HEWLETT PACKARD

Language Tools for Intel 8086/8088, 80C186EA/EB/EC/XL, 80C188EA/EB/EC/XL Processors for Embedded Software Development

Technical Data

Integrated, workstation-based tools for total development lifecycle support

HP offers a full line of compilers, assemblers, linkers, and librarians for the Intel 8086/80186 family of microprocessors and microcontrollers. Known as the **Advanced Cross Language System (AxLS)**, these tools are fully integrated with the **HP Embedded Debug Environment.** Used together, these tools provide a powerful, easy-to-use, software development environment that aids in the creation of on-time, defect-free, specification-compliant embedded software. This complete suite of development tools is offered on both HP and Sun Microsystems workstations.

Each product offers the user a different way to perform measurements of both the target hardware and software. Since all components of the **Debug Environment** are based on OSF/Motif* and X-Windows, they share a common look and feel, minimizing the time you spend learning how to use each application.

These tools provide a wide spectrum of easy-to-use measurement capabilities ranging from true real-time, nonintrusive timing measurements to high-level, source code/assembly references.

The **Advanced Cross Language System** consists of an optimizing C compiler, assembler, linker, and librarian. These tools are tightly integrated with the rest of the HP embedded development solution. For example, the language tools generate all necessary C and assembly symbolic information for the debuggers and emulators/analyzers.

*OSF/Motif is a trademark of the Open Software Foundation.

The **HP Embedded Debug Environment** is a suite of tools that address the specific tasks of embedded systems design, from emulation and emulation bus analysis to high-level debug of C and C++ to CASE tool integration. These tools offer many ways to look at both your target system hardware and software, which assists in solving the toughest design problems. The debug environment also operates within the optional HP SoftBench environment, which brings advanced CASE tools and techniques to the realm of embedded software design.

The **Advanced Cross Language System** products are completely supported by the **HP Debug Environment**. In combination, these tools will assist you in tackling the toughest, real-time, embedded software development problems.

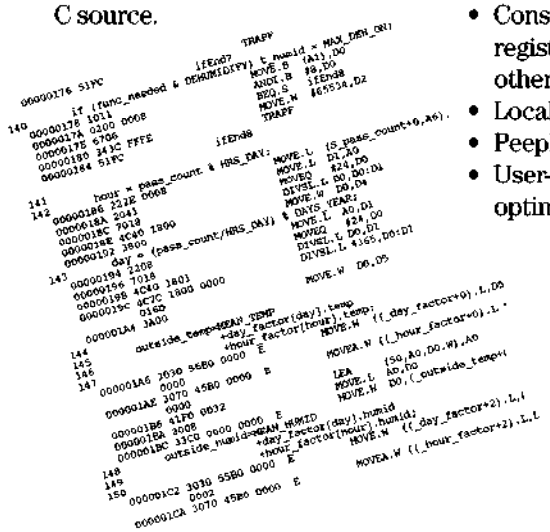
ANALYZE/SPECIFY DESIGN CONSTRUCT SOFTWARE INTEGRATE SYSTEM INTEGRATE & TUNE RELEASE/MAINTAIN

Advanced C Cross Compiler

Description

The HP Advanced C Cross Compiler is an optimizing compiler based on innovative technology that takes maximal advantage of the 8086/80186 instruction set and address modes. Highly space- and time-efficient code is generated at very competitive compile speeds with flexible optimization options. The quality and reliability of this HP compiler have been ensured through a comprehensive process emphasizing object-oriented design and exhaustive testing.

The compiler uses the HP B1449A Assembler/Linker and provides support for the emulation subsystems for the 8086, 8088, 80186/188, 80C186/C188, 80C186EA/EB/EC/XL, and 80C188EA/EB/EC/XL microprocessors and microcontrollers. Customized configuration files for the emulation systems provide immediate access to emulation and simulated I/O. The C Cross compiler offers features developed especially for designers of embedded systems that are generally unavailable in high-level languages. In particular, you may choose the memory model for calling functions and accessing static data, specify interrupt handler functions, and embed assembly language in the C source.



