02247 Compiler Construction
What is a compiler?

Andrea Vandin
(*Slides based on: Swami Iyer, UMass Boston, http://www.swamiiyer.net/cs451/slides.html*)

Compilers are organized in several phases/modules

Figure: Compiler as a black-box

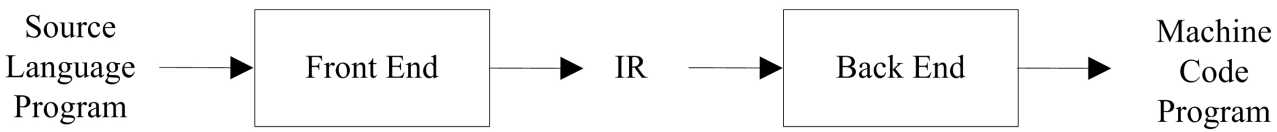


Figure: Compiler components: Front and Back End

Front End Phases

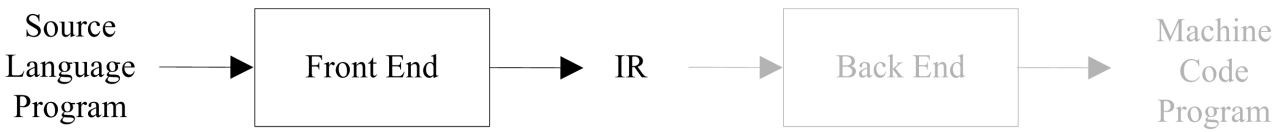


Figure: Compiler components: Front End

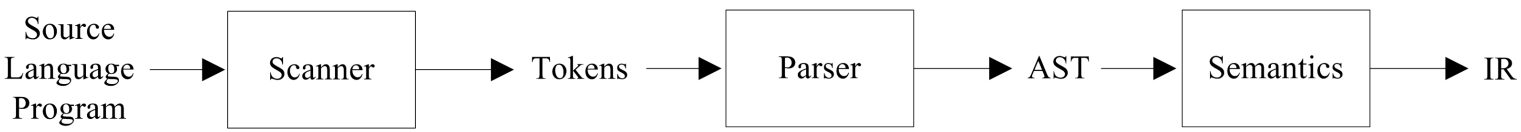


Figure: Front End phases

Back End Phases

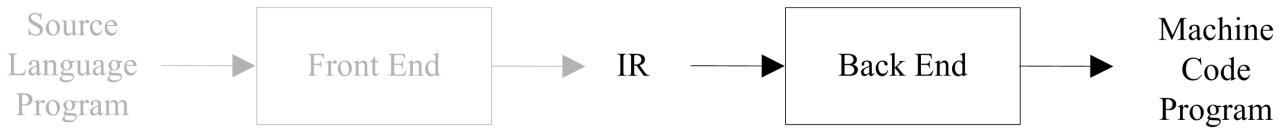


Figure: Compiler components: Back End

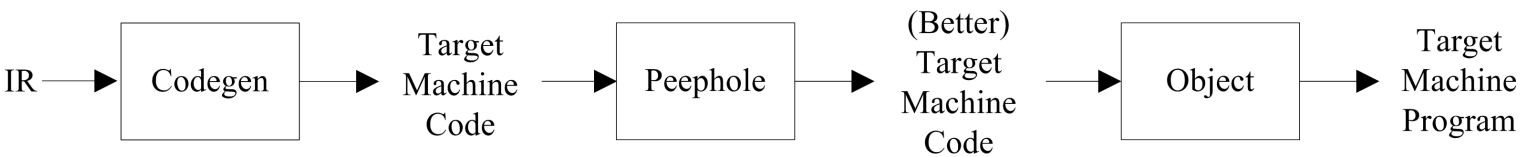


Figure: Back End phases

The Moddle End: Optimizer

