Chapter 3
IFs in BAL: Comparing Character Fields

Objectives

Upon completion of this chapter you will be able to:
- Determine the condition code resulting from alphanumeric comparisons using the `CLC` and `CLI` instructions,
- Explain how the branch on condition instruction is used to alter the sequence of processing based on the condition code,
- List the extended mnemonics for the branch on condition instruction,
- Use the `CLC` and `CLI` instructions to implement the logic in a given flowchart or pseudocode, and
- Write an extract program.

Introduction

In the previous chapters we have seen several ways to produce a list of the records in the `TEACHER` file. In each case we have listed the entire file. We now look at how to list selected records only. Such programs are commonly referred to as "extract" programs.

Our next program will produce a list of tenured instructors. The new report will appear as follows:

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890
      LIST OF TENURED INSTRUCTORS

ID#        Name         PhD?    Phone
---   ---------------   ----   --------
XXX   XXXXXXXXXXXXXXX    X     517-XXXX
XXX   XXXXXXXXXXXXXXX    X     517-XXXX
XXX   XXXXXXXXXXXXXXX    X     517-XXXX
                         :      :
                         :      :.....  All campus phones
                         :              begin with '517-'
                         :
                         :...........  'Y' if highest degree
                                        is PhD, 'N' otherwise
```

In order to extract records, we need to learn the `BAL` equivalent to the `IF` verb as found in most other languages. In this program, we will need the `IF` for two purposes:
- To determine if an instructor is tenured and should, therefore, be included in the report, and
- To determine if the instructor has a PhD, and to print a `Y` or an `N` accordingly.

Compares in BAL: The Condition Code

When two fields are compared in `BAL`, a condition code is set. *That's all that happens: a condition code is set*. The condition code is a special area of four bits set aside in the CPU for just this purpose. (Another instruction, Branch on Condition, is used to indicate what action should be taken based on the resultant condition code. This instruction is discussed later in this chapter.)

There are three possible conditions as a result of a compare operation:
- The first operand is equal to the second operand,
- The first operand is less than the second operand, or
- The first operand is greater than the second operand.

The condition code for each of the above conditions is:

Compare A : B	**First bit**	**Second bit**	**Third bit**	**Fourth bit**	**Decimal Value**	**Result**
A = B	1	0	0	0	8	equal
A < B	0	1	0	0	4	low
A > B	0	0	1	0	2	high

Note that in `BAL`, there is no equivalent to a test for equal, or a test for greater than, etc. There is only a compare, which sets a condition code. It is then up to you to test the condition code and have the program branch, or "goto", the desired location in your program accordingly.

Compares in BAL: The CLC Instruction

The general format for a character compare is `CLC OP1,OP2`

`CLC` stands for Compare Logical Character. It works much the same way as the `MVC` instruction in that the length of the compare (the number of bytes compared) is determined by the length of the first operand (in this case, `OP1`) *regardless of the length of the second operand*. A maximum of 256 bytes can be compared with a single `CLC`.

Let's try some compares, and determine the resulting condition code. We will refer to the following data throughout our discussion of `IFs` in `BAL`:

Field Definitions

```
AREC      DS    0CL28                    1-28
ANAME     DS    0CL16                    1-16
AFIRST    DS    CL8                      1- 8
ALAST     DS    CL8                      9-16
AZIP      DS    CL5                     17-21
AAGE      DS    CL2                     22-23
          DS    CL2                     24-25
ASEX      DS    CL1                     26-26
ACRLF     DS    XL2     PC/370 Only     27-28
```

```
BREC     DS    0CL28                     1-28
BNAME    DS    0CL18                     1-18
BFIRST   DS    CL9                       1- 9
BLAST    DS    CL9                      10-18
BZIP     DS    CL5                      19-23
BAGE     DS    CL2                      24-25
BSEX     DS    CL1                      26-26
BCRLF    DS    XL2     PC/370 Only      27-28

CREC     DS    0CL28                     1-28
CNAME    DS    0CL14                     1-14
CFIRST   DS    CL7                       1- 7
CLAST    DS    CL7                       8-14
CZIP     DS    CL9                      15-23
CAGE     DS    CL2                      24-25
CSEX     DS    CL1                      26-26
CCRLF    DS    XL2     PC/370 Only      27-28
```

We will also refer to the following work areas:

```
BLANKS   DC    CL20' '
WNAME    DS    0CL20
WFIRST   DS    CL10
WLAST    DS    CL10
WZIP     DS    CL5
WAGE     DS    CL2
WSWITCH  DS    CL1
```

The Data

```
                  1         2
        12345678901234567890123456
File A  CHERYL  HAVLIK  6018333  F
File B  APRIL    HAVLIK   6055430F
File C  KEVEN  FOOTE  60183025828M
```

* * * * * * * * * * * * * * * * * * * *

1. `CLC   AFIRST,BFIRST`

Since `AFIRST` is defined as `CL8`, a total of eight bytes will be compared, even though `BFIRST` is defined as `CL9`: the last byte of `BFIRST` is not included in the compare.

