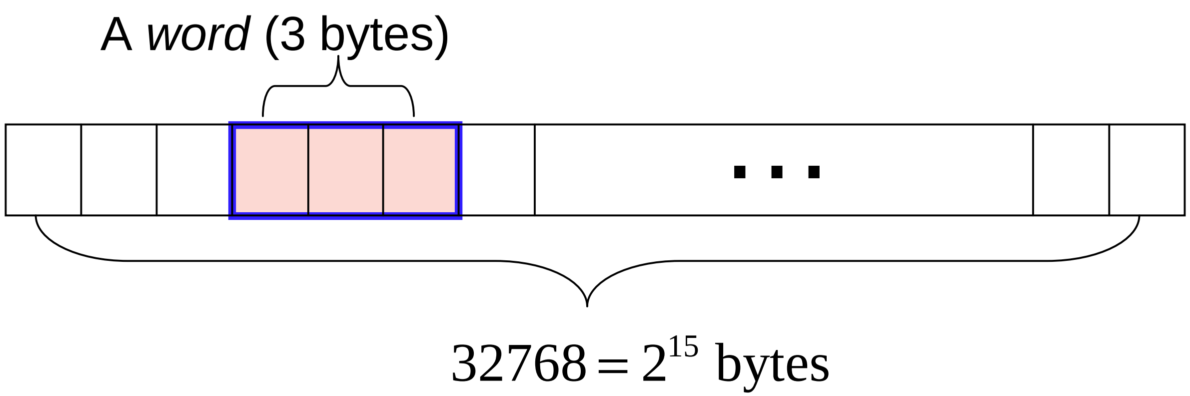
Simplified Instructional Computer (SIC)

- A hypothetical computer that includes the hardware features most often found on real machines
  - SIC standard model
  - SIC/XE
- Upward compatible
  - Programs for SIC can run on SIC/XE

SIC machine architecture

- Memory
  - 8-bit bytes
  - 3 consecutive bytes form a word
    - `Addressed by the lowest number byte`
  - 2^{15} (32768) bytes in the computer memory



SIC machine architecture

- Registers (5 registers / each 24-bits)

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)

- SIC does not have any stack. It uses the linkage register to store the return address.
- It is difficult to write the recursive program. A programmer has to maintain memory for return addresses when he write more than one layer of function call.

SIC machine architecture

- Data formats
  - Characters
    - 8-bit ASCII codes
  - Integers
    - 24-bit binary numbers
    - 2's complement for negative values
      - $-N \Leftrightarrow 2^n - N$
      - e.g., if n = 4, $-1 \Leftrightarrow 2^4 - 1 = (1111)_2$.
  - No floating-point numbers (exist in SIC/XE)

SIC machine architecture

- Instruction formats
  - 24-bits format

opcode (8)	x	address (15)

  - Note that the memory size of SIC is 2^{15} bytes
  - X is to indicate index-address mode
- Addressing modes

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

SIC machine architecture

- Instruction set
  - Load and store instruction
    - LDA, LDX, STA, STX
    - Ex: LDA ALPHA ⇔ (A) ← (ALPHA)
      STA ALPHA ⇔ (ALPHA) ← (A)
  - Arithmetic instruction
    - involve register A and a word in memory
    - ADD, SUB, MUL, DIV
    - Ex: ADD ALPHA ⇔ (A) ← (A) + (ALPHA)
  - Comparison instruction
    - involves register A and a word in memory
    - save result in the condition code (CC) of SW
    - COMP
    - Ex: COMP ALPHA ⇔ CC ←(<,+,>) of (A)?(ALPHA)

SIC machine architecture

- Instruction set (Cont.)
  - Conditional jump instructions
    - according to CC
    - JLE, JEQ, JGT
      - test CC and jump accordingly
  - Subroutine linkage instructions
    - JSUB
      - jumps and places the return address in register L
    - RSUB
      - returns to the address in L

SIC machine architecture

- Input and output
  - Input and output 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-bits code
  - Three I/O instructions
    - The Test Device (TD) instruction
      - tests whether the addressed device is ready to send or receive a byte of data
      - CC : <   : ready
      - CC : =   : busy
    - Read Data (RD)
    - Write Data (WD)

Simple I/O example for SIC

- Page 19, Figure 1.6

cc : = denotes device busy

```
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’     OUTPOUT DEVICE NUMBER
DATA     RESB    1         ONE-BYTE VARIABLE
```

