PDP-11

DETAILS OF DESIGN, KA11

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

PDP-11

DATE: September 4, 1969

SUBJECT: Details of Design, KA11

TO: Those Concerned

FROM: Jim O'Loughlin

Attached are initial notes on the details of processor design for the KA11. These notes concern the Block Diagram and the Instruction Flow Diagram. Some information on basic timing is provided.

Additional notes will discuss bus operation, waveforms, and each logic print. Equations for the combinational logic and most flip-flops are partially completed. This information will be available to training, technical publications and others at an undetermined time.

Note 1: (Use with Block Diagram and Instruction Flow Diagram), 8-21-69

The processor for the PDP-11 consists primarily of data paths, its memory, and control. The data paths have the adders A and B input gates, latches upon these input gates to hold data, and, after the adders, shift and rotate gating. The memory and the data paths are 16 bits wide, with the data paths segmented on modules 8 bits wide. This segmentation of the data paths was to facilitate module construction and also allow some field servicing by interchanging modules. The segmentation also allows access to certain signals on the rotate and shift gating necessary for byte manipulation. The memory is 16 bits wide to match the data paths. Note: reference to memory refers to the processor scratch pad memory; core memory appears only as a bus address. The memory (processor's) is general with four separate gatable inputs to the address matrix, with only the positive data input and output provided. Further expansion might require an inhibit upon address, this can be added at the expense of some over voltage clamps.

The memory input is from the bottom of the data paths, that is, the output of the shift and rotate gating. The memory output is connected to the A and B gates of the data paths. This ring-like structure accesses the bus after the shift and rotate gates; it also accesses the Status register at this point. The ring is, in turn, accessed by the bus at the top of the A and B input gating. This structure represents a compromise between speed and simplicity. It would be faster if access was made to the bus both from the memory and from the top of the data paths; it would be more complicated. The Bus Address Register (BAR) receives its input from the bottom of the data paths. Console displays are of the bottom of data paths and the BAR. A further note on the speed vs. simplicity compromise. The single bus access at the bottom of the data paths results in autodecrements and auto-increment (which does not change the address immediately) requiring the same time.

The access to the IR register is off the bus directly. It is possible to place it at the bottom of the data paths along with Status but this involves some additional time in every instruction, with the transmission of the bus information through the data paths to the IR. The deciding factor in the IR location will be the number of pins available on specific control boards. The use of the data paths does cut down the number of pins considerably.

The general function of control is to manipulate the memory outputs and the bus inputs to the data paths to provide the instructions specified. The flow diagram indicates these manipulations. The basic symbols on the flow diagram are bus operations DATI, and DATO, both of which are separate sequences within control. Also indicated on the flow diagram are circles with R and W in them. These represent, respectively, Read cycles of the memory and Write cycles of the memory. Note that processor references to memory are to its own 16 bit scratch pad; 8 words of which are addressable in instruction allowing internal addresses; all 16 words (includes registers used in machine operation) can be EXAMined or DEPosited into from the console. The registers for this purpose have the quasi-bus-address noted; this address cannot be used except from the console.

The basic time for a Read or Write within this memory represents a basic timing unit of the machine. There is always a Read or Write cycle while the processor clock is operating. This Read or Write Write cycle consists of 250 ns and is derived from a 2 bit R/W shift register driven by both phases of a square wave oscillator of 125ns period (62.5ns up, 62.5ns down).

Referencing the flow diagram again, the FETCH cycle consists of a DATI where the PC is used as the bus address, and is incremented by 2. The bus buffered instruction data is loaded directly into the IR register.

A basic machine time unit of 250ns is allowed for decoding. After that time a decision is made; whether an error has occurred within unimplemented or illegal instruction decoded; whether further address information is necessary; or whether no further information is necessary and the instruction can be executed. The appropriate factors in making decisions are noted at the branch points on the flow diagram. If additional address information is necessary, or if additional data is necessary that requires address cycles, the address segments of the flow is entered. If a Source exists, DATI's are done to the extent that the M and D bits of the address mode specify. The register reference is used as the base in calculating the addresses. When the data is obtained, it is stored at a location in memory called Source. This is one of the temporary storage registers in scratch pad memory. After Source has been obtained; again decisions are made as to whether or not Destination information is required externally. If it is, the address sequences entered again with DATI's being done until the last bus transaction is made. This transaction, where the actual data is being obtained, is a DATIP cycle. The DATIP holds the bus device for modified data; processor maintains bus control. This data is stored in the latch input gating. All Destination information that requires an exterior transfer is placed here and is assumed to be here during the EXECUTE cycle. The actual latch (LATCH A or LATCH B) depends upon the instruction being executed and is noted elsewhere.

