Albert Ludwigs Universität Freiburg

Technische Fakultät

PicoC-Compiler

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

Bachelorarbeit

Abgabedatum: 28th April 2022

Author:
Jürgen Mattheis

Gutachter:
Prof. Dr. Scholl

Betreung:
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

Abbildungsverzeichnis

Codeverzeichnis

Tabellenverzeichnis

Definitionsverzeichnis

Grammatikverzeichnis

1 Implementierung

1.1 Architektur

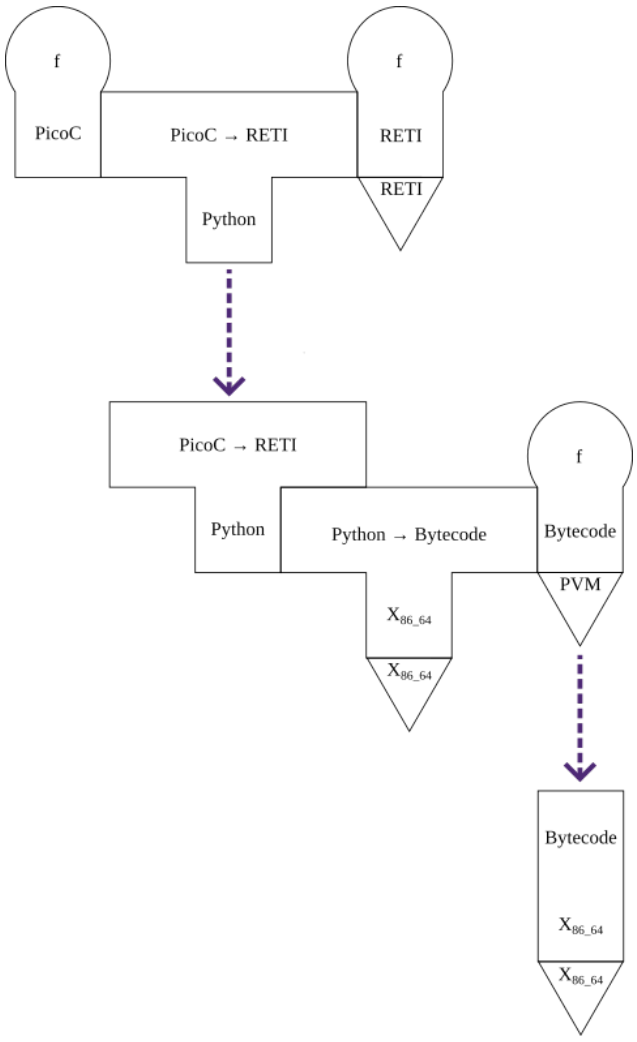

Abbildung 1.1: Cross-Compiler Kompiliervorgang ausgeschrieben

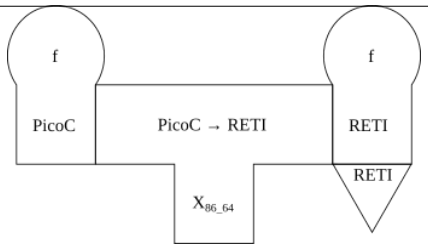

Abbildung 1.2: Cross-Compiler Kompiliervorgang Kurzform

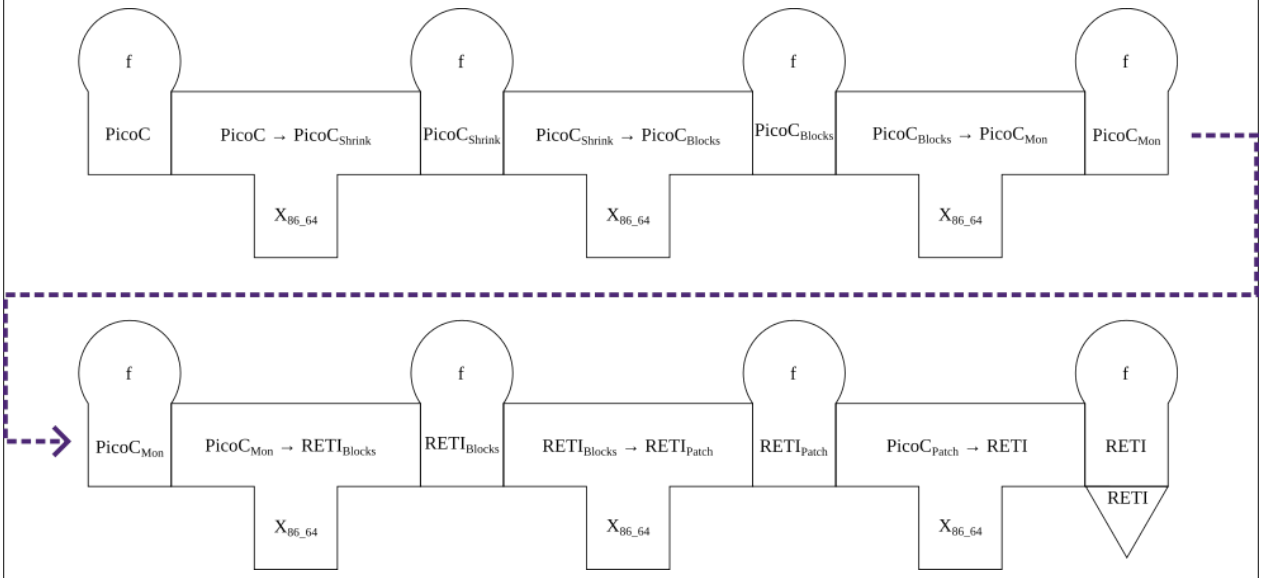

Abbildung 1.3: Architektur mit allen Passes ausgeschrieben

1.2 Lexikalische Analyse

1.2.1 Verwendung von Lark

COMMENT	::=	"//" /[^\n]*/ | "/*" /(.|\n)*?/ "*/"	*L_Comment*
RETI_COMMENT.2	::=	"//""␣"?"#"/[^\n]*/	
DIG_NO_0	::=	"1" | "2" | "3" | "4" | "5"	*L_Arith*
	|	"6" | "7" | "8" | "9"	
DIG_WITH_0	::=	"0" | *DIG_NO_0*	
NUM	::=	"0" | *DIG_NO_0DIG_WITH_0**	
ASCII_CHAR	::=	"␣".."~"	
CHAR	::=	"'"*ASCII_CHAR*"'"	
FILENAME	::=	*ASCII_CHAR* + ".*picoc*"	
LETTER	::=	"*a*".."*z*" | "*A*".."*Z*"	
NAME	::=	(*LETTER* | "_") (LETTER — DIG_WITH_0 — "_")*	
name	::=	*NAME* | *INT_NAME* | *CHAR_NAME*	
	|	*VOID_NAME*	
NOT	::=	"~"	
REF_AND	::=	"&"	
un_op	::=	*SUB_MINUS* | *LOGIC_NOT* | *NOT*	
	|	*MUL_DEREF_PNTR* | *REF_AND*	
MUL_DEREF_PNTR	::=	"*"	
DIV	::=	"/"	
MOD	::=	"%"	
*prec*1*_op*	::=	*MUL_DEREF_PNTR* | *DIV* | *MOD*	
ADD	::=	"+"	
SUB_MINUS	::=	"−"	
*prec*2*_op*	::=	*ADD* | *SUB_MINUS*	
LT	::=	"<"	*L_Logic*
LTE	::=	"<="	
GT	::=	">"	
GTE	::=	">="	
rel_op	::=	*LT* | *LTE* | *GT* | *GTE*	
EQ	::=	"=="	
NEQ	::=	"!="	
eq_op	::=	*EQ* | *NEQ*	
LOGIC_NOT	::=	"!"	
INT_DT.2	::=	"*int*"	
INT_NAME.3	::=	"*int*" (*LETTER* | *DIG_WITH_0* | "_")+	*L_Assign_Alloc*
CHAR_DT.2	::=	"*char*"	
CHAR_NAME.3	::=	"*char*" (*LETTER* | *DIG_WITH_0* | "_")+	
VOID_DT.2	::=	"*void*"	
VOID_NAME.3	::=	"*void*" (*LETTER* | *DIG_WITH_0* | "_")+	
prim_dt	::=	*INT_DT* | *CHAR_DT* | *VOID_DT*	

Grammar 1.2.1: Konkrette Syntax des Lexers in EBNF

1.2.2 Basic Parser

1.3 Syntaktische Analyse

1.3.1 Verwendung von Lark

In 1.3.1

prim_exp	::=	*name* | *NUM* | *CHAR* | "("*logic_or*")"	*L_Arith +*
post_exp	::=	*array_subscr* | *struct_attr* | *fun_call*	*L_Array +*
		| *input_exp* | *print_exp* | *prim_exp*	*L_Pntr +*
un_exp	::=	*un_opun_exp* | *post_exp*	*L_Struct + L_Fun*
input_exp	::=	"*input*""("")"	*L_Arith*
print_exp	::=	"*print*""("*logic_or*")"	
*arith_prec*1	::=	*arith_prec*1 *prec*1*_op un_exp* | *un_exp*	
*arith_prec*2	::=	*arith_prec*2 *prec*2*_op arith_prec*1 | *arith_prec*1	
arith_and	::=	*arith_and* "&" *arith_prec*2 | *arith_prec*2	
