Shibo Xing

April 23, 2020

Dr. Xulon Tang

CS2210

Final Exam Answer Sheet



\_\_T\_\_ Initial values of global variables are part of the executable image file of a program. \_\_F\_\_ Dynamic type checking typically results in faster program execution.

\_\_T\_\_ A static type system can be designed to reject all bad programs and accept all good programs.

\_\_T\_\_ Optimizations done on a common low-level IR can be applied to all languages frontends and all machine backends of a compiler.

\_\_T\_\_ Performing GCSE before applying SSA and after applying SSA can generate different versions of a given program.

\_\_T\_\_ In SSA representation, a variable is assigned to only once in the IR.

\_\_T\_\_ Global compiler optimizations optimize the entire program across function boundaries.

\_\_F\_\_ The loop transformation matrix must be nonsingular.

\_\_F\_\_ A legal distance vector is a vector that is lexicographically positive.

2.

1. SDT defines the semantic actions which are done in the post-order traversal of the tree whereas SDD defines the semantic rules which imply no order to attribute evaluation.
2. x = (- a) \* b + F(–d) / G(e – f).

G1.val = - a.val

F1.val = -d.val

F2.val = e.val - f.val

E1.val = G1.val \* b.val

E2.val = F(F1.val) / G(F2.val)

E.val = E1.val + E2.val

c.

Since each integer takes 4 bytes (4 \* 0xFF),

Array A takes 24 bytes,

B takes 8 bytes,

C takes 48 bytes

|  |
| --- |
| 0xFA00 |
| ... |
| 0xFA18 |
| ... |
| 0xFA20 |
| ... |
| 0xFA50 |

A

B

C



E1.false = S.next

E2.false = S.next

S1.next = S.next

E1.true = newlabel()

E2.true = E1.true

S.code = E1.code || emit(E1.true’:’) || E2.code || emit(E2.true’:’) || S1.code;



backpatch(E1.holes\_true, E2.quad);

backpacth(E2.holes\_false, S1.holes\_next);

S1.holes\_next = S1.holes\_next

Avail:

Def: e1 = R2 \* R3

NKilled: e1, e3



R1 = R2 \* R3

s

L1:

R4 = R1/C

If (R4<W) goto L3

Avail: e1

Def: e2 = R1/C

Nkilled: e1, e2, e3

L2:

R5 = R2 \* R3

If (R5<W) goto L4

R6 = R1/C

goto L5

L3:

R7 = R2 \* R3

goto L6

Avail: e2

Def: e1 = R2 \* R3

e2 = R1 / C

Nkilled: e1, e2, e3

Avail: e1, e2

Def: e1 = R2 \* R3

Nkilled: e1, e2, e3

L4:

R2 = R3 / C

L5:

R9 = R1 / C

If (R6 < W) goto L2

L6:

halt

Avail: e2

Def:

Nkilled: e1, e2, e3

Avail: e1, e2

Def: e3 = R3 / C

Nkilled: e2, e3

Avail: e2

Def: e2 = R1 / C

Nkilled: e1, e2, e3

We can eliminate expression e2 in L3’s basic block and expression e2 in L5’s basic block



Distance vector: [1, -1]

B matrix is only read so there is no dependence within B matrix

If we visualize the loop dependence, we will a graph as the following

**j**

1 2 3

**i**

1

2

3

We can tell that there are no dependences along each row. All the dependences are across the rows. Moreover, in the triple loop example, the inner-most loop with index k is repeating some identical assignments to the entries on each row. Thus, we can completely erase the innermost loop, by performing a loop fusion and loop reversal, rewriting the loop structure into:

do i = 1,n

do j = 1,i+m

A(i,j) = A(i-1,j+1) + B(i,j-1)

enddo

enddo

Since we’ve concluded that no dependence exists along each row, we can re-arrange the iteration order of the inner loop. And also, now the inner loop is what we will parallelize.