Recovering Software with the SIMH Simulator

J. David Bryan, 12-Mar-2020

Old software and data, typically archived on half-inch magnetic tape, can be successfully recovered onto modern platforms by writing custom decoding programs. However, it can be much simpler to use the original archiving programs on the original operating systems running on a simulated machine. This paper presents two such recoveries: a simple case where the original archiving program was still available, and a more complex case where the required software no longer existed.

Background

Recovery of information that had once been stored on computer systems that have been decommissioned may be desirable for many reasons. Data might be needed to confirm an historical event or to obtain information that had been printed but subsequently lost. Old programs might be examined for academic interest, or to reuse already developed algorithms in new programs.

Old systems were frequently backed up onto half-inch reel-to-reel magnetic tape. In many cases, these tapes have outlived the systems on which they were made. In some instances, the tapes were intentionally kept and carefully preserved. In others, individual computer operators and system managers kept private copies offsite as a hedge against catastrophic loss.

Although tapes could have been written in a "bare" format, i.e., as a simple set of records per file, this was rarely done. Instead, any one of a plethora of proprietary tape archiving programs was used, generally to ensure that a file directory was written to the tape, and to ensure efficient use of the tape by packing the data into a smaller number of large records. File access control information, such as passwords, was also included in the archive to ensure fidelity when restoring.

Recovering a file from one of these old tapes requires two steps. First, a binary image of the tape must be transferred to a modern platform, such as a PC. Second, the image must be decoded to extract the file or files desired into files compatible with the host platform.

Surprisingly, the first step is usually simpler than the second. Half-inch tape drives were ruggedly built, had long service lives, and are relatively easy to obtain on the used market. Several solutions exist for connecting an old drive to a modern PC. An example would be to connect a Hewlett-Packard model 88780 tape drive (circa 1990) via a SCSI interface to a PC running the Linux operating system. PC software to read the tape and create a binary image is readily available; an example is the *tapeutils* package.

Once the tape image is present on the PC, though, extracting the desired file data may be difficult. Tape archive formats were often vendor-specific and proprietary, and a given vendor often supported several different and incompatible formats that were

incompletely documented. The operating manuals of the time that may have documented the tape formats often are no longer available. Even if the format can be identified and documentation found, a host-PC program would still have to be written to extract the desired files into a host-compatible format.

As an example of the magnitude of the problem, the Computer History Museum offers the *Hewlett-Packard 1000 Software Collections*. The HP 1000 was a series of 16-bit minicomputers employing up to one megaword of memory that were sold from 1974 until 1999. The CHM arranged with HP to host all of their software for the 1000 and make it available for non-commercial use.

The collection takes the form of tape images, archived in one of eleven different and mostly incompatible forms. They are described here:

```
http://www.bitsavers.org/bits/HP/HP_1000_software_collection/
```

Recovering files from the images directly would require writing eleven host-PC programs, each of which would have to be tested to verify data integrity before relying on the recovered data.

A simpler solution to this step is to run the *original* archive programs on a simulated machine. SIMH, the Computer History Simulator, offers a large variety of CPU and device simulations running on contemporary PC platforms. In most cases, kits containing original period operating systems are available and can be run easily on a PC. By running the original OS and original archiving program, tape format compatibility issues are avoided, as is the need to write and test decoding programs on the host platform. Obtaining the recovered data is simply a matter of copying to a simulated peripheral device that uses plain text or binary as an output medium.

Case Study 1: Retrieving an Old Purchase Order

The owner of a piece of used capital equipment (a metalworking lathe) wanted to know the original vendor, purchase price, and ordered options. This information was on the original purchase order written by the company taking the factory delivery. That company's orders were kept on an HP 1000 system running the Real-Time Executive (RTE-IVB) operating system. Unfortunately, the system was decommissioned in 1995, and no hard copy of the order existed. However, the system operator kept the final system backup tapes taken before decommissioning.

