AN EMMY BASED EMULATION OF THE CDC 6000 SERIES CPU

by

Ellard T. Roush

July 1977

Technical Note No. 120

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

The work described herein was supported in part by the Ballistic Missile Defense Systems Command under contract no. DASG60-77-C-0073.

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Technical Note 120

July 1977

AN EMMY BASED EMULATION OF THE CDC 6000 SERIES CPU

by

Ellard T. Roush

ABSTRACT

An emulator for a CDC 6400 computer has been written for the EMMY computer. The full standard instruction set of the CDC 6400 is supported. The optional compare and move unit is not supported. The emulator duplicates precisely those actions that are visible to the user. Certain things that are not visible or accessible, such as the field length register, are not emulated. The emulator causes error halts for the same reasons as the true machine, Infinite and Indefinite operands, address out of bounds, and attempt to execute Location 0.

The work described herein was supported in part by the Ballistic Missile Defense Systems Command under contract no. DASG60-77-C-0073.

1. INTRODUCTION

An emulator for a CDC 6400 computer has been written for the EMMY computer. The full standard instruction set of the CDC 6400 is supported. The optional compare and move unit is not supported. The emulator duplicates precisely those actions that are visible to the user. Certain things that are not visible or accessible, such as the field length register, are not emulated. The emulator causes error halts for the same reasons as the true machine, Infinite and Indefinite operands, address out of bounds, and attempt to execute Location 0.

2. CDC 6000 SERIES COMPUTERS

Control Data Corporation builds three computers, the 6200, 6400 and 6600, that execute the same instruction set with the CDC 6600 having a couple more instructions. (They are now sold as Cyber 70 Models 72, 73 and 74.) Architecturally the machine is separated into a fast CPU that executes the programs, and up to 10 Peripheral Processors that handle I/O and most of the operating system functions.

The machine is based on a 60 bit word. Memory is accessed on a full word basis only. The machine may have up to 18 bits = 256K words of memory. The CDC 6000 series computers have 3 sets of 8 registers. The X-Registers are 60 bits long and are used to manipulate the data. The Address Registers are 18 bits long and contain memory addresses. Setting an address register has certain side effects. Setting A1-A5 causes X1-X5 to get the value of the memory location in the A register. Setting A6-A7 causes a store while A0 has not side effect. The Base registers are also 18 bits long. B0 is permanently set to positive zero. The instruction set contains conditional and unconditional jumps, Boolean

1. INTRODUCTION

An emulator for a CDC 6400 computer has been written for the EMMY computer. The full standard instruction set of the CDC 6400 is supported. The optional compare and move unit is not supported. The emulator duplicates precisely those actions that are visible to the user. Certain things that are not visible or accessible, such as the field length register, are not emulated. The emulator causes error halts for the same reasons as the true machine, Infinite and Indefinite operands, address out of bounds, and attempt to execute Location 0.

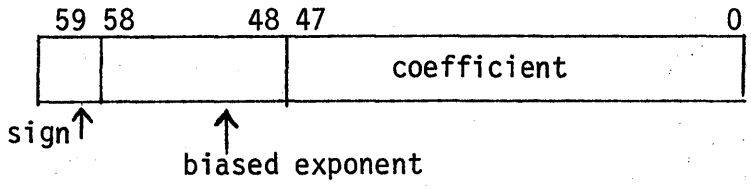
2. CDC 6000 SERIES COMPUTERS

Control Data Corporation builds three computers, the 6200, 6400 and 6600, that execute the same instruction set with the CDC 6600 having a couple more instructions. (They are now sold as Cyber 70 Models 72, 73 and 74.) Architecturally the machine is separated into a fast CPU that executes the programs, and up to 10 Peripheral Processors that handle I/O and most of the operating system functions.

The machine is based on a 60 bit word. Memory is accessed on a full word basis only. The machine may have up to 18 bits = 256K words of memory. The CDC 6000 series computers have 3 sets of 8 registers. The X-Registers are 60 bits long and are used to manipulate the data. The Address Registers are 18 bits long and contain memory addresses. Setting an address register has certain side effects. Setting A1-A5 causes X1-X5 to get the value of the memory location in the A register. Setting A6-A7 causes a store while A0 has not side effect. The Base registers are also 18 bits long. B0 is permanently set to positive zero. The instruction set contains conditional and unconditional jumps, Boolean

operations, shifts, floating point arithmetic, integer add and subtract, normalize, integer-floating point conversions, numerous ways of getting values into the A, B and X registers, plus a few others.

The machine works with one's complement arithmetic. The floating point word consists of a 48 bit coefficient, 1 bit for sign and an 11 bit biased exponent.



The exponent has values reserved for an infinite exponent and an indefinite exponent (zero divided by zero). The exponent is to powers of 2.

