A request for input on implementation of the proposed Microprocessor Assembly Language Standard

Wayne P. Fischer, chairman
Microprocessor Assembly Language Standard Committee

The Microprocessor Assembly Language Standard Committee has reviewed comments received from readers of the proposed standard published in the December 1979 Computer. Two issues requiring additional input from the user community have been identified. The first is the ordering of operands; the other, the length of mnemonics. We think these issues can best be resolved by an informal poll of potential users of the proposed standard. Readers are invited to examine and express their opinions on the following arguments or to present their own ideas.

Arguments in favor of proposed operand sequence

Appropriateness to CPU architectures. Programmers have tended to think of themselves at the logical "center" of the CPU, and this thinking is reflected in the proposed standard, by the position of the mnemonic on the instruction line and by the relative positions of the operands. This ordering is a particularly good model of register- and stack-oriented CPUs, but not such a good model of memory-oriented CPUs. It presumes a two-level memory architecture, in which the registers (or stack) are more central (i.e., closer to the mnemonic) than main memory, and transfers are made to or from this central location. In such machines, ALU operations generally leave their results in the register (or on the stack). In contrast, memory-oriented machines consider their registers to be merely a part of memory more easily addressed than main memory, but not particularly distinguished as the destination of ALU operations. At present, only the LSI-11 can properly be considered a memory-oriented CPU; all other micros are register-oriented. Three factors affect future trends: (1) aesthetics and the fact that the PDP-11 is the standard against which micros are measured tend to motivate designers to produce memory-oriented architectures, but with varying degrees of success; (2) high-level language people press for stack machines, although these are known to be somewhat inefficient; and (3) register machines are consistently selected for execution efficiency and code compactness. To the extent that such efficiency issues figure in new designs, register machines will continue to dominate. It seems unwise, therefore, to compel the majority of CPUs to conform to an assembly language sequence that is more appropriate to a minority of CPU architectures.

Rules for exceptions. While source-to-destination ordering of the operands may be conceptually simple and easy to learn, it cannot be uniformly applied to all multiple-operand instructions, since nearly every microprocessor has at least one instruction with two sources and no destination (e.g., compare), or two destinations and no source (e.g., DBNZ). Arbitrary rules must be contrived for these exceptions.

Conformity to English syntax. Most of the mnemonics in the standard are derived from English words which are used according to pre-existing rules of syntax. For example:

- **Shift**. In English, the object to be shifted may be followed by qualifiers specifying how much or where or how: "Shift this box over to the left three inches."
- **Move**. English sentences usually place a source before a destination: "I helped my grandmother move from New York to Florida."
- **Load**. In English a small container is the target of objects from a larger area (in that order): "You load a truck from a dock."
- **Subtract**. English order is normally subtrahend followed by minuend; but in mathematical notation (taught in grade school), the minuend is first: "Subtract A from B" is written "B − A."

While present programmers may be temporarily comforted by changing habits formed under PDP-11, Intel 8080, and Zilog assemblers, new users can adapt more quickly to operand orderings that correspond to the natural English or mathematical syntax of the same words.

Simplification of learning task. Since the proposed standard’s ordering rules differ from those of some widely used assemblers, they may be (perhaps rightly) considered difficult to learn, at least by those who must learn this as another one of many assembly languages to be used interchangeably. However, when the proposed standard is adopted, and most microprocessors have a conforming assembler, it will be the only language to learn; the exceptions will not seem so exceptional, but natural, as intended.

Argument against proposed operand sequence

The principal objection to the proposed operand sequence is a perception on the part of several commentators that it is inconsistent, as noted above. Many potential users of the proposed standard would apparently be more comfortable with a sequence that always places the destination first, and the source(s) after.
Another argument is that different macro-definitions are required to distinguish the use of a parameter in a register or memory location. Finally, it is argued that the load and store instruction violates the prohibition on embedding the address mode within the mnemonic. The use of a mnemonic-to-destination-to-source(s) sequence would in fact eliminate the need for separate load and store instructions.

Argument in favor of proposed mnemonic length

In one of the early drafts of this standard, the mnemonics were much shorter, and those who saw it agreed “it should be designed for the reader, not the writer.” The draft committee accepted this opinion, and it is reflected in the longer mnemonics in the present draft.

Argument against proposed mnemonic length

On the other hand, it has been suggested that some of the proposed mnemonics could be shortened without significant loss of comprehensibility, e.g., TEST to TST, SKIP to SKP, CALL to CAL, and PUSH to PSH. Of these, only CALL appears controversial.

The IEEE Microprocessor Assembly Language Standard Committee solicits potential users’ opinions, including the reasons for them, on these two issues. Please forward your comments to Wayne P. Fischer, Kylux, Inc., 420 Bernardo Avenue, Mountain View, CA 94043.

Airborne computer terminals

A. B. Bell
University of Pennsylvania

One solution to the air traffic control problems at major airports would be the installation of computer terminals in passenger planes. Airborne terminals would eliminate the need for all but one human traffic controller at any airport. And they would provide a new line of passenger services.

Human airport traffic controllers are in short supply. They work excessively long hours and suffer frequent illness caused by the nervous strain of managing high-density traffic. A system consisting of airborne terminals linked to an airport computer could handle most of the duties of human controllers, saving the high cost of salaries and fringe benefits. One human traffic controller would still be needed, as backup in case of computer foul-up, and to control small planes without computer terminals.

The airport computer would keep track of all planes; runways; lighting; weather; emergency equipment and crews; conditions at alternate airports; availability of maintenance, fuel, food, police and fire personnel, medical care, substitute flight personnel; and miscellaneous information.

The pilot would communicate with the computer on his terminal keyboard by means of a code replacing standard spoken sentences with one- or two-digit numbers, much as Western Union preselected birthday messages are transmitted as code numbers which identify the entire message. He could easily memorize the code for common messages, just as he memorizes his home address, telephone number, and birth date. The message he selected would be displayed on the screen, so he could tell if he made a mistake. He would find infrequently used messages listed in an alphabetical manual.

The airport computer would reply in a series of standard messages, which contain blanks filled in with numerical data, such as visibility, ceiling, speed, altitude, and so on. To guard against error, an error-detection code would be used. All conversations would be recorded with their time and date on magnetic tape.

One advantage of this system is that communication problems between foreign pilots and American traffic controllers, or vice versa, would be eliminated.

Because the airborne terminal can easily be carried or wheeled from the cockpit to the passenger compartment, it could also be used by passengers. They could entertain themselves by playing simulated games, like those played on home television sets. And they could continue their office work, retrieving data, storing new data, and making calculations. (For game-playing and simple calculating, it would be less expensive to use a microprocessor than the ground computer.)

Airlines would finance the airborne computer system by charging passengers a fee for use of the terminals and by a rebate or discount from the airport in return for the decreased costs of air traffic control.