Name : Devesh Mali

Class : B – B2

Roll no :13228

//Pass 1 code:

import os

def get\_op(opcode):

    optab = {

        "STOP": ("IS", "00"),

        "ADD": ("IS", "01"),

        "SUB": ("IS", "02"),

        "MULT": ("IS", "03"),

        "MOVER": ("IS", "04"),

        "MOVEM": ("IS", "05"),

        "COMP": ("IS", "06"),

        "BC": ("IS", "07"),

        "DIV": ("IS", "08"),

        "READ": ("IS", "09"),

        "PRINT": ("IS", "10"),

        "START": ("AD", "01"),

        "END": ("AD", "02"),

        "ORIGIN": ("AD", "03"),

        "EQU": ("AD", "04"),

        "LTORG": ("AD", "05"),

        "DC": ("DL", "01"),

        "DS": ("DL", "02")

    }

    return optab.get(opcode, None)

def get\_reg\_id(reg):

    regtab = {

        "AREG": 1,

        "BREG": 2,

        "CREG": 3,

        "DREG": 4

    }

    return regtab.get(reg, -1)

def get\_condition\_code(cond):

    condtab = {

        "LT": 1,

        "LE": 2,

        "EQ": 3,

        "GT": 4,

        "GE": 5,

        "ANY": 6

    }

    return condtab.get(cond, -1)

def present\_st(sym, symtab):

    return any(sym == entry[1] for entry in symtab)

def get\_sym\_id(sym, symtab):

    for i, entry in enumerate(symtab):

        if sym == entry[1]:

            return i

    return -1

def get\_sym\_address(sym, symtab):

    for entry in symtab:

        if sym == entry[1]:

            return entry[2]

    return None

def present\_lt(lit, littab):

    return any(lit == entry[1] for entry in littab)

def get\_lit\_id(lit, littab):

    for i, entry in enumerate(littab):

        if lit == entry[1]:

            return i

    return -1

def handle\_literal\_declaration(littab, pooltab, lc):

    start\_index = len(pooltab) + 1

    for i in range(len(littab)):

        if littab[i][2] == -1:

            littab[i] = (littab[i][0], littab[i][1], lc)

            lc += 1

    pooltab.append(start\_index)

    return lc

def resolve\_expression(expr, symtab):

    try:

        # Direct numeric values

        return int(expr)

    except ValueError:

        # Resolve symbolic expressions

        parts = expr.split('+')

        base = parts[0].strip()

        offset = int(parts[1].strip()) if len(parts) > 1 else 0

        base\_address = get\_sym\_address(base, symtab)

        if base\_address is not None:

            return base\_address + offset

        else:

            raise ValueError(f"Symbol {base} not found in symbol table")

def main():

    project\_path = r"C:\Users\Harshad\Desktop\pass"

    asm\_path = os.path.join(project\_path, "input.asm")

    ic\_path = os.path.join(project\_path, "ic.txt")

    st\_path = os.path.join(project\_path, "symtable.txt")

    lt\_path = os.path.join(project\_path, "littable.txt")

    pt\_path = os.path.join(project\_path, "pooltable.txt")

    LC = 0

    symtab = []

    littab = []

    pooltab = []

    scnt = 0

    lcnt = 0

    with open(asm\_path, 'r') as asm, open(ic\_path, 'w') as ic, open(st\_path, 'w') as st, open(lt\_path, 'w') as lt, open(pt\_path, 'w') as pt:

        for line in asm:

            tokens = line.split()

            if len(tokens) == 2:

                label, opcode = tokens[0], tokens[1]

                op1, op2 = "NAN", "NAN"

            elif len(tokens) == 3:

                label, opcode, op1 = tokens[0], tokens[1], tokens[2]

                op2 = "NAN"

            elif len(tokens) == 4:

                label, opcode, op1, op2 = tokens[0], tokens[1], tokens[2], tokens[3]

            else:

                continue

            op = get\_op(opcode)

            if op is None:

                continue

            if label != "NAN":

                if present\_st(label, symtab):

                    symtab[get\_sym\_id(label, symtab)] = (get\_sym\_id(label, symtab) + 1, label, LC)

                else:

                    symtab.append((scnt + 1, label, LC))

                    scnt += 1

            if opcode == "START":

                LC = int(op1)

                ic.write(f"---\t({op[0]},{op[1]}) (C,{op1}) NAN\n")

            elif opcode == "END":

                ic.write(f"---\t({op[0]},{op[1]}) NAN NAN\n")

                LC = handle\_literal\_declaration(littab, pooltab, LC)

                break

            elif opcode == "LTORG":

                ic.write(f"---\t({op[0]},{op[1]}) NAN NAN\n")

                LC = handle\_literal\_declaration(littab, pooltab, LC)

            elif opcode == "ORIGIN":

                LC = resolve\_expression(op1, symtab)

            else:

                lc = LC

                LC += 1

                if opcode in ["DS", "DC"]:

                    if opcode == "DS":

                        ic.write(f"{lc}\t({op[0]},{op[1]}) (C,{op1}) NAN\n")

                        LC += int(op1) - 1

                    else:

                        ic.write(f"{lc}\t({op[0]},{op[1]}) (C,{op1[1:-1]}) NAN\n")

                else:

                    op1\_code = get\_reg\_id(op1) if opcode != "BC" else get\_condition\_code(op1)

                    if op2.startswith("="):

                        if not present\_lt(op2, littab):

                            littab.append((lcnt + 1, op2, -1))

