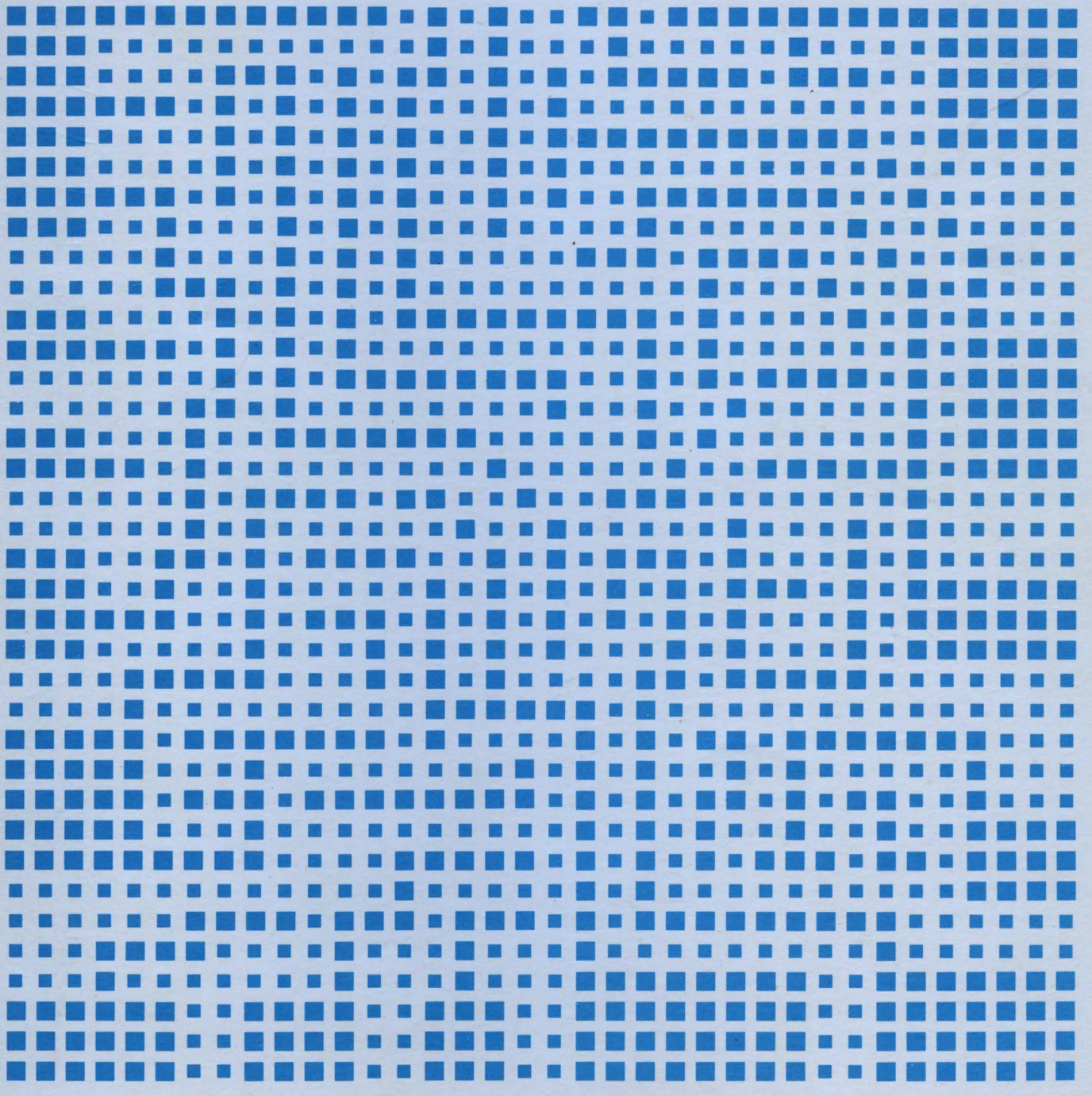
CONTROL DATA

CYBER 18 Hardware Characteristics



Supplementary Reference Manual

CYBER 18 Hardware Characteristics

CONTROL DATA

Pub. No. 76361239

Printed in the United States of America.
2 3 4 5 6 7 8 9/85 84 83 82 81

Table of Contents

Block 1

Block Diagram Analysis

Characteristics and Block Diagram (Text)

This reading examines the characteristics of the CYBER 18-20 processor and also presents a block diagram of it. Much of the information here will be familiar to you from previous readings concerning CYBER 18-20 software and programming. The review is provided to give you the framework needed for studying CYBER 18-20 hardware.

Processor Characteristics

The CYBER 18-20 is a general purpose, parallel mode, stored program, digital computer. What does this mean?

- "General purpose" means that the CYBER 18-20 was not designed for one specific application. Due to its microprogramming capability, the types of jobs that can be done are restricted only by the programs written and the overall hardware limitations. The CYBER 18-20 processor can work alone or can be connected through communication lines to other processors.
- "Parallel mode" refers to the type of data handling within the processor. Parallel instead of serial mode is used because it permits faster data transfers in storage and data manipulations.
- "Stored program" means that all the instructions of a program or subprogram must be stored in memory before the computer begins executing the program or subprogram.
- "Digital computer" means that the computer works on voltage levels interpreted as a 0 or a 1. The alternative to a digital computer is an analog computer, which operates with voltage and circuit changes that represent approximate-function parameters and results.

In the CYBER 18-20 processor, the size of the data word is sixteen bits. When this word is stored in the computer's memory, two more bits are created to provide a total working-size word of eighteen bits. These two additional bits are the parity bit (which is used to detect errors that may occur as data is stored, retrieved, or transferred) and the protect bit (which is used to specify words or areas in memory that are protected against undesirable changes).

Also included in this processor is a sixteen-level priority interrupt system. The interrupt system allows external equipment to initiate communication with the computer and also allows specified internal conditions to be sensed. At any one time, sixteen different pieces of equipment can be requesting use of the processor programming.

Before any computer can execute an instruction, the instruction must be read from memory and decoded. Decoding of a computer instruction enables the necessary circuitry to complete a sequence of events defined by the instructions. Various methods are used to decode computer instructions. One method used is to pass the instruction to be executed through gates and other associated logic. The gates decode the instruction and provide enabling signals that cause the instruction to be executed. This method is referred to as hardware decoding. The second method of instruction decoding uses two different memories. To keep the two memories separate, they are given names. "Macro" refers to the main memory; the instructions associated with it are referred to as instructions only or as macroinstructions. "Micro" refers to the second memory; its instructions are referred to as microinstructions. When the computer uses this second type of decoding, the macroinstruction is used to address memory locations in micromemory. These locations contain groups of microinstructions called microsubroutines. It is the microsubroutine that causes the macroinstruction to be executed. The program contained in micromemory is called firmware. The names given to this method of macroinstruction decoding are "firmware decoding," "microdecoding," and "emulation." The CYBER 18-20 is referred to as a microprocessor because it uses the firmware method of macroinstruction decoding.

The main memory in the CYBER 18-20 processor (often referred to as the macromemory, storage memory, or just memory) is either core- or MOS-type memory. If the memory is core type, its size can vary from 8K words (a word being eighteen bits) to 32K. When macromemory is MOS type, size varies from 16K words to 131K words. The sizes of macromemory just indicated can be doubled through the use of a second memory bank.

The micromemory (or the high-speed control memory) of the CYBER 18-20 is used for decoding program instructions. This memory is a MOS read only memory (ROM) or a read/write memory (RAM), depending on user requirements. With ROM, new data cannot be stored in the memory; only old data can be read. The size of micromemory can be either 512 words or 2048 words, with each word being thirty-two bits long.

CYBER 18–20 Block Diagram

Any central processor unit (CPU) and its operation can be described in terms of four main blocks.

- Memory
- Arithmetic (arithmetic logic unit, ALU)
- Control
- Input/output (I/O)

The memory section is used to store data and instructions that are used by the computer. The arithmetic section performs all math and logical functions. The control section generates the main timing and control signals used to operate the computer; it also decodes the instructions and delegates responsibilities to the other computer sections. Information being transferred into or out of the computer is controlled by the I/O section, which generates all signals required to complete I/O operation.

The simplified block diagram for a general computer is shown in figure 1-1.

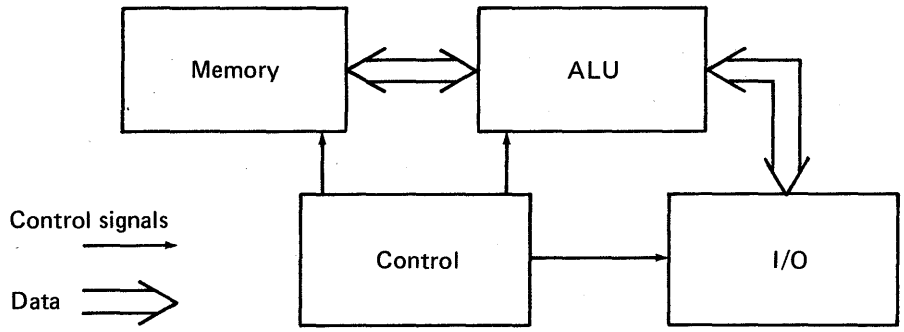


Figure 1-1. General CPU Block Diagram

Like any other computer, the CYBER 18-20 processor can be described by the general block diagram in figure 1-1. However, in studying a specific computer, more detailed block diagrams are useful. The basic block diagram for the CYBER 18-20 in figure 1-2 illustrates the operation of the CYBER 18 better than the general block diagram.

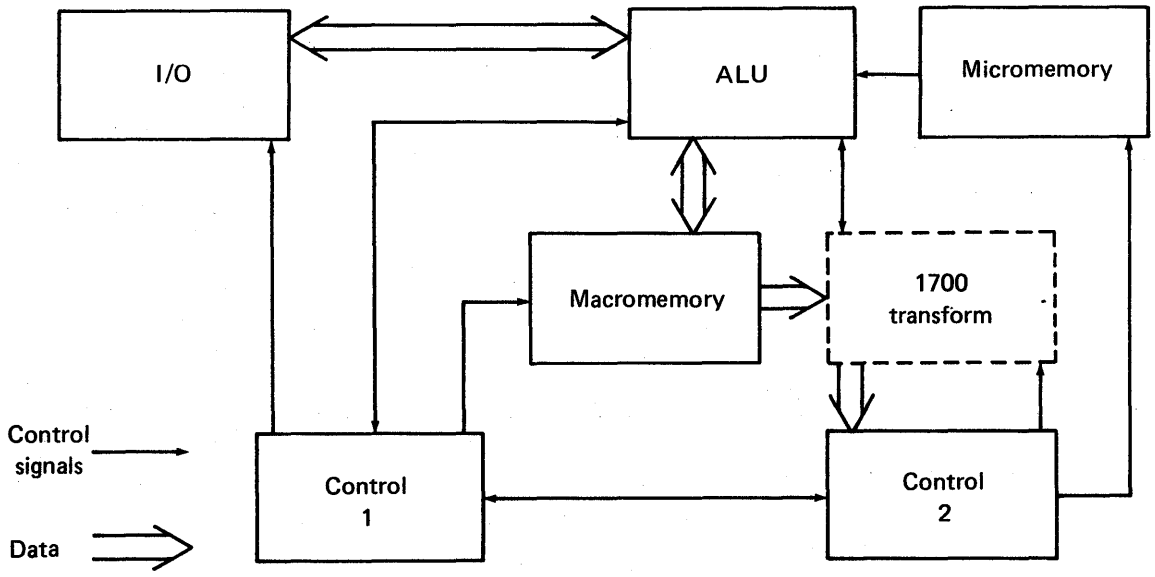


Figure 1-2. CYBER 18-20 Block Diagram

Control

The control section of the CYBER 18-20 has been broken into blocks called 1700 transform, control 2, micromemory, and control 1. The transform section is used in the initial decoding of instructions. Control 2 and micromemory have the job of further decoding the program instructions. Control 1 delegates responsibilities to all the other processor blocks and provides the master timing signals specifying what to do and when to do it.

The CYBER 18-20 uses the transform to route, under microprogram control, fields from the instruction through hard-wired paths to microaddresses and control registers. This microprogram-controlled vectoring (or transform) reduces the time spent selecting the proper microprogram entry; as a result, execution begins as quickly as possible. Instructions may be decoded without the use of the transform block but at a much slower rate.

In control 2, the instruction being decoded goes to the 1700 transform block. The data from the transform block is used in control 2 to select the micromemory output. In turn, the output from micromemory selects the signals used to control operations in the ALU, I/O, memory, and consecutive microlevel operators.

Memory

The main memory section is also called macromemory; it is used for program-controlled storage and retrieval of all instructions and data used by the processor. (Note that microinstructions are not implied unless "micro" is part of the word.) The memory section is divided into three functional units:

- Data interface unit (DIU)
- Address and control interface (A&CI)
- Memory array

Data enters memory through the data interface unit (DIU) where a parity bit is generated and appended to the data for error-checking purposes. When data is read from memory, the parity is checked in the DIU. The address and control interface (A&CI) generates the timing signals required to coordinate operations in the memory section and encodes the address for memory array selects. The memory array is where information is actually stored.

All of the data going to or coming from macromemory must go to or come from the ALU section. All the signals that control the macromemory come from control 1 and control 2.

Arithmetic Logic Unit

The ALU section is where all the logical and arithmetic operations are performed. Addition, subtraction, shifts, and exclusive OR are just a few of the thirty-two operations performed in the ALU. The ALU also handles data for input/output operations. All transfers of information into and out of the CYBER 18-20 go through the I/O section; however, this information must pass through the ALU also. The ALU holds the data being transferred as well as the address of the device where the data is coming from or going to.

The ALU consists of registers, selectors, the adder, and various other gates required for proper operations. The ALU registers used to hold data for ALU processing are the A, Q, P, and X registers. The selectors consist of multiplexer chips used to gate data from the selected registers to the adder. The adder is the device that performs the thirty-two logical and arithmetic functions. Sometimes the term "ALU" is used in the industry to refer to the adder, but in this unit, to avoid confusion, it refers only to the math section of the computer.

Data in the ALU must come from the I/O section or macromemory. The operation of the ALU is governed by control 1.

Input/Output

The I/O section is used to transfer data and instructions into and out of the CYBER 18-20 processor. Incoming data from the I/O section passes through the ALU and into memory. When data is to be output from the processor to an external device, a device address accompanies the data to the I/O section. From the I/O section, the data is sent to the device being addressed. This type of I/O is referred to as an A/Q scheme because it involves the A and Q registers in the ALU section.

There is another, faster type of data transfer that communicates directly with the memory section of the CYBER 18-20 processor; it is known as direct storage access (DSA) or direct memory access (DMA).

