The Simplified Instructional Computer (SIC)

System Programmer's Reference

Last Revision 9/2/2025

Section 1: Overview

This machine's sole design purpose is to illustrate typical hardware features in actual systems. The design of the SIC is simple and avoids idiosyncrasies often found in real computer systems. The SIC is targeted for the embedded systems market. As a result, its design does not scale to fit the requirements of an interactive system.

The SIC actually "ships" in two versions: the standard version and an XE version. The XE stands for Extended Edition (or Extra eXpensive). The design of the SIC architecture is upward compatible: an object program for the SIC machine will execute with no modification and produce the same results as the SIC/XE machine.

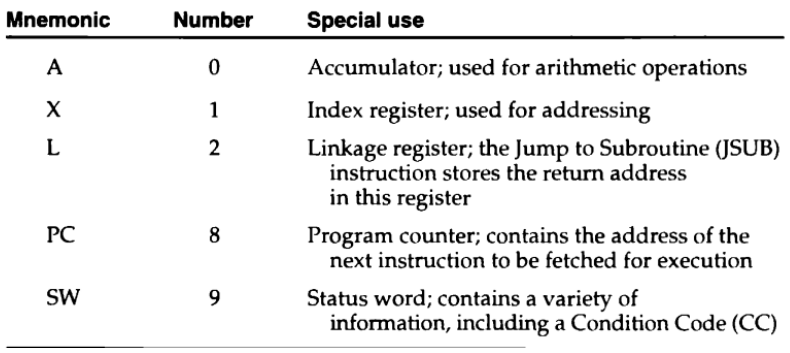
Section 2: SIC Machine Architecture

2.1 Memory

Memory in the SIC consists of 8-bit bytes, and any three consecutive bytes form a *word* (24 bits). The SIC is limited to 32K of RAM (2^{15} bytes).

2.2 CPU Registers

The SIC CPU has five registers. All five registers are reserved for special use. Every register can store a full word (24 bits). The following table indicates numbers, mnemonics, and the special use for each register. The numbering for these registers on the SIC has been chosen for upward compatibility with the SIC/XE:



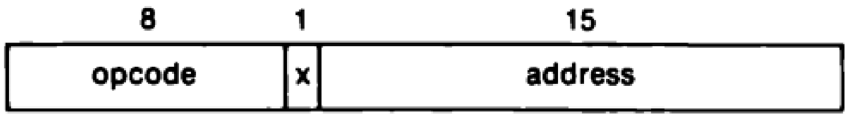
Mnemonic	Number	Special use
A	0	Accumulator; used for arithmetic operations
X	1	Index register; used for addressing
L	2	Linkage register; the Jump to Subroutine (JSUB) instruction stores the return address in this register
PC	8	Program counter; contains the address of the next instruction to be fetched for execution
SW	9	Status word; contains a variety of information, including a Condition Code (CC)

2.3 Data Formats

All integers are stored as 24-bit binary numbers and are _assumed to be signed_ using two’s complement representation. Characters are stored using their 8-bit ASCII codes. The SIC has no support for floating-point.

2.3 Instruction Formats

All SIC instructions have the following 24-bit format:



The x bit is used to indicate the indexed-addressing mode when set

Note: The SIC/XE supports this 24-bit SIC format and has additional instruction formats.

2.4 Addressing Modes

The SIC supports two addressing modes. Each mode is selected by setting the x bit in the instruction. The following table describes how the target address is calculated from the address provided in the instruction. Parenthesis indicates the contents of a register or memory location is used.

Mode	**Indication**	**Target address calculation**
Direct	x = 0	TA = address
Indexed	x = 1	TA = address + (X)

2.5 SIC Instruction Set

The SIC provides a set of instructions which support a basic set of simple tasks. Instructions for loading and storing registers to and from memory and performing integer arithmetic are included. A comparison instruction is included for doing comparisons and setting CPU condition codes. There are also instructions for jumping to a different address (like GOTO) and jumps which return to the next instruction from caller when finished (JSUB and RSUB).

2.6 Programmed Input/Output (PIO)

Input/Output operations are performed on the SIC one byte at a time to or from the rightmost 8 bits of register A. Each connected device is assigned an 8-bit device ID. The SIC has three I/O instructions, each of which has a one-byte operand which specifies the device ID.

There is a test device instruction (TD) which tests whether the device ID is ready to send or receive a single byte of data. The CPU condition code (CC) is set to indicate the result of this test.

2.7 Halting Execution / Return to Monitor

Halthing the currently running program and returning to the caller or system monitor is accomplished by placing a zero in the PC register via two instructions:

```
LDL ZERO

RSUB
```