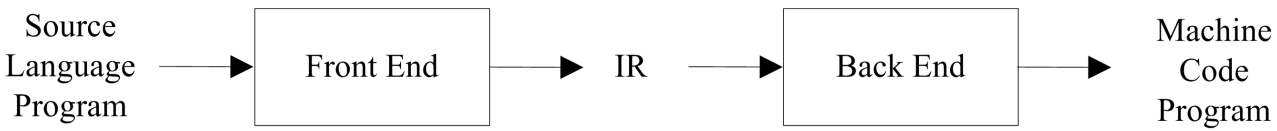


Figure: Compiler components: Front and Back End

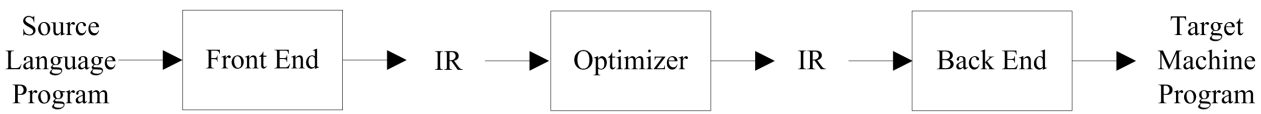


Figure: Compiler components: Front, Middle, and Back End

Front End phases

Example Input

```
float exp(float b, int n){
  // compute exp(b,n) naively
  // assuming n>=0
  float result = 1;
  while (n!=0){
    result = result * b;
    n = n-1;
  }
  return result;
}
```

Front End phases

Scanner (Lexical Analysis)

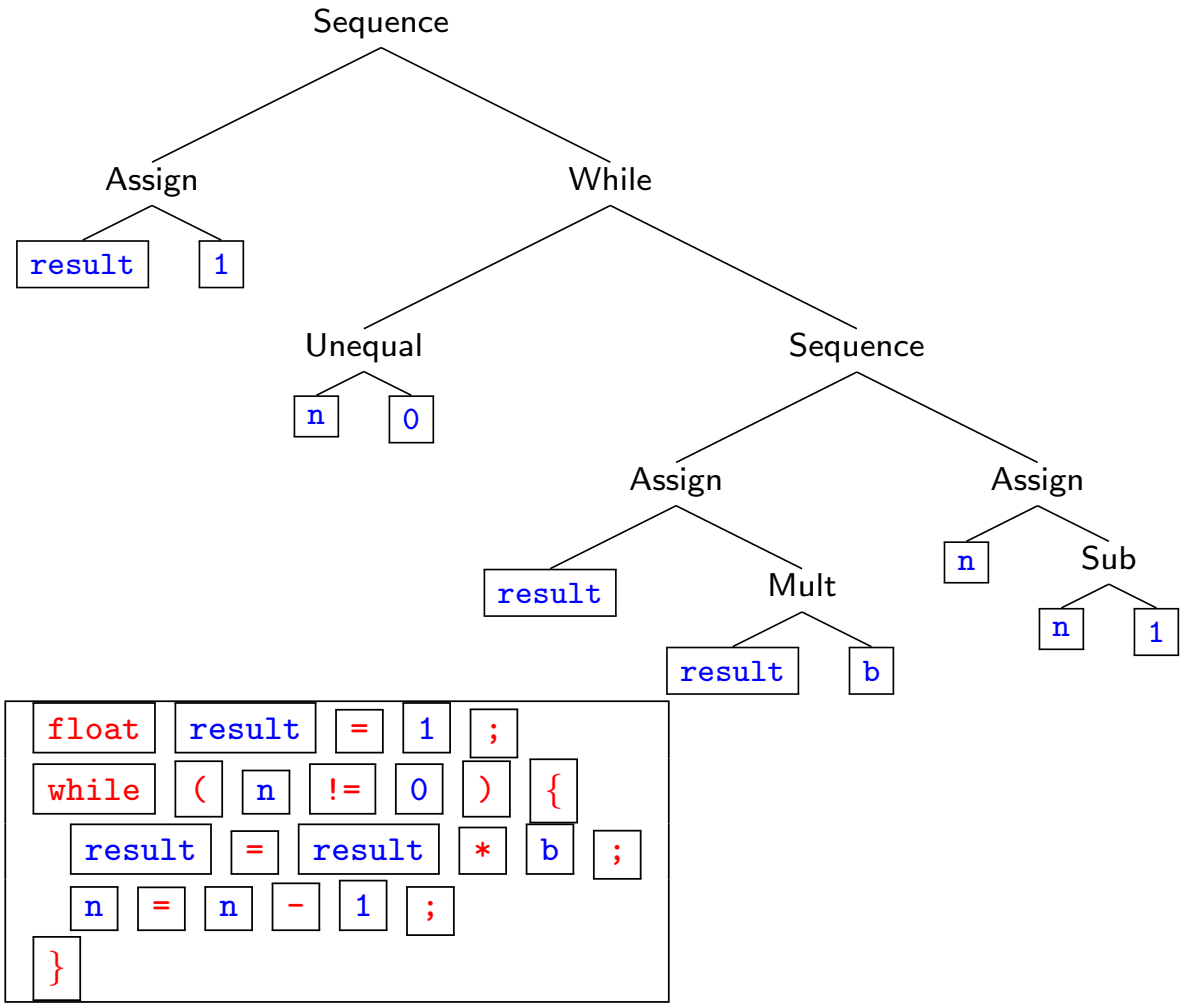
```
float exp ( float b , int n ) {
  // compute exp(b,n) naively
  // assuming n>=0
  float result = 1 ;
  while ( n != 0 ) {
    result = result * b ;
    n = n - 1 ;
  }
  return result ;
}
```