Data transfers go between the I/O section and the ALU. The I/O section gets control signals from control 1.

Summary

The CYBER 18-20 central processor unit is a

- General purpose
- Parallel mode
- Stored program
- Digital computer

It has

- A data word with a working size of eighteen bits
- A sixteen-level priority-interrupt system
- A macromemory capacity of 8K-32K (core type) and 16K-131K (MOS type)
- An ROM or RAM micromemory with a capacity of 512 or 2048 words

A basic block diagram of the CYBER 18-20 includes

- Control, which decodes program instructions and delegates responsibilities to other parts of the computer; consists of control 1, control 2, transform, and micromemory
- Memory, which provides storage of information; consists of data interface unit, address and control interface, and memory array
- Arithmetic logic unit, which performs arithmetic and logical functions; consists of registers, selectors, and adder
- Input/output, which transfers data in and out of the processor

Characteristics and Block Diagram (Exercise)

DIRECTIONS: Complete the following statements, answer the following questions, or follow the directions accompanying a particular question.

1. To increase the speed at which it can operate, the CYBER 18-20 processor uses a parallel mode of data transfer.

2. The memory section of the CYBER 18-20 can store 18 bits in a location.

3. The data word in the CYBER 18-20 is 16 bits long.

4. The method of instruction decoding used in the CYBER 18-20 is called Micro Instruction decoding.

5. When the macromemory is of the MOS type, a maximum of 131K words can be stored in one bank.

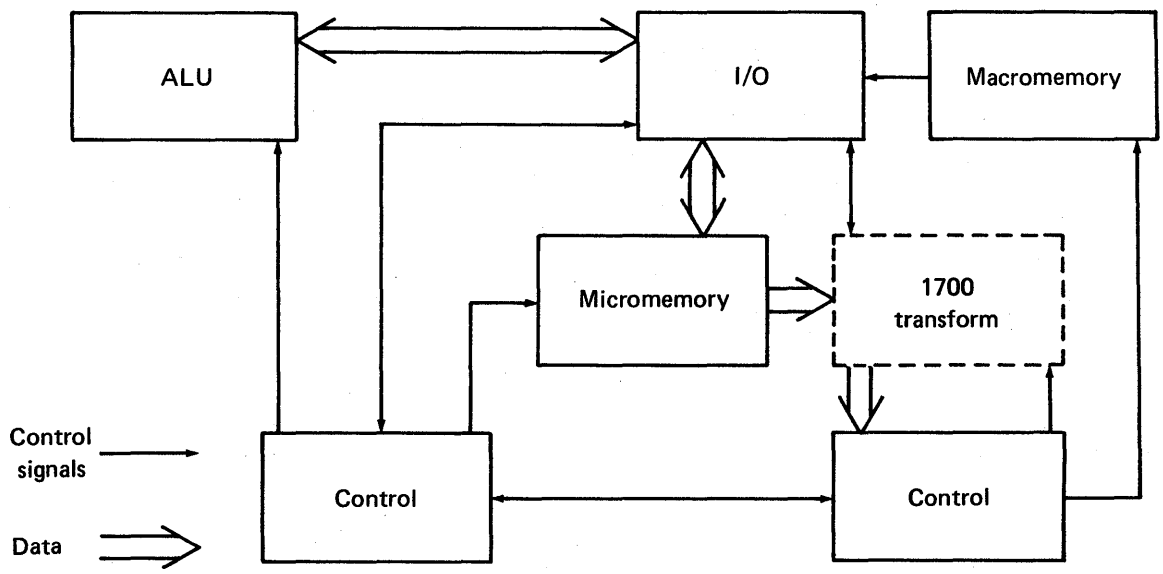
6. The block that is used to do calculations is called the ALU.

7. All functions performed in the CYBER 18-20 are sequenced by the section called the Control block.

8. Data that is to go outside the processor must pass through the I/O section.

9. The 1700 block is used to speed up the instruction-decoding process.
Transform

10. Correct the following incomplete block diagram of the CYBER 18-20 by correcting any incorrect names and adding missing blocks and wires.



11. The memory section of the CYBER 18-20 processor is divided into 3 units.

12. The unit where data is stored in memory is called the Memory Array unit.

13. When data is going to memory, parity is generated in the Data Interface unit.

14. What are two functions performed in the ALU section of the CYBER 18-20 processor? Arithmetic and Logic.

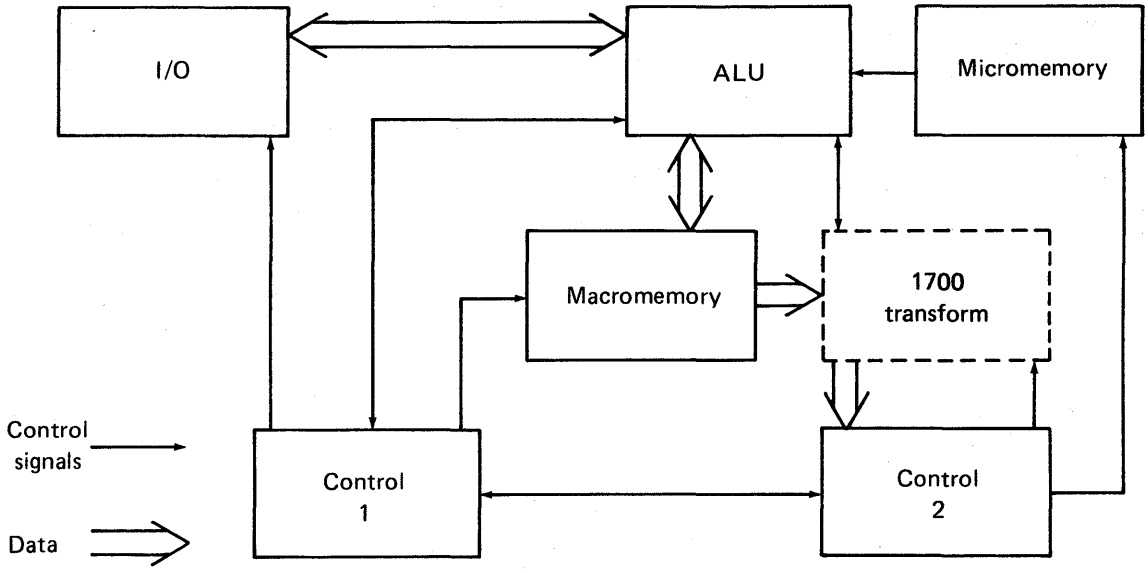
15. Two types of data transfers handled in the I/O section of the CYBER 18-20 processor are A/O and DMA.

16. List the three functions performed in the control section of the CYBER 18-20.
Instruction decoding, Master Timing, Initiating Sub op's

17. When the CYBER 18-20 is doing a direct memory access, data is transferred directly to or from Memory.

ANSWERS

1. Parallel 2. 18 3. 16 4. Firmware or microinstruction
5. 131K 6. ALU 7. Control 1 8. I/O 9. 1700 transform
10.



11. 3 12. Memory array 13. Data interface 14. Logic, arithmetic
15. A/Q, DSA, DMA 16. Instruction decoding, master timing, initiating suboperations 17. Memory

Card Layout (Text)

The circuits used in the CYBER 18-20 are found on printed circuit boards, or cards. These cards, which are eleven by fourteen inches, fit into specific, prewired slots within the CYBER 18-20 cabinet. Electrical connection is made between the pins on the cards and the associated pins in the rear of the cabinet's card slots. On the back of these card slots, also called the back plane, wires connect pins from one slot to the pins of other slots; these wires provide the electrical connections between the circuits on the printed circuit boards. To ensure that correct circuit connections are made, printed circuit boards must go into the positions indicated on the card layout (shown in figure 1-3).

At the top of the card layout are letters that serve as card position identifiers. For example, slot U contains the card named "panel interface." Some positions are left blank; if additional equipment is added to the system, cards containing the necessary additional logic or interfaces can be added to these blank positions.

The card positions are divided into three categories: memory, main processor, and input/output.

Memory

In this section, card positions Y and X are the memory array boards. If memory is to be enlarged, additional array boards can be placed in slots Z and AC. Slot W is used for the memory address and control interface boards. The memory data interface board is placed in slot V.

Main Processor

In this section, slot U is used for the control panel interface board. Slot R contains the 1700 transform board. The boards for control 1 and control 2 go into slots P and N respectively. Slot M contains the ALU board. Slot L is for the status mode interrupt board; this board is used in conjunction with the interrupt circuits and register that contain the operating status of the processor.

Input/Output

In this section, slot K contains the I/O-TTY board; this board includes the I/O block, the controllers for the teletype (TTY), and the conversational display terminal (CDT). Slot J holds the card reader and line printer interface board. Slot H holds the disk interface, also called the storage module drive (SMD); the disk uses the direct memory access channel. Slot F is a board used in interfacing the processor with telephone communications and is called the dual communications line adapter (DCCLA). Slot E contains the floppy disk adapter. The magnetic tape controller card goes in position AB.

The following card positions are left blank: AA, A, B, C, D, G, S, and T.

	Card Position Identifier	
Input/output	AB	Magnetic tape controller
	AA	
	A	
	B	
	C	
	D	
	E	Floppy disk (adapter)
	F	DCCLA
	G	
	H	Storage module drive I/F
	J	Card reader/line printer I/F
	K	I/O-TTY
Main processor	L	Status mode interrupt (SMI)
	M	ALU with file 1
	N	Control 2
	P	Control 1
	R	1700 transform with ROM
	S	
	T	
	U	Panel Interface and breakpoint controller
Memory	V	Memory interface (data)
	W	Memory interface (address)
	X_i	MOS memory array
	Y_i	MOS memory array
	Z_i	(MOS memory expansion)
	AC_i	(MOS memory expansion)

Figure 1-3. CYBER 18-20 Card Layout

Card Layout (Exercise)

DIRECTIONS: Answer the following questions or complete the following statements.

1. What card is in card position R in the CYBER 18-20? 1700 Transform.
2. The magnetic tape controller is in which card slot? AB.
3. The cards can/cannot (circle one) be interchanged from one card slot to another.
4. Card slots V through AC can be categorized as used only for Memory boards.
5. What card is in position E? Floppy Disk
6. What card is in card position N? Control 2.
7. What card is in position T? Blank.

ANSWERS

1. 1700 transform 2. AB 3. Cannot 4. Memory
5. Floppy disk 6. Control 2 7. Card slot T is left blank

Block 2

Magnetic Core Memory

Magnetic Core Memory

At some point in your study of computers, you probably have noticed the term "magnetic core storage." It refers to a type of computer memory that uses the properties of magnetic fields to store and retrieve data. Such storage is common in the computer industry. To understand how it works, you need to know a bit about magnetic fields. This reading will give you a background in the principles of magnetic fields as they relate to magnetic core storage.

Properties of Magnets

As you know, magnets have a north and a south pole. When two magnets are near each other, their like poles repel and their unlike poles attract. A somewhat similar phenomenon occurs when certain types of metals (ferrous, or iron-based) are brought near magnetic fields. From being near a magnet, these metals acquire their own magnetic fields. The part of a piece of metal nearest a magnet takes on the value of the pole opposite that of the near magnet pole. (E.g., if the end of a piece of metal is near the south pole of a magnet, that end becomes the north pole.) The other end of the metal takes on the value of the remaining magnetic pole. Even when the magnet is removed, the metal retains a certain amount of magnetism. This is called "residual magnetism."

Residual magnetism is the key to magnetic core memory. The degree to which residual magnetism appears depends basically on two factors: (1) the strength of the original magnetic field and (2) the type of metal.

Figure 2-1 illustrates how a piece of metal acquires residual magnetism. The process is called "magnetic induction." As the steel bar is moved near the magnet, it begins to pick up the magnetic field (as in part B). Lines of force go directly between the nearest ends of the magnet and the bar and between the two ends which are furthest apart. When the bar and the magnet are close together (as in part C), almost all of the magnet's magnetic field reaches the bar, and the bar acquires its own north and south poles. As the bar moves away from the magnet (as in Part D), the magnetic field from the magnet has increasingly less influence, until it does not reach the bar at all (part E). Nevertheless, the steel bar, as shown, has retained a small amount of magnetism. By being placed in a magnetic field, it has become a magnet itself, with its own magnetic field.

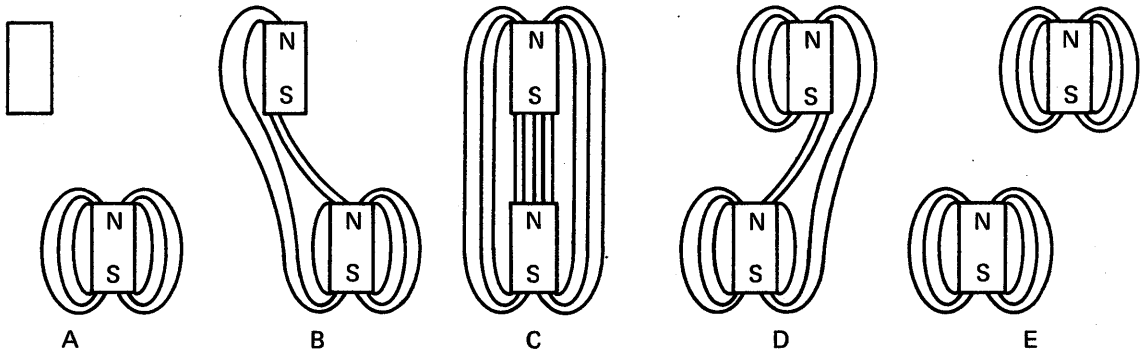


Figure 2-1. Magnetic Induction

Properties of Electromagnets

An electromagnet affects a steel bar just as an ordinary magnet does. A similar effect can be achieved by holding the steel bar in one place and increasing the current through the electromagnet. Increasing the current increases the strength of the magnetic field. As the field builds up (see figure 2-2), it attracts the steel bar and induces a magnetic field in it. When the current is reduced, the magnetic field no longer influences the bar. The steel bar, however, retains a part of its magnetic field. It is now a magnet.

Reversing the flow of electricity through an electromagnet reverses its polarity. Figure 2-3 shows the effects of this reversal on the stationary steel bar. As the reversed current increases in the opposite direction, the electromagnet has the effect of magnetizing the steel bar in the opposite direction. The steel bar magnet first loses its original polarity and then acquires the same polarity as the electromagnet. Now, if the power is decreased in the electromagnet, the steel bar still retains this new polarity.

This principle of reversing the flow of electricity is used in the computer to magnetize small metallic rings in one direction or the other.

