The Memory Circuits of a Digital Computer

Jeff Folkman

June, 1970

THE MEMORY CIRCUITS OF A DIGITAL COMPUTER

Section I - Computer Memory

There are basically two types of electronic computers - analog and digital. The analog computer is based, in concept, upon analogs, proportional relationships between numbers, while the digital computer is based upon the relationships between actual digits. It is the digital computer with which this report is concerned.

The digital computer is made up of six basic parts of sections - input, output, control, arithmetic and logic, internal memory, and external memory.[1] Each of these parts is of equal importance because a computer could not function at its optimum performance level without all of them.

Two of the six parts of a computer have the ability to store data, these being the internal memory and the external memory. Before I discuss the difference between these categories of memory devices, it is probably best if background material on computer memories in general is given.

The memory of a digital computer has one function: the retention of data for immediate or future use, either in the form of binary numbers or in the form of symbols easily converted to binary numbers.[2] Although most computer programs deal in decimal numbers, the computer does its calculations in binary, a number system that uses only two digits - one and zero. The computer will automatically convert numbers from decimal to binary. These ones and zeros can also be made to correspond to a switch being on(1) or off(0), to a positive voltage(1) or a negative voltage(0), or, in fact, to any device that is bistable (having two distinct states). A light switch is an example of a common bistable device; it is either on or off. Consequently, the memory in a computer remembers only groups of ones and zeros.

The fact that a computer can remember only ones and zeros does not limit the number of available memory devices. The most important of these devices are magnetic tapes, magnetic drums, magnetic disks, magnetic cores[3], punched cards, and punched paper tape.[4]

As one can see, many memory devices are based on the principles of magnetism. A magnetic tape system is very much like a home tape recorder, except that the tape drive mechanism has the ability to "search" the tape for a specific piece of data.[5] A magnetic drum is a cylinder, that rotates at speeds up to 15,000 RPM. It is coated with an iron alloy that is magnetized by the many record heads (they can also read) that surround the cylinder, or as it is normally called, the drum.[6] A magnetic disk is a stack of eight to sixteen disks which resemble phonograph records, covered with a ferrite material, that revolves at very high speeds with movable record and read heads that fit between the separate disks of each stack.[7] Magnetic cores are "toroids (small washers) of a magnetic material such as magnesium manganese or nickel-zinc ferrite...that can be magnetized in either a clockwise or counter-clockwise direction."[8] Punched paper tape is the same as the tape used to feed the computer at U.S.

When an engineer designs a computer he must be familiar with the many types of memory devices mentioned above. Since each of them has its own peculiarities, engineers have classified memory devices according to these peculiarities. Each is classified in terms of capacity, access time, kind of access, and permanence.[9]

Before discussing the meaning of the terms mentioned above, one concept should be fully understood - that of memory layout. In any type of memory system, each place where data can be stored has an address. This address tells the computer where to store data and where to find a specific piece of data. This address can be an actual number or it can be a label. For example, the command:

```
SEARCH, ADD
```

tells the computer to find the memory location labeled ADD on a magnetic tape.

Now the meaning of access and access time can be discussed. There are two types of access - random and sequential. Random access is the ability to "go to" any specific memory address within a pre-determined and constant span of time.[10] Sequential access is the ability to "go to" any specific memory address, but not within a constant time span, no matter what the address is.[11] An example of a sequential access memory is a magnetic tape system. If the needed piece of data is at the end of the tape, the tape drive must wind through the whole tape to get to that needed piece of data.

The capacity of a memory system only tells how many pieces of data that the system can hold. The capacity varies greatly from memory system to memory system[12] and is usually measured in bits. The word "bit" is "a contraction of the words Binary digit."[13] The bit is the smallest unit of data that a computer can handle.[14] In many computers there are 32 or 64 bits to a memory location.

Permanence only refers to whether the stored data can be erased or changed as part of normal computer operation.[15]

We now come to an important distinction in computer memories - that of internal memory versus external memory. Internal memory is the basic memory unit with which a computer operates.[16] It is usually a random access system, with a very fast access time. Its capacity is usually small compared to that of an external memory.

An external memory usually "serves as a kind of a 'reference library' for the computer, storing large quantities of data (and programs) that are only occasionally called for."[17] It is also used by the programmer to store large amounts of data that are not accessed many times during the running of a program. The external memory is characterized by slow access time, large capacity, and sequential access.[18]

Section II - Register Memory

All the memory systems mentioned in section I are part of either the external memory or the internal memory. In addition there is one other type of memory that is in neither of these sections, the register.

There are usually 16 or more registers in a computer, of which 15 are normally accessible to the user. These registers are located in the arithmetic and logic section of the computer and are used as a place to do calculations.[19] For example, the program:

```
ZAP  10
ZAP  11
L    10,ADD1
L    11,ADD2
AR   10,11
```

would tell the computer to set registers 10 and 11 to zero(ZAP). Then the number at address ADD1 would be put into register 10, and the number at address ADD2 would be put into register 11("L" means load). Finally, the contents of register 11 would be added to that of 10 and the result would be placed into register 10("AR" means to add registers).

