Starter의 기능 설명 및 테스트 계획서

지난 주 AbsoluteLoader를 통해 Starter를 받아와 새로운 Boot파일을 만드는 실습을 하였습니다.

* 아래는 어셈블리 후 생성된 Loader의 리스트입니다.

	cassette/my	Test.lmc		
	_	ABSOLUTE LOADER	СПУБП	0
		KEYBOARD DEC		10
		-	EQU	
			EQU	12
			EQU	SCREEN_DEC+1
000	00:0000 5 6	BOOT_DEV_ADDR	EQU	30
000		PROGNAME	RESBOX	10
		I_VAL	RESDBOX	
		LENGTH	RESDBOX	
		EXEC_START		
000	00:0016 11 12	PTR_START	RESDBOX	1
6700 0022 0000 000	00:0018 13		CALL	MAIN
0700 000	00:0021 14		СОВ	
	15			
	16			
9630 000		MAIN	TIO	BOOT DEV ADDR
	00:0023 18		IN	BOOT_DEV_ADDR
	00:0024 19		SUB	A #72
	00:0024 19		JZ	
	00:0027 20			HEAD_IS_FINE
9710 000	22		INT	10
		//#boain		
		//#begin	. MOV. C	41 0
4225 0010 0000 000		//% HEAD_IS_FINE		
4325 0010 0000 000		HEAD_IS_FINE //#end	LD C #1	10
		//#end		
	24	/ / //1 ! -		
		//#begin		
4045 0000 0000 000		//% MOV B #0	"	
4315 0000 0000 000	00:0034 25		LD B #0)
	0.5	1 1 11 1		
		//#end		
2522	26			
0530 000	26 00:0037 27	LOAD_NAME	IN	BOOT_DEV_ADDR
0530 000	26 00:0037 27 28	LOAD_NAME //#begin		BOOT_DEV_ADDR
	26 00:0037 27 28 28	LOAD_NAME	IE A	
	26 00:0037 27 28 28 00:0038 28	LOAD_NAME //#begin //% MOV %PROGNAM	IE A	BOOT_DEV_ADDR
	26 00:0037 27 28 28 00:0038 28 28	LOAD_NAME //#begin	IE A	
5205 0000 000	26 00:0037 27 28 28 00:0038 28 28 29	LOAD_NAME //#begin //% MOV %PROGNAM	1E A ST A %I	PROGNAME
5205 0000	26 00:0037 27 28 28 00:0038 28 28 29 00:0040 30	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I	PROGNAME B #1
5205 0000	26 00:0037 27 28 28 00:0038 28 28 29 00:0040 30 00:0043 31	LOAD_NAME //#begin //% MOV %PROGNAM	1E A ST A %I	PROGNAME
1315 0001 0000 000 5990 0037 0000 000	26 00:0037 27 28 28 00:0038 28 28 29 00:0040 30 00:0043 31 32	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP	PROGNAME B #1 LOAD_NAME
1315 0001 0000 000 5990 0037 0000 000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A % I ADD JLOOP	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR
5205 0000 0000 1315 0001 0000 0000 5990 0037 0000 0000 0530 0016 0000 0000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32 00:0046 33 00:0047 34	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP IN ST	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR A PTR_START
5205 0000 0000 1315 0001 0000 000 5990 0037 0000 000 0530 0016 0000 000 0530 000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32 00:0046 33 00:0047 34 00:0050 35	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP IN ST IN	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR A PTR_START BOOT_DEV_ADDR
5205 0000 0000 1315 0001 0000 000 5990 0037 0000 000 0530 0016 0000 000 0530 000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32 00:0046 33 00:0047 34 00:0050 35 00:0051 36	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP IN ST	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR A PTR_START
5205 0000 0000 1315 0001 0000 000 5990 0037 0000 000 0530 0016 0000 000 0530 000 5300 0017 0000 000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32 00:0047 34 00:0050 35 00:0051 36 37	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP IN ST IN	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR A PTR_START BOOT_DEV_ADDR A PTR_START+1
5205 0000 0000 1315 0001 0000 0000 5990 0037 0000 0000 0530 0016 0000 0000 0530 0017 0000 0000 0530 0017 0000 0000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32 00:0047 34 00:0050 35 00:0051 36 37 00:0054 38	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP IN ST IN ST	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR A PTR_START BOOT_DEV_ADDR
5205 0000 0000 1315 0001 0000 000 5990 0037 0000 000 0530 0016 0000 000 0530 000 5300 0017 0000 000 0530 000 5300 0017 0000 000 5300 0012 0000 000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32 00:0047 34 00:0050 35 00:0051 36 37 00:0054 38 00:0055 39	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP IN ST IN ST	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR A PTR_START BOOT_DEV_ADDR A PTR_START+1 BOOT_DEV_ADDR A LENGTH
5205 0000 0000 1315 0001 0000 000 5990 0037 0000 000 0530 0016 0000 000 0530 0017 0000 000 0530 0017 0000 000 0530 0012 0000 000 0530 0012 0000 000 0530 0012 0000 000	26 00:0037 27 28 28 00:0038 28 29 00:0040 30 00:0043 31 32 00:0047 34 00:0050 35 00:0051 36 37 00:0054 38	LOAD_NAME //#begin //% MOV %PROGNAM	ME A ST A %I ADD JLOOP IN ST IN ST	PROGNAME B #1 LOAD_NAME BOOT_DEV_ADDR A PTR_START BOOT_DEV_ADDR A PTR_START+1 BOOT_DEV_ADDR