When the machine enters the EXECUTE cycle, it implements the instructions specified in the IR.

In the unaries and the binaries where specific data was referenced, some address mode combinations exist where the data still has to be obtained. If it was an exterior reference for destination it would have to be brought out from memory. Source is either a register reference in the scratch pad memory or is located in a temporary location (Source).

When byte manipulations are done, odd byte information is transferred to the lower 8 bits by the DATI or DATIP bus sequence. The re-transfer of information is done by the rotate/shift gating at the bottom of the data paths during the last portion of data manipulation. This saves a machine sequence consisting of a Write and a Read. Roatate and Shift instructions do require these sequences as the rotate/shift gating must first be used for the data manipulation. A DATO represents the finishing of the DATIP cycle previously started in the address sequence; a write to scratch pad completes the EXECUTE for internal memory destinations.

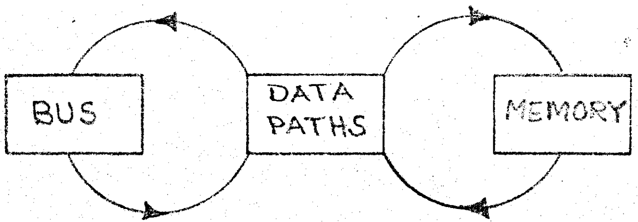
After EXECUTE, the SERVICE state is entered; depending upon the status of several flags the machine might remain in SERVICE or fall directly through to FETCH. In SERVICE, machine cycles exist for the sequencing of trap instructions, error traps and the trap response to bus interrupt.

The JMP and JSR instruction deserves special note, their address is calculated and the bus sequence is aborted. The address is read into the memory: either directly into the PC for a JMP or into a temporary location for JSR.

Note 1 A: (Use Block Diagram and Latch Data Sheet), 9/2/69

Essential to understanding the KA11 is a knowledge of the data flow through the data paths and memory. Note that the memory referred to is the processors memory. This is a 16 word by 16 bit solid state memory located internal to the processor and accessed without bus operations. Core memory or other types of memory located external to the processor are addressed by bus operations and these bus operations are indestinguishable from bus operations to other peripherals such as paper tape readers disk or tapes. For this reason memory operations of the processor refer only to its memory and not to bus operations.

Data flow through the machine can be considered as two adjacent circles (see below) with the data paths common to both at the junction of the circles,



the bus at the left, and the memory at the right. The data paths manipulate the data. Within them are the adders, shift and rotate gating, latches which temporarily store data while being operated upon, and gate inputs that allow the addition of constants to the data. There are two latch inputs to the adders. (A Latch and B Latch). Each of these gate inputs allows the introduction of memory data and bus data, either in its true or complement form. The particular inputs are carefully allocated between the A and B input gating to allow the various manipulations required within the instructions. The inputs are also arranged so that either bus or memory data reference may be used in the data manipulations.

The B input gating has a memory input, a received bus data input, and a trap address input. This gate is used in general transfers of information through the machine. When it is desired to move data from the memory to the bus or from the memory to the memory the B input gating is usually used. Another reason for utilization of the B gate is that constants which are added in address calculations and in certain instructions are introduced on the A input gating. In order for the data to be altered by these constants, it must be on the other, or the B input, to the adder.

The A input gating has a memory input, a complement memory input and a complement receive bus data input. The compliment inputs facilitate the use of two's complement arithmetic inputs, and complement inputs in the logic instructions. The presence of both a complement and uncomplemented memory inputs allows the generation of constants within the A input gating. Segmentation of the gating across the 16 bits with bit Ø being gated separately on its M and -M inputs allows the generation of the specific constants desired. This allows the generation of contants: +1 by activating both M and -M inputs on bit Ø and activating no gate inputs on bits 1 → 15, -1 in two's complement by gating M and -M on all bits 0 → 16; -2 by gating bits 1 → 15 from M and -M and not gating bit Ø.