The accounting system was written in-house in BASIC, and the purchase orders were stored in BASIC-formatted data files in a rather complicated multiple-linked-list structure. The format was documented, albeit in a text file stored on the system and hence stored on one of the backup tapes. Obtaining a copy of the order directly would involve first locating and extracting the purchase order system documentation, then locating and extracting the BASIC data file containing the order, searching the file for the specific order needed, and finally extracting the text of the order from the binary file data. This would require writing at least two custom programs.

Alternatively, the RTE system could be restored under simulation from the last backup tapes, and the original accounting program could be run to obtain a printout of the order directly. This approach turned out to be much simpler.

The physical backup tapes were first transferred to tape image files on a PC. Restoring the RTE operating system under simulation required two steps: restoring RTE itself from a disc image backup, and restoring the company data files from a separate file-by-file backup.

The operating system backup tape contained a track-by-track copy of the physical disc preceded by an "offline" restoration program in absolute binary format. The tape image was mounted on the simulated magnetic tape drive, a new, blank disc image was mounted on the simulated disc drive, and the magnetic tape boot loader was invoked to read the restoration program into memory. Running the program restored the disc image from tape to the simulated disc drive.

Once that was accomplished, the RTE system was bootstrapped directly from the simulated disc, and the "online" file restoration program was run to restore all of the data files from a second tape backup image.

With the system fully restored, obtaining the required purchase order was simple. The resident BASIC interpreter was used to run the accounting system, where the integral search function was used to locate the desired order. A text copy of the order was obtained by printing the order to the simulated line printer whose output medium is a plain-text file on the host PC.

While only one specific purchase order was needed, this approach made the full set of orders, inventory, etc. available with no additional effort. The majority of the time spent on recovering the information was in copying physical tapes to PC-hosted tape images. Restoring the images to a simulated system was easy and required little time.

The retrieved target of the exercise is reproduced on the next page.

Purchase order # 13193 Ordered 8/20/81 Due 9/20/81 Our job R&D

Purchased from
ZAGER MACHINERY CO. INC.
P.O. BOX 501
FOLCROFT, PA 19032-0501

Vendor contact is STEPHEN BECK Vendor phone number is 215-534-5440
Vendor's terms are NET 30 Vendor f.o.b. point is INDIANA

Shipping Instructions:

ITEM	ORIGNL	P.QUAN	SHIPPED	STK#	DESCRIPTION / PART NUMBER	PRICE	/ UNIT
1	1	1	10/05/81		South Bend 10in. Tool room Lathe, Model CL8187RB	6932.00	each
2	1	1	10/05/81		South Bend 1hp 208/3ph/60Hz CE2625P Motor	399.00	each

Comments: Item 1: 4-1/2 ft. Bed, 33 in. on centers.
Standard parts included with lathe: CL1545R taper attachment
CL2180L large face plate, CE1770L collet rack, and a
CL0968R mic. carriage stop. Machine was purchased without
a motor per JDB. Confirming to Ms. Erbe.

Case Study 2: Recovering HP 2000 Access Source Files

A project was undertaken to recover the source code for the Hewlett-Packard 2000 Access Time-Shared BASIC (hereafter, "Access") operating system. The goal was to obtain the assembly language source file set in a form that could be modified, reassembled into a binary Master Control Program tape, and loaded into a physical or simulated computer system.

Introduction

The Access operating system uses dual HP 2100, 21MX (1000 M-Series), or 21MX-E (1000 E-Series) processors to support up to 32 concurrent users. The primary CPU, designated as the System Processor ("SP"), runs the BASIC interpreter and provides disc and magnetic tape storage. The secondary CPU, designated as the I/O Processor ("IOP"), handles I/O through the terminal multiplexers and supports the line printer, paper tape punch, and paper tape reader. The system can be configured to use one of four system disc models.

