
System Programming

**System Software:
An Introduction to Systems Programming**

**Leland L. Beck
3rd Edition
Addison-Wesley, 1997**

http://web.thu.edu.tw/ctyang/

Vita - Welcome to CT Yang's homepage - Microsoft Internet Explorer

檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 連結 網址 http://web.thu.edu.tw/ctyang/vita.php Google

Welcome to CT Yang's homepage



Vita

Chao-Tung Yang was born on November 9, 1968 in Yilan, Taiwan, R.O.C. He received a B.S. degree in Computer Science and Information Engineering from Tunghai University, Taichung, Taiwan in 1990, and an M.S. degree in Computer Science from National Chiao Tung University, Hsinchu, Taiwan in 1992. He received a Ph.D. degree in Computer Science from National Chiao Tung University in July 1996. He won the 1996 Acer Dragon Award for outstanding Ph.D. Dissertation. He has worked as an associate researcher for ground operations in the ROC SAT Ground System Section (RGS) of the National Space Program Office (NSPO) in Hsinchu's Science-based Industrial Park since 1996. In Aug. 2001, he joined the faculty of the Department of Computer Science and Information Engineering at Tunghai University, where he is currently an Associate Professor. His researches have been sponsored by Taiwan agencies National Science Council (NSC), National Center for High Performance Computing (NCHC), and Ministry of Education, as well as a number of industrial organizations. His present research interests are in grid and cluster computing, parallel and high-performance computing, and internet-based applications. He is both



Navigation

- [Home](#)
- [Vita](#)
- [Education](#)
- [Current Position and Past Experiences](#)
- [Recent and Past Activities](#)
- [Reviewing Activities](#)
- [Project](#)
- [Schedule](#)
- [Teaching](#)
- [Publications](#)

Links

- [NSC](#)
- [NCHC](#)
- [NSPO](#)
- [THU](#)
- [CSIE in THU](#)
- [HPC Lab.](#)
- [UniGrid \[1\] \[2\]](#)
- [TIGER](#)
- [TIGER Broker](#)
- [GGF](#)

Contact and Additional Information

Email: ctyang@thu.edu.tw
Phone: +886-4-2359-0415
Fax: +886-4-2359-1567

- [Schedule](#)
- [Teaching](#)
- [Publications](#)
- [HPC Lab.](#)

CT YANG@HPC

網際網路

Score List

- **Participation: 5%**
- **Two quizzes: 20% (each 10%)**
- **Two or three homework: 10%**
- **A mid exam: 20%**
- **A final exam: 25%**
- **A final project: 25%**

System Programming

- **Chapter 1: Background**
- **Chapter 2: Assemblers**
- **Chapter 3: Loaders and Linkers**
- **Chapter 4: Macro Processors**
- **Chapter 5: Compilers**
- **Operating Systems**
- **Other System Software**
- **Software Engineering Issues**

Chapter 1

Background

Outline

- Introduction
- System Software and Machine Architecture
- The Simplified Instructional Computer (SIC)
 - SIC Machine Architecture
 - SIC/XE Machine Architecture
 - SIC Programming Examples
- Traditional (CISC) Machines
- RISC Machines

1.1 Introduction

- **System Software consists of a variety of programs that support the operation of a computer.**
- **The software makes it possible for the users to focus on an application or other problem to be solved, without needing to know the details of how the machine works internally.**

1.1 Introduction

- Machine **dependency** of system software
 - System programs are intended to support the operation and use of the computer.
 - Machine architecture differs in:
 - Machine code
 - Instruction formats
 - Addressing mode
 - Registers
- Machine **independency** of system software
 - General design and logic is basically the same:
 - Code optimization
 - Subprogram linking

1.2 System Software and Machine Architecture

- One characteristic in which most system software differs from application software is **machine dependency**.
- System programs are **intended to support the operation and use of the computer itself**, rather than any particular application.
- e.g. of system software
 - Text editor, assembler, compiler, loader or linker, debugger, macro processors, operating system, database management systems, software engineering tools, ...

