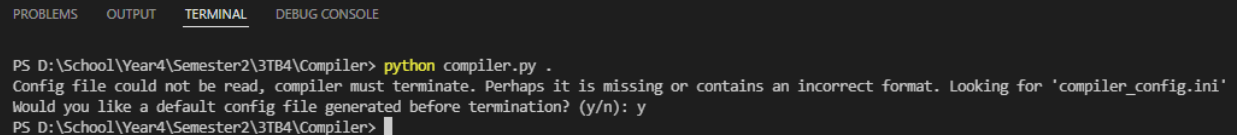


3TB4 Lab 5 Compiler Tutorial

The compiler provided is a single file python program, which will help you to quickly write and test code for the Stepper Motor Controller ASIP. It requires only python 3.6 or greater installed on your machine. To get up and running with this compiler, do the following:

1. Move the comiler.py file into a folder where you want all of your programming to be done. The compiler will generate the .mif files and place them in the same directory as the compiler.
2. Run the compiler once to generate a configuration file. This file will be placed in the same directory as the compiler and allows you to change the functionality of the compiler should you wish. However, the default configuration file generated should work properly for this lab.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS D:\School\Year4\Semester2\3TB4\Compiler> python compiler.py .
Config file could not be read, compiler must terminate. Perhaps it is missing or contains an incorrect format. Looking for 'compiler_config.ini'
Would you like a default config file generated before termination? (y/n): y
PS D:\School\Year4\Semester2\3TB4\Compiler> █
```

3. Tutorial files are provided in this .zip folder for you to read through and follow along with. Their contents and output are also pasted below. Happy coding!

≡ compiler_tut_01.txt

```
1 ; Welcome to compiler tutorial #1
2
3 ; Syntax:
4 ; 1. Semicolon's are comments, anything after a Semicolon
5 ; will be ignored during compilation
6 ; 2. All instructions start with an instruction name, in
7 ; all caps. We currently support the following instructions:
8 ; BR,BRZ,ADDI,SUBI,SR0,SRH0,CLR,MOV,MOVA,MOVR,MOVRHS,PAUSE
9 ; 3. Immediate constants are preceded by a pound, #
10 ; 4. Register addresses are preceded by the letter r
11 ; 5. Instruction parameters can be separated by a space, comma
12 ; or both. The compiler is not that fussy
13 ; 6. Only one instruction is allowed per line
14
15 ; Let's see some code
16
17 CLR r0 ; clear Register 0 on start
18 CLR r1 ; clear Register 1 on start
19
20 SR0 #10 ; Set lower half of Register 0 to b1010
21 MOV r1 r0 ; Move the Register 0 to Register 1
22
23 MOVR r1 ; Move the motor by the value in Register 1
24 PAUSE ; Pause for 8/100's of a second
25 PAUSE ; Pause for 8/100's of a second
26 MOVRHS r1 ; Move the motor half steps by the value in Register 1
27 PAUSE ; Pause for 8/100's of a second
28 PAUSE ; Pause for 8/100's of a second
29 SUBI r0 #1 ; Subtract 1 from Register 0
30 BRZ #2 ; If Register 0 is 0, jump forward 2 instructions
31 BR #-8 ; Jump backward 8 instructions
32
33 SR0 #8 ; Set lower half of Register 0 to b1000
34 SRH0 #8 ; Set upper half of Register 0 to b1000
35 | | | ; Register 0 is now equal to signed -120
36
37 MOVRHS r0 ; Move the motor half steps by the value in Register 0
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
PS D:\School\Year 3\Semester 2\3TB4\Lab5> python compiler.py compiler_tut_01.txt
Memory Initialization File Successfully Generated!
Output File Name 'compiler_tut_01.mif'
```

ASM compiler_tut_02.s

```
1  ; Welcome to compiler tutorial #2
2
3  ; Command Line Parameters:
4  ; 1. To get a full list of command line parameters, run the compiler
5  ;    with the oh flag
6  ; 2. For this tutorial, we will change the DEPTH of the mif file to 512
7  ; 3. We will also specify the name of the output file
8
9  ; Handling Errors:
10 ; 1. The compiler will handle most errors without crashing
11 ; 2. Even if I forget a Semicolon, the compiler will figure out
12 ;    that these lines should have been comments
13 ; 3. Any errors in the assembly code will be printed out to the terminal
14 ;    when the compiler is run
15 ; 4. If you forget to prepend an instruction with # or r, the compiler
16 ;    will try its best to infer what you meant to do
17 ; 5. Even if there are errors in your assembly code, the compiler will
18 ;    generate the mif file anyways, along with a warning. Just in case
19 ;    what you did was intentional. Though be warned, it likely will not
20 ;    function as you thought it would.
21
```

```
24 ; Let's see some code
25
26 CLR 0          ; Here I forgot and r before the Register, but it will
27 CLR 1          ; be inferred by the compiler
28
29 SR0 #10
30 MOV r1 r0
31
32 MOVR r1
33 PAUSE
34 PAUSE
35 MOVRHS r1
36 PAUSE
37 PAUSE
38 SUBI 0 1       ; Here I forgot both the r and #.
39 BRZ #2
40 BR #-8
41
42 SR0 #8
43 SRH0 #8
44
45
46 MOVRHS r0
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS D:\School\Year4\Semester2\3TB4\Compiler> python compiler.py .\compiler_tut_02.s -o compiler_tutorial_002.s -d 512
Warning on line 26: Operator Value '0' missing identifier. Inferring as Register

Warning on line 27: Operator Value '1' missing identifier. Inferring as Register

Warning on line 38: Operator Value '0' missing identifier. Inferring as Register

Warning on line 38: Operator Value '1' missing identifier. Inferring as Immediate Constant

Memory Initialization File Successfully Generated With WARNINGS. This Code May Not Function As Intended!
Output File Name 'compiler_tutorial_002.mif'
```