HP provided the Access source code on magnetic tape. The *HP 2000 Computer System Sources and Listings Documentation* manual (HP part number 22687-90020) describes the format of the distribution tapes. The associated part numbers are:

- 22703-11001 — Listings on ½-inch tape at 1600 bpi PE
- 22703-11002 — Sources on ½-inch tape at 1600 bpi PE

Each tape is a "selective dump" of an Access user account: Z101 for the source files, and Z102 for the listing files. The files are BASIC data files containing strings, with one source or listing line per string. The file names are given in the above manual.

With the source tape, one could, presumably, recreate the binary files that were distributed for installation on a new Access system. The Master Control Program ("MCP") installation tape part numbers are:

- 22687-11001 for the 21MX or 21MX-E on ½-inch tape at 1600 bpi PE
- 22687-11002 for the 2100 on ½-inch tape at 1600 bpi PE

With the SIMH simulator for the HP 2100/1000, one could then explore the operation of the Access system and, if desired, modify it experimentally. The modified MCP tape image could also be written to a physical tape and loaded into a physical computer system. To reach this goal, two steps were needed:

- Assemble the set of source files into corresponding binary modules
- Write the binary modules to a magnetic tape image in the MCP format

Step 1: Assembling the Source Files

The first step was to obtain a set of source files that could be assembled into the identical files present on the original MCP installation tape. The final Access revision was date code 1812. The files on the source tape presumably reflected this. Unfortunately, this was not the case.

In attempting to reconstruct the source files that would assemble to the binaries distributed on the MCP revision 1812 tapes, several obstacles were encountered.

The first problem was that the Access system provided no assembler. Therefore, assembly of the sources would have to be performed on another system. Given that Access used both floating-point and special I/O Processor microcode, this meant using the RTE (Real-Time Executive) assembler, which supported both of these requirements. A means of transporting the sources to an RTE system in a format acceptable to the RTE assembler was needed.

The next problem was that there was no simple way of dumping the sources to a transport medium (e.g., paper tape or magnetic tape). To the Access system, the source files were simply BASIC data files containing strings. These strings would have to be read and "printed" to a tape punch or magnetic tape drive to obtain plain-text files that could be submitted to the assembler.

A simple BASIC program ("LISTER") was provided with the listings tape to copy a specified listing file to a given device. No comparable program was provided for the source files. Moreover, the listing files contained carriage-control characters in column one of each line, whereas the source files did not. A modified version of LISTER was therefore needed to print each source file to a simulated paper tape punch.

The resulting set of paper tape images was then loaded into individual File Manager ("FMGR") files on an RTE-IVB system. The files were named with the RTE convention by changing the leading "S" of each filename on the source tape to "&" (ampersand).

The choice of RTE system was made by selecting the latest system that still used the historic assembler ("ASMB"). The later and more capable RTE-6/VM system did not offer ASMB, replacing with a new macro assembler ("MACRO"). While MACRO could operate in an ASMB-compatible mode, it produced a different relocatable format (called the "new relocatable" format) that was incompatible with the Access IOP system generator program.

Immediately, a new problem arose. The 80-some source files were written in three different assembly dialects, reflecting the three different assemblers originally used by HP. Based on comments in the documentation manual and identifying marks in the corresponding listing files, their identities were:

- HP 24158A assembler running DOS-III
- HP 32223A cross-assembler for the HP 3000 running MPE

- HP 22594A cross-assembler for the IBM 360 running OS/360

The DOS assembler had been used for nine of the fourteen absolute assembly programs. The IBM cross-assembler had been used for one absolute program — the main program for the System Processor ("&TSB"). The 3000 cross-assembler had been used for the remaining absolute, as well as all of the relocatable assembly programs.

Some of the source files lacked the assembler directive (e.g., "ASMB,R,L") that is required as the first line of each file; these had to be added. Some files had the proper directive but preceded them by cross-assembler directives that had to be removed. Most files had sequence numbers in columns 72-80 that normally would not be a problem (they would be seen as comments), except in one case: a (sequence) number following the END pseudo-op is seen as a transfer address, which causes an assembly error. Again, these had to be removed (and, in fact, it was easier to remove all of the sequence numbers, as they served no purpose in the assembly).