An AND/NOR gate with 4 two input AND gates is common to both Latch A and Latch B. On these two inputs of the AND gates are a gate signal which is clocked, and the input data itself. An exception to this is the sign extension input. Clocking is necessary upon the gating signal because of the possibility of transience occuring during state changes. Information is occasionally held in the latches across state changes.

The complement inputs on A also serve in the logical anding of 2 sets of data. This is done by utilizing DeMorgan's Theorem, and "oring" the complement of the two sets of data and then complementing the result. This ploy is necessary because no AND gating exists for data in the data paths; OR gating does exist with additional cycles required.

Note 2: (Use block diagram and timing controlprints), 9/2/69

The basic timing of the KA11 is provided by a discrete component clock of 125ns. This clock is a square wave clock not a pulse generator. Both phases of this clock are used to shift a two-bit shift register called the Read/Write Shift Register. This Read/Write Shift Register provides the basic timing of the KA11 memory (the scratch pad memory). Off of this Read/Write Shift Register is generated the system clock to the KA11. This system clock is of 250ns period and provides the timing for the Instruction Shift Register (ISR), and Bus Shift Register (BSR).

It also provides various discrete signals to data path latches and the condition code flipflops. These discrete signals are used to load and clear these FF's independent of the 250ns machine state boundary provided by the system clock. This independence is necessitated by the particular timing technique used. The discrete signals periods are named R/WØ, R/W1, R/W3, R/W2, and are of 62.5ns period.

Also associated with the basic clock is an inhibit circuit and flipflop which allows the clock to be turned off, thus operating asynchronously with bus transmisions. This characteristic combined with the alternate shifting, loading, and quiesence of the various shift registers provides an asynchronous flow of states. Few rest states are provided and machine states are skipped if unnecessary. This saves time, cost IC's.

The various shift registers are interconnected by combinational logic which determines the next state. Additional combination logic operates off the shift registers and the Instruction Register to provide the conditional levels necessary for data path and memory control. These levels are not transient free. Particular care must be taken after the 250ns machine state boundaries. Transient free transitions in the machine are provided by the basic clock and the Read/Write Shift Register.

Various factors were considered in the evolution of the KA11 timing. They are listed as follows.

1. The processor had to be contained on a few printed circuit boards. Any extensive use of the delay lines such as in the 8/I was precluded.

2. The compactness of the KA11 noted in 1. allows a central clock to be used with adequate control of clock, skew, and loading.
3. The Signetics 8270 four-bit shift register provides a great deal of logic power. It is a shift register which can be loaded in parallel, shifted or prohibited from reacting to the clock. These characteristics have formed a basis for the timing. Especially important is the ability to prohibit reaction to clock.
4. With shift registers it is possible to provide transient-free time states. Additional imputs to the decoding gates and a special attention to the combinational logic is necessary with the decoded time states placed immediately

before the output of the combinational logic. In a large system, this requires an inordinate amount of effort. For this reason the combinational logic of the KA11 is not considered to be transient free; nor is any dependence made on this characteristic.

5. To counter transients and state transisions, some techniques have been used: A) Latch flipflops are provided for control of the data latches, in addition the gate inputs are gated by the Read/Write Shift Register (this shift register does not have transients across state transisions); B) The use of discrete time states from the Read/Write Shift Register as pulses. This has been done for the write pulse on the scratch pad memory, for the clocking signal on the status register, and for various control FF's.

6. The use of combinational logic between the shift registers, and between the shift registers and the controlled segments of the processor, allows a segregation of the control which is especially beneficial in module debug. This involves the separation of the shift registers and basic clock and the combinational logic. The combinational logic is not simple and in many cases is quite wide with multiple inputs reducing to single outputs. While this logic is not simple, it is reasonably easy to debug using the automatic testing. There are no closed loops. The separation of the shift register combinational logic along with the other combinational logic is easily effected. The benefit

of easy module debug has a consequence in that machine operation is dependent upon combining the debugged segments successfully.

The mechanization of the combinational logic in the KA11 has tried to maintain a 25∅ns system clock. Some compromises have been made in simplicity to effect this, although a wide gate is not excessively more costly than one where unavailable common signals are generated. The word "unavailable" is used because of pin limitations which will exist when the logic is allocated to specific module boards. In order to allow the machine to run at a faster rate it will be necessary to reduce the delay between certain points in the combinational logic. Present critical paths are between the setup of states and the gating signals to the top of the data paths. These include the -1, +1 signals and CARRY ∅.

