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

Copyright \odot 2019 by Stephen H. Kaisler

All rights for this book reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owner.

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

TABLE OF CONTENTS

List of Figures	ix
List of Tables	. xii
Acknowledgements	xvi
Introduction	1
Chapter One	3
Project Stretch	
1.1 Technical and Organizational Goals	4
1.2 STRETCH Architecture	7
1.3 STRETCH Instruction Set	10
1.4 STRETCH Software	12
1.5 STRETCH Programming	13
1.6 STRETCH Innovations	14
1.7 STRETCH Assessment	. 17
Chapter Two IBM 7030	19
2.1 IBM 7030 System Architecture	20
2.2 Central Processor	21
2.3 Main Storage	25
2.4 I/O System	26
2.5 IBM 7030 Instruction Set	27
2.6 Master Control Program	53
2.7 IBM 7030 Assessment	. 59
Chapter Three IBM 7040 Series	61
3.1 Central Processing Unit	63
3.2 IBM M44/44X	70
3.3 7040/7044 Operating System	71
3.4 IBM 7040 Assessment	80

Chapter Four	81
IBM 7070 Series	
4.1 IBM 7070 System Configuration	84
4.2 IBM 7070 Instruction Set	96
4.3 IBM 7070 System Software	107
4.4 IBM 7070 Programming Example	115
4.5 IBM 7072	115
4.6 IBM 7074	115
4.7 IBM 707x Assessment	117
Chapter Five IBM 7080	119
5.1 IBM 7080 Configuration	121
5.2 IBM 7080 Peripherals	135
5.3 IBM 7080 Instruction Set	145
5.4 Input/Output Control System	160
5.5 Autocoder III	170
5.6 IBM 7080 Assessment	176
Chapter Six IBM 7090/7094	176
6.1 IBM 709x System Configuration	179
6.2 IBM 7151 System Console	198
6.3 IBM 7094 Addressing	202
6.4 IBM 709x Instruction Categories	208
6.5 I/O Processing	230
6.6 IBM 709x Data Channels	232
6.7 Peripheral Devices	253
6.8 Direct-Coupled Systems	264
6.9 IBM and the Space Program	269
6.10 The 7090 and Missile Defense	271
6.11 Eliza/Doctor on the IBM 7094	271
6.12 IBM 7090 and ALGOL	272
6.13 IBM 7094 and Airline Reservations Systems	273
6.14 IBM 7090/7094 Series Assessment	274
Chapter Seven	276
IBM 7090/7094 Systems Software	_,0
7.1 SHARE Operating System.	276
7.2 Time-Sharing System Monitor	278
<i>c</i> ,	

Chapter Eight	281
CTSS	
8.1 The Supervisor	283
8.2 Command Format	285
8.3 CTSS Innovations	289
8.4 The End of CTSS	290
8.5 CTSS Assessment	291
Chapter Nine	292
SNOBOL	
9.1 SNOBOL Data Structure	294
9.2 Statements	295
9.3 Pattern Matching	296
9.4 A Few SNOBOL Applications	297
9.5 SNOBOL Assessment	297
Chapter Ten	299
IBM 7010 Data Processing System	
10.1 System Configuration	301
10.2 Magnetic Core Storage	301
10.3 Instruction Set	302
10.4 The IBM 7000 Series Assessment	303
Chapter Eleven	305
IBSYS – The Master Monitor	
11.1 System Monitor	307
11.2 IBJOB	320
11.3 Language Processors	327
11.4 I/O Control System	337
11.5 Debugging Package	340
11.6 The Loader	340
11.7 The Subroutine Library	350
11.8 The Utility Monitor	351
11.9 Other Subsystems	352
11.10 Operator Role	353
11.11 IBSYS Resources Usage	353
11.12 IBSYS Assessment	354

Table of Contents

Further Reading	. 355
Exercises for the Reader	. 358
Glossary	. 361
References	. 362
Index	. 370

LIST OF FIGURES

- 1-1 IBM STRETCH System Architecture
- 1-2 IBM STRETCH Instruction Formats
- 2-1 IBM 7030 System Architecture
- 2-2 IBM 7030 Arithmetic/Logical Unit
- 2-3 Instruction Format
- 2-4 Full-length Instruction Format
- 2-5 Floating Point Operation Format
- 2-6 Indicators for VLO and FPO
- 2-7 FPO Only Indicators
- 2-8 Integer Operation Format
- 2-9 Connective Operations Format
- 2-10 Direct/Immediate Index Instruction Format
- 2-11 Index Word Format
- 2-12 Branching Operation Formats
- 2-13 TRANSMIT Operation Format
- 2-14 I/O Instruction Format
- 2-15 Miscellaneous Instruction Format
- 2-16 I/O Control Table Relationships
- 2-17 Sample FORTRAN Deck
- 3-1 IBM 7040 at University of Saskatchewan
- 3-2 Simplified Data Flow for Arithmetic/Control Units
- 3-3 IBM 7040 CPU Front Panel
- 3-4 IBM 7040 OS Control Card Format
- 4-1 IBM 7070 SMS Card
- 4-2 IBM 7070 Data Processing System
- 4-3 IBM 7070 System Configuration
- 4-4 Record Definition Words in Table Example
- 4-5 IBM 7070 Data Channel Architecture
- 4-6 IBM 1414 I/O Synchronizer
- 4-7 IBM 7070 Instruction Format
- 4-8 Sample IOCS Program
- 4-9 IBM 7074 Data Processing System

