A general introductory description of the logical structure of SYSTEM/360 *is given in preparation for the more detailed analyses occurring in the other parts of the paper.*

The functional units, the principal registers and formats, and the basic addressing and sequencing principles of the system are indicated.

The structure of SYSTEM/360

Part I—Outline of the logical structure

by G. A. Blaauw and F. P. Brooks, Jr.

SYSTEM/360 is distinguished by a design orientation toward very large memories and a hierarchy of memory speeds, a broad spectrum of manipulative functions, and a uniform treatment of input/output functions that facilitates communication with a diversity of input/output devices. The overall structure lends itself to program-compatible embodiments over a wide range of performance levels.

The system, designed for operation with a supervisory program, has comprehensive facilities for storage protection, program relocation, nonstop operation, and program interruption. Privileged instructions associated with a supervisory operating state are included. The supervisory program schedules and governs the execution of multiple programs, handles exceptional conditions, and coordinates and issues input/output (I/O) instructions. Reliability is heightened by supplementing solid-state components with built-in checking and diagnostic aids. Interconnection facilities permit a wide variety of possibilities for multisystem operation.

The purpose of this discussion is to introduce the functional units of the system, as well as formats, codes, and conventions essential to characterization of the system.

Functional structure

The SYSTEM/360 structure schematically outlined in Figure 1 has seven announced embodiments. Six of these, namely, MODELS 30, 40, 50, 60, 62, and 70, will be treated here.[1] Where requisite I/O devices, optional features, and storage capacity are present, these six models are logically identical for valid programs that contain explicit time dependencies only. Hence, even though the allowable channels or storage capacity may vary from model to model (as discussed in Part II), the logical structure can be discussed without reference to specific models.

input/output

Direct communication with a large number of low-speed terminals and other I/O devices is provided through a special *multiplexor* channel unit. Communication with high-speed I/O devices is accommodated by the *selector* channel units. Conceptually, the input/output system acts as a set of subchannels that operate concurrently with one another and the processing unit. Each subchannel, instructed by its own control-word sequence, can govern a data transfer operation between storage and a selected I/O device. A multiplexor channel can function either as one or as many subchannels; a selector channel always functions as a single subchannel. The control unit of each I/O device attaches to the channels via a standard mechanical-electrical-programming *interface.*

Figure 1 Functional schematic of System/360

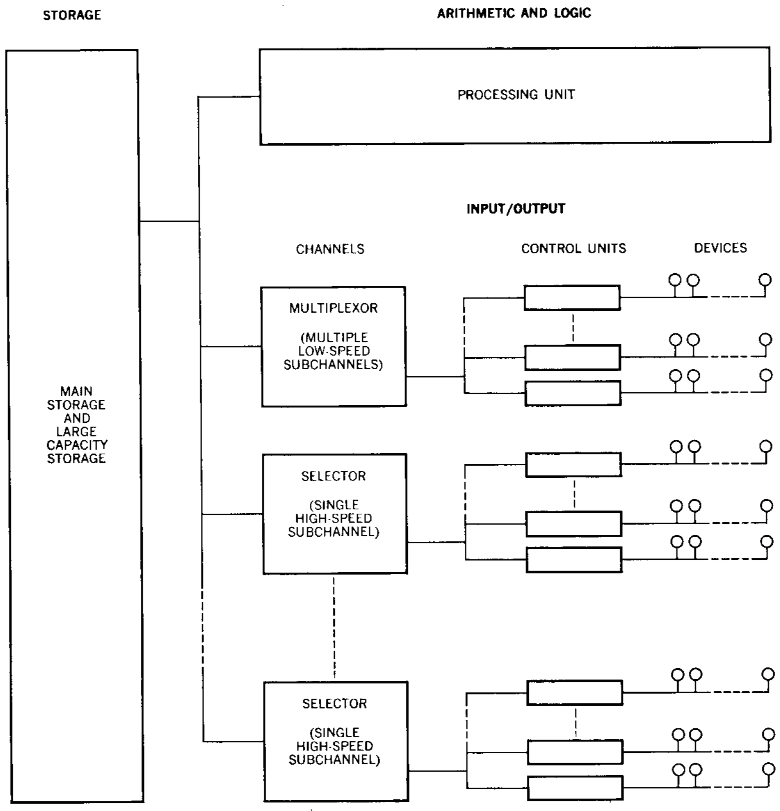
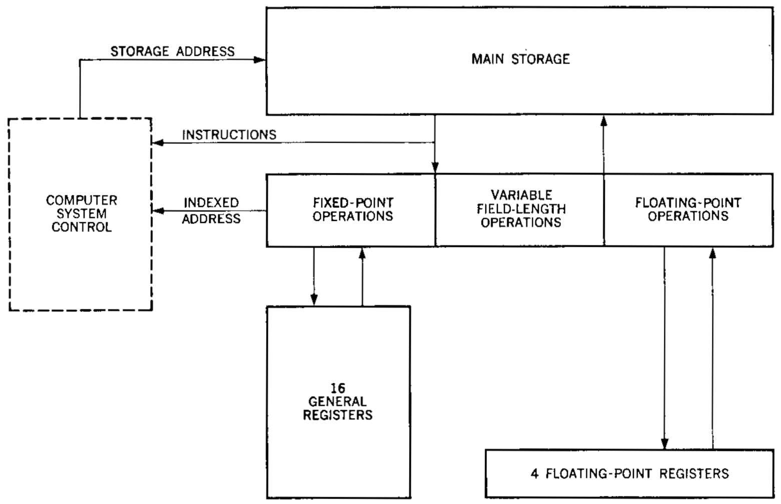

