Übersicht über den e2-Zwischencode, v 1.0

1 Operanden

Name	erlaubte Typen	Beschreibung
`IRVariable`	`int`, `real`, Array (*eindimensional*)	Variable des Quellprogramms od. virtuelles Register
`IRIntConstant`	`int`	`int`-Konstante
`IRRealConstant`	`real`	`real`-Konstante

Wichtig: Variablennamen müssen (zumindest innerhalb einer Funktion) *eindeutig* sein.

2 Instruktionen

Name	Ziel, Sprungmarke	Quelle	Beschreibung
`NOP`	–	–	*No Operation*
`MOV`	Variable	Konstante od. Variable	Kopieroperation
`PHI`	Variable	Konstante od. Variable pro KFG-Vorgänger	φ-Funktion für SSA-Form (UE2)
`LOAD`	Variable	Array, Index (Konstante od. Variable)	lesender Array-Zugriff
`STORE`	Array, Index (s.o.)	Konstante od. Variable	schreibender Array-Zugriff
`ADD`	Variable	`a`, `b`: Variablen od. Konstanten	Addition, `a+b`
`SUB`	Variable	`a`, `b`: Variablen od. Konstanten	Subtraktion, `a-b`
`MUL`	Variable	`a`, `b`: Variablen od. Konstanten	Multiplikation, `a*b`
`DIV`	Variable	`a`, `b`: Variablen od. Konstanten	Division, `a/b`
`I2R`	Variable	Konstante od. Variable	Typkonvertierung `int` $\mapsto$ `real`
`R2I`	Variable	Konstante od. Variable	Typkonvertierung `real` $\mapsto$ `int`
`LABEL`	–	–	Sprungmarke
`JMP`	Sprungmarke	–	unbedingter Sprung
`JEQ`	Sprungmarke	`a`, `b`: Variablen od. Konstanten	Sprung, falls `a` $=$ `b`
`JNE`	Sprungmarke	`a`, `b`: Variablen od. Konstanten	Sprung, falls `a` $\neq$ `b`
`JLT`	Sprungmarke	`a`, `b`: Variablen od. Konstanten	Sprung, falls `a` $<$ `b`
`JLE`	Sprungmarke	`a`, `b`: Variablen od. Konstanten	Sprung, falls `a` $\leq$ `b`
`JGT`	Sprungmarke	`a`, `b`: Variablen od. Konstanten	Sprung, falls `a` $>$ `b`
`JGE`	Sprungmarke	`a`, `b`: Variablen od. Konstanten	Sprung, falls `a` $\geq$ `b`
`CALL`	Variable (optional)	Funktionsname, Argumente (Variablen od. Konstanten)	Funktionsaufruf
`RET`	–	optional Rückgabewert (Variable od. Konstante)	Rücksprung aus Funktion

Wichtig: Alle Instruktionen sind *typisiert* und die Typen der Operanden müssen *zueinanderpassen*.

3 Textuelle Repräsentation

Die textuelle Repräsentation des Zwischencodes verwendet die folgenden Schlüsselwörter:

- `.global`: Deklaration einer globalen Variable
- `.func`: Beginn einer Funktionsdefinition
- `.param`: Deklaration eines Funktionsparameters
- `.local`: Deklaration beliebig vieler lokaler Variablen
- `.virt`: Deklaration beliebig vieler virtueller Register
- `.code`: Beginn des Funktionsrumpfs

4 Beispiel

```
var mem : int[51];

func fib(n : int): int
  var sol : int;

  if n == 0 or n == 1 then
    sol := n;
  else
    if mem[n] == 0 then
      mem[n] := fib(n-1) + fib(n-2);
    end
    sol := mem[n];
  end
  return sol;
end

func main(): int
  writeInt(fib(50));
  writeChar(10);
  return 0;
end
```

```
.global _mem : int[51]

.func fib : int
  .param n$0 : int
  .local sol$0 : int
  .virt %v0 %v1 %v2 %v3 %v4 %v5 %v6 : int
  .code
      jeq n$0, $0, L0
      jmp L3
    L3:
      jeq n$0, $1, L0
      jmp L1
    L0:
      sol$0 = mov n$0
      jmp L2
    L1:
      %v0 = load _mem[n$0]
      jeq %v0, $0, L4
      jmp L5
    L4:
      %v2 = sub n$0, $1
      %v1 = call fib(%v2)
      %v4 = sub n$0, $2
      %v3 = call fib(%v4)
      %v5 = add %v1, %v3
      _mem[n$0] = store %v5
      jmp L6
    L5:
    L6:
      %v6 = load _mem[n$0]
      sol$0 = mov %v6
    L2:
      ret sol$0

.func main : int
  .virt %v7 %v8 %v9 : int
  .code
      %v8 = call fib($50)
      %v7 = call writeInt(%v8)
      %v9 = call writeChar($10)
      ret $0
```