```
                     CLC   AFIRST,BFIRST
                                :      :
 |C|H|E|R|Y|L|ƀ|ƀ|.........:      :
                                       :
 |A|P|R|I|L|ƀ|ƀ|ƀ|................:
```

The comparison takes place from left to right, comparing the first byte of each field, then the second, then the third, etc., until all bytes have been compared or until a difference is found. In this case, the first byte is different (`C` vs. `A`) so the compare ends after one byte only. And since `C` is greater than (comes after) `A`, the resulting condition code is `0010`, or 2, or high.

2. `CLC   AZIP,BZIP`

   Since `AZIP` is defined as `CL5`, a total of five bytes will be compared. The fact that `BZIP` is also defined as `CL5` is irrelevant.

```
                        CLC   AZIP,BZIP
                                 :    :
 |6|0|1|8|3|...............:    :
                                      :
 |6|0|5|5|4|....................:
```

   The first difference occurs at the third byte. Since `1` is less than `5`, `AZIP` is less than `BZIP` and the resulting condition code is `0100`, or 4, or <u>low</u>.

3. `CLC   CSEX,BSEX`

   Since `CSEX` is defined as `CL1`, the first (and only) byte of each field will be compared. The fact that `BSEX` is also defined as `CL1` is irrelevant.

```
                        CLC   CSEX,BSEX
                                 :    :
 |M|.......................:    :
                                      :
 |F|...........................:
```

   Since `M` is greater than (comes after) `F`, the resulting condition code is `0010`, or 2 or <u>high</u>.

4. `CLC   AZIP,CZIP`

   Since `AZIP` is defined as `CL5`, a total of five bytes will be compared, even though `CZIP` is defined as `CL9`: the four rightmost bytes of `CZIP` are not included in the compare.

```
                        CLC   AZIP,CZIP
                                 :    :
 |6|0|1|8|3|...............:    :
                                      :
 |6|0|1|8|3|....................:
```

   All five bytes are equal. The resulting condition code is `1000`, or 8, or <u>equal</u>.

5. `CLC   CZIP,AZIP`

   At first glance, it may appear that the answer should be the same as for number 4, but such is <u>not</u> the case. Since `CZIP` is defined as `CL9`, a total of nine bytes will be compared, even though `AZIP` is defined as `CL5`. The nine bytes compared will be the contents of `AZIP` plus the next four bytes in the record; that is, the age (two bytes) and a filler (two bytes).

```
                          CLC    CZIP,AZIP
                                  :    :
     |6|0|1|8|3|0|2|5|8|.......:    :
                                       :
     |6|0|1|8|3|3|3|b̶|b̶|...........:
```

The first difference occurs with the sixth byte. Since `0` is less than `3`, the resulting condition code is `0100`, or 4, or <u>low</u>.

* * * * * * * * * * * * * * * * * * * *

<u>You Try It...</u>

Determine the resulting condition code for the following compares:

1. `CLC    ALAST,BLAST`
2. `CLC    BLAST,ALAST`

* * * * * * * * * * * * * * * * * * * *

With `CLC`, as with `MVC`, explicit displacement and/or length can be used to override the default displacement (zero) and length (field length) respectively. Explicit displacement may be used on either operand, while explicit length may be used on the first operand only. For example, to compare bytes 3-5 of `FLDA` with bytes 2-4 of `FLDB`, code:

```
CLC    FLDA+2(3),FLDB+1
```

Literals can be used with `CLC`. For example, the following are valid and functionally equivalent:

```
CLC    OPTION,=CL3'YES'
CLC    OPTION,=C'YES'
CLC    OPTION,YES         where  YES  DC  CL3'YES'
```

<u>Compares in BAL: The CLI Instruction</u>

The `CLI`, or Compare Logical Immediate, instruction is to `CLC` as `MVI` is to `MVC`: use it to compare a single byte to a literal. For example, to check to see if the first byte of a field is an asterisk, code:

```
CLI    FLDA,C'*'
```

As with `MVI`:
- Explicit displacement can be used,
- Explicit length cannot be used,
- There is no equal sign on the literal, and
- Equated values can help to improve readability and maintainability.

You Try It...

Determine the resulting condition code for the following compares:

```
3.    CLI   ASEX,C'M'
4.    CLI   CZIP,ZERO      where  ZERO EQU C'0'
```

* * * * * * * * * * * * * * * * * * * *

Compares in BAL: The BC Instruction

Once the condition code has been set, you use the Branch on Condition instruction to indicate where to go based upon the value of the condition code. The Branch on Condition, or BC, instruction has two operands: the test value, or **mask**, for the condition code, and the **label** to which the program should branch. For example:

```
                CLC    A,B
                BC     8,EQUAL
                BC     4,ALOW
                BC     2,AHIGH
                BC     7,NOTEQUAL         Note:  7 = 15 - 8
                BC     11,ANOTLOW               11 = 15 - 4
                BC     13,ANOTHIGH              13 = 15 - 2
                 :
       EQUAL    EQU    *
                 :
       AHIGH    EQU    *
                 :
                etc.
```

The BC instruction works as follows: after a compare, the condition code is set. The mask in the instruction is then compared to the condition code. If any "on" bit in the mask has a corresponding "on" bit in the condition code, then the branch is taken. For example:

```
       Condition code     1000       8: A = B
       Instruction mask   0010       2: Branch not taken

       Condition code     0100       4: A < B
       Instruction mask   0110       6: Branch taken

       Condition code     0010       2: A > B
       Instruction mask   1111      15: Branch taken
```

Note the new use of EQU in the above example! An asterisk, when used as an operand in an instruction, refers to the address of that instruction. Here, EQU is used to equate a label with an address in the program. That's a fancy way of saying `label EQU *` is a convenient way of defining a paragraph in BAL. It will be used in all examples and programs from now on.

Clearly these condition codes are not easily committed to memory, so BAL provides the following **extended mnemonic instructions**:

BE	Branch on A Equal B
BH	Branch on A High
BL	Branch on A Low
BNE	Branch on A Not Equal B
BNH	Branch on A Not High
BNL	Branch on A Not Low

So, for example, the previous BC instructions could be rewritten as:

```
         CLC   A,B
         BE    EQUAL
         BH    AHIGH
         BL    ALOW
         BNE   NOTEQUAL
         BNH   ANOTHIGH
         BNL   ANOTLOW
```

One important BC remains: BC 15 is an unconditional branch. Its mnemonic is simply B.

All of these branches are the same as GOTOs in other languages. If you have been taught to use structured programming or, more specifically, "go-to-less" programming, this may not sit well with you. But get used to it! You cannot do go-to-less programming in BAL. (There are some macros commonly used which allow an IF-ENDIF construct, but they are an extension of BAL and will not be discussed here.)

* * * * * * * * * * * * * * * * * * * *

<u>You Try It...</u>

5. Will BC 8,SKIP branch to SKIP if the condition code is 8?
6. Will BC 12,SKIP branch to SKIP if the condition code is 2?
7. Will BC 15,SKIP branch to SKIP if the condition code is 8?
8. Will BE SKIP branch to SKIP if the condition code is 8?
9. Will B SKIP branch to SKIP if the condition code is 4?

* * * * * * * * * * * * * * * * * * * *

Compares in BAL: Sample IFs

BAL is usually not the first language someone learns. What follows is an easy way to learn how to do IFs in BAL which takes advantage of existing programming knowledge.

To learn to do IFs in BAL, you must think <u>very low level</u>. BAL is, after all, a low level (second generation) language. So let's step back in time. Let's pretend we are coding in a very primitive form of BASIC. This version of BASIC is entirely unstructured and has the following restrictions:

1. There is only one line per statement. (In more recent versions of BASIC a colon can be used to put multiple BASIC statements on the same line. This will not be the case in our "old BASIC".)
2. There is an IF verb, but this IF does not allow compound conditions (AND or OR).
3. The only thing you can do as a result of an IF is a GOTO. (This was *really* the case with some early versions of BASIC!)
4. Futhermore, in our "old BASIC", the only thing you can GOTO is a REM (remark) statement; specifically, you can GOTO a REM THEN, REM ELSE, REM ENDIF, REM AND, or REM OR only.

That's what it's like coding IFs in BAL! Consider the following examples. (If you don't know BASIC but do know some other language, you should be able to understand the examples. Don't get hung up on the BASIC syntax, such as dollar signs to indicate string variables, etc. It's the concept of programming IFs in a low-level language that we are most concerned with here.)

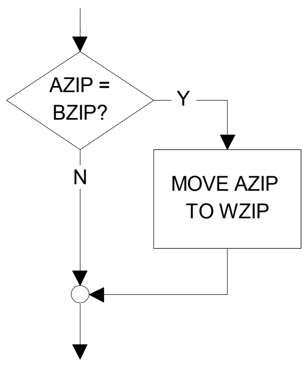
A note concerning the flowcharts contained herein: If you are familiar with structured programming, then you know that all programming problems can be solved using three simple constructs: sequence, selection (IF-THEN-ELSE) and iteration (DOWHILE). All flowcharts should be constructed so that they can be broken down into combinations of these three constructs only. The flowcharts you will see here do not fit these constructs. The reason is that these constructs (particularly selection and iteration) generally allow compound conditions (AND and OR). Since BAL does not allow for these compound conditions, multiple selections must be used; that is, a simple DOWHILE in some languages will require multiple CLCs in BAL. These flowcharts could have been written so that they were fully structured, but to do so would require repeating the code for the THEN and/or ELSE actions. The flowcharts here have been drawn to reflect the code as it would generally be written in the real world.

Example #1 - IF..THEN

Pseudocode

