Second Generation Mainframes

Second Generation Mainframes:

The IBM 7000 Series

By

Stephen H. Kaisler

Cambridge
Scholars
Publishing

Second Generation Mainframes: The IBM 7000 Series
Historical Computing Machine Series

By Stephen H. Kaisler

This book first published 2019

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-5275-2675-5
ISBN (13): 978-1-5275-2675-4

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This book would not have been possible with the assistance of my two proofreaders – Rebecca Williams and Eric Ward. They did yeoman duty in carefully scrutinizing every word and formatting to make sure I followed the procedures established by Cambridge Scholars Publishing.

I also want to acknowledge the contributions of the family cats – Scooby, Izzy, and Tatiana - who kept me company during periods of intense writing and who were always willing to interrupt to get their daily treats and pettings.

I want to acknowledge the assistance of Max Campbell, archivist on the Reference Desk at the IBM Archives. Max reviewed the photos I had included in the book, then went and searched for high-resolution copies of these photos and scanned them at high resolution. I used these photos and they have made the phots clear and more useful to the readers.

Lastly, I acknowledge my wife, Chryl, who has provided the time to write this volume and the many more to come in this series. This book is dedicated to her.

INTRODUCTION

THE IBM 7000 SYSTEMS

International Business Machines (IBM) Corporation was one of the first computer manufacturers to develop large computer systems. Its early computer systems – the 700 series and the 7000 series – introduced the concept of a 'family' of computers although individual models might be incompatible with each other. In this case, the family concept was focused on a technology base for the central processors - a set of processes to manufacture computer systems repeatedly to the same set of specifications and reliability, and allowed the sharing of storage devices and unit record peripherals among the different models.

The early history of IBM computing machines in the post-650 vacuum tube era comprises two families of mainframe computer systems – each with its own instruction set and operating system. These are:

IBM 70x - the first generation mainframes
IBM 70xx family – the 2nd generation mainframes

By mainframes, we mean large computer systems – usually encompassing a large room – that comprised a central processing unit and its memory; high speed, large storage peripherals such as magnetic tapes, drums and discs; and a variety of slow-speed unit record peripherals. The term "mainframe" conjured the image of men (almost always men) wearing white lab coats and ties and scurrying around the computer room containing the large-scale system. Mainframes of the early era also had large operator consoles with lights, switches and dials allowing a great degree of control over the machine's operation.

The IBM 70x family was implemented using vacuum tubes for the CPU, but magnetic drums for the main memory. As we saw in *First Generation Mainframes: The IBM 700 Series*, IBM had developed an early series of vacuum tube machines. Building on the design of the 700 series, IBM introduced the first of the IBM 7000 series in the mid-1950s. For the 70xx series, the CPUs were implemented using transistors and the main memory was implemented using magnetic cores. There were six types of

machines in the 7000 series: 7010, 7030, 704x, 707x, 7080, and 709x – with some types having second generation iterations. I have placed these machines in the early mainframe category because they were clearly intended to serve as enterprise-level machines.

CHAPTER ONE

PROJECT STRETCH

IBM had competed for the LARC (Livermore Advanced Research Computer) contract in 1955, but was not able to deliver the capability in the time frame desired. In late 1955, IBM made a proposal to the Los Alamos Scientific Laboratory (LASL) for a computer system with a performance capability '100 to 200 times the 704'. Project STRETCH was formally initiated in January 1956 at an initial cost of $4,800,000 (Buchholz 1962; Brooks 2010).

In November 1956, after a competitive bidding process, IBM was awarded a contract to develop the STRETCH computer system. At the time, IBM was engaged in three different supercomputer designs: the STRETCH, the ACS-1, and the IBM 360/9x series (Smotherman and Specier 2010). Ultimately, the STRETCH was delivered in limited quantities and the ACS-1 was canceled, but the IBM System 360 series proved to be very successful. Table 1-1 presents the STRETCH's basic characteristics.

