Saltzer

Multics Task Report MTR-136

To: Distribution

From: Steve Webber

Date: 10/08/76

Subject: Language Group Status Report

FORTRAN (5.0)

The major FORTRAN tasks for 5.0 center around providing the FAST FORTRAN compiler as the standard Multics FORTRAN compiler. This requires the addition of all currently missing features needed to bring it up to the level of the released software. In particular, the following features, that are currently missing or incomplete, must be provided (C = critical, N = not critical, D = done):

C 1. A debugged, compatible version of fortran_io_ (the runtime I/O program for FORTRAN programs) that will support either new or old FORTRAN object code.

N 2. A cleaned-up mechanism for handling ERR= and END= alternate returns from I/O statements.

D 3. The handling of calls to non-FORTRAN programs that require descriptors.

D 4. The generation of an acceptable listing file.

D 5. Completing the design and documentation of the various I/O conventions, file handling, and the like. This is necessary for the new FORTRAN Manual.

C 6. A symbol table for debugging purposes.

C 7. Handling the NAMELIST feature. (Also requires the symbol table).

D 8. Generate relocation information so the FORTRAN programs can be bound.

C 9. Update the FORTRAN Manual to reflect the language as of MR5.0

Some of these "requirements" could possible be relaxed in that the old FORTRAN compiler will remain in the system.

The old FORTRAN compiler has no required tasks other than to keep up with the current PL/I code generator. It might be a good idea to freeze a copy of the code generator so that future changes to PL/I need not cause changes in old FORTRAN.

LINKER CHANGES (5.0)

The linker must be changed in several ways. The primary changes are changing combined linkage regions into standard PL/I areas and the inclusion of the handling of *system links. The prelinker must be changed in parallel with the linker for these kinds of changes.

The binder must be changed to understand and act appropriately upon *system links. This task will be integrated with the installation of the linker.

Also needed with the new linker facilities are several commands to manage the areas, select which area is to be used for which purposes, list and dump the areas, and manage the "external variables" resulting from *system links.

PL/I (5.0)

The 5.0 PL/I compiler is installed at MIT. The only changes that might be made to this compiler before its release to the field are minor bug fixes.

BASIC AND FAST/DFAST (5.0)

The BASIC and Fast FORTRAN compilers must be changed to use a new compiler_source_info structure which provides the entry name of the source segment (as well as the entry name of the target of the source segment if a link is used). The FAST and DFAST subsystems must be changed to use this new structure.

MISCELLANEOUS TASKS

There are a few miscellaneous task that we would like to get done soon. These include (a) a run command capable of managing run units for PL/I, FORTRAN, and COBOL programs, and (b) a new version of the BASIC runtime programs that optimize string operations where possible.

APL (6.0)

It is our current plan to release the version of APL affectionately known either as v2apl or super APL as the standard APL with MR6.0. This requires extensive rewriting of the APL manual as well as extensive bug fixing. A few new features will also be added making it a much more competitive product. More and more potential Multics customers are listing APL as an important part of a Multics system. Although we will probably not be able to finish the manual until 6.0 release time, we do hope to get an accurate version available for early release for any potential customers.

FORTRAN (6.0)

By the 6.0 release, FORTRAN should have all current features as well as a few new enhancements. The 6.0 features include:

1. The first release of the FORTRAN optimizer. This optimizer will perform many optimizations, but will not attempt any kind of loop optimization.

2. The OPEN and CLOSE statements compatible with what the ANSI FORTRAN people are talking about.

3. Several optimizations in the runtime I/O analogous to those done for PL/I.

4. The handling of implied do loops.

5. Support for the $ format mechanism.

6. Support for double precision complex arithmetic.

7. The -profile and -long_profile options.

8. A rewrite of the code generator macros for possible optimizations.

9. Redo of the statement map for new format maps.

10. Extensions to the language for string handling.

11. Updating the FORTRAN Manual to include new features.

PL/I (6.0)

Among the many changes we would like to get into the 6.0 version of the compiler are most of those proposed by the MSPL committee. However, at this time there are far too many additions and optimizations to get done by 6.0 release time and since it is not yet clear exactly which we will push for, the following list includes more than will be done for MR6.0. Of prime importance, though, is the work to be done in code optimization. This includes mainly the loop optimizer, but also includes such optimizations as special casing certain picture conversions and the redo of the get edit and get list runtime analogous to what was done for output. At any rate, the following list includes more than can be done for MR6.0:

1. Convert any_to_any_ to handle unsigned data. (Maybe for 5.0)

2. Convert any_to_any_ to handle compact (packed) decimal (in whatever form is agreed upon). (Maybe for 5.0)

3. Add the handling of octal (quarternary and hex also) constants to the compiler and the runtime.

4. Change the pl1 command to use get_temp_segments_.

5. Change the compiler and runtime to use the iox_ call forwarders instead of the iocb entry values.

6. Change the compiler to use the standard (no free) allocation code.

7. Change the compiler to use template node values initialized with create_data_segment programs.

8. Add the following builtins to the language:

   stacq
   clock
   vclock
   stack_frame_ptr
   stack_base_ptr
   max_length
   rtrim
   ltrim

9. Integrate the additional math builtins to bring us up to full ANSI level.

10. Reimplement the following builtins for efficiency:

    after
    before
    decat

11. Design, code, and debug the first phase of the loop optimizer.

12. Add several listing enhancements proposed in the MSPL meetings (but not officially adopted).

13. Document all changes to the language in AG94.

14. Optimize the code for fixed binary long (double precision).

15. Optimize the code generator to special case the testing of several bits in a single machine word.

16. Generate a list (for AG94) of differences between our PL/I and ANSI PL/I. It has also been proposed that a compiler option be available that gives a warning when a nonANSI construct is used (there are some we can not detect, though).

17. Add the unsigned attribute to the compiler (initially only for fixed binary data).

18. Add the abnormal attribute to the compiler.

19. Rewrite the lex phase of the compiler taking more advantage of EIS.

20. Change the compiler to generate *system links when appropriate.

21. Change the format of the statement map generated.

22. Special case several cases of picture packing and unpacking.

23. Add the -long_profile option.

24. Implement the STOP statement.

25. Add several formats of packed decimal data to the language. Add the compact attribute to be used for this as well as pointer data.

MISCELLANEOUS TASKS (6.0)

A project has been started aimed at developing a tasking mechanism that will be useful for Multics jobs. This project will first consist of studying a prototype tasking facility with the hope of learning enough to come up with a generally useful facility.

The PL/I standards group has developed a special committee designed to specify a real-time subset language for PL/I. Rules of subsetting PL/I require that anything the committee decides should be standard (and the full committee agrees) must be standard in the full PL/I. We are therefore watching and participating with interest (we are represented on the committee twice -- one chairperson).

Although not yet begun, the task of developing a general text processing macro language still appears to be quite reasonable. Such a high-level language would be very useful for such conversions as are today done with ted, teco, PL/I programs and a combination of all of these.

It has become obvious to some (and still denied by others) that some extension to iox_ is needed to easily handle switch openings and closings. Two new routines have been proposed, iox_$gen_open and iox_$gen_close, that solve some of the problems encountered. We hope to get a document out soon on these proposals so that some action can be taken by 6.0.

GROUP Multics Language Group DATE 11/02/76 PAGE 1 / 1

PROJECT FORTRAN AREA Fast FORTRAN for 5.0

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Design new fortran_io_	Levin Schoeman Webber	09/20/76	10/12/76	____	Done
Code and checkout new fortran_io_	Schoeman	10/04/76	12/15/76	____	
Design new handling of ERR= and END= alternate I/O returns	Barnes Levin Webber	10/11/76	10/18/76	____	Delayed
Code and checkout compiler and runtime for ERR= and END=	Levin Schoeman	10/18/76	10/22/76	____	Delayed
Add descriptor calls to Fast FORTRAN	Barnes	10/04/76	10/15/76	____	Done
Generate -list control arg to compiler	Levin	09/13/76	10/01/76	____	Done
Generate -assembly_list control arg to compiler	Levin Barnes	09/27/76	10/08/76	____	Done
Generate a standard symbol table for FORTRAN programs	Levin	10/11/76	11/15/76	____	
Support NAMELIST statement	Levin	10/25/76	11/15/76	____	
Generate relocation information for FORTRAN programs	Barnes	10/22/76	10/29/76	____	Done
Update the FORTRAN Manual for MR5.0 level.	Levin Webber Boyd	10/22/76	11/05/76	____	Will be late
Generate installation plan for new FORTRAN	Webber	10/11/76	10/15/76	____	
Change compiler to use compiler_source_info include file	Levin	10/24/76	11/31/76	____	Coordination with BASIC and FAST required

