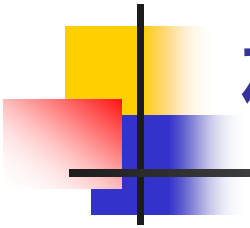
代码生成：栈式计算机

编译原理

华保健

bjhua@ustc.edu.cn

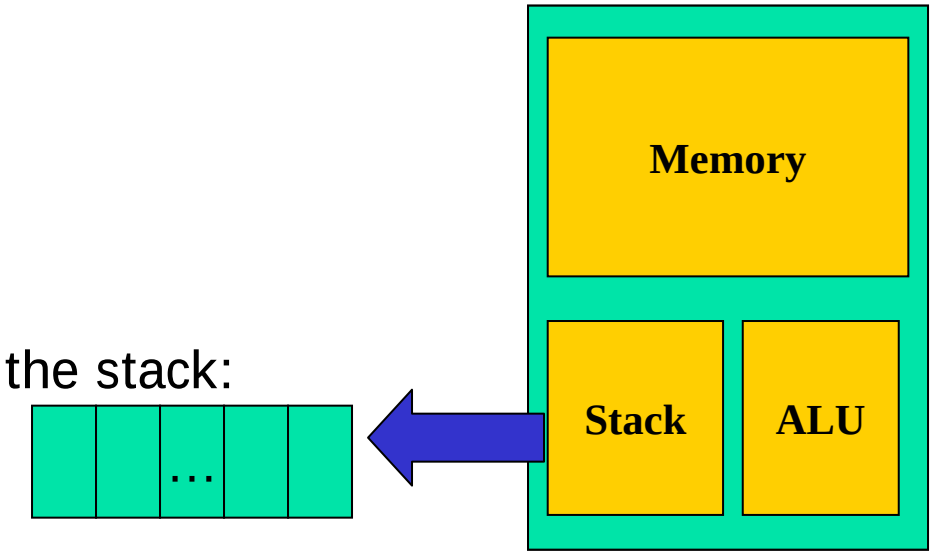


栈式计算机

- 栈式计算机在历史上非常流行
  - 上世纪70年代曾有很多栈式计算机
- 但今天已经基本上已经退出了历史舞台
  - 效率问题
- 我们还要讨论栈式计算机的代码生成技术的原因是：
  - 给栈式计算机生成代码是最容易的
  - 仍然有许多栈式的虚拟机
    - Pascal P code
    - Java virtual machine (JVM)
    - Postscript
    - ...

栈式计算机Stack的结构

- 内存
  - 存放所有的变量
- 栈
  - 进行运算的空间
- 执行引擎
  - 指令的执行



栈计算机的指令集

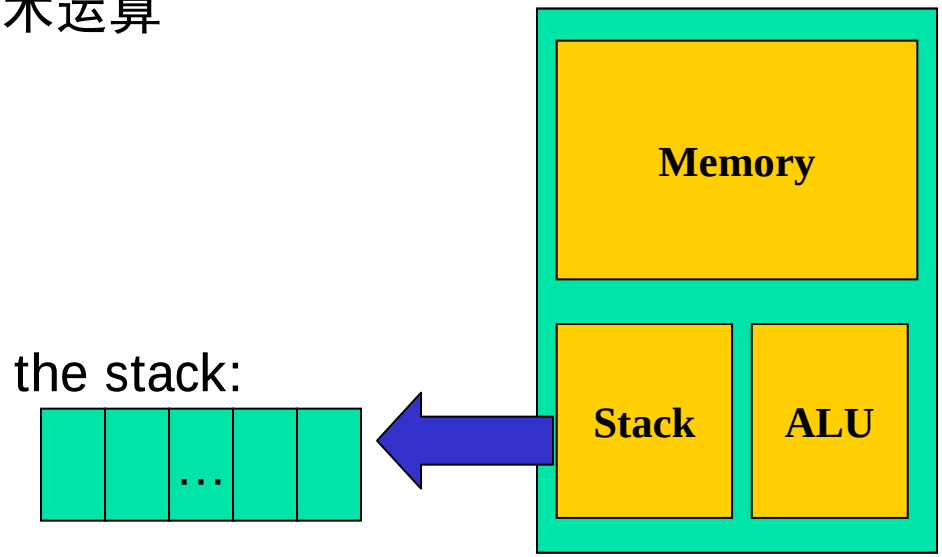
```
// 指令的语法
s -> push NUM
   | load x
   | store x
   | add
   | sub
   | times
   | div
```

