Behavior Research Methods & Instrumentation
1975, Vol. 7 (2), 143-144

A remote assembler for PDP-8 programs

KENNETH W. HOADLEY and DAVID J. GETTY
Brown University, Providence, Rhode Island 02912

The general advantages of using large computers to assemble code for laboratory minicomputers are discussed. A PDP-8 assembler implemented under IBM OS/360 is presented, and its use is described.

The assemblers supported by many laboratory minicomputers used for real-time experiment control are severely limited in features, capacity, and speed. Often, however, a large time-shared computer is available, offering superior hardware and more powerful software. Using this large computer for the remote assembly of minicomputer source code not only frees the small machine for the process control applications for which it is well suited, but also permits the development of an assembler free from the limitations imposed by the small machine. PDP8ASM, a program to assemble DEC PDP-8 programs under IBM OS/360, illustrates this philosophy.

ADVANTAGES OF REMOTE ASSEMBLY

The typical large time-shared system contains a memory several orders of magnitude greater in capacity than that of the average small computer. Virtual memory systems can provide even more substantial storage capabilities. Coupled with a fast instruction cycle, a large memory makes possible an arbitrarily large symbol table, expanded error messages, numerous assembler pseudo-ops, and rapid assembly of source code.

The large computer is equally well suited for editing minicomputer programs. Increased storage capacity permits the rapid editing of large programs. Certain editor functions, particularly those involving large amounts of scanning (e.g., uniform substitution), are simply too slow on a small machine, if they are available at all. Other utilities, such as backup tape dumping and reproducing of source decks, are also available. Disk-resident program libraries may be maintained, and program listings can be produced on high-speed line printers too expensive for most laboratories. Remotely assembling minicomputer programs provides the researcher with access to all of these facilities. Furthermore, a time-shared system allows many laboratory users to perform these functions simultaneously, yet none needs to tie up the laboratory machine while doing so.

Special thanks are due John Crawford for his assistance in designing and programming the LR(1) recognizer. Requests for reprints should be addressed to David J. Getty, Department of Psychology, Brown University, Providence, Rhode Island 02912. It is anticipated that source code and complete documentation will be made available through DECUS.

Table 1
DEC Assembler Pseudo-Ops Implemented in PDP8ASM

DECIMAL	FLTG
DEVICE	I
DUBL	NOPUNCH
EJECT	OCTAL
ENPUNCH	PAGE
EXPUNGE	PAUSE
FIELD	TEXT
FILENAME	XLIST
FIXMRI	Z
FIXTAB	ZBLOCK

COMPATIBILITY OF PDP8ASM WITH EXISTING ASSEMBLERS

A PDP-8 assembler designed to meet these needs has been implemented under OS/360. It combines the features of four common DEC assemblers—PAL-III, MACRO-8, 4K PAL-D, and PAL-8 (see Table 1). Unlike some previous assemblers (Powers, 1967), it is largely upward compatible with these four DEC assemblers; therefore, few, if any, modifications need be made to most programs acceptable to one of the DEC assemblers.

The major differences between PDP8ASM and other assemblers result from inconsistencies between the DEC assemblers themselves, hardware differences between the 360 and the PDP-8, attempts to simplify the use of several pseudo-ops, and several advanced features which have not yet been implemented. Whenever an inconsistency was found between the two DEC assemblers, the implementation judged more convenient for the researcher was chosen. For instance, MACRO-8 recognizes memory reference instructions by their context, but the other three DEC assemblers use position or notations in the symbol table to distinguish them from operate microinstructions. The latter approach was followed, since it allows checking for syntactic errors not detected by MACRO-8.

The use of EBCDIC rather than ASCII input media necessitated a few minor changes, including the substitution of a vertical bar for the up arrow as the PAL-8 division operator. Another hardware difference, input from disk or cards rather than paper tapes, made the "PAUSE" pseudo-op meaningless, so it is ignored by PDP8ASM.

