Chapter 1
Getting Started with PC/370

Objectives

Upon completion of this chapter you will be able to:
- Briefly describe the generations of programming languages,
- Install PC/370,
- Enter, assemble, link, and execute a simple program,
- Create datasets for use with PC/370, and
- Produce an unformatted list of the records in a file.

Introduction

PC/370 is a shareware product developed by Don Higgins which enables the user to write and execute System/370 (mainframe) assembler language programs on a personal computer. Prior to the development of PC/370, a mainframe computer was required in order to do so.

System/370 assembler language is often referred to as `BAL` for Basic Assembler Language. `BAL` is a second generation language. The first generation of languages was machine language. Machine language programs were written in binary code (usually represented by ones and zeroes.) For example, the machine language code necessary to move one field to another might be as follows:

```
110100100000001111010000101010111101000010100000
```

As you might imagine, such programs were very difficult to write and even more difficult to maintain. Assembler language was developed to overcome this problem. The idea was to assign a simple mnemonic code to each operation and its operands. For example, the first eight bits of the machine code shown above, `11010010`, indicates a character move operation, so `MVC` would be used in its place: `MVC` is, afterall, much easier to remember than `11010010`! Another program, called the assembler, would translate these mnemonics to their equivalent machine code, since ultimately, all programs must be at the machine code level in order for the computer to understand them.

The problem with coding in assembler is that it is still coding at the machine level, and each machine has its own language. Consequently, a machine language program written for one platform (such as a mainframe) cannot run on another platform (such as a PC). This was a part of the impetus for the next generation of languages, the third generation. This generation includes `FORTRAN`, `COBOL`, `BASIC`, `RPG`, and `C`. The intent is that a programmer would write source code which would largely be independent of the platform the code would run on. The source code would then be run through a compiler which would translate the source code to the machine code approriate for the target platform. Only the compiler (not the source code) would be machine dependent.

This discussion raises two questions. First, how can we run BAL on a PC, and second, why would we want to learn BAL? PC/370 enables us to run BAL on a PC because it emulates a mainframe environment. That this is possible should come as no surprise: portability is, afterall, a function of the translator rather than a function of the source code.

There are several reasons to learn BAL. First, there are many legacy applications in business written in BAL. It is simply not economical to rewrite these programs in another language. Instead, we maintain them, and we must know BAL in order to do so. Second, some applications are simply easier to write in BAL. As an information system professional, each programming language is another tool in your tool belt. As a professional, you should know which tool to use for which job. While COBOL, C, or even Lotus may be the best choice for some jobs, BAL is the language of choice for others. Third, since each language must ultimately be reduced to its machine language equivalent (again, this is the function of the compiler), and since BAL is closest to machine language, learning BAL will make you a better programmer regardless of the language you normally code in. Lastly, some of us would argue that it's fun!

For completeness, it should be mentioned that the fourth generation languages (commonly referred to as 4GLs) are non-procedural languages which support database queries and "ad hoc" reporting. These include SQL, RAMIS, FOCUS, NOMAD, and others.

Installation

The diskette included with this text has three directories: \FULL, \MIN, and \SOURCE. The \FULL directory contains the entire PC/370 package. (Most shareware authors require that their software be distributed in its entirety.) Much of this you will never use. To simplify your use of PC/370, the \MIN directory contains the minimum files needed to write and execute all of the programs contained in this text.

The programs can be stored on and executed from a floppy disk. If you intend to do this, then you should make a backup copy of the distribution diskette. This can be done with DOS' DISKCOPY command: from the C:\> prompt type `diskcopy a: a:` and press Enter. Swap disks as directed. Keep the distribution diskette in a safe place. (If you will be using a double density drive, you need only copy the \MIN directory from the distribution diskette to your work diskette.)

If you intend to store and execute your programs from a hard drive, then create a directory (such as \PC370) and use DOS' XCOPY command to copy all directories and files from the diskette to that directory. For example,

```
C:\>md pc370
C:\>cd pc370
C:\PC370>xcopy a:*.* /s
```

Throughout this text, it is assumed that PC/370 is being execute from the \MIN directory of a floppy disk.

The \SOURCE directory contains the source code for all of the examples contained in this book. You are encouraged to assemble and execute these programs. To do so, you must copy them (one at a time) to the \MIN directory. Experiment with them. Like anyother language, that's how you will really learn!