3. SIC Instructions

Mnemonic	Format	Opcode	Effect	Notes
ADD m	3/4	18	A<- (A) + (m..m+2)	
ADDF m	3/4	58	F<- (F) + (m..m+5)	X, F
ADDR r1, r2	2	90	R2 <- (r2) + (r1)	X
AND m	3/4	40	A <- (A) & (m..m+2)	
CLEAR r1	2	B4	R1 <- 0	X
COMP m	3/4	28	(A): (m..m+2)	C
COMPF m	3/4	88	(F) : (m..m+5)	X, F, C
COMPR r1, r2	2	A0	(r1) : (r2)	X, C
DIV m	3/4	24	A <- (A) / (m..m+2)	
DIVF m	3/4	64	F <- (F) / (m..m+5)	X, F
DIVR r1, r2	2	9C	r2 <- (r2) / (r1)	X
FIX	1	C4	A <- (F) [convert float to integer]	X, F
FLOAT	1	C0	F <- (A) [convert to floating]	X, F
HIO	1	F4	Halt Input/Output Channel number (A)	P, X
J m	3/4	3C	(PC) <- m	
JEQ m	3/4	30	(PC) <- m if CC set to =	
JGT m	3/4	34	(PC) <- m if CC set to >	
JLT m	3/4	38	(PC) <- m if CC set to <	
JSUB m	3/4	48	L <- (PC); (PC) <- m	
LDA m	3/4	00	A <- (m..m+2)	
LDB m	3/4	68	B <- (m..m+2)	X
LDCH m	3/4	50	A[rightmost byte] <- (m)	
LDF m	3/4	70	F <- (m..m+5)	X, F
LDL m	3/4	08	L <- (m..m+2)	
LDS m	3/4	6C	S <- (m..m+2)	X
LDT m	3/4	74	T <- (m..m+2)	X
LDX m	3/4	04	X <- (m..m+2)	
LPS m	3/4	D0	Load processor status from information beginning at address m	P, X
MUL m	3/4	20	A <- (A) * (m..m+2)	
MULF m	3/4	60	(f) <- (F) * (m..m+5)	X, F
MULR r1, r2	2	98	r2 <- (r2) * (r1)	X
NORM	1	C8	F <- (F) [normalized]	X, F
OR m	3/4	44	A <- (A) | (m..m+2)	
RD m	3/4	D8	A[rightmost byte] <- data from device specified by (m)	P
RMO r1, r2	2	AC	r2 <- (r1)	X
RSUB	3/4	4C	PC <- (L). Return from subroutine	
SHIFTL r1, n	2	A4	r1 <- (r1). Left circular shift n bits.	X
SHIFTR r1, n	2	A8	r1 <- (r1). Left circular shift n bits.	X

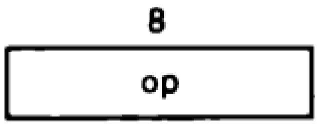
SIO	1	FO	Start Input/Output Channel number (A). Address of channel program is given by (S)	P, X
SSK m	3/4	EC	Protection key for address m<-(A)	P, X
STA m	3/4	OC	m..m+2 <- (A)	
STB m	3/4	78	m..m+2 <- (B)	X
STCH m	3/4	54	m <- (A) (rightmost byte)	
STF m	3/4	80	m..m+5 <- (F)	X, F
STI m	3/4	D4	Interval timer value <- (m..m+2)	P, X
STL m	3/4	14	m..m+2 <- (L)	
STS m	3/4	7C	m..m+2 <- (S)	X
STSW m	3/4	E8	m..m+2 <- (SW)	P
STT m	3/4	84	m..m+2 <- (T)	X
STX m	3/4	10	m..m+2 <- (X)	
SUB m	3/4	1C	A <- (A) – (m..m+2)	
SUBF m	3/4	5C	F <- (F) = (m…m+5)	X, F
SUBR r1, r2	2	94	R2 <- (r2) – (r1)	X
SVC n	2	B0	Generate a service interrupt (Assembled instruction, R1 = n)	X
TD m	3 4	E0	Test device specified by m	P, C
TIO	1	F8	Test I/O Channel Number (A)	P, X, C
TIX m	3 4	2C	X <- (x) + 1; (X): (m..m+2)	C
TIXR r1	2	B8	X <- (x) + 1; (x): (r1)	X, C
WD m	3 4	DC	Device specified by (m) <- (A) [rightmost byte]	P

The letters in the notes column above have the following meanings:

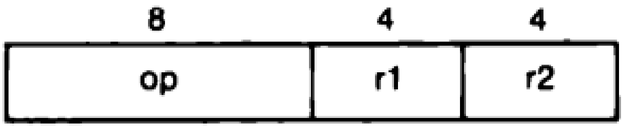
Letter	Meaning
P	Privileged instruction
X	XE Only Instruction
F	Floating-point instruction
C	Condition Code Set to indicate result

Instruction formats:

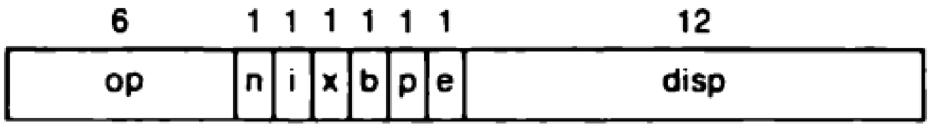
Format 1 (1 byte):



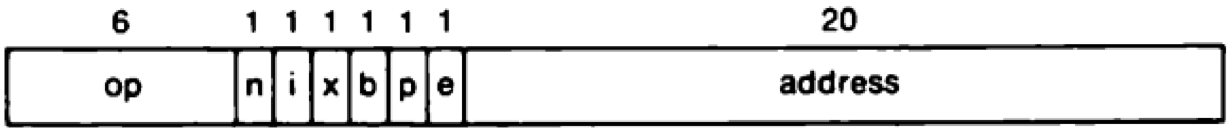
Format 2 (2 bytes):



Format 3 (3 bytes):



Format 4 (4 bytes):



Revision History

09/01/2020 – Initial Release