栈操作

访存

算术运算

是Java字节码的一个子集！

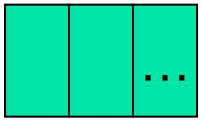


指令的语义：push

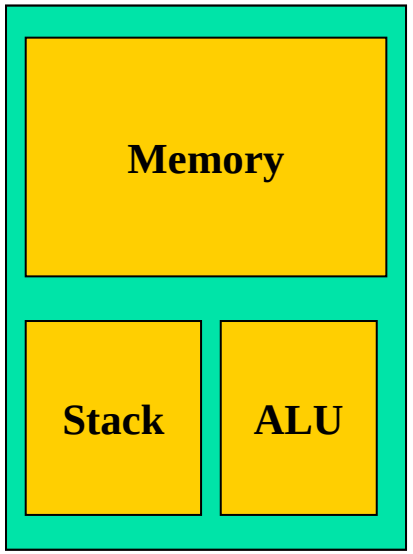
```
// 指令的语法
s -> push NUM
   | load x
   | store x
   | add
   | sub
   | times
   | div
```

```
push NUM:
   top++;
   stack[top] = NUM;
```

执行前：



执行后：

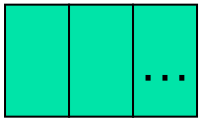


指令的语义：load x

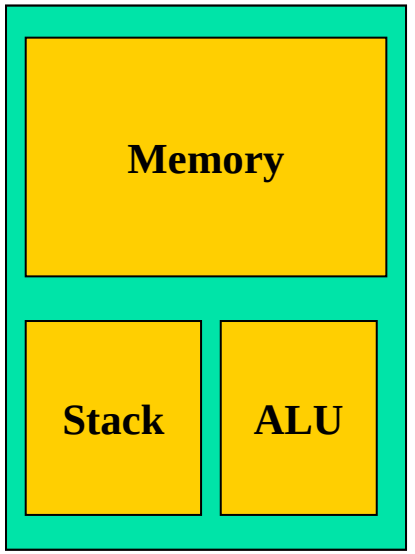
```
// 指令的语法
s -> push NUM
   | load x
   | store x
   | add
   | sub
   | times
   | div
```

```
load x:
   top++;
   stack[top] = x;
```

执行前：



执行后：

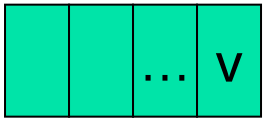


指令的语义：store x

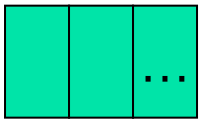
```
// 指令的语法
s -> push NUM
   | load x
   | store x
   | add
   | sub
   | times
   | div
```

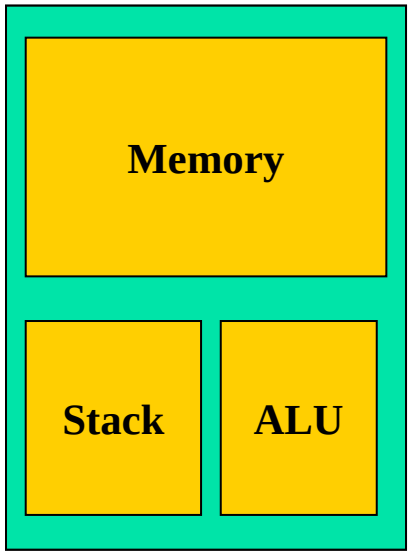
```
store x:
   x = stack[top];
   top--;
```

执行前：



执行后：



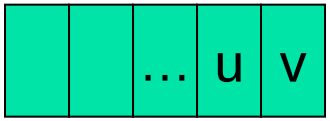


指令的语义：add

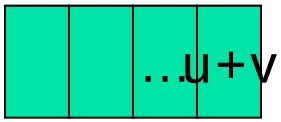
```
// 指令的语法
s -> push NUM
   | load x
   | store x
   | add
   | sub
   | times
   | div
```

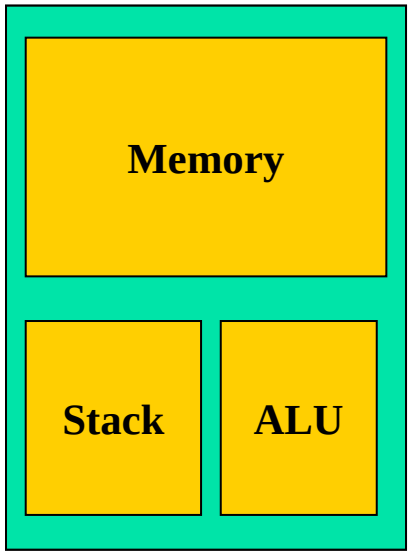
```
add:
   temp = stack[top-1]
              +stack[top];
   top -= 2;
   push temp;
```

