EECS483 D11: Dataflow Analysis: Example

Chun-Hung Hsiao

March 29, 2013

Announcements

- PP4 due date extended to 4/3
- PP5 will ask you to implement optimizations based on your PP4

Example Decaf

```
int sum_positive(int[] a, int n){
    int pos;
    int neg;
    int i;
    pos = 0;
    neg = 0;
    for (i = 0; i < n; i = i + 1) {
        if (a[i] > 0) {
            pos = pos + a[i];
        } else {
            neg = neg + a[i];
        }
    }
    return pos;
}
```

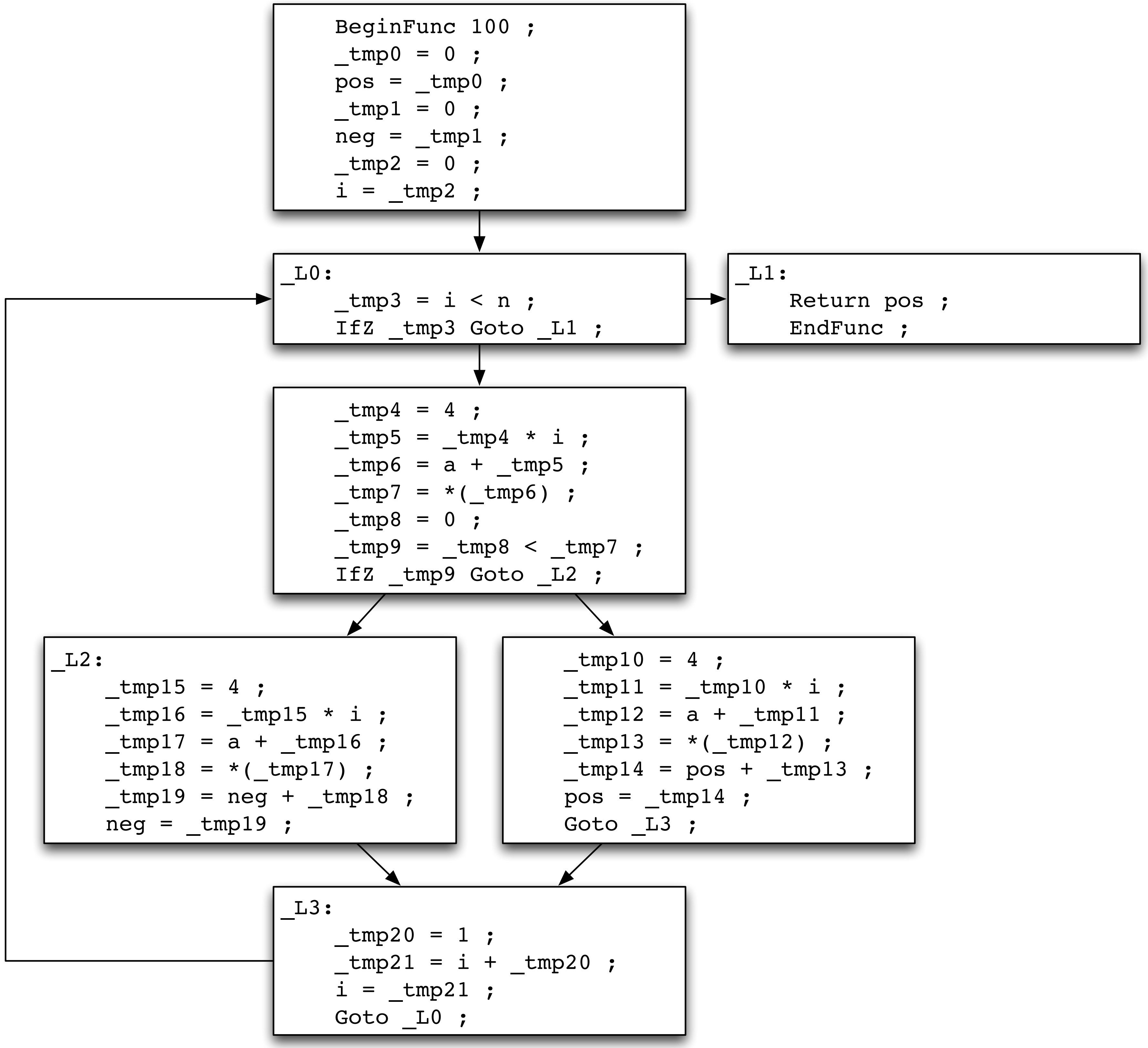
Example TAC

```
_sum_positive:
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = 0 ;
    neg = _tmp1 ;
    _tmp2 = 0 ;
    i = _tmp2 ;
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
    _tmp10 = 4 ;
    _tmp11 = _tmp10 * i ;
    _tmp12 = a + _tmp11 ;
    _tmp13 = *(_tmp12) ;
    _tmp14 = pos + _tmp13 ;
    pos = _tmp14 ;
    Goto _L3 ;
_L2:
    _tmp15 = 4 ;
    _tmp16 = _tmp15 * i ;
    _tmp17 = a + _tmp16 ;
    _tmp18 = *(_tmp17) ;
    _tmp19 = neg + _tmp18 ;
    neg = _tmp19 ;
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
_L1:
    Return pos ;
    EndFunc ;
```

Example TAC

```
_sum_positive:
     BeginFunc 100 ;
     _tmp0 = 0 ;
     pos = _tmp0 ;
     _tmp1 = 0 ;
     neg = _tmp1 ;
     _tmp2 = 0 ;
     i = _tmp2 ;
_L0:
     _tmp3 = i < n ;
     IfZ _tmp3 Goto _L1 ;
     _tmp4 = 4 ;
     _tmp5 = _tmp4 * i ;
     _tmp6 = a + _tmp5 ;
     _tmp7 = *(_tmp6) ;
     _tmp8 = 0 ;
     _tmp9 = _tmp8 < _tmp7 ;
     IfZ _tmp9 Goto _L2 ;
     _tmp10 = 4 ;
     _tmp11 = _tmp10 * i ;
     _tmp12 = a + _tmp11 ;
     _tmp13 = *(_tmp12) ;
     _tmp14 = pos + _tmp13 ;
     pos = _tmp14 ;
     Goto _L3 ;
_L2:
     _tmp15 = 4 ;
     _tmp16 = _tmp15 * i ;
     _tmp17 = a + _tmp16 ;
     _tmp18 = *(_tmp17) ;
     _tmp19 = neg + _tmp18 ;
     neg = _tmp19 ;
_L3:
     _tmp20 = 1 ;
     _tmp21 = i + _tmp20 ;
     i = _tmp21 ;
     Goto _L0 ;
_L1:
     Return pos ;
     EndFunc ;
```

