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 Knoten `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(Alloc(Writeable(), IntType('int'), Name('var')), Num('42'))
10         Assign(Alloc(Writeable(), PntrDecl(Num('1'), IntType('int')), Name('pntr')),
           ↪  Ref(Name('var')))
11       ]
12   ]
```

Code 2: Abstract Syntax Tree für Pointer Referenzierung

Im **PicoC-Mon Pass** in Code 3 wird der Knoten `Ref(Name('var'))` durch die Knoten `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` z.B. ein `Subscr(Attr(Name('var')))` ist, sind noch weitere Anweisungen zwischen den Zeilen `11` und `12` nötig, die sich in diesem Beispiel um das Übersetzen von `Subscr(exp)` und `Attr(exp)` nach dem Schema in Subkapitel 0.0.4.2 kümmern.

```
1  File
2    Name './example_pntr_ref.picoc_mon',
3    [
4      Block
5        Name 'main.0',
6        [
7          // Assign(Name('var'), Num('42'))
8          Exp(Num('42'))
9          Assign(Global(Num('0')), Stack(Num('1')))
10         // Assign(Name('pntr'), Ref(Name('var')))
11         Ref(Global(Num('0')))
```

```
12        Assign(Global(Num('1')), Stack(Num('1')))
13        Return(Empty())
14      ]
15  ]
```

Code 3: PicoC Mon Pass für Pointer Referenzierung

Im **PicoC-Blocks Pass** in Code 4 werden die **PicoC-Knoten** `Ref(Global(Num('0')))` und `Assign(Global(Num('1')), Stack(Num('1')))` durch ihre entsprechenden **RETI-Knoten** ersetzt.

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

Code 4: RETI Blocks Pass für Pointer Referenzierung

0.0.1.2 Dereferenzierung durch Zugriff auf Arrayindex ersetzen

Die **Dereferenzierung** `*var` wird im Folgenden anhand des Beispiels in Code 5 erklärt.

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

Code 5: PicoC Code für Pointer Dereferenzierung

Der Knoten `Deref(Name('var'))` repräsentiert in der **Abstrakten Syntax** in Code 6 eine **Dereferenzierung** `*var`.

```
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

Im **PicoC-Shrink Pass** in Code 7 wird ein Trikc angewandet, bei dem der Knoten `Exp(Deref(Name('pntr'), Num('0')))` einfach durch den Knoten `Exp(Subscr(Name('pntr'), Num('0')))` ersetzt wird. Der Trick besteht

```
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

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

Code 8: PicoC Code für Array Initialisierung

```
1 File
2   Name './example_array_init.ast',
3   [
4     FunDef
5       VoidType 'void',
6       Name 'main',
7       [],
8       [
9         Assign(Alloc(Writeable(), ArrayDecl([Num('2'), Num('1')], IntType('int')),
          ↪  Name('ar')), Array([Array([Num('4')]), Array([Num('2')])]))
10      ]
11  ]
```

Code 9: Abstract Syntax Tree für Array Initialisierung

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

Code 10: Symboltabelle für Array Initialisierung

```
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;
        # Exp(Num('2'))
        SUBI SP 1;
        LOADI ACC 2;
        STOREIN SP ACC 1;
        # Ref(Subscr(Stack(Num('2')), Stack(Num('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;
        # Return(Empty())
        LOADIN BAF PC -1;
      ]
  ]
```

Code 16: RETI Blocks Pass für Zugriff auf Arrayindex

0.0.2.3 Zuweisung an Arrayindex

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

Code 17: PicoC Code für Zuweisung an Arrayindex

```
File
  Name './example_array_assignment.ast',
```

```
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Exp(Alloc(Writeable(), ArrayDecl([Num('2')], IntType('int')), Name('ar')))
        Assign(Subscr(Name('ar'), Num('2')), Num('42'))
      ]
  ]
```

Code 18: Abstract Syntax Tree für Zuweisung an Arrayindex

```
File
  Name './example_array_assignment.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        // Assign(Subscr(Name('ar'), Num('2')), Num('42'))
        Exp(Num('42'))
        Ref(Global(Num('0')))
        Exp(Num('2'))
        Ref(Subscr(Stack(Num('2')), Stack(Num('1'))))
        Assign(Stack(Num('1')), Stack(Num('2')))
        Return(Empty())
      ]
  ]
```

Code 19: PicoC Mon Pass für Zuweisung an Arrayindex