				42			
4320	0012	0000	0000:0062	43		LD	C LENGTH
				44	//#begin		
				44	//% MOV X #0		
4345	0000	0000	0000:0065	44		LD X #	0
				44	//#end		
0530			0000:0068	45	LOAD_TAPE	IN	BOOT_DEV_ADDR
5303	0016	0000	0000:0069	46		ST	A *@PTR_START
				47			
1345	0001	0000	0000:0072	48		ADD	X #1
5990	0068	0000	0000:0075	49		JLOOP	LOAD_TAPE
				50			
0530			0000:0078	51		IN	BOOT_DEV_ADDR
				52	//#begin		
				52	//% MOV B #69		
4315	0069	0000	0000:0079	52		LD B #	69
4315	0069	0000	0000:0079		//#end	LD B#	69
4315 1401	0069	0000			//#end	LD B #	69 A B
1401	0069		0000:0082	52	//#end		
1401			0000:0082	52 53	//#end	SUB	АВ
1401 5910			0000:0082 0000:0083	52 53 54	//#end	SUB JZ	A B END_IS_FINE
1401 5910			0000:0082 0000:0083	52 53 54 55 56	//#end END_IS_FINE	SUB JZ	A B END_IS_FINE
1401 5910 9710		0000	0000:0082 0000:0083 0000:0086	52 53 54 55 56		SUB JZ INT	A B END_IS_FINE 10
1401 5910 9710	0087	0000	0000:0082 0000:0083 0000:0086	52 53 54 55 56 57		SUB JZ INT	A B END_IS_FINE 10 BOOT_DEV_ADDR
1401 5910 9710 0530 5300 0530	0087	0000	0000:0082 0000:0083 0000:0086 0000:0087 0000:0088	52 53 54 55 56 57 58		SUB JZ INT IN	A B END_IS_FINE 10 BOOT_DEV_ADDR A EXEC_START
1401 5910 9710 0530 5300 0530 5300	0087	0000	0000:0082 0000:0083 0000:0086 0000:0087 0000:0088 0000:0091	52 53 54 55 56 57 58 59		SUB JZ INT IN ST IN	A B END_IS_FINE 10 BOOT_DEV_ADDR A EXEC_START BOOT_DEV_ADDR
1401 5910 9710 0530 5300 0530 5300	0087 0014 0015	0000	0000:0082 0000:0083 0000:0086 0000:0087 0000:0088 0000:0091 0000:0092	52 53 54 55 56 57 58 59 60		SUB JZ INT IN ST IN ST	A B END_IS_FINE 10 BOOT_DEV_ADDR A EXEC_START BOOT_DEV_ADDR A EXEC_START+1
1401 5910 9710 0530 5300 0530 5300	0087 0014 0015	0000	0000:0082 0000:0083 0000:0086 0000:0087 0000:0088 0000:0091 0000:0092	52 53 54 55 56 57 58 59 60 61 62		SUB JZ INT IN ST IN ST	A B END_IS_FINE 10 BOOT_DEV_ADDR A EXEC_START BOOT_DEV_ADDR A EXEC_START+1
1401 5910 9710 0530 5300 0530 5300	0087 0014 0015	0000	0000:0082 0000:0083 0000:0086 0000:0087 0000:0088 0000:0091 0000:0092 0000:0095	52 53 54 55 56 57 58 59 60 61 62	END_IS_FINE	SUB JZ INT IN ST IN ST JUMP	A B END_IS_FINE 10 BOOT_DEV_ADDR A EXEC_START BOOT_DEV_ADDR A EXEC_START+1 *EXEC_START

.....

* 아래는 어셈블리 후 생성된 Starter의 리스트입니다.

