . . .	59
0.1.2.2	Laufzeitfehler . . . . .	59

---

---

# Abbildungsverzeichnis

---

---

# Codeverzeichnis

1	PicoC Code für Pointer Referenzierung . . . . .	9
2	Abstract Syntax Tree für Pointer Referenzierung . . . . .	9
3	PicoC Mon Pass für Pointer Referenzierung . . . . .	10
4	RETI Blocks Pass für Pointer Referenzierung . . . . .	11
5	PicoC Code für Pointer Dereferenzierung . . . . .	11
6	Abstract Syntax Tree für Pointer Dereferenzierung . . . . .	11
7	PicoC Shrink Pass für Pointer Dereferenzierung . . . . .	12
8	PicoC Code für Array Initialisierung . . . . .	12
9	Abstract Syntax Tree für Array Initialisierung . . . . .	13
10	Symboltabelle für Array Initialisierung . . . . .	13
11	PicoC Mon Pass für Array Initialisierung . . . . .	14
12	RETI Blocks Pass für Array Initialisierung . . . . .	14
13	PicoC Code für Zugriff auf Arrayindex . . . . .	15
14	Abstract Syntax Tree für Zugriff auf Arrayindex . . . . .	15
15	PicoC Mon Pass für Zugriff auf Arrayindex . . . . .	16
16	RETI Blocks Pass für Zugriff auf Arrayindex . . . . .	17
17	PicoC Code für Zuweisung an Arrayindex . . . . .	17
18	Abstract Syntax Tree für Zuweisung an Arrayindex . . . . .	18
19	PicoC Mon Pass für Zuweisung an Arrayindex . . . . .	18
20	RETI Blocks Pass für Zuweisung an Arrayindex . . . . .	19
21	PicoC Code für Deklaration von Structs . . . . .	19
22	Symboltabelle für Deklaration von Structs . . . . .	20
23	PicoC Code für Initialisierung von Structs . . . . .	20
24	Abstract Syntax Tree für Initialisierung von Structs . . . . .	22
25	Symboltabelle für Initialisierung von Structs . . . . .	23
26	PicoC Mon Pass für Initialisierung von Structs . . . . .	24
27	RETI Blocks Pass für Initialisierung von Structs . . . . .	24
28	PicoC Code für Zugriff auf Structattribut . . . . .	24
29	Abstract Syntax Tree für Zugriff auf Structattribut . . . . .	25
30	PicoC Mon Pass für Zugriff auf Structattribut . . . . .	26
31	RETI Blocks Pass für Zugriff auf Structattribut . . . . .	27
32	PicoC Code für Zuweisung an Structattribut . . . . .	27
33	Abstract Syntax Tree für Zuweisung an Structattribut . . . . .	28
34	PicoC Mon Pass für Zuweisung an Structattribut . . . . .	28
35	RETI Blocks Pass für Zuweisung an Structattribut . . . . .	29
36	PicoC Code für den Einleitungsteil . . . . .	30
37	Abstract Syntax Tree für den Einleitungsteil . . . . .	31
38	PicoC Mon Pass für den Einleitungsteil . . . . .	32
39	RETI Blocks Pass für den Einleitungsteil . . . . .	32
40	PicoC Code für den Mittelteil . . . . .	33
41	Abstract Syntax Tree für den Mittelteil . . . . .	34
42	PicoC Mon Pass für den Mittelteil . . . . .	35
43	RETI Blocks Pass für den Mittelteil . . . . .	36
44	PicoC Code für den Schlussteil . . . . .	36
45	Abstract Syntax Tree für den Schlussteil . . . . .	38
46	PicoC Mon Pass für den Schlussteil . . . . .	40
47	RETI Blocks Pass für den Schlussteil . . . . .	42

48	PicoC Code für 3 Funktionen . . . . .	43
49	Abstract Syntax Tree für 3 Funktionen . . . . .	43
50	PicoC Blocks Pass für 3 Funktionen . . . . .	44
51	PicoC Mon Pass für 3 Funktionen . . . . .	45
52	RETI Blocks Pass für 3 Funktionen . . . . .	45
53	PicoC Code für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . . . .	46
54	PicoC Mon Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . . . .	46
55	PicoC Blocks Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . . . .	47
56	PicoC Patch Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . . . .	47
57	PicoC Code für Funktionen, wobei eine Funktion vorher deklariert werden muss . . . . .	48
58	Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss . . . . .	49
59	PicoC Code für Funktionsaufruf ohne Rückgabewert . . . . .	49
60	PicoC Mon Pass für Funktionsaufruf ohne Rückgabewert . . . . .	50
61	RETI Blocks Pass für Funktionsaufruf ohne Rückgabewert . . . . .	51
62	RETI Pass für Funktionsaufruf ohne Rückgabewert . . . . .	51
63	PicoC Code für Funktionsaufruf mit Rückgabewert . . . . .	51
64	PicoC Mon Pass für Funktionsaufruf mit Rückgabewert . . . . .	52
65	RETI Blocks Pass für Funktionsaufruf mit Rückgabewert . . . . .	53
66	RETI Pass für Funktionsaufruf mit Rückgabewert . . . . .	54
67	PicoC Code für Call by Sharing für Arrays . . . . .	54
68	PicoC Mon Pass für Call by Sharing für Arrays . . . . .	55
69	Symboltabelle für Call by Sharing für Arrays . . . . .	56
70	RETI Block Pass für Call by Sharing für Arrays . . . . .	57
71	PicoC Code für Call by Value für Structs . . . . .	57
72	PicoC Mon Pass für Call by Value für Structs . . . . .	58
73	RETI Block Pass für Call by Value für Structs . . . . .	59

---

---

# Tabellenverzeichnis

---

---

# Definitionsverzeichnis



---

---

# Grammatikverzeichnis

## 0.0.1 Umsetzung von Pointern

### 0.0.1.1 Referenzierung

Die **Referenzierung** `&var` wird im Folgenden anhand des Beispiels in Code 1 erklärt.

```
1 void main() {
2     int var = 42;
3     int *pntr = &var;
4 }
```

Code 1: PicoC Code für Pointer Referenzierung

Der Node `Ref(Name('var'))` repräsentiert in der **Abstrakten Syntax** in Code 2 eine **Referenzierung** `&var`.

```
1 File
2   Name './example_pntr_ref.ast',
3   [
4     FunDef
5       VoidType 'void',
6       Name 'main',
7       [],
8       [
9         Assign
10          Alloc
11            Writeable,
12            IntType 'int',
13            Name 'var',
14            Num '42',
15          Assign
16            Alloc
17              Writeable,
18              PtrDecl
19                Num '1',
20                IntType 'int',
21                Name 'pntr',
22              Ref
23                Name 'var'
24            ]
25          ]
26   ]
```

Code 2: Abstract Syntax Tree für Pointer Referenzierung

Im **PicoC-Mon Pass** in Code 3 wird der Node `Ref(Name('var'))` durch die Nodes `Ref(GlobalRead(Num('0')))` und `Assign(GlobalWrite(Num('1')), Tmp(Num('1')))` ersetzt. Im Fall, dass in `Ref(exp)` das `exp` vielleicht nicht direkt ein `Name('var')` enthält und `exp` vielleicht ein `Subscr(Attr(Name('var')))` ist, sind noch weitere Anweisungen zwischen Zeile 14 und 15 nötig, die sich in diesem Beispiel um das Übersetzen von `Subscr(exp)` und `Attr(exp)` kümmern. Die Vorgehen hierfür ist in Subkapitel 0.0.4.2 erklärt.

```

1 File
2   Name './example_pntr_ref.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         // Assign(Name('var'), Num('42')),
8         Exp
9           Num '42',
10        Assign
11          Global
12            Num '0',
13          Stack
14            Num '1',
15        // Assign(Name('pntr'), Ref(Name('var'))),
16        Ref
17          Global
18            Num '0',
19        Assign
20          Global
21            Num '1',
22          Stack
23            Num '1',
24        Return
25          Empty
26      ]
27  ]

```

Code 3: PicoC Mon Pass für Pointer Referenzierung

Im **PicoC-Blocks Pass** in Code 4 wird der die Nodes

```

1 File
2   Name './example_pntr_ref.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         # // Assign(Name('var'), Num('42')),
8         # Exp(Num('42')),
9         SUBI SP 1,
10        LOADI ACC 42,
11        STOREIN SP ACC 1,
12        # Assign(Global(Num('0')), Stack(Num('1'))),
13        LOADIN SP ACC 1,
14        STOREIN DS ACC 0,
15        ADDI SP 1,
16        # // Assign(Name('pntr'), Ref(Name('var'))),
17        # Ref(Global(Num('0'))),
18        SUBI SP 1,
19        LOADI IN1 0,
20        ADD IN1 DS,
21        STOREIN SP IN1 1,
22        # Assign(Global(Num('1')), Stack(Num('1'))),

```

```

23     LOADIN SP ACC 1,
24     STOREIN DS ACC 1,
25     ADDI SP 1,
26     # Return(Empty()),
27     LOADIN BAF PC -1
28 ]
29 ]

```

Code 4: RETI Blocks Pass für Pointer Referenzierung

### 0.0.1.2 Pointer Dereferenzierung durch Zugriff auf Arrayindex ersetzen

```

1 void main() {
2     int var = 42;
3     int *pntr = &var;
4     *pntr;
5 }

