CS314 Fall 2018

Assignment 5 Solutions

1 Lexical/Dynamic Scoping

Recall that this is the procedure defined:

```
procedure main():
    int var = 10;
    procedure set_var(int val):
        var = val;
    end set_var
    procedure proc1():
        set_var(1);
    end proc1
    procedure proc2():
        int var = 2;
        set_var(4);
        print var;
    end proc2
    print var;
    set_var(41);
    proc1();
    print var;
    proc2();
end main
```

(a) With lexical scoping, the procedure **main()** will print 10, 1, 2.

(b) With dynamic scoping, the procedure **main()** will print 10, 1, 4, since the call "set_var(4);" in **proc2()** change the variable **var** in the scope of **proc2()**, not **main()**.

2 Lexical Scoping

Recall that this problem uses the following procedure, and that this procedure uses static (lexical) scoping for variables.

```
procedure main():
    int a;
    procedure proc1(int i):
        int b;
        b = a + 1;
        procedure recursion(int k):
            print b;
            b = b - 1;
            if (b > 1):
                recursion(k * b);
            else:
                a = k;
        end recursion
        b = b + i;
        recursion(1);
    end proc1
    a = 1;
    proc1(4);
    print a;
end main
```

(a) The output of the procedure **main()** is: 6, 5, 4, 3, 2, 120.

(b)
```
procedure main():
    int (1, 1);
    procedure (1, 2)(int (2, 1)):
        int (2, 2);
        (2, 2) = (1, 1) + 1;
        procedure (2, 3)(int (3, 1)):
            print (2, 2);
            (2, 2) = (2, 2) - 1;
            if ((2, 2) > 1):
                (2, 3)((3, 1) * (2, 2));
            else:
                (1, 1) = (3, 1);
        end (2, 3)
        (2, 2) = (2, 2) + (2, 1);
        (2, 3)(1);
    end (1, 2)
    (1, 1) = 1;
    (1, 2)(4);
    print (1, 1);
end main
```

(c) Assume that the current frame pointer is in register R0. In order for **proc1()** to find the variable a, it must take the value of a from **main()**'s runtime stack. Therefore, the RISC commands would look like the following:

```
LOADI R1, #-4;
ADD R2, R0, R1; //main's access pointer
LOAD R3, R2; //main
LOADI R4, #4;
ADD R5, R3, R4; //address of a
LOAD R6, R5
```

(d) The stack frames at the beginning of procedure **proc1** is shown below.

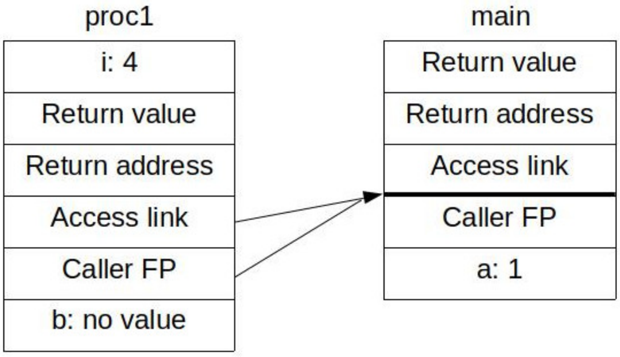

Figure 1: Figure for the frame layout