APRIL 1974
Fred Horne
National Semiconductor

MICROPROCESSORS – AN INTRODUCTION

"Computers" have attracted widespread interest only recently, although computing devices have long been known. The Antikythera mechanism (an ancient Greek astronomical computer), the astrolabes of the middle ages and Pascal's calculator are examples of early computational devices. However, the present usage of the term "computer" includes neither those relatively primitive aids for computation, nor later developments such as the slide rule or the desk-top and portable calculators. We now define a computer as a machine which performs a computation automatically and without human intervention, once it is set up for a specific problem.

The present use of the term "computer" has a second connotation—it usually refers to an electronic device, although mechanical and electromechanical computers do exist. Two important factors dictate the intimate association between computers and electronics: No known principle other than electronics allows a machine to attain the speeds now commonplace in both large and small-scale computers; no other principle permits comparable design convenience.

Even though almost all modern computers (or "information processors") operate electronically, there still exist at least two distinct classes of machines. We do not mean circuit element differences, such as tubes or transistors, but basic differences in design philosophy. The most significant distinction is the analog or digital nature of a computer.

An analog computer represents values of interest by physical quantities. The slide rule, an analog device (although by definition, not a computer), uses the physical quantity "length" to represent computational values. Electronic analog computers use higher or lower voltages to represent larger or smaller quantities, respectively. In contrast, a digital computer uses numbers, much as we do in paper-and-pencil calculations. Numbers are represented by the presence or absence of a voltage level or pulse on a given line. A single line defines one "bit" (short for 'binary digit,' a base-2 number); a group of lines considered as a unit is called a 'word,' where a word may represent a computational quantity or a machine directive.

For purposes of illustration, we shall compare two systems for solving simple mathematical expressions, both of which are comprised of the classical elements of a computer: An input/output device, a memory, a control section and an arithmetic and logic unit or ALU (the computational element). The control unit, together with the ALU is considered to be the central processing unit (CPU).

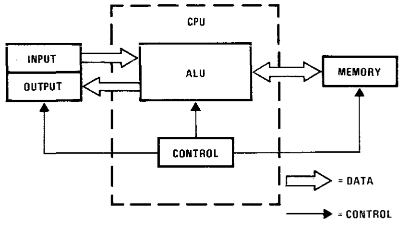

FIGURE 1. Basic Elements of a Digital Computer

The first, shown in Figure 2, is comprised of a man with a calculator; the man's fingers represent the input, his eyes coupled with the calculator's output represent

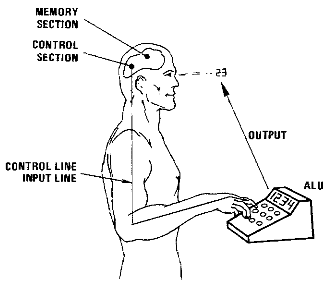

FIGURE 2. Man + Calculator $\overset{?}{=}$ Computer

the output, his brain is the control element while the calculator electronics function as the ALU, finally his brain also serves as the memory.

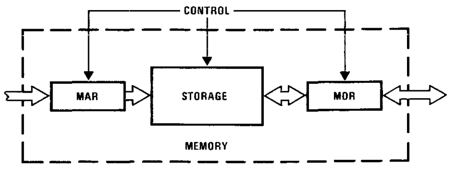

FIGURE 3. Elements of a Memory

Let us examine the sequence of events which occur when our man-calculator solves the problem 6 + 2 = ?

1. Brain accesses first number of be added, a "6,"
2. Brain orders hand to depress "6" key,
3. Brain identifies addition operation,

4. Brain orders hand to depress "+" key,
5. Brain accesses second number to be added, a "2,"
6. Brain determines all necessary information has been provided and signals the "ALU" to complete computation by ordering hand to depress "=" key,
7. ALU (calculator) makes computation,
8. ALU displays result on readout,
9. Eyes signal brain, brain recognizes this number as the result of the specified calculation,
10. Brain stores result, "8," in a location which it appropriately identifies to itself to facilitate later recall.

We shall now develop a classical computer and illustrate how it might be used to solve the same problem. First, the memory which is composed of storage space for a large number of "words," with each storage space identified by a unique "address." The word stored at a given address might be either computational data or a machine directive (such as add, read from memory, etc.). Two temporary store registers, each capable of containing one word, complete our memory (Figure 3). These registers are designated as "memory address register" (MAR) and "memory data register" (MDR). The MAR contains the binary representation of the address where information is tó be read from memory or written (stored) into memory, while the MDR contains the data being exchanged with memory.

