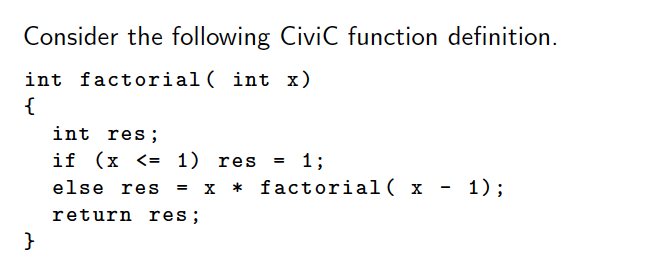
**Assignment 6**

Andrea & Aynel



**Assignment 21: Code generation**

[a] Manually generate CiviC-VM assembly code for the above function definition making use of labels to mark destinations of jump instructions.

[b] Point out the relationship between assembly code and source code through line comments in the assembly code.

See above.

[c] Add the number of bytes required for each line of CiviC-VM assembly code. Assume here jump instructions would take byte code offsets as arguments and not labels.

[d] Compute the proper byte code offset for each jump instruction; consult the CiviC-VM manual for details on individual instructions. ???

|  |  |  |  |
| --- | --- | --- | --- |
| [a] | | [b] | [c] & [d] |
| **factorial**:  esr 2  istore 0  iload 1 | // go into factorial  // load variable res­  // load variable x­ ­­­­ | | Bytes: 1 + 1  bytes: 1 + 1  bytes: 1 + 1 |
| iloadc\_1 | // load constant 1 | | bytes: 1 |
| ile | // if x <= 1 | | bytes: 1 |
| branch\_f **L1** | // go into if block if true | | bytes: 2 (offset: 13) |
| istore 0 | // res = 1; | | bytes: 1 + 1 |
| jump **L2** | // skip else block | | bytes: 2 (offset: |
| **L1:**  iload 0  iloadc\_1  isub  jsr 2  iload 0  imul  istore 1 | // else  // load variable x  // load constant 1  // sub x – 1  // factorial(x - 1)  // load variable x  // multiply x \* factorial ( x – 1)  // store result in res | | bytes: 1 + 1  bytes: 1  bytes: 1  bytes: 1 + 2 + 1  bytes: 1 + 1  bytes: 1  bytes: 1 + 1 |
| **L2:**  ireturn | // return res | | bytes: 1 |

**Assignment 22: Compilation Schemes Revisited**

Devise a compilation scheme that replaces each occurrence of a for-loop in the body of a CiviC function by semantically equivalent CiviC code that makes use of a while-loop instead. As a simplification consider only for-loops without a step specification and assume that CiviC would support arbitrary interleaving of variable declarations and statements in function bodies following the example of C99.

Original for-loop:

| for ( int i = lower, upper) { |

C | *Body* |

| } |

| *Rest* |

Replacement:

| int i = lower; |  
 | while (i < upper) { |

-> | C|*Body*| |

| i += 1; |

| } |

| C|*Rest|* |