AN/FSQ-7—1954-197?

no brochure, only performance

by Douglas L. Jordan

"It has been determined that this property is no longer needed:

1. Miscellaneous Metals, Scrap: Consisting of electrical power and control cable, and cable support items including the following:
   . . . Estimated quantity 93,200 pounds.
2. Miscellaneous Metals, Scrap: Ferrous and non-ferrous. Consisting of cannibalized components of AN-FSQ-7 computer.
   . . . Estimated total weight 180,000 pounds."

GSA announcement of surplus materials

The AN/FSQ-7's were fantastic machines before they became nearly 300,000 pounds of scrap metal. They occupied nearly 20,000 square feet of floor space, with another 12,000 square feet for display consoles, and nearly 10,000 square feet for telephone input equipment. Nearly 30 of the duplex monsters were built, but their day is now passing. Already one-half of the number have been scrapped, and the others are showing their age. In their prime, however, these beasts required an army to feed them, and perhaps some of that departed army would like to recall for a moment the early days of SAGE and the SAGE computer.

"Where do I find the computer?" were often the first words of the new programmer trainee as he stood in the midst of the AN/FSQ-7 computer frames (Fig. 1). These were trainees in the purest sense—formerly teachers, farmers, machinists, preachers, professional cardplayers, professional ballplayers, draftsmen, draft dodgers, and a few, a very few, programmers. They came, learned basic programming, and went on to other activities in and out of the programming profession. Today it would be surprising to find a programming concern that had no one who had worked on the Q-7 at some time.

The army was assembled rapidly. In the summer of 1955 it numbered barely 100. A year later it was approaching 1000. During 1957, it increased at the rate of nearly 50 per week with predictable dislocation to management organizations. The popular phrase of the day was, "If my boss calls, get his name."

In those early days, it was not difficult to become an expert. The story that comes to mind is that of the ex-teacher who completed the basic IBM class in coding and reported to his new supervisor. He was handed the specification for the digital display package with the words, "If you have any problems just come ask." The value of the proffered help became questionable the next week when the supervisor came in to ask for clarification on the makeup of a particular display.

It was the practice to load the entire system from cards, a seemingly endless task, given the tendency of the card reader to drop bits. Because of this, all computer users were ordered never to write or erase the Aux Memory drums which held the program. Programmers being what they are, the Aux Memory was regularly destroyed, until a manual interlock switch was placed on the drums. At about the same time, the program went onto tape and the problem ceased to exist; but the interlock is there today, unused and nearly forgotten—a monument to the unwillingness of programmers to conform.

The genealogy of the Q-7 has a Biblical ring: Whirlwind begat XD-1, begat Q-7 (8K), begat Q-7 (69K).[1] The XD-1 was an experimental prototype used during the early development of the SAGE programs. Like it, the early Q-7's were built with 8K of memory, but this soon proved too small in relation to the 150K of drum storage, and by 1960 all Q-7's had been retrofitted with 69K of core memory. With this modification, and the inactivity and I/O interrupt circuits which were added later, the Q-7 was complete. Its statistics were almost unbelievable in 1957 and they remain impressive even today: 69K of core memory all directly addressable; 150K of drum storage; add four 16-bit numbers in two memory cycles; multiply in three memory cycles; divide in nine memory cycles; I/O operations proceeding independently of the central processor. The Q-7 was the heart of the first real-time command and control system. Designed to accept, process, and display large quantities of digital data, it was capable of simultaneously driving 100 display consoles, while accepting data from 100 on-line operators and 30 remote computers, and providing output data and commands to these same computers, and up to 25 Teletypes.

Like its contemporaries, the 704 and 709, the Q-7 was designed using vacuum-tube technology (Fig. 2). Its thousands of vacuum tubes forced speeds which are now surpassed by an order of magnitude. However, a memory cycle of 6 usec was twice as fast as the 704, and nearly up to that of the 709. Basic to the Q-7 design was a division of labor so that no one system would be overloaded. This is best illustrated by its method of interfacing with the external world—during Air Defense operations, all input and output messages were routed via the buffer drums so that the

[1] The line continued on to the AN/FSQ-8, a near twin of the Q-7 used in SAGE Combat Centers; the AN/FSQ-31, used by SAC; and the AN/FSQ-32, the original SDC time-sharing machine.

Mr. Jordan is presently a member of the System Development Corp. technical staff with the AWACS project office. He has been connected with SAGE for 10 years, working with the Detroit SAGE Sector, the Santa Monica test team, the Puerto Rico Medical Center project and the SAGE operational group. He has an AB from Harvard and an AM in teaching, and is a member of ACM.

independent input and output systems could process them without interfering with the central computer. Similarly, all displays were placed on drums so that both the central computer and the display system had access on a noninterfering basis. Thus the Q-7 was unchallenged at its own game.

