F6

TO: MPN Distribution

FROM: R.M. Graham

DATE: November 6, 1968

SUBJ: Multics Language Processors

The attached report is a first attempt to document the status of all the language processors which are currently being implemented for Multics. Each section describes one language processor, giving status information for each of its major tasks. The report begins with a chart sumarizing the status of all the language processors.

Since this is the first issue, the information is incomplete. This report will be reissued frequently (at least once every two weeks). The following information is included this time for each language processor:

1. Breakdown into tasks.
2. The language(s) in which the procedures of a task are coded (or will be coded).
3. Personnel presently assigned to each task.
4. Percent of task which has been specified.
5. Percent of task coded.
6. Percent of task unit checked.
7. Percent of task integrated into its own subsystem in Multics.
8. Final integration into Multics. When this is done it means that the processor actually works as a Multics command in a standard system.
9. Percent of completed documentation for the task.
10. Date the language processor will be submitted to the standard library.
11. Performance data. This information is not always comparable. For compilers it is measured in statements (or lines) compiled per minute. (For interpretive processors it is measured in statements executed per minute and is not very meaningful.)

Whenever a percentage is called for, if the figure is less than 100 percent the projected completion date is given, if known. A question mark (?) indicates a reasoned guess.

An attempt will be made to include the following information in future reports:

1. Size information: number of modules and their size.
2. Breakdown of documentation into implementation and user interface.
3. Qualitative appraisal of status of each processor.
4. List of the most important problems which remain to be solved.
5. Information on unscheduled, but desirable, programs, e.g., rework of binder, rework of EPLBSA, Fast Fortran.

Task	Language	People Working	Specified	Coded	Unit Checked	Integrated	Standard Library	MSPM or Other	Lines Per Minute	Note
MLP.1	AED	5	12/1/68	? 10%	? 2%					
MLP.2	BASIC	1	done	11/11/68 ? 90%	11/18/68 ? 90%	11/25/68	12/1/68		(NA)	
MLP.3	BCPL	2	done	done	done	done	done	? 75%	300	1
MLP.4	EPL	1	done	done	11/8/68 ? 40%	11/15/68	11/18/68	done		
MLP.5	EPLBSA	0	done	done	done	done	done	done	1200	2
MLP.6	FL	1	done	done	done	done	11/11/68	1/1/69	800	
MLP.7	FORTRAN IV	1	11/11/68	? 75%	? 70%	? 80%			600-1000	2
MLP.8	MAD	3	95%	? 83%	? 68%	? 50%				
MLP.9	PL/I(No optimizer)	5	done	1/1/69 ? 60%	2/1/69 ? 38%	? 28%		1/1/69		
MLP.10	POPS	0	done	done	done	done	11/11/68	done	(NA)	
MLP.11	SNOBOL	0	done	done	done	done	done		(NA)	
MLP.12	TMG (New)	2	done	11/11/68 ? 90%					(NA)	

Notes

1. Does not include assembly time.
2. Estimate from GECOS execution.

MLP.1 AED

The majority of AED is written in AED. The AED compiler has been successfully bootstrapped from one machine to another more than once in the past. Based on this experience it is well known exactly which modules must be modified or re-written.

Complete Integration:

Standard Library:

Performance:

1.1 Character conversion routines: ASCII↔internal code

1.2 Input/Output interface: AED↔Multics

1.3 Run-time support: principally the loader and linkage segment builder.

1.4 Initialization of the lexical phase

1.5 Initialization of the syntactic phase

1.6 Initialization of the semantic phase

1.7 Generation of driving tables

1.8 Code generation

1.9 Redefinition of data structures

1.10 Character string manipulation routines

MLP.1 AED (G. Rodriguez, T. Colligan, S. Ohyain, C. Garman, J. Brackett)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	No
1.1	Character Conversion	AED	T. Colligan	12/1/68					
1.2	I/O Interface	EPL AED	S. Ohyain	12/1/68					
1.3	Run-time support	EPLBSA EPL	C. Garman	done		started			
1.4	Initial Lexical	AED	(none)	12/1/68					
1.5	Initial Syntactic	AED	(none)	12/1/68					
1.6	Initial Semantic	AED	(none)	12/1/68					
1.7	Generate Tables	AED	(none)	12/1/68					
1.8	Code Generation	AED	(none)	12/1/68					
1.9	Redefine data structures	AED	J. Brackett	done		done	? 20%		
1.10	Char string manipulate	AED	J. Brackett	done		done	? 20%		

Notes

1. More than half coded

MLP.2 BASIC

BASIC is a compile and go system. Timing figures are meaningful only for compile and execute together.

