CMU-CS-78-104

Impact of Implementation Design Tradeoffs on Performance: The PDP-11, A Case Study

Edward A. Snow
Daniel P. Siewiorek
February 19, 1978

Departments of Electrical Engineering
and Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania

CMU-CS-78-104

Impact of Implementation Design Tradeoffs on Performance: The PDP-11, A Case Study

Edward A. Snow
Daniel P. Siewiorek
February 19, 1978

Departments of Electrical Engineering
and Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania

In order to develop methodologies that are useful in the design of complex systems, existing designs must be studied. The DEC PDP-11 was selected for a case study since there are a number of designs (eight considered here), the designs span a wide range in basic performance (7:1) and component technology (bipolar SSI to MOS LSI), and the designs represent relatively complex systems.

The goals of the paper are two-fold: 1) to provide actual data about design tradeoffs and 2) to suggest design methodologies based on this data. An archetypical PDP-11 implementation is described followed by model specific variations. These variations represent the design tradeoffs which are classified by area: technology, control, and data path.

Two methodologies are presented. A top-down approach uses microcycle and memory read pause times to account for 90% of the variation in processor performance. This approach can be used in initial system planning. A bottom-up approach uses relative frequency of functions to determine the impact of design tradeoffs on performance. This approach can be used in design space exploration of a single design. Finally, the general cost/performance design tradeoffs used in the PDP-11 are summarized.

This research was supported in part by the National Science Foundation under grant GJ-32758X and by an IBM Fellowship. Engineering documentation was supplied by Digital Equipment Corporation.

Contents

1. Introduction

As semiconductor technology has evolved, the digital systems designer has been presented with an ever increasing set of primitive components from which to construct systems: standard SSI, MSI, and LSI as well as custom LSI components. This expanding choice makes it more difficult to arrive at a near-optimal cost/performance ratio in a design. In the case of highly complex systems, the situation is even worse since different primitives may be cost effective in different subareas of such systems.

Historically, digital system design has been more of an art than a science. Good designs evolved from a mixture of experience, intuition, and trial and error. Only rarely have design methodologies been developed (e.g. two-level combinatorial logic minimization, wire wrap routing schemes, etc.). Effective design methodologies are essential for the cost-effective design of more complex systems. In addition, if the methodologies are sufficiently detailed, they can be applied in high-level design automation systems [Siew76].

Design methodologies may be developed by studying the results of the human design process. There are at least two ways to study this process. The first involves a controlled design experiment where several designers perform the same task. By contrasting the results, the range of design variation and technique can be established [Thom77]. However, this approach is limited to fairly small design situations due to the redundant use of the human designers.

The second approach examines a series of existing designs that meet the same functional specification while spanning a wide range of design constraints in terms of cost, performance, etc. This paper considers the second approach and uses the DEC PDP-11[1] minicomputer line as a basis of study. The PDP-11 was selected due to the large number of implementations (eight are considered here) with designs spanning a wide range in performance (roughly 7:1) and component technology (bipolar SSI, MSI, MOS custom LSI). The designs are relatively complex and seem to embody good design tradeoffs as ultimately reflected by their price/performance and commercial success.

The design tradeoffs considered fall into three categories: circuit technology, control unit implementation, and data path topology. All three will be seen to have considerable impact on performance. Attention here is focused mainly upon the CPU. Memory performance enhancements such as caching are considered only insofar as they impinge upon CPU performance.

This paper is divided into three major parts. The first part (Section 2) provides an overview of the PDP-11 functional specification (e.g. architecture) and serves as background for subsequent discussion of design tradeoffs. The second part (Sections 3, 4, 5) presents an archetypical implementation followed by the model-specific variations from the archetype. These variations represent the design tradeoffs. The last part (Sections 6 and 7) presents methodologies for determining the impact of

[1] DEC, PDP, LSI-11, UNIBUS, and Fastbus are registered trademarks of Digital Equipment Corporation.

various design parameters on system performance. The magnitude of the impact is quantified for several parameters and the use of the results in design situations discussed.

2. Architectural Overview

The PDP-11 family is a set of small- to medium-scale stored-program central processors with compatible instruction sets [Bell70]. The family evolution in terms of increased performance, constant cost, and constant performance successors is traced in Figure 1[2]. Since the 11/45, 11/55 and 11/70 use the same processor, only the 11/45 is treated in this study.

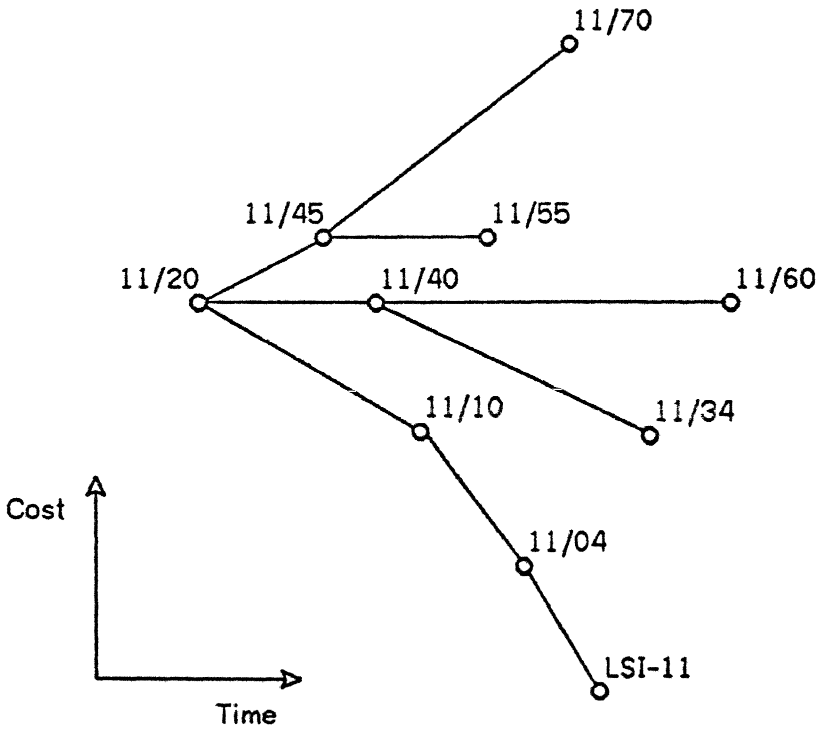
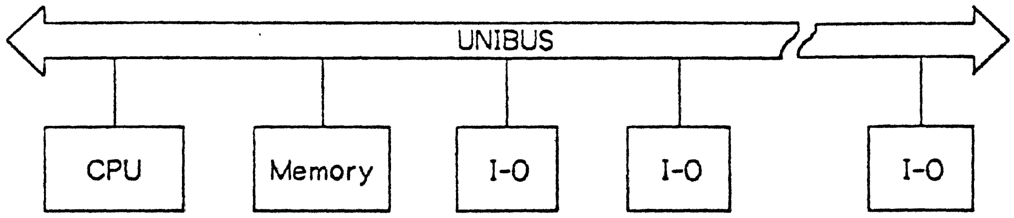


Figure 1: PDP-11 Family Tree

[2] The original equipment manufacturer (OEM) versions of the 11/10, 11/20, and 11/40 are the 11/05, 11/15, and 11/35 respectively. The OEM machines are electrically identical (or nearly so) to their end-user counterparts, the distinction being made for marketing purposes only.

A PDP-11 system consists of three parts: a PDP-11 processor, a collection of memories and peripherals, and a link called the UNIBUS over which they all communicate (Figure 2).



(Figure courtesy of Digital Equipment Corporation)

Figure 2: Typical PDP-11 Configuration

A number of features, not otherwise considered here, are available as options on certain processors. These include memory management and floating-point arithmetic. The next three subsections summarize the major architectural features of the PDP-11 including memory organization, processor state, addressing modes, instruction set, and UNIBUS protocol. The references list a number of processor handbooks and other documents which provide a more precise definition of the PDP-11 architecture than is possible here.

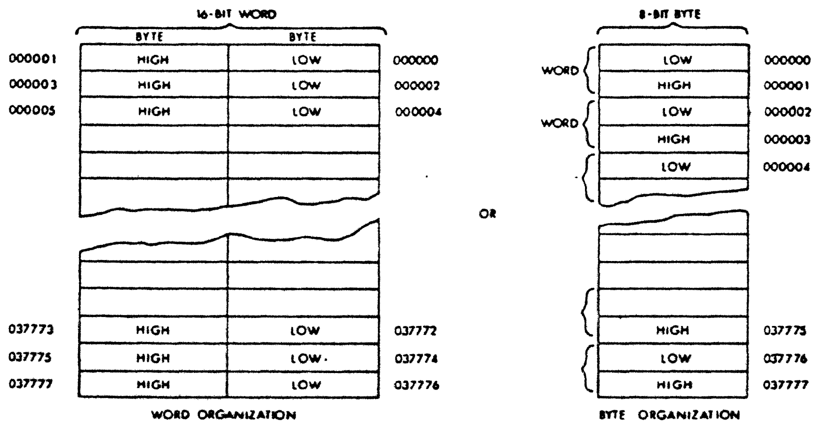
2.1 Memory and Processor State

The central processor contains the control logic and data paths for instruction fetching and execution. Processor instructions act upon operands located either in memory or in one of eight general registers. These operands may be either 8-bit bytes or 16-bit words.

Memory is byte or word addressable. Word addresses must be even. If N is a word address, then N is the byte address of the low-order byte of the word and N+1 is the byte address of the high-order byte of the word (Figure 3).

The control and data registers of peripheral devices are also accessed through the memory address space and the top 4K words of the space are reserved for this purpose.

The general registers are 16 bits in length and are referred to as R0 through R7. R6 is used as the system stack pointer (SP) to maintain a push-down list in memory upon which subroutine and interrupt linkages are kept. R7 is the program



(Figure courtesy of Digital Equipment Corporation)

Figure 3: PDP-11 Byte and Word Addressing

counter (PC) and always points to the next instruction to be fetched from memory. With minor exceptions (noted below) the SP and PC are accessible in exactly the same manner as any of the other general registers (R0 through R5).

Data manipulation instructions fall into two categories: arithmetic instructions (which interpret their operands as two's complement integers) and logical instructions (which interpret their operands as bit vectors). A set of condition code flags is maintained by the processor and is updated according to the sign and presence of carry/overflow from the result of any data manipulation instruction. The condition codes, processor interrupt priority, and a flag enabling program execution tracing are contained in a processor status word (PS), which is accessible as a word in the memory addressing space.

2.2 Addressing Modes and Instruction Set

The PDP-11 instruction set allows source and destination operands to be referenced via eight different addressing modes. An operand reference consists of a field specifying which of the eight modes is to be used and a second field specifying which of the eight general registers is to be used. The addressing modes are:

Mode 0 - *Register* - The operand is contained in the specified register.

Mode 1 - *Register deferred* - The contents of the specified register are used to address the memory location containing the operand.

Mode 2 - *Autoincrement* - The contents of the specified register are used to address the memory location containing the operand after which the register is incremented.

Mode 3 - *Autoincrement deferred* - The contents of the specified register address a word in memory containing the address of the operand in memory. The specified register is incremented after the reference.

Mode 4 - *Autodecrement* - The contents of the specified register are first decremented and then used to address the memory location containing the operand.

Mode 5 - *Autodecrement deferred* - The contents of the specified register are first decremented and then used to address a word in memory containing the address of the operand in memory.

Mode 6 - *Indexed* - The word following the instruction is fetched and added to the contents of the specified general register to form the address of the memory location containing the operand.

Mode 7 - *Indexed deferred* - The word following the instruction is fetched and added to the contents of the specified general register to form the address of a word in memory containing the address of the operand in memory.

The various addressing modes simplify the manipulation of diverse data structures such as stacks, tables, etc. When used with the program counter these modes enable immediate operands, absolute, and PC-relative addressing. The deferred modes permit indirect addressing.

Autoincrement/autodecrement modes operate differently for byte and word instructions. When a byte is referenced, the increment/decrement is by 1. In references to words (including addresses in the deferred modes) the increment/decrement is by 2. The use of R6 (SP) or R7 (PC) with these modes is an exceptional case. Since they generally must point to word addresses because of their use by the processor, R6 and R7 are always incremented/decremented by 2 and a word transfer made, even with byte instructions.

The PDP-11 instruction set is made up of the following types of instructions:

Single-operand instructions - A destination operand is fetched by the CPU, modified in accordance with the instruction, and then restored to the destination.

Double-operand instructions - A source operand is fetched followed by the destination operand. The appropriate operation is performed on the two operands and the result restored to the destination. In a few double operand instruction such as exclusive OR (XOR), source mode 0 (register addressing) is implicit.

Branch instructions - The condition specified by the instruction is checked, and if true, a branch is taken using a field contained in the instruction as a displacement from the current instruction address.

Jumps - Jump instructions allow sequential program flow to be altered either permanently (jump) or temporarily (jump to subroutine).

Control, trap, and miscellaneous instructions - Various instructions are available for subroutine and interrupt returns, halts, etc.

Floating-point instructions - A floating-point processor is available as an option with several PDP-11 CPUs. Floating-point implementation will not be considered in this paper.

A summary of PDP-11 addressing modes, instruction set, and other programming information is given in Table 1.

For the purposes of looking at the instruction execution cycle of the various PDP-11 processors, each cycle shall be broken into five distinct phases[3]:

Fetch - This phase consists of fetching the current instruction from memory and interpreting its opcode.

Source - This phase entails fetching the source operand for double operand instructions from memory or a general register and loading it into the appropriate register in the data paths in preparation for the execute phase.

Destination - This phase is used to get the destination operand for single and double operand instructions into the data paths for manipulation in the execute phase. For JMP and JSR instructions the jump address is calculated.

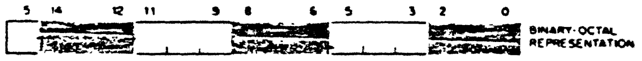
Execute - During this phase the operation specified by the current instruction is performed and any result rewritten into the destination.

Service - This phase is only entered between execution of the last instruction and fetch of the next to grant a pending bus request, acknowledge an interrupt, or enter console mode after the execution of a HALT instruction or activation of the console halt key.

The transitions from phase to phase are indicated in Figure 4.

[3] N.B.: The names are identical to those used by DEC to refer to instruction phases; however, their application here to a state within a given machine may differ from DEC's since the attempt here is to make the discussion consistent over all machines.

WORD FORMAT:



Mode	Name	Symbolic	Description
0	register	R	(R) is operand [ex. R2=%2]
1	register deferred	(R)	(R) is address
2	auto-increment	(R)+	(R) is adrs; (R) + (1 or 2)
3	auto-incr deferred	@(R)+	(R) is adrs of adrs; (R) + 2
4	auto-decrement	−(R)	(R) − (1 or 2); (R) is adrs
5	auto-decr deferred	@−(R)	(R) − 2; (R) is adrs of adrs
6	index	X(R)	(R) + X is adrs
7	index deferred	@X(R)	(R) + X is adrs of adrs

PROGRAM COUNTER ADDRESSING: Reg = 7

2	immediate	±n	operand n follows instr
3	absolute	@±A	address A follows instr
6	relative	A	instr adrs + 4 + X is adrs
7	relative deferred	@A	instr adrs + 4 + X is adrs of adrs

LEGEND:

Op Codes

■	= 0 for word/1 for byte
SS	= source field (6 bits)
DD	= destination field (6 bits)
R	= gen register (3 bits), 0 to 7
XXX	= offset (8 bits), +127 to −128
N	= number (3 bits)
NN	= number (6 bits)

Operations

( )	= contents of
s	= contents of source
d	= contents of destination
r	= contents of register
←	= becomes
X	= relative address
%	= register definition

Boolean

∧	= AND
V	= inclusive OR
∨̶	= exclusive OR
~	= NOT

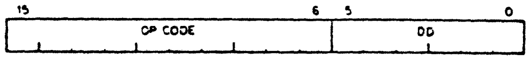
Condition Codes

*	= conditionally set/cleared
−	= not affected
0	= cleared
1	= set

NOTE:

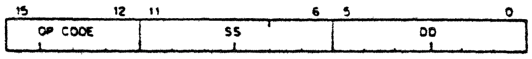
▲ = Applies to the 11/35, 11/40, 11/45 & 11/70 computers
● = Applies to the 11/45 & 11/70 computers

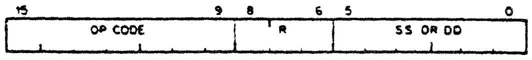
SINGLE OPERAND: OPR dst



Mnemonic	Op Code	Instruction	dst Result	N	Z	V	C
General							
CLR(B)	■ 050DD	clear	0	0	1	0	0
COM(B)	■ 051DD	complement (1's)	~d	*	*	0	1
INC(B)	■ 052DD	increment	d + 1	*	*	*	−
DEC(B)	■ 053DD	decrement	d − 1	*	*	*	−
NEG(B)	■ 054DD	negate (2's compl)	−d	*	*	*	*
TST(B)	■ 057DD	test	d	*	*	0	0
Rotate & Shift							
ROR(B)	■ 060DD	rotate right	→ C, d	*	*	*	*
ROL(B)	■ 061DD	rotate left	C, d ←	*	*	*	*
ASR(B)	■ 062DD	arith shift right	d/2	*	*	*	*
ASL(B)	■ 063DD	arith shift left	2d	*	*	*	*
SWAB	0003DD	swap bytes		*	*	*	0
Multiple Precision							
ADC(B)	■ 055DD	add carry	d + C	*	*	*	*
SBC(B)	■ 056DD	subtract carry	d − C	*	*	*	*
▲SXT	0067DD	sign extend	0 or −1	−	*	0	−

DOUBLE OPERAND: OPR src, dst OPR src, R or OPR R, dst





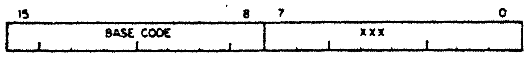
Mnemonic	Op Code	Instruction	Operation	N	Z	V	C
General							
MOV(B)	■ 1SSDD	move	d ← s	*	*	0	−
CMP(B)	■ 2SSDD	compare	s − d	*	*	*	*
ADD	06SSDD	add	d ← s + d	*	*	*	*
SUB	16SSDD	subtract	d ← d − s	*	*	*	*
Logical							
BIT(B)	■ 3SSDD	bit test (AND)	s ∧ d	*	*	0	−
BIC(B)	■ 4SSDD	bit clear	d ← (~s) ∧ d	*	*	0	−
BIS(B)	■ 5SSDD	bit set (OR)	d ← s v d	*	*	0	−
▲Register							
MUL	070RSS	multiply	r ← r x s	*	*	0	*
DIV	071RSS	divide	r ← r/s	*	*	*	*
ASH	072RSS	shift arithmetically		*	*	*	*
ASHC	073RSS	arith shift combined		*	*	*	*
XOR	074RDD	exclusive OR	d ← r ∨̶ d	*	*	0	−

(Table courtesy of Digital Equipment Corporation)

Table 1: PDP-11 Programming Summary

BRANCH: B - - location

If condition is satisfied:
Branch to location,
$\text{New PC} \leftarrow \underbrace{\text{Updated PC}\ +}_{\text{adrs of br instr} + 2} (2 \times \text{offset})$



Op Code = Base Code + XXX

Mnemonic	Base Code	Instruction	Branch Condition	
Branches				
BR	000400	branch (unconditional)	(always)	
BNE	001000	br if not equal (to 0)	≠ 0	Z = 0
BEQ	001400	br if equal (to 0)	= 0	Z = 1
BPL	100000	branch if plus	+	N = 0
BMI	100400	branch if minus	-	N = 1
BVC	102000	br if overflow is clear		V = 0
BVS	102400	br if overflow is set		V = 1
BCC	103000	br if carry is clear		C = 0
BCS	103400	br if carry is set		C = 1
Signed Conditional Branches				
BGE	002000	br if greater or eq (to 0)	≥0	N ⊻ V = 0
BLT	002400	br if less than (0)	<0	N ⊻ V = 1
BGT	003000	br if greater than (0)	>0	Z v (N ⊻ V) = 0
BLE	003400	br if less or equal (to 0)	≤0	Z v (N ⊻ V) = 1
Unsigned Conditional Branches				
BHI	101000	branch if higher	>	C v Z = 0
BLOS	101400	branch if lower or same	≤	C v Z = 1
BHIS	103000	branch if higher or same	≥	C = 0
BLO	103400	branch if lower	<	C = 1

JUMP & SUBROUTINE:

Mnemonic	Op Code	Instruction	Notes
JMP	0001DD	jump	PC ← dst
JSR	004RDD	jump to subroutine	use same R
RTS	00020R	return from subroutine	
▲MARK	0064NN	mark	aid in subr return
▲SOB	077RNN	subtract 1 & br (if ≠ 0)	(R) − 1, then if (R) ≠ 0: PC ← Updated PC − (2 x NN)

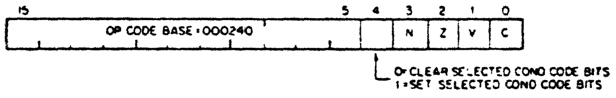
TRAP & INTERRUPT:

Mnemonic	Op Code	Instruction	Notes
EMT	104000 to 104377	emulator trap (not for general use)	PC at 30, PS at 32
TRAP	104400 to 104777	trap	PC at 34, PS at 36
BPT	000003	breakpoint trap	PC at 14, PS at 16
IOT	000004	input/output trap	PC at 20, PS at 22
RTI	000002	return from interrupt	
▲RTT	000006	return from interrupt	inhibit T bit trap

MISCELLANEOUS:

Mnemonic	Op Code	Instruction
HALT	000000	halt
WAIT	000001	wait for interrupt
RESET	000005	reset external bus
NOP	000240	(no operation)
●SPL	00023N	set priority level (to N)
▲MFPI	0065SS	move from previous instr space
▲MTPI	0066DD	move to previous instr space
●MFPD	1065SS	move from previous data space
●MTPD	1066DD	move to previous data space