3. CDC 6400 EMULATOR

The particular machine that was chosen to emulate was the CDC 6400 CPU. (Only the addition of a couple instructions would be necessary to emulate a CDC 6600 CPU.) The source for details was the Compass Reference Manual. The emulator duplicates the action of a CDC 6400 CPU on correct programs. Error conditions are also set and when a fatal error occurs, (use of Indifinite or Infinite operands in a floating point instruction, address out of bounds or attempt to execute the contents of location zero) the machine is halted. The Peripheral Processors have not been emulated.

The host machine, the Stanford Emmy, has eight 32 bit registers, 4K of 32 bit control store memory and 64K bytes of main memory (8 bits = 1 byte). The Emmy performs arithmetic in 2's complement.

Thus, the CDC 6400 does not map at all well on the host machine. The 60 bit memory words, 60 bit registers, 18 bit registers, and even the one's

complement arithmetic have to be simulated with software.

Emmy main memory can be accessed in 4 byte blocks at a time. 4 bytes = 32 bits so one 60 bit CDC word takes 8 bytes of Emmy main memory and requires 2 Emmy fetches per CDC word. 8 bytes = 64 bits which does not equal 60 bits so even this mapping has overhead. The CDC word is broken up into two 30 bit quantities and each is packed into the high 30 bits of each 32 bit Emmy word. The high order bits of the CDC word (bits 59-30) are located in the low order Emmy word (even address) and the low order bits (29-0) are located in the high order Emmy word (odd address). This means that the 64K bytes of Emmy main memory yield 8K CDC 60 bit words.

Emmy does not have 24 registers so the CDC A, B and X registers are located in Emmy Control Store. The 60 bit X registers are stored in control store the same way that they are in Main memory. The 18 bit Address and Base registers have their bits packed into the high 18 bits of a 32 bit Emmy word.

CDC 6400 Instructions take up either 15 or 30 bits. There can be 2, 3 or 4 instructions per word. Instructions are not allowed to extend across words. This does allow a 30 bit instruction to be in 2 different Emmy words. This means that when an instruction is being processed a flag has to be kept to tell where in the word it is. The instructions are formatted as 6 bits for an opcode and 3 bits each for the I, J and K fields. In a 30 bit instruction the K field is 18 bits long.

The CDC 6400 emulator loops forever through an instruction decode step followed by the execution of that instruction. The emulator stops when the Program Stop instruction is executed or a fatal error occurs.

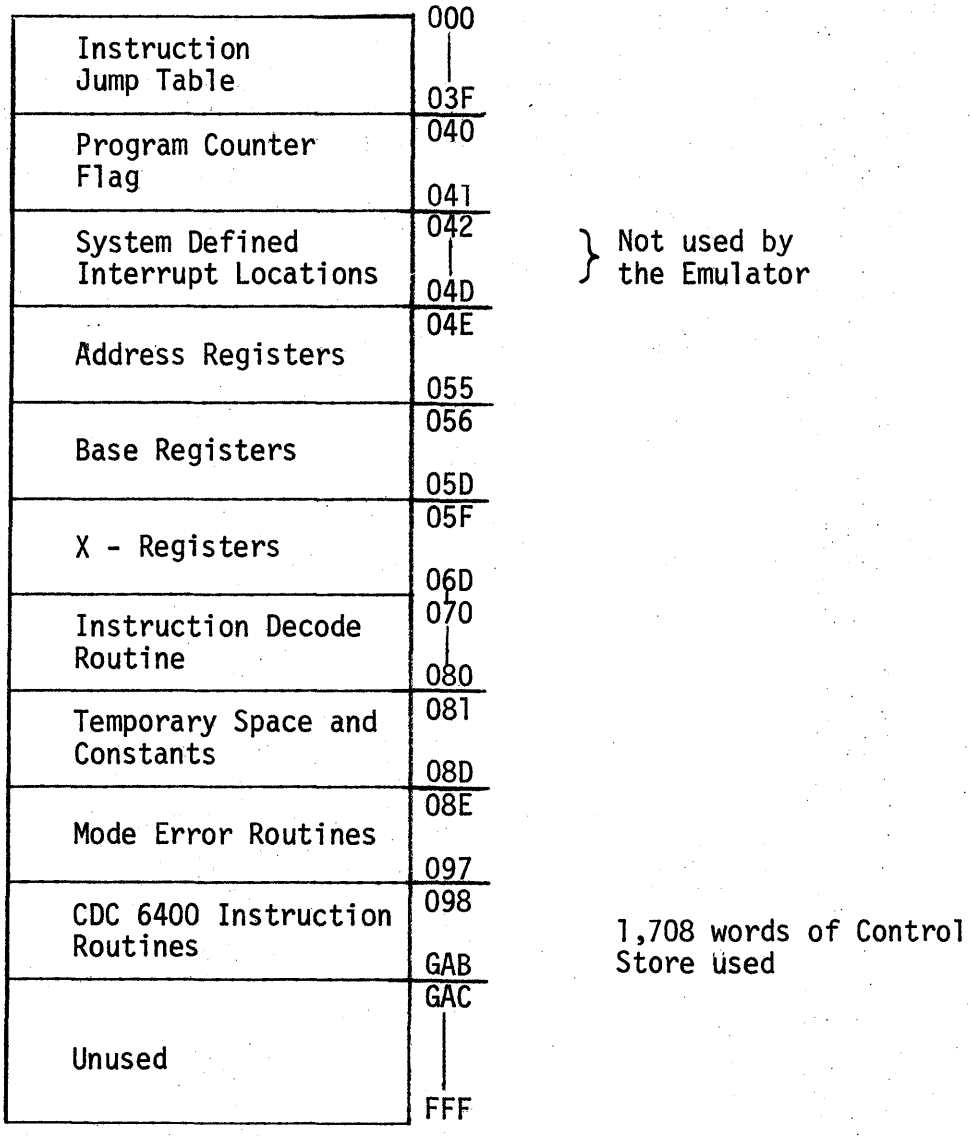
The Instruction routine fetches the instruction word if necessary. It sets the flag telling where in the word the instruction is from and maintains the Program Counter. It tests for address out of bounds and for an attempt to execute

