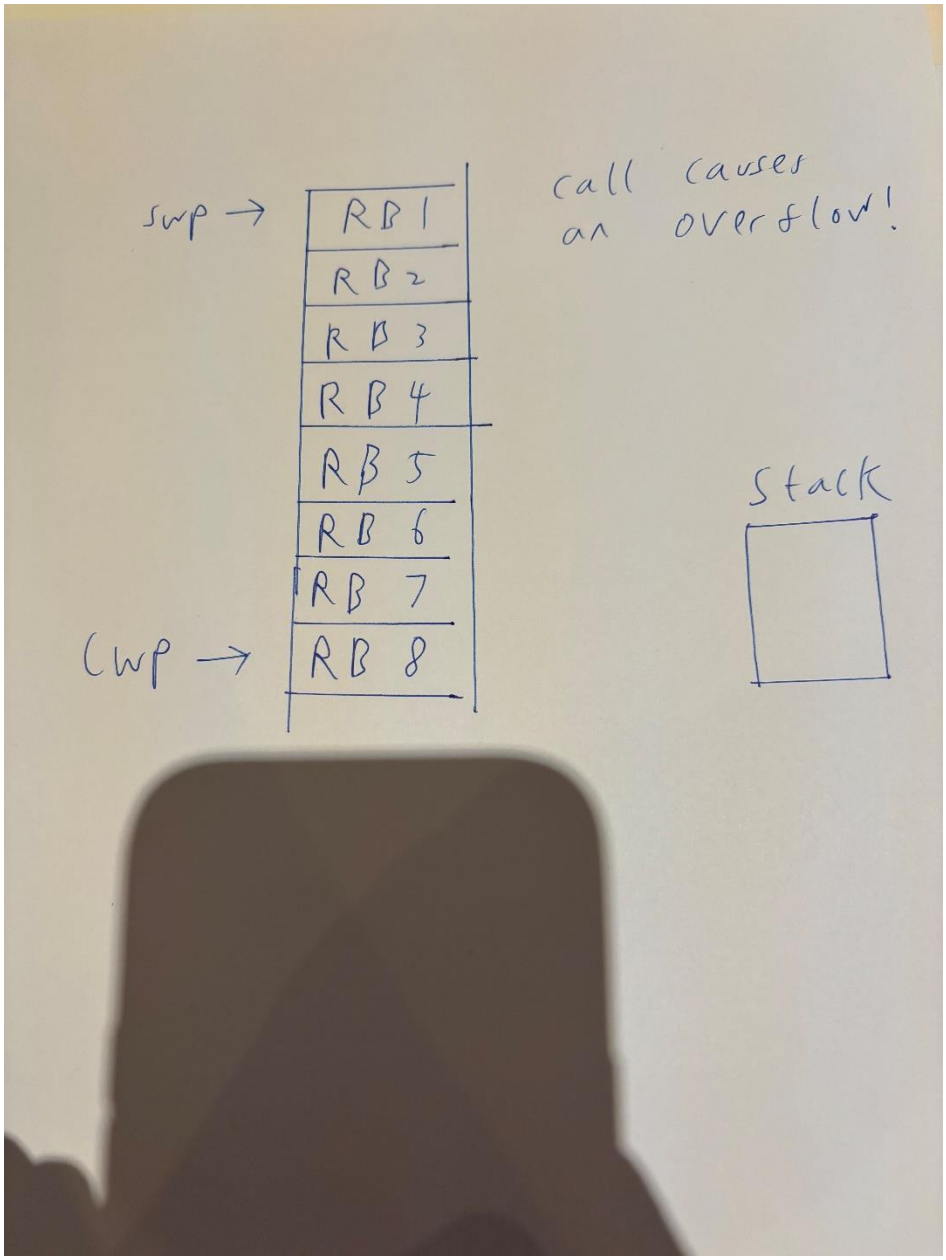
Gerard Moylan. ID: 21364007

Calculating the number of calls to ack3way is simple, I just keep a global counter that gets incremented each time.

Counting the number of overflow occurrences:

Overflow occurs when you call a function and all 8 of the windows have been used. The oldest window gets pushed onto the stack, and then those registers get given to the function that has just been called. Below is an illustration of a situation where a function call would cause an overflow as all 8 register windows are used up.