STORAGE
ARITHMETIC AND LOGIC
PROCESSING UNIT
MAIN STORAGE AND LARGE CAPACITY STORAGE
INPUT/OUTPUT
CHANNELS
CONTROL UNITS
DEVICES
MULTIPLEXOR (MULTIPLE LOW-SPEED SUBCHANNELS)
SELECTOR (SINGLE HIGH-SPEED SUBCHANNEL)
SELECTOR (SINGLE HIGH-SPEED SUBCHANNEL)

Figure 2 Schematic of basic registers and data paths



processing

The processing unit has sixteen general purpose 32-bit registers used for addressing, indexing, and accumulating. Four 64-bit floating-point accumulators are optionally available. The inclusion of multiple registers permits effective use to be made of small high-speed memories. Four distinct types of processing are provided: logical manipulation of individual bits, character strings and fixed words; decimal arithmetic on digit strings; fixed-point binary arithmetic; and floating-point arithmetic. The processing unit, together with the central control function, will be referred to as the central processing unit (CPU). The basic registers and data paths of the CPU are shown in Figure 2.

The CPU's of the various models yield a substantial range in performance. Relative to the smallest model (MODEL 30), the internal performance of the largest (MODEL 70) is approximately 50 : 1 for scientific computation and 15 : 1 for commercial data processing.

control

Because of the extensive instruction set, SYSTEM/360 control is more elaborate than in conventional computers. Control functions include internal sequencing of each operation; sequencing from instruction to instruction (with branching and interruption); governing of many I/O transfers; and the monitoring, signaling, timing, and storage protection essential to total system operation. The control equipment is combined with a programmed supervisor, which coordinates and issues all I/O instructions, handles exceptional conditions, loads and relocates programs and data, manages storage, and supervises scheduling and execution of multiple programs. To a problem programmer, the supervisory program and the control equipment are indistinguishable.

The functional structure of SYSTEM/360, like that of most computers, is most concisely described by considering the data formats, the types of manipulations performed on them, and the instruction formats by which these manipulations are specified.

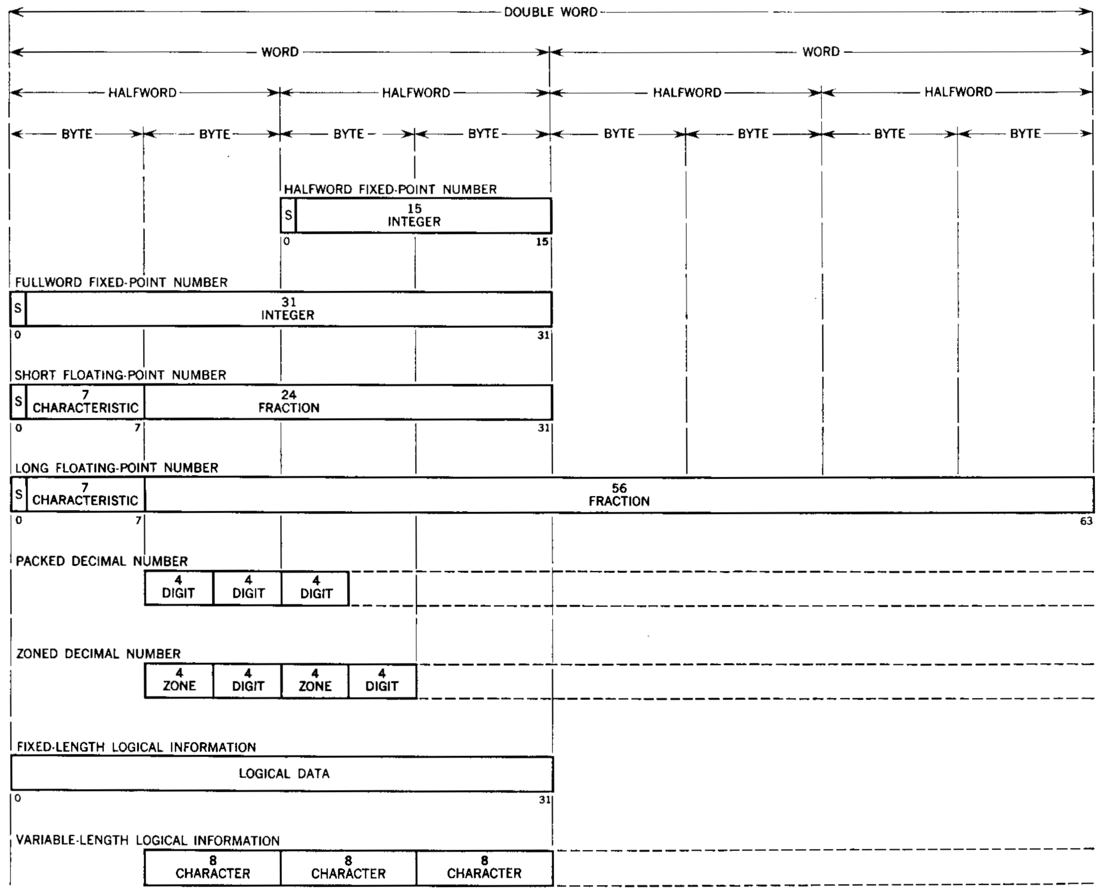
information formats

The several SYSTEM/360 data formats are shown in Figure 3. An 8-bit unit of information is fundamental to most of the formats. A consecutive group of n such units constitutes a *field of length n.* Fixed-length fields of length one, two, four, and eight are termed *bytes, halfwords, words,* and *double words,* respectively. In many instructions, the operation code implies one of these four fields as the length of the operands. On the other hand, the length is explicit in an instruction that refers to operands of variable length.

