# Experiment no.1

## Theory:

*TwoPassAssembler: The two-pass assembler reads the entire source program, searching only for label definitions. All labels are gathered, assigned an address, and placed in the symbol table in this pass; no instructions are assembled, and at the end the symbol table should contain all the labels defined in the program. To assign addresses to labels, the assembler maintains a Location Counter (LC). In the second pass, the instructions are once again read and assembled using the symbol table. Essentially, the assembler goes through the program one line at a time and generates machine code for that instruction. Then it moves on to the next instruction, creating the entire machine code program. For the majority of instructions, this process is fine; for example, for instructions that only reference registers, the assembler can compute the machine code.*

*OnePass and TwoPass Assemblers: In the first pass, it looks for label definitions and introduces them in the symbol table (a dynamic table that includes the name and address for each label in the source program). In the second pass, after the symbol table is complete, it does the actual assembly by tracing the labels. The dif icult part is to solve future label references (the problem of forward referencing) and assembly code in one pass. The one pass assembler prepares an intermediate file, which is used as input by the two pass assembler. The two pass assemblers do two passes over the source file (the second pass may be over an intermediate file generated in the first pass of the assembler).*

**Code :**

```
Created on Sat Feb 3 10:47:42 2024

@author: ADMIN
"""


fp = open('program3.txt','r')
program = fp.read().split("\n")
# print(program)
fp.close()

mnemonic_tab = {'STOP':'00','ADD':'01', 'SUB':'02', 'MULT':'03', 'MOVER':'04', 'MOVEM':'05',
'COMP':'06', 'BC':'07', 'DIV':'08', 'READ':'09',
'PRINT':'10','DC':'01','DS':'02','START':'01','END':'02','ORIGIN':'03','EQU':'04','LTORG':'05'}
reg_code = {'AREG':1, 'BREG':2, 'CREG':3, 'DREG':4}
condition_code = {'LT':1,'LE':2,'EQ':3,'GT':4,'GE':5,'ANY':6}
optab =
{'START':'AD','READ':'IS','MOVER':'IS','MOVEM':'IS','MULT':'IS','ADD':'IS','COMP':'IS','BC':'IS','PR
I NT':'IS','STOP':'AD','DS':'DL','DC':'DL','END':'AD','DIV':'IS'}
sym_table = {}

print('Content of Mnemonic table is-\n')
print('Mnemonic ', 'Code\n')
for k,v in mnemonic_tab.items():
    print('{:<8}{:>8} '.format(k,v))

print()
print('Content of Opcode table is-\n')
```

```
print('Mnemonic ', 'Class\n')
for k,v in optab.items():
    print('{:<8}{:>8}'.format(k,v))

print()
print('****Input Assembly source code*****')
print()

#print the source code
# set the value of lc

for line in program:
    a = line.split()
    if a[0]=='START':
        lc=int(a[1])
        temp = lc
    print(line)

# Build the symbol table
for line in program:
    l = line.split()
    for i in l:
        if i not in optab and i not in reg_code and i.isdigit()!=True and i not in condition_code:
            sym_table[i]=lc
            lc+=1
print()
print('Content of Symbol table is-')
print('Symbol Name ','Address')
for k,v in sym_table.items():
    print(' {0}  {1}'.format(k,v))


lc = temp
print()
print('*******Itermediate Code After PASS-I**********')
print()
a= list(sym_table.keys())
for line in program:
    lexeme = line.split()
    if(len(lexeme)==4): #if label is there,remove it exam. AGAIN MULT AREG FIVE
        lexeme.remove(lexeme[0])
    if lexeme[0] in optab:
        if optab[lexeme[0]]=='AD':
            if(len(lexeme)==1):
                print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]))
                lc+=1
            else:
                if(lexeme[0]=='START'):
```

```
                print(' ',(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),'(C,',lexeme[1],')')

    if lexeme[0] in optab:
        if optab[lexeme[0]]=='IS':
            if len(lexeme)==3:
                #a= list(sym_table.keys())
                if lexeme[0]=='BC':

print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),condition_code[lexeme[1]],'(S',a.index(lexeme[2]),
')')
                    lc+=1
                else:

print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),reg_code[lexeme[1]],'(S',a.index(lexeme[2]),')')
                    lc+=1
            if(len(lexeme)==2):
                print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),'(S',a.index(lexeme[1]),')')
                lc+=1

    if lexeme[0] not in optab:
        if len(lexeme)==3:
            print(lc,(optab[lexeme[1]],mnemonic_tab[lexeme[1]]),'(C',lexeme[2],')')
            lc+=1

        if len(lexeme)==4:
            print(lc,(optab[lexeme[1]],mnemonic_tab[lexeme[1]]),)
            lc+=1

print()
print('********Machine Code After Pass II******\n')

lc = temp

for line in program:
    lexeme = line.split()
    if len(lexeme)==4:
        lexeme.remove(lexeme[0])

    if lexeme[0] in optab:
        if optab[lexeme[0]]=='AD':
            if(len(lexeme)==1):
                print()
                lc+=1
            else:
                if(lexeme[0]=='START'):pass
                    #print((optab[lexeme[0]],mnemonic_tab[lexeme[0]]),'(C,',lexeme[1],')')

    if lexeme[0] in optab:
```

```
        if optab[lexeme[0]]=='IS':
            if len(lexeme)==3:
                #a= list(sym_table.keys())
                if lexeme[0]=='BC':
                    print(lc,mnemonic_tab[lexeme[0]],condition_code[lexeme[1]],sym_table[lexeme[2]])
                    lc+=1
                else:
                    print(lc,mnemonic_tab[lexeme[0]],reg_code[lexeme[1]],sym_table[lexeme[2]])
                    lc+=1
            if(len(lexeme)==2):
                print(lc,mnemonic_tab[lexeme[0]],sym_table[lexeme[1]])
                lc+=1

    if lexeme[0] not in optab:
        if len(lexeme)==3:
            print(lc,mnemonic_tab[lexeme[1]],lexeme[2])
            lc+=1

        if len(lexeme)==4:
            print(mnemonic_tab[lexeme[1]])
            lc+=1
```

