Principles of Programming Languages

CS 314

Recitation 6

RUTGERS

Topics Today

Lexical/Dynamic Scoping

Runtime Stack

Accessing Non-local Variables using RISC

Renaming variables to (level, offset) pairs for lexical scoping

Lexical/Dynamic Scoping

Runtime Stack

Accessing Non-local Variables using RISC

Renaming variables to (level, offset) pairs for lexical scoping

Dynamic Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

If A is dynamically scoped, what will be the output of this “print A” statement?

https://stackoverflow.com/questions/22394089/

Dynamic Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

Before running proc2,

A has a value of 20, in main().

Dynamic Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

Call procedure proc2(). A is newly declared in proc2(). proc2().A has a value of 1.

Dynamic Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

Call procedure proc1() now.

Because A is dynamically scoped, proc1.A keeps the value of proc2.A and adds 2. proc1.A becomes 3.

Dynamic Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

But since A is dynamically scoped, main.A did not change. So, inside main(), “print A” will still print 20.

Lexical Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

If A is lexically scoped, what will be the output of this “print A” statement?

Lexical Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

Before running proc2(),

A has a value of 20, in main().

Lexical Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

Call procedure proc2(). proc2().A has a value of 1.

Lexical Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

Call procedure proc1() now.

Because A is lexically scoped, proc1.A takes the value of main.A. So, main.A becomes 22.

Lexical Scoping – Example

```
program main():
    int A = 0;
    procedure proc1():
        A = A + 2;
    end proc1;
    procedure proc2():
        int A = 1;
        proc1();
    end proc2;
    A = A + 20;
    proc2();
    print A;
end main;
```

Since A is lexically scoped, inside main(), "print A" will print 22.

Lexical/Dynamic Scoping

Runtime Stack

Accessing Non-local Variables using RISC

Renaming variables to (level, offset) pairs for lexical scoping

Runtime Stack – Example

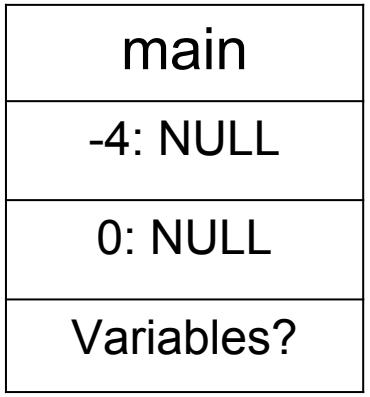
```
program main():
    int b = 0;
    procedure proc1():
        int c = 0;
        procedure proc2():
            b = b + 2;
            print b;
        end proc2;
        print c;
    end proc1;
    b = b + 20;
    proc1();
end main;
```

Suppose b and c are lexically scoped. What is the runtime stack at the time of starting proc1()?

Runtime Stack – Example

```
program main():
    int b = 0;
    procedure proc1():
        int c = 0;
        procedure proc2():
            b = b + 2;
            print b;
        end proc2;
        print c;
    end proc1;
    b = b + 20;
    proc1();
end main;
```

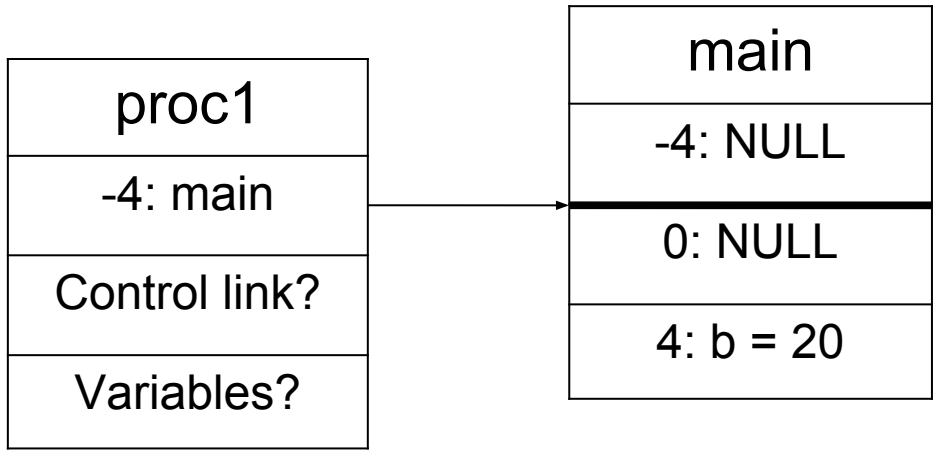
First, we need a stack frame for main(). Since main() is the entry point, both links are NULL.



