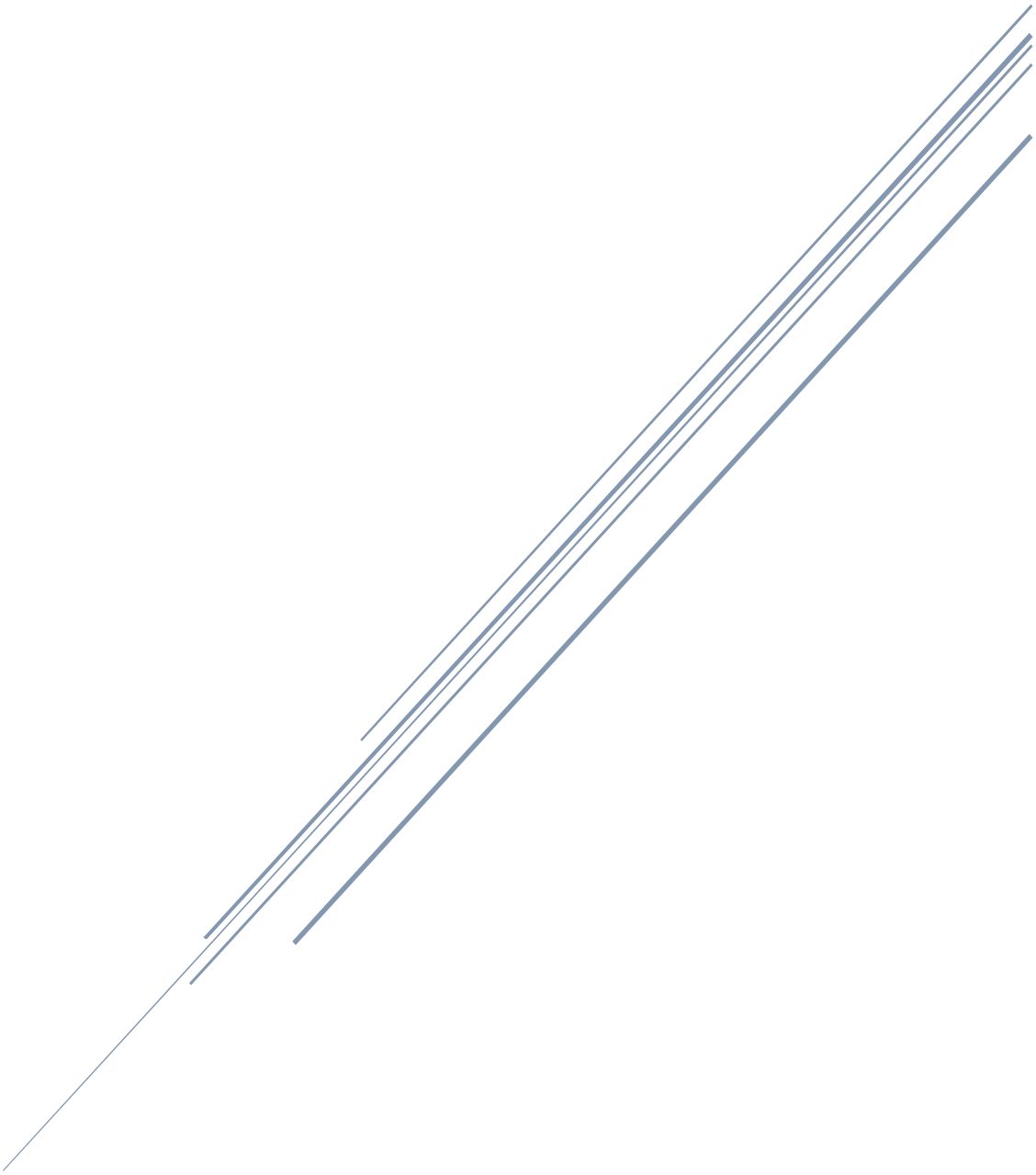
NOTES ON SYSTEM SOFTWARE

Mrinal Paul



Department of Computer Science
Assam University, Silchar

Unit – I

❖ SYSTEM SOFTWARE:

System software consists of a variety of programs that support the operation of a computer.