The scanner recognizes

- `Keywords` and `Symbols`
- User-defined `identifiers` for variables, functions, ...
- `Constants` like `0`
- Filters comments and whitespaces

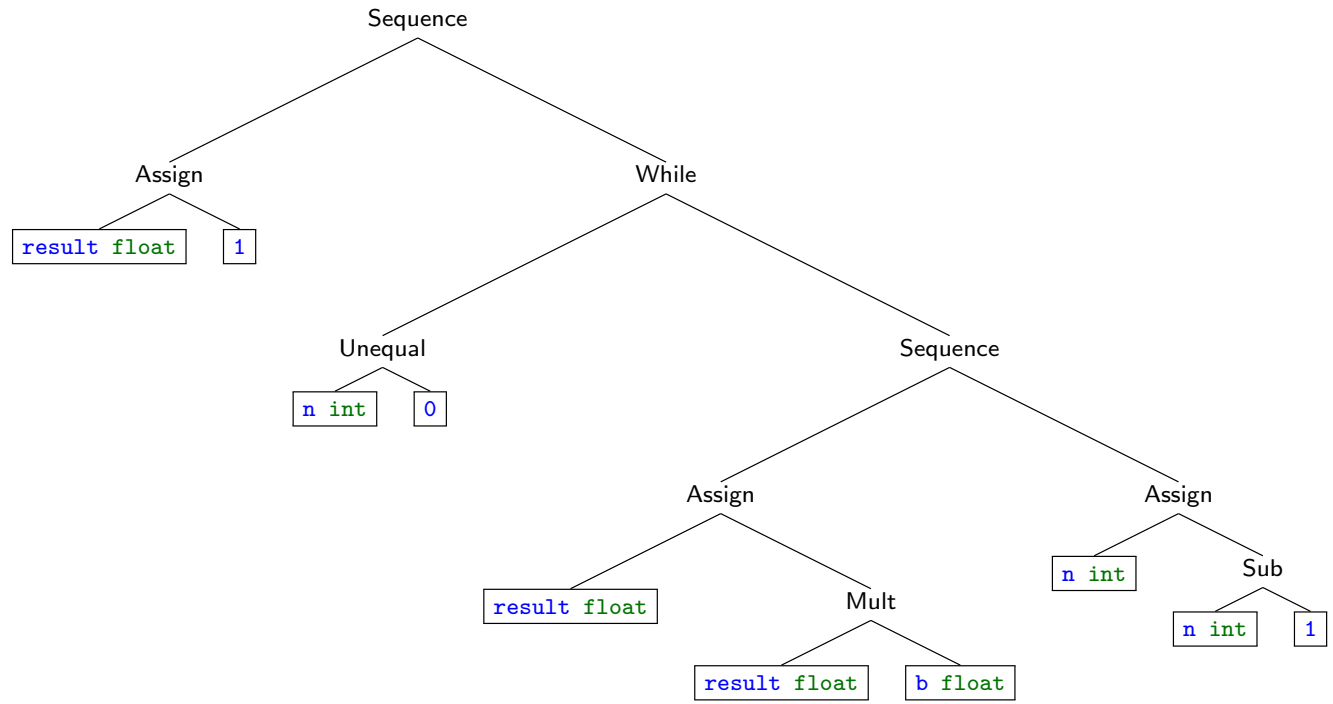
Front End phases

Parser (Syntax Analysis)