Our First PC/370 Program

The following program will display a message on the screen. Use your preferred editor to create HELLO.MLC as shown below, then follow the script on the following page to verify that PC/370 has been installed properly.

```
         PRINT NOGEN
*************************************************************
*        FILENAME:  HELLO.MLC                               *
*        AUTHOR  :  Bill Qualls                             *
*        SYSTEM  :  PC/370 R4.2                             *
*        REMARKS :  This program will display a message.    *
*************************************************************
         START 0                                          ↑
         REGS                                      Careful!
BEGIN    BEGIN                                  Don't go beyond
         WTO   MESSAGE                             column 71!
         RETURN
MESSAGE  DC    C'Hello world!'
         END   BEGIN
```

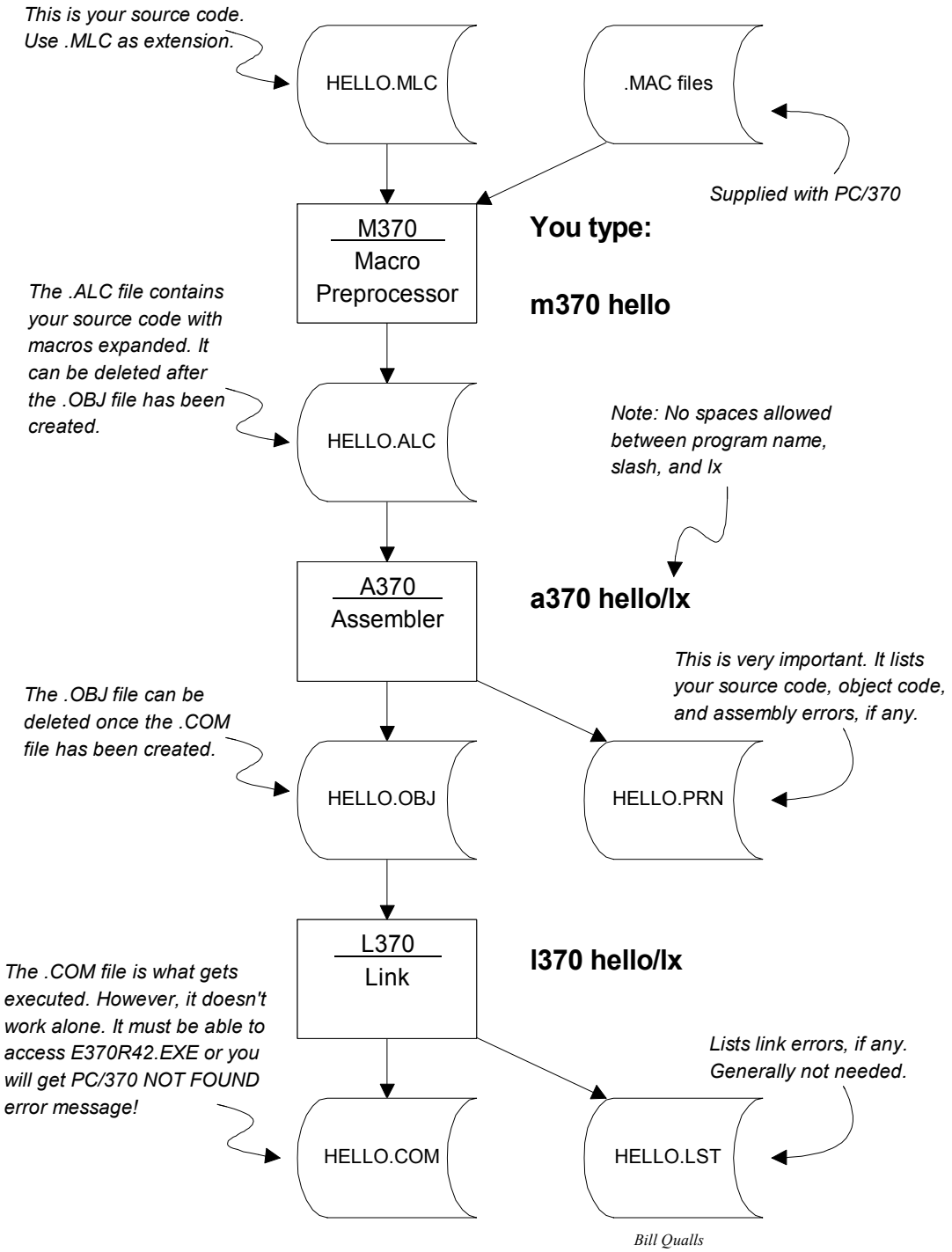
In keying this program observe the following:

- The program is created in the \MIN directory.
- An asterisk (*) in column one indicates the line is a comment.
- Labels (such as BEGIN and MESSAGE) always start in column one.
- There must be at least one blank between labels and operations: by convention, operations (such as PRINT) begin in column ten.
- There must be at least one blank between operations and operands: by convention, operands (such as NOGEN) begin in column sixteen.
- Labels, operations and operands must be in upper case; comments can be in upper case or lower case.
- All code must not extend beyond column 71.

