Chapter 4
How to Structure a BAL Program

Objectives

Upon completion of this chapter you will be able to:

- Define register in the context of the System/370 architecture,
- Define a fullword,
- Explain what effect, if any, each of the following instructions have on a register: BAL, BR, ST, and L.
- Explain how the BAL, BR, ST, and L instructions can be used to structure an assembler program,
- Explain the purpose of PC/370's REGS macros,
- Describe the typical components of the MAINLINE, SETUP, INPUT, PROCESS, OUTPUT, and WRAPUP sections of a structured program, and
- Write a structured program.

Introduction

Some would say that "structured BAL" is an oxymoron; that it cannot be done. BAL has a very limited instruction set; it is, afterall, a low level language. Most programmers have been taught to use structured programming, a significant part of which is GO-TO-less programming. The BAL equivalent to a COBOL or BASIC GOTO is the branch instruction, the use of which is unavoidable in most BAL programs. If GO-TO-less programming is part of your criteria for structured programming, then it is true that BAL cannot be "structured". But selective use of the rules of structured programming can go a long way towards improving program readability, maintainability, and reusability.

In this chapter we learn how to structure a BAL program. Specifically, we will learn to use the following instructions: BAL, BR, ST, and L, as well as another use for EQU.

SETUP, PROCESS, and WRAPUP Sections

We will continue with program TEACH3A.MLC from the previous chapter. That program produces a list of tenured instructors from the TEACHER file. Our first step is to break up the program into three sections:

- SETUP, or those things that happen one time only before any records are processed,
- PROCESS, or those things that happen once per record, and
- WRAPUP, or those things that happen one time only after all records have been processed.

Most business programs follow a similar structure. They might use the same names, or something similar. For example, SETUP is sometimes called HOUSEKEEPING (commonly abbreviated as HSK), INITIALIZATION, or BOJ (Begin of Job), and WRAPUP is sometimes called TERMINATION or EOJ (End of Job).

Below is the logic portion (only) of `TEACH3A.MLC`. A box has been drawn around those portions of the program which represent the `SETUP`, `PROCESS`, and `WRAPUP` sections.

```
         PRINT NOGEN
****************************************************************
*        FILENAME:  TEACH3A.MLC                                *
*        AUTHOR  :  Bill Qualls                                *
*        SYSTEM  :  PC/370 R4.2                                *
*        REMARKS :  Produce a list of tenured instructors.     *
****************************************************************
         START 0
         REGS
BEGIN    BEGIN
                                                     SETUP
         WTO   'TEACH3A ... 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
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
                                                     PROCESS
LOOP     EQU   *
         GET   TEACHERS,IREC      Read a single teacher record
         CLI   ITTEN,C'Y'         Is teacher tenured?
         BNE   LOOP               No, then skip this record
         MVC   OTID,ITID          Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME      Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'    Highest degree = PhD?
         BE    YESPHD             .. Yes, branch
         MVI   OPHD,C'N'          .. No, Show PhD = 'N'
         B     OTHERS             .. Branch around YES logic
YESPHD   EQU   *                  Highest degree is PhD, so...
         MVI   OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU   *                  Continue moving other fields...
         MVC   O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE    Move phone nbr to output
         MVC   OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
         PUT   REPORT,OREC        Write report line
         B     LOOP
                                                     WRAPUP
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH3A ... Teacher list on REPORT.TXT'
         WTO   'TEACH3A ... Normal end of program'
         RETURN
```

The logic which drives these sections of code is commonly referred to as the MAINLINE. In order to so structure our program, we need an instruction which will enable us to invoke subroutines; that is, something comparable to BASIC's GOSUB or COBOL's PERFORM whereby we can go to a subroutine and then come back once that subroutine has finished. In BASIC, every GOSUB must have a RETURN; the RETURN is a required statement. In COBOL, the return is implied by the end of the designated procedure, be it a paragraph or section. The BAL implementation of this process is more similar to BASIC than to COBOL, in that the return must be coded.

The BAL equivalent to a GOSUB is BAL, or Branch and Link. Don't confuse BAL (Basic Assembly Language) with BAL (the Branch and Link instruction)! The BAL equivalent to a RETURN is BR, or Branch Register. In order to understand the BAL and BR instructions, we must understand a little bit about **registers**.