The "&TSB" program required additional work. This source file contained 880 header-line pairs embedded within that had to be removed. More significantly, all quotation marks had been stripped, and lines were broken where the quotes had been. For example, the assembly-language line:

```
CBFLG EQU B20M      "BREAK" COMMAND GIVEN
```

...was rendered as:

```
CBFLG EQU B20M
REAK
COMMAND GIVEN
```

At each occurrence of a quotation mark, the mark and the following character were deleted, and a line break was inserted. In the example above, the leading quote and the B (of BREAK) were deleted, as was the trailing quote and the space before the word COMMAND. In most cases, these lines could be rejoined and reconstructed by inspection. However, some cases were not obvious. For these, inspection of corresponding sections of the existing source code for the HP 2000 F TSB system (the predecessor to Access) was used to infer the missing characters.

Once all of this editing was complete, the files were assembled. Some files assembled cleanly, but most reported multiple error codes. These were categorized as follows:

- ***OV*** (numeric operand overflow). These occurred on every RAM pseudo-op.
- ***M*** (illegal operand). These occurred on some uses of the =L expression literal and the EQU pseudo-op.
- ***SO*** (symbol table overflow). This occurred on the &TSB file.

The RAM pseudo-op is used to generate the firmware extension instructions used by the I/O Processor. According to page 4-24 of the *RTE-IV Assembler Reference Manual*

(92067-90003, October 1980), the operand to RAM "must evaluate to an absolute expression in the range 0 to 377 octal." This is because RAM adds its operand to the octal constant 105000, producing a machine instruction in the range 105000-105377.

Unfortunately, the IOP microcode instructions fall into the range 101400-101437 and 105400-105763. So a use such as:

```
RAM SAVE
```

...where SAVE is equated to 474 octal causes an assembler overflow error.

Page 2-6 of the assembler manual says that the operand to an =L literal is "an expression which, when evaluated, will result in an absolute value." The illegal operand error has many causes, but one of them is "a negative operand is used with an opcode other than ABS, DEX, DEC, OCT, and BYT." The errors occurred on uses that produced negative results, such as:

```
AND =L177777B-HLDTA
```

...where HLDTA is equated to 20B (the B suffix indicates an octal constant) or:

```
LDA =LDFLAG-STCC
```

...where label DFLAG occurs at a lower address than label STCC. The EQU references were also negative, such as:

```
BIT15 EQU 100000B
```

Changing the 60 or so files that had assembly errors to correct the above uses would have been prohibitively difficult. Moreover, the resulting sources would not result in binaries identical to those on the MCP tape. So instead, the source of the RTE assembler, available from the HP Software Collection, was modified to permit the semantics employed by the cross-assemblers.

The symbol table overflow was a more difficult issue. The RTE assembler constructs its symbol table in the free memory available in the partition in which the assembler is loaded. In RTE-IVB, the largest possible partition is 29 1K-word pages in size. This was insufficient for the symbol table needed by the TSB program that, at some 46000+ lines, is by far the largest source program. This is undoubtedly the same reason this one program was cross-assembled on an IBM 360 instead of on the HP 3000. As RTE-IVB does not offer virtual memory, the TSB program could not be assembled on the RTE-IVB system.

The RTE-6/VM assembler, MACRO, uses virtual memory, so symbol table space is not a problem, and indeed, the TSB program did assemble successfully using MACRO in its ASMB-compatible mode. However, the MACRO listing does not handle line numbers larger than 32767 correctly; it prints the line numbers as negative values (without the minus signs), so the line numbers decrease rather than increase. In addition, MACRO outputs absolute records up to 256 words in size, whereas ASMB records are limited to

120 words. This may or may not have caused a problem when loading the binary into a system but was cause for concern.