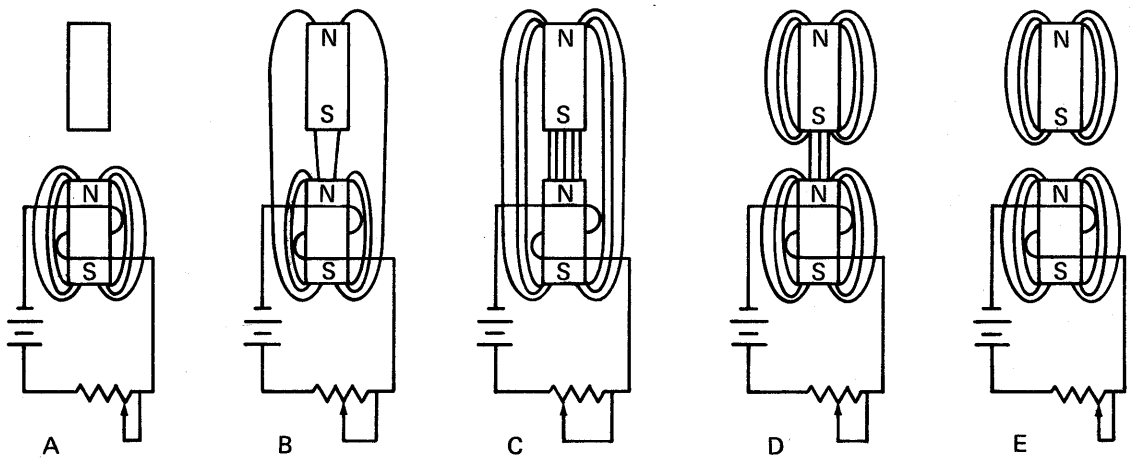


Figure 2-2. Electromagnetic Induction

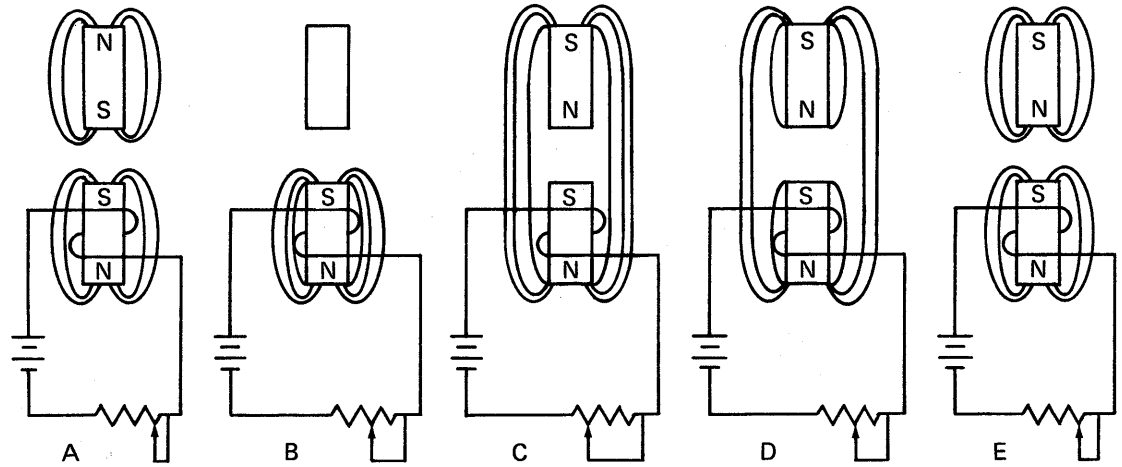


Figure 2-3. Reversing Polarity

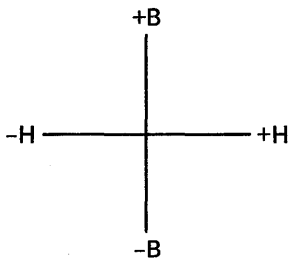
Hysteresis Loops

Hysteresis loops are graphic ways of showing the amount of current and its direction and the resulting amount of magnetism in the metal and the direction of the magnetic force. Figure 2-4, based on the examples used in figures 2-2 and 2-3, shows how each change in current can be plotted to draw a hysteresis loop.

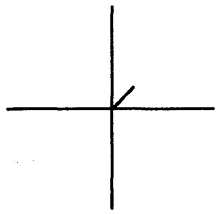
For our purposes, these loops are important only as a way of showing that some metals retain more of a magnetic charge than others. Figure 2-5 shows the loops for both steel and ferrite material. Ferrite material is an alloy of iron oxide and manganese, with nonmetallic oxides and a ceramic binder used to create and hold forms. The much fatter loop for ferrite shows that it will retain a high level of magnetic charge.

Examining the hysteresis loop for ferrite also shows that at a certain point in the current, the magnetic field reverses itself very rapidly. As the magnetizing force or current (–H in the figure) is reversed substantially, the magnetic field is quickly reversed between points Y and Z of figure 2-5.

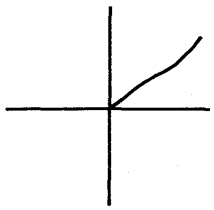
The ability of ferrite to retain a magnetic field and to reverse that field rapidly when the current is reversed substantially makes ferrite an ideal substance for computer memories. When ferrite is magnetized in one direction, it represents a binary 1; in the opposite direction, a binary 0. Using a binary code, a series of ferrite pieces stores information.



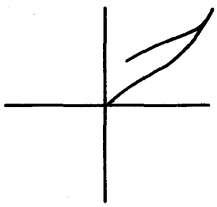
See figure 2-2(A)
A



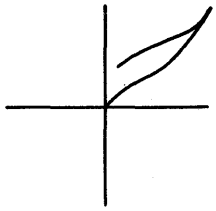
See figure 2-2(B)
B



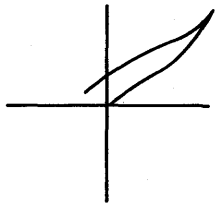
See figure 2-2(C)
C



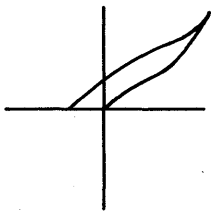
See figure 2-2(D)
D



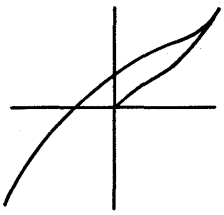
See figure 2-2(E)
E



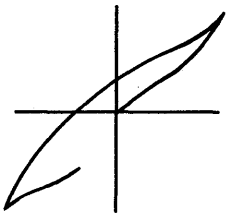
See figure 2-3(A)
F



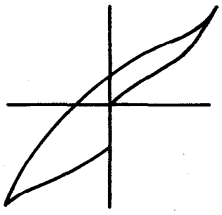
See figure 2-3(B)
G



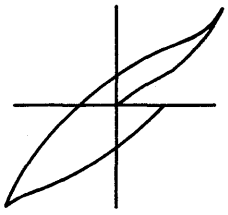
See figure 2-3(C)
H



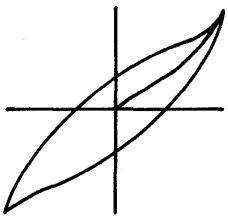
See figure 2-3(D)
I



See figure 2-3(E)
J



K



L

KEY
H = Amount of current
B = Amount of magnetism in steel rod
+,- = Direction of current flow or magnetic force

Figure 2-4. Plotting the Hysteresis Loop

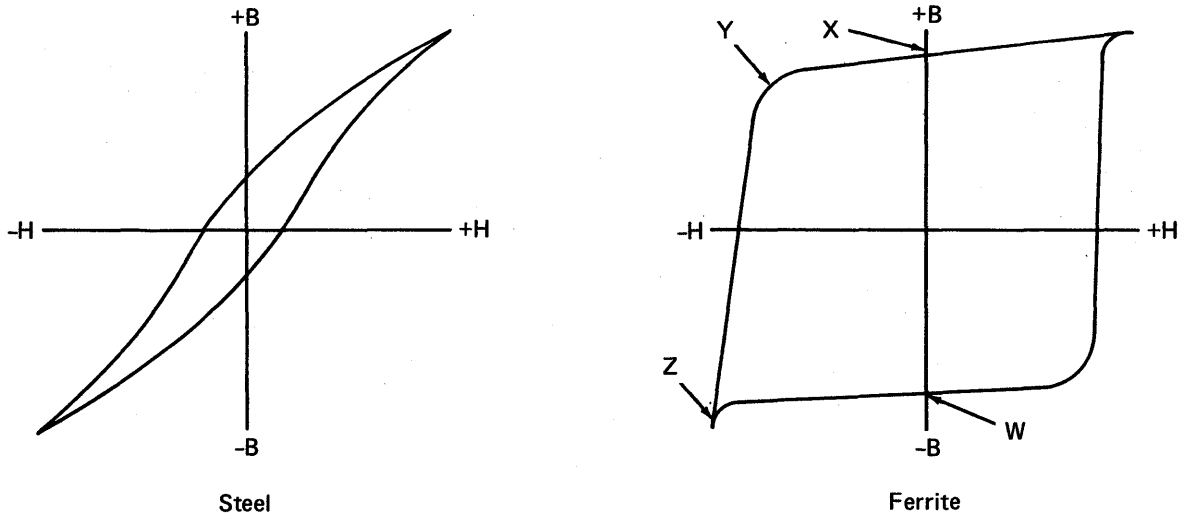


Figure 2-5. Hysteresis Loops

Summary

Magnetic core storage uses the principles of magnetic fields to store information. When a magnet is brought near certain metals, it induces a magnetic field in those metals. When current is changed in an electromagnet, it can affect the magnetic field in nearby metals. Various metals are affected differently by these changes of current. Ferrite easily acquires a magnetic field and will quickly reverse that field when a substantial change in current direction is made. This makes ferrite ideal for computer memory use.

4-Wire Core Memory (Text)

To function, a computer needs access to information, and that information must be stored when the computer is not using it. There must be a means of accessing the information when it is needed, and there must be a way to change information already stored. In core memories, all this is possible through the use of small rings called toroids. This text explains how a computer uses toroids to:

- Store data in a ferrite core
- Read data from a ferrite core
- Restore data after it is read
- Store new data

It also explains how a grid plane is used to make possible the multiple selection of bits of information.

Storing Data in a Ferrite Core

Ferrite is a magnetic substance, usually containing iron oxide. In core memories, ferrite material is used to create a thin doughnut-shaped ring. This is called a "toroidal core," or simply a "toroid." Through the center of the toroid is threaded a wire called a "drive line." By passing a current in one direction or the other through this line, the magnetic state of the toroid can be changed. The direction of the current in this magnetic drive line follows the "left-hand rule" from basic electronics. For example, by placing the switch in position "0," as in figure 2-6, current flows from left to right to polarize magnetically the ferrite material as shown. (In this and subsequent diagrams, EMF means electromagnetic force.) When the switch is in the off position, the ferrite material retains this magnetic orientation. (See figure 2-7.) When the switch is in the "1" position, current travels through the conductor in the opposite direction, and magnetic polarization within the ferrite material switches to the opposite direction, as shown in figure 2-8.

In each of the figures that follow, the direction of the polarization (or magnetic force field) depends upon the physical relationship of the conducting wire to the toroid. Remember that it is the direction of the lines of magnetic force that induces the polar states. For example, if a conductor were threaded over and then under the toroid surface from left to right (as in figure 2-6), polarization would be in the opposite direction than if the conductor were threaded under and then over. This is because the directions of the lines of force would be different.

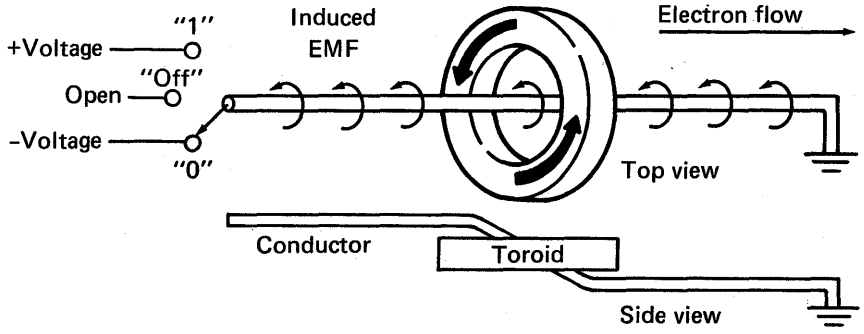


Figure 2-6. Write a Logic 0

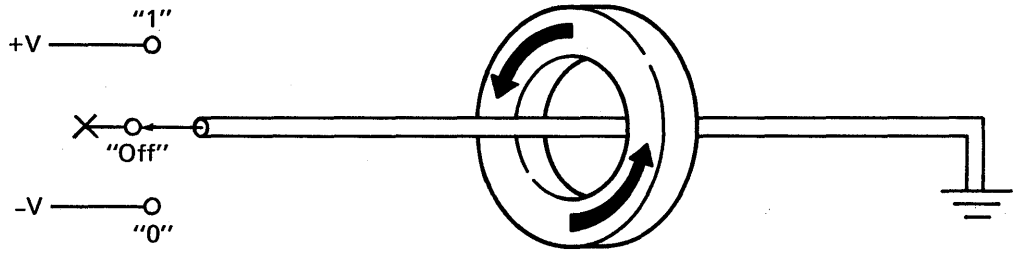


Figure 2-7. Store a Logic 0

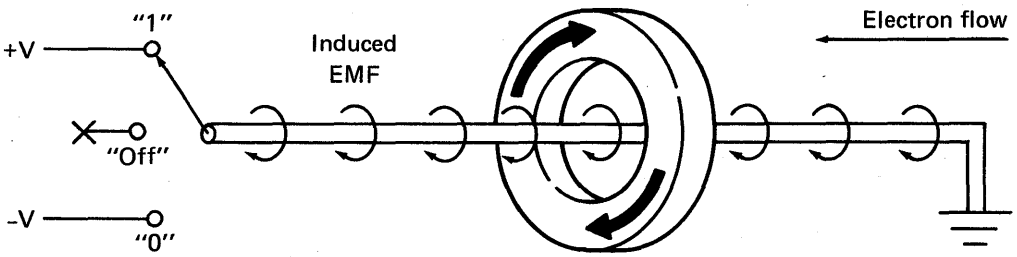


Figure 2-8. Write a Logic 1

Polarization occurs only if the correct amount of current is run through the wire conductor. If too little current is sent through, no stable polarization occurs; the toroid falls back into its original state. What determines the right amount of current is the physical size of the toroid. The smaller the toroid, the less current needed and the faster the toroid can be forced to change polar states. Today's toroids are between 14 and 20 mils (thousandths of an inch) across; a period on this page would cover a half-dozen of them. The amount of current needed to change their polar state is correspondingly small.