The flowchart on page 5 shows the inputs and outputs for each step of the assembly process.

```
A:\>cd min

A:\MIN>m370 hello

A:\MIN>a370 hello/lx
 ***********************************************
 *    PC/370 System Release 4.2    01/07/88    *
 *    Copyright (C) 1988 Donald S. Higgins     *
 *                                             *
 * You are encouraged to copy and share this   *
 * package with other users on the condition   *
 * the package is not distributed in modified  *
 * form, and that no fee is charged.  If you   *
 * find PC/370 useful, send 45 dollars to the  *
 * address below to become registered user and *
 * support continued shareware development.    *
 * Registered users will receive notices of    *
 * future PC/370 releases.                     *
 *                                             *
 *      Don Higgins                            *
 *      6365 - 32 Avenue, North                *
 *      St. Petersburg, Florida 33710          *
 ***********************************************
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
STATS SYM=00021  MAXSTD=00005  LIT=00000  MAXLTD=00000  BTMEM=51900
NO ERRORS FOUND

A:\MIN>l370 hello/lx
 ***********************************************
 *    PC/370 System Release 4.2    01/07/88    *
 *    Copyright (C) 1988 Donald S. Higgins     *
 *                                             *
 * You are encouraged to copy and share this   *
 * package with other users on the condition   *
 * the package is not distributed in modified  *
 * form, and that no fee is charged.  If you   *
 * find PC/370 useful, send 45 dollars to the  *
 * address below to become registered user and *
 * support continued shareware development.    *
 * Registered users will receive notices of    *
 * future PC/370 releases.                     *
 *                                             *
 *      Don Higgins                            *
 *      6365 - 32 Avenue, North                *
 *      St. Petersburg, Florida 33710          *
 ***********************************************
PC/370 LINKAGE EDITOR  OPTIONS ON = LXEFIP
STATS SYM=00002  MAXSTD=00001  BTMEM=57756
NO ERRORS FOUND

A:\MIN>hello
Hello world!

A:\MIN>
```

Note: I have a letter on file from Mr. Don Higgins indicating that he no longer accepts registrations for PC/370. You are free to use this product without registration or fee. So disregard this message! Thanks, Don!



Creating PC/370 Datasets

Most of the programs shown throughout this text will use the files of the Small Town Community College database. That database is shown here in its entirety:

STUDENT

SID	SNAME	SSEX	SMAR
713	HILMER, D.R.	F	M
421	QUALLS, G.E.	M	S
701	ARIAS, I.L.	F	M
125	MORALES, L.A.	F	M
896	QUALLS, D.M.	F	S
626	MERCIER, J.L.	F	M
402	FOOTE, A.K.	F	M
263	HAVLIK, K.M.	M	M

COURSE

CID	CDESC	CHRS
AC101	ACCOUNTING	3
BU101	BUSINESS	3
EG101	ENGLISH I	3
EG102	ENGLISH II	3
MA101	ALGEBRA	3
MA107	STATISTICS	3
PE121	FIRST AID	1
PE151	AEROBICS	1

TEACHER

TID	TNAME	TDEG	TTEN	TPHONE
732	BENSON, E.T.	PHD	N	5156
218	HINCKLEY, G.B.	MBA	N	5509
854	KIMBALL, S.W.	PHD	Y	5594
626	YOUNG, B.	MBA	Y	5664
574	SMITH, J.	MS	Y	5320

OFFER

SEM	CID	SECT	TID	ROOM
W92	EG102	1	732	A1
W92	MA107	1	218	A2
W92	PE151	1	574	GYM
F92	AC101	1	218	B1
F92	BU101	1	218	B1
F92	EG101	1	732	A1
F92	EG101	2	732	A1
F92	MA101	1	626	A2
F92	MA101	2	626	A2
F92	PE151	1	574	GYM
W93	EG102	1	854	A1
W93	MA101	1	626	A2
W93	MA107	1	626	A3
W93	PE151	1	574	GYM

GRADE

SID	SEM	CID	SECT	GRADE
626	W92	EG102	1	A
896	W92	PE151	1	A
263	W92	PE151	1	C
896	F92	AC101	1	C
896	F92	BU101	1	C
896	F92	EG101	1	A
713	F92	EG101	2	C
421	F92	EG101	2	B
713	F92	MA101	1	B
896	F92	MA101	1	B
125	F92	MA101	2	F
701	F92	MA101	2	B
263	F92	PE151	1	B
701	F92	PE151	1	A
713	W93	EG102	1	B
421	W93	EG102	1	A
896	W93	EG102	1	B
125	W93	MA101	1	C
713	W93	MA107	1	B
896	W93	MA107	1	A
701	W93	MA107	1	D
263	W93	PE151	1	A

* *

Consider the following table (file) of teachers taken from the Small Town Community College database:

TID	TNAME	TDEG	TTEN	TPHONE
732	BENSON, E.T.	PHD	N	5156
218	HINCKLEY, G.B.	MBA	N	5509
854	KIMBALL, S.W.	PHD	Y	5594
626	YOUNG, B.	MBA	Y	5664
574	SMITH, J.	MS	Y	5320

The following record layout describes the TEACHER file:

Field Nbr	Field Name	Description	Begins	Ends	Len	Format
1	TID	Teacher ID	1	3	3	ZD
2	TNAME	Teacher name	4	18	15	CH
3	TDEG	Highest degree	19	22	4	CH
4	TTEN	Tenured?	23	23	1	Y/N
5	TPHONE	Teacher phone	24	27	4	ZD
6	TCRLF	PC/370 Only	28	29	2	CR/LF

ZD stands for zoned decimal. This means that the field is numeric, but stored in an immediately readable format (as opposed to packed or binary, which we will discuss later.) If you know COBOL, this is the same as a PIC 9 field.

