Alias Analysis

15-411/15-611 Compiler Design

Seth Copen Goldstein

November 12, 2019

Constant Propagation on Memory

$$l_1 : M[q] \leftarrow 4$$
$$l_2 : M[p] \leftarrow 8128$$
$$l_3 : x \leftarrow M[q]$$

- Can we replace l_3 with **x ← 4**?

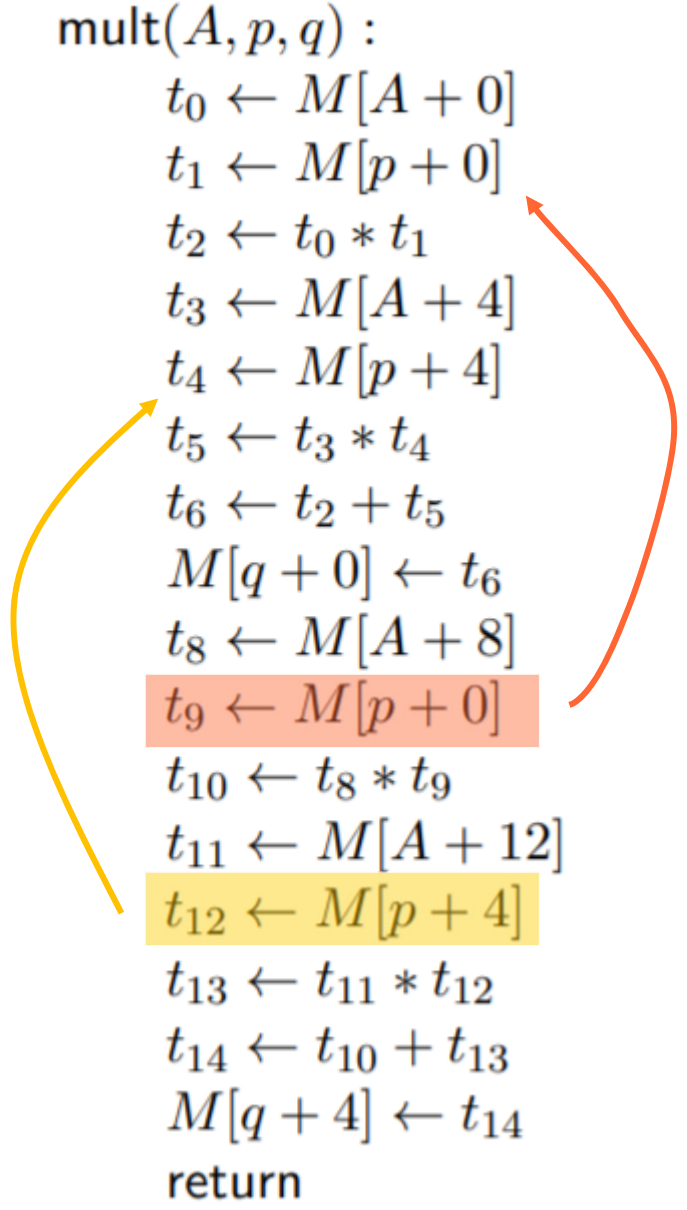
CSE

```
foo() {
        int a,k;
        extern int *q;
        ...
        ... // maybe &a or &k
        ...
        k = a+6;
        f(a,&k);
        *q = 13;
        k = a+6
        ...
}
```

CSE on Memory

```
struct point {
  int x;
  int y;
};
typedef struct point pt;

void mult(int[] A, pt* p, pt* q) {
  q->x = A[0] * p->x + A[1] * p->y;
  q->y = A[2] * p->x + A[3] * p->y;
  return;
}
```



Dependence Information

```
foo(int[] a, int[] b, int n) {
     int i;
     for (i=1; i<n; i++) {
          a[i] = b[i-1]+a[i]
     }
}
```