                            lcnt += 1

                        op2\_code = f"(L,{get\_lit\_id(op2, littab) + 1})"

                    else:

                        if present\_st(op2, symtab):

                            op2\_code = f"(S,{get\_sym\_id(op2, symtab) + 1})"

                        else:

                            symtab.append((scnt + 1, op2, -1))

                            scnt += 1

                            op2\_code = f"(S,{scnt})"

                    ic.write(f"{lc}\t({op[0]},{op[1]}) ({op1\_code}) {op2\_code}\n")

        for entry in symtab:

            st.write(f"{entry[0]}\t{entry[1]}\t{entry[2]}\n")

        for entry in littab:

            lt.write(f"{entry[0]}\t{entry[1]}\t{entry[2]}\n")

        for entry in pooltab:

            pt.write(f"#{entry}\n")  # Write pool table with correct index

if \_\_name\_\_ == "\_\_main\_\_":

    main()

//Input.asm :

NAN START 200 NAN

NAN MOVER AREG ='5'

NAN MOVEM AREG A

LOOP MOVER AREG A

NAN MOVER CREG B

NAN ADD CREG ='1'

NAN MOVER AREG A

NAN MOVER CREG B

NAN MOVER AREG A

NAN MOVER CREG B

NAN MOVER AREG A

NAN BC ANY NEXT

NAN LTORG NAN NAN

NAN MOVER AREG A

NEXT SUB AREG ='1'

NAN BC LT BACK

LAST STOP NAN NAN

NAN ORIGIN LOOP+2 NAN

NAN MULT CREG B

NAN ORIGIN LAST+1 NAN

A DS 1 NAN

BACK EQU LOOP NAN

B DS 1 NAN

NAN END NAN NAN

//Intermediate code :

--- (AD,01) (C,200) NAN

200 (IS,04) (1) (L,1)

201 (IS,05) (1) (S,1)

202 (IS,04) (1) (S,1)

203 (IS,04) (3) (S,3)

204 (IS,01) (3) (L,2)

205 (IS,04) (1) (S,1)

206 (IS,04) (3) (S,3)

207 (IS,04) (1) (S,1)

208 (IS,04) (3) (S,3)

209 (IS,04) (1) (S,1)

210 (IS,07) (6) (S,4)

--- (AD,05) NAN NAN

213 (IS,04) (1) (S,1)

214 (IS,02) (1) (L,2)

215 (IS,07) (1) (S,5)

216 (IS,00) (-1) (S,7)

204 (IS,03) (3) (S,3)

217 (DL,02) (C,1) NAN

218 (AD,04) (-1) (S,7)

219 (DL,02) (C,1) NAN

--- (AD,02) NAN NAN

//Symbol table :

1 A 217

2 LOOP 202

3 B 219

4 NEXT 214

5 BACK 218

6 LAST 216

7 NAN -1

//Literal table :

1 ='5' 211

2 ='1' 212

//Pool table :

#1

#2

//Pass 2 code:

import os

def table\_lookup(table\_file, num):

    with open(table\_file, 'r') as table:

        for line in table:

            no, name, addr = line.split()

            if no == num:

                return addr

    return "NAN"

def main():

    project\_path = r"C:\Users\Harshad\Desktop\pass"

    ic\_path = os.path.join(project\_path, "ic.txt")

    st\_path = os.path.join(project\_path, "symtable.txt")

    lt\_path = os.path.join(project\_path, "littable.txt")

    mc\_path = os.path.join(project\_path, "machine\_code.txt")

    with open(ic\_path, 'r') as ic, open(mc\_path, 'w') as mc:

        print("\n -- ASSEMBLER PASS-2 OUTPUT --\n")

        print("LC\t<INTERMEDIATE CODE>\t\t\tLC\t<MACHINE CODE>\n")

        for line in ic:

            lc, ic1, ic2, ic3 = line.split()

            mc\_line = ""

            if ic1.startswith("(AD") or (ic1.startswith("(DL") and ic1.endswith("02)")):

                mc\_line = "-No Machine Code-"

            elif ic1.startswith("(DL,01)"):

                mc\_line = f"00\t0\t00{ic2[3]}"

            else:

                if ic1 == "(IS,00)":

                    mc\_line = f"{ic1[4:6]}\t0\t000"

                elif ic2.startswith("(S"):

                    addr = table\_lookup(st\_path, ic2[4])

                    mc\_line = f"{ic1[4:6]}\t0\t{addr}"

                else:

                    if ic3.startswith("(S"):

                        addr = table\_lookup(st\_path, ic3[4])

                    else:

                        addr = table\_lookup(lt\_path, ic3[4])

                    mc\_line = f"{ic1[4:6]}\t{ic2[1]}\t{addr}"

            print(f"{lc}\t{ic1}\t{ic2}\t{ic3}\t\t\t{lc}\t{mc\_line}\n")

            mc.write(f"{lc}\t{mc\_line}\n")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

//Output

Machine code:

--- -No Machine Code-

200 04 1 NAN

201 05 1 NAN

202 04 1 NAN

203 04 3 NAN

204 01 3 NAN

205 04 1 NAN

206 04 3 NAN

207 04 1 NAN

208 04 3 NAN

209 04 1 NAN

210 07 6 NAN

--- -No Machine Code-

213 04 1 NAN

214 02 1 NAN

215 07 1 NAN

216 00 0 000

204 03 3 NAN

217 -No Machine Code-

218 -No Machine Code-

219 -No Machine Code-

--- -No Machine Code-