A **register** is a special storage area within the CPU (central processing unit). The size of a register will depend upon the CPU. Most PCs, for example, have two-byte registers, while the IBM 370 computer has four-byte registers. The size of a computer's register is referred to as its **word size**. Each register is capable of holding an address, and it stands to reason that the larger the register, or word size, the larger the address it can hold. (How this is represented internally will be discussed in more detail later.) The number of registers will also vary by CPU type. The IBM 370 computer has sixteen registers. These registers are numbered 0 through 15 and are referred to by number. Even though the PC has a different number of registers, PC/370 is emulating a mainframe, so we program as if we had all 16 registers.

As the name implies, the Branch and Link instruction does a branch, and a little more. For example, the instruction BAL 10,SETUP says to put the address of the next instruction (the instruction immediately following the BAL instruction) into register 10, and *then* branch to the label SETUP.

So how do we know when and where to return? Well, since register 10 has the address of the next instruction, we simply return to the address to which register 10 is pointing. To do so, we code BR 10. **Do not confuse this with a branch to register 10!** The instruction B 10 will assemble and will execute, but with unpredictable results! The BR instruction says to branch to the address *contained in* the stated register. If you have coded in C before, this will be an easy concept to pick up. If your only prior coding experience is in BASIC or COBOL, this *will* take some getting used to.

I do not *have* to use register 10. To repeat, there are sixteen registers. You cannot use register 13 with PC/370 as this is your **base register**. (It is used by PC/370's BEGIN macro to establish addressibility for the program. This will be discussed in more detail later.) Some BAL instructions and many macros will modify registers 0 and 1, therefore it is common practice to avoid using those registers. Most installations will have a standard as to which register(s) to use for branch-and-links.

The new version of our program, TEACH4A.MLC, follows:

```
         PRINT NOGEN
****************************************************************
*        FILENAME:  TEACH4A.MLC                                *
*        AUTHOR  :  Bill Qualls                                *
*        SYSTEM  :  PC/370 R4.2                                *
*        REMARKS :  This is a revision of TEACH3A.MLC.         *
*                   Produce list of tenured instructors.       *
*                   How to structure a BAL program.            *
****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TEACH4A ... Begin execution'
         BAL   10,SETUP
MAIN     EQU   *
         BAL   10,PROCESS
         B     MAIN
ATEND    EQU   *
         BAL   10,WRAPUP
         WTO   'TEACH4A ... Normal end of program'
         RETURN
****************************************************************
*        SETUP - Those things which happen one time only,      *
*                before any records are processed.             *
****************************************************************
SETUP    EQU   *
         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
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         BR    10
****************************************************************
*        PROCESS - Those things which happen once per record.  *
****************************************************************
PROCESS  EQU   *
         GET   TEACHERS,IREC       Read a single teacher record
         CLI   ITTEN,C'Y'          Is teacher tenured?
         BNE   PROCESSX            No, then skip this record
         MVC   OTID,ITID           Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME       Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'     Highest degree = PhD?
         BE    YESPHD              .. Yes, branch
         MVI   OPHD,C'N'           .. No, Show PhD = 'N'
         B     OTHERS              .. Branch around YES logic
```

(continued)

```
YESPHD   EQU   *                  Highest degree is PhD, so...
         MVI   OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU   *                  Continue moving other fields...
         MVC   O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE    Move phone nbr to output
         MVC   OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
         PUT   REPORT,OREC        Write report line
PROCESSX EQU   *
         BR    10
****************************************************************
*        WRAPUP - Those things which happen one time only,     *
*                 after all records have been processed.       *
****************************************************************
WRAPUP   EQU   *
         CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH4A ... Teacher list on REPORT.TXT'
         BR    10
```

(Remainder of program is the same as TEACH3A.MLC.)

The PROCESS Routine Revisited...

Program TEACH4A.MLC contains two rather serious violations of the rules of structured programming. First, each module or function should have a single, well-defined purpose. This is not the case with our PROCESS module: this module reads a record, evaluates that record according to the extract criteria, formats a report line, and writes a report line:

```
****************************************************************
*        PROCESS - Those things which happen once per record.  *
****************************************************************
PROCESS  EQU   *
```

READ

```
         GET   TEACHERS,IREC      Read a single teacher record
```

EXTRACT

```
         CLI   ITTEN,C'Y'         Is teacher tenured?
         BNE   PROCESSX           No, then skip this record
```