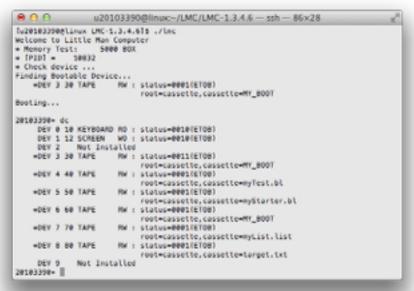
Execute% Assemble 3:cassette/myStarter.lmc

```
1 // System Monitor Program Entry
                0000:2000
                           2 ROM BEGIN
                                             EQU 91000000U
                0000:2000
                           3 BIOS
                                              EQU 90011000U
                0000:2000 4 M PROMPT
                                              EQU 30
                0000:2000 5 M_QUIT
                                              EQU 31
                0000:2000
                          6 M_LOADDEV
                                              EQU 32
                0000:2000
                          7 M_LOADBL
                                              EQU 33
                0000:2000
                          8 M_LOADRL
                                              EQU 34
                0000:2000
                          9 M_LOADLL
                                              EQU 35
                0000:2000 10 M_LOAD
                                              EQU 36
                0000:2000 11 M_MOVE
                                              EQU 37
                0000:2000
                          12 M_RUN
                                              EQU 38
                                                              해당하는 명령어로 이동하기
                0000:2000
                          13 M DUMP CMOS
                                              EQU 39
                                                              위해, 명령어에 해당하는 값
                0000:2000
                          14 M_DUMP
                                              EQU 40
                0000:2000
                          15 M_ASM
                                              EQU 41
                                                              을 지정해 둡니다.
                0000:2000
                          16 M DASM
                                              EQU 42
                          17 M_INTERPRET
                0000:2000
                                              EQU 43
                                                              프롬프트에서 입력을 받으면
                0000:2000
                          18 M_HELP
                                              EQU 44
                          19 M CC
                0000:2000
                                              EQU 45
                                                              이 값을 참고하여 해당 명령
                0000:2000
                          20 M_CAT
                                              EQU 46
                                                              어로 이동 후 실행합니다.
                0000:2000
                          21 M_MKBOOT
                                              EQU 47
                          22 M VIEWBL
                                              EQU 48
                0000:2000
                          23 M_VIEWRL
                0000:2000
                                              EQU 48
                0000:2000
                          24 M_VIEWLL
                                              EQU 48
                          25
                                              // Monitor
                0000:2000
                          26 ROM3
                                              START
                                                    2000 // ROM3
9993
                0000:2000
                          27 MONITOR_ENTRY
                                              INTON
                           28
                           29
```

