

---

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

<b>1</b>	<b>Implementierung</b>	<b>8</b>
1.1	Architektur	8
1.2	Lexikalische Analyse	8
1.2.1	Verwendung von Lark	8
1.2.2	Basic Parser	8
1.3	Syntaktische Analyse	8
1.3.1	Verwendung von Lark	8
1.3.2	Umsetzung von Präzidenz	8
1.3.3	Derivation Tree Generierung	11
1.3.4	Early Parser	11
1.3.5	Derivation Tree Vereinfachung	11
1.3.6	Abstrakt Syntax Tree Generierung	11
1.4	Code Generierung	11
1.4.1	Passes	11
1.4.2	Umsetzung von Pointern	11
1.4.3	Umsetzung von Arrays	15
1.4.4	Umsetzung von Structs	23
1.4.5	Umsetzung des Zusammenspiels der Derived Datatypes	33
1.4.6	Umsetzung von Funktionen	39
1.4.7	Umsetzung kleinerer Details	53
1.5	Fehlermeldungen	53
1.5.1	Error Handler	53
1.5.2	Arten von Fehlermeldungen	53

---

---

# Abbildungsverzeichnis

1.1	Cross-Compiler Kompiliervorgang ausgeschrieben . . . . .	9
1.2	Cross-Compiler Kompiliervorgang Kurzform . . . . .	9
1.3	Architektur mit allen Passes ausgeschrieben . . . . .	10

---

---

# Codeverzeichnis

1.1	PicoC Code für Pointer Referenzierung . . . . .	11
1.2	Abstract Syntax Tree für Pointer Referenzierung . . . . .	12
1.3	PicoC Mon Pass für Pointer Referenzierung . . . . .	13
1.4	RETI Blocks Pass für Pointer Referenzierung . . . . .	13
1.5	PicoC Code für Pointer Dereferenzierung . . . . .	13
1.6	Abstract Syntax Tree für Pointer Dereferenzierung . . . . .	14
1.7	PicoC Shrink Pass für Pointer Dereferenzierung . . . . .	15
1.8	PicoC Code für Array Initialisierung . . . . .	15
1.9	Abstract Syntax Tree für Array Initialisierung . . . . .	16
1.10	PicoC Mon Pass für Array Initialisierung . . . . .	17
1.11	RETI Blocks Pass für Array Initialisierung . . . . .	17
1.12	PicoC Code für Zugriff auf Arrayindex . . . . .	17
1.13	Abstract Syntax Tree für Zugriff auf Arrayindex . . . . .	18
1.14	PicoC Mon Pass für Zugriff auf Arrayindex . . . . .	19
1.15	RETI Blocks Pass für Zugriff auf Arrayindex . . . . .	20
1.16	PicoC Code für Zuweisung an Arrayindex . . . . .	20
1.17	Abstract Syntax Tree für Zuweisung an Arrayindex . . . . .	21
1.18	PicoC Mon Pass für Zuweisung an Arrayindex . . . . .	22
1.19	RETI Blocks Pass für Zuweisung an Arrayindex . . . . .	23
1.20	PicoC Code für Deklaration von Structs . . . . .	23
1.21	Symboltabelle für Deklaration von Structs . . . . .	24
1.22	PicoC Code für Initialisierung von Structs . . . . .	25
1.23	Abstract Syntax Tree für Initialisierung von Structs . . . . .	26
1.24	PicoC Mon Pass für Initialisierung von Structs . . . . .	26
1.25	RETI Blocks Pass für Initialisierung von Structs . . . . .	27
1.26	PicoC Code für Zugriff auf Structattribut . . . . .	27
1.27	Abstract Syntax Tree für Zugriff auf Structattribut . . . . .	28
1.28	PicoC Mon Pass für Zugriff auf Structattribut . . . . .	29
1.29	RETI Blocks Pass für Zugriff auf Structattribut . . . . .	30
1.30	PicoC Code für Zuweisung an Structattribut . . . . .	30
1.31	Abstract Syntax Tree für Zuweisung an Structattribut . . . . .	31
1.32	PicoC Mon Pass für Zuweisung an Structattribut . . . . .	32
1.33	RETI Blocks Pass für Zuweisung an Structattribut . . . . .	33
1.34	PicoC Code für Definition von Variablen . . . . .	33
1.35	Symboltabelle für Definition von Variablen . . . . .	35
1.36	PicoC Code für Zugriff auf Variablen mit Derived Datatypes . . . . .	35
1.37	Abstract Syntax Tree für Zugriff auf Variablen mit Derived Datatypes . . . . .	36
1.38	PicoC Mon Pass für Zugriff auf Variablen mit Derived Datatypes . . . . .	37
1.39	RETI Blocks Pass für Zugriff auf Variablen mit Derived Datatypes . . . . .	38
1.40	PicoC Code für 3 Funktionen . . . . .	39
1.41	Abstract Syntax Tree für 3 Funktionen . . . . .	39
1.42	PicoC Blocks Pass für 3 Funktionen . . . . .	40
1.43	PicoC Mon Pass für 3 Funktionen . . . . .	41
1.44	RETI Blocks Pass für 3 Funktionen . . . . .	41
1.45	PicoC Code für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . . . .	42
1.46	PicoC Mon Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . . . .	42
1.47	PicoC Blocks Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . . . .	43

