

PROJECT REPORT

CSN 252

ALPHABETS

TEAM Members:

- Avvari Sreedhar - 19114016
- Rohith Mamidi - 19114051
- Sasi preetham - 19114062
- Jaideep Putta - 19114066
- Akanti Giri Nandan - 19115016

Project statement

To print all the alphabets repeatedly using assembly language (SIC INSTRUCTIONS) and simulate it using SIC tools.

Pseudo Code

1. Clear accumulator then store accumulator into (offset) a variable
 2. Initialize a loop (to print 'A' to 'Z' , 'a' to 'z' in a line) and store the required character to print in the accumulator.
 3. Wait for the device until it is ready and then write the required character into the device.
 4. Increase the offset that corresponds to the next character and verify the loop termination condition i.e. offset < 26.
 5. When the offset is greater than or equal to 26 we need to start printing small case alphabets.
 6. Then after printing each small letter we run the same loop and when we print all characters (small and upper case) we go to point 7.
 7. If the loop is terminated (i.e. Set of all alphabets were printed) store a newline character inorder to print the next repeated line in a fresh line and write to the device
 8. Then jump to instruction 1 to clear the accumulator and repeat the same process.
-

SIC Assembly Code :

```
abc      START      0
repeat   CLEAR      A          .Clear the accumulator value
          STA        offset     .Stores accumulator into the offset

loop      LDA        offset     .Start of the loop
          ADD        asciiA     .Update accumulator for next alphabet

td1       TD         device     .Tests the device to update the equality flag
          JEQ        td1        .Wait until the device is ready
          WD         device     .Writes accumulator to device

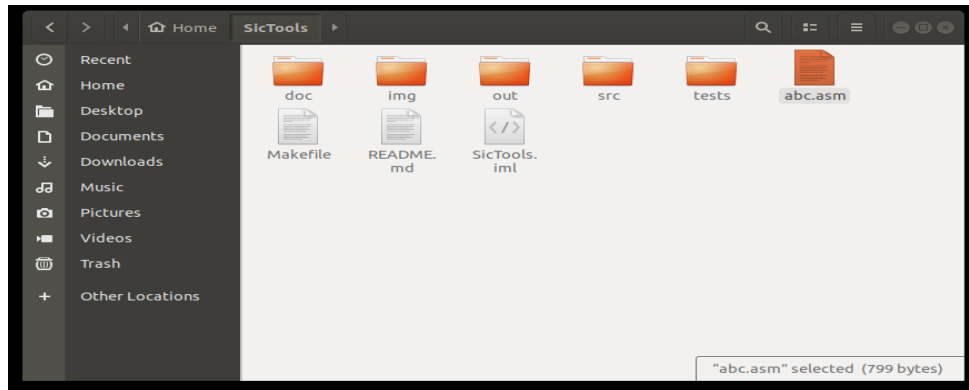
          LDA        offset
          ADD        one        .Increase the offset by 1
          STA        offset
          COMP       count
          JLT        loop       .Jumps to the loop if offset < count(26)
          JEQ        loop1
          COMP       count1
          JLT        loop
          LDA        newline    .Update accumulator to newline character

td2       TD         device
          JEQ        td2
          WD         device     .Writes newline to device
          J          repeat     .Jump to beginning

loop1     TD         device     .Tests the device
          JEQ        loop1
          ADD        six
          STA        offset
          WD         device
          J          loop

. constants
device BYTE  X'01'
newline WORD  10
asciiA WORD  65
count WORD  26
count1 WORD  58
. variables
offset WORD  0
one WORD  1
six WORD  6
          END          loop
```

File Structure:



Source code (asm file):

```
Open  abc.asm  Save
abc    START    0
repeat CLEAR    A
      STA      offset    .Clear the accumulator value
                        .Stores accumulator into the offset
loop   LDA      offset    .Start of the loop
      ADD      asciiA    .Update accumulator for next alphabet
td1    TD       device    .Tests the device to update the equality flag
      JEQ      td1       .Wait until the device is ready
      WD       device    .Writes accumulator to device

      LDA      offset
      ADD      one       .Increase the offset by 1
      STA      offset
      COMP     count
      JLT      loop      .Jumps to the loop if offset < count(26)
      JEQ      loop1     .Jumps to loop1 after printing all the upper case alphabets
      COMP     count1
      JLT      loop      .Termination occurs after printing all the alphabets in a line
      LDA      newline   .Update accumulator to newline character
td2    TD       device
      JEQ      td2       .Writes newline to device
      WD       device

      J        repeat    .Jump to beginning
loop1  TD       device    .Tests the device
      JEQ      loop1     .Wait until the device is ready
      ADD      six       .To neglect characters between 'Z' & 'a'
      STA      offset    .Stores accumulator into the offset
      WD       device    .Writes accumulator to device
      J        loop      .Jumps to the loop

. constants
device BYTE    X'01'
newline WORD    10
asciiA  WORD    65
asciia  WORD    97
count   WORD    26
count1  WORD    58

. variables
offset  WORD    0
one     WORD    1
six     WORD    6
      END      loop

Plain Text  Tab Width: 8  Ln 9, Col 86  INS
```