**Input 1 :**

```
START 200
MOVER AREG FIRST
ADD AREG SECOND
MOVEM AREG RESULT
PRINT RESULT
RESULT DS 1
FIRST DC 5
SECOND DC 7
END
```

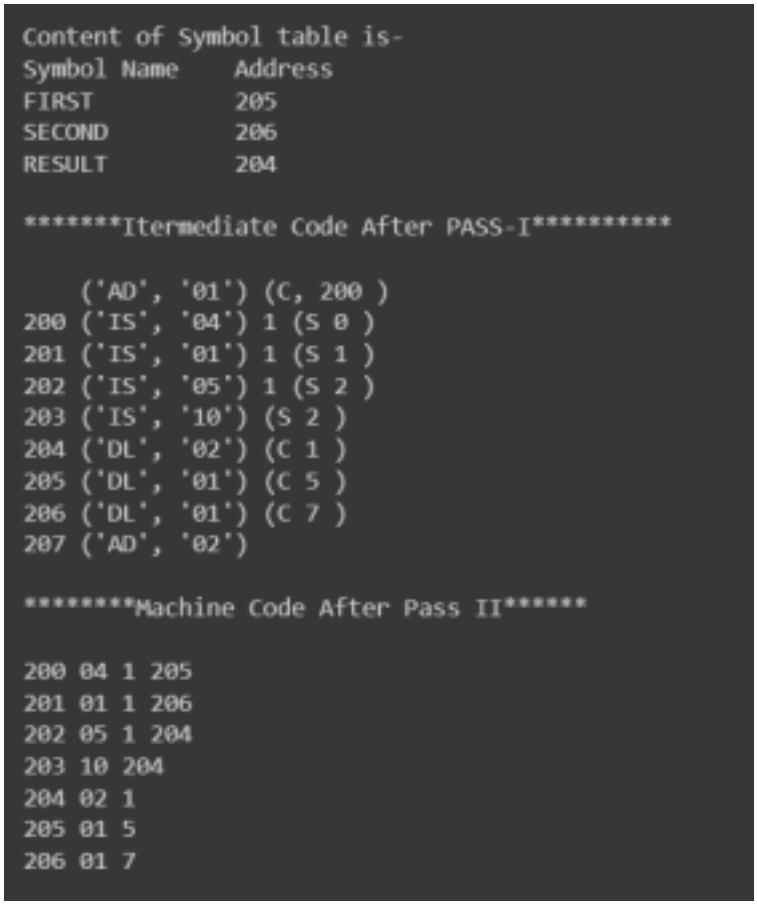
**OUTPUT 1 :**

Content of Mnemonic table is-

| Mnemonic | Code |
|---|---|
| STOP | 00 |
| ADD | 01 |
| SUB | 02 |
| MULT | 03 |
| MOVER | 04 |
| MOVEM | 05 |
| COMP | 06 |
| BC | 07 |
| DIV | 08 |
| READ | 09 |
| PRINT | 10 |
| DC | 01 |
| DS | 02 |
| START | 01 |
| END | 02 |
| ORIGIN | 03 |
| EQU | 04 |
| LTORG | 05 |

Content of Opcode table is-

| Mnemonic | Class |
|---|---|
| START | AD |
| READ | IS |
| MOVER | IS |
| MOVEM | IS |
| MULT | IS |
| ADD | IS |
| COMP | IS |
| BC | IS |
| PRINT | IS |
| STOP | AD |
| DS | DL |
| DC | DL |
| END | AD |
| DIV | IS |




**Code :**

```
Created on Sat Feb 3 10:47:42 2024

@author: ADMIN
"""



fp = open('program3.txt','r')
program = fp.read().split("\n")
# print(program)
fp.close()

mnemonic_tab = {'STOP':'00','ADD':'01', 'SUB':'02', 'MULT':'03', 'MOVER':'04', 'MOVEM':'05',
'COMP':'06', 'BC':'07', 'DIV':'08', 'READ':'09',
'PRINT':'10','DC':'01','DS':'02','START':'01','END':'02','ORIGIN':'03','EQU':'04','LTORG':'05'}
reg_code = {'AREG':1, 'BREG':2, 'CREG':3, 'DREG':4}
condition_code = {'LT':1,'LE':2,'EQ':3,'GT':4,'GE':5,'ANY':6}
optab =
 {'START':'AD','READ':'IS','MOVER':'IS','MOVEM':'IS','MULT':'IS','ADD':'IS','COMP':'IS','BC':'IS','PR
I NT':'IS','STOP':'AD','DS':'DL','DC':'DL','END':'AD','DIV':'IS'}
 sym_table = {}

print('Content of Mnemonic table is-\n')
print('Mnemonic ', 'Code\n')
 for k,v in mnemonic_tab.items():
```

```
    print('{:<8}{:>8} '.format(k,v))

print()
print('Content of Opcode table is-\n')
print('Mnemonic ', 'Class\n')
for k,v in optab.items():
    print('{:<8}{:>8}'.format(k,v))

print()
print('****Input Assembly source code*****')
print()

#print the source code
# set the value of lc

for line in program:
    a = line.split()
    if a[0]=='START':
        lc=int(a[1])
        temp = lc
    print(line)

# Build the symbol table
for line in program:
    l = line.split()
    for i in l:
        if i not in optab and i not in reg_code and i.isdigit()!=True and i not in condition_code:
            sym_table[i]=lc
            lc+=1
print()
print('Content of Symbol table is-')
print('Symbol Name ','Address')
for k,v in sym_table.items():
    print(' {0} {1}'.format(k,v))


lc = temp
print()
print('*******Itermediate Code After PASS-I**********')
print()
a= list(sym_table.keys())
for line in program:
    lexeme = line.split()
    if(len(lexeme)==4): #if label is there,remove it exam. AGAIN MULT AREG FIVE