1.48 PicoC Patch Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist . . .	44
1.49 PicoC Code für Funktionen, wobei eine Funktion vorher deklariert werden muss . . . . .	44
1.50 Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss . . . . .	45
1.51 PicoC Code für Funktionsaufruf ohne Rückgabewert . . . . .	46
1.52 PicoC Mon Pass für Funktionsaufruf ohne Rückgabewert . . . . .	46
1.53 RETI Blocks Pass für Funktionsaufruf ohne Rückgabewert . . . . .	47
1.54 RETI Pass für Funktionsaufruf ohne Rückgabewert . . . . .	48
1.55 PicoC Code für Funktionsaufruf mit Rückgabewert . . . . .	48
1.56 PicoC Mon Pass für Funktionsaufruf mit Rückgabewert . . . . .	49
1.57 RETI Blocks Pass für Funktionsaufruf mit Rückgabewert . . . . .	50
1.58 RETI Pass für Funktionsaufruf mit Rückgabewert . . . . .	51
1.59 PicoC Code für eine Funktionsdefinition . . . . .	51
1.60 Symboltabelle für eine Funktionsdefinition . . . . .	53

---

---

# Tabellenverzeichnis

1.1 Präzidenzregeln von PicoC . . . . .	10
---	----

---

---

# Definitionen

1.1	Symboltabelle . . . . .	11
-----	-------------------------	----



---

---

# 1 Implementierung

## 1.1 Architektur

## 1.2 Lexikalische Analyse

### 1.2.1 Verwendung von Lark

### 1.2.2 Basic Parser

## 1.3 Syntaktische Analyse

### 1.3.1 Verwendung von Lark

### 1.3.2 Umsetzung von Präzidenz

Die **PicoC Sprache** hat dieselben **Präzidenzregeln** implementiert, wie die **Sprache C**<sup>1</sup>. Die **Präzidenzregeln** von **PicoC** sind in Tabelle 1.1 aufgelistet.

---

<sup>1</sup>[C Operator Precedence - cppreference.com](http://c.operatorprecedence-cppreference.com).

