EE4013 - C and Data Structures

G Krupateja

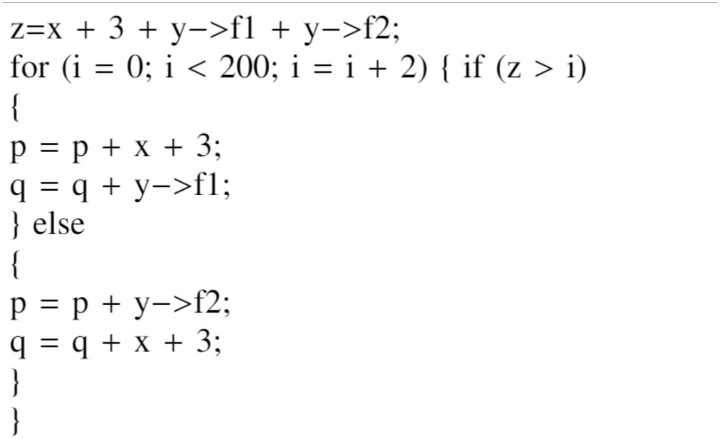
IIT Hyderabad

EE18BTECH11015

Overview

Problem

- Consider the following ANSI C Program.



```
z=x + 3 + y->f1 + y->f2;
for (i = 0; i < 200; i = i + 2) { if (z > i)
{
p = p + x + 3;
q = q + y->f1;
} else
{
p = p + y->f2;
q = q + x + 3;
}
}
```

Problem

- Assume that the variable y points to a struct (allocated on the heap) containing two fields f1 and f2, and the local variables x, y, z, p, q, and i are allotted registers. Common sub-expression elimination (CSE) optimization is applied on the code. The number of addition and the dereference operations (of the form y $\rightarrow$ f1 or $\rightarrow$ f2) in the optimized code, respectively, are:

Solution(Optimized Code)

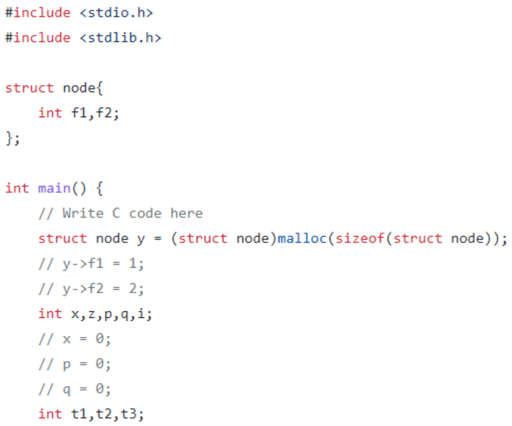
- Optimized code could be:

```
t1 = x + 3 // 1 addition
t2 = y->f1; // 1 dereference
t3 = y->f2; // 1 dereference
z = t1 + t2 + t3 // 2 additions
for (i = 0; i < 200; i += 2) {
    if (z > i) {
        p = p + t1; // 1 addition
        q = q + t2; // 1 addition
    } else {
        p = p + t3; // 1 addition
        q = q + t1; // 1 addition
    }
}
```

Solution continued

- Whether we take if or else block we get 2 additions, the loop runs exactly 2002=100 times, so from loop we get 2100=200 additions plus 100 additions for incrementing the value of i(i.e,(i+2)), before loop we had perform 3 additions, so total additions 303. We only do two de-reference outside the for loop, so total de-references =2. So, the number of additions and dereferences are 303 and 2 respectively.

C Code



```c
#include <stdio.h>
#include <stdlib.h>

struct node{
    int f1,f2;
};

int main() {
    // Write C code here
    struct node y = (struct node)malloc(sizeof(struct node));
    // y->f1 = 1;
    // y->f2 = 2;
    int x,z,p,q,i;
    // x = 0;
    // p = 0;
    // q = 0;
    int t1,t2,t3;
    t1 = x+3;
    t2 = y->f1;
    t3 = y->f2;

    z = t1+t2+t3;
    for(i=0;i<200;i++){
        if(z>i){
            p += t1;
            q += t3;
        }else{
            p += t3;
            q += t1;
        }
    }
    // printf("%d-%d-%d-%d-%d-%d\n",x,y->f1,y->f2,z,p,q);

    return 0;
}
```

The End