INTERMEDIATE FORMS OF SOURCE PROGRAMS

- POLISH NOTATION
- ABSTRACT SYNTAX TREE
- DIRECT THREADED CODE
- INDIRECT THREADED CODE
- P-CODE FOR PASCAL-MACHINE

POLISH NOTATION (Triple Notation)

⟨operator⟩, ⟨operand$_1$⟩, ⟨operand$_2$⟩

The expression $W * X + (Y + Z)$ is represented in triple notation as follows:

1. *, W, X
2. +, Y, Z
3. +, (1), (2)

Each triple is numbered, and the result of a previous triple is specified by its number in parentheses. Thus the (1) and (2) which appear in triple 3 specify that the results of triple 1 and triple 2 are to be used as operands.

The conditional statement

If $X > Y$
then $Z \leftarrow X$
else $Z \leftarrow Y + 1$

can be represented in triple notation as follows:

1. −, X, Y
2. BMZ, (1), 5
3. ←, Z, X
4. BR, , (7)
5. +, Y, 1
6. ←, Z, (5)
7. ⋮

POLISH NOTATION
(Indirect Triple Notation)

$A \leftarrow B + C * D/E$
$F \leftarrow C * D$

can be represented by indirect triples as follows:

Operations	Triples
1. (1)	(1) *, C, D
2. (2)	(2) /, (1), E
3. (3)	(3) +, B, (2)
4. (4)	(4) ←, A, (3)
5. (1)	(5) ←, F, (1)
6. (5)	

POLISH NOTATION
(Quadruple Notation)

Quadruple notation is another type of n-tuple notation in which each instruction has four fields of the form:

⟨operator⟩, ⟨operand$_1$⟩, ⟨operand$_2$⟩, ⟨result⟩

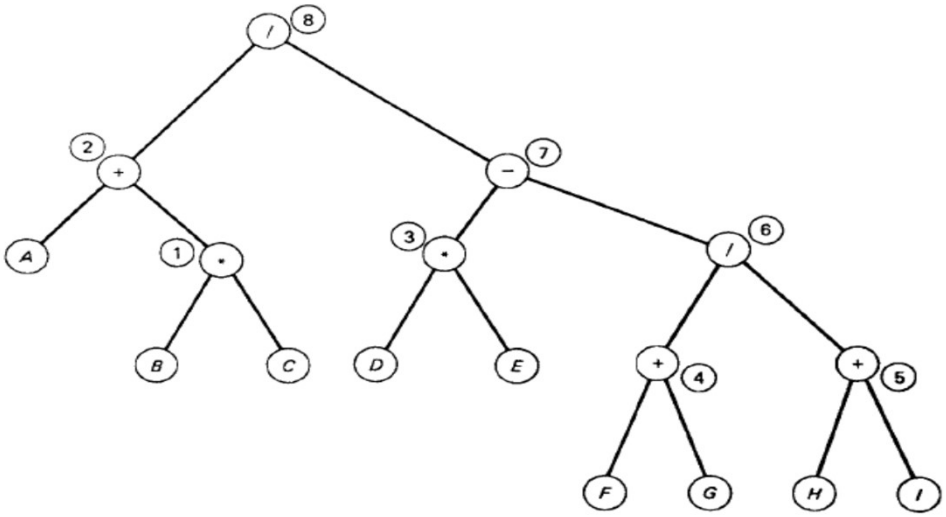
where ⟨operand$_1$⟩ and ⟨operand$_2$⟩ denote the first and second operand, respectively, and ⟨result⟩ specifies the result of the operation. The result is usually a temporary variable. Such a variable may be assigned to either a register or main memory location later on by the compiler. For example, the expression $(A + B)*(C + D) - E$ can be represented by the following sequence of quadruples:

+ , A, B, T_1
+, C, D, T_2
*, T_1, T_2, T_3
−, T_3, E, T_4

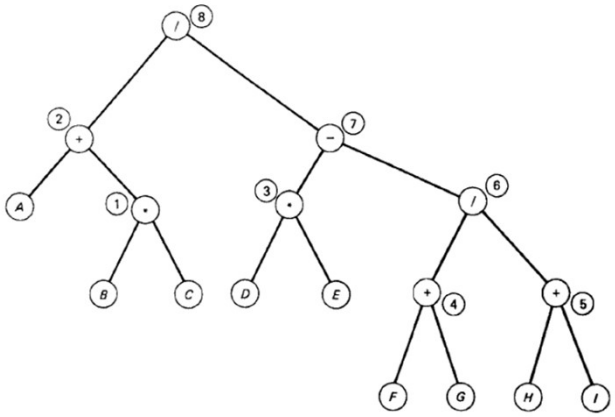
where T_1, T_2, T_3, and T_4 are temporary variables.

Quadruples facilitate the performance of several program optimizations. One drawback of using quadruples is that the allocation of temporary names must be managed. Such a management strategy will reuse some temporaries.

