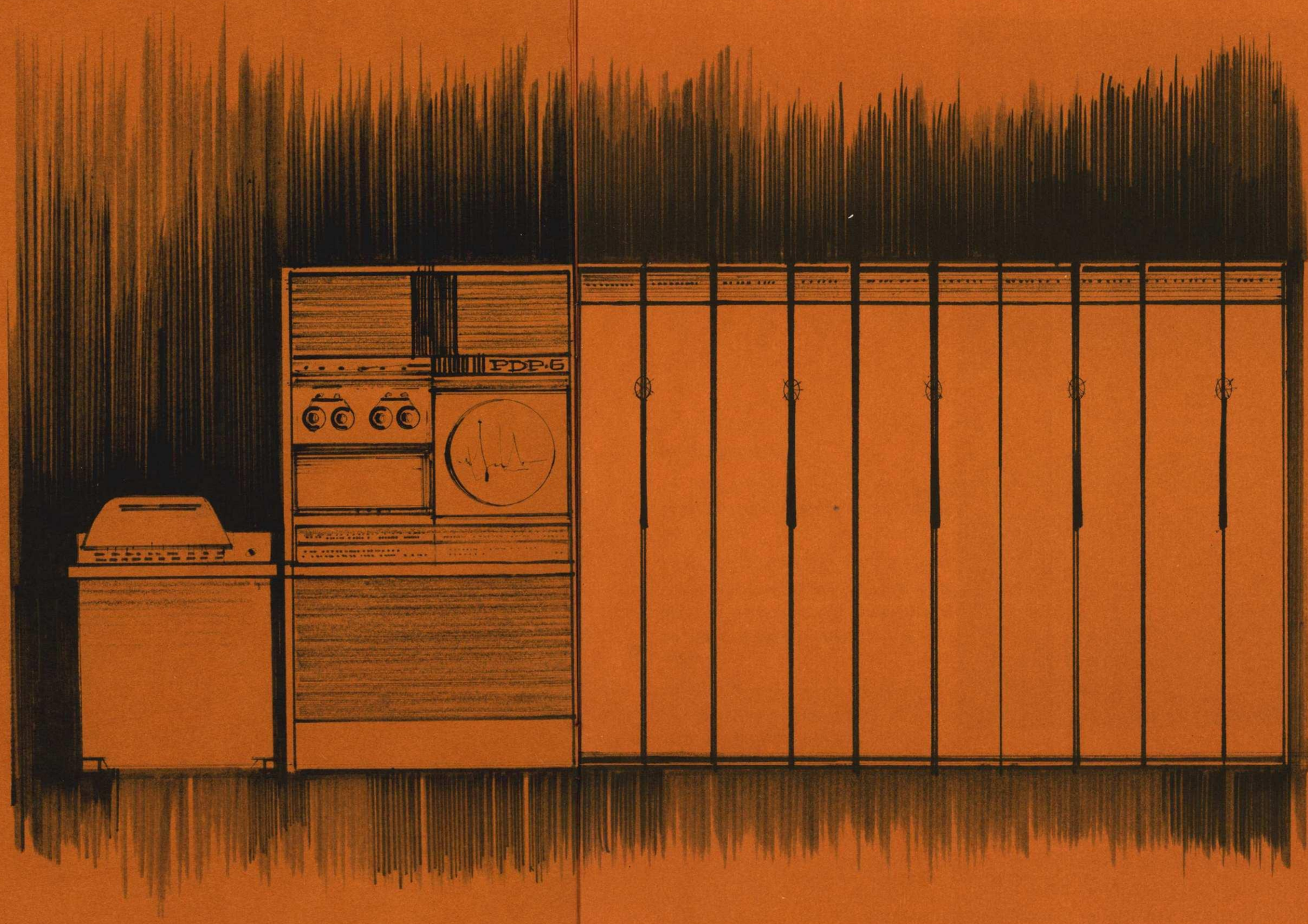
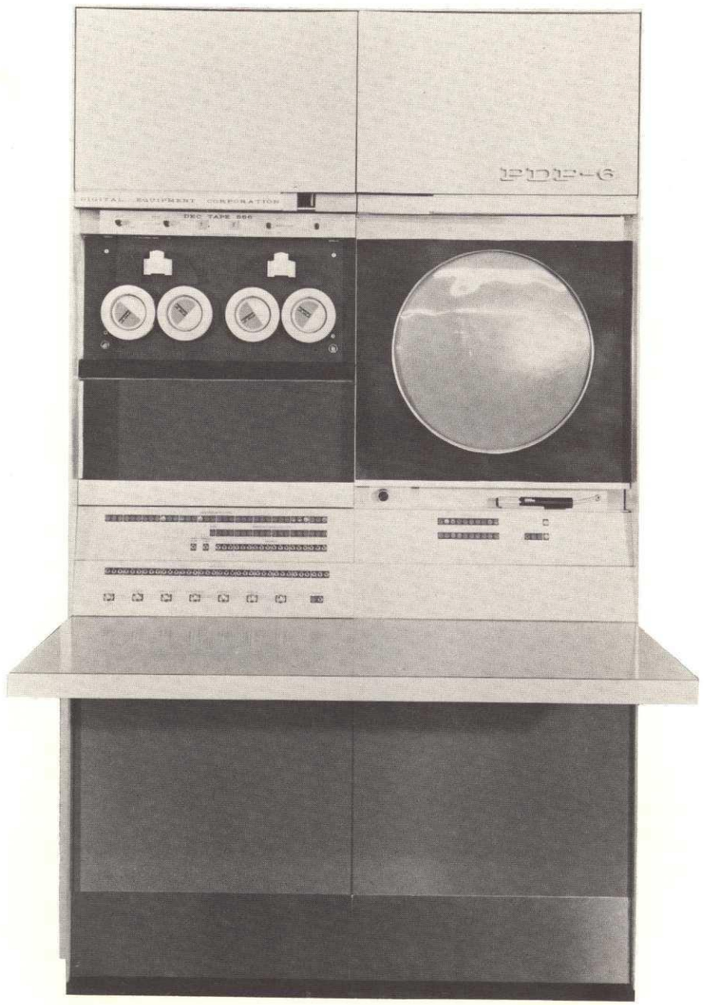
DIGITAL EQUIPMENT CORPORATION

