Architecture II Tutorial 3 Report

John Sinclair - 16325734

Q1

```
        add r0, #4, r2          ; 0 + 4 = r2 = inp_int

max     add r0, r26, r1         ; r1 = first param
        sub r27, r1, r0, {c}    ; compare b (second param) with v
        jle max1
        add r0, r27, r1
max1    sub r28, r1, r0, {c}    ; compare c (3rd param) with v
        jle max2
        add r0, r28, r1
max2    ret


max5    add r0, r2, r10     ; inp_int into first param r10
        add r0, r26, r11    ; i into r11 ready for max call
        add r0, r27, r12    ; j
        callr max
        add r0, r1, r10     ; put answer into r10 ready for next call
        add r0, r28, r11    ; k into r11
        add r0, r29, r12    ; l into r12
        callr max
        ret
```

Q2

```
; assumptions:
; mod - takes two params in r10 and r11, returning r10%r11 in r1
; div - takes two params r10 and r11, returns r10/r11 in r1

fun     sub r27, r0, r0, {c}
        jne fun1
        add r0, r0, r1
        ret

fun1    add r0, r27, r10
        add r0, #2, r11
        callr mod
        sub r1, r0, r0, {c}
        jne fun2

        ;add r0, r27, r10
        ;add r0, #2, r11
        callr div            ; b/2 now in r1
        add r26, r26, r10   ; a+a into r10
        add r0, r1, r11     ; b/2 in r11
        callr fun
        ret

fun2    ;add r0, r27, r10
        ;add r0, #2, r11
        callr div            ; b/2 now in r1
        add r26, r26, r10   ; a+a into r10
        add r0, r1, r11     ; b/2 in r11
        callr fun
        add r1, r26, r1
        ret
```

Q3/4

Code:

Imports and global variables:

```
#include <iostream>
#include <time.h>
using namespace std;

int procedureCalls, depth, tempDepth, underflows, overflows, windowSize, currentWindows, minWindows, windowSizeReduction;
```

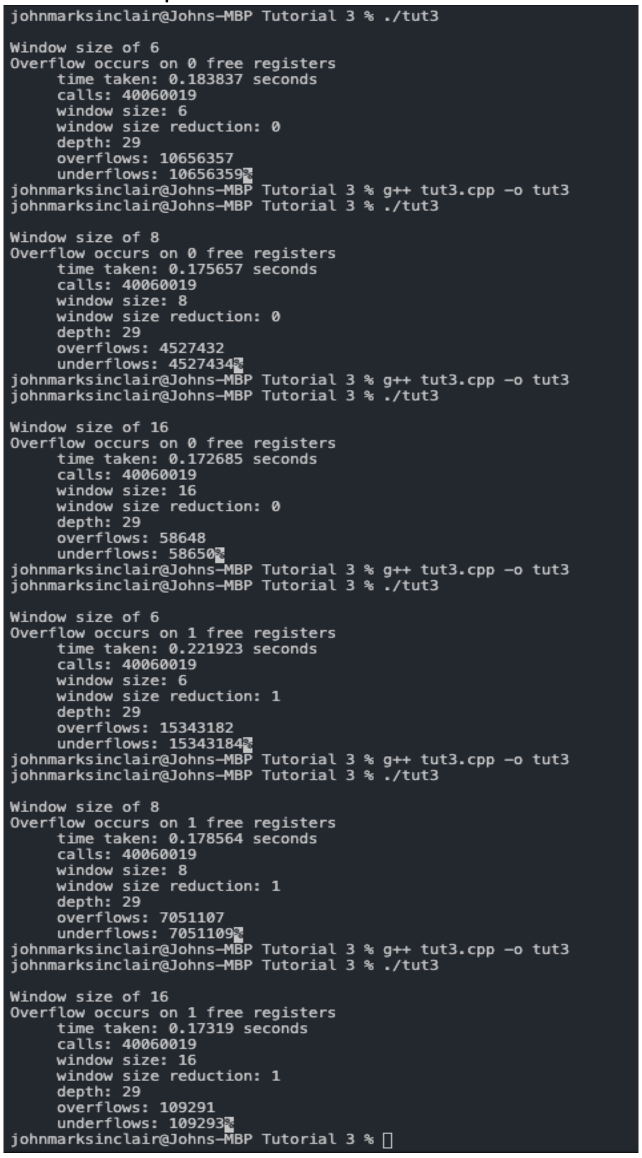
compute_pascal function:

```
int compute_pascal(int row, int position)
{
    procedureCalls++;
    tempDepth++;
    if (tempDepth > depth)
    {
        depth = tempDepth;
    }

    currentWindows++;
    if (currentWindows >= (windowSize - windowSizeReduction))
    {
        overflows++;
        currentWindows--;
    }

    if (position == 1)
    {
        tempDepth--;
        currentWindows--;
        if (currentWindows < minWindows)
        {
            underflows++;
            currentWindows++;
        }
        return 1;
    }
    else if (position == row)
    {
        tempDepth--;
        currentWindows--;
        if (currentWindows < minWindows)
        {
            underflows++;
            currentWindows++;
        }
        return 1;
    }
    else
    {
        int ans = compute_pascal(row - 1, position) + compute_pascal(row - 1, position - 1);
        tempDepth--;
        currentWindows--;
        if (currentWindows < minWindows)
        {
            underflows++;
            currentWindows++;
        }
        return ans;
    }
```