Features

Standards

- ANSI standard C compiler
- ANSI standard C preprocessor
- Complete C support and math libraries from ANSI standard for nonhosted environments
- Standard UNIX command-line interface

Embedded system

- Fully integrated with the HP Embedded Debug Environment
- Option to separate initialized and uninitialized data into separate segments
- Pragma for embedding assembly code in C source
- Pragma for named segment specification in C source
- User-specifiable choice of multiple memory models for function calls and static data access
- Listings with generated assembly language intermixed with C source and cross reference
- Fully reentrant code
- Options for uninitialized static data and inhibition of I/O
- Option for run-time checking of array bounds and NULL pointer dereferences
- Option to copy initial value data from ROM to RAM at "load time"

Optimization

- Constant folding, automatic register variable selection, and other global optimizations
- Locally optimal code
- Peephole optimizer
- User-selectable time versus space optimization

Generation of highly optimal and reliable code for embedded designs

Emulation

- Library providing system file I/O functions using HP's emulation systems' simulated I/O facility
- Full symbol information and C source-line numbers provided to emulation and analysis

Standards ensure portability and familiarity

The language accepted is ANSI standard C. This is a superset of "K&R C" and is the most rigorous C language definition currently available. This helps ensure portability of users' C sources. The ANSI C preprocessor provides several major additions to the standard UNIX C preprocessor including the "stringization" of a macro parameter, recursive function handling, macro parameter concatenation, and **#elif** preprocessing directive. The ANSI standard **#pragma** directives are used to support cross compiler options.

*UNIX is a registered trade-mark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

Sample List of Support Library Functions

string.h		*math.h*	*stdlib.h*		*stdio.h*	
strcat	*acos*	*log*	*abs*	*malloc*	*fclose*	*fwrite*
strchr	*asin*	*log10*	*atexit*	*qsort*	*fflush*	*getc*
strcmp	*atan*	*modf*	*atof*	*rand*	*fgetc*	*getchar*
strcpy	*atan2*	*pow*	*atoi*	*realloc*	*fgets*	*gets*
strcspn	*cos*	*sinh*	*atol*	*srand*	*fopen*	*printf*
strlen	*cosh*	*tanh*	*bsearch*	*strtod*	*fprintf*	*putc*
strncat	*exp*		*calloc*	*strtol*	*fputc*	*putchar*
strncmp	*fabs*		*div*	*strtoul*	*fputs*	*puts*
strpbrk	*floor*		*exit*		*fread*	*remove*
strrchr	*fmod*		*free*		*freopen*	*rewind*
strspn	*frexp*		*labs*		*fscanf*	*scanf*
strtok	*ldexp*		*ldiv*		*fseek*	*ungetc*

Figure 1. C support library functions and math library functions are ANSI standard and include I/O, string manipulation, and math functions.

The cross compiler command-line interface is patterned after the native HP-UX C compiler. That is, the preprocessor, compiler body, peephole optimizer, assembler, lister, and linker are all invoked with a single command. This eases *makefile* construction and provides a tool familiar to UNIX users.

The C support library functions and math library functions are ANSI standard and include familiar I/O, string manipulation, and math functions (figure 1). Routines involving real numbers implement the IEEE P754 real number representations.

Embedded system features

HP Advanced C Cross Compilers incorporate many features unique to microprocessor development. The ANSI standard C **const** type modifier causes data to be located in a named constant segment. Similarly, the **volatile** type modifier is appropriate for declarations of I/O ports in the target system (this overrides optimizations which might otherwise "optimize out" reads from or writes to the I/O port).

ANSI C's mechanism for options in the C source, **#pragma** directives, are used to provide several cross-development features:

#pragma ASM allows you to embed Intel 8086/80186 assembly language statements anywhere that a C statement could be placed. This is convenient for inserting otherwise inaccessible 8086/80186 instructions without resorting to a function call, or for implementing a routine entirely in assembly language but surrounding it with a C interface.

#pragma SEGMENT allows naming of program, data, and constant output segments for convenient grouping and link-time specification of load addresses.