The remainder of the registers are used for either of two purposes--as a place for the computer to keep track of such information as time, last address accessed, programs, errors, etc., and as buffers. "The purpose of the buffers is to compensate for inequities in the speed between two portions of the computer system when information must be passed from one to the other. For instance, the buffer might receive a group of numbers at high speed from the computer and hold them until used at the lower rate by a printing device."[20]

No matter what its use, there are basically two types of registers, parallel and shift.[21] A parallel register is one that receives all the information to be stored at one time. A shift register is one that stores data one bit or word at a time. For example, if a computer had a shift register that held four words of data, each of the four words in the register would have an address of 0,1,2, or 3. When data is stored in the register, the first word is placed in location 0. The second word follows within a few microseconds and is placed in location 0, while at the same time the data that was in 0 is shifted one word to the right to location 1. When the third data word is stored in location 0, each of the two words already stored is shifted to the location at the next higher address. This process will continue until the register is filled.[22]

Until now, registers have been discussed as being entities in themselves; but, in reality, they are made up of combinations of electrical devices called flip-flops, of which only the simplest will be discussed.

A flip-flop is a bistable device that is stable in either state until an external pulse switches it to its other state.[23] A familiar example of a flip-flop is a light switch. It is stable in either state until the toggle is moved, at which time the switch becomes stable in the other state.

One of the states of a flip-flop corresponds to the bit one and the other to zero. Since a flip-flop can hold only one bit, there are as many flip-flops in a register as there are bits of data.

Flip-flops are characterized as having a short access time[24] and a low capacity (one bit). Because of its low capacity and relatively high cost, it is never used as the main internal or external memory of a computer.

Section III - A Model of a Register Memory

The register memory that I built has eight two-bit registers. Their addresses are 000,001,011,100,101,110, and 111. Each register can store the number 00,01,10, or 11. This is because the registers are limited to two digit numbers.

As was mentioned in the section preceding, the register memory is made up of flip-flops. For this model, the latching-relay was chosen to act as a flip-flop. A latching-relay is a relay that has two coils. When one coil is pulsed, all four of its contacts close. When the other coil is pulsed the four contacts open. Thus, a latching-relay also acts as a four-pole, two-throw switch.

My model has sixteen latching-relays, two for each register; each one is used to hold a separate bit.

Because almost all of the wiring in the model is not directly used for the registers (most of it is for input and output), the schematic diagram of the wiring will not be given. If one is interested in the actual wiring of a register, one should read some of the books listed in the bibliography.

The actual means of working the model is quite simple. First, turn the power on. The light labeled power should go on, if not, un-plug, something is wrong. To write a number into a register, set the address switches to the register number. Be sure to throw the switches completely up for a one and down for a zero. Next, set the read-write switch to "write". To clear the register of any previous data stored in it, press the load button. Finally, press the data buttons, depending upon the data to be stored. To read data from a register, set the address switches to the proper address and set the read-write switch to "read". The output lights will light according to what is stored.

The first eight columns of lights on the top of the panel correspond to the eight registers. They are there mainly to remind the user that the registers will store data indefinitely, even though power goes to the register for only a few hundredths of a second.

The remaining four lights are used to show which register is being accessed and whether the power is on.

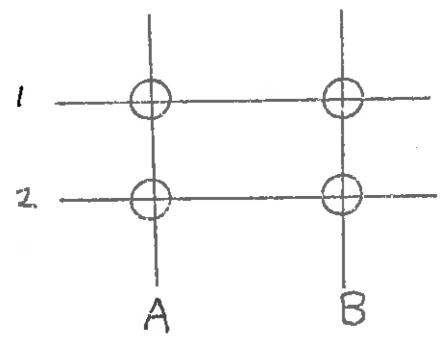
Section IV - Core Memory

The core memory is based upon the magnetic properties of small ferrite cores. If a positive current is applied to a wire that goes through a core, the core will become magnetized in one direction. If a negative current is applied to that same wire, the core will become magnetized in the other direction.[25]

The above method is great--if one just wants to show how to magnetize a core, but how are the cores arranged to make a working core memory?

In a core matrix four wires go through each core.

