

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 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. $\text{offset} < 26$.
 5. If the loop is terminated (i.e. Set of all alphabets were printed) store a newline character in order to print the next repeated line in a fresh line and write to the device
 6. 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     .Increase the offset by 1
          ADD        one
          STA        offset
          COMP       count
          JLT        loop      .Jumps to the loop if offset < count(26)

          LDA        newline    .Update accumulator to newline character

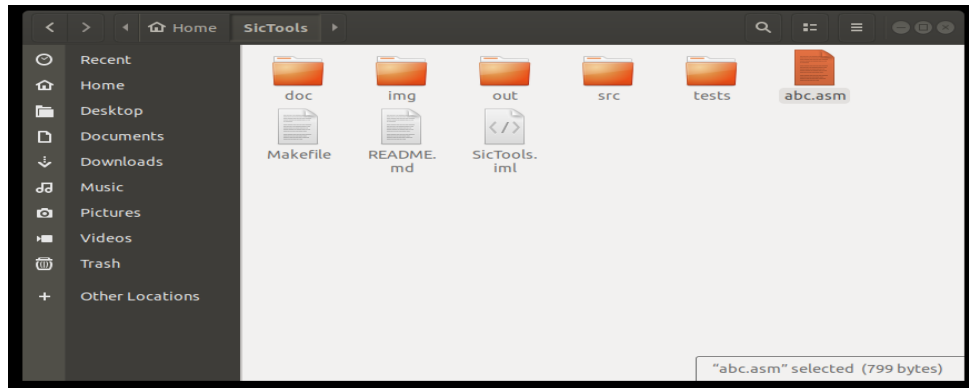
td2      TD         device
          JEQ        td2
          WD         device     .Writes newline to device

          J          repeat     .Jump to beginning

. constants
device   BYTE      X'01'
newline  WORD       10
asciiA   WORD       65
count    WORD       26

. variables
offset   WORD       0
one      WORD       1
          END        loop
```

File Structure:



Source code:

```
Open  abc.asm  Save
~/SicTools

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  .Increase the offset by 1
                ADD        one
                STA        offset
                COMP       count
                JLT        loop    .Jumps to the loop if offset < count(26)
                LDA        newline .Update accumulator to newline character
td2             TD         device
                JEQ        td2
                WD         device  .Writes newline to device
                J          repeat  .Jump to beginning

. constants
device BYTE    X'01'
newline WORD    10
asciiA  WORD    65
count   WORD    26