arith_oplus	::=	*arith_oplus* "∧" *arith_and* | *arith_and*	
arith_or	::=	*arith_or* "|" *arith_oplus* | *arith_oplus*	
rel_exp	::=	*rel_exp rel_op arith_or* | *arith_or*	*L_Logic*
eq_exp	::=	*eq_exp eq_oprel_exp* | *rel_exp*	
logic_and	::=	*logic_and* "&&" *eq_exp* | *eq_exp*	
logic_or	::=	*logic_or* "||" *logic_and* | *logic_and*	
type_spec	::=	*prim_dt* | *struct_spec*	*L_Assign_Alloc*
alloc	::=	*type_spec pntr_decl*	
assign_stmt	::=	*un_exp* "=" *logic_or*";"	
initializer	::=	*logic_or* | *array_init* | *struct_init*	
init_stmt	::=	*alloc* "=" *initializer*";"	
const_init_stmt	::=	"*const*" *type_spec name* "=" *NUM*";"	
pntr_deg	::=	"∗"∗	*L_Pntr*
pntr_decl	::=	*pntr_deg array_decl* | *array_decl*	
array_dims	::=	("["*NUM*"]")∗	*L_Array*
array_decl	::=	*name array_dims* | "("*pntr_decl*")"*array_dims*	
array_init	::=	"{"*initializer*("," *initializer*) ∗ "}"	
array_subscr	::=	*post_exp*"["*logic_or*"]"	
struct_spec	::=	"*struct*" *name*	*L_Struct*
struct_params	::=	(*alloc*";")+	
struct_decl	::=	"*struct*" *name* "{"*struct_params*"}"	
struct_init	::=	"{""."*name*"="*initializer* ("," "."name"="initializer)*"}"	
struct_attr	::=	*post_exp*"."*name*	
if_stmt	::=	"*if*""("*logic_or*")" *exec_part*	*L_If_Else*
if_else_stmt	::=	"*if*""("*logic_or*")" *exec_part* "*else*" *exec_part*	
while_stmt	::=	"*while*""("*logic_or*")" *exec_part*	*L_Loop*
do_while_stmt	::=	"*do*" *exec_part* "*while*""("*logic_or*")"";"	

Grammar 1.3.1: Konkrette Syntax des Parsers in EBNF, Teil 1

decl_exp_stmt	::=	*alloc*";"	*L_Stmt*
decl_direct_stmt	::=	*assign_stmt* | *init_stmt* | *const_init_stmt*	
decl_part	::=	*decl_exp_stmt* | *decl_direct_stmt* | *RETI_COMMENT*	
compound_stmt	::=	"{" *exec_part* ∗ "}"	
exec_exp_stmt	::=	*logic_or*";"	
exec_direct_stmt	::=	*if_stmt* | *if_else_stmt* | *while_stmt* | *do_while_stmt*	
	|	*assign_stmt* | *fun_return_stmt*	
exec_part	::=	*compound_stmt* | *exec_exp_stmt* | *exec_direct_stmt*	
	|	*RETI_COMMENT*	
decl_exec_stmts	::=	*decl_part* ∗ *exec_part*∗	
fun_args	::=	[*logic_or*("," *logic_or*)∗]	*L_Fun*
fun_call	::=	*name*"(" *fun_args*")"	
fun_return_stmt	::=	"*return*" [*logic_or*]";"	
fun_params	::=	[*alloc*("," *alloc*)∗]	
fun_decl	::=	*type_spec pntr_deg name*"(" *fun_params*")"	
fun_def	::=	*type_spec pntr_deg name*"(" *fun_params*")" "{" *decl_exec_stmts*"}"	
decl_def	::=	(*struct_decl* | *fun_decl*)";" | *fun_def*	*L_File*
decls_defs	::=	*decl_def*∗	
file	::=	*FILENAME decls_defs*	

Grammar 1.3.2: Konkrette Syntax des Parsers in EBNF, Teil 2

1.3.2 Umsetzung von Präzidenz

Die **PicoC** Programmiersprache hat dieselben **Präzidenzregeln** implementiert, wie die Programmiersprache **C**[1]. Die **Präzidenzregeln** von **PicoC** sind in Tabelle 1.1 aufgelistet.