```
IF AZIP is equal to BZIP
   move AZIP to WZIP
ENDIF
```

Flowchart



"Old BASIC" Solution

```
10 IF AZIP$ <> BZIP$ THEN GOTO 30
20 LET WZIP$ = AZIP$
30 REM ENDIF
```

BAL Solution

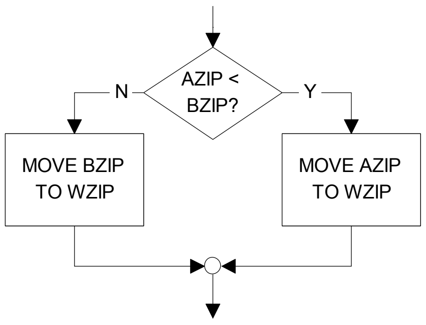
```
         CLC   AZIP,BZIP
         BNE   SKIP
         MVC   WZIP,AZIP
SKIP     EQU   *
```

Example #2 - `IF..THEN..ELSE`

Pseudocode

```
IF AZIP is less than BZIP
   move AZIP to WZIP
ELSE
   move BZIP to WZIP
ENDIF
```

Flowchart



"Old BASIC" Solution

```
10 IF AZIP$ >= BZIP$ THEN GOTO 40
20 LET WZIP$ = AZIP$
30 GOTO 60
40 REM ELSE
50 LET WZIP$ = BZIP$
60 REM ENDIF
```

BAL Solution

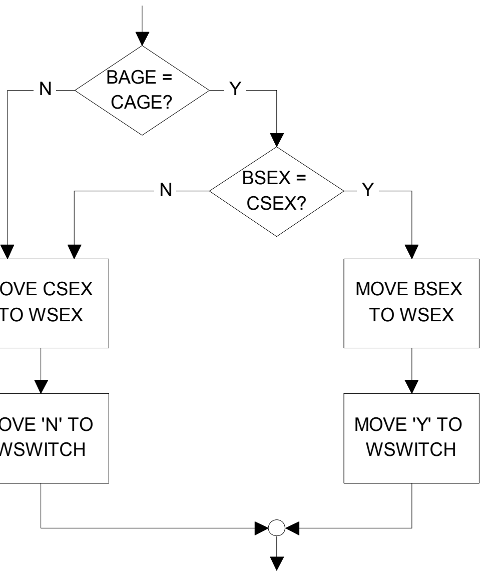
```
         CLC   AZIP,BZIP
         BNL   USEB
         MVC   WZIP,AZIP
         B     SKIP
USEB     EQU   *
         MVC   WZIP,BZIP
SKIP     EQU   *
```

Example #3 - **`IF..AND..THEN..ELSE`**

Pseudocode

```
IF (BAGE is equal to CAGE) AND (BSEX = CSEX)
   move BSEX to WSEX
   move 'Y' to WSWITCH
ELSE
   move CSEX to WSEX
   move 'N' to WSWITCH
ENDIF
```

Flowchart



"Old BASIC" Solution

