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

0.0.1 Umsetzung von Pointern

0.0.1.1 Referenzierung

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

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

Code 1: PicoC Code für Pointer Referenzierung

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

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

Code 2: Abstract Syntax Tree für Pointer Referenzierung

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

```
File
  Name './example_pntr_ref.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Name('var'), Num('42')),
        Exp
          Num '42',
        Assign
          Global
            Num '0',
          Stack
            Num '1',
        // Assign(Name('pntr'), Ref(Name('var'))),
        Ref
          Global
            Num '0',
        Assign
          Global
            Num '1',
          Stack
            Num '1',
        Return
          Empty
      ]
  ]
```

Code 3: PicoC Mon Pass für Pointer Referenzierung

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

```
File
  Name './example_pntr_ref.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Name('var'), Num('42')),
        # Exp(Num('42')),
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        # Assign(Global(Num('0')), Stack(Num('1'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        # // Assign(Name('pntr'), Ref(Name('var'))),
        # Ref(Global(Num('0'))),
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        # Assign(Global(Num('1')), Stack(Num('1'))),
```

```
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        # Return(Empty()),
        LOADIN BAF PC -1
      ]
  ]
```

Code 4: RETI Blocks Pass für Pointer Referenzierung

0.0.1.2 Pointer Dereferenzierung durch Zugriff auf Arrayindex ersetzen

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

Code 5: 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,
            PntrDecl
              Num '1',
              IntType 'int',
            Name 'pntr',
          Ref
            Name 'var',
        Exp
          Deref
            Name 'pntr',
            Num '0'
      ]
  ]
```

Code 6: Abstract Syntax Tree für Pointer Dereferenzierung

```
File
  Name './example_pntr_deref.picoc_shrink',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            IntType 'int',
            Name 'var',
          Num '42',
        Assign
          Alloc
            Writeable,
            PntrDecl
              Num '1',
              IntType 'int',
            Name 'pntr',
          Ref
            Name 'var',
        Exp
          Subscr
            Name 'pntr',
            Num '0'
      ]
  ]
```

Code 7: PicoC Shrink Pass für Pointer Dereferenzierung

0.0.2 Umsetzung von Arrays

0.0.2.1 Initialisierung von Arrays

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

Code 8: PicoC Code für Array Initialisierung

```
File
  Name './example_array_init.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
```

```
    [
      Assign
        Alloc
          Writeable,
          ArrayDecl
            [
              Num '2',
              Num '1'
            ],
            IntType 'int',
          Name 'ar',
        Array
          [
            Array
              [
                Num '4'
              ],
            Array
              [
                Num '2'
              ]
          ]
    ]
]
```

Code 9: Abstract Syntax Tree für Array Initialisierung

```
SymbolTable
  [
    Symbol(
      {
        type qualifier:         Empty()
        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 10: Symboltabelle für Array Initialisierung

```
File
  Name './example_array_init.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Name('ar'), Array([Array([Num('4')]), Array([Num('2')])])),
        Exp
          Num '4',
        Exp
          Num '2',
        Assign
          Global
            Num '0',
          Stack
            Num '2',
        Return
          Empty
      ]
  ]
```

Code 11: PicoC Mon Pass für Array Initialisierung

```
File
  Name './example_array_init.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Name('ar'), Array([Array([Num('4')]), Array([Num('2')])])),
        # Exp(Num('4')),
        SUBI SP 1,
        LOADI ACC 4,
        STOREIN SP ACC 1,
        # Exp(Num('2')),
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        # Assign(Global(Num('0')), Stack(Num('2'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        # Return(Empty()),
        LOADIN BAF PC -1
      ]
  ]
```

Code 12: RETI Blocks Pass für Array Initialisierung

0.0.2.2 Zugriff auf Arrayindex

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

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

Code 13: PicoC Code für Zugriff auf Arrayindex

```
File
  Name './example_array_access.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            ArrayDecl
              [
                Num '2'
              ],
              IntType 'int',
            Name 'ar',
          Array
            [
              Num '1',
              Num '2'
            ],
        Exp
          Subscr
            Name 'ar',
            Num '2'
      ]
  ]
```