Präzidenz	Operator	Beschreibung	Assoziativität
1	`a()`	Funktionsaufruf	Links, dann rechts →
	`a[]`	Indexzugriff	
	`a.b`	Attributzugriff	
2	`-a`	Unäres Minus	Rechts, dann links ←
	`!a ~a`	Logisches NOT und Bitweise NOT	
	`*a &a`	Dereferenz und Referenz, auch Adresse-von	
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`	Logiches 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] *C Operator Precedence - cppreference.com.*

1.3.3 Derivation Tree Generierung

1.3.3.1 Early Parser

1.3.4 Derivation Tree Vereinfachung

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

Code 1.1: PicoC Code für Pointer Referenzierung

1.3.5 Abstrakt Syntax Tree Generierung

1.3.5.1 PicoC Nodes

1.3.5.2 RETI Nodes

1.3.5.3 Kompositionen mit besonderer Bedeutung

1.3.5.4 Abstrakte Syntax

un_op	$::=$	$Minus() \mid Not()$	*L_Arith*
bin_op	$::=$	$Add() \mid Sub() \mid Mul() \mid Div() \mid Mod()$	
	$\mid$	$Oplus() \mid And() \mid Or()$	
exp	$::=$	$Name(str) \mid Num(str) \mid Char(str)$	
	$\mid$	$BinOp(\langle exp\rangle, \langle bin\_op\rangle, \langle exp\rangle)$	
	$\mid$	$UnOp(\langle un\_op\rangle, \langle exp\rangle) \mid Call(Name('input'), None)$	
exp_stmts	$::=$	$Alloc(\langle type\_qual\rangle, \langle dataype\rangle, Name(str))$	
	$\mid$	$Call(Name('print'), \langle exp\rangle)$	
un_op	$::=$	$LogicNot()$	*L_Logic*
rel	$::=$	$Eq() \mid NEq() \mid Lt() \mid LtE() \mid Gt() \mid GtE()$	
bin_op	$::=$	$LogicAnd() \mid LogicOr()$	
exp	$::=$	$Atom(\langle exp\rangle, \langle rel\rangle, \langle exp\rangle)$	
	$\mid$	$ToBool(\langle exp\rangle)$	
$type\_qual$	$::=$	$Const() \mid Writeable()$	*L_Assign_Alloc*
$datatype$	$::=$	$IntType() \mid CharType() \mid VoidType()$	
$assign\_lhs$	$::=$	$Alloc(\langle type\_qual\rangle, \langle dataype\rangle, Name(str)) \mid \langle rel\_loc\rangle$	
exp_stmts	$::=$	$Alloc(\langle type\_qual\rangle, \langle dataype\rangle, Name(str))$	
$stmt$	$::=$	$Assign(\langle assign\_lhs\rangle, \langle exp\rangle)$	
	$\mid$	$Exp(\langle exp\_stmts\rangle)$	
$datatype$	$::=$	$PntrDecl(Num(str), \langle datatype\rangle)$	*L_Pntr*
$deref\_loc$	$::=$	$Ref(\langle ref\_loc\rangle) \mid \langle ref\_loc\rangle$	
ref_loc	$::=$	$Name(str)$	
	$\mid$	$Deref(\langle deref\_loc\rangle, \langle exp\rangle)$	
	$\mid$	$Subscr(\langle deref\_loc\rangle, \langle exp\rangle)$	
	$\mid$	$Attr(\langle ref\_loc\rangle, Name(str))$	
exp	$::=$	$Deref(\langle deref\_loc\rangle, \langle exp\rangle)$	
	$\mid$	$Ref(\langle ref\_loc\rangle)$	
$datatype$	$::=$	$ArrayDecl(Num(str)+, \langle datatype\rangle)$	*L_Array*
exp	$::=$	$Subscr(\langle deref\_loc\rangle, \langle exp\rangle) \mid Array(\langle exp\rangle+)$	
$datatype$	$::=$	$StructSpec(Name(str))$	*L_Struct*
exp	$::=$	$Attr(\langle ref\_loc\rangle, Name(str))$	
	$\mid$	$Struct(Assign(Name(str), \langle exp\rangle)+)$	
$decl\_def$	$::=$	$StructDecl(Name(str),$	
		$\quad Alloc(Writeable(), \langle datatype\rangle, Name(str))+)$	
$stmt$	$::=$	$If(\langle exp\rangle, \langle stmt\rangle*)$	*L_If_Else*
	$\mid$	$IfElse(\langle exp\rangle, \langle stmt\rangle*, \langle stmt\rangle*)$	
$stmt$	$::=$	$While(\langle exp\rangle, \langle stmt\rangle*)$	*L_Loop*
	$\mid$	$DoWhile(\langle exp\rangle, \langle stmt\rangle*)$	
exp	$::=$	$Call(Name(str), \langle exp\rangle*)$	*L_Fun*
exp_stmts	$::=$	$Call(Name(str), \langle exp\rangle*)$	
$stmt$	$::=$	$Return(\langle exp\rangle)$	
$decl\_def$	$::=$	$FunDecl(\langle datatype\rangle, Name(str),$	
		$\quad Alloc(Writeable(), \langle datatype\rangle, Name(str))*)$	
	$\mid$	$FunDef(\langle datatype\rangle, Name(str),$	
		$\quad Alloc(Writeable(), \langle datatype\rangle, Name(str))*, \langle stmt\rangle*)$	
$file$	$::=$	$File(Name(str), \langle decl\_def\rangle*)$	*L_File*

Grammar 1.3.3: Abstrakte Syntax für L_{PiocC}

1.4 Code Generierung

1.4.1 Passes

1.4.1.1 PicoC-Shrink Pass

1.4.1.2 PicoC-Blocks Pass

$decl\_def$	::=	$FunDef(\langle datatype\rangle, Name(str),$ $Alloc(Writeable(), \langle datatype\rangle, Name(str))*, \langle block\rangle*)$	*L_Fun*
$block$	::=	$Block(Name(str), \langle stmt\rangle*)$	*L_Blocks*
$stmt$	::=	$Goto(Name(str)) \mid NewStackframe(Name(), Goto(str))$	
	|	$RemoveStackframe() \mid SetScope(Name(str))$	
	|	$SingleLineComment(str, str)$	

