Chapter 15
Table Processing

Objectives

Upon completion of this chapter you will be able to:
- Define a table of data,
- Code a sentinel-controlled loop,
- Code a counter-controlled loop,
- Use the `LA` and `BCT` instructions,
- Load a table at run time,
- Use table processing techniques to process a field one character at a time, and
- Use table processing techniques to process look-ups.

Introduction

BAL is probably not the first programming language you have learned. In your other work, you have probably used arrays or tables. You are probably aware of the benefits of table processing capabilities in a programming language. Common applications include:

- When validating (editing) data, we often want to make sure that a field is one of several permissable values. These legitimate values can be placed in a table, and the field in question can be compared to each entry within the table.
- Shipping costs may vary by zone, and each state is assigned a zone number. The state field within each record can be used to search a state/zone table to find the corresponding zone, and that zone could be used to search a zone/rate table to find the appropriate shipping cost.
- A user may request a report (similar to Cogsworths' Sales Recap) with a column showing percent-to-total for each detail. In order to produce such a report, the total must be known. This will require either passing the file once just to determine the total, or storing all of the necessary data in a table which is then printed at EOF when the total is known.

The ability to process tables will require a significant degree of competence in working with registers. In this chapter, we will review table load and table search logic, and introduce the `LA` and `BCT` instructions.

Defining a Table; Sentinel-controlled Loops

Consider the following example: the output from most business programs will contain record counts for audit and reconciliation purposes. The following output is from an update program:

```
Transactions In                          74
Transactions Rejected                     4
Old Masters In                        1,690
Old Masters Deleted                      12
Old Masters Changed                      21
New Masters Added                        37
New Masters Out                       1,715
```

How might we code this using what we already know? One solution would be to define the following fields:

```
#TRANSIN DC    PL4'0'   Transactions In
#REJECTS DC    PL4'0'   Transactions Rejected
#OLDIN   DC    PL4'0'   Old Masters In
#DELETED DC    PL4'0'   Old Masters Deleted
#CHANGED DC    PL4'0'   Old Masters Changed
#ADDED   DC    PL4'0'   New Masters Added
#NEWOUT  DC    PL4'0'   New Masters Out
EDCOUNT  DC    X'40206B2020206B202120'
OLINE    DS    0CL37
ODESC    DS    CL25
OCOUNT   DS    CL10    BZ,ZZZ,ZZ9
OCRLF    DC    X'0D25'
```

...and write the following (redundant) code:

```
MVC   ODESC,=CL25'Transactions In'
MVC   OCOUNT,EDCOUNT
ED    OCOUNT,#TRANSIN'
BAL   R10,WRITE
MVC   ODESC,=CL25'Transactions Rejected'
MVC   OCOUNT,EDCOUNT
ED    OCOUNT,#REJECTS
BAL   R10,WRITE
MVC   ODESC,=CL25'Old Masters In'
MVC   OCOUNT,EDCOUNT
ED    OCOUNT,#OLDIN
:
: etc.
:
MVC   ODESC,=CL25'New Masters Out'
MVC   OCOUNT,EDCOUNT
ED    OCOUNT,#NEWOUT
BAL   R10,WRITE
```

Such redundant code may be fine if you only have seven counts to print as in the above example, but what if you have one or more count *for each state?*

What's needed is some way to repeat the same code but with different counts and labels; that is, some way to process these counts as part of a table, or array. For example, in BASIC, the above could have been done as follows. (This program uses a special character (an asterisk) to indicate the end of the table. When a character is used in such a way it is sometimes referred to as a **sentinel**.)

```
100 REM Initialize Counts - In BASIC, numerics
110 REM are automatically initialized to zero.
120 DIM DESC$(8), COUNT(7)
130 DESC$(1) = "Transactions In"
140 DESC$(2) = "Transactions Rejected"
150 DESC$(3) = "Old Masters In"
160 DESC$(4) = "Old Masters Deleted"
170 DESC$(5) = "Old Masters Changed"
180 DESC$(6) = "New Masters Added"
190 DESC$(7) = "New Masters Out"
200 DESC$(8) = "*"

500 REM Print counts
510 MASK$ = "\                      \ ##,###"
520 LET I = 1
520 WHILE DESC$(I) <> "*"
530     PRINT USING MASK$; DESC$(I), COUNT(I)
540     LET I = I + 1
550 WEND
```

So what, then, is the assembler solution? First, we want to define the fields so the data appears as a table. One solution might be:

0 24	25 28
Transactions In	0
Transactions Rejected	0
Old Masters In	0
Old Masters Deleted	0
Old Masters Changed	0
New Masters Added	0
New Masters Out	0
*	

Each row occupies 29 bytes. The `0`, `24`, `25`, and `28` refer to displacements into a row. The code to define such a table is as follows:

```
COUNTS   DS    0CL29
         DC    CL25'Transactions In'
#TRANSIN DC    PL4'0'
         DC    CL25'Transactions Rejected'
#REJECTS DC    PL4'0'
         DC    CL25'Old Masters In'
#OLDIN   DC    PL4'0'
         DC    CL25'Old Masters Deleted'
#DELETED DC    PL4'0'
         DC    CL25'Old Masters Changed'
#CHANGED DC    PL4'0'
         DC    CL25'New Masters Added'
#ADDED   DC    PL4'0'
         DC    CL25'New Masters Out'
#NEWOUT  DC    PL4'0'
         DC    CL1'*'
```

To increment these counts, we simply use the `AP` command as usual; for example: `AP #TRANSIN,=P'1'`

To print these counts, we use the following code:

```
        LA    R8,COUNTS          R8 POINTS TO FIRST ROW
LOOP    EQU   *
        MVC   ODESC,0(R8)        DESC COMES FROM WHERE R8 POINTS
        MVC   OCOUNT,EDCOUNT     MOVE MASK PRIOR TO EDIT
        ED    OCOUNT,25(R8)      COUNT COMES FROM 25 BYTES PAST
        BAL   R10,PRINT          WHERE R8 POINTS
        AH    R8,=H'29'          POINT TO NEXT ROW
        CLI   0(R8),C'*'         IS FIRST BYTE AN ASTERISK?
        BNE   LOOP               NO - REPEAT UNTIL THEN
```

Explanation:

- `LA R8,COUNTS` Load the address of `COUNTS` into register 8; that is; register 8 points to the first byte of the first entry in the table.
- `MVC ODESC,0(R8)` Move into the description the data to which register 8 is pointing. The zero indicates a displacement of zero; that is, register 8 points directly to the field. Recall that the length of an `MVC` is determined by the length of the receiving field, hence there is no length operator in the second operand.
- `ED OCOUNT,25(R8)` Edit into the count the packed number located 25 bytes past where register 8 is pointing. Recall that the length of an `ED` is determined by the length of the receiving field, hence there is no length operator in the second operand.
- `AH R8,=H'29'` Add 29 to register 8; that is, point register 8 to the next row in the table.
- `CLI 0(R8),C'*'` See if the byte to which register 8 is pointing is an asterisk. Recall that the immediate instructions (`MVI` and `CLI`) do not use a length operator: the length for these instructions is *always* one.

You Try It...

1. What change would be made to the table definition and program segment if the end of the table was indicated by a dollar sign instead of an asterisk? The word `END` instead of an asterisk? `X'FF'` instead of an asterisk? What are these fields (dollar sign, `END`, and `X'FF`) called when used in this fashion?
2. Each count was defined as `PL4`, which means each can hold up to 9,999,999. What changes would be made to the table definition and program segment if each count was changed from `PL4` to `PL3`? Indicate the new range of values for these counts.

Counter-controlled Loops

We could just as well have implemented this as a **counter controlled loop**. In BASIC, this would appear as follows:

```
500 REM Print counts
510 MASK$ = "\                       \ ##,###"
520 FOR I = 1 TO 7
530     PRINT USING MASK$; DESC$(I), COUNT(I)
540 NEXT I
```

The BAL implementation of a counter-controlled loop can be done as follows. The line comments should suffice for explaining the code.

```
         LA    R8,COUNTS         R8 POINTS TO FIRST ROW
         LH    R7,=H'7'          THERE ARE SEVEN ROWS
LOOP     EQU   *
         MVC   ODESC,0(R8)       DESC COMES FROM WHERE R8 POINTS
         MVC   OCOUNT,EDCOUNT    MOVE MASK PRIOR TO EDIT
         ED    OCOUNT,25(R8)     COUNT COMES FROM 25 BYTES PAST
         BAL   R10,PRINT         WHERE R8 POINTS
         AH    R8,=H'29'         POINT TO NEXT ROW
         SH    R7,=H'1'          SUBTRACT ONE FROM ROW COUNTER
         BH    LOOP              IF RESULT > ZERO THEN REPEAT
```

* *

It's useful to note that we could have coded the counts table as follows:

```
COUNTS   DS    0CL29
#TRANSIN DC    PL4'0',CL25'Transactions In'
#REJECTS DC    PL4'0',CL25'Transactions Rejected'
#OLDIN   DC    PL4'0',CL25'Old Masters In'
#DELETED DC    PL4'0',CL25'Old Masters Deleted'
#CHANGED DC    PL4'0',CL25'Old Masters Changed'
#ADDED   DC    PL4'0',CL25'New Masters Added'
#NEWOUT  DC    PL4'0',CL25'New Masters Out'
```

- We have dropped the sentinel in favor of a counter-controlled loop.
- Each of the count fields (`#TRANSIN`, `#REJECTS`, etc.) are considered by the assembler to be four bytes long, as that is the length of the first field on that line (`PL4`). This can be verified by viewing the "stuff on the left" for an instruction referencing these fields.
- The `DS 0CL29` used on `COUNTS` is not required, but is a convenient way to indicate the length of each row. We use a multiplier of zero because `COUNTS` itself does not occupy any space.

