MICROCOMPUTERS
EN PASCAL

H.J.C. OTTEN

Voor microcomputers zijn een aantal goed bruikbare implementaties van Pascal beschikbaar. Om daarmee te kunnen werken is wel wat meer nodig dan voor het werken met bijvoorbeeld een Basic-interpreter. In het algemeen kan worden gesteld dat Pascal een microcomputer vereist met een RAM-geheugencapaciteit van 56K of meer en twee floppy disk-drives. Als in een advertentie wordt gesteld dat met minder ook kan worden gewerkt, dan moet de gebruiker wel erg veel geduld opbrengen.

Het implementeren van Pascal op een microprocessor van 8 bit is niet erg eenvoudig. Pascal is sterk stack-geörienteerd en vereist veel manipulaties met getallen van 16 bit. Ook moet zuinig met het geheugen worden omgegaan. Daarnaast is het vervelend dat er zoveel microprocessoren zijn die snel verouderen en worden opgevolgd door afwijkende microprocessoren.
De meeste implementaties van Pascal op microcomputers van 8 bit bestaan daarom uit een compiler-interpretercombinatie. De compiler genereert een code voor een processor die niet bestaat in hardware, maar erg vriendelijk is voor Pascal. Deze niet bestaande processor noemt men meestal Pascal-machine en de code noemt men p-code. Dit levert een relatief eenvoudige compiler, die niet van de microprocessor afhankelijk is. De p-code is vrij compact en geoptimaliseerd voor Pascal. De interpreter interpreteert de p-code-instructie en voert met een aantal machine-instructies de p-code-instructie uit. Dit is vergelijkbaar met de manier waarop Basic werkt. De voordelen van de compiler-interpretercombinatie zullen duidelijk zijn:
- Universele compiler.
- Weinig geheugen nodig.
- Programma's zijn gemakkelijk van de ene naar de andere computer over te dragen.

Er is ook een nadeel, wat de bovenstaande vergelijking met Basic al aankondigde: de interpreter vertraagt de programma-uitvoering.
Bekende compiler-interpretercombinaties zijn UCSD-Pascal, Pascal/M, Apple-Pascal, TCL Pascal en Atari-Pascal.
De echte compilers die rechtstreeks machine-code opleveren zijn superieur in snelheid, maar de programma's zijn in het algemeen iets groter. Door goede optimalisatie-technieken zijn bekende implementaties die op deze wijze werken, zoals JRT-Pascal, Pascal/MT+ en Pascal/Z goed bruikbaar.
Voor de nieuwe microprocessoren van 16 bit zoals de 68000 en de 8086 is het veel gemakkelijker een goede Pascal-compiler te schrijven. Een voorbeeld daarvan is Pascal/86MT+.
Alle in dit artikel genoemde Pascal-implementaties wijken in meer of mindere mate af van standaard-Pascal. Meestal zijn er vele nuttige uitbreidingen op het gebied van de in/uit- en de string-behandeling. De onderlinge verschillen zijn meestal aanzienlijk.
Van de genoemde Pascal-implementaties zijn UCSD-Pascal en de identieke Apple-Pascal het meest ingrijpend omdat er een volledig operating-systeem bijhoort inclusief een eigen filestructuur. Dat is niet altijd even handig, want het is altijd moeilijk files uit te wisselen tussen programma's op dezelfde microcomputer als er meerdere operating-systemen op draaien met afwijkende fysieke filestructuur.
De meeste Pascal-implementaties zijn onder CP/M geïmplementeerd. Dit zijn ook de meest professionele implementaties.
JRT-Pascal is een eigenzinnig buitenbeentje met goede prestaties, maar op essentiële punten duidelijk afwijkend van standaard-Pascal. Het is overigens de goedkoopste Pascal-compiler die ik ken ($ 29,95).

■ Boeken en Pascal

Een aantal boeken zijn in de loop van de jaren op mijn boekenplank terechtgekomen, die ik niet meer zou willen missen. Alle hierna genoemde boeken kan ik van harte aanbevelen.
Het gebruik van Pascal in het onderwijs heeft aanleiding gegeven tot een groot aantal goede en minder goede inleidingen in het gebruik van Pascal. Twee daarvan hebben op mij een uitstekende indruk gemaakt. Het boek „Programming in Pascal" van Peter Grogono is een uitstekende inleiding tot Pascal en alle daarna verschenen inleidingen tot Pascal of een kopie of bieden hetzelfde of een lager niveau. Het boek „An introduction to programming and problem solving with Pascal" van Schneider, Weingart en Perlman is meer dan een inleiding tot Pascal, het is tevens een uitstekend geschreven inleiding tot het programmeren in het algemeen. De tekst is hier en daar voorzien van „style clinics" en van deze „wijze" lessen heb ik erg veel geleerd.
Een probleem met inleidende boeken is dat ze weinig aandacht kunnen schenken aan moeilijke, maar wel belangrijke zaken. Een voorbeeld is het ontduiken van type-checking. In veel in Pascal geschreven Pascal-compilers is