The location of a stored field is specified by the address of the leftmost byte of the field. Variable-length fields may start on any byte location, but a fixed-length field of two, four, or eight bytes must have an address that is a multiple of 2, 4, or 8, respectively. Some of the various alignment possibilities are apparent from Figure 3.

Storage addresses are represented by binary integers in the system. Storage capacities are always expressed as numbers of bytes.

Figure 3 The data formats



Processing operations

The SYSTEM/360 operations fall into four classes: fixed-point arithmetic, floating-point arithmetic, logical operations, and decimal arithmetic. These classes differ in the data formats used, the registers involved, the operations provided, and the way the field length is stated.

fixed-point arithmetic

The basic arithmetic operand is the 32-bit fixed-point binary word. Halfword operands may be specified in most operations for the sake of improved speed or storage utilization. Some products and all dividends are 64 bits long, using an even-odd register pair.

Because the 32-bit words accommodate the 24-bit address, the entire fixed-point instruction set, including multiplication, division, shifting, and several logical operations, can be used in address computation. A two's complement notation is used for fixed-point operands.

Additions, subtractions, multiplications, divisions, and comparisons take one operand from a register and another from either a register or storage. Multiple-precision arithmetic is made convenient by the two's complement notation and by recognition of the carry from one word to another. A pair of conversion instructions, CONVERT TO BINARY and CONVERT TO DECIMAL, provide transition between decimal and binary radices without the use of tables. Multiple-register loading and storing instructions facilitate subroutine switching.

floating-point arithmetic

Floating-point numbers may occur in either of two fixed-length formats—short or long. These formats differ only in the length of the fractions, as indicated in Figure 3. The fraction of a floating-point number is expressed in 4-bit hexadecimal (base 16) digits. In the short format, the fraction has six hexadecimal digits; in the long format, the fraction has 14 hexadecimal digits. The short length is equivalent to seven decimal places of precision. The long length gives up to 17 decimal places of precision, thus eliminating most requirements for double-precision arithmetic.

The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit. To provide the proper magnitude for the floating-point number, the fraction is considered to be multiplied by a power of 16. The characteristic portion, bits 1 through 7 of both formats, is used to indicate this power. The characteristic is treated as an excess 64 number with a range from -64 through $+63$, and permits representation of decimal numbers with magnitudes in the range of 10^{-78} to 10^{75}.

Bit position 0 in either format is the fraction sign, S. The fraction of negative numbers is carried in true form.

Floating-point operations are performed with one operand from a register and another from either a register or storage. The result, placed in a register, is generally of the same length as the operands.

logical operations

Operations for comparison, translation, editing, bit testing, and bit setting are provided for processing logical fields of fixed and variable lengths. Fixed-length logical operands, which consist of one, four, or eight bytes, are processed from the general registers. Logical operations can also be performed on fields of up to 256 bytes, in which case the fields are processed from left to right, one byte at a time. Moreover, two powerful scanning instructions permit byte-by-byte translation and testing via tables. An important special case of variable-length logical operations is the one-byte field, whose individual bits can be tested, set, reset, and inverted as specified by an 8-bit mask in the instruction.

character codes

Any 8-bit character set can be processed, although certain restrictions are assumed in the decimal arithmetic and editing operations. However, all character-set-sensitive I/O equipment assumes either the Extended Binary-Coded-Decimal Interchange Code (EBCDIC) of Figure 4 or the code of Figure 5, which is an eight-bit extension of a seven-bit code proposed by the International Standards Organization.

decimal arithmetic

Decimal arithmetic can improve performance for processes requiring few computational steps per datum between the source input and the output. In these cases, where radix conversion from decimal to binary and back to decimal is not justified, the use of registers for intermediate results usually yields no advantage over storage-to-storage processing. Hence, decimal arithmetic is provided in SYSTEM/360 with operands as well as results located in storage, as in the IBM 1400 series. Decimal arithmetic includes

Figure 4 Extended Binary-Coded-Decimal Interchange Code

BIT POSITIONS 01	00				01				10				11			
23 / 4567	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0000	NULL				SP	&	_									0
0001							/		a	j			A	J		1
0010									b	k	s		B	K	S	2
0011									c	l	t		C	L	T	3
0100	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	LC	BS	EOB	UC					f	o	w		F	O	W	6
0111	DEL	IL	PRE	EOT					g	p	x		G	P	X	7
1000									h	q	y		H	Q	Y	8
1001									i	r	z		I	R	Z	9
1010			SM		¢	!		:								
1011					.	$	,	#								
1100					<	*	%	@								
1101					(	)	_	'								
1110					+	;	>	=								
1111					|	¬	?	"								

PF	Punch off	BS	Backspace	SM	Set mode
HT	Horizontal tab	IL	Idle	PN	Punch on
LC	Lower case	BYP	Bypass	RS	Reader stop
DEL	Delete	LF	Line feed	UC	Upper case
RES	Restore	EOB	End of block	EOT	End of transmission
NL	New line	PRE	Prefix	SP	Space

Figure 5 Eight-bit representation for proposed international code*

BIT POSITIONS → 76	00				01				10				11			
X5 / 4321	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0000	NULL	DLE			SP	0					_	P			@	p
0001	SOH	DC1			!	1					A	Q			a	q
0010	STX	DC2			"	2					B	R			b	r
0011	ETX	DC3			#	3					C	S			c	s
0100	EOT	DC4			$	4					D	T			d	t
0101	ENQ	NACK			%	5					E	U			e	u
0110	ACK	SYNC			&	6					F	V			f	v
0111	BELL	ETB			'	7					G	W			g	w
1000	BS	CNCL			(	8					H	X			h	x
1001	HT	EM			)	9					I	Y			i	y
1010	LF	SS			*	:					J	Z			j	z
1011	VT	ESC			+	;					K	[			k	{
1100	FF	FS			,	<					L	CS2			l	|
1101	CR	GS			—	=					M	]			m	}
1110	SO	RS			.	>					N	^			n	¬
1111	SI	US			/	?					O	`			o	DEL

*Third ISO draft proposal for 6 and 7 bit coded character sets for information processing interchange, International Standards Organization, June 1964.