Our new solution is as follows:

```
         LA    R8,COUNTS         R8 POINTS TO FIRST ROW
         LA    R7,7              THERE ARE SEVEN ROWS
LOOP     EQU   *
         MVC   ODESC,4(R8)       DESC FROM 4 PAST WHERE R8 POINTS
         MVC   OCOUNT,EDCOUNT    MOVE MASK PRIOR TO EDIT
         ED    OCOUNT,0(R8)      COUNT COMES FROM WHERE R8 POINTS
         BAL   R10,PRINT
         LA    R8,L'COUNTS(R8)   POINT TO NEXT ROW
         BCT   R7,LOOP           SUBTRACT ONE FROM ROW COUNTER
*                                 IF RESULT > ZERO THEN REPEAT
```

Explanation:

- `LA R7,7` — Load into register 7 the address of the seventh byte in memory. But the address of the seventh byte *is* seven, so the number seven is loaded into register 7. This is a convenient way to put a constant into a register.

- `LA R8,L'COUNTS(R8)` — Load into register 8 the address of the field which is (length of `COUNTS`) or (29) bytes past where register 8 is currently pointing. In other words, point to the next row. (This is why we used `0CL29` on `COUNTS` above.)

- `BCT R7,LOOP` — The `BCT` or Branch On Count instruction is very common in table processing. Think of it as the BAL equivalent to a `FOR..NEXT` loop. Each time the `BCT` is executed, the specified register is decremented by one. If the result is greater than zero, then the program branches to the specified label.

You Try It...

3. What change would be made to the table definition and program segment if a new count were added to the table. The field name is `#TRANSOK` and the description is `'Transactions used'`. The new field follow the `#REJECTS` field.
4. Each count was defined as `PL4`. What changes would be made to the table definition and program segment if each count was changed from `PL4` to `PL5`?

Loading a Table at Run-time

In the previous example, the contents of the table (i.e., counts and labels) were known at the time the program was written. It is not uncommon to have to build a table *while the program is running*. Consider the following scenario. Columns 1-6 of a record contain a title (`MR.`, `MRS.`, etc.) I need a count of each title that appears in the file, but I have no idea which titles will be used. (There are, after all, some pretty strange abbreviations for military titles!) I will set aside space for some arbitrary number of title (let's say up to 100.) As each record is read, I will check the table for that title. If it is found, then I will add 1 to the count for that title. If the title is*not* found, then

I will add it to the end of the table, and initialize the count for that title to 1. I will define the table as follows:

```
#TTLS    DC    H'0'
#MAXTTLS DC    H'100'
TITLES   DS    100CL10
*              Positions 1-6 are title, CL6
*              Positions 7-10 are count, PL4
```

The code to search the table (and add to it if necessary) is as follows. (This would likely appear somewhere within the `PROCESS` section of the program.)

```
          LA    R6,TITLES          Point R6 to start of table
          LH    R7,#TTLS           Nbr table entries to R7
          CH    R7,=H'0'           No entries yet?
          BE    NOTFOUND           Then add title to table
LOOK      EQU   *
          CLC   0(6,R6),ITITLE     Is table title EQ input?
          BE    FOUND              Yes - Increment count
          LA    R6,L'TITLES(R6)    No - Point to next row
          BCT   R7,LOOK            Repeat till all entries
          LH    R7,#TTLS           Reload actual nbr entries
NOTFOUND  EQU   *                  Want to add title to table
          CH    R7,#MAXTTLS        Is table already full?
          BNL   TOOMANY            Yes - then message
          MVC   0(6,R6),TITLE      No - then add title
          ZAP   6(4,R6),=P'1'      Initialize count to one
          LA    R7,1(R7)           Increment nbr entries
          STH   R7,#TTLS           and save
          B     LOOKED             Done with new entry
TOOMANY   EQU   *                  Table not large enough
          WTO   'Too many titles for table'
          RETURN
FOUND     EQU   *                  Found title in table
          AP    6(4,R6),=P'1'      Increment count for title
LOOKED    EQU   *                  Title has been processed
```

Explanation:

- `MVC 0(6,R6),ITITLE` Move the title from the input record to the position to which register 6 is pointing. Recall that the length of an `MVC` is determined by the length of the first operand. We must specify a length in the first operand, hence the 6 inside the parenthesis. Remember the following format: `D(L,B)`, where `D` is the displacement, `L` is the length, and `B` is the (base) register.

  Reminder: specify the length only if the format of the instruction requires it. In a previous example, we used `MVC ODESC,4(R8)` to move the description from the table to the output. A length operand here would have been inappropriate; that is, `MVC ODESC,4(25,R8)` is invalid.

- `ZAP 6(4,R6),=P'1'` Move a packed one to the four bytes located six bytes beyond where register 6 is pointing. As in the previous note, `6` is the displacement, `4` is the length, and `R6` is the register. The `ZAP` command requires a length operator for both operands, hence the `4.` (`=P'1'` defaults to a length of one.)

The code to print the table is as follows. (This would likely appear somewhere within the `WRAPUP` section of the program.)

```
            LA    R6,TITLES          Point R6 to start of table
            LH    R7,#TTLS           Nbr table entries to R7
            CH    R7,=H'0'           No entries yet?
            BE    DONE               Then no titles to print
LOOP        EQU   *
            BAL   R10,CHKLNS         Check for full page
            MVC   OTITLE,0(R6)       Move title to output
            MVC   OCOUNT,=X'40206B2020206B202120'
            ED    OCOUNT,6(R6)       Edit count to output
            BAL   R10,WRITE          Write line
            LA    R6,L'TITLES(R6)    Point to next row
            BCT   R7,LOOP            Repeat for all titles
DONE        EQU   *
```

You Try It...

5. What changes would be made to the table definition and program segments to allow up to 200 different titles?
6. What changes would be made to the table definition and program segments to allow for titles up to and including eight bytes in length?
7. What changes would be made to the table definition and program segments if the counts were changed from `PL4'0'` (packed) to `H'0'` (binary)? Note: the table must be halfword aligned. Each count would automatically be halfword aligned since the titles are an even number in length (6, or 8 after previous change).
8. What changes would be made to the table definition and program segments if the number of titles, `#TTLS`, was changed from `H'0'` to `PL2'0'`?

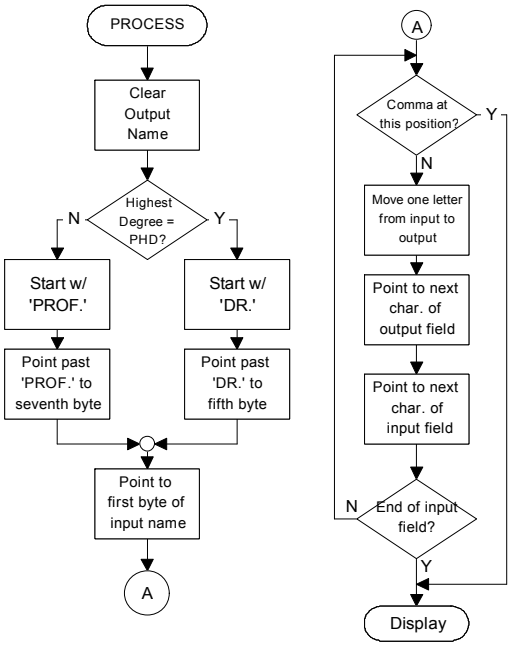
Fields as tables: Processing a field character-by-character

In our next example, we look at how to manipulate a field character-by-character. Recall that the teacher file of the Small Town Community College database shows the teacher name (last name followed by a comma and initials) and the highest degree earned. We would like to process these records as if to produce a mailing label for each teacher; that is, we will use a title of `DR.` (doctor) if the highest degree earned is `PHD`, otherwise we will use a title of `PROF.` (professor). The execution of this program appears as follows:

```
A\:>teach15a
TEACH15A ... Begin execution
TEACH15A ... Input  <BENSON, E.T.   PHD >
TEACH15A ... Output <DR. BENSON              >
TEACH15A ... Input  <HINCKLEY, G.B. MBA >
TEACH15A ... Output <PROF. HINCKLEY          >
TEACH15A ... Input  <KIMBALL, S.W.  PHD >
TEACH15A ... Output <DR. KIMBALL             >
TEACH15A ... Input  <YOUNG, B.      MBA >
TEACH15A ... Output <PROF. YOUNG             >
TEACH15A ... Input  <SMITH, J.      MS  >
TEACH15A ... Output <PROF. SMITH             >
TEACH15A ... Normal end of program
```

The flowchart and code for the `PROCESS` routine as as follows:

```
PROCESS  EQU   *
         ST    R10,SVPROC
         MVC   ONAME,=CL25' '
         CLC   TTDEG,=CL4'PHD'
         BNE   PROFESS
         MVC   ONAME(3),=CL3'DR.'
         LA    R3,ONAME+4
         B     PROC2
PROFESS  EQU   *
         MVC   ONAME(5),=CL5'PROF.'
         LA    R3,ONAME+6
PROC2    EQU   *
         LA    R4,TTNAME
         LA    R5,L'TTNAME
PROC3    EQU   *
         CLI   0(R4),C','
         BE    PROC4
         MVC   0(1,R3),0(R4)
         LA    R3,1(R3)
         LA    R4,1(R4)
         BCT   R5,PROC3
PROC4    EQU   *
         MVC   WTOINAME,TTNAME
         MVC   WTOIDEG,TTDEG
         WTO   WTOMSG1
         MVC   WTOONAME,ONAME
         WTO   WTOMSG2
         BAL   R10,READTCH
PROCESSX EQU   *
         L     R10,SVPROC
         BR    R10
```



The complete program, `TEACH15A.MLC`, follows.