By 1959, the first engineering model of the STRETCH computer began to be assembled. As delivered in 1961, STRETCH consisted of "15 feet of 5-foot high, 4-foot deep rollagons" (Brooks 2010). Smotherman and Spicer (2010) have a picture of the STRETCH operator console with its rollagon cabinets stretching (no pun intended) off into the distance.

In February, 1957, the National Security Agency (NSA) developed a concept for a high-speed, non-arithmetic processing system called "HARVEST". IBM wanted to bid the STRETCH concept for this application as well as develop a machine for commercial sale. Further studies resulted in some redesign when the "three-in-one" concept could not be fully realized. A customized version of the STRETCH, named the IBM 7950 HARVEST system, was delivered to NSA in 1962 (Brooks 2010)

Werner Buchholz (1962) edited a volume of contributions from participants in Project STRETCH that describe many facets of the program and how they influenced the development of the IBM 7030.

Table 1-1. STRETCH Basic Characteristics

Characteristic	Value/Explanation
Internal Representation	Fixed Point Binary; Floating Point Binary
# Bits/Word	64
# Instructions/Word	2
# Instructions	> 700
# Bits/Instruction	Varied
Instruction Type	One Address
CPU Technology	Drift transistors: 2.1 microsecond cycle time
CPU Registers	16 index registers
Main Memory	Magnetic Core: 256K words
Add Time	Floating Point: 0.8 microseconds
Multiply Time	Floating Point: 1.4 microseconds
Divide Time	Unknown

1.1 Technical and Organizational Goals

IBM defined several technical and organizational goals for Project STRETCH that could yield a computer system that would give IBM a competitive advantage in the high-speed computing arena (Dunwell 1956). Remember that at this time (mid-1950s) most computers were used by the government for scientific and engineering purposes. Having lost the LARC competition, IBM was determined to be more competitive the next time around. Table 1-2 describes the primary goals for Project STRETCH.

Table 1-2. Project STRETCH Goals

Goal	Description
Performance	Using the IBM 704 as a reference point, STRETCH should be 100 times as fast.
Reliability	Use solid-state components which were more reliable, operated faster, and consumed less power than vacuum tubes used in older machines.
Checking	STRETCH would incorporate enhanced checking circuits to detect any errors and localize faults. Storage devices would be equipped with error-correction facilities to ensure data could be recovered despite the occasional error.

Generality	To ensure the broadest applicability to different domains, it was decided to include the best features of scientific, data processing, and real-time computers.
High-speed Arithmetic	A high-speed floating point unit to support scientific calculations to perform additions in 0.8 microseconds and multiplications in 1.4 microseconds.
Editing	A separate serial computer to provided editing of variable length data for I/O.
Memory	An initial capacity of 16,384 words operating with a 2 microsecond cycle time.
I/O Exchange	A telephone-like exchange to provide simultaneous data exchange between units of the system.
High-speed Magnetic Disks	External data storage units would have a capacity of 2 million words.

The goal of achieving "100 times" the performance speed of the IBM 704 was extremely aggressive. Technology enhancements were yielding improvements of 1-20% in logic circuits and 6% in memory systems. The STRETCH designers determined to make up the difference with extensive parallelism (Dunwell 1956). As a result, STRETCH introduced many innovations in computer architecture, which are briefly discussed in Section 1.6. Blosk (1960) described the design and performance goals for the STRETCH instruction unit.

STRETCH was designed by a team of IBM engineers. Initial concepts were developed by Dunwell and Buchholz. Gene Amdahl worked on the design of a transistorized, high-performance scientific computer for the proposed Livermore machine. He and John Backus helped write the proposal for an alternate machine after Livermore awarded the LARC contract to UNIVAC. Although not accepted, the ideas contained in that proposal strongly influenced the STRETCH design (Smotherman 2010).

When Stephen Dunwell was assigned to head the project, Amdahl left the company to do other work. Dunwell recruited several key people, including Gerritt Blaauw, Frederick Brooks Jr., Robert Blosk, John Cocke, and Harwood Kolsky – all of whom became famous for their contributions to computer architecture. Cocke and Kolsky (1959) designed a virtual memory system for the STRETCH that was the first of its kind in any IBM mainframe. It influenced the virtual memory design for the System/360

which continues to the present. Blosk and Brooks investigated key ideas in architecture, such as the interrupt system and indexing (Smotherman 2010).