Next we add the ALU. The simplest ALU is comprised of an "adder" which adds (or performs similar logical operations upon, e.g., "or") two inputs, A and B, and produces an output at C; and an "accumulator," which contains intermediate results of a computation or numbers for a pending computation.

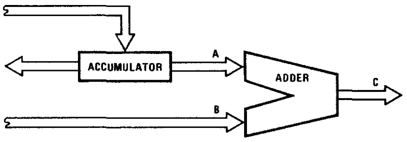

FIGURE 4. Arithmetic and Logic Unit

The remainder of our CPU, the control portion is implemented using an "Instruction Register" (IR), a "control decoder and sequencer," and a program counter (PC). A machine directive (instruction) is transferred into the IR and is subsequently interpreted by the decoder/sequencer, which issues the appropriate control pulses to the other computer elements. The PC contains, at any given time, the address in memory of the next machine directive, or instruction. This counter is normally incremented by one immediately following the reading of a new instruction. The PC contents may be replaced by the contents of a specified memory location if the last instruction was of the "jump" class. This causes the next instruction to be read from a program-specified location instead of from the next sequential location, as is the general rule.

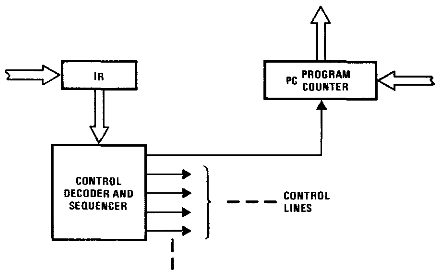

FIGURE 5. Computer Control

Finally, a means of Input/Output (I/O) is provided via an "I/O Register," through which data exchanges take place with external, or peripheral devices.

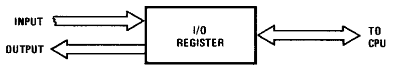

FIGURE 6. I/O Register Interface

We have now collected together all of the basic elements of a computer; all that remains to do is to interconnect them into a functioning automatic processor, as shown in Figure 7.

Voila! A complete computer! Let's now continue our analysis by executing the problem described below (note that it is the same problem used to illustrate the man-calculator):

> "Read in a number from the I/O. Store it in memory location 50. Read in another number from the I/O. Store it in memory location 51. Add the two numbers together. Store the result in memory location 60, and halt."

A "program" has been written to execute this task, and is stored in consecutive memory locations beginning at 100. This program, written in an artificial symbolic "language," is shown below.

TABLE I. Sample Program

Memory Location	Instruction (Contents)
100	Input to accumulator
101	Store accumulator at 50
102	Input to accumulator
103	Store accumulator at 51
104	Add accum, Loc. 50 Place result in accumulator
105	Store accumulator at 60
106	Halt

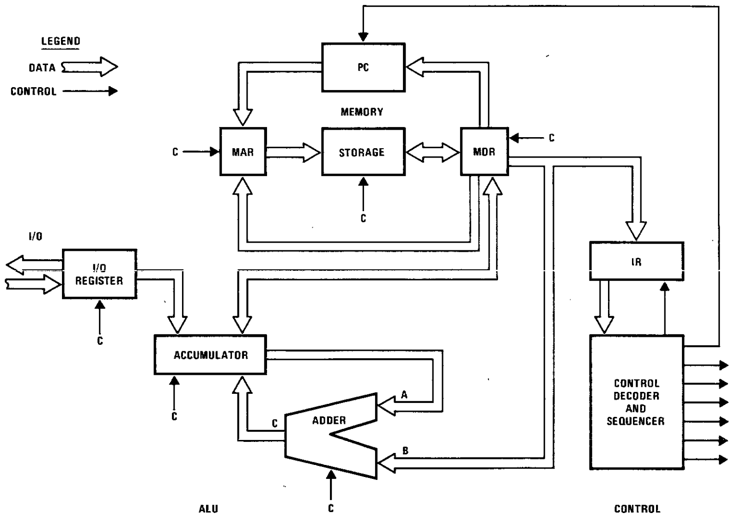

FIGURE 7. Simplified CPU + Memory

All computers spend approximately equal periods of time residing in one of two distinct "states": Fetch or execute. In the fetch state, the computer reads from memory the next sequential instruction and places it in the instruction register (IR). During the execute state-time, that instruction is carried out as a series of transfers from one register to another and various ALU operations. We shall now examine the program shown in Table I as it actually executes, by specifying the contents of each register at each machine cycle (time interval) and assuming the computer is now ready to fetch the first instruction in our program. (See Table II for this analysis.)

