PDP PROGRAMMING NOTE

FRAP AND DECAL

DEC PROGRAMMED DATA PROCESSOR

DIGITAL EQUIPMENT CORPORATION • MAYNARD • MASSACHUSETTS

INTRODUCTION

The Programmed Data Processor is a high performance, low cost digital computer. A powerful instruction list (see the PDP-1 Manual), enables the programmer to take excellent advantage of the high speed and unusually powerful arithmetic and logical properties of the machine.

This computer and its instruction code are particularly well suited to special purpose problem solving, with basic machine simplicity and flexible logical operations.

Writing the program in binary machine language, or even in octal for conversion to binary, is usually a tedious task. In general, the programmer wants to express the program in problem-language using alphabetic words and easily understood numbers and symbols.

Several systems are available for PDP which enable the use of convenient symbolic language in preparing the program. The initial program preparation is done on off-line facilities and the described programming systems, which are loaded into PDP, accept the initial symbolic language and prepare a final program tape which can be read directly into PDP for performance of the program. These systems are:

FRAP - An assembly program which enables program preparation with PDP mnemonic instructions, octal numbers, symbolic addresses and other special words.

DECAL - An assembly and a compiler program with FRAP capability plus the capability for interpreting algebraic statements and translating one symbolic statement to several machine language instructions.

In both cases, the result of FRAP or DECAL preparation of a symbolic tape is a binary machine language tape which is in the READ-IN MODE format for direct operation by the computer.

FRAP

SYMBOLIC ASSEMBLY PROGRAM

The FRAP program operates on the symbolic tape, assigning numerical values in a one-to-one fashion to words which will contribute to output items. These values are combined into output items which are punched on paper tape during the second of two passes required for the translation and assembly process. The machine language tape prepared by FRAP can then be read directly into the computer.

The symbols used by the programmer to write the original program are the instruction code of PDP-1, octal numbers, and special words which do not result in an output item (labels and control words). The ability to define new symbols and to call in subroutines make this system powerful and expandable.