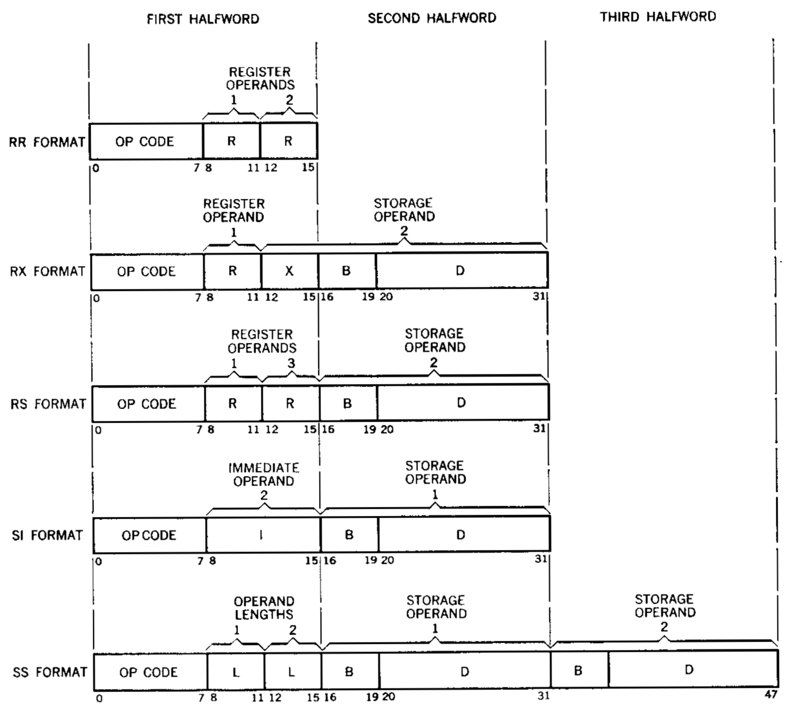
NULL	Null/idle	HT	Horizontal tabulation	DC2	Device control	ESC	Escape
SOH	Start of heading	LF	Line feed	DC3	Device control	FS	File separator
STX	Start of text	VT	Vertical tabulation	DC4	Device control (stop)	GS	Group separator
ETX	End of text	FF	Form feed	NACK	Negative acknowledge	RS	Record separator
EOT	End of transmission	CR	Carriage return	SYNC	Synchronous idle	US	Unit separator
ENQ	Enquiry	SO	Shift out	ETB	End of transmission block	SP	Space, normally non-printing
ACK	Acknowledge	SI	Shift in	CNCL	Cancel	CS2	Currency symbol
BELL	Audible or attention signal	DLE	Data link escape	EM	End of medium	`	Grave accent
BS	Backspace	DC1	Device control	SS	Start of special sequence	DEL	Delete

addition, subtraction, multiplication, division, and comparison.

The decimal digits 0 through 9 are represented in the 4-bit binary-coded-decimal form by 0000 through 1001, respectively. The patterns 1010 through 1111 are not valid as digits and are interpreted as sign codes: 1011 and 1101 represent a minus, the other four a plus. The sign patterns generated in decimal arithmetic depend upon the character set preferred. For EBCDIC, the patterns are 1100 and 1101; for the code of Figure 5, they are 1010 and 1011. The choice between the two codes is determined by a mode bit.

Decimal digits, packed two to a byte, appear in fields of variable length (from 1 to 16 bytes) and are accompanied by a sign in the rightmost four bits of the low-order byte. Operand fields can be located on any byte boundary, and can have lengths up to 31 digits and sign. Operands participating in an operation have independent lengths. Negative numbers are carried in true form. Instructions are provided for packing and unpacking decimal numbers. Packing of digits leads to efficient use of storage, increased arithmetic performance, and improved rates of data transmission. For purely decimal fields, for example, a 90,000-byte/second tape drive reads and writes 180,000 digits/second.

Figure 6 Five basic instruction formats



instruction formats

Instruction formats contain one, two, or three halfwords, depending upon the number of storage addresses necessary for the operation. If no storage address is required of an instruction, one halfword suffices. A two-halfword instruction specifies one address; a three-halfword instruction specifies two addresses. All instructions must be aligned on halfword boundaries.

The five basic instruction formats, denoted by the format mnemonics RR, RX, RS, SI, and SS are shown in Figure 6. RR denotes a register-to-register operation, RX a register and indexed-storage operation, RS a register and storage operation, SI a storage and immediate-operand operation, and SS a storage-to-storage operation.

In each format, the first instruction halfword consists of two parts. The first byte contains the operation code. The length and format of an instruction are indicated by the first two bits of the operation code.

