- TAC instructions must be represented in such a way that further processing and optimization of code are possible

  - replacing code with simpler code

  - moving code around

- Two alternatives are quite common:

  1. Quadruples: a TAC instruction has three addresses and one op.

  ```
  result := x op y
  ```

  2. Triples: result location is implied by position of the instruction.

```
    a:= a*b+c*d

t1 := a*b

t2:= c*d

a:= t1+t2 or (t3:= t1+t2; a:= t3)?
```

pos	op	arg1	arg2	res
0	*	a	b	t1
1	*	c	d	t2
2	+	t1	t2	a

pos	op	arg1	arg2
0	*	a	b
1	*	c	d
2	+	(0)	(1)
3	copy	a	(2)

In quadruples, temporary variables are part of name space (hence go into the symbol table). In triples, they are not represented in name space; only refer to IC locations and their contents.

- How to represent `x[i]:=y` ? 1 entry in quadruples, 2 entries (but one instruction) in triples.

  - `[]=` is array l-value op; `=[]` is r-value.

  quad : `0 []= i y x`

  triple: `0 []= x i`

  `1 y`

  what about `y:=x[i]` ? 2 instructions in triples

  `0 =[] x i`

  `1 copy y (0)`

Which one is preferable?

- Quadruples can be moved around

- Triples must change when moved (one solution is indirect triples)