dat een belangrijke stap. Aan complexe in/uit-zaken, zoals het op interactieve manier met een terminal werken, komen de inleidingen nauwelijks toe en zal het door de harde praktijk duidelijk moeten worden.
Naast het Pascal-rapport heeft Wirth nog een tweetal boeken geschreven over het programmeren in het algemeen en met Pascal als hulpmiddel. Het oudste boek „Systematic programming: an introduction" is bijna geheel opgenomen in het uiterst interessante tweede boek „Data structures + Algorithms = Programs". Wirth gaat hier in op gevorderde programmeertechnieken, met vele voorbeelden en algorithmen in Pascal.
„Software tools in Pascal" is een bewerking van een eerder boek van dezelfde schrijvers: „Software tools" (in Fortran). Kernighan en Pauger gaan er van uit dat programmeren op een gestructureerde manier het beste kan worden geleerd door complete programma's met een flinke omvang te bestuderen. Op een netjes gestructureerde wijze introduceren ze een aantal bruikbare programma's in Pascal geschreven.
De programma's zijn als werktuig voor programmeurs geschikt en zelfs grote programma's als een tekst-editor, een formatter en een macroprocessor zijn, naast vele handige filterprogramma's, tot in detail beschreven. Een uitstekend boek.
„Fundamentals of Interactive Computer Graphics" is een voorbeeld van een boek waar de algorithmen in de taal Pascal zijn weergegeven om zo duidelijk mogelijk aan de lezer de werking te vertellen. Het resultaat is goed leesbaar.
Een standaard boek in tot nu toe drie delen is „The Art of computer programming" van Donald Knuth. In dit boek wordt Pascal niet gebruikt, het bestond toen dit boek werd geschreven nog niet. Dit boek is tot op heden het standaardboek over datastructueren en fundamentele algorithmen. Pascal zou dit boek nog beter leesbaar hebben gemaakt.

■ Pascal en de toekomst

Het klinkt een beetje vreemd om op de toekomst en de opvolgers van Pascal in te gaan als Pascal net een vaste voet aan de grond begint te krijgen. Er zijn echter een aantal ontwikkelingen op het gebied van programmeertalen die interessant en ingrijpend zijn. De eerste ontwikkeling is die van de door Wirth ontworpen opvolger voor Pascal: Modula-2. Daarin zijn vrijwel alle genoemde problemen van Pascal opgelost.
Een tweede en waarschijnlijk nog belangrijkere ontwikkeling is de definitie van een nieuwe programmeertaal: Ada.
Deze taal is op initiatief van het Amerikaanse ministerie van defensie ontworpen met de bedoeling dit de standaard-programmeertaal te laten worden in alle regeringsopdrachten. Dat betekent dat veel programmatuur in Ada zal worden geschreven. Ada is een erg krachtige taal met alle kwaliteiten van een gestructureerde productie-taal. Er zijn echter zoveel mogelijkheden in de taal gestopt dat het geen eenvoudige en kleine taal is zoals Pascal. Vele specialisten twijfelen daarom aan de betrouwbaarheid van de onvermijdelijk erg complexe compilers. Een microcomputer-implementatie van Ada (er zijn al experimentele versies) zal noodzakelijkerwijze veel weglaten van de vele Ada-mogelijkheden. Voorlopig kunnen we nog het beste met een goede implementatie van Pascal werken.

■ Pascal implementatie op microcomputers

UCSD-Pascal

Diverse microprocessoren, eigen

operating-systeem, compiler-interpretercombinatie en afwijkingen en toevoegingen van de ISO-standaard.

Apple-Pascal
6502 en implementatie van UCSD-Pascal op de Apple-microcomputers.

Pascal/M
Z80/8080/8085, CP/M-operating-systeem, compiler-interpretercombinatie en afwijkingen en toevoegingen van de ISO-standaard.

Pascal/Z
Z80, CP/M-operating-systeem, compiler-assemblercombinatie en afwijkingen en toevoegingen voor de ISO-standaard.

Pascal/MT+
Z80/8080/8085, CP/M-operating systeem, compiler en toevoegingen van de ISO-standaard.

TCL Pascal
6502, Commodore DOS, compiler-interpretercombinatie en afwijkingen en toevoegingen ISO-standaard.

Atari Pascal
6502, Atari DOS, compiler-interpretercombinatie en afwijkingen en toevoegingen van de ISO-standaard.

JRT-Pascal
Z80/8080/8085, CP/M, compiler en afwijkingen en toevoegingen van de ISO-standaard.

■ Besproken boeken

„Pascal Manual and Report (2nd Edition)" van K. Jensen en N. Wirth, uitgegeven door Springer-Verlag, 1978.

„Systematic programming: An introduction" van N. Wirth, uitgegeven door Prentice Hall, 1973.

„Algorithms + Data structures = Programs" van N. Wirth, uitgegeven door Prentice Hall, 1976.

„Software tools in Pascal" van B. W. Kernighan en P. J. Plauger, uitgegeven door Addison-Wesley Publishing Company, 1981.

„An introduction to programming and problem solving with Pascal" van G. M. Schneider, S. W. Weingart en D. M. Perlman, uitgegeven door J. Wiley & Sons, 1978.

„Programming in Pascal" van P. Grogono, uitgegeven door Addison-Wesley Publishing Company 1980.

„Fundamentals of Interactive Computer Graphics" van J. D. Foley en A. van Dam, uitgegeven door Addison-Wesley Publishing Company, 1982.

„Structured programming" van O. J. Dahl, E. W. Dijkstra en C. A. E. Hoare, uitgegeven door Academic Press, 1972.

„A discipline of programming" van E. W. Dijkstra, uitgegeven door Prentice Hall, 1976.

„The Art of Computer programming" Volume 1: Fundamental Algorithms, Volume 2: Seminumerical Algorithms en Volume 3: Sorting and Searching van D. E. Knuth, uitgegeven door Addison-Welsey Publishing Company, 1973.

■ Artikelen met commentaar op standaard-Pascal

„Why Pascal is not my favorite programming language" van B. M. Kernighan, gepubliceerd in Computing Science Technical Report no. 100 en uitgegeven door Bell Laboratories July 18, 1981.

„Ambiguities and insecurities in Pascal" van J. Welsh, J. W. J. Sneeringer en C. A. R. Hoare, gepubliceerd in Software Practice and Experience 7 1977 pag. 685 t.e.m. 696.