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  TEACH15A.MLC                                *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  List of teachers with title, such as        *
*                   'DR. BENSON' or 'PROF. HINCKLEY'.           *
*****************************************************************
```

```
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TEACH15A ... Begin execution'
         BAL   R10,SETUP
MAIN     EQU   *
         CLI   EOFTEACH,C'Y'
         BE    EOJ
         BAL   R10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   R10,WRAPUP
         WTO   'TEACH15A ... 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
         OPEN  TEACHERS
         BAL   R10,READTCH
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS  EQU   *
         ST    R10,SVPROC
         MVC   ONAME,=CL25' '
         CLC   TTDEG,=CL4'PHD'    Is highest degree PHD?
         BNE   PROFESS            No, then title is 'PROF.'
         MVC   ONAME(3),=CL3'DR.' else title is 'DR.'
         LA    R3,ONAME+4         Will move name starting at
         B     PROC2              the fifth byte.
PROFESS  EQU   *
         MVC   ONAME(5),=CL5'PROF.'
         LA    R3,ONAME+6         Will move name starting at
PROC2    EQU   *                  the seventh byte.
         LA    R4,TTNAME          Point R4 to 1st byte of input
         LA    R5,L'TTNAME        R5 is max nbr characters
PROC3    EQU   *
         CLI   0(R4),C','         Find comma separating name
         BE    PROC4              from initials? Yes, done.
         MVC   0(1,R3),0(R4)      Else move this letter.
         LA    R3,1(R3)           Point to next letter output
         LA    R4,1(R4)           Point to next letter input
         BCT   R5,PROC3           Repeat till done
PROC4    EQU   *
         MVC   WTOINAME,TTNAME    Show input name
         MVC   WTOIDEG,TTDEG      Show input degree
         WTO   WTOMSG1
         MVC   WTOONAME,ONAME     Show output name
         WTO   WTOMSG2
         BAL   R10,READTCH
PROCESSX EQU   *
         L     R10,SVPROC
         BR    R10
```

(continued)

```
*****************************************************************
*        READTCH - Read a teacher record.                       *
*****************************************************************
READTCH  EQU   *
         ST    R10,SVREADT
         GET   TEACHERS,TREC      Read a single teacher record
         B     READTX
ATENDTCH EQU   *
         MVI   EOFTEACH,C'Y'
READTX   EQU   *
         L     R10,SVREADT
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE TEACHERS
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
TEACHERS DCB   LRECL=29,RECFM=F,MACRF=G,EODAD=ATENDTCH,
               DDNAME='TEACHER.DAT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP  DC    F'0'               SETUP
SVPROC   DC    F'0'               PROCESS
SVREADT  DC    F'0'               READTCH
SVWRAP   DC    F'0'               WRAPUP
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
EOFTEACH DC    CL1'N'             End of teacher file? (Y/N)
ONAME    DC    CL25' '            Name with 'DR.' or 'PROF.'
WTOMSG1  DS    0CL41
         DC    CL21'TEACH15A ... Input  <'
WTOINAME DS    CL15
WTOIDEG  DS    CL4
         DC    CL1'>'
WTOMSG2  DS    0CL47
         DC    CL21'TEACH15A ... Output <'
WTOONAME DS    CL25
         DC    CL1'>'
*****************************************************************
*        Input record definition - Teacher                      *
*****************************************************************
TREC     DS    0CL29      1-29    Teacher record
TTID     DS    CL3        1- 3    Teacher ID nbr
TTNAME   DS    CL15       4-18    Teacher name
TTDEG    DS    CL4       19-22    Highest degree
TTTEN    DS    CL1       23-23    Tenured?
TTPHONE  DS    CL4       24-27    Phone nbr
TTCRLF   DS    CL2       28-29    PC/370 only - CR/LF
         END   BEGIN
```

Sample Program - Producing Transcripts

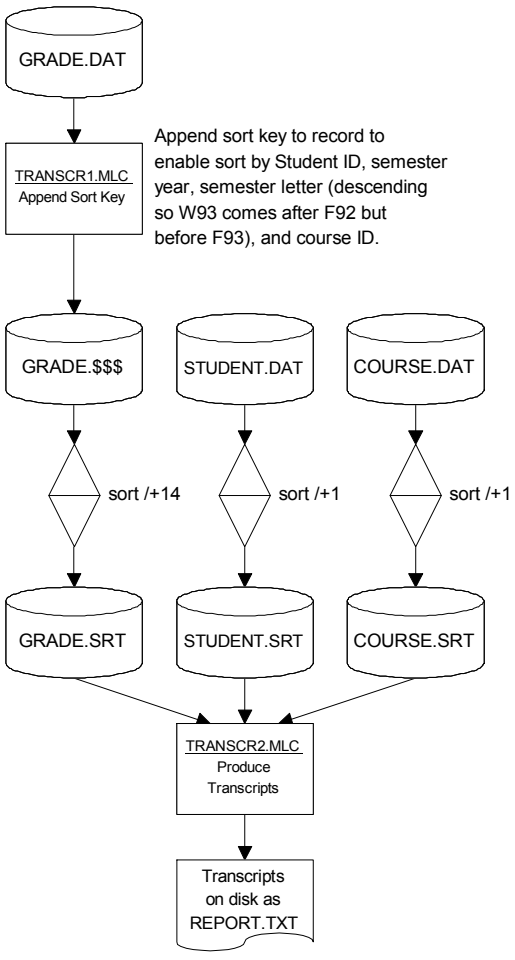
We will include two other programs in this chapter. Our objective is to produce transcripts for all students at the Small Town Community College. In order to do so, we will need the following fields from the following files:

STUDENT
Student ID
Student name

GRADE
Student ID
Semester
Course ID
Grade

COURSE
Course ID
Description
Hours

The transcripts will be produced in student ID sequence, so the `STUDENT` and `GRADE` files will need to be sorted accordingly. Furthermore, for each student, the courses will be listed in the sequence in which they were taken. This is a problem. The semester consists of a letter and a year; for example: `F92`. But fall 1992 comes after winter 1992, so we need the semester to be sorted on the year portion ascending, then the season portion descending. This is not a problem with a decent sort package, but we can't do it with DOS' `SORT` command. So we will need to write a program which appends a sort key to the `GRADE` file. The semester will be codified: `W92` will be `921` and `F92` will be `922`. We will also include the course ID in this sort key so that within each semester, the courses are listed in sequence. For each course we will want the course description. We will also need the course hours in order to determine hours attempted and grade point averages. Therefore we will sort the `COURSE` file by course ID and load it into a table in the transcript program.



Here is a list of the commands to produce the transcripts, as well as the transcripts themselves.

```
A:\MIN>transcr1
TRANSCR1 ... Begin execution
TRANSCR1 ...  22 grade records reformatted.
TRANSCR1 ... Normal end of program
A:\MIN>sort /+14 < grade.$$$ > grade.srt
A:\MIN>sort /+1 < student.dat > student.srt
A:\MIN>sort /+1 < course.dat > course.srt
A:\MIN>transcr2
TRANSCR2 ... Begin execution
TRANSCR2 ... Transcripts on REPORT.TXT
TRANSCR2 ... Normal end of program
A:\MIN>type report.txt
```

```
_____________________________________________________________________
        TRANSCRIPT FOR (125) MORALES, L.A.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
  F92     MA101   ALGEBRA           F       3      0
  W93     MA101   ALGEBRA           C       3      6
--------  ------  --------------  -----  -----  ------
                                  TOTAL     6      6
                                  GPA    1.00

_____________________________________________________________________
        TRANSCRIPT FOR (263) HAVLIK, K.M.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
  W92     PE151   AEROBICS          C       1      2
  F92     PE151   AEROBICS          B       1      3
  W93     PE151   AEROBICS          A       1      4
--------  ------  --------------  -----  -----  ------
                                  TOTAL     3      9
                                  GPA    3.00

_____________________________________________________________________
        TRANSCRIPT FOR (402) FOOTE, A.K.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
                                  TOTAL     0      0
                                  GPA    0.00

_____________________________________________________________________
        TRANSCRIPT FOR (421) QUALLS, G.E.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
  F92     EG101   ENGLISH I         B       3      9
  W93     EG102   ENGLISH II        A       3     12
--------  ------  --------------  -----  -----  ------
                                  TOTAL     6     21
                                  GPA    3.50

_____________________________________________________________________
        TRANSCRIPT FOR (626) MERCIER, J.L.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
  W92     EG102   ENGLISH II        A       3     12
--------  ------  --------------  -----  -----  ------
                                  TOTAL     3     12
                                  GPA    4.00
```

```
_____________________________________________________________
        TRANSCRIPT FOR (701) ARIAS, I.L.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
  F92     MA101   ALGEBRA           B       3      9
  F92     PE151   AEROBICS          A       1      4
  W93     MA107   STATISTICS        D       3      3
--------  ------  --------------  -----  -----  ------
                                  TOTAL     7     16
                                  GPA    2.29
_____________________________________________________________
        TRANSCRIPT FOR (713) HILMER, D.R.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
  F92     EG101   ENGLISH I         C       3      6
  F92     MA101   ALGEBRA           B       3      9
  W93     EG102   ENGLISH II        B       3      9
  W93     MA107   STATISTICS        B       3      9
--------  ------  --------------  -----  -----  ------
                                  TOTAL    12     33
                                  GPA    2.75
_____________________________________________________________
        TRANSCRIPT FOR (896) QUALLS, D.M.

