A TRANSLATOR WRITING SYSTEM FOR MICROCOMPUTER HIGH-LEVEL LANGUAGES AND ASSEMBLERS

W. Robert Collins*
Computer Sciences Corporation
Hampton, Virginia

John C. Knight
Langley Research Center
Hampton, Virginia

and

Robert E. Noonan**
College of William and Mary
Williamsburg, Virginia

NASA LaRC uses many dedicated microprocessors in aerospace research. Few software tools are available for these machines, and in particular, very few have any form of high-level language facility. Since the Langley environment involves considerable experimentation, a great deal of software is experimental and may change frequently. It has to be prepared relatively quickly and at low cost.

In order to implement high-level languages whenever possible, a Translator Writing System of advanced design has been developed. It is intended for routine production use by many programmers working on different projects. As well as a fairly conventional parser generator, it includes a system for the rapid generation of table driven code generators. This code generation system is the result of research performed at the College of William and Mary under NASA sponsorship. The parser generator was developed from a prototype version written at the College of William and Mary.

The Translator Writing System includes various tools for the management of the source text of a compiler under construction. In addition, it supplies various "default" source code sections so that its output is always compilable and executable. The system thereby encourages iterative enhancement as a development methodology by ensuring an executable program from the earliest stages of a compiler development project.

This presentation will describe the Translator Writing System and some of its applications. These include the PASCAL/48 compiler, three assemblers, and two compilers for a subset of HAL/S. PASCAL/48 is a Pascal-like language for the Intel-8748 microcomputer. The assemblers which have been built are for assembly language subsets for the Intel-8080, the Motorola M68000, and the NSSC-II. The HAL/S subset was implemented for the Intel-8080 and the GE 703. Detailed measurements of the use of the system to build the code generators for the HAL/S compilers will be given.

*Work performed under NASA contract numbers NAS1-14900 and NAS1-16078.
**Work performed under NASA grant number NSG-1435.

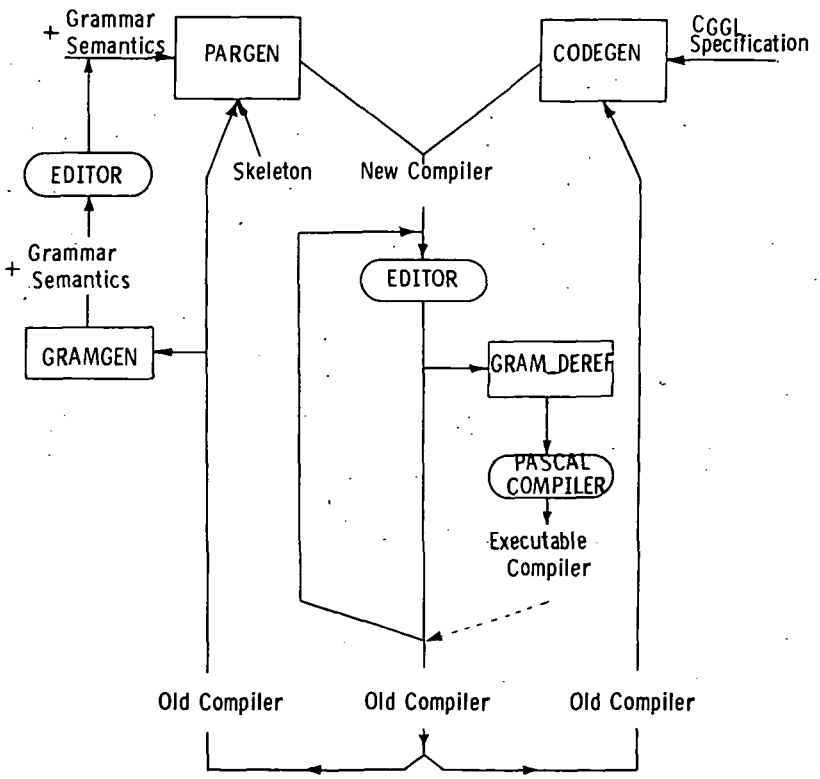
THE PROBLEM

- NEED HIGH-LEVEL LANGUAGES, HENCE COMPILERS
- NEED ASSEMBLERS
- ONE SOLUTION IS A TWS

TWS CRITERIA

- ENCOURAGE ITERATIVE ENHANCEMENT
  - EARLIEST POSSIBLE EXECUTION
  - TEXT MANAGEMENT TO RELIEVE TEDIUM
- FLEXIBILITY IN ITS USE
- TRANSPORTABLE IMPLEMENTATION

TRANSLATOR WRITING SYSTEM



USE OF TWS

1. IF PARSER NEEDED, RUN PARGEN, EXECUTE RESULTING COMPILER TO TEST.
2. CHANGE GRAMMAR AS NECESSARY, RERUN PARGEN.
3. ADD SEMANTICS USING EDITOR.
4. RECOVER GRAMMAR AND SEMANTICS WITH GRAMGEN IF NECESSARY TO RERUN PARGEN.
5. IF CODE GENERATION NEEDED, PREPARE CGGL SPECIFICATION AND RUN CODGEN.
6. MODIFY CGGL AS NEEDED.
7. ITERATE THROUGH ABOVE STEPS ADDING LANGUAGE FEATURES AS DESIRED.