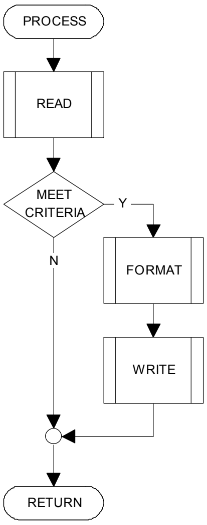
FORMAT

```
         MVC   OTID,ITID          Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME      Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'    Highest degree = PhD?
         BE    YESPHD             .. Yes, branch
         MVI   OPHD,C'N'          .. No, Show PhD = 'N'
         B     OTHERS             .. Branch around YES logic
YESPHD   EQU   *                  Highest degree is PhD, so...
         MVI   OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU   *                  Continue moving other fields...
         MVC   O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE    Move phone nbr to output
         MVC   OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
```

WRITE

```
         PUT   REPORT,OREC        Write report line
```

```
PROCESSX EQU   *
         BR    10
```



A module with a single, well-defined purpose is said to be **cohesive**. The programmer should always strive to **maximize cohesion**. Accordingly, we should split the `PROCESS` module into three modules. As shown in this flowchart, the extract logic will remain a part of the `PROCESS` section, but the `READ`, `FORMAT`, and `WRITE` logic will each become a separate subroutine.

The new `PROCESS` section becomes:

```
****************************************************************
*        PROCESS - Those things which happen once per record.  *
****************************************************************
PROCESS  EQU   *
         BAL   10,READ            Read a single teacher record
         CLI   ITTEN,C'Y'         Is teacher tenured?
         BNE   PROCESSX           No, then skip this record
         BAL   10,FORMAT          Otherwise, format report line
         BAL   10,WRITE           and write report line
PROCESSX EQU   *
         BR    10
****************************************************************
*        READ A RECORD                                         *
****************************************************************
READ     EQU   *
         GET   TEACHERS,IREC      Read a single teacher record
         BR    10
****************************************************************
*        FORMAT A LINE                                         *
****************************************************************
FORMAT   EQU   *
         MVC   OTID,ITID          Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME      Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'    Highest degree = PhD?
         BE    YESPHD             .. Yes, branch
         MVI   OPHD,C'N'          .. No, Show PhD = 'N'
         B     OTHERS             .. Branch around YES logic
YESPHD   EQU   *                  Highest degree is PhD, so...
         MVI   OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU   *                  Continue moving other fields...
         MVC   O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE    Move phone nbr to output
         MVC   OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
****************************************************************
*        WRITE A LINE                                          *
****************************************************************
WRITE    EQU   *
         PUT   REPORT,OREC        Write report line
         BR    10
```

But this won't work! When invoking a module, most languages (such as COBOL's PERFORM and BASIC's GOSUB) utilize a "stack" so as to enable nested invocations such as this. This is *not* the case with BAL. The above example will cause an endless loop. Can you see why? Let's take a closer look.

```
(1)          BAL   10,PROCESS
(2)          B     MAIN
             :
(3)  PROCESS EQU   *
(4)          BAL   10,READ
             CLI   ITTEN,C'Y'
             BNE   PROCESSX
(5)          BAL   10,FORMAT
(6)          BAL   10,WRITE
     PROCESSX EQU  *
(7)          BR    10
```

At the time that instruction *(1)* is executed, register 10 points to the next instruction, instruction *(2)*. The intent is that when we reach the end of the PROCESS module, instruction *(7)* will cause flow of control to transfer to the instruction immediately following its invocation, that being instruction *(2)*. **But this will not happen....**

Let's continue. As a result of instruction *(1)*, flow of control goes to instruction *(3)*. This is a label only; so flow of control falls through to instruction *(4)*. When instruction *(4)* is executed, register 10 points to the next instruction, instruction *(5)*. **We are in trouble already: the value of register 10 has changed!** How will we get back to instruction *(2)*? The BR 10 at the end of the READ routine will bring us back to instruction *(5)*. The same thing happens with instructions *(5)* and *(6)*. When instruction *(6)* is executed, register 10 points to instruction *(7)*. The BR 10 at the end of the WRITE routine will bring us back to instruction *(7)*. Instruction *(7)* says to go to wherever register 10 is pointing. But register 10 is pointing to instruction *(7)*, which says to go to wherever register 10 is pointing.... Endless loop.

One solution to this problem is to use a different register for each branch-and-link. But this solution is unsatisfactory for obvious reasons: you will run out of registers. (There are only sixteen.) And you have to be very careful to invoke a routine (with BAL) using the same register by which that routine will return (with BR).