In figures 2-6 through 2-8, a counterclockwise magnetic field signifies a logic 0 and a clockwise magnetic field, a logic 1. Due to the magnetic properties of a toroid, there is no state between clockwise and counterclockwise. This qualifies a toroidal core as a bistable device, well suited for binary data storage. Toroids form the computer's "memory," or data storage facility.

This type of memory is referred to as magnetic core memory, or magnetic core storage. In some cases, the word "core" refers to all of the core storage; other times, it refers to an individual core.

When a toroid is polarized one way or another to represent a binary digit, either 1 or 0, this placement of information into memory is called writing or storing. A series of polarized toroids represents a stored word of information.

Reading Data

To read information from a core, a second wire called a "sense winding" is passed through the core, as shown in figure 2-9. This wire is used only when reading information. When a significant voltage is induced in the sense winding, the voltage amplifier (AMP) amplifies the signal, outputs a logic 1, and sets the flip-flop. When the sense line receives less than the required amount of voltage, the amplifier will output a logic 0. This required amount of voltage is called the "threshold voltage."

If a core has previously been set at a logic 0, as shown in figure 2-10, and the switch is moved to position "0," a small amount of current is induced in the sense line. This current is not enough to cause the amplifier to output a logic 1; instead, it outputs a logic 0, and the flip-flop remains at 0, as shown in figure 2-10. In this case, the induced EMF of the drive line is in the same direction as the polarization lines of force, and the only change "sensed" is the rise and decay of drive line EMF.

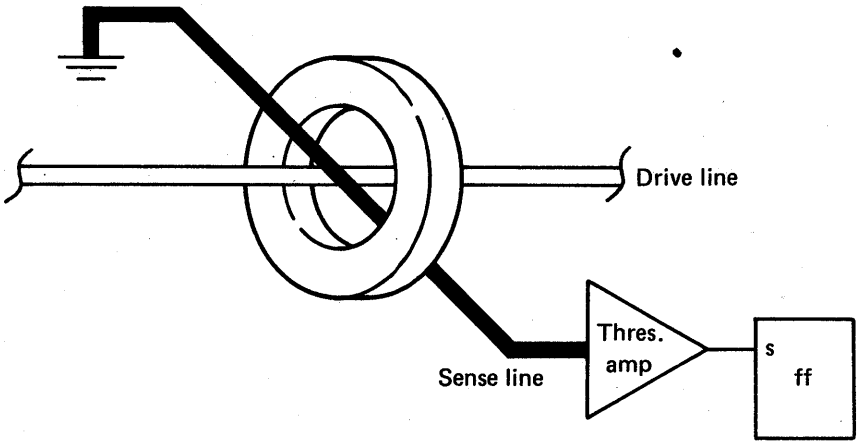


Figure 2-9. The Sense Line

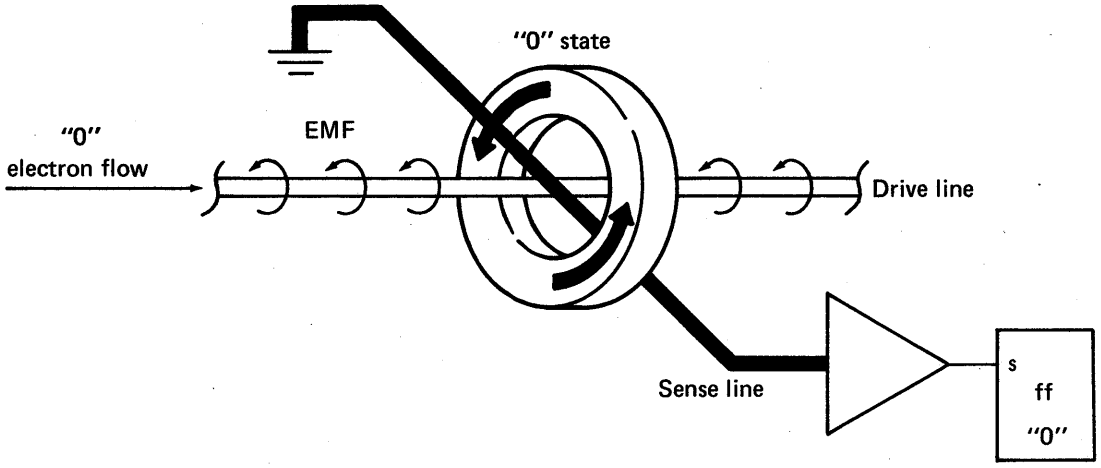


Figure 2-10. Read a Logic 0

Figure 2-11 shows the sensed voltage seen on the sense line as "induced EMF." This voltage is well under the built-in threshold of the amplifier. This is to keep the amplifier output at a logic 0. Notice that the sensed voltage reflects the rise of drive line EMF (T0 on the graph) and the decay of drive line EMF (T1). Either the positive voltage, negative voltage, or both can be used by the amplifier. For our examples, assume that we use just the positive voltage.

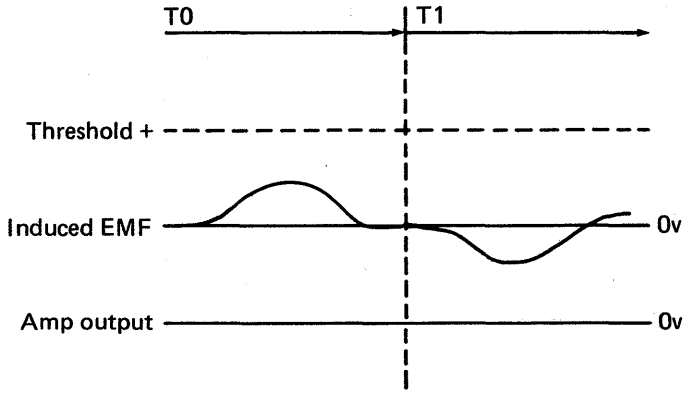


Figure 2-11. Logic Level 0

When the core is polarized to indicate a logic 1 and the switch is moved to the "0" position, as in figure 2-12, the field around the drive line is in opposition to the field of the core.

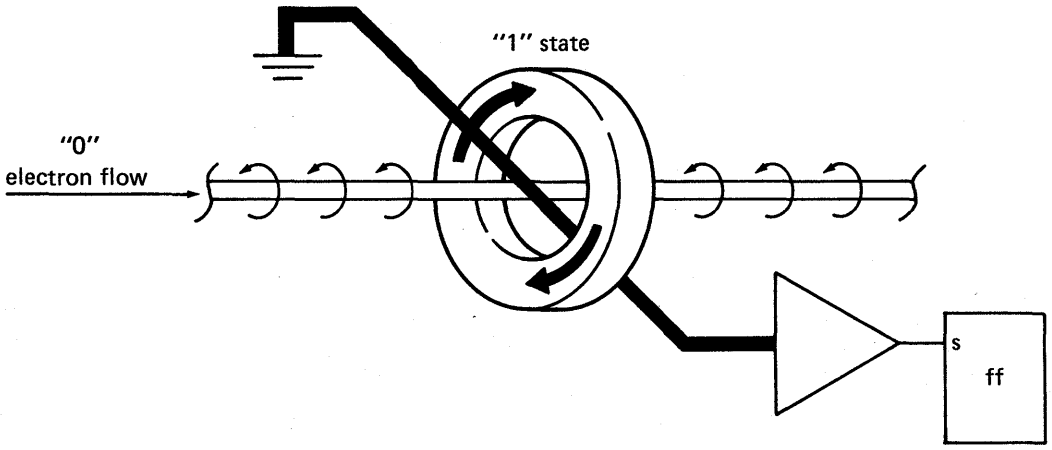


Figure 2-12. Opposition of Drive Line and Core Fields

This opposition forces the core's field to switch from a 1 to a 0, as shown in figure 2-13. In doing so, the field collapses and builds up again in the opposite direction. This collapsing field cuts through the sense winding and induces a large voltage into it. This voltage causes the amplifier to output a 1, setting the flip-flop.

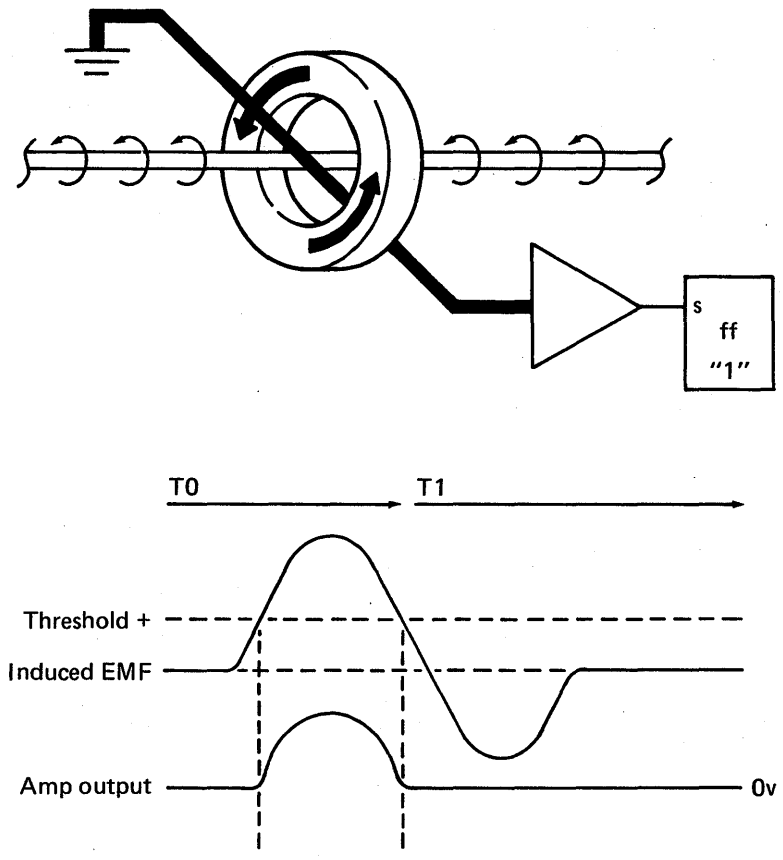


Figure 2-13. Read a Logic 1

One might think that the EMF changes of the drive line and sense line would be comparable in strength and, therefore, in induced voltage. This is not the case. Ferrite material is retentive; polarized molecules, like bar magnets, work together to resist changing their state. When the EMF of the drive line finally reaches a high enough point, however, the polarity of the toroid is overcome and rapidly changes. Just as the ferrite material works together to resist change, it also suddenly works with the drive EMF to achieve the opposite state. This much greater "switch" EMF permits simpler threshold and amplifier circuits.

The contents of a core can be read by passing a current through the drive line in the same direction as in writing a logic 0 and then testing the flip-flop to see if the magnetic field of the core has changed or collapsed. If the magnetic field has collapsed, then the flip-flop would have been set, meaning that the core was storing a logic 1. If the magnetic field has not collapsed, then the flip-flop would have been clear, meaning that the core was storing a logic 0.

In any case, reading the contents of a core leaves the core storing a logic 0, as shown in figure 2-13. This means that the process of reading from the core reduces its contents to zero, regardless of what was entered earlier. For this reason, core memories are sometimes referred to as destructive readout (DRO) memories.

It is desirable to read the contents of a core and have the core retain its original information. To do this, another step is needed: rewriting, or restoring the core to its original state.

If the flip-flop is set after a read operation, the restore operation requires re-establishing a logic 1 in the core. If the flip-flop is clear after the read operation, then the core was originally clear, so a logic 0 should be placed back in the core. Whichever the case, it is the flip-flop's image of what was stored that controls restoring that core.

Restoring Data After a Read

To accomplish restoring information in the core, a third wire is normally used. This third wire, an "inhibit winding," is used only when restoring information in the core. (See figure 2-14.)

In figure 2-14, several components have been added. During the read cycle, A0 outputs current in a direction that causes the field around the core to indicate a logic 0. During the restore cycle, A1 accepts current in a direction that causes the magnetic field around the core to indicate a logic 1. Now, assume that a read has been executed and that the flip-flop holds a 1 or 0. During the restore cycle, if the flip-flop were clear or holding a 0, the AND gate B4 would feed into A2, causing A2 to output a current in the inhibit line in the opposite direction of the current from A1, and in effect canceling the EMF of the restore current. Thus, at the end of the read/store operation, the core and the flip-flop would reflect a logic 0.

During the read cycle, if a 1 is read out of the core, the flip-flop is set. As it is set, the clear side of the flip-flop breaks the AND gate B4 into A2. A2 is not able to output a current flow and, because its EMF is in a logic 1 direction and unopposed by inhibit current, the core is restored to its original state, which is 1.

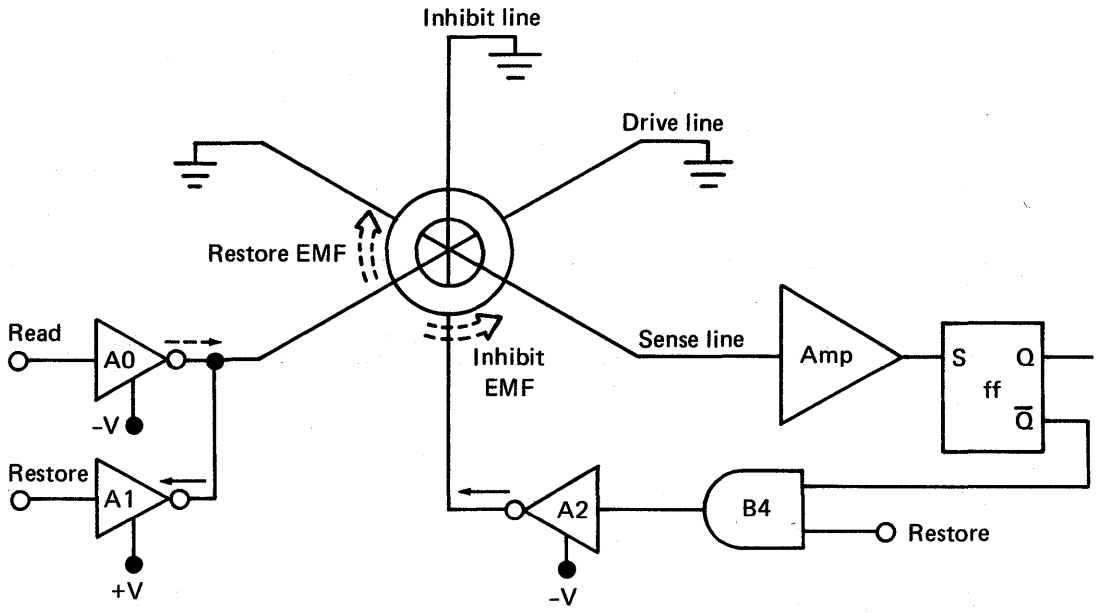


Figure 2-14. Restoring Information

When a 0 is read out of the core, however, the flip-flop remains clear, and during the restore cycle, the clear side of the flip-flop (a logic 1) enables the AND gate B4 into A2. A2 then outputs a current in the logic 0 direction during the restore cycle. A1 outputs in a 1 direction. Because A1 and A2 output currents producing EMFs in the opposite directions, the fields that they create at the core are in opposition; there is, therefore, no effect on the core. It remains in the 0 state—as it was before information was read from it.

