Chapter 14

Extending S2

println and print productions

```
printlnStatement → "println" "(" printlnArg ")" ";"

printStatement   →   "print" "(" printArg ")" ";"
```

where printlnArg and printArg are defined with

```
printlnArg → expr
printlnArg → <STRING>
printlnArg → λ

printArg   → expr
printArg   → <STRING>
```

In place of printlnArg (and its corresponding method in the parser), we can represent the arguments for the println statement with printArg|λ. Our printlnStatement production then becomes

```
printlnStatement → "println" "(" (printArg|λ) ")" ";"
```

printlnStatement()

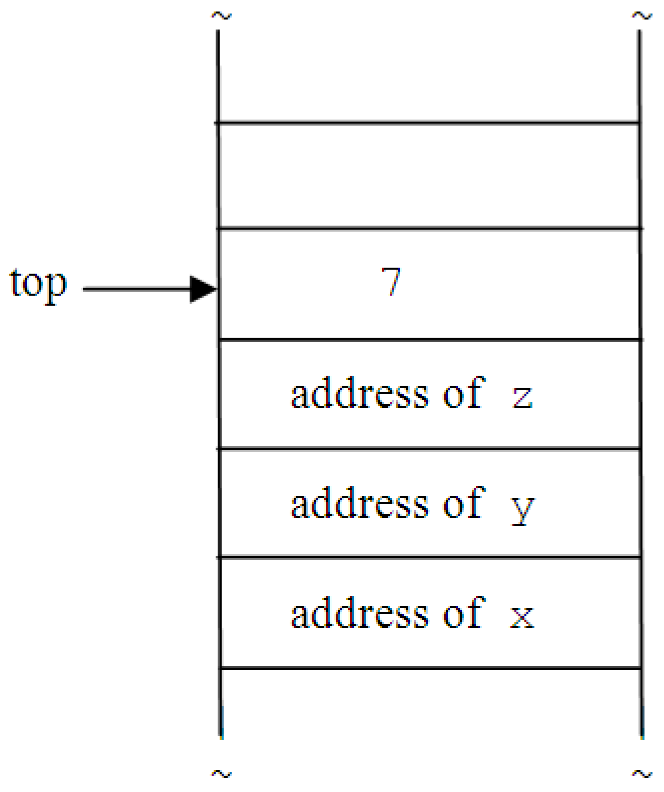
```
void printlnStatement(): {}
{
    "println"
    "("

     // this paren starts a list of alternatives
    (
       printArg()           // first alternative
     |
       {}                   // second alternative
    ) // this paren ends the list of alternatives

    {codeGen.emitInstruction("pc", "'\\n'");}
    {codeGen.emitInstruction("aout");}
    ")"
    ";"
}
```

Cascaded assignment statement

```
void assignmentStatement(): {Token t;}
{
        t=<ID>
        {symTab.enter(t.image, "0");}
        {codeGen.emitInstruction("pc", t.image);}
        "="
        assignmentTail()
        {codeGen.emitInstruction("stav");}
}
//------------------------------
void assignmentTail(): {Token t;}
{
        LOOKAHEAD(2)  // <------lookahead specified here
        t=<ID>
        {symTab.enter(t.image, "0");}
        {codeGen.emitInstruction("pc", t.image);}
        "="
        assignmentTail()
        {codeGen.emitInstruction("dupe");}
        {codeGen.emitInstruction("rot");}
        {codeGen.emitInstruction("stav");}
  |
        expr()
        ";"
}
```

