The Design of A Pascal Compiler

Mohamed Sharaf, Devaun McFarland, Aspen Olmsted

Part I

Mohamed Sharaf

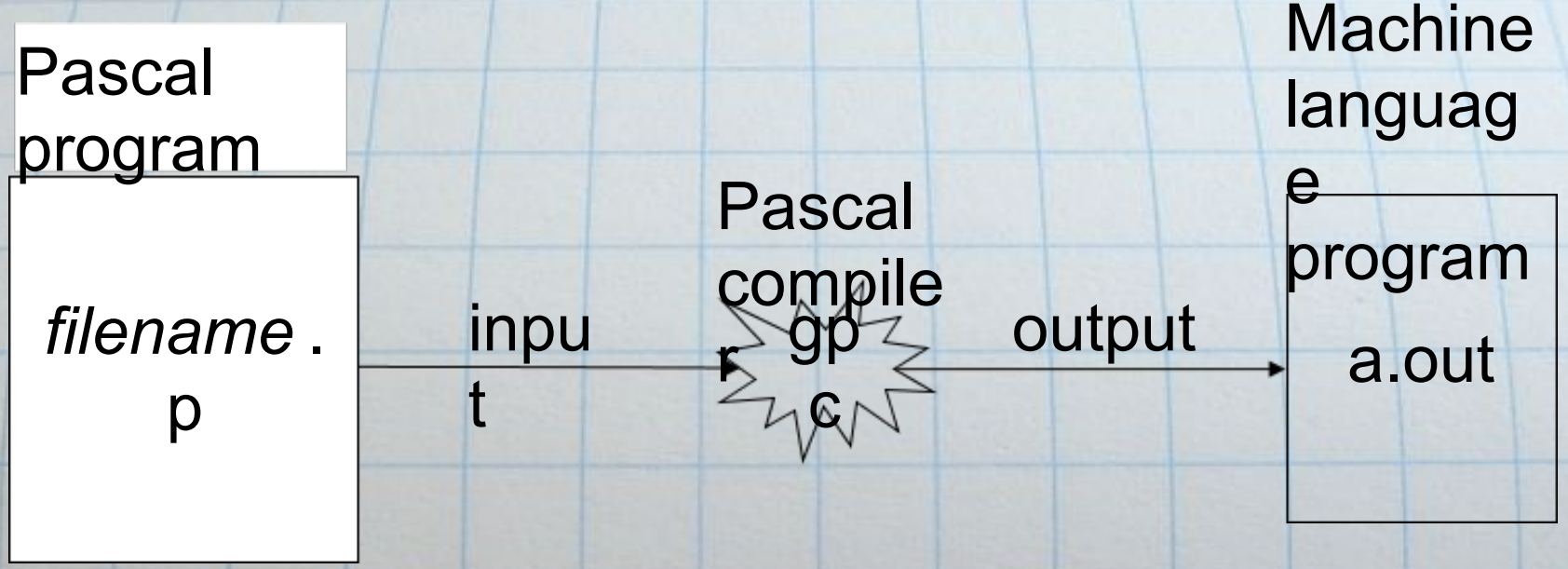
Introduction

- The Compiler is for the programming language PASCAL.
- The design decisions Concern the layout of program and data, syntax analyzer.
- The compiler is written in its own language.
- The compiler is intended for the CDC 6000 computer family.
  - CDC 6000 is a family of mainframe computer manufactured by Control Data Corporation in the 1960s.
  - It consisted of CDC 6400, CDC 6500, CDC 6600 and CDC 6700 computers, which all were extremely rapid and efficient for their time.
  - It had a distributed architecture and was a reduced instruction set (RISC) machine many years before such a

Pascal Language

- Imperative Computer Programming Language, developed in 1971 by Niklaus Wirth.
- The primary unit in Pascal is the procedure.
- Each procedure is represented by a data segment and the program/code segment. The two segments are disjoint.