The second byte is used either as two 4-bit fields or as a single 8-bit field. This byte is specified from among the following:

- Four-bit operand register designator (R)
- Four-bit index register designator (X)
- Four-bit mask (M)
- Four-bit field length specification (L)
- Eight-bit field length specification
- Eight-bit byte of immediate data (I)

The second and third halfwords each specify a 4-bit base register designator (B), followed by a 12-bit displacement (D).

An effective storage address E is a 24-bit binary integer given, in the typical case, by

addressing

$$E = B + X + D$$

where B and X are 24-bit integers from general registers identified by fields B and X, respectively, and the displacement D is a 12-bit integer contained in every instruction that references storage.

The base B can be used for static relocation of programs and data. In record processing, the base can identify a record; in array calculations, it can specify the location of an array. The index X can provide the relative address of an element within an array. Together, B and X permit double indexing in array processing.

The displacement provides for relative addressing of up to 4095 bytes beyond the element or base address. In array calculations, the displacement can identify one of many items associated with an element. Thus, multiple arrays whose indices move together are best stored in an interleaved manner. In the processing of records, the displacement can identify items within a record.

In forming an effective address, the base and index are treated as unsigned 24-bit positive binary integers and the displacement as a 12-bit positive binary integer. The three are added as 24-bit binary numbers, ignoring overflow. Since every address is formed with the aid of a base, programs can be readily and generally relocated by changing the contents of base registers.

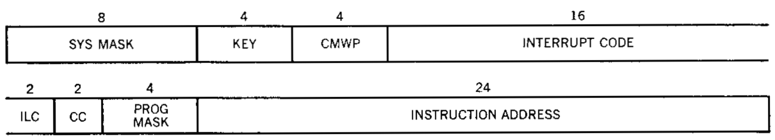
A zero base or index designator implies that a zero quantity must be used in forming the address, regardless of the contents of general register 0. A displacement of zero has no special significance. Initialization, modification, and testing of bases and indices can be carried out by fixed-point instructions, or by BRANCH AND LINK, BRANCH ON COUNT, or BRANCH ON INDEX instructions. LOAD EFFECTIVE ADDRESS provides not only a convenient housekeeping operation, but also, when the same register is specified for result and operand, an immediate register-incrementing operation.

Sequencing

Normally, the CPU takes instructions in sequence. After an instruction is fetched from a location specified by the instruction counter, the instruction counter is increased by the number of bytes in the instruction.

Conceptually, all halfwords of an instruction are fetched from storage after the preceding operation is completed and before execution of the current operation, even though physical storage word size and overlap of instruction execution with storage access may cause the actual instruction fetching to be different. Thus, an instruction can be modified by the instruction that immedi-

Figure 7 Program status word format



SYSTEM MASK— MPX channel
SEL channels 1-6
External

KEY— Storage protection key

CMWP— Character-set mode
Mach check
Wait state
Problem state

ILC— Instruction length code

CC— Condition code

PROGRAM MASK— Fixed point overflow
decimal overflow
exponent underflow
significance

ately precedes it in the instruction stream, and cannot effectively modify itself during execution.

branching

Most branching is accomplished by a single BRANCH ON CONDITION operation that inspects a 2-bit *condition register*. Many of the arithmetic, logical, and I/O operations indicate an outcome by setting the condition register to one of its four possible states. Subsequently a conditional branch can select one of the states as a criterion for branching. For example, the condition code reflects such conditions as non-zero result, first operand high, operands equal, overflow, channel busy, zero, etc. Once set, the condition register remains unchanged until modified by an instruction execution that reflects a different condition code.

The outcome of address arithmetic and counting operations can be tested by a conditional branch to effect loop control. Two instructions, BRANCH ON COUNT and BRANCH ON INDEX, provide for one-instruction execution of the most common arithmetic-test combinations.

program status word

A program status word (PSW), a double word having the format shown in Figure 7, contains information required for proper execution of a given program. A PSW includes an instruction address, condition code, and several mask and mode fields. The active or controlling PSW is called the *current* PSW. By storing the current PSW during an interruption, the status of the interrupted program is preserved.

interruption

Five classes of interruption conditions are distinguished: input/output, program, supervisor call, external, and machine check.

For each class, two PSW's, called *old* and *new*, are maintained in the main-storage locations shown in Table 1. An interruption in a given class stores the current PSW as an old PSW and then takes the corresponding new PSW as the current PSW. If, at the conclusion of the interruption routine, old and current PSW's are interchanged, the system can be restored to its prior state and the interrupted routine can be continued.

The system mask, program mask, and machine-check mask bits in the PSW may be used to control certain interruptions. When masked off, some interruptions remain pending while others are merely ignored. The system mask can keep I/O and external interruptions pending, the program mask can cause four of the 15 program interruptions to be ignored, and the machine-check mask can cause machine-check interruptions to be ignored. Other interruptions cannot be masked off.

Appropriate CPU response to a special condition in the channels and I/O units is facilitated by an I/O *interruption*. The addresses of the channel and I/O unit involved are recorded in the old PSW. Related information is preserved in a channel status word that is stored as a result of the interruption.

Unusual conditions encountered in a program create *program interruptions*. Eight of the fifteen possible conditions involve overflows, improper divides, lost significance, and exponent underflow. The remaining seven deal with improper addresses, attempted execution of privileged instructions, and similar conditions.

A *supervisor-call interruption* results from execution of the instruction SUPERVISOR CALL. Eight bits from the instruction format are placed in the interruption code of the old PSW, permitting a message to be associated with the interruption. SUPERVISOR CALL permits a problem program to switch CPU control back to the supervisor.

