PennState

Mini-Computer PDP-8 ISA

Simulator Design and Verification

Students Authors:

Rajath Mavathur Basavaraj

Ranjith Kumar M

Supervised By:

Dr. Elarabi

PennState Behrend

Why ISA simulator

- Simulation at the ISA -Instruction Set Architecture - level is particularly useful for Computer Architects and designers.
    - Without needing to simulate at a deeper machine organization or register level, we can explore the impact of instruction set architecture changes on CPI.

Motivation

- The process of learning , understanding and implementing an Instruction Set Architecture would improve an individual’s knowledge and create a new and efficient ISA, to improve with existing architectures.
- The standard PDP is a complete hardware and software system ready to perform general purpose computations like **mini computer**.
- Simple ISA and simple constraints in Programmed Data Processor

PennState
Behrend

Introduction: PDP ISA

- Fully Parallel Arithmetic Operations.
- Continues Display of Active registers like Accumulator, Link, Memory Buffer Register.
- Major state generator are continuously indicated.
- Micro programmable Instructions.
- Bus I/O system.
- Buffered Input /Output.

PDP ISA Functional Components:

Addressing modes of PDP-8 and effective address calculation in the following addressing modes.

(a) Zero page addressing
(b) Current page addressing
(c) Indirect addressing
(d) Auto indexing

PDP ISA Functional Components

A. Memory reference instructions :
AND, TAD, ISZ, DCA, JMS, JMP

B. Micro-instructions:
(i) Group-1: NOP, CLA, CLL, CMA,CML, IAC, RAR, RTR, RAL, RTL
(ii) Group-2: SMA, SZA, SNL, SPA, SNA, SZL, SKP, CLA, OSR, HLT

C. Registers (Accumulator, Program Counter), Flags (Link) and their behavior in all instructions

PDP ISA Instructions

Group 1 NOP: do nothing (except incr 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)

A Functional Components

- **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)
    - Link (L),
    - Program Counter (PC)
    - Memory Address Register (MA)
    - Memory Buffer Register (MB)
    - Instruction Register (IR)
    - Major State Generator
    - Switch Register (SR)
    - Output Bus Drivers
- CORE MEMORY Capacity of 4096 words, 12 bits in length
- OPERATOR CONSOLE: Allows manual address and data storage