Runtime Stack – Example

```
program main():
    int b = 0;
    procedure proc1():
        int c = 0;
        procedure proc2():
            b = b + 2;
            print b;
        end proc2;
        print c;
    end proc1;
    b = b + 20;
    proc1();
end main;
```

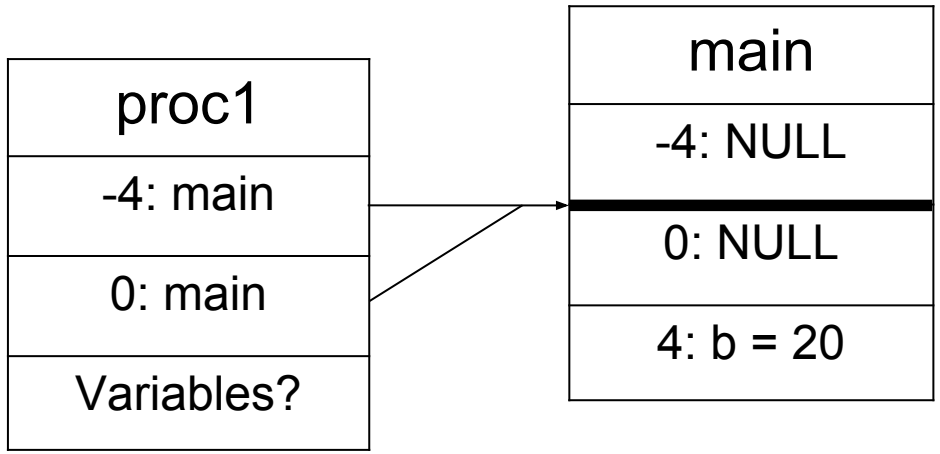
At the time of calling proc1(), b has a value of 20 in main(), since b is lexically scoped. proc1()'s access link points to main().



Runtime Stack – Example

```
program main():
    int b = 0;
    procedure proc1():
        int c = 0;
        procedure proc2():
            b = b + 2;
            print b;
        end proc2;
        print c;
    end proc1;
    b = b + 20;
    proc1();
end main;
```

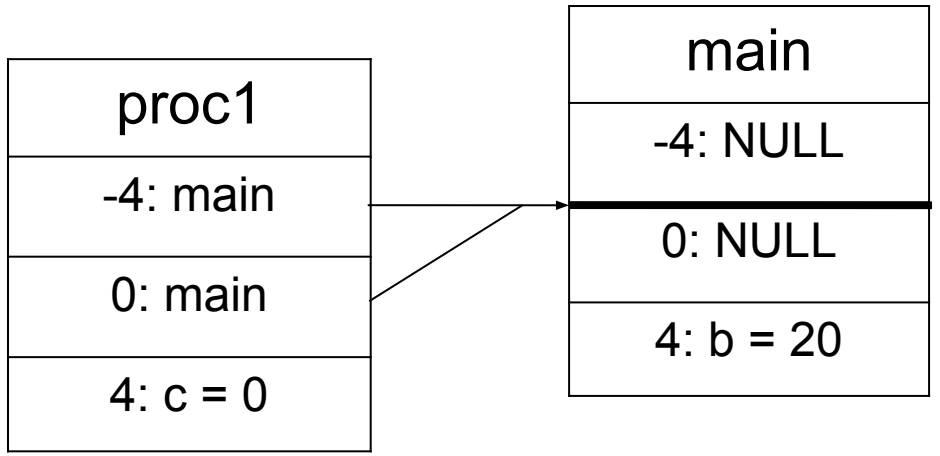
Since main() is the caller of proc1(), main() is also the control link of proc1().



Runtime Stack – Example

```
program main():
    int b = 0;
    procedure proc1():
        int c = 0;
        procedure proc2():
            b = b + 2;
            print b;
        end proc2;
        print c;
    end proc1;
    b = b + 20;
    proc1();
end main;
```

proc1()'s local variable c has value 0.



Lexical/Dynamic Scoping

Runtime Stack

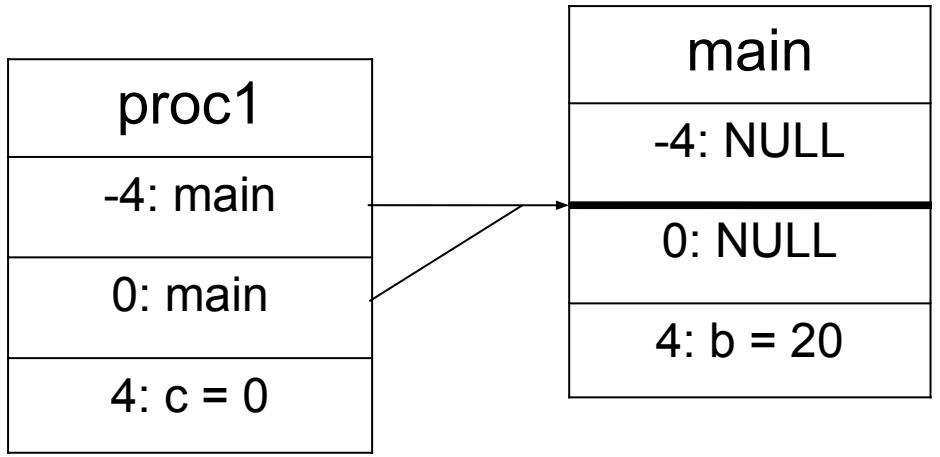
Accessing Non-local Variables using RISC

