

Example of SSA form optimization

Common subexpression elimination and aggressive optimization

Source program in C

```
/* cse_test.c */
#include <stdio.h>

int main ()
{
    int a;
    int b;
    int c;
    int d;
    int x;
    int y;
    int z;

    a = 1;
    b = 2;
    x = 3;

    y = a + b;
    c = x + y;

    if (a < 10)
        z = a + b; // a + b is a common subexpr, and is equal to y.
    else
        z = a + b; // a + b is a common subexpr, and is equal to y.
                    // therefore, z is equal to y after the if statement.
    d = x + z;     // z is equal to y. thus, x + z is equal to x + y, or c.
                    // as a result, d is equal to c.

    printf("%d\n",d); // d is equal to c
}
```

Optimized code after common subexpression elimination

Actually an assembly language program is generated, but for the reader's convenience, we show an equivalent C program generated using our "lir2c" option.

```
// C file generated by LIR-to-C after common subexpression elimination

void main(void) {
    int a_1_;
```

```
int b_2_;
int c_3_;
int d_4_;
int x_5_;
int y_6_;
int z_7_;
int returnvalue_8_;
int functionvalue_9_;
int a_1__0;
int a_1__1;
int b_2__0;
int b_2__1;
int x_5__0;
int x_5__1;
int y_6__0;
int y_6__1;
int c_3__0;
int c_3__1;
int z_7__0;
int z_7__1;
int z_7__2;
int z_7__3;
int d_4__0;
int d_4__1;
int functionvalue_9__0;
int functionvalue_9__1;
```

_L1:

```
a_1__0 = ((int)( 0));
b_2__0 = ((int)( 0));
x_5__0 = ((int)( 0));
y_6__0 = ((int)( 0));
c_3__0 = ((int)( 0));
z_7__0 = ((int)( 0));
d_4__0 = ((int)( 0));
functionvalue_9__0 = ((int)( 0));
goto _L2;
```

_L2:

```
a_1__1 = ((int)( 1));
b_2__1 = ((int)( 2));
x_5__1 = ((int)( 3));
y_6__1 = ((int)((((int)(a_1__1 + b_2__1)))));
c_3__1 = ((int)((((int)(x_5__1 + y_6__1)))));
if ((a_1__1 < 10)) { goto _L3;} else { goto _L4;}
```

_L3:

```
goto _L5;
```

_L4:

```
goto _L5;
```

_L5:

```
functionvalue_9__1 = printf((unsigned char *)&(string_10), c_3__1);
goto _L6;
```

_L6:

```
return;
```

```
}
```

Optimized code by enabling all SSA optimization flags

This is also a C program generated using our "lir2c" option.

```
// C file generated by LIR-to-C after all optimization

void main(void) {
    int a_1_;
    int b_2_;
    int c_3_;
    int d_4_;
    int x_5_;
    int y_6_;
    int z_7_;
    int returnvalue_8_;
    int functionvalue_9_;
    int a_1__0;
    int a_1__1;
    int b_2__0;
    int b_2__1;
    int x_5__0;
    int x_5__1;
    int y_6__0;
    int y_6__1;
    int c_3__0;
    int c_3__1;
    int z_7__0;
    int z_7__1;
    int z_7__2;
    int z_7__3;
    int d_4__0;
    int d_4__1;
    int functionvalue_9__0;
    int functionvalue_9__1;

_L1:
goto _L2;

_L2:
goto _L3;

_L3:
goto _L5;

_L5:
functionvalue_9__1 = printf((unsigned char *)&(string_10), 6);
goto _L6;

_L6:
return;
}
```

```
}
```

This is equivalent to the following C program in usual style.

```
void main(void) {  
    printf("%d\n", 6);  
}
```