Grammar 1.4.1: Abstrakte Syntax für L_{PicoC_Blocks}

1.4.1.3 PicoC-Mon Pass

ref_loc	::=	$Tmp(Num(str)) \mid StackRead(Num(str))$	*L_Assign_Alloc*
	|	$StackWrite(Num(str)) \mid GlobalRead(Num(str))$	
	|	$GlobalWrite(Num(str))$	
$error\_data$	::=	$\langle exp\rangle \mid Pos(Num(str), Num(str))$	
exp	::=	$Stack(Num(str)) \mid Ref(\langle ref_loc\rangle, \langle datatype\rangle, \langle error_data\rangle)$	
$stmt$	::=	$Exp(\langle exp\rangle)$	
	|	$Assign(Alloc(Writeable(), StructSpec(Name(str)), Name(str)),$ $Struct(Assign(Name(str), \langle exp\rangle)+, \langle datatype\rangle\ ))$	
	|	$Assign(Alloc(Writeable(), ArrayDecl(Num(str)+, \langle datatype\rangle)),$ $Name(str)), Array(\langle exp\rangle+, \langle datatype\rangle))$	
$symbol\_table$	::=	$SymbolTable(\langle symbol\rangle)$	*L_Symbol_Table*
$symbol$	::=	$Symbol(\langle type_qual\rangle, \langle datatype\rangle, \langle name\rangle, \langle val\rangle, \langle pos\rangle, \langle size\rangle)$	
$type\_qual$	::=	$Empty()$	
$datatype$	::=	$BuiltIn() \mid SelfDefined()$	
$name$	::=	$Name(str)$	
val	::=	$Num(str) \mid Empty()$	
pos	::=	$Pos(Num(str), Num(str)) \mid Empty()$	
$size$	::=	$Num(str) \mid Empty()$	

Grammar 1.4.2: Abstrakte Syntax für L_{PicoC_Mon}

Definition 1.1: Symboltabelle

1.4.1.4 RETI-Blocks Pass

$program$	$::=$	$Program(Name(str), \langle block \rangle *)$	*L_Program*
exp_stmts	$::=$	$Goto(str)$	*L_Blocks*
$instrs\_before$	$::=$	$Num(str)$	
num_instrs	$::=$	$Num(str)$	
$block$	$::=$	$Block(Name(str), \langle instr \rangle *, \langle instrs\_before \rangle, \langle num\_instrs \rangle)$	
$instr$	$::=$	$Goto(Name(str))$	

Grammar 1.4.3: Abstrakte Syntax für L_{RETI_Blocks}

1.4.1.5 RETI-Patch Pass

$stmt$	$::=$	$Exit(Num(str))$

Grammar 1.4.4: Abstrakte Syntax für L_{RETI_Patch}

1.4.1.6 RETI Pass

reg	$::=$	$ACC() \mid IN1() \mid IN2() \mid PC() \mid SP() \mid BAF()$	*L_Program*
	$\mid$	$CS() \mid DS()$	
arg	$::=$	$Reg(\langle reg \rangle) \mid Num(str)$	
rel	$::=$	$Eq() \mid NEq() \mid Lt() \mid LtE() \mid Gt() \mid GtE()$	
	$\mid$	$Always() \mid NOp()$	
op	$::=$	$Add() \mid Addi() \mid Sub() \mid Subi() \mid Mult()$	
	$\mid$	$Multi() \mid Div() \mid Divi()$	
	$\mid$	$Mod() \mid Modi() \mid Oplus() \mid Oplusi() \mid Or()$	
	$\mid$	$Ori() \mid And() \mid Andi()$	
	$\mid$	$Load() \mid Loadin() \mid Loadi()$	
	$\mid$	$Store() \mid Storein() \mid Move()$	
$instr$	$::=$	$Instr(\langle op \rangle, \langle arg \rangle +) \mid Jump(\langle rel \rangle, Num(str)) \mid Int(Num(str))$	
	$\mid$	$RTI() \mid Call(Name('print'), \langle reg \rangle) \mid Call(Name('input'), \langle reg \rangle)$	
	$\mid$	$SingleLineComment(str, str)$	
$program$	$::=$	$Program(Name(str), \langle instr \rangle *)$	

Grammar 1.4.5: Abstrakte Syntax für L_{RETI}

1.4.2 Umsetzung von Pointern

1.4.2.1 Referenzierung

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

Code 1.2: PicoC Code für Pointer Referenzierung

```
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            PntrDecl
19              Num '1',
20              IntType 'int',
21            Name 'pntr',
22          Ref
23            Name 'var'
24      ]
25  ]
```

Code 1.3: Abstract Syntax Tree für Pointer Referenzierung

```
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:               IntType('int')
16        name:                   Name('var@main')
17        value or address:       Num('0')
18        position:               Pos(Num('2'), Num('6'))
19        size:                   Num('1')
20      },
21    Symbol(
22      {
23        type qualifier:         Writeable()
24        datatype:               PntrDecl(Num('1'), IntType('int'))
25        name:                   Name('pntr@main')
26        value or address:       Num('1')
27        position:               Pos(Num('3'), Num('7'))
28        size:                   Num('1')
29      }
30  ]
```

Code 1.4: Symboltabelle für Pointer Referenzierung

```
1 File
2   Name './example_pntr_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 1.5: PicoC Mon Pass für Pointer Referenzierung

```
File
  Name './example_pntr_ref.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.6: RETI Blocks Pass für Pointer Referenzierung

1.4.2.2 Pointer Dereferenzierung durch Zugriff auf Arrayindex ersetzen

```
void main() {
  int var = 42;
  int *pntr = &var;
  *pntr;
}
```

Code 1.7: PicoC Code für Pointer Dereferenzierung

```
File
  Name './example_pntr_deref.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            IntType 'int',
            Name 'var',
          Num '42',
        Assign
          Alloc
            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 1.8: 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 1.9: 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 1.10: 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 1.11: Abstract Syntax Tree für Array Initialisierung

```
1 SymbolTable
2   [
3     Symbol(
4       {
5         type qualifier:          Empty()
6         datatype:                FunDecl(VoidType('void'), Name('main'), [])
```

```
      name:                   Name('main')
      value or address:       Empty()
      position:               Pos(Num('1'), Num('5'))
      size:                   Empty()
    },
  Symbol(
    {
      type qualifier:         Writeable()
      datatype:               ArrayDecl([Num('2'), Num('1')], IntType('int'))
      name:                   Name('ar@main')
      value or address:       Num('0')
      position:               Pos(Num('2'), Num('6'))
      size:                   Num('2')
    }
]
```

Code 1.12: Symboltabelle für Array Initialisierung

```
File
  Name './example_array_init.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '4',
        Exp
          Num '2',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '2',
        Return
          Empty
      ]
  ]
```

Code 1.13: PicoC Mon Pass für Array Initialisierung

