LMC				Comment	Extended LMC							
Label	Addr NO.	code	mnemonic		Addr.NO.			format	code			mnemonic
main	**0**	510	IN 10	n까지 더할값에 일정하게 뺄 수 1을 입력합니다.	**0**	1	2	OORMFA	**1300**	**0100**	**9200**	**LD_A 9200 0100**
	1	252	STA Val_3	입력받은 숫자는 52번 레지스터에 저장됩니다.	**3**	4	5	OORMFA	2200	0056	0000	ST A Val_3
	2	510	IN 10	1~n까지 더하기 위한 n값을 입력받습니다.	**6**	7	8	OORMFA	**1300**	**0100**	**9200**	**LD_A 9200 0100**
	3	251	STA Val_2	n값을 51번 레지스터에 저장합니다.	**9**	**10**	**11**	OORMFA	2200	0053	0000	ST A Val_2
BgnLoop	**4**	452	SUBA Val_3	A.reg 에 저장되어 있는 숫자에 52번지에 저장된 1을 뺌	**12**	**13**	**14**	OORMFA	4300	0056	0000	SUB A Val_3
	5	250	STA Val_1	뺀 값을 50번지에 저장	**15**	**16**	**17**	OORMFA	2200	0050	0000	ST A Val_1
	6	802	SKIP 2	조건문: 음수이면 연산종료, 다음문장을 넘어갑니다.	**18**	**19**	**20**	OOOR	**9822**	**0000**	**0000**	**SKP3 2**
	7	912	JMP_OutLoop	12번 문장으로 이동하면서 Loop 탈출	**21**	**22**	**23**	OORMFA	7300	0036	0000	**JMP_OutLoop**
	8	351	ADDA Val_2	51번지 값과 A.reg값을 더한다.	**24**	**25**	**26**	OORMFA	3300	0053	0000	ADD A Val_2
	9	251	STA Val_2	51번지에 더한 값을 저장한다.	**27**	**28**	**29**	OORMFA	2200	0053	0000	ST A Val_2
	10	150	LDA Val_1	50번지 값을 불러온다.	**30**	**31**	**32**	OORMFA	1300	0050	0000	LD A Val_1
	11	904	JMP_BgnLoop	4번지로 이동.	**33**	**34**	**35**	OORMFA	**7300**	0012	0000	**JMP_BgnLoop**
OutLoop	**12**	151	LDA Val_2	51번지 값을 불러온다.	**36**	**37**	**38**	OORMFA	1300	0053	0000	LD A Val_2
	13	612	OUT 12	12번 바구니(모니터)에 A.reg 값을 출력	**39**	**40**	**41**	OORMFA	**2200**	**0120**	**9200**	**ST_A 9200 0120**
	14	700	COE 0	홀트	**42**	**43**	**44**	OOAA	0700	0000	0000	COE 0
	•	•	•	•	•	•	•	•	•			•
	•	•	•	•	•	•	•	•	•			•
	•	•	•	•	•	•	•	•	•			•
	•	•	•	•	•	•	•	•	•			•
Val_1	**50**	"int valForSumToRslt;"		결과에 더해질 각 요소들을 임시로 저장하는 변수입니다.	**50**	**51**	52	"long int valForSumToRslt;"				// 50->50
Val_2	**51**	"int valToPrntRslt;"		결과를 도출해내어 마지막에 출력하기 위한 변수입니다.	**53**	**54**	55	"long int valToPrntRslt;"				// 51->53
Val_3	**52**	"int valToAbst = 1;"		n까지 수를 더하기 위해 더할값에 1씩 빼기위한 변수입니다	**56**	**57**	58	"long int valToAbst = 1;"				// 52->56
			1.	주석에서 언급한 long int는 그냥 긴 정수라는 의미의 주석이며, 자료형이랑은 아무런 상관이 없습니다.								