Semester  Course    Description    Grade  Hours  Points
--------  ------  --------------  -----  -----  ------
  W92     PE151   AEROBICS          A       1      4
  F92     AC101   ACCOUNTING        C       3      6
  F92     BU101   BUSINESS          C       3      6
  F92     EG101   ENGLISH I         A       3     12
  F92     MA101   ALGEBRA           B       3      9
  W93     EG102   ENGLISH II        B       3      9
  W93     MA107   STATISTICS        A       3     12
--------  ------  --------------  -----  -----  ------
                                  TOTAL    19     58
                                  GPA    3.05
```

Following is a list of program `TRANSCR1.MLC`, the program to append the sort key to the `GRADE` file. There's nothing new here. There's no need for explanation: it is included for completeness.

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  TRANSCR1.MLC                                *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  Reformat grade file so it can be sorted     *
*                   properly so as to produce transcripts.      *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TRANSCR1 ... Begin execution'
         BAL   R10,SETUP
```

(continued)

```
MAIN     EQU   *
         CLI   EOFGRADE,C'Y'
         BE    EOJ
         BAL   R10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   R10,WRAPUP
         WTO   'TRANSCR1 ... Normal end of program'
RETURN   EQU   *
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                before any records are processed.              *
*****************************************************************
SETUP    EQU   *
         ST    R10,SVSETUP
         OI    GRADEIN+10,X'08'   PC/370 ONLY - Convert all
*                                  input from ASCII to EBCDIC
         OI    GRADEOUT+10,X'08'  PC/370 ONLY - Convert all
*                                  output from EBCDIC to ASCII
         OPEN  GRADEIN
         OPEN  GRADEOUT
         BAL   R10,READGRAD       Priming read - GRADEIN
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS  EQU   *
         ST    R10,SVPROC
         BAL   R10,FORMAT
         BAL   R10,WRITE
         BAL   R10,READGRAD
PROCESSX EQU   *
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        FORMAT - Format a single record, with sort key.        *
*****************************************************************
*        Copy the entire record, with sort key appended.        *
*        Sort key consists of student ID, year portion of       *
*        semester, '1' for (W)inter or '2' for (F)all, and      *
*        the course ID. This appended sort key is used to       *
*        overcome limitations of DOS' SORT command, which       *
*        allows a single sort field only. The recoding of       *
*        the semester was done so Fall classes will come        *
*        after winter classes. For example, semester 'W92'      *
*        becomes '921', while semester 'F92' becomes '922'.     *
*****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORMAT
         MVC   SORTDATA,GREC
         MVC   SORTSID,GSID
         MVC   SORTSEM(2),GSEM+1
         CLI   GSEM,C'F'
         BNE   FORMAT2
         MVI   SORTSEM+2,C'2'
         B     FORMAT3
```

(continued)

```
FORMAT2  EQU   *
         MVI   SORTSEM+2,C'1'
FORMAT3  EQU   *
         MVC   SORTCID,GCID
         MVC   SORTCRLF,WCRLF
FORMATX  EQU   *
         L     R10,SVFORMAT
         BR    R10
*****************************************************************
*        READGRAD - Read a Grade record.                        *
*****************************************************************
READGRAD EQU   *
         ST    R10,SVREADG
         GET   GRADEIN,GREC
         B     READGX
ATENDGRA EQU   *
         MVI   EOFGRADE,C'Y'
READGX   EQU   *
         L     R10,SVREADG
         BR    R10
*****************************************************************
*        WRITE - Write a single output record.                  *
*****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   GRADEOUT,SORTREC
         AP    #OUT,=P'1'
         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 GRADEIN
         CLOSE GRADEOUT
         ED    MSG#OUT,#OUT
         WTO   MSG
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*        Note: $$$ is common DOS extension for temporary file   *
*****************************************************************
GRADEIN  DCB   LRECL=15,RECFM=F,MACRF=G,EODAD=ATENDGRA,
               DDNAME='GRADE.DAT'
GRADEOUT DCB   LRECL=26,RECFM=F,MACRF=P,
               DDNAME='GRADE.$$$'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP  DC    F'0'               SETUP
SVPROC   DC    F'0'               PROCESS
SVREADG  DC    F'0'               READCOUR
```

(continued)

```
SVWRITE  DC    F'0'              WRITE
SVWRAP   DC    F'0'              WRAPUP
SVFORMAT DC    F'0'              FORMAT
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'           PC/370 ONLY - EBCDIC CR/LF
EOFGRADE DC    CL1'N'            End of grades file? (Y/N)
#OUT     DC    PL2'0'
MSG      DS    0CL43
         DC    CL12'TRANSCR1 ...'
MSG#OUT  DC    XL4'40202120'
         DC    CL27' grade records reformatted.'
*****************************************************************
*        Input record definition - Grade                        *
*****************************************************************
GREC     DS    0CL15      1-15   Grade record
GSID     DS    CL3        1- 3   Student ID nbr
GSEM     DS    CL3        4- 6   Semester
GCID     DS    CL5        7-11   Course ID nbr
GSECT    DS    CL1       12-12   Section number
GGRADE   DS    CL1       13-13   Grade earned
GGCRLF   DS    CL2       14-15   PC/370 only - CR/LF
*****************************************************************
*        Output record definition - Grade w/ sort key           *
*****************************************************************
SORTREC  DS    0CL26      1-26   Sort record
SORTDATA DS    CL13       1-13   Grade record (without CRLF)
SORTKEY  DS    0CL11     14-24   Sort key, including:
SORTSID  DS    CL3       14-16   Student ID nbr
SORTSEM  DS    CL3       17-19   Semester (recoded)
SORTCID  DS    CL5       20-24   Course ID nbr
SORTCRLF DS    CL2       25-26   PC/370 only - CR/LF
         END   BEGIN
```

Input to TRANSCR1.MLC: GRADE.DAT

```
626W92EG1021A
896W92PE1511A
263W92PE1511C
896F92AC1011C
896F92BU1011C
896F92EG1011A
713F92EG1012C
421F92EG1012B
713F92MA1011B
896F92MA1011B
125F92MA1012F
701F92MA1012B
263F92PE1511B
701F92PE1511A
713W93EG1021B
421W93EG1021A
896W93EG1021B
125W93MA1011C
713W93MA1071B
896W93MA1071A
701W93MA1071D
263W93PE1511A
```

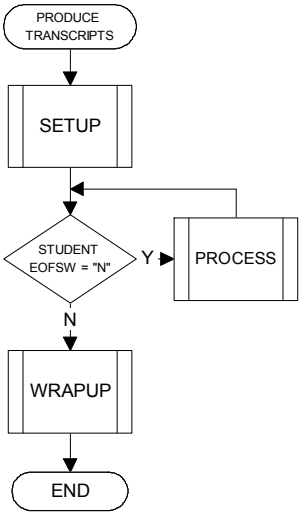
Output from TRANSCR1.MLC: GRADE.$$$
Input to TRANSCR2.MLC

```
626W92EG1021A626921EG102
896W92PE1511A896921PE151
263W92PE1511C263921PE151
896F92AC1011C896922AC101
896F92BU1011C896922BU101
896F92EG1011A896922EG101
713F92EG1012C713922EG101
421F92EG1012B421922EG101
713F92MA1011B713922MA101
896F92MA1011B896922MA101
125F92MA1012F125922MA101
701F92MA1012B701922MA101
263F92PE1511B263922PE151
701F92PE1511A701922PE151
713W93EG1021B713931EG102
421W93EG1021A421931EG102
896W93EG1021B896931EG102
125W93MA1011C125931MA101
713W93MA1071B713931MA107
896W93MA1071A896931MA107
701W93MA1071D701931MA107
263W93PE1511A263931PE151
```

Following are the flowcharts and code for TRANSCR2.MLC. This program produces the transcripts. There are, by design, several different examples of table processing in the program.

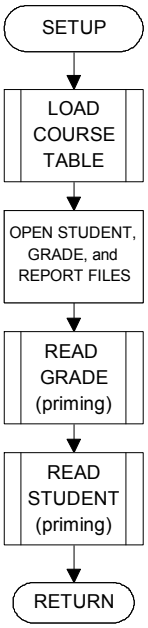
Mainline Section

The Mainline section is the same as we've seen so many times before...




```
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TRANSCR2 ... Begin execution'
         BAL   R10,SETUP
MAIN     EQU   *
         CLI   EOFSTUD,C'Y'
         BE    EOJ
         BAL   R10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   R10,WRAPUP
         WTO   'TRANSCR2 ... Normal end of
                  program'
RETURN   EQU   *
         RETURN
```

The SETUP Routine



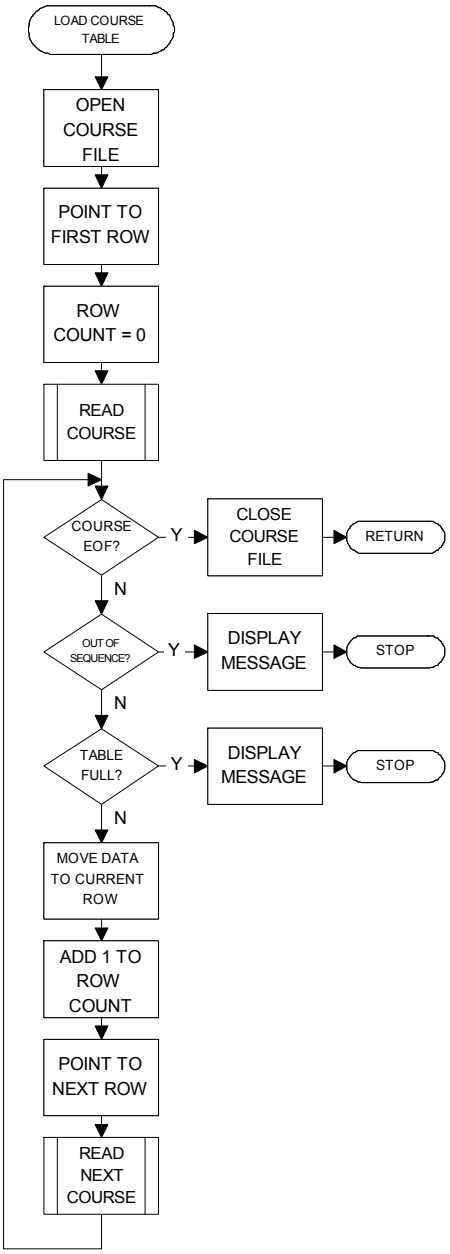
The SETUP routine is very similar to what we've seen before. It includes OPENs and priming reads. What's new is logic to load the course data from the COURSE file into an internal table. That logic is the object of the next routine...

```
SETUP    EQU   *
         ST    R10,SVSETUP
         BAL   R10,LOADTBL
         OI    STUDENT+10,X'08'
         OI    GRADE+10,X'08'
         OI    REPORT+10,X'08'
         OPEN  STUDENT
         OPEN  GRADE
         OPEN  REPORT
         BAL   R10,READSTUD
         BAL   R10,READGRAD
         L     R10,SVSETUP
         BR    R10
```

Loading the Course Table

Before any student records are processed, we load all COURSE records into an internal table. We verify that the courses are in course number sequence and that there is enough room to hold all courses.