The solution to this problem is:

Save the return address for each routine immediately upon entry, and
Restore the address to the proper register immediately before leaving.

This way we can use the same register for all Branch and links.

To save the return address, which is in a register, we use the `ST` (Store) instruction. Where will we save it? In a field which we will define and set aside just for this purpose. Recall the IBM 370 computer has four-byte registers, and the size of a computer's register is referred to as its word size. So we will use a `DC` with field type of **fullword** to save the contents of the register. For example:

```
READ      EQU   *
          ST    10,SVREAD
```

where

```
SVREAD    DC    F'0'
```

(Note that the Store instruction is one of a select few where the first operand is the sending field, not the receiving field. It's backwards of most instructions.)

To restore the value to the register, we use the `L` (Load) instruction. For example:

```
          L     10,SVREAD
          BR    10
```

This is a simple but effective technique for invoking modules and will be used throughout this book. *Note: This technique will not support recursive invocation, but the need for such processing is rare in business applications. If you want to do recursive invocations, you will need to define a stack, and code the equivalent of `PUSH` and `POP` stack operations. We haven't learned enough `BAL` to do that yet.*

Following is a partial listing of `TEACH4B.MLC`, our new version of the program.

```
          PRINT NOGEN
****************************************************************
*         FILENAME:  TEACH4B.MLC                               *
*         AUTHOR  :  Bill Qualls                               *
*         SYSTEM  :  PC/370 R4.2                               *
*         REMARKS :  This is a revision of TEACH4A.MLC.        *
*                    Produce list of tenured instructors.      *
*                    How to structure a BAL program.           *
****************************************************************
          START 0
          REGS
BEGIN     BEGIN
          WTO   'TEACH4B ... Begin execution'
          BAL   10,SETUP
MAIN      EQU   *
          BAL   10,PROCESS
          B     MAIN
ATEND     EQU   *
          BAL   10,WRAPUP
          WTO   'TEACH4B ... Normal end of program'
          RETURN
```

(continued)

```
****************************************************************
*        SETUP - Those things which happen one time only,      *
*                before any records are processed.             *
****************************************************************
SETUP    EQU   *
         ST    10,SVSETUP
         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
         BAL   10,HDGS
         L     10,SVSETUP
         BR    10
****************************************************************
*        HDGS - Print headings.                                *
****************************************************************
HDGS     EQU   *
         ST    10,SVHDGS
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         L     10,SVHDGS
         BR    10
****************************************************************
*        PROCESS - Those things which happen once per record.  *
****************************************************************
PROCESS  EQU   *
         ST    10,SVPROC
         BAL   10,READ
         CLI   ITTEN,C'Y'         Is teacher tenured?
         BNE   PROCESSX           No, then skip this record
         BAL   10,FORMAT          Otherwise format a line
         BAL   10,WRITE           ...and write it
PROCESSX EQU   *
         L     10,SVPROC
         BR    10
****************************************************************
*        READ - Read a record.                                 *
****************************************************************
READ     EQU   *
         ST    10,SVREAD
         GET   TEACHERS,IREC      Read a single teacher record
         L     10,SVREAD
         BR    10
****************************************************************
*        FORMAT - Format a single detail line.                 *
****************************************************************
FORMAT   EQU   *
         ST    10,SVFORM
         MVC   OTID,ITID          Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME      Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'    Highest degree = PhD?
         BE    YESPHD             .. Yes, branch
         MVI   OPHD,C'N'          .. No, Show PhD = 'N'
         B     OTHERS             .. Branch around YES logic
```

(continued)

```
YESPHD   EQU   *                  Highest degree is PhD, so...
         MVI   OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU   *                  Continue moving other fields...
         MVC   O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE    Move phone nbr to output
         MVC   OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
         L     10,SVFORM
         BR    10
****************************************************************
*        WRITE - Write a single detail line.                   *
****************************************************************
WRITE    EQU   *
         ST    10,SVWRITE
         PUT   REPORT,OREC        Write report line
         L     10,SVWRITE
         BR    10
****************************************************************
*        WRAPUP - Those things which happen one time only,     *
*                 after all records have been processed.       *
****************************************************************
WRAPUP   EQU   *
         ST    10,SVWRAP
         CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH4B ... Teacher list on REPORT.TXT'
         L     10,SVWRAP
         BR    10
****************************************************************
*        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=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
****************************************************************
*        RETURN ADDRESSES                                      *
****************************************************************
SVSETUP  DC    F'0'               SETUP
SVHDGS   DC    F'0'               HDGS
SVPROC   DC    F'0'               PROCESS
SVREAD   DC    F'0'               READ
SVFORM   DC    F'0'               FORMAT
SVWRITE  DC    F'0'               WRITE
SVWRAP   DC    F'0'               WRAPUP
```

(Remainder of program is the same as TEACH3A.MLC)

The READ Routine Revisited...