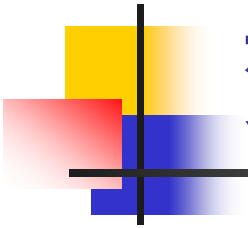
执行前：



执行后：



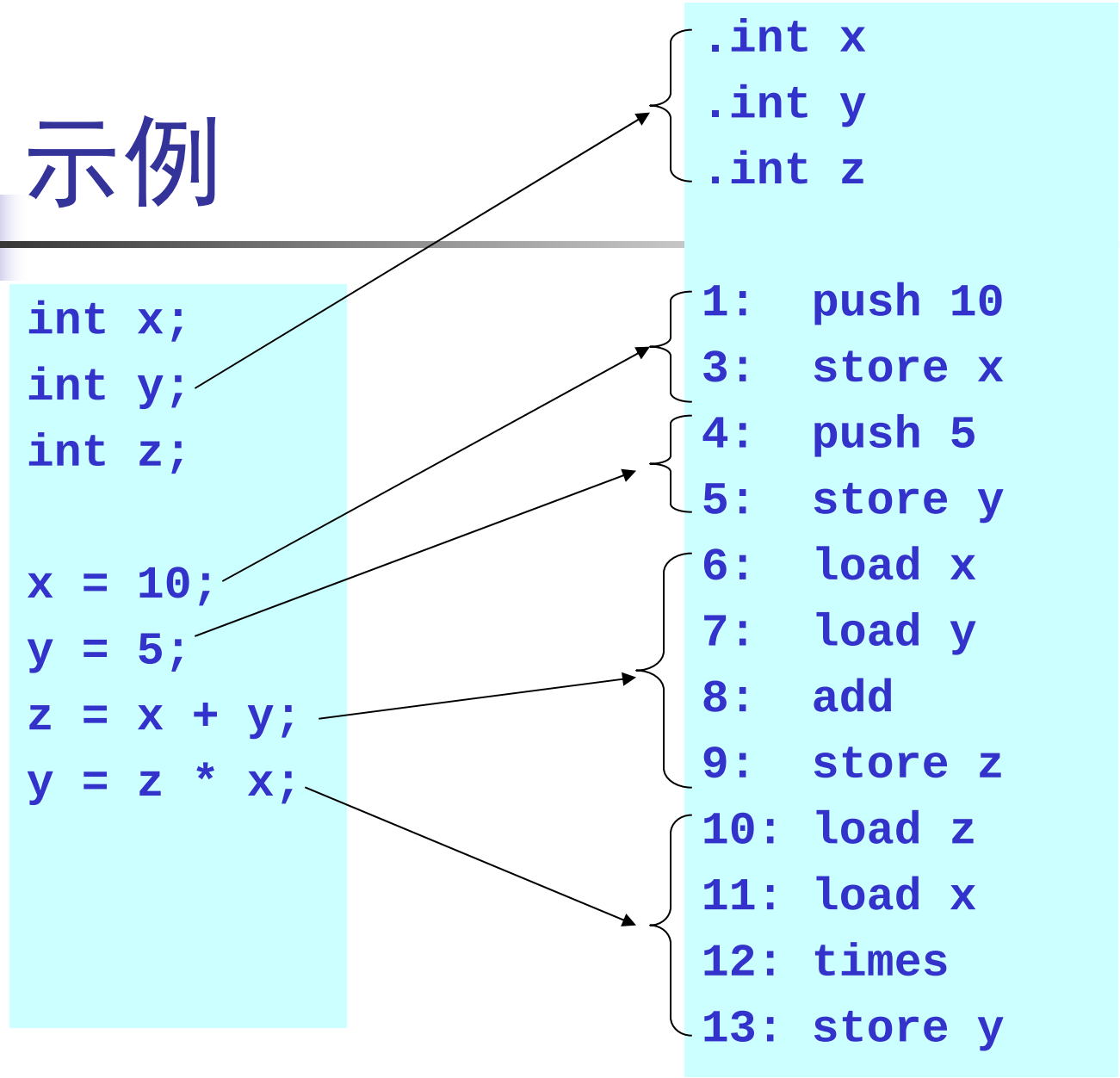




变量的内存分配伪指令

- Stack机器只支持一种数据类型int，并且给变量x分配内存的伪指令是：
  - .int x
- Stack机器在装载一个程序时，就会读取伪指令，并且给相关变量分配内存空间

示例


int x;
int y;
int z;
x = 10;
y = 5;
z = x + y;
y = z * x;
.int x
.int y
.int z
1: push 10
3: store x
4: push 5
5: store y
6: load x
7: load y
8: add
9: store z
10: load z
11: load x
12: times
13: store y