Compiling Programs: Basic View



Representation of Data

- Compute all the addresses at compile time to optimize certain index calculation.
- Entire variables always are assigned at least one full PSU “Physical Storage Unit” i.e CDC6000 has ‘wordlength’ of 60 bits.
- Scalar types
- Array types

the first term is computed by the compiler

w=a+(i-l)*s

- Record types: reside only within one PSU if it is represented as packed. If it is not packed its size will be the size of the largest possible variant.

Data types …

- Powerset types
  - The set operations of PASCAL are realized by the conventional bit-parallel logical instructions ‘and ‘ for intersection, ‘or’ for union
- File types
  - The data transfer between the main store buffer and the secondary store is performed by a Peripheral Processor (PP).
  - The CPU actions caused by the standard procedures *put* and *get* by just change pointers.

    s ” buffer size”

    n ” n>2”

    s’ ”File component size”

    s=n*s’
  - The buffer should be able to hold at least one Physical Record Unit (PRU). “PRU : the unit that is used to represent file on secondary storage”
- Class types
  - Domain: the component of the class variable to which they are bound.
  - The allocated area of memory is calculated by the compiler.

Basic Structure Of Pascal Programs

Program name* .p *(Pascal source code)

Part I: Header

```
Program documentation
program name  (input, output);
```

Part II: Declarations

```
const
:
```

Part III: Statements

```
begin
:
end.
```

Header

- Program documentation
  - Comments for the reader of the program (and not the computer)
    - (* Marks the Start of the documentation
    - *) Marks the End of the documentation
- Program heading
  - Keyword: program, Name of program, if input and/or output operations performed by the program.

Example Header

```
(*
* Tax-It v1.0: This program will
* electronically calculate your tax
* return.

* This program will only allow you to
* complete a Canadian tax return
*)
```

Documentation

```
program taxIt (input, output);
```

Heading

Declarations

- List of constants
- List of variables

Reserved Words

- Have a predefined meaning in Pascal that cannot be changed

- and
- array
- begin
- case
- const
- div
- do
- downto
- else
- end
- file
- for
- foward
- function
- goto
- if
- in
- label

For more information on reserved words go to the url: http://www.gnu-pascal.de/gpc/index.html

Reserved Words

- Have a predefined meaning in Pascal that cannot be changed

- and
- array
- begin
- case
- const
- div
- do
- downto
- else
- end
- file
- for
- foward
- function
- goto
- if
- in
- label
- mod
- nil
- not
- of
- or
- packed
- procedure
- program
- record
- repeat
- set
- then
- to
- type
- until
- var
- while

For more information on reserved words go to the url: http://www.gnu-pascal.de/gpc/index.html

Standard Identifiers

- Have a predefined meaning in Pascal that ***SHOULD NOT*** be changed
- Predefined constants
  - false
  - true
  - maxint
- Predefined types
  - boolean
  - char
  - integer
  - real
  - text
- Predefined files
  - input
  - output

For more information on standard identifiers go to the url: http://www.gnu-pascal.de/gpc/index.html

Predefined Functions

- abs
- arctan
- chr
- cos
- eof
- eoln
- exp
- ln
- odd
- ord
- pred
- round
- sin
- sqr
- sqrt
- succ
- trunc

Know the ones in Table 3.1 of your book.

Predefined Procedures

- dispose
- get
- new
- pack
- page
- put
- read
- readln
- reset
- rewrite
- unpack
- write
- writeln

Declaring Variables

Declare variables between the 'begin' and 'end.'

Part I: Header

```
Program documentation

program name  (input,
output);
```

Part II: Declarations

```
const

:
```

Part III: Statements

```
begin

:

end.
```

Declare variables just after the 'begin'

Procedure Parameters

- Parameters denoting a constant " no assignment is allowed"
- Parameters denoting a variable.
- Parameters denoting procedure:
  - To represent procedure uniquely:
    - The address of the entry point of the code.
    - The address of the data segment of that procedure declared local variables.

Code Optimization

- Taking array index into consideration. This done mutually by HW or by Compiler.
- The 2nd important optimization is arithmetic optimization
  - x div c if c is 2,4,8… Just shift right 1,2,3.. times.
  - x*c and c is 2,4,8… Just shift left 1,2,3.. times.