CH stands for character. This means the field may or may not be alphabetical. Anything goes. In COBOL, this is the same as a PIC X field.

CR/LF stands for carriage return/line feed. This is a very important consideration when using PC/370. In a mainframe enviroment, datasets usually use fixed length records; that is, all records have the same number of characters, and each field of a record begins in the same position in all records. If your programming experience is limited to BASIC, this will likely be very new to you.

Whenever you use a PC to create a dataset such as this one, you will, naturally, press the Enter key after each record or row. On a PC, pressing the Enter key adds two characters to a record: carriage return (CR) and line feed (LF). These are true characters, each of which occupies one byte of memory or storage, as would any other character, such as an 'A' or '*'. The CR/LF must be accounted for in determining the length of a record when using PC/370.

In the following example, DOS' COPY command is used to create the TEACHER file in the \MIN directory. CON is a DOS device name meaning console, or keyboard. This is a convenient way to create small datasets using DOS. User entries are shown in bold. Note that ^Z is read as "control Z" and is entered by pressing Z while holding down the Ctrl key, or by pressing the F6 function key. Press Enter after each line, including after ^Z.

```
A:\>cd min
A:\MIN>copy con teacher.dat
732BENSON, E.T.   PHD N5156
218HINCKLEY, G.B. MBA N5509
854KIMBALL, S.W.  PHD Y5594
626YOUNG, B.      MBA Y5664
574SMITH, J.      MS  Y5320
^Z
        1 File(s) copied

A:\MIN>dir teacher.dat

 Volume in drive A has no label
 Directory of  A:\MIN

TEACHER  DAT      145   6-23-93   8:01a
        1 File(s)    932864 bytes free

A:\MIN>type teacher.dat
732BENSON, E.T.   PHD N5156
218HINCKLEY, G.B. MBA N5509
854KIMBALL, S.W.  PHD Y5594
626YOUNG, B.      MBA Y5664
574SMITH, J.      MS  Y5320
```

Note from the DIR command that the total file length is 145 bytes. Why? Each record uses 27 bytes for data and 2 bytes for the CR/LF, for a total of 29 bytes per record. There are 5 records in the file, and 5 * 29 = 145 bytes. This shows that, as stated earlier, when using a PC, pressing the Enter key adds two characters to a record.

There is a real disadvantage to using COPY CON to create a dataset: if you make a mistake you must press ^Z and Enter to end, and then start over. It is much easier to use an editor to create a dataset.

WARNING: some PC editors "trim" trailing blanks. Such editors are fine for writing PC/370 source code, but cannot be used to create PC/370 datasets. If you are not sure how your editor works, use the DIR command to check the file size and verify it just as we did above.

If, after going through the above process to validate the file size, you find that you are off by 1 byte only, don't worry. Your editor probably added a special end of file (EOF) character to your file. That's OK.

Here's another example. Consider the course offerings table from the Small Town Community College database:

SEM	CID	SECT	TID	ROOM
W92	EG102	1	732	A1
W92	MA107	1	218	A2
W92	PE151	1	574	GYM
F92	AC101	1	218	B1
F92	BU101	1	218	B1
F92	EG101	1	732	A1
F92	EG101	2	732	A1
F92	MA101	1	626	A2
F92	MA101	2	626	A2
F92	PE151	1	574	GYM
W93	EG102	1	854	A1
W93	MA101	1	626	A2
W93	MA107	1	626	A3
W93	PE151	1	574	GYM

The following record layout describes the OFFER file:

Field Nbr	Field Name	Description	Begins	Ends	Len	Format
1	SEM	Semester	1	3	3	CH
2	CID	Course ID	4	8	5	CH
3	SECT	Section number	9	9	1	ZD
4	TID	Teacher ID	10	12	3	ZD
5	ROOM	Room number	13	16	4	CH
6	OCRLF	PC/370 Only	17	18	2	CR/LF

Notice that the room number is defined as four bytes long, but its values are only two or three bytes long (e.g., A1 or GYM). Since that field is defined as four bytes long, four bytes must be entered! In the following example, I pressed the space bar twice after each two byte room number, and once after each GYM, before pressing Enter. Try it yourself. As you can see, the total number of bytes is 252. This is because (16 + 2) * 14 = 18 * 14 = 252.

```
A:\MIN>copy con offer.dat
W92EG1021732A1
W92MA1071218A2
W92PE1511574GYM
F92AC1011218B1
F92BU1011218B1
F92EG1011732A1
F92EG1012732A1
F92MA1011626A2
F92MA1012626A2
F92PE1511574GYM
W93EG1021854A1
W93MA1011626A2
W93MA1071626A3
W93PE1511574GYM
^Z
        1 File(s) copied
```

```
A:\MIN>dir offer.dat

 Volume in drive A has no label
 Directory of  A:\MIN

OFFER    DAT      252   6-23-93  10:25a
        1 File(s)    899072 bytes free
```

