**Tutorial 3 Report**

**Eligijus Skersonas 19335661**

**Question 2**

For question 2, I used Java to analyse computePascal(30,20).

**Main Method**

For the main method I have set the numberOfOverlappingWindows to 6,8 and 16 for each analysis and then called a method analyse to perform the analysis. Outside the main method I have defined a number of global variables as shown in the image below.

Text

Description automatically generated

**analysePascal method**

In this method I have a lot of print statements to show the results desired in question 2.

These results are obtained by calling the function riscComputePascal(30,20) which simulates a risc I simulation of compute Pascal.

This includes the number of overflows, procedure calls, underflows, maximum depth. I first calculate the results for the condition that overflows occur when all registers are utilised and then for when one register is empty. In each condition the variables are reset using the resetVariables() method.

Text

Description automatically generated

**riscComputePascal**

In this method I have the same code that was provided and call the methods onEntry() when entering the function and onExit() when exiting the function.

Text

Description automatically generated

**onEntry and onExit**

onEntry increments the depth and the number of calls when we enter the function. It then sets the max depth if the current depth is greater than the maxDepth. If the windows used is equal to the number of overlapping windows (i.e CWP+1 == SWP) then we increment the number of overflows otherwise we increment the number of register windows used.

onExit decrements the depth when we exit a function. If the windowsUsed is equal to the minimum number of active windows in this case 2 (i.e CWP-1 == SWP) then we increment underflows otherwise we decrement the number of windows being used.

Text

Description automatically generated

**6 Overlapping Registers Results 8 Overlapping Registers Results**

Text

Description automatically generated

Text

Description automatically generated

**16 Overlapping Registers Results**

Text

Description automatically generated

**Overflow when all registers windows utilised VS one register window empty**

It is evident from screenshots on the previous page that having overflows for when all registers are utilised results in significantly less amount of overflows than overflowing when one register windows is empty. Thus resulting in less read\write operations on the memory. To conclude overflowing when all registers are ultilised is more efficient.

Side note:

However from the lectures the increase in efficiency becomes smaller as the number of register windows get larger. In this case the efficiency would be significant and it would be worth providing more hardware. However if we were to use 128 overlapping the increased demand on hardware would be very questionable in realation to the increase in efficiency.

**Question 3**

For question 3, I used Java to time computePascal(30,20), I used the built in System.nanoTime() method to get the time just before calling the computePascal function and then used the method to get the time after returning from computePascal. I got the absolute difference between the 2 times and divided by 1000000. I decided to use this as it gives nanosecond precision. And the margin of error in the difference between the two times is 1ms. I did not use System.currentTimeMillis() as it contains more overhead leading to a larger margin of error.

Text

Description automatically generated