Steve Dunwell, the IBM Project Manager, was made the scapegoat when the STRETCH was perceived as a failure. It was not until the success of the IBM System/360 that the engineering achievements his team developed became clear. He was later given a formal apology by IBM and made an IBM Fellow - the highest honor a *scientist*, engineer, or programmer at IBM can achieve.

Stephen W. Dunwell (1913-1994)
Stephen Dunwell was born April 3, 1913 in Kalamazoo, Michigan. He began work for IBM in 1933 as a cooperative education student while attending Antioch College of Ohio. From 1934 to 1942, he worked in the Future Demands group. During World War II, Dunwell led a team that extended commercial IBM punch card machines with special relay calculators to achieve high performance of 150 cards per minute which was equivalent to about 1 million comparisons per second. In 1942, he was recruited by William Friedman of the Army Signal Corps to lead a mechanical branch to develop a machine to break Japanese codes. After the war, he rejoined IBM where he introduced concepts from his war activities into the IBM 603 Electronic Multiplier. He also participated in the design of several other IBM calculating machines. He worked at IBM headquarters in future planning from 1946 until 1954 when he moved to Poughkeepsie, New York to work under T. Vincent Learson. Stephen Dunwell led the team that developed the STRETCH Computer and later contributed to several other IBM computing systems. After the successes of the IBM System/360 were recognized, Dunwell was named an IBM Fellow in 1966 and received a public apology from Thomas Watson, Jr at an IBM Awards Dinner. He received an IEEE Computer Pioneer Award in 1992. Later in his career, Dunwell owned a consulting company. Source: https://www-03.ibm.com/ibm/history/exhibits/builders/builders_dunwell.html