Example CFG



```
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = 0 ;
    neg = _tmp1 ;
    _tmp2 = 0 ;
    i = _tmp2 ;
```

```
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
```

```
_L1:
    Return pos ;
    EndFunc ;
```

```
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
```

```
_L2:
    _tmp15 = 4 ;
    _tmp16 = _tmp15 * i ;
    _tmp17 = a + _tmp16 ;
    _tmp18 = *(_tmp17) ;
    _tmp19 = neg + _tmp18 ;
    neg = _tmp19 ;
```

```
    _tmp10 = 4 ;
    _tmp11 = _tmp10 * i ;
    _tmp12 = a + _tmp11 ;
    _tmp13 = *(_tmp12) ;
    _tmp14 = pos + _tmp13 ;
    pos = _tmp14 ;
    Goto _L3 ;
```

```
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
```

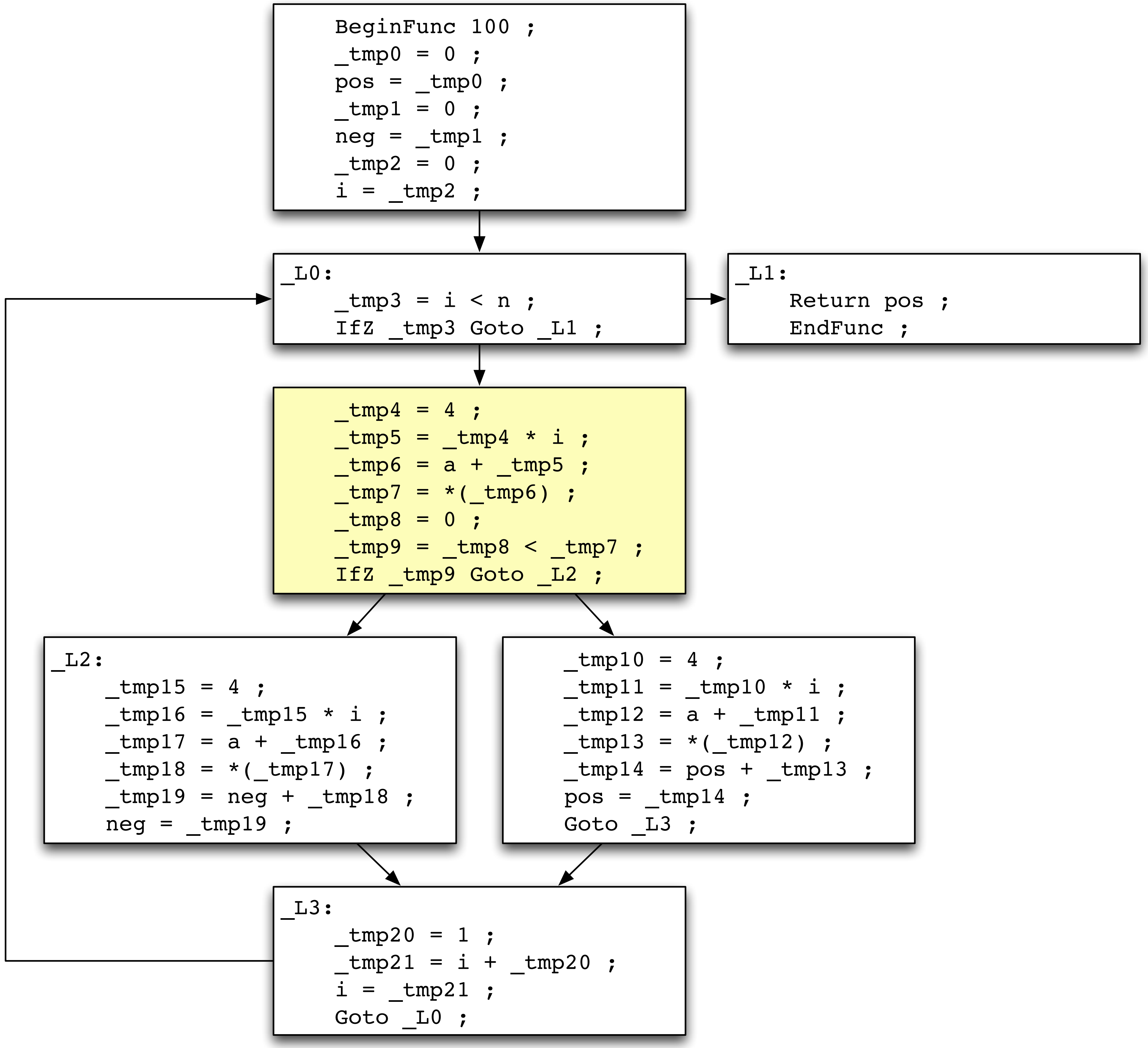
Local Available Expression Analysis

- Input
  - Basic block BB
  - IN = {available expressions before entering BB}
- Algorithm

```
OUT = IN
for each TAC s in BB in forward order:
    G = expressions generated by s
    K = expressions destroyed by s
    OUT = OUT + G - K
```