- 一群支援電腦運作的程式。
- 使得使用者可以專注於開發應用程式與解決問題，而不需要了解機器的內部運作。
- 應用程式(Application)
 - 是以電腦為工具，用來解決某些問題。
- 系統程式(System Program)
 - 是用來支援使用者對電腦的使用與操作。
 - 與機器的結構有關，但與機器的特性無關。
- 本課程將以Simplified Instructional Computer (SIC、SIC/XE)系列的電腦作為系統程式的討論平台。

1.2 System Software and Machine Architecture

- **Text editor**

- ❑ To create and modify the program

- **Compiler and assembler**

- ❑ You translated these programs into machine language

- **Loader or linker**

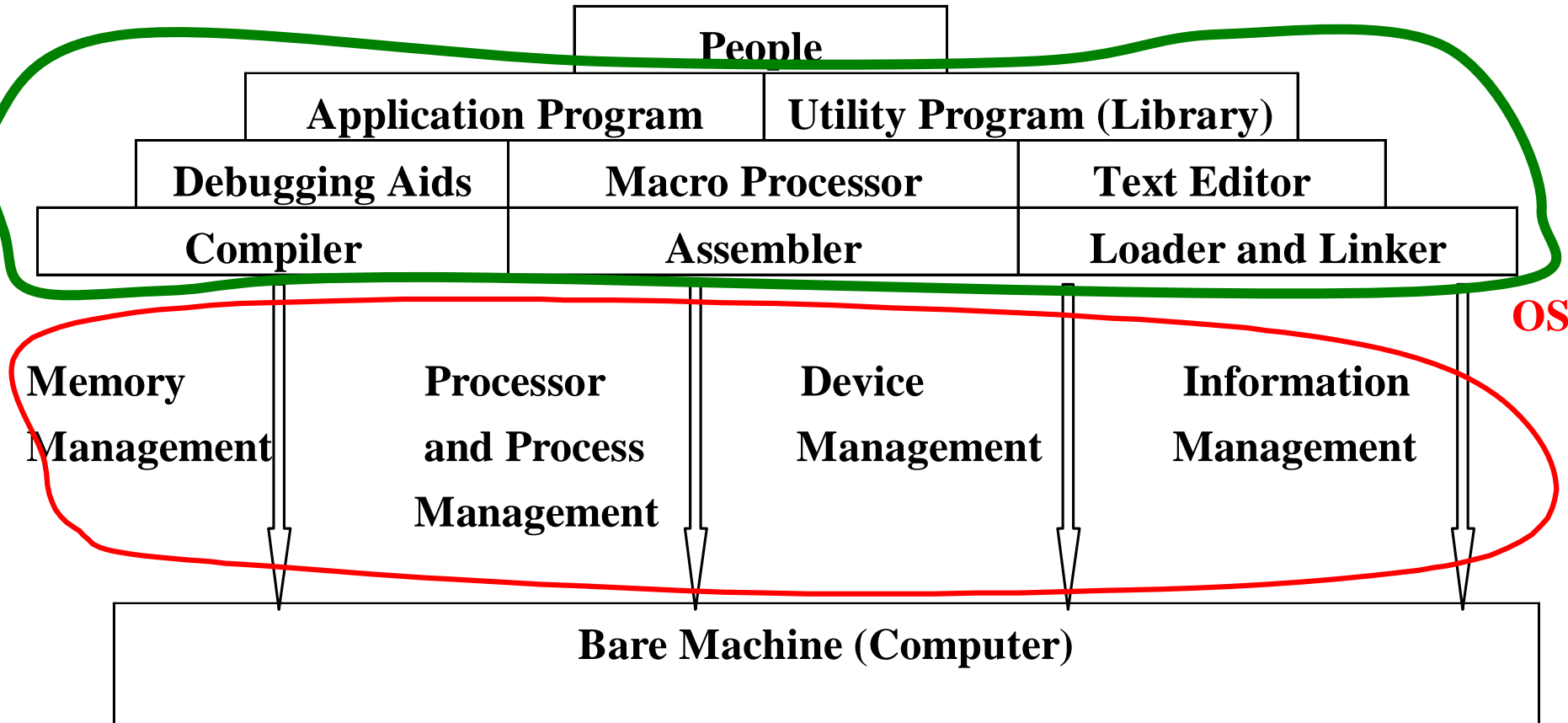
- ❑ The resulting machine program was loaded into memory and prepared for execution