Starting the simulator via command line:

```
rohith@rohith-Lenovo: ~/SicTools
File Edit View Search Terminal Help

rohith@rohith-Lenovo:~/SicTools$ java -jar out/make/sictools.jar abc.asm
```

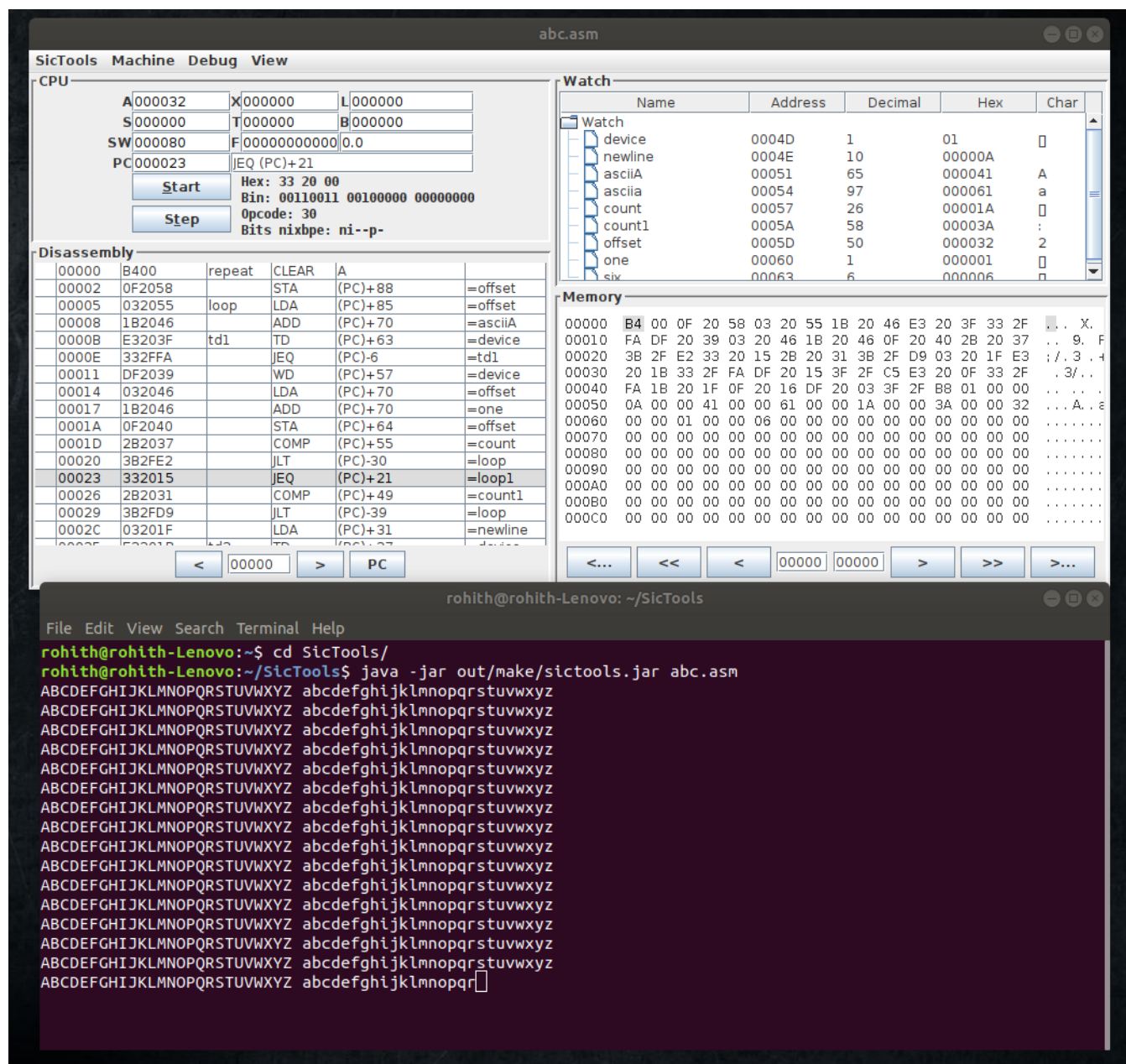
Simulating tool (initially):