Front End phases

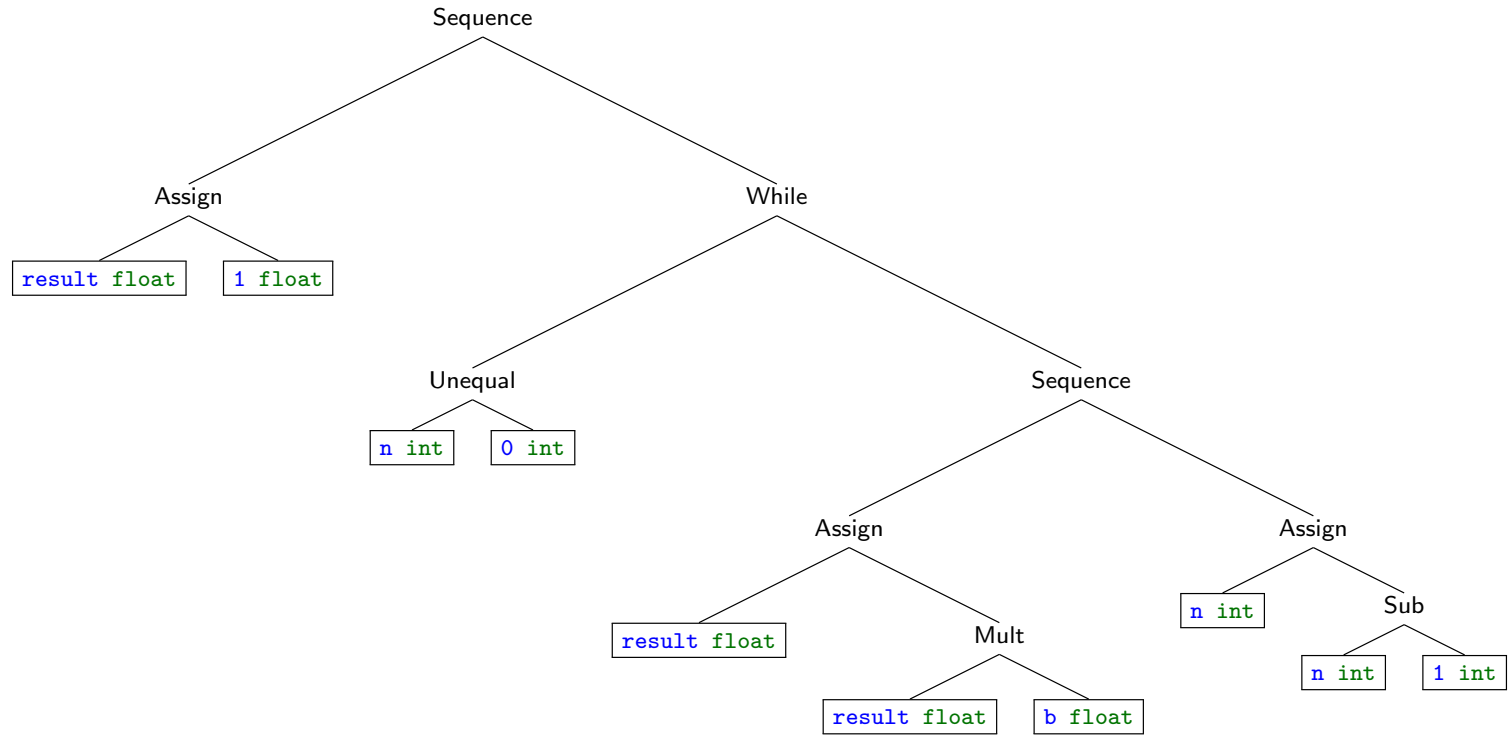
Type Checking (Semantic Analysis)




```
float result = 1 ;
while ( n != 0 ) {
  result = result * b ;
  n = n - 1 ;
}
```