- **Debugger**

- ❑ To help detect errors in the program

系統程式所包含的範圍

SP



1.3 The Simplified Instructional Computer

- Like many other products, SIC comes in two versions
 - The standard model
 - An XE version
 - “extra equipments”, “extra expensive”
- The two versions has been designed to be upward compatible
- SIC (Simplified Instructional Computer)
- SIC/XE (Extra Equipment)

1.3 The Simplified Instructional Computer

■ SIC

- ❑ Upward compatible
- ❑ Memory consists of 8-bit bytes, 3 consecutive bytes form a word (24 bits)
- ❑ There are a total of 32768 bytes (32 KB) in the computer memory.
- ❑ 5 registers, 24 bits in length
 - A 0 Accumulator
 - X 1 Index register
 - L 2 Linkage register (JSUB)
 - PC 8 Program counter
 - SW 9 Status word (Condition Code)

1.3.1 SIC Machine Architecture

■ Data Formats

- ❑ Integers are stored as **24-bit binary number**
- ❑ **2's complement** representation for negative values
- ❑ Characters are stored using **8-bit ASCII codes**
- ❑ **No** floating-point hardware on the standard version of SIC

1.3.1 SIC Machine Architecture

■ Instruction format

- 24-bit format
- The flag **bit x** is used to indicate **indexed-addressing mode**



■ Addressing Modes

- There are two addressing modes available
 - Indicated by x bit in the instruction
 - (X) represents the contents of reg. X

Mode	Indication	Target address calculation
Direct	$x = 0$	$TA = \text{address}$
Indexed	$x = 1$	$TA = \text{address} + (X)$

1.3.1 SIC Machine Architecture

■ Instruction set

- ❑ Format 3
- ❑ Load and store registers (LDA, LDX, STA, STX, etc.)
- ❑ Integer arithmetic operations (ADD, SUB, MUL, DIV)
- ❑ Compare instruction (COMP)
- ❑ Conditional jump instructions (JLT, JEQ, JGT)
- ❑ JSUB jumps to the subroutine, placing the return address in register L.
- ❑ RSUB returns by jumping to the address contained in register L.

1.3.1 SIC Machine Architecture

■ I/O

- ❑ I/O are performed by **transferring 1 byte at a time** to or from the rightmost 8 bits of **register A**.
- ❑ Each device is assigned a unique 8-bit code.
- ❑ Test Device (TD): tests whether the addressed device is ready to send or receive
- ❑ Read Data (RD)
- ❑ Write Data (WD)

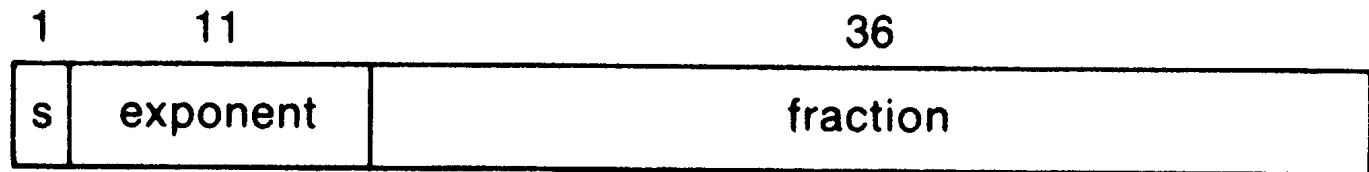
1.3.2 SIC/XE Machine Architecture

- 1 megabytes (1024 KB) in memory
- 3 additional registers, 24 bits in length
 - B 3 Base register; used for addressing
 - S 4 General working register
 - T 5 General working register
- 1 additional register, 48 bits in length
 - F 6 Floating-point accumulator (48 bits)

1.3.2 SIC/XE Machine Architecture

■ Data format

- ❑ 24-bit binary number for integer, 2's complement for negative values
- ❑ 48-bit floating-point data type
- ❑ The exponent is between 0 and 2047
- ❑ $f * 2^{(e-1024)}$
- ❑ 0: set all bits to 0



1.3.2 SIC/XE Machine Architecture

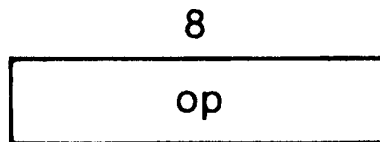
■ Instruction formats

- ❑ Relative addressing (相對位址) - **format 3 (e=0)**
- ❑ Extend the address to 20 bits (絕對位址) - **format 4 (e=1)**
- ❑ Don't refer memory at all - **formats 1 and 2**