ABSTRACT SYNTAX TREE

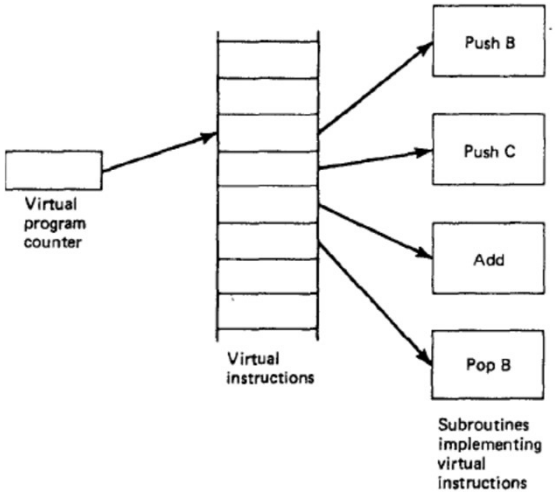


ABSTRACT SYNTAX TREE

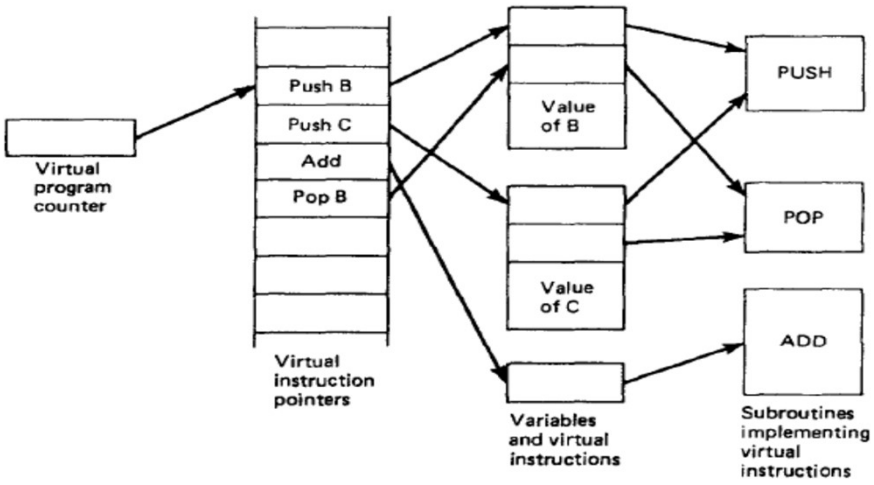


(1) *, *B*, *C*
(2) +, *A*, (1)
(3) *, *D*, *E*
(4) +, *F*, *G*
(5) +, *H*, *I*
(6) /, (4), (5)
(7) −, (3), (6)
(8) /, (2), (7)

DIRECT THREADED CODE (PDP-11)



INDIRECT THREADED CODE (PDP-11)



P-CODE FOR P-MACHINE

A program is said to be *portable* if it can be moved to another machine with relative ease. That is, the effort to move the program is substantially less than the initial effort to write it. On the other hand, a program is *adaptable* if it can be readily customized to meet several user and system requirements.

Suppose that we want to move a given compiler from, say, machine X to a different machine Y. To realize such a move, we must rewrite the code-generation routines of the existing compiler to produce code for machine Y. The rewriting task is much easier to accomplish if the original compiler has been divided into two parts with a well-defined interface. The first part (or front end) deals with the source language and the second (or back end) with the target machine. For a well-defined interface only the target-machine part need be changed. One form of interface is a symbolic program in an intermediate assembly system.

The flow of information between the two parts of a compiler takes the form of language constructs in one direction (front end to back end), and target-machine information in the other direction (back end to front end). The interface can be realized by using an abstract machine. The source-language constructs can be mapped into pseudo operations on this abstract machine. The abstract machine is designed for the particular source language, for example, a PASCAL machine.

P-CODE FOR P-MACHINE

The *Pascal-P* compiler (Nori et al., 1981) is a portable compiler for essentially "standard PASCAL." The compiler produces object code (P-code) for an abstract machine.

The abstract machine called a *P-machine* is a simple machine-independent stack computer. The Pascal-P language is easily transported, at low cost, to a variety of real machines.

The hypothetical stack machine has five registers and a memory. The registers are

1. **PC**—the program counter
2. **NP**—the *new* pointer
3. **SP**—the *stack* pointer
4. **MP**—the *mark* pointer
5. **EP**—the *extremestack* pointer

The last four pointers are associated with the management of storage in memory.