Complete Integration: 11/25/68

Standard Library: 12/1/68

Performance: 16000 statements per minute. Based on one test in CTSS which executed 6000 statements in 22 seconds.

2.1 Executive

2.2 Compiler

2.3 Evaluator

2.4 Support routine: string routines, conversion routines

2.5 Improvements and Additions: Principally; arrays, floating point arithmetic, the FOR-statement, the PRINT-statement, and functions.

MLP.2 BASIC (W. Southworth)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Notes
2.1	Executive	BCPL	W. Southworth	done		done	done	done	
2.2	Compiler	BCPL	W. Southworth	done		done	done	done	
2.3	Evaluator	BCPL	W. Southworth	done		done	done	done	
2.4	Support routines	BCPL	W. Southworth	done		done	done	done	
2.5	Improvements and extensions	BCPL	W. Southworth	done		11/11/68 ? 50%	11/18/68 ? 50%	11/25/68	

MPL.3 BCPL

BCPL compiles EPLBSA code

Complete Integration: done

Standard Library:

Performance: 300 lines per minute (not including EPLBSA time)

3.1 Syntax module

3.2 Translate module

3.3 Code Generation

3.4 Command driver

3.5 Run-time support

MLP.3 BCPL (R. Canaday, D. Ritchie)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Note
.1	Syntax	BCPL	D. Ritchie	done		done	done	done	
.2	Translate	BCPL	D. Ritchie	done		done	done	done	
3	Code generation	BCPL	D. Ritchie	done	done	done	done	done	
4	Command driver	EPL	D. Ritchie	done	done	done	done	done	
5	Run-time support	EPLBSA BCPL EPL	D. Ritchie	done	done	done	done	done	

MLP.4 EPL

Complete Integration: 11/15/68

Standard Library: 11/18/68

Performance:

4.1 Command driver

4.2 TMG

4.3 Driving Tables

MLP.4 EPL (N. Adleman)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Not
	Command driver	EPL	N. Adleman	done	done	done	11/8/68 90%	11/15/68	
2	TMG	GMAP	N. Adleman	done	done	done	11/8/68	11/15/68	
3	Driving Tables	TMG GMAP	N. Adleman	done	done	(NA)	(NA)	11/15/68	

MLP.5 EPLBSA

Complete Integration: done

Standard Library:

Performance: 1200 lines per minute. Based on a test in run in GECOS.

5.1 Command driver

5.2 Assembler

MLP.5 EPLBSA

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Note
5.1	Command driver	EPL	(none)	done	done	done	done	done	
5.2	Assembler	FORTRAN GMAP	(none)	done	done	done	done	done	

MLP.6 FL

FL uses the POPS interpreter.

Complete Integration: done

Standard Library: 11/11/68

Performance: 800 lines per minute

6.1 Driving Table needed by POPS.

6.2 Data segment needed by POPS.

6.3 Command driver

MLP.6 FL (M. Meer)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	No
6.1	POPS driving table	POPSL	M. Meer	done	1/1/69	done	done	done	1
6.2	POPS data segment	POPSL	M. Meer	done	1/1/69	done	done	done	1
6.3	Command driver	EPL	M. Meer	done	done	done	done	done	2

Notes

1. An implementation manual (not an MSPM section) scheduled for completion by 1/1/69.
2. A language manual for users (a GOO document) scheduled for completion by 11/1/68.
3. A few minor improvements are currently in unit check and scheduled for integration by 11/11/68

MLP.7 FORTRAN IV

FORTRAN generates machine code directly.

Complete Integration:

Standard Library:

Performance:

7.1 POPS Driving Table

7.2 POPS Data Segment

7.3 Command driver

7.4 Math library

7.5 Input/Output library

MLP.7 FORTRAN IV (K. Shih)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	N
7.1	POPS Driving Table	POPSL	K. Shih	done	done	done			
7.2	POPS Data Segment	POPSL	K. Shih	done		done			
7.3	Command Driver	EPL	K. Shih	done		done			
7.4	Math Library	EPL	K. Shih	done		? 75%	? 50%		
7.5	I/O Library	EPL	(none)	11/11/68					

Notes

1. Non-complex functions are all unit checked; complex are currently being coded.
2. First draft of specification exists.