- 5-1 IBM 7080 Data Processing System
- 5-2 IBM 7080 System Configuration
- 5-3 IBM 7080 Memory Banks 0 and 1 Structure
- 5-4 IBM 7080 Selecting the ASU
- 5-5 IBM 7080 Channel Word Structure
- 5-6 IBM 7080 CASU Word 15
- 5-7 IBM 7080 System Console
- 5-8 IBM 7153 Closeup
- 5-9 IBM 7621 Tape Control
- 5-10 IBM 729 Magnetic Tape Drive
- 5-11 IBM 7631 Instruction Format
- 5-12 IBM 1301 Disk Storage Unit
- 5-13 IBM 7080 Instruction Format
- 5-14 IBM 7080 IOCS Tape Table
- 6-1 Alfred E. Neumann from MAD Magazine
- 6-2 IBM 7090 System Configuration
- 6-3 IBM 7090 Data Processing System CPU
- 6-4 IBM 7302 Core Storage Unit
- 6-5 IBM 7090 CPU Registers
- 6-6 Index Adders Role
- 6-7 IBM 709x Select channel Instruction Format
- 6-8 Reset and Load Channel Instruction Format
- 6-9 Example of Magnetic Tape Writing
- 6-10 IBM 7094 System Console
- 6-11 IBM 7090 Instruction Formats
- 6-12 IBM 709x Floating Point Formats
- 6-13 MQ and Table Structure for Convert Instructions
- 6-14 IBM 7607 Data Channel
- 6-15 IBM 709x Data Flow
- 6-16 IBM 7909 Internal Data Flow
- 6-17 IBM 709x I/O Instruction
- 6-18 IBM 7320 Drum Storage Unit
- 6-19 IBM 7631 Order Format
- 6-20 IBM 729 Model II Magnetic Tape Unit Operator Console
- 6-21 IBM 704x/709x Direct Coupled System
- 9-1 The SNOBOL 4 Green Book
- 10-1 IBM 7010 Data processing System
- 10-2 IBM 7010 Character Format

X

- 10-3 IBM 7010 Instruction Format
- 11-1 IBSYS Software Architecture
- 11-2 IBSYS Core Dump
- 11-3 IBJOB Program Preparation Process
- 11-4 IBJOB Storage Map
- 11-5 IBJOB Program Execution Storage Map
- 11-6 Assembly Program and Cross-Reference Table
- 11-7 Overlay Structure
- 11-8 Overlay Storage Allocation

LIST OF TABLES

- 1-1 STRETCH Basic Characteristics
- 1-2 Project STRETCH Goals
- 1-3 STRETCH Interrupt Categories
- 1-4 STRETCH Instruction Vocabulary
- 1-5 STRETCH Multiprogramming Requirements
- 1-6 STRETCH Innovations
- 2-1 IBM 7030 Basic Characteristics
- 2-2 Reserved Memory Locations
- 2-3 IBM 7030 Addressing Modes
- 2-4 IBM 7030 Index Indicators
- 2-5 Integer Operations
- 2-6 Radix Conversion Operations
- 2-7 Connectives Operations
- 2-8 IBM 7030 Index Indicators
- 2-9 Direct Index Arithmetic Instructions
- 2-10 Immediate Index Arithmetic Instructions
- 2-11 Count and Branch Operation Formats
- 2-12 Transmit Instructions
- 2-13 Input/Output Instructions
- 2-14 Refill Operations
- 2-15 Data Reset Instruction
- 2-16 Miscellaneous Instructions
- 2-17 MCP I/O Control Tables
- 2-18 Selected MCP Control Cards
- 3-1 Arithmetic/Control Unit Registers
- 3-2 IBM 7904 Data Channel Registers
- 3-3 IBM 7040 OS Operational Control Cards
- 3-4 IBM 7040 MCP Subsystems Control Cards
- 3-5 IBM 7040 MCP Unit Assignment Control Cards
- 4-1 IBM 7070 Characteristics
- 4-2 Sample IBM 7070 Configuration
- 4-3 IBM 1414 Commands