PARGEN

- INPUTS
  - GRAMMAR IN STANDARD BNF
  - SEMANTICS IN PASCAL
  - SKELETON OR OLD COMPILER
- OUTPUT IS AN EXECUTABLE COMPILER INCLUDING
  - SCANNER
  - LALR (1) PARSER
  - SEMANTICS ROUTINE
- TEXT MANAGER PRESERVES PROGRAMMER'S CONTRIBUTION TO COMPILER E.G., SYMBOL TABLE ROUTINES

CODGEN

- INPUTS
  - CGGL SPECIFICATION
  - SKELETON OR OLD COMPILER
- OUTPUT IS AN EXECUTABLE COMPILER INCLUDING A CODE GENERATOR.
- CGGL IS A NON-PROCEDURAL LANGUAGE FOR DESCRIBING THE CODE-GENERATION PROCESS.
- TEXT MANAGER PRESERVES PROGRAMMER'S CONTRIBUTION TO COMPILER E.G., MACHINE LANGUAGE FORMATTER.

PASCAL/48

- INTEL-8748
  - MICROCOMPUTER
  - 8-BIT CPU
  - 64 WORD RAM
  - 1024 WORD ROM
  - 27 I/O LINES
- PASCAL/48
  - PASCAL DERIVATIVE FOR 8748
  - EXTENSIONS TO ALLOW CONTROL OVER GENERATED CODE
  - RESTRICTIONS TO PROHIBIT INEFFICIENT FEATURES
  - COMPILER AVAILABLE ON CDC CYBERS

ASSEMBLERS

- CUSTOMIZED SKELETON FOR ASSEMBLERS
  - TWO PASSES
  - STANDARD LISTING BY DEFAULT
  - FLEXIBLE INPUT FORMAT CONVENTIONS
  - HANDLES MACROS WITHOUT PARAMETERS
- COMPARED TO META-ASSEMBLER, ASSEMBLER BUILT FOR NSSC-II
  - WAS PRODUCED MORE QUICKLY
  - EXECUTES 5 TIMES FASTER
  - USES ONE FOURTH THE SPACE

EXAMPLE PASCAL/48 PROGRAM

```
NASA, LANGLEY RESEARCH CENTER      80/08/26.      08.51.12.              PAGE  1
PASCAL 8748C    VERSION 1.0.0    PROGRAM   MAIN                        CSC/NASA

  1 PROGRAM FOR_YOU;
  2
  3    VAR      I[2] : INTEGER;
  4             A[16_300, ROM] : ARRAY [100] OF INTEGER;
  5
  6    VALUE    A = (99 OF 0, 1);
  7
  8
  9    PROCEDURE GET_INPUT;
 10    BEGIN
 11       REPEAT
 12       UNTIL PORT1 BIT 3
 13    END;  (* GET_INPUT *)
 14
 15
 16 BEGIN  (* PROGRAM FOR_YOU *)
 17    FOR I := 100 DOWNTO 1 DO
 18       BEGIN
 19       GET_INPUT;
 20       PORT1 := PORT1 AND 2_11100011;
 21       PORT2 := A[I] + PORT1 XOR I
 22       END  (* FOR I := 100 DOWNTO 1 DO BEGIN *)
 23 END.  (* PROGRAM FOR_YOU *)
```

GENERATED CODE FOR EXAMPLE PROGRAM

```
        JMP     L009
        NOP
L003:   JMP     L003
        NOP
        NOP
L007:   JMP     L007
L009:   CLR     A
        MOV     PSW,A
        JMP     L012
L00D:   IN      A,P1            ; LINE   9
        CPL     A               ; LINE  13
        JB3     L00D            ; LINE  13
        RET                     ; LINE  13
L012:   MOV     R2,#99          ; LINE  14
L014:   CALL    L00D            ; LINE  18
        ANL     P1,#227         ; LINE  21
        IN      A,P1            ; LINE  22
        MOV     R1,A            ; LINE  22
        MOV     A,P2            ; LINE  22
        MOVP3   A,@A            ; LINE  22
        ADD     A,R1            ; LINE  22
        XRL     A,R2            ; LINE  22
        OUTL    P2,A            ; LINE  22
        DJNZ    R2,L014         ; LINE  22
```

SEPARATE CODE GENERATION USING CGGL

LANGUAGE: HAL/S

INTERMEDIATE CODE LANGUAGE: HALMAT

- 178 OPERATORS TOTAL
- 30 OPERATORS IMPLEMENTED
- 25 GENERATE CODE
- BASICALLY AN INTEGER SUBSET WITH SIMPLE CONTROL STRUCTURES

CODE GENERATORS

- ONE PASS
- NO PRE-OPTIMIZATION PASS
- NO PEEPHOLE OPTIMIZATION
- INTEL 8080, GE 703

Implementations

Intel 8080

- 8 bit machine
- single accumulator
- no index register
- 1, 2, 3 byte instructions
- hardware stack
- only integer add, subtract
- 16 bit addresses

GE 703

- 16 bit machine
- single accumulation
- index register
- one word instructions
- no hardware stack
- integer add, subtract, multiply, divide
- only address current page, page zero
- page: 256 words

703 Code Generator

Task	Time (days)
Reading manual	.5
CGGL program	1.5
Writing Pascal routines	1.5
Debugging	1.0
	4.5 days

Notes:

1. All programs were coded and keyed by Noonan.
2. Some of debugging time was used in cleanup.
3. One debugging run was used to fix a bug introduced by cleanup.
4. A total of 6 runs (execution) were used.
5. One CGGL bug.

703 Implementation

Source of Code	No. Procedures	% Lines	% Instr. Storage
8080 Imple.	46	58%	58%
Modified 8080	4	8%	6%
Noonan	9	10%	10%
CGGL	1	24%	26%

Notes:

1. CGGL program: 292 lines
2. Pascal program: 890 lines
3. For an earlier non-table-driven implementation, CGGL accounted for 83% of lines and 77% of storage.