RTE-6/VM supports normal (i.e., non-virtual-memory) programs up to 32 pages in size. As a trial, the ASMB from RTE-IVB was linked and run on an RTE-6/VM system in a 32-page partition, and indeed, it did have enough space to assemble the TSB program. So the decision was made to use RTE-6/VM to assemble all of the Access sources using a modified ASMB that follows the cross-assembler semantics for RAM, =L, and EQU.

Almost all of the sources specify the "C" option in the assembler control statement that causes a symbol cross-reference table to be appended to the end of the assembler listing. This requires the "XREF" program from the RTE-IVB assembler package, and this program was linked as well on RTE-6/VM for a 32-page partition. However, even with the larger memory area available, a "TABLE OVERFLOW" error occurred when processing the TSB source file.

XREF offers a method of processing symbols in batches to overcome this error. When invoked by the assembler, it processes all symbols, sorts them alphabetically, and then appends them to the assembler listing, starting with the page number following the last assembler page. By invoking XREF manually, subranges of the symbol set may be specified, which limits the amount of memory required.

For the TSB source file, three subranges were needed, as shown in the invocation below (the invocation parameters are given on page I-2 of the assembler manual):

```
:RU,XREF,&TSB::TS,-,1,-1031
 /XREF: ENTER LIMITS <LH> OR </E> ?] H
 /XREF: ENTER LIMITS <LH> OR </E> ?]IR
 /XREF: ENTER LIMITS <LH> OR </E> ?]S_
 /XREF: ENTER LIMITS <LH> OR </E> ?]/E
 /XREF: $END  <*****>
```

The limits are the first and last leading character of each subrange, and the "-1031" parameter indicates that the assembly listing ended with page number 1031 and so XREF should start with the next page. This method produced a full cross-reference. However, XREF only reports line numbers up to 32767. As the TSB program is some 46000 lines long, some line number references appear as "#####" to indicate numeric overflow.

Once all of the sources assembled correctly, a check was needed to ensure that the resulting binaries matched those on the MCP tape. An examination of the tape revealed that 16 files were present. Each file except the first is preceded by an ID number. File 13 has ID number 26000, but there is no corresponding source file that produces a binary with that number! However, the omission was easily resolved: file SC79XX may be conditionally assembled either as ID 25000 or ID 26000. The assembler directive in the file specifies assembly as 25000. Therefore, SC79XX was copied to a new source file, "SC79X3", with a directive that specifies assembly as ID 26000.

Tape files 1, 2, and 5-15 each consisted of single absolute programs, while files 2 and 3 each consisted of multiple relocatable programs forming the IOP module set. All programs are preceded and optionally followed by absolute records that are used as file and group ID markers. These present no special problem for the absolute programs, as they are created by inclusions in the corresponding sources. However, they are not part of the relocatable programs and must be excluded from any comparisons.

Another problem arose here. It has been observed in previous reconstruction attempts that the various HP assemblers produce differing (though equivalent) binaries for the same source input. Consider, for example, a file containing one absolute record of 20 words loading at address 100 and another file containing two absolute records of 10 words each loading at addresses 100 and 110.

The files would not compare, although the CPU memory images from loading the two files would be identical. Similar problems exist with relocatable records. Therefore, comparing the newly assembled binary files with those on the MCP tape would not necessarily work.

To work around this, inverse assemblies were performed on the new object files, and the resulting listing were compared with inverse assemblies of the corresponding MCP tape files. If the listings were identical, then the source files would be correct.

Earlier work on reconstructing the HP 2000A sources showed that the *DOS-M ABSOLUTE OBJECT DECODER* (22415-80001 Revision B) is the only convenient inverse assembler for absolute binaries. This program runs under the DOS-III operating system, so the *DOS-M RELOCATABLE REVERSE ASSEMBLER* (22438-80001 Revision A) was selected to handle the relocatable files.