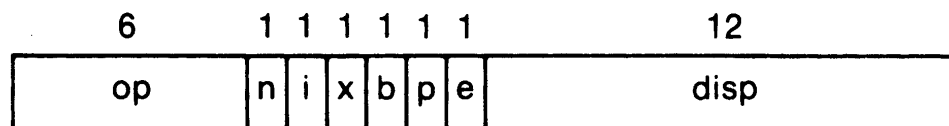
Format 2 (2 bytes):



Format 1 (1 byte):



Format 3 (3 bytes):



Format 4 (4 bytes):



1.3.2 SIC/XE Machine Architecture

■ Addressing modes

□ n i x b p e

□ Simple $n=0, i=0$ (SIC) or $n=1, i=1, TA=disp$

□ Immediate $n=0, i=1$ **Disp=Value**

□ Indirect $n=1, i=0$ $TA=(Operand)=(TA_1)$

□ Base relative $b=1, p=0$ **$TA=(B)+disp$**

$0 \leq disp \leq 4095$

□ PC relative $b=0, p=1$ **$TA=(PC)+disp$**

$-2048 \leq disp \leq 2047$

Mode	Indication	Target address calculation	
Base relative	$b = 1, p = 0$	$TA = (B) + disp$	$(0 \leq disp \leq 4095)$
Program-counter relative	$b = 0, p = 1$	$TA = (PC) + disp$	$(-2048 \leq disp \leq 2047)$

1.3.2 SIC/XE Machine Architecture

■ Addressing mode

- ❑ **Direct** $b=0, p=0$ $TA=disp$
- ❑ **Index** $x=1$ $TA_{new}=TA_{old}+(X)$
- ❑ **Index+Base relative** $x=1, b=1, p=0$ $TA=(B)+disp+(X)$
- ❑ **Index+PC relative** $x=1, b=0, p=1$ $TA=(PC)+disp+(X)$
- ❑ **Index+Direct** $x=1, b=0, p=0$ $TA=disp+(X)$
- ❑ **Format 4** $e=1$