Syntax Analysis

- Conway “Separable transition diagram”:
  - The syntax of the language is presented as a finite set of *pseudo-finite-state* recognizers. This is because the basic symbols to be recognized are replaced by sentences are replaced by the member of this set. Using TD Parsing.
  - The syntax of the language is formulated as a set S of finite graphs.
  - It is straightforward to translate to and from the diagrams to BNF and it is easy to verify unambiguity.
  - To strictly adhere to the constraint of a one-symbol lookahead .

Part II

Devaun McFarland

Performance and statistical data

- At a Glance
  - The Source Program
    - 4000
    - 130,000
    - 33
  - Contents
    - Distinct identifiers
    - Word-delimiters
      - End, begin, if, then, and else

The object program:

- Field length requires 19,000 words
  - Compiler Program proper – 67.8%
  - Object code Buffering – 4.7%
  - Object Table – 9.2%
  - Other Data – 4.5%
  - Input and Output Buffering – 8.3%
  - Interface and I/O routines - 5.5%

Program Instruction Set

- Program consists of 32,700 instructions as follows:
  - Long instructions(30-bit) = 48.7%
  - Short instructions(15-bit)=28.7%
  - Padding Instructions(NOOP)=22.8%

  - Long/Short instruction breakdown
    - Fetch/store, load literal, arithmetic, logical/shift, base address register, and jumps/subroutine calls.

- On registers
- X-registers – used as a stack, holds results while evaluating expressions
  - X1, X2, X3,X4, and X5 percentages.
- B-registers – are used for the display D
  - B1, B2,B3, and B4 percentages.

- Performance on recompilation
  - Time to load and compile (the source program)

    40 sec(CP)+15 sec(PP)
  - Yielding an average of
    - 100 lines of source code processed per (CP) second.
    - 820 instructions generated per second.

Compiler Design Technique

- 1968 – Earlier version of PASCAL
  - Compiler written in FORTRAN - the motive here is a result of wanting a compiler that could be available automatically for multiple computers.
- 1969 – Written in PASCAL
  - Here the compiler was translated ‘by hand’ and did not attempt to optimize. Several features were omitted.

- Task division
  - Type definitions, variable declarations and procedure headings including formal parameter list.
  - Expressions and Statements.
  - Interface with the operating system.

Part III

Aspen Olmsted

Relationship Between The Complexity of Compilation and Computer Architecture

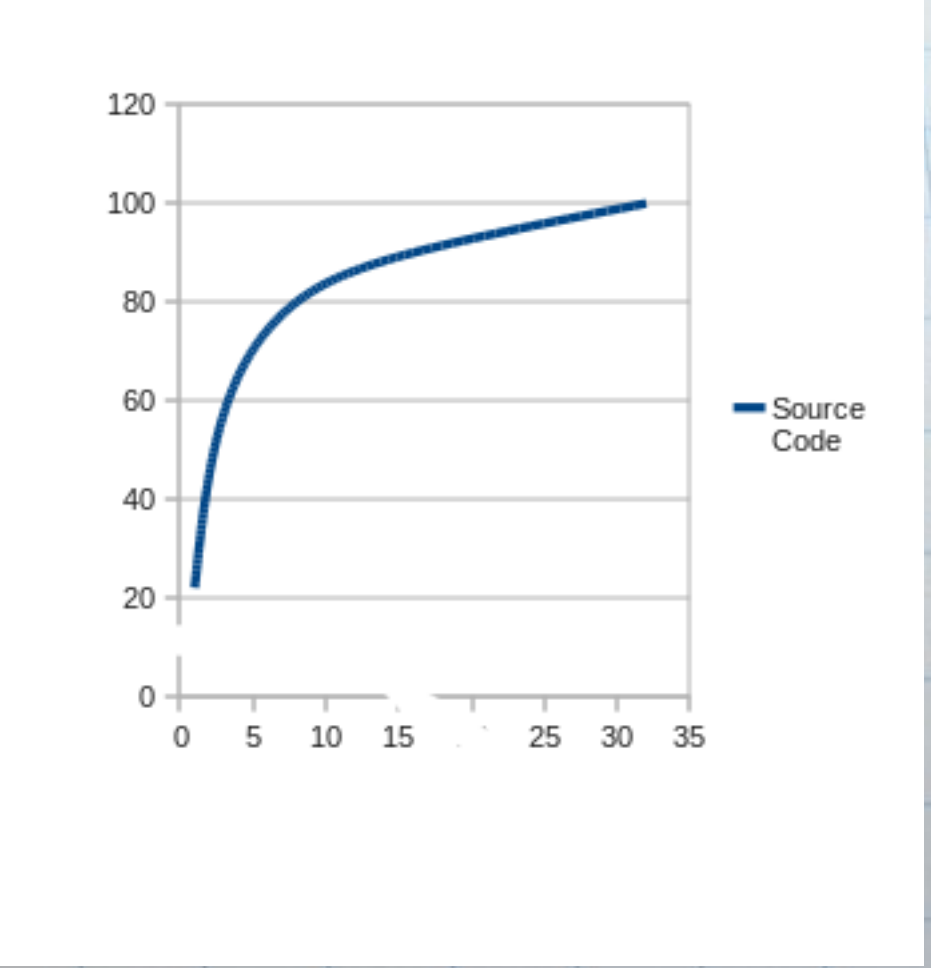
Desirable Computer Architecture Properties