100 100		2.0	
1000 1000	1250 0022	30	ID CD NEAD -4000H
1945 1947 1948			
9730			
1040 00000 2000			
1640 2036 0000 000012011 36 38 87 31 // PROMPTE INT30을 반변 131	9730		
1144 000012013 39 ADD X X X / 2X, DBOX Index:ing 4200 0011 000012016 41 PUSH A 6802 0023 000012017 42 CALL NEAR #NONITOR_TABLE 4500 0000 000012017 43 ADD SP NEAR = 2 5800 9982 000012011 44 JUMP NEAR FROMFT 46 LFORK 4900 0000 000012023 46 S = 4700	4140	0000:2009 35	
144	1640 2036 0000	0000:2010 36	SUB X =31 // PROMPT는 INT값을 반환
1144 000012013 39		37	// 31부터시작함으로 31을 뺀다.
1144 000012013 39		38	// M PROMPT가 30임으로 31부터
980 0001 00012014 40	1144	0000:2013 39	
988			
1			
1250 0019 00012011 44 14			
5800 9982 0000:2012 44 54 54 54 54 54 54 5			-
변경			
LTORG	3800 9982		JUMP NEAR PROMPT
990 0000 0000 2003 46 \$ = 4900			LHODG
4700 0000 000012027 46 \$ =4700 의 주소값을 입력받게 됩니다. 00048	4000 0000		
0000			
0050 0001 2002 46 '0' '0' 프롬포트 실행 후, 파라미터 20051 0001 20029 46 '1' 프롬포트 실행 후, 파라미터 20051 0001 20031 46 '3' 로넘어온 값을 통해 해당하는 명령모으로 넘어갑니다. 0051 00001 20032 46 '3' '0' 로넘어온 값을 통해 해당하는 명령모으로 넘어갑니다. 0051 00001 20033 46 '9' '0' 0000 00001 20035 46 '0' '0' 0000 00001 20035 46 '10' 0001 0000 00001 2035 46 '10' 0000 0000 00001 2036 46 \$ =31 0002 0000 00001 49 '8 MONITOR_TABLE INT M_QUIT AFFINAL STANDARD		·	· · · · · · · · · · · · · · · · · · ·
0048 00012029 46 10' 0' 고롱포트 실행 후, 파라미터 2 넘어온 값을 통해 해당하 0051 00012031 46 33' 는 명령문으로 넘어갑니다. 0051 00012032 46 '3' 는 명령문으로 넘어갑니다. 0057 00012033 46 '9' ' 0048 00012034 46 '9' ' 0040 0000 00012035 46 '0' '0' ' 0001 0000 00012036 46 \$ 31		·	ş ='2'
0018 00012030 46 '0' '0' 로 넘어온 값을 통해 해당하 10051 00012031 46 '3' 는 명령문으로 넘어갑니다. 10057 000012033 46 '9' '0' '0' '0' '0' '0' '0' '0' '0' '0'			
0051 0000:2031 46 '3' 도담이온 쇼를 등에 에당이 10057 00001 2033 46 '9' 19' 100000 2033 46 '9' 10' 10' 10' 10' 10' 10' 10' 10' 10' 10			
0051 00002031 46 33 는 명령문으로 넘어갑니다. 0057 00002033 46 99 0048 000012034 46 0000000 000012035 46 100 0031 0000 000012035 46 100 0001 000012035 46 100 0001 000012035 46 100 0002 0000 000012036 46 \$ =31 77 9731 000012040 48 MONITOR_TABLE INT M_OUIT RET 9732 000012041 49 RET 9732 000012042 50 INT M_LOADDEV 9999 000012043 51 RET 9733 000012044 52 INT M_LOADBL 9999 000012045 53 RET 9734 000012045 53 RET 9735 000012045 56 INT M_LOADLL 9999 000012047 55 RET 9736 000012048 56 INT M_LOADLL 9999 000012049 57 RET 9737 000012049 57 RET 9738 000012050 58 INT M_LOADL 9999 000012051 59 RET 9737 000012052 60 INT M_RUN 9999 000012053 61 RET 9738 000012054 62 INT M_RUN 9999 000012055 63 RET 9739 000012056 64 INT M_RUN 9999 000012057 65 RET 9740 000012057 65 RET 9741 000012050 68 INT M_DUMP_CMOS 9999 000012057 65 RET 9741 000012050 71 RET 9742 000012050 71 RET 9744 000012050 73 RET 9745 000012067 75 RET 9746 000012068 77 RET 9746 000012070 78 INT M_CCT INT M_DC 9999 000012070 78 INT M_CAT ONLY M_BIDD 9747 000012070 78 INT M_MCAT ONLY M_BIDD 9747 000012070 78 INT M_MCAT ONLY M_BIDD 9747 000012070 78 INT M_MCAT ONLY M_BIDD 9747 000012070 78 INT M_MEDOT TABLE 9746 000012070 78 INT M_MKBOOT TABLE 9999 000012071 79 RET 9747 000012070 78 INT M_MKBOOT TABLE 9999 000012071 79 RET	0048	0000:2030 46	'0' 로 넌어오 갔을 통해 해당하
0057	0051	0000:2031 46	
001	0051	0000:2032 46	'3' 는 멍텅문으로 딤어갑니다.
0000	0057	0000:2033 46	'9'
0031 0000 0000:2036 46 \$ =31	0048	0000:2034 46	101
0000 0000 0000:2038 46 \$ =2 개인화한 프롬포트 출력문이 47 가장되어 있습니다. 9731 0000:2041 49 RET 9732 0000:2043 51 RET 9733 0000:2044 52 INT M_LOADDEV 9999 0000:2045 53 RET 9734 0000:2046 54 INT M_LOADDLL 9999 0000:2047 55 RET 9735 0000:2048 56 INT M_LOADLL 9999 0000:2047 55 RET 9736 0000:2048 56 INT M_LOADLL 9999 0000:2049 57 RET 9737 0000:2050 58 INT M_LOADL 9999 0000:2051 59 RET 9738 0000:2051 59 RET 9738 0000:2055 61 RET 9738 0000:2055 62 INT M_ROVE 9999 0000:2057 65 RET 9739 0000:2058 61 RET 9739 0000:2059 62 INT M_ROVE 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP_CMOS 9999 0000:2059 67 RET 9741 0000:2050 68 INT M_DUMP_CMOS 9999 0000:2055 67 RET 9741 0000:2056 64 INT M_DUMP 9999 0000:2057 65 RET 9741 0000:2050 70 RET 9742 0000:2051 71 RET 9743 0000:2065 73 RET 9744 0000:2066 74 RET 9745 0000:2067 75 RET 9746 0000:2067 75 RET 9747 0000:2067 75 RET 9748 0000:2067 75 RET 9749 0000:2067 75 RET 9740 0000:2067 75 RET 9741 0000:2067 75 RET 9744 0000:2067 75 RET 9745 0000:2067 75 RET 9746 0000:2067 76 RET 9747 0000:2070 78 RET INT M_CC 1NT M_CC 1NT M_CC 1NT M_CAT ON CL GODG GODG GODG GODG GODG GODG GODG GOD	0000	0000:2035 46	1/01