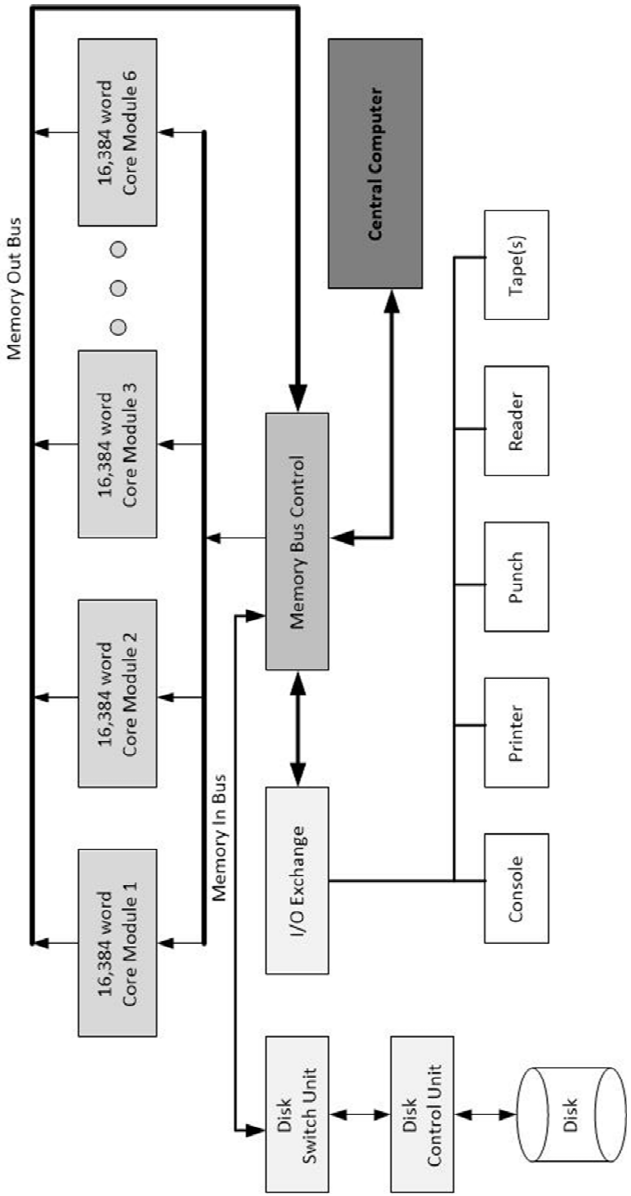

Figure 1-1. IBM STRETCH System Architecture
Source: Adapted from Bloch 1959

1.2 STRETCH ARCHITECTURE

Source: IBM 1960o

As Buchholz (1962) has already collected many of the key papers for the STRETCH, this section will provide a brief overview of some of the architectural concepts. Readers are referred to Buchholz's book, *Planning a Computer: Project STRETCH*, which is available at http://amturing.acm.org/Buchholz_102636426.pdf. A detailed description of the STRETCH architecture can be found in GA. Blaauw and FP. Brooks Jr. 1997. *Computer Architecture: Concepts and Evolution*, Addison-Wesley. Figure 1-1 depicts the STRETCH system architecture.

1.2.1 INTERPRETIVE CONSOLE

Most systems had a fixed operator's console which controlled the entire computer. Pressing the HALT or STOP button on the console terminated all activity in the computer until a START button was pressed. However, in multiprogramming systems, there was a need to communicate with a particular program while other programs ran unattended. This means that pressing the STOP button would only halt the currently executing program, rather than the entire machine.

In the STRETCH system, the operator console was treated as an I/O device and was not directly connected to the CPU. The switches and lights were not predefined and had no particular meaning except that determined by the executing program. This allowed multiple consoles to be connected to the STRETCH computer.

1.2.2 MEMORY PROTECTION SYSTEM

The CPU checked all references to memory. If the specified address was located in either a fixed area or a variable area, the access was suppressed and an interrupt was generated. The boundaries of the variable area were stored in two addresses in the fixed area. These addresses could only be changed when the interrupt system was disabled. These addresses were loaded each time a program switch occurred to a different user program. Thus, different areas of memory allocated to user programs - the variable area – could be protected depending on the user program storage requirements.

1.2.3 INTERRUPTS

The STRETCH interrupt system classified interrupts into five categories as presented in Table 1-3 (Codd, Lowry, McDonough, and Scalzi 1959).

Table 1-3. STRETCH Interrupt Categories

Category	Description
Signals	Hardware interrupts from I/O units, channels, and other CPUs.
Data Exceptions	Anomalous data conditions such as divide by zero, negative operands in square root operations, overflow and underflow, etc.
Result Exceptions	Anomalous conditions after certain operations such as lost carries, floating point values exceeding exponent ranges, or partial fields.
Instruction Exceptions	Errors in executing instructions due to faulty operands or undefined instructions.
Machine Malfunctions	Any of different machine errors occurring during the execution of instructions.

During concurrent execution, interrupts 2 through 4 were specific to the executing program. Each of these interrupts could be suppressed by a mask bit. Interrupt conditions 1 and 5 were general to all programs and could not be suppressed.

Each interrupt had a numeric identification. When an interrupt occurred, the ID was used as an index into an interrupt table to obtain an address for an interrupt handler. Control was transferred to the interrupt handler without changing the value of the program counter. Thus, upon entering the interrupt handler, the location in the user program at which the interrupt occurred could be saved for later analysis and/or return after the interrupt was processed.

The base address of the interrupt table was variable. Each user program might have its own interrupt table, which was accessed through a pointer. Only one interrupt table was active at any time.

1.2.4 SYSTEM CLOCKS

The STRETCH system was equipped with two system clocks that were usable by programs. The elapsed time clock was a 36-bit counter that was incremented once every millisecond. This clock could be read but not

changed by the program. It was used by accounting and logging programs to provide a time stamp for specific events.

The second clock, the interval timer, was a 15-bit counter that was automatically decremented by one every millisecond. The interval timer could be set and/or read by a program. An interrupt was generated whenever the timer reached zero. This clock specified a fixed amount of time for specific tasks, such as for executing a program in scheduling, or waiting for a response from some device.