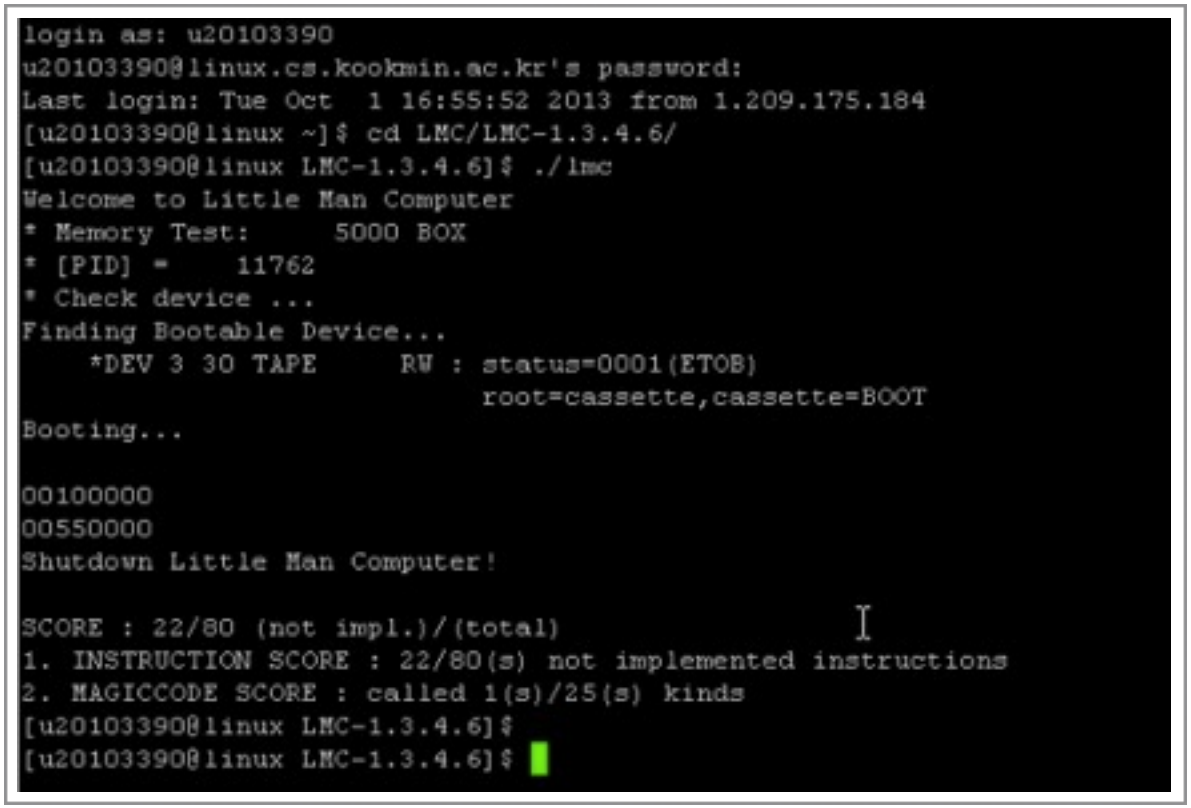
Optimized Extended LMC									Comment
Label	Addr. NO.			format	code			mnemonic	
#	0	1	2	#	0001	0000	0000	NOP	계산을 위해 일정하게 뺄 수를 코드에 기입해 놓습니다.
main	3	4	5	OORMFA	1330	0001	0000	LD X 1	X.Reg 를 0으로 초기화
	6	7	8	OORMFA	1300	0100	9200	LD_A 9200 0100	A.Reg 에 n입력
BgnLoop	9	10	11	OORR	3230	0000	0000	ADD X A	X.Reg에 A.reg 값 더해 넣기
	12	13	14	OORMFA	4300	0000	0000	SUB A 0	A.Reg-1
	15	16	17	OOAA	9821	0000	0000	SKP3 1	조건문 : 0이 되면 연산종료, 다음문장을 넘어갑니다.
	18	19	20	OORMD	7300	0009	0000	JMP BgnLoop	12번 주소로 이동
	21	22	23	OORMFA	2230	0120	9200	ST_X 9200 0120	X.Reg 출력.
	24	25	26	OOAA	700	0000	0000	COE 0	홀트
	.	.	.	.	.	.	.	.	.
		1. 0, 1 번지는 DBOX로 각각 1과 0의 값을 가집니다. OP.Code가 0이기에 실행에 문제는 없었습니다.							

2. LMC 머신에서 해당 프로그램을 실행하는 과정은 어떻게 되는가?

=> LMC 머신을 작동시키면 우선 가상의 CMOS가 시작되면서 미리 정해진 순서에 따라 시작**(Booting)**하게 됩니다.
우선 Cassette에 저장된 정해진 위치에 있는 정해진 범위만큼의 자료를 LMC 머신의 Memory의 정해진 위치로 이동**(Loading)**시킵니다.
그 후 CPU는 Memory에 저장된 자료를 정해진 순서에 따라 실행**(Fetch & Execution)**하게 됩니다.

3. 상수 1과 변수 n을 메일 박스의 어디에 배치할 것인가?

=> **상수**는 변경되어선 안되는 값이기에 코드가 확장되어도 상관 없도록 실행코드 앞에 위치하는 것이 좋습니다. 최적화한 확장된 LMC프로그램에서 이를 구현 해 보았습니다. **변수**는 프로그램 진행 중에 필요에 따라 생기는 값이기에 코드 뒤에 두되, 대신 프로그램을 실행하는 코드에 영향을 주지 못하도록 가능하면 뒤에 두는 것이 좋습니다. 확장하기 전의 작은 LMC에서 이를 구현해 보았습니다.


login as: u20103390
u20103390@linux.cs.kookmin.ac.kr's password:
Last login: Tue Oct 1 16:55:52 2013 from 1.209.175.184
[u20103390@linux ~]$ cd LMC/LMC-1.3.4.6/
[u20103390@linux LMC-1.3.4.6]$ ./lmc
Welcome to Little Man Computer
* Memory Test: 5000 BOX
* [PID] = 11762
* Check device ...
Finding Bootable Device...
*DEV 3 30 TAPE RW : status=0001(ETOB)
root=cassette,cassette=BOOT
Booting...
00100000
00550000
Shutdown Little Man Computer!
SCORE : 22/80 (not impl.)/(total)
1. INSTRUCTION SCORE : 22/80(s) not implemented instructions
2. MAGICCODE SCORE : called 1(s)/25(s) kinds
[u20103390@linux LMC-1.3.4.6]$
[u20103390@linux LMC-1.3.4.6]$