This handout summarizes a simple TAC intermediate language. There are many choices as to the exact instructions to include in such a language, and you will probably want to modify and extend this variant when we translate IC programs into TAC.

### Instruction Forms

- **Arithmetic and Logic Instructions.**

  The basic instruction forms are:

  \[
  a = b \, \text{OP} \, c \quad a = \text{OP} \, b
  \]

  where \( \text{OP} \) can be

  - an arithmetic operator: \( \text{ADD, SUB, DIV, MUL} \)
  - a logic operator: \( \text{AND, OR, XOR} \)
  - a comparison operator: \( \text{EQ, NEQ, LE, LEQ, GE, GEQ} \)
  - a unary operator: \( \text{MINUS, NEG} \)

- **Data Movement Instructions.**

  Copy:

  \[
  a = b
  \]

  Load/store:

  \[
  a = *b \quad *a = b
  \]

  Array load/store:

  \[
  a = b[i] \quad a[i] = b
  \]

  Field load/store:

  \[
  a = b.f \quad a.f = b
  \]

- **Branch Instructions.**

  Label:

  \[
  \text{label L}
  \]

  Unconditional jump:

  \[
  \text{jump L}
  \]

  Conditional jump:

  \[
  \text{cjump a L} \quad (\text{jump to L if a is true})
  \]

- **Function Call Instructions.**

  Call with no result:

  \[
  \text{call f(a_1, \ldots, a_n)}
  \]

  Call with result:

  \[
  a = \text{call f(a_1, \ldots, a_n)}
  \]

  (Note: there is no explicit TAC representation for parameter passing, stack frame setup, etc.)