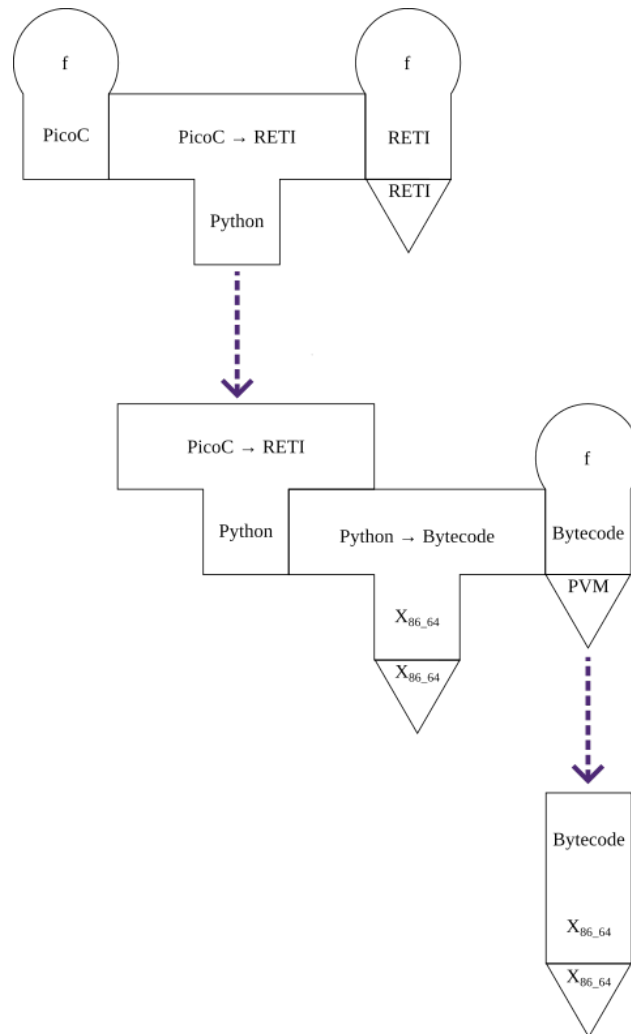


Abbildung 1.1: Cross-Compiler Kompiliervorgang ausgeschrieben

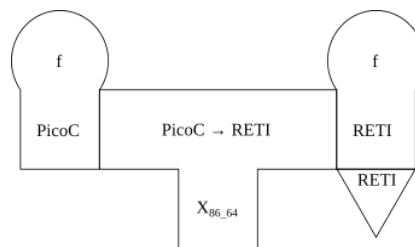


Abbildung 1.2: Cross-Compiler Kompiliervorgang Kurzform

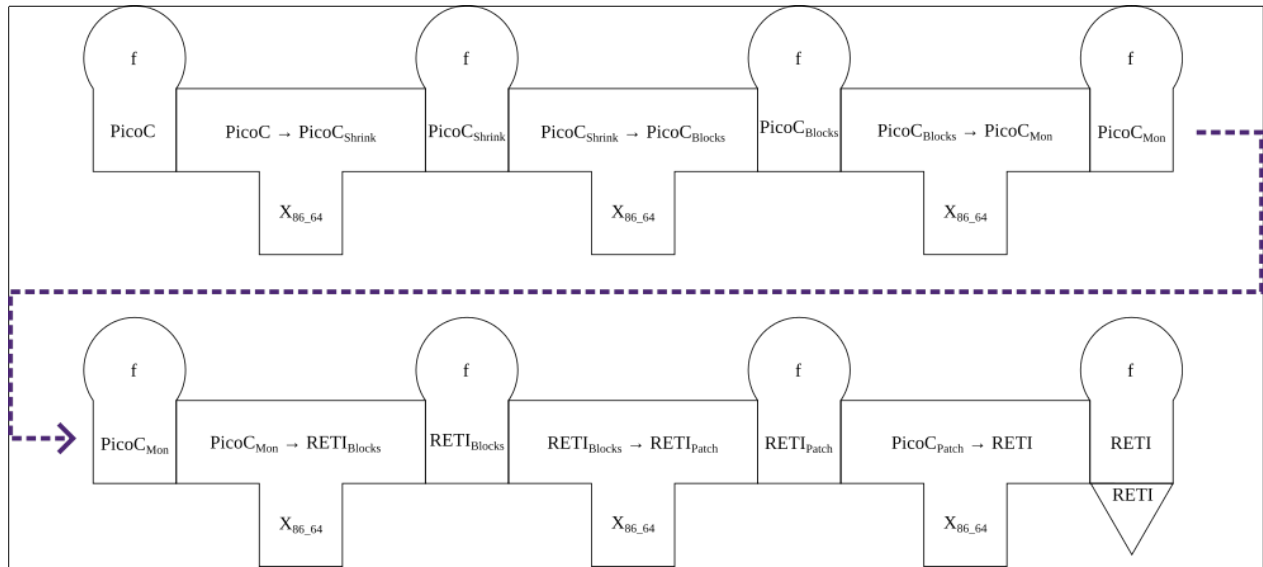


Abbildung 1.3: Architektur mit allen Passes ausgeschrieben

Präzidenz	Operator	Beschreibung	Assoziativität
1	a() a[] a.b	Funktionsaufruf Indezzugriff Attributzugriff	Links, dann rechts →
2	-a !a ~a *a &a	Unäres Minus Logisches NOT und Bitweise NOT Dereferenz und Referenz, auch Adresse-von	Rechts, dann links ←
3	a*b a/b a%b	Multiplikation, Division und Modulo	Links, dann rechts →
4	a+b a-b	Addition und Subtraktion	
5	a<b a<=b a>b a>=b	Kleiner, Kleiner Gleich, Größer, Größer gleich	
6	a==b a!=b	Gleichheit und Ungleichheit	
7	a&b	Bitweise UND	
8	a^b	Bitweise XOR (exclusive or)	
9	a b	Bitweise ODER (inclusive or)	
10	a&& b	Logisches UND	
11	a   b	Logisches ODER	
12	a=b	Zuweisung	Rechts, dann links ←
13	a, b	Komma	Links, dann rechts →

Tabelle 1.1: Präzidenzregeln von PicoC

### 1.3.3 Derivation Tree Generierung

### 1.3.4 Early Parser

### 1.3.5 Derivation Tree Vereinfachung

### 1.3.6 Abstrakt Syntax Tree Generierung

#### 1.3.6.1 ASTNode

#### 1.3.6.2 PicoC Nodes

#### 1.3.6.3 RETI Nodes

## 1.4 Code Generierung

### 1.4.1 Passes

#### 1.4.1.1 PicoC-Shrink Pass

#### 1.4.1.2 PicoC-Blocks Pass

#### 1.4.1.3 PicoC-Mon Pass

#### Definition 1.1: Symboltabelle

#### 1.4.1.4 RETI-Blocks Pass

#### 1.4.1.5 RETI-Patch Pass

#### 1.4.1.6 RETI Pass

### 1.4.2 Umsetzung von Pointern

#### 1.4.2.1 Referenzierung

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

Code 1.1: PicoC Code für Pointer Referenzierung

```
1 File
2   Name './example_ptr_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 'ptr',
22              Ref
23                Name 'var'
24      ]
25 ]
```

Code 1.2: Abstract Syntax Tree für Pointer Referenzierung

```
1 File
2   Name './example_ptr_ref.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         Exp
8           Num '42',
9         Assign
10          GlobalWrite
11            Num '0',
12          Tmp
13            Num '1',
14        Ref
15          GlobalRead
16            Num '0',
17        Assign
18          GlobalWrite
19            Num '1',
20        Tmp
21          Num '1',
22      Return
23        Empty
24    ]
25 ]
```

Code 1.3: PicoC Mon Pass für Pointer Referenzierung

```
1 File
2   Name './example_pntr_ref.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         SUBI SP 1,
8         LOADI ACC 42,
9         STOREIN SP ACC 1,
10        LOADIN SP ACC 1,
11        STOREIN DS ACC 0,
12        ADDI SP 1,
13        SUBI SP 1,
14        LOADI IN1 0,
15        ADD IN1 DS,
16        STOREIN SP IN1 1,
17        LOADIN SP ACC 1,
18        STOREIN DS ACC 1,
19        ADDI SP 1,
20        LOADIN BAF PC -1
21      ]
22    ]
```

