System Software Practicals

Assignment 3

Krunal Rank
U18CO081

1. Implement First Pass Assembler(Symbol Table,Literal Table,Pool Table and Table of Incomplete Instructions) for multiplication of two numbers.

```
import argparse

# Machine Opcode Table
MOT =
{"MOVER":1,"MOVEM":2,"ADD":3,"SUB":4,"MULT":5,"DIV":6,"SIC":7,"COMP":8,"PRINT":9,"REA
D":10}

# Pseudo Operation Table
POT = {"START":1,"END":2,"EQU":3,"ORIGIN":4,"LTORG":5}

DL = {"DS":1,"DC":2}

# Registers
REGISTERS = {"AREG":1,"BREG":2,"CREG":3,"DREG":4}

# Parsing FilePath as Arguments
parser = argparse.ArgumentParser(description="Generates Target Code for Assembly
Source Code")
parser.add_argument('file_path',metavar='filePath',help='File Path to Assembly Source
Code')
args = parser.parse_args()
file_path = args.file_path

# Table of Incomplete Instructions
TII = {}

# Symbol Table
ST = {}

# Pool Table
PT = []

# Literal Table
LT = {}
```

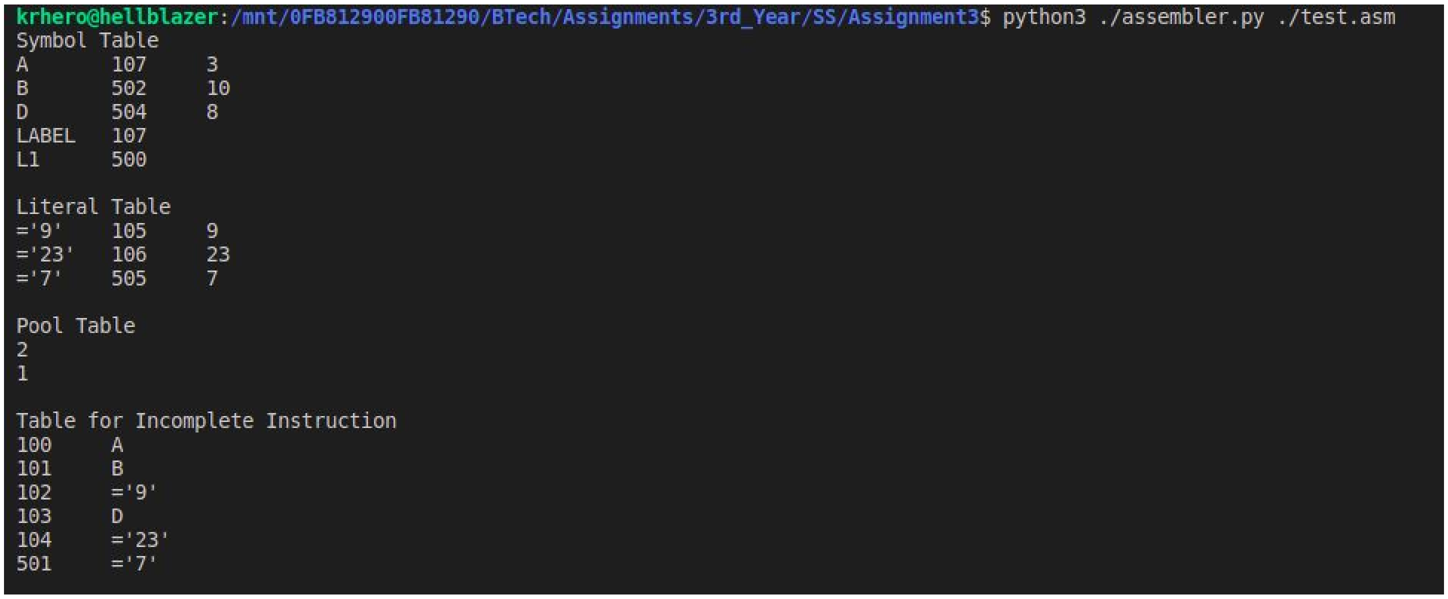
```
# Program Counter
PC = 0



# Target Code
TC = {}
temp = []

# Parsing the File
try:
   with open(file_path,'r') as f:
       lines = f.readlines() # lines = List of lines in file f
       line_count = 1
       for line in lines:
           line = line.strip().upper()
           line_code = []
           operator = line.split(' ')[0]
           if MOT.get(operator,-1)!=-1:
               if(operator=="PRINT" or operator=="READ"):
                   line_code.append(PC)
                   line_code.append(operator)
                   operands = line.split(' ')[1:]
                   if len(operands)!=1:
                       raise Exception("Invalid Operand Count! Only 1 Operand of type
Symbol Format Required")
                   if ST.get(operands[0],-1)==-1:
                       ST[operands[0]] = {"Address":-1,"Value":"-1"}
                       TII[PC] = operands[0]
                   line_code.append(operands[0])
                   PC = PC + 1
                   continue

               operands = line.split(' ')[1].split(',')
               operands = [a.strip() for a in operands]
               if REGISTERS.get(operands[0],-1)==-1:
                   raise Exception("Invalid Instruction Format! Expected Register!")
               if(operands[1].startswith('=')):
                   if LT.get(operands[1],-1)==-1:
                       LT[operands[1]]={"Address":-1,"Value":int(operands[1][2:-1])}
                       TII[PC]=operands[1]
               else:
                   if ST.get(operands[1],-1)==-1:
                       ST[operands[1]]={"Address":-1,"Value":"-1"}
                       TII[PC]=operands[1]
```

```
                line_code.append(PC)
                line_code.append(operator)
                line_code.append(operands[0])
                line_code.append(operands[1])
                PC = PC + 1

            elif POT.get(operator,-1)!=-1:
                if operator=="START":
                    operand = int(line.split(' ')[1])
                    line_code.append(PC)
                    line_code.append(operator)
                    line_code.append(operand)
                    if operand<PC:
                        raise Exception("Operand cannot be less than current Program
Counter!")
                    PC = operand
                elif operator=="ORIGIN":
                    operand = int(line.split(' ')[1])
                    line_code.append(PC)
                    line_code.append(operator)
                    line_code.append(operand)
                    if operand<PC:
                        raise Exception("Operand cannot be less than current Program
Counter!")
                    PC = operand
                elif operator=="LTORG":
                    cnt = 0
                    for (key,val) in TII.items():
                        if(LT.get(val,-1)!=-1):
                            if(LT[val]["Address"]==-1):
                                LT[val]["Address"] = PC
                                line_code = []
                                line_code.append(PC)