- Output
  - OUT = {available expressions after executing BB}

Example CFG



```
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = 0 ;
    neg = _tmp1 ;
    _tmp2 = 0 ;
    i = _tmp2 ;
```

```
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
```

```
_L1:
    Return pos ;
    EndFunc ;
```

```
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
```

```
_L2:
    _tmp15 = 4 ;
    _tmp16 = _tmp15 * i ;
    _tmp17 = a + _tmp16 ;
    _tmp18 = *(_tmp17) ;
    _tmp19 = neg + _tmp18 ;
    neg = _tmp19 ;
```

```
    _tmp10 = 4 ;
    _tmp11 = _tmp10 * i ;
    _tmp12 = a + _tmp11 ;
    _tmp13 = *(_tmp12) ;
    _tmp14 = pos + _tmp13 ;
    pos = _tmp14 ;
    Goto _L3 ;
```

```
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
```

Local Available Expressions

```
_tmp4 = 4 ;

_tmp5 = _tmp4 * i ;

_tmp6 = a + _tmp5 ;

_tmp7 = *(_tmp6) ;

_tmp8 = 0 ;

_tmp9 = _tmp8 < _tmp7 ;

IfZ _tmp9 Goto _L2 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;

_tmp5 = _tmp4 * i ;

_tmp6 = a + _tmp5 ;

_tmp7 = *(_tmp6) ;

_tmp8 = 0 ;

_tmp9 = _tmp8 < _tmp7 ;

IfZ _tmp9 Goto _L2 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;
OUT = {_tmp4 = 4}
_tmp5 = _tmp4 * i ;

_tmp6 = a + _tmp5 ;

_tmp7 = *(_tmp6) ;

_tmp8 = 0 ;

_tmp9 = _tmp8 < _tmp7 ;

IfZ _tmp9 Goto _L2 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;
OUT = {_tmp4 = 4}
_tmp5 = _tmp4 * i ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i}
_tmp6 = a + _tmp5 ;

_tmp7 = *(_tmp6) ;

_tmp8 = 0 ;

_tmp9 = _tmp8 < _tmp7 ;

IfZ _tmp9 Goto _L2 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;
OUT = {_tmp4 = 4}
_tmp5 = _tmp4 * i ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i}
_tmp6 = a + _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5}
_tmp7 = *(_tmp6) ;

_tmp8 = 0 ;

_tmp9 = _tmp8 < _tmp7 ;

IfZ _tmp9 Goto _L2 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;
OUT = {_tmp4 = 4}
_tmp5 = _tmp4 * i ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i}
_tmp6 = a + _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5}
_tmp7 = *(_tmp6) ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6)}
_tmp8 = 0 ;

_tmp9 = _tmp8 < _tmp7 ;

IfZ _tmp9 Goto _L2 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;
OUT = {_tmp4 = 4}
_tmp5 = _tmp4 * i ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i}
_tmp6 = a + _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5}
_tmp7 = *(_tmp6) ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6)}
_tmp8 = 0 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0}
_tmp9 = _tmp8 < _tmp7 ;

IfZ _tmp9 Goto _L2 ;
```

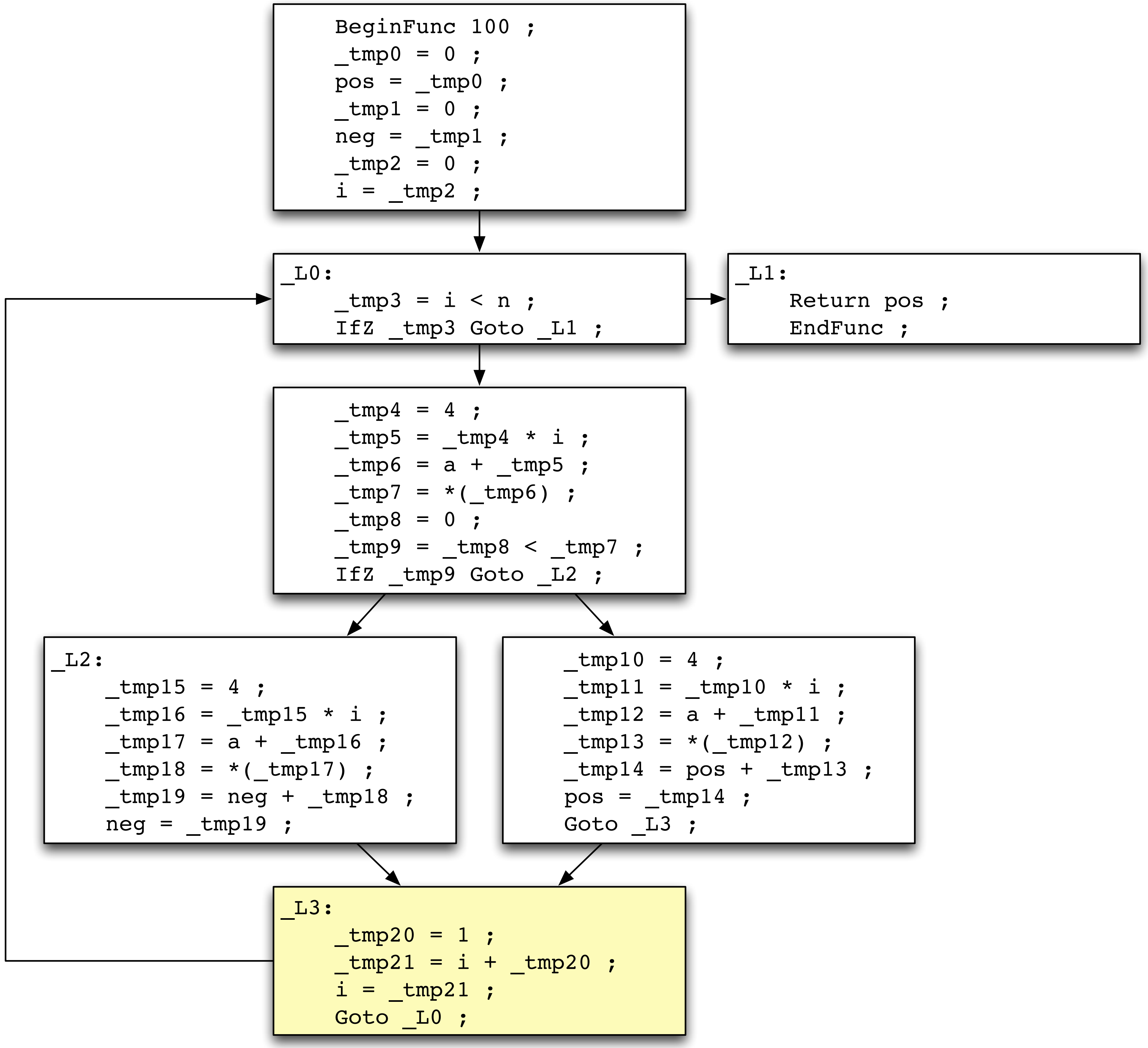
Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;
OUT = {_tmp4 = 4}
_tmp5 = _tmp4 * i ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i}
_tmp6 = a + _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5}
_tmp7 = *(_tmp6) ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6)}
_tmp8 = 0 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0}
_tmp9 = _tmp8 < _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
IfZ _tmp9 Goto _L2 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp4 = 4 ;
OUT = {_tmp4 = 4}
_tmp5 = _tmp4 * i ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i}
_tmp6 = a + _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5}
_tmp7 = *(_tmp6) ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6)}
_tmp8 = 0 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0}
_tmp9 = _tmp8 < _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
IfZ _tmp9 Goto _L2 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
```

Example CFG