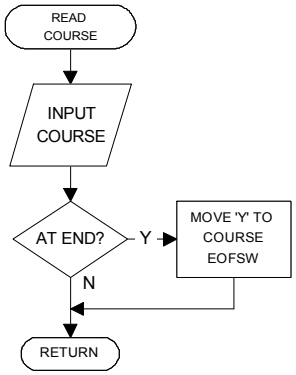
```
LOADTBL  EQU   *
         ST    R10,SVLOAD
         OI    COURSE+10,X'08'
         OPEN  COURSE
         LA    R3,TABLE
         SR    R4,R4
LOADTBL2 EQU   *
         BAL   R10,READCOUR
         CLI   EOFCOURS,C'Y'
         BE    LOADTBLX
         CLC   PREVCID,CCID
         BNL   LOADTBL3
         MVC   PREVCID,CCID
         CH    R4,MAXROWS
         BNL   LOADTBL4
         MVC   0(5,R3),CCID
         MVC   5(15,R3),CCDESC
         PACK  20(1,R3),CCHRS
         LA    R3,L'TABLE(R3)
         LA    R4,1(R4)
         B     LOADTBL2
LOADTBL3 EQU   *
         WTO   'TRANSCR2 ... Course file
                  not sequenced by CID'
         B     RETURN
LOADTBL4 EQU   *
         WTO   'TRANSCR2 ... Nbr of
                  courses exceeds table
                  size'
         B     RETURN
LOADTBLX EQU   *
         STH   R4,ROWS
         CLOSE COURSE
         L     R10,SVLOAD
         BR    R10
```

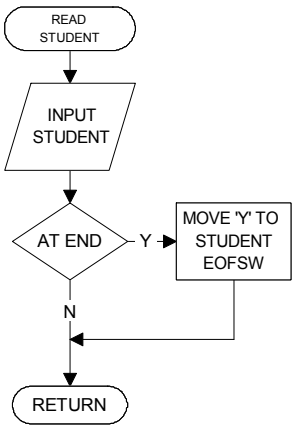


The READ Routines

There are three input files: courses, students, and grades. The READ routines for all three are very similar...

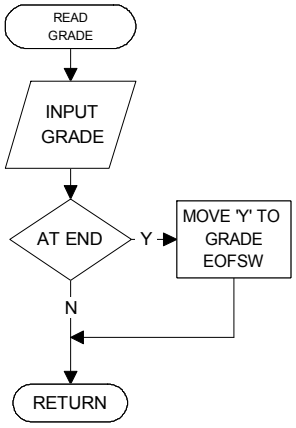
```
READCOUR EQU   *
         ST    R10,SVREADC
         GET   COURSE,CREC
         B     READCX
ATENDCRS EQU   *
         MVI   EOFCOURS,C'Y'
READCX   EQU   *
         L     R10,SVREADC
         BR    R10
```





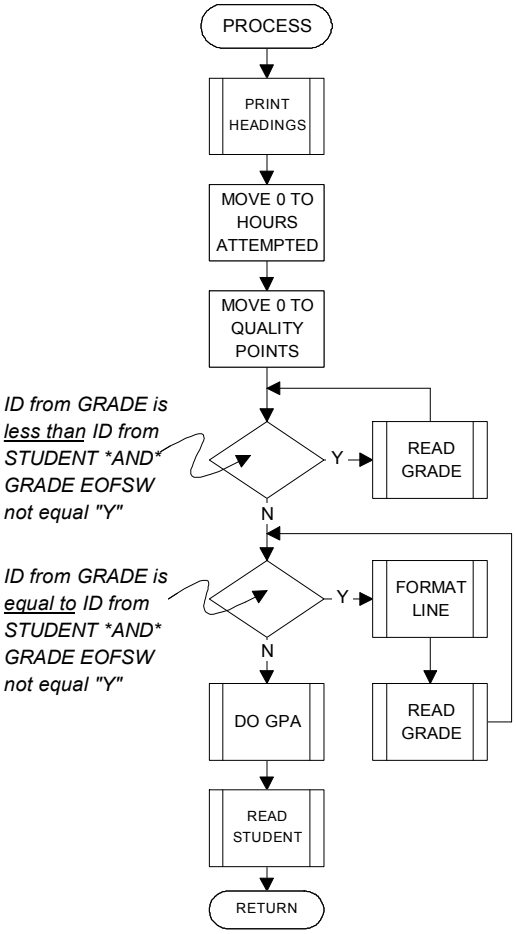

```
READSTUD EQU   *
         ST    R10,SVREADS
         GET   STUDENT,SREC
         B     READSX
ATENDSTU EQU   *
         MVI   EOFSTUD,C'Y'
READSX   EQU   *
         L     R10,SVREADS
         BR    R10
```

```
READGRAD EQU   *
         ST    R10,SVREADG
         GET   GRADE,GREC
         B     READGX
ATENDGRA EQU   *
         MVI   EOFGRADE,C'Y'
READGX   EQU   *
         L     R10,SVREADG
         BR    R10
```



The PROCESS Routine

Upon entry to the PROCESS routine, we have a single STUDENT record. This routine will process all GRADE records for that student. We begin by producing the transcript heading. For each GRADE record, we format a single transcript line and accumulate hours attempted and quality points (see FORMAT routine below). After processing all GRADE records for this student, we calculate and print the GPA. We then read the next STUDENT record before returning to the mainline.

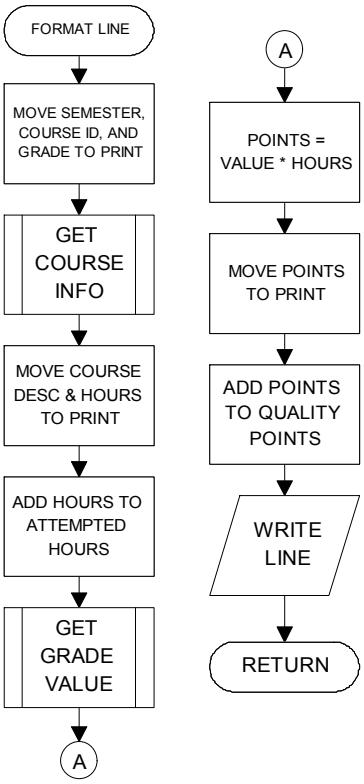



```
PROCESS  EQU   *
         ST    R10,SVPROC
         BAL   R10,HDGS
         ZAP   ATTEMPT,=P'0'
         ZAP   QUALITY,=P'0'
PROC2    EQU   *
         CLI   EOFGRADE,C'Y'
         BE    PROC3
         CLC   GSID,SSID
         BNL   PROC3
         BAL   R10,READGRAD
         B     PROC2
PROC3    EQU   *
         CLI   EOFGRADE,C'Y'
         BE    PROC4
         CLC   GSID,SSID
         BNE   PROC4
         BAL   R10,FORMAT
         BAL   R10,READGRAD
         B     PROC3
PROC4    EQU   *
         BAL   R10,DOGPA
         BAL   R10,READSTUD
PROCESSX EQU   *
         L     R10,SVPROC
         BR    R10
```

```
HDGS     EQU   *
         ST    R10,SVHDGS
         MVC   HDSID,SSID
         MVC   HDSNAME,SSNAME
         PUT   REPORT,FORMFEED
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         L     R10,SVHDGS
         BR    R10
```

The FORMAT Routine

Upon entry to the FORMAT routine, we have a single GRADE record for a single STUDENT. We move the semester, course ID, and grade to the output area. We then use the course ID to find the course description and hours (see GET COURSE INFO below). We add the course hours to the total hours attempted for this student. We then determine the quality points based on the grade (see GET GRADE VALUE below). We move the quality points to the output area and add these to the total quality points for this student. We then write the transcript line.

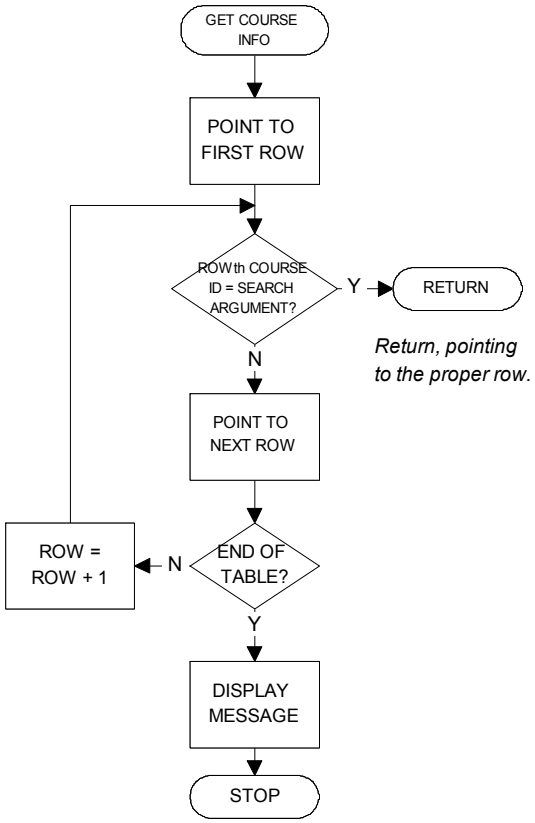



```
FORMAT   EQU   *
         ST    R10,SVFORMAT
         MVC   OSEM,GSEM
         MVC   OCID,GCID
         MVC   OGRADE,GGRADE
         BAL   R10,CRSDATA
         MVC   OCDESC,5(R3)
         ZAP   PK2,20(1,R3)
         MVC   OCHRS,=X'40202120'
         ED    OCHRS,PK2
         AP    ATTEMPT,PK2
         BAL   R10,GRADEVAL
         ZAP   PK4,PK2
         MP    PK4,VALUE
         MVC   OPOINTS,=X'40202120'
         ED    OPOINTS,PK4+2
         AP    QUALITY,PK4
         MVC   OCRLF,WCRLF
         BAL   R10,WRITE
FORMATX  EQU   *
         L     R10,SVFORMAT
         BR    R10
```

```
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   REPORT,OREC
         L     R10,SVWRITE
         BR    R10
```

GET COURSE INFO Routine

This routine uses a table lookup to determine the course description and hours for a given course ID. Note: this lookup uses a counter controlled loop.