9731 0000:2040 48 MONITOR_TABLE INT M_OUIT 9799 0000:2041 49 9732 0000:2042 50 INT M_LOADDEV 9799 0000:2043 51 RET 9733 0000:2045 53 RET 9734 0000:2046 54 INT M_LOADDEL 97999 0000:2046 54 INT M_LOADDEL 97999 0000:2046 54 INT M_LOADLE 97999 0000:2046 56 INT M_LOADLE 97999 0000:2048 56 INT M_LOADLE 97999 0000:2049 57 RET 9735 0000:2048 56 INT M_LOADLE 97999 0000:2051 59 RET 9737 0000:2051 59 RET 9738 0000:2051 62 INT M_RUN 9999 0000:2053 61 RET 9738 0000:2056 64 INT M_RUN 9999 0000:2059 67 RET 9738 0000:2059 67 RET 9740 0000:2059 66 INT M_DUMP_CMOS 97999 0000:2059 67 RET 9740 0000:2059 67 RET 9741 0000:2050 68 INT M_DUMP 97999 0000:2051 69 RET 9741 0000:2050 68 INT M_DUMP 97999 0000:2051 69 RET 9741 0000:2050 68 INT M_DUMP 97999 0000:2051 69 RET 9741 0000:2050 77 RET 9743 0000:2051 75 RET 9744 0000:2066 74 INT M_DASM 9999 0000:2067 75 RET 9744 0000:2068 76 INT M_DASM 9999 0000:2067 75 RET 9745 0000:2067 75 RET 9746 0000:2067 75 RET 9746 0000:2077 75 RET 9746 0000:2077 78 RET 9746 0000:2077 78 RET 9746 0000:2077 78 RET 9747 0000:2077 78 RET 9746 0000:2077 78 RET 9747 0000:2077 78 RET 9747 0000:2077 78 RET 9748 0000:2077 78 RET 9749 0000:2077 78 RET 9740 0000:2077 78 RET 9746 0000:2077 78 RET 9747 0000:2077 78 RET 9747 0000:2077 78 RET 9748 0000:2077 78 RET 9749 0000:2077 78 RET 9740 0000:2077 78 RET 9747 0000:2077 78 RET 9748 0000:2077 78 RET 9749 0000:2077 78 RET 9740 0000:2077 78 RET 9747 0000:2077 78 RET 9748 0000:2077 78 RET 9749 0000:2077 78 RET	0031 0000	0000:2036 46 \$	
9731 0000:2040 48 MONITOR_TABLE RET	0002 0000	0000:2038 46 \$	\$ =2 개인화한 프롬포트 출력문이
9731 0000:2041 49 RET 9732 0000:2042 50 INT M_COADDEV 9999 0000:2043 51 RET 9733 0000:2044 52 INT M_LOADDEL 9999 0000:2045 53 RET 9734 0000:2046 54 INT M_LOADDEL 9999 0000:2047 55 RET 9735 0000:2048 56 INT M_LOADDEL 9999 0000:2047 55 RET 9736 0000:2049 57 RET 9736 0000:2050 58 INT M_LOADDE 9999 0000:2051 59 RET 9737 0000:2052 60 INT M_MOVE 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2055 63 RET 9739 0000:2055 64 INT M_DUMP_CMOS 9999 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2059 67 RET 9741 0000:2050 68 INT M_ASM 9999 0000:2050 71 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9744 0000:2064 72 INT M_INTERPRET 7999 0000:2065 73 RET 9744 0000:2065 74 RET 9744 0000:2066 74 INT M_INTERPRET 7999 0000:2067 75 RET 9744 0000:2068 76 INT M_CC INT M_DC 9999 0000:2067 75 RET 9744 0000:2068 76 INT M_CC		47	저자디어 이스니다
9999 0000:2041 49 RET	9731	0000:2040 48 N	MONITOR TABLE INT M QUIT
9999 0000:2043 51 RET 9733 0000:2045 53 RET 9734 0000:2046 54 INT M_LOADBL 9999 0000:2047 55 RET 9735 0000:2048 56 INT M_LOADL 9999 0000:2049 57 RET 9736 0000:2049 57 RET 9737 0000:2050 58 INT M_LOAD RET 9999 0000:2051 59 RET 9737 0000:2052 60 INT M_MOVE 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 82999 0000:2055 63 RET 9738 0000:2055 63 RET 9739 0000:2055 63 RET 9739 0000:2055 63 RET 9739 0000:2055 64 INT M_DUMP_CMOS 8999 0000:2055 65 RET 9740 0000:2057 65 RET 9741 0000:2058 66 INT M_DUMP RET 9741 0000:2060 68 INT M_ASM 8999 0000:2065 70 INT M_DASM 8999 0000:2065 73 RET 9742 0000:2065 73 RET 9744 0000:2064 72 INT M_DASM 8999 0000:2065 73 RET 9744 0000:2066 74 INT M_DASM 8ET 9745 0000:2065 75 RET 9746 0000:2066 76 INT M_CC I	9999	0000:2041 49	RET
9733	9732	0000:2042 50	INT M_LOADDEV
9999 0000:2045 53 RET 9734 0000:2046 54 INT M_LOADRL 9999 0000:2047 55 RET 9735 0000:2048 56 INT M_LOADLL 9999 0000:2049 57 RET 9736 0000:2050 58 INT M_LOAD 9999 0000:2051 59 RET 9737 0000:2052 60 INT M_RUN 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2055 63 RET 9739 0000:2055 64 INT M_DUMP_CMOS 9999 0000:2055 65 RET 9740 0000:2056 64 INT M_DUMP 9999 0000:2057 65 RET 9741 0000:2066 68 INT M_DUMP 9999 0000:2056 67 RET 9741 0000:2060 68 INT M_DASM 9999 0000:2061 69 RET 9742 0000:2061 69 RET 9742 0000:2063 71 RET 9744 0000:2064 72 INT M_DASM 9999 0000:2065 73 RET 9744 0000:2065 73 RET 9744 0000:2067 75 RET 9745 0000:2067 75 RET 9746 0000:2067 75 RET 9746 0000:2067 75 RET 9745 0000:2067 75 RET 9745 0000:2067 75 RET 9746 0000:2067 76 RET INT M_CC INT	9999	0000:2043 51	RET
9734	9733	0000:2044 52	INT M_LOADBL
9999 0000:2047 55 RET 9735 0000:2048 56 INT M_LOADLL 9999 0000:2049 57 RET 9736 0000:2050 58 INT M_LOAD 9999 0000:2051 59 RET 9737 0000:2052 60 INT M_MOVE 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2058 66 INT M_DUMP 9999 0000:2058 66 INT M_ASM 9999 0000:2056 68 INT M_ASM 9999 0000:2060 68 INT M_DASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9743 0000:2065 73 RET 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC INT M_CC 1NT M_CAT ON QLE GGO 3000:2071 79 RET 1NT M_CAT ON QLE GGO 3000:2071 79 RET 1NT M_CAT ON QLE GGO 3000:2071 79 RET ON QLE GGO 3000:2071 70	9999	0000:2045 53	RET
9735 0000:2048 56 INT M_LOADLL 9999 0000:2049 57 RET 9736 0000:2050 58 INT M_LOAD 9999 0000:2051 59 RET 9737 0000:2052 60 INT M_MOVE 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2058 66 INT M_ASM 9999 0000:2058 66 INT M_ASM 9999 0000:2050 69 RET 9741 0000:2060 68 INT M_DASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9744 0000:2065 73 RET 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET INT M_HELP 9999 0000:2068 76 INT M_CC 9999 0000:2068 76 INT M_CC 9999 0000:2070 78 RET INT M_CAT OH QLE BE	9734	0000:2046 54	INT M_LOADRL
9999 0000:2051 58 INT M_LOAD 9999 0000:2051 59 RET 9737 0000:2052 60 INT M_MOVE 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2057 65 RET 9741 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2051 69 RET 9742 0000:2061 69 RET 9743 0000:2063 71 RET 9743 0000:2063 71 RET 9744 0000:2065 73 RET 9799 0000:2066 74 INT M_INTERPRET 9799 0000:2067 75 RET 9744 0000:2068 76 INT M_HELP 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2071 79 RET INT M_CAT CAT CAT CAT CAT CAT CAT CAT CAT CAT	9999	0000:2047 55	RET
