Mini-Computer PDP-8 ISA Simulator Design and Verification

Tarek Elarabi, Ranjith Kumar, Rajath Mavathur Basavaraj

Electrical and Computer Engineering Department
Penn State Behrend, USA.

***Abstract* - This paper introduces a simple instruction set architecture simulator for the PDP-8 mini-computer. It generates a memory trace file from an assembled input object file. The memory trace file indicates whether the access was an instruction fetch, data read or data write. The simulator supports all instructions except for input/output and group 3 microinstructions. A summary of the total number of instructions executed, clock cycles consumed and instruction count by mnemonic is outputted at the end of simulation. A branch trace file is generated listing instruction branches by program counter, branch type, target address and whether taken or not.**

Keywords - PDP-8 architecture, PDP-8 simulation, micro-architecture, ISA simulator.

I. INTRODUCTION

The simulator is implemented in Verilog and accepts an input object file in ASCII hexadecimal format readable by $readmemh (start address at 2008). The object file specified on the simulation command line is read into a memory array. A main while loop is used to fetch, decode and execute each instruction. Each macro instruction is decoded by opcode and executed within a matching case item. The effective address is calculated in the addressCalc task from the mode and offset fields for each instruction. The data operand is then read from or written to memory, and the instruction executed updating processor state. The simulator continues to loop and fetch instructions until a HLT instruction is encountered.

The pgmCntr variable is incremented at the beginning of the loop to point to the next instruction. During the loop execution cpma (central processor memory address) represents the current value of the program counter. It is the value used to reference memory operands and may change during the execution depending on the function of the instruction. At the end of the loop, cpma is updated to the pgmCntr or target address.

The memory and branch trace files are updated on the fly as each instruction is executed. The readMem and writeMem tasks log the memory trace file entries. And the branchTrace task logs the branch trace file entries. Upon simulation completion the instruction statistics are output to the transcript. Unsupported instructions are treated as NOPs, but included in the instruction/mnemonic counts.

The program counter, instruction register, link bit and accumulator values following each instruction can be output with an enabling define entered on the Verilog compiler command line. A memory dump of valid locations before and after simulation is also provided. The simulator behaviour is implemented as specified in existing PDP-8 functional documentation.

II. STANDARD PDP-8 FUNCTIONAL COMPONENTS

The standard PDP-8, in any physical arrangement, involves of a processor, core memory, operator console, input/output facilities, and a Teletype input/output device. The standard PDP-8 constitutes a complete computer system capable of performing general purpose computations and/or control functions. Expansion of the system by addition of optional equipment can be accomplished to perform a specific function or to serve in a particular application.

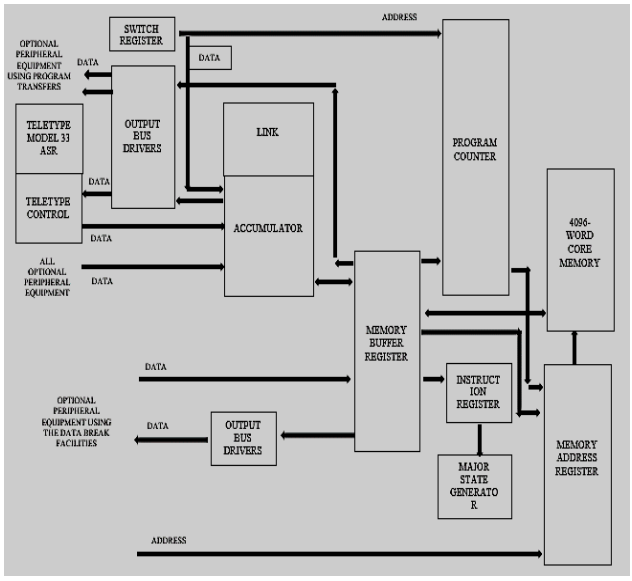

Fig. 1. Standard PDP-8 Block Diagram [1].