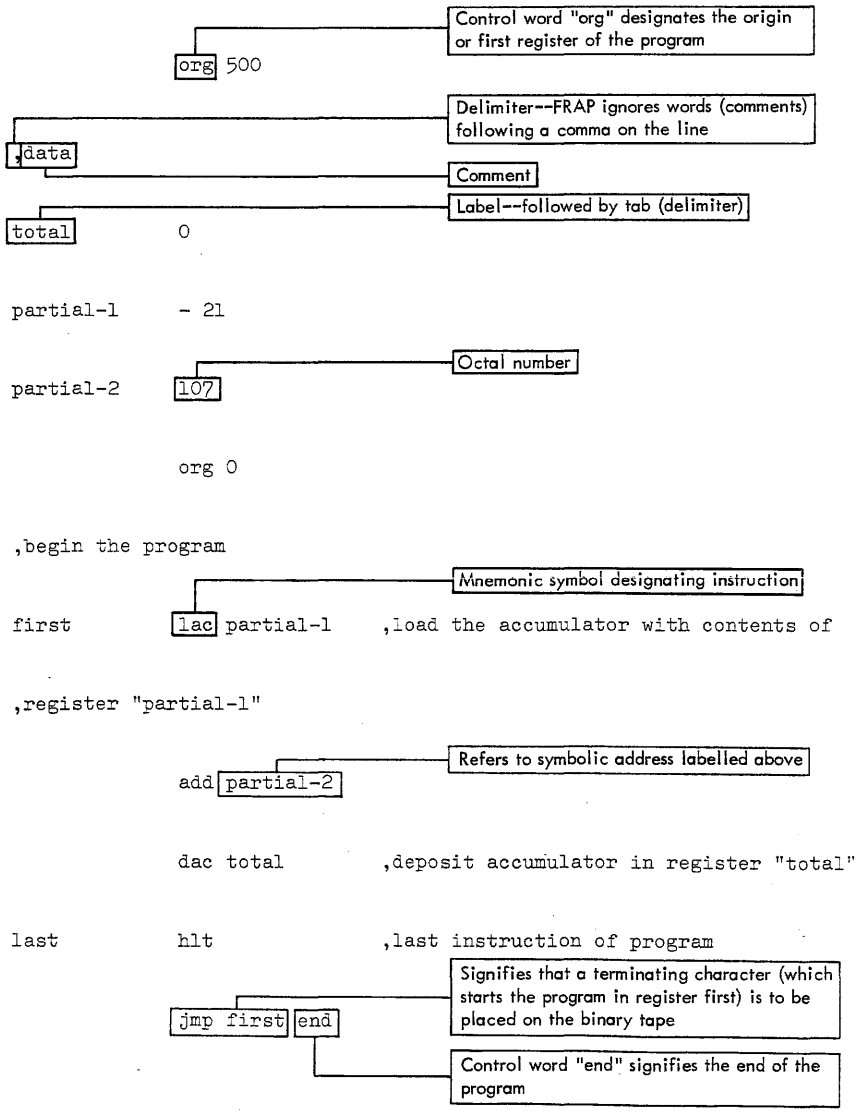
BASIC FRAP TERMINOLOGY

Symbolic Program	The initial program written in alphabetic and octal numeric symbols to be translated and assembled by FRAP.
Machine Language Tape	The output paper tape prepared by FRAP which can be read directly into PDP-1.
Character	Any key on the typewriter, including upper case, backspace, etc.
Word	Any ordered string of characters occurring between two delimiters. Various classes of words are instruction symbols, numbers, labels, and control words.
Statement	A line of the typewritten symbolic program which is compiled by FRAP into an output item (machine instructions).
Delimiter	A character used to segregate words and statements of the symbolic program. Word delimiters are tab, space and comma. A statement delimiter is carriage return.
Instruction Symbols	These are alphabetic PDP mnemonic instructions. The PDP instruction list is permanently stored in FRAP in numerical (octal) form.
Numbers	Octal constants or addresses where assigned values are the numbers themselves.
Labels	Register name used for symbolic addresses. The first undefined word in a statement.

Control words	FRAP defined special symbols used for controlling or modifying the assembly process.
Output Item	Three lines of tape in binary format representing one typewritten line of program (one data or instruction word).

SYMBOLIC PROGRAM FORMAT

The following example illustrates the typewritten format of the symbolic program. The program adds the contents of two registers which have been arbitrarily assigned symbolic address names of "partial 1" and "partial 2." The result is stored in a register named "total."



FLOATING POINT LANGUAGE FOR FRAP

Floating point arithmetic is handled by subroutines which are addressed by FRAP definitions. Floating point numbers are represented in the form: $a \times 2^b$. Here, a is in the range $1/2 \leq a < 1$, and b is an integer. Normally a would be stored in one register and b in another. The subroutines are called by name and may require one operand.

Each floating point function may be defined to make PDP-1 look exactly like a machine which has built-in floating point arithmetic. Registers are set aside for the subroutines and a pseudo accumulator (two memory registers) is used within the routines. As an example, "floating load the contents of register X" into the pseudo accumulator would be defined in FRAP as follows: opd fld jsp a47

The FRAP control word "opd" defines fld so that each time fld is written on the symbolic tape, jsp a47 is placed on the binary machine tape. The subroutine starting address is a47. Thus the floating point subroutine fld would be called as follows:

fld , load the floating accumulator with the contents of "a"

a , the location of register "a" must follow fld

FLOATING POINT SUBROUTINES

These routines handle floating point arithmetic and can be directly addressed by FRAP.

Symbol	Function	Description	Time Required
fld x	Load	C(X)[1] → C(pac)[2], C(X) are not changed	115 μsec
fst x	Store	C(pac) → C(X)	125 μsec
fad x	Add	C(pac) + C(X) → C(pac)	660 μsec
fsb x	Subtract	C(pac) - C(X) → C(pac)	740 μsec
fmp x	Multiply	C(pac) . C(X) → C(pac)	700 μsec
fdh x	Divide	C(pac) / C(X) → C(pac)	750 μsec
fsq[3]	Square Root	$\sqrt{C(pac)}$ → C(pac)	4.2 μsec
ffb n	Floating to binary conversion	C(pac) (floating binary) → C(por)[4] (binary). N is the binary point position. Binary signed integer and signed faction appear in por, por + 1. C(pac) are not changed.	--