```
File
  Name './example_array_assignment.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        # // Assign(Subscr(Name('ar'), Num('2')), 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;
        # Exp(Num('2'))
        SUBI SP 1;
        LOADI ACC 2;
        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')
```

```
    },
  Symbol
    {
      type qualifier:         Empty()
      datatype:               StructSpec(Name('st1'))
      name:                   Name('st@st2')
      value or address:       Empty()
      position:               Pos(Num('3'), Num('23'))
      size:                   Num('3')
    },
  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('3')
    },
  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()
    }
]
```

Code 22: Symboltabelle für Deklaration von Structs

0.0.3.2 Initialisierung von Structs

```
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',
  [
```

```
4     Block
5       Name 'main.0',
6       [
7         // Assign(Name('var'), Num('42'))
8         Exp(Num('42'))
9         Assign(Global(Num('0')), Stack(Num('1')))
10        // Assign(Name('st'), Struct([Assign(Name('st'), Struct([Assign(Name('pntr'),
          ↪  Array([Array([Ref(Name('var'))])]))])]))]))
11        Ref(Global(Num('0')))
12        Assign(Global(Num('1')), Stack(Num('1')))
13        Return(Empty())
14      ]
15  ]
```

Code 26: PicoC Mon Pass für Initialisierung von Structs

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

Code 27: RETI Blocks Pass für Initialisierung von Structs

0.0.3.3 Zugriff auf Structattribut

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

Code 28: PicoC Code für Zugriff auf Structattribut

```
1 File
2   Name './example_struct_attr_access.ast',
3   [
4     StructDecl
5       Name 'pos',
6       [
7         Alloc
8           Writeable,
9           IntType 'int',
10          Name 'x',
11        Alloc
12          Writeable,
13          IntType 'int',
14          Name 'y'
15      ],
16    FunDef
17      VoidType 'void',
18      Name 'main',
19      [],
20      [
21        Assign(Alloc(Writeable(), StructSpec(Name('pos')), Name('st')),
          ↪  Struct([Assign(Name('x'), Num('4')), Assign(Name('y'), Num('2'))]))
22        Exp(Attr(Name('st'), Name('y')))
23      ]
24  ]
```

Code 29: Abstract Syntax Tree für Zugriff auf Structattribut

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

```
12        Ref(Attr(Stack(Num('1')), Name('y')))
13        Exp(Stack(Num('1')))
14        Return(Empty())
15      ]
16  ]
```

Code 30: PicoC Mon Pass für Zugriff auf Structattribut

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

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

void main() {
  struct pos st = {.x=4, .y=2};
  st.y = 42;
}
```

Code 32: 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 33: Abstract Syntax Tree für Zuweisung an Structattribut

```
File
  Name './example_struct_attr_assignment.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')))
        // Assign(Attr(Name('st'), Name('y')), Num('42'))
```

```
12        Exp(Num('42'))
13        Ref(Global(Num('0')))
14        Ref(Attr(Stack(Num('1')), Name('y')))
15        Assign(Stack(Num('1')), Stack(Num('2')))
16        Return(Empty())
17      ]
18    ]
```

Code 34: PicoC Mon Pass für Zuweisung an Structattribut

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

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(Alloc(Writeable(), ArrayDecl([Num('3')], StructSpec(Name('ar_with_len'))),
           ↪  Name('st_ar')))
26         Exp(Alloc(Writeable(), PntrDecl(Num('1'), ArrayDecl([Num('3')], PntrDecl(Num('1'),
           ↪  IntType('int')))), Name('pntr2')))
27         Exp(Name('pntr2'))
```

```
28      ],
29    FunDef
30      VoidType 'void',
31      Name 'fun',
32      [],
33      [
34        Exp(Alloc(Writeable(), ArrayDecl([Num('3')], StructSpec(Name('ar_with_len'))),
          ↪  Name('st_ar')))
35        Exp(Alloc(Writeable(), PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int'))),
          ↪  Name('pntr1')))
36        Exp(Name('pntr1'))
37      ]
38  ]
```

Code 37: Abstract Syntax Tree für den Einleitungsteil

```
1 File
2   Name './example_derived_dts_introduction_part.picoc_mon',
3   [
4     Block
5       Name 'main.1',
6       [
7         Exp(Global(Num('9')))
8         Return(Empty())
9       ],
10    Block
11      Name 'fun.0',
12      [
13        Exp(Stackframe(Num('9')))
14        Return(Empty())
15      ]
16  ]
```

Code 38: PicoC Mon Pass für den Einleitungsteil

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

