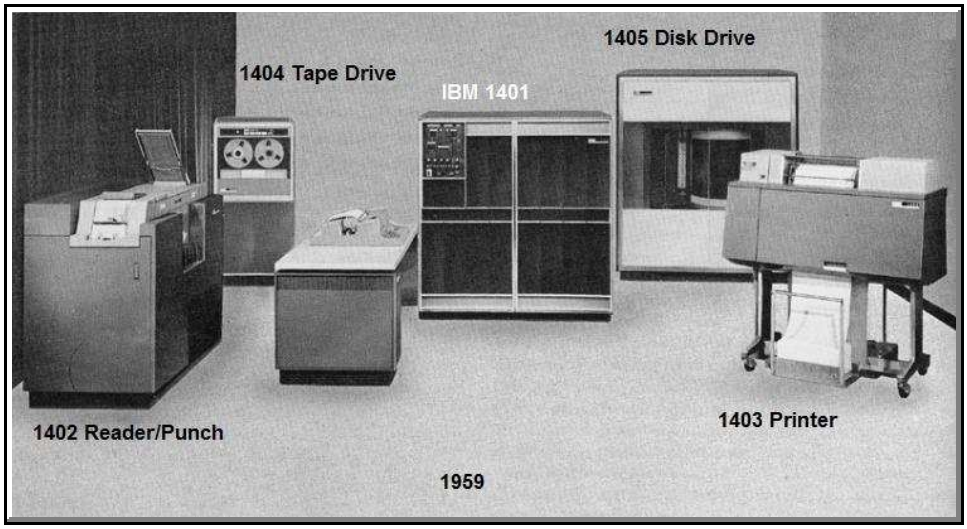
# The IBM 1401




I joined IBM in the late summer of 1959, and that fall, took a 1-week class on the programming of the 1401. I then taught several courses to customers for the following 6 months. I also programmd it for 6 more years and found it to be an intriguing machine.

The first noticeable thing was the 1403 printer which printed at 600 lines per minute. Five sets of printable characters were contained on a chain that rotated at high speed past a set of 132 hammers. When the appropriate character passed the place where it was to be printed, the hammer behind that position would fire, making the imprint. It was quite noisy, in spite of the acoustically insulated box around it, and someone programmed a print out-that caused the printer to play Beethoven's Ode to Joy – I kid you not.

A separate box contained a card reader that could read 800 cards per minute, and a card punch that could punch at 400 cards per minute. Note the card tray sticking up from the reader box.

Most, if not all of the preceding computers had memory units of words, all of the same size. Decimal based computer words usually were 10 or so digits long with a sign. The binary machines had words of 32 or, in the case of the IBM 7090, 36 bits. So typically, they could handle integers as large as +/-9,999,999,999 or $2^{31}$, which was +/-36,364,056,439. Arithmetic in prior computers was carried out in special registers on these words. If one wanted more digits in a calculation, one would have to do operations on multiple words, and handle the overflows or carries from the lower order to the higher order word.

But the 1401's memory was made up of 8-bit bytes, as current computers are – but with a twist. Six bits were used for data, 4 of them were used to represent a numeric digit. Two of them reflected the X, and Y holes inherited from the punch-card, were combined with the digits to produce alphabetic and special characters. There was a parity bit for catching errors, and an 8th bit called a *word mark*. From the programmer's point of view, there were no special registers for arithmetic. Everything

was done in memory-to-memory fashion[1]. Arithmetic was actually carried out using a one-digit adder, byte-wise, right to left, with the computer handling the carries from one byte (digit) to the next automatically. The word mark indicated where the left end of the participating data occurred. A collection of a group of bytes delimited by a word mark was called a *field*. An instruction could then add two fields of any length together. There is no doubt that the 1401 could compute 30 factorial with the simplest code of any computer before or since (well, except the IBM 1620 which came out at the same time and worked on the same principle).

So a programmer could vary the lengths of the data being operated on. Three bytes could be used for a person's age. Twenty seven bytes could be used for an astronomical calculation. This was a very important capability because the memory sizes on the 1401 were small – minuscule by today's standards. One could initially purchase a 1401 with 1.4K, 2K or 4K memories! Later on this was expanded to 16K, the largest it ever got.

But that's not all. Instructions were also variable length, ranging from 1 to 8 bytes. Instructions had a 1-byte mnemonic[2] op-code and one or two 3-byte addresses. The 3 digits could specify addresses 000-999, The X and Y bits of the high order digit, indicated which of the four millennia, and those bits of the low order digit specified one of 3 index registers. Of course, the Autocoder assembler took care of the formatting.