Note 3. (use with Instruction Flow Diagram)

This note contains information on the servicing of NPR's and BR's; the sequencing of ISR and BSR; and the meaning of various address modes. Trap service is also noted.

Upon initial powering up, a power clear signal zeros the ISR, BSR, state and flag FF's and sets the CONSF. Since all state FF's zero is SERVICE, and CONSF (1) indicates console control; the machine starts in a rest state under console control. It waits for a PERIF RELEASE signal which is a passive return of bus control to the processor. This signal is the non-assertion of BBSY, SACK, GRANT and SSYN. A release from console differs from other releases in that the BR and NPR flip-flops are cleared; the machine goes to FETCH.

The servicing of NPR's occurs after the bus cycles DATI and DATO (DATOB). No requests are serviced after a DATIP, until the completion of its restoring DATO. The machine restarts (it halts for the asynchronous data transfers). No INTR routines are allowed and the servicing of NPR's continues until PERIF RELEASE occurs. The updating of the NPR flip-flop during this period is accomplished by MSYN .

BR's are serviced upon the entrance to SERVICE if no machine flags are set. These flags include the trap instructions, WAITF, HALTF and TRACF. If these flags are set the machine services these traps in the order noted in Table 4.1. If a flag is serviced the machine will recycle through SERVICE for other machine flags; it will not recycle for BR's. (NPR's are serviced within the noted DATO's and DATI's).

A BR, therefore, is serviced only upon entry to SERVICE with no machine flags set. BR's will continue to be serviced until all are done (PERIF RELEASE) or an INTR sequence is initiated. The INTR sequence results in a trap sequence with the Trap Marker provided by the external device; a PERIF RELEASE allows entry into FETCH.

Table 4.1: Order of Service

Order	Flag	Trap Adrs	Use
1	HALTF	Read R0	Halt machine with R0 displayed Rest state is SERVICE *ISR0
2	BERRF	(04)	TIME OUT & ODD ADRS ERR.
3	TRAPF	(04)	ILL INSTR
		(10)	RSVD INSTR
		(14)	TRT INSTR
		(20)	IOT INSTR
		(30)	STR INSTR
		(34)	UTR INSTR
4	TRACF	(14)	Trap one instr. after T bit set
5	OVFLF	(04)	Trap after instr. if stack overflow
6	PWRF	(24)	Trap if Power Fail occurs

In FETCH the machine exits the SERVICE segment of the instruction flow by simultaneous setting: the FETCH flip-flop: BSR to 1 (this is a bus cycle); and ISR to Ø. ISR remains in zero while the BSR sequences through a DATI. In that DATI, the PC is used as the address on the bus; and during BSR 7, the same PC is autoincremented by two. The data obtained in this DATI is clocked into the IR and latched into LATCH B of the data paths. The data in LATCH B is used for an address vector, in a BRANCH instruction.

In all instruction, however, the IR is decoded. The instruction information was obtained by DATI, the last state of which is BSR 7. If no bus error or NPR occured, the present state of BSR 7 and ISR Ø and FETCH equals 1, would cause ISR Ø to be shifted to ISR 1 And BSR 7 would become BSR Ø. The uses of BSR zero need redefinition. They have been expanded from the previous use of a readtime in a DATM operation and a rest state for PTR sequences. It has become necessary to provide for the BSR shift register, a general rest state. This rest state is BSR Ø, and is entered at the end of all bus operations, to elimate the gating of all signals dependent upon bus operation.

If an NPR request had occured, the loading of BSR to Ø would still be appropriate, except that an exit signal from the bus operation would not be generated. Instead, bus control passes to a peripheral (or console), the actual peripheral selected is done assynshronously to machine operation in the Priority Control section. Such an operation does not occur unless bus control is to be transferred. The processor, would become dormant and wait for a return from a NPR request. This return is a passive return upon a non-assertion of BBSY, SACK, GRANT and SSYN . Upon this return, an exit signal is generated from the bus operation (the original DATI of FETCH), and ISR Ø is shifted to ISR 1. The instruction is now decoded. Upon certain instructions (such as the operate instructions, clearing or setting the condition codes, a WAIT instruction or a halt instruction) the appropriate flag flip-flops are set during the latter part of ISR 1.