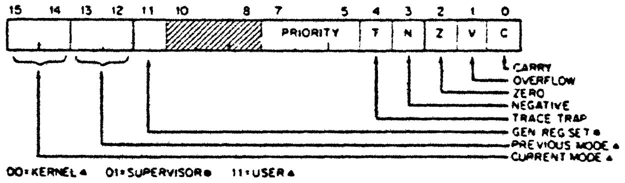
CONDITION CODE OPERATORS:


15
5
4
3
2
1
0
OP CODE BASE = 000240
N
Z
V
C
0 = CLEAR SELECTED COND CODE BITS
1 = SET SELECTED COND CODE BITS

Mnemonic	Op Code	Instruction	N Z V C
CLC	000241	clear C	- - - 0
CLV	000242	clear V	- - 0 -
CLZ	000244	clear Z	- 0 - -
CLN	000250	clear N	0 - - -
CCC	000257	clear all cc bits	0 0 0 0
SEC	000261	set C	- - - 1
SEV	000262	set V	- - 1 -
SEZ	000264	set Z	- 1 - -
SEN	000270	set N	1 - - -
SCC	000277	set all cc bits	1 1 1 1

PROCESSOR REGISTER ADDRESSES:

Processor Status Word
PS - 777 776



▲Stack Limit Register — 777 774

●Program Interrupt Request — 777 772

General Registers (console use only)	R0 — 777 700	R4 — 777 704
	R1 — 777 701	R5 — 777 705
	R2 — 777 702	R6 — 777 706
(not for 11/45)	R3 — 777 703	R7 — 777 707

Console Switches & Display Register — 777 570

Table 1 (continued): PDP-11 Programming Summary

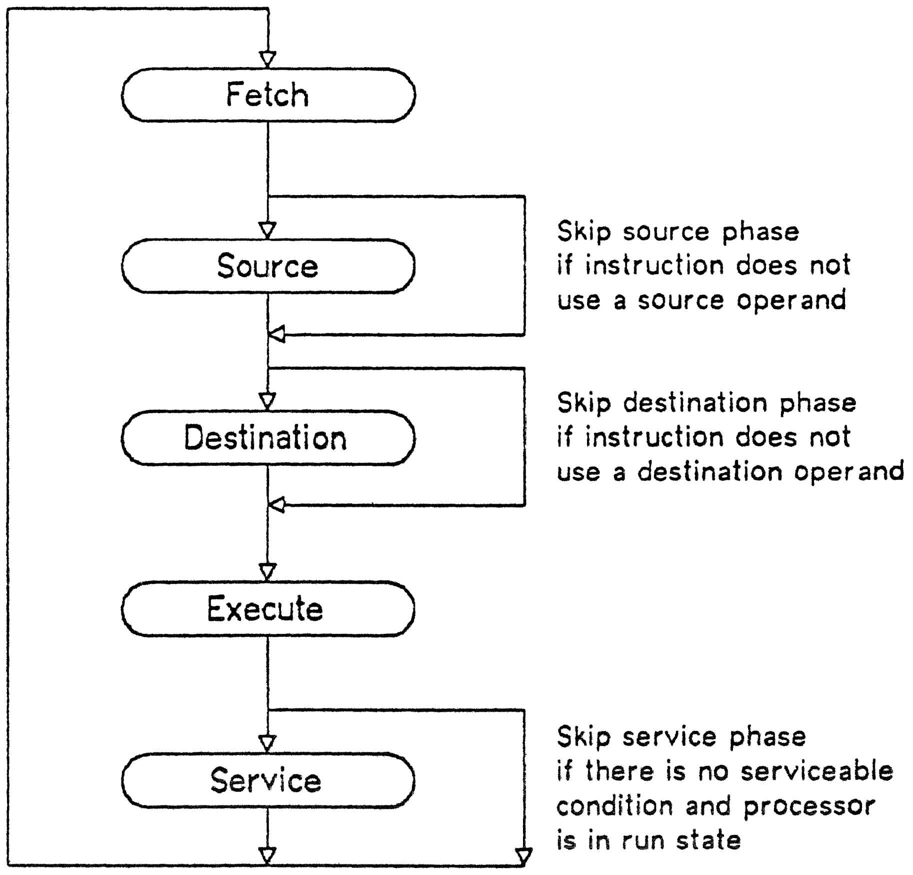

Fetch
Source
Destination
Execute
Service
Skip source phase if instruction does not use a source operand
Skip destination phase if instruction does not use a destination operand
Skip service phase if there is no serviceable condition and processor is in run state

Figure 4: PDP-11 Instruction Interpretation Cycle

2.3 The UNIBUS

All communication among the components of a PDP-11 system takes place on a set of bidirectional lines referred to collectively as the UNIBUS. The LSI-11 is an exception and uses an adaptation of the UNIBUS as explained in Section 4. The UNIBUS lines carry address, data, and control signals to all memories and peripherals attached to the CPU. Transactions on the UNIBUS are asynchronous with the processor. At any given time there will be one device which is bus master. The bus master may initiate communication with any device which it addresses, the addressed device becoming the bus slave. This communication may consist of data transfers or, in the case of the processor being slave, an interrupt request. The data transfers which may be initiated by the master are:

DATO - Data out - A word is transferred from master to slave.

DATOB - Data out, byte - A byte is transferred from master to slave.

DATI - Data in - A word is transferred from slave to master.

DATIP - Data in, pause - A word is transferred from slave to master and the slave awaits a transfer from master back to slave to replace the information that was read. The UNIBUS control allows no other data transfer to intervene between the read and the write cycles. This makes possible the reading and alteration of a memory location as an indivisible operation. In addition it permits the use of a read/modify/write cycle with core memories in place of the longer sequence of a read cycle followed by a write cycle.

3. Implementation of Medium-Performance PDP-11s

The broad middle range of PDP-11s have comparable implementations yet their performances vary by a factor of two. The processors making up this group are the PDP-11/04, 11/10, 11/20, 11/34, 11/40, and 11/60. This section discusses the features common to these implementations and the variations found between machines which provide the dimensions along which they may be characterized.

3.1 Common Implementation Features

All PDP-11 implementations, be they low-, medium-, or high-performance, can be decomposed into a set of data paths and a control unit. The data paths store and operate upon byte and word data and interface to the UNIBUS permitting them to read from and write to memory and peripheral devices. The control unit provides all the signals necessary to evoke the appropriate operations in the data paths and UNIBUS interface. Midrange PDP-11s have comparable data path and control unit implementations allowing them to be contrasted in a uniform way. In this section a basis for comparing these machines shall be established and used to characterize them.

3.1.1 Data Paths

An archetype may be constructed from which the data paths of all midrange PDP-11s differ but minimally. This archetype is diagrammed in Figure 5. All major registers and processing elements as well as the links and switches which interconnect them are indicated. The data path illustrations for individual implementations are grouped with Figure 5 at the end of the paper. These figures are laid out in a common format to encourage comparison. Note that with very few exceptions, all data paths are 16 bits wide (PDP-11 word size).

The heart of the data paths is the arithmetic/logic unit or ALU through which all data circulates and where most of the processing actually takes place. Among the operations performed by the ALU are addition, subtraction, ones and twos complementation, and logical ANDing and ORing.

The inputs to the ALU are the A leg and the B leg. The A leg is normally fed from a multiplexor (Aleg MUX) which may select from an operand supplied it from the scratchpad memory (SPM) and possibly from a small set of constants and/or the processor status register (PS). The B leg also is typically fed from its own MUX (Bleg MUX), its selections being among the B register and certain constants. In addition the Bleg MUX may be configured so that byte selection, sign extension, and other functions may be performed on the operand which it supplies to the ALU.

Following the ALU is a multiplexor (the AMUX) typically used to selects between the outpuf of the ALU, the data lines of the UNIBUS, and certain constants. The output of the AMUX provides the only feedback path in all midrange PDP-11 implementations except the 11/60 and acts as an input to all major processor registers.

The internal registers lie at the beginning of the data paths. The instruction register (IR) contains the current instruction. The bus address register (BA) holds the address placed on the UNIBUS by the processor. The program status register (PS) contains the processor priority, memory-management-unit modes, condition code flags, and instruction trace trap enable bit. The scratchpad memory (SPM) is an array of sixteen individually addressable registers which include the general registers (R0-R7) plus a number of internal registers not accessible to the programmer. The B register (Breg) is used to hold the B leg operand supplied to the ALU.

The variations from this archetype are minor as will be seen in Subsection 3.2. Variations to be encountered include routings for bus address and processor status register, the point of generation for certain constants, the positioning of the byte swapper, sign extender, and rotate/shift logic, and the use of of certain auxiliary registers present in some designs and not others. In general these variations are all peripheral to the major elements and interconnections of the data paths.

3.1.2 Control Unit

The control unit for all PDP-11 processors (with the exception of the PDP-11/20) is microprogrammed [Wilk53]. The considerations leading to the use of this style of control implementation in the PDP-11 are discussed in [Olou75]. The major advantage of microprogramming is flexibility in the derivation of control signals to gate register transfers, synchronize with UNIBUS logic, control microcycle timing, and evoke changes in control flow. The way in which a microprogrammed control unit accomplishes all of these actions impacts performance.

Figure 6 represents the archetypical PDP-11 microprogrammed control unit. The contents of the microaddress register determine the current control unit state and are used to access the next microinstruction word from the control store. Pulses from the clock generator strobe the microword and microaddress registers loading them with the next microword and next microaddress respectively. Repeated clock pulses thus cause the control unit to sequence through a series of states. The period spent by the control unit in one state is called a microcycle (or simply cycle when this does not lead to confusion with memory or instruction cycles) and the duration of the state as determined by the clock is known as the cycle time. The microword register shortens cycle time by allowing the next microword to be fetched from the control store while the current microword is being used.

Most of the fields of the microword supply signals for conditioning and clocking the data paths. Many of the fields act directly or with a small amount of decoding, supplying their signals to multiplexors and registers to select routings for data and to enable registers to shift, increment, or load on the master clock. Other fields are decoded based upon the state of the data paths. An instance of this is the use of auxiliary ALU control logic to generate function select signals for the ALU as a function of the instruction contained in the IR. Performance as determined by microcycle count is in large measure established by the connectivity of the data paths and the degree to which their functionality can be evoked by the data path control fields of the microprogram word.

The complexity of the clock logic varies with each implementation. Typically the clock is fixed at a single period and duty cycle; however, processors such as the 11/34 and 11/40 can select from two or three different clock periods for a given cycle depending upon a field in the microword register. This can significantly improve performance in machines where the longer cycles are necessary only infrequently. The clock logic must provide some means for synchronizing processor and UNIBUS operation since the two operate asynchronously with respect to one another. Two alternate approaches are employed in midrange implementations. Interlocked operation, the simpler approach, shuts off the processor clock, when a UNIBUS operation is initiated and turns it back on when the operation is complete. This effectively keeps microprogram flow and UNIBUS operation in lockstep with no overlap. Overlapped operation is a somewhat more involved approach which continues processor clocking after a DATI or DATIP is initiated. The microinstruction requiring the result of the operation has a function bit set which turns off the processor clock until the result is available. This approach makes it possible for the processor to continue running for several microcycles while a data transfer is being performed, improving performance.

The sequence of states through which the control unit passes would be fixed if not for the branch-on-microtest (BUT) logic. This logic generates a modifier based upon the current state of the data paths and UNIBUS interface (contents of the instruction register, current bus requests, etc.) and a BUT field in the microword currently being accessed from the control store which selects the condition on which the branch is to be based. The modifier (which will be zero in the case that no branch is selected or that the condition is false) is ORed in with the next microinstruction address so that the next control unit state is not only a function of the current state but also a function of the state of the data paths as well. Instruction decoding and addressing mode decoding are two prime examples of the application of BUTs. Certain code points in the BUT field do not select branch conditions, but rather provide control signals to the data paths, UNIBUS interface, or the control unit itself. These are known as active or working BUTs.

The JAM logic is a part of the microprogram flow-altering mechanism. This logic forces the microaddress register to a known state in the event of an exceptional condition such as a memory access error (bus timeout, stack overflow, parity error, etc.) or power up by ORing all ones into the next microaddress through the BUT logic. A microroutine beginning at the all-ones address handles these trapped conditions. The old microaddress is not saved (an exception to this occurs in the case of the

PDP-11/60); consequently, the interrupted microprogram sequence is lost and the microtrap ends by restarting the instruction interpretation cycle with the fetch phase.

The structure of the microprogram is determined largely by the BUTs available to implement it and by the degree to which special cases in the instruction set are exploited by these BUTs. This may have a measurable influence on performance as in the case of instruction decoding. The fetch phase of the instruction cycle is concluded by a BUT that branches to the appropriate point in the microcode based upon the contents of the instruction register. This branch can be quite complex since it is based upon source mode for double operand instructions, destination mode for single operand instructions, and opcode for all other types of instructions. Some processors can perform the execute phase of certain instructions like set/clear condition code during the last cycle of the fetch phase meaning that the fetch or service phases for the next instruction might also be entered from BUT IRDECODE. Complicating the situation is the large number of possibilities for each phase. For instance, there are not only eight different destination addressing modes, but also subcases for each that vary for byte and word and for memory modifying, memory non-modifying, MOV, and JMP/JSR instructions.

Some PDP-11 implementations such as the 11/10 make as much use of common microcode as possible to reduce the number of control states. This allows much of the IR decoding to be deferred until some time into a microroutine which might handle a number of different cases, for instance, byte and word operand addressing is done by the same microroutine in a number of PDP-11s. With the cost of control states dropping with the cost of control store ROM, there has been a trend toward providing separate microroutines optimized for each special case as in the 11/60. Thus more special cases must be broken out at the BUT IRDECODE making the logic to implement this BUT increasingly involved. There is a payoff, though, because there is a smaller number of control states for IR decoding and fewer BUTs. Performance is boosted as well since frequently occurring special cases such as MOV register to destination can be optimized.

3.1.3 Typical Instruction Interpretation Cycle

To get a feel for the PDP-11 data paths and control unit in operation, consider the interpretation of a representative instruction by the archetypical PDP-11. The instruction to be followed is a word bit set (BIS), an instruction which takes its source operand, logically ORs it with the destination operand, and returns the result to the destination. Register addressing with register 2 is used for the source, indexed addressing with register 7 used for the destination. This means that general register 2 will supply the source operand; the destination operand is in a memory location with address calculated by adding the contents of register 7 to the contents of the memory location following the instruction. Since register 7 is the program counter, the index following the instruction is effectively a displacement from the instruction to the destination operand.

What follows is the sequence of microinstructions evoked during the execution of the macroinstruction described above. Each microinstruction is numbered and consists of the register transfers and any UNIBUS operation or branch-on-microtest initiated by the microword.

Notation used in microinstructions:

B	= B register
BA	= bus address register
BUSDATA	= UNIBUS data lines
CLKOFF	= stop the processor clock until a UNIBUS transaction is completed, used for processor/UNIBUS overlap
IR	= instruction register
PC	= program counter (scratchpad register 7)
RD	= scratchpad register addressed by macroinstruction destination field (IR<2:0>)
RS	= scratchpad register addressed by macroinstruction source field (IR<8:6>)
SRCOPR	= scratchpad register 10 (not accessible to programmer), used as a temporary for source operands
a OP b	= operand a (on the A leg of the ALU) and operand b (on the B leg of the ALU) are combined according to the operation specified by the macroinstruction. The ALU function is selected by the auxiliary ALU logic as described in (3.1.2).
a ← b	= register a is loaded with operand b

Phase	Cycle	Operation	Explanation
FETCH	1	BA ← PC; DATI; CLKOFF	A read operation is initiated to fetch the instruction addressed by the program counter.
	2	IR ← BUSDATA	The instruction is placed in the instruction register.
	3	PC ← PC+2; BUT IRDECODE	The program counter is incremented to address the next location in the instruction stream (in this case the location containing the index for the destination). The instruction (held in the IR) is decoded by the BUT and found to be a double-operand instruction causing a branch to the microcode for source mode 0.

SOURCE	4	SRCOPR ← RS; BUT DESTINATION	The contents of the register addressed by the source field of the instruction (register 2) are copied into the scratchpad register reserved for source operands. The next state is determined by the destination addressing mode and the fact that BIS is a word instruction which modifies its destination.
DESTINATION	5	BA ← PC; DATI	A read operation is initiated to get the index word (pointed to currently by the program counter) for the effective address of the destination operand.
	6	PC ← PC+2; CLKOFF	The program counter is incremented to point to the next instruction. Note that this cycle is overlapped with the DATI started in cycle 5.
	7	B ← BUSDATA	The index is stored for use in the next cycle.
	8	BA ← RD+B; DATIP; CLKOFF	The index is added to the contents of the destination register to form the effective address of the destination operand. A DATIP is performed to read the operand since the operand is to be modified and then restored to its original location in memory.
	9	B ← BUSDATA	The destination operand is stored so it is available to the B leg of the ALU.

EXECUTE	10	BUSDATA ← SRCOPR OP B; DATO; CLKOFF; BUT SERVICE	The source and destination operands are logically ORed together and put out on the UNIBUS to be rewritten into the memory location from which the destination operand was read. (Note that the destination address is still in BA.) Upon completion of the DATO, the control unit will branch into the service phase if a serviceable condition is pending, otherwise it will branch back to repeat the fetch phase for the next instruction. Although it performs an execute phase function, this microinstruction is part of the same destination mode microroutine that generated cycles 5 through 9.

At a detailed level, the instruction interpretation process of each PDP-11 implementation will vary significantly from that outlined above; however, the scenario is still highly representative of the operation of the control unit and data paths in the designs to be considered.

3.2 Characterization of Individual Implementations

A set of common implementation features may be used to characterize each midrange PDP-11 to provide the raw data upon which comparisons may be based. A summary of these characteristics is given in Tables 2 and 3.

3.2.1 PDP-11/20

The 11/20 was the original member of the PDP-11 family. The 11/20 is atypical in a number of important aspects. Because the semiconductor read-only memory technology which makes microprogramming economically attractive was largely undeveloped when the PDP-11/20 was designed, control was implemented in random logic in contrast to the microprogrammed control used in all the succeeding members of the PDP-11 family. This causes control to be forced into a very stylized form so as to minimize the number of control unit states. Finally, the UNIBUS control generates a number of signals controlling the operation of the data paths. This makes it necessary for the UNIBUS and processor control unit to operate in tight lockstep with each other with no possibility of asynchronous data transfer.

The absence of MSI also has significant impact on the implementation of the data paths (Figures 7 and 8). The extensive use of SSI logic has several ramifications beyond increased cost and complexity. The Aleg and Bleg MUXes are set up to act as latches in addition to acting as data selectors (Figure 8). One may think of a Breg being placed between the Bleg MUX and the ALU. The ALU is a simple adder in contrast to the multifunctioned TTL MSI 74181 ALUs used in every other medium-performance PDP-11. Logical operations are carried out in the Aleg MUX/latch. The MUX can select either the true or complemented form of operands to support logical NOT. Logical OR is accomplished by gating the two operands into the MUX simultaneously (one operand may have been latched beforehand). Logical AND is performed by making use of DeMorgan's Rule ($A \wedge B \equiv \sim[\sim A \vee \sim B]$). Since there is no logic for complementing the output of the Aleg MUX/latch, two cycles are necessary: the first to form $\sim A \vee \sim B$, the second to run it through the Aleg MUX again to form the complement. The rotate/shift/byte swap logic is built into the MUX following the adder. A final peculiarity of the 11/20 is the separate paths provided from the UNIBUS for the IR and PS. Interestingly enough, even with all of these rather striking differences in implementation, the PDP-11/20 still shows a strong kinship to its successors.

3.2.2 PDP-11/40

The PDP-11/40 was designed to improve upon the performance of the PDP-11/20 without an increase in price by taking advantage of the TTL MSI technology arising after the introduction of the 11/20. With the exception of the PDP-11/60 (and the 11/20 which exceeds the 11/40 in cost), the 11/40 is both the fastest and most expensive midrange PDP-11 processor.

The data paths of the 11/40 (Figure 9) correspond closely to those of the archetype except in the immediate vicinity of the ALU. What has been indicated as the Aleg MUX is really the negative-logic wired OR of a number of signals. Options such as the floating-point processor are added by simply tying them into the DMUX output and Aleg. Two paths exist out of the PS: one running to the Aleg MUX as in the archetype and a second running directly to the UNIBUS as in the 11/20. A path from the Aleg MUX directly to the DMUX (equivalent to the AMUX of other models) exists allowing the ALU (and thus the propagation delay incurred by passing through it) to be bypassed in those cases where the contents of the SPM or PS are to be routed directly back to the Breg or SPM. Single-bit shifts and rotates right are handled in the DMUX in a fashion similar to the 11/20. Rotate/shifts to the left, however, are performed in the ALU. Sign extension and byte swapping are performed in the Bleg MUX. Since the scratchpad register may not be both simultaneously read and written, the D register (Dreg) is used to hold results generated while the SPM is being read in one processor clock phase so that during a later phase they may be written back into the scratchpad. In this way the Dreg permits read/write access of the SPM within a single cycle. A final feature is the presence of two paths into the bus address register, one from the Aleg MUX and one from the ALU. This is of benefit in such operations as autoincrement and autodecrement addressing modes in which the contents of a register can be modified and either the premodification (autoincrement) or postmodification (autodecrement) value of the the register put into the bus address register in a single cycle.