- Pascal is a language designed without any specific computer in mind
- At Least Two Registers
- Simplicity of Instruction Set
- Make optimizations unnecessary

CDC 6000 Architecture

- Regularity and brevity of instruction set
- 64 Total Insturctions
- 42 used in compiler (66 percent)

Graph of Instructions By %Source Code



Conclusions

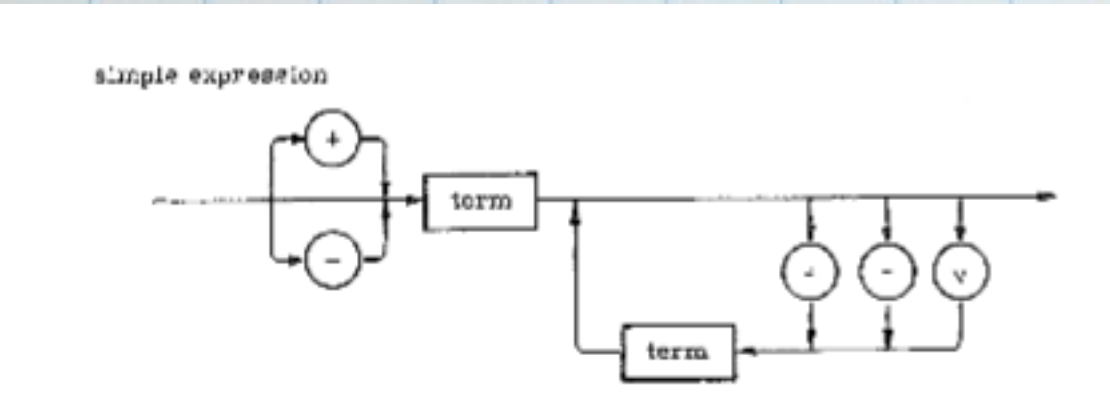
Program Comparison

- Compared Algol, Fortran & Pascal on 4 programs:
  - Matrix multiplcation B: A*A, no output
  - Sorting an array of 2,000 numbers
  - Finding all possible additive partitions of integers 1-30
  - Counting the characters in a file
- The performance differences between languages was negligible
- The reliability of the code generated by Pascal was higher