Earlier we said there were two violations of the rules of structured programming. The second rule is **each module should have a single entry and a single exit**. This is not entirely possible given our limited instruction set, but some techniques are better than others. Recall that the `EODAD` parameter of the `DCB` macro indicates where the program should go when the corresponding input file reaches end-of-file. In `TEACH4B.MLC` the `EODAD` sends control back to the mainline (to `ATEND`);

that is, *it leaves the READ routine from other than the usual exit (the BR at the end of READ), and returns to the mainline without ever returning to the PROCESS routine which invoked READ in the first place.* This is a violation of the single entry, single exit rule.

For the benefit of the COBOL programmer, consider the COBOL equivalent to what we have just seen:

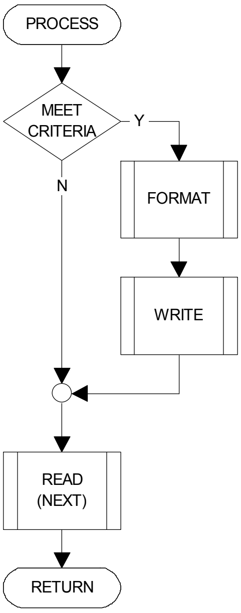
```
PROCEDURE DIVISION.

MAINLINE.
    PERFORM SETUP.

MAIN-LOOP.
    PERFORM PROCESS-A-RECORD.
    GO TO MAIN-LOOP.

AT-END-OF-FILE.
    PERFORM WRAPUP.
    GOBACK.

PROCESS-A-RECORD.
    PERFORM READ-A-RECORD.
    IF extract-criteria-met
       PERFORM FORMAT-A-RECORD
       PERFORM WRITE-A-RECORD.

READ-A-RECORD.
    READ TEACHER-FILE INTO WS-TEACHER-RECORD
        AT END GO TO AT-END-OF-FILE.
```

It is more common in a COBOL program to use an **end-of-file switch**. It is also common to include a **priming read** as the last instruction within the SETUP, and another read as the last instruction in the PROCESS paragraph. This is illustrated in the following program segment.

```
WORKING-STORAGE SECTION.
01  WS-MISC.
    05  END-OF-FILE-SW        PIC X(1) VALUE 'N'.
        88  END-OF-FILE                VALUE 'Y'.
    :
    :

PROCEDURE DIVISION.
MAINLINE.
    PERFORM SETUP.
    PERFORM PROCESS-A-RECORD
      UNTIL END-OF-FILE.
    PERFORM WRAPUP.
    GOBACK.

SETUP.
    :
    :
    PERFORM READ-A-RECORD.
```

```
PROCESS-A-RECORD.
    IF extract-criteria-met
       PERFORM FORMAT-A-RECORD
       PERFORM WRITE-A-RECORD.
    PERFORM READ-A-RECORD.

READ-A-RECORD.
    READ TEACHER-FILE INTO WS-TEACHER-RECORD
        AT END MOVE 'Y' TO END-OF-FILE-SW.

FORMAT-A-RECORD.
    :
    :
```

We can't match this in BAL but we can come close. The problem is that BAL does not have anything equivalent to COBOL's PERFORM UNTIL (which is actually a DOWHILE, or "test before" loop.) However, we can "fake" a DOWHILE with IFs (CLC or CLI) and branches (BC or its mnemonics).



The flowchart shows the PROCESS portion of this improved structure. Note the READ at the bottom of the process, which accomplishes the input for all but the first record. The priming read will be issued from the SETUP routine.

