

## Plan for Today

---

### Converting 3-address code to Assem(MIPS)

- chart with copy involving temps, binary op, param, call, branch, and label

### Lvalue versus Rvalue

CS453 Lecture

Lvalues and Rvalues

1

## A Low-Level IR: 3-address code

---

### 3-address code

- Linear representation
- Typically language-independent
- Nearly corresponds to machine instructions

### Example operations

- Copy `x = y, t1 = t2`
- Indexed copy `x = y[i], x[i] = y, t1 = y[i]`
- Unary op `x = op y`
- Binary op `x = y op z, t1 = t2 op t3`
- Address of `p = & y`
- Load `x = *p`
- Store `*p = y,`
- Pass param `param x_1`
- Call `t1 = call f, 1`
- Branch `goto L1`
- Cbranch `if (x==y) goto L1`

CS453 Lecture

Lvalues and Rvalues

2

## Assem intermediate representation

---

### Assem.Instr

- "assembly language instruction without register assignments"

### OPER(String assem, List<Temp> dst, List<Temp> src, List<Label> jumps)

- contains a string with holes for registers indicated by `d# and `s# and holes for labels indicated by `j#
- dst and src are lists of Temps whose register assignment should fill holes
  - first entry in src is associated with `s0, second with `s1, etc.
  - first entry in dst is associated with `d0, etc.
- jumps is a list of labels for filling in label holes

CS453 Lecture

Lvalues and Rvalues

3

## Assem intermediate representation cont ...

---

### LABEL(String assem, Label label)

- a label statement in the target code

### MOVE(String assem, Temp dst, Temp src)

- similar to OPER in that assem string contains holes, but ..
  - no jumps
  - only one src and dst Temp

### CJUMP(String a, Temp.Temp src1, RELOP op, Temp.Temp src2, Temp.Label t, Temp.Label f)

- similar to OPER in that assem string contains holes, but ..
  - only jumps to true and false target
  - only two source Temps for comparison
  - explicit conditional operation, which enables later changes in code layout

CS453 Lecture

Lvalues and Rvalues

4

## Lvalues versus Rvalues

---

### Lvalue

- "result of an expression that can occur on the left of an assignment statement"
- examples: `*(&a)`, `b.membervar`, `p->membervar`

### Rvalue

- an expression whose result can only appear as a subexpression or on the rhs of a statement
- examples: `&a`, `3*4`, `new Foo()`

### Why?

- explains compiler errors you might see in the future, e.g. non-lvalue assignment
  - `x+3 = 5`
  - `4 = 5`
- emphasizes the difference between equality in math and assignment in programming languages
- think about where values are stored during the progression of a program
- what if we could pass around user defined types (not just references to them)?