Renaming variables to (level, offset) pairs for lexical scoping

Accessing Non-local Variables – Example

```
program main():
    int b = 0;
    procedure proc1():
        int c = 0;
        b = b + 2;
        procedure proc2():
            b = b + 2;
            print b;
        end proc2;
        print c;
    end proc1;
    b = b + 20;
    proc1();
end main;
```

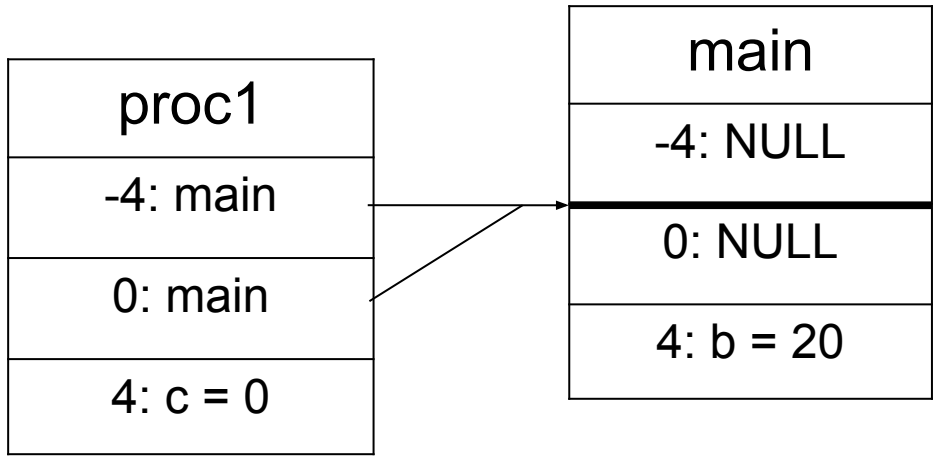
What are the RISC commands for how proc1() accesses the lexical parent of b (i.e. the b in main())? (Assume R0 holds the frame pointer.)



Accessing Non-local Variables – Example

```
LOADI R2, #-4
ADD R3, R0, R2
LOAD R4, R3
```

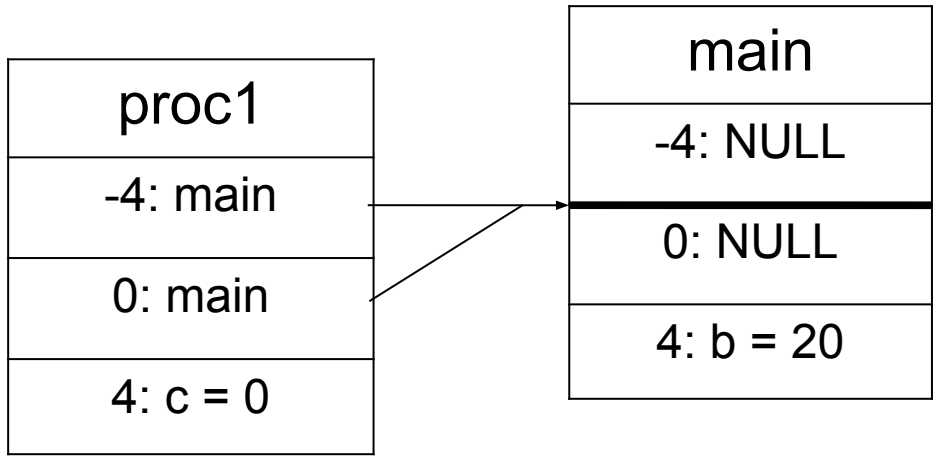
This sequence of instructions allows one to get the address of main's frame. R4 is the address of main's frame.