Symbol	Function	Description	Time Required
fbf n	Binary to floating conversion	C(pac), with binary point at binary position N, is converted to a floating point number.	--
fsn	Sine	$\text{Sin.}\ \frac{\pi}{2}\ .\ [C(pac)] \rightarrow C(pac)$	5.5 μsec
fcs	Cosine	$\text{Cos.}\ \frac{\pi}{2}\ .\ [C(pac)] \rightarrow C(pac)$	6 μsec
fat	Arctangent	$\text{Tan}^{-1}\ [C(pac)] \rightarrow C(pac)$	11 msec
fex	Exponential	$e[C(pac)] \rightarrow C(pac)$	8.5 msec
fln	Natural Logarithm	$\ln[C(pac)] \rightarrow C(pac)$	2 msec
fbd	Floating Binary to Floating Decimal	$C(pac) \rightarrow [c \times 10^{d}] \rightarrow C(por)$	2.5 msec
fdb	Floating Decimal to Floating Binary	C(a c, i q) → floating decimal C(pac) → floating binary	3.0 msec

NOTES:
1. Pseudo register, actually two registers X, and X + 1
2. Pseudo accumulator, actually register pac, pac + 1
3. Requires no operand
4. Pseudo operand register, actually registers por, por + 1

FRAP PROGRAM FOR EXPRESSION EVALUATION

The following program evaluates an expression and stores the result in a register, vectormagnitude. The expression is:

$$\text{vectormagnitude} = (a \sin 2\pi wt)^2 + (b \cos 2\pi wt)^2$$

, constants and program parameters

```
four            org 0   200000      ,4 in floating point ie. .5 (the mantissa)
                        3           ,the exponent is 3, thus .5 x 2 cubed is 4
a                       -           ,a mantissa
                        -           ,a exponent
b                       -           ,b mantissa
                        -           ,b exponent
temp1                   0           ,
                        0           ,
temp2                   0           ,
                        0           ,
vectormagnitude         -           ,answer mantissa
                        -           ,answer exponent
omega                   -           ,omega mantissa
                        -           ,omega exponent
time                    -           ,time mantissa
                        -           ,time exponent
, begin program
begin                   fld         ,omega
                        omega
                        fmp         ,4 omega
```

```
four
fmp             ,4 omega t
time
fst
temp1
fcs             ,cos 2 pi omega t
fmp
b
fst
temp2
fmp
temp2
fst             ,(b cos 2 pi omega t) squared
temp2
fld
temp1
fsn             ,sin 2 pi omega t
fmp
a
fst
temp1
fmp
temp1
fad             ,sum of components squared
temp2
```

```
                    fsq

                    fst

                    vectormagnitude
finish              hlt

                    jmp begin end
```

DECAL

COMPILER, ASSEMBLER AND LINKING LOADER PROGRAM

DECAL (Digital Equipment Compiler, Assembler, and Linking loader for PDP) is an integrated programming system for PDP. It incorporates in one system all of the essential features of advanced assemblers, compilers, and loaders.

DECAL is both an assembler and compiler. It combines the one-to-one translation facilities of an assembler, and the one-to-many translation facilities of a formula translation compiler. Problem oriented language statements may be freely intermixed with symbolic machine language instructions. A flexible loader is available to allow the specification of program location at load time. The programmer may specify that certain variables and constants are "systems" variables and constants. The symbols so defined are universally used in a system of many routines. Thus, communications between parts of a major program are facilitated even though these parts may be compiled separately. Storage requirements for a large program are lessened by this technique.

DECAL is an open-ended programming system which can be modified without a detailed understanding of the internal operation. This is achieved by means of a recursive definition facility based on a skeleton compiler with a small set of logical capabilities. The skeleton compiler acts as a bootstrap for introducing more sophisticated facilities.

DECAL PROGRAM FOR EXPRESSION EVALUATION

The expression evaluated above for the computation of vectormagnitude may be written for DECAL compilation as follows:

```
vectormagnitude =
    sqrt (  (a*sin2*3.14159*omega*t)squared+(b*cos2*3.14159*omega*t)squared)
```

MISCELLANEOUS ROUTINES

One of the more important of the additional programs which are provided is the Typewriter Interrogator Program (TIP). TIP allows the typewriter to be used most effectively as an input-output link by which programs and data are examined and modified. The features include request for printing of a series of registers, interrogation and modification of the contents of registers, and the ability to request new tapes after programs have been suitably modified. Communication is done completely via the typewriter in either octal numbers, decimal numbers, or alphanumeric codes. Register contents are presented in similar form.

Other miscellaneous routines handle arithmetic processes, e.g. number conversions, and communication with the input or output devices. These routines include various format print outs, paper tape and magnetic tape read in programs, and oscilloscope character display subroutines.