Note that, in order to inhibit switching a core, an inhibit current need not be equal in strength to the restore current. As a general design rule, you will find that inhibit currents generally are just a bit more than half the restore drive. Recall that the core material itself resists change to a considerable degree.

Storing New Data

To write new information into a core, the old information is read out, leaving the core a logic 0. The information read out is not, however, allowed to be placed into the flip-flop. Instead of the old information's being gated into the flip-flop, the new information will be gated in. During the restore cycle, the new information will be placed into the core.

Before considering the entry of new data, review the read/restore by examining figure 2-15. As seen in this figure, when a bit is read from the core, the term WRITE is a 0 and the term $\overline{\text{WRITE}}$ is a 1. The circuit in figure 2-15 is, therefore, the same for reading as the circuit in figure 2-14.

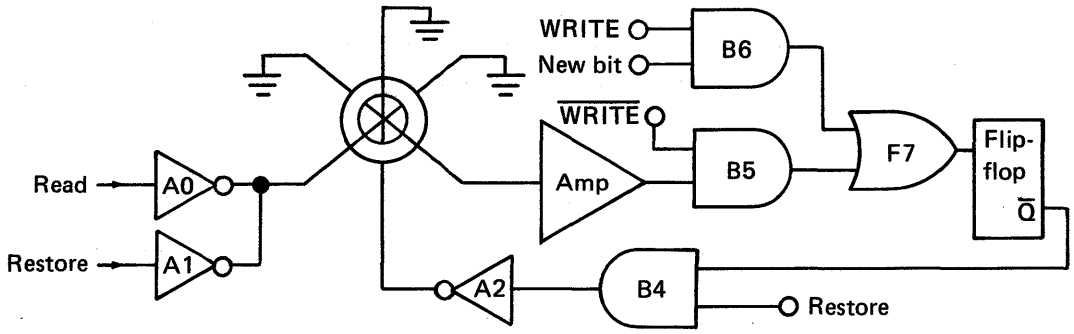


Figure 2-15. Entering New Data During Restore

With $\overline{\text{WRITE}}$ being a logic 1, the output of the amplifier is gated into the flip-flop. The flip-flop contains the information that was in the core. The restore cycle would now be initiated in the manner previously described.

To write a new bit into the core, the term $\overline{\text{WRITE}}$ is a 0 and the term WRITE is a 1. The B5 AND gate into the flip-flop is constantly broken. No matter what is read from the core, the flip-flop is not affected by the read cycle. When the bit to be written into the core from an outside source (such as a processor) is a logic 1, the AND gate B6 is enabled because WRITE and new bit are both logic 1's, setting the flip-flop. During the restore cycle, the logic 1 is entered into the core. When the new bit is a logic 0, the AND gates B5 and B6 are broken and the flip-flop remains clear. When a new bit is written into the core, the flip-flop holds the new bit and this new bit is written into memory during the restore cycle.

Review of Toroids

Before going further in this text, be certain that you know the answers to these questions:

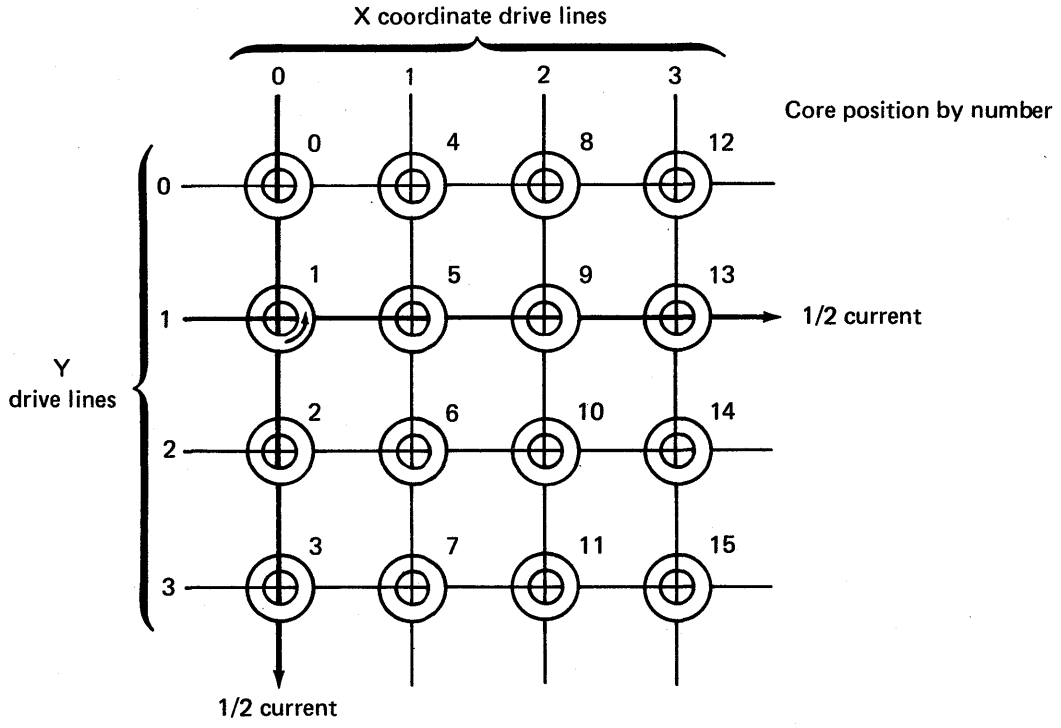
- What is a toroid or core?
- How is data read from a toroid, and how is data restored after it is read?
- How is new data stored?

A toroid is ferrite material in a doughnut shape used to store a bit of information. Whether the stored bit is a 1 or a 0 depends on the polarity of the toroid. A drive line and a sense line are threaded through the toroid. When the toroid is to be read, a current that would cause a polarity equivalent to 0 is sent through the drive line; if the toroid is currently a 1, the polarity changes and induces a current in the sense line, and if it is currently a 0, nothing happens. In either case, the flip-flop connected to the sense amp can determine what value the toroid held. This form of reading causes all toroids to be 0 after reading; thus the original information must be restored. This is done through a third wire, an inhibit wire, which—depending on whether a 0 or 1 was read—carries a countercharge to the restore charge going through the drive line. The drive line and the inhibit wire determine the polarity of the toroid. To enter new data, the flip-flop is kept from being affected by the read cycle, and the new bit of information is used to affect the inhibit wire and thus set the toroid to the 1 state.

Multiple Bit Selection Using a Grid Plane

A computer may contain several million cores. Each core will store one bit of data. Because a computer word is made up of a predetermined number of bits, several cores are needed to store it. The actual number of bits in a computer word is determined by the design requirements of the computer. In the examples that follow, the computer word is eighteen bits long. To read from or write into memory requires transferring all eighteen bits at a time. Using the method of core drive just described, decoding eighteen cores for reading or writing would be a complex job. To decrease the complexity of decoding circuits, other methods of core selection are used.

The most commonly used method is the random access grid plane, or array method. In this method of accessing bits in memory, the cores are arranged in rows and columns called a grid matrix. For simplicity, a four-by-four grid is shown in figure 2-16. A usable grid would have a far greater number of intersects to accommodate a greater number of bits. Keep in mind that, in the following description of the grid, only one bit position of a word is being found.



Note: Sense and inhibit lines are not shown.

Figure 2-16. Grid Plan Method

Reading

In the grid shown in figure 2-16, each core has four wires through it: one sense and one inhibit winding (not shown) and two drive lines (X and Y). Each of the drive lines is capable of providing half the current required to cause a core to switch states. To read a bit from memory, appropriate X and Y drive lines are selected. The two drive lines together carry the current required to switch the core. In figure 2-16, the core receiving drive from both X and Y drivers (the number 1 core) is selected to be read. The X0 and Y1 drive lines are selected. X0 passes through not only core 1 but cores 0, 2, and 3. Because there is insufficient (only one-half) current through X0, there is not enough field around the wire to cause any other core to switch. Drive line Y1 passes through cores 1, 5, 9, and 13. The Y drive line also provides one-half current and, therefore, one-half field to each core. Core 1 has one-half field provided by drive line X0 and one-half by Y1. Since these two fields aid one another, there is a sufficient field build up around the X and Y drive lines to cause core 1 to switch.

Core 1 is said to be "selected" because it has the amount of electron flow required to switch it. The other cores have only half the amount of current needed to switch them, so they are referred to as being "half-selected." The remaining cores, such as core 4, have no electron flow through them and are referred to as being "not selected." Only the core that is fully selected will switch.

The desired cores are selected by decoding a specific address. The decoded address enables current to flow through the desired X and Y drive lines and thus selects a specific core.

The address required to select a specific core in figure 2-16 is a four-bit address. Given the address-decoding format in table 2-1, it can be seen that, for a given address, certain X and Y drive lines are selected, and they, in turn, select a specific core.

TABLE 2-1
Address-Decoding Format

Address (Binary)	Selected Drive (Octal)		Core Number (Decimal)
	X	Y	
0000	0	0	0
0001	0	1	1
0010	0	2	2
0011	0	3	3
0100	1	0	4
0101	1	1	5
0110	1	2	6
0111	1	3	7
1000	2	0	8
1001	2	1	9
1010	2	2	10
1011	2	3	11
1100	3	0	12
1101	3	1	13
1110	3	2	14
1111	3	3	15

For example, let's say that the address supplied is 0010. Using the address and the address-decoding format, we find that the drive lines selected are X0 and Y2 and that these two drive lines select core 2. Figure 2-16 corroborates the selection of core 2 when current is placed in drive lines X0 and Y2. This method of core selection is the same, regardless of the cycle being processed: read, restore, or write. The only difference

in the drive line selection depends on the direction of current in the drive lines. In a read operation, current flows in the direction required to write a logic 0, and in a restore operation, current flows in the direction required to write a logic 1.

Since only one core will switch at one time, each core in this X/Y plane uses the same sense winding. The flip-flop in figure 2-17 would be set only if the selected core were a logic 1 when read from memory. The other cores are not capable of switching at this time because they have not been fully selected. If the selected core were a 0, it would not switch during the read cycle because the read current tries to switch the core to logic 0. In this case, the flip-flop would not be set. In either case, however, the flip-flop in figure 2-17 reflects the state of only the selected core.

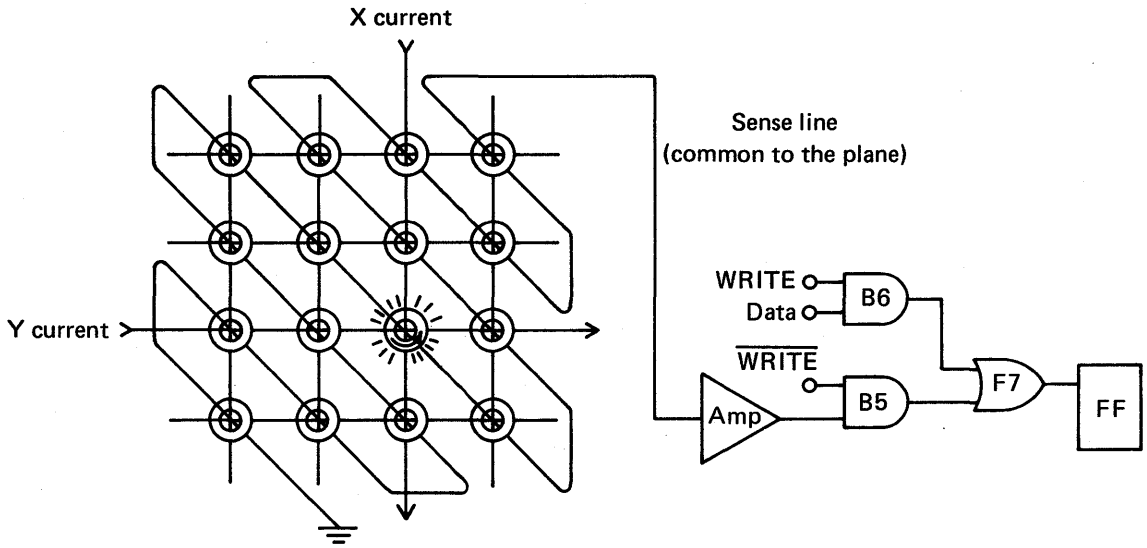
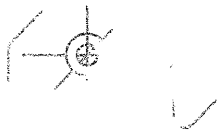


Figure 2-17. Selecting a Core: Reading

Restoring

Once the information has been read from a core, it must be restored. To select the same core for the restore cycle as selected for the read cycle, the same drive lines are chosen. This time the current will be reversed, but it will still be at one-half current for each of the two drives. Together, the two drives have an electron flow sufficient to switch the core. Again the state of the flip-flop (which reflects the state of the bit while it was in the core) determines whether or not a 1 is to be written into the core during the restore cycle.

If the flip-flop is clear, the inhibit line, as shown in figure 2-18, needs to provide only one-half current in a direction to oppose the writing of a 1 into the core. The one-half current in the inhibit line is not strong enough to affect other cores, but in the fully selected core it has the effect of counterbalancing one of the drive lines. This produces the effect of an electron flow only half as strong as needed to switch the core. The result would be the same as if the core were only one-half selected. On the other hand, if the original bit contained in the core is a logic 1, the inhibit line would be turned off and no field would be built up to stop the writing of a 1 in the core during the restore cycle.