■ Appendix and Fig. 1.1 Example

Figure 1.1

(B) = 006000

(PC) = 003000

(X) = 000090

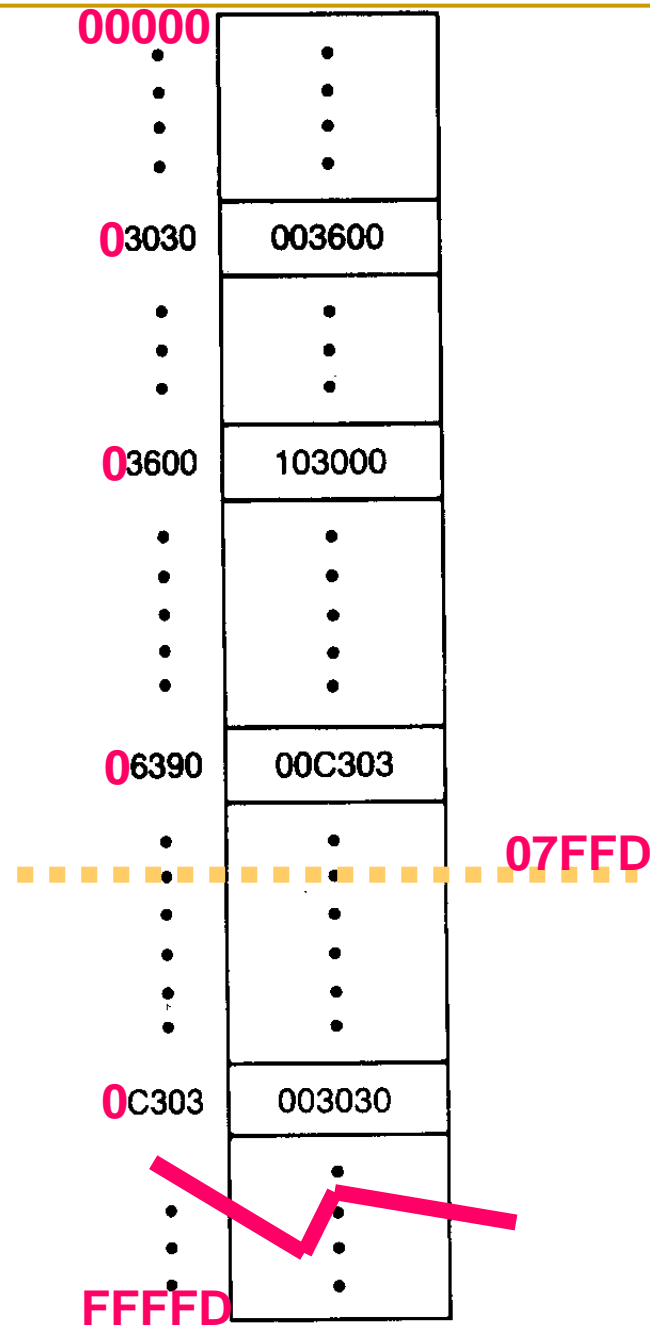
■ Memory address

□ 00000

(0000 0000 0000 0000 0000)

□ ~FFFF (Byte)

(1111 1111 1111 1111 1111)



(a)

(B) = 006000

(PC) = 003000

(X) = 000090

Machine instruction

Hex	Binary								Target address	Value loaded into register A
	op	n	i	x	b	p	e	disp/address		
032600	000000	1	1	0	0	1	0	0110 0000 0000	03600	103000
030300	000000	1	1	1	1	0	0	0011 0000 0000	06390	00C303
022030	000000	1	0	0	0	1	0	0000 0011 0000	03030	103000
010030	000000	0	1	0	0	0	0	0000 0011 0000	30	000030
003600	000000	0	0	0	0	1	1	0110 0000 0000	3600	103000
0310C303	000000	1	1	0	0	0	1	0000 1100 0011 0000 0011	0C303	003030

(b)

1.3.2 SIC/XE Machine Architecture

■ Instruction set

- ❑ Format 1, 2, 3, or 4
- ❑ Load and store registers (LDB, STB, etc.)
- ❑ Floating-point arithmetic operations (ADD_F, SUB_F, MUL_F, DIV_F)
- ❑ Register-to-register arithmetic operations (ADD_R, SUB_R, MUL_R, DIV_R)
- ❑ A special supervisor call instruction (SVC) is provided

■ I/O

- ❑ 1 byte at a time, TD, RD, and WD
- ❑ SIO, TIO, and HIO are used to start, test, and halt the operation of I/O channels.

1.3.3 SIC Programming Examples

■ Sample data movement operations

- No **memory-to-memory move** instructions (Fig. 1.2)



LDA FIVE
STA ALPHA
LDCH CHARZ
STCH C1
.
.
.

LOAD CONSTANT 5 INTO REGISTER A
STORE IN ALPHA
LOAD CHARACTER 'Z' INTO REGISTER A
STORE IN CHARACTER VARIABLE C1

ALPHA	<u>RESW</u>	1	ONE-WORD VARIABLE
FIVE	WORD	5	ONE-WORD CONSTANT
CHARZ	BYTE	C'Z'	ONE-BYTE CONSTANT
C1	<u>RESB</u>	1	ONE-BYTE VARIABLE

1.3.3 SIC Programming Examples

	LDA	#5		LOAD VALUE 5 INTO REGISTER A
	STA	ALPHA		STORE IN ALPHA
	LDA	#90	00005A	LOAD ASCII CODE FOR 'Z' INTO REG A
	STCH	C1		STORE IN CHARACTER VARIABLE C1
	.			
	.			
	.			
ALPHA	RESW	1		ONE-WORD VARIABLE
C1	RESB	1		ONE-BYTE VARIABLE

(b)

Figure 1.2 Sample data movement operations for (a) SIC and (b) SIC/XE.

1.3.3 SIC Programming Examples

■ Sample arithmetic operations

□ (ALPHA+INCR-1) assign to BETA (Fig. 1.3)

□ (GAMMA+INCR-1) assign to DELTA

LDA	ALPHA	LOAD ALPHA INTO REGISTER A
ADD	INCR	ADD THE VALUE OF INCR
SUB	ONE	SUBTRACT 1
STA	BETA	STORE IN BETA
LDA	GAMMA	LOAD GAMMA INTO REGISTER A
ADD	INCR	ADD THE VALUE OF INCR
SUB	ONE	SUBTRACT 1
STA	DELTA	STORE IN DELTA
.		
.		
.		
ONE	WORD	1
		ONE-WORD CONSTANT
.		ONE-WORD VARIABLES
ALPHA	RESW	1
BETA	RESW	1
GAMMA	RESW	1
DELTA	RESW	1
INCR	RESW	1