If a function is called in this situation, the register pointed to by SWP is pushed to the stack, in this case RB1. SWP is then incremented to point to RB2 and CWP is incremented cyclically to point to the top position, which will now be taken by RB 9.

In my code, I counted overflows by keeping a count of windowsUsed. If it was ever equal to the total number of windows (8), windowsUsed would stay the same but numOverflows would be incremented as seen in this code:

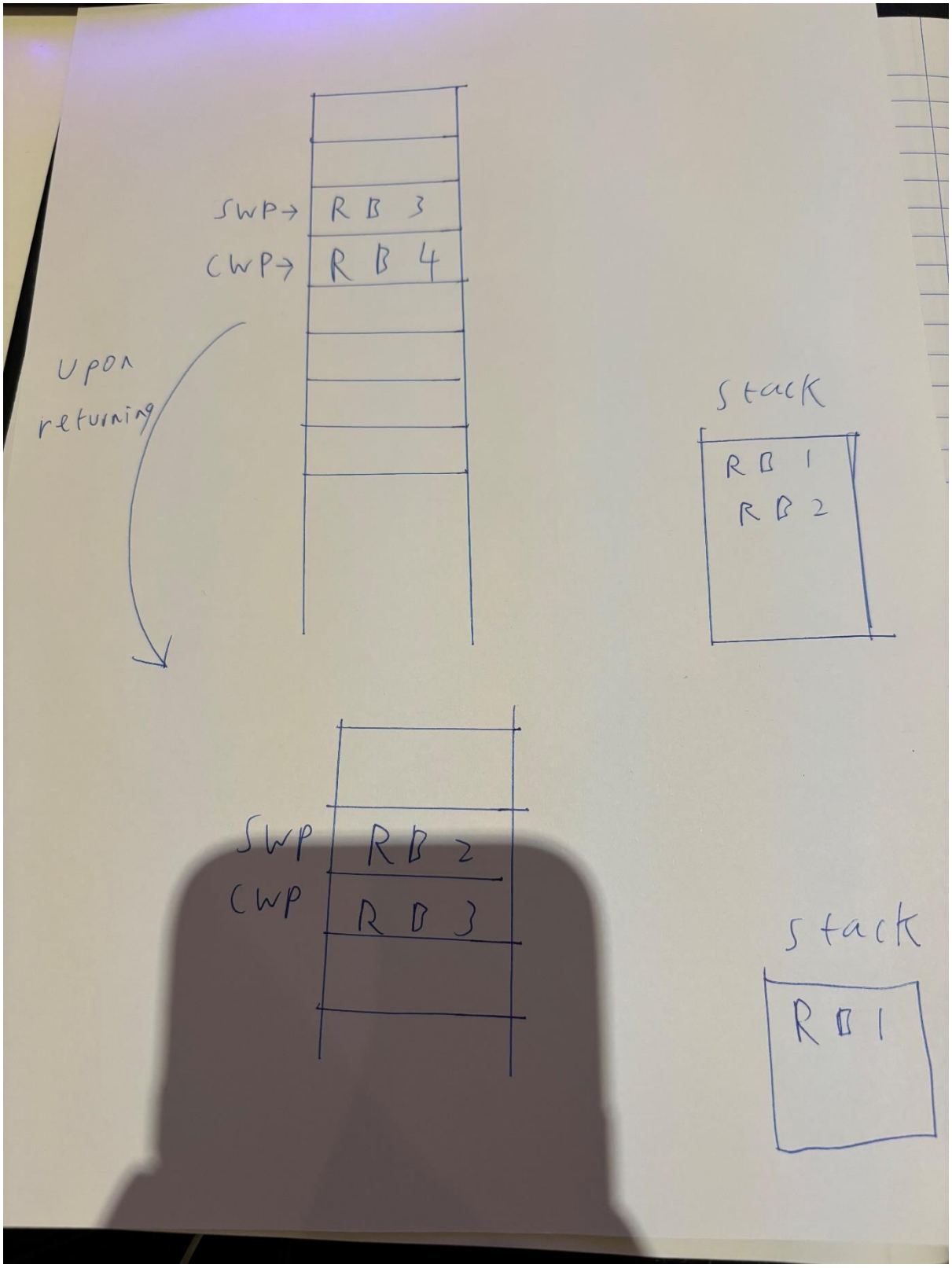
```
if (windowsUsed == numWindows){
        numOverflows++;
    }
    else{
        windowsUsed++;
    }
```

Important to note that 2 windows are occupied by default, 1 for global registers and 1 for function parameter registers. Upon entering main, a 3rd window is taken up meaning that upon entering ack3way for the first time, 4 windows are used up.