```

Code 5: PicoC Code für Pointer Dereferenzierung

```

1 File
2   Name './example_pntr_deref.ast',
3   [
4     FunDef
5       VoidType 'void',
6       Name 'main',
7       [],
8       [
9         Assign
10          Alloc
11            Writeable,
12            IntType 'int',
13            Name 'var',
14            Num '42',
15          Assign
16            Alloc
17              Writeable,
18              PntrDecl
19                Num '1',
20                IntType 'int',
21                Name 'pntr',
22              Ref
23                Name 'var',
24            Exp
25              Deref
26                Name 'pntr',
27                Num '0'
28          ]
29 ]

```

Code 6: Abstract Syntax Tree für Pointer Dereferenzierung

```
1 File
2   Name './example_pntr_deref.picoc_shrink',
3   [
4     FunDef
5       VoidType 'void',
6       Name 'main',
7       [],
8       [
9         Assign
10          Alloc
11            Writeable,
12            IntType 'int',
13            Name 'var',
14            Num '42',
15          Assign
16            Alloc
17              Writeable,
18              PntrDecl
19                Num '1',
20                IntType 'int',
21                Name 'pntr',
22              Ref
23                Name 'var',
24            Exp
25              Subscr
26                Name 'pntr',
27                Num '0'
28          ]
29   ]
```

Code 7: PicoC Shrink Pass für Pointer Dereferenzierung

## 0.0.2 Umsetzung von Arrays

### 0.0.2.1 Initialisierung von Arrays

```
1 void main() {
2   int ar[2][1] = {{4}, {2}};
3 }
```

Code 8: PicoC Code für Array Initialisierung

```
1 File
2   Name './example_array_init.ast',
3   [
4     FunDef
5       VoidType 'void',
6       Name 'main',
7       [],
```

```

8      [
9      Assign
10     Alloc
11     Writeable,
12     ArrayDecl
13     [
14     Num '2',
15     Num '1'
16     ],
17     IntType 'int',
18     Name 'ar',
19     Array
20     [
21     Array
22     [
23     Num '4'
24     ],
25     Array
26     [
27     Num '2'
28     ]
29     ]
30 ]
31 ]

```

Code 9: Abstract Syntax Tree für Array Initialisierung

```

1 SymbolTable
2 [
3   Symbol(
4     {
5       type qualifier:      Empty()
6       datatype:            FunDecl(VoidType('void'), Name('main'), [])
7       name:                Name('main')
8       value or address:    Empty()
9       position:            Pos(Num('1'), Num('5'))
10      size:                 Empty()
11    },
12   Symbol(
13     {
14       type qualifier:      Writeable()
15       datatype:            ArrayDecl([Num('2'), Num('1')], IntType('int'))
16       name:                Name('ar@main')
17       value or address:    Num('0')
18       position:            Pos(Num('2'), Num('6'))
19       size:                 Num('2')
20     }
21 ]

```

Code 10: Symboltabelle für Array Initialisierung

```

1 File
2   Name './example_array_init.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         // Assign(Name('ar'), Array([Array([Num('4')]), Array([Num('2')])])),
8         Exp
9           Num '4',
10        Exp
11          Num '2',
12        Assign
13          Global
14            Num '0',
15          Stack
16            Num '2',
17        Return
18          Empty
19      ]
20  ]

```

Code 11: PicoC Mon Pass für Array Initialisierung

```

1 File
2   Name './example_array_init.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         # // Assign(Name('ar'), Array([Array([Num('4')]), Array([Num('2')])])),
8         # Exp(Num('4')),
9         SUBI SP 1,
10        LOADI ACC 4,
11        STOREIN SP ACC 1,
12        # Exp(Num('2')),
13        SUBI SP 1,
14        LOADI ACC 2,
15        STOREIN SP ACC 1,
16        # Assign(Global(Num('0')), Stack(Num('2'))),
17        LOADIN SP ACC 1,
18        STOREIN DS ACC 1,
19        LOADIN SP ACC 2,
20        STOREIN DS ACC 0,
21        ADDI SP 2,
22        # Return(Empty()),
23        LOADIN BAF PC -1
24      ]
25  ]

```

Code 12: RETI Blocks Pass für Array Initialisierung

### 0.0.2.2 Zugriff auf Arrayindex

Der Zugriff auf einen bestimmten Index eines Arrays ist wie folgt umgesetzt:

```
1 void main() {  
2   int ar[2] = {1, 2};  
3   ar[2];  
4 }
```

Code 13: PicoC Code für Zugriff auf Arrayindex

```
1 File  
2   Name './example_array_access.ast',  
3   [  
4     FunDef  
5       VoidType 'void',  
6       Name 'main',  
7       [],  
8       [  
9         Assign  
10          Alloc  
11            Writeable,  
12            ArrayDecl  
13              [  
14                Num '2'  
15              ],  
16              IntType 'int',  
17              Name 'ar',  
18              Array  
19                [  
20                  Num '1',  
21                  Num '2'  
22                ],  
23              Exp  
24                Subscr  
25                  Name 'ar',  
26                  Num '2'  
27              ]  
28    ]
```

Code 14: Abstract Syntax Tree für Zugriff auf Arrayindex

```
1 File  
2   Name './example_array_access.picoc_mon',  
3   [  
4     Block  
5       Name 'main.0',  
6       [  
7         // Assign(Name('ar'), Array([Num('1'), Num('2')])),  
8         Exp  
9           Num '1',  
10        Exp  
11        Num '2',  
12      ]  
13    ]
```



```

12     Assign
13     Global
14     Num '0',
15     Stack
16     Num '2',
17     // Exp(Subscr(Name('ar'), Num('2'))),
18     Ref
19     Global
20     Num '0',
21     Exp
22     Num '2',
23     Ref
24     Subscr
25     Stack
26     Num '2',
27     Stack
28     Num '1',
29     Exp
30     Stack
31     Num '1',
32     Return
33     Empty
34 ]
35 ]

```

Code 15: PicoC Mon Pass für Zugriff auf Arrayindex

```

1 File
2 Name './example_array_access.reti_blocks',
3 [
4     Block
5     Name 'main.0',
6     [
7         # // Assign(Name('ar'), Array([Num('1'), Num('2')])),
8         # Exp(Num('1')),
9         SUBI SP 1,
10        LOADI ACC 1,
11        STOREIN SP ACC 1,
12        # Exp(Num('2')),
13        SUBI SP 1,
14        LOADI ACC 2,
15        STOREIN SP ACC 1,
16        # Assign(Global(Num('0')), Stack(Num('2'))),
17        LOADIN SP ACC 1,
18        STOREIN DS ACC 1,
19        LOADIN SP ACC 2,
20        STOREIN DS ACC 0,
21        ADDI SP 2,
22        # // Exp(Subscr(Name('ar'), Num('2'))),
23        # Ref(Global(Num('0'))),
24        SUBI SP 1,
25        LOADI IN1 0,
26        ADD IN1 DS,
27        STOREIN SP IN1 1,

```

```

28     # Exp(Num('2')),
29     SUBI SP 1,
30     LOADI ACC 2,
31     STOREIN SP ACC 1,
32     # Ref(Subscr(Stack(Num('2')), Stack(Num('1')))),
33     LOADIN SP IN1 2,
34     LOADIN SP IN2 1,
35     MULTI IN2 1,
36     ADD IN1 IN2,
37     ADDI SP 1,
38     STOREIN SP IN1 1,
39     LOADIN SP IN1 1,
40     LOADIN IN1 ACC 0,
41     STOREIN SP ACC 1,
42     # Return(Empty()),
43     LOADIN BAF PC -1
44 ]
45 ]

```

Code 16: RETI Blocks Pass für Zugriff auf Arrayindex

### 0.0.2.3 Zuweisung an Arrayindex

```

1 void main() {
2     int ar[2];
3     ar[2] = 42;
4 }

```

Code 17: PicoC Code für Zuweisung an Arrayindex

```

1 File
2   Name './example_array_assignment.ast',
3   [
4     FunDef
5       VoidType 'void',
6       Name 'main',
7       [],
8       [
9         Exp
10          Alloc
11            Writeable,
12            ArrayDecl
13              [
14                Num '2'
15              ],
16            IntType 'int',
17            Name 'ar',
18          Assign
19            Subscr
20              Name 'ar',
21              Num '2',
22              Num '42'

```

```

23   ]
24 ]

```

Code 18: Abstract Syntax Tree für Zuweisung an Arrayindex

```

1 File
2   Name './example_array_assignment.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         // Assign(Subscr(Name('ar'), Num('2')), Num('42')),
8         Exp
9           Num '42',
10        Ref
11          Global
12            Num '0',
13        Exp
14          Num '2',
15        Ref
16          Subscr
17            Stack
18              Num '2',
19            Stack
20              Num '1',
21        Assign
22          Stack
23            Num '1',
24          Stack
25            Num '2',
26        Return
27          Empty
28      ]
29 ]

```

Code 19: PicoC Mon Pass für Zuweisung an Arrayindex

```

1 File
2   Name './example_array_assignment.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         # // Assign(Subscr(Name('ar'), Num('2')), Num('42')),
8         # Exp(Num('42')),
9         SUBI SP 1,
10        LOADI ACC 42,
11        STOREIN SP ACC 1,
12        # Ref(Global(Num('0'))),
13        SUBI SP 1,
14        LOADI IN1 0,