Front End phases

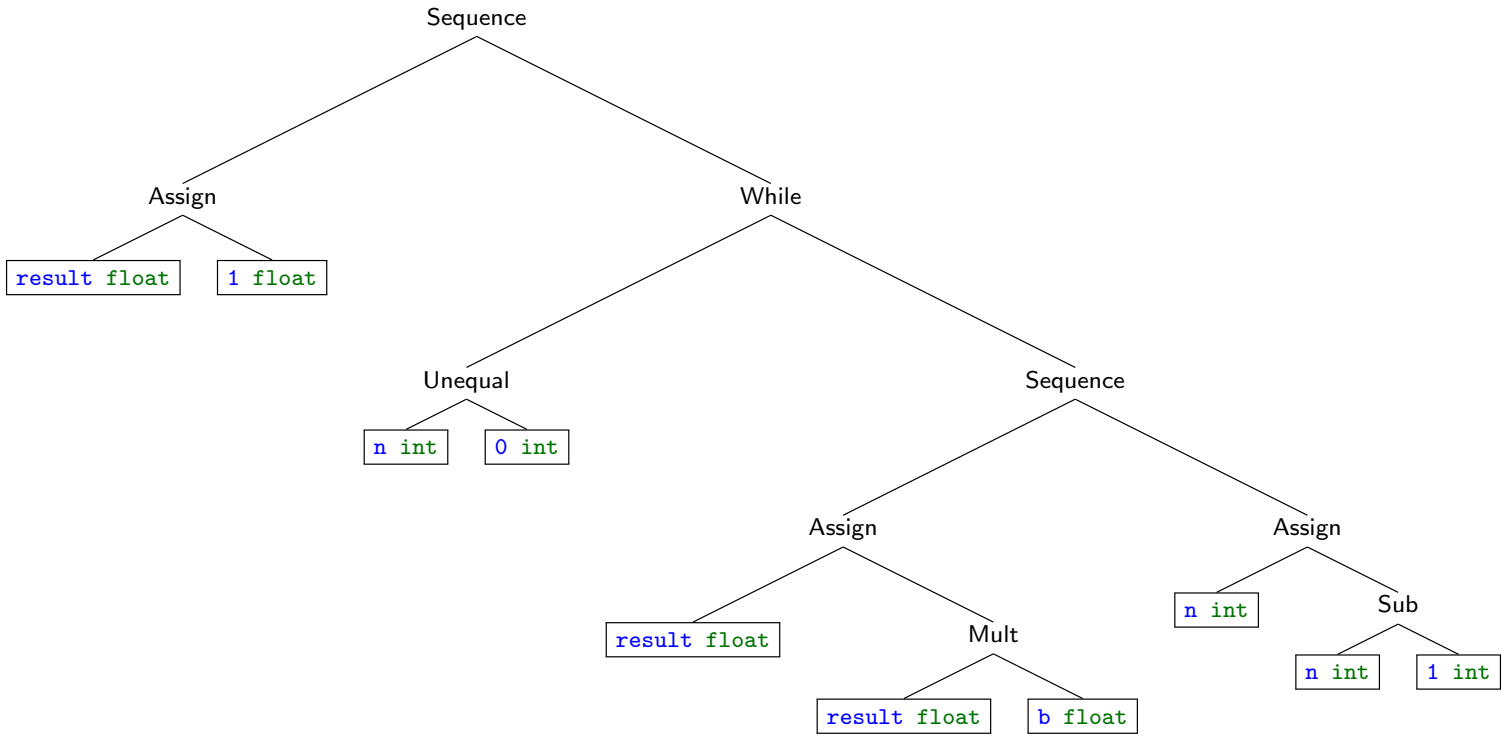
Type Checking (Semantic Analysis)




```
float result = 1 ;
while ( n != 0 ) {
  result = result * b ;
  n = n - 1 ;
}
```