- It is a set of programs to perform a variety of system functions as file editing, resource management, I/O management and storage management.
- The characteristic in which system software differs from application software is machine dependency.
- An application program is primarily concerned with the solution of some problem, using the computer as a tool.
- System programs on the other hand are intended to support the operation and use of the computer itself, rather than any particular application.
- For this reason, they are usually related to the architecture of the machine on which they are run.
- For example, assemblers translate mnemonic instructions into machine code. The instruction formats, addressing modes are of direct concern in assembler design.
- There are some aspects of system software that do not directly depend upon the type of computing system being supported. These are known as machine-independent features.
- For example, the general design and logic of an assembler is basically the same on most computers.

❖ TYPES OF SYSTEM SOFTWARE:

1. Operating system
2. Language translators
    a) Compilers
    b) Interpreters
    c) Assemblers
    d) Preprocessors
3. Loaders
4. Linkers
5. Macro processors

❖ Machine Structure:

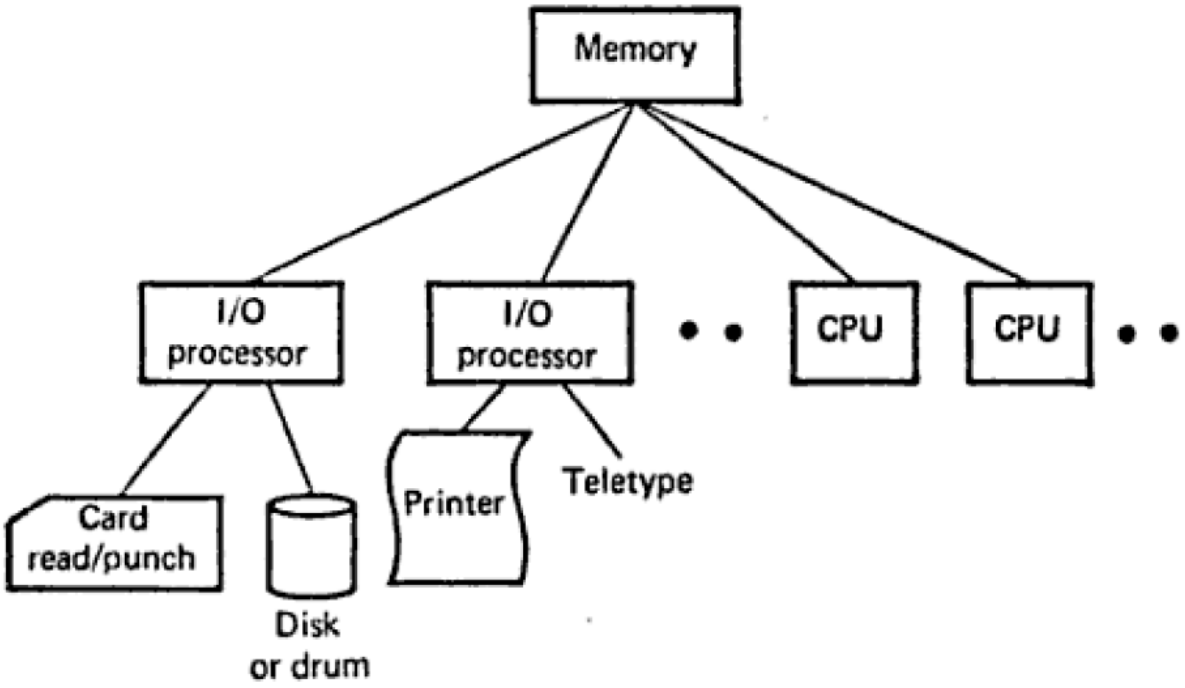


Fig: General hardware organization of a computer system

Memory is the device where information is stored. Processors are the devices that operate on this information. One may view information was stored as being stored in the form of ones and zeros. Each one or zero is separate binary digit called a bit. Bits are typically grouped in units that are called words, characters or bytes. Memory location are specified by address, where each address identifies a specific byte, word or character.

The component of a word may be interpreted as data (values to be operated) or instruction (operation to be performed). A processor is a device that performs a sequence of operation specified by instruction in memory. A program (or procedure) is a sequence of instructions.

Memory may be thought of as mailboxes consisting groups of ones and zeros. Below we depict a series of memory location whose address are 10,000 through 10.002.

Address	Contents
10,000	0000 0000 0000 0001
10,001	0011 0000 0000 0000
10,002	0000 0000 0000 0100

❖ THE SIMPLIFIED INSTRUCTIONAL COMPUTER (SIC):

It is similar to a typical microcomputer. It comes in two versions:

- The standard model
- XE version

SIC Machine Structure:

Memory:

- It consists of bytes (8 bits), words (24 bits which are consecutive 3 bytes) addressed by the location of their lowest numbered byte.
- There are totally 32,768 bytes in memory.