```

```

15      ADD IN1 DS,
16      STOREIN SP IN1 1,
17      # Exp(Num('2')),
18      SUBI SP 1,
19      LOADI ACC 2,
20      STOREIN SP ACC 1,
21      # Ref(Subscr(Stack(Num('2')), Stack(Num('1')))),
22      LOADIN SP IN1 2,
23      LOADIN SP IN2 1,
24      MULTI IN2 1,
25      ADD IN1 IN2,
26      ADDI SP 1,
27      STOREIN SP IN1 1,
28      LOADIN SP IN1 1,
29      LOADIN SP ACC 2,
30      ADDI SP 2,
31      STOREIN IN1 ACC 0,
32      # Return(Empty()),
33      LOADIN BAF PC -1
34  ]
35 ]

```

Code 20: RETI Blocks Pass für Zuweisung an Arrayindex

### 0.0.3 Umsetzung von Structs

#### 0.0.3.1 Deklaration von Structs

```

1 struct st1 {int *ar[3];};
2
3 struct st2 {struct st1 st;};
4
5 void main() {
6 }

```

Code 21: PicoC Code für Deklaration von Structs

```

1 SymbolTable
2 [
3   Symbol(
4     {
5       type qualifier:      Empty()
6       datatype:            ArrayDecl([Num('3')], PtrDecl(Num('1'), IntType('int')))
7       name:                Name('ar@st1')
8       value or address:    Empty()
9       position:            Pos(Num('1'), Num('17'))
10      size:                 Num('3')
11    },
12    Symbol(
13      {
14        type qualifier:      Empty()

```

```

15     datatype:      StructDecl(Name('st1'), [Alloc(Writable(),
16       ↪ ArrayDecl([Num('3')], PtrDecl(Num('1'), IntType('int'))), Name('ar'))])
17     name:          Name('st1')
18     value or address: [Name('ar@st1')]
19     position:      Pos(Num('1'), Num('7'))
20     size:          Num('3')
21   },
22   Symbol(
23     {
24       type qualifier: Empty()
25       datatype:      StructSpec(Name('st1'))
26       name:          Name('st@st2')
27       value or address: Empty()
28       position:      Pos(Num('3'), Num('23'))
29       size:          Num('3')
30     },
31     Symbol(
32       {
33         type qualifier: Empty()
34         datatype:      StructDecl(Name('st2'), [Alloc(Writable(),
35           ↪ StructSpec(Name('st1')), Name('st'))])
36         name:          Name('st2')
37         value or address: [Name('st@st2')]
38         position:      Pos(Num('3'), Num('7'))
39         size:          Num('3')
40       },
41       Symbol(
42         {
43           type qualifier: Empty()
44           datatype:      FunDecl(VoidType('void'), Name('main'), [])
45           name:          Name('main')
46           value or address: Empty()
47           position:      Pos(Num('5'), Num('5'))
48           size:          Empty()
49         }
50       )
51     )
52   ]

```

Code 22: Symboltabelle für Deklaration von Structs

### 0.0.3.2 Initialisierung von Structs

```

1 struct st1 {int *ptr[1];};
2
3 struct st2 {struct st1 st;};
4
5 void main() {
6     int var = 42;
7     struct st1 st = {.st={.ptr={{&var}}}};
8 }

```

Code 23: PicoC Code für Initialisierung von Structs

```
1 File
2   Name './example_struct_init.ast',
3   [
4     StructDecl
5       Name 'st1',
6       [
7         Alloc
8           Writeable,
9           ArrayDecl
10            [
11              Num '1'
12            ],
13            PtrDecl
14              Num '1',
15              IntType 'int',
16            Name 'pntr'
17          ],
18      StructDecl
19        Name 'st2',
20        [
21          Alloc
22            Writeable,
23            StructSpec
24              Name 'st1',
25              Name 'st'
26          ],
27      FunDef
28        VoidType 'void',
29        Name 'main',
30        [],
31        [
32          Assign
33            Alloc
34              Writeable,
35              IntType 'int',
36              Name 'var',
37              Num '42',
38          Assign
39            Alloc
40              Writeable,
41              StructSpec
42                Name 'st1',
43                Name 'st',
44              Struct
45                [
46                  Assign
47                    Name 'st',
48                    Struct
49                      [
50                        Assign
51                          Name 'pntr',
52                          Array
53                            [
54                              Array
55                                [
56                                  Ref
57                                    Name 'var'
```

```

58         ]
59     ]
60 ]
61 ]
62 ]
63 ]

```

Code 24: Abstract Syntax Tree für Initialisierung von Structs

```

1 SymbolTable
2 [
3     Symbol(
4         {
5             type qualifier:      Empty()
6             datatype:            ArrayDecl([Num('1')], PtrDecl(Num('1'), IntType('int')))
7             name:                 Name('ptr@st1')
8             value or address:     Empty()
9             position:             Pos(Num('1'), Num('17'))
10            size:                 Num('1')
11        },
12     Symbol(
13         {
14             type qualifier:      Empty()
15             datatype:            StructDecl(Name('st1'), [Alloc(Writeable(),
16                 ⇨ ArrayDecl([Num('1')], PtrDecl(Num('1'), IntType('int'))), Name('ptr'))])
17             name:                 Name('st1')
18             value or address:     [Name('ptr@st1')]
19             position:             Pos(Num('1'), Num('7'))
20             size:                 Num('1')
21        },
22     Symbol(
23         {
24             type qualifier:      Empty()
25             datatype:            StructSpec(Name('st1'))
26             name:                 Name('st@st2')
27             value or address:     Empty()
28             position:             Pos(Num('3'), Num('23'))
29             size:                 Num('1')
30        },
31     Symbol(
32         {
33             type qualifier:      Empty()
34             datatype:            StructDecl(Name('st2'), [Alloc(Writeable(),
35                 ⇨ StructSpec(Name('st1')), Name('st'))])
36             name:                 Name('st2')
37             value or address:     [Name('st@st2')]
38             position:             Pos(Num('3'), Num('7'))
39             size:                 Num('1')
40        },
41     Symbol(
42         {
43             type qualifier:      Empty()
44             datatype:            FunDecl(VoidType('void'), Name('main'), [])
45             name:                 Name('main')

```

```

44     value or address:    Empty()
45     position:           Pos(Num('5'), Num('5'))
46     size:               Empty()
47 },
48 Symbol(
49 {
50     type qualifier:      Writeable()
51     datatype:            IntType('int')
52     name:                Name('var@main')
53     value or address:    Num('0')
54     position:            Pos(Num('6'), Num('6'))
55     size:                Num('1')
56 },
57 Symbol(
58 {
59     type qualifier:      Writeable()
60     datatype:            StructSpec(Name('st1'))
61     name:                Name('st@main')
62     value or address:    Num('1')
63     position:            Pos(Num('7'), Num('13'))
64     size:                Num('1')
65 }
66 ]

```

Code 25: Symboltabelle für Initialisierung von Structs

```

1 File
2   Name './example_struct_init.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         // Assign(Name('var'), Num('42')),
8         Exp
9           Num '42',
10        Assign
11          Global
12            Num '0',
13          Stack
14            Num '1',
15        // Assign(Name('st'), Struct([Assign(Name('st'), Struct([Assign(Name('pntr'),
16        ↪   Array([Array([Ref(Name('var'))]))]))]),
17        Ref
18          Global
19            Num '0',
20        Assign
21          Global
22            Num '1',
23          Stack
24            Num '1',
25        Return
26          Empty
27      ]
28  ]

```



Code 26: PicoC Mon Pass für Initialisierung von Structs

```

1 File
2   Name './example_struct_init.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         # // Assign(Name('var'), Num('42')),
8         # Exp(Num('42')),
9         SUBI SP 1,
10        LOADI ACC 42,
11        STOREIN SP ACC 1,
12        # Assign(Global(Num('0')), Stack(Num('1'))),
13        LOADIN SP ACC 1,
14        STOREIN DS ACC 0,
15        ADDI SP 1,
16        # // Assign(Name('st'), Struct([Assign(Name('st')), Struct([Assign(Name('pntr')),
17        ↪   Array([Array([Ref(Name('var'))])])])]),
18        # Ref(Global(Num('0'))),
19        SUBI SP 1,
20        LOADI IN1 0,
21        ADD IN1 DS,
22        STOREIN SP IN1 1,
23        # Assign(Global(Num('1')), Stack(Num('1'))),
24        LOADIN SP ACC 1,
25        STOREIN DS ACC 1,
26        ADDI SP 1,
27        # Return(Empty()),
28        LOADIN BAF PC -1
29      ]
30    ]

```

Code 27: RETI Blocks Pass für Initialisierung von Structs

### 0.0.3.3 Zugriff auf Structattribut

```

1 struct pos {int x; int y;};
2
3 void main() {
4   struct pos st = {.x=4, .y=2};
5   st.y;
6 }

```

Code 28: PicoC Code für Zugriff auf Structattribut

```

1 File
2   Name './example_struct_attr_access.ast',
3   [

```

```

4   StructDecl
5     Name 'pos',
6     [
7       Alloc
8         Writeable,
9         IntType 'int',
10        Name 'x',
11      Alloc
12        Writeable,
13        IntType 'int',
14        Name 'y'
15    ],
16  FunDef
17    VoidType 'void',
18    Name 'main',
19    [],
20    [
21      Assign
22        Alloc
23          Writeable,
24          StructSpec
25            Name 'pos',
26            Name 'st',
27          Struct
28            [
29              Assign
30                Name 'x',
31                Num '4',
32              Assign
33                Name 'y',
34                Num '2'
35            ],
36      Exp
37        Attr
38          Name 'st',
39          Name 'y'
40    ]
41 ]