        lexeme.remove(lexeme[0])
    if lexeme[0] in optab:
        if optab[lexeme[0]]=='AD':
            if(len(lexeme)==1):
```

```
            print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]))
            lc+=1
        else:
            if(lexeme[0]=='START'):
                print(' ',(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),'(C,',lexeme[1],')')

    if lexeme[0] in optab:
        if optab[lexeme[0]]=='IS':
            if len(lexeme)==3:
                #a= list(sym_table.keys())
                if lexeme[0]=='BC':

print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),condition_code[lexeme[1]],'(S',a.index(lexeme[2]),
')')
                    lc+=1
                else:

print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),reg_code[lexeme[1]],'(S',a.index(lexeme[2]),')')
                    lc+=1
            if(len(lexeme)==2):
                print(lc,(optab[lexeme[0]],mnemonic_tab[lexeme[0]]),'(S',a.index(lexeme[1]),')')
                lc+=1

    if lexeme[0] not in optab:
        if len(lexeme)==3:
            print(lc,(optab[lexeme[1]],mnemonic_tab[lexeme[1]]),'(C',lexeme[2],')')
            lc+=1

        if len(lexeme)==4:
            print(lc,(optab[lexeme[1]],mnemonic_tab[lexeme[1]]),)
            lc+=1

print()
print('********Machine Code After Pass II******\n')

lc = temp

 for line in program:
    lexeme = line.split()
    if len(lexeme)==4:
        lexeme.remove(lexeme[0])

    if lexeme[0] in optab:
        if optab[lexeme[0]]=='AD':
            if(len(lexeme)==1):
                print()
                lc+=1
            else:
```

```
            if(lexeme[0]=='START'):pass
                #print((optab[lexeme[0]],mnemonic_tab[lexeme[0]]),'(C,',lexeme[1],')')

    if lexeme[0] in optab:
        if optab[lexeme[0]]=='IS':
            if len(lexeme)==3:
                #a= list(sym_table.keys())
                if lexeme[0]=='BC':
                    print(lc,mnemonic_tab[lexeme[0]],condition_code[lexeme[1]],sym_table[lexeme[2]])
                    lc+=1
                else:
                    print(lc,mnemonic_tab[lexeme[0]],reg_code[lexeme[1]],sym_table[lexeme[2]])
                    lc+=1
            if(len(lexeme)==2):
                print(lc,mnemonic_tab[lexeme[0]],sym_table[lexeme[1]])
                lc+=1

    if lexeme[0] not in optab:
        if len(lexeme)==3:
            print(lc,mnemonic_tab[lexeme[1]],lexeme[2])
            lc+=1

        if len(lexeme)==4:
            print(mnemonic_tab[lexeme[1]])
            lc+=1
```