Code 14: Abstract Syntax Tree für Zugriff auf Arrayindex

```
File
  Name './example_array_access.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Name('ar'), Array([Num('1'), Num('2')])),
        Exp
          Num '1',
        Exp
          Num '2',
```

```
      Assign
        Global
          Num '0',
        Stack
          Num '2',
      // Exp(Subscr(Name('ar'), Num('2'))),
      Ref
        Global
          Num '0',
      Exp
        Num '2',
      Ref
        Subscr
          Stack
            Num '2',
          Stack
            Num '1',
      Exp
        Stack
          Num '1',
      Return
        Empty
    ]
]
```

Code 15: PicoC Mon Pass für Zugriff auf Arrayindex

```
File
  Name './example_array_access.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Name('ar'), Array([Num('1'), Num('2')])),
        # Exp(Num('1')),
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        # Exp(Num('2')),
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        # Assign(Global(Num('0')), Stack(Num('2'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        # // Exp(Subscr(Name('ar'), Num('2'))),
        # Ref(Global(Num('0'))),
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
```

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

Code 16: RETI Blocks Pass für Zugriff auf Arrayindex

0.0.2.3 Zuweisung an Arrayindex

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

Code 17: PicoC Code für Zuweisung an Arrayindex

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

```
23      ]
24  ]
```

Code 18: Abstract Syntax Tree für Zuweisung an Arrayindex

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

Code 19: PicoC Mon Pass für Zuweisung an Arrayindex

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

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

Code 20: RETI Blocks Pass für Zuweisung an Arrayindex

0.0.3 Umsetzung von Structs

0.0.3.1 Deklaration von Structs

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

Code 21: PicoC Code für Deklaration von Structs

```
1  SymbolTable
2    [
3      Symbol(
4        {
5          type qualifier:          Empty()
6          datatype:                ArrayDecl([Num('3')], PntrDecl(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(),
          ↪  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 22: Symboltabelle für Deklaration von Structs

0.0.3.2 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 23: 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 24: 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')
```

```
      value or address:       Empty()
      position:               Pos(Num('5'), Num('5'))
      size:                   Empty()
    },
  Symbol(
    {
      type qualifier:         Writeable()
      datatype:               IntType('int')
      name:                   Name('var@main')
      value or address:       Num('0')
      position:               Pos(Num('6'), Num('6'))
      size:                   Num('1')
    },
  Symbol(
    {
      type qualifier:         Writeable()
      datatype:               StructSpec(Name('st1'))
      name:                   Name('st@main')
      value or address:       Num('1')
      position:               Pos(Num('7'), Num('13'))
      size:                   Num('1')
    }
]
```

Code 25: Symboltabelle für Initialisierung von Structs

```
File
  Name './example_struct_init.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Name('var'), Num('42')),
        Exp
          Num '42',
        Assign
          Global
            Num '0',
          Stack
            Num '1',
        // Assign(Name('st'), Struct([Assign(Name('st'), Struct([Assign(Name('pntr'),
        ↪  Array([Array([Ref(Name('var'))])]))]))])),
        Ref
          Global
            Num '0',
        Assign
          Global
            Num '1',
          Stack
            Num '1',
        Return
          Empty
      ]
  ]
```

Code 26: PicoC Mon Pass für Initialisierung von Structs

```
File
  Name './example_struct_init.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Name('var'), Num('42')),
        # Exp(Num('42')),
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        # Assign(Global(Num('0')), Stack(Num('1'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        # // Assign(Name('st'), Struct([Assign(Name('st'), Struct([Assign(Name('pntr'),
        ↪  Array([Array([Ref(Name('var'))])]))]))])),
        # Ref(Global(Num('0'))),
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        # Assign(Global(Num('1')), Stack(Num('1'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        # Return(Empty()),
        LOADIN BAF PC -1
      ]
  ]
```

Code 27: RETI Blocks Pass für Initialisierung von Structs

0.0.3.3 Zugriff auf Structattribut

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

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

Code 28: 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 29: Abstract Syntax Tree für Zugriff auf Structattribut

```
File
  Name './example_struct_attr_access.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Name('st'), Struct([Assign(Name('x'), Num('4')), Assign(Name('y'),
        ↪  Num('2'))])),
        Exp
          Num '4',
        Exp
          Num '2',
        Assign
```

```
          Global
            Num '0',
          Stack
            Num '2',
        Ref
          Global
            Num '0',
        Ref
          Attr
            Stack
              Num '1',
            Name 'y',
        Exp
          Stack
            Num '1',
        Return
          Empty
      ]
  ]
```