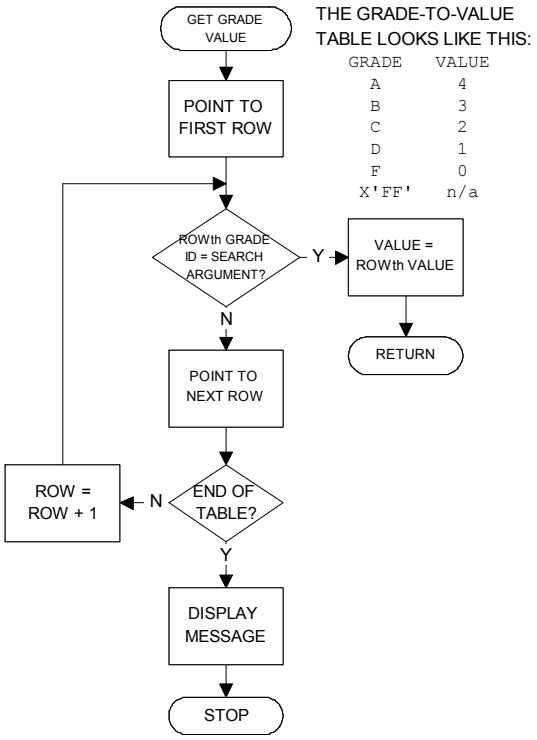

```
CRSDATA  EQU   *
         ST    R10,SVCRSDAT
         LA    R3,TABLE
         LH    R4,ROWS
CRSDATA2 EQU   *
         CLC   GCID,0(R3)
         BE    CRSDATAX
         LA    R3,L'TABLE(R3)
         BCT   R4,CRSDATA2
         WTO   'TRANSCR2 ... Bad
                course ID in
                Grade file'
         B     RETURN
CRSDATAX EQU   *
         L     R10,SVCRSDAT
         BR    R10
```

where...

```
TABLE    DS    10CL21
*   1- 5    Course ID
*   6-20    Course description
*  21-21    Hours (packed)
```

GET GRADE VALUE Routine

This routine uses a table lookup to determine the quality points for a given grade. Note: this lookup uses a sentinel controlled loop.



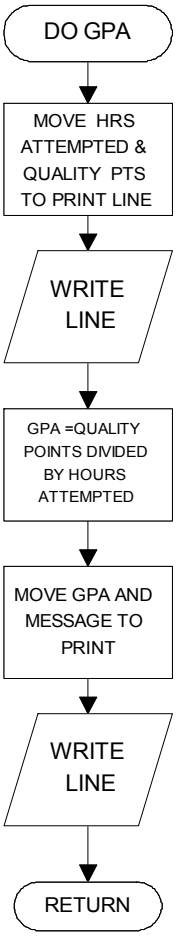

```
GRADEVAL EQU   *
         ST    R10,SVGRDVAL
         LA    R7,GRADETBL
GRADEVA2 EQU   *
         CLC   0(1,R7),GGRADE
         BE    GRADEVAX
         LA    R7,L'GRADETBL(R7)
         CLI   0(R7),X'FF'
         BNE   GRADEVA2
         WTO   'TRANSCR2 ...
                 Invalid grade
                 in grade file'
         B     RETURN
GRADEVAX EQU   *
         ZAP   VALUE,1(1,R7)
         L     R10,SVGRDVAL
         BR    R10
```

where...

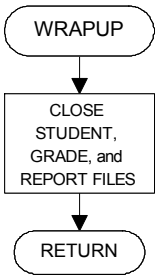
```
VALUE    DC    PL1'0'
GRADETBL DS    0CL2
         DC    CL1'A',PL1'4'
         DC    CL1'B',PL1'3'
         DC    CL1'C',PL1'2'
         DC    CL1'D',PL1'1'
         DC    CL1'F',PL1'0'
         DC    X'FF'
```

DO GPA Routine

This routine moves total hours attempted and total quality points for this student to the transcript, determines the GPA by dividing total quality points by total hours attempted, and moves GPA to the transcript.




```
DOGPA     EQU   *
          ST    R10,SVDOGPA
          CP    ATTEMPT,=P'0'
          BE    DOGPA2
          PUT   REPORT,HD4
DOGPA2    EQU   *
          MVC   TATTEMPT,=X'40202120'
          ED    TATTEMPT,ATTEMPT
          MVC   TQUALITY,=X'40202120'
          ED    TQUALITY,QUALITY
          PUT   REPORT,TREC
          ZAP   DIVIDEND,QUALITY
          SRP   DIVIDEND,3,0
          ZAP   DIVISOR,ATTEMPT
          BZ    DOGPA3
          DP    DIVIDEND,DIVISOR
          SRP   QUOTIENT,64-1,5
          ZAP   PK3,QUOTIENT
          B     DOGPA4
DOGPA3    EQU   *
          ZAP   PK3,=P'0'
DOGPA4    EQU   *
          MVC   WK7,=X'402021204B2020'
          ED    WK7,PK3
          MVC   AGPA,WK7+2
          PUT   REPORT,AREC
DOGPAX    EQU   *
          L     R10,SVDOGPA
          BR    R10
```



The WRAPUP Routine

When transcripts for all students have been printed, all open files are closed.

```
WRAPUP    EQU   *
          ST    R10,SVWRAP
          CLOSE STUDENT
          CLOSE GRADE
          CLOSE REPORT
          WTO   'TRANSCR2 ... Transcripts on
                  REPORT.TXT'
          L     R10,SVWRAP
          BR    R10
```

For completeness, the fully annotated source code for program `TRANSCR2.MLC` follows:

```
         PRINT NOGEN
*****************************************************************
*        FILENAME:  TRANSCR2.MLC                                *
*        AUTHOR  :  Bill Qualls                                 *
*        SYSTEM  :  PC/370 R4.2                                 *
*        REMARKS :  Produce transcipts.                         *
*                   This program illustrates table logic.       *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TRANSCR2 ... Begin execution'
         BAL   R10,SETUP
MAIN     EQU   *
         CLI   EOFSTUD,C'Y'
         BE    EOJ
         BAL   R10,PROCESS
         B     MAIN
EOJ      EQU   *
         BAL   R10,WRAPUP
         WTO   'TRANSCR2 ... Normal end of program'
RETURN   EQU   *
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                before any records are processed.              *
*****************************************************************
SETUP    EQU   *
         ST    R10,SVSETUP
         BAL   R10,LOADTBL
         OI    STUDENT+10,X'08'   PC/370 ONLY - Convert all
*                                 input from ASCII to EBCDIC
         OI    GRADE+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  STUDENT
         OPEN  GRADE
         OPEN  REPORT
         BAL   R10,READSTUD       Priming read - STUDENT
         BAL   R10,READGRAD       Priming read - GRADE
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        LOADTBL - Load all courses into table. Verify:         *
*           (1) courses are in course nbr sequence, and         *
*           (2) there is enough room in table for all.          *
*****************************************************************
LOADTBL  EQU   *
         ST    R10,SVLOAD
         OI    COURSE+10,X'08'    PC/370 ONLY - Convert all
*                                 input from ASCII to EBCDIC
         OPEN  COURSE
         LA    R3,TABLE           Point to start of table
         SR    R4,R4              Initialize row count to zero
LOADTBL2 EQU   *
         BAL   R10,READCOUR       Read single course record
```

(continued)

```
          CLI   EOFCOURS,C'Y'      At end?
          BE    LOADTBLX           Yes - Load complete.
          CLC   PREVCID,CCID       Sequence check
          BNL   LOADTBL3           Fatal error...
          MVC   PREVCID,CCID       Save course ID for seq check
          CH    R4,MAXROWS         Table full already?
          BNL   LOADTBL4           Yes - Fatal error...
          MVC   0(5,R3),CCID       Move course ID nbr,
          MVC   5(15,R3),CCDESC     course description, and
          PACK  20(1,R3),CCHRS       course hours to row.
          LA    R3,L'TABLE(R3)     Point to next row
          LA    R4,1(R4)           Increment row count
          B     LOADTBL2           Repeat
LOADTBL3  EQU   *
          WTO   'TRANSCR2 ... Course file not sequenced by CID'
          B     RETURN
LOADTBL4  EQU   *
          WTO   'TRANSCR2 ... Nbr of courses exceeds table size'
          B     RETURN
LOADTBLX  EQU   *
          STH   R4,ROWS
          CLOSE COURSE
          L     R10,SVLOAD
          BR    R10
*****************************************************************
*         HDGS - Print headings.                                *
*****************************************************************
HDGS      EQU   *
          ST    R10,SVHDGS
          MVC   HDSID,SSID         Move student ID to first hdg
          MVC   HDSNAME,SSNAME     Move student name to first hdg
          PUT   REPORT,FORMFEED    PC/370 ONLY
          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
          BAL   R10,HDGS           Start student on a new page
          ZAP   ATTEMPT,=P'0'      Init hrs attempted to zero
          ZAP   QUALITY,=P'0'      Init quality pts to zero
PROC2     EQU   *                  Check for student ID found
          CLI   EOFGRADE,C'Y'      on GRADE but not on STUDENT.
          BE    PROC3              This is a serious error,
          CLC   GSID,SSID          but for this program we will
          BNL   PROC3              just skip all such records.
          BAL   R10,READGRAD
          B     PROC2
PROC3     EQU   *                  Process all grades records
          CLI   EOFGRADE,C'Y'      for the current student.
          BE    PROC4
          CLC   GSID,SSID
          BNE   PROC4
          BAL   R10,FORMAT
          BAL   R10,READGRAD
          B     PROC3
```

