Experiment No. 7

Aim: To implement two pass assembler.(Data Structures)

Theory:

Simplified Instructional Computer (SIC):

Simplified Instructional Computer (SIC) is a hypothetical computer that has hardware features which are often found in real machines. There are two versions of this machine:

1. SIC standard Model
2. SIC/XE(extra equipment or expensive)

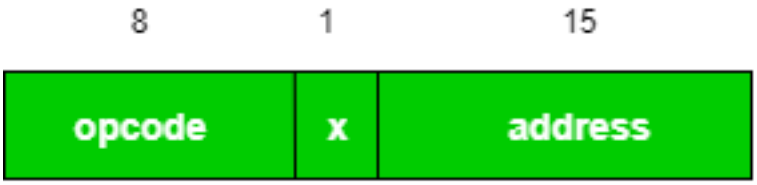
Object program for SIC can be properly executed on SIX/XE which is known as upward compatability.

SIC Machine Architecture/Components:

1. **Memory :**
    a. Memory is byte addressable that is words are addressed by location of their lowest numbered byte.
    b. There are 2^15 bytes in computer memory (1 byte = 8 bits) 3 consecutive byte = 1 word (24 bits = 1 word)
2. **Registers:**
    There are 5 registers in SIC. Every register has an address associated with it known as register number. Size of each register is 4 bytes. On basis of register size, integer size is dependent.
    I. A(Accumulator-0): It is used for mathematical operations.
    II. X(Index Register-1): It is used for addressing.
    III. L(Linkage Register-2): It stores the return address of instruction in case of subroutines.
    IV. PC(Program Counter-8): It holds the address of next instruction to be executed.
    V. SW(Status Word-9): It contains the variety of information
3. **Data Format :**
    a. Integers are represented by 24 bit.
    b. Negative numbers are represented in 2's complement.
    c. Characters are represented by 8 bit ASCII value.
    d. No floating point representation is available.

4. **Instruction Format:**

All instructions in SIC have 24 bit format.



- If x=0 it means direct addressing mode.
- If x=1 it means indexed addressing mode.

5. **Instruction Set :**
   - Load And Store Instructions: To move or store data from accumulator to memory or vice-versa. For example LDA, STA, LDX, STX etc.
   - Comparison Instructions: Used to compare data in memory by contents in accumulator. For example COMP data.
   - Arithmetic Instructions: Used to perform operations on accumulator and memory and store result in accumulator. For example ADD, SUB, MUL, DIV etc.
   - Conditional Jump: compare the contents of accumulator and memory and performs task based on conditions. For example JLT, JEQ, JGT
   - Subroutine Linkage: Instructions related to subroutines. For example JSUB, RSUB
6. **Input and Output :**
   It is performed by transferring 1 byte at a time from or to rightmost 8 bits of accumulator.
   Each device has 8 bit unique code.
   There are 3 I/O instructions:
   - Test Device (TD) tests whether device is ready or not. Condition code in Status Word Register is used for this purpose. If cc is < then device is ready otherwise device is busy.
   - Read data(RD) reads a byte from device and stores in register A.
   - Write data(WD) writes a byte from register A to the device.

Example:

```
LDA FIVE
STA ALPHA
LDCH CHARZ
STCH C1

ALPHA RESW 1
FIVE WORD 5
CHARZ BYTE C'Z'
C1 RESB 1
```

Mnemonic Table:

Mnemonic	Opcode	Operand1
LDA	00	FIVE
STA	0C	ALPHA
LDCH	50	CHARZ
STCH	54	C1

File:

```
LDA #5
STA ALPHA
LDCH #90
STCH C1
```

Code:

```
OPCODETAB =
{'ADD':[3,'18',1],'ADDF':[3,'58',1],'ADDR':[2,'90',2],'AND':[3,'40',1],'CLEAR':[2
,'B4',1],'COMP':[3,'28',1],'COMPF':[3,'88',1],'COMPR':[2,'A0',2],'DIV':[3,'24',1]
,'DIVF':[3,'64',1],'DIVR':[2,'9C',2],'FIX':[1,'C4',0],'FLOAT':[1,'C0',0],'HIO':[1
,'F4',0],'J':[3,'3C',1],'JEQ':[3,'30',1],'JGT':[3,'34',1],'JLT':[3,'38',1],'JSUB'
:[3,'48',1],'LDA':[3,'00',1],'LDB':[3,'68',1],'LDCH':[3,'50',1],'LDF':[3,'70',1],
'LDL':[3,'08',1],'LDS':[3,'6C',1],'LDT':[3,'74',1],'LDX':[3,'04',1],'LPS':[3,'D0'
,1],'MUL':[3,'20',1],'MULF':[3,'60',1],'MULR':[2,'98',2],'NORM':[1,'C8',0],'OR':[
3,'44',1],'RD':[3,'D8',1],'RMO':[2,'AC',2],'RSUB':[3,'4C',0],'SHIFTL':[2,'A4',2],
'SHIFTR':[2,'A8',2],'SIO':[1,'FO',0],'SSK':[3,'EC',1],'STA':[3,'OC',1],'STB':[3,'
78',1],'STCH':[3,'54',1],'STF':[3,'80',1],'STI':[3,'D4',1],
'STL':[3,'14',1],'STS':[3,'7C',1],'STSW':[3,'E8',1],'STT':[3,'84',1],'STX':[3,'10
',1],
'SUB':[3,'1C',1],'SUBF':[3,'5C',1],'SUBR':[2,'94',2],'SVC':[2,'B0',1],'TD':[3,'E0
',1],
'TIO':[1,'F8',0],'TIX':[3,'2C',1],'TIXR':[2,'B8',1],'WD':[3,'DC',1]
            }

#program='''LDA #5
#STA ALPHA
#LDCH #90
#STCH C1'''

program= open('progam.txt', 'r').read()
print(program)
lines=program.split('\n')
try:
```

```
    lines.remove('')
except:
    pass
print(lines)
i=1
for line in lines:
  print("Line "+str(i))
  word=line.split(' ')
  print(word)
  print("Mnemonic "+word[0])
  print("Operand  "+str(word[1:]))
  print("Opcode   "+str(OPCODETAB[word[0].strip()][1]))

  i=i+1


#   Python 3.6.1 (default, Dec 2015, 13:05:11)
# [GCC 4.8.2] on linux
# LDA #5
# STA ALPHA
# LDCH #90
# STCH C1
# ['LDA #5', 'STA ALPHA', 'LDCH #90', 'STCH C1']
# Line 1
# ['LDA', '#5']
# Mnemonic LDA
# Operand  ['#5']
# Opcode   00
# Line 2
# ['STA', 'ALPHA']
# Mnemonic STA
# Operand  ['ALPHA']
# Opcode   OC
# Line 3
# ['LDCH', '#90']
# Mnemonic LDCH
# Operand  ['#90']
# Opcode   50
# Line 4
# ['STCH', 'C1']
# Mnemonic STCH
# Operand  ['C1']
# Opcode   54
```

Conclusion: I got to learn about SIC, it's mnemonic and corresponding opcodes by creating mnemonic table after scanning sic program file