

Implementation

This section is a comprehensive guide to all the programs that we coded on the processor and can of great help for anyone who wishes to begin writing a complex routine on the processor.

#Program 1 - Adding two Numbers

ASCII : ?@?@<

HEX :

	0	1	2	3	4	5	6	7	8
00000000	3F	40	11	3F	12	1D	15	40	3C

Description :

This is a simple addition program, the assembly equivalent is as follows.

```
SCANF("%d", &STACK[SP])
PRINTF("%d", STACK[SP])
MOV A,STACK[SP]
SCANF("%d", &STACK[SP])
MOV B,STACK[SP]
ADD A, B
MOV STACK[SP],A
PRINTF("%d",STACK[SP])
HALT
```

Output :

```
+-----+
|  OIUM Virtual Processor  |
+-----+

[+] Reading the Boot-Rom.
[+] Read successfully.
[+] Total Size : 9.
[!] Registers initialised to 0.
[!] Booting using the Boot-Rom.
-----
122
122 567
689
Halt.
Process returned 0 (0x0)   execution time : 19.679 s
Press any key to continue.
```

#PROGRAM 2 : While Loop

ASCII : ㄟ%-昞澁穢

HEX :

	0	1	2	3	4	5	6	7	8	9	A	B
00000000	3F	11	25	00	2F	06	15	40	41	3D	07	3C

Description :

This is basically a while loop where is we ask the person to enter a number and decrement the number at each instruction till it becomes zero.

Following is the assembly implementation.

```
scanf("%d",&STACK[SP])
MOV A,STACK[SP]
FLAG=A-ROM[++IP]
JE 6 BYTES
MOV STACK[SP],A
PRINTF("%d",STACK[SP])
-- A
JMP -7 BYTES
HALT
```

OUTPUT :

```
+-----+
|  OIUM Virtual Processor  |
+-----+

[+] Reading the Boot-Rom.
[+] Read successfully.
[+] Total Size : 12.
[!] Registers initialised to 0.
[!] Booting using the Boot-Rom.
-----

20
2019181716151413121110987654321
Halt.
Process returned 0 (0x0)   execution time : 2.864 s
Press any key to continue.
```

#PROGRAM 3 : ARRAY CREATION

ASCII : ㄟ L 一 嶧 稜 〇 & ㄣ 縊 漁 穢

HEX :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
00000000	B F	1 1	1 4	2 5	0 0	2 F	0 6	3 8	3 F	4 1	3 D	0 7	2 6	0 0	2 F	0 6	4 0	4 3	4 2
00000013	3 D	0 7	3 C																

Description :

In this program the person is asked to enter the size of the array, followed by entering the elements in the array, then those elements are displayed last first.

```
scanf("%d",&STACK[SP])
MOV A,STACK[SP]
MOV B,A
FLAG=A-ROM[++IP]
JE 6 BYTES
++SP
scanf("%d",&STACK[SP])
--A
JUMP -7 BYTES
FLAG=B-ROM[++IP]
JE 6 BYTES
PRINTF("%d",STACK[SP])
--SP
--B
JMP -7BYTES
HALT
```

OUTPUT

```
+-----+
| OIUM Virtual Processor |
+-----+

[+] Reading the Boot-Rom.
[+] Read successfully.
[+] Total Size : 22.
[!] Registers initialised to 0.
[!] Booting using the Boot-Rom.
-----

5
10 20 40 30 94
9430402010
Halt.
Process returned 0 (0x0)   execution time : 11.999 s
Press any key to continue.
```

#PROGRAM 4 : CIPHER ENCRYPTION, REPLACEMENT.

ASCII : ㄟL一嶧稜oX一ㄥW篡滃ㄥ%-搥滄穢

HEX :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
00000000	3F	11	14	25	00	2F	06	38	3F	41	3D	07	13	26	00	2F	09	11	1C
00000013	05	15	42	43	3D	0A	11	25	00	2F	06	38	3A	41	3D	07	3C		

DESCRIPTION :

In this program the user enters the number of elements to be encrypted. Then those many elements are to be entered, then the string is displayed after adding 5 to the values entered.

The logic of importance here is that after the stack pointer is decremented the number of times equal to the number of values, the value just as one lower position is equal to the number of values.

OUTPUT :

```
+-----+
[+] Reading the Boot-Rom.
[+] Read successfully.
[+] Total Size : 36.
[!] Registers initialised to 0.
[!] Booting using the Boot-Rom.
-----
10
23
44
91
7
80
62
39
77
52
37
L1`?UC,R9*
Halt.
Process returned 0 (0x0)   execution time : 52.741 s
Press any key to continue.
```

#PROGRAM 5 : LINEAR SEARCH

ASCII : ?% /8?A=8?& / CH

/

B= @<

HEX :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
00000000	3F	11	14	25	00	2F	06	38	3F	41	3D	07	38	3F	11	26	00
00000011	2F	0B	43	48	00	2F	04	42	3D	0A	19	15	40	3C			

Description : In this program we simply as the use to enter a variable number of values and then enter the value to be searched. The index of the value being searched is returned.