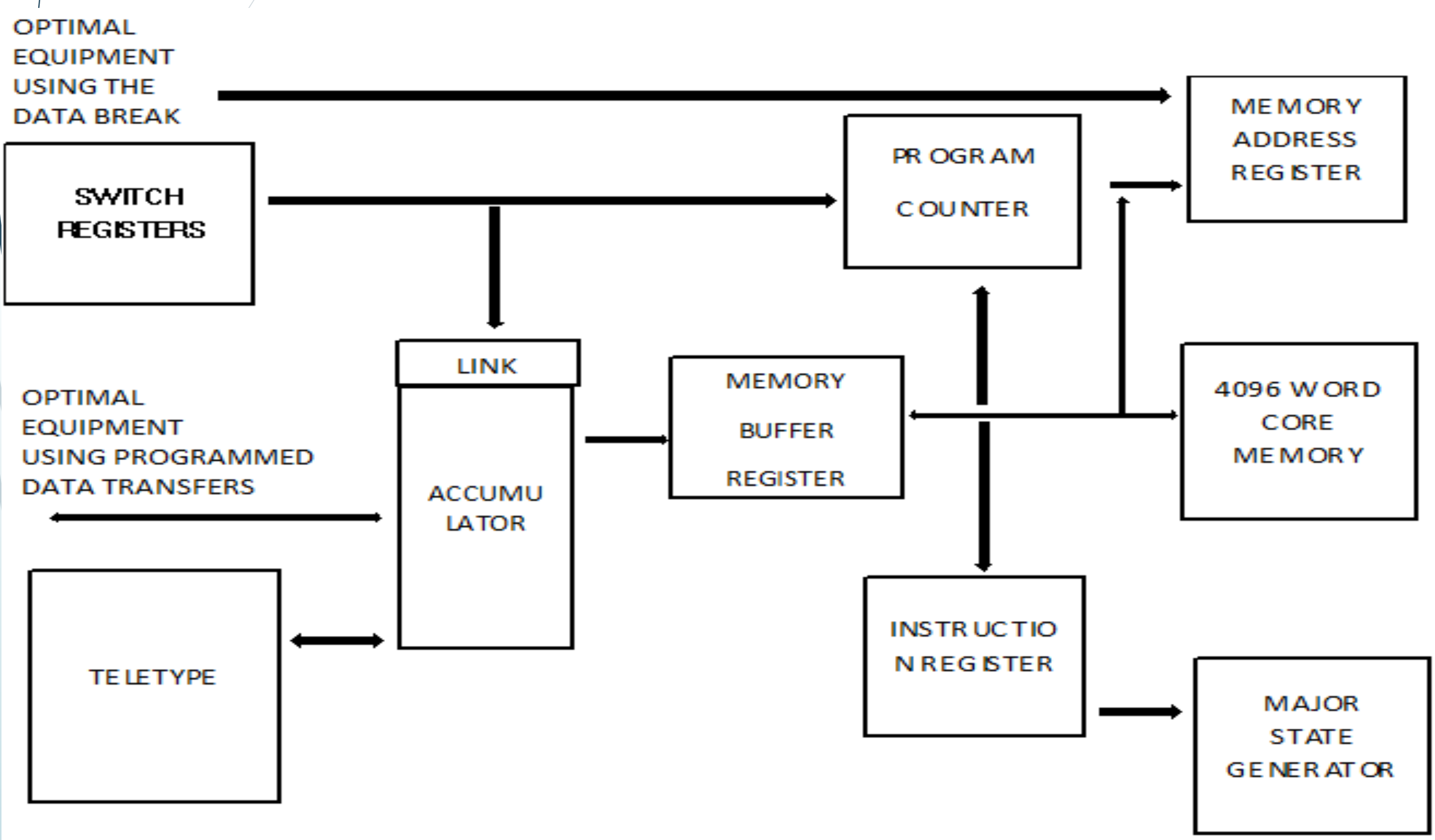
Research plan

I. To implement the PDP-8 instruction set and validate them and recognize any constraints and find optimum solutions to improvise the constraints.

II. To Implement 12 bit Accumulator to access the address.

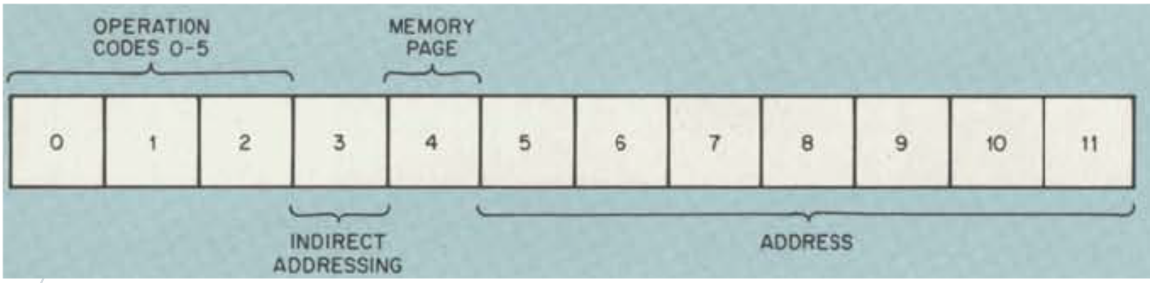
III. To execute the simple and complex instructions faster than existing architecture.