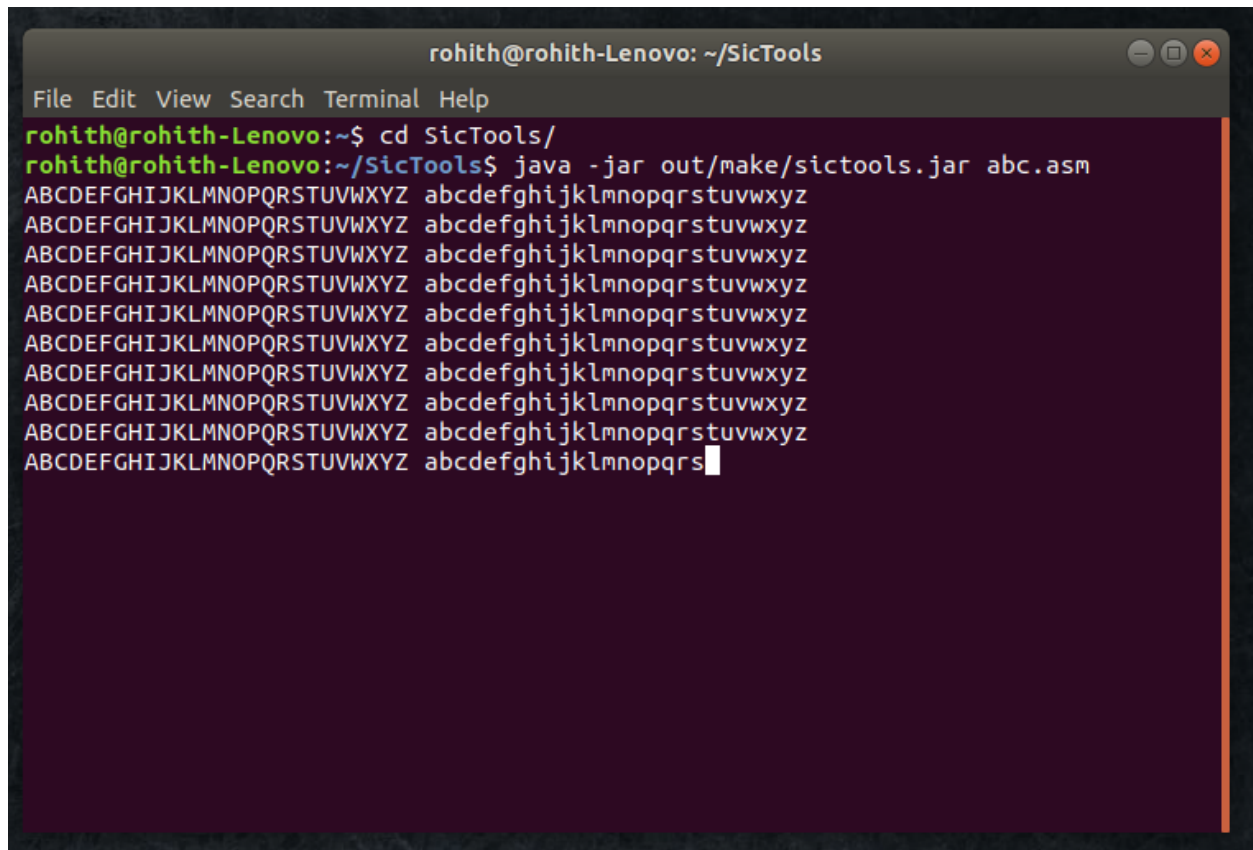
The screenshot displays the SicTools simulator interface for the file `abc.asm`. The interface is divided into several panes:

- CPU:** Shows the state of the 8080-style CPU registers. The PC (Program Counter) is at address 000005, containing the instruction `LDA (PC)+85`. The instruction is being executed, and the PC is updated to 000006.
- Disassembly:** A list of instructions with their addresses and opcodes. The current instruction at address 000005 is `LDA (PC)+85`.
- Watch:** A table of variables being watched, including `device`, `newline`, `asciiA`, `count`, `count1`, `offset`, `one`, and `six`.
- Memory:** A view of the memory contents, showing addresses from 000000 to 0000FF and their corresponding hexadecimal values.

The CPU pane also includes a `Start` button and a `Step` button. The `Disassembly` pane shows the instruction `LDA (PC)+85` being executed, with the PC updated to 000006.

Screenshots of running program:





```
rohith@rohith-Lenovo: ~/SicTools
File Edit View Search Terminal Help
rohith@rohith-Lenovo:~$ cd SicTools/
rohith@rohith-Lenovo:~/SicTools$ java -jar out/make/sictools.jar abc.asm
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqr
```

Project Repository

Repository Name: alphabet

Github Link: <https://github.com/AvvariSreedhar/alphabet/tree/main>