Programming examples - Data movement

- Page 13, Figure 1.2 (a)

```
        LDA     FIVE    LOAD CONSTANT 5 INTO REGISTER A
        STA     ALPHA   STORE IN ALPHA
        LDCH    CHARZ   LOAD CHARACTER 'Z' INTO REGISTER A
        STCH    C1      STORE IN CHARACTER VARIABLE C1
        .
        .
ALPHA   RESW    1       ONE-WORD VARIABLE
FIVE    WORD    5       ONE-WORD CONSTANT
CHARZ   BYTE    C'Z'    ONE-BYTE CONSTANT
C1      RESB    1       ONE-BYTE VARIABLE
```

Programming examples - Arithmetic

- Page 15, Figure 1.3 (a)

```
        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
```

Programming examples -Looping and indexing

- Page 16, Figure 1.4 (a)

```
        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
```

Programming examples - Indexing and looping

- Page 17, Figure 1.5 (a)

```
        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
```

Programming examples - Subroutine call and record input

- Page 20, Figure 1.7 (a)

```
          JSUB    READ        CALL READ SUBROUTINE
          .
.                             SUBROUTINE TO READ 100-BYTE RCORD
READ      LDX     ZERO        INITAILIZE 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
```

Define storage

- WORD/BYTE
  - Reserve one word/byte of storage
- RESW/RESB
  - Reserve one or more words/bytes of storage
- Example

ALPHA	RESW	1
FIVE	WORD	5
CHARZ	BYTE	C`Z’
C1	RESB	1

Special symbols (SIC & SIC/XE)

- # : immediate addressing
- @ : indirect addressing
- + : format 4
- * : the current value of PC
- C` ‘: character string
- op m, x : x denotes the index addressing

SIC/XE machine architecture

- Memory
  - Maximum memory available on a SIC/XE system is 1 megabyte (2^{20} bytes)
  - Instruction format and addressing modes are changed
- Register (Additional registers)

Mnemonic	Number	Special use
B	3	Base register; used for addressing
S	4	General working register-no special use
T	5	General working register-no special use
F	6	Floating-point accumulator (48bits)

  - Registers S and T are only for storing data. They can not use for accumulator
    - Ex: ADDR S, A $A \leftarrow A+S$
      COMPR X, T

SIC/XE machine architecture

- Data formats
  - There is a 48-bit floating-point data type

1	11	36
s	exponent	fraction

  - sign bit s (0: +, 1: -)
  - fraction f: a value between 0 and 1
  - exponent e: unsigned binary number between 0 and 2047
  - value: $s * f * 2^{(e-1024)}$
  - Ex: $5 = 2^2+2^0=(2^{-1}+2^{-3})*2^3= (2^{-1}+2^{-3})*2^{1027-1024}$

    0,10000000011,1010000….0

SIC/XE machine architecture

- Instruction formats
  - Since the memory used by SIC/XE may be 2^{20} bytes, the instruction format of SIC is not enough.
    - Solutions
      - Use relative addressing
      - Extend the address field to 20 bits

8	1	15
opcode	x	address

  - SIC/XE instruction formats

Format								
Format 1 (1 byte)	op (8)							
Format 2 (2 byte)	op (8)	r1 (4)	r2 (4)					
Format 3 (3 byte)	op (6)	n	i	x	b	p	e	disp (12)
Format 4 (4 byte)	op (6)	n	i	x	b	p	e	address (20)

e=0: format 3, e=1: format 4

SIC/XE machine architecture

- Addressing modes
  - New relative addressing modes for format 3

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

  - When base relative mode is used, disp is a 12-bits unsigned integer
  - When program-counter relative mode is used, disp is a 12-bits signed integer
    - 2's complement
  - Direct addressing for formats 3 and 4 if b=p=0
  - These two addressing mode can combine with index addressing if x=1

SIC/XE machine architecture

- Addressing modes
  - Bits x,b,p,e: how to calculate the target address
    - relative, direct, and indexed addressing modes
  - Bits i and n: how to use the target address (TA)

Mode	Indication	Operand value
Immediate addressing	i=1, n=0	**TA**: TA is used as the operand value, no memory reference
Indirect addressing	i=0, n=1	**((TA))**: The word at the TA is fetched. Value of TA is taken as the address of the operand value
Simple addressing	i=0, n=0	Standard SIC
	i=1, n=1	**(TA)**:TA is taken as the address of the operand value

Addressing mode example

(B)=006000
(PC)=003000
(X)=000090

.	.
.	.
.	.
3030	003600
.	.
.	.
.	.
3600	103000
.	.
.	.
.	.
6390	00C303
.	.
.	.
.	.
C303	003030
.	.
.	.
.	.