PROCESSOR: All logic and arithmetic operations, and control functions are performed by the processor. The major registers and control elements are as follows:

Accumulator (AC): The AC performs the major arithmetic and logic operations and serves as the data input/output register. The AC also serves as a digital buffer for the possible Analog-to-Digital Converter. [1]

Link (L): This one-bit register serves as an extension of the AC. The content of this register can be program sampled and program modified. Overflow into the L from

DOI 10.1109/UKSim.2018.00042

the AC can be checked by the program to greatly speed up single and multiple precision arithmetic routines [1].

Program Counter (PC): The PC determines the core memory address from which the next instruction of a stored program is to be taken. The sequence in which instructions are performed is called program control, and is determined by the PC [1].

Memory Address Register (MA): The vicinity in core memory which is selected for data storage or recovery is determined by the MA. This register can directly address all 4096 words of the standard core memory or in any preselected field of extended core memory [1].

Memory Buffer Register (MB): The Memory Buffer serves as a buffer register for all data passing between the processor, the core memory, serves as a storage directly between core memory, and other external devices or equipment during data break information transfers. The MB is used as a distributor shift register for the Analog-to-Digital Converter [1].

Instruction Register (IR): When an instruction word is read from core memory the three most significant bits (the operation code of all types of instructions) are loaded into the IR. The IR, then decodes the three-bit operation code and determines the basic functions to be performed by the computer, and determines the use to be made of the least significant nine bits of the instruction [1].

Major State Generator: One or more main control states are entered to control and execute an instruction. During any one instruction, a state lasts for one computer cycle, or 1.6 microseconds. The main state originator determines the machine state during each and every cycle as a task of the current instruction, the present state, and the situation of the Break Request signal supplied to an input bus by peripheral equipment [1].

Switch Register (SR): Twelve toggle switches on the operator console provide a means of manually establishing a word to be set into the computer. The content of the SR can be transferred into the PC as an address by pressing the load key, or can be stored in core memory at the address contained in the PC by pressing the deposit key [1].

Output Bus Drivers: All major output signals from the standard PDP-8, used in programmed and data break information transfers, are power amplified by bus driver modules to allow them to drive a very heavy circuit load. [1]

CORE MEMORY: Storage for instructions to be performed, or for data to be processed or distributed is provided by the core memory. Core memory for the standard PDP-8 has a capacity of 4096 words, 12 bits in length [1].

OPERATOR CONSOLE: Keys, switches, indicators, and the switch register on the operator console allow manual address and data storage, core memory data examination, the normal start/stop/continue control, and single-step or single-instruction operation that allows each step or each instruction of the program to be monitored visually for demonstration or maintenance.

III. STANDARD PDP-8 INSTRUCTIONS

Instruction words are of two types: memory reference and augmented. Memory reference instructions store or retrieve data from core memory, while augmented instructions do not. All instructions utilize bits 0, 1, and 2 to specify the operation code. Operation codes of O8 through 58 specify memory reference instructions and codes of 68 and 78 specify augmented instructions.

Memory reference instructions: Since the standard PDP-8 system contains a 4096-word core memory, 12 bits are required to address all locations. To simplify addressing, the core memory is divided into blocks, or pages, of 128 words (ZOO8 address), A memory reference instruction can directly address any one of 256 words; 128 words on page 0 or 128 words on the current page. All other core memory locations are addressed indirectly [1].

Input/ Output Transfer Instructions: Microinstructions of the IOT instruction addresses peripheral equipment to transfer information with it or to initiate operations within it. This instruction contains a 6-bit device selection code and three bits which can be programmed to produce pulses in the selected device that effect a transfer or initiate an operation [1].

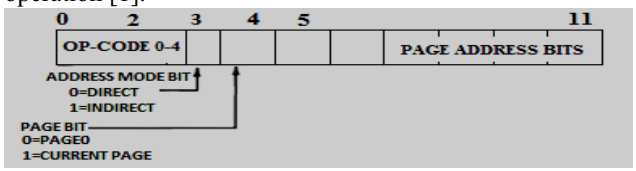

Fig. 1. Memory reference instructions Format [1].

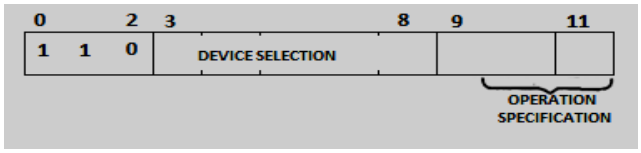

Fig. 2. IOT Instruction Format [1].