Output :

```
[+] Reading the Boot-Rom.
[+] Read successfully.
[+] Total Size : 31.
[!] Registers initialised to 0.
[!] Booting using the Boot-Rom.
-----
10
0
1
2
3
4
5
6
7
8
9
3
4
Halt.
Process returned 0 (0x0)   execution time : 18.356 s
Press any key to continue.
```

#PROGRAM 6 : On the fly decryption of code

ASCII : LB& MJ8B=L +C% / "AC= 8LA% MT8A=D*4 =DFB +4 ! GHb*4 =?FB A

HEX :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4C	14	42	26	00	4D	07	4A	15	38	42	3D	08	4C	20	2B
00000010	17	43	25	00	2F	09	12	22	05	16	41	43	3D	0A	38	4C
00000020	41	25	00	4D	07	12	54	38	41	3D	08	44	16	19	2A	05
00000030	34	0B	3D	44	46	42	0C	18	2B	05	34	0E	16	21	0A	1A
00000040	47	48	42	0F	16	2A	05	34	0B	3D	3F	46	42	0C	41	

DESCRIPTION :

This is one of the most important programs, in this program first the entire ROM gets decrypted (which is originally encrypted using a plus 5 substitution cipher) and is then executed on the fly.

Following is the assembly level description.

[DECODING STUB]

MOV A, fileLen

MOV B,A

--B

FLAG=B-ROM[++IP]

JL 7 BYTES

MOV A,ROM[B]

MOV STACK[SP],A

++SP

--B

JMP -8 BYTES

MOV A, fileLen

A=A-ROM[++IP]

JNE 23 BYTES

--SP

FLAG:A-[ROM++IP]

JE 9 BYTES

MOV B,STACK[SP]

B=B-ROM[++IP]

MOV STACK[SP],B

--A

--SP

JMP -16 BYTES

```

++SP
MOV A ,fileLen
--A
FLAG=A-[ROM++IP]
JL 7 BYTES
MOV B,STACK[SP]
MOV ROM[A],B
++SP
[ ACTUAL ENCRYPTED CODE]

```

OUTPUT : The OUTPUT is exactly the same as the encryption program we used earlier.

```

+-----+
[+] Reading the Boot-Rom.
[+] Read successfully.
[+] Total Size : 36.
[!] Registers initialised to 0.
[!] Booting using the Boot-Rom.
-----
10
23
44
91
7
80
62
39
77
52
37
-1'♀UC,R9*
Halt.
Process returned 0 (0x0)   execution time : 52.741 s
Press any key to continue.

```

#PROGRAM 7 : On the fly code expansion

ASCII : ?%/?8?8A= <L!%/|gcUda8g|c8Uda87A=ÉE??h<%/†Å8A=

HEX :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	3F	11	25	00	2F	08	3F	38	3F	38	41	3D	09	1B	3C	4C
00000010	21	25	00	2F	14	7C	67	63	55	64	61	38	67	7C	63	38
00000020	55	64	61	38	37	41	3D	15	C9	45	3F	11	3F	12	18	68
00000030	3C	25	00	2F	08	12	86	C5	38	41	3D	09	00	02	04	06
00000040	08															

DESCRIPTION :

This program is a proof of concept program where in we dynamically enter the hex codes in an index based dictionary which is used later to expand and execute the code. The stub is 60 bytes long. We in this program have expanded the earlier program we wrote for addition of two numbers by 1:2 ratio. Following is the assembly level description :

[EXPANSION STUB]

```
scanf("%d",&STACK[SP])
MOV A,STACK[SP]
FLAG=A-ROM[++IP]
JE 8 BYTES
scanf("%d",&STACK[SP])
++SP
scanf("%d",&STACK[SP])
++SP
--A
JMP -9 BYTES
MOV B,ROM[++IP] //STUB LENGTH
A = fileLen
A=A-B
FLAG=A-ROM[++IP]
JE 14 BYTES
MOV C ,ROM[B]
MOV D,SP
MOV SP,C
MOV C,STACK[SP]
MOV SP,D
MOV STACK[SP],C
```



```

++SP
MOV D,SP
MOV C,ROM[B]
MOV SP,C
++SP
MOV C,STACK[SP]
MOV SP,D
MOV STACK[SP],C
++SP
++B
--A
JMP -21 BYTES
ROM=(char *)realloc(ROM,ROM[++IP]) //FILE EXPANDED
scanf("%d",&STACK[SP])
MOV A,STACK[SP]
scanf("%d",&STACK[SP])
MOV B,STACK[SP]
MOV SP,B
MOV C,ROM[++IP]
FLAG=A-ROM[++IP]
JE 8 BYTES
MOV B,STACK[SP]
MOV ROM[C],B
++C
++SP
--A
JMP -9 BYTES
[COMPRESSED FORM]

```

OUTPUT :

```

+-----+
|  OIUM Virtual Processor  |
+-----+

[+] Reading the Boot-Rom.
[+] Read successfully.
[+] Total Size : 65.
[!] Registers initialised to 0.
[!] Booting using the Boot-Rom.
-----

5
63 64
17 63
18 29
21 64
60 60
10
10
124
124 45
169
Halt.
Process returned 0 (0x0)   execution time : 49.517 s
Press any key to continue.

```