1.3.3 SIC Programming Examples

■ SIC/XE example

```
LDS      INCR
LDA      ALPHA
ADDR     S,A
SUB      #1
STA      BETA
LDA      GAMMA
ADDR     S,A
SUB      #1
STA      DELTA
```

```
.
.
.
```

```
LOAD VALUE OF INCR INTO REGISTER S
LOAD ALPHA INTO REGISTER A
ADD THE VALUE OF INCR
SUBTRACT 1
STORE IN BETA
LOAD GAMMA INTO REGISTER A
ADD THE VALUE OF INCR
SUBTRACT 1
STORE IN DELTA
```

ONE WORD VARIABLES

```
.
ALPHA    RESW      1
BETA     RESW      1
GAMMA    RESW      1
DELTA    RESW      1
INCR     RESW      1
```

1.3.3 SIC Programming Examples

■ String copy - SIC example

	LDX	ZERO	INITIALIZE INDEX REGISTER TO 0
MOVECH	LDCH	STR1,X	LOAD CHARACTER FROM STR1 INTO REG A
	STCH	STR2,X	STORE CHARACTER INTO STR2
	TIX	ELEVEN	ADD 1 TO INDEX, COMPARE RESULT TO 11
	JLT	MOVECH	LOOP IF INDEX IS LESS THAN 11
	.		
	.		
	.		
STR1	BYTE	C'TEST STRING'	11-BYTE STRING CONSTANT
STR2	RESB	11	11-BYTE VARIABLE
.			ONE-WORD CONSTANTS
ZERO	WORD	0	
ELEVEN	WORD	11	

1.3.3 SIC Programming Examples

■ String copy - SIC/XE example

	LDT	#11	INITIALIZE REGISTER T TO 11
	LDX	#0	INITIALIZE INDEX REGISTER TO 0
MOVECH	LDCH	STR1,X	LOAD CHARACTER FROM STR1 INTO REG A
	STCH	STR2,X	STORE CHARACTER INTO STR2
	TIXR	T	ADD 1 TO INDEX, COMPARE RESULT TO 11
	JLT	MOVECH	LOOP IF INDEX IS LESS THAN 11
	.		
	.		
	.		
STR1	BYTE	C'TEST STRING'	11-BYTE STRING CONSTANT
STR2	RESB	11	11-BYTE VARIABLE

1.3.3 SIC Programming Examples

	LDA	ZERO	INITIALIZE INDEX VALUE TO 0
	STA	INDEX	
ADDLP	LDX	INDEX	LOAD INDEX VALUE INTO REGISTER X
	LDA	ALPHA,X	LOAD WORD FROM ALPHA INTO REGISTER A
	ADD	BETA,X	ADD WORD FROM BETA
	STA	GAMMA,X	STORE THE RESULT IN A WORD IN GAMMA
	LDA	INDEX	ADD 3 TO INDEX VALUE
	ADD	THREE	
	STA	INDEX	
	COMP	K300	COMPARE NEW INDEX VALUE TO 300
	JLT	ADDLP	LOOP IF INDEX IS LESS THAN 300
	.		
	.		
	.		
INDEX	RESW	1	ONE-WORD VARIABLE FOR INDEX VALUE
.			ARRAY VARIABLES--100 WORDS EACH
ALPHA	RESW	100	
BETA	RESW	100	
GAMMA	RESW	100	
.			ONE-WORD CONSTANTS
ZERO	WORD	0	
K300	WORD	300	
THREE	WORD	3	

1.3.3 SIC Programming Examples

	LDS	#3	INITIALIZE REGISTER S TO 3
	LDT	#300	INITIALIZE REGISTER T TO 300
	LDX	#0	INITIALIZE INDEX REGISTER TO 0
ADDLP	LDA	ALPHA,X	LOAD WORD FROM ALPHA INTO REGISTER A
	ADD	BETA,X	ADD WORD FROM BETA
	STA	GAMMA,X	STORE THE RESULT IN A WORD IN GAMMA
	ADDR	S,X	ADD 3 TO INDEX VALUE
	COMPR	X,T	COMPARE NEW INDEX VALUE TO 300
	JLT	ADDLP	LOOP IF INDEX VALUE IS LESS THAN 300
	.		
	.		
	.		
.			ARRAY VARIABLES--100 WORDS EACH
ALPHA	RESW	100	
BETA	RESW	100	
GAMMA	RESW	100	

(b)

Figure 1.5 Sample indexing and looping operations for (a) SIC and (b) SIC/XE.