```
File
  Name './example_array_init.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 4,
        STOREIN SP ACC 1,
        SUBI SP 1,
        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 1.14: 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 1.15: 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 1.16: Abstract Syntax Tree für Zugriff auf Arrayindex

```
File
  Name './example_array_access.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '1',
        Exp
          Num '2',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '2',
        Ref
          GlobalRead
            Num '0',
        Exp
          Num '2',
        Ref
          Subscr
            Tmp
              Num '2',
            Tmp
              Num '1',
        Exp
          Subscr
            Tmp
              Num '1',
            Num '0',
        Return
          Empty
      ]
  ]
```

Code 1.17: PicoC Mon Pass für Zugriff auf Arrayindex

```
File
  Name './example_array_access.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI ACC 2,
```

```
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP IN1 2,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN IN1 ACC 0,
        STOREIN SP ACC 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.18: RETI Blocks Pass für Zugriff auf Arrayindex

1.4.3.3 Zuweisung an Arrayindex

```
void main() {
  int ar[2];
  ar[2] = 42;
}
```

Code 1.19: PicoC Code für Zuweisung an Arrayindex

```
File
  Name './example_array_assignment.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Exp
          Alloc
            Writeable,
            ArrayDecl
              [
                Num '2'
              ],
```

```
16              IntType 'int',
17            Name 'ar',
18          Assign
19            Subscr
20              Name 'ar',
21              Num '2',
22            Num '42'
23        ]
24    ]
```

Code 1.20: 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 1.21: PicoC Mon Pass für Zuweisung an Arrayindex

```
1 File
2   Name './example_array_assignment.reti_blocks',
3   [
4     Block
5       Name 'main.0',
6       [
```

```
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP IN1 2,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN SP ACC 2,
        ADDI SP 2,
        STOREIN IN1 ACC 0,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.22: RETI Blocks Pass für Zuweisung an Arrayindex

1.4.4 Umsetzung von Structs

1.4.4.1 Deklaration von Structs

```
struct st1 {int *ar[3];};

struct st2 {struct st1 st;};

void main() {
}
```

Code 1.23: PicoC Code für Deklaration von Structs

```
SymbolTable
  [
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               ArrayDecl([Num('3')], PntrDecl(Num('1'), IntType('int')))
        name:                   Name('ar@st1')
        value or address:       Empty()
        position:               Pos(Num('1'), Num('17'))
        size:                   Num('3')
      },
    Symbol(
```

```
13      {
14        type qualifier:         Empty()
15        datatype:               StructDecl(Name('st1'), [Alloc(Writeable(),
          ↪  ArrayDecl([Num('3')], PntrDecl(Num('1'), IntType('int'))), Name('ar'))])
16        name:                   Name('st1')
17        value or address:       [Name('ar@st1')]
18        position:               Pos(Num('1'), Num('7'))
19        size:                   Num('3')
20      },
21    Symbol(
22      {
23        type qualifier:         Empty()
24        datatype:               StructSpec(Name('st1'))
25        name:                   Name('st@st2')
26        value or address:       Empty()
27        position:               Pos(Num('3'), Num('23'))
28        size:                   Num('3')
29      },
30    Symbol(
31      {
32        type qualifier:         Empty()
33        datatype:               StructDecl(Name('st2'), [Alloc(Writeable(),
          ↪  StructSpec(Name('st1')), Name('st'))])
34        name:                   Name('st2')
35        value or address:       [Name('st@st2')]
36        position:               Pos(Num('3'), Num('7'))
37        size:                   Num('3')
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  ]
```

Code 1.24: Symboltabelle für Deklaration von Structs

1.4.4.2 Initialisierung von Structs

```c
struct st1 {int *pntr[1];};

struct st2 {struct st1 st;};

void main() {
  int var = 42;
  struct st1 st = {.st={.pntr={{&var}}}};
}
```

Code 1.25: PicoC Code für Initialisierung von Structs

```
File
 Name './example_struct_init.ast',
 [
   StructDecl
     Name 'st1',
     [
       Alloc
         Writeable,
         ArrayDecl
           [
             Num '1'
           ],
           PntrDecl
             Num '1',
             IntType 'int',
         Name 'pntr'
     ],
   StructDecl
     Name 'st2',
     [
       Alloc
         Writeable,
         StructSpec
           Name 'st1',
         Name 'st'
     ],
   FunDef
     VoidType 'void',
     Name 'main',
     [],
     [
       Assign
         Alloc
           Writeable,
           IntType 'int',
           Name 'var',
         Num '42',
       Assign
         Alloc
           Writeable,
           StructSpec
             Name 'st1',
           Name 'st',
         Struct
           [
             Assign
               Name 'st',
               Struct
                 [
                   Assign
                     Name 'pntr',
                     Array
                       [
                         Array
                           [
                             Ref
                               Name 'var'
```

```
                            ]
                        ]
                    ]
                ]
            ]
        ]
    ]
```

Code 1.26: Abstract Syntax Tree für Initialisierung von Structs

```
SymbolTable
  [
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               ArrayDecl([Num('1')], PntrDecl(Num('1'), IntType('int')))
        name:                   Name('pntr@st1')
        value or address:       Empty()
        position:               Pos(Num('1'), Num('17'))
        size:                   Num('1')
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               StructDecl(Name('st1'), [Alloc(Writeable(),
        ↪  ArrayDecl([Num('1')], PntrDecl(Num('1'), IntType('int'))), Name('pntr'))])
        name:                   Name('st1')
        value or address:       [Name('pntr@st1')]
        position:               Pos(Num('1'), Num('7'))
        size:                   Num('1')
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               StructSpec(Name('st1'))
        name:                   Name('st@st2')
        value or address:       Empty()
        position:               Pos(Num('3'), Num('23'))
        size:                   Num('1')
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               StructDecl(Name('st2'), [Alloc(Writeable(),
        ↪  StructSpec(Name('st1')), Name('st'))])
        name:                   Name('st2')
        value or address:       [Name('st@st2')]
        position:               Pos(Num('3'), Num('7'))
        size:                   Num('1')
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               FunDecl(VoidType('void'), Name('main'), [])
        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 1.27: Symboltabelle 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 1.28: PicoC Mon Pass für Initialisierung von Structs

```
File
  Name './example_struct_init.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.29: RETI Blocks Pass für Initialisierung von Structs

1.4.4.3 Zugriff auf Structattribut

```
struct pos {int x; int y;};

void main() {
  struct pos st = {.x=4, .y=2};
  st.y;
}
```