location zero. Next it strips the Opcode, I, J and K fields and places them in separate registers. Then it does a 64-way branch to the routine that executes the instruction. In only one case does it have to do a further decode to get to the particular instruction. The conditional jump on an X register has 8 different tests specified by the I field. It does an 8-way branch on I to the particular test instruction.

The Instruction routine then executes a particular CDC 6400 instruction and returns to the the Instruction decode routine.

CDC 6400 EMULATOR

CONTROL STORE MEMORY MAP



4. STATUS OF THE CDC 6400 EMULATOR

All of the instructions have been written and assembled. All the instructions, except the floating point multiply, divide, and addition instructions, have been debugged. (The Datapoint crashed before the last changes could be implemented; but they worked when stepped through by hand.)

The instructions have been timed by hand. When various possible branches occurred, each one was given an equal chance and were averaged out. For instructions where the amount of work depended on the input such as the shift instruction, the value in the middle of the range possible was chosen. When all the instruction times were summed up, it averaged around 65.8 Kips with the fastest being 147.6 Kips for the No-Op and the slowest 3.3 Kips for the floating point divide.

INSTRUCTION TIMES 35 nsec cycles

Opcode	Mnemonic	Instruction	Cycles	Time in μsec	Thousand instructions per second
00	PS	Program Stop	207	7.1	141.1
01	RJ K	Return Jump	335.5	11.7	85.1
02	JP BI,K	Jump	299	10.4	95.5
03		X Register Jump			
030	ZR XJ,K	Jump If Zero	395	13.8	72.3
031	NZ XJ,K	Jump If Not Zero	395	13.8	72.3
032	PL XJ,K	Jump If Positive	395	13.8	72.3
033	NG XJ,K	Jump If Negative	395	13.8	72.3
034	IR XJ,K	Jump If In Range	395	13.8	72.3
035	OR XJ,K	Jump If Out Of Range	395	13.8	72.3
036	DF XJ,K	Jump If Definite	395	13.8	72.3
037	ID XJ,K	Jump If Indifinite	395	13.8	72.3
		CONDITIONAL JUMP ON B REG			
04	EQ BI,BJ,K	Jump BI=BJ	356	12.4	80.2
05	NE BI,BJ,K	Jump BI≠BJ	356	12.4	80.2
06	BE BI,BJ,K	Jump BI≥BJ	356	12.4	80.2
07	LT BI,BJ,K	Jump BI<BJ	356	12.4	80.2
10	BXI XJ	Move IJ to XI	290.5	10.1	98.3
11	BXI XJ*XK	AND	336.5	11.7	84.9
12	BXI XJ+XK	OR	336.5	11.7	84.9
13	BXI XJ-XK	Exclusive OR	336.5	11.7	84.9
14	BXI -XK	NOT			
15	BXI -XK*XJ		353.5	12.3	80.8
16	BXI -XK+XJ		353.5	12.3	80.8
17	BXI -XK-XJ		353.5	12.3	80.8
20	LXI JK	Left Circular Shift	579.5	20.3	49.3
21	AXI JK	Right Arithmetic Shift	406	14.2	70.4
22	LXI BJ,XK	Left Circular Shift varies by $0 \leq X \leq 30$ $X_{av}=15$	579.5	20.3	49.3
23	AXI BJ,XK	Right Circular Shift varies by $0 \leq X \leq 64$ $X_{av}=32$	406	14.2	70.4

