Aod8 Processor Architecture

Aod8 is an 8-bit Abstract processor. Although the size of the registers can vary, the opcodes & the data (X) are all 1 byte in size thus making it fundamentally a pseudo 8-bit processor. The boot-rom can only contain opcodes & wherever applicable, opcode followed by 1-byte data. Feel free to copy / modify / redistribute this project & all the associated code / programs / tools. This project was created as a part of **MalCon-2011 : Capture the Mal Challenge.**

Registers

IP	= Instruction Pointer.	(Size can be infinite.)
SP	= Stack Pointer.	(Size can be infinite.)
Stack Size	= 65535 bytes or more.	(Can be infinite.)
Flag	= Flag register.	(Atleast 1 byte.)
A	= General Register.	(Atleast 1 byte.)
B	= General Register.	(Atleast 1 byte.)

Note :

(All opcodes are in Hexadecimal.)

[SP] = The value at the memory location where SP (Stack Pointer) is pointing in the stack. (Stack[SP])

"**X**" = The next byte in the bootrom after the specified OpCode.

Instruction OpCodes

Opcode	Instruction	Size in the Rom	Description
11	mov A,[SP]	1 byte	A= value at [SP]
12	mov B,[SP]	1 byte	B = value at [SP]
13	mov A,B	1 byte	A = B
14	mov B,A	1 byte	B = A
15	mov [SP],A	1 byte	value at [SP] = A
16	mov [SP],B	1 byte	value at [SP] = B
17	mov SP,A	1 byte	SP = A
18	mov SP,B	1 byte	SP = B
19	mov A,SP	1 byte	A = SP
1A	mov A,X	2 bytes (opcode+X)	A = X
1B	mov B,X	2 bytes (opcode+X)	B = X
1C	add A,X	2 bytes (opcode+X)	A = A+X
1D	add A,B	1 byte	A = A+B

1E	add B,X	2 bytes (opcode+X)	B = B+X
1F	add B,A	1 byte	B = B+A
20	sub A,X	2 bytes (opcode+X)	A = A-X
21	sub A,B	1 byte	A = A-B
22	sub B,X	2 bytes (opcode+X)	B = B-X
23	sub B,A	1 byte	B = B-A
24	cmp A,B	1 byte	Flag = A-B
25	cmp A,X	2 bytes (opcode+X)	Flag = A-X
26	cmp B,X	2 bytes (opcode+X)	Flag = B-X
27	jmp A	1 byte	IP = A
28	jmp B	1 byte	IP = B
29	jmp X	2 bytes (opcode+X)	IP = IP+X
2A	jne A	1 byte	If (Flag != 0) IP = A
2B	jne B	1 byte	If (Flag != 0) IP = B
2C	jne X	2 bytes (opcode+X)	If (Flag != 0) IP = IP+X
2D	je A	1 byte	If (Flag == 0) IP = A
2E	je B	1 byte	If (Flag == 0) IP = B
2F	je X	2 bytes (opcode+X)	If (Flag == 0) IP = IP + X
30	jge A	1 byte	If (Flag >= 0) IP = A
31	jge B	1 byte	If (Flag >= 0) IP = B
32	jge X	2 bytes (opcode+X)	If (Flag >= 0) IP = IP+X
33	jle A	1 byte	If (Flag <= 0) IP = A
34	jle B	1 byte	If (Flag <= 0) IP = B
35	jle X	2 bytes (opcode+X)	If (Flag <= 0) IP = IP+X
36	inc A	1 byte	A = A+1
37	inc B	1 byte	B = B+1
38	inc SP	1 byte	SP = SP+1
39	input	1 byte	Get 1 byte input & store it at Stack[SP]
3A	output	1 byte	Print 1 byte from Stack[SP] or [SP]
3B	nop	1 byte	Increment IP
3C	halt	1 byte	End execution
3D	loop X	2 bytes (opcode+X)	IP = IP-X
3E	sleep X	2 bytes (opcode+X)	Sleep X seconds

Bootrom Structure

Bootrom is a continuous stream of OpCodes & data. Based on the instruction & the size of the instruction's OpCode , the Instruction-Pointer (IP) is modified.

Sample bootrom-stream :

Offset	0	1	2	3
Opcode	0x1A	0x01	0x14	0x3C

Disassembly :

```
 [ IP = 0,  A= 0, B = 0, SP = 0, [SP] = 0, Flag = 0 ]
mov A,0x01
 [ IP = 2,  A= 0x1, B = 0, SP = 0, [SP] = 0, Flag = 0 ]
mov B,A
 [ IP = 3,  A= 0x1, B = 0x1, SP = 0, [SP] = 0, Flag = 0 ]
halt
```

About

- Author : Aodrulez. (Atul Alex Cherian.)
- Blog : http://Aodrulez.blogspot.com
- Email : f3arm3d3ar@gmail.com
- Project : https://github.com/Aodrulez/Aod8
- Company : http://www.orchidseven.com

I am an “Information Security Researcher/Analyst” currently working with Orchidseven. My interests include programming, reverse-engineering, shellcode development, exploit writing, hardware hacking, web-application testing & so on. This is one of my personal projects designed specifically for MalCon-2011. I hope that someone learns something new out of this & that this inspires people to think different.

Have a wonderful day ahead!

Notes :

- Its recommended to turn off buffering before output while programming the Processor. The above technique can be implemented in c/c++ by adding "**setbuf(stdout, NULL);"** to the code & by adding "**$| = 1;"** in case of PERL.
- Special thanks to Esoteric Languages like **Brainfuck** & the Reverse-Engineering community for inspiring me.
- I'd like to mention that I owe a lot of my RE skills to the legendary +ORC tutorials that got me fascinated & started as well as to the amazing video tutorials by Lena.
- Last but not least, this would have been impossible without the freedom at work & support of my boss, Mr. Rajshekhar Murthy & my company **Orchidseven.com**. :)