Back End phases

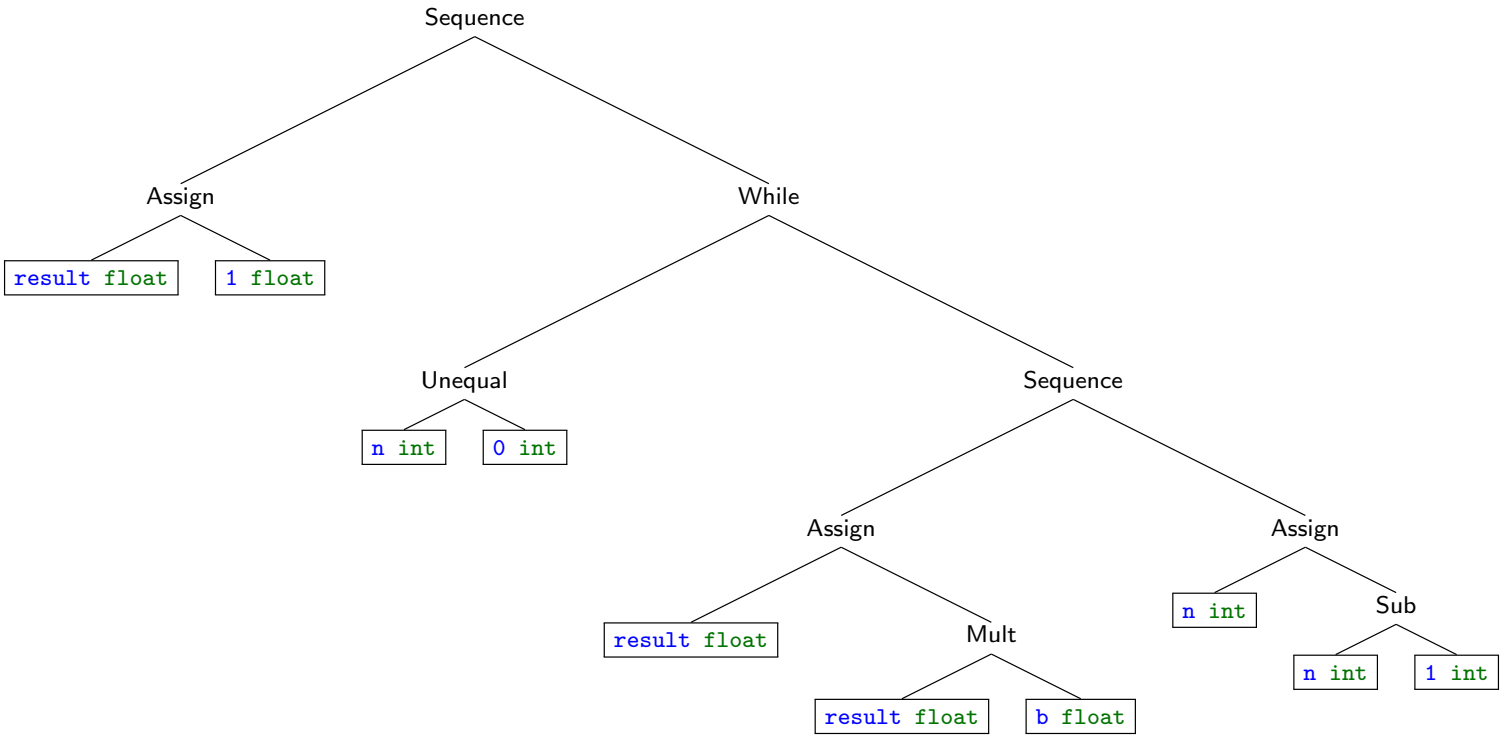
Code Generation




```
         result:=1
loop:    if n compare.int 0 goto end
         result := result mult.float b
         n:= n sub.int 1
         goto loop
end:     ...
```