9999 0000:2049 57 RET 9736 0000:2050 58 INT M_LOAD 9999 0000:2051 59 RET 9737 0000:2052 60 INT M_MOVE 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2057 65 RET 9741 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2061 69 RET 9742 0000:2063 71 RET 9743 0000:2063 71 RET 9744 0000:2066 74 INT M_INTERPRET 8 OF MACE 9999 0000:2065 73 RET 9744 0000:2066 74 INT M_INTERPRET 9745 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC INT M_CC 9999 0000:2071 79 RET INT M_CAT PRET 9746 0000:2071 79 RET INT M_CAT PRET 9747 0000:2072 80 INT M_MKBOOT 기됩니다.	9735	0000:2048 56	INT M_LOADLL
9999 0000:2051 59 RET	9999	0000:2049 57	
9737 0000:2052 60 INT M_MOVE 9999 0000:2053 61 RET 9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9744 0000:2068 76 INT M_CC 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2070 78 INT M_CAT 어 있는 명령어를 실행하 9746 0000:2071 79 RET ON M_MKBOOT 게 됩니다.	9736	0000:2050 58	INT M_LOAD
9999 0000:2053 61 RET 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9999 0000:2065 73 RET 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC INT M_CC INT M_CC INT M_CC INT M_CC INT M_CC INT M_CAT ON OUT OF OUT ON OUT OF OUT ON OUT	9999	0000:2051 59	RET
9738 0000:2054 62 INT M_RUN 9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9999 0000:2065 73 RET 97944 0000:2066 74 INT M_HELP 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT M_CC 9999 0000:2070 78 INT M_CAT ON CHECK	9737	0000:2052 60	INT M MOVE
9999 0000:2055 63 RET	9999	0000:2053 61	RET
9999 0000:2055 63 RET 9739 0000:2056 64 INT M_DUMP_CMOS 9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9799 0000:2065 73 RET 9799 0000:2066 74 INT M_HELLP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT M_CC 9999 0000:2070 78 INT M_CAT ON CO INT M_CAT ON CO INT M_DASH 9999 0000:2071 79 RET INT M_CAT ON CO INT M_DASH 9999 0000:2071 79 RET INT M_CAT ON CO INT M_MKBOOT 게 됩니다.	9738	0000:2054 62	INT M_RUN
9999 0000:2057 65 RET 9740 0000:2058 66 INT M_DUMP 9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9999 0000:2065 73 RET 해당하는 명령어로 점프 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT M_CC 1NT M_CC 9999 0000:2070 78 INT M_CAT ON ONE 960 960 9747 RET INT M_CAT ON ONE 960 960 9747 NET INT M_CAT ON ONE 960 960 9745 ON ONE 2071 79 RET INT M_KRBOOT 게 됩니다.	9999	0000:2055 63	
9740 0000:2058 66 INT M_DUMP 9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 하당하는 명령어로 점프 9744 0000:2065 73 RET 하당하는 명령어로 점프 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET INT M_CC 9999 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT M_CAT ON CAT	9739	0000:2056 64	INT M_DUMP_CMOS
9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9999 0000:2065 73 RET 해당하는 명령어로 점프 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET INT M_CC 9999 0000:2069 77 RET INT M_CC 9999 0000:2070 78 INT M_CAT OUT OF OUT	9999	0000:2057 65	RET
9999 0000:2059 67 RET 9741 0000:2060 68 INT M_ASM 9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9999 0000:2065 73 RET 해당하는 명령어로 점프 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET INT M_CC 9999 0000:2069 77 RET INT M_CC 9999 0000:2070 78 INT M_CAT OUT OUT OUT OUT OUT OUT OUT OUT OUT OU	9740	0000:2058 66	INT M_DUMP
9999 0000:2061 69 RET 9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 해당하는 명령어로 점프 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT M_CAT ON CONTROL OF CONTROL	9999	0000:2059 67	RET
9742 0000:2062 70 INT M_DASM 9999 0000:2063 71 RET 9743 0000:2064 72 INT M_INTERPRET 9999 0000:2065 73 RET 해당하는 명령어로 점프 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT M_CAT 9746 0000:2070 78 INT M_CAT 9999 0000:2071 79 RET 9747 0000:2072 80 INT M_MKBOOT 게됩니다.	9741	0000:2060 68	INT M_ASM
9999 0000:2063 71 RET	9999	0000:2061 69	RET
9743 0000:2064 72 INT M_INTERPRET	9742	0000:2062 70	INT M_DASM
9999 0000:2065 73 RET 해당하는 명령어로 점프 9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT 를통해 이미 정의되 9746 0000:2070 78 INT M_CAT 9999 0000:2071 79 RET 9747 0000:2072 80 INT M_MKBOOT 게됩니다.	9999	0000:2063 71	RET
9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 하여 실행하게 됩니다 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT 를통해 이미 정의되 9746 0000:2070 78 INT M_CAT 어있는 명령어를 실행하 9999 0000:2071 79 RET 기됩니다.	9743	0000:2064 72	
9744 0000:2066 74 INT M_HELP 하여 실행하게 됩니다 9999 0000:2067 75 RET 하여 실행하게 됩니다 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT 를통해 이미 정의되 9746 0000:2070 78 INT M_CAT 어있는 명령어를 실행하 9999 0000:2071 79 RET 기됩니다.	9999	0000:2065 73	RET 🥻 해당하는 명령어로 점프 🛚
9999 0000:2067 75 RET 9745 0000:2068 76 INT M_CC 9999 0000:2069 77 RET INT 를통해 이미 정의되 9746 0000:2070 78 INT M_CAT RET 9747 0000:2072 80 INT M_MKBOOT 게 됩니다.	9744	0000:2066 74	
9999 0000:2069 77 RET INT 를 통해 이미 정의되 9746 0000:2070 78 INT M_CAT 어 있는 명령어를 실행하 9999 0000:2071 79 RET 게 됩니다.	9999	0000:2067 75	
9999 0000:2069 77 RET INT 를 통해 이미 정의되 9746 0000:2070 78 INT M_CAT 어 있는 명령어를 실행하 9999 0000:2071 79 RET 게 됩니다.	9745	0000:2068 76	INT M_CC
9746 0000:2070 78 INT M_CAT 이 있는 명령어를 실행하 9999 0000:2071 79 RET 이 있는 명령어를 실행하 기됩니다.	9999	0000:2069 77	
9999 000012071 79 RET 9747 000012072 80 INT M_MKBOOT 게 됩니다.	9746	0000:2070 78	TNITE M. CAMI
— <i>y</i>	9999	0000:2071 79	RET
	9747	0000:2072 80	INT M_MKBOOT 🍶 게 뇝니다.
	9999	0000:2073 81	RET

