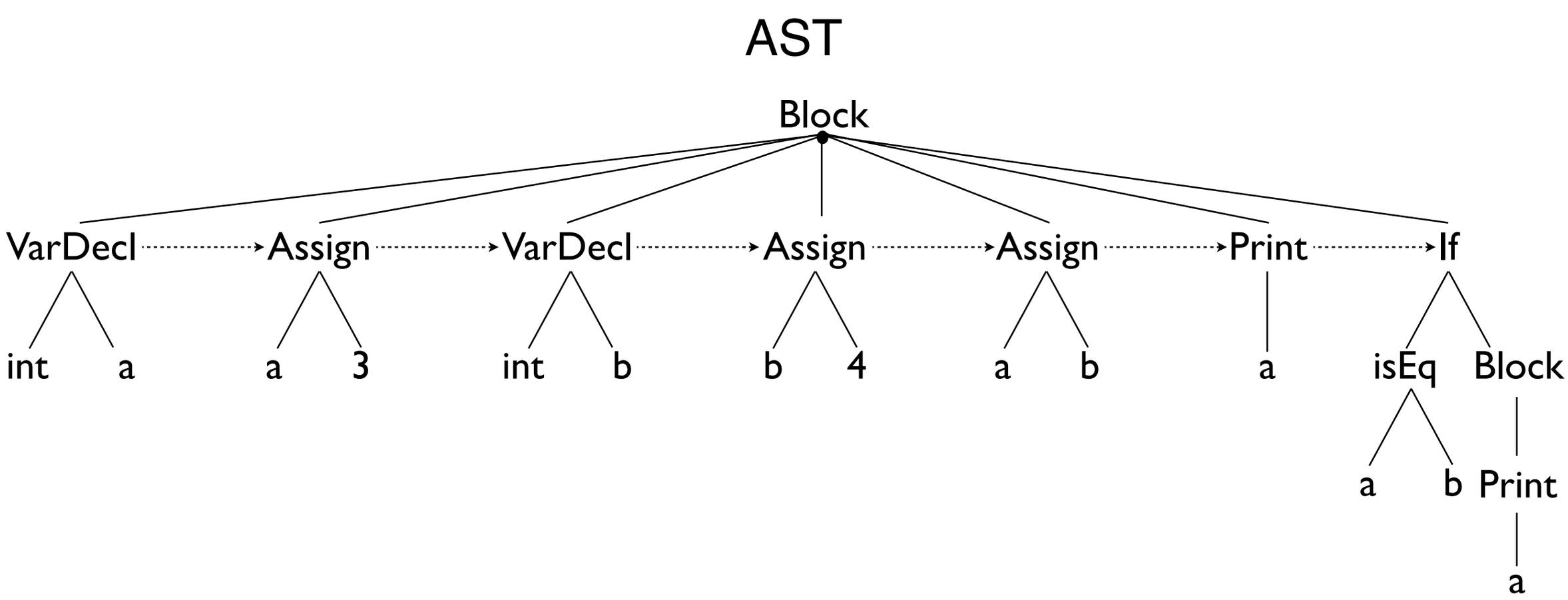
Source Code

```
{
    int a
    a = 3
    int b
    b = 4
    a = b
    print(a)
    if (a == b) {
       print(a)
    }
}
```

AST



Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
}
```

Our compiler generates code to initialize integers to 0.

Load the accumulator with 0.

Store the accumulator in location Temp0, denoted as **T0XX**. We'll fill this in later, once we have calculated the beginning address of the static area.

Make entry in Static table. →

Execution Environment

0	A9	00	8D	T0	XX			
8								
10								
18								
20								
28								
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
}
```

Load the accumulator with 3.

Store the accumulator in location Temp0, denoted as T0XX. We'll fill this in later, once we have calculated the beginning address of the static area.

Opportunity for optimization here.

Execution Environment

0	A9	00	8D	T0	XX	**A9**	**03**	**8D**
8	**T0**	**XX**						
10								
18								
20								
28								
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
}
```

Our compiler generates code to initialize integers to 0.

Load the accumulator with 0.

Store the accumulator in location Temp1, denoted as **T1XX**. We'll fill this in later, once we have calculated the beginning address of the static area.

Make entry in Static table. →

Execution Environment

0	A9	00	8D	T0	XX	A9	03	8D
8	T0	XX	**A9**	**00**	**8D**	**T1**	**XX**	
10								
18								
20								
28								
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0
T1XX	b	+=len(T0XX)

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
}
```

Load the accumulator with 4.

Store the accumulator in location Temp1, denoted as T1XX. We'll fill this in later, once we have calculated the beginning address of the static area.

Opportunity for optimization here.

Execution Environment

0	A9	00	8D	T0	XX	A9	03	8D
8	T0	XX	A9	00	8D	T1	XX	**A9**
10	**04**	**8D**	**T1**	**XX**				
18								
20								
28								
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0
T1XX	b	+=len(T0XX)

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
}
```

Load the accumulator with the contents of b, which we look up to find at temp address T1XX.

Store the accumulator in the address for a, which we look up and find is temp address T0XX.

Execution Environment

0	A9	00	8D	T0	XX	A9	03	8D
8	T0	XX	A9	00	8D	T1	XX	A9
10	04	8D	T1	XX	**AD**	**T1**	**XX**	**8D**
18	**T0**	**XX**						
20								
28								
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0
T1XX	b	+=len(T0XX)

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
}
```

Load the Y register with the contents of a, which we look up to find at temp address T0XX.

