STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING

Dr. Robert N. Cook
Univ. of Southern Colorado
Pueblo, CO 81001

Introduction

For those of us who are essentially high level programmers, the intricacies and lack of structure in assembly language programs are often an insurmountable barrier to effective assembly language programming. This paper attempts to show a way to overcome this barrier. Structured pseudocode is used to solve the problem just as if the solution were to be coded in PL/I, PASCAL, ADA, or some other structured high level language. Then the structured pseudocode is "compiled" into assembly language using appropriate labels to show the structure of the assembly language program. Formal roles for the "compilation" are not given in the paper. Instead, the various control structures are demonstrated in BAL along with the corresponding pseudocode. Branch instructions abound, of course, as they must in any assembly language program. The readability of the program, though, is immensely increased by good statement labels. By using a header, and perhaps even including the pseudocode in the program through comments, the program can be made readable even to non-assembly language programmers. Assignment statements in the assembly language are commented to show the equivalent high level statement of the pseudocode. Then most, if not all, statements in the assembly language are made readable.

IBM BAL is used to demonstrate the concepts in the paper; however, the techniques are adaptable to any assembly language. The remainder of the paper details the technique for the fundamental control structures, other control structures, and assignment statements. Other documentation techniques to increase readability are discussed, and a complete program is given to document the techniques. All techniques discussed are those taught to students in CST 210--Assembly Language Programming, at the University of Southern Colorado.

Fundamental Control Structures

The three fundamental control structures of sequential execution, alternation, and looping are described here using pseudocode and BAL. One of the easier looping structures to implement in BAL is the DO WHILE DATA AVAILABLE. Pseudocode for this structure is

```
DO WHILE DATA AVAILABLE
    GET A,B,C,D
     .
     .
     .   PROCESS DATA
     .
     .
    PUT A,B,C,D,X...
END DO.
```

As the usual two GET statements are not necessary, in BAL, they are not used in the pseudocode. To implement this structure, these BAL statements may be used

```
IN         DTFCD     DEVADDR=SYSRDR,
                     EOFADDR=ENDDO,...
OUT        DTFPR     DEVADDR=SYSLST,...
            .
            .
            .
DOWHILE    GET       IN, INAREA     GET A,B,C,D
            .
            .
            .        PROCESS DATA
            .
            .
           PUT       OUT, OUTAREA   PUT A,B,C,D,X
           B         DOWHILE
ENDDO      NOP       ENDDO
            .
            .
            .
```

For those not familiar with the details of BAL, a few remarks on this example are in order. Note first that the structure of the pseudocode is clearly visible in the BAL statement. Contrast this with the usual method employed, where the GET would be labeled LOOP. This certainly shows the beginning of the loop; but where is the end of the loop? The B DOWHILE is the statement that creates the loop. An unconditional branch to DOWHILE occurs whenever this statement is reached. But what stops the loop? The DTFCD macro entry EOFADDR=ENDDO specifies the address to which the program branches when an end of file condition occurs on file IN. Details of the storage associated with the GET and PUT macros may be seen in the program example in the Appendix.

Another familiar DO WHILE statement in pseudocode form is

```
DO WHILE A < B
 .
 .
 .   BODY OF LOOP
 .
 .
END DO.
```

Assuming that A and B contain packed decimal numbers, this pseudocode may be implemented in BAL using the Compare Packed instruction.

```
DOWHILE    CP     A,B        A < B
           BH     ENDDO
            .
            .
            .  BODY OF LOOP
            .
            .
           B      DOWHILE
ENDDO      NOP    ENDDO
```

The BH instruction branches to ENDDO if A > B. Again, notice the readability of the structure. The No Operation may appear somewhat strange in the previous two BAL segments. Even though it does not generate any executable code, it must have an operand field to avoid a syntax error. The statement label (ENDDO) of the NOP instruction provides this operand to avoid the syntax error.

Alternation may be accomplished in BAL using similar techniques. For example, the pseudocode

```
IF X = Y
THEN
 .
 .
 .
ELSE
 .
 .
 .
ENDIF
```

assuming that X and Y contain packed decimal numbers, may be implemented in BAL as

```
IF        CP     X,Y        X = Y
          BE     THEN
          B      ELSE
THEN       .
           .
           .   X = Y CLAUSE
           .
           .
          B      ENDIF
ELSE
           .
           .
           .   X ≠ Y CLAUSE
           .
           .
ENDIF     NOP    ENDIF.
```

Notice the B ENDIF statement at the end of the THEN clause. This statement insures that the THEN and ELSE clauses are not both executed by branching over the ELSE clause to ENDIF when the THEN clause is reached.