. variables
offset  WORD    0
one     WORD    1
        END      loop
```

Plain Text Tab Width: 8 Ln 11, Col 58 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

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

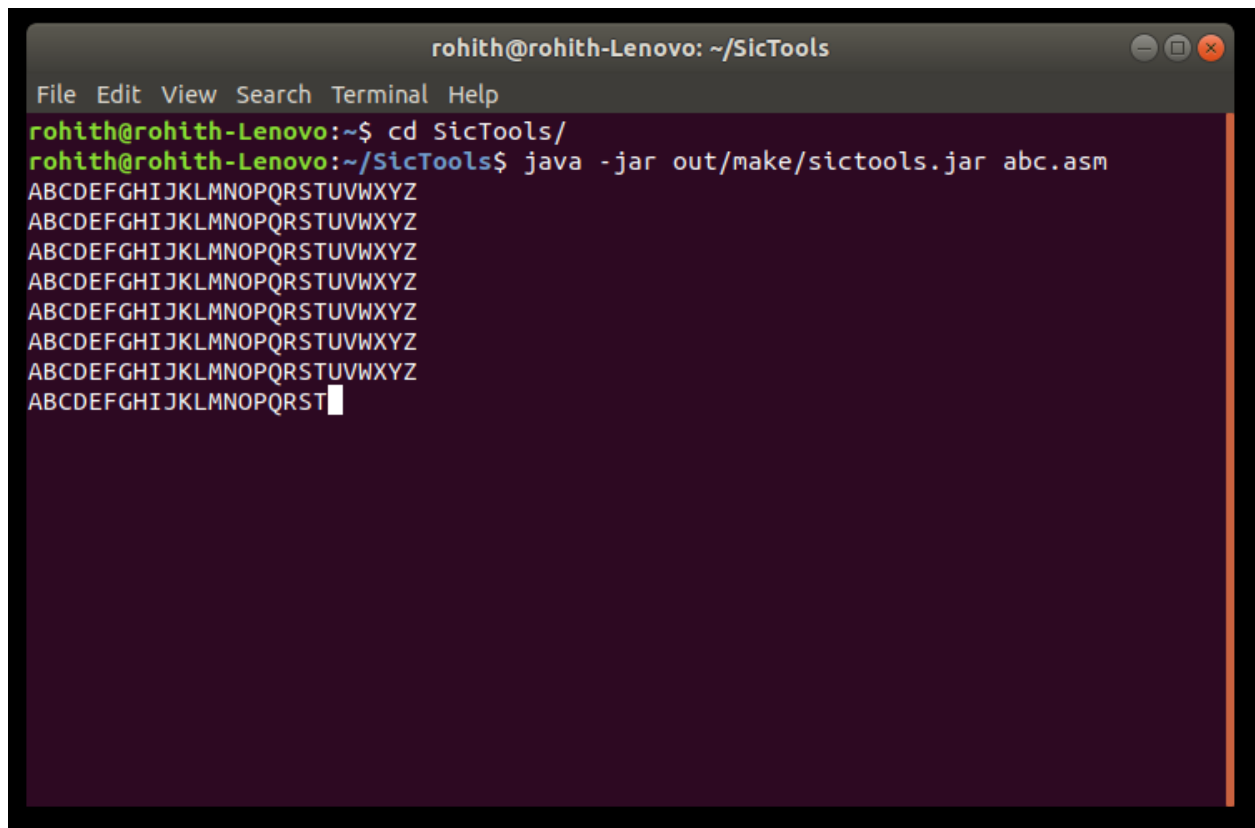
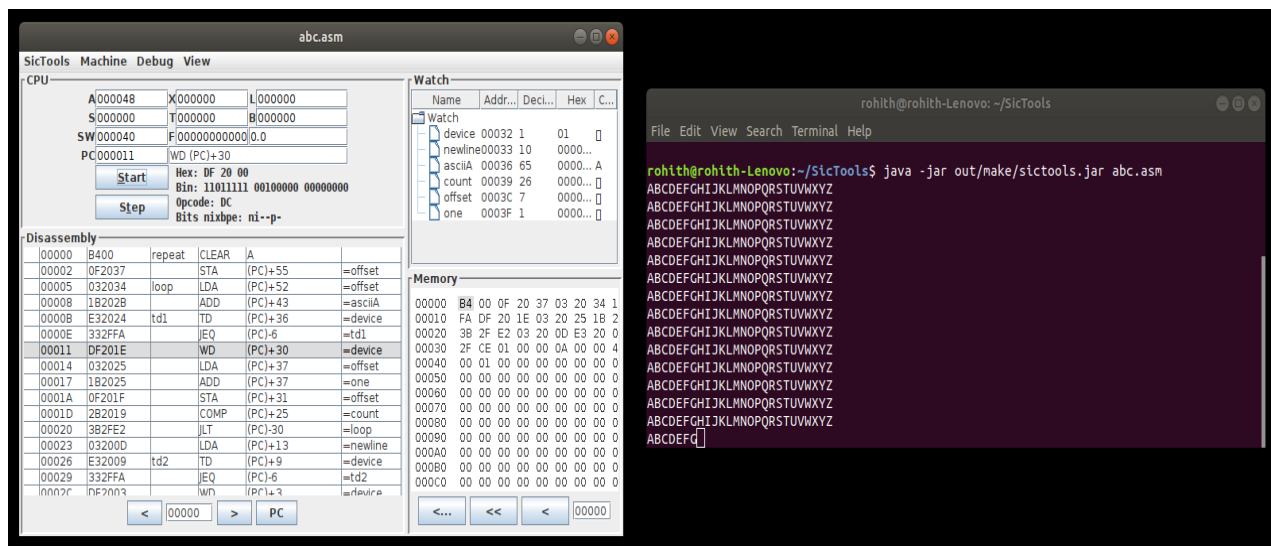
- CPU:** Shows registers A, S, T, B, SW, F, and PC. Below the registers are buttons for 'Start' and 'Step'. The status bar indicates 'Hex: 03 20 00', 'Bin: 00000011 00100000 00000000', 'Opcode: 00', and 'Bits nixbpe: ni--p-'.
- Disassembly:** A table showing the assembly code being executed.
- Watch:** A table showing variables being watched.
- Memory:** A table showing the memory contents.

Address	Hex	Bin	Char
00000	B400	repeat	CLEAR
00002	0F2037	STA	(PC)+55
00005	032034	loop	LDA (PC)+52
00008	1B202B	ADD	(PC)+43
0000B	E32024	td1	TD (PC)+36
0000E	332FFA	JEQ	(PC)-6
00011	DF201E	WD	(PC)+30
00014	032025	LDA	(PC)+37
00017	1B2025	ADD	(PC)+37
0001A	0F201F	STA	(PC)+31
0001D	2B2019	COMP	(PC)+25
00020	3B2FE2	JLT	(PC)-30
00023	03200D	LDA	(PC)+13
00026	E32009	td2	TD (PC)+9
00029	332FFA	JEQ	(PC)-6
0002C	DF2003	WD	(PC)+3
0002F	3F2FCE	J	(PC)-50
00032	010000	device	LDA #0
00035	0A0000	LDL	@0

Name	Address	Decimal	Hex	Char
device	00032	1	01	
newline	00033	10	00000A	
asciiA	00036	65	000041	A
count	00039	26	00001A	
offset	0003C	0	000000	
one	0003F	1	000001	

Address	Hex	Bin	Char
00000	B4	00 0F 20 37 03 20 34 1B 20 2B E3 20 24 33 2F	... 7. 4. +. \$3/
00010	FA DF 20 1E 03 20 25 1B 20 25 0F 20 1F 2B 20 19	... % % . + .	
00020	3B 2F E2 03 20 0D E3 20 09 33 2F FA DF 20 03 3F	:/... .. 3/... ?	
00030	2F CE 01 00 00 0A 00 00 41 00 00 1A 00 00 00 00	/.....A.....	
00040	00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

Screenshots of running program:



Project Repository

Repository Name: alphabet

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