Our First File Processing Program

For our first file processing program, we will produce a list of the records in the TEACHER file. This program will read a record, write a record, with no formatting at all. Such a list is commonly referred to as an 80/80 list or card-image list. The term goes back to the days of punched cards. Each punched card held 80 characters, and it was common to have a program which would simply list the cards in a deck. That is what our program will do. We'll clean it up later.

Our purpose in doing this program is to introduce those "overhead" items found in all of our programs. In many cases they won't be explained in detail, anymore than the PROCEDURE DIVISION statement would be explained in a beginning COBOL class: we will simply point it out and say how to use it without getting into the whys at this point.

The program and its output are shown on the next page. Here is our brief introduction to these very common commands. We'll talk about most of them in more detail later.

PRINT NOGEN	Don't print the macro expansions. That means nothing to you now and, in fact, it has no effect in PC/370. But it's something that you will find at the beginning of almost every BAL program, and you should use it too.
*	Asterisk in column one indicates that the entire line is a comment. Use them liberally.
START 0	Marks the start of your program. Assembler directives (such as PRINT NOGEN) and comments may come before this statement. That's a zero after START. Always include this statement.
REGS	A macro that assigns a name to each register (for example, R6 for register six). The need for such a macro is so common that a similar macro will be found in most BAL shops. It may be called EQUATE, REGEQU, or something similar. It is not needed in this particular program, but it is usually added to all programs by habit, and its use here serves to double check that you have PC/370 properly installed.

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  TEACH1A.MLC                                 *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  A card-image list of teacher records.       *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         OI    TEACHERS+10,X'08'  PC/370 ONLY - Convert all
*                                 input from ASCII to EBCDIC
         OPEN  TEACHERS
LOOP     GET   TEACHERS,IREC      Read a single teacher record
         WTO   IREC               Display the record
         B     LOOP               Repeat
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         RETURN
*
*        Literals, if any, will go here
*
         LTORG
*
*        File definitions
*
TEACHERS DCB   LRECL=29,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='TEACHER.DAT'
*
*        Input record definition
*
IREC     DS    CL29               Teacher record
         END   BEGIN
```