```

Code 29: Abstract Syntax Tree für Zugriff auf Structattribut

```

1 File
2   Name './example_struct_attr_access.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         // Assign(Name('st'), Struct([Assign(Name('x'), Num('4')), Assign(Name('y'),
8         ↪ Num('2'))])),
9       Exp
10        Num '4',
11      Exp
12        Num '2',
13      Assign

```

```

13     Global
14         Num '0',
15     Stack
16         Num '2',
17     Ref
18     Global
19         Num '0',
20     Ref
21     Attr
22     Stack
23         Num '1',
24     Name 'y',
25     Exp
26     Stack
27         Num '1',
28     Return
29     Empty
30 ]
31 ]

```

Code 30: PicoC Mon Pass für Zugriff auf Structattribut

```

1 File
2   Name './example_struct_attr_access.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         # // Assign(Name('st'), Struct([Assign(Name('x'), Num('4')), Assign(Name('y'),
8         ↪   Num('2'))])),
9         # Exp(Num('4')),
10        SUBI SP 1,
11        LOADI ACC 4,
12        STOREIN SP ACC 1,
13        # Exp(Num('2')),
14        SUBI SP 1,
15        LOADI ACC 2,
16        STOREIN SP ACC 1,
17        # Assign(Global(Num('0')), Stack(Num('2'))),
18        LOADIN SP ACC 1,
19        STOREIN DS ACC 1,
20        LOADIN SP ACC 2,
21        STOREIN DS ACC 0,
22        ADDI SP 2,
23        # Ref(Global(Num('0'))),
24        SUBI SP 1,
25        LOADI IN1 0,
26        ADD IN1 DS,
27        STOREIN SP IN1 1,
28        # Ref(Attr(Stack(Num('1')), Name('y'))),
29        LOADIN SP IN1 1,
30        ADDI IN1 1,
31        STOREIN SP IN1 1,
32        LOADIN SP IN1 1,

```

```

32     LOADIN IN1 ACC 0,
33     STOREIN SP ACC 1,
34     # Return(Empty()),
35     LOADIN BAF PC -1
36 ]
37 ]

```

Code 31: RETI Blocks Pass für Zugriff auf Structattribut

#### 0.0.3.4 Zuweisung an Structattribut

```

1 struct pos {int x; int y;};
2
3 void main() {
4     struct pos st = {.x=4, .y=2};
5     st.y = 42;
6 }

```

Code 32: PicoC Code für Zuweisung an Structattribut

```

1 File
2   Name './example_struct_attr_assignment.ast',
3   [
4     StructDecl
5       Name 'pos',
6       [
7         Alloc
8           Writeable,
9           IntType 'int',
10          Name 'x',
11         Alloc
12           Writeable,
13           IntType 'int',
14           Name 'y'
15       ],
16     FunDef
17       VoidType 'void',
18       Name 'main',
19       [],
20       [
21         Assign
22           Alloc
23             Writeable,
24             StructSpec
25               Name 'pos',
26               Name 'st',
27             Struct
28               [
29                 Assign
30                   Name 'x',
31                   Num '4',
32                 Assign

```

```

33         Name 'y',
34         Num '2'
35     ],
36     Assign
37     Attr
38     Name 'st',
39     Name 'y',
40     Num '42'
41 ]
42 ]

```

Code 33: Abstract Syntax Tree für Zuweisung an Structattribut

```

1 File
2   Name './example_struct_attr_assignment.picoc_mon',
3   [
4     Block
5     Name 'main.0',
6     [
7       // Assign(Name('st'), Struct([Assign(Name('x'), Num('4')), Assign(Name('y'),
8       ↪ Num('2'))])),
9       Exp
10      Num '4',
11      Exp
12      Num '2',
13      Assign
14      Global
15      Num '0',
16      Stack
17      Num '2',
18      // Assign(Attr(Name('st'), Name('y')), Num('42')),
19      Exp
20      Num '42',
21      Ref
22      Global
23      Num '0',
24      Ref
25      Attr
26      Stack
27      Num '1',
28      Name 'y',
29      Assign
30      Stack
31      Num '1',
32      Stack
33      Num '2',
34      Return
35      Empty
36    ]
37  ]

```

Code 34: PicoC Mon Pass für Zuweisung an Structattribut

```

1 File
2   Name './example_struct_attr_assignment.reti_blocks',
3   [
4     Block
5     Name 'main.O',
6     [
7       # // Assign(Name('st'), Struct([Assign(Name('x'), Num('4')), Assign(Name('y'),
8         ↪ Num('2'))])),
9       # Exp(Num('4')),
10      SUBI SP 1,
11      LOADI ACC 4,
12      STOREIN SP ACC 1,
13      # Exp(Num('2')),
14      SUBI SP 1,
15      LOADI ACC 2,
16      STOREIN SP ACC 1,
17      # Assign(Global(Num('0')), Stack(Num('2'))),
18      LOADIN SP ACC 1,
19      STOREIN DS ACC 1,
20      LOADIN SP ACC 2,
21      STOREIN DS ACC 0,
22      ADDI SP 2,
23      # // Assign(Attr(Name('st'), Name('y')), Num('42')),
24      # Exp(Num('42')),
25      SUBI SP 1,
26      LOADI ACC 42,
27      STOREIN SP ACC 1,
28      # Ref(Global(Num('0'))),
29      SUBI SP 1,
30      LOADI IN1 0,
31      ADD IN1 DS,
32      STOREIN SP IN1 1,
33      # Ref(Attr(Stack(Num('1')), Name('y'))),
34      LOADIN SP IN1 1,
35      ADDI IN1 1,
36      STOREIN SP IN1 1,
37      LOADIN SP IN1 1,
38      LOADIN SP ACC 2,
39      ADDI SP 2,
40      STOREIN IN1 ACC 0,
41      # Return(Empty()),
42      LOADIN BAF PC -1
43    ]
44  ]

```

Code 35: RETI Blocks Pass für Zuweisung an Structattribut

## 0.0.4 Umsetzung der Derived Datatypes im Zusammenspiel

### 0.0.4.1 Einleitungsteil für Globale Statische Daten und Stackframe

```
1 struct ar_with_len {int len; int ar[2];};
2
3 void main() {
4     struct ar_with_len st_ar[3];
5     int (*pntr2)[3];
6     pntr2;
7 }
8
9 void fun() {
10    struct ar_with_len st_ar[3];
11    int (*pntr1)[3];
12    pntr1;
13 }
```

Code 36: PicoC Code für den Einleitungsteil

```
1 File
2   Name './example_derived_dts_introduction_part.ast',
3   [
4     StructDecl
5       Name 'ar_with_len',
6       [
7         Alloc
8           Writeable,
9           IntType 'int',
10          Name 'len',
11        Alloc
12          Writeable,
13          ArrayDecl
14            [
15              Num '2'
16            ],
17          IntType 'int',
18          Name 'ar'
19        ],
20      FunDef
21        VoidType 'void',
22        Name 'main',
23        [],
24        [
25          Exp
26            Alloc
27              Writeable,
28              ArrayDecl
29                [
30                  Num '3'
31                ],
32              StructSpec
33                Name 'ar_with_len',
34                Name 'st_ar',
```

```

35     Exp
36     Alloc
37     Writeable,
38     PtrDecl
39     Num '1',
40     ArrayDecl
41     [
42     Num '3'
43     ],
44     PtrDecl
45     Num '1',
46     IntType 'int',
47     Name 'ptr2',
48     Exp
49     Name 'ptr2'
50 ],
51 FunDef
52 VoidType 'void',
53 Name 'fun',
54 [],
55 [
56     Exp
57     Alloc
58     Writeable,
59     ArrayDecl
60     [
61     Num '3'
62     ],
63     StructSpec
64     Name 'ar_with_len',
65     Name 'st_ar',
66     Exp
67     Alloc
68     Writeable,
69     PtrDecl
70     Num '1',
71     ArrayDecl
72     [
73     Num '3'
74     ],
75     IntType 'int',
76     Name 'ptr1',
77     Exp
78     Name 'ptr1'
79 ]
80 ]

```

Code 37: Abstract Syntax Tree für den Einleitungsteil

```

1 File
2   Name './example_derived_dts_introduction_part.picoc_mon',
3   [
4     Block
5     Name 'main.1',

```



```
6      [
7          Exp
8              Global
9              Num '9',
10         Return
11         Empty
12     ],
13 Block
14     Name 'fun.0',
15     [
16         Exp
17             Stackframe
18             Num '9',
19         Return
20         Empty
21     ]
22 ]
```

Code 38: PicoC Mon Pass für den Einleitungsteil

```
1 File
2     Name './example_derived_dts_introduction_part.reti_blocks',
3     [
4         Block
5             Name 'main.1',
6             [
7                 # Exp(Global(Num('9'))),
8                 SUBI SP 1,
9                 LOADIN DS ACC 9,
10                STOREIN SP ACC 1,
11                # Return(Empty()),
12                LOADIN BAF PC -1
13            ],
14        Block
15            Name 'fun.0',
16            [
17                # Exp(Stackframe(Num('9'))),
18                SUBI SP 1,
19                LOADIN BAF ACC -11,
20                STOREIN SP ACC 1,
21                # Return(Empty()),
22                LOADIN BAF PC -1
23            ]
24    ]
```