```
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = 0 ;
    neg = _tmp1 ;
    _tmp2 = 0 ;
    i = _tmp2 ;
```

```
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
```

```
_L1:
    Return pos ;
    EndFunc ;
```

```
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
```

```
_L2:
    _tmp15 = 4 ;
    _tmp16 = _tmp15 * i ;
    _tmp17 = a + _tmp16 ;
    _tmp18 = *(_tmp17) ;
    _tmp19 = neg + _tmp18 ;
    neg = _tmp19 ;
```

```
    _tmp10 = 4 ;
    _tmp11 = _tmp10 * i ;
    _tmp12 = a + _tmp11 ;
    _tmp13 = *(_tmp12) ;
    _tmp14 = pos + _tmp13 ;
    pos = _tmp14 ;
    Goto _L3 ;
```

```
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
```

Local Available Expressions

```
_tmp20 = 1 ;

_tmp21 = i + _tmp20 ;

i = _tmp21 ;

Goto _L0 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp20 = 1 ;

_tmp21 = i + _tmp20 ;

i = _tmp21 ;

Goto _L0 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp20 = 1 ;
OUT = {_tmp20 = 1}
_tmp21 = i + _tmp20 ;

i = _tmp21 ;

Goto _L0 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp20 = 1 ;
OUT = {_tmp20 = 1}
_tmp21 = i + _tmp20 ;
OUT = {_tmp20 = 1,_tmp21 = i + _tmp20}
i = _tmp21 ;

Goto _L0 ;
```

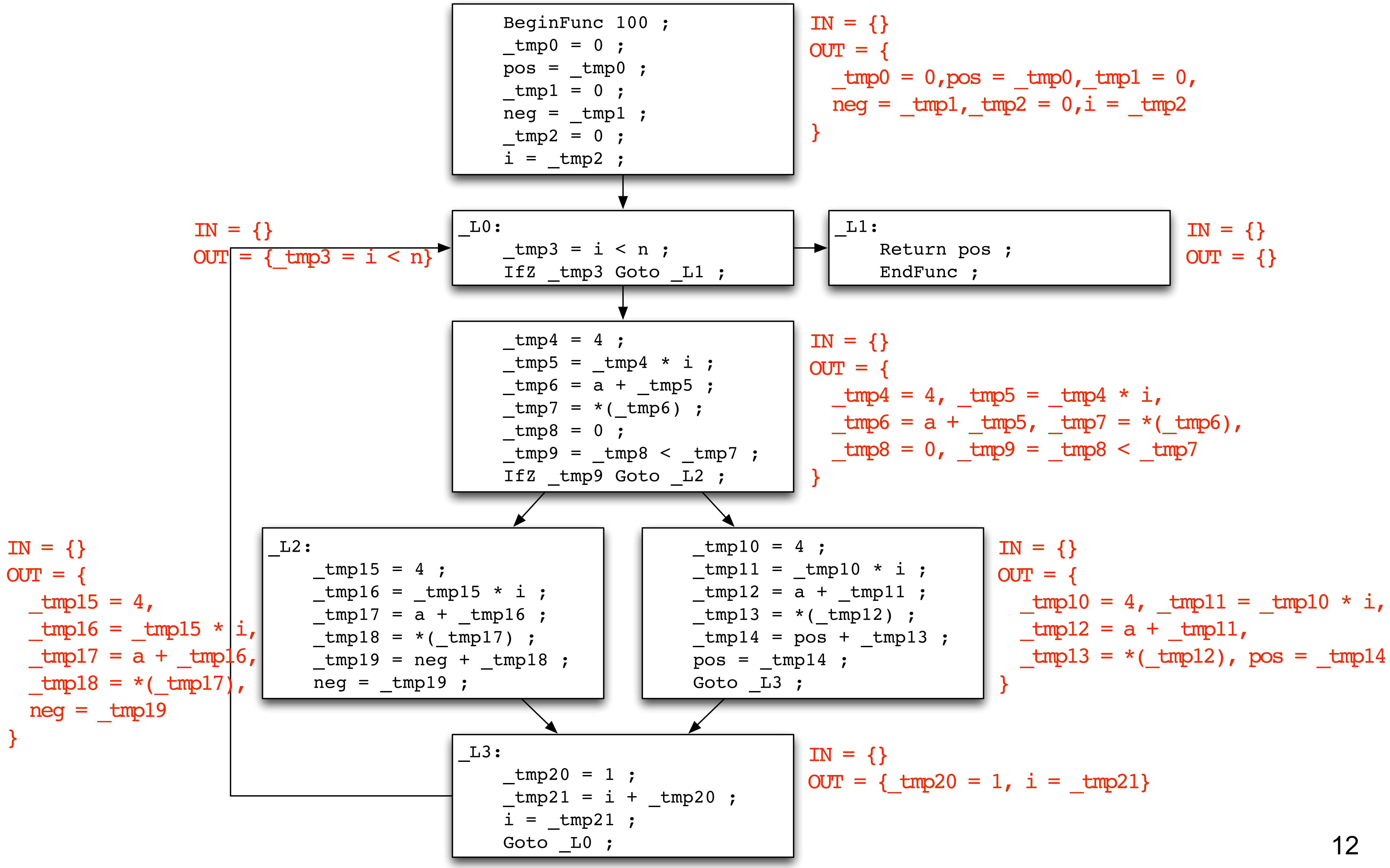
Local Available Expressions

