Pascal-P on the HP-3000

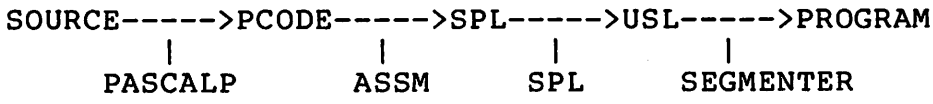
Bob Fraley
HP Labs

January 17, 1980

The Pascal-P compiler, available in the user contributed library, is a derivative of the Pascal-P compiler developed in Switzerland at ETH. The system compiles to HP3000 machine language, rather than being an interpretive system. The Pascal-P system has been quite reliable, although somewhat inefficient. This system is available in the HP User Group's contributed library.

System Structure

The Pascal-P/3000 system consists of three major components: the P-compiler, the P-code assembler, and the run-time library. The P-compiler produces P-code intermediate, which is converted by the assembler ASSM to SPL source. The SPL compiler may be used to produce a relocatable module:



This sequence of commands may be performed using a UDC (User-defined command) file, or using the program PPREP which was provided with some releases of Pascal-P. As the segmenter produces a program file from the relocatable SPL output, it adds the Pascal runtime library as an extra segment.

The Pascal P-compiler is a modification of the compiler supplied by Ammann, Nori, and Jacobi of ETH in Zurich. The principal changes which have been made are:

1. Correction of errors.

2. Adding standard Pascal features not in the original compiler, such as string constants longer than 10 characters, most of the I/O facilities, several built-in types and constants, and some portions of record variants.

3. Language extensions, including direct access files, a source inclusion facility, sub-array access, external procedure access, and separate compilation.

The compiler implements most of the features of Pascal, as defined by the book "Pascal: User Manual and Report" by Jensen and Wirth. Most of the features which have been omitted are difficult or time consuming to add, due to the structure of the compiler or the deficiencies of P-code.

The ASSM program translates P-code into SPL source. It does not translate directly to a USL file format primarily because of the lack of reliable, pre-coded, documented routines for generating USL files. ASSM is a rather simple translator, doing an instruction by instruction translation. The resulting code has rather poor quality, but is faster than an interpretive implementation. ASSM relieves the programmer from the necessity of dividing the program into segments. It segments the program based on the number of P-code instructions which it processes. In spite of the arbitrary segmentation, many programs execute without thrashing.

The Pascal run-time library establishes the Pascal environment and provides the I/O interface routines. The file support allows communication with ASCII files as Pascal TEXT files, and also supports Pascal internal file formats. Formatting routines read and write integers, real numbers, and characters. Booleans and strings may also be written. Special care is taken to allow the use of variable record files in all cases except direct access.

A number of submitted modifications have not been included in the distributed compiler, primarily due to a lack of time. One of the important characteristics of this system is its reliability; each change requires additional testing to be certain that the compiler still works correctly.

Error Messages

The Pascal system produces two types of error messages. Compilation messages occur during the translation of a program, indicating deviations of the source code from the implemented Pascal language. Run-time messages occur during execution when correct results cannot be obtained or bad values are used in operations.

Compile time errors are noted in the source listing as follows:

```
****                    ^104 ^59    ^104,59
```

The pointers "^" point to the token in the line above which was being examined when the error was detected, and the number is the number of the error which occurred. The error numbers are defined in Jensen and Wirth, page 119 [1]. Some error numbers unique to the HP 3000 implementation are given in the manual which is distributed with the system.

Run-time errors are divided into range errors and file errors. When a bound is exceeded or a bad pointer used, the Pascal error message is accompanied by the P-code location where the error occurred. This indicates the source line in which the error occurred. Unfortunately, in the current implementation, there is no indication of which compilation is being referenced when separately compiled procedures are used.

File errors are reported in a different manner. Unfortunately, the P-code location is not available at the time that file errors occur. The error is identified instead by the name of the file which was being used. This seems adequate for most error situations.

The Pascal system allows five run-time errors to occur before stopping execution. There is a parameter in the DEC file which may be changed to control this number. Of course, some bounds errors will cause addressing errors which MPE catches, causing the run to be aborted anyway. But in many situations the error is less important, and continuing execution will enable testing to proceed. The error limit in the DEC file may be changed to fit the needs of a specific installation. An individual user may only change the DEC file by making a local copy of the UDC file and changing the file associated with ASSMDECL.