- What are the dependence distances?

```
struct intlist {
        int val;
        struct intlist* next;
};

struct int foo() {
      struct intlist* p;
      struct intlist* q;
      int a = 1;

      q = alloc(struct intlist);
      q->val = 0;
      q->next NULL;
      p = alloc(struct intlist);
      p->val = 0;
      p->next = q;
      q->val = 5;
      a = p->val;
      return a;
}
```

```
1:  MOVE   t2        <- 0                          p
2:  MOVE   t3        <- 0                          q
3:  MOVE   t4        <- 1                          a
4:  CALL   t5        <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1        <- t5,    0
6:  STORE  MEM[u1]   <-  0                         q->val = 0
7:  PLUS   u2        <- t5,    4
8:  STORE  MEM[u2]   <-  0                         q->next = 0
9:  MOVE   t3        <- t5                         q
10: CALL   t6        <- malloc(8)                  p <- alloc
11: PLUS   u3        <- t6,    0
12: STORE  MEM[u3]   <-  0                         p->val = 0
13: PLUS   u4        <- t6,    4
14: STORE  MEM[u4]   <- t3                         p->next = q
15: MOVE   t2        <- t6                         p
16: PLUS   u5        <- t3,    0
17: STORE  MEM[u5]   <-  5                         q->val = 5
18: PLUS   u6        <- t2,    0                   p
19: LOAD   t4        <- MEM[u6]                    a
20: RET              t4
```

```
1:  MOVE   t2        <- 0                          p
2:  MOVE   t3        <- 0                          q
3:  MOVE   t4        <- 1                          a
4:  CALL   t5        <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1        <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]   <-  0                         q->val = 0
7:  PLUS   u2        <- t5,    4
8:  STORE  MEM[u2]   <-  0                         q->next = 0
9:  MOVE   t3        <- t5                         q
10: CALL   t6        <- malloc(8)                  p <- alloc
11: PLUS   u3        <- t6,    0
12: STORE  MEM[u3]   <-  0                         p->val = 0
13: PLUS   u4        <- t6,    4
14: STORE  MEM[u4]   <- t3                         p->next = q
15: MOVE   t2        <- t6                         p
16: PLUS   u5        <- t3,    0
17: STORE  MEM[u5]   <-  5                         q->val = 5
18: PLUS   u6        <- t2,    0                   p
19: LOAD   t4        <- MEM[u6]                    a
20: RET              t4
```

```
1:  MOVE   t2        <- 0                          p
2:  MOVE   t3        <- 0                          q
3:  MOVE   t4        <- 1                          a
4:  CALL   t5        <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1        <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]   <-  0                         q->val = 0
7:  PLUS   u2        <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]   <-  0                         q->next = 0
9:  MOVE   t3        <- t5                         q
10: CALL   t6        <- malloc(8)                  p <- alloc
11: PLUS   u3        <- t6,    0
12: STORE  MEM[u3]   <-  0                         p->val = 0
13: PLUS   u4        <- t6,    4
14: STORE  MEM[u4]   <- t3                         p->next = q
15: MOVE   t2        <- t6                         p
16: PLUS   u5        <- t3,    0
17: STORE  MEM[u5]   <-  5                         q->val = 5
18: PLUS   u6        <- t2,    0                   p
19: LOAD   t4        <- MEM[u6]                    a
20: RET              t4
```

```
1:  MOVE   t2        <- 0                          p
2:  MOVE   t3        <- 0                          q
3:  MOVE   t4        <- 1                          a
4:  CALL   t5        <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1        <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]   <-  0                         q->val = 0
7:  PLUS   u2        <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]   <-  0                         q->next = 0
9:  MOVE   t3        <- t5           (t3,4,0)      q
10: CALL   t6        <- malloc(8)                  p <- alloc
11: PLUS   u3        <- t6,    0
12: STORE  MEM[u3]   <-  0                         p->val = 0
13: PLUS   u4        <- t6,    4
14: STORE  MEM[u4]   <- t3                         p->next = q
15: MOVE   t2        <- t6                         p
