# WORKSHEET E: COMPILERS AND MACHINE CODE

Version 1.0
BSc Computing for Games
COMP110

Dr Ed Powley

## Introduction

In this worksheet, you will complete some exercises involving translation of programs between assembly code and high-level Python code.

Nowadays, programmers rarely write assembly code by hand as modern compilers do a much better job. However, some understanding of how high-level coding constructs translate into low-level machine code is useful in understanding the behaviour and performance characteristics of your programs.

To complete this worksheet, complete parts (a)–(e) below.

(a) **Write** a piece of Python code equivalent to the following MIPS assembly code:

```
        addi $s2, $zero, 0
        addi $s3, $s0, 0
inner:  add $s2, $s2, $s1
        addi $s3, $s3, -1
        bne $s3, $zero, inner
```

(b) Thus **explain** why the code has the effect of multiplying the values in registers `$s0` and `$s1` and storing the result in `$s2`.

(c) **Write** a piece of Python code equivalent to the following MIPS assembly code:

```
        addi $s0, $zero, 10
        addi $s1, $zero, 1

outer:  addi $s2, $zero, 0
        addi $s3, $s0, 0
inner:  add $s2, $s2, $s1
        addi $s3, $s3, -1
        bne $s3, $zero, inner
        addi $s1, $s2, 0
        addi $s0, $s0, -1
        bne $s0, $zero, outer
```

(d) Thus **explain** why the code has the effect of calculating the factorial of the value in `$s0` and storing the value in `$s1`; that is, if `$s0` $= 10$ then `$s0` $= 10 \times 9 \times \cdots \times 2 \times 1 = 3\,628\,800$.


The MIPS architecture was popular in the 1980s–2000s, and is still in use today in embedded applications and in education.

A more efficient Python program for calculating the factorial could look like this:

```
s0 = 10
s1 = 1

while s0 != 0:
    s1 *= s0
    s0 -= 1
```

(e) **Write** a MIPS assembly program equivalent to the Python code above. Hint: you will need to find out how to use the `mult` and `mflo` MIPS instructions. Check your program using a MIPS simulator.

## Submission instructions

If you did not already do so for a previous worksheet, **fork** the GitHub repository at the following URL:

`https://github.com/Falmouth-Games-Academy/comp110-worksheets`

Create a file named `worksheet_E/readme.md`, and write your solutions to the above tasks. When you have finished, open a **pull request**. To correctly format code listings in a markdown document, they should be wrapped in triple backticks like so (the backtick symbol is just below the Escape key on a standard British PC keyboard):

````
Text here

```
def my_code_listing():
    print "Hello, world!"
```

More text here
````

Attend the scheduled worksheet feedback session on **Monday November 21st 2016**, ensuring that you have uploaded all material to GitHub and opened a pull request before this time.

Note that the deadline for this worksheet may be very close to the deadline for final summative submission of all worksheets. Please see MyFalmouth for details of deadlines, and the Assignment Brief for COMP110 assignment 1 for summative submission instructions.

## Marking criteria

Remember that **it is better to submit incomplete work than to submit nothing at all**. Any attempt, even unfinished, will receive a passing grade.

Your work will be marked according to the following criteria:

- Where appropriate, are your answers **correct**?
- Are your explanations **clear**, **concise** and **accurate**?
- Where you have obtained information from external sources, are they **properly cited**?