Operate Instructions: The OPR instruction consists of two basic groups of microinstructions. When bit 3 contains a binary zero, group OPR 1 is designated and the content of bits 4 through 11 is decoded to cause clear, complement, rotate, and increment operations. When bit 3 contains a binary one, group OPR 2 is designated and the content of bits 4 through 11 is decoded to cause operations that check the content of the AC and the L and continue to, or skip, the next instruction based on the results of the check. Any logical combination of bits within one group can be microprogrammed into one instruction [1].

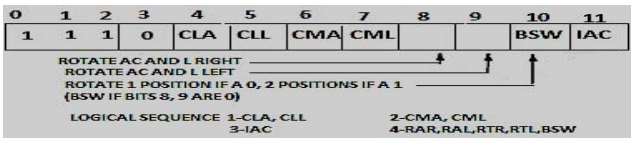

Fig. 3. Group1 Operate Microinstruction Format [1].

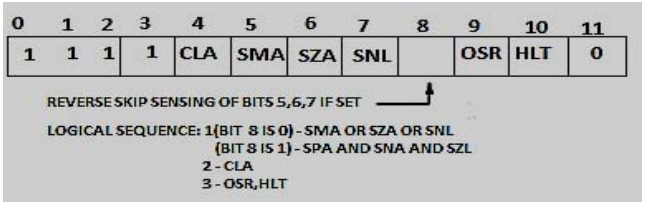

Fig. 4. Group2 Operate Microinstruction Format [1].

IV. TEST PLAN

The following describes the testing strategy used to verify the simulator function:

- Test each individual memory reference instruction. Create an individual test for each that targets all functional aspects as follows:

a) TAD: Validating two's complement add of accumulator and memory data in accumulator result. Verify that a carry out of accumulator MSB inverts the Link Bit [2].
b) JMS: Validating that the program counter value is stored in the referenced memory location. And that the program counter is set to the referenced memory location + 1 [2].
c) ISZ: Validating that referenced memory location data is increased by 1 in value. Verify that a resulting value of 0 causes the program counter to be incremented by 1 [2].
d) JMP: Validating that the program counter is set to the referenced memory location [2].
e) DCA: Validating that the accumulator is stored to the referenced memory location and that after the store the accumulator value is 0 [2].
f) AND: Validating bit-wise AND operation of accumulator and memory data in accumulator result [2].

Test each of the addressing modes with AND or TAD, ISZ and JMS. Create a test that targets the following:

a) Zero page: Verify that the instruction references memory correctly in the lowest page.
b) Current page: Verify that the instruction references memory correctly in the page where the instruction is located.
c) Indirect zero page: Verify that the instruction references memory pointed to by the contents of a reference location in the lowest page.
d) Indirect current page: Verify that the instruction references memory pointed to by the contents of a reference location in the current page.
e) Auto indexing: Verify that an indirect memory reference to locations 0010-00178 causes the contents to be incremented by 1, and used as the reference memory address for the instruction.

- Test all valid combinations of the OPR instructions (micro instructions). Create tests that target the following:(#) denotes sequence number.

TABLE I.1. AND – DIRECT MODE WITH ZERO PAGE ADDRESSING

Test Name	Address Mode	Tested Instructions	Expected Final Results
AND_I0_P0	Direct Zero Page	AND	AC= 0 Out0 = 0403 Out1 =1023 Out2 =4220 Number of instructions executed = 11 Number of clock cycles = 20 Number of AND = 3 Number of TAD = 3 Number of DCA = 3 Number of OPR = 2

Group 1

NOP:	do nothing (except increment pc)	
CLA:	clear the accumulator	(1)
CLL:	clear the link bit	(1)
CMA:	complement the accumulator	(2)
CML:	complement the link bit	(2)
IAC:	increment the accumulator	(3)
RAR:	rotate accumulator and link right	(4)
RTR:	rotate accumulator and link right twice	(4)
RAL:	rotate accumulator and link left	(4)
RTL:	rotate accumulator and link left twice	(4)