Code 30: PicoC Mon Pass für Zugriff auf Structattribut

```
File
  Name './example_struct_attr_access.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Name('st'), Struct([Assign(Name('x'), Num('4')), Assign(Name('y'),
        ↪  Num('2'))])),
        # Exp(Num('4')),
        SUBI SP 1,
        LOADI ACC 4,
        STOREIN SP ACC 1,
        # Exp(Num('2')),
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        # Assign(Global(Num('0')), Stack(Num('2'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        # Ref(Global(Num('0'))),
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        # Ref(Attr(Stack(Num('1')), Name('y'))),
        LOADIN SP IN1 1,
        ADDI IN1 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
```

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

Code 31: RETI Blocks Pass für Zugriff auf Structattribut

0.0.3.4 Zuweisung an Structattribut

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

Code 32: PicoC Code für Zuweisung an Structattribut

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

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

Code 33: Abstract Syntax Tree für Zuweisung an Structattribut

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

Code 34: PicoC Mon Pass für Zuweisung an Structattribut

```
File
  Name './example_struct_attr_assignment.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Name('st'), Struct([Assign(Name('x'), Num('4')), Assign(Name('y'),
        ↪  Num('2'))])),
        # Exp(Num('4')),
        SUBI SP 1,
        LOADI ACC 4,
        STOREIN SP ACC 1,
        # Exp(Num('2')),
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        # Assign(Global(Num('0')), Stack(Num('2'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        # // Assign(Attr(Name('st'), Name('y')), Num('42')),
        # Exp(Num('42')),
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        # Ref(Global(Num('0'))),
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        # Ref(Attr(Stack(Num('1')), Name('y'))),
        LOADIN SP IN1 1,
        ADDI IN1 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN SP ACC 2,
        ADDI SP 2,
        STOREIN IN1 ACC 0,
        # Return(Empty()),
        LOADIN BAF PC -1
      ]
  ]
```

Code 35: RETI Blocks Pass für Zuweisung an Structattribut

0.0.4 Umsetzung der Derived Datatypes im Zusammenspiel

0.0.4.1 Einleitungsteil für Globale Statische Daten und Stackframe

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

Code 36: PicoC Code für den Einleitungsteil

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

```
35        Exp
36          Alloc
37            Writeable,
38            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 37: Abstract Syntax Tree für den Einleitungsteil

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

```
      [
        Exp
          Global
            Num '9',
        Return
          Empty
      ],
    Block
      Name 'fun.0',
      [
        Exp
          Stackframe
            Num '9',
        Return
          Empty
      ]
  ]
```

Code 38: PicoC Mon Pass für den Einleitungsteil

```
File
  Name './example_derived_dts_introduction_part.reti_blocks',
  [
    Block
      Name 'main.1',
      [
        # Exp(Global(Num('9'))),
        SUBI SP 1,
        LOADIN DS ACC 9,
        STOREIN SP ACC 1,
        # Return(Empty()),
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun.0',
      [
        # Exp(Stackframe(Num('9'))),
        SUBI SP 1,
        LOADIN BAF ACC -11,
        STOREIN SP ACC 1,
        # Return(Empty()),
        LOADIN BAF PC -1
      ]
  ]
```

Code 39: RETI Blocks Pass für den Einleitungsteil

0.0.4.2 Mittelteil für die verschiedenen Derived Datatypes

```
struct st1 {int (*ar)[1];};

void main() {
  int var[1] = {42};
  struct st1 st_first = {.ar=&var};
  (*st_first.ar)[0];
}
```

Code 40: PicoC Code für den Mittelteil