```
OUT = IN = {}
_tmp20 = 1 ;
OUT = {_tmp20 = 1}
_tmp21 = i + _tmp20 ;
OUT = {_tmp20 = 1,_tmp21 = i + _tmp20}
i = _tmp21 ;
OUT = {_tmp20 = 1,i = _tmp21}
Goto _L0 ;
```

Local Available Expressions

```
OUT = IN = {}
_tmp20 = 1 ;
OUT = {_tmp20 = 1}
_tmp21 = i + _tmp20 ;
OUT = {_tmp20 = 1,_tmp21 = i + _tmp20}
i = _tmp21 ;
OUT = {_tmp20 = 1,i = _tmp21}
Goto _L0 ;
OUT = {_tmp20 = 1,i = _tmp21}
```

CFG w/ Local Available Expressions



```
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = 0 ;
    neg = _tmp1 ;
    _tmp2 = 0 ;
    i = _tmp2 ;
```

IN = {}
OUT = {
  _tmp0 = 0,pos = _tmp0,_tmp1 = 0,
  neg = _tmp1,_tmp2 = 0,i = _tmp2
}

```
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
```

IN = {}
OUT = {_tmp3 = i < n}

```
_L1:
    Return pos ;
    EndFunc ;
```

IN = {}
OUT = {}

```
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
```

IN = {}
OUT = {
  _tmp4 = 4, _tmp5 = _tmp4 * i,
  _tmp6 = a + _tmp5, _tmp7 = *(_tmp6),
  _tmp8 = 0, _tmp9 = _tmp8 < _tmp7
}

```
_L2:
    _tmp15 = 4 ;
    _tmp16 = _tmp15 * i ;
    _tmp17 = a + _tmp16 ;
    _tmp18 = *(_tmp17) ;
    _tmp19 = neg + _tmp18 ;
    neg = _tmp19 ;
```

IN = {}
OUT = {
  _tmp15 = 4,
  _tmp16 = _tmp15 * i,
  _tmp17 = a + _tmp16,
  _tmp18 = *(_tmp17),
  neg = _tmp19
}

```
    _tmp10 = 4 ;
    _tmp11 = _tmp10 * i ;
    _tmp12 = a + _tmp11 ;
    _tmp13 = *(_tmp12) ;
    _tmp14 = pos + _tmp13 ;
    pos = _tmp14 ;
    Goto _L3 ;
```

IN = {}
OUT = {
  _tmp10 = 4, _tmp11 = _tmp10 * i,
  _tmp12 = a + _tmp11,
  _tmp13 = *(_tmp12), pos = _tmp14
}

```
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
```

IN = {}
OUT = {_tmp20 = 1, i = _tmp21}

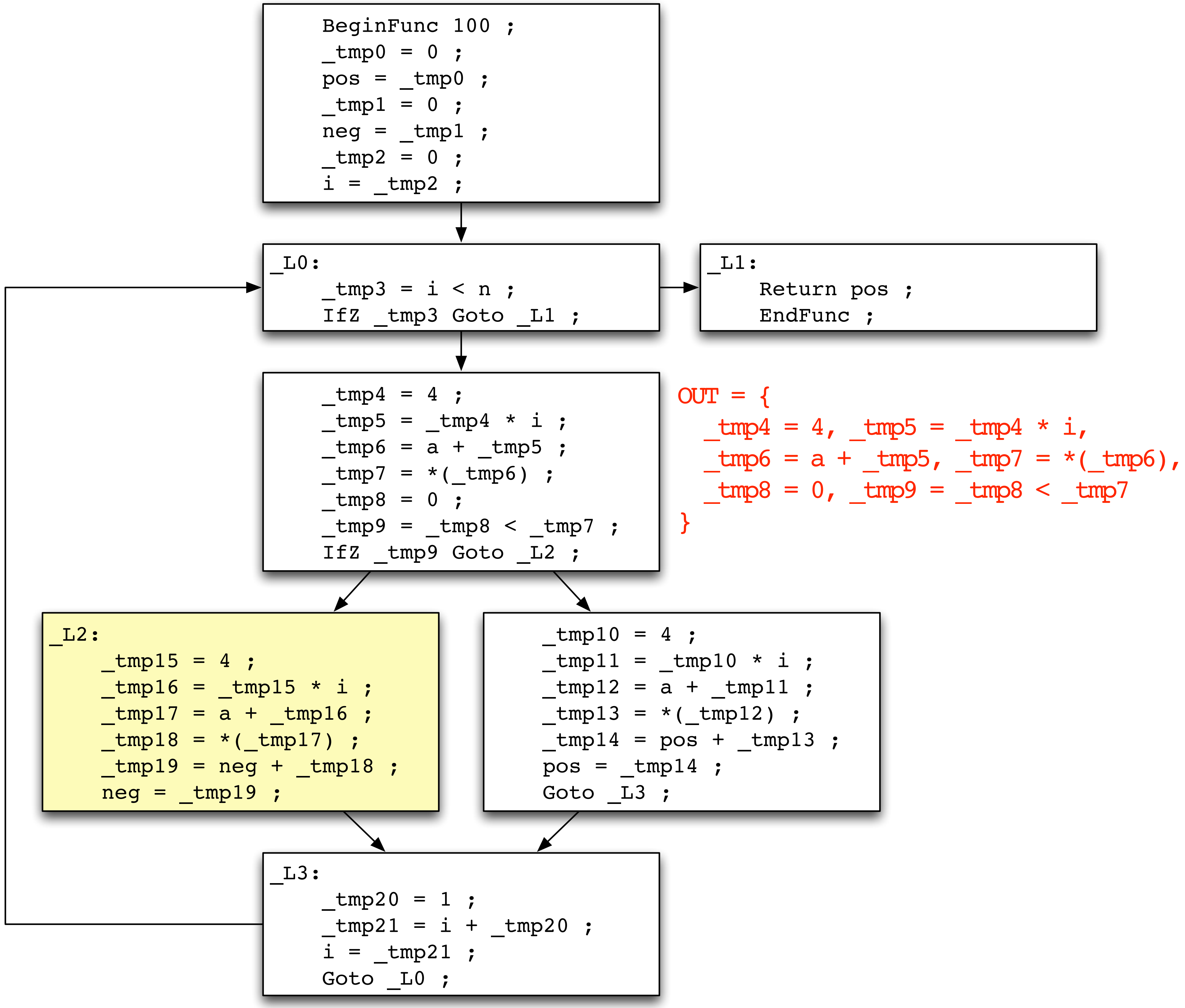
Global Available Expression Analysis

- Input
  - Control flow graph CFG
- Algorithm

