# Cmpe344 Fall 2021 FF34 Experiment #3: Writing a non-leaf procedure

In this experiment, you will write a non-leaf procedure in MIPS to complete a given program.

Once complete, the program will read elements and in pairs from the two input arrays N and K (given in the data segment) until they end. For each pair, it will write the result of  *choose*  to the output array C.

* The arrays N and K are of equal length.
* Arrays end with a zero, which are not considered as members.
* All the input and output values will be **unsigned words**.
* We guarantee you that for all .

The following two procedures are already written and **must be left as they are:**

| factorial: | Clears all the temporary registers as a challenge. Takes an argument in $a0 and returns its factorial in $v0. |
| --- | --- |
| main: | Stores values into saved registers as a challenge. Reads elements from N and K pairwise, calls the combination procedure with each pair, and writes the result into the array C. |

Your task is to write the combination procedure by the following constraints:

1. It should take two arguments via $a0 and $a1, and return the result of

*choose*

in $v0, where denotes the factorial of .

1. Each of the three factorials must be obtained using the factorial procedure. (You may **NOT** simplify the formula.)
2. Registers that should be preserved[[1]](#footnote-0) must be preserved.
3. Registers that do not need preserving1 must be assumed to have been altered after each time calling the factorial procedure.

| See the promised incomplete program **and the question** on the following **two pages.** |
| --- |

| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52** | .text  .globl main  main:  *li* $s1,1;*li* $s2,2;*li* $s3,3 # every time the made\_up procedure ends,  *li* $s4,4;*li* $s5,5 # these registers’ original values must be restored  *li* $s6,6;*li* $s7,7 # VALUES WILL CHANGE, use stack to store-restore  *addi* $sp, $sp, -4  *sw* $ra, 0($sp) # push($ra)  *li* $s0, 0 # $s0 = array indexer  main\_loop:  *lw* $a0, N($s0) # $a0 = N[$s0]  *beqz* $a0, main\_done # end loop if N[$s0] == 0  *lw* $a1, K($s0) # $a1 = K[$s0]  *jal* combination # $v0 = $a0 choose $a1  *sw* $v0, C($s0) # C[$s0] = $v0  *addi* $s0, $s0, 4 # $s0 += 4  *b* main\_loop # loop  main\_done:  *sw* $zero, C($s0) # mark the end of the array C  *lw* $ra, 0($sp) # pop($ra)  *addi* $sp, $sp, 4  *jr* $ra  # DO NOT CHANGE THE CODE ABOVE  combination:  # WRITE YOUR CODE HERE  *jr* $ra  # DO NOT CHANGE THE CODE BELOW (you can change the data arrays)  factorial:  *li* $a1,0;*li* $a2,0;*li* $a3,0;*li* $v1,0 # combination procedure  *li* $t0,0;*li* $t1,0;*li* $t2,0;*li* $t3,0 # should not rely on these  *li* $t4,0;*li* $t5,0;*li* $t6,0 # being preserved, because...  *li* $t7,0;*li* $t8,0;*li* $t9,0 # all the temporaries are cleared!  *li* $v0, 1 # $v0 = 1  factorial\_loop:  *beqz* $a0, factorial\_done # end loop if $a0 == 0  *mul* $v0, $v0, $a0 # $v0 \*= $a0  *addi* $a0, $a0, -1 # $a0 -= 1  *b* factorial\_loop # loop  factorial\_done:  *jr* $ra  .data  N: .word 5, 6, 12, 3, 2, 0  K: .word 2, 2, 3, 2, 1, 0  C: .word 0 |
| --- | --- |

Running the completed program should write the following array beginning at C:

10, 15, 220, 3, 2, 0

## Question

In MIPS, assume that the **and R1, R2, R3** was not a real instruction, but instead a pseudo-instruction. The assembler would then have to convert it into a sequence of one or more real instructions. Provide such a sequence below. Use **R1**, **R2**, and **R3** to denote the operands passed to **and**. Use **R4**, **R5**, **R6**, … to denote any additional register you use.

nor R4, R2, R2 #NEGATE R2

nor R5, R3, R3 #NEGATE R3

#since (r2 ^ r3) = (r2’ v r3’)’

nor R1,R4,R5

## Demo and Submission

**Make a demo** of your code to an assistant until 12:50, strict.

**Submit the following** to Moodle (for the online participants) or as directed by the assistants (for the on-site participants) until 12:50, strict:

1. **This document** with your answers filled in, with the file name

lab3\_<GROUPID>.docx

1. **Your source code** performing the described operations, with either one of the following file names

lab3\_<GROUPID>.asm  
lab3\_<GROUPID>.txt

E-mail submissions will **NOT** be accepted.

1. See the 7th note in the *MIPS assembly language overview* section of the tutorial. [↑](#footnote-ref-0)