Code 39: RETI Blocks Pass für den Einleitungsteil

#### 0.0.4.2 Mittelteil für die verschiedenen Derived Datatypes

```
1 struct st1 {int (*ar)[1];};
2
3 void main() {
4     int var[1] = {42};
5     struct st1 st_first = {.ar=&var};
6     (*st_first.ar)[0];
7 }
```

Code 40: PicoC Code für den Mittelteil

```
1 File
2   Name './example_derived_dts_main_part.ast',
3   [
4     StructDecl
5       Name 'st1',
6       [
7         Alloc
8           Writeable,
9           PtrDecl
10            Num '1',
11            ArrayDecl
12              [
13                Num '1'
14              ],
15            IntType 'int',
16            Name 'ar'
17          ],
18      FunDef
19        VoidType 'void',
20        Name 'main',
21        [],
22        [
23          Assign
24            Alloc
25              Writeable,
26              ArrayDecl
27                [
28                  Num '1'
29                ],
30              IntType 'int',
31              Name 'var',
32          Array
33            [
34              Num '42'
35            ],
36          Assign
37            Alloc
38              Writeable,
39              StructSpec
40                Name 'st1',
41                Name 'st_first',
42          Struct
```

```

43      [
44          Assign
45              Name 'ar',
46              Ref
47                  Name 'var'
48      ],
49  Exp
50      Subscr
51      Deref
52      Attr
53          Name 'st_first',
54          Name 'ar',
55          Num '0',
56          Num '0'
57  ]
58 ]

```

Code 41: Abstract Syntax Tree für den Mittelteil

```

1 File
2   Name './example_derived_dts_main_part.picoc_mon',
3   [
4       Block
5       Name 'main.0',
6       [
7           // Assign(Name('var'), Array([Num('42')]])),
8           Exp
9           Num '42',
10          Assign
11          Global
12          Num '0',
13          Stack
14          Num '1',
15          // Assign(Name('st_first'), Struct([Assign(Name('ar'), Ref(Name('var')))])),
16          Ref
17          Global
18          Num '0',
19          Assign
20          Global
21          Num '1',
22          Stack
23          Num '1',
24          // Exp(Subscr(Subscr(Attr(Name('st_first'), Name('ar')), Num('0')), Num('0'))),
25          Ref
26          Global
27          Num '1',
28          Ref
29          Attr
30          Stack
31          Num '1',
32          Name 'ar',
33          Exp
34          Num '0',
35          Ref

```

```

36         Subscr
37         Stack
38         Num '2',
39         Stack
40         Num '1',
41     Exp
42     Num '0',
43     Ref
44     Subscr
45     Stack
46     Num '2',
47     Stack
48     Num '1',
49     Exp
50     Stack
51     Num '1',
52     Return
53     Empty
54 ]
55 ]

```

Code 42: PicoC Mon Pass für den Mittelteil

```

1 File
2   Name './example_derived_dts_main_part.reti_blocks',
3   [
4     Block
5     Name 'main.0',
6     [
7       # // Assign(Name('var'), Array([Num('42')])),
8       # Exp(Num('42')),
9       SUBI SP 1,
10      LOADI ACC 42,
11      STOREIN SP ACC 1,
12      # Assign(Global(Num('0')), Stack(Num('1'))),
13      LOADIN SP ACC 1,
14      STOREIN DS ACC 0,
15      ADDI SP 1,
16      # // Assign(Name('st_first'), Struct([Assign(Name('ar'), Ref(Name('var')))])),
17      # Ref(Global(Num('0'))),
18      SUBI SP 1,
19      LOADI IN1 0,
20      ADD IN1 DS,
21      STOREIN SP IN1 1,
22      # Assign(Global(Num('1')), Stack(Num('1'))),
23      LOADIN SP ACC 1,
24      STOREIN DS ACC 1,
25      ADDI SP 1,
26      # // Exp(Subscr(Subscr(Attr(Name('st_first'), Name('ar')), Num('0')), Num('0'))),
27      # Ref(Global(Num('1'))),
28      SUBI SP 1,
29      LOADI IN1 1,
30      ADD IN1 DS,
31      STOREIN SP IN1 1,

```

```

32     # Ref(Attr(Stack(Num('1')), Name('ar'))),
33     LOADIN SP IN1 1,
34     ADDI IN1 0,
35     STOREIN SP IN1 1,
36     # Exp(Num('0')),
37     SUBI SP 1,
38     LOADI ACC 0,
39     STOREIN SP ACC 1,
40     # Ref(Subscr(Stack(Num('2')), Stack(Num('1')))),
41     LOADIN SP IN2 2,
42     LOADIN IN2 IN1 0,
43     LOADIN SP IN2 1,
44     MULTI IN2 1,
45     ADD IN1 IN2,
46     ADDI SP 1,
47     STOREIN SP IN1 1,
48     # Exp(Num('0')),
49     SUBI SP 1,
50     LOADI ACC 0,
51     STOREIN SP ACC 1,
52     # Ref(Subscr(Stack(Num('2')), Stack(Num('1')))),
53     LOADIN SP IN1 2,
54     LOADIN SP IN2 1,
55     MULTI IN2 1,
56     ADD IN1 IN2,
57     ADDI SP 1,
58     STOREIN SP IN1 1,
59     LOADIN SP IN1 1,
60     LOADIN IN1 ACC 0,
61     STOREIN SP ACC 1,
62     # Return(Empty()),
63     LOADIN BAF PC -1
64 ]
65 ]

```

Code 43: RETI Blocks Pass für den Mittelteil

#### 0.0.4.3 Schlussteil für die verschiedenen Derived Datatypes

```

1 struct st {int attr[2];};
2
3 void main() {
4     int ar1[1][2] = {{42, 314}};
5     struct st ar2[1] = {.attr={42, 314}};
6     int var = 42;
7     int *pntr1 = &var;
8     int **pntr2 = &pntr1;
9
10    ar1[0];
11    ar2[0];
12    *pntr2;
13 }

```

Code 44: PicoC Code für den Schlussteil

```
1 File
2   Name './example_derived_dts_final_part.ast',
3   [
4     StructDecl
5       Name 'st',
6       [
7         Alloc
8           Writeable,
9           ArrayDecl
10            [
11              Num '2'
12            ],
13            IntType 'int',
14            Name 'attr'
15        ],
16      FunDef
17        VoidType 'void',
18        Name 'main',
19        [],
20        [
21          Assign
22            Alloc
23              Writeable,
24              ArrayDecl
25                [
26                  Num '1',
27                  Num '2'
28                ],
29                IntType 'int',
30                Name 'ar1',
31              Array
32                [
33                  Array
34                    [
35                      Num '42',
36                      Num '314'
37                    ]
38                ],
39            Assign
40              Alloc
41                Writeable,
42                ArrayDecl
43                  [
44                    Num '1'
45                  ],
46                StructSpec
47                  Name 'st',
48                  Name 'ar2',
49                Struct
50                  [
51                    Assign
52                      Name 'attr',
53                      Array
54                        [
55                          Num '42',
56                          Num '314'
```

```

57         ]
58     ],
59     Assign
60     Alloc
61     Writeable,
62     IntType 'int',
63     Name 'var',
64     Num '42',
65     Assign
66     Alloc
67     Writeable,
68     PtrDecl
69     Num '1',
70     IntType 'int',
71     Name 'ptr1',
72     Ref
73     Name 'var',
74     Assign
75     Alloc
76     Writeable,
77     PtrDecl
78     Num '2',
79     IntType 'int',
80     Name 'ptr2',
81     Ref
82     Name 'ptr1',
83     Exp
84     Subscr
85     Name 'ar1',
86     Num '0',
87     Exp
88     Subscr
89     Name 'ar2',
90     Num '0',
91     Exp
92     Deref
93     Name 'ptr2',
94     Num '0'
95 ]
96 ]

```

Code 45: Abstract Syntax Tree für den Schlussteil

```

1 File
2   Name './example_derived_dts_final_part.picoc_mon',
3   [
4     Block
5     Name 'main.0',
6     [
7       // Assign(Name('ar1'), Array([Array([Num('42'), Num('314')])))),
8       Exp
9       Num '42',
10      Exp
11      Num '314',

```

```

12      Assign
13      Global
14      Num '0',
15      Stack
16      Num '2',
17      // Assign(Name('ar2'), Struct([Assign(Name('attr'), Array([Num('42'),
18      ↪ Num('314')]))])),
19      Exp
20      Num '42',
21      Exp
22      Num '314',
23      Assign
24      Global
25      Num '2',
26      Stack
27      Num '2',
28      // Assign(Name('var'), Num('42')),
29      Exp
30      Num '42',
31      Assign
32      Global
33      Num '4',
34      Stack
35      Num '1',
36      // Assign(Name('pntr1'), Ref(Name('var'))),
37      Ref
38      Global
39      Num '4',
40      Assign
41      Global
42      Num '5',
43      Stack
44      Num '1',
45      // Assign(Name('pntr2'), Ref(Name('pntr1'))),
46      Ref
47      Global
48      Num '5',
49      Assign
50      Global
51      Num '6',
52      Stack
53      Num '1',
54      // Exp(Subscr(Name('ar1'), Num('0'))),
55      Ref
56      Global
57      Num '0',
58      Exp
59      Num '0',
60      Ref
61      Subscr
62      Stack
63      Num '2',
64      Stack
65      Num '1',
66      Exp
67      Stack
68      Num '1',