The BAL equivalent to the COBOL program segment is as follows:

```
         BAL   10,SETUP
MAIN     EQU   *
         CLI   EOFSW,C'Y'
         BE    EOJ
         BAL   10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   10,WRAPUP
         :
         :
****************************************************************
*        SETUP - Those things which happen one time only,      *
*                before any records are processed.             *
****************************************************************
SETUP    EQU   *
         :
         :
         BAL   10,READ
         L     10,SVSETUP
         BR    10
         :
         :
```

```
****************************************************************
*        PROCESS - Those things which happen once per record.  *
****************************************************************
PROCESS  EQU   *
         ST    R10,SVPROC
         CLI   ITTEN,C'Y'         Is teacher tenured?
         BNE   PROC2              No, then skip this record
         BAL   R10,FORMAT         Otherwise format a line
         BAL   R10,WRITE          ...and write it
PROC2    EQU   *
         BAL   R10,READ           Read next
PROCESSX EQU   *
         L     10,SVPROC
         BR    10
****************************************************************
*        READ - Read a record.                                 *
****************************************************************
READ     EQU   *
         ST    10,SVREAD
         GET   TEACHERS,IREC      Read a single teacher record
         B     READX
ATEND    EQU   *
         MVI   EOFSW,C'Y'
READX    EQU   *
         L     10,SVREAD
         BR    10
         :
         :
****************************************************************
*        Miscellaneous field definitions                       *
****************************************************************
EOFSW    DC    CL1'N'             End of file? (Y/N)
```

One more comment before we present our final version of the program. If you look at the **cross reference listing** produced by PC/370 (or any other 370 assembler) you will not see any reference to register 10 (or to any other register). (The cross reference listing for PC/370 is `pgmname.PRN`, which is created by `A370`.) Register usage is so common within programs that it is highly desireable to have them included in the cross reference listing for desk-checking and debugging purposes. Furthermore, we wish to distinguish 10 (the register) from 10 (the number, or address, or length, or displacement). This is done with the `EQU` (Equate) instruction. For example: `R10 EQU 10`

This is so common, that PC/370 (and most installations) have a macro which equates all registers for you. I have seen this macro called `EQUATE`, `REGEQU`, and others. PC/370 calls this macro `REGS`. It is coded immediately after the `START` command. When the macro preprocessor, `M370`, encounters this macro, it generates a similar `EQU` instruction for all 16 registers. (You can see this by viewing the `pgmname.ALC` file created by `M370`.) We will use it in `TEACH4C` and all subsequent programs. (Up to this point we have included the `REGS` macro in our programs, but it was optional since we did not reference any registers.) We can then use `R10` in place of 10 whenever we refer to register 10. `R10` will then appear in the cross reference listing produced by the assembler.

Our final version of the program, `TEACH4C.MLC`, follows.

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  TEACH4C.MLC                                 *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  This is a revision of TEACH4B.MLC.          *
*                   Produce list of tenured instructors.        *
*                   How to structure a BAL program.             *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TEACH4C ... Begin execution'
         BAL   R10,SETUP
MAIN     EQU   *
         CLI   EOFSW,C'Y'
         BE    EOJ
         BAL   R10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   R10,WRAPUP
         WTO   'TEACH4C ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                before any records are processed.              *
*****************************************************************
SETUP    EQU   *
         ST    R10,SVSETUP
         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
         BAL   R10,HDGS
         BAL   R10,READ           Priming read
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU   *
         ST    R10,SVHDGS
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS  EQU   *
         ST    R10,SVPROC
         CLI   ITTEN,C'Y'         Is teacher tenured?
         BNE   PROC2              No, then skip this record
         BAL   R10,FORMAT         Otherwise format a line
         BAL   R10,WRITE          ...and write it
```

(continued)

```
PROC2    EQU   *
         BAL   R10,READ           Read next
PROCESSX EQU   *
         L     R10,SVPROC
         BR    R10
****************************************************************
*        READ - Read a record.                                 *
****************************************************************
READ     EQU   *
         ST    R10,SVREAD
         GET   TEACHERS,IREC      Read a single teacher record
         B     READX
ATEND    EQU   *
         MVI   EOFSW,C'Y'
READX    EQU   *
         L     R10,SVREAD
         BR    R10
****************************************************************
*        FORMAT - Format a single detail line.                 *
****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORM
         MVC   OTID,ITID          Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME      Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'    Highest degree = PhD?
         BE    YESPHD             .. Yes, branch
         MVI   OPHD,C'N'          .. No, Show PhD = 'N'
         B     OTHERS             .. Branch around YES logic
YESPHD   EQU   *                  Highest degree is PhD, so...
         MVI   OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU   *                  Continue moving other fields...
         MVC   O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE    Move phone nbr to output
         MVC   OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
         L     R10,SVFORM
         BR    R10
****************************************************************
*        WRITE - Write a single detail line.                   *
****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   REPORT,OREC        Write report line
         L     R10,SVWRITE
         BR    R10
****************************************************************
*        WRAPUP - Those things which happen one time only,     *
*                 after all records have been processed.       *
****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH4C ... Teacher list on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
****************************************************************
*        Literals, if any, will go here                        *
****************************************************************
         LTORG
```

(continued)