Accessing Non-local Variables – Example

```
LOADI R2, #-4
ADD R3, R0, R2
LOAD R4, R3
LOADI R5, #4
```

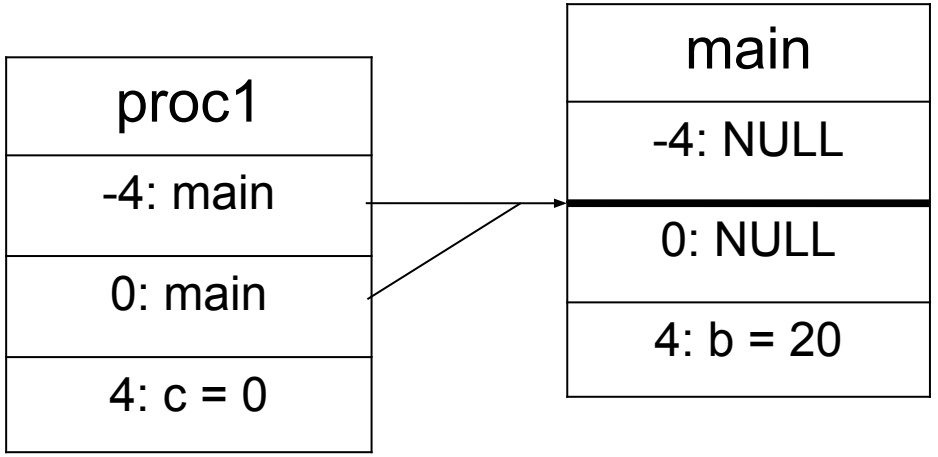
The address of b in main's frame is #4.



Accessing Non-local Variables – Example

```
LOADI R2, #-4
ADD R3, R0, R2
LOAD R4, R3
LOADI R5, #4
ADD R6, R4, R5
```

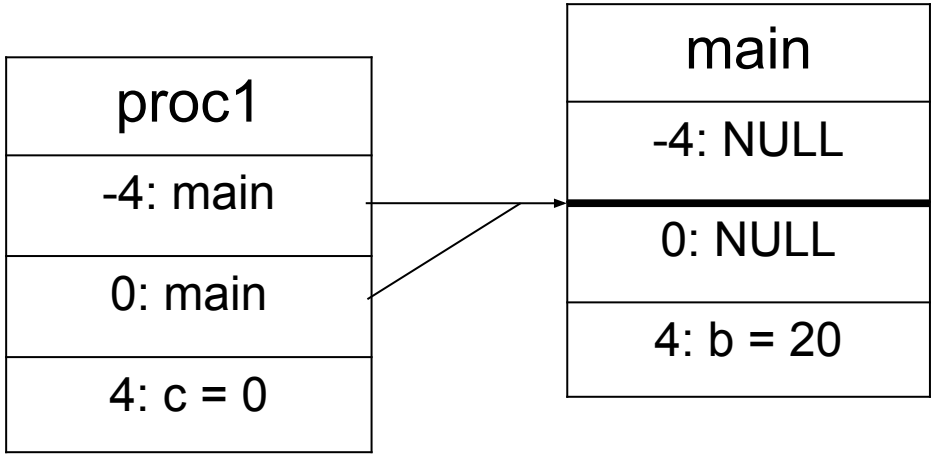
R6 now holds the address of b in main's frame.



Accessing Non-local Variables – Example

```
LOADI R2, #-4
ADD R3, R0, R2
LOAD R4, R3
LOADI R5, #4
ADD R6, R4, R5
LOAD R7, R6
```

R7 is the value of b in main's frame. This is how proc1() gets the value of b in main().



Lexical/Dynamic Scoping

Runtime Stack

Accessing Non-local Variables using RISC

Renaming variables to (level, offset) pairs for lexical scoping

(Level, Offset) Pairs – Example

```
program main():
    int A = 0;
    procedure proc1():
        int b = 2;
        procedure proc2():
            A = A + b;
        end proc2;
    end proc1;
    A = A + 20;
    proc1();
end main;
```

Consider the following program. Suppose this program uses lexical scoping, i.e. all variables are lexically scoped. What are the (level, offset) pairs for each variable?

(Level, Offset) Pairs – Example

```
program main():
    int (1, 1) = 0;
    procedure proc1():
        int b = 2;
        procedure proc2():
            A = A + b;
        end proc2;
    end proc1;
    A = A + 20;
    proc1();
end main;
```

A is the first variable declared in main(). Since main() is the first procedure ran, A has (level, offset) pair (1, 1).

(Level, Offset) Pairs – Example

```
program main():
    int (1, 1) = 0;
    procedure proc1():
        int b = 2;
        procedure proc2():
            (1, 1) = (1, 1) + b;
        end proc2;
    end proc1;
    (1, 1) = (1, 1) + 20;
    proc1();
end main;
```

Since this program uses lexical scoping, the (level, offset) pair for A is kept throughout the program, even inside the procedures.

(Level, Offset) Pairs – Example

```
program main():
    int (1, 1) = 0;
    procedure proc1():
        int (2, 1) = 2;
        procedure proc2():
            (1, 1) = (1, 1) + (2, 1);
        end proc2;
    end proc1;
    (1, 1) = (1, 1) + 20;
    proc1();
end main;
```

b is a variable declared in proc1(), so it's in the second level, instead of the first. Thus, its (level, offset) pair is (2, 1).

Note that the function call proc1() could've had a (level, offset) pair of (1, 2).