The operation is complete. No human intervention was required—all operations were automatic.

All computers (processors, CPU's, etc.) operate in a similar manner, regardless of their size or intended purpose. It must be emphasized that many variations are possible within this basic architectural framework. More common variations include highly sophisticated I/O structures (some of which have direct and/or autonomous communication with memory), multiple accumulators for programming flexibility, index registers which allow a memory address to be modified by a computed value, multi-level interrupt capability, and on and on.

One of the most exciting architectural concepts to gain popularity in the past few years is that of microprogrammed control. A microprogrammed computer differs from the classical example just presented in the control unit implementation. Our classical machine has for its control unit an assemblage of logic elements (gates, counters, flip-flops, etc.) interconnected to realize certain combinatorial and sequential boolean equations. On the other hand, a microprogrammed machine utilizes the concept of a "computer within a computer." That is, the control unit has all the functional elements which comprise a classical computer, including read only memory (ROM).

The purpose of this "inner computer," which is not apparent to the user, is to execute the user's program instructions by executing a series of it's own microinstructions, thereby, controlling data transfers and all functions from *computed* results. Herein lies the key distinction: The control signals, hence the very "personality" of the computer, are controlled by computed results. The implication is immediately obvious—by simply changing the stored microprogram which generates the control signals, we may alter the entire complexion of the computer. By altering a few words stored in the ROM, we can cause our computer to behave in an entirely new fashion—to execute a completely different set of instructions—to emulate other computers—to tailor itself to a specified application. It is this capability for "custom-tailoring" that allows such a machine to be optimized for a given usage. By so extracting the utmost measure of efficiency, a microprogram-controlled machine is more reliable, less costly and easier to adapt to any given situation, no matter how diverse or demanding—such as that associated with an on-board automotive microprocessor. National Semiconductor recognized the enormous advantage of being able to "have your cake and eat it too," and therefore has proceeded to exploit the power of re-microprogrammability.

TABLE II. Register Content

Notes	PC	Accum.	MAR	MDR	I/O Reg.	IR	Memory Read-R Write-W	State
Enter	100	?	?	?	?	?	?	Fetch
	100	?	100	(100)*	?	?	R	Fetch
	101	?	100	(100)	6	(100)	?	Fetch
Input	101	6	100	(100)	6	(100)	?	Execute
	101	6	101	(101)	?	(100)	R	Fetch
	102	6	101	(101)	?	(101)	?	Fetch
(Store)	102	6	50	6	2	(101)	W	Execute
	102	6	102	(102)	2	(101)	R	Fetch
	103	6	102	(102)	2	(102)	?	Fetch
(Input)	103	2	102	(102)	?	(102)	?	Execute
	103	2	103	(103)	?	(102)	R	Fetch
	104	2	103	(103)	?	(103)	?	Fetch
(Store)	104	2	51	2	?	(103)	W	Execute
	104	2	104	(104)	?	(103)	R	Fetch
	105	2	104	(104)	?	(104)	?	Fetch
(Add)	105	2	50	(50)	?	(104)	R	Execute
	105	8	50	(50)	?	(104)	?	Execute

Note: Addition has occurred; content of MDR (6) is at adder input B while content of accumulator (2) is at adder input A; result at C replaces previous accumulator contents.

Notes	PC	Accum.	MAR	MDR	I/O Reg.	IR	Memory Read-R Write-W	State
	105	8	105	(105)	?	(104)	R	Fetch
	106	8	105	(105)	?	(105)	?	Fetch
(Store)	106	8	60	8	?	(105)	W	Execute
	106	8	106	(106)	?	(105)	R	Fetch
	107	8	106	(106)	?	(106)	?	Fetch
(Exit)	107		HALT					

*(100) is read as "contents of 100"

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 732-5000/TWX (910) 339-9240
National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 27649
National Semiconductor (UK) Ltd.
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632