**Input 2 :**

```
START 101
READ N
MOVER BREG ONE
MOVEM BREG TERM
AGAIN MULT BREG TERM
MOVER CREG TERM
ADD CREG TERM
COMP CREG N
BC LE AGAIN
DIV BREG TWO
MOVEM BREG RESULT
PRINT RESULT
STOP
N DS 1
RESULT DS 1
ONE DC 1
TERM DS 1
TWO DC 2
END
```

**OUTPUT :**

```
Content of Mnemonic table is-

Mnemonic    Code

STOP          00
ADD           01
SUB           02
MULT          03
MOVER         04
MOVEM         05
COMP          06
BC            07
DIV           08
READ          09
PRINT         10
DC            01
DS            02
START         01
END           02
ORIGIN        03
EQU           04
LTORG         05
Content of Opcode table is-
```

| Mnemonic | Class |
|---|---|
| START | AD |
| READ | IS |
| MOVER | IS |
| MOVEM | IS |
| MULT | IS |
| ADD | IS |
| COMP | IS |
| BC | IS |
| PRINT | IS |
| STOP | AD |
| DS | DL |
| DC | DL |
| END | AD |
| DIV | IS |

```
*******Itermediate Code After PASS-I**********

    ('AD', '01') (C, 101 )
101 ('IS', '09') (S 0 )
102 ('IS', '04') 2 (S 1 )
103 ('IS', '05') 2 (S 2 )
104 ('IS', '03') 2 (S 2 )
105 ('IS', '04') 3 (S 2 )
106 ('IS', '01') 3 (S 2 )
107 ('IS', '06') 3 (S 0 )
108 ('IS', '07') 2 (S 3 )
109 ('IS', '08') 2 (S 4 )
110 ('IS', '05') 2 (S 5 )
111 ('IS', '10') (S 5 )
112 ('AD', '00')
113 ('DL', '02') (C 1 )
114 ('DL', '02') (C 1 )
115 ('DL', '01') (C 1 )
116 ('DL', '02') (C 1 )
117 ('DL', '01') (C 2 )
118 ('AD', '02')
```

```
********Machine Code After Pass II******

101 09 113
102 04 2 115
103 05 2 116
104 03 2 116
105 04 3 116
106 01 3 116
107 06 3 113
108 07 2 109
109 08 2 117
110 05 2 114
111 10 114

113 02 1
114 02 1
115 01 1
116 02 1
117 01 2
```

**Conclusion :** *Hence,we studied and implemented two pass assemblers.*