```
File
  Name './example_derived_dts_main_part.ast',
  [
    StructDecl
      Name 'st1',
      [
        Alloc
          Writeable,
          PntrDecl
            Num '1',
            ArrayDecl
              [
                Num '1'
              ],
              IntType 'int',
          Name 'ar'
      ],
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            ArrayDecl
              [
                Num '1'
              ],
              IntType 'int',
            Name 'var',
          Array
            [
              Num '42'
            ],
        Assign
          Alloc
            Writeable,
            StructSpec
              Name 'st1',
            Name 'st_first',
          Struct
```

```
            [
              Assign
                Name 'ar',
                Ref
                  Name 'var'
            ],
        Exp
          Subscr
            Deref
              Attr
                Name 'st_first',
                Name 'ar',
              Num '0',
            Num '0'
      ]
  ]
```

Code 41: Abstract Syntax Tree für den Mittelteil

```
File
  Name './example_derived_dts_main_part.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Name('var'), Array([Num('42')])),
        Exp
          Num '42',
        Assign
          Global
            Num '0',
          Stack
            Num '1',
        // Assign(Name('st_first'), Struct([Assign(Name('ar'), Ref(Name('var')))])),
        Ref
          Global
            Num '0',
        Assign
          Global
            Num '1',
          Stack
            Num '1',
        // Exp(Subscr(Subscr(Attr(Name('st_first'), Name('ar')), Num('0')), Num('0'))),
        Ref
          Global
            Num '1',
        Ref
          Attr
            Stack
              Num '1',
            Name 'ar',
        Exp
          Num '0',
        Ref
```

```
          Subscr
            Stack
              Num '2',
            Stack
              Num '1',
        Exp
          Num '0',
        Ref
          Subscr
            Stack
              Num '2',
            Stack
              Num '1',
        Exp
          Stack
            Num '1',
        Return
          Empty
      ]
  ]
```

Code 42: PicoC Mon Pass für den Mittelteil

```
File
  Name './example_derived_dts_main_part.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Name('var'), Array([Num('42')])),
        # Exp(Num('42')),
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        # Assign(Global(Num('0')), Stack(Num('1'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        # // Assign(Name('st_first'), Struct([Assign(Name('ar'), Ref(Name('var')))])),
        # Ref(Global(Num('0'))),
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        # Assign(Global(Num('1')), Stack(Num('1'))),
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        # // Exp(Subscr(Subscr(Attr(Name('st_first'), Name('ar')), Num('0')), Num('0'))),
        # Ref(Global(Num('1'))),
        SUBI SP 1,
        LOADI IN1 1,
        ADD IN1 DS,
        STOREIN SP IN1 1,
```

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

Code 43: RETI Blocks Pass für den Mittelteil

0.0.4.3 Schlussteil für die verschiedenen Derived Datatypes

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

Code 44: PicoC Code für den Schlussteil

```
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'
```

```
                ]
            ],
        Assign
          Alloc
            Writeable,
            IntType 'int',
            Name 'var',
          Num '42',
        Assign
          Alloc
            Writeable,
            PntrDecl
              Num '1',
              IntType 'int',
            Name 'pntr1',
          Ref
            Name 'var',
        Assign
          Alloc
            Writeable,
            PntrDecl
              Num '2',
              IntType 'int',
            Name 'pntr2',
          Ref
            Name 'pntr1',
        Exp
          Subscr
            Name 'ar1',
            Num '0',
        Exp
          Subscr
            Name 'ar2',
            Num '0',
        Exp
          Deref
            Name 'pntr2',
            Num '0'
      ]
  ]
```

Code 45: Abstract Syntax Tree für den Schlussteil

```
File
  Name './example_derived_dts_final_part.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Name('ar1'), Array([Array([Num('42'), Num('314')])])),
        Exp
          Num '42',
        Exp
          Num '314',
```

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

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

Code 46: PicoC Mon Pass für den Schlussteil

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

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

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

Code 47: RETI Blocks Pass für den Schlussteil

0.0.5 Umsetzung von Funktionen

0.0.5.1 Funktionen auflösen zu RETI Code

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

Code 48: PicoC Code für 3 Funktionen

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

Code 49: Abstract Syntax Tree für 3 Funktionen

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