Most instructions will involve an exterior bus address reference. If such a reference is called by either the SOURCE or the DESTINATION, it may involve multiple bus operations. Accounting of these bus operations is done by comparing the ISR state with the address mode information within the IR. This information is contained in IR bits 5, 4, 3 for DESTINATION and 11, 10, and 9 for SOURCE. The address modes are register, autoincrement, autodecrement, and index, with all of these either direct or deferred. Additional address information is provided by bits 2, 1, and Ø for DESTINATION and 8, 7, 6 for SOURCE. This information is the base register within the processor upon which the address calculations are to be made. Figure 4-1 summarizes the address modes. They are listed in order of complexity (i.e. number of bus operations found to obtain the final data). In the logic prints the address modes are given names from ADRS MODE Ø which is to M=ØØ,D=Ø to ADRS MODE 7 with M=11 and D=1.

ADDRESS MODES

Figure 4-1

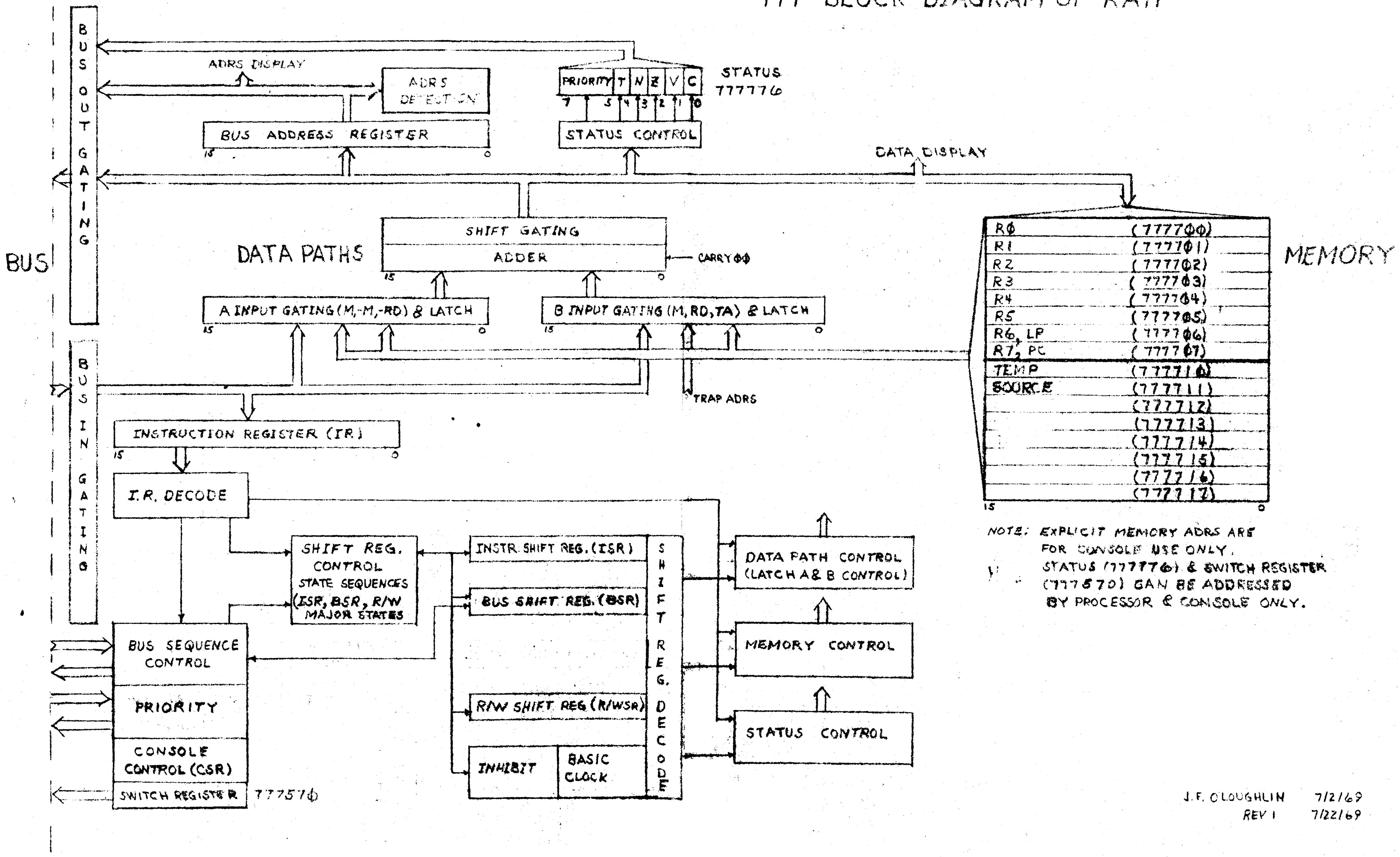
ADRS MODE	M	D		
Ø	ØØ	Ø	ADRS is a REG, DATA = (REG)	
1	ØØ	1	ADRS is (REG), DATA = ((REG))	

All other modes have an implied level of defer.

