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 Funktionen

0.0.1.1 Grundsätzliche Umsetzung

Die Umsetzung von Pro mittels Funktionen wird im Folgenden mithilfe des Beispiels in Code 0.1 erklärt.

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

Code 0.1: 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 0.2: 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 0.3: RETI-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 0.4: 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 0.5: RETI-Blocks Pass für 3 Funktionen

0.0.1.1.1 Sprung zur Main Funktion

```c
void fun1() {
}

int fun2() {
  return 1;
}

void main() {
  return;
}
```

Code 0.6: 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 0.7: 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 0.8: RETI-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')))
        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 0.9: PicoC-Patch Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

0.0.1.2 Funktionsdeklaration und -definition und Umsetzung von Scopes

```
int fun2(int var);

void fun1() {
}

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

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

```
13 }
```

Code 0.10: PicoC-Code für Funktionen, wobei eine Funktion vorher deklariert werden muss

Bei mehreren Funktionen werden die **Scopes** der unterschiedlichen **Funktionen** mittels eines **Suffix** `"<fun_name>@"` umgesetzt, der an den **Variablennamen** `<var>` drangehängt wird: `<var>@<fun_name>`. Dieser **Suffix** wird geändert sobald beim **Top-Down**[1] Durchiterieren über den **Abstract Syntax Tree** des aktuellen **Passes** nach dem **Depth-First-Search** Schema über den

```
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')
```

[1] D.h. von der Wurzel zu den Blättern eines Baumes

```
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 0.11: Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss

0.0.1.3 Funktionsaufruf

0.0.1.3.1 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 0.12: 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          // Exp(Alloc(Writeable(), ArrayDecl([Num('2'), Num('3')], StructSpec(Name('st'))),
           ↪  Name('local_var')))
8          // Exp(Call(Name('stack_fun'), [Name('local_var')]))
9          StackMalloc(Num('2'))
10         Ref(Global(Num('0')))
11         NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
12         Exp(GoTo(Name('stack_fun.0')))
13         RemoveStackframe()
14         Return(Empty())
15       ],
16     Block
17       Name 'stack_fun.0',
18       [
19         // Exp(Alloc(Writeable(), ArrayDecl([Num('3')], StructSpec(Name('st'))),
           ↪  Name('param')))
20         // Exp(Alloc(Writeable(), IntType('int'), Name('local_var')))
```

```
21        Return(Empty())
22      ]
23  ]
```

Code 0.13: 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         # // Exp(Alloc(Writeable(), ArrayDecl([Num('2'), Num('3')], StructSpec(Name('st'))),
          ↪  Name('local_var')))
8         # // Exp(Call(Name('stack_fun'), [Name('local_var')]))
9         # StackMalloc(Num('2'))
10        SUBI SP 2;
11        # Ref(Global(Num('0')))
12        SUBI SP 1;
13        LOADI IN1 0;
14        ADD IN1 DS;
15        STOREIN SP IN1 1;
16        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
17        MOVE BAF ACC;
18        ADDI SP 3;
19        MOVE SP BAF;
20        SUBI SP 4;
21        STOREIN BAF ACC 0;
22        LOADI ACC GoTo(Name('addr@next_instr'));
23        ADD ACC CS;
24        STOREIN BAF ACC -1;
25        # Exp(GoTo(Name('stack_fun.0')))
26        Exp(GoTo(Name('stack_fun.0')))
27        # RemoveStackframe()
28        MOVE BAF IN1;
29        LOADIN IN1 BAF 0;
30        MOVE IN1 SP;
31        # Return(Empty())
32        LOADIN BAF PC -1;
33      ],
34    Block
35      Name 'stack_fun.0',
36      [
37        # // Exp(Alloc(Writeable(), ArrayDecl([Num('3')], StructSpec(Name('st'))),
          ↪  Name('param')))
38        # // Exp(Alloc(Writeable(), IntType('int'), Name('local_var')))
39        # Return(Empty())
40        LOADIN BAF PC -1;
41      ]
42  ]
```