The 11/40 microprogrammed control unit is quite elaborate to gain full benefit

Table 2 - PDP-11 Circuit Technology and Data Paths

Model	Performance Relative to LSI-11	Circuit Technology		Data Paths					
		Logic Family	Level of Integration	Scratchpad Memory	ALU	Sign Extension	Rotate/shift	Byte Swap	other Features
LSI-11	1.000	N-channel MOS	LSI	• organized 26 registers × 8 bits • 1 write/2 read ports	8-bit nMOS ALU	not needed done in microcode	in ALU	not needed done in microcode	• 8-bit wide data paths, 16-bit operands require two cycles • Non-UNIBUS, data/address lines mixed
11/04	1.455	TTL	MSI	16 × 16 with SPreg for write after read	74181's with 74182 carry lookahead	in Bleg MUX	Breg is bidirectional shift register	before SPM	• Complementer at ALU A leg for subtract instruction
11/10	1.436	TTL	MSI	16 × 16 read and write may not take place within same cycle	74181's with 74182 carry lookahead	in Bleg MUX	Breg is bidirectional shift register	none performed as 8 shifts	—
11/20	1.667	TTL	SSI	16 × 16 with input latches for write after read	7482 adders, ripple carry plus combinatorial logic	in Bleg MUX/latch	following adder	following adder	• Bus data has own path to IR & PS • PS has own path out to bus data, no other outgoing paths
11/34	1.942	TTL S-TTL	MSI	16 × 16 write while read	74S181's with 74S182 carry lookahead	following AMUX	Breg is bidirectional shift register	following AMUX, speeds odd-byte accesses	• B extension register (BXreg) for EIS instructions
11/40	2.819	TTL	MSI	16 × 16 Dreg and multi-phase cycle allow write after read	74181's with 74182 carry lookahead	in Bleg MUX	to left in ALU to right in DMUX	in Bleg MUX	• Bypass from Aleg MUX to AMUX around ALU & Dreg • Two paths into BA
11/45	6.820 (with bipolar memory)	S-TTL	MSI	• two banks of 16 × 16 for 1 write/2 read ports • read and write may not occur the same cycle	74S181's with 74182 carry lookahead	in ALU	to left in ALU to right in SHFMUX	in SHFMUX	• PC broken out separately from scratchpads • Multiple paths into ALU • Fastbus supports semiconductor memory
11/60	3.727 (87% cache hit ratio)	S-TTL	MSI	• two banks of 32 registers × 16 bits • only R0-R7 & user R6 duplicated • write after read	74S181's with 74182 carry lookahead	in shift tree	in shift tree	in shift tree	• Shift tree allows multibit shifts • Scratchpad C for constants, bus input, and status logging • 3-state logic used extensively

Table 3 - PDP-11 Control Unit and Physical Assembly

Model	Controller						Physical Assembly		
	Control Derivation	Cycle Time(s) (nanoseconds)	Processor/UNIBUS Synchronization	Control Store Size (bits × words)	Control Store Words Used	Other Features	Circuit Boards	Integrated Circuit Packages	Integrated Circuit Types
LSI-11	Vertical microcode	400	interlocked	22 × 1024 (expandable to 2048)	994	• No next microaddress in microword, microwords are selected sequentially until a branch, jump, or translate is encountered	1 quad (4 positions)	48	24
11/04	Horizontal microcode	260	interlocked	40 × 256	249	-	1 hex (6 positions)	138	40
11/10	Horizontal microcode	300 (150 for fast shift)	overlapped	40 × 256	249	• Microword is not buffered	2 hex (12 positions)	203	60
11/20	Random logic	280	interlocked	-	-	• Control states are encoded in major- and minor-state shift registers	6 quad, 6 double, 2 single (38 positions)	523	27
11/34	Horizontal microcode	180 240	interlocked	48 × 512	488	-	2 hex (12 positions)	231	54
11/40	Horizontal microcode	140 200 300	overlapped	56 × 256	251	• BUT field is buffered, BUT must be placed one microinstruction ahead of where it is to take place	4 hex, 1 quad (28 positions)	417	53
11/45	Horizontal microcode	150	overlapped	64 × 256	256	• Forks and microbranches may be enabled together, μbranches taking precedence	7 hex, 1 quad (46 positions)	696	78
11/60	Horizontal microcode	170	interlocked	48 × 2560 (excluding user control store space)	2410 (including integral floating point)	• Multilevel microsubroutines • Page-addressed microstore • Extensive use of residual control • Control store available to user through WCS	6 hex (36 positions)	648	74

of the potential of the data paths. Among its features are overlapped processor/ UNIBUS operation and three selectable microcycle clock periods. The latter feature increases performance immensely since the maximum cycle time of 300 nanoseconds is needed only when a full circle from scratchpad through ALU and back to scratchpad is made. In cycles which do not write into the scratchpad, a 200 nanosecond cycle may be selected. When the data paths are unused and only microbranching is involved, an even shorter cycle time of 140 nanoseconds is possible. A final unique feature of the 11/40 is a variation in the branch-on-microtest logic from that of the archetypical control unit. To increase microbranch speed, the microword BUT select field is buffered in the microword register rather than being routed directly from the control store to the BUT logic. This causes a one-cycle delay in processing the branch and forces all BUTs to be placed one microinstruction ahead of where they are to take effect. In some cases dummy steps are required to provide sufficient lead time for BUT action to occur, somewhat offsetting the speedup of this arrangement.

One way in which the 11/40 uses its processor/UNIBUS overlap feature to advantage is by prefetching words from memory whenever possible. At the end of the fetch phase, a check is made to see if the next memory reference fetches an instruction or operand index. If it does, the read access is begun immediately using the contents of the PC as the address. Exceptions to this are when the PC is used as a destination or when a service request is pending, both of which mean that the current value of the PC won't be the address of the next instruction. Starting the access early allows it to proceed in parallel with the execution of the current instruction. This reduces the time the processor idles waiting for the accessed word. Updating of the PC is deferred until the proper point in the instruction interpretation process is reached. This guarantees that references to the PC will result in the proper value being used.

3.2.3 PDP-11/10

The PDP-11/10 was designed as a minimal-cost processor. The implementation is again TTL MSI but stripped to the bare essentials without the elaboration of the 11/40. The data paths of the 11/10 (Figure 10) follow the conventions of the archetype closely. A constant zero may be selected onto the AMUX in addition to ALU or UNIBUS data. The ALU Aleg multiplexor allows selection of the PS, some constants, and some internal addresses as well as the scratchpad memory. The Breg is implemented as a universal bidirectional shift register so that single-bit shifts and rotates may be performed without additional logic. The ALU Bleg multiplexor includes the constants one and zero and permits sign extension of the low-order byte of the B register. The scratchpad memory may not be both read and written in the same cycle, thus operations such as incrementing the PC which takes only a single microcycle on other processors take two microcycles to complete on the 11/10. A byte swapping path is absent in the 11/10. As a consequence odd-byte addressing and swapping must be accomplished by a series of eight shifts or rotates.

The 11/10 control unit has a relatively austere implementation. There is no microword register in the control unit although there is necessarily a microaddress register. As a consequence, the output of the control store is used directly to condition the data paths. This precludes the overlap of current microinstruction execution with next microinstruction fetch. Hence, the propagation delay of the control store must be added to that of the data paths in setting the microcycle time, causing it to be a relatively long 300 nanoseconds. The simplicity of the data paths allows the use of a microword only 40 bits wide. The microcode contains very few frills and gains very little in performance from special cases. A notable example of this is the jump address calculation for JMP and JSR instructions. The 11/10 uses the same section of microcode for JMP and JSR destination modes as it uses to fetch conventional destination operands. This costs an extra memory reference over the separate microroutines used in other PDP-11 processors since not only is the effective address of the jump calculated, but also its contents are fetched (the microprogram logic precludes using this operand as a prefetched instruction even though this is effectively what it is). Overlapped processor/UNIBUS operation allows some of the extra microcycles necessitated by the data paths to be effectively hidden by putting them in parallel with UNIBUS accesses. The other concession to performance is clock speed doubling during shift operations to partially compensate for the performance lost in the absence of a byte swapper.

3.2.4 PDP-11/04

The PDP-11/04 is the simplest PDP-11 except for the LSI-11. Although simple, the 11/04 embodies a very good set of design tradeoffs. Figure 11 diagrams the 11/04 data paths. The scratchpad memory has a register (SPreg, part of the SPM shown in Figure 11) sitting between it and the AMUX. This register allows the scratchpad to support read/modify/write accesses saving a microcycle in each such access over the 11/10. A multiplexor sitting before the SPM implements the swap byte operation, allowing the halves of a word to be interchanged. This improves byte operation performance considerably over the 11/10 and obviates the need for the 11/10's fast shift logic. Also eliminated is overlapped processor/UNIBUS operation since the savings from it are reduced with the overall reduction in number of microcycles.

The AMUX (the major data bus and the multiplexor which drives it) can select the PS and a number of constants in addition to ALU output and UNIBUS data. Between the SPM and ALU is a ones complementor so that the 74181 ALU may be used to perform the Bleg minus Aleg operation used in the subtract instruction in addition to the Aleg minus Bleg operation used in the compare instruction. The Aleg MUX also directly drives the UNIBUS address lines without a bus address register (if processor/ UNIBUS overlap were used, a BA register would have been necessary). Between the Breg and ALU is a multiplexor which allows the Breg, sign-extended low-order byte of the Breg, or the constants zero or one to be selected into the Bleg of the ALU in a manner identical to that of the Bleg MUX of the 11/10. The Breg is also identical to that of the 11/10 in that it is a bidirectional shift register implementing rotate/shifts.

The final contributor to increased performance of the 11/04 is the decrease in cycle time from 300 nanoseconds in the 11/10 to 260 nanoseconds, made possible in part by pipelining the microword fetch. On the whole, the 11/04 is superior in performance to the 11/10 in all cases except the fetch phase and certain addressing modes where the use of its processor/UNIBUS overlap capability is sufficient to put the 11/10 ahead.

3.2.5 PDP-11/34

The PDP-11/34 is an elaboration of the 11/04. The 11/34 data paths (Figure 12) bear close resemblance to those of the 11/04. The 11/04 complementor has been replaced in the 11/34 by additional microcode which reverses the placement of source and destination operands on the A and B legs of the ALU during the subtract instruction from that of the other double operand instructions. This frees the 11/34 from performing the adjustments that must be made in the data paths of other PDP-11 processors to make the subtract instruction operate correctly under the restrictions of the 74181 ALU. Added is a B extension register (BXreg) which, when concatenated with the Breg, forms a 32-bit register for double-width operands and results manipulated by extended instruction set operations such as multiply and divide. Also notable is the relocation of the byte swapper to the tail of the AMUX allowing odd-byte accessing to occur as data is entered from or placed upon the UNIBUS without the customary extra microcycle needed in other implementations to right adjust the byte. Included with the byte swapper is the sign extension logic. Schottky TTL is used in critical places in the data paths, notably the ALU, to speed up microcycle time from the 260 nsec of the 11/04 to 180 nsec. Additional hardware for memory management (not shown in Figure 12) and extended instruction set microcode are standard features.

The 11/34 microprogrammed control unit makes some concessions to the improved performance of the data paths. In addition to the normal 180 nanosecond cycle, there is a 240 nanosecond cycle used primarily for UNIBUS operations. Again, there is no processor/UNIBUS overlap feature because considerations of simplicity (i.e. cost) outweighed the incremental improvement in performance that would be netted. Because of its additional logic, the PDP-11/34 has a wider microword than the 11/04 (48 bits versus 40 bits). Also, since many more cases are broken out by the BUT IRDECODE in the 11/34 than in the machines preceding it, the size of the control store has been increased to 512 words, double that of earlier horizontally microprogrammed implementations.

3.2.6 PDP-11/60

The PDP-11/60 is the latest implementation covered in this paper and in many ways the most unique. Its design exploits advances in circuit technology occurring since the introduction of the earlier models giving it a number of features which set it apart from other PDP-11 family members. Two major enhancements are a larger microcode addressing space, making an integral floating-point instruction set and a writable control store option feasible, and a cache memory[4]. Both are possible due to increases in the density and decreases in the cost of bipolar ROM and RAM [Mudg77].

[4] The PDP-11/70 also uses a cache.

As illustrated in Figure 13, the 11/60 data paths show significant differences from those of other midrange implementations. A major difference is the presence of three scratchpad memories feeding the ALU. Scratchpads A and B are 32-word-by-16-bit register arrays, each having twice the number of registers of the single scratchpad found in other midrange designs. As with the 11/45 (Section 5), the contents of the general registers are kept in both scratchpads allowing different registers to be read onto the A and B legs of the ALU simultaneously within the same cycle. This speeds register-to-register operations. The additional registers in the A and B scratchpads are used as floating-point registers by the integral floating-point microcode, working storage by user microprograms, and console, maintenance, and status registers by the processor. Scratchpad C is a 16-word-by-16-bit array which holds bus data and constants used by the processor and takes the place of the constants ROM on the B leg of other midrange implementations. During exceptional situations these constants may be overwritten with other information but must be restored before execution of the base machine microcode may be resumed.

The 11/60 is the first PDP-11 implementation to make use of three-state devices to eliminate many of the multiplexors used in other designs (the 11/40 uses open-collector logic on the Aleg bus to the same effect). For instance, instead of actual Aleg and Bleg MUXes, the 11/60 uses registers and combinatorial elements with three-state outputs that can be independently enabled onto a common bus for each ALU leg. The ALU itself is the conventional '181 type used in all of the other MSI implementations. As in the 11/40, the D register (Dreg) latches the ALU output so that results may be rewritten to the scratchpads during a later clock phase of the microcycle in which they are generated. The output of the Dreg is the major, but not sole, feedback route in the data paths.

The bus address register (BA) is loaded from the Aleg bus as in the 11/04 and 11/34. The address out bus is driven by the BA and supplies addresses to the memory subsystem (cache, relocation hardware, and UNIBUS interface). The bus data in (DIN) bus routes data into the processor from the memory subsystem, internal registers accessed via UNIBUS addresses such as the PS, and constants emitted by the microinstruction word. Scratchpad C and the instruction register are loaded directly from DIN in a manner reminiscent of the 11/20. A register in SPM C is set aside specifically for transfers from memory to the data paths. Results are routed from the data paths back to the memory subsystem and internal registers via a separate bus data out (DOUT) bus.

As compared to the other midrange machines, several data path elements are unique to the 11/60. The counter (Cntr) is an iteration counter used by the extended instruction set and floating-point microcode. The shift register and shift register guard (shown together as the SR in Figure 13) can be loaded in parallel with Dreg and shifted one position right or left. Either all or the low-order seven bits of the SR may be gated onto the Aleg bus through the XMUX (not shown). The shift tree is a network of multiplexors used for byte swapping, sign extension, and field isolation and positioning. It is unusual in that it allows right shifts of from 1 to 14 bit positions combinatorially in a single microcycle.

The PDP-11/60 control unit is horizontally microprogrammed in much the same manner as the other midrange implementations. Extensive use of Schottky logic throughout the processor allows a fixed 170 nanosecond microcycle time. Processor/ UNIBUS communication is interlocked unlike either the 11/40 or 11/45. There are several significant differences from the more conventional implementations. Many of these differences are generalizations of the microprogram flow control mechanism to allow more functions of the base machine to be performed by microcode rather than hardwired logic and to create a user microprogramming environment which can be put to uses beyond executing the PDP-11 instruction set. The 11/60 has a larger and more generalized set of BUTs than earlier machines. Also included for the first time in a horizontally microprogrammed machine is a multilevel microsubroutine call/return capability.

Increased reliance on microcode has expanded the control store to 4096 words by 48 bits. 2560 words of this are used to implement the basic machine. The remaining 1536 words are available to the user through a ROM control store option; 1024 are available through a writable control store option. Since addressing the microstore requires 12 bits, a page-addressing scheme has been adopted to avoid widening the microword. Page size is 512 words reducing microaddresses to 9 bits within a page. Microbranches across a page boundary require that an additional 3-bit page field be specified.

Another concept used extensively in the 11/60 to reduce microword size is residual control. In this technique relatively static control information is kept in set-up registers separately from the microword. The microprogram must load these registers to affect the data path elements which they control. Set-up registers are used in the 11/60 to gate registers onto to DIN bus, enable data into registers from the DOUT bus, select SR functions, and control certain actions of the shift tree.

The overlapping of a number of different control fields by bit steering is a final means of keeping the microword relatively narrow. Certain bits in the microword control the interpretation of corresponding microword fields. This allows a single field to control several different functions. The one drawback of this technique is that these functions become mutually exclusive within a single microword since their simultaneous use would involve two different interpretations of the same microfield.

Hardwired logic in the memory subsystem detects internal addresses in a manner similar to other PDP-11 processors. However, the actual access to these registers is accomplished through microcode instead of additional control logic. Internal address access has been added to the exceptional conditions detected by the JAM logic of the 11/60. If the JAM microroutine finds that a microtrap has been caused by an internal address access, then an intraprocessor transfer to or from the addressed register is performed. Unlike other JAM sequences, such transfers are terminated by resuming the interrupted microprogram. Microcoded register access requires much more time than the corresponding hardwired access. Reading the PS, for instance, takes 33 microcycles or 5.610 microseconds using microcode where a single microcycle suffices for the hardwired approach. This is justified, however, by the decreased cost of microcode versus hardwired logic and by the infrequent access made to these registers.

Like the 11/40, the 11/60 prefetches instructions and operand indices whenever possible. Unlike the 11/40, the PC is incremented at the time the prefetch is performed. Because of this, prefetching cannot be done when the current instruction uses the PC as either a source or destination register. A second difference is that service requests are not polled until the end of the current instruction, when the next instruction may already be prefetched and the PC updated. When this occurs, two microcycles must be spent to decrement the PC to restore its old value before proceeding with the service phase.

4. Implementation of a Minimal-Cost PDP-11

The LSI-11 (known in packaged form as the PDP-11/03) is designed for the low-end market where there is more concern for low cost than high performance. Integrated circuit package count and printed circuit board area, the main determinants of manufacturing cost, are kept low through an n-channel MOS LSI technology implementation of the CPU. The result is a PDP-11 processor with four kilowords of semiconductor memory on a single 8.5" x 10.5" (standard DEC quad height) printed circuit board which can execute the entire PDP-11/40 instruction set.

The constraints imposed by current semiconductor technology dictate much of the implementation of the LSI-11. The entire CPU consists of four LSI packages plus a number of standard TTL SSI and MSI packages for clock generation and bus interfacing. A system control chip provides microinstruction addressing logic plus an interface to external signals used in bus control. A data paths chip contains the registers and arithmetic/logic unit of the machine. Two chips are microcode ROMs (MICROMs). Each contains 512 microinstruction words with a width of 22 bits. An optional third MICROM adds the extended instruction set/floating-point instruction set option of the PDP-11/40. To decrease the complexity of the machine, the traditional UNIBUS was abandoned in favor of a scheme requiring fewer bus lines. Most notable is the multiplexing of both data and addresses onto a single set of 18 data/address lines, DAL<17:00>. A significant savings over the 34 lines dedicated to data and address in the UNIBUS results at the expense of bus cycle speed.

The 22 bit microinstruction word of the LSI-11 is quite narrow compared to the microwords of the horizontally microprogrammed PDP-11s which range from 40 to 64 bits wide. Four bits are not decoded and provide direct TTL-compatible signals which are used by logic external to the CPU chips. Another two bits are used within the CPU chips to control next microinstruction addressing. The remaining 16 bits are decoded as a microinstruction by the CPU chips. LSI-11 microinstructions differ little in form from conventional minicomputer instructions with their opcode and operand (which may be register, microcode address, or literal) fields. These require a great deal more decoding than the horizontal microinstructions of other designs.

The LSI-11 microstore is larger than the control store of any other PDP-11 except the 11/60. Since LSI-11 microinstructions lack the possibilities for parallelism inherent in the horizontal microinstructions, more LSI-11 microinstructions are needed to code a given operation. In addition, certain functions which are handled with combinatorial logic in other PDP-11 control units and data paths are microcoded in the LSI-11. Finally, the LSI-11 has more elaborate console microcode than the other implementations. As a result, the LSI-11 has 22528 bits of microstore versus 14336 bits for the PDP-11/40, 16384 bits for the PDP-11/45, and 122880 bits for the PDP-11/60. The narrow microword is used in spite of its attendant problems due to a limitation imposed by the packaging of the MOS CPU chips. Only 40 pins are available to carry power and signals to and from each chip, limiting the number of lines available for transmitting the microword from the MICROMs to the control and data path chips.

Technology also imposes a serious constraint on instruction decoding. The equivalent of a branch on microtest allows only 8 bits to be decoded at a time. This is sufficient for decoding the majority of instructions; however, the remainder require additional decoding which may consume as many as 8 microcycles. This is in marked contrast with all other PDP-11s which require only a single microcycle to do the initial instruction decode at the end of the fetch phase (BUT IRDECODE)[5]. The effect that this has on the average duration of the LSI-11 fetch phase is evident from Table 4.

Figure 14 details the data paths around which the operands of the macroinstruction level machine circulate. As with the medium-performance implementations, the ALU is the hub of activity, operating upon quantities supplied from the scratchpad memory. The AMUX selects among the output of the ALU, the high or low byte of the data/address lines, and the processor flags. The selected quantity is fed back to be rewritten into the scratchpad. Constants supplied as literals from the microinstruction word may be gated into the data paths through the Bleg MUX to the ALU. Additional paths exist for transmitting information in and out on the data/address lines.

Significant differences exist between the data paths of the LSI-11 and the midrange machines in addition to the similarities. One major difference is in the width of the data paths. The LSI-11 is the only member of the PDP-11 family with data paths 8 bits rather than 16 bits wide. This is necessitated by limitations in current semiconductor chip density. Bus paths in particular occupy large amounts of chip real estate dictating their reduction in width. Since only 8 bits of data can be processed at a time, two microcycles are required to accomplish any 16-bit operation. A second effect is the elimination of logic that would otherwise be necessary to configure the data paths for both byte and word operations. A last unique characteristic is the absence of a B register for feeding the B leg of the ALU. Instead, the B leg is fed from a second read port into the scratchpad memory. In this the LSI-11 bears a curious resemblance to the PDP-11/45 and 11/60. The difference is that while the LSI-11 uses this feature to eliminate cycles that would be needed to load a Breg, there is not sufficient logic to allow source and destination registers to be accessed simultaneously. Consequently, multiple cycles are still required to set up register/ register operations on the LSI-11.

