Sotheanith Sok

CECS 424

12/10/18

Assignment 12 Solution

1. Elaborate on the reasons why even though parameters passed in registers may sometimes need to have locations in the stack

> **Answer**: The reason why there sometime is a need to have location in the stack is due to the memory management. Registers is fast but its space is very limited and as such, any data that larger than the size of a stack or use infrequently will move to a different location and the address to that location will be store in the stack.

2. Consider the following function and call:

```
procedure f(x, y, z)
        x := x + 1
        y := z
        z := z + 1
. . .
i := 1; a[1] := 10; a[2] := 11
f(i, a[i], i)
print(i, a[1], a[2])
```

2.1 Passed by value

```
procedure f(x, y, z)          //x = 1, y = 10, z = 1;
    x := x + 1                //x = 2
    y := z                    //y = 1;
    z := z + 1                //z = 2;
. . .
i := 1; a[1] := 10; a[2] := 11
f(i, a[i], i)
print(i, a[1], a[2])
```

Answer: 1, 10, 11

2.2 Passed by reference

```
procedure f(x, y, z)          //x = i, y = a[1], z = i;
    x := x + 1                //i = 2
    y := z                    //a[1] =2;
    z := z + 1                //i = 3;
. . .
i := 1; a[1] := 10; a[2] := 11
f(i, a[i], i)
print(i, a[1], a[2])
```

Answer: 3, 2, 11

2.3 Passed by name

```
procedure f(x, y, z)          //x = i, y = a[i], z = i;
    x := x + 1                //i = 2
    y := z                    //a[i] = a[2] = 2;
    z := z + 1                //i = 3;
. . .
i := 1; a[1] := 10; a[2] := 11
f(i, a[i], i)
print(i, a[1], a[2])
```

Answer: 3, 10, 2

3. Check out the output of the following program in C in your system and suggest an explanation in terms of activation records in the stack.

```
void foo() {

        int i;

        printf("%d ", i++);

}

int main() {

        int j;

        for (j = 1; j <= 10; j++) {

                foo();

        }

}
```

Output: 2009101242 2009101243 2009101244 2009101245 2009101246 2009101247 2009101248 2009101249 2009101250 2009101251

Answer: The output values of the program are "garbage values." In C, there is no garbage collector and thus, when the activation records is out of bound, values contains within memory location of that activation records will still remain there. In addition, when a variable is declared but not initialize, its value refers to whatever data exists at that memory location. This two factors results in the output that observe above where variable **i** is always referred to the same memory due to the fact that foo()'s activation records is also created at the same memory every call.