1.3.3 SIC Programming Examples

INLOOP	TD	INDEV	TEST INPUT DEVICE
	JEQ	INLOOP	LOOP UNTIL DEVICE IS READY
	RD	INDEV	READ ONE BYTE INTO REGISTER A
	STCH	DATA	STORE BYTE THAT WAS READ
	.		
	.		
	.		
OUTLP	TD	OUTDEV	TEST OUTPUT DEVICE
	JEQ	OUTLP	LOOP UNTIL DEVICE IS READY
	LDCH	DATA	LOAD DATA BYTE INTO REGISTER A
	WD	OUTDEV	WRITE ONE BYTE TO OUTPUT DEVICE
	.		
	.		
	.		
INDEV	BYTE	X'F1'	INPUT DEVICE NUMBER
OUTDEV	BYTE	X'05'	OUTPUT DEVICE NUMBER
DATA	RESB	1	ONE-BYTE VARIABLE

Figure 1.6 Sample input and output operations for SIC.

1.3.3 SIC Programming Examples

	JSUB	READ	CALL READ SUBROUTINE
	.		
	.		
	.		
.			SUBROUTINE TO READ 100-BYTE RECORD
READ	LDX	ZERO	INITIALIZE INDEX REGISTER TO 0
RLOOP	TD	INDEV	TEST INPUT DEVICE
	JEQ	RLOOP	LOOP IF DEVICE IS BUSY
	RD	INDEV	READ ONE BYTE INTO REGISTER A
	STCH	RECORD,X	STORE DATA BYTE INTO RECORD
	TIX	K100	ADD 1 TO INDEX AND COMPARE TO 100
	JLT	RLOOP	LOOP IF INDEX IS LESS THAN 100
	RSUB		EXIT FROM SUBROUTINE
	.		
	.		
	.		
INDEV	BYTE	X'F1'	INPUT DEVICE NUMBER
RECORD	RESB	100	100-BYTE BUFFER FOR INPUT RECORD
.			ONE-WORD CONSTANTS
ZERO	WORD	0	
K100	WORD	100	

1.3.3 SIC Programming Examples

	JSUB	READ	CALL READ SUBROUTINE
	.		
	.		
	.		
.			SUBROUTINE TO READ 100-BYTE RECORD
READ	LDX	#0	INITIALIZE INDEX REGISTER TO 0
	LDT	#100	INITIALIZE REGISTER T TO 100
RLOOP	TD	INDEV	TEST INPUT DEVICE
	JEQ	RLOOP	LOOP IF DEVICE IS BUSY
	RD	INDEV	READ ONE BYTE INTO REGISTER A
	STCH	RECORD,X	STORE DATA BYTE INTO RECORD
	TIXR	T	ADD 1 TO INDEX AND COMPARE TO 100
	JLT	RLOOP	LOOP IF INDEX IS LESS THAN 100
	RSUB		EXIT FROM SUBROUTINE
	.		
	.		
	.		
INDEV	BYTE	X'F1'	INPUT DEVICE NUMBER
RECORD	RESB	100	100-BYTE BUFFER FOR INPUT RECORD

Traditional (CISC) Machines

- **Complex Instruction Set Computers (CISC)**
 - ❑ complicated instruction set
 - ❑ different instruction formats and lengths
 - ❑ many different addressing modes
 - ❑ e.g. VAX or PDP-11 from DEC
 - ❑ e.g. Intel x86 family
- **Reduced Instruction Set Computer (RISC)**

RISC Machines

■ RISC system

□ Instruction

- standard, fixed instruction format
- single-cycle execution of most instructions
- memory access is available only for load and store instruction
- other instructions are register-to-register operations
- a small number of machine instructions, and instruction format
- Instructional-level parallelism

□ A large number of general-purpose registers

□ A small number of addressing modes

□ Three RISC machines

- SPARC family
- PowerPC family
- Cray T3E