- 4-4 Operations Involving Accumulators
- 4-5 Branch Instructions
- 4-6 Compare Instructions
- 4-7 Index Word Instructions
- 4-8 Block Transmission Operations
- 4-9 Block Transmission Instructions
- 4-10 Table Lookup Instruction
- 4-11 Autocoder Advantages
- 4-12 Standard I/O Subroutines
- 4-13 Standard Macro-instructions
- 4-14 File Specification Entries
- 5-1 IBM 7080 Basic Characteristics
- 5-2 IBM 7080 System Checks
- 5-3 Compatibility Model Selection
- 5-4 IBM 729 Magnetic Tape Drive Characteristics
- 5-5 IBM 7631 Instructions
- 5-6 IBM 7631 Instructions Fields
- 5-7 Arithmetic Instructions
- 5-8 Shift Instructions
- 5-9 Data Transmission Instructions
- 5-10 Decision Instructions
- 5-11 Status and Mode Instructions
- 5-12 Control Instructions
- 5-13 Read and Write Instructions
- 5-14 IOCS Sections
- 5-15 IOCS Communication Mechanisms
- 5-16 IBM 7080 IOCS Tape Table
- 5-17 Descriptive Macro-instructions
- 5-18 Functional Linkage Macro-instructions
- 5-19 Modification Macro-instructions
- 5-20 Area Definition Statements
- 5-21 Selected General-Purpose Macro-instructions
- 6-1 IBM 709x Basic Characteristics
- 6-2 IBM 709x Programmer Accessible Registers
- 6-3 IBM 709x Non-Accessible Registers
- 6-4 Channel Command Words
- 6-5 I/O Synchronization Instructions
- 6-6 Relocation Mode Instructions
- 6-7 Protection Mode Instructions

- 6-8 IBM Model 7151 Register Displays
- 6-9 IBM 7151 Console Switches
- 6-10 IBM 7151 Console Panel Indicators
- 6-11 IBM 7151 Panel Lights
- 6-12 IBM 7070901 Instruction Formats
- 6-13 IBM 7090 Types of Addressing
- 6-14 Specifying Index Registers
- 6-152 Complement Arithmetic
- 6-16 Fixed Point Instructions
- 6-17 Control Instructions
- 6-18 Floating Point Instructions
- 6-19 Shifting Instructions
- 6-20 Word Transmission Instructions
- 6-21 Transfer Instructions
- 6-22 Index Transmission Instructions
- 6-23 Logical Instructions
- 6-24 Sense Indicator Instructions
- 6-25 Conversion Instructions
- 6-26 Magnetic Tape Instructions
- 6-27 IBM 709x CSU-Relevant Instructions
- 6-28 Types of I/O Instructions
- 6-29 IBM 7607 Registers
- 6-30 Data Channel Indicators
- 6-31 IBM 7909 Interrupt Conditions
- 6-32 IBM 7909 Data Channel Registers
- 6-33 IBM 709x Instructions
- 6-34 IBM 7909 Commands
- 6-35 IBM 7909 Sense Conditions Word 1
- 6-36 IBM 7320 Usage
- 6-37 IBM 7631 Orders
- 6-38 IBM 729 Operator Console Switches and Lights
- 6-39 Job Processing
- 6-40 IBSYS Modules
- 6-41 DCMUP Utilities
- 8-1 Selected CTSS Console Commands
- 8-2 CTSS Innovations
- 10-1 IBM 7010 Basic Characteristics
- 11-1 Selected IBSYS Control Cards

- 11-2 IBSYS Functional I/O Unit Assignments
- 11-3 Disk/Drum Utility Routines
- 11-4 IBJOB Control Card Options
- 11-5 IEDIT Control Card Options
- 11-6 IBFTC Compiler Options
- 11-7 FORTRAN IV Logical Unit Assignments
- 11-8 COBOL Control Card Options
- 11-9 \$IBMAP Control Card Options
- 11-10 Selected MAP Pseudo-Operations
- 11-11 IBLDR Options
- 11-12 LOADER Control Cards
- 11-13 \$ORIGIN Control Card Options
- 11-14 System Subroutines
- 11-15 IBSYS Utility Routines

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

Introduction

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.

Characteristic	Value/Explanation
Internal	Fixed Point Binary; Floating Point Binary
Representation	
# 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

Table 1-1. STRETCH Basic Characteristics

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
TT' 1 1	computers.
High-speed	A high-speed floating point unit to support
Arithmetic	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	External data storage units would have a capacity
Magnetic	of 2 million words.
Disks	

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).

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

Table 1-3. STRETCH Interrupt Categories

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

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

Table 1-4. STRETCH Instruction Vocabulary

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.

Requirement	Description
Independence of	Programs should be independently written,
Program Preparation	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	The programmer should not have to provide
Information Required	any information for the program to run in
of the Programmer	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	Some features of the machine should not be
Programmer	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.

Table 1-5. STRETCH Multiprogramming Requirements

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	The multiprogramming system must handle
Supervision	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	The multiprogramming system should
of Space and Time	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.