Code 1.30: PicoC Code für Zugriff auf Structattribut

```
File
  Name './example_struct_attr_access.ast',
  [
    StructDecl
      Name 'pos',
      [
        Alloc
          Writeable,
          IntType 'int',
          Name 'x',
        Alloc
          Writeable,
          IntType 'int',
          Name 'y'
```

```
      ],
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            StructSpec
              Name 'pos',
            Name 'st',
          Struct
            [
              Assign
                Name 'x',
                Num '4',
              Assign
                Name 'y',
                Num '2'
            ],
        Exp
          Attr
            Name 'st',
            Name 'y'
      ]
  ]
```

Code 1.31: Abstract Syntax Tree für Zugriff auf Structattribut

```
File
  Name './example_struct_attr_access.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '4',
        Exp
          Num '2',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '2',
        Ref
          GlobalRead
            Num '0',
        Ref
          Attr
            Tmp
              Num '1',
            Name 'y',
        Exp
```

```
25          Subscr
26            Tmp
27              Num '1',
28            Num '0',
29        Return
30          Empty
31      ]
32  ]
```

Code 1.32: 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 1.33: 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 1.34: PicoC Code für Zuweisung an Structattribut

```
File
  Name './example_struct_attr_assignment.ast',
  [
    StructDecl
      Name 'pos',
      [
        Alloc
          Writeable,
          IntType 'int',
          Name 'x',
        Alloc
          Writeable,
          IntType 'int',
          Name 'y'
      ],
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            StructSpec
              Name 'pos',
            Name 'st',
          Struct
            [
              Assign
                Name 'x',
                Num '4',
              Assign
                Name 'y',
                Num '2'
            ],
        Assign
          Attr
            Name 'st',
            Name 'y',
          Num '42'
      ]
  ]
```

Code 1.35: Abstract Syntax Tree für Zuweisung an Structattribut

```
File
  Name './example_struct_attr_assignment.picoc_mon',
  [
    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 1.36: 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 1.37: RETI Blocks Pass für Zuweisung an Structattribut

1.4.5 Umsetzung der Derived Datatypes im Zusammenspiel

1.4.5.1 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 1.38: 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            PntrDecl
39              Num '1',
40              ArrayDecl
41                [
42                  Num '3'
43                ],
44                PntrDecl
45                  Num '1',
46                  IntType 'int',
47            Name 'pntr2',
48        Exp
49          Name 'pntr2'
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            PntrDecl
70              Num '1',
```

```
71              ArrayDecl
72                [
73                  Num '3'
74                ],
75                IntType 'int',
76            Name 'pntr1',
77        Exp
78          Name 'pntr1'
79      ]
80  ]
```

Code 1.39: 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           GlobalRead
9             Num '9',
10        Return
11          Empty
12      ],
13    Block
14      Name 'fun.0',
15      [
16        Exp
17          StackRead
18            Num '9',
19        Return
20          Empty
21      ]
22  ]
```

Code 1.40: 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         SUBI SP 1,
8         LOADIN DS ACC 9,
9         STOREIN SP ACC 1,
10        LOADIN BAF PC -1
11      ],
12    Block
13      Name 'fun.0',
```

```
14      [
15        SUBI SP 1,
16        LOADIN BAF ACC -11,
17        STOREIN SP ACC 1,
18        LOADIN BAF PC -1
19      ]
20    ]
```

Code 1.41: RETI Blocks Pass für den Einleitungsteil

1.4.5.2 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 1.42: 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           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 1.43: 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         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
```

```
            Num '1',
        Ref
          Attr
            Tmp
              Num '1',
            Name 'ar',
        Exp
          Num '0',
        Ref
          Subscr
            Tmp
              Num '2',
            Tmp
              Num '1',
        Exp
          Num '0',
        Ref
          Subscr
            Tmp
              Num '2',
            Tmp
              Num '1',
        Exp
          Subscr
            Tmp
              Num '1',
            Num '0',
        Return
          Empty
      ]
  ]
```

Code 1.44: PicoC Mon Pass für den Mittelteil

```
File
  Name './example_derived_dts_main_part.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        SUBI SP 1,
```

```
        LOADI IN1 1,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        ADDI IN1 0,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 0,
        STOREIN SP ACC 1,
        LOADIN SP IN2 2,
        LOADIN IN2 IN1 0,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 0,
        STOREIN SP ACC 1,
        LOADIN SP IN1 2,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN IN1 ACC 0,
        STOREIN SP ACC 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.45: RETI Blocks Pass für den Mittelteil

1.4.5.3 Schlussteil für die verschiedenen Derived Datatypes

```c
struct st {int attr[2];};

void main() {
  int ar1[1][2] = {{42, 314}};
  struct st ar2[1] = {.attr={42, 314}};
  int var = 42;
  int *pntr1 = &var;
  int **pntr2 = &pntr1;

  ar1[0];
  ar2[0];
  *pntr2;
}
```

Code 1.46: PicoC Code für den Schlussteil

```
File
 Name './example_derived_dts_final_part.ast',
 [
   StructDecl
     Name 'st',
     [
       Alloc
         Writeable,
         ArrayDecl
           [
             Num '2'
           ],
           IntType 'int',
         Name 'attr'
     ],
   FunDef
     VoidType 'void',
     Name 'main',
     [],
     [
       Assign
         Alloc
           Writeable,
           ArrayDecl
             [
               Num '1',
               Num '2'
             ],
             IntType 'int',
           Name 'ar1',
         Array
           [
             Array
               [
                 Num '42',
                 Num '314'
               ]
           ],
       Assign
         Alloc
           Writeable,
           ArrayDecl
             [
               Num '1'
             ],
             StructSpec
               Name 'st',
           Name 'ar2',
         Struct
           [
             Assign
               Name 'attr',
               Array
                 [
                   Num '42',
                   Num '314'
                 ]
```

```
58            ],
59          Assign
60            Alloc
61              Writeable,
62              IntType 'int',
63              Name 'var',
64            Num '42',
65          Assign
66            Alloc
67              Writeable,
68              PntrDecl
69                Num '1',
70                IntType 'int',
71              Name 'pntr1',
72            Ref
73              Name 'var',
74          Assign
75            Alloc
76              Writeable,
77              PntrDecl
78                Num '2',
79                IntType 'int',
80              Name 'pntr2',
81            Ref
82              Name 'pntr1',
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 'pntr2',
94              Num '0'
95        ]
96    ]
```

Code 1.47: 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         Exp
8           Num '42',
9         Exp
10          Num '314',
11        Assign
12          GlobalWrite
```