```
      Name 'main',
      [],
      [
        Block
          Name 'main.2',
          [
            Return
              Empty
          ]
      ],
    FunDef
      VoidType 'void',
      Name 'fun1',
      [],
      [
        Block
          Name 'fun1.1',
          []
      ],
    FunDef
      IntType 'int',
      Name 'fun2',
      [],
      [
        Block
          Name 'fun2.0',
          [
            Return
              Num '1'
          ]
      ]
  ]
```

Code 50: 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',
      [
        // Return(Num('1')),
```

```
      Exp
        Num '1',
      Return
        Stack
          Num '1'
    ]
]
```

Code 51: PicoC Mon Pass für 3 Funktionen

```
File
  Name './example_3_funs.reti_blocks',
  [
    Block
      Name 'main.2',
      [
        # Return(Empty()),
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun1.1',
      [
        # Return(Empty()),
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun2.0',
      [
        # // Return(Num('1')),
        # Exp(Num('1')),
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        # Return(Stack(Num('1'))),
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 52: RETI Blocks Pass für 3 Funktionen

0.0.5.1.1 Sprung zur Main Funktion

```c
void fun1() {
}

int fun2() {
  return 1;
}

```

```
8  void main() {
9    return;
10 }
```

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

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

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

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

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

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

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

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

0.0.5.2 Funktionsdeklaration und -definition

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

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

```
1  SymbolTable
2    [
3      Symbol(
4        {
5          type qualifier:          Empty()
6          datatype:                FunDecl(IntType('int'), Name('fun2'), [Alloc(Writeable(),
           ↪  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 58: Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss

0.0.5.3 Funktionsaufruf

0.0.5.3.1 Ohne Rückgabewert

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

Code 59: PicoC Code für Funktionsaufruf ohne Rückgabewert

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

```
      Exp
        GoTo
          Name 'stack_fun.0',
      RemoveStackframe,
      Return
        Empty
    ],
  Block
    Name 'stack_fun.0',
    [
      Return
        Empty
    ]
]
```

Code 60: PicoC Mon Pass für Funktionsaufruf ohne Rückgabewert

```
File
  Name './example_fun_call_no_return_value.reti_blocks',
  [
    Block
      Name 'main.1',
      [
        # StackMalloc(Num('2')),
        SUBI SP 2,
        # Ref(Global(Num('0'))),
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr'))),
        MOVE BAF ACC,
        ADDI SP 3,
        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.0',
        # RemoveStackframe(),
        MOVE BAF IN1,
        LOADIN IN1 BAF 0,
        MOVE IN1 SP,
        # Return(Empty()),
        LOADIN BAF PC -1
      ],
    Block
      Name 'stack_fun.0',
      [
        # Return(Empty()),
```

```
38          LOADIN BAF PC -1
39        ]
40    ]
```

Code 61: RETI Blocks Pass für Funktionsaufruf ohne Rückgabewert

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

Code 62: RETI Pass für Funktionsaufruf ohne Rückgabewert

0.0.5.3.2 Mit Rückgabewert

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

Code 63: PicoC Code für Funktionsaufruf mit Rückgabewert

```
File
  Name './example_fun_call_with_return_value.picoc_mon',
  [
    Block
      Name 'stack_fun.1',
      [
        // Return(Num('42')),
        Exp
          Num '42',
        Return
          Stack
            Num '1'
      ],
    Block
      Name 'main.0',
      [
        // Assign(Name('var'), Call(Name('stack_fun'), [])),
        StackMalloc
          Num '2',
        NewStackframe
          Name 'stack_fun',
          GoTo
            Name 'addr@next_instr',
        Exp
          GoTo
            Name 'stack_fun.1',
        RemoveStackframe,
        Assign
          Global
            Num '0',
          Stack
            Num '1',
        Return
          Empty
      ]
  ]
```