On the other hand, its capabilities as a general purpose processor had certain limitations. The on-line printer ran at 150 lines per minute (Fig. 3) and the tapes at 3000 words per second. These Q-7 capabilities were found both interesting and confusing by the programmers. For example, many programmers learned by experience that the Q-7 was perfectly capable of clearing a print image in the time that

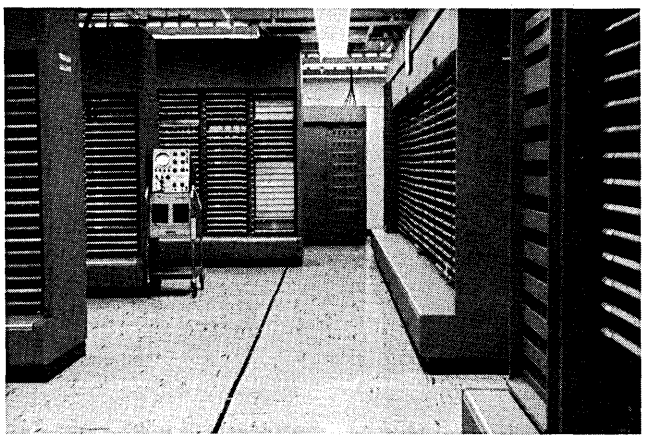

Fig. 1. One corner of the AN-FSQ-7 main frame.

elapsed after the print command was executed but before the data were transferred to the printer. These same programmers often "processed" data that had yet to be stored in memory by the tape, drum, or card input operations. The results of such processing were often remarkable, but never useful.

The tape drives on the Q-7 were, by design, secondary storage devices. Their reliability was below that of other parts of the machine, and their capabilities had some nota-

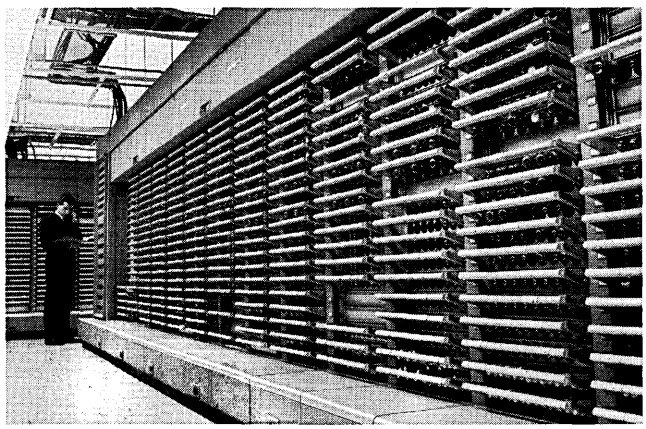

Fig. 2. A sample of the AN-FSQ-7's vacuum tubes.

ble consequences. During the early years, many tapes were snapped by programmers who forgot that a tape backspace command would be executed even if the tape were moving forward. This problem no longer exists, but only because mylar tape will stretch rather than snap.

The secondary character of the tape drives came to light most frequently during the operation of the simulation system. The sim system design required that there be five records for each five seconds of time which was simulated. All the required data was placed in only two records, and the sequence was completed by three dummy records. The chant "KIS, TDS, Dummy, Dummy, Dummy" was always good for a laugh. It also represented a rhythmic sequence of short records which created endless problems for the tape drives and, on at least one occasion, caused a side panel to fall off.

The Q-7 had some data characteristics which set it apart from the rest of the world of computers. Being designed to solve problems involving coordinates, its data word was always considered to be two signed half-words. This made the problem of calculating with positional data a relatively rapid one, since both the X and Y coordinates could be processed simultaneously. On the other hand, it caused a large number of unused bits to ride along during the majority of data processing. Of more concern and consternation to the Q-7 programmer was its use of one's complement arithmetic. Most of the neophyte programmers who first met the machine had never heard of the existence of a negative zero—the majority of the old-timers often wish they had never heard of it. Many programmers have built a reputation for trouble-shooting solely on their constant search for instances in which the coder forgot that a few bits taken from the middle of a negative zero will look nothing at all like zero. The behavior of negative zero has undoubtedly caused more testing and error correction than any other single characteristic of the machine.

The testing of computer programs was even more of an unknown art in 1958 than it is today. Everyone had his own ideas, and everyone's ideas were tried. One mathematician felt that it would be possible to enter random inputs and then to verify performance by predicting the outputs on a statistical basis. His effort bore no fruit; however, the basic

Fig. 3. The lethargic printer and a view of the AN-FSQ-7 console.

idea was implemented by a programmer to test out the switch programs. He labored for several weeks producing an input tape which contained randomly selected combinations of all possible switch actions, legal and illegal. His test was finally run (at midnight naturally), and a huge stack of computer printout was delivered to him the next morning. It was only then that he learned that his first, randomly selected, switch action had turned off the simulation input tape!

If testing was an area of individuality, documentation was even more so. Everyone had his pet way for showing the operation of a program. Register reference diagrams, branch diagrams, functional flow diagrams, and dynamic flow diagrams were only a few of the methods which were tried. All had their uses, and most have passed into history. Now the machine that spawned the diagrams—and also had its uses—must follow them. ■