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

F-61 Printed in U.S.A. 5M 5/63


PDP-6



Programmed Data Processor-6 is a medium scale general purpose computer with a high operating speed, unique logical organization, and growth capability. It is designed for use as a research tool in controlling experiments, collecting data, and automatically analyzing data. In addition, the computer is exceptionally well suited for use in scientific computation centers. The PDP-6 prototype is now under construction, and production deliveries will begin in early 1964.

PDP-6

8 ACCUMULATORS — Increase effective computing speeds

15 INDEX REGISTERS — Provide extensive capability for address modification in program loops

16-WORD FAST MEMORY — Has a 0.5-microsecond cycle time, is used interchangeably for accumulators, index registers, list pointers, program flags, and small program loops

16,384-WORD MAIN MEMORY — Has a 4-microsecond cycle time, is asynchronous with Central Processor

262,144 WORDS MAXIMUM — All directly addressable. Memory can be expanded and mixed in speed. Overlapping increases system speed

520 INSTRUCTIONS — Well organized and complete, provide excellent manipulative power. Features include character manipulation, list processing, and floating point hardware

CENTRAL PROCESSOR

The PDP-6 Central Processor is designed to operate asynchronously with respect to memory. After the memory has transmitted the requested word to the Central Processor, each operates at its maximum speed independent of the other. Eight accumulators and 15 index registers are stored in the first eight memory addresses of the Fast Memory. Most instructions can simultaneously and directly address any of the accumulators and any memory word and use any index register. Thus in 8 microseconds the Processor can make up to four memory references in carrying out one instruction. The asynchronous design assures that all parts of the system are running at top speed and that waiting times are minimized.