Group 2

SMA:	skip on minus accumulator	(1)
SZA:	skip on zero accumulator	(1)
SNL:	skip on nonzero link bit	(1)
SPA:	skip on plus accumulator	(1)
SNA:	skip on nonzero accumulator	(1)
SZL:	skip on zero link bit	(1)
SKP:	skip always	(1)
CLA:	clear accumulator	(2)
HLT:	halt program	(3)

- Using tests above verify correct instruction count, clock cycles and mnemonic count for all macro instructions. Verify that AND, TAD, ISZ, DCA and JMS instructions take 2 cycles, JMP and OPR take 1 cycle and IOT takes 0 cycles. Verify that indirect address mode adds 1 cycle and auto indexing adds 2 cycles for memory reference instructions.

- Verify trace file output for ISZ & JMS instructions, and direct, indirect and auto indexing address modes. Check that all instruction fetches are listed, all data reads are listed and all data writes are listed.

- Test branch trace file output for ISZ, JMS, JMP instructions and all the group 2 skip instructions. Verify correct program counter, branch type (conditional, unconditional, subroutine call) and whether the branch was taken or not for all encountered branches.

V. TEST CASES AND RESULTS

The errors that were discovered are fixed during design and test execution. A specific. mem file is provided as input to the simulator in each test case, after being converted from assembly level code [3] [4] [1] using pal command in windows command prompt. The simulator then takes the input object file [3] [5] containing hexadecimal values which was converted, to display following results.

Test case 1:

In this program input operands A0=04378, A1=10238, A2=63208, B0=06438, B1=77338, B2=42368 are fetched and then AND operation is between them. The result is stored in Out0, Out1, Out2 variables and Accumulator is cleared. The transcript result is shown in Table I.1.

a) Using Direct mode with Zero-page addressing. Outputs are stored at location starting from 00708.

b) Using Direct mode with Current page addressing. Outputs are stored at location starting from 02708.The transcript result is shown in Table I.2.

TABLE I.2. AND - DIRECT MODE WITH CURRENT PAGE ADDRESSING

Test	**Address Mode**	**Tested Instructions**	**Expected Final Results**
AND_I0_P1	Direct Current Page	AND	AC= 0 Out0 = 0403 Out1 =1023 Out2 =4220 Number of instructions executed = 11 Number of clock cycles = 20 Number of AND = 3 Number of TAD = 3 Number of DCA = 3 Number of OPR = 2

Test case 2:

In this program input operands A0=34278, A1=77778, A2=63208, A3=11118, B0=22658, B1=00018, B2= 42368, B3= 76548 are fetched and then TAD operation is between them. The result is stored in Out0, Out1, out 2, Out 3 variables and Accumulator is cleared. The transcript result is shown in Table II.1.

a) Using Direct mode with Zero-page addressing. Outputs are stored at location starting from 00708.

b) Using Indirect mode with Zero-page Auto indexing. Outputs are stored at location starting from *00128. The transcript result is shown in Table II.2.

TABLE II.1. TAD - DIRECT MODE WITH CURRENT PAGE ADDRESSING

Test Name	Address Mode	Tested Instructions	Expected Final Results
TAD_I0_P0	Direct Zero Page	TAD Link Bit	AC= 0 Out0 = 5714 Out1 =0000 Out2 =2556 Out3 =0765 Link bit = 1 Number of instructions executed = 14 Number of clock cycles = 26 Number of TAD = 8 Number of DCA = 4 Number of OPR = 2

TABLE II.2. TAD - INDIRECT MODE WITH ZERO PAGE ADDRESSING

Test	Address Mode	Tested Instructions	Expected Final Results
TAD_I1_P0_A1	Indirect Zero Page Auto indexing	TAD Link bit	AC= 0 Out0 = 5714 Out1 =0000 Out2 =2556 Out3 =0765 Link bit = 1 Number of instructions executed = 14 Number of clock cycles = 50 Number of TAD = 8 Number of DCA = 4 Number of OPR = 2

Test case 3:

In DCA direct program, input values were taken as A=18, B=28, C=38 which are stored in current page starting at location 02608 and sum of these registers is stored at D. The value of D is restored into the Accumulator. The transcript result is shown in Table III.1.