The "DUBL" and "FLTG" MACRO-8 pseudo-ops have been simplified to apply only to the first number following them. Nevertheless, the pseudo-ops can be easily coded in such a manner as to be acceptable to both assemblers. The floating-point conversion routines of PDP8ASM yield more accurate results than do those of MACRO-8; therefore, the octal code generated may not be identical.

Finally, the conditional assembly feature of PAL-8 (IFZERO, IFNZRO, IFDEF, IFNDEF), the macro facility of MACRO-8, and literals have not yet been implemented. Hopefully, at least the last of these will be added in the future.

IMPLEMENTATION

PDP8ASM has been coded primarily in PL/1, supplemented by a few assembly language subroutines. Its modular design readily permits expansion; indeed, the gradual development of the assembler over the past year required such an approach.

Two passes are made by the assembler. During the first pass, all user symbols are defined, and an intermediate work file is created. The PDP-8 program listing and octal code are produced in pass two. Any syntactic errors are flagged in this output listing, and, when the assembler is used interactively, error messages are displayed at the terminal as well. Also shown in the listing are address links optionally generated to resolve off-page references.

In both passes, expressions are evaluated by a table-driven LR(1) recognizer.[1] Hence, the assembler language syntax may be redefined merely by changes in a metalanguage expressing the grammar; the recognizer itself need not be changed. It is therefore likely that the PDP8ASM assembler could service machines other than the PDP-8. Utility programs are provided to convert syntactic productions to tables for the recognizer.

Because the system symbol table is read in at assembly time, a single copy of the assembler may be used with many different instruction sets. During assembly, symbols are stored in a binary tree containing both user and system symbols. An alphabetic cross-reference listing of identifiers is optionally included in the output listing. If desired, a new system symbol table file, in the correct format for a future assembly, may also be generated. A utility program orders the symbol table file in such a way as to insure a well balanced tree of system symbols.

SYSTEM REQUIREMENTS

PDP8ASM presently runs under OS/360 or a compatible operating system, such as the Cambridge Monitor System (CP-67/CMS). With minor modifications, it should work equally well on a 370 operating system, such as VM370/CMS or VS2. A 150K region is more than adequate for most programs, but additional memory obviously allows faster assembly of large programs. To date, PDP8ASM has been run mainly on a large virtual machine; system paging has therefore made unnecessary the assembly work file on disk.

Some type of communications link to the laboratory minicomputer is needed. A direct telecommunications link allows either conversion to a PDP-8 storage medium (disk, paper tape) or cross loading directly into PDP-8 memory. Alternatively, paper tape, magnetic tape, or card output may be produced at the 360, depending on available hardware, or the octal file may be retained on a 360 disk and transmitted later. The assembler is itself device independent in this respect; the desired output medium may be specified at execution time through instructions to the operating system. Whatever method is selected, a check sum produced by the assembler guards against transmission errors.

COST-BENEFIT COMPARISON

The assembler, in production use since December 1973, processes approximately 2,000 lines per CPU minute (including the time for an alphabetized cross-reference listing) on a 360/67 under CP-67/CMS. A slightly slower rate for small programs (less than 300 lines) results from the fixed overhead of opening and closing files and reading in the system symbol table. Clearly, the time required to assemble any particular program depends on the proportions of comments, identifiers, and pseudo-ops.

The assembler presently in use was compiled by the PL/1 (F) compiler. The use of the PL/1 optimizing compiler might well provide a significant increase in assembly speed; this compiler, unfortunately, was not available for comparison purposes. In particular, the many subroutine calls, especially external ones, could be executed more rapidly.

The cost is small, however, if the laboratory minicomputer is released for more experiment control. When, at the same time, improved editing and assembly capabilities can be achieved, the cost seems fully justified.

REFERENCE

Powers, Michael. A PDP-8 assembler for IBM 360/50 and above. DECUS Program Library, 1967, DECUS 8-124.

NOTE

1. PDP-8 assembly language expressions can be unambiguously evaluated in a single left-to-right scan, provided that the recognizer is equipped with a push-down stack and the ability to look ahead one syntactic token during expression evaluation.