The final important performance factor is again a direct result of the circuit technology employed. NMOS logic is not as fast as the bipolar logic found in every other PDP-11 implementation so that the microcycle time of the LSI-11 is 400

[5] The 11/60 requires two microcycles to decode certain instructions.

nanoseconds or one-third slower than the next slowest PDP-11. This coupled with the larger number of microcycles necessary to execute a given macroinstruction causes the LSI-11 to lag in performance.

5. Implementation of a High-Performance PDP-11

The PDP-11/45 was designed for maximum performance and followed the 11/20 to become the second member of the PDP-11 family. Maximum performance is achieved with a complex set of data paths allowing highly parallel operation and an optional high-speed semiconductor memory (bipolar or MOS) with its own path into the processor called the Fastbus. The extensive use of Schottky TTL in the processor makes possible a 150 nanosecond cycle time, half as long as that in some midrange designs.

The complexity of the PDP-11/45 data paths is evident from Figure 15 even with several of the special-purpose registers and buses omitted for clarity. The overall organization still bears some resemblance to the midrange PDP-11 data paths, however. The ALU remains the hub of data path activity with its output the primary feedback path to the processor registers, although not the only one as in other implementations. The ALU is based upon the Schottky equivalent of the 74181 chip used in most other PDP-11 designs. The difference begins with the multiplexors driving the A and B legs of the ALU. These MUXes allow operands to be routed directly to the proper leg without using additional cycles to move operands from register to register. KOMUX and K1MUX (combined in Figure 15) are multiplexors used in conjunction with the BMUX to gate constants, trap vector addresses, and branch offsets into the B leg of the ALU.

Among the registers supplying the AMUX and BMUX are the source and destination operand registers (Sreg and Dreg respectively). These are in turn supplied by the SRMUX and DRMUX which select data from individual scratchpad registers or the program counter. Besides holding operands from the general registers, the Sreg and Dreg act as working registers. In particular Dreg is a shift register used to accumulate the less significant half of results during multiply and divide.

Separate scratchpads are maintained so that source and destination general registers may be read simultaneously and independently. This necessitates both scratchpads being written together to keep their contents identical. Each scratchpad is organized as 16 words of 16 bits each. Fifteen words in each scratchpad are actually used: two sets of general registers R0 through R5 and three sets of stack pointers (R6). Register set selection is controlled by status bits in the PS and permits fast context switching by eliminating the need to save and restore registers.

The program counter is not maintained in the scratchpad registers as in other PDP-11s. Rather, it is held separately so that it may be routed directly to the BAMUX while the Sreg and Dreg are occupied with other operations. Moreover, two program counters are implemented. PCB holds the current value of the program counter and is used as a general register or bus address. PCA holds the new value of the program counter allowing the PC to be updated while the old PC value is still in use, after which PCB is clocked to load it with the new value contained in PCA.

The SHFMUX can right shift or byte swap data from the ALU before it is clocked into the scratchpads. It also provides a route from PCB to the Sreg and/or Dreg when the PC is used as a general register. This arrangement precludes the shifting or byte swapping of data being loaded into the PC that is possible with data destined for one of the other general registers residing in the scratchpads. As a consequence, arithmetic shift left and byte swap operations on the PC do not cause the PC to be modified, although the condition codes are updated as though it were.

Processor access to the UNIBUS, Fastbus, and internal registers is via the bus register MUX (BRMUX), the bus register (BR and BRA), and the data out MUX (DMUX). The BR and BRA (the duplication is due to electrical loading considerations) are logically a single register as shown in Figure 15. They receive all incoming data and transmit almost all outgoing data in addition to accumulating the more significant half of results during multiply and divide. The BRMUX selects the input to the BR (and BRA) from among the two external buses and internal input bus for input to the processor and from the SHFMUX for output from the processor via the BR and DMUX to the external buses and internal output bus. The internal buses connect a number of special registers and an optional floating-point processor to the data paths. Of these, only the PS is indicated in Figure 15. The instruction register (duplicated as IR and AFIR, again for electrical loading reasons) are also loaded from the BRMUX but are only clocked when an instruction is fetched.

Bus addresses are applied directly to the UNIBUS or to an optional memory mapping unit by the bus address multiplexor (BAMUX). No bus address register is needed since memory access and processor clocking are fully interlocked except during an overlapped fetch in which case the PCB is held selected while operations continue in other parts of the data paths.

The PDP-11/45 control unit is horizontally microprogrammed and is for the most part quite similar to the archetype described for midrange PDP-11 implementations. The control store is 256 words by 64 bits. The relatively wide microword is necessary for generating the large number of control signals used in conditioning and clocking the complicated data paths. An additional source of complexity is the timing logic needed to produce and use the five processor clock phases.

There are two classes of microsequence-altering functions corresponding to the BUTs of other PDP-11s. The first class consists of simple branches having four or fewer possible branch addresses. These operate in the same fashion as BUTs. The second class of branches consists of three complex instruction decoding functions called forks. The first, fork A, does the initial instruction decode and corresponds to the BUT IRDECODE of other implementations. Fork B dispatches to an execute phase microroutine following a destination operand fetch. Fork C dispatches to a destination phase microroutine following a source operand fetch. A fork enable field in the microword is used to enable one fork at most during a cycle. When a fork and branch are combined in the same cycle, the fork is disabled if the branch is taken. This permits the implementation of certain functions without the use of additional cycles.

The 11/45 microcode is structured to take full advantage of the data paths and processor/UNIBUS overlap. Besides intensively exploiting special cases in the addressing modes and instruction set, the microprogram implements operand and instruction fetch overlap in much the same way as the 11/40. The one difference between the two prefetch mechanisms is that the 11/45 updates the PC value in PCB and stores it in PCA at the time the prefetch is started. References to the PC work correctly because PCB holds the old PC value until it is updated at the appropriate time.

All the design decisions described above are directed toward implementing the fastest system possible. Tradeoffs involving circuit technology and control unit and data path organization have all been made with this end in mind.

6. Measuring the Effect of Design Tradeoffs on Performance

There are two alternative approaches to the problem of determining just how the particular binding of different design decisions affects the performance of each machine:

1) *Top-down approach* - Attempt to isolate the effect of a particular design tradeoff over the entire space of implementations by fitting the individual performance figures for the whole family of machines to a mathematical model which treats the design parameters as independent variables and performance as the dependent variable.

2) *Bottom-up approach* - Make a detailed sensitivity analysis of a particular tradeoff within a particular machine by comparing the performance of the machine both with and without the design feature while leaving all other design features the same.

Each approach has its assets and liabilities for assessing design tradeoffs. The first method requires no information about the implementation of a machine, but does require a sufficiently large collection of different implementations, a sufficiently small number of independent variables, and an adequate mathematical model in order to explain the variance in the dependent variable to some reasonable level of statistical confidence. The second method, on the other hand, requires a great deal of knowledge about the implementation of the given system and a correspondingly great amount of analysis to isolate the effect of the single design decision on the performance of the complete system. The information that is yielded is quite exact, but applies only to the single point chosen in the design space and may not be generalized to other points in the space unless the assumptions concerning the machine's implementation are similarly generalizable. In the following subsections the first method is used to determine the dominant tradeoffs and the second method is used to estimate the impact of individual implementation tradeoffs.

6.1 Quantifying Performance

Measuring the change in performance of a particular PDP-11 processor model due to design changes presupposes the existence of some performance metric. Average instruction execution time was chosen because of its obvious relationship to instruction stream throughput. Neglected are such overhead factors as direct memory access, interrupt servicing, and, on the LSI-11, dynamic memory refresh. Average instruction execution times may be obtained by benchmarking or by calculation from instruction frequency and timing data. The latter method was chosen due to its freedom from the extraneous factors noted above and from the normal clock rate variations found from machine to machine of a given model. This method also allows us to calculate the change in average instruction execution time that would result from some change in the implementation. Such frequency-driven design has already been applied in practice to the PDP-11/60 [Mudg77].

The instruction frequencies are tabulated in Appendix A and include the frequencies of the various addressing modes. These figures were calculated from measurements made by Strecker [Stre76b] on 7.6 million instruction executions traced in 10 different PDP-11 instruction streams encountered in various applications. While there is a reasonable amount of variation of frequencies from one stream to the next, the figures of Appendix A should be representative.

Instruction times are tabulated in Appendices B through I. These times were calculated from the engineering documents for each machine. The times vary from those published in the PDP-11 processor handbooks for two reasons. First, in the handbooks, times have been redistributed among phases to ease the process of calculating instruction times. In the appendices the attempt has been to accurately characterize each phase. Second, there are inaccuracies in the handbooks arising from conservative timing estimates and engineering revisions. The figures included here may be considered more accurate.

A performance figure is arrived at for each machine by weighting its instruction times by frequency. The results, given in Table 4, form the basis of the analyses to follow.

	Fetch	Source	Dest.	Execute	Total	Speed Relative to LSI-11
LSI-11	2.514	0.689	1.360	1.320	5.883	1.000
PDP-11/04	1.940	0.610	0.811	0.682	4.043	1.455
PDP-11/10	1.500	0.573	0.929	1.094	4.096	1.436
PDP-11/20	1.490	0.468	0.802	0.768	3.529	1.667
PDP-11/34	1.630	0.397	0.538	0.464	3.029	1.942
PDP-11/40	0.958	0.260	0.294	0.575	2.087	2.819
PDP-11/45 (bipolar memory)	0.363	0.101	0.213	0.185	0.863	6.820
PDP-11/60 (87% cache hit ratio)	0.541	0.185	0.218	0.635	1.578	3.727

Table 4: Average PDP-11 Instruction Execution Times in Microseconds

6.2 Analysis of Variance of PDP-11 Performance: Top-Down Approach

The first method of analysis described above will be employed in an attempt to explain most of the variance in PDP-11 performance in terms of two parameters:

1) *Microcycle time* - The microcycle time is used as a measure of processor performance which excludes the effect of the memory subsystem.

2) *Memory read pause time* - The memory read pause time is defined as the period of time during which the processor clock is suspended during a memory read. For machines with processor/UNIBUS overlap, the clock is assumed to be turned off by the same microinstruction which initiates the memory access. Memory read pause time is used as a measure of the

memory subsystem's impact on processor performance. Note that this time is less than the memory access time since all PDP-11 processor clocks will continue to run at least partially concurrently with a memory access.

The choice of these two factors is motivated by their dominant contribution to, and (approximately) linear relationship with, performance. Keeping the number of independent variables low is also important due to the small number of data points being fit to the model.

The model itself is of the form:

$$t_i = k_1 c_{1i} + k_2 c_{2i}$$

, where t_i is the average instruction execution time of machine i from Table 4.
c_{1i} is the microcycle time of machine i (for machine with selectable microcycle times, the predominant time is used).
c_{2i} is the memory read pause time of machine i.

This model is only an approximation since it assumes k_1 and k_2 will be constant over all machines. In general this will not be the case. k_1 is the number of microcycles expected in a canonical instruction. This number will be a function mainly of data path connectivity and strictly speaking another factor should be included to take that variability into account; however, since the data path organization of all PDP-11 implementations considered here (excepting the 11/03, 11/45, and 11/60) are quite comparable, the simplifying assumption of calling them all identical at the price of explaining somewhat less of the variance shall be made. k_2 is the number of memory accesses expected in a canonical instruction and also exhibits some variability from machine to machine. A small part of this is due to the fact that some PDP-11s actually take more memory cycles to perform a given instruction than do others (this is really only a factor in certain 11/10 instructions, notably JMP and JSR, and the 11/20 MOV instruction). A more important source of variability is the UNIBUS/processor overlap logic incorporated into some PDP-11 implementations which effectively reduces the actual contribution of the k_2c_{2i} term by overlapping more memory access time with processor operation than is excluded from the memory read pause time.

Given the model and the dependent and independent data for each machine as given in Table 5, a linear regression was applied to determine the coefficients k_1 and k_2 and to find out how much of the variance is explained by the model.

Applying the regression over all eight processors: $k_1 = 11.580$, $k_2 = 1.162$, $R^2 = 0.904$. R^2 is the amount of variance accounted for by the model or 90.4%. If the regression is applied to just the six midrange processors: $k_1 = 10.896$, $k_2 = 1.194$, $R^2 = 0.962$. R^2 increases to 96.2% partly because fewer data points are being fit to the model and partly because the LSI-11 and 11/45 can be expected to have different k coefficients than the midrange machines and hence don't fit the model as well. Note that if two midrange machines, the 11/04 and the 11/40, are eliminated instead of the LSI-11 and 11/45, then R^2 decreases to 89.3% rather than increasing. The k coefficients are close to what should be expected for average microcycle and memory

	Independent Variables		Dependent Variable
	ucycle Time	Memory Read Pause Time	Average Instruction Execution Time
LSI-11	0.400	0.400	5.883
PDP-11/04	0.260	0.940	4.043
PDP-11/10	0.300	0.600	4.096
PDP-11/20	0.280	0.370	3.529
PDP-11/34	0.180	0.940	3.029
PDP-11/40	0.140	0.500	2.087
PDP-11/45 (bipolar memory)	0.150	0.000	0.863
PDP-11/60 (87% cache hit ratio)	0.170	0.140	1.578

Table 5: Top-Down Model Parameters in Microseconds

cycle counts. Since k_1 is much larger than k_2, average instruction time is more sensitive to microcycle time than to memory read pause time by a factor of k_1/k_2 or approximately 10. The implication for the designer is that much more performance can be gained or lost by perturbing the microcycle time than memory read pause time.

Although this method lacks statistical rigor, it is reasonably safe to say that memory and microcycle speed do have by far the largest impact on performance and that the dependency is quantifiable to some degree.

6.3 Measuring Second-Order Effects: Bottom-Up Approach

It is a great deal harder to measure the effect of other design tradeoffs on performance. The approximate methods employed in the previous section cannot be used because the effects being measured tend to be swamped out by first-order effects and often either cancel or reinforce one another making linear models useless. For these reasons such tradeoffs must be evaluated on a design-by-design basis as explained above. This subsection will evaluate several design tradeoffs in this way.

6.3.1 Effect of Adding a Byte Swapper to the 11/10

It is evident that the lack of a byte swapper on the PDP-11/10 has a negative effect on performance. In this subsection the performance gained by the addition of a byte swapper either before the B register or as part of the Bleg multiplexor is calculated. Adding a byte swapper would change five different parts of the instruction interpretation process: the source and destination phases where an odd-byte operand is read from memory, the execute phase where a swap byte instruction is executed in destination mode 0 and in destination modes 1 through 7, and the execute phase where an odd-byte address is modified. In each of these cases seven fast shift cycles would be eliminated and the remaining normal-speed shift cycle could be replaced by a byte swap cycle resulting in a savings of seven fast shift cycles or 1.050 usec. None of this time is overlapped with UNIBUS operations; hence, all would be saved. This savings is only effected; however, when a byte swap or odd-byte access is actually performed. The frequency with which this occurs is just the sum of the frequencies of the individual cases noted above or 0.0640. Multiplied by the time saved per occurrence gives a savings of 0.0672 usec or 1.64% of the average instruction execution time. The insignificance of this savings could well be used to support the decision for leaving the byte swapper out of the PDP-11/10.

6.3.2 Effect of Adding Processor/UNIBUS Overlap to the 11/04

Processor/UNIBUS overlap is not a feature of the 11/04 control unit. Adding this feature involves altering the control unit/UNIBUS synchronization logic so that the processor clock continues to run until a microcycle requiring the UNIBUS data from a DATI or DATIP is detected. A bus address register must also be added to drive the UNIBUS lines after the microcycle initiating the DATI/P is completed. This alteration allows time to be saved in two ways. First, processor cycles may be overlapped with memory read cycles as explained in Subsection 3.1.2. Second, since UNIBUS data is not read into the data paths during the cycle in which the DATI/P occurs, the path from the ALU through the AMUX and back to the registers is freed. This permits certain operations to be performed in the same cycle as the DATI/P, for example, the microword BA←PC; DATI; PC←PC+2 could be used to start fetching the word pointed to by the PC while simultaneously incrementing the PC to address the next word. The cycle following could then load the UNIBUS data directly into a scratchpad register rather than loading the data into the Breg and then into the scratchpad on the following cycle as is necessary without overlap logic. A savings of two microcycle times would result.

DATI and DATIP operations are scattered liberally throughout the 11/04 microcode; however, only those cycles in which an overlap would produce a time savings need be considered. An average of 0.730 cycles can be saved or overlapped during each instruction. If all of the overlapped time is actually saved, then 0.190 usec or 4.70% will be pared from the average instruction execution time. This amounts to a 4.93% increase in performance.

6.3.3 Effect of Caching on the 11/60

The PDP-11/60 uses a cache to decrease its effective memory read pause time. The degree to which this time is reduced depends upon three factors: the cache read hit pause time, the cache read miss pause time, and the ratio of cache read hits to total memory read accesses. A write-through cache is assumed; therefore, the timing of memory write accesses is not affected by caching and only read accesses need be considered. The performance of the 11/60 as measured by average instruction execution time is modeled exactly as a function of the above three parameters by the equation:

$$t = k_1 + k_2(k_3 a + k_4[1-a])$$

, where t is the average instruction execution time.
a is the cache hit ratio.
k_1 is the average execution time of a PDP-11/60 instruction excluding memory read pause time but including memory write pause time (1.339 usec).
k_2 is the number of memory reads per average instruction (1.713).
k_3 is the memory read pause time for a cache hit (0.000 usec).
k_4 is the memory read pause time for a cache miss (1.075 usec).

The above equation can be rearranged to yield:

$$t = (k_1 + k_2 k_4) - k_2(k_4 - k_3)a$$

The first term and the coefficient of the second term in the equation above evaluate to 3.181 usec and 1.842 usec respectively with the given k parameter values. This reduces the average instruction time to a function of the cache hit ratio making it possible to compare the effect of various caching schemes on 11/60 performance in terms of this one parameter.

The effect of various cache organizations on the hit ratio is described for the PDP-11 family in general in [Stre76a] and for the PDP-11/60 in particular in [Mudg77]. If no cache is provided, the hit ratio is effectively zero and the average instruction execution time reduces to the first term in the model or 3.181 usec. A set associative cache with a set size of 1 word and a cache size of 1024 words has been found through simulation to give a .87 hit ratio. An average instruction time of 1.578 usec results for a 101.52% improvement in performance over that without the cache.

The cache organization described above is that actually employed in the 11/60.

It has the virtue of being relatively simple to implement and therefore reasonably inexpensive. Set size or cache size can be increased to attain a higher hit ratio at a correspondingly higher cost. One alternative cache organization is a set size of 2 words and a cache size of 2048 words. This organization boosts the hit ratio to .93 resulting in an instruction time of 1.468 usec, an increase in performance of 7.53%. This increased performance must be paid for, however, since twice as many memory chips are needed. Because the performance increment derived from the second cache organization is much smaller than that of the first while the cost increment is approximately the same, the first organization is more cost effective.

6.3.4 Design Tradeoffs Affecting the Fetch Phase

The fetch phase holds much potential for performance improvement since it consists of a single short sequence of microoperations that, as Table 4 clearly shows, involves a sizable fraction of the average instruction time due to the inevitable memory access and possible service operations. In this subsection two approaches to cutting this time are evaluated for four different processors.

The UNIBUS interface logic of the PDP-11/04 and 11/34 are very similar. Both insert a delay into the initial microcycle of the fetch phase to allow time for bus grant arbitration circuitry to settle so that a microbranch can be taken if a serviceable condition exists. If the arbitration logic were redesigned to eliminate this delay, the average instruction execution time would drop by 0.220 usec for the 11/04 and 0.150 usec for the 11/34[6]. The resulting increases in performance would be 5.75% and 5.21% respectively.

Another example of a design feature affecting the fetch phase is the operand/instruction fetch overlap mechanism of the 11/40, 11/45, and 11/60. From the normal fetch times in the appendices and the actual average fetch times given in Table 4, the savings in fetch phase time alone can be calculated to be 0.162 usec for the 11/40, 0.087 usec for the 11/45, and 0.118 usec for the 11/60 or an increase of 7.77%, 10.07%, and 8.11% over what their respective performances would be if fetch phase time were not overlapped.

These examples demonstrate the practicality of optimizing sequences of control states that have a high frequency of occurrence rather than just those which have long durations. The 11/10 byte swap logic is quite slow, but is utilized infrequently causing its impact upon performance to be small while the bus arbitration logic of the 11/34 exacts only a small time penalty, but does so each time an instruction is executed and results in a larger performance impact. The usefulness of frequency data should thus be apparent since the bottlenecks in a design are often not where intuition says they should be.

[6] These figures are typical. Since the delay is set by an RC circuit and Schmitt trigger, the delay may vary considerably from machine to machine of a given model.

7. Summary and Use of the Methodologies

The PDP-11 offers an interesting opportunity to examine an architecture with numerous implementations spanning a wide range of price and performance. The implementations appear to fall into three distinct categories: the midrange machines (PDP-11/04/10/20/34/40/60), an inexpensive, relatively low-performance machine (LSI-11), and a comparatively expensive, but high-performance machine (PDP-11/45). The midrange machines are all minor variations on a common theme with each implementation introducing much less variability than might be expected. Their differences reside in the presence or absence of certain embellishments rather than in any major structural differences. This common design scheme is still quite recognizable in the LSI-11 and even in the PDP-11/45. The deviations of the LSI-11 arise from limitations imposed by semiconductor technology rather than directly from cost or performance considerations although the technology decision derives from cost. In the PDP-11/45, on the other hand, the quantum jump in complexity is purely motivated by the desire to squeeze the maximum performance out of the architecture.

From the overall performance model presented in Section 6.2, it is evident that instruction stream processing can be speeded up either by improving the performance of the memory subsystem or the performance of the processor. Memory subsystem performance depends upon number of memory accesses in a canonical instruction and the effective memory read pause time. There is not much that can be done about the first number since it is a function of the architecture and thus largely fixed. The second number may be improved, however, by the use of faster memory components or techniques such as caching.