Registers:

There are 5 registers namely

1. Accumulator (A)
2. Index Register(X)
3. Linkage Register(L)
4. Program Counter(PC)
5. Status Word (SW).
    - Accumulator is a special purpose register used for arithmetic operations.
    - Index register is used for addressing.
    - Linkage register stores the return address of the jump of subroutine instructions (JSUB).
    - Program counter contains the address of the current instructions being executed.
    - Status word contains a variety of information including the condition code.

Data formats:

- Integers are stored as 24-bit binary numbers: 2's complement representation is used for negative values characters are stored using their 8 bit ASCII codes.
- They do not support floating – point data items.

Instruction formats:

All machine instructions are of 24-bits wide

Opcode (8)	X (1)	Address (15)

- X-flag bit that is used to indicate indexed-addressing mode.

Addressing modes:

- Two types of addressing are available namely,
    1. Direct addressing mode
    2. Indexed addressing mode or indirect addressing mode

Mode	Indication	Target Address calculation
Direct	X=0	TA=Address
Indexe	X=1	TA=Address + (X)
d		

- Where(x) represents the contents of the index register(x)

Instruction set:

It includes instructions like:

1. Data movement instruction Ex: LDA, LDX, STA, STX.
2. Arithmetic operating instructions Ex: ADD, SUB, MUL, DIB.
   This involves register A and a word in memory, with the result being left in the register.
3. Branching instructions Ex: JLT, JEQ, TGT.
4. Subroutine linkage instructions Ex: JSUB, RSUB.

Input and Output:

- I/O is performed by transferring one byte at a time to or from the rightmost 8 bits of register A.
- Each device is assigned a unique 8-bit code.
- There are 3 I/O instructions,
  1. The Test Device (TD) instructions tests whether the addressed device is ready to send or receive a byte of data.
  2. A program must wait until the device is ready, and then execute a Read Data (RD) or Write Data (WD).
  3. The sequence must be repeated for each byte of data to be read or written.

❖ SIC/XE ARCHITECTURE & SYSTEM SPECIFICATION

Memory:

1 word = 24 bits (3 8-bit bytes)

Total (SIC/XE) = 2^{20} (1,048,576) bytes (1Mbyte)

Registers:

10 x 24 bit registers

Data Format:

- Integers are stored in 24 bit, 2's complement format
- Characters are stored in 8-bit ASCII format

MNEMONIC	Register	Purpose
A	0	Accumulator
X	1	Index register
L	2	Linkage register (JSUB/RSUB)
B	3	Base register
S	4	General register
T	5	General register
F	6	Floating Point Accumulator (48 bits)
PC	8	Program Counter (PC)

SW	9	Status Word (includes Condition Code, CC)

- Floating point is stored in 48 bit signed-exponent-fraction format:

s	exponent {11}	fraction {36}

- The fraction is represented as a 36 bit number and has value between 0 and 1.
- The exponent is represented as a 11 bit unsigned binary number between 0 and 2047.
- The sign of the floating point number is indicated by s : 0=positive, 1=negative.
- Therefore, the absolute floating point number value is: $f*2^{(e-1024)}$

Instruction Format:

- There are 4 different instruction formats available:

Format 1 (1 byte):

op {8}

Format 2 (2 bytes):

Op {8}	R1 {4}	R2 {4}

Format 3 (3 bytes):

Op {6}	n	i	x	b	p	e	displacement {12}

Format 4 (4 bytes):

Op {6}	n	i	x	b	p	e	address {20}

Formats 3 & 4 introduce addressing mode flag bits:

- n=0 & i=1
  Immediate addressing - TA is used as an operand value (no memory reference)
- n=1 & i=0
- Indirect addressing - word at TA (in memory) is fetched & used as an address to fetch the operand from
- n=0 & i=0
  Simple addressing TA is the location of the operand
- n=1 & i=1
  Simple addressing same as n=0 & i=0

Flag x:

x=1 Indexed addressing add contents of X register to TA, calculation Flag b & p (Format 3 only):

- b=0 & p=0

Direct addressing displacement/address field containsTA (Format 4 always uses direct addressing)

- b=0 & p=1
  PC relative addressing - TA=(PC)+disp (-2048<=disp<=2047)*
- b=1 & p=0
  Base relative addressing - TA=(B)+disp (0<=disp<=4095)**

Flag e:

e=0 use Format 3

e=1 use Format 4

Instructions:

SIC provides 26 instructions, SIC/XE provides an additional 33 instructions (59 total)

SIC/XE has 9 categories of instructions:

- Load/store registers (LDA, LDX, LDCH, STA, STX, STCH, etc.)
- integer arithmetic operations (ADD, SUB, MUL, DIV) these will use register A and a word in memory, results are placed into register A
- compare (COMP) compares contents of register A with a word in memory and sets CC (Condition Code) to <, >, or =
- conditional jumps (JLT, JEQ, JGT) - jumps according to setting of CC
- subroutine linkage (JSUB, RSUB) - jumps into/returns from subroutine using register L
- input & output control (RD, WD, TD) - see next section
- floating point arithmetic operations (ADDF, SUBF, MULF, DIVF)
- register manipulation, operands-from-registers, and register-to-register arithmetics (RMO, RSUB, COMPR, SHIFTR, SHIFTL, ADDR, SUBR, MULR, DIVR, etc)

Input and Output (I/O):

- 2^8 (256) I/O devices may be attached, each has its own unique 8-bit address
- 1 byte of data will be transferred to/from the rightmost 8 bits of register A

Three I/O instructions are provided:

- RD Read Data from I/O device into A
- WD Write data to I/O device from A
- TD Test Device determines if addressed I/O device is ready to send/receive a byte of data. The CC (Condition Code) gets set with results from this test:

device is ready to send/receive

device isn't ready

SIC/XE Has capability for programmed I/O (I/O device may input/output data while CPU does other work) - 3 additional instructions are provided:

- SIO Start I/O
- HIO Halt I/O
- TIO Test I/O

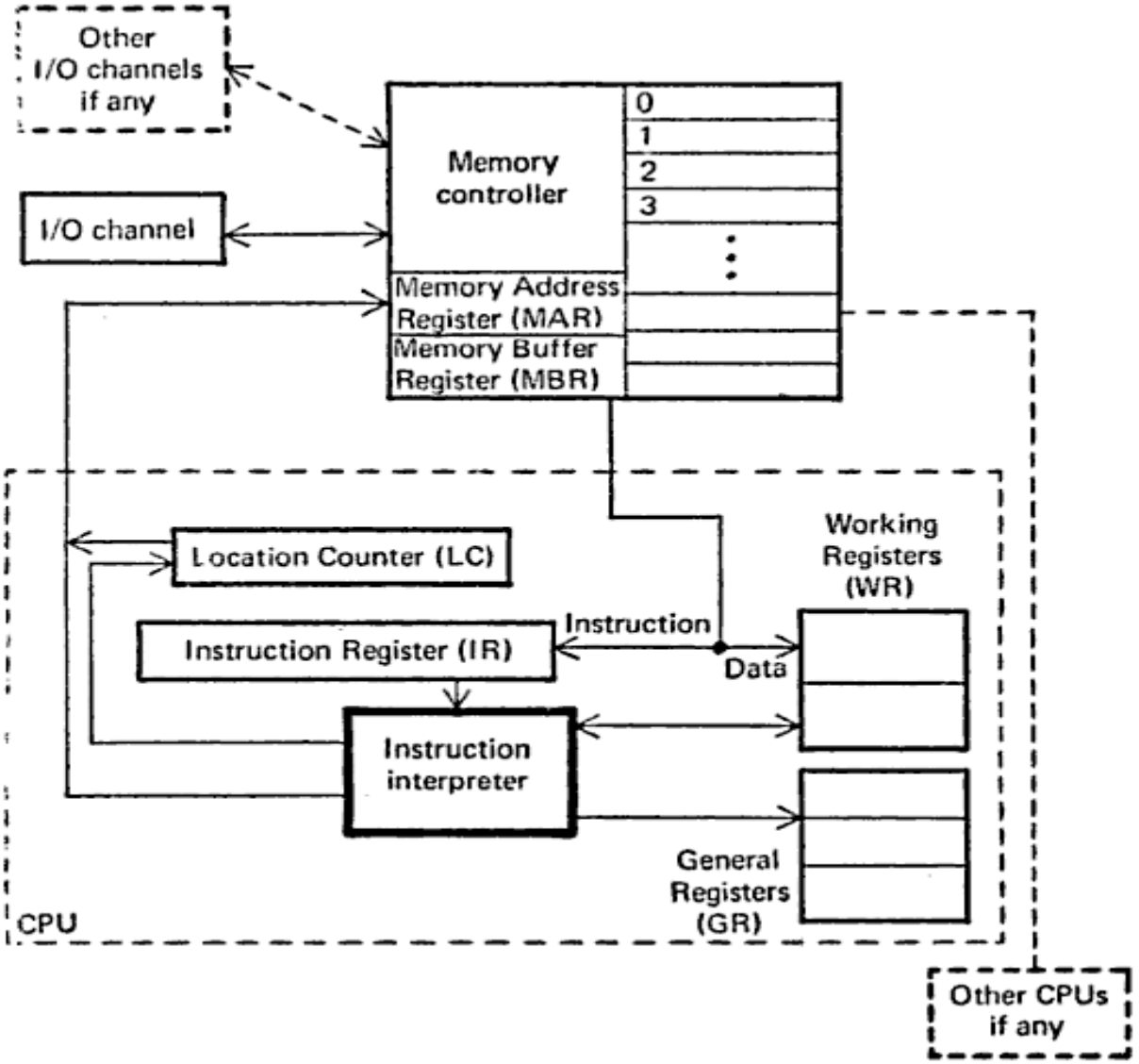
❖ SIC/XE Addressing Modes:

Addressing Type	Flag Bits						Notation	Calculation of Target Address	Operand	Notes
	n	i	x	b	p	e				
Simple	1	1	0	0	0	0	op c	disp	(TA)	Direct-addressing Instruction
	1	1	0	0	0	1	+op m	addr	(TA)	Format 4 & Direct-addressing Instruction
	1	1	0	0	1	0	op m	(PC) + disp	(TA)	Assember selects either base-relative or program-counter relative mode
	1	1	0	1	0	0	op m	(B) + disp	(TA)	Assember selects either base-relative or program-counter relative mode
	1	1	1	0	0	0	op c,X	disp + (X)	(TA)	Direct-addressing Instruction
	1	1	1	0	0	1	+op m,X	addr + (X)	(TA)	Format 4 & Direct-addressing Instruction
	1	1	1	0	1	0	op m,X	(PC) + disp + (X)	(TA)	Assember selects either base-relative or program-counter relative mode
	1	1	1	1	0	0	op m,X	(B) + disp + (X)	(TA)	Assember selects either base-relative or program-counter relative mode

	0	0	0	-	-	-	op m	b/p/e/disp	(TA)	Direct-addressing Instruction; SIC compatible format.
	0	0	1	-	-	-	op m,X	b/p/e/disp + (X)	(TA)	Direct-addressing Instruction; SIC compatible format.
Indirect	1	0	0	0	0	0	op @c	disp	((TA))	Direct-addressing Instruction
	1	0	0	0	0	1	+op @m	addr	((TA))	Format 4 & Direct-addressing Instruction
	1	0	0	0	1	0	op @m	(PC) + disp	((TA))	Assember selects either base-relative or program-counter relative mode
	1	0	0	1	0	0	op @m	(B) + disp	((TA))	Assember selects either base-relative or program-counter relative mode
Immediate	0	1	0	0	0	0	op #c	disp	TA	Direct-addressing Instruction
	0	1	0	0	0	1	op #m	addr	TA	Format 4 & Direct-addressing Instruction
	0	1	0	0	1	0	op #m	(PC) + disp	TA	Assember selects either base-relative or program-counter relative mode
	0	1	0	1	0	0	op #m	(B) + disp	TA	Assember selects either base-relative or program-counter relative mode

❖ General Machine Structure:

The CPU consists of an instruction interpreter, a location counter, an instruction register and various working registers and general registers. The *instruction interpreter* is a group of electrical circuits (*hardware*), that performs the intent of instructions fetched from memory. The *Location Counter* (LC), also called



Program Counter (PC) or *Instruction Counter* (IC), is a hardware memory device which denotes the location of the current instruction being executed. A copy of the current instruction is stored in the *Instruction Register* (IR). The *working registers* are memory devices that serve as "scratch pads" for the instruction interpreter, while the *general registers* are used by the programmer as storage locations and for special functions.

The primary interface between the memory and the CPU is via the memory address register and the memory buffer register. The *Memory Address Register* (MAR) contains the address of the memory location that is to be read from or stored into. The *Memory Buffer Register* (MBR) contains a copy of the designated memory location specified by the MAR after a "read," or the new contents of the memory location prior to a "write." The *memory controller* is hardware that transfers data between the MBR and the core memory location the address of which is in the MAR.