```
17        # Exp(Stackframe(Num('9')))
18        SUBI SP 1;
19        LOADIN BAF ACC -11;
20        STOREIN SP ACC 1;
21        # Return(Empty())
22        LOADIN BAF PC -1;
23      ]
24    ]
```

Code 39: RETI Blocks Pass für den Einleitungsteil

0.0.4.2 Mittelteil für die verschiedenen Derived Datatypes

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

Code 40: PicoC Code für den Mittelteil

```
1 File
2   Name './example_derived_dts_main_part.ast',
3   [
4     StructDecl
5       Name 'st1',
6       [
7         Alloc
8           Writeable,
9           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(Alloc(Writeable(), ArrayDecl([Num('1')], IntType('int')), Name('var')),
          ↪  Array([Num('42')]))
24        Assign(Alloc(Writeable(), StructSpec(Name('st1')), Name('st_first')),
          ↪  Struct([Assign(Name('ar'), Ref(Name('var')))]))
25        Exp(Subscr(Deref(Attr(Name('st_first'), Name('ar')), Num('0')), Num('0')))
26      ]
27  ]
```

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;
```

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

Code 43: RETI Blocks Pass für den Mittelteil

0.0.4.3 Schlussteil für die verschiedenen Derived Datatypes

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

```
 9
10  ar1[0];
11  ar2[0];
12  *pntr2;
13 }
```

Code 44: PicoC Code für den Schlussteil

```
 1 File
 2   Name './example_derived_dts_final_part.ast',
 3   [
 4     StructDecl
 5       Name 'st',
 6       [
 7         Alloc
 8           Writeable,
 9           ArrayDecl
10             [
11               Num '2'
12             ],
13             IntType 'int',
14           Name 'attr'
15       ],
16     FunDef
17       VoidType 'void',
18       Name 'main',
19       [],
20       [
21         Assign(Alloc(Writeable(), ArrayDecl([Num('1'), Num('2')], IntType('int')),
           ↪  Name('ar1')), Array([Array([Num('42'), Num('314')])]))
22         Assign(Alloc(Writeable(), ArrayDecl([Num('1')], StructSpec(Name('st'))),
           ↪  Name('ar2')), Struct([Assign(Name('attr'), Array([Num('42'), Num('314')]))]))
23         Assign(Alloc(Writeable(), IntType('int'), Name('var')), Num('42'))
24         Assign(Alloc(Writeable(), PntrDecl(Num('1'), IntType('int')), Name('pntr1')),
           ↪  Ref(Name('var')))
25         Assign(Alloc(Writeable(), PntrDecl(Num('2'), IntType('int')), Name('pntr2')),
           ↪  Ref(Name('pntr1')))
26         Exp(Subscr(Name('ar1'), Num('0')))
27         Exp(Subscr(Name('ar2'), Num('0')))
28         Exp(Deref(Name('pntr2'), Num('0')))
29       ]
30   ]
```

Code 45: Abstract Syntax Tree für den Schlussteil

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

```
7         // Assign(Name('ar1'), Array([Array([Num('42'), Num('314')])]))
8         Exp(Num('42'))
9         Exp(Num('314'))
10        Assign(Global(Num('0')), Stack(Num('2')))
11        // Assign(Name('ar2'), Struct([Assign(Name('attr'), Array([Num('42'),
          ↪  Num('314')]))]))
12        Exp(Num('42'))
13        Exp(Num('314'))
14        Assign(Global(Num('2')), Stack(Num('2')))
15        // Assign(Name('var'), Num('42'))
16        Exp(Num('42'))
17        Assign(Global(Num('4')), Stack(Num('1')))
18        // Assign(Name('pntr1'), Ref(Name('var')))
19        Ref(Global(Num('4')))
20        Assign(Global(Num('5')), Stack(Num('1')))
21        // Assign(Name('pntr2'), Ref(Name('pntr1')))
22        Ref(Global(Num('5')))
23        Assign(Global(Num('6')), Stack(Num('1')))
24        // Exp(Subscr(Name('ar1'), Num('0')))
25        Ref(Global(Num('0')))
26        Exp(Num('0'))
27        Ref(Subscr(Stack(Num('2')), Stack(Num('1'))))
28        Exp(Stack(Num('1')))
29        // Exp(Subscr(Name('ar2'), Num('0')))
30        Ref(Global(Num('2')))
31        Exp(Num('0'))
32        Ref(Subscr(Stack(Num('2')), Stack(Num('1'))))
33        Exp(Stack(Num('1')))
34        // Exp(Subscr(Name('pntr2'), Num('0')))
35        Ref(Global(Num('6')))
36        Exp(Num('0'))
37        Ref(Subscr(Stack(Num('2')), Stack(Num('1'))))
38        Exp(Stack(Num('1')))
39        Return(Empty())
40      ]
41    ]
```