Code 1.4: RETI Blocks Pass für Pointer Referenzierung

#### 1.4.2.2 Pointer Dereferenzierung durch Zugriff auf Arrayindex ersetzen

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

Code 1.5: PicoC Code für Pointer Dereferenzierung

```
1 File
2   Name './example_ptr_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              PtrDecl
19                Num '1',
20                IntType 'int',
21                Name 'ptr',
22              Ref
23                Name 'var',
24            Exp
25              Deref
26                Name 'ptr',
27                Num '0'
28      ]
29   ]
```

Code 1.6: 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 1.7: PicoC Shrink Pass für Pointer Dereferenzierung

### 1.4.3 Umsetzung von Arrays

#### 1.4.3.1 Initialisierung von Arrays

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

Code 1.8: 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 1.9: Abstract Syntax Tree für Array Initialisierung

```
1 File
2   Name './example_array_init.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         Exp
8           Num '4',
9         Exp
10          Num '2',
11        Assign
12          GlobalWrite
13            Num '0',
14          Tmp
15            Num '2',
16        Return
17          Empty
18      ]
19  ]
```

Code 1.10: PicoC Mon Pass für Array Initialisierung

```
1 File
2   Name './example_array_init.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         SUBI SP 1,
8         LOADI ACC 4,
9         STOREIN SP ACC 1,
10        SUBI SP 1,
11        LOADI ACC 2,
12        STOREIN SP ACC 1,
13        LOADIN SP ACC 1,
14        STOREIN DS ACC 1,
15        LOADIN SP ACC 2,
16        STOREIN DS ACC 0,
17        ADDI SP 2,
18        LOADIN BAF PC -1
19      ]
20    ]
```

Code 1.11: RETI Blocks Pass für Array Initialisierung

#### 1.4.3.2 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 1.12: 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 1.13: 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         Exp
8           Num '1',
9         Exp
10          Num '2',
11        Assign
12          GlobalWrite
13            Num '0',
14          Tmp
15            Num '2',
16        Ref
17          GlobalRead
18            Num '0',
19        Exp
20          Num '2',
21        Ref
22          Subscr
23            Tmp
24              Num '2',
25            Tmp
26              Num '1',
27        Exp
28          Subscr
29            Tmp
30              Num '1',
31            Num '0',
32        Return
33          Empty
34      ]
35  ]
```

Code 1.14: 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         SUBI SP 1,
8         LOADI ACC 1,
9         STOREIN SP ACC 1,
10        SUBI SP 1,
11        LOADI ACC 2,
12        STOREIN SP ACC 1,
13        LOADIN SP ACC 1,
14        STOREIN DS ACC 1,
15        LOADIN SP ACC 2,
16        STOREIN DS ACC 0,
17        ADDI SP 2,
18        SUBI SP 1,
19        LOADI IN1 0,
20        ADD IN1 DS,
21        STOREIN SP IN1 1,
22        SUBI SP 1,
23        LOADI ACC 2,
24        STOREIN SP ACC 1,
25        LOADIN SP IN1 2,
26        LOADIN SP IN2 1,
27        MULTI IN2 1,
28        ADD IN1 IN2,
29        ADDI SP 1,
30        STOREIN SP IN1 1,
31        LOADIN SP IN1 1,
32        LOADIN IN1 ACC 0,
33        STOREIN SP ACC 1,
34        LOADIN BAF PC -1
35      ]
36    ]

```

Code 1.15: RETI Blocks Pass für Zugriff auf Arrayindex

### 1.4.3.3 Zuweisung an Arrayindex

```

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

```

Code 1.16: 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 1.17: 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         Exp
8           Num '42',
9         Ref
10          GlobalRead
11            Num '0',
12        Exp
13          Num '2',
14        Ref
15          Subscr
16            Tmp
17              Num '2',
18            Tmp
19              Num '1',
20        Assign
21          Subscr
22            Tmp
23              Num '1',
24            Num '0',
25          Tmp
26            Num '2',
27        Return
28          Empty
29      ]
30  ]
```

Code 1.18: 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         SUBI SP 1,
8         LOADI ACC 42,
9         STOREIN SP ACC 1,
10        SUBI SP 1,
11        LOADI IN1 0,
12        ADD IN1 DS,
13        STOREIN SP IN1 1,
14        SUBI SP 1,
15        LOADI ACC 2,
16        STOREIN SP ACC 1,
17        LOADIN SP IN1 2,
18        LOADIN SP IN2 1,
19        MULTI IN2 1,
20        ADD IN1 IN2,
21        ADDI SP 1,
22        STOREIN SP IN1 1,
23        LOADIN SP IN1 1,
24        LOADIN SP ACC 2,
25        ADDI SP 2,
26        STOREIN IN1 ACC 0,
27        LOADIN BAF PC -1
28      ]
29    ]

```

Code 1.19: RETI Blocks Pass für Zuweisung an Arrayindex

## 1.4.4 Umsetzung von Structs

### 1.4.4.1 Deklaration von Structs

```

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

```

Code 1.20: 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(Writeable(),
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(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('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     )
53   )
54 ]

```

Code 1.21: Symboltabelle für Deklaration von Structs

## 1.4.4.2 Initialisierung von Structs