The *I/O channels* may be thought of as separate computers which interpret special instructions for inputting and outputting information from the memory.

❖ Machine Structure – 360 and 370:

1. MEMORY

The basic unit of memory in the 360 is a byte – eight bits of information. That is, each addressable position in memory can contain eight bits of information. There are facilities to operate on contiguous bytes in basic units. The basic units are as follows:

Unit of memory	*Bytes*	*Length in bits*
Byte	1	8
Halfword	2	16
Word	4	32
Doubleword	8	64

A unit of memory consisting of four bits is sometimes referred to as a *nibble*

The size of the 360 memory is up to 2^{24} bytes (about sixteen million).

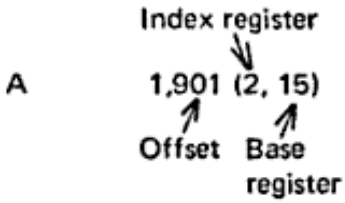
The addressing on the 360 memory may consist of three components. Specifically, the value of an address equals the value of an offset, plus the contents of a base register, plus the contents of an index register. We will give examples of this addressing later.

In general, operations on units of memory are specified by the low-order byte address. For example, when addressing a word (four bytes) the address of the word is that of the low order byte.

2. REGISTERS

The 360 has 16 general-purpose registers consisting of 32 bits each. In addition there are 4 floating-point registers consisting of 64 bits each. It has a 64-bit Program Status Word (PSW) that contains the value of the location counter, protection information, and interrupt status.

The general-purpose registers may be used for various arithmetic and logical operations and as base registers. When the programmer uses them in arithmetic or logical operations, he thinks of these registers as scratch pads to which numbers are added, subtracted, compared, and so forth. When used as base registers, they aid in the formation of the address. Take for example the instruction



It is interpreted as an add instruction. A number is to be added to the contents of register 1.

The location of the number is 901 (*offset*) plus the contents of register 2 (*index*) plus the contents of register 15 (*base*). That is, if those three numbers were added together, the result would be the address of the memory location whose contents we wish to add to the contents of register 1.

3. DATA

The 360 may store several different types of data as is depicted in Figure 2.3. That is, groups of bits stored in memory are interpreted by a 360 processor in several ways. If a 360 interprets the contents of two bytes as an integer (Fig. 2.3a), it interprets the first bit as a sign and the remaining 15 as a binary number (e.g., 0000 0010 0001 1101 is interpreted as the binary number equivalent to the decimal number +541).[3] If a 360 interprets the contents of two bytes as a packed decimal (Fig. 2.3c), it would interpret the first byte as two BCD coded digits, the first four bits of the second byte as a BCD digit, and the last four as a sign (e.g., 0000 0010 0001 1101 is interpreted as the decimal number -021).

0 2 1 Sign

All data and instructions are physically stored as sequences of ones and zeros. Thus, a 16 – bit fixed point half word with decimal value +300 would be stored in binary as '0000 0001 0010 1100'. For convenience, binary numbers are usually written in the hexadecimal (base 16). Note that every hexadecimal digit can be replaced by exactly four binary digits and vice versa.