```
A:\MIN>teach1a
732BENSON, E.T.   PHD N5156

218HINCKLEY, G.B. MBA N5509

854KIMBALL, S.W.  PHD Y5594

626YOUNG, B.      MBA Y5664

574SMITH, J.      MS  Y5320
```

`BEGIN BEGIN`

The first `BEGIN` assigns a name to the statement: anything other than an asterisk in position one indicates a label, such as a field name or paragraph name. The second `BEGIN` is a macro which saves the values of the registers so that when your program is finished, the operating system will be able to return to where it came from. As with the `REGS` macro, this is so common that a similar macro will be found in most `BAL` shops. It may be called many things: I've seen it as `BEGIN`, `BASE13`, and `GENESIS`. For our purposes, code it just as you see it here.

`OI TEACHERS+10,X'08'`

I won't explain the instruction at this time, except to say that this command must be used for all files. Recall that PCs use the ASCII character set to represent data, whereas IBM mainframes (and hence PC/370) use the EBCDIC character set. Consequently, all inputs must be converted from ASCII to EBCDIC as they are read, and all outputs must be converted from EBCDIC to ASCII as they are written. This instruction is the means by which you tell PC/370 to do this. The file name (`TEACHERS` in this case) corresponds to the name of a file on a `DCB` statement (see below).

This is for PC/370 only. Do not do this with mainframe `BAL`.

`OPEN TEACHERS`

This command opens the `TEACHERS` file. As with all programming languages, a file must be opened before it can be used. The `OPEN` macro for PC/370 does not include a mode (such as `INPUT` or `OUTPUT`); you will need to specify a mode when opening a file with mainframe `BAL`.

The `MACRF` parameter of the corresponding `DCB` indicates the mode. `MACRF=G` indicates open for input (`G` = get), while `MACRF=P` indicates open for output (`P` = put). This differs slightly from mainframe `BAL`.

By now you may be thinking that PC/370 is quite different from mainframe `BAL`. It really isn't. Most of the differences have to do with file structure and macros, all of which is discussed here. Converting from PC/370 to mainframe `BAL` is really quite trivial!

`LOOP GET TEACHERS,IREC` This is the main loop in the program, hence the `LOOP` label. The `GET` macro reads a single record from the `TEACHERS` file into the `IREC` record area.

`WTO IREC` `WTO` stands for Write to Operator. It is a macro which, as we saw in the `HELLO.MLC` program, is used to display a message on the screen. In this case we are using it to display a record from the `TEACHERS` file.

Notice that the records were displayed double-spaced. This is because each record ended with `CR/LF`. The `WTO` macro always sends its own `CR/LF` as well, so in this case the records are double-spaced. We will fix this in our next program.

`B LOOP` This a a `BRANCH` to `LOOP`. Similar to a `GO TO` statement in `COBOL` or `BASIC`.

`ATEND CLOSE TEACHERS` `ATEND` is a label to which the program will branch when the `TEACHERS` file is at end-of-file. The `EODAD` parameter of the `DCB` macro tells the assembler where to branch at end of file. Having reached the end of the file, we close the file just as we would in `COBOL` or `BASIC`.

`RETURN` The `RETURN` macro returns to the calling program (usually the operating system). It undoes what the `BEGIN` macro did. It serves the same purpose as a `GOBACK` in `COBOL`. As with the `BEGIN` macro, this is so common that a similar macro will be found in most `BAL` shops. It may be called many things: I've seen it as `RETURN`, `EOJ`, and `EXODUS`. We will use it in all of our programs.

`LTORG` This command tells the assembler to place any generated literals here. Literals are generated when you use a literal constant rather than defining a field with the same value. I always place this after the `RETURN` statement. The reason for doing so has to do with addressibility; something we will not discuss at this time.

`TEACHERS DCB` `DCB` stands for Data Control Block. There is one `DCB` for each file in the program. Since this `DCB` has a label of `TEACHERS`, that will be the name by which I refer to this `DCB`. The `DCB` macro has many parameters, but usually only a few of them are used. They are `LRECL`, `RECFM`, `MACRF`, `EODAD`, and `DDNAME`.

`LRECL=29`	`LRECL` stands for Logical Record Length. The `LRECL` includes the `CR/LF` discussed earlier.
`RECFM=F`	`RECFM` (read as "rec form") stands for Record Format. Record format could be `F` (fixed) or `V` (variable). `RECFM=F` indicates that all records in the file have the same number of characters. `RECFM=V` indicates that the record length may vary from record to record. We will discuss how `BAL` works with variable length records later. Note that this is not the same as variable length records on a PC, such as comma-delimited files in `BASIC`.
`MACRF=G`	`MACRF` stands for Macro Format. Use `G` (get) for input files, and `P` (put) for output files.
`EODAD=ATEND`	As mentioned above, this indicates the label to which the program should go when after attempting to read a record, an end-of-file condition was detected. This is the same as `AT END` processing in `COBOL`, which is why I have chosen `ATEND` as the label here. This parameter is not used on output files.
`DDNAME='TEACHER.DAT'`	The `DDNAME` parameter is used to indicate which DOS file is to read from (in the case of `MACRF=G`) or created or replaced (in the case of `MACRF=P`). This is similar to the `ASSIGN TO` clause of the `SELECT` statement in `COBOL`, or the filename parameter of the `OPEN` statement in `BASIC`. (In mainframe `BAL`, this would refer to the `DDNAME` in the `JCL`.)
`IREC DS CL29`	`DS` stands for Define Storage. This indicates that 29 bytes of storage are to be reserved for a record which will be referred to as `IREC`. This is similar to a `PIC X(29)` in `COBOL`.
`END BEGIN`	This is the end of the program segment called `BEGIN`. The `BEGIN` refers to the `BEGIN` label of the `BEGIN` macro discussed earlier. This will always be the last statement in your program.

* *

When we discussed the `WTO` command above, we pointed out that the records were displayed double-spaced because each record ended with `CR/LF` and that the `WTO` macros always sends its own `CR/LF` as well. Therefore, if we would like these records to appear single spaced, we need to `WTO` the first 27 bytes of the record only. In order to do so, we

need to be able to redefine our input record, so that we can refer to the entire record (such as when we GET it), but display only a portion of it. To do so, we code the following:

```
IREC     DS    0CL29              Teacher record
IDATA    DS    CL27               Teacher data
ICRLF    DS    CL2                PC/370 only - CR/LF
```

We have replaced CL29 with 0CL29. This means that IREC refers to the next twenty nine bytes, but that this field will be further broken down. Notice that the first twenty seven bytes are called IDATA and bytes twenty eight and twenty nine are called ICRLF. In COBOL this would look like:

```
01  IREC.
    05  IDATA       PIC X(27).
    05  ICRLF       PIC X(2).
```

We then WTO the IDATA portion only. The complete program, TEACH1B.MLC, and its output is shown on the next page.

* *

We have introduced the WTO command here because of its simplicity. In practice, the WTO command would never be used to list the records in a file such as we have done here. Typically the command is used for sending a few, select messages to the console, such as how the program is progressing (for example, BEGINNING SORT and SORT COMPLETED), and for debugging. When creating a report, we write the lines, or records, to a file. We can then decide what to do with that file. On a PC, we may use DOS' TYPE command to view it, or use the COPY command to send it to the printer, or import it into another package, such as a word processor or spreadsheet. In the mainframe world, we would use JCL to assign the file to a specific device, such as a print queue, tape drive, disk drive, microfiche, etc.

Our next change to this program removes the WTOs and, instead, writes the records to a file. The following changes are necessary:

`OI REPORT+10,X'08'`	Convert the output from EBCDIC to ASCII before writing.
`OPEN REPORT`	A file must be opened before it can be read from or written to.
`PUT REPORT,IREC`	Write a record. Note LRECL=29 in the DCB, the length of IREC is 29, and that IREC ends with a CR/LF, which has been read in from the input file.