Separate Compilation

One of the major extensions to the Pascal compiler has been the ability to divide a program into pieces which may be compiled separately. Each compilation contains procedures which may also be referenced from other compilations. One of the compilation units must contain the main program. The other units look identical except that the main program is omitted. The procedures which are defined in one compilation may be referenced from another.

```
Program P1;          Program P2;          Program P3;
  ...                  ...                  ...
Procedure A;         Procedure C;         Procedure G;
  ...                  ...                  ...
Procedure B;         Procedure D;         .
  ...                  ...
Begin                Procedure E;
  ...                  ...
End.                 .
```

In theory, each of the three compilations above can reference any of the procedures shown. In practice, however, each of the procedures used within any compilation must be declared, even if they aren't defined in the same compilation. For example, if the main program calls procedures D and G, the declarations

```
Procedure D;  External;
Procedure G;  External;
```

would have to be included in the main program. If D and G have parameters, the parameters and their types must be included in the external declarations like any other procedure declaration.
Separate compilation requires a modification of the command sequence. The compilation is the same through the SPL compilation. USL files from each of the compilation units are combined later to produce the program file. The UDC command PASCALP may be used with the appropriate parameter to produce this modified command sequence. (PASCALP provides this parameter to ASSM for generating the appropriate SPL commands.)

After all of the USL files have been created, the PASCJOIN command may be used to combine the files into a single program file. The BUILD program used by PASCJOIN may also be used separately to consolidate several USL files into a single USL file. The standard practice of placing many relocatable programs into a single USL file is not used due to the generated segment names.

Interfacing Other Languages

Pascal programs may be used to call programs written in other languages. The facility for doing this is quite primitive, however, so care should be taken. The difficulties are caused by Pascal's ignorance of optional parameters, by the data formats used by Pascal, and by the requirements that data types be provided for all parameters of the procedure which is being called.

The first step in calling an external procedure is writing a Pascal "Intrinsic" declaration for the procedure. Here are some examples:

```
Function Mail (Pin: integer; Var count: integer)
                  : Integer;  Intrinsic;
Procedure Quit (Num: Integer);  Intrinsic;
```

Reference and array parameters should specify "Var" in the Pascal declaration. The word "Intrinsic" specifies that the called routine is not written in Pascal; it does not necessarily refer to an MPE intrinsic. Procedures written in other languages, whose calling sequence is equivalent to that of SPL, may be called directly from Pascal.

The catch, however, is producing the proper Pascal declaration for the procedure. Pascal does not support double word integers, and at this time does not pack characters two to a word within character strings. Some parameters might not be describable as a single Pascal type. Optional parameters may not be defined in Pascal. The programmer must use some ingenuity to solve these problems, or write interfacing procedures in SPL.

One problem which should be solved in the near future is reference to strings. Pascal will soon implement packed character arrays, where two characters are packed per word as in other HP3000 languages. The address of such an array will be a byte address when the array is passed as a reference parameter. This will simplify the use of Intrinsic procedures.
A second problem is the classification of parameters according to Pascal types. Pascal's type checking may be bypassed by specifying the type "undefined". This may only be used as a "Var" parameter to an "Intrinsic" procedure. It allows any type of actual parameter to be used in the call.

A very ugly problem is caused by intrinsic functions which return a double word integer. Since Pascal does not allow arbitrary types to be returned from a procedure, the result type must be a built-in type. The procedure may be declared to have a "Real" result, which is then unpacked. The following excerpt illustrates this technique with the Clock intrinsic:

```
Var  Cvt :  Record case integer of
                 1: (R: real);
                 2: (Hi, Low: Integer);
               End;
     Hours, Minutes, Seconds: Integer;
Function Clock: Real;  Intrinsic;
Begin
   Cvt.r := Clock;
   Hours := Cvt.Hi div 256;
   Minutes := Cvt.Hi mod 256;
   Seconds := Cvt.Low div 256;
End.
```

Pascal files may not be passed directly to MPE intrinsics. The first word of a Pascal file, after it has been opened, contains the Filenum parameter needed by the file intrinsics of MPE. This may only be accessed by reference. A user wishing to access this value would need to write a procedure which returns this value as an integer, then use the integer for the intrinsic parameter. Note that the Filenum is only present after the file has been opened.

Conclusion

The Pascal-P compiler for the 3000 has a number of quirks which makes it awkward for advanced usage. Generally, however, it is easy to use and quite reliable. Pascal is a useful addition to the HP 3000.

Bibliography

[1] Jensen and Wirth "Pascal User Manual and Report" Springer-Verlag, New York, 1976.