1.3 STRETCH INSTRUCTION SET

Source: Bloch 1959, Brooks 1962

Brooks noted that because the machine would be memory limited, a rich instruction set would accomplish much with each memory access. He noted that the there was only one branch instruction, but a 6-bit code could select any one of 64 machine indicators, a bit to specify on or off conditions, and a bit to permit resetting of the indicator. There were few basic instructions, but many modifiers which meant the machine had fewer instructions than the IBM 709/7090 series, but many more distinct operations.

The STRETCH instruction set consisted of three formats as depicted in Figure 1-2.

The instruction set consisted of sets of instructions in nine different categories. It included a full set of instructions for single- and double precision arithmetic for floating point representations, variable length arithmetic – both binary and decimal – for integer arithmetic, and index modification and branching set. Bloch provides a table that depicted how the instructions were organized as depicted in Table 1-4 (Source: Adapted from Bloch 1959).

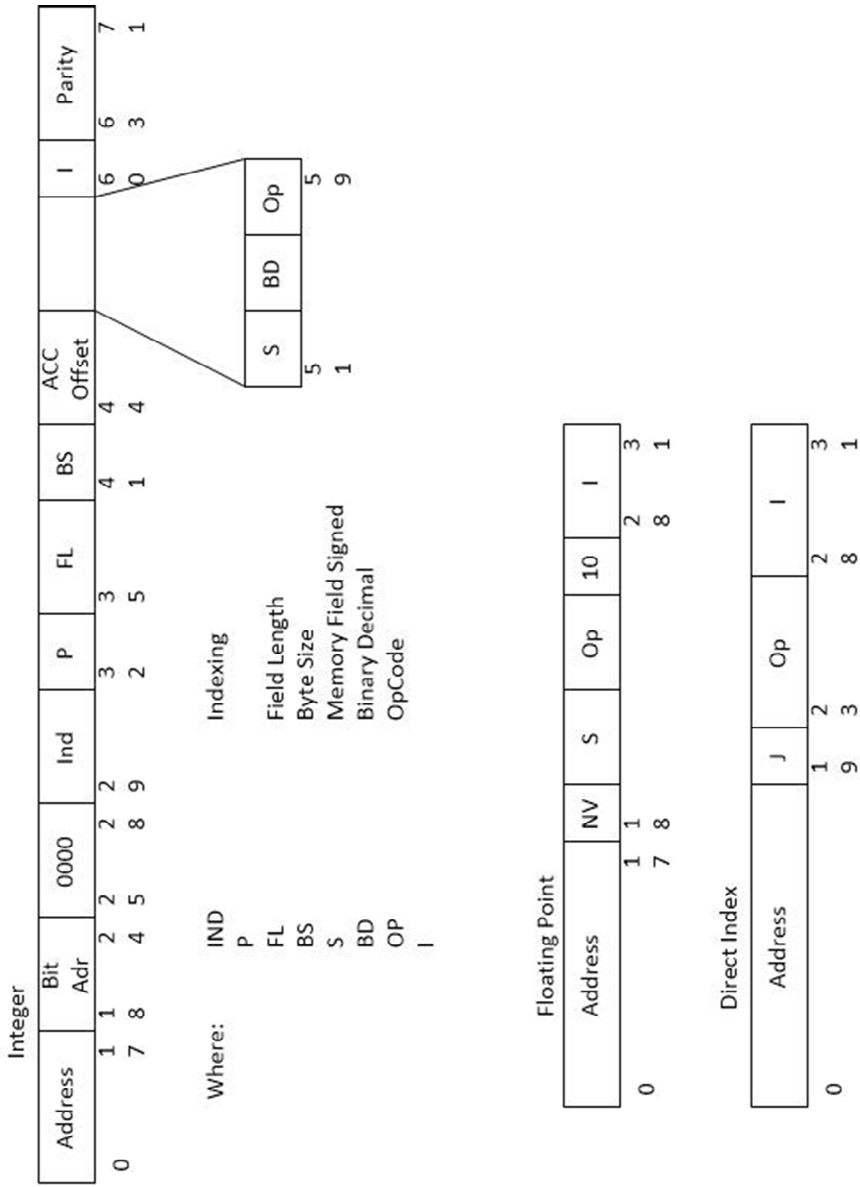

Figure 1-2. IBM STRETCH Instruction Formats
Source: Adapted from Bloch 1959

Table 1-4. STRETCH Instruction Vocabulary

Instruction Category	Class	Modifier	# Instructions
Variable Length Field Arithmetic	Binary Decimal	Signed Unsigned Same Sign Negative Sign	280
Radix Conversion	Binary/Decimal		32
Logic Connects			48
Floating Point Arithmetic	Normalized Unnormalized	Same Sign Opposite Sign Negative Sign Noisy Mode	240
Indexing Arithmetic	Direct Immediate Progressive		43
Branches	Unconditional Indexing Indicator Bit	 If (1,0) Set 0 Leave 0 Invert Bit	68
Transmit/Swap I/O Instructions			24
		Total	735

1.4 STRETCH SOFTWARE

IBM provided several software programs with STRETCH. STRAP was the STRETCH Symbolic Assembly Program. SMAC, the STRETCH Macro System, was an extension to STRAP that provided the programmer with more flexibility in writing assembly language programs. It included about 30 macro instructions and provided facilities for the programmer to develop their own macros and stored them in a library.

According to Sheldon (1963), the British Atomic Energy Commission developed a FORTRAN II compiler for STRETCH. This compiler generated

object programs rapidly, but the object code was not very efficient. IBM provided a FORTRAN IV compiler for STRETCH. However, this compiler operated almost three times slower than the FORTRAN II compiler operating on the IBM 7090.

1.5 STRETCH PROGRAMMING

STRETCH was designed to support multiprogramming whereby multiple programs could be accommodated in memory and assigned to different CPUs. Additionally, the hardware units, such as the channels and disk units were designed to operate independently of the CPUs once an I/O operation was initiated. The general idea, as Codd, Lowry, McDonough, and Scalzi (1959) noted, was to balance the workload across the hardware units of the machine to minimize computational bottlenecks.