Counting the number of underflow occurrences:

Underflow occurs when you are returning from a function call, but there are only 2 register windows left. The number of windows used cannot drop below 2 as 2 windows are occupied by default as mentioned above. In this case, a register window has to be taken from the stack and placed back at the position above SWP, while the window at position CWP is returned.

Below is a diagram illustrating a situation in which underflow would occur.



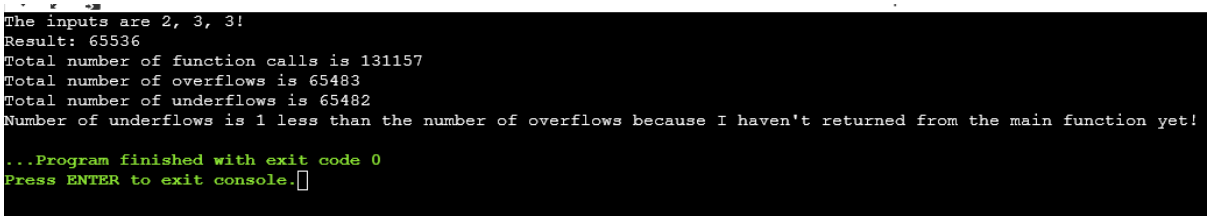
So above is the situation. Upon returning, SWP is decremented, RB2 moved to where SWP is, the window at CWP is returned and CWP is decremented to point to the position where RB3 is, as illustrated in the lower section of the drawing.

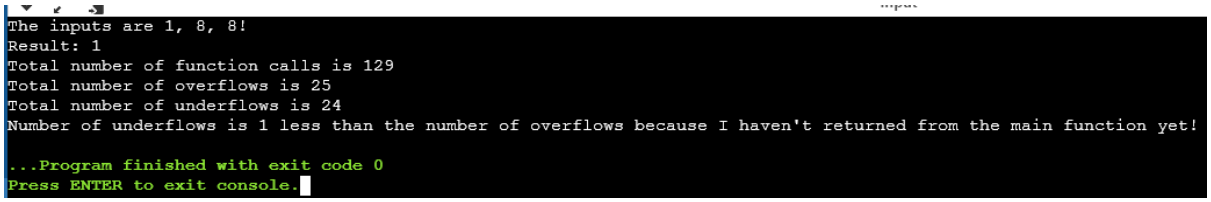
Code wise, this is counted by having this code before each return statement:

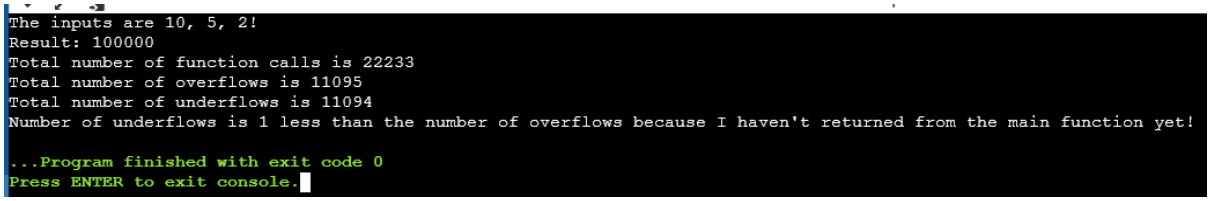
```
    currentFunctionDepth--;
    if (windowsUsed == 2){
        numUnderflows++;
    }
    else{
        windowsUsed--;
    }
```

If there are only 2 windows being used when returning, then an underflow occurs. Otherwise the number of windows used is decremented.

Results for the 3 different inputs:







As mentioned in the print statements, the number of underflows is 1 less than the number of overflows as the printing takes place within the main function. Once main returns, the number of underflows will be equal to the number of overflows.