```
for each basic block BB in CFG:
    OUT(BB) = {}
change = true
while(change):
    for each basic block BB in CFG:
        IN(BB) = Intersection(OUT(predecessors of BB))
        OUT(BB) = LocalAvailableExpr(BB, IN(BB))
        if any changes:
            change = true
```

- Output
  - IN(BB) = {available expressions before entering BB}
  - OUT(BB) = {available expressions after executing BB}

CFG w/ Available Expressions



```
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = 0 ;
    neg = _tmp1 ;
    _tmp2 = 0 ;
    i = _tmp2 ;
```

```
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
```

```
_L1:
    Return pos ;
    EndFunc ;
```

```
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
```

```
OUT = {
  _tmp4 = 4, _tmp5 = _tmp4 * i,
  _tmp6 = a + _tmp5, _tmp7 = *(_tmp6),
  _tmp8 = 0, _tmp9 = _tmp8 < _tmp7
}
```

```
_L2:
    _tmp15 = 4 ;
    _tmp16 = _tmp15 * i ;
    _tmp17 = a + _tmp16 ;
    _tmp18 = *(_tmp17) ;
    _tmp19 = neg + _tmp18 ;
    neg = _tmp19 ;
```

```
    _tmp10 = 4 ;
    _tmp11 = _tmp10 * i ;
    _tmp12 = a + _tmp11 ;
    _tmp13 = *(_tmp12) ;
    _tmp14 = pos + _tmp13 ;
    pos = _tmp14 ;
    Goto _L3 ;
```

```
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
```

Copy Propagation & Common Subexpression Elimination

```
_tmp15 = 4 ;

_tmp16 = _tmp15 * i ;

_tmp17 = a + _tmp16 ;

_tmp18 = *(_tmp17) ;

_tmp19 = neg + _tmp18 ;

neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
```

```
_tmp15 = 4 ;

_tmp16 = _tmp15 * i ;

_tmp17 = a + _tmp16 ;

_tmp18 = *(_tmp17) ;

_tmp19 = neg + _tmp18 ;

neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;

_tmp16 = _tmp15 * i ;

_tmp17 = a + _tmp16 ;

_tmp18 = *(_tmp17) ;

_tmp19 = neg + _tmp18 ;

neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp15 * i ;

_tmp17 = a + _tmp16 ;

_tmp18 = *(_tmp17) ;

_tmp19 = neg + _tmp18 ;

neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp4 * i ;

_tmp17 = a + _tmp16 ;

_tmp18 = *(_tmp17) ;

_tmp19 = neg + _tmp18 ;

neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;

_tmp17 = a + _tmp16 ;

_tmp18 = *(_tmp17) ;

_tmp19 = neg + _tmp18 ;

neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = a + _tmp16 ;


_tmp18 = *(_tmp17) ;


_tmp19 = neg + _tmp18 ;


neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = a + _tmp5 ;


_tmp18 = *(_tmp17) ;


_tmp19 = neg + _tmp18 ;


neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;

_tmp18 = *(_tmp17) ;

_tmp19 = neg + _tmp18 ;

neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6}
_tmp18 = *(_tmp17) ;


_tmp19 = neg + _tmp18 ;


neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6}
_tmp18 = *(_tmp6) ;


_tmp19 = neg + _tmp18 ;


neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6}
_tmp18 = _tmp7 ;


_tmp19 = neg + _tmp18 ;


neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6}
_tmp18 = _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6,_tmp18 = _tmp7}
_tmp19 = neg + _tmp18 ;


neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6}
_tmp18 = _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6,_tmp18 = _tmp7}
_tmp19 = neg + _tmp7 ;


