F

The IBM 360/370 Architecture for Mainframe Computers

We are not at all humble in this announcement. This is the most important product announcement that this corporation has ever made in its history. It's not a computer in any previous sense. It's not a product, but a line of products … that spans in performance from the very low part of the computer line to the very high.

IBM spokesman
at announcement of System/360 (1964)

F.1 Introduction

The term "computer architecture" was coined by IBM in 1964 for use with the IBM 360. Amdahl, Blaauw, and Brooks [1994] used the term to refer to the programmer-visible portion of the instruction set. They believed that a family of machines of the same architecture should be able to run the same software. Although this idea may seem obvious to us today, it was quite novel at the time. IBM, even though it was the leading company in the industry, had five different architectures before the 360. Thus, the notion of a company standardizing on a single architecture was a radical one. The 360 designers hoped that six different divisions of IBM could be brought together by defining a common architecture. Their definition of architecture was

> . . . the structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for that machine.

The term "machine language programmer" meant that compatibility would hold, even in assembly language, while "timing independent" allowed different implementations.

The IBM 360 was introduced in 1964 with six models and a 25:1 performance ratio. Amdahl, Blaauw, and Brooks [1994] discuss the architecture of the IBM 360 and the concept of permitting multiple object-code–compatible implementations. The notion of an instruction set architecture as we understand it today was the most important aspect of the 360. The architecture also introduced several important innovations, now in wide use:

1. 32-bit architecture
2. Byte-addressable memory with 8-bit bytes
3. 8-, 16-, 32-, and 64-bit data sizes

In 1971, IBM shipped the first System/370 (models 155 and 165), which included a number of significant extensions of the 360, as discussed by Case and Padegs [1978], who also discuss the early history of System/360. The most important addition was virtual memory, though virtual memory 370s did not ship until 1972 when a virtual memory operating system was ready. By 1978, the high-end 370 was several hundred times faster than the low-end 360s shipped ten years earlier. In 1984, the 24-bit addressing model built into the IBM 360 needed to be abandoned, and the 370-XA (eXtended Architecture) was introduced. While old 24-bit programs could be supported without change, several instructions could not function in the same manner when extended to a 32-bit addressing model (31-bit addresses supported) because they would not produce 31-bit addresses. Converting the operating system, which was written mostly in assembly language, was no doubt the biggest task.

Several studies of the IBM 360 and instruction measurement have been made. Shustek's thesis [1978] is the best known and most complete study of the 360/370 architecture. He made several observations about instruction set complexity that were not fully appreciated until some years later. Another important study of the

360 is the Toronto study by Alexander and Wortman [1975] done on an IBM 360 using 19 XPL programs.

F.2 System/360 Instruction Set

The 360 instruction set is shown in the following tables, organized by instruction type and format. System/370 contains 15 additional user instructions.

Integer/Logical and Floating-Point R-R Instructions

The * indicates the instruction is floating point, and may be either D (double precision) or E (single precision).

Instruction	Description
ALR	Add logical register
AR	Add register
A*R	FP addition
CLR	Compare logical register
CR	Compare register
C*R	FP compare
DR	Divide register
D*R	FP divide
H*R	FP halve
LCR	Load complement register
LC*R	Load complement
LNR	Load negative register
LN*R	Load negative
LPR	Load positive register
LP*R	Load positive
LR	Load register
L*R	Load FP register
LTR	Load and test register
LT*R	Load and test FP register
MR	Multiply register
M*R	FP multiply
NR	And register
OR	Or register
SLR	Subtract logical register
SR	Subtract register
S*R	FP subtraction
XR	Exclusive or register

Branches and Status Setting R-R Instructions

These are R-R format instructions that either branch or set some system status; several of them are privileged and legal only in supervisor mode.

Instruction	Description
BALR	Branch and link
BCTR	Branch on count
BCR	Branch/condition
ISK	Insert key
SPM	Set program mask
SSK	Set storage key
SVC	Supervisor call

Branches/Logical and Floating-Point Instructions—RX Format

These are all RX format instructions. The symbol "+" means either a word operation (and then stands for nothing) or H (meaning half word); for example, A+ stands for the two opcodes A and AH. The symbol "*" is D or E standing for double- or single-precision floating point.

Instruction	Description
A+	Add
A*	FP add
AL	Add logical
C+	Compare
C*	FP compare
CL	Compare logical
D	Divide
D*	FP divide
L+	Load
L*	Load FP register
M+	Multiply
M*	FP multiply
N	And
O	Or
S+	Subtract
S*	FP subtract
SL	Subtract logical
ST+	Store
ST*	Store FP register
X	Exclusive or

Branches and Special Loads and Stores—RX format

Instruction	Description
BAL	Branch and link
BC	Branch condition
BCT	Branch on count
CVB	Convert-binary
CVD	Convert-decimal
EX	Execute
IC	Insert character
LA	Load address
STC	Store character

RS and SI Format Instructions

These are the RS and SI format instructions. The symbol "*" may be A (arithmetic) or L (logical).