Performance of the PDP-11 processor itself can be enhanced in two ways: by cutting the number of processor cycles to perform a given function or by cutting the time used per microcycle. Several approaches to decreasing the effective microcycle count have been demonstrated:

1) *Structure the data paths for maximum parallelism* - The PDP-11/45 can perform much more in a given microcycle than any of the midrange PDP-11s and thus needs fewer microcycles to complete an instruction. To obtain this increased functionality, however, a much more elaborate set of data paths is required in addition to a highly developed control unit to excercise them to maximum potential. Such a change is not an incremental one and involves rethinking the entire implementation.

2) *Structure the microcode to take best advantage of instruction features* - All processors except the 11/10 handle JMP/JSR addressing modes as a special case in the microcode. Most do the same for the destination modes of the MOV instruction because of its high frequency. Varying degrees of sophistication in instruction dispatching from the BUT IRDECODE at the end of every fetch is evident in different machines resulting in various performance improvements.

3) *Cut effective microcycle count by overlapping processor and UNIBUS*

operation - The PDP-11/10 demonstrates that a large microcycle count can be effectively reduced by placing cycles in parallel with memory access operations whenever possible.

Increasing microcycle speed is perhaps more generally useful since it can often be applied without making substantial changes to an entire implementation. Several of the midrange PDP-11s achieve most of their performance improvement by increasing microcycle speed in the following ways:

1) *Make the data paths faster* - The PDP-11/34 demonstrates the improvement in microcycle time that can result from the judicious use of Schottky TTL in such heavily travelled points as the ALU. Replacing the ALU and carry-lookahead logic alone with Schottky equivalents saves approximately 35 nanoseconds in propagation delay. With cycle times running 300 nanoseconds and less, this amounts to better than a 10% increase in speed.

2) *Make each microcycle take only as long as necessary* - The 11/34 and 11/40 both use selectable microcycle times to speed up cycles which don't entail long data path propagation delays.

Circuit technology is perhaps the single most important factor in performance. It is only stating the obvious to say that doubling circuit speed will double total performance. Aside from raw speed, circuit technology dictates what it is economically feasible to build as witnessed by the SSI PDP-11/20, the MSI PDP-11/40, and the LSI-11. Just the limitations of a particular circuit technology at a given point in time may dictate much about the design tradeoffs that can be made as in the case of the LSI-11.

Turning to the methodologies, the two presented in Section 6 can be used at various times during the design cycle. The top-down approach can be used to estimate the performance of a proposed implementation, or to plan a family of implementations, given only the characteristics of the selected technology and a general estimate of data path and memory cycle utilization.

The bottom-up approach can be used to perturb an existing or planned design to determine the performance payoff of a particular design tradeoff. The relative frequencies of each function (e.g. addressing modes, instructions, etc.), while required for an accurate prediction, may not be available. There are, however, alternative ways to estimate relative frequencies. Consider the three following situations:

1) *At least one implementation exists* - An analysis of the implementation in typical usage (i.e. benchmark programs for a stored-program computer) can provide the relative frequencies.

2) *No implementation exists, but similar systems exist* - The frequency data may be extrapolated from measurements made on a machine with a similar architecture. For example, the Gibson Mix [Bell71] provided the relative frequencies of IBM 7090 functions from which the relative frequencies of IBM 360 functions were estimated.

3) *No implementation exists and there are no prior similar systems* - From knowledge of the specifications, a set of most-used functions can be estimated (e.g. instruction fetch, register and relative addressing, move and add instructions for a stored-program computer). The design is then optimized for these functions.

Of course, the relative frequency data should always be updated to take into account new data.

Our purpose in writing this paper has been two-fold: to provide data about design tradeoffs and to suggest design methodologies based on this data. It is hoped that the design data will stimulate the study of other methodologies while the results of the design methodologies presented here have demonstrated their usefulness to designers.

Figure 5: Archetypical Medium-Range PDP-11 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.

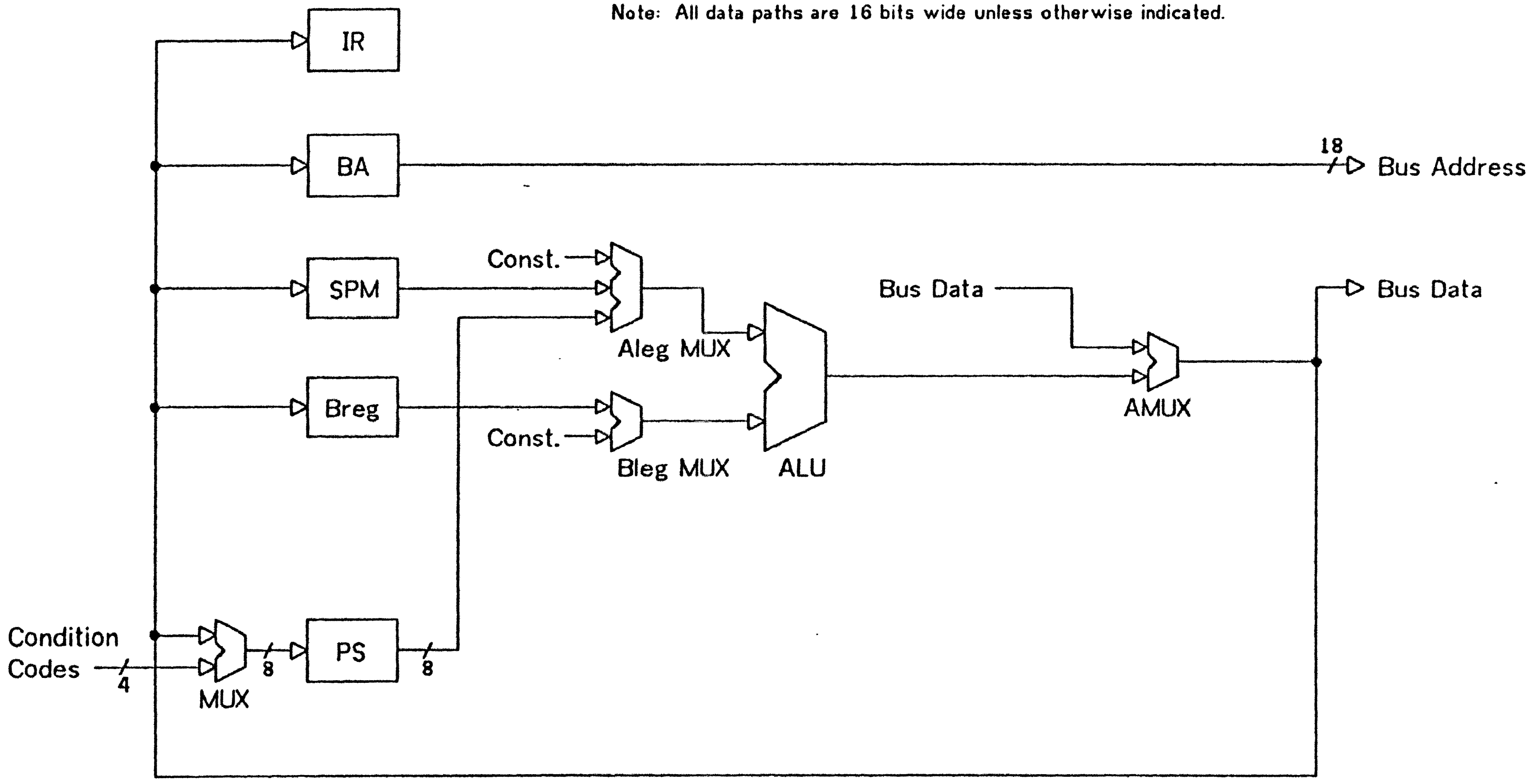


Figure 6: Archetypical Microprogrammed PDP-11 Control Unit

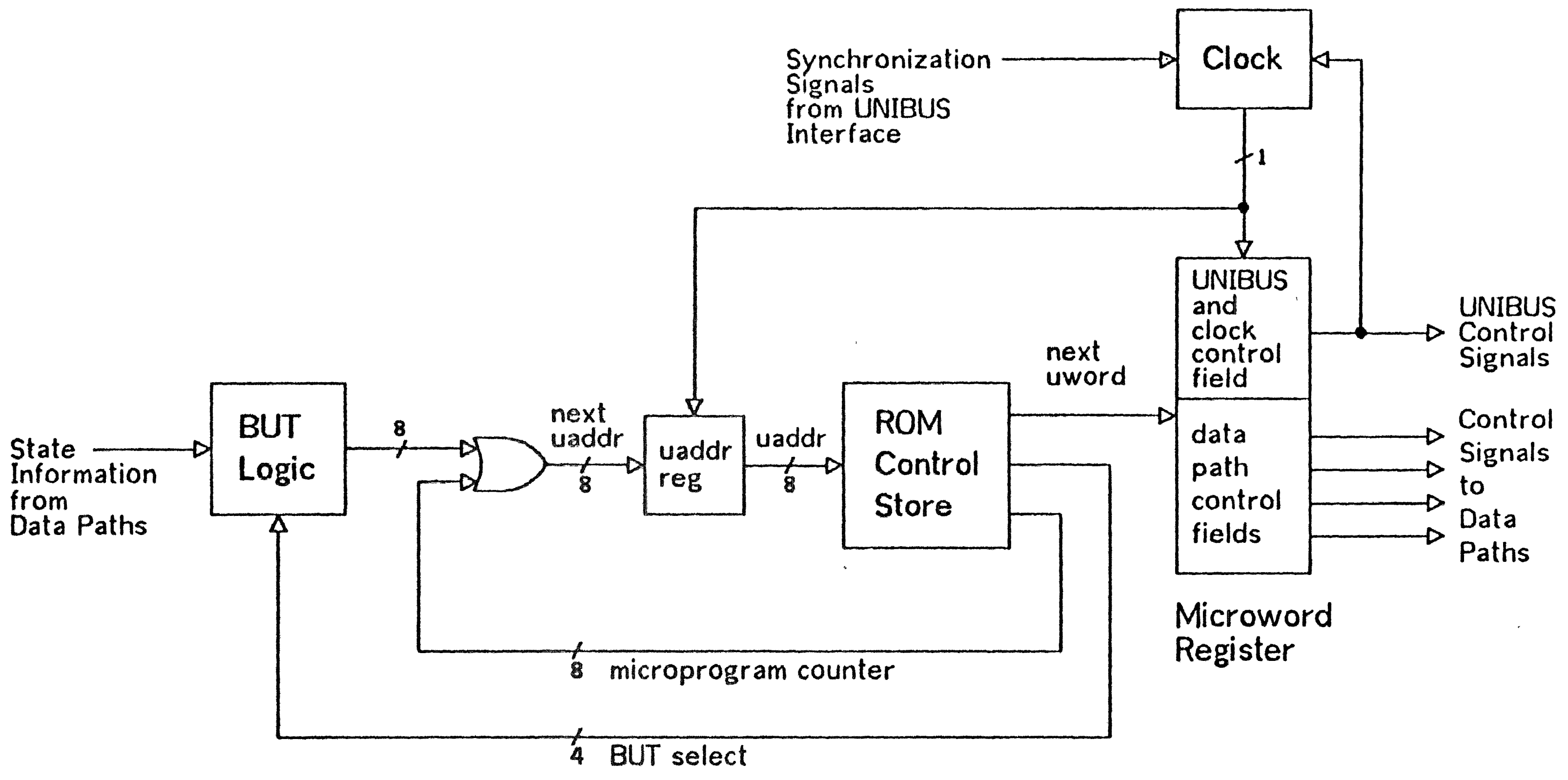


Figure 7: PDP-11/20 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.

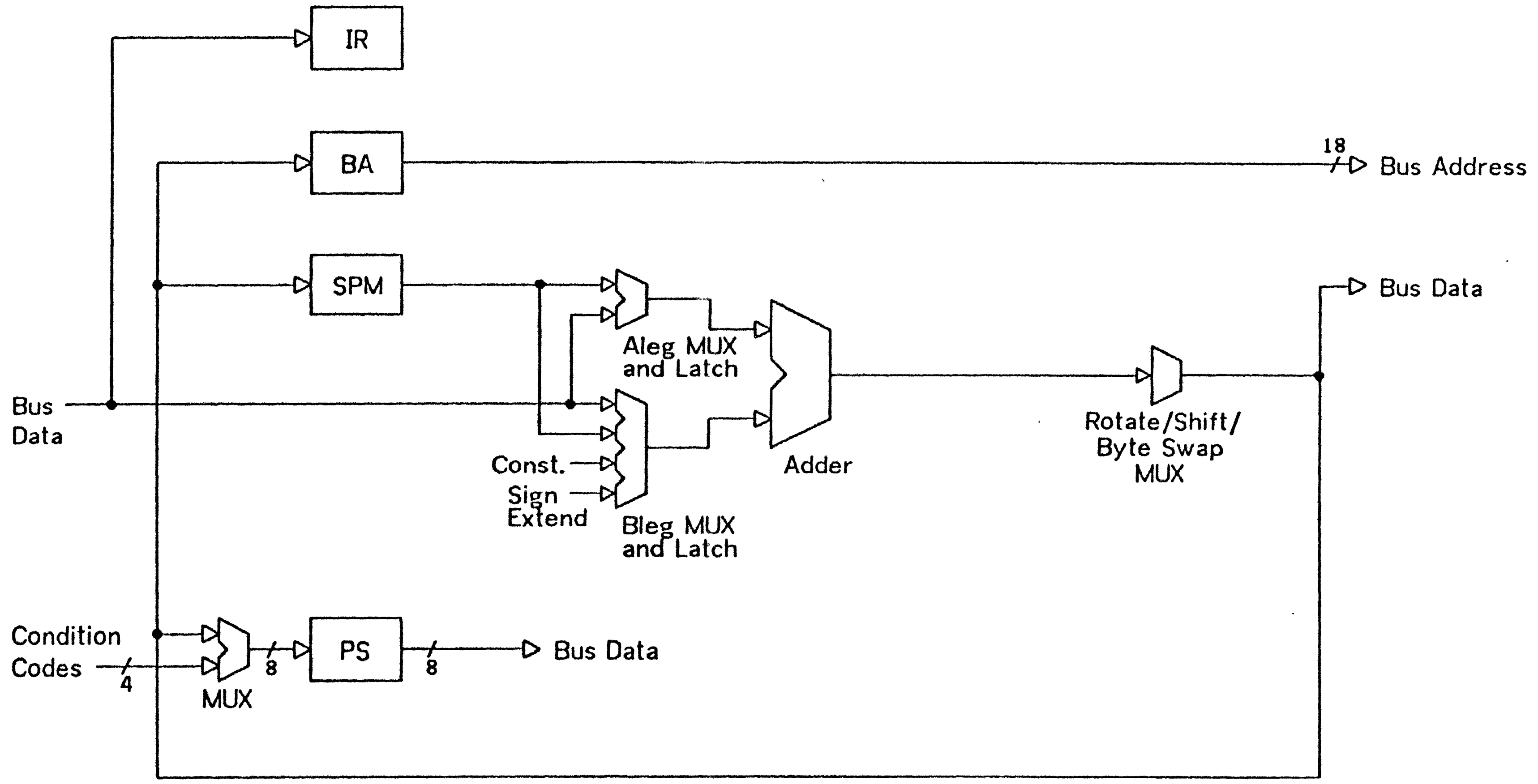
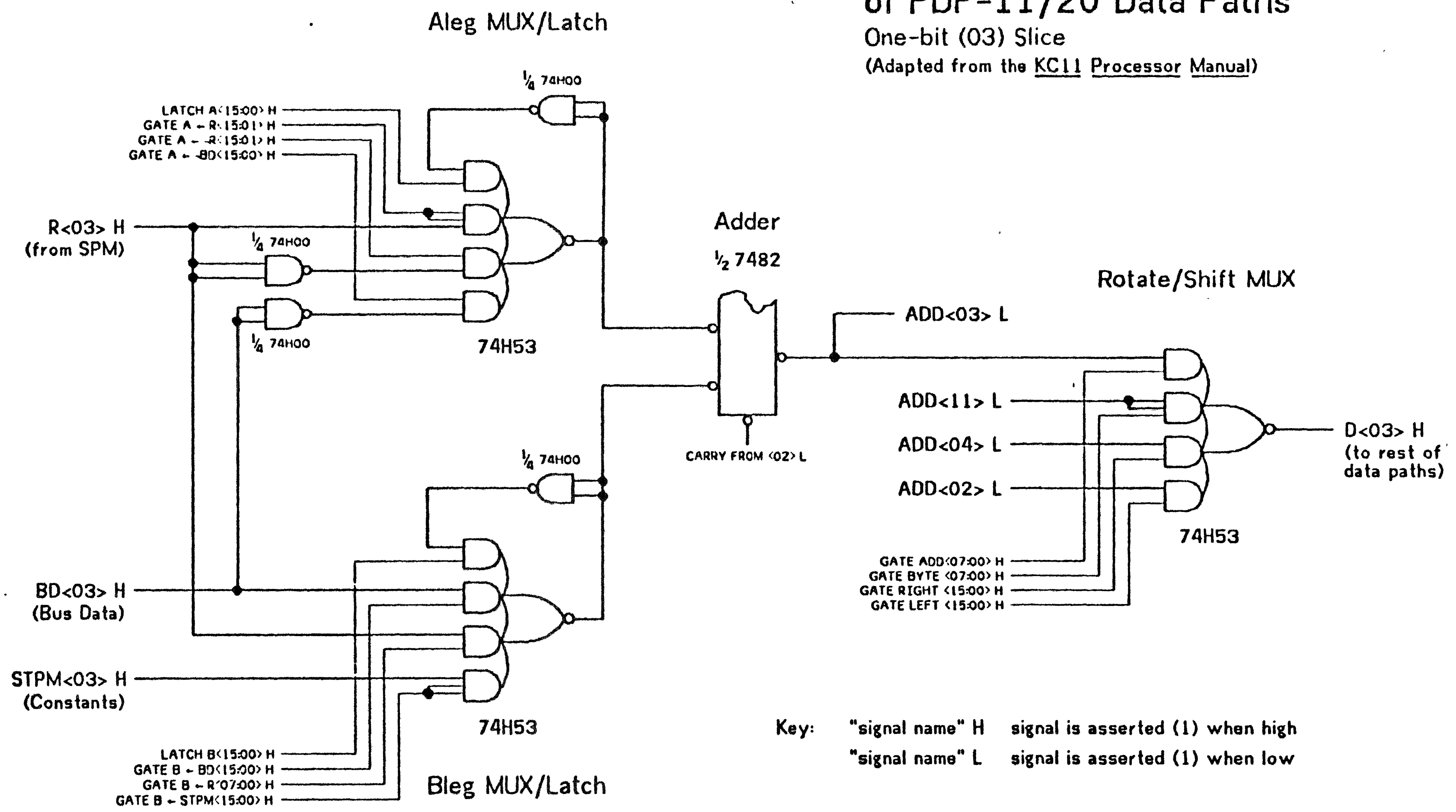


Figure 8: Detail of Central Part of PDP-11/20 Data Paths

One-bit (03) Slice

(Adapted from the KC11 Processor Manual)



Key:	"signal name" H	signal is asserted (1) when high
	"signal name" L	signal is asserted (1) when low

Figure 9: PDP-11/40 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.

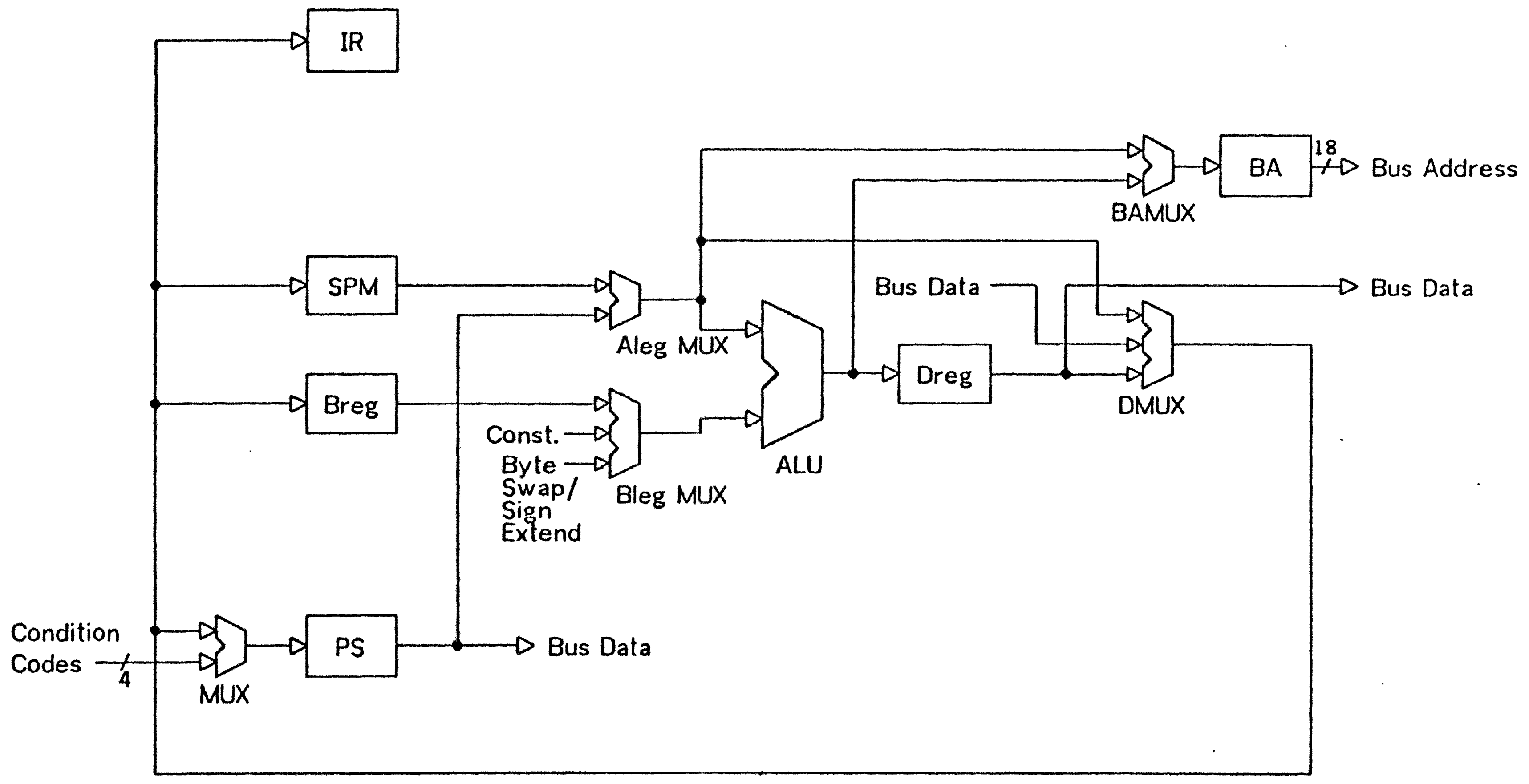


Figure 10: PDP-11/10 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.