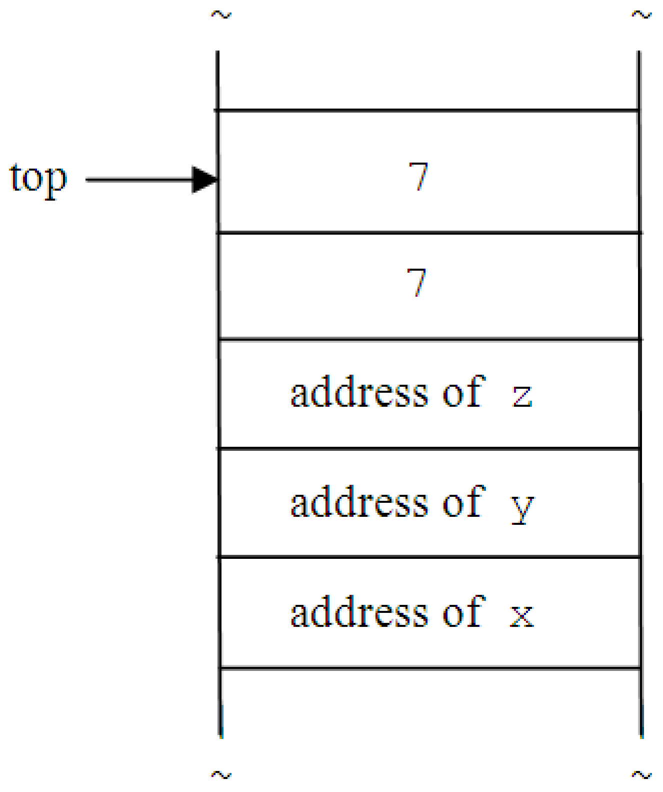
Stack cascaded assignment statement

a) before dupe



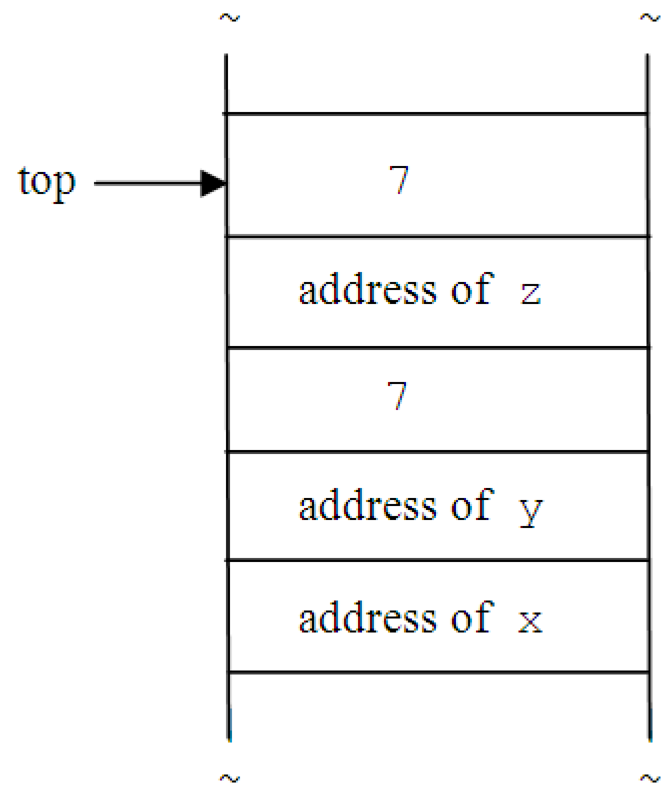
Duplicate value

b) after `dupe`, before `rot`



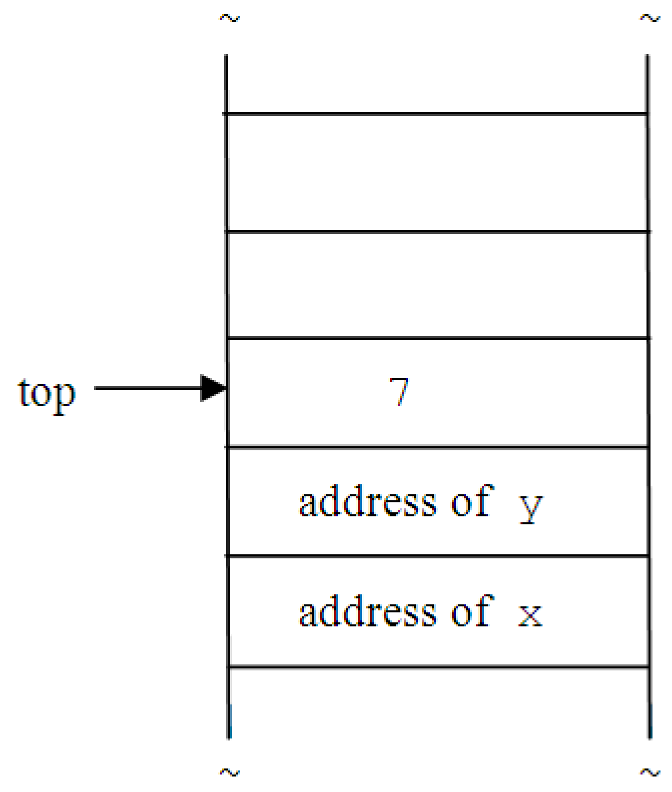
Reposition value

c) after `rot`, before `stav`



Perform the assign

d) after `stav`



Unary minus

```
void factor(): {Token t;}
{
   t=<UNSIGNED>
   {cg.emitInstruction("pwc", t.image);}
 |
   t=<ID>
   {st.enter(t.image);}
   {cg.emitInstruction("p", t.image);}
 |
   "(" expr() ")"
 |
   "+"
   factor()
 |
   "-"
   (
      t=<UNSIGNED>
      {cg.emitInstruction("pwc", "-" + t.image);}
    |
      t=<ID>
      {st.enter(t.image);}
      {cg.emitInstruction("p", t.image);}
      {cg.emitInstruction("neg");}
    |
      "("
      expr()
      ")"
      {cg.emitInstruction("neg");}
    |
      "+"
      factor()
      {cg.emitInstruction("neg");}
    |
      "-"
      factor()
   )
}
```

readint Statement

```
statement         → readintStatement
readintStatement → "readint" "(" <ID> ")" ";"
```

```
readint(x);

   pc    x
   din
   stav
```