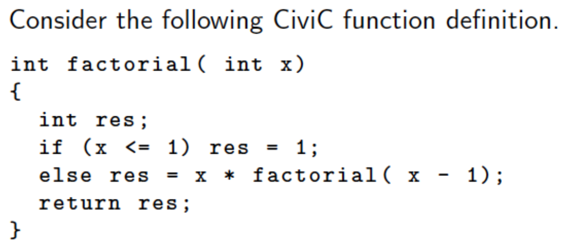
Assignment 6

Andrea & Aynel


Consider the following CiviC function definition.

```
int factorial( int x)
{
  int res;
  if (x <= 1) res = 1;
  else res = x * factorial( x - 1);
  return res;
}
```

Assignment 21: Code generation

[a]	[b]	[c] & [d]
factorial:		
esr 1	// go into factorial function	bytes: 2
istore 0	// init variable res	bytes: 2
iload 1	// load variable x	bytes: 2
iloadc_1	// load constant 1	bytes: 1
ile	// if x <= 1	bytes: 1
branch_f **L1**	// go into if ^ == true	bytes: 3 (offset: 9)
iloadc_1	// load constant 1	bytes: 1
istore 1	// store res = 1;	bytes: 2
jump **L2**	// skip else block	bytes: 3 (offset: 17)
L1:	// go into if x <= 1 == false	
isrg	// start functiecall factorial	bytes: 1
iload 0	// load variable x	bytes: 2
iloadc_0	// load constant 1	bytes: 1
isub	// sub x – 1	bytes: 1
jsr 1	// factorial(x - 1)	bytes: 4
iload 0	// load variable x	bytes: 2
imul	// multiply x * factorial (x – 1)	bytes: 1
istore 1	// store result in res	bytes: 2
L2:		
ireturn	// return res	bytes: 1

Assignment 22: Compilation Schemes Revisited

Original for-loop:

```
    | for ( int i = lower, upper) {          |
C |       Body                               |
    | }                                      |
    |       Rest                             |
```

Replacement:

```
     | int i = lower;                        |
     | while (i < upper)  {                  |
->  |     C |Body |                          |
     |     i += 1;                           |
     | }                                     |
     |  C |Rest |                            |
```