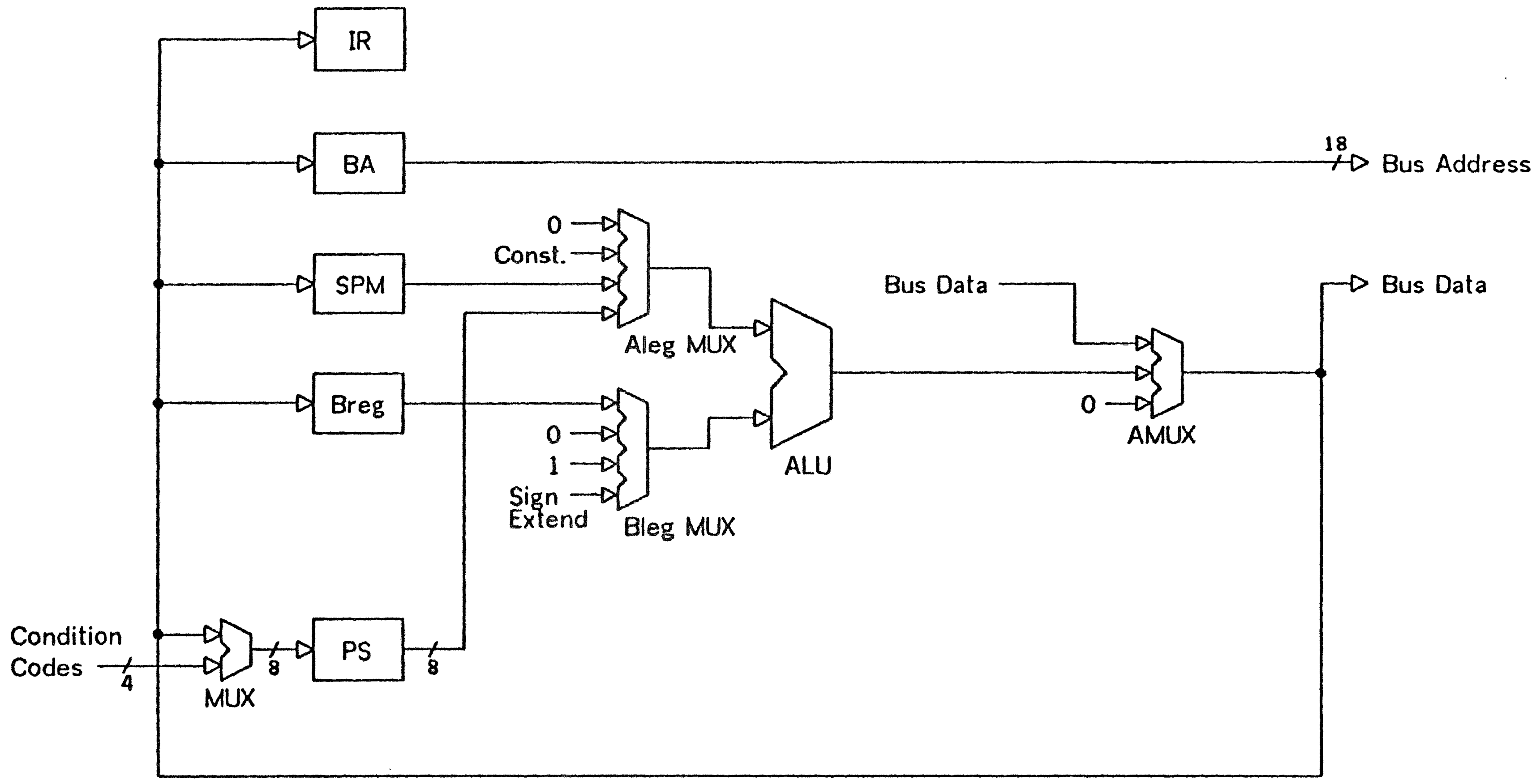


Figure 11: PDP-11/04 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.

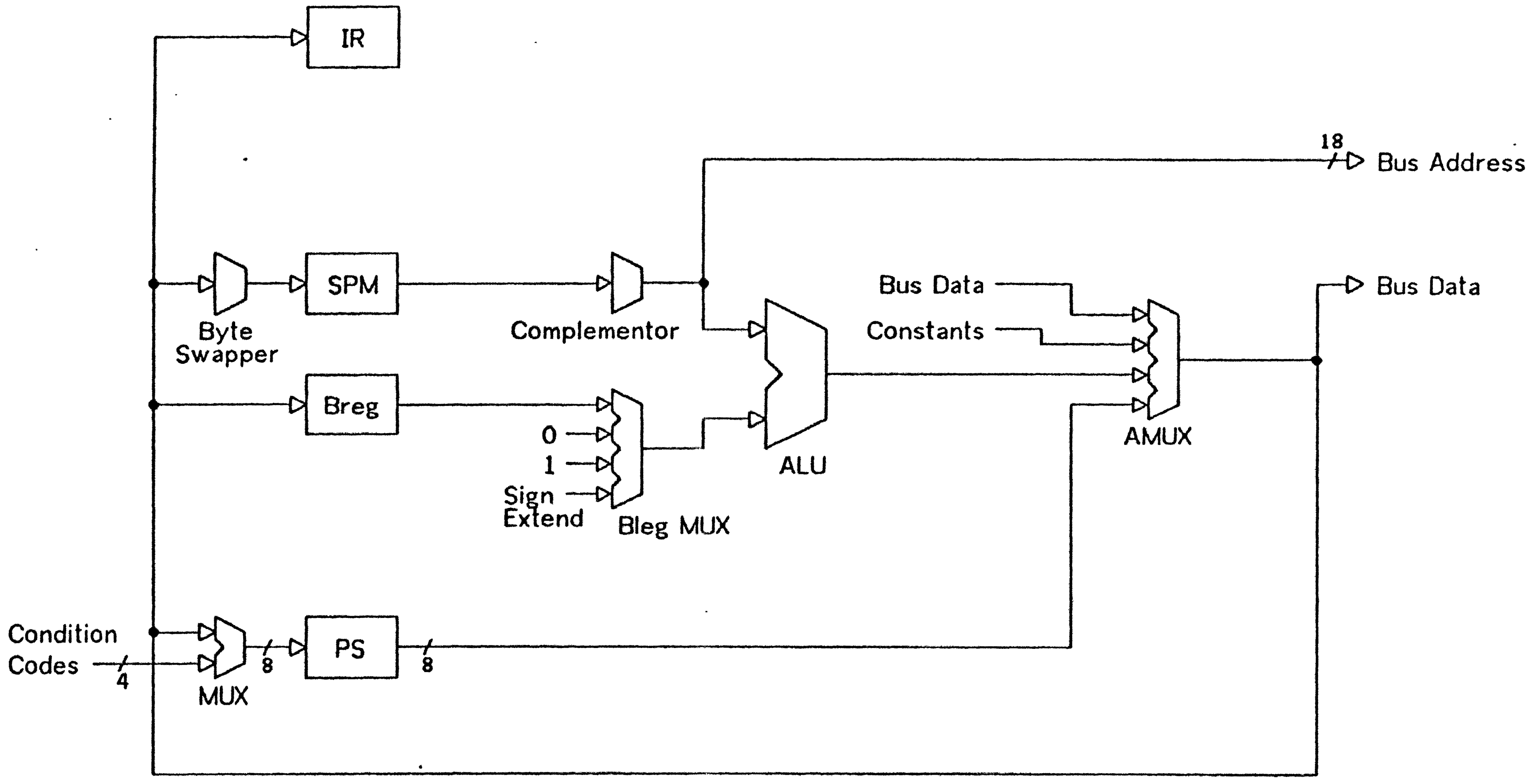


Figure 12: PDP-11/34 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.

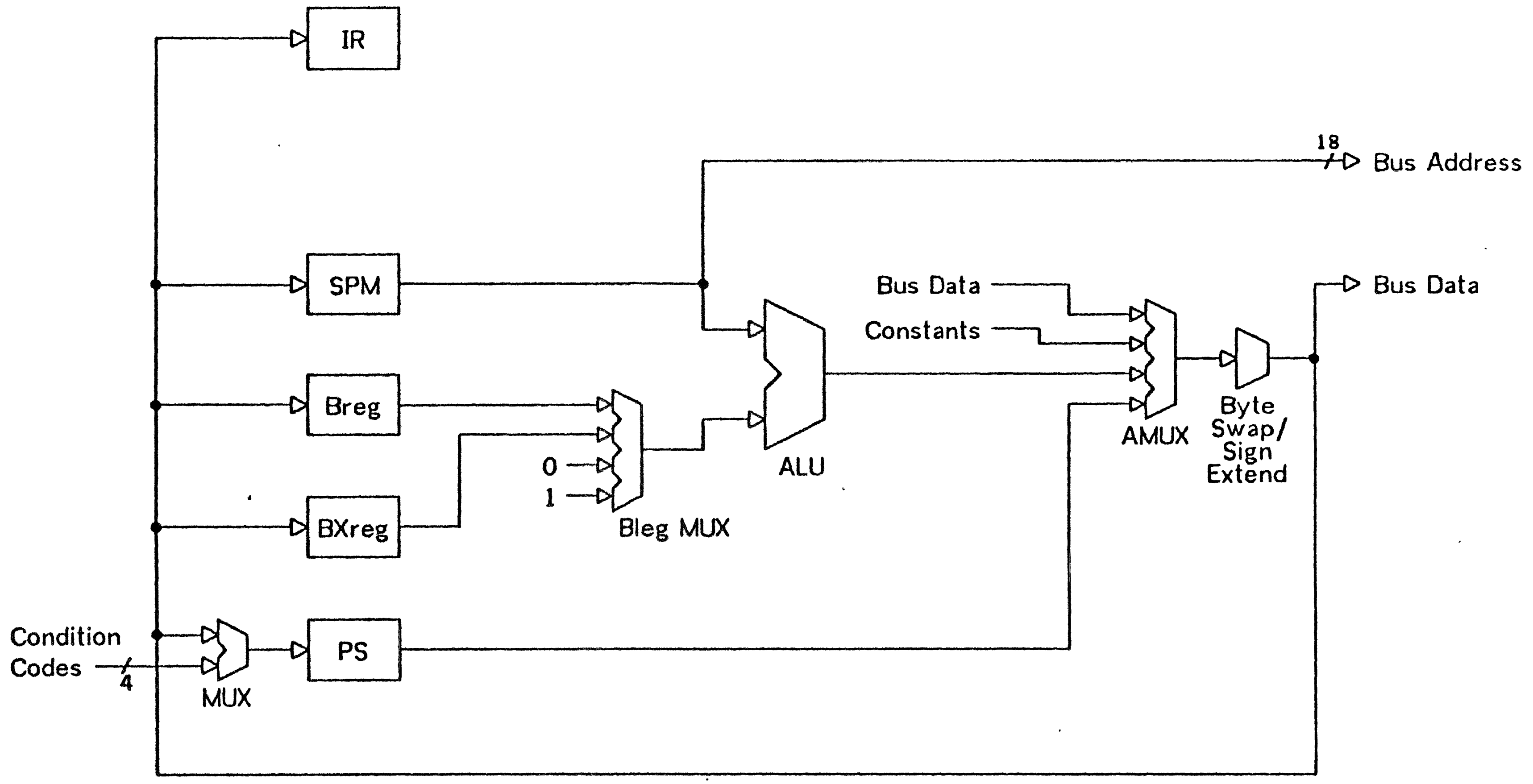


Figure 13: PDP-11/60 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.

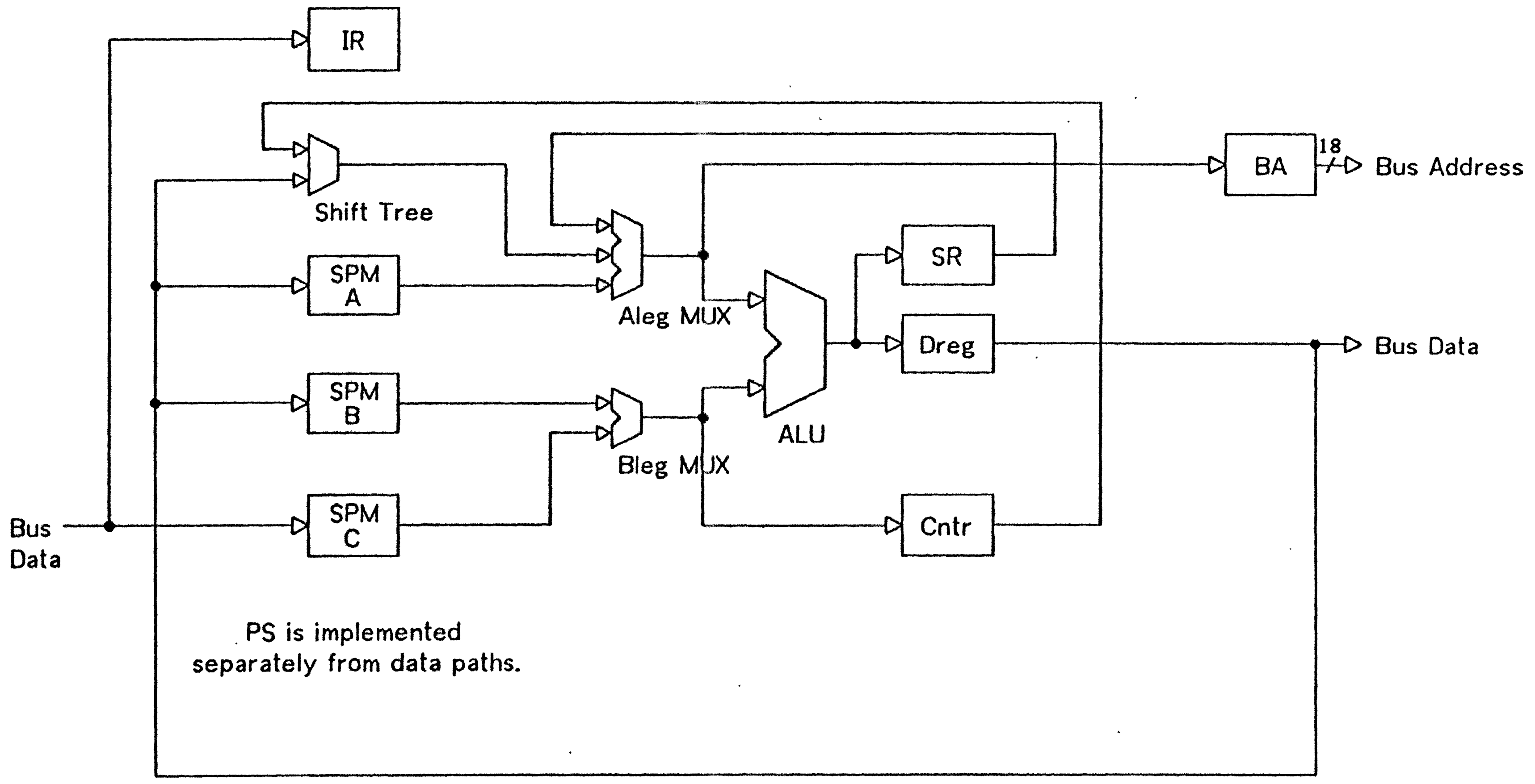


Figure 14: LSI-11 Data Paths

Note: All data paths are 8 bits wide unless otherwise indicated.

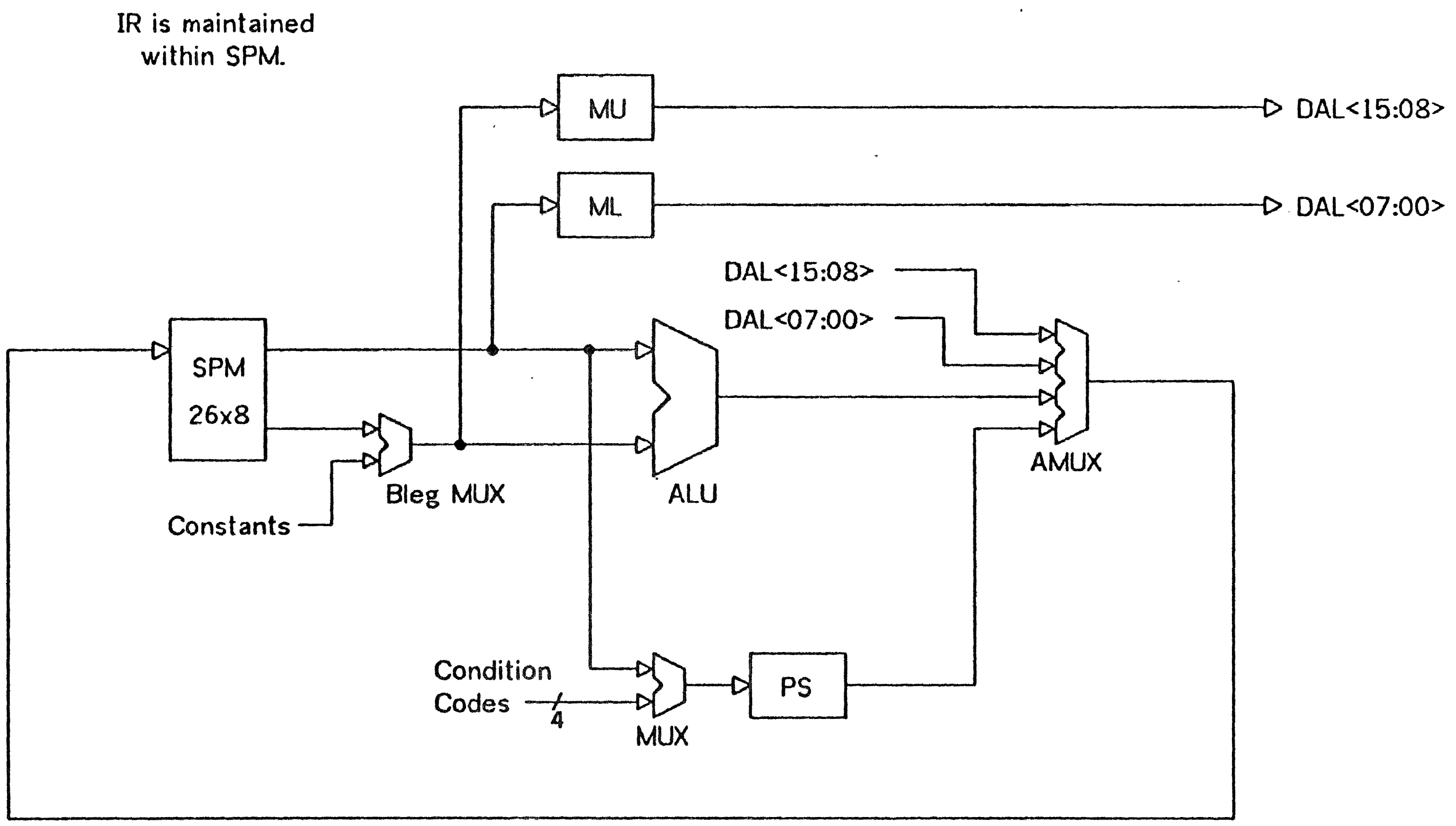
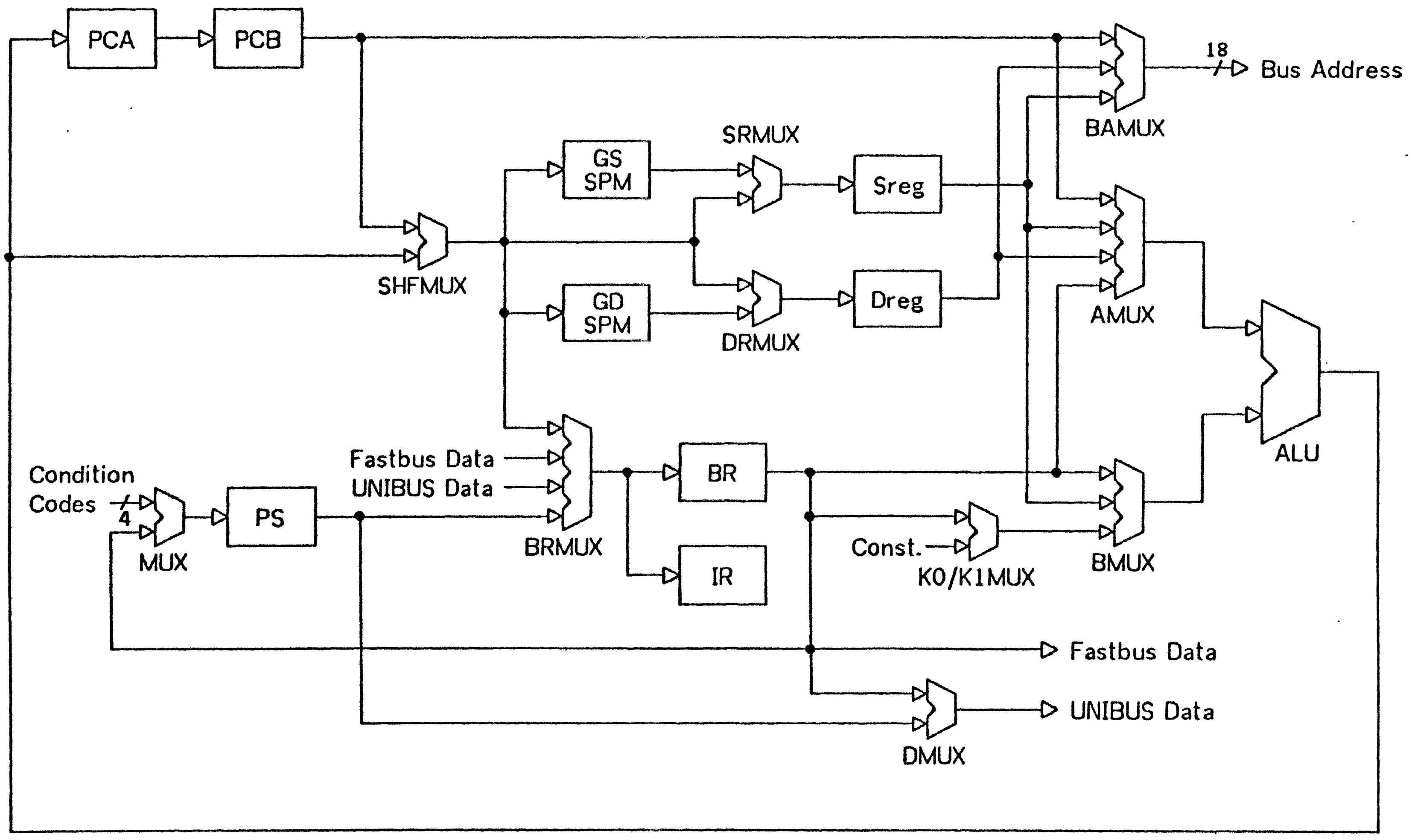


Figure 15: PDP-11/45 Data Paths

Note: All data paths are 16 bits wide unless otherwise indicated.



