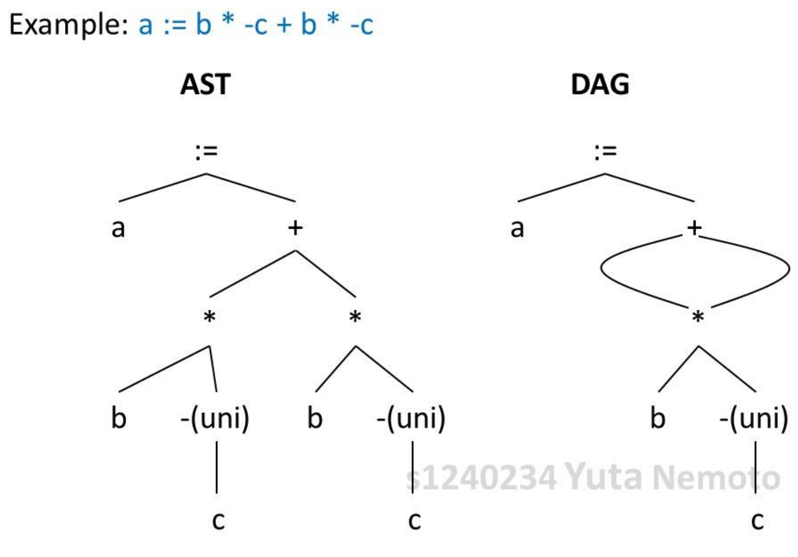
Quiz 11

1. **Write briefly about intermediate representation?**
    - Similar terms: Intermediate representation, intermediate language
    - Ties the front and back ends together
    - Language and Machine neutral
    - Many forms
    - More than one intermediate language may be used by a compiler

2. **What are the intermediate language types?**
    - **Graphical IRs:**
        - **Abstract Syntax trees (AST):** retain essential structure of the parse tree, eliminating unneeded nodes.
        - **Directed Acyclic Graphs (DAG):** compacted AST to avoid duplication – smaller footprint as well.
        - **Control Flow Graphs (CFG):** explicitly model control flow.
        - etc.
    - **Linear IRs:**
        - **Stack based (postfix)**
        - **Three address code (quadruples)**
        - etc.

3. **Compare with example between AST and DAG?**



With the DAG graph, we can manage the part of b * -c within one group. Graph is quite simple compared to the AST graph, and it means we can improve the efficiency with this and save the memory usage on the program.

On the other hand, with the AST tree structure, we have to describe the b * -c part twice, because tree structure doesn't allow multi line for one part. It's not efficient compared to DAG.

4. **Write about stack based intermediate representations?**

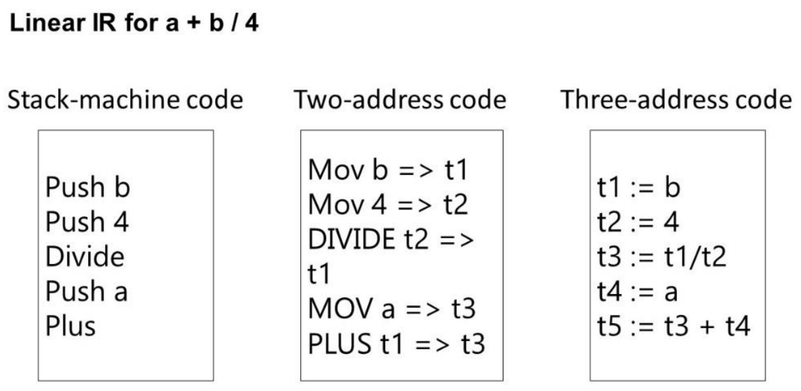
Low level IL before final code generation

- A linear sequence of low-level instructions
- Resemble assembly code for an abstract machine
  - ✧ Explicit conditional branches and goto jumps
- Reflect instruction sets of the target machine
  - ✧ Stack-machine code and three address code
- Implemented as a collection (table or list) of tuples

5. **What are the three address code instructions?**

- **Assignment**
  - x j= y op z
  - x = op y
  - x = y
- **Indexed assignment**
  - x = y[i]
  - x[i] = y
- **Pointer**
  - x = &y
  - x = *y
  - *x = y
- **Function**
  - param x
  - call p, n
  - return y
- **Jump**
  - goto L
  - if x relop y goto L

6. **Give the linear representation for the statement: a + b / 4?**



Linear IR for a + b / 4

Stack-machine code

```
Push b
Push 4
Divide
Push a
Plus
```

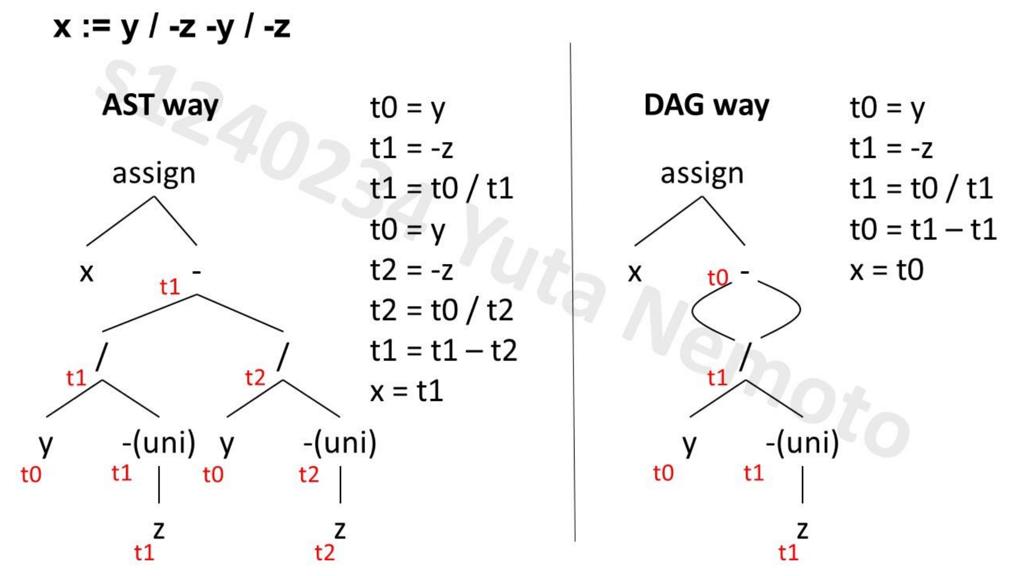
Two-address code

```
Mov b => t1
Mov 4 => t2
DIVIDE t2 =>
t1
MOV a => t3
PLUS t1 => t3
```

Three-address code

```
t1 := b
t2 := 4
t3 := t1/t2
t4 := a
t5 := t3 + t4
```

7. Generate intermediate code for the input statement: x := y / -z –y / -z?



x := y / -z -y / -z

AST way

```
t0 = y
t1 = -z
t1 = t0 / t1
t0 = y
t2 = -z
t2 = t0 / t2
t1 = t1 – t2
x = t1
```

DAG way

```
t0 = y
t1 = -z
t1 = t0 / t1
t0 = t1 – t1
x = t0
```