Load the X register with 1.

System Call.

Execution Environment

0	A9	00	8D	T0	XX	A9	03	8D
8	T0	XX	A9	00	8D	T1	XX	A9
10	04	8D	T1	XX	AD	T1	XX	8D
18	T0	XX	**AC**	**T0**	**XX**	**A2**	**01**	**FF**
20								
28								
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0
T1XX	b	+=len(T0XX)

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
}
```

Load the X register with the contents of a.

Compare the X register to the contents of b.

Branch on NOT EQUAL, jumping ahead some number of bytes (**temp J0**) to AFTER the generated code for the “if true” statement list.

Print a (Same op codes as prior statement.)

Execution Environment

0	A9	00	8D	T0	XX	A9	03	8D
8	T0	XX	A9	00	8D	T1	XX	A9
10	04	8D	T1	XX	AD	T1	XX	8D
18	T0	XX	AC	T0	XX	A2	01	FF
20	**AE**	**T0**	**XX**	**EC**	**T1**	**XX**	**D0**	**J0**
28	**AC**	**T0**	**XX**	**A2**	**01**	**FF**		
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0
T1XX	b	+=len(T0XX)

Temp	Distance
J0	?

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
   print(a)
} (break)
```

This is the end of the program.

Break. (Just to be safe.)

Execution Environment

0	A9	00	8D	T0	XX	A9	03	8D
8	T0	XX	A9	00	8D	T1	XX	A9
10	04	8D	T1	XX	AD	T1	XX	8D
18	T0	XX	AC	T0	XX	A2	01	FF
20	AE	T0	XX	EC	T1	XX	D0	J0
28	AC	T0	XX	A2	01	FF	**00**	
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0
T1XX	b	+=len(T0XX)

Temp	Distance
J0	?

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
} (break)
```

Calculate the distance to jump for temporary jump value J0.

(7 bytes)

Backpatch the code with this value.

Execution Environment

0	A9	00	8D	T0	XX	A9	03	8D
8	T0	XX	A9	00	8D	T1	XX	A9
10	04	8D	T1	XX	AD	T1	XX	8D
18	T0	XX	AC	T0	XX	A2	01	FF
20	AE	T0	XX	EC	T1	XX	D0	**07**
28	AC	T0	XX	A2	01	FF	00	
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	+0
T1XX	b	+=len(T0XX)

Temp	Distance
J0	**7**

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
} (break)
```

If were are not concerned with byte alignment, we can begin the Static variable area at location 2F.

Find all T0XX and replace with 2F00 (little endian)

Execution Environment

0	A9	00	8D	**2F**	**00**	A9	03	8D
8	**2F**	**00**	A9	00	8D	T1	XX	A9
10	04	8D	T1	XX	AD	T1	XX	8D
18	**2F**	**00**	AC	**2F**	**00**	A2	01	FF
20	AE	**2F**	**00**	EC	T1	XX	D0	07
28	AC	**2F**	**00**	A2	01	FF	00	used a
30								
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	**2F 00**
T1XX	b	+=len(T0XX)

Temp	Distance
J0	7

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
} (break)
```

Integers are one byte, so the address of T1XX is the base address for Static storage (002F) + the length of T0XX (1), or 0030.

Find all T1XX and replace with 3000 (little endian)

Execution Environment

0	A9	00	8D	2F	00	A9	03	8D
8	2F	00	A9	00	8D	**30**	**00**	A9
10	04	8D	**30**	**00**	AD	**30**	**00**	8D
18	2F	00	AC	2F	00	A2	01	FF
20	AE	2F	00	EC	**30**	**00**	D0	07
28	AC	2F	00	A2	01	FF	00	used a
30	**used b**							
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	2F 00
T1XX	b	**30 00**

Temp	Distance
J0	7

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
} (break)
```

We can begin dynamic allocation on the HEAP, at location 0031. Or we can start dynamic allocation at 005F in this example) and work our way back to 0031.

Execution Environment

0	A9	00	8D	2F	00	A9	03	8D
8	2F	00	A9	00	8D	30	00	A9
10	04	8D	30	00	AD	30	00	8D
18	2F	00	AC	2F	00	A2	01	FF
20	AE	2F	00	EC	30	00	D0	07
28	AC	2F	00	A2	01	FF	00	used a
30	used b							
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	2F 00
T1XX	b	30 00

Temp	Distance
J0	7

Source Code

```
int a
a = 3
int b
b = 4
a = b
print(a)
if (a == b) {
  print(a)
} (break)
```

Machine Code

Let’s try the program:

```
A9 00 8D 2F 00 A9 03 8D
2F 00 A9 00 8D 30 00 A9
04 8D 30 00 AD 30 00 8D
2F 00 AC 2F 00 A2 01 FF
AE 2F 00 EC 30 00 D0 07
AC 2F 00 A2 01 FF 00 00
```

Execution Environment

0	A9	00	8D	2F	00	A9	03	8D
8	2F	00	A9	00	8D	30	00	A9
10	04	8D	30	00	AD	30	00	8D
18	2F	00	AC	2F	00	A2	01	FF
20	AE	2F	00	EC	30	00	D0	07
28	AC	2F	00	A2	01	FF	00	used a
30	used b							
38								
40								
48								
50								
58								

Temp	Var	Address
T0XX	a	2F 00
T1XX	b	30 00

Temp	Distance
J0	7