The Central Processor also provides control for in-out transfers. For slow in-out devices, like typewriters, the information is moved from the typewriter buffer under program control. The typewriter subroutine normally would be initiated by a program interrupt originating with the typewriter.

For higher speed in-out devices, like magnetic tape, the Central Processor handles the data movement automatically to memory with no program required for individual word transfers. This type of block transfer is started by the program, but then continues with no interruption of normal computing other than the memory cycle required to store or fetch the in-out word.

For some real time applications where absolutely no interruption of normal computing can be allowed, a separate In-Out Processor is used. It moves information in or out of one memory module while the Central Processor is computing with another memory module.

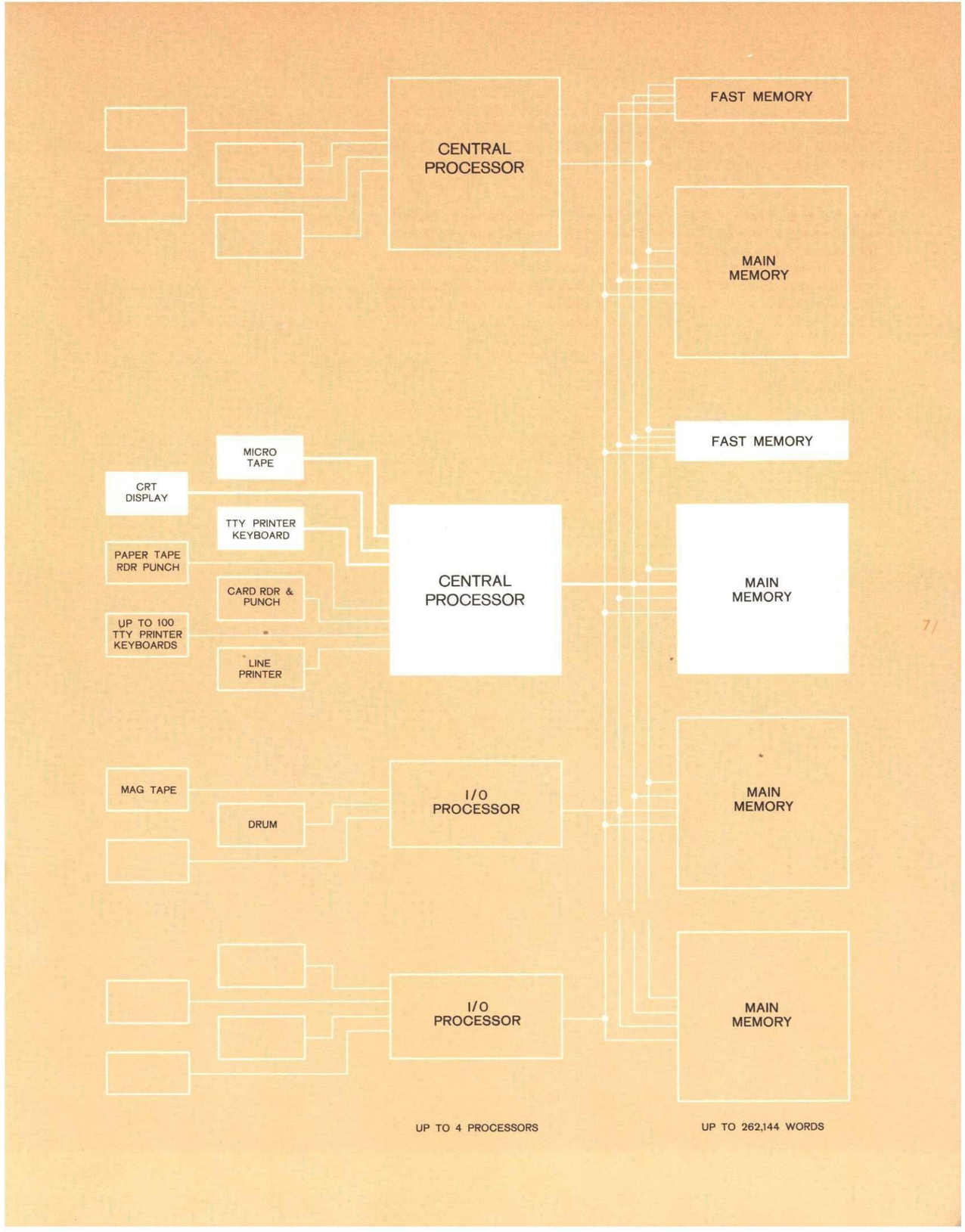
The instruction times for a basic configuration will vary, depending on the way in which the two memories are used. The following examples illustrate add times.