Through an *external interruption*, a CPU can respond to signals from the interruption key on the system control panel, the timer, other CPU's, or special devices. The source of the interruption is identified by an interruption code in bits 24 through 31 of the PSW.

The occurrence of a machine check (if not masked off) terminates the current instruction, initiates a diagnostic procedure, and subsequently effects a *machine-check interruption*. A machine check is occasioned only by a hardware malfunction; it cannot be caused by invalid data or instructions.

Table 1 Permanent storage assignments

Address	*Byte length*	*Purpose*
0	8	Initial program loading PSW
8	8	Initial program loading CCW 1
16	8	Initial program loading CCW 2
24	8	External old PSW
32	8	Supervisor call old PSW
40	8	Program old PSW
48	8	Machine check old PSW
56	8	Input/output old PSW
64	8	Channel status word
72	4	Channel address word
76	4	Unused
80	4	Timer
84	4	Unused
88	8	External new PSW
96	8	Supervisor call new PSW
104	8	Program new PSW
112	8	Machine check new PSW
120	8	Input/output new PSW
128		Diagnostic scan-out area*

* The size of the diagnostic scan-out area is configuration dependent.

interrupt priority

Interruption requests are honored between instruction executions. When several requests occur during execution of an instruction, they are honored in the following order: (1) machine check, (2) program or supervisor call, (3) external, and (4) input/output. Because the program and supervisor-call interruptions are mutually exclusive, they cannot occur at the same time.

If a machine-check interruption occurs, no other interruptions can be taken until this interruption is fully processed. Otherwise, the execution of the CPU program is delayed while PSW's are appropriately stored and fetched for each interruption. When the last interruption request has been honored, instruction execution is resumed with the PSW last fetched. An interruption subroutine is then serviced for each interruption in the order (1) input/output, (2) external, and (3) program or supervisor call.

program status

Overall CPU status is determined by four alternatives: (1) *stopped* versus *operating* state, (2) *running* versus *waiting* state, (3) *masked* versus *interruptable* state, and (4) *supervisor* versus *problem* state.

In the stopped state, which is entered and left by manual procedure, instructions are not executed, interruptions are not accepted, and the timer is not updated. In the operating state, the CPU is capable of executing instructions and of being interrupted.

In the running state, instruction fetching and execution proceeds in the normal manner. The wait state is typically entered by the program to await an interruption, for example, an I/O interruption or operator intervention from the console. In the wait state, no instructions are processed, the timer is updated, and I/O and external interruptions are accepted unless masked. Running versus waiting is determined by the setting of a bit in the current PSW.

The CPU may be interruptable or masked for the system, program, and machine interruptions. When the CPU is interruptable for a class of interruptions, these interruptions are accepted. When the CPU is masked, the system interruptions remain pending, but the program and machine-check interruptions are ignored. The interruptable states of the CPU are changed by altering mask bits in the current PSW.

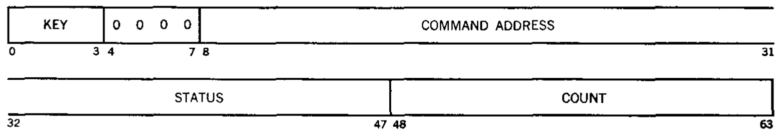
In the problem state, processing instructions are valid, but all I/O instructions and a group of control instructions are invalid. In the supervisor state, all instructions are valid. The choice of problem or supervisor state is determined by a bit in the PSW.

Supervisory Facilities

timer

A timer word in main storage location 80 is counted down at a rate of 50 or 60 cycles per second, depending on power line frequency. The word is treated as a signed integer according to the rules of fixed-point arithmetic. An external interrupt occurs when the value of the timer word goes from positive to negative. The full cycle time of the timer is 15.5 hours.

Figure 8 Channel status word format



Bits 0 through 3 contain the storage protection key used in the operation.
Bits 4 through 7 contain zeros.
Bits 8 through 32 specify the location of the last CCW used.
Bits 32 through 47 contain an I/O-device-status byte and a channel-status byte. The status bytes provide such information as data-check, changing check, control-unit end, etc.
Bits 48 through 63 contain the residual count of the last CCW used.

electrical, logical, and buffering capabilities necessary for I/O device operation. From the programming point of view, most control-unit and I/O device functions are indistinguishable. Sometimes the control unit is housed with an I/O device, as in the case of the printer.

A control unit functions only with those I/O devices for which it is designed, but all control units respond to a standard set of signals from the channel. This control-unit-to-channel connection, called the I/O *interface*, enables the CPU to handle all I/O operations with only four instructions.

I/O instructions

Input/output instructions can be executed only while the CPU is in the supervisor state. The four I/O instructions are START I/O, HALT I/O, TEST CHANNEL, and TEST I/O.

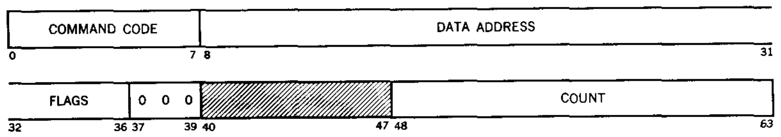
START I/O initiates an I/O operation; its address field specifies a channel and an I/O device. If the channel facilities are free, the instruction is accepted and the CPU continues its program. The channel independently selects the specified I/O device. HALT I/O terminates a channel operation. TEST CHANNEL sets the condition code in the PSW to indicate the state of the channel addressed by the instruction. The code then indicates one of the following conditions: channel available, interruption condition in channel, channel working, or channel not operational. TEST I/O sets the PSW condition code to indicate the state of the addressed channel, subchannel, and I/O device.

channels

Channels provide the data path and control for I/O devices as they communicate with main storage. In the multiplexor channel, the single data path can be time-shared by several low-speed devices (card readers, punches, printers, terminals, etc.) and the channel has the functional character of many subchannels, each of which services one I/O device at a time. On the other hand, the selector channel, which is designed for high-speed devices, has the functional character of a single subchannel. All subchannels respond to the same I/O instructions. Each can fetch its own control word sequence, govern the transfer of data and control signals, count record lengths, and interrupt the CPU on exceptions.