An objection to this implementation might be that it uses one more branch instruction than the "unstructured" BAL program would. A shorter implementation is then

```
IF        CP     X,Y        X = Y
          BNE    ELSE
THEN       .
           .
           .   X = Y CLAUSE
           .
           .
          B      ENDIF
ELSE       .
           .
           .   X ≠ Y CLAUSE
           .
           .
ENDIF     NOP    ENDIF.
```

This implementation is no longer than the unstructured version except for the NOP instruction. As with the ENDDO, this ENDIF label serves to delimit the control structure and greatly increase the readability of the BAL program.

If the ELSE clause is not present, the BAL implementation becomes

```
IF        CP     X,Y        X = Y
          BNE    ENDIF
THEN       .
           .
           .
ENDIF     NOP    ENDIF
```

or

```
IF        CP     X,Y        X = Y
          BE     THEN
          B      ENDIF
THEN       .
           .
           .
ENDIF     NOP    ENDIF
```

The choice between the above forms is left to the student. The two branch method is easier for most students to understand. For those who think that the two branch method is inefficient, the single branch method is available. In either case, the resulting code is readable.

What happens when more than one IF THEN ELSE ENDIF structure appears in a program? Certainly, other labels are needed to avoid duplicate statement labels. What about nested IF THEN structures? This pseudocode

```
IF      A < B
THEN    IF B < C
        THEN
         .
         .
         .
        ELSE
         .
         .
         .
        ENDIF
ELSE    IF C < D
        THEN
         .
         .
         .
        ELSE
         .
         .
         .
        ENDIF
ENDIF
```

may be implemented in BAL as

```
IF      CP     A,B        A < B
        BNH    ELSE
THEN    CP     B,C
IF1     BNH    ELSE1      B < C
         .
         .
         .     A < B, B < C
         .
         .
        B      ENDIF1
ELSE1    .
         .
         .     A < B, B ≥ C
         .
         .
ENDIF1  NOP    ENDIF1
        B      ENDIF
ELSE    CP     C,D        C < D
IF2     BNH    ENDIF2
THEN2    .
         .     A ≥ B, C < D
         .
         .
        B      ENDIF2
ELSE2    .
         .     A ≥ B, C ≥ D
         .
         .
ENDIF2  NOP    ENDIF2
ENDIF   NOP    ENDIF.
```

Certainly, this BAL is more difficult to follow than previous examples, but the high level control structures are still visible. Contrast this example with this equivalent BAL code

```
        CP     A,B
        BNH    X
        CP     B,C
        BNH    Y
         .
         .
         .
        B      P
Y        .
         .
         .
        B      P
X       CP     C,D
        BNH    P
         .
         .
         .
        B      P
         .
         .
         .
P
```

Which of the last two examples do you think a student is more likely to get working? While the first example may be tedious to follow, the second equivalent example is certainly more difficult to follow.

Assignment statements need only be properly commented to assure their readability. For example, the pseudocode

```
X = (A + B) * C/D
```

may be written in BAL

```
ZAP    X,A     X = A
AP     X,B     X = A + B
MP     X,C     X = (A + B) * C
DP     X,D     X = (A + B) * C/D
```

The evolving assignment statement is shown in the comment field to completely document the BAL statements. Details of the Define Constant statements associated with the packed decimal instructions may be seen in the program in the Appendix.

Other Control Structures

While the set of control structures shown so far is a sufficient set to write any program, other control structures are useful. The REPEAT UNTIL, CASE, and iterated DO are shown here. The PASCAL-like REPEAT UNTIL structure in pseudocode form is

```
REPEAT
 .
 .
 .
UNTIL A = B
```

In BAL we have

```
REPEAT    NOP    REPEAT
           .
           .
           .     BODY OF LOOP
           .
           .
          CP     A,B
UNTIL     BNE    REPEAT     A = B.
```

Again, the beginning and the end of the loop are both clearly visible to make a readable BAL program.