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(Empty())
10       ],
11     FunDef
12       VoidType 'void',
13       Name 'fun1',
14       [],
15       [],
16     FunDef
17       IntType 'int',
18       Name 'fun2',
19       [],
20       [
21         Return(Num('1'))
22       ]
23   ]
```

Code 49: Abstract Syntax Tree für 3 Funktionen

```
1  File
2    Name './example_3_funs.picoc_blocks',
3    [
4      FunDef
5        VoidType 'void',
6        Name 'main',
7        [],
```

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

Code 50: PicoC Blocks Pass für 3 Funktionen

```
1 File
2   Name './example_3_funs.picoc_mon',
3   [
4     Block
5       Name 'main.2',
6       [
7         Return(Empty())
8       ],
9     Block
10      Name 'fun1.1',
11      [
12        Return(Empty())
13      ],
14    Block
15      Name 'fun2.0',
16      [
17        // Return(Num('1'))
18        Exp(Num('1'))
19        Return(Stack(Num('1')))
20      ]
21  ]
```

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

```
void fun1() {
}

int fun2() {
  return 1;
}

void main() {
  return;
}
```

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

```
File
  Name './example_3_funs_main.picoc_mon',
```

```
  [
    Block
      Name 'fun1.2',
      [
        Return(Empty())
      ],
    Block
      Name 'fun2.1',
      [
        // Return(Num('1'))
        Exp(Num('1'))
        Return(Stack(Num('1')))
      ],
    Block
      Name 'main.0',
      [
        Return(Empty())
      ]
  ]
```

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

```
File
  Name './example_3_funs_main.reti_blocks',
  [
    Block
      Name 'fun1.2',
      [
        # Return(Empty())
        LOADIN BAF PC -1;
      ],
    Block
      Name 'fun2.1',
      [
        # // 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;
      ],
    Block
      Name 'main.0',
      [
        # Return(Empty())
        LOADIN BAF PC -1;
      ]
  ]
```

Code 55: 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',
      [
        # Return(Empty())
        LOADIN BAF PC -1;
      ],
    Block
      Name 'fun2.1',
      [
        # // 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;
      ],
    Block
      Name 'main.0',
      [
        # Return(Empty())
        LOADIN BAF PC -1;
      ]
  ]
```

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

0.0.5.2 Funktionsdeklaration und -definition

```c
int fun2(int var);

void fun1() {
}

void main() {
  int var = fun2(42);
  return;
}

int fun2(int var) {
  return var;
}
```

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

```
SymbolTable
  [
    Symbol
      {
        type qualifier:         Empty()
        datatype:               FunDecl(IntType('int'), Name('fun2'), [Alloc(Writeable(),
        ↪  IntType('int'), Name('var'))])
        name:                   Name('fun2')
        value or address:       Empty()
        position:               Pos(Num('1'), Num('4'))
        size:                   Empty()
      },
    Symbol
      {
        type qualifier:         Empty()
        datatype:               FunDecl(VoidType('void'), Name('fun1'), [])
        name:                   Name('fun1')
        value or address:       Empty()
        position:               Pos(Num('3'), Num('5'))
        size:                   Empty()
      },
    Symbol
      {
        type qualifier:         Empty()
        datatype:               FunDecl(VoidType('void'), Name('main'), [])
        name:                   Name('main')
        value or address:       Empty()
        position:               Pos(Num('6'), Num('5'))
        size:                   Empty()
      },
    Symbol
      {
        type qualifier:         Writeable()
        datatype:               IntType('int')
        name:                   Name('var@main')
        value or address:       Num('0')
        position:               Pos(Num('7'), Num('6'))
        size:                   Num('1')
      },
    Symbol
      {
        type qualifier:         Writeable()
        datatype:               IntType('int')
        name:                   Name('var@fun2')
        value or address:       Num('0')
        position:               Pos(Num('11'), Num('13'))
        size:                   Num('1')
      }
  ]
```