Code 0.14: RETI-Blocks Pass für Funktionsaufruf ohne Rückgabewert

```
# // Exp(GoTo(Name('main.1')))
# // not included Exp(GoTo(Name('main.1')))
# // Exp(Alloc(Writeable(), ArrayDecl([Num('2'), Num('3')], StructSpec(Name('st'))),
↪  Name('local_var')))
# // Exp(Call(Name('stack_fun'), [Name('local_var')]))
# 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 14;
ADD ACC CS;
STOREIN BAF ACC -1;
# Exp(GoTo(Name('stack_fun.0')))
JUMP 5;
# RemoveStackframe()
MOVE BAF IN1;
LOADIN IN1 BAF 0;
MOVE IN1 SP;
# Return(Empty())
LOADIN BAF PC -1;
# // Exp(Alloc(Writeable(), ArrayDecl([Num('3')], StructSpec(Name('st'))), Name('param')))
# // Exp(Alloc(Writeable(), IntType('int'), Name('local_var')))
# Return(Empty())
LOADIN BAF PC -1;
```

Code 0.15: RETI-Pass für Funktionsaufruf ohne Rückgabewert

0.0.1.3.2 Mit Rückgabewert

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

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

Code 0.16: 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 0.17: 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')))
```

```
33          Exp(GoTo(Name('stack_fun.1')))
34          # RemoveStackframe()
35          MOVE BAF IN1;
36          LOADIN IN1 BAF 0;
37          MOVE IN1 SP;
38          # Assign(Global(Num('0')), Stack(Num('1')))
39          LOADIN SP ACC 1;
40          STOREIN DS ACC 0;
41          ADDI SP 1;
42          # Return(Empty())
43          LOADIN BAF PC -1;
44        ]
45    ]
```

Code 0.18: RETI-Blocks Pass für Funktionsaufruf mit Rückgabewert

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

Code 0.19: RETI-Pass für Funktionsaufruf mit Rückgabewert

0.0.1.3.3 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 0.20: PicoC-Code für Call by Sharing für Arrays

```
File
  Name './example_fun_call_by_sharing_array.picoc_mon',
  [
    Block
      Name 'stack_fun.1',
      [
        // Exp(Alloc(Writeable(), PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int'))),
        ↪  Name('param1')))
        // Exp(Alloc(Writeable(), ArrayDecl([Num('3')], IntType('int')), Name('param2')))
        Return(Empty())
      ],
    Block
      Name 'main.0',
      [
        // Exp(Alloc(Writeable(), ArrayDecl([Num('2'), Num('3')], IntType('int')),
        ↪  Name('local_var1')))
        // Exp(Alloc(Writeable(), ArrayDecl([Num('2'), Num('3')], IntType('int')),
        ↪  Name('local_var2')))
        // Exp(Call(Name('stack_fun'), [Name('local_var1'), Name('local_var2')]))
        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 0.21: PicoC-Mon Pass für Call by Sharing für Arrays

```
SymbolTable
  [
    Symbol
      {
        type qualifier:         Empty()
```

```
6         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'))])
7         name:                   Name('stack_fun')
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:               PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
16        name:                   Name('param1@stack_fun')
17        value or address:       Num('0')
18        position:               Pos(Num('1'), Num('21'))
19        size:                   Num('1')
20      },
21    Symbol
22      {
23        type qualifier:         Writeable()
24        datatype:               PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
25        name:                   Name('param2@stack_fun')
26        value or address:       Num('1')
27        position:               Pos(Num('1'), Num('37'))
28        size:                   Num('1')
29      },
30    Symbol
31      {
32        type qualifier:         Empty()
33        datatype:               FunDecl(VoidType('void'), Name('main'), [])
34        name:                   Name('main')
35        value or address:       Empty()
36        position:               Pos(Num('4'), Num('5'))
37        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 0.22: Symboltabelle für Call by Sharing für Arrays