#pragma INTERRUPT specifies that a C function is an interrupt handler. This causes the compiler to buffer all registers and generate an IRET instruction rather than an RET at function exit.

Special command-line options allow for specification of memory models, uninitialized static data, and no I/O. The memory model option permits selection of either the small or large model. The small memory model allows one code segment and one data/stack segment. In this model, function calls are considered to be "NEAR," and 16-bit pointers are used to access static data.

The large memory model permits one or more code segments, one stack segment, and one or more static data segments. In the large model, 32-bit pointers are used for static data accesses, and function calls are considered to be "FAR" except for the user-specifiable option to access static function as "NEAR." An option is provided to accommodate the typical embedded system circumstance of static data being uninitialized (in its absence such data is initialized to zero). Finally, the "no I/O" option caters to an embedded system by avoiding the overhead of unused I/O routines.

In addition, the C language allows for "prior to execution" initialization of static variables, the program's start-up code can optionally invoke a run-time routine, **_initdata()**, to copy initial value data from ROM to RAM for these variables.

The compiler produces C source listings with intermixed assembly code, expansion of include files, and C cross reference. All assembly code uses Intel syntax and symbol names incorporating the user's C symbol names. This makes referencing C symbols from embedded assembly easy and aids the readability of intermixed listings.

All code generated by the compiler is reentrant to ensure that interrupt functions written in C work properly. Small and large memory model versions of libraries (run-time, support, and math) have been provided.

Suite	*No. Lines*	*Execution*	*Coverage*
1	*61,675*	*Y*	*95%*
2	*87,200*	*Y*	*90%*
3	*346,300*	*Y*	*84%*
4	*251,500*	*Y*	*80%*

Figure 2. Summary of the four test suites. The Coverage column refers to the percentage of branches in the compiler covered by the suite as measured by a branch flow analysis tool. The Execution column indicates whether or not the suite's tests are executed on the HP 8086 emulator. Many of these tests are run multiple times with variations on memory models, optimization, and debug options.

Investment in total quality

Optimizing compilers are complex software programs. Hewlett-Packard has made a major investment to ensure the reliability of our advanced C cross compiler. Conscientious focus on highest quality was maintained throughout compiler development. Modular object-oriented design was used to "design in" quality. Code reviews were used to inspect for errors in logic before testing.

Four independent test suites were used to validate quality (figure 2). The first test suite contains "white-box" tests — it was written with a knowledge of the compiler's internal data structures and program flow. The other test suites contain "black-box" tests designed independently from the compiler. The second test suite is tailored to the cross compiler and tests options and pragmas in all combinations. The third is a test suite developed by HP for the host C compiler for the HP Precision Architecture (PA-RISC) family of computers. The fourth is the Plum-Hall C test suite which tests conformance with the ANSI C standard.

Automatic and discretionary optimizations

The HP Advanced C Cross Compiler places great emphasis on generation of optimal code. Compile-time inexpensive optimizations are always performed. The first of these is the automatic assignment of local variables or addresses of static variables to the register variable even when no **register** declaration appears in the C source. Another automatic optimization is constant folding; that is, arithmetic on compile-time constants is performed at compile time rather than run time. Expression simplification replaces a specified operation with an equivalent, less expensive operation (such as replacing a multiplication by a power of two with an equivalent shift). Looping constructs are optimized to remove an initial test if this can be recognized as unnecessary. Switch statements are optimized to a hybrid binary/linear search whenever the jump table would be less than 20 percent full.

Compile-time expensive optimizations are performed only in the presence of a command-line option. You may specify time or space optimization. The result of these optimizations is locally optimal code making extensive use of every 8086/80186 address mode.

Finally, a separate pass over the assembly code is made performing classic "peephole" optimizations. These include branch shortening where conditional and unconditional branches are reduced to the minimal size; tail merging in which two common code sections are merged to form one; redundant register load elimination, avoiding a reload of a register with a value already in any register; branch chain elimination, removing jumps to jumps; unreachable code elimination; and elimination of redundant array scale calculations.