```
****************************************************************
*        File definitions                                      *
****************************************************************
TEACHERS DCB   LRECL=29,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='TEACHER.DAT'
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
****************************************************************
*        RETURN ADDRESSES                                      *
****************************************************************
SVSETUP  DC    F'0'               SETUP
SVHDGS   DC    F'0'               HDGS
SVPROC   DC    F'0'               PROCESS
SVREAD   DC    F'0'               READ
SVFORM   DC    F'0'               FORMAT
SVWRITE  DC    F'0'               WRITE
SVWRAP   DC    F'0'               WRAPUP
****************************************************************
*        Miscellaneous field definitions                       *
****************************************************************
WCRLF    DC    X'0D25'            PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'             End of file? (Y/N)
****************************************************************
*        Input record definition                               *
****************************************************************
IREC     DS    0CL29              Teacher record
ITID     DS    CL3                Teacher ID nbr
ITNAME   DS    CL15               Teacher name
ITDEG    DS    CL4                Highest degree
ITTEN    DS    CL1                Tenured?
ITPHONE  DS    CL4                Phone nbr
ITCRLF   DS    CL2                PC/370 only - CR/LF
****************************************************************
*        Output (line) definition                              *
****************************************************************
OREC     DS    0CL62
OTID     DS    CL3                Teacher ID nbr
         DC    CL3' '
OTNAME   DS    CL15               Teacher name
         DC    CL4' '
OPHD     DS    CL1                PhD? (Y/N)
         DC    CL5' '
OPHONE   DS    0CL8               Phone nbr
O517     DS    CL4
OTPHONE  DS    CL4                Phone nbr
         DC    CL21' '
OCRLF    DS    CL2                PC/370 only - CR/LF
****************************************************************
*        Headings definitions                                  *
****************************************************************
HD1      DS    0CL62
         DC    CL40'      LIST OF TENURED INSTRUCTORS       '
         DC    CL20' '
         DC    XL2'0D25'
HD2      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL40'ID#        Name         PhD?    Phone   '
         DC    CL20' '
         DC    XL2'0D25'
```

(continued)

```
HD4      DS    0CL62
         DC    CL40'---   ---------------   ----   -------- '
         DC    CL20' '
         DC    XL2'0D25'
         END   BEGIN
```

If one compares this program (`TEACH4C.MLC`) to the chapter 3 version (`TEACH3A.MLC`), the initial reaction may be that we have succeeded in making a mountain out of a molehill. At first glance the new program appears much more complicated. But it will soon become apparent that programs structured in this manner are much easier to maintain. Also, the use of small, cohesive modules increases the reusability of the code: it's much easier to use portions of one program in another without modification. We will use this structure throughout the remainder of the book. You may be surprised at how little we have to change as we introduce new concepts!

Exercises

1. True or false.

T F a. The size of a computer's registers is referred to as its word size.
T F b. The System/370 computer has sixteen registers numbered 1 thru 16.
T F c. PC/370 uses register 13 as its base register.
T F d. The Store instruction (ST) works like most other BAL instructions; that is, the first operand is the receiving field.
T F e. The Load instruction (L) works like most other BAL instructions; that is, the first operand is the receiving field.
T F f. The Load instruction (L) loads a register into a fullword.
T F g. We use the branch register (BR) instruction to return from a subroutine.
T F h. A different register should be used to invoke each subroutine.
T F i. The techniques shown in this chapter do not support recursion.
T F j. To maximize cohesion means to put as much code into a single subroutine as is possible.
T F k. Each module should have a single entry and a single exit.
T F l. SETUP is those things which happen one time only before any records are processed.
T F m. The READ routine is invoked from the SETUP *and* PROCESS routines.

2. Complete Exercise 14, 15, or 16 of Chapter 3. Your program should be structured, making full use of the concepts shown in this chapter.

3. (Refer to the Small Town Hardware Store database in More Datasets.) Write a program which will list those items in the TOOL file where quantity on hand is at or below the minimum quantity. Do not list wrappers, which are indicated by a sell price of zero. The report should appear as follows:

```
         1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345
                    SMALL TOWN HARDWARE STORE
                       ITEMS TO BE ORDERED

TID        Description          Cost     Sell      QOH     MIN
---   ----------------------   ------   ------   -----   -----
XXX   XXXXXXXXXXXXXXXXXXXXXX   XXX.XX   XXX.XX    XXX     XXX
XXX   XXXXXXXXXXXXXXXXXXXXXX   XXX.XX   XXX.XX    XXX     XXX
XXX   XXXXXXXXXXXXXXXXXXXXXX   XXX.XX   XXX.XX    XXX     XXX
```

Cost and sell price are stored without decimal points, but you should print them so they *do* show a decimal. For example, a value stored as 00425 should be printed as 004.25. Do not be concerned about leading zeroes at this time. Your program should be structured, making full use of the concepts shown in this chapter.

Exercises

4. Predict the results for each of the following programs.

a.

```
         PRINT NOGEN
         START 0
         REGS
BEGIN    BEGIN
         BAL   R9,SUB1
         BAL   R9,SUB2
         RETURN
SUB1     EQU   *
         WTO   'BEGIN SUB1'
         WTO   'LEAVE SUB1'
         BR    R9
SUB2     EQU   *
         WTO   'BEGIN SUB2'
         WTO   'LEAVE SUB2'
         BR    R9
         LTORG
         END   BEGIN
```

b.

```
         PRINT NOGEN
         START 0
         REGS
BEGIN    BEGIN
         BAL   R9,SUB1
         BAL   R8,SUB2
         RETURN
SUB1     EQU   *
         WTO   'BEGIN SUB1'
         BAL   R8,SUB2
         WTO   'LEAVE SUB1'
         BR    R9
SUB2     EQU   *
         WTO   'BEGIN SUB2'
         WTO   'LEAVE SUB2'
         BR    R8
         LTORG
         END   BEGIN
```

c.

```
         PRINT NOGEN
         START 0
         REGS
BEGIN    BEGIN
         BAL   R9,SUB1
         BAL   R9,SUB2
         RETURN
SUB1     EQU   *
         WTO   'BEGIN SUB1'
         BAL   R8,SUB2
         WTO   'LEAVE SUB1'
         BR    R9
SUB2     EQU   *
         WTO   'BEGIN SUB2'
         WTO   'LEAVE SUB2'
         BR    R9
         LTORG
         END   BEGIN
```

d.

```
         PRINT NOGEN
         START 0
         REGS
BEGIN    BEGIN
         BAL   R9,SUB1
         BAL   R9,SUB2
         RETURN
SUB1     EQU   *
         WTO   'BEGIN SUB1'
         BAL   R9,SUB2
         WTO   'LEAVE SUB1'
         BR    R9
SUB2     EQU   *
         WTO   'BEGIN SUB2'
         WTO   'LEAVE SUB2'
         BR    R9
         LTORG
         END   BEGIN
```

e.

```
         PRINT NOGEN
         START 0
         REGS
BEGIN    BEGIN
         BAL   R9,SUB1
         BAL   R9,SUB2
         RETURN
SUB1     EQU   *
         ST    R9,SVSUB1
         WTO   'BEGIN SUB1'
         BAL   R9,SUB2
         WTO   'LEAVE SUB1'
         L     R9,SVSUB1
         BR    R9
SUB2     EQU   *
         ST    R9,SVSUB2
         WTO   'BEGIN SUB2'
         WTO   'LEAVE SUB2'
         L     R9,SVSUB2
         BR    R9
         LTORG
SVSUB1   DC    F'0'
SVSUB2   DC    F'0'
         END   BEGIN
```

f.

```
         PRINT NOGEN
         START 0
         REGS
BEGIN    BEGIN
         BAL   R9,SUB1
         BAL   R9,SUB2
         RETURN
SUB1     EQU   *
         ST    R9,SVSUB1
         WTO   'BEGIN SUB1'
         BAL   R9,SUB2
         WTO   'LEAVE SUB1'
         L     R9,SVSUB2
         BR    R9
SUB2     EQU   *
         ST    R9,SVSUB2
         WTO   'BEGIN SUB2'
         WTO   'LEAVE SUB2'
         L     R9,SVSUB2
         BR    R9
         LTORG
SVSUB1   DC    F'0'
SVSUB2   DC    F'0'
         END   BEGIN
```