```
File
  Name './example_fun_call_by_sharing_array.reti_blocks',
  [
    Block
      Name 'stack_fun.1',
      [
        # // Exp(Alloc(Writeable(), PntrDecl(Num('1'), ArrayDecl([Num('3')],
        ↪  IntType('int'))), Name('param1')))
        # // Exp(Alloc(Writeable(), ArrayDecl([Num('3')], IntType('int')), Name('param2')))
        # Return(Empty())
        LOADIN BAF PC -1;
      ],
    Block
      Name 'main.0',
      [
        # // Exp(Alloc(Writeable(), ArrayDecl([Num('2'), Num('3')], IntType('int')),
        ↪  Name('local_var1')))
        # // Exp(Alloc(Writeable(), ArrayDecl([Num('2'), Num('3')], IntType('int')),
        ↪  Name('local_var2')))
        # // Exp(Call(Name('stack_fun'), [Name('local_var1'), Name('local_var2')]))
        # 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')))
        Exp(GoTo(Name('stack_fun.1')))
        # RemoveStackframe()
        MOVE BAF IN1;
        LOADIN IN1 BAF 0;
        MOVE IN1 SP;
        # Return(Empty())
        LOADIN BAF PC -1;
      ]
  ]
```

Code 0.23: RETI-Block Pass für Call by Sharing für Arrays

0.0.1.3.4 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 0.24: 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         // Exp(Alloc(Writeable(), StructSpec(Name('st')), Name('param')))
8         Return(Empty())
9       ],
10    Block
11      Name 'main.0',
12      [
13        // Exp(Alloc(Writeable(), StructSpec(Name('st')), Name('local_var')))
14        // Exp(Call(Name('stack_fun'), [Name('local_var')]))
15        StackMalloc(Num('2'))
16        Assign(Stack(Num('3')), Global(Num('0')))
17        NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
18        Exp(GoTo(Name('stack_fun.1')))
19        RemoveStackframe()
20        Return(Empty())
21      ]
22  ]
```

Code 0.25: PicoC-Mon Pass für Call by Value für Structs

```
1 File
2   Name './example_fun_call_by_value_struct.reti_blocks',
3   [
4     Block
5       Name 'stack_fun.1',
6       [
7         # // Exp(Alloc(Writeable(), StructSpec(Name('st')), Name('param')))
8         # Return(Empty())
9         LOADIN BAF PC -1;
10      ],
11    Block
```

```
12      Name 'main.0',
13      [
14        # // Exp(Alloc(Writeable(), StructSpec(Name('st')), Name('local_var')))
15        # // Exp(Call(Name('stack_fun'), [Name('local_var')]))
16        # StackMalloc(Num('2'))
17        SUBI SP 2;
18        # Assign(Stack(Num('3')), Global(Num('0')))
19        SUBI SP 3;
20        LOADIN DS ACC 0;
21        STOREIN SP ACC 1;
22        LOADIN DS ACC 1;
23        STOREIN SP ACC 2;
24        LOADIN DS ACC 2;
25        STOREIN SP ACC 3;
26        # NewStackframe(Name('stack_fun'), GoTo(Name('addr@next_instr')))
27        MOVE BAF ACC;
28        ADDI SP 5;
29        MOVE SP BAF;
30        SUBI SP 5;
31        STOREIN BAF ACC 0;
32        LOADI ACC GoTo(Name('addr@next_instr'));
33        ADD ACC CS;
34        STOREIN BAF ACC -1;
35        # Exp(GoTo(Name('stack_fun.1')))
36        Exp(GoTo(Name('stack_fun.1')))
37        # RemoveStackframe()
38        MOVE BAF IN1;
39        LOADIN IN1 BAF 0;
40        MOVE IN1 SP;
41        # Return(Empty())
42        LOADIN BAF PC -1;
43      ]
44  ]
```

Code 0.26: RETI-Block Pass für Call by Value für Structs

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