An unconditional branch instruction took 4 bytes. A conditional branch took an additional byte to specify the condition for branching. There was also a compare and branch instruction that was 8 bytes long.

Furthermore, there was a concept of chaining consecutive instructions, in which the instructions had nothing but the 1-byte op-code. This was particularly useful when one was accumulating a series of numbers located next to each other in memory. One would start with the fields in the highest memory location and add (subtract or move) a pair. The address registers would decrease as the operation proceeded, and be left in that state at the end of the operation. If an operation had no operands, it would use the values left in the address registers. A sequence might be

```
M FROM,TO
M
M
etc
```

Specific low memory locations were allocated for input, output and indexing. Locations 000-079 were for the card reader, 100-179 for the card punch, and 200-231were for printing. I forget where the 3 index registers were, but at 3-bytes each, they could have been at 080-088.

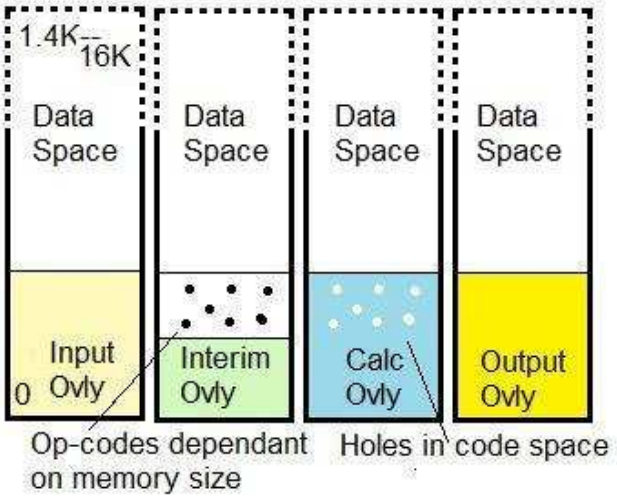
The small size of the computer led to some very innovative coding practices. In particular, a FORTRAN compiler was written for the 1401. Traditionally, the compiler resides in memory, reads the source program a line at a time from an input file, and produces object code into an output file. With the 1401 the source program resides in memory and many compiler overlays are passed against it, morphing it into the object code. The

[1] These types of instructions were included in the instruction set of the IBM 360.

[2] Add was `A`, Subtract was `S`, Move was `M` and Branch was `B`.

compiler came as a deck of cards that nearly filled the reader card tray. Fortunately, there was a utility that allowed it to be run from a tape drive.

One of the first programs I wrote for the 1401 was a program that solved multiple simultaneous equations. The size of the problem was dictated by the amount of memory available on the computer, with the minimum being 2K (for the program). The program was divided into 3 parts, each of which overlaid the others in memory. The first part read the data and converted into an internal form. The second part solved the equation(s). Arithmetic was done by floating point subroutines that I wrote. The third overlay printed out the results. The source data was sandwiched between the first and second parts of the program deck.




I wrote a simpler but much more useful and widely used 1401 program that would convert card-to-tape, tape-to-printer and tape-to punch. It was used in conjunction with the many IBM 7090 computers that came on line in the 1960s. A 729 tape drive was dedicated to each of the three functions.

The 1401 had 6 sense-switches on the control panel. The program operated in an endless loop, checking the sense switches each time around. A switch was dedicated to each of the 3 functions, and when one was turned on, a single card read, punch or line of print would be processed each time around the loop. When they were all were turned on, it was quite a sight to behold, with the card reader, punch and printer with their associated tape drives going all at one time. A fourth switch was used for backing up the tape associated with a single running operation. This was usefully if a card or paper jam occurred. (A strange side effect of this was that one could print a file backwards.) The program, called P3, was improved (and renamed P4) by the addition of a library capability, in which 7090 programs could be stored on tape and inserted into the card-to-tape stream upon recognition of a special card in the reader.

Other interesting programs I wrote for the 1401 were:

A compiler that translated a language (APT) with algebraic expressions written in reverse polish notation.

A program, borrowing the P4 library idea, that loaded 1401 programs from a tape to be run with data following the program ID card.

A program that tested English-like expressions to determine whether they were true, false, tautological, or indeterminate.

An economics game simulating a totally free market, and demonstrating that in time all players would arrive at a no-profit situation.

H J Myers – written12/7/2009, amended 10/23/2010.