Name : Devesh Mali

Class : B – B2

Roll no:13228

// Macro Pass1 code :

```
import re


def main():
    # Open files
    with open("macro_input.asm", "r") as br, \
         open("mnt.txt", "w") as mnt, \
         open("mdt.txt", "w") as mdt, \
         open("kpdt.txt", "w") as kpdt, \
         open("pntab.txt", "w") as pnt, \
         open("intermediate.txt", "w") as ir:

        pntab = {}
        line = None
        Macroname = None
        mdtp = 1
        kpdtp = 0
        paramNo = 1
        pp = 0
        kp = 0
        flag = 0

        for line in br:
            line = line.strip()
            parts = re.split(r'\s+', line)

            if parts[0].upper() == "MACRO":
                flag = 1
                line = next(br).strip()
                parts = re.split(r'\s+', line)
```

```python
    Macroname = parts[0]

    if len(parts) <= 1:
        mnt.write(f"{parts[0]}\t{pp}\t{kp}\t{mdtp}\t{kp if kp == 0 else kpdtp + 1}\n")
        continue

    for i in range(1, len(parts)):  # Processing parameters
        parts[i] = re.sub(r'[&,]', '', parts[i])
        if '=' in parts[i]:
            kp += 1
            keywordParam = parts[i].split('=')
            pntab[keywordParam[0]] = paramNo
            paramNo += 1
            kpdt.write(f"{keywordParam[0]}\t{keywordParam[1] if len(keywordParam) == 2 else '-'}\n")
        else:
            pntab[parts[i]] = paramNo
            paramNo += 1
            pp += 1

    mnt.write(f"{parts[0]}\t{pp}\t{kp}\t{mdtp}\t{kp if kp == 0 else kpdtp + 1}\n")
    kpdtp += kp

elif parts[0].upper() == "MEND":
    mdt.write(line + "\n")
    flag = kp = pp = 0
    mdtp += 1
    paramNo = 1
    pnt.write(Macroname + ":\t")
    for key in pntab:
        pnt.write(f"{key}\t")
    pnt.write("\n")
    pntab.clear()

elif flag == 1:
```

```
            for i in range(len(parts)):
                if '&' in parts[i]:
                    parts[i] = re.sub(r'[&,]', '', parts[i])
                    mdt.write(f"(P,{pntab[parts[i]]})\t")
                else:
                    mdt.write(parts[i] + "\t")
            mdt.write("\n")
            mdtp += 1

        else:
            ir.write(line + "\n")

    print("Macro Pass 1 Processing done. :)")

if __name__ == "__main__":
    main()

// Macro Input :
START   100
+MOVER          AREG    10
+ADD    AREG    ='1'
+MOVER          CREG    20
+ADD    CREG    ='5'
+MOVER          BREG    100
+MOVER          AREG    200
+ADD    BREG    ='15'
+ADD    AREG    ='10'
END

// Intermediate code:
START   100
M1      10, 20
M2      100, 200, &V=AREG, &U=BREG
END
```

```
//MDT :
MOVER (P,3)     (P,1)
ADD   (P,3)     ='1'
MOVER (P,4)     (P,2)
ADD   (P,4)     ='5'
MEND
MOVER (P,3)     (P,1)
MOVER (P,4)     (P,2)
ADD   (P,3)     ='15'
ADD   (P,4)     ='10'
MEND

//MNT:
M1    2    2    1    1
M2    2    2    6    3

//PNTAB:
M1:   X    Y    A    B
M2:   P    Q    U    V

//KPTAB :
A     AREG
B     CREG
U     CREG
V     DREG
```

```
// Macro Pass 2 code:
import re


class MNTEntry:
    def __init__(self, name, pp, kp, mdtp, kpdtp):
        self.name = name
        self.pp = pp
        self.kp = kp
        self.mdtp = mdtp
        self.kpdtp = kpdtp

def main():
    # Open files
    with open("intermediate.txt", "r") as irb, \
         open("mdt.txt", "r") as mdtb, \
         open("kpdt.txt", "r") as kpdtb, \
         open("mnt.txt", "r") as mntb, \
         open("pass2.txt", "w") as fr:

        mnt = {}
        aptab = {}
        aptab_inverse = {}

        mdt = []
        kpdt = []

        # Reading MDT file
        mdt = [line.strip() for line in mdtb]

        # Reading KPDT file
        kpdt = [line.strip() for line in kpdtb]

        # Reading MNT file
        for line in mntb:
            parts = line.split()
            mnt[parts[0]] = MNTEntry(parts[0], int(parts[1]), int(parts[2]), int(parts[3]), int(parts[4]))

        # Reading Intermediate file and processing
        for line in irb:
            line = line.strip()
            parts = re.split(r'\s+', line)
```

```
if parts[0] in mnt:
    entry = mnt[parts[0]]
    pp = entry.pp
    kp = entry.kp
    kpdtp = entry.kpdtp
    mdtp = entry.mdtp
    param_no = 1

    # Processing positional parameters
    for i in range(1, pp + 1):
        if param_no < len(parts):
            parts[param_no] = parts[param_no].replace(",", "")
            aptab[param_no] = parts[param_no]
            aptab_inverse[parts[param_no]] = param_no
            param_no += 1
        else:
            print(f"Warning: Positional parameter {param_no} missing in intermediate line")

    # Processing keyword parameters
    j = kpdtp - 1
    for i in range(kp):
        if j < len(kpdt):
            temp = kpdt[j].split("\t")
            if len(temp) == 2:
                aptab[param_no] = temp[1]
                aptab_inverse[temp[0]] = param_no
                j += 1
                param_no += 1
            else:
                print(f"Warning: Keyword parameter format incorrect at index {j}")
        else:
            print(f"Warning: No more keyword parameters available in kpdt at index {j}")

    # Replacing parameters in the intermediate code
```

```
            for i in range(pp + 1, len(parts)):
                parts[i] = parts[i].replace(",", "")
                splits = parts[i].split("=")
                if len(splits) == 2:
                    name = re.sub(r'&', '', splits[0])
                    if name in aptab_inverse:
                        aptab[aptab_inverse[name]] = splits[1]
                    else:
                        print(f"Warning: Parameter name '{name}' not found in aptab_inverse")
                else:
                    print(f"Warning: Incorrect parameter replacement format in {parts[i]}")

            # Writing to the output file
            i = mdtp - 1
            while i < len(mdt) and not mdt[i].upper().startswith("MEND"):
                splits = re.split(r'\s+', mdt[i])
                fr.write("+")
                for k in range(len(splits)):
                    if "(P," in splits[k]:
                        splits[k] = re.sub(r'[^\d]', '', splits[k])  # Extract number
                        value = aptab.get(int(splits[k]), "UNKNOWN")
                        fr.write(f"{value}\t")
                    else:
                        fr.write(f"{splits[k]}\t")
                fr.write("\n")
                i += 1

            aptab.clear()
            aptab_inverse.clear()
        else:
            fr.write(line + "\n")

print("Macro Pass 2 Processing done. :)")
```

```
if __name__ == "__main__":
    main()


//Pass 2 output:
START   100
+MOVER          AREG    10
+ADD    AREG    ='1'
+MOVER          CREG    20
+ADD    CREG    ='5'
+MOVER          BREG    100
+MOVER          AREG    200
+ADD    BREG    ='15'
+ADD    AREG    ='10'
END
```