Two modes of operation, *burst* and *multiplex*, are provided for multiplexor channels. In burst mode, the channel facilities are monopolized for the duration of data transfer to or from a particular I/O device. The selector channel functions only in the burst mode. In multiplex mode, the multiplexor channel sustains several simultaneous I/O operations: bytes of data are interleaved

Figure 9 Channel command word format



Bits 0 through 7 specify the command code.
Bits 8 through 31 specify the location of a byte in main storage.
Bits 32 through 36 are flag bits.
Bit 32 causes the address portion of the next CCW to be used.
Bit 33 causes the command code and data address in the next CCW to be used.
Bit 34 causes a possible incorrect length indication to be suppressed.
Bit 35 suppresses the transfer of information to main storage.
Bit 36 causes an interruption.
Bits 37 through 39 must contain zeros.
Bits 40 through 47 are ignored.
Bits 48 through 63 specify the number of bytes in the operation.

and then routed between selected I/O devices and desired locations in main storage.

At the conclusion of an operation launched by START I/O or TEST I/O, an I/O interruption occurs. At this time a channel status word (CSW) is stored in location 64. Figure 8 shows the CSW format. The CSW provides information about the termination of the I/O operation.

Successful execution of START I/O causes the channel to fetch a channel address word from main-storage location 72. This word specifies the storage-protection key that governs the I/O operation, as well as the location of the first eight bytes of information that the channel fetches from main storage. These 64 bits comprise a channel command word (CCW). Figure 9 shows the CCW format.

channel program

One or more CCW's make up the channel program that directs channel operations. Each CCW points to the next one to be fetched, except for the last in the chain which so identifies itself.

Six channel commands are provided: read, write, read backward, sense, transfer in channel, and control. The read command defines an area in main storage and causes a read operation from the selected I/O device. The write command causes data to be written by the selected device. The read-backward command is akin to the read command, but the external medium is moved in the opposite direction and bytes read backward are placed in descending main storage locations.

The control command contains information, called an *order*, that is used to control the selected I/O device. Orders, peculiar to the particular I/O device in use, can specify such functions as rewinding a tape unit, searching for a particular track in disk storage, or line skipping on a printer. In a functional sense, the CPU executes I/O instructions, the channels execute commands, and the control units and devices execute orders.

The sense command specifies a main storage location and transfers one or more bytes of status information from the selected control unit. It provides details concerning the selected I/O device, such as a stacker-full condition of a card reader or a file-protected condition of a magnetic-tape reel.

A channel program normally obtains CCW's from a consecutive string of storage locations. The string can be broken by a transfer-in-channel command that specifies the location of the next CCW to be used by the channel. External documents, such as punched cards or magnetic tape, may carry CCW's that can be used by the

Table 2 System/360 Instructions

RR Format

	BRANCHING AND STATUS SWITCHING		FIXED-POINT FULLWORD AND LOGICAL		FLOATING-POINT LONG		FLOATING-POINT SHORT	
xxxx	0000xxxx		0001xxxx		0010xxxx		0011xxxx	
0000			LPR	LOAD POSITIVE	LPDR	LOAD POSITIVE	LPER	LOAD POSITIVE
0001			LNR	LOAD NEGATIVE	LNDR	LOAD NEGATIVE	LNER	LOAD NEGATIVE
0010			LTR	LOAD AND TEST	LTDR	LOAD AND TEST	LTER	LOAD AND TEST
0011			LCR	LOAD COMPLEMENT	LCDR	LOAD COMPLEMENT	LCER	LOAD COMPLEMENT
0100	SPM	SET PROGRAM MASK	NR	AND	HDR	HALVE	HER	HALVE
0101	BALR	BRANCH AND LINK	CLR	COMPARE LOGICAL				
0110	BCTR	BRANCH ON COUNT	OR	OR				
0111	BCR	BRANCH/CONDITION	XR	EXCLUSIVE OR				
1000	SSK	SET KEY	LR	LOAD	LDR	LOAD	LER	LOAD
1001	ISK	INSERT KEY	CR	COMPARE	CDR	COMPARE	CER	COMPARE
1010	SVC	SUPERVISOR CALL	AR	ADD	ADR	ADD N	ALR	ADD N
1011			SR	SUBTRACT	SDR	SUBTRACT N	SER	SUBTRACT N
1100			MR	MULTIPLY	MDR	MULTIPLY	MER	MULTIPLY
1101			DR	DIVIDE	DDR	DIVIDE	DER	DIVIDE
1110			ALR	ADD LOGICAL	AWR	ADD U	AUR	ADD U
1111			SLR	SUBTRACT LOGICAL	SWR	SUBTRACT U	SUR	SUBTRACT U