Back End phases

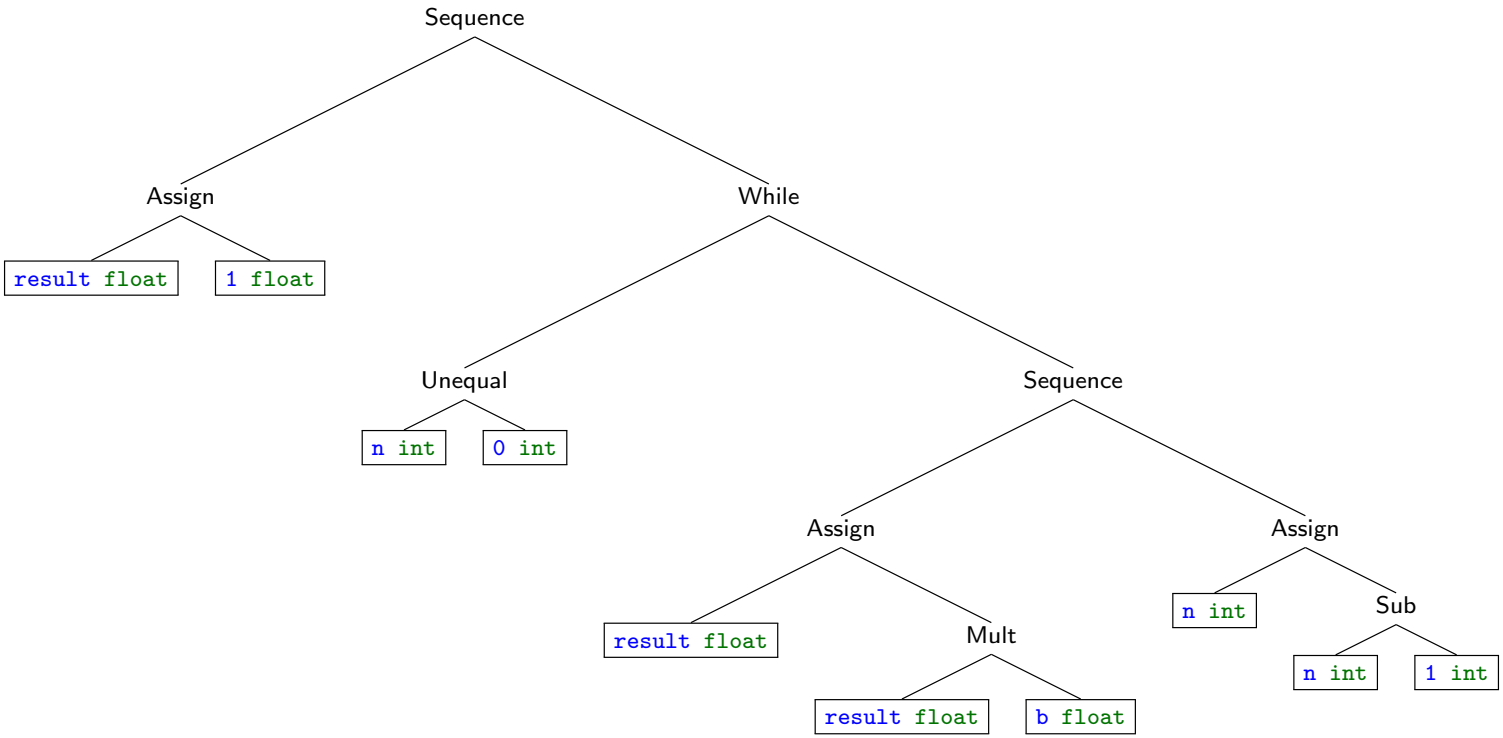
Code Generation – Register Allocation




```
b in r1, n in r2, result in r3, r4 empty
        r3:=1
loop:   if r2 compare.int 0 goto end
        r3 := r3 mult.float r1
        r2 := r2 sub.int 1
        goto loop
end:    ...
```

