System Software Practicals

Assignment 5

Krunal Rank
U18CO081

1.Generate Macro Definition Table(MDT) for given macro definition.

__main__.py:

```python
# Required libraries
import argparse


MNT = {}  # Macro Name Table
PNTAB = {}  # Parameter Name Table
KPDTAB = {}  # Keyword Parameter Table
EVNTAB = {}  # Expansion Variable Name Table
SSNTAB = {}  # Sequencing Symbol Name Table
MDT = {}  # Macro Definition Table
APTAB = {}  # Actual Parameter Table

# Indcies of Tables
SSNTAB_IDX  = 1
PNTAB_IDX = 1
EV_IDX = 1
SSTAB_IDX = 4
MDT_IDX = 12
KPDTAB_IDX = 7
APTAB_IDX = 1

mode = "DEFAULT"
parse_macro_declaration = 0
start_inserting_in_mdt = 0

macro = ''

if __name__ == "__main__":

   # Parsing FilePath as Arguments
   parser = argparse.ArgumentParser(description="Generates MDT For given Macro Code")
   parser.add_argument("file_path", metavar="filePath", help="File Path to Macro
Code")
   args = parser.parse_args()
   file_path = args.file_path
```

```python
# Parsing the File
# try:
with open(file_path, "r") as f:
    lines = f.readlines()  # lines = List of lines in file f
    line_count = 0
    for line in lines:
        line_count = line_count + 1

        decoded_line = line.replace('\n','').replace(' ','').split("-")
        label = decoded_line[0]
        operator = decoded_line[1]
        operands = decoded_line[2]
        if parse_macro_declaration:

            macro_name = operator
            parameters = operands.replace(' ','').replace(",", "").split("&")
            pp = 0
            kp = 0
            mdtp = MDT_IDX
            kpdtp = KPDTAB_IDX
            sstp = SSTAB_IDX

            for parameter in parameters:
                if len(parameter)==0:
                    continue
                if "=" in parameter:
                    kp = kp + 1
                    p_specs = parameter.split("=")
                    PNTAB[p_specs[0]] = {"IDX": PNTAB_IDX}
                    PNTAB_IDX = PNTAB_IDX + 1
                    KPDTAB[p_specs[0]] = {
                        "IDX": KPDTAB_IDX,
                        "DEFAULT_VAL": p_specs[1],
                    }
                    KPDTAB_IDX = KPDTAB_IDX + 1
                else:
                    pp = pp + 1
                    PNTAB[parameter] = {"IDX": PNTAB_IDX}
                    PNTAB_IDX = PNTAB_IDX + 1

            MNT[macro_name] = {
                "#PP": pp,
                "#KP": kp,
                "#EV": 0,
                "MDTP": mdtp,
                "KPDTP": kpdtp,
                "SSTP": sstp,
            }
            macro = macro_name


            start_inserting_in_mdt = 1
            parse_macro_declaration = 0
            continue


        if operator == "MACRO":
            if mode == "MACRO_DETECTED":
                raise Exception(
                    "Invalid Operator MACRO detected while parsing Macro
definition"
                )
            elif mode == "DEFAULT":
                mode = "MACRO_DETECTED"
                parse_macro_declaration = 1
        elif operator=="LCL":
            if start_inserting_in_mdt==0:
                raise Exception('Invalid Operator LCL found')

            evs = operands.replace(' ','').replace(',','').split('&')
            cnt = 0
            for ev in evs:
                if len(ev)==0:
                    continue
                cnt = cnt + 1
                EVNTAB[ev] = {"IDX":EV_IDX,"VALUE":-1}
                EV_IDX = EV_IDX + 1
            MNT[macro]["#EV"] = MNT[macro]["#EV"] + cnt
        elif operator=="SET":
            if start_inserting_in_mdt==0:
                raise Exception('Invalid Operator SET found')

            key = label.replace('&','').replace(' ','')
            if EVNTAB.get(key,-1)==-1:
                raise Exception('Expansion Variable not found!')
            EVNTAB[key]["VALUE"] = operands
        elif operator=="MEND":
            mode = 'DEFAULT'
```

```
                start_inserting_in_mdt = 0
            elif MNT.get(operator,-1)!=-1:
                if mode!='DEFAULT':
                    raise Exception('Macro Called inside Macro Definition')
                params = operands.replace(' ','').split(',')
