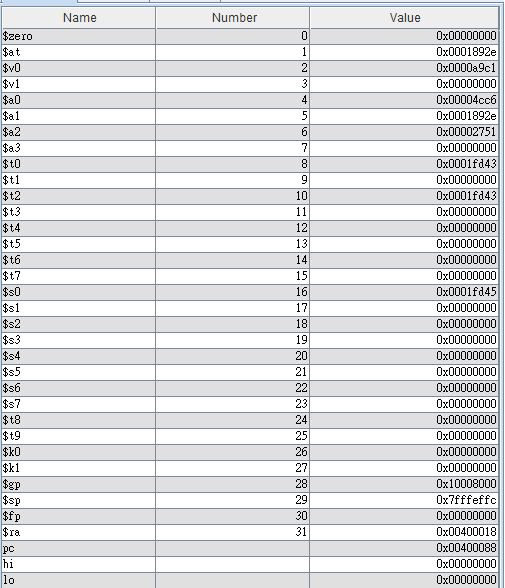
**Computer Organization 2019**

**HOMEWORK 2**

系級: 資訊112 學號: F74082086 姓名: 王維瀚Program 1 (Find the average)

**實驗結果圖:**



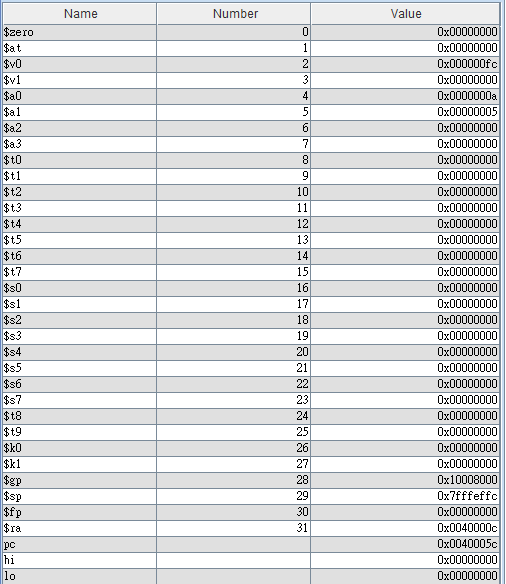
**程式運作流程:**

本題我利用趨近的方式來取得除以3所得到的效果。a/3可表示成a(1/4+1/16+1/64+……)，根據等比級數的公式，後面的係數會趨近於1/3。而1/4, 1/16, 1/64等分別可透過right shift 2, 4, 6得到。所以我先將三個數相加的結果存在$s0，接著就是將$s0>>2, $s0>>4, $s0>>6等數相加一直加到$s0>>32。這樣的算法可以支持32 bits的總和。但因為在right shift的過程中會有lose bits的問題(數字不能被4整除)，而本來的總和也有可能無法被3整除的問題。所以我先將答案相加三次。得到的結果跟本來的總和-2(除以3餘數最多為2)做比較，不論原本總和/3是餘0,餘1或餘2，減2後的數如果還比得到的結果大，表示答案一定有lose bits，所以要對答案做微調(加1加特定次數)，直到答案相加三次的結果不比總和-2小為止。

**心得**

原本這題我想用課本上敘述MIPS的除法運作原理來寫，而且是使用nonrestoring的方法，但是在將演算法轉換成過程中可能有錯誤而導致沒有辦法實現。我需要再更熟悉這個原理運作的流程，或許就可以成功的使用這個原理了。在用後來的方法時，為了修正答案而進行了一些微調，不確定是否會有其他inputs導致這樣的微調無法得到正確答案。總感覺還可以再寫得更完善一點。

Program 2 (Pascal)

**實驗結果圖:**

**程式運作流程:**

先將參數存進stack中。對每一層caller而言，都會保留他們各自的參數(比如對C2取1而言，保存了$a0 = 2, $a1 = 1)。接著判斷是否有$a0 = $a1或0的狀況，若有，直接跳到L1，將回傳值+1(初始化$v0為0)，然後就直接返回原本的caller；否則，就先將$a0-1做為新的參數呼叫pascal，也就是實現pascal(n-1, m)的部分。等到返回後再將$a1也-1做為新參數呼叫pascal(傳入的參數為n-1, m-1)。每個呼叫的pascal如果不符合$a0 = $a1或0，都會呼叫下一層的pascal，直到展開到符合狀況為止。只有在此時才會去增加$v0的值。這裡的$v0有點像全域變數的概念，不論現在的caller是哪一層，$v0都是共用的。最後就是將stack中的參數和$ra都load回來，回復stack的狀況。再返回原caller，也就是主函式。運算的結果則已經存在$v0裡面了。

**心得**

本題看起來有點複雜，實際上只要知道呼叫遞迴的時機就變得相當的簡潔了。一開始會想輪流將$a0-1, $a1和$a0-1, $a1-1都傳進pascal後，各自得到的結果是不是還要加起來，就像高階語言所描述的那樣。後來想到如果$a0 = $a1或0會直接branch到回傳1的部分，那就在那個時候(一個組合已經展開到不能展開時)，直接把回傳的1加到結果就好。所以不管是哪一個組合，就是將它拆解成許多的1相加，看最後加起來是多少即為答案。這樣一想之後，程式就易懂許多了。