```



```

68      // Exp(Subscr(Name('ar2'), Num('0'))),
69      Ref
70      Global
71      Num '2',
72      Exp
73      Num '0',
74      Ref
75      Subscr
76      Stack
77      Num '2',
78      Stack
79      Num '1',
80      Exp
81      Stack
82      Num '1',
83      // Exp(Subscr(Name('pntr2'), Num('0'))),
84      Ref
85      Global
86      Num '6',
87      Exp
88      Num '0',
89      Ref
90      Subscr
91      Stack
92      Num '2',
93      Stack
94      Num '1',
95      Exp
96      Stack
97      Num '1',
98      Return
99      Empty
100 ]
101 ]

```

Code 46: PicoC Mon Pass für den Schlussteil

```

1 File
2   Name './example_derived_dts_final_part.reti_blocks',
3   [
4     Block
5     Name 'main.0',
6     [
7       # // Assign(Name('ar1'), Array([Array([Num('42'), Num('314')])))),
8       # Exp(Num('42')),
9       SUBI SP 1,
10      LOADI ACC 42,
11      STOREIN SP ACC 1,
12      # Exp(Num('314')),
13      SUBI SP 1,
14      LOADI ACC 314,
15      STOREIN SP ACC 1,
16      # Assign(Global(Num('0')), Stack(Num('2'))),
17      LOADIN SP ACC 1,

```

```

18     STOREIN DS ACC 1,
19     LOADIN SP ACC 2,
20     STOREIN DS ACC 0,
21     ADDI SP 2,
22     # // Assign(Name('ar2'), Struct([Assign(Name('attr'), Array([Num('42'),
23     ↪ Num('314')]))])),
24     # Exp(Num('42')),
25     SUBI SP 1,
26     LOADI ACC 42,
27     STOREIN SP ACC 1,
28     # Exp(Num('314')),
29     SUBI SP 1,
30     LOADI ACC 314,
31     STOREIN SP ACC 1,
32     # Assign(Global(Num('2')), Stack(Num('2'))),
33     LOADIN SP ACC 1,
34     STOREIN DS ACC 3,
35     LOADIN SP ACC 2,
36     STOREIN DS ACC 2,
37     ADDI SP 2,
38     # // Assign(Name('var'), Num('42')),
39     # Exp(Num('42')),
40     SUBI SP 1,
41     LOADI ACC 42,
42     STOREIN SP ACC 1,
43     # Assign(Global(Num('4')), Stack(Num('1'))),
44     LOADIN SP ACC 1,
45     STOREIN DS ACC 4,
46     ADDI SP 1,
47     # // Assign(Name('pntr1'), Ref(Name('var'))),
48     # Ref(Global(Num('4'))),
49     SUBI SP 1,
50     LOADI IN1 4,
51     ADD IN1 DS,
52     STOREIN SP IN1 1,
53     # Assign(Global(Num('5')), Stack(Num('1'))),
54     LOADIN SP ACC 1,
55     STOREIN DS ACC 5,
56     ADDI SP 1,
57     # // Assign(Name('pntr2'), Ref(Name('pntr1'))),
58     # Ref(Global(Num('5'))),
59     SUBI SP 1,
60     LOADI IN1 5,
61     ADD IN1 DS,
62     STOREIN SP IN1 1,
63     # Assign(Global(Num('6')), Stack(Num('1'))),
64     LOADIN SP ACC 1,
65     STOREIN DS ACC 6,
66     ADDI SP 1,
67     # // Exp(Subscr(Name('ar1'), Num('0'))),
68     # Ref(Global(Num('0'))),
69     SUBI SP 1,
70     LOADI IN1 0,
71     ADD IN1 DS,
72     STOREIN SP IN1 1,
73     # Exp(Num('0')),
74     SUBI SP 1,

```

```

74      LOADI ACC 0,
75      STOREIN SP ACC 1,
76      # Ref(Subscr(Stack(Num('2')), Stack(Num('1')))),
77      LOADIN SP IN1 2,
78      LOADIN SP IN2 1,
79      MULTI IN2 2,
80      ADD IN1 IN2,
81      ADDI SP 1,
82      STOREIN SP IN1 1,
83      # // Exp(Subscr(Name('ar2'), Num('0'))),
84      # Ref(Global(Num('2'))),
85      SUBI SP 1,
86      LOADI IN1 2,
87      ADD IN1 DS,
88      STOREIN SP IN1 1,
89      # Exp(Num('0')),
90      SUBI SP 1,
91      LOADI ACC 0,
92      STOREIN SP ACC 1,
93      # Ref(Subscr(Stack(Num('2')), Stack(Num('1')))),
94      LOADIN SP IN1 2,
95      LOADIN SP IN2 1,
96      MULTI IN2 2,
97      ADD IN1 IN2,
98      ADDI SP 1,
99      STOREIN SP IN1 1,
100     LOADIN SP IN1 1,
101     LOADIN IN1 ACC 0,
102     STOREIN SP ACC 1,
103     # // Exp(Subscr(Name('pntr2'), Num('0'))),
104     # Ref(Global(Num('6'))),
105     SUBI SP 1,
106     LOADI IN1 6,
107     ADD IN1 DS,
108     STOREIN SP IN1 1,
109     # Exp(Num('0')),
110     SUBI SP 1,
111     LOADI ACC 0,
112     STOREIN SP ACC 1,
113     # Ref(Subscr(Stack(Num('2')), Stack(Num('1')))),
114     LOADIN SP IN2 2,
115     LOADIN IN2 IN1 0,
116     LOADIN SP IN2 1,
117     MULTI IN2 1,
118     ADD IN1 IN2,
119     ADDI SP 1,
120     STOREIN SP IN1 1,
121     # Return(Empty()),
122     LOADIN BAF PC -1
123 ]
124 ]

```

Code 47: RETI Blocks Pass für den Schlussteil

## 0.0.5 Umsetzung von Funktionen

### 0.0.5.1 Funktionen auflösen zu RETI Code

```
1 void main() {  
2     return;  
3 }  
4  
5 void fun1() {  
6 }  
7  
8 int fun2() {  
9     return 1;  
10 }
```

Code 48: PicoC Code für 3 Funktionen

```
1 File  
2   Name './example_3_funs.ast',  
3   [  
4     FunDef  
5       VoidType 'void',  
6       Name 'main',  
7       [],  
8       [  
9         Return  
10        Empty  
11      ],  
12     FunDef  
13       VoidType 'void',  
14       Name 'fun1',  
15       [],  
16       [],  
17     FunDef  
18       IntType 'int',  
19       Name 'fun2',  
20       [],  
21       [  
22         Return  
23         Num '1'  
24       ]  
25   ]
```

Code 49: Abstract Syntax Tree für 3 Funktionen

```
1 File  
2   Name './example_3_funs.picoc_blocks',  
3   [  
4     FunDef  
5       VoidType 'void',
```

```
6      Name 'main',
7      [],
8      [
9          Block
10         Name 'main.2',
11         [
12             Return
13             Empty
14         ]
15     ],
16     FunDef
17     VoidType 'void',
18     Name 'fun1',
19     [],
20     [
21         Block
22         Name 'fun1.1',
23         []
24     ],
25     FunDef
26     IntType 'int',
27     Name 'fun2',
28     [],
29     [
30         Block
31         Name 'fun2.0',
32         [
33             Return
34             Num '1'
35         ]
36     ]
37 ]
```

Code 50: PicoC Blocks Pass für 3 Funktionen

```
1 File
2   Name './example_3_funs.picoc_mon',
3   [
4       Block
5       Name 'main.2',
6       [
7           Return
8           Empty
9       ],
10      Block
11      Name 'fun1.1',
12      [
13          Return
14          Empty
15      ],
16      Block
17      Name 'fun2.0',
18      [
19          // Return(Num('1')),
```

```

20     Exp
21     Num '1',
22     Return
23     Stack
24     Num '1'
25 ]
26 ]

```

Code 51: PicoC Mon Pass für 3 Funktionen

```

1 File
2   Name './example_3_funs.reti_blocks',
3   [
4     Block
5       Name 'main.2',
6       [
7         # Return(Empty()),
8         LOADIN BAF PC -1
9       ],
10    Block
11      Name 'fun1.1',
12      [
13        # Return(Empty()),
14        LOADIN BAF PC -1
15      ],
16    Block
17      Name 'fun2.0',
18      [
19        # // Return(Num('1')),
20        # Exp(Num('1')),
21        SUBI SP 1,
22        LOADI ACC 1,
23        STOREIN SP ACC 1,
24        # Return(Stack(Num('1'))),
25        LOADIN SP ACC 1,
26        ADDI SP 1,
27        LOADIN BAF PC -1
28      ]
29 ]

```

Code 52: RETI Blocks Pass für 3 Funktionen

#### 0.0.5.1.1 Sprung zur Main Funktion

```

1 void fun1() {
2 }
3
4 int fun2() {
5     return 1;
6 }
7

```

```

8 void main() {
9     return;
10 }

```

Code 53: PicoC Code für Funktionen, wobei die main Funktion nicht die erste Funktion ist

```

1 File
2   Name './example_3_funs_main.picoc_mon',
3   [
4     Block
5       Name 'fun1.2',
6       [
7         Return
8         Empty
9       ],
10    Block
11      Name 'fun2.1',
12      [
13        // Return(Num('1')),
14        Exp
15          Num '1',
16        Return
17          Stack
18          Num '1'
19      ],
20    Block
21      Name 'main.0',
22      [
23        Return
24        Empty
25      ]
26  ]