```

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

```

Code 1.22: 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 1.23: Abstract Syntax Tree für Initialisierung von Structs

```

1 File
2   Name './example_struct_init.picoc_mon',
3   [
4       Block
5       Name 'main.0',
6       [
7           Exp
8           Num '42',
9           Assign
10          GlobalWrite
11          Num '0',
12          Tmp
13          Num '1',
14          Ref
15          GlobalRead
16          Num '0',
17          Assign
18          GlobalWrite
19          Num '1',
20          Tmp
21          Num '1',
22          Return
23          Empty
24      ]
25 ]

```

Code 1.24: 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         SUBI SP 1,
8         LOADI ACC 42,
9         STOREIN SP ACC 1,
10        LOADIN SP ACC 1,
11        STOREIN DS ACC 0,
12        ADDI SP 1,
13        SUBI SP 1,
14        LOADI IN1 0,
15        ADD IN1 DS,
16        STOREIN SP IN1 1,
17        LOADIN SP ACC 1,
18        STOREIN DS ACC 1,
19        ADDI SP 1,
20        LOADIN BAF PC -1
21      ]
22    ]

```

Code 1.25: RETI Blocks Pass für Initialisierung von Structs

#### 1.4.4.3 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 1.26: 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 1.27: 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         Exp
8           Num '4',
9         Exp
10          Num '2',
11        Assign
12          GlobalWrite
13            Num '0',
14          Tmp
15            Num '2',
16        Ref
17          GlobalRead
18            Num '0',
19        Ref
20          Attr
21            Tmp
22              Num '1',
23            Name 'y',
24        Exp
25          Subscr
26            Tmp
27              Num '1',
28            Num '0',
29        Return
30          Empty
31      ]
32  ]
```

Code 1.28: 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         SUBI SP 1,
8         LOADI ACC 4,
9         STOREIN SP ACC 1,
10        SUBI SP 1,
11        LOADI ACC 2,
12        STOREIN SP ACC 1,
13        LOADIN SP ACC 1,
14        STOREIN DS ACC 1,
15        LOADIN SP ACC 2,
16        STOREIN DS ACC 0,
17        ADDI SP 2,
18        SUBI SP 1,
19        LOADI IN1 0,
20        ADD IN1 DS,
21        STOREIN SP IN1 1,
22        LOADIN SP IN1 1,
23        ADDI IN1 1,
24        STOREIN SP IN1 1,
25        LOADIN SP IN1 1,
26        LOADIN IN1 ACC 0,
27        STOREIN SP ACC 1,
28        LOADIN BAF PC -1
29      ]
30    ]

```

Code 1.29: RETI Blocks Pass für Zugriff auf Structattribut

#### 1.4.4.4 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 1.30: 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 1.31: 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         Exp
8           Num '4',
9         Exp
10          Num '2',
11        Assign
12          GlobalWrite
13            Num '0',
14          Tmp
15            Num '2',
16        Exp
17          Num '42',
18        Ref
19          GlobalRead
20            Num '0',
21        Ref
22          Attr
23            Tmp
24              Num '1',
25            Name 'y',
26        Assign
27          Subscr
28            Tmp
29              Num '1',
30            Num '0',
31          Tmp
32            Num '2',
33        Return
34          Empty
35      ]
36  ]
```

Code 1.32: PicoC Mon Pass für Zuweisung an Structattribut

```

1 File
2   Name './example_struct_attr_assignment.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         SUBI SP 1,
8         LOADI ACC 4,
9         STOREIN SP ACC 1,
10        SUBI SP 1,
11        LOADI ACC 2,
12        STOREIN SP ACC 1,
13        LOADIN SP ACC 1,
14        STOREIN DS ACC 1,
15        LOADIN SP ACC 2,
16        STOREIN DS ACC 0,
17        ADDI SP 2,
18        SUBI SP 1,
19        LOADI ACC 42,
20        STOREIN SP ACC 1,
21        SUBI SP 1,
22        LOADI IN1 0,
23        ADD IN1 DS,
24        STOREIN SP IN1 1,
25        LOADIN SP IN1 1,
26        ADDI IN1 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        LOADIN BAF PC -1
33      ]
34    ]

```

Code 1.33: RETI Blocks Pass für Zuweisung an Structattribut

## 1.4.5 Umsetzung des Zusammenspiels der Derived Datatypes

### 1.4.5.1 Definition von Variablen mit den Derived Datatypes

```

1 struct ar_with_len {int ar[2]; int len;};
2
3 void main() {
4   struct ar_with_len st_ar[3];
5   int (*pntr1)[3];
6   int *(*pntr2)[3];
7 }

```

Code 1.34: PicoC Code für Definition von Variablen