PDP-8 ISA IMPLEMENTATION

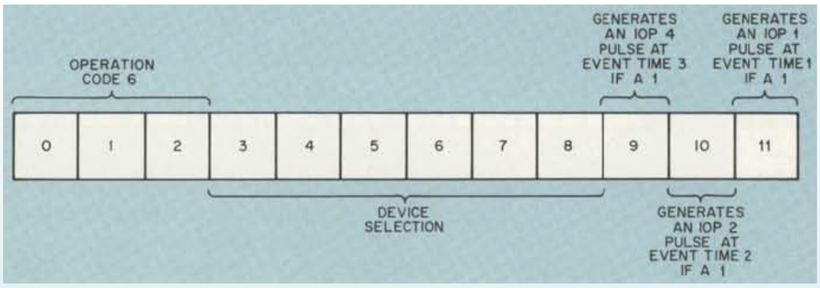


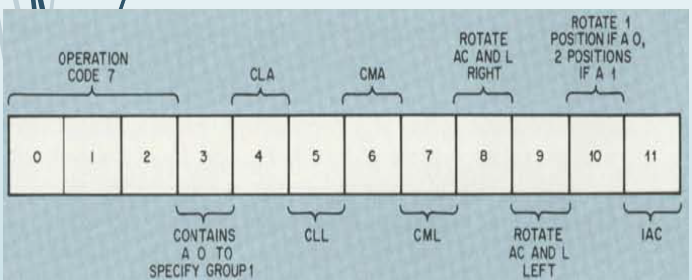
Instruction Format:

Memory reference instruction format

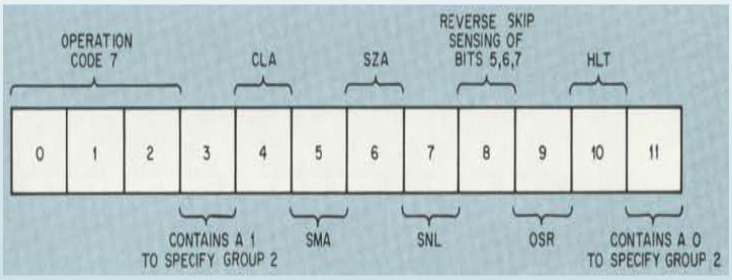


IOT instruction format





Group1Operate instruction format



Group 2 Operate instruction format

Implementation

The simulator is implemented in **Verilog** and accepts an input object file in **ASCII hexadecimal** format.

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.