(continued)

```
PROC4    EQU   *                  Having processed any and all
         BAL   R10,DOGPA          grades records for the current
         BAL   R10,READSTUD       student, do GPA line and read
PROCESSX EQU   *                  the next student.
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        FORMAT - Format a single transcript line.              *
*****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORMAT
         MVC   OSEM,GSEM          Move semester,
         MVC   OCID,GCID           course ID nbr, and
         MVC   OGRADE,GGRADE         grade earned to output
         BAL   R10,CRSDATA        Find course data in table
         MVC   OCDESC,5(R3)       Course desc comes from table
         ZAP   PK2,20(1,R3)
         MVC   OCHRS,=X'40202120'
         ED    OCHRS,PK2          Course hours comes from table
         AP    ATTEMPT,PK2        Accumulate hours attempted
         BAL   R10,GRADEVAL
         ZAP   PK4,PK2
         MP    PK4,VALUE
         MVC   OPOINTS,=X'40202120'
         ED    OPOINTS,PK4+2
         AP    QUALITY,PK4
         MVC   OCRLF,WCRLF
         BAL   R10,WRITE
FORMATX  EQU   *
         L     R10,SVFORMAT
         BR    R10
*****************************************************************
*        CRSDATA - Find course data in table                    *
*****************************************************************
CRSDATA  EQU   *
         ST    R10,SVCRSDAT
         LA    R3,TABLE
         LH    R4,ROWS
CRSDATA2 EQU   *
         CLC   GCID,0(R3)
         BE    CRSDATAX
         LA    R3,L'TABLE(R3)
         BCT   R4,CRSDATA2
         WTO   'TRANSCR2 ... Bad course ID in Grade file'
         B     RETURN
CRSDATAX EQU   *
         L     R10,SVCRSDAT
         BR    R10
*****************************************************************
*        GRADEVAL - Find point value for grade                  *
*****************************************************************
GRADEVAL EQU   *
         ST    R10,SVGRDVAL
         LA    R7,GRADETBL          Point to start of table
GRADEVA2 EQU   *
         CLC   0(1,R7),GGRADE       Compare grade to the grade
         BE    GRADEVAX             in table. If equal, done.
         LA    R7,L'GRADETBL(R7)    Else point to next row
         CLI   0(R7),X'FF'          See if at end of table.
         BNE   GRADEVA2             No, repeat.
```

(continued)

```
         WTO   'TRANSCR2 ... Invalid grade in grade file'
         B     RETURN              Fatal error...
GRADEVAX EQU   *
         ZAP   VALUE,1(1,R7)       Save grade value
         L     R10,SVGRDVAL
         BR    R10
*****************************************************************
*        DOGPA - Calculate and format GPA                       *
*****************************************************************
DOGPA    EQU   *
         ST    R10,SVDOGPA
         CP    ATTEMPT,=P'0'
         BE    DOGPA2
         PUT   REPORT,HD4
DOGPA2   EQU   *
         MVC   TATTEMPT,=X'40202120'
         ED    TATTEMPT,ATTEMPT
         MVC   TQUALITY,=X'40202120'
         ED    TQUALITY,QUALITY
         PUT   REPORT,TREC
         ZAP   DIVIDEND,QUALITY
         SRP   DIVIDEND,3,0
         ZAP   DIVISOR,ATTEMPT
         BZ    DOGPA3
         DP    DIVIDEND,DIVISOR
         SRP   QUOTIENT,64-1,5
         ZAP   PK3,QUOTIENT
         B     DOGPA4
DOGPA3   EQU   *
         ZAP   PK3,=P'0'
DOGPA4   EQU   *
         MVC   WK7,=X'402021204B2020'
         ED    WK7,PK3
         MVC   AGPA,WK7+2
         PUT   REPORT,AREC
DOGPAX   EQU   *
         L     R10,SVDOGPA
         BR    R10
*****************************************************************
*        READSTU - Read a student record.                       *
*****************************************************************
READSTUD EQU   *
         ST    R10,SVREADS
         GET   STUDENT,SREC
         B     READSX
ATENDSTU EQU   *
         MVI   EOFSTUD,C'Y'
READSX   EQU   *
         L     R10,SVREADS
         BR    R10
*****************************************************************
*        READCRS - Read a course record.                        *
*****************************************************************
READCOUR EQU   *
         ST    R10,SVREADC
         GET   COURSE,CREC
         B     READCX
ATENDCRS EQU   *
         MVI   EOFCOURS,C'Y'
READCX   EQU   *
```

(continued)

```
         L     R10,SVREADC
         BR    R10
*****************************************************************
*        READGRAD - Read a Grade record.                        *
*****************************************************************
READGRAD EQU   *
         ST    R10,SVREADG
         GET   GRADE,GREC
         B     READGX
ATENDGRA EQU   *
         MVI   EOFGRADE,C'Y'
READGX   EQU   *
         L     R10,SVREADG
         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 STUDENT
         CLOSE GRADE
         CLOSE REPORT
         WTO   'TRANSCR2 ... Transcripts on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
STUDENT  DCB   LRECL=22,RECFM=F,MACRF=G,EODAD=ATENDSTU,
               DDNAME='STUDENT.SRT'
COURSE   DCB   LRECL=23,RECFM=F,MACRF=G,EODAD=ATENDCRS,
               DDNAME='COURSE.SRT'
GRADE    DCB   LRECL=26,RECFM=F,MACRF=G,EODAD=ATENDGRA,
               DDNAME='GRADE.SRT'
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
SVREADS  DC    F'0'               READSTUD
SVREADC  DC    F'0'               READGRAD
SVREADG  DC    F'0'               READCOUR
SVLOAD   DC    F'0'               LOADTBL
SVWRITE  DC    F'0'               WRITE
```

(continued)

```
SVWRAP   DC    F'0'               WRAPUP
SVFORMAT DC    F'0'               FORMAT
SVCRSDAT DC    F'0'               CRSDATA
SVGRDVAL DC    F'0'               GRADEVAL
SVDOGPA  DC    F'0'               DOGPA
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'            PC/370 ONLY - EBCDIC CR/LF
EOFSTUD  DC    CL1'N'             End of students file? (Y/N)
EOFCOURS DC    CL1'N'             End of course file? (Y/N)
EOFGRADE DC    CL1'N'             End of grades file? (Y/N)
POINTS   DC    PL2'0'             Points for this course
QUALITY  DC    PL2'0'             Total points
ATTEMPT  DC    PL2'0'             Hours attempted
ACCUM    DC    PL2'0'             Accumulated points
PREVCID  DC    XL5'00'            Sequence check on course ID
WK7      DC    CL7' '
PK2      DC    PL2'0'
PK3      DC    PL3'0'
PK4      DC    PL4'0'
         COPY  DIVISION
*****************************************************************
*        Table to determine value of a letter grade.            *
*****************************************************************
VALUE    DC    PL1'0'             How much this grade is worth
GRADETBL DS    0CL2
         DC    CL1'A',PL1'4'
         DC    CL1'B',PL1'3'
         DC    CL1'C',PL1'2'
         DC    CL1'D',PL1'1'
         DC    CL1'F',PL1'0'
         DC    X'FF'
*****************************************************************
*        Input record definition - Student                      *
*****************************************************************
SREC     DS    0CL22      1-22    Student record
SSID     DS    CL3        1- 3    Student ID nbr
SSNAME   DS    CL15       4-18    Student name
SSSEX    DS    CL1       19-19    Gender
SSMAR    DS    CL1       20-20    Marital status
SSCRLF   DS    CL2       21-22    PC/370 only - CR/LF
*****************************************************************
*        Input record definition - Course                       *
*****************************************************************
CREC     DS    0CL23      1-23    Course record
CCID     DS    CL5        1- 5    Course ID nbr
CCDESC   DS    CL15       5-20    Course description
CCHRS    DS    CL1       21-21    Hours
CCCRLF   DS    CL2       22-23    PC/370 only - CR/LF
*****************************************************************
*        Input record definition - Grade                        *
*****************************************************************
GREC     DS    0CL26      1-26    Grade record
GSID     DS    CL3        1- 3    Student ID nbr
GSEM     DS    CL3        4- 6    Semester
GCID     DS    CL5        7-11    Course ID nbr
GSECT    DS    CL1       12-12    Section number
GGRADE   DS    CL1       13-13    Grade earned
GKEY     DS    CL11      14-24    Sort key (see TRANS1.MLC)
GGCRLF   DS    CL2       25-26    PC/370 only - CR/LF
```

(continued)

```
*****************************************************************
*        Course table                                           *
*****************************************************************
ROWS     DC    H'0'               Entries in course table
MAXROWS  DC    H'10'              Max entries in course table
TABLE    DS    10CL21             Each row consists of:
*                          1- 5    Course ID
*                          6-20    Course description
*                         21-21    Hours (packed)
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62       1-62   Report record
         DC    CL2' '      1- 2
OSEM     DS    CL3         3- 5   Semester
         DC    CL5' '      6-10
OCID     DS    CL5        11-15   Course ID
         DC    CL3' '     16-18
OCDESC   DS    CL15       19-33   Course Description
         DC    CL4' '     34-37
OGRADE   DS    CL1        38-38   Grade
         DC    CL4' '     39-42
OCHRS    DS    CL4        43-46   Course Hours (BZZ9)
         DC    CL3' '     47-49
OPOINTS  DS    CL4        50-53   Quality Points (BZZ9)
         DC    CL7' '     54-60
OCRLF    DS    CL2        61-62   PC/370 only - CR/LF
*****************************************************************
*        Totals line definition                                 *
*****************************************************************
TREC     DS    0CL62       1-62
         DC    CL35' '     1-35
         DC    CL7'TOTAL'
TATTEMPT DS    CL4        43-46   Course Hours (BZZ9)
         DC    CL3' '     47-49
TQUALITY DS    CL4        50-53   Quality Points (BZZ9)
         DC    CL7' '     54-60
         DC    X'0D25'    61-62   PC/370 only - CR/LF
*****************************************************************
*        Totals line definition                                 *
*****************************************************************
AREC     DS    0CL62       1-62
         DC    CL35' '     1-35
         DC    CL6'GPA'
AGPA     DS    CL5        42-46   GPA (B9.99)
         DC    CL14' '    54-60
         DC    X'0D25'    61-62   PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
FORMFEED DS    0CL62              PC/370 only
*        DC    X'0C'              EBCDIC formfeed
*        DC    CL59' '
         DC    60C'_'             For testing...
         DC    X'0D25'            EBCDIC CR/LF
HD1      DS    0CL62
         DC    CL24'        TRANSCRIPT FOR ('
HDSID    DS    CL3
         DC    CL2') '
```

(continued)

```
HDSNAME  DS    CL15
         DC    CL16' '
         DC    XL2'0D25'
HD2      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL40'Semester  Course    Description    Grade'
         DC    CL20'  Hours  Points     '
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'--------  ------  ---------------  -----'
         DC    CL20'  -----  ------     '
         DC    XL2'0D25'
         END   BEGIN
```