GROUP Multics Language Group DATE PAGE 1 / 1

PROJECT Linker Changes AREA Areas and *system links

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Convert linkage regions into areas	Weaver	08/23/76	09/20/76		Done
Add handling of *system links to the system	Weaver	09/27/76	10/17/76		Done
Convert prelinker to handle areas	Weaver	09/20/76	09/27/76		Done
Convert prelinker to handle *system links	Weaver	10/24/76	11/07/76		
Write and debug new commands to manage system areas	Weaver	10/21/76	11/07/76		
Install new commands to manage system areas	Weaver	11/17/76			
Document data base changes for stacks and linkage in MPM	Weaver	10/04/76	10/15/76		Done
Change subroutines (get_system_free_area_, etc.) for new linker	Weaver	09/13/76	09/27/76		Done
Change the binder to handle *system links	Weaver	10/15/76	10/24/76		Done
Generate new get_defptr_ and get_definition_	Morris Weaver	10/24/76	11/07/76		Done

GROUP Multics Language Group DATE PAGE 1 / 1

PROJECT BASIC AREA Changes for MR5.0

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Change BASIC to use new compiler_source_info include file	Weaver	11/01/76	11/07/76		Requires coordination of BASIC, FAST/DFAST, Fast FORTRAN
Change BASIC runtime to optimize certain string operations in ALM operators	Weaver				
Upgrade the BASIC manual to correct flaws and errors	Weaver				
Fix bugs in BASIC	Weaver				

GROUP Multics Language Group DATE PAGE 1 / 2

PROJECT APL AREA Changes for MR6.0

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Generate list of known APL bugs	Kennerly Green Phillips	07/01/76	10/24/76		
Generate a set of test programs, scripts for checking out APL	Kennerly Phillips	10/01/76			
Generate edited version of APL manual -- corrections only	Kennerly	07/01/76	10/31/76		
Rewrite APL manual, correcting examples, style, etc.	Kennerly	07/01/76	01/15/77		
Train terminal operator for APL manual editing/generation	Green Kennerly Archer	09/24/76	10/11/76		
Input changes to APL manual for 'accurate only' version	Archer	09/24/76	10/31/76		
Input changes to APL manual for final version	Archer	11/01/76	01/31/77		
Analyze APL performance problems	Green Kennerly	09/15/76			
Design enhancements to improve APL performance	Green	10/15/76	11/15/76		
Fix APL bugs	Green				
Implement improvements for APL performance	Green				
Design 'file system' mechanism for APL	Green Kennerly	12/01/76	01/01/77		
Implement 'file system' mechanism for [illegible]	Green	12/15/76	02/15/77		

GROUP Multics Language Group DATE PAGE 2 / 2

PROJECT APL AREA Changes for MR6.0

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Design)CONTINUE feature for APL	Green				Requires answering service changes
Implement)CONTINUE feature for APL	Green				Answering service mostly

GROUP_____Multics Language Group_____ DATE_____ PAGE__1__/__1__

PROJECT_____Tasking and Run Units_____ AREA_____Tasking and Run Units_____

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Write up proposed new run unit - as MTB	Webber	08/01/76	09/15/76	___	Draft being circulated
Schedule and hold design review for run units	Webber	___	___	___	
Implement run unit manager program		___	___	___	
Change makeknown_/makeunknown_ to handle reference count reporting	Bratt	___	___	___	
Change reference name manager to work on user-ring RNT	Bratt	08/01/76	08/15/76	___	Done
Integrate run units and tasking		01/01/77	02/01/77	___	
Write up proposed tasking facility	Asherman	10/04/76	10/18/76	___	
Experiment with possible tasking implementations and strategies	Asherman	09/01/76	11/20/76	___	
Generate user documentation of Multics tasking facilities	Asherman	12/01/76	01/01/77	___	