MLP.8 MAD

MAD generates EPLBSA code.

Complete Integration:

Standard Library:

Performance:

8.1 Command driver

8.2 Pass 1

8.3 Pass 2

8.4 Reductions Loader: Needed to translate and load driving table

8.5 Run-time support: Read data and print results subroutines.

MLP.8 MAD (A. Evans, H. Ancona, V. Voydock)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	N
.1	Command driver		(none)	? 25%					
.2	Pass 1	BCPL	H. Ancona	done		done	done		
.3	Pass 2	BCPL	H. Ancona	done		90 %	80 %		
.4	Reductions Loader	BCPL	(none)				done		
.5	Run-Time	EPL	V. Voydock	done		60 %			

Notes

1. Read data about 95 percent coded and print results about 25 percent coded.

MLP.9 PL/I

PL/I will generate machine code directly.

Complete Integration: ? 5/1/69 earliest date for 9.8 completion.

Standard Library:

Performance:

9.1 Lexical Phase

9.2 Syntactic Phase

9.3 Declaration Processor

9.4 Semantic Phase

9.5 Optimizer

9.6 Code Generation

9.7 Storage Allocation

9.8 Bootstrap initial version of compiler into Multics: First version of compiler only complete enough to compile itself in 6.36. This task consists of getting into Multics and upgrading compiler to complete specification.

MLP.9 (A. Kvilekval, J. Mills, R. Frieburghouse, B. Wolman, Chang)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Note
9.1	Lexical	EPL	J. Mills	done	done	done	done	95%	1
9.2	Syntactic	EPL	J. Mills	done	1/1/69 draft	done	done	95%	1
9.3	Declaration Processor	EPL	R. Frieburghouse	done	1/1/69 draft	done	done	95%	1
9.4	Semantic	EPL	R. Frieburghouse Chang	done	1/1/69 draft	1/1/69 ? 60%	2/1/69		
9.5	Optimizer	EPL	(none)	done	1/1/69 draft				
9.6	Code Generation	EPL	B. Wolman	done	1/1/69 draft	done	1/1/69 ? 40%	start 12/1/68	
9.7	Storage Allocation	EPL	B. Wolman	done	1/1/69 draft	1/1/69 ? 40%	12/1/68		
9.8	Bootstrap		(none)						

Notes

1. Integration of Lexical, Syntactic, and Declaration Processor is in its final stages.

MLP.10 POPS

Complete Integration: done

Standard Library: 11/11/68

Performance:

10.1 Interpreter

10.2 Driving Tables

10.3 Initialized Data segment

MLP.10 POPS

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Note
10.1	Interpreter	FL	(none)	done	done	done	done	done	
10.2	Driving Tables	FL	(none)	done	done	done	done	done	
10.3	Data segment	FL	(none)	done	done	done	done	done	

MLP.11 SNOBOL

Complete Integration: done

Standard Library: done

Performance: 3 to 5 times longer than CTSS. SNOBOL is very highly subroutinized. It is conjectured that the performance degradation is due to the increased time in Multics for subroutine calls.

11.1 Command driver

11.2 Input/Output Interface

11.3 Interpreter and Compiler

MPL.11 SNOBOL

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Note
11.1	Command driver	EPL	(none)	done		done	done	done	
11.2	I/O Interface	EPL	(none)	done		done	done	done	
11.3	Interpreter/Compiler	BCPL	(none)	done		done	done	done	

MLP.12 New TMG

This is a new version of TMG written from scratch for the Multics environment.

Complete Integration:

Standard Library:

Performance:

12.1 Interpreter

12.2 Driving Tables

MLP.12 (D. McIlroy, M. Wagner)

	Task Name	Language Used	Assigned Personnel	Specified	MSPM Sections	Coded	Unit Check	Integrated	Note
12.1	Interpreter	EPLBSA	M. Wagner	done		done			
12.2	Driving Tables	EPLBSA	M. Wagner	done		11/11/68 ? 75%			