To facilitate multiprogramming, Codd, Lowry, McDonough and Scalzi (1959) have identified six requirements as presented in Table 1-5.

Table 1-5. STRETCH Multiprogramming Requirements

Requirement	Description
Independence of Program Preparation	Programs should be independently written, compiled and linked to form an executable program. The decision as to which programs should be concurrently executed was made at runtime, not during compilation.
Minimum Information Required of the Programmer	The programmer should not have to provide any information for the program to run in multiprogramming mode. However, the programmer should be able to provide information to the program and constraints upon its execution such as an execution time limit and storage needs. Such information, could be used by the multiprogramming system to economically and efficiently execute the program.
Maximum Control by Programmer	Some features of the machine should not be controllable by the programmer, such as modifying the real-time clocks. But such constraints should be kept to a minimum and should not reduce the logical power available to the programmer.

Noninterference	Programs executing concurrently should not be allowed to introduce errors in other programs. This implied a separation of address spaces and memory protection. Programs that fail should not cause any other program to fail. Programs should not cause undue delay in the execution of other programs consistent with the scheduling algorithms of the multiprogramming system.
Automatic Supervision	The multiprogramming system must handle the complexity of concurrent program execution. In particular, control of hardware resources should be reserved to the multiprogramming system rather than the individual user programs. The multiprogramming system should be responsible for intercepting, handling and reporting machine malfunctions, programming errors, and operator mistakes to the operator. The scheduling algorithms of the multiprogramming system should operate automatically with minimal operator intervention except to adjust system parameters.
Flexible Allocation of Space and Time	The multiprogramming system should allocate space, time, and other resources, including devices based on the needs of the individual programs, not some predefined allocation based on the hardware configuration or operation.

1.6 STRETCH INNOVATIONS

Source: Bashe, Buchholz, Hawkins, Ingram and Rochester 1981, http://www.brouhaha.com/~eric/retrocomputing/ibm/stretch/

While the IBM STRETCH was not considered a successful machine in reaching its performance or financial goals, many innovative ideas were incorporated into the design as depicted in Table 1-6.