The 13 absolute files were copied from the MCP tape to a single paper tape image using the RTE file manager. Then the corresponding absolute binary files produced by assembling the corresponding sources were copied to a second paper tape image. These images were fed as input to the DOS-III absolute inverse assembler (program "DCODE"), and the resulting listings were compared. Discrepancies were noted in the following source files:

- ***C2883***: The source file was version 1705 while the MCP file was version 1624. The only difference was the constant declaration following label "38" — it was "36 DEC 36" in 1705 and "40 DEC 40" in 1624, but in both cases the constant is not used (i.e., is not referenced).
- ***C7900***: This file had the same issue as C2883 (they are in fact copies of each other, with C2883 specifying the "Z" assembler option and C7900 specifying the "N" option).
- ***C79X3***: This source file did not exist, although the MCP binary file did. It was created by copying C79XX and specifying the "Z" assembler option instead of the "N" option. A few other minor changes brought it into compliance with the MCP version.

- ***MCPRG***: The source file was version 1705 while the MCP file was version 1812. The only difference was the date code — "PATDC EQU 1705" in 1705 and "PATDC EQU 1812" in 1812.

The same process was used with the relocatable files, except that program "RBMSA" was used as the reverse-assembler. Discrepancies were noted in the following source files:

- ***D274E***: One entry in the EBCD conversion table was updated.
- ***D50CD***: The anticipated read length for a write/read call was increased from 1043 to 1199 bytes.
- ***LTC***: The original NAM record had a comment, which was ignored by the assembler used for the MCP tape. The RTE assembler includes NAM comments in the relocatable file, so the comment was removed.
- ***RPC***: A timer-running check was added for Revision 1812.
- ***SYNIH***: A parameter string length check was added, and a timeout bit check was corrected.

After making the above changes, the assembly/reverse-assembly/compare process was repeated until every module compared with the corresponding one on the original MCP installation tape. At that point, the set of source files exactly matched the final Access code distribution. The assembly process also produced a set of listing files that matched the distribution.

Step 2: Writing the New MCP Tape

The second step involved writing the various assembler output files to a new magnetic tape image in the format required. Fortunately, the MCP tape format was documented in the *HP 2000 Computer System Sources and Listings Documentation* manual.

From the manual, and from examining a dump of the original MCP installation tape, it was determined that the tape consists of 16 files. The first file is the master program, ***MCPRG***, and the second is the IOP configurator, ***IOPC***. Files 3 and 4 contain the IOP relocatable modules. File 5 contains the warm start program, ***WSPRG***. Files 6-10 contain loaders for the various disc drive models (***7900***, ***2883***, ***79X0***, ***79X3***, and ***79XX***). File 11 is the BASIC interpreter, ***TSB***. Files 12-15 are the file conversion utilities (***C79XX***, ***C79X3***, ***C7900***, and ***C2883***). File 16 is empty, except for an ID record.

Each of the files, except the first, is preceded by a five-word ID record, as is each of the relocatable modules collected in files 3 and 4. Each word is 16 bits, and the format is:

```
word 1: 001000 octal
word 2: 002001 octal
word 3: ID number
word 4: information number
word 5: checksum
```

ID numbers are assigned to specific modules and are in ascending order on the tape. These records are built-in to the absolute binary images but must be added externally to the relocatable images. With the aid of a small FORTRAN IV program that writes these ID records to tape, an RTE File Manager command file may be constructed that will create a new MCP tape from the assembler output files.

The validity of the new tape was verified by using it to generate a new Access system that was then run under simulation and confirmed operational.

Recovering the Access operating system software involved running three other operating systems (RTE-IVB, RTE-6/VM, and DOS-III) under simulation in order to use the tools available with those systems to aid recovery. Attempting to write the same set of tools for execution on a host PC would have required a prohibitive effort.

Summary

Recovering information from old computer systems may be performed either by writing, testing, and running custom programs for each retrieval item or by running the original systems under simulation. The latter is far easier, especially in complex scenarios. The availability of SIMH simulators for a broad range of hardware greatly simplifies the recovery process.