TABLE III.1. DCA –DIRECT

Test	Address Mode	Tested Instruction	Expected Final Results
DCA_P1	Direct Current Page	DCA	AC=6 Number of instructions execute = 7 Number of clock cycles = 12 Number of TAD = 4 Number of DCA = 1 Number of OPR = 2

TABLE III.2. DCA– INDIRECT

Test Name	Address Mode	Tested Instruction	Expected Final Results
DCA_I1_P1	Indirect Current Page	DCA	AC=6 Number of instructions executed = 7 Number of clock cycles = 17 Number of TAD = 4 Number of DCA = 1 Number of OPR = 2

In DCA indirect program, input values were taken as A=18, B=28, C=38 which are stored in current page starting at location *02608 and sum of these registers is stored at D. The value of D is restored into the Accumulator. The transcript result is shown in Table III.2.

Test case 4:

The JMS direct program subtracts two numbers A=178 and B=228 which are stored in page zero starting at location 00508; Using subroutine. The output value is negated and then stored in the Accumulator. The transcript result is shown in Table IV.1.

TABLE IV.1. JMS – DIRECT

Test	Address Mode	Tested Instructions	Expected Final Results
JMS_P0	Direct Zero Page	JMS SMA CMA IAC Branch type	AC= 3 CMA AC= 0022 => AC= 7755 IAC AC =7755 => AC= 7756 Number of instructions executed = 12 Number of clock cycles = 17 Number of TAD = 2 Number of DCA = 1 Number of JMS = 1 Number of JMP = 1 Number of OPR = 7

TABLE IV.2. JMS – INDIRECT

Test	Address Mode	Tested Instructions	Expected Final Results
JMS_I1_P0	Indirect Zero Page	JMS SMA CMA IAC Branch type	AC= 3 CMA AC= 0022 => AC= 7755 IAC AC =7755 => AC= 7756 Number of instructions executed = 12 Number of clock cycles = 20 Number of TAD = 2 Number of DCA = 1 Number of JMS = 1 Number of JMP = 1 Number of OPR = 7

The JMS indirect program subtracts two numbers A=178 and B= 228 which are stored in page zero starting at location *00408; Using subroutine. The output value is negated and then stored in the Accumulator. The transcript result is shown in Table IV.2.

Test case 5:

The JMP direct program subtracts two numbers A=178 and B=228 which are stored in page zero starting at location 00508. The output value is then stored in the Accumulator. The transcript result is shown in Table V.1.

TABLE V.1. JMP - DIRECT

Test Name	Address Mode	Tested Instructions	Expected Final Results
JMP_P0	Direct Zero Page	JMP CMA IAC	AC= 7775 CMA 22; AC= 7755 IAC 7755; AC= 7756 Number of instructions executed=7 Number of clock cycles = 10 Number of TAD = 2 Number of DCA = 1 Number of OPR = 4

The JMP indirect program subtracts two numbers A=178 and B=228 which are stored in current page starting at location *02358. The output value is then stored in the Accumulator. The transcript result is shown in Table V.2.

TABLE V.2. JMP – INDIRECT

Test	Address Mode	Tested Instructions	Expected Final Results
JMP_I1_P1	Indirect Current Page	JMP CMA IAC	AC= 7775 CMA 22; AC= 7755 IAC 7755; AC= 7756 Number of instructions executed = 7 Number of clock cycles = 13 Number of TAD = 2 Number of DCA = 1 Number of OPR = 4

Test case 6:

Description: This program verifies the group 1 micro instructions. It is also used to verify accumulator and link bit operation. Refer to the assembly listing for line –by-line expected results. The output value is then stored in the Accumulator. The transcript result is shown in Table VI.