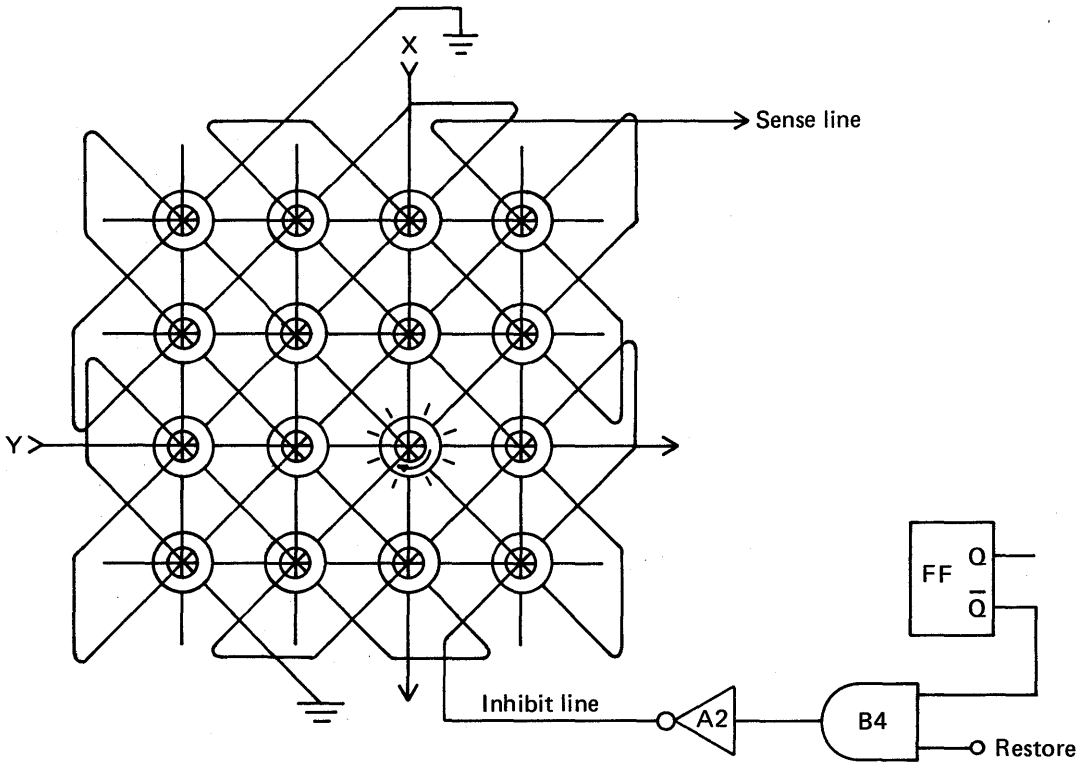


Figure 2-18. Selecting a Core: Restoring

Review of Grid Planes

In a grid plane, there are two drive lines, one set vertically and one set horizontally. To access a particular core, one-half current is sent through the vertical wires, and one-half through the horizontal wires passing through the core. When the two currents meet in the core, they create an electron flow strong enough to cause the toroid to switch. The same sense winding is threaded through all cores in the plane; similarly, the same inhibit line goes through all. During a restore operation, the inhibit line (based on whether a 1 or a 0 was read) sends out a half-current that has the effect of knocking out one of the half-currents in the drive line and, thus, preventing the core from switching.

4-Wire Core Memory (Exercise)

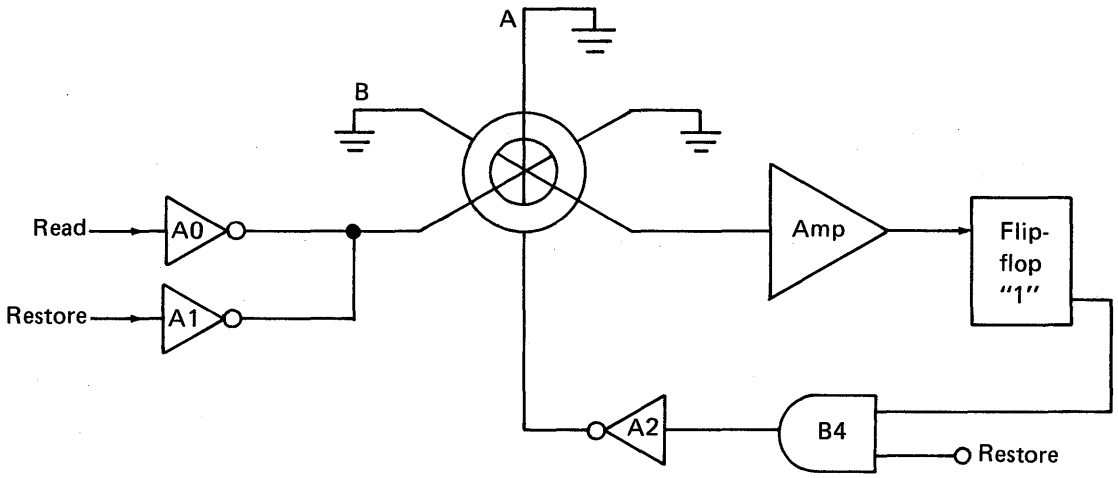
DIRECTIONS: Complete the following statements.

1. A wire used in the read and restore process is called the Drive line.

2. The wire required in the read process is the Sense line.

3. The wire required in the restore process is the Inhibit line.

DIRECTIONS: For questions 4, 5, 6, 7, and 8, refer to the following figure.



4. The wire marked A is the Inhibit line, and the wire marked B is the sense line.

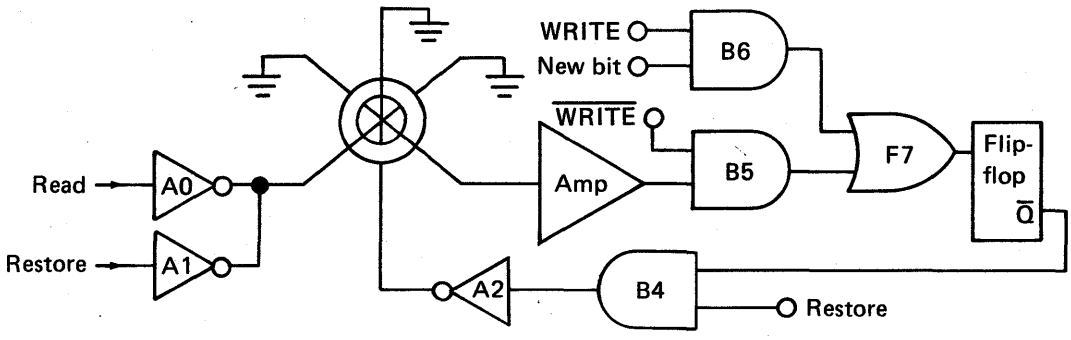
5. After a read cycle, the selected core will be storing a logic 0 .

6. The preceding figure shows the conditions in a core circuit after a read cycle. When the restore cycle is complete, a logic 1 will be placed in the core.

7. During a restore cycle after a logic 0 is read from the core, which drivers (A0 or A1 or A2), if any, will output a current? A1 & A2.

8. During a read cycle, the current in the drive line will flow in the same direction as the current in the inhibit line when a logic 0 is being written into the core.

DIRECTIONS: For questions 9, 10, and 11, refer to the following figure.



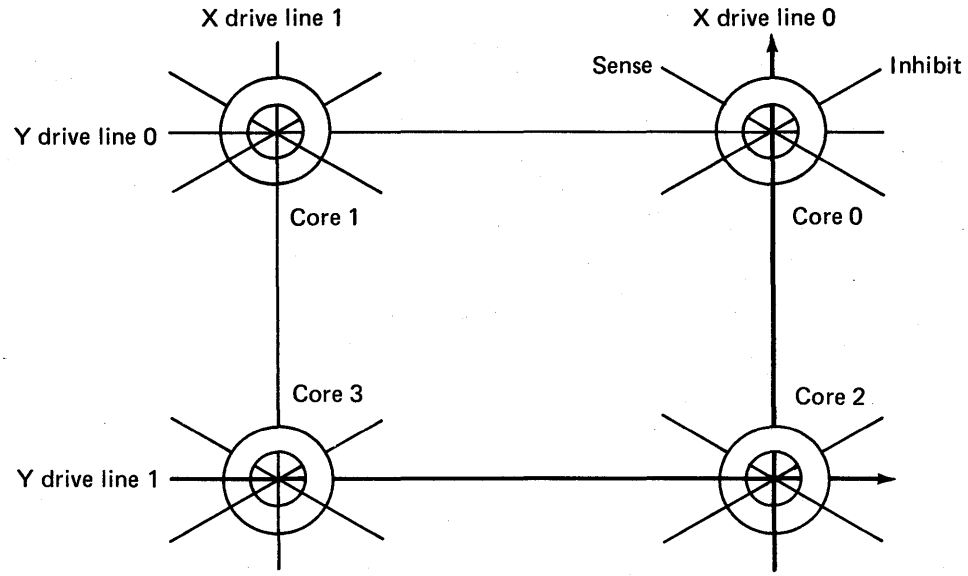
9. Gate B5 will output a logic 1 during the Read cycle if the core is storing a logic 1.

10. Driver A2 will cause a current to flow in the inhibit line, during a restore cycle, if the flip-flop is 0.

11. If the flip-flop is clear during the write cycle, then the new data bit will be a logic 0.

12. In the grid plane method for selecting specific cores, the cores are arranged in Rows and columns.

DIRECTIONS: For questions 13 and 14, refer to the following figure.



13. Given an address of X1 and Y1, core 3 will be selected and core 0 will not be selected.

14. How many sense lines would be required to read data from the grid plane? 1.

ANSWERS

1. Drive 2. Sense 3. Inhibit 4. Inhibit, sense 5. 0 6. 1
7. A1 and A2 8. 0 9. Read, a logic 1 10. Clear, or storing a logic 0
11. 0 12. Rows, columns 13. 3, 0 14. 1

4K Memory Matrix (Text)

In "4-Wire Core Memory," the grid plane method was discussed as a means of accessing one core from a group of cores. For each read or write cycle, only one core was selected; in real applications, however, when data is being written into or removed from memory, more than one bit of data is desired. By duplicating the grid plane for each desired bit, the grid plane method can be used when more than one core is to be accessed at one time.

Using the Grid Plane

Let's use, as an example, a core memory that can store four words when each word is three bits long. To do this, the core memory will need three grid planes with four cores in each grid plane, as shown in figure 2-19.

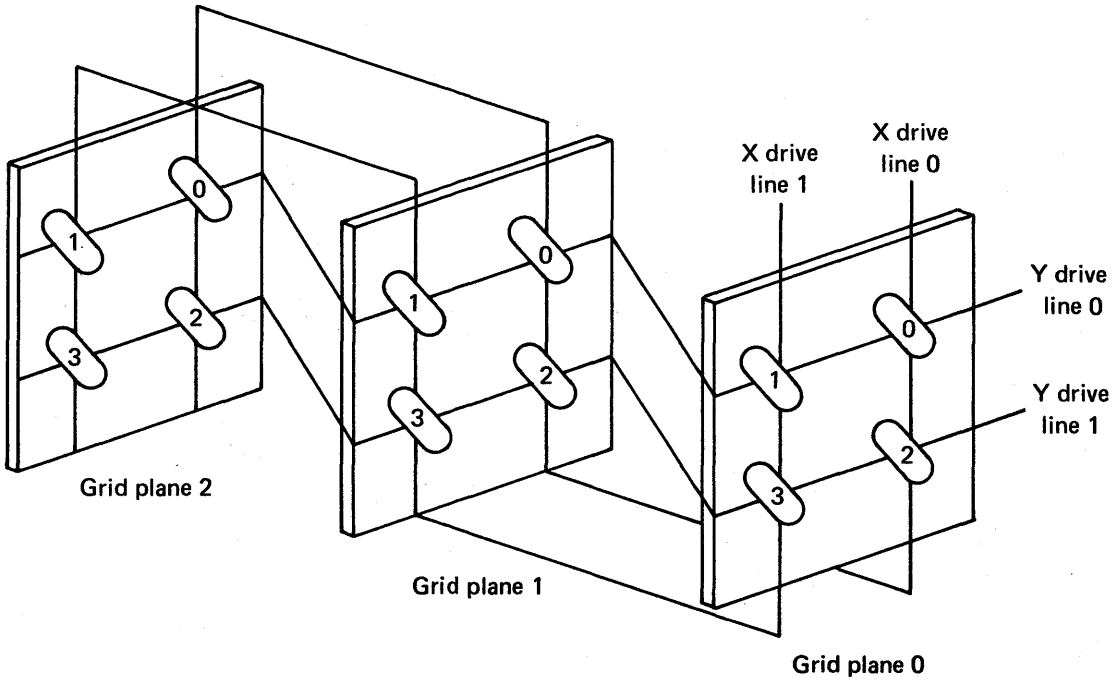


Figure 2-19. 4-Word Core Memory

Four drive lines—two X lines and two Y lines—are used. Each drive line goes through each grid plane. Drive line X0, for example, goes through grid planes 0, 1, and 2. Although no sense or inhibit lines are shown in the figure, there are separate sense and inhibit lines for each grid plane. The drive lines can be common to all planes in the memory. In order to read a bit from each plane, however, it is necessary for each plane to have its own sense and inhibit lines.

The cores in each grid plane are numbered by their position. For example, in figure 2-19, the core in the upper right-hand corner of each grid plane is numbered 0. Table 2-2 is an address-decoding format that can be used to select specific cores from figure 2-19.

TABLE 2-2
Address-Decoding Format

Addresses (Binary)		Selected Drive Lines		Selected Cores (Decimal)
		Y	X	
0	0	0	0	0
0	1	0	1	1
1	0	1	0	2
1	1	1	1	3

Given an address of 01 from table 2-2, the drive lines X1 and Y0 would be used. Core 1, then, has been selected in all planes. Using figure 2-19, follow X1 and Y0. These two drive lines cross once on each grid plane. The cores on each plane where the lines cross make up the three-bit word being stored.

When a read operation is performed on the core memory in figure 2-19, the content of the core selected in grid plane 0 is placed in bit position 0 of the word being read. Likewise, the contents of the cores selected in grid planes 1 and 2 are placed in bit positions 1 and 2, respectively, of the word being read. (Refer to figure 2-20.) As in the earlier description of a grid plane, each bit register position contains a flip-flop, inhibit drives, and the associated gates for differentiating between read and write operations. Note again that, while X and Y drives (an address function) are common to all planes, bit sense and inhibit lines (a data function) are unique to each plane.

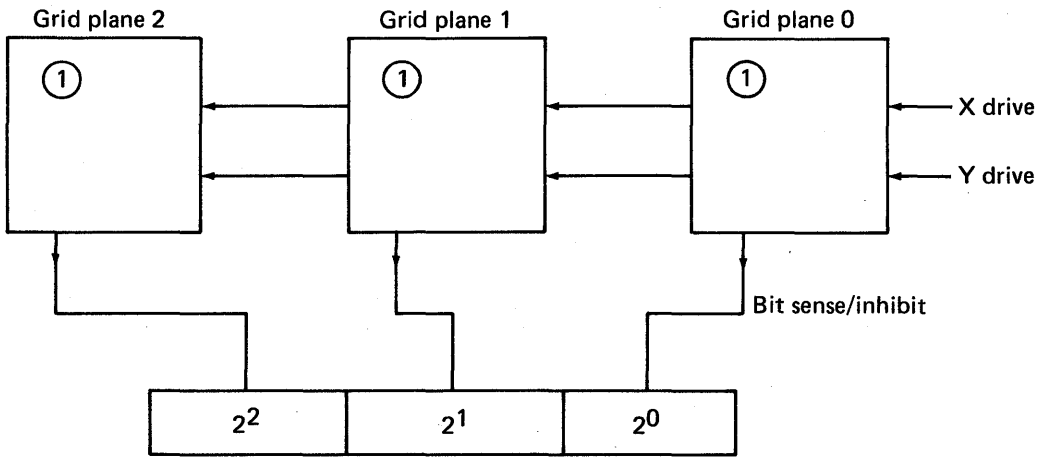


Figure 2-20. Word Read

If the operation being performed is a write or a "store," then each bit of the word would be placed into the address-specified core in each of the grid planes. For example, if the address of 11 were given, then core number 3 in each grid plane would be selected. (Refer to figure 2-21.)