Example 1: add instruction is in Main Memory
operand is in Main Memory
indexing occurs
Complete Execution Time — 8 μsec

Example 2: add instruction is in Fast Memory
operand is in Main Memory
indexing occurs
Complete Execution Time — 3.5 μsec

MEMORY

PDP-6 memory is divided into two parts. Main Memory consists of 8192-word modules having a 4-microsecond cycle time. Fast Memory consists of 16-word modules having a 0.5-microsecond cycle time. Each memory module has facilities for connecting as many as four processors. These can be either Central Processors or In-Out Processors. Memory modules are self contained and can be operated simultaneously in parallel. This can occur with a single processor or with multiple processors.



The Fast Memory is addressable with normal memory addresses. Registers 0-7 can be addressed as index registers and/or accumulators. Memory overlap is automatically achieved due to the asynchronous relationship between the processors and memory.

This short program illustrates one use of the PDP-6 Fast Memory (0.5 microsecond) to increase the speed of performing iterative operations. Four instructions move Table 1 to Table 2 (table size 1000) in the times indicated. Times for performing the same operations exclusively in the Main Memory (4-microsecond) are shown for comparison.

Move "table 1" to "table 2," of length, size

	INSTRUCTIONS			FAST AND MAIN MEM.	MAIN MEM. ONLY
	6,mov	i	1000	(4 + 0.5)*	(4 + 4)*
y	5,mov		table 1,6	4 + 0.5	4 + 4
	5,mov	d	table 2,6	4 + 0.5	4 + 4
	6,soa	g	y	1	4
				10 μsec	20 μsec
				× 1000 = 10 ms	× 1000 = 20 ms