Opcode	Menmonic	Instruction	Cycles	Time in μsec	Thousand instructions per second
24	NXI BJ,XK	Normalize varies by $0 \leq X \leq 48$ Xav=24 X.59	2,069.5	72.4	13.8
25	ZXI BJ,XK	Rounded Normalize	2,069.5	72.4	13.8
26	UXI BJ,XK	Real → Integer	490.5	17.1	58.2
27	PXI BJ,XK	Integer → Real	385.5	13.5	74.1
30	FXI XJ+XK	Real Add	1,261	44.1	22
31	FXI XJ-XK	Real subtract	1,261	44.1	22
32	DXI XJ+XK	Read Add low 48 bits	1,261	44.1	22
33	DXI XJ-XK	Real Subtract	1,261	44.1	22
34	RXI XJ-XK	Real Add Rounded	1,261	44.1	22
35	RXI XJ-XK	Real Subtract	1,261	44.1	22
36	IXI XJ-XK	Integer Add	423	14.8	67.5
37	IXI XJ-XK	Integer Subtract	423	14.8	67.5
40	VXI XJ*XK	Real Multiply	989	34.6	28.8
41	RXI XJ*XK	Rounded Real Multiply	989	34.6	28.8
42	DXI XJ*XK	Real Multiply Low 48 bits	989	34.6	28.8
43	MXI JK	Mask varies by $0 \leq X \leq 64$ Xav=32	325	12.1	82.8
44	FXI XJ/XK	Real Divide	8,611	301.3	3.3
45	FXI XJ/XK	Real Divide Rounded	8,611	301.3	3.3
46	No	No Op	193.5	6	147.6
47	CXI XK	Count One's in XK	2,597	90.9	10.9
50	SAI AJ+K	Set A Register			
	A0	Scratch	386	13.5	74
	A1-A5	Fetch	521	18.2	54.8
	A6-A7	Store	535	18.7	53.4
51	SAI BJ+K			Similar to the above	
52	SAI XJ+K			" " " "	
53	SAI XJ+BJ			Similar to below	
54	SAI AJ+BJ			" " "	
55	SAI AJ-BJ			" " "	
56	SAI BJ+BJ				
	A0		320.5	11.2	89.1
	A1-A5		455.5	15.9	62.7
	A6-A7		469.5	16.4	60.8
57	SAI BJ-BJ			Similar to above	

Opcode	Menmonic	Instruction	Cycles	Time in μsec	Thousand instructions per second
60	SBI AJ+K	Set B Register	394	13.8	72.5
61	SBI BJ+K		394	13.8	72.5
62	SBI XJ+K		394	13.8	72.5
63	SBI XJ+BJ		310	10.8	92.2
64	SBI AJ+BJ		310	10.8	92.2
65	SBI AJ-BJ		310	10.8	92.2
66	SBI BJ+BJ		310	10.8	92.2
67	SBI BJ-BJ		310	10.8	92.2
70	SXI AJ_K	Set X Register	445.5	15.6	64.1
71	SXI BJ+K		445.5	15.6	64.1
72	SXI XJ+K		445.5	15.6	64.1
73	SXI XJ+BK		370	12.9	77.2
74	SXI AJ+BK		370	12.9	77.2
75	SXI AJ-BK		370	12.9	77.2
76	SXI BJ+BK		370	12.9	77.2
77	SXI BJ-BK		370	12.9	77.2

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: DSL Tech. Note # 120	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): AN EMMY BASED EMULATION OF THE CDC 6000 SERIES CPU	5. TYPE OF REPORT & PERIOD COVERED: Technical Note
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Ellard T. Roush	8. CONTRACT OR GRANT NUMBER(s): DASG60-77-C-0073
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Stanford Electronics Laboratories, Stanford University, Stanford, CA 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 6.33.04.A
11. CONTROLLING OFFICE NAME AND ADDRESS: Ballistic Missile Defense Advanced Technology Ctr., ATC-P, P.O. Box 1500, Huntsville, AL 35807	12. REPORT DATE: July 1977
	13. NUMBER OF PAGES: 9
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Reproduction in whole or in part is permitted for any purpose of the United States Government.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An emulator for a CDC 6400 computer has been written for the EMMY computer. The full standard instruction set of the CDC 6400 is supported. The optional compare and move unit is not supported. The emulator duplicates precisely those actions that are visible to the user. Certain things that are not visible or accessible, such as the field length register, are not emulated. The emulator causes error halts for the same reasons as the true machine, Infinite and Indefinite operands, address out of bounds, and attempt to execute Location 0.