Two of the wires are used for accessing that core. For example, below is a diagram of a small core matrix:



If half the necessary current needed to magnetize a core were applied to wire 1, and if this same amount of current were applied to wire A, the core that is at their intersection would become magnetized. The cores at the intersection of 2 and A, and 1 and B would only get half the necessary current and would not magnetize.[26] This is called coincident current switching.[27]

If a third wire were passed through all of the cores in a plane, it would be possible to tell the value (one or zero) of a specific core, which depends, as mentioned before, upon the direction in which that core is magnetized. If a specific core is fully selected, that is, both of its address wires have current applied to them with negative current, a current would be generated in the third wire, assuming that a "one" was stored in that core. This is because a current is generated in any nearby conductor when a magnetic field changes states or direction. If the core were in the zero state, and the core were fully selected with negative current, no current would be generated in the third wire, because the magnetic field in that core did not change states. This phenomenon occurs only when a positive current represents a "one". The third winding is called the sense winding.[28]

Now, if many of these planes are wired together, we have a core matrix. To see how one works, assume that a computer uses a two by two by three matrix. This means that it has two times two, or four, memory locations, each of which can hold a three digit number. Each of the three digits of a number stored in this matrix is in a different plane. If the computer were programmed to store the number "101" at location 1,A (using the diagram shown earlier), each of the cores in that location in each of the three planes of the matrix would be fully selected. This means that the number "111" would be written into storage, which is not what we want. To stop this from happening, a fourth wire goes through each core in a plane. If a zero is to be stored in the fully selected core of a plane, the fourth winding of that plane is pulsed, at the same time that the core is fully selected, with a negative current equal to half the necessary current needed to magnetize a core. Now, if one adds the total current applied to the fully selected core:

$$1/2H + 1/2H - 1/2H = 1/2H$$

("H" is the amount of current needed to magnetize a core), one sees that the actual current applied to the fully selected core is not enough to magnetize it. Consequently, it is possible to store a zero. This fourth winding is called the inhibit winding.[29]

Section V - A Model of a Core Memory

For my model of the core memory, three relays are used to represent each core. Two of these relays are normal four-pole two-throw relays, while the third is a latching-relay.

These relays are wired in such a manner that one of the normal relays corresponds to the numeric coordinate, and the other, the alphabetic coordinate, on the diagram shown earlier. The latching-relay is wired to act as the inhibit winding and also to clear any stored data out of the location that the relay is in.

This model has the capability of storing four two digit binary numbers. This corresponds to a core matrix of two by two by two. For ease of accessing, I have numbered the various locations 00,01,10, and 11.

The method of operation is the same as the register memory. But, there are two things to watch for. When a "one" is stored, notice that the two non-latching relays, of the location that the number is being stored in, change positions. When that happens, that location is being fully selected. Also, once a number is stored and the address switches have been changed, the non-latching relays fall back to their non-energized position. This shows that the pulse used to magnetize a core is only a momentary one, lasting a few microseconds.

Section VI - Conclusion

The material presented in this report only scratches the surface of the amount of information that can be learned about computer memory. If one wants to delve further into this subject, I suggest reading the books listed in the bibliography and also the books listed in the bibliographies of the books that I read.

Footnotes

1. Allan Lytel, ABC's of Computers (New York, 1966), pg. 13.
2. Wayne C. Irwin, Digital Computer Principles (New York, 1960), pg. 133.
3. *Ibid.*, pp. 121-129.
4. Ronald M. Benrey, Understanding Digital Computers (New York, 1964), pg. 128.
5. Lytel, pg. 104.
6. *Ibid.*, pp. 105-106.
7. *Ibid.*, p. 109.
8. *Ibid.*, p. 96.
9. Benrey, pp. 126-127.
10. Technical Education and Management, Inc., Computer Basics (New York, 1962), pg. 35.
11. Benrey, pg. 126.
12. *Ibid.*
13. *Ibid.*, pg. 42.
14. *Ibid.*
15. *Ibid.*, pg. 127.
16. *Ibid.*
17. *Ibid.*
18. *Ibid.*
19. *Ibid.*, pg. 83.
20. Irwin, pg. 115.
21. *Ibid.*, pp. 157-158.
22. *Ibid.*, pg. 158.
23. Mitchell P. Marcus, Switching Circuits for Engineers (New York, 1967), pg. 233.
24. Irwin, pg. 152.
25. James D. Fahnstock, Computers and How They Work (New York, 1959), pp. 148-149.
26. Irwin, pg. 130.
27. Technical Education and Management, Inc., pg. 31.
28. Thomas C. Bartee, Digital Computer Fundamentals (New York, 1966), pg. 222.
29. *Ibid.*, pg. 220.

Bibliography

Bartee, Thomas C., Digital Computer Fundamentals, McGraw-Hill Book Company, 1966

Benrey, Ronald M., Understanding Digital Computers, John F. Rider Publishing, Inc., 1964

Fahnstock, James D., Computers and How They Work, Ziff-Davis Publishing Co., 1959

Irwin, Wayne C., Digital Computer Principles, D. Van Nostrand Company, Inc., 1960

Lytel, Allan, ABC's of Computers, Howard W. Sams Co., Inc., 1966

Marcus, Mitchell P., Switching Circuits for Engineers, Prentice-Hall, Inc., 1967

Technical Education and Management, Inc., Computer Basics, Howard W. Sams Co., Inc., 1962