ASSIGNMENT-7 COMPILERS

(Itish Agarwal, 18CS30021)

(26th Oct' 2020)

Q1. New quads : Old quads

New quads		Old quads
	100 :	t1 = 10 xxx
100 :	101 :	n = 10 ← def-use
	102 :	t2 = 0 xxx
101 :	103 :	i = 0 ← def-use
102 :	104 :	if i<n goto 109
103 :	105 :	goto 125
	106 :	t3 = i ← unused xxx
104 :	107 :	i = i + 1
105 :	108 :	goto 104
106 :	109 :	t4 = i<<2 (strength reduction)
107 :	110 :	t5 = a[t4]
108 :	111 :	t6 = t5 & 1 (strength reduction)
	112 :	t7 = 0 ← Unused xxx
109 :	113 :	if t6 = 0 goto 115
110 :	114 :	goto 120
111 :	115 :	t8 = i<<2
112 :	116 :	t9 = c + t8
	117 :	t10 = 0 ← Unused xxx

```
113 : 118 : *t9 = 0        ← def-use
114 : 119 : goto 106
115 : 120 : t11 = i<<2 (strength reduction)
116 : 121 : t12 = c + t11
        122 : t13 = 1      xxx
117 : 123 : *t12 = 1       ← def-use
118 : 124 : goto 106
119 : 125 : return
```

Three-Address Code after Peephole Optimization:

```
100: n = 10
101: i = 0
102: if i<n, goto 106
103: goto 119
104: i = i+1
105: goto 102
106: t4 = i<<2
107: t5 = a[t4]
108: t6 = t5 & 1
109: if t6=0 goto 111
110: goto 115
111: t8 = i<<2
112: t9 = c + t18
113: *t9 = 0
114: goto 104
```

```
115 :   t11 = i<<2
116 :   t12 = c + t11
117 :   *t12 = 1

118 :   goto 104
119 :   return
```

Q2. (a) Lines of code that prepare the stack and registers for use within the function:

```
push ebp        (Push base pointer on the stack)
move ebp, esp   (Update base pointer to the base of the callee function frame)
sub esp, 8      (Make space to store variables on the stack)
push edi        (pushing the registers)
push esi
:
& so on.
```

(b) PTO

(b) Lines of code which restore the stack and registers to the state they were in before the function was called:

```
pop esi        (pop in reverse order of push)
pop edi

mov esp, ebp

pop ebp        (restore caller's base pointer value)
ret
```

Q3. Live variables

	Live variables
000:	// a, n
001: count = 0	// a, n, count
002: i = 0	// a, n, count, i
003: L0: if i<n goto L2	// a, n, count, i
004: goto L3	// a, n, count, i
005: L1: i = i+1	// a, n, count, i
006: goto L0	// a, n, count, i
007: L2: t0 = 4 * i	// a, n, count, i, t0
008: t1 = a[t0]	// a, n, count, i, t0, t1
009: t2 = t1 % 2	// a, n, count, i, t1, t2
010: if t2 != 0 goto L1	// a, n, count, i, t2
011: count = count + 1	// a, n, count, i
012: goto L1	// a, n, count, i

013: L3: return count // coun