GROUP Multics Language Group DATE PAGE 1 / 2

PROJECT FAST FORTRAN AREA Changes for MR6.0

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Design first phase of Fast FORTRAN optimizer	Chang	05/01/76	09/01/76		Done
Write up Fast FORTRAN optimizer design	Chang	08/01/76	09/01/76		Done
Review design of Fast FORTRAN optimizer - phase 1	Many	09/01/76	09/07/76		Done
Implement phase 1 of Fast FORTRAN optimizer	Chang	10/11/76	12/31/76		
Check out phase 1 Fast FORTRAN optimizer	Chang	01/01/77	03/01/76		
Add the OPEN statement to FORTRAN	Levin	01/01/77	01/15/76		
Add the CLOSE statement to FORTRAN	Levin	01/01/77	01/15/77		
Change fortran_io_ to handle OPEN and CLOSE statements	Schoeman	01/01/77	01/21/77		
Change FORTRAN runtime I/O to optimize certain record operation in operators	Schoeman				
Handle implied do loops in FORTRAN efficiently	Levin	02/01/77			
Add support for $ format control in FORTRAN	Levin	01/01/77			
Support double precision complex arithmetic in FORTRAN	Levin Chang	02/01/77			
Provide the -profile and -long_profile options for FORTRAN	Levin Barnes	01/01/77			

GROUP Multics Language Group DATE PAGE 2 / 2

PROJECT FAST FORTRAN AREA Changes for MR6.0

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Optimize current code generator macros	Barnes				
Reformat the statement map	Barnes				
Add string handling features to FORTRAN	Levin Barnes				
Update the FORTRAN Manual for latest features	Boyd Levin	01/01/77	02/01/77		

GROUP Multics Language Group DATE PAGE 1 / 3

PROJECT PL/I AREA Tasks for MR6.0 and beyond

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Convert any_to_any_ to handle unsigned fixed binary data.					
Convert any_to_any_ to handle packed decimal data					
Add handling of octal (etc) constants to PL/I and runtime	Schoeman	09/15/76	09/30/76		Done
Change pl1 command to use get_temp_segments_	Schoeman				
Change pl1 compiler and runtime to use iox_ call forwarders	Schoeman				
Change the pl1 compiler to use standard, no_free areas	Schoeman				
Change the pl1 compiler to use template nodes created with cds	Schoeman				
Add the stacq, clock and vclock builtins to the Multics PL/I language					MSPL request
Add the stack_frame_ptr and stack_base_ptr builtins to the Multics PL/I language					MSPL request
Add the max_length builtin to the Multics PL/I language					
Add the rtrim and ltrim builtins to the Multics PL/I language					MSPL request
Add math builtins to complete our set to full ANSI	Schoeman				
Reimplement (for efficiency) the before, after and decat builtins					

GROUP Multics Language Group DATE PAGE 2 / 3

PROJECT PL/I AREA Tasks for MR6.0 and beyond

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Study, design, code and checkout the first phase of the PL/I loop optimizer	Barnes	07/01/76	04/01/77		
Add several listing enhancements to PL/I					MSPL request
Update AG94 to include a list of deviations from ANSI PL/I	Barnes				
Update AG94 for any language changes to be officially described	Barnes				
Optimize code for fixed binary long (double precision) arithmetic					
Optimize code generator to special case testing several bits in a single word	Barnes				
Add the unsigned attribute to the compiler					MSPL request
Add the abnormal attribute to the compiler					MSPL request
Add the compact attribute to the compiler					
Rewrite the lex phase using EIS	Green				
Change the compiler to generate *system links	Barnes				Done
Reformat the statement map	Barnes				
Optimize several cases of picture packir- and unpacking					

GROUP Multics Language Group DATE PAGE 3 / 3

PROJECT PL/I AREA Tasks for MR6.0 and beyond

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Add -long_profile options to pl1	Barnes				
Implement the STOP statement					
Add packed decimal data to the Multics PL/I language					

GROUP Multics Language Group DATE PAGE 1 / 1

PROJECT MISCELLANEOUS AREA

TASK DESCRIPTION	PERSONNEL	START	FINISH	M-W	CHANGES-STATUS
Design a high-level text oriented MACRO processor for Multics					
Implement the MACRO processor					
Add multisegment file capability to blocked vfiles.	Asherman				Done
Allow vfile_ 'file_status' control order to be called with switch closed	Asherman				Done
Design and code new vblocked file type for vfile_	Webber Asherman	10/24/76	11/20/76		
Implement a new, more general area manager for vfile_	Asherman	10/28/76	11/20/76		
Extend vfile_ to manage single-segment indexed files	Asherman	11/20/76	12/20/76		