```
          PRINT NOGEN
*****************************************************************
*         FILENAME:  TEACH1B.MLC                                *
*         AUTHOR  :  Bill Qualls                                *
*         SYSTEM  :  PC/370 R4.2                                *
*         REMARKS :  A card-image list of teacher records.      *
*****************************************************************
          START 0
          REGS
BEGIN     BEGIN
          OI    TEACHERS+10,X'08'  PC/370 ONLY - Convert all
*                                  input from ASCII to EBCDIC
          OPEN  TEACHERS
LOOP      GET   TEACHERS,IREC      Read a single teacher record
          WTO   IDATA              Display the record
          B     LOOP               Repeat
*
*         EOJ processing
*
ATEND     CLOSE TEACHERS
          RETURN
*
*         Literals, if any, will go here
*
          LTORG
*
*         File definitions
*
TEACHERS DCB    LRECL=29,RECFM=F,MACRF=G,EODAD=ATEND,
                DDNAME='TEACHER.DAT'
*
*         Input record definition
*
IREC      DS    0CL29              Teacher record
IDATA     DS    CL27               Teacher data
ICRLF     DS    CL2                PC/370 only - CR/LF
          END   BEGIN
```

```
A:\MIN>teach1b
732BENSON, E.T.   PHD N5156
218HINCKLEY, G.B. MBA N5509
854KIMBALL, S.W.  PHD Y5594
626YOUNG, B.      MBA Y5664
574SMITH, J.      MS  Y5320
```

`REPORT DCB`	Each file needs a `DCB`.
`LRECL=29,RECFM=F`	Same as the `TEACHERS` file.
`MACRF=P`	`P` stands for put; this file is opened for output.
`DDNAME='REPORT.TXT'`	The filename of the disk file to contain the output records is `REPORT.TXT`. If this file already exists, it will be deleted and a new file will be created.

The complete program, `TEACH1C.MLC`, and its output is shown on the next page. Note how the DOS `TYPE` command is used to display the output.

* *

One problem with TEACH1C.MLC is that the user must *remember* that the output report is found on REPORT.TXT. Let's make one last change to this program for this chapter: let's add three WTOs as follows:

`WTO` to indicate	**Placement of `WTO`**
Program has begun execution	After the `BEGIN` statement
Output file is available, and where	After the `CLOSE` statement for that file
Program has completed normally	Before the `RETURN` statement

The new program is TEACH1D.MLC: the program and its output follow.

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  TEACH1C.MLC                                 *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  A card-image list of teacher records.       *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         OI    TEACHERS+10,X'08'  PC/370 ONLY - Convert all
*                                 input from ASCII to EBCDIC
         OI    REPORT+10,X'08'    PC/370 ONLY - Convert all
*                                 output from EBCDIC to ASCII
         OPEN  TEACHERS
         OPEN  REPORT
LOOP     GET   TEACHERS,IREC      Read a single teacher record
         PUT   REPORT,IREC        Write the record
         B     LOOP               Repeat
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
         RETURN
*
*        Literals, if any, will go here
*
         LTORG
*
*        File definitions
*
TEACHERS DCB   LRECL=29,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='TEACHER.DAT'
REPORT   DCB   LRECL=29,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*
*        Input record definition
*
IREC     DS    0CL29              Teacher record
IDATA    DS    CL27               Teacher data
ICRLF    DS    CL2                PC/370 only - CR/LF
         END   BEGIN
```

```
A:\MIN>teach1c

A:\MIN>type report.txt
732BENSON, E.T.   PHD N5156
218HINCKLEY, G.B. MBA N5509
854KIMBALL, S.W.  PHD Y5594
626YOUNG, B.      MBA Y5664
574SMITH, J.      MS  Y5320
```

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  TEACH1D.MLC                                 *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  A card-image list of teacher records.       *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TEACH1D ... Begin execution'
         OI    TEACHERS+10,X'08'  PC/370 ONLY - Convert all
*                                 input from ASCII to EBCDIC
         OI    REPORT+10,X'08'    PC/370 ONLY - Convert all
*                                 output from EBCDIC to ASCII
         OPEN  TEACHERS
         OPEN  REPORT
LOOP     GET   TEACHERS,IREC      Read a single teacher record
         PUT   REPORT,IREC        Write the record
         B     LOOP               Repeat
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH1D ... Teacher list on REPORT.TXT'
         WTO   'TEACH1D ... Normal end of program'
         RETURN
```

(Remainder of program is unchanged)

