David Giacobbi
CPSC 260: Assignment #2 LMC

1.)

	INP	00	901	← numeric LMC program
	STA x	01	316	
	INP	02	901	Example (x=3, y=2):
	STA y	03	317	acc: 3, 2, -1
	SUB x	04	216	x: 3
	BRP reverse	05	811	y: 2
	LDA x	06	516	neg-flag: 0, 1
	OUT	07	902	
	LDA y	08	517	output: 3, 2
	OUT	09	902	
	BRA done	10	615	Example (x=2, y=3):
reverse	LDA y	11	517	acc: 2, 3, 1
	OUT	12	902	x: 2
	LDA x	13	516	y: 3
	OUT	14	902	neg-flag: 0, 0
done	HLT	15	000	
x	DAT 000	16	000	output: 3, 2
y	DAT 000	17	000	* Output nums in descending orde

2.)

	INP	00	901	→ numeric LMC program
	STA op1	01	316	
	INP	02	901	Example (op1 = 5, op2 = 3)
	STA op2	03	317	acc: 5, 3, 3, 2, 0, 5, 2, 1, 5,
loop	LDA op2	04	517	op1: 5 | 10, 1, 0, 10, 15, 0, 15
	BRZ done	05	712	op2: 3, 2, 1, 0
	SUB one	06	215	sum: 0, 5, 10, 15
	STA op2	07	317	zero-flag: 0, 0, 1
	LDA SUM	08	518	
	ADD op1	09	116	output: 15
	STA sum	10	318	* multiply op1 • op2 in return product
	BRA loop	11	604	
done	LDA sum	12	518	

→ see back for #2

	OUT	13	902	→ numeric LMC program (cont.)
	HLT	14	000	
one	DAT 001	15	001	
op1	DAT 000	16	000	
op2	DAT 000	17	000	
sum	DAT 000	18	000	

3.)

	INP	00	901	→ numeric LMC program
	BRZ done	01	714	
	STA n	02	317	Example (n=3, r=2, i=1):
	INP	03	901	i: 1, 2, 3, 4 r+i
	STA r	04	318	n: 3
loop	INP	05	901	r: 2
	ADD r	06	118	acc: 3, 2, 1, 3, 1, 2, 3,
	OUT	07	902	1, 4, 6, 2, 3, 3, 0, 3, 5, 34,
	LDA i	08	516	neg-flag: 3, -1
	ADD one	09	115	zero-flag: 0 1, 4, 3
	STA i	10	316	output: 3, 6, 5
	LDA n	11	517	
	SUB i	12	216	if (i ≤ n)
	BRP loop	13	805	print (r + INP)
done	HLT	14	000	
one	DAT 001	15	001	for (int i=1; i ≤ n; i++)
i	DAT 001	16	001	print (input + r)
n	DAT 000	17	000	
r	DAT 000	18	000	

4.) The first program prints the two user inputs in descending order.

* See C++ code on next page →

```
int x, y;
cout << "Enter an integer:";
cin >> x;
cout << "Enter an integer:";
cin >> y;
if (y > x)
   cout << y << ", " << x;
else
   cout << x << ", " << y;
```

5.) The second program takes in two user integers and outputs the calculated product between them.

```
int op1, op2;
Cout << "Enter an integer: ";
cin >> op1;
cout << "Enter an integer: ";
cin >> op2;
cout << op1 * op2;
```

6.) The third program regulates a for loop, requesting for n number of loops and r constant, printing user input + r and outputting each loop.

```
int n, r;
cout << "Enter an integer for n:";
cin >> n;
cout << "Enter an integer for r:";
cin >> r;
for (int i = 1; i <= n; i++){
   cout << "Enter an integer to add to r:";
   int input;
   cin >> input;
   cout << input + r;
}
```

7.)

PC	mnemonic	opcode	description
00	INP	901	collect n input
01	BRZ done	713	check n=0 case
02	SUB one	214	subtract 1 from n
03	STA n	315	store updated n in mailbox
04	loop INP	901	collect first num to add
05	ADD sum	116	add to current sum
06	STA sum	316	store updated sum
07	LDA n	515	load n into acc.
08	SUB one	214	subtract 1 from n
09	STA n	315	store updated n
10	BRP loop	804	check if loop complete
11	LDA sum	516	load sum in acc.
12	OUT	902	output sum
13	done HLT	000	end program
14	one DAT 001	001	one value
15	n DAT 000	000	n value
16	sum DAT 000	000	sum value

Example (n = 3, 1+2+3 = 6)

n = 2, 1, 0, -1

sum = 0, 1, 3, 6

acc. = 3, 2, 1, 1, 2, 1, 2, 3, 1, 0, 3, 6, 0, -1, 6

reg-flag = 0, 0, 1

output 6

8.)

PC	mnemonic	opcode	description
01	INP	901	dividend input
02	BRZ quit	717	if 0, exit
03	STA a	318	store dividend
04	INP	901	divisor input
05	BRZ quit	717	if 0, exit
06	STA b	319	store divisor
07	loop LDA quo	521	load quotient
08	ADD one	120	add 1 to quo
09	STA quo	321	store new quo
10	LDA a	518	load dividerd
11	SUB b	219	subtract divisor from dividend
12	STA a	318	store new a val
13	BRP loop	807	check if curr a negative
14	done LDA quo	521	load quotient
15	SUB one	220	sub 1 from off-set quo
16	OUT	902	output quotient
17	quit HLT	000	exit program
18	a DAT 000	000	data store dividend
19	b DAT 000	000	data store divisor
20	one DAT 001	001	data store one
21	quo DAT 000	000	data store quotient

Example: (14 // 4 = 3)

a: 14, 10, 6, 2

b: 4

quo: 0, 1, 2, 3, 4

acc: 14, 4, 0, 1, 14, 10, 1, 2, 10, 6, 2, 3, 6, 2, 3, 4, 2, -2, 4, 3

neg-flag: 0, 0, 0, 1

output: 3