```

Code 54: PicoC Mon Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

```

1 File
2   Name './example_3_funs_main.reti_blocks',
3   [
4     Block
5       Name 'fun1.2',
6       [
7         # Return(Empty()),
8         LOADIN BAF PC -1
9       ],
10    Block
11      Name 'fun2.1',
12      [
13        # // Return(Num('1')),
14        # Exp(Num('1')),
15        SUBI SP 1,
16        LOADI ACC 1,

```

```

17     STOREIN SP ACC 1,
18     # Return(Stack(Num('1'))),
19     LOADIN SP ACC 1,
20     ADDI SP 1,
21     LOADIN BAF PC -1
22 ],
23 Block
24   Name 'main.0',
25   [
26     # Return(Empty()),
27     LOADIN BAF PC -1
28   ]
29 ]

```

Code 55: PicoC Blocks Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

```

1 File
2   Name './example_3_funs_main.reti_patch',
3   [
4     Block
5       Name 'start.3',
6       [
7         Exp
8         GoTo
9         Name 'main.0'
10      ],
11     Block
12       Name 'fun1.2',
13       [
14         # Return(Empty()),
15         LOADIN BAF PC -1
16      ],
17     Block
18       Name 'fun2.1',
19       [
20         # // Return(Num('1')),
21         # Exp(Num('1')),
22         SUBI SP 1,
23         LOADI ACC 1,
24         STOREIN SP ACC 1,
25         # Return(Stack(Num('1'))),
26         LOADIN SP ACC 1,
27         ADDI SP 1,
28         LOADIN BAF PC -1
29      ],
30     Block
31       Name 'main.0',
32       [
33         # Return(Empty()),
34         LOADIN BAF PC -1
35      ]
36   ]

```

Code 56: PicoC Patch Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist



### 0.0.5.2 Funktionsdeklaration und -definition

```

1 int fun2(int var);
2
3 void fun1() {
4 }
5
6 void main() {
7     int var = fun2(42);
8     return;
9 }
10
11 int fun2(int var) {
12     return var;
13 }

```

Code 57: PicoC Code für Funktionen, wobei eine Funktion vorher deklariert werden muss

```

1 SymbolTable
2 [
3     Symbol(
4         {
5             type qualifier:      Empty()
6             datatype:            FunDecl(IntType('int'), Name('fun2'), [Alloc(Writeable(),
7                                     ↪ IntType('int'), Name('var'))])
8             name:                Name('fun2')
9             value or address:     Empty()
10            position:            Pos(Num('1'), Num('4'))
11            size:                Empty()
12        },
13        Symbol(
14            {
15                type qualifier:      Empty()
16                datatype:            FunDecl(VoidType('void'), Name('fun1'), [])
17                name:                Name('fun1')
18                value or address:     Empty()
19                position:            Pos(Num('3'), Num('5'))
20                size:                Empty()
21            },
22            Symbol(
23                {
24                    type qualifier:      Empty()
25                    datatype:            FunDecl(VoidType('void'), Name('main'), [])
26                    name:                Name('main')
27                    value or address:     Empty()
28                    position:            Pos(Num('6'), Num('5'))
29                    size:                Empty()
30                },
31                Symbol(
32                    {
33                        type qualifier:      Writeable()
34                        datatype:            IntType('int')
35                        name:                Name('var@main')
36                        value or address:     Num('0')

```

```

36     position:          Pos(Num('7'), Num('6'))
37     size:              Num('1')
38   },
39   Symbol(
40   {
41     type qualifier:    Writeable()
42     datatype:          IntType('int')
43     name:              Name('var@fun2')
44     value or address:  Num('0')
45     position:          Pos(Num('11'), Num('13'))
46     size:              Num('1')
47   }
48 ]

```

Code 58: Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss

### 0.0.5.3 Funktionsaufruf

#### 0.0.5.3.1 Ohne Rückgabewert

```

1 struct st {int attr1; int attr2[2];};
2
3 void stack_fun(struct st param[2][3]);
4
5 void main() {
6     struct st local_var[2][3];
7     stack_fun(local_var);
8     return;
9 }
10
11 void stack_fun(struct st param[2][3]) {
12     int local_var;
13 }

```

Code 59: PicoC Code für Funktionsaufruf ohne Rückgabewert

```

1 File
2   Name './example_fun_call_no_return_value.picoc_mon',
3   [
4     Block
5       Name 'main.1',
6       [
7         StackMalloc
8           Num '2',
9         Ref
10          Global
11            Num '0',
12          NewStackframe
13            Name 'stack_fun',
14            GoTo
15              Name 'addr@next_instr',

```

```

16     Exp
17     GoTo
18         Name 'stack_fun.0',
19     RemoveStackframe,
20     Return
21     Empty
22 ],
23 Block
24     Name 'stack_fun.0',
25     [
26         Return
27         Empty
28     ]
29 ]

```

Code 60: PicoC Mon Pass für Funktionsaufruf ohne Rückgabewert

```

1 File
2     Name './example_fun_call_no_return_value.reti_blocks',
3     [
4         Block
5             Name 'main.1',
6             [
7                 # StackMalloc(Num('2')),
8                 SUBI SP 2,
9                 # Ref(Global(Num('0'))),
10                SUBI SP 1,
11                LOADI IN1 0,
12                ADD IN1 DS,
13                STOREIN SP IN1 1,
14                # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr'))),
15                MOVE BAF ACC,
16                ADDI SP 3,
17                MOVE SP BAF,
18                SUBI SP 4,
19                STOREIN BAF ACC 0,
20                LOADI ACC GoTo
21                    Name 'addr@next_instr',
22                ADD ACC CS,
23                STOREIN BAF ACC -1,
24                Exp
25                    GoTo
26                        Name 'stack_fun.0',
27                # RemoveStackframe(),
28                MOVE BAF IN1,
29                LOADIN IN1 BAF 0,
30                MOVE IN1 SP,
31                # Return(Empty()),
32                LOADIN BAF PC -1
33            ],
34        Block
35            Name 'stack_fun.0',
36            [
37                # Return(Empty()),

```

```

38     LOADIN BAF PC -1
39   ]
40 ]

```

Code 61: RETI Blocks Pass für Funktionsaufruf ohne Rückgabewert

```

1 # StackMalloc(Num('2'))
2 SUBI SP 2;
3 # Ref(Global(Num('0')))
4 SUBI SP 1;
5 LOADI IN1 0;
6 ADD IN1 DS;
7 STOREIN SP IN1 1;
8 # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
9 MOVE BAF ACC;
10 ADDI SP 3;
11 MOVE SP BAF;
12 SUBI SP 4;
13 STOREIN BAF ACC 0;
14 LOADI ACC 14;
15 ADD ACC CS;
16 STOREIN BAF ACC -1;
17 JUMP 5;
18 # RemoveStackframe()
19 MOVE BAF IN1;
20 LOADIN IN1 BAF 0;
21 MOVE IN1 SP;
22 # Return(Empty())
23 LOADIN BAF PC -1;
24 # Return(Empty())
25 LOADIN BAF PC -1;

```

Code 62: RETI Pass für Funktionsaufruf ohne Rückgabewert

#### 0.0.5.3.2 Mit Rückgabewert

```

1 void stack_fun() {
2   return 42;
3 }
4
5 void main() {
6   int var = stack_fun();
7 }

```

Code 63: PicoC Code für Funktionsaufruf mit Rückgabewert

```

1 File
2   Name './example_fun_call_with_return_value.picoc_mon',
3   [
4     Block
5       Name 'stack_fun.1',
6       [
7         // Return(Num('42')),
8         Exp
9           Num '42',
10        Return
11          Stack
12            Num '1'
13        ],
14      Block
15        Name 'main.0',
16        [
17          // Assign(Name('var'), Call(Name('stack_fun'), [])),
18          StackMalloc
19            Num '2',
20          NewStackframe
21            Name 'stack_fun',
22            GoTo
23              Name 'addr@next_instr',
24          Exp
25            GoTo
26              Name 'stack_fun.1',
27          RemoveStackframe,
28          Assign
29            Global
30              Num '0',
31            Stack
32              Num '1',
33          Return
34            Empty
35        ]
36    ]

```

Code 64: PicoC Mon Pass für Funktionsaufruf mit Rückgabewert

```

1 File
2   Name './example_fun_call_with_return_value.reti_blocks',
3   [
4     Block
5       Name 'stack_fun.1',
6       [
7         # // Return(Num('42')),
8         # Exp(Num('42')),
9         SUBI SP 1,
10        LOADI ACC 42,
11        STOREIN SP ACC 1,
12        # Return(Stack(Num('1'))),
13        LOADIN SP ACC 1,
14        ADDI SP 1,
15        LOADIN BAF PC -1