16: PLUS   u5        <- t3,    0
17: STORE  MEM[u5]   <-  5                         q->val = 5
18: PLUS   u6        <- t2,    0                   p
19: LOAD   t4        <- MEM[u6]                    a
20: RET              t4
```

```
1:  MOVE   t2         <- 0                              p
2:  MOVE   t3         <- 0                              q
3:  MOVE   t4         <- 1                              a
4:  CALL   t5         <- malloc(8)     (t5,4,0)         q <- alloc
5:  PLUS   u1         <- t5,    0      (u1,4,0)
6:  STORE  MEM[u1]    <-  0                             q->val = 0
7:  PLUS   u2         <- t5,    4      (u2,4,4)
8:  STORE  MEM[u2]    <-  0                             q->next = 0
9:  MOVE   t3         <- t5            (t3,4,0)         q
10: CALL   t6         <- malloc(8)     (t6,10,0)        p <- alloc
11: PLUS   u3         <- t6,    0
12: STORE  MEM[u3]    <-  0                             p->val = 0
13: PLUS   u4         <- t6,    4
14: STORE  MEM[u4]    <- t3                             p->next = q
15: MOVE   t2         <- t6                             p
16: PLUS   u5         <- t3,    0
17: STORE  MEM[u5]    <-  5                             q->val = 5
18: PLUS   u6         <- t2,    0                       p
19: LOAD   t4         <- MEM[u6]                        a
20: RET               t4
```

```
1:  MOVE   t2         <- 0                          p
2:  MOVE   t3         <- 0                          q
3:  MOVE   t4         <- 1                          a
4:  CALL   t5         <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1         <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]    <-  0                         q->val = 0
7:  PLUS   u2         <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]    <-  0                         q->next = 0
9:  MOVE   t3         <- t5           (t3,4,0)      q
10: CALL   t6         <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3         <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]    <-  0                         p->val = 0
13: PLUS   u4         <- t6,    4
14: STORE  MEM[u4]    <- t3                         p->next = q
15: MOVE   t2         <- t6                         p
16: PLUS   u5         <- t3,    0
17: STORE  MEM[u5]    <-  5                         q->val = 5
18: PLUS   u6         <- t2,    0                   p
19: LOAD   t4         <- MEM[u6]                    a
20: RET               t4
```

```
1:  MOVE   t2        <- 0                          p
2:  MOVE   t3        <- 0                          q
3:  MOVE   t4        <- 1                          a
4:  CALL   t5        <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1        <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]   <-  0                         q->val = 0
7:  PLUS   u2        <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]   <-  0                         q->next = 0
9:  MOVE   t3        <- t5           (t3,4,0)      q
10: CALL   t6        <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3        <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]   <-  0                         p->val = 0
13: PLUS   u4        <- t6,    4     (u4,10,4)
14: STORE  MEM[u4]   <- t3                         p->next = q
15: MOVE   t2        <- t6                         p
16: PLUS   u5        <- t3,    0
17: STORE  MEM[u5]   <-  5                         q->val = 5
18: PLUS   u6        <- t2,    0                   p
19: LOAD   t4        <- MEM[u6]                    a
20: RET              t4
```

```
1:  MOVE   t2         <- 0                          p
2:  MOVE   t3         <- 0                          q
3:  MOVE   t4         <- 1                          a
4:  CALL   t5         <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1         <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]    <-  0                         q->val = 0
7:  PLUS   u2         <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]    <-  0                         q->next = 0
9:  MOVE   t3         <- t5           (t3,4,0)      q
10: CALL   t6         <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3         <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]    <-  0                         p->val = 0