ADRS MODE	M	D		
2	Ø1	Ø	ADRS is (REG), DATA = ((REG)) after ADRS is used (REG)+2 → (REG)	AUTOINCREMENT
3	Ø1	1	ADRS is ((REG)), DATA = (((REG))) after ADRS is used (REG)+2 → (REG)	
4	1Ø	Ø	ADRS is (REG)-2, DATA = ((REG)-2) ADRS decrement before use, also (REG)-2 → (REG)	AUTODECREMENT
5	1Ø	1	ADRS is ((REG)-2), DATA = (((REG)-2)) ADRS decrement before use, also (REG)-2 → (REG)	
6	11	Ø	ADRS is (REG)+Q (next word in exterior memory) DATA = ((REG)+Q) Sequence: 1) after instruction decoded, PC used to bring next word, Q, and up-date PC: (PC)+2 → (PC) 2) read out of internal memory (REG) and add to Q 3) use (REG)+Q as bus address of data	INDEX
7	11	1	ADRS is ((REG)+Q), DATA is (((REG)+Q))	

Note: Above use of () indicates "contents of."

PM BLOCK DIAGRAM OF KA11



NOTE: EXPLICIT MEMORY ADRS ARE FOR CONSOLE USE ONLY. STATUS (777776) & SWITCH REGISTER (777570) CAN BE ADDRESSED BY PROCESSOR & CONSOLE ONLY.

J.F. CLOUGHLIN 7/2/69
REV 1 7/22/69

LOCATION OF SOURCE & DESTINATION DATA IN THE LATCHES DURING EXECUTE:

INSTRUCTION	DESTINATION	SOURCE
MOV	DO NOT GATE	B LATCH ← +M, SOURCE if ($\overline{\text{SOURCE MODE }\phi}$) B LATCH ← +M, REG if (SOURCE MODE ϕ)
ADD	B LATCH ← +RD if ($\overline{\text{DEST MODE }\phi}$) B LATCH ← +M, REG if (DEST MODE ϕ)	A LATCH ← +M, SOURCE if ($\overline{\text{SOURCE MODE }\phi}$) A LATCH ← +M, REG if (SOURCE MODE ϕ)
SUB	B LATCH ← +RD if ($\overline{\text{DEST MODE }\phi}$) B LATCH ← +M, REG if (DEST MODE ϕ) CARRY ϕϕ IS ACTIVE TO PROVIDE TWO'S COMPLEMENT	A LATCH ← -M, SOURCE if ($\overline{\text{SOURCE MODE }\phi}$) A LATCH ← -M, REG if (SOURCE MODE ϕ)
CMP	A LATCH ← -RD if ($\overline{\text{DEST MODE }\phi}$) A LATCH ← -M, REG if (DEST MODE ϕ)	B LATCH ← +M, SOURCE if ($\overline{\text{SOURCE MODE }\phi}$) B LATCH ← +M, REG if (SOURCE MODE ϕ)
BIT	A LATCH ← -RD if ($\overline{\text{DEST MODE }\phi}$) A LATCH ← -M, REG if (DEST MODE ϕ)	A LATCH ← -M, SOURCE if ($\overline{\text{SOURCE MODE }\phi}$) A LATCH ← -M, REG if (SOURCE MODE ϕ)
BIC	A LATCH ← -RD if ($\overline{\text{DEST MODE }\phi}$) A LATCH ← -M, REG if (DEST MODE ϕ)	A LATCH ← M, SOURCE if ($\overline{\text{SOURCE MODE }\phi}$) A LATCH ← M, REG if (SOURCE MODE ϕ)
BIS	B LATCH ← +RD if ($\overline{\text{DEST MODE }\phi}$) B LATCH ← +M, REG if (DEST MODE ϕ)	B LATCH ← M, SOURCE if ($\overline{\text{SOURCE MODE }\phi}$) B LATCH ← M, REG if (SOURCE MODE ϕ)
TST	B LATCH ← RD if ($\overline{\text{DEST MODE }\phi}$) B LATCH ← M, REG if (DEST MODE ϕ)	DO NOT GATE
COM	A LATCH ← -RD if ($\overline{\text{DEST MODE }\phi}$) A LATCH ← -M if (DEST MODE ϕ)	DO NOT GATE
INC ADC*C(1)	B LATCH ← RD if ($\overline{\text{DEST MODE }\phi}$) B LATCH ← M, REG if (DEST MODE ϕ)	CARRY ϕϕ ACTIVE DURNING WRITE
DEC SBC*C(1)	B LATCH ← RD if ($\overline{\text{DEST MODE }\phi}$) B LATCH ← M, REG if (DEST MODE ϕ)	BOTH A LATCH ← -M, +M ACTIVE TO PROVIDE -1
CLR	DO NOT GATE	DO NOT GATE
NEG	A LATCH ← -RD if ($\overline{\text{DEST MODE }\phi}$) A LATCH ← -M, REG if (DEST MODE ϕ)	DO NOT GATE
SWAB	B LATCH ← RD if ($\overline{\text{DEST MODE }\phi}$) B LATCH ← M, REG if (DEST MODE ϕ)	DO NOT GATE