```

```

16     ],
17     Block
18     Name 'main.0',
19     [
20         # // Assign(Name('var'), Call(Name('stack_fun'), [])),
21         # StackMalloc(Num('2')),
22         SUBI SP 2,
23         # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr'))),
24         MOVE BAF ACC,
25         ADDI SP 2,
26         MOVE SP BAF,
27         SUBI SP 2,
28         STOREIN BAF ACC 0,
29         LOADI ACC GoTo
30             Name 'addr@next_instr',
31         ADD ACC CS,
32         STOREIN BAF ACC -1,
33         Exp
34             GoTo
35             Name 'stack_fun.1',
36         # RemoveStackframe(),
37         MOVE BAF IN1,
38         LOADIN IN1 BAF 0,
39         MOVE IN1 SP,
40         # Assign(Global(Num('0')), Stack(Num('1'))),
41         LOADIN SP ACC 1,
42         STOREIN DS ACC 0,
43         ADDI SP 1,
44         # Return(Empty()),
45         LOADIN BAF PC -1
46     ]
47 ]

```

Code 65: RETI Blocks Pass für Funktionsaufruf mit Rückgabewert

```

1 JUMP 7;
2 # // Return(Num('42'))
3 # Exp(Num('42'))
4 SUBI SP 1;
5 LOADI ACC 42;
6 STOREIN SP ACC 1;
7 # Return(Stack(Num('1')))
8 LOADIN SP ACC 1;
9 ADDI SP 1;
10 LOADIN BAF PC -1;
11 # // Assign(Name('var'), Call(Name('stack_fun'), []))
12 # StackMalloc(Num('2'))
13 SUBI SP 2;
14 # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
15 MOVE BAF ACC;
16 ADDI SP 2;
17 MOVE SP BAF;
18 SUBI SP 2;
19 STOREIN BAF ACC 0;

```

```

20 LOADI ACC 17;
21 ADD ACC CS;
22 STOREIN BAF ACC -1;
23 JUMP -15;
24 # RemoveStackframe()
25 MOVE BAF IN1;
26 LOADIN IN1 BAF 0;
27 MOVE IN1 SP;
28 # Assign(Global(Num('0')), Stack(Num('1')))
29 LOADIN SP ACC 1;
30 STOREIN DS ACC 0;
31 ADDI SP 1;
32 # Return(Empty())
33 LOADIN BAF PC -1;

```

Code 66: RETI Pass für Funktionsaufruf mit Rückgabewert

### 0.0.5.3.3 Umsetzung von Call by Sharing für Arrays

```

1 void stack_fun(int (*param1)[3], int param2[2][3]) {
2 }
3
4 void main() {
5     int local_var1[2][3];
6     int local_var2[2][3];
7     stack_fun(local_var1, local_var2);
8 }

```

Code 67: PicoC Code für Call by Sharing für Arrays

```

1 File
2   Name './example_fun_call_by_sharing_array.piloc_mon',
3   [
4     Block
5       Name 'stack_fun.1',
6       [
7         Return
8         Empty
9       ],
10    Block
11      Name 'main.0',
12      [
13        StackMalloc
14        Num '2',
15        Ref
16        Global
17        Num '0',
18        Ref
19        Global
20        Num '6',
21        NewStackframe

```

```

22     Name 'stack_fun',
23     GoTo
24     Name 'addr@next_instr',
25   Exp
26     GoTo
27     Name 'stack_fun.1',
28   RemoveStackframe,
29   Return
30   Empty
31 ]
32 ]

```

Code 68: PicoC Mon Pass für Call by Sharing für Arrays

```

1 SymbolTable
2 [
3   Symbol(
4     {
5     type qualifier:      Empty()
6     datatype:            FunDecl(VoidType('void'), Name('stack_fun'),
7     ↪ [Alloc(Writable(), PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int'))),
8     ↪ Name('param1')), Alloc(Writable(), ArrayDecl([Num('3')], IntType('int')),
9     ↪ Name('param2'))])
10    name:                 Name('stack_fun')
11    value or address:      Empty()
12    position:              Pos(Num('1'), Num('5'))
13    size:                  Empty()
14  },
15  Symbol(
16    {
17    type qualifier:        Writable()
18    datatype:              PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
19    name:                   Name('param1@stack_fun')
20    value or address:       Num('0')
21    position:               Pos(Num('1'), Num('21'))
22    size:                   Num('1')
23  },
24  Symbol(
25    {
26    type qualifier:        Writable()
27    datatype:              PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
28    name:                   Name('param2@stack_fun')
29    value or address:       Num('1')
30    position:               Pos(Num('1'), Num('37'))
31    size:                   Num('1')
32  },
33  Symbol(
34    {
35    type qualifier:        Empty()
36    datatype:              FunDecl(VoidType('void'), Name('main'), [])
37    name:                   Name('main')
38    value or address:       Empty()
39    position:               Pos(Num('4'), Num('5'))
40    size:                   Empty()

```



```

38     },
39     Symbol(
40     {
41         type qualifier:      Writeable()
42         datatype:            ArrayDecl([Num('2'), Num('3')], IntType('int'))
43         name:                 Name('local_var1@main')
44         value or address:     Num('0')
45         position:             Pos(Num('5'), Num('6'))
46         size:                 Num('6')
47     },
48     Symbol(
49     {
50         type qualifier:      Writeable()
51         datatype:            ArrayDecl([Num('2'), Num('3')], IntType('int'))
52         name:                 Name('local_var2@main')
53         value or address:     Num('6')
54         position:             Pos(Num('6'), Num('6'))
55         size:                 Num('6')
56     }
57 ]

```

Code 69: Symboltabelle für Call by Sharing für Arrays

```

1 File
2   Name './example_fun_call_by_sharing_array.reti_blocks',
3   [
4     Block
5       Name 'stack_fun.1',
6       [
7         # Return(Empty()),
8         LOADIN BAF PC -1
9       ],
10    Block
11      Name 'main.0',
12      [
13        # StackMalloc(Num('2')),
14        SUBI SP 2,
15        # Ref(Global(Num('0'))),
16        SUBI SP 1,
17        LOADI IN1 0,
18        ADD IN1 DS,
19        STOREIN SP IN1 1,
20        # Ref(Global(Num('6'))),
21        SUBI SP 1,
22        LOADI IN1 6,
23        ADD IN1 DS,
24        STOREIN SP IN1 1,
25        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr'))),
26        MOVE BAF ACC,
27        ADDI SP 4,
28        MOVE SP BAF,
29        SUBI SP 4,
30        STOREIN BAF ACC 0,
31        LOADI ACC GoTo

```

```

32         Name 'addr@next_instr',
33     ADD ACC CS,
34     STOREIN BAF ACC -1,
35     Exp
36     GoTo
37     Name 'stack_fun.1',
38     # RemoveStackframe(),
39     MOVE BAF IN1,
40     LOADIN IN1 BAF 0,
41     MOVE IN1 SP,
42     # Return(Empty()),
43     LOADIN BAF PC -1
44 ]
45 ]

```

Code 70: RETI Block Pass für Call by Sharing für Arrays

#### 0.0.5.3.4 Umsetzung von Call by Value für Structs

```

1 struct st {int attr1; int attr2[2];};
2
3 void stack_fun(struct st param) {
4 }
5
6 void main() {
7     struct st local_var;
8     stack_fun(local_var);
9 }

```

Code 71: PicoC Code für Call by Value für Structs

```

1 File
2   Name './example_fun_call_by_value_struct.picoc_mon',
3   [
4     Block
5       Name 'stack_fun.1',
6       [
7         Return
8         Empty
9       ],
10    Block
11      Name 'main.0',
12      [
13        StackMalloc
14        Num '2',
15        Assign
16        Stack
17        Num '3',
18        Global
19        Num '0',
20        NewStackframe

```

```

21     Name 'stack_fun',
22     GoTo
23     Name 'addr@next_instr',
24   Exp
25     GoTo
26     Name 'stack_fun.1',
27   RemoveStackframe,
28   Return
29   Empty
30 ]
31 ]

```

## Code 72: PicoC Mon Pass für Call by Value für Structs

```

1 File
2   Name './example_fun_call_by_value_struct.reti_blocks',
3   [
4     Block
5       Name 'stack_fun.1',
6       [
7         # Return(Empty()),
8         LOADIN BAF PC -1
9       ],
10    Block
11      Name 'main.0',
12      [
13        # StackMalloc(Num('2')),
14        SUBI SP 2,
15        # Assign(Stack(Num('3')), Global(Num('0'))),
16        SUBI SP 3,
17        LOADIN DS ACC 0,
18        STOREIN SP ACC 1,
19        LOADIN DS ACC 1,
20        STOREIN SP ACC 2,
21        LOADIN DS ACC 2,
22        STOREIN SP ACC 3,
23        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr'))),
24        MOVE BAF ACC,
25        ADDI SP 5,
26        MOVE SP BAF,
27        SUBI SP 5,
28        STOREIN BAF ACC 0,
29        LOADI ACC GoTo
30          Name 'addr@next_instr',
31        ADD ACC CS,
32        STOREIN BAF ACC -1,
33      Exp
34        GoTo
35        Name 'stack_fun.1',
36      # RemoveStackframe(),
37      MOVE BAF IN1,
38      LOADIN IN1 BAF 0,
39      MOVE IN1 SP,
40      # Return(Empty()),

```

```
41      LOADIN BAF PC -1
42    ]
43  ]
```

Code 73: RETI Block Pass für Call by Value für Structs

## 0.0.6 Umsetzung kleinerer Details

# 0.1 Fehlermeldungen

## 0.1.1 Error Handler

## 0.1.2 Arten von Fehlermeldungen

### 0.1.2.1 Syntaxfehler

### 0.1.2.2 Laufzeitfehler

---

---

# Literatur