                                line_code.append(LT[val]["Value"])
                                temp.append(line_code)
                                line_code = []
                                cnt = cnt + 1
                                PC = PC + 1
                    if cnt!=0:
                        PT.append(cnt)
                elif operator=="END":
                    cnt = 0
                    for (key,val) in TII.items():
                        if(LT.get(val,-1)!=-1):
```

```python
                        if(LT[val]["Address"]==-1):
                            LT[val]["Address"] = PC
                            line_code = []
                            line_code.append(PC)
                            line_code.append(LT[val]["Value"])
                            temp.append(line_code)
                            line_code = []
                            cnt = cnt + 1
                            PC = PC + 1
                for (key,val) in TII.items():
                    if(ST.get(val,-1)!=-1):
                        if(ST[val]["Address"]==-1):
                            ST[val]["Address"] = PC
                            line_code = []
                            line_code.append(PC)
                            line_code.append(ST[val]["Value"])
                            temp.append(line_code)
                            line_code = []
                            cnt = cnt + 1
                            PC = PC + 1
                if cnt!=0:
                    PT.append(cnt)
            elif operator=="EQU":
                line_code.append(PC)
                operand = line.split(' ')[1]
                if(ST.get(operand,-1)==-1):
                    raise Exception("Invalid Operand! Symbol Not Found!")
                line_code.append(operand)
                PC = PC + 1
        elif 'EQU' in line:
            ops = line.split(' ')
            label = ops[0][:-1].strip()
            ref = ops[2].strip()
            line_code.append(PC)
            line_code.append(label)
            line_code.append(ops[1])
            line_code.append(ref)
            if(ST.get(label,-1)!=-1):
                raise Exception("Label Already Used!")
            if(ST.get(ref,-1)==-1):
                raise Exception("Reference not found!")
            ST[label]={"Address":ST[ref]["Address"]}
            PC = PC + 1
        elif ':' in line:
```

```
        label = line.split(':')[0]
        ST[label] = {"Address":PC}
        line_code.append(PC)
        line_code.append(label)
        PC = PC + 1
    else:
        operands = line.split(' ')
        symbol = operands[0]
        ST[symbol] = {"Address":PC,"Value":int(operands[2])}
        line_code.append(PC)
        line_code.append(symbol)
        line_code.append(operands[1])
        line_code.append(int(operands[2]))
        PC = PC + 1
    if(len(line_code)):
        temp.append(line_code)
    line_count = line_count + 1

print("Symbol Table")
for (k,v) in ST.items():
    print(k,end='\t')
    print(v.get('Address',''),end='\t')
    print(v.get('Value',''),end='\t')
    print('')
print('')

print("Literal Table")
for (k,v) in LT.items():
    print(k,end='\t')
    print(v['Address'],end='\t')
    print(v['Value'],end='\t')
    print('')
print('')

print('Pool Table')
for loc in PT:
    print(loc)
print('')

print('Table for Incomplete Instruction')
for (k,v) in TII.items():
    print(k,end='\t')
    print(v,end='\t')
    print('')
```

```
        print('')

        for i in temp:
            print(i)

        f.close()

except Exception as e:
    print('Error in Line',line_count,':',end=' ')
    print(e)
```



```
krhero@hellblazer:/mnt/0FB812900FB81290/BTech/Assignments/3rd_Year/SS/Assignment3$ python3 ./assembler.py ./test.asm
Symbol Table
A       107     3
B       502     10
D       504     8
LABEL   107
L1      500

Literal Table
='9'    105     9
='23'   106     23
='7'    505     7

Pool Table
2
1

Table for Incomplete Instruction
100     A
101     B
102     ='9'
103     D
104     ='23'
501     ='7'
```