*occurs first time only

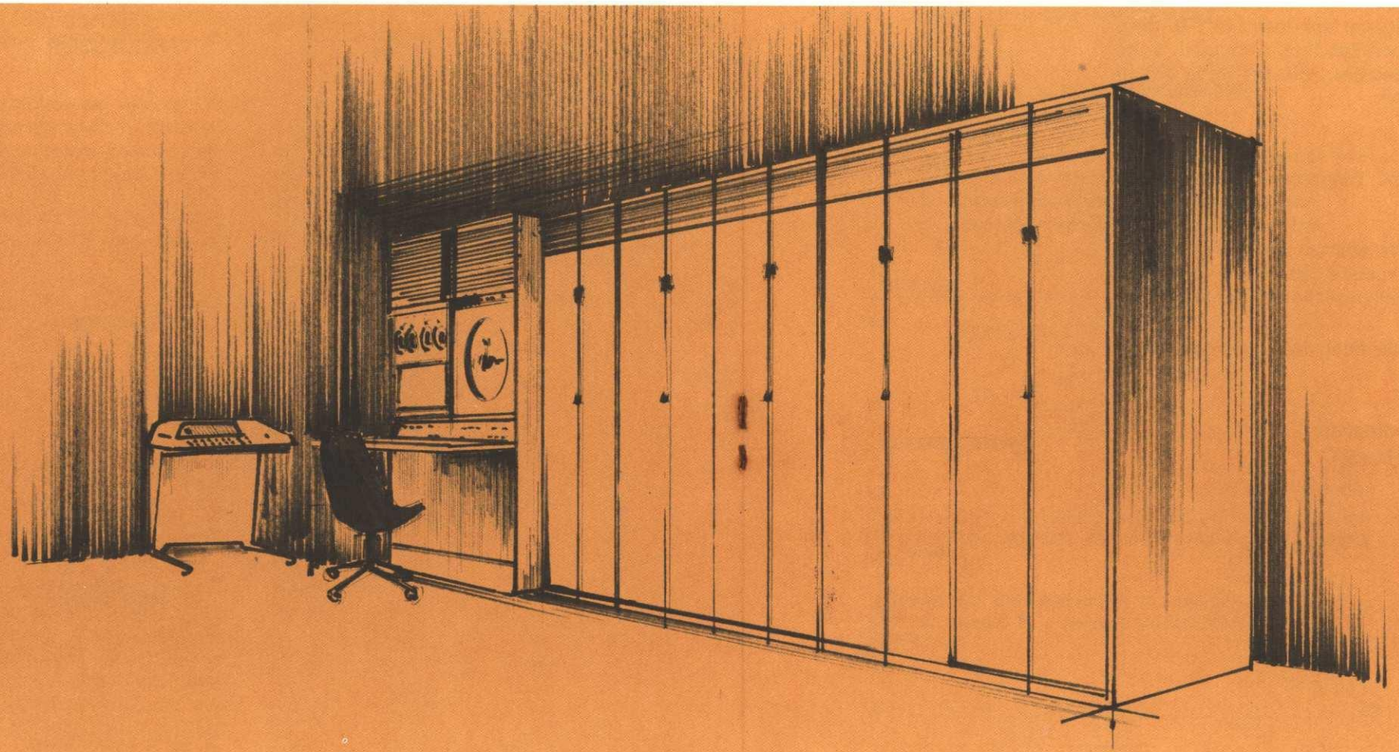
INPUT-OUTPUT EQUIPMENT

Three types of input-output devices are included in the standard PDP-6: a dual Micro Tape Transport (fixed address magnetic tape for high speed loading and readout, as well as program updating), a Teletype Printer-Keyboard, and a Precision CRT Display with Light Pen for direct, high speed man-machine communication.

A variety of optional devices can be added from Digital's standard line of peripheral equipment: high speed line printer, card reader and punch, perforated tape reader and punch, magnetic tape transports and controls, serial and parallel drums, ultra-precision and special purpose digital CRT displays, and other digital equipment compatible with PDP-6.

SOFTWARE

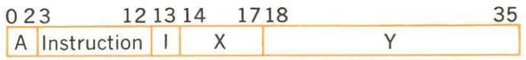
A complete software package is furnished with PDP-6. It includes an algebraic compiler, a symbolic assembler, an executive system, and utility routines. The executive system permits editing of Micro Tape, calls the compiler and assembler when needed, and calls subprograms that facilitate program checkout.



INSTRUCTIONS

PDP-6 has an exceptionally powerful set of 520 instructions, including such special features as immediate, direct, and indirect addressing, push-down list, character handling, memory modification and conditional branching, and operand preservation by one instruction. While the number of instructions is large, they are logically related and consistent. Good organization results in straightforward programming.

The primary instruction format is shown below. The instruction part is made up of the operation and the mode. Normally the operation specifies the action to be taken and the mode determines either the location of the results or conditions to be tested.



Each instruction word has an effective address, computed using I, X, and Y. Y is the address part, which may specify any of 262,144 locations. When the X, or index, part is non-zero, the contents of the specified index register are added to Y to form a new address. When the I, or indirect bit, is 0, the address thus computed is the effective address. In addition each instruction word specifies, with the three bits in A, one of eight accumulators to be used by the instruction.

