How a Computer Computes? Hardware and Software Based Pacemakers

MAREK MALIK

From the Department of Cardiological Sciences, St. George's Hospital Medical School, London, England

Introduction

During recent years, we have witnessed serious disputes about the security of software based cardiac stimulators. In order to comprehend the basis of these disputes and to realize why the software based systems are possibly less secure than the hardware systems, one has to understand the difference between hardware and software based computing systems.

The difference between hardware and software founded computations will be described. As the software based computation is the groundwork for each computer, this article will also explain how a computer computes and what enables the various computer applications to be as flexible as they are.

Hardware Computing

Broadly speaking, electronic components can be classified into two categories: analog and digital. The analog circuits (e.g., the input parts of a sensor of a pacemaker) operate with electrical impulses of variable voltage (e.g., the intracardiac electrocardiogram); the digital circuits operate with impulses that have only two voltage levels—high and low.

Computing—that is, execution of a more or less complex chain of logical actions—can be based on analog and digital electronics. Analog computers, in which the computation is based on tracing the changes of voltages and electric currents, are especially suited for (and practically restricted to) solving differential equations. The peak of their use appeared roughly two to three decades ago when digital computers were substantially slower than they are today. Advances in digital electronics, together with the restrictions of the analog computers, have made analog computing obsolete so that only digital electronics will be addressed.

Special electrical circuits can be designed that perform various operations on digital impulses, such as inverting the high-low pulses, counting the changes from low to high, etc. Combination of circuits implementing such elementary operations is frequently done within one chip (hence the name *integrated circuit* for the building clocks of electronic components). Combination of several integrated circuits enables the specialized hardware to perform very complicated operations. Every particular algorithm, that is, a way and concept of computing something, e.g., DDD pacing mode, solution of a given differential equation, guiding a sophisticated missile, etc., can be implemented in a form of specialized hardware—that is, a complex circuit that is unable to calculate anything else.

Frequently, hardware based computing systems incorporate *memory*, i.e., special circuits that can obtain, hold, and export numeric information. The memory is then used to store parameters of the actions that are performed by the fixed "computing circuits." Memory can also contain switches that activate different parts of the computing circuits thus changing the nature of operations performed by the whole electronic system. In no case, however, can a hardware based system perform operations other than those for which the computing circuits, or individual parts of the computing circuits, were built. For this reason, the hardware based systems are also sometimes called *fixed logic equipment.*

Address for reprints: Prof. Marek Malik, Department of Cardiological Sciences, St. George's Hospital Medical School, London SW17 ORE, England. Fax: 081-767-7141.

Received May 21, 1992; accepted May 28, 1992.

A typical example of fixed logic equipment is a standard hardware based pacemaker. Even a multiprogrammable hardware based device that can operate in several modes, say, ranging from VOO to DDD, and that can be programmed to various values of refractory periods, pulse durations, etc., cannot operate outside of the framework of its logic. For example, it is obvious that if such a device does not contain circuits implementing an antitachycardia option, we cannot force it into an antitachycardia mode by reprogramming the contents of its memory.

Software Computing

Naturally, each computer is also based on complicated electronic circuits. However, compared to devices with fixed logic equipment, these circuits are of a completely different type. A computer memory contains not only a set of parameters of the currently performed calculations, but also—and more importantly—contains the program, i.e., the description of the operation that should be performed.

In order to achieve this, the electronic circuits of the computer (called its *processor*) have to be able to decode the contents of the memory and to interpret it as a sequence of individual actions. The actions that are coded in the memory are rather elementary, such as adding two numbers that are written somewhere else in memory, and writing the results into a specific place in memory.

The codes of these elementary operations, called the *instructions*, conform to the hardware design of the processor. In other words, an instruction is a number of the corresponding operation. The design of the processor dictates that, say, 427 means adding two numbers, 764 comparing two numbers, etc.

Similar to the possibility of constructing different hardware for each particular algorithm, each algorithm can be converted into a chain of instructions—that is, into a *program*. Frequently, especially with less complicated algorithms, it is substantially less costly to produce a program for a universal processor than to build specialized hardware.

From a purely theoretical point of view, there is little conceptual difference between the processor of a computer and fixed logic electronic circuits. Abstractly speaking, the logic of the actions performed by the processor is also fixed; that is, the processor decodes individual instructions and carries out the corresponding operations. From a pragmatic point of view, however, the difference is enormous. The practical meaning of the actions performed by the processor depends solely on the program written in the memory and is independent of the hardware structure of the processor (with some possible limitations regarding the speed, amount of data, etc.).

Thus, by changing the contents of the memory, we may change the logic of operations performed by the processor completely, and even force the processor into fulfilling actions that were not at all foreseen by the designers of the hardware.

How a Computer Operates

As a rule, software oriented processors operate with memory of at least two types: conventional memory and the so-called *registers*. The size of the registers is much smaller than that of the conventional memory but the possibilities of operations that the processor can perform with the contents of the registers are much more flexible than the possibilities of operations that can be carried out with the contents of the memory. For example, all common processors can perform many numerical operations (e.g., division by 2) only with the contents of registers.

One of the registers of every software oriented processor has a special meaning and controls the flow of the program, that is determines the sequence of executed instructions. This register is called the *program status word* (PSW) and contains the memory address (i.e., the locations within the memory) of the instruction that has to be executed next.

The flow of the program is organized by a part of the processor hardware that operates in rather simple steps. At each step, it reads the contents of the memory at the address specified by the current contents of PSW. The instruction that is read from the memory is passed to other circuits of the processor for decoding and execution and the contents of PSW is increased by the size of the instruction that has just been read. This means that if the contents of PSW are not changed during the

execution of the instruction, the next instruction of the program will be read from the immediately following memory address. However, some instructions (the so-called jumps) change the contents of the PSW register in order to change the sequence of the executed instructions. Branching from one part of the program to another is enabled in this way.

What is Wrong with Software Based Pacemakers

From what has been said, the principal limitations of the security of software based pacemakers are obvious.

It is very unlikely that an external interference, such as a strong electric field, would change the contents of the memory of the pacer. However, should this happen in a hardware based stimulator with multiprogrammable capabilities, the pacemaker may change its mode, for example, from VVI to DDD, and operate according to inappropriate settings of refractory periods, pacing stimuli durations, etc. However, all these parameters would still be within the ranges of the multiprogrammable settings because the principle algorithm built into the computing circuits of a hardware based pacemaker is fixed and cannot be changed by any external interference (programming of such a pacemaker as well as a random modification of its memory can only change the parameters of the fixed algorithm).

If, however, such a random change of the memory contents occurred with a software based stimulator, the results can be really disastrous. The PSW register would point to a place in the memory that would no longer contain a sensible instruction but this "instruction" would still be interpreted and executed by the processor. As a result, the behavior of the pacemaker will be completely unpredictable. For instance, there is no guarantee that after a random change of the memory the pacemaker will not deliver a fast and irregular burst of long duration pacing stimuli to the ventricles, which will initiate ventricular fibrillation.

It must be repeated that the chance of a random change of pacemaker memory is very remote, but the principal differences between hardware and software based devices are such as I have just described. Implementing a new pacing mode into a software based stimulator is much less expensive than developing a new hardware based unit and therefore, these devices will probably become more common than they are today. However, securing the contents of the memory of the software based stimulators must have the highest priority in their design.