```
10 IF BAGE$ <> CAGE$ THEN GOTO 60
20 IF BSEX$ <> CSEX$ THEN GOTO 60
30 LET WSEX$ = BSEX$
40 LET WSWITCH$ = "Y"
50 GOTO 90
60 REM ELSE
70 LET WSEX$ = CSEX$
80 LET WSWITCH$ = "N"
90 REM ENDIF
```

BAL Solution

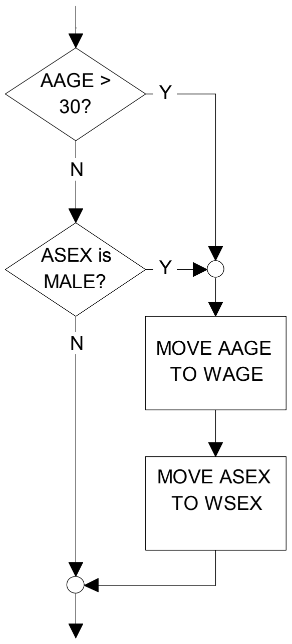
```
         CLC   BAGE,CAGE
         BNE   NOPE
         CLC   BSEX,CSEX
         BNE   NOPE
         MVC   WSEX,BSEX
         MVI   WSWITCH,C'Y'
         B     DONE
NOPE     EQU   *
         MVC   WSEX,CSEX
         MVI   WSWITCH,C'N'
DONE     EQU   *
```

Example #4 - `IF..OR..THEN`

Pseudocode

```
IF (AAGE greater than 30) OR (ASEX is male)
   move AAGE to WAGE
   move ASEX to WSEX
ENDIF
```

Flowchart



"Old BASIC" Solution

```
10 IF AAGE$ > "30" THEN GOTO 40
20 IF ASEX$ = "M" THEN GOTO 40
30 GOTO 70
40 REM THEN
50 LET WAGE$ = AAGE$
60 LET WSEX$ = ASEX$
70 REM ENDIF
```

BAL Solution

```
         CLC   AAGE,=CL2'30'
         BH    YES
         CLI   ASEX,C'M'
         BE    YES
         B     MORE
YES      EQU   *
         MVC   WAGE,AAGE
         MVC   WSEX,ASEX
MORE     EQU   *
```

* * * * * * * * * * * * * * * * * * * *

Comprehensive Example

Assume we would like to determine if a middle name field contains an initial only and, if so, to move a period after the initial. Assume the following field definition and value:

```
         MNAME     DS     CL12      |P|ƀ|ƀ|ƀ|ƀ|ƀ|ƀ|ƀ|ƀ|ƀ|ƀ|ƀ|
```

We will assume that there is an initial (only) if
- the first byte is not blank, and
- the remaining bytes are blank.

One solution is as follows:

```
                   CLI    MNAME,C' '
                   BE     NOINIT
                   CLC    MNAME+1(11),=CL11' '
                   BNE    NOINIT
                   MVI    MNAME+1,C'.'
         NOINIT    EQU    *
```

Note the use of `EQU *` to define a label to which the program can branch, or "go to", based on the condition code resulting from `CLI` or `CLC`.

Sample Program

Recall that the programming problem presented at the beginning of this chapter was:
- To determine if an instructor is tenured and should, therefore, be included in the report, and
- To determine whether or not the instructor has a PhD, and to print a `Y` or an `N` accordingly.

The new program is `TEACH3A.MLC`: the program and its output follow:

```
         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
         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
```

(continued)

```
         OPEN  TEACHERS
         OPEN  REPORT
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
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
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH3A ... Teacher list on REPORT.TXT'
         WTO   'TEACH3A ... Normal end of program'
         RETURN
*
*        Literals, if any, will go here
*
         LTORG
*
*        File definitions
*
TEACHERS DCB   LRECL=29,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='TEACHER.DAT'
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*
*        Miscellaneous field definitions
*
WCRLF    DC    X'0D25'            PC/370 ONLY - EBCDIC CR/LF
*
*        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
*
```

(continued)

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

```
A:\MIN>teach3a
TEACH3A ... Begin execution
TEACH3A ... Teacher list on REPORT.TXT
TEACH3A ... Normal end of program