Assembler/Linker/Librarian

Description

The cross assembler consists of an assembler, a macro string preprocessor, a linking loader, and an object module librarian. HP 64000 absolute and symbol formats may optionally be produced for downloading to HP emulators and analyzers.

The relocatable macro assembler for the Intel 8086/80186 family of microprocessors translates symbolic machine instructions into object code for execution by an 8086/88, 80186/188, 80C186/C188, 80C186/EA/EB/EC/XL, or 80C188EA/EB/EC/XL microprocessor. The assembler also translates instructions specific to the 8087 floating-point coprocessor into machine-readable code for coprocessor execution.

Object code is produced in a relocatable format by the assembler. Multiple relocatable modules are then linked into a single absolute module by the linking loader. When an absolute load is performed, relocatable addresses are transformed into absolute addresses, external references between modules are resolved, and the final absolute symbol value is substituted for each external symbol reference. The loader allows the specification of the program segment addresses, external definitions, and assign the final load address and segment loading order.

The linking loader also contains an incremental linking feature. The linking loader can combine multiple relocatable object modules into a single relocatable object module suitable for relinking with other modules. The linking loader produces an output object module file in either HP-OMF86 absolute format, HP 64000 absolute format, or Intel hexadecimal format. Global symbols, local symbols, or line number and module information can be included in the output file for symbolic debugging.

The linking loader provides the ability to load object modules from a library. The librarian is used to create such a library. The loader will include only those modules from a library that are necessary to resolve external references.

Features

Assembler

- Code generation for complete Intel 8086/88 and 80186/188 instruction set
- Support for Intel 8087 floating-point coprocessor instruction set
- Powerful, string-oriented macro preprocessor
- Repeated definitions of the same or of different code, data, and constant segments within a single source file
- High-level-language-like data structures
- Symbolic memory references via symbol name
- Flexible set of assembly control statements (conditional assembly, structured control, listing and output control)
- Detailed, well-documented error messages
- Extensive program listing capabilities including symbol table/cross reference information
- UNIX tailored command-line interface

Code generation for complete microprocessor instruction set and addressing modes

Linker

- Incremental linking (produces relocatable object modules for later relinking)
- Generation of HP-OMF86 absolute format, HP 64000 absolute format, and Intel hexadecimal object file format absolute
- Independent specification of all relocatable segments load addresses
- Specification of relocatable segment loading order
- Segment groupings into either GROUP or CLASS
- User-generated libraries (librarian)
- Definition of and modifications to external symbols at load time
- Extensive listing capabilities including cross reference and memory map information
- Detailed, well-documented error messages
- UNIX tailored command-line interface

Integration of Advanced Cross Language System with the HP Debug Environment

The Advanced Cross Language System includes many features that work in conjunction with the HP 64762G, 64763G, 64764G, 64764H, 64765J, and 64767A/B/C emulation subsystems for the 8086/80186 family. The default linker command file links in the emulation monitor and locates program, data, and constant segments at addresses coordinated with the provided default emulation configuration file. These files provide a working environment familiar to application writers on UNIX (K&R's "hello world" program works as expected) and they also provide a model on which a user may expand.

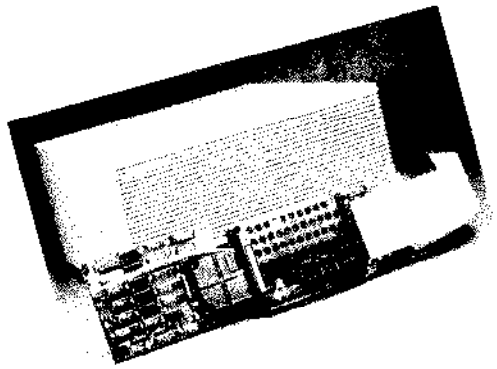
Also included with the compiler is an environment-dependent library providing the "system" functions required by the C support libraries. All sources for this library tailored to the HP 8086/80186 family emulators are provided. The functions in this library include I/O routines, such as open, close, read, write, and unlink; a utility to display messages on the emulation status line; and the system function for returning dynamically allocated memory.