B'0000 0001 0010 1100' in binary
X' 0 1 2 C in hexadecimal

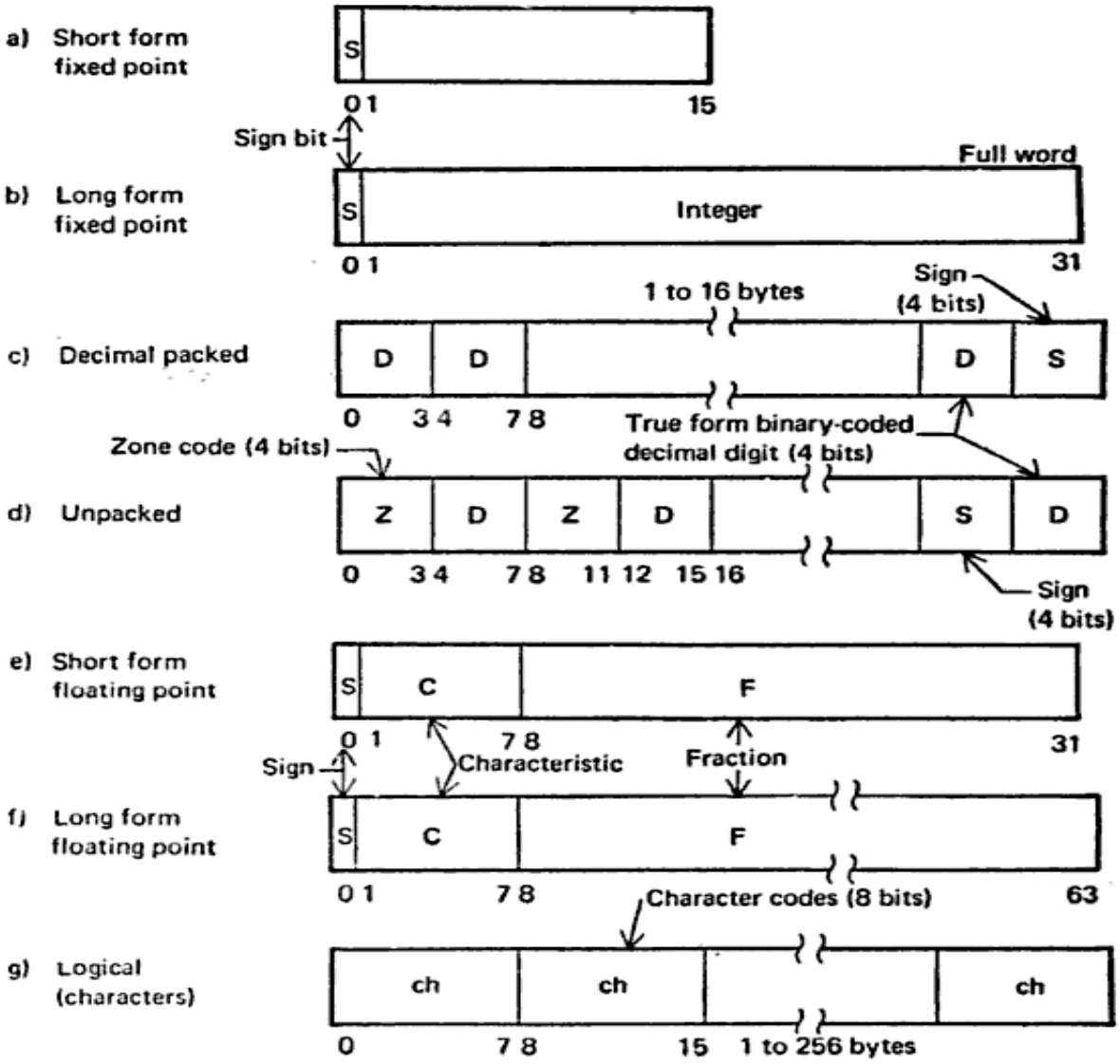


FIGURE 2.3 Data formats for the system/360 and 370

Hexadecimal	*Binary*	*Decimal*
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