```
13            Num '0',
14          Tmp
15            Num '2',
16        Exp
17          Num '42',
18        Exp
19          Num '314',
20        Assign
21          GlobalWrite
22            Num '2',
23          Tmp
24            Num '2',
25        Exp
26          Num '42',
27        Assign
28          GlobalWrite
29            Num '4',
30          Tmp
31            Num '1',
32        Ref
33          GlobalRead
34            Num '4',
35        Assign
36          GlobalWrite
37            Num '5',
38          Tmp
39            Num '1',
40        Ref
41          GlobalRead
42            Num '5',
43        Assign
44          GlobalWrite
45            Num '6',
46          Tmp
47            Num '1',
48        Ref
49          GlobalRead
50            Num '0',
51        Exp
52          Num '0',
53        Ref
54          Subscr
55            Tmp
56              Num '2',
57            Tmp
58              Num '1',
59        Exp
60          Subscr
61            Tmp
62              Num '1',
63            Num '0',
64        Ref
65          GlobalRead
66            Num '2',
67        Exp
68          Num '0',
69        Ref
```

```
70          Subscr
71            Tmp
72              Num '2',
73            Tmp
74              Num '1',
75        Exp
76          Subscr
77            Tmp
78              Num '1',
79            Num '0',
80        Ref
81          GlobalRead
82            Num '6',
83        Exp
84          Num '0',
85        Ref
86          Subscr
87            Tmp
88              Num '2',
89            Tmp
90              Num '1',
91        Exp
92          Subscr
93            Tmp
94              Num '1',
95            Num '0',
96        Return
97          Empty
98      ]
99  ]
```

Code 1.48: 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         SUBI SP 1,
8         LOADI ACC 42,
9         STOREIN SP ACC 1,
10        SUBI SP 1,
11        LOADI ACC 314,
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 ACC 314,
23          STOREIN SP ACC 1,
24          LOADIN SP ACC 1,
25          STOREIN DS ACC 3,
26          LOADIN SP ACC 2,
27          STOREIN DS ACC 2,
28          ADDI SP 2,
29          SUBI SP 1,
30          LOADI ACC 42,
31          STOREIN SP ACC 1,
32          LOADIN SP ACC 1,
33          STOREIN DS ACC 4,
34          ADDI SP 1,
35          SUBI SP 1,
36          LOADI IN1 4,
37          ADD IN1 DS,
38          STOREIN SP IN1 1,
39          LOADIN SP ACC 1,
40          STOREIN DS ACC 5,
41          ADDI SP 1,
42          SUBI SP 1,
43          LOADI IN1 5,
44          ADD IN1 DS,
45          STOREIN SP IN1 1,
46          LOADIN SP ACC 1,
47          STOREIN DS ACC 6,
48          ADDI SP 1,
49          SUBI SP 1,
50          LOADI IN1 0,
51          ADD IN1 DS,
52          STOREIN SP IN1 1,
53          SUBI SP 1,
54          LOADI ACC 0,
55          STOREIN SP ACC 1,
56          LOADIN SP IN1 2,
57          LOADIN SP IN2 1,
58          MULTI IN2 2,
59          ADD IN1 IN2,
60          ADDI SP 1,
61          STOREIN SP IN1 1,
62          SUBI SP 1,
63          LOADI IN1 2,
64          ADD IN1 DS,
65          STOREIN SP IN1 1,
66          SUBI SP 1,
67          LOADI ACC 0,
68          STOREIN SP ACC 1,
69          LOADIN SP IN1 2,
70          LOADIN SP IN2 1,
71          MULTI IN2 2,
72          ADD IN1 IN2,
73          ADDI SP 1,
74          STOREIN SP IN1 1,
75          LOADIN SP IN1 1,
76          LOADIN IN1 ACC 0,
77          STOREIN SP ACC 1,
78          SUBI SP 1,
```

```
79          LOADI IN1 6,
80          ADD IN1 DS,
81          STOREIN SP IN1 1,
82          SUBI SP 1,
83          LOADI ACC 0,
84          STOREIN SP ACC 1,
85          LOADIN SP IN2 2,
86          LOADIN IN2 IN1 0,
87          LOADIN SP IN2 1,
88          MULTI IN2 1,
89          ADD IN1 IN2,
90          ADDI SP 1,
91          STOREIN SP IN1 1,
92          LOADIN BAF PC -1
93        ]
94    ]
```

Code 1.49: RETI Blocks Pass für den Schlussteil

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 1.50: 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 1.51: 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 1.52: PicoC Blocks Pass für 3 Funktionen

```
File
  Name './example_3_funs.picoc_mon',
  [
    Block
      Name 'main.2',
      [
        Return
          Empty
      ],
    Block
      Name 'fun1.1',
      [
        Return
          Empty
      ],
    Block
      Name 'fun2.0',
      [
        Exp
          Num '1',
        Return
          Tmp
            Num '1'
      ]
  ]
```

Code 1.53: PicoC Mon Pass für 3 Funktionen

```
File
  Name './example_3_funs.reti_blocks',
  [
    Block
      Name 'main.2',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun1.1',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun2.0',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.54: 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 1.55: 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 1.56: 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
```

```
      ],
    Block
      Name 'fun2.1',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
      [
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.57: PicoC Blocks Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

```
File
  Name './example_3_funs_main.reti_patch',
  [
    Block
      Name 'start.3',
      [
        Exp
          GoTo
            Name 'main.0'
      ],
    Block
      Name 'fun1.2',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun2.1',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
      [
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.58: 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    int var = fun2(42);
8    return;
9  }
10
11 int fun2(int var) {
12   return var;
13 }
```

Code 1.59: 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(),
           ↪  IntType('int'), Name('var'))])
7          name:                   Name('fun2')
8          value or address:       Empty()
9          position:               Pos(Num('1'), Num('4'))
10         size:                   Empty()
11       },
12     Symbol(
13       {
14         type qualifier:         Empty()
15         datatype:               FunDecl(VoidType('void'), Name('fun1'), [])
16         name:                   Name('fun1')
17         value or address:       Empty()
18         position:               Pos(Num('3'), Num('5'))
19         size:                   Empty()
20       },
21     Symbol(
22       {
23         type qualifier:         Empty()
24         datatype:               FunDecl(VoidType('void'), Name('main'), [])
25         name:                   Name('main')
26         value or address:       Empty()
27         position:               Pos(Num('6'), Num('5'))
28         size:                   Empty()
29       },
30     Symbol(
31       {
32         type qualifier:         Writeable()
33         datatype:               IntType('int')
34         name:                   Name('var@main')
35         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 1.60: Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss

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 1.61: 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 1.62: 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 1.63: RETI Blocks Pass für Funktionsaufruf ohne Rückgabewert

