编译原理 - 作业(5)： 代码生成与优化

Q1: (p408, Exercises 6.6.1) Add rules to the syntax-directed definition of Fig. 6.36 for the following control-flow constructs:

a. A repeat-statement **repeat** S **while** B.

b. A for-loop **for** (S_1; B; S_2) S_3.

PRODUCTION	SEMANTIC RULES
$P \to S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \to$ **assign**	$S.code =$ **assign**$.code$
$S \to$ **if** $(B)\ S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \to$ **if** $(B)\ S_1$ **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('goto'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$
$S \to$ **while** $(B)\ S_1$	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('goto'\ begin)$
$S \to S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

Figure 6.36: Syntax-directed definition for flow-of-control statements.

Production	Semantic Rules
a) $S \to$ repeat S_1 while B	$S_1.next = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S.code = label(B.true) \mid\mid S_1.code$ $\mid\mid label(S_1.next) \mid\mid B.code$
b) $S \to for(S_1; B; S_2)\ S_3$	$S_1.next = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_2.next = S_1.next$ $S_3.next = newlabel()$ $S.code = S_1.code$

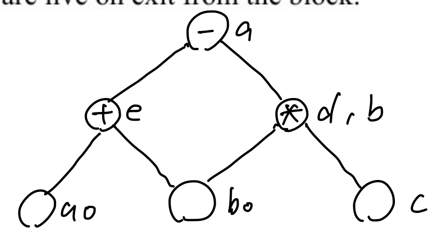
|| label (S_1.next) || B.code
|| label (B.true) || S_3.code
|| label (S_3.next) || S_2.code
|| gen ('goto' S.next)

Q2: (p541, Exercises 8.5.1&2) For the basic block

d = b * c
e = a + b
b = b * c
a = e - d

a. Construct the DAG of the block.
b. Simplify the three-address code of the block, assuming
   1). Only *a* is live on exit from the block.
   2). *a*, *b*, and *c* are live on exit from the block.

a)



b) 1) e = a + b
d = b * c
a = e - d

2) e = a + b
b = b * c
a = e - b

Q3: For the code segment below:

```
1:        x = 0
2:        y = 0
3:   L0: if n / 2 goto L1
4:        x = x + n
5:        y = y + 1
6:        goto L2
7:   L1: y = y + n
8:        c = 4 / 2
9:        t1 = x * c
10:       t2 = c - 1
11:       x = x + t2
12: L2: n = n -1
13:       if n > 0 goto L0
14:       return x
```

a. Partition the code segment into basic blocks.

b. Construct the control flow graph.

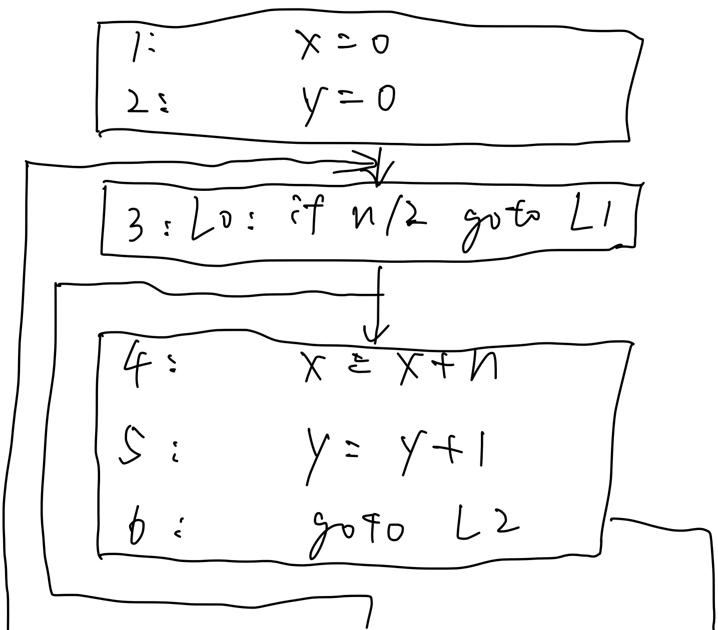
c. For lines 7-11, list two optimization techniques.

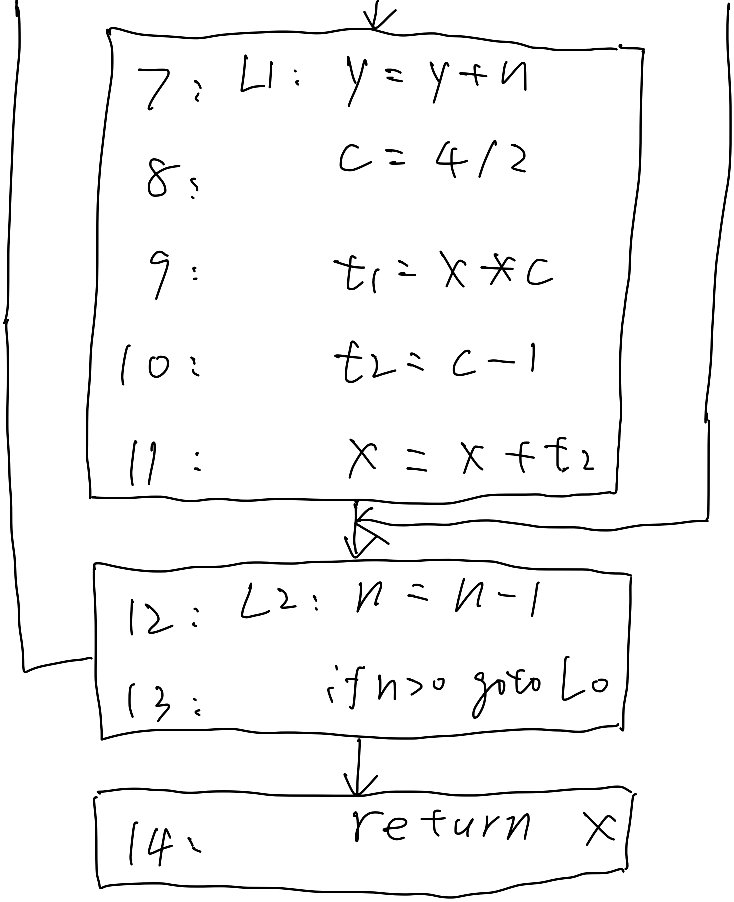
d. Suppose the whole segment is from a function '*int Func(int n)*', where *n* is the argument, *x* and *y* are local variables, then how to retrieve *n*, *x* and *y* when *Func()* is called in the final target code? Hint: consider \$fp and \$sp.

a. B1: 1~2 B2: 3 B3: 4~6

B4: 7~11 B5: 12~13 B6: 14

b.





c. 常量折叠 ① 与传播

$y = y + n$

$c = 2$

$t_1 = x * 2$

$t_2 = 1$

$x = x + 1$

② 删除无用代码 与强度削减

$y = y + n$

$t_1 = x << 1$

$x = x + 1$

注：若使用全局优化，可继续删除无用代码

即直接变为 $x = x+1$

d. 取出 n： $0(\$sp)$

取出 x： $-4(\$fp)$

取出 y： $-8(\$fp)$