SEPTEMBER 1975
REVISED: OCTOBER 1975

MDC G5886

NASA CR-144034

INVESTIGATING TECHNIQUES IN THE GENERATION OF SUPPORT SOFTWARE

Final Report
CONTRACT NAS8-30907

MCDONNELL DOUGLAS ASTRONAUTICS COMPANY

MCDONNELL DOUGLAS CORPORATION

PREFACE

The final report for "Investigating Techniques in the Generation of Support Software" is submitted to the National Aeronautics and Space Administration, Marshall Space Flight Center in accordance with the provisions of the contract number NAS8-30907.

During the duration of the contract Mr. Bobby Hodges of NASA/MSFC has provided the necessary direction and supplemented the work with his technical assistance. His effort and direction are very much appreciated and gratefully acknowledged by the project team.

If any additional information is desired, please contact any of the following McDonnell Douglas or NASA representatives as appropriate:

- Mr. Z. Jelinski, Project Manager (MDAC)
  Huntington Beach, California
  Telephone: 714-896-5060

- Mr. A. J. Edwards, Principal Investigator (MDAC)
  Huntington Beach, California
  Telephone: 714-896-3872

- Mr. B. C. Hodges, Project COR (NASA)
  Marshall Space Flight Center, Alabama
  Telephone: 205-453-0134

CONTENTS

Section 1
INTRODUCTION

The Space Ultrareliable Modular Computer (SUMC) is a computer family that is the result of the research and development efforts of the Astrionics Laboratory at the Marshall Space Flight Center (MSFC). Its design incorporates features that ensure flexibility, reliability and speed. The Computation Laboratory at MSFC is developing a generalized SUMC support software system to mee current and future requirements for SUMC software development. The primary objective of this activity is the creation of a highly generalized and flexible software system which is host computer transferable and designed to support the generation of operating systems and application programs for the SUMC.

As part of the overall support software system development, McDonnell Douglas provided an assembler processor under Contract NAS8-27202. This processor is capable of accommodating assembly language statements as well as microcode language statements and producing object module outputs for both languages. This processor, as an integral module of the SUMC support software system, represents the essential capability to respond to unique firmware and instruction set requirements of each project involving the utilization of the SUMC.

Section 2
OBJECTIVE

Since the initial assembler processor has been delivered and installed at MSFC by MDAC, a period of checkout and utilization is required to verify the performance of the processor. During this time it is necessary to have provision for product maintenance and modification requirements during the verification phase. It is therefore the objective of this effort for MDAC to ensure proper assembler processor performance and provide for modifications/enhancements identified during the verification process.

Section 3
TECHNICAL PERFORMANCE

The primary technical tasks of this contract were directed towards maintenance and modification of the Meta Assembler during the initial utilization of the assembler for SUMC software development. The following subsections describe the tasks defined by MSFC and performed under the contract period.

3.1 FORTRAN COMPILER INTERFACE

The SUMC FORTRAN compiler developed at MSFC utilizes the Meta Assembler to generate the machine instructions. It was anticipated during its development that some modifications to the Meta Assembler would be required.

The following modifications were implemented:

1. The external linkage directives were modified such that the LINK directive was replaced with the REF (external reference), DEF (external definition) and BREF (blank common reference) directives.

Label	Operation	Operands
blank	REF DEF BREF	symbol[,symbol....]

This directive change resulted in a redesign of the Global Symbol Dictionary record in the object module.

WORD LENGTH	VALUE	VT	EXTYP	SYMBOL STRING
WORD 1	WORD 2			WORDS 3-n

WORD LENGTH (word 1, bits 31-1)

The number of words n the symbol string.

VALUE (word 2, bits 31-8)

The value assigned by the assembler for the DEF symbol (SBTYP=1).

VT (word 2, bit 7)

The VALUE type of the DEF symbol.

(0=address, 1=value)

EXTYP (word 2, bits 6-1)

The external entry type. (0=REF, 1=DEF, 4=BREF)

SYMBOL STRING (words 3-n)

The REF/DEF/BREF symbol string in packed integer code format.

2. The parsing function was modified to allow a maximum of 15 characters for symbols.

3.2 IBM 360/65 HOST INSTALLATION

It became desireable to rehost the SUMC software development facility (S4) at MSFC on the newly acquired IBM 360/65. As a processor within the S4 system the Meta Assembler was installed on the IBM 360/65. In conformance with the IBM 360 FORTRAN compiler and Linkage Editor functions the following adjustments were made to the Meta Assembler FORTRAN source.

1. EQUIVALENCE statement subscript notations were respecified where necessary due to the FORTRAN compiler requirements.

2. DATA statements were added to the BLOCK DATA module for the Linkage Editor to perform specific COMMON block allocation required by the Meta Assembler for execution.