A:\MIN>type report.txt
      LIST OF TENURED INSTRUCTORS

ID#        Name         PhD?    Phone
---   ---------------   ----   --------
854   KIMBALL, S.W.      Y     517-5594
626   YOUNG, B.          N     517-5664
574   SMITH, J.          N     517-5320
```

Exercises

1. True or false.

   T F a. The condition code is a special area of the CPU consisting of eight bits.
   T F b. The BAL equivalent of `IF gender IS male THEN GO TO domale` requires two instructions.
   T F c. `BC 15,SKIP` is an unconditional branch.
   T F d. The `8` in `BC 8,SKIP` is also referred to as the mask.
   T F e. Up to 256 bytes can be compared with a single `CLI` instruction.
   T F f. Equated values are commonly used with `CLI`.
   T F g. `CLI INITIAL,=C' '` and `CLC INITIAL(1),C' '` will each compare the first byte of `INITIAL` to a blank.
   T F h. `ELSE EQU *` may be used repeatedly in a program.
   T F i. `BE` is an example of an extended mnemonic.
   T F j. `BC 2,XXX` and `BL XXX` are equivalent.
   T F k. In order to implement `AND` and `OR` conditions, the `CLC` and `CLI` instructions allow multiple comparisons in a single instruction.
   T F l. Explicit displacement may be used with `CLI`.
   T F m. Explicit length may be used with `CLI`.

2. Given the following input fields:

```
NAZIP9   DS    0CL9     ZIP CODE 9 DIGITS
NAZIP5   DS    CL5      ..ZIP CODE FIRST 5
NAZIP4   DS    CL4      ..ZIP CODE "PLUS 4"
```

   Write the BAL code necessary to format the zip code for printing as per the following field definitions:

```
PRZIP9   DS    0CL10  FORMATTED 9-DIGIT ZIP CODE
PRZIP5   DS    CL5    .. ZIP CODE FIRST 5
PRDASH   DS    CL1    .. HYPHEN IF "PLUS 4" EXISTS
PRZIP4   DS    CL4    .. "PLUS 4" IF IT EXISTS
```

   (A "plus 4" exists if it is not blanks and not all zeroes.)

3. Given:

```
FLDA     DS    CL3
FLDB     DS    CL3
FLDC     DS    CL3
MAX      DS    CL3
```

   Write the BAL code to move the maximum of `FLDA`, `FLDB`, and `FLDC` to `MAX`.

Exercises

4. Given the following field definitions...

```
NACITY   DS    CL12    56-67  CITY
NASTATE  DS    CL2     68-69  STATE
NAPHONE  DS    0CL10   70-79  PHONE...
NAAREA   DS    CL3     70-72  ...AREA CODE
NAEXCH   DS    CL3     73-75  ...EXCHANGE
NALINE   DS    CL4     76-79  ...LINE
NACODE   DS    CL1     80-80  TRANS CODE (A/C/D)

WSWITCH  DS    CL1     (Y)ES OR (N)O
```

Flowchart the following IFs, and write the corresponding "old BASIC" and BAL code.

```
a.     IF state is Illinois
          move yes to switch
       ENDIF

b.     IF state is Utah or California
          move yes to switch
       ELSE
          move no to switch
       ENDIF

c.     IF trans code is other than 'A', 'C', 'D'
          move yes to switch
       ELSE
          move no to switch
       ENDIF

d.     IF area code = "312"             Careful! Watch
          AND city is not "CHICAGO"     the length on the
              move "708" to area code   CLC for the city!
       ENDIF
```

5. Using the field definitions found on pages 2-3 of this chapter, flowchart the following IFs and write the corresponding "old BASIC" and BAL code:

```
a.     IF (ANAME is blank) AND (CNAME is blank)
          move all File B fields to corresponding work area fields
       ELSE
          IF (AAGE is greater than or equal to CAGE)
             move all File A fields to corresponding work area fields
          ELSE
             move all File C fields to corresponding work area fields
          ENDIF
       ENDIF

b.     IF ((ASEX is male) AND (AAGE is 18 or over))
       OR ((ASEX is female) AND (AAGE is 21 or over))
          move 'Y' to WSWITCH
       ELSE
          move 'N' to WSWITCH
       ENDIF
```

Exercises

6. The following is an excerpt from a school catalog:

> Individuals in one or more of the following categories are placed on academic advisory:
> - students who have a cumulative grade point average below 2.40 in any term following completion of the third course.
> - students with more than three withdrawls.
> - students with more than three repeated courses

Given the following field definitions, determine if a student should be placed on academic advisory:

```
GPA      DS    CL3   Example: 2.97 = '297'
#COMP    DS    CL2   Number of courses completed
#WD      DS    CL1   Number of withdrawls
#REP     DS    CL1   Number of repeated courses
ADVISE   DS    CL1   Academic advisory (Y/N)
```

7. The following is an excerpt from a school catalog:

> Federal financial aid regulations require financial aid recipients to make incremental progress towards their degrees. In order to retain eligibility for federal financial aid, students must make incremental progress according to the chart below:

Years elapsed since initial enrollment	Minimum number of courses completed with passing grade		
	MBA	MHRM	MPM
1	2	2	2
2	5	5	4
3	8	8	7
4	12	12	10
5	16	15	13

Given the following field definitions, determine if a student is eligible for federal financial aid (`FFA`):

```
YEARS    DS    CL1   Years elapsed
#COMP    DS    CL2   Number of courses completed
DEGREE   DS    CL4   Degree sought
FFA      DS    CL1   Fed Fin Aid eligibility (Y/N)
```

Exercises

8. Given the following adjacent field definitions:

```
BOY     DC   CL7'WILLIAM'
GIRL1   DC   CL4'CORA'
GIRL2   DC   CL6'HANNAH'
GIRL3   DC   CL4'Emma'
MID     DC   CL3'KAY'
BLANKS  DC   CL5' '
MISC    DS   CL12
```

For each of the following `CLCs` and `CLIs`, determine which characters are actually being compared, and then determine the resulting condition code (8, 4, or 2). If any of the compares would produce a compiler error, then so indicate.

a. `CLC   GIRL1,GIRL2`
b. `CLC   BOY+2(1),BOY+3`
c. `CLC   BOY+2(2),BOY+3`
d. `CLC   GIRL2+2(1),GIRL2+3(1)`
e. `CLC   BOY+4(4),=C'LIAM'`
f. `CLC   GIRL1+3(2),C'AH'`
g. `CLC   BOY+6(1),GIRL3+2`
h. `CLC   GIRL3+3(1),GIRL2`
i. `CLI   GIRL1,=C'A'`
j. `CLI   GIRL3+3,C'A'`

Hannah's middle name is Kay. Write the code necessary to move "`HANNAH KAY   `" to `MISC` using the above fields only.

9. Draw the flowchart for the following `BAL` code. Assume all fields are `CL1`.

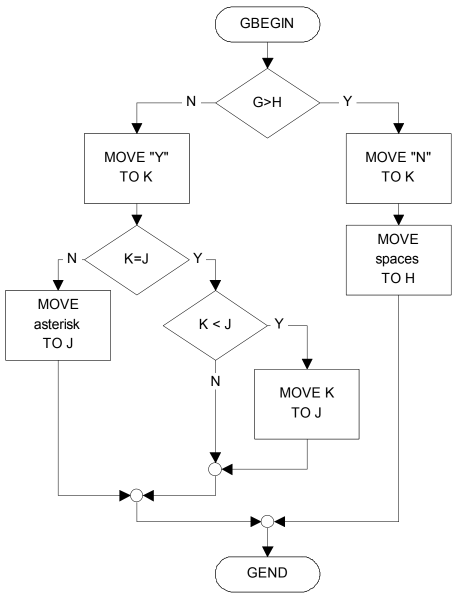
```
SBEGIN   EQU   *
         MVC   T,S
         CLC   T,U
         BNL   SEND
         CLC   T,V
         BNH   SEND
         CLC   T,W
         BNE   SKIP
         MVC   V,W
         B     SEND
SKIP     EQU   *
         MVC   V,T
SEND     EQU   *
```

Exercises

10. Draw the flowchart for the following BAL code. Assume all fields are CL1.

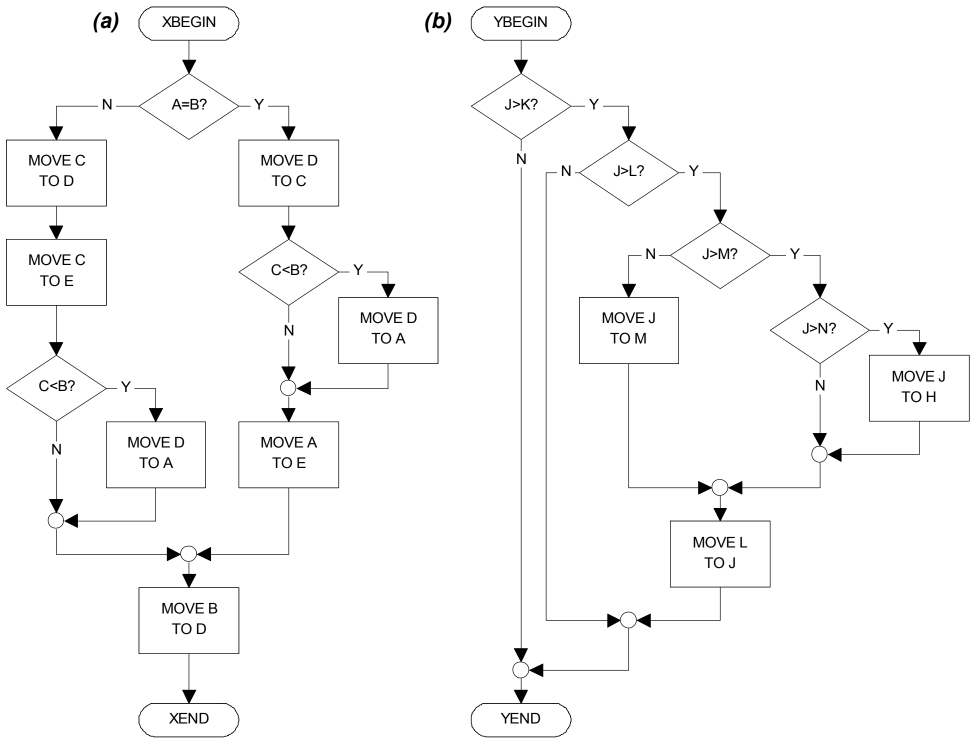
```
ZBEGIN   EQU   *
         MVC   S,R
         CLC   S,T
         BH    ZSKIP2
         CLC   S,U
         BNL   ZSKIP1
         MVC   T,R
         B     ZEND
ZSKIP1   EQU   *
         MVC   R,T
         B     ZEND
ZSKIP2   EQU   *
         MVC   S,U
         CLC   S,T
         BE    ZSKIP3
         MVC   S,Q
         CLC   Q,U
         BNL   ZSKIP4
         MVC   T,S
         B     ZSKIP4
ZSKIP3   EQU   *
         MVC   T,Q
ZSKIP4   EQU   *
         MVC   R,T
ZEND     EQU   *
```

11. Write the "old BASIC" and BAL for the following flowchart. Note: G and H are defined as CL4, while J and K are defined as CL1. Identify those portions of the flowchart and code which will never be executed.



Exercises

12. Write the "old `BASIC`" and `BAL` for the following flowcharts. Assume all fields are defined as `CL1`.



13. Write the `BAL` code for the following pseudocode.

(a)
```
IF (X > Y)
   IF (X > Z)
      MOVE Z TO W
   ELSE
      IF (X = Z)
         MOVE W TO Z
      ENDIF
      MOVE Y TO W
   ENDIF
ENDIF
```

(b)
```
IF (A = B) AND (C ≠ D)
   MOVE X TO Y
ELSE
   IF (A > B) OR (C ≤ D)
      MOVE W TO Y
   ENDIF
   MOVE X TO Z
ENDIF
```

Exercises

14. The Psi Chi guys have asked for a list of single female students. Write the program to produce a such a list from the `STUDENT` file. The report should appear as follows:

```
         1         2         3
123456789012345678901234567890123
         ELIGIBLE PARTIES
        FOR OUR NEXT PARTY

ID#    Student Name     Sex   Mar
---   ---------------   ---   ---
XXX   XXXXXXXXXXXXXXX    X     X
XXX   XXXXXXXXXXXXXXX    X     X
XXX   XXXXXXXXXXXXXXX    X     X
```

15. Write a program which will produce a formatted list of those courses which are less than 3 semester hours credit. The report should appear as follows:

```
         1         2         3         4
1234567890123456789012345678901234567890
COURSES WITH LESS THAN 3 HOURS CREDIT

Course      Description      Hours
------    ---------------    -----
XXXXX     XXXXXXXXXXXXXXX      X
XXXXX     XXXXXXXXXXXXXXX      X
XXXXX     XXXXXXXXXXXXXXX      X
```

16. The English department has requested a list of all grades for English classes for the 1992-93 school year (semesters `F92` and `W93` only.) Write a program to produce such a list in the following format:

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890
     ENGLISH GRADES FOR 1992-93 SCHOOL YEAR
                 (Confidential)

               Course   Student
         Sem   Number     ID#     Grade
         ---   ------   -------   -----
         XXX   XXXXX      XXX       X
         XXX   XXXXX      XXX       X
         XXX   XXXXX      XXX       X
```