9744 되어있음.	0000:2074	82	INT M_HELP	// VIEW BL이어야 하지만 구현이 안
9999	0000:2075	83	RET	
9744	0000:2076	84	INT M HELP	// VIEW RL이어야 하지만 구현이 안
되어있음.			_	
9999	0000:2077	85	RET	
9744	0000:2078	86	INT M_HELP	// VIEW LL이어야 하지만 구현이 안
되어있음.				
9999	0000:2079	87	RET	
		88	END	

* 아래는 mkboot 를 통해 생성된 MY_BOOT의 목적코드 입니다.



AbsoluteLoader를 통해 3번 장치에 장 착되어 있는 Starter의 Tape값을 2000 번지로 옮겨 적어(Loading) 그 위치로 Jump하는 과정을 진행합니다.

이를 mkboot하여 새로운 BOOT파일을 생성하여 작동시켰고, 이로써 새로 생성된 STARTER가 정상 작동함을 확인하였습 니다.

- * myTest.bl 은 AbsloluteLoader에 해당합니다.
- * myStarter 는 Starter에 해당합니다.

위 기존의 Starter를 보면, 기본적인 프롬프트 명령어들이 정의가 되어 있습니다. 하지만 82번부터 87까지 VIEW BL/RL/LL 이 구현이 되어있지 않음을 확인 할 수 있습니다. 또한 프롬프트를 통해 입력받은 문자열을 따로 저장하지 않아, 다양한 기능을 구현함에 있어 제한이 있습니다.