Successes

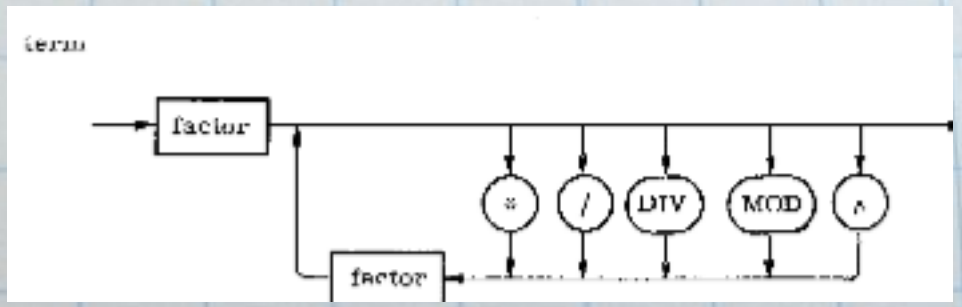
- High Reliability
- Scheme of syntax analysis allows separate features to be tested separately
- Recursive Descent for syntax analysis - requires implementation language supporting recursion
- Syntax designed in flow diagrams instead of BNF (giving readability)

Syntax Diagram - Simple Expression



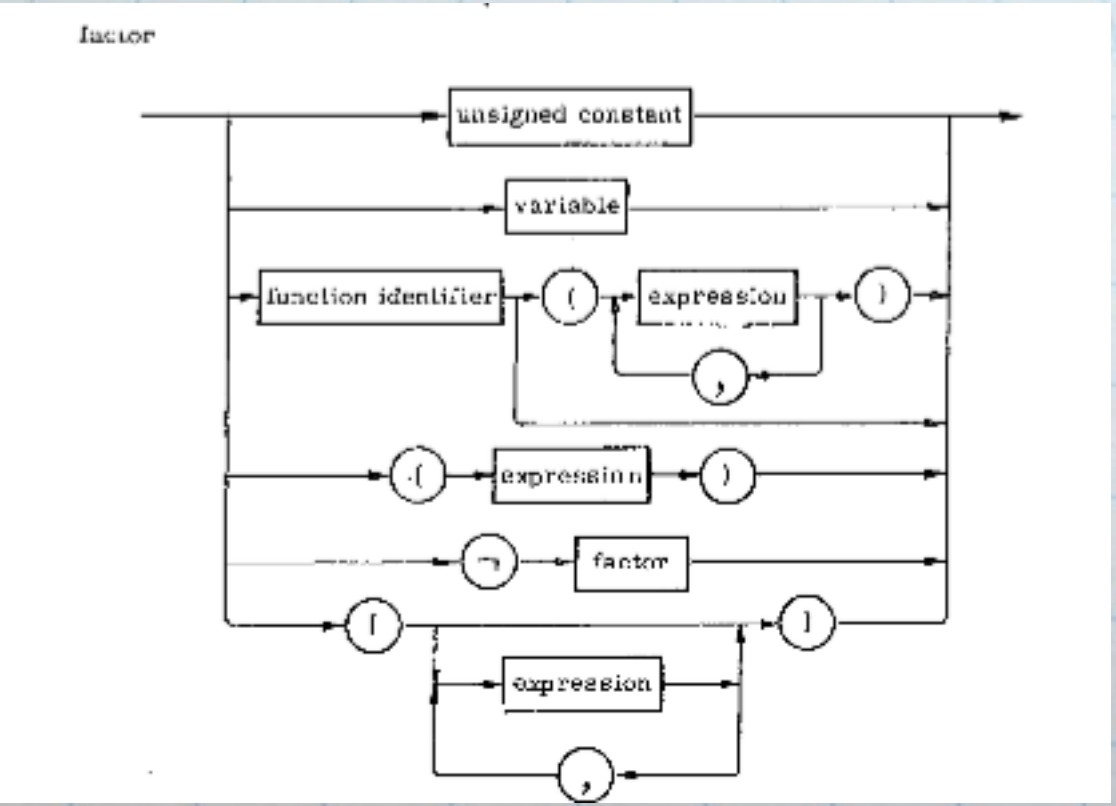
<Simple Expression> :== <Term> | <Simple Expression> <adding operator><Term> | <adding operator><Term>

Syntax Diagram - Term



<Term> :== <Factor> | <Term> <multiplying operator><Factor>

Syntax Diagram - Factor



<Factor> :== <variable> | <unsigned constant> | <function designator> | <set> | (<expression>) | !<factor>