                for param in params:
                    APTAB[param]={"IDX":APTAB_IDX}
                    APTAB_IDX = APTAB_IDX + 1
            elif operator!='MOVER' and operator!='MOVEM' and operator!='AIF':
                raise Exception('Invalid Operator')


            if start_inserting_in_mdt:
                if label.startswith('.') and SSNTAB.get(label,-1)==-1:
                    SSNTAB[label] = {"IDX":SSNTAB_IDX,"MDT_ENTRY":MDT_IDX}
                    SSNTAB_IDX = SSNTAB_IDX + 1
                for param in PNTAB.keys():
                    replacer = '&'+param
                    operands =
operands.replace(replacer,'(P,'+str(PNTAB[param]["IDX"])+')')
                    label = label.replace(replacer,'(P,'+str(PNTAB[param]["IDX"])+')')
                for param in EVNTAB.keys():
                    replacer = '&'+param
                    operands =
operands.replace(replacer,'(P,'+str(EVNTAB[param]["IDX"])+')')
                    label =
label.replace(replacer,'(P,'+str(EVNTAB[param]["IDX"])+')')
                for param in SSNTAB.keys():
                    operands =
operands.replace(param,'(P,'+str(SSNTAB[param]["IDX"])+')')
                    label = label.replace(param,'(P,'+str(SSNTAB[param]["IDX"])+')')

                MDT[MDT_IDX] = {"LABEL":label,"OPERATOR":operator,"OPERAND":operands}

                MDT_IDX = MDT_IDX + 1



    # except Exception as e:
    #     print("Error in Line", line_count, ":", end=" ")
    #     print(e)
    #     exit(0)
    print('PARAMETER NAME TABLE')
    print('IDX\t\tNAME\t\t')
```

```
for (key,val) in PNTAB.items():
    print(val['IDX'],end='\t\t')
    print(key,end='\n')

print('')
print('')
print('EXPANSION VARIABLE NAME TABLE')
print('IDX\t\tNAME')
for (key,val) in EVNTAB.items():
    print(val['IDX'],end='\t\t')
    print(key,end='\n')

print('')
print('')
print('KEYWORD PARAMETER DEFAULT TABLE')
print('IDX\t\tNAME\t\tDEFAULT_VAL')
for (key,val) in KPDTAB.items():
    print(val['IDX'],end='\t\t')
    print(key,end='\t\t')
    print(val['DEFAULT_VAL'],end='\n')

print('')
print('')
print('SEQUENCING SYMBOL NAME TABLE')
print('IDX\t\tNAME\t\tMDT_ENTRY')
for (key,val) in SSNTAB.items():
    print(val['IDX'],end='\t\t')
    print(key,end='\t\t')
    print(val['MDT_ENTRY'],end='\n')

print('')
print('')
print('ACTUAL PARAMETER TABLE')
print('IDX\t\tNAME\t\t')
for (key,val) in APTAB.items():
    print(val['IDX'],end='\t\t')
    print(key,end='\n')

print('')
print('')
print('MACRO NAME TABLE')
print('NAME\t\t#PP\t\t#KP\t\t#EV\t\tMDTP\t\tKPDTP\t\tSSTP')
for (key,val) in MNT.items():
    print(key,end='\t')
```

```
        print(val["#PP"],end='\t\t')
        print(val["#KP"],end='\t\t')
        print(val["#EV"],end='\t\t')
        print(val["MDTP"],end='\t\t')
        print(val["KPDTP"],end='\t\t')
        print(val["SSTP"],end='\n')

    print('')
    print('')
    print('MACRO DEFINITION TABLE')
    print('IDX\t\tLABEL\t\tOPERATOR\t\tOPERANDS')
    for (key,val) in MDT.items():
        print(key,end='\t\t')
        print(val['LABEL'],end='\t\t')
        print(val['OPERATOR'],end='\t\t')
        print(val['OPERAND'],end='\n')
```

Output:

```
PARAMETER NAME TABLE
IDX         NAME
1           X
2           N
3           REG


EXPANSION VARIABLE NAME TABLE
IDX         NAME
1           M


KEYWORD PARAMETER DEFAULT TABLE
IDX         NAME        DEFAULT_VAL
7           REG             AREG


SEQUENCING SYMBOL NAME TABLE
IDX         NAME        MDT_ENTRY
1           .MORE       15


ACTUAL PARAMETER TABLE
IDX         NAME
1           AREA
2           10


MACRO NAME TABLE
NAME        #PP         #KP         #EV         MDTP        KPDTP       SSTP
CLEARMEM    2           1           1           12          7           4


MACRO DEFINITION TABLE
IDX         LABEL       OPERATOR         OPERANDS
12                      LCL              (P,1)
13          (P,1)       SET              0
14                      MOVER            (P,3),='0'
15          (S,1)       MOVEM            (P,3),(P,1)+(P,1)
16          (P,1)       SET               (P,1)+1
17                      AIF               ((P,1)NE(P,2))(S,1)
```