Debug Environment Component	Function
HP Emulator/Analyzer	Controls HP 64700 series emulator for setting breakpoints, probing target memory, analyzing traces using target hardware, and debugging.
HP Debugger/Emulator	Provides the ability to perform real-time, in-circuit debugging of target code using an emulator.
HP Debugger/Simulator	Provides C and C++ debug capability without having to connect to an emulator. Assists in debugging target code before the target hardware is ready.
HP Branch Validator	Assists in testing your code by providing branch coverage analysis to insure the highest quality software.
HP Software Performance Analyzer	Assists in meeting code performance objectives by providing real-time measurement of code execution.

All assembly language symbols (typically C symbols with a prepended underscore) are available in emulation, plus C source-line number information. This allows you, for example, to trace on a function name or C source-line number. It also allows display of trace information with intermixed C source lines providing a real-time "intermixed listing" similar to that produced at compile time.

The compiler also has an option for performing run-time checking of array accesses for bounds violations as well as checking for NULL pointer dereferences. This mechanism is coupled with the emulation monitor's status line display to provide immediate run-time error message feedback. In this "debug mode," the compiler ensures the association of each assembly language label with unique executed code by generating a NOP machine instruction at each such label. This eliminates the inability to trace on, for example, the beginning of an outermost loop whose initial code is shared with that of an inner loop.

The compiler generates symbol information including function entry, exit symbols, and line numbers. This information can be used with all HP emulators for the Intel 8086/80186 family processors and with the HP 64708A Software Performance Analyzer.



Ordering Information

Software is licensed on a per-user basis. One license must be purchased for each, concurrent user of the software. Manuals and software media are purchased separately. These software tools support Intel 8086/8088, 80186/188, 80C186/188, 80C186EA/EB/EC/XL, 80C188EA/EB/EC/XL microprocessors.

B1493A 8086/186 Advanced ANSI C Cross Compiler

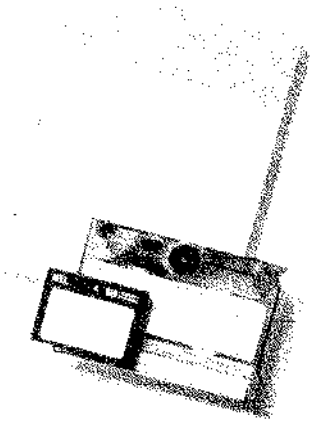
Opt AAX	HP 9000 Series 300/400 Media (DC-600 1/4" tape) and Documentation
Opt AAH	HP 9000 Series 300/400 Media (DAT) and Documentation
Opt AAY	HP 9000 Series 700 Media (DAT) and Documentation
Opt AAV	SUN SPARC Media (1/4" tape) and Documentation
Opt UBX	HP 9000 Series 300/400 Single User License
Opt UBY	HP 9000 Series 700 Single User License
Opt UBK	SUN SPARC Single User License
Opt UDY	IBM Single User License
Opt AJ4	IBM 3-1/2" Media and Documentation
Opt AJ5	IBM 5-1/4" Media and Documentation

B1449A 8086/186 Assembler/Linker/Librarian

Opt AAX	HP 9000 Series 300/400 Media (DC-600 1/4" tape) and Documentation
Opt AAH	HP 9000 Series 300/400 Media (DAT) and Documentation
Opt AAY	HP 9000 Series 700 Media (DAT) and Documentation
Opt AAV	SUN SPARC Media (1/4" tape) and Documentation
Opt UBX	HP 9000 Series 300/400 Single User License
Opt UBY	HP 9000 Series 700 Single User License
Opt UBK	SUN SPARC Single User License
Opt UDY	IBM Single User License
Opt AJ4	IBM 3-1/2" Media and Documentation
Opt AJ5	IBM 5-1/4" Media and Documentation

Complete software development bundles that combine these products with those of the HP Embedded Debug Environment are also available. Contact your HP 64000 field engineer for details.

Please call your HP 64000 field engineer for configuration information, supported emulators, and latest software options.



Data subject to change.
Printed in U.S.A. 8/93
5091-8671E