main function:

```
int main()
{
    procedureCalls = 0;
    depth = 0;
    tempDepth = 0;
    windowSize = 16;
    windowSizeReduction = 1;
    currentWindows = 0;
    minWindows = 2;
    overflows = 0;
    underflows = 0;
    clock_t start = clock();
    compute_pascal(30, 20);
    clock_t timeTaken = clock() - start;
    cout << "\nWindow size of " << windowSize;
    cout << "\nOverflow occurs on " << windowSizeReduction << " free registers";
    cout << "\n     time taken: " << (float)timeTaken / CLOCKS_PER_SEC << " seconds";
    cout << "\n     calls: " << procedureCalls;
    cout << "\n     window size: " << windowSize;
    cout << "\n     window size reduction: " << windowSizeReduction;
    cout << "\n     depth: " << depth;
    cout << "\n     overflows: " << overflows;
    cout << "\n     underflows: " << underflows;
    return 0;
}
```

Console Output:



```
johnmarksinclair@Johns-MBP Tutorial 3 % ./tut3

Window size of 6
Overflow occurs on 0 free registers
     time taken: 0.183837 seconds
     calls: 40060019
     window size: 6
     window size reduction: 0
     depth: 29
     overflows: 10656357
     underflows: 10656359%
johnmarksinclair@Johns-MBP Tutorial 3 % g++ tut3.cpp -o tut3
johnmarksinclair@Johns-MBP Tutorial 3 % ./tut3

Window size of 8
Overflow occurs on 0 free registers
     time taken: 0.175657 seconds
     calls: 40060019
     window size: 8
     window size reduction: 0
     depth: 29
     overflows: 4527432
     underflows: 4527434%
johnmarksinclair@Johns-MBP Tutorial 3 % g++ tut3.cpp -o tut3
johnmarksinclair@Johns-MBP Tutorial 3 % ./tut3

Window size of 16
Overflow occurs on 0 free registers
     time taken: 0.172685 seconds
     calls: 40060019
     window size: 16
     window size reduction: 0
     depth: 29
     overflows: 58648
     underflows: 58650%
johnmarksinclair@Johns-MBP Tutorial 3 % g++ tut3.cpp -o tut3
johnmarksinclair@Johns-MBP Tutorial 3 % ./tut3

Window size of 6
Overflow occurs on 1 free registers
     time taken: 0.221923 seconds
     calls: 40060019
     window size: 6
     window size reduction: 1
     depth: 29
     overflows: 15343182
     underflows: 15343184%
johnmarksinclair@Johns-MBP Tutorial 3 % g++ tut3.cpp -o tut3
johnmarksinclair@Johns-MBP Tutorial 3 % ./tut3

Window size of 8
Overflow occurs on 1 free registers
     time taken: 0.178564 seconds
     calls: 40060019
     window size: 8
     window size reduction: 1
     depth: 29
     overflows: 7051107
     underflows: 7051109%
johnmarksinclair@Johns-MBP Tutorial 3 % g++ tut3.cpp -o tut3
johnmarksinclair@Johns-MBP Tutorial 3 % ./tut3

Window size of 16
Overflow occurs on 1 free registers
     time taken: 0.17319 seconds
     calls: 40060019
     window size: 16
     window size reduction: 1
     depth: 29
     overflows: 109291
     underflows: 109293%
johnmarksinclair@Johns-MBP Tutorial 3 % 
```

Q4

I used the clock function of clock_t imported from time.h. the clock function can result in inaccuracies as the function itself is arbitrary so you must use the macro CLOCKS_PER_SEC to convert the value to real time.
I found the length of time to compute the release version of compute_pascal was about 0.18 seconds.