```
SUBI SP 2;
SUBI SP 1;
LOADI IN1 0;
ADD IN1 DS;
STOREIN SP IN1 1;
MOVE BAF ACC;
ADDI SP 3;
MOVE SP BAF;
SUBI SP 4;
STOREIN BAF ACC 0;
LOADI ACC 14;
ADD ACC CS;
STOREIN BAF ACC -1;
JUMP 5;
MOVE BAF IN1;
LOADIN IN1 BAF 0;
MOVE IN1 SP;
LOADIN BAF PC -1;
LOADIN BAF PC -1;
```

Code 1.64: RETI Pass für Funktionsaufruf ohne Rückgabewert

1.4.6.3.2 Mit Rückgabewert

```
void stack_fun() {
  return 42;
}

void main() {
  int var = stack_fun();
}
```

Code 1.65: PicoC Code für Funktionsaufruf mit Rückgabewert

```
File
  Name './example_fun_call_with_return_value.picoc_mon',
  [
    Block
      Name 'stack_fun.1',
      [
        Exp
          Num '42',
        Return
          Tmp
            Num '1'
      ],
    Block
      Name 'main.0',
      [
```

```
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 1.66: 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 1.67: 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 1.68: RETI Pass für Funktionsaufruf mit Rückgabewert

1.4.6.3.3 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 1.69: PicoC Code für Call by Sharing für Arrays

```
File
  Name './example_fun_call_by_sharing_array.picoc_mon',
  [
    Block
      Name 'stack_fun.1',
      [
        Return
          Empty
      ],
    Block
      Name 'main.0',
      [
        StackMalloc
          Num '2',
        Ref
          GlobalRead
            Num '0',
        Ref
          GlobalRead
            Num '6',
        NewStackframe
          Name 'stack_fun',
          GoTo
            Name 'addr@next_instr',
        Exp
          GoTo
            Name 'stack_fun.1',
        RemoveStackframe,
        Return
          Empty
      ]
  ]
```

Code 1.70: PicoC Mon Pass für Call by Sharing für Arrays

```
SymbolTable
  [
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               FunDecl(VoidType('void'), Name('stack_fun'),
        ↪  [Alloc(Writeable(), PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int'))),
        ↪  Name('param1')), Alloc(Writeable(), ArrayDecl([Num('3')], IntType('int')),
        ↪  Name('param2'))])
        name:                   Name('stack_fun')
        value or address:       Empty()
        position:               Pos(Num('1'), Num('5'))
        size:                   Empty()
      },
    Symbol(
      {
        type qualifier:         Writeable()
        datatype:               PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
        name:                   Name('param1@stack_fun')
```

```
      value or address:       Num('0')
      position:               Pos(Num('1'), Num('21'))
      size:                   Num('1')
    },
  Symbol(
    {
      type qualifier:         Writeable()
      datatype:               PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
      name:                   Name('param2@stack_fun')
      value or address:       Num('1')
      position:               Pos(Num('1'), Num('37'))
      size:                   Num('1')
    },
  Symbol(
    {
      type qualifier:         Empty()
      datatype:               FunDecl(VoidType('void'), Name('main'), [])
      name:                   Name('main')
      value or address:       Empty()
      position:               Pos(Num('4'), Num('5'))
      size:                   Empty()
    },
  Symbol(
    {
      type qualifier:         Writeable()
      datatype:               ArrayDecl([Num('2'), Num('3')], IntType('int'))
      name:                   Name('local_var1@main')
      value or address:       Num('0')
      position:               Pos(Num('5'), Num('6'))
      size:                   Num('6')
    },
  Symbol(
    {
      type qualifier:         Writeable()
      datatype:               ArrayDecl([Num('2'), Num('3')], IntType('int'))
      name:                   Name('local_var2@main')
      value or address:       Num('6')
      position:               Pos(Num('6'), Num('6'))
      size:                   Num('6')
    }
]
```

Code 1.71: Symboltabelle für Call by Sharing für Arrays

```
File
  Name './example_fun_call_by_sharing_array.reti_blocks',
  [
    Block
      Name 'stack_fun.1',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
```

```
      [
        SUBI SP 2,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI IN1 6,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        MOVE BAF ACC,
        ADDI SP 4,
        MOVE SP BAF,
        SUBI SP 4,
        STOREIN BAF ACC 0,
        LOADI ACC GoTo
                    Name 'addr@next_instr',
        ADD ACC CS,
        STOREIN BAF ACC -1,
        Exp
          GoTo
            Name 'stack_fun.1',
        MOVE BAF IN1,
        LOADIN IN1 BAF 0,
        MOVE IN1 SP,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.72: RETI Block Pass für Call by Sharing für Arrays

1.4.6.3.4 Umsetzung von Call by Value für Structs

```
struct st {int attr1; int attr2[2];};

void stack_fun(struct st param) {
}

void main() {
  struct st local_var;
  stack_fun(local_var);
}
```

Code 1.73: PicoC Code für Call by Value für Structs

```
File
  Name './example_fun_call_by_value_struct.picoc_mon',
  [
    Block
      Name 'stack_fun.1',
      [
```

```
      Return
        Empty
    ],
  Block
    Name 'main.0',
    [
      StackMalloc
        Num '2',
      Assign
        Tmp
          Num '3',
        GlobalRead
          Num '0',
      NewStackframe
        Name 'stack_fun',
        GoTo
          Name 'addr@next_instr',
      Exp
        GoTo
          Name 'stack_fun.1',
      RemoveStackframe,
      Return
        Empty
    ]
]
```

Code 1.74: PicoC Mon Pass für Call by Value für Structs

```
File
  Name './example_fun_call_by_value_struct.reti_blocks',
  [
    Block
      Name 'stack_fun.1',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
      [
        SUBI SP 2,
        SUBI SP 3,
        LOADIN DS ACC 0,
        STOREIN SP ACC 1,
        LOADIN DS ACC 1,
        STOREIN SP ACC 2,
        LOADIN DS ACC 2,
        STOREIN SP ACC 3,
        MOVE BAF ACC,
        ADDI SP 5,
        MOVE SP BAF,
        SUBI SP 5,
        STOREIN BAF ACC 0,
        LOADI ACC GoTo
                    Name 'addr@next_instr',
```

```
27          ADD ACC CS,
28          STOREIN BAF ACC -1,
29          Exp
30            GoTo
31              Name 'stack_fun.1',
32          MOVE BAF IN1,
33          LOADIN IN1 BAF 0,
34          MOVE IN1 SP,
35          LOADIN BAF PC -1
36        ]
37    ]
```

Code 1.75: RETI Block Pass für 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 (besucht am 27. 04. 2022).