TABLE VI. GROUP1 INSTRUCTIONS

Test	Tested Instructions	Expected Final Results
group1	CLA CLL CMA CML IAC RAR RTR RAL RTL NOP TAD	Link bit = 0, AC=0 Number of instructions executed = 33 Number of clock cycles = 38 Number of TAD = 5 Number of OPR = 28

Test case 7:

The group 2 SKIP program verifies group 2 micro instructions, not including the conditional skips. Refer to the assembly listing for line –by- line expected results. The transcript result is shown in Table VII.

TABLE VII. GROUP 2 SKIP INSTRUCTIONS

Test	Tested Instructions	Expected Final Results
group2skp	SKP CLA OSR HLT	Link bit = 0, AC=0 Number of instructions executed = 7 Number of clock cycles = 11 Number of TAD = 3 Number of DCA = 1 Number of OPR = 3 Unconditional branches taken = 1

Test case 8:

The group 2 OR program verifies the group 2 OR skip micro instructions. Refer to the assembly listing for line –by- line expected results. The transcript result is shown in Table VIII.

TABLE VIII. GROUP 2 OR INSTRUCTIONS

Test Name	Tested Instructions	Expected Final Results
group2or	SMA, SZA, SNL	Link bit = 1, AC=2 Number of instructions executed = 35 Number of clock cycles = 46 Number of TAD = 11 Number of JMP = 5 Number of OPR = 19 Conditional branches taken = 3 Conditional branches not taken = 5 Unconditional branches taken = 5

Test case 9:

The group 2 AND program verifies the group 2 AND skip micro instructions. Refer to the assembly listing for line –by- line expected results. The transcript result is shown in Table IX.

TABLE IX. GROUP2 AND INSTRUCTIONS

Test Name	Tested Instructions	Expected Final Results
group2and	SPA, SNA, SZL	Link bit = 1, AC=0 Number of instructions executed = 55 Number of clock cycles = 76 Number of TAD = 21 Number of JMP = 11 Number of OPR = 23 Conditional branches taken = 7 Conditional branches not taken = 9 Unconditional branches taken = 12

TABLE X. TRACE AND BRANCH FILE OUTPUT

Test Name	Tested Instructions	Expected Final Results
trace	AND, TAD, ISZ, DCA,JMS,JMP	Number of instructions executed = 43 Number of clock cycles = 104 Number of AND=1, TAD =9, ISZ=9, DCA=9 Number of JMS=2, JMP=8, OPR = 5 Subroutine calls = 2 Conditional branches taken = 3 Conditional branches not taken = 6 Unconditional branches taken = 8

Test case 10:

The TRACE and BRANCH program is used to verify the trace and branch file output. It also uses all 5 addressing modes, and all memory reference instructions. Refer to the assembly listing for line –by- line expected results. The transcript result is shown in Table X.

VI. CONCLUSION

The PDP-8 is a 12bit single accumulator machine which to be used to address up to 32K, 12 bit words. It has 8 basic instructions and executes them in 1.2 microsecond for simple instructions and 4 microsecond for complex memory level reference instructions. In our design we were able to achieve it in 1.2 microsecond in microarchitecture. We are estimating that this design will give the machine about a 5 MIPS rating. Although this speed of the processor might seem to be slow compared to that of modern standards, but it is a very good start as this paper showcase the efficiency of the design through some of the test cases on memory reference, group1 and group2 instructions. The test results are verified and the design can be explored more with regression tests conducted with various probability scenarios. This will be the gateway for a wide horizon of prospects for future work to verify the results in micro-architecture level through ISA simulator and start aiming to have a better processor for future endeavour.

REFERENCES

[1] Digital equipment co-orporation, PDP8_Users Handbook. Maynard, Massachusetts, 1966

[2] Digital equipment co-orporation, MACRO-8 Programming Manual. Maynard, Massachusetts, 1969

[3] Shelburne B “Teaching computer organization using a pdp-8 simulator”, 34th SIGCSE Technical Symposium on Computer Science Education, 69-73.

[4] Shelburne B “A pdp-8 emulator program. Journal on Educational Resources in Computing” (JERIC), 2(1), 17-47.

[5] Wolffe Greg, Yurcik William, Osborn Hugh, Holliday Mark “Teaching Computer Organization/Architecture With Limited Resources Using Simulators” SIGSCE Bulletin 34, 176 - 180.