LAB 04

HUMANOID COMPILER

Your Job:

Complete and compile the given C code into LC-3 assembly language manually, and then assemble it into LC-3 object file.

Code Snippet:

```
typedef int i16;
typedef unsigned int u16;

i16 func(i16 n, i16 a, i16 b, i16 c, i16 d, i16 e, i16 f){  //Lots of arguments
    i16 t = GETC() - '0' + a + b + c + d + e + f;
    if(n > 1){
        i16 x = func(n - 1, a, b, c, d, e, f);
        i16 y = func(n - 2, a ,b, c, d, e, f);
        return x + y + t - 1;
    }else{
        return t;
    }
}


i16 main(void){
    i16 n = GETC() - '0';
    return func(n, 0, 0, 0, 0, 0, 0);
}

_Noreturn void __start(){
    /*
        Here is where this program actually starts executing.
        Complete this function to do some initialization in your compiled assembly.
        TODO: Set up C runtime.
    */
    u16 __R0 = main(); //The return value of function main() should be moved to R0.

    HALT();
}
```

Details:

- The code must be translated AS IT IS. You might want to add some optimizations, but DO FOLLOW the C standard. In additional, optimizations must ONLY be done within basic blocks, which means you CANNOT change the control flow structure. For example, you cannot optimize the recursion into iteration, although the functionality is the same.
- Any usage of 3rd party C compiler is NOT allowed.
- __start() is actually not a function. The program starts from there, and then calls main function. At first the registers and memory are in random status. So, you should design and complete the

initialization process yourself. Notice that this process should be general, which means it must have nothing to do with the content of the main() or any other functions in this program.

- GETC() and HALT() are traps. Trap routines may alter the contents of registers and memory, thus disturbing your program. So, you may work out a standard for trap routines to follow. Notice that all callees in your program should also follow this standard.
- If another program wants to call func() in your program, it must have to follow certain rules. Design these rules as a standard. Also, all callers in your program should follow this standard.
- Your program will be RANDOMLY loaded to x3000 – xC000. Thus the .ORIG clause and the first two bytes of object file will be ignored.
- Your report should at least contain: 1. Initialization process; 2. Calling conventions; 3. Other standards you designed; 4. Error handling.

Submit your program:

The program you submit to our server is the object file, the source code file and the report.

Save your .obj file, and give it the name ID_Lab04.obj.

Save your .asm file, and give it the name ID_Name_Lab04.asm.

Give your report the name ID_Name_Lab04.pdf.
Put all above in a directory named after your student number in uppercase and pack it using TAR with GZIP compression.

Your scores:

Correctness	50%
Report	40%
Great ideas	10%