When the indirect bit is 1, the address computed above is used to reference a memory location whose contents are used to form a new address. This procedure is repeated until a location is referenced whose I bit is 0.

Direct or indirect instructions designate as the effective address a memory register which contains the data to be used. A third type of instruction can specify (by choice of mode) an immediate address, in which case Y itself is taken as the number to be used.

The accumulators and index registers are stored in identical fast memory registers. Their addresses correspond as shown below.

Fast Memory Addresses	Accumulator Addresses	Index Register Addresses	
0	0	10	unused
1	1	11	1
:	:	:	
7	7	17	7
10			
:			
17			

DATA TRANSMISSION

FULL WORD TRANSMISSION

Operations

mov	Move full word
move	Move full word with right and left halves exchanged
movc	Move complement of full word
movn	Move negative of full word
movm	Move absolute magnitude of full word
mmov	Multiple move. Move full words 0, 1 . . .A

HALF WORD TRANSMISSION

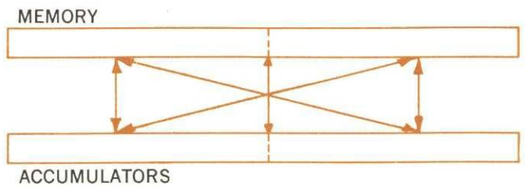
Operations

movr	Move right half only
movl	Move left half only
mvlr	Move left half to right half
mvrl	Move right half to left half

FULL OR HALF WORD TRANSMISSION

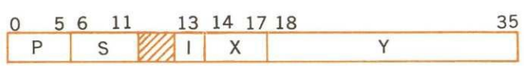
Modes

—	Load accumulator with direct address
i	Load accumulator with immediate address
d	Deposit accumulator with direct address
x	Exchange accumulator with direct address



Full flexibility for making full word or half word transfers between memory and accumulators is built into the PDP-6

CHARACTER MANIPULATION — Characters of arbitrary size are located by a position pointer word of the following format:



I, X, and Y are used in the usual manner to locate the word containing the character. P is the position of the character such that the rightmost bit of the character is 35-P. S is the size of the character, which may be from 1 to 36 bits.

ldc	Load accumulator (right-justified) with character specified by the pointer word. The effective address of the instruction locates the pointer word.
dpc	Deposit character (right-justified) of the accumulator in character position specified by pointer, as above.
cao	Add one character position to character pointer word. The position is moved to the right by the size of the character. If the result is off the right end of the word, the Y part is incremented and the P part is reset to the leftmost character position.
ldci	Same as ldc, but increments pointer word by one character after loading.
dpci	Same as dpc, but increments pointer word by one character after depositing.

PUSH DOWN LIST

psh	Takes the word at the effective address and puts it on the push-down list. The accumulator address part points to the next free location at the completion of the instruction. The left half of the accumulator is counted up by one.
pshs	Push and skip. Same as psh, but skips if the accumulator bits 0-17 are not zero
pop	Subtract one from each half of the accumulator. Take the word specified by the address of the accumulator and put it at the effective address.
pops	Pop and skip. Same as pop, but skips if contents of accumulator bits 0-17 are not zero.

INPUT-OUTPUT — Set initial conditions in I/O equipment, interrogate status, and transfer data between devices and memory.

inm	Input word from device n to memory
inz	Input, mask with immediate, and skip if zero
inn	Input, mask with immediate, and skip if non-zero
inch	Input character from device n to memory
outm	Output word to device n from memory
outi	Output immediate to device n
outch	Output character to device n from memory

ARITHMETIC

FIXED POINT ARITHMETIC — Data (either immediate or direct) and accumulator can be combined using 2's complement arithmetic. Mode specifies location of the results. In general the carry flag will be set if the result is greater than 2^{36}; the overflow flag will be set if the result is greater than 2^{35}.