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(Num('2'))
8          Ref(Global(Num('0')))
9          NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
10         Exp(GoTo(Name('stack_fun.0')))
11         RemoveStackframe()
12         Return(Empty())
13       ],
14     Block
15       Name 'stack_fun.0',
16       [
17         Return(Empty())
18       ]
19   ]
```

Code 60: PicoC Mon Pass für Funktionsaufruf ohne Rückgabewert

```
1  File
2    Name './example_fun_call_no_return_value.reti_blocks',
3    [
4      Block
5        Name 'main.1',
6        [
7          # StackMalloc(Num('2'))
8          SUBI SP 2;
```

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

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())
```

```
LOADIN BAF PC -1;
# Return(Empty())
LOADIN BAF PC -1;
```

Code 62: RETI Pass für Funktionsaufruf ohne Rückgabewert

0.0.5.3.2 Mit Rückgabewert

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

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

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
```

```
5       Name 'stack_fun.1',
6       [
7         # // Return(Num('42'))
8         # Exp(Num('42'))
9         SUBI SP 1;
10        LOADI ACC 42;
11        STOREIN SP ACC 1;
12        # Return(Stack(Num('1')))
13        LOADIN SP ACC 1;
14        ADDI SP 1;
15        LOADIN BAF PC -1;
16      ],
17    Block
18      Name 'main.0',
19      [
20        # // Assign(Name('var'), Call(Name('stack_fun'), []))
21        # StackMalloc(Num('2'))
22        SUBI SP 2;
23        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
24        MOVE BAF ACC;
25        ADDI SP 2;
26        MOVE SP BAF;
27        SUBI SP 2;
28        STOREIN BAF ACC 0;
29        LOADI ACC GoTo(Name('addr@next_instr'));
30        ADD ACC CS;
31        STOREIN BAF ACC -1;
32        Exp(GoTo(Name('stack_fun.1')))
33        # RemoveStackframe()
34        MOVE BAF IN1;
35        LOADIN IN1 BAF 0;
36        MOVE IN1 SP;
37        # Assign(Global(Num('0')), Stack(Num('1')))
38        LOADIN SP ACC 1;
39        STOREIN DS ACC 0;
40        ADDI SP 1;
41        # Return(Empty())
42        LOADIN BAF PC -1;
43      ]
44  ]
```

Code 65: RETI Blocks Pass für Funktionsaufruf mit Rückgabewert

```
1 JUMP 7;
2 # // Return(Num('42'))
3 # Exp(Num('42'))
4 SUBI SP 1;
5 LOADI ACC 42;
6 STOREIN SP ACC 1;
7 # Return(Stack(Num('1')))
8 LOADIN SP ACC 1;
9 ADDI SP 1;
10 LOADIN BAF PC -1;
11 # // Assign(Name('var'), Call(Name('stack_fun'), []))
```

```
# 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()
      },
    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 69: Symboltabelle für Call by Sharing für Arrays

```
File
  Name './example_fun_call_by_sharing_array.reti_blocks',
  [
    Block
      Name 'stack_fun.1',
      [
        # Return(Empty())
        LOADIN BAF PC -1;
      ],
    Block
      Name 'main.0',
      [
        # StackMalloc(Num('2'))
        SUBI SP 2;
        # Ref(Global(Num('0')))
        SUBI SP 1;
        LOADI IN1 0;
        ADD IN1 DS;
        STOREIN SP IN1 1;
        # Ref(Global(Num('6')))
        SUBI SP 1;
        LOADI IN1 6;
        ADD IN1 DS;
        STOREIN SP IN1 1;
        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
        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')))
        # RemoveStackframe()
```

```
36          MOVE BAF IN1;
37          LOADIN IN1 BAF 0;
38          MOVE IN1 SP;
39          # Return(Empty())
40          LOADIN BAF PC -1;
41        ]
42    ]
```

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(Empty())
8       ],
9     Block
10      Name 'main.0',
11      [
12        StackMalloc(Num('2'))
13        Assign(Stack(Num('3')), Global(Num('0')))
14        NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
15        Exp(GoTo(Name('stack_fun.1')))
16        RemoveStackframe()
17        Return(Empty())
18      ]
19    ]
```

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())
        LOADIN BAF PC -1;
      ]
  ]
```

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