Automating the Process with a Batch File

This is not intended to be a book on PC/MS-DOS. Nevertheless, the reader who is proficient in PCs will recognize that a batch file could (and should) be used to simplify the execution of this program. My solution follows:

```
A:\MIN>type transcr.bat
@echo off
rem TRANSCR.BAT by BQ - Produce transcripts
rem Append sort key to grade record...
transcr1
echo Sorting the reformatted grade file...
sort /+14 < grade.$$$ > grade.srt
rem Temporary file is no longer needed...
del grade.$$$
echo Sorting the student file...
sort /+1 < student.dat > student.srt
echo Sorting the course file...
sort /+1 < course.dat > course.srt
rem Produce transcripts...
transcr2
echo Cleaning up...
del grade.srt
del student.srt
del course.srt
echo Done!

A:\MIN>transcr
TRANSCR1 ... Begin execution
TRANSCR1 ...  22 grade records reformatted.
TRANSCR1 ... Normal end of program
Sorting the reformatted grade file...
Sorting the student file...
Sorting the course file...
TRANSCR2 ... Begin execution
TRANSCR2 ... Transcripts on REPORT.TXT
TRANSCR2 ... Normal end of program
Cleaning up...
Done!
A:\MIN>
```

A Summary of the Most Common Table Processing Instructions

To point a register to a table:

```
LA     R8,TABLE
L      R8,=A(TABLE)
```

To place a count in a register:

```
L      R7,=F'5'
LH     R7,=H'5'
LA     R7,5
```

To point to the next row:

```
A      R7,=F'29'
AH     R7,=H'29'
LA     R7,29(R7)
```

To "backup" a register, as in to point to a previous row:

```
S      R7,=F'29'
SH     R7,=H'29'
```

If you are "backing up" by a value of one (only) you can also use:

```
BCTR   R7,0
```

To clear (zero) a register:

```
L      R5,=F'0'
LH     R5,=H'0'
SR     R5,R5
SLR    R5,R5
```

Exercises

1. True or false. All questions refer to the following program segment:

```
REGIONS  DS    0CL10
EAST     DC    PL2'0',CL8'EAST'
MIDWEST  DC    PL2'0',CL8'MIDWEST'
WEST     DC    PL2'0',CL8'WEST'
         DC    X'FF'
TOTAL    DC    PL3'0'
```

T F a. The `X'FF'` at the end of the table is referred to as a centennial.
T F b. To add 1 to `MIDWEST`, we can use `AP MIDWEST,=P'1'`
T F c. To point register 3 to the start of the table, we can use `LA R3,REGIONS`
T F d. If register 3 is pointing to a row, we can use `LA R3,10` to point to the next row.
T F e. If register 3 is pointing to a row, we can use `A R3,=F'10'` to point to the next row.
T F f. If register 3 is pointing to a row, we can use `LA R3,L'REGIONS(R3)` to point to the next row.
T F g. If register 3 is pointing to a row, we can use `SH R3,=H'10'` to point to the previous row.
T F h. If register 3 is pointing to `WEST`, then `AP WEST,=P'1'` is the same as `AP 0(R3),=P'1'`
T F i. If register 3 is pointing to a row, to add that row's count to `TOTAL`, we can use `AP TOTAL,0(2,R3)`
T F j. To see if register 3 is pointing to the end of the table, we can use `CLI 0(1,R3),X'FF'`
T F k. If register 3 is pointing to a row, we can use `CP 0(2,R3),=P'0'` to see if that row's count is equal to zero.
T F l. If we replace all `PL2'0'` with `H'0'` then we must make sure `REGIONS` is halfword aligned.
T F m. If we replace all `PL2'0'` with `H'0'`, and register 3 is pointing to a row, then we can use `AH 0(2,R3),=H'1'` to add 1 to that row's count.

2. Given the following field definitions:

```
WSTATE   DS    CL2
FOUNDSW  DS    CL1
NO       EQU   C'N'
YES      EQU   C'Y'

#STATES  DC    H'50'
STATES   DC    CL2'AK'
         DC    CL2'AL'
         etc.
```

(continued)

Exercises

Fill in the blanks:

```
* IS WSTATE IN TABLE?
         ____  R8,________  Point to start of table
         ____  R7,________  Load nbr of table entries
LOOP     EQU   *
         ____  ___________  Match?
         ____  ___________  Yes - Go to found
         ____  ___________  No - point to next entry
         ____  ___________  Repeat if not at end
         ____  ___________  Indicate not found
         ____  ___________  Exit
FOUND    EQU   *            Match was found
         ____  ___________  So indicate
EXIT     EQU   *
```

3. Given the following field definitions:

```
STREC    DS    0CL7
STSTATE  DS    CL2
STRATE   DS    CL3          (V999)
STCRLF   DS    CL2          PC/370 only

EOFSW    DC    CL1'N'
YES      EQU   C'Y'

#ENTRIES DS    H
TAXTABLE DS    50CL4
*              1-2  STATE (Character)
*              3-4  SALES TAX RATE V999 (Packed)
```

Fill in the blanks:

```
* LOAD TABLE W/ STATE AND SALES TAX RATE
         ____  R6,________  Point to start of table
         ____  R5,________  Set count to zero
         BAL   R10,READST   Get State/Rate record
LOOP     EQU   *
         ____  ___________  End of file?
         ____  ___________  Yes - Done.
         ____  ___________  Move state to table
         ____  ___________  Move rate to table
         ____  ___________  Point to next entry
         ____  ___________  Increment
         ____  ___________  Get another record
         ____  ___________  Repeat
DONE     EQU   *
         ____  ___________  Save counter
```

Exercises

4. Given the following field definitions:

```
         DS    0F               Force fullword alignment
IREC     DS    0CL40    1 - 40  Input Record, EBCDIC, no CR/LF
INAME    DS    CL24     1 - 24  First and last name
ITLY     DS    F       25 - 28  Total sales last year Jan-June
IMTY     DS    6H      29 - 40  Monthly sales this year, Jan-June
*
OLINE    DS    0CL100   1 -100  Output Line
ONAME    DS    CL24     1 - 24  First and last name
         DS    CL4     25 - 28
OMTY     DS    6CL8    29 - 76  Monthly sales this year, BZZ,ZZ9-
OTTY     DS    CL11    77 - 87  Total this year, BZ,ZZZ,ZZ9-
OTLY     DS    CL11    88 - 98  Total last year, BZ,ZZZ,ZZ9-
OCRLF    DS    CL2     99 -100  PC/370 only - CR/LF
*
DBL      DS    D
FULL     DS    F
WTTY     DS    F                Work - Total this year
PK3      DS    PL3
```

Fill in the blanks:

```
* Determine total sales this year for this employee
         _____ R6,_____          Point to first month
         _____ R5,_____          How many months?
         ____  R4,_____          Use R4 to hold total
LOOP1    EQU   *
         ____  ________________  Add monthly sales to total
         ____  ________________  Point to next month
         ____  ________________  Repeat till all months done
         ____  _____,WTTY        Save Result


* Move Monthly Sales (only) to Output
         ____  R7,_____          Point to first month-Output
         ____  R6,_____          Point to first month-Input
         ____  R5,_____          How many months?
LOOP2    EQU   *
         ____  R3,_____          Place months sales in register
         ____  ________________  Convert to decimal
         ____  PK3,____________  Move to smaller field
         ____  ________________                        Move mask
         ____  ________________  Edit amount
         ____  ________________  Point to next month-Output
         ____  ________________  Point to next month-Input
         ____  ________________  Repeat till all months done
```

5. (Refer to the Small Town Blood Bank database in More Datasets.) Note the donor name in the DONOR table is stored with last name and first name is separate fields. Write a short program (similar to TEACH15A.MLC in this chapter) which will display each donor name as a single field; that is, with a single space separating the first name and last name.

Exercises

6. (Refer to the Small Town Self-Storage database in More Datasets.) Note the customer name in the CUST table is stored as last name, followed by a comma, followed by first name; for example: QUILTY, CECELIA. Write a short program (similar to TEACH15A.MLC in this chapter) which will display each customer name with first name first and no comma; for example: CECELIA QUILTY.

7. (Refer to the Small Town Blood Bank database in More Datasets.) Create a detailed history of donations. For each donor, list the date of each donation and the name of group credited with the donation. Your report should appear as follows:

```
         1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345
                     SMALL TOWN BLOOD BANK              Page BZZ9
                     Detailed Donor History

ID#      Donor Name       Type        Group             Date
--- ---------------------- ---- --------------------  --------
XXX XXXXXXXXXX XXXXXXXXXX XXX  XXXXXXXXXXXXXXXXXXXX  mm/dd/yy
                               XXXXXXXXXXXXXXXXXXXX  mm/dd/yy

XXX XXXXXXXXXX XXXXXXXXXX XXX  XXXXXXXXXXXXXXXXXXXX  mm/dd/yy
                               XXXXXXXXXXXXXXXXXXXX  mm/dd/yy
```

8. (Refer to the Small Town Hardware Store database in More Datasets.) Produce a list of kits and their tools as follows:

```
         1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345
                     SMALL TOWN HARDWARE STORE          Page BZZ9
                            Kits List

           KIT                          TOOL
--------------------------  --------------------------
ID#      Description         ID#      Description          Cost
--- ----------------------  --- ----------------------  --------
XXX XXXXXXXXXXXXXXXXXXXXXX  XXX XXXXXXXXXXXXXXXXXXXXX  BZZZ.99
                            XXX XXXXXXXXXXXXXXXXXXXXX  BZZ9.99
                                TOTAL COST              BZZ9.99

XXX XXXXXXXXXXXXXXXXXXXXXX  XXX XXXXXXXXXXXXXXXXXXXXX  BZZZ.99
                            XXX XXXXXXXXXXXXXXXXXXXXX  BZZ9.99
                                TOTAL COST              BZZ9.99
```

You will need to use the KIT, MAKEUP, and TOOL files. Load the TOOL file into a table. Match KIT and MAKEUP on kit ID. Use the tool ID in the MAKEUP file to search the table to find the matching description and cost.