neg = _tmp19 ;
```

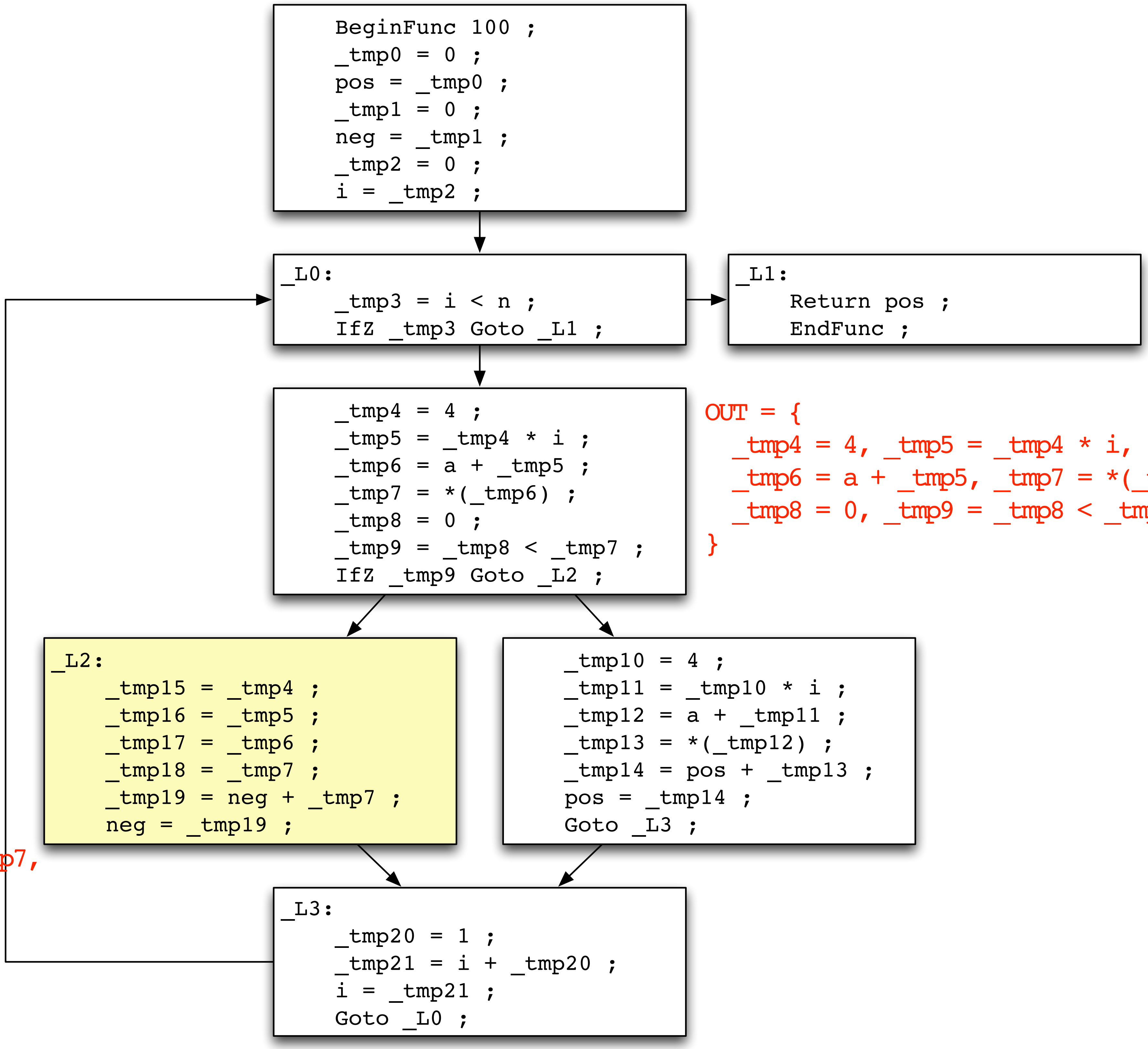
Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6}
_tmp18 = _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6,_tmp18 = _tmp7}
_tmp19 = neg + _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6,_tmp18 = _tmp7,_tmp19 = neg + _tmp7}
neg = _tmp19 ;
```

Copy Propagation & Common Subexpression Elimination

```
IN = Intersection(OUT(predecessors of BB))
   = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
OUT = IN
    = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7}
_tmp15 = _tmp4 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4}
_tmp16 = _tmp5 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5}
_tmp17 = _tmp6 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6}
_tmp18 = _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6,_tmp18 = _tmp7}
_tmp19 = neg + _tmp7 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6,_tmp18 = _tmp7,_tmp19 = neg + _tmp7}
neg = _tmp19 ;
OUT = {_tmp4 = 4,_tmp5 = _tmp4 * i,_tmp6 = a + _tmp5,_tmp7 = *(_tmp6),_tmp8 = 0,_tmp9 = _tmp8 < _tmp7,
       _tmp15 = _tmp4,_tmp16 = _tmp5,_tmp17 = _tmp6,_tmp18 = _tmp7,neg = _tmp19}
```

CFG w/ Available Expressions



```
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = 0 ;
    neg = _tmp1 ;
    _tmp2 = 0 ;
    i = _tmp2 ;
```

```
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
```

```
_L1:
    Return pos ;
    EndFunc ;
```

```
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
```

```
OUT = {
  _tmp4 = 4, _tmp5 = _tmp4 * i,
  _tmp6 = a + _tmp5, _tmp7 = *(_tmp6),
  _tmp8 = 0, _tmp9 = _tmp8 < _tmp7
}
```