```
A:\MIN>teach1d
TEACH1D ... Begin execution
TEACH1D ... Teacher list on REPORT.TXT
TEACH1D ... Normal end of program

A:\MIN>type report.txt
732BENSON, E.T.   PHD N5156
218HINCKLEY, G.B. MBA N5509
854KIMBALL, S.W.  PHD Y5594
626YOUNG, B.      MBA Y5664
574SMITH, J.      MS  Y5320
```

Exercises

1. True or false.

   T F a. BAL is an acronym for Beginning Assembler Language.
   T F b. BAL is a first generation programming language.
   T F c. Labels, if any, must begin in column 1.
   T F d. Operations precede their operands in BAL.
   T F e. By convention, an operation is coded beginning in column 10.
   T F f. The file extension for the programmer's source code is .MLC.
   T F g. Source code errors are shown in the .LST file.
   T F h. PC/370's L370 produces a stand-alone executable module.
   T F i. The CR/LF is a special character sequence occupying four bytes.
   T F j. Some PC editors "trim" trailing blanks.
   T F k. In PC/370, as with most programming languages, the file mode is included on the OPEN statement.
   T F l. PCs use the ASCII collating sequence.
   T F m. The LRECL parameter of the DCB should include the CR/LF if any.

2. Modify HELLO.MLC to display your name and full address on three lines.

3. Create the student file for the Small Town Community College database. Call your file STUDENT.DAT.

Field Nbr	Field Name	Description	Begins	Ends	Len	Format
1	SID	Student ID	1	3	3	ZD
2	SNAME	Student name	4	18	15	CH
3	SSEX	Gender	19	19	1	CH
4	SMAR	Marital Status	20	20	1	CH
5	SCRLF	PC/370 Only	21	22	2	CR/LF

SID	SNAME	SSEX	SMAR
713	HILMER, D.R.	F	M
421	QUALLS, G.E.	M	S
701	ARIAS, I.L.	F	M
125	MORALES, L.A.	F	M
896	QUALLS, D.M.	F	S
626	MERCIER, J.L.	F	M
402	FOOTE, A.K.	F	M
263	HAVLIK, K.M.	M	M

Exercises

4. Create the grades file for the Small Town Community College database. Call your file GRADE.DAT.

Field Nbr	Field Name	Description	Begins	Ends	Len	Format
1	SID	Student ID	1	3	3	ZD
2	SEM	Semester	4	6	3	CH
3	CID	Course ID	7	11	5	CH
4	SECT	Section number	12	12	1	ZD
5	GRADE	Grade earned	13	13	1	CH
6	GCRLF	PC/370 Only	14	15	2	CR/LF

SID	SEM	CID	SECT	GRADE
626	W92	EG102	1	A
896	W92	PE151	1	A
263	W92	PE151	1	C
896	F92	AC101	1	C
896	F92	BU101	1	C
896	F92	EG101	1	A
713	F92	EG101	2	C
421	F92	EG101	2	B
713	F92	MA101	1	B
896	F92	MA101	1	B
125	F92	MA101	2	F
701	F92	MA101	2	B
263	F92	PE151	1	B
701	F92	PE151	1	A
713	W93	EG102	1	B
421	W93	EG102	1	A
896	W93	EG102	1	B
125	W93	MA101	1	C
713	W93	MA107	1	B
896	W93	MA107	1	A
701	W93	MA107	1	D
263	W93	PE151	1	A

5. Create the course file for the Small Town Community College database. Call your file COURSE.DAT.

Field Nbr	Field Name	Description	Begins	Ends	Len	Format
1	CID	Course ID	1	5	5	CH
2	CDESC	Description	6	20	15	CH
3	CHRS	Hours	21	21	1	ZD
4	CCRLF	PC/370 Only	22	23	2	CR/LF

(continued)

Exercises

5. *(continued)*

CID	CDESC	CHRS
AC101	ACCOUNTING	3
BU101	BUSINESS	3
EG101	ENGLISH I	3
EG102	ENGLISH II	3
MA101	ALGEBRA	3
MA107	STATISTICS	3
PE121	FIRST AID	1
PE151	AEROBICS	1

6. Modify TEACH1D.MLC to produce a list of the records in the student file, STUDENT.DAT. Be sure to change the program name, file names, and comments as necessary so that they are appropriate.

7. Modify TEACH1D.MLC to produce a list of the records in the course file, COURSE.DAT. Be sure to change the program name, file names, and comments as necessary so that they are appropriate.

8. Modify TEACH1D.MLC to produce a list of the records in the grades file, GRADE.DAT. Be sure to change the program name, file names, and comments as necessary so that they are appropriate.