Introduction to the Appendices

Appendix A tabulates the frequencies of PDP-11 instructions and addressing modes. This data was derived as explained in Subsection 6.1. Frequencies are given for the occurrence of each phase (e.g. source, which occurs only during double-operand instructions), each subcase of each phase (e.g. jump destination, which occurs only during jump or jump to subroutine instructions), and each instance of each phase such as a particular addressing mode or instruction. The frequency with which the phase is skipped is listed for source and destination phases. Source and destination odd-byte-addressing frequencies are listed as well due to their effect on instruction timing.

Appendices B through I tabulate the calculated instruction execution times for all the PDP-11 processors reviewed here. These calculations have been made assuming certain processor and memory timing characteristics described at the end of each appendix. Normal timing variations from machine to machine can be significant; therefore, the times given here can only be taken as typical.

Appendix A: Instruction Time Component Frequencies

	Frequency
Fetch	1.0000
Source Mode	0.4069
0 R	0.1377
1 @R or (R)	0.0338
2 (R)+	0.1587
3 @(R)+	0.0122
4 -(R)	0.0352
5 @-(R)	0.0000
6 X(R)	0.0271
7 @X(R)	0.0022
No Source	0.5931
NOTE: Frequency of odd-byte addressing (SM1-7) = 0.0252.	
Destination	0.6872
Data Manipulation Mode	0.6355
0 R	0.3146
1 @R or R	0.0599
2 (R)+	0.0854
3 @(R)+	0.0307
4 -(R)	0.0823
5 @-(R)	0.0000
6 X(R)	0.0547
7 @X(R)	0.0080
NOTE: Frequency of odd-byte addressing (DM1-7) = 0.0213.	
Jump (JMP/JSR) Mode	0.0517
0 R	0.0000 (ILLEGAL)
1 @R or (R)	0.0000
2 (R)+	0.0000
3 @(R)+	0.0079
4 -(R)	0.0000
5 @-(R)	0.0000
6 X(R)	0.0438
7 @X(R)	0.0000
No Destination	0.3128

	Frequency
Execute Instruction	1.0000
Double Operand	0.4069
ADD	0.0524
SUB	0.0274
BIC	0.0309
BICB	0.
BIS	0.0012
BISB	0.0013
CMP	0.0626
CMPB	0.0212
BIT	0.0041
BITB	0.0014
MOV	0.1517
MOVB	0.0524
XOR	0.
Single Operand	0.2286
CLR	0.0186
CLRB	0.0018
COM	0.
COMB	0.
INC	0.0224
INCB	0.
DEC	0.0809
DECB	0.
NEG	0.0038
NEGB	0.
ADC	0.0070
ADCB	0.
SBC	0.
SBCB	0.
ROR	0.0036
RORB	0.
ROL	0.0059
ROLB	0.
ASR	0.0069
ASRB	0.
ASL	0.0298
ASLB	0.
TST	0.0329
TSTB	0.0079
SWAB	0.0038
SXT	0.

	Frequency
Branch	0.2853
All Branches (true)	0.1744
All Branches (false)	0.1109
SOB (true)	0.
SOB (false)	0.
Jump	0.0517
JMP	0.0272
JSR	0.0245
Control, Trap, and Miscellaneous	0.0270
Set/Clear Condition Codes	0.0017
MARK	0.
RTS	0.0236
RTI	0.
RTT	0.
IOT	0.
EMT	0.0017
TRAP	0.
BPT	0.

NOTES:
Frequency of destination odd-byte addressing (DM1-7) = 0.0213.
Execution frequencies indicated as 0. have an aggregate frequency < 0.0050.

Appendix B: LSI-11 Instruction Execution Times

Microcycle Time = 0.400 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)	Notes
Fetch Time	1	5	2.400	

NOTE:
The following instructions take additional ucycles to decode:
All single-operand instructions except SWAB, SXT, MFPS, and MTPS add 1 ucycle (+0.400 usec).
XOR, JMP, RTS, RTI, RTT, set/clear condition codes add 1 ucycle (+0.400 usec).
SWAB adds 2 ucycles (+0.800 usec).
MFPS, MTPS add 4 ucycles (+1.600 usec).
SXT adds 5 ucycles (+2.000 usec).
BPT, IOT add 6 ucycles (+2.400 usec).
MARK adds 8 ucycles (+3.200 usec).

Source Times

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	1	0.400	
1 @R or (R)	1	3	1.600	(1)
2 (R)+	1	4	2.000	(3)
3 @(R)+	2	7	3.600	(1)
4 -(R)	1	5	2.400	(2)
5 @-(R)	2	8	4.000	(1)
6 X(R)	2	9	4.400	(1)
7 @X(R)	3	12	6.000	(1)

NOTES:
(1) Byte addressing subtracts 1 ucycle (-0.400 usec).
(2) Byte addressing adds 1 ucycle (+0.400 usec).
(3) If register ≠ R6 or R7, byte addressing adds 1 ucycle (+0.400 usec).

Destination Times

Data Manipulation

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	1	0.400	
1 @R or (R)	1	4	2.000	
2 (R)+	1	5	2.400	(1)
3 @(R)+	2	8	4.000	
4 -(R)	1	6	2.800	(1)
5 @-(R)	2	9	4.400	
6 X(R)	2	10	4.800	
7 @X(R)	3	13	6.400	

NOTES:
For MOV: DM0 subtracts 1 ucycle (-0.400 usec). DM1-7 subtracts 2 ucycles and memory read (-1.200 usec).
Byte addressing (DM1-7) subtracts 1 ucycle (-0.400 usec).
(1) If register = R6 or R7, byte addressing adds 2 ucycles (+0.800 usec) additive to the time noted directly above.

	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes

Jump (JMP/JSR)					
Mode					
0 R	ILLEGAL				
1 @R or (R)	0		3	1.200	
2 (R)+	0		5	2.000	
3 @(R)+	1		5	2.400	
4 -(R)	0		5	2.000	
5 @-(R)	1		6	2.800	
6 X(R)	1		7	3.200	
7 @X(R)	2		10	4.800	

Execute Times					
Instruction					

Double Operand					
ADD, SUB, BIC, BIS, XOR		1	3	1.600	(3)
BICB, BISB		1	2	1.200	(3)
CMP, BIT		0	2	0.800	
CMPB, BITB		0	1	0.400	
MOV		1	3	1.600	(2)
MOVB		1	2	1.200	(1)

Single Operand					
CLR		1	3	1.600	(2)
CLRB		1	3	1.600	(2)
COM, NEG		1	4	2.000	(2)
COMB, NEGB		1	3	1.600	(2)
INC, DEC, ADC, SBC		1	5	2.400	(3)
INCB, DECB, ADCB, SBCB		1	4	2.000	(3)
ROR		1	8	3.600	(3)
RORB		1	5	2.400	(3)
ASR		1	9	4.000	(3)
ASRB		1	8	3.600	(4)
ROL, ASL		1	4	2.000	(3)
ROLB, ASLB		1	3	1.600	(3)
TST		0	4	1.600	
TSTB		0	3	1.200	
SWAB		1	3	1.600	(2)
SXT		1	6	2.800	(3)
MFPS		1	8	3.600	(1,9)
MTPS		1	10	4.400	(2,5,9,10)

	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes

Branch					
All Branches (true)			4	1.600	
All Branches (false)			4	1.600	
SOB (true)			8	3.200	
SOB (false)			6	2.400	

Jump					
JMP			2	0.800	
JSR (register = R7)		1	6	2.800	
JSR (register ≠ R7)		1	15	6.400	

Control, Trap, and Miscellaneous					
Set/Clear Condition Codes			3	1.200	
MARK	1		16	6.800	
RTS	1		6	2.800	
RTI	2		15	6.800	(5,6)
RTT	2		15	6.800	(5,7)
IOT, EMT, TRAP, BPT	2	2	33	14.800	(5,8)

NOTES:
(1) DMO adds 1 ucycle and subtracts memory write (+0.000 usec).
(2) DMO subtracts memory write (-0.400 usec).
(3) DMO subtracts 1 ucycle and memory write (-0.800 usec).
(4) DMO subtracts 3 ucycles and memory write (-1.600 usec).
(5) If new PS has bit 7 clear, add 1 ucycle (+0.400 usec).
(6) If new PS has bit 4 set, add 9 ucycles (+3.600 usec).
(7) If new PS has bit 4 set, add 10 ucycles (+4.000 usec).
(8) If new PS has bit 4 set, add 1 ucycle (+0.400 usec).
(9) Byte instruction.
(10) Use destination rather than source times.

============================

NOTE:
Times given apply to microcode revision 2(4), MICROMs CP1631-10 (DEC 23-088A5) and CP1631-07 (DEC 23-087A5).

Times Assumed for All Calculations
----
1) Microcycle time is 0.400 usec.
2) Microcycle time is extended by 0.400 usec during DATI/DATIP/DATO/DATOB. (Note: 1 extra wait ucycle is actually generated for each memory access; however, these ucycles have not been tallied in the microcycle counts above.)

Appendix C: PDP-11/04 Instruction Execution Times

Microcycle Time = 0.260 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)
Fetch Time	1	3	1.940
Source Times			
Mode			
0 R	0	2	0.520
1 @R or (R)	1	2	1.460
2 (R)+	1	3	1.720
3 @(R)+	2	5	3.180
4 -(R)	1	3	1.720
5 @-(R)	2	5	3.180
6 X(R)	2	6	3.440
7 @X(R)	3	8	4.900

NOTE:
Odd-byte addressing (SM1-7) adds 2 ucycles (+0.520 usec).

Destination Times

Data Manipulation

Mode	Memory Reads	Micro-Cycles	Time (usec)
0 R	0	1	0.260
1 @R or (R)	1	1	1.200
2 (R)+	1	2	1.460
3 @(R)+	2	4	2.920
4 -(R)	1	2	1.460
5 @-(R)	2	4	2.920
6 X(R)	2	5	3.180
7 @X(R)	3	7	4.640

NOTE:
Odd-byte addressing (DM1-7) adds 2 ucycles (+0.520 usec).

Jump (JMP/JSR)

Mode	Memory Reads	Micro-Cycles	Time (usec)
0 R	ILLEGAL		
1 @R or (R)	0	2	0.520
2 (R)+	0	3	0.780
3 @(R)+	1	3	1.720
4 -(R)	0	3	0.780
5 @-(R)	1	3	1.720
6 X(R)	1	4	1.980
7 @X(R)	2	6	3.440

	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes
Execute Times					
Instruction					
Double Operand					
ADD, SUB, BIC(B), BIS(B)		1	2	1.060	(1)
CMP(B), BIT(B)		0	1	0.260	
MOV(B)		1	2	1.060	(1,2)
Single Operand					
CLR(B), COM(B), INC(B), DEC(B), NEG(B), ADC(B), SBC(B)		1	2	1.060	(1)
ROR(B), ROL(B), ASR(B), ASL(B)		1	3	1.320	(1)
TST(B)		0	1	0.260	
SWAB		1	3	1.320	(1)
Branch					
All Branches (true)			3	0.780	
All Branches (false)			0	0.000	
Jump					
JMP			0	0.000	
JSR		1	7	2.360	
Control, Trap, and Miscellaneous					
Set/Clear Condition Codes			2	0.520	
RTS	1		5	2.240	
RTI	2		6	3.440	
IOT, EMT, TRAP, BPT	2	2	12	6.080	

NOTES:
(1) Destination odd-byte addressing (DM1-7) adds 2 ucycles (+0.520 usec). DM0 subtracts memory write (-0.540 usec).
(2) DM0 subtracts 1 additional ucycle (-0.260 usec).

Times Assumed for All Calculations

1) Microcycle time is 0.260 usec.
2) Microcycle time is extended by 0.220 usec by bus priority arbitration delay during BUT SERVICE.
3) Microcycle time is extended by 0.940 usec during DATI/DATIP (MOS memory).
4) Microcycle time is extended by 0.540 usec during DATO/DATOB (MOS memory).

Appendix D: PDP-11/10 Instruction Execution Times

Microcycle Time = 0.300 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)	
Fetch Time	1	5	1.500	
Source Times				
Mode				
0 R	0	2	0.600	
1 @R or (R)	1	3	1.500	
2 (R)+	1	5	1.500	
3 @(R)+	2	7	2.700	
4 -(R)	1	4	1.500	
5 @-(R)	2	6	2.700	
6 X(R)	2	7	2.700	
7 @X(R)	3	9	3.900	

NOTE:
Odd-byte addressing (SM1-7) adds 7 fast shift (0.150 usec/ucycle) and 1 regular ucycle for a total of +1.350 usec.

Destination Times

Data Manipulation

	Memory Reads	Micro-Cycles	Time (usec)	
Mode				
0 R	0	2	0.600	(1)
1 @R or (R)	1	3	1.500	
2 (R)+	1	5	1.500	
3 @(R)+	2	7	2.700	
4 -(R)	1	4	1.500	
5 @-(R)	2	6	2.700	
6 X(R)	2	7	2.700	
7 @X(R)	3	9	3.900	

NOTE:
Odd-byte addressing (DM1-7) adds 7 fast shift (0.150 usec/ucycle) and 1 regular ucycle for a total of +1.350 usec.
(1) MOV subtracts 1 ucycle (-0.300 usec).

Jump (JMP/JSR)

	Memory Reads	Micro-Cycles	Time (usec)
Mode			
0 R	ILLEGAL		
1 @R or (R)	1	1	0.900
2 (R)+	1	3	0.900
3 @(R)+	2	5	2.100
4 -(R)	1	2	0.900
5 @-(R)	2	4	2.100
6 X(R)	2	5	2.100
7 @X(R)	3	7	3.300

Execute Times Instruction	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes
Double Operand					
ADD, SUB, BIC(B), BIS(B)		1	4	1.800	(1)
CMP(B), BIT(B)		0	2	0.600	
MOV(B)		1	4	1.800	(1)
Single Operand					
CLR(B), COM(B), INC(B), DEC(B), NEG(B), ADC(B), SBC(B), ROR(B), ROL(B), ASR(B), ASL(B)		1	5	2.100	(1)
TST(B)		0	3	0.900	
SWAB		1	12	3.150	(1,2)
Branch					
All Branches (true)			3	0.900	
All Branches (false)			1	0.300	
Jump					
JMP			2	0.600	
JSR		1	9	3.300	
Control, Trap, and Miscellaneous					
Set/Clear Condition Codes			3	0.900	
RTS	1		7	2.100	
RTI	2		9	2.700	
IOT, EMT, TRAP, BPT	2	2	13	6.300	

NOTES:

(1) Destination odd-byte addressing (DM1-7) adds 7 fast shift ucycles (0.150 usec/ucycle) for a total of +1.050 usec. DM0 subtracts 2 ucycles and memory write (-1.200 usec).

(2) Byte swap consists of 7 fast shift (0.150 usec/ucycle) and 1 regular ucycle for a total of +1.350 usec.

NOTE:

Times given apply to the M7261 microprogram module, revision R. Earlier versions use additional ucycles.

Times Assumed for All Calculations

1) Microcycle time is 0.300 usec.
2) A CKOFF following a DATI/DATIP/DATO/DATOB extends ucycle time by 0.600 usec minus 0.300 usec for each ucycle that the CKOFF is removed from the cycle initiating the bus transaction.

Appendix E: PDP-11/20 Instruction Execution Times

Microcycle Time = 0.280 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)
Fetch Time	1	4	1.490
Source Times			
Mode			
0 R	0	0	0.000
1 @R or (R)	1	4	1.490
2 (R)+	1	4	1.490
3 @(R)+	2	7	2.700
4 -(R)	1	4	1.490
5 @-(R)	2	7	2.700
6 X(R)	2	7	2.700
7 @X(R)	3	10	3.910

NOTE:
Odd-byte addressing (SM1-7) adds 2 ucycles (+0.560 usec).

Destination Times

Data Manipulation

	Memory Reads	Micro-Cycles	Time (usec)
Mode			
0 R	0	1	0.280
1 @R or (R)	1	4	1.390
2 (R)+	1	4	1.390
3 @(R)+	2	7	2.600
4 -(R)	1	4	1.390
5 @-(R)	2	7	2.600
6 X(R)	2	7	2.600
7 @X(R)	3	10	3.810

NOTES:
Odd-byte addressing (DM1-7) adds 2 ucycles (+0.560 usec).
Non-modifying instruction (CMP(B), BIT(B), TST(B)) adds 0 ucycles (+0.100 usec for DATI in place of DATIP).

Jump (JMP/JSR)

	Memory Reads	Micro-Cycles	Time (usec)
Mode			
0 R	ILLEGAL		
1 @R or (R)	0	4	1.120
2 (R)+	0	4	1.120
3 @(R)+	1	7	2.330
4 -(R)	0	4	1.120
5 @-(R)	1	7	2.330
6 X(R)	1	7	2.330
7 @X(R)	2	10	3.540

Execute Times Instruction	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes
Double Operand					
ADD, SUB, BIS(B), MOV(B)		1	3	0.840	(1)
BIC(B)		1	5	1.400	(1)
BIT(B)		0	4	1.120	
CMP(B)		0	2	0.560	
Single Operand					
CLR(B), COM(B), INC(B), DEC(B),.NEG(B), ADC(B), SBC(B)		1	3	0.840	(1)
ROR(B), ROL(B), ASR(B), ASL(B)		1	3	0.840	(1,2)
TST(B)		0	2	0.560	
SWAB		1	3	0.840	(1)
Branch					
All Branches (true)			4	1.120	
All Branches (false)			0	0.000	
Jump					
JMP			0	0.000	
JSR		1	10	2.800	
Control, Trap, and Miscellaneous					
Set/Clear Condition Codes			0	0.000	
RTS	1		6	2.050	
RTI	2		9	3.260	
IOT, EMT, TRAP, BPT	2	2	21	6.620	

NOTES:
(1) DM0 subtracts 1 ucycle and memory write (-0.280 usec). PS as destination adds 1 ucycle (+0.280 usec).
(2) Odd-byte addressing (DM1-7) adds 2 ucycles (+0.560 usec).

Times Assumed for All Calculations

1) Microcycle time is 0.280 usec.
2) Microcycle time is extended by 0.370 usec during DATI.
3) Microcycle time is extended by 0.270 usec during DATIP.
4) Microcycle time is extended by 0.000 usec during DATO/DATOB.

Appendix F: PDP-11/34 Instruction Execution Times

Microcycle Time = 0.180/0.240 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)	Notes
Fetch Time	1	3	1.630	

Source Times

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	1	0.180	(1)
1 @R or (R)	1	1	1.120	
2 (R)+	1	2	1.300	
3 @(R)+	2	3	2.420	
4 -(R)	1	2	1.300	
5 @-(R)	2	3	2.420	
6 X(R)	2	4	2.600	
7 @X(R)	3	5	3.720	

NOTE:
(1) DM0 subtracts 1 ucycle (-0.180 usec).

Destinations Times

Data Manipulation

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	1	0.180	(1,2)
1 @R or (R)	1	1	1.120	
2 (R)+	1	2	1.300	(1)
3 @(R)+	2	3	2.420	
4 -(R)	1	2	1.300	
5 @-(R)	2	3	2.420	
6 X(R)	2	4	2.600	
7 @X(R)	3	5	3.720	

NOTES:
MOV(B) and DM1-7 changes long to short ucycle and subtracts memory read (-1.000 usec).
(1) MOV(B) subtracts an additional ucycle (-0.180 usec).
(2) Single-operand instruction except NEG(B) subtracts 1 ucycle (-0.180 usec).

Jump (JMP/JSR)

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	ILLEGAL			
1 @R or (R)	0	0	0.000	(1)
2 (R)+	0	2	0.360	
3 @(R)+	1	2	1.300	
4 -(R)	0	1	0.180	
5 @-(R)	1	2	1.300	
6 X(R)	1	2	1.300	(1)
7 @X(R)	2	4	2.600	

NOTE:
(1) JSR adds 1 ucycle (+0.180 usec).