Instruction	Description
BXH	Branch/high
BXLE	Branch/low-equal
CLI	Compare logical immediate
HIO	Halt I/O
LPSW	Load PSW
LM	Load multiple
MVI	Move immediate
NI	And immediate
OI	Or immediate
RDD	Read direct
SIO	Start I/O
SL*	Shift left A/L
SLD*	Shift left double A/L
SR*	Shift right A/L
SRD*	Shift right double A/L
SSM	Set system mask
STM	Store multiple
TCH	Test channel
TIO	Test I/O
TM	Test under mask
TS	Test-and-set
WRD	Write direct
XI	Exclusive or immediate

SS Format Instructions

These are add decimal or string instructions.

Instruction	Description
AP	Add packed
CLC	Compare logical chars
CP	Compare packed
DP	Divide packed
ED	Edit
EDMK	Edit and mark
MP	Multiply packed
MVC	Move character
MVN	Move numeric
MVO	Move with offset
MVZ	Move zone
NC	And characters
OC	Or characters
PACK	Pack (Character → decimal)
SP	Subtract packed
TR	Translate
TRT	Translate and test
UNPK	Unpack
XC	Exclusive or characters
ZAP	Zero and add packed

F.3 360 Detailed Measurements

Figure F.1 shows the frequency of instruction usage for four IBM 360 programs.

Instruction	PLIC	FORTGO	PLIGO	COBOLGO	Average
Control	**32%**	**13%**	**5%**	**16%**	**16%**
BC, BCR	28%	13%	5%	14%	15%
BAL, BALR	3%			2%	1%
Arithmetic/logical	**29%**	**35%**	**29%**	**9%**	**26%**
A, AR	3%	17%	21%		10%
SR	3%	7%			3%
SLL		6%	3%		2%
LA	8%	1%	1%		2%
CLI	7%				2%
NI				7%	2%
C	5%	4%	4%	0%	3%
TM	3%	1%		3%	2%
MH			2%		1%
Data transfer	**17%**	**40%**	**56%**	**20%**	**33%**
L, LR	7%	23%	28%	19%	19%
MVI	2%		16%	1%	5%
ST	3%		7%		3%
LD		7%	2%		2%
STD		7%	2%		2%
LPDR		3%			1%
LH	3%				1%
IC	2%				1%
LTR		1%			0%
Floating point		**7%**			**2%**
AD		3%			1%
MDR		3%			1%
Decimal, string	**4%**			**40%**	**11%**
MVC	4%			7%	3%
AP				11%	3%
ZAP				9%	2%
CVD				5%	1%
MP				3%	1%
CLC				3%	1%
CP				2%	1%
ED				1%	0%
Total	**82%**	**95%**	**90%**	**85%**	**88%**

Figure F.1 Distribution of instruction execution frequencies for the four 360 programs. All instructions with a frequency of execution greater than 1.5% are included. Immediate instructions, which operate on only a single byte, are included in the section that characterized their operation, rather than with the long character-string versions of the same operation. By comparison, the average frequencies for the major instruction classes of the VAX are 23% (control), 28% (arithmetic), 29% (data transfer), 7% (floating point), and 9% (decimal). Once again, a 1% entry in the average column can occur because of entries in the constituent columns. These programs are a compiler for the programming language PL-I and run time systems for the programming languages FORTRAN, PL/I, and Cobol.

F.4 Historical Perspective and References

The IBM 360 was the first computer to sell in large quantities with both byte addressing using 8-bit bytes and general-purpose registers. The 360 also had register-memory and limited memory-memory instructions. This architecture blazed the path for binary compatibility, which others have followed.

The architects of the IBM 360 were aware of the importance of address size and planned for the architecture to extend to 32 bits of address. Only 24 bits were used in the IBM 360, however, because the low-end 360 models would have been even slower with the larger addresses in 1964. Unfortunately, the architects didn't reveal their plans to the software people, and programmers who stored extra information in the upper 8 "unused" address bits foiled the expansion effort. Virtually every computer since then will check to make sure the unused bits stay unused, and will trap if the bits have the wrong value.

IBM officially extended the address to 32 bits in 1970 with the IBMs/370 architecture. Only recently did IBM expand this architecture to a flat, 64-bit address, with the IBMs/390.

References

Alexander, W. G., and D. B. Wortman [1975]. "Static and dynamic characteristics of XPL programs," *IEEE Computer* 8(11) (November), 41–46.

Amdahl, G. M., G. A. Blaauw, and F. P. Brooks, Jr. [1964]. "Architecture of the IBM System/360," *IBM J. Research and Development* 8:2 (April), 87–101.

Case, R. P., and A. Padegs [1978]. "The architecture of the IBM System/370," *Communications of the ACM,* 21:1, 73–96.

Shustek, L. J. [1978]. *Analysis and Performance of Computer Instruction Sets*. Ph.D. dissertation, Stanford University (January).