Back End phases

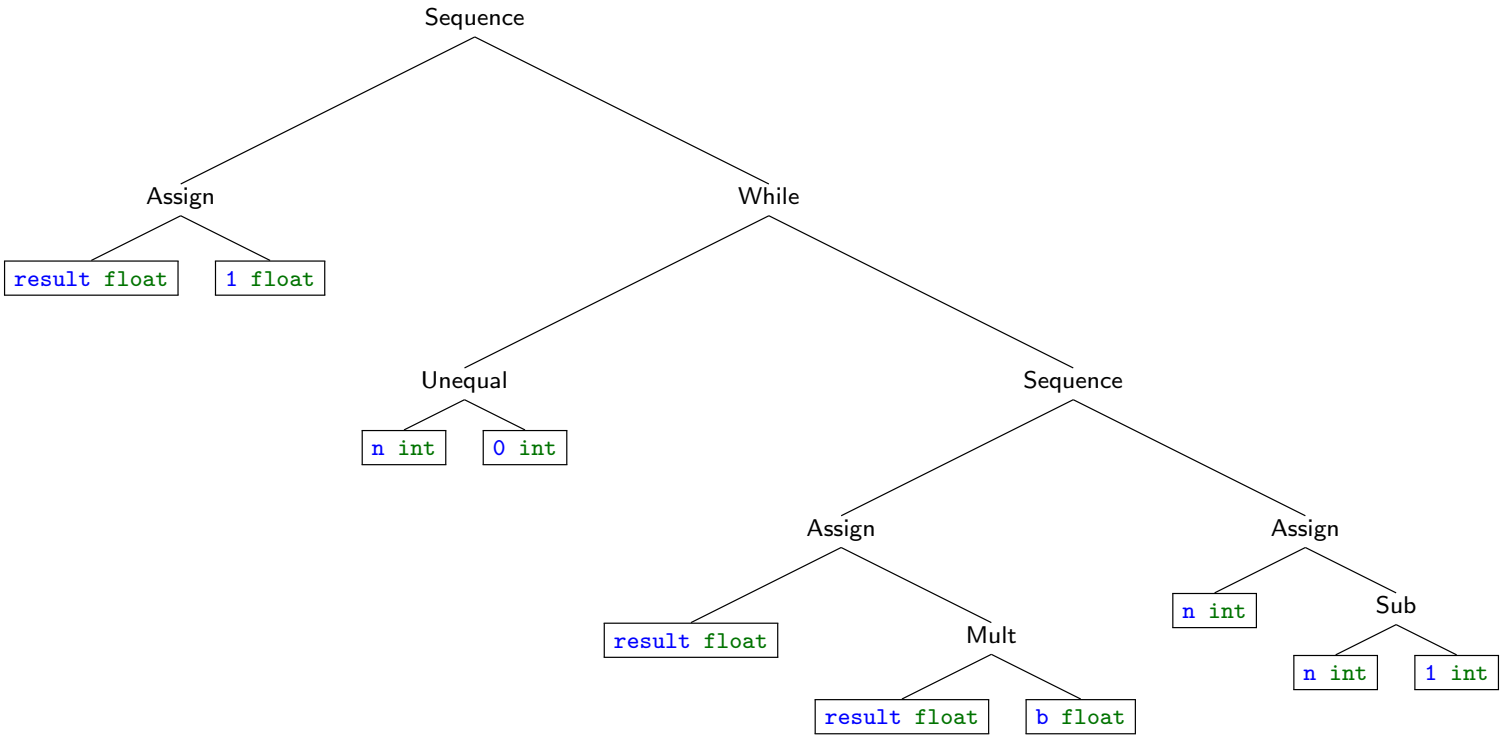
Peephole




```
b in r1, n in r2, result in r3, r4 empty
          r3:=1
loop:     if r2 compare.int 0 goto end
          r3 := r3 mult.float r1
          goto asgn_r2
asgn_r2:  r2 := r2 sub.int 1
          goto loop
end:      ...
```

Back End phases

Peephole




```
b in r1, n in r2, result in r3, r4 empty
          r3:=1
loop:     if r2 compare.int 0 goto end
          r3 := r3 mult.float r1

asgn_r2:  r2 := r2 sub.int 1
          goto loop
end:      ...
```

The Middle End: Optimizer

```
int n = 5;
int i = 0;
while(i <= n-1){
    int j = n * 2;
    System.out.println("i+j = " + (i + j));
    i++;
}
```

```
int n = 5;
int i = 0;
int j = n * 2;
while(i <= n-1){
    System.out.println("i+j = " + (i + j));
    i++;
}
```

```
int n = 5;
int i = 0;
int j = 10;
while(i <= 4){
    System.out.println("i+j = " + (i + j));
    i++;
}
```

The Middle End: Optimizer

```
int n = 5;
int i = 0;
while(i <= n-1){
    int j = n * 2;
    System.out.println("i+j = " + (i + j));
    i++;
}
```

```
int n = 5;
int i = 0;
int j = n * 2;
while(i <= n-1){
    System.out.println("i+j = " + (i + j));
    i++;
}
```

```
int n = 5;
int i = 0;
int j = 10;
while(i <= 4){
    System.out.println("i+j = " + (i + j));
    i++;
}
```

The Middle End: Optimizer

```
int n = 5;
int i = 0;
while(i <= n-1){
    int j = n * 2;
    System.out.println("i+j = " + (i + j));
    i++;
}
```

```
int n = 5;
int i = 0;
int j = n * 2;
while(i <= n-1){
    System.out.println("i+j = " + (i + j));
    i++;
}
```

```
int n = 5;
int i = 0;
int j = 10;
while(i <= 4){
    System.out.println("i+j = " + (i + j));
    i++;
}
```