Execution Time Instruction	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes
Double Operand					
ADD, SUB, BIC(B), BIS(B), MOV(B), XOR		1	1	0.780	(1)
CMP(B), BIT(B)		0	1	0.180	
Single Operand					
CLR(B), COM(B), INC(B), DEC(B), ADC(B), SBC(B), SXT		1	1	0.780	(1)
NEG(B)		1	2	0.960	(1)
ROR(B), ROL(B), ASR(B), ASL(B)		1	2	0.960	(2)
TST(B)		0	1	0.180	
SWAB		1	1	0.780	(2)
Branch					
All Branches (true)			3	0.540	
All Branches (false)			0	0.000	
SOB (true)			4	0.780	
SOB (false)			2	0.420	
Jump					
JMP			1	0.180	
JSR		1	5	1.500	
Control, Trap, and Miscellaneous					
Set/Clear Condition Codes			2	0.360	
MARK	1		8	2.380	
RTS	1		4	1.660	
RTI, RTT	2		6	2.960	
IOT, EMT, TRAP, BPT	2	2	13	5.420	

NOTES:
(1) DMO subtracts memory write and changes long to short ucycle (-0.600 usec).
(2) DMO subtracts memory write, changes long to short ucycle, and adds 1 ucycle (-0.420 usec).

Times Assumed for All Calculations

1) Microcycle times are 0.180 and 0.240 usec.
2) Microcycle time is extended by 0.150 usec by bus priority arbitration delay during BUT SERVICE.
3) Microcycle time is extended by 0.940 usec during DATI/DATIP (MOS memory).
4) Microcycle time is extended by 0.540 usec during DATO/DATOB (MOS memory).
5) Memory management unit delay is not included (+0.120 usec/memory cycle when enabled).

Appendix G: PDP-11/40 Instruction Execution Times

Microcycle Time = 0.140/0.200/0.300 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)	Notes
Fetch Time	1	4	1.120	

NOTE:
Execute phase of previous instruction may be overlapped with fetch. Consult execute phase notes for effect on timing.

Source Times

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	0	0.000	
1 @R or (R)	1	3	0.780	
2 (R)+	1	3	0.840	
3 @(R)+	2	5	1.720	
4 -(R)	1	3	0.840	
5 @-(R)	2	5	1.720	
6 X(R)	2	5	1.340	
7 @X(R)	3	7	2.120	

NOTE:
Odd-byte addressing (SM1-7) adds 2 ucycles (+0.340 usec).

Destinations Times

Data Manipulation (except MOV(B))

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	0	0.000	
1 @R or (R)	1	3	0.780	
2 (R)+	1	3	0.840	
3 @(R)+	2	5	1.720	
4 -(R)	1	3	0.840	
5 @-(R)	2	5	1.720	
6 X(R)	2	5	1.780	(1)
7 @X(R)	3	7	2.560	(1)

NOTES:
Odd-byte addressing (DM1-7) adds 2 ucycles (+0.340 usec).
(1) Single-operand instruction or SM0 subtracts 0 ucycles (-0.440 usec).

MOV(B)

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	0	0.000	
1 @R or (R)	0	2	0.340	
2 (R)+	0	2	0.340	
3 @(R)+	1	3	1.140	
4 -(R)	0	2	0.340	
5 @-(R)	1	3	1.140	
6 X(R)	1	3	1.140	(1)
7 @X(R)	2	5	1.980	(1)

NOTE:
(1) SM0 subtracts 0 ucycles (-0.440 usec).

	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes

Jump (JMP/JSR)					
Mode					
0 R	ILLEGAL				
1 @R or (R)	0		2	0.340	
2 (R)+	0		3	0.640	
3 @(R)+	1		2	0.940	
4 -(R)	0		2	0.440	
5 @-(R)	1		2	0.940	
6 X(R)	1		4	0.840	
7 @X(R)	2		4	1.340	
============================					
Execution Time					
Instruction					

Double Operand					
ADD, BIC(B), BIS(B), XOR		1	3	0.540	(1,2)
SUB		1	4	0.680	(1)
CMP(B), BIT(B)		0	3	0.480	(3)
MOV(B)		1	3	0.640	(4)

Single Operand					
CLR(B), COM(B), INC(B), DEC(B), ADC(B), SBC(B), ROL(B), ASL(B), SXT		1	4	0.620	(1,2)
NEG(B)		1	3	0.540	(1,2)
ROR(B), ASR(B)		1	4	0.840	(5)
TST(B)		0	4	0.620	(1,2)
SWAB		1	3	0.540	(1)

Branch					
All Branches (true)			3	0.640	
All Branches (false)			2	0.280	
SOB (true)			5	1.240	
SOB (false)			5	0.920	

Jump					
JMP			2	0.340	
JSR		1	6	1.480	

Control, Trap, and Miscellaneous					
Set Condition Codes			2	0.600	
Clear Condition Codes			3	0.900	
MARK	1		6	1.540	
RTS	1		4	1.280	
RTI, RTT	2		6	2.320	
IOT, EMT, TRAP, BPT	2	2	14	4.180	

NOTES:

If (single-operand instruction or SMO and double-operand instruction except MOVB), DMO, destination ≠ register 7, and no service request pending, then next fetch is overlapped (-1 ucycle/-0.640 usec from next fetch).

(1) If DMO, phase takes 3 ucycles and memory write is not done (0.480 usec).

(2) If odd-byte addressing (DM1-7), phase takes 5 ucycles (1.020 usec).

(3) If odd-byte addressing (DM1-7), phase takes 5 ucycles (0.820 usec).

(4) If byte instruction and DM1-7, phase takes 4 ucycles (0.880 usec). For DMO: If word instruction, phase takes 2 ucycles (0.340 usec). If byte instruction, phase takes 4 ucycles (0.680 usec).

(5) For DMO: If word instruction, phase takes 3 ucycles (0.740 usec). If byte instruction, phase takes 4 ucycles (0.880 usec). In neither case is memory write done.

Times Assumed for All Calculations

----

1) Microcycle times are 0.140, 0.200, and 0.300 usec.

2) A CLKOFF following a DATI/DATIP extends ucycle time by 0.500 usec minus sum of cycle times between DATI/DATIP (exclusive) and CLKOFF (inclusive).

3) A CLKOFF following a DATO/DATOB extends ucycle time by 0.200 usec minus sum of cycle times between DATO/DATOB (exclusive) and CLKOFF (inclusive).

4) Memory management unit delay is not included (+0.150 usec/memory cycle when enabled).

Appendix H: PDP-11/45 Instruction Execution Times

Microcycle Time = 0.150 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)	Notes
Fetch Time	1	3	0.450	

NOTE:
Execute phase of previous instruction may be overlapped with fetch. Consult execute phase notes for effect on timing.

Source Times

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	0	0.000	
1 @R or (R)	1	2	0.300	
2 (R)+	1	2	0.300	
3 @(R)+	2	5	0.750	
4 -(R)	1	3	0.450	
5 @-(R)	2	6	0.900	
6 X(R)	2	4	0.600	
7 @X(R)	3	7	1.050	

Destinations Times

Data Manipulation

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	0	0.000	
1 @R or (R)	1	2	0.300	
2 (R)+	1	2	0.300	
3 @(R)+	2	5	0.750	
4 -(R)	1	3	0.450	
5 @-(R)	2	6	0.900	
6 X(R)	2	5	0.750	(1)
7 @X(R)	3	8	1.200	(1)

NOTE:
MOV and DM1-7 subtracts memory read (-0.000 usec).
Odd-byte addressing (DM1-7) adds 1 ucycle (+0.150 usec).
(1) Single-operand instruction or SMO subtracts 1 ucycle (+0.150 usec).

Jump (JMP/JSR)

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	ILLEGAL			
1 @R or (R)	0	2	0.300	
2 (R)+	0	2	0.300	
3 @(R)+	1	4	0.600	
4 -(R)	0	2	0.300	
5 @-(R)	1	5	0.750	
6 X(R)	1	3	0.450	
7 @X(R)	2	6	0.900	

Execution Time Instruction	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes
Double Operand					
ADD, SUB, BIC(B), BIS(B), MOVB, XOR		1	2	0.300	(1)
CMP(B), BIT(B)		0	1	0.150	(1,2)
MOV		1	0	0.000	(1,3)
Single Operand					
CLR(B), COM(B), INC(B), DEC(B), ADC(B), SBC(B), ROL(B), ASL(B), SWAB, SXT		1	2	0.300	(1)
NEG(B)		1	4	0.600	(4)
ROR(B), ASR(B)		1	2	0.300	(1,5)
TST(B)		0	1	0.150	(1,2)
Branch					
All Branches (true)			1	0.150	
All Branches (false)			0	0.000	(6)
SOB (true)			3	0.450	(6)
SOB (false)			2	0.300	(6)
Jump					
JMP			1	0.150	
JSR		1	5	0.750	
Control, Trap, and Miscellaneous					
Set/Clear Condition Codes			1	0.150	
MARK	1		4	0.600	(6)
RTS	1		4	0.600	
RTI, RTT	2		7	1.050	
BPT	2	2	12	1.800	
IOT	2	2	11	1.650	
EMT, TRAP	2	2	13	1.950	

NOTES:

(1) For DMO:

If double-operand instruction, destination ≠ register 7, and SM1-7:

If odd-byte addressing, then phase takes 2 ucycles (0.300 usec), else phase takes 1 ucycle (0.150 usec). If no service request is pending, then next fetch is overlapped (-1 ucycle/-0.150 usec from next fetch).

If double-operand instruction, destination = register 7, and SM1-7:

Phase takes 2 ucycles (0.300 usec).

Otherwise (single-operand instruction or SMO):

Phase takes 1 ucycle (0.150 usec). If destination ≠ register 7 and no service request is pending, then next fetch is overlapped (-2 ucycles/-0.300 usec from next fetch).

No memory write is done.

(2) For DM1-7, if destination fetch is via Fastbus and no service request is pending, then next instruction fetch is overlapped (-1 ucycle/-0.150 usec from next fetch).

(3) DM1-2 adds 1 ucycle (+0.150 usec). If no service request is pending, then next fetch is overlapped (-1 ucycle/-0.150 usec from next fetch).

(4) DMO subtracts 2 ucycles and memory write (-0.300 usec).

(5) Odd-byte addressing adds 1 ucycle (+0.150 usec).

(6) If no service request is pending, then next fetch is overlapped (-1 ucycle/-0.150 usec from next fetch).

==============================

Times Assumed for All Calculations

----

1) Microcycle time is 0.150 usec.

2) Memory access time does not influence microcycle times (bipolar memory).

3) Memory management unit delay is not included (+0.090 usec/memory cycle when enabled).

Appendix I: PDP-11/60 Instruction Execution Times

Microcycle Time = 0.170 Microseconds

	Memory Reads	Micro-Cycles	Time (usec)	Notes
Fetch Time	1	3	0.510	

NOTES:

The following instructions take 1 additional ucycle (+0.170 usec) to decode: XOR, SWAB, SXT, JSR, set/clear condition codes, MARK, SOB, RTS, RTI, RTT, IOT, EMT, TRAP, BPT, MFPI(D), MTPI(D).

Fetch or execute phase of previous instruction may be overlapped with fetch. Consult execute phase notes for effect on timing.

Source Times

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	0	0.000	
1 @R or (R)	1	2	0.340	
2 (R)+	1	2	0.340	
3 @(R)+	2	5	0.850	
4 -(R)	1	3	0.510	
5 @-(R)	2	6	1.020	
6 X(R)	2	4	0.680	
7 @X(R)	3	7	1.190	

NOTE:

For SM1-7: Word instruction except MOV and DM1-7 adds 1 ucycle (+0.170 usec). Byte instuction adds 2 ucycles (+0.340 usec).

Destination Times

Data Manipulation (except MOV(B) and MTPI(D))

Mode	Memory Reads	Micro-Cycles	Time (usec)	Notes
0 R	0	0	0.000	
1 @R or (R)	1	2	0.340	
2 (R)+	1	2	0.340	
3 @(R)+	2	5	0.850	
4 -(R)	1	3	0.510	
5 @-(R)	2	6	1.020	
6 X(R)	2	5	0.850	(1)
7 @X(R)	3	8	1.360	(1)

NOTES:

Byte addressing (DM1-7) adds 2 ucycles (+0.340 usec).

(1) Single-operand instruction except SWAB or SXT or SM0 and double-operand instruction except XOR subtracts 1 ucycle (-0.170 usec).

	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes

MOV(B) and MTPI(D)					
Mode					
0 R	0		0	0.000	
1 @R or (R)	0		0	0.000	
2 (R)+	0		1	0.170	
3 @(R)+	1		3	0.510	
4 -(R)	0		1	0.170	(1)
5 @-(R)	1		4	0.680	
6 X(R)	1		3	0.510	(2)
7 @X(R)	2		6	1.020	(2)

NOTES:
MOVB, SMO, and DM1-7 adds 2 ucycles (+0.340 usec) for even byte, 3 ucycles (+0.510 usec) for odd byte.
(1) MOV and SMO adds 1 ucycle (+0.170 usec).
(2) MOV(B) and SMO subtracts 1 ucycle (-0.170 usec).

	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes

Jump (JMP/JSR)					
Mode					
0 R	ILLEGAL				
1 @R or (R)	0		1	0.170	
2 (R)+	0		2	0.340	
3 @(R)+	1		2	0.340	
4 -(R)	0		1	0.170	
5 @-(R)	1		3	0.510	
6 X(R)	1		2	0.340	
7 @X(R)	2		5	0.850	
==============================					
Execute Times					
Instruction					

Double Operand					
ADD, MOV		1	2	1.170	(1,6)
SUB		1	3	1.340	(1,7)
BIC(B), BIS(B)		1	2	1.170	(1,6,12)
CMP(B)		0	1	0.170	(1,11)
BIT(B)		0	1	0.170	(1)
MOVB		1	2	1.170	(4)
XOR		1	3	1.340	(7)

Single Operand					
CLR(B), COM(B)		1	3	1.340	(2,7)
INC(B), DEC(B), ADC(B), ROL(B), ASL(B)		1	3	1.340	(2,7,8)
NEG(B)		1	4	1.510	(7,8)
SBC(B)		1	4	1.510	(6,8)
ROR(B)		1	4	1.510	(6)
ASR(B)		1	5	1.680	(7,9)
TST(B)		0	2	0.340	(2,5)
SWAB		1	5	1.680	(7)
SXT		1	6	1.850	(7)
MFPI(D)		1	21	4.400	(13)
MTPI(D)	1	1	22	4.570	(6)

	Memory Reads	Memory Writes	Micro-Cycles	Time (usec)	Notes

Branch					
All Branches (true)			4	0.680	(3)
All Branches (false)			2	0.340	
SOB (true)			10	1.700	(3)
SOB (false)			7	1.190	

Jump					
JMP			1	0.170	
JSR		1	6	1.850	(3,10)

Control, Trap, and Miscellaneous					
Set/Clear Condition Codes			8	1.190	
MARK	1		9	1.530	
RTS	1		4	0.680	
RTI	2		10	1.700	
RTT	2		19	3.230	
IOT, EMT, TRAP, BPT	2	2	22	5.400	(14)

NOTES:

(1) If SMO, DMO, source ≠ register 7, and destination ≠ register 7, then fetch overlap is attempted. If no service request is pending at conclusion of instruction, then next fetch is overlapped (-2 ucycles/-0.340 usec from next fetch); otherwise, add 2 ucycles (+0.340 usec) to service phase following instruction for PC rollback, add 1 memory read (+0.000 usec) to next fetch for instruction refetch.

(2) If DMO and destination ≠ register 7, then fetch overlap is attempted. If no service request is pending at conclusion of instruction, then next fetch is overlapped (-2 ucycles/-0.340 usec from next fetch); otherwise, add 2 ucycles (+0.340 usec) to service phase following instruction for PC rollback, add 1 memory read (+0.000 usec) to next fetch for instruction refetch.

(3) If no service request is pending, then next fetch is overlapped (-2 ucycles/-0.340 usec from next fetch); otherwise, subtract 1 ucycle (-0.170 usec) from execute.

(4) For DMO: SMO subtracts memory write (-0.830 usec). SM1-7 subtracts 1 ucycle and memory write (-1.000 usec).

(5) DMO subtracts 1 ucycle (-0.170 usec).

(6) DMO subtracts 1 ucycle and memory write (-1.000 usec).

(7) DMO subtracts 2 ucycle and memory write (-1.170 usec).

(8) DM1-7 and byte addressing adds 1 ucycle (+0.170 usec).

(9) DM1-7 and byte addressing adds 3 ucycles (+0.510 usec).

(10) DM3,5-7 adds 1 ucycle (+0.170 usec).

(11) SM1-7, DMO, and word addressing adds 1 ucycle (+0.170 usec).

(12) SMO, DM1-7, and byte addressing adds 1 ucycle (+0.170 usec).

(13) SMO adds 1 ucycle (+0.170 usec).

(14) If new PC odd: Microcontrol transfers to writable control store if present and instruction timing does not apply; otherwise, trap sequence continues normally with 3 extra ucycles (+0.510 usec).

=============================

NOTE:
Accessing the following internal addresses invokes microcode which adds additional microcycles in all phases:

772300-16	Kernel Page Descriptor Registers
772340-56	Kernel Page Address Registers
777540	Writable Control Store Status Register
777542	Writable Control Store Address Register
777544	Writable Control Store Data Register
777570	Console Switch and Display Register
777572	Memory Management Status Register 0
777574	Memory Management Status Register 1
777576	Memory Management Status Register 2
777600-16	User Page Descriptor Registers
777640-56	User Page Address Registers
777744	Memory System Error Register
777746	Cache Control Register
777752	Cache Hit/Miss Register
777766	CPU Error Register
777770	Microprogram Break Register
777774	Stack Limit Register
777776	Processor Status Word

Times Assumed for All Calculations
----

1) Microcycle time is 0.170 usec.
2) Microcycle time is extended by 0.000 usec during DATI/DATIP with cache hit (all tabulated times assume cache hit on read).
3) Microcycle time is extended by 1.075 usec during DATI/DATIP with cache miss.
4) Microcycle time is extended by 0.830 usec during DATO/DATOB.
5) Memory management unit adds no delay when enabled.

References

[Bell70] Bell, C. G., R. Cady, H. McFarland, B. Delagi, J. O'Loughlin, R. Noonan, and W. Wulf, "A New Architecture for Minicomputers - The DEC PDP-11," *AFIPS Conference Proceedings*, vol. 36, pp. 657-675, 1970.

[Bell71] Bell, C. G. and A. Newell, *Computer Structures: Readings and Examples*, McGraw-Hill, New York, 1971.

[Mudg77] Mudge, J. C., "Design Decisions Achieve Price/Performance Balance in Mid-Range Minicomputers," *Computer Design*, vol. 16, no. 8, pp. 87-95, August 1977.

[Olou75] O'Loughlin, J. F., "Microprogramming a Fixed Architecture Machine," Infotech State of the Art Report 23 *Microprogramming and Systems Architecture*, pp. 205-224, 1975.

[Siew76] Siewiorek, D. P. and M. R. Barbacci, "The C-MU RT-CAD System: An Innovative Approach to Computer Aided Design," *AFIPS Conference Proceedings*, vol. 45, pp. 643-655, 1976.

[Stre76a] Strecker, W. D., "Cache Memories for PDP-11 Family Computers," *3rd Symposium on Computer Architecture Proceedings*, pp. 155-158, 1976.

[Stre76b] Strecker, W. D., private communication, 1976.

[Thom77] Thomas, D. E. and D. P. Siewiorek, "Measuring Designer Performance to Verify Design Automation Systems," *Design Automation Conference Proceedings*, vol. 14, pp. 411-418, 1977.

[Wilk53] Wilkes, M. V. and J. B. Stringer, "Microprogramming and the Design of the Control Circuits in an Electronic Digital Computer," *Proceedings of the Cambridge Philosophical Society*, pt. 2, vol. 49, pp. 230-238, April 1953.

The following Digital Equipment Corporation documents define the architecture and instruction set of the PDP-11 in addition to detailing features peculiar to individual processor implementations:

PDP-11/20/15/R20 Processor Handbook, EB-01855-71.
PDP-11/04/05/10/35/40/45 Processor Handbook, EB-05138-76.
PDP-11/04/34/45/55 Processor Handbook, EB-A5138-76.
PDP-11/60 Processor Handbook, EB-06498-20.
Microcomputer Handbook (LSI-11), EB-07948-53.
PDP-11 Peripherals Handbook, 1976, EB-05961-76.
PDP-11 Programming Reference Card, EH-S1046-76.
LSI-11/PDP-11/03 Programming Reference Card, EK-04779-75.

The following Digital Equipment Corporation documents were the source for information on individual PDP-11 processor implementations:

LSI-11

LSI-11, PDP-11/03 User's Manual, EK-LSI11-TM-002.
Western Digital Corporation Microprocessor Set (MCP) Functional Specification, DEC internal document.
KD11-F Print Set, MP00049.
LSI-11 microcode, DEC internal document.
LSI-11 EIS/FIS microcode, DEC internal document.

PDP-11/04

KD11-D Processor Manual (Preliminary), EK-KD11D-TM-PRE.
KD11-D Print Set, MP00020.

PDP-11/10

KD11-B Processor Maintenance Manual, EK-KD11B-MM-001.
PDP-11/05-S, 11/10-S Systems Engineering Drawings.

PDP-11/20

KC11 Processor Manual, 11-HKCB-D.
KC11 Processor Engineering Drawings.

PDP-11/34

KD11-E Processor Manual (Preliminary), EK-KD11E-TM-PRE.
KD11-E Print Set, MP00043.

PDP-11/40

KD11-A Processor Maintenance Manual, EK-KD11A-MM-001.
PDP-11/40 System Engineering Drawings.

PDP-11/45

KB11-A,D Central Processor Unit Maintenance Manual, EK-KB11A-MM-004.
PDP-11/45 System Engineering Drawings.

PDP-11/60

KD11-K Processor Manual.
KD11-K Print Set.
PDP-11/60 Microprogramming Specification, draft.
PDP-11/60 microcode, DEC internal document.