If the word being stored were 101, then core 3 of bit plane 0 would be storing a logic 1; core 3 of bit plane 1, a logic 0; and core 3 of bit plane 2, a logic 1. If a read operation were then performed on location 3, the word being read would be 101. Using the terminology of the computer business, we would say, "There is a binary 101 at address 3" or "There is a 101 in location 3."

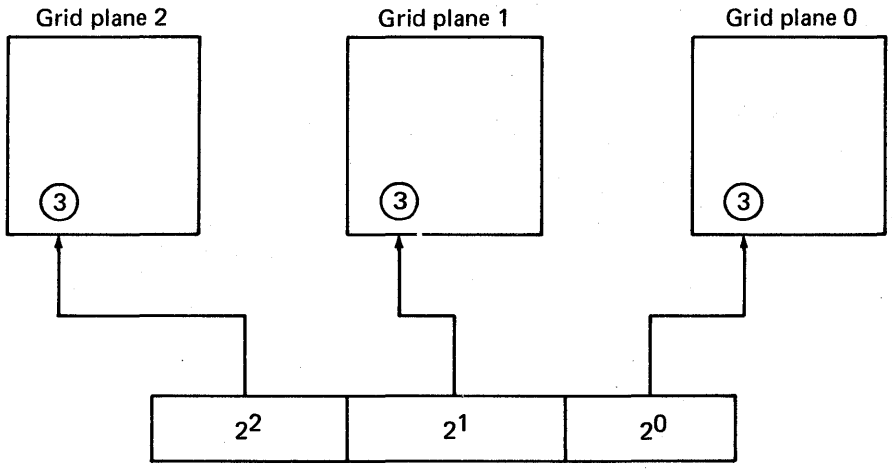


Figure 2-21. Word Stored

Makeup of Core Memory

Grid planes are commonly referred to as bit planes. The number of cores in a bit plane is the same as the number of words of memory; the number of bits in a memory word is the same as the number of bit planes in memory. As an example, if a computer word has twelve bits and the memory size is 1024 words, this implies that the memory would be made up of twelve bit planes, with 1024 cores within each bit plane.

The same grid plane method we have been describing is applied to memories of a more useful size. A common core memory is one that can store 4096 (4K) words, with eighteen bits per word. Each grid plane, or bit plane, contains 4096 cores, the same as the number of words in memory. These 4096 cores are usually arranged in 64 rows and 64 columns. (Symmetry in the design of core memories yields advantages like compact physical packaging and commonality in circuit components, such as the X and Y driver decoders and current drivers.)

A memory this size requires 64 X drive lines and 64 Y drive lines. Figure 2-22 represents one bit plane of a 4K memory. At each of the intersections of the X and Y drive lines will be one core.

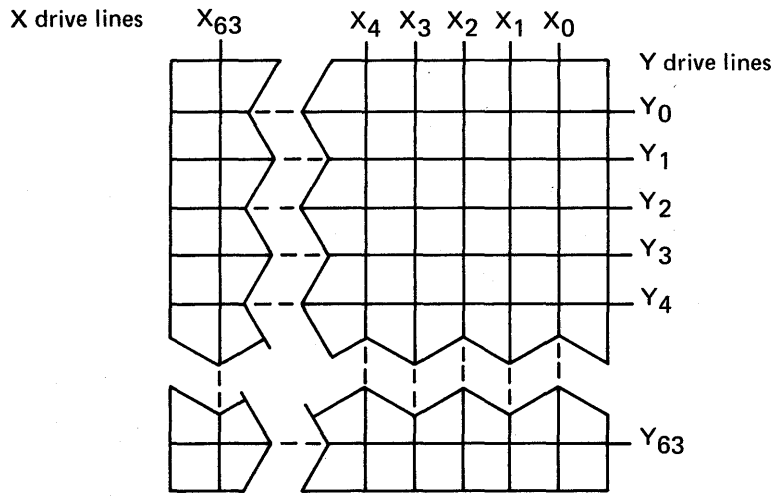


Figure 2-22. One Bit Plane of a 4K Memory

Since there are eighteen bits per word, there are eighteen bit planes, as shown in figure 2-23. The bit planes are numbered 0 through 17. In bit plane 0, there are 4096 cores representing bit 0 for all 4K words of memory; in bit plane 1, 4096 cores representing bit 1; and so forth. Every memory access, whether read or write, selects one core from each bit plane. The X and Y drive lines go through every bit plane exactly as shown on the earlier and simpler figures.

If drive lines X0 and Y0 are selected, then the cores where X0 and Y0 cross—one from each bit plane— will switch. The contents of the core read from bit plane 0 goes into bit position 0 for the word being read, the contents of the core read from bit plane 1 goes into bit position 1, and so on for the entire word being read.

Every 4K of memory is called a "packaged functional assembly," or a "module." When the size of a computer memory is increased, it is generally increased by modules, or, in our example, by 4K modules. The maximum size of a memory inside a computer chassis might be 32K; this would be eight modules. To expand a memory even more would require another chassis, called an "expansion" chassis. The expansion chassis is often called "the upper 32K," as compared with the memory in the computer, often called "the lower 32K." Keep in mind that these terms and figures may differ among different manufacturers.

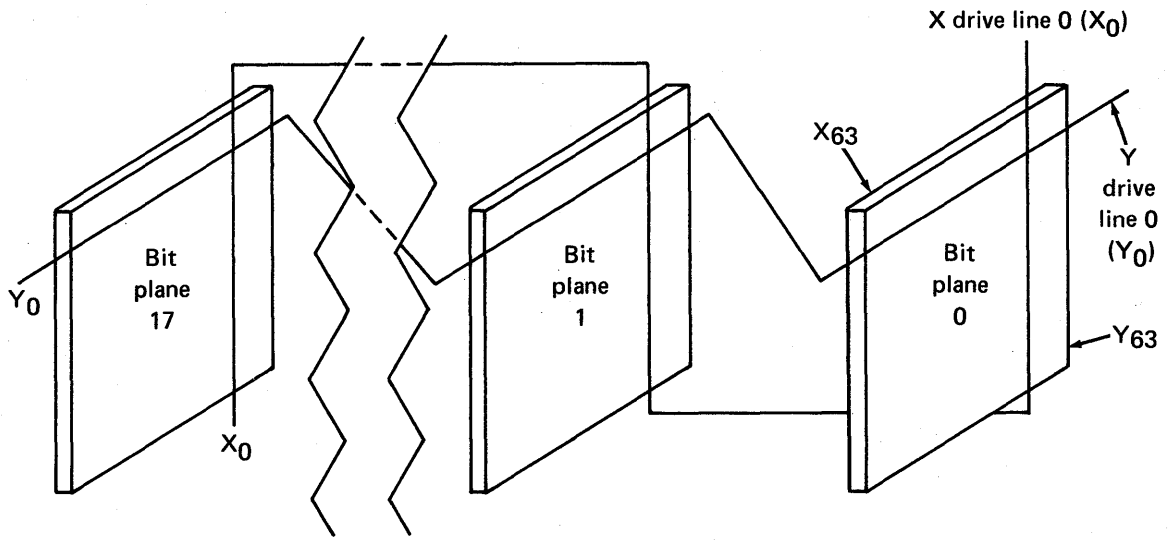


Figure 2-23. 4K Memory Module

Address Decoding

Every computer word in memory is stored in a location designated by an address much like the street address for a house. The computer must take the given address and decode the chassis, module, and the X and Y drive lines where a computer word is stored. The computer follows a fixed format for decoding the address. As an example, refer to figure 2-24.

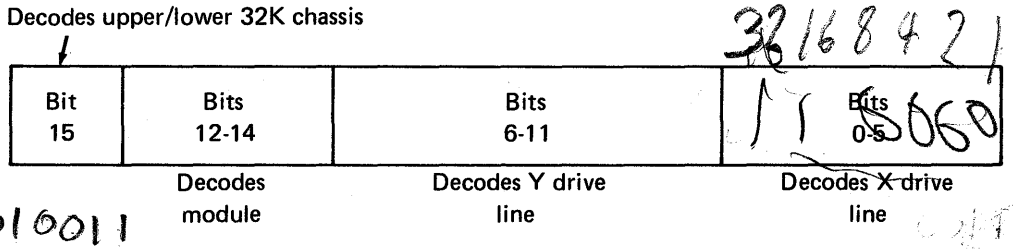


Figure 2-24. Memory Address Format

In the example format in figure 2-24, bit 15 is used by the hardware to decode the upper (bit 15 = a logic 1) or lower (bit 15 = a logic 0) 32K chassis of memory. Bits 12 through 14 are used to determine which of the eight memory modules contains the address. Bits 6 through 11 decode and enable one of the sixty-four Y drive lines. Bits 0 through 5 select one of the sixty-four X drive lines.

Using the format described in figure 2-24, let's decode the address 0111001011011010: lower 32K, module 7, Y drive line OB_{16}, and X drive line $1A_{16}$. Let's try another example of addressing decoding: 1101101001011100. Here the decoded value is upper 32K, module 5, Y drive line 29_{16}, X drive line $1C_{16}$. (Computer memories may differ from our example, but keep in mind that all core memories would be similar.)

Other Terms Relating to Magnetic Storage

Our examples have used a four-wire core memory. This memory has dimensions that consist of width (X select), height (Y select), and depth (number of planes). Within the industry, such a system is commonly referred to as a "3D 4-wire core" memory.

There are, however, other types of magnetic storage memory that we have not covered in our examples. If our examples had used just one wire per plane that served, during the read, as the sense line and, during the write, as the inhibit line, the type of memory would have been described as a "3D 3-wire core."

To properly describe a number of magnetic memories would be beyond the scope of this unit. It would be beneficial, however, for you to use available reference books to scan descriptions of three-, two and one-half-, and two-dimensional memories, as well as core, thin film, and wire types of storage mediums. If you do this, you will find that each type of memory has advantages and disadvantages; no one type best satisfies all applications. When compared to other random access storage devices, such as IC circuit data storage, magnetic data storage devices do have the advantage of being able to retain data reliably through long power-off periods. This characteristic is generally referred to as "nonvolatile storage."

Summary

In order to access whole words of memory, cores are arranged in a series of planes. Drive lines go through all planes; sense and inhibit lines are unique to each plane. The number of cores in a plane determines the number of words in storage; the number of planes determines the number of bits in each word. Every word is stored in a location that has a unique address specifying chassis, module, and X and Y drive lines.

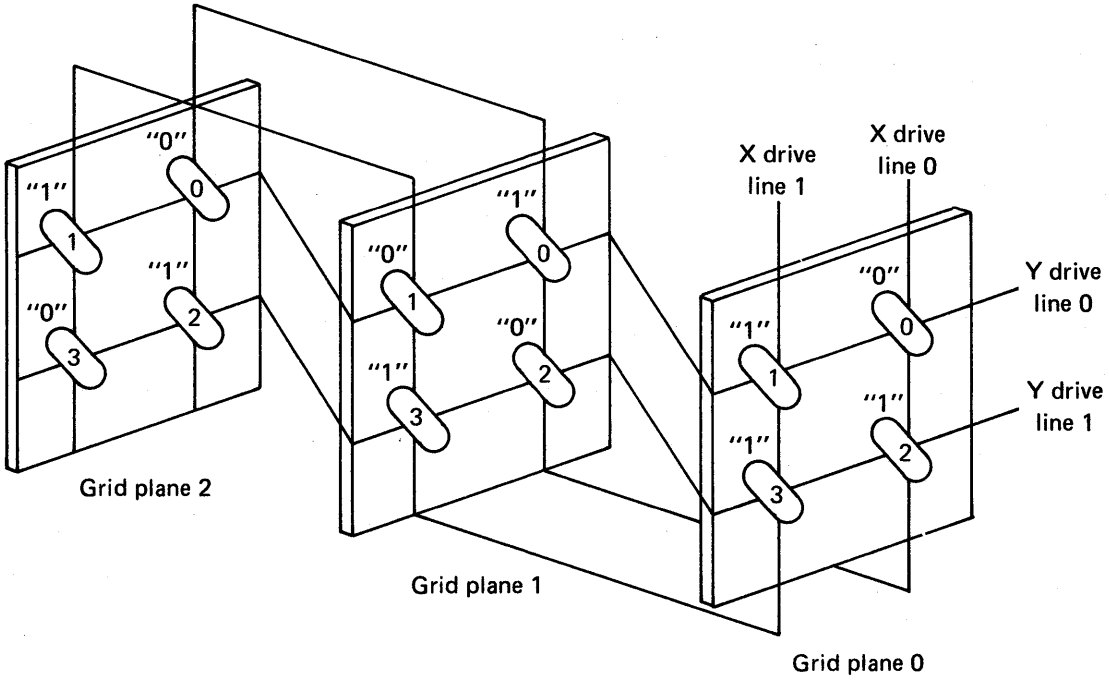
4K Memory Matrix (Exercise)

DIRECTIONS: Answer the following questions, or complete the following statements.

1. A core memory that has individual cores arranged in rows and columns has _____drive lines through each core.

2. A core memory can store 1024 words when each word is ten bits long. How many cores are in one bit plane?_____. In this core memory, there are_____ bit planes.

3. Each core in the core memory shown in the following figure has the value it is storing written beside it. The address supplied to the memory during a read cycle is 11. What is the content of the word being read?_____.



4. A bit plane from a 4K memory can store _____ bits.

5. Decode the following address, and determine the X and Y drive lines enabled in hexadecimal, the module enabled, and the 32K enabled.

   Address: 0101011011001001

   X drive line _____
   Y drive line _____
   Module _____
   _____ 32K

6. In a 4K core memory, whose words are sixteen bits long, there are _____ inhibit lines and _____ sense lines.

ANSWERS

1. 2 2. 1024, 10 3. 011 4. 4096 bits 5. 09, 1B, 5, lower
6. 16, 16

3-Wire Core Memory (Text)

Computer manufacturers, like other manufacturers, attempt to produce equipment that does the job in the simplest and most economical way possible. This text explains how a core memory can be simplified by having the same wire serve as both the sense and inhibit wire.