13: PLUS   u4         <- t6,    4     (u4,10,4)
14: STORE  MEM[u4]    <- t3                         p->next = q
15: MOVE   t2         <- t6           (t2,10,0)     p
16: PLUS   u5         <- t3,    0
17: STORE  MEM[u5]    <-  5                         q->val = 5
18: PLUS   u6         <- t2,    0                   p
19: LOAD   t4         <- MEM[u6]                    a
20: RET               t4
```

```
1:  MOVE   t2         <- 0                          p
2:  MOVE   t3         <- 0                          q
3:  MOVE   t4         <- 1                          a
4:  CALL   t5         <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1         <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]    <-  0                         q->val = 0
7:  PLUS   u2         <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]    <-  0                         q->next = 0
9:  MOVE   t3         <- t5           (t3,4,0)      q
10: CALL   t6         <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3         <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]    <-  0                         p->val = 0
13: PLUS   u4         <- t6,    4     (u4,10,4)
14: STORE  MEM[u4]    <- t3                         p->next = q
15: MOVE   t2         <- t6           (t2,10,0)     p
16: PLUS   u5         <- t3,    0     (u5,4,0)
17: STORE  MEM[u5]    <-  5                         q->val = 5
18: PLUS   u6         <- t2,    0                   p
19: LOAD   t4         <- MEM[u6]                    a
20: RET               t4
```

```
1:  MOVE   t2        <- 0                          p
2:  MOVE   t3        <- 0                          q
3:  MOVE   t4        <- 1                          a
4:  CALL   t5        <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1        <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]   <-  0                         q->val = 0
7:  PLUS   u2        <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]   <-  0                         q->next = 0
9:  MOVE   t3        <- t5           (t3,4,0)      q
10: CALL   t6        <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3        <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]   <-  0                         p->val = 0
13: PLUS   u4        <- t6,    4     (u4,10,4)
14: STORE  MEM[u4]   <- t3                         p->next = q
15: MOVE   t2        <- t6           (t2,10,0)     p
16: PLUS   u5        <- t3,    0     (u5,4,0)
17: STORE  MEM[u5]   <-  5                         q->val = 5
18: PLUS   u6        <- t2,    0     (u6,4,0)      p
19: LOAD   t4        <- MEM[u6]                    a
20: RET              t4
```

```
1:  MOVE   t2         <- 0                          p
2:  MOVE   t3         <- 0                          q
3:  MOVE   t4         <- 1                          a
4:  CALL   t5         <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1         <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]    <-  0                         q->val = 0
7:  PLUS   u2         <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]    <-  0                         q->next = 0
9:  MOVE   t3         <- t5           (t3,4,0)      q
10: CALL   t6         <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3         <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]    <-  0                         p->val = 0
13: PLUS   u4         <- t6,    4     (u4,10,4)
14: STORE  MEM[u4]    <- t3                         p->next = q
15: MOVE   t2         <- t6           (t2,10,0)     p
16: PLUS   u5         <- t3,    0     (u5,4,0)
17: STORE  MEM[u5]    <-  5                         q->val = 5
18: PLUS   u6         <- t2,    0     (u6,10,0)     p
19: LOAD   t4         <- MEM[u6]                    a
20: RET               t4
```

```
1:  MOVE   t2         <- 0                          p
2:  MOVE   t3         <- 0                          q
3:  MOVE   t4         <- 1                          a
4:  CALL   t5         <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1         <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]    <-  0                         q->val = 0
7:  PLUS   u2         <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]    <-  0                         q->next = 0
9:  MOVE   t3         <- t5           (t3,4,0)      q