Code 64: PicoC Mon Pass für Funktionsaufruf mit Rückgabewert

```
File
  Name './example_fun_call_with_return_value.reti_blocks',
  [
    Block
      Name 'stack_fun.1',
      [
        # // Return(Num('42')),
        # Exp(Num('42')),
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        # Return(Stack(Num('1'))),
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
```

```
    ],
  Block
    Name 'main.0',
    [
      # // Assign(Name('var'), Call(Name('stack_fun'), [])),
      # StackMalloc(Num('2')),
      SUBI SP 2,
      # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr'))),
      MOVE BAF ACC,
      ADDI SP 2,
      MOVE SP BAF,
      SUBI SP 2,
      STOREIN BAF ACC 0,
      LOADI ACC GoTo
                  Name 'addr@next_instr',
      ADD ACC CS,
      STOREIN BAF ACC -1,
      Exp
        GoTo
          Name 'stack_fun.1',
      # RemoveStackframe(),
      MOVE BAF IN1,
      LOADIN IN1 BAF 0,
      MOVE IN1 SP,
      # Assign(Global(Num('0')), Stack(Num('1'))),
      LOADIN SP ACC 1,
      STOREIN DS ACC 0,
      ADDI SP 1,
      # Return(Empty()),
      LOADIN BAF PC -1
    ]
]
```

Code 65: RETI Blocks Pass für Funktionsaufruf mit Rückgabewert

```
JUMP 7;
# // Return(Num('42'))
# Exp(Num('42'))
SUBI SP 1;
LOADI ACC 42;
STOREIN SP ACC 1;
# Return(Stack(Num('1')))
LOADIN SP ACC 1;
ADDI SP 1;
LOADIN BAF PC -1;
# // Assign(Name('var'), Call(Name('stack_fun'), []))
# StackMalloc(Num('2'))
SUBI SP 2;
# NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
MOVE BAF ACC;
ADDI SP 2;
MOVE SP BAF;
SUBI SP 2;
STOREIN BAF ACC 0;
```

```
LOADI ACC 17;
ADD ACC CS;
STOREIN BAF ACC -1;
JUMP -15;
# RemoveStackframe()
MOVE BAF IN1;
LOADIN IN1 BAF 0;
MOVE IN1 SP;
# Assign(Global(Num('0')), Stack(Num('1')))
LOADIN SP ACC 1;
STOREIN DS ACC 0;
ADDI SP 1;
# Return(Empty())
LOADIN BAF PC -1;
```

Code 66: RETI Pass für Funktionsaufruf mit Rückgabewert

0.0.5.3.3 Umsetzung von Call by Sharing für Arrays

```
void stack_fun(int (*param1)[3], int param2[2][3]) {
}

void main() {
  int local_var1[2][3];
  int local_var2[2][3];
  stack_fun(local_var1, local_var2);
}
```

Code 67: 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
          Global
            Num '0',
        Ref
          Global
            Num '6',
        NewStackframe
```

```
        Name 'stack_fun',
        GoTo
          Name 'addr@next_instr',
      Exp
        GoTo
          Name 'stack_fun.1',
      RemoveStackframe,
      Return
        Empty
    ]
]
```

Code 68: 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()
```

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

Code 69: Symboltabelle für Call by Sharing für Arrays

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

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

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

0.0.5.3.4 Umsetzung von Call by Value für Structs

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

Code 71: PicoC Code für Call by Value für Structs

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

```
          Name 'stack_fun',
          GoTo
            Name 'addr@next_instr',
        Exp
          GoTo
            Name 'stack_fun.1',
        RemoveStackframe,
        Return
          Empty
      ]
  ]
```

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

```
File
  Name './example_fun_call_by_value_struct.reti_blocks',
  [
    Block
      Name 'stack_fun.1',
      [
        # Return(Empty()),
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
      [
        # StackMalloc(Num('2')),
        SUBI SP 2,
        # Assign(Stack(Num('3')), Global(Num('0'))),
        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,
        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr'))),
        MOVE BAF ACC,
        ADDI SP 5,
        MOVE SP BAF,
        SUBI SP 5,
        STOREIN BAF ACC 0,
        LOADI ACC GoTo
                    Name 'addr@next_instr',
        ADD ACC CS,
        STOREIN BAF ACC -1,
        Exp
          GoTo
            Name 'stack_fun.1',
        # RemoveStackframe(),
        MOVE BAF IN1,
        LOADIN IN1 BAF 0,
        MOVE IN1 SP,
        # Return(Empty()),
```

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

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

0.0.6 Umsetzung kleinerer Details

0.1 Fehlermeldungen

0.1.1 Error Handler

0.1.2 Arten von Fehlermeldungen

0.1.2.1 Syntaxfehler

0.1.2.2 Laufzeitfehler

Literatur