Addressing mode example

	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	3600	103000
03C300	000000	1	1	1	1	0	0	0011 0000 0000	6390	00C303
022030	000000	1	0	0	0	1	0	0000 0011 0000	3030	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	C303	003030

Addressing mode summary

Addressing type	Flag bits n i x b p e	Assembler lenguage notation	Calculation of target address TA	Operand	Notes
Simple	1 1 0 0 0 0	op c	disp	(TA)	D
	1 1 0 0 0 1	+op m	addr	(TA)	4 D
	1 1 0 0 1 0	op m	(PC)+disp	(TA)	A
	1 1 0 1 0 0	op m	(B)+disp	(TA)	A
	1 1 1 0 0 0	op c,X	disp+(X)	(TA)	D
	1 1 1 0 0 1	+op m,X	addr+(X)	(TA)	4 D
	1 1 1 0 1 0	op m,X	(PC)+disp+(X)	(TA)	A
	1 1 1 1 0 0	op m,X	(B)+disp+(X)	(TA)	A
	0 0 0 - - -	op m	b/p/e/disp	(TA)	D S
	0 0 1 - - -	op m,X	b/p/e/disp+(X)	(TA)	D S
Indirect	1 0 0 0 0 0	op @c	disp	((TA))	D
	1 0 0 0 0 1	+op @m	addr	((TA))	4 D
	1 0 0 0 1 0	op @m	(PC)+disp	((TA))	A
	1 0 0 1 0 0	op @m	(B)+disp	((TA))	A
Immediate	0 1 0 0 0 0	op #c	disp	TA	D
	0 1 0 0 0 1	+op #m	addr	TA	4 D
	0 1 0 0 1 0	op #m	(PC)+disp	TA	A
	0 1 0 1 0 0	op #m	(B)+disp	TA	A

SIC/XE machine architecture

- Instruction set
  - Standard SIC's instruction
  - Load and store registers (B, S, T, F)
    - `LDB, STB, …`
  - Floating-point arithmetic operations
    - `ADDF, SUBF, MULF, DIVF`
  - Register-register arithmetic operations
    - `ADDR, SUBR, MULR, DIVR`
  - Register move operations
    - `RMO`
  - Supervisor call (SVC)
    - `generates an interrupt for OS (Chap 6)`
- Input/Output
  - SIO, TIO, HIO: start, test, halt the operation of I/O device

SIC/XE machine architecture

- Instruction set
  - Refer to Appendix A for all instructions (Page 496)
  - Notations for appendix
    - A ← (m..m+2): move word begin at m to A
    - P: privileged instruction
    - X: instruction available only in SIC/XE
    - C: condition code CC

Programming examples (SIC/XE) - Data movement

- Page 13, Figure 1.2 (b)

```
        LDA     #5      LOAD VALUE 5 INTO REGISTER A
        STA     ALPHA   STORE IN ALPHA
        LDA     #90     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
```

Programming examples (SIC/XE) - Arithmetic

- Page 15, Figure 1.3 (b)

```
         LDS     INCR      LOAD VALUE OF INCR INTO REGISTER S
         LDA     ALPHA     LOAD ALPHA INTO REGISTER A
         ADDR    S,A       ADD THE VALUE OF INCR
         SUB     #1        SUBTRACT 1
         STA     BETA      STORE IN BETA
         LDA     GAMMA     LOAD GAMMA INTO REGISTER A
         ADDR    S,A       ADD THE VALUE OF INCR
         SUB     #1        SUBTRACT 1
         STA     DELTA     STORE IN DELTA
         .
         .
.                          ONE WORD VARIABLES
ALPHA    RESW    1
BETA     RESW    1
GAMMA    RESW    1
DELTA    RESW    1
INCR     RESW    1
```

Programming examples (SIC/XE) -Looping and indexing

- Page 16, Figure 1.4 (b)

```
MOVECH   LDT    #11              INITIALIZE REGISTER T TO 11
         LDX    #0               INITIALIZE INDEX REGISTER TO 0
         LDCH   STR1,X           LOAD CHARACTER FROM STR1 INTO REG A
         STCH   STR2,X           SOTRE 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
```

Programming examples (SIC/XE) - Indexing and looping

- Page 17, Figure 1.5 (b)

```
            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
```

Programming examples (SIC/XE) - Subroutine call and record input

- Page 20, Figure 1.7 (a)

```
                JSUB    READ        CALL READ SUBROUTINE
                .
                .
.                                   SUBROUTINE TO READ 100-BYTE RECORD
READ            LDX     #0          INITIALIZE INDEX REGISTER TO 0
RLOOP           LDT     #100        INITIALIZE REGISTER T TO 100
                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
```