Combining Sense and Inhibit

During a read cycle in the 4-wire core memory, the X and Y drive lines select one core on each bit plane. The sense line is used to read the condition of the core. The inhibit line is not used. During a restore cycle, the X and Y drive lines are selected and the inhibit line is used to control what is rewritten back into the core. The sense line is not used. In fact, no core memory operation requires the concurrent use of sense and inhibit lines. Since they are not used concurrently, they may be combined into one.

In 3-wire core memory, there are three wires through every core. They are the X and Y drive lines and the sense/inhibit line. During the read cycle, the sense/inhibit line functions as a sense line, and during the restore or write cycle, it functions as an inhibit line. (See figure 2-25.) Except for having one less wire, 3-wire core memory operates no differently from four-wire core memory.

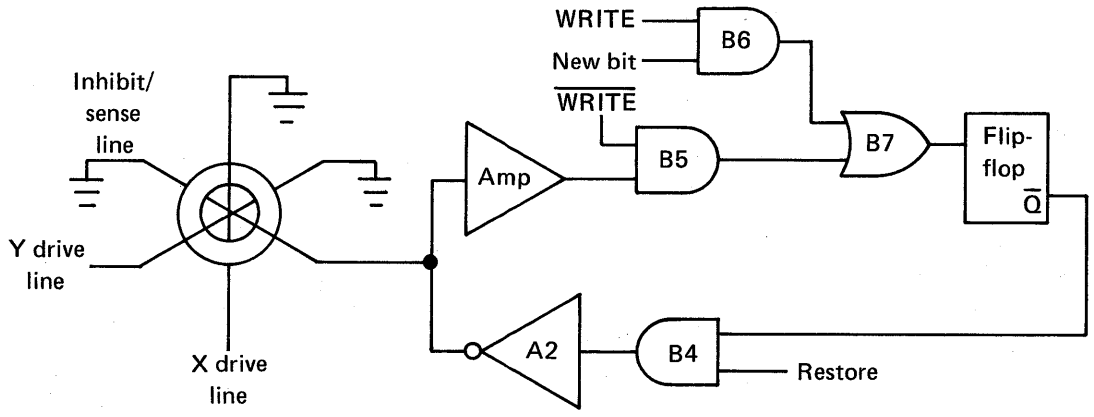


Figure 2-25. 3-Wire Core Memory

To read from 3-wire core memory, the address first selects the proper X and Y drive lines. The current in the drive lines will cause one core in each bit plane to switch to 0 if it has been previously switched to a 1. If the core switches, a current is induced into the sense/inhibit line. The amplifier outputs a logic 1 and sets the flip-flop. (Usually, preread timing would have cleared all the flip-flops prior to a cycle.) Reading will have left the addressed cores storing a logic 0. In the restoration of core content, gate A2, which is controlled by gate B4, outputs a current if the flip-flop is holding a 0. The output current from A2 opposes the current in the drive lines. This causes a logic 0 to be left in the core. If the flip-flop is holding a 1, then A2 will not output a current and a logic 1 will be written into the core.

During the write cycle, the states of some of the selected cores are usually (depending on the data) switched, but are not gated to the flip-flop input. The purpose of the read cycle during a write operation is to clear the cores before storing new data. The new data will be used to set or clear the flip-flops. The restore cycle works as described previously.

Summary

Remember that each bit plane has only one sense/inhibit line, one sense amplifier and flip-flop, and one drive gate to operate the inhibit portion of the sense/inhibit line. So in all operations the 3- and 4-wire core memories are identical except for the shared use of a single sense/inhibit line.

3-Wire Core Memory (Exercise)

DIRECTIONS: Answer the following questions.

1. In 3-wire core memory, when is the sense line used? _____ _____.

2. When is the inhibit line used?_____.

3. What is the difference between the read cycles of 4-wire and 3-wire core memories? _____

4. What is the difference between the write cycles of 4-wire and 3-wire core memories?_____.

5. Name the wires that go through the cores in 3-wire core memory. _____.

ANSWERS

1. Read cycle 2. During restore 3. No difference 4. No difference
5. X drive lines, Y drive lines, sense/inhibit lines

Appendix A

Test Item Diagrams

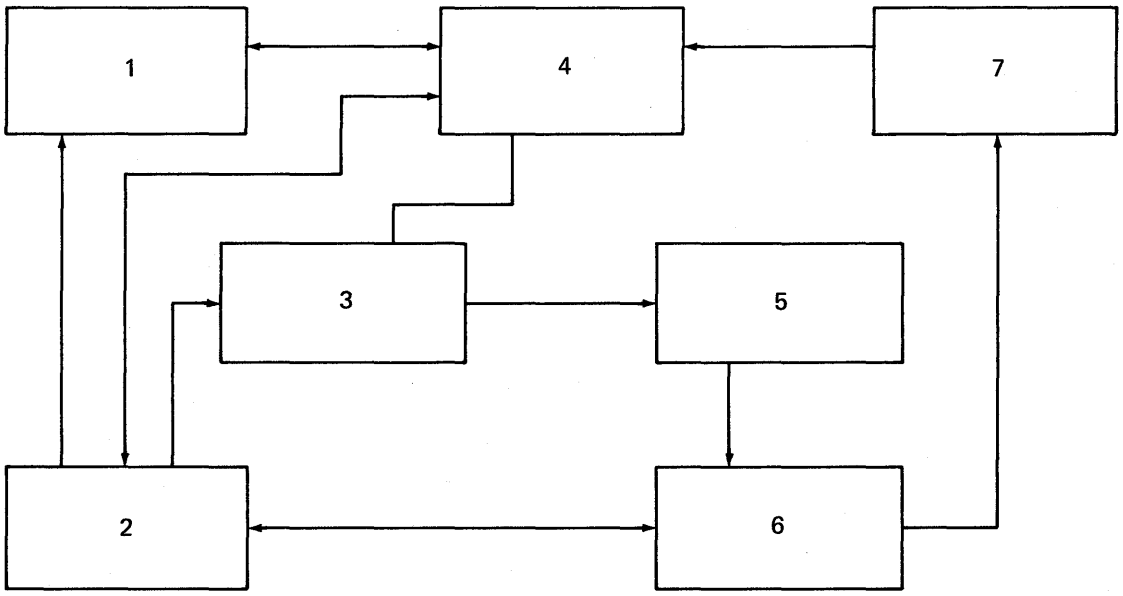


Diagram 2.1

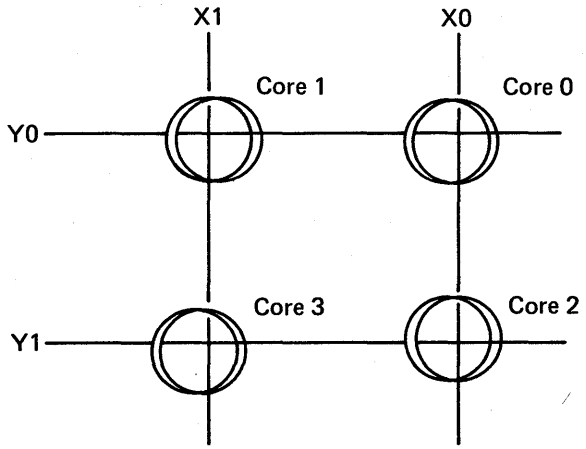


Diagram 2.2

Appendix A

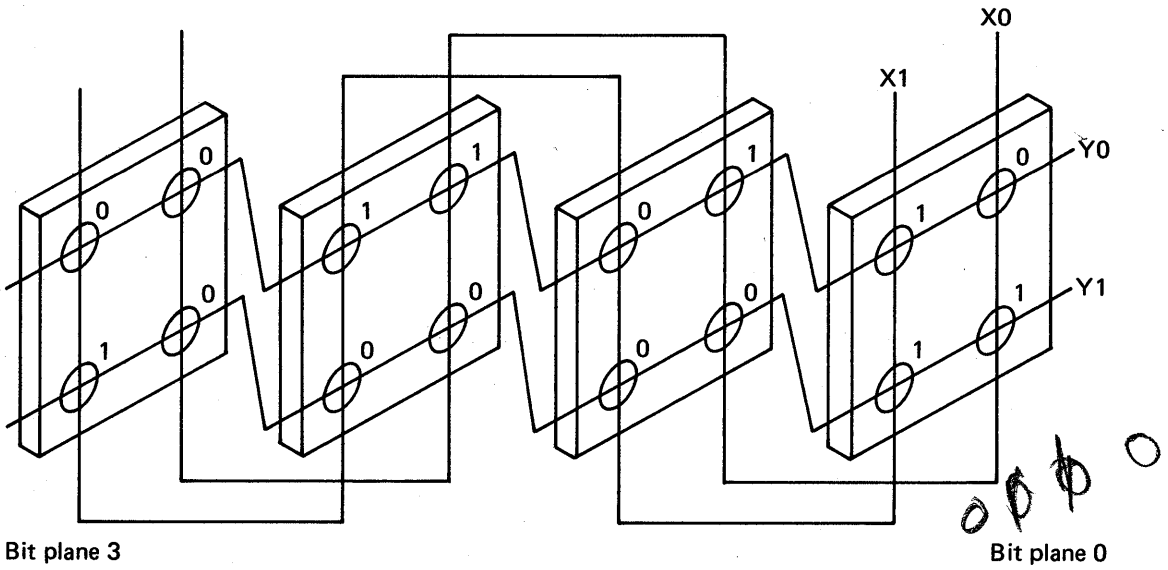


Diagram 2.3

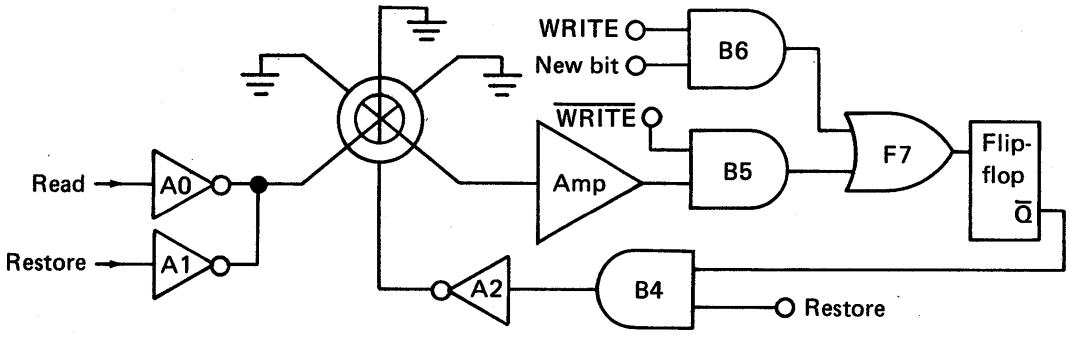


Diagram 2.4

Four-Word Core Memory

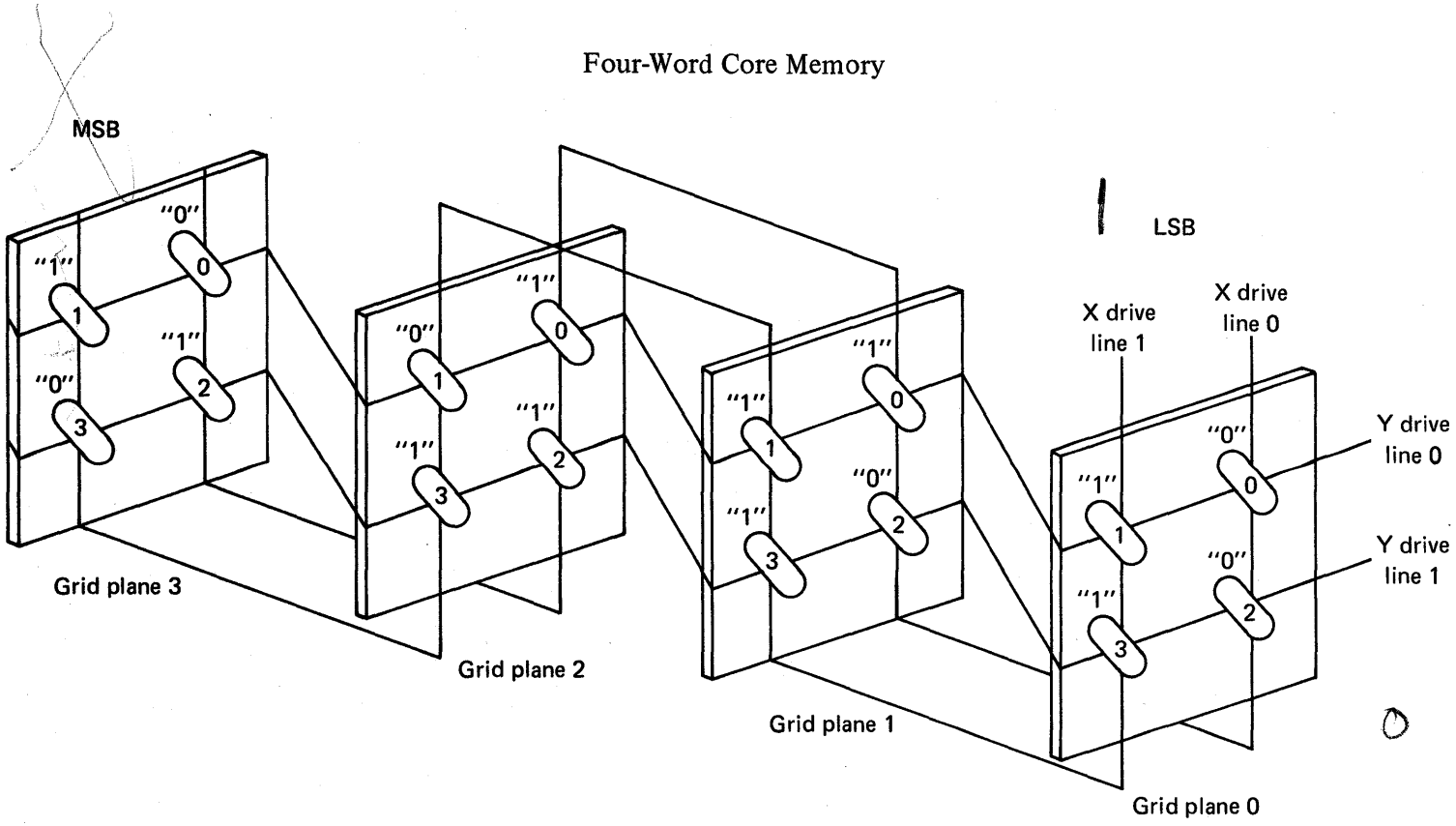


Diagram 2.5