NOTES:

1. SOURCE DATA EXTERIOR TO THE PROCESSOR IS OBTAINED IMMEDIATELY AFTER FETCH (BEFORE ANY DESTINATION DATA). IT IS STORED IN A TEMPORARY REGISTER NAMED, "SOURCE." DESTINATION INFORMATION IS NEXT OBTAINED, EITHER EXTERNAL OR INTERNAL, AND PLACED IN AN APPROPRIATE LATCH, SOURCE DATA IS THEN OBTAINED FROM INTERNAL MEMORY: FROM "SOURCE" IF ORIGINAL DATA EXTERNAL OR FROM REG IF INTERNAL
2. SOURCE OR DESTINATION MODE ϕ INDICATES INTERNAL DATA (M=ϕϕ, D=ϕ)
3. LATCH A OR LATCH B INPUTS ARE USED ALONG WITH DIRECT OR COMPLEMENTED DATA TO EFFECT CORRECT LOGIC OPERATION.
4. BAR SYMBOL IN FRONT OF LOGIC SYMBOL REPRESENTS COMPLEMENT: $-A = \overline{A}$. ASTERIK IS USED FOR LOGICAL "AND": $A*B = A \cdot B$
5. BIT AND BIS ARE VARIATIONS OF AN "AND" INSTR. WITH THE FOLLOWING SEQUENCE REQUIRED: WRITE THE "INCLUSIVE OR" OF SOURCE AND DEST, READ COMPLEMENT.
6. BIT, BIC AND BIS REQUIRE LATCH CLEAR INHIBIT ON SOURCE LOAD. BTST DOES NOT WRITE IN DEST,
7. SOURCE REFERENCES IN NON-BINARY ALLOW FOR CONSTANTS,
8. CHART CAN BE EXPANDED TO INCLUDE OTHER MACHINE STATES; HOWEVER, B LATCH USE IS USUAL IN ADRS OR TRANSFER SITUATIONS.

	REV ϕ	6/26/69
PM INSTR SET	REV 1	7/2/69
CORRECTIONS	REV 2	7/22/69

PM PDP 11 INSTRUCTIONS

BINARY GROUP

B	OP	SOURCE	DESTINATION
15	14 12	11 6	5 0

B	OP	
0-WORD	001	MOV : S→D
1-BYTE	010	CMP : TEST S-D
	011	BIT : TEST $S \wedge D$
	100	BIC : $\overline{S} \wedge D \rightarrow D$
	101	BIS : $S \vee D \rightarrow D$

WORD OP'S

0	110	ADD : S+D→D
1	110	SUB : D-S→D
X	111	RESERVED

SOURCE DESTINATION

M	D	REG	M	D	REG
11 10	9	8 6	5 4	3	2 0

M	ADDRESS BASE MODE
00	REGISTER
01	AUTO-INC (AFTER)
10	AUTO-DEC (BEFORE)
11	INDEX

D	
0	-DIRECT
1	-DEFERRED

REG	
000	RØ
001	R1
010	R2
011	R3
100	R4
101	R5
110	SP
111	PC

NOTE: SOURCE AND DESTINATION CODES ARE SIMILIAR.

