



# System Software CSN-252 Assembler

Refer L. L. Beck book, 3<sup>rd</sup> Ed.



## Tutorial 4 (OS: Ubuntu)



Step 1: Install Pascal compiler

```
sudo apt-get update
```

```
sudo apt-get install fp-compiler
```

Step 2: Go to procedure initialize and uncomment lines 1734-1741. Similarly, uncomment the close statements at lines 831, 1855-1860 (Ref: SICSIM.doc)

Step 3: place the object program to be loaded in file DEVF2, the object code for the loader in file DEVF1, and the object code for the SIC bootstrap in file DEV00. (Ref: SICLDR.doc)

```
%cp SICBOOT.OBJ DEV00
```

```
%cp SICLDR.OBJ DEVF1
```

## Tutorial 4 (OS: Ubuntu)



Step 4: Compile the program

```
%fpc sicsim.pas
```

Step 5: Run the simulator

```
%sicsim
```

```
S
```

```
B 1003
```

```
R (multiple times till breakpoint is reached)
```

```
D 1000-101F
```

```
D R
```

IIT ROORKEE

## Tutorial 4 (OS: Ubuntu)



Exercise 1: Compile SICSIM.PAS to install SIC simulator

Exercise 2: Write object program for the SIC program in Tut 3 Q3 (store 2 at memory location labelled as K10). Store this file in DEVF2.

Exercise 3: Trace the execution of the test program (file DEVF2) using this simulator

(a) Set breakpoints 1003, 1006, and 1009

(b) Display the contents of Accumulator and memory (1000 – 101F)

IIT ROORKEE

## Disassembly



HTEST 001000000014

T0010000C00100F0C100C501012541013

T00100F040000055A

E001000



IIT ROORKEE

1	test	start	1000	14	rdrec	ldx	zero
2	<b>first</b>	<b>stl</b>	<b>retadr</b>	15		<b>lda</b>	<b>zero</b>
3	cloop	jsub	rdrec	<b>16</b>	<b>loop</b>	<b>td</b>	<b>input</b>
4		lda	length	<b>17</b>		<b>jeq</b>	<b>loop</b>
5		comp	zero	18		rd	input
6		jeq	endfil	19		comp	zero
7		j	cloop	20		jeq	exit
8	<b>endfil</b>	<b>ldl</b>	<b>retadr</b>	21		stch	buffer,x
9		<b>rsub</b>		22		tix	maxlen
10	zero	word	0	23		jlt	loop
11	retadr	resw	1	24	exit	stx	length
12	length	resw	1	25		<b>rsub</b>	
13	buffer	resb	4096	26	input	byte	x'f3'
	:			27	maxlen	word	4096
				28		end	first

- assembly process does not require any understanding of the program being assembled

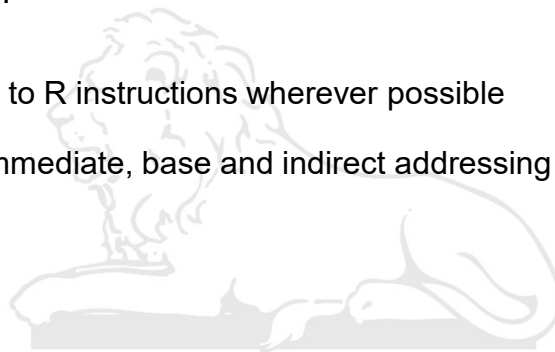
- Buffer is needed because the I/O rates for the two devices may be very different
- The end of each record is marked with a null character (hex 00)
- If a record is longer than the length of the buffer (4096 bytes), only the first 4096 bytes are copied
- The end of the file to be copied is indicated by a zero-length record

## SIC/XE Program



Differences:

1. Use of R to R instructions wherever possible
2. Use of immediate, base and indirect addressing



1	COPY	START 0	1	test	start	1000
2	FIRST	STL RETADR	2	<b>first</b>	<b>stl</b>	<b>retadr</b>
3		LDB #LENGTH	3	cloop	jsub	rdrec
4		BASE LENGTH	4		lda	length
5	CLOOP	+JSUB RDREC	5		comp	zero
6		LDA LENGTH	6		jeq	endfil
7		COMP #0	7		j	cloop
8		JEQ ENDFIL	8	endfil	ldl	retadr
9		J CLOOP	9		<b>rsub</b>	
10	ENDFIL	<b>J @RETADR</b>	10	zero	word	0
11	RETADR	RESW 1	11	retadr	resw	1
12	LENGTH	RESW 1	12	length	resw	1
13	BUFFER	RESB 4096	13	buffer	resb	4096
	:			:		

14	RDREC	CLEAR X	14	rdrec	ldx	zero
15		CLEAR A	15		lda	zero
16		CLEAR S	16	loop	td	input
17		+LDT #4096	17		jeq	loop
18	RLOOP	TD INPUT	18		rd	input
19		JEQ RLOOP	19		comp	zero
20		RD INPUT	20		jeq	exit
21		COMPR A,S	21		stch	buffer,x
22		JEQ EXIT	22		tix	maxlen
23		STCH BUFFER,X	23		jlt	loop
24		TIXR T	24	exit	stx	length
25		JLT RLOOP	25		<b>rsub</b>	
26	EXIT	STX LENGTH	26	input	byte	x'f3'
27		RSUB	27	maxlen	word	4096
28	INPUT	BYTE X'F3'	28		end	first
29		END FIRST				

**Machine Dependent Assembler Features**

Main differences (SIC/XE program )

- Use of register-to-register instructions
- Use of indirect and immediate addressing
- Instructions that refer to memory - assembled using either the program-counter relative or the base relative mode.
- Assembler directive BASE is used in conjunction with base relative addressing.
- If the displacements required for both program-counter relative and base relative addressing are too large – 4 byte extended format is used
- Advantages: These changes improve the execution speed of the program.