A CASE structure is shown by the pseudocode

```
BEGIN CASE
CASE A = 0
 .
 .
 .
CASE A < 0
 .
 .
 .
CASE A > 0
 .
 .
 .
END CASE.
```

In BAL we might have this implementation

```
BEGINCASE    ZAP    X,A
             BZ     CASEØ
             BM     CASELTØ
             BP     CASEGTØ
CASEØ         .                  A = 0
              .
              .
             B      ENDCASE
CASELTØ       .                  A < 0
              .
              .
             B      ENDCASE
CASEGTØ       .                  A > 0
              .
              .
ENDCASE      NOP    ENDCASE.
```

The structure of this code is clearly shown by the statement labels and the comment fields.

The iterated DO structure is also easily implemented in BAL. For example, if the pseudocode is

```
DO I = 1 TO 75 BY 2
 .
 .
 .
ENDDO
```

The BAL implementation might be

```
          L      6, =F'1'      R6 = 1 TO 75 BY 2
DO        C      6, =F'75'
          BH     ENDDO
           .
           .
           .
          A      6, =F'2'
          B      DO
ENDDO     NOP    ENDDO
```

An advantage of the pseudocode approach is that any known high level structure may be used in pseudocode. Whatever structures are preferred may be implemented in BAL. Thus, in problem solving students are not limited to the structures of any past, present, or future high level language. For example, the structure below could well replace the case structure of a previous example

```
IF A = 0
THEN
 .
 .
 .
ELSE1 A < 0
 .
 .
 .
ELSE2 A > 0
 .
 .
 .
ENDIF.
```

The BAL implementation is straightforward

```
IF        ZAP    X,A
          BZ     THEN
          BM     ELSE1
          BP     ELSE2
THEN       .                  A = 0
           .
           .
          B      ENDIF
ELSE1      .                  A < 0
           .
           .
          B      ENDIF
ELSE2      .                  A > 0
           .
           .
ENDIF     NOP    ENDIF.
```

This is easier for some students to follow than the case structure. Thus, absolutely any useful structure can be used in problem solving and then implemented in a readable way in BAL or another assembly language. In addition to control structures, other documentation standards are used to improve readability.

Other Documentation Standards

A most important inclusion in any program is a header to explain the purpose of the program. Simple programs only require a line or two to briefly describe what the program does. The pseudocode itself may be included as comments at the top of the program in order to document the program. Both the input and output formats may be described. An appropriate place to describe these formats is where they are declared. Interior comments in the program are, for the most part, unnecessary. The structure stands out enough to make the program readable without separate comment lines.

All of these documentation techniques are illustrated in the sample program included in the Appendix. This program is self-documenting. Its purpose and flow of control are easy to follow even for readers who do not know BAL.

Summary and Conclusions

By solving the problem first using structured pseudocode and them "compiling" the pseudocode into assembly language, students can write readable self-documenting programs beginning with their first simple assignments. As the problems become more complex, the advantages of the approach described here become even more obvious. A side benefit is the availability of absolutely any control structure in the pseudocode. Students are not restricted to the structures of any single high level language.

Previous to taking assembly language, it is extremely useful, but not absolutely necessary, if the students have taken an introductory course using the pseudocode approach to problem solving. If they have never seen any of the pseudocode structures before, they must learn pseudocode and BAL at the same time. Most students do, in fact, accept this approach readily, perhaps because it makes it much easier for me and the consultants to help them debug their programs. This approach minimizes the number of runs it takes to produce a working program. In addition to assembly language this approach of pseudocoding the solution and then manufacturing appropriate control structures can be applied to any high level language such as FORTRAN or BASIC that lacks some useful control structures (for example see reference 3).

In conclusion, I would urge prospective authors and editors of BAL textbooks to consider using this approach in your books. The resulting books will be greatly enhanced and perhaps even more marketable.

Acknowledgements

I would like to thank my students for their feedback while I have been evolving this approach to teaching assembly language. In particular I would like to thank Ms. Kris Payte, who wrote the sample BAL program in the Appendix. Comments of the reviewers of this paper are gratefully acknowledged.

