USING A COMPUTER TO SIMULATE A COMPUTER

By GEORGE R. TRIMBLE

ABOUT THIS ARTICLE

Innovations in computer design, which motivate us to replace existing computer systems with new and better systems, also compel us to develop techniques for simulating the new computer before it arrives. Called simulation and emulation, these techniques enables us to imitate all the functions of the new computer using the old computer. Not only does this facilitate conversion from one system to another, but it also enables us to test programs for a new computer even before the machine is built. Simulation mainly concerns the software which enables the use of existing machine applications on the new system prior to reprogramming. Emulation involves hardware features specifically designed within the new computer to promote simulation compatibility. Many problems arise because of basic differences in machine operation and simulation objectives.

One of the more exotic applications of digital computers is to simulate a digital computer on another entirely different type of computer. Using a simulation program, application programs developed for the first computer, the source computer, may be executed on a second computer, the object computer.

Simulation obviously provides many advantages in situations where a computer is replaced by a different computer, for which the applications have not yet been programmed. Simulation techniques enable an installation to continue solving problems using existing programs after the new computer has been installed and the old one removed.

Simulation of the replaced computer is obviously a much less efficient means of solving the problem than reprogramming the applications for the new computer. When such a switchover is made, however, it is not practical to reprogram all the applications for the new machine immediately. Use of a simulator permits the installation to continue running its programs as reprogramming proceeds on a reasonable schedule. In fact, for some of the very infrequently used programs, reprogramming may not be worthwhile.

Another situation in which simulation is advantageous is during the development of a new computer. Once specifications for the new computer have been established, programming of applications for the computer can proceed in parallel with hardware development. The use of a simulator in this situation enables the users to debug their applications before the hardware is actually available.

Simulation of an unbuilt computer is advantageous for other reasons as well. Sometimes new equipment has bugs in it which may not be discovered during assembly and final test procedures. Debugging a computer program on a new piece of equipment frequently results in errors which can not be firmly associated with either the equipment or the program. If, however, the program has been debugged on a simulator, greater assurance can be placed in the program and it is more likely that problems encountered are equipment problems.

Another advantage of simulation is that sophisticated debugging aids can be built into the simulator which would be very difficult, if not impossible, to build into a debugging system for the computer being simulated. Selective dumping and tracing are examples of the type of debugging facilities which can be incorporated into a simulator.

Simulation Objectives

A simulator has two principal objectives. First and

foremost, the simulator must faithfully duplicate the functions of the computer being simulated. The results of arithmetical and logical operations must be precisely the same as those which would be obtained if the program were run on the source computer itself. Simulation of the timing of asychronous functions may not be as critical, but in some situations even the timing must be simulated as precisely as possible.

By faithful simulation it is not to be implied that functions on the circuit level must be duplicated, but rather functions which are pertinent to the programmer must be simulated. For example, the contents of the arithmetic registers and the status of various indicators must be duplicated. The functions of an adder or a shift register do not have to be duplicated precisely, but the effects of these devices on the contents of the registers which are accessible to the programmer must be duplicated.

The second principal objective of a simulator is speed of simulation. Since the process of simulation is inherently inefficient, it may occur that the program may operate slower on the object computer than it does on the source computer even though the object computer is considerably faster than the source computer. Advantage must be taken of every feature of the object computer to increase speed of simulation. This implies that great care must be taken in the coding of critical portions of the simulator, such as the instruction fetch and basic instruction execution routines. It frequently occurs that a function which is common to many subroutines should not be made a subroutine itself but instead should be programmed in its entirety each time that function is required. For example, a subroutine to simulate a subtract instruction differs from the subroutine to simulate an add instruction only in that the sign of the operand is reversed. It would be much more efficient from a storage view point to reverse the sign of the operand and then transfer control to some point within the subroutine which simulates an add instruction. This approach, however, would require at least one additional branch instruction to be executed when a subtract instruction is simulated. It is, therefore, faster to write out the subtract subroutine in its entirety, separate and distinct from the add subroutine, even though 90 per cent of the instructions may be identical.

This philosophy could be carried out to an extreme, however, and judgment of when to duplicate coding must be based on the frequency with which specific functions are performed. If the object computer is limited in its storage available, it may be necessary to sacrifice speed in order to make the program even fit within the memory space available.

Approach to Simulation

The characteristics of simulators described above point out the necessity for having two types of simulators. Changing an installation from one computer to another means that the programs to be executed have already been debugged and are in a production state.

ABOUT THE AUTHOR

George R. Trimble, Jr., was graduated from St. John's College in 1948 and obtained a B.A. and an M.A. in mathematics from the University of Delaware in 1951. Formerly with the Computing Laboratory of the Ballistics Research Laboratories and a senior staff member in the Applied Science Div. of IBM, Mr. Trimble joined Computer Usage Co., Inc., in 1955. Utilizing experience on a multitude of computer systems, he now performs analyses and supervises major programs for many applications using a wide variety of machines, including IBM 360, 650, 702, 704, 1401, 7030, 7070, 7090, 7740, Honeywell 800, Bendix G-15, ISI-609, Univac 1107 and ASI-420.

Speed is the most important factor in this situation and debugging facilities are minimal or non-existent. This type of simulator is called a production simulator.

The other type of simulator is used to debug programs for an unbuilt computer and is called a debugging simulator. Speed is also important here but many additional features are built into the simulator to facilitate debugging of programs. In fact, the debugging facilities made available in the simulator can be so significantly superior to those available in the object computer that it is preferable to debug programs using the simulator even if the object computer has been built and is available.

The classical organization of digital computers is into control, arithmetic, memory, and I/O sections. The development of a simulator very much parallels this organization

The memory of a computer may consist of words or characters. A portion of the object computers memory is set aside so that the words or characters of the source computers memory can be simulated. If both computers have word memories, for example, one word of the object computer may be used to represent one word of the source computer.

Obviously this depends upon the word length of the two computers. In some cases it may be possible to represent two source computer words in one object computer word, while in other cases it may be necessary to use two object computer words to represent one source computer word. An analogous comparison can be made for simulating a character memory in a word memory in which several characters of the source computer may be packed in one word of the object computer's memory for efficient utilization of memory.

The control section of the computer is simulated by having a register which performs the same functions as the program location counter in the source computer. An instruction fetch routine is used to refer to the contents of the simulated program location counter in order to determine the location of the next instruction to be simulated. The instruction fetch routine then refers to simulated memory to obtain the instruction and the contents of the simulated memory location are analyzed to determine what must be done. The instruc-