Operations

add	Add full word
addr	Add to right half only
addl	Add to left half only
addrl	Add right half to left half
addlr	Add left half to right half
adds	Add both halves swapped
sub	Subtract full word
subr	Subtract from right half only
subl	Subtract from left half only
subrl	Subtract right half from right half
sublr	Subtract left half from right half
subs	Subtract right half from left half
iml	Integer multiply
idv	Integer divide
imlu	Integer multiply unsigned
idvu	Integer divide unsigned
mul	Multiply, giving double length result. Low order part goes to A + 1 when appropriate
umul	Unsigned multiply, double length result
div	Divide, giving quotient and remainder. Remainder goes to A + 1 when appropriate
undiv	Unsigned divide.

Modes

—	Direct address combined with accumulator, results to accumulator
d	Direct address combined with accumulator, results to memory
b	Direct address combined with accumulator, results to both memory and accumulator
i	Immediate address combined with accumulator, results to accumulator

FLOATING POINT ARITHMETIC AND CONVERSION — Floating point numbers can be combined using a word format of 1 bit for sign, 8 bits for exponent, and 27 bits for fraction. The mode specifies location of the results. Conversion is fixed-to-floating and floating-to-fixed. The effective address is used as a scaling constant.

Instructions

fad	Floating Add
fsb	Floating Subtract
fmp	Floating Multiply
fdv	Floating Divide

Modes

—	Accumulator is combined with the direct address, results to Accumulator
d	Same, but results to direct address
b	Same, but results to both direct address and accumulator
r	Same but results to accumulator, remainder and low significant bits to accumulator plus one

Instructions

flo	Convert fixed to floating
fix	Convert floating to fixed
dflo	Convert double length fixed to floating
dfix	Convert double length floating to fixed
fsc	Floating point scale

LOGICAL

Operations

and	And
ior	Inclusive Or
xor	Exclusive Or
anc	And complement. Clear selected bits

Modes

—	Direct address combined with accumulator, results to accumulator
d	Direct address combined with accumulator, results to direct address
b	Direct address combined with accumulator, results to direct address and accumulator address
i	Immediate address combined with accumulator, results to accumulator

SHIFTING

There are three kinds of shifting, as follows:

rotate — bits leaving one end enter at the other

arithmetic shift — performs 2's complement multiplication by powers of 2. The sign is unchanged. When going to the right, the first bit shifted in is the Exclusive Or of the sign and overflow, and remaining bits are the sign. When going left, overflow may be set.

logical shift — bits leaving one end are lost, and zeros enter the other end

Shifts may be of an accumulator, A, or of a combined word consisting of A and A + 1.

ral	Rotate accumulator left
rar	Rotate accumulator right
aal	Arithmetic shift accumulator left
aar	Arithmetic shift accumulator right
lal	Logical shift accumulator left
lar	Logical shift accumulator right
rcl	Rotate combined registers left
rcr	Rotate combined registers right
acl	Arithmetic shift combined registers left
acr	Arithmetic shift combined registers right
lcl	Logical shift combined registers left
lcr	Logical shift combined registers right

MEMORY AND ACCUMULATOR MODIFICATION AND TESTING

There are two sets of like instructions in this group:

Memory modification instructions — Modify the direct addressed register. The result is transmitted to the specified accumulator, except that a zero in the A position specifies no accumulator is to be changed. If the condition specified by the mode is met, the next instruction is skipped.

Accumulator modification instructions — Modify the specified accumulator. If the condition specified by the mode is met, control is transferred to the effective address.