RX Format

	FIXED-POINT HALFWORD AND BRANCHING		FIXED-POINT FULLWORD AND LOGICAL		FLOATING-POINT LONG		FLOATING-POINT SHORT	
xxxx	0100xxxx		0101xxxx		0110xxxx		0111xxxx	
0000	STH	STORE	ST	STORE	STD	STORE	STE	STORE
0001	LA	LOAD ADDRESS						
0010	STC	STORE CHARACTER						
0011	IC	INSERT CHARACTER						
0100	EX	EXECUTE	N	AND				
0101	BAL	BRANCH AND LINK	CL	COMPARE LOGICAL				
0110	BCT	BRANCH ON COUNT	O	OR				
0111	BC	BRANCH/CONDITION	X	EXCLUSIVE OR				
1000	LH	LOAD	L	LOAD	LD	LOAD	LE	LOAD
1001	CH	COMPARE	C	COMPARE	CD	COMPARE	CE	COMPARE
1010	AH	ADD	A	ADD	AD	ADD N	AE	ADD N
1011	SH	SUBTRACT	S	SUBTRACT	SD	SUBTRACT N	SE	SUBTRACT N
1100	MH	MULTIPLY	M	MULTIPLY	MD	MULTIPLY	ME	MULTIPLY
1101			D	DIVIDE	DD	DIVIDE	DE	DIVIDE
1110	CVD	CONVERT-DECIMAL	AL	ADD LOGICAL	AW	ADD U	AU	ADD U
1111	CVB	CONVERT-BINARY	SL	SUBTRACT LOGICAL	SW	SUBTRACT U	SU	SUBTRACT U

RS, SI Format

	BRANCHING STATUS SWITCHING AND SHIFTING		FIXED-POINT LOGICAL AND INPUT/OUTPUT			
xxxx	1000xxxx		1001xxxx		1010xxxx	1011xxxx
0000	SSM	SET SYSTEM MASK	STM	STORE MULTIPLE		
0001			TM	TEST UNDER MASK		
0010	LPSW	LOAD PSW	MVI	MOVE		
0011		DIAGNOSE	TS	TEST AND SET		
0100	WRD	WRITE DIRECT	NI	AND		
0101	RDD	READ DIRECT	CLI	COMPARE LOGICAL		
0110	BXH	BRANCH/HIGH	OI	OR		
0111	BXLE	BRANCH/LOW-EQUAL	XI	EXCLUSIVE OR		
1000	SRL	SHIFT RIGHT SL	LM	LOAD MULTIPLE		
1001	SLL	SHIFT LEFT SL				
1010	SRA	SHIFT RIGHT S				
1011	SLA	SHIFT LEFT S				
1100	SRDL	SHIFT RIGHT DL	SIO	START I/O		
1101	SLDL	SHIFT LEFT DL	TIO	TEST I/O		
1110	SRDA	SHIFT RIGHT D	HIO	HALT I/O		
1111	SLDA	SHIFT LEFT D	TCH	TEST CHANNEL		

SS Format

		LOGICAL			DECIMAL	
xxxx	1100xxxx	1101xxxx		1110xxxx	1111xxxx	
0000						
0001		MVN	MOVE NUMERIC		MVO	MOVE WITH OFFSET
0010		MVC	MOVE		PACK	PACK
0011		MVZ	MOVE ZONE		UNPK	UNPACK
0100		NC	AND			
0101		CLC	COMPARE LOGICAL			
0110		OC	OR			
0111		XC	EXCLUSIVE OR			
1000					ZAP	ZERO AND ADD
1001					CP	COMPARE
1010					AP	ADD
1011					SP	SUBTRACT
1100		TR	TRANSLATE		MP	MULTIPLY
1101		TRT	TRANSLATE AND TEST		DP	DIVIDE
1110		ED	EDIT			
1111		EDMK	EDIT AND MARK			

NOTE: N = NORMALIZED
SL = SINGLE LOGICAL
DL = DOUBLE LOGICAL
U = UNNORMALIZED
S = SINGLE
D = DOUBLE

channel to govern the reading of the documents.

The input/output interruptions caused by termination of an I/O operation, or by operator intervention at the I/O device, enable the CPU to provide appropriate programmed response to conditions as they occur in I/O devices or channels. Conditions responsible for I/O interruption requests are preserved in the I/O devices or channels until recognized by the CPU.

During execution of START I/O, a command can be rejected by a busy condition, program check, etc. Rejection is indicated in the condition code of the PSW, and additional detail on the conditions that precluded initiation of the I/O operation is provided in a CSW.

manual control

The need for manual control is minimal because of the design of the system and supervisory program. A control panel provides the ability to reset the system; store and display information in main storage, in registers, and in the PSW; and load initial program information. After an input device is selected with the load unit switches, depressing a load key causes a read from the selected input device. The six words of information that are read into main storage provide the PSW and the CCW's required for subsequent operation.

instruction set

The SYSTEM/360 instructions, classified by format and function, are displayed in Table 2. Operation codes and mnemonic abbreviations are also shown. With the previously described formats in mind, much of the generality provided by the system is apparent in this listing.

Summary

In the SYSTEM/360 logical structure, processing efficiency and versatility are served by multiple accumulators, binary addressing, bit-manipulation operations, automatic indexing, fixed and variable field lengths, decimal and hexadecimal radices, and floating-point as well as fixed-point arithmetic. The provisions for program interruption, storage protection, and flexible CPU states contribute to effective operation. Base-register addressing, the standard interface between channels and input/output control units, and the machine-language compatibility among models contribute to flexible configurations and to orderly system expansion.

FOOTNOTE

1. A seventh embodiment, the Model 92, is not discussed in this paper. This model does not provide decimal data handling and has a few minor differences arising from its highly concurrent, speed-oriented organization. A paper on Model 92 is planned for future publication in the *IBM Systems Journal.*