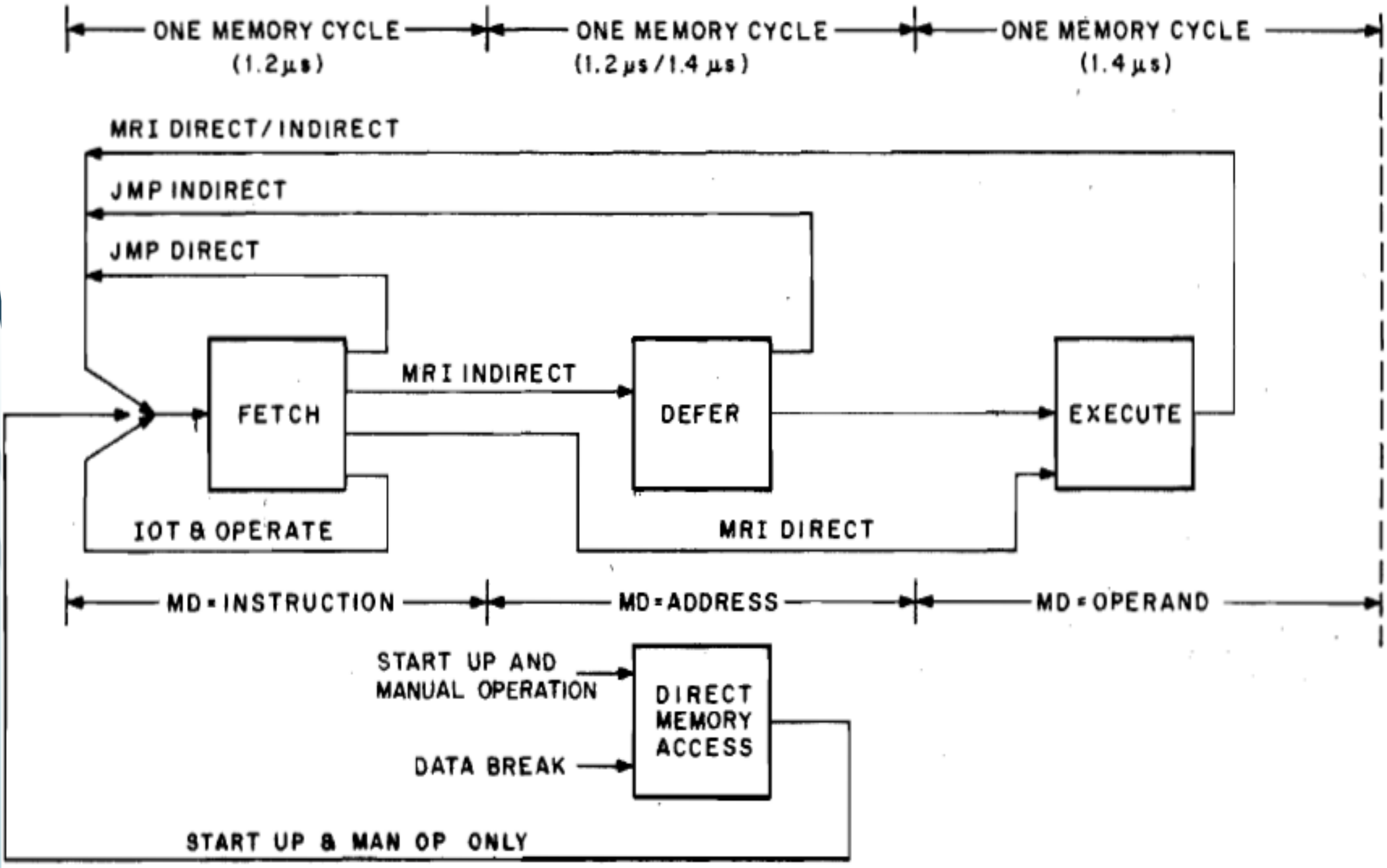
Implementation

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.

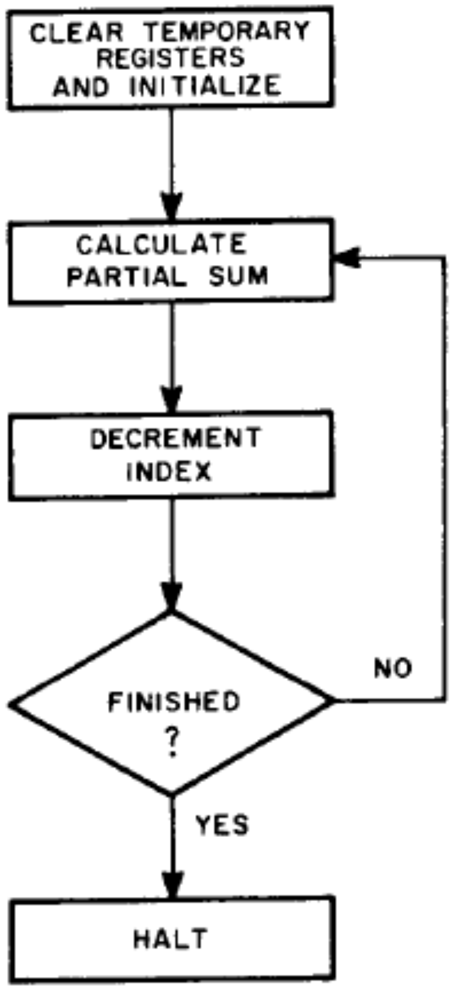
The memory and branch trace files are updated on the fly as each instruction is executed. The **readMem** and **writeMem** tasks log the memory trace file entries. And the **branchTrace** task logs the branch trace file entries. Upon simulation completion the instruction statistics are output to the 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.

Implementation Block Diagram



Implementation flow



Simulation Results

PennState Behrend

<u>Case 1</u>:

Input operands A0=04378, A1=10238, A2=63208, B0=06438, B1=77338, B2=42368 are fetched and then AND operation. Result is stored in Out0, Out1, Out 2 variables and Accumulator is cleared

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

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

Using Direct mode with Current page addressing. Outputs are stored at location starting from 02708

Test Name	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 = 21 Number of AND = 3 Number of TAD = 3 Number of DCA= 3 Number of OPR = 2

Simulation Results

rent page addressing.