10: CALL   t6         <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3         <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]    <-  0                         p->val = 0
13: PLUS   u4         <- t6,    4     (u4,10,4)
14: STORE  MEM[u4]    <- t3                         p->next = q
15: MOVE   t2         <- t6           (t2,10,0)     p
16: PLUS   u5         <- t3,    0     (u5,4,0)
17: STORE  MEM[u5]    <-  5                         q->val = 5
18: PLUS   u6         <- t2,    0     (u6,10,0)     p
19: MOVE   t4         <- 0                          a
20: RET               t4
```

```
1:  MOVE   t2        <- 0                          p
2:  MOVE   t3        <- 0                          q
3:  MOVE   t4        <- 1                          a
4:  CALL   t5        <- malloc(8)    (t5,4,0)      q <- alloc
5:  PLUS   u1        <- t5,    0     (u1,4,0)
6:  STORE  MEM[u1]   <-  0                         q->val = 0
7:  PLUS   u2        <- t5,    4     (u2,4,4)
8:  STORE  MEM[u2]   <-  0                         q->next = 0
9:  MOVE   t3        <- t5           (t3,4,0)      q
10: CALL   t6        <- malloc(8)    (t6,10,0)     p <- alloc
11: PLUS   u3        <- t6,    0     (u3,10,0)
12: STORE  MEM[u3]   <-  0                         p->val = 0
13: PLUS   u4        <- t6,    4     (u4,10,4)
14: STORE  MEM[u4]   <- t3                         p->next = q
15: MOVE   t2        <- t6           (t2,10,0)     p
16: PLUS   u5        <- t3,    0     (u5,4,0)
17: STORE  MEM[u5]   <-  5                         q->val = 5
18: PLUS   u6        <- t2,    0     (u6,10,0)     p
19: MOVE   t4        <- 0                          a
20: RET              0
```

```
struct intlist* foo() {
      struct intlist* p;
      struct intlist* q;
      int a = 1;

      p = alloc(struct intlist);
      p->val = 0;
      p->next = 0;

      q = alloc(struct intlist);
      q->val = 6;
      q->next = p;

      if (a == 0) p = q;
      p->val = 4;
      return p;
}
```

```
1:  CALL   t2       <- malloc(8)              p
2:  PLUS   u1       <- t2,0
3:  STR    MEM[u1] <-  0                     p->val = 0
4:  PLUS   u2       <- t2, 4
5:  STR    MEM[u2] <- 0                      p->next = 0
6:  MOVE   t3       <- t2                     x=p
7:  CALL   t4       <- malloc(8)              q
8:  PLUS   u3       <- t4,0
9:  STR    MEM[u3] <- 6                      q->val = 6
10: PLUS   u4       <- t4,0
11: STR    MEM[u4] <- t3                     q->next = p
12: MOVE   t5       <- t4                     q
13: MOVE   t6       <- 1                      a
14: CJUMP  t6 == 0, ifT0,ifF1                 a == 0 ?
ifT0:
16: MOVE   t3'      <- t5                     x=q
ifF1:
18: PHI    t3''     <- ϕ(t3,t3')
19: PLUS   u5       <- t3'', 0
20: STR    MEM[u5] <- 4
21: RET    t2
```

```
1:  CALL   t2       <- malloc(8) (t2,1,0)      p
2:  PLUS   u1       <- t2,0
3:  STR    MEM[u1] <-  0                       p->val = 0
4:  PLUS   u2       <- t2, 4
5:  STR    MEM[u2] <- 0                        p->next = 0
6:  MOVE   t3       <- t2                      x=p
7:  CALL   t4       <- malloc(8)               q
8:  PLUS   u3       <- t4,0
9:  STR    MEM[u3] <- 6                        q->val = 6
10: PLUS   u4       <- t4,0
11: STR    MEM[u4] <- t3                       q->next = p
12: MOVE   t5       <- t4                      q
13: MOVE   t6       <- 1                       a
14: CJUMP  t6 == 0, ifT0,ifF1                  a == 0 ?
ifT0:
16: MOVE   t3'      <- t5                      x=q
ifF1:
18: PHI    t3''     <- ϕ(t3,t3')
19: PLUS   u5       <- t3'', 0
20: STR    MEM[u5] <- 4
21: RET    t2
```

```
1:  CALL   t2       <- malloc(8) (t2,1,0)      p
2:  PLUS   u1       <- t2,0      (u1,1,0)
3:  STR    MEM[u1]  <-  0                      p->val = 0
4:  PLUS   u2       <- t2, 4     (u2,1,4)
5:  STR    MEM[u2]  <- 0                       p->next = 0
6:  MOVE   t3       <- t2                      x=p
7:  CALL   t4       <- malloc(8)               q
8:  PLUS   u3       <- t4,0
9:  STR    MEM[u3]  <- 6                       q->val = 6
10: PLUS   u4       <- t4,0
11: STR    MEM[u4]  <- t3                      q->next = p
12: MOVE   t5       <- t4                      q
13: MOVE   t6       <- 1                       a
14: CJUMP  t6 == 0, ifT0,ifF1                  a == 0 ?
ifT0:
16: MOVE   t3'      <- t5                      x=q
ifF1:
18: PHI    t3''     <- ϕ(t3,t3')
19: PLUS   u5       <- t3'', 0
20: STR    MEM[u5]  <- 4
21: RET    t2
```

```
1:  CALL   t2       <- malloc(8) (t2,1,0)      p
2:  PLUS   u1       <- t2,0      (u1,1,0)
3:  STR    MEM[u1]  <-  0                      p->val = 0
4:  PLUS   u2       <- t2, 4     (u2,1,4)
5:  STR    MEM[u2]  <- 0                       p->next = 0
6:  MOVE   t3       <- t2        (t3,1,0)      x=p
7:  CALL   t4       <- malloc(8) (t4,7,0)      q
8:  PLUS   u3       <- t4,0      (u3,7,0)
9:  STR    MEM[u3]  <- 6                       q->val = 6
10: PLUS   u4       <- t4,0      (u4,7,0)
11: STR    MEM[u4]  <- t3                      q->next = p
12: MOVE   t5       <- t4        (t5,7,0)      q
13: MOVE   t6       <- 1                       a
14: CJUMP  t6 == 0, ifT0,ifF1                  a == 0 ?
ifT0:
16: MOVE   t3'      <- t5                      x=q
ifF1:
18: PHI    t3''     <- ϕ(t3,t3')
19: PLUS   u5       <- t3'', 0
20: STR    MEM[u5]  <- 4
21: RET    t2
```

```
1:  CALL   t2       <- malloc(8) (t2,1,0)      p
2:  PLUS   u1       <- t2,0      (u1,1,0)
3:  STR    MEM[u1]  <-  0                      p->val = 0
4:  PLUS   u2       <- t2, 4     (u2,1,4)
5:  STR    MEM[u2]  <- 0                       p->next = 0
6:  MOVE   t3       <- t2        (t3,1,0)      x=p
7:  CALL   t4       <- malloc(8) (t4,7,0)      q
8:  PLUS   u3       <- t4,0      (u3,7,0)
9:  STR    MEM[u3]  <- 6                       q->val = 6
10: PLUS   u4       <- t4,0      (u4,7,0)
11: STR    MEM[u4]  <- t3                      q->next = p
12: MOVE   t5       <- t4        (t5,7,0)      q
13: MOVE   t6       <- 1                       a
14: CJUMP  t6 == 0, ifT0,ifF1                  a == 0 ?
ifT0:
16: MOVE   t3'      <- t5        (t3',7,0)     x=q
ifF1:
18: PHI    t3''     <- ϕ(t3,t3')
19: PLUS   u5       <- t3'', 0
20: STR    MEM[u5]  <- 4
21: RET    t2
```

```
1:  CALL   t2       <- malloc(8) (t2,1,0)      p
2:  PLUS   u1       <- t2,0      (u1,1,0)
3:  STR    MEM[u1]  <-  0                      p->val = 0
4:  PLUS   u2       <- t2, 4     (u2,1,4)
5:  STR    MEM[u2]  <- 0                       p->next = 0
6:  MOVE   t3       <- t2        (t3,1,0)      x=p
7:  CALL   t4       <- malloc(8) (t4,7,0)      q
8:  PLUS   u3       <- t4,0      (u3,7,0)
9:  STR    MEM[u3]  <- 6                       q->val = 6
10: PLUS   u4       <- t4,0      (u4,7,0)
11: STR    MEM[u4]  <- t3                      q->next = p
12: MOVE   t5       <- t4        (t5,7,0)      q
13: MOVE   t6       <- 1                       a
14: CJUMP  t6 == 0, ifT0,ifF1                  a == 0 ?
ifT0:
16: MOVE   t3'      <- t5        (t3',7,0)     x=q
ifF1:
18: PHI    t3''     <- ϕ(t3,t3') (t3'',7,0),(t3'',1,0)
19: PLUS   u5       <- t3'', 0
20: STR    MEM[u5]  <- 4
21: RET    t2
```

```
1:  CALL   t2       <- malloc(8) (t2,1,0)      p
2:  PLUS   u1       <- t2,0      (u1,1,0)
3:  STR    MEM[u1]  <-  0                      p->val = 0
4:  PLUS   u2       <- t2, 4     (u2,1,4)
5:  STR    MEM[u2]  <- 0                       p->next = 0
6:  MOVE   t3       <- t2        (t3,1,0)      x=p
7:  CALL   t4       <- malloc(8) (t4,7,0)      q
8:  PLUS   u3       <- t4,0      (u3,7,0)
9:  STR    MEM[u3]  <- 6                       q->val = 6
10: PLUS   u4       <- t4,0      (u4,7,0)
11: STR    MEM[u4]  <- t3                      q->next = p
12: MOVE   t5       <- t4        (t5,7,0)      q
13: MOVE   t6       <- 1                       a
14: CJUMP  t6 == 0, ifT0,ifF1                  a == 0 ?
ifT0:
16: MOVE   t3'      <- t5        (t3',7,0)     x=q
ifF1:
18: PHI    t3''     <- ϕ(t3,t3') (t3'',7,0),(t3'',1,0)
19: PLUS   u5       <- t3'', 0   (u5,7,0),(u5,1,0)
20: STR    MEM[u5]  <- 4
21: RET    t2
```

NEW COURSE: LANGUAGE DESIGN & PROTOTYPING

17-396/17-696 – SPRING 2020

Little languages are everywhere! Would you like to – or do you need to – design your own?

In this course, you will:

- Learn how to **critique a language design**
- Practice several **language prototyping** approaches (interpreters, transpilers, fluent APIs)
- Apply techniques for **evaluating language designs with users**
- **Design and prototype your own language** in the final project

Prof. Jonathan Aldrich – T/Th 3-4:20

http://www.cs.cmu.edu/~aldrich/courses/17-396/