NOTES:

1. INSTRUCTION LENGTH:
   ONE WORD: OPERATE AND BRANCH ; OTHERS IF SOURCE AND DESTINATION NOT INDEXED OR AUTO-INC ON PC.
   TWO WORDS: EITHER SOURCE OR DESTINATION IS INDEXED BY NEXT WORD OR THE AUTO-INC OF PC PROVIDES AN IMMEDIATE OPERAND. A WORD IS REQUIRED FOR THIS DATA.
   THREE WORDS: BOTH SOURCE AND DESTINATION ARE INDEXED OR HAVE AN IMMEDIATE OPERAND. NEXT WORD AFTER INSTRUCTION REFERS TO SOURCE; NEXT WORD REFERS TO DESTINATION.
2. CODES NOT NOTED ARE RESERVED.
3. RESERVED INSTRUCTION CODES ARE TRAPPED TO LOCATION 1Ø; ILLEGAL INSTRUCTIONS ARE TRAPPED TO LOCATION 4.
4. TRAP: ST AND PC ARE PUSHED ON SP. NEW PC AND ST FROM TRAP LOCATIONS.

OPERATE GROUP

GROUP CODE	SG	OP	X
15 9	8	7 5	4 0

GROUP CODE	SG	OP	X	INSTRUCTION
0 000 000	0	00 0	00 000	HALT
			001	WAIT
			010	RTI : POP PC, ST OFF LP
			011	TPT : TRAP TO 14
			100	IOT : TRAP TO 2Ø
			101	RESET: EXTERNAL INIT.
			REMAINDER	RESERVED
		01	DESTINATION	JMP : DE → PC (M=00, D=0 ILLEGAL)
		10 0 00	REG 2 0	RTS : REG → PC $(LP)^{\uparrow} \rightarrow$ REG
			REMAINDER	RESERVED
		10 1	S N Z V C 4 3 2 1 0	MICRO-PROGRAMMED OPERATES ON CONDITION CODES
		11	DESTINATION	SWAB: $DE_{15/8} \rightarrow DE_{7/0}$ $DE_{7/0} \rightarrow DE_{15/8}$
0 000 000	1		OFFSET	BRANCH ALWAYS (BR) OFFSET IS ±128 WORDS

S=0, CLEAR
S=1, SET

BRANCH ON CONDITION

GROUP CODE	T	OFFSET
15 9	8	7 0

GROUP CODE	COND.
0 000 001	=
0 000 010	<
0 000 011	≤
1 000 000	N
1 000 001	Z+C
1 000 010	V
1 000 011	C

T	OFFSET
0-FALSE	±128 WORDS
1-TRUE	

OPERATION OF INSTR.
COND. MET: PC+OFFSET→PC

SUBROUTINE CALL

GROUP CODE	REG	DESTINATION
15 9	8 6	5 0

GROUP CODE	REG	DESTINATION
0 000 100	XXX	XXX XXX

JSR : REG→-(LP), PC→REG, DE→PC
(M=00, D=0-ILLEGAL)

TRAPS

GROUP CODE	
15 8	7 Ø

GROUP CODE		
1 000 100	1	: TRAP, USER TRAP TO 34
1 000 100	0	: EMT, SYSTEM TRAP TO 3Ø

UNARY GROUP

B	GROUP CODE	OP	DESTINATION
15	14 9	9 6	5 0

GROUP CODE	OP	INSTR.	DESTINATION
000 101	000	CLR: 0→D	AS NOTED ABOVE
	001	COM: $\overline{D} \rightarrow D$	
	010	INC : D+1→D	
	011	DEC : D-1→D	
	100	NEG : $\overline{D}+1 \rightarrow D$	
	101	ADC: D+C→D	
	110	SBC: D-C→D	
	111	TST : TEST D-0	

B	
0	-WORD
1	-BYTE

ROTATE/SHIFT

B	GROUP CODE	0	S	L	DESTINATION
15	14 9	8	7	6	5 0

GROUP CODE	L	DESTINATION
000 110	0 - RIGHT	AS NOTED ABOVE
	1 - LEFT	

B	S
0-WORD	0 - ROTATE
1-BYTE	1 - SHIFT

X	000 110	1	RESERVED
X	000 111	X	RESERVED

REV 2	JFOL	7/29/69
3	RCC	9/3/69