앞으로 'Imcc' 확장자로 작성되는 Starter를 통해, STRCMP, STRTOK, STRLEN 명령어를 이용하여 프롬프트에서 입력받은 문자열을 저장할 것 입니다. 그리고 이를 이용하여 VIEW BL/RL/LL 및 LOAD RL 등의 기본적인 프롬프트 명령어를 구현(1-필수)하고 더 나아가 저장된 문자열을 이용하여 기존에 없는, 예를 들어 간단한 도형출력 등의 기능을 추가(2-옵션)하여 볼 것입니다.

```
// MyStarter Program
M PROMPT
               EOU 30
M_QUIT
               EQU 31
M_LOADDEV
               EQU 32
M LOADBL
               EQU 33
                                                해당하는 명령어로 이동하기 위해, 명령어
M LOADRL
               EQU 34
M LOADLL
               EQU 35
                                                에 해당하는 값을 지정해 둡니다.
M LOAD
               EQU 36
M MOVE
               EQU 37
                                                프롬프트에서 입력을 받으면 이 값을 참고
               EQU 38
M RUN
M DUMP CMOS
               EQU 39
                                                하여 해당 명령어로 이동 후 실행합니다.
M DUMP
               EQU 40
M ASM
               EQU 41
               EQU 42
                                                또한 새로운 값을 추가하여 기존에 없는 명
M DASM
               EQU 43
M INTERPRET
                                                령어를 추가 할 수 있습니다. 이때,
M HELP
               EQU 44
                                                'memory/roms/monitor.h'
M CC
               EQU 45
M CAT
               EQU 46
                                                'memory/roms/monitor.c'
M MKBOOT
               EQU 47
                                                파일에도 새로운 값을 추가해야 합니다.
VIEWBL
               EQU 48
VIEWRL
               EQU 49
               EQU 50
VIEWLL
               // MyStart 시작
                                                           프롬프트 출력문을 개인화 할
               START 2000
MYSTARTER
               LD SP #STACK BTM
                                                                 수 있습니다.
PROMPT
                    A #BUF // BUFFER 주소
               LD
                    X #=C'U20103390' // PROMPT 문자열 주소.
               LD
               INT M PROMPT
                    X A // Return value X에 저장
               _{
m LD}
               SUB X #31
                    C X
               LD
               ADD X C
               ADD X C // 3X, 3BOX Indexing
                    A #BUF
               LD
               PUSH A
               CALL NEAR @MONITOR TABLE
               ADD SP #2
               JUMP NEAR PROMPT
MONITOR_TABLE
               INT M_QUIT
                                      QUIT를 입력받기 전까지는
               RET
                                      STARTER 프로그램을 반복
               NOP
               INT M_LOADDEV
                                     하여 프롬프트를 실행 합니다.
               RET
               NOP
               INT M_LOADBL
               RET
               NOP
               INT M LOADRL
               RET
               NOP
               INT M_LOADLL
               RET
               NOP
               INT M_LOAD
               RET
               NOP
               INT M_MOVE
               RET
               NOP
               INT M_RUN
               RET
               NOP
```

```
INT M_DUMP_CMOS
               RET
               NOP
               INT M_DUMP
               RET
               NOP
               INT M ASM
               RET
               NOP
               INT M_DASM
               RET
               NOP
               INT M_INTERPRET
                                                 해당하는 명령어로 점프하여 실행하게
               RET
                                                 됩니다. INT 를 통해 이미 정의되어 있
               NOP
               INT M_HELP
                                                    는 명령어를 실행하게 됩니다.
               RET
               NOP
               INT M CC
                                                새로 추가되는 명령어는 Starter의 아
               RET
                                                 래부분에 작성되며, 이를 실행하기 위
               NOP
                                                 해 해당하는 명령어가 작성된 주소로
               INT M_CAT
               RET
                                                   JUMP 하여 실행하게 됩니다.
               NOP
               INT M_MKBOOT
               RET
               NOP
               JUMP NEAR VIEWBL /* VIEW BL 아래 의사코드 구현 */
               NOP
                          // VIEW RL이어야 하지만 구현이 안되어있음.
               INT M HELP
               RET
               NOP
               INT M HELP
                          // VIEW LL이어야 하지만 구현이 안되어있음.
               RET
               NOP
VIEWBL
               #pragma C {
               // 입력이 "VIEW BL 50" 일경우
               // BUF에도 C'VIEW BL 50' 이 저장되어 있다.
               char* cmd = strtok(BUF); // "VIEW"
                                                          추가하게 될 VIEWBL이
                                      // "BL"
               char* arg1 = strtok(BUF);
                                                          의사코드로 작성되어 있습
               char* arg2 = strtok(BUF); // "50";
               int ioaddr = asm
                                                          니다. 이와 같은 식으로 구
                 {
                                                          현되지 않거나 혹은 새로운
                  LD.I.CW A arg2
                                                          명령어를 추가하게 될 것
                  ST A ioaddr
                 } // atoi(arg2);
                                                                 입니다.
               while( feof(ioaddr) ) // TIO 50
                                   // SKZ3 ...
                  ch = fgetc(ioaddr); // IN ioaddr
                  if( ch == 'H' ) {
                  } else ... {
                  }
                 }
               } // end #pragma C
               RET
               LTORG
BUF
               RESBOX 100
STACK
               RESDBOX 200
                                  프롬프트에서 입력받은 문자
STACK BTM
               EQU
                      $
                                    열이 저장되어 있습니다.
               END
```