References

1. IBM Corporation, "System 370 Principles of Operation."
2. Yarmish and Yarmish, Assembly Language Fundamentals, Addison-Wesley, 1979.
3. Cook, Robert N., "Structured Programming Using BASIC," SIGCSE Bulletin, Vol. 12, No. 1, (Presented at Eleventh Annual SIGCSE Symposium).

```
ADDR2  STMT   SOURCE STATEMENT                                         DOS/VSE ASSEMBLER 12.14 

          1 ***********************************************************************
          2 *         THIS IS THE THIRD PROGRAMME DUE IN CST 210, FALL OF 1981     *
          3 *         IT IS A PROGRAMME TO CALCULATE SALARY FOR SALESMEN DEPENDING*
          4 *         ON THEIR SALARY BASE AND BONUS, IF ANY.  WRITTEN BY K TSPB  *
          5 ***********************************************************************
          6 *         PSEUDOCODE                                                   *
          7 ***********************************************************************
          8 *
          9 *                   OPEN FILES
         10 *                   PRINT HEADERS
         11 *                   DOWHILE DATA EXISTS
         12 *                         GET SALNO,SALARY, SALE AMOUNT
         13 *                         IF SALES>=5000
         14 *                         THEN BONUS='YES'
         15 *                              SALARY=SALARY+200
         16 *                         ELSE IF SALES>=1000
         17 *                              THEN BONUS='YES'
         18 *                              SALARY=SALARY+100
         19 *                              ELSE BONUS=' NO'
         20 *                              ENDIF
         21 *                         ENDIF
         22 *                   ENDDO
         23 *                   PRINT SALNO,SALARY,AMOUNT,BONUS
         24 *                   CLOSE FILES
         25 *                   END
         26 *
         27           PRINT NOGEN
         28           START
         29 *
         30 BEGIN     BALR  11,0                                   SET UP BASE REGISTER
00002    31           USING *,11
         32 *
         33           OPEN  IN,CUT                                 OPEN FILES
         42           PUT   OUT,BLANKS                             PRINT BLANK LINE
         48           PUT   OUT,BLANKS                             PRINT BLANK LINE
         54           PUT   OUT,BLANKS                             PRINT BLANK LINE
         60           PUT   OUT,HEADER1                            PRINT HEADER
         66           PUT   OUT,BLANKS                             PRINT BLANK LINE
         72           PUT   OUT,BLANKS                             PRINT BLANK LINE
         78           PUT   CUT,HEADER2                            PRINT COLUMN HEADER
         84           PUT   OUT,HEADER3                            PRINT COLUMN HEADER
         90           PUT   OUT,BLANKS                             PRINT BLANK LINE
         96 *
         97 DOWHILE   GET   IN,INAREA                              SALNO,SAL,SALE AMT
00203   103           MVC   OSALNO,SALNO                           MOVE SALNO TO PRINT
002C7   104           PACK  PSAL,SAL                               PACK SALARY AMOUNT
0020A   105           PACK  PAMT,AMT                               PACK AMOUNT OF SALE
00532   106           PACK  PBBSALE,BBSALE                         PBBSALE=P'5000'
00536   107           PACK  PBSALE,BSALE                           PBSALE=P'1000'
0051C   108 IF        CP    PAMT,PBBSALE                           SALES=5000
        109           BL    ELSE
0053A   110 THEN      PACK  PBBON,BBON                             PBBON=P'200'
00524   111           AP    PBBON,PSAL                             SALARY=SALARY+200
0052E   112           UNPK  OPAY,PBBON
00318   113           MVZ   OPAY+3(1),OPAY+2                       FIX LAST DIGIT
0020A   114           MVC   OAMT,AMT                               MOVE OAMT TO PRINT
00542   115           MVC   OACK,YES                               BONUS='YES'
        116           B     ENDIF
        117 ELSE      NOP   ELSE
00520   118 IF1       CP    PAMT,PBSALE                            SALES=1000
        119           BL    ELSE1                                  SALES LESS THAN 1000
0053E   120           PACK  PSBON,SBON                             PSBON=P'100'
00524   121 THEN1     AP    PSBON,PSAL                             SALARY=SALARY+100
0052A   122           UNPK  OPAY,PSBON
00318   123           MVZ   OPAY+3(1),OPAY+2                       FIX LAST DIGIT
0020A   124           MVC   OAMT,AMT                               MOVE OAMT TO PRINT
00542   125           MVC   OACK,YES                               BONUS='YES'
        126           B     ENDIF
0020A   127 ELSE1     MVC   OAMT,AMT                               MOVE OAMT TO PRINT
00207   128           MVC   OPAY,SAL                               MOVE OPAY TO PRINT
00545   129           MVC   OACK,NO                                BONUS=' NO'