3.3 TARGET DEFINITION EFFICIENCY CAPABILITY

The original design of the Meta Assembler required the target definition source statements to be processed for each assembly. At MSFC it became apparent that multiple assemblies (back-to-back) all utilizing the same target definition resulted in time consuming redundant processing. The Meta Assembler was subsequently modified to reduce the target definition statement processing by saving the COMMON blocks as initialized by the target definition the first time it is processed, then restoring the COMMON blocks for subsequent assemblies in the same job step (back-to back). This was implemented for both machine level and micro level target definitions without restrictions to the sequence of back-to-back assemblies.

3.4 GENERAL ENHANCEMENTS

This task was the upgrading of the MSFC Meta Assembler S4 version with general enhancements implemented in the Meta Assembler standard version.

1. TWORDS directive to control order of characters in textual string object.

2. GOIF, TYPE directive for symbol type determination.

3. Deferred addressing notation.

4. Improved DATA processing for negative format and multiple precision.

5. Inclusion of logical operators .AND. and .OR. in expressions.

6. Error check for duplicate micro mnemonic specification on the same statement.

7. DATA bounding function parameterized.

   QTARGV(11)machine, QTARGV(14) micro: 0 = no bounding, 1 = bounding

8. Source line continuation parameterized

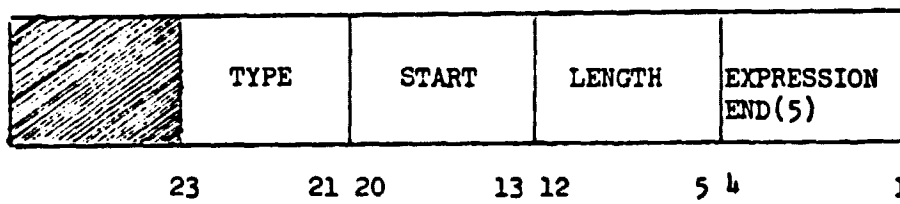
QTARGV(12) machine	0 = continuation
QTARGV(13) micro	1 = no continuation

This applies only to the assembly portion of the source input, not the machine definition segment where continuation is always in effect.

9. EFORM/FFORM pseudo operations brougnt up to current specifications.

10. Object format modification

The last word of a polish term block now contains a field in bits 21-23 which describes the addressing scheme specified in the IFORM directive format:



3.5 MSFC S4 MODIFICATIONS

This task represents the MSFC requested modifications and error corrections to the S4 Meta Assembler.

Modifications

1. The assembly listing has been extended to include TITLE,EJECT and SPACE directives.

2. The LIBEND statement terminating the machine definition phase has been deleted from the listing.

3. The printer eject at the start of the listing has been inhibited.

4. The setting of QCOMA(12) has been modified to reflect the disc pointer to the first physical record after GSD section output.

5. Replacing 'SUMC' in the title line with 'MSFC S4'.

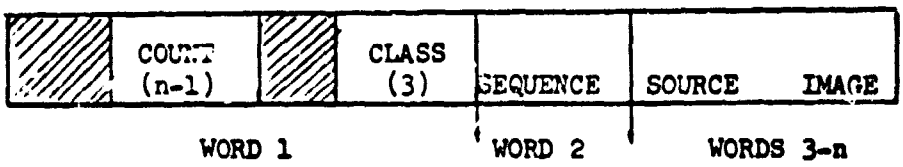
6. The setting of QCOMA(15) has been implemented to reflect the status of the object module after the GSD section. (0=no VSD and/or TXT,1=VSD and/or TXT).

7. The assembler has been modified to abort when the machine definition library module is not found.

8. The listing output format has been modified as follows:

Print Column(s):		
2-5	decimal card number	
6	blank	
7-11	location counter	
12	blank	
13	type (R,E,etc.)	
14	blank	
15-(n+14)	object (where n is the number of characters required to print the object code)	
N+15	blank	
(n+16) - 114	source card image (col. 1 of card printed in Col. n+16. Card truncated from Col. 80 towards Col. 1 in print line if Col. 80 cannot be printed in Col. 114 or less).	
115	blank	
116 - 119	line # (decimal)	
120	blank	Columns 81-93 of source
121 - 132	modification history	

9. The source card object output for micro assembly has been modified as follows:



COUNT (word 1, bits 16-6)

The number of words remaining in the current source image record: (n-1)

CLASS (word 1, bits 4-1)

The text source image subtype: 3

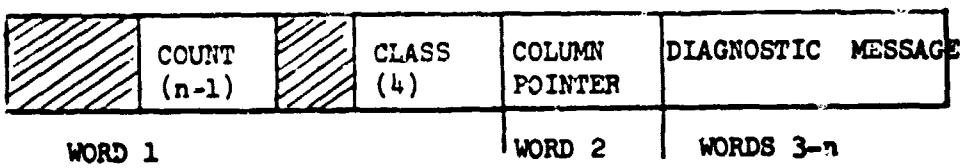
SEQUENCE (word 2, bits 31-1)

The source card sequence number displayed in columns 2-5 of the assembly listing. A value of zero is used for source images that are listed with columns 2-5 blank.