FIGURE 2.4 Hexadecimal-binary-decimal conversion

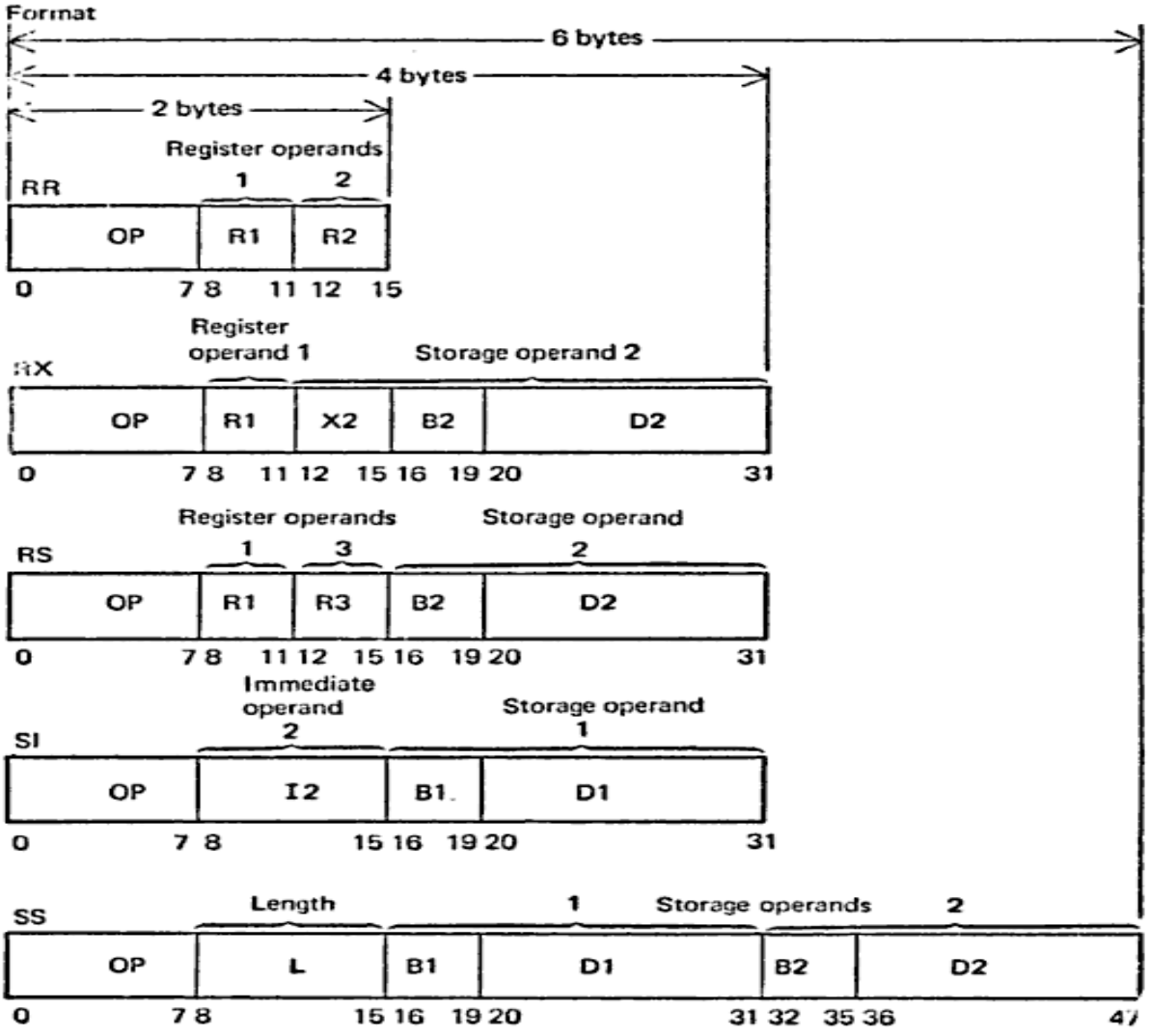
4. INSTRUCTIONS

The 360 has arithmetic, logical, control or transfer, and special interrupt instructions.

The formats of the 360 instructions are depicted in Figure 2.5.

These five types of instructions differ basically in the types of operands they use.

Register operands refer to data stored in one of the 16 general registers (32 bits long), which are addressed by a four-bit field in the instruction. Since registers are usually constructed of high-speed circuitry, they provide faster access to data than does core storage.

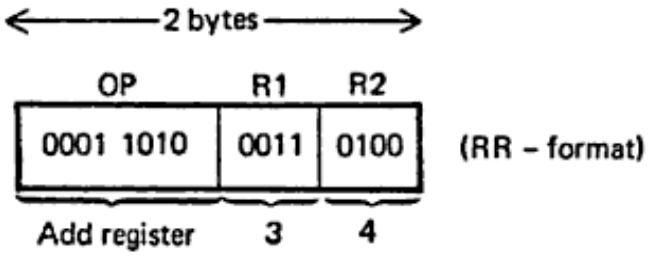


Mnemonics used:

OP	~	operation code
Ri	~	contents of general register used as operand
Xi	~	contents of general register used as index
Bi	~	contents of general register used as base
Di	~	displacement
Ii	~	immediate data
L	~	operand length

FIGURE 2.5 Basic 360 instruction formats

For example, the instruction Add register 3, 4



causes the contents of general register 4 (32 bits) to be added to the contents of general register 3 (32 bits) and the resulting sum to be left in general register 3.

Q: Write an assembly language program to adding instructions.

Ans.

Aosolute address	*Relative address*	*Instructions*	*Comments*
48	0	L 2,904(0,1)	Add 49 to a number
52	4	A 2,900(0,1)	
56	8	ST 2,904(0,1)	
60	12	L 2,0(0,1)	Increase displacement of load instruction by 4
64	16	A 2,896(0,1)	
68	20	ST 2,0(0,1)	
72	24	L 2,8(0,1)	Increase displacement of store instruction by 4
76	28	A 2,896(0,1)	
80	32	ST 2,8(0,1)	
		Branch to relative location 0 nine times	
⋮	⋮		
944	896	4	
948	900	49	
952	904	Numbers	
⋮	⋮		

FIGURE 2.10 Program for addition problem using instruction modification