        130 ENDIF1    NOP   ENDIF1
        131 ENDIF     PUT   OUT,OUTAREA                            PRINT OUTPUT
        137           B     DOWHILE
        138 *
        139 ENDDO     CLOSE IN,CUT                                 CLOSE THE FILES
        148           EOJ                                          END OF JOB
        151 *
```

```
ADDR2  STMT   SOURCE STATEMENT                                          DOS/VSE ASSEMBLER 12.14 8

        152 *
        153 ***********************************************************************
        154 *           INPUT LENGTH DEFINITIONS                                  *
        155 ***********************************************************************
        156 *
        157 IN        DTFCD  DEVADDR=SYSRDR,                                          *
                             IOAREA1=INA,                                             *
                             EOFADDR=ENDDO,                                           *
                             BLKSIZE=80,                                              *
                             WORKA=YES,                                               *
                             RECFORM=FIXUNB
        178 INA       DS     CL80
        179 *
        180 INAREA    DS     0CL80
        181           DS     CL1
        182 SALNO     DS     CL4
        183 SAL       DS     CL3
        184 AMT       DS     CL4
        185           DS     CL68
        186 *
        187 ***********************************************************************
        188 *           OUTPUT LENGTH DEFINITIONS                                 *
        189 ***********************************************************************
        190 *
        191 OUT       DTFPR  DEVADDR=SYSLST,                                          *
                             IOAREA1=OUTA,                                            *
                             BLKSIZE=132,                                             *
                             WORKA=YES,                                               *
                             RECFORM=FIXUNB
        212 OUTA      DS     CL132
        213 *
        214 OUTAREA   DS     0CL132
        215           DC     CL4' '
        216 OSALNO    DS     CL4
        217           DC     CL5' '
        218 OPAY      DS     CL4
        219           DC     CL5' '
        220 OAMT      DS     CL4
        221           DC     CL3' '
        222 OACK      DS     CL3
        223           DC     CL100' '
        224 *
        225 ***********************************************************************
        226 *           HEADER LINES                                              *
        227 ***********************************************************************
        228 *
        229 HEADER1   DC     CL132'      SALESMAN SALARY AND BONUS'
        230 HEADER2   DC     CL132'   SALESMAN                AMOUNT'
        231 HEADER3   DC     CL132'    NUMBER     SALARY     SOLD   BONUS'
        232 *
        233 ***********************************************************************
        234 *           PACKED AREA DEFINITIONS                                   *
        235 ***********************************************************************
        236 *
        237 PBBSALE   DS     PL4                      BIG BONUS SALE OVER 5000
        238 PBSALE    DS     PL4                      BONUS SALE OVER 1000
        239 PSAL      DS     PL3                      SALESMANS SALARY
        240 PAMT      DS     PL3                      AMOUNT OF SALES
        241 PSBON     DS     PL4                      SMALL BONUS =$100
        242 PBBON     DS     PL4                      BIG BONUS =$200
        243 *
        244 ***********************************************************************
        245 *           SPACE DEFINITIONS FOR OTHER VARRIABLES                    *
        246 ***********************************************************************
        247 *
        248 BBSALE    DC     C'5000'
        249 BSALE     DC     C'1000'
        250 BBON      DC     C'0200'
        251 SBON      DC     C'0100'
        252 YES       DC     C'YES'
        253 NO        DC     C' NO'
        254 BLANKS    DC     CL132' '
        255 *
 00000  256           END    BEGIN
        257                  =C'$$BOPEN '
        258                  =C'$$BCLOSE'
        259                  =A(OUT)
        260                  =A(BLANKS)
```

DIAGNOSTICS AND STATISTICS

```
NO ERRORS FOUND

THE FOLLOWING MACRO NAMES HAVE BEEN FOUND IN MACRO INSTRUCTIONS
   OPEN      PUT       GET       CLOSE     EOJ       DTFCD     DTFPR

OPTIONS FOR THIS ASSEMBLY - ALIGN, LIST, NOXREF, LINK, NORLD, NODECK, NOEDECK

THE ASSEMBLER WAS RUN IN 65416 BYTES
END OF ASSEMBLY
```

SALESMAN SALARY AND BONUS

SALESMAN NUMBER	SALARY	AMOUNT SOLD	BONUS
1234	800	500	NO
1235	600	600	NO
1236	400	800	NO
1237	1000	2000	YES
1238	1099	4000	YES
9964	0525	8000	YES