SOURCE IMAGE (words 3-n)

The source image representing 96 characters in packed integer format.

10. The diagnostic messages are generated as a subtype of the TXT object section for micro-assembly as follows:



COUNT (word 1, bits 16-6)

The number of words remaining in the current diagnostic message record: (n-1)

CLASS (word 1, bits 4-1)

The test diagnostic message subtype: 4

COLUMN POINTER (word 2, bits 31-1)

The error column pointer value associated with the diagnostic. A value of zero denotes no error column pointer.

DIAGNOSTIC MESSAGE (words 3-n)

The diagnostic message image representing 132 characters in packed integer format.

Error Correction

1. Erroneous line number (0) cross reference listing for symbols defined in the machine definition and not reference in the assembly.

2. Line number error (actual line no.+1) in the cross reference for symbols used on an ORG statement.

3. Divide check detected on IBM 360.

4. PROC expansion during an assembly where the machine definition was restored from disk.

5. Utilize QHOSTV (19) for record length instead of 512 in subroutine MDEFIO.

3.6 USER MANUAL PUBLICATION

During this contractual effort the User Manual underwent a rewrite, review by MSFC, publication and delivery to MSFC.

Section 4
SPECIAL PURPOSE SYNTAX COMPATIBILITY

While the Meta Assembler was initially developed for the SUMC computer family, this does not preclude its application as a general purpose cross assembler for numerous target computers. When it is desired to configure the Meta Assembler for a target computer that has a resident assembler (e.g. PDP-11, RAYTHEON 706) a syntax compatibility problem becomes apparent. This is due to the ad hoc development of assembler languages. Each assembler language definer is free to implement the syntactic structure of his assembler statements as he desires. Assembler languages are composed of directives and machine instructions.

Assembler Directives

Since the semantics of assembler directive statements are normally consistent (e.g. data declaratives, location counter control, listing control, parameter definition) there is a prolipheration of syntactic structures for these assembler directives. The following illustrates equivalent assembler data declarative statements:

```
 DC        F'5'        IBM-360
.WORD      5           PDP-11
DATA       5           RAYTHEON 706
D          5           RAYTHEON 706
FIX        5           BENDIX BDX-920
```

Machine Instructions

The machine instructions directly reflect hardware characteristics therefore, the instruction set mnemonic operations and operands represent special purpose assembler statements. The syntactic structure for machine instruction operands often utilize special characters for notation. For example:

```
L    5,ABC(4)
MOV  (%2)+,-(%4)
```

However, the object values generated by the assembler for machine instructions are comprised of subfields that have commonality among different machines (e.g. operation code, index register, address, indirect, etc.).

Meta Assembler Design Objective

The Meta Assembler was designed to perform cross assembly via equivalent assembler directives and machine instruction statements. The target definition capability provides flexibility at the machine instruction level only. The mnemonic operations are defined to the Meta Assembler while the operands are designated by type and must conform to syntax rules for source notation. The assembler directive level statements are completely built into the Meta Assembler and are not available for redefinition. Therefore, a source program prepared for a target resident assembler (e.g. PDP-11) must be modified syntactically for the Meta Assembler to perform an equivalent cross assembly.

For example:

PDP-11	Meta Assembler
.=1000	ORG O'1000'
A=10	A EQU O'10'
B=100.	B EQU 100
.WORD A,B	DATA A,B
LI: MOV (%2),(%6)	LI MOV @2,6
.PAGE	EJECT

Meta Assembler with Syntax Modifier

The approach toward solving this syntax compatibility problem is to utilize the MDAC proprietary Meta Translator to produce a syntax modifier for a special purpose assembler language. The function of the syntax modifier is to parse the special purpose assembler language and transform the statements syntactically such that equivalent statements are given to the Meta Assembler. The utilization of the Meta Translator provides an automated tool for the syntax modifier development and since it is used to produce the Meta Assembler itself the syntax modifier is produced as an integral part of the Meta Assembler version. Integration of the syntax modifier within the Meta Assembler provides the following benefits:

- Internal syntax modification without additional I/O as with stand alone pre-processing.

- Line by line pre-procesing during assembly provides maximum information to syntax modifier (e.g. symbol table).

- Assembly listing reflects the input source images rather than the modified source statements.

- Extends the Meta Assembler capability rather than restricts since the syntax modifier is optionally invoked at assembly time.

Section 5
CONCLUSIONS

During the performance of this contract MSFC has conducted the verification of the Meta Assembler in conjunction with the initial application utilization.

The significant conclusions of this effort strongly support the original concept of the Meta Assembler as a target reconfigurable support software tool.

4.1 META ASSEMBLER IMPROVEMENTS

The enhancements, modifications and error corrections have increased the reliability of the assembler as well as extended its target capability.

4.2 META ASSEMBLER APPLICATIONS

MSFC has configured the Meta Assembler to provide cross assembly capability for various target computers. Not only did this illustrate the flexibility of the target definition capability but also the rapid implementation that was accomplished by MSFC personnel.