Memory Skips	*Accumulator Transfers*	*Operations*
skp	tra	No modification, test full word
skpl	tral	No modification, test left half only
skpr	trar	No modification, test right half only
clm	cla	Clear register
aom	ado	Add one to full word
aoml	adol	Add one to left half, test left half only
aomr	ador	Add one to right half, test right half only
aomb	adob	Add one to both halves, test full word
som	sbo	Subtract one from full word
soml	sbol	Subtract one from left half, test left half only
somr	sbor	Subtract one from right half, test right half only
somb	sbob	Subtract one from both halves, test full word
msw	asw	Swap two halves of word, test full word
cmm	cma	Logical complement
ngm	nga	Negate (take 2's complement)
mgm	mga	Take magnitude (2's complement if negative)

Modes

Memory Skips	*Accumulator Transfers*	*Skip if Condition Met*
—	—	Never
s	t	Always
z	z	Equals zero
n	n	Not equal zero
ge	ge	Greater than or equal to zero
g	g	Greater than zero
le	le	Less than or equal to zero
l	l	Less than zero

COMPARE ACCUMULATOR WITH MEMORY AND SKIP — Compare a data word, which may be an immediate or direct address, with an accumulator by subtraction, and skip according to the mode selection. The results of the comparison do not affect memory or the accumulator.

Instructions

cfw	Compare accumulator with word
cfi	Compare accumulator with address
clh	Compare left half of accumulator with left half of direct addressed word
cli	Compare the left half of accumulator with zero (or left half of immediate address)
crh	Compare right half of accumulator with right half of word
cri	Compare right half of accumulator with address
clr	Compare left half of accumulator with right half of addressed word
clri	Compare left half of accumulator with immediate address
cao	Add one to accumulator, then compare the full words
caoi	Add one to accumulator, then compare the accumulator with the immediate address

Modes

—	Never skip
s	Always skip
e	Skip if equal
n	Skip if not equal
ge	Skip if accumulator is greater than or equal to data
g	Skip if accumulator is greater than data
le	Skip if accumulator is less than or equal to data
l	Skip if accumulator is less than data

ACCUMULATOR BIT MODIFICATION AND TESTING — Bits of an accumulator that are masked by bits of a data word can be tested and/or modified. The data word may be either direct or immediate. Mode specifies whether to use direct or immediate address for data word (normal or right and left halves exchanged) and conditions on which to skip.

Instructions

tst	Test only, do not change accumulator
clb	Clear bits in accumulator masked one by data word
cmb	Complement bits in accumulator masked one by data word
stb	Set bits in accumulator masked one by data word
stst	Test only with the data word, right and left halves exchanged. The data word and accumulator are unchanged
sclb	Clear bits in the accumulator masked one by the data word, right and left halves exchanged
scmb	Complement bits in the accumulator masked one by the data word, right and left halves exchanged
sstb	Set bits in the accumulator, masked one by the data word, right and left halves exchanged

Modes

—	Perform specified operation using directly addressed data
o	Skip if any accumulator bits masked one by the directly addressed data word are one, then perform the specified operation
z	Skip if all accumulator bits masked one by the directly addressed data word are one. Then perform the specified operation
s	Always skip, use directly addressed data word
i	Use immediate addressed data word
io	Same as o, but immediate addressed data word
iz	Same as z, but immediate addressed data word
is	Same as s, but immediate addressed data word

MISCELLANEOUS

SUBROUTINE CALLING

tsp	Transfer and save program counter. The PC and effective address are saved in the right and left halves of the accumulator, respectively. Control is transferred to the effective address.
tsa	Transfer and save accumulator. The accumulator is stored in the effective address. The PC and effective address are saved in the accumulator, as above. Control is transferred to the effective address plus one.
cal	Call subroutine. The accumulator is stored in memory register 20. The PC and effective address are saved in the accumulator as above.
trr	Transfer and restore accumulator. This is the return instruction that matches tsa. The accumulator is replaced by the word addressed in its left half. Control is transferred to the effective address.

EXECUTE

xct	Execute instruction at effective address

NO OPERATION

nop	No operation is performed

HALT

hlt	Halt and transfer

The above specifications are subject to change without notice.