Pascal P4 System

model for most other Pascal systems (UCSD)
compiler generates assembly language P4 intermediate code
assembler/interpreter assembles and executes P4 code

advantages
  • readable intermediate code
  • resolving of forward references in single pass in interpreter
  • portable system

problems
  • very slow

possible improvements
  • compiler generates binary P4 code
  • direct threaded code interpreter or JIT compiler
The P4 Virtual Machine

- **Program Memory**
- **Data Memory**
- **Constants**
- **Heap**
- **Stack**

- **Registers**
  - PC: Program Counter
  - SP: Stack Pointer
  - MP: Mark Stack Pointer
  - EP: Extreme Stack Pointer
  - NP: New Pointer

- **Overm**
Stack Frame

- local stack
- locals
- parameters
- return address
- old EP
- dynamic link
- static link
- function value

EP SP MP

- mst mark stack
- cup call user procedure
- ent enter block
- ret return
2 instructions with 2 operands are stored in one machine word

<table>
<thead>
<tr>
<th>op1</th>
<th>p1</th>
<th>q1</th>
<th>op2</th>
<th>p2</th>
<th>q2</th>
<th>codemax</th>
</tr>
</thead>
</table>

- MCP: strings
- BCP: boundary pairs
- SCP: set constants
- RCP: real constants
- ICP: integer constants

stack/heap

store

OVERM
OVERB
OVERS
OVERR
OVERI
maxstack
Assembler

instruction names stored in linear table
multiple type instructions are translated into different instructions
identical constants are stored only once
Interpreter

maximum of 4 files

subroutines for

• post mortem dump
• computation of base
• string comparison
• standard input/output procedures

instruction fetch

case statement
Pascal P4 Compiler

single pass compiler

control part: syntactic analysis calls lexical analysis (insymbol), semantic check and code generation

generated code: assembly language source

about 4000 lines of Pascal code

portable through constant definitions
Lexical Analysis

program driven lexical analysis (main routine: insymbol)

determines identifiers, keywords, numbers and other symbols
skips comments (option recognition)
output of source and error messages
spaces in identifiers are added economically
storeate of constants
integer computation can cause overflow
Tables of Lexical Analysis

1. frw
2. if
3. do
4. of
5. to
6. in
7. or
8. end
9. for
10. forward
11. program
12. function
13. procedure
14. ifsy
15. dosy
16. ofsy
17. tosy
18. relop
19. addop
20. noop
21. noop
22. noop
23. inop
24. orop
Syntax Analysis

program driven: recursive descent parser

procedure whilestatement;
begin
  expression (fsys+[dosy]);
  if sy = dosy
  then insymbol
  else error (54);
  statement(fsys)
end;

skip skips symbol until continuation is possible
Semantic Analysis

enterid, searchid, searchsection, getbounds, equalbounds, comptypes

no endless recursion for cyclic date structures (pointers)
Code Generation

gen0, gen1, gen2, gen0t, gen1t, gen2t

generate code for 0, 1 or 2 parameters with or without types

mes: computes maximum step depth

genfjp, genujpxjp, gencupent: branch switch and procedure call

alignquot, align: address computations

load, store, loadaddress: operand loads and stores

checkbnds: checks bounds

genlabel, putlabel: generation of labels