P-CODE FOR P-MACHINE

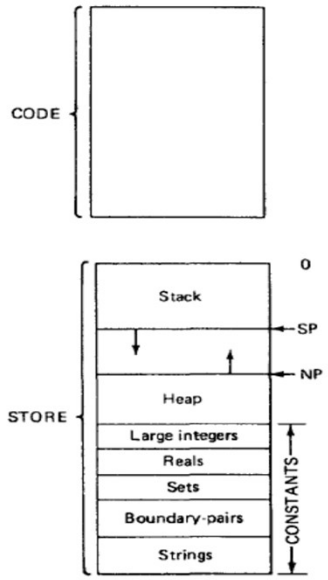
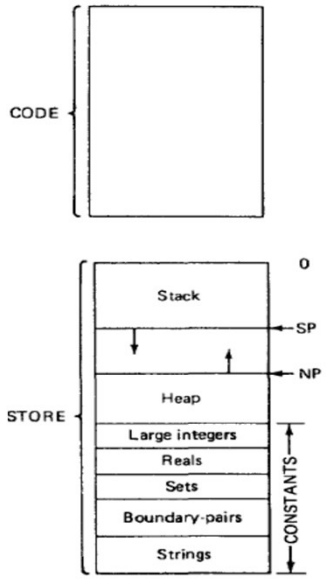


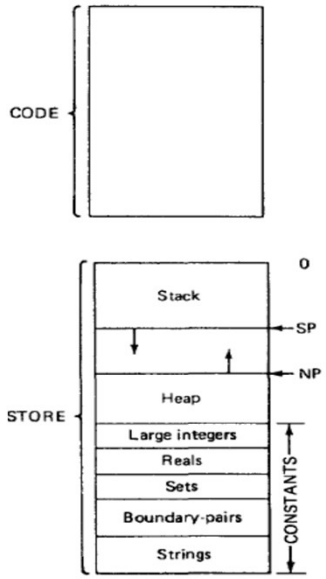
Figure 10-4 Memory layout of stack computer

P-CODE FOR P-MACHINE



The memory, which can be viewed as a linear array of words, is divided into two main parts. The first part of memory, CODE, contains the machine instructions. The second part of memory, STORE, contains the data (i.e., noninstructions) part of a program. The layout of the computer's memory is illustrated in Fig. 10-4. Observe that PC refers to the location of an instruction in CODE. On the other hand, the pointers EP, MP, NP, and SP refer to positions in STORE.

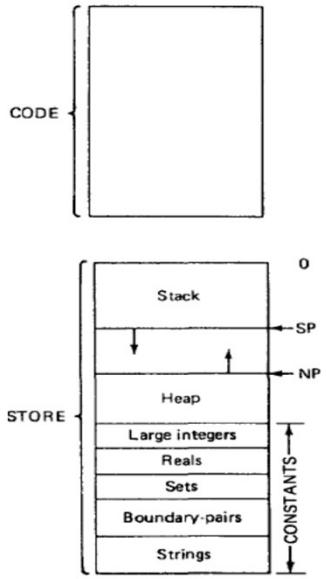
P-CODE FOR P-MACHINE



STORE contains two parts; the first part represents the various constants of a given program while the second part is dedicated to the other data requirements of the executing program.

The stack, whose top element is denoted by SP, contains all directly addressable data according to the data declarations in the program. The heap, with associated top element NP, consists of the data that have been created as a result of direct programmer control (i.e., **new** statements). The heap is similar to a second stack structure.

P-CODE FOR P-MACHINE



The stack consists of a sequence of *data segments*. Each data segment is associated with the activation of a procedure or a function.

Each data segment contains the following sequence of items:

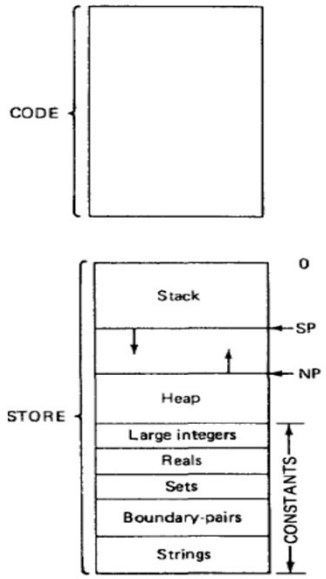
1. A *mark stack* part
2. A (possibly empty) parameter section for the routine associated with the data segment
3. A *local-data* section for any local variables
4. Temporary elements that are required by the processing statements

The mark stack part has the following fields:

1. Value field for a function (present but not used in a procedure)
2. Static link
3. Dynamic link
4. Maximum stack size value
5. Return address

Observe that the static and dynamic links refer to STORE and the return address denotes a position in CODE.

P-CODE FOR P-MACHINE



The basic modes of PASCAL (e.g., INTEGER, REAL) are supported on the stack computer. The machine has several classes of instructions, for example, arithmetic (for integers and real), logical, and relational. Each instruction has an opcode and possibly two parameters. The second parameter denotes an address. Many of the instructions in the instruction repertoire are closely related to the PASCAL language. For example, there is an instruction for set, set intersection, set union, and set membership to support the operations on sets in PASCAL.

The Pascal-P compiler has been transported to several real machines. In fact, an implementation kit has been prepared to assist systems people to move the compiler.