N73-19210

AUTOMATED DOCUMENTATION OF AN ASSEMBLY PROGRAM

Valerie L. Thomas
NASA Goddard Space Flight Center

The programmer's burden is greatly reduced if he is willing to invest some time in setting up his program so that it can be used as input to a program which will automatically document it. This paper will present ideas of how this could be implemented for an assembly language program; the examples will be from a META-SYMBOL program which is run on the XDS 930 computer. For the purpose of this paper, the documentation program will be called DOCMNT.

Before the characteristics of DOCMNT can be explained, some format rules must be established for the assembly language program so that DOCMNT can scan the input program (see fig. 1) and find all the information that is necessary for the documentation. The program should be divided into four major parts: identification, main routine, subroutines, and data.

The identification should consist of comment cards having asterisks in the first column and in the second column information pertaining to the program, such as, the name of the program, the programmer, and the operating system. DOCMNT scans these card images looking for the following format:

```
*   PROGRAM–OFOINT
*   PROGRAMMER–VALERIE L. THOMAS
*   OPERATING SYSTEM–MULTISATELLITE OPERATING SYSTEM (MOS)
*   MACHINE AND LANGUAGE–XDS 930, META-SYMBOL
*   DATE–JULY 14, 1970
```

DOCMNT gets OFOINT, VALERIE L. THOMAS, etc., and stores this information in the appropriate place in the documentation. (See fig. 2 for the format of the documentation.) DOCMNT would have the capability of storing OFOINT MAIN beside ROUTINE and of storing for each of the subroutines of the program, the subroutine name beside the word SUBROUTINE; for example:

```
Main routine                      Subroutine

ROUTINE:   OFOINT MAIN            SUBROUTINE:  COMP
PROGRAM:   OFOINT                 PROGRAM:     OFOINT
```

The second major part of the program is the main routine. This should be preceded by some comment cards, one of which contains the words FUNCTIONAL DESCRIPTION and is followed by the description of the main routine; for example:

PRECEDING PAGE BLANK NOT FILMED

```
*       PROGRAM—OFOINT
*       PROGRAMMER—VALERIE L. THOMAS
*       OPERATING SYSTEM—MULTI-SATELLITE OPERATING SYSTEM (MOS)
*       MACHINE AND LANGUAGE—XDS 930, METASYMBOL
*       DATE—JULY 14, 1970
*       FUNCTIONAL DESCRIPTION—OFOINT PROMPTS
*          THE OPERATOR AND USES THE ANSWERS
*          TO UPDATE THE PROMPT TABLE.

STRT    PZE
        LDX     MOSINK
        LDA     1,2
        STA     COMPU
        MIN     STRT+2
        MIN     STRT+3
        MIN     GTCTR
        LDA     GTCTR
        SKG     =30
        BRU     STRT+2
PROMTS  BRM     COMP
        BRM     CHNREQ      COMPUTER
                            CHANNEL REQUEST

        LDA     TABCTR
        SKE     =506
        BRU     LDCA-1
        BRM     MOSXIT

*       FUNCTIONAL DESCRIPTION—COMP DETERMINES WHICH COMPUTER
*       IS BEING USED. DEFAULT IS COMPUTER 1.

COMP    PZE
        LDA     =0
        STA     PRMNO
        LDX     =-20
        LDA     PROTS,2
        BRM     PRNTT
        BRM     MOSREQ
CM1     PZE     TYPOT
        PZE     RESERVE
        PZE     =20
        .
        .
        .
        .
        .
        .
        BRU     CM
*       ERROR CONDITIONS—COMP IS NOT EQUAL
*          TO 1 OR 2. "MESSAGE GARBLED"
*          IS TYPED OUT.
        BRM     GARB
        LDX     PRMNO
        BRU     PROMTS,2
        BRR     COMP

*
```

Figure 1.—Sample program.

```
        .
        .
        .
*       DATA
DISA    DATA   044160000
DISB    DATA   012160000
TIME    RES    2

        .
        .
        .
*       DATA   FORMAT-THE PROMPT TABLE
COMPU   DATA   01         1
CHANL   DATA   022        2
WALL    DATA   045        3
*       DATA   FORMAT-THE CALIBRATION TABLE
A4L1M   DATA   1,0,51
        DATA   0,51
A5L1M   DATA   2,0,0377
*       DATA   0,128,255

        .
        .
        .
        END
```

Figure 1 (continued).–Sample program.

```
*     FUNCTIONAL DESCRIPTION–OFOINT PROMPTS
*        THE OPERATOR AND USES THE ANSWERS
*        TO UPDATE THE PROMPT TABLE
```

DOCMNT continues to scan the program and picks up this information and stores it in the documentation under the heading FUNCTIONAL DESCRIPTION.

DOCMNT will follow the same format for documenting the main routine and each of the subroutines. The argument to the subroutine and the output from the subroutine are assumed to be in the A register. If there is more than one argument, DOCMNT will check for a comment card containing a key (OP1, OP2, OP3, or OP4) to locate the argument list; for example:

```
*  OP1
*  OP2
```

When the key is found, the following information is known:

OP1	A calling sequence was used.
OP2	The argument list follows the instruction (branch to the subroutine).
OP3	There is a pointer to the argument list in the A register.
OP4	The programmer must insert additional comment cards to explain where the input and output arguments are located.