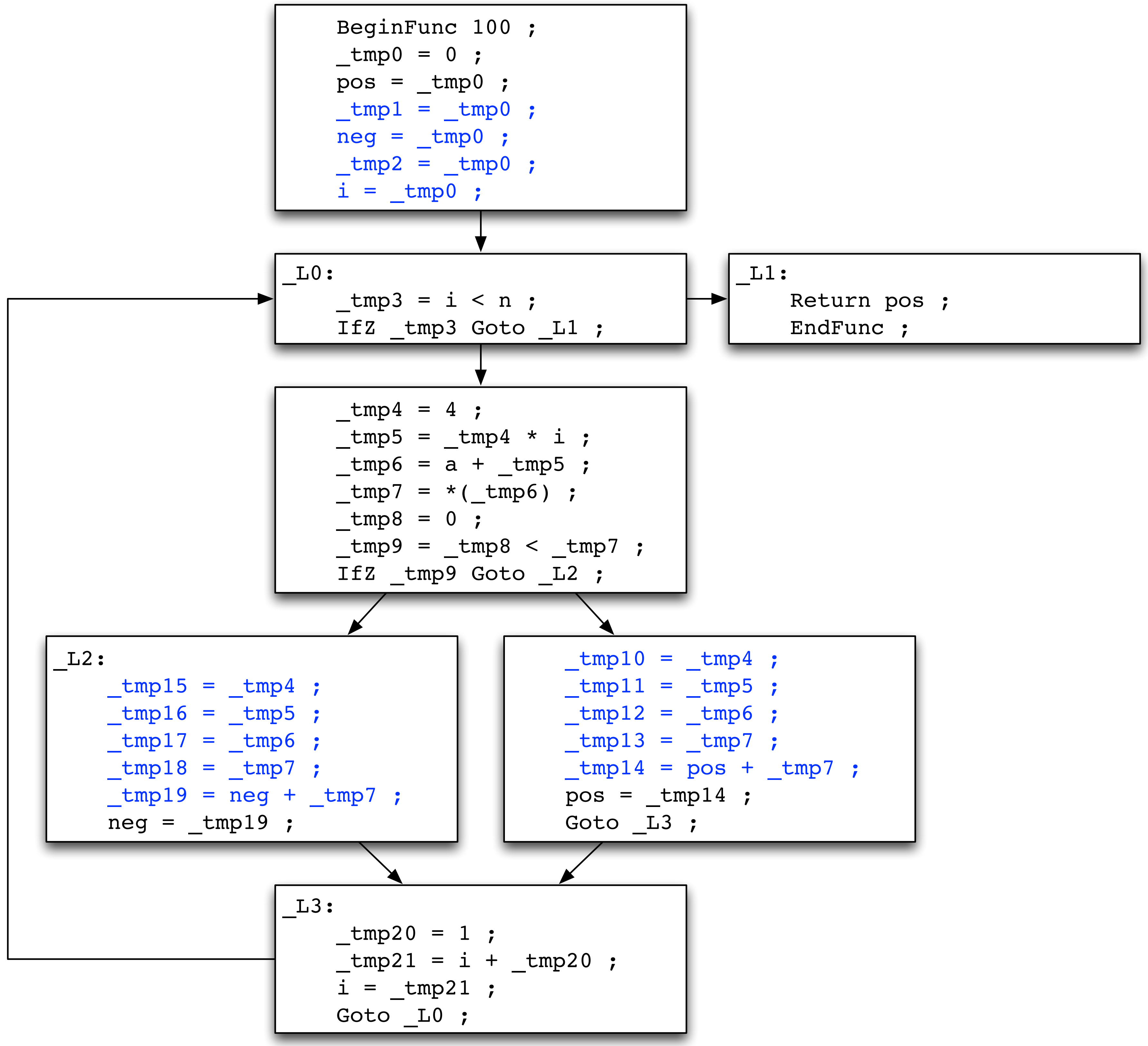
```
_L2:
    _tmp15 = _tmp4 ;
    _tmp16 = _tmp5 ;
    _tmp17 = _tmp6 ;
    _tmp18 = _tmp7 ;
    _tmp19 = neg + _tmp7 ;
    neg = _tmp19 ;
```

```
    _tmp10 = 4 ;
    _tmp11 = _tmp10 * i ;
    _tmp12 = a + _tmp11 ;
    _tmp13 = *(_tmp12) ;
    _tmp14 = pos + _tmp13 ;
    pos = _tmp14 ;
    Goto _L3 ;
```

```
OUT = {
  _tmp4 = 4,
  _tmp5 = _tmp4 * i,
  _tmp6 = a + _tmp5,
  _tmp7 = *(_tmp6),
  _tmp8 = 0,
  _tmp9 = _tmp8 < _tmp7,
  _tmp15 = _tmp4,
  _tmp16 = _tmp5,
  _tmp17 = _tmp6,
  _tmp18 = _tmp7,
  neg = _tmp19
}
```

```
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
```

Improved CFG



```
    BeginFunc 100 ;
    _tmp0 = 0 ;
    pos = _tmp0 ;
    _tmp1 = _tmp0 ;
    neg = _tmp0 ;
    _tmp2 = _tmp0 ;
    i = _tmp0 ;
```

```
_L0:
    _tmp3 = i < n ;
    IfZ _tmp3 Goto _L1 ;
```

```
_L1:
    Return pos ;
    EndFunc ;
```

```
    _tmp4 = 4 ;
    _tmp5 = _tmp4 * i ;
    _tmp6 = a + _tmp5 ;
    _tmp7 = *(_tmp6) ;
    _tmp8 = 0 ;
    _tmp9 = _tmp8 < _tmp7 ;
    IfZ _tmp9 Goto _L2 ;
```

```
_L2:
    _tmp15 = _tmp4 ;
    _tmp16 = _tmp5 ;
    _tmp17 = _tmp6 ;
    _tmp18 = _tmp7 ;
    _tmp19 = neg + _tmp7 ;
    neg = _tmp19 ;
```

```
    _tmp10 = _tmp4 ;
    _tmp11 = _tmp5 ;
    _tmp12 = _tmp6 ;
    _tmp13 = _tmp7 ;
    _tmp14 = pos + _tmp7 ;
    pos = _tmp14 ;
    Goto _L3 ;
```

```
_L3:
    _tmp20 = 1 ;
    _tmp21 = i + _tmp20 ;
    i = _tmp21 ;
    Goto _L0 ;
```

Local Liveness Analysis

- Input
  - Basic block BB
  - OUT = {live variables after executing BB}
- Algorithm

```
IN = OUT
for each TAC s in BB in backward order:
    G = variables consumed by s
    K = variables destroyed by s
    IN = IN - K + G
```

- Output
  - IN = {live variables required before entering BB}

Global Liveness Analysis

- Input
  - Control flow graph CFG
- Algorithm

```
for each basic block BB in CFG:
    IN(BB) = {}
change = true
while(change):
    for each basic block BB in CFG:
        OUT(BB) = Union(IN(successors of BB))
        IN(BB) = LocalLiveVar(BB, OUT(BB))
        if any changes:
            change = true
```

- Output
  - IN(BB) = {live variables required before entering BB}
  - OUT(BB) = {live variables after executing BB}

Comparison

	Available Expression Analysis	**Liveness Analysis**
Direction	Forward	Backward
Transfer Function	OUT = OUT + G - K	IN = IN - K + G
Meet Operator	IN(BB) = Intersection(OUT(predecessors of BB))	OUT(BB) = Union(IN(successors of BB))

Thanks & all the best!