# Assignment1: Micro Stack Machine Description

Radu Ionita, Nicolae Mariuta

14.09.2014

Unfortunately the project is not running but we worked hard on it, but because we had problems with understanding how Monads work, we did not manage to solve much of the task.

So far, we managed to do the following:

* We initiated the types *Prog, Stack, Regs*
* Data *State* which cotrains the Program to be executed, the stack , registers and program counter
* We created data *Error* to include the *errorType* and the *state* so when an error appears, we will be able to show details about the error and the program stats when the error appears
* The function *initial* that initializes the MSM with a program, a set of registers, the stack and the PC which is set at 0
* We created the MSM *newtype* to be instance of Monad, Functor and Application to make the instructions and other functions to be easier to implement
* function *get* and *set* which change and return the state of the running MSM
* function *modify* to change the state of the MSM
* *getinst*  function that returns the stats of the current program and returns error if Program Counter is not valid
* *interp* function that runs the MSM by running the instructions
* *interpInst* function which reads the instruction and based on the type of instruction, the corresponding function is called
* *interpInst*  function that interprets a given instruction
* *showError* function that displays the type of error that appeared

We also implemented the functions corresponding to the instructions of the MSM:

* Function PUSH that adds an element to stack and returns message error (True if it was successful and False in case of stack overflow )
* function POP that removes the first element in the stack and returns it
* function DUP
* function SWAP
* function NEWREG which creates a new register
* function LOAD
* function STORE
* function NEG which changes the sign of the number
* Function ADD which pops first 2 elements in the stack and adds them. The result is pushed back to the stack
* function JMP that makes the MSM to jump at specified instruction
* function CJMP which is conditional jump; makes jump at instruction if condition is true

## Conclusion

We still have to implement all the instructions and make the Micro Stack Machine to work properly. We do not want to give up with solving the assignment and we would like to keep working on it because we have some clues how to continue, and if we get little more help from the TAs during the next week we will have it finished.