IDENTIFICATION

ROUTINE–OFOINT MAIN
PROGRAM–OFOINT
OPERATING SYSTEM–MULTI-SATELLITE OPERATING SYSTEM (MOS)
PROGRAMMER–VALERIE L. THOMAS
MACHINE AND LANGUAGE–XDS 930, META-SYMBOL
DATE–JULY 14, 1970

ARGUMENT LIST: NONE

FUNCTIONAL DESCRIPTION: OFOINT PROMPTS THE OPERATOR AND USES THE ANSWERS TO UPDATE THE PROMPT TABLE.

METHOD

(1) THE PROMPT TABLE IS PICKED UP FROM OFORUN.
(2) A SUBROUTINE IS EXECUTED FOR EACH PROMPT AND THE PROMPT TABLE IS UPDATED.

READ/WRITE

R/W	DATA AREA OR DEVICE	CHARACTERS/WORD
NA	NA	NA

REQUIREMENTS

SUBROUTINES USED: COMP, CHNREQ
MEMORY: 20 OCTAL WORDS
RUNNING TIME: 30+ CYCLES

ERROR CONDITIONS: NONE

SUPPORTING INFORMATION

THE PROMPTS ARE PICKED UP FROM THE PROMPT TABLE IN OFORUN, UPDATED, AND STORED BACK IN THE PROMPT TABLE IN OFORUN. THIS ALLOWS THE PROGRAMMER TO SAVE THE PROMPT ANSWERS ONCE THEY HAVE BEEN CHANGED FROM THE DEFAULT VALUES.

IRREGULAR RETURNS FROM SUBROUTINES: NONE

DATA FORMATS

THE PROMPT TABLE

```
COMPU   DATA   01    1
CHANL   DATA   022   2
WALL    DATA   045   3
```

THE CALIBRATION TABLE

```
A4L1M   DATA   1,0,51
        DATA     0,51
A5L1M   DATA   2,0,0377
        DATA   0,128,255
```

EXTERNAL REFERENCES

MOSLNK
MOSXIT

Figure 2.–Automatic documentation format.

The information pertaining to the arguments goes under the heading ARGUMENT LIST. DOCMNT scans the routine for branches outside of the routine and puts the information under the heading IRREGULAR RETURNS FROM SUBROUTINES. DOCMNT stores the identification information and the functional description. It then goes through the remainder of the routine to gather more information for the documentation. It counts the instructions as they are investigated to provide the octal count of the number of words of memory used. It has a table which it consults to determine the minimum number of cycles used for the running time. If the routine contains a loop, an unconditional branch, or a branch to a subroutine, the octal word count will have a plus sign next to it, denoting that the minimum time is calculated. DOCMNT searches the routine for the branches to subroutines and lists the subroutine names beside the heading SUBROUTINES USED. If any input/output is used in the routine, a word can be found which gives information pertaining to the device used, whether it is a read or write, and the number of characters per word. This is stored under the heading READ/WRITE. If the routine contains an error routine, the branch to the error routine should be preceded by the following:

```
*      ERROR CONDITIONS—COMP IS NOT EQUAL
*         TO 1 OR 2. "MESSAGE GARBLED"
*.        IS TYPED OUT.
```

DOCMNT scans for error conditions and stores the information that follows in the documentation under ERROR CONDITIONS. DOCMNT also searches for external references, which are listed under the appropriate heading.

The third major part of the program is the group of subroutines each of which is preceded by comment cards (containing the functional description card) and followed by at least one card with an asterisk in the first column.

The last major part of the program is the section containing the data. This section is preceded by the following card:

```
*    DATA
```

To put some sections or tables from the data under the heading DATA FORMATS (in the documentation), a data format card with comments is inserted ahead of the data and an asterisk card after the data; for example:

```
*    DATA FORMAT—THE PROMPT TABLE
COMPU     DATA    01     1
CHANL     DATA    022    2
WALL      DATA    045    3
*
```

The information for the headings METHOD, SUPPORTING INFORMATION, and EXTERNAL REFERENCES is obtained from card input. The cards in the deck containing the external references can be read in and the information stored into a table of external references until DOCMNT is ready to use it. The cards containing the supporting data can be read in and stored in a table. Because every routine will have cards in the method deck, it is best to read in the cards for each routine as it is being used by DOCMNT.

Flowcharts for each routine are done by AUTOFLOW.

After the documentation is completed, a table of contents can be printed out.

DISCUSSION

MEMBER OF THE AUDIENCE: How do you get the running time for the subroutine?

THOMAS: Each instruction has a certain number of cycles, and we can get an estimate of the minimum time by counting out the number of cycles per instruction. If there is a branch to a routine we take the minimum time, which would be just the instruction and the branches to the routine. If there is a loop, we will just take each instruction within a loop. So if we have a case like that we will put beside the running time a plus, which indicates that it is the minimum time.

MEMBER OF THE AUDIENCE: Do you have anything in the system that forces the programmer to provide the information that you need?

THOMAS: No. At present, this is a theoretical system. But if you do not put it in, you will not get out what you want. If you want something that is detailed, you will put in what is necessary. These keywords can be inserted after the program is written, but this should be done before for a very well-documented program listing besides having the automated documentation.

MEMBER OF THE AUDIENCE: Are there plans to get the system built and to get it operational?

MEMBER OF THE AUDIENCE: For what it is worth, there is a system similar to this in operation in a large industrial organization. It is used as a prelist and a requirement, and it works very well. In other words, the programmers must comply. They have no great turnover because of this, particularly these days. The system actually is not used on keyboard. It is based on COBOL, and it does go through, picks out all comments, notes, and paragraphs; sets it up; and does an absolutely excellent job. They also take this and automate it to a quick-input index, which is distributed worldwide and is used as a catalog library similar to that described by COSMIC.