Outputs are stored at location starting from $*0270_8$

Test Name	Address Mode	Tested Instructions	Expected Final Results
AND_I1_P1	Indirect Current Page	AND	AC= 0 Out0 = 0403 Out1 =1023 Out2 =4220 Number of instructions executed = 11 Number of clock cycles = 29 Number of AND = 3 Number of TAD = 3 Number of DCA= 3 Number of OPR = 2

Test Name	Address Mode	Tested Instructions	Expected Final Results
AND_I1_P0_A1	Indirect Zero Page Auto indexing	AND	AC= 0 Out0 = 0403 Out1 =1023 Out2 =4220 Number of instructions executed = 11 Number of clock cycles = 38 Number of AND = 3 Number of TAD = 3 Number of DCA= 3 Number of OPR = 2

Using Indirect mode with Zero page Auto indexing

Outputs are stored at location starting from $*0012_8$

PennState Behrend

Simulation Results

Case 2: 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. Result is stored in Out0, Out1, Out 2, Out 3 variables and Accumulator is cleared.

Using Direct mode with Zero page addressing.
Outputs are stored at location starting from 0070_8

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

Using Indirect mode with Zero page Auto indexing
Outputs are stored at location starting from $*0012_8$

Test Name	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

Simulation Results

Test Name	Address Mode	Tested Instructions	Expected Final Results
ISZ_P1	Direct Current Page	ISZ	AC=10 Number of instructions executed = 18 Number of clock cycles = 29 Number of TAD = 5 Number of ISZ = 5 Number of DCA = 1 Number of JMP = 4 Number of OPR = 3

Case 3: Description: This program adds the number A= 2 for given counter bit =5. Input values stored in current page at location 0250_8.

Test Name	Address Mode	Tested Instructions	Expected Final Results
ISZ_I1_P0	Indirect Zero Page	ISZ	AC=10 Number of instructions executed = 18 Number of clock cycles = 40 Number of TAD = 5 Number of ISZ = 5 Number of DCA = 1 Number of JMP = 4 Number of OPR = 3

Description: This program adds the number A= 2_8 for given counter bit =5. Input value stored in Zero page starting from location *0070_8.

PennState Behrend

Simulation Results

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

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

Case 4: **In this program we take the input values as A=1_8, B=2_8, C=3_8 which are stored in current page starting at location 0260_8 and sum of these registers is stored at D. The value of D is restored into the Accumulator.**

Description: In this program we take the input values as A=1_8, B=2_8, C=3_8 which are stored in current page starting at location *0260_8 and sum of these registers is stored at D. The value of D is restored into the Accumulator.

PennState Behrend

Simulation Results

Test Name	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

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.

Test Name	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

Description: This program verifies group 2 micro instructions, not including the conditional skips. Refer to the assembly listing for line –by-line expected results.

PennState Behrend

Conclusion

- The PDP-8 is a 12bit single accumulator machine which can 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 simulator. We are estimating that this design will give the machine about a .5 MIPS rating.
- The test results are verified and the design can be explored more with regression tests conducted with various probability scenarios.
- This will be enable us to verify the results in micro-architecture level through ISA simulator and start aiming to have a better processor for future endeavor.

PennState
Behrend

Work in Progress

Improve the overall efficiency of the processor.

Improve its throughput and response time.

Implement multithreading process and pipelining concepts.

References

- Van De Goor, A. , & Bell, C. (1969). A control unit for a dec pdp-8 computer and a burroughs disk. IEEE Transactions on Computers, C-18(11), 1044-1048.
- Maslin, E. , Terry, S. , & Wraight, L. (1969). Use of a pdp-8 computer for operation of several neutron beam experiments. Nuclear Instruments and Methods.
- Van De Goor, A. , Bell, C. , & Witcraft, D. (1969). Design and behavior of tss/8: A pdp-8 based time-sharing system. IEEE Transactions on Computers, C-18(11), 1038-1043
- PDP-8 : http://homepage.cs.uiowa.edu/~jones/pdp8/man/index.html

Thank You