```

1 SymbolTable
2 [
3   Symbol(
4     {
5       type qualifier:      Empty()
6       datatype:            ArrayDecl([Num('2')], IntType('int'))
7       name:                Name('ar@ar_with_len')
8       value or address:    Empty()
9       position:            Pos(Num('1'), Num('24'))
10      size:                 Num('2')
11    },
12   Symbol(
13     {
14       type qualifier:      Empty()
15       datatype:            IntType('int')
16       name:                Name('len@ar_with_len')
17       value or address:    Empty()
18       position:            Pos(Num('1'), Num('35'))
19       size:                 Num('1')
20     },
21   Symbol(
22     {
23       type qualifier:      Empty()
24       datatype:            StructDecl(Name('ar_with_len'), [Alloc(Writeable(),
25         ↪ ArrayDecl([Num('2')], IntType('int')), Name('ar')), Alloc(Writeable(),
26         ↪ IntType('int'), Name('len'))])
27       name:                Name('ar_with_len')
28       value or address:    [Name('ar@ar_with_len'), Name('len@ar_with_len')]
29       position:            Pos(Num('1'), Num('7'))
30       size:                 Num('3')
31     },
32   Symbol(
33     {
34       type qualifier:      Empty()
35       datatype:            FunDecl(VoidType('void'), Name('main'), [])
36       name:                Name('main')
37       value or address:    Empty()
38       position:            Pos(Num('3'), Num('5'))
39       size:                 Empty()
40     },
41   Symbol(
42     {
43       type qualifier:      Writeable()
44       datatype:            ArrayDecl([Num('3')], StructSpec(Name('ar_with_len')))
45       name:                Name('st_ar@main')
46       value or address:    Num('0')
47       position:            Pos(Num('4'), Num('21'))
48       size:                 Num('9')
49     },
50   Symbol(
51     {
52       type qualifier:      Writeable()
53       datatype:            PtrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
54       name:                Name('pntr1@main')
55       value or address:    Num('9')
56       position:            Pos(Num('5'), Num('8'))
57       size:                 Num('1')

```

```

56     },
57     Symbol(
58     {
59         type qualifier:      Writeable()
60         datatype:           PntrDecl(Num('1'), ArrayDecl([Num('3')], PntrDecl(Num('1'),
61                               ↪ IntType('int'))))
62         name:               Name('pntr2@main')
63         value or address:    Num('10')
64         position:           Pos(Num('6'), Num('9'))
65         size:               Num('1')
66     }
67 ]

```

Code 1.35: Symboltabelle für Definition von Variablen

#### 1.4.5.2 Zugriff auf Variablen mit 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 1.36: PicoC Code für Zugriff auf Variablen mit Derived Datatypes

```

1 File
2   Name './example_derived_dts_combined.ast',
3   [
4     StructDecl
5       Name 'st1',
6       [
7         Alloc
8           Writeable,
9           PntrDecl
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 1.37: Abstract Syntax Tree für Zugriff auf Variablen mit Derived Datatypes

```
1 File
2   Name './example_derived_dts_combined.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         Exp
8           Num '42',
9         Assign
10          GlobalWrite
11            Num '0',
12          Tmp
13            Num '1',
14        Ref
15          GlobalRead
16            Num '0',
17        Assign
18          GlobalWrite
19            Num '1',
20          Tmp
21            Num '1',
22        Ref
23          GlobalRead
24            Num '1',
25        Ref
26          Attr
27            Tmp
28              Num '1',
29              Name 'ar',
30        Exp
31          Num '0',
32        Ref
33          Subscr
34            Tmp
35              Num '2',
36            Tmp
37              Num '1',
38        Exp
39          Num '0',
40        Ref
41          Subscr
42            Tmp
43              Num '2',
44            Tmp
45              Num '1',
46        Exp
47          Subscr
48            Tmp
49              Num '1',
50            Num '0',
51        Return
52          Empty
53      ]
54  ]
```

Code 1.38: PicoC Mon Pass für Zugriff auf Variablen mit Derived Datatypes

```

1 File
2   Name './example_derived_dts_combined.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
7         SUBI SP 1,
8         LOADI ACC 42,
9         STOREIN SP ACC 1,
10        LOADIN SP ACC 1,
11        STOREIN DS ACC 0,
12        ADDI SP 1,
13        SUBI SP 1,
14        LOADI IN1 0,
15        ADD IN1 DS,
16        STOREIN SP IN1 1,
17        LOADIN SP ACC 1,
18        STOREIN DS ACC 1,
19        ADDI SP 1,
20        SUBI SP 1,
21        LOADI IN1 1,
22        ADD IN1 DS,
23        STOREIN SP IN1 1,
24        LOADIN SP IN1 1,
25        ADDI IN1 0,
26        STOREIN SP IN1 1,
27        SUBI SP 1,
28        LOADI ACC 0,
29        STOREIN SP ACC 1,
30        LOADIN SP IN2 2,
31        LOADIN IN2 IN1 0,
32        LOADIN SP IN2 1,
33        MULTI IN2 1,
34        ADD IN1 IN2,
35        ADDI SP 1,
36        STOREIN SP IN1 1,
37        SUBI SP 1,
38        LOADI ACC 0,
39        STOREIN SP ACC 1,
40        LOADIN SP IN1 2,
41        LOADIN SP IN2 1,
42        MULTI IN2 1,
43        ADD IN1 IN2,
44        ADDI SP 1,
45        STOREIN SP IN1 1,
46        LOADIN SP IN1 1,
47        LOADIN IN1 ACC 0,
48        STOREIN SP ACC 1,
49        LOADIN BAF PC -1
50      ]
51    ]

```

Code 1.39: RETI Blocks Pass für Zugriff auf Variablen mit Derived Datatypes

## 1.4.6 Umsetzung von Funktionen

### 1.4.6.1 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 1.40: 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 1.41: 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 1.42: 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        Exp
20        Num '1',
21        Return
22        Tmp
23        Num '1'
24      ]
25  ]
```

Code 1.43: PicoC Mon Pass für 3 Funktionen

```
1 File
2   Name './example_3_funs.reti_blocks',
3   [
4     Block
5       Name 'main.2',
6       [
7         LOADIN BAF PC -1
8       ],
9     Block
10      Name 'fun1.1',
11      [
12        LOADIN BAF PC -1
13      ],
14     Block
15      Name 'fun2.0',
16      [
17        SUBI SP 1,
18        LOADI ACC 1,
19        STOREIN SP ACC 1,
20        LOADIN SP ACC 1,
21        ADDI SP 1,
22        LOADIN BAF PC -1
23      ]
24  ]
```

Code 1.44: RETI Blocks Pass für 3 Funktionen

**1.4.6.1.1 Sprung zur Main Funktion**

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

Code 1.45: 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        Exp  
14        Num '1',  
15        Return  
16        Tmp  
17        Num '1'  
18      ],  
19    Block  
20      Name 'main.0',  
21      [  
22        Return  
23        Empty  
24      ]  
25  ]
```

Code 1.46: 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         LOADIN BAF PC -1
8       ],
9     Block
10      Name 'fun2.1',
11      [
12        SUBI SP 1,
13        LOADI ACC 1,
14        STOREIN SP ACC 1,
15        LOADIN SP ACC 1,
16        ADDI SP 1,
17        LOADIN BAF PC -1
18      ],
19    Block
20      Name 'main.0',
21      [
22        LOADIN BAF PC -1
23      ]
24  ]
```

Code 1.47: 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         LOADIN BAF PC -1
15       ],
16     Block
17       Name 'fun2.1',
18       [
19         SUBI SP 1,
20         LOADI ACC 1,
21         STOREIN SP ACC 1,
22         LOADIN SP ACC 1,
23         ADDI SP 1,
24         LOADIN BAF PC -1
25       ],
26     Block
27       Name 'main.0',
28       [
29         LOADIN BAF PC -1
30       ]
31   ]

```

Code 1.48: PicoC Patch Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

#### 1.4.6.2 Funktionsdeklaration und -definition

```

1 int fun2(int var);
2
3 void fun1() {
4 }
5
6 void main() {
7   fun1();
8   fun2(42);
9   return;
10 }
11
12 int fun2(int var) {
13   return var;
14 }

```

Code 1.49: 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@fun2')
36            value or address:    Num('0')
37            position:            Pos(Num('12'), Num('13'))
38            size:                Num('1')
39          }
40        )
41      )
42    )
43  ]

```

Code 1.50: Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss

#### 1.4.6.2.1 Allocation von Variablen

#### 1.4.6.3 Funktionsaufruf

##### 1.4.6.3.1 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 1.51: 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          GlobalRead
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 1.52: 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         SUBI SP 2,
8         SUBI SP 1,
9         LOADI IN1 0,
10        ADD IN1 DS,
11        STOREIN SP IN1 1,
12        MOVE BAF ACC,
13        ADDI SP 3,
14        MOVE SP BAF,
15        SUBI SP 4,
16        STOREIN BAF ACC 0,
17        LOADI ACC GoTo
18          Name 'addr@next_instr',
19        ADD ACC CS,
20        STOREIN BAF ACC -1,
21      Exp
22        GoTo
23          Name 'stack_fun.0',
24        MOVE BAF IN1,
25        LOADIN IN1 BAF 0,
26        MOVE IN1 SP,
27        LOADIN BAF PC -1
28      ],
29    Block
30      Name 'stack_fun.0',
31      [
32        LOADIN BAF PC -1
33      ]
34  ]
```

Code 1.53: RETI Blocks Pass für Funktionsaufruf ohne Rückgabewert



```
1 SUBI SP 2;
2 SUBI SP 1;
3 LOADI IN1 0;
4 ADD IN1 DS;
5 STOREIN SP IN1 1;
6 MOVE BAF ACC;
7 ADDI SP 3;
8 MOVE SP BAF;
9 SUBI SP 4;
10 STOREIN BAF ACC 0;
11 LOADI ACC 14;
12 ADD ACC CS;
13 STOREIN BAF ACC -1;
14 JUMP 5;
15 MOVE BAF IN1;
16 LOADIN IN1 BAF 0;
17 MOVE IN1 SP;
18 LOADIN BAF PC -1;
19 LOADIN BAF PC -1;
```

Code 1.54: RETI Pass für Funktionsaufruf ohne Rückgabewert

#### 1.4.6.3.2 Mit Rückgabewert

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

Code 1.55: 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         Exp
8           Num '42',
9         Return
10          Tmp
11          Num '1'
12        ],
13     Block
14       Name 'main.0',
15       [
16         StackMalloc
17           Num '2',
18         NewStackframe
19           Name 'stack_fun',
20         GoTo
21           Name 'addr@next_instr',
22         Exp
23           GoTo
24             Name 'stack_fun.1',
25         RemoveStackframe,
26         Assign
27           GlobalWrite
28             Num '0',
29           Tmp
30             Num '1',
31         Return
32           Empty
33       ]
34   ]
```

Code 1.56: 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         SUBI SP 1,
8         LOADI ACC 42,
9         STOREIN SP ACC 1,
10        LOADIN SP ACC 1,
11        ADDI SP 1,
12        LOADIN BAF PC -1
13      ],
14    Block
15      Name 'main.0',
16      [
17        SUBI SP 2,
18        MOVE BAF ACC,
19        ADDI SP 2,
20        MOVE SP BAF,
21        SUBI SP 2,
22        STOREIN BAF ACC 0,
23        LOADI ACC GoTo
24        Name 'addr@next_instr',
25        ADD ACC CS,
26        STOREIN BAF ACC -1,
27        Exp
28        GoTo
29        Name 'stack_fun.1',
30        MOVE BAF IN1,
31        LOADIN IN1 BAF 0,
32        MOVE IN1 SP,
33        LOADIN SP ACC 1,
34        STOREIN DS ACC 0,
35        ADDI SP 1,
36        LOADIN BAF PC -1
37      ]
38    ]
```

Code 1.57: RETI Blocks Pass für Funktionsaufruf mit Rückgabewert

```

1 JUMP 7;
2 SUBI SP 1;
3 LOADI ACC 42;
4 STOREIN SP ACC 1;
5 LOADIN SP ACC 1;
6 ADDI SP 1;
7 LOADIN BAF PC -1;
8 SUBI SP 2;
9 MOVE BAF ACC;
10 ADDI SP 2;
11 MOVE SP BAF;
12 SUBI SP 2;
13 STOREIN BAF ACC 0;
14 LOADI ACC 17;
15 ADD ACC CS;
16 STOREIN BAF ACC -1;
17 JUMP -15;
18 MOVE BAF IN1;
19 LOADIN IN1 BAF 0;
20 MOVE IN1 SP;
21 LOADIN SP ACC 1;
22 STOREIN DS ACC 0;
23 ADDI SP 1;
24 LOADIN BAF PC -1;

```

Code 1.58: RETI Pass für Funktionsaufruf mit Rückgabewert

#### 1.4.6.3.3 Umsetzung von Call by Sharing für Arrays

```

1 int stack_fun(int (*param1)[3], int param2[2][3]) {
2     int local_var;
3 }
4
5 void main() {
6     int local_var1[2][3];
7     int local_var2[2][3];
8     stack_fun(local_var1, local_var2);
9 }

```

Code 1.59: PicoC Code für eine Funktionsdefinition

```

1 SymbolTable
2 [
3     Symbol(
4         {
5             type qualifier:      Empty()

```

```

6      datatype:      FunDecl(IntType('int'), Name('stack_fun'),
   ↪ [Alloc(Writable(), PtrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int'))),
   ↪ Name('param1')), Alloc(Writable(), ArrayDecl([Num('3')], IntType('int')),
   ↪ Name('param2'))])
7      name:          Name('stack_fun')
8      value or address: Empty()
9      position:      Pos(Num('1'), Num('4'))
10     size:          Empty()
11 },
12 Symbol(
13 {
14     type qualifier:  Writable()
15     datatype:        PtrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
16     name:            Name('param1@stack_fun')
17     value or address: Num('0')
18     position:        Pos(Num('1'), Num('20'))
19     size:            Num('1')
20 },
21 Symbol(
22 {
23     type qualifier:  Writable()
24     datatype:        PtrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
25     name:            Name('param2@stack_fun')
26     value or address: Num('1')
27     position:        Pos(Num('1'), Num('36'))
28     size:            Num('1')
29 },
30 Symbol(
31 {
32     type qualifier:  Writable()
33     datatype:        IntType('int')
34     name:            Name('local_var@stack_fun')
35     value or address: Num('2')
36     position:        Pos(Num('2'), Num('6'))
37     size:            Num('1')
38 },
39 Symbol(
40 {
41     type qualifier:  Empty()
42     datatype:        FunDecl(VoidType('void'), Name('main'), [])
43     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:  Writable()
51     datatype:        ArrayDecl([Num('2'), Num('3')], IntType('int'))
52     name:            Name('local_var1@main')
53     value or address: Num('0')
54     position:        Pos(Num('6'), Num('6'))
55     size:            Num('6')
56 },
57 Symbol(
58 {
59     type qualifier:  Writable()

```

```
60     datatype:      ArrayDecl([Num('2'), Num('3')], IntType('int'))
61     name:          Name('local_var2@main')
62     value or address: Num('6')
63     position:      Pos(Num('7'), Num('6'))
64     size:          Num('6')
65 }
66 ]
```

Code 1.60: Symboltabelle für eine Funktionsdefinition

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

### 1.4.7 Umsetzung kleinerer Details

## 1.5 Fehlermeldungen

### 1.5.1 Error Handler

### 1.5.2 Arten von Fehlermeldungen

#### 1.5.2.1 Syntaxfehler

#### 1.5.2.2 Laufzeitfehler

---

---

# Literatur

## Online

- *C Operator Precedence* - *